Meta-level Learning for the Effective Reduction of Model Search Space

Abbas Raza Ali

Faculty of Science and Technology

Bournemouth University

A thesis submitted for the degree of

Doctor of Philosophy

July 2019

Statement of Originality

This thesis is solely the work of its author. No part of it has previously been submitted for any degree, or is currently being submitted for any other degree. To the best of my knowledge, any help received in preparing this thesis, and all sources used, have been duly acknowledged.

Abstract

The exponential growth of volume, variety and velocity of the data is raising the need for investigation of intelligent ways to extract useful patterns from the data. It requires deep expert knowledge and extensive computational resources to find the mapping of learning methods that leads to the optimized performance on a given task. Moreover, numerous configurations of these learning algorithms add another level of complexity. Thus, it triggers the need for an intelligent recommendation engine that can advise the best learning algorithm and its configurations for a given task. The techniques that are commonly used by experts are; trial-and-error, use their prior experience on the specific domain, etc. These techniques sometimes work for less complex tasks that require thousands of parameters to learn. However, the state-of-the-art models, e.g. deep learning models, require well-tuned hyper-parameters to learn millions of parameters which demand specialized skills and numerous computationally expensive and time-consuming trials. In that scenario, Meta-level learning can be a potential solution that can recommend the most appropriate options efficiently and effectively regardless of the complexity of data. On the contrary, Meta-learning leads to several challenges; the most critical ones being model selection and hyper-parameter optimization.

The goal of this research is to investigate model selection and hyper-parameter optimization approaches of automatic machine learning in general and the challenges associated with them. In machine learning pipeline there are several phases where Meta-learning can be used to effectively facilitate the best recommendations including 1) pre-processing steps, 2) learning algorithm or their combination, 3) adaptivity mechanism parameters, 4) recurring concept extraction, and 5) concept drift detection. The scope of this research is limited to feature engineering for problem representation, and learning strategy for algorithm and its hyper-parameters recommendation at Meta-level.

There are three studies conducted around the two different approaches of automatic machine learning which are model selection using Meta-learning and hyper-parameter optimization. The first study evaluates the situation in which the use of additional data from a different domain can improve the performance of a meta-learning system for time-series forecasting, with focus on cross-domain Meta-knowledge transfer. Although the experiments revealed limited room for improvement over the overall best base-learner, the meta-learning approach turned out to be a safe choice, minimizing the risk of selecting the least appropriate base-learner. There are only 2% of cases recommended by meta-learning that are the worst performing base-learning methods. The second study

proposes another efficient and accurate domain adaption approach but using a different meta-learning approach. This study empirically confirms the intuition that there exists a relationship between the similarity of the two different tasks and the depth of network needed to fine-tune in order to achieve accuracy comparable with that of a model trained from scratch. However, the approach is limited to a single hyper-parameter which is fine-tuning of the network depth based on task similarity. The final study of this research has expanded the set of hyper-parameters while implicitly considering task similarity at the intrinsic dynamics of the training process. The study presents a framework to automatically find a good set of hyper-parameters resulting in reasonably good accuracy, by framing the hyper-parameter selection and tuning within the reinforcement learning regime. The effectiveness of a recommended tuple can be tested very quickly rather than waiting for the network to converge. This approach produces accuracy close to the state-of-the-art approach and is found to be comparatively 20% less computationally expensive than previous approaches. The proposed methods in these studies, belonging to different areas of automatic machine learning, have been thoroughly evaluated on a number of benchmark datasets which confirmed the great potential of these methods.

Contents

List of Tables

List of Figures

Terminologies and Mathematical Definitions

Table 1: Terminologies

Symbol	Description
$\mathcal{A}$	Set of finite actions, $a \in \mathcal{A}$
$\mathcal{D}$	Input/class dataset
s	Possible experiences of $\mathcal{D}$
T_r	Training experience of $\mathcal{D}$ at any given moment
$\mathcal{D}_{T_r}$	Domain-specific dataset
f	Function/Predictive model
p	Probability function
α	Learning rate
$\mathcal{L}$	Loss function
ϵ	Cross-validation error on training data
ϕ	Classification errors
π_{theta}	Set of base-level classifiers
θ	Parameters of each classifier
L_{mu}	Supervised learning
μ	Set of chosen base-algorithms
p	Momentum
ϕ	Probability distribution function
ρ	Correlation coefficient
ω	Training window (adaptability)
ς	Step size (adaptability)
τ	Tune temperature (adaptability)
λ	Set of hyper-parameters
R	Reward function
$\mathcal{S}$	Set of finite states, $s \in \mathcal{S}$
$\mathcal{T}$	State transition probability
γ	Discount factor
π	Policy

Glossary of Terms

A

A2C Actor-Critics. 37

AL Average Linkage. 36

ARIMA Auto-regressive Integrated Moving Average. 22, 32, 34, 36, 58, 59, 61, 62, 65–67

ARR Adjusted Ratio of Ratios. 15, 22

Auto-ML Automatic Machine Learning. 1, 4, 28, 30, 44, 45, 50, 68, 83

Auto-WEKA Automatic model selection and hyper-parameter optimization in WEKA. 28, 30

Average Nodes Average Nodes Learner. 107

B

b Number of Binary Features. 105

BLL Base-level Learning. 4, 20, 21, 23, 30, 31, 38, 42, 43, 52, 65, 95

C

C4.5 C4.5 Decision Tree algorithm. 15, 21, 22, 35

C5.0 boost C5.0 Adaptive Boosting. 15, 20–22, 35, 107

C5.0 rules C5.0 Rule Induction. 21, 22, 35, 107

C5.0 tree C5.0 Decision Tree. 15, 21, 22, 35, 107

CANCOR Canonical Correlation. 105

CART Classification and Regression Trees. 21, 22, 41

CASH Combined Algorithm Selection and Hyper-parameter Optimization. 28, 30

CASTLE Causal Structure for Inductive Learning. 21

CBR Case-based Reasoning. 14

CL Complete Linkage. 36

CN2 CN2 Induction Algorithm. 21, 35

CNN Convolutional Neural Network. 45–48, 74, 80, 86, 87, 89, 98

CORR Mean Absolute Correlation Coefficient. 31, 36, 106

CoV Coefficient of Variation. 33

CV Cross-Validation. 21, 22, 31

D

DBS DB-Scan. 36

DCT Dataset Characterization Tool. 14–17, 19, 30, 34

DDPG Deep Deterministic Policy Gradients. 49

Decision Nodes Decision Nodes Learner. 35, 107

DiscFunc Number of Discriminant Functions. 106

DL Deep Learning. 12, 44, 45, 47, 95

DMA Data Mining Advisor. 25, 27, 29, 34

DNN Deep Neural Networks. 1, 4, 19, 20, 36, 37, 43–45, 47, 50, 69, 82, 83, 94, 95, 97

DP Dynamic Programming. 102

DSIT Descriptive, Statistical, and Information-Theoretic. 2, 13–19, 25–27, 30, 34–37, 40, 43

DT Decision Trees. 16, 17, 19, 32, 36, 37, 62, 64–66

DW Durbin-Watson statistic of regression residual. 106

E

e-LICO e-Laboratory for Interdisciplinary Collaborative Research. 28, 29, 34

ENAS Efficient Neural Architecture Search. 49, 84, 94, 97

e-NN Elite-Nearest Neighbour. 35, 107

EoD Examples of Datasets. 5, 11, 12, 19, 23, 28, 44, 65, 68, 69, 95, 96

ES Exponential Smoothing. 22, 36

F

FC Fully-Connected. 76, 77, 86

FF Farthest First. 36

FFT Fast Fourier Transform. 60

FLD Fisher's Linear Discriminant. 21

FRACT Relative proportion of largest Eigenvalue. 106

G

GPU Graphics Processing Unit. 47, 49, 73, 90, 91

H

HC Entropy of Classes. 106

HCX Joint Entropy of Classes. 106

HPO Hyper-parameter Optimization. 1, 4, 28, 30, 45, 47, 48, 50, 71, 80, 94, 95, 98

I

IBL Instance-based Learning. 21, 22, 35

ICA Independent Component Analysis. 17

ID3 Iterative Dichotomiser 3. 35

IDA Intelligent Discovery Assistant. 26–28, 34

ILSVRC ImageNet Large-Scale Visual Recognition Challenge. 47, 76

INDCART Inductive CART. 21, 35

K

k Number of Classes. 105

KD Knowledge Discovery. 26

k-M k-Means. 36

k-NN k-Nearest Neighbour. 17, 19–22, 31, 32, 35, 36, 107

KURT Kurtosis. 105, 106

L

LazyDT Lazy Decision Trees. 35

LDA Linear Discriminant Analysis. 15, 17, 21, 22, 35, 107

LSTM Long Short-term Memory. 85

Ltree Linear Discriminant Trees. 21, 22, 35, 107

LVQ Learning Vector Quantization. 35

M

M Mixture Models. 36

MA Moving Average. 22, 32, 57–59, 61, 64–67, 69

MAE Mean Absolute Error. 21–23

MAML Model-Agnostic Meta-Learning. 43, 44, 48

MARS Multivariate Adaptive Regression Splines. 22

MCMLPS Multi-component, Multi-level Predictive System. 4

MCX Average Mutual Information between Class and Nominal Features. 106

MDP Markov Decision Process. 47

MDS Multi-dimensional Scaling. 17

ME Meta-example. 35, 41, 51, 52

METAL Meta-learning Assistant. 6, 15, 25, 27, 30, 31

METALA Meta-learning Architecture. 25, 27

Meta-RL Meta-Reinforcement Learning. xiii, 4, 19, 44, 45, 48, 53, 83–85, 89, 97, 98

MF Meta-feature. xi, 1, 2, 5, 9–21, 23–27, 30–35, 38, 40–44, 51, 52, 56, 57, 59–62, 65, 66, 68, 69, 82, 95, 96

MK Meta-knowledge. 1, 5, 11, 12, 20, 23, 28, 30, 31, 33, 35, 37, 51–53, 55–57, 61, 62, 64, 65, 67, 71

ML Machine Learning. 1, 5, 9, 11, 12, 24, 28, 30, 32, 37, 44, 47, 50, 53, 83, 95, 97

MLL Meta-level Learning. xiii, 1–6, 9–12, 15–17, 19–21, 23–35, 37–45, 47, 49–53, 55–57, 61, 64–69, 71, 95–98

MLP Multi-layer Perceptron. 21, 22, 31, 33–36

MLR Multiple Linear Regression. 22

MLT Machine Learning Toolbox. 24, 27

MSE Mean Squared Error. 22, 31, 36

MSHPO Model Selection and Hyper-parameters Optimization. 3, 53, 54, 97

N

N Total Instances. 105

n Number of Numeric features. 105

NAS Neural Architecture Search. 1, 45, 48–50, 84, 94, 95, 97, 98

NB Naive Bayes classifier. 15, 17, 21, 22, 35, 36, 107

NBT Naive Bayes/Decision-Tree. 35

NN Neural Network. 16, 22, 32, 36, 37, 59, 62, 64–67, 83

NoiseRaio Noise to Signal Ratio. 106

O

OC1 Oblique Classifier-1. 35

OneR One Rule Learner. 22, 35, 36

OpenML Open Machine Learning. 8, 9, 11, 12

OPGA On-policy Gradient Algorithms. 49

OPSRL Optimistic Posterior Sampling for Reinforcement Learning. 36

P

p Number of Features. 105

PaREn Pattern Recognition Engineering. 26, 28, 34

PCA Principal Component Analysis. 17, 19, 107

PDF Probability Density Function. 19

PEBLS Parallel Exemplar-Based Learning System. 35

PMF Probability Mass Function. 19, 20, 77, 79

PNAS Progressive Neural Architecture Search. 48, 49

PPO Proximal Policy Optimization. 49

PPR Projection Pursuit Regression. 22

Q

QPC Quality of Projected Clusters. 10, 17

Quadra Quadratic Classifier. 21

R

r Number of Training instances. 105

Randomly Chosen Nodes Randomly Chosen Nodes Learner. 35, 107

RapidAnalytics open-source data-mining and predictive analysis solution. 29

RBF Radial-basis Function. 21, 22, 32, 35, 36, 62

ReLU Rectified Linear Units. 87

ResNet Residual Networks. 45, 86

RF Random Forests. 22, 36, 41, 61

Ripper Rule Learner. 15, 21, 22, 35, 107

RL Reinforcement Learning. 35–39, 43, 44, 47–49, 84, 94, 97, 98

RMSE Root Mean Squared Error. 22

RNN Recurrent Neural Network. 35, 46, 48, 85–87, 98

RW Random Walk. 22, 36

S

s Number of Nominal features. 105

S/D Ratio Homogeneity of Covariances. 105

SKEW Skewness. 105, 106

SL Single Linkage. 36

SMAPE Symmetric Mean Absolute Percentage Error. xi, 22, 33, 58, 59, 61, 62, 64–66, 106

SMART Smooth Multiple Additive Regression Technique. 21

SMBO Sequential Model-Based Optimization. 11, 28, 48

SMOTE Synthetic Minority Over-sampling TEchnique. 61

SNN Shared Nearest Neighbours. 36

SP Spectral Clustering. 36

SRCC Spearman's Rank Correlation Coefficient. 22, 33

STABB Shift To A Better Bias. 24, 26

StatLog Statistical and Logical learning. 13, 14, 16, 25, 27, 30

StdDev Standard Deviation. xi, 58–60, 62, 64–66, 105–107

SVM Support Vector Machines. 17, 21, 22, 31–33, 36, 37, 41, 62, 64–66

SVR Support Vector Regression. 33, 34

T

t Number of Test instances. 105

TL Transfer Learning. 45, 47, 72, 76, 79, 80, 82

TPOT Tree-based Pipeline Optimization Tool. 28

TPU Tensor Processing Unit. 47

TRPO Trust Region Policy Optimization. 36, 49

TS Time-series. 4, 7, 8, 10, 20, 23, 31, 32, 34, 36, 55, 97, 106

U

UCI UCI Machine Learning Repository. 6, 7, 9–11, 15, 16, 28, 30

V

VBMS Variable-bias Management System. 13, 17, 24–26

VGG Visual Geometry Group. 76

W

Wlambda Wilks'lambda Distribution. 106

Worst Nodes Worst Nodes Learner. 107

X

XM X-Means. 36

Acknowledgements

First of all, I would like to thank my supervisors Prof. Bogdan Gabrys and Prof. Marcin Budka for their support, expert advice and invaluable feedback.

I would like to thank my colleagues and friends at Bournemouth University, specially to Manuel, Rashid, Amir and Bassma. I would also like to thank Bournemouth University staff for always being nice and helpful to me. Special thanks to Dr. Emili Balaguer, Dr. Damien Fay and Naomi Bailey.

Not forgetting wife Moona, for her constant support and understanding. Finally, I would like to express my gratitude to my parents for always encouraging me in the right direction in both personal and academic sense.

I would like to dedicate this thesis to my parents and wife...

Chapter 1

Introduction

This chapter presents the Doctoral research, its area, and an overview of the contributions in the space of Meta-level Learning (MLL) and related areas. In order to provide a clear motivation, this chapter outlines the main challenges and goals which lead to the aims and objectives of this research. The details of the research challenges that lead to several research questions can be found in Chapter 2.7.

1.1 Background and Motivation

One of the major challenges in Machine Learning (ML) is to predict when one algorithm is more adequate than another to solve a learning problem (Prudencio et al., 2011). Traditionally, estimating the performance of algorithms involves an intensive trial-and-error process which often demands massive execution time and memory together with the support of expert advice that is not always easy to acquire (Giraud-Carrier et al., 2004). MLL arises as a potential solution of this problem; it uses examples from various domains to produce an ML model, known as Meta-learner, which is responsible for associating the characteristics of a problem with the candidate algorithm giving optimized accuracy. The knowledge which is used by a Meta-learner is acquired from previously solved problems, where each problem is characterized by several features, known as Meta-features (MFs). MFs are combined with performance measures of ML algorithms, e.g., accuracy, to build a Meta-knowledge (MK) database. Learning at the base-level gathers experience within a specific problem, while MLL is concerned with accumulating experience over several learning problems (Giraud-Carrier, 2008).

Along with the MLL, Hyper-parameter Optimization (HPO) and Neural Architecture Search (NAS) are also key methods of Automatic Machine Learning (Auto-ML) (Yao et al., 2019). The goal of HPO is to find a set of hyper-parameters of an ML task which gives optimized performance. It becomes crucial for Deep Neural Networks (DNN) which, in turn, comes with a wide range of hyper-parameter choices. The success of the DNN is mostly credited to its ability to automatically extract the task-dependent features. This automation is now expanding towards architecture engineering to automatically design complex neural architectures, known as NAS.

MLL started to appear in the ML domain in 1980's and was referred to by different, such as, dynamic bias selection (Rendell et al., 1987), algorithm recommender (Brazdil et

al., 2008), etc. Sometimes it is also confused with Ensemble methods (Duch et al., 2011). In order to get a comprehensive view of exactly what MLL is, a number of definitions have been proposed in various studies. Vilalta and Drissi (2002a) and Vanschoren (2011) define MLL as the understanding of how learning itself can become flexible according to the domain or task and how it tends to adapt its behaviour to perform better. Giraud-Carrier (2008) describes it as the understanding of the interaction between the mechanism of learning and concrete context in which that mechanism is applicable. Brazdil et al. (2008) view on MLL is that it is the study of methods that exploit Meta-knowledge to obtain efficient models and solutions by adapting the learning algorithms. To some extent, this definition is followed in this research as well.

Extracting MFs from a dataset plays a vital role in the MLL task. Several MF generation approaches are available to extract a variety of information from previously solved problems. The most commonly used approaches are descriptive (or simple), statistical, information theoretic, landmarking and model-based. The Descriptive, Statistical, and Information-Theoretic (DSIT) features are easy to extract from the dataset as compared to the other approaches. Most of them have been proposed in the same period and are often used together. These approaches are used to estimate the similarity of new data with the already analyzed datasets (Bensusan et al., 2000). Landmarking is the most recent approach that tries to relate the performance of candidate algorithms to the performance obtained by simpler and computationally more efficient learners (Pfahringer et al., 2000). The Model-based approach captures the characteristics of a problem from the structural shape and size of a model induced by the dataset (Peng et al., 2002). The decision tree models are mostly used in this approach, where properties are extracted from the tree, such as tree depth, shape, nodes per feature, etc. (Giraud-Carrier, 2008).

1.2 Aims and Objective

The research described in this thesis is closely related to INFER[1], a European project which aimed to develop a software platform for predictive modelling applicable in different industries and to work in the adaptive soft sensors for real-time prediction, monitoring, and control in the process industry. The goal of this work is to do research on MLL strategies and approaches for effective reduction of the model search space. There are multiple areas of a predictive system where MLL can be used to efficiently recommend the most appropriate methods and techniques. Therefore, three areas of evolving predictive systems are identified where the applicability of MLL can be an effective and efficient approach. These areas are thoroughly discussed in Section 2.7.

1. A Learning Path Recommendation: An optimal learning path recommendation of the three interlinked components including; pre-processing steps, learning algorithms or their combination, and adaptivity mechanism parameters.

2. Recurring Concepts Extraction: In a non-stationary environment, the underlying distribution of the incoming data keeps changing which in turn makes the most recent

[1] `http://infer.eu/`

historical concept ineffective. A MLL system can extract the relevant concepts of the data to adapt the out-dated model.

3. Concept Drift Detection: In an adaptive mechanism retraining of model is usually triggered by a change detection process. MLL can help to automatically detect the concept drift and trigger the algorithm retraining process instantly.

1.3 Research Challenges

There has been a lot of interest in MLL approaches and significant progress has been made. There are still a number of outstanding issues some of which have been addressed in the earlier approaches. The main challenge of this work is research on MLL strategies and approaches in context of; feature engineering for problem representation and learning strategy for algorithm recommendation. This problem leads to several research questions which are outlined as follows and discussed in detail in Section 2.7 along with the goals and objectives of this work.

1. Gathering examples of datasets to build a static Meta-knowledge database
2. Base-level Learning strategy to compute performance measures of Meta-examples
3. Feature generation and selection to represent a problem at Meta-level
4. Representation and storage of dynamically growing complex Meta-Knowledge database
5. Meta-level Learning strategy for algorithm and its hyper-parameter recommendation

From the above five research questions, 3 and 5 are addressed in this research.

1.4 Contributions

A thorough survey of the existing techniques has been performed aiming at giving a comprehensive overview of the research directions pursued under the umbrella of MLL. It reconciles different approaches given in scientific literature while designing the MLL systems. There are three studies conducted in this thesis around model selection and hyper-parameter search using MLL. These studies are addressing one or more research challenges which are described in the above section. The original contributions of this work are:

1. Formulation of Model Selection and Hyper-parameters Optimization (MSHPO) along with three key areas of an evolving predictive system which leads to several research challenges (see Section 2.7).
2. An MLL approach for evaluating the hypothesis whether the additional cross-domain training data can be beneficial to achieve reasonably good performance on a new task in the context of an MLL system for time-series forecasting. Chapter 3 illustrates it in detail.

3. An empirical study on the relationship between various characteristics describing the similarity of two tasks, and based on that, the amount of fine-tuning of a deep neural network required by a new task to achieve accuracy close to state-of-the-art. Further details can be found in Chapter 4.

4. An original approach for automatic hyper-parameter optimization of a Multi-component, Multi-level Predictive System (MCMLPS) which frames an efficient hyper-parameters selection and tuning as a reinforcement learning problem. Chapter 5 further elaborates this contribution.

5. A framework to automatically find a good set of hyper-parameters resulting in reasonably good accuracy, which at the same time is less computationally expensive than the existing approaches (see Chapter 5).

A significant part of the research presented in this thesis has appeared in the following publications:

1. Abbas Ali, Bogdan Gabrys and Marcin Budka. *Cross-domain Meta-learning for Time-series Forecasting.* In Procedia Computer Science, 126(1), 9-18, Elsevier, 2018.

2. Abbas Ali, Marcin Budka and Bogdan Gabrys. *Towards Meta-learning of Deep Architectures for Efficient Domain Adaptation.* In the 16^{th} Pacific RIM International Conference on Artificial Intelligence (PRICAI), 2019.

3. Abbas Ali, Marcin Budka and Bogdan Gabrys. *A Meta-Reinforcement Learning Approach to Optimize Parameters and Hyper-parameters Simultaneously.* In the 16^{th} Pacific RIM International Conference on Artificial Intelligence (PRICAI), 2019.

1.5 Organisation of the Thesis

The next chapter covers the existing research in Auto-ML area, including some important components of an MLL system. Those components include sources of existing and automatic generation of datasets, Meta-feature generation, and selection using various approaches and Base-level Learning (BLL) algorithms performance measures; such as accuracy, execution time, etc. This is followed by sections discussing existing MLL systems in the context of their applicability to supervised and unsupervised algorithms. Furthermore, Chapter 2 illustrates the adaptive mechanism and HPO areas in detail. Based on the conclusions and recommendations explored from the literature review, the final sections describe the research challenges and problem formulation of this research. An experimental investigation of cross-domain MLL for Time-series (TS) Forecasting is elaborated in Chapter 3. Chapter 4 consists of an empirical study to identify how deep a pre-trained image classifier needs to be fine-tuned based on the characteristics of the new task. Chapter 5 discusses a Meta-Reinforcement Learning (Meta-RL) approach to optimize the parameters and hyper-parameters tuning of DNN simultaneously. This report is concluded in Chapter 6 with future directions for the next phase.

Chapter 2

Existing Research

Immense research has been concentrating on automating Machine Learning (ML) algorithm selection for the last three decades (Zöller and Huber, 2019). The focus of those studies is to explore various components of Meta-level Learning (MLL). The scope of the literature review is confined to areas that are related to this research. The high-level overview of the components which are discussed in this chapter is shown in Figure 2.1. The first section presents ways of gathering real-world datasets and techniques to create synthetic datasets which are known as Examples of Datasets (EoD). These EoD are used to generate Meta-features (MFs) and associated performance measures which are discussed in Sections 2.2 and 2.3 respectively. MF are combined with performance measures to build Meta-knowledge (MK) dataset which becomes the input of MLL. The last section illustrates adaptive mechanisms in the context of MLL which are an important aspect of this research.

2.1 Repository of Datasets

A repository of datasets representing various problems is one of the key components of the MLL system. As Vanschoren (2011) states, 'there is no lack of experiments being done, but the datasets and information obtained often remain in the *people's heads and labs*'. This section explores the sources of real-world datasets that are used in the existing studies to build MK database. However, real-world datasets are usually hard to obtain but

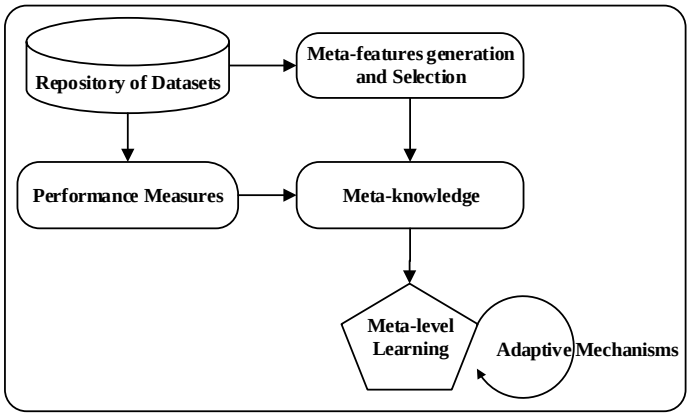

Figure 2.1: Scope of existing research review

artificially generated datasets would be a possible solution to this problem. The following subsections sumerize the studies that are dealing with real-world data, those which elaborate the techniques to generate artificial datasets, and the published resources.

2.1.1 Real-world Datasets

The real-world datasets can be difficult to find and gather in the desired format. An effort has been made to extract useful sources of data from various studies. Table 2.1 presents datasets that are used in different researches for MLL purpose. Most of them are gathered from UCI Machine Learning Repository (UCI) (Bache and Lichman, 2013).

Table 2.1: Real-world datasets used in various studies

Research Work	**Datasets**	**Sources**	**Dataset Filters**
King et al. (1995)	12	Satellite image, Hand-written digits, Karhunen-Loeve digits, Vehicle silhouettes, Segment data, Credit risk, Belgian data, Shuttle control, Diabetes, Heart disease, German credit, Head injury (King, 1995)	
Lindner and Studer (1999)	80	UCI and DaimlerChrysler	-
Sohn (1999)	19	Satellite image, Hand-written digits, Karhunen-Loeve digits, Vehicle silhouettes, Segment data, Credit risk, Belgian data, Shuttle control, Diabetes, Heart disease, German credit, Head injury (King, 1995) and 7 other datasets used in StatLog project	Three datasets of StatLog having cost information involved in misclassification
Berrer et al. (2000)	58	Meta-learning Assistant (METAL) project datasets	38 datasets with no missing values
Soares et al. (2001)	45	UCI and DaimlerChrysler	Dataset with more than 1000 instances
Bernstein and Provost (2001)	15	Balance Scale, Breast Cancer, Heart disease, Heart disease - compressed glyph visualization, German Credit Data, Diabetes, Vehicle silhouettes, Horse colic, Ionosphere, Vowel, Sonar, Anneal, Australian credit data, Sick, Segment data (Bache and Lichman, 2013)	-
Todorovski et al. (2002)	65	UCI and METAL project datasets	38 datasets with no missing values
Brazdil et al. (2003)	53	UCI and DaimlerChrysler	Datasets with more than 100 instances

Bernstein et al. (2005)	23	Balance Scale, Heart disease, Heart disease, Heart disease - compressed glyph visualization, German Credit Data, Diabetes, Vehicle silhouettes, Ionosphere, Vowel, Anneal, Australian credit data, Sick, Segment data, Robot Moves, DNA, Gene, Adult 10, Hypothyroid, Waveform, Page blocks, Optical digits, Insurance, Letter, Adult (Bache and Lichman, 2013)	-
Peng et al. (2002)	47	UCI	-
Kopf and Iglezakis (2002)	78	UCI	Dataset with less than 1066 instances and the number of attributes ranged from 4 to 69
Prudencio and Ludermir (2004)	I: 99 Time-series (TS) and II: 645	I: Time-series Data Library[1] and II: M3 competition[2]	I: Stationary data and II: Yearly data
Prudencio and Ludermir (2008)	50	WEKA project[3]	On average datasets contain 4,392 instances and 14 features
Wang et al. (2009)	46 and 5	Time Series Data-mining Archive[4] and Time Series Data Library[5]	
Kadlec and Gabrys (2009)	3	Thermal oxidiser, Industry drier, and Catalyst activation datasets of process industry	On-line prediction datasets
Lemke and Gabrys (2010a)	2 consisting of 111 TS	NN3[6] - Monthly business with 52-126 observations and NN5[6]- daily cash machine withdrawals with 735 observations in each series	NN5 including some missing values
Abdelmessih et al. (2010)	90	UCI	Datasets with more than 100 instances
Duch et al. (2011)	5 and 2	Leukemia, Heart, Wisconsin, Spam, and Ionosphere are real-world datasets gathered from UCI and two synthetic datasets parity and monks	

[1] `http://datamarket.com/data/list/?q=provider:tsdl`
[2] `http://forecasters.org/resources/time-series-data/m3-competition`
[3] Machine Learning Group at University of Waikato `http://www.cs.waikato.ac.nz/ml/weka`
[4] `http://www.cs.ucr.edu/~eamonn/time_series_data`
[5] `http://datamarket.com/data/list/?q=provider:tsdl`
[6] Neural Network forecasting competition

Rossi et al. (2014)	2	Travel Time Prediction (TTP) consists of 24,975 instances and Electricity Demand Prediction (EDP) consists of 27,888 instances	
Feurer et al. (2014)	57	Open Machine Learning (OpenML) datasets	-
Kuhn et al. (2018)	38	OpenML datasets	The datasets have no missing values and a binary outcome
Ali et al. (2018)	2 consisting of 111 TS	NN3 and NN5	NN5 including some missing values

Warden (2011) handbook and Stanford and Iriondo (2018) cover the most useful sources of publicly available datasets. A lot of new sources of free and public data that have emerged over the last few years are discussed. Apart from discussing data-sources, methods to get datasets in bulk from those sources are also discussed in detail. Table 2.2 presents most of the sources from the author's book.

Table 2.2: List of publicly available Data Repositories

Source	**Description**	**Datasets**	**Domain**
AnalcatData	Datasets that are used by Jeffrey S. Simonoff in his book *Analyzing Categorical Data*, published in July 2003	83	Cross-domain
Amazon Web Services	A centralized repository of public datasets		Astronomy, Biology, Chemistry, Climate, Economics, Geographic and Mathematics
Bioassay data	Virtual screening of bioassay (active/inactive compounds) data by Amanda Schierz	21	Life Sciences
Canada Open Data	Canadian government and geospatial data		Government & Geospatial
Datacatalogs	List of the most comprehensive open data catalogs		
data.gov.uk	Data of UK central government departments, other public sector bodies and local authorities	9616	Government and Public Sector
data.london.gov.uk	Data of UK central government departments, other public sector bodies and local authorities	563	Government and Public sector
Data.gov/ Education	Educational high-value datasets	70,897	Cross-domain

ELENA	Non-stationary streaming data of flight arrival and departure details for all the commercial flights within the USA	13 features and 116 million instances	Aviation
KDD Cup	Annual Data Mining and Knowledge Discovery competition datasets		Cross-domain
National Government Statistical Web Sites			
Open Data Census US Census Bureau	Assesses the state of open data around the world		Government and Public sector
OpenData from Socrata	Freely available datasets	10,000	Business, Education, Government, Social and Entertainment
Open Source Sports	Many sports databases, including Baseball, Football, Basketball and Hockey		Entertainment
UCI	A collection of databases, domain theories, and data generators that are used by the ML community for the empirical analysis of learning algorithms	199	Physical Sciences, Computer Science & Engineering, Social Sciences, Business and Game
Yahoo Sandbox datasets	Language, graph, ratings, advertising and marketing, competition, computing systems and image datasets	-	Cross-domain
OpenML	From 100 OpenML classification datasets, 38 datasets without missing values and with a binary outcome have been used	38	Cross-domain

2.1.2 Synthetic Datasets

MFs are used as predictors in an MLL system. Typically, many MFs are extracted from a dataset, thereby leading to a high-dimensional sparsely populated feature space which has always been a challenge for learning algorithms. Hence, to overcome this problem sufficient number of datasets is required which may not be possible only from real-world datasets as they can be hard to obtain. So, artificially generated datasets might help in solving this issue. Rendell and Cho (1990) work on systematic artificial data generation is considered as one of the initial efforts in this regard.

Bensusan and Giraud-Carrier (2000) used 320 artificially generated boolean datasets with 5 to 12 features in each one. These artificial datasets were benchmarked on 16 UCI and DaimlerChrysler real-world datasets. Similarly Pfahringer et al. (2000) generated 222 datasets, each containing 20 numeric and nominal features having 1K to 10K instances classified between 2 to 5 classes. Additionally, 18 real-world UCI problems were used to evaluate the proposed approach.

Soares (2009) proposed a method to generate a large number of datasets by transforming the existing datasets, known as *datasetoids*. An artificial dataset was generated against each symbolic attribute of a given dataset, obtained by switching the selected attribute with the target variable. This method was experimented on 64 datasets gathered from the UCI repository and it generated a total of 983 *datasetoids*. At the end, potential anomalies related to the artificial datasets are also discussed along with their proposed solutions are presented. Those anomalies could be: 1) the new target variable has missing values, 2) it is very skewed, and/or 3) the corresponding target variable might be completely unrelated to remaining features. One very simple solution proposed for these problems was to simply discard the *datasetoids* which showed any of the above mentioned properties. This method produced promising results, therefore enabling the generation of new datasets which could solve the scarce datasets problems.

Wang et al. (2009) used both synthetic and real-world TS from diverse domains for MLL based forecasting method selection study. The details of real-world datasets are given in Table 2.1 while remaining synthetic datasets were generated using statistical simulation to facilitate the detailed analysis of forecasting association with data characteristics. A total of 264 artificial datasets posses certain characteristics, i.e., perfect and strong trend, perfect seasonality, noise. The data is transformed into samples of 1000 instances for each original TS while it is unchanged for the number of data-points smaller than 1000.

Soares et al. (2009) generated 160 artificial datasets to obtain a wide representative range of cluster structures. There were two methods used to generate datasets; 1) a standard cluster model using Gaussian multi-variate normal distributions, and 2) Ellipsoid cluster generator. There were three parameters selected for both techniques including i) the number of clusters which were the same for both cases (2, 4, 8, 16), ii) dimensions (2, 20 for Gaussian, and 50, 100 for Ellipsoid), and iii) the size of each cluster for both techniques were the same (uniformity in [10, 100] for 2 and 4 clusters case and [5, 50] for 8 and 16 clusters case). For each of the 8 combinations of cluster number and dimension, 10 different instances were generated, giving 80 datasets in each method.

Duch et al. (2011) used two artificially generated datasets out of a total of seven whereas the remaining five are real-world problems. One artificially generated dataset has binary features, named as *Parity*, whereas the other one with nominal features is known as *Monks*. These optimal support features are computed using Quality of Projected Clusters (QPC) projection (Grochowski et al., 2008).

Reif et al. (2012a) presents a novel data generator approach for numerical features and classification datasets that can be used as input dataset for MLL; i.e. an entirely different approach from the Soares (2009). The proposed system was able to generate datasets using genetic programming with customized parameters. Also in this setting MLL can be supported in two different ways: 1) the MFs space can be filled in a more controlled way and the discovered "empty areas" can be populated rather than generating random datasets, and 2) thoroughly investigating MFs based on their descriptive power which can be useful for certain MLL problems and generating datasets with MFs allows more controlled experiments that might lead to the significant utilization of particular MFs. Since the dataset was generating multiple MFs therefore this task was treated as a multi-objective optimization problem. The proposed system was able to incorporate a variable set of arbitrary MFs. The

user was able to build a custom set of MFs simply by providing the functions that compute the MFs.

Feurer et al. (2014) obtained 57 datasets, from the OpenML project (Rijn et al., 2013), to study the impact of their MLL based initialization of Sequential Model-Based Optimization (SMBO) variants. Lemke and Gabrys (2010a) and Ali et al. (2018) have used 222 univariate time-series from two different sources, NN3 (Crone, 2006) and NN5 (Crone, 2008) competitions. Each data-source contains 111 series with monthly and daily occurrences respectively. Kuhn et al. (2018) chose a subset from the OpenML classification datasets. The authors used the datasets without missing values and with a binary class.

2.1.3 Datasets from Published Research

Another source of EoD are the published ML studies. As ML is one of the most active research areas since last few decades where several experimentations have been conducted. These experiments become a very useful way of gathering EoD representing various domains. Also, the additional factor that usually comes with most of the datasets used in existing researches is performance measures. It is used as target variable in the context of an MLL system. It is very time, memory and processor consuming task to compute performance measures against a massive amount of datasets and numerous configurations of learning algorithms.

Due to space limitation on publications, researches usually publish only final results with minimal details. However, in context of MLL, relying on this minimal information leads to several problems, for example, in most of the instances researches only report the best algorithm, usually report limited number and detail of experimentations, mostly skip detailed configurations of the algorithms, etc. Vanschoren et al. (2014) introduced a novel platform for ML research known as OpenML. ML researchers can share datasets, algorithms, their configurations, and experiment setups on this platform which other researchers can use to compare results. OpenML framework is one of the possible solutions for most of the mentioned concerns which resolves two key challenges of MLL systems; i) gathering a massive number of datasets from different domains, and ii) performances of datasets.

2.1.4 Discussion and Summary

An ML system relies on training dataset to build a model. Similarly, at Meta-level, the MK dataset is used as training-set of MLL, whereas this MK dataset is dependent on sufficient number of EoD from different domains. These EoD are used to generate MFs which act as predictors and performance measures of these EoD on various learning algorithms and are used as target variable in the MK dataset. However, gathering a sufficient number of real-world datasets is quite difficult. The real-world datasets which are used in various studies for experimentations are listed in Table 2.1. Most of the studies gathered datasets from UCI with different filtering options and the remaining few studies gathered datasets from different data-mining competitions. In most cases, the number of EoD that are used to build MK is very low. The MLL systems can perform as better as trial-and-error approach by providing a significant number of EoD representing various domains. Table 2.2 identifies a number of real-world datasets representing different domains.

Some MLL researches resolved this problem by building their MK datasets using artificially generated EoD. They have adopted two different approaches to generate synthetic datasets; 1) by transforming real-world datasets; and 2) by utilizing statistical and genetic programming approaches. Bensusan and Giraud-Carrier (2000), Pfahringer et al. (2000), Soares (2009) and Wang et al. (2009) proposed different feature transformation approaches to generate different combinations of datasets from the limited number of real-world datasets. The statistical and genetic programming approaches are proposed by Soares et al. (2009) and Duch et al. (2011) for MLL systems. In some approaches, statistical functions with threshold (or cut-off) values are used to generate data while others used optimization techniques. Reif et al. (2012a) proposed an intelligent technique which does not generate random data, but fill the MFs in a more controlled way by discovering and populating the empty areas of real-world datasets.

Combining all the proposed approaches iteratively seems to be a potential solution of dataset scarcity; i.e., initially gathering the existing available real-world problems, then transforming those datasets generating several others and finally applying various other techniques to generate artificial datasets independently (see Figure 2.2). Although this solution seems useful if the purpose is only gathering a massive number of EoD. But in the context of this research, the purpose of gathering these datasets is twofold; i) to generate MFs and ii) compute performance measures of these datasets against numerous learning algorithms and their configurations. Considering all three components of an MLL system, gathering datasets from published research seems more convincing where the performance measures are bundled with the EoD. However, there are a lot of challenges coupled with it for an MLL system which include reporting only the top performing learning algorithm, publishing limited information of experimentations, availability of datasets used in the research, detailed configurations of learning algorithm, etc. OpenML platform addresses most of these issues to some extent but it is in the preliminary stage which leads to several issues, for example, i) problem representation at Meta-level is covering very few domains, ii) most of the publications are using very few commonly used learning algorithms, etc.

Deep Learning (DL) is one of the recent advancement in ML which brought a paradigm shift in MLL (Minar and Naher, 2018). This shift has minimized the dependency of the MLL systems on a large repository of datasets (Hutter et al., 2018; Zöller and Huber, 2019; Yao et al., 2019).

