Master thesis
Artificial Intelligence: Intelligent Technologies

Radboud University

And

ASMPT

Efficient Test-Time Data Augmentation Pipelines and Ensemble Selection Using Evolution

Author:
Nadezhda Dobreva
s1033115

First supervisor/assessor:
Prof. Dr. Marcel van Gerven
marcel.vangerven@donders.ru.nl

Daily supervisor:
Sigur de Vries
sigur.devries@donders.ru.nl

Second assessor:
Mahyar Shahsavari
mahyar.shahsavari@donders.ru.nl

April 30, 2025

Abstract

Model generalizability in the face of data shift is crucial, but often, when encountered with out-of-distribution data, the performance suffers significantly. In many real-world applications this can lead to great risks or costs, and improving the model's predictive capabilities on shifted data efficiently is desired. Ensembling, i.e., combining the predictions of multiple models, is one approach to tackle the issue. Another is augmenting the data with predefined transformations with the goal of bringing it closer to the data used at train-time. In this thesis, we explore the combination of ensembling and data augmentation at test-time as a possible means to improve generalizability without retraining models. In particular, we use evolutionary algorithms for optimal sparse ensemble selection and integration as well as for optimizing a pipeline of transformations to apply on the data in an attempt to reverse the data shift. We propose three different ways of combining ensembling and data augmentation pipelines at test-time, and evaluate them thoroughly on problems of classification and segmentation with data shift and on data from a real-world use case.

The results indicate that the combination of ensembling and data augmentation is promising, consistently outperforming using a single model or averaging model library predictions. Furthermore, our proposed methods require less samples than retraining and are less costly in terms of time. Overall, our algorithms can efficiently improve the performance on out-of-distribution data and deserve further investigation on the way towards improving model generalizability.

Contents

1 Introduction **3**

2 Background **5**
2.1 Evolutionary Algorithms 5
2.2 Data Shift 7
2.3 Model Ensembling 8
2.4 Semantic Segmentation for Defect Detection at ASMPT 9

3 Related Work **11**
3.1 Ensembling 11
3.2 Data Augmentation Pipelines 12
3.3 Current Practices in Tackling Data Shift 13

4 Methodology **14**
4.1 Evolutionary Algorithm for Ensemble Selection – ENSEC 14
4.2 Evolutionary Algorithm for Data Augmentation Pipelines – DAP 15
4.3 Combining Ensemble Selection and Data Augmentations 17
4.3.1 ENSEC Followed by DAP 19
4.3.2 DAP Followed by ENSEC 19
4.4 Data 20
4.5 Evaluation Metrics 22

5 Experimental Setup **23**
5.1 Experiments 23
5.2 Models and Performance Baselines 25
5.3 System Specifications 28

6 Results **29**
6.1 EAs for Ensemble Selection: ENSEC 29
6.2 EAs for DA Pipeline Discovery: DAP 30
6.3 Combining ENSEC and DAP 31
6.4 Visualizing ENSEC and DAP Results 32
6.5 Computation Time 33
6.5.1 Time Cost of EAs 33
6.5.2 Inference Time 35

7 Discussion **37**

8 Conclusion **42**

Chapter 1

Introduction

Neural networks (NNs) find applications more and more often in the real world for a broad range of tasks: from localization and object detection for autonomous driving [71], to cell membrane and nucleus segmentation and classification [60], to circuit board welding defect detection [34]. The capability to generalize is therefore crucial for a reliable and safe to use model. A neural network is said to generalize well if there is a consistency between training and testing errors, showing similar performance between the two [3]. This usually concerns robustness against unseen cases similar to the data the model was trained on, but can extend to expecting it to perform well on related but different domains [55, 59]. If a model does not generalize well, it tends to exhibit a drop in performance when applied to new instances in comparison to its performance on train data, which is undesirable especially in real-world applications that could endanger human lives.

One commonly encountered cause for reduced generalizability is the difference between the data used at train time and that encountered in the real world, i.e., data shift [57]. To tackle this problem, models are often retrained and finetuned [62, 69], or trained with the idea of robustness in mind from the very beginning [65]. However, training is a time- and compute-expensive process, and a method that can circumvent training while improving generalizability at test-time would be desirable at times. Two possible such methods are model ensembles and test-time data augmentation. An ensemble of models is composed of a number of independent base learners, whose predictions are in some form aggregated together into one. The aim of ensembles is to combine the strengths of multiple models that likely have learned to process data differently and can use this diversity to correct wrong and overconfident predictions [1]. On the other hand, test-time data augmentation refers to transforming the test data in a number of ways before feeding it to a fully-trained model and pooling the predictions over all to obtain a "smoothed" and more robust prediction [61]. In our work a related but novel approach is utilized, test-time augmentation adaptation, where a sequence of transformations is identified that makes the test data more closely related to the data used during training. Both identifying the optimal ensemble from a library of models, and finding the optimal augmentations to be applied to the input require an optimization method. However, there are no gradients in such problems so standard backpropagation ways of optimizing cannot be used.

One possible gradient-free optimization technique are evolutionary algorithms (EA), an op-

timization algorithm relying on mechanisms inspired by biological evolution. It creates a population of solutions, each of which is evaluated based on how close it is to achieving a certain goal. The closest and thus best solutions are selected for the generation of new solutions by combining their features and possibly introducing new ones. The mechanism of natural selection drives the creation of newer generations of solutions, until a best one, as defined by some fitness criteria, is found. The flexibility to choose and customize the fitness function is one advantage of EAs. Another one is the fact that it is gradient-free: consequently, the search is a lot more global and less prone to getting stuck in a local optimum, and problems with discontinuous or noisy target functions can still be tackled.

Evolutionary ensemble selection has already been investigated with great success [10], while test-time augmentation adaptation is a novel method. This thesis investigates how well they can be combined to create a fully training-less approach for improving generalizability efficiently. Thusly, we aim to answer *How effective is combining ensemble selection with data augmentation pipeline selection via evolution for mitigating the problem of data shift at test-time?* Our goal is to optimize not only predictive performance, but also make the test-time adaptation as efficient as possible, i.e., remove the need for re-training models, achieve sparse ensembles, and utilize as few resources as possible. Thus, the main contributions of this thesis are as follows:

- Implementing an EA for sparse ensemble selection with the goal of increased generalizability.
- Implementing a novel EA for data augmentation pipeline selection for reverting data shifts at test-time.
- Implementing novel pipelines of EAs combining ensemble selection and data augmentation techniques for efficient test-time adaptation.
- Evaluating the proposed approaches on a set of datasets to asses performance, both for classification and segmentation. This includes benchmark data as well as data from a real-world case of data shift to be tackled.
- Showcasing the benefits of our proposed approaches by analyzing their computational and time resources, and comparing their predictive performance to current practices in increasing generalization to out-of-distribution data.

In the next sections we provide background information (Chapter 2) and review relevant literature (Chapter 3), elaborate our proposed approaches and used data (Chapter 4), present the experimental setup and results, and discuss their implications (Chapters 5, 6 and 7), and finally summarize our conclusions (Chapter 8).

Chapter 2

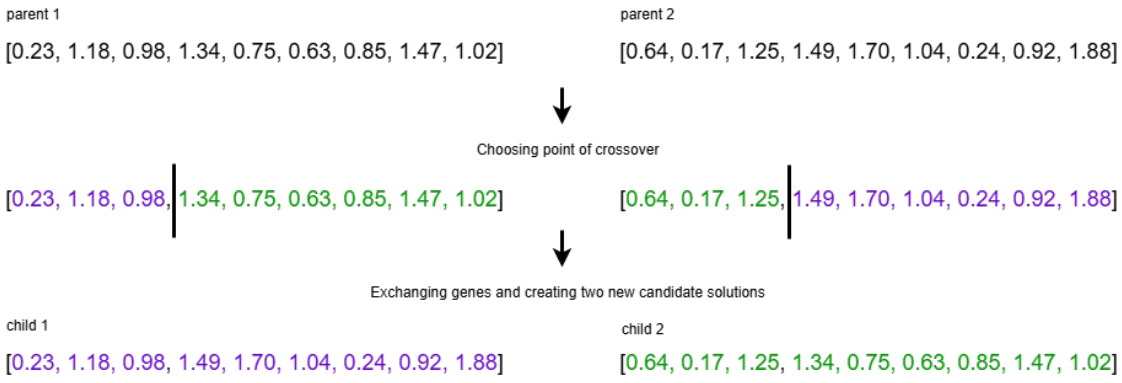
Background

2.1 Evolutionary Algorithms

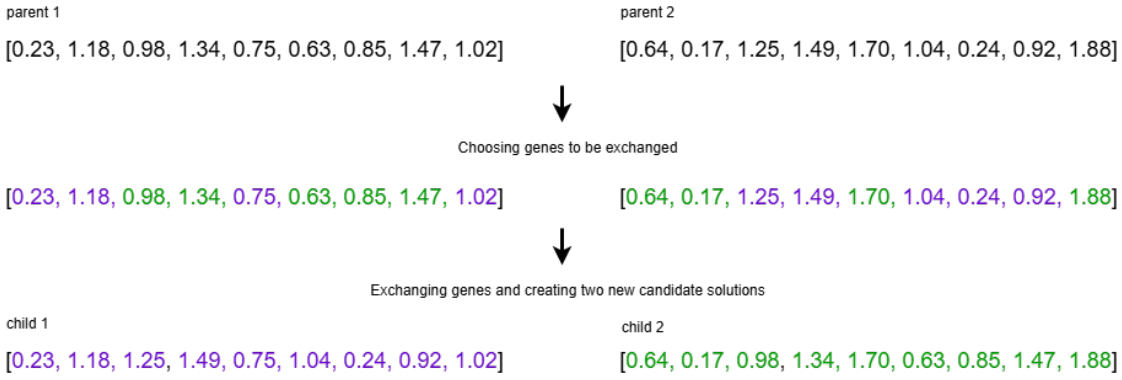
An evolutionary algorithm (EA) is a population-based method for solving optimization problems that uses mechanisms inspired by biological evolution, such as reproduction, mutation, and selection. They often perform well in finding an approximate solution to a problem because they do not make any assumptions about the underlying objective function landscape. In other words, EAs can work well even when the problem has multiple optimal solutions, is noisy, discontinuous, etc., as they simply evaluate solutions and apply evolutionary steps.

A genetic algorithm (GA) is one instance of the broader class of EAs, based on the principle of evolution and natural selection [63]. It generates an initial population of candidate solutions to a problem (i.e., individuals) and evolves them towards better solutions. The individuals have a set of characteristics (or genes) which the GA alters in the direction of improvement as they are passed from one generation to the next. A fitness function, i.e., an objective function summarizing in a single number how close a solution is to the set goal, gets used to evaluate the candidate solutions. The fittest individuals according to it are selected for passing on their characteristics to the next generation. Reproduction is a key operation for the creation of the new population. Usually, two individuals are used for reproducing an offspring. One way to achieve that is one-point crossover, where a random point along the parents' genes is chosen and the child inherits all genes up until that point from one parent and all after it from the other. Another type is uniform crossover, in which every gene can be inherited from each parent with equal probability. The two crossover methods are visualized in Figure 2.1. Both start off global and become more local generation after generation as the population tends to converge. However, uniform crossover has been shown to be much less biased as it allows any gene of the parent to be transferred to the offspring with the same probability at any stage of the algorithm [53].

When it comes to creating the new population, a frequently used strategy is elitism. It allows the best individuals from a generation to pass on to the next one without modification in order to preserve their genes and increase the convergence speed [4]. Culling, on the other hand, removes the most inferior solutions from the population to create space for better individuals. Mutation is another crucial characteristic of evolutionary algorithms, which involves a predefined chance that an individual's genes will undergo a random change. This strategy allows



(a) One point crossover.



(b) Uniform crossover.

Figure 2.1: Examples of crossover operations, for a GA optimizing a vector of floating point values.

for completely random parameter values to be introduced. It also ensures genetic diversity and that the GA does not get stuck in a local minimum or maximum (i.e., finding a solution that is fitter than the ones nearby, but possibly worse than solutions at a greater distance in the search space). An example of the mutation operator is given in Figure 2.2.

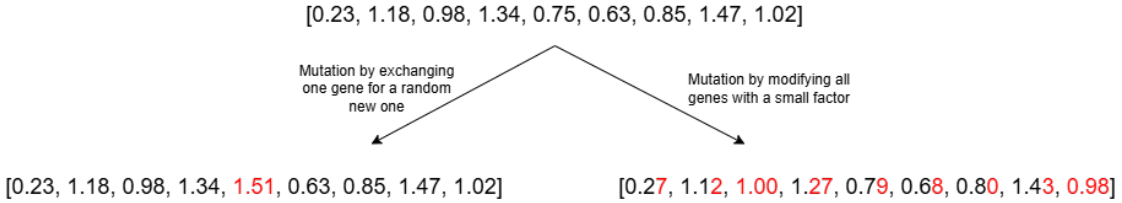


Figure 2.2: Example of possible mutations for a vector of floating point values.

The general structure of EAs is summarized in the pseudocode in Algorithm 1.

Algorithm 1 EA

Input: number of generations n_gen and population size N.

