Learning About the Learning Process: From Active Querying to Fine-tuning

Kunkun Pang

Doctor of Philosophy

Institute of Perception, Action and Behaviour

School of Informatics

University of Edinburgh

2019

Abstract

The majority of research on academic machine learning addresses the core model fitting part of the machine learning workflow. However, prior to model fitting, data collection and annotation is an important step; and subsequently to this, knowledge transfer to different but related problems is also important. Recently, the core model fitting step in this workflow has been upgraded using learning-to-learn methodologies, where learning algorithms are applied to improve the fitting algorithm itself in terms of computation or data efficiency. However, algorithms for data collection and knowledge transfer are still commonly hand-engineered. In this doctoral thesis, we upgrade the pre-and post-processing steps of the machine learning pipeline with the learning-to-learn paradigm.

We first present novel learning-to-learn approaches that improve the algorithms for this pre-processing step in terms of label efficiency. The inefficiency of data annotation is a common issue in the field: To fit the desired model, a large amount of data is usually collected and annotated, much of which is useless. Active learning aims to address this by selecting the most suitable data for annotation. Since conventional active learning algorithms are hand-engineered and heuristically designed for a specific problem, they typically cannot be adapted across nor even within datasets. The data efficiency of active learning can be improved either by online learning active learning within a specific problem, or by transferring active learning knowledge between related problems. We begin by investigating the framework of leaning active learning online, which learns to select the best criteria for a particular dataset as queries are made. It enables online adaptation, along with the state of the model and dataset changes, while guaranteeing performance. Subsequently, we upgrade the previous framework to a data-driven learning-based approach by learning a transferable active-learning policy end-to-end. The framework is thus capable of directly optimising the accuracy of the underlying classifier, and can adapt to the statistics of any given dataset. More importantly, the learned active-learning policy is domain agnostic and generalises to new learning problems.

We next turn to knowledge transfer from a well-learned problem to a novel target problem. We develop a new learning-to-learn technique to improve the effectiveness and efficiency of fine-tuning-based transfer learning. Conventional transfer learning approaches are heuristic: Most commonly, small learning-rate stochastic gradient descent starting from the source model as a condition, and keeping the architecture constant. However, the typical transfer learning pipeline transfers learning from a general model or dataset to a more specific one. Thus, we propose a transfer learning algorithm for neural networks, which simultaneously prune the size of the target networks architecture and updates its weights. This enables the model complexity to be reduced, as training iterations increase, and both efficiency and efficacy are improved compared to conventional fine-tuning knowledge transfer.

Lay Summary

Most research on machine learning focuses on designing algorithms for learning to solve real-world applications. Nevertheless, the efficiency and efficacy of machine learning techniques are not only determined by the effectiveness of learning algorithm itself but also depend on the surrounding processes: (i) collecting and annotating data before applying the learning algorithm or (ii) subsequently to implementing the learning algorithm, reusing the knowledge to help solve different but related problems. However, annotating useless data or transferring unrelated knowledge reduces efficiency. This motivates research to enhance the surrounding processes of collecting and annotating the most related data or selecting the most relevant knowledge to transfer to other problems. However, most those methods are hand-designed, which limits their efficacy.

In this thesis, we aim to improve these surrounding processes by applying machine learning to data annotation and knowledge transfer. We first train the machine to select the best hand-designed algorithm for data annotation acquisition. We next enable the machine to learn its own algorithm to predict which is the most useful data to acquire annotation for. Finally, we train the machine to remove irrelevant data when transferring knowledge to a new problem.

Acknowledgements

I would like to thank my supervisor, Timothy Hospedales, for his help, support and guidance throughout my PhD. I would also like to thank Yang Wu with the Nara Institute of Science and Technology for giving me the invaluable opportunity to visit and work with him. I want to especially thank Mingzhi Dong for all of his support and help throughout my studies. I also want to thank Yongxin Yang, Chenyang Zhao, and Lucy Liu for providing suggestions for my works and this thesis. Finally, I would like to thank my family for all the years of support and love.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Kunkun Pang*)

Table of Contents

List of Figures

List of Tables

List of Acronyms

AL	Active Learning
ALBL	Active Learning by Learning
BMDR	Batch Mode Active Learning
BvSB	Best versus Second Best Criterion
COMB	Combination of Active Learning Online
CEM	Classification Entropy Maximisation
DE	DEnsity sampling
DEAL	Dynamic Ensemble Active Learning
DG	Domain Generalisation
DRL	Deep Reinforcement Learning
DUAL	Dual Strategy Active Learning
DFF	Distance Furthest First
DNN	Deep Neural Network
EA	Evolutionary Algorithm
EXP3	Exponential-weight algorithm for Exploration and Exploitation
EXP4	Exponential-weight algorithm for Exploration and Exploitation with Expert advice
EXP4.P	Exponential-weight algorithm for Exploration and Exploitation with Expert advice and bounds on the weak regret that hold with high Probability
GMM	Gaussian Mixture Models
IWA	Important Weighted Accuracy
LAL	Learning Active Learning
LALO	Learning Active Learning Online

LinUCB	Linear Upper Confidence Bound
LTAL	Learning Transferable Active Learning
LFT	Learning to Fine-tune
LSA	Linear Strategy Aggregation
LOO	Leave One Out
MAB	Multi-Arm Bandit
MAML	Model-Agnostic Meta Learning
MAB	Model Compression
MDP	Markov Decision Process
MLP-GAL	Meta Learned Policy for General Active Learning
NAS	Network Architecture Search
QBB	Query By Bagging
QUIRE	Query Informative and Representative Examples
RAND	Random Sampling
REXP3	Restarting Exponential-weight algorithm for Exploration and Exploitation
REXP4	Restarting Exponential-weight algorithm for Exploration and Exploitation with Expert advice
REINFORCE	REward Increment Non-negative Factor times Offset Reinforcement times Characteristic Eligibility algorithm
RL	Reinforcement Learning
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
UCB	Upper Confidence Bound
US	Uncertainty Sampling

List of Symbols

$\boldsymbol{\xi}$	Probability vector encoding the preference over arms (instances)
$\boldsymbol{x}$	Data point/instance
Δ_T	The time epoch for REXP4 to learn
ℓ	Loss function of supervised learning model and a data point
$\mathcal{D}$	Dataset for supervised learning
$\mathcal{D}^{te}$	Test dataset for evaluating the supervised learning model
$\mathcal{D}^{tr}$	Training dataset for training the supervised learning model
$\mathcal{F}$	Meta objective function
$\mathcal{L}$	Labelled set
$\mathcal{R}$	Regularisation term of the objective function
$\mathcal{U}$	Unlabelled set
$\mathfrak{D}$	Meta dataset for meta learning
$\mathfrak{D}^{final-test}$	Meta final test dataset which is held-out set of the meta dataset $\mathfrak{D}$
$\mathfrak{D}^{final-train}$	Meta final training dataset which is held-out set of the meta dataset $\mathfrak{D}$
$\mathfrak{D}^{final}$	Meta final set which is held-out set of the meta dataset $\mathfrak{D}$ to evaluate the learned supervised learner's performance
$\mathfrak{D}^{te}$	Meta test dataset for the supervised learner's parameter Θ
$\mathfrak{D}^{tr}$	Meta training dataset for training the supervised learner's parameter Θ
μ	Expected reward of an arm
ω	Model's parameters
π	The acting policy
a	Action
$b(\cdot)$	Learner takes dataset as input and output a desired model or query point
$b(\cdot;\Theta)$	Trainable learner: add parameter Θ to train the learner

$b_{\text{params}(\cdot;\Theta)}$	Trainable learner to synthesise the parameter of a model
$b_{AL}(\cdot)$	Active learner takes set and classifier as input and predict the most useful instance
$b_{LALO}(\cdot;\Theta)$	Online trainable active learner
$b_{LAL}(\cdot;\Theta)$	Trainable active learner
$b_{LFT}(\cdot)$	Trainable fine-tuning (SGD) learner
$b_{LTAL}(\cdot;\Theta)$	Transferable trainable learning active learner
$b_{MAML}(\cdot;\Theta)$	Trainable learner of model-agnostic meta-learning (MAML) approach
$b_{SGD}(\cdot)$	Stochastic gradient descent update learner
c	Supervised learning model's output
d	Data point's dimension
f	Supervised learning model
$g(\cdot)$	Meta learner aims to optimise the learner so that the learner could produce a better model
L	Loss function of supervised learning model and datasets
M	Number of data points
q	The queried data point
s	State
T	Time horizon
t	Time step
y	Expected reward of an expert
$\mathbf{Var}(\cdot)$	Variance

List of My Publications

- Kunkun Pang, Mingzhi Dong, Yang Wu, and Timothy Hospedales. Dynamic Ensemble Active Learning: A Non-Stationary Bandit with Expert Advice. In *International Conference on Pattern Recognition* (ICPR) 2018. Best student paper awarded.
- Kunkun Pang, Mingzhi Dong, Yang Wu, and Timothy Hospedales. Meta-learning transferable active learning policies by deep reinforcement learning. In *International Conference on Machine Learning, International Workshop on Automatic Machine Learning* (ICML AutoML) 2018. Best paper awarded.
- Mingzhi Dong, Kunkun Pang, Yang Wu, Jinghao Xue, Timothy Hospedales, and Tsukasa Ogasawara. Transferring CNNs to Multi-Insance Multi-Label Classification on Small Datasets. In *International Conference on Image Processing* (ICIP) 2017.

Chapter 1

Introduction

1.1 Machine Learning

As a core sub-field of artificial intelligence, machine learning has emerged to answer the question *'Can a machine learn?'*. A machine is said to learn if its performance, with respect to some tasks and the associated performance measures, improves with data and experience [Mitchell, 1997]. Machine learning enables machines to learn from data and experience without being explicitly programmed [Samuel, 1959]. The science of machine learning draws on concepts from diverse disciplines, including computer science, statistics, mathematics, etc. [Bishop, 2006; Barber, 2012; Murphy, 2012]. Researchers have categorised conventional machine learning into three paradigms.

Supervised Learning In this paradigm, data is provided with a pair of feature and true labels. The aim of supervised learning is to distinguish between features and learn a mapping from feature to true label.

Unsupervised Learning Unlike a supervised learning setting, in which a machine learns the relation between features and labels, unsupervised learning aims to learn from features alone (i.e. without labels) and discover latent representation of data.

Reinforcement Learning In reinforcement learning, an agent can only acquire a reward by interacting with its environment. The agent is expected to learn how to act in the environment and collect as many rewards as possible [Sutton and Barto, 1998].

In this thesis, we study the supervised learning setting. Although the majority of previous research has focused on the core learning problem, the learning algorithm does not exist in isolation. It is necessary to examine the surrounding processes that are necessary for it to be effective, which are often exceptionally important in practice: namely, data collection prior to learning and knowledge transfer after learning.

1.1.1 Data Collection

Data collection is a prerequisite of the general learning problem. According to the previously noted definition of supervised learning, all of the data-driven learning approaches can only be undertaken once data is provided. Generally, in the data collection step, we establish a dataset by collecting examples (features) and then annotating them with labels. In many applications, examples can be collected automatically and relatively inexpensively (by e.g. downloading images from the internet). The labels, however, require time and are expensive to provide. Thus, the dataset size in supervised learning is often governed by an annotation budget how many examples can be afforded to obtain manual annotation. Once a dataset is established, the collected data is provided to the core learning algorithm.

Learning from poor quality data leads to weak performance [Zhu and Wu, 2004; Zhu et al., 2004]. A lack of quality data can be considered from two perspectives: data featurisation and instance selection. Data featurisation refers to the choice of description for each data point. If featurisation is poor, some attributes may be noisy or redundant. As a result, the learning algorithm must work harder to distinguish the signal from the noise, which may lead to overfitting and diminished performance. To address this, data cleansing and feature selection are proposed to correct the corrupted features or select only the most important features [Zhu and Wu, 2004; Guyon and Elisseeff, 2003]. With regard to instance selection, it refers to the choice of a subset of the entire dataset. Selecting inferior data points may not be useful for training, since different instances have the diverse utility of the learning algorithm. For example, an image recogniser may already know how to recognise easy examples of a category, and therefore observing more annotated examples of those is unhelpful. In addition, useless examples risk biasing the training, which may further diminish performance. However, observing unusual or ambiguous annotated examples can improve performance. This motivates the research of active learning to estimate the usefulness of data points for annotation before actually annotating them, thus reducing annotation costs by enabling annotation to be selectively applied to only the most useful data points. One of the contributions of this thesis is to improve the estimation of data point utility by developing better active learning methods.

Active learning aims to reduce total labelling costs by carefully selecting the informative points to label. The active learning process queries the entire dataset and returns the most useful data to annotate, thus improving performance without requiring more effort. The research problem of active learning concerns how to determine which data points are the most desirable. Given a limited budget, it is more effective to annotate only the most informative data points. However, the most useful data points to annotate depend on the state of the underlying learning model, which changes as more data is annotated. Thus, the process of active learning needs to annotate only the most informative data point and update the model iteratively. This learning process improves data efficiency in terms of reducing the number of labels required for the base learner to reach a given level of performance.

1.1.2 Knowledge Transfer

Knowledge transfer is the process of improving learning performance on a target problem using knowledge extracted from previously known source problems. Without knowledge transfer, it would be necessary to train a model from scratch for a given task. In many real-world applications, it is expensive to collect a large amount of training data for each new task. In addition, training a complex model from scratch might cause overfitting when a new task is given with a small training sample size. The act of dataset rebuilding, in addition to model retraining, limit the practical feasibility of training complex non-convex models, such as neural networks, from scratch. Thus, a common approach to alleviating these issues is to adapt and reuse previous shareable knowledge to a new task. Transfer learning refers to the transfer of knowledge from the source task to the target task [Pan and Yang, 2010]. Knowledge can be transferred across tasks (such as recognising different object categories) or domains (such as different camera types or lighting conditions). The success of transfer learning generally depends on the relatedness of the tasks and the efficacy of the transfer learning algorithm [Goodfellow et al., 2016].

There are numerous approaches to transfer learning, including feature-based [Evgeniou and Pontil, 2004], subspace-based [Kumar and Daumé, 2012], Bayesian-prior-based [Fei-Fei et al., 2006], and regularisation-based [Yang et al., 2007]. However, by far the most popular approach in recent deep learning era has been the fine-tuning-based transfer learning. Fine-tuning is typically applied to transfer knowledge in situations where there is a target task with sparse data, and a related available source task with plentiful data.

In fine-tuning, a deep network is pre-trained on a source problem, and then the weights of this task are transferred and used as the initial condition for learning in the target task. Given the local gradient-based optimisation used in a deep neural network (DNN), this means that the target problem can be learned: (i) more quickly (since it starts closer to a good optimum) and (ii) with fewer data (since it is less likely to run into a terrible local minima when starting near a good solution). Since the small learning rate of gradient-descent learning in a non-convex function, such as DNN, does not move too far from the initial condition, fine-tuning can be seen as the most related to the regularisation of or prior-based approaches to transfer learning. However, the standard stochastic gradient (SGD) based learning only updates the weights for target problems. We provide a learning-to-learn generalisation toward this approach that undertakes learning to update both the architecture and weights of the DNN for target problems.

1.2 Learning to Learn

In conventional machine learning approaches, a model improves with data and experience by following a particular learning algorithm. However, the learning algorithm itself is a product of human engineering; it is a pre-programmed and fixed piece of computer code. We aim to improve this by allowing the learning algorithm itself to improve with experience, which is known as the learning-to-learn paradigm. To explain learning-to-learn, it is useful to define the concepts of 'model' and 'learner'. Both the model and the learner can be considered as two unknown functions that can be improved with experience from different perspectives. From the perspective of the model, its job is to solve the actual task (for example, dog recognition task). It inputs a datum (e.g. an animal image) and produces the estimated label (e.g. the animal is a dog) as its output. From a learner perspective, its job is to fit the model. Here, the learner corresponds to the learning algorithm. It inputs a dataset and emits the model that is capable of predicting labels from features in that dataset. It may also operate iteratively by inputting the dataset and old model, and subsequently emitting an improved model. Thus, the learner acts as a function to establish parameters of the model so that the model's outputs are more accurate.

The learning process is undertaken to improve the parameters of the model, so that it can ultimately produce better labels. By analogy, the learning-to-learn (meta-learning) process aims to improve the learning algorithm, so that it is better at fitting the base model. More specifically, we define the concept of the 'meta-learner' that establishes the parameters of the learning algorithm, so that the learning algorithm can produce better models. Depending on different learning problems, the term 'better' could refer to various properties: higher accuracy, fewer label requirements, or less computational time required for learning.

1.2.1 Learning Core Learning Problem

Recent meta-learning research has successfully improved core learning problems and applications. More specifically, the meta-learner helps the learner to improve the models performance in a variety of ways, such as improving the gradient-descent update rule and the gradient-descent updating initial condition, as well as learning to synthesise a model. In regard to gradient-descent update, the learning performance of the SGD is sensitive to the chosen hyper-parameters. A significant amount of time and effort will be necessary to determine the optimal hyper-parameters. Previous meta-learning research has addressed learning these hyper-parameters, including the step size for gradient descent [Andrychowicz et al., 2016; Li and Malik, 2017]. In this case , the gradient descent learning rule can be formalised as a recurrent neural network, whose unknown parameters correspond to the step size in gradient descent. The role of the meta-learner is thus to train those parameters, such that the learning rule becomes more effective.

In gradient descent learning, the choice of initial condition can significantly impact the efficiency of DNN learning and the efficacy of the resulting solution. Thus, learning-to-learn has also been applied to find the initial condition that maximises the performance of subsequent gradient-based learning [Finn et al., 2017; Grant et al., 2018].

Another type of meta-learning, typically applied to few-shot learning, is to learn to synthesise a model using a given set of labels and data as the input [Mishra et al., 2018; Snell et al., 2017; Sung et al., 2018; Bertinetto et al., 2016]. Subsequently, the meta-learners train the meta-network (i.e. the learner) to generate parameters for a DNN that solves the target task as exemplified by provided input examples.

1.2.2 Learning-to-learn for the Full Machine Learning Pipeline

Meta-learning enhancements to the core learning process have been studied in-depth. However, the majority of prior methods for active learning and subsequent transfer learning are hand-engineered. In active learning, the conventional active learner queries data points based on the various estimators of the usefulness of the data points, such as uncertainty, representativeness, largest future error reduction, etc. The fine-tuning approach to transfer learning employs the fixed heuristic of initialising optimisation on the target problem, given weights copied from the source problem. Meta-learning extensions of these processes have not been well studied by comparison. In this thesis, we explore how to improve both active learning and transfer learning using learning-to-learn methodologies.

1.2.3 Learning Active Learning

Typical active learning algorithms query data points based on different motivations. For example, uncertainty (or margin-based sampling) queries the most ambiguous point, which is the one closest to the decision boundary [Lewis and Gale, 1994; Tong and Koller, 2002]. Expected error reduction aims to query the points to reduce future error Roy and McCallum [2001]; Hospedales et al. [2012]. Another approach is to annotate the most representative samples to ensure that the major clusters within the dataset are correctly estimated Cohn et al. [1995]; Chattopadhyay et al. [2012]; Yu et al. [2006]. In addition to these approaches, query-by-committee queries the data points based on the disagreement among a committee of classifiers Seung et al. [1992]; Abe and Mamitsuka [1998]; Loy et al. [2012]. These algorithms have been shown to improve passive learning. However, it is ambiguous which of the appealing intuitions underlying each should be preferred in principle; moreover, in practice, there is no a single algorithm that provides a clear winner in all circumstances. Thus, rather than guess which query criterion will be most beneficial to learning performance, we directly train a criterion with a meta-learner, which optimises learning performance in practice.

Instead of manually designing the active learner, the meta-learner aims to train an improved active-learner query criterion. Based on the learning-active-learning framework, the meta-learner establishes the active learning parameters (acquisition function) to select the most useful data points for manual annotation. Within this paradigm, we propose two new learning-active-learning frameworks that improve the online adaptation and transferability of the active query policy.

Online Active Learning: No single active query criterion is best suited for every dataset nor at every stage of learning (e.g. early initial modelling of the concept from the first few instances versus later refinement of the concept after many instances are acquired). Our first contribution is a dynamic, online active learner that learns to select the best active query criterion for any given dataset, and adapt it over time as the base learner improves. Specifically, in Chapter 3, we employ an adaptive ensemble approach, and learn a preference over an ensemble of base criteria (adapted for a given dataset and over time as learning proceeds). The key research challenge concerns updating the preference over the ensemble in an efficient manner since, in a learning-active-learning context, all the learning must be achieved within a few examples (active queries) to be helpful. The second challenge concerns how to balance exploitation (of good ensemble members) with exploration (to identify ensemble members that have only become useful later). To this end, we propose a non-stationary bandit algorithm to solve these challenges of online adaptive learning.

Transferable Active Learning: Though our adaptive online learning approach is effective, it does have a cold-start issue. Thus, for each dataset, active learning is conducted from scratch. Therefore, we next study a framework for learning-active-learning (LAL), in which we learn an AL query criterion from a source dataset and transfer it to a new target dataset. This has the benefit that you can directly optimise the quantity of interest for AL (base classifier accuracy in a limited budget), rather than relying on intuition about how diverse heuristics affect this quantity. However, it is important to note that we cannot learn an AL criterion on the same dataset on which we intend to test it, because either the available annotations are too few (as in the previous method) or, if we have a large query budget, then we do not need active learning. Therefore, the key research challenge revolves around how to learn an AL criterion on one annotation-rich problem that is general enough to apply directly to a new sparse-annotation problem. To this end, we propose a framework that uses the multi-domain training of a meta-network and inputs a dataset and classifier embedding to dynamically synthesise the active learning query criteria. Note that this kind of approach is analogous to the direct weight synthesis approaches in conventional learning-to-learn methods (Section 1.2.1), but it is applied to the synthesis of an active query criterion rather than a base model.

1.2.4 Learning Transfer Learning

Transfer learning based on fine-tuning is currently the defacto standard, now widely used in the deep learning era. While effectively compared to tabula rasa learning, there is still room for improving the *efficiency* of fine-tuning, and the *accuracy* of the resulting fine-tuned model, through learning-to-learn. In a learning-to-transfer paradigm, we would solve many transfer learning problems, and then update the transfer learning algorithm itself to be more efficient and effective based on this experience.

In regard to the conventional learning-to-learn paradigm, several approaches could be taken to improve the fine-tuning process through learning; for example, a fine-tuning specific update rule could be learned to replace the use of a generic hand-crafted rule. However, the extant research on the meta-learned SGD update rule could potentially be applied directly to this [Li and Malik, 2017; Andrychowicz et al., 2016]. In this thesis, we focus on a different aspect of the transfer learning process that of fine-tuning the *structure* of the target model, rather than solely the parameters.

This is motivated by the observation that the typical use-case for transfer learning is to pre-train on a large and generic dataset (such as ImageNet), before fine-tuning on a smaller and more specific target dataset of interest (such as birds, faces, or medical images). In this case, while the source possesses useful and related information, it also contains irrelevant and useless information, since it comes from a more general purpose dataset. Therefore, a fine-tuning algorithm that also predicts which parts of the source model to *ignore* by pruning parameters could be faster and, ultimately, improve accuracy (since the parameter reduction results in fewer computations, and thus reduces overfitting in the target problem). Therefore, we perform learning-to-learn to obtain a fine-tuning algorithm that is both efficient and effective via refining both the weights and the structure of the network to better suit the target problem.

1.3 Thesis Outline

The rest of this thesis is organised into five chapters:

Chapter 2: We summarise the background related to (i) learning-to-learn methods and (ii) learning process problems. For the learning-to-learn methods, we summarise the main techniques of meta-learning and how meta-learning potentially improves the supervised learning. Regarding the learning process problem, we summarise how meta-learning can help to improve active learning and transfer learning.

Chapter 3: We start by studying learning-active-learning online. First, we identify the existence of non-stationarity in active learning; in other words, the ideal active learning criterion varies over time as the base learner improves. Subsequently, we propose a new non-stationary bandit learning approach to learn the best active learning over different time periods. This work was published in ICPR 2018 [Pang et al., 2018a].

Chapter 4: Rather than restricting our final AL criterion to being a weighted ensemble of a pool of existing criteria, we investigate the learning of a DNN model for an active query criterion. To avoid the cold start problem discussed in Chapter 3, we focus on learning a transferable policy that can be trained on source dataset(s), and subsequently applied to any target dataset, even across heterogeneous feature spaces. Parts of this work were published in ICML-AutoML workshop 2018 [Pang et al., 2018b].

Chapter 5: We propose a novel learning-transfer-learning framework, which involves learning a transferable fine-tuning rule of network structure. For this, we train a fine-tuning policy to determine which and how many source neurons to delete in order to obtain the best fine-tuned target model. We also generalise the policies to different datasets and different initial values in the network.

Chapter 6: We conclude the thesis and summarise the recommendations for future research.

1.4 Thesis Contributions

In this thesis, we propose learning-to-learn techniques for the active learning and transfer learning tasks, which is currently relatively less studied in the machine learning pipeline. The main contributions are listed as follows:

- We formulate a general meta-learning framework for machine learning tasks where active learning and transfer learning are the special cases of this framework.
- We identify the existence of the non-stationary phenomenon in the active learning criterion selection. Our experimental results show that the most useful criterion will be varied at a different active learning stage.
- We extend the previous stationary bandit learning algorithm with expert advice to a non-stationary setting, which also has a theoretical guarantee about the

worst-case. Based on the proposed non-stationary bandits, we further develop an active learning algorithm to select the most promising criterion at a different stage so that we can choose the different useful criteria dynamically.

- We develop a learning transferable active learning algorithm which is trained with deep reinforcement learning in an end-to-end learning way. More importantly, the proposed method is to learn the active learning policy directly from the raw feature and also can transfer to different datasets regardless of the variant dimensionality. Besides, this is also able to avoid the extra effort and resource wastes which are introduced by the cold start learning active learning scheme.
- We present to learn a novel update rule for neural network architecture in transfer learning with deep reinforcement learning. The proposed update rule can delete the useless source neurons to the target problem so that we can dynamically tune and shrink the size of network architecture during the fine-tuning process. Our experiment result shows that it is capable of improving the network generalisation performance by reducing the model complexity.

Chapter 2

Background and Problem Statement

Since the efficacy and efficiency of supervised learning depend on data collection and knowledge transfer processes, this thesis aims to improve these processes to enhance overall performance. The general supervised learning problem is the basis for the successive research problem. Then, we demonstrate the concept of meta learning and provide a general meta learning for any learning processes. Thus, we aim to show how this general configuration can help improve supervised learning.

Next, we discuss the setting of active learning and show how active learning intuitively improves the data collection process. Instead of reviewing all of the conventional work here, we present a general formalisation of active learning that encompasses several algorithms, and focus on building the connection between active learning and meta-learning.

Finally, we introduce the stochastic gradient descent (SGD) based fine-tuning process, and show how it can be formalised so that it can be improved using meta-learning. We further demonstrate the difference between the core learning process and SGD based fine-tuning process.

2.1 Supervised Learning and Meta Learning

2.1.1 Supervised Learning

It is first necessary to describe the formalisation of the supervised learning setting. Let f represent a model with parameter ω, where the *model* f is used to map an instance with d-dimension $\boldsymbol{x} \in \mathbb{R}^d$ to an output c. The parameter term ω here could either be a scalar or a vector; we use this symbol to depict all the parameters involved in supervised learning. The ground truth and estimated outcome are distinguished by denoting the ground truth as c and the predicted variable as $\hat{c}$. According to the discrete or continuous variable type of the output c, supervised learning can be divided into either classification or regression tasks. Learning the mapping function $f(\boldsymbol{x};\omega) \rightarrow c$ from an instance $\boldsymbol{x}$ to the desired output c is the task of supervised learning.

Before applying the model to predict the desired output, we need to collect the dataset for learning the mapping function. Let $\mathcal{D} = \{\mathcal{D}^{tr}, \mathcal{D}^{te}\}$ denote the dataset for supervised learning tasks, where the $\mathcal{D}^{tr}$ denotes the training set and $\mathcal{D}^{te}$ denotes the test set. Generally, a pair of $\boldsymbol{x}$ and c will be collected and grouped into the training set $\mathcal{D}^{tr} = \{(\boldsymbol{x}_i, c_i)\}_{i=1}^{M^{\text{tr}}}$ and the test dataset $\mathcal{D}^{te} = \{(\boldsymbol{x}_i, c_i)\}_{i=1}^{M^{\text{te}}}$ during the data collection process. The training set $\mathcal{D}^{tr}$ is always visible for both the training and test times, whereas the test set is only available for testing. To simplify the notation, we use M to denote the number of data points in the training set. The loss function is:

$$L(f(\cdot;\omega), \mathcal{D}^{tr}) = \frac{1}{M}\sum_{i=1}^{M} \ell\,[f(\boldsymbol{x}_i;\omega), c_i] \tag{2.1}$$

where $\ell(f(\boldsymbol{x}_i;\omega), c_i) \rightarrow \mathbb{R}$ is the loss on one data point $\boldsymbol{x}_i$ with its corresponding ground truth c_i, and $L(f(\cdot;\omega), \mathcal{D})$ is the loss on the entire dataset $\mathcal{D}$.