2.2 Meta-features Generation and Selection

One of the primary applications of MLL is to recommend the best learning algorithm or to rank various algorithms against a problem that is further described by some new features, known as MFs. The role of such systems is to estimate the similarity between various problems which, in turn, requires the ability to estimate the similarity of new data with the already analysed datasets. There are three most commonly used MF generation approaches which allow to induce mapping between characteristics of a problem and learning algorithms. These approaches are discussed in the following sections.

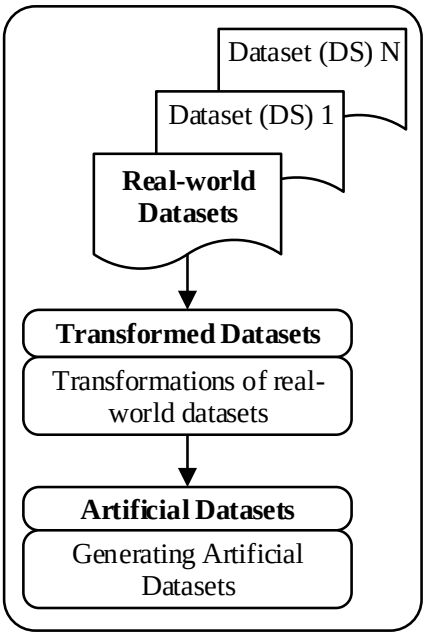

Figure 2.2: Phase-wise collection of Examples of Datasets

2.2.1 Descriptive, Statistical and Information-Theoretic Approach

The Descriptive, Statistical, and Information-Theoretic (DSIT) approach is the simplest and most commonly used MF generation approach that extracts a number of DSIT measures of a problem. These measures are used to map the features of the problem to the algorithms (Giraud-Carrier, 2008). It is also supported by the empirical results that these simple characteristics, such as the size of the training set and input space play a vital role in differentiating suitability of various learning algorithms to solve such problems. The research works that have been performed using DSIT approach are reviewed below.

Rendell et al. (1987) proposed Variable-bias Management System (VBMS) that was one of the earliest efforts towards data characterization. Only two descriptive MFs, namely: the number of training instances and the number of features, were used to select the best among three symbolic learning algorithms. Later Rendell and Cho (1990) enhanced the existing system by adding useful MFs of complexity based on shape, size, and structure. Statistical and Logical learning (StatLog) project by King et al. (1995) further extended VBMS MFs by considering a larger number of dataset characteristics. A problem was described in the context of its descriptive and statistical properties. Several characteristics of a problem spanning from simple (descriptive) to more complex (statistical) ones were generated and later used by various studies. These characteristics were used to investigate why certain algorithms perform better on a particular problem as well as to produce a thorough empirical analysis of the learning algorithms.

Sohn (1999) initially used most of the datasets and MFs that were used in StatLog project which were later on enhanced with information-theoretic MFs. Furthermore, three new descriptive features were added by transforming the existing measures, for example, in the form of ratios. These MFs were used to rank several classification algorithms with considerably better performance as compared to the previous studies. The author has also

claimed classification error and execution-time as important response variables to choose an appropriate classification algorithm for a problem.

In the same year Lindner and Studer (1999) proposed an extensive list of DSIT features of a problem under the name of Dataset Characterization Tool (DCT). The authors distinguished three categories of dataset characteristics; namely simple, statistical and information-theory based measures. The descriptive MFs have been used to extract general characteristics of the dataset, whereas statistical characteristics were mainly extracted from numeric attributes, while information-theoretic based measures from nominal attributes. As in StatLog, rules were generated to choose an algorithm for a given task. Having this in mind authors were motivated to propose Case-based Reasoning (CBR) approach to select the most suitable algorithm against a problem.

Reif et al. (2012b) presented a novel approach of generating informative MFs by simply averaging over all attributes of the source datasets. They proposed a two-fold approach; in the first fold MFs generate the DSIT features of the datasets using the traditional approach. The second fold describes differences over datasets that are not accessible using the typically used mean of Meta-measures that have been computed in the first fold. This approach preserves more information about such MFs while producing a feature vector with a fixed size. An additional level of features are extracted to automatically select the most useful features out of the available ones.

MFs that are used in the above studies are shown in Figure 2.3.

2.2.2 Landmarking Approach

Another technique of MF generation is Landmarking which characterizes a dataset using the performance of a set of simple learners. Its main goal is to identify areas in the input space where each of the simple learners can be regarded as an expert (Vilalta and Drissi, 2002a).

The basic idea behind landmarking is outlined as the performance of a learning algorithm on a task and discovering information about its nature. In this approach, the performance of a Base-learner on a problem reveals information about the nature of the problem. A landmark learner or landmarker is defined as the learning mechanism whose performance is used to describe a problem (Bensusan and Giraud-Carrier, 2000). Landmarkers posses a key property that their execution time is always shorter than the Base-learner's time, otherwise, this approach would bring no benefit. Further, in this section, various studies dealing with Landmarking approach are discussed in detail.

One of the earliest studies on Landmarking was conducted by Bensusan and Giraud-Carrier (2000). This approach is claimed to be simpler, more intuitive and effective than the DSIT measures. A set of 7 landmarkers were trained on 10 different sets of equal size. Each dataset was then described by a vector of MFs (see Landmarkers branch of Figure 2.3), which are the error rates of the 7 landmarkers, and labelled by the target learners (see Landmarking section of Appendix B) which produce the highest accuracy. Several experimentations have been performed to compare the landmarking approach with DSIT. In the first experiment Landmarking was compared with 6 information-theoretic DCT features of Lindner and Studer (1999) (see information-theoretic MFs section of Figure 2.3).

In most of the cases of this experiment landmarking simply outperformed DSIT approach. In another experiment, the ability of landmarking to describe a problem and discriminate between two areas of expertise are highlighted. In most of the cases C5.0 Adaptive Boosting (C5.0 boost) (Quinlan, 1998) landmarker performed best. The last experiment benchmarked 16 real-world datasets from UCI (Bache and Lichman, 2013) and DaimlerChrysler where again landmarking approach has produced the overall best performance.

Pfahringer et al. (2000) also presented Landmarking while comparing it with the DSIT MF generation approach - DCT. They performed three types of experiments, namely 1) Artificial rule list and sets; 2) Selecting learning models, and 3) Comparing landmarking with information-theoretic approach. These experiments were almost the same as performed by Bensusan and Giraud-Carrier (2000), and the target learners (see Landmarking section of Appendix B) were the same as used in METAL project. In the first experiment, the set of landmarkers consisted of a Linear Discriminant Analysis (LDA), Naive Bayes classifier (NB), and C5.0 Decision Tree (C5.0 tree) learner. While base-learners performance relative to each other was predicted using C5.0 boost, LDA, and Rule Learner (Ripper). In addition to three landmarkers, 5 descriptive MFs (shown in descriptive approach of Figure 2.3) have also been extracted from 216 datasets. The Ripper was found to be the top performer in this experimentation. For selecting the best learning model experiment, the authors tried to investigate the capability of landmarking in deciding whether a learner involving multiple learning algorithms performs better than the other candidate algorithms. Here only C4.5 Decision Tree algorithm (C4.5) was used as Meta-learner trained with 222 artificial boolean datasets and tested with 18 UCI problems (Bache and Lichman, 2013). Even though the Landmarking accuracy was higher but it does not reflect on the overall performance of a system whose end goal is to accurately select a learning model. In the last experiment, the landmarking approach has been compared with the DSIT and also the combination of both approaches. 320 artificially generated binary datasets were produced where the combined approach performed best for all 10 Meta-learners followed by Landmarking with a significant difference as compared to DCT measure.

Soares et al. (2001) sample-based landmarkers were estimates of the performance of algorithms on a small sample of the data that had been used as predictors of the performance of those algorithms on the entire dataset. Additionally, relative landmarker addressed the inability of earlier landmarker to assess the relative performance of algorithms. This sampling-based relative landmarking approach was later compared with the DSIT DCT MFs (Lindner and Studer, 1999) as done by most of the landmarking studies. The ten algorithms, mentioned in Appendix B are used on 45 datasets, with more than 1000 instances, mostly gathered from UCI (Bache and Lichman, 2013) and DaimlerChrysler. These datasets have been ranked by the *Nearest-Neighbour* using Adjusted Ratio of Ratios (ARR) measure. To observe the performance of the ranking method authors vary the value of k from 1 to 25. In comparison with other studies reported in the literature, a sample-based relative landmarking approach has shown improvements in algorithm ranking as compared with the traditional DCT measures.

Kopf and Iglezakis (2002) proposed a new approach of task description for model selection in context of MLL. It evaluates the performance for assessing the quality standards for case-bases when used for supervised MLL. The case-base properties were used to assess the

quality of given case-bases in terms of measures such as redundancy. A brief overview of necessary requirements for the implementation of the case-based properties has also been provided in their study. A comprehensive experimentation was performed to compare variants of DCT DSIT approach, landmarking and their combinations. MFs were constructed for the experiments from UCI datasets (see Table 2.1) which contained up to 25% missing values. Error rates for ten different classification algorithms from the METAL project were determined for different subsets of data characteristics mentioned in Appendix B and restricted to three Base-learners that are shown in Figure 2.3. The empirical results show the proposed approach in combination with DSIT, and landmarking approaches as a promising one.

Abdelmessih et al. (2010) presented an overview of the RapidMiner's Landmarking operator and its evaluation. This landmarking operator was developed in an open-source data-mining tool RapidMiner. As mentioned repeatedly in the above studies, landmarkers selection is a critical process and the criteria to select an optimal landmarker consists of three characteristics: 1) a landmarker has to be simple, 2) it require minimum execution (processing) time and 3) it has to be simpler than the target learner(s). Following these conditions, RapidMiner provided the landmarkers shown in Figure 2.3 and target algorithms, for which the accuracy was predicted (see landmarking section of Appendix B). For the evaluation of these landmarkers, 90 datasets from the UCI (Bache and Lichman, 2013) and other sources are gathered with at least 100 samples in each. By following the existing studies, the landmarking operator has been compared with the DSIT MFs of StatLog (King et al., 1995) and DCT (Lindner and Studer, 1999), where Landmarking has shown *5.1-8.3%* overall boost in all cases.

2.2.3 Model-based Approach

Model-based MF generation is another effort towards task characterization in MLL domain. In this approach the dataset is represented in a data structure that can incorporate the complexity and performance of the induced hypothesis. Later the representation can serve as a basis to explain the reasons behind the performance of the learning algorithm Giraud-Carrier (2008). Several research works utilizing the Model-based approach are discussed as follows.

Bensusan et al. (2000) study was an initial effort towards the model-based approach. The authors proposed to capture the information directly from the induced decision trees for characterizing the learning complexity. Figure 2.3 lists the 10 descriptors computed from induced decision trees. Using these MFs, a task representation and algorithm to store and compare two different tree structures has been explained in detail with examples. Authors also elaborated the motivation of using the induced decision trees directly rather than the predefined properties used in decision tree based MFs that, in turn, made explicit properties implicit in the tree structure. Finally, higher-order MLL approach has been generalized by proposing data structures to characterize other algorithms. Tree-like structure was used for Decision Trees (DT) in this work, sets have been proposed for *rule sets* and graphs for Neural Networks (NNs).

Peng et al. (2002) effort was towards improving the dataset characterization by capturing structural shape and size of the decision tree induced from the dataset. For that purpose 15 features were proposed known as *DecT*, shown in Figure 2.3, which do not overlap with Bensusan et al. (2000). These measures have been used to rank 10 learning algorithms in various experiments. In the first experiment DCT (Lindner and Studer, 1999) DSIT MFs and 5 landmarkers (Worst Nodes Learner, Average Nodes Learner, NB, and LDA) were compared with DecT. The results proved the performance enhancement of the proposed approach. In another experiment, DecT measures have been compared with the same DCT measures and landmarkers to rank the learning algorithms based on accuracy and time where again DecT performed better. The last experiment was performed to select MFs by reducing the number of features to 25, 15 and 8 respectively. The k-Nearest Neighbour algorithm, with various values of k between 1 to 40, was used to select k datasets for ranking the performance of learning algorithms. The results suggested that feature selection does not significantly influence the performance of either DecT or even DCT, furthermore, DecT outperformed other approaches.

The Neuro-cognitive inspired mechanism is proposed by Duch et al. (2011) to analyse learning based transformations that generate useful hidden features for MLL. The types of transformations include restricted random projections, optimization using projection pursuit methods, similarity and general kernel-based features, conditionally defined features, and features derived from partial successes of various learning algorithms. The binary features were extracted from DT and rule-based algorithms where continuous features were discovered by projection pursuit, linear Support Vector Machines (SVM) and simple projections. NB provides posterior probabilities along these lines while k-Nearest Neighbour (k-NN) and kernel methods find similarity-based features. The proposed approach illustrates Multi-dimensional Scaling (MDS) mappings and Principal Component Analysis (PCA), Independent Component Analysis (ICA), QPC, SVM projections in the original, one- and two-dimensional space. Various real-world and synthetic datasets (details can be found in Table 2.1) were used for visualization and to analyse the kind of structures they create. The classification accuracies of the datasets are predicted using five classifiers including NB, k-NN, Separability Split Value Tree (SSV), Linear and Gaussian kernel SVM in the original, one- and two-dimensional spaces. The results show an significant improvement almost in all five algorithms as compared to the existing approach of the authors.

2.2.4 Discussion and Summary

There are three common MF generation approaches proposed in the reviewed publications for MLL: 1) DSIT, 2) Landmarking and 3) Model-based. The DSIT MFs approach was introduced at the early stage of MLL development where Rendell et al. (1987) proposed two descriptive features for VBMS. Later on Rendell and Cho (1990) added more descriptive features to the original list. The statistical MFs were introduced by King et al. (1995), and Sohn (1999) proposed information-theoretic features combined with some existing descriptives to represent a problem at Meta-level. Finally, Lindner and Studer (1999) proposed an extensive list of DSIT MFs, known as DCT. The DCT measures became a benchmarked approach to represent a problem using DSIT approach. These measures were later used in

several studies for experimentation, e.g. Berrer et al. (2000), Giraud-Carrier (2005), etc., and compared with other MF approaches.

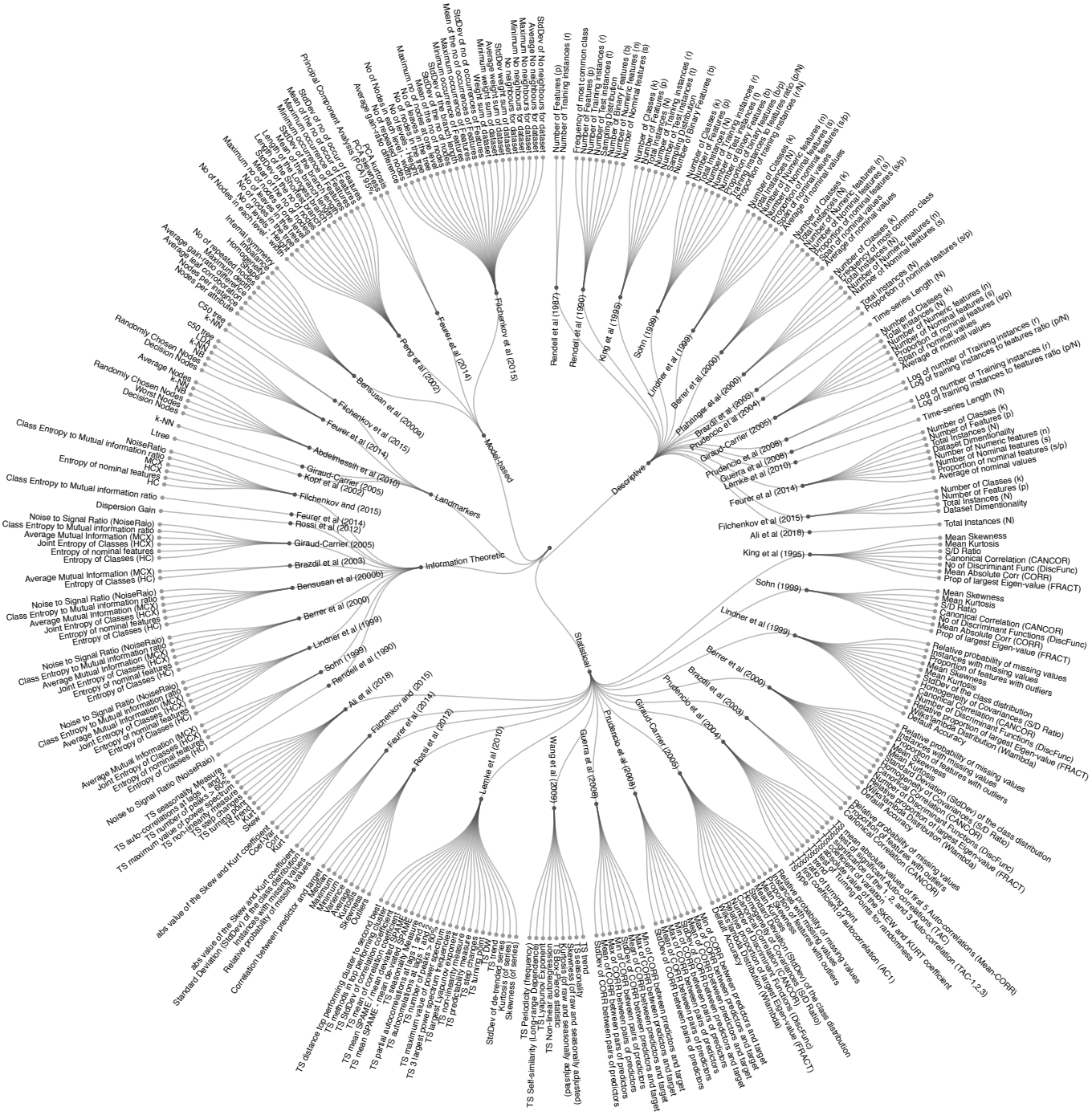

Figure 2.3: Meta-features used in various studies[7]

Landmarking and Model-based approaches are more recent ones and have been outperforming the DSIT in almost all the comparative studies. The earliest study on Landmarking was conducted by Bensusan and Giraud-Carrier (2000) where the approach was claimed to be simpler, more intuitive and efficient than DSIT. The proposed approach was compared

[7]Tabular representation of the visualization can be seen in Appendix B)

with and outperformed information-theoretic measures of DCT with a significant difference. Though one common deficiency that is observed in several MLL studies is the use of a smaller number of EoD for experimentations which raised a question on the significance of the reported results. Pfahringer et al. (2000) used a different set of Landmarkers but the same target learners as Bensusan and Giraud-Carrier (2000). This work can be considered to offer improvements to the previous one in two aspects: 1) a large number of synthetic datasets were used; and 2) some descriptive MFs were combined with the Landmarkers. This approach was also compared with DCT features where Landmarking showed significant improvement in the results. Similarly Soares et al. (2001), Kopf and Iglezakis (2002) and Abdelmessih et al. (2010) have used different sets of target learners, landmarkers, number of dataset examples and compared their approach with a different set of DSIT measures. All of them reported improved results of Landmarking approach over DSIT.

Bensusan et al. (2000) approach to characterizing the learning complexity by directly inducing from the model is the earliest work towards the model-based approach. In this work, 10 descriptors were computed from the induced decision trees which can be seen in Figure 2.3. Peng et al. (2002) effort was towards improving this characterization by focusing on structural shape and size of the decision tree induced from the datasets. The other dimension of this work was to compare the proposed model-based approach with DCT, DSIT and Landmarking measures. Various experimentations have been performed with variations of MFs and Landmarkers where the model-based approach consistently performed better. A problem with these Meta-level problem representations is that they can not facilitate the non-stationary environment. Most of the effort has been dedicated to the stationary environment, even though some recent studies are addressing MFs for a dynamic environment, i.e. Rossi et al. (2014), but these are not mature enough to represent the entire domain. Although Rossi et al. (2014) used traditional MF that are used to characterize stationary data, only those MFs were computed that characterize individual variables. Moreover, there are separate features computed for training and selecting windows. Their reliability is associated with the number of examples, thus the larger the number of examples in a window, the higher the reliability of problem representation at Meta-level. However, in a rapidly changing environment, a limited number of examples accumulate between consecutive concept changes. Hence there is an unaddressed need for useful MFs calculated from small data.

Lemke and Gabrys (2010a) and Ali et al. (2018) have used three different groups of MF extracted from univariate time-series, including descriptive statistics, frequency domain and auto-correlation features. Feurer et al. (2014) has obtained 46 MF from five different groups including simple, DSIT, and PCA. Filchenkov and Pendryak (2015) has also gathered a comprehensive set of DSIT features for classification task. Also model-based MF computed from DT, k-NN, and perceptron are combined with DSIT.

From the above studies, it can be observed that combining significant MFs from different feature generation approaches might be useful as shown in Figure 2.4.

The recent paradigm shift of MLL brought a different set of model-based MF (Minar and Naher, 2018). Ali et al. (2019b) introduces the Probability Mass Function (PMF) of the final layer of Deep Neural Networks (DNN); another variation is Probability Density Function (PDF) of the network's activations. In another study on Meta-Reinforcement Learning

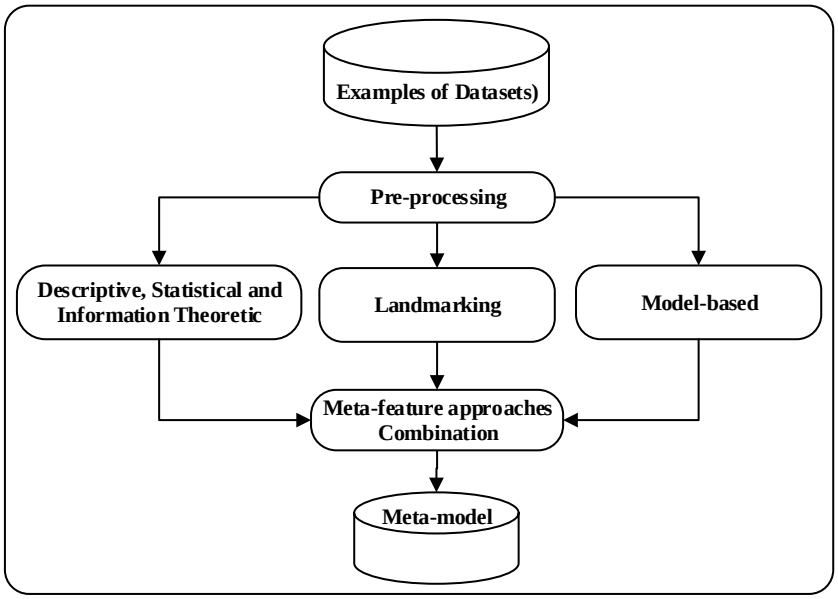

Figure 2.4: Combining Significant Meta-features from various approaches

(Meta-RL) (Ali et al., 2019a), a DNN is used as Base-learner which sits in the environment. Moreover, the behaviour of the environment (state) is represented as MF which is composed of the training loss, accuracy, and PMF of the network's final layer outcome.

2.3 Base-level Learning

In the context of MLL, Base-level Learning (BLL) algorithms are used to build predictive models on input datasets and for MLL purposes are used to compute a set of performance measures, i.e, accuracy, execution-time, etc. These performance measures are combined together with their respective MFs in MK database. A Meta-learner uses these performances as a target variable. The remaining sections discuss several studies relevant to individual and combination of BLL algorithm techniques. Moreover, the combination of Base-learners uses multiple individual models to achieve overall boosted performances which are mapped with their respective MFs.

Brazdil et al. (2003) proposed an MLL based approach to rank candidate algorithms where k-NN was used to identify the datasets that were most similar to the query dataset. The pool of candidate algorithms contained an ensemble method, namely C5.0 boost, which performed well for 19 out of 53 datasets in the presence of 9 other algorithms. The performance of ensemble methods was ranked with individual learning algorithms. In general, several researches used C5.0 boost ensemble method with other individual algorithms and found it as the top performing method in the ranking list.

The applicability of MLL on TS task is demonstrated by Lemke and Gabrys (2010a). Several individual and combination of forecasting algorithms were used to investigate which model works best in which situation. In the experiments, five forecasting combination methods have been used including 1) *simple average* where all available forecasts are averaged, 2) *simple average with trimming* which do not take the worst performing 20% models into account, 3) *variance-based method* where weights for a linear combination of forecasts are

determined using past forecasting performance, 4) *out-performance method* which determines weights based on the number of times a method performed best in the past, and 5) *variance-based pooling* which first groups past forecast performance into 2-3 clusters and then takes their average to obtain final forecast. The results of these experiments show that the forecast combination methods perform better than individual model selection which are listed in Table 2.3. Further discussion of this work can be found in Section 2.4.

Menahem et al. (2011) proposed a new MLL based ensemble scheme for one-class problems know as TUPSO. The TUPSO combine one-class Base-classifiers via Meta-classifier to produce a single prediction. The BLL component generates predictions of classifiers which are used to extract aggregated MFs as well as one-class accuracy and f-score estimates. The one-class performance evaluator computed each Base-classifier on only positive labelled instance using 4 algorithms including 1) global density estimation, 2) peer group analysis, 3) SVM, and 4) attribute distribution function approximation (ADIFA) on 53 distinct datasets (details can be seen in Table 2.1). There are 15 aggregated MFs computed from the predictions of Base-classifiers that are clustered into four groups: 1) summation-based (votes, predictions, weighted predictions, power and log of weighted predictions), 2) variance-based (votes, predictions, and weighted), 3) histogram-based, and 4) representation-length based. In empirical evaluation, an ensemble combining method, Fixed-rule, produced worse classification accuracy when compared to MLL based ensembles - TUPSO. Filchenkov and Pendryak (2015) has used five different classifiers to avoid bias and leave-one-out cross-validation to estimate the performance. Ali et al. (2018) has used both simple and complex forecasting methods which are originally introduced by Lemke and Gabrys (2010a).

Table 2.3: Base-level learning strategies used in different studies

Research Work	**Sampling Strategy**	**Base-learners**	**Performance Measure**
King et al. (1995)	9-fold Cross-Validation (CV) for datasets with less than 2500 instances	k-NN, Radial-basis Function (RBF), Density Estimation, Classification and Regression Trees (CART), Inductive CART (INDCART), Back-propagation, NewID, C4.5, CN2 Induction Algorithm (CN2), Quadratic Classifier (Quadra), Cal5, AC^2, Smooth Multiple Additive Regression Technique (SMART), Logistic Regression, Fisher's Linear Discriminant (FLD), ITrule, Causal Structure for Inductive Learning (CASTLE), NB	Misclassification error, Run-time speed
Bensusan and Giraud-Carrier (2000)	stratified 10-fold CV	NB, Multi-layer Perceptron (MLP), RBF, C5.0 tree, C5.0 Rule Induction (C5.0 rules), C5.0 boost, Instance-based Learning (IBL), LDA, Ripper, Linear Discriminant Trees (Ltree)	
Pfahringer et al. (2000)	10-fold CV	NB, MLP, RBF, C5.0 tree, C5.0 rules, C5.0 boost, IBL, LDA, Ripper, Ltree	Mean Absolute Error (MAE)
Soares et al. (2001)		NB, MLP, RBF, C5.0 tree, C5.0 rules, C5.0 boost, IBL, LDA, Ripper, Ltree	

Peng et al. (2002)	10-fold CV	C5.0 tree, C5.0 rules, C5.0 boost, Ltree, LDA, NB, IBL, MLP, RBF, Ripper	Mean Squared Error (MSE), Run-time speed
Todorovski et al. (2002)	10-fold CV	C5.0 tree, C5.0 rules, C5.0 boost, Ltree, Ripper, NB, k-NN [8], LDA	MSE and Spearman's Rank Correlation Coefficient (SRCC)
Kopf and Iglezakis (2002)	10-fold CV	NB, MLP, RBF, C5.0 tree, C5.0 rules, C5.0 boost, IBL, LDA, Ripper, Ltree	
Brazdil et al. (2003)	10-fold CV	C5.0 tree, C5.0 rules, C5.0 boost, Ltree, IBL, Ripper, LDA, NB, MLP, RBF	ARR
Prudencio and Ludermir (2004)	I: Train and test and II: train, test and validate	I: J.48 and II: MLP	MAE
Giraud-Carrier (2005)	10-fold CV	NB, MLP, RBF, C5.0 tree, C5.0 rules, C5.0 boost, IBL, LDA, Ripper, Ltree	
Guerra et al. (2008)	10-fold CV	MLP[8]	Normalized MSE
Wang et al. (2009)	80% Training and 20% testing partition	Exponential Smoothing (ES), Autoregressive Integrated Moving Average (ARIMA), Random Walk (RW), NN	
Kadlec and Gabrys (2009)	Leave-one-out CV	Multiple Linear Regression (MLR), MLP, RBF, Lazy-learning	MSE and SRCC
Lemke and Gabrys (2010a)	10-fold CV	ARIMA, Structural model, Iterated (single exponential smoothing, Taylor smoothing, theta, NN, elman NN), Direct (regression, theta Moving Average (MA), single exponential smoothing, Taylor smoothing, NN)	Symmetric Mean Absolute Percentage Error (SMAPE)
Abdelmessih et al. (2010)	10-fold CV	NB, k-NN, MLP, C5.0 tree, Random Forests (RF), One Rule Learner (OneR), SVM	Root Mean Squared Error (RMSE)
Rossi et al. (2012)	Training and testing	RF, SVM, CART, Projection Pursuit Regression (PPR)	Normalized MSE
Rossi et al. (2014)	Training and testing	RF, SVM, CART, PPR, Multivariate Adaptive Regression Splines (MARS)	Normalized MSE
Filchenkov and Pendryak (2015)	Leave-one-out CV	C4.5, PART (Frank and Witten, 1998), NB, BayesNet, IB3 Aha et al. (1991)	RPR (Filchenkov and Pendryak, 2015)
Ali et al. (2018)	Leave-one-out CV	MA, ARIMA, Structural model, NN	SMAPE

[8]k=1

[8]hidden nodes = 1, 2, 3, 8, 16, 32

2.3.1 Discussion and Summary

The MK database usually consists of MFs and performance measures (as target) of different learning algorithms which are predicted accuracies of EoD. These predictive values are computed, in the context of MLL, through BLL. Another level of complexity is introduced by the different parametrizations of the algorithms which were overlooked by several studies where only default configurations were considered. Furthermore, most of them selected only the best algorithm from the pool to minimize the complex representation of MK dataset, therefore very few of them stored ranking. Table 2.3 shows different learning strategies, Base-learners and performance measures that various MLL studies used at Base-level. It can be observed that the 10-fold cross validation strategy, MAE accuracy measure and some learning algorithms have become a norm to use at Base-level. The same Base-level learning strategies are used in some MLL studies for TS with different ARIMA and Exponential smoothing algorithms. A common deficiency that can be concluded from various studies is related to the granularity of information that is being stored in MK database.

Some published literature is segregated into four different performance measure categories (target variable for an MLL system in Table 2.4.

Table 2.4: Different Performance Measures that are used in various literatures

Performance Measure(s)	**Description**	**Research Work**
Best learning algorithm	The performance measure only consists of the classification accuracy of best learning algorithm for each single dataset	Utgoff (1984); Graner et al. (1994); King et al. (1995); Bensusan et al. (2000)
Ranking of learning algorithms	To predict a ranked list of learning algorithms in a pool which are sorted based on a performance measure, e.g. classification accuracy, run-time, etc.	King et al. (1995); Brazdil et al. (2003); Vilalta et al. (2004)
Quantitative Prediction (Reif, 2012)	Directly predict the performance of the target learning algorithm in an appropriate unit, i.e., by training separate regression model for each target algorithm	Gama and Brazdil (1995); Sohn (1999); Kopf and Iglezakis (2002); Bensusan and Kalousis (2001); Reif (2012)
Predicting Parameters	The MLL target variable could be one parameter value or a set of values	Soares et al. (2004); Soares and Brazdil (2006); Kadlec and Gabrys (2009); Lemke and Gabrys (2010a); Filchenkov and Pendryak (2015); Ali et al. (2018)

2.4 Meta-learning

The MK induced by MLL provides a means of informed-decisions based on which algorithms are ranked for a given problem (Giraud-Carrier, 2008). This chapter presents the history of

the most promising decision-support systems for algorithm selection, followed by a review of the applicability of MLL to supervised and unsupervised learning algorithms.

2.4.1 Existing Systems

This section contains a number of MLL systems developed over last couple of decades.

2.4.1.1 Shift To A Better Bias

Based on various studies, a doctoral thesis of Utgoff (1984) is considered as the earliest effort towards MLL systems where a system named Shift To A Better Bias (STABB) was proposed. It was a demonstration that a learner's bias could be adjusted dynamically. Later this work became an initial point of reference and was enhanced in several studies. One of them was Variable-bias Management System (VBMS) by Rendell et al. (1987), where a relatively simple MLL system was proposed. VBMS selected the best among three symbolic learning algorithms as a function of only two dataset characteristics, namely, the number of training instances and the number of features. Rendell and Cho (1990) has further worked on characterizing and investigating the extensive role of data character as a determiner of system behaviour in empirical concept learning. Two main contributions have been brought up: 1) a useful set of MFs based on concept (function or surface over instance) complexity, i.e., shape, size, and structure, that relates a real-world problem to learning algorithm; and 2) an approach of systematic artificial data generation. The results, which focus on measures of complexity, showed that shape and specifically concentration have significant effects.

2.4.1.2 Machine Learning Toolbox

Machine Learning Toolbox (MLT) project by Graner et al. (1994) was one of the initial attempts to address the applications of MLL. MLT produced a toolbox consisting of 10 symbolic learning algorithms for classification. The part of MLT project that assists with the algorithm selection is known as Consultant. The Consultant was based on a stand-alone expert system which maintained a knowledge-base that considered the experiences acquired from the evaluation of learning algorithms. Considerable insight into many important ML issues was gained which had been translated into rules that formed the basis of Consultant-2. Consultant-2 was also an expert system for algorithm selection which gathered user inputs through a set of questions about the data, the domain, and user preferences. Based on the user response relevant rules lead to either additional questions or, eventually, a classification algorithm recommendation. Although its knowledge base had been built through an expert-driven knowledge engineering rather than via MLL it still stands out as the first automatic tool that systematically related application domain and dataset characteristics to classification algorithms. Additionally, Consultant-3 provides advice and help on the combination of learning algorithms. It is also able to perform self-experimentation to determine the effectiveness of an algorithm on a learning problem.

2.4.1.3 Statistical and Logical Learning Project

In Statistical and Logical learning (StatLog) project King et al. (1995) presented the results of comprehensive experiments on classification algorithms. The project was an extension of VBMS by considering a larger number of MFs, together with a broad class of candidate models for algorithm selection. It aimed to compare several symbolic learning algorithms on twelve large real-world classification tasks. Some MLL algorithms were used for model selection task where statistical measures, e.g., skewness, kurtosis, and covariance, that produced higher accuracy have been reported. Additionally, a thorough empirical analysis of 16 classifiers on 12 large real-world datasets and learning models using accuracy and execution time measures of performance were produced. There is no single algorithm that performed best in the experimentation phase. Symbolic algorithms showed maximum accuracy for datasets with extreme distribution, i.e., where distribution was far from normal (i.e., specifically with skew > 1 and kurtosis > 7), and worst in the scenarios where the data is equally distributed. On the contrary, the Nearest Neighbour algorithm was found to be accurate for datasets containing equally scaled and important features.

2.4.1.4 Meta-learning Assistant

Similarly, the Meta-learning Assistant (METAL) project was developed to facilitate the selection of the best-suited classification algorithm for a data-mining task (Berrer et al., 2000). It guides the user in two ways: 1) in discovering new and relevant MFs; and 2) in the selection or ranking of classifiers using MLL process. The main deliverable of this project is the Data Mining Advisor (DMA), a Web-based MLL system for the automatic selection of classification learning algorithms (Giraud-Carrier, 2005). The DMA returned a list of ten algorithms that were ranked according to how well they met the stated goals in terms of accuracy and training time. It implemented ranking mechanisms by exploiting the ratio of accuracy and training time. The choice of algorithm ranking, rather than selecting best-in-class, is motivated by a desire to give as much information as possible and later any number of algorithms could be executed on the dataset.

2.4.1.5 Meta-learning Architecture

The Meta-learning Architecture (METALA), developed by Botia et al. (2001), is an agent-based architecture for distributed Data Mining, supported by MLL. The system supports an arbitrary number of algorithms and tasks, and automatically selects an algorithm that appears best from the pool of available algorithms. Like DMA, each task was characterized by DSIT features relevant to its usage, including the type of input data it required, the type of model it induced, and how well it handled noise. It had been designed to automatically carry out experiments with each learner and task, and induce a Meta-model for algorithm selection. As new tasks and learning algorithms are added to the system, corresponding experiments are performed and the Meta-model is updated.

2.4.1.6 Intelligent Discovery Assistant

The Intelligent Discovery Assistant (IDA) provided a Knowledge Discovery (KD) ontology that defines the existing techniques and their properties (Bernstein and Provost, 2001). It has supported three algorithmic steps of the KD process, including preprocessing, data modelling and post-processing. The approach used in this system was the systematic enumeration of valid data-mining processes so that potentially fruitful options are not overlooked and effective ranking of these valid processes based on user-defined preferences e.g., prediction accuracy, execution speed, etc. IDA systematically searches for an operation whose pre-conditions have been met and whose indicators are consistent with the user-defined preferences. Similarly, its post-conditions search for an operation and the process terminates once the goal has been reached. Once all valid KD processes have been generated, a heuristic ranker is applied to return user-specified goals. Bernstein et al. (2005) research has focused on extending the IDA approach by leveraging the interaction between ontologies to extract deep knowledge and case-based reasoning for MLL. The system also uses procedural information in the form of rules fired by an expert system. The case-base is built around 53 features to describe cases and the ontology comes from human experts.

2.4.1.7 Pattern Recognition Engineering

Mierswa et al. (2006) developed a landmarking operator in RapidMiner as part of Pattern Recognition Engineering (PaREn) project, which is an open-source system for data mining. This operator extracts landmarking features from a given dataset by applying seven fast computable classifiers on it (shown in Figure 2.3).

Table 2.5: Existing Meta-learning Systems

Research Work	**Name**	**Approach**	**Contributions**	**Limitations**
Utgoff (1984)	STABB	Statistical	Initial effort towards MLL	Limited to altering only one kind of learner's bias with fixed order of choices
Rendell et al. (1987)	VBMS	Descriptive	Biases are dynamically located and adjusted according to problem characteristics and prior experience	VBMS is a relatively simple MLL system that learns to select the best among three symbolic learning algorithms as a function of only two dataset characteristics
Rendell and Cho (1990)	Empirical Learning as a Function of Concept Character	DSIT	Complex MFs based on shape, size and concentration, and artificial data generation is used	These complex MFs are expensive to compute

Graner et al. (1994)	MLT	Rule-based	An expert system for algorithm selection by gathering user input through questions and trigger relevant rules while the knowledge-base was built through expert-driven knowledge engineering	Its knowledge base was built through expert-driven knowledge engineering rather than MLL
King et al. (1995)	StatLog	Statistical	A thorough empirical analysis of learning algorithms and models is produced by comparing several symbolic learning algorithms on twelve real-world classification tasks	For a given dataset, algorithms were characterized only as applicable or non-applicable, i.e., they do not provide a way to rank the algorithms; furthermore, that characterization was based on a simple comparison of accuracies regardless of any statistical significance test
Berrer et al. (2000) and Giraud-Carrier (2005)	METAL - DMA	DSIT and landmarking	Discovers new and relevant MFs and algorithm ranking in terms of accuracy and execution time	The outcome of the prediction model is only the best classifier for the new dataset. It does not support multi-operator workflows
Botia et al. (2001)	METALA	Model-based	Agent-based architecture for distributed data-mining, automatically carry out experiments and induce a Meta-model for algorithm selection, it provides architectural mechanisms necessary to scale the DMA	DMA's MFs are used to represent a problem, no contribution to introduce new features
Bernstein and Provost (2001)	IDA	Model-based	Its goal is to rank preprocessing, modelling and post-processing steps that are both valid and consistent with the user-defined preferences	The data should be already pre-processed considerably by the user for IDA to model it and evaluate the resulting models

Bernstein et al. (2005)	IDA - An Ontology-based Approach	Model-based	Extending IDA approach by leveraging the interaction between ontology for deep knowledge and Case-Based Reasoning for MLL	The case-based is built on fixed 53 features and the system is still in the early stages of implementation
Mierswa et al. (2006)	PaREn	Landmarking	A landmarking operator for MLL developed in RapidMiner	Very limited EoD (from UCI) are used to build MK
eLICO (2012)	e-Laboratory for Interdisciplinary Collaborative Research (e-LICO)	Model-based	An e-Laboratory for interdisciplinary collaborative research in data-mining and data-intensive science	Meta-learning component is using RapidMiner's landmarking system which is built on only 90 UCI datasets
Thornton et al. (2013) and Kotthoff et al. (2017)	Automatic model selection and hyper-parameter optimization in WEKA (Auto-WEKA)	Bayesian Optimisation	Tackle Combined Algorithm Selection and Hyper-parameter Optimization (CASH) problem	Some algorithm require Hyper-parameter Optimization (HPO) which is not supported
Komer et al. (2014)	Hyperopt-sklearn	Random search and SMBO	Provides automatic algorithm configuration of the Scikit-learn ML library	-
Feurer et al. (2015)	Auto-sklearn	Bayesian Optimisation	Provides ensemble improvements for Automatic Machine Learning (Auto-ML)	Limited to HPO of shallow learning algorithm
Olson et al. (2018)	Tree-based Pipeline Optimization Tool (TPOT)	Genetic Programming	Optimizes feature preprocessing and section of ML models	Ensemble methods are not supported

2.4.1.8 e-LICO

e-LICO is a project for data-mining and data-intensive science (eLICO, 2012). This project comprises of three layers: 1) e-Science, 2) Application, and 3) Data-mining. The e-Science and data-mining layers form a generic environment that is adapted to different scientific domains by customizing the application layer. The architecture of e-LICO project is shown in Figure 2.5.

