A STUDY ON THE ROLE OF HYPERPARAMETER OPTIMIZATION (HPO) IN IMAGE CLASSIFICATION PROBLEMS USING AUTOML MODELS

Thesis

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

AMALA MARY VINCENT

(Reg No: 207050MA002)

DEPARTMENT OF MATHEMATICAL & COMPUTATIONAL SCIENCES

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

SURATHKAL, MANGALORE - 575025

JUNE, 2024

A STUDY ON THE ROLE OF HYPERPARAMETER OPTIMIZATION (HPO) IN IMAGE CLASSIFICATION PROBLEMS USING AUTOML MODELS

Thesis

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

AMALA MARY VINCENT

(Reg No: 207050MA002)

Under the guidance of

Dr. Jidesh P.

DEPARTMENT OF MATHEMATICAL & COMPUTATIONAL SCIENCES

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

SURATHKAL, MANGALORE - 575025

JUNE, 2024

DECLARATION

I hereby *declare* that the Research Thesis entitled **A STUDY ON THE ROLE OF HYPERPARAMETER OPTIMIZATION (HPO) IN IMAGE CLASSIFICATION PROBLEMS USING AUTOML MODELS** which is being submitted to the **National Institute of Technology Karnataka, Surathkal** in partial fulfilment of the requirements for the award of the Degree of **Doctor of Philosophy** in **Machine Learning** is a *bonafide report of the research work carried out by me*. The material contained in this Research Thesis has not been submitted to any University or Institution for the award of any degree.

Amala Mary Vincent

Reg. No: 207050MA002

Department of Mathematical and Computational Sciences

Place: NITK, Surathkal

Date: 19th June 2024

CERTIFICATE

This is to *certify* that the Research Thesis entitled **A STUDY ON THE ROLE OF HYPERPARAMETER OPTIMIZATION (HPO) IN IMAGE CLASSIFICATION PROBLEMS USING AUTOML MODELS** submitted by **Amala Mary Vincent**, (Reg. No: 207050MA002) as the record of the research work carried out by her, is *accepted as the Research Thesis submission* in partial fulfilment of the requirements for the award of degree of **Doctor of Philosophy**.

Dr. Jidesh P.
Research Supervisor
Associate Professor
Department of Mathematical and Computational Sciences
National Institute of Technology Karnataka

for Chairman - DRPC
Department of Mathematical and Computational Sciences
National Institute of Technology Karnataka

ACKNOWLEDGMENTS

I express my heartfelt gratitude to all those who have been instrumental in guiding and supporting me on my research journey at NITK. Their contributions, both direct and indirect, have played a pivotal role in the successful completion of my research.

I extend my deepest gratitude to my research supervisor, **Dr. Jidesh P.**, Department of MACS, for the continued guidance and support throughout my research journey. Sir, I admire you for your impeccable work ethics and is grateful for your valuable research insights. Your constant encouragement has been invaluable in shaping my academic and research pursuits. I sincerely thank you for taking the time to discuss, share insights, and make corrections, all of which have significantly contributed to enhancing my research work. I am truly fortunate to have you as a mentor, and I am deeply thankful for your invaluable contributions to my growth and success.

I am extremely grateful to the Research Progress Assessment Committee members - **Dr. Jaidhar C. D.**, Department of Information Technology and **Dr. Pushparaj Shetty**, Department of MACS, for their insightful suggestions. I also express my sincere thanks to **Dr. Sam Johnson**, Head of the Department of MACS for all the assistance and encouragement. I also extend my sincere gratitude to **Dr. R. Madhusudhan** and **Dr. Shyam S. Kamath** who were former Head of the Department of MACS for their relentless support. I convey my profound gratitude to **Dr. B. R. Shankar** and **Dr. Santhosh George** for their kind support and encouragement. I will always be grateful to all faculty members of the Department of MACS, for their support and motivation, throughout my tenure as a research scholar in the Department. I express my deepest appreciation to **Mrs. Gayathri, Mrs. Bhavani, Mrs. Sushma** and **Mr. Naveen**, non-teaching staff of the Department of MACS. I sincerely acknowledge the help and assistance rendered by all fellow research scholars of Department of MACS.

I thank the Ministry of Education, Government of India for providing financial support as Fellowship to carry out the research at NITK. I am sincerely grateful to all the

anonymous reviewers and editors-in-chief of Scientific Reports and Applied Soft Computing. The constructive criticism and comments received from them played a crucial role in enhancing the quality of my work.

Words fall short to tell you how much grateful I am for your support, **Partha**. Your guidance and expertise have been instrumental in my journey. Your willingness to share knowledge and offer help has not only enriched my understanding but has also been a cornerstone in overcoming challenges. I will always appreciate the assistance you provided me. **Dinu** and **Tom**, I owe a great deal of my success to your influence and motivation. Thank you for being my pillars of support. And **Ashi**, I cannot thank you enough for the consistent moral and emotional support you offered, particularly during challenging times.

Architha, I thank you for your kindness to lend a helping hand anytime needed. I am truly fortunate to have had you by my side as a dependable ally. I also acknowledge the help rendered by **Smitha** and **Abhishek**. I extend my thanks for all the moral support offered by my dearest friends: **Swaroop, Hajira, Athira, Alka, Rony, Binoy, Arya** and **Athul**. I also cherish the encouragement I received from my friends: **Niveditha, Sadhik, Naseeha, Ajay** and other friends from AMD, during my master's to pursue Ph.D.

I am deeply indebted to my grandparents, my brother, **Anupam**, my sister, **Anjitha**, my sister-in-law, **Libi**, my cousins and all family members for consistently supporting me. Thank you for your love, guidance, and encouragement throughout my journey. **Mummy, Papa**, I thank you for the principles of hard work, perseverance, and resilience that you instilled in me from my early years. Your faith in the transformative power of education has shaped my ambitions. Thank you for believing in me. I am eternally grateful to you and I hope to make you proud in all that I do.

ABSTRACT

Hyperparameter optimization (HPO) has a profound impact on the performance of machine learning models. This work investigates various HPO techniques and offers valuable insights into their effectiveness, particularly in the context of image classification with the aid of AutoML models. The study places a special focus on Bayesian optimization and introduces the innovative application of genetic algorithms, differential evolution, and covariance matrix adaptation - evolutionary strategy for acquisition function optimization. Comparative analysis reveals that these evolutionary variants significantly enhance the performance of standard Bayesian optimization, with genetic algorithms emerging as less effective for acquisition function optimization.

In real-time scenarios, where deep learning models often involve a multitude of hyperparameters, finding the optimal configuration becomes even more challenging. To illustrate the practical implications of HPO, the thesis presents two real-time case studies. The first case study centers on the analysis of land use and land cover changes in the Ernakulam district of Kerala using sentinel-2 images from 2019 and 2023. Employing machine learning models aided by evolutionary-based hyperparameter optimization, the study successfully tracks spatial and temporal changes in land use patterns. Furthermore, a post-classification change detection was carried out to elucidate the scale and pace of urban growth following the COVID-19 pandemic.

The second study addresses flood vulnerability prediction in Kerala, India, leveraging machine learning models and AutoML systems. A three-dimensional convolutional neural network (CNN), coupled with Bayesian optimization and evolutionary algorithms like differential evolution and covariance matrix adaptation, leads to enhanced accuracy, precision, recall, AUC, and kappa scores compared to AutoML models.

This thesis also ventures into the realm of meta-learning, a potential approach for tackling the data scarcity inherent in real-time image classification applications. Focusing on metric-based meta-learning models, it explores the integration of hyperparameter optimization techniques to enhance the performance of these models. The innovative bilevel optimization framework, combining meta-learning and hyperparameter optimization, is introduced, with Bayesian optimization and its recent variants employed

to fine-tune hyperparameters. Computational experiments conducted on Omniglot and ImageNet datasets provide insights into the impact of different hyperparameter optimization algorithms, with meta-testing accuracy serving as the basis for conclusions.

In summary, the thesis focuses on the critical role of hyperparameter optimization in improving the machine learning model performance, emphasizing its practical applications through real-time case studies. Additionally, it explores the synergy of meta-learning and hyperparameter optimization, offering a comprehensive view of cutting-edge techniques in the field.

Keywords: Hyperparameter Optimization; AutoML; Machine Learning; Bayesian optimization; Meta-learning

Contents

List of Figures

List of Tables

List of Abbreviations

ACO	Ant Colony Optimization
ANIL	Almost No Inner Loop
ANN	Artificial Neural Network
AutoML	Automated Machine Learning
BDC	Brownian Distance Covariance
BO	Bayesian Optimization
BOCA	Bayesian optimization with Continuous Approximations
BOHB	Bayesian Optimization with Hyperband
CMA-ES	Covariance Matrix Adaptation Evolutionary Strategy
CNN	Convolutional Neural Network
CSN	Conditionally Shifted Neurons
DE	Differential Evolution
DEHB	Differential Evolution Hyperband
EA	Evolutionary Algorithm
EMD	Earth Mover's Distance
ES	Evolutionary Strategies
FSL	Few-Shot Learning
FSM	Flood Susceptibility Mapping
GA	Genetic Algorithm
GEE	Google Earth Engine
GIS	Geographical Information System
GP	Gaussian Process
HPO	Hyperparameter Optimization
LULC	Land Use Land Cover
MAML	Model Agnostic Meta-Learning
MFGP-UCB	Multi-Fidelity Gaussian Process Upper Confidence Bound
ML	Machine Learning
NAS	Neural Architecture Search
NDVI	Normalized Difference Vegetation Index
PSO	Particle Swarm Optimization
RF	Random Forest
RPMN	Relative Position and Map Network
RPN	Relative Position Network
RMN	Relative Map Network
SAR	Synthetic Aperture Radar
SH	Successive Halving
SI	Swarm Intelligence
SMAC	Sequential Model-based Algorithm Configuration

SMBO	Sequential Model-Based Optimization
SNAIL	Simple Neural Attentive Meta-Learner
SPI	Stream Power Index
SVM	Support Vector Machine
TPE	Tree Parzen Estimator
TPOT	Tree-based Pipeline Optimization Tool
TWI	Topographic Wetness Index
UCB	Upper Confidence Bound
XGBoost	Extreme Gradient Boost

List of Symbols

λ	—	Hyperparameter configuration
μ	—	Mean
σ	—	Standard deviation
$\mathbb{E}$	—	Expected value
F	—	Objective function
G	—	Gaussian process model
E	—	Acquisition function
T	—	Maximal number of iterations
n	—	Initial number of solutions in the population
p_c	—	Crossover probability
p_m	—	Mutation probability
C	—	Covariance matrix

Chapter 1

INTRODUCTION

1.1 AN OVERVIEW OF HYPERPARAMETER OPTIMIZATION

The integration of Machine Learning (ML) strategies has revolutionized automated models across numerous application domains. The spectrum of applications ranges over various domains, from atmospheric analysis to medical diagnosis. These applications are design-sensitive, meaning that the model's performance depends on factors like the chosen machine learning algorithm, training procedures, and regularization methods. Most importantly, the selection of the optimal set of hyperparameters plays a critical role in influencing the effectiveness of these models.

Every machine learning model has two types of parameters, a set that is trained by the model and another that controls the learning process. The former set of parameters is determined using the training dataset and is called the model parameters. The latter are values that can be tuned and adjusted by the user before running the model. They have a major role in shaping the behaviour of the model and are called hyperparameters. The weights of a neural network are model parameters that are derived and fitted by training, whereas the learning rate of a neural network, the regularization parameter, and the kernel parameter are all examples of hyperparameters. Different machine learning algorithms require various sets of hyperparameters. Other than a few simple models such as least square regression, most machine learning models have hyperparameters.

Different hyperparameter configurations are required for different datasets. It is important to find a hyperparameter setting that performs optimally for a given algorithm trained over a particular dataset. This is done by tuning the hyperparameters and the

technique is known as Hyperparameter Optimization (HPO) (Feurer and Hutter, 2018). With the optimal set of hyperparameters, the algorithm learns the model parameters from the data.

1.2 HYPERPARAMETER OPTIMIZATION

Every machine learning algorithm aims to identify a function that optimizes the loss. Let M be the given machine learning algorithm with $\{\lambda_1, \lambda_2, ...\lambda_n\}$ the parameters to be tuned. Each hyperparameter λ_i can have a value within the interval $[a_i, b_i]$ in a hyperparameter configuration space $\Lambda = [a_1, b_1] \times ... \times [a_n, b_n]$. Here F is a function that determines the performance or loss value. Further, F maps each possible configuration $\lambda \in \Lambda$ to a numerical value $F : \Lambda \rightarrow R^*$. The objective of hyperparameter optimization is to find the best configuration λ^* that minimizes $F(\lambda)$.

$$\lambda^* = \underset{\lambda \in \Lambda}{\operatorname{argmin}} F(\lambda). \tag{1.1}$$

The objective function F is a black-box function, which means the actual function is unknown, but one can observe its output based on specific given inputs. Since the function cannot be accessed, there is no information about its derivative. The function evaluation is pretty time-consuming, implying that each iteration takes a considerable amount of time. For a small dataset, this process may take minutes, whereas for larger datasets, it can extend to hours or even days. As a result, solving the optimization problem is difficult, and obtaining the ideal configuration in a few trials necessitates a specific approach.

Some of the popular HPO methods are grid search and random search, Bayesian Optimization (BO), gradient descent, evolution-based techniques and multi-fidelity methods like hyperband and successive halving.

An extensive analysis of existing HPO techniques is carried out in order to propose a novel HPO algorithm. Considering various advantages of Bayesian optimization in solving black-box optimization problems, different ways to improve the conventional BO with the help of evolutionary algorithms were examined. Evolutionary algorithms were used taking into account their success in solving optimization problems and major advantages like ease of implementation, robustness and parallelizability. An empirical

comparison of the application of evolutionary algorithms for acquisition function optimization was performed. In particular, the use of genetic algorithm, differential evolution and covariance matrix adaptation - evolutionary strategy for acquisition function optimization were proposed. And on comparing these variants of Bayesian optimization with conventional Bayesian optimization it was observed that the use of covariance matrix adaptation - evolutionary strategy and differential evolution improves the performance of standard Bayesian optimization. Additionally, it was observed that the performance of Bayesian optimization tends to be suboptimal when a genetic algorithm is employed for the optimization of the acquisition function.

1.3 AUTOML SYSTEMS

Different ML models are appropriate for different applications. To determine the algorithm that best fits the requirements, the straightforward approach of applying and optimizing all known learning algorithms is not practical in most cases. This process of finding the best ML algorithm and creating the optimal architecture by setting the best hyperparameters is a complex and demands a significant amount of time. Here comes the importance of an Automated Machine Learning (AutoML) system that determines the optimal configurations for a particular application with the best performance within the time constraints. For a deep learning network, AutoML not only performs HPO to automatically set the optimal hyperparameters but also selects the right neural architecture for each layer. It also provides tools and approaches for enabling ML to be accessible to non-experts, increasing performance, and speeding up ML research.

1.4 APPLICATIONS OF HPO

Finding the ideal set of hyperparameters is crucial for any machine learning architecture. In real-world scenarios, it is required to build deep learning models with a large number of hyperparameters which makes it even more difficult to decide the potential optimal set of hyperparameters. Two real-world datasets are used on which machine learning models assisted with hyperparameter optimization were build. One of the case studies is predicting the flood vulnerability of a region. The second case study is deter-

mining the land use and land cover changes over the past years.

1.4.1 Land use land cover classification

The process of classifying and mapping the various land use and land cover elements in a given area using remotely sensed data, such as satellite imaging, is known as Land Use Land Cover (LULC) classification (Rwanga et al., 2017; Talukdar et al., 2020). LULC classification is essential in environmental and natural resource management, urban planning, and land use decision making. When it comes to urban planning, LULC is crucial in identifying suitable areas for building infrastructure and detecting urban expansion.

1.4.2 Flood susceptibility mapping

Floods are becoming more severe and frequent, affecting more people than any other natural hazard. This is majorly due to changes in climate, land use, infrastructure, and population demographics. According to the United Nations Office for Disaster Risk Reduction (UNDRR), out of 7348 disaster events, 3254 flood events were recorded worldwide over a period of 20 years from 2000 to 2019. A total of 1.81 billion individuals or 23% of the world's population is found to be directly vulnerable to flood (Rentschler et al., 2022). Of these, 390 million belong to the Indian population. In India, the prevalence of a subtropical monsoon-dominated climate and a wide riverine plain are the most conducive factors for the occurrence of severe floods each year in a number of different regions of the nation (Pal et al., 2022). There were major incidents of flooding in the Indian states of Bihar (2020), Kerala (2018), Assam (2016) and Uttrakhand (2013) in the past decade. The flood of 2000 in West Bengal is considered one of the most notable floods as it resulted in significant financial losses and casualties, totalling roughly 2.21 crores of lives and 5560.65 crores of economic output (Government of India, 2023). In August 2018, a devastating flood occurred in the Indian state of Kerala, resulting in the loss of 504 lives and impacting nearly 23 million individuals. Consequently, the state also saw substantial flooding in the monsoon seasons of 2019, 2020, and 2021. This makes proper flood monitoring crucial in reducing fatalities and asset losses. Using Flood Susceptibility Mapping (FSM), the regions that regularly

flood during periods of above-average precipitation can be identified.

1.5 META-LEARNING

Machine learning techniques have made remarkable advancements across various domains. Successful breakthroughs have mainly occurred in areas that allow for the simulation of extensive datasets and leverage high-powered computing capabilities. This eliminates various scenarios where computational resources are limited, or were availability of data is limited. In such situations, the significance of meta-learning becomes apparent. Meta-learning is an architecture that learns to learn across multiple tasks even with limited data (Elsken et al., 2020; Vanschoren, 2018). Meta-learning does this in a data-driven manner. It first learns from meta-data that contains information about previously learned tasks (Rivolli et al., 2022). This meta-data includes the details of the learning algorithm such as the learned model parameters, model accuracy, training time etc (Hartmann et al., 2019). The meta-learning system also learns about the statistical characteristics of the tasks (Castiello et al., 2005). All of this learned knowledge is subsequently employed to guide the meta-learning framework to find optimal models for new unseen tasks (Huisman et al., 2021; Hospedales et al., 2021).

When integrated with contemporary deep learning models, meta-learning offers the ability to address many of the primary concerns raised about deep learning Tian et al. (2022). But similar to deep learning models, meta-learners also have hyperparameters, a type of parameter that needs to be provided as input to the training. Since they have a pivotal function in deciding the model's accuracy, these parameters have to be adjusted and tuned before running the model. Different datasets perform well with different hyperparameter configurations (Yang and Shami, 2020). It is most crucial to identify the hyperparameter settings that work optimally for a given model over a specific dataset (Bischl et al., 2023). Therefore we tune and adjust the hyperparameter values to find the optimal configuration using HPO algorithms (Vanschoren, 2018; Bergstra et al., 2011a). After determining the optimal hyperparameters, the algorithm proceeds to deduce the model parameters using the training data.

HPO is a crucial aspect of meta-learning similar to how it is for classic machine

learning models. Meta-learning models have a number of hyperparameters that influence their efficiency and adaptation capability (Liu et al., 2022b). Meta-learning entails learning from a variety of tasks. Hyperparameters that perform well for one task may fail to adapt well to another. The model can achieve quicker adaption and greater performance on new jobs with properly configured hyperparameters (Khalid and Javaid, 2020). Effective hyperparameter tuning also helps in preventing overfitting (Montesinos López et al., 2022). The resilience and stability of the model is affected by various hyperparameter configurations. With proper hyperparameter optimization, the meta-learning model performs consistently across tasks and maintains its robustness in the presence of data variances. That is, it performs better on a range of tasks when hyperparameters are tuned to guarantee that it responds to the unique characteristics of each task.

Comparing the use of BO and its variants for optimizing hyperparameters of meta-learning benchmark models, bi-level frameworks are built to integrate HPO modules with meta-learning. A series of experiments is conducted to assess the performance of different combinations. In particular, the focus is on assessing how effectively the HPO algorithms have improved the efficiency of metric-based meta-learning models.

1.6 ORGANIZATION OF THE THESIS

This thesis is divided into seven chapters. Chapter 1 provides an introduction to the topics of hyperparameter optimization and applications of hyperparameter optimization which includes a brief overview of flood and LULC analysis. It also provides a description of meta-learning. Chapter 2 extensively deals with the literature review and analyses the relevant past research on the topic. Chapter 3 studies Bayesian optimization delves into a comprehensive exploration of Bayesian optimization and suggests employing genetic algorithm, differential evolution and covariance matrix adaptation - evolutionary strategy for acquisition function optimization. Chapter 4 provides an in-depth description of the methodology and data used for LULC analysis along with the details of machine learning models assisted with the HPO techniques proposed in chapter 3 followed by its outcomes. Chapter 5 describes a flood susceptibility model built

using deep learning models assisted with the HPO techniques proposed. It examines the methodology in detail and analyses the outcomes of flood susceptibility mapping. Chapter 6 discusses meta-learning in detail. It describes the datasets and benchmark models used for solving image classification combined with the proposed HPO models. The chapter also analyses the results of the few-shot meta-learning. It encapsulates the key findings and the inferences derived from the research. Lastly, Chapter 7 presents the conclusion to the thesis. It delineates how this research enhances current contributions and explores the potential for future work. The concluding chapter offers a concise recapitulation of how the initially set research objectives are achieved through the implementation of innovative algorithms.

Chapter 2

BACKGROUND AND LITERATURE REVIEW

2.1 HYPERPARAMETER OPTIMIZATION

The first and basic approach put forward for performing HPO was grid search. Grid search performs an exhaustive search through the Cartesian product of manually specified, finite sets of hyperparameters (Belete and Huchaiah, 2022; Fuadah et al., 2023). It is time-consuming and endures the problem of "curse of dimensionality". In high-dimensional spaces, random search proves to be more efficient than grid search (Bergstra and Bengio, 2012). It randomly chooses a set of hyperparameters from the search space.

Sequential Model-Based Optimization (SMBO) is a formalization of BO (Hutter et al., 2015; Bergstra et al., 2011b; Victoria and Maragatham, 2021; Dewancker et al., 2015; Hazan et al., 2018; Eggensperger et al., 2013; Shahriari et al., 2015). The BO approach treats the black-box objective function as a random function and assumes a prior distribution over the loss function which is updated from new observations to a posterior distribution. In other words, it constructs a probabilistic model that maps the hyperparameters to a probability score, denoted as $p(x|y)$. This model is a surrogate for the expensive-to-evaluate objective function. SMBO is a sequential model that runs multiple trials one after another, finding a more promising set of hyperparameters each time (Zulfiqar et al., 2022; ALGorain and Clark, 2022). The criterion used for selecting the next best hyperparameter values is called an acquisition function. This function utilizes information from the surrogate function to determine the next point for evaluation. These hyperparameter values are then used to evaluate the objective function and the

(probability score, hyperparameter) pairs are finally used to update the surrogate model.

Widely used surrogate models are Gaussian Process (GP), Random Forest (RF) Regressions, and Tree Parzen Estimator (TPE) while the most preferred choice of acquisition function is Expected Improvement. Unlike grid search and random search, SMBO keeps track of past evaluation results. Even though the method is inherently serial and difficult to parallelize, it runs faster than random search (Snoek et al., 2012).

A significant limitation of SMBO lies is the uncertainty regarding the choice of acquisition function. Since it is a sequential model, achieving parallelization is almost futile. Another issue is that the expense for different data varies significantly as the function evaluation step is a laborious procedure.

Further, there is another class of algorithms that are population-based, nature-inspired meta heuristic approaches. The term meta-heuristics comes from two words, meta meaning "beyond" and heuristics meaning "to find". Meta-heuristics are an algorithmic framework that efficiently modifies domain-specific knowledge into heuristics to produce better solutions (Blum, 2003; Maier et al., 2019). This modification generally involves two procedures: exploration (diversification) and exploitation (intensification) and is done with the help of heuristic operators. These heuristic operators are typically derived from natural inspiration and vary across various evolutionary algorithms. Diversification seeks to encompass as much of the search area as possible, while intensification aims to thoroughly exploit the search space to quickly attain improved solutions.

Like all other optimization algorithms, this family of algorithms also try to optimize the objective function in an iterative fashion, by updating the parameter (solution) values. This can be represented by the equation, $y_i = y_{i-1} + \Delta y_{i-1}$ where i indicates the number of iterations, then y_{i-1} is the solution vector in the previous iteration, y_i is the new vector of solutions, and Δy_{i-1} signifies the change in the solution vector after one iteration. Based on how Δy_{i-1} is determined, there are two general categories of algorithms: "Evolutionary Algorithm (EA)" and "Swarm Intelligence (SI)". The former includes algorithms like Genetic Algorithm (GA), Genetic Programming, Evolutionary Strategies (ES), Evolutionary Programming etc., that are based on biological evolution,

with selection, crossover (recombination), and mutation phases. EAs have been proven effective in finding good hyperparameter settings for a wide range of ML problems and are particularly useful when dealing with large and intricate search spaces or when the objective function is noisy or expensive to evaluate (Tani et al., 2021). SI includes Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) and is influenced by the social conduct of natural organisms like birds, fish and ants (Deb, 2005; Orive et al., 2014).

GA are a class of evolutionary algorithms, hence both iterative and population-based (Goldberg, 1989). On each iteration, they work with several solutions collectively called a population rather than a single solution. GA initialises a random population as the solution and updates this solution with the help of three operators, namely, selection, crossover and mutation (variation operators). This process continues until the specified stopping criteria are met. A single iteration is designated as one generation of GA. The term genetic algorithm comes from the similarity of the representation of solutions to chromosomes and that of GA operators to genetic operators. GA has been used to find optimal hyperparameter settings for many ML problems (Ahmadlou et al., 2022; Demir and Åžahin, 2022).

PSO is an algorithm that is based on the behaviour of fish and birds moving in a group (Kennedy and Eberhart, 1995; Lorenzo et al., 2017). ACO is another category of swarm intelligence algorithms which is influenced by the nature of ant colonies searching for food (Dorigo et al., 1996; Costa and Rodrigues, 2018). Both of these algorithms have been used for hyperparameter optimization of different ML models (Lakra and Jena, 2022; Xiong et al., 2021).

Grid search, random search and population-based methods like the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) (Loshchilov and Hutter, 2016) are the common model-free paradigms used for hyperparameter tuning in AutoML systems. Apart from these model-free black-box optimization techniques, Bayesian optimization, HPO in AutoML focuses on multi-fidelity methods that are cheaper compared to the pure black-box methods. This includes an early stopping algorithm, a simple selection algorithm and other adaptive choices of fidelity.

A major hurdle in the optimization procedure is the exhaustive runtime of the algorithms which increases with the amount of data and number of hyperparameters. Multi-fidelity methods focus on finding a solution to the time and resource constraints in HPO. Reducing the data set for training and bringing down the number of features, iterations, and cross-validation folds are a few manual approaches used to accelerate the tuning process. These ideas are used by low fidelity methods to find an approximation of the actual objective function to minimise.

Learning curve-based prediction is one such method that gets rid of configurations that are anticipated to perform badly (Domhan et al., 2015). It's an early stopping algorithm that examines the learning curve during HPO and stops the training operation for a certain hyperparameter setting if the curve isn't expected to meet the performance level of the best model produced up until that point in the optimization process. Implementation of these models combined with the Bayesian optimization technique called the Freeze-Thaw Bayesian optimization is mentioned in Swersky et al. (2014). The algorithm maintains a set of frozen configurations and uses an information-theoretic decision framework to either thaw (defreeze) a setting chosen by the Bayesian optimization and continue training or train a new configuration to find the best hyperparameter settings.

Selection algorithms like Successive Halving (SH) and Hyperbands are bandit-based strategies that are pure-exploration focused resource allocation problems (Soper, 2023). Pure exploration problems are also called best-arm identification problems. The goal is to choose the best arm (here, the best settings with the maximum expected performance) with maximum confidence. Successive halving is one such non-stochastic best-arm identification problem proposed by Jamieson and Talwalkar (2015). The algorithm can be summarized in 3 steps. 1) At first, it uniformly allocates resources to each set of hyperparameter configurations. 2) And then assess how they perform. 3) Lastly, the algorithm removes half of the worst-performing group. The process is iterated till a single configuration remains. Through the sequential elimination step, the algorithm guarantees more resources for more reliable configurations (Karnin et al., 2013).

Hyperband is an extension of the successive halving algorithm put forward by Li

et al. (2018). In successive halving, the fixed budget of resources is uniformly distributed to all configurations initially. If there are n configurations and a total budget B of resources, then SH allocates B/n resources for each setting. Whereas hyperband takes into consideration the fact that a large number of the configurations (large n) will require only a small amount of resource (small B/n) or vice versa. This is called the n versus B/n issue. Hyperband addresses this by letting different possible values of n for a fixed B and allocating a minimum resource r for all the configurations before the elimination step. And then calls SH on random samples of configurations as a subroutine (Dores et al., 2018).

The multi-task Bayesian optimization technique is an adaptive fidelity technique that learns from previously trained models or a trained subset of a large dataset (Swersky et al., 2013). They use multi-task Gaussian process models for fastening the Bayesian optimization by studying the correlation among tasks. Other adaptive choices of fidelity include algorithms like Bayesian Optimization with Hyperband (BOHB) (Falkner et al., 2018), Multi-Fidelity Gaussian Process Upper Confidence Bound (MFGP-UCB) (Kandasamy et al., 2016) and Bayesian optimization with Continuous Approximations (BOCA) (Kandasamy et al., 2015). BOCA employs an Upper Confidence Bound (UCB) acquisition function to aid the optimization process. Apart from the above mentioned methods, there are many more recent works in HPO on different applications (Cui and Bai, 2019; Baldib and L. Gillen, 2022; Mohan and Badrah, 2022; Osama Ahmed et al., 2020; Guo et al., 2019; Zahedi et al., 2021; Amirabadi et al., 2020). Direct search class of black-box optimization methods have also been adapted to HPO of deep ML models (Lakhmiri et al., 2021; Lakhmiri and Digabel, 2022).

2.2 AUTOML SYSTEMS

AutoML systems streamline the complete machine learning process by automating four distinct phases - data preparation, feature processing, model generation and estimation. The user only needs to submit data, and the AutoML system will automatically decide which strategy is optimal for a particular application. The primary goal of AutoML systems is to optimise the performance by automatically setting the best hyperparam-

eters, i.e., automating the hyperparameter optimization part. It also has several other use cases. It decreases the amount of human labour required to apply machine learning, increases the effectiveness of machine learning algorithms, and makes scientific investigations more reproducible. Some of the popular AutoML frameworks and the optimizers they use are listed below:

1. ML models-based frameworks

   (a) Auto-WEKA

   Auto-WEKA is a fully automated Automl system that includes feature selection. It uses a Bayesian optimization framework for HPO (Kotthoff et al., 2018).

   (b) Auto-sklearn

   Auto-sklearn has feature processing and pre-processing units. It considers 15 classifiers and 110 hyperparameters and considers previous performances on similar datasets. Auto-sklearn can also construct ensemble models using a TPE-based BO for HPO (Feurer et al., 2018).

   (c) Tree-based Pipeline Optimization Tool (TPOT)

   TPOT is an AutoML system based on genetic programming. It employs a GA to optimize a sequence of feature preprocessors and machine learning models, thereby improving classification accuracy (Olson and Moore, 2018).

2. DL models-based frameworks

   (a) AutoKeras

   AutoKeras focuses on deep learning tasks and uses BO to guide neural architecture search. It also employs a neural network kernel and a tree-structured acquisition function optimization technique to effectively evaluate the search space (Jin et al., 2019).

   (b) Auto-PyTorch

   Auto-PyTorch is a Neural Architecture Search (NAS) library based on the

same principles as AutoKeras, but with a PyTorch backend. It employs BO combined with hyperband for HPO (Zimmer et al., 2020).

(c) AutoGluon

AutoGluon uses multi-layer ensemble models and performs complex data processing and deep learning (Erickson et al., 2020).

(d) H2O AutoML

Make use of one layer of ensemble stacking combined with bagging and a strong base model - XGBoost tree ensemble (LeDell and Poirier, 2020). H2O uses random search for hyperparameter tuning.

3. Hyperparameter tuners

Packages that use Bayesian optimization include Sequential Model-based Algorithm Configuration (SMAC) (Hutter et al., 2011; Lindauer et al., 2022), Spearmint, Hyperopt (Komer et al., 2019), Scikit-optimize, BoTorch etc. Packages like Differential Evolution Hyperband (DEHB) (Awad et al., 2021), DEAP (Fortin et al., 2012) and Nevergrad make use of evolutionary algorithms, whereas Optuna (Akiba et al., 2019), Orion and RayTune implement both Bayesian optimization and evolutionary algorithms.

2.3 LAND USE LAND COVER CLASSIFICATION

Land use and land cover (LULC) classification involves analyzing satellite imagery and other geospatial data to categorize regions of the Earth's surface based on their physical characteristics and human usage. This process employs algorithms to classify pixels in the imagery into predefined categories, such as forests, urban areas, water bodies, and agricultural lands, aiding in environmental monitoring, urban planning, and resource management. The number of classes in LULC classification problems can vary depending on the specific study and the level of detail required. Commonly, LULC classifications include a range of 5 to 20 classes, but some detailed classifications can include even more. Typical classes include: Barren land, build-up, dense vegetation (forest), sparse vegetation (agriculture) and waterbody. There are numerous studies related to LULC classification. These studies often focus on developing and applying various

methods and algorithms to improve the accuracy and efficiency of LULC classification.

Pijanowski et al. (2002) used ANNs to assess LULC models, its predictive power and understand the regional patterns of land development in Grand Traverse Bay Watershed, Michigan. Rienow and Goetzke (2015) produced an LULC model in Federal state of North Rhine Westphalia using SVM and binomial logistic regression (BLR). They used cellular automata (CA) in conjunction with SVM and BLR to examine spatially explicit simulation of changes in land-use and land-cover to predict urban expansion. Ansari and Golabi (2019) used an maximum likelihood classifier (MLC) in the wetlands of Meighan, Iran using the ERDAS 2014 Imagine model to produce LULC images. The resulting LULC images were analysed using the change detection feature. To enhance the assessment of wetland deterioration, the output from the LULC was employed to predict future LULC changes using a land change model. Mu et al. (2019) generated a LULC model in Wuhan, the city of Hubei Province in China using MLC for classification and deep learning for prediction. For predicting urban LULC shift, they used a self-adaptive cellular-based deep learning model using data from several sources. And a long short term memory (LSTM) network which is excellent at handling sequence data, is used for urban land use change forecasting. Tassi and Vizzari (2020) developed and tested an object-oriented classification approach and employed random forest (RF) and SVM for performing a LULC classification in Meighan Wetland Iran. Improved k-NN, logistic regression, RF and SVM were put into use by Nguyen et al. (2020) in the regions of Dak Nong, Vietnam. They used images of the years 2017 and 2018. The rainy season, dry season, the complete year of 2017, and the combination of the dry and wet seasons were each tested and classified independently. Five of the eleven LULC classes that were differentiated were forests. The overall accuracy of the model for all historical periods and classifiers ranged from 63.9% to 80.35%. Liu et al. (2020a) developed a LULC model using RF algorithm in Gannan Prefecture in the Northeastern Tibetan Plateau, China. The outcomes were verified using a confusion matrix. A geographical detector was employed to explore the impact of anthropogenic and natural variables on LULC changes in Gannan Prefecture, and the land use transfer matrix was utilised to analyse the change in each LULC type. Silva et al. (2020) classified the land-use land

cover of the Taperoa River basin in Northeastern Brazil. Maxver classification was employed for LULC classification and an multilayer perceptron (MLP) model was used for the dynamic modelling of the land cover. Sertel et al. (2022) classified the Marmara Region of Aksu and Kestel using DeepLabv3+ architecture with ResNeXt50 encoder. They conducted a comparison of various segmentation architectures and encoders to ascertain the optimal configuration for generating highly accurate LULC maps. The best performance was obtained for the DeepLabv3+ architecture with a ResNeXt50 encoder, which has a recall of 94.49%, an IoU of 89.46%, an accuracy of 94.25%, and an F-1 score of 94.35%. Clark et al. (2023) employed a UNet architecture to build a LULC map of North Queensland, Australia. They demonstrated the impact of adjusting network hyper-parameters on the efficacy of the model for semantic segmentation of land use and land cover. The paper examined the impacts on classification accuracy and training time by altering parameters, including the number of initial convolutional filters, activation functions, network depth, learning rate, kernel size, kernel initializer, loss function, and the optimization function used to optimize loss.

In India, Jat et al. (2017) developed a LULC map of Ajmer city of Rajasthan using the MLC. In order to examine urban expansion, LULC maps were created from the categorization of moderate-resolution remote sensing images using MLC. Only spectral reflectance values are used by the MLC for classification. Misclassifications were noted as a result of shared reflectance properties of a few LULC classes, variability in built-up areas as a result of various construction methods and materials, and shortage of open areas between the built-up units. Bose and Chowdhury (2020) made use of MLC to classify land use land cover of Siliguri in West Bengal. MLC was employed to do supervised classification using ArcGIS 10.2.2 software to produce an ANN-MLP model urban growth probability map. Abijith and Saravanan (2022) used RF algorithm to classify the regions of the Northern Tamil Nadu coast. Landsat images were used to create LULC images, which are then categorised in GEE using RF. The CA-Markov model was then employed to produce LULC maps in order to project future LULC changes. Kulithalai Shiyam Sundar and Deka (2022) classified the Vembanad Lake system, Kerala, using RF, SVM classifier and classification and regression trees (CART). Three ML

approaches were used to map LULC using GEE. SVM is found to perform poorly when the results of these three algorithms were compared, with an average accuracy of about 82.5%. CART came second with an accuracy of 87.5%. RF model performed well with an average accuracy of 89.5%.