$generation \leftarrow 0$
$population \leftarrow$ initialize_population(N)

while $generation \neq n\_gen$ **do**
  $pop\_fitnesses \leftarrow$ evaluate($population$)
  $new\_population \leftarrow \emptyset$
  while len($new\_population$) $\neq N$ **do**
    $new\_candidate \leftarrow$ reproduce($population$)
    $new\_candidate \leftarrow$ mutate($new\_candidate$)
    $new\_population \leftarrow new\_population + new\_candidate$
  end while
  $population \leftarrow new\_population$
  $generation \leftarrow generation + 1$
end while

$best\_candidate \leftarrow$ best_fitness($population$)
return $best\_candidate$

2.2 Data Shift

Dataset shift is a common challenge for machine learning predictive models, which occurs when the joint distribution of inputs and outputs differs between the training and test stages [57]. The deviation can have different origins, such as changes in style, shape, size, lighting conditions or various forms of corruption, and is present in most practical applications. This often leads to a performance drop of the model during inference, when confronted with data from a distribution unlike the one used at train time. One way to mitigate this is retraining or fine-tuning the network using a sufficient amount of this out-of-distribution data [73]. An alternative approach is applying data augmentation techniques during the training process in order to expose the network to "shifted" data and help it generalize better against unseen samples [3]. For digital image datasets, this consists of image manipulations such as resizing the image, cropping it, and adjusting image brightness, contrast, gamma and sharpness. These are visualized in Figure 2.3.

Test-time adaptation (TTA) is another technique offering promising solutions to the data shift problem [68]. Work into TTA can be split into two main categories [47]: Test-Time Training (TTT) which adapts the model by, for instance, optimizing an unsupervised loss during training, and Fully TTA which does not alter the training process and can adapt already trained models to the test data without access to the original training data. Fully TTA approaches include, e.g., batchnorm statistics adaptation and test-time entropy minimization [46]. Our work also follows the Fully TTA setting.

(a) Original image.

(b) Brightness adjusted with factor of -0.5.

(c) Brightness adjusted with factor of 0.5.

(d) Center-cropped.

(e) Contrast adjusted with factor of 0.5.

(f) Contrast adjusted with factor of 1.5.

(g) Resized.

(h) Gamma adjusted with factor of 0.5.

(i) Gamma adjusted with factor of 1.5.

Figure 2.3: Examples of image augmentations.

2.3 Model Ensembling

An ensemble of models is a set of trained models (i.e., base learners) whose individual outputs are combined in some way to produce a decision for new data samples [18]. Ensembles have repeatedly demonstrated the capacity to improve upon the performance of a single model for classification [8, 18, 22], regression [30, 42], segmentation [28, 44] and more tasks. There are

three main reasons combining the votes of multiple models increases performance. Firstly, multiple base learners may bring better performance than a single strong one, since in theory no one model will have the best predictive performance for all problems, and in practice, selecting the best learner for any given dataset is a very difficult problem [10]. Secondly, ensembling multiple predictions reduces bias and variance [30]. Lastly, an ensemble reduces the risk of getting stuck in a local optima since the local search is performed via different starting points, leading to a better approximation of the truth [18]. In simpler terms, training multiple networks, each with a different random initial parametrization, means there is a higher chance that one of them gets close to the optimum in the loss function landscape, and having one or more models with strong predictive capabilities within the ensemble leads to improved performance. As a consequence of these ensembling characteristics the resulting ensembles can offer increased robustness to dataset shift: an ensemble of possibly diverse predictions prevents the model from relying on one model's predictions which may be incorrect but also overconfident [72].

Diversity among the models plays a key role: for an ensemble to perform better than its individual members, the base learners must be diverse in prediction and make different errors on the same inputs [7, 18]. If the individual models have highly correlated outputs the benefit of combining them is lost. Some work argues there is little correlation between diversity and accuracy [9, 51], but the majority of reviewed literature claims diversity helps achieve higher performance and presents empirical results supporting that [12, 23, 41].

Ensemble learning is composed of three main steps: generation, selection and integration [10]. In the first stage, a pool of both accurate and diverse base learners is obtained. Next, a selection is made of the final set of models that will make up the ensemble, such that performance and diversity is optimized. Selectively choosing the optimal subset from a pool of pretrained models aims to retain models that are most complementary to each other, and both theoretical analyses and empirical studies support the effectiveness of this step in boosting the generalization ability of ensembles [56]. Finally, a decision must be made on the method of aggregation of the base models' predictions; They are typically combined by weighted or unweighted voting [13], such as majority vote where, e.g. in classification, the predicted class is the one most models have predicted and the minority class predictions are ignored. In weighted voting, the overall ensemble prediction is a sum of the independent models' predictions but each model's vote is given a certain amount of strength because it is multiplied with a weight. In the simplest case all predictions are averaged out, i.e., the voting weight of each model in the ensemble is $\frac{1}{M}$ if there are M models.

2.4 Semantic Segmentation for Defect Detection at ASMPT

In computer vision, segmentation is the process of partitioning a digital image into several regions with the aim to, e.g., find and segment objects. In semantic segmentation in particular, every pixel of the image is assigned a category label [26]. Given an image, a semantic segmentation algorithm should output which pixels of the image belong together semantically, but that often includes assigning the same label to pixels of largely differing colors. This makes the task one of the most challenging ones in computer vision, with the mechanic in the human brain that achieves this still unknown [25]. It is, however, also a task with a myriad of real-world applications, one of which chip defect detection.

ASMPT is a company that specializes in equipment and solutions for semi-conductor and electronics assembly industries. Image segmentation is an important part of the offered services by ASMPT, with the aim to detect potential defects. For instance, if a manufactured chip has a broken connection in the wiring, the semantic segmentation models developed at ASMPT would be able to flag it as faulty.

A single high-performing segmentation model is used at inference time, but when applied to devices of new customers, it is often retrained. This is due to the significant variations in the visual appearance of the chips produced by different customers, i.e., data shift. However, the process of retraining, re-evaluating and providing the customer with a model introduces delays that ASMPT would like to minimize. In an ideal scenario, they should be able to quickly adapt the segmentation models to improve predictions on the test-time data. Our work aims to offer a potential solution by generating different data augmentation pipelines and selecting a sparse ensemble with increased generalizability for different clients easily at test-time.

Chapter 3

Related Work

3.1 Ensembling

Ensemble learning can include ensemble generation, selection and integration, but given the focus of this thesis, only the latter two are reviewed. The majority of relevant work on ensembles focuses on classification tasks where ensembling has been shown to lead to great boosts in generalizability to out-of-distribution data: around 2-5% increase in accuracy over using a single model [50, 56, 58]. Creating an ensemble of segmentation models has also been done for a number of tasks and data domains, such as medical images [17, 23, 38], aerial scenes [45] and black board pictures [48]. Unfortunately, the focus is rarely on tackling data shift in segmentation, but ensembles consistently outperform the best-performing base learner in the ensemble by 1-2% when it comes to data from the same distribution. Similarly to some of the reviewed literature, we aim to find an optimal subset of the model library instead of using all available models. This is mainly with the aim of achieving efficiency, one aspect of which is using as few models as possible, from a large model library, without the predictive performance suffering.

One method to select suitable models is cluster-based: a clustering algorithm groups together models that make similar predictions and then every cluster is pruned with the goal of removing redundant models and increasing the overall ensemble diversity [54]. Greedy ensemble selection, on the other hand, attempts to find the globally best ensemble. An example is starting with the empty set, then iteratively adding to the set a model from the library based on its added value to the already existing ensemble [11]. When the ensemble size is large, selecting base learners can be computationally expensive if all ensemble options are considered, making evolutionary algorithms an attractive meta-heuristic for ensemble selection [10]. Usually a genetic algorithm is implemented to identify a suitable ensemble membership, with the individuals encoded as binary strings where 1 denotes the presence of a model in the ensemble and 0 denotes its absence [14, 70]. Alternatively, one can use a vector of real-valued numbers $\in [0, 1]$, and search for an optimal weighting of the contribution of the separate models' votes [13, 74]. Some algorithms attempt to construct sparser ensembles for higher efficiency, by, e.g., not considering the votes of models with a weight below a certain threshold [74] or pruning models based on a metric, such as the reduction of errors [40].

When it comes to fitness measures for ensemble selection, effectiveness is most popular among

design of fitness functions, followed by diversity metrics [10]. Ensemble effectiveness is usually well-defined, with, for instance, accuracy being used for classifier models, mean squared error – for regressor models, intersection over union (IoU) – for segmentation models. Diversity has no single well-defined metric, but this is also out of scope for this thesis.

Ensemble integration can also be achieved in a number of ways. For example, the predictions of the models can be combined by a meta-model which learns non-linear integration schemes [6]. EAs can once again be used for optimizing the voting weights of a weighted majority voting rule, or for selecting optimal techniques of combining the outputs of base learners [10].

3.2 Data Augmentation Pipelines

A data augmentation (DA) pipeline consists of a sequence of transformations to be applied on data samples with the aim of improving predictive performance of the models working with that data. One of the most successful works on automatizing the discovery of optimal DA pipelines is the AutoAugment procedure [16]. With AutoAugment, different image processing operations ordered in various ways represent policies, and a search algorithm is deployed to find the best policy such that the neural network yields highest performance on a target task after training using that policy. A Reinforcement Learning (RL) algorithm is used to predict a policy which augments the train data of a model. After being trained, the model's generalization performance is evaluated on a withheld test set as a proxy measure of the policy goodness, which is then used to train the RL controller. The resulting final policy can be used for train-time augmentation towards a robust network.

Evolutionary algorithms have also found application in the search of optimal DA techniques for various use cases. For instance, for deep face detection, in order to potentially enrich the training data and improve performance on unseen data, a genetic algorithm was used to find new face instances with the genotype consisting of numbers encoding different facial parts [15]. Another work reduces overfitting in image classification models by including DA in the training procedure, and a genetic algorithm is used to obtain the optimal augmentation pipeline [52]. In the proposed approach, a gene is a number that encodes a transformation function (e.g., flipping, scaling) and an array corresponding to the parameters used by the function. Mutation occurs by adding a transformation, removing one, or modifying its corresponding parameter. This approach to DA is similar to the one utilized in this thesis, with three key differences: (i) the obtained augmented data is used for the training of models, while we operate strictly at test-time, (ii) this algorithm optimizes the data augmentation pipeline alone, while we aim to optimize it alongside ensemble learning, and (iii) we fix the set of transformations and optimize only their parameters to improve performance.

Data augmentation can also be used as a method for test-time augmentation tackling domain shift. One such work proposes learning an optimal TTA policy for DA to improve the performance of medical image segmentation models on unseen samples [66]. In this approach, multiple "policies" consisting of transformations (e.g., rotation, cropping, resizing) are generated and the parameters of these transformations are optimized using gradient descent. The best policy is kept and fine-tuned before it is used for the generation of augmented views of target images to be fed to the model. Similarly to this thesis, this TTA approach does not require access to the training procedure, nor to the train data. However, our goal is developing a gradient-free solution that reduces time and computation effort. Furthermore,

we aim to change the target data such that it resembles the training data more, instead of augmenting it in different ways for a more robust prediction.

3.3 Current Practices in Tackling Data Shift

Ensemble selection is a reliable method of improving neural network performance on out-of-distribution data [2], given that, as already mentioned, combining multiple models to diversify the decision process can improve the robustness and accuracy of predictions. Furthermore, ensembling has been shown to achieve stronger generalization ability than bagging and boosting [75], and outperform other probabilistic deep learning methods such as Monte Carlo dropout and temperature scaling [49].

Domain adaptation is an alternative approach that addresses data shift by improving the model's ability to already generalize to different data distributions at train time. It aims to learn a model from a source labeled data that generalizes well to a target domain by minimizing the difference between domain distributions [20]. One way to achieve this is via feature alignment, where the network maps the source to the target data by learning a transformation that extracts invariant feature representation across domains. Although this is similar to the proposed data augmentation pipeline that we attempt to learn, our approach is gradient-free and does not require access to the target data during training.

Another technique that is commonly applied in practice with great success when the target labels are available is retraining or fine-tuning the model with the out-of-distribution data. Those are rather expensive operations both in terms of compute power but also because they require a substantial amount of samples to train on. For instance, in a healthcare setting, a model trained on one hospital's data and fine-tuned with data from a second hospital has been shown to rival the performance of a network specifically trained on data from the second hospital only [69]. However, the fine-tuning was conducted with 50% of the train set, meaning it necessitated up to a few thousand samples. Some more sophisticated approaches have been developed that enable test-time training based on only a single test sample yet achieving significant generalizability boosts [64, 73]. However, this method still relies on back-propagation and can be costly for applications where latency is unacceptable. A related approach is continuous training where a network is automatically and continuously retrained to adapt to changes in the data or learn additional tasks [35, 67]. The idea here is to introduce new data to the model, and have it transfer its previous knowledge to it, fine-tune its current task or incrementally learn new ones [31]. However, measures must be taken to avoid catastrophic forgetting, i.e., losing the knowledge already obtained in previous trainings. The simplest way to do that is retraining from scratch or adding the new data to the old train set and continuing training with both in a sort of joint training.