We then use the collected data to train a model and evaluate the model's performance. During training, the training dataset $\mathcal{D}^{tr}$ is the input, and the output is a well-trained model with optimal parameters ω^*, which is achieved by minimising the loss function:

$$\omega^* = \operatorname*{argmin}_{\omega} \frac{1}{M}\sum_{i=1}^{M} \ell\,[f(\boldsymbol{x}_i;\omega), c_i] + \mathcal{R}(w) \tag{2.2}$$

with respect to the parameters ω. This loss function L could denote the misclassification loss for the classification problem or the error function for the regression task. Once the training is finished, we evaluate the well-trained models f_{ω^*} performance on the unseen test dataset $\mathcal{D}^{te}$. To further distinguish between 'model' and 'learner', the model uses the instance as the input and emits the corresponding output value $f(\boldsymbol{x};\omega) \rightarrow \hat{c}$, while the hand-crafted learner b inputs the dataset and outputs the well-trained model $b(\mathcal{D}^{tr}) \rightarrow f_{\omega^*}$.

2.1.2 Meta Learning

Meta learning aims to lead the learner towards a better learning performance from a higher-level perspective. More specifically, meta-learning treats the experience of learning as a training example, and aims to set the learners parameters to improve learning performance in the meta-learning phase. Note that meta-learning typically uses learning as a subroutine. Each learning experience provides one training example to the meta-learner, similar to how each data point provides one training example in regular learning. As can be seen in Figure 2.1, the trained learner $b(\mathcal{D}^{tr};\Theta)$ plays an analogous role to a hand-crafted learner $b(\mathcal{D}^{tr})$, but adds trainable parameters Θ that allow it to improve its performance. In this section, we provide a general formalisation of meta learning.

We next introduce the concept of a meta learning task. Formally, a base learner $b(\mathcal{D}^{tr};\Theta) \rightarrow f_{\omega^{\Theta^*}}$ aims to output a better model with meta-learned parameters Θ by inputting the training set $\mathcal{D}^{tr}$. Unlike the hand-engineered learner, which trains the model in a pre-programmed manner, the meta-learner improves the base learners performance via observing the learning experience from a meta training set $\mathfrak{D}^{tr}$. However, it is important to note that the meta training set and meta test set have various definitions based on their different uses [Finn et al., 2017; Li et al., 2018]. Here, we provide a unified definition in a general meta learning scenario. The *meta dataset* is defined as $\mathfrak{D} = \{\mathfrak{D}^{tr}, \mathfrak{D}^{te}\}$, which includes both the meta training set $\mathfrak{D}^{tr}$ and the meta test set $\mathfrak{D}^{te}$. More specifically, the *meta training set* is grouped by multiple training datasets $\mathfrak{D}^{tr} = \{\mathcal{D}_{\mathrm{j}}^{tr}\}_{\mathrm{j}=1}^{\mathfrak{J}}$ to provide learning experiences for the meta learner, whereas the *meta test set* $\mathfrak{D}^{te} = \{\mathcal{D}_{\mathrm{j}}^{te}\}_{\mathrm{j}=1}^{\mathfrak{J}}$ is collected to provide the training target for the meta learner. We further define *final set* $\mathfrak{D}^{final} = \{\mathcal{D}^{final-train}, \mathcal{D}^{final-test}\}$ to evaluate the meta learner performance, where $\mathcal{D}^{final-train}$ contains the training samples for learning the model's parameters ω^{Θ^*} and $\mathcal{D}^{final-test}$ includes the test samples to compute the loss for the well-trained model. Table 2.1 summarises the related notations and terminologies in the meta learning scenario.

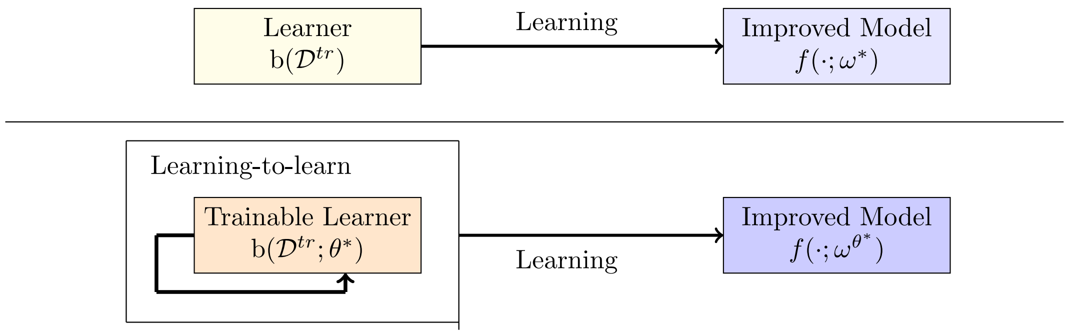


Figure 2.1: The difference between conventional learner and the trained learning.

By rolling out the meta learner $g(\cdot)$ on the meta training set, the meta-learner establishes a sound meta parameter Θ^*, so that the base learner can then establish the model's parameter $b(\cdot;\Theta^*) \rightarrow \omega^{\Theta^*}$ to improve performance. Thus, we use the base learner with Θ^* to produce an improved model $f_{\omega^{\Theta^*}}$, and the performance of that model on the meta test set $\mathfrak{D}^{te}$ provides the meta-learning loss with respect to Θ. Here we define the meta objective function in a supervised learning setting as:

$$\mathcal{F}(b_\Theta, \mathfrak{D}^{tr}, \mathfrak{D}^{te}) = \sum_{\mathrm{j}=1}^{\mathfrak{J}} L(f_{b(\mathcal{D}_{\mathcal{J}}^{tr};\Theta)}, \mathcal{D}_{\mathrm{j}}^{te}) \tag{2.3}$$

$$= \sum_{\mathrm{j}=1}^{\mathfrak{J}} \sum_{i=1}^{M_{\mathrm{j}}^{te}} \ell\left[f(\boldsymbol{x}_{i,\mathrm{j}}; b(\mathcal{D}_{\mathrm{j}}^{tr};\Theta)), c_{i,\mathrm{j}}\right] \tag{2.4}$$

where the feedback of objective function $\mathcal{F}(b_\Theta, \mathfrak{D}^{tr}, \mathfrak{D}^{te}) \rightarrow \mathbb{R}$ is computed from the meta test set $\mathfrak{D}^{te}$. Note that we use b_Θ to summarise $b(\cdot;\Theta)$ to reduce clutter. To train such a base learner, we optimise the meta-learning objective function $\mathcal{F}$ to obtain the optimal meta parameter Θ^*.

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}}\, \mathcal{F}(b_\Theta, \mathfrak{D}^{tr}, \mathfrak{D}^{te}) = \underset{\Theta}{\operatorname{argmin}} \sum_{\mathrm{j}=1}^{\mathfrak{J}} \sum_{i=1}^{M_{\mathrm{j}}^{te}} \ell\left[f(\boldsymbol{x}_{i,\mathrm{j}}; b(\mathcal{D}_{\mathrm{j}}^{tr};\Theta)), c_{i,\mathrm{j}}\right] \tag{2.5}$$

Thus, differences between the conventional supervised learning and the upgraded approach using meta learning can be observed. One difference is that the receiving source of the feedback is different. Recalling the loss function in Equation 2.1, the base learner generally obtains the loss from its corresponding training set, whereas the meta-learner collects the loss from the meta-test set, as illustrated in Equation 2.3 and 2.4. Another difference is the different optimised parameters. According to Equation 2.2, the base learner directly minimises the loss function with respect to parameter ω. The meta learner then optimises the meta-objective function and learns to establish suitable parameters for the base learner ω^{Θ^*}, as illustrated in Equation 2.5.

2.1.3 The Property of General Meta Learning Framework

The general meta learning formalisation framework has potential to perform better than the conventional learning algorithm. For example, the hand-crafted learner for supervised learning can only minimise the loss function for the fitted model on the training set, whereas the trained learner considers how to best set the parameters to improve performance for both the training set and the test set. In addition, for the hand-crafted learner, it is necessary to pre-programme the hyper-parameter for the learning problem, which might require large human efforts for finding the good one. Meta-learning learns a Θ to establish ideal parameters for the learner without intensive exploration. Therefore, it is intuitive that meta-learning can improve both the efficacy and efficiency of conventional learning approaches.

This meta learning formalisation reveals the desired flexibility of the meta learning approach. One advantage of this flexibility is that the meta learning technique can adapt to a variety of learning processes, such as supervised learning, active learning, fine-tuning etc. Depending on the different aims of the learning process, the meta learner aims to establish good parameters for the learner of a particular process, so that the trained learner can achieve respectable performance. For example, if we want to improve the annotation efficiency of active learning, we can meta-learn how to establish parameters for the active learner to acquire fewer annotations while improving the model's performance. This means that the meta learner can observe the experience of the active learner and train the active learner to query a useful data point for annotation. However, we may also want to learn to improve the process of fine-tuning-based transfer learning by removing useless neurons. Subsequently, we can meta-learn how to set the parameters for such a learner to improve the fine-tuning performance.

Another advantage of this flexibility is that it enables the meta-learning approach to improve the learner using different settings, wherein the settings are determined by the design of the meta dataset. The learning settings are not only determined by the meta learning algorithm itself, but are also highly correlated to the evaluation dataset. Based on the different sources of feedback acquisition, the meta learning tasks will also vary. For example, if the meta learner only has the training set to evaluate the trained learner performance, then the meta learning task involves enabling the learner to adapt quickly to the training set. Conversely, if the meta evaluation dataset uses the test set, the meta learning task focuses on enabling the learner to generalise well on the test set.

2.1.4 Learning Supervised Learning

The majority of recent studies on learning-to-learn types methodologies focus on learning the core learning process. Therefore, we introduce the relevant research on supervised learning and formalise the learner as a supervised learning process problem. One well-studied meta-learning technique within a supervised learning context involves learning the update rule of stochastic gradient descent based deep learning approaches. Let b_{sgd} and b_{sgd_Θ} indicate fixed and trainable update rules, respectively.

Symbol	Terminology	Details
$b(\cdot)$	hand-crafted learner	input: $\mathcal{D}^{tr}$ output: ω^*
$b(\cdot;\Theta)$	trained learner	input: $\mathcal{D}^{tr}$ output: ω^{Θ^*}
$g(\cdot)$	meta learner	input: $\mathfrak{D}^{tr}$ output: Θ^*
$\mathcal{D}$	general dataset	a training and test set of a learning problem
$\mathcal{D}^{tr}$	training set	a set used to train a model
$\mathcal{D}^{te}$	test set	a set used to evaluate a well-trained model
$\mathfrak{D}$	general meta dataset	a parent set of meta training and meta test set
$\mathfrak{D}^{tr}$	meta training set	a set of multiple training dataset
$\mathfrak{D}^{te}$	meta test set	a set of multiple test set
$\mathfrak{D}^{final}$	final set	a held out set to evaluate the meta learner
$\mathfrak{D}^{final-train}$	final train set	a set of training the trained learner
$\mathfrak{D}^{final-test}$	final test set	a set of evaluating the trained learner

Table 2.1: The summary of related symbols in meta learning

The fixed trainable update rule could be the stochastic gradient descent update rule $b_{\text{sgd}}(\mathfrak{D}^{tr},\omega) := \omega - \alpha\nabla_{\omega}L(f_{\omega},\mathfrak{D}^{tr})$, where α is the learning rate and $\nabla_{\omega}L(f_{\omega},\mathfrak{D}^{tr})$ is the gradient with respect to the parameter ω. The trained base learner $b_{sgd}(\mathfrak{D}^{tr},\omega;\Theta) \rightarrow \omega^{\Theta^*}$ uses previously updated parameters ω as inputs to establish suitable post-update parameter ω^{Θ^*} that improve the performance on the meta test set $\mathfrak{D}^{te}$. Then, model $f_{\omega^{\Theta^*}}$ uses the post-update parameters to predict the new test instance $\boldsymbol{x}$ for either the classification or regression problem $\hat{c} = f(\boldsymbol{x};b_{sgd}(\mathfrak{D}^{tr},\omega;\Theta^*))$. The forms of learning to update the parameters are defined as:

$$\min_{\Theta}\mathcal{F}(b_{\Theta},\mathfrak{D}^{tr},\mathfrak{D}^{te}) = \min_{\Theta}\sum_{\text{j}=1}^{\mathcal{J}} L(f_{b_{\text{sgd}}(\mathcal{D}^{tr}_{\text{j}};\Theta)},\mathcal{D}^{te}_{\text{j}}) \tag{2.6}$$

A very recent study on meta-learning provides another angle of updating the parameters with stochastic gradient descent [Finn et al., 2017]. The goal of this method is to learn a better initialisation representation that can quickly adapt to a given test dataset. We represent the trainable learner of the model-agonostic meta-learning (MAML) approach as:

$$b_{\text{maml}}(\mathcal{D}^{tr},\omega,\Theta) := \omega - \alpha\nabla_{\Theta}L(\mathcal{D}^{tr})$$

The difference between the general framework of meta learning and the MAML is that the MAML regards each previously updated parameter ω as a meta parameter Θ that needs to be optimised. Then, the MAML optimises the meta-optimisation problem of one-step or multiple-steps SGD updates towards better performance on the meta test set. The form of training the meta parameter is given as follows:

$$\min_{\Theta} \mathcal{F}(b_\Theta, \mathfrak{D}^{tr}, \mathfrak{D}^{te}) = \min_{\Theta} \sum_{j=1}^{\mathfrak{J}} L(f_{b_{\mathrm{sgd}}(\mathcal{D}^{tr},\Theta)}, \mathcal{D}_J^{te}) \tag{2.7}$$

After that the meta-parameter Θ^* is updated, the MAML learner update the original parameter ω with the gradient of the meta parameter $\alpha\nabla_\Theta L(\mathcal{D}^{tr})$. Next, the model uses the updated parameter ω^{Θ^*} to provide the output $\hat{c} = f(\boldsymbol{x}; \omega^{\Theta^*})$ for a given instance $\boldsymbol{x}$.

Another meta-learning approach, $b_{\mathrm{params}}(\boldsymbol{x};\Theta) \to \omega$, learns to synthesise model parameters using data $\boldsymbol{x}$ from a new dataset $\mathcal{D}$ as the inputs [Andrychowicz et al., 2016; Ha et al., 2017]. This meta predicted parameter can reduce the training computation time or deal with large dimensionality [Romero et al., 2017; Ha et al., 2017]. Subsequently, the model uses the predicted parameter ω^{Θ^*} to produce a corresponding output $\hat{c}$ for the supervised learning task $f(\boldsymbol{x}; b_{\mathrm{params}}(\mathcal{D}^{tr};\Theta^*)) \to \hat{c}$.

$$\min_{\Theta} \mathcal{F}(b_\Theta, \mathfrak{D}^{tr}, \mathfrak{D}^{te}) = \min_{\Theta} \sum_{j=1}^{\mathfrak{J}} L(f_{b_{\mathrm{params}}(\mathcal{D}_j^{tr};\Theta)}, \mathcal{D}_j^{te}) \tag{2.8}$$

2.2 Meta Learning About the Learning process

2.2.1 Learning the Active Learning Process

Introduction of Active Learning: We introduce the pool-based active learning. We denote the pool of data with M samples as $\mathcal{D} = \{\boldsymbol{x}_i, c_i, \ldots, \boldsymbol{x}_M, c_M\}$, where the instances are $\boldsymbol{x}_i \in \mathbb{R}^d$, and the labels are $c \in \{1, \ldots, C\}$, most or all of which are unknown in advance. In an active learning scenario, the data $\mathcal{D}$ is initially a labelled set $\mathcal{L}$ and an unlabelled set $\mathcal{U} = \mathcal{D} \setminus \mathcal{L}$, where $|\mathcal{L}| \ll |\mathcal{U}|$. Training a classifier f on the samples in the initial set $\mathcal{L}$, the algorithm starts to query instances q from $\mathcal{U}$ during iterations $t = \{1, \ldots, T\}$. After the supervision of instance q is obtained, q is removed from the unlabelled set $\mathcal{U}$ and added to the labelled set $\mathcal{L}$, from which classifier f_t is retrained. This means that a pool-based active learner b_{AL} selects an instance/point from the unlabelled pool/set $\mathcal{U}$ to query its label, which can be formulated as $b_{\mathrm{AL}}(\mathcal{L}, \mathcal{U}, f) \to q$, where $q \in \mathcal{U}$. Subsequently, the classifier f is retrained based on the updated labelled set $\mathcal{L}$. Based on these, we formalise the objective function of the active learning task on the test dataset $\mathcal{D}^{te}$ as:

$$q^* = \underset{q \in \mathcal{U}}{\operatorname{argmin}} \frac{1}{M^{te}} \sum_{i=1}^{M^{te}} \ell \left[f(\boldsymbol{x}_i; \omega(\mathcal{L} \cup q)), c_i \right] \tag{2.9}$$

$$\text{where:} \omega(\mathcal{L} \cup q) = \underset{\omega}{\operatorname{argmin}} \frac{1}{|\mathcal{L} \cup q|} \sum_{j=1}^{|\mathcal{L} \cup q|} \ell \left[f(\boldsymbol{x}_j; \omega, \mathcal{L} \cup q), c_j \right] \tag{2.10}$$

As illustrated by the objective function 2.9, active learning aims to achieve a respectable performance with fewer annotated data points $\mathcal{L}$ on the test set $\mathcal{D}^{te}$. The queried instance q^* is selected from the unlabelled set $q^* \in \mathcal{U}$ to minimise the test set loss, which are then appended to the labelled set $\mathcal{L} = \mathcal{L} \cup q^*$. However, optimising such an objective function for every data point in a large unlabelled set is intractable, since each data point is needed to retrain and test the classifiers performance to acquire its usefulness. In addition, the test set is invincible to the supervised learner during the learning phase. Thus, active learning algorithms are developed to estimate the usefulness of data points without retraining the classifier.

Conventional Active Learning: Most conventional active learning algorithms are heuristic, and the active learning criteria are pre-defined based on different motivations. These criteria observe the state of the labelled set, unlabelled set, and classifier, and subsequently predict the most useful instance $b_{AL}(\mathcal{L}, \mathcal{U}, f) \rightarrow q^*$ [Lewis and Gale, 1994; Tong and Koller, 2002; Roy and McCallum, 2001; Cohn et al., 1995; Chattopadhyay et al., 2012; Yu et al., 2006; Seung et al., 1992; Abe and Mamitsuka, 1998; Loy et al., 2012; Huang et al., 2010; Wang and Ye, 2015; Wang et al., 2017; Hospedales et al., 2013, 2012]. However, all of these algorithms are heuristically designed with various beliefs about what constitutes a 'good' data point to annotate. For example, the uncertainty sampling approach aims to locate the least confident data point to annotate. Although some of the proposed active learning algorithms assemble multiple motivations, they still rely on the heuristic switch scheme to switch one criterion to another [Donmez et al., 2007; Hospedales et al., 2013]. Other researchers have suggested meta learning about the active learning criterio n. However, the related meta-learning methodologies of active learning are relatively less studied than the learning supervised learning problem. For this reason, we investigate meta-learning of active learning from two perspectives: learning active learning process online and learning transferable active learning policy.

Learning Active Learning (LAL): We also introduce meta learning the active learning process. In this instance, $b_{\text{LAL}}(\mathcal{L}, \mathcal{U}, f; \Theta)$ denotes the learned active learner and inputs the labelled set, unlabelled set, and model. It learns to emit the data point to query $b_{\text{LAL}}(\mathcal{L}, \mathcal{U}, f; \Theta) \to q^*$, and performs the pool-based active learning process by removing the instance from the unlabelled set $\mathcal{U} = \mathcal{U} \backslash q^*$ and append the point to the labelled set $\mathcal{L} = \mathcal{L} \cup q^*$. Then, we define the meta objective function for the active learning task:

$$\min_{\Theta} \mathcal{F}(b_\Theta, \mathfrak{D}^{tr}, \mathfrak{D}^{te}) = \min_{\Theta} \sum_{t=1}^{T} \sum_{\mathrm{j}=1}^{\mathfrak{J}} L\left[f_{b_{\text{LAL}}(\mathfrak{D}_{\mathrm{j}}^{tr};\Theta)}, \mathfrak{D}_{\mathrm{j}}^{te} \right] \tag{2.11}$$

To train this, the meta-learner optimises the meta objective function $\mathcal{F}(b_\Theta, \mathfrak{D}^{tr}, \mathfrak{D}^{te})$, where the function $\mathcal{F}$ depends on the adopted meta learning methods (the details of the proposed methods will be described in Chapter 3 and Chapter 4). Here, we briefly summarise the presented methods in a meta learning scenario. We further define two special cases of this general framework: learning active learning online and learning transferable active learning policy.

Learning Active Learning Online (LALO): Regarding learning active learning online, we denote the meta training set as a single training dataset $\mathfrak{D}^{tr} = \mathcal{D}^{tr}$, where the meta test set is also the training set, but the feedback is computed on either the labelled set or the unlabelled set $\mathfrak{D}^{te} = \mathcal{D}^{tr} = \{\mathcal{L} \cup \mathcal{U}\}$ [Baram et al., 2004; Hsu and Lin, 2015; Chu and Lin, 2016]. For example, [Baram et al., 2004] measures the classification entropy of the queried instance on the unlabelled set, whereas [Hsu and Lin, 2015] uses the importance weighted estimator of the test accuracy on the labelled set. It is evident that the updates for the labelled set $\mathcal{L}$ will successively affect the update of the meta learner. Thus, learning active learning online needs to update both the labelled set and the meta learner iteratively to improve data efficiency. The learning active learning online is defined below:

$$\min_{\Theta} \mathcal{F}(b_\Theta, \mathfrak{D}^{tr}, \mathfrak{D}^{te}) = \min_{\Theta} \sum_{t=1}^{T} L\left[f_{b_{\text{LALO}}(\mathcal{L}_t, \mathcal{U}_t, f_t; \Theta)}, \mathcal{D}^{te} \right] \tag{2.12}$$

Learning Transferable Active Learning (LTAL): Different from the online learning active learning scheme, the objective function of the learning transferable active learning policy observes the experience from multiple datasets $\mathfrak{D}^{tr} = \{\mathcal{D}_1^{tr}, \dots, \mathcal{D}_{\mathfrak{J}}^{tr}\}$ and evaluates $\mathfrak{D}^{te} = \{\mathcal{D}_J^{te}, \dots, \mathcal{D}_{\mathfrak{T}}^{te}\}$ [Konyushkova et al., 2017a]. Training on these varied sets enables the learned policy to be dataset-agnostic, which helps learn a general active learning criterion for all of the different datasets.

$$\min_{\Theta} \mathcal{F}(b_{\Theta}, \mathfrak{D}^{tr}, \mathfrak{D}^{te}) = \min_{\Theta} \sum_{t=1}^{T} \sum_{J=1}^{\mathcal{J}} L\left[f_{b_{\mathrm{LTAL}}(\mathcal{L}_{J,t}, \mathcal{U}_{J,t}, f_{J,t};\Theta)}, \mathcal{D}_{J}^{te}\right] \tag{2.13}$$

2.2.2 Learning About the Fine-tuning Process

Fine-tuning: We now introduce the stochastic gradient descent (SGD) based fine-tuning in deep learning scenario. This gradient-descent iterative approximation method can update the model's weight by back-propagating the gradients that are computed from a differentiable objective function using randomly selected instances $\omega^* = \omega - \alpha\nabla_{\omega}L(f(\cdot;\omega), \mathcal{D}^{tr})$ [Robbins and Monro, 1951]. Moreover, the SGD-based algorithms input both the updated parameters and the dataset to produce new parameters for the deep networks $b_{sgd}(f_{\omega_{t-1}}, \mathcal{D}^{tr}) \to f_{\omega_t}$.

Learning to fine-tune (LFT): Unlike the previous meta-learning method for the update rule, which only updates the model's parameters, we propose learning an update rule that applies to tune the architecture of the deep network (we discuss the meta-learning method in Chapter 5). We could consider tuning the network architecture as updating the parameter for a specific optimiser. The meta learner is defined as: $b_{LFT}(f_{\omega_{t-1}}, \mathcal{D}^{tr};\Theta) \to f_{\omega_t^{\Theta*}}$. Thus, the proposed meta-learner generates new parameters for the SGD optimisation algorithm. We define the objective function for learning the meta-learned update rule as follows:

$$\min_{\Theta} \mathcal{F}_{\mathrm{LTAL}}(b_{\Theta}, \mathfrak{D}^{tr}, \mathfrak{D}^{te}) = \min_{\Theta} \sum_{t=1}^{T} \sum_{\mathrm{j}=1}^{\mathfrak{J}} L\left[f_{b_{\mathrm{LFT}}(f_{\omega_{t-1}}, \mathcal{D}_{\mathrm{j}}^{tr};\Theta)}, \mathcal{D}_{\mathrm{j}}^{te}\right] \tag{2.14}$$

In this thesis, we aim to train a dataset and model agnostic architecture update rules. Similar to the transferable scheme of learning active learning, transferability is achieved by training on multiple meta training sets $\mathfrak{D}^{tr} = \{\mathcal{D}_1^{tr}, \ldots, \mathcal{D}_{\mathfrak{J}}^{tr}\}$ and test sets $\mathfrak{D}^{te} = \{\mathcal{D}_1^{te}, \ldots, \mathcal{D}_{\mathfrak{J}}^{te}\}$.

Chapter 3

Learning Active Learning Online

Most conventional active learning algorithms are hand-engineered based on various philosophies concerning what constitutes a good criterion. Different criteria perform well on different datasets, and there is no single criterion that performs best for all datasets. This single fact has motivated research into ensembles of active learners to learn what constitutes a good criterion in a given scenario by re-weighting the ensemble members.

In this chapter, we present a learning approach to select the best active learning criterion online. Moreover, we demonstrate that the best criterion is not only different for each dataset but also varies as more points are queried. Given this observation, the proposed approach re-selects the winning criterion periodically. Unlike previous techniques that only aim to select one criterion per dataset, the goal of our algorithm is to estimate a potentially changing winner as the annotation process proceeds. The primary contribution in this chapter is that our proposed approach addresses the problem of selecting a dynamically changing best criterion, and also comes with a performance guarantee. In addition, experiment results demonstrate the effectiveness of our proposed approach in terms of performance vs annotation effort, particularly for datasets for which the best criterion evolves as the annotation proceeds.

3.1 Introduction

The key barrier to scaling or applying supervised learning in practice is often the cost of obtaining sufficient annotation. Active Learning (AL) aims to address this by designing query algorithms that effectively predict which points are useful to annotate, thus enabling the efficient allocation of human annotation effort. There are many different AL algorithms, each with appealing yet completely entirely different motivations for what constitutes a good question to ask underpinning their design. For example, uncertainty or margin-based sampling [Lewis and Gale, 1994; Tong and Koller, 2002] suggests querying the most uncertain or ambiguous point, that is the closest point to the decision boundary. Expected error reduction [Roy and McCallum, 2001; Hospedales et al., 2012] queries points that the current model predicts will reduce its future error. Another typical approach is to label the most representative samples [Cohn et al., 1995; Chattopadhyay et al., 2012; Yu et al., 2006] to ensure the major clusters within the dataset are correctly estimated. Besides these approaches, query-by-committee active learning queries points based on the disagreement between a committee of classifiers [Seung et al., 1992; Abe and Mamitsuka, 1998; Loy et al., 2012]. More recent studies have investigated hybrid criteria that balance multiple motivations [Huang et al., 2010; Wang and Ye, 2015; Wang et al., 2017].

Although these are all good ideas, there are situations where each is ineffective. For example, if the classes are heavily overlapped in an area of feature-space, uncertainty sampling will tie up querying points in an impossible to solve region. Moreover, if the current model is poor, expected error reduction cannot accurately estimate its own future error. If the main data clusters are already well classified, representative sampling approaches may not fine-tune them. These thought experiments are reflected empirically. The best algorithm for pool-based AL, in practice, varies both across datasets and also with the progress of learning within a given dataset [Baram et al., 2004; Hsu and Lin, 2015]. This observation has motivated research into both learning dataset and time-specific weightings for an AL algorithm ensemble. [Donmez et al., 2007; Hospedales et al., 2013] has developed a heuristic approach for switching AL algorithms that are typically good at early- vs late-stage learning. The time-specific weighting means that the preference of the assembled active learning criteria will vary as the number of annotations increases. However, these time-specific weighting schemes are still heuristically designed, which will not be always the most appropriate schemes for the application in the real world. In contrast, some other researchers have developed methods for the rapid online meta learning of a dataset-specific weighting for algorithms within an AL-ensemble [Baram et al., 2004; Hsu and Lin, 2015].

The key insights of the Combination of Active Learning Online (COMB) [Baram et al., 2004] and Active Learning by Learning (ALBL) [Hsu and Lin, 2015] algorithms is to formalise the query criteria selection task as a multi-armed bandit (MAB) problem. MAB problems have been well studied and many powerful algorithms with optimality guarantees exist. For example, if each query criterion in the ensemble is considered to be a bandit arm, and the learning improvement achieved after executing a criterion is considered to be the arms reward, then MAB algorithms, such as EXP3 (Exponential-weight algorithm for Exploration and Exploitation) [Auer et al., 2002b], can be applied to quickly learn the efficacy of the arms (AL criteria), as this is guaranteed to achieve a near optimal overall reward (learning improvement). A variant of this is to consider data-points as arms, and AL criteria as experts that suggest which arms are promising. Subsequently, MAB with *expert advice* algorithms, such as EXP4.P (Exponential-weight algorithm for Exploration and Exploitation using Expert advice with high probability regret bound) [Beygelzimer et al., 2011], optimise the exploration and exploitation of experts, and achieve provably near optimal reward.