Anthropogenic and natural activities have an effect on the land cover, changing its landscapes and the ensuing nature of biological events (Silva et al., 2020). Tracking and evaluating urban expansion benefits in managing and using natural resources now and in the near future. The Earth's land surfaces have changed due to human activity on nearly half of the planet (Tayyebi and Pijanowski, 2014). These alterations are referred to as land use and land cover changes (LULCC). Growth in the economy and population in recent decades have resulted in unforeseen urbanization and industrialization. The need for vital infrastructures including sewage services, water supplies and recreational activities surged as a result of this massive urban growth. Additionally, it has a negative impact on urban heat island (UHI), pollution, concerns associated with changes in the climate, urban flooding, and congested roadways. Consequently, LULC change is viewed as a crucial environmental problem with significant worldwide implications. Urbanisation, erosion, overgrazing, and land degradation are examples of human-induced factors that contribute significantly to biodiversity and habitat loss (Saxena et al., 2021). Additionally, nature also adds to this change (Halmy et al., 2015; Lambin, 1997). A complex interplay of variables, including policy management, environmental concerns, social demands, economics and cultural norms lead to changes in LULC. Altering the LULC can significantly impact water quality, as agricultural and urban activities contribute to the presence of phosphates and nitrates in freshwater (Álvarez-Cabria et al., 2016).

Developing nations like India, Malaysia, Indonesia and Sri Lanka are seriously threatened by the rapid urbanisation that is taking place in various regions. With 1.38 billion people, India is one of the populous countries in the world. Numerous Indian cities, including Bengaluru, Delhi, and Chennai, may experience 'Day Zero' in the future years, as per the composite water management index (CWMI) report released by the Government of India (GOI)-affiliated National Institution for Transforming India

(NITI) Aayog in August 2019 (Abijith et al., 2020). In order to cater for the needs of the increasing population and ensuing urbanisation, strategies for planning, evaluating, and monitoring land use transformations will be necessary. These vast areas of forests are being turned into other types of land uses, severely damaging the soil. Rapid soil erosion may cause devastating landslides and floods that harm downstream residents. So, for long-term habitation and environmental betterment, a sustainable LULC is essential (Mishra et al., 2020).

Substantial changes have occurred in the LULC of India during the last 140 years, like the loss of vegetation, altered agricultural land, and increased urbanisation. As per UN anticipation, by 2050, 60% of the world's agrarian populations will be located in metropolitan cities (Parvinnezhad et al., 2021) due to inadequate planning and development, resulting in unplanned urban expansion. The intricacies of future agricultural output, accompanying LULC change, and environmental effects may be investigated using spatial modelling. Anthropogenic and environmental processes relating to temporal dynamics and potential changes in the land cover must be properly understood. The expansion of LULC on a global scale and the assessment of the transformation of different land types represent just a couple of the land use analyses facilitated by the advancement of remote sensing and the incorporation of machine learning (Wang et al., 2022).

LULCC is key to modifying the global environment, which significantly influences diversity, biological cycles, and ecosystem dynamics (Santos et al., 2021; Schwalm et al., 2017). These modifications immediately degrade the landscape and affecting the land surface, resulting in greenhouse gas emissions, biodiversity loss, degraded soil resources, and a shift in the global climate system (Hamad et al., 2018b,a; Liu et al., 2020b). To model future LULCC in this situation, it is crucial to identify the root causes of these alterations (Giri et al., 2003; Ruben et al., 2020).

2.4 FLOOD SUSCEPTIBILITY MAPPING

Flood susceptibility prediction involves using statistical and machine learning models to identify areas at risk of flooding based on factors like topography, rainfall, soil type, and land use. Flood susceptibility mapping visualizes these predictions on a map, high-

lighting regions with varying levels of flood risk to aid in disaster preparedness and management. The number of classes in flood susceptibility prediction typically ranges from 2 to 5, depending on the granularity of the classification scheme used. Common classes include very low susceptibility, low susceptibility, moderate susceptibility, high susceptibility and very high susceptibility. In simpler models, the classification might be binary, with just two classes: susceptible and non susceptible. There are numerous literature on flood susceptibility. Studies often focus on developing and comparing different predictive models and mapping techniques to enhance the accuracy and usefulness of flood susceptibility assessments.

Traditional hydrological approaches were employed for flood susceptibility modelling in the earlier days. These methods demand substantial funding and fieldwork for data collection, due to which statistical and data-driven approaches were put forward for FSM. Flood modelling has also been greatly aided by the Geographical Information System (GIS) and remote sensing techniques, which include a variety of data sources, superior data quality, day and night data collection capabilities, and speedy analysis for FSM (Tehrany et al., 2013; Fenglin et al., 2023).

The deployment of all these approaches is hampered by the need to obtain data from multiple providers as well as computational speed limitations. Google Earth Engine (GEE) is an open cloud computing platform that helps in the processing of enormous amounts of remote sensing data rapidly and efficiently. Synthetic Aperture Radar (SAR) imagery is popularly used for flood inundation (Jiang et al., 2021; Shen et al., 2019), flood mapping (Anusha and Bharathi, 2020; Li et al., 2019; Weydahl, 1996; Amitrano et al., 2018) and flooding water depth estimation (Iervolino et al., 2015). SAR can function at wavelengths that are not hampered by cloud cover or a lack of illumination and can attain data during day or night, in any weather. Sentinel-1's C-band SAR instrument can provide dependable, repeated extensive monitoring with variable resolution, ranging down to 5 meters, and coverage extending up to 400 kilometers. It supports dual polarisation, has brief revisit times, and delivers products quickly.

From 2018 through 2021, the Indian state of Kerala was repeatedly vulnerable to catastrophic flooding, yet there have been no studies employing machine learning to

identify these flood-prone areas of the entire state. Therefore, it is necessary for the study to model the area's sensitivity to flooding. The region's sensitivity to flooding is evaluated by considering 12 flood conditioning factors. Multiple machine learning models are used and compared based on their performance in predicting flood risk zones. This includes 4 decision tree-based ensemble ML models (AdaBoost, RF, Gradient Boost, and Extreme Gradient Boost (XGBoost)), Convolutional Neural Network (CNN) - 1D, 2D and 3D. AutoML models (AutoKeras and TPOT) to aid FSM are also used, where more than 15 machine learning models are employed for flood susceptibility mapping, giving the best-performing model at high computational speed. Hyperparameter optimization is employed on hyperparameters of the CNN model that were considered crucial in improving the efficiency of the model. In particular, an evolutionary hyperparameter optimization-based deep learning technique is presented that helps to produce effective flood susceptibility mapping. Various evaluation metrics including accuracy, cross-entropy loss and AUC were used to gauge the model's effectiveness.

Different decision-makers and hazard managers can use flood susceptibility maps to lessen flooding-related harm and damage to the built infrastructure. The overall process of our work involves the following steps: (a) data preparation and collection of flood conditioning factors; (b) developing the flood inventory map; (c) modelling flood susceptibility with AutoML, ML models and CNN models; (d) developing flood susceptibility maps; and (e) validation and evaluation of the models using different metrics and flood susceptibility maps.

Identifying flood-prone areas helps in reducing the consequences of floods. Accurate susceptibility analyses are carried out by scientists and governments to alleviate the adverse consequences of floods. Various statistical (Mousavi et al., 2022) and ML - supervised, semi-supervised and unsupervised methods (Ghosh, 2023; Elkhrachy, 2022; Tanim et al., 2022; Shahabi et al., 2020; Mateo-Garcia et al., 2018; Prasad et al., 2022; Abedi et al., 2022; Zhao et al., 2019; Saravanan and Abijith, 2022; Hasanuzzaman et al., 2022; McGrath and Gohl, 2022; Mahdizadeh Gharakhanlou and Perez, 2023) have been employed in building flood susceptibility maps over the last few years. The fundamen-

tal tenet of statistical approaches is that past flood events are strongly correlated with present flood risk factors, such as frequency ratio, the weight of evidence, logistic regression etc.

The flood modelling approaches can be generally categorized into four basic groups based on the available research on flood susceptibility models. They are (i) hydrological models including Hydrologic Engineering Center-River Analysis System (HEC-RAS) (Jha and Afreen, 2020), Hydrologic Engineering Center's Hydrologic Modeling System (HEC-HMS) (Romali et al., 2018; Sarchani et al., 2021), Soil and Water Assessment Tool (SWAT) (Sufiyan and Magaji, 2018; Yu et al., 2018), HYDROTEL (Aissia et al., 2012), HSAMI (Bajamgnigni Gbambie et al., 2017), Hydrologiska Byrans Vattenbalansavdelning model (HBV-light) (Sarchani et al., 2021). (ii) data-driven and statistical approaches such as Monte Carlo (Garrote et al., 2021), Evident Belief Function (EBF) (Chowdhuri et al., 2020), Logistic Regression (LR) (Giovannettone et al., 2020), Frequency Ratio (FR) (Sarkar and Mondal, 2020), Weight of Evidence (WoE) (Rahmati et al., 2016a). (iii) Multi-criteria decision-making models such as Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) (Rafiei-Sardooi et al., 2021), Vlsekriterijumska Optimizacija I Kompromisno Resenje (VIKOR) (Malekian and Azarnivand, 2016), Simple Additive Weighting (SAW) (Nawindah, 2017), Analytical Hierarchical Process (AHP) (Patrikaki et al., 2018; Rahmati et al., 2016b), Analytical Network Process (ANP) (Yariyan et al., 2020), Fuzzy-AHP (Ekmekcioğlu et al., 2021). (iv) ML techniques such as Naive Bayes (Li et al., 2023c), artificial neural networks (Wang et al., 2021; Ahmadlou et al., 2020; Cui et al., 2023), decision trees (Tehrany et al., 2013; Bui et al., 2019), RF (Abedi et al., 2022), and Support Vector Machine (SVM) (Youssef et al., 2022; Liu et al., 2022a; Mehravar et al., 2023). More recently, CNN (Ahmadlou et al., 2020; Khosravi et al., 2020; Zhao et al., 2020), fuzzy computing techniques (Balogun et al., 2022), ensemble models (Yaseen et al., 2022; Li and Hong, 2023) and meta-heuristic algorithms (Arabameri et al., 2022; Razavi Termeh et al., 2018; Bui et al., 2020) have been applied to FSM in many works and were found to be promising. Zhao et al. (Zhao et al., 2020) employed a fixed LeNet-5 design and achieved satisfactory results more quickly than with other models.

Lee et al. (2017) studied the FSM for Seoul metropolitan city in Korea and found that the RF algorithm performed better than the Boosted Tree models. The findings of Chakrabortty et al. (2022) showed the practical applicability of the PSO model over the traditional Artificial Neural Network (ANN) and Deep Learning (DL) neural networks. Saravanan et al. (2023) stated that the decision tree-based ensemble models show better predictability in the application of floods and landslides susceptibility. Deep learning has garnered significant attention due to its ability to yield reliable outcomes that are comparable to or surpass traditional machine learning techniques (Saha et al., 2022). DL possesses numerous advantages compared to alternative methods. Its applicability and usefulness grow significantly with the expansion of training dataset sizes. Moreover, DL models have grown in size over time due to advancements in computer infrastructure and speed. DL models have the capability to solve intricate real-world problems by progressively enhancing their accuracy. Additionally, DL models can perform unsupervised and semi-supervised learning. In the study conducted by Wang et al. (2019), the application of 1D, 2D, and 3D CNN was employed to analyze landslide susceptibility mapping. The findings revealed that the predictive accuracy of the 2D CNN model surpassed that of the 1D and 3D CNN models in generating susceptibility maps. Khosravi et al. (2020) studied the prediction of flood hazards using 2D CNN for Iranian cities. (Zhao et al., 2020) conducted a research study on urban flood susceptibility, wherein the performance of two CNN models, namely SCNN (Simple CNN) and LeNet-5, were compared with that RF and SVM. The results of the study unveiled that both SCNN and RF exhibited superior performance compared to the other two models. A crucial element that has been absent in all of these studies is the inclusion of efficient hyperparameter tuning of models. HPO helps in improving efficiency, bettering the performance of the model and understanding the model behaviour. The aim of this study is to utilize various HPO techniques to aid in developing FSM models.

2.5 META-LEARNING

Though machine learning models have been proven highly effective in applications that utilise huge amount of data, they often struggle with inadequate or limited data. Few-

Shot Learning (FSL) was suggested as a solution to this issue. Humans have demonstrated an exceptional capacity for learning from a limited set of examples and generalising to a wide range of unseen scenarios (Song et al., 2023). In order to replicate this human ability of generalisation, FSL was proposed to make quick generalisations to unfamiliar tasks including merely a small amount of samples with supervised information using prior knowledge (Wang et al., 2020b). In FSL, learning several parameters using only a few examples will be quite challenging and most certainly result in overfitting. Meta-learning techniques have been introduced to address the challenges posed by few-shot learning (Xu et al., 2021).

Meta-learning includes gathering information from diverse learning tasks, usually spanning a variety of related tasks, and applying it to enhance subsequent learning performance. This “learning-to-learn" approach is similar to how humans learn, where the methods of learning advance over the course of a lifetime and across evolutionary periods (Wang, 2021). It can be used in both single-task and multi-task scenarios. In single-task cases, the same problem is fixed over the course of several sessions. Task-agnostic knowledge is gained through a set of tasks in multitask contexts and utilised to enhance the learning of novel, unencountered tasks belonging to the group (Vilalta and Drissi, 2002; Bhatt et al., 2012).

Meta-learning involves initially training the network on basic classes, often referred to as common classes, which has a sufficient number of samples. Subsequently, the model is allowed to learn new categories, even when provided with only a limited number of instances. The central concept of a meta-learning framework for few-shot learning is learning to learn (Thrun and Pratt, 1998; Sun et al., 2019). That is, acquire the ability to fine-tune a base learner to a new task in situations where the available number of labeled samples is limited. This is achieved by leveraging a substantial number of analogous few-shot problems (Li et al., 2017). In particular, the meta-learner selects training samples from the base classes and optimises the model in order to do well on the few-shot classification tasks. Generally, an n-way and k-shot classification task typically consists of n classes, each of which has k support samples and q query samples. The objective is to group these n x q query samples into n classes using the n x k support

samples as a basis (Chen et al., 2021).

Luo et al. (2022) classified meta-learning into 3 types based on the information or experiences meta-learners intend to acquire. These are a) optimization-based, b) model-based and c) metric-based.

1. Optimization-based

   In optimization-based systems, meta-task is considered an optimization problem. There are two levels for this optimization. The goal of the outer-level optimisation is to get the meta-data or information from the meta-task. While the inner-level optimization attempts to enhance the learning of target-task. This method is usually assisted by gradient descent. Model Agnostic Meta-Learning (MAML) (Finn et al., 2017), Reptile (Nichol and Schulman, 2018) and Almost No Inner Loop (ANIL) (Raghu et al., 2019) are some of the examples.

2. Model-based

   A model-based meta-learning framework has networks with inner-level and outer-level components of memory. These are black-box meta-learning techniques (Luo et al., 2022). Model-based approaches learn from a limited number of training samples, relying on the model parameters for learning. The meta-learners learn the model parameters from meta-training data (Santoro et al., 2016). These approaches work well for few-shot learning. Some of the commonly used model-based learners include Simple Neural Attentive Meta-Learner (SNAIL) (Mishra et al., 2018), Conditionally Shifted Neurons (CSN) (Munkhdalai et al., 2018).

3. Metric-based

   Metric-based meta-learning model learns on a distance metric. It is a non-parametric learning method. It learns from the similarity between training data and testing data points. These models are similar to K-Nearest Neighbors (KNN) classifiers. The evaluation metrics signify the similarity between meta-task and target-task. Examples include Siamese Network (Koch et al., 2015), Matching Network (Vinyals et al., 2016), Prototypical Network (Snell et al., 2017), Relation Network (Sung et al., 2018) etc.

2.6 RESEARCH GAP

The examination of existing literature revealed the presence of the following research gaps.

- Despite the success of many HPO techniques, some machine learning models and problems still need more efficient and novel HPO methods to optimize their hyperparameters effectively. Traditional HPO methods can be computationally expensive and time-consuming, often failing to explore the hyperparameter space adequately. Consequently, there is a growing demand for innovative HPO strategies that can address these limitations and enhance model performance across diverse and complex datasets.

- HPO has been proven effective on many publicly available datasets (e.g., CIFAR-10 and MNIST). Those datasets are usually for research purposes; therefore, most of the images are well-labelled. However, in real-world situations, the data is often inadequate and mislabelled. It is yet to explore how well the HPO-assisted ML models work for real-world datasets. Moreover, real-world data frequently presents additional challenges such as class imbalance, noisy labels, and diverse data distributions, which are not typically encountered in clean, well-curated research datasets. These factors can significantly impact the performance of HPO techniques, necessitating further investigation into their robustness and adaptability to real-world conditions. Addressing these challenges is crucial for developing HPO methods that are not only theoretically sound but also practically effective in diverse and imperfect real-world environments.

- It is proven that for large-scale datasets, HPO techniques identify potential configuration settings faster than humans. However, it is yet to be analyzed how well they work in scenarios where labeled data are scarce, specifically for image classification problems. Traditional HPO methods often rely on abundant labeled data to evaluate model performance, which may not be feasible in many real-world applications. Additionally, in the context of few-shot learning and semi-supervised learning, the effectiveness of existing HPO techniques remains uncertain. There-

fore, further research is needed to develop and validate HPO methods that can efficiently handle limited labeled data scenarios, ensuring robust performance in such challenging conditions.

2.7 RESEARCH OBJECTIVES

Based on the research gaps, the following objectives are formulated for the present research work.

- Analyse different hyperparameter optimization methods for large-scale datasets for image classification problems with the help of AutoML methods and to develop novel approaches that can outperform the existing methods in terms of accuracy by using nature-inspired algorithms for optimization.

- Use the existing AutoML models and proposed HPO methods to handle real-time image classification problems and compare performance with present ML models used to solve them.

- Use meta-learning approaches assisted by HPO to analyse how they capture underlying patterns and representations of image classification tasks with few-shot samples, allowing the model to generalise more effectively to new, unseen images.

2.8 CONTRIBUTION OF THE THESIS

The research contributes significantly to the field of image classification by leveraging HPO methods to enhance performance of machine learning models. Initially, contemporary HPO models were evaluated, focusing on Bayesian optimization. Evolutionary algorithms were then employed to maximize the acquisition function within Bayesian optimization, resulting in a hybrid model that demonstrated superior performance compared to traditional Bayesian methods. This hybrid approach harnesses the robustness and parallelizability of evolutionary algorithms, which are adept at handling noisy objective functions without requiring gradient information.

In addressing real-time image classification problems, the study applied the proposed HPO methods to LULC classification and flood susceptibility mapping. By using multiple ensemble models augmented with the novel HPO techniques, significant improvements in classification accuracy were observed. For LULC classification, the enhanced models accurately categorized various land types, demonstrating the efficacy of the proposed methods in environmental monitoring and urban planning. Similarly, the flood susceptibility mapping benefited from the optimized CNN models, leading to better predictions and aiding disaster management efforts.

Further, the research explored meta-learning approaches to address few-shot learning scenarios. By integrating HPO with metric-based meta-learning models, the study achieved improved generalization to new, unseen images, even with limited samples. This demonstrates the potential of combining HPO with meta-learning to capture underlying patterns and representations more effectively.

Overall, the research advances the state of the art in image classification by introducing robust optimization techniques, improving real-time application performance, and enhancing few-shot learning capabilities. These contributions provide a foundation for further exploration and application in various domains requiring accurate and efficient image classification.

Chapter 3

HYPERPARAMETER OPTIMIZATION

3.1 INTRODUCTION

Hyperparameter optimization is a crucial aspect and ubiquitous problem in machine learning that can drastically affect the performance of a model. Despite multi-fidelity optimization's popularity and success, there exist machine learning challenges that haven't been explicitly addressed by existing HPOs and may require unique methodologies. Because it is so expensive to train even a small neural network on massive datasets, there is a lack of research in the domain of hyperparameter optimization for deep neural networks. Therefore, it is beneficial to study and analyse prevailing techniques to determine effective ways to improve them and to discover novel approaches that outperform existing ones.

In order to achieve this, the contemporary models used for HPO were analysed with the help of AutoML tools. Then focus was directed towards Bayesian optimization and methods for enhancing the traditional Bayesian optimization framework were explored. With the purpose of achieving this objective, evolutionary algorithms are employed to maximize the acquisition function. The combination of three evolutionary algorithms with Bayesian optimization were compared to determine which of them could outperform traditional Bayesian optimization.

The class of evolutionary algorithms is commonly used as global optimizers. They are known for their robustness in evaluating noisy objective functions and are easy to be parallelized. EAs are conceptually simple algorithms that are powerful in capturing the global optimum for complex optimization problems. These methods are straight-

forward to implement and don't demand stringent constraints. Unlike gradient descent, they do not require any gradient information. EAs are not influenced by the continuity or differentiability of the objective function. Though most of the EAs require longer runtime for convergence, they tend to improve traditional optimization techniques when combined as hybrid models (Srinivasan and Seow, 2003; Pant et al., 2008; Mashwani, 2008).

3.2 MATERIALS AND METHODS

This section compares HPO models that combine Bayesian optimization with evolutionary algorithms. The concept of combining BO and EA has been explored in earlier studies. STEADE is such a hybrid model, which is an evolutionary algorithm with surrogate assistance for HPO of ML models (Biswas et al., 2020). STEADE uses a mix of surrogate models (Radial Basis Function (RBF) and GP). The RBF model is used for the initial parameter space exploration, and the knowledge is then transferred to a GP-based Bayesian optimization framework that is additionally enhanced by a Differential Evolution (DE) method. Cho et al. (2022) present a hybrid method for NAS which combines evolutionary algorithms with Bayesian optimization. The method called B^2EA, uses two Bayesian optimization surrogate models within an EA to guide the search process: one to optimize the architecture-level hyperparameters and one to optimize the weight-level hyperparameters.

Another hybrid approach that merges BO with a genetic algorithm called GA-PARSIMONY to search for parsimony models was introduced by de Pison et al. (2019). Parsimony models are machine learning models that are simple and interpretable, and this method aims to find such models and involves a combination of hyperparameter optimization and feature selection. The method starts with BO being used to optimize the hyperparameters of the model. Then it uses the GA-PARSIMONY algorithm to perform feature selection by searching for a subset of features that results in the best performance.

Lan et al. (2022) proposed a method that integrates Bayesian optimization with evolutionary algorithms with an aim to reduce the total time taken for optimization. The Bayesian optimization algorithm is used to optimize the parameters of the evolutionary

algorithm, such as the population size or the mutation rate. This way, the method aims to balance the global search capability of evolutionary algorithms with the local search capability of Bayesian optimization. Our work focuses on a single surrogate model (GP) and explores the use of EAs for acquisition function optimization. The working of Bayesian optimization and details of each evolutionary algorithm used in this section (along with their pseudocodes) are described briefly in the following paragraphs.

3.2.1 Bayesian Optimization

Bayesian optimization is an optimization technique that employs a probabilistic method based on Bayes' theorem to identify the global optimum of a black-box function. It has two major parts. The first component is a surrogate model that is probabilistic and comprises a prior distribution which represents the unknown objective function. The acquisition function is the second component, and it is optimised for selecting the next point to sample.

A surrogate model is the probability representation of the objective function, F. It produces a posterior probability distribution using the Bayes rule, which represents possible $F(\lambda)$ values for a given hyperparameter configuration λ. The commonly used surrogate model is Gaussian Process. A Gaussian process generates a posterior probability distribution to approximate the actual objective function F. That is, it gives a mean (μ) and a standard deviation (σ) of the function F at any point λ. The equation for GP is given below:

$$G(\lambda|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(\lambda-\mu)^2}{2\sigma^2}}. \tag{3.1}$$

In order to get a more accurate approximation of F, it is necessary to observe F at new query points. To determine the next query point, a function called acquisition function is used. The point where the maximum value of the acquisition function is identified is chosen as the subsequent query point. The new query point is given by the equation:

$$\lambda = \underset{\lambda\in\Lambda}{\operatorname{argmax}}\ AF(\lambda;D), \tag{3.2}$$

where AF is the acquisition function to maximize and D is the data observed so far.

The popularly employed acquisition function is called Expected Improvement, E.

Expected Improvement of a single point for a Gaussian posterior is given by:

$$E(\lambda) = \mathbb{E}[max(f(\lambda) - f(\lambda_{best}), 0)], \tag{3.3}$$

where $\mathbb{E}$ denotes the expected value and $f(\lambda_{best})$ is the best function value observed so far.

Upon observing F at a new λ, the Gaussian process (the posterior distribution) is updated. And then the value of expected improvement is updated. In the next iteration the expected improvement is maximized leading to a new query point. This process is repeated until a number of iterations or until the new query point remains a constant. Then that particular point is considered as the minimum value of objective function. The algorithm 1 represents the pseudo-code for BO. And the Figure 3.1 illustrates BO on a one dimensional function.

Algorithm 1: Pseudo-code for BO

Input: Objective function F, Gaussian process model G, acquisition function E, hyperparameter search space Λ, maximum number of iterations T
Output: Best hyperparameter configuration λ_T

```
Initialize data D_0 with initial observations
for t ← 1 to T do
    Fit the Gaussian model G_t on D_{t-1}
    Select next query point: λ_t that maximise E(λ; D_{t-1}, G_t)
    Query F(λ_t)
    Update data: D_t ← D_{t-1} ∪ {< λ_t, F(λ_t) >}
```

The use of evolutionary algorithms like Differential Evolution, Genetic Algorithm and Evolutionary Strategy for maximizing E is explored in this chapter. The foundation of every evolutionary algorithm is biological evolution, which is natural selection and continual blending of variation via recombination and mutation. New individuals (candidate solutions) are produced in each iteration (generation) by variation, typically in a stochastic way, from the existing parental individuals. Then, based on their fitness score, some individuals are chosen to become the parents of the following generation. After each generation, individuals with greater and better fitness values are generated in this manner.

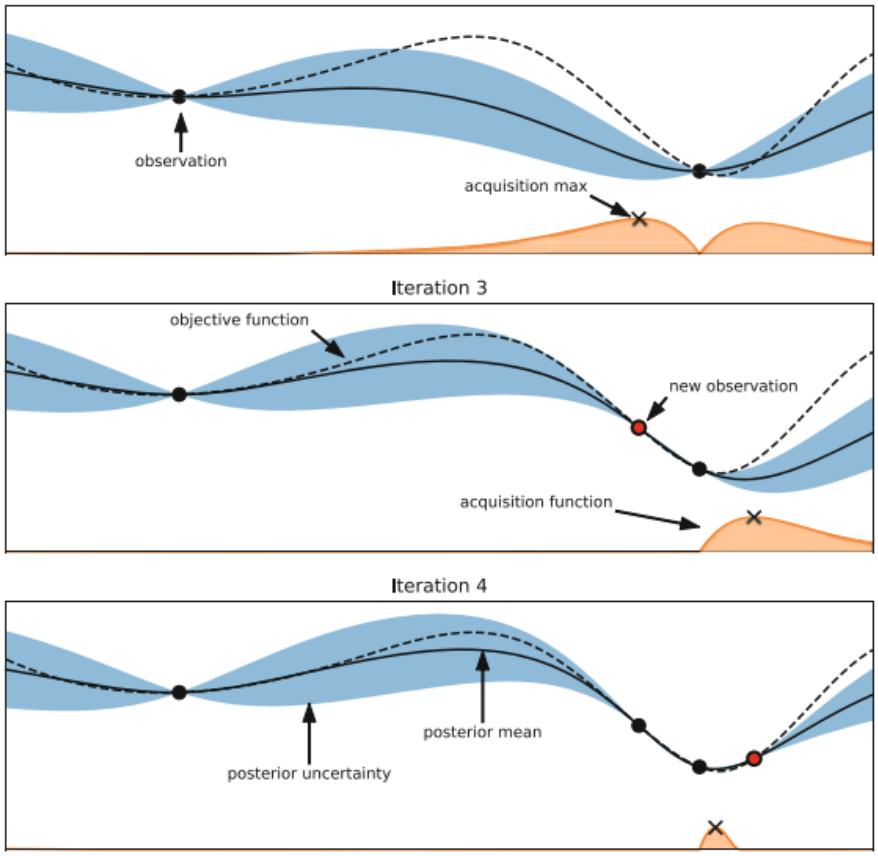

Figure 3.1 Illustration of Bayesian Optimization on a 1*D* function. The graph depicts a Gaussian process approximation of the objective function and an acquisition function in the lower portion. The mean of the objective function is represented by the solid line and the dashed line represents the actual objective function. The blue region shows the predictive uncertainty. The acquisition function is represented by the orange curve. The acquisition function is maximum where the Gaussian process gives a low objective value with high uncertainty. *Note.* Adapted from *Automated Machine Learning*, by Feurer, M. and Hutter, F., 2018, Springer, p. 10.

3.2.2 Genetic Algorithm

Genetic Algorithm or GA is a search strategy rooted in Charles Darwin's concept of natural selection. Initialising the population, Calculating the fitness function, Selection, Crossover and Mutation are the five phases in GA. The process begins with a group of people known as the population. Each individual is denoted by a finite-length vector of components, similar to a chromosome and represents a solution in search space for a given problem. Genes are equivalent to these variable components. Thus, each chromosome (individual) is made up of multiple genes (variable components). The fitness function determines an individual's ability to compete against others. It specifies each individual a score. This fitness score determines the likelihood of an individual being

selected for the next generation.

The objective of the selection phase is to identify the fittest individuals and enable them to carry on their traits to the coming generation. This is done by calculating the fitness score for each individual and selecting the highest scored pair. Physically fit individuals have a higher possibility of being selected for generating offspring. The most crucial part of the genetic algorithm is the crossover phase. A crossover site within the DNA is picked at random for each pair of parents to be mated. The genes at this point are then swapped, resulting in the creation of a totally new individual called the offspring. These new children are then included in the population. Some genes in new offspring are vulnerable to mutation, meaning that some of the bits in the DNA can be swapped. The benefits of mutation include maintaining population diversity and avoiding premature population convergence. If the population has converged, it will not generate children who are distinct from the previous generation, causing the algorithm to cease.

Algorithm 2: Pseudo-code for BO-GA

Input: Objective function F, Gaussian process model G, acquisition function E, hyperparameter configuration λ, maximum number of iterations T, initial number of solutions in the population n, crossover probability p_c, mutation probability p_m, number of iterations for GA k

Output: Best hyperparameter configuration λ_T

Initialize data D_0 with initial observations
for $t \leftarrow 1$ **to** T **do**
    Fit the Gaussian process model G_t on D_{t-1}
    // Select next query point: λ_t that maximise $E(\lambda; D_{t-1}, G_t)$
    Generate n feasible solutions randomly as $P_0 = \{\lambda_1, \lambda_2, ..., \lambda_n\}$
    Evaluate each solution in P_0 using E
    for $i \leftarrow 1$ **to** k **do**
        Select the best solutions that maximizes E from P_{i-1}
        Perform crossover on the selected parents based on p_c and generate offspring
        Perform mutation on this offspring based on p_m
        Evaluate the offspring on E
        Add them to next generation, P_i
    Save P_k as λ_t
    Query $F(\lambda_t)$
    Update data: $D_t \leftarrow D_{t-1} \cup \{< \lambda_t, F(\lambda_t) >\}$

3.2.3 Differential Evolution

Differential Evolution or DE is an evolutionary algorithm proposed by Storn and Price (1997). DE makes minimal, if any, assumptions about the underlying optimization issue and can rapidly explore enormous design spaces. Unlike standard evolutionary algorithms, it can deal with multi-dimensional real-parameter optimization problems.

DE is also a population-based stochastic approach. Like other evolutionary algorithms, genome/chromosome is the term given to each solution. Each chromosome goes through mutation followed by recombination. Unlike the genetic algorithm that represents candidate solutions using sequences of bits, DE keeps a population of candidate solutions in the form of real-valued vectors, also called target vectors. The target vector is made of a certain number of decision variables. In each loop of the algorithm, a new donor vector is generated after mutation and a trial vector is formed out of recombination. Once the trial vectors have been generated, the best solution is selected by a greedy approach conducted among all target and trial vectors.

During mutation, DE creates new vectors called donor vectors by multiplying a third vector to the weighted difference between two population vectors. Donor vector (V) of a chromosome λ^i is formulated as:

$$V = \lambda_{x_1} + (\lambda_{x_2} - \lambda_{x_3})\, s, \tag{3.4}$$

where s is the scaling factor (0, 1 or 2), x_1, x_2, x_3 are random solutions $\in \{1, 2, 3, ..., n\}$ and $x_1 \neq x_2 \neq x_3 \neq i$. Four vectors are involved in the generation of the target vector via mutation. Therefore the size of the population n should be greater than or equal to 4. The target vector is not involved in mutation.

Recombination is performed by binomial crossover. This step increases the diversity in the population. The trial vector is created out of recombination as follows:

$$u^j = \begin{cases} v^j & \text{if } r \leq p_c \text{ OR } j = \delta \\ \lambda^j & \text{if } r > p_c \text{ AND } j \neq \delta, \end{cases} \tag{3.5}$$

where p_c is the crossover probability, r is a random number (0 or 1), δ is a randomly selected variable location. $\delta \in \{1, 2, ...d\}$ where d is the number of decision variables in target vector. In the expressions, u^j represents the j-th variable of the trial vector, v^j denotes the j-th variable of the donor vector, and λ^j corresponds to the j-th variable of

the target vector. δ assures that at least one variable from the donor vector is selected. The value of p_c is generally set high, indicating more crossover, i.e., more variables from the donor.

During selection, a fitness function is evaluated for each offspring. The population is updated using greedy selection. If trial vector gives a better fitness score, then they are selected to the next generation. Else target vectors are added to the next generation. The selection procedure is carried out only after all solutions generate offspring.

Algorithm 3: Pseudo-code for BO-DE

Input: Objective function F, Gaussian process model G, acquisition function E, hyperparameter configuration λ, maximum number of iterations T, initial number of solutions in the population n, crossover probability p_c, number of decision variables in target vector D, number of iterations for DE k

Output: Best hyperparameter configuration λ_T

Initialize data D_0 with initial observations
for $t \leftarrow 1$ **to** T **do**
    Fit the Gaussian model G_t on D_{t-1}
    // Select next query point: λ_t that maximise $E(\lambda; D_{t-1}, G_t)$
    Generate initial population of target vectors, $P_0 = \{\lambda_1, \lambda_2, ..., \lambda_n\}$
    for $i \leftarrow 1$ **to** k **do**
        Generate random numbers $x_1, x_2, x_3 \in \{1, 2, ..., n\}$ with $x_1 \neq x_2 \neq x_3 \neq i$
        Create donor vector (V_i) using $\lambda_{x_1}, \lambda_{x_2}, \lambda_{x_3}$ and scaling factor s
        Create trial vector (U_i) using randomly generated $\delta \in 1, 2, .., d$ and r as 0 or 1 based on p_c
        Evaluate the target and trial vectors on E and add the best vector (λ_i or U_i) to P^i
    Save P^k as λ^t
    Query $F(\lambda_t)$
    Update data: $D_t \leftarrow D_{t-1} \cup \{< \lambda_t, F(\lambda_t) >\}$

3.2.4 Evolutionary Strategy

Evolutionary Strategy or ES employs evolutionary principles, focusing on mutation and selection, to iteratively improve solutions for optimization problems. It does not use crossover like other evolutionary algorithms, instead, candidate solution modification is limited to mutation operators. The algorithm uses a population of potential solutions that are created at random. Each iteration of the method begins with an evaluation of

the population of solutions, followed by truncation selection, which entails removing all but a subset of the best solutions. The remaining solutions (parents) are each utilised to generate a set of new candidate solutions (mutation) that replace or compete with the parents for a place.

This approach has several variations. The number of parents chosen per iteration is measured in p and the number of offspring is o. One of the variations is represented as (p, o)-ES, a version where children take the place of parents. Another variation is represented as (p + o)-ES where children and parents are added to the population.

CMA-ES is the most powerful variation of ES (Hansen and Ostermeier, 2001). For an initial population $P = \{\lambda_1, \lambda_2, ...\lambda_n\}$ of size n, the major differences between CMA-ES and classic ES are:

1. Offsprings are not generated by mutation of a single individual but by the weighted mean of the current population:

$$\lambda_j = \mu + \sigma y_j, \tag{3.6}$$

where σ (> 0) is step-size, y is a random vector for $j = 1, 2, ..., n$ and

$$\mu = \frac{1}{\sum_{j=1}^{m} w_j} \sum_{j=1}^{m} w_j \lambda_j, \quad w_j > 0. \tag{3.7}$$

μ gives the weighted mean, w_j is the set of positive weight coefficients for recombination and λ_j is the set of m best individuals out of the current population.

2. Unlike standard ES, y is not chosen to be independent of j, but chosen such that:

$$y \sim N(0, C). \tag{3.8}$$

3. C undergoes updates at each iteration $(t + 1)$ using a rank-1 update:

$$C^{(t+1)} = (1 - \alpha)C^{(t)} + \alpha QQ^T, \tag{3.9}$$

where Q is the cumulative path parameter, initialised as 0 and updated using the equation:

$$Q^{(t+1)} = (1 - \beta)Q^{(t)} + \beta\sqrt{\beta(2 - \beta)m}\Big(\frac{\mu^{(t+1)} - \mu^{(t)}}{\sigma^{(t)}}\Big). \tag{3.10}$$

α, β are learning rates $\in (0, 1]$.

4. Step size control is integrated into CMA-ES. That is, σ in equation 3.6 is chosen in a way that prevents the population from converging prematurely.

The major advantage of CMA-ES over other evolutionary algorithms is that the population size can be freely chosen. There is no inherent need to use large population sizes.

