ABSTRACT

Title of Dissertation: Graph-based Methods for Efficient, Interpretable and Reliable Machine Learning

Yujunrong Ma

Dissertation Directed by: Professor Shuvra S. Bhattacharyya
Dept. of Electrical and Computer Engr., and
Institute for Advanced Computer Studies
and
Professor Kiminori Nakamura (co-advisor)
Dept. of Criminology and Criminal Justice

Machine learning algorithms have revolutionized fields such as computer vision, natural language processing, and speech recognition by offering the capability to analyze and extract information from vast datasets, a task far beyond human capacity. The deployment of these algorithms in high-stakes applications, including medical diagnosis, computational finance and criminal justice, underscores their growing importance. However, the decision-making processes of the so-called black-box models used in such areas raise considerable concerns. Therefore, enhancing the interpretability of these models is crucial, as it helps address issues like biases and inconsistencies in predictions, thereby making the models more comprehensible and trustworthy to end-users.

Moreover, interpretability facilitates a deeper understanding of model behavior, such as the distribution of contributions across inputs. This deeper understanding can be applied to significantly improve efficiency. This is especially relevant as machine learning models find applications on edge devices, where computational resources are often limited. For

such applications, significant improvements in energy efficiency and resource requirements can be obtained by optimizing and adapting model implementations based on an understanding of the models' internal behavior. However, such optimization introduces new challenges that arise due to factors such as complex, dynamically-determined dependency management among computations.

This dissertation presents five main contributions. The first contribution is the development of a novel type of interpretable machine learning model for applications in criminology and criminal justice (CCJ). The model involves graphical representations in the form of single decision trees, where the trees are constructed in an optimized fashion using a novel evolutionary algorithm. This approach not only enhances intrinsic interpretability but also enables users to understand the decision-making process more transparently, addressing the critical need for clarity in machine learning models' predictions. At the same time, the application of evolutionary algorithm methods enables such interpretability to be provided without significant degradation in model accuracy.

In the second contribution, we develop new multi-objective evolutionary algorithm methods to find a balance between fairness and predictive accuracy in CCJ applications. We build upon the single-decision-tree framework developed in the first contribution of the dissertation, and systematically integrate considerations of fairness and multi-objective optimization.

In the third contribution, we develop new methods for crime forecasting applications. In particular, we develop new interpretable, attention-based methods using convolutional long short-term memory (ConvLSTM) models. These methods combine the power of ConvLSTM models in capturing spatio-temporal patterns with the interpretability of

attention mechanisms. This combination of capabilities allows for the identification of key geographic areas in the input data that contribute to predictions from the model.

The fourth contribution introduces a dynamic dataflow-graph-based framework to enhance the computational efficiency and run-time adaptability of inference processes, considering the constraints of available resources. Our proposed model maintains a high degree of analyzability while providing greater freedom than static dataflow models in being able to manipulate the computations associated with inference process at run-time.

The fifth contribution of the dissertation builds on insights developed in the fourth, and introduces a new parameterized design approach for image-based perception that enables efficient and dynamic reconfiguration of convolutions using channel attention. Compared to switching among sets of multiple complete neural network models, the proposed reconfiguration approach is much more streamlined in terms of resource requirements, while providing a high level of adaptability to handle unpredictable and dynamically-varying operational scenarios.

Graph-based Methods for Interpretable and Efficient Machine Learning

by

Yujunrong Ma

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy 2024

Advisory Committee:
Professor Shuvra S. Bhattacharyya, Chair/Advisor
Professor Kiminori Nakamura, co-Chair/Advisor
Professor Behtash Babadi
Professor Sahil Shah
Professor Yang Tao, Dean's Representative

© Copyright by
Yujunrong Ma
2024

Acknowledgments

My sincerest appreciation extends to the numerous individuals who have contributed to the realization of this thesis, making my graduate experience truly unforgettable.

I am particularly grateful to Professor Shuvra S. Bhattacharyya for the unparalleled opportunity to engage in challenging and fascinating projects over the past five years. His availability for guidance and support has been invaluable. Working alongside and learning from such a distinguished mentor has been an honor.

Gratitude is also due to my co-advisor, Dr. Kiminori Nakamura, whose extraordinary domain expertise was critical to the feasibility of this thesis. Additionally, I am thankful to Professor Behtash Babadi, Professor Sahil Shah, and Professor Yang Tao for their willingness to serve on my thesis committee and for their thorough review of my manuscript. I also extend my thankfulness to Dr. Jani Boutellier, who generously supported me throughout various research projects.

The unwavering support and understanding from my family, especially my parents, have been the cornerstone of my resilience and success against all odds. My partner Wenxi has been a constant source of support and care. Their strength has been my refuge in those challenging times.

I am equally grateful to my friends, lab mates, and home mates, whose encouragement was pivotal to my success. Mario and Peaches, my beloved cats, have been a

comforting presence, bringing light during darker times.

Lastly, thank you all!

Table of Contents

1 Introduction 1

2 EADTC: An Approach to Interpretable and Accurate Crime Prediction 5
- 2.1 Chapter Overview . . . 5
- 2.2 Introduction . . . 6
- 2.3 Background and Related Work . . . 9
  - 2.3.1 Interpretability . . . 9
  - 2.3.2 Commonly-Used Models . . . 10
  - 2.3.3 Application of Evolutionary Algorithms to Machine Learning . . . 12
  - 2.3.4 Fairness . . . 13
- 2.4 Methods . . . 13
  - 2.4.1 Decision Tree Modeling in EADTC . . . 14
  - 2.4.2 Evolutionary Algorithm . . . 15
  - 2.4.3 Fairness and Equity . . . 17
- 2.5 Experiments . . . 19
  - 2.5.1 Data Preparation . . . 20
  - 2.5.2 Evolutionary Algorithm Hyperparameters . . . 22
  - 2.5.3 Model Performance . . . 23
  - 2.5.4 Fairness Metrics . . . 25
- 2.6 Conclusions . . . 27

3 Balancing Fairness and Accuracy in Criminology Using Multi-objective Optimization Methods 29
- 3.1 Chapter Overview . . . 29
- 3.2 Introduction . . . 30
- 3.3 Background and Related work . . . 32
  - 3.3.1 Fairness in Criminology . . . 32
  - 3.3.2 Multi-objective EAs . . . 34
- 3.4 Proposed Learning Model . . . 35
- 3.5 Experiments . . . 40
  - 3.5.1 Data Preparation . . . 40
  - 3.5.2 Experimental Setup . . . 41
  - 3.5.3 Optimization of Accuracy . . . 42
  - 3.5.4 Joint Optimization of Accuracy and Fairness . . . 43
  - 3.5.5 Pareto Front Evolution . . . 46
- 3.6 Conclusion . . . 48

4 Towards Interpretable, Attention-Based Crime Forecasting 53
- 4.1 Chapter Overview . . . 53
- 4.2 Introduction . . . 53
- 4.3 Background and Related Work . . . 56
- 4.4 Proposed Learning Model . . . 60

4.4.1 A General Framework of Attention-based ConvLSTM 61
4.4.2 Post-hoc Interpretability of Attention Models 64
4.5 Experiments 66
4.5.1 Moving MNIST Dataset 66
4.5.2 Baltimore City Crime Dataset 68
4.5.3 Results 69
4.5.4 Visualization of Feature Maps 71
4.6 Discussion and Conclusion 71

5 Decidable variable-rate dataflow for heterogeneous signal processing systems 74
5.1 Chapter Overview 74
5.2 Introduction 75
5.3 Background and Related Work 77
5.4 The proposed VR-PRUNE MoC 80
5.4.1 VR-PRUNE actor types 81
5.4.2 VR-PRUNE design rules 83
5.4.3 VR-PRUNE subgraphs and control tables 84
5.4.4 VR-PRUNE MoC summary 86
5.5 Experimental results 87
5.5.1 Adaptive digital predistortion 88
5.5.2 Dynamic-update predistortion filter 90
5.5.3 Adaptive Convolutional Neural Network 91
5.6 Conclusion 92

6 Dynamically Reconfigurable Perception using Dataflow Parameterization of Channel Attention 96
6.1 Chapter Overview 96
6.2 Introduction 97
6.3 Background and Related Work 99
6.3.1 Channel Attention 100
6.3.2 Pruning CNN Channels 101
6.3.3 Dataflow-based Modeling of DNNs 102
6.4 Methods 104
6.4.1 Restrictions on the Decomposition Process 105
6.4.2 Decomposing MLP Layers 107
6.4.3 Decomposing Convolutional Layers 109
6.4.4 Decomposition Algorithm 111
6.4.5 Dataflow Mapping 111
6.5 Experimental results 115
6.6 Conclusion 120

7 Conclusion and Future Work 123
7.1 Conclusion 123
7.2 Future Work 125

Chapter 1

Introduction

Recent discussions about ChatGPT [1] have shown that people are getting increasingly interested in, as well as concerned about the capabilities of machine learning models. In numerous application areas, such as traffic forecasting [2] and pattern recognition [3], the use of machine learning models has been saving large amounts of human labor. However, for some high-stakes application areas, such as medical diagnosis and criminal justice, people demand more evidence to have confidence that the models are understandable, unbiased and trustworthy [4]. These desirable attributes of machine learning model are highly dependent on whether we can interpret the models with a reasonable amount of effort and with sufficient clarity.