Lastly, under the Fully TTA setting, a number of gradient-free approaches have gathered attention. One of the most successful methods is adjusting the batch normalization layers such that they normalize the features leveraging statistics from the target data batch, instead of statistics from the training phase [43]. Such approaches are popular due their simplicity of integration with any off-the-shelf libraries, and the fact that they do not rely on additional data or changes to the underlying model [61]. In addition, TTA techniques have been found to boost predictive performance of ensembling [5]. However, their application can be unstable in the case of very few test samples, leading to overfitting or even divergence [47].

Chapter 4

Methodology

The purpose of this work is to explore a new method of tackling test-time adaptation in the presence of a data shift: Using evolutionary techniques to both select an optimal ensemble from a library of models, and choose an appropriate data augmentation pipeline that combined can tackle out-of-distribution data without the need for retraining a model's parameters. In the following sections the design of our evolutionary algorithms is presented.

4.1 Evolutionary Algorithm for Ensemble Selection – ENSEC

Given the already proven success of ensembling techniques for dealing with data shift, we opted to utilize ensemble selection. It is performed by an evolutionary algorithm which takes a target task and a library of fully-trained models for a source task and outputs an optimal ensemble for the target task, consisting of a subset of the library. Similarly to the reviewed previous work on ensemble selection, a vector of voting weights is optimized, the length of which depends on the size of the model library. The weights then determine each model's contribution to the ensemble prediction as their independent predictions are weighted before being summed. Thus, a candidate solution is of the form $w = \langle w_1, w_2, ..., w_M \rangle$ for M available models. The initial population of size N can be generated in two ways. In the first one, it is completely random, i.e., the candidate vectors consist of randomly sampled values $\in (0, 1)$. In the second way, the vector is initialized with values sampled from a normal distribution with $\mu = 1$ and $\sigma = 0.2$. This way the population starts close to what we define as our baseline ensemble: where the full library of models is included in the ensemble, and each model's prediction has an equal vote with weight 1, i.e., there is vote averaging.

To generate the subsequent new population, individuals from the current one are selected, combined and adjusted. The parents are randomly chosen from the population, using the fitness scores as a probability distribution. The reproduction between them happens via uniform crossover, i.e., a new candidate vector is created by taking the weight of each model from the library from one of the two parent vectors with equal probability. Mutation is then applied to the new individual with 100% chance, and it consists of modifying all weights in the candidate vector by a small factor σ. Furthermore, σ-decay is utilized, meaning that the mutation factor decreases over the generations, making the changes to the new candidates smaller and smaller as the algorithm converges towards an optimal solution. The newly made

solutions are added to the old population to create the candidate pool and a new population of N is chosen by selecting the weight vectors with highest fitness. Elitism is also utilized, meaning that a percentage of the old generation, e, with best performance is copied to the new one without modification, to conserve the good genes.

The fitness of the candidate solutions is based on their performance on the target dataset. The individual models' predictions $p_0, ..., p_M$ are combined as a weighted sum, using the normalized vector weights of the weight vector $w = \langle w_0, ..., w_M \rangle$ to form the ensemble's prediction $\hat{y}$:

$$\hat{y} = p_0 w_0 + ... + p_M w_M$$

This prediction is compared to the ground truth, y, using an appropriate metric for the dataset, which results in the candidate's fitness score.

Apart from optimizing predictive performance, we also strive to create sparse ensembles for increased efficiency. Similarly to previous work [74], the votes of models whose weight in the weight vector is below a certain threshold z are not considered. This thresholding serves a dual purpose, and essentially ensures that the weight vector is constrained to non-negative values, given that negative weights have been found to be unreliable besides not having an intuitive meaning [32, 74]. The algorithm does not simply ignore the models whose weight is less than z; It encourages sparseness by incorporating a penalty term into the loss function that penalizes larger ensembles more than sparser ones. L_0 normalization is utilized – an approach also often used for learning sparse neural networks by penalizing the number of non-zero weights [36]. Analogously, our penalty term p is defined as

$$p = \lambda ||w||_0$$

where λ is a weighting factor for the regularization and w is the weight vector. Therefore, the overall fitness function f can be defined as follows:

$$f(w) = metric(\hat{y}, y) - p$$

The GA for ENeemble SEleCtion and integration (ENSEC) is summarized in the pseudocode for Algorithm 2.

The algorithm thus has a number of hyperparameters to set: number of generations n_gen, population size N, elitism percentage e, the mutation factor σ and decay d, the threshold weight value below which a model's contribution is zeroed out z, and the penalty term λ.

To assess the performance of the genetic algorithm, its output is evaluated, i.e., the candidate solution with highest fitness in the final generation. The ensemble corresponding to the resulting weight vector is assessed on unseen data with a fitting metric of performance, such as accuracy or mean IoU, depending on the data task, in order to demonstrate its generalization abilities. More on these metrics later in this chapter.

4.2 Evolutionary Algorithm for Data Augmentation Pipelines – DAP

Before combining the task of ensemble selection with the task of generating a suitable DA pipeline, we design an evolutionary algorithm that focuses on test-time data augmentation

Algorithm 2 ENSEC

Input: target data D, model library m, number of generations n_gen, population size N, and elitism percentage e.

$generation \leftarrow 0$
$population \leftarrow N$ weight vectors of length len(m)
$pop\_fitnesses \leftarrow$ evaluate($population, m, D$)

while $generation \neq n\_gen$ **do**
  $new\_candidates \leftarrow \emptyset$
  while len($new\_candidates$) $\neq N$ **do**
    $parents \leftarrow$ 2 candidates chosen from $population$ with probability $pop\_fitnesses$
    $candidate \leftarrow$ crossover($parents$)
    $candidate \leftarrow$ mutate($candidate$)
    $new\_candidates \leftarrow new\_candidates + candidate$
  end while
  $new\_candidates\_fitnesses \leftarrow$ evaluate($new\_candidates, m, D$)

  $elites \leftarrow$ elitism($population, e$)
  $all\_candidates \leftarrow population + new\_candidates - elites$
  $population \leftarrow elites$
  while len($population$) $\neq N$ **do**
    $best\_candidate \leftarrow$ best_fitness($all\_candidates$)
    $population \leftarrow best\_candidate$
    $all\_candidates \leftarrow all\_candidates - best\_candidate$
  end while
  $generation \leftarrow generation + 1$
end while

$best\_candidate \leftarrow$ best_fitness($population$)
return $best\_candidate$

adaptation only. This is both necessary for the designed combinations of the two approaches, and also serves as an ablation test, which investigates whether a data augmentation pipeline on its own already achieves better performance without the need for ensembling. Thus, the second EA implemented in this thesis has the aim to evolve an optimal augmentation pipeline, i.e., a sequence of transformations that once applied to the data at hand, helps decrease the data shift and thus increase the performance of the model at test-time. In this algorithm, once again a vector of continuous values is optimized, but in this case they correspond to the parameters of the transformation functions for data augmentation. In our implementation, this concerns five functions: brightness adjustment, contrast adjustment, gamma correction, sharpening and resizing. The initial population of size N is generated by randomly sampling a parameter within the appropriate range for each of the five transformations. A new candidate solution is created by mutating an old solution, chosen randomly with the fitness acting as probability distribution. The mutation consists of modifying the values in the vector by a small term σ, sampled from a normal distribution. This term also decreases over the generations, depending on a σ-decay value. The new population is generated by taking the individuals with highest fitness from the combined candidate pool consisting of the previous population and the new solutions, and mutating them.

The fitness of the solutions is calculated using a specific model and dataset. The samples from the optimization set of the data are transformed using the transformation functions parametrized by the vector of parameters, and only then presented to the model. The model's output $\hat{y}$ is compared to the ground truth y and the result comprises the fitness. Thus, given transformations $t_0, ..., t_T$ and their corresponding parameters $p = \langle p_0, ..., p_T \rangle$, a batch of data samples *input* and model m the fitness function can be defined as follows:

$$\hat{y} = m(t_T(...t_1(t_0(input, p_0), p_1), p_T))$$

$$f(p) = metric(\hat{y}, y)$$

The GA for Data Augmentation Pipeline optimization (DAP) is summarized by the pseudocode in Algorithm 3.

The algorithm's tunable parameters are: number of generations n_gen, population size N, and the mutation factor σ applied to the different parameter values. Each transformation's σ can be selected separately. The number and types of transformations, as well as the model(s) used for evaluation can also be seen as a hyperparameter.

To evaluate the performance of this algorithm, again the candidate solution with the highest fitness from the final generation is assessed. The data augmentation pipeline corresponding to the resulting vector of parameter values is applied to unseen data, which is then fed through a model and the outputs are scored with a fitting metric of performance, such as accuracy or mean IoU, depending on the task.

4.3 Combining Ensemble Selection and Data Augmentations

Finally, the main focus of this thesis is presented: incorporating ensembling with DA pipelines. By combining the two evolutionary algorithms in a number of sequential pipelines, we hope to harness the benefits of both and further improve test-time performance on out-of-distribution

Algorithm 3 DAP

Input: target data D, trained model m, number of generations n_gen, population size N, and elitism percentage e.

$generation \leftarrow 0$
$population \leftarrow N$ transformation pipelines
$pop\_fitnesses \leftarrow$ evaluate$(population, m, D)$

while $generation \neq n\_gen$ **do**
  $new\_candidates \leftarrow \emptyset$
  while len$(new_candidates) \neq N$ **do**
    $old \leftarrow$ a candidate chosen from $population$ with probability $pop\_fitnesses$
    $candidate \leftarrow$ mutate(old)
    $new\_candidates \leftarrow new\_candidates + candidate$
  end while
  $new\_candidates\_fitnesses \leftarrow$ evaluate$(new_candidates, m, D)$

  $elites \leftarrow$ elitism$(population, e)$
  $all\_candidates \leftarrow population + new\_candidates - elites$
  $population \leftarrow elites$
  while len$(population) \neq N$ **do**
    $best\_candidate \leftarrow$ best_fitness$(all_candidates)$
    $population \leftarrow population + best\_candidate$
    $all\_candidates \leftarrow all\_candidates - best\_candidate$
  end while
  $generation \leftarrow generation + 1$
end while

$best\_candidate \leftarrow$ best_fitness$(population)$
return $best\_candidate$

data. Two main directions are explored, both resulting in an ensemble of models best fitting the task and data augmentation pipelines that bring the unseen data samples closer to the train data distribution.

4.3.1 ENSEC Followed by DAP

In the first proposed approach, ENSEC is used initially to identify a strong ensemble for the target data from the available library. Next, one or more DA pipelines specific to that ensemble and its needs are optimized using DAP. This ensures that the DA pipeline is not influenced or biased by the predictions of models that do not become ensemble members. There are two options possible for running DAP:

1. Learning a "general" pipeline that is uniform across all models within the ensemble. In that case, the candidate solutions are the same as when DAP is run independently. Every candidate pipeline is evaluated using the whole ensemble, i.e., the data samples are modified with the transformations before being fed to all models in the ensemble and their predictions are weighted and combined as per ENSEC's resulting weight vector to make up the output that the fitness metric is calculated based on.

2. Learning an "individual" pipeline for each model within the ensemble. In this case, a candidate solution comprises of a list of DA pipelines with length equal to the number of models in the ensemble. We iteratively optimize one pipeline at a time for a few epochs, then freeze it, and move on to the pipeline of the next model in the ensemble. While the pipeline of the first model in the ensemble is being optimized, the fitness evaluation is based on the weighed combined predictions of the ensemble models, with the input being modified with the pipeline before being presented to the first model and being presented as-is to the rest of the models. Therefore, each model's individual pipeline is evaluated for its contribution to the overall ensemble performance.

The two algorithm pipelines will be henceforth referred to as ENSDAP(gen) and ENSDAP(ind) respectively for brevity. Besides the above mentioned changes in logic in DAP, the algorithm and its hyperparameters, fitness function, etc. remain the same.

4.3.2 DAP Followed by ENSEC

For the next approach the order of the two algorithms is swapped, with the goal of first optimizing transformations to reverse the data shift and then choosing the best-performing ensemble for that transformed data. First, DAP is run to obtain one or more pipelines of data transformations best fitting the target dataset. Then, ENSEC is run, where the discovered augmentations are used to modify the data samples before they are fed to the models during optimization. Once again, we can learn either a "general" pipeline to be used by all models or multiple "individual" pipelines. The latter approach would require learning a set of transformations for each of the models in the model library. Due to the required large amount of time and compute resources per experiment, this option is unfortunately not explored.

We do, however, examine the general pipeline approach. To ensure that the pipeline is indeed general and works with all models from the library, the candidate solutions are not evaluated only with a single model. Instead, at every generation, a subset of models is randomly

sampled from the library and used for the fitness calculation of the whole population. A candidate pipeline is evaluated with every model from the random sample independently and the scores are averaged out to obtain the overall candidate fitness. Changing the subset of models for evaluation at every generation will hopefully prevent bias towards a particular model's predictions. The general DA pipeline is then used during ENSEC: when a candidate ensemble is evaluated, a data sample is first transformed and then passed on to the models. This algorithm pipeline will be named DAPENS for brevity. Once again, besides the described changes to the semantics of the algorithms, they remain the same.