Algorithm 4: Pseudo-code for BO-ES

Input: Objective function F, Gaussian process model G, acquisition function E, hyperparameter configuration λ, maximum number of iterations T, initial number of solutions in the population n, initial mean vector $\mu^{(0)}$, initial step size $\sigma^{(0)}$, initial covariance matrix $C^{(0)} = I$, initial number of solutions in population n, number of parents chosen per iteration m, number of iterations for ES k

Output: Best hyperparameter configuration λ_T

Initialize data D_0 with initial observations
for $t \leftarrow 1$ **to** T **do**
    Fit the Gaussian process model G_t on D_{t-1}
    `// Select next query point:` λ_t `that maximise` $E(\lambda; D_{t-1}, G_t)$
    Sample initial population of size n, $P_0 = \{\lambda_1, \lambda_2, ..., \lambda_n\}$
    for $i \leftarrow 1$ **to** k **do**
        Create new solutions from P_{i-1} using the equation, $\lambda_j^{(i)} = \mu^{(i-1)} + \sigma^{(i-1)} y_j^{(i-1)}$, where $y_j \sim N(0, C^{(i-1)})$
        Evaluate these solutions and select the m top samples that give maximum values for E
        $P_i = P_{i-1} + \lambda_j^{(i)}$, where $j \in \{1, 2, .., m\}$
        Update mean : $\mu^{(i)} = \mu^{(i-1)} + \sigma^{(i-1)} \bar{y}$
        Update step size, $\sigma^{(i)}$
        Update covariance matrix, $C^{(i)}$
    Save P_k as λ_t
    Query $F(\lambda_t)$
    Update data: $D_t \leftarrow D_{t-1} \cup \{< \lambda_t, F(\lambda_t) >\}$

The computational complexity of BO employing a Gaussian process surrogate model is $O(n^3)$, where n is the number of hyperparameter values (Yang and Shami, 2020). GA has an asymptotic run time of $O(n^2)$. For DE and CMA-ES, it is $O(n^3)$ (N. Knight and Lunacek, 2007). The computational complexity of BO-GA, BO-DE and BO-ES are $O(n^3)$.

3.3 RESULTS AND DISCUSSION

Three sets of experiments were carried out, all of which uses the same sets of data and are run on the same configuration, which is an Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz with 1 NVIDIA Tesla V100 GPU. The datasets used include CIFAR-10, MNIST, SVHN and CALTECH-101.

CIFAR-10 comprises 6000 images belonging to 10 different classes. It includes images of aeroplanes, birds, frogs, cats, deer, horses, cars, ships, dogs and trucks. MNIST consists of 60,000 grayscale square images (28×28 pixels) of handwritten single digits from 0 to 9. Street View House Number dataset contains 6,00,000 RGB images (32×32 pixels) of printed digits from 0 to 9 clipped from photographs of house number plates. CALTECH dataset has 101 object categories, with about 40 to 800 images belonging to each category (300 x 200 pixels).

Setup 1:

Experiment setup 1 aims to analyse four HPO methods - Random Search (RS), BO, Hyperband and GA with the help of AutoML systems. AutoML systems with different optimizers are evaluated based on their performance in image classification problems. AutoKeras and TPOT are used in this experiment.

AutoKeras is an open-source software library for AutoML. It is built on the foundation of the Keras deep learning library and can be employed to automatically search for the best model for a given dataset. AutoKeras includes a preprocessing step that automatically detects the type of problem and the format of the given data and then selects the appropriate ML models to try. It also uses a technique called NAS to search for the optimal neural network architecture for the input data. It starts by generating a large number of randomly initialized neural network architectures and then trains and evaluates each one on the dataset. The best-performing architectures are then selected and used to generate new architectures through mutations and crossovers. This process is repeated until a satisfactory architecture is found or a pre-defined stopping criterion is met. AutoKeras also supports transfer learning, which allows the user to fine-tune a model that has been pre-trained on a particular task or dataset. This can notably ac-

celerate the training process and enhance performance, especially on smaller datasets. AutoKeras includes a feature for HPO. AutoKeras employs random search by default, and it additionally offers support for other HPO methods such as Bayesian optimization and Hyperband. Users have the flexibility to specify their preferred optimization method. Finally, it outputs the best architecture or machine learning model along with the most suitable set of hyperparameters for a given data.

TPOT (Tree-based Pipeline Optimization Tool) is another open-source software library for AutoML, built on top of the scikit-learn library. It can be employed to automatically search for the best pipeline of preprocessing steps and ML models for a given dataset. TPOT uses genetic programming to search for the best pipeline. It starts by generating a population of randomly initialized pipelines and then trains and evaluates each one on the dataset. The best-performing pipelines are then selected and used to generate new pipelines through mutations and crossovers. This process is repeated until a satisfactory pipeline is found or a pre-defined stopping criterion is met. TPOT includes a wide range of preprocessing steps, such as feature scaling and feature selection, as well as ML models, such as linear regression, decision trees, and neural networks. The HPO in TPOT is performed using GA. Last, it returns the best ML model for the specified dataset with the best combination of hyperparameters.

The comparison of test error rates and wallclock time for experiment setup 1 are shown in Figure 3.2 and Figure 3.3. Figure 3.2 represents a line graph showing the error rates obtained for the classification of CIFAR, MNIST, SVHN and CALTECH data using BO, RS, Hyperband and GA for hyperparameter optimization. Overall, random search gives better results than the other three HPO techniques in terms of test error rate for all four datasets.

The green line in the graph represents genetic algorithm. It is observed that genetic algorithm has the highest error rates implying low classification accuracy for three of the datasets. The blue line depicting the performance of bayesian optimization shows that it produces a better outcome compared to GA in all four cases. Hyperband performs better than GA in most cases, but for CALTECH data, it gives an error rate of 0.7, indicating a very low accuracy. Random search, presented by the red line, has the lowest error rates

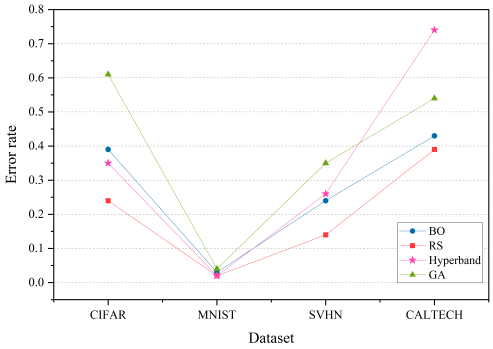

Figure 3.2 Test set classification error plot of AutoKeras and TPOT. The figure shows the average test error rate across the datasets - CIFAR, MNIST, SVHN and CALTECH. The HPO techniques compared in the graph are BO, RS, Hyperband and GA. AutoKeras is run 3 times - using BO, RS and Hyperband for HPO for each dataset. TPOT is run once for each dataset and it uses GA for HPO.

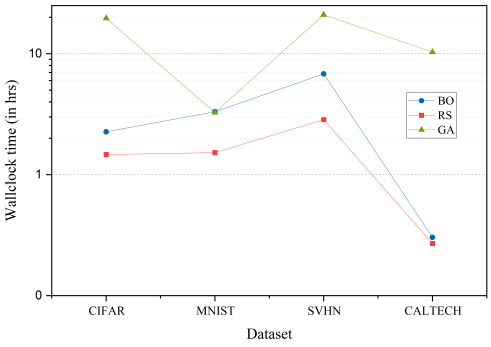

Figure 3.3 Wallclock time comparison plot of AutoKeras and TPOT across the datasets - CIFAR, MNIST, SVHN and CALTECH. The HPO techniques compared in the graph are BO, RS, Hyperband and GA. AutoKeras is run 3 times - using BO, RS and Hyperband for HPO for each dataset. TPOT is run once for each dataset and it uses GA for HPO.

for all classification problems.

Bayesian optimization is not the optimal optimizer for the image classification datasets used here. Methods were explored to enhance Bayesian optimization, including the utilization of other optimization algorithms for acquisition function maximization. Emphasis was placed on Evolutionary Algorithms for this objective. Given the already observed performance of GA, an assessment will be made of the performance of Differential Evolution (DE) with Bayesian optimization in setup 2. Finally, in setup 3, an examination will be conducted to determine if the combinations could enhance the efficiency of conventional Bayesian optimization.

Figure 3.3 shows datasets used on the x-axis and the time taken by AutoKeras (with BO, RS and Hyperband used for HPO) and TPOT (uses GA for HPO) on the y-axis. The graph is a semi-logarithmic plot that uses a logarithmic scale on the y-axis to indicate the variation in wallclock time from seconds to hours.

It is observed from the graph that TPOT requires hours-to-days to complete the image classification tasks. Random search takes rather less time compared to BO for all datasets. Hyperband is not included in the comparison because it uses early stopping criteria and requires very less time in contrast to other optimization techniques. TPOT

is recommended only in the case where time is not considered a constraint.

Setup 2:

To focus on BO and look for possibilities to improve its performance, a second experiment was conducted to compare optimizer packages that use BO and EA. Considering that the first experiment involved GA, a package utilizing a different evolutionary algorithm was chosen. Consequently, DEHB, which utilizes differential evolution, was selected, and its performance was analyzed in comparison to BO. The datasets used for this analysis are also the same. These packages are employed to optimize the hyperparameters of AlexNet, a deep learning architecture comprising 8 layers. It consists of 5 convolution layers and a mix of max-pooling layers. There are three layers that are fully connected. AlexNet introduced the use of non-saturating ReLU function as activation function, which resulted in improved training accuracy over tanh function and sigmoid. Two dropout layers are used. The final layer is a softmax activation function.

AlexNet is a fast GPU execution of a convolutional neural network. AlexNet won the ImageNet Large Scale Visual Recognition Challenge in 2012. The main conclusion of the original research on AlexNet was that the model's depth was critical for its high performance, which was computationally costly but made possible by the use of GPUs during training (Krizhevsky et al., 2012).

Sl. No	Hyperparameter	Range
1	Learning rate	[0.0001 - 1.0]
2	Batch size	[32 - 1024]
3	Epochs	[10 - 100]
4	Momentum	[0.9 - 0.999]

Table 3.1 List of hyperparameter values of AlexNet and DenseNet models used for tuning.

The experiment considers an AlexNet model and 4 of its hyperparameters. The hyperparameters and their respective value ranges for tuning are provided in Table 3.1. The optimizers used in the experiment are SMAC and DEHB. The main core of SMAC combines Bayesian optimization with a mechanism to quickly determine which of the two configurations is the better performer. RF is used as the surrogate model. On the

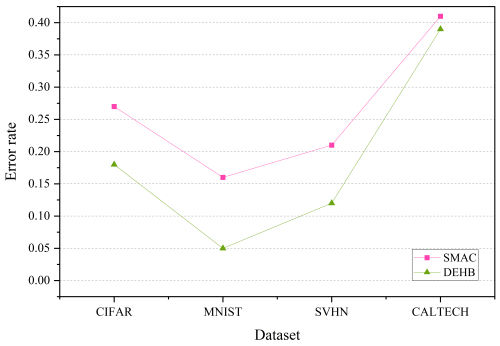

Figure 3.4 Test set classification error plot of an AlexNet model that uses the HPO packages - SMAC and DEHB. The figure represents the average test error rate across the datasets - CIFAR, MNIST, SVHN and CALTECH (based on 20 iterations).

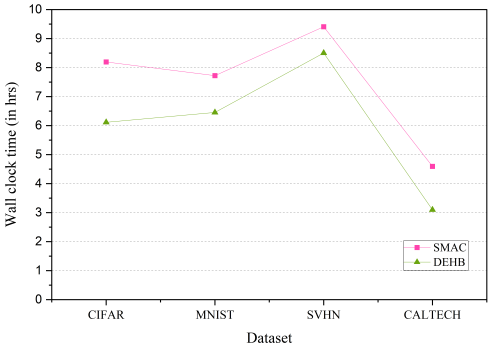

Figure 3.5 Wallclock time comparison plot of AlexNet model across the datasets - CIFAR, MNIST, SVHN and CALTECH. The HPO packages used here are SMAC and DEHB.

other hand, DEHB combines the advantages of the popular bandit-based HPO method Hyperband and the evolutionary search approach of Differential Evolution. Figure 3.4 illustrates the comparison of the two HPO packages based on error rates obtained using the Wilcoxon Rank-Sum test on samples from 20 iterations with a statistical significance of p-value < 0.05. From Figure 3.4, it is clearly noticeable that the green line representing DEHB has lower error rates for all four datasets compared to the pink depicting SMAC. DEHB with the advantages of both Hyperband and Differential Evolution proves to be more promising than SMAC. This outcome gave us the notion that combining DE and BO might improve the performance of BO.

Figure 3.5 shows the wall clock time comparison of the AlexNet model on CIFAR, MNIST, SVHN and CALTECH. The model uses SMAC and DEHB for HPO. The x-axis represents the datasets and y-axis represents the time taken by SMAC and DEHB. It is observed from the graph that DEHB takes lesser time compared to SMAC for all four classification problems.

Setup 3:

Setup 3 attempts to compare various combinations of Bayesian optimization with evolutionary algorithm models with traditional BO, random search and genetic algorithm. The HPO models compared here are RS, GA, BO, BO-DE, BO-CMAES and BO-GA

in terms of error rate and time efficiency. The test considers an AlexNet model and four of its hyperparameters for optimization.

The experiment is executed by implementing the BO variants on the same set of data. Three HPO models that employ DE, GA and CMA-ES for acquisition function maximization are compared against traditional BO, RS and GA. The results are analysed comparing the error rates obtained using the Friedman test on samples from 20 iterations with a statistical significance of p-value < 0.05 and is shown in Figure 3.6.

All variants of BO used here are compared to the standard Bayesian optimization. Conventional BO usually uses random optimizers for maximizing EI. A combination of random sampling (cheap) and the 'L-BFGS-B' (Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm with bound constraints) optimization method is used this case. L-BFGS-B is a variant of L-BFGS that uses a limited amount of computer memory to approximate the L-BFGS algorithm. It is a second-order quasi-Netwon method and has a faster convergence rate.

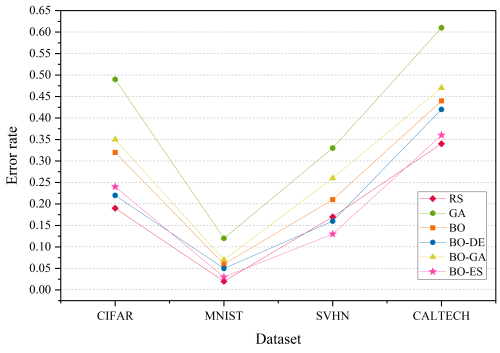

Figure 3.6 Test set classification error plot of an AlexNet model that uses RS, GA, a BO framework for HPO with L-BFGS, BO-DE, BO-GA and BO-ES. The figure depicts the average test error rate across the datasets - CIFAR, MNIST, SVHN and CALTECH (based on 20 iterations). The results are obtained using the Friedman test with a statistical significance of p$<$ 0.05.

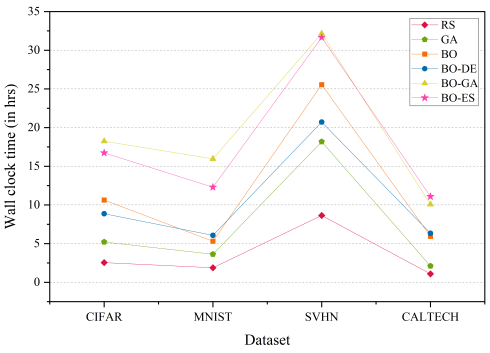

Figure 3.7 Wallclock time comparison plot of AlexNet model across the datasets - CIFAR, MNIST, SVHN and CALTECH. The model uses RS, GA, a BO framework for HPO with L-BFGS, BO-DE, BO-GA and BO-ES.

Figure 3.6 shows that the overall efficiency of BO combined with CMA-ES has better classification accuracy compared to combinations of BO with other evolutionary algorithms in most cases. BO combined with DE produces better results than GA.

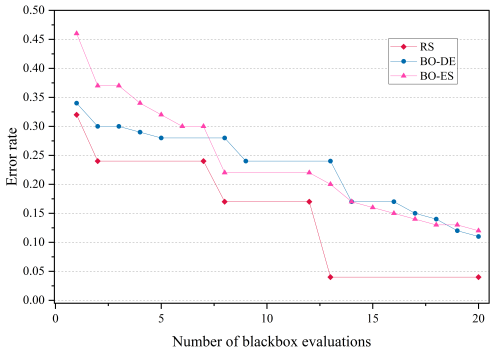

Figure 3.8 Comparison of how error rates of AlexNet evolve with the number of function evaluations for RS, BO-DE and BO-ES on CIFAR data.

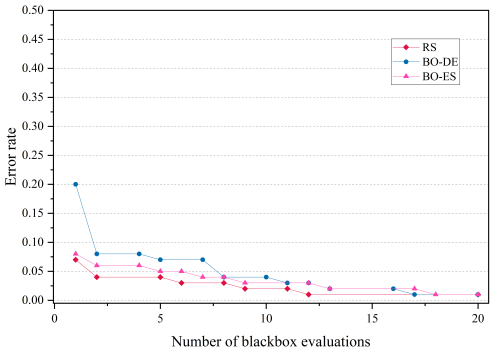

Figure 3.9 Comparison of the error rates of AlexNet against the number of function evaluations for BO-DE, BO-ES, and RS using MNIST data.

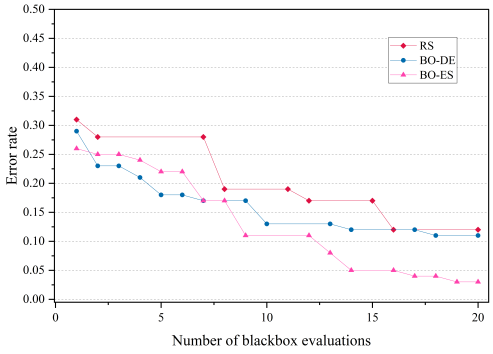

Figure 3.10 Comparison of the error rates of AlexNet against the number of function evaluations for BO-DE, BO-ES, and RS using SVHN data.

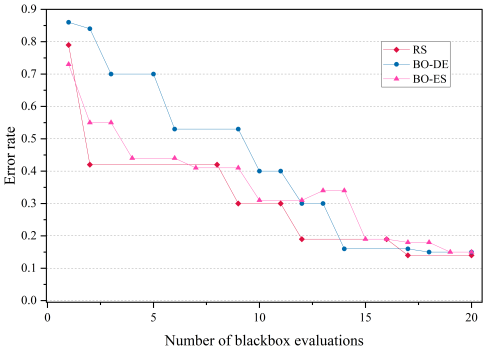

Figure 3.11 Comparison of how error rates of AlexNet evolve with the number of function evaluations for RS, BO-DE and BO-ES on CALTECH data.

The graph also proves that the performance of standard BO can be improved by using DE and CMA-ES for acquisition function maximization. RS gives the best accuracy for most datasets. This is because RS performs well in small hyperparameter spaces. Since the experiment considered only 4 hyperparameters, which forms a low-dimension search space, RS delivers good performance. But this cannot be guaranteed for higher dimensional hyperparameter search spaces. The results are obtained using the Friedman test with a statistical significance of $p < 0.01$ and plotted based on average test error rates from 20 iterations.

Figure 3.7 shows the wall clock time comparison of the AlexNet model on CIFAR, MNIST, SVHN and CALTECH. The model uses a BO framework for HPO with

L-BFGS, DE, GA and ES for acquisition function optimization. BO-DE and BO perform better than BO-ES and BO-GA in terms of run-time. RS and GA took less time compared to other HPO models. RS may not be time-efficient in high-dimensional hyperparameter search space. In a high-dimensional space, the chance of finding the optimal hyperparameters by chance becomes very small, and the search process can become extremely inefficient and time-consuming. In such cases, RS may require a large number of trials to cover the entire search space, which can be computationally expensive and time-consuming. This is because RS generates random combinations of hyperparameters without considering any prior information or structure in the search space, leading to a lot of redundant trials and a low probability of finding the optimal hyperparameters. GA performs poorly in all cases.

The error rate versus number of blackbox evaluation graphs are given in Figure 3.8, Figure 3.9, Figure 3.10 and Figure 3.11 corresponding to CIFAR, MNIST, SVHN and CALTECH respectively. The comparison focused on the three best-performing models: RS, BO-DE and BO-ES. Only successful error rate reductions are depicted in the graph. The downward trend in the graphs mean that over time, more promising regions are being examined more frequently. A flat trend indicates very little learning from prior experiences. BO-DE and BO-ES finds more number of better configuration than RS in all figures.

Setup 4:

Similar to setup 3, setup 4 compares RS, GA, BO, BO-DE, BO-CMAES and BO-GA. However the experiment is carried out on a Densenet model considering four of its hyperparameters. The list and range of the hyperparameters are in Table 3.1.

DenseNet-121 is a convolutional neural network (CNN) architecture for image classification. It is comprised of the following components: 1) Initial Convolution: The network starts with a standard convolutional layer to extract low-level features. 2) Dense Blocks: The core of DenseNet121 is made up of dense blocks, which contain multiple layers with different numbers of filters and every layer is connected to all the preceding layers. This enables the network to learn more robust and diverse features, and improve feature reuse. 3) Transition Layers: Between each dense block, a transition layer is

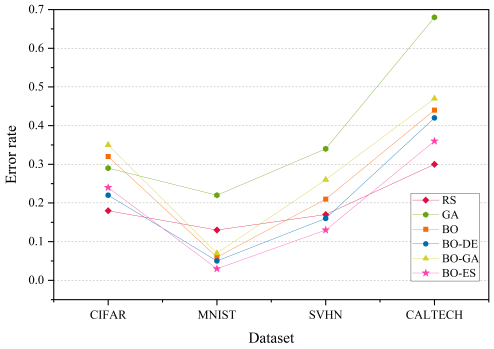

Figure 3.12 Test set classification error plot of DenseNet model that uses RS, GA, a BO framework for HPO with L-BFGS, BO-DE, BO-GA and BO-ES. The figure depicts the average test error rate across the datasets - CIFAR, MNIST, SVHN and CALTECH (based on 20 iterations). The results are obtained using the Friedman test with a statistical significance of $p < 0.01$.

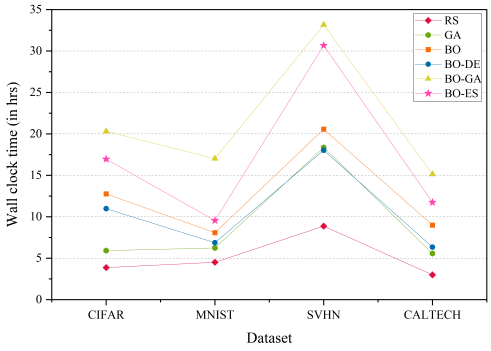

Figure 3.13 Wallclock time comparison plot of DenseNet model across the datasets - CIFAR, MNIST, SVHN and CALTECH. The model uses RS, GA, a BO framework for HPO with L-BFGS, BO-DE, BO-GA and BO-ES.

employed to decrease the number of filters, and also to perform dimensionality reduction by using pooling to reduce the spatial dimensions. 4) Final Classification: The network ends with a standard fully connected layer and a softmax activation function to produce the final class probabilities. In total, DenseNet-121 has 120 convolution layers, 4 average pooling layers and 1 fully connected layer. In this architecture, features from previous layers are concatenated with the features of the current layer, leading to increased feature reuse and reduced risk of overfitting. The architecture is well suited for tasks involving large amounts of image data and has achieved state-of-the-art results on benchmarks such as ImageNet (Huang et al., 2017).

Figure 3.12 illustrates the results in terms of the error rate determined by the Friedman test on samples from 20 iterations with a statistical significance of p-value < 0.01. RS gives the best accuracies for CIFAR and CALTECH data while BO-ES gives better results for MNIST and SVHN datasets. As for AlexNet, BO-ES and BO-DE perform better than standard BO. GA and BO-GA deliver the lowest accuracies for most datasets.

The wall clock time comparison of the DenseNet model on CIFAR, MNIST, SVHN, and CALTECH is presented in Figure 3.13. RS takes the least time for all datasets. As

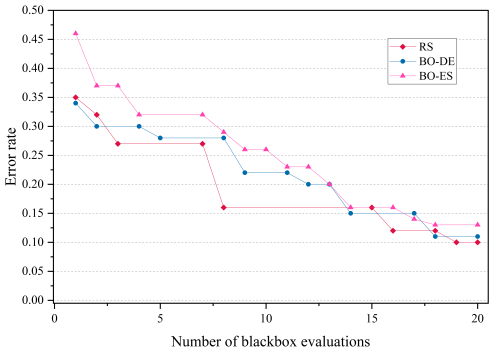

Figure 3.14 Comparison of how error rates of DenseNet evolve with the number of function evaluations for RS, BO-DE and BO-ES on CIFAR data.

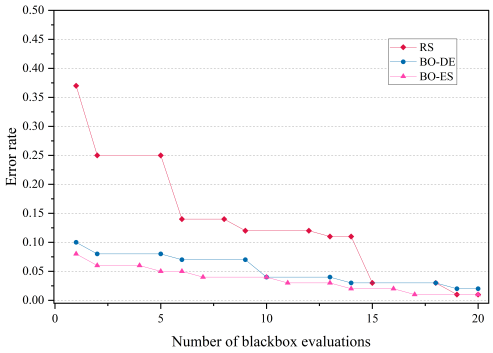

Figure 3.15 Comparison of the error rates of DenseNet against the number of function evaluations for BO-DE, BO-ES, and RS using MNIST data.

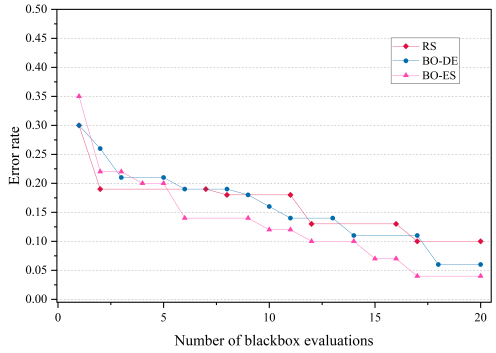

Figure 3.16 Comparison of the error rates of DenseNet against the number of function evaluations for BO-DE, BO-ES, and RS using SVHN data.

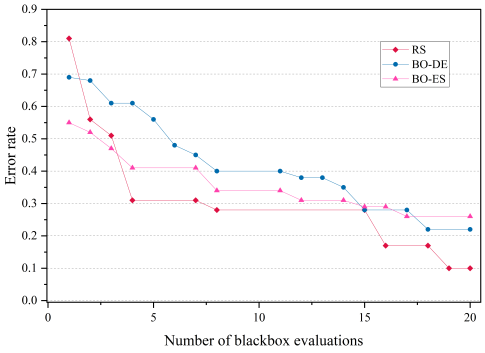

Figure 3.17 Comparison of how error rates of DenseNet evolve with the number of function evaluations for RS, BO-DE and BO-ES on CALTECH data.

mentioned earlier, the time efficiency of RS depends on the dimensionality of hyperparameter space. BO-GA and BO-ES take more time than standard BO and BO-DE.

Figure 3.14, Figure 3.15, Figure 3.16 and Figure 3.17 depicts the development of optimization techniques on CIFAR, MNIST, SVHN and CALTECH datasets. The three models RS, BO-DE, and BO-ES that perform the best have been compared. The graph only shows successful error rate reductions. The steady decline in graphs indicates that more promising configurations are being looked at more often over time. In all figures, BO-DE and BO-ES discover more instances of better configuration than RS.

3.4 CLOSING REMARKS

Hyperparameters of ML models must be tuned to fit particular datasets before being deployed to practical problems. However, the volume of created data has substantially grown in practice. And manually setting all hyperparameters is tremendously resource-intensive. Therefore, it has become critical to optimise hyperparameters automatically. The efficacy of AutoML systems in managing large-scale image classification datasets has been demonstrated. Additionally, the efficiency of the SMAC and DEHB packages for hyperparameter optimization has been observed.

In particular, different HPO models on AlexNet and DenseNet architectures were compared. Out of the HPO frameworks studied, it is examined that the efficiency of BO-CMAES and BO-DE proves it to be worth adopting in AutoML systems. The results from this experiments can be extended to other models and datasets. The number of hyperparameters can also be scaled along with the range of values considered for tuning. The above experiments are carried out for 20 iterations. For better results, the number of iterations can be increased. In the scope of future work, there is a plan to implement this algorithm comparison in different applications. Additionally, similar experiments are intended for real-time image classification problems.

Chapter 4

LAND USE LAND COVER MAPPING

4.1 INTRODUCTION

Numerous platforms, methods, and techniques are used in the classification process, including object-based image analysis, supervised and unsupervised classification, GIS, and ML algorithms like decision trees and neural networks (Macarringue et al., 2022; MohanRajan et al., 2020). A strong framework for LULC classification is offered by GIS, which enables the integration of various datasets and the visualisation of trends in land cover over enormous areas. LULC using ML involves using input data such as satellite imagery, terrain data, and other environmental variables to train a model to recognise and classify various land cover and land use patterns, such as forests, water bodies, urban areas, etc (Talukdar et al., 2020; Devi et al., 2021).

Neural networks, decision trees, random forests, SVM, ANN, and deep learning models are popular machine learning approaches approaches are widely popular for land use and land cover analysis (Jozdani et al., 2019). Every algorithm has pros and cons of its own, and the optimal method to choose will depend on the specific characteristics of the data and the given task. The initial stage of LULC categorization using machine learning is gathering representative and high-quality data, such as aerial photography or satellite imaging. Preprocessing this data is necessary to extract pertinent aspects from the images, such as spectral and textural information, and to address errors in the data, such as atmospheric effects or geometric distortions (Darem et al., 2023).

The preprocessed data can be utilized to train machine learning models. Subsequently, the data is partitioned into training and validation sets for the training phase,

followed by selecting an appropriate algorithm and its hyperparameters, and using the training set to teach the algorithm to recognize patterns in the data. Following that, the model's performance is assessed with the validation set and the chosen metrics, including kappa coefficient and overall accuracy. Machine learning-based LULC classification has a number of limitations and restrictions, including the requirement of diverse and high-quality training data, the choice of suitable features and methods, and the balance between interpretability and accuracy (Li et al., 2023b). On the other hand, LULC categorization is now more precise and efficient because of the advancements in machine learning algorithms and remote sensing technology, which have also created new dimensions for urban planning.

A map that recognises and classifies various land use and land cover elements, such as forests, croplands, urban areas, and water bodies, is the outcome of the LULC classification process. This map can be used to track changes over the time and guide the management decisions. Urban planners can find and preserve green places like parks, forests, and wetlands with the use of LULC classification. Planning for additional green spaces is beneficial in locations where there aren't many. The effects of urbanisation on the environment, such as a decline in biodiversity and a rise in air and water pollution, can be evaluated by planners. With this information, they may create plans for sustainable development that lessen the adverse effects of urbanisation.

4.2 MATERIAL AND METHODS

4.2.1 Study Area

Ernakulam is a district in the state of Kerala, located in the southwestern region of India (Figure 4.1). Also known as the commercial capital of Kerala. Ernakulam district is bounded by the Arabian Sea on the west, Thrissur district on the north, Idukki district on the east, and Alappuzha and Kottayam districts on the south. The district is 30.53 square kilometres in size. The topography is primarily flat, with a few mountainous areas in the east. The district is intersected by numerous water bodies, including the Periyar, Muvattupuzha, and Chalakudy rivers, along with various canals. Ernakulam has a tropical climate with intense monsoon rains from June to September. The district exhibits a heterogeneous land use and land cover pattern, encompassing a blend of rural

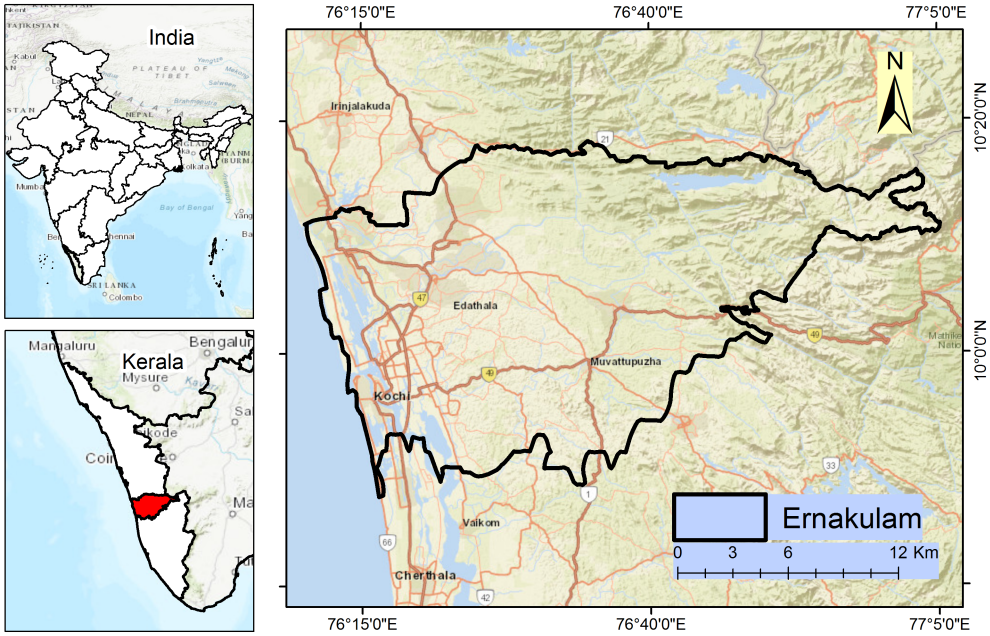

Figure 4.1 Location of study area for LULC classification.

and urban regions, forests, aquatic bodies, and wetlands. The district's built-up regions comprise various towns and villages in addition to the city of Kochi, Kerala's principal commercial and industrial centre. These areas are defined by high-rise buildings, extensive highways, and various critical infrastructure elements, including seaports, airports, and railway stations (Government of Kerala, 2023b).

4.2.2 Data Source

We use Sentinel-2 data, a satellite mission by Copernicus program that employs a multispectral optical imaging sensor (Gascon et al., 2014). They capture images in 13 spectral bands including visible and near infrared light. These images are of 10 meters spatial resolution with a revisit time of 5 days.

Sentinel-2 data can be accessed through google earth engine (GEE). Over recent years, GEE has gained considerable attention in the realm of remote sensing (RS) big data processing. GEE serves as an interactive development environment (IDE) that houses a vast multi-petabyte collection of geospatial data, which is integrated with a

high-performance, inherently parallel computation service. This platform facilitates swift data visualization and analysis, all of which can be controlled through an application program interface (API) accessible over the Internet. The data is then preprocessed, cloud removed, and mosaicked in GEE to optimize computational efficiency. The Earth Engine code editor is managed using client libraries available in JavaScript and Python (Gomes et al., 2020).

The training and testing data is generated as polygons in GEE with each class: Water, Barren land, Sparse vegetation, Dense vegetation and Built up, containing 75 samples. The rest of the processsing is carried out in Jupyter Notebook which is an interactive computing environment that supports multiple languages including Python. The training and testing data is then imported as shape files in Jupyter Notebook where it is used to train machine learning models. The details of machine learning models used in the study are discussed in the following paragraphs.

4.2.3 Machine Learning Models

We have used 4 ensemble models : random forest (RF), gradient boost, extreme gradient boost (XGBoost) and adaboost for LULC mapping. Ensemble models combine the advantages of several machine learning models to deliver a powerful model with the combined strengths of individual models. These discrete models may be neural networks, decision trees, random forests, gradient boosting, or other models. Ensemble methods, such as boosting and bagging, establish a cooperative framework in which every model shares its knowledge to produce predictions that are more reliable and accurate. This variety in model designs reduces overfitting and boosts generalization.

RF is made up of several decision trees and trained on various subsets of the data using bootstrapping. To lessen overfitting and improve prediction accuracy, RF selects feature randomly using bagging (Breiman, 2001).

Gradient Boosting combines the predictions of several weak learners, usually decision trees. Gradient boosting evaluates errors made by previous learners in each step and ranks them based on the errors, after which it fits a new learner to learn those errors. Iteratively, the process proceeds, with each new model concentrating on the remaining errors and progressively raising the total predictive accuracy (Friedman, 2001).

Hyperparameter	Range
Number of estimators	[100 - 1000]
Maximum depth	[10 - 50]
Maximum features	[2 - 20]
Maximum samples	[0.1 - 1.0]
Maximum leaf nodes	[20 - 200]
Minimum samples split	[100 - 1000]
Minimum samples leaf	[100 - 1000]

Table 4.1 List of hyperparameter values of random forest model used for tuning.

XGB is an extreme version of gradient descent boosting. However, there are variations in the specifics of the modelling. To improve performance, XGBoost specifically adopted a more regularised model formalisation to limit over-fitting and supports parallel processing (Chen and Guestrin, 2016).

AdaBoost is another boosting technique that employs an iterative approach. At first, it gives each training example the same weight. Initially the data is used to train weak learners. AdaBoost gives the data points that the prior weak learners incorrectly identified higher weights after each cycle. This concentrates the attention of the subsequent weak learners on the more difficult-to-classify cases. AdaBoost assigns weights to each weak learner based on their performance and combines their predictions. A weak learner's performance has a greater impact on the ultimate prediction if they do better after each iteration. The weighted forecasts of each weak learner are added together to produce the final prediction (Dou and Chen, 2017).

The hyperparameters of RF model listed in Table 4.1 are optimized using HPO algorithms mentioned in Chapter 3.

4.2.4 Change Detection

LULC change detection is a procedure that encompasses the analysis and comparison of distinct land use and land cover datasets to detect and measure alterations over time (Civco et al., 2002; Viana et al., 2019). It is an invaluable resource for comprehending landscape dynamics and evaluating the effects of environmental changes, natural processes, and human activity. Change detection algorithms are used to find locations where notable changes in land cover have taken place, once the land cover classes have

been allocated to each time period. Various techniques exist for change detection, such as post-classification comparison, image differencing and object-based algorithms.

In a post-classification comparison, two or more datasets from various time periods are independently classified, and the resulting land cover classifications are compared (Gómez et al., 2016). Pixel by pixel comparisons between the allocated land cover classes or their aggregated identification into broader geographical units (e.g., patches or regions) are used to identify changes. Although there may be classification discrepancies between the various datasets, this method can be useful in identifying changes.

In order to do image differencing, pixel values of images from various time periods are subtracted. This method can be applied to both spectral and multispectral images and is centred on identifying changes in pixel values. Changes are detected by establishing a threshold to separate meaningful differences from noise and other random variations. Image differencing will also require normalisation and calibration as it can be sensitive to radiometric changes between images.

In this work we have used a post classification comparison technique to identify the LULC changes in the study area between 2019 and 2023.