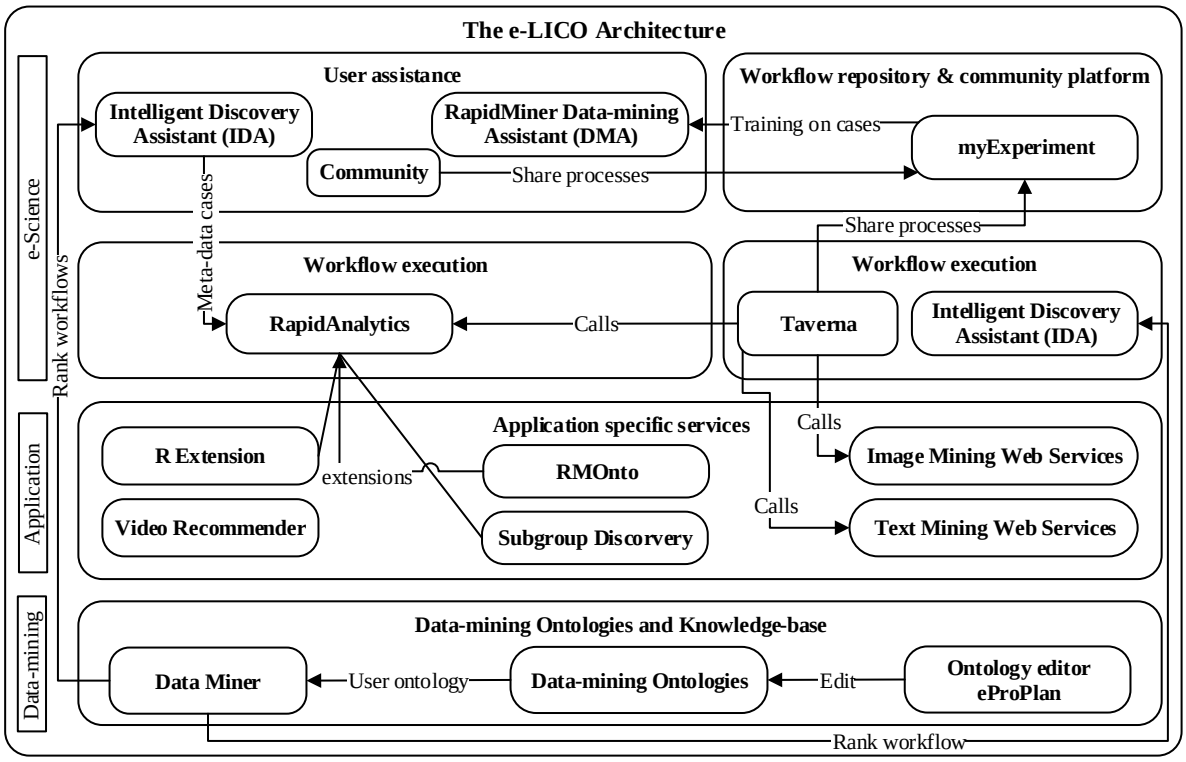

Figure 2.5: e-LICO project architecture

The e-Science layer is built on an open-source e-science infrastructure that supports content creation through collaboration at multiple scales in dynamic virtual communities. The Taverna[9], open-source data-mining and predictive analysis solution (RapidAnalytics) and RapidMiner (Mierswa et al., 2006) components have been used to design and enact data-analysis workflows. The system also provides a variety of general-purpose and application-specific services and a broad tool-kit in designing and sharing such workflows with data-miners all over the word using *myExperiment* portal. The IDA (Bernstein and Provost, 2001) exposed MLL capabilities by automatically creating processes tailored for the specification of input data and a modelling task. The RapidMiner's DMA component helps to design processes by recommending operators that fit well with the existing operators in a process. The data-mining layer provides comprehensive multimedia data-mining tools that are augmented with preprocessing and learning algorithms developed specifically to meet challenges of data-intensive, knowledge-rich sciences. The knowledge-driven data-mining assistant relies on a data-mining ontology and knowledge-base to propose ranked workflows for a given task. The application layer initially comes as an empty shell that has to be built by the domain user from different components of the system. At the application layer, e-LICO is showcased in two application domains: 1) a systems biology, and 2) a video recommendation task.

[9]A suite of tools used to design and execute scientific workflows and experimentation. http://www.taverna.org.uk

2.4.1.9 Auto-WEKA

Auto-WEKA is a Bayesian optimization based tool that tackled the CASH problem (Thornton et al., 2013; Kotthoff et al., 2017). The CASH problem selects an algorithm and optimizes its hyper-parameters simultaneously (Thornton et al., 2013). Auto-WEKA provides a number of BLL and MLL algorithms. The key discrepancies of the tool include; no single base-learner performs well on all the tasks and few ML algorithms require HPO. Auto-sklearn is another Auto-ML toolkit which is an alternative of scikit-learn estimator (Feurer et al., 2015). It uses Bayesian optimization for hyper-parameter tuning of shallow ML algorithms that are implemented within Scikit-learn (Pedregosa et al., 2011).

2.4.2 Regression and Classification

This section covers MLL that is used for regression and classification tasks in different systems.

Todorovski et al. (2002) addressed a novel approach of predictive clustering trees to rank classification algorithms using dataset properties. The approach was to illustrate ML algorithms ranking where the relative performance of the algorithms has to be predicted from a given dataset's MFs. For that purpose the performance of eight Base-level algorithms, mentioned in Table 2.3, has been measured on 65 classification tasks gathered from UCI and METAL project. Furthermore, DSIT dataset characteristics from StatLog and DCT were combined to create an MK dataset consisting of 33 MFs. These properties of individual attributes are aggregated using average, minimum or maximum functions. Landmarking approach has been used in this study with 7 simple and fast learners, shown in Figure 2.3, to investigate the performance of ranking. The proposed dataset characterization approach with clustering tree outperformed DCT and histogram approach which found a grained aggregation of DCT properties with a significant margin.

Vilalta and Drissi (2002a) presented four approaches to MLL consisting of learning from Base-learners, namely, 1) Stacked generalization, 2) Boosting, 3) Landmarking and 4) Meta-decision trees. The information collected from the performance of BLL algorithms is incorporated into the MLL process. Stacked generalization is considered a form of MLL where each set of Base-learners is trained on a dataset and the original feature representation is then extended with the predictions of Base-learners. These predictions are received by successive layers as input and the output is passed on to the next layer. On the contrary, a single Meta-learner at the topmost layer computes the final prediction. Boosting is another approach that is considered as a form of MLL. It generates a set of Base-learners by generating variants of the training set using sampling with replacement technique under a weighted distribution. This distribution is modified for every new variant by assigning more weights to the incorrectly classified examples using the most recent hypothesis. Boosting takes the predictions of each hypothesis over the original training set to progressively improve the classification of those examples for which the last hypothesis failed.

In the last proposed approach, the Base-learners consisted of a combination of several inductive models induced from Meta-decision trees. A decision tree is built where each internal node represented an MF that predicts the class probability for a given example by a set of models whereas the leaf nodes correspond to a predictive model. Given a new

example, Meta-decision tree selects a model that obtains optimized accuracy at predicting the target value.

An instance-based learning algorithm, k-NN, is used to identify the datasets that are most similar to the one at hand by Brazdil et al. (2003). On the contrary, candidate Base-learning algorithms are not ranked but selected based on a multi-criteria aggregated measure that takes accuracy and time into account. The proposed methodology has been evaluated using various experiments and analysis at Base- and Meta-level learning. The Meta-data used in this study was obtained from METAL project which contains estimates of accuracy and time for 10 algorithms (listed in Table 2.3) on 53 datasets, using 10-fold CV. The k-NN algorithm was used at Meta-level to select a candidate algorithm giving the best performance on the given task. Two values of the number of neighbours, 1 and 5, where the k-NN showed significant improvement in the results, particularly with k=1, as compared to the trial-and-error approach.

Two MLL approaches were investigated to select models for TS forecasting by Prudencio and Ludermir (2004) in different case-studies. In the first case-study, single BLL algorithm was used to select models to forecast stationary TS. The base-level and meta-level learning algorithms and configurations are given in Table 2.3 and Table 2.6 for both case studies while details of datasets and MFs are listed in Table 2.1 and Figure 2.3 respectively. In another case study, a more recent and sophisticated approach - NOEMON (Kadlec and Gabrys, 2009) was used to rank three models of the M3-Competition. In both case studies, the experiments revealed significant results by taking into account the quality of algorithm selection and forecasting algorithm performance aspects of the selected models.

Active MLL method, in combination with *Uncertainty Sampling* and outlier detection, has been proposed by Prudencio and Ludermir (2008) to support the selection of informative and anomaly-free Meta-examples for MLL. Some experiments were performed in a case study where MLP was used to predict the accuracies of 50 regression problems at Base-level learning (detail can be seen in Table 2.1) and k-NN[10] at Meta-level. The MFs used in the case study consisted of 10 simple and statistical measures which can be seen in Figure 2.3. The results of the experiments revealed that the proposed approach was significantly better than the previous work on Active MLL. Also, the *Uncertainty Sampling* method increased the performance when the outliers were eliminated from the MK which were only 5% of the data.

Guerra et al. (2008) used SVM, with different kernel functions, as a Meta-regressor to predict the performance of the candidate algorithm, MLP, based on descriptive and statistical features of the learning tasks. For experimentation purposes, the input datasets and MFs used in this study were the same as of Prudencio and Ludermir (2008) work. The MLP was used as a base-learner to compute the normalized MSE which was averaged over 10 training runs. Table 2.3 contains details of the learning strategy which were used at the base-level. At the meta-level, SVM with different kernel functions (listed in Table 2.6) were applied to predict normalized MSE and Mean Absolute Correlation Coefficient (CORR) between predicted and actual target values of the MLP. Later the performance of the Meta-regressor (SVM) was compared with three different benchmarked regression algorithms which were

[10]k = 1, 3, 5, 7, 9 and 11 nearest neighbours

used in the previous work including Linear Regression, k-NN[11] and M5 algorithm (DT Quinlan (1992)). The experiments revealed that the SVM with RBF kernel (particularly with γ=0.1) obtained better performance as Meta-regressor when compared to the mentioned benchmark algorithms.

Kadlec and Gabrys (2009) proposed a generic architecture for the development of online evolving predictive systems. The architecture defined an environment that links four techniques of ML: 1) ensemble methods, 2) local learning, 3) meta-level learning and 4) adaptability and also the interaction between them. The Meta-level learning is discussed in this section whereas adaptability aspects of this paper are discussed in Section 2.5 respectively.

The Meta-level Learning module of Kadlec and Gabrys (2009) architecture was responsible for high-level learning, control, and decision making. Meta-level is the most complex but least diverse top layer of the architecture. In this study, a Meta-learner is defined as building a high-level global knowledge of the model which is incrementally grown by applying the model to various tasks. The main goal of Meta-level layer was to optimize the predictions in terms of the global performance function which can be achieved by; 1) controlling the population at lower levels to cover unexplored parts of the input space, 2) looking for relations between algorithm configurations of the paths and the achieved performance, and 3) adapting the combinations in order to reflect the current state of the data. In general, this layer was used to learn the dependency between the pool of learning algorithms and the performance at various levels. Several experiments have been performed using three real-world datasets from the process industry where adaptive and static techniques were compared. The automated data pre-processing and model selection takes a lot of the model development effort away from the user.

An empirical study on rule induction based forecasting method selection for univariate TS was conducted by Wang et al. (2009). The study aimed to identify characteristics of univariate TS and evaluated the performance of four popular forecasting methods (listed in Table 2.3) using a large collection of datasets listed in Table 2.1. These two components are integrated into a MLL framework which automatically discovers the relations between forecasting methods and data characteristics (shown in Figure 2.3). Furthermore, C4.5 decision tree learning technique was used to generate quantitative rules of MFs and categorical rules are constructed using unsupervised clustering analysis.

Lemke and Gabrys (2010a) investigated applicability of MLL for TS prediction and identified an extensive set of MFs that were used to describe the nature of TS. The feature pool consisted of the general statistical, frequency spectrum, autocorrelation and behaviour of forecasting methods (diversity) measures (see Figure 2.4). These measures are extracted for two datasets, see Table 2.1 for details, and the target was to predict the next 18 observations for NN3[12] and 56 for NN5[12]. Using these datasets empirical experiments have been performed that have provided the basis for further MLL analysis. An extensive list of simple (seasonal), complex (ARIMA), structural and computational intelligence (Feed-forward NN), and forecast combination methods are used for experimentation which can be seen in Table 2.3. From the pool of individual algorithms, NN and MA performed quite

[11] k=1

[12] Neural Network forecasting competition, http://www.neural-forecasting-competition.com

well for NN3 series while for NN5 the SMAPE, in general, was quite high where a combination method *variance-based pooling* out-performed all the individual and combination algorithms. At the end three experiments were performed to explore MFs using decision trees, comparing various MLL approaches (details are given in Table 2.6), and simulating NN5 on *zoomed ranking* method and on its combination. This study concludes that the ranking-based combination of forecasting methods outperformed the individual methods in all experiments.

2.4.3 Clustering

This section discusses the use of MLL in the context of unsupervised learning. De-Souto et al. (2008) presented a novel framework that applies an MLL approach to clustering algorithms, which was one of the initial efforts towards unsupervised algorithms. The proposed architecture was very similar to the MLL approach used to rank regression and classification algorithms. It extracted features of input examples of datasets and associated them with performance of the candidate algorithms in clustering that data to construct MK database. The MK database was used as an input dataset for Meta-level learning and generated a Meta-model that was used in the selecting or ranking of the candidate algorithms at test mode. Some implementation issues were also addressed which include: 1) the selection of datasets; 2) the selection of candidate clustering algorithms; and 3) the selection of the set of MFs that can better represent the problem at Meta-level. In order to evaluate the framework, a case study in the context of cancer gene expression microarray datasets was conducted. Seven candidate algorithms, listed in Table 2.6, and eight descriptive and statistical MFs were extracted, namely, *log10* of the number of examples and ratio of total examples by total features, multi-variant normality, percentage of outliers, percentage of missing values, skewness of Hotelling T^2-test, Chip - type of microarray and percentage of features that were kept after applying selection filter. Also, regression SVM algorithm was used as the Meta-learner. The results were compared with the default ranking, where the average performance was suggested for all datasets. The mean and standard deviation of the SRCC for both rankings generated by the proposed approach was found to be more correlated and significantly higher than the default one.

Soares et al. (2009) employed the De-Souto et al. (2008) framework in the ranking task of candidate clustering algorithms in a range of artificial clustering problems with two different sets of MFs. The first set had five MFs that were calculated using univariate statistics: quartiles, skewness, and kurtosis, in order to summarize the multivariate nature of the datasets. This set included Coefficient of Variation (CoV), CoV of second and third quartiles, CoV of skewness and kurtosis while the other set had the same first four MFs as presented in De-Souto et al. (2008). In this paper, three new candidate clustering algorithms were applied on each learning task that are listed in Table 2.6 and two Meta-learners were used, i.e., Support Vector Regression (SVR) and MLP. The methodology was evaluated using 160 artificially generated datasets (see Section 2.1). Both Meta-learners were applied to the two sets of MFs separately and then compared with the default ranking method. The rankings predicted by the SVR and MLP methods were found to be more correlated and significantly higher than the default ranking. However, there was no significant difference

between the correlation values of MLP and SVR methods for both Meta-datasets. Finally, the authors had also highlighted the selection of MFs in the context of unsupervised MLL as an important issue that could be subjected to further analysis.

2.4.4 Discussion and Summary

There have been several MLL systems developed since the inception of this area. Almost all the systems are developed for algorithm recommendations of classification and regression tasks. Three main MF generation approaches were used in these systems which are listed in Table 2.5, where DSIT approach is found to be the most widely used. A landmarking based algorithm recommendation system is available as a part of RapidMiner, a commonly used open-source data-mining software. It was part of PaREn project where landmarking functionality is available as an operator in the software. One of the most recent and large-scale projects related to MLL is e-LICO, the purpose of which was to solve data-mining and data-intensive problems. This project used MLL for algorithm recommendation by leveraging the existing systems, i.e., IDA and RapidMiner's DMA component proposed by (Bernstein and Provost, 2001). Limitations of those systems are discussed in Table 2.5.

Apart from the existing system, there are several researches where MLL is being used for Regression including forecasting, classification and clustering tasks. Several MF based problem representations are proposed for these regression and classification tasks. Most of the comparisons in those studies are between different MF approaches, selection of candidate algorithms and a different set of Meta-Learners. The problem representation using MFs is the most important aspect where landmarking and Model-based approaches are compared with DCT DSIT features, and outperformed DSIT approach with a significant difference. Not much effort has been put on Model-based approach in the last few years as landmarking with additional DSIT features has been considered as an overall better approach. The landmarking has also been proposed to solve problems other than algorithm recommendations, e.g., Kadlec and Gabrys (2009) used landmarking approach for recurrent concept extraction. Various researches investigated the applicability of MLL for TS problems including Prudencio and Ludermir (2004), Wang et al. (2009), and Lemke and Gabrys (2010a). Prudencio and Ludermir (2004) proposed descriptive and statistical features to represent TS task to rank various seasonal and ARIMA models. Later on Lemke and Gabrys (2010a) used an extensive list of MF covering statistical, frequency spectrum, autocorrelation and diversity measures for a TS prediction task. The pool of TS algorithms contained seasonal, ARIMA, structure and computational intelligence, and forecasting combination methods. The features used in this study to represent TS task at Meta-level were better as compared to the previous studies.

There are a few studies that apply MLL to clustering algorithms. De-Souto et al. (2008) effort was the initial step in investigating knowledge representation for unsupervised problems. Landmarking was used to rank several unsupervised candidate algorithms, as listed in Table 2.6, combined with eight descriptive and statistical MFs which were used to represent unsupervised problems at Meta-level. Most of them were the same as used by several regression and classification problem representations. Soares et al. (2009) employed De-Souto et al. (2008) framework by enhancing the list of Landmarkers and proposed two different

MF representations of unsupervised task. One of the MFs list consisted of features proposed by De-Souto et al. (2008). The results show improvement of the proposed approach over default base-line, but no significant difference is observed in two different representations of unsupervised problems. Finally, the authors have also highlighted the selection of MFs in the context of unsupervised MLL as an important issue that could be subjected to further analysis. All the existing MLL studies discussed in this section are only facilitating the stationary environment. Additionally, these systems have the same issue which was discussed in previous sections that the MK dataset does not have a sufficient number of Meta-examples (MEs).

The idea of using Reinforcement Learning (RL) for algorithm recommendation which introduces an optimal strategy for the tasks sharing a similar structure was proposed by Duan et al. (2016) and Wang et al. (2017a). Wang et al. (2017a) presented a general approach that uses Recurrent Neural Network (RNN) as Meta-learner whose weights are trained 'slowly' over several trials of multiple episodes. The authors show that the strategy based MLL agent outperforms the hand-designed strategies proposed by Auer et al. (2002) and Gittins (1979). In recent years, there have been incredible advances in MLL which are covered in Section 2.6.

Table 2.6: Meta-level learning strategy used in various studies

Research Work	**Learning Strategy**	**Meta-learners**	**Performance**
Sohn (1999)	DSIT approach	Disc, QDisc, LoGID, k-NN, Back-propagation, Learning Vector Quantization (LVQ), Kohonen, RBF, INDCART, C4.5, Bayesian Trees	Disc algorithm ranked as top performing algorithm
Lindner and Studer (1999)	Numeric, Symbolic and Mixed features characterization	NB, MLP, RBF, CN2, Iterative Dichotomiser 3 (ID3), MC4, T2, Winnow, Oblique Classifier-1 (OC1), OneR, Ripper, IBL[13], C5.0 tree, Naive Bayes/Decision-Tree (NBT), Lazy Decision Trees (LazyDT), Parallel Exemplar-Based Learning System (PEBLS)	Numeric and mixed features characterization performed better
Bensusan and Giraud-Carrier (2000)	Landmarking approach compared with Information-Theoretic characterization	NB, k-NN[14], Elite-Nearest Neighbour (e-NN), Decision Nodes Learner (Decision Nodes), Worst Nodes Learner, Randomly Chosen Nodes Learner (Randomly Chosen Nodes), LDA	Landmarking (C5.0 rules) approach outperformed Information-Theoretic
Pfahringer et al. (2000)	Landmarking approach compared with DSIT characterization	C5.0 tree, Ripper, Ltree	Landmarking (C5.0 boost) performed better than others

[13] 0-4
[14] k=1

Peng et al. (2002)	Model-based approach compared with landmarking and DSIT characterization	k-NN	Model-based approach outperformed the remaining two
Prudencio and Ludermir (2004)	Descriptive and Statistical approach	I: Simple ES and Time-delay NN and II: RW, Holt's linear ES (HL), Auto-regressive (AR), NOEMON	I: Simple ES and II: NOEMON performed better
De-Souto et al. (2008)	Landmarking approach to rank unsupervised learning algorithms	Single Linkage (SL), Complete Linkage (CL), Average Linkage (AL), k-Means (k-M), Mixture Models (M), Spectral Clustering (SP), Shared Nearest Neighbours (SNN)	The proposed approach outperformed the default ranking
Guerra et al. (2008)	Descriptive and Statistical approach	SVM with linear, quadratic, and RBF (γ=0.1, 0.05, 0.01) functions	Normalized MSE and CORR between predicted and target values
Soares et al. (2009)	Landmarking approach to rank unsupervised learning algorithms	SL, CL, AL, k-M, M, SNN, Farthest First (FF), DB-Scan (DBS), X-Means (XM)	The proposed approach outperformed the default ranking
Wang et al. (2009)	Statistical approach on TS	ES, ARIMA, RW, NN	
Lemke and Gabrys (2010a)	Statistical approach on TS	NN, DT, SVM, Zoomed ranking (best method and combination)	The proposed approach showed superiority over simple model selection approaches
Abdelmessih et al. (2010)	Landmarking approach compared with Descriptive, DSIT characterization	NB, k-NN, MLP, OneR, RF	Landmarking approach (k-NN) outperformed others
Rossi et al. (2012)	DSIT	RF	MetaStream outperformed default and ensemble approaches
Rossi et al. (2014)	DSIT	RF, NB, k-NN	MetaStream outperformed default and ensemble approaches
Duan et al. (2016)	RL - Trust Region Policy Optimization (TRPO)	DNN	RL^2 outperformed Optimistic Posterior Sampling for Reinforcement Learning (OPSRL) (Osband and Van Roy, 2016)

Wang et al. (2017b)	RL - Actor-Critics (A2C)	DNN	The proposed approach outperformed Auer et al. (2002) and Gittins (1979)
Ali et al. (2018)	DSIT	NN, DT, SVM	Performed well on cross-domain tasks

2.5 Adaptive Mechanisms

The ML and heuristic search algorithms require tuning of their parameters to achieve optimal performance. It can be achieved through off-line sensitivity analysis by testing different parameters to determine their best value in stationary environment (Sikora, 2008). However, the optimal set of values for the parameters keep changing over time in a non-stationary environment because of the change in the underlying distribution of data where off-line sensitivity analysis becomes ineffective. The dynamic problem domain MLL mechanism is considered to be one of the most effective techniques to learn the optimal set of parameters (Sikora, 2008). The rest of this section discusses various techniques of acquiring and exploiting MK in non-stationary environments, that have been proposed in the context of the existing predictive systems.

2.5.1 Recurring Concept Extraction

One of the earliest efforts employing an MLL based approach to achieve adaptivity in a non-stationary environment was presented by Widmer (1997). MLL is applied in time-varying environments for the purpose of selecting the most appropriate learning algorithm. For a traditional two-level learning model different types of attributes are defined at Base- and Meta-level. The predictive attributes are used to induce models at Base-level on raw examples from datasets if there exists a significant correlation between the predictors and the observed class distribution. On the other hand, contextual attributes are employed to identify the current concept associated with the data and systematic changes in their values which indicate a concept drift. These attributes are identified using an MLL approach which is proposed in Widmer (1997). This allows a learning algorithm to select the examples that have the same context as training data and newly arrived examples. These conceptual clues help in adapting the systems faster by filtering the historical instances for training that have the same context as newly arrived instances. The proposed technique was evaluated by comparing two operational systems at the Meta-level that differ in the underlying learning algorithm as well as their way of processing contextual information including METAL(B) that uses Bayesian classifier and METAL(IB) that was based on instance-based learning. The instance-based learner was used in four variants which include: 1) context relevant instance selection; 2) instance weighting; 3) feature weighting; and 4) combination of instance and feature weighting. The general conclusion of numerous experiments that were

performed using real-world and synthetic datasets was that MLL produced a quite significant improvement over the existing approaches for changing environments. Additionally, from the results, it can be observed that the METAL(B) approach proved to be effective in domains (datasets) with high noise rates and several irrelevant attributes whereas instance-based approach showed higher accuracy for the remaining domains.

Klinkenberg (2005) proposed an MLL framework for automatically selecting the most promising algorithm and its parametrization at each step in time where the data was arriving in batches. For each batch a set of MFs (as listed in Table 2.8) are extracted directly from the raw data which is used in the BLL to create a Meta-example. A number of Meta-examples are used to induce a Meta-learner whenever a new batch becomes available, which in turn, helps in predicting the best learning algorithm and the best set of instances at a given time point. The MFs used in this work are more relevant to the problem under analysis. Furthermore, this work also investigates the aspects used to speed-up the algorithm selection process using the proposed MLL approach without losing the gained reduction in error rate. The proposed drifting concept approaches, i.e., adaptive time window and batch selection strategy, were evaluated by comparing them with three non-adaptive mechanisms: 1) full memory, 2) no memory, and 3) fixed size window. The experiments were performed using two real-world problems: 1) information filtering of unstructured business news data; and 2) predicting business cycle from economics domain. For business news dataset both adaptive techniques outperformed trivial non-adaptive approaches. Two evaluations were performed for business cycle dataset where the data was split into 5 and 15 equal sized batches where the fixed size window approach performed slightly better than the adaptive techniques.

2.5.2 Periodic Algorithm Selection

Sikora (2008) proposed MLL mechanism to learn the optimal parameters while the learning algorithm is trying to learn its target concept in a non-stationary environment. MLL is used to tune temperature (τ) parameter of Softmax RL algorithm using Boltzmann distribution. Moreover, the time-weighted method has been used where the action value estimates are the sample average of prior rewards. The Softmax algorithm becomes a random search in case of higher τ value, whereas for the low value it approaches a greedy search. The effectiveness of the proposed MLL algorithm is evaluated by dynamically learning the optimal value of τ using two case-studies: 1) k-Armed bandit - the classic RL problem, and 2) bidding strategy - stylized e-procurement problem. In k-Armed bandit problem the variable k is defined as actions available to an agent and each action returns a reward from a different distribution. In this work (k=) 10 actions (1,...,10) were available to an agent where each action returns a reward using Normal distribution. The effectiveness of MLL in the non-stationary environment is tested by rotating the reward distributions among the 10 actions. The algorithm is tested with three different temperature parameter values of 5, 50 and 500 for stationary and dynamic environments. For the stationary environment, the performance of τ=5 approaches the best action with the maximum average reward. As the environment becomes more and more dynamic these awards keep falling. On the contrary, the performance of the MLL algorithm returns a better reward in both environments as well as responds faster to the changes in the environment. The bidding problem was analysed as a 2 player symmetric

game (2 homogeneous sellers) with n actions, where n is the variable cost (price) range split into equally sized bands. One of the sellers was modelled using softmax RL algorithm while the other one was supposed to be using different learning algorithms, i.e., ϵ-greedy - a genetic algorithm proposed by Goldberg (1989). The same three values of τ were used for both stationary and dynamic environments, where the stationary environment produced the best result for the lowest value of temperature. However, no single value of temperature did best in the dynamic environment, while MLL algorithm approached the best reward for both environments. Furthermore, it was observed from the experiments that the best value of τ was achieved from MLL approach in all the scenarios.

Kadlec and Gabrys (2009) architecture supports life-long learning by providing several adaptation mechanisms across computational path level (preprocessing methods followed by individual base-level algorithms), path combination level (combination of base-level algorithms) and Meta-level hierarchical structure. There are four adaptation loops defined across various levels of hierarchy including self-adaptation capability of the computational and combination layer, whereas the remaining two loops connect Meta-level layer to the lower layers. These loops help the proposed architecture to keep the validity of the models in the changing environment. It can be achieved by switching particular modules to the incremental mode. The computational path level adaptation loop consists of the predictions feedback which are compared to the actual (target) values. Whereas at path combination level the combinations are represented in the same way as in the computational path, which is a benefit of this representation that similar adaptation mechanisms can be applied at different levels. In case of weighted combinations, the contribution of particular computation paths is dynamically changed to the final prediction by modifying the weights. Meta-level adaptation has an influence on the dynamic behaviour of the entire architecture. At this level, the performance measures are gathered from all levels of the architecture together with global performance. It allows to analyze the performance achieved across various levels and also to estimate the influence of the changes at different states of the model. Several experiments demonstrate that the variety of adaptation mechanisms applied at different levels may have a significant effect on the performance of the models. One of the key contribution of the proposed architecture, which in turn, has opened space for future research that will focus on the interaction between different techniques, dynamic behaviour, implementation of novel adaptation techniques and application of more sophisticated approaches for the meta-level methods.

A comprehensive framework, design problems, taxonomy of adaptive learning and different areas of learning under concept drift is presented by Zliobaite (2010). The proposed framework is used to analyze the problem of training set formation where two areas, i.e., 1) incremental learning; and 2) causes of concept drift are discussed. The incremental learning explains the difference between concept drift and periodic seasonality with examples while the causes of concept drift are elaborated on using Bayesian decision theory, where three causes are highlighted that might change over time. There are four design sub-problems and techniques addressed within the framework that need to be solved: 1) future assumptions about source and target instances; 2) structural change types or configuration patterns of data over time; 3) identified four key learner adaptivity areas and 4) model selection which is further categorized into two different groups. The taxonomy of concept drift learners

is categorized as evolving learner where four methods are proposed and the methods that determined how the models or instances are changed at a given time are grouped separately under triggering concept. At the end three major research areas are outlined: 1) time context; 2) transfer learning by gaining knowledge from a similar type of past problems; and 3) models which have properties of adaptation incorporated into learners. Also, several dimensions which are relevant to the applications implementing concept drift are defined. Figure 2.6 presents all the key areas and available solutions of 'learning under concept drift'.

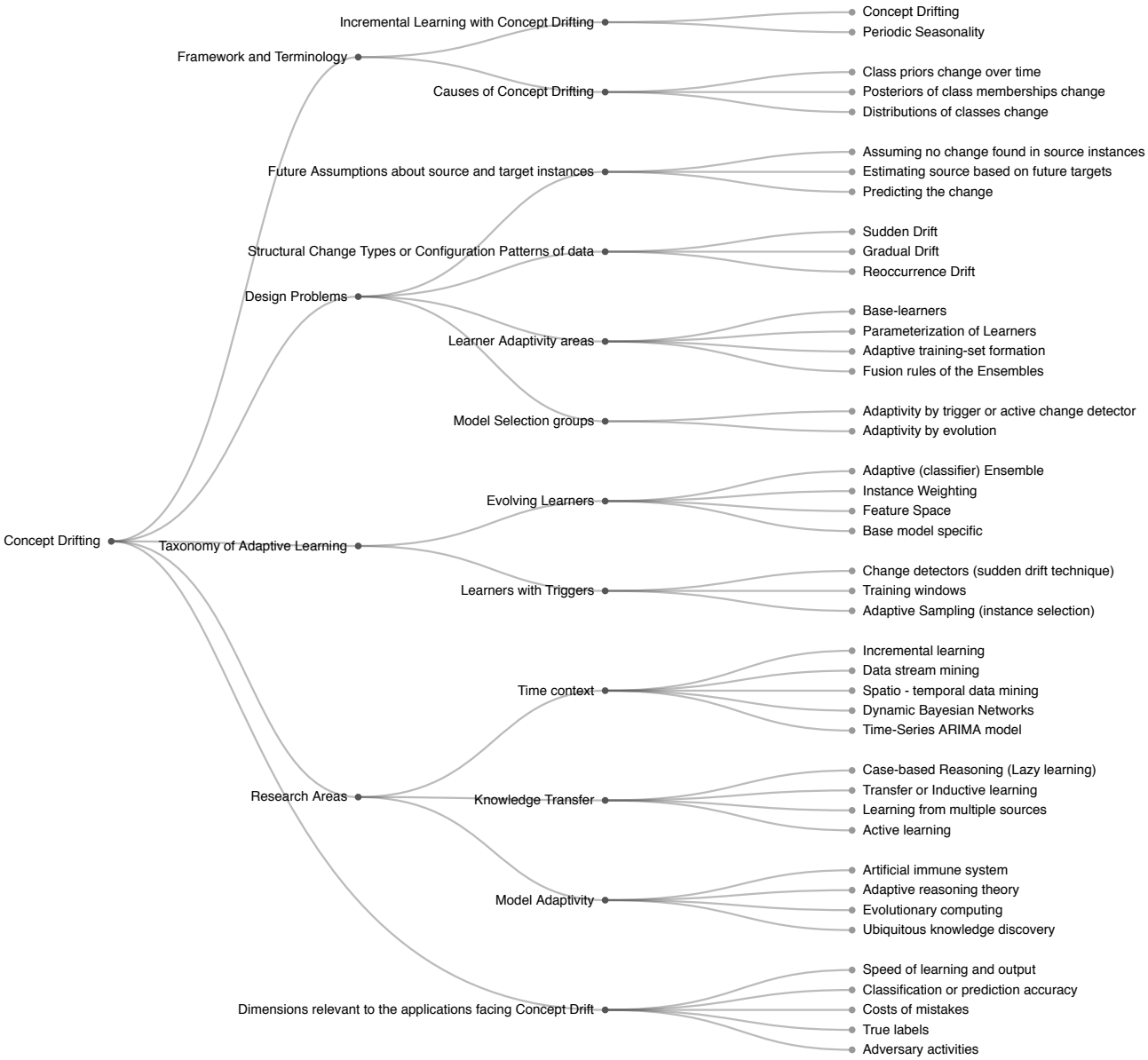

Figure 2.6: Learning under Concept Drifting (Zliobaite, 2010)

An MLL approach for periodic and automatic algorithm selection for time-changing data, named Meta-Stream, is presented by Rossi et al. (2012). A Meta-classifier is periodically applied to predict a learning algorithm optimized for the best performance on a new unlabelled chunk of data. General DSIT MFs of Travel Time Prediction (TTP) problem are extracted from the historical and new data (as shown in Figure 2.3) and mapped

together with their predictive performance computed from different models to induce the Meta-classifier. Experiments are performed to compare the performance of the MetaStream to the default trial-and-error approach for both static and dynamic updating strategies at Meta- and Base-levels. Moreover, the Base-level MetaStream and Default results are compared with the dynamic Ensemble approach. The learning strategy adopted at Base-level can be seen in Table 2.3, also the training window (ω) of 1000 instances with a step size (ς) of 1 was used at this level. The Meta-level learning strategy is presented in Table 2.6. The MEs labelled as tie are investigated separately by keeping and discarding them from the training and test sets. The empirical results show that MetaStream outperformed baseline and ensemble approaches with a significant margin in most of the cases for both stationary and dynamic environments. In general, the two pairs of algorithms, e.g., RF-CART and SVM-CART were found to be the best algorithms for TTP problem. Finally, the authors also realized that the MFs should be related to the non-stationary data problem rather than characteristics which are extracted for traditional MLL problems.

2.5.3 Meta-level Representation of Non-stationary Problems

Rossi et al. (2014) extended the original work (Rossi et al., 2012) in two main directions: 1) instead of selecting only a single algorithm, combination of multiple regressors can be selected, when the average of the predictions perform better than the individual; and 2) more comprehensive experimental evaluation is performed by adding another real-world problem - *Electricity Demand Prediction (EDP)* (see Table 2.1). Furthermore the list of MFs extracted from the data is also enhanced in this work, as listed in Table 2.7. The characteristics are extracted separately from training and evaluation windows because the training window has target information available from where supervised characteristics can be extracted, i.e., the information about the relationship between predictive and target variables. The pool of Base- and Meta-level algorithms with their configurations are listed in Table 2.3 and Table 2.6 respectively. The experimental results show that for TTP dataset the pair of regressors, regardless of the presence of tie resolution strategy, outperformed Default and Ensemble approaches. However, in case of EDP, MetaStream clearly outperformed default but was worse than Ensemble which can lead to a conclusion that the observations made for pairs of regressors are also valid for multi-regressors. Moreover, the slightly higher error rate is recorded for RF Meta-learner of the MetaStream than the Default but was lower than Ensemble approach for the TTP dataset, whereas for EDP dataset MetaStream outperformed Default but was worse than Ensemble. These results show that MetaStream is able to select the best algorithm more accurately than baseline trial-and-error and ensemble-based approaches in a time-changing environment.

Table 2.7: Meta-features used in MetaStream to characterize the data

Meta-features	**Training window**	**Selection window**
Average, Variance, Minimum, Maximum and Median of continuous features	✔	✔
Average, Variance, Minimum, Maximum and Median of the target	✔	

Correlation between numeric features		✔
Correlation of numeric attributes to the target	✔	
Possibility of existence of outliers in numeric features		✔
Possibility of existence of outliers in the target	✔	
Dispersion gain	✔	
Skewness of numeric features		✔
Kurtosis of numeric features		✔

2.5.4 Discussion and Summary

This section covers the adaptability mechanisms of the system which lead to the thorough study of several existing studies. In these studies, the main focus was put on the applicability of MLL particularly in the context of non-stationary environments. MLL can be very beneficial for this environment in minimizing the processing-time that is consumed to periodically train the model, extracting recurring concepts, automatically detecting concept drift and estimating dynamic adaptive window size, which in-turn generate accurate predictions in dynamic environments. However, applying MLL to support adaptive mechanism is a quite recent and emerging area. As a result most of the research take into account the same MFs for time-varying environment which have been used to represent the algorithm recommendation problem in the context of stationary environments. Whereas if MLL is introduced in any system then the overall performance becomes dependent on appropriate representation of the problem at Meta-level in the form of MFs. The drawback of using a set of MFs which are usually used in stationary environment is that the entire target dataset should be available at-once when MLL is applied to find the learning algorithm that obtain optimal performance for that dataset; which is not the case when instances or batches of data keep coming because there are some useful MFs which cannot be computed in the absence of target variable of the incoming data.

Widmer (1997) work on applying MLL for non-stationary environment is considered to be the earliest effort. The author addressed two key areas in context of dynamic environment: 1) dynamic tracking of changes and 2) extraction of recurring concepts. The problem representation of Widmer (1997) was quite general, in that, very few predictive and contextual MFs were extracted, therefore neither of the two proposed MLL approaches performed better then the Default for several domains. On the other hand, the adaptive parameters, such as, window size, were fixed in this work. Klinkenberg (2005) used different BLL algorithms and their parametrization which are automatically selected at Meta-level. Additionally, Meta-level approach for adaptive time window and recurring concept extraction for the target concept were part of the research. The research is one of the initial efforts to represent adaptivity problem with the relevant MFs rather than using general features which are usually productive for stationary environment. Although these features (as listed in Table 2.8) are not enough to represent non-stationary environment at Meta-level, but they are still better than general features (used to represent stationary problems) supported by the experiments, which showed a significant improvement.