The fundamental limitation of existing MAB-based approaches to AL is that their underlying MAB algorithms do not take into account the temporal dynamics of active learning: different criteria are effective at different learning stages [Donmez et al., 2007; Hospedales et al., 2013]. We identify this issue on various AL algorithms such as uncertainty sampling (US), representative sampling (RS), density sampling (DE), random sampling (RAND) and the proposed dynamic ensemble active learning algorithm (DEAL). Following the queried decision made by a particular AL algorithm, we pre-compute the accuracy increment of all mentioned heuristic algorithms at each time step. Then, we quantify the winning proportion and relative accuracy increments for a fixed time interval with window size $\Delta_T = 10$. The time step corresponds to the number of annotated instance.

In Figure 3.1, the first issue is illustrated that the most effective criterion varies across the entire time horizon. On *fourclass*, following a different AL algorithm's queried decision, the effectiveness of the AL algorithm could be varied. For example, following the queried decision of US (1st row) suggest that DE will be useful at first and RS could be slightly better at a later point. While the queried decision is made by DE (3rd row), querying some of the US could bring the most beneficial to the AL task. On *ILPD* or *german*, representative (RS) and density (DE) sampling are better at the crucial early stages, before uncertainty becomes better.

A second issue is that the scale of an accuracy-based reward decreases dramatically over time (Figure 3.2). Because of this stationary bandit learners are unduly biased by the high reward gained from an initial observation and fail to adapt later. For example, in ILDP, a stationary learner may fail to switch from DE to US, because later rewards in favour of US are small in scale compared to the initial reward in favour of DE.

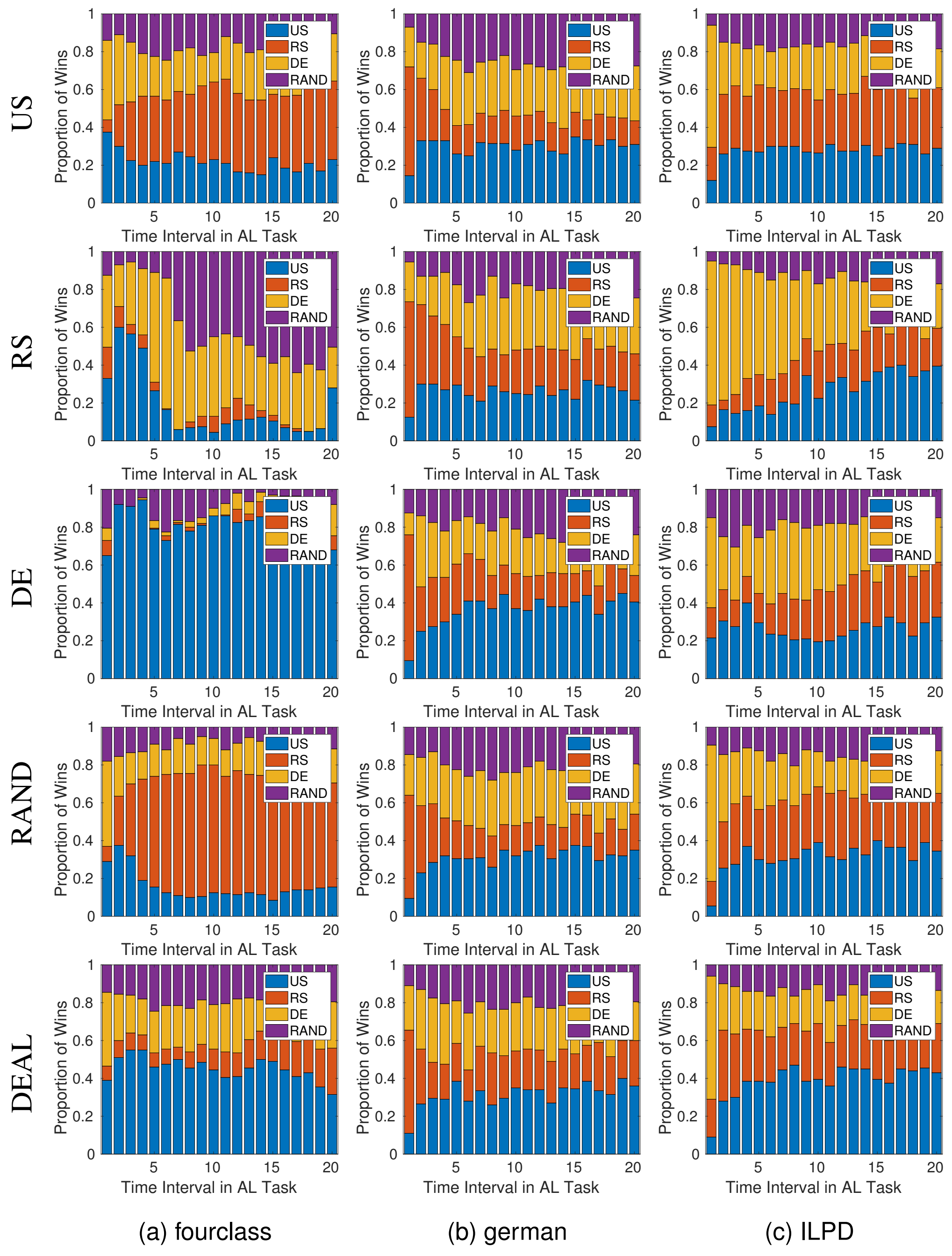
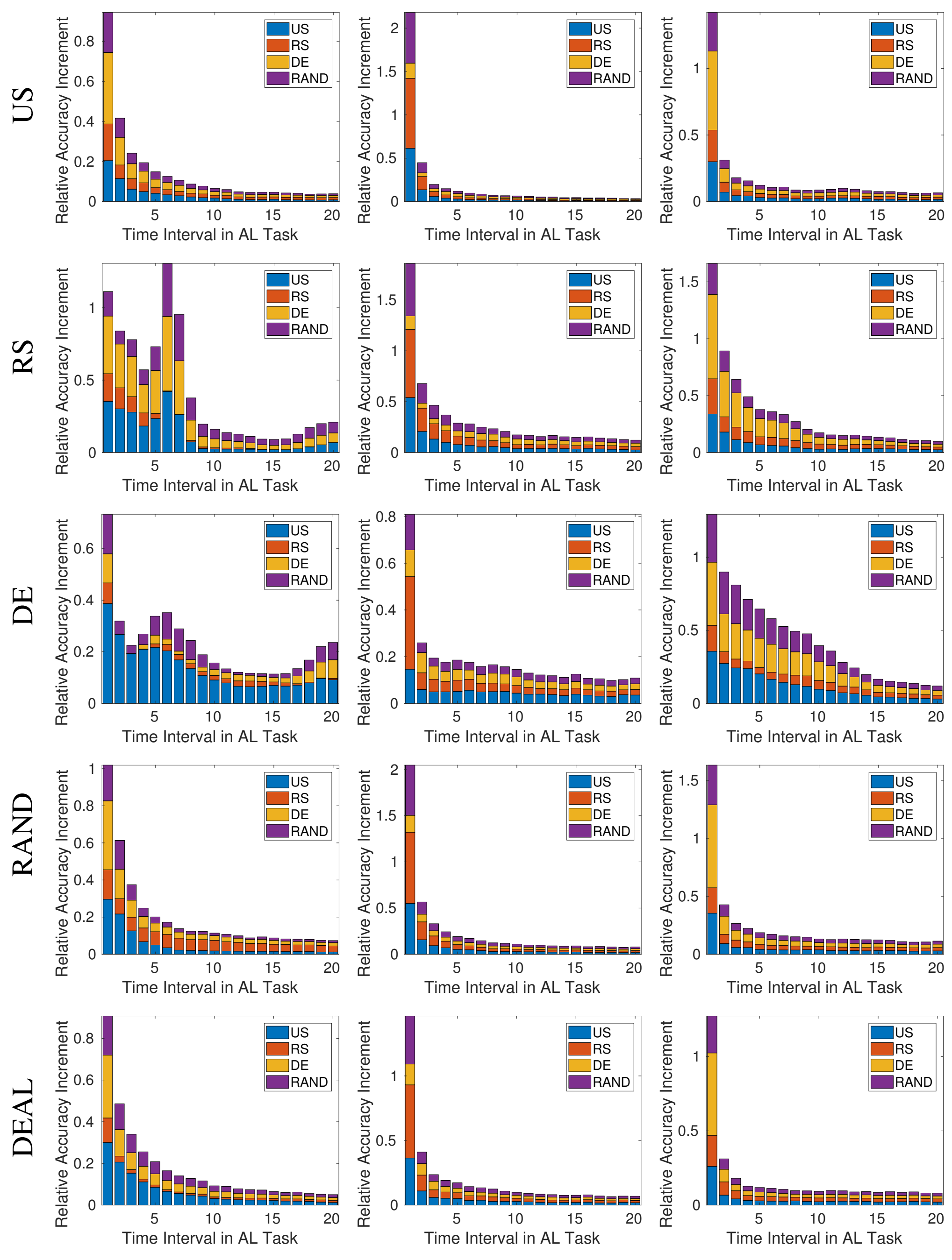


Figure 3.1: Examples of non-stationary AL in UCI datasets "fourclass", "german", "ILPD" using five algorithms/criteria: US, RS, DE, RAND, and DEAL. Proportion of times each criterion generates the largest increase in accuracy. Rows: The five actor algorithms (US, RS, DE, Rand, DEAL) used to collect the trajectories for the rollout. Bars: The effect of querying each criteria at the given iteration, given the the rollout generated by the actor algorithm on the left.



(a) fourclass (b) german (c) ILPD

Figure 3.2: Examples of non-stationary AL in UCI datasets “fourclass”, “german”, “ILPD” using five algorithms/criteria: US, RS, DE, RAND, and DEAL. In the relative part all increments are re-scaled by subtracting the minimum increment of accuracy over all criteria in each bin. Rows: The five actor algorithms (US, RS, DE, Rand, DEAL) used to collect the trajectories for the rollout. Bars: The effect of querying each criteria at the given iteration, given the the rollout generated by the actor algorithm on the left.

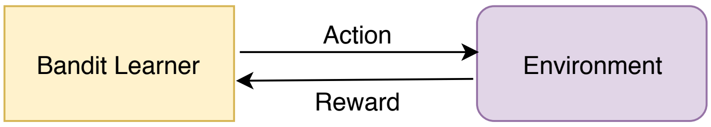


Figure 3.3: The illustration of bandit learning

Therefore, there are non-stationary aspects both in reward scale and in reward distribution per-arm (MAB perspective) or per-expert (MAB with expert advice perspective). Thus, the MAB problem is formally non-stationary, violating a fundamental assumption required that is necessary to guarantee existing MAB algorithms' optimality bounds .

Here, we develop a performance-guaranteed stochastic MAB with expert advice[1] algorithm in a *non-stationary* environment. Applying this to AL means that, like [Hsu and Lin, 2015], if there is a single best (but a priori unknown) AL algorithm for a dataset, we are able to quickly discover it, and thus approach the performance of an oracle that knows the best algorithm for each dataset. More importantly, however, when different algorithms' efficacies vary over time within a given dataset, we can adapt and approach the performance of an oracle that knows the best AL algorithm *at each iteration*.

3.2 Related Work

Multi-Armed Bandit: In multi-armed bandit (MAB) problems, a player pulls a lever from a set $\mathcal{K} = \{1, \ldots, K\}$ of slot machines in a sequence of time steps $\mathcal{T} = \{1, \ldots, T\}$ to maximise her payoff. During the game, she only observes the reward $r^k(t) \in [0,1]$ of the specific arm pulled k at time step t, where the reward is unknown distribution. The aim of the player is to maximise their return, which is the sum of the rewards over the sequence of pulls. This requires a trade-off between exploration (collect information to estimate the arm with the highest return) and exploitation (focus on the arm with the highest estimated return). As illustrated in the Figure 3.3, the bandit learner interacts with the environment by emitting the actions and receiving a random reward, where environment is the MAB problem. Training a bandit learner to solve a MAB problem is then formalised as minimising the *regret* between the actions chosen by the player's

[1]We use terminology from [Auer et al., 2002b]. It also has other names, including 'contextual bandit' [Beygelzimer et al., 2011; Langford and Zhang, 2008], 'partial-label problem' [Kakade et al., 2008], and 'associative bandit problem' [Strehl et al., 2006].

strategy $a_k \sim \pi$, and the best arm where policy π is the probability for taking action a_k. This regret is to measure the learning speed of bandit learner.

For stochastic MAB, the reward of each arm is sampled from an unknown distribution. Let μ_t denote the expectation of the reward distribution of arm k. The goal is then to minimise the regret which is defined as:

$$T \max_k \mu_k - \mathbb{E}(\sum_{t=1}^{T} r_t^{\pi}). \tag{3.1}$$

Algorithms such as upper confidence bound (UCB)1 [Auer et al., 2002a] can be shown to have near minimal possible regret. However, the stochastic MAB action is completely determined by a fixed distribution of rewards. This may be an invalid assumption in real world situations where the reward distribution can evolve over time.

A later more general variant is the Adversarial (Non-stochastic) MAB, in which the rewards can be set by an adversary. This adversarial MAB relaxes the stationary reward distribution assumption. The exponential-weight algorithm for exploration and exploitation (EXP3) algorithm [Auer et al., 2002b] minimises, for any finite time horizon T, the "*static regret*" between the player's reward and the best arm in retrospect:

$$\max_k \sum_{t=1}^{T} r_t^k - \mathbb{E}(\sum_{t=1}^{T} r_t^{\pi}). \tag{3.2}$$

where the term 'static' means there is only one best arm $\max_k \sum_{t=1}^{T} r_t^k$ in the entire time horizon.

Contextual Multi-armed Bandit: In many practical problems, some contextual information is available that provides a cue about the likely reward of an arm at a given time. This can be addressed with the contextual multi-armed bandit formalisation where at each time step t, a context is observed to help to describe arms. The goal of contextual bandits is to build a relationship between available context information $\boldsymbol{h}_t \in \mathbb{R}^d$ and the reward function $r_{a_t^*}(t) \to \mathbb{R}$ over all arms. For example, linear Upper Confidence Bound (LinUCB) [Chu et al., 2011] makes the linear realizability assumption that there exists an unknown weight vector $\boldsymbol{\theta}^* \in \mathbb{R}^d$ with $||\theta^*|| \leq 1$ so that regret is minimised, where $r_{a_t^*}(t) = \boldsymbol{\theta}^{*\top}\boldsymbol{h}_t$ and $r_{a_t}(t) = \boldsymbol{\theta}^{\top}\boldsymbol{h}_t$.

$$\sum_{t=1}^{T} r_{a_t^*}(t) - \sum_{t=1}^{T} r_{a_t}(t) \tag{3.3}$$

Applying such context information can improve the efficiency of solving the trade-off between exploration and exploitation. Another kind of contextual information is expert information about the likely efficacy of each arm. Expert information about the likely efficacy of each arm is often available [Auer et al., 2002b] thus introduced an exponential-weight algorithm for exploration and exploitation with expert advice algorithm (EXP4) that exploits N experts giving advice vectors (probabilities $\boldsymbol{\xi}^n(t) \in [0,1]^K$ over levers) to the learner at each time. In contrast to MAB without expert advice, the goal is now to identify the best expert rather than the best arm. In this setting the regret to minimise is the difference between the return of the best expert in retrospect and the player:

$$\max_n \sum_{t=1}^{T} y_t^n - \mathbb{E}(\sum_{t=1}^{T} y_t^{\pi}) \tag{3.4}$$

where $y_t^n = \sum_{k=1}^{K} \xi_k^n(t) \times r^k(t)$ is the expected reward of an expert and y_t^{π} is the expected reward of our policy.

3.2.1 Stationary Bandit Learning for Active Learning

MAB algorithms such as those described in the previous section provide a convenient formalisation for estimating the relative efficacy of different active learning criteria on a given dataset. For active learning using a MAB with expert advice algorithm, the N experts correspond to our ensemble of active learning criteria and the K arms are available points in the pool. Each expert (criterion) n provides a probability vector encoding preference $\boldsymbol{\xi}^n(t)$ over arms (instances). Active learners based on MAB with expert advice aim to learn the best criterion for a specific dataset. In COMB [Baram et al., 2004], the authors propose to use MAB with expert advice in active learning and heuristically designed the classification entropy maximization (CEM) score as the reward of the EXP4 bandit algorithm [Auer et al., 2002b]. A more recent paper [Hsu and Lin, 2015] (ALBL) proposed to replace the CEM reward with an unbiased estimation of test accuracy Important Weighted Accuracy (IWA) and used an upgraded bandit algorithm EXP4.P [Beygelzimer et al., 2011], which improves the earlier EXP4 method. Similarly, another recent paper [Chu and Lin, 2016] applied linear upper confidence bound contextual bandit algorithm (LinUCB) to train an ensemble and transferred the knowledge to other datasets. All of these algorithms enable the selection of a suitable active learning criteria for a given dataset. Our contribution is also to perform AL in a dataset-specific way by optimally tuning the exploration and exploitation of an ensemble of AL algorithms; but more importantly to do so dynamically, thus allowing the optimal tuning to vary as learning progresses. Unlike [Baram et al., 2004; Hsu and Lin, 2015; Chu and Lin, 2016] we are able to deal with the non-stationary nature of this process. And unlike the heuristics in [Donmez et al., 2007; Hospedales et al., 2013], we have a theoretical guarantees, and can work with more than two criteria.

3.2.2 Non-stationary Property of Bandit learning and Active Learning

Demonstration of Non-stationarity: Stationary bandits learners assume that the true best arm holds for the entire time horizon. However, in practice the actual situation is that the arm that provides the best reward at the beginning can provide a sub-optimal reward later on (and vice-versa). Under this non-stationary reward distribution, estimating a single best-action is sub-optimal. In contrast, in the stochastic MAB problem, the non-stationary assumption is that the expected reward of arm μ_k would change over time.

We next describe a preliminary experiment to demonstrate empirically the existence of non-stationary reward distributions for a MAB formalisation of AL. Following the learning trajectory of our method, we use an oracle to score all the available query points at each iteration (i.e., hypothetically label each point, update the classifier, and check the test accuracy). Using the actual test accuracy as the reward, we can obtain the true expected reward of the nth expert $y_t^n = \boldsymbol{\xi}^n(t)\boldsymbol{r}(t)$ at each time step t. Figure 3.2 summarises the resulting average reward obtained in every 10 iterations of AL. Based on this, we can further compute the proportion of times that each criterion would obtain the highest reward. It can be seen that the MAB problem is non-stationary as the rewards vary systematically, and there is not a single criterion (expert) which obtains the highest proportion of wins throughout learning. Additionally, the ideal combination of criteria varies across datasets. For example, as illustrated in Figure 3.1, density and uncertainty sampling show better complementary in ILPD, while representative and uncertainty sampling are more complementary in german dataset.

Existing MAB ensembles are not robust to non-stationarity: The non-stationary property in the MAB formalisation of AL also highlights the key weakness of COMB and ALBL: they use EXP4/EXP4.P [Auer et al., 2002b; Beygelzimer et al., 2011] expert advice bandit algorithms which provide guarantees against an inappropriate (static) regret that is only relevant in a stationary problem. In a non-stationary problem, it is clear that even an algorithm that perfectly estimates the best single expert (optimal w.r.t static oracle Equation 3.4) can be arbitrarily worse than one which can choose the best expert at each step (optimal w.r.t dynamic oracle). Here, we develop an non-stationary stochastic MAB algorithm REXP4 (Restarting Exponential-weight algorithm for Exploration and Exploitation using Expert advice) with bounds against a stricter dynamic oracle notion of optimality more suited for (non-stationary) AL.

Prior attempts at non-stationary active learners: A few previous active learning studies also observed that different algorithms are effective at different stages of learning and proposed heuristics for switching two base query criteria (e.g., density sampling at an early stage, and uncertainty sampling later on) [Donmez et al., 2007; Hospedales et al., 2013]. But these only adapt 2 criteria (density and uncertainty) unlike MAB ensembles which learn to combine many criteria, and their heuristics do not provide a principled and optimal way to learn when to switch.

Prior attempts at non-stationary MABs: Some previous studies have extended MAB without expert advice learning to the non-stationary setting [Garivier and Moulines, 2008; Besbes et al., 2014] and provided regret bounds to guarantee the algorithms' performance. However bandits with expert advice are preferable because they can achieve tighter learning bounds [Auer et al., 2002b; Hsu and Lin, 2015] and they do not treat each criterion as a black box, so that one observation can be informative about many arms. Consider an AL situation where two criteria prefer the same instance. In the MAB interpretation (criteria=arms), after observing a reward, you only learn about the criterion/arm chosen at that iteration. In the MAB with expert advice interpretation (criteria=experts), the observed reward generates updates about the efficacy of all criteria that expressed opinions about the point.

Those few MABs extended to the non-stationary setting have other stronger assumptions. For example, the discounted/sliding-window UCB algorithm [Garivier and Moulines, 2008] assumes the nature of the non-stationarity is that the reward distribution is piece-wise and the number of changes is known. Similarly [Yu and Mannor, 2009] makes the easier piecewise assumption, and also that the retrospective rewards for un-pulled arms are available – but they are not in active learning. In [Wei et al., 2016], the authors proposed to measure the total statistical variance of the consecutive distributions at each time interval. Their result provides a big picture of the regret landscape for full information and bandit settings. Their proposed method addresses non-stationary environments but only for the regular MAB problem. Despite the use of the term expert in the title, it does not address the Expert-advice variant of the MAB problem relevant to us. It addresses arms rather than experts over arms.

We propose a non-stationary MAB with expert advice algorithm that has performance guarantees, and validate its practical application to active learning.

3.3 Non-stationary Bandit Learning with Expert Advice

We first introduce our new non-stationary MAB algorithm to exploit expert information for tracking the best bandit arm when the unknown reward distribution per arm evolves over time. To formalise the problem, we assume the expected reward y_t^n of each expert n can change at any time step t. The total variation of the expected reward over all T steps is

$$\sum_{t=1}^{T-1} \sup_n |y_t^n - y_{t+1}^n| \tag{3.5}$$

Following [Besbes et al., 2015, 2014], we assume this total variation in expected reward is bounded by a variation budget V_T. The variation budget captures our assumed constraints on the non-stationary environment. It allows a wide variety of reward changes – from continuous drift to discrete jumps – yet provides sufficient constraint to permit a bandit algorithm to learn in a non-stationary environment. Temporal uncertainty set $\mathcal{V}$ is defined as the set of reward vector sequences that are subject to the variation budget V_T over all T steps.

$$\mathcal{V} = \left\{ y \in [0,1]^{N \times K} : \sum_{t=1}^{T-1} \sup_n |y_t^n - y_{t+1}^n| \leq V_T \right\}$$

To bound the performance of a bandit learner in a non-stationary environment, we work with the regret between the learner and a *dynamic* oracle. The regret is defined as the worst-case difference between the expected policy return and the return of using the best expert at each time t.

Definition 1. *Dynamic Regret for Multi-Armed Bandit with Expert Advice*

$$R^{\pi}(\mathcal{V}, T) = \sup_{y \in \mathcal{V}} \{ \sum_{t=1}^{T} y_t^* - \mathbb{E}^{\pi}[\sum_{t=1}^{T} y_t^{\pi}] \} \tag{3.6}$$

where $y_t^* = \max_n y_t^n$ is the best possible expected reward among all experts at time t. Our regret is against this dynamic oracle, in contrast to prior MABs' static oracle (Equation 3.4).

Our non-stationary MAB with expert advice algorithm REXP4 minimises the dynamic regret in Equation 3.6. As shown in Algorithm 2, it trades off between the need to remember and forget by breaking the task into batches and applying EXP4 [Auer et al., 2002b] on each batch. As the reward distribution changes, it adapts to the change as by re-estimating each expert's reward distribution at each batch. We show the worst case bound on the regret between this REXP4 procedure and the dynamic oracle.

3.3.1 Regret Bound for REXP4

The regret bound for REXP4 is illustrated in the following theorem. The theorem is proved by following the proof structure of [Besbes et al., 2014] and replacing the term μ in [Besbes et al., 2014] with the expected reward term y in our paper.

Theorem 1. *Let* π *be the REXP4 policy with a batch size* $\Delta_T = \lceil (A\log N)^{1/3}(T/V_T)^{2/3}\rceil$ *and* $\gamma = \min\{1, \sqrt{\frac{A\log N}{(e-1)\Delta_T}}\}$*. Then, there is some constant* C *such that for every* $T \geq 1, K \geq 2, N \geq 2,$ *and* $V_T \in [A^{-1}, A^{-1}T]$

$$R^{\pi}(\mathcal{V}, T) \leq C(A\log N \cdot V_T)^{1/3} T^{2/3} \tag{3.7}$$

where $A = \min\{N, K\}$ *indicates the smaller number of experts or arms.*

Proof. We follow the proof structure of [Besbes et al., 2014] as follows. First, we break the total trials T into a sequence of epochs of size Δ_T each. We then decompose the regret bounds into two parts: (a) The performance gap between the dynamic oracle and the static oracle, (b) The performance gap between the static oracle and EXP4, which is the regret bound of EXP4. Then, we analyse the bounded properties of (a)(b) for each epoch respectively. Finally, we sum over epochs to establish the regret of REXP4 relative to the dynamic oracle.

Step 1 (Decomposition)

We break T into a sequence of batches $T_1, \ldots, T_J$ of size Δ_T each (except possibly T_J). Let $j \in \{1, \ldots, J\}$ be the index of epochs, $n \in \{1, \ldots, N\}$ be the index of experts. We decompose the regret in batch j as follows

$$\begin{aligned} &\sum_{t \in T_j} y_t^* - \mathbb{E}^{\pi}[y_t^{\pi}] \\ &= \underbrace{\left\{ \sum_{t \in T_j} y_t^* - \max_n \sum_{t \in T_j} y_t^n \right\}}_{\text{Term 1}} + \underbrace{\left\{ \max_n \sum_{t \in T_j} y_t^n - \mathbb{E}^{\pi}[y_t^{\pi}] \right\}}_{\text{Term 2}} \end{aligned}$$

Step 2: (Bound Term 1 and Term 2 respectively)

(1) Based on EXP4 [Auer et al., 2002b], Term 2 can be bounded as follows

$$\begin{aligned}\left\{\max_n \sum_{t\in T_j} y_t^n - \mathbb{E}^\pi[y_t^\pi]\right\} &\le 2\sqrt{(e-1)G_{max}(\Delta T)A\log N} \\ &\le 2\sqrt{(e-1)\Delta_T A\log N}\end{aligned} \tag{3.8}$$

where $G_{max}(\Delta T)$ indicates the maximum value of $\sum_{t\in T_j} y_t^i$ and could be bounded by Δ_T because $0 \le y_t^i \le 1$.

(2) Term 1 can be bounded as follows

$$\begin{aligned}\sum_{t\in T_j} y_t^* - \max_n \sum_{t\in T_j} y_t^n = \sum_{t\in T_j}(y_t^* - y_t^{n_0}) \\ \le \Delta_T \max_t\{y_t^* - y_t^{n_0}\} \\ \overset{(a)}{\le} \Delta_T 2V_j\end{aligned} \tag{3.9}$$

where $n_0 = \text{argmax}_n \sum_{t\in T_j} y_t^n$ and inequality (a) holds because otherwise there exits a time $t_o \in T_j$ and

$$y_{t_0}^* - y_{t_0}^{n_0} > 2V_j$$

which indicates

$$y_{t_0}^* - V_j > y_{t_0}^{n_0} + V_j \tag{3.10}$$

Let $n_1 = \text{argmax}_n y_{t_0}^n$. In such case, for all $t \in T_j$, we have

$$y_t^{n_1} \overset{(a)}{\ge} y_{t_0}^{n_1} - V_j \overset{(b)}{>} y_{t_0}^{n_0} + V_j \ge y_t^{n_0}$$

where inequality (a) is based on the definition of V_j and inequality (b) is based on (3.10). Sum the above inequality with respect to t, $\sum_{t\in T_j} y_t^{n_1} > \sum_{t\in T_j} y_t^{n_0}$ contradicts the optimality of n_0.