4.3 RESULTS AND DISCUSSION

4.3.1 Evaluation Metrics

Evaluating accuracy is a crucial step in assessing the results of a classification problem. It is imperative that the end user understands the accuracy of the generated output in order to use land-cover data efficiently. Accuracy, typically used in classification tasks, quantifies the percentage of correctly classified instances. It is calculated as the ratio of correctly categorized instances to the total number of instances within the dataset. The formula for accuracy is given as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{4.1}$$

where TP is true positives, TN is true negatives, FP is false positives and FN is false negatives. Figure 4.2 depicts a confusion matrix. The row represents the actual value and the column represents the predicted value.

The Kappa coefficient is a statistical metric used to assess the level of agreement

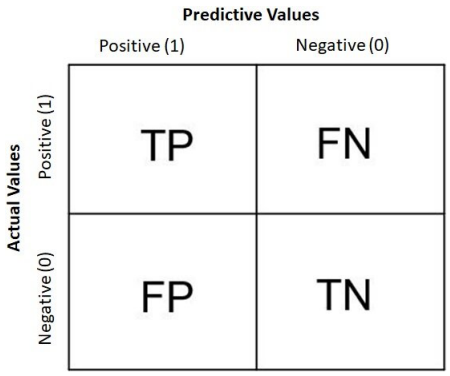

Figure 4.2 Illustration of the confusion matrix with true positives, true negatives, false positives and false negatives.

between two maps, one representing a classified dataset and the other a reference dataset (Verma et al., 2020). It quantifies the extent to which each classification differs from a random classification of class types. K is commonly employed to evaluate LULC classification. It is calculated as follows:

$$K = \frac{P_{obs} - P_{exp}}{1 - P_{exp}}, \tag{4.2}$$

where P_{obs} is the observed agreement, which is the same as overall accuracy. It is calculated as:

$$P_{obs} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{4.3}$$

P_{exp} is the expected agreement by chance or random accuracy. P_{exp} in terms of TP, TN, FP and FN is given by:

$$P_{exp} = \frac{((TP + FN)(TP + FP) + (FP + TN)(FN + TN))}{(TP + TN + FN + FP)^2}. \tag{4.4}$$

The K value falls within the range of -1 to 1, where a score of 1 signifies perfect agreement, 0 suggests no agreement beyond what would be expected by chance, and -1 implies complete disagreement. When there is no discernible correlation between the assessments made by two raters, K takes on a negative value. A value more than 0.75 indicated high agreement beyond chance, but a value less than 0.40 indicated poor agreement (Landis and Koch, 1977; Chopade et al., 2023).

4.3.2 Model Evaluation

Overall accuracy and Kappa values of all models are specified in Table 4.2. The RF model outperformed other ensemble models in terms of accuracy and Kappa score in our study. RF model combines multiple decision trees and reduces the variance of indi-

Model	HPO	2019		2023	
		Accuracy	Kappa	Accuracy	Kappa
Gradient Boost	-	86.61	0.84	84.73	0.81
XGBoost	-	87.25	0.85	86.25	0.84
AdaBoost	-	45.74	0.39	63.14	0.58
RF	-	**90.13**	**0.88**	**90.85**	**0.89**
RF	Random search	85.03	0.82	91.13	0.89
RF	BO	91.28	0.89	90.89	0.89
RF	BO-DE	90.45	0.88	90.86	0.89
RF	**BO-ES**	**92.00**	**0.9**	**92.80**	**0.91**

Table 4.2 Accuracy and Kappa score of ML models.

vidual trees. The model reduces overfitting and is more robust and generalizable compared to other ensemble models. Feature selection is inbuilt in RF model. When splitting features in the trees, it assesses each feature's significance, and determine which features are more relevant to the current task. The model is also resistant to noisy data and outliers. Individual data points that do not accurately reflect the general pattern in the data are less likely to have an impact.

Comparing the RF model's performance with and without HPO clearly shows the benefits of applying HPO. Without HPO, the RF model achieved an accuracy of 90.13% and a Kappa of 0.88 for the 2019 LULC map, and 90.85% accuracy and 0.89 Kappa for the 2023 map. However, with BO, the accuracy improved to 91.28% and 90.89%, and the Kappa to 0.89 in both years. The BO-ES method further enhanced performance to 92.00% accuracy and 0.90 Kappa for 2019, and 92.80% accuracy and 0.91 Kappa for 2023. This demonstrates that employing HPO, particularly BO and BO-ES, significantly enhances model accuracy and consistency.

For the 2019 LULC map, the RF model optimized with BO achieved an accuracy of 91.28% and a Kappa of 0.89, whereas the BO-DE method resulted in a slightly lower accuracy of 90.45% and a Kappa of 0.88. Similarly, for the 2023 LULC map, BO optimization maintained an accuracy of 90.89% and a Kappa of 0.89, while BO-DE produced an accuracy of 90.86% and a Kappa of 0.89. These comparisons indicate that BO generally performs marginally better than BO-DE, particularly in terms of accuracy for the 2019 LULC map. However, the performance difference between the two methods is relatively minor, with BO-DE showing comparable Kappa values in both years

and a very slight dip in accuracy.

The BO-ES method significantly outperformed all other HPO approaches, achieving the highest accuracy and Kappa values in both years. These results indicate that Bayesian Optimization when combined with evolutionary strategies (BO-ES), substantially enhances the performance obtained by BO.

A confusion matrix, also known as an error matrix, is a table of numerical values organized in rows and columns. It shows how many test units polygons, pixel clusters, or individual pixels are assigned to different classes and how closely these assignments match the actual ground-truth classifications.

Pixels that belong to a class but are incorrectly classified to other classes are referred to as omission errors, also called exclusion errors. Commission error or inclusion error represents pixels that are originally part of different classes but are misclassified into a specific class. The overall accuracy is defined as the ratio of by total number of correctly classified pixels by the total numbers of reference pixels. Overall accuracy does not reflect on how well individual classes are classified. To provide insight into the accuracy of each individual class two terms called producer accuracy and user accuracy were introduced. The values of which depended on the omission and commission accuracy. Producer's accuracy measures the probability of a specific pixel on the ground to be classified correctly. It is calculated by dividing the number of pixels accurately classified in a particular category by the total number of sample polygons (or pixels) taken for that category.

User's accuracy is a measure of the likelihood that a pixel labeled as a specific class on a map is indeed that class in reality. It is calculated by dividing the number of pixels correctly classified as a particular category by the total number of pixels classified as that category. The producer accuracy and user accuracy are not the same, as they assess different aspects of classification performance. Producer accuracy focuses on the probability that a feature on the ground is correctly classified, while user accuracy assesses the probability that a pixel on the map is accurately labeled (Lu and Weng, 2007; Congalton, 1991).

Table 4.3 and Table4.4 gives the producers and users accuracy for Sentinel-2 2019

Classified	Water	Barren land	Sparse vegetation	Dense Vegetation	Built up	Total (User)	User's Accuracy
Barren land	68	4	1	2	0	75	91%
Built up	7	66	0	2	0	75	88%
Dense vegetation	0	0	72	2	1	75	96%
Sparse vegetation	0	0	3	70	2	75	93%
Waterbody	0	0	1	5	69	75	92%
Total (Producer)	75	70	77	81	72	375	
Producer's Accuracy	91%	97%	92%	86%	96%		

Table 4.3 Accuracy assessment of land cover map generated for 2019 Sentinel-2 using RF model with BO-ES. Total Correct: 345 Total Samples: 375 Overall Accuracy: 92% Kappa Statistics: 0.9.

Classified	Water	Barren land	Sparse vegetation	Dense Vegetation	Built up	Total (User)	User's Accuracy
Barren land	67	2	1	3	2	75	89%
Built up	3	67	2	2	1	75	89%
Dense vegetation	0	0	70	3	2	75	93%
Sparse vegetation	0	0	2	73	0	75	97%
Waterbody	0	0	1	3	71	75	95%
Total (Producer)	70	69	76	84	76	375	
Producer's Accuracy	96%	97%	92%	87%	93%		

Table 4.4 Accuracy assessment of land cover map generated for 2023 Sentinel-2 using RF model with BO-ES. Total Correct: 348 Total Samples: 375 Overall Accuracy: 92.8% Kappa Statistics: 0.91.

and 2023 LULC classifications generated using RF model with BO-ES. Producer's accuracy is greater for built-up in both years. User's accuracy is highest for dense vegetation in the year 2019 and sparse vegetation in 2023.

4.3.3 LULC Maps

Figure 4.3 and Figure 4.4 depict the LULC maps for the years 2019 and 2023 produced using RF model with BO-ES, respectively. Table 4.5 provides data on the area in square kilometers and the percentage of area of each LULC class for both years. A noticeable trend observed from both the maps and the table is a reduction in barren land. Barren lands have undergone a transformation, giving way to vegetation and potential urban

Classes	2019 Sentinel		2023 Sentinel	
	Area (sq. km)	%	Area (sq. km)	%
Barren land	231.32	7.57	168.43	5.52
Built up	516.25	16.90	534.13	17.49
Dense vegetation	611.82	20.03	696.07	22.79
Sparse vegetation	1399.87	45.84	1475.80	48.33
Waterbody	294.61	9.65	179.44	5.88

Table 4.5 Area belonging to different land cover in 2019 and 2023.

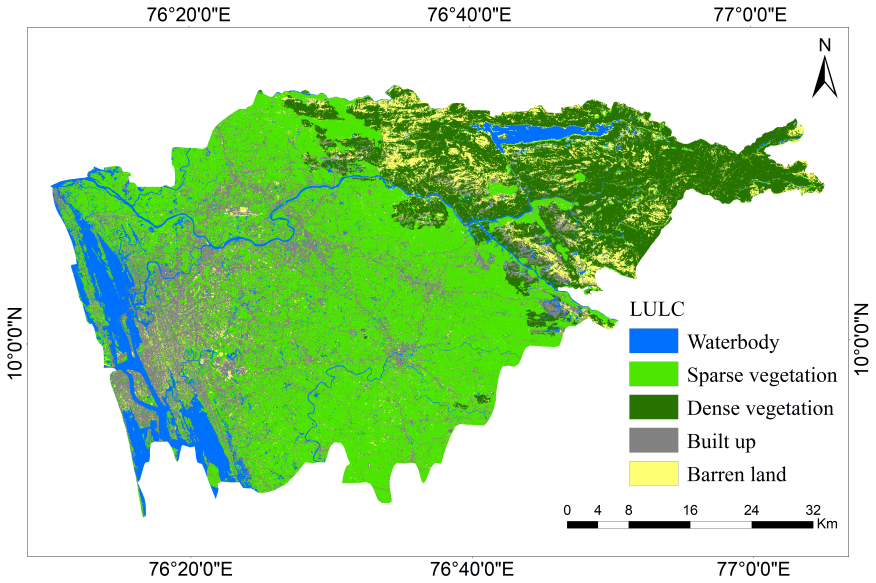

Figure 4.3 Land use land cover map generated from Sentinel-2 2019 satellite image using RF model with BO-ES.

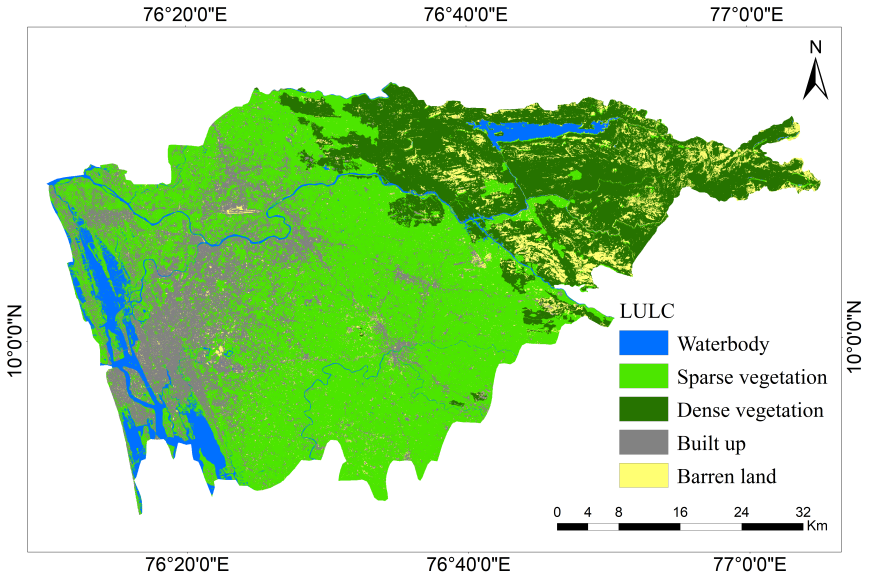

Figure 4.4 Land use land cover map generated from Sentinel-2 2023 satellite image using RF model with BO-ES.

development. Additionally, post-COVID-19 years have seen only a minor expansion in the built-up area, which is not entirely unexpected. Sparse vegetation and barren lands have made way for urban structures, particularly in flat terrains. Higher elevation regions remain unsuitable for urban development. The density of built-up areas is on the rise, especially as one moves from the city center towards the outskirts, notably along highways and other road networks.

4.4 CLOSING REMARKS

LULC classification using Sentinel-2 images from the years 2019 and 2023 were done. The LULC classes identified are waterbody, barren land, built up, sparse vegetation and dense vegetation. Ensemble machine learning models were employed for the purpose of classification. RF model gave the best performance in terms of accuracy and kappa-coefficient. The model was then aided with HPO to improve the classification. RF supported by BO-ES gave better results compared to other HPO techniques used. The lulc map obtained in both the years were compared and change in areas of different classes were analysed.

Chapter 5

FLOOD SUSCEPTIBILITY MAPPING

5.1 INTRODUCTION

Flooding is one of the prevalent natural hazard with highly adverse consequences. Understanding which areas are vulnerable to flooding is crucial to addressing these effects. Machine learning models and AutoML systems are used to generate flood susceptibility map of Kerala, India. Additionally a three-dimensional CNN architecture is used for this purpose. The CNN model was assisted with hyperparameter optimization techniques that combine Bayesian optimization with evolutionary algorithms like differential evolution and covariance matrix adaptation evolutionary strategies. All model performances are evaluated in terms of cross-entropy loss, accuracy, area under the curve (AUC) and kappa score.

5.2 MATERIAL AND METHODS

5.2.1 Study Area

Kerala (38,863 square kilometres) is bordered to the east by the Western Ghats and to the west by the Arabian Sea (Lakshadweep Sea) (Figure 5.1). Kerala has 44 rivers, 41 of which flow westward, while three flow eastward originating from the Western Ghats (Government of Kerala, 2023c). The rivers of Kerala are not large in terms of their width, length, or water flow. The rivers flow fast a direct outcome of the mountainous terrain and close proximity of the Western Ghats to the sea. All of the rivers receive their water from the monsoon, and most of them dry up in the summer. Kerala is home to 34 backwaters. The backwaters are dominated by Lake Vembanad, India's largest body of water, which lies between Alappuzha and Kochi and covers an area of more

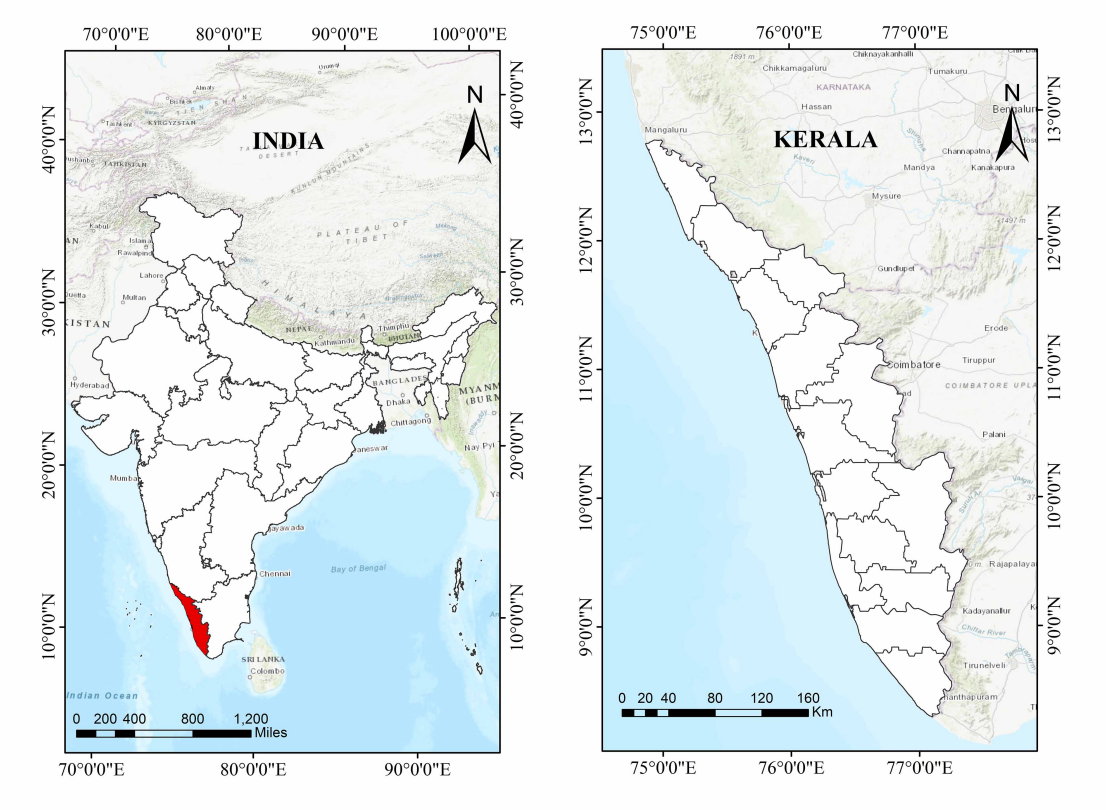

Figure 5.1 Location of the study area for FSM.

than 200 sq. km. Severe floods affected Kerala in 2018 due to extreme rainfall. There were subsequent flood events in 2019, 2020 and 2021. The cause and impact of these consecutive flood occurrences have been analysed in various studies (Vijaykumar et al., 2021; Parthasarathy et al., 2021; Vilasan and Kapse, 2022; Bhuyan et al., 2022).

5.2.2 Data Source

Past records of flood occurrences can be estimated by analysing before and after flood images. Sentinel 1 SAR images from 2018, 2019, 2020 and 2021 floods are considered in preparing the flood inventory map. Sentinel 1 is data sourced from Copernicus Programme, conducted by the European Space Agency, with a mission of monitoring land and water. The pre and post-flood Sentinel 1 scenes were selected from each year to identify flood-prone areas by contrasting SAR backscattering values between two dates. The 32-bit Sentinel 1 level 1 Ground Range Detected (GRD) product in Interferometric Wide (IW) swath mode is preprocessed in GEE. Speckle filtering is carried out to reduce noise. Subsequently, the digital pixel values undergo a process of conversion to

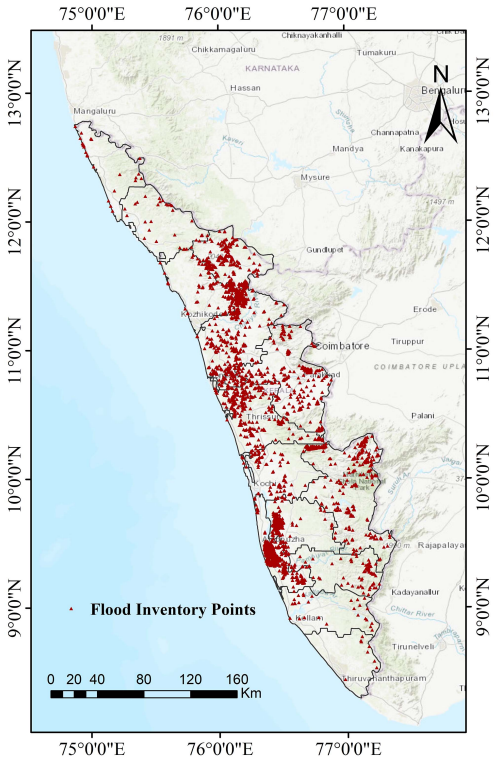

Figure 5.2 Location of flood inventories in the study area.

achieve radiometrically calibrated SAR backscatter across the entire Sentinel-1 image. Since the SAR sensors are side-viewing instruments, a Digital Elevation Model (DEM) is employed to perform terrain geo-coding on the data (Jiang et al., 2021). And then the backscatter intensities are converted to decibels (dB) using a logarithmic transformation. Figure 5.2 shows the developed flood inventory map of the study area. The ML models employed are trained using a dataset comprising 4,000 flooded points and 4,000 non-flooded points. The training (80%) and testing set (20%) are randomly selected from the sample points.

5.2.3 Flood Conditioning Features

The 12 flood-triggering factors considered as thematic layers include Elevation, Slope angle, Curvature, Aspect, Topographic Wetness Index (TWI), Rainfall, Distance to river, Stream Power Index (SPI), Soil type, Lithology, Normalized Difference Vegetation Index (NDVI) and Land use.

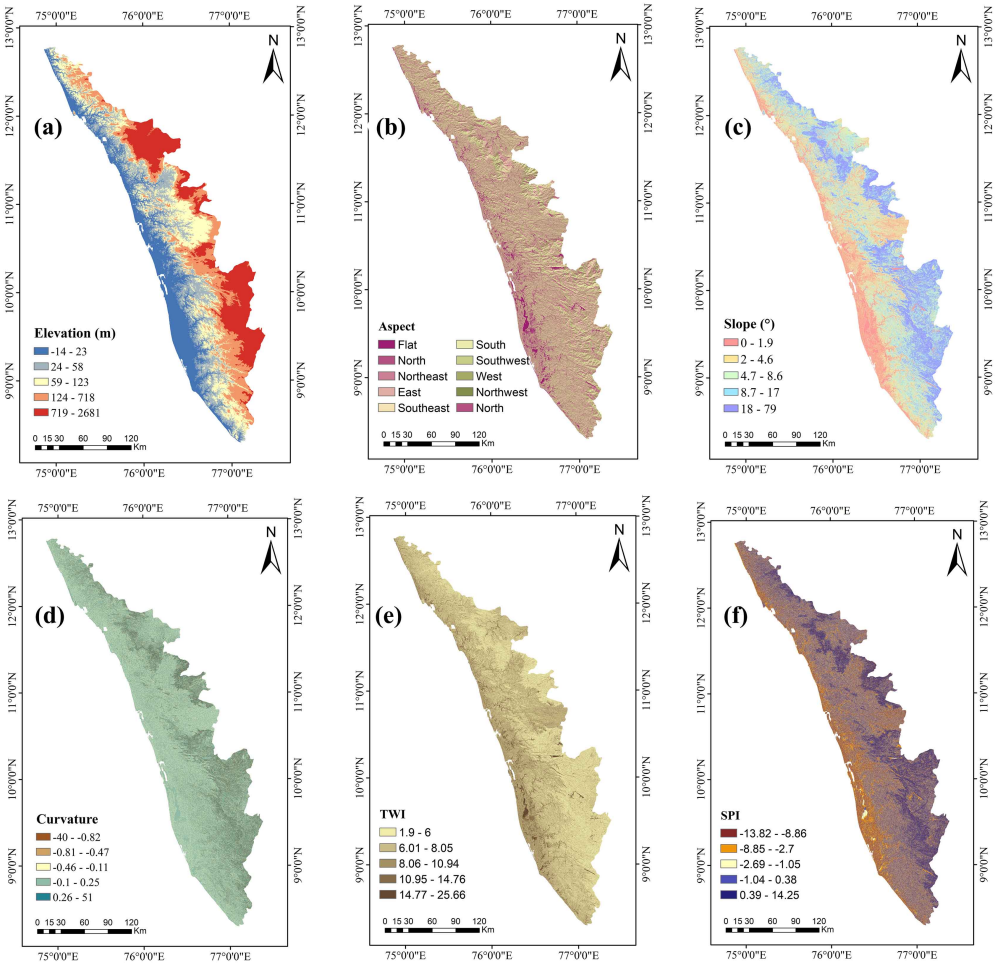

Figure 5.3 Thematic maps of the study area. (a) Elevation, (b) Aspect, (c) Slope, (d) Curvature, (e) TWI, (f) SPI, (g) Rainfall, (h) Distance to rivers, (i) Geology, (j) Soil, (k) Land use, (l) NDVI.

1. Elevation:

   Flood frequency and river discharge increase as elevation decreases. Flooding is inversely proportional to elevation (Pham et al., 2021; Kazakis et al., 2015). Elevation ranges from -14 to 2681 km in the region of study.

2. Aspect:

   The term aspect refers to the direction of the steepest slope on the surface of the terrain. It influences soil moisture, infiltration, local climate and solar radiation. Thus indirectly affects flood (Pham et al., 2021; Wang et al., 2020a). The nine classes that make up the aspect layer are grouped into flat ((-1°)-0°), north (0°-22.5° & 337.5°-360°), northeast (22.5°-67.5°), east (67.5°-112.5°), southeast

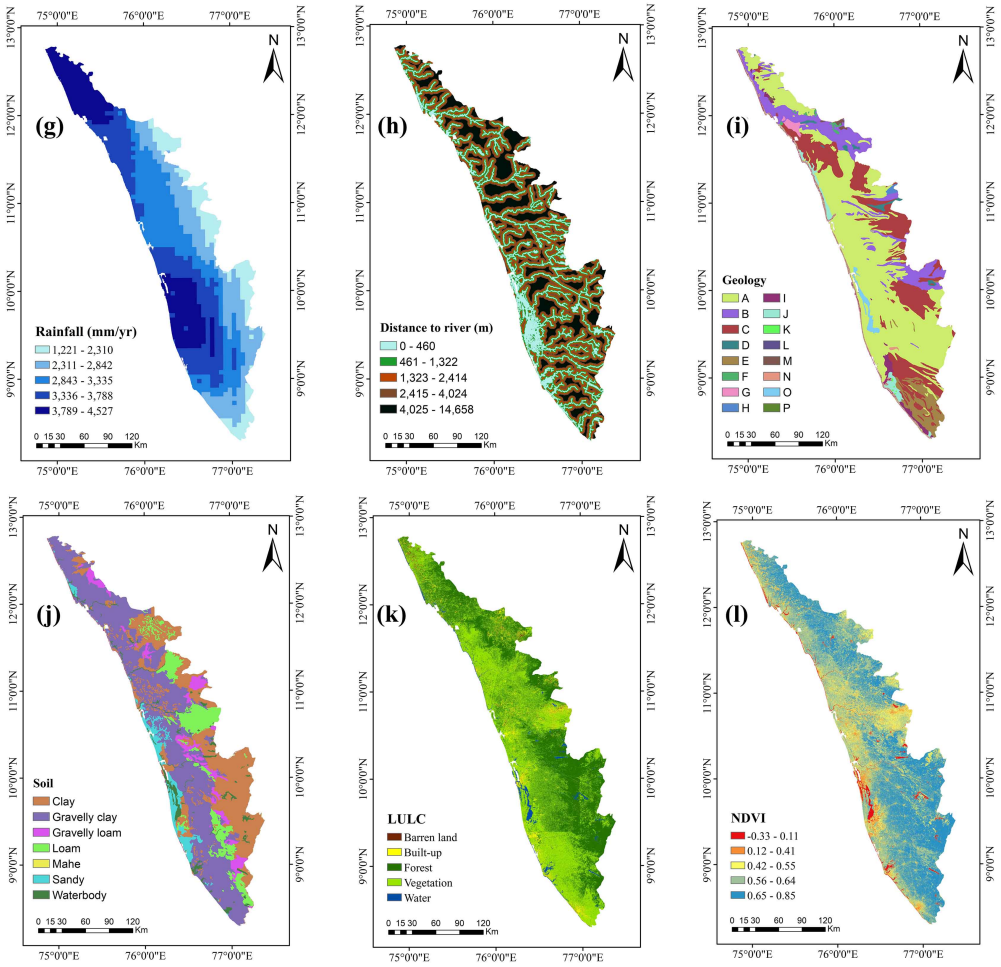

Figure 5.3 (continued.)

(112.5°-157.5°), south (157.5°-202.5°), southwest (202.5°-247.5°), west (247.5°-292.5°), and northwest (292.5°-337.5°).

3. Slope:

   Slope affects the quantity of surface runoff and penetration as water runs from higher to lower altitudes. Flat low-elevation areas may flood faster than higher-elevation areas with steeper slopes (Kazakis et al., 2015). Slope varies from 0 to 79°.

4. Curvature:

   Curvature measures the level of slope surface distortion and reflects the topography's morphology in a certain area. A convex slope indicates a positive curvature

and a concave slope represents a negative slope, while planar surfaces have values between 0 and 1 (Pham et al., 2021). In the study area, the value of curvature ranges from -40 to 51.

5. TWI:

   The proportion of the basin area to the slope is known as TWI. It gives an indication of the water retention capacity and soil moisture content in a watershed. It can accurately depict the areas of a watershed that are susceptible to flooding (Chapi et al., 2017) and is computed as follows:

   $$TWI = ln\frac{A_s}{tan\beta}, \tag{5.1}$$

   where A_s represents the upslope area per unit contour length and β (in radians) represents the slope angle. TWI ranges from 1.9 to 26, indicating an extremely high TWI value for the study.

6. SPI:

   SPI serves as an indicator of flowing water's erosive power. The strength of the flood increases with increasing SPI value. Smaller SPI values indicate areas where flow accumulation is possible. The value of SPI varies from -13.82 to 14.25 and can be calculated using the following equation (Wang et al., 2020a):

   $$SPI = A_s\, tan\beta. \tag{5.2}$$

7. Rainfall:

   Flooding is frequently brought on by a lot of rain in a short amount of time. With more rain, there is a greater chance of a flood (Chapi et al., 2017). The average annual rainfall from the years 2018, 2019, 2020 and 2021 are considered. The annual rainfall ranges from 1221 to 4527 mm/yr in the case study.

8. Distance to river:

   It is the Euclidean distance from the nearest river. Flood-prone areas are in closer proximity to river networks which serve as the primary conduits for flood flow and expansion (González-Arqueros et al., 2018; O'Neill, 2016). The distance from river varies from 0 to 14658 m in the case study.

9. Geology:

   The permeability of rocks affects the rate of infiltration. Impermeable rocks cause surface runoff resulting in a severe flood. The major geological classes in the study area include charnokite, khondalite group, peninsular gneiss, migmatite-gneiss and granite (Das, 2019; Hao et al., 2021). Geology is classified into 16 classes - charnokite group (A), peninsular gneiss (B), migmatite-gneiss (C), granite (D), khondalite group (E), wayanad group (F), vengad group (G), satya-mangalam group (H), warkhali group (I), undiffined coastal sediments (J), quilon formation (K), cuddalore formation (L), gabbro and anorthosite co (M), alkali syenite (N), water body (O) and closepet granite (P).

10. Soil types:

    Different soil textures have varying water storage potential. Small-sized particles in soils are characterised by narrow pore sizes, which decrease water infiltration leading to flooding susceptibility (Wang et al., 2020a; González-Arqueros et al., 2018). Soil is classified into clay, gravelly clay, gravelly loam, loam, mahe, sandy and waterbody.

11. Land use:

    Due to its potential to change flood flow and sediment transport, land use influences the likelihood of flooding. While forest and lush vegetation encourage penetration, urban and grazing areas stimulate flooding (Kazakis et al., 2015). LULC is classified into barren land, build-up, forest, vegetation and water.

12. NDVI:

    NDVI shows the vegetation coverage in a region and is calculated as follows:

$$NDVI = \frac{S_{NIR} - S_R}{S_{NIR} + S_R}, \tag{5.3}$$

    where S_R and S_{NIR} indicate, the spectral reflectance measured in the red and near-infrared regions, respectively. Spectral reflectance is the ratio of the reflected light intensity from a surface at different wavelengths to the incident light intensity. It has a range of -1 to +1. A value of -1 indicates perfect absorption (no reflection), while a value of +1 indicates perfect reflection (no absorption). NDVI variations

indicate changes in vegetation and surface water cover (Wang et al., 2020a). It can show the connection between vegetation and flooding. In the study region, higher vegetation concentrations are believed to lessen the chance of flooding. NDVI value varies from -0.33 to 0.85.

The thematic maps of the study area are shown in Figure 5.3. The factors slope, aspect, curvature, SPI and TWI are derived from DEM using ArcGIS 10 software. Each factor is categorized into a different number of classes.

5.2.4 Machine Learning Models

AutoML Models

An AutoML system automates machine learning from start to finish involving automating data preparation, feature processing, model generation and estimation. Once the data is submitted, the AutoML system will automatically decide which model is optimal for that particular application. AutoML models like AutoKeras and TPOT are used to predict flood susceptibility.

AutoKeras is a Keras-based AutoML model that specialises in deep learning challenges and uses Bayesian Optimization (BO) to direct the search for neural architecture. It also uses a neural network kernel and a tree-structured acquisition function optimisation technique to precisely evaluate the search space. Three different options for hyperparameter optimization are offered by AutoKeras. Users can choose between hyperband, random search, and BO for optimization.

TPOT is a genetic programming-based AutoML model. Using Genetic Algorithm (GA) for hyperparameter tuning, TPOT enhances classification accuracy by optimising several machine learning models and feature preprocessors.

Ensemble Models

Ensemble algorithms like the Random Forest, Gradient Boost, XGBoost and AdaBoost are considered for building a flood susceptibility model. The flood-triggering factors mentioned in the material and methods section are used as the predictor variables for the ML models. The factors: slope, aspect, elevation, curvature, SPI, TWI, rainfall,

distance to river and NDVI are continuous or discrete values. Geology, soil type and land use are categorical variables.

Multiple decision trees are used in the Random Forest ensemble learning technique for regression and classification (Breiman, 2001). It is a bagging technique. Bagging is the process of randomly selecting samples from the initial data set. It is also called bootstrap aggregation. Bootstrap is the process of creating sample datasets for each decision tree by performing row sampling and feature sampling on the original dataset. In the case of classification, the output is determined by the class that the majority of the trees in the ensemble yields. This step is called aggregation.

Gradient Boost also uses a boosting technique. It creates models one at a time, each one aiming to fix the shortcomings of the one before it (Friedman, 2001). In contrast to Adaboost, each predictor is trained using the predecessor's residual error as labels rather than changing the training instance weights.

Like Gradient Boost, XGBoost also uses residuals rather than the actual class labels to build the trees (Chen and Guestrin, 2016). Since residuals are continuous values, base estimators here are regression trees and not classification trees.

Adaptive boosting or AdaBoost is an ensemble modelling that uses the boosting algorithm (Dou and Chen, 2017). Boosting is a technique that uses a number of weak classifiers to build a strong classifier. Initially, a model is built using the training data set. Then, in an effort to address the shortcomings of the first model, a second model is developed. These steps are reiterated until either the maximum number of models has been added or the entire training dataset has been accurately predicted.

CNN Architectures

In processing classification-related tasks, CNNs have produced generally positive results. The main objective of a flood susceptibility assessment is to forecast the location of a flood occurrence. This can be accomplished by classifying a region as "flood" or "non-flood" using a binary classification algorithm. Wang et al. introduced the use of CNN - 1D, 2D and 3D models for flood prediction in Shangyou County, China (Wang et al., 2020a). Similar data representations and architectures are used for the CNN models used in this work. The following are the key phases in applying the CNN models.

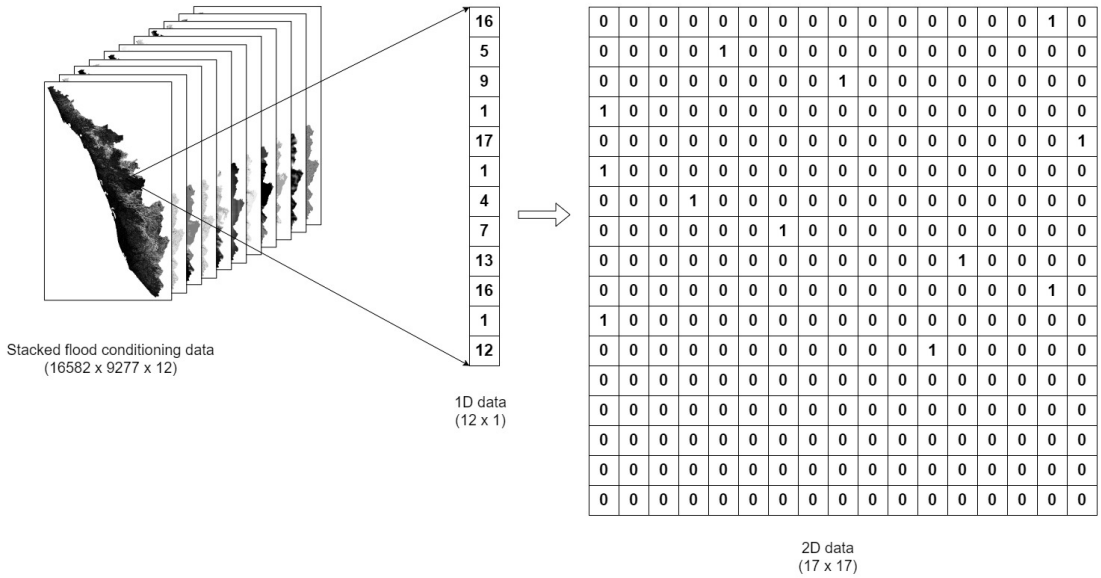

Figure 5.4 One-dimensional and two-dimensional input data representation for CNN model.

The raw data is initially translated into various data representations as shown in Figure 5.4 and Figure 5.5. The data is then subjected to a sequence of convolutional and max pooling layers, followed by fully connected layers. Lastly, applying a nonlinear activation function, the data is converted into binary outputs. All the flood predictor variables are converted to categorical values before being input into the CNN models.

1D-CNN: The 1D-CNN structure consists of five layers: input, convolutional, max pooling, fully connected, and output layers. It was created and trained using one-dimensional data. There are 12 flood conditioning factors in the input data. The first convolution layer has 20 convolution kernels of size 3 x 1. The second layer is a max pooling layer with kernels of size 2 x 1, which reduces the length of feature vectors. A fully connected layer, containing 20 neural units that flattens the extracted representations, follows next. The output layer consists of 1 neural unit. Figure 5.6 represents the one-dimensional CNN architecture.