A straightforward way to make a model more interpretable is to lower its complexity. Although there is no clear boundary between what are considered interpretable and non-interpretable models, simple models, such as shallow single decision trees [5], and logistic regression [6] models, are usually considered as fully interpretable. This is because their decision-making paths can be traced easily by humans, and humans can readily simulate the inference process in their minds.

Ensemble-tree algorithms, which use multiple decision trees together with some way of aggregating or "voting" across the decisions produced by individual trees, are popular for tabular data, but suffer from reduced interpretability compared to single decision tree

algorithms — especially for ensembles that involve large numbers of trees. The gradient boosting decision tree approach is an example of a well-known approach for ensemble-tree algorithms [7].

Deep learning models have become very popular in numerous application areas, and their popularity and application continues to grow. The most powerful deep learning models can have millions or even billions of parameters. The complexity of these models typically makes their underlying decision making processes impossible to fully understand by humans.

Rather than trying to expose direct insight into their decision-making process, many researchers are taking a different path to explaining the behavior of deep learning models — this is through a concept called post-hoc interpretability. There are various forms of post-hoc interpretability, such as contribution visualization [8], feature importance analysis [9], and sensitivity analysis [10], which have been developed for different types of deep learning models and different applications. Although post-hoc interpretability techniques generally do not allow people to fully understand the details of the decision making process, such techniques can build more trust in models and also they can identify bias to some extent.

In this dissertation, we investigate the use of graph-based methods to address problems of *interpretability*, including post-hoc interpretability, and *efficiency* for practical deployment of state-of-the-art machine learning models. We propose a new type of interpretable machine learning model that is based on the use of graphical representations in the form of single decision trees, where the decision tree structures are optimized using evolutionary algorithms. We also extend our evolutionary algorithm framework into a

multi-objective optimization context — specifically, to address the joint optimization of accuracy and fairness in criminology and criminal justice (CCJ) applications.

We refer to our developed evolutionary algorithm approach as the Evolutionary Algorithm Decision Tree for Crime prediction (EADTC) method, and we refer to its multi-objective extension as the Evolutionary algorithm for multi-objective EQuity and RP optimization (E2QRP), where RP stands for Recidivism Prediction. Optimized models derived using the EADTC or E2QRP methods are self-interpretable models, where the underlying decision processes of the models are exposed directly.

For CCJ applications, we also develop a post-hoc interpretability method, which applies self-attention to a Convolutional Long Short-Term Memory (ConvLSTM) model and then modifies the attention map to better interpret the contributions of the different input features. We demonstrate the effectiveness of this method through experiments on real-world crime data. We show that our model produces accurate crime predictions while providing insightful visualization that enhances the interpretability of prediction results.

For efficiency optimization of machine learning models and other types of signal and information processing applications, we develop a novel dynamic dataflow-graph-based framework called the Variable Rate PSM Runtime Environment. (VR-PRUNE), where PSM stands for Parameterized Sets of Modes. VR-PRUNE is designed to enhance the computational efficiency and run-time adaptability of inference processes, considering constraints of the available processing resources. The VR-PRUNE model of computation maintains a high degree of analyzability while providing greater freedom than static dataflow models in being able to manipulate the computations associated with inference processes at run-time.

Additionally, we develop a novel machine learning framework using dynamic dataflow graphs and post-hoc interpretability analysis to streamline inference efficiency subject to a given set of resource constraints. This framework is designed specifically to facilitate deployment of machine learning models at the network edge, where available computational resources are often very limited, and operating conditions can be highly unpredictable.

Our new methods for decision-tree- and dataflow-graph-based machine learning are demonstrated in the context of applications in CCJ, as described above, and on embedded computer vision applications, respectively.

The remainder of this dissertation is organized as follows. Chapter 2 and Chapter 4 present the EADTC and E2QRP algorithms, which address important CCJ applications in single- and multi-objective contexts, respectively. In Chapter 4, we present our post-hoc interpretability framework designed around ConvLSTM methods. Chapter 5 presents the VR-RPUNE model of computation and demonstrates its effectiveness through a number of design examples. Chapter 6 presents our methods for integrating dynamic dataflow graph modeling and post-hoc interpretability analysis to optimize inference efficiency on resource-constrained devices. Chapter 7 discusses directions for proposed future work.

Chapter 2

EADTC: An Approach to Interpretable and Accurate Crime Prediction

2.1 Chapter Overview

Machine learning applications related to high-stakes decisions are often surrounded by significant amounts of controversy. This has led to increasing interest in interpretable machine learning models. A well-known class of interpretable models is that of decision trees (DTs), which mirror a common strategy used by humans to arrive at solutions through a series of well-defined decisions. However, much of previous research on DTs for criminal justice predictions has focused primarily on collections (ensembles) of DTs whose results are aggregated together. Such DT ensembles are used to help improve accuracy; however, their increased complexity and deviation from human decision-making processes makes them much less interpretable compared to single-DT approaches. In this chapter, we present a new DT model for criminal recidivism prediction that is designed with high interpretability, accuracy, and fairness as core objectives. The interpretability of the model stems from its formulation in terms of a single DT structure, while accuracy is achieved through an intensive optimization process of DT parameters that is carried out using a novel evolutionary algorithm. Through extensive experiments, we analyze the performance of our proposed EADTC (Evolutionary Algorithm Decision Tree for Crime prediction) method on relevant datasets. Our experiments show that the EADTC approach achieves competitive accuracy and fairness with respect to state-of-the-art ensemble DT models,

while achieving higher interpretability due to the simpler, single-DT structure.

Material in this chapter was published in partial, preliminary form in [11].

2.2 Introduction

Machine learning techniques have been increasingly studied and deployed in high-stakes decision making contexts, where the safety and well being of the public, especially vulnerable members, can be impacted, such as medical diagnosis [12], health care [13], and criminal justice [14]. In criminal justice settings, risk assessment, which assesses the likelihood of criminal recidivism and other adverse outcomes, has been an area that has received the most attention in terms of machine learning research. While machine learning techniques can improve predictive accuracy and promote better outcomes for disadvantaged populations over human judgment [15], the techniques have also been a source of concern, especially due to their limited transparency [16].

In recent years, there has been increasing attention to the transparency and interpretability of machine learning models, precisely because limited interpretability, or the difficulty for humans to comprehend why and how predictions are made, can diminish the credibility and legitimacy of decisions based on these models [17]. Even models that provide very high accuracy can fail to be convincing if their operational process is encapsulated within a black box or is otherwise unexplainable.

There have been increasing research efforts on improving the interpretability of machine learning models. These efforts can be classified into two main types. The first type focuses on explaining black box models. Some studies have used more interpretable

models, such as shallow feed-forward networks and decision trees, to mimic and explain how the black-box models work [18] [19], while others have used techniques such as guided class activation maps to analyze the models post-hoc [20]. The second type of effort focuses on directly designing models that are inherently interpretable [21] [22]. Unlike the models that mimic the output of black-box models but still do not allow for the understanding of how predictions are generated, these interpretable models provide predictions that can be easily traceable to well-defined decision rules.

In this chapter, we present a new decision tree (DT) model for recidivism prediction that is designed to provide high interpretability while maintaining competitive predictive accuracy. DTs have been demonstrated to have various advantages for interpretability: 1) the hierarchical sequence of well-defined decision rules in a DT is completely transparent and similar to multi-step decision-making processes used by humans; 2) the complexity of a DT is easy to control (e.g., by setting the maximum depth of the tree); and 3) the feature selection process can be automated, which safeguards against subjective bias that can be introduced by manual filtering and configuration of features.

Prior development of DT-based models shows that ensemble methods (e.g., random forest, gradient boosting), which incorporate multiple trees and aggregate the predictions produced from the different trees, can have a better regularization performance — leading to increased predictive accuracy — compared to single-DT models (e.g., see [23, 7, 24]). However, their increased complexity and deviation from human decision-making processes makes ensemble-DT models much less interpretable compared to single-DT approaches.

The DT method that we introduce in this chapter is designed as a single-DT model to provide high interpretability. To overcome the potential degradation in accuracy compared

to ensemble-DT models, we incorporate a novel evolutionary algorithm approach to intensively optimize parameters of the proposed single-DT framework. We refer to our integrated evolutionary algorithm and DT method as the EADTC (Evolutionary Algorithm Decision Tree for Crime prediction) method.