Step 3: (Sum over epochs)

$$\begin{aligned}R^\pi(V,T) &= \sum_{t=1}^T y_t^* - \mathbb{E}^\pi[\sum_{t=1}^T y_t^\pi] \\ &\overset{(a)}{\le} \sum_{j=1}^J (2\sqrt{e-1}\sqrt{\Delta_T A\log N} + 2V_j\Delta T) \\ &\overset{(b)}{\le} \left(\frac{T}{\Delta_T}+1\right) 2\sqrt{e-1}\sqrt{\Delta_T A\log N} + 2\Delta_T V_T\end{aligned}$$

where inequality (a) is due to the boundness of Term 1 and Term 2; inequality (b) is due to the definition of V_T. Setting $\Delta_T = \lceil (A\log N)^{1/3}(T/V_T)^{2/3}\rceil$, we obtain the bound

$$R^\pi(V,T) \overset{(a)}{\le} \left((2+2\sqrt{2})\sqrt{e-1}+4\right)(A\log N\cdot V_T)^{\frac{1}{3}}(T)^{\frac{2}{3}} \tag{3.11}$$

Algorithm 1 Pseudocode of algorithm EXP4

```
Inputs: $\gamma \in (0,1]$ and $\boldsymbol{w}_n = \mathbf{1}$
    1. get advice vectors $\boldsymbol{\xi}_t^n$
    2. Set $W_t = \sum_{n=1}^{N} w_n(t)$ and for $k = 1,\dots,K$ set
           $$p_k(t) = (1-\gamma)\sum_{i=n}^{N} \frac{w_n(t)\xi_k^n(t)}{W_t} + \frac{\gamma}{K}$$
    3. Draw arm $k_t$ randomly according to the probability $p_1(t),\dots,p_K(t)$
    4. Receive reward $x_{j_t}(t) \in [0,1]$
    5. For $k = 1,\dots,K$ set
           $$\hat{x}_k(t) = \begin{cases} x_k(t)/p_k(t) & \text{if } k = k_t \\ 0 & \text{otherwise} \end{cases}$$
    6. For $n = 1,\dots,N$ set
           $$\hat{y}_t(t) = \boldsymbol{\xi}^n(t)^T \hat{\boldsymbol{x}}(t)$$
           $$w_n(t+1) = w_n(t)\exp(\gamma \hat{y}_i(t)/K)$$
```

where (a) follows from $T \geq K \geq 2$,$T \geq N \geq 2$, and $V_T \in [A^{-1}, A^{-1}T]$. This concludes the proof. □

The result is an upper bound on the regret between our REXP4 policy and the dynamic oracle. As $A = \min\{N,K\}$, it is favourable if either the number of experts N or arms K is small. This also means it is relatively robust to many arms (as in AL, where arms=data points). If V_T is sub-linear in T (total variation in reward grows slower than timesteps), then performance converges to that of the oracle.

3.4 Dynamic Ensemble Active Learning

Based on our REXP4 algorithm for MAB with expert advice, we present DEAL-REXP4 (Dynamic Ensemble Active Learning) for active learning based on REXP4. Our dynamic ensemble learner will update both base learner f_t and active criteria weights $\boldsymbol{w}(t)$ iteratively. More specifically, each ensemble criterion will predict scores $\boldsymbol{s}_t^n$ for all unlabelled instances. We use exponential ranking normalisation $-\exp(-\alpha\ rank)$ to avoid the issue of different criterion scales, and apply the Gibbs measure $\frac{\exp(-\beta \boldsymbol{s}_t^n)}{\sum_k \exp(-\beta s_{t,k}^n)}$ where the parameters α, β control the sharpness of the distribution. The *rank* denotes

Algorithm 2 Pseudocode of algorithm REXP4

```
Inputs: $\gamma \in (0,1]$ and an epoch size $\Delta_T$
  1. Set Epoch index $j = 1$
  2. Repeat while $j \leq \lceil T/\Delta_T \rceil$
       • Set $\tau = (j-1)\Delta_T$
       • Initialisation: for any expert $n$ set weight $w_n(t) = 1$
       • Repeat for $t = \tau+1, \dots, \min\{T, \tau+\Delta_T\}$, Call EXP4 Algorithm[Auer
         et al., 2002b]
       • Set $j = j+1$ and return to the beginning of step 2
```

the ranking position of the instance's score where the ranking order is determined by the criterion strategy's ordering. For example, the entropy criterion prefers points with maximum entropy, so the maximum entropy point has rank 1. Similarly, the minimum margin criterion prefers points with low distance to margin, so the minimum distance point has rank 1. Based on the current suggestions from the criteria members, the active learning ensemble will select an instance for label querying $p_k(t) \sim b_{LALO}(\mathcal{L}, \mathcal{U}, f; \boldsymbol{w}(t))$. Then, the base learner f_{t+1} will be updated with the new labelled data and the ensemble parameter $\boldsymbol{w}(t+1)$ will be updated successively based on the performance improvement of the updated base learner. To learn the non-stationary reward distribution, we use our proposed REXP4 algorithm to learn the weights of active learning criteria in an online adaptive way by introducing the restart scheme. Giving the current within-batch index $\tau \in \{1, \cdots, \Delta_T\}$, the restart scheme will be activated when $\tau > \Delta_T$, otherwise updates follow the EXP4 rule. The details are described in Algorithm 3 with an illustration in Figure 3.4.

In DEAL-REXP4 we set the reward as the resulting accuracy after a classifier update. Thus in the context of active learning, the bound given in Equation. 3.7 means that we know that the total area under the reward curve obtained by DEAL-REXP4 is within a bound of the best case scenario that would occur only *if we had known the best criterion to use at each iteration.* Moreover, if the variation budget V_T grows sub-linearly with T, DEAL-REXP4 converges towards this best-expert-per-iteration upper bound scenario.

Algorithm 3 DEAL: Dynamic Ensemble Active Learning via REXP4

```
Inputs: $\gamma \in (0,1]$, initial weight $\boldsymbol{w}(1) = 1$, $\Delta_T = 10$, $\tau = 1$,labelled set $\mathcal{L}_0$, unlabelled set $\mathcal{U}_0$, initial classifier $f_0$
for $t = 1 \to T$ do
  1. Get scores of instance $\boldsymbol{s}_t^n$ from criteria
  2. Normalised the score vector $\boldsymbol{s}_t^n = -\exp(-\alpha\, \boldsymbol{rank})$
  3. Obtain the advise vector with $\boldsymbol{\xi}^n(t) = \frac{\exp(-\beta \boldsymbol{s}_t^n)}{\sum_k \exp(-\beta s_{t,k}^n)}$
  4. Set $W_t = \sum_{n=1}^N w_n(t)$ and for $k = 1,\ldots,K$ set
       $$p_k(t) = (1-\gamma)\sum_{i=n}^{N} \frac{w_n(t)\xi_k^n(t)}{W_t} + \frac{\gamma}{K}$$
  5. Query the label of instance $\boldsymbol{x}_{k_t}$ randomly from $\mathcal{U}_t$ according to probability $p_1(t),\ldots,p_K(t)$
  6. Move the instance $\boldsymbol{x}_{k_t}$ from $\mathcal{U}_t$ to $\mathcal{L}_t$
  7. Retrain the classifier $f_t$ and receive reward $r_t^k \in [0,1]$
  8. For $k = 1,\ldots,K$ set
       $$\hat{r}_k(t) = \begin{cases} r_k(t)/p_k(t) & \text{if } k = k_t \\ 0 & \text{otherwise} \end{cases}$$
  9. For $n = 1,\ldots,N$ set
       $$\hat{y}_t^n = \boldsymbol{\xi}^n(t)^T \hat{\boldsymbol{r}}(t)$$
       $$w_n(t+1) = w_n(t)\exp(\gamma \hat{y}_t^n / K)$$
  10. $\tau = \tau + 1$
  if $\tau > \Delta_T$ then
    Reset $\tau = 1$ and $\boldsymbol{w}(t+1) = \boldsymbol{1}$
  end if
end for
```

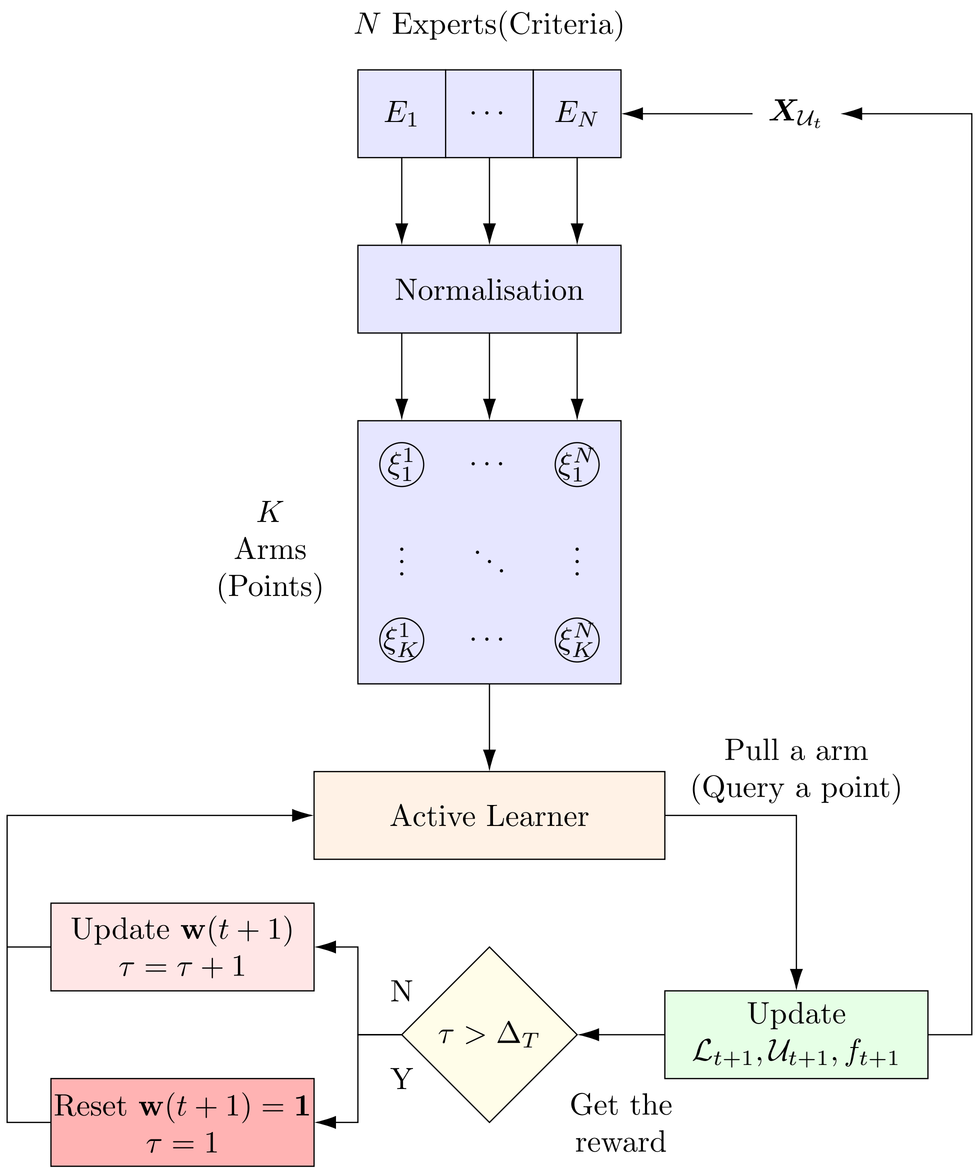

N Experts(Criteria)
E_1
$\cdots$
E_N
$\boldsymbol{X}_{\mathcal{U}_t}$
Normalisation
K Arms (Points)
ξ_1^1
ξ_1^N
ξ_K^1
ξ_K^N
Active Learner
Pull a arm (Query a point)
Update $\mathbf{w}(t+1)$ $\tau = \tau + 1$
N
$\tau > \Delta_T$
Y
Update $\mathcal{L}_{t+1}, \mathcal{U}_{t+1}, f_{t+1}$
Reset $\mathbf{w}(t+1) = \mathbf{1}$ $\tau = 1$
Get the reward

Figure 3.4: Illustration of DEAL system. Light blue: Taking the unlabelled set $\boldsymbol{X}_{\mathcal{U}_t}$ as the input, each expert will output a score that is normalised before input to the DEAL active learner. ξ_K^N is the Nth criterion score of Kth instance. Orange: the active learner to make a decision. Green: updating the labelled set, unlabelled set, and the classifier. Light yellow: The restart detection scheme. Ensemble weights are then updated differently between (light red) or at (dark red) restarts.

3.4.1 Discussion of Static and Dynamic Active Learning

We divide active learning algorithms into static/dynamic based on the stationary/non-stationary assumption on the importance of each criteria over different time periods.

Static Active Learning: Single criterion algorithms are all static, since they solve active learning with only one criterion. Regarding active learning algorithms with multiple motivations: if they are formalised as a single fixed mixture of criteria, they are also static. Since the coefficients of different motivations are fixed over all time steps, they assume that a single weighted combination is suitable at any learning stage. For example, Query Informative and Representative Examples (QUIRE) [Huang et al., 2010], Learning Active Learning (LAL) [Konyushkova et al., 2017b], and Discriminative and Representative Queries for Batch Mode Active Learning (BMDR) [Wang and Ye, 2015] are static active algorithms with multiple motivations.

Previously proposed ensemble algorithms ALBL [Hsu and Lin, 2015], COMB [Baram et al., 2004], and Linear Strategy Aggregation (LSA) [Chu and Lin, 2016] are also static in the sense that, although the weight proportion of their ensemble members changes as data is gathered, their underlying bandit learner is a stationary one, assuming there is only one best expert or best linear combination over all time.

Dynamic Active Learning: In our dynamic active learning research question, we avoid a stationarity assumption on criteria importance over time. A non-stationary algorithm should adapt its weighting proportions over time in response to learning progress. Prior attempts proposed heuristics for classifier switching or reweighting [Donmez et al., 2007; Hospedales et al., 2013] between density and uncertainty sampling. Our DEAL-REXP4 improves on these in that it can use an arbitrary number of criteria of any type beyond 2 specified criteria; and in contrast to prior heuristics, it contains a principled underlying learner with theoretical guarantees. We provide a summary of related prior active learning algorithms in Table 3.1, where the generality and strong notion of regret in DEAL-REXP4 is clear.

3.5 Synthesis Experiment

Firstly, we investigate the effectiveness of REXP4 on the synthetic bandit learning problem. We synthesise a two armed bandit problem with two deterministic experts. Each expert consistently suggests to pick a particular arm over the total time horizon $T = 10000$. This is equivalent to the setting of two armed bandit learning problem

Table 3.1: The summary of active learning algorithms

Single Criterion						
Algorithm	Motivation		Stationarity	Importance of Criterion	Ensemble Members	Property
US [Lewis and Gale, 1994] [Settles, 2009; Joshi et al., 2009]	Querying the least confidence		Stationary	Fixed	US	Static
RS [Xu et al., 2003]	Query a cluster within Margin		Stationary	Fixed	RS	Static
DE [Donmez et al., 2007]	Query the major cluster		Stationary	Fixed	DE	Static
Multiple Criteria						
Algorithms	Motivations		Stationarity	Importance of Criterion	Ensemble Members	Property
QUIRE [Huang et al., 2010]	Combining informativeness and representativeness		Stationary	Equal effect	QUIRE	Static
BMDR [Wang and Ye, 2015]	Combining discriminative and representativeness		Stationary	Equal effect	BMDR	Static
LAL [Konyushkova et al., 2017b]	Combining Multiple motivations		Stationary	Equal effect	Any Criteria	Static
DUAL[Donmez et al., 2007]	Switching from DE to US once		Non-stationary	Varying	US, DE	Dynamic
ALGD [Hospedales et al., 2013]	Switching between DE to US		Non-stationary	Varying	US, DE	Dynamic
Bandit Ensemble Algorithms						
Algorithm	Bandit	Regret	Stationarity	Importance of Criterion	Ensemble Members	Property
COMB [Baram et al., 2004]	EXP4 [Auer et al., 2002b]	$\max_n \sum_{t=1}^{T} y_t^n - \mathbb{E}(\sum_{t=1}^{T} y_t^{\pi})$	Stationary	Single best	Any Criteria	Static
ALBL [Hsu and Lin, 2015]	EXP4.P [Beygelzimer et al., 2011]	$\max_n \sum_{t=1}^{T} y_t^n - \mathbb{E}(\sum_{t=1}^{T} y_t^{\pi})$	Stationary	Single best	Any Criteria	Static
LSA [Chu and Lin, 2016]	LinUCB [Chu et al., 2011]	$\sum_{t=1}^{T} r_{a_t^*}(t) - \sum_{t=1}^{T} r_{a_t}(t)$	Stationary	Single best combination	Any Criteria	Static
DEAL	REXP4	$\sum_{t=1}^{T} \max_n y_t^n - \mathbb{E}(\sum_{t=1}^{T} y_t^{\pi})$	Non-Stationary	Dynamic best	Any Criteria	Dynamic

without expert advice. As illustrated in the Figure 3.5, the reward of both arms are under the sine function with a fixed variation budget. This means that the arm with maximum reward will periodically switch with a fixed batch size $\Delta_T = 100$.

In this experiment, we compare the actual regret of multiple batch size settings of REXP4 with the proposed theoretical regret bound. We collect the actual regret for the proposed bandit algorithm with 3 batch size settings: the ground truth batch size $\Delta_T = 100$ and the corresponding batch sizes based on various variation budget $V_T = A^{-1}$ and $V_T = A^{-1}T$. Here, the batch size based on variation budget is defined as $\Delta_T = \lceil (A \log N)^{1/3} (T/V_T)^{2/3} \rceil$. This means that the batch size with $V_T = A^{-1}$ is an infrequent restart REXP4 and $V_T = A^{-1}T$ is the random REXP4 due to the high-frequency rate of restarting the algorithm. The theoretical regret bound is based on the Equation 3.7 where the term $V_T \in [A^{-1}, A^{-1}T]$ could monotonically increase the numerical value of the bound. Therefore, we refer the bound with $V_T = A^{-1}$ as the tightest bound and the bound with $V_T = A^{-1}T$ as the loosest bound.

According to the Figure 3.5, if the ground truth of batch size $\Delta_T = 100$ is provided, REXP4 is able to learn a strategy to achieve less regret than the tightest and loosest bound. Besides, the figure shows that actual regret with the batch size based on $V_T = A^{-1}$ and $V_T = A^{-1}T$ are similar but both of them are worse than the tightest regret. This suggests that the choice of the batch size Δ_T has a significant impact on REXP4 performance. Since the ideal variation budget setting of REXP4 is still the open question of the non-stationary bandit learning problem, we will empirically set the fixed batch size for the active learning experiment.

3.6 Active Learning Experiments

To evaluate our algorithm, we use 13 datasets from UCI[2] and LibSVM[3] [Chang and Lin, 2011] repositories. These datasets are selected following previous relevant papers [Chu and Lin, 2016; Hsu and Lin, 2015; Huang et al., 2010; Chattopadhyay et al., 2012]. We use linear SVM [Fan et al., 2008] as the base learner. If the datasets do not include a pre-defined training/testing split, we randomly split 60% for training and the rest for testing. In each trial, we start with 1 randomly labelled point per class. Each experiment is repeated 200 times and the average testing accuracy is reported.

[2]https://archive.ics.uci.edu/ml/datasets.html
[3]https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/binary.html

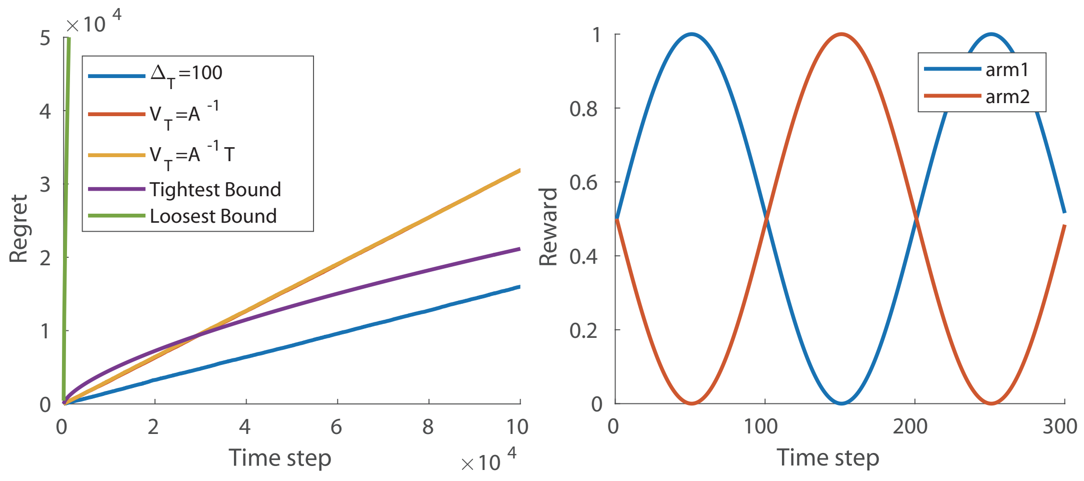


Figure 3.5: The result and reward distribution of the synthesised experiment. Left: the actual regret of the REXP4 with various setting of the batch Δ_T (blue/red/yellow) and the regret bound according to the equation 3.7. Right: The two arms reward distribution.

Dataset Property A dataset with stationary reward distribution would tend to have a consistent winner, and vice-versa. Although (non)stationarity is a continuum, we will describe a dataset as stationary if at least two criteria have a fraction of wins above threshold 10%.

Criteria Ensemble: The ensemble of base learners includes: *US*: picking the instances with max-entropy (min margin) instance in binary class datasets [Lewis and Gale, 1994; Settles, 2009] or minimum Best-versus-Second-Best (BvSB) [Joshi et al., 2009] in multiclass datasets. *RS*: clustering the points near the margin [Xu et al., 2003] then scoring unlabelled points by their distances to the largest centroid. *Distance-Furthest-First (DFF)*: Focuses on exploration by selecting the furthest unlabeled instance to the nearest labeled instance [Hochbaum and Shmoys, 1985]. We use *DFF* [Hochbaum and Shmoys, 1985] to replace the RS in multiclass datasets as originally RS is designed for binary class datasets. Both are motivated by exploring the datasets, but DFF does not depend on binary classifiers. *Density Estimation (DE):* Picking the instance with maximum density in a GMM with 20 diagonal covariance components [Donmez et al., 2007]. *RAND:* Randomly selecting points can be hard to beat on datasets unsuited to a given criterion. Moreover, including a random expert (for exploration) is necessary to guarantee the performance of the EXP4 subroutine [Auer et al., 2002b; Beygelzimer et al., 2011].

Competitors: We compare our method to ALBL [Hsu and Lin, 2015], COMB [Baram et al., 2004] and DUAL [Donmez et al., 2007]. For COMB, we follow their recommended settings with CEM reward and $\beta = 100$. For the ALBL, we use their settings and importance-weighted accuracy reward.

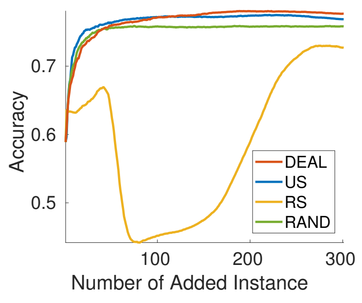
For direct comparison, ALBL, COMB and REXP4 use the same ensemble of criteria described above. DUAL is engineered for a specific pair of criteria, so we apply its original version using Uncertainty Sampling and Density-Weighted Uncertainty Sampling. It is also only defined for binary classification problems unlike the others.

DEAL-REXP4 Settings: For reward, we follow [Hsu and Lin, 2015; Chu and Lin, 2016] in using the IWA for unbiased estimation of test accuracy. To produce probabilistic preferences for points from all AL criteria, we use exponential ranking normalisation and a Gibbs measure with $\alpha = 0.1, \beta = 100$. We use batch size $\Delta_T = 10$ throughout. The choice $\Delta_T = 10$ is based on observing the typical coarse duration of performance gaps among different criteria. For example, RS wins first 20 iterations in Figure 3.6(b). The reason for parameterizing in terms of Δ_T rather than V_T is that it has intuitive meaning in AL context (batch-size), yet implies a corresponding variation budget for any given T (Theorem 1).

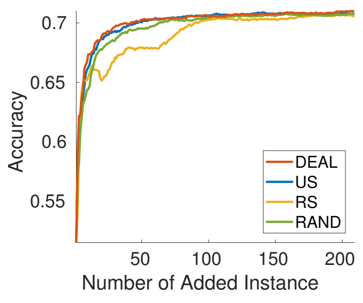
Characterising Dataset Stationarity: We first investigate each dataset to characterise its (non)stationarity. We use our DEAL trajectory, and use an oracle to measure the % wins of each criterion at each batch Δ_T in terms of performance increase. A dataset with stationary reward distribution would tend to have a consistent winner, and vice-versa. Although (non)stationarity is a continuum, we will describe a dataset as stationary if at least two criteria have a fraction of wins above threshold $\theta = 10\%$.

Table 3.2: Characterising datasets as (S) stationary or (NS) non-stationary according to the win proportion of the criteria.

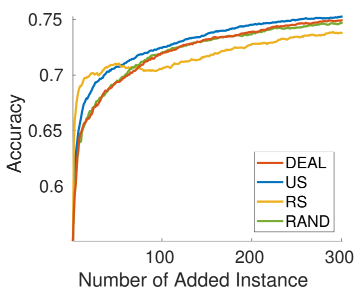
Dataset	Num of Instances	Num of Classes	Classes Proportion	US	RS/DFF	Density	RAND	NS/S
austra	690	2	0.55/0.45	0.46	0.13	0.29	0.12	NS
breast	683	2	0.65/0.35	0.88	0.08	0.04	0.00	S
diabetes	768	2	0.65/0.35	0.74	0.11	0.11	0.04	NS
fourclass	862	2	0.65/0.35	0.86	0.00	0.10	0.04	NS
german	1000	2	0.70/0.30	0.52	0.24	0.14	0.10	NS
haberman	306	2	0.74/0.26	0.30	0.20	0.30	0.20	NS
heart	270	2	0.56/0.44	0.45	0.33	0.22	0.00	NS
ILPD	583	2	0.71/0.29	0.35	0.45	0.05	0.15	NS
liver	345	2	0.42/0.57	0.75	0.08	0.00	0.17	NS
monk1	556	2	0.5/0.5	1.00	0.00	0.00	0.00	S
wdbc	569	2	0.63/0.37	0.90	0.10	0.00	0.00	S
wine	178	3	0.33/0.40/0.27	1	0.00	0.00	0.00	S
letter	15500	26	Almost uniform	0.75	0.25	0.00	0.00	NS



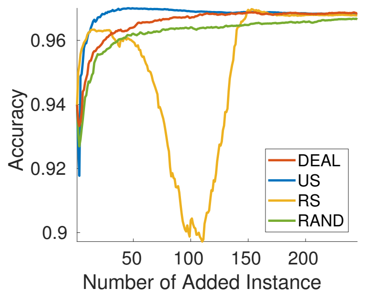
(a) fourclass



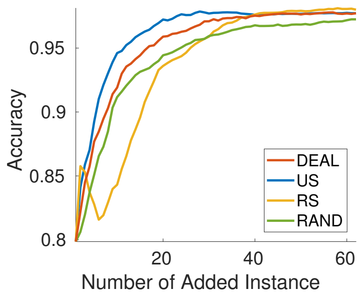
(b) ILPD



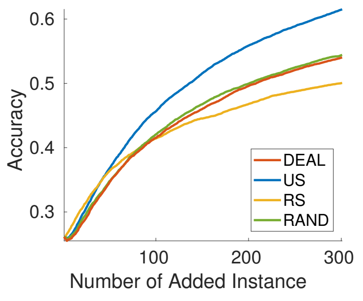
(c) german



(d) breast



(e) wine



(f) letter

Figure 3.6: Comparison of DEAL-REXP4 versus individual ensemble members.

Tab. 3.2 summarises the datasets, and we can see that there is a mix of stationary and non-stationary datasets.

3.6.1 Dynamic Ensemble Active Learning vs Conventional Criterion

Examples comparing the performance of DEAL and individual criteria in the ensemble are shown in Figure 3.6. There is no single criterion that works best for all datasets, moreover different criteria are effective at different stages of learning. While DEAL is not best across all datasets and all time-steps (this would require the actual dynamic oracle upper bound), it performs well overall. This is summarised quantitatively across all 13 datasets in Table 3.3. Each method's performance is evaluated by the area under the learning curve at different proportions of added instances. The results show the number of wins/ties/losses of DEAL versus the alternative ensemble member of specified highest rank according to two-sided t-test. This shows for example that DEAL often ties with the top-ranked ensemble member (30 draws vs 1st rank), is usually at least as good as the second ranked member (50 wins and 45 ties vs only 35 losses) and is never the worst (0 losses vs 4th rank).