Sikora (2008) proposed reinforcement learning approach to address the automatic algorithm recommendation problem using MLL in a non-stationary environment. The focus of the research was to find the optimal value of the Softmax algorithm's parameter τ where it would recommend the best algorithm for target concept at Meta-level. The same deficiency is observed in this work that the non-stationary problem representation was not addressed in detail and focus was only on algorithm recommendation using MFs which were proposed for static data. Kadlec and Gabrys (2009) proposed life-long learning architecture that provided several adaptation mechanisms across a pool of candidate learning algorithms and their combinations. The dynamic behaviour of the entire architecture is analyzed at Meta-level where the global performances and information from both pools can be analyzed to estimate the influence of the changes at different levels of the model. The decrease in prediction ability of local model below a certain level is considered as a new concept which leads to building a new receptive field. The landmarking approach is quite simple and effective to detect concept drift, and based on that, periodically train new local predictor. The effectiveness of MLL for the two mentioned areas is supported by improved results recorded from two case-studies.

Rossi et al. (2012) approach was quite similar to Klinkenberg (2005) where periodic algorithm selection for time-changing data was proposed. Likewise in various other studies, the authors computed the DSIT MFs. Although the Meta-level approach performed better than the Base-level, but there is no comparison shown with the other MLL system from where it could be concluded that even the general representation of the problem can work for the non-stationary environment. Problem representation using general MFs is the discrepancy of this effort which is being tried to overcome in Rossi et al. (2014). The authors computed separate MFs for historical and incoming data. As target variable has been absent from the incoming data so unsupervised features were computed for the data available in the evaluation window. The performance of the proposed approach was better than BLL and worse than Ensemble but still it was considered to be a good effort towards representing time-varying problem at Meta-level. In almost all the researches that are discussed in this section MLL outperformed the BLL methods. However, a common discrepancy is observed in problem representation at Meta-level for time-varying data. Most of the work used general MFs whereas some tried to focus on this area by proposing some features for non-stationary data.

Model-Agnostic Meta-Learning (MAML) is an optimal fast adaptation method which learns a model initialization in few shots such that it can be adapted to solve a new task (Finn et al., 2017). MAML first learns task-specific parameters by performing one gradient step at a time and then learns model parameters in a way to minimize the expected loss across multiple tasks. The objective is to learn a model initialization that can be generalized well to a new task in a few gradient updates. Nagabandi et al. (2018) proposed a method to learn incoming stream of data using DNN along with MLL and applied it to the model-based RL. The authors used MAML to learn the initial weights whereas Chinese restaurant process is used to learn task distribution.

Table 2.8: Adaptive mechanisms used in previous studies

Research Work	**Adaptivity mechanisms addressed**	**Meta-features/Parameters**
Widmer (1997)	Recurring concept extraction	ω=100 and significance level=0.01
Klinkenberg (2005)	Recurring concept extraction, adaptive time window, periodic algorithm selection	No. of batches used for training at previous batch No. of non-interrupted most recent training batches Most successful learner on the previous batch Most successful learner overall on all batches have seen so far
Kadlec and Gabrys (2009)	Concept drift detection and Periodic algorithm selection	Landmarking
Rossi et al. (2012)	Periodic algorithm selection	ML: ω=1000, ς=1, η=0 MLL: ω=300, γ=25, ς=1, η= 0
Rossi et al. (2014)	Periodic algorithm selection (with more relevant representation of the non-stationary problem)	TTP dataset: ML: ω=1000, ς=1, η=2; MLL: ω=300, γ=24, ς=1, η=0 EDP dataset: ML: ω=672, ς=336, η=0; MLL: ω=300, γ=25, ς=1, η=0
Finn et al. (2017)	Gradient based few shot learning adaptation method	$\mathcal{T}_i$ is time horizon (e.g., $\mathcal{T}_i = 1$ for classification tasks) q_i is the transition distribution ($q_i(x_1)$ is prior over initial observations) $\mathcal{L}_{\mathcal{T}i}$ is loss function (cross entropy for classification tasks) $p(\mathcal{T}_i)$ is distribution to draw a task
Nagabandi et al. (2018)	Adaptation of DNN using MLL and applied it to multi-task RL	MAML

2.6 Hyper-parameter Optimization

The previous sections provide a thorough understanding of various phases of MLL. This method of Auto-ML learns from prior experience in a systematic way. This section explores two new methods of Auto-ML. DL is one of the recent advancement in ML which triggered a paradigm shift in MLL (Minar and Naher, 2018). This shift minimizes the impact of the some phases of the 'traditional meta-learning' pipeline. Arguably, the gathering of related EoD, and MF generation and selection tasks of MLL are intrinsically taken-over by DNN automatic feature extraction and RL mechanisms. The MLL applied on RL is known as Meta-RL. In Meta-RL a task is specified through a reward function and the agent needs to improve its performance by acting in the environment. The agent receives a reward from the

environment and adjusts its strategy accordingly. Hutter et al. (2018) presents an overview of the different methods of Auto-ML which are further categorized into three key approaches including MLL, HPO, and Neural Architecture Search (NAS) as shown in the Figure 2.7. In this research, the model selection is dealt with 'learning from task properties' technique of MLL, however, the hyper-parameters optimization is targeted with 'learning from prior models' technique and 'reinforcement learning in the context of NAS'. An overview of the existing work on Transfer Learning (TL) and Meta-RL in context of Auto-ML is discussed in the following sections:

2.6.1 Transfer Learning of Deep Models

TL mainly focuses on learning the common features that can get benefit for multiple tasks. In Auto-ML, its applications are mostly in network architecture search, however, the knowledge transfer process from one task to the other is not addressed in an automated manner.

The DNN have attained tremendous success by consistently outperforming the shallow learning techniques. However, solving complex tasks need deeper and wider networks which are considered hard to design. Transfer learning, often, works well on simple and more general tasks whereas complex tasks require effort to design a customized network. The network designing process requires specialized skills and numerous trials which is a time consuming and computationally expensive task. The state-of-the-art networks require well-tuned hyper-parameters which often demand numerous computationally intensive trials.

Among the key developments in the field of DL, Convolutional Neural Networks (CNNs) stands out as the workhorse of Computer Vision. Training a large CNN with millions of parameters is a computationally intensive task which also requires a significant amount of training data. However, several state-of-the-art image classification architectures trained on large image datasets are publicly available, including Visual Geometry Group Network (VGGNet) Simonyan and Zisserman (2014), Inception (Szegedy et al., 2015), Residual Networks (ResNet) (He et al., 2016) and Inception-ResNet (Szegedy et al., 2017). These networks are trained on the ImageNet (Russakovsky et al., 2015) dataset which consists of 1.2 million images and 1000 classes.

Training of these types of deep networks from scratch on a huge dataset is a computationally demanding task. As a result, TL, i.e. reusing parts of the pre-trained models either as-is or as a starting point within the training process, quickly became a de-facto standard in Computer Vision tasks. The general consensus seems to be that the more data one has, the more 'aggressive' the re-training process can be (e.g. re-training more final layers). Conversely, the more similar the new dataset is to the one used to train the original model, the fewer layers need to be fine-tuned. Despite the wide adoption of TL in the context of CNNs, to the best of our knowledge, there is still no principled way of approaching this process. The number of layers to re-train or even the network architectures themselves are chosen in an ad-hoc manner and tested one after the other, which is a computationally inefficient procedure.

Recently, MLL has become a crucial component of DL for the selection of hyper-parameters of a specific architecture. Miikkulainen et al. (2017) proposed a comprehensive set of global and node level hyper-parameters which are critical in optimizing deep learning

Figure 2.7: A holistic view of Automatic Machine Learning areas and systems

architectures through evolution. The use of Reinforcement learning to generate CNN and RNN architectures have been proposed by Baker et al. (2016) and Zoph and Le (2016). They have used Q-learning to produce new CNN architectures. Finn et al. (2017) introduced a simple but powerful approach, model-agnostic meta-learning, which provides an optimal initialization of model parameters that lead to fast learning on new tasks.

TL has been positioned to effectively adapt pre-trained networks to a new domain by fine-tuning their final layers. Some studies, such as Wang et al. (2017b) and Shin et al. (2016), propose re-training of only final fully-connected (FC) layers of the network which does not guarantee state-of-the-art accuracy, particularly on relatively dissimilar tasks. On the contrary, domain adaptation becomes beneficial by fine-tuning an increasing number of layers based on the complexity and relevance of the new task (Yosinski et al., 2014). Therefore, a question arises as to how many blocks need fine-tuning to adapt to a new domain based on the complexity, size and domain relevance.

The significant breakthrough in the field of ML and computer vision began when AlexNet achieved state-of-the-art image classification accuracy against all the traditional approaches in 2012 Krizhevsky et al. (2012). Since then CNN based architectures have been consistently outperforming other approaches in the end-to-end image and video recognition tasks Krizhevsky et al. (2012). The key reasons of this success are large public image datasets, such as ImageNet (Deng et al., 2009) and CIFAR ("CIFAR-10 and CIFAR-100"), high-performance computing – Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs), and ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2014). Indeed, ILSVRC served as a platform for several state-of-the-art DL architectures which are trained on ImageNet.

Regardless of the proven success of CNNs, some limitations are still tagged with this area. They require large amounts of labeled data and massive processing to optimize millions of parameters. This limitation has been overcome by leveraging TL which acquires knowledge on a specific problem and reduces it to a different but related task (Yosinski et al., 2014).

2.6.2 Meta-Reinforcement Learning

The previous section discusses TL of DNN for related tasks, an area of MLL which learns from prior models (Tan et al., 2018). This section gives an overview of HPO of DNN using RL. A typical setting of RL consisting of an agent which performs actions on the environment (Sutton and Barto, 2015). The agent observes different states of the environment in different time-steps. The environment can have several states. Mostly, the agent observes a specific state at a time-step to choose a set of actions. The agent uses a policy to choose which actions to take. The selected actions generate a reward which can be used with the behaviour of the environment. The behaviour of the environment can help the RL to understand the effectiveness of the recommended actions. In fact, an effective agent is the one which maximizes the expected reward.

An RL problem is defined as a Markov Decision Process (MDP) which is characterised as the tuple of $< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma >$:

- s $\in \mathcal{S}$ set of finite states
- a $\in \mathcal{A}$ set of finite Actions
- $\mathcal{R}$ is a reward where its function is defined as $R_s^a = \mathbb{E}[R_{t+1}|S_t = s, A_t = a]$
- $\mathcal{T}$ is a state transition probability which is defined as $T_{ss'}^a = \mathbb{P}[S_{t+1} = s'|S_t = s, A_t = a]$
- γ is a discount factor defined as $\gamma \in [0, 1]$

- π is a policy which is defined as $\pi(a|s) \leftarrow \mathbb{P}[A_t = a|S_t = s]$
- The expected reward is defined as $R_t \leftarrow \sum_{i=0}^{\infty} \gamma^i R_{t+i}$

There are a number of recent studies around HPO using RL. The earliest effort of Meta-RL was made by Duan et al. (2016) where an RNN based agent is used to learn the behaviour of the environment. The goal of the agent is to learn a policy for learning new policies. The Meta-RL is defined in this work in a way that the agent gets trained once on a problem and transfer learned on similar kind of tasks. Moreover, the idea is a learning policy to learn another policy in a family of similar Markov Decision Processes (MDPs). A Meta-agent adjusts its policy after training for a few episodes and validates on an unseen environment. This approach worked well on both small- and large-scale problems. Another simple, yet powerful Meta-RL approach is MAML (Finn et al., 2017). MAML does not initialize model parameters randomly but rather it provides a good initialization to achieve optimal and efficient learning on a new task. The fine-tuning requires a small number of gradient steps. The key aspect of the MAML is that the model can be trained using a gradient descent including CNNs with a variety of potential loss functions. Additionally, it is equally effective for regression, classification and reinforcement learning, where it outperformed a number of previous approaches.

Ravi and Larochelle (2017) proposed a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) based approach to train a meta-classifier. The few-shot learning method finds the optimal set of parameters. However, Finn and Levine (2018) claims that the MAML initialization of the model parameters is more resilient to over-fitting, particularly, for smaller datasets. Also, it is more effective when the model is dealing with new unseen tasks. Similarly, Ali et al. (2019b) proposed an effective and efficient domain adaption approach by fine-tuning the final layers of a CNN for both small- and large-scale problems.

An agent trained using Q-learning with an epsilon-greedy exploration strategy that can generate high-performing CNNs on a given task has been proposed in Baker et al. (2017). The agent designs new architectures without human involvement. The proposed approach was tested on a number of image classification tasks where it outperformed existing meta-modeling approaches applied for network design tasks. The agent makes sequential decisions to generate a network configuration. However, the HPO of a task requires intense computation for several days.

NAS is another effort towards Meta-RL based network search (Zoph and Le, 2017). NAS uses an RNN based controller that samples a candidate architecture known as child network. The child network is trained till convergence to obtain accuracy on a hold-out validation-set. The accuracy is used as an immediate reward which further updates the controller. The controller generates better architectures over time where the weights are updated by policy gradient. The approach seems quite simple and powerful but it is tested on very small size tasks. Another observation is that the search space of the child network was limited. The reason behind limiting the experiment to small tasks is the inefficiency of the approach.

Progressive Neural Architecture Search (PNAS) proposes a different approach to architecture search known as SMBO strategy (Liu et al., 2018). In SMBO, instead of randomly recommending and testing out the blocks, they are tested and structures are searched in

order of increasing complexity. Instead of traversing the entire search space, this approach starts off simple and only gets complex when required. PNAS claims to be significantly less computationally expensive than NAS. Another effort to make architecture search more efficient is known as, Efficient Neural Architecture Search (ENAS), proposed by Pham et al. (2018). ENAS allows sharing of weights across all the models instead of training every model from scratch. The idea is to reuse the weights of a block which are already trained. Thus, the system uses transfer learning to train a new model which makes convergence very fast. It is a very effective method and comparatively less computationally expensive than PNAS. The only observation about this approach is that it keeps a large number of architectures in the memory.

Xu et al. (2018) proposed a different approach of learning to do exploration in off-policy RL which is Deep Deterministic Policy Gradients (DDPG). The authors compared two different policy gradient RL approaches: a) On-policy Gradient Algorithms (OPGA) which includes algorithms like Proximal Policy Optimization (PPO) and b) TRPO where a stochastic policy is used for exploration of RL environment. A separate policy has been used instead of a simple heuristic one for the exploration. This policy is trained using OPGA methods where the reward for training is a relative improvement in the performance of the exploitation policy network. Experimental results show faster convergence of DDPG with higher rewards. Zoph et al. (2018) further extended NAS where they also replaced REINFORCE with PPO.

Table 2.9 is showing the comparison of various systems with neural architecture search systems. The comparison includes the number of GPUs used in the experiments, exploration time and the accuracy of the best performing architecture.

Table 2.9: Hyper-parameter search techniques used in previous studies

Research Work	**GPUs**	**Exploration time (days)**	**Error rate (%)**
DenseNet (DeVries and Taylor, 2017)	-	-	3.46
NAS with Q-Learning (Baker et al., 2017)	10	8-10	6.92
NAS (Zoph and Le, 2017)	450	3-4	3.41
PNAS (Liu et al., 2018)	100	1.5	3.63
ENAS (Pham et al., 2018)	1	0.45	2.89

2.7 Research Challenges

The goal of MLL is to recommend a learning algorithm that gives the optimized performance on new tasks based on the previously solved problems and with minimal or no intervention of human experts (Duch et al., 2011). The existing approach of analysing the problem and selecting an optimal learning algorithm is to apply a wide range of algorithms, with many possible parametrizations, on a problem simultaneously and then select an algorithm from a ranked list based on performance estimates like accuracy, execution-time, etc. Also choosing an algorithm optimized for the best performance in an ever increasing number of models and

their numerous configurations is a challenging task. Even with sophisticated and parallel learning algorithms, the computational power in terms of execution-time, memory and the overall human effort is still one of the biggest limitations. Every task leads to new challenges and demands dedicated effort for detailed analysis and modelling.

In recent years, Auto-ML is getting traction and has become a key area of ML. The ML pipeline consists of several task dependent phases, such as feature engineering, model selection and HPO (Yao et al., 2019). These phases require human intervention to be carefully tuned based on the complexity of a given task. Thus, with the emergence of DNN, the NAS approach of Auto-ML is becoming critical.

The main theme of this work is research on MLL strategies and approaches for effective reduction of the model search space. There are multiple areas of a predictive system where MLL can be used to efficiently recommend the most appropriate methods and techniques. Therefore, three areas of evolving predictive systems are identified where the applicability of MLL can be an effective and efficient approach. These are listed below:

1. Learning Path Recommendation:
   A learning path includes pre-processing steps, learning algorithms or their combination and adaptivity mechanism parameters. These three components are interlinked with each other where MLL recommends the learning algorithm or their combinations preceded by optimized pre-processing steps from a pool of available methods. The adaptivity mechanism parameters are the additional parameters which are linked with the algorithm's configuration. Figure 2.8 shows the complex learning path recommender.

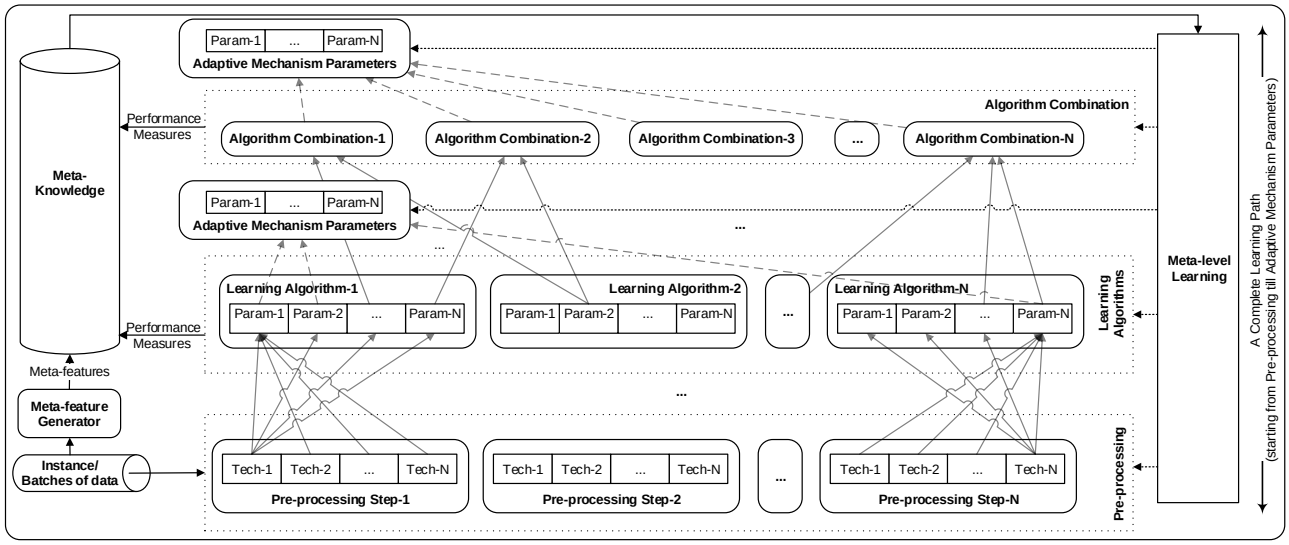

Figure 2.8: Learning Path Recommendation

   i. Pre-processing Steps Recommendation:
      MLL can be applied to find the most appropriate combination of pre-processing steps. Since in time-varying environment trying various pre-processing methods and techniques to find the best combination for a concept will make the entire system ineffective. Instead of spending time on testing various methods on every concept

drift detection MLL can help to instantly recommend the best pre-processing steps from the methods under observations.

ii. Algorithm or Combination Recommendation:
Finding an optimal algorithm for a dataset is a traditional application of MLL (Giraud-Carrier, 2008). Automatic discovery of optimal algorithm can be beneficial for both stationary and particularly non-stationary environments where it can help in minimizing the processing time which is usually spent on the rigorous testing of various learning algorithms with their different parametrizations. MLL can recommend the best learning algorithm, its parametrization and their combination instantly from the pool of available learners.

iii. Adaptivity Mechanism Parameters:
The adaptive mechanism with static parameters, i.e., training and evaluation window size, step size and delay, would be ineffective for the dynamic environments where the underlying distribution of incoming data keeps changing. These parameters can be bound with learning algorithm configuration. The most appropriate set of adaptivity parameters can be extracted at Meta-level based on the best learning algorithm selected for the current concept.

2. Recurring Concepts Extraction:
In a non-stationary environment, the underlying distribution of the incoming data keeps changing which in turn makes the most recent historical concept ineffective to retrain the model for current concept. Using MLL the historical batches (concepts) of data could be extracted from MK, which in turn, can be used as a training-set for the current data. This process can be named as *Reverse Knowledge Extraction* where MFs of the current concept can be used to extract the MEs of relevant concepts from MK datasets. These MEs will ultimately lead to extracting the model whose underlying distribution follows the concept which is currently under observation. This model can be retrained to incorporate the new concept in the existing model.

3. Concept Drift Detection:
In an adaptive mechanism retraining of model is usually triggered by a change detection process. MLL can help in automatically identifying a drift to maximize the efficiency of the system. MLL can help to automatically detect the concept drift and trigger the algorithm retraining process instantly. For instance, the MFs of incoming data can be computed as well as cumulated on arrival of every batch and simultaneously compared with the set of MEs, from MK dataset, whose learning algorithm (used as target variable in MK) is used to score the current batches of data. The concept drift is detected at Meta-level if the ME of the current concept does not match with the cluster of MEs whose learning algorithm is currently selected.

The scope of this research is limited to the feature engineering and learning strategy for algorithm recommendation which falls under Algorithm or Combination Recommendation. The applicability of MLL on this area leads to several research questions which are listed below:

1. Gathering examples of datasets to build a static Meta-knowledge database:

    i. The time-changing environments require dynamic MK databases which must be updated with the MFs of different batches of data having a different distribution. A dynamic MK database keeps on growing with the ME of new concepts. Apart from the dynamically growing database which will gradually build-up, a static MK database may be required at least for the initial phase of the system. When do the benefits of a static database outweigh the costs of maintaining it? Furthermore, what are the alternative techniques of utilizing MLL without having prior knowledge particularly for the initial phase of the system?

    ii. Building-up a static MK database would raise another research challenge of what strategy should be adapted to generate synthetic MEs, i.e., either by directly transforming the existing MEs which are generated by limited real-world datasets or by generating artificial examples of datasets?

2. Base-level Learning strategy to compute performance measures of Meta-examples:

    i. BLL is used to build predictive models using examples of datasets to compute a set of performance measures which are mapped with their respective MEs. What strategy would be adopted to select the best learning algorithm and its parametrization for an ME at Base-level, i.e., level of granularity of algorithm parametrization, algorithm ranking or combination, model validation, and performance measures?

3. Feature generation and selection to represent a problem at Meta-level:

    i. The traditional MF generation approaches which are usually specialized for algorithm recommendation task would be adequate to represent three new proposed areas of the system at Meta-level or based on the complexity of the new problems a different representation would be required?

    ii. In a non-stationary environment, the target variable would not be available at the time of algorithm selection at Meta-level. It will restrict the computing of some important MFs, e.g., the correlation between target and predictors. What would be the impact of the absence of these significant features on the performance of MLL and in later stage how MK database will be updated when the target variable will be known?

4. Representation and storage of dynamically growing complex Meta-Knowledge database:

    i. What level of granularity would be required for the appropriate representation of a problem? For instance, the target variable of the MEs would be only the best learning algorithm, ranking, algorithm parametrization or combination?

    ii. What type of performance measures will be stored in MK database for three different areas, e.g., accuracy, run-time speed? For instance, the run-time speed measure might be useful particularly for a non-stationary environment which helps to identify accurate as well as an efficient learning algorithm.

5. Meta-level Learning strategy for algorithm and its hyper-parameter recommendation:

   i. What strategies and algorithms would be used at Meta-level to efficiently search the target objectives of the mentioned three areas from MK database?

   ii. If MLL process recommends a different learning algorithm and its parametrization for the target concept then what would be the strategy of replacing the current algorithm and how this change would impact the overall performance of the system?

From the above five research questions, 3 and 5 are addressed in this research.

2.8 Problem Formulation

This section formulates the problem of selecting an appropriate algorithm for a given task and finding its configuration leads to the best results. It is considered as a crucial step towards the automation of ML pipeline (Feurer et al., 2015). Although Model Selection and Hyper-parameters Optimization (MSHPO) are conceptually different areas, however, they are linked with each other. Since the selection of an appropriate algorithm with a poor choice of configuration, for a given task, leads to low accuracy. The following formulation is applicable to classification problems, however, it can be extended to the regression tasks as well.

An example of a classification task is represented as a pair (x, y), x is a vector of feature values whereas y is its corresponding class. A dataset $\mathcal{D}$, expended in Equation 2.1, is a set of examples which is consumed by a classification algorithm. A classification algorithm C is a function that learns patterns from $\mathcal{D}$ and apply it on hold-out instances from their feature values $\bar{x}$ to predict the class $\bar{y}$ as shown in Equation 2.2. Moreover the equivalent Meta-RL representation is shown in Equation 2.3 where s is a set of states and a is set of finite actions.

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\} \tag{2.1}$$

$$\text{Supervised Meta-learning } C : \{\mathcal{D}, \bar{x}\} \rightarrow \{\bar{y}\} \tag{2.2}$$

$$\text{Meta-Reinforcement Learning } C : \{\mathcal{D}, s\} \rightarrow \{a\} \tag{2.3}$$

A set of all the possible classification algorithms is represented as $\mathcal{C} = \{C^1, C^2, ..., C^k\}$. A classification algorithm $\mathcal{C}$ requires a set of hyper-parameters $\mathcal{P}_c$ where λ_c is a configuration of the hyper-parameter, $\lambda_c \in \mathcal{P}_c$. The set of hyper-parameters for the i^{th} algorithm in $\mathcal{C}$ denoted by $\boldsymbol{\lambda}^i = (\lambda_a^i, \lambda_b^i, \lambda_c^i, ...)$. This set of all the possible values is represented by Λ^i that $\boldsymbol{\lambda}^i$ can take. A realization of classification algorithm C for a specific configuration $\boldsymbol{\lambda}$ is known as a classification model (C_λ). The error function $\mathcal{E}$ of the classification model C_λ on held-out instances is computed as shown in Equation 2.4.

$$\mathcal{E} : C^i \in \mathcal{C}, \boldsymbol{\lambda}^i \in \Lambda^i \tag{2.4}$$

The feature values of the instances x is used to train an algorithm which is applied on the feature values of $\bar{x}$ to predict its class $\bar{y}$. Based on the underlying distribution of the trained model the instances with similar feature values tend to belong to the same class. It formulates the MSHPO problem as:

$$C^*_{\boldsymbol{\lambda}^*} = \underset{C^i \in \mathcal{C}, \boldsymbol{\lambda}^i \in \Lambda^i}{\operatorname{argmin}} \; \mathcal{E}(C^i_{\boldsymbol{\lambda}^i}, \mathcal{D}) \tag{2.5}$$

Equation 2.5 chose an algorithm and associated configuration that obtain optimized performance at predicting labels on the given task. This equation only defines the structure and general behaviour of the different components of the optimization process and not the scoring function and other details. Furthermore, the assumption that a single model and its configuration $C^*_{\lambda^*}$ is significantly better than the rest of the candidates can not be guaranteed.

Chapter 3

Cross-domain Meta-learning for Time-series Forecasting

In accordance with the research challenges identified in the previous chapter, a thorough study has been conducted to evaluate whether the Meta-knowledge (MK) of a specific domain can be applied on the problems of other domains to find the best learning algorithm. The previous work on Meta-level Learning (MLL) for Time-series (TS) forecasting resulted in Lemke and Gabrys (2010a) and Lemke and Gabrys (2010b). The use of proposed MLL approaches and data from NN3 and NN5 competitions in Lemke and Gabrys (2010a) and supplementing the available NN-GC1 data has led to our research group's[1] winning of the NN-GC1 forecasting competition. In Lemke and Gabrys (2010b) it was stipulated (though not verified by any further analysis) that a particularly good predictive performance resulting from deploying the MLL approach and a Meta-ranking algorithm on the NNGC-C dataset (monthly interval) and NNGC-E (daily interval) might have been due to additional use of the NN3 and NN5 (111 daily series each) datasets for generating MK and training Meta-learners. In this chapter the concentration is on attempting to understand if indeed the use of additional time-series from NN3 and NN5 competitions have been the main reason for the best performance of the MLL on series NNGC-C and NNGC-E of the NN-GC1 competition. Through an extended analysis of the results describing for which NN-GC1 time-series the MLL performs best or worst. Also an attempt has been made to answer a more general question of when and under what circumstances the use of datasets from other domains (NN3 and NN5 competitions in current context) could be beneficial for recommending well-performing forecasting methods for a problem at hand (using MLL approaches on 6 different NN-GC1 datasets in this work).

The key focus would be on finding evidence that revolves around the following questions:

1. Whether the use of additional training data has been the main reason for the best performance of the MLL?

2. Whether the use of data from different domain could be beneficial for recommending well-performing forecasting methods for a problem at hand?

More investigations have been required to find the evidence whether NNGC-C and NNGC-E performed well on NN3 and NN5 Meta-model because of the similar frequency

[1]Smart Technology Research Centre

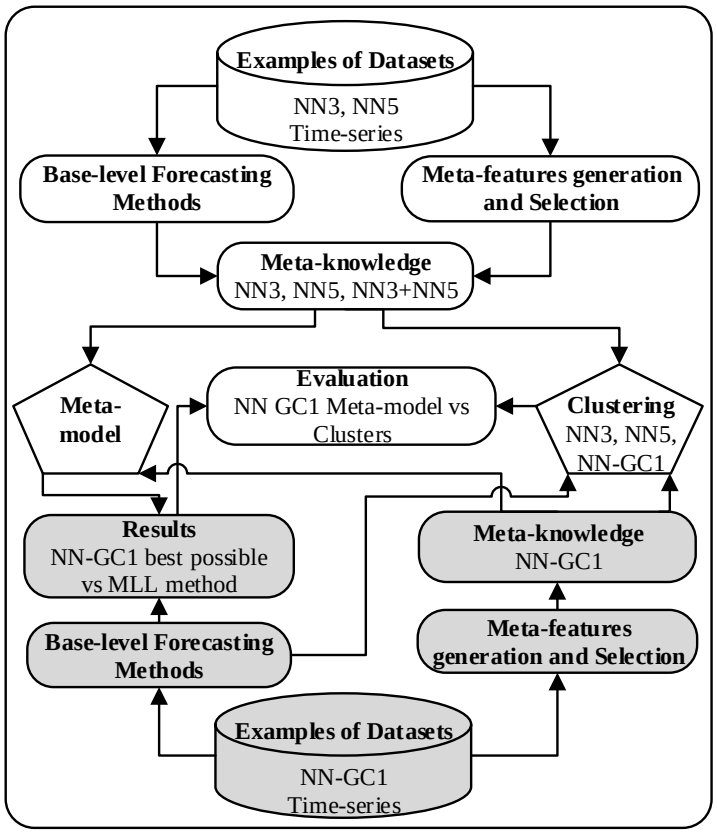

Figure 3.1: Methodology of Cross-domain MLL

of observation recording or the Meta-level problem representation is tilted more towards time-series sample-rate characteristics than the others? This could be a reason that MLL performed well for only time-series datasets with similar frequency. On the other hand, investigation is required to analyze whether increasing the size of training dataset by adding data from different domains could enhance overall MLL algorithm prediction accuracy? Additionally, it leads to another problem of not finding the significant amount of patterns from the cross-domain data, for example, NN3 and NN5 contain 222 instances which is a relatively small number with a lot of variations in the data. It raises the question of whether adding data from only the same domain can enhance Meta-level accuracy?

3.1 Methodology

To examine the questions stated in the above section an experimentation environment has been established containing key components required by an MLL system. Figure 3.1 provides a high-level overview of the MLL system setup for this work. Apart from MLL system a cluster analysis has been performed on MK. The results of both the systems are correlated to find evidence that could lead to the answers of the questions raised in the above section.

The MLL system is divided into two phases; i) Meta-modelling, ii) Meta-ranking. For Meta-modelling two datasets, NN3 and NN5, are used from different domains, empirical business observations and cash machine transactions. Several Meta-features (MFs) and performance measures are computed from these datasets. These performance measures

are mapped with features of each time-series to build an MK for the both datasets and a combined NN3+NN5 MK has been built. There are three different Meta-models built against these MK.

These Meta-models have been evaluated in Meta-ranking phase against six datasets of NN-GC1 which are from a different domain (i.e., transportation) than NN3 and NN5. Furthermore, NN-GC1 has different observation sampling rates. The same MFs which are used in Meta-modelling phase, have been extracted from NN-GC1 for Meta-ranking. The Meta-models, that are trained on NN3 and NN5, are used to estimate the most appropriate forecasting method on the Meta-examples of NN-GC1. These estimates are evaluated against the best possible forecasting method which is computed by evaluating base-learners as NN-GC1. Figure 3.1 provides an overview of the cross-domain MLL system.

Apart from Meta-modelling, Cluster Analysis has been performed on three different combinations of MK including NN3 versus NN-GC1, NN5 versus NN-GC1 and NN3+NN5 versus NN-GC1. A hierarchical approach is applied with different link methods and distance similarity measures to extract most appropriate clusters on the mentioned three combinations of MK.

3.2 Experimentation Environment

An experimentation environment comprises of all the key components of an MLL system and it has been designed to perform an extensive set of experiments. The MFs, and Base-level forecasting and MLL methods are taken from Lemke and Gabrys (2010a). Additionally, a cluster analysis component is added in the environment to perform unsupervised MLL. The base-level forecasting algorithms and MFs used in this work are taken from Lemke and Gabrys (2010a).

3.2.1 Examples of Datasets

The Examples of Datasets (EoD) is a repository of usually large number of datasets from various domains. In this chapter, the EoD consists of 222 univariate time-series from two different sources, NN3 (Crone, 2006) and NN5 (Crone, 2008) competitions. Each data-source contains 111 series whereas NN3 dataset has monthly empirical business observations while NN5 has daily cash machine withdrawals observations. These two data-sources have been used to train the Meta-model.

The Meta-models have been tested on six NN-GC1 (Crone, 2010) competition data-sources consisting of 66 univariate time-series. Each data-source consists of 11 series with different frequency of observations and prediction horizon. Table 3.1 shows the number of time-series, their frequency and horizon of all the above mentioned datasets.

3.2.2 Base-level Forecasting Methods

Performance of four Base-level forecasting methods has been estimated against each of the time-series. Those algorithms vary from simple, such as Moving Average (MA), to more complex algorithms, including Automatic Box-Jenkins, structural and Neural Networks.

Table 3.1: NN3, NN5 and NN-GC1 datasets which are used to build Meta-modelling and its evaluation

Datasets	**Series**	**Observations**	**Frequency**	**Horizon**
NN datasets used for Meta-Modelling				
NN3	111	52-126	Monthly	18
NN5	111	735	Daily	56
NN-GC1 datasets used for model evaluation				
NNGC-A	11	23-37	Yearly	0
NNGC-B	11	31-148	Quarterly	4
NNGC-C	11	48-228	Monthly	12
NNGC-D	11	527-1181	Weekly	52
NNGC-E	11	377-747	Daily	7
NNGC-F	11	902-1742	Hourly	24

The algorithms are evaluated using Symmetric Mean Absolute Percentage Error (SMAPE) and Standard Deviation (StdDev) measures. The evaluation protocol consisted of training the models on 75% of the series and testing on the remaining 25% of instances. Table 3.3 shows SMAPES and StdDev of NN3 and NN5 datasets. Several R libraries have been used to compute the performance measures (R Development Core Team, 2008), where the configuration of the algorithms is given in the following subsections.

3.2.2.1 Simple time-series Algorithms

MA is a simple time-series method where the arithmetic mean of the last k observations has been computed iteratively, see Equations 3.1. The optimal value of k is selected using grid-search from 3 to 24 where the step size is 3. At each value of k, mean squared error (MSE) has been calculated on the validation-set and the k is selected where the error value is lowest.

$$\hat{y}_{t+1} = \frac{1}{k} \sum_{i=t-k+1}^{t} y_i \tag{3.1}$$

3.2.2.2 Complex time-series Algorithms

Additionally, three complex time-series algorithms are used as Base-learner:

1. Auto-regressive Integrated Moving Average (ARIMA) fall under the complex time-series forecasting techniques (Box and Jenkins, 1970). The configuration of ARIMA used in this work was obtained by performing a grid-search over possible models within the first and second differences with starting stepwise value of 1 (Hyndman and Kh, 2008). The lag value that produced the lowest MSE on the validation-set has been automatically selected. The reason for selecting maximum second-order difference as described in Lemke and Gabrys (2010a) is that the data usually only involves non-stationarity at maximum second-level.

2. Structural technique is a linear state-space model for univariate time-series based on various components of the series such as trends, etc. (Petris and Petrone, 2011). The maximum likelihood estimates of the local level model is used to get the time-varying slope

dynamics. The structural technique produces fitted Kalman filter and smoother (Tusell, 2011).

3. An iterative version of feed-forward Neural Network (NN) has been used. The network is configured with a single hidden layer, 12 neurons and up to a lag of 12 observations to reflect weekly or yearly seasonality for NN5 and NN3 respectively. The predictions have been averaged over ten trained networks to obtain the final forecasts.

Table 3.2 depicts various parameters that are used by the base-learning forecasting methods.

Table 3.2: Methods and their configurations that are used to compute performance measures

Methods	**Parameter**	**Description**	**Value**
MA	k	Number of observations	3-24
ARIMA	maxQ	Maximum number of order difference	2
Structural	type	Maximum likelihood estimates	level
NN	neurons and seasonality	number of neurons in hidden layer and lag	12 and 12

The SMAPE and StdDev of base learners are reported in Table 3.3. The MA and ARIMA consistently performed well for NN3 and NN5 respectively where MA comes out as the best algorithm for 41% of time-series in NN3 and ARIMA outperformed the other candidates in 72% of time-series. There is not much difference in overall SMAPES and StdDev of remaining three candidate algorithms where StdDev of NN3 is almost double that of NN5 dataset. The reason for high-StdDev of NN3 dataset is the high variations in comparatively short time-series, which make the dataset less stable than the NN5 dataset.

Table 3.3: SMAPE and StdDev of Base-level forecasting methods

Method	**NN3 SMAPE**	**NN3 σ**	**NN5 SMAPE**	**NN5 σ**
MA	**15.68**	**14.74**	35.17	7.73
ARIMA	18.83	15.88	**28.02**	**8.17**
Structural	17.57	15.35	36.09	9.05
NN	17.05	13.56	34.89	7.37

3.2.3 Meta-feature Generation

There are three different groups of MFs extracted from univariate time-series which include descriptive statistics, frequency domain and auto-correlation (Lemke and Gabrys, 2010a). These features have been computed using grid searched parameters of the methods that are available in R (R Development Core Team, 2008). Table 3.4 contains the list of features and their descriptions that have been extracted from time-series.

3.2.3.1 Descriptive Statistics

The descriptive statistics have been computed on detrended time-series using polynomial regression as mentioned in Lemke and Gabrys (2010a). Statistics that are computed using detrended series include StdDev, skewness and kurtosis. Another feature, *trend*, has been

Table 3.4: List of MFs and their descriptions

Features	Description	Formalisation
Descriptive Statistics		
std	StdDev of de-trended series	$detrend = detrend(polyfit(series, 3))$ $std(detrend\_series)$
skew	Skewness of series	$skew(detrend\_series)$
kurt	Kurtosis of series	$kurt(detrend\_series)$
length	Length of series	$length(series)$
trend	trended series	$std(series)/std(detrend\_series)$
turn	Turning points	$count(y_{i-k} > ... > y_i, y_i < ... < y_{i+m})$
step	Step changes	$count(\|y_i - \mu(y_1..y_{i-1})\| > 2\sigma(y_1...y_{i-1}))$ where y_i is an observation of a series
non-lin	Non-linearity measure	$lin = lm(detrend\_series)$ $nonLin = lm(poly(detrend\_series, 2))$ $isSignificant(anova(lin, nonLin))$
Frequency Domain		
maxSpec	Power spectrum: maximal value	$spect = fffta(detrend\_series)$ $maxSpec = max(spect.spectrum)$
ff	No. of peaks not lower than 60% of the max	$length(spect[spect >= maxSpec * 0.6])$
Auto-correlation		
acf[1, 2]	Auto-correlations at lags one and two	$acf = acf(series)$ $acf[1], acf[2]$
pacf[1, 2]	Partial auto-correlations at lags one and two	$pacf = pacf(series)$ $pacf[1], pacf[2]$
season	Seasonality measure	$pacf[12]$ for NN3, $pacf[7]$ for NN5

calculated to add the variability of time-series in the feature set. The turning points and step changes of time-series have been computed as described by Shah (1997). The turning points provide information of local minima or maxima within a series while a step change is detected when the mean of the series is greater than twice the StdDev at each observation of the series. The number of turning points and step changes have been cumulated within a series and normalized by the number of observations in a time-series. Furthermore, the Durbin-Watson test and non-linearity measure has been calculated on polynomial regression of order three (Lemke and Gabrys, 2010a).