Evolutionary algorithms are population-based optimization algorithms that are inspired from natural selection in biological species, and are effective in their ability to escape local optima. An evolutionary algorithm is initialized with a random population of candidate solutions. This population then evolves for multiple generations until a given stopping criterion is reached. A fitness function, which is an important design component of an evolutionary algorithm, is used to assess the quality of a given candidate solution. The process of deriving the next generation G_{i+1} of an evolutionary algorithm population from the current generation G_i generally involves selecting pairs of candidate solutions (parents) from G_i, and deriving one or more candidate solutions (children) in G_{i+1} from each pair of selected parents. The fitness function is used to guide the selection of parents and other aspects in the evolution of generations. For more details on evolutionary algorithms, we refer the reader to [25].

In our prototyping and experiments presented in this chapter, we focus on the task of predicting criminal recidivism, which typically refers to the return to criminal behavior or other undesirable outcomes after contact with the criminal justice system. In the remainder of this chapter, we assume the context of recidivism when defining details of the EADTC approach, and presenting experiments that demonstrate the approach. We present the results on both prediction accuracy and fairness metrics. Although the developments in this chapter are focused on recidivism, the underlying DT framework is flexible, and we

envision that it can be adapted to other predictive applications in criminal justice.

2.3 Background and Related Work

In this section, we first discuss defining elements of model interpretability in the literature and how they vary across most commonly used machine learning models, with special attention to applications in criminal justice settings. We then discuss the application of evolutionary algorithms to machine learning models, followed by a discussion on fairness metrics.

2.3.1 Interpretability

While the concept of interpretability can be domain specific, three distinct aspects have been identified [26]: (1) Simulatability: the ability that a model allows humans to use input data together with a model to reproduce every calculation step necessary to make the prediction. For example, this allows humans to understand changes in model parameters caused by the training data. (2) Decomposability: the feature that a model can provide intuitive explanations for all of the model parameters. (3) Algorithmic transparency: the ability to explain the working of the training algorithm that optimizes the parameters of a model. In the following section, we review several classes of machine learning models that are commonly used for recidivism prediction in light of these aspects of interpretability.

2.3.2 Commonly-Used Models

(1) Logistic Regression. Unlike the usual hard-label output of most predictive models, the probability output of logistic regression makes it easier to reason about relative risk levels between different input instances. Furthermore, the interpretability of logistic regression is considered as being in the top-tier. This is because in a logistic regression model it is inherently clear how much weight is being given to each feature in the model. An important variant of logistic regression in the context of interpretable recidivism prediction is the concept of scoring systems, such as RiskSLIM, where feature weights are constrained to be integers [21]. Such scoring systems can improve interpretability, especially decomposability, since integer-valued weights are generally easier for humans to understand compared to weights that can have arbitrary real values.

(2) Decision Trees. A decision tree (DT) is a decision support tool that uses a branching model of decisions and their possible consequences. A DT is a graphical, tree-based model in which vertices represent decisions, and edges from a parent vertex to its children correspond to different decision outcomes associated with the parent vertex [27]. For example, random forests, an ensemble-DT method, has been applied to predict domestic violence for informing pretrial release decisions [28]. When examining a prediction produced by this approach, one can see how the prediction has been arrived at by examining selected paths in the random forest that are associated with the prediction. In particular, there is one path to examine through each constituent tree whose output agrees with the prediction result of the ensemble. This property of identifying path-sets for prediction results provides good simulatability. However, compared to single-DT models, interpretability of ensemble

models is significantly weaker. This is in part because the relationship between multiple paths within a given path-set is in general not clear for a given prediction result.

(3) Boosting-based Models. Models that incorporate boosting optimization are among the best performing models on tabular data (e.g., see [7]). Boosting models involve the integration of relatively low-complexity "weak learners" into higher accuracy "strong learners" that combine the results of the weak learners. The integration of weak learners into a strong learner is performed through an iterative optimization process. Despite their relatively high accuracy, the simulatability of boosting-based methods is considered to be relatively weak. This is because the iterative optimization process and the process of combining results from different weak learners introduce complexities that make the resulting models difficult to interpret.

(4) Deep Neural Networks (DNNs). DNNs and their variants are widely-used in many application areas, such as computer vision and natural language processing, to name a few. In recent years, DNNs have been applied to crime prediction (e.g., see [29]). However, the underlying mechanisms of DNNs are usually considered as being non-interpretable. This is due in part to the uncertainty and ambiguity introduced by non-linearities in the employed computational process. Non-linear layers, such as Rectified Linear Unit (ReLU) layers, can significantly increase the expressive power of DNNs, but the introduction of such components comes with a corresponding decrease in interpretability. Moreover, DNNs typically involve very large numbers of parameters, which further degrades their interpretability — both in terms of decomposability and simulatability.

2.3.3 Application of Evolutionary Algorithms to Machine Learning

Evolutionary algorithms have been applied to crime prediction independently of machine learning methods. For example, Wu and Grubesic used a multi objective evolutionary algorithm to identify crime hot-spots [30]. On the other hand, evolutionary algorithms have been used to construct DNN models, such as recurrent neural networks (RNNs) [31], convolutional neural networks (CNNs) [32] and hybrid networks [33]. Evolutionary algorithms have also been used to construct rule-based models [34]. Due to characteristics of their structure, tree-based models require special consideration when designing encodings, and crossover and mutation operators for integration with evolutionary algorithms. Issues in the design of encoding and operators for EA-integrated tree models, such as deleting nodes and subtrees, were discussed in [35], and modified operators, such as subtree transplanting, were introduced in [36].

To the best of our knowledge, the EADTC method is the first method for recidivism prediction that integrates evolutionary algorithms and single-DT machine learning models. In particular, the framework of evolutionary algorithms is applied to provide powerful optimization capability that enhances the quality of single-DT crime prediction. The result is a novel method that achieves accuracy and fairness levels that are competitive with state-of-the-art prediction methods based on ensemble-DT models, while providing the significantly enhanced interpretability of single-DT models. Our work leads to a new way of studying machine learning for recidivism prediction in terms of evolutionary algorithms and single DT modeling.

2.3.4 Fairness

In addition to accuracy and interpretability, *fairness* is another important criterion for developing predictive models in many application areas, including criminal justice. There is concern that being heavily dependent on machine learning can cause so-called run-away loops, where imbalanced data causes models to produce imbalanced results, which leads to decisions that exacerbate imbalances in the data, which in turn leads to more imbalanced models, and so on. There are different definitions of fairness in the literature. Corbett-Davies et al. [37] describe three popular definitions of algorithmic fairness: statistical parity, conditional statistical parity and predictive equality.

2.4 Methods

The proposed evolutionary algorithm approach, EADTC, provides an alternative to conventional approaches for single-DT optimization, such as the well-known top-down induction of decision trees (TDIDT) approach [27]. Most of the concepts in genetic algorithms, which is the most popular type of EA, are adopted in the proposed method. EADTC enables exploration of a large search space of possible DT configurations, which is especially useful when considering applications such as crime prediction, where the features involved can be diverse and can be interrelated in complex ways. Indeed, the ability of EADTC to explore a large search space, by leveraging the randomized search process of the general evolutionary algorithm framework, enables intensive optimization of single-DT structures. As we demonstrate in the next section, the optimized single-DT structures produced by EADTC are competitive in performance with more complex (less interpretable)

ensemble-DT structures.

2.4.1 Decision Tree Modeling in EADTC

When deriving a DT for recidivism using the EADTC approach, we are given a dataset $D = I_1, I_2, \ldots, I_n$ consisting of n instances, where each instance I_i is assigned a binary classification label $L(I_i) \in \{0, 1\}$. Intuitively, each I_i corresponds to a unique individual (person) and $L(I_i)$ indicates whether or not the individual experienced recidivism — $L(I_i) = 1$ indicates that individual I_i experienced recidivism, whereas $L(I_i) = 0$ indicates the absence of recidivism. Part of the design of the DT is the definition of a set of features $\{f_1, f_2, \ldots, f_n\}$, where each feature f_i is a mapping of instances: $f_i : D \to \mathcal{R}$, where $\mathcal{R}$ denotes the set of real numbers. For a given instance $I_i \in D$, the vector $F(I_i)$, defined by $F(I_i) = (f_1(I_i), f_2(I_i), \ldots, f_m(I_i))$, is referred to as the *feature vector* associated with I_i. Examples of features that are of relevance in recidivism prediction are an individual's current age, first-crime age, employment status, features related to any history of drug abuse, and violent-crime flag.

Following common practice in the design and implementation of DTs, we restrict our attention to binary DTs — that is, trees in which each internal vertex has exactly two child vertices. For each internal vertex, one of the child vertices is designated as the *left* child, while the other is designated as the *right* child. In the EADTC approach, each internal vertex v in a DT is associated with a unique feature index $k(v) \in \{1, 2, \ldots, m\}$, and a decision function $\phi_v : f_{k(v)}(D) \to \{0, 1\}$, which effectively maps feature vectors (using only the $k(v)$th component) into decision paths that follow the edge to the the left

(0 value) or right (1 value) child vertex, respectively.