4.4 Data

The proposed approaches are evaluated on a number of datasets to assess their generality and efficacy for different tasks. For the task of image classification, a benchmark dataset is chosen: CIFAR-100, which has 100 classes representing everyday objects and animals in 32x32 images [29]. For the task of image segmentation, PASCAL Visual Object Classes (PascalVOC) is utilized: A benchmark dataset containing 15,000 images of size 512x512 with 20 object categories including vehicles, household, animals, and other [19]. We make use of the images' pixel-level segmentation annotations. Importantly, some images have multiple labels, and all images contain an additional background class, that pixels are attributed to if they do not match the characteristics of any other class. Examples of both datasets are visualized in Figures 4.1 and 4.2.

Figure 4.1: Example images of the CIFAR-100 data. The labels are included above the images.

Figure 4.2: Example images of the PascalVOC data. The first sample has two labels: train and person. The second one also: car and bike. The third sample has only one label: cow.

Finally, the last datasets originate from a real-world use case: namely, the data utilized at ASMPT and their customers, consisting of images of chips for segmentation. To be more concrete, ASMPT has provided three datasets belonging to real customers and containing a real data shift. To follow privacy guidelines, we will refer to these datasets as Anon #1, Anon #2 and Anon #3, and will not include visualizations. In the chip images, there are four classes to segment: wire, ball, wedge, and epoxy, and example samples are given in Figure 4.3. Once again, due to privacy limitations, the visualized images come from a synthetic dataset. For these three datasets, each sample has two images available with two different lighting conditions: one with top lighting and one with lighting from the side. Both are fed to the model as a multi-channel input and mapped to a single prediction.

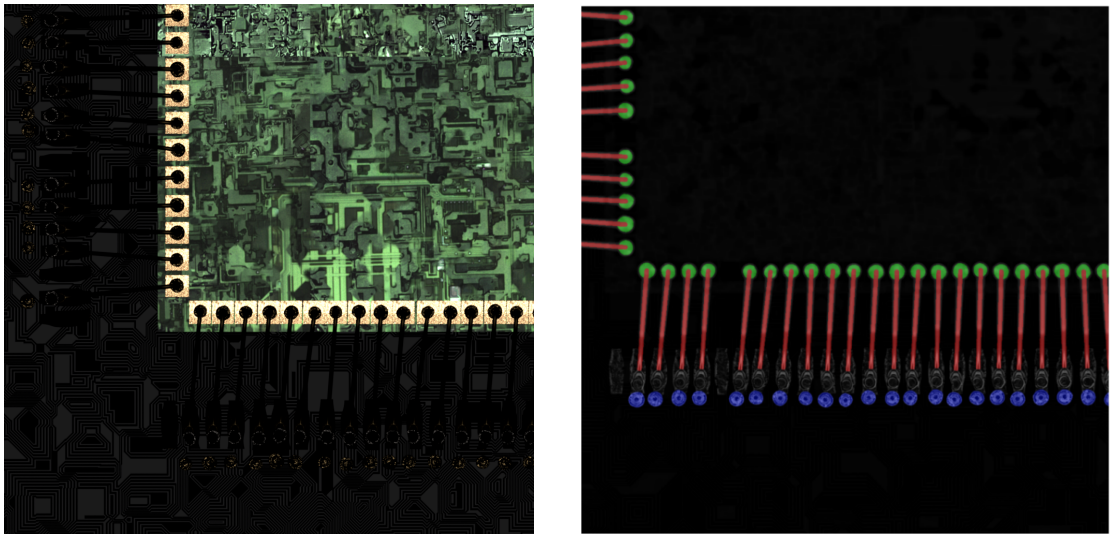


Figure 4.3: On the left is an example sample from the ASMPT-provided synthesized dataset, showcasing a chip image used for segmentation. On the right is an example of a segmented and labeled sample from the same synthetic dataset, where red is the wire class, green is the ball class, and blue is the wedge class.

For each of the datasets, fully trained models are acquired. Given the heavy computational load of ensembles, especially for the task of segmentation, and our aim to make the test-time

adaptation process as efficient as possible, the number of the data samples used during the EA is restricted. The optimization and test set sizes are reported in the next chapter. There is another reason for this choice: In a real-world scenario, such as that at ASMPT, where a data shift has occurred and we attempt to alleviate the subsequent drop in performance, it is likely that not a lot of (labeled) examples are available to work with. It would be ideal for ASMPT's customers to, e.g., only have to provide a handful of images containing the data shift and be given an optimized model or set of models quickly.

4.5 Evaluation Metrics

For the classification tasks, the models and ensembles are evaluated using the accuracy metric, defined as the ratio of correctly labeled samples to the total number of samples. When it comes to evaluating the performance of the semantic segmentation models, the frequently used metric Jaccard index or Intersection over Union (IoU) [23, 27] is used. The IoU score between a ground-truth segmentation mask Y and a predicted mask $\hat{Y}$ is defined as:

$$IoU(Y, \hat{Y}) = \frac{|Y \cap \hat{Y}|}{|Y \cup \hat{Y}|}$$

IoU measures the amount of overlap between the ground truth and predicted segmentation, with values closer to 1 indicating a better segmentation [48]. The intersection here is defined by how much the detected and ground truth segmentations overlap with each other, while the union is the total area occupied by both bounding boxes. For multi-class applications, the overall IoU for multiple classes is calculated, i.e., the Mean IoU (MIoU) [34, 45].

Chapter 5

Experimental Setup

All implemented algorithms, evaluation scripts, data loading and transformations are included in our GitHub repository, ready to be used with the publicly available datasets. The conducted experiments and the specific models and data used for them are detailed below.

5.1 Experiments

To assess the proposed evolutionary algorithms and evolutionary pipelines a number of experiments are conducted.

1) Evaluating the performance and efficiency of EAs for ensemble selection. For this goal, two types of tests are performed using ENSEC. Firstly, investigating the performance of the algorithm on two datasets whose test data comes from the same distribution as the train data: CIFAR-100 and PascalVOC. Secondly, assessing the performance when data shift is present, using the data from the three anonymous ASMPT customers. Furthermore, CIFAR-100 and PascalVOC are also utilized, but with an artificially introduced data shift. The following 12 datasets are generated by augmenting the original CIFAR-100 and PascalVOC datasets using image manipulations:

- CIFAR-100-BRIGHT and PascalVOC-BRIGHT: created by adjusting the image brightness with a factor of 0.5.
- CIFAR-100-CONTRAST and PascalVOC-CONTRAST: created by adjusting the image contrast with a factor of 1.5.
- CIFAR-100-SHARP and PascalVOC-SHARP: created by adjusting the image sharpness with a factor of 1.5.
- CIFAR-100-GAMMA and PascalVOC-GAMMA: created by adjusting the image gamma with a factor of 1.5.
- CIFAR-100-BLUR and PascalVOC-BLUR: created by adding Gaussian blur with with kernel size 5x9 and standard deviation $\in (0.1, 1.5)$.
- CIFAR-100-COMBI and PascalVOC-COMBI: created by adjusting the image brightness by a small factor in $[-0.4, 0.4]$, and adjusting the contrast, sharpness and gamma by

small factors in $[0.6, 1.4]$. This results in a more complex and multidimensional data shift.

The results of this experiment will be considered one baseline to improve on as ensembles have already proven useful for tackling data shift.

2) Evaluating the performance and efficiency of EAs for DA pipeline discovery to reverse data shifts. In this set of experiments DAP is evaluated using the same tests as for ENSEC: on data with a test set from the same distribution as the train set, and on data with a shift present. The same datasets listed in 1) are also utilized here. In the experiments with artificially-induced data shift, the focus is first on augmenting the data with transformations present in the pipeline optimized by DAP, to demonstrate that the resulting pipeline can reverse specific shifts in the data. After, we consider Gaussian blur to investigate whether this method is also successful when the applied transformation is not part of the pipeline and thus not directly reversible. Similarly, the data in the practical use cases was not transformed with fixed augmentations, and is likely out-of-distribution given many factors. It is important that the pipeline remains robust to unknown transformations and can still improve the performance of the model on such data.

3) Evaluating the performance and efficiency of combining EAs for ensemble selection and DA pipeline discovery. This set of experiments covers the three methods of combining ENSEC and DAP: ENSDAP(gen), ENSDAP(ind), and DAPENS. For all approaches, the following datasets are considered:

- Test data sampled from the same distribution as the train data: CIFAR-100 and PascalVOC.
- CIFAR-100-COMBI and PascalVOC-COMBI.
- Data from the three anonymous ASMPT customers.

Due to the stochastic nature of genetic evolution, every described experiment is repeated 3 times and the scores reported are averaged over those trials. Given how compute-heavy and numerous the experiments are, a full, broad hyperparameter search is not conducted. The utilized values are ones identified empirically via an initial brief exploration. The hyperparameters of the evolutionary algorithms are given in Tables 5.1 and 5.2. In the case of ENSEC, the initial population is generated differently for the different datasets. For CIFAR- and PascalVOC-derived data, where the baseline ensemble already improves on the performance of the independent separate models, the initial candidate solutions have all model weights being close to 1. For the ASMPT-provided data, where the performance of the baseline ensemble is actually lower than that of the best base learner, the initial population consists of vectors of random values between 0 and 1. The fitness function across all algorithms for the CIFAR- and PascalVOC-derived data consists of the cross entropy loss, while for the ASMPT data MIoU is used.

n_gen	**population size** N	**elitism** e%	**mutation factor** σ	σ**-decay** d	**penalty term** λ	**threshold** z
15	30	20	0.2	0.01	0.01	0.7 / 0.1

Table 5.1: The hyperparameter values of ENSEC. Note that the threshold for zeroing out model weights is 0.1 when the initial population is generated by randomly sampling values $\in (0, 1)$, and 0.7 when the initial population starts close to the baseline ensemble.

n_gen	**population size** N	**elitism** e%	**mutation factor** σ	σ**-decay** d
20	15	0	0.1	0.9

Table 5.2: The hyperparameter values of DAP.

When the two EAs are run sequentially, the hyperparameters are the same with one exception: ENSDAP(ind) has a different number of generations. It is a variable number, $10m$, determined by the number of models m in the ensemble that require an individual pipeline.

Segmentation is a rather compute-heavy and time-consuming operation, especially when multiple models must be run and evaluated for every EA candidate solution. This would make ENSEC rather inefficient and motivates us to implement a speed-up technique. Essentially, in addition to working with the model library, ENSEC can also work with stored predictions of models. For Pascal-derived data and ASMPT-provided data, the model predictions are stored prior to running ENSEC, and this provides us with a 22x speed-up on average in ENSEC run-time. All reported run-time information is collected with prediction storing and loading enabled. This optimization could make DAP more efficient as well, especially in the ENSDAP(ind) setting, but is less suitable, given that the new image pipelines modify the data given to the model and thus the stored predictions obtained with one pipeline are no longer applicable for another.

5.2 Models and Performance Baselines

Model library details. In the case of the CIFAR-100 dataset, an online source provides 10 already pre-trained models: https://github.com/chenyaofo/pytorch-cifar-models/tree/master. The models use the ResNet, MobileNetV2 and ShuffleNet architectures which have at most 5.37M trainable parameters. In the case of PascalVOC, a pre-trained backbone offered by PyTorch[1], and fully train 15 MobileNetV3 models with the same structure and different random seeds. Each has 11.03M trainable parameters. Lastly, for the ASMPT-provided wire datasets, the company provides eight deep CNNs with an encoder-decoder architecture. The encoder consists of four blocks of convolutions, batch normalization and ReLU activations, with the feature dimension increasing from 64 to 512 while the spatial resolution is reduced through stridden convolutions. In the decoder, upsampling operations, convolutional and batch normalization layers produce and refine the segmentation output. The models were trained on in-house data specifically for research and are not representative of the production

[1]An overview of the segmentation models offered by PyTorch and the backbone weights are given here: https://pytorch.org/vision/stable/models.html#semantic-segmentation is utilized. We use the weights `DeepLabV3_MobileNet_V3_Large_Weights.COCO_WITH_VOC_LABELS_V1`.

Dataset	Optimization set size		Test set size
	ENSEC	DAP	
CIFAR-100	30	150	9850
PascalVOC	30	75	1298
Anon #1	20	20	6
Anon #2	6	6	3
Anon #3	9	9	3

Table 5.3: Details of the optimization and test sets used in the EAs and subsequent evaluation of the best solutions, both of which contain test samples unseen during training. The CIFAR- and Pascal-derived datasets share the statistics of CIFAR-100 and PascalVOC respectively. For CIFAR-100 and PascalVOC, 150 samples are set aside, from which the optimization set is obtained.

models used at ASMPT. The networks all have the same architecture but were trained using different seeds. Each has 20.29M trainable parameters.

It is important to note that none of the utilized models have state-of-the-art competitive predictive ability as the focus of this thesis falls on the level of performance improvement with the various proposed methods.

Choice of baselines. As already established, one of our baselines is the performance of ENSEC, given the widespread application of ensembling as a data shift solution. Two more performance baselines are considered for each experiment: the best performing base learner in the library for the corresponding dataset, and the baseline ensemble consisting of the full model library, with the predictions of the learners averaged out (used often in literature as a baseline [56]). Furthermore, for the ASMPT-provided data only, there is a fourth baseline to compare to, namely continuous training – another state-of-the-art method for tackling out of distribution data [31]. For that, for each of the three datasets, we continue training the best model of the library for 100 epochs, using the train set but with the optimization samples from the target distribution added to it.