3.2.3.2 Frequency Domain and Autocorrelations

In frequency domain, two features of the Fast Fourier Transform (FFT) have been extracted from the detrended time-series which include the maximum value of power spectrum and number of peaks greater than 60%. The maximum value of the power spectrum provides the strength of the strongest seasonal or cyclic component. While the top 40% of peaks in the power spectrum identifies the number of times strong recurring components are found in a time-series (Lemke and Gabrys, 2010a).

There are five autocorrelation and partial autocorrelation features which are part of MFs to capture information of stationarity and seasonality of a time-series. These correlations are computed for lags 1 and 2. Additionally, the seasonality introduced partial autocorrelation of lag 12 for the NN3 dataset which has the monthly frequency of occurrence and partial autocorrelation of lag 7 for the NN5 consisting data of weekly frequency.

3.2.4 Meta-knowledge Preparation

MLL requires an extensive and diverse set of data to build a reliable knowledge-base on a given problem domain. The MK is composed of MFs mapped with the performance measures of respective EoD. The performance measures used to evaluate four base-models is SMAPE whereas only lowest SMAPE against every EoD is selected as the target variable of the MK. The size of the MK is varied for different experiments ranging from 111 to 288 instances with 15 MFs which are too many for small datasets. In order to reduce the dimensionality, a Random Forests (RF) based feature extraction method is applied to the MK. This method computes the importance of each feature which is listed in Table 3.5. The features are further divided into three sets including the top three most important features, features with importance greater than the mean importance and full dataset. The RF based scoring method is used as described in Genuer et al. (2010). This method builds a tree and computes the amount of impurity that each feature decreases. The more a feature decreases the impurity, the more important the feature is ranked. In general, the impurity decrease from each feature is averaged across trees to determine the final importance of the variable.

Table 3.5: MFs Importance

NN3		**NN5**		**NN3-NN5**	
Features	**Imp.**	**Features**	**Imp.**	**Features**	**Imp.**
season	1.00	kurt	1.00	season	1.00
turn	0.85	season	0.96	turn	0.76
acf2	0.79	trend	0.90	trend	0.76
trend	0.72	step	0.89	pacf2	0.73
pacf2	0.72	pacf2	0.83	acf2	0.72
kurt	0.70	nonlin	0.81	pacf1	0.71
skew	0.70	turn	0.79	acf1	0.70
pacf1	0.70	maxSpec	0.79	kurt	0.68
acf1	0.67	std	0.79	nonlin	0.68
step	0.66	ff	0.76	skew	0.67
std	0.65	skew	0.76	std	0.66
maxSpec	0.65	acf1	0.74	step	0.62
nonlin	0.64	pacf1	0.74	maxSpec	0.62
ff	0.46	acf2	0.74	ff	0.51
length	0.41	length	0.00	length	0.40

The MK for both NN3 and NN5 are biased towards MA and ARIMA respectively which leads to imbalanced dataset problem. In the current scenario the imbalanced MK is producing biased classifiers that have a higher estimation accuracy for the majority classes, i.e., MA and ARIMA, but lower accuracy for the minority classes. This problem has been solved using Synthetic Minority Over-sampling TEchnique (SMOTE) which balances the dataset by over-sampling the minority classes. SMOTE synthetically generates more instances of the minority class by broadening their decision regions (Chawla et al., 2002). The proportion of class instances of raw and datasets balanced using SMOTE are shown in Table 3.6

Table 3.6: Proportion Raw and balanced classes

Target	**NN3**		**NN5**	
Method	**Raw**	**Balanced**	**Raw**	**Balanced**
MA	42.34(%)	25.82(%)	5.41(%)	25.00(%)
ARIMA	9.91(%)	24.73(%)	72.97(%)	25.31(%)
Structural	24.32(%)	25.27(%)	8.11(%)	25.31(%)
NN	23.42(%)	24.18(%)	13.51(%)	24.38(%)

3.2.5 Meta-learning

The MK dataset contains predictors (MFs) and class labels (most promising forecasting algorithm) which makes it a classification task. Three supervised learning algorithms have been used as Meta-learners: Neural Network (NN), Decision Trees (DT) and Support Vector Machines (SVM). These Meta-learners are using leave-one-out cross validation training strategy. The methods have been evaluated and compared using SMAPE and classification accuracy. Following is the configuration used for Meta-learners:

1. Feed-forward NN is used with six different number of neurons in the hidden layer = 10, 15, 20, 25, 30, 35, 40 and weight decay = 0.01.

2. DT C5.0 with trials, number of boosting iterations, from 1 to 100.

3. SVM Radial-basis Function (RBF) kernel with sigma = 0.05, 0.01, 0.1 and cost = 30, 35, 40, 45, 50, 55, 60, 65, 70.

The overall accuracies and StdDev of the above Meta-learners are recorded in Table 3.7. Figure 3.2 shows the number of times a particular base and Meta-learner performs best for NN3, NN5 and combined NN3+NN5 data respectively. These accuracy estimates have been computed using the predicted method recommended by Meta-learner and the best algorithm out of four candidate time-series forecasting methods for each time-series. There were three different experiments performed at Meta-level where the number of predictors were varied:

1. In the first experiment three most important features have been used. The SVM Meta-learner performed slightly better than the others on NN3 and NN5. Whereas DT outperformed the remaining two Meta-learners on combined NN3+NN5 dataset.

2. In another experiment, the features with above average importance have been selected. DT and SVM are found to be consistently dominating Meta-learners for all three datasets.

3. All the features are used in the last experiment where NN performed well for simpler time-series, NN3, while SVM outperformed the remaining for complex datasets including NN5 and combined NN3+NN5.

From the above experiments the Meta-learner that outperformed all others have been chosen to predict the best forecasting method for datasets from different domains.

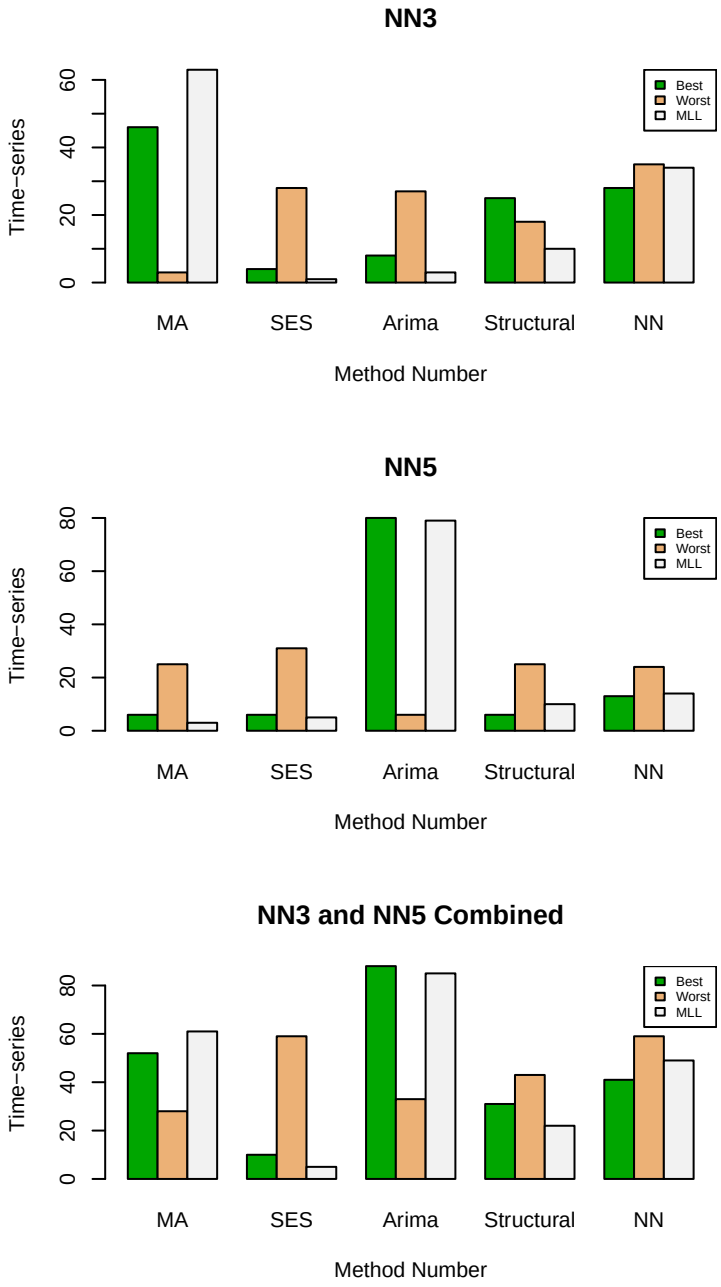

Figure 3.2: Histogram showing number of times a particular base and Meta-learner performs best for NN3, NN5 and combined NN3+NN5 data

Table 3.7: SMAPE (and Accuracy) of various Meta-learners

Method	NN3	NN5	NN3+NN5
Top three important features			
NN	49.54 (20.98)	72.83 (8.63)	61.26 (14.06)
DT	52.25 (20.97)	72.97 (7.84)	**63.06 (13.18)**
SVM	54.05 (21.52)	74.77 (7.74)	56.76 (16.05)
Above average important features			
NN	52.25 (20.06)	60.36 (12.46)	57.20 (16.61)
DT	53.15 (20.79)	74.77 (7.48)	62.61 (13.96)
SVM	49.02 (21.85)	**76.58 (7.38)**	59.00 (16.90)
All features			
NN	**56.76 (17.47)**	71.17 (9.40)	62.16 (15.74)
DT	51.35 (21.87)	71.87 (9.13)	59.46 (15.87)
SVM	52.25 (22.43)	75.68 (7.82)	62.16 (14.70)

3.2.6 Cluster Analysis

The hierarchical cluster analysis has been performed on MK to further analyze whether there is any correlation between high MLL accuracy and homogeneous clusters. This analysis can validate whether MLL works on the new domain. Atleast for the high-performing series of NN-GC1 at Meta-level which are also clustered with NN, it can be considered that MLL does effectively works. There have been different combinations of clustering methods and distance measures experimented to find the most appropriate one on different sets of MK. Four link methods of hierarchical clustering were part of the experiment including ward (Murtagh and Legendre, 2014), and single, complete and average link. These methods have been used in combination with two distance measures: Euclidean and Manhattan. The analysis has been performed for 10 and 20 clusters.

3.3 Results

The Meta-Models have been applied on 66 time-series provided by the NN-GC1 competition Crone (2010) as shown in Table 3.1. These Meta-models have been tested on NN-GC1 dataset where the accuracies are summarized in Table 3.8. The best forecasting method is MA with average SMAPE 16.0 whereas Meta-learner could achieve a small improvement over it. Also by analyzing the results of 6 NN-GC1 series it can be concluded that there is no significant difference in SMAPE and StdDev of Base- versus Meta-learner.

The detailed results of NN-GC1 datasets on various combination of Meta-models as well as three different sets of predictors (based on feature importance) are recorded in Table 3.9. The first part of the table consists of NN-GC Meta-model results whereas the rest of the table represents NN-GC results trained on NN. The top-performing models against each series of NN-GC are represented in bold. Moreover, each column has two bold values, top performing base-learner, and b) top-performing Meta-model trained on NN. The average SMAPE of the best possible base-level forecasting algorithm has been compared with different combinations of Meta-models' average SMAPE that came out from various experimentations. In six

Table 3.8: SMAPE (and StdDev) of NN-GC1 series

Dataset	**Base-learning**		**MLL**	
NN-GC1	**Method**	**SMAPE (StD.)**	**Method**	**SMAPE (StD.)**
NNGC-A	Structural	**7.8 (4.7)**	SVM MLL(NN5)→GC [All features]	**7.8 (4.7)**
NNGC-B	MA	**7.2 (3.6)**	DT MLL(NN5)→GC [Top 3 features]	7.2 (4.8)
NNGC-C	NN	13.4 (9.5)	NN MLL(NN3)→GC [Above Average features]	**12.5 (9.8)**
NNGC-D	MA	**9.5 (9.1)**	SVM MLL(NN3)→GC [All features]	10.0 (9.0)
NNGC-E	MA	26.9 (20.8)	NN MLL(NN3)→GC [All features]	**26.6 (20.7)**
NNGC-F	MA	60.1 (5.8)	NN MLL(NN3)→GC [All features]	**59.8 (5.6)**
Average	MA	16.0	NN MLL(NN3)→GC [Above Average features]	**15.9**

out of nine cases MLL(NN3)→GC (NN3 Meta-model on NN-GC1) came out as the best Meta-model whereas in remaining four cases combination of both NN3 and NN5 datasets MLL(NN3+NN5)→GC outperformed the rest. Whereas by analyzing the average SMAPE of different experiments, it is found that NN Meta-learner performed reasonably well on the set of features whose importance is above average followed by SVM and DT. At the deeper level the six NN-GC1 are analyzed, apart from a few cases, there was no significant difference between the best possible base-learning and MLL SMAPE.

The individual NN-GC1 time-series were further diagnosed for those series which are unable to show the minimum error at Meta-level. The analysis showed that 44% of the series recommended by the MLL were ranked as second best followed by 24% series ranked as third best, whereas only 2% series were ranked as the worst. Overall 70% time-series are reported better than the average SMAPE. Overall, the difference between MA base-learner (which performed the best among others) and best possible Base-level Learning (BLL) score gives very little room for improvement which is a problem to show MLL significance. However MLL fall between these two and can be used for recommendation of the predictive algorithm with a minimum probability that a bad predictor will be recommended.

Figure 3.3 shows the histograms with the number of times a particular method performed best at Base and Meta-level for NN-GC1 time-series. The NNGC-A, NNGC-B, NNGC-C and NNGC-F are showing the mixed base-level class distribution where NNGC-D and NNGC-E are biased towards MA and NN respectively. However, for NNGC-D and NNGC-E datasets MLL recommends ARIMA for most of the time-series.

3.4 Analysis

Various experiments have been performed to investigate the reported MLL results in detail. The reliability of MK is further analyzed for the three sets of MFs which were formed based on the importance of the features. There was not much accuracy variation found among these three sets which indicate that not all the MFs are contributing in Meta-modelling. The knowledge representation can be improved by increasing Examples of Datasets (EoD) which are the source of computing MFs because one of the challenges in this work is scarce input data with large variations within different time-series of a dataset. In particular, NN5 dataset is containing only 111 time-series even within this small number few subsets were representing different trends and patterns which made it difficult to build a stable Meta-model.

Table 3.9: SMAPE (and StdDev) of NN-GC1 series

Method	NNGC-A	NNGC-B	NNGC-C	NNGC-D	NNGC-E	NNGC-F	Average
MA	12.8 (9.4)	**7.2 (3.6)**	14.6 (11.9)	**9.5 (9.1)**	**26.9 (20.8)**	**60.1 (5.8)**	**16.0**
ARIMA	10.1 (6.9)	8.6 (4.7)	16.2 (19.1)	14.0 (9.7)	42.4 (52.0)	62.2 (6.5)	21.0
Structural	**7.8 (4.7)**	7.6 (4.7)	24.1 (25.8)	14.1 (9.5)	30.4 (23.3)	84.3 (16.1)	21.0
NN	13.1 (6.7)	14.9 (6.9)	**13.4 (9.5)**	15.8 (12.6)	39.2 (30.1)	61.4 (5.7)	19.1
Base Learning (Best Possible)	6.5 (4.6)	6.0 (3.6)	11.9 (9.8)	9.4 (8.9)	25.5 (20.0)	59.1 (5.8)	14.3
MLL on top 3 MFs							
NN Meta-Model							
MLL(NN3)→GC	11.3 (8.0)	8.1 (3.9)	14.3 (11.6)	12.4 (9.9)	28.6 (23.0)	61.8 (6.5)	16.6
MLL(NN5)→GC	9.9 (6.7)	8.3 (5.4)	15.3 (12.3)	13.3 (8.9)	43.6 (51.2)	75.1 (12.4)	21.9
MLL(NN3+NN5)→GC	11.3 (8.3)	7.8 (4.5)	14.8 (12.2)	12.6 (9.9)	40.5 (47.9)	63.7 (9.6)	20.3
DT Meta-Model							
MLL(NN3)→GC	12.6 (8.1)	7.4 (3.8)	18.4 (21.4)	10.4 (9.3)	28.0 (22.2)	63.5 (9.3)	17.9
MLL(NN5)→GC	10.1 (7.2)	**7.2 (4.8)**	15.2 (12.4)	10.2 (9.0)	36.8 (47.3)	61.8 (6.6)	19.0
MLL(NN3+NN5)→GC	11.5 (7.5)	7.9 (4.4)	17.5 (19.1)	10.8 (9.5)	28.4 (23.0)	63.9 (9.5)	17.8
SVM Meta-Model							
MLL(NN3)→GC	12.6 (8.1)	8.3 (5.7)	14.4 (11.6)	**10.0 (9.0)**	27.8 (21.5)	60.4 (5.9)	16.3
MLL(NN5)→GC	12.0 (8.2)	8.1 (4.7)	16.8 (19.2)	11.6 (9.3)	41.6 (51.8)	61.9 (6.9)	21.0
MLL(NN3+NN5)→GC	11.1 (8.3)	7.9 (4.4)	14.0 (11.7)	10.3 (9.2)	27.3 (22.4)	60.5 (6.2)	16.1
MLL on MFs whose importance is greater than mean							
NN Meta-Model							
MLL(NN3)→GC	10.0 (6.1)	7.7 (4.6)	**12.5 (9.8)**	12.2 (9.4)	28.9 (22.8)	60.9 (5.8)	**15.9**
MLL(NN5)→GC	9.0 (7.6)	8.4 (5.4)	23.6 (26.1)	14.3 (10.3)	30.2 (23.4)	81.5 (16.8)	21.4
MLL(NN3+NN5)→GC	12.5 (8.1)	9.1 (7.3)	13.0 (10.0)	12.0 (12.1)	42.1 (47.9)	64.1 (8.3)	20.5
DT Meta-Model							
MLL(NN3)→GC	12.1 (7.8)	8.5 (5.8)	13.4 (10.4)	10.4 (9.3)	27.5 (21.8)	61.2 (6.0)	16.2
MLL(NN5)→GC	10.1 (7.2)	7.4 (4.7)	15.2 (12.4)	10.2 (9.0)	36.8 (47.3)	61.8 (6.6)	19.1
MLL(NN3+NN5)→GC	11.8 (8.3)	8.0 (4.5)	13.2 (10.1)	12.1 (12.9)	32.8 (25.0)	61.1 (6.3)	17.2
SVM Meta-Model							
MLL(NN3)→GC	12.9 (7.8)	8.3 (5.7)	13.4 (10.4)	10.0 (9.0)	27.7 (22.2)	60.4 (5.9)	16.1
MLL(NN5)→GC	10.6 (7.0)	7.7 (4.7)	16.4 (19.0)	14.0 (9.7)	42.4 (52.0)	62.2 (6.5)	21.0
MLL(NN3+NN5)→GC	12.8 (9.4)	7.9 (4.4)	16.1 (18.5)	10.7 (9.1)	36.6 (47.6)	61.0 (6.3)	20.0
MLL on all the MFs							
NN Meta-Model							
MLL(NN3)→GC	10.5 (7.3)	7.6 (3.9)	17.0 (19.2)	12.1 (10.7)	**26.6 (20.7)**	60.3 (5.8)	16.8
MLL(NN5)→GC	8.4 (5.3)	9.2 (5.1)	23.7 (26.0)	14.4 (10.3)	30.6 (23.2)	84.3 (16.1)	21.4
MLL(NN3+NN5)→GC	10.6 (7.9)	7.4 (4.4)	17.1 (19.0)	11.6 (10.7)	31.8 (25.0)	61.6 (6.7)	17.8
DT Meta-Model							
MLL(NN3)→GC	12.7 (8.0)	8.4 (5.7)	13.7 (11.3)	11.2 (10.6)	36.7 (47.4)	61.2 (6.0)	19.4
MLL(NN5)→GC	10.3 (7.8)	8.3 (4.4)	15.0 (12.6)	10.2 (8.5)	36.8 (47.3)	61.2 (6.5)	19.1
MLL(NN3+NN5)→GC	11.3 (8.0)	8.5 (4.5)	14.0 (11.3)	10.3 (9.3)	36.9 (47.4)	61.0 (6.3)	19.1
SVM Meta-Model							
MLL(NN3)→GC	12.5 (8.3)	8.1 (5.0)	14.8 (11.8)	**10.0 (9.0)**	26.9 (20.8)	**59.8 (5.6)**	16.0
MLL(NN5)→GC	**7.8 (4.7)**	7.6 (4.7)	24.1 (25.8)	14.1 (9.5)	30.4 (23.3)	84.3 (16.1)	21.0
MLL(NN3+NN5)→GC	11.0 (7.3)	7.6 (4.4)	16.3 (19.1)	10.5 (9.5)	36.3 (47.4)	61.3 (5.9)	19.7

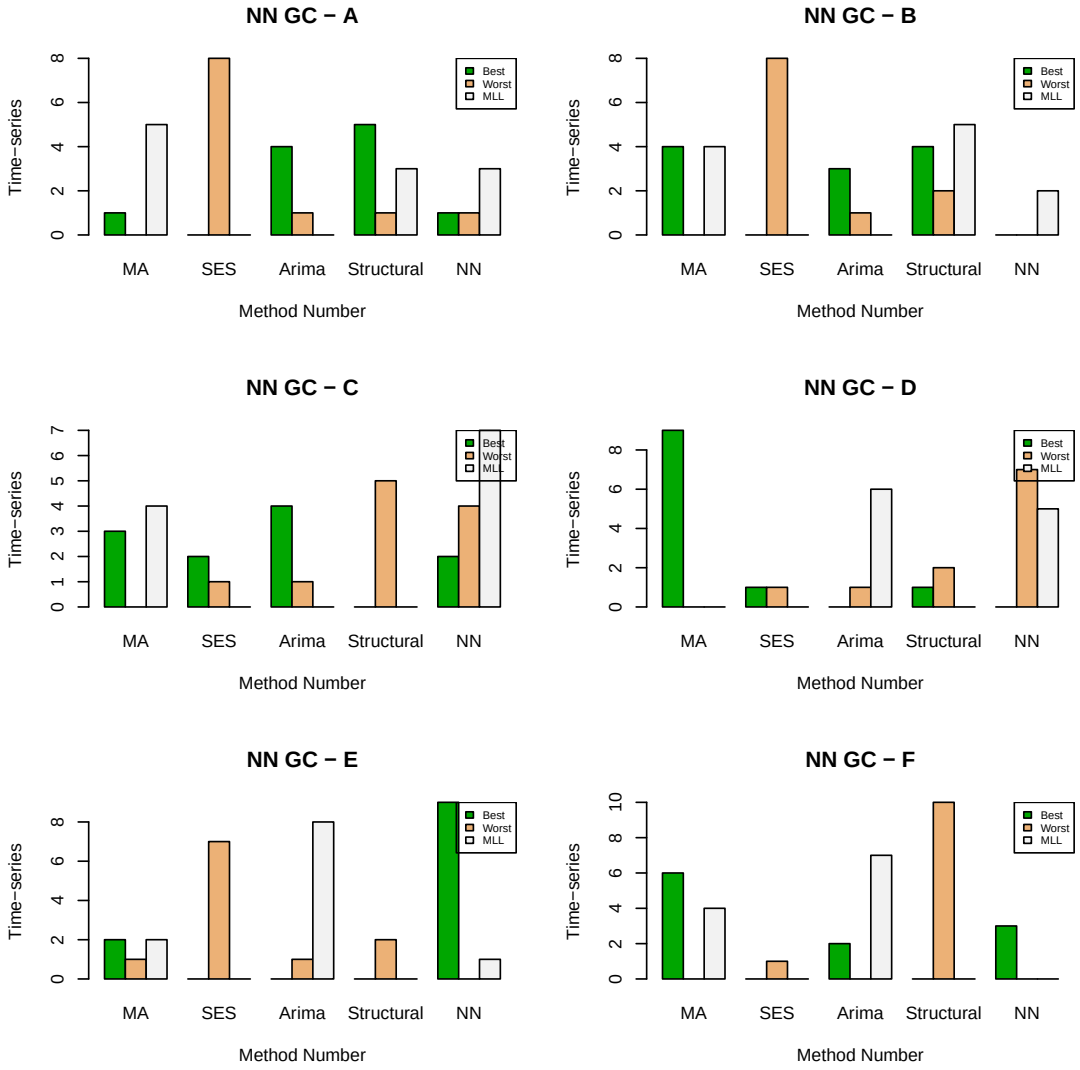

Figure 3.3: Histogram showing number of times a particular method performs best for NN-GC1

In another experiment, the Meta-learner was built using both NN3 and NN5 datasets to analyze whether increasing the MK instances would have any impact on Meta-learner. The combined results are not promising either. The reason is found in their cluster analysis where very few time-series are clustered together based on the similarity of features. Hence the Meta-model was unable to learn significant patterns from cross-domain time-series.

MLL performed reasonably well even in presence of biased class distribution (for NN5 single base-level forecasting method was performing best for 72% of time-series) while applying Meta-model on NN-GC1. Even though NN3 dataset is found to be simpler time-series than NN5 but the overall Meta-level accuracy of NN5 is significantly higher than NN3. The reason is that ARIMA came out as the best base-learning algorithm for 72% of the time-series which is the cause of biased class distribution. However, for NN3 dataset MA is the best algorithm for 41% of the time-series followed by the contribution of NN and Structural methods with more than 20% each. So these evidences suggest that Meta-learner worked well for biased class distribution as compared to mixed class. In this work the MLL se-

lected majority class while predicting the most appropriate algorithm for the cross-domain time-series.

The final experiment was performed to analyze whether any correlation exists between high MLL accuracies versus homogeneous clusters of MFs. There were three different combinations of Meta-examples clustered with the Meta-examples of NN-GC1 which included NN3, NN5 and combined NN3+NN5 data. The clusters of NN3 and NN5 are found to be heterogeneous since there were very few time-series clustered together. In this work, the focus is to find the reasons for the best performance of the MLL on NNGC-C and NNGC-E datasets, however, all the Meta-examples of NN3 are clustered with NNGC-C. In Figure 3.4 it can be observed that the Base- and Meta-learner are same in most of the clusters for NN3 and NNGC-C series. On contrary to this NN5 is not clustered with NNGC-E in most of the cases. It is further analysed by observing combined NN3+NN5 and NN-GC1 clusters. It is noted that except one all the NNGC-E series are clustered with NN3 and NNGC-C.

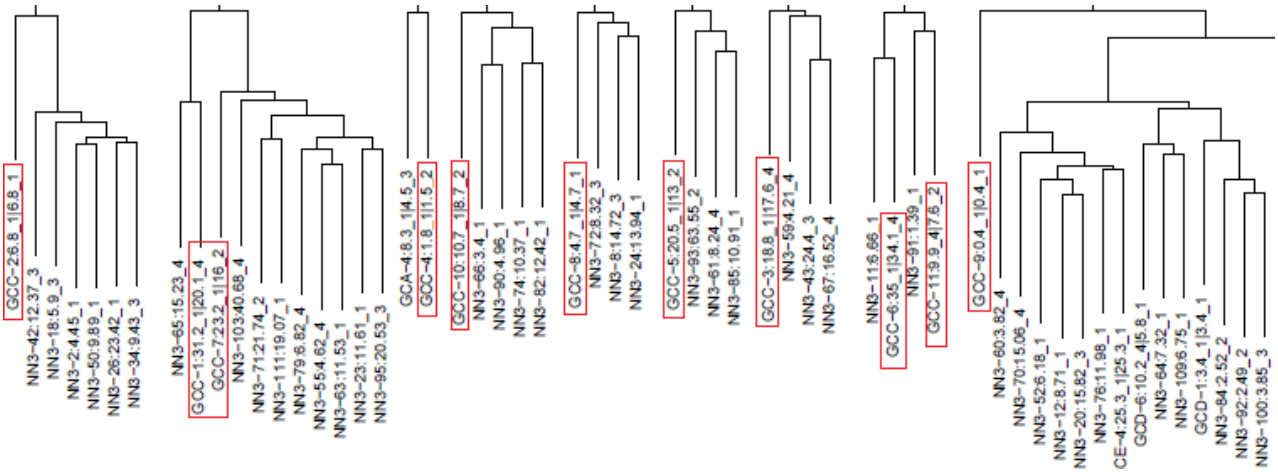

Figure 3.4: NN3 clustered together with NGGC-C dataset where the cluster cut over is at k = 20

3.5 Summary

The MLL is applied to various univariate time-series belonging to different domains. The key focus of this work is to investigate whether the use of additional training data from a different domain is beneficial in order to achieve better MLL performance? This work belongs to the 'learning from the task properties' method of the Automatic Machine Learning (Auto-ML). It addresses model selection and, to some extent, MF generation and selection research challenges, which are defined in Section 2.7, in the context of MLL.

Several experimentations were conducted to find the evidence around those questions which has lead to various challenges including:

1. Very few EoD were available from each domain for Meta-modelling.

2. There are several MFs computed on the time-series which resulted in sparse feature space problem due to very few EoD.

3. The above two problems cause difficulties with training Meta-classifier.

4. Again the test EoD - NN-GC1 contain very few instances with huge variations in the trends and distribution of its six different sub-datasets.

In consideration of the above challenges, the performance of MLL on the cross-domain problem was satisfactory. It is also validated from the clustering of MFs where the Meta-examples grouped together are ranked as best or second best at Meta-level. There were only 2% of cases recommended by MLL that were the worst performing base methods for the respective time-series.

There are a few key observations that can be made from experimentation results and analysis that helped in answering the questions raised in the problem statement:

1. The additional data was not the reason for better MLL performance. In several experimentations, only NN3 Meta-model performed better than the combined NN3+NN5 Meta-model. One of the reasons found from cluster analysis is that both NN3 and NN5 have very few Meta-examples that are similar to each other. Additionally, NN-GC1 has more similarity with only NN3 than the combined NN3+NN5 dataset.

2. Both NNGC-C and NNGC-E are clustered with NN3, unexpectedly there were very few instances of NNGC-E clustered with NN5. It is also verified by MLL where MLL(NN3) consistently performed well for NNGC-E. Here it is hard to say that whether the similar frequency of observation recording was the reason for better MLL accuracy.

In considering several data related challenges the performance of MLL on the cross-domain problem was quite satisfactory. Even though there wasn't much room of improvement for Meta-learner but it made its place between the best possible Base-learning and MA (the best performing base-learner among three others). This study addresses mainly the model selection problem of MLL, particularly, for shallow learning algorithms. It can be enhanced to Deep Neural Networks (DNN) for the same cross-domain knowledge transfer problem.

Chapter 4

Towards Meta-learning of Deep Architectures for Efficient Domain Adaptation

An investigation of the situations in which the use of additional cross-domain data can improve the performance of a Meta-level Learning (MLL) system has been carried out in Chapter 3 with focus on the cross-domain transfer of Meta-knowledge (MK). In this chapter a Hyper-parameter Optimization (HPO) approach to tackle the cross-domain knowledge transfer problem has been proposed. The objective is to identify how many blocks (i.e. groups of consecutive layers) of a pre-trained image classification network need to be fine-tuned based on the characteristics of the new task. In order to investigate it, a number of experiments have been conducted using different pre-trained networks and image datasets. The networks were fine-tuned, starting from the blocks containing the output layers and progressively moving towards the input layer, on various tasks with characteristics different from the original task. The amount of fine-tuning of a pre-trained network (i.e. the number of top layers requiring adaptation) is usually dependent on the complexity, size and domain similarity of the original and new tasks. Considering these characteristics, a question arises of how many blocks of the network need to be fine-tuned to get maximum possible accuracy? Which from the number of available pre-trained networks require fine-tuning of the minimum number of blocks to achieve this accuracy? The experiments, that involve three network architectures each divided into 10 blocks on average and five datasets, empirically confirm the intuition that there exists a relationship between the similarity of the original and new tasks and the depth of network needed to fine-tune in order to achieve accuracy comparable with that of a model trained from scratch.

4.1 Methodology

In order to carry out the investigations, a platform has been implemented to conduct experiments with different combinations of pre-trained networks, their hyper-parameters and image datasets. The experiments have been designed to investigate the relationships among these three key components while fine-tuning the pre-trained networks on new tasks. There

are several characteristics which can be considered but the two most important features selected for this study are the size and similarity of the new task. The four Transfer Learning (TL) scenarios are based on these two features. A schematic view of transfer learning is shown in Figure 4.1 where Task-A is representing the original problem and Task-B the new problem datasets.

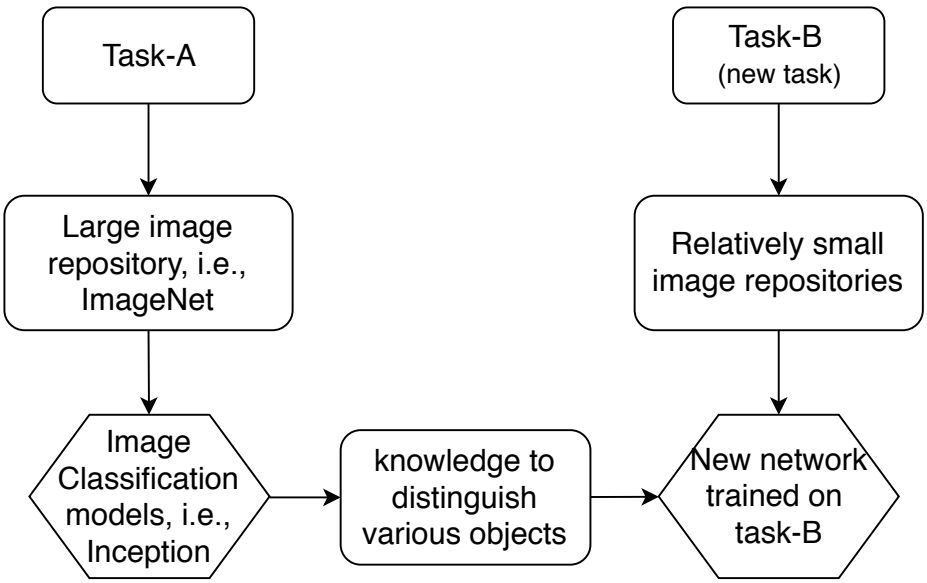

Figure 4.1: Schematic diagram of transfer learning

Despite the popularity of TL in computer vision, there is no principled way of finding the relation between characteristics of a dataset and depth of the network that needs to be re-trained. In this work, an effort has been made to find this relationship by identifying a pre-trained network where the minimum number of blocks need to be re-trained to achieve state-of-the-art accuracy. Moreover, instead of learning the general characteristics of the dataset which is usually practiced in shallow learning, e.g. feature statistics Ali et al. (2018), a higher level characteristics have been pursued, such as layer activations. The focus of the experiments was to investigate the following key scenarios:

1. If Task-B is small in size and similar to Task-A (e.g. both tasks are concerned with natural images), re-training of the entire network might lead to over-fitting. The higher-level features of the pre-trained network, Task-A, are usually relevant for Task-B. Hence, the re-training of a single or a few final layer(s) becomes very effective.

2. If Task-B is large and similar to Task-A, there is less possibility of over-fitting while fine-tuning more layers of the network.

3. If Task-B is small but less similar to Task-A, there is a possibility that Task-A does not contain relevant features for Task-B. In this case TL might not be very useful, however, re-training of final layers might give reasonable results.

4. If Task-B is large and very different from Task-A, both the training of the network from scratch and initialization of the network with the weights of the pre-trained model would be beneficial.

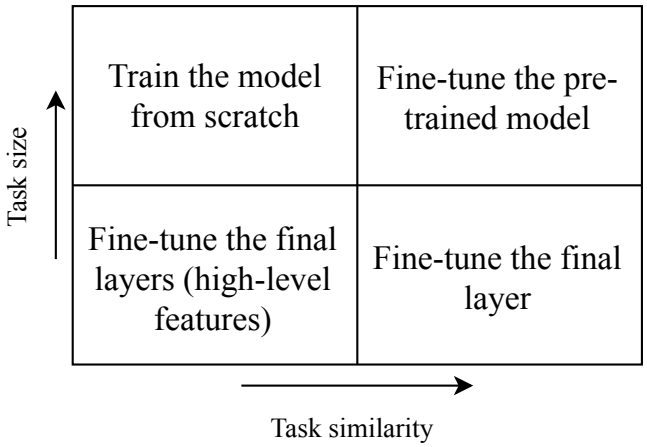

Figure 4.2: Transfer learning scenarios

Figure 4.2 is summarising the above four scenarios.

Datasets with appropriate characteristics have been gathered for the experiments to cover the above four scenarios. The network architectures used in this study vary greatly, hence the groups of layers are fine-tuned rather than individual layers. Please refer to Figure 4.3 for more details on how the layers of each architecture have been grouped into what is referred to as 'blocks'. Moreover, the source of the Inception-v3, Inception-ResNet-v2 and VGG-19 baseline architectures, without block definition, are Szegedy et al. (2015), Szegedy et al. (2017) and Simonyan and Zisserman (2014) respectively.

The pre-trained networks have been fine-tuned on each of the new tasks. The experimental approach was to fine-tune an iteratively increasing number of blocks of each network, starting from the final block, while the lower blocks of the network act as a fixed feature extractor for Task-B. The train and test accuracies have been recorded on every iteration. In some cases, where Task-B is similar, the re-training of only the final layer produces close to the state-of-the-art accuracy. On the contrary, it is hardly applicable when both tasks are very different. In that case, more final layers need to be re-trained. In general, a network learns the hierarchy of features starting from generic ones, e.g., colors, edges, curves, etc., which can be reused for most of the tasks. Conversely, the later layers respond to more specific features of the original task which can only be reusable in case the new task is similar.

4.2 Experimentation Environment

To further investigate the questions raised in the previous section, a comprehensive experimentation environment has been setup. It comprises of five datasets of different characteristics and three state-of-the-art pre-trained image classification networks. The complexity of the experiments has been calculated as the number of datasets times the number of trainable blocks of all the networks. Therefore, computational power becomes a critical factor to perform these experiments in a reasonable time. There were 5 Nvidia 1080Ti Graphics Processing Units (GPUs) used to train around 200 models.

Table 4.1: Open-source image repositories

Dataset	**Training-set**	**Testing-set**	**Classes**	**Size**
ImageNet (Russakovsky et al., 2015)	1.2 million	50,000	1000	very large
Food (Bossard et al., 2014)	75,750	25,250	101	large
Caltech-101 (Fei-Fei et al., 2007)	6,144	2,096	101	large
ChestXray (Demner-Fushman et al., 2016)	5,943	1,487	2	small
Flowers (Nilsback and Zisserman, 2008)	2,753	917	5	small
Coco-Animals (Lin et al., 2014)	800	200	8	small

4.2.1 Datasets

In this work, five publicly available datasets have been used with different domain and characteristics. They can be divided into two categories based on their size and number of classes; large and small as shown in Table 4.1. The pre-trained networks which are selected for this work are trained on ImageNet. The Food dataset, introduced by Bossard et al. (2014), is a challenging collection of 101 food categories and 101,000 instances. Likewise, Caltech dataset also has 101 categories with 82 images per category on average (Fei-Fei et al., 2007). The images are not specific to any particular domain. Chest-Xray (Demner-Fushman et al., 2016) is a relatively smaller dataset, originally published with 14 classes. The images were mostly tagged with multiple labels which are converted to the two-class problem where every image can be classified as either normal or nodule. This dataset is composed of frontal-view X-ray images of the screening and diagnosis of many lung-related diseases. Similarly, Flowers is another small dataset consisting of five different categories of flower species (Nilsback and Zisserman, 2008). Microsoft has gathered a large dataset consisting of 91 categories, known as Common Objects in Context (Coco) (Lin et al., 2014). Coco-Animals (Animals) is a subset of the original Coco dataset which is composed of 8 animal categories.