In the EADTC approach, each leaf vertex w in a DT is associated with a predicted value $p(w) \in \{0, 1\}$. Intuitively, to compute the recidivism prediction result for a given instance I_i, we start at the root vertex $v[0]$ of the DT, compute $z_0 = \phi_{v[0]}(f_{k(v[0])}(I_i))$, and then set the next vertex $v[1]$ to visit by selecting the left or right child of $v[0]$ depending on whether z_0 is 0 or 1, respectively. We then compute $z_1 = \phi_{v[1]}(f_{k(v[1])}(I_i))$, and similarly, set the next vertex $v[2]$ depending on the value of z_1. This process is repeated $(d-3)$ more times, where d is the depth of the tree until we arrive at a leaf vertex $v[d-1]$. The value predicted by the DT model for $I(i)$ is then derived as $p(v[d-1])$.

In our experiments, the DT is trained with some subset of D and tested with the remaining elements of D. We randomly select 75% of the instances for training. More details on our experiments are discussed in Section 2.5.

2.4.2 Evolutionary Algorithm

The design of an evolutionary algorithm can be divided into several key components: the representation, parent selection process, crossover operator, mutation operator, and termination condition. In the remainder of this subsection, we present the design of these components in EADTC. For background on how these components are used in the overall operation of an evolutionary algorithm, we refer the reader to [25].

(1) Representation: Representation concepts include the genotype, phenotype and representation mapping. For example, we may represent an integer (phenotype) as a binary array (genotype): 37 as 00100111 and 116 as 01110100. In EADTC, we adopt a

representation that is specifically geared toward decision trees. For a given decision tree $T = \{n_1, n_2, \ldots, n_k\}$, the representation encodes the set of nodes in T, where each node is a tuple of the form $n(d, f, t, l, r)$. Here, d is the depth of n within T, f is the feature associated with n that determines the decision outcome associated with the node, t is the threshold associated with the decision, and l and r are the left and right children of n.

(2) Parent Selection: The parent selection process is used to select parent individuals for crossover operations, where the suitability of a parent individual is quantified by its fitness. In EADTC, we use accuracy as the fitness function and proportional fitness as the selection metric. If $Fit(T_i)$ indicates the fitness of tree T_i, $N_{correct}$ and N_{all} represent the number of correctly classified records and the total number of records, respectively, and $P(T_i)$ is the probability for tree T_i to be selected as a parent, then we can write:

$$Fit(T_i) = \frac{N_{correct}}{N_{all}} \tag{2.1}$$

$$P(T_i) = \frac{Fit(T_i)}{\sum_{j=1}^{n} Fit(T_j)} \tag{2.2}$$

(3) Crossover Operators: In addition to determining how pairs of parents are selected for crossover, we need to determine how the characteristics of the parents are used to derive offspring through the crossover process. Typically, the crossover operator merges the components of two parents in some way to construct a new individual (candidate solution). For example, the individual parent representations may be split into parts, and then selected parts from the two parents may be mixed to derive a new individual. For the tree representation in EADTC, the crossover operator exchanges a subtree of one parent

with the corresponding subtree in the other parent. Fig. 2.1 illustrates this type of crossover operation.

(4) Mutation Operators: Mutation operators simulate the biological mechanism of mutation, where certain parts of a genome may undergo random changes in nature. Mutation operators for tree-based models in EAs include: (a) replacing the decision feature or threshold within a node — $M(n(d, f, t, l, r)) = n(d, \hat{f}, \hat{t}, l, r)$, where M is the mutation operator and $\hat{f}, \hat{t}$ give the new feature and threshold values; and (b) replacing a subtree by a leaf $n(d, class, None, None, None)$. As shown in Fig. 2.1, the right subtree of the individual can mutate into a leaf node.

(5) Termination Condition: The termination condition specifies when an evolutionary algorithm should end. Commonly, the stopping condition is specified in terms of a fixed number of generations through which the EA should execute. In EADTC, we incorporate a tolerance term, which stops evaluating new generations when a certain number of consecutive generations does not produce any improvement. This allows the EA to continue until it converges, at least in a local sense of convergence.

2.4.3 Fairness and Equity

In fairness-integrated approaches to machine learning, some features may be designated as being associated with *protected groups*, which means that the features should not be directly used as input to the models. However, even when direct indicators of protected group membership, such as race and gender, are not included as input, some correlations between these measures and legitimate predictors can result in unfairness. This enlightens

us that only discarding sensitive information is not enough to eliminate bias; we also need to define fairness and design corresponding metrics. As seen in the recent literature on fairness and recidivism prediction, it is not possible to maximize accuracy and all types of fairness at the same time [27] (see also [38] [39]).

Two useful metrics for assessing algorithmic fairness in recidivism prediction are conditional procedure error CPE and conditional use error CUE associated with a given protected group (e.g., a race category).

Suppose that we have a predictive model Z that is tested on a dataset D that includes members from multiple protected groups, and the underlying prediction problem involves discrimination between two classes, which we refer to as positive and negative classes. For example, in our experiments in Section 2.5, we consider the positive and negative classes to correspond to high-risk and low-risk members, respectively, of the population being studied.

The CPE and CUE each have two components. For CPE and a given protected group g, these components are defined as follows: $\alpha_1(g) = FN(g)/(TP(g) + FN(g))$, $\alpha_2(g) = FP(g)/(FP(g) + TN(g))$, where $TP(g)$, $FP(g)$, $TN(g)$, and $FN(g)$ respectively represent the counts of true positives, false positives, true negatives, and false negatives in the given testing experiment for Z across all instances in the given dataset δ that are associated with protected group g. Similarly, for CUE, there are two components, which we can express as $\beta_1(g) = FP(g)/(TP(g) + FP(g))$, and $\beta_2(g) = FN(g)/(FN(g) + TN(g))$.

Given two protected groups g_1 and g_2, we can define disparity metrics associated with their CPE and CUE values as follows:

$$\Delta_\alpha(g_1, g_2) = \frac{abs(\alpha_1(g_1) - \alpha_1(g_2)) + abs(\alpha_2(g_1) - \alpha_2(g_2))}{2},$$

and similarly,

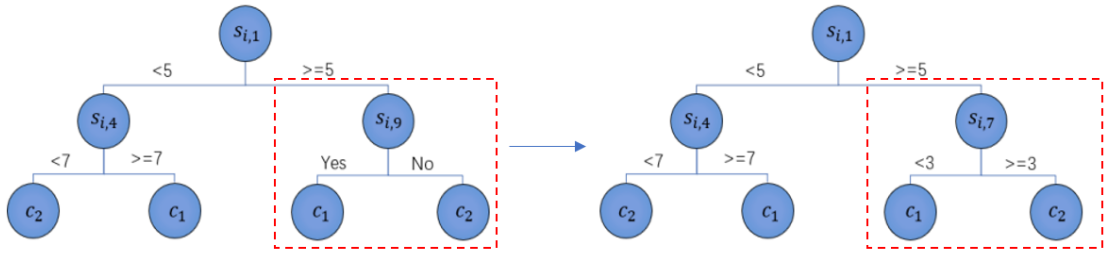
$$\Delta_\beta(g_1, g_2) = \frac{abs(\beta_1(g_1) - \beta_1(g_2)) + abs(\beta_2(g_1) - \beta_2(g_2))}{2},$$

where $abs(x)$ represents the absolute value of the given number x. Observe from this formulation that Δ_α and Δ_β always lie in the interval $[0, 1]$.

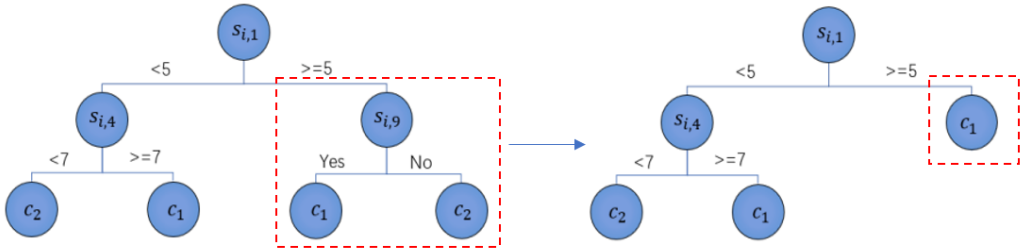
The concepts of CPE and CUE are used in two forms of fairness assessment, which are referred to as conditional procedure accuracy equality (CPAE) and conditional use accuracy equality, respectively (CUAE). CPAE is said to be achieved across a given set $G = \{g_1, g_2, \ldots, g_n\}$ of protected groups when $\Delta_\alpha(g_i, g_j) = 0$ for all $i, j \in \{1, 2, \ldots, n\}$. Similarly, CUAE is said to be achieved across G when $\Delta_\beta(g_i, g_j) = 0$ for all $i, j \in \{1, 2, \ldots, n\}$.