Baseline Performance. Different datasets have different amount of data available, meaning that the optimization/test split differs. Table 5.3 contains a summary of the samples used in the EAs and for testing.

The performance of the utilized models for ensemble selection on the described datasets and their corresponding test sets, are included in Tables 5.4, 5.5 and 5.6. The latter two baselines, best base learner and full-library ensemble, are also present. Note that for the Pascal-derived datasets with induced data shift, the model performance barely drops. This is likely due to the simplicity of the one-dimensional shifts introduced: the 3-channel information is still there but in a different scale. The models should just be scale-invariant to be able to still accurately work on such data, and they happen to be. Given that the models are built on pre-trained weights, it is possible that the pre-training included a set of augmentations for more robust results.

ModelID	Original	BRIGHT	CONTRAST	SHARP	GAMMA	BLUR	COMBI
1	68.4	66.3	59.1	31.5	66.7	39.1	44.4
2	70.0	68.1	61.0	32.2	68.0	39.0	44.8
3	71.2	69.4	64.4	32.5	69.4	38.6	46.5
4	72.2	71.0	64.4	32.5	70.2	41.2	47.9
5	70.7	69.0	61.2	33.1	68.8	42.1	46.2
6	73.7	72.2	64.4	36.0	71.8	42.5	49.6
7	**74.3**	**72.5**	**66.0**	**36.1**	**72.3**	42.3	**50.1**
8	72.2	70.0	63.2	31.0	70.4	41.9	46.0
9	74.1	72.1	64.5	33.0	71.8	**45.3**	47.1
10	67.4	65.1	54.9	30.1	64.8	38.5	40.6
ALL	79.8	78.7	72.4	78.3	78.3	46.7	57.5

Table 5.4: Accuracy (%) of the 10 pre-trained CIFAR-100 models on the test set. The best performing base learners are highlighted in bold. For CIFAR-100-COMBI, the average performance over the three combination augmentations is reported and the standard deviation is omitted as it is always 0.0. The last row contains the baseline ensemble score.

ModelID	Original	BRIGHT	CONTRAST	SHARP	GAMMA	BLUR	COMBI
1	52.9	53.2	51.1	51.0	53.6	51.3	51.5 (± 1.1)
2	52.6	52.6	51.3	51.6	52.9	52.1	50.7 (± 1.1)
3	53.0	52.9	51.0	50.9	53.2	51.4	50.3 (± 1.3)
4	52.5	52.1	50.7	51.7	52.8	52.0	50.5 (± 1.3)
5	**54.8**	**54.7**	52.7	**53.8**	**55.4**	**53.7**	52.7 (± 1.4)
6	53.0	53.1	51.0	51.4	53.5	52.1	51.3 (± 1.5)
7	54.2	54.3	52.2	53.1	54.8	53.0	52.4 (± 1.5)
8	51.9	51.9	50.1	51.2	52.2	51.1	50.1 (±1.8)
9	51.0	51.7	49.8	50.0	51.4	50.5	49.6 (± 1.1)
10	53.7	53.6	51.5	52.2	54.2	52.8	51.9 (± 1.7)
11	52.2	51.9	51.2	51.9	52.2	51.9	50.9 (± 1.6)
12	53.3	53.7	51.4	52.6	53.9	52.9	51.6 (± 2.0)
13	54.1	53.9	**53.2**	53.7	54.1	53.3	**52.9 (± 1.4)**
14	51.9	52.0	49.7	50.9	52.6	51.1	49.8 (± 1.7)
15	52.5	52.4	51.3	51.8	52.6	51.5	51.4 (± 1.3)
ALL	56.9	56.7	55.5	55.8	57.5	56.3	54.9 (± 1.3)

Table 5.5: Mean IoU (%) of the 15 trained models on the PascalVOC test dataset. For PascalVOC-COMBI, the average performance over the three combination augmentations is reported. The best performing base learner is highlighted in bold. The last row contains the baseline ensemble score.

ModelID	Anon #1	Anon #2	Anon #3
1	54.0	83.0	74.0
2	61.0	83.0	76.0
3	**78.0**	84.0	74.0
4	68.0	81.0	76.0
5	66.0	81.0	66.0
6	58.0	77.0	73.0
7	73.0	**85.0**	74.0
8	44.0	79.0	**79.0**
ALL	69.1	83.5	76.0

Table 5.6: Mean IoU (%) of the 8 segmentation models on the ASMPT-provided test datasets. The best performing base learners are highlighted in bold. The last row contains the baseline ensemble score.

Lastly, the results of the continuous training on ASMPT data are summarized in Table 5.7. The training is conducted in 5 hours, for about 100 epochs.

Anon #1	Anon #2	Anon #3
86.9	87.2	86.1

Table 5.7: Results in MIoU (%) of continuous training task for the ASMPT-provided data. The best base learner per dataset is used for the training.

5.3 System Specifications

All of our experiments are conducted on a compute cluster offered by ASMPT. The training and evaluation of our proposed approaches was done on an NVDIA A100 Tensor Core PU with 80GB memory. We also make use of AMD Epyc 9334 CPU @ 3.85GHz.

Chapter 6

Results

This chapter presents the performance of the proposed approaches: ENSDAP(gen) where an ensemble is optimized and then a general DA pipeline for all of its members, ENSDAP(ind) where an ensemble is optimized and then individual DA pipelines for each of its models, and DAPENS where a general DA pipeline is found for all library models, and then an ensemble is built on top of the transformed data. We report both predictive performance and run-time as evaluated on CIFAR- and Pascal-derived datasets and ASMPT-provided data. The results are compared to the four baselines (best model in the library, baseline ensemble, ENSEC for sparse ensembles, continuous training) and DAP which only evolves DA pipelines.

6.1 EAs for Ensemble Selection: ENSEC

ENSEC is evaluated on the 17 datasets introduced in the previous chapter. We report the performance of the identified best ensemble on the test data in terms of accuracy for the CIFAR-100-derived datasets, and in terms of MIoU for the PascalVOC-derived ones and ASMPT-provided data. The results are summarized in Table 6.1. They will act as a baseline to exceed, given that ensembling is already a well-established mitigation method for data shift.

As we can see, across almost all datasets for both standard and augmented data, ENSEC results in ensembles that consistently outperform the best base learner. Especially for the classification task, there is a large improvement of 2-5%, similarly to what is observed in relevant literature on ensembling [50, 56, 58]. The performance of the baseline ensemble is also majorly better than the single model. These two results highlight the benefit of ensembling. For the CIFAR- and Pascal-derived datasets, the baseline ensembles obtain higher scores than those produced by ENSEC. However, in the case of ASMPT-provided data, ENSEC works significantly better than the baseline ensemble, showcasing a key characteristic that makes our approach to ensembling more appealing: excluding some poor-performing models from the library and combining only the strong ones that synergize well can lead to consistent and reliable results. Furthermore, the ENSEC ensembles on average contain less than half the model library, which makes them significantly faster.

Dataset	Best base learner	Baseline ensemble	ENSEC	ENSEC ensemble size
CIFAR-100	74.3	**79.8**	77.7 (± 0.1)	5
CIFAR-100-BRIGHT	72.5	**78.7**	76.6 (± 0.0)	4
CIFAR-100-CONTRAST	66.0	**72.4**	69.5 (± 0.2)	3
CIFAR-100-SHARP	36.1	**41.0**	40.3 (± 0.2)	5
CIFAR-100-GAMMA	72.3	**78.3**	77.0 (± 0.0)	6
CIFAR-100-BLUR	45.3	46.7	**47.1 (± 0.3)**	6
CIFAR-100-COMBI	50.5 (± 1.7)	**57.5 (± 0.0)**	54.8 (± 2.7)	3
PascalVOC	54.8	**56.9**	56.0 (± 0.4)	2
PascalVOC-BRIGHT	54.7	**56.7**	56.1 (± 0.2)	4
PascalVOC-CONTRAST	53.2	**55.5**	54.0 (± 0.4)	3
PascalVOC-SHARP	53.8	**55.8**	54.9 (± 0.5)	3
PascalVOC-GAMMA	55.4	**57.4**	56.8 (± 0.0)	3
PascalVOC-BLUR	53.7	**56.3**	54.3 (± 0.3)	2
PascalVOC-COMBI	52.9 (± 1.4)	**54.9 (± 1.3)**	53.5 (± 0.5)	2
Anon #1	78.0	69.1	**79.0 (± 0.4)**	3
Anon #2	85.0	83.5	**85.2 (± 0.2)**	1
Anon #3	**79.0**	76.0	78.7 (± 0.1)	2

Table 6.1: ENSEC results with scores averaged over the three runs with different seeds. Classification is evaluated using accuracy (%) and segmentation is evaluated using MIoU (%). Ensemble size is reported in terms of median number of models. Note that for CIFAR-100 there are 10 models in the library, for PascalVOC there are 15 and for ASMPT-provided data there are 8. The best result per dataset is in bold, and second best underlined.

6.2 EAs for DA Pipeline Discovery: DAP

DAP is also evaluated on the full set of 17 datasets. Performance is reported using the identified best DA pipelines on the test data in terms of accuracy for the CIFAR-100-derived datasets, and in terms of MIoU for the PascalVOC-derived ones and ASMPT-provided data. The results are summarized in Table 6.2.

The DAP results seem to exceed the best base learner for almost all datasets. For CIFAR-derived data, it outperforms the three baselines on 3 datasets with ∼10% points in score. For Pascal-derived data, the improvements are minimal, at ∼0.3% point, but still present for the majority of datasets. For the ASMPT-provided data, DAP once again results in the highest score of all four reported methods, exceeding the single best model by a couple of points on average. This shows that this novel approach of test-time augmentation adaptation as a technique independent of ensembling is also a good method for tackling data shift.

Dataset	Best base learner	Baseline ensemble	ENSEC	DAP
CIFAR-100	74.3	**79.8**	77.7 (± 0.1)	73.6 (± 0.5)
CIFAR-100-BRIGHT	72.5	**78.7**	76.6 (± 0.0)	73.1 (± 0.3)
CIFAR-100-CONTRAST	66.0	72.4	69.5 (± 0.2)	**73.3 (± 0.4)**
CIFAR-100-SHARP	36.1	41.0	40.3 (± 0.2)	**47.3 (± 0.5)**
CIFAR-100-GAMMA	72.3	**78.3**	77.0 (± 0.0)	73.0 (± 0.3)
CIFAR-100-BLUR	45.3	46.7	**47.1 (± 0.3)**	46.7 (± 0.5)
CIFAR-100-COMBI	50.5 (± 1.7)	57.5 (± 0.0)	54.8 (± 2.7)	**66.4 (± 0.9)**
PascalVOC	54.8	**56.9**	56.0 (± 0.4)	55.1 (± 0.3)
PascalVOC-BRIGHT	54.7	**56.7**	56.1 (± 0.2)	54.8 (± 0.3)
PascalVOC-CONTRAST	53.2	**55.5**	54.0 (± 0.4)	53.4 (± 0.2)
PascalVOC-SHARP	53.8	**55.8**	54.9 (± 0.5)	52.6 (± 0.1)
PascalVOC-GAMMA	55.4	**57.4**	56.8 (± 0.0)	55.0 (± 0.3)
PascalVOC-BLUR	53.7	**56.3**	54.3 (± 0.3)	53.8 (± 0.2)
PascalVOC-COMBI	52.9 (± 1.4)	**54.9 (± 1.3)**	53.5 (± 0.5)	53.4 (± 1.1)
Anon #1	78.0	69.1	79.0 (± 0.4)	**80.9 (± 1.3)**
Anon #2	85.0	83.5	85.2 (± 0.2)	**86.7 (± 0.2)**
Anon #3	79.0	76.0	78.7 (± 0.1)	**80.3 (± 0.9)**

Table 6.2: Results of DAP, using the best base models for evaluation. All DAP scores are averaged over the three runs with different seeds; for classification accuracy (%) is reported and for segmentation MIoU (%). The three baselines are also included, and the best result is in bold, while the second best is underlined.

6.3 Combining ENSEC and DAP

The three sequential combinations of the two algorithms are evaluated on the test data in terms of accuracy for the CIFAR-100-derived datasets, and in terms of MIoU for the PascalVOC-derived ones and ASMPT-provided data. The results are presented in Table 6.3.

Dataset	Best base learner	Baseline ensemble	ENSEC	ENSDAP(gen)	ENSDAP(ind)	DAPENS
CIFAR-100	74.3	**79.8**	77.7 (± 0.1)	77.7 (± 0.2)	74.4 (± 0.8)	77.2 (± 0.8)
CIFAR-COMBI	50.5 (± 1.7)	57.5 (± 0.0)	54.8 (± 2.7)	**69.8 (± 3.4)**	48.1 (± 7.7)	69.3 (± 1.0)
PascalVOC	54.8	**56.9**	56.0 (± 0.4)	56.2 (± 0.7)	56.4 (± 0.6)	56.6 (± 0.9)
Pascal-COMBI	52.9 (± 1.4)	**54.9 (± 1.3)**	53.5 (± 0.5)	54.5 (± 0.2)	52.5 (± 0.7)	53.9 (± 0.5)
Anon #1	78.0	69.1	79.0 (± 0.4)	**81.1 (± 1.6)**	76.5 (± 0.7)	75.7 (± 2.0)
Anon #2	85.0	83.5	85.2 (± 0.2)	**86.8 (± 0.4)**	84.1 (± 0.6)	85.3 (± 0.1)
Anon #3	79.0	76.0	78.7 (± 0.1)	**81.2 (± 2.8)**	77.6 (± 0.5)	78.8 (± 0.0)

Table 6.3: Results of the three proposed approaches on the test data. All scores are averaged over the three runs with different seeds. The three baselines are also included, and the best result is in bold, while the second best is underlined. Note that for the CIFAR-derived data, scores are in accuracy (%) while for all other datasets – in MIoU (%).