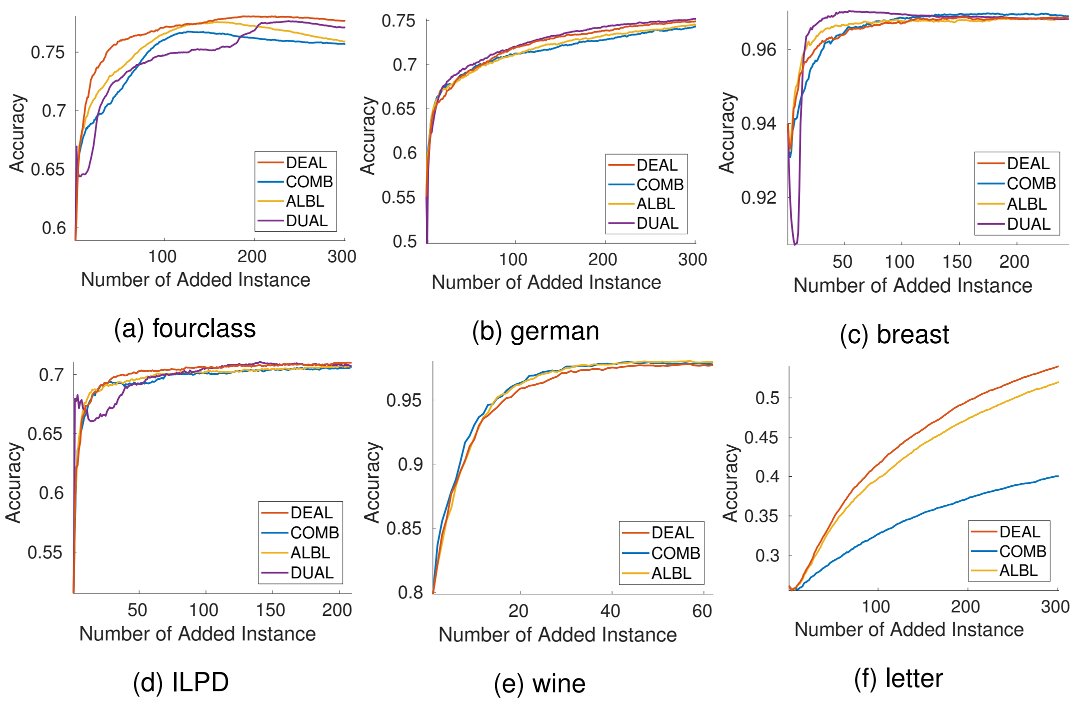


Figure 3.7: Comparison of active learning with our DEAL-REXP versus alternative state of the art bandit algorithms.

Table 3.3: Win/Tie/Loss counts of DEAL-REXP4 versus ensemble members in terms of AUC at specified learning stage.

Rank	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%	Total
1st	0/4/9	0/3/10	0/3/10	0/2/11	0/2/11	0/4/9	0/4/10	0/3/10	0/3/10	0/3/10	0/30/100
2nd	2/6/5	4/5/4	4/6/3	5/4/4	6/3/4	6/3/4	6/3/3	6/4/3	6/4/3	5/6/2	50/45/35
3rd	7/5/1	7/4/2	7/5/1	7/5/1	7/4/2	7/4/2	7/6/0	7/6/0	8/5/0	8/5/0	72/49/9
4th	11/2/0	12/1/0	13/0/0	13/0/0	13/0/0	13/0/0	13/0/0	13/0/0	13/0/0	13/0/0	127/3/0
Total	20/17/15	23/13/16	24/14/14	25/11/16	26/9/17	26/11/15	26/13/13	26/13/13	27/12/13	26/14/12	249/127/144

3.6.2 Dynamic Ensemble Active Learning vs Ensemble Learner

We compare our DEAL-REXP4 with state-of-the-art alternatives to tuning an AL-ensemble. Sometimes DUAL performs well, but it is highly variable depending on whether the criterion switch heuristic makes a good choice or not, as seen in Figure 3.7. Note that DUAL applies to binary classification problems only so it is not evaluated in Figure 3.7(e,f). Similar to the Table 3.3, we also apply the two-sided t test to compare the ensemble AL algorithms. Table 3.4 summarises the results across all datasets in terms of AUC wins/draws/losses of each approach against the alternatives. DUAL has a lower row-total as it is defined for binary problems only, so not evaluated on wine and letter datasets. The main observation is that DEAL outperforms the alternatives particularly on non-stationary datasets. On stationary datasets we are only

Table 3.4: Win/Tie/Loss counts of DEAL-REXP4 and state of the art alternatives at specified learning stages.

Algorithm	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%	Total
Non-Stationary Datasets											
ALBL	8/15/3	9/13/4	9/12/5	9/11/6	7/13/6	6/14/6	6/15/5	4/16/6	4/16/6	4/16/6	66/141/53
COMB	4/13/9	2/14/10	2/13/11	2/12/12	1/12/13	2/12/12	2/13/11	1/15/10	0/16/10	0/16/10	16/136/108
DUAL	7/9/8	7/7/10	7/7/10	8/6/10	9/8/7	8/8/8	6/11/7	7/12/5	7/14/3	7/15/2	73/97/70
DEAL	6/15/5	9/14/3	11/12/3	12/11/3	12/11/3	12/12/2	11/13/2	12/11/3	11/12/3	10/13/3	**106/124/30**
Stationary Datasets											
ALBL	4/6/1	6/3/2	7/3/1	6/3/2	7/2/3	6/3/2	6/3/2	5/5/1	5/5/1	4/6/1	**56/39/15**
COMB	2/4/5	2/2/7	1/4/6	1/2/8	1/2/8	1/2/8	1/3/7	1/4/6	1/4/6	1/4/6	12/31/67
DUAL	0/2/7	3/1/5	3/3/3	5/2/2	5/2/2	5/2/2	4/4/1	4/4/1	4/4/1	4/4/1	37/28/25
DEAL	7/4/0	6/2/3	3/4/4	4/3/4	4/2/5	4/3/4	3/4/4	2/5/4	2/5/4	2/6/3	37/38/35

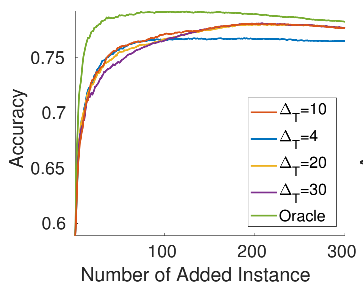
slightly worse than ALBL. This is expected as REXP4 performs forgetting in order to adapt to changes in expert efficacy, meaning that we cannot exploit the best criterion as aggressively as ALBL's EXP4.P MAB learner. Nevertheless, overall DEAL is fairly robust to stationary datasets (small margin behind ALBL), while ALBL is not robust to non-stationary datasets (larger margin behind DEAL).

3.6.3 Dynamic Ensemble Active Learning vs Random REXP4

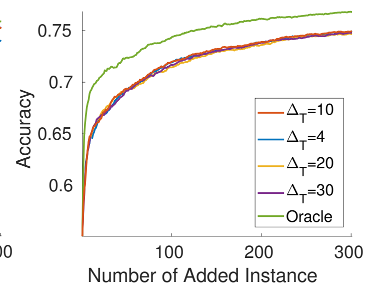
We further explore the performance of DEAL-REXP4 with various batch size $\Delta_T = \{4, 10, 20, 30\}$ and show performance gap between the DEAL-REXP4 and the dynamic oracle. Dynamic oracle $\text{argmax}\, y_t^*$ is to randomly select a data point based on the expert with maximum expected test accuracy increment. As seen in Figure 3.8, DEAL-REXP4 with $\Delta_T = \{10, 20, 30\}$ has similar performance on all datasets. But the smaller batch size $\Delta_T = 10$ DEAL-REXP4 is slightly better than the proposed algorithm with batch size $\Delta_T = \{20, 30\}$ in Figure 3.8 (a,f). Though DEAL-REXP4 with $\Delta_T = 10$ has high-frequency rate to restart, it is more robust than the batch size $\Delta_T = 4$ in Figure 3.8(a,c,e). This indicates that DEAL-REXP4 can learn the non-stationary AL information within the short time interval rather than relying on completely random selection.

3.7 Summary

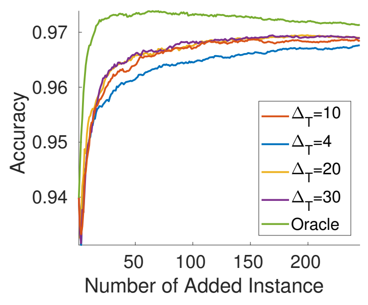
We proposed a non-stationary multi-armed bandit with expert advice algorithm REXP4, and demonstrated its application to online learning of a criterion ensemble in active learning. The theoretical results provide bounds on REXP4's optimality. The empir-



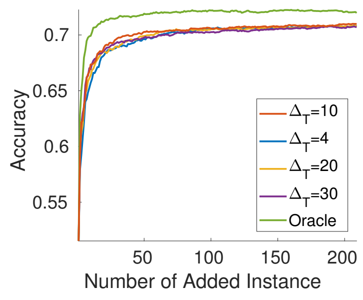
(a) fourclass



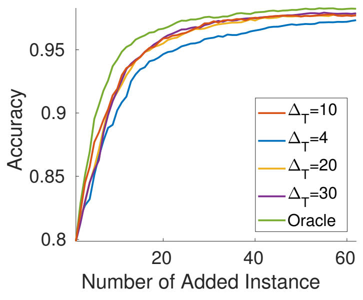
(b) german



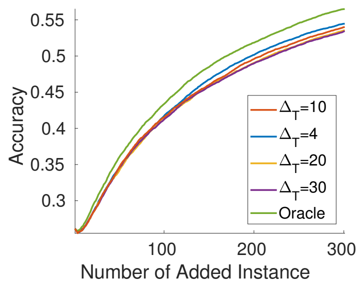
(c) breast



(d) ILPD



(e) wine



(f) letter

Figure 3.8: Comparison of active learning with our DEAL-REXP versus alternative state of the art bandit algorithms.

ical results show that active learning with DEAL-REXP4 tends to perform near the best criterion in the ensemble. It performs comparable to state of the art alternative ensembles on stationary datasets, and outperforms them on non-stationary datasets.

Chapter 4

Learning a Transferable Active Learning Policy

In Chapter 3, we propose an online learning technique to learn to dynamically select the best active learning criterion. Although the proposed technique is able to improve the data annotation process in terms of efficiency, the scheme relies entirely on online learning, and must inefficiently perform exploration for the best criterion repeatedly on each dataset. There is no storage and re-use of experience. This re-exploration requires extra effort to find the next best criterion. In addition, this technique will be sensitive to the chosen ensemble criteria. It depends on the assumption that there is at least one criterion that is ideally suited for a given dataset over a given time period.

In this chapter, we present an end-to-end meta learning approach to learn a general active learning criterion from previous experiences that can be generalised across datasets. This relieves the reliance on the carefully designing and assembling the criteria for the previous approach. The main contribution of this chapter is to learn an active learning criterion end-to-end by effectively extracting the information from raw data, and moreover to do so in a way that produces a cross-dataset transferable criterion.

4.1 Introduction

In many applications, supervision is costly relative to the data volume. In this setting, active query selection methods can be invaluable to predict which instances a base classifier would find it informative to label. By carefully choosing the training data, a classifier can perform well even with relatively sparse supervision. This vision has motivated a large body of work in active learning that has collectively proposed dozens of query criteria based on different theoretical or intuitive motivations , such as margin [Tong and Koller, 2002] and uncertainty-based [Kapoor et al., 2007] sampling, expected error reduction [Roy and McCallum, 2001], representative and diversity-based [Chattopadhyay et al., 2012] sampling, or combinations thereof [Hsu and Lin, 2015]. As illustrated in Chapter 3, it is difficult to choose a clear winner from these, because each is based on a reasonable and appealing but completely different – motivation; and there is no one that consistently outperforms the others on all datasets.

Rather than hand-designing an intuitive criterion and hoping that it performs well, we propose taking a data-driven learning-based approach. We treat active learning algorithm development as a meta-learning problem and train an active learning policy represented by a neural network using deep reinforcement learning (DRL). It is natural to represent AL as a sequential decision making problem since each action (queried point) affects the context (available query points, state of the base learner) successively for the next decision. In this way the active query policy trained by RL can potentially learn a powerful and non-myopic policy. By treating the increasing accuracy of the base learner as the reward, we optimise for the final goal: the accuracy of the classifier. As the class of deep neural network (DNN) models we use includes many classic criteria as special cases, we can expect this approach should be at least as good as existing methods and likely better due to exploiting more information and non-myopic optimisation of the actual evaluation metric. This idea of learning the best criterion within a general function class is appealing. A recent study [Bachman et al., 2017], although similarly inspired, was not able to provide a general solution to AL since the learned criterion is not generalisable across diverse datasets/learning problems [Bachman et al., 2017]. With DRL we can likely learn an excellent query policy for any given dataset, on the condition that all labels are provided.

4.2 Related Work

4.2.1 Reinforcement Learning

Reinforcement Learning (RL): Reinforcement learning leverages the formal framework of Markov decision processes (MDP) to define the interaction between an agent and its environment in terms of states, actions, and rewards [Sutton and Barto, 1998]. More specifically, an agent interacts with an environment $\mathcal{E}$ over a number of discrete time steps t. At each time step, the agent receives the state $s_t \in \mathcal{S}$ from the environment

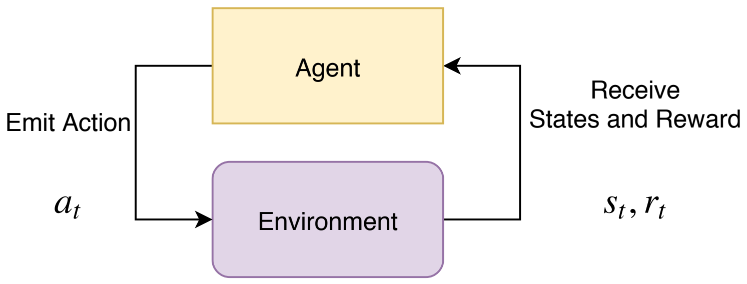


Figure 4.1: The illustration of reinforcement learning

and selects an action $a_t \in \mathcal{A}$ based on its policy $\pi(a_t|s_t)$ which is a mapping from state to action. The agent then receives a new state s_{t+1} and immediate reward r_t from $\mathcal{E}$. The aim of RL is to maximise the return $R = \sum_{t=1}^{\infty} \gamma^{t-1} r_t$ which is the accumulated immediate rewards with discount factor $\gamma \in (0, 1]$.

Connection between MAB and RL: Recall from Chapter 3 that MAB is a special case of RL. Different from MAB, the agent in RL receives both an updated state s_{t+1} and reward r_t by interacting with the environment $\mathcal{E}$. The updated state follows the state-action transition probability $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)$, wherein the transition probability characterises the dynamics of a finite MDP. Since MAB only receives rewards after pulling an arm, and without changing the state of the environment, MAB could be viewed as a one-step MDP which is simpler than the full RL problem. In addition, the updated state information s_{t+1} provides a powerful information to help the agent to make a wiser decision by considering the current state. Though the related methods of contextual bandits or the bandits with expert advice are proposed to improve MAB via giving additional information, this contextual information only describes the arms and provides a weak representation of a complex environment. Lastly, the aim of RL is to learn the future accumulated rewards whereas the MAB only learns which is the best action, as there is no evolving state. This means that the MAB aims to maximise the one-step best reward myopically and the RL learns to maximise long term non-myopic decision for complex problems. Therefore, RL solves a more general problem than the MAB models.

Policy Optimisation of Reinforcement Learning: There are multiple approaches to learn the policy π, and most recent approaches exploit RL with function approximation to achieve excellent performance in various applications [Kober and Peters, 2009; Mnih et al., 2015]. In this thesis, we use direct policy search based RL, which learns π by gradient ascent on the objective function $J_\pi(\theta) = \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_\theta(a|s) R(s, a)$, where $d(s)$ is stationary distribution of Markov chain for π_θ and the expected future reward $R(s,a) = \mathbb{E}\{r_{t+1}|s_t = s, a_t = a\}$ [Williams, 1992]. This policy-gradient based approach is practical in variety of fields since it only assumes the availability of the gradient $\nabla J_\pi(\theta) = \nabla_\theta \log \pi_\theta(a|s) R(s,a)$, and a finite-step problem.

Baseline function for Policy-Gradient: However, one problem with policy gradient is the high variance that exists as a result of the large numerical scale of return. This large-scale problem is caused by the Monte Carlo sampled long-term trajectory. One common approach for reducing the variance is to standardise return by subtracting the average return without biasing the gradient estimator $\nabla_\theta \log \pi_\theta(a|s)(R(s,a) - b)$ [Williams, 1992], where b denotes the baseline function. Subsequently, the baseline function could be viewed as a control variate problem, for which a control variate method is a variance reduction technique used in Monte Carlo methods [Greensmith et al., 2004; Peters and Schaal, 2008; Zhao et al., 2011]. The optimal baseline can be obtained by minimising the variance in the gradient estimates with respect to the baseline.

$$b^*_{\mathrm{PG}} := \arg\min_b \mathbf{Var}[\nabla_\theta J_\pi(\theta)] = \frac{\mathbb{E}\left[R(s_t,a_t)||\sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_t|s_t)||^2\right]}{\mathbb{E}\left[||\sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_t|s_t)||^2\right]} \tag{4.1}$$

Another variance reduction scheme is to use a function approximator to model the baseline function to reduce the variance [Mnih et al., 2016]. However, this baseline is a models estimate; therefore, although it may be low variance, it introduces instability during training. This instability makes some RL tasks even more complicated. In this thesis, we use a simple return standardisation method to reduce variance.

$$\bar{b}_{\mathrm{PG}} := \frac{\mathbb{E}\left[R(s,a)\right]}{\mathbb{E}\left[(R(s,a) - \mathbb{E}\left[R(s,a)\right])^2\right]} \tag{4.2}$$

Connection between RL and AL: We propose to model an active learning algorithm as a neural network, and formalise discovery of the ideal criterion as a deep reinforcement learning problem. Let the state of the world s_t consist of a featurisation of the dataset and the state of the base classifier $s_t = \{\mathcal{L}_t, \mathcal{U}_t, f\}$. Let an active learning criterion be a policy $\pi(a_i|s)$ where the action index $i \in \{1, \ldots, |\mathcal{U}|\}$ selects a point in the unlabelled set to query. Upon querying a point the world state is updated to s_{t+1} as that point is moved from $\mathcal{U}$ to $\mathcal{L}$ and f is updated as the base classifier is retrained. Assume the policy is a neural network paramaterised by weights θ that selects actions as $\pi(a_i|s_t) \propto \exp^{\Phi_\theta(a_i,s_t)}$, where $i \in \{1, \ldots, |\mathcal{U}|\}$ is the index of the unlabelled instances. Finally we define the return $J(\theta)$ of an episode, which is the quantity we wish to maximise. For example, if the budget is N queries and we only care about the accuracy after the Nth query, then we let $J(\theta) = Acc_N$ where Acc_N is the accuracy after the Nth query. Alternatively, if we care about the performance during all the N queries, we can use $J(\theta) = \sum_{t=1}^{N} \gamma^{t-1} Acc_t$. (This illustrates an important advantage of the learning to do active learning approach: we can tune the learned criterion to suit the requirements of the AL application.) In interpreting AL criterion learning as a DRL problem, there is the consideration that unlike general RL problems, each action can only be chosen once in an episode. We will achieve this by defining a fully convolutional policy network architecture where the dimensionality of the output softmax $\pi(a_i|s_t)$ can vary with t. In this chapter, our proposed method is focus on binary classification problems, i.e., $C = 2$.

4.2.2 Related Methods

Active Learning by Learning: A few works have very recently also treated finding an AL criterion as a learning problem. [Konyushkova et al., 2017b] proposes to learn a criterion based on a vector of expert features (e.g., classifier confidence, label imbalance). However by using expert features, this misses the chance to learn the representation from raw features as in our approach; and by using supervised rather than RL to train the policy, it is not optimally non-myopic. [Bachman et al., 2017] and [Woodward and Finn, 2017] use RL to train a single model that provides both the base classifier and the active learner. This tight integration has a drawback of being constrained to a specific base learner, so losing the ability to use an arbitrary base learner as per our framework. More importantly, while these methods learn effective non-myopic policies, they are trained and tested on different classes within the same dataset, so the generalisation challenge and evaluation is minimal. There is no mechanism to ensure effective transfer across datasets of different statistics or to allow any transfer at all across datasets of different dimensionalities. Finally, unlike [Woodward and Finn, 2017; Bachman et al., 2017] our framework is agnostic to the base classifier. Treating the underlying learner as part of the environment to be optimised means our framework can be applied to improve the label efficiency of any existing learning architecture or algorithm.

Active Learning Ensembles: Different AL algorithms perform well on different datasets, or at different learning stages. For this reason studies have proposed heuristics to switch criteria from early to late stage learning [Donmez et al., 2007; Baram et al., 2004], or use multi-armed bandit (MAB) approaches to estimate the best criterion for a given dataset within an ensemble [Hsu and Lin, 2015], or both [Pang et al., 2018a]. But aside from being myopic, MAB learners do not learn transferrable knowledge. They perform all their learning within a single rollout, and their need to explore/learn online is fundamentally at odds with active learning. [Chu and Lin, 2016] ameliorates this somewhat with regularisation, but still needs dataset-specific learning. Our approach can address these issues. Besides non-myopic policy learning with RL, a DNN has capacity to encode multiple criteria and apply different ones at different learning stages. By learning a meta-policy that paramaterises a dataset-specific policy, it customises the overall active learning strategy to the target dataset; thus it can transfer knowledge for immediate efficacy on a new dataset without dataset specific learning.

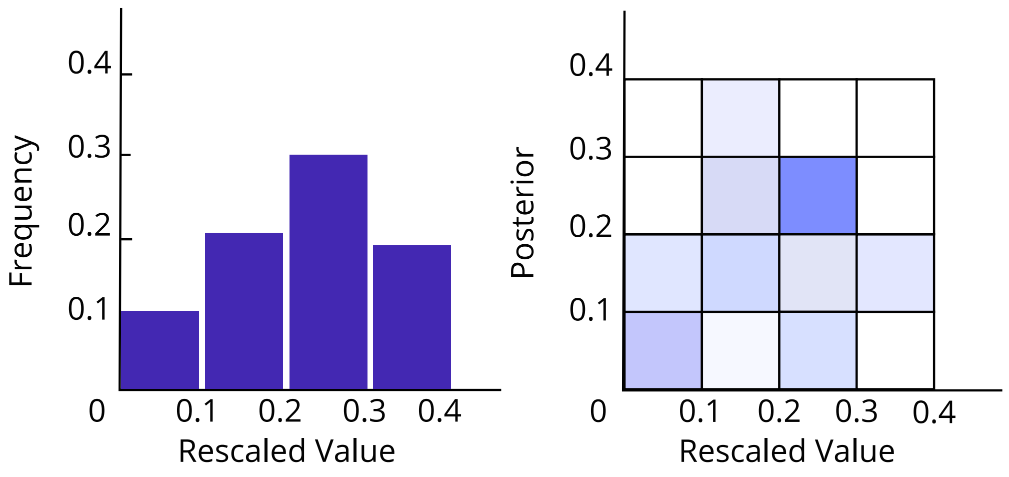
Domain Generalisation and Adaptation: Our task-agnostic AL's goal is related to Domain Generalisaton (DG) [Muandet et al., 2013; Li et al., 2018] and Domain Adpatation (DA) [Ganin and Lempitsky, 2015] in supervised learning in that we would like to train on one dataset and perform well when testing on another dataset. Our framework has aspects of DG (multi-task training to increase generality) and DA (adapting to target data, via dataset embedding meta network) methods. But we are not aware of any dataset embedding approaches to achieving DA within supervised learning.

Parameter Prediction: Models predicting the parameters of other models are increasingly applied [Ha et al., 2017]. In robot control, such 'contextual' or 'paramaterised' policies are used to solve tasks like target-conditional reaching [Kupcsik et al., 2013]. [Romero et al., 2017] used auxiliary networks for parameter reduction but training and testing on the same dataset. To our knowledge this strategy has not been applied for domain adaptation.

4.3 Learning a Transferable Active Query Policy

4.3.1 Non-Myopic Active Learning Policy

Recall that our aim is to obtain the parameters θ of an effective dataset-agnostic active query policy $\pi_\theta(a|s)$. The key challenge is how to learn such a policy given that: (i) the test dataset statistics may be different from training dataset statistics, and moreover (ii) different datasets have different feature dimensionality d. This challenge is addressed by defining the overall policy $\pi_\theta(a|s)$ in terms of two sub-networks – a policy network and a meta network – described as follows.



(a) Representative (b) Discriminative

Figure 4.2: The illustration of embedding

Overall the policy network π inputs all N unlabelled instances $\boldsymbol{Z}_u \in \mathbb{R}^{N\times d}$ and its output is an N-way softmax distribution for selecting which instance to query. We assume the policy models actions via the softmax $\pi(a_i|s) \propto \exp^{\Phi_{\theta_q}(\boldsymbol{W}_e^T \boldsymbol{z}_i)}$, where $\boldsymbol{z}_i \in \mathbb{R}^d$ is the ith unlabelled instance in $\boldsymbol{Z}_u$ and $\boldsymbol{W}_e \in \mathbb{R}^{d\times k}$ encodes the pool of instances. Although dimensionality d varies by dataset, the encoding $\boldsymbol{u}_i = \boldsymbol{W}_e^T \boldsymbol{z}_i \in \mathbb{R}^k$ does not, so the rest of the policy network $\pi(a_i|s) \propto \exp^{\Phi_{\theta_q}(\boldsymbol{u}_i)}$ is independent of d. The key is then how to obtain encoder $\boldsymbol{W}_e$ which will be provided by the meta network. Following previous work [Bachman et al., 2017; Konyushkova et al., 2017b] we also allow the instances to be augmented by instance-level expert features so $\boldsymbol{Z} = [\boldsymbol{X}, \boldsymbol{\xi}(\boldsymbol{X})]$ where $\boldsymbol{X}$ are the raw instances and $\boldsymbol{\xi}(\boldsymbol{X})$ are the expert features of each raw instance.

4.3.2 Meta Learning the Transferability

The encoding parameters $\boldsymbol{W}_e \in \mathbb{R}^{d\times k}$ of the policy network are obtained from the meta network: $\Psi_{\theta_m^e} : \{(\mathcal{L}, \mathcal{U}, f) \rightarrow \boldsymbol{W}_e; \theta_m^e\}$. Following [Romero et al., 2017] we also use the $\boldsymbol{W}_d \in \mathbb{R}^{k\times d}$ dimensional decoder $\Psi_{\theta_m^d} : \{(\mathcal{L}, \mathcal{U}, f) \rightarrow \boldsymbol{W}_d; \theta_m^d\}$to regularise this process by reconstructing the input features. The meta network synthesises these weight matricies based on dataset-embeddings of $\boldsymbol{Z}^T$ described in the following section.

The meta network not only learns to generate parameters appropriate to the *statistics* of a given dataset, but also deals with the heterogeneous dimensionality problem by generating parameters appropriate to the *dimensionality* of the data in the target problem. The idea of meta networks to predict weights for a target network was recently used in [Romero et al., 2017]. There the meta network inputs an embedding of $\boldsymbol{X}^T$ and predicts the weights for a main network that inputs $\boldsymbol{X}$, with the purpose of reducing the total number of parameters if $\boldsymbol{X}$ is high dimensional. In [Romero et al., 2017] all the training and testing are performed on the same dataset. Here we are inspired by this idea in proposing a meta-network strategy for achieving end-to-end learning of multiple-domains. By multi-task training on multiple datasets, the meta-network learns to generate dataset-specific weights for the policy network to ensure its generalisation ability.

Dimension-wise Embedding Strategy: The auxiliary meta-network first builds a dataset size independent dimension-wise embedding of the input $(\mathcal{L}, \mathcal{U}, f)$, as shown in its light blue part of Figure 4.3. Then it predicts $\boldsymbol{W}_e \in \mathbb{R}^{d \times k}$, with

$$(\boldsymbol{W}_e)_j = \Psi\Big([\boldsymbol{e}^1_j(\boldsymbol{Z}^T_u), \boldsymbol{e}^1_j(\boldsymbol{Z}^T_l), \boldsymbol{e}^2_j([\boldsymbol{Z}^T_u, \boldsymbol{Z}^T_l], f)]\Big). \tag{4.3}$$

Here e is a non-linear feature embedding, j indexes features, selecting the jth embedded feature and the jth row of $\boldsymbol{W}_e$, and Ψ is the non-linear mapping of the meta-network, which outputs a vector of dimension k. Similarly, the meta-network also predicts the weight matrix $\boldsymbol{W}_d$ used for auto-encoding reconstruction (Figure 4.3). Although d is dataset dependent, the meta network generates weights for a policy network of appropriate dimensionality ($d \times k$) to the target problem. The specific embeddings used are explained next.

Choice of Embeddings: We use two ‘representative’ and ‘discriminative’ histogram style embeddings. For the *representative* embedding ($\boldsymbol{e}^1_j(\boldsymbol{Z}^T_u)$ and $\boldsymbol{e}^1_j(\boldsymbol{Z}^T_l)$), we encode each feature dimension as a histogram over the instances in that dimension, as illustrated in Figure 4.2(a). Specifically, we rescale the ith dimension features into $[0, 1]$ and divide the dimension into 10 bins. Then we count the proportion of labelled and unlabelled data for each bin. This gives a 1×20 histogram embedding for each dimension that encodes its moments. For the *discriminative* embedding ($\boldsymbol{e}^2_j([\boldsymbol{Z}^T_u, \boldsymbol{Z}^T_l], f)$), we create a 2-D histogram of 10 bins per dimension. In this histogram we count the frequency of instances with feature values within each bin (as per the previous embedding) jointly with the frequency of instances with posterior values within each bin (i.e., binning on the [0,1] posterior of the binary base classifier.) Finally, the procedure counts in a 10×10 grid (Figure 4.2(b)), which we vectorise to 1×100. Concatenating these two embeddings we have a $E = 120$ dimensional representation of each feature dimension for processing by the meta network.