2D-CNN: The 2D input data is in the form of a 17 x 17 image. This 2D image is generated from the 1D representation. The idea is to convert the 1D array to a *n*x*n* matrix, where *n* is the largest between the number of flood conditioning factors and the number of categories belonging to a factor. A 17 x 17 image is used because the

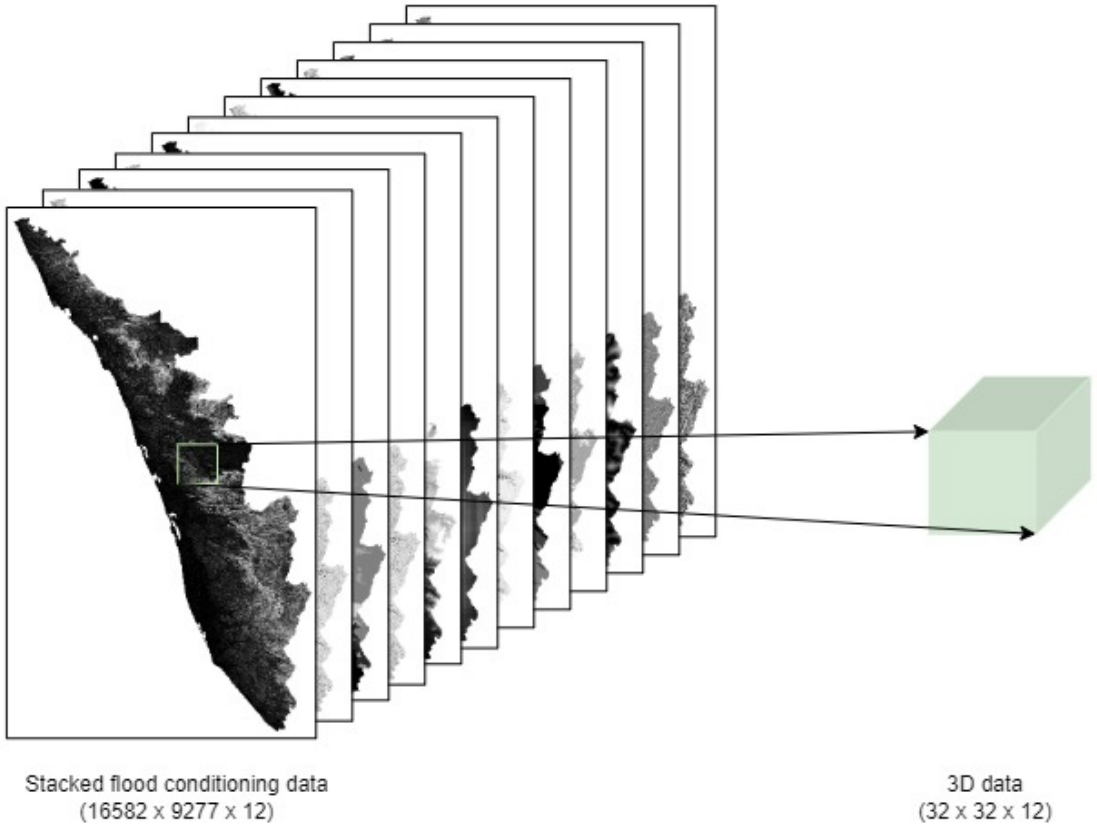

Figure 5.5 Three-dimensional input data representation for CNN model.

maximum number of categories among the 12 features is 17 (belonging to geology, including the null category). Each cell value in the 1D array is converted to a row of length 17 in 2D representation. For example, a value of 5 in the second cell of the 1D array (representing attribute value "Southeast" in Aspect) corresponds to the row [0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0] in 2D structure. Here the fifth element is assigned 1 and the rest of the elements are assigned 0s. A sample conversion of the 12 x 1 data to the 17 x 17 matrix is depicted in Figure 5.4.

Two convolutional, two max-pooling and one fully connected layer make up the two-dimensional CNN structure. The first layer is a convolutional layer with 20 kernels, each of size 3 x 3. A max pooling layer with filters of size 2 x 2 follows. The second convolutional layer has 10 kernels with a size of 3 x 3 and receives the generated feature maps. The second 2 x 2 max pooling layer receives the produced 10 feature maps and uses a fully connected layer containing 50 neural units to detect the extracted features. In the end, the 2D-CNN design produces 1 neural unit. Figure 5.7 shows the

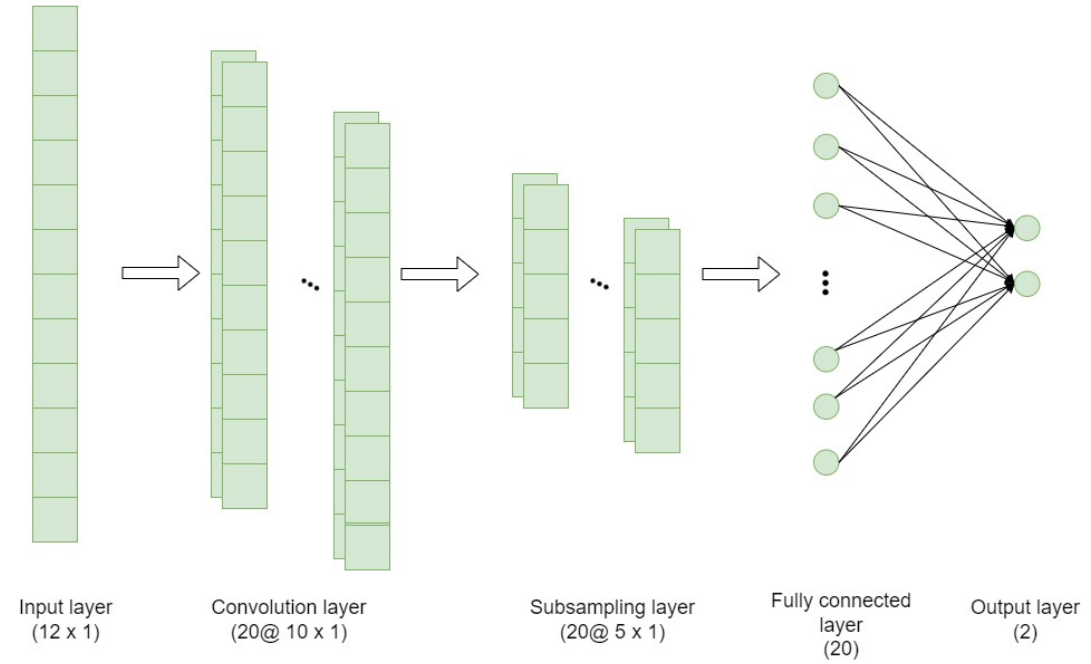

Figure 5.6 One-dimensional CNN architecture.

2D architecture of the CNN model.

3D-CNN: For each point (flooded and non-flooded), the adjacent pixels are considered forming a training sample of size 32 x 32 x 12. The label of this sample is the same as the flooded/non-flooded pixel. The three-dimensional data representation of input samples is depicted in Figure 5.5. The 3D CNN model comprises of 2 convolution layers and a mix of max-pooling layers. The kernel size for convolution layers is 3. The subsampling layer has filters of size 2. There is one fully connected layer with

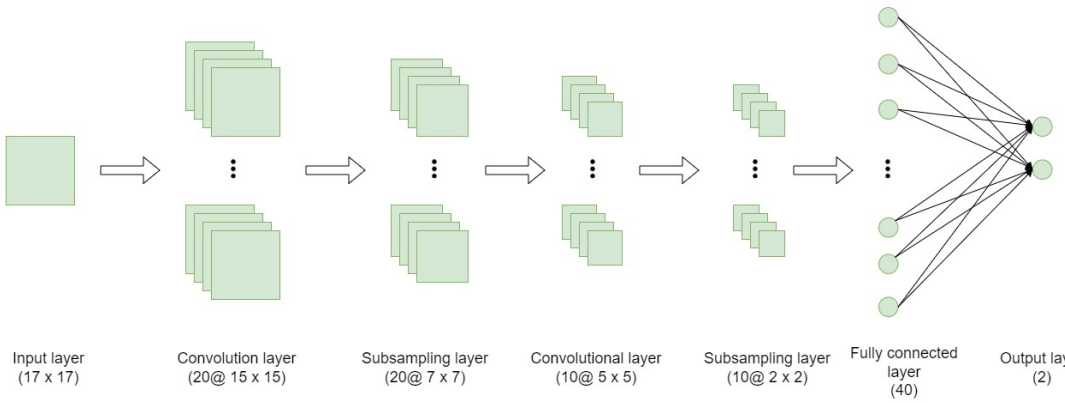

Figure 5.7 Two-dimensional CNN architecture.

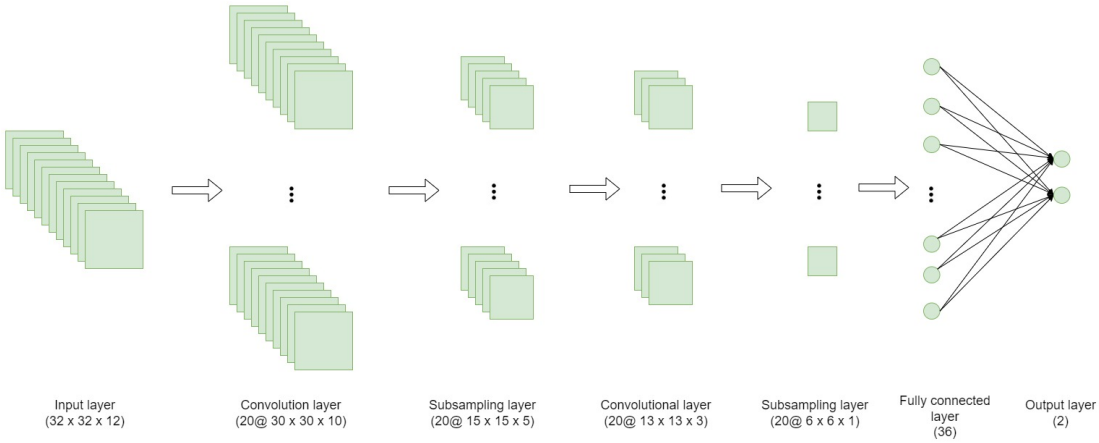

Figure 5.8 Three-dimensional CNN architecture.

36 neural units and a final output layer with 1 neural unit. The number of kernels used is 20. Figure 5.8 illustrates the 3D CNN architecture.

The parameters of CNN are optimized by algorithms like BO-DE and BO-CMAES which are combinations of Bayesian optimization with evolutionary algorithms - Differential Evolution and Evolutionary Strategies. The hyperparameters optimized include momentum, learning rate, learning rate decay factor, optimizer, the number of epochs, weight decay and batch size. The overall methodology of using different models for FSM is explained in detail in Figure 5.9.

Hyperparameter	Range
Initial learning rate	[0.0001 - 1.0]
Batch size	[32 - 1024]
Epochs	[10 - 100]
Momentum	[0.9 - 0.999]
Optimizer	[Adam, RMSprop, Adagrad, SGD]
Learning rate decay factor	[0.1 - 0.0001]
Weight decay	[0 - 0.1]

Table 5.1 List of hyperparameter values of flood susceptibility models used for tuning.

The hyperparameters given in Table 5.1 of 3D-CNN model are optimized using the algorithms proposed in Chapter 3.

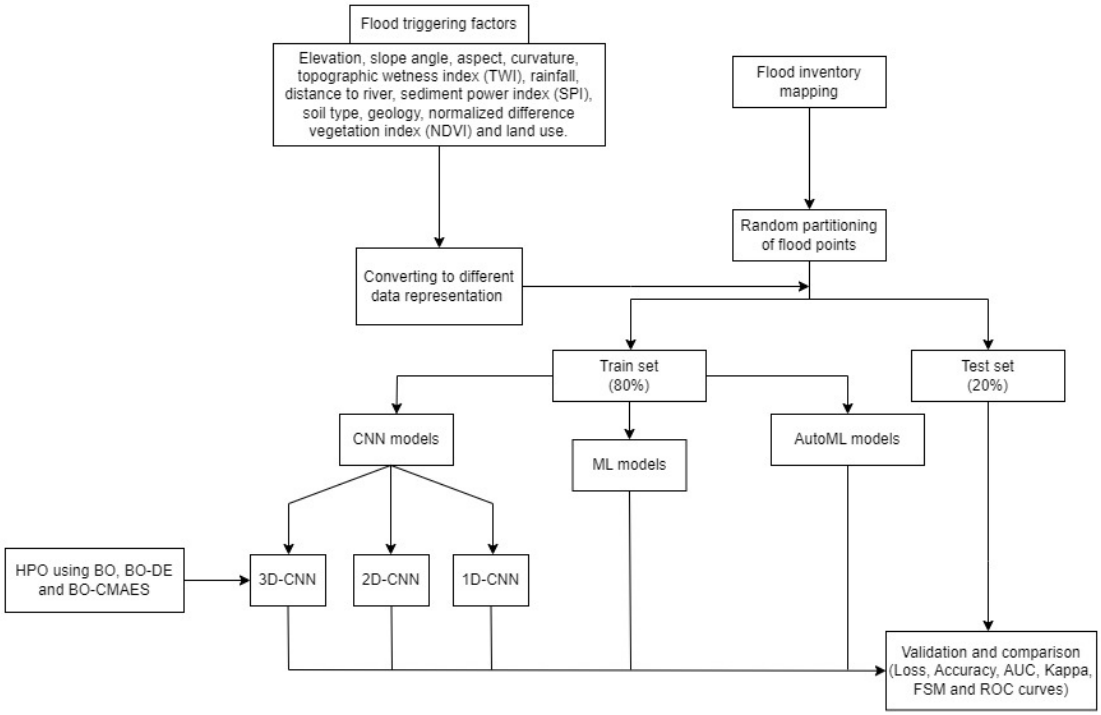

Figure 5.9 Flowchart of the methodological steps employed for FSM.

5.3 RESULTS AND DISCUSSION

5.3.1 Evaluation Metrics

The data is trained for ML and CNN models separately depending on the data representation used for each model. Training and testing data is split, and k-fold cross-validation is used before obtaining the results.

The evaluation metrics used here are accuracy and binary cross-entropy loss. The accuracy score is an evaluation metric used in classification problems, indicating the number of correct predictions obtained. By definition, it is the ratio of the number of accurate predictions to all of the predictions. The formula for accuracy is given in Equation 4.1.

Loss numbers represent the discrepancy between actual and anticipated target values. Binary cross entropy compares predicted probabilities to actual binary class output. The score is determined by penalising the probabilities according to how far they are from the expected value.

$$L = \frac{1}{N}\sum_{i=1}^{n} y_i.ln(p(y_i)) + (1-y_i).ln(1-p(y_i)), \tag{5.4}$$

where n is the number of samples, y is the true value and $p(y)$ is the predicted value. Loss indicates how well a model learns and accuracy indicates how well the model predicts. A model with high accuracy and low loss value indicates that the model makes highly accurate predictions with few small errors. If a model has low accuracy and high loss value, it is inferred that the model makes large number of incorrect predictions with many big errors. Even though both metrics give interpretations of the model performance, there is no direct relationship between the two metrics.

Precision is the measure of a model's ability to correctly classify the positive instances among all the instances it has predicted positive. It is given by the formula:

$$Precision = \frac{TP}{TP+FP}. \tag{5.5}$$

A high value of precision indicates that when the model predicts a positive outcome, it is more likely to be correct. The value of precision improves with a decrease in number of false positives.

Recall also called True Positive Rate (TPR) is defined as follows:

$$Recall\ (TPR) = \frac{TP}{TP+FN}. \tag{5.6}$$

Out of all true positive samples, recall calculates the percentage of true positive outcomes that the model correctly classified. A high value of recall indicates that the model could correctly identify a large portion of the positive instances. The value of recall increases with a decline in number of false negatives.

The formula for False Positive Rate (FPR) is:

$$FPR = \frac{FP}{FP+FN}. \tag{5.7}$$

A graph plotting TPR against FPR is called a ROC curve (receiver operating characteristic curve). AUC is the Area under the ROC Curve. It estimates the ROC curve's two-dimensional region underneath it.

Cohen's Kappa (K) is a statistic that gauges inter-rater reliability. K is generally used to evaluate the effectiveness of flood prediction models. It is measured by the Equation 4.2, where P_{obs} is the level of observable agreement between raters and P_{exp} is the expected likelihood of coincidence.

P_{exp} in terms of TP, TN, FP and FN is given by:

$$P_{exp} = \frac{((TP+FN)(TP+FP)+(FP+TN)(FN+TN))}{\sqrt{(TP+TN+FN+FP)}}. \tag{5.8}$$

AutoML Models	HPO	Loss	Accuracy
AutoKeras	BO	0.4819	77.81 ± 0.18
	Random Search	0.2648	91.87±0.46
	Hyperband	0.6377	60.38±0.51
TPOT	GA	0.8761	56.42±0.03

Table 5.2 Loss and accuracy for AutoML models with different HPOs.

ML & CNN Models	Loss	Accuracy	Precision	Recall	AUC	Kappa
Random Forest	3.9365	**88.60±0.13**	**0.89**	0.90	**0.9505**	**0.7720**
Gradient Boost	4.1589	87.96±0.04	0.83	0.81	0.9451	0.7592
XGBoost	4.0922	88.15±0.21	0.86	0.87	0.9482	0.7630
AdaBoost	5.7380	83.39±0.18	0.89	0.88	0.9143	0.6678
1D-CNN	0.3731	84.74±0.31	0.79	0.88	0.9181	0.6948
2D-CNN	0.8258	81.19±0.07	0.81	0.81	0.8917	0.6278
3D-CNN	**0.3038**	87.32±0.36	0.80	**0.96**	0.9417	0.7422

Table 5.3 Loss and accuracy for ML and CNN models without HPO.

$K = 1$ denotes full agreement amongst the raters. $K = 0$ if there is no agreement among the raters beyond what can be predicted by chance (as indicated by P_{exp}). If there is no correlation between the evaluations of the two raters, it is feasible for the statistic to be negative.

5.3.2 Model Evaluation

The binary cross-entropy loss and accuracy obtained (with 95% confidence intervals) for AutoML models are given in Table 5.2. AutoKeras with random search used for HPO delivers the highest accuracy and the lowest log loss value. TPOT performs poorly compared to AutoKeras. It gives the lowest accuracy of 56%. An accuracy of 60% is attained by AutoKeras with hyperband. Better hyperparameter configuration is provided by Bayesian optimization, which increases accuracy from hyperband by 17%.

Table 5.3 compares results obtained from all ensemble models and the CNN models considered. The hyperparameters of these models are set to their default values. That is, no HPO method is used for tuning the hyperparameters.

A suitable model for a particular problem depends majorly on the type of the prob-

HPO methods	Loss	Accuracy	Precision	Recall	AUC	Kappa
BO	1.3732	89.24 ± 0.16	0.85	0.96	0.9493	0.7863
BO-DE	0.8625	91.12 ± 0.22	0.87	0.95	0.9682	0.8127
BO-CMAES	**0.6938**	$\mathbf{96.39 \pm 0.07}$	**0.95**	**0.97**	**0.9914**	**0.8651**

Table 5.4 Loss, accuracy and AUC score for 3D-CNN models with different HPOs.

lem and the nature of the data. In terms of accuracy, Random Forest performed the best with 88.60% followed by XGBoost, Gradient Boost and 3DCNN with 88.15%, 87.96% and 87.32% accuracy respectively (Table 5.3). Random Forest also has the highest AUC score of 0.95 and kappa score of 0.77. While CNNs are particularly powerful in learning hierarchical patterns from spatially structured data, such as images; ensemble models (Random Forest, XGBoost and Gradient Boost) are efficient in learning from smaller datasets. While deep learning models, including CNNs, often require a significant quantity of labelled data to attain good performance, ensemble models can perform well even with little data. Ensemble models combine multiple base models (weak learners in Gradient Boosting or decision trees in Random Forest) yielding a more diverse set of predictions. The risk of overfitting is reduced by this diversified learning, which aids in capturing multiple aspects of the data and results in improved generalisation. In general, ensemble models are more resistant to noise or anomalies in data. Individual errors from one model may get balanced out by accurate predictions from other models resulting in more reliable overall performance. Whereas CNNs can be sensitive to noisy data, and outliers may have a bigger effect on their performance. These are the possible reasons why the ensemble models performed better than CNN in terms of accuracy for the FSM dataset.

Table 5.3 also gives the recall and precision values obtained for ML and CNN models. A high precision value means the model is making fewer false positive errors. It can be observed that RF and AdaBoost give the highest precision value of 0.89. Recall is an evaluation metric that is particularly important in scenarios where missing positive instances can have serious consequences. Like in the case of FSM models, failing to identify a location susceptible to flood can cause major harm. The model is expected

to identify locations that are vulnerable to flood without fail. Therefore it is crucial to have a lower number of false negatives (or type 2 errors) in the FSM model. Fewer false negatives indicate a higher recall value. Here the 3D CNN model has the highest recall value of 0.96.

In terms of cross-entropy loss, CNN provides lower loss values, meaning the model learns better with fewer small errors. With high recall and lower loss values, the 3D CNN model performed better overall. The attempt was made to improve the accuracy of the 3D CNN model through hyperparameter tuning. The analysis of the 3D CNN model involved three different hyperparameter optimizations (HPOs), focusing on metrics such as loss, accuracy, precision, recall, AUC score, and kappa, as presented in the Table 5.4. All of the results are derived using the Friedman test with a statistical significance of p-value < 0.05 on data from 20 iterations.

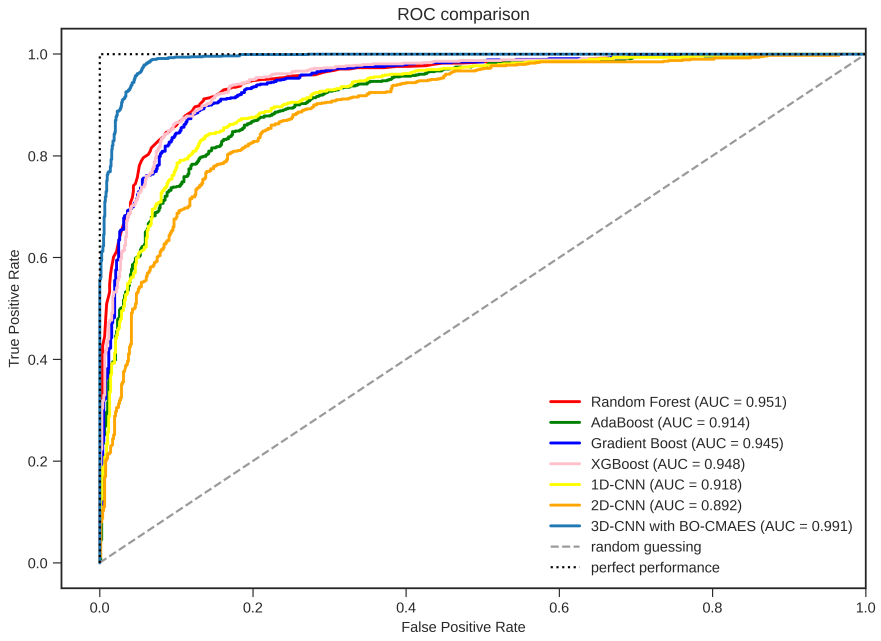

Figure 5.10 ROC curves of different susceptibility models used in the study area.

The hyperparameters used in the study are crucial in determining the efficiency of the CNN model. It is evident from the results that fine-tuning these parameters has significantly increased the performance of the model. The 3D-CNN model's accuracy improved more as a result of the usage of hyperparameter optimization approaches. The model that uses Bayesian optimization enhances the accuracy by 2% (from 87 to

89%). The best set of hyperparameter configurations is obtained with BO-CMAES. The accuracy for evolutionary algorithm-assisted BO approaches rose by 2 and 7% (from 89 to 91 and 96, respectively). There is an increase of 0.0189 and 0.0424 AUC scores for BO-DE and BO-CMAES techniques respectively. In terms of precision, there is an improvement of 0.02 and 0.1 for BO-DE and BO-CMAES respectively compared to BO. Also, BO-CAMES has the highest recall of 0.97. All of this indicates that the HPO techniques have helped find better hyperparameter combinations that improved the prediction capability of the CNN model. Figure 5.10 shows an analysis of model performances using the ROC curve.

5.3.3 Flood Susceptibility Maps

The final step in the methodology is to map machine learning model outputs into several flood-prone zone types. FSMs are plotted with five types of susceptibility (very low, low, moderate, high and very high) using ArcGIS's commonly used natural breaks (Jenks) classification approach (Chapi et al., 2017; Dano et al., 2019). Figure 5.11 displays the flood susceptibility maps produced by several ML and CNN models. The percentage of area that falls under each zone, predicted by all the models, is represented by a stacked bar chart in Figure 5.12.

Based on the flood susceptibility maps generated by the models, the eastern section of the study, which is located at higher altitudes, is less susceptible to floods than the western part, which is located at lower altitudes (Figure 5.4 (a)). The region around the districts - Cochin, Alleppey and parts of Wayanad is depicted in all models as high and very high flood susceptible zones. Cochin experiences flooding due to heavy rainfall and high build-up density in low-lying areas. The district mostly consists of flat terrain (less slope) and lower elevation regions making it more vulnerable to flood.

Alleppey has a diverse topography. It is a sandy stretch of land that is divided by rivers, canals, and lagoons. Except for a few isolated hillocks, there are no mountains or hills in the district. All of the taluks of Cherthala, Ambalappuzha, Kuttanad, and Karthikappally are located in lowland areas. 80% of the district is in the coastline zone, with the remaining 20% being in the midland region (Government of Kerala, 2023a). The district's 82 km of shoreline is continuous. In all of Kerala, Alleppey is the only

Model	Susceptibility level	Area (km^2)	Number of flooding points	Percentage of area (%)	Percentage of flooding points (%)	Flooding points density ($1/km^2$)
Random Forest	Very low	14607.5	21	38.5	0.5	0
	Low	9270.2	21	24.4	0.5	0
	Moderate	5973.1	63	15.7	1.6	0.01
	High	4489.5	280	11.8	7	0.06
	Very high	3595.2	3615	9.5	90.4	1.01
Gradient Boost	Very low	13095.7	65	34.5	1.6	0
	Low	9127.7	129	24.1	3.2	0.01
	Moderate	7339.8	264	19.3	6.6	0.04
	High	4977.7	536	13.1	13.4	0.11
	Very high	3394.6	3006	8.9	75.2	0.89
XGBoost	Very low	15652.9	40	41.3	1	0
	Low	8627.3	39	22.7	1	0
	Moderate	5612.2	54	14.8	1.4	0.01
	High	4268.3	202	11.3	5.1	0.05
	Very high	3774.9	3665	10	91.6	0.97
AdaBoost	Very low	6557.2	15	17.3	0.4	0
	Low	9503.4	105	25.1	2.6	0.01
	Moderate	11235.9	434	29.6	10.9	0.04
	High	7454.2	1118	19.6	28	0.15
	Very high	3184.8	2328	8.4	58.2	0.73

Table 5.5 Flooding point density on flood susceptibility maps of ML models.

district devoid of highland and forested areas. 13% of the district is made up of water bodies, and Kuttanad is located below sea level. The soil in this region is clay dominant which has low permeability and causes waterlogging (Figure 5.4 (j)). All of this makes Alleppey significantly more flood-prone.

Wayanad is exposed to flood due to heavy rainfall and the resulting overflow of rivers and dams in the region. The region's topography, with its hills and valleys, can lead to water build-up and flash floods in low-lying areas. All of Kerala's significant flood-prone areas have been accurately identified in all FSMs. Based on these maps, the very high flood susceptibility ranges from 8.4% to 14.2% of the study's total area across all models, and the high susceptible regions run from 10.6% to 19.6%.

The count of flood points and distribution of area for each susceptibility level for different models are shown in Table 5.5 and Table 5.6. The ratio of the number of flood points to the area in each susceptibility zone is called the flooding point density (FPD). In an ideal situation, the value of FPD increases from an extremely low vulnerability to extremely high susceptibility (Zhao et al., 2019). The highly susceptible locations had

Model	Susceptibility level	Area (km^2)	Number of flooding points	Percentage of area (%)	Percentage of flooding points (%)	Flooding points density ($1/km^2$)
1D-CNN	Very low	10705.5	82	28.2	2.1	0.01
	Low	8649.2	153	22.8	3.8	0.02
	Moderate	7660.5	256	20.2	6.4	0.03
	High	6686.5	555	17.6	13.9	0.08
	Very high	4233.9	2954	11.2	73.9	0.7
2D-CNN	Very low	15649.4	81	41.3	2	0.01
	Low	8180.8	51	21.6	1.3	0.01
	Moderate	5854.7	100	15.4	2.5	0.02
	High	4539.2	295	12	7.4	0.06
	Very high	3711.4	3473	9.8	86.8	0.94
3D-CNN	Very low	10618.3	19	28	0.5	0
	Low	9366	21	24.7	0.5	0
	Moderate	7187.2	27	18.9	0.7	0
	High	5856.4	40	15.4	1	0.01
	Very high	4907.6	3893	12.9	97.3	0.79
3D-CNN BO	Very low	10674.3	10	28.1	0.3	0
	Low	9183	104	24.2	2.6	0.01
	Moderate	6866.3	298	18.1	7.5	0.04
	High	5816.1	200	15.3	5	0.03
	Very high	5395.8	3388	14.2	84.7	0.63
3D-CNN BO-DE	Very low	17000.1	615	44.8	15.4	0.04
	Low	8291.6	24	21.9	0.6	0
	Moderate	5373.4	10	14.2	0.3	0
	High	4038.6	9	10.6	0.2	0
	Very high	3231.7	3342	8.5	83.6	1.03
3D-CNN BO-ES	Very low	13659.9	0	36	0	0
	Low	8946.7	0	23.6	0	0
	Moderate	6190.4	0	16.3	0	0
	High	4653	7	12.3	0.2	0
	Very high	4485.5	3993	11.8	99.8	0.89

Table 5.6 Flooding point density on flood susceptibility maps of CNN models.

the highest FPD in the flood susceptibility map produced by 3D-CNN (that uses BO-DE for HPO). But for this model, FPD does not increase from the lowest to the highest level of susceptibility. The very high susceptibility regions covered by 3D-CNN with BO-ES model have the highest percentage of flooding points (99.8%). The model also estimated 0 flooding spots in the very low susceptible zone, which is the lowest among all the models compared.

The flood susceptibility maps produced from 3D CNN BO-ES and other models for two specific regions is given in Figure 5.13 and Figure 5.14. Box 1 is a region in the Kasargod district of Kerala State in India (geographical location given in the figure). It

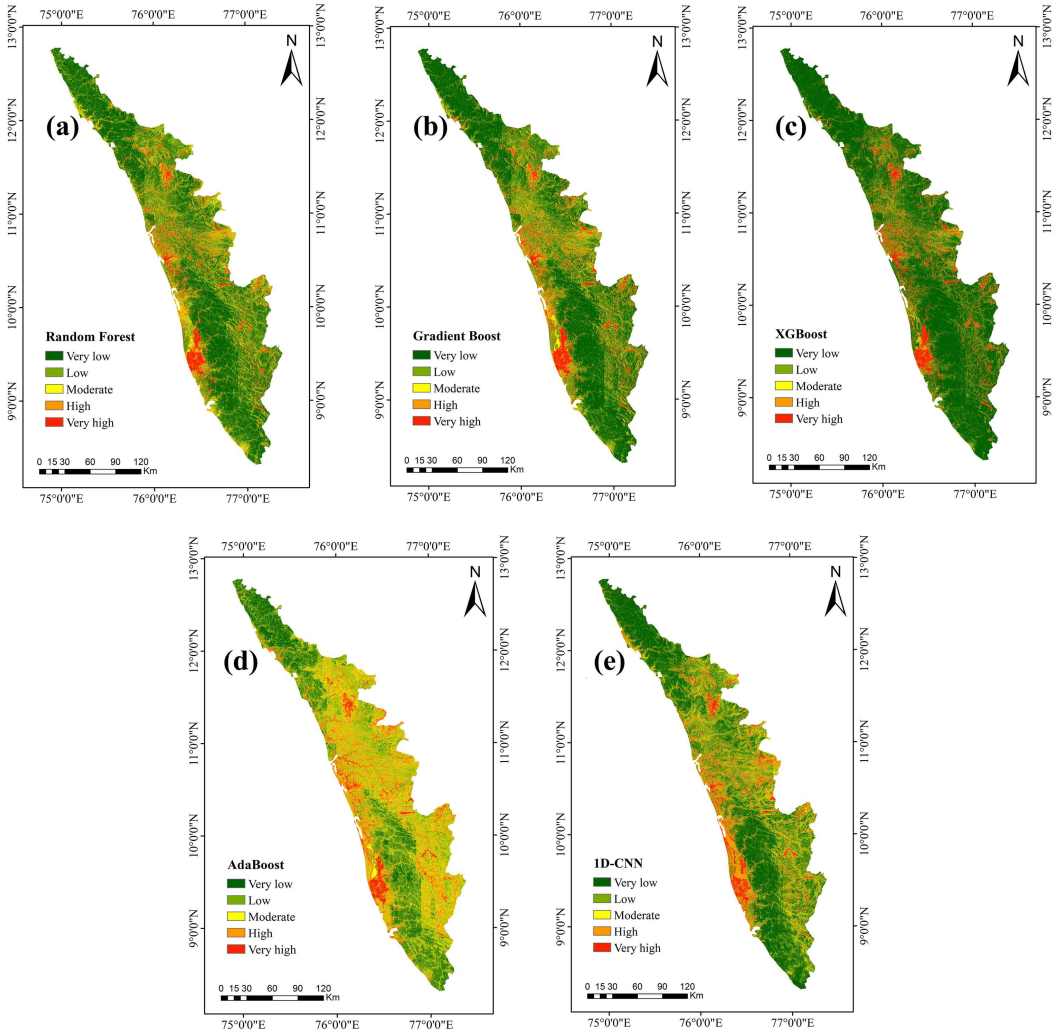

Figure 5.11 Flood susceptibility maps of the study area obtained using (a) Random Forest, (b) Gradient Boost, (c) XGBoost, (d) AdaBoost, (e) 1D-CNN, (f) 2D-CNN, (g) 3D-CNN, (h) 3D-CNN with BO, (i) 3D-CNN with BO-DE, (j) 3D-CNN with BO-CMAES.

has a higher elevation compared to the surrounding region and has no inundation sites. The northeastern part of the box comprises a portion of Chandragiri River, which is a moderately flood-vulnerable area. Theoretically, there shouldn't be any particularly high or vulnerable regions. The southwest part of the box has an irrigation tank which can be a flooding point in case of torrential rainfall. Figure 5.13 shows that 3D CNN BO-ES successfully detect these regions. Other models identify a section of box 1 as very high and high susceptible areas. 3D CNN BO-ES describes more details of the roads and stream showing low and moderate vulnerability, respectively than the other models.

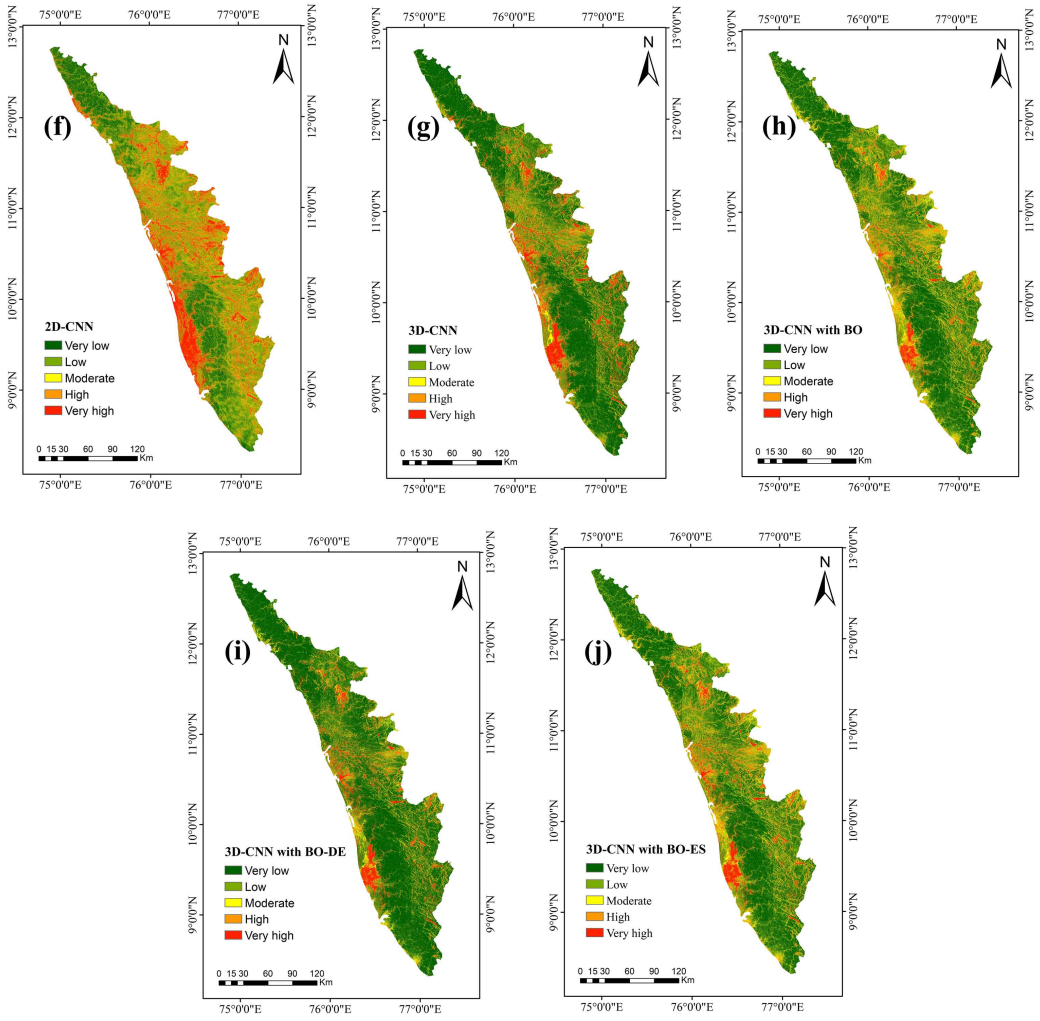

Figure 5.11 (continued.)