4.2.2 Pre-trained Image Classification Networks

Three pre-trained image classification and detection Convolutional Neural Network (CNN) have been used in this work. These networks are trained on ImageNet dataset which consists of 1000 classes (Russakovsky et al., 2015), however, their internal architecture, depth and other aspects differ considerably. The first few layers of the networks capture low-level features of the image like edges, curves, etc. The subsequent layers learned shapes and more abstract features related to the problem domain. The final layers have learned more specific features corresponding to a particular category which is eventually used to classify the images. The pre-trained networks are listed in Table 4.2 along with the number of layers and accuracy in the ImageNet dataset.

Table 4.2: Benchmarking of various pre-trained image classification models

Network	**Layers**	**Top-1 Accuracy**	**Top-5 Accuracy**
Inception-v3 (Szegedy et al., 2016)	22	78.0	93.9
Inception-ResNet-v2	152	80.4	95.3
VGG-19 (Simonyan and Zisserman, 2014)	19	71.1	89.8

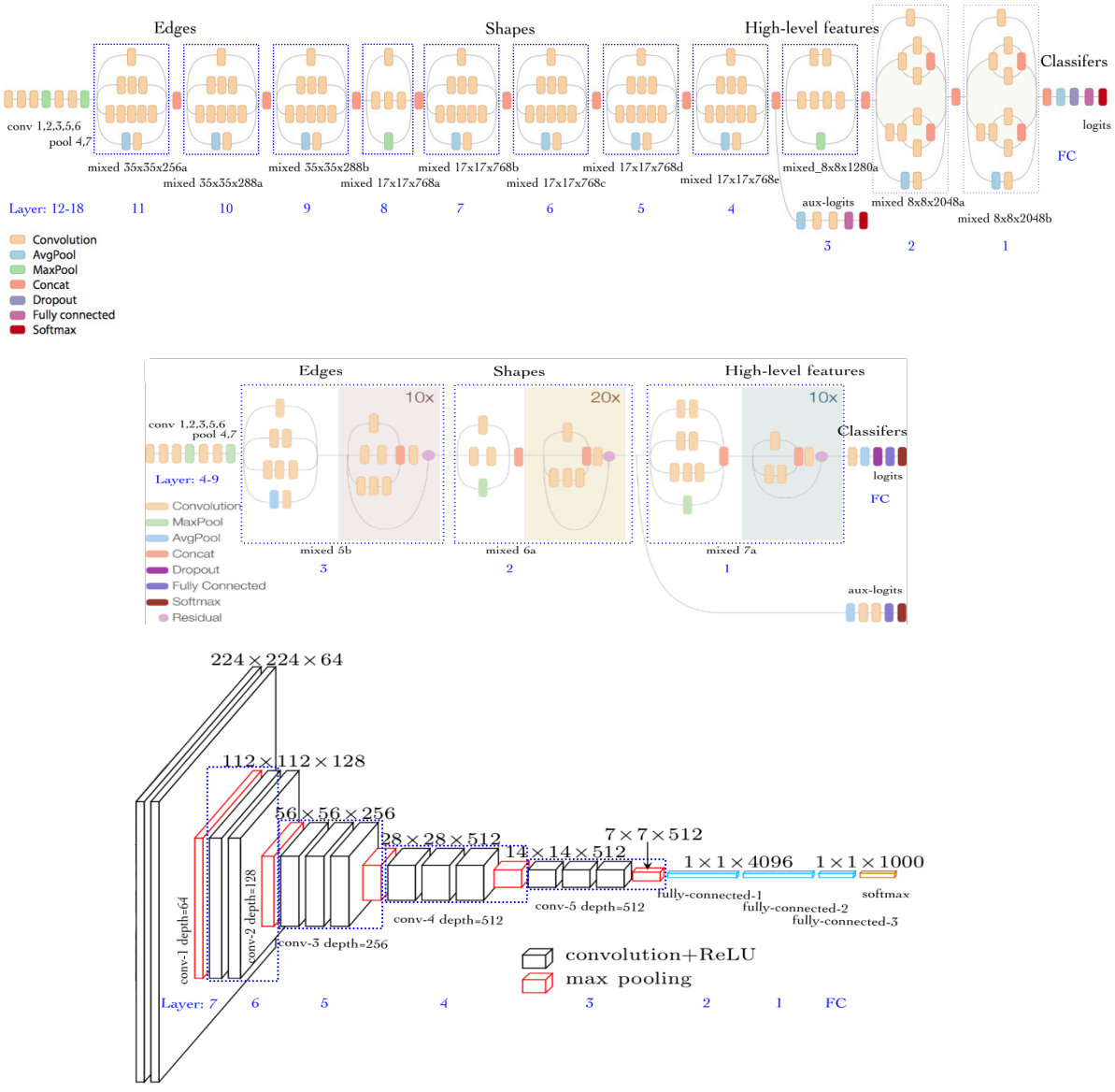

Figure 4.3: Schematic view of Inception-v3, Inception-ResNet-v2 and VGG-19 networks where the blue colour is representing a re-trainable layer/block.

4.2.2.1 Inception-ResNet-v2

Google released Inception-ResNet in 2016 and it became a state-of-the-art image classification network of ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)-2016. Inception-ResNet-v2 is a deeper but simplified version of Inception-v3. The residual connections allow the model to be even deeper, leading to better performance. ResNet relies on micro-architecture modules which consist of building blocks.

A schematic view of different pre-trained architectures can be seen in Figure 4.3. The architectures are also labelled with the block numbers, in blue, that can be subject to fine-tuning.

4.2.2.2 VGG-19

Visual Geometry Group (VGG) network was developed by Visual Geometry Group of Oxford University which secured first place in the ImageNet ILSVRC-2014. It has two versions which consist of 16 and 19 layers. The 19 layer version has been used in our experiments. The VGG network uses 3x3 convolutions stacked on top of each other in increasing depth which makes it relatively simpler than AlexNet. The convolutional layers are followed by two Fully-Connected (FC) layers, each one consisting of 4,096 neurons and a Softmax classifier.

4.2.2.3 Inception-v3

Inception, or GoogLeNet, was developed by Google and was state-of-the-art for image classification and detection in the ILSVRC-2015. Inception-v3 is a 22 layers deep network but computationally inexpensive (Szegedy et al., 2015).

4.2.3 Transfer Learning

In TL three pre-trained networks are re-trained/fine-tuned sequentially on the same task. The training process fine-tunes a range of blocks per training iteration, starting from the final block. This process has been repeated for all the pre-trained networks and datasets. The hyper-parameters have also been updated layer-wise one by one where the learning rate initializes from a comparatively large number to iteratively smaller. Conversely, the number of training epochs parameter has been initialized from a smaller number which gets bigger as more layers need to re-train. The *rmsprop* optimizer (Hinton et al., 2014) and layer dropout of 20-30% have been used while re-training the network. The learning rate and the number of training epochs are dependent on the nature of the tasks and depth of the network. The training begins with the higher value of learning rate and lower number of epochs which gradually decreases and increases, respectively, as more layers of the network require fine-tuning. Their values are changed with a small factor upon the addition of a new layer for fine-tuning. The idea is to use the lower value of learning rate and a higher number of epochs for larger datasets. Table 4.3 shows hyper-parameters that are used in our experiments.

Table 4.3: Hyper-parameters that are used for transfer learning

Datasets	**Learning rate**	**Training epochs**	**Dropout**
Food	$10^{-3} - 10^{-7}$	180-1000	20%
Caltech	$10^{-4} - 10^{-7}$	180-1000	20%
ChestXray	$10^{-3} - 10^{-6}$	120-800	20-30%
Flowers	$10^{-3} - 10^{-6}$	120-800	20%
Animals	$10^{-3} - 10^{-6}$	120-800	20%

Table 4.4: Transfer learning accuracies of various datasets, classification architectures, and their layers

Network re-training accuracy (train %–test %) upon fine-tuning a range of blocks, one block per iteration															
Dataset	**FC**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Inception-v3 network															
Food	85–53	84–48	84–49	72–55	70–57	73–66	73–66	76–68	79–62	84–75	82–79	85–80	**86–82**	79–69	80–70
Caltech	95–84	95–87	96–86	**96–89**	96–87	95–86	93–84	88–78	82–76	69–61	48–45	37–31	24–18	15–16	12–14
Chest-Xray	96–50	96–52	96–52	96–59	96–61	96–67	96–68	96–69	96–71	96–70	96–71	95–71	92–71	78–72	**87–75**
Flowers	84–25	84–27	87–19	88–27	91–34	95–86	96–86	**96–89**	96–88	95–89	92–82	86–83	78–70	56–58	41–38
Animals	54–14	47–17	58–48	81–59	88–61	90–64	**92–69**	91–62	90–69	88–56	81–52	69–46	58–51	47–44	38–41
Inception-ResNet-v2 network															
Food	86–56	86–64	86–74	86–74	87–78	**89–85**	86–84	80–76	71–73	78–75	–	–	–	–	–
Caltech	96–83	**96–84**	95–83	94–84	94–79	93–79	91–78	88–64	79–64	67–59	–	–	–	–	–
Chest-Xray	91–61	89–43	88–44	91–79	91–79	91–44	91–44	91–79	**91–80**	91–79	–	–	–	–	–
Flowers	89–22	89–28	91–35	92–26	93–81	**94–87**	95–83	96–85	96–84	94–81	–	–	–	–	–
Animals	65–49	71–60	77–70	77–56	**82–77**	85–74	86–68	88–68	88–68	85–66	–	–	–	–	–
VGG-19 network															
Food	85–69	85–67	85–67	90–77	**81–80**	77–73	77–73	–	–	–	–	–	–	–	–
Caltech	**79–78**	72–70	80–77	74–70	68–66	66–53	71–66	–	–	–	–	–	–	–	–
Chest-Xray	89–43	87–43	89–61	88–78	89–74	**89–78**	89–78	–	–	–	–	–	–	–	–
Flowers	83–59	81–80	83–81	**86–84**	79–72	79–63	90–39	–	–	–	–	–	–	–	–
Animals	78–71	**79–76**	70–57	74–49	72–38	73–37	79–34	–	–	–	–	–	–	–	–

4.3 Results and Analysis

An extensive set of experiments has been performed to analyze the relationship of size and similarity of a task with the depth of pre-trained network that needs to be fine-tuned. The depth of the pre-trained networks, which is fine-tuned, is varied from 7 to 18 blocks. The layer-wise training and validation accuracies have been reported in Table 4.4. The table shows accuracies of five datasets against three different architectures and the number of fine-tuned blocks. The top-performing numbers of blocks are in bold. The relationship between the validation accuracy and the number of blocks has been depicted in Figure 4.4. The layer-wise training and validation accuracies have been reported in Table 4.4.

The accuracy of a pre-trained network after fine-tuning every block, also known as a block-wise result, is validated with dataset similarity analysis. The networks that are used in this work were originally trained on the ImageNet dataset. Therefore, the validation set of all the datasets have been inferred by the pre-trained networks to compute their similarity with ImageNet. As a result, the maximum of the Probability Mass Function (PMF) of an image over the 1000 classes, which is referred to as image similarity to ImageNet, and entropy have been calculated and averaged over the number of images N in the dataset. The number of classes of datasets other than ImageNet is denoted by M. The similarity and entropy are calculated using Equations 4.1 and 4.2, respectively.

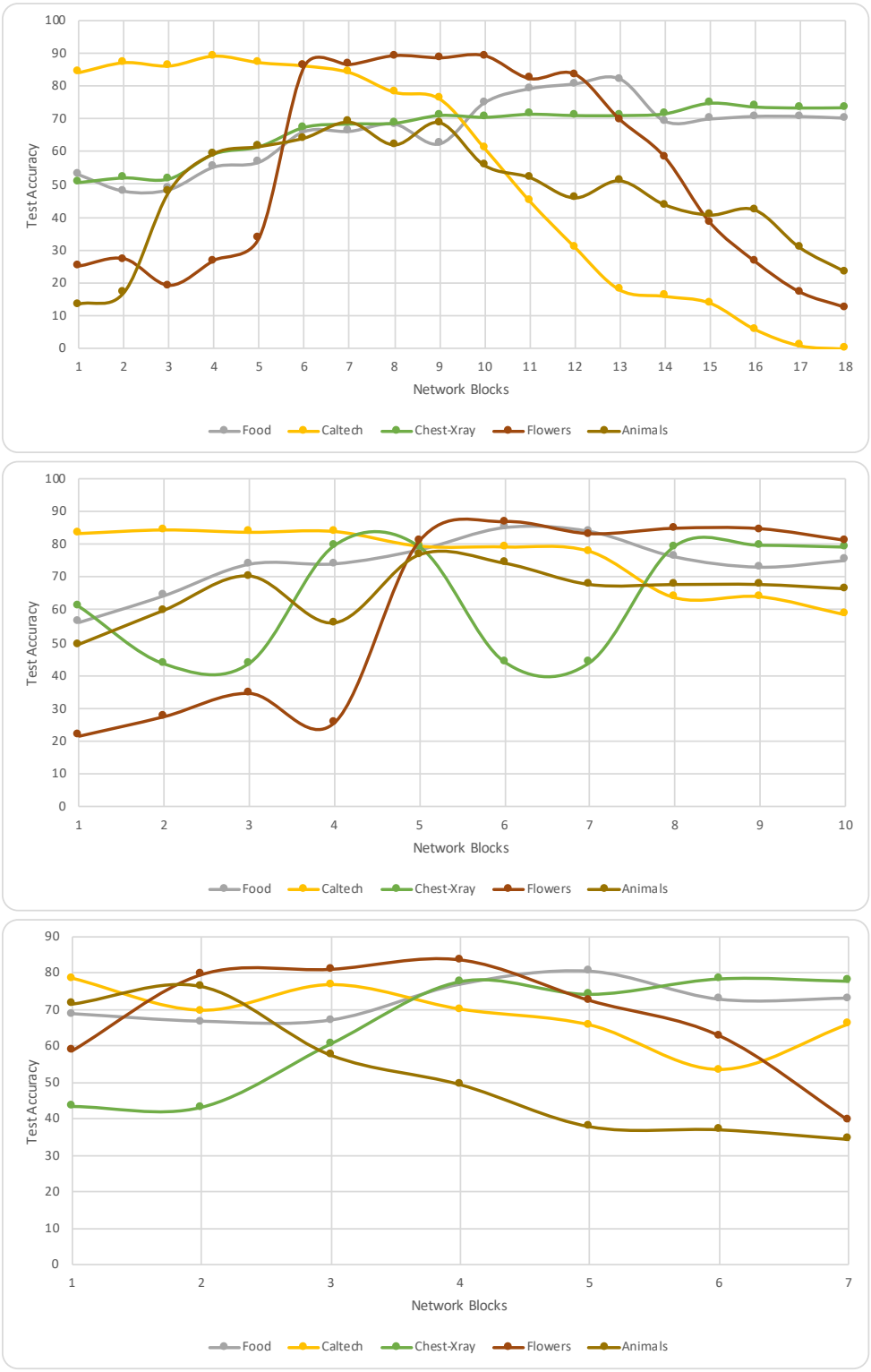

Figure 4.4: Transfer learning accuracies of pre-trained networks; (a) Inception, (b) Inception-ResNet and (c) VGG-19 on ImageNet

$$\text{similarity} = \frac{1}{N}\sum_{i=1}^{N}\max(\mathbf{f}(x_i)) \tag{4.1}$$

where $\mathbf{f}(x_i)$ is the PMF over classes conditioned on the input image x_i, typically the output of the softmax layer.

$$\text{average entropy} = \frac{1}{N}\sum_{i=1}^{N}(-\sum_{j=1}^{M}(\mathbf{f}_j(x_i) * \log_2(\mathbf{f}_j(x_i)))) \tag{4.2}$$

The similarity of an image from the new domain with the original domain is computed by feeding the image to the original pre-trained network and examining the output probability distribution over the classes. The dataset similarity scores have been recorded in Table 4.6. The similarity results are correlated with the number of blocks that are needed to fine-tune networks on new tasks. Figure 4.6 shows that for tasks where similarity is higher (and the entropy is lower), fewer blocks need to be fine-tuned. On the contrary, more blocks need to be fine-tuned where the datasets are less similar (having low similarity and higher entropy values). This supports our claim that TL is effective for related tasks regardless of their size. However, TL is also useful for dissimilar tasks, i.e., Chest-Xray and Food, but more blocks need to be re-trained to get good results. Moreover, similar tasks require fine-tuning of either only fully-connected layer(s) or high-level features block in some cases. Accordingly, less similar tasks require fine-tuning of more deeper layers, i.e., blocks representing shapes and edges blocks.

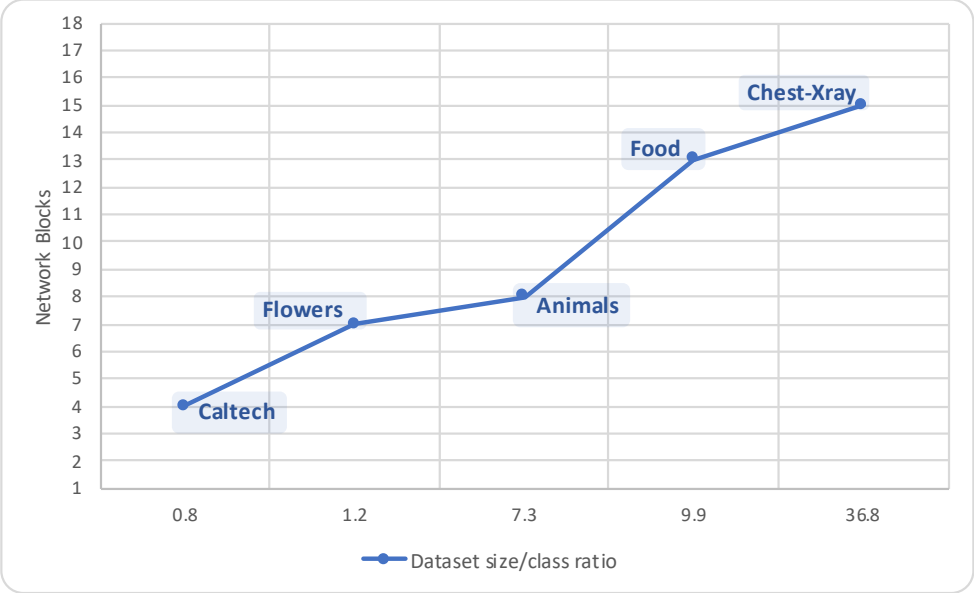

Figure 4.5: Inception-v3 blocks vs dataset size/class ration trend

Figure 4.5 shows that the size meta-feature has a good correlation with the depth of the network that is fine-tuned. The contribution of the similarity of a dataset dominates over its size when both datasets are similar. However, size becomes critical when both datasets have less similarity between them. It only supports the network to generalize while fine-tuning more deeper blocks of the network, e.g., Food and Chest-Xray dataset. The Food

Table 4.5: The state-of-the-art accuracy (training of the network from scratch) versus best possible accuracy from this work

Dataset	**Accuracy of the network from scratch**	**Architecture**	**Reference**	**Accuracy from this work**
Food	88.28%	InceptionV3	Hassannejad et al. (2016)	84.93%
Caltech	91.44%	SPP-Net	Hem et al. (2014)	89.00%
ChestXray	84.11%	CheXNet	Rajpurkar et al. (2017)	79.52%
Flowers	91.52%	CNN-SVM	Lin et al. (2015)	89.06%
Animals	-	-	-	76.70%

datasets consist of over 100,000 examples with over 100 classes whereas Chest-Xray has around 8,000 instances with only 2 classes. Based on the number of classes both datasets have a reasonable size to class ratio which allow them to fine-tune more deeper networks.

The Food and Chest-Xray datasets' domains are different from ImageNet. Consequently, more deeper blocks have been fine-tuned. TL is more effective than training the model from scratch for these tasks. The maximum validation accuracy of fine-tuned Food and Chest-Xray is closer to the model which is trained from scratch. These accuracies as compared to the training of the network from scratch, thus reported by various studies, are presented in Table 4.5. However, TL requires much less effort and resources, in terms of parameter tuning and computation.

Table 4.6: The similarity and average entropy of different datasets

Dataset	**Inception-v3**	**Inception-ResNet-v2**	**VGG-19**
ImageNet	76.61% – 2.11	78.77% – 1.84	72.7% – 2.23
Food	53.40% – 3.52	59.23% – 3.47	51.24% – 3.83
Caltech	60.41% – 3.27	64.30% – 2.62	57.58% – 2.40
Chest-Xray	40.88% – 4.88	43.25% – 4.40	34.72% – 4.74
Flowers	52.25% – 4.01	60.19% – 3.15	49.72% – 3.12
Animals	54.88% – 3.68	64.87% – 2.68	53.08% – 2.79

4.4 Summary

This work presents an empirical study of the relationship between various characteristics describing the similarity of two datasets, and based on that, the amount of fine-tuning required to achieve accuracy close to state-of-the-art. It addresses model's hyper-parameter optimization research challenge, which is defined in Section 2.7, in the context of HPO. Even though the experiments were limited to only two characteristics, size and similarity with the original task, still as per some studies these are most important in this context. The datasets with both similar and different domains as well as different sizes have been used. Also, three state-of-the-art image classification networks trained on ImageNet were used in the experiments. Extensive experiments have been conducted on different combinations of pre-trained networks (and their blocks), datasets and hyper-parameters. The block-wise

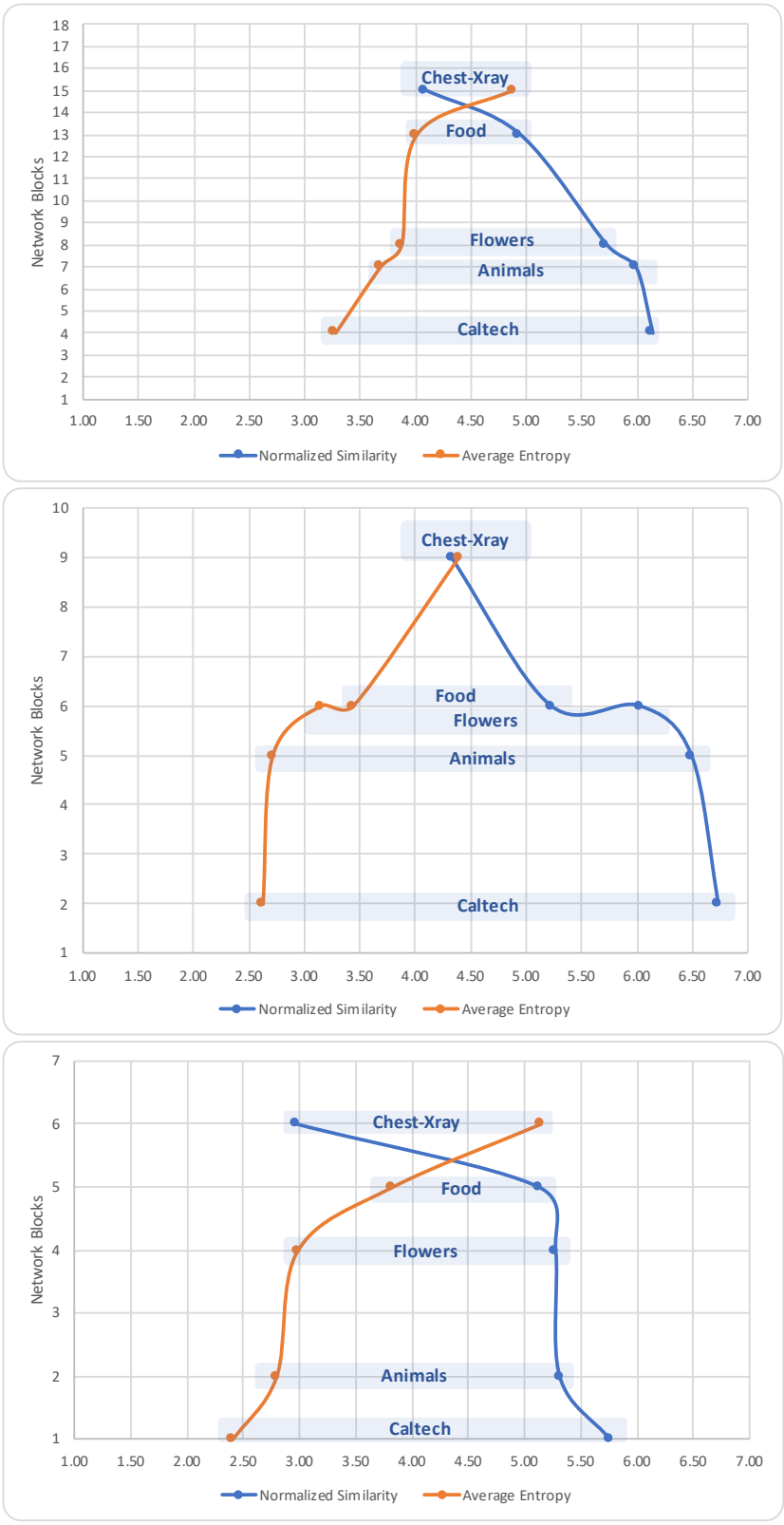

Figure 4.6: Datasets similarity with ImageNet for; (a) Inception-v3, (b) Inception-ResNet-v2 and (c) VGG-19 architectures. The similarity is normalized so that it can fit in between the scale of 1-10 with entropy. It is multiplied by 10.

results are validated with dataset similarity analysis where the probability of match and entropy of the datasets are correlated with the fine-tuning of the number of blocks. The proposed approach first computes the similarity of the new task with the original one and combines it with the size of the new task to identify which section of the architecture needs fine-tuning.

The experiments were designed around two Meta-feature (MF) where only the datasets having different characteristics were considered. In general, TL is found to be effective for tasks similar to the original one, regardless of the size, where mostly fine-tuning of the final blocks produces close to state-of-the-art accuracy. On the other hand, this work is handy for the tasks having less or no similarity with the original task with very few training examples, i.e., problems related to Medical Imaging (Rajpurkar et al., 2017). It allows to find the minimum number of blocks a pre-trained network require fine-tuning to achieve the best possible accuracy based on the characteristics of two tasks. It also identifies the portion of the pre-trained network which can be reusable based on the similarity and size among the two tasks. The key characteristic of TL is that it saves significant computation and training time while achieving similar accuracy to the networks trained from scratch. This study preserves the key characteristics of TL atleast for less similar tasks which verify the intuition that one can more effectively reuse pre-trained network. This study is limited only to network depth hyper-parameter which is not sufficient for Deep Neural Networks (DNN). The DNN deals with a wide range of hyper-parameter choices which arise the need to extend the search to more than one hyper-parameter.

Chapter 5

A Meta-Reinforcement Learning Approach to Optimize Parameters and Hyper-parameters Simultaneously

This chapter presents a framework to automatically find a good set of hyper-parameters resulting in a reasonably good accuracy, which at the same time is less computationally expensive. In continuity with the study conducted in Chapter 4, which is limited to a single hyper-parameter, fine-tuning of the network depth based on task similarity. This study has expanded the set of hyper-parameters while implicitly considering the task similarity at the intrinsic dynamics of the training process. The idea pursued here is to frame the hyper-parameter selection and tune within the reinforcement learning regime.

Every phase of the Machine Learning (ML) pipeline involves choices of algorithms and their hyper-parameters. These choices have a direct impact on the performance of the model. The recent advances in Neural Network (NN), Deep Neural Networks (DNN), is crucially dependent on tuning of a number of hyper-parameters. Thus, the optimal selection of architecture and its hyper-parameters is considered as a key area of Automatic Machine Learning (Auto-ML) (Feurer et al., 2015). Auto-ML is primarily dealing with the end-to-end automation of the ML pipeline consisting of data pre-processing, algorithm selection and hyper-parameters tuning.

In recent years, Meta-Reinforcement Learning (Meta-RL) has become a de-facto standard to automatically search for optimal hyper-parameters. The proposed framework uses Meta-RL to efficiently explore the optimal hyper-parameters of a deep network from the given search space. The exploration happens simultaneously for both the policy network and the DNN. Given a tuple of hyper-parameters that is generated by a policy network, a network is built and trained for a number of steps. The network computes accuracy on hold-out validation-set whose delta is used as a reward. Furthermore, this reward along with the state of the network comprising statistics of the probability distribution over the number of classes and training loss, are back-propagated to the policy network which generates a tuned tuple for the next time-step. The network is initialized once where different tuples of hyper-parameters are tested on the go without resetting the network. Therefore, a tuple of

hyper-parameters is not required to train until convergence of the network, which saves a significant amount of computation.

The proposed approach is an efficient form of Neural Architecture Search (NAS) and Efficient Neural Architecture Search (ENAS) to find optimal neural architecture. The shortcomings of NAS is its limitation to small tasks because it is computationally very expensive. On the other hand, ENAS keeps numerous architectures in the memory so that the new architectures can share the weights of the pre-trained blocks. This work further simplifies architecture search problem which is equally effective for large datasets. The approach tunes the hyper-parameters of the network during training rather than waiting until convergence which saves significant computation time. The effectiveness of a tuple of hyper-parameters is tested by training for a few steps. Further, the feedback of the tuple is used to tune the policy gradient at the same time-step.

This method significantly reduces the computational complexity of the optimal hyper-parameter search problem. Along with minimal computation, the approach requires a substantially smaller amount of memory by optimizing a single instance of the network rather than creating and keeping numerous architectures in the memory. The simplicity of the approach does not affect the accuracy of the network and makes it equally effective for more complex and bigger tasks. This is the key contribution of this study.

5.1 Methodology

The primary goal of this study is to efficiently explore the optimal set of hyper-parameters for a given task. This is achieved by optimizing the meta-learner parameters and network hyper-parameters at the same time. Typically, the policy network needs to train for several episodes so that it can start producing an effective outcome. In case of hyper-parameter tuning using Meta-RL, the child network needs to be sequentially trained on a task at hand using all the tuples, recommended by the meta-learner, until convergence in order to conclude their effectiveness. It becomes a time and computationally intensive task. Hence, this challenge has been tackled and addressed in this study.

In order to evaluate the proposed approach, a framework is designed using a typical Reinforcement Learning (RL) setting which consists of two key components: an agent and an environment (Sutton et al., 1999). The environment can be in different states ($\mathcal{S}$) which are observed by the agent at different time-steps (t). Given its knowledge of the state and a set of available actions, the agent chooses an action ($\mathcal{A}$). These actions affect the state of the environment and in return, generate a reward ($\mathcal{R}$). To find the optimal set of hyper-parameters the agent needs to find the actions that lead to maximizing expected reward, see Equation 5.1. The γ is a discount factor, which allows the agent to maximize its expected reward on either short- or long-term transitions based on its value. However, the reward is non-differentiable and hence needs a policy gradient method to iteratively update θ as formulated in Equation 5.2. The stochastic policy $\pi(a|s)$ describes a probability distribution over the set of actions.

$$R_t \leftarrow \sum_{i=0}^{\infty} \gamma^i R_{t+i}, \quad \gamma \in [0, 1] \tag{5.1}$$

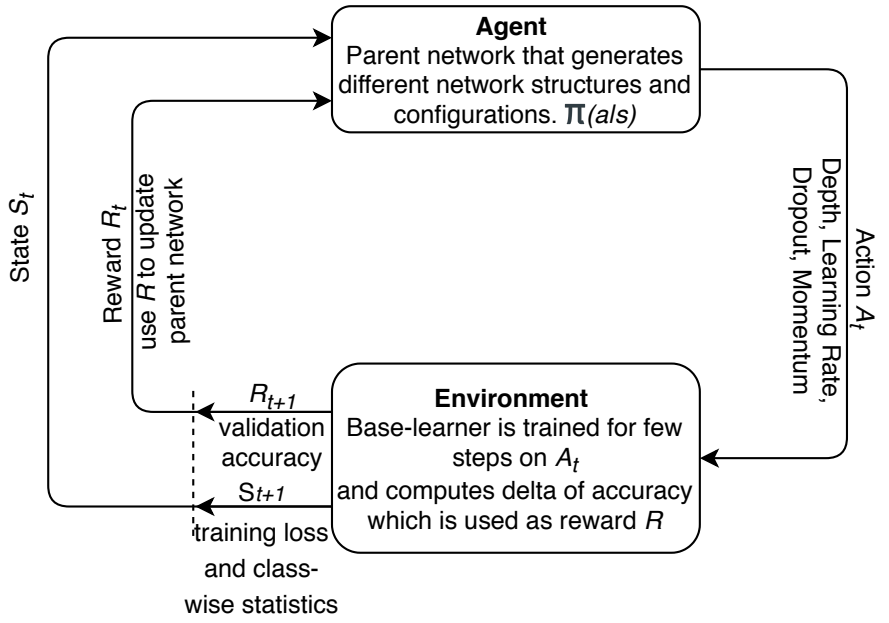

Figure 5.1: A typical setting of Meta-RL framework where agent contains a policy gradient and network sits in the environment

$$\theta \leftarrow \theta + \alpha \nabla_\theta log \pi_\theta(s_t, a_t) r_t \tag{5.2}$$

The agent generates a tuple of hyper-parameters using a Recurrent Neural Network (RNN) which is known as meta-learner. This tuple specifies a neural network, architecture known as base-learner, in the framework. The base-learner is trained on a task and evaluated on the held-out validation-set. The base-learner provides feedback to the meta-learner to get a well-tuned tuple in the next time-step. Figure 5.1 shows the setting of the proposed Meta-RL framework.

5.1.1 Meta-learner

The Meta-learner consists of a stochastic policy gradient which makes weight adjustments in a direction that lies along the gradient of expected reinforcement. It is a statistical gradient-based approach known as REINFORCE as described by Williams (1992). It makes weights adjustment without explicitly computing gradient estimates with back-propagation. The Meta-learner initializes a base-learner once with the initial values of hyper-parameters from search space except for depth. However, the depth is initialized with the maximum value which is defined for a task. For instance, if the maximum depth is 34 in the search space, the network is initialized once with the maximum depth. The meta-learner is a two-layer RNN Long Short-term Memory (LSTM) with 35 neurons per layer. The network is trained with Adam optimizer (Kingma and Ba, 2015). An initial learning rate of 0.0006 has been used. The weights are initialized with Xavier-initialization (Glorot and Bengio, 2010). A discount factor of 0.97 is used to prevent the total reward from reaching infinity. The meta-learner is updated via a policy gradient method which is computed using an immediate reward.

Algorithm 1 Computing immediate reward of an episode

1: $beta = 0.8$
2: Time-step $= t$
3: $episode = e$
4:
5: $reward_t = (accuracy - moving\_accuracy_{t-1})$
6: $reward_t = clip(reward, -0.1, 0.1)$
7:
8: $moving\_accuracy_t = (1 - beta) * accuracy_e$
9: $moving\_accuracy_t \mathrel{+}= beta * moving\_accuracy_{t-1}$

5.1.2 Base-learner

The base-learner used in this work is a modified form of Residual Networks (ResNet) (He et al., 2016). It is constructed by stacking a set of residual blocks on top of the input layer and followed by a Fully-Connected (FC) layer. A block consists of a sequence of two convolutional layers with filter sizes 1x1 and 3x3, respectively, where a stride of 2 is used by the first convolutional layer to reduce feature map size. Also, there is a bottleneck setting of the block which consists of three convolutional layers with filter sizes of 1x1, 3x3 and 1x1, respectively. The bottleneck block is used for the networks with a depth of 50 or more. The benefit of using ResNet architecture is two-fold: a) residual blocks have repeated units of convolutions with fixed hyper-parameters, namely, kernels and strides, and b) it has a skip-connection feature that provides flexibility to change the depth of the network during training. The base-leaner has been initiated once and its hyper-parameters are modified during the training cycles.

Table 5.1: Hyper-parameter search space and parameters covering behaviour of the network that is used as states $_{t+1}$

Parameters	**Values (range)**
A. Hyper-parameter search space	
Number of layers (D)	2-50
Dropout Rate (DR)	0.5-1.0
Learning Rate (LR)	0.0001-0.9
Momentum (M)	0.6-0.99
B. Representation of the environment (states)	
Network training loss	0-1.0
Mean entropy of class probabilities	0-1.0
Standard deviation entropy of class probabilities	0-1.0

The meta-learner (RNN) suggests a tuple of hyper-parameters from the search space which are listed in Table 5.1 (A). The table shows the search space range of all the hyper-parameters. Based on the suggested hyper-parameters, the existing Convolutional Neural Network (CNN) architecture is trained for 50 steps with a batch size of 32. Furthermore, the delta of validation accuracy has been computed which becomes the immediate reward. The reward that is used to update the meta-learner is the delta of validation accuracy and

moving accuracy of the recent two episodes. The procedure to compute the immediate reward is formulated in Algorithm 1. Apart from the reward few other parameters of the environment are computed at time-step t comprising of network training loss and entropy of probability distribution over the number of classes. The entropy is averaged over an episode, see Equation 5.3, where x is the output of the softmax layer and N is the size of the episode. Further, the mean and standard deviation of the entropy have been computed over the number of images, N, processed in an episode, see Table 5.1 (B). These parameters are utilized by meta-learner as the state information to generate a tuned tuple for time-step $t+1$. The network is trained with Momentum optimizer with Nesterov momentum (Sutskever et al., 2013).

$$\text{entropy} = -\sum_{j=1}^{N}(\mathbf{f}_j(x_i) * \log_2(\mathbf{f}_j(x_i))) \tag{5.3}$$

$$x_{l+1} = ReLU(x_l + \mathbf{f}(x_l, W_l)) \tag{5.4}$$

5.1.2.1 Residual Block with Stochastic Depth

A residual block is composed of convolution layers, batch normalization (BatchNorm) (Ioffe and Szegedy, 2015) and Rectified Linear Units (ReLU) (Nair and Hinton, 2010) which is represented as function f in Equation 5.4. x_l represents skip-connection path and $f(x_l, W_l)$ is a residual block. A configuration of the base-learner with maximum depth 4 is shown in Figure 5.2. The meta-learner has recommended a depth size 3 so the last residual block has been disabled for the current episode. Hence, the gradient update of the last block is stopped for the current episode.

The depth of the network is controlled by stochastic depth approach presented by Huang et al. (2016). It leverages the skip-connection path of the residual block x_l to control network depth even during the training of the network. The idea of original stochastic depth work, Huang et al. (2016), is to randomly skip the residual blocks by letting through only the identity of the raw feature in order to skip a path. In this work rather than randomly skipping the blocks, meta-learner suggests which blocks to skip. Therefore, when a block is skipped, the identity path has been chosen which stops updating the block's gradients.

5.2 Formulation

The approach to optimize parameters and hyper-parameters simultaneously is outlined in Algorithm 2. It has two components: a) meta-learner and b) base-learner. A meta-learner is an RNN which suggests a tuple of hyper-parameters in the form of actions. These actions are applied to the environment which is a base-learner. The base-learner is a CNN which trains the task at hand on the actions of current time-step for a few steps. Furthermore, the network computes accuracy on a hold-out validation-set which is used as an immediate reward at the time-step t. This reward and the state of the network is observed and used to update the weights of the meta-learner that generates new actions for time-step $t+1$ which are dependent on how well the base-learner performs.