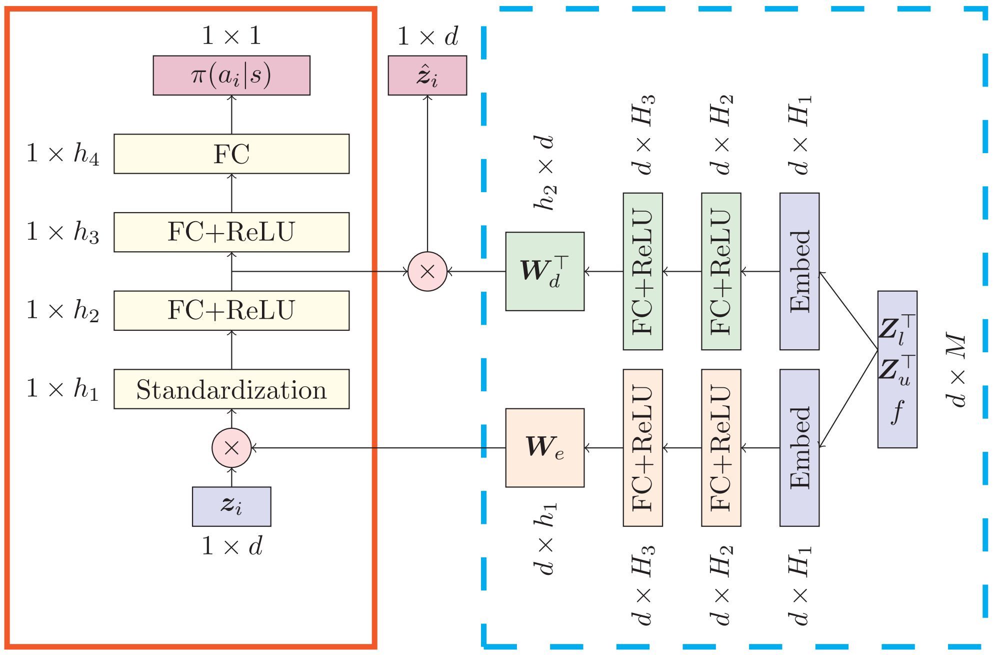
Algorithm 4 Reinforcement Learning of a Transferable Query Policy

```
1: Input: Initialised policy network and meta network
2: for each iteration do
3:     for each episode do
4:         Pick source dataset randomly
5:         Initialise label and unlabelled pool
6:         for each timestep do
7:             Sample action $\pi(a_i|s) \propto \exp^{\Phi_{\theta_q}(\mathbf{W}_e^T \mathbf{z}_i)}$
8:             Update the $\mathbf{Z}_u, \mathbf{Z}_l$ and base learner $f$
9:             Record the triplet $< \mathbf{Z}_u, a, r >$
10:        end for
11:        Standardise episode-collected return
12:    end for
13:    Update Policy with standardised return
14: end for
15: return Trained Active Query Policy
```

4.3.3 Meta-learned Policy for General Active Learning

Training for Cross Dataset Generalisation: We train policy networks and meta networks by using the policy gradient method REINFORCE [Williams, 1992] to ensure that the generated policies maximise the return (active learning accuracy) with the desired reward discounting. To ensure that our pair of networks achieve the desired dataset (active learning problem) invariance, we perform multi-task training on multiple source datasets: (i) In every mini batch we sample a random subset of source datasets, and set the return to the average return over all the sampled datasets. Thus achieving a high return means the meta network has learned to synthesise suitable per-dataset weights for the policy network based on the dataset embedding, and that together they generalise across multiple tasks/datasets. (ii) To further promote cross-dataset generalisation, we apply the baseline method to standardise the return from each episode which compensates diverse return scales across different datasets. This relative return alleviates the risk of domination by a single dataset with large return due to differing scale of accuracy increments among datasets of varying difficulties. The overall training algorithm is summarised in Algorithm 4.

The ideal active learner should query the instance that maximally improves the base learner's performance. The reward that reflects the quantity we care about is



(a) Policy Network (b) Meta Network

Figure 4.3: Policy and meta network architecture for deep reinforcement learning of a task-agnostic active query policy. Policy net inputs data-point z_i and outputs a probability of querying it $\pi(a_i|s)$. The policy network is paramaterised by weights $\boldsymbol{W}_e$ that dynamically determined by the meta network based on an embedding of the dataset and classifier $s_t = \{\mathcal{L}_t, \mathcal{U}_t, f\}$.

therefore the increase of test split accuracy $r_t = Acc_t - Acc_{t-1}$. To optimise this quantity non-myopically, we define the return of an active learning session as $J(\theta) = \mathbb{E}[\sum_{t=1}^{\infty} \gamma^{t-1} r_t(s, \pi_\theta(\cdot, s))]$. We then use policy gradient to train the policy and meta-networks to optimise the objective $J(\theta)$.

Auxiliary Regularisation Losses: Besides optimising the obtained reward, we also optimise for two auxiliary regularisation losses. **Reconstruction:** the policy network should reconstruct the unlabelled input data using $\boldsymbol{W}_d$ predicted by the meta-network [Romero et al., 2017]. We optimise $A(\boldsymbol{Z}_u) = |\boldsymbol{Z}_u - \hat{\boldsymbol{Z}}_u|_F$, the mean square reconstruction error of the autoencoder. **Entropy:** following [Mnih et al., 2016], we also prefer a policy that maintains a high-entropy posterior over actions so as to continue to explore and avoid pre-emptive convergence to an over-confident solution.

With these two auxiliary supervised tasks, we train both networks end-to-end. We minimise the whole objective function F by reversing the sign of policy gradient objective function:

$$F = -J_\theta(\Phi) + \lambda_1 A_{\theta_m^d}(\boldsymbol{Z}_u) - \lambda_2 \mathcal{H}(\pi_\theta(\boldsymbol{a}|\boldsymbol{Z}_u)) \tag{4.4}$$

where $\theta = \{\theta_q, \theta_m^e\}$. The network (Figure 4.3) trained by Equation 4.4 using Algorithm 4 learns to synthesise policies that are effective active query criteria (high return J) on any domain/dataset (synthesising domain specific network parameters via auxiliary network), adapting to the statistics of the dataset and independent of the dimensionality of the dataset.

4.4 Experiments

4.4.1 Datasets and Settings

Datasets: We experiment with a diverse set of 14 datasets from UCI machine learning repository. These include *austra*, *heart*, *german*, *ILPD*, *ionospheres*, *pima*, *wdbc*, *breast*, *diabetes*, *fertility*, *fourclass*, *habermann*, *livers*, *planning*. For our main experiment, we use the leave-one-out (LOO) setting: training on 13 datasets, and evaluating on the held out dataset.

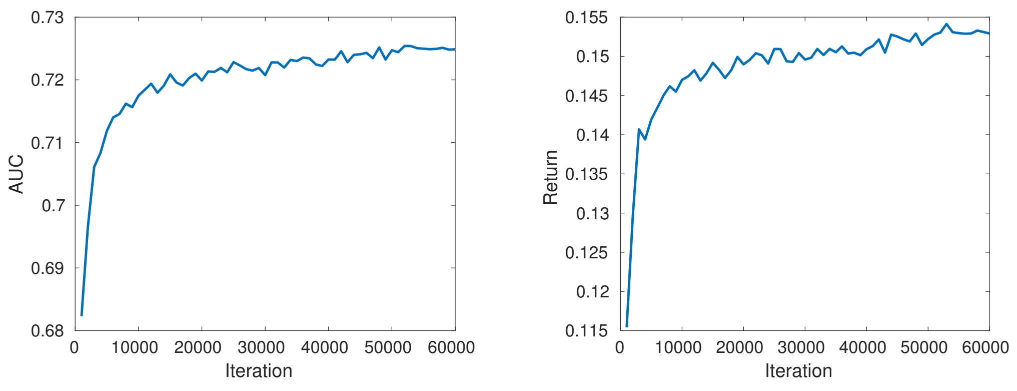
Architecture: The auxiliary network for encoder has fully connected layers with of size $120, 100, 100$ ($E = 120, k = 100$) and that for the decoder has an analogous structure. The policy network has layers of size $N \times d$ ($N \times d$ input matrix $\boldsymbol{Z}_u$), $N \times 100$, $N \times 50$, $N \times 10$, $N \times 1$ (N-way output). All penultimate layers use ReLU activation. The transition of the input to first hidden layer of policy network is provided by the auxiliary network. Thereafter for efficient implementation with few parameters and to deal with the variable sized input and output, the policy network is implemented convolutionally. We convolve a $h_1 \times h_2$ sized fillter across the N dimension of each $N \times h_1$ shaped layer to obtain the next $N \times h_2$ layer.

Experiment Settings: We train using Adam optimiser with initial learning rate 0.001 and hyperparameters $\lambda_1 = 0.03$, $\alpha = \lambda_2 = 0.005$ and discount factor $\gamma = 0.99$. During RL training, we use two tricks to stabilise the policy gradient. 1) We use a relatively large batch size of 32 episodes. 2) We smooth the gradient by accumulation $G_t = (1-\alpha)G_{t-1} + \alpha g_t$ where g_t is the gradient of the a_t in time step t and the G_t is the accumulated gradient. Intuitively, the accumulated gradient G_t relies on earlier time step actions. We train the policy and meta network simultaneously for a fixed 50,000 iterations and perform active learning over a time horizon (budget) of 20. As base learner we explore linear SVM and RBF SVM (kernel bandwidth 0.5) with class balancing. All results shown are averages over 100 trials of training and testing dataset splits.

Expert Features: To enhance the low-level feature of each instance in $\boldsymbol{X}$ we define expert features $\xi(\boldsymbol{X})$ to include distance furthest first (DFF) and uncertainty (US) as the augmented feature. In the SingleRL, we follow the similar setting of LAL Konyushkova et al. [2017a] which used expert features to summarise the relevant active learning information for each data point. The state of the learning process at time t are consist as follows: a) the proportion of negative class in labelled set $p(c=-1|\mathcal{L}_t)$; b) the proportion of positive class in labelled set $p(c=1|\mathcal{L}_t)$; c) the proportion of negative class in labelled set $p(c=-1|\mathcal{U}_t)$; d) the proportion of positive class in labelled set $p(c=1|\mathcal{U}_t)$; e) the budget ratio $\frac{t}{T}$.

Alternatives: We compare our learning approach to AL with three classic approaches uncertainty/margin-based sampling (US) [Tong and Koller, 2002; Kapoor et al., 2007], furthest-first sampling (DFF) [Baram et al., 2004] and query-by-bagging (QBB) [Abe and Mamitsuka, 1998], as well as to random sampling (RAND) as a lower bound. Uncertainty sampling is a simple deterministic approach that queries the instance with minimum certainty (maximum entropy). While simple, and not the most state-of-the-art criterion, it is consistently very competitive to more sophisticated criteria and more robust in the sense of hardly ever being a very poor criterion. We also compare with QUIRE [Huang et al., 2010] — a representative more sophisticated approach, and ALBL [Hsu and Lin, 2015] — a recent (within-dataset) learning based approach. We denote our method meta-learned policy for general active learning (MLP-GAL). As a related alternative we propose SingleRL. This is our RL approach, but without the meta-network, so a single model is learned over all datasets. This SingleRL is the incremental improved method with training the policy with non-myopic return rather than the myopic one-step classification loss. Without the meta-network it can only use expert features $\xi(\boldsymbol{X})$ so that dimensionality is fixed over datasets. To give SingleRL an advantage we concatenate some extra global features to the input space[1]. This method can also be seen as a version of one of the few state-of-the-art learning-based alternatives [Konyushkova et al., 2017b], with an important upgrade from the more myopic supervised learning used there to the reinforcement learning.

[1]Variance of classifier weights, proportion of labelled and (predicted) unlabelled positive or negative instances, proportion of budget used [Konyushkova et al., 2017b].



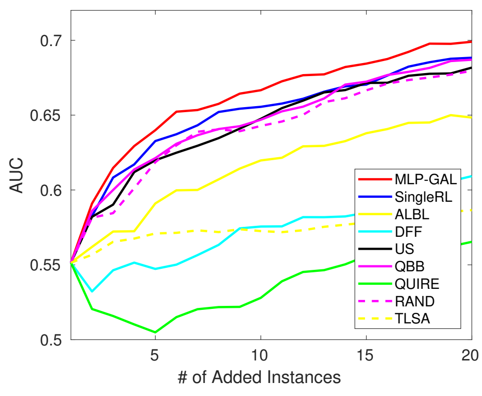
(a) AUC vs RL iterations (b) Return vs RL iterations

Figure 4.4: Convergence of active learning policy during training. Average over all training datasets (linear SVM).

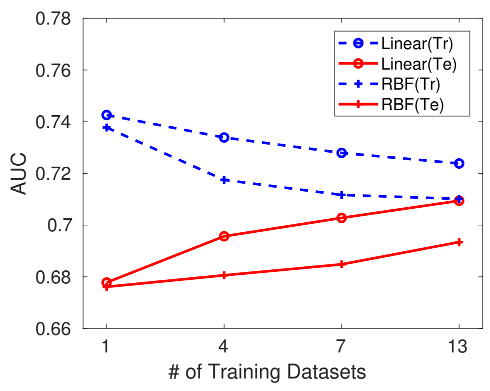
4.4.2 Results

Multi-Task Training Evaluation: We first verify that it is indeed possible to learn a single policy that generalises across multiple training datasets. In our leave-one-out setting, this means generalising across 13 datasets in a given training split. Figure 4.4 shows the convergence of multi-task RL training. The result is quantified in Table 4.3, where the MLP-GAL (Tr) column shows average classifier AUC across all the 13 combinations of LOO training. We can see that MLP-GAL learns an effective criterion that outperforms its competitors. There is potential for overfitting as the policy has seen each dataset during training (datasets randomly selected in minibatches). However it is interesting to see that it is possible to learn a *single* query policy that performs well on such a diverse set of datasets.

Cross-Task Generalisation: Next we evaluate whether the multi-task trained query policy can generalize to novel datasets. In the leave-one-out setting, each row in Table 4.3 (right section) represents a testing set, and the MLP-GAL (Te) result is the performance on this test set after training on all 13 other datasets. Our MLP-GAL outperforms alternatives in both average performance and number of wins. SingleRL is generally also effective compared to prior methods, showing the efficacy of training a policy with RL. However it does not benefit from a meta network, so is not as effective as our MLP-GAL. From the table it is also interesting to see that while sophisticated methods such as QUIRE sometimes perform very well, they also often perform very badly – even worse than random. Meanwhile the simple and classic uncertainty-sampling and QBB methods perform consistently well. Their robustness



(a) Illustrative active learning curves from evaluating our learned policy on held out UCI dataset diabetes.



(b) Cross-dataset generalisation. Average performance (AUC) of MLP-GAL over all training and testing sets as a function of the number of training domains.

Figure 4.5: Further analysis

is the reason for their continued use in practice despite their age and simplicity. This dichotomy illustrates the challenge in building sophisticated AL algorithms that generalise to datasets that they were not engineered on. In contrast, although our approach MLP-GAL (Te) has not seen these datasets during training, it performs consistently well due to adapting to each dataset via the meta-network. Figure 4.5(a) shows the resulting active learning curve for an example dataset.

Linear vs RBF SVM learner: An advantage of our approach compared to related methods such as [Bachman et al., 2017; Woodward and Finn, 2017] is that it treats the base learner as part of its environment to be optimised against rather than tying to a particular learner. With RBF SVM as the base learner, we can see that the results in Table 4.3 (top vs bottom) are similar to linear SVM (expected given the difficulty of learning a non-linear model in a budget of 20 points). Our approach is again superior overall — it is able to learn a policy customised to this new base learner.

Dependence on Number of Training Domains: We next investigate how performance depends on the number of training domains. We train MLP-GAL with an increasing number of source datasets – 1, 4, 7 (multiple splits each), and 13 (LOO split setting). Then we compute the average performance over all training and all testing domains, in all of their multiple occurrences across the splits. From the results in Figure 4.5(b) we see that the training performance becomes worse when doing a higher-way multi-task training. More specifically, according to Tables 4.4 and 4.5 (left

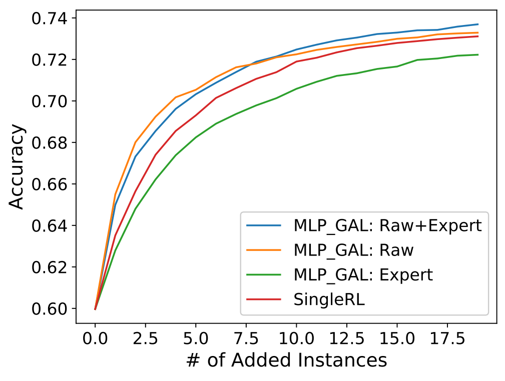
Item	Featurisation	Dims
State s_t	Representative dataset Embedding (1D histogram bin size:10)	10
	Discriminative dataset Embedding (2D histogram, bin size:10)	100
Action z_t	Raw features: $\boldsymbol{x}$	d
	Expert feature: US	1
	Expert feature: DFF	1

Table 4.1: The summary of the featurisation of MLP-GAL

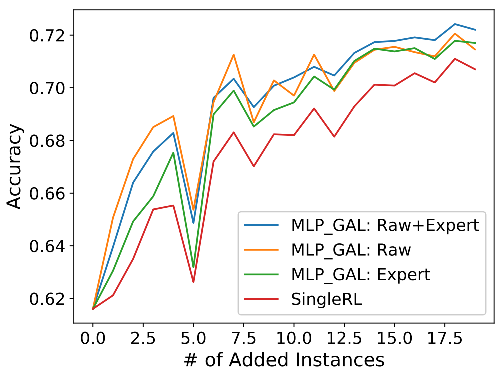
sections), only a few datasets show better training performance when jointly trained together with an increasing number of other training datasets – most show worse training performance. This is intuitive: it becomes harder to overfit a single model to more datasets simultaneously. The result is that in terms of *testing* performance (Tables 4.4 and 4.5, right sections), MLP-GAL trained on multiple datasets has better mean and lower variance for both linear SVM and RBF SVM. This is because when training on multiple datasets, the meta-network learns to tune the policy network in a dataset-specific way. It then successfully generalizes this dynamic policy synthesis to held out test datasets.

Ablation Study: We next investigate whether MLP-GAL learns extracting meaningful information for the active learning task from raw data without the reliance of the expert features. To do so, we apply ablation study on the MLP-GAL by empirically comparing the average accuracy over all time horizon of a variety of input featurisation in both linear and RBF settings on 13 UCI datasets. As illustrated on the Table 4.1, we compare the MLP-GAL with 3 different types of action featurisation: using raw and expert features (RAW & Expert), using only raw features (RAW), and only using expert features (Expert). The comparison also includes the SingleRL, where SingleRL act as the baseline for general learning active learning methods.

According to the Table 4.2 and Figure 4.2, the result shows that the MLP-GAL with or without using the expert features have almost equal performance and outperform the rest of competitors in both linear and RBF SVM settings. Comparing between these two types of featurisation, MLP-GAL without expert features are relatively worse than adding the additional information. Moreover, the result demonstrates that MLP-GAL with only using the expert feature has the worst performance. These demonstrations indicates that the expert features has fewer contribution than the raw features and the learned active learning policy are heavily relied on the raw features rather than the



(a) Linear SVM



(b) RBF SVM

Figure 4.6: The performance of MLP-GAL with/without using expert features

expert features. Since both of uncertainty and representatives of expert features are relatively useless to the learned policy, MLP-GAL learned a different motivation for the active learning task.

	MLP-GAL: Raw & Expert	MLP-GAL:Raw	MLP-GAL:Expert	SingleRL
linear SVM	70.94	70.93	69.16	70.19
RBF SVM	69.34	69.32	68.62	67.64

Table 4.2: The comparison of MLP-GAL with various action featurisation and the SingleRL.

4.5 Summary

We have proposed a learning-based perspective on the problem of active query criteria design. Such learning-based algorithm design elegantly obtains AL models by optimising the ultimate goal of classification performance with few labels. However aside from the widely-shared questions of good network architecture and RL training algorithms, the key challenge is whether general enough policies can be learned for being widely useful and applicable, rather than requiring dataset-specific training which contradicts the motivation of AL. Our key contribution is to provide the first solution to this issue through multi-task training of a meta-network that synthesises dataset-specific active query policies.

Our study thus far has the main limitation that we have only evaluated our method on a binary base classifier (an assumption shared by [Konyushkova et al., 2017b]).

In the future we would like to evaluate our method on deep multi-class classifiers by designing embeddings which can represent the state of such learners, as well as explore application to the stream-based AL setting.

Table 4.3: Comparison of active learning algorithms, leave one dataset out setting. Linear/RBF SVM base learner. AUC averages (%) over 100 trials (and 13 training occurrences for MLP-GAL (Tr)). Winning AL algorithm is bolded in each row.

Linear SVM	MLP-GAL (Tr)	MLP-GAL (Te)	SingleRL (Te)	Entropy	DFF	RAND	ALBL	T-LSA	QUIRE	QBB	DEAL
austra	80.14	78.09	75.72	78.24	75.63	75.87	75.31	72.98	64.46	**78.58**	76.21
breast	96.67	95.95	94.78	95.41	95.76	94.71	95.67	**96.21**	95.60	95.73	94.86
diabetes	67.53	**65.99**	64.78	64.18	57.31	64.05	61.35	57.34	53.75	64.46	64.26
fertility	78.26	75.09	**77.86**	75.79	70.44	71.28	66.92	71.18	54.93	73.87	68.25
fourclass	74.79	**74.11**	71.83	69.55	71.26	69.08	68.69	69.98	64.48	70.81	69.56
haberman	67.31	**65.61**	64.91	60.16	60.26	57.40	52.49	59.67	45.89	60.58	58.31
heart	76.68	72.77	72.84	73.38	**73.99**	73.06	71.78	71.52	67.07	73.36	72.53
german	68.01	**64.68**	63.35	63.34	61.78	62.77	61.74	58.75	51.82	64.16	61.86
ILPD	62.48	59.30	**61.08**	57.60	50.97	57.62	52.91	53.15	48.57	56.77	57.69
ionospheres	74.96	**71.46**	69.78	70.47	59.64	69.81	68.44	58.95	57.84	70.40	69.73
liver	55.66	55.51	**55.62**	53.45	52.87	52.87	51.25	51.36	48.11	52.13	53.39
pima	67.64	**67.01**	64.67	64.18	57.31	63.69	61.27	57.03	53.75	64.24	62.93
planning	60.74	**58.63**	56.75	55.09	52.77	54.17	49.46	52.04	39.90	55.43	52.95
wdbc	90.90	90.09	88.72	**90.93**	87.55	88.52	88.41	85.15	82.17	90.68	88.59
Avg	72.98	**70.94**	70.19	69.41	66.25	68.21	66.12	65.38	59.17	69.37	67.94
Num Wins	-	**7**	3	1	1	0	0	1	0	1	0
Win Ratio	-	**50.00**	21.42	7.14	7.14	0.00	0.00	7.14	0.00	7.14	0.00

RBF SVM	MLP-GAL (Tr)	MLP-GAL (Te)	SingleRL (Te)	Ent	DFF	RAND	ALBL	T-LSA	QUIRE	QBB	DEAL
austra	80.84	79.07	76.35	**79.36**	77.15	78.47	76.57	72.32	68.98	78.83	78.23
breast	96.25	**95.96**	95.46	95.40	95.78	95.14	95.92	93.28	95.21	95.43	95.27
diabetes	66.55	**64.89**	62.52	62.59	59.81	62.70	59.09	59.38	58.48	61.98	62.07
fertility	80.83	78.97	75.75	**79.49**	75.81	75.21	73.55	72.58	64.67	76.83	72.99
fourclass	71.66	67.21	66.41	66.88	**68.62**	66.29	66.43	61.80	64.85	63.35	65.32
haberman	58.01	58.36	53.88	56.60	58.67	53.58	64.44	62.66	61.83	**64.97**	52.80
heart	77.47	**74.79**	71.87	73.63	74.05	72.27	72.57	69.70	68.98	72.95	72.25
german	67.94	**66.99**	64.18	65.01	65.60	63.26	57.70	51.49	55.57	53.96	63.84
ILPD	54.50	50.30	51.04	50.99	47.29	52.3	47.62	47.92	46.54	51.15	**52.77**
ionospheres	80.94	76.39	72.87	**77.76**	61.49	75.17	75.00	64.23	61.72	77.18	75.07
liver	51.91	**50.91**	50.76	50.31	51.04	50.21	47.60	48.25	46.75	50.27	50.45
pima	66.60	**65.58**	63.15	62.59	59.81	63.01	58.13	58.99	58.48	61.74	61.92
planning	53.05	51.46	**52.61**	49.95	50.07	50.99	47.10	49.86	41.68	50.49	50.68
wdbc	91.97	89.84	90.04	**91.54**	89.37	90.24	89.52	87.76	88.14	90.34	90.64
Avg	71.32	**69.34**	67.64	68.72	66.75	67.77	66.52	64.30	62.99	67.82	67.54
Num Wins	-	**6**	1	4	1	0	0	0	0	1	1
Win Ratio	-	**42.86**	7.14	28.57	7.14	0.00	0.00	0.00	0.00	7.14	7.14

Table 4.4: MLP-GAL training and testing performance as a function of number of training datasets. AUC average and standard deviation. Linear SVM base classifier. Each dataset is evaluated both as train and test during cross validation.

	Train Performance				**Test Performance**			
Num Train Sets:	1	4	7	13	1	4	7	13
austra	**81.49**	80.81 ± 0.57	80.56 ± 0.60	80.14 ± 0.65	72.78 ± 2.99	72.90 ± 1.97	74.07 ± 3.11	**78.09**
breast	**96.94**	96.85 ± 0.12	96.76 ± 0.16	96.67 ± 0.13	94.55 ± 1.00	95.36 ± 0.66	95.31 ± 0.36	**95.95**
diabetes	**69.23**	67.25 ± 0.29	67.22 ± 0.48	67.53 ± 0.45	63.05 ± 2.63	65.03 ± 1.91	65.56 ± 1.16	**65.99**
fertility	**79.90**	79.20 ± 0.30	78.38 ± 0.36	78.26 ± 0.61	72.90 ± 1.91	73.86 ± 1.44	74.91 ± 2.37	**75.09**
fourclass	**76.03**	75.36 ± 0.48	75.08 ± 0.37	74.79 ± 0.47	69.15 ± 2.15	71.24 ± 2.69	73.02 ± 0.78	**74.11**
haberman	**71.33**	68.06 ± 0.74	66.91 ± 0.84	67.31 ± 0.62	59.28 ± 3.57	62.00 ± 2.52	64.97 ± 0.54	**65.61**
heart	**80.30**	78.46 ± 1.19	77.48 ± 0.56	76.68 ± 0.74	70.38 ± 2.84	72.50 ± 1.79	72.35 ± 0.92	**72.77**
german	68.55	**68.60** ± 0.36	68.10 ± 0.16	68.01 ± 0.33	64.05 ± 2.04	64.44 ± 0.99	**65.00** ± 1.67	64.68
ILPD	**65.26**	64.01 ± 1.04	62.97 ± 0.82	62.48 ± 1.07	56.17 ± 2.73	58.37 ± 1.50	58.11 ± 1.56	**59.30**
ionospheres	75.29	**75.80** ± 1.68	75.21 ± 1.06	74.96 ± 0.78	68.04 ± 3.85	70.27 ± 1.95	70.12 ± 1.57	**71.46**
liver	54.88	**56.59** ± 0.41	56.04 ± 0.35	55.66 ± 0.34	54.37 ± 1.07	54.82 ± 0.46	54.86 ± 0.27	**55.51**
pima	**69.77**	67.83 ± 0.31	66.78 ± 0.65	67.64 ± 0.60	63.00 ± 2.59	65.15 ± 1.76	66.20 ± 1.28	**67.01**
planning	**62.61**	61.37 ± 0.51	60.71 ± 0.79	60.74 ± 0.98	54.90 ± 3.14	57.28 ± 3.07	57.23 ± 1.31	**58.63**
wdbc	**91.40**	91.25 ± 0.19	90.78 ± 0.58	90.90 ± 0.25	86.60 ± 2.49	88.76 ± 0.92	89.16 ± 0.91	**90.09**
Average	**74.50**	73.67 ± 0.59	73.07 ± 0.55	72.98 ± 0.57	67.80 ± 2.50	69.43 ± 1.69	70.06 ± 1.27	**70.94**
Num Wins	**11**	3	0	0	0	0	1	**13**

Table 4.5: MLP-GAL training and testing performance as a function of number of training datasets. AUC average and standard deviation. RBF SVM base classifier. Each dataset is evaluated both as train and test during cross validation.