Box 2 is a region below Vembanad Lake in the Alleppey district of Kerala State in India (geographical location given in the figure) with a large number of historical flooding observations. It has high stream density where three rivers: Manimala, Pamba and Achankovil meet. Thus the region has very high and high vulnerability to flooding. With the exception of AdaBoost, all models successfully identify the majority of the flooding regions in box 2 as very high sensitive areas, as shown in Figure 5.14. AdaBoost shows most regions as high and moderate susceptible to flood. On the banks of the Pamba river which is seen in the eastern part of box 2, the region is moderately vulnerable due to the larger width of the river. At the point where the three rivers meet, due to higher stream density the region is more vulnerable to flood. Moving outwards, there

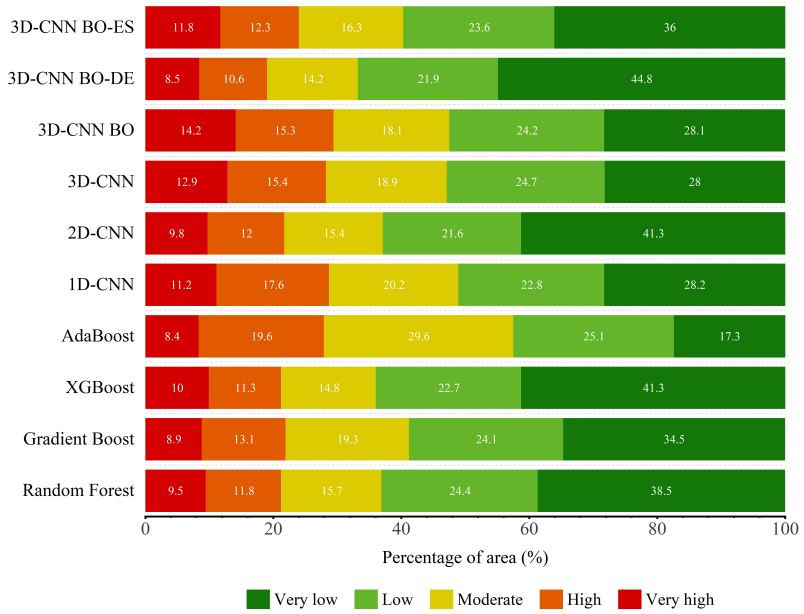

Figure 5.12 Percentage of area susceptible to flood given by different models

is a gradual decrease in vulnerability, i.e., from very high to moderate and from moderate to low susceptibility. This is correctly predicted by the 3D CNN BO-ES model. Though the 3D CNN model correctly predicts the high susceptible regions, the gradual change in susceptibility is not well depicted by this model. XGBoost classifies mostly to very high and very low flood-prone regions, showing only the extremes. Although 3D CNN BO-DE captures the gradual decline in susceptibility, it fails to identify the details, especially the river line. To sum up, 3D CNN BO-ES delivers accurate, comprehensive findings in areas that are both high and low susceptible.

5.4 CLOSING REMARKS

The fundamental concepts underlying flood susceptibility mapping encompass the evaluation of risk and the implementation of primary prevention strategies. It is imperative to provide timely and comprehensive warnings to residents residing in regions prone to flooding, in order to effectively communicate the associated risks and potential consequences. It is the responsibility of land-use planners and governmental bodies to disseminate up-to-date flood susceptibility assessments and regulations that prohibit

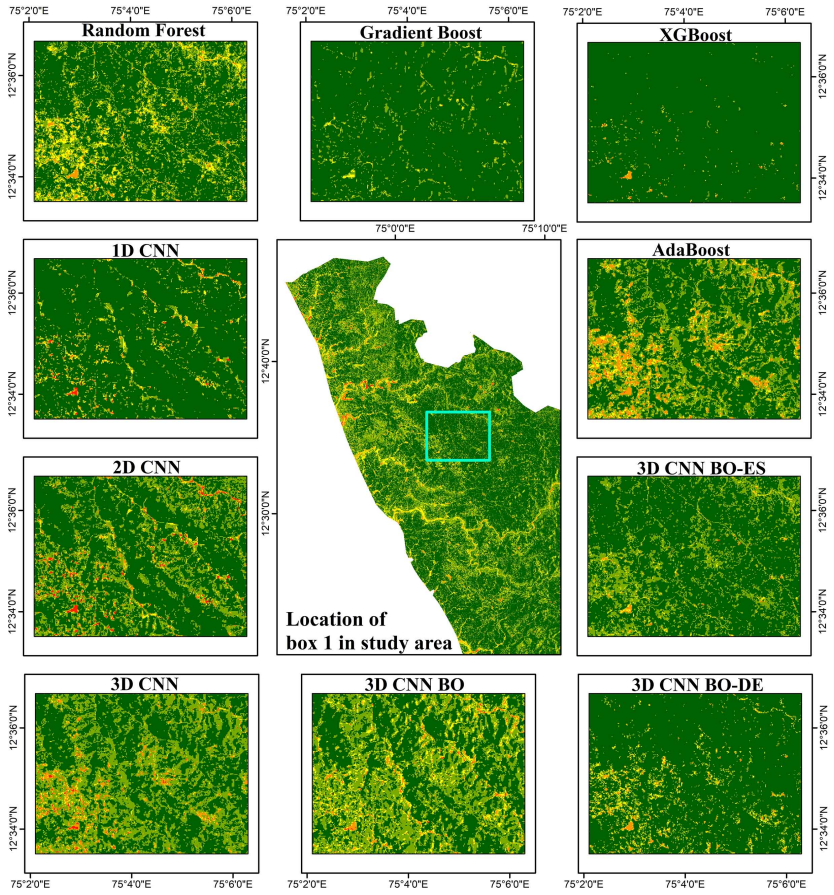

Figure 5.13 Comparison of flood susceptibility maps in detail within box 1.

the initiation of new projects in flood-prone areas to local communities. This chapter successfully determined the regions prone to flooding and assess the effectiveness of machine learning and deep learning techniques in disaster management. The ultimate goal is to enhance emergency response efforts during flood events. This work holds significant potential for governmental and non-governmental organizations in the areas of risk evaluation and reduction during times of crisis, as well as early evacuation and effective preparedness measures.

It is shown that the mapping of flood susceptibility can be enhanced by using machine learning approaches, which is a challenging part of flood modelling. On comparing different flood models it is found an efficient model that improves prediction accuracy. The use of AutoML models and CNN in FSM incorporating different HPO techniques to obtain optimized hyperparameters was also examined. For this, data related to 12 flood-triggering factors were obtained and flood events from 2018 to 2021

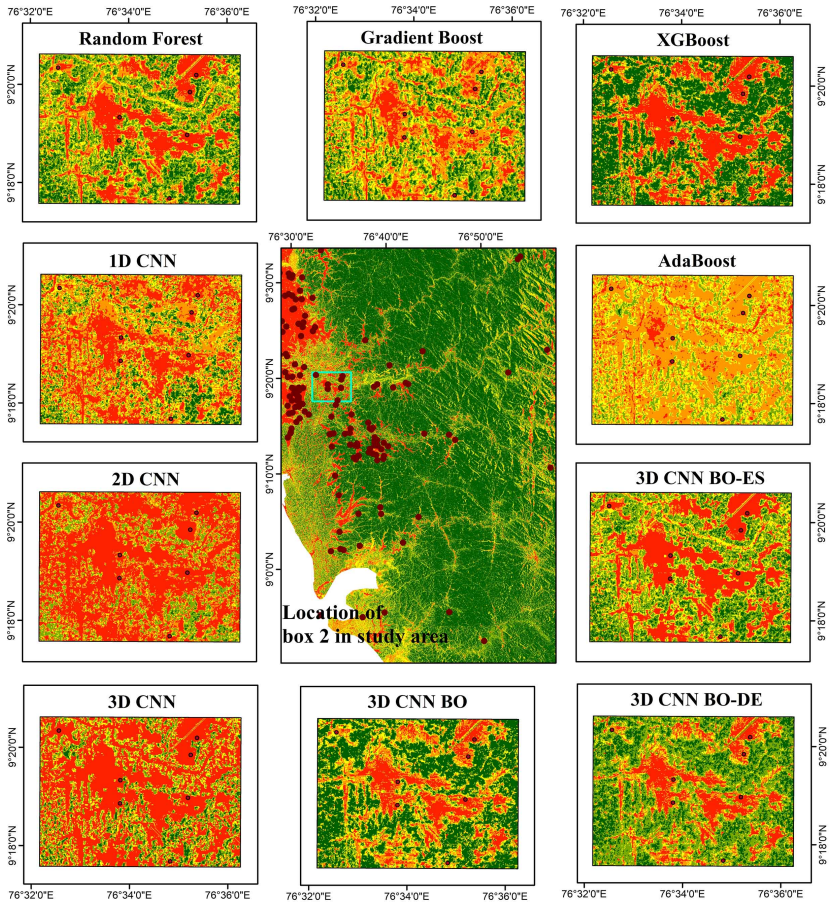

Figure 5.14 Comparison of flood susceptibility maps in detail within box 2.

were considered. For the 3D CNN model, the data was represented in three-dimensional form and was provided as input. BO and its variations were used to optimise the model's hyperparameters. It was learned that the 3D CNN model gave better performance than the AutoML models. Also, BO-CMAES gave better-optimized results than BO and BO-DE. Along with the set of values taken into consideration for tuning, the number of hyperparameters can also be scaled. The aforementioned experiments are run through 20 iterations. The number of iterations can be increased for better outcomes.

Chapter 6

META-HPO

6.1 INTRODUCTION

In a conventional machine learning algorithm, say $f_\theta(x)$, a predictive model learns by solving the equation:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} L(D_{train}; \theta, \omega), \tag{6.1}$$

where L represents the loss function with training data D_{train} and θ indicates the model parameters. ω denotes meta-data, such as information about any hyperparameters (Vilalta et al., 2010; Giraud-Carrier et al., 2004; Lemke et al., 2015).

Instead of assuming that the learning algorithm is fixed and specific, meta-learning attempts to master the learning algorithm itself. The meta-training phase of 'learning how to learn' can be specified as:

$$\omega^* = \underset{\omega}{\operatorname{argmin}} \sum_i L_{meta}(D_{source}; \theta_i^*, \omega). \tag{6.2}$$

During meta-testing, we train the base model on every target task i that was not observed previously using the learned meta-knowledge ω^*:

$$\theta_i^* = \underset{\theta}{\operatorname{argmin}} L_{task}(D_{train-target_i}; \theta, \omega*), \tag{6.3}$$

where L_{meta} represents the outer objective and L_{task} denotes the inner objective. Here ω implies the learning parameters (such as hyperparameters), and θ is the model trained in the outer-level optimization. The model that is trained to excel on the source datasets with parameters ω is represented by $\theta_i^*(\omega)$. $D_{source} = \{D_{source-validation}, D_{source-train}\}$ represents the dataset used for meta-learning which is also the primary task used to learn ω and $D_{target} = \{D_{target-train}, D_{target-test}\}$ represents the individual tasks used to learn θ. We evaluate the performance of the meta-learner by determining the accuracy

θ_i^* of the test split of each target task $D_{target-test}$.

Meta-learning is particularly useful in scenarios where there is limited labeled data. Its adaptability and ability to quickly learn from various tasks make it applicable in diverse domains such as computer vision, natural language processing, anomaly detection and more (Li et al., 2023a; Singh et al., 2021; Mohammadi et al., 2020).

6.2 MATERIAL AND METHODS

The benchmark meta-learning models used in the study are explained in this section (Table 6.1 and Table 6.2). HPOs mentioned in Chapter 3 are used to identify the best set of meta-learning hyperparameters.

6.2.1 Optimization-based Methods

MAML

MAML is a few-shot meta-learning algorithm that is model-agnostic. With the help of meta-learning, MAML can learn the parameters of any standard model and prepare that model for quick adaptation (Finn et al., 2017). The model is first optimised on a set of tasks before being used to quickly adapt to new tasks. By specifically training a model to learn a good initialization for a task, MAML enables a model to achieve good performance on a new task with a small amount of further training. The model's parameters are optimised to minimise the average loss over a collection of tasks, with the restriction that only a small number of samples are provided for training for each task.

Let $D(T)$ represent the task distribution, to which our model should be able to adjust. The model is trained to learn a new task in the k-shot learning environment. The new task T_i is drawn from $D(T)$ from only k samples. We consider a model with the objective function f defined on model parameters θ. MAML updates this θ using one or more gradient descent updates on task T_i using the following equation:

$$\theta_i' = \theta - \alpha \nabla_\theta L_{T_i}(f_\theta). \tag{6.4}$$

The pseudo-code for MAML is given in algorithm 5.

Algorithm 5: Pseudo-code for MAML

Input: Task distribution: D(T)
Hyperparameters: α and β
Initialize θ with random values
Select T_i from $D(T)$
for *all* T_i **do**
    Calculate loss function L for an objective function f_θ: $\nabla_\theta L_{T_i}(f_\theta)$
    With gradient descent compute new parameter θ_i': $\theta_i' = \theta - \alpha\nabla_\theta L_{T_i}(f_\theta)$
Update $\theta \leftarrow \theta - \beta\nabla_\theta \sum_{T_i \sim D(T)} L_{T_i}(f_{\theta'})$

Reptile

Nichol and Schulman (2018) introduced the meta-learning programme Reptile in 2018. Like MAML, Reptile aims to initialise a neural network's parameters such that it may be adjusted with a minimal quantity of data from a new task. Reptile just conducts stochastic gradient descent (SGD) on each task in a typical manner; it does not calculate any second derivatives, in contrast to MAML, which unrolls and differentiates across the computation graph of the gradient descent method. As a result, Reptile uses less memory and processing than MAML. Algorithm 6 represents pseudo-code for Reptile.

Algorithm 6: Pseudo-code for Reptile

Input: Task distribution: D(T)
Hyperparameters: α and β
Number of iterations: n
Initialize θ with random values
for $i \leftarrow 1$ **to** n **do**
    Sample task T_i from $D(T)$ whose loss is L_{T_i}
    Perform $k > 1$ steps of Stochastic Gradient Descent: $\theta_i' = SGD(L_{T_i}, \theta, k)$
    Update $\theta \leftarrow \theta + \varepsilon(\theta_i' - \theta))$

6.2.2 Model-based Methods

SNAIL

The architecture known as SNAIL utilizes a straightforward and adaptable meta-learning approach, incorporating a unique combination of temporal convolutions and soft attention modules. Temporal convolutions combine information from previous experiences, whereas soft attention focuses on particular details.

Type	Model	Year	Paper
Optimization-based	MAML	2017	Finn et al. (2017)
	Reptile	2018	Nichol and Schulman (2018)
Model-based	SNAIL	2018	Mishra et al. (2018)
	CSN	2018	Munkhdalai et al. (2018)
Metric-based	Siamese Network	2015	Koch et al. (2015)
	Matching Network	2016	Vinyals et al. (2016)
	Prototypical Network	2017	Snell et al. (2017)
	Relation Network	2018	Sung et al. (2018)
	RPMN	2020	Xue et al. (2020)
	Deep EMD	2020	Zhang et al. (2020)
	Deep BDC	2022	Xie et al. (2022)

Table 6.1 Timeline of different benchmark meta-learning models used in the study.

Model	Metric	Network architecture
Siamese Neural Network	Euclidean Distance	4 Conv Blocks
Matching Network	Cosine Similarity	4 Conv Blocks
Prototypical Network	Euclidean Distance	4 Conv Blocks
Relation Network	Fully Connected layers	4 Conv Blocks
RPMN	Based on Attention Maps	4 Conv Block
Deep EMD	Earth Mover's Distance	ResNet12
Deep BDC	Brownian Distance	ResNet12

Table 6.2 Details of different metric-based meta-learning models used in the study.

Temporal convolutions generate context, and this context is employed through a causal attention operation. The interleaving of temporal convolution layers with causal attention layers allows SNAIL to have unrestricted high-bandwidth access to its past experiences. By incorporating attention at various stages within an end-to-end trained model, SNAIL learns not only what information to extract from its accumulated experience but also a feature representation conducive to this extraction process.

The SNAIL architecture consists of three blocks. a dense block that uses D and R filters for the dilation rate with single 1D-convolution. A temporal convolution block, involves a series of dense blocks with dilation rates that increase exponentially, until the receptive field surpasses the target sequence length. Finally an attention block which is responsible for a single key-value lookup and concatenating the output with the input (Mishra et al., 2018).

This architectures provide advantages over traditional RNNs like LSTM or GRUs. They are easier to train due to the mitigation of optimization challenges related to temporally-linear hidden state dependencies. Moreover, SNAIL architectures offer efficient implementation, allowing the processing of entire sequences in a single forward pass.

CSN

CSN rapidly adjust their activation values to new data to assist neural networks solve new tasks. For this they use a key-value memory module that has task-specific, additive shifts information obtained using few examples.

CSN model has two parts. One is a base learner and other is a meta learner. The neural network that uses conditional shifts to modify its neurons in order to generate predictions on data is known as the base learner. The meta learner retrieves information from the base learner, calculates conditional shift values, and stores them in its memory. These values are later utilized by the base learner to dynamically adapt to the current task (Munkhdalai et al., 2018).

The fundamental concept behind conditionally shifted neurons is to dynamically alter a network's activation values in real-time, shifting them based on auxiliary conditioning information. The structure of the CSN is expressed as follows for the hidden

layer t and the output layer T:

$$h_t = \begin{cases} \sigma(a_t) + \sigma(\beta_t) & t \neq T, \\ softmax(a_t + \beta_t) & t = T. \end{cases} \tag{6.5}$$

The pre-activation vector at a_t, for a layer with L_t neurons, can assume different forms, depending upon the specific architecture of the network. The nonlinear function σ performs element-wise activation. The vector β_t represents the layer-wise conditional shift and is determined based on conditioning information specific to that layer. Subsequently, the model utilizes this information to generate activation shifts, enabling it to adapt to the specified task, and then stores these shifts in a key-value memory module. And uses these while predicting the unseen data points.

6.2.3 Metric-based Methods

Siamese Network

A Siamese Network is a particular kind of neural network design that consists of two or more neural networks that are identical to one another, share the same set of weights, and are trained to carry out a certain task (Koch et al., 2015). Siamese Network is used for tasks that involve comparing two inputs and determining their similarity or dissimilarity. The two identical neural networks are trained to produce similar outputs for similar inputs and dissimilar outputs for dissimilar inputs.

Let the network have N fully connected layers and $h_{1,l}$ represents the hidden vector in layer l for the first twin and $h_{2,l}$ that of the second twin. A distance function d is used to compare the features determined by the twins at $(L-1)$th layer.

$$d = \sigma\big(\sum_i \alpha_i |h^i_{1,l} - h^i_{2,l}|\big), \tag{6.6}$$

where σ is the sigmoidal activation function. The last layer L, calculates a similarity score for the two feature vectors using the features from $(L-1)$th hidden layer. α is a parameter learned by the model during training, which weighs the significance of the component-wise distance.

Matching Network

Matching Network (matching nets) is a metric-based meta-learning framework that learns a network that uses a small labelled support set and maps an unlabelled query

set to its label. Vinyals et al. (2016) introduced Matching Network, which apply an attention mechanism over a trained embedding of the support set to predict classes of the query set. Matching Network are similar to a weighted nearest-neighbor classifier. It trains on sampled mini-batches of data known as episodes, where each episode is created by subsampling classes and data points to replicate the few-shot job. The introduction of episodes enhances generalisation by bringing the training problem closer to the test setting.

The basic idea of Matching Network is learning a metric space where similarity or dissimilarity between examples can be measured (Vinyals et al., 2016). The framework consists of an embedding network and a matching module, with the embedding network transforming inputs into lower-dimensional representations. The matching module computes a matching score between a query example and support examples. The score can be computed using various methods, such as cosine similarity, Euclidean distance, or more complex attention mechanisms. During training, the network minimizes a loss function to encourage high scores for similar examples and low scores for dissimilar ones.

The model maps support set S that has input-label pairs of a small number of samples (say, k) to a classifier C. The classifier maps $\hat{a}$, test sample to a probability distribution over output labels $\hat{b}$.

$$S = \{(a_i, b_i)\}_{i=1}^{k}. \tag{6.7}$$

The mapping from $S \rightarrow C$ is defined to be $P(\hat{b}|\hat{a}, S)$. P is defined by a neural network. And for a new support set S', we use P to make prediction for $\hat{a}$ from $\hat{b}$, i.e., $P(\hat{b}|\hat{a}, S')$. The probability is computed using the following formula:

$$P(\hat{b}|\hat{a}, S) = \sum_{i=1}^{k} A(\hat{a}, a_i) b_i, \tag{6.8}$$

where A is the attention mechanism used by the model. The most commonly used attention module is a softmax function over cosine distance d.

$$A(\hat{a}, a_i) = \frac{e^{d(f(\hat{a}), g(a_i))}}{\sum_{j=1}^{k} e^{d(f(\hat{a}), g(a_j)))}}, \tag{6.9}$$

where f and g are appropriate neural networks (embedding functions). Cosine distance d for any two vectors X and Y (with angle ϕ between them) is defined as:

$$d = \cos\phi = \frac{X.Y}{||X||\ ||Y||}. \tag{6.10}$$

Prototypical Network

With few shot learning overfitting becomes a concern due to the small dataset size and the likelihood that the model will memorise the training examples rather than generalising successfully to new unseen data. Snell et al. (2017) proposed Prototypical Network aiming to address this issue. Prototypical Networks are based on the hypothesis that there is an embedding in which each class possess an individual prototype representation and points belonging to that class group around this prototype representation. For every training set class, this “prototype" representation has to be learned using Prototypical Network. The network learns about the metric space suitable for classification by calculating the distances between prototype representations of each class. The feature vectors derived from all instances of that particular class in the training set are simply averaged to create this prototype. Once they learn the representation, the model categorises the embedded query point by locating the nearest class prototype.

Prototypical Network can be seen as similar to clustering, where the class means are represented as prototypes and the distances are calculated using a Bregman divergence, such as squared Euclidean distance (Snell et al., 2017).

For a classification problem with k classes, a prototype can be represented by p_k as follows:

$$p_k = \frac{1}{|S_k|} \sum_{(a_i,b_i)\in S_k} f_\theta(a_i), \tag{6.11}$$

where S_k is the support set, f_θ is the embedding function with learning parameters θ. The feature vector of i th sample is denoted by a_i and b_i represents the corresponding label.

The model computes the probability distribution over the distance between each class’s prototypes and the query instance during testing. After that, the class with the closest prototype to the query instance is given the query instance. The distance function used in Prototypical Network is typically the Euclidean distance, but other distance metrics such as cosine similarity can also be used. For a given distance function d, the network calculates the probability distribution, P_θ and determines the label for a query point a.

$$P_\theta(b = k|a) = \frac{e^(-d(f_\theta(a), p_k)}{\sum_{k'} e^{-d(f_\theta(a), p_{k'})}}. \tag{6.12}$$

Further, the model learns by minimizing the negative log probability $-ln(P_\theta(b = k|a))$ of the true class k using SGD.

Relation Network

A Relation Network computes relationships between different elements in a given task or problem. These relationships can be defined in various ways, such as pairwise interactions, graph structures, or other forms of dependencies. In few-shot learning, Relation Network are used to compare support and query examples and make predictions based on the learned relationships (Sung et al., 2018). The architecture of a Relation Network typically consists of two main components: a feature extractor (embedding module), f and a relation module r. The feature extractor processes the input data (e.g., images, sequences, graphs) and maps it into a lower-dimensional feature space. Suppose we have a_i from sample set S and a_j from the query set. We give them as inputs to the embedding module and it generates feature maps $f(a_i)$ and $f(a_j)$. These feature maps are then concatenated $C(f(a_i), f(a_j))$ and fed into the relation module, which in turn computes the pairwise relations between them. This similarity is called a relation score whose value lies in the range of 0 to 1.

$$score_{i,j} = \sum_{i=1}^{k} r(C(f(a_i), f(a_j)), \tag{6.13}$$

where $i = 1ton$ for a n-way one-shot learning. For k shot problems where $k > 1$, the element-wise sum over the embedding module outputs of all samples from each training class is considered as the feature map of that class. The generated feature maps of all classes are pooled and then concatenated to the feature map of the query image. This is then fed into the relation module. The model learns by minimizing the mean squared error loss.

$$MSE = \sum_{i=1}^{m} \sum_{j=1} n(score_{i,j} - \mathbf{1}(b_i == b_j))^2, \tag{6.14}$$

where $-\mathbf{1}$ is a conditional operation. If $x == y$ is true, then $-\mathbf{1}(x == y)$ evaluates to -1 and if $x == y$ is false, evaluates to 0. Hence matched pairs will have a similarity of 1 and the mismatched pair will have a similarity of 0.

RPMN

Metric learning and the attention mechanism form the foundation of Relative Position and Map Network (RPMN). There are two modules in it: Relative Position Network (RPN) and Relative Map Network (RMN). RPN compares various pairings of training and query images using an attention technique to determine their inherent correspondences. RMN uses the attention maps derived from RPN to calculate the similarity between the query and training images. RPN determines which positions is important for model to compare, while RMN compares each of these images separately (Xue et al., 2020).

RPN generates weights for different positions in feature maps. Based on attention mechanism RPN determines which positions are important for model to compare. Samples x^S and x^Q are drawn from the query set (Q) and support set (S), respectively. F_{x^S} and F_{x^Q} are their feature maps. $V^S_{i,j}$ is the position vector of ith row and jth column in the feature map F_{x^S}. Likewise, there exists a position vector in the query set. An encoder referred as $H()$ converts the support and query position vectors $V^S_{i,j}, V^Q_{i,j}$ into a relative position vector by concatenating them.

$$V^{s,q}_{i,j} = H([V^S_{i,j}, V^Q_{i,j}]). \tag{6.15}$$

Now feature map of the query set is updated as follows:

$$F_{x^Q} := F_{x^Q} + Att * F_{x^Q}, \tag{6.16}$$

where $*$ indicates element-wise production operation. Att represents a relative position score that is calculated using $V^{s,q}_{i,j}$ for each i, j.

In the embedding model, distance between feature maps $F^i_{x^S}$ and $F^i_{x^Q}$ are learned. The RMN uses a single full-connected layer to calculate the weighted sum $W_{S,Q}$ of each and every output; this is the final similarity score between F_{x^S} and F_{x^Q}.

Deep EMD

The Earth Mover's Distance (EMD) is a measure designed to calculate the distance between structured representations. Initially it was used for image retrieval purposes. Within the Deep EMD model, EMD is employed as a metric to calculate the structural distance between image representations that are dense, which helps determine the

significance of an image. The best matching flows between structural elements are produced using EMD, minimizing matching costs. This information is then utilized to represent image distances for classification purposes. To derive the crucial weights for elements in the EMD formula, Deep EMD model uses a cross-reference mechanism between query and support image. This mechanism proves effective in minimizing the influence of cluttered backgrounds and significant intra-class appearance variations (Zhang et al., 2020).

For the implementation of k-shot classification, the model uses a structured fully connected layer that is intended to use EMD for direct classification of dense image representations. The distance is determined by the following equation:

$$EMD(Q,S) = \sum_{(i,j)} x_{ij}(1 - c_{ij}), \tag{6.17}$$

where x_{ij} represents the vector of optimal matching flows and c_{ij} represents the matching cost between vector u_i andv_j. The variables u_i and v_j belong to U and V, respectively. U and V represents the feature maps of the query image Q and the support image S respectively.

Deep BDC

The Brownian Distance Covariance (BDC) metric is characterized as the Euclidean distance between the joint characteristic function and the product of the marginals. This metric inherently measures the inter-dependency between two random variables. In case of discrete features, the BDC is decomposed, allowing us to formulate BDC as a pooling layer. Effortless integration into a deep network is achieved by incorporating this layer, where feature maps serve as input, and the output is a BDC matrix representing an image. And the inner product between the BDC matrices of two images is used to calculate how similar they are (Xie et al., 2022).

The Deep BDC model is initiated with a Protonet module, aiming to learn a metric space. This metric space facilitates classification by evaluating distances to the prototype of each class. For a specific task $(T_{support}, T_{query})$, we input the image x_i into the network to generate the BDC matrix $B(x_i)$. The prototype of the support class k is

calculated as the average of the BDC matrices associated with that class:

$$P_k = \frac{1}{K} \sum_{(x_i, y_i) \in S_k} BDC(x_i), \tag{6.18}$$

where the collection of examples in $T_{support}$ labelled with class k is denoted by S_k. We generate a class distribution using a softmax over the distances to the support classes' prototypes.

6.3 RESULTS AND DISCUSSION

Franceschi et al. (2018) introduced a framework grounded in bilevel programming, which integrates gradient-based hyperparameter optimization and meta-learning. We put forward in this chapter a bilevel optimization framework that incorporates both meta-learning and hyperparameter optimization to solve image classification problems. We implement multiple sets of experiments to evaluate the effectiveness these frameworks. All experiments are conducted on the Omniglot and miniImageNet dataset. An overall workflow of the experiments conducted in the study is given in Figure 6.1.

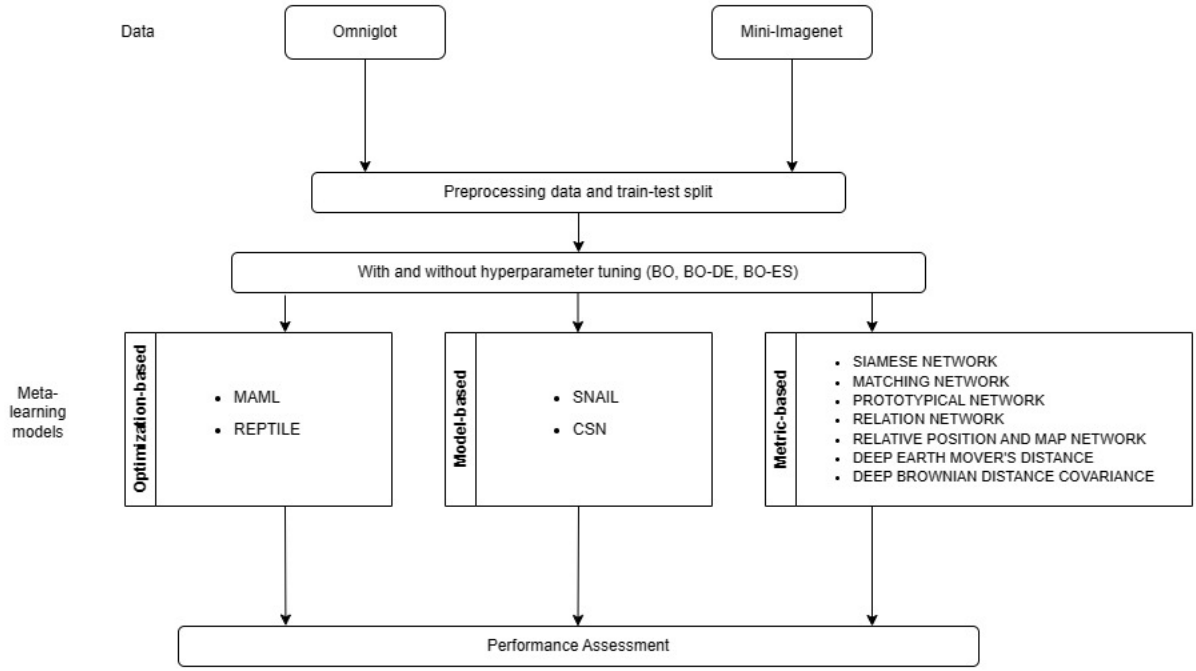

Figure 6.1 A comprehensive overview of the experimental methodology followed in the study.

Hyperparameter	Range
Inner and outer learning rates	[0.0001 - 1.0]
Learning rate decay factor	[0.0000001 - 0.1]
Learning rate after number of epochs	[2 - 10]
Weight decay	[0.00001 - 0.1]
Optimizer	[Adam, RMSprop, Adagrad, SGD]

Table 6.3 List of hyperparameter values of meta-learning models used for tuning.

Model	Omniglot 5-way		miniImageNet 5-way	
	1-shot	5-shot	1-shot	5-shot
MAML	79.01 ± 1.21	83.73 ± 1.83	30.41 ± 2.33	43.44 ± 1.60
Reptile	76.99 ± 2.52	84.31 ± 1.76	30.78 ± 2.24	47.68 ± 1.32
SNAIL	80.07 ± 0.37	81.55 ± 0.54	35.66 ± 0.29	49.64 ± 0.43
CSN	79.42 ± 0.81	81.14 ± 0.32	36.73 ± 0.25	54.7 ± 0.56
Prototipical Network	78.8 ± 0.44	81.47 ± 0.65	31.37 ± 0.33	48.56 ± 0.73
Relation Network	80.13 ± 0.26	81.59 ± 0.32	32.19 ± 0.45	46.08 ± 0.56

Table 6.4 Test accuracies of metric-based meta-learning models on Omniglot and miniImageNet **without HPO**.

Setup 1:

In the first set of experiments we consider 6 benchmark meta learning models. This includes MAML, Reptile, SNAIL, MANN, Prototypical and Relational Networks. MAML and Reptile are optimization-based meta-learning models. SNAIL and MANN are model-based meta-learners. Prorotypical and Relation Network are metric-based meta learning models. The set of hyperparameters and the range of values they can take are listed in Table 6.3.

An n-way k-shot learning challenge makes up the basis for each of our tests. A set of k-labelled examples from each of the n classes that have not yet been trained upon are provided for each approach. The aim is to categorize a disjointed group of unlabeled instances into one of these n classes.

We conducted 5-way 1-shot and 5-way 5-shot experiments on Omniglot (Lake et al., 2011) and ImageNet datasets (Russakovsky et al., 2015). Omniglot contains handwritten characters from a wide range of alphabets. The dataset contains 1,623 different

Model	Omniglot 5-way		miniImageNet 5-way	
	1-shot	5-shot	1-shot	5-shot
MAML	75.7 ± 1.97	79.67 ± 0.78	21.95 ± 0.94	38.87 ± 1.46
Reptile	69.68 ± 2.30	76.25 ± 0.56	24.32 ± 1.62	45.75 ± 1.52
SNAIL	81.38 ± 1.23	81.61 ± 0.65	34.4 ± 0.48	50.57 ± 0.45
CSN	76.73 ± 0.98	78.2 ± 0.86	37.19 ± 0.84	50.27 ± 0.87
Prototipical Network	82.11 ± 0.95	84.53± 0.33	34.83± 0.43	51.49 ± 0.51
Relation Network	83.91 ± 0.31	84.86 ± 0.23	35.93 ± 0.68	52.01 ± 0.34

Table 6.5 Test accuracies of metric-based meta-learning models on Omniglot and miniImageNet **with BO**.

handwritten characters from 50 different alphabets with 20 samples for each character.

For the ImageNet ILSVRC-2012 dataset, we build a miniImageNet version based on the splits proposed by Ravi and Larochelle (2016). The miniImageNet data has 60000 colour images belonging to 100 classes each of size 84 x 84 images of size 84 x 84. Among these, 80 classes were designated for training, while 20 were allocated for testing.

The meta-training and meta-testing sets share the episodic training strategy that facilitates generalization ability across tasks. The episodic training ensures consistency between meta-training and meta-testing, which is crucial for the meta-learning methods. Results are obtained over 300 test episodes with 95% confidence intervals.

Table 6.4 gives the test accuracies of all the meta learning models. Table 6.5 shows the test accuracies obtained for all models assisted with HPO. The results are obtained using the Wilcoxon Rank-Sum test on samples from 100 iterations with a statistical significance of $p<0.05$. If we compare the tables, it can be observed that there is an increase in accuracy for both the metric based models when BO was used for hyperparameter optimization. That is, Prototypical and Relational Networks performed better with their hyperparameters optimized with BO.

Metric-based approaches tend to be more straightforward to implement than optimization-based or model-based alternatives. They embrace a more generic and flexible methodology, avoiding strong assumptions about the underlying model. This adaptability proves advantageous in various settings. Consequently, when integrated with hyperparame-

ter optimization, metric-based approaches could improve the model easily with limited number of epochs.

Model-based and optimization-based approaches are inherently complex, and the inclusion of a hyperparameter optimization module further increases the algorithm's complexity. This complexity poses challenges in identifying the optimal hyperparameter settings, leading to lower performance compared to the default configurations.

Setup 2:

In this section, we present the findings of numerous *n*-way *k*-shot tests on 7 metric-based models with the inclusion of different HPOs. The models used are Siamese Network, Matching Network, Prototypical Network, Relation Network, Relative Position and Map Network (RPMN), Deep Earth Mover's Distance (Deep EMD) and Deep Brownian Distance Covariance (Deep BDC). The optimization algorithms compared are Bayesian optimization, BO-DE and BO-ES.

There are 8 set of experiments in this section. This includes 5-way 1-shot, 5-way 5-shot, 10-way 1-shot and 10-way 5-shot experiments on Omniglot and ImageNet datasets. The results for Omniglot are given in Figure 6.2, Figure 6.3, Figure 6.4 and Figure 6.5. And results for miniImagenet are provided in Table 6.6, Table 6.7, Table 6.8 and Table 6.9. The results are derived from the Friedman test conducted on samples from 100 iterations, with statistical significance $p<0.05$.

Network Architecture

For the Siamese, Matching and Prototypical and Relation Network, the basic experiment setup is similar to the one used by Vinyals et al. (2016). We select examples from the *n* unseen character class independent of the alphabet. Let this be denoted as S (the support set), which is a subset of the total dataset D. The results are evaluated on batch B (the query set) which is also a subset of D. The Matching Network uses an attention mechanism over a learned embedding of the labelled S to predict classes for the unlabeled.