2.5 Experiments

We developed a prototype implementation of the EADTC method using Python 3.7 along with sci-kit of version 0.23.2. In this section, we report on experimental results that we derived from this implementation. The evolutionary algorithm in EADTC follows a typical procedure imitating the natural selection process: representing individuals, parent selection, crossover and mutation (e.g. see [25]). For EADTC, most of the steps are similar to other EA applications, but due to the special tree representation of each individual, we made



(a) Crossover example, right subtree comes from the other parent.



(b) Mutation example, right subtree mutates to final class c_1.

Figure 2.1: Illustrations of crossover and mutation operators in EADTC. Here, c_1 and c_2 represent the two classes for classification, and $s_{i,j}$ represents the jth feature of the ith sample.

adjustments to traditional bit-wise crossover and mutation and directly exchanged nodes between parents as shown in Fig. 2.1.

Key hyperparameters involved in the evolutionary algorithm are summarized in Section 2.5.2.

2.5.1 Data Preparation

We used two data sets for the experiments. The first one, called *Recidivism of Prisoners Released in 1994*, is provided by ICPSR (Inter-University Consortium for Political and Social Research)[1], and includes 38,624 release records from 1994. The records were randomly sampled from 302,309 prisoners in 15 states (Arizona, California, Delaware, Florida, Illinois, Maryland, Michigan, Minnesota, New Jersey, New York, North Carolina,

[1]IRB reference number: 1793282-2 Machine Learning Recidivism Assessment

Ohio, Oregon, Texas, and Virginia) [40]. The released individuals were tracked for three years following their release. The features include demographic information, such as date of birth, sex, and race. The features also include arrest information, such as the date of arrest, prior arrests, and an out-of-state indicator; as well as sentencing and prison term information, such as the length of sentence, crime type, and time served.

We first determined a set of features that we expected to be especially useful for purposes of prediction. The selected features are summarized in Table 2.1. The age variables are calculated by simply subtracting the birth date from the arrest/release date. Non-categorical features were scaled to a form with zero mean and unit variance, which helps to avoid overfitting on features that have larger magnitudes.

Table 2.1: Variables from the ICPSR dataset that were used in our experiments.

Variable	Dtype	Description
SEX1	int64	Sex of prisoner
LENG_SEN	int64	Sentence length
DRUGAB	bool	Drug abuser
ALCABUS	bool	Alcohol abuser
DRUGTRT	bool	Drug treatment
ALCTRT	bool	Alcohol treatment
DFM	bool	Felony or misdemeanor
DDMV	bool	Domestic violence involved
DFIR	bool	Firearms involved
PRIR	int64	Number of prior arrests
RELTYP	category	Release type
NFRCTNS	int64	Number of infractions
TMSRV	int64	Length of prison term served
AGE_1stARR	float64	Age at first arrest
AGE_1stRLS	float64	Age at release

The second recidivism dataset that we used represents individuals in Pennsylvania state prisons. The individuals represented in this dataset were released into parole supervision between 2006 and 2008. The set of features that we used for this second dataset,

which we refer to as the PA dataset, is summarized as shown in Table 2.2.

Table 2.2: Variables from the PA dataset that were used in our experiments.

Variable	Dtype	Description
SEX	int64	Sex of prisoner
AGE	int64	Age at release
VIO	bool	Violent crime
PRO	bool	Property crime
DRUG	bool	Drug crime
PUBLIC	bool	Public order crime
OTHER	bool	Other crime
CRT_ADM	bool	Court admission
TM_SERVE	bool	Length of prison term served
FIRST	bool	First-time incarceration
LSIR	int64	Risk Score

For both datasets, the label we targeted on is whether the released prisoner experienced a rearrest within 3 years of release.

2.5.2 Evolutionary Algorithm Hyperparameters

In our experiments, the settings for key hyperparameters in the evolutionary algorithm engine underlying EADTC are selected from a dense grid search, and the finalized set is as follows.

- `n_trees = 100` (number of decision trees in each generation).
- `max_depth = 6` (maximum depth of each decision tree).
- `max_iter = 4000` (maximum number of iterations to run the evolutionary algorithm).
- `early_stopping = 200` (number of iterations without improvement that triggers early termination of the evolutionary algorithm).

- `selection = proportional_fitness` (use fitness-proportional selection to select parents for recombination).
- `mutation_prob = 0.1` (probability of mutating an individual).
- `selected_parent = 10` (the number of parents to be kept for the next generation).

These parameters were determined empirically through iterative experimentation.

2.5.3 Model Performance

All experiments were carried out with 4-fold cross validation. We first compared the prediction accuracy of EADTC with several relevant models from the literature: logistic regression (LR), which uses a linear combination of variables to estimate a probability for each class [6]; ridge regression, which is a variation of linear regression [41]; the classification and regression tree (CART) method, which is a modern variation on the single decision tree approach, as described in [27]; the random forest (RF), which is an ensemble of decision trees, and provides the foundation for the approach of Berk and Bleich (see Section 2.3.2); gradient boosting decision trees (GBDTs), which use gradient descent techniques for loss minimization that are generalized to trees [7].

The results of our comparison between EADTC and the previously-developed models are summarized in Table 2.3. The accuracy values are in the range of $[0, 1]$, where an accuracy of 1 represents 100% accuracy.

Given that recidivism data can be highly unbalanced, the Area Under the Receiver Operator Characteristic Curve (AUC) can provide better insight, compared to prediction

Table 2.3: Comparison between EADTC and several reference models in terms of prediction accuracy

	LR	Ridge	CART	RF	GBDT	EADTC
ICPSR	0.693	0.675	0.688	0.701	**0.708**	0.697
PA	0.672	0.665	0.669	0.686	**0.688**	0.676

Table 2.4: AUC comparison of all models

	LR	Ridge	CART	RF	GBDT	EADTC
ICPSR	0.757	0.755	0.757	0.762	**0.769**	0.760
PA	0.701	0.695	0.696	0.715	**0.718**	0.707

accuracy, into the relative performance of different prediction methods. Table 2.4 compares the measured performance of EADTC with the reference models in terms of the AUC metric.

From the results of Table 2.3 and Table 2.4, we can see that EADTC shows improvement both in terms of accuracy and AUC over all of other interpretable models, including CART, logistic regression and ridge regression. This indicates that EADTC is competitive even when handling unbalanced data sets. EADTC also helps to the narrow gap of performance between interpretable models and ensemble models such as RF and GBDT, which lack interpretability.

In addition to the reference models that we experimented with for the comparisons reported in Table 2.3 and Table 2.4, we also compared EADTC with the RiskSLIM approach [21] discussed in Section 2.3.2. However, we did not include RiskSLIM in our comparison tables. We omitted RiskSLIM from the tables because it strictly requires all features to be Boolean-valued, which can make it more difficult to apply compared to the methods that are included in the tables. In particular, the restriction to Boolean

data types requires manual bagging to handle non-categorical features. The need for such bagging can introduce two problems: (1) models can be difficult to adapt to new datasets, and (2) information can be lost in the bagging process that is useful for our prediction purposes. Although we did not include RiskSLIM in Table 2.3 and Table 2.4, we did experiment with the approach using the ICPSR dataset. We found that the best performance levels we measured on RiskSLIM, after hand-tuning its hyperparameters, are an accuracy of 0.674 and an AUC of 0.731, which are both lower than the corresponding results produced by EADTC. Due to the limitations described above associated with the bagging requirement for RiskSLIM, we did not carry out experiments on the PA dataset using RiskSLIM. Regardless of the performance of RiskSLIM, its use brings about limitations in connection with the objectives of this chapter — in particular, the objective of being efficiently adaptable to different datasets — and warrants further comparative research.

2.5.4 Fairness Metrics

We evaluated the prediction model derived by EADTC in term of fairness metrics defined in Section 2.4.3. We selected the $Race$ attribute as our protected attribute, and examined the protected groups corresponding to white and black members of the released prison population, respectively. We employed the ICPSR dataset in this experiment.

Table 2.5 and Table 2.6 show the confusion matrices resulting from application of our predictive model on each of the two protected groups. The tables also provide CPE and CUE values. From these tables, we see that our model exhibits similar false rates on the protected groups, where the false rate is simply the sum of the false positive and false

Table 2.5: Confusion matrix for white prisoners.

White	High risk prediction	Low risk prediction	CPE
Rearrest	737	265	0.267
Non-rearrest	371	668	0.361
CUE	0.338	0.288	

Table 2.6: Confusion matrix for black prisoners.

Black	High risk prediction	Low risk prediction	CPE
Rearrest	1263	361	0.171
Non-rearrest	334	420	0.491
CUE	0.225	0.407	

negative rates.

An illustration of the prediction results on protected groups in the form of pie charts is shown in Fig. 2.2.