For the CIFAR-100 data with no data shift, there is only a small improvement over the single model, but for the CIFAR-COMBI dataset ENSDAP(gen) and DAPENS score much higher than all baselines, achieving a boost of 12% points over the baseline ensemble. For the PascalVOC dataset with no data shift, there is a boost in performance in all three approaches, and for Pascal-COMBI ENSDAP(gen) outperforms two of the baselines. For the ASMPT-provided data, DAPENS scores second for one customer, but does not exceed the scores of the single model for the other two. ENSDAP(ind) performs worse than the best base learner, and ENSDAP(gen) outperforms all three baselines for all three datasets with improvements of around $1 - 2\%$ points. For the ASMPT data, we can also compare to the outcome of continuous training, included in the previous chapter. Our approach does not exceed this particular state-of-the-art baseline, obtaining 1-5% points less.

Overall, it appears that ENSDAP(ind) achieves the worst and ENSDAP(gen) the best performance out of the three proposed approaches. Significance testing is performed between the five explored algorithms, choosing the Wilcoxon test as it does not assume equal variance in the results [21]. Given time constraints, the test is only conducted for the Anon #1 data, where the algorithms are run with additional four seeds for a larger sample size. We identify that ENSDAP(ind) performs significantly different, poorer, than the other algorithms with p-value < 0.05. However, when it comes to ENSEC, DAP, ENSDAP(gen) and DAPENS there is no statistical significant difference.

6.4 Visualizing ENSEC and DAP Results

In a more qualitative manner of evaluating the performance of ENSEC and DAP, a few random data samples are displayed for manual examination. To visually inspect whether the ensembles found with ENSEC reduce noise in predictions and improve details, we review a few images and their segmentation masks. Figure 6.1 contains examples of PascalVOC images from the test sets in three states: the original image, the segmentation mask obtained with the best base learner, and the segmentation mask obtained with an optimized by ENSEC ensemble. As can be seen, overall the two predictions are quite similar – for the horse riding image they are almost identical. However, there are certain places that the prediction is improved through ensembling: The wrongly identified blue segments in the tram picture are corrected and the segments in the urban image are more refined, i.e., the borders of the segments better follow the objects.

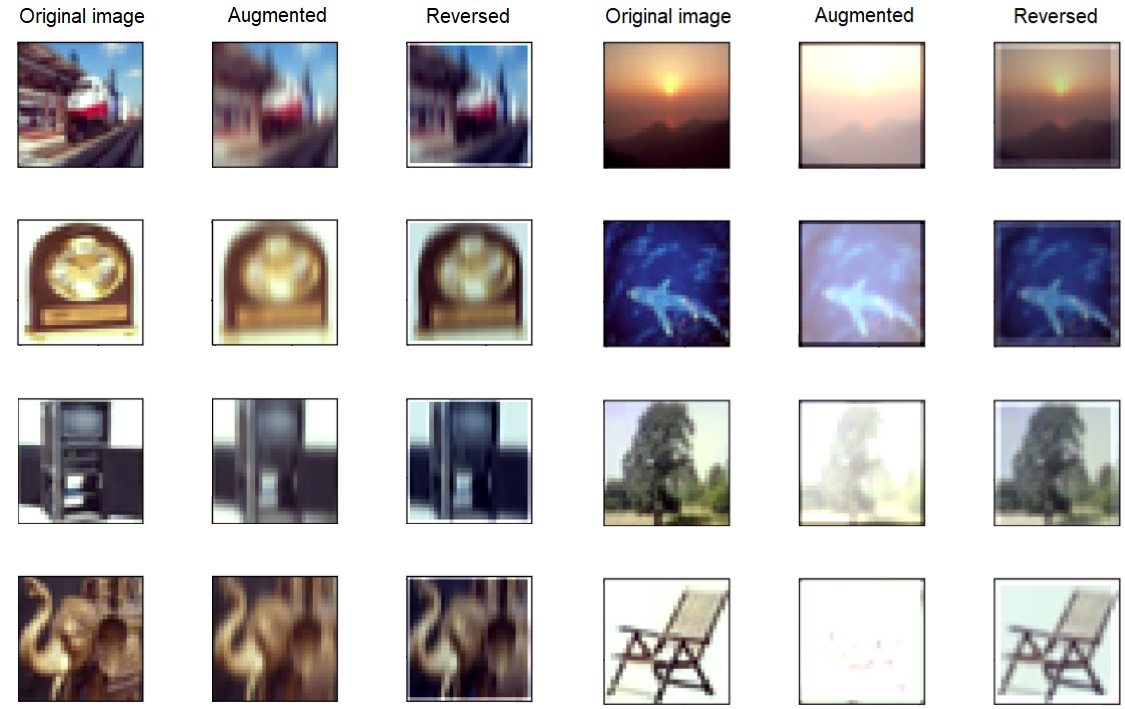
Next, a manual inspection of the effect of the optimized DA pipelines is in order. Figure 6.2 contains random examples of CIFAR-100 and PascalVOC images from the test sets in three states: the original image as-is, the image after the artificial data shift has been introduced, and the shifted image after the found DA pipeline has been applied. As we can see, the data shift is quite noticeable, especially in some cases. The COMBI setting for PascalVOC leads to almost unrecognizable to the human eye images and yet the base learners' performance barely drops. As previously discussed, this is likely due to the fact that the introduced shifts mainly require scale invariance. In any case, the images are generally reverted back to their original look after the application of the DA pipeline. This confirms that the algorithm indeed optimizes a set of transformations to reduce the present distribution shift while maintaining semantic information.

Figure 6.1: A few example images showing the effect of ensembling as a result of ENSEC. The left image is the original one from the dataset as-is, the middle image includes an overlaid segmentation mask as predicted by the best base learner, and the right image includes an overlaid segmentation mask as predicted by an ensemble identified with ENSEC.

6.5 Computation Time

6.5.1 Time Cost of EAs

The proposed approaches were timed during optimization to assess and compare their efficiency. The average run time per EA generation is reported for all datasets in Table 6.4. Note that for the execution of ENSEC, the predictions of the segmentation models are stored in advance for faster computation, as the task of segmenting an image is very time-consuming.



(a) CIFAR-100, Gaussian blur.

(b) CIFAR-100, COMBI setting.

(c) PascalVOC, brightness adjustment.

(d) PascalVOC, COMBI setting.

Figure 6.2: A few example images showcasing the effect of the optimized DA pipelines. The left image is the original one from the dataset as-is, the middle image is the artificially modified image to insert a data shift, and the right image is the one obtained after applying the optimal pipeline on the middle image. The corresponding dataset and shift are specified in the subfigure captions.

Dataset	ENSEC	DAP	ENSDAP(gen)		ENSDAP(ind)		DAPENS	
			ENSEC	DAP	ENSEC	DAP	DAP	ENSEC
CIFAR-100	46.2	26.4	46.2	29.7	46.2	38.3	38.3	53.6
PascalVOC	231.9	229.6	231.9	462.8	231.9	1198.0	1872.3	437.0
ASMPT-provided	1383.7	328.5	1383.7	423.3	1383.7	441.0	1061.2	1560.3

Table 6.4: The average run time in seconds of a single generation of the EAs for the different datasets. For the combined EAs, the run time per generation for the two stages is presented separately. The run times are averaged over all runs with the corresponding datasets, i.e., for PascalVOC we report the average time over all seeds run with the original PascalVOC data and the Pascal datasets with artificial data shift induced.

The best performing and most consistent proposed approach is ENSDAP(gen) and given the time per generation and the number of generations used, its reported results were achieved after optimization of about 40min for CIFAR-100, 3.2h for PascalVOC, and 8h for ASMPT-provided data. The large difference in run time between CIFAR-100 and the other two datasets comes from the fact that for the classification task a single label is produced per image, while for segmentation, every pixel is assigned a class, making the task a lot slower. The run-time on ASMPT data is longest because of the large size of the images that must be segmented pixel-wise.

In comparison, the reported results for continuous training were obtained after 5 hours of training, making our approach similarly efficient.

6.5.2 Inference Time

A fast solution at inference time is a primary objective for this thesis, which is why we also inspect the inference time given ensemble size and additional latency of the pipeline. The inference time of the sparse ensembles resulting from our proposed approaches is compared to that of the baseline ensemble. Furthermore, the overhead added by the usage of pipelines at test-time is illustrated. The average inference time for a single data sample recorded for the three main sets of data is visualized in Figures 6.3. Note that these numbers represent the average per dataset; For example, in the case of the ASMPT-provided data, we record the inference time of the models for all three used datasets and take the average of that. Also importantly, the reported numbers include feeding the models with the input samples, post-processing the outputs and aggregating them, but do not include data and model loading.

As expected, the larger the ensemble, the longer inference takes. The ensembles produced by ENSEC are quite sparse and about 3x faster than the baseline ensemble. The application of the DA pipeline adds some overhead. In the case of ENSDAP(gen) and DAPENS, where a single general pipeline is applied, it is more or less constant over the differently sized ensembles. However, for ENSDAP(ind), where each model has a different individual pipeline, the overhead is much larger and becomes more noticeable the bigger the ensemble is. Note that the added latency of augmentation for the ASMPT-provided data is more pronounced; For this data, a different backend is used, leading to this difference.

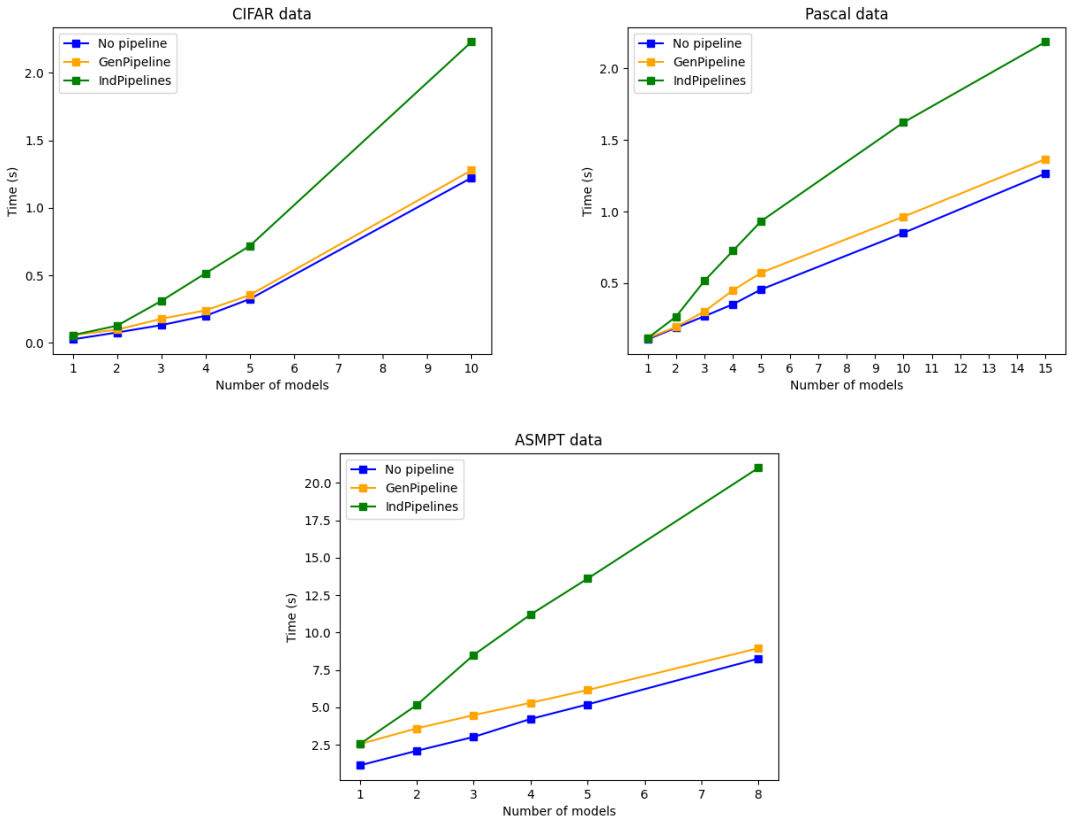


Figure 6.3: Average inference time per data sample for the different datasets and different number of models in an ensemble. We report for the cases of no DA pipeline applied, a single general DA pipeline (ENSDAP(gen) and DAPENS) and multiple individual DA pipelines (ENSDAP(ind)). Note the different scales of the y axis.

Chapter 7

Discussion

The problem this thesis tackles is that of data shift: when encountered with out-of-distribution data, models tend to perform poorly. We investigate whether the combination of ensembling and data augmentation pipelines, both optimized using evolution, can boost the performance at test-time when data shift is present. Three approaches are proposed: identifying an optimal ensemble followed by learning a general DA pipeline for all models within it (ENSDAP(gen)), identifying an optimal ensemble followed by optimizing an individual DA pipeline for every model in the ensemble (ENSDAP(ind)), and generating a general DA pipeline suitable for all models in the library followed by creating an ensemble on top of the transformed data (DAPENS).