	Train Performance				**Test Performance**			
Num Train Sets:	1	4	7	13	1	4	7	13
austra	**81.84**	81.73 ± 0.54	80.99 ± 0.31	80.84 ± 0.34	76.47 ± 2.01	76.01 ± 0.82	76.50 ± 2.71	**79.07**
breast	95.94	96.08 ± 0.17	96.27 ± 0.19	**96.25** ± 0.24	94.74 ± 0.89	95.52 ± 0.32	95.68 ± 0.27	**95.96**
diabetes	**70.37**	66.96 ± 1.26	66.86 ± 0.96	66.55 ± 1.05	63.57 ± 2.98	63.80 ± 3.13	64.67 ± 2.20	**64.89**
fertility	81.70	**81.71** ± 0.66	81.44 ± 0.89	80.83 ± 0.58	75.91 ± 2.80	76.74 ± 1.88	77.92 ± 0.61	**78.97**
fourclass	**73.90**	71.49 ± 0.65	70.73 ± 0.66	71.66 ± 0.66	66.28 ± 1.61	67.11 ± 1.68	**67.81** ± 1.32	67.21
haberman	**65.95**	61.95 ± 1.79	60.41 ± 1.77	58.01 ± 1.56	54.97 ± 2.24	56.37 ± 2.26	54.87 ± 1.33	**58.36**
heart	**79.70**	79.09 ± 0.93	77.90 ± 0.82	77.47 ± 0.68	72.26 ± 2.09	73.76 ± 1.85	73.58 ± 1.56	**74.79**
german	**70.23**	68.73 ± 0.66	68.10 ± 1.34	67.94 ± 0.36	64.63 ± 3.91	64.14 ± 2.29	65.27 ± 1.35	**66.99**
ILPD	**62.10**	57.89 ± 2.87	55.10 ± 0.89	54.50 ± 0.55	**52.45** ± 2.82	50.94 ± 2.20	50.54 ± 1.81	50.30
ionospheres	80.81	**81.50** ± 0.36	81.67 ± 0.46	80.94 ± 0.55	73.54 ± 3.49	74.97 ± 2.40	**76.87** ± 3.04	76.39
liver	**56.01**	51.81 ± 1.18	52.26 ± 0.72	51.91 ± 0.85	50.79 ± 0.79	**50.96** ± 1.06	**50.96** ± 0.57	50.91
pima	**71.56**	67.73 ± 0.83	66.57 ± 1.70	66.60 ± 1.24	63.36 ± 2.82	63.59 ± 2.32	65.31 ± 1.63	**65.58**
planning	**57.87**	54.20 ± 2.15	53.38 ± 0.58	53.05 ± 0.89	51.48 ± 1.26	52.02 ± 1.07	**52.13** ± 0.63	51.46
wdbc	**92.92**	92.20 ± 0.30	92.31 ± 0.70	91.97 ± 0.28	88.87 ± 1.17	88.86 ± 0.97	**90.31** ± 0.53	89.84
Average	**74.35**	72.36 ± 1.03	71.71 ± 0.86	71.32 ± 0.70	67.81 ± 2.21	68.20 ± 1.73	68.74 ± 1.40	**69.34**
Num Wins	**11**	2	0	1	1	1	5	**8**

Chapter 5

Learning to Fine-tune

5.1 Introduction

Transfer learning is the process of extracting knowledge from a well-learned source task to accelerate or improve the learning of a target task, typically one that has less data than the source, and would therefore be difficult to learn from scratch. This is particularly important in deep-learning context. Since the basic data requirements for deep learning methods are tremendous, many problems do not possess an adequate amount of data to be learned from scratch; since the learning process is slow, any efficiency benefits are welcome. By far the most common approach to transfer learning in deep learning is the pre-train/fine-tune pipeline where a model is trained from scratch for a data-rich source task, and the weights are copied and used as the initial condition for the gradient-descent-based learning of a data-sparse target task.

Although fine-tuning-based transfer learning improves significantly over tabula rasa learning, it is limited because it only fine-tunes the weights of the target problem, keeping the structure intact. This property is particularly salient in a fine-tuning context, because the source problem is often a very general problem compared to a more specific target problem; for example, in the common case of fine-tuning an ImageNet pre-trained model for more specific tasks, such as person re-identification [Fan et al., 2018], face recognition [Chowdhury et al., 2016], or bird recognition [Cui et al., 2018]. In this case, only a subset of the knowledge encoded in the source model is likely to be relevant to the target. This means that the process is slower than necessary (due to the intensive computation from the large capacity model) and also produces relatively inaccurate final models (due to overfitting from these additional parameters), compared to an alternative ideal case in which we know which of the source-task parameters are relevant to the target task. In this ideal case, we could learn faster and generalise better due to working with fewer parameters.

Estimating the relevance of our source knowledge to focus on transferring relevant information is non-trivial if we bar the prohibitively costly approach of trying all options of what to transfer and retraining the target model for every option. If we consider selecting the knowledge to transfer at the neuron-level, the number of options for what to transfer is exponential, due to the number of source neurons combination. We address this issue by proposing a fine-tuning algorithm that transforms this subset selection problem into a sequential decision-making problem. Our algorithm iteratively prunes several neurons while fine-tuning the DNN on target problem.

There are two key challenges to realise our vision. The first challenge concerns how to train such a learner, and the second involves ensuring that it generalises to a novel problem. (1) We trained our meta learner (i.e. the structure pruning and weight fine-tuning algorithm) with reinforcement learning, because we framed the problem of selecting knowledge for transfer as a sequential issue, and also because the objective function (performance of the target model after its structure is updated and the weights refined by an updated step of the learner) is not straightforwardly differentiable with respect to the weights of the learner. (2) Although we can train a structure and weight fine-tuning algorithm with RL, this requires a prohibitive amount of computational costs for any given problem. Therefore, similarly to the case in Chapter 4, we need to be able to train it on a set of source problems and apply it to a novel target problem, so that the policy-training cost can be amortised. To this end, we performed multi-task training on a batch of source problems to learn a task-agnostic fine-tuning policy.

Overall, we contribute a process for learning a fine-tuning algorithm that both refines the weights and prunes the structure during transfer learning, thus improving both the efficiency and the accuracy of transferring learning to a target problem. Moreover, it produces a more compact target model. We show that this approach improves vanilla transfer learning, as well as related alternatives such as dropout and random pruning strategies.

5.2 Related Work

In this section, we review the studies on network architecture search (NAS) of the optimal structure. Then, we discuss model compression, which may be regarded as a special case of NAS that searches a compact structure from a good initial condition.

Finally, we discuss the works on meta learning of supervised learning and transfer learning.

5.2.1 Network Architecture Search

The majority of deep learning techniques require human effort on manually architecture exploration to achieve the desired performance [Lecun et al., 1998; Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Szegedy et al., 2015; He et al., 2016; Huang et al., 2017]. Motivated by this, the research field of NAS learns and explores the architecture for given datasets. The difficulties with NAS include the large search space for finding the optimal architecture and the intensive computational effort required. Here, we discuss the research on NAS with meta-learning, with regard to the improvement of effectiveness and efficiency.

However, NAS could be improved by applying either evolutionary algorithms (EA) or reinforcement learning [Zoph and Le, 2017; Real et al., 2017; Baker et al., 2017]. The EA treats NAS as a black-box function optimisation without any assumptions, whereby the search space is the space of architectures, and the function value is the validation accuracy of a model trained using a given architecture. The EA approaches perform an approximated global search by maintaining a population of diverse architectures and selecting for fitness validation accuracy. Moreover, RL-based NAS provides improved search efficiency and assumes that defining the architecture is a sequential decision. However, the training approach of RL still requires intensive computation for exploring the ideal architecture. The efficiency of NAS could be upgraded by approximating the solution with either supervised learning or learning a transferable architecture. A recent NAS approach involves learning architecture iteratively by back-propagating the gradient [Liu et al., 2018]. It relaxes the search space of the layer operation to be continuous with softmax in training time and uses the max function to select the best function in test time. This enables the training to be efficient, since the architecture can be searched through the back-propagation without needing to conduct an expensive exploration during the supervised learning process. Another efficient NAS approach is to learn a transferable architecture with RL, where the transferability relies on a predefined search space [Zoph et al., 2017]. Then, RL learns to combine basic operations in a specific search space, reduces the required computation time, and learns a transferable architecture. Since transferability depends on the particular architecture, this property may fail to apply to a newly defined architecture.

Summary: All of these NAS studies reveal the effectiveness of searching for the optimal architecture for a given problem. Although several studies have attempted to reduce training time, the training of NAS is still generally expensive. In addition, other NAS studies have attempted to improve the transfer learning process. These gaps in the literature motivate our learning to fine-tune the research, which enables the network architecture to be pruned for a specific transfer learning task.

5.2.2 Model Compression

We next review model compression (MC), which is related to NAS. However, there are two main differences. Typically, NAS begins by searching a random architecture with the aim of finding the best-performing architecture. However, MC starts searching from a pre-defined, well-performing architecture, and tries to find a smaller architecture that maintains a similar level of performance to that of the input model architecture. We briefly review the typical model compression research and discuss how model compression could be improved through meta learning.

Conventional model compression has been proposed to reduce the size of the deep network based on various motivations, such as parameter pruning, low-rank approximation, and knowledge distillation. Parameter pruning techniques remove redundant and non-informative weights in a pre-trained convolutional model using either magnitude of weight or the Hessian of the loss function [Cun et al., 1990; Hassibi and Stork, 1993]. Later, parameter pruning methods were upgraded by combining the technique of binning network parameters with a Hash function [Han et al., 2015]. Another angle of model compression is to apply low-rank approximation to factorise the weights of deep networks [Jaderberg et al., 2014; Tai et al., 2015; Yu et al., 2017]. This method approximates the original weight tensors and matrices by minimising their reconstruction error with respect to a low-rank approximation with fewer parameters. Generally, both parameter pruning and low-rank approximation will narrow down the size of the network, while the depth might remain the same or even become deeper.

To address this, distillation techniques train a new model with a smaller architecture that achieves a similar performance to the original model [Ba and Caruana, 2014; Korattikara Balan et al., 2015; Luo et al., 2016]. The designed model not only has a compact representation for the original problem, but also has a similar performance of the pre-trained model. As a result, the model compression problem will become flexible on the choice of the architecture. Although the design of the smaller target architecture is an important factor, it is still typically left to human experts. Recent model compression research has proposed using RL to find the best new compressed architecture given a pre-trained large architecture [Ashok et al., 2017; He et al., 2018]. Here, knowledge distillation is formalised as a sequential decision-making process, wherein policies iterate with multiple knowledge distillation processes to control layer sparsity and removal. It enables finding the best compressed architecture without conducting an exhaustive architecture search or manually designing the target architecture.

Summary: All of these methods focus on a single task. They assume that the base network has already achieved decent performance on this task. However, it is not clear how to combine them with task-transfer in the form of fine-tuning, as there is an issue of compressing before or after fine-tuning process.

5.2.3 Meta Learning for Supervised Learning

We now introduce the SGD-based meta learning, which has been widely studied for supervised learning. Unlike the conventional SGD-based learning algorithms that rely on human effort to tune the hyper-parameters and design, meta learning methods improve these by learning the best optimisation strategy [Li and Malik, 2017; Andrychowicz et al., 2016; Ravi and Larochelle, 2017] or learning the best initial condition to optimise model-agnostic meta-learning (MAML) [Finn et al., 2017]. One of the most popular learning approaches is MAML, due to its simplicity. For this reason, it has been extended in various ways, including probabilistic generalisations wherein the MAML is the special case of the probabilistic approach with point estimation [Finn et al., 2017; Grant et al., 2018]. Another extension of MAML is learning a subspace and mask that describe which weights to update [Lee and Choi, 2018].

5.2.4 Meta Learning for Transfer Learning

Several studies have applied meta-learning to improve transfer learning for cross-task transfer and cross-domain transfer [Li et al., 2018; WEI et al., 2018]. In [WEI et al., 2018], multiple cross-task transfer learning experiences are gathered and used as training data, and the meta-learning procedure involves learning to estimate the performance improvement ratio when given triplets (source domain, target domain, transfer parameters). This knowledge is then exploited for a new transfer learning task at test time, at which point the estimated performance improvement ratio is maximised with respect to new target parameters. However, this approach is limited in the sense that it is for shallow models, and thus does not benefit deep-network transfer learning. In [Li et al., 2018; Balaji et al., 2018] multiple cross-domain transfer learning experiences are generated, and the meta-learning procedure tries to update the source domain model so that it is more robust to cross-domain transfer. However, none of these methods refine the models architecture to benefit the target problem. Conversely, we aim to meta-learn a transferrable non-myopic architecture tuning rule for transfer learning. This will allow both parameters and weights of the target problems network to be fine-tuned.

Summary: None of the previously proposed studies have considered the effect of the pruning network's architecture during the fine-tuning-based transfer learning phase. Here, our presented method meta-learns a transferable non-myopic architecture tuning rule for fine-tuning with reinforcement learning. As a result, this rule not only to tunes the parameters of a network but also the network architecture.

5.3 Learning a Neuron Deletion Policy

In this work, we present a novel learning transfer learning framework that enables a transfer learning process to learn the architecture and weights for the target problems. Since the practical use scenario for TL is to transfer from a general model (such as image-net trained) to a specific model (such as birds), our architecture fine-tuning approach focuses on neuron deletion.

For the purpose of learning architecture dynamically from a general model, we propose to remove useless neurons for the specific target problem during fine-tuning by evaluating the usefulness of the neurons. This requires the parameter refinement process to interact with the neuron deletion process, so as to suggest neurons to delete before updating the model's parameters. It is natural to represent the interactive learning process as a sequential decision-making problem, since each deleted neuron affects both the next available neurons and the state of the neural network successively for the next decision. In this way, the neuron deletion policy, trained by reinforcement learning, is able to learn a powerful and non-myopic policy.

Definition of Deletion Policy: We propose to model the neuron deletion policy as a neural network and formalise the process of training this policy as a deep reinforcement learning problem. We use s_t as the state of the base learner f_t to summarise the base learner information in training. And we further denote the deletion policy as $\pi(a_{i,t}|s_t)$, where action $a_{i,t}$ permanently deletes the corresponding neuron $i \in O_t$, and O_t is the candidate set of all available deleted neurons in the deep network at the tuning step t. Here, we regard each entire feature map in the convolution layer as a low-resolution neuron to simplify the process. Based on the deleted neuron, the state s_{t+1} is updated successively to fine-tune both the architecture and the weight of the base learner f_{t+1}.

We optimise the deletion policy to reduce the required training time and improve the overall performance by using the accuracy of the base learner f_{t+1} as a reward. Let $J(\theta) = \sum_{t=1}^{T} \gamma^{t-1} Acc_t$ denote the objective function to optimise where the Acc_t is the base learner accuracy at time step t and T is the total time horizon. Maximising the accumulated accuracy is equivalent to optimising the area under the accuracy curve, which optimises the learning speed of the architecture fine-tuning process. Additionally, since the final accuracy Acc_T has a larger numerical scale and repeated occurrence than the early time step accuracy Acc_1 in the objective function, the policy also learns how to perform better in the end.

Single Neuron Deletion Policy: Similar to the transferable active learning policy, the deletion policy is coordinate-wise on the neurons of the base learner. This allows the deletion policy to overcome the variable size of the neurons over various time steps and architectures. We assume the policy models the actions via the Softmax function $\pi_{\theta_d}(a_{i,t}|s_t) \propto \exp^{\Phi_{\theta_d}\left(\phi(s_t,o_{i,t})\right)}$, where the featurisation $\phi(s_t, o_{i,t})$ summarises information regarding the observed forward and backward values of a neuron $o_{i,t}$ and the current base learner state. The policy samples a neuron to delete at each reinforcement learning time step from the Softmax distribution of all available neurons. Here, each action $a_{i,t}$ corresponds to a particular neuron $o_{i,t}$ at time step t.

Batch Neurons Deletion Policy: However, deleting a single neuron at a time is not an ideal choice for architecture tuning. Indeed, it is impractical for a large capacity network to delete a neuron at each update iteration when the target problem has an extremely small number of samples. Since the pace of architecture tuning is much slower than the speed of converging to the local minima, the model can completely memorise the target dataset. In addition, the optimal deleting action might consist of a combination of useless neurons rather than only one neuron. Thus, the question shifts from ”which is the most useless neuron” to ”what is the most useless combination of neurons”, which we denote as a *batch action* policy. This combination of useless neurons is an N-choose-K combination problem, where the optimal number of K depends on various factors in the process, including: target sample size, update iterations, and network capacity. In this work, we simply chose a constant K for all different domains

Algorithm 5 Per Time Step Batch Deletion Policy

```
1: Input: Given a deletion network $\Phi_{\theta_d}$, the model $f_t$, initialised index set $I$, candidate set $O_t^k$, and the featurisation for all neurons $\phi(s_t, \boldsymbol{o}_t)$
2: for $k \leq K$ do
3:     Sample action $\pi(a_{I_k,t}|s) \propto \exp^{\Phi_{\theta_d}}\left(\phi(s_t, o_{i,t})\right)$
4:     Append the deleted action $a_{I_k,t}$ into $\tilde{a}_t$
5:     Append the deleted neuron index into the index set $I_k$
6:     Remove the deleted neuron from candidate set $O_k$
7: end for
8: return A batch neurons $\tilde{a}_t = \{a_{I_1,t}, a_{I_2,t}, \ldots, a_{I_K,t}\}$ to be deleted
```

and architectures.

For this batch action policy, the action space is changed from $|\mathcal{A}| = N$ to $|\mathcal{A}| = C_K^N$, for which finding the optimal combination from this finite set is known as a combinatorial optimisation problem. Next, we explore and exploit a solution to the combinatorial optimisation problem using reinforcement learning. The policy models the batch-action decision as an iterative sampling process that samples K times from a softmaxed distribution without replacement. This could be regarded as meta-learning an iterative sampler for approximating a solution to the combinational optimisation problem as a non-myopic sequence of decisions. The batch action now represents a set of neurons to delete as $\tilde{a}_t = \{a_{I_1,t}, a_{I_2,t}, \ldots, a_{I_K,t}\}$ where I is a set of deleted neurons index. More specifically, we iterate the previous deletion procedure to update both the candidate set O_t^k and the deleted index set I_k K times. Subsequently, we will have the probability for the batch action as follows:

$$\pi_{\theta_d}(\tilde{a}_t|s_t) \propto \prod_{k=1}^{K} \frac{\exp^{\Phi_{\theta_d}}\left(\phi(s_t, o_{i,t}^k)\right)}{\sum_{j \setminus I_k}^{|O_t^k|} \exp^{\Phi_{\theta_d}}\left(\phi(s_t, o_{j,t}^k)\right)} \tag{5.1}$$

While the method explained here can learn to delete redundant neurons for the fine-tuned target problem, it is impractical to have a neuron deletion policy. A non-terminating deletion policy will continually prune the network until all neurons are deleted. This leads to an extremely low capacity network with weak representation and generalisation performance. Thus, in the next section, we introduce a complementary policy to determine when to stop deleting neurons.

5.4 Learning Architecture Stopping Tuning Policy

In this section, we describe the necessity of introducing a stopping policy to learn the fine-tune task. Next, we illustrate how to formalise the task and train the stopping policy in a more data-efficient manner. The stopping policy attempts to estimate the optimal stopping point for the deletion policy; this allows for a trade-off between the benefits of simplifying the target model and the drawbacks of the eventual lack of representation power. We present an approach to meta-learning a stopping policy that determines whether to stop or continue to tune (prune) the architecture at each time step.

We propose to meta-learn a high-level stopping network to estimate a termination condition for the neuron deletion policy. The stopping policy is denoted as $\pi_{\theta_s}(a_{s_t}|s_t)$, where action a_{s_t} is a binary value for stopping or continuing. At each tuning step, the stopping policy decides whether to apply the deletion policy or fine-tune the network directly. The training of this process can be formalised from two different angles: supervised learning and reinforcement learning. Generally, it is more elegant to optimise the stopping policy with reinforcement learning. However, this is the more expensive option. In this thesis, we simplify the training of a stopping network as a supervised learning problem for computational efficiency. For supervised learning, the collected samples (network features at different time-steps) need to be annotated in regard to whether a stop or a continuation is the optimal action. To obtain this annotation, we define a greedy heuristic oracle that determines a stop/continue label based on the recent validation accuracy change, as per conventional early-stopping. The stopping policy can thus be viewed as a one-step heuristic optimiser for the neuron deletion policy.

5.5 Domain and Architecture Invariant Featurisation

Based on the learning of transferable active learning policy proposed in Chapter 3, to amortise the cost of training our fine-tuning policy, we need it to be transferrable across problems and architectures. Thus, we need a state featurisation that is domain and architecture invariant and summarises both neuron and learner information.

Learner Featurisation: We first depict the featurisationfor the learner information. The state s summarises the learner information, which contains: (i) the current compression rate of the neurons $\frac{\text{Current Nums of Neurons}}{\text{Initial Nums of Neurons}}$; (ii) the current compression rate of the weight $\frac{\text{Current Nums of Params}}{\text{Initial Nums of Params}}$; (iii) the current training accuracy Acc_t; (iv) the accuracy increments from the last update $Acc_t - Acc_{t-1}$; (v) the current used budget of the fine-tuning iterations $\frac{t}{T}$.

Neuron Featurisation: We next describe the featurisation for the neuron information. This featurisation is supposed to depict either inter-neurons the or intra-neuron perspective. The intra-neuron information is summarised as: (i) mean, variance, and histogram of forward and backward value for each individual neuron, where the histogram divides the corresponding features into 10 bins. Note that the forward and backward values are the intermediate result of each neuron feeding batch data to the DNNs. Here, the backward value is the gradient. (ii) Rescaled bias value and its gradient. The inter-neurons information is summarised as: (i) histogram of the cosine similarity between the forward and backward values, (ii) histogram of cosine similarity between the weight values, (iii) histogram of cosine similarity between the weight gradients.

5.6 Cross-Datasets Training

We now combine these two networks together and formalise the training of architecture tuning. The deletion policy is trained with vanilla REINFORCE to learn a non-myopic way to maximise the return (accumulated accuracy). The stopping policy is trained by minimising the cross-entropy loss function to estimate the test performance increase from the given learner state.

Baseline Method for Deletion Policy: Similar to the work of learning transferable active learning, we also expect the universal deletion and stopping policy to be domain agnostic and transferable for different domains. We perform multi-task training on multiple source datasets: (i) In every reinforcement learning training iteration, we sample multiple batches of trajectories, where each batch uses the same source dataset. Next, we set the return as the discounted accumulated reward. Achieving a high return means that the deletion policy learns to delete the most useless neurons for the fine-tuning problem. (ii) We apply an intra-batch baseline method to rescale each time step return to $[-1,1]$ by $2 \times \left(\frac{x - \boldsymbol{x}_{\min}}{\boldsymbol{x}_{\max} - \boldsymbol{x}_{\min}}\right) - 1$. This intra-batch baseline encourages the policy to learn from the long-term "good" or "bad" action, which enables the policy to discover a good approximation for the combinational optimisation problem. In addition, the relative return again avoids the risk of domination by a single dataset with

Item	**Featurisation**	**Dims**
State s_t Dims: 5	Neuron Compression Rate	1
	Weight Compression Rate	1
	Training Accuracy for Last Training Batch	1
	Training Accuracy Increment from Last Update	1
	Budget of Fine-tuning Iteration	1
Neuron $o_{i,t}$ Dims: 86	Mean of forward value	1
	Variance of forward value	1
	Histogram of forward value	10
	Bias value	1
	Mean of backward value	1
	Variance of backward value	1
	Histogram of backward value	10
	Bias gradient	1
	Histogram of cosine similarity for the histogram of forward value	10
	Histogram of cosine similarity for the histogram of backward value	10
	Histogram of cosine similarity for the histogram of the weight of linking the neuron o_i from last layer's neurons	10
	Histogram of cosine similarity for the histogram of the weight of linking the neuron o_i to next layer's neurons	10
	Histogram of cosine similarity for the histogram of the weight's gradient of linking the neuron o_i to next layer's neurons	10
	Histogram of cosine similarity for the histogram of the weight's gradient of linking the neuron o_i to next layer's neurons	10

Table 5.1: The summary of the featurisation $\phi(s_t, o_{i,t})$. The term 'Dims' means the number of dimensions.

Algorithm 6 Training Time: Transferable Deletion and Stopping Policies

```
 1: **Input:** Initialised deletion network and stopping network
 2: **for** each RL iteration **do**
 3:     **for** each batch **do**
 4:         Pick source dataset randomly
 5:         **for** each episode **do**
 6:             Initialise base network architecture
 7:             **for** each time step $t$ **do**
 8:                 Predict label $\pi_{\theta_s}(a_{s_t}|s_t) = \arg\max \exp^{\Psi_{\theta_s}(\phi(s_t))}$
 9:                 Sample deletion action $\pi_{\theta_d}(\tilde{a}_t|s_t) \propto \prod_{k=1}^{K} \frac{\exp^{\Phi_{\theta_d}\left(\phi(s_t,o_{i,t}^k)\right)}}{\sum_{j\setminus I_k}^{|O_t^k|} \exp^{\Phi_{\theta_d}\left(\phi(s_t,o_{j,t}^k)\right)}}$
10:                 Update the candidate set $O$ and base learner $f_t$
11:                 Record the triplet $< \boldsymbol{o}_t, \tilde{a}_t, r_t >$
12:                 Record the doublet $< \boldsymbol{s}_t, a_{s_t} >$, and collect the annotation $a_{s_t}^*$
13:             **end for**
14:             Rescale the collected return for each batch
15:         **end for**
16:     **end for**
17:     Update deletion policy $\theta_d$ with the rescaled return
18:     Update stopping policy $\theta_s$ with the collected annotations
19: **end for**
20: **return** Trained Deletion and Stopping Policy
```

a large return, due to the variant scale of accuracy among the datasets. The overall training algorithm is summarised in Algorithm 6.

Training for Deletion and Stopping Policy: To train the deletion policy, the reward $r_t = Acc_t$ is the relevant performance metrics for normal neural network training. To optimise this quantity non-myopically, we define the return of a deletion session as $J_{\mathrm{del}}(\theta_d) = \mathbb{E}[\sum_{t=1}^{\infty} \gamma^{t-1} r_t(s, \pi_{\theta_d}(\cdot, s))]$. We then use a policy gradient to train the policy to optimise the objective $J_{\mathrm{del}}(\theta_d)$.

In addition, we also optimise the stopping network by solving the classification task. During the policy gradient sampling process, we create a heuristic oracle to annotate the stop label with $Acc_{t+\Delta} - Acc_{t-1} < 0.1$ and continue otherwise where Δ are the number of the future time step. Thus, the stopping network should output the label of the stop or continue by observing the state s_t. We optimise $J_{\mathrm{stop}}(\theta_s) =$

Algorithm 7 Test time: Transferable Deletion and Stopping Policies

```
1:  Input: Given a deletion network and stopping network
2:  for each time step t do
3:      Predict label $\pi_s(a_{s_t}|s_t) = \arg\max \exp^{\Psi_{\theta_s}(\phi(s_t))}$
4:      if Continue architecture tuning then
5:          Sample batch deletion action $\pi_{\theta_d}(\tilde{a}_t|s_t) \propto \prod_{k=1}^{K} \frac{\exp^{\Phi_{\theta_d}\left(\phi(s_t, o^k_{i,t})\right)}}{\sum_{j\setminus I_k}^{|O^k_t|} \exp^{\Phi_{\theta_d}\left(\phi(s_t, o^k_{j,t})\right)}}$
6:          Update the candidate set $O$ and base learner $f_t$
7:      end if
8:      Update base learner $f_t$
9:  end for
10: return Fine-tuned base learner $f_t$
```

$-\big((a^*_{s_t})\log(\pi_{\theta_s}(a_{s_t}|s_t)) + (1-a^*_{s_t})\log(1-\pi_{\theta_s}(a_{s_t}|s_t))\big)$ is the cross-entropy loss for binary classification. The combined objective function is then given as:

$$\mathcal{F} = J_{\text{del}}(\theta_d) + J_{\text{stop}}(\theta_s) \tag{5.2}$$

Test Processes: After these two policies are trained, they can be applied to a new target problem. When the new problem and base learner are given, the higher-level stopping policy will make a decision about whether it needs to continue to tune the architecture or not. The deletion policy will then decide which K neurons to delete when it is necessary to tune the structure of the network. If not, the learning framework will directly update the weight with the current architecture. The algorithm in test-time is summarised in Algorithm 7.

5.7 Experiments

We then validate the proposed learning to fine-tune the framework in a transfer learning problem. We further evaluate the performance of both deletion and stopping networks in terms of the cross-domain but related context, and different trained parameters but same base-network architecture.

5.7.1 Experiment Setting

Implementation: We implement the learning to fine-tune framework with Pytorch [Paszke et al., 2017]. The Pytorch deep-learning tool supports the dynamic computation graph, and runs it according to any defined order of operation. It enables the network to train with arbitrary order and size of the parameter. Based on this, we can use the strength of the dynamic computation graph by reinitialising a new network architecture without exhaustively masking both the forward value and backward gradient for each neuron. Similar to the method in Chapter 4.1, our coordinate-wise deletion policy is able to feed all the candidate sets O at once and then compute their softmaxed probability. This enables the deletion policy to train and test with any size of network architecture.

Dataset: We evaluate our proposed method on a hand-written image dataset with multiple language recognition problems. The Omniglot dataset contains 50 alphabets and 1,623 characters in total. Each language has roughly 14-55 characters, and each character contains exactly 20 black and white images. The default split in the dataset is divided into two groups: the background set and the evaluation set. The background set includes 30 alphabets, while the evaluation set contains the rest of the alphabets. Generally, this dataset is used for transfer- or meta-learning by giving the background set alphabets for learning background language-agnostic knowledge, and then studying the efficiency and efficacy of learning to recognise characters in the evaluation set alphabets. We evaluate the learning to fine-tune framework on this dataset. We divide the background set randomly into two equally sized subsets to learn a pre-trained base learner and to meta-learn the fine-tuning task. We use one subset to train the base learner and another to train how to fine-tune with both the deletion and stopping policies. Once the policies are well-trained, we evaluate them based on their performance by fine-tuning the base learner on the evaluation set.