From the results in Table 2.5 and Table 2.6, we can calculate the disparity metrics we defined in Section 2.4.3 by

$$\begin{aligned}\Delta_\alpha(g_w, g_b) &= \frac{abs(\alpha_1(g_w) - \alpha_1(g_b)) + abs(\alpha_2(g_w) - \alpha_2(g_b))}{2} \\ &= \frac{abs(0.267 - 0.171) + abs(0.361 - 0.491)}{2} \\ &= 0.113\end{aligned}$$

and

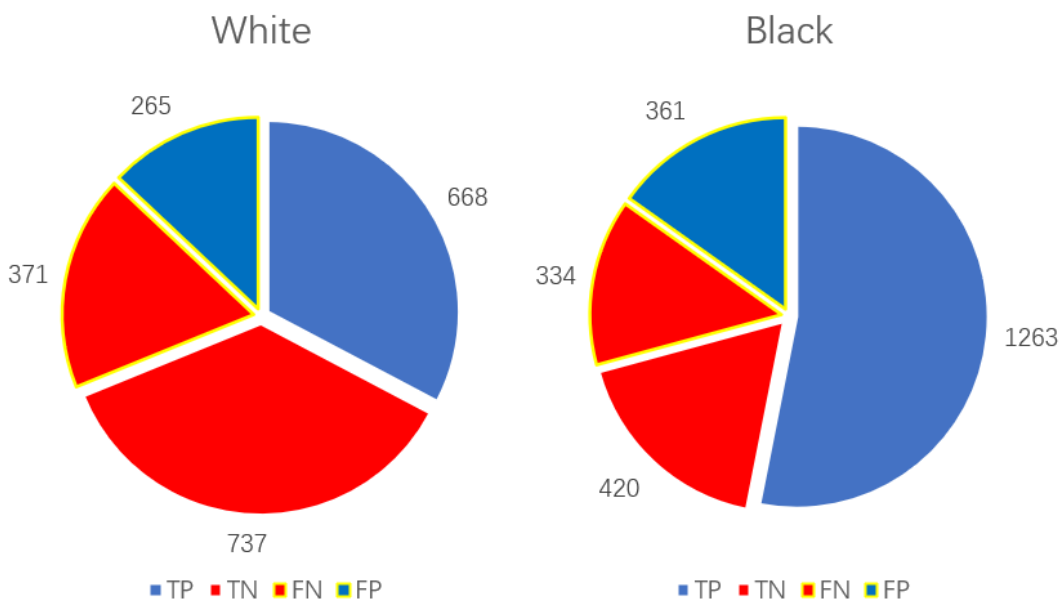


Figure 2.2: Illustration of prediction results on protected groups.

$$\begin{aligned}\Delta_\beta(g_w, g_b) &= \frac{abs(\beta_1(g_w) - \beta_1(g_b)) + abs(\beta_2(g_w) - \beta_2(g_b))}{2} \\ &= \frac{abs(0.338 - 0.225) + abs(0.288 - 0.407)}{2} \\ &= 0.116.\end{aligned}$$

Here, g_b and g_w denote the black and white protected groups, respectively.

2.6 Conclusions

In summary, our work makes the following contributions:

- Instead of introducing more complexity by using a traditional greedy induction strategy, we propose a new evolutionary algorithm algorithm approach, called

EADTC (Evolutionary Algorithm Decision Tree for Crime prediction). EADTC incorporates optimization to increase the effectiveness of the decision tree model without losing interpretability.

- Unlike most of the existing risk assessment models that adopt scoring systems, models produced by EADTC do not depend heavily on domain specific feature engineering, such as age bagging. Additionally, hyper-parameter tuning and data processing are relatively straightforward for tabular data to be compatible with the model.

- We report on comprehensive experiments to analyze models produced by EADTC in terms of relevant metrics of predictive performance and fairness, and to provide comparison against several alternative prediction approaches. The results demonstrate the effectiveness of the EADTC approach in terms of both predictive performance and fairness.

Overall, the EADTC method that we designed in this chapter outperforms single CART while maintaining the same high level of interpretability. While this chapter makes significant contributions in the area of interpretability and transparency in machine learning, future research is warranted to formalize model interpretability and systematically improve key aspects of interpretability that help foster legitimacy in machine learning-supported decision making.

Chapter 3

Balancing Fairness and Accuracy in Criminology Using Multi-objective Optimization Methods

3.1 Chapter Overview

In the field of criminological predictive modeling, the dual challenges of ensuring fairness and maintaining interpretability are crucial. This section addresses these challenges by introducing a new approach to optimizing decision trees using evolutionary algorithms (EAs). Our approach focuses on refining decision trees to achieve a balance between accuracy and fairness, a task complicated by the inherent bias present in historical crime data. By leveraging the principles of multi-objective optimization, our model systematically trades off prediction accuracy and fairness. The evolutionary process characterized by selection, crossover, and mutation is tailored to fit the decision tree structure, ensuring that model development is not only accurate but also ethically sound. Experimental results demonstrate the effectiveness of our approach in providing interpretable and fair predictive models suitable for high-stakes applications in criminology. More broadly, this research makes a significant contribution to the field of explainable machine learning, providing a powerful framework for engineering systems that are transparent, fair, and adaptable to different data environments.

The contributions of this chapter were developed jointly with Xiaowen Qi, another

Ph.D. student at the University of Maryland.

3.2 Introduction

Machine learning models have demonstrated remarkable predictive capabilities across various domains. However, despite their success, the opacity of complex models, particularly deep learning algorithms, has raised significant concerns in high-stake applications. For instance, in medical diagnostics, interpretability is crucial for clinical decision-making and patient trust [42, 43]. In the financial sector, the opaqueness of models can lead to risky investments and regulatory challenges [44, 45]. Similarly, in criminal justice, the black-box nature of these models can result in unfair decision making and diminished legitimacy [46, 11]. Algorithmic fairness in criminal justice emphasizes that decisions made within the system treat people equally, without discrimination based on race, gender, or other protected characteristics.

To address these challenges, we further improve the framework. Our method enhances this innate interpretability, while simultaneously striving to address fairness concerns through multi-objective optimization, where metrics associated with accuracy and fairness are jointly considered. In particular, our method jointly considers the maximization of predictive precision and fairness, where fairness is measured by fairness metrics, such as disparate impact or equal opportunity [47].

In the criminal justice application of recidivism prediction and risk assessment, unfair recidivism predictions can have long-lasting social and economic implications. Biased outcomes may result in the over-penalization of certain populations, resulting in

destructive feedback loops where cumulative effects on criminal records further entrench the same individuals in the criminal justice system [48]. Predictions that are fair and that correct for bias are essential for minimizing such feedback loops and promoting societal well-being.

There are various definitions of algorithmic fairness in recidivism prediction. It has been proved that some of these are mutually exclusive, so theoretically it is not possible to achieve all types of fairness simultaneously in an optimal sense [49]. Our proposed framework allows the user (e.g., a public safety or public policy professional) to select from among a variety of fairness metrics, and to derive an optimized trade-off curve between the selected fairness metric and overall prediction accuracy. The framework is therefore a multi-objective optimization (MOO) framework that allows the user to understand the underlying fairness/accuracy trade-off space quantitatively, and to select an operating point within this space that is found to be most favorable in the desired usage context.

Traditional optimization approaches focus on a single metric (objective) and produce a solution that achieves or approximates the optimal achievable value of the objective. In contrast, MOO approaches consider multiple objectives and produce a set of multiple "candidate solutions", which achieve or approximate the optimal trade-offs among the targeted objectives (e.g., see [50]). The set of solutions produced by an MOO approach is called a Pareto front. Each point p in a Pareto front has the property that no other point on the front outperforms p in all targeted objectives. That is, for distinct points p and q in a Pareto front, either (1) p and q achieve equal values in *all objectives* or (2) p outperforms q in at least one objective, and conversely q outperforms p in another objective.

Our study presents experimental results on real-world datasets to illustrate the effec-

tiveness of the approach. By adopting single decision trees, the model becomes transparent and interpretable, making it suitable for applications requiring human-understandable explanations. Simultaneously, the multi-objective optimization process helps mitigate potential biases, promoting fairness in decision-making. The results showcase that the proposed evolutionary algorithm-driven optimization approach can lead to decision trees that strike a balance between predictive accuracy and fairness, offering a promising path towards responsible and interpretable machine learning.

The remainder of this section is organized as follows. In Section 3.3, we review relevant literature. In Section 3.4, we introduce our proposed method to apply EA methods on single decision trees whose decisions are to be assessed in terms of multiple objectives. In Section 3.5, we conduct a series of experiments — using relevant, real-world datasets — to demonstrate the effectiveness of predictive models that are derived using our EA-based, multiobjective optimization approach. In Section 3.6, we summarize the findings of our section, and present useful directions for future work.

3.3 Background and Related work

In this section, we discuss relevant previous works about the development of EAs, machine learning models, and multi-objective optimization.