Results. ENSEC optimizes sparse ensembles that work well across all datasets and tasks. The baseline ensemble comprising all models in the library still works best, but not in the case of ASMPT-provided data, making our solution more reliable. We believe the low scores of the baseline ensemble for ASMPT data are due to the larger variance in performance of the base learners in the ASMPT library of models. However, it is not a viable solution to simply remove the "poor" models from the library altogether as different models show to work well for different datasets. This highlights one of the benefits of choosing an optimal subset of models for the ensemble with ENSEC. Furthermore, given that the wire data from ASMPT represents a real world case where the data shift is not artificially induced, we believe our method demonstrates robustness and efficiency which is important in real life settings.

When it comes to using test-time data augmentation adaptation only and no ensembling, an improvement is observed over the best base learner and in the case of the wire data, there is also an improvement over the baseline ensemble. This shows that identifying a fitting augmentation pipeline to reverse the data shift is a viable method for improving test-time performance on out-of-distribution data on its own. Another benefit to this novel approach is interpretability as the optimized pipeline contains information about the setting in which the target images were recorded.

Out of the three proposed combinations of ensembling and DA pipelines, the best performing appears to be ENSDAP(gen). It achieves the highest (or second highest) score of all tested approaches for all datasets with present data shift.

ENSDAP(ind), on the other hand, has rather poor performance, essentially scoring lower than all of our baselines. This could be due to the introduced incoherence by the differently transformed data. Each model encounters a differently processed version of the same image, making the outputs harder to meaningfully combine. In contrast, with ENSDAP(gen), the ensemble is leveraging the same corrected image. Alternatively, this could be an indication that in combination with ensembling, a single set of transformations to tackle the data shift is best. We operated under the assumption that different models have learned different representations of the data and possibly pay attention to different aspects of the images, and consequently some could benefit from, e.g., brightening an image, while others might not. However, it could be the case that this diminishes the advantages of ensembles. The different pipelines might increase the variance of the ensemble predictions so high that the combined prediction is completely unreliable.

Lastly, DAPENS displays inconsistent performance. For some datasets it outperforms two of our baselines while for others it achieves scores lower than the best base learner's. This is likely due to the inherent variance introduced into DAP by sampling random models to evaluate on during optimization. For example, it is possible that some candidate augmentation pipelines have a high fitness score because they happen to work well on a particular subset of the model library, but they are not a good fit when applied to other models. Furthermore, it could be that DAP converges towards pipelines that do not generalize well across the model library or that the pipeline keeps evolving in a way that suits the needs for the subset of models at the particular generation, which does not guarantee good ensemble behavior. The fitness landscape keeps changing from generation to generation, so it is difficult to take steps towards good solutions. This could potentially be improved on by using all models from the library for evaluation at every generation, instead of a subset.

For the ASMPT-provided data, continuous training was also conducted and the obtained results are a few percent points higher than the scores of our algorithms. This is almost expected, given that the continuous training is essentially the same as joint training in the used setting, and joint training is sometimes seen as an upper bound to the achievable performance [33]. However, the proposed approaches have a few benefits over (re)training: the previously mentioned interpretability, no need to access the training data, needing inference only, and not requiring further manual intervention. Removing training from the equation could be advantageous in situations similar to ASMPT's, where the customers might not want to share samples of their data. Providing the customer with our optimization algorithms would also further reduce time expenses. Additionally, we have a guarantee that the model will not learn unwanted behaviours.

Overall, the combination of ensembling and DA pipelines appears promising for tackling out-of-distribution data at test-time. It is important to note, however, that the results of ENSDAP(gen), DAPENS, ENSEC and DAP are not significantly different. We cannot claim with statistical confidence that one algorithm truly outperforms the others. According to the Wilcoxon test on Anon #1 data, we can only with certainty say that ENSDAP(ind) underperforms the rest. Also, the hyperparameters of the proposed algorithms are not fully optimized. A full hyperparameter search can potentially result in a boost in performance or significant differences. Investigating whether this is the case is also left for future work.

Computational efficiency. The proposed methods are relatively efficient, with the best performing one, ENSDAP(gen), achieving 2-3% MIoU improvement in the ASMPT-provided data in about 8 hours. It is important to note that, especially in the case of segmentation, ENSEC can be rather time-consuming given that it evaluates a number of models for every candidate solution in every generation. Storing the models' predictions and loading those instead of constantly reevaluating the models has lead to a great speed up and improved efficiency. The small optimization set also contributes to increased computational efficiency. This is both a conscious decision (in the case of CIFAR- and Pascal-derived data) with the aim of fast computation and a restriction imposed by data (in the case of ASMPT) as often in practice we only have a limited amount of samples available. We believe that achieving such improvements over the baselines with only a handful of samples is a very promising outcome when it comes to real world applications. Although continuous training leads to higher performance boosts in roughly the same time frame, we believe that there are many ways to reduce the run time per generation of the proposed algorithms, given that optimizing their speed was not a main focus of the thesis. For instance, by cropping the images or downscaling them. A parallelized version of the algorithms would also lead to a great speedup and is possible since the evaluations of the different candidate solutions are independent of each other.

Inference time efficiency is crucial, especially for real-world applications. The sparseness of the optimized ensembles leads to solutions that are between 2 to 5 times faster than an ensemble made of all models from the library. They are also about 2-3 times slower than using a single model – but better in predictive performance, which in some applications can be a worthwhile trade off. The augmentation pipeline unfortunately adds further overhead to the image processing, but there are ways to potentially reduce the added time by augmentations. For instance, the resizing factor has a role to play in the speed of the pixel-level segmentation; The more pixels there are the more processing and outputs the model must produce, so limiting the image size and thus the resizing factor is one possible way to speed up computation. Furthermore, an additional step of distilling the distributions of the ensemble predictions into a single model [39] or of retraining a model with the identified pipeline can be introduced. Lastly, as mentioned previously, for the ASMPT data a different backend for transformations is used leading to a larger latency due to the augmentations. There is a lot of opportunity to reduce that overhead, for example by utilizing the in-build PyTorch image transformation functions.

Datasets. As already mentioned, we strive to use as few data samples for optimization as possible. However, there is a stark difference between the 6-20 samples used for the ASMPT-provided data and the choice to utilize 150 samples for CIFAR-derived data in DAP. The reason for this is that the segmentation models identify wires from beginning to end, i.e., all possible classes for the dataset occur in every single image. On the other hand, for CIFAR-100 one sample corresponds to a single class, and taking only a few images for optimization will most definitely lead to the model or ensemble overfitting to a particular set of classes and performing poorly once images of a new class are encountered. In preliminary tests where DAP was run with only 30 samples, poor results were obtained, which improved drastically once the optimization set size was increased. 150 samples does not guarantee all 100 classes are included in the set, but it does decrease the chance of overfitting. It is worthwhile to explore the performance of the different algorithms for different amounts of optimization

data, and possibly optimize that as another hyperparameter. For ENSEC, using a smaller number of samples (30, in our case) is still possible and leads to improvements because the task is essentially to prune bad or redundant models from the library and overfitting is not a risk.

One limitation in relation to the data in our work is that for the CIFAR- and Pascal-derived datasets, the data shift is artificial and quite simplified. It is likely not representative of the data encountered in the wild. For example, the adjusted brightness is a data shift in just one direction, while in the real world there can be quite diverse and complex differences in data distributions which are not directly reversible. One future step is to conduct further tests with out-of-distribution data coming from real applications, similarly to the ASMPT-provided data. This includes exploring even more tasks in computer vision, and also outside of that domain. Furthermore, in follow-up work, the proposed approaches should be also evaluated on datasets dedicated to the data shift problem so that a direct comparison to related work can be made. Some such frequently used benchmarks are DomainBed [24], CIFAR-10.1 and CIFAR-10.2 [37].

Design choices. In the implementation of ENSEC, an alternative approach would have been using bitstrings instead of weight vectors, thus only determining model presence in the ensemble and having all models contribute equally for the collective prediction. The bitstring solution representation has a smaller search space and thus can result in faster convergence, but we decided against using it because of the more flexible weight representation. Optimizing a weight vector provides a more fine-grained control over the different models' contributions, which for some cases could be desirable. Also for ENSEC, to conduct crossover parent candidates are sampled based on fitness scores, i.e., solutions with a higher fitness have a higher chance of being chosen for reproduction. This seemed to work better, or at least lead to faster convergence, in comparison to random sampling, but there are other options as well. For instance, one could include candidate age in the sampling, or conduct tournament selection instead.

When it comes to DAP, the set of available transformation functions and their order is fixed, with the algorithm only optimizing their parameters. This choice was made with efficiency in mind as it narrows down the search space in comparison to also having to tune the presence and order of the transformations. In future work, however, it would be interesting to explore alternative implementations.

Another design choice that had to be taken was in regards to DAPENS and the generation of a general pipeline. We considered an alternative option, where the DAP is run with the best base learner multiple times with different seeds and then the resulting transformation parameters are averaged out to produce a single value. This approach is more time efficient given that the candidate solutions are evaluated on a single model only instead of multiple. However, the idea was deemed suboptimal for two main reasons: Firstly, a bias in the augmentation pipeline towards this best learner's predictions might be introduced, thus making it ineffective for the other models in the library. Secondly, taking the average of the transformation parameters might lead to unpredictable results. For example, one seeds produces a positive parameter value and another produces a negative value and they both happen to work well with the problem, their average will be around 0 which is not guaranteed to work well too. Our choice of randomly picking different networks from the library for evaluation is likely the cause for

the inconsistent behaviour of DAPENS, however. One possible solution that fixes the fitness landscape in place is utilizing the whole library of models for the candidate evaluation.

Lastly, evolution was used as a black-box optimizer as it is a gradient-free method very flexible when it comes to fitness functions and less likely to get stuck in a local optima. However, there are other methods that could be utilized, such as Bayesian optimization. Future work can explore such alternatives to determine what is the most suitable and time efficient approach.

Unexplored directions. There are also a number of approaches of combining ensembling and data augmentation that are unfortunately not explored in this thesis due to time and compute constraints. The three sequential combinations examine are the ones deemed most promising, but a few more methods were considered, which merit investigation in future work. One idea is to run DAP first, followed by ENSEC, but instead of creating a general pipeline for the whole library as in DAPENS, a pipeline for each model is optimized. Then when ENSEC is run, every model receives data after it is modified with its individual pipeline. Another direction is to fully merge ENSEC and DAP, so that the ensemble and pipeline optimization does not happen sequentially but in parallel. Essentially, this would consist in designing an EA in which the candidate solutions are represented by both a weight vector and a transformation parameter vector. At each evolutionary step, we would then either tweak both or alternate between them. This algorithm would require a number of design choices (type of mutations, simultaneous optimization or alternating, and if alternating – how often to switch between the two problems, etc.) and a longer run time given the larger search space, which is why it is left for future work, given our time constraints. We believe it is interesting to investigate, however, especially while considering ways of optimizing the run time, such as restricting the number of models the ensemble can comprise and the number of transformations to be added to the pipeline, or discretizing the possible parameter and weight values.

Chapter 8

Conclusion

In this thesis, we look at the problem of data shift which leads to performance drops in the wild when models encounter unseen samples, and try to tackle it fully at test-time in order to reduce the amount of resources and computation required and remove the need for retraining. For that, we propose combining a technique often used for the task, ensembling, and a novel test-time data augmentation adaptation method. Evolutionary algorithms for ensemble selection and DA pipeline optimization are implemented and three sequential combinations of the two are proposed. Although there is no significant difference between the majority of the investigated methods, they offer consistent boosts in performance over using a single model or a non-optimized ensemble for a varied set of datasets. However, these approaches still fall short to the hard-to-beat state-of-the-art of continual learning. Nonetheless, given that our proposed algorithms do achieve a boost in scores, are computationally efficient and do not require access to the train set, we believe that this research line should be explored further as it holds promise for an efficient increase in model generalization. There are also a lot of other avenues to be investigated in the future, such as different means of combining ensembling and DA, using another optimization technique, and exploring techniques to speed up the inference time of our solutions.

Acknowledgements

This thesis is conducted in collaboration with the AI & Vision Team of the ASMPT Center of Competency situated within the ASMPT branch ALSI, located in Beuningen, The Netherlands. We want to thank Rob van Gastel and Faysal Boughorbel for their external supervision.

Bibliography

[1] Abad, M., Casas-Roma, J., and Prados, F. Generalizable disease detection using model ensemble on chest x-ray images. *Scientific Reports 14*, 1 (2024), 5890.

[2] Abe, T., Buchanan, E. K., Pleiss, G., Zemel, R., and Cunningham, J. P. Deep ensembles work, but are they necessary? *Advances in Neural Information Processing Systems 35* (2022), 33646–33660.

[3] Adhikarla, E., Zhang, K., Yu, J., Sun, L., Nicholson, J., and Davison, B. D. Robust computer vision in an ever-changing world: A survey of techniques for tackling distribution shifts. *arXiv preprint arXiv:2312.01540* (2023).