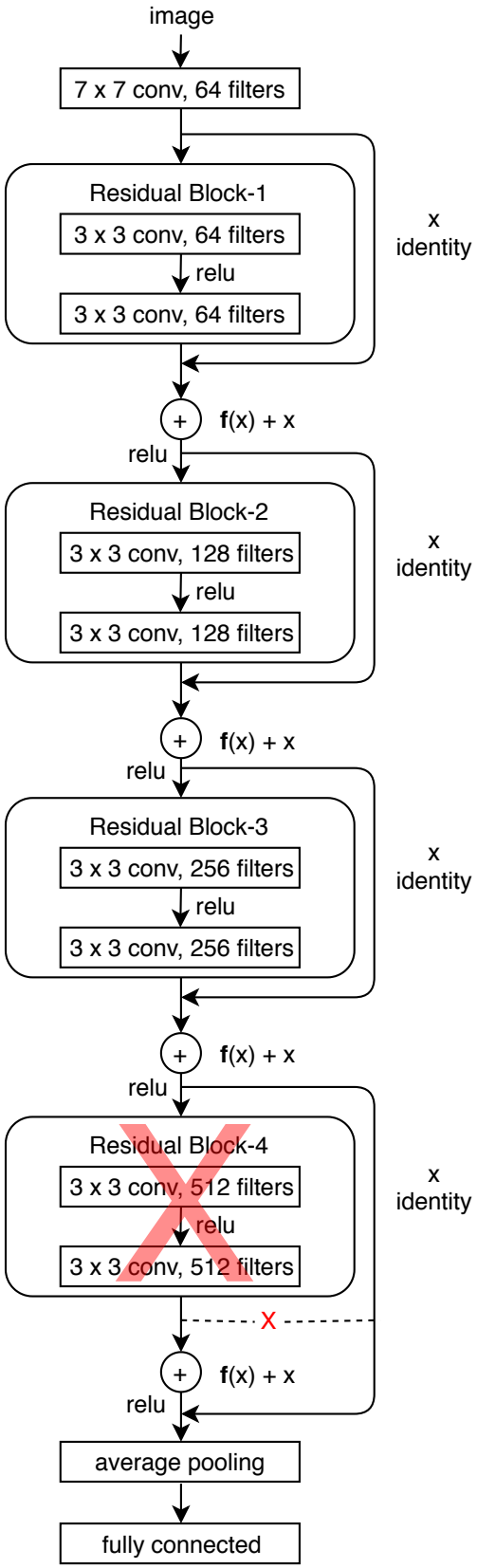

Figure 5.2: A schematic view of base-learner with maximum depth 4 and current depth 3

Algorithm 2 Meta-RL algorithm to optimize parameters and hyper-parameters simultaneously

▷ META-LEARNER
Network depth = D
Dropout rate = DR
Base-learner's Learning rate = α_b
Momentum = p
Actions $(a) = < D, DR, \alpha_b, p >$
Time-step = t
Meta-learner's Learning rate = α_m
Reward at time $t = r_t$
Differential policy at time t which maps actions to probabilities = $\pi_\theta(s_t, a_t)$
Initialize the policy parameter: θ = Xavier-initialization
Initialize base-learner CNN: $model \leftarrow ResNet(a)$

for $episode \leftarrow 1$ to $\pi_\theta : s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T$ **do**
    ▷ policy network
    for $t \leftarrow 1$ to $T - 1$ **do**
        $\theta \leftarrow \theta + \alpha_m \nabla_\theta log \pi_\theta(s_t, a_t) r_t$ ▷ gradient update

        ▷ BASE-LEARNER
        ▷ Tune the hyper-parameters of network with θ
        for $s \leftarrow 1$ to $Steps \leftarrow 50$ **do**
            $features \leftarrow next\_batch(train, labels)$
            if $training = True$ **then**
                $fit\_model \leftarrow model.fit(a, features)$
            end if
        end for
        if $testing = True$ **then**
            $test\_accuracy \leftarrow fitted\_model(testset)$
        end if

        $r_t = test\_accuracy_t - moving\_accuracy_{t-1}$
        $s_t^1 = train\_loss$ ▷ states of t
        $s_t^2 = final\_layer\_statistics$
    end for
end for

Algorithm 3 shows how stochastic depth approach is modified for this work. The base-learner only updates the gradients of the residual blocks which are less than the suggested depth (D). For the rest of the layers, a skip-connection path has opted. The base-learner is initialized with a maximum value of the depth once and modifies, often, on every episode.

Algorithm 3 Stochastic Depth routine

Depth suggested by meta-learner = D
Maximum depth of a network = $maxD$

for $block\_no \leftarrow 1$ to $maxD$ **do**
    if $block\_no >= D$ **then** $x \leftarrow ReLU(x + \mathbf{f}(x, W))$ ▷ residual block
    end if
    if $block\_no < D$ **then** $x \leftarrow Identity(x)$ ▷ shortcut
    end if
end for

5.3 Experimentation Environment

In order to evaluate the proposed approach, a number of experiments have been performed. These experiments use different image classification tasks listed in Table 5.2. A tuple of hyper-parameters is tested for only a few steps rather than till convergence. Hence, the number of steps the base-learner trains on a tuple of hyper-parameters is a critical parameter. Thus, different values of step-size and batch size have been tested to obtain the optimal values which can evaluate a recommended tuple in the shortest time. The experiments suggest a step-size 50 with a batch size 32 which is sufficient to test a tuple of hyper-parameters efficiently. Likewise, capturing the appropriate parameters which can better represent the state of the network after a training episode is key. The accuracy or loss can be sufficient if for each of the generated tuples the network is trained until convergence. Hence, the behaviour of the environment, at every episode, has been captured to evaluate the effectiveness of the recommended tuple. The effectiveness of a tuple is measured using the validation accuracy.

5.3.1 Datasets

In this work, five publicly available datasets have been used with different characteristics and complexity levels. The proposed approach is equally effective for both small and large datasets unlike most of the neural architecture search approaches which are only tested on small datasets. The datasets size, number of classes and image resolution is listed in Table 5.2. The datasets are divided into training- and validation-set with 80-20 split.

5.4 Results and Analysis

A comprehensive set of experiments is conducted to evaluate the effectiveness of the proposed approach. The experiments were performed on 5 Nvidia 1080Ti Graphics Processing

Table 5.2: Image datasets used in this work

Dataset	**Training-set**	**Testing-set**	**Classes**	**Dimensions**
Mnist (LeCun et al., 1999)	50,000	10,000	10	28x28x1
Fashion-mnist (Xiao et al., 2017)	60,000	10,000	10	28x28x1
Cifar-10 ("CIFAR-10 and CIFAR-100")	50,000	10,000	10	32x32x3
Cifar-100 ("CIFAR-10 and CIFAR-100")	50,00	10,000	100	32x32x3
Tiny-imagenet (Le and Yang, 2015)	100,000	20,000	200	64x64x3

Units (GPUs), one dataset per GPU. A comparison of the proposed approach with other architecture search approaches is shown in Table 5.3. This comparison is only available for Cifar-10 dataset as most of the previous studies used it in their experiments. A plot of validation accuracy against the time taken can be seen in Figure 5.3. The vertical red dotted line is pointing to the top accuracy whose hyper-parameters settings are mentioned in Table 5.4.

Table 5.3: Comparison with different architecture search approaches on Cifar-10 dataset

Method	**GPUs**	**Exploration time (days)**	**Parameters (millions)**	**Error rate (%)**
DenseNet (DeVries and Taylor, 2017)	-	-	26.20	3.46
NASNet-A (Zoph et al., 2018)	450	3-4	3.30	3.41
PNAS (Liu et al., 2018)	100	1.5	3.20	3.63
ENAS (Pham et al., 2018)	1	0.60	4.60	2.89
This work (Cifar-10)	1	0.40	4.58	3.11

There are 5 datasets used for experiments with different complexity-levels. The exploration of hyper-parameters for the datasets posses different behaviours in terms of the number of episodes and time. The Mnist, Fasion-mnist and Cifar-10 datasets were comparatively easier to learn. On the other hand, the exploration of Cifar-100 and tiny-imagenet was hard. The complex datasets took many more episodes to explore the optimal parameters from the search space. Moreover, the maximum depth of the architectures was bigger for complex datasets. So a large increase of depth size from one episode to other, particularly in the initial phase, makes the training quite unstable. Figure 5.3 shows a consistent accuracy after 100 minutes of training till 630 followed by a spike on a tuple. This tuple produced the maximum accuracy which is reported in Table 5.4. At the beginning of the training, a much bigger improvement in accuracy has been observed with a tuple which is different than the highest performing hyper-parameter tuple. A network is trained separately from scratch using the highest performing tuple until convergence which produces an error rate of 3.19 which is close to the one mentioned in Table 5.3. This approach is repeated for the rest of the datasets which produces the accuracy close to the one reported in Table 5.4 with a marginal difference range of ± 0.15. It depicts the effectiveness of the reported highest performing hyper-parameters tuple in the shortest time.

Figure 5.4 shows the policy loss, reward and network validation accuracy of the 5 datasets. The plots show a vertical line along y-axis representing maximum accuracy. The best hyper-parameters found against each dataset are reported in Table 5.4 along with the

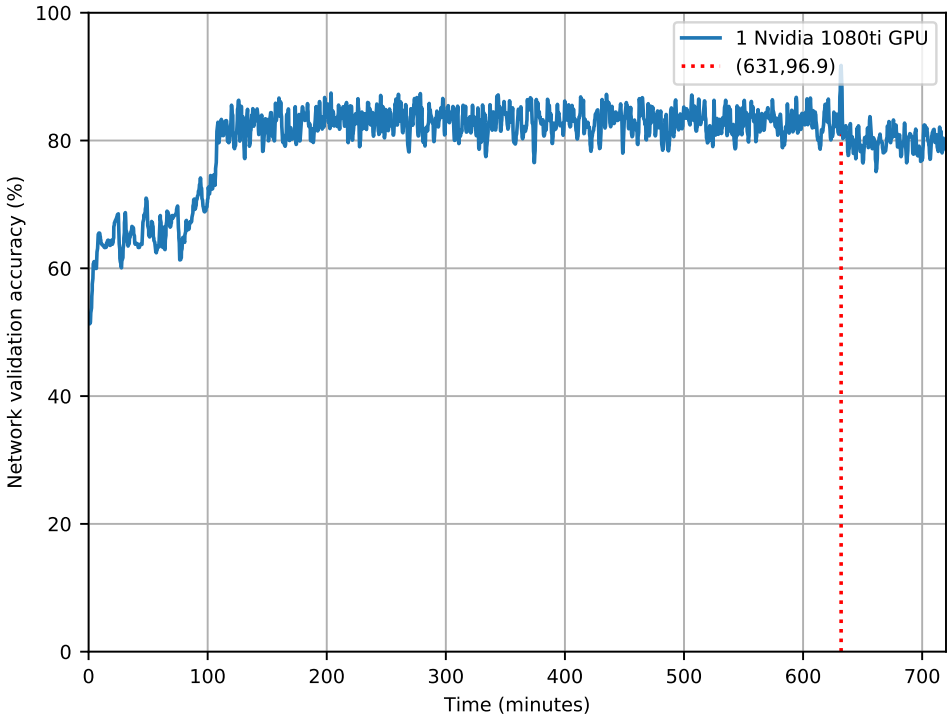

Figure 5.3: Cifar-10 time taken versus network validation accuracy plot

exploration and network accuracy information. The mnist and fashion-mnist tasks took very few episodes to find the top performing hyper-parameters. On the contrary, the complex tasks, cifar-100 and tiny-imagenet, took many more episodes to try different permutations of the hyper-parameters.

Table 5.4: Accuracy of various datasets including optimal parameters and episodes required to achieve the optimal value

Dataset	**Network Hyper-parameters** $[D, DR, \alpha, p]$	**Episodes**	**Duration (hours)**	**Network Error (%)**
Mnist	[4, 0.06, 0.02, 0.95]	720	0.72	1.71
Fashion-mnist	[4, 0.06, 0.02, 0.95]	466	0.36	4.63
Cifar-10	[4, 0.3, 0.006, 0.95]	7,203	10.53	3.11
Cifar-100	[11, 0.2, 0.0007, 0.93]	9,810	19.39	23.06
Tiny-imagenet	[16, 0.25, 0.0004, 0.89]	13,770	36.83	35.61

The network hyper-parameters are initialized once and tuned after every 50 steps. The policy gradient took more episodes to learn the hyper-parameters for more complex tasks. To evaluate a recommended tuple, 50 steps are very limited, hence the behaviour of the network was captured and provided to the meta-learner to more fully observe the impact of the tuple.

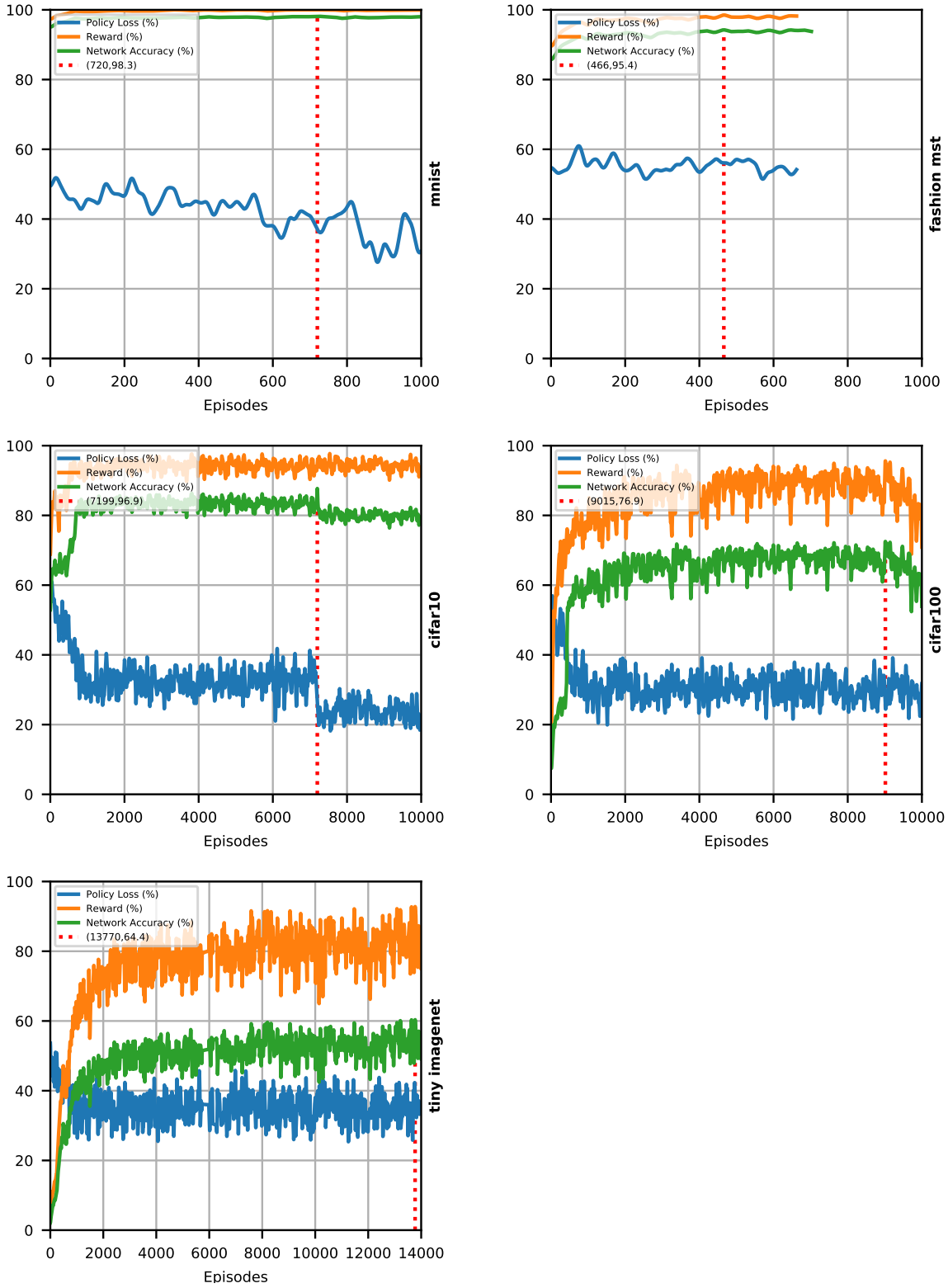

Figure 5.4: Statistics of different datasets including policy loss, reward and network accuracy

5.5 Summary

This study has presented an efficient approach to hyper-parameters search of deep models. It uses RL search strategy to explore the space of deep neural architectures. It addresses the Hyper-parameter Optimization (HPO) of DNN research challenges, which is defined in Section 2.7, in the context of NAS. A Policy-based RL method is used to generate a tuple of hyper-parameters. The tuple is used by the target network, base-learner, which is initialized once with random hyper-parameters and, often, tunes on every episode. In each episode, a validation accuracy has been computed after training for 50 steps with a batch size of 32. The delta of the accuracy, which is referred to as reward, is fed back to the policy network along with the behaviour of the environment. The attributes that represent behaviour are training loss and statistics of the target network's final layer outcome. A more refined tuple of hyper-parameters, in turn, is generated for the next episode. This cycle tunes the parameters of the policy network and hyper-parameters of the network at the same time which makes the overall process more computationally efficient than the existing approaches.

In conclusion, the proposed approach demonstrates a quick and effective hyper-parameter search approach. Unlike previous studies, it is equally effective for both small and large datasets. Although the exploration takes more time if the range of the network depth parameter gets bigger, still using one GPU the exploration takes less than a day for a complex task. This approach is 20% less computation expensive than ENAS with marginally higher error-rate. The depth hyper-parameter is found to be the most effective one where the change of the depth causes a significant jump in the accuracy. There are many possible directions for future work. Currently, only four hyper-parameters are part of the search space which can be enhanced. Accordingly, to evaluate the effectiveness of a tuple of hyper-parameters, state of the intermediary layers of the network can be observed rather than only the statistics of final layer outputs.

Chapter 6

Conclusions and Future Work

The main theme of this thesis was to investigate and explore Meta-level Learning (MLL) strategies and approaches for effective reduction of the predictive model search space. In particular, the two areas of focus are Meta-level feature engineering for problem representation and learning strategy for algorithm and its hyper-parameters recommendation.

The initial chapters of the thesis present the key challenges of an MLL system in general. A thorough study of existing MLL systems has been conducted which leads to several research questions. It thus resulted in narrowing down the scope of this research. The literature review covers a detailed study of five key components of an MLL system. These components include Examples of Datasets (EoD), Meta-features (MFs) generation and selection, Base-level Learning (BLL), MLL and adaptive mechanism. Moreover, various methods to gather EoD are discussed where all of them have some limitations. Considering those limitations, gathering datasets, MFs and performance measures of Base-models from the published research appeared as most appealing for the current research. Similarly, a comprehensive review of feature generation and selection techniques revealed that most of them are suitable for a stationary MLL system. Hence a lot of effort is required to evaluate the features that are proposed by stationary systems along with finding new features to represent a non-stationary problem at Meta-level. An MLL system also expects performance measures of Base-models on EoD. It comes out as the most time and processor intensive task to compute performance measures of a large number of EoD against learning algorithms and their numerous configurations. Thus extracting MFs and performance measures from existing Machine Learning (ML) publications minimize the complexity of this problem, which requires most of the effort and resources. A number of MLL systems are discussed in detail which include the application of MLL to both supervised and unsupervised learning problems. The evolution of MLL field since the last three decades has been discussed and various systems are compared with the previous ones.

In the last few years, the emergence of Deep Learning (DL) naturally shifted the MLL domain from models selection towards hyper-parameters and (neural) architecture search. The Hyper-parameter Optimization (HPO) finds a set of hyper-parameters of an ML task which gives optimal performance. The success of the Deep Neural Networks (DNN) is mostly credited to its ability to automatically extract the task-dependent features. This automation is now expanding towards architecture engineering to automatically design complex neural architectures; this approach is known as Neural Architecture Search (NAS). NAS hides

most of the steps of the 'traditional meta-learning' pipeline including EoD and manual MF extraction and generation.

A thorough literature review that leads to several research questions is outlined as follows:

6.1 Research Challenges

The goal of this work is research on MLL strategies and approaches for effective reduction of the model search space. Three areas of evolving predictive systems were identified where the applicability of MLL can be an effective and efficient approach.

1. A Learning Path Recommendation: An optimal learning path recommendation of the three interlinked components includes pre-processing steps, learning algorithms or their combination, and adaptivity mechanism parameters.

2. Recurring Concepts Extraction: In a non-stationary environment, the underlying distribution of the incoming data keeps changing which in turn makes the most recent historical concept ineffective. A MLL system can extract the relevant concepts of data to adapt the out-date model.

3. Concept Drift Detection: In an adaptive mechanism retraining of model is usually triggered by a change detection process. MLL can help to automatically detect the concept drift and trigger the algorithm retraining process instantly.

These areas lead to several research questions which are outlined as follows:

1. Gathering examples of datasets to build a static Meta-knowledge database

2. Base-level Learning strategy to compute performance measures of Meta-examples

3. Feature generation and selection to represent a problem at Meta-level

4. Representation and storage of dynamically growing complex Meta-Knowledge database

5. Meta-level Learning strategy for algorithm and its hyper-parameter recommendation

From the above five research questions, 3 and 5 were addressed in this research.

The rest of the chapters cover experimental evaluation of the three studies conducted around model selection and hyper-parameter search using MLL. These studies address one or more research challenges.

6.2 Main Findings and Contributions

The original contributions of this work are:

1. Formulation of Model Selection and Hyper-parameters Optimization (MSHPO):
   The formulation of MSHPO along with three key areas of an evolving predictive system which leads to several research challenges.

2. Cross-domain MLL for Time-series (TS) forecasting:
   This work covers an effort towards the experimental evaluation of cross-domain MLL which started building an understanding of the complexity of an MLL system. The key focus of the study was to investigate whether the use of additional training data from a different domain is beneficial for improving the performance of the Meta-learner. The MLL was not always giving the highest possible performance, however, it helps to alleviate the risk of selecting the worst model and hence, tends to be a robust approach.

3. Towards MLL of deep architectures for domain adaptation:
   An empirical study to investigate the relationship between various characteristics describing the similarity of two datasets, and based on that, the number of layers of a deep model pre-trained on a dataset need to be fine-tuned to achieve close to state-of-the-art accuracy. Although the experiments were limited to only two characteristics of the new task, size and similarity with the original task, still these are the most important features in this context. This study preserves the key characteristics of transfer learning, particularly, for less similar tasks towards experimentally verified intuition that one can more effectively reuse pre-trained network. The proposed approach was limited to only one hyper-parameter, network depth, which was insufficient for DNN considering the wide range of hyper-parameter choices.

4. A Meta-Reinforcement Learning (Meta-RL) approach to optimize parameters and hyper-parameters simultaneously:
   The final study presents an efficient approach to NAS, by optimizing parameters and hyper-parameters of a network simultaneously. A Policy-based Reinforcement Learning (RL) method was used to generate a tuple of hyper-parameters. The tuple was used by the target network, base-learner, which was initialized once with random hyper-parameters and, often, tunes on every episode. The 'network depth' hyper-parameter was the most effective one among others where the change of the depth causes a significant jump in the accuracy. The proposed approach was found to be 20% less computationally expensive than Efficient Neural Architecture Search (ENAS), an existing system, with marginally higher error-rate.

6.3 Future Research

The ML algorithm and its underlying model structure are well studied using MLL techniques so that this process can be automated. This research direction has already achieved remarkable success based on the use of different optimization techniques (Brazdil and Giraud-

Carrier, 2018). Thus, the feature research direction will revolve around two areas of MLL investigated in this thesis.

RL, particularly policy gradient, provides an intrinsic mechanism of meta-exploration algorithm (Gupta et al., 2018). The MLL that is applied to RL algorithms is known as Meta-RL. The objective of Meta-RL is to learn a policy along with an RL agent. The success of Meta-RL based approaches to automatically design Convolutional Neural Network (CNN) architectures has been proven particularly for image classification tasks. However, it is achieved at a very high cost because the learner network needs to come up with its learning strategy from scratch. Based on these facts there are some potential future research directions, given as follows:

- Addressing the discrepancy of the learning strategy of Meta-RL by making the meta-exploration process efficient using extensions of approaches like weights sharing (Bender et al., 2018) and the one addressed in this thesis. The combination of these approaches may lead to a potential solution to the limitations raised in Chapter 5.

- The study discussed in Chapter 4 finds the amount of fine-tuning a pre-trained network requires based on the characteristics of the original and new tasks. This work can be extended by computing the domain similarity of two tasks from the behaviour of network activations while transferring knowledge rather than relying only on the probability distribution of the final layer outcome over the classes.

- The study discussed in Chapter 5 evaluates the effectiveness of a tuple of hyper-parameters by training it for very few steps rather than till convergence of the network which saves a significant amount of computation. Therefore, the impact of the training of a tuple needs to be analyzed very closely which, in-turn, evaluates the effectiveness of the tuple and suggests a tuned tuple for next training iteration. Incorporating the behaviour of the network activations while computing the reward may boost the effectiveness of the proposed approach.

- Going beyond image classification task by exploring domains like language modeling (Zoph and Le, 2017), speech processing (Wang and Zheng, 2015), network compression (Ashok et al., 2018). Most of the HPO and NAS methods gain tremendous success in the computer vision domain, particularly in the image classification task. These methods could be extended for speech and language processing domain.

- Recurrent Neural Network (RNN) is difficult to train as compared to CNN, so extending the Meta-RL based HPO approach for RNN could be another potential future direction (Hutter et al., 2018). This future direction is in-line with the previous one as RNN is mostly targeted for speech and language processing domain.

Appendix A

Definitions

Definitions of the Statistical concepts used in the literature are stated below (Duda and Hart, 1973; Bishop and Hart, 1995; Cressie, 1993; Li, 1995; Sutton et al., 1999; Sutton and Barto, 2015):

- **Statistical Learning**
  An approach to machine intelligence which is based on statistical modeling of data. With a statistical model in hand, one applies probability theory and decision theory to get an algorithm.

- **Classification**
  Assigning a class to a measurement, or equivalently, identifying the probabilistic source of a measurement. The only statistical model that is needed is the conditional model of the class variable given the measurement. This conditional model can be obtained from a joint model or it can be learned directly.

- **Regression**
  Predicting the value of random variable y from measurement x.

- **Nonparametric regression/density estimation**
  An approach to regression/density estimation that doesn't require much prior knowledge but only a large amount of data. For regression, it includes nearest-neighbor, weighted average and locally weighted regression. For density estimation, it includes histograms, kernel smoothing and nearest-neighbor.

- **Parameter Estimation**
  Density estimation when the density is assumed to be in a specific parametric family.

- **Principal Component Analysis**
  Constructing new features which are the principal components of a data set. The principal components are random variables of maximal variance constructed from linear combinations of the input features. Equivalently, they are the projections onto the principal component axes, which are lines that minimize the average squared distance to each point in the data set. To ensure uniqueness, all of the principal component axes must be orthogonal. PCA is a maximum-likelihood technique for linear regression in the presence of Gaussian noise on both inputs and outputs.

- **Clustering**
  Grouping similar objects in a multidimensional space. Some algorithms, like k-means, simply partition the feature space. Other algorithms, like single-link agglomeration, create nested partitionings which form a taxonomy.

- **K-means**
  A parametric algorithm for clustering data into exactly k clusters.

- **Feature Selection**
  Not extracting new features but rather removing features which seem irrelevant for modeling. This is a combinatorial optimization problem.

- **Linear Regression**
  A conditional statistical model of random vector y given measurement vector x, where y is a linear transformation of x followed by additive noise.

- **Radial Basis Function regression**
  Basis function regression where each new feature is based on the distance to a prototype, hence the basis is *radial*. The basis functions are adapted by moving the prototypes or reshaping the bumps.

- **Time-series**
  A time series is a sequence of observations which are ordered in time (or space).

  - **Continuous**
    Continuous time-series is the one where an observation at every instant of time, e.g., lie detectors, electrocardiograms. We denote this using observation X at time t, X^t.
  - **Discrete**
    Where we have an observation at (usually regularly) spaced intervals. We denote this as X^t.

- **Feed-forward Neural Network Regression**
  Basis function regression with adaptive basis functions. Given a measurement vector, each layer of the network makes a linear transformation and then applies a non-linearity to each vector component.

- **Back-propagation**
  A method for maximum likelihood estimation of a feed-forward neural network. It is equivalent to steepest-descent optimization.

- **Support Vector Machine**
  A generalized linear classifier with a maximum-margin fitting criterion. This fitting criterion provides regularization which helps the classifier generalize better. The classifier tends to ignore many of the features.

- **Autoregression**
  A Markov chain model for a sequence of variables, where the next variable is predicted from the previous variable via regression, typically linear.

- **Maximum likelihood**
  A parameter estimation heuristic that seeks parameter values that maximize the likelihood function for the parameter.

- **Maximum A Posteriori**
  A parameter estimation heuristic that seeks parameter values that maximize the posterior density of the parameter.

- **Bootstrapping**
  A technique for simulating new data sets, to assess the robustness of a model or to produce a set of likely models. The new data sets are created by re-sampling with replacement from the original training set, so each datum may occur more than once.

- **Bagging**
  Generate a bunch of models via bootstrapping and then average their predictions.

- **Boosting**
  A technique for combining models based on adaptive resampling: different data is given to different models. The idea is to successively omit the 'easy' data points, which are well modeled, so that the later models focus on the 'hard' data.

- **Cross-validation**
  A method for evaluating a statistical model or algorithm that has free parameters. Divide the training data into several parts, and in turn use one part to test the procedure fitted to the remaining parts. It can be used for model selection or for parameter estimation when there are many parameters.

- **Neural Networks**
  Neural networks are a class of models that are built with layers. Commonly used types of neural networks include convolutional and recurrent neural networks.

- **Activation function**
  Activation functions are used at the end of a hidden unit to introduce non-linear complexities to the model. Here are the most common ones:

  - Sigmoid: $g(z) = \frac{1}{1+e^{-z}}$
  - Tanh: $g(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$
  - Rectified Linear Units (ReLU): $g(z) = max(0, z)$

- **Cross-entropy loss**
  In Neural Networks, the cross-entropy loss $L(z, y)$ is commonly used as:
  $L(z, y) = -[y \log(z) + (1 - y) \log(1 - z)]$

- **Learning rate**
  The learning rate, often represented as , indicates at which pace the weights get updated.

- **Back-propagation**
  Backpropagation is a method to update the weights in the neural network by taking into account the actual output and the desired output. The derivative with respect to weight w is computed using chain rule:
  $\frac{\partial L(z,y)}{\partial w} = \frac{\partial L(z,y)}{\partial a} * \frac{\partial a}{\partial z} * \frac{\partial z}{\partial w}$

- **Dropout**
  Dropout is a technique meant at preventing over-fitting the training data by dropping out units in a neural network with probability p or kept with probability $(1 - p)$.

- **Batch normalization**
  It is a step of hyper-parameter γ, β that normalizes the batch x_i, the mean and variance of that we want to correct to the batch.

- **Long Short-term Memory**
  A long short-term memory (LSTM) network is a type of RNN model that avoids the vanishing gradient problem by adding 'forget' gates.

- **Reinforcement Learning**
  An algorithm, or agent, learns by interacting with its environment. The agent receives rewards by performing correctly and penalties for performing incorrectly. The agent tries to maximize total reward.

- **Meta-Reinforcement Learning**
  Meta-learning applied on reinforcement learning is known as Meta-Reinforcement Learning.

- **Agent**
  A system that is embedded in an environment and takes actions to change the state of the environment.

- **Discount factor**
  A scalar value between 0 and 1 which determines the present value of future rewards. If the discount factor is 0, the agent is concerned with maximizing immediate rewards. As the discount factor approaches 1, the agent takes more future rewards into account.

- **Dynamic programming**
  Dynamic Programming (DP) is a class of solution methods for solving sequential decision problems with a compositional cost structure. Richard Bellman was one of the principal founders of this approach.

- **Environment**
  The external system that an agent is 'embedded' in, and can perceive and act on.

- **Markov decision process**
  A probabilistic model of a sequential decision problem, where states can be perceived exactly, and the current state and action selected determine a probability distribution on future states. Essentially, the outcome of applying an action to a state depends only on the current action and state.

- **Model**
  The agent's view of the environment, which maps state-action pairs to probability distributions over states. Note that not every reinforcement learning agent uses a model of its environment.

- **Monte Carlo methods**
  A class of methods for learning of value functions, which estimates the value of a state by running many trials starting at that state, then averages the total rewards received on those trials.

- **Policy**
  The decision-making function (control strategy) of the agent, which represents a mapping from situations to actions. It is a function $\pi : S \longrightarrow A$ that maps states to actions.

- **Reward**
  A scalar value which represents the degree to which a state or action is desirable. Reward functions can be used to specify a wide range of planning goals, for example, an agent can be guided towards learning the fastest route to the final state.

- **State**
  State can be viewed as a summary of the past history of the system that determines its future evolution.

- **Value function**
  Value function is a mapping from states to real numbers, where the value of a state represents the long-term reward achieved starting from that state and executing a particular policy.

Appendix B

Meta-features

Table B.1: Meta-features used in various studies

Meta-Features	Rendell et al. (1987) Rendell and Cho (1990)	King et al. (1995)	Sohn (1999)	Lindner and Studer (1999) Berrer et al. (2000) Giraud-Carrier (2005)	Bensusan et al. (2000)	Bensusan and Giraud-Carrier (2000)	Pfahringer et al. (2000)	Todorovski et al. (2002)	Peng et al. (2002)	Kopf and Iglezakis (2002)	Brazdil et al. (2003)	Prudencio and Ludermir (2004)	Prudencio and Ludermir (2008) Guerra et al. (2008)	Wang et al. (2009)	Lemke and Gabrys (2010a)	Abdelmessih et al. (2010)	Rossi et al. (2012)	Feurer et al. (2014)	Filchenkov and Pendryak (2015)	Ali et al. (2018)
Descriptive Meta-features																				
Number of Classes (k)		✔	✔	✔			✔											✔	✔	
Frequency of most common class	✔						✔													
Number of Features (p)	✔[1]	✔	✔															✔	✔	
Total Instances (N)		✔	✔	✔			✔				✔	✔			✔			✔	✔	✔
Dataset Dimentionality																		✔	✔	
Number of Training instances (r)	✔[1]	✔	✔										✔[2]							
Number of Test instances (t)	✔	✔	✔																	
Sampling Distribution	✔	✔																		
Number of Binary Features (b)	✔	✔	✔																	
Number of Numeric features (n)	✔			✔			✔											✔		
Number of Nominal features (s)	✔			✔			✔											✔		
Proportion of binary features (b/p)			✔																	
Proportion of nominal features (s/p)				✔							✔							✔		
Span of nominal values				✔																
Average of nominal values				✔														✔		
Training instances to features ratio (N/p)			✔										✔[2]							
Proportion of training instances (r/N)			✔																	
Statistical Meta-features																				
Relative probability of missing values				✔							✔							✔		
Instances with missing values				✔														✔		
Proportion of features with outliers				✔							✔						✔			
Mean Skewness (SKEW)		✔	✔	✔										✔	✔[3]		✔			✔[3]
Mean Kurtosis (KURT)		✔	✔	✔										✔	✔[4]		✔		✔	✔
Average																	✔			
Variance																	✔			
Minimum																	✔			
Maximum																	✔			
Median																	✔			
Correlation between predictor and target																	✔			
Standard Deviation (StdDev) of the class distribution				✔											✔[5]			✔	✔	✔[6]
Homogeneity of Covariances (S/D Ratio)		✔	✔	✔																
Canonical Correlation (CANCOR)		✔	✔	✔							✔									

[1] only these two features are used in Rendell et al. (1987), they are also part of Rendell and Cho (1990)
[2] Log
[3] of series
[5] of de-trended series

Number of Discriminant Functions (DiscFunc)				✔															
Mean Absolute Correlation Coefficient (CORR)		✔	✔															✔	
Relative proportion of largest Eigenvalue (FRACT)		✔	✔	✔															
Wilks'lambda Distribution (Wlambda)				✔															
Default Accuracy				✔															
coefficient of variation (COEFVAR)											✔							✔	
absolute value of the SKEW and KURT coefficient											✔						✔	✔	
Time-series (TS) mean absolute values of first 5 auto-correlations (Mean-CORR)											✔								
TS test of significant auto-correlations (TAC)											✔								
TS significance of the 1, 2, and 3 Auto-correlation (TAC-1,2,3)											✔								
TS test of Turning Points for randomness											✔								
TS first coefficient of auto-correlation (AC1)											✔								
TS type											✔								
TS trend											✔		✔	✔[7]					✔[8]
TS turning point											✔[9]			✔					✔
TS Durbin-Watson statistic of regression residual (DW)														✔					
TS step changes														✔					✔
TS predictability measure														✔					
TS non-linearity measure														✔					✔
TS largest Lyapunov exponent													✔	✔					
TS 3 largest power spectrum frequencies														✔					
TS maximum value of power spectrum														✔					✔
TS number of peaks > 60%														✔					✔
TS auto-correlations at lags 1 and 2														✔					✔
TS partial auto-correlations at lags 1 and 2														✔					✔
TS seasonality Measure													✔	✔					✔
TS mean Symmetric Mean Absolute Percentage Error (SMAPE) - mean deviated SMAPE														✔					
TS mean SMAPE / mean deviated SMAPE														✔					
TS mean of correlation coefficient														✔					
TS StdDev of correlation coefficient														✔					
TS methods in top performing cluster														✔					
TS distance top performing cluster to second best														✔					
TS Serial CORR Box-Pierce statistic													✔[10]						
TS Non-linear autoregressive structure													✔[11]						
TS Self-similarity (Long-range Dependence													✔						
TS Periodicity (frequency)													✔						
Min. of CORR between predictors and target												✔							
Max. of CORR between predictors and target												✔							
Mean of CORR between predictors and target												✔							
StdDev of absolute value of CORR between predictors and target												✔							
Min. of CORR between pairs of predictors												✔							
Max. of CORR between pairs of predictors												✔							
Mean of CORR between pairs of predictor												✔							
StdDev of absolute value of CORR between pairs of predictors												✔							
Information Theoretic Meta-features																			
Entropy of Classes (HC)				✔	✔		✔					✔							✔	
Entropy of nominal features				✔	✔		✔												✔	
Joint Entropy of Classes (HCX)				✔	✔		✔												✔	
Average Mutual Information between Class and Nominal Features (MCX)				✔	✔		✔					✔							✔	
Class Entropy to Mutual information ratio					✔		✔											✔	✔	
Noise to Signal Ratio (NoiseRaio)	✔			✔			✔												✔	
Dispersion Gain																	✔			

[7] StdDev of series / StdDev of de-trended series

[9] ratio

[10] of raw and trend/seasonally adjusted

[11] of raw and trend/seasonally adjusted

Landmarkers																				
Decision Nodes Learner (Decision Nodes)						✔		✔							✔		✔			
Worst Nodes Learner (Worst Nodes)						✔									✔					
Randomly Chosen Nodes Learner (Randomly Chosen Nodes)						✔									✔		✔			
Naive Bayes classifier (NB)						✔		✔							✔		✔			
k-Nearest Neighbour (k-NN)				✔[12]		✔[13]	✔	✔[13]		✔					✔[13]		✔	✔		
Elite-Nearest Neighbour (e-NN)						✔														
Linear Discriminant Analysis (LDA)						✔		✔							✔		✔			
C5.0 Decision Tree (C5.0 tree)								✔									✔	✔		
C5.0 Adaptive Boosting (C5.0 boost)							✔	✔												
C5.0 Rule Induction (C5.0 rules)							✔	✔		✔										
Rule Learner (Ripper)							✔													
Linear Discriminant Trees (Ltree)							✔			✔										
Average Nodes Learner (Average Nodes)															✔					
Model-based Meta-features																				
Nodes per attribute					✔															
Nodes per instance					✔															
Average leaf corroboration					✔															
Average gain-ratio difference					✔															
Maximum depth					✔														✔	
No. of repeated nodes					✔															
Shape					✔														✔	
Homogeneity					✔															
Imbalance					✔															
Internal symmetry					✔															
No. of Nodes in each level - width									✔										✔	
No. of levels - Height									✔										✔	
No. of nodes in the tree									✔										✔	
No. of leaves in the tree									✔										✔	
Maximum no. of nodes at one level									✔										✔	
Mean of the no. of nodes									✔										✔	
StdDev of the no. of nodes									✔										✔	
Length of the Shortest branch									✔											
Length of the Longest branch									✔											
Mean of the branch length									✔											
StdDev of the branch length									✔										✔	
Minimum occurrence of Features									✔										✔	
Maximum occurrence of Features									✔										✔	
Mean of the no. of occurrences of Features									✔										✔	
StdDev of no. of occurrences of Features									✔										✔	
Weight sum of dataset																			✔	
Minimum weight sum of dataset																			✔	
Average weight sum of dataset																			✔	
StdDev weight sum of dataset																			✔	
No. neighbours for dataset																			✔	
Minimum No. neighbours for dataset																			✔	
Maximum No. neighbours for dataset																			✔	
Average No. neighbours for dataset																			✔	
StdDev of No. neighbours for dataset																			✔	
Principal Component Analysis (PCA) 95%																	✔			
PCA skewness																	✔			
PCA kurtosis																	✔			
Total Meta-features	9	13	19	25	10	14	8	7	15	3	7	11	10	9	23	7	10	22	40	13

[12] k = 3 used only in Giraud-Carrier (2005)

[13] k = 1

Appendix C

Summary of Literature Review

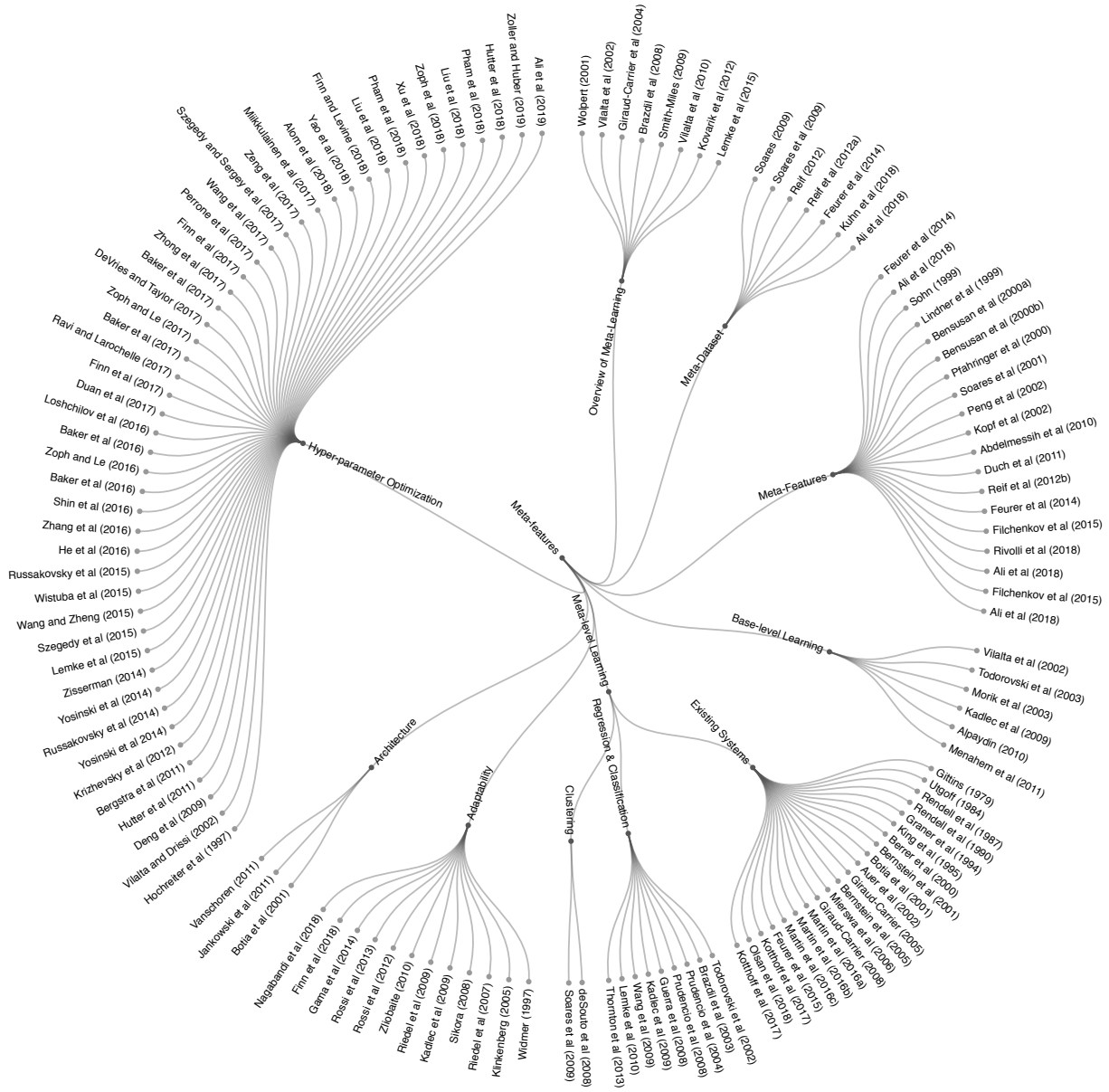

Figure C.1: Area-wise distribution of Publications

References

Abdelmessih, Sarah D., Faisal Shafait, Matthias Reif, and Markus Goldstein (2010). "Landmarking for Meta-Learning using RapidMiner". In: *RapidMiner Community Meeting and Conference*. Online.