3.3.1 Fairness in Criminology

The issue of fairness in predictions is crucial for criminal justice applications due to ethical concerns, the pursuit of social equity and legal consequences. Addressing this

issue is challenging for two major reasons. First, most previous metrics for assessing fairness in algorithms have been binary, classifying outcomes simply as fair or not fair; incorporating quantitative metrics that can take on a range of numeric values is important to consider fairness on a more nuanced scale [51]. Second, enforcing fairness constraints can reduce a model's overall accuracy by necessitating compromises on the predictions for the purpose of equity (e.g., to ensure that inaccurate predictions do not occur more in certain populations compared to others) [52].

The widely used Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) tool for recidivism prediction has come under scrutiny for potential biases [53], sparking extensive debate and research in the field [54, 39].Chouldechova's pivotal analysis underscores the tool's disparate impact on different demographic groups, particularly along racial lines [38]. Selbst's work critically examines the oversimplification of societal complexities in machine learning applications in the criminal justice system, advocating for a multi-stakeholder approach to tackle inherent biases [55]. Hardt et al. propose a constrained optimization framework, focusing on achieving equality while addressing disparate impact [56]. Lipton further explores the trade-offs between model complexity and interpretability, a crucial consideration in ethical AI [57].

Interpretability becomes a key factor in the development of many kinds of machine learning models, including those applied within the criminal justice system. Decision-tree-based methods stand out in this context due to their intrinsic interpretability and simplicity [58]. These methods allow for easy understanding and visualization of the decision-making process. The traceable decision making paths of decision trees make them well-suited for applications where understanding the basis of predictions is paramount [20].

3.3.2 Multi-objective EAs

In this section, we view model accuracy and fairness both as critically important objectives, and we develop a novel MOO approach to optimize the trade-offs between these two objectives. This allows the user to select the trade-off that is found to be most favorable, based on the user's understanding of the application context, while being able to select from a set of candidate trade-offs within an optimized Pareto front. As discussed in Section 3.2, MOO is an important methodology in decision-making processes that involve competing goals [59, 60], where for example, optimization of one goal is not acceptable when it comes with excessive degradation of another.

EAs, renowned for their flexibility and adaptability, are well-suited for MOO [61]. The stochastic nature of EAs, combined with their ability to explore a wide solution space, makes them an effective choice for navigating complex multi-objective landscapes. EAs can efficiently search for Pareto-optimal solutions, balancing between exploration of the solution space and exploitation of the best solutions found. Advancements in MOO and EAs have led to increasingly sophisticated approaches in handling real-world problems (e.g., see [62]).

In the realm of criminal justice, multi-objective EA methods have been investigated in Guedes et al.'s research on multi-agent models for optimizing police dispatch systems [63]. Guedes's work provides an example showing the potential of MOO and EA methods in addressing complex societal challenges, where multiple objectives must be balanced, such as resource allocation, response time, and ethical considerations in law enforcement.

3.4 Proposed Learning Model

Since criminal justice-related predictions require thorough scrutiny and reasoning by human decision makers, decision trees often are favored for their interpretability. To overcome the passive presence of bias in training data, we develop methods in this section to derive decision trees that jointly maximize the overall accuracy of recidivism prediction (RP) and minimize equity disparity. As with many multi-objective problems, optimality of these two objectives in general cannot be achieved simultaneously. Consequently, we develop a multi-objective EA to derive a set of Pareto-optimal decision trees that provide non-dominated trade-offs between the two objectives we seek to optimize.

In this context, a decision tree D_1 is said to *dominate* another decision tree D_2 if D_1 is superior to D_2 in terms of either RP accuracy or equity and D_1 is superior to or equal to (i.e., not worse than) D_2 in the other objective. More generally, in an optimization problem with n criteria, a candidate solution s_1 dominates another candidate solution s_2 if s_1 is superior to s_2 in at least one criterion, and s_1 is not worse than s_2 in terms of all of the other $(n-1)$ criteria. This concept of domination in multidimensional optimization is sometimes referred to as *Pareto dominance*.

The non-dominated set of decision trees derived by our EA for a given training dataset T is also referred to as the Pareto front produced by the EA, as described in Section 3.2. The Pareto front produced by the EA for a given training dataset is in general distinct from the *true Pareto front*, which is the set of all non-dominated decision trees from among all possible decision trees, as evaluated by T. This distinction arises because EAs in general form a heuristic class of optimization methods. EAs are suitable for complex

optimization problems in which the derivation of true Pareto fronts is prohibitive in terms of computational complexity. The joint RP and equity optimization problem that we address in this section is such a problem for which computationally tractable methods are not known for deriving true Pareto fronts.

All EAs share several standard steps: initialization, evaluation, selection, crossover, mutation, and termination (e.g., see [25]). The general procedure of EAs mimics the natural selection process of living species and is outlined in Algorithm 1. We use the previously-developed EADTC (Evolutionary Algorithm Decision Tree for Crime prediction) [11] approach as a starting point for the multi-objective EA that we develop in this work. EADTC is a single-objective EA that has been developed to optimize accuracy. In this section, we build on the design of EADTC, and we make significant adaptations to address the joint optimization of RP and equity, which is the central objective of this section. In the remainder of this section, we refer to the multi-objective EA developed in the section as EEQRP or E2QRP, which stands for Evolutionary algorithm for multi-objective EQuity and RP optimization.

Algorithm 1 The general procedure of EAs.

```
Initialize population with randomly-generated candidate solutions
Evaluate the fitness of each individual in the population
while termination condition not met do
    Select parents from the population
    Perform crossover and mutation to generate new offspring
    Evaluate the fitness of the new offspring
    Select individuals for the next generation
end while
```

E2QRP differs significantly form EADTC in the initialization and selection phases. Another major difference between E2QRP and EADTC is that the output of E2QRP is

a Pareto front, whereas the output of EADTC is a single solution. In the remainder of this section, we describe key aspects of E2QRP, with emphasis on those aspects that are different from EADTC.

For initialization, while EADTC selects features and thresholds randomly, E2QRP applies the Gini index to initialize the population with higher quality solutions than a random generation approach would typically produce.

The Gini index, often used in the context of decision trees and machine learning, is a measure of impurity or purity used when creating decision trees in algorithms such as the CART (Classification and Regression Trees) algorithm [5]. The Gini index quantifies the variability or diversity of a distribution. By optimizing for the Gini index in the E2QRP algorithm, the decision-making process is guided towards creating splits that are more informative and discriminative. This leads to decision trees that are likely to perform better in classifying instances.

For selection, E2QRP goes beyond EADTC by incorporating methods to accommodate the intricacies of multi-objective optimization. In particular, E2QRP incorporates two distinct fitness assignment strategies: one employs a weighted sum that merges the dual objectives of RP and equity into a scalar value for comparative simplicity, while the other applies the concept of Pareto dominance to assign fitness, respecting the inherent multi-dimensional nature of our targeted problem, and enabling the derivation of a Pareto front as the output of the optimization process.

E2QRP incorporates an optimization approach that effectively balances the discovery of diverse solutions with the need for convergence towards the true Pareto front. Among the various approaches available for integrating Pareto dominance concepts with EAs (e.g.,

see [62] for a survey on such approaches), E2QRP applies the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [64] because of its demonstrated success in many different application areas.

The comparison operator in NSGA-II is called the crowded-comparison operator. The crowded-comparison operator combines two different ways of assessing candidate solutions that are relevant in multi-objective optimization: the non-domination rank and the crowding distance. Intuitively, the non-domination rank is used to assess the overall quality of a candidate solution σ in a multi-objective context, and the crowding distance is used to assess how σ contributes to the solution diversity in the design evaluation space. Here, by diversity, we mean the degree to which candidate solutions are spread out across the space of multiple objectives as opposed to being clustered together. A more diverse set of candidate solutions in general provides a broader range of trade-offs, and a more representative sampling of the true Pareto front.

In the general multi-objective optimization approach provided by NSGA-II that E2QRP applies, candidate solutions in a population are partitioned into subsets $F_1, F_2, \ldots F_n$ such that for $i = 1, 2, \ldots, (n-1)$, the individuals in each F_i dominate the individuals in all $F_{i+1}, F_{i+2}, \ldots, F_n$. We refer to each F_i as an *ordered front* within the enclosing population. The number of ordered fronts n may vary from one population (EA iteration) to another. In general, as evolution progress, the number of ordered fronts decreases and the first ordered front F_1 expands. The *non-domination rank* of an individual within a given population is given simply by the index of the ordered front in which the individual exists.

Fig. 3.1 provides a simple illustration of ordered fronts within a population. Here,

$n = 2$, and F_1 and F_2 correspond to the yellow- and green-colored subsets of individuals, assuming that we are trying to *minimize* each of the two objective criteria.