Omniglot data is enhanced by introducing random rotations in increments of 90 degrees. The training sample consisted of 1200 characters and the remaining character

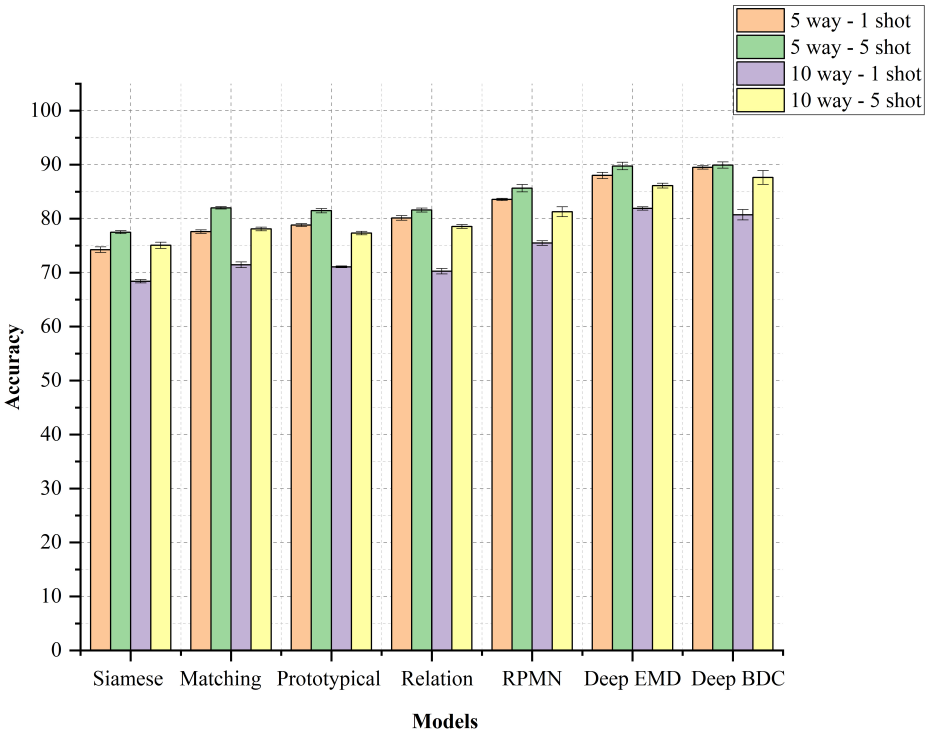

Figure 6.2 Test accuracies of metric-based meta-learning models on **Omniglot without HPO**.

classes were for evaluation. All images are resized to shape 28 x 28.

The embedding CNN network is made of four 3 × 3 convolution layers with 64 filters. Following each convolutional layer, there is batch normalization, a Rectified Linear Unit (ReLU), and 2 × 2 max-pooling along with dropouts. The final feature map is 1 × 1 × 64 in size and is followed by a softmax non-linearity. For miniImagenet we used the same configuration as Omniglot.

In the Prototypical Network setup, the grayscale images of Omniglot are resized to 28 x 28. The character classes are enhanced with rotations in multiples of 90 degrees like in the matching nets configuration. The number of characters used for training and testing are also the same (1200 for training remaining for testing). The embedding structure, which consists of four convolutional blocks, is the same as that used in the matching nets. It produced a 64-dimensional output for the 28 x 28 Omniglot images.

For miniImagenet dataset, we employ the same dataset splits as mentioned in Vinyals et al. (2016). We employ the same four-block embedding architecture as used for Omniglot, but because the images are larger, the output space is 1600 dimensions instead.

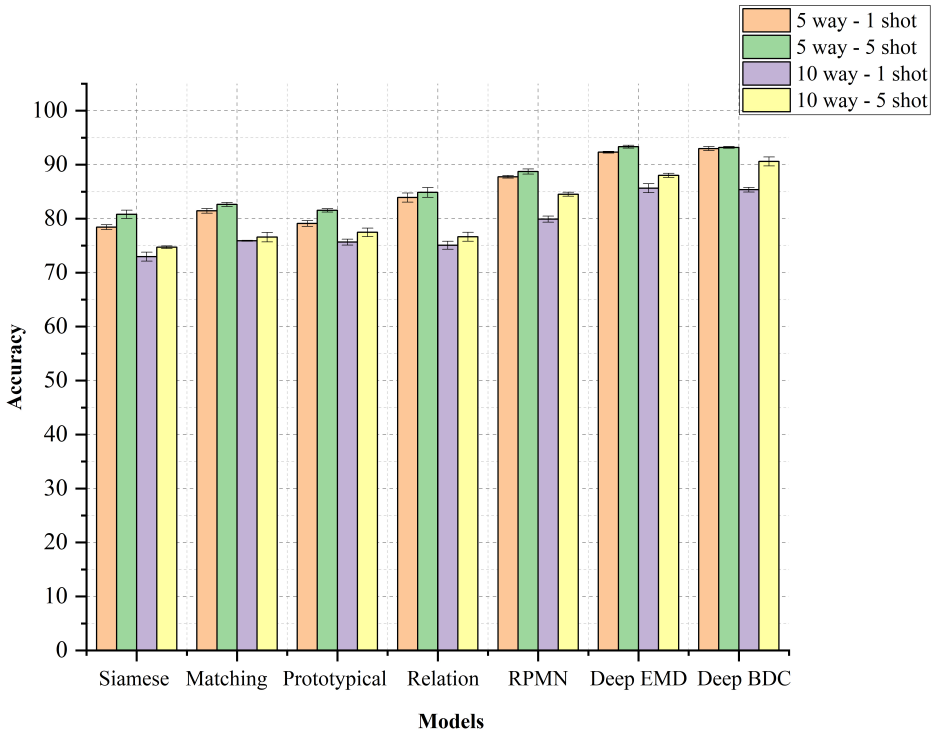

Figure 6.3 Test accuracies of metric-based meta-learning models on **Omniglot** with **BO**.

For Relation Network, again we use the same embedding module. The relation module consists of two convolutional blocks and two fully-connected layers. Each of convolutional block is a 3 x 3 convolution with 64 filters followed by batch normalisation, ReLU non-linearity and 2 x 2 max-pooling. The output size of last max pooling layer is $H = 64$ and $H = 64 * 3 * 3 = 576$ for Omniglot and miniImageNet respectively. The two fully-connected layers are 8 and 1 dimensional, respectively. All fully-connected layers are ReLU except the output layer is Sigmoid in order to generate relation scores in a reasonable range for all versions of the network architecture.

The RMN module in RPMN use the combination of a 4 convolution layer with 3 × 3 kernel without padding, a batch norm layer and a ReLU function, and two hidden layers for full connection layers.

In Deep BDC and Deep EMD the backbone network is ResNet12. The input resolution of images is 84×84 for ResNet12. Rest of the setup remains the same. For every experimental setting, the hyperparameters for each meta-learning technique have been tuned separately.

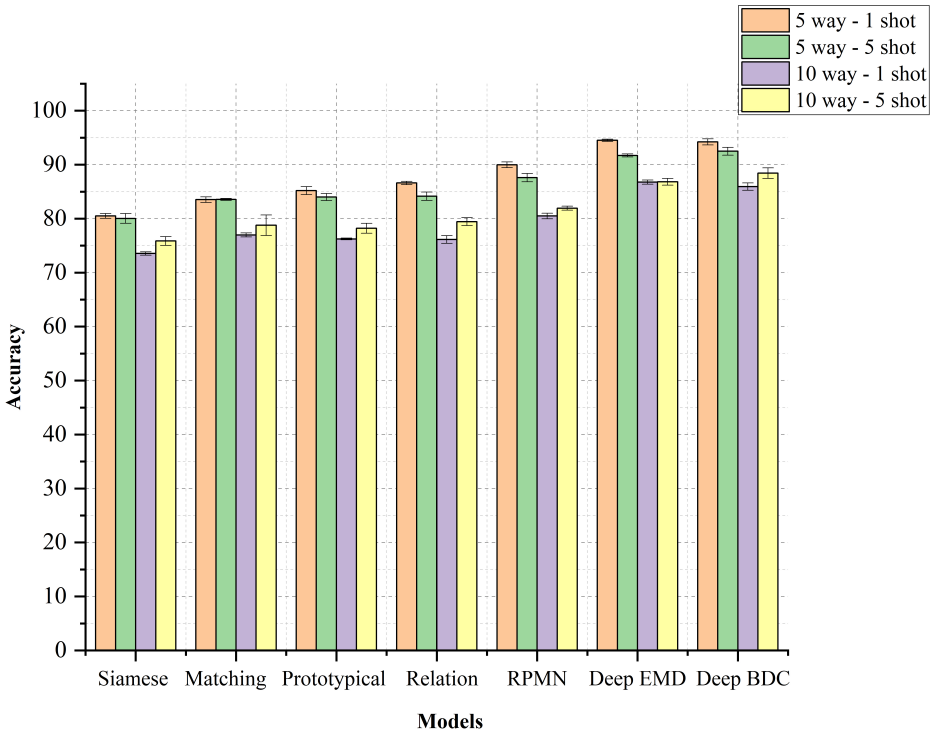

Figure 6.4 Test accuracies of metric-based meta-learning models on **Omniglot** with **BO-DE**.

Comparing n and k

In general, using greater k in k-shot, helps all models perform better. With additional examples all models (both 10-way and 5-way), improved in terms of test accuracy. This can be observed from all the tables. Also, across all settings, it is observed that with greater n, the test accuracy went down. That is performance tends to drop as the configuration gets more complex.

Comparing the datasets

In comparison to Omniglot, miniImageNet has a wider range of object categories and more diverse images, making it a more intricate dataset. The increased diversity and complexity of miniImageNet can hamper the network's ability to generalize effectively. This is the reason why typically the models exhibit lower performance on miniImageNet when compared to Omniglot.

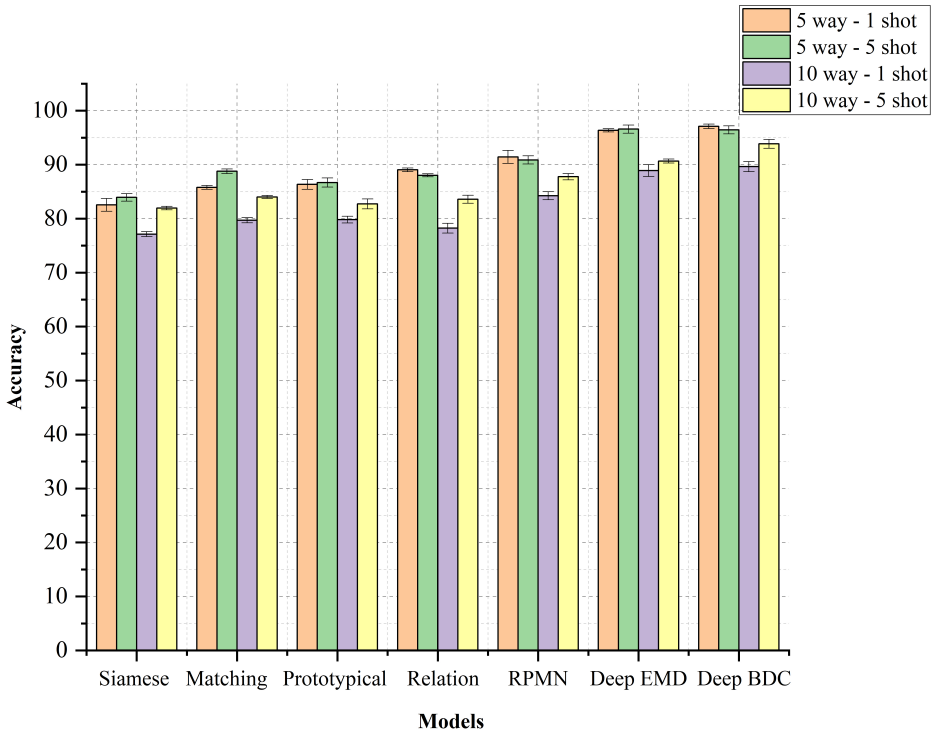

Figure 6.5 Test accuracies of metric-based meta-learning models on **Omniglot** with **BO-ES**.

Comparing the metric

Siamese Network performs poorly compared to other meta learning models for both datasets. This can be because of insufficient model capacity to handle the complexity of the task compared to the other architectures. The underlying architecture of Siamese Network learns a shared representation for pairs of samples. They might lack the necessary complexity required to generalize for certain tasks. In contrast, Matching Network might excel in handling the complexity of those tasks.

Prototypical and Matching Network obtained almost similar accuracies in most cases. Prototypical Network was proposed for the problem of few-shot classification and it obtained better state-of-the-art results for miniImageNet and Omniglot datasets compared to Matching Network. However, the results were compared for a larger number of epochs (10,000 epochs). In our experiment, we use only 100 epochs for training both datasets. It is observed that with a smaller number of epochs Matching Network converge faster compared to Prototypical Network.

Relation Network yielded better outcomes in comparison to all the above mentioned

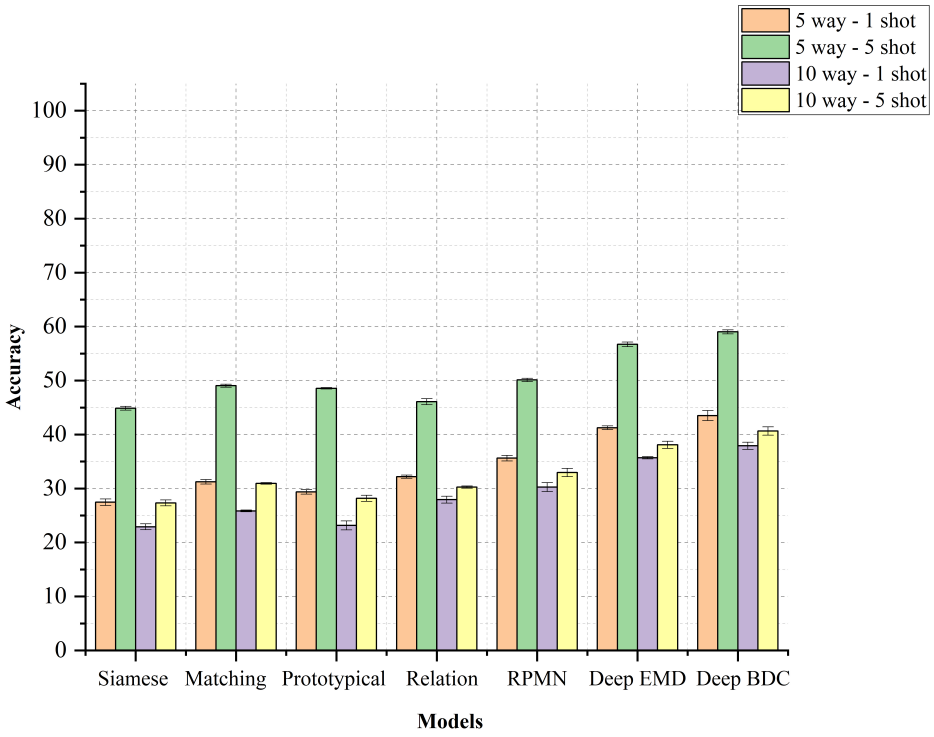

Figure 6.6 Test accuracies of metric-based meta-learning models on **miniImageNet without HPO**.

models. Relation Network are great at capturing intricate relationships between input pairs, especially in tasks that require a deep understanding of pairwise relationships. On the other hand, Siamese Network, Matching Network and Prototypical Network may struggle to capture higher-order relationships due to inherent architectural limitations. The flexibility of Relation Network allows easy adaptation to diverse tasks by adjusting the input representations and relation functions.

With two modules Relative Position Network (RPN) and Relative Map Network (RMN), Relative Position and Map Network (RPMN) works better than Relation Network. The RPN module identifies crucial positions for model comparison, while the RMN module independently compares the images at these identified positions. The attention mechanism within the RPN modules enhances its ability to generate a more effective similarity measure, leading to superior overall model performance.

Deep EMD and Deep BDC gave better results than all other metric models. Earth's Movers Distance is employed by Deep EMD as a metric to calculate the structural distance between dense image representations. Deep BDC uses Brownian Distance Co-

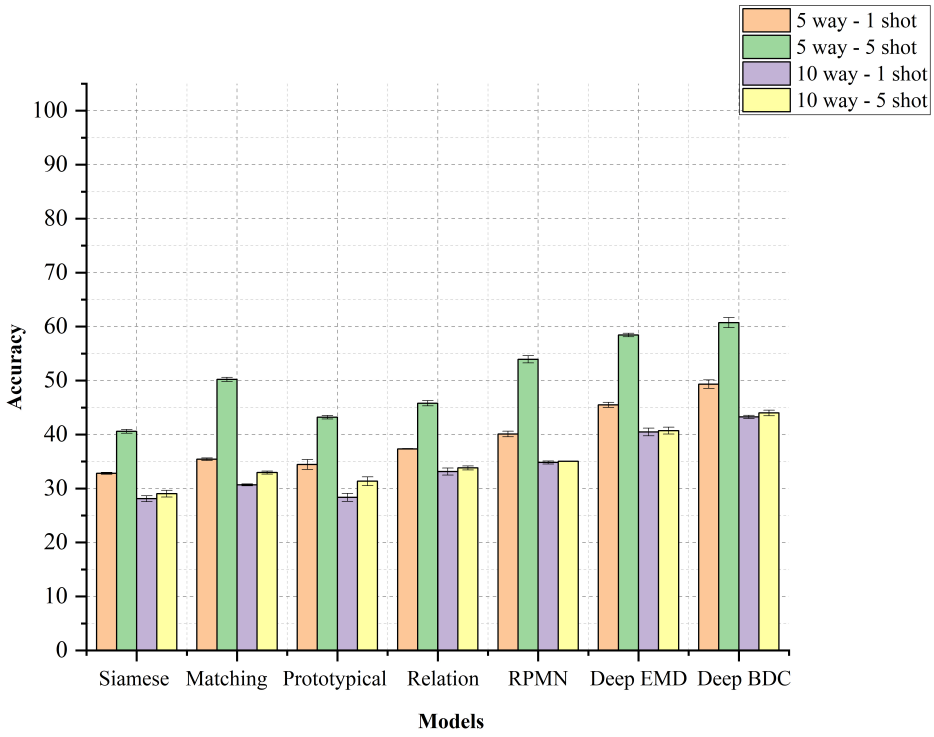

Figure 6.7 Test accuracies of metric-based meta-learning models on **miniImageNet** with **BO**.

variance metric to measures the dependency between two random variables. These two distance metric are efficient in finding similarirty between support and query images compared to other metrics.

EMD can be considered as the minimum amount of work needed to reshape one distribution of pixels into the other, treating pixel values as mass. EMD has the ability to withstand variations in localized changes within distributions. This characteristic makes it apt for comparing images with similar structures or objects exhibiting local variations.

BDC is a measure of dependence between two random variables or distributions. In image comparison, it can capture how changes in one part of an image are related to changes in another part. BDC demonstrates greater sensitivity to nonlinear relationships. In image comparison, it serves as an advantage when trying to capture complex and non-linear relationships between pixel values.

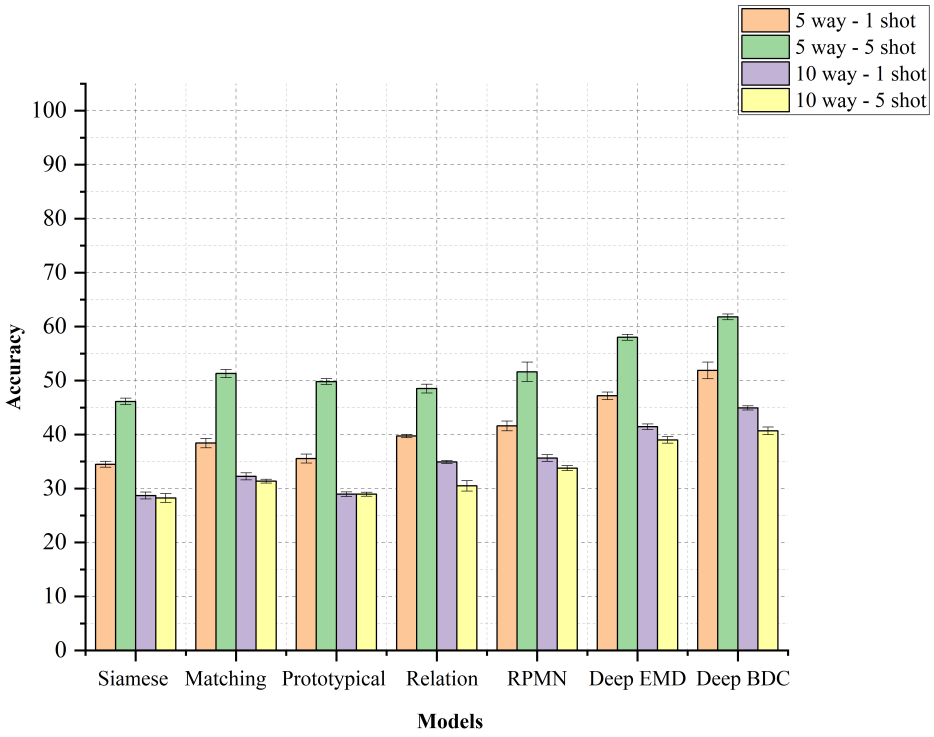

Figure 6.8 Test accuracies of metric-based meta-learning models on **miniImageNet** with **BO-DE**.

Comparing HPO

When compared to settings without HPO, accuracy with BO increased for most instances. If we compare Figure 6.3 and Figure 6.4, and also Figure 6.6 and Figure 6.7 we can see that the accuracy has gone higher for most cases. Especially for the 1-shot results there has been an evident increase. BO is known for being data-efficient. BO actively balances exploration and exploitation, making it efficient in settings where the number of function evaluations is restricted. Therefore in 1-shot cases, BO perform better as only fewer evaluations are required to make informed decisions.

BO-DE performed better than BO in multiple scenarios. Upon comparing Figure 6.3 with Figure 6.4 and Figure 6.7 with Figure 6.8, it becomes evident that in the majority of 1-shot cases, BO-DE exhibits better performance than BO. However, in 5-shot scenarios, BO achieves comparable or lower accuracies.

BO-ES, however, outperformed BO and BO-DE in most cases. That is, BO-ES could find the optimal hyperparameter better than BO and BO-DE. Figure 6.5 and Figure 6.9 gives the results from BO-ES. ES is known for its flexibility and adaptability. It

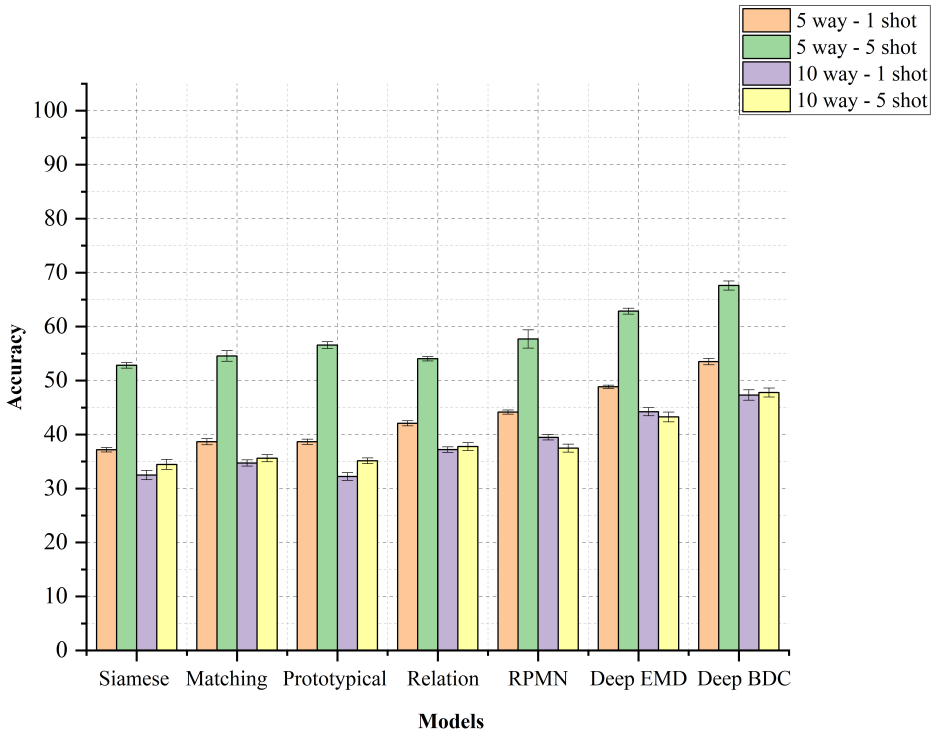

Figure 6.9 Test accuracies of metric-based meta-learning models on **miniImageNet** with **BO-ES**.

can easily handle noisy and stochastic objective functions. ES is also less sensitive to the choice of initial hyperparameter values compared to other optimization methods.

In meta-learning, models aim to generalize across various tasks, and each task may have different optimal hyperparameter configurations. A "one-size-fits-all" hyperparameter setting may not be suitable in this case. Meaning using the same set of hyperparameters for all tasks is not optimal in meta-learning. However, since tasks in meta-learning can vary significantly, having an optimization method that adapts and searches for task-specific hyperparameters is beneficial. Out of the three hyperparameter optimization models we assessed, BO-ES distinguishes itself with remarkable flexibility and resilience. Our findings demonstrate that this adaptability is advantageous in the context of meta-learning, where opting for a uniform hyperparameter setting may not be the most effective strategy.

6.4 CLOSING REMARKS

This work analyses the application and importance of hyperparameter optimization-assisted meta-learning in image classification problems. An ablation study was carried out to determine the effectiveness of HPO techniques in meta-learning. We used the meta-learning benchmark datasets - miniImageNet and Omniglot datasets for this purpose. Main conclusions drawn from the study after analysing different hyperparameter optimization techniques on meta-learning benchmark models are:

- Meta-learning involves generalizing across diverse tasks, each potentially requiring different optimal hyperparameter configurations. Our findings highlight the importance of adapting hyperparameters to individual tasks rather than employing a uniform setting across all tasks.
- The study advances the importance of HPO within meta-learning and underscores the significance of employing methods like BO-ES to achieve optimal performance. BO-ES's ability to adapt and search for task-specific hyperparameters distinguishes it as the most suitable option for meta-learning contexts.

The algorithm used for HPO is scalable and can be employed to optimize multiple hyperparameters as required. As future work, we intend to use these meta-learning models combined with HPO for image classification on real-time data.

Chapter 7

CONCLUSION AND FUTURE WORKS

Hyperparameters of ML models must be tuned to fit particular datasets before being deployed to practical problems. However, because the volume of created data has substantially grown in practice, and manually setting hyperparameters is tremendously resource-intensive, it has become critical to optimise hyperparameters automatically. The efficacy of AutoML systems in handling large-scale image classification datasets has been examined. Additionally, the performance of SMAC and DEHB packages for hyperparameter optimization was analysed. Out of the proposed HPO frameworks, it was observed that the efficiency of BO-CMAES proves it to be worth adopting in AutoML systems. The results from this work can be extended to other models and datasets.

These HPO algorithms were put into use for LULC classification using Sentinel-2 images from 2019 and 2023. LULC map of the city of Ernakulam under the jurisdiction of the state of Kerala in India is generated to identify waterbody, barren land, builtup, sparse vegetation and dense vegetation regions. In particular, random forest aided by an evolutionary-based hyperparameter optimization model was used to classify the land use land cover pattern. Without HPO, the RF model achieved an accuracy of 90.13% and a Kappa of 0.88 for the 2019 LULC map, and 90.85% accuracy and 0.89 Kappa for the 2023 map. The BO-ES method enhanced RF's performance to 92.00% accuracy and 0.90 Kappa for 2019, and 92.80% accuracy and 0.91 Kappa for 2023.

The HPO techniques proposed were also used along machine learning models to build a flood susceptibility map in Kerala, India. Particularly, a three-dimensional CNN architecture was used for generating FSM. The CNN model was assisted with hyperparameter optimization techniques that combine Bayesian optimization with evolution-

ary algorithms like differential evolution and covariance matrix adaptation evolutionary strategies. The performances of all models were compared in terms of cross-entropy loss, accuracy, AUC and kappa score. The CNN model showed better performance than the AutoML models. Evolutionary algorithm-assisted hyperparameter optimization methods improved the efficiency of the CNN model by 4 and 9 percent in terms of accuracy and by 0.0265 and 0.0497 with reference to the AUC score.

The study also explores metric-based meta-learning models and examine hyperparameter optimization techniques and their potential to boost the performance of metric-based models. Bayesian optimization and one of the proposed variants were employed to tune the hyperparameters of these models on Omniglot and ImageNet datasets. Computational experiments were carried out to compare the impact of different hyperparameter optimization algorithms on these benchmark models. In summary, the thesis investigates the impact of HPO on machine learning models employed in image classification problems.

It is critical to ensure that HPO methods can scale efficiently. Future work can focus on developing adaptive HPO techniques that can dynamically adjust to varying data sizes and computational resources. Another future direction is extending the application of HPO-assisted ML models to various domains beyond image classification, such as natural language processing, time series analysis, and bioinformatics, can provide a broader understanding of their capabilities and limitations. This would involve adapting and fine-tuning HPO methods to meet the specific needs of different types of data and tasks.

Bibliography

Abedi, R., Costache, R., Shafizadeh-Moghadam, H., and Pham, Q. B. (2022). Flash-flood susceptibility mapping based on xgboost, random forest and boosted regression trees. *Geocarto International*, 37(19), 5479–5496.

Abijith, D. and Saravanan, S. (2022). Assessment of land use and land cover change detection and prediction using remote sensing and ca markov in the northern coastal districts of tamil nadu, india. *Environmental Science and Pollution Research*, 29(57), 86055–86067.

Abijith, D., Saravanan, S., Singh, L., et al. (2020). Gis-based multi-criteria analysis for identification of potential groundwater recharge zones-a case study from ponnaniyaru watershed, tamil nadu, india. *HydroResearch*, 3, 1–14.

Ahmadlou, M., Al-Fugara, A., Al-Shabeeb, A. R., et al. (2020). Flood susceptibility mapping and assessment using a novel deep learning model combining multilayer perceptron and autoencoder neural networks. *Journal of Flood Risk Management*, 14(1), e12683.

Ahmadlou, M., Ghajari, Y. E., and Karimi, M. (2022). Enhanced classification and regression tree (CART) by Genetic Algorithm (GA) and Grid Search (GS) for flood susceptibility mapping and assessment. *Geocarto International*, 37(26), 13638–13657.

Aissia, M.-A. B., Chebana, F., Ouarda, T. B., et al. (2012). Multivariate analysis of flood characteristics in a climate change context of the watershed of the baskatong reservoir, province of québec, canada. *Hydrological Processes*, 26(1), 130–142.

Akiba, T., Sano, S., Yanase, T., et al. (2019). Optuna: A Next-generation Hyperparam-

eter Optimization Framework. In *Proceedings of the 25th International Conference on Knowledge Discovery & Data Mining*, page 2623–2631.

ALGorain, F. T. and Clark, J. A. (2022). Bayesian Hyper-Parameter Optimisation for Malware Detection. *Electronics*, 11(10), 1640.

Álvarez-Cabria, M., Barquín, J., and Peñas, F. J. (2016). Modelling the spatial and seasonal variability of water quality for entire river networks: Relationships with natural and anthropogenic factors. *Science of the Total Environment*, 545, 152–162.

Amirabadi, M., Kahaei, M., and Nezamalhosseini, S. (2020). Novel suboptimal approaches for hyperparameter tuning of deep neural network. *Physical Communication*, 41.

Amitrano, D., Di Martino, G., Iodice, A., et al. (2018). Unsupervised rapid flood mapping using sentinel-1 grd sar images. *IEEE Transactions on Geoscience and Remote Sensing*, 56(6), 3290–3299.

Ansari, A. and Golabi, M. H. (2019). Prediction of spatial land use changes based on lcm in a gis environment for desert wetlands–a case study: Meighan wetland, iran. *International soil and water conservation research*, 7(1), 64–70.

Anusha, N. and Bharathi, B. (2020). Flood detection and flood mapping using multitemporal synthetic aperture radar and optical data. *The Egyptian Journal of Remote Sensing and Space Science*, 23(2), 207–219.

Arabameri, A., Danesh, A. S., Santosh, M., et al. (2022). Flood susceptibility mapping using meta-heuristic algorithms. *Geomatics, Natural Hazards and Risk*, 13(1), 949–974.

Awad, N., Mallik, N., and Hutter, F. (2021). DEHB: Evolutionary Hyperband for Scalable, Robust and Efficient Hyperparameter Optimization. In *Proceedings of IJCIA*, 2147–2153.

Bajamgnigni Gbambie, A., Poulin, A., Boucher, M., and Arsenault, R. (2017). Added value of alternative information in interpolated precipitation datasets for hydrology. 18 (1): 247–264. doi: 10.1175. Technical report, JHM-D-16-0032.1.

Baldib, P. and L. Gillen, D. (2022). Reproducible Hyperparameter Optimization. *Journal of computational and graphical statistics*, 31, 84–99.

Balogun, A.-L., Sheng, T. Y., Sallehuddin, M. H., et al. (2022). Assessment of data mining, multi-criteria decision making and fuzzy-computing techniques for spatial flood susceptibility mapping: a comparative study. *Geocarto International*, 37(26), 12989–13015.

Belete, D. M. and Huchaiah, M. D. (2022). Grid search in hyperparameter optimization of machine learning models for prediction of HIV/AIDS test results. *International Journal of Computers and Applications*, 44(9), 875–886.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011a). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011b). Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*, 2546–2554. ACM.

Bergstra, J. and Bengio, Y. (2012). Random Search for Hyperparameter Optimization. *Journal of Machine Learning Research*, 13, 281–305.

Bhatt, N., Thakkar, A., and Ganatra, A. (2012). A survey and current research challenges in meta learning approaches based on dataset characteristics. *International Journal of Soft Computing and Engineering*, 2, 234–247.

Bhuyan, K., Van Westen, C., Wang, J., and Meena, S. R. (2022). Mapping and characterising buildings for flood exposure analysis using open-source data and artificial intelligence. *Natural Hazards*, 119(2), 805–835.

Bischl, B., Binder, M., Lang, M., et al. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(2), e1484.

Biswas, S., Cobb, A. D., Sistrunk, A., et al. (2020). Better call Surrogates: A hybrid Evolutionary Algorithm for Hyperparameter optimization.

Blum, C. (2003). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, 35, 268–308.

Bose, A. and Chowdhury, I. R. (2020). Monitoring and modeling of spatio-temporal urban expansion and land-use/land-cover change using markov chain model: a case study in siliguri metropolitan area, west bengal, india. *Modeling Earth Systems and Environment*, 6, 2235–2249.

Breiman, L. (2001). Random forests. *Machine Learning*, 45, 5–32.

Bui, D. T., Tsangaratos, P., Ngo, P.-T. T., et al. (2019). Flash flood susceptibility modeling using an optimized fuzzy rule based feature selection technique and tree based ensemble methods. *Science of The Total Environment*, 668, 1038–1054.

Bui, Q.-T., Nguyen, Q.-H., Nguyen, X. L., et al. (2020). Verification of novel integrations of swarm intelligence algorithms into deep learning neural network for flood susceptibility mapping. *Journal of Hydrology*, 581, 124379.

Castiello, C., Castellano, G., and Fanelli, A. M. (2005). Meta-data: Characterization of input features for meta-learning. In *International Conference on Modeling Decisions for Artificial Intelligence*, 457–468. Springer.

Chakrabortty, R., Chandra Pal, S., Rezaie, F., et al. (2022). Flash-flood hazard susceptibility mapping in kangsabati river basin, india. *Geocarto International*, 37(23), 6713–6735.

Chapi, K., Singh, V. P., Shirzadi, A., et al. (2017). A novel hybrid artificial intelligence approach for flood susceptibility assessment. *Environmental Modelling & Software*, 95, 229–245.

Chen, T. and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA. Association for Computing Machinery.

Chen, Y., Liu, Z., Xu, H., et al. (2021). Meta-Baseline: Exploring Simple Meta-Learning for Few-Shot Learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 9062–9071.

Cho, H., Shin, J., and Rhee, W. (2022). B2EA: An Evolutionary Algorithm Assisted by Two Bayesian Optimization Modules for Neural Architecture Search.

Chopade, M. R., Mahajan, S., and Chaube, N. (2023). Assessment of land use, land cover change in the mangrove forest of ghogha area, gulf of khambhat, gujarat. *Expert Systems with Applications*, 212, 118839.

Chowdhuri, I., Pal, S. C., and Chakrabortty, R. (2020). Flood susceptibility mapping by ensemble evidential belief function and binomial logistic regression model on river basin of eastern india. *Advances in Space Research*, 65(5), 1466–1489.

Civco, D. L., Hurd, J. D., Wilson, E. H., et al. (2002). A comparison of land use and land cover change detection methods. In *ASPRS-ACSM Annual Conference*, 21, 18–33.

Clark, A., Phinn, S., and Scarth, P. (2023). Optimised u-net for land use–land cover classification using aerial photography. *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 1–23.

Congalton, R. G. (1991). A review of assessing the accuracy of classifications of remotely sensed data. *Remote sensing of environment*, 37(1), 35–46.

Costa, V. O. and Rodrigues, C. R. (2018). Hierarchical Ant Colony for Simultaneous Classifier Selection and Hyperparameter Optimization. In *IEEE Congress on Evolutionary Computation (CEC)*, 1–8. IEEE.

Cui, H. and Bai, J. (2019). A new hyperparameters optimization method for convolutional neural networks. *Pattern Recognition Letters*, 125, 828–834.

Cui, H., Quan, H., and Jin, R. (2023). Flood susceptibility mapping using novel hybrid approach of neural network with genetic quantum ensembles. *KSCE Journal of Civil Engineering*, 27, 431–441.

Dano, U. L., Balogun, A.-L., Matori, A.-N., et al. (2019). Flood susceptibility mapping using gis-based analytic network process: A case study of perlis, malaysia. *Water*, 11(3), 615.

Darem, A. A., Alhashmi, A. A., Almadani, A. M., et al. (2023). Development of a map for land use and land cover classification of the northern border region using remote sensing and gis. *The Egyptian Journal of Remote Sensing and Space Science*, 26(2), 341–350.