[4] Ahn, C. W., and Ramakrishna, R. S. Elitism-based compact genetic algorithms. *IEEE Transactions on Evolutionary Computation 7*, 4 (2003), 367–385.

[5] Ashukha, A., Lyzhov, A., Molchanov, D., and Vetrov, D. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. *arXiv preprint arXiv:2002.06470* (2020).

[6] Bakurov, I., Castelli, M., Gau, O., Fontanella, F., and Vanneschi, L. Genetic programming for stacked generalization. *Swarm and Evolutionary Computation 65* (2021), 100913.

[7] Bhowan, U., Johnston, M., Zhang, M., and Yao, X. Reusing genetic programming for ensemble selection in classification of unbalanced data. *IEEE Transactions on Evolutionary Computation 18*, 6 (2013), 893–908.

[8] Breiman, L. Bagging predictors. *Machine learning 24* (1996), 123–140.

[9] Cagnini, H. E., Basgalupp, M. P., and Barros, R. C. Increasing boosting effectiveness with estimation of distribution algorithms. In *2018 IEEE Congress on Evolutionary Computation (CEC)* (2018), IEEE, pp. 1–8.

[10] Cagnini, H. E., Dôres, S. C. D., Freitas, A. A., and Barros, R. C. A survey of evolutionary algorithms for supervised ensemble learning. *The Knowledge Engineering Review 38* (2023), e1.

[11] Caruana, R., Niculescu-Mizil, A., Crew, G., and Ksikes, A. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning* (2004), p. 18.

[12] Chandra, A., and Yao, X. Ensemble learning using multi-objective evolutionary algorithms. *Journal of Mathematical Modelling and Algorithms 5* (2006), 417–445.

[13] Chawla, N. V., and Sylvester, J. Exploiting diversity in ensembles: Improving the performance on unbalanced datasets. In *International Workshop on Multiple Classifier Systems* (2007), Springer, pp. 397–406.

[14] Chen, Y., and Zhao, Y. A novel ensemble of classifiers for microarray data classification. *Applied soft computing 8*, 4 (2008), 1664–1669.

[15] Correia, J., Martins, T., and Machado, P. Evolutionary data augmentation in deep face detection. In *Proceedings of the genetic and evolutionary computation conference companion* (2019), pp. 163–164.

[16] Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 113–123.

[17] Dang, T., Nguyen, T. T., McCall, J., Elyan, E., and Moreno-García, C. F. Two-layer ensemble of deep learning models for medical image segmentation. *Cognitive Computation* (2024), 1–20.

[18] Dietterich, T. G. Ensemble methods in machine learning. In *International workshop on multiple classifier systems* (2000), Springer, pp. 1–15.

[19] Dong, Z., Xu, K., Yang, Y., Bao, H., Xu, W., and Lau, R. W. Location-aware single image reflection removal. In *Proceedings of the IEEE/CVF international conference on computer vision* (2021), pp. 5017–5026.

[20] Farahani, A., Voghoei, S., Rasheed, K., and Arabnia, H. R. A brief review of domain adaptation. *Advances in data science and information engineering: proceedings from ICDATA 2020 and IKE 2020* (2021), 877–894.

[21] Fix, E., and Hodges Jr, J. Significance probabilities of the wilcoxon test. *The Annals of Mathematical Statistics* (1955), 301–312.

[22] Freund, Y., Schapire, R. E., et al. Experiments with a new boosting algorithm. In *icml* (1996), vol. 96, Citeseer, pp. 148–156.

[23] Georgescu, M.-I., Ionescu, R. T., and Miron, A. I. Diversity-promoting ensemble for medical image segmentation. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing* (2023), pp. 599–606.

[24] Gulrajani, I., and Lopez-Paz, D. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434* (2020).

[25] Guo, Y., Liu, Y., Georgiou, T., and Lew, M. S. A review of semantic segmentation using deep neural networks. *International journal of multimedia information retrieval 7* (2018), 87–93.

[26] Hao, S., Zhou, Y., and Guo, Y. A brief survey on semantic segmentation with deep learning. *Neurocomputing 406* (2020), 302–321.

[27] Hong, Y., Pan, H., Sun, W., and Jia, Y. Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes. *arXiv preprint arXiv:2101.06085* (2021).

[28] Kamnitsas, K., Bai, W., Ferrante, E., McDonagh, S., Sinclair, M., Pawlowski, N., Rajchl, M., Lee, M., Kainz, B., Rueckert, D., et al. Ensembles of multiple models and architectures for robust brain tumour segmentation. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: Third International Workshop, BrainLes 2017, Held in Conjunction with MICCAI 2017, Quebec City, QC, Canada, September 14, 2017, Revised Selected Papers 3* (2018), Springer, pp. 450–462.

[29] Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images.

[30] Krogh, A., and Vedelsby, J. Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems 7* (1994).

[31] Lee, C. S., and Lee, A. Y. Clinical applications of continual learning machine learning. *The Lancet Digital Health 2*, 6 (2020), e279–e281.

[32] Li, L., Stolkin, R., Jiao, L., Liu, F., and Wang, S. A compressed sensing approach for efficient ensemble learning. *Pattern recognition 47*, 10 (2014), 3451–3465.

[33] Li, Z., and Hoiem, D. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence 40*, 12 (2017), 2935–2947.

[34] Ling, Z., Zhang, A., Ma, D., Shi, Y., and Wen, H. Deep siamese semantic segmentation network for pcb welding defect detection. *IEEE Transactions on Instrumentation and Measurement 71* (2022), 1–11.

[35] Liu, B. Lifelong machine learning: a paradigm for continuous learning. *Frontiers of Computer Science 11*, 3 (2017), 359–361.

[36] Louizos, C., Welling, M., and Kingma, D. P. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312* (2017).

[37] Lu, S., Nott, B., Olson, A., Todeschini, A., Vahabi, H., Carmon, Y., and Schmidt, L. Harder or different? a closer look at distribution shift in dataset reproduction. In *ICML Workshop on Uncertainty and Robustness in Deep Learning* (2020), vol. 5, p. 15.

[38] Lyksborg, M., Puonti, O., Agn, M., and Larsen, R. An ensemble of 2d convolutional neural networks for tumor segmentation. In *Image Analysis: 19th Scandinavian Conference, SCIA 2015, Copenhagen, Denmark, June 15-17, 2015. Proceedings 19* (2015), Springer, pp. 201–211.

[39] Malinin, A., Mlodozeniec, B., and Gales, M. Ensemble distribution distillation. *arXiv preprint arXiv:1905.00076* (2019).

[40] Martinez-Munoz, G., Hernández-Lobato, D., and Suárez, A. An analysis of ensemble pruning techniques based on ordered aggregation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31*, 2 (2008), 245–259.

[41] Masisi, L., Nelwamondo, F. V., and Marwala, T. The effect of structural diversity of an ensemble of classifiers on classification accuracy. *arXiv preprint arXiv:0804.4741* (2008).

[42] Mendes-Moreira, J., Soares, C., Jorge, A. M., and Sousa, J. F. D. Ensemble approaches for regression: A survey. *Acm computing surveys (csur) 45*, 1 (2012), 1–40.

[43] Nado, Z., Padhy, S., Sculley, D., D'Amour, A., Lakshminarayanan, B., and Snoek, J. Evaluating prediction-time batch normalization for robustness under covariate shift. *arXiv preprint arXiv:2006.10963* (2020).

[44] Nanni, L., Lumini, A., Loreggia, A., Formaggio, A., and Cuza, D. An empirical study on ensemble of segmentation approaches. *Signals 3*, 2 (2022), 341–358.

[45] Nigam, I., Huang, C., and Ramanan, D. Ensemble knowledge transfer for semantic segmentation. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2018), IEEE, pp. 1499–1508.

[46] Niu, S., Wu, J., Zhang, Y., Chen, Y., Zheng, S., Zhao, P., and Tan, M. Efficient test-time model adaptation without forgetting. In *International conference on machine learning* (2022), PMLR, pp. 16888–16905.

[47] Niu, S., Wu, J., Zhang, Y., Wen, Z., Chen, Y., Zhao, P., and Tan, M. Towards stable test-time adaptation in dynamic wild world. *arXiv preprint arXiv:2302.12400* (2023).

[48] Ogwok, D., and Ehlers, E. M. Jaccard index in ensemble image segmentation: An approach. In *Proceedings of the 2022 5th International Conference on Computational Intelligence and Intelligent Systems* (2022), pp. 9–14.

[49] Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems 32* (2019).

[50] Pagliardini, M., Jaggi, M., Fleuret, F., and Karimireddy, S. P. Agree to disagree: Diversity through disagreement for better transferability. *arXiv preprint arXiv:2202.04414* (2022).

[51] Parhizkar, E., and Abadi, M. Beeowa: A novel approach based on abc algorithm and induced owa operators for constructing one-class classifier ensembles. *Neurocomputing 166* (2015), 367–381.

[52] Pereira, S., Correia, J., and Machado, P. Evolving data augmentation strategies. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)* (2022), Springer, pp. 337–351.

[53] Poli, R., and Langdon, W. B. On the ability to search the space of programs of standard, one-point and uniform crossover in genetic programming.

[54] Qiang, F., Shang-Xu, H., and Sheng-Ying, Z. Clustering-based selective neural network ensemble. *Journal of Zhejiang University-Science A 6* (2005), 387–392.

[55] Qiao, F., and Peng, X. Uncertainty-guided model generalization to unseen domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2021), pp. 6790–6800.

[56] Qiao, F., and Peng, X. Ensemble pruning for out-of-distribution generalization. In *Forty-first International Conference on Machine Learning* (2024).

[57] Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. *Dataset shift in machine learning*. Mit Press, 2022.

[58] Rame, A., Kirchmeyer, M., Rahier, T., Rakotomamonjy, A., Gallinari, P., and Cord, M. Diverse weight averaging for out-of-distribution generalization. *Advances in Neural Information Processing Systems 35* (2022), 10821–10836.

[59] Robey, A., Pappas, G. J., and Hassani, H. Model-based domain generalization. *Advances in Neural Information Processing Systems 34* (2021), 20210–20229.

[60] Saha, M., and Chakraborty, C. Her2net: A deep framework for semantic segmentation and classification of cell membranes and nuclei in breast cancer evaluation. *IEEE Transactions on Image Processing 27*, 5 (2018), 2189–2200.

[61] Shanmugam, D., Blalock, D., Balakrishnan, G., and Guttag, J. Better aggregation in test-time augmentation. In *Proceedings of the IEEE/CVF international conference on computer vision* (2021), pp. 1214–1223.

[62] Shenouda, M., Whitney, H. M., Giger, M. L., and Armato III, S. G. Impact of retraining and data partitions on the generalizability of a deep learning model in the task of covid-19 classification on chest radiographs. *Journal of Medical Imaging 11*, 6 (2024), 064503–064503.

[63] Sivanandam, S., and Deepa, S. Genetic algorithms. In *Introduction to genetic algorithms*. Springer, 2008, pp. 15–37.

[64] Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A., and Hardt, M. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning* (2020), PMLR, pp. 9229–9248.

[65] Therrien, R., and Doyle, S. Role of training data variability on classifier performance and generalizability. In *Medical Imaging 2018: Digital Pathology* (2018), vol. 10581, SPIE, pp. 58–70.

[66] Tomar, D., Vray, G. M. G., Bozorgtabar, B., and Thiran, J.-P. Opttta: Learnable test-time augmentation for source-free medical image segmentation under domain shift. In *Proceedings of Machine Learning Research, Volume 172: International Conference on Medical Imaging with Deep Learning, 6-8 July 2022, Zurich, Switzerland* (2022), PMLR, pp. 1192–1217.

[67] Van de Ven, G. M., and Tolias, A. S. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734* (2019).

[68] Wang, Z., Luo, Y., Zheng, L., Chen, Z., Wang, S., and Huang, Z. In search of lost online test-time adaptation: A survey. *International Journal of Computer Vision* (2024), 1–34.

[69] Yang, J., Soltan, A. A., and Clifton, D. A. Machine learning generalizability across healthcare settings: insights from multi-site covid-19 screening. *NPJ digital medicine 5*, 1 (2022), 69.

[70] Yao, X., and Liu, Y. Making use of population information in evolutionary artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 28*, 3 (1998), 417–425.

[71] Yurtsever, E., Lambert, J., Carballo, A., and Takeda, K. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access 8* (2020), 58443–58469.

[72] Zaidi, S., Zela, A., Elsken, T., Holmes, C. C., Hutter, F., and Teh, Y. Neural ensemble search for uncertainty estimation and dataset shift. *Advances in Neural Information Processing Systems 34* (2021), 7898–7911.

[73] Zhang, M., Levine, S., and Finn, C. Memo: Test time robustness via adaptation and augmentation. *Advances in neural information processing systems 35* (2022), 38629–38642.

[74] Zhao, J., Jiao, L., Xia, S., Fernandes, V. B., Yevseyeva, I., Zhou, Y., and Emmerich, M. T. Multiobjective sparse ensemble learning by means of evolutionary algorithms. *Decision Support Systems 111* (2018), 86–100.

[75] Zhou, Z.-H., Wu, J., and Tang, W. Ensembling neural networks: many could be better than all. *Artificial intelligence 137*, 1-2 (2002), 239–263.