Architecture: We use a modified LeNet as the base neural network architecture with 4 convolutional layers and 2 fully connected layers. Both of the convolutional and fully connected linear operations are followed by a ReLU non-linear operation. The first convolutional layer has 16 filters, with a window size of 10×10, which is followed by 2×2 max pooling. The second convolutional layer has 32 filters, with a window size of 5×5, and followed by 2×2 max pooling. The third convolutional layer has 32 filters, with a window size of 4×4, which is followed by 2×2 max pooling. The fourth convolutional layer has 64 filters, with a window size of 4×4. We then flatten the features map and follow witha fully connected layer with 512 neurons and number of classes as output size.

The deletion and stopping networks are fully connected feedforward networks.

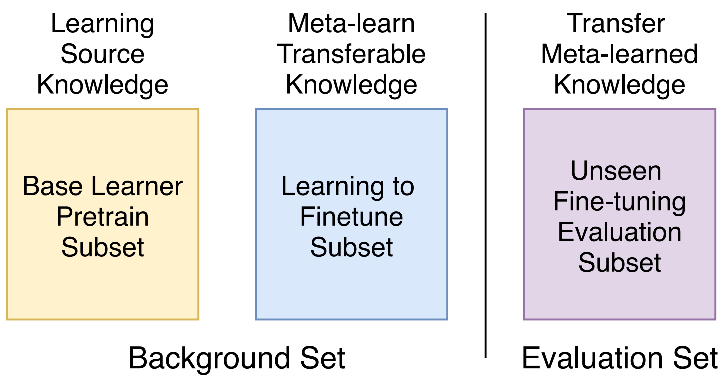

Learning Source Knowledge
Meta-learn Transferable Knowledge
Transfer Meta-learned Knowledge
Base Learner Pretrain Subset
Learning to Finetune Subset
Unseen Fine-tuning Evaluation Subset
Background Set
Evaluation Set

Figure 5.1: Illustration of learning transferable fine-tuning rule on an Omniglot dataset. Note that this is a class-wise split, so the label space of each subset is disjointed. Each class contains its own train/test splits.

The deletion policy is composed of 2 hidden layers with entering the neural features with 91 dimensional after inputting the 91-dimensional neural input features, and 1-dimensional output corresponding to the deletion scores for the corresponding neuron.The first hidden layer has 50 neurons, while the second hidden layer has 20 hidden units. The stopping policy contains 2 layers, and each layer has 10 hidden units. Moreover, the stopping policy has 2 output units, which correspond to the stopping and continuing actions.

Settings: For the base network, we used the Adam optimiser with the Pytorch default setting, and then set the initial learning rate $\eta = 0.0001$ to both train from scratch and fine-tune the target problem. Learning the general character recognition knowledge from scratch, the network is trained by a stochastic gradient descent with 50 epochs and a batch size of 256. After the network is well-trained, we apply the deletion and stopping policies to fine-tune the model on the target set with 60 epochs, where the epochs correspond to the total RL time steps. Here, we simplify the fine-tuning process by replacing the SGD with a batch gradient descent.

Regarding the deletion and stopping networks, both use the Adam optimiser by setting the initial learning rate $\eta = 0.001$. They are trained with 10000 reinforcement learning iterations. For the reinforcement learned deletion policy, the discount factor is set to $\gamma = 0.99$ and $K = 10$ neurons are deleted in every RL time step. Moreover, they are trained by observing the transfer learning experience of any-way (5-15 characters cross languages in the training subsets) and any-shot (0.05%-95% of training samples, and the rest are grouped as a test set) from the given training subset. Here, any-way and any-shot mean that at each sample of a task in the multi-task training, the number of ways of performing multi-class recognition, and number of samples for fine-tuning are chosen randomly to promote invariance in these factors. It enables these policies to learn a general knowledge for both few-shot and middle-shot tuning tasks.

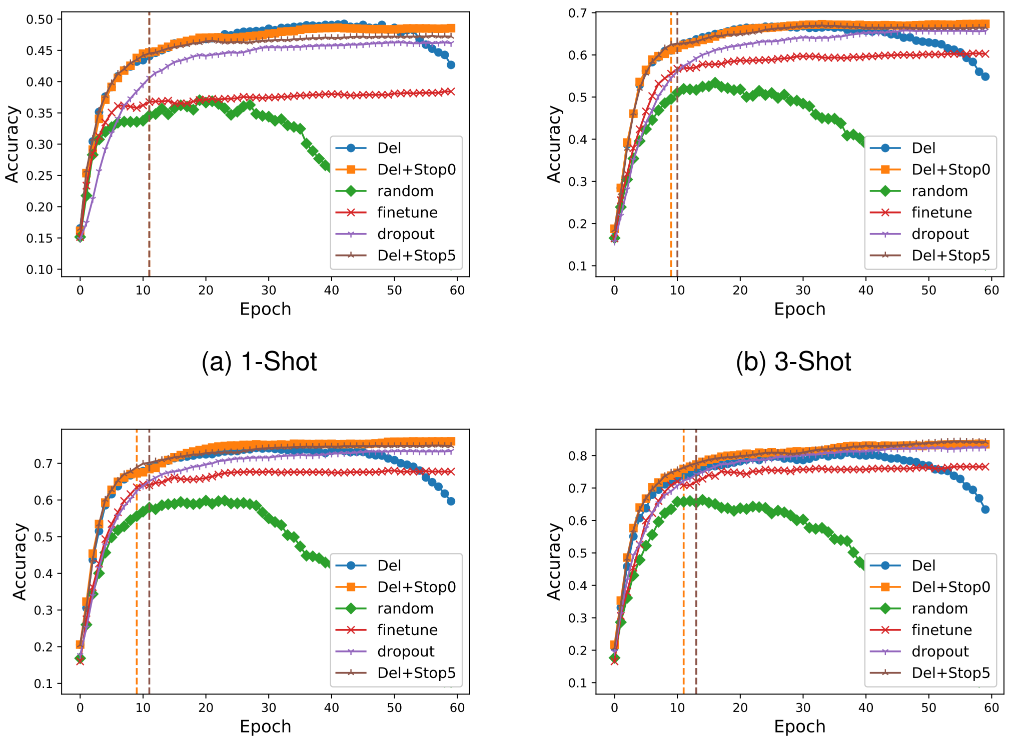
In each RL iteration, we randomly sample 4 conditions of ways and shots from the training subset. For each condition, we sample 8 different trajectories. In total, the RL collected training samples from these 32 rollouts to train both deletion and stopping networks. For the deletion policy, we average the logarithm for numerical stability $logp(\tilde{a}_i) = \frac{1}{K}\sum_k^K log(\pi(a_{I_k,t}))$.

Baseline: We compare the proposed framework with 3 baseline methods; we first compare it with the random deletion policy, which is a simple baseline that allows us to verify whether our model has learned a smart neuron deletion policy, or if any random network shrinking strategy is adequate. Subsequently, we compare it with the normal fine-tuning process, and then compare the normal fine-tuning process with dropout. Both of these two baseline methods are the SGD updates without the tuning network architecture. The difference is that the dropout baseline introduces noise and redundancy to improve the generalisation performance.

Evaluation Metrics: To evaluate effectiveness and efficiency, we first use the accuracy learning curve as an evaluation metric to measure and evaluate both learning speed and the asymptotic performance on the unseen evaluation subset. Meanwhile, RL should optimise testing accuracy on the learning-to-fine-tune subset. We next adopt an evaluation metric to measure both accuracy and compression rate during the fine-tuning process. Ideally, the base learner can achieve a better performance at a higher compression rate. This metric evaluates the performance of our architecture tuning task.

5.7.2 Results

We now evaluate the presented method in terms of effectiveness and efficiency. According to the Figure 5.2, the results indicate that our method achieves faster convergence speed and better performance on a fine-tuning task. It is important to note that this evaluation is for fine-tuning a novel held-out task that is completely different from any task observed during training. In addition, we evaluate the proposed scheme on an unseen initial condition of the base learner. Thus, a good performance here verifies that our method is able to generalise to unseen language domains within Omniglot.



(a) 1-Shot

(b) 3-Shot

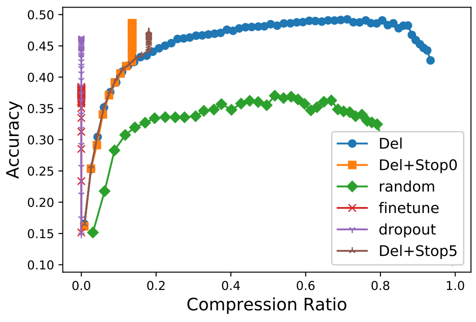
(c) 5-Shot

(d) 65% Training Samples

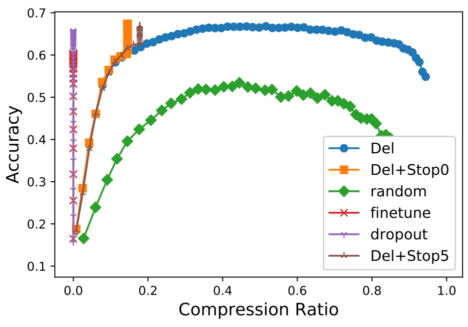
Figure 5.2: Accuracy and learning speed evaluation. The vertical lines indicate the epoch of stopping deletion. Del: deletion policy. Del+Stop 0: Deletion and stopping policy, and the stopping label is according to $Acc_{t+\Delta} - Acc_{t-1} < 0.1$,where $\Delta = 0$. Del+Stop 0: Deletion and stopping policy, and the stopping label is according to $Acc_{t+\Delta} - Acc_{t-1} < 0.1$,where $\Delta = 5$. random: random deletion without stopping

Accuracy vs Learning Speed: We first analyse the result of the proposed deletion and stopping scheme in a few shot-learning settings. According to the Figure 5.2 (a,b,c), the proposed scheme performs consistently better than other competitors over the entire RL time horizon. This indicates that deleting the useless neurons improves both generalisation and learning speed in few-shot fine-tuning. Although the dropout can alleviate this overfitting by regularising the network during the fine-tuning process, it still learns slower and worse than the proposed structure tuning scheme. Thus, it is necessary to prune the size of a complex network to improve the fine-tuning process.

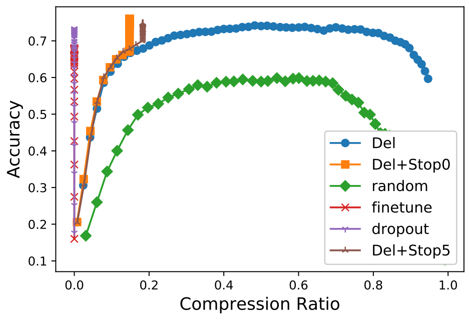
We next investigate the presented method's performance in a middle-shot learning setting (65% of samples of a given dataset are used to fine-tune, while the rest are treated as test samples). As illustrated in the Figure 5.2 (d), the presented method trains the model faster and, as a result, it performs slightly better than the dropout on



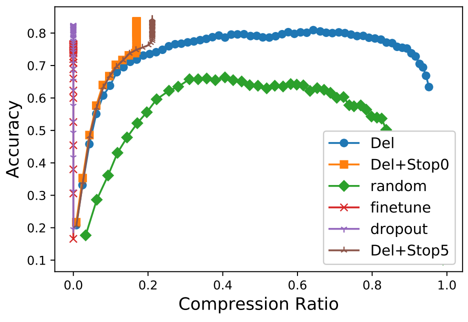
(a) 1-Shot



(b) 3-Shot



(c) 5-Shot



(d) 65% Training Samples

Figure 5.3: Evaluation of both Accuracy and Compression Rate

the target problem. This demonstrates that removing the neurons also provides lesser benefits to the fine-tuning process when a larger dataset is available. Nevertheless, this is not the only benefit of our approach. While the fine-tuning competitor keeps the same network size, our architecture tuning method has the additional benefit of compressing the architecture.

In addition, the Figure 5.2 further illustrated that stopping policy learns a conservative strategy that prefers to stop the deletion earlier. We can see that the labelling oracle with long-term consideration $\Delta = 5$ is able to train a stopping policy to be less conservative. As shown in the Figure 5.2 (d), the long-term consideration oracle could improve slightly better performance on the middle shot learning when the fine-tuning process is finished.

Accuracy vs Compression Rate: We now analyse the presented method on the evaluation metric of the accuracy vs compression rate for considering the network capacity. According to the Figure 5.3, both learned and random deletion policies reduce the size of the network by removing most of the neurons. However, these two approaches will continue deleting all neurons, resulting in an unstable performance without realising the performance is decreased. As a result of these deletion policies, the model will perform poorly since the network is over-pruned. Conversely, the proposed deletion and stopping scheme perform the classification well and are able to predict the specific compression rate at which to stop without over-pruning the size of the network. It is straightforward that the long-term consideration stopping policy would encourage to delete more neurons on the network. This leads to a higher compression rate than the stopping policy with short-term consideration. Meanwhile, the fine-tuning and dropout policies achieve reasonable accuracy, but do not provide any compression benefit. Thus, we show that learning when to stop the deletion is helpful to stabilise the model's performance.

5.8 Summary

In this chapter, we propose a meta-learning approach to learn how to fine-tune the neural network for transfer learning tasks. We propose to learn how to delete the neurons to prune the size of the network, as well as learn when to stop the pruning to stabilise the architecture tuning process. This enables the architecture and weight to be updated simultaneously. In the experiment, we verified that the presented method could generalise well on different domains and different well-trained conditions with the same architecture. In addition, we empirically show that this structure pruning scheme could improve the fine-tuning process in terms of efficiency and effectiveness.

Chapter 6

Conclusion and Future Work

6.1 Summary

This thesis could be summarised into two major goals. The first goal was to present meta-learning about the active learning process for improving the efficiency of machine data annotation for machine learning. In pursuit of the first goal, we first identified the existence of a non-stationary phenomenon in active learning criterion selection when learning active learning online. We subsequently developed a non-stationary bandit learning algorithm with expert advice, which adds a periodic restart scheme to re-explore and re-exploit the best expert for the given bandit tasks. This method is robust on non-stationarity, and we also proved a worst-case bound to guarantee the performance. In our experiment, the developed dynamic ensemble active learning approach is more data efficient for non-stationary datasets than both alternative approaches to static ensemble online active learning and dynamic ensemble heuristics.

Next, we developed a new learning transferable active learning framework which is able to learn a non-myopic and transferable active learning policy that can apply to any unseen dataset without constraint on the number of dimensions. To achieve this, we presented a framework built on a reinforcement learned policy and a meta-network to dynamically synthesise the policy given a target dataset. The reinforcement learned policy provides a non-myopic solution to an active learning task. The meta-network adapts the policy to a given dataset of any dimensionality by inputting a dataset embedding, and mapping this into the policy of the appropriate dimension for a target dataset.

We show that the framework created by combining these two networks achieves better performance than conventional active learning algorithms. Thus, learned non-myopic and transferable knowledge can help improve the data efficiency of the active learning task. Another significant result in the ablation study is that the expert features used in the framework contribute little to the final task. This means that the framework is able to learn effective non-myopic AL criteria almost entirely driven by raw data.

The second goal of this thesis is to present learning transfer learning to improve both efficiency and efficacy in transfer learning problems. The presented algorithm meta-learns the fine-tuning process, which is the special case of transfer learning. This allows meta-level knowledge to be transferable to the new unseen transfer learning tasks. Unlike the previous other hyperparameter optimisation techniques, our presented meta-learning approach provides a non-myopic update rule for the architecture of the network by deleting useless neurons. Deleting useless neurons can improve the networks generalisation by reducing the models complexity. This technique is useful when fine-tuning a large capacity pre-trained base network for a new target problem. The results indicate that fine-tuning with the deletion policy elicits a reasonable performance from either few-shot or middle-shot learning, in terms of higher accuracy and less training iterations.

6.2 Limitation and Future Work

Together, these meta-learning techniques contribute to the efficiency and efficacy of the full machine learning pipeline, which covers both data annotation and knowledge transfer. This is in contrast to prior work that focuses on meta-learning the core supervised learning problem. However, there are still several limitations. Below, we discuss both of the open questions and their future direction to improve and extend the presented methods.

6.2.1 Learning Active Learning Online

As presented in Chapter 3, conventional active learning is improved by meta-learning the best criterion for each dataset at each timestep. The adopted meta-learning approach is a cold-start online bandit learning algorithm that explores different winning criterion during the active learning process. This cold-start scheme does not exploit any previous knowledge of criteria success that may be available, and instead learns each dataset from scratch. In addition, the proposed bandit algorithm with restart scheme will periodically reset the belief of the best criterion and re-explore, thus potentially wasting time on unnecessary exploration when the best arm is not changing. Moreover, the re-start schedule needs to be set empirically. Both the cold-start and restart schemes leave the question about how to further improve the efficiency without frequent re-exploration either cross or within the dataset.

One possible direction to reduce the frequency of re-exploration within a dataset is to upgrade the non-stationary-based algorithm using a change point detection scheme. Unlike the restart scheme, which periodically resets the parameters, the parameters should only be reset when necessary. This could cover both stationary and non-stationary environments efficiently by attempting to detect when environmental changes.

6.2.2 Learning Transferable Active Learning Policy

Although we achieved good efficiency for the tabular-style UCI datasets, our presented framework is weak on image datasets. The presented framework is still weak on image datasets. This may be due to the fact that the current dataset embedding is more suitable for tabular rather than image-style data. A potential route to address this could be to learn a raw dataset-embedding, for example with deep set-embedding, rather than using a hand-crafted dataset embedding.

6.2.3 Transferable and Online Active Learning

With regard to the active learning component in general, our contributions to dynamic online ensemble learning and transferable policy learning address different challenges of the AL problem. An ideal solution would provide both of the favourable properties of these algorithms simultaneously. This could be achieved, for example, by learning to fine-tune the MLP-GAL policy for a target dataset on the fly. In other words, the MLP-GAL policy should be treated as a warm-start and extended with the ability to update online, as in the DEAL-REXP4 algorithm.

6.2.4 Learning to Fine-tune

Our final goal with learning to fine-tune is to update both architecture and weight simultaneously to achieve a better performance and good structure during the fine-tuning phase. We have thus far only implemented neuron deletion, with the motivation of removing unrelated neurons from a more general source. However, it may be beneficial to support neuron addition to represent specialist information about a more specific target. Additionally, we currently only learn how to update the architecture. Therefore, the next step could be to unify this algorithm with existing (parameter-level) learning-to-learn approaches, and thus improve both the parameter and architecture update strategies.

Finally, this work provides a basic of architecture tuning, which could also be developed for more general automatic online learning of network architecture when learning from scratch.

Bibliography

Abe, N. and Mamitsuka, H. (1998). Query learning strategies using boosting and bagging. In *ICML*.

Andrychowicz, M., Denil, M., Gómez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In *NIPS*.

Ashok, A., Rhinehart, N., Beainy, F., and Kitani, K. M. (2017). N2n learning: Network to network compression via policy gradient reinforcement learning. In *ICLR*.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002a). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002b). The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77.

Ba, L. J. and Caruana, R. (2014). Do deep nets really need to be deep? In *NIPS*.

Bachman, P., Sordoni, A., and Trischler, A. (2017). Learning algorithms for active learning. In *ICML*.

Baker, B., O. Gupta, N. N., and Raskar, R. (2017). Designing neural network architectures using reinforcement learning. In *ICLR*.

Balaji, Y., Sankaranarayanan, S., and Chellappa, R. (2018). Metareg: Towards domain generalization using meta-regularization. In *NIPS*.

Baram, Y., El-Yaniv, R., and Luz, K. (2004). Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5:255–291.

Barber, D. (2012). *Bayesian Reasoning and Machine Learning*. Cambridge University Press, New York, NY, USA.

Bertinetto, L., Henriques, J. a. F., Valmadre, J., Torr, P., and Vedaldi, A. (2016). Learning feed-forward one-shot learners. In *NIPS*.

Besbes, O., Gur, Y., and Zeevi, A. (2014). Stochastic multi-armed-bandit problem with non-stationary rewards. In *NIPS*.

Besbes, O., Gur, Y., and Zeevi, A. J. (2015). Non-stationary stochastic optimization. *Operations Research*, 63:1227–1244.

Beygelzimer, A., Langford, J., Li, L., Reyzin, L., and Schapire, R. E. (2011). Contextual bandit algorithms with supervised learning guarantees. In *AISTATS*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chattopadhyay, R., Wang, Z., Fan, W., Davidson, I., Panchanathan, S., and Ye, J. (2012). Batch mode active sampling based on marginal probability distribution matching. *ACM Transactions on Knowledge Discovery from Data*, 7(13).

Chowdhury, A. R., Lin, T., Maji, S., and Learned-Miller, E. (2016). One-to-many face recognition with bilinear cnns. In *WACV*.

Chu, H. and Lin, H. (2016). Can active learning experience be transferred? In *ICDM*.

Chu, W., Li, L., Reyzin, L., and Schapire, R. E. (2011). Contextual bandits with linear payoff functions. In *AISTATS*.

Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1995). Active learning with statistical models. In *NIPS*.

Cui, Y., Song, Y., Sun, C., Howard, A., and Belongie, S. (2018). Large scale fine-grained categorization and domain-specific transfer learning. In *CVPR*.

Cun, Y. L., Denker, J. S., and Solla, S. A. (1990). Optimal brain damage. In *NIPS*.

Donmez, P., Carbonell, J. G., and Bennett, P. N. (2007). Dual strategy active learning. In *ECML*.

Evgeniou, T. and Pontil, M. (2004). Regularized multi–task learning. In *SIGKDD*.

Fan, H., Zheng, L., Yan, C., and Yang, Y. (2018). Unsupervised person re-identification: Clustering and fine-tuning. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 14(4):83:1–83:18.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874.

Fei-Fei, L., Fergus, R., and Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *ICML*.

Garivier, A. and Moulines, E. (2008). On upper-confidence bound policies for non-stationary bandit problems. In *ALT*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. The MIT Press.

Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T. L. (2018). Recasting gradient-based meta-learning as hierarchical bayes. In *ICLR*.

Greensmith, E., Bartlett, P. L., and Baxter, J. (2004). Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5:1471–1530.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

Ha, D., Dai, A. M., and Le, Q. V. (2017). Hypernetworks. In *ICLR*.

Han, S., Mao, H., and Dally, W. J. (2015). Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *ICLR*.

Hassibi, B. and Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. In *NIPS*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*.

He, Y., Lin, J., Liu, Z., Wang, H., Li, L., and Han, S. (2018). AMC: automl for model compression and acceleration on mobile devices. In *ECCV*.

Hochbaum, D. S. and Shmoys, D. B. (1985). A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184.

Hospedales, T. M., Gong, S., and Xiang, T. (2012). A unifying theory of active discovery and learning. In *ECCV*.

Hospedales, T. M., Gong, S., and Xiang, T. (2013). Finding rare classes: Active learning with generative and discriminative models. *IEEE Transactions on Knowledge and Data Engineering*, 25(2):374–386.

Hsu, W.-N. and Lin, H.-T. (2015). Active learning by learning. In *AAAI*.

Huang, G., Liu, Z., v. d. Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *CVPR*.

Huang, S.-J., Jin, R., and Zhou, Z.-H. (2010). Active learning by querying informative and representative examples. In *NIPS*.

Jaderberg, M., Vedaldi, A., and Zisserman, A. (2014). Speeding up convolutional neural networks with low rank expansions. In *BMVC*.

Joshi, A. J., Porikli, F., and Papanikolopoulos, N. (2009). Multi-class active learning for image classification. In *CVPR*.

Kakade, S. M., Shalev-shwartz, S., and Tewari, A. (2008). Efficient bandit algorithms for online multiclass prediction. In *ICML*.

Kapoor, A., Grauman, K., Urtasun, R., and Darrell, T. (2007). Active learning with gaussian processes for object categorization. In *ICCV*.

Kober, J. and Peters, J. R. (2009). Policy search for motor primitives in robotics. In *NIPS*.

Konyushkova, K., Sznitman, R., and Fua, P. (2017a). Learning active learning from data. In *NIPS*.

Konyushkova, K., Sznitman, R., and Fua, P. (2017b). Learning active learning from data. In *NIPS*.

Korattikara Balan, A., Rathod, V., Murphy, K. P., and Welling, M. (2015). Bayesian dark knowledge. In *NIPS*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*.

Kumar, A. and Daumé, III, H. (2012). Learning task grouping and overlap in multi-task learning. In *ICML*.

Kupcsik, A. G., Deisenroth, M. P., Peters, J., and Neumann, G. (2013). Data-efficient generalization of robot skills with contextual policy search. In *AAAI*.

Langford, J. and Zhang, T. (2008). The epoch-greedy algorithm for multi-armed bandits with side information. In *NIPS*.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, Y. and Choi, S. (2018). Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*.

Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *SIGIR*.

Li, D., Yang, Y., Song, Y., and Hospedales, T. M. (2018). Learning to generalize: Meta-learning for domain generalization. In *AAAI*.

Li, K. and Malik, J. (2017). Learning to optimise. In *ICLR*.

Liu, H., Simonyan, K., and Yang, Y. (2018). Darts: Differentiable architecture search. In *ICLR*.

Loy, C. C., Hospedales, T. M., Xiang, T., and Gong, S. (2012). Stream-based joint exploration-exploitation active learning. In *CVPR*.

Luo, P., Zhu, Z., Liu, Z., Wang, X., and Tang, X. (2016). Face model compression by distilling knowledge from neurons. In *AAAI*.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. (2018). A simple neural attentive meta-learner. In *ICLR*.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *ICML*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518:529–533.

Muandet, K., Balduzzi, D., and Scholkopf, B. (2013). Domain generalization via invariant feature representation. In *ICML*.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Pang, K., Dong, M., Wu, Y., and Hospedales, T. M. (2018a). Dynamic Ensemble Active Learning: A Non-Stationary Bandit with Expert Advice. In *ICPR*.

Pang, K., Dong, M., Wu, Y., and Hospedales, T. M. (2018b). Meta-learning transferable active learning policies by deep reinforcement learning. In *ICML AutoML 2018*.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. *ICLR*.

Peters, J. and Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697.

Ravi, S. and Larochelle, H. (2017). Optimization as a model for few-shot learning. In *ICLR*.

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q. V., and Kurakin, A. (2017). Large-scale evolution of image classifiers. In *ICML*.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407.

Romero, A., Carrier, P. L., Erraqabi, A., Sylvain, T., Auvolat, A., Dejoie, E., Legault, M., Dube, M., Hussin, J. G., and Bengio, Y. (2017). Diet networks: Thin parameters for fat genomics. In *ICLR*.

Roy, N. and McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. In *ICML*.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229.

Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Seung, H. S., Opper, M., and Sompolinsky, H. (1992). Query by committee. In *COLT*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. In *NIPS*.

Strehl, A. L., Mesterharm, C., Littman, M. L., and Hirsh, H. (2006). Experience-efficient learning in associative bandit problems. In *ICML*.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. (2018). Learning to compare: Relation network for few-shot learning. In *CVPR*.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *CVPR*.

Tai, C., Xiao, T., Wang, X., and E, W. (2015). Convolutional neural networks with low-rank regularization. In *ICLR*.

Tong, S. and Koller, D. (2002). Support vector machine active learning with applications to text classification. In *ICML*.

Wang, Z., Du, B., Zhang, L., Zhang, L., and Jia, X. (2017). A novel semisupervised active-learning algorithm for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(6):3071–3083.

Wang, Z. and Ye, J. (2015). Querying discriminative and representative samples for batch mode active learning. *ACM Transactions on Knowledge Discovery from Data*, 9(3):17:1–17:23.

Wei, C.-Y., Hong, Y.-T., and Lu, C.-J. (2016). Tracking the best expert in non-stationary stochastic environments. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *NIPS*.

WEI, Y., Zhang, Y., Huang, J., and Yang, Q. (2018). Transfer learning via learning to transfer. In *ICML*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

Woodward, M. and Finn, C. (2017). Active one-shot learning. *CoRR*, abs/1702.06559.

Xu, Z., Yu, K., Tresp, V., Xu, X., and Wang, J. (2003). Representative sampling for text classification using support vector machines. In *ECIR*.

Yang, J., Yan, R., and Hauptmann, A. G. (2007). Cross-domain video concept detection using adaptive svms. In *ACMMM*.

Yu, J. Y. and Mannor, S. (2009). Piecewise-stationary bandit problems with side observations. In *ICML*.

Yu, K., Bi, J., and Tresp, V. (2006). Active learning via transductive experimental design. In *ICML*.

Yu, X., Liu, T., Wang, X., and Tao, D. (2017). On compressing deep models by low rank and sparse decomposition. In *CVPR*.

Zhao, T., Hachiya, H., Niu, G., and Sugiyama, M. (2011). Analysis and improvement of policy gradient estimation. In *NIPS*.

Zhu, X. and Wu, X. (2004). Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review*, 22(3):177–210.

Zhu, X., Wu, X., and Yang, Y. (2004). Error detection and impact-sensitive instance ranking in noisy datasets. In *AAAI*.

Zoph, B. and Le, Q. V. (2017). Neural architecture search with reinforcement learning. In *ICLR*.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2017). Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012.