Das, S. (2019). Geospatial mapping of flood susceptibility and hydro-geomorphic response to the floods in ulhas basin, india. *Remote Sensing Applications: Society and Environment*, 14, 60–74.

de Pison, F. M., Gonzalez-Sendino, R., Aldama, A., et al. (2019). Hybrid methodology based on Bayesian optimization and GA-PARSIMONY to search for parsimony models by combining hyperparameter optimization and feature selection. *Neurocomputing*, 354, 20–26. Recent Advancements in Hybrid Artificial Intelligence Systems.

Deb, K. (2005). Practical Optimization Using Evolutionary Methods KanGAL Report.

Demir, S. and Åžahin, E. K. (2022). Liquefaction prediction with robust machine learning algorithms (SVM, RF, and XGBoost) supported by genetic algorithm-based feature selection and parameter optimization from the perspective of data processing. *Environmental Earth Sciences*, 81, 459.

Devi, M. R. S., Kumar, V. V., and Sivakumar, P. (2021). A review of image classification and object detection on machine learning and deep learning techniques. In *2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 1–8. IEEE.

Dewancker et al. (2015). Bayesian optimization primer. *SigOpt.*, 1–2.

Domhan, T., Springenberg, J. T., and Hutter, F. (2015). Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *IJCAI'15: Proceedings of the 24th International Conference on Artificial Intelligence*, page 3460–3468.

Dores, S. C. N. d., Soares, C., and Ruiz, D. (2018). Bandit-Based Automated Machine Learning. In *7th Brazilian Conference on Intelligent Systems (BRACIS)*, 121–126.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29–41.

Dou, P. and Chen, Y. (2017). Remote sensing imagery classification using adaboost with a weight vector (wv adaboost). *Remote Sensing Letters*, 8(8), 733–742.

Eggensperger, K., Feurer, M., Hutter, F., et al. (2013). Towards an empirical foundation for assessing Bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, 10, 3.

Ekmekcioğlu, Ö., Koc, K., and Özger, M. (2021). District based flood risk assessment in istanbul using fuzzy analytical hierarchy process. *Stochastic Environmental Research and Risk Assessment*, 35, 617–637.

Elkhrachy, I. (2022). Flash flood water depth estimation using sar images, digital elevation models, and machine learning algorithms. *Remote Sensing*, 14(3), 440.

Elsken, T., Staffler, B., Metzen, J. H., and Hutter, F. (2020). Meta-learning of neural architectures for few-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 12365–12375.

Erickson, N., Mueller, J., Shirkov, A., et al. (2020). AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. *arXiv preprint arXiv:2003.06505*.

Falkner, S., Klein, A., and Hutter, F. (2018). BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, PMLR 80.

Fenglin, W., Ahmad, I., and Zelenakova, M. (2023). Exploratory regression modeling for flood susceptibility mapping in the gis environment. *Scientific Reports*, 13, 247.

Feurer, M. and Hutter, F. (2018). *Hyperparameter Optimization*. Springer International Publishing.

Feurer, M., Klein, A., Eggensperger, K., et al. (2018). Auto-sklearn: Efficient and Robust Automated Machine Learning. *Automated Machine Learning*, 113–134.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. *Proceedings of the 34th International Conference on Machine Learning*, 10, 1126–1135.

Fortin, F., De Rainville, F., Gardner, M., et al. (2012). DEAP: Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research*, 13, 2171–2175.

Franceschi, L., Frasconi, P., Salzo, S., et al. (2018). Bilevel programming for hyperparameter optimization and meta-learning. *International conference on machine learning*, 1568–1577.

Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), 1189–1232.

Fuadah, Y. N., Pramudito, M. A., and Lim, K. M. (2023). An Optimal Approach for Heart Sound Classification Using Grid Search in Hyperparameter Optimization of Machine Learning. *Bioengineering*, 10(1), 45.

Garrote, J., Peña, E., and Díez-Herrero, A. (2021). Probabilistic flood hazard maps from Monte Carlo derived peak flow values-An application to flood risk management in Zamora city (Spain). *Applied Sciences*, 11(14), 6629.

Gascon, F., Cadau, E., Colin, O., et al. (2014). Copernicus Sentinel-2 mission: products, algorithms and Cal/Val. In *Earth observing systems XIX*, 9218, 455–463.

Ghosh, B. (2023). Flood susceptibility assessment and mapping in a monsoon-dominated tropical river basin using GIS-based data-driven bivariate and multivariate statistical models and their ensemble techniques. *Environmental Earth Sciences*, 82(1), 28.

Giovannettone, J., Sangameswaran, S., Maderia, C., and Batten, B. (2020). Spatial analysis of flood susceptibility throughout currituck county, north carolina. *Journal of Hydrologic Engineering*, 25(8), 05020021.

Giraud-Carrier, C., Vilalta, R., and Brazdil, P. (2004). Introduction to the special issue on meta-learning. *Machine learning*, 54, 187–193.

Giri, C., Defourny, P., and Shrestha, S. (2003). Land cover characterization and mapping of continental southeast asia using multi-resolution satellite sensor data. *International journal of remote Sensing*, 24(21), 4181–4196.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Boston, United States.

Gomes, V. C., Queiroz, G. R., and Ferreira, K. R. (2020). An overview of platforms for big earth observation data management and analysis. *Remote Sensing*, 12(8), 1253.

Gómez, C., White, J. C., and Wulder, M. A. (2016). Optical remotely sensed time series data for land cover classification: A review. *ISPRS Journal of photogrammetry and Remote Sensing*, 116, 55–72.

González-Arqueros, M. L., Mendoza, M. E., Bocco, G., and Solís Castillo, B. (2018). Flood susceptibility in rural settlements in remote zones: The case of a mountainous basin in the sierra-costa region of michoacán, mexico. *Journal of Environmental Management*, 223, 685–693.

Government of India (2023). National disaster management authority. `https://ndma.gov.in/`, Last accessed on 2023-09-17.

Government of Kerala (2023a). Allapuzha district, topography. `https://alappuzha.nic.in/en/about-district/topography/`, Last accessed on 2023-02-01.

Government of Kerala (2023b). Kerala, ernakulam district website. https://ernakulam.nic.in/, Last accessed on 2023-02-11.

Government of Kerala (2023c). Kerala, topography. https://www.kerala.gov.in/subdetail/NTM1ODMxNzQuNDg=/MjA0ODc2ODQuMzY=, Last accessed on 2023-17-09.

Guo, B., Hu, J., Wu, W., et al. (2019). The Tabu Genetic Algorithm: A Novel Method for Hyper-Parameter Optimization of Learning Algorithms. *Electronics*, 8, 579–598.

Halmy, M. W. A., Gessler, P. E., Hicke, J. A., and Salem, B. B. (2015). Land use/land cover change detection and prediction in the north-western coastal desert of Egypt using Markov-CA. *Applied Geography*, 63, 101–112.

Hamad, R., Balzter, H., and Kolo, K. (2018a). Predicting land use/land cover changes using a ca-markov model under two different scenarios. *Sustainability*, 10(10), 3421.

Hamad, R., Kolo, K., and Balzter, H. (2018b). Land cover changes induced by demining operations in halgurd-sakran national park in the kurdistan region of iraq. *Sustainability*, 10(7), 2422.

Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2), 159–195.

Hao, C., Yunus, A. P., Siva Subramanian, S., and Avtar, R. (2021). Basin-wide flood depth and exposure mapping from sar images and machine learning models. *Journal of Environmental Management*, 297, 113367.

Hartmann, T., Moawad, A., Schockaert, C., et al. (2019). Meta-modelling meta-learning. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*, 300–305.

Hasanuzzaman, M., Islam, A., Bera, B., and Shit, P. K. (2022). A comparison of performance measures of three machine learning algorithms for flood susceptibility mapping of river silabati (tropical river, india). *Physics and Chemistry of the Earth, Parts A/B/C*, 127, 103198.

Hazan, E., Klivans, A., and Yuan, Y. (2018). Hyperparameter Optimization: A Spectral Approach. In *International Conference on Learning Representations*, 17, page 2.

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2021). Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9), 5149–5169.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

Huisman, M., Van Rijn, J. N., and Plaat, A. (2021). A survey of deep meta-learning. *Artificial Intelligence Review*, 54(6), 4483–4541.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential Model-Based Optimization for General Algorithm Configuration. In *Learning and Intelligent Optimization*, 6683, page 507–523. Springer.

Hutter, F., Lucke, J., and Schmidt-Thieme, L. (2015). Beyond Manual Tuning of Hyperparameters. *KI - Kunstliche Intelligenz*, 29, 329–337.

Iervolino, P., Guida, R., Iodice, A., and Riccio, D. (2015). Flooding water depth estimation with high-resolution sar. *IEEE Transactions on Geoscience and Remote Sensing*, 53(5), 2295–2307.

Jamieson, K. and Talwalkar, A. (2015). Non-stochastic Best Arm Identification and Hyperparameter Optimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 51, 240–248.

Jat, M. K., Choudhary, M., and Saxena, A. (2017). Application of geo-spatial techniques and cellular automata for modelling urban growth of a heterogeneous urban fringe. *The Egyptian journal of remote sensing and space science*, 20(2), 223–241.

Jha, M. K. and Afreen, S. (2020). Flooding urban landscapes: Analysis using combined hydrodynamic and hydrologic modeling approaches. *Water*, 12(7), 1986.

Jiang, X., Liang, S., He, X., et al. (2021). Rapid and large-scale mapping of flood inundation via integrating spaceborne synthetic aperture radar imagery with unsupervised deep learning. *ISPRS Journal of Photogrammetry and Remote Sensing*, 178, 36–50.

Jin, H., Song, Q., and Hu, X. (2019). Auto-Keras: An Efficient Neural Architecture Search System. In *KDD '19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 1946–1956.

Jozdani, S. E., Johnson, B. A., and Chen, D. (2019). Comparing deep neural networks, ensemble classifiers, and support vector machine algorithms for object-based urban land use/land cover classification. *Remote Sensing*, 11(14), 1713.

Kandasamy, K., Dasarathy, G., Oliva, J., et al. (2015). Multi-fidelity Bayesian Optimisation with Continuous Approximations. In *Precup and Teh*, page 1799–1808.

Kandasamy, K., Dasarathy, G., Oliva, J., et al. (2016). Gaussian Process Bandit Optimisation with Multi-fidelity Evaluations. *Neural Computation*, page 992–1000.

Karnin, Z., Koren, T., and Somekh, O. (2013). Almost Optimal Exploration in Multi-Armed Bandits. *Dasgupta and McAllester*, 28(3), 1238–1246.

Kazakis, N., Kougias, I., and Patsialis, T. (2015). Assessment of flood hazard areas at a regional scale using an index-based approach and analytical hierarchy process: Application in rhodope–evros region, greece. *Science of The Total Environment*, 538, 555–563.

Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1942–1948. IEEE.

Khalid, R. and Javaid, N. (2020). A survey on hyperparameters optimization algorithms of forecasting models in smart grid. *Sustainable Cities and Society*, 61, 102275.

Khosravi, K., Panahi, M., Golkarian, A., et al. (2020). Convolutional neural network approach for spatial prediction of flood hazard at national scale of iran. *Journal of Hydrology*, 591, 125552.

Koch, G., Zemel, R., Salakhutdinov, R., et al. (2015). Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, 2, 1–8.

Komer, B., Bergstra, J., and Eliasmith, C. (2019). Hyperopt-sklearn. *Automated Machine Learning: Methods, Systems, Challenges*, 97–111.

Kotthoff, L., Thornton, C., Hoos, H. H., et al. (2018). Auto-WEKA: Automatic Model Selection and Hyperparameter Optimization in WEKA. *Automated Machine Learning*, 81–95.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 1097–1105.

Kulithalai Shiyam Sundar, P. and Deka, P. C. (2022). Spatio-temporal classification and prediction of land use and land cover change for the vembanad lake system, kerala: a machine learning approach. *Environmental Science and Pollution Research*, 29(57), 86220–86236.

Lake, B., Salakhutdinov, R., Gross, J., and Tenenbaum, J. (2011). One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, 33, 1–7.

Lakhmiri, D. and Digabel, S. (2022). Use of Static Surrogates in Hyperparameter Optimization. *SN Operations Research Forum*, 3(1), 1–18.

Lakhmiri, D., Digabel, S. L., and Tribes, C. (2021). HyperNOMAD: Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search. *ACM Trans. Math. Softw.*, 47(3).

Lakra, A. V. and Jena, S. (2022). Optimization of random forest hyperparameter using improved pso for handwritten digits classification. In *Computing, Communication and Learning*, 1729, 266–276.

Lambin, E. F. (1997). Modelling and monitoring land-cover change processes in tropical regions. *Progress in physical geography*, 21(3), 375–393.

Lan, G., Tomczak, J. M., Roijers, D. M., and Eiben, A. (2022). Time efficiency in optimization with a Bayesian-Evolutionary algorithm. *Swarm and Evolutionary Computation*, 69, 100970–100983.

Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *biometrics*, 159–174.

LeDell, E. and Poirier, S. (2020). H2O AutoML: Scalable Automatic Machine Learning. In *7th ICML Workshop on Automated Machine Learning*, 2020.

Lee, S., Kim, J.-C., Jung, H.-S., et al. (2017). Spatial prediction of flood susceptibility using random-forest and boosted-tree models in seoul metropolitan city, korea. *Geomatics, Natural Hazards and Risk*, 8(2), 1185–1203.

Lemke, C., Budka, M., and Gabrys, B. (2015). Metalearning: a survey of trends and technologies. *Artificial Intelligence Review*, 44, 117–130.

Li, C., Li, S., Wang, H., et al. (2023a). Attention-based deep meta-transfer learning for few-shot fine-grained fault diagnosis. *Knowledge-Based Systems*, 264, 110345.

Li, F., Yigitcanlar, T., Nepal, M., et al. (2023b). Machine learning and remote sensing integration for leveraging urban sustainability: A review and framework. *Sustainable Cities and Society*, 31, 104653–104681.

Li, L., Jamieson, K., DeSalvo, G., et al. (2018). Hyperband A Novel Bandit Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research*, 18, 1–52.

Li, Y. and Hong, H. (2023). Modelling flood susceptibility based on deep learning coupling with ensemble learning models. *Journal of Environmental Management*, 325, 116450.

Li, Y., Martinis, S., Wieland, M., et al. (2019). Urban flood mapping using sar intensity and interferometric coherence via bayesian network fusion. *Remote Sensing*, 11(19), 2231–2252.

Li, Y., Osei, F. B., Hu, T., and Stein, A. (2023c). Urban flood susceptibility mapping based on social media data in chengdu city, china. *Sustainable Cities and Society*, 88, 104307.

Li, Z., Zhou, F., Chen, F., and Li, H. (2017). Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*.

Lindauer, M., Eggensperger, K., Feurer, M., et al. (2022). SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization. *Journal of Machine Learning Research*, 23, 1–9.

Liu, C., Li, W., Zhu, G., et al. (2020a). Land use/land cover changes and their driving factors in the northeastern tibetan plateau based on geographical detectors and google earth engine: A case study in gannan prefecture. *Remote Sensing*, 12(19), 3139.

Liu, J., Wang, J., Xiong, J., et al. (2022a). Assessment of flood susceptibility mapping using support vector machine, logistic regression and their ensemble techniques in the belt and road region. *Geocarto International*, 37(25), 9817–9846.

Liu, L., Liang, Y., and Hashimoto, S. (2020b). Integrated assessment of land-use/coverage changes and their impacts on ecosystem services in gansu province, northwest china: Implications for sustainable development goals. *Sustainability Science*, 15, 297–314.

Liu, X., Wu, J., and Chen, S. (2022b). A context-based meta-reinforcement learning approach to efficient hyperparameter optimization. *Neurocomputing*, 478, 89–103.

Lorenzo, P. R., Nalepa, J., Kawulok, M., et al. (2017). Particle swarm optimization for hyper-parameter selection in deep neural networks. In *GECCO '17: Proceedings of the Genetic and Evolutionary Computation Conference*, 481–488.

Loshchilov, I. and Hutter, F. (2016). CMA-ES for Hyperparameter Optimization of Deep Neural Networks.

Lu, D. and Weng, Q. (2007). A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing*, 28(5), 823–870.

Luo, S., Li, Y., Gao, P., et al. (2022). Meta-seg: A survey of meta-learning for image segmentation. *Pattern Recognition*, 126, 108586.

Macarringue, L. S., Bolfe, É. L., and Pereira, P. R. M. (2022). Developments in land use and land cover classification techniques in remote sensing: A review. *Journal of Geographic Information System*, 14(1), 1–28.

Mahdizadeh Gharakhanlou, N. and Perez, L. (2023). Flood susceptible prediction through the use of geospatial variables and machine learning methods. *Journal of Hydrology*, 617, 129121.

Maier, H. R., Razavi, S., Kapelan, Z., et al. (2019). Introductory overview: Optimization using evolutionary algorithms and other metaheuristics. *Environmental Modelling and Software*, 114, 195–213.

Malekian, A. and Azarnivand, A. (2016). Application of integrated shannon's entropy and vikor techniques in prioritization of flood risk in the shemshak watershed, iran. *Water Resources Management*, 30, 409–425.

Mashwani, W. K. (2008). Comprehensive Survey of the Hybrid Evolutionary Algorithms. *International Journal of Applied Evolutionary Computation (IJAEC)*, 4(2), 1–19.

Mateo-Garcia, G., Veitch-Michaelis, J., Smith, L., et al. (2018). Towards global flood mapping onboard low cost satellites with machine learning. *Scientific Reports*, 11(1), 7249.

McGrath, H. and Gohl, P. N. (2022). Accessing the impact of meteorological variables on machine learning flood susceptibility mapping. *Remote Sensing*, 14(7), 1656.

Mehravar, S., Razavi-Termeh, S. V., Moghimi, A., et al. (2023). Flood susceptibility mapping using multi-temporal sar imagery and novel integration of nature-inspired algorithms into support vector regression. *Journal of Hydrology*, 617, 129100.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. (2018). A Simple Neural Attentive Meta-Learner. In *International Conference on Learning Representations*, 1–17.

Mishra, P. K., Rai, A., and Rai, S. C. (2020). Land use and land cover change detection using geospatial techniques in the sikkim himalaya, india. *The Egyptian Journal of Remote Sensing and Space Science*, 23(2), 133–143.

Mohammadi, F. G., Amini, M. H., and Arabnia, H. R. (2020). An introduction to advanced machine learning: Meta-learning algorithms, applications, and promises. *Optimization, Learning, and Control for Interdependent Complex Networks*, 129–144.

Mohan, B. and Badrah, J. (2022). A novel automated SuperLearner using a genetic algorithm-based hyperparameter optimization. *Advances in Engineering Software*, 175.

MohanRajan, S. N., Loganathan, A., and Manoharan, P. (2020). Survey on land use/land cover (lu/lc) change analysis in remote sensing and gis environment: Techniques and challenges. *Environmental Science and Pollution Research*, 27, 29900–29926.

Montesinos López, O. A., Montesinos López, A., and Crossa, J. (2022). Overfitting, model tuning, and evaluation of prediction performance. *Multivariate statistical machine learning methods for genomic prediction*, 109–139.

Mousavi, S. M., Ataie-Ashtiani, B., and Hosseini, S. M. (2022). Comparison of statistical and mcdm approaches for flood susceptibility mapping in northern iran. *Journal of Hydrology*, 612, 128072–128091.

Mu, L., Wang, L., Wang, Y., et al. (2019). Urban land use and land cover change prediction via self-adaptive cellular based deep learning with multisourced data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(12), 5233–5247.

Munkhdalai, T., Yuan, X., Mehri, S., and Trischler, A. (2018). Rapid adaptation with conditionally shifted neurons. In *International conference on machine learning*, 3664–3673.

N. Knight, J. and Lunacek, M. (2007). Reducing the space-time complexity of the cma-es. *Genetic and Evolutionary Computation Conference, GECCO*, 658 - 665.

Nawindah (2017). Simple Additive Weighting (SAW) Mathematics Method for Warehouse Disaster Location Selection In Central Jakarta, Indonesia. *Int. J. Pure Appl. Math.*, 117(15), 795–803.

Nguyen, H. T. T., Doan, T. M., Tomppo, E., and McRoberts, R. E. (2020). Land Use/land cover mapping using multitemporal Sentinel-2 imagery and four classification methods - A case study from Dak Nong, Vietnam. *Remote Sensing*, 12(9), 1367.

Nichol, A. and Schulman, J. (2018). Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2(3), 4.

Olson, R. S. and Moore, J. H. (2018). TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning. *Automated Machine Learning*, 151–160.

O'Neill, E. (2016). The impact of perceived flood exposure on flood-risk perception: The role of distance. *Risk analysis: an official publication of the Society for Risk Analysis*, 36(11), 2158–2186.

Orive, D., Sorrosal, G., Borges, C. E., et al. (2014). Evolutionary Algorithms for Hyperparameter Tuning on Neural Network Models. In *Proceedings of the 26th European Modelling & Simulation Symposium*, 402–409.

Osama Ahmed, M., Vaswani, S., and Schmidt, M. (2020). Combining Bayesian optimization and Lipschitz optimization. *Machine Learning*, 109, 79–102.

Pal, S. C., Chowdhuri, I., Das, B., et al. (2022). Threats of climate change and land use patterns enhance the susceptibility of future floods in india. *Journal of Environmental Management*, 305, 114317.

Pant, M., Thangaraj, R., Grosan, C., and Abraham, A. (2008). Hybrid Differential Evolution - Particle Swarm Optimization Algorithm for Solving Global Optimization Problems. In *2008 Third International Conference on Digital Information Management*, 2008, 18–24.

Parthasarathy, K., Deka, P. C., Saravanan, S., et al. (2021). Assessing the impact of 2018 tropical rainfall and the consecutive flood-related damages for the state of Kerala, India. *Disaster Resilience and Sustainability*, 379–395.

Parvinnezhad, D., Delavar, M. R., Pijanowski, B. C., and Claramunt, C. (2021). Integration of adaptive neural fuzzy inference system and fuzzy rough set theory with support vector regression to urban growth modelling. *Earth Science Informatics*, 14, 17–36.

Patrikaki, O., Kazakis, N., Kougias, I., et al. (2018). Assessing flood hazard at river basin scale with an index-based approach: The case of mouriki, greece. *Geosciences*, 8(2), 50.

Pham, B. T., Luu, C., Phong, T. V., et al. (2021). Can deep learning algorithms outperform benchmark machine learning algorithms in flood susceptibility modeling? *Journal of Hydrology*, 592, 125615.

Pijanowski, B. C., Brown, D. G., Shellito, B. A., and Manik, G. A. (2002). Using neural networks and gis to forecast land use changes: a land transformation model. *Computers, environment and urban systems*, 26(6), 553–575.

Prasad, P., Loveson, V. J., Das, B., and Kotha, M. (2022). Novel ensemble machine learning models in flood susceptibility mapping. *Geocarto International*, 37(16), 4571–4593.

Rafiei-Sardooi, E., Azareh, A., Choubin, B., et al. (2021). Evaluating urban flood risk using hybrid method of topsis and machine learning. *International Journal of Disaster Risk Reduction*, 66, 102614.

Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. (2019). Rapid learning or feature reuse? Towards understanding the effectiveness of MAML. *arXiv preprint arXiv:1909.09157*.

Rahmati, O., Pourghasemi, H. R., and Zeinivand, H. (2016a). Flood susceptibility mapping using frequency ratio and weights-of-evidence models in the golastan province, iran. *Geocarto International*, 31(1), 42–70.

Rahmati, O., Zeinivand, H., and Besharat, M. (2016b). Flood hazard zoning in yasooj region, iran, using gis and multi-criteria decision analysis. *Geomatics, Natural Hazards and Risk*, 7(3), 1000–1017.

Ravi, S. and Larochelle, H. (2016). Optimization as a model for few-shot learning. In *International conference on learning representations*, 1–11.

Razavi Termeh, S. V., Kornejady, A., Pourghasemi, H. R., and Keesstra, S. (2018). Flood susceptibility mapping using novel ensembles of adaptive neuro fuzzy inference system and metaheuristic algorithms. *Science of The Total Environment*, 615, 438–451.

Rentschler, J., Salhab, M., and Jafino, B. A. (2022). Flood exposure and poverty in 188 countries. *Nature Communications*, 13, 2041–1723.

Rienow, A. and Goetzke, R. (2015). Supporting sleuth–enhancing a cellular automaton with support vector machines for urban growth modeling. *Computers, Environment and Urban Systems*, 49, 66–81.

Rivolli, A., Garcia, L. P., Soares, C., et al. (2022). Meta-features for meta-learning. *Knowledge-Based Systems*, 240, 108101.

Romali, N., Yusop, Z., and Ismail, A. (2018). Hydrological modelling using HEC-HMS for flood risk assessment of Segamat Town, Malaysia. *IOP Conference Series: Materials Science and Engineering*, 318, 012029.

Ruben, G. B., Zhang, K., Dong, Z., and Xia, J. (2020). Analysis and projection of land-use/land-cover dynamics through scenario-based simulations using the ca-markov model: A case study in guanting reservoir basin, china. *Sustainability*, 12(9), 3747.

Russakovsky, O., Deng, J., Su, H., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115, 211–252.

Rwanga, S. S., Ndambuki, J. M., et al. (2017). Accuracy assessment of land use/land cover classification using remote sensing and GIS. *International Journal of Geosciences*, 8(04), 611.

Saha, S., Gayen, A., and Bayen, B. (2022). Deep learning algorithms to develop flood susceptibility map in data-scarce and ungauged river basin in india. *Stochastic Environmental Research and Risk Assessment*, 36(10), 3295–3310.

Santoro, A., Bartunov, S., Botvinick, M., et al. (2016). Meta-learning with memory-augmented neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, 48, 1842–1850.

Santos, M. J., Smith, A. B., Dekker, S. C., et al. (2021). The role of land use and land cover change in climate change vulnerability assessments of biodiversity: a systematic review. *Landscape Ecology*, 1–16.

Saravanan, S. and Abijith, D. (2022). Flood susceptibility mapping of northeast coastal districts of tamil nadu india using multi-source geospatial data and machine learning techniques. *Geocarto International*, 37(27), 15252–15281.

Saravanan, S., Abijith, D., Reddy, N. M., et al. (2023). Flood susceptibility mapping using machine learning boosting algorithms techniques in idukki district of kerala india. *Urban Climate*, 49, 101503.

Sarchani, S., Awol, F. S., and Tsanis, I. (2021). Hydrological analysis of extreme rain events in a medium-sized basin. *Applied Sciences*, 11(11), 4901.

Sarkar, D. and Mondal, P. (2020). Flood vulnerability mapping using frequency ratio (fr) model: a case study on kulik river basin, indo-bangladesh barind region. *Applied Water Science*, 10(1), 1–13.

Saxena, A., Jat, M. K., and Clarke, K. C. (2021). Development of sleuth-density for the simulation of built-up land density. *Computers, Environment and Urban Systems*, 86, 101586.

Schwalm, C. R., Anderegg, W. R., Michalak, A. M., et al. (2017). Global patterns of drought recovery. *Nature*, 548(7666), 202–205.

Sertel, E., Ekim, B., Ettehadi Osgouei, P., and Kabadayi, M. E. (2022). Land use and land cover mapping using deep learning based segmentation approaches and vhr worldview-3 images. *Remote Sensing*, 14(18), 4558.

Shahabi, H., Shirzadi, A., Ghaderi, K., et al. (2020). Flood detection and susceptibility mapping using sentinel-1 remote sensing data and a machine learning approach: Hybrid intelligence of bagging ensemble based on k-nearest neighbor classifier. *Remote Sensing*, 12(2), 266.

Shahriari, B., Swersky, K., Wang, Z., et al. (2015). Taking the Human Out of the Loop: A Review of Bayesian Optimization. In *Proceedings of the IEEE*, 104(1), page 148–175.

Shen, X., Anagnostou, E. N., Allen, G. H., et al. (2019). Near-real-time non-obstructed flood inundation mapping using synthetic aperture radar. *Remote Sensing of Environment*, 221, 302–315.

Silva, L. P., Xavier, A. P. C., da Silva, R. M., and Santos, C. A. G. (2020). Modeling land cover change based on an artificial neural network for a semiarid river basin in northeastern brazil. *Global Ecology and Conservation*, 21, e00811.

Singh, R., Bharti, V., Purohit, V., et al. (2021). Metamed: Few-shot medical image classification using gradient-based meta-learning. *Pattern Recognition*, 120, 108111.

Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 1–11.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances on Neural Information Processing Systems*, 2–8.

Song, Y., Wang, T., Cai, P., et al. (2023). A comprehensive survey of few-shot learning: Evolution, applications, challenges, and opportunities. *ACM Computing Surveys*.

Soper, D. S. (2023). Hyperparameter Optimization Using Successive Halving with Greedy Cross Validation. *Algorithms*, 16(1), 17.

Srinivasan, D. and Seow, T. (2003). Particle swarm inspired evolutionary algorithm (PS-EA) for multiobjective optimization problems. In *The 2003 Congress on Evolutionary Computation*, 4, 2292–2297.

Storn, R. and Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11, 341–359.

Sufiyan, I. and Magaji, J. (2018). Modeling flood hazard using SWAT and 3D analysis in Terengannu Watershed. *J Clean WAS*, 2, 19–24.

Sun, Q., Liu, Y., Chua, T.-S., and Schiele, B. (2019). Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 403–412.

Sung, F., Yang, Y., Zhang, L., et al. (2018). Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1199–1208.

Swersky, K., Snoek, J., and Adams, R. P. (2013). Multi-Task Bayesian Optimization. *Advances in Neural Information Processing Systems*, 26, 121–126.

Swersky, K., Snoek, J., and Adams, R. P. (2014). Freeze-thaw bayesian optimization.

Talukdar, S., Singha, P., Mahato, S., et al. (2020). Land-use land-cover classification by machine learning classifiers for satellite observations—a review. *Remote Sensing*, 12(7), 1135.

Tani, L., Rand, D., Veelken, C., and Kadastik, M. (2021). Evolutionary algorithms for hyperparameter optimization in machine learning for application in high energy physics. *The European Physical Journal*, 81(170), 1434–6052.

Tanim, A. H., McRae, C. B., Tavakol-Davani, H., and Goharian, E. (2022). Flood detection in urban areas using satellite imagery and machine learning. *Water*, 14(7), 1140.

Tassi, A. and Vizzari, M. (2020). Object-oriented lulc classification in google earth engine combining snic, glcm, and machine learning algorithms. *Remote Sensing*, 12(22), 3776.

Tayyebi, A. and Pijanowski, B. C. (2014). Modeling multiple land use changes using ann, cart and mars: Comparing tradeoffs in goodness of fit and explanatory power of data mining tools. *International Journal of Applied Earth Observation and Geoinformation*, 28, 102–116.

Tehrany, M. S., Pradhan, B., and Jebur, M. N. (2013). Spatial prediction of flood susceptible areas using rule based decision tree (DT) and a novel ensemble bivariate and multivariate statistical models in GIS. *Journal of Hydrology*, 504, 69–79.

Thrun, S. and Pratt, L. (1998). Learning to learn: Introduction and overview. *Learning to learn*, 3–17.

Tian, Y., Zhao, X., and Huang, W. (2022). Meta-learning approaches for learning-to-learn in deep learning: A survey. *Neurocomputing*, 494, 203–223.

Vanschoren, J. (2018). Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*.

Verma, P., Raghubanshi, A., Srivastava, P. K., and Raghubanshi, A. (2020). Appraisal of kappa-based metrics and disagreement indices of accuracy assessment for paramet-

ric and nonparametric techniques used in lulc classification and change detection. *Modeling Earth Systems and Environment*, 6, 1045–1059.

Viana, C. M., Oliveira, S., Oliveira, S. C., and Rocha, J. (2019). Land use/land cover change detection and urban sprawl analysis. In *Spatial modeling in GIS and R for earth and environmental sciences*, 621–651. Elsevier.

Victoria, A. H. and Maragatham, G. (2021). Automatic tuning of hyperparameters using Bayesian optimization. *Evolving Systems*, 12, 217–223.

Vijaykumar, P., Abhilash, S., Sreenath, A., et al. (2021). Kerala floods in consecutive years - Its association with mesoscale cloudburst and structural changes in monsoon clouds over the west coast of India. *Weather and Climate Extremes*, 33, 100339.

Vilalta, R. and Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18, 77–95.

Vilalta, R., Giraud-Carrier, C., and Brazdil, P. (2010). *Meta-Learning - Concepts and Techniques*. Springer.

Vilasan, R. and Kapse, V. (2022). Evaluation of the prediction capability of AHP and F-AHP methods in flood susceptibility mapping of Ernakulam district (India). *Natural Hazards*, 112, 1767–1793.

Vinyals, O., Blundell, C., Lillicrap, T., et al. (2016). Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 1–9.

Wang, J., Bretz, M., Dewan, M. A. A., and Delavar, M. A. (2022). Machine learning in modelling land-use and land cover-change (lulcc): Current status, challenges and prospects. *Science of the Total Environment*, 822, 153559.

Wang, J. X. (2021). Meta-learning in natural and artificial intelligence. *Current Opinion in Behavioral Sciences*, 38, 90–95.

Wang, Y., Fang, Z., and Hong, H. (2019). Comparison of convolutional neural networks for landslide susceptibility mapping in yanshan county, china. *Science of the total environment*, 666, 975–993.

Wang, Y., Fang, Z., Hong, H., et al. (2021). Flood susceptibility mapping by integrating frequency ratio and index of entropy with multilayer perceptron and classification and regression tree. *Journal of Environmental Management*, 289, 112449.

Wang, Y., Fang, Z., Hong, H., and Peng, L. (2020a). Flood susceptibility mapping using convolutional neural network frameworks. *Journal of Hydrology*, 582, 124482.

Wang, Y., Yao, Q., Kwok, J. T., and Ni, L. M. (2020b). Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys*, 53(3).

Weydahl, D. (1996). Flood monitoring in norway using ers-1 sar images. In *IGARSS '96. 1996 International Geoscience and Remote Sensing Symposium*, 1, 151–153.

Xie, J., Long, F., Lv, J., et al. (2022). Joint distribution matters: Deep brownian distance covariance for few-shot classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 7972–7981.

Xiong, Y., Zhou, Y., Wang, F., et al. (2021). Landslide Susceptibility Mapping Using Ant Colony Optimization Strategy and Deep Belief Network in Jiuzhaigou Region. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14, 11042–11057.

Xu, H., Wang, J., Li, H., et al. (2021). Unsupervised meta-learning for few-shot learning. *Pattern Recognition*, 116, 107951.

Xue, Z., Xie, Z., Xing, Z., and Duan, L. (2020). Relative position and map networks in few-shot learning for image classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 932–933.

Yang, L. and Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415, 295–316.

Yariyan, P., Avand, M., Abbaspour, R. A., et al. (2020). Flood susceptibility mapping using an improved analytic network process with statistical models. *Geomatics, Natural Hazards and Risk*, 11(1), 2282–2314.

Yaseen, A., Lu, J., and Chen, X. (2022). Comparison of statistical and mcdm approaches for flood susceptibility mapping in northern iran. *Stoch Environ Res Risk Assess*, 36, 3041–3061.

Youssef, A. M., Pradhan, B., Dikshit, A., and Mahdi, A. M. (2022). Comparative study of convolutional neural network (cnn) and support vector machine (svm) for flood susceptibility mapping: a case study at ras gharib, red sea, egypt. *Geocarto International*, 37(26), 11088–11115.

Yu, D., Xie, P., Dong, X., et al. (2018). Improvement of the swat model for event-based flood simulation on a sub-daily timescale. *Hydrology and Earth System Sciences*, 22(9), 5001–5019.

Zahedi, L., Mohammadi, F. G., and Amini, M. H. (2021). HyP-ABC: A Novel Automated Hyper-Parameter Tuning Algorithm Using Evolutionary Optimization. *arXiv preprint arXiv:2109.05319*.

Zhang, C., Cai, Y., Lin, G., and Shen, C. (2020). Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 12203–12213.

Zhao, G., Pang, B., Xu, Z., et al. (2019). Assessment of urban flood susceptibility using semi-supervised machine learning model. *Science of The Total Environment*, 659, 940–949.

Zhao, G., Pang, B., Xu, Z., et al. (2020). Urban flood susceptibility assessment based on convolutional neural networks. *Journal of Hydrology*, 590, 125235.

Zimmer, L., Lindauer, M., and Hutter, F. (2020). AutoPyTorch Tabular: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL. *arXiv preprint arXiv:2006.13799*.

Zulfiqar, M., Gamage, K. A. A., Kamran, M., and Rasheed, M. B. (2022). Hyperparameter Optimization of Bayesian Neural Network Using Bayesian Optimization and Intelligent Feature Engineering for Load Forecasting. *Sensors*, 22(12), 4446.

LIST OF PUBLICATIONS

Journal Publications

1. Amala Mary Vincent, Jidesh P. (2023). “An improved hyperparameter optimization framework for AutoML systems using evolutionary algorithms", Scientific Reports (Nature portfolio), Vol. 13, 4737, https://doi.org/10.1038/s41598-023-32027-3, (SCI IF: 4.99).

2. Amala Mary Vincent, Parthasarathy K. S. S. and Jidesh P. (2023).“Flood susceptibility mapping using AutoML and a deep learning framework with evolutionary algorithms for hyperparameter optimization". Applied Soft Computing (2023): 110846, https://doi.org/10.1016/j.asoc.2023.110846, (SCI IF: 8.7).

Papers Submitted to Journal

1. Amala Mary Vincent, Parthasarathy K. S. S. and Jidesh P. (2024). “Analysis of hyperparameter optimization in machine learning models for land use, land cover classification".

2. Amala Mary Vincent, Jidesh P. (2024).“An analysis of importance of hyperparameter optimization in meta-learning".

BIO-DATA

Name :	Amala Mary Vincent
Date of Birth :	07-12-1993
Mobile No :	+91 9497296095
Email Id :	amalamaryvincent@gmail.com
Permanent Address :	Cheenkallel House
	Mukkom P.O, Kozhikode
	Kerala, India, 673602.
Qualifications :	B.Tech (Computer Science)
	M.Tech (Computational Mathematics)
Publications :	International Journals - 2