Aha, David W., Dennis Kibler, and Marc K. Albert (1991). "Instance-based Learning Algorithms". In: *Machine Learning*, pp. 37–66.

Ali, Abbas Raza, Bogdan Gabrys, and Marcin Budka (2018). "Cross-domain Meta-learning for Time-series Forecasting". In: *Procedia Computer Science, Elsevier* 126, pp. 9–18.

Ali, Abbas Raza, Marcin Budka, and Bogdan Gabrys (2019a). "A Meta-Reinforcement Learning Approach to Optimize Parameters and Hyper-parameters Simultaneously". In: *Proceedings of the 16th Pacific RIM International Conference on Artificial Intelligence (PRICAI)*.

Ali, Abbas Raza, Marcin Budka, and Bogdan Gabrys (2019b). "Towards Meta-level Learning of Deep Neural Networks for Fast Adaptation". In: *Proceedings of the 16th Pacific RIM International Conference on Artificial Intelligence (PRICAI)*.

Alom, Zahangir, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, and et al. (2018). "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches". In: *Computing Research Repository (CoRR)* abs/1803.01164.

Alpaydin, Ethem (2010). *Introduction to Machine Learning*. 2nd. The MIT Press.

Ashok, A., N. Rhinehart, F. Beainy, and K. M. Kitani (2018). "N2N learning: Network to network compression via policy gradient reinforcement learning". In: *International Conference on Learning Representations (ICLR)*.

Auer, Peter, Nicolo Cesa-Bianchi, and Paul Fischer (2002). "Finite-time analysis of the multiarmed bandit problem". In: *Machine learning* 47 (2).

Bache, Kevin and Moshe Lichman (2013). *UCI Machine Learning Repository*. URL: `http://archive.ics.uci.edu/ml`.

Baker, Bowen, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar (2016). "Designing Neural Network Architectures using Reinforcement Learning". In: *Computing Research Repository (CoRR)* abs/1611.02167.

Baker, Bowen, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar (2017). "Designing Neural Network Architectures using Reinforcement Learning". In: *Computing Research Repository (CoRR)* abs/1611.02167.

Bakirov, Rashid, Bogdan Gabrys, and Damien Fay (2018). "Generic adaptation strategies for automated machine learning". In: *Computing Research Repository (CoRR)* abs/1812.10793.

Bender, Gabriel, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le (2018). "Understanding and simplifying one-shot architecture search". In:

Bensusan, Hilan and Christophe G. Giraud-Carrier (2000). "Discovering Task Neighbourhoods Through Landmark Learning Performances". In: *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*. London, UK, UK: Springer-Verlag, pp. 325–330.

Bensusan, Hilan and Alexandros Kalousis (2001). "Estimating the Predictive Accuracy of a Classifier". In: *Proceedings of the 12th European Conference on Machine Learning (EMCL)*. London, UK: Springer-Verlag, pp. 25–36.

Bensusan, Hilan, Christophe G. Giraud-Carrier, and Claire J. Kennedy (2000). "A Higher-order Approach to Meta-learning". In: *Proceedings of the ECML workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*, 109–117.

Bergstra, J., R. Bardenet, Y. Bengio, and B. Keegl (2011). "Practical Network Blocks Design with Q-Learning". In: *Proceedings of the 25th International Conference on Advances in Neural Information Processing Systems*.

Bernstein, Abraham and Foster Provost (2001). "An Intelligent Assistant for the Knowledge Discovery Process". In: *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI) Workshop on Wrappers for Performance Enhancement in KDD*. Seattle, Washington, USA.

Bernstein, Abraham, Foster Provost, and Shawndra Hill (2005). "Toward Intelligent Assistance for a Data Mining Process: An Ontology-Based Approach for Cost-Sensitive Classification". In: *IEEE Transactions on Knowledge and Data Engineering* 17.4, pp. 503–518.

Berrer, Helmut, Iain Paterson, and Jorg Keller (2000). "Evaluation of Machine-Learning Algorithm Ranking Advisors". In: *Proceedings of the PKDD-2000 Workshop on DataMining, Decision Support, Meta-Learning and ILP: Forum for Practical Problem Presentation and Prospective Solutions*.

Bishop, C. and P. E. Hart (1995). "Neural Networks for Pattern Recognition". In:

Bossard, Lukas, Matthieu Guillaumin, and Luc J. Van Gool (2014). "Food-101 - Mining Discriminative Components with Random Forests." In: *ECCV (6)*. Vol. 8694. Lecture Notes in Computer Science. Springer, pp. 446–461.

Botia, Juan A., Antonio F. Gomez-Skarmeta, Mercedes Valdes, and Antonio Padilla (2001). "METALA: A Meta-learning Architecture". In: *Proceedings of the International Conference, 7th Fuzzy Days on Computational Intelligence, Theory and Applications*. London, UK: Springer-Verlag, pp. 688–698.

Box, George and Gwilym Jenkins (1970). "Time Series Analysis". In:

Brazdil, Pavel and Christophe Giraud-Carrier (2018). "Metalearning and Algorithm Selection: Progress, State of the Art and Introduction to the 2018 Special Issue". In: *Machine Learning* 107.1, pp. 1–14.

Brazdil, Pavel, Christophe Giraud-Carrier, Carlos Soares, and Ricardo Vilalta (2008). *Metalearning: Applications to Data Mining*. 1st ed. Springer Publishing Company, Incorporated.

Brazdil, Pavel B., Carlos Soares, and Joaquim Pinto Da Costa (2003). "Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results". In: *Journal of Machine Learning* 50.3, pp. 251–277.

Chawla, Nitesh V., Kevin Bowyer, Lawrence Hall, and Philip Kegelmeyer (2002). "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16, pp. 321–357.

Cressie, N. A. C. (1993). "Statistics for spatial data". In:

Crone, Sven (2006). *NN3 Forecasting Competition [Online]*. URL: `http://www.neural-forecasting-competition.com/NN3`.

Crone, Sven (2008). *NN5 Forecasting Competition [Online]*. URL: `http://www.neural-forecasting-competition.com/NN5`.

Crone, Sven (2010). *NN-GC1 Forecasting Competition [Online]*. URL: `http://www.neural-forecasting-competition.com`.

De-Souto, Marcilio, Ricardo Bastos Cavalcante Prudencio, Rodrigo Soares, and et al. (2008). "Ranking and selecting clustering algorithms using a meta-learning approach". In: *IEEE International Joint Conference on Neural Networks*, pp. 3729–3735.

Demner-Fushman, Dina, M. D. Kohli, M. B. Rosenman, S. E. Shooshan, and et al. (2016). "Preparing a collection of radiology examinations for distribution and retrieval". In: *Journal of the American Medical Informatics Association* 23.2.

Deng, Jia, Wei Dong, Richard Socher, Li-jia Li, and et al. (2009). "Imagenet: A large-scale hierarchical image database". In: *In CVPR*.

DeVries, Terrance and Graham W. Taylor (2017). "Improved regularization of convolutional neural networks with cutout". In: *Computing Research Repository (CoRR)* abs/1708.04552.

Duan, Yan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel (2016). "RL2: Fast Reinforcement Learning via Slow Reinforcement Learning". In: *Computing Research Repository (CoRR)* abs/1611.02779.

Duch, Wlodzislaw, Tomasz Maszczyk, and Marek Grochowski (2011). "Optimal Support Features for Meta-Learning". In: *Meta-Learning in Computational Intelligence*. Vol. 358. Studies in Computational Intelligence. Springer, pp. 317–358.

Duda, R. O. and P. E. Hart (1973). "Pattern Classification and Scene Analysis". In:

eLICO (2012). *An e-Laboratory for Interdisciplinary Collaborative Research in Data Mining and Data-Intensive Sciences*. URL: `http://www.e-lico.eu`.

Fei-Fei, Li, Rob Fergus, and Pietro Perona (2007). "Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories". In: *Comput. Vis. Image Underst.* 106.1, pp. 59–70. ISSN: 1077-3142.

Feurer, Matthias, Jost Tobias, and Frank Hutter (2014). "Using meta-learning to initialize bayesian optimization of hyperparameters". In: *Proceedings of the International Conference on Meta-learning and Algorithm Selection (MLAS)*, pp. 3–10.

Feurer, Matthias, Aaron Klein, Katharina Eggensperger, Jost Springenberg, and et al. (2015). "Efficient and Robust Automated Machine Learning". In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, pp. 2962–2970.

Filchenkov, Andray and Arseniy Pendryak (2015). "Dataset metafeature description for recommending feature selection". In: *ISMW FRUCT*, pages 11–18.

Finn, Chelsea and Sergey Levine (2018). "Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm". In: *Computing Research Repository (CoRR)* abs/1710.11622.

Finn, Chelsea, Pieter Abbeel, and Sergey Levine (2017). "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. International Convention Centre, Sydney, Australia: PMLR, pp. 1126–1135.

Frank, Eibe and Ian H. Witten (1998). "Generating Accurate Rule Sets Without Global Optimization". In: *Fifteenth International Conference on Machine Learning*, pp. 144–151.

Gama, J. and P. Brazdil (1995). "Characterization of Classification Algorithms". In:

Gama, João, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia (2014). "A Survey on Concept Drift Adaptation". In: *ACM Comput. Surv.* 46.4.

Gama, Joao, Pedro Medas, Gladys Castillo, and Pedro Rodrigues (2004). "Learning with Drift Detection". In: *Advances in Artificial Intelligence* 3171.10, pp. 286–295.

Genuer, Robin, Jean-Michel Poggi, and Christine Tuleau-Malot (2010). "Variable Selection Using Random Forests". In: *Pattern Recogn. Lett.* 31.14, pp. 2225–2236.

Giraud-Carrier, Christophe (2005). "The Data Mining Advisor: Meta-learning at the Service of Practitioners". In: *Proceedings of the Fourth International Conference on Machine Learning and Applications (ICMLA)*. Washington, DC, USA: IEEE Computer Society, pp. 113–119.

Giraud-Carrier, Christophe (2008). "Meta-learning - A Tutorial". In: *Proceedings of the Seventh International Conference on Machine Learning and Applications (ICMLA)*. San Diego, CA, USA.

Giraud-Carrier, Christophe, Ricardo Vilalta, and Pavel Brazdil (2004). "Introduction to the Special Issue on Meta-Learning". In: *Journal of Machine Learning* 54.3, pp. 187–193.

Gittins, John (1979). "Bandit processes and dynamic allocation indices". In: *Series B (Methodological)* abs/1607.00215.

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feed-forward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, PMLR*.

Goldberg, David E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0201157675.

Grabczewski, Krzysztof and Norbert Jankowski (2007). "Versatile and Efficient Meta-Learning Architecture: Knowledge Representation and Management in Computational Intelligence". In: *IEEE Symposium on Computational Intelligence and Data Mining*, pp. 51–58.

Graner, Nicolas, Sunil Sharma, Derek H. Sleeman, and et al. (1994). "The Machine Learning Toolbox Consultant". In: *International Journal on Artificial Intelligence Tools* 2.3, pp. 307–328.

Grochowski, M, W Duch, and et al. (2008). "Projection Pursuit Constructive Neural Networks-Based on Quality of Projected Clusters". In: *Lecture Notes in Computer Science*, pp. 754–762.

Guerra, Silvio B., Ricardo B. Prudencio, and Teresa Ludermir (2008). "Predicting the Performance of Learning Algorithms Using Support Vector Machines as Meta-regressors". In: *Proceedings of the 18th international conference on Artificial Neural Networks (ICANN)*. Berlin, Heidelberg: Springer-Verlag, pp. 523–532.

Gupta, Abhishek, Russell Mendonca, Yuxuan Liu, Pieter Abbeel, and Sergey Levine (2018). "Meta-Reinforcement Learning of Structured Exploration Strategies". In:

Hassannejad, Hamid, Guido Matrella, Paolo Ciampolini, Ilaria De Munari, and et al. (2016). "Food Image Recognition Using Very Deep Convolutional Networks". In: *Proceedings of the 2Nd International Workshop on Multimedia Assisted Dietary Management (MADiMa)*. New York, NY, USA: ACM, pp. 41–49.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

Hem, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2014). "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *Computing Research Repository (CoRR)* abs/1406.4729.

Hinton, Geff, N. Srivastava, and Swersky K. (2014). *Overview of mini-batch gradient descent lecture of Neural Networks for Machine Learning course*. URL: `http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture\_slides\_lec6.pdf`.

Hochreiter, Sepp and Jurgen Schmidhuber (1997). "Long short-term memory". In: *Neural Computation*, pp. 1735–1780.

Huang, Gao, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger (2016). "Deep Networks with Stochastic Depth". In: *Computing Research Repository (CoRR* abs/1603.09382.

Hutter, F., H. Hoos, and K Leyton-Brown (2011). "Sequential model-based optimization for general algorithm configuration". In: *Proceedings of the Fifth International Conference on Learning and Intelligent Optimization (LION)*.

Hutter, Frank, Lars Kotthoff, and Joaquin Vanschoren, eds. (2018). *Automated Machine Learning: Methods, Systems, Challenges*. Springer.

Hyndman, Rob J. and Yeasmin Kh (2008). "Automatic time series forecasting: The forecast package for R". In: *Journal of Statistical Software*.

Ioffe, Sergey and Christian Szegedy (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International Conference of Machine Learning (ICML)*.

Jankowski, Norbert and Krzysztof Grabczewski (2011). "Universal Meta-Learning Architecture and Algorithms". In: *Meta-Learning in Computational Intelligence*. Ed. by Norbert Jankowski, Wlodzislaw Duch, and Krzysztof Grabczewski. Vol. 358. Studies in Computational Intelligence. Springer, pp. 1–76.

Kadlec, Petr and Bogdan Gabrys (2009). "Architecture for development of adaptive on-line prediction models". In: *Memetic Computing* 1.4, pp. 241–269.

Kalousis A., Hilario M. (2001). "Model selection via meta-learning: a comparative study". In: *Internation Journal on Artificial Intelligence Tools* 10.4, pp. 525–554.

King, Ross (1995). *Statlog Project Data Set*. URL: `http://mlr.cs.umass.edu/ml/datasets/Statlog+Project`.

King, Ross, C Feng, and Alistair Sutherland (1995). "StatLog: Comparison of Classification Algorithms on Large Real-World Problems". In: *Journal of Applied Artificial Intelligence* 9.3, pp. 289–334.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations (ICLR)*.

Klinkenberg, Ralf (2005). "Meta-Learning, Model Selection, and Example Selection in Machine Learning Domains with Concept Drift". In: *Annual workshop of the special interest group on machine learning, knowledge discovery, and data mining of the German Computer Science Society (GI)*. Saarbrucken, Germany.

Komer, Brent, James Bergstra, and Chris Eliasmith (2014). "Hyperopt-sklearn: automatic hyper-parameter configuration for scikit-learn". In: *ICML workshop on AutoML*.

Kopf, Christian and Ioannis Iglezakis (2002). "Combination of Task Description Strategies and Case Base Properties for Meta-learning". In: *Proceedings of the 2nd International Workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-learning (IDDM)*. Helsinki, Finland, pp. 65–76.

Kordik, Pavel and Jan Cerny (2014). "Building predictive models in two stages with meta-learning templates optimized by genetic programming". In: *2014 IEEE Symposium on Computational Intelligence in Ensemble Learning, CIEL*, pp. 27–34.

Kotthoff, Lars, Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown (2017). "Auto-WEKA 2.0: Automatic Model Selection and Hyper-parameter Optimization in WEKA". In: *Journal of Machine Learning Reseasrch* 18.1, pp. 826–830.

Kovarik, Oleg and Richard Malek (2012). *Meta-learning and Meta-optimization*. Tech. rep. Prague, Czech Republic: Technical Report, Czech Technical University.

Krizhevsky, Alex, Vinod Nair, and Geoffrey Hinton. "CIFAR-10 and CIFAR-100". In: *Canadian Institute for Advanced Research*.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS)*. Vol. 1. USA: Curran Associates Inc., pp. 1097–1105.

Kuhn, Daniel, Philipp Probst, Thomas Janek, and Bernd Bischl (2018). "Automatic Exploration of Machine Learning Experiments on OpenML". In: *Computing Research Repository (CoRR)* 1806.10961.

Le, Ya and Xuan Yang (2015). "Tiny ImageNet Visual Recognition Challenge". In: *Stanford CS 231N*.

LeCun, Yann, Corinna Cortes, and Christopher J. C. Burges (1999). "The MNIST Dataset Of Handwritten Digits". In:

Lemke, Christiane and Bogdan Gabrys (2010a). "Meta-learning for time series forecasting and forecast combination". In: *Journal of Neurocomputing* 73.10-12, pp. 2006–2016.

Lemke, Christiane and Bogdan Gabrys (2010b). "Meta-learning for time series forecasting in the NN GC1 competition". In: *Fuzzy Systems (FUZZ)*, pp. 1–5.

Lemke, Christiane, Marcin Budka, and Bogdan Gabrys (2015). "Metalearning: a survey of trends and technologies". In: *Artificial Intelligence Review* 44 (1).

Li, S. Z. (1995). "Markov Random Field Modeling in Computer Vision". In:

Lin, Kevin, Huei-Fang Yang, and Chu-Song Chen (2015). "Flower classification with few training examples via recalling visual patterns from deep CNN". In: CVGIP, pp. 41–49.

Lin, Tsung-Yi, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, and et al. (2014). "Microsoft COCO: Common Objects in Context". In: *Computing Research Repository (CoRR)* abs/1405.0312.

Lindner, Guido and Rudi Studer (1999). "AST: Support for Algorithm Selection with a CBR Approach". In: *Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*. London, UK, UK: Springer-Verlag, pp. 418–423.

Liu, Chenxi, Barret Zoph, Maxim Neumann, Jonathon Shlens, and et al. (2018). "Progressive Neural Architecture Search". In: *Computing Research Repository (CoRR)* abs/1712.00559.

Loshchilov, I. and F. Hutter (2016). "Practical Network Blocks Design with Q-Learning". In: *In: International Conference on Learning Representations (ICLR) Workshop track*.

Martin S., Manuel, Marcin Budka, and Bogdan Gabrys (2016a). "Adapting Multi-component Predictive Systems using Hybrid Adaptation Strategies with Auto-WEKA in Process Industry". In: *Proceedings of the Workshop on Automatic Machine Learning*. Vol. 64. New York, New York, USA: PMLR, pp. 48–57.

Martin S., Manuel, Marcin Budka, and Bogdan Gabrys (2016b). "Automatic composition and optimisation of multi-component predictive systems". In: *Computing Research Repository (CoRR)* abs/1612.08789.

Martin S., Manuel, Marcin Budka, and Bogdan Gabrys (2016c). "Towards Automatic Composition of Multi-component Predictive Systems". In: *HAIS*. Vol. 9648. Lecture Notes in Computer Science. Springer, pp. 27–39.

Maszczyk, Tomasz, Marek Grochowski, and Wlodzislaw Duch, eds. (2010). *Advances in Machine Learning II*. Vol. 263. Studies in Computational Intelligence. Springer.

Menahem, Eitan, Lior Rokach, and Yuval Elovici (2011). "Combining One-Class Classifiers via Meta-Learning". In: *Computing Research Repository (CoRR)* abs/1112.5246.

Mierswa, Ingo, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler (2006). "YALE: Rapid Prototyping for Complex Data Mining Tasks". In: *In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*. Ed. by Lyle Ungar, Mark Craven, Dimitrios Gunopulos, and Tina Eliassi-Rad. New York, NY, USA: ACM, pp. 935–940.

Miikkulainen, Risto, Jason Zhi Liang, Elliot Meyerson, Aditya Rawal, and et al. (2017). "Evolving Deep Neural Networks". In: *Computing Research Repository (CoRR)* abs/1703.00548.

Minar, Matiur R. and Jibon Naher (2018). "Recent Advances in Deep Learning: An Overview". In: *Computing Research Repository (CoRR)* abs/1807.08169.

Morik, Katharina and Martin Scholz (2003). "The MiningMart Approach to Knowledge Discovery in Databases". In: *In Ning Zhong and Jiming Liu, editors, Intelligent Technologies for Information Analysis*. Springer, pp. 47–65.

Movielens (1998). *MovieLens Data Sets*. URL: `http://grouplens.org/node/12`.

Murtagh, Fionn and Pierre Legendre (2014). "Ward's hierarchical agglomerative clustering method: which algorithms implement Ward's criterion?" In: *Journal of Classification* 31.3, pp. 274–295.

Nagabandi, Anusha, Chelsea Finn, and Sergey Levine (2018). "Deep Online Learning via Meta-Learning: Continual Adaptation for Model-Based RL". In: *Computing Research Repository (CoRR)* abs/1812.07671.

Nair, Vinod and Geoffrey E. Hinton (2010). "Rectified linear units improve restricted Boltzmann machines". In: *International Conference of Machine Learning (ICML)*.

Nilsback, Maria-Elena and Andrew Zisserman (2008). "Automated Flower Classification over a Large Number of Classes". In: *2008 Sixth Indian Conference on Computer Vision, Graphics Image Processing*, pp. 722–729.

Olson, R.S., N. Bartley, R.J. Urbanowicz, and J.H. Moore (2018). "Evaluation of a tree-based pipeline optimization tool for automating data science". In: *In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* abs/1808.01974, pp. 485–492.

Osband, Ian and Benjamin Van Roy (2016). "Why is posterior sampling better than optimism for reinforcement learning". In: *Computing Research Repository (CoRR)* abs/1607.00215.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). "Scikit-learn: Machine Learning in Python ". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Peng, Yonghong, Peter A. Flach, Carlos Soares, and Pavel Brazdil (2002). "Improved Dataset Characterisation for Meta-learning". In: *Proceedings of the 5th International Conference on Discovery Science (DS)*. London, UK: Springer-Verlag, pp. 141–152.

Perrone, V., R. Jenatton, M. Seeger, and C. Archambeau (2017). "Multiple adaptive Bayesian linear regression for scalable Bayesian optimization with warm start". In: *Computing Research Repository (CoRR)* abs/1712.02902.

Petris, Giovanni and Sonia Petrone (2011). "State Space Models in R". In: *Journal of Statistical Software* 41 (4).

Pfahringer, Bernhard, Hilan Bensusan, and Christophe Giraud-Carrier (2000). *Meta-learning by landmarking various learning algorithms*. Tech. rep. Bristol, UK: University of Bristol.

Pham, Hieu, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean (2018). "Efficient Neural Architecture Search via Parameter Sharing". In: *Computing Research Repository (CoRR)* abs/1802.03268.

Prudencio, Ricardo, Marcilio deSouto, and Teresa Ludermir (2011). *Selecting Machine Learning Algorithms Using the Ranking Meta-Learning Approach*. Meta-Learning in Computational Intelligence, Studies in Computational Intelligence. Springer Berlin Heidelberg.

Prudencio, Ricardo B. C. and Teresa B. Ludermir (2004). "Meta-learning approaches to selecting time series models". In: *Journal of Neurocomputing* 61, pp. 121–137.

Prudencio, Ricardo B. C. and Teresa B. Ludermir (2008). "Active Meta-Learning with Uncertainty Sampling and Outlier Detection". In: *IEEE International Joint Conference on Neural Networks*, pp. 346–351.

Quinlan, John R. (1992). "Learning With Continuous Classes". In: *AI'92 (Adams and Sterling, Eds)*. Singapore: World Scientific, pp. 343–348.

Quinlan, John R. (1998). *C5.0: An Informal Tutorial*. URL: `http://www.rulequest.com/see5-unix.html`.

R, Core Development Team (2010). *R: A Language and Environment for Statistical Computing*. Vienna, Austria. URL: `http://www.r-project.org/`.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. Vienna, Austria. URL: `http://www.R-project.org/`.

Rajpurkar, Pranav, Jeremy Irvin, Kaylie Zhu, Brandon Yang, and et al. (2017). "CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning". In: *Computing Research Repository (CoRR)* abs/1711.05225.

Ravi, Sachin and Hugo Larochelle (2017). "Optimization as a model for few-shot learning". In: *International Conference on Learning Representations (ICLR)*.

Real, E., S. Moore, A. Selle, S. Saxena, Y.L. Suematsu, Q.V. Le, and A. Kurakin (2017). "Large-scale evolution of image classifier". In: *In International Conference on Machine Learning*.

Reif, Matthias (2012). "A Comprehensive Dataset for Evaluating Approaches of various Meta-Learning Tasks". In: *First International Conference on Pattern Recognition Applications and Methods*. Vilamura, Algarce, Portugal.

Reif, Matthias, Faisal Shafait, and Andreas Dengel (2012a). "Dataset Generation for Meta-Learning". In: *KI-2012: Poster and Demo Track*. Saarbrucken, pp. 69–73.

Reif, Matthias, Faisal Shafait, and Andreas Dengel (2012b). "Meta2-Features: Providing Meta-Learners More Information". In: *KI-2012: Poster and Demo Track*. Saarbrucken, pp. 74–77.

Rendell, Larry and Howard Cho (1990). "Empirical Learning as a Function of Concept Character". In: *Journal of Machine Learning* 5.3, pp. 267–298. ISSN: 0885-6125.

Rendell, Larry, Raj Sheshu, and David Tcheng (1987). "Layered concept-learning and dynamically variable bias management". In: *Proceedings of the 10th international joint conference on Artificial intelligence (IJCAI)*. Vol. 1. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 308–314.

Riedel, Silvia and Bogdan Gabrys (2007). "Dynamic Pooling for the Combination of Forecasts Generated Using Multi Level Learning". In: *IJCNN*, pp. 454–459.

Riedel, Silvia and Bogdan Gabrys (2009). "Pooling for Combination of Multilevel Forecasts". In: *IEEE Transactions on Knowledge and Data Engineering* 21.12, pp. 1753–1766.

Rijn, J. N. van, B. Bischl, L. Torgo, B. Gao, V. Umaashankar, and et al. (2013). "OpenML: a collaborative science platform". In: *in Proceedings of ECML/PKDD13*.

Rijn, Jan N. van, Salisu Abdulrahman, Pavel Brazdil, and Joaquin Vanschoren (2015). "Fast Algorithm Selection Using Learning Curves". In: *In International Symposium on Intelligent Data Analysis*.

Rivolli, A., L. Garcia, C. Soares, J. Vanschoren, and A. de Carvalho (2018). "Towards reproducible empirical research in meta-learning". In: *Computing Research Repository (CoRR)* abs/1808.10406.

Rossi, Andre Luis Debiaso, Andre Carlos Ponce de Leon Ferreira de Carvalho, and Carlos Soares (2012). "Meta-Learning for Periodic Algorithm Selection in Time-Changing Data". In: *Brazilian Symposium on Neural Networks*, pp. 7–12.

Rossi, Andre Luis Debiaso, Andre Carlos Ponce de Leon Ferreira de Carvalho, and et al. (2014). "MetaStream: A meta-learning based method for periodic algorithm selection in time-changing data". In: *Journal of Neurocomputing* 127, 5264.

Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, and et al. (2014). *ImageNet Large Scale Visual Recognition Challenge*. URL: `http://arxiv.org/abs/1409.0575`.

Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, and et al. (2015). "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal Computer Vision* 115.3, pp. 211–252. ISSN: 0920-5691.

Shah, Chandra (1997). "Model selection in univariate time series forecasting using discriminant analysis". In: *International Journal of Forecasting* 13.4, pp. 489–500.

Shin, Hoo-Chang, Kirk Roberts, Le Lu, Dina Demner-Fushman, Jianhua Yao, and Ronald M. Summers (2016). "Learning to Read Chest X-Rays: Recurrent Neural Cascade Model for Automated Image Annotation". In: *Computing Research Repository (CoRR)* abs/1603.08486.

Sikora, Riyaz T. (2008). "Meta-learning optimal parameter values in non-stationary environments". In: *Journal of Knowledge-Based Systems* 21.8, pp. 800–806.

Simonyan, Karen and Andrew Zisserman (2014). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *Computing Research Repository (CoRR)* abs/1409.1556.

Smith-Miles, Kate (2009). "Cross-disciplinary perspectives on meta-learning for algorithm selection". In: *ACM Computing Surveys* 41.1, pp. 1–25.

Soares, Carlos (2009). "UCI++: Improved Support for Algorithm Selection Using Datasetoids". In: *Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD)*. Berlin, Heidelberg: Springer-Verlag, pp. 499–506.

Soares, Carlos and Pavel B. Brazdil (2006). "Selecting Parameters of SVM Using Meta-learning and Kernel Matrix-based Meta-features". In: *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC)*. New York, NY, USA: ACM, pp. 564–568.

Soares, Carlos, Johann Petrak, and Pavel Brazdil (2001). "Sampling-Based Relative Landmarks: Systematically Test-Driving Algorithms Before Choosing". In: *Proceedings of the 10th Portuguese Conference on Artificial Intelligence on Progress in Artificial Intelligence, Knowledge Extraction, Multi-agent Systems, Logic Programming and Constraint Solving*. London, UK: Springer-Verlag, pp. 88–95.

Soares, Carlos, Petr Kuba, and Peter Flach (2004). "A meta-learning method to select the kernel width in support vector regression". In: *Mach. Learning*, pp. 195–209.

Soares, Rodrigo G., Teresa B. Ludermir, and Francisco A. Carvalho (2009). "An Analysis of Meta-learning Techniques for Ranking Clustering Algorithms Applied to Artificial Data". In: *Proceedings of the 19th International Conference on Artificial Neural Networks (ICANN)*. Berlin, Heidelberg: Springer-Verlag, pp. 131–140.

Sohn, So Y. (1999). "Meta analysis of classification algorithms for pattern recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.11, pp. 1137–1144.

Stanford, Stacy and Roberto Iriondo (2018). *The Best Public Datasets for Machine Learning and Data Science*. URL: `https://towardsai.net/datasets`.

Sutskever, Ilya, James Martens, George Dahl, and Geoffrey E. Hinton (2013). "Practical Network Blocks Design with Q-Learning". In: *International Conference of Machine Learning (ICML)*.

Sutton, Richard S. and Andrew G. Barto (2015). "Introduction to Reinforcement Learning". In: *MIT Press* (2).

Sutton, Richard S., David McAllester, Satinder Singh, and Yishay Mansour (1999). "Policy gradient methods for reinforcement learning with function approximation". In: *NIPS*.

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, and et al. (2015). "Going Deeper with Convolutions". In: *Computer Vision and Pattern Recognition (CVPR)*. Vol. abs/1409.4842.

Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, and et al. (2016). "Rethinking the Inception Architecture for Computer Vision". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 2818–2826.

Szegedy, Christian, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi (2017). "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. Pp. 4278–4284.

Tan, Chuanqi, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu (2018). "A Survey on Deep Transfer Learning". In: *Computing Research Repository (CoRR)* abs/1808.01974.

Thornton, Chris, Frank Hutter, Holger Hoos, and Kevin Leyton-Brown (2013). "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classi
cation Algorithms". In: *ACM International Conference on Knowledge Discovery and Data Mining*, pp. 847–855.

Todorovski, Ljupco, Hendrik Blockeel, and Saso Dzeroski (2002). "Ranking with Predictive Clustering Trees". In: *Proceedings of the 13th European Conference on Machine Learning (ECML)*. London, UK: Springer-Verlag, pp. 444–455.

Todorovski, Ljupvco and Savso Dvzeroski (2003). "Combining Classifiers with Meta Decision Trees". In: *Journal of Machine Learning* 50.3, pp. 223–249.

Tusell, Fernando (2011). "Kalman Filtering in R". In: *Journal of Statistical Software* 39 (2).

Utgoff, Paul Everett (1984). "Shift of bias for inductive concept learning". PhD thesis. New Brunswick, NJ, USA: Rutgers University.

Vanschoren, Joaquin (2011). "Meta-Learning Architectures: Collecting, Organizing and Exploiting Meta-Knowledge". In: *Meta-Learning in Computational Intelligence*. Ed. by Norbert Jankowski, WlMeta-learningodzislaw Duch, and Krzysztof Grabczewski. Vol. 358. Studies in Computational Intelligence. Springer, pp. 117–155.

Vanschoren, Joaquin, Jan N. Rijn, Bernd Bischl, and Luis Torgo (2014). "OpenML: Networked Science in Machine Learning". In: *SIGKDD Explor. Newsl.* 15.2, pp. 49–60.

Vilalta, Ricardo and Youssef Drissi (2002a). "A perspective view and survey of meta-learning". In: *Artificial Intelligence Review* 18.2, pp. 77–95.

Vilalta, Ricardo and Youssef Drissi (2002b). "A Perspective View and Survey of Meta-learning". In: *Artif. Intell. Rev.* 18.2, pp. 77–95. ISSN: 0269-2821.

Vilalta, Ricardo, Christophe Giraud-carrier, Pavel Brazdil, and Carlos Soares (2004). "Using Meta-Learning to Support Data Mining". In:

Vilalta, Ricardo, Christophe Giraud-Carrier, and Pavel Brazdil (2010). *Meta-Learning - Concepts and Techniques*. Data Mining and Knowledge Discovery Handbook. US: Springer.

Wang, Dong and Thomas Fang Zheng (2015). "Transfer learning for speech and language processing". In: *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pp. 1225–1237.

Wang, Jane X., Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, and et al. (2017a). "Learning to Reinforcement Learn". In: *Computing Research Repository (CoRR)* abs/1611.05763.

Wang, Xiaosong, Yifan Peng, Le Lu, Zhiyong Lu, and et al. (2017b). "ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases". In: *Computing Research Repository (CoRR)* abs/1705.02315.

Wang, Xiaozhe, Kate Smith-Miles, and Rob Hyndman (2009). "Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series". In: *Journal of Neurocomputing* 72.10-12, pp. 2581–2594.

Warden, Pete (2011). *Data Source Handbook - A Guide to Public Data*. OŔeilly Media.

Widmer, Gerhard (1997). "Tracking Context Changes through Meta-Learning". In: *Journal of Machine Learning* 27.3, pp. 259–286.

Williams, Ronald J. (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine Learning* (9), pp. 41–49.

Wistuba, M., N. Schilling, and L. Schmidt-Thieme (2015). "Learning hyper-parameter optimization initializations". In: *In: IEEE International Conference on Data Science and Advanced Analytics (DSAA)*.

Wolpert, David (2001). "The supervised learning no-free-lunch Theorems". In: *Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications*, pp. 25–42.

Xiao, Han, Kashif Rasul, and Roland Vollgraf (2017). "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". In: *Computing Research Repository (CoRR)* abs/1708.07747.

Xu, Tianbing, Qiang Liu, Liang Zhao, and Jian Peng (2018). "Learning to Explore with Meta-Policy Gradient". In: *Computing Research Repository (CoRR)* abs/1803.05044.

Yao, Quanming, Mengshuo Wang, Hugo Escalante, Isabelle Guyon, Yi-Qi Hu Hu, and et al. (2019). "Taking Human out of Learning Applications: A Survey on Automated Machine Learning". In: *Computing Research Repository (CoRR)* abs/1810.13306.

Yosinski, Jason, Jeff Clune, Yoshua Bengio, and Hod Lipson (2014). "How Transferable Are Features in Deep Neural Networks?" In: *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS)*. Vol. 2. Cambridge, MA, USA: MIT Press, pp. 3320–3328.

Zeng, X. and G. Luo (2017). "Progressive sampling-based Bayesian optimization for efficient and automatic machine learning model selection". In: *Health Information Science and Systems* 5 (1).

Zhang, Zewang, Zheng Sun, Jiaqi Liu, Jingwen Chen, and et al. (2016). "An Experimental Comparison of Deep Neural Networks for End-to-end Speech Recognition". In: *Computing Research Repository (CoRR)* abs/1611.07174.

Zhong, Zhao, Junjie Yan, and Cheng-Lin Liu (2017). "Practical Network Blocks Design with Q-Learning". In: *Computing Research Repository (CoRR)* abs/1708.05552.

Zhong, Zhao, Junjie Yan, Wei Wu, Jing Shao, Liu, and Cheng-Lin (2018). "Practical Network Blocks Design with Q-Learning". In: *Computing Research Repository (CoRR)* abs/1708.05552.

Zliobaite, Indre (2010). "Learning under Concept Drift: An Overview". In: *Computing Research Repository (CoRR)* abs/1010.4784.

Zöller, Marc-André and Marco F. Huber (2019). "Survey on Automated Machine Learning". In: *Computing Research Repository (CoRR)* abs/1904.12054.

Zoph, Barret and Quoc V. Le (2016). "Neural Architecture Search with Reinforcement Learning". In: *Computing Research Repository (CoRR)* abs/1611.01578.

Zoph, Barret and Quoc V. Le (2017). "Neural Architecture Search with Reinforcement Learning". In: *International Conference on Learning Representations (ICLR)*.

Zoph, Barret, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le (2018). "Learning transferable architectures for salable image recognition". In: *Computer Vision and Pattern Recognition (CVPR)*.