The crowding distance is the sum of the normalized Euclidean distance of two closest solutions on both side of this solution (i.e. the slightly larger one and the slightly smaller one) along each of the objective, indicating the neighborhood crowding of a solution. Combining them, the crowded-comparison operator works in this way: if two solutions have different non-dominated ranks, it favors the solution with the lower non-dominated rank. Otherwise, if the solutions exist on the same front, it is advantageous for the solutions to be located in sparse regions, which helps to find a diverse set of solutions.

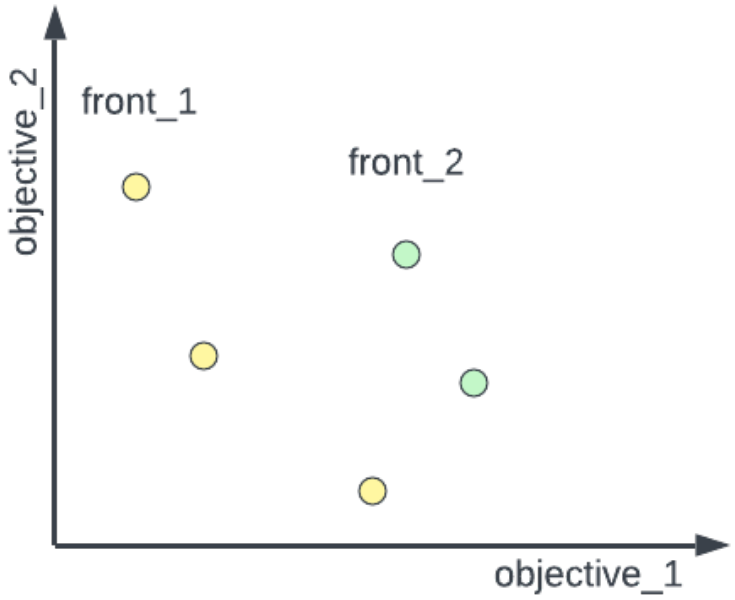


Figure 3.1: An illustration of ordered fronts in the population of a two-objective EA.

The crossover and mutation operators in E2QRP are similar to those employed in EADTC. For details, we refer the reader to [11].

In Step 7 of Algorithm 1, the population P_{t+1} for the next EA iteration is derived by selecting from the combination of the previous population P_t and the associated offspring

Q_t using the crowded-comparison operator described above. The crowded comparison operator, when used in this way, guarantees that an elitist strategy will be implemented. This means that the best solutions (i.e., all non-dominated solutions) are automatically carried over from P_t to P_{t+1}.

3.5 Experiments

In this section, we present experimental results that demonstrate the capabilities of the proposed E2QRP algorithm. E2QRP is implemented with Python 3.9 and DEAP (Distributed Evolutionary Algorithms in Python) version 1.0 [65]. DEAP is a software framework that facilitates rapid prototyping of novel EAs and EA methods.

3.5.1 Data Preparation

In our study, we utilized a real life dataset relevant to RP to conduct the experiments. The dataset, representing individuals from Pennsylvania state prisons, focuses on those released into parole supervision between 2006 and 2008. We refer to this dataset as the PA Dataset. We selected a relevant set of features from this dataset, including demographic information such as age and race. We also selected features related to crime history, such as the time previously served (if applicable), and whether or not a given offense is the subject's first offense.

3.5.2 Experimental Setup

Hyperparameter settings for E2QRP that we used in our experiments are summarized as follows.

- `n_trees = 200` (number of decision trees in each generation).
- `max_depth = 5` (maximum depth of each decision tree).
- `max_iter = 1000` (total number of iterations to run the evolutionary algorithm).
- `crossover_prob = 0.5` (probability with which two individuals are crossed over).
- `mutation_prob = 0.2` (probability of mutating an individual).

Using the above-listed hyperparameters for E2QRP, we conducted two sets of experiments: the first set involved single-objective optimization of RP accuracy, and the second set involved multi-objective optimization of RP accuracy and fairness. In each set of experiments, we compared the performance of E2QRP with previously-developed prediction models that have been applied to related tasks. Specifically, we compared E2QRP with the following relevant prediction models from the literature.

- Logistic Regression (LR): a statistical model used for binary classification tasks [66].
- Classification and Regression Trees (CART): a decision tree learning technique introduced by Breiman et al. [67].
- Random Forests (RF): an ensemble learning method for classification and regression that operates by constructing multiple decision trees [68].

- Gradient Boosting Decision Trees (GBDT): an ensemble machine learning technique for regression and classification problems, which provides for optimization of arbitrary differentiable loss functions [69].

We discuss these experiments and their results in more detail in Section 3.5.3 and Section 3.5.4.

3.5.3 Optimization of Accuracy

In this set of experiments, we examine the performance of E2QRP on accuracy as a single objective. We do this by post-processing the result of E2QRP to extract the solution with highest accuracy from the optimized Pareto front that is output by E2QRP. We refer to this post-processed, single-objective (SO)) version of E2QRP as SO-E2QRP.

We experiment with two variants of SO-E2QRP, which correspond to applying E2QRP for joint optimization of accuracy and Δ_α, and applying E2QRP for joint optimization of accuracy and Δ_β. The metrics Δ_α and Δ_β are fairness metrics whose values reside in the interval $[0, 1]$ and for which lower values correspond to better fairness. More details about Δ_α and Δ_β are discussed in Section 3.5.4. We refer to the two variants of SO-E2QRP as SOER1 and SOER2, respectively, where SOER stands for Single-Objective Equity / Recidivism prediction optimization.

One question we would like to answer in these experiments is whether E2QRP is useful even in the single-objective case or whether another algorithm should be used when accuracy is the only concern. The utility of multi-objective methods for single-objective problems has been demonstrated in various contexts (e.g., see [70]). Such prior work gives

further motivation to assess the effectiveness of SO-E2QRP compared to state-of-the-art single-objective methods.

Table 3.1: Comparison between SO-E2QRP and several previously developed prediction models in terms of prediction accuracy.

	LR	CART	RF	GBDT	SOER1	SOER2
accuracy	0.684	0.591	0.665	**0.688**	0.682	0.686

From table 3.1, it is clear that SOER1 and SOER2 are competitive with other reference models. At the same time, SOER1 and SOER2 provide significantly higher interpretability since they produce predictions models in the form of single decision trees.

3.5.4 Joint Optimization of Accuracy and Fairness

To assess equity (fairness) quantitatively, we adopt the fairness metrics introduced in [11]. These metrics, denoted Δ_α and Δ_β, measure equity as it relates to protected groups in a population about which inferences are being made by a prediction model. We refer to these metrics as the conditional procedure error disparity (CPED), and conditional use error disparity (CUED), respectively. For example, people of a specific race or age group or from a specific geographic region may be used to define a protected group.

The metrics Δ_α and Δ_β correspond to two forms of equity assessment that are referred to in fairness literature as conditional procedure accuracy equality (CPAE) and conditional use accuracy equality (CUAE), respectively. For background on CPAE and CUAE, we refer the reader to [49], and for detailed formulations of the Δ_α and Δ_β metrics, we refer the reader to [11].

The Δ_α and Δ_β metrics are each defined in terms of a pair of protected groups. The

metrics give an indication of the degree of disparity (inequity) between the two groups P_1 and P_2 in the pair. Intuitively, the disparity measured is that between prediction-accuracy levels achieved for P_1 and P_2. In our experiments, we select P_1 and P_2 to correspond to (1) black members of the population, and (2) members of the population who are not black. Thus, unfavorable values of Δ_α and Δ_β indicate that prediction accuracy is on average significantly worse for black compared to non-black members or vice versa.

The metrics Δ_α and Δ_β are formulated such that their values always lie within the interval $[0, 1]$ and smaller values imply better fairness. By systematically measuring accuracy disparities associated with a prediction model between different protected groups, the Δ_α and Δ_β metrics can help to address and mitigate unfairness in algorithmic decisions, particularly in contexts where equitable treatment across different demographic groups is paramount. Our contribution in this section represents a specific algorithm that incorporates these metrics to incorporate fairness into the prediction process for a relevant application. For more details on the Δ_α and Δ_β metrics, we refer the reader to [11].
The formulations of these metrics ensure that both Δ_α and Δ_β are always constrained within the interval $[0, 1]$, and smaller values implies better fairness.

We first calculate the metrics Δ_α and Δ_β based on the performance of SO-E2QRP on the dataset described in Section 3.5.1. We also measure the performance in terms of these metrics for the prediction models from the literature that we have experimented with. The results are reported in Table 3.2. From this table, we can see that SO-E2QRP (as represented by SOER1 and SOER2) achieves superior performance in terms of both Δ_α and Δ_β compared to all of the other methods. In Table 3.2, we report only the Δ_α value for SOER1 and only the Δ_β value for SOER2. This is because SOER1 and SOER2 are

designed specifically for Δ_α and Δ_β, respectively, as the fairness metric.

Table 3.2: Comparison between SO-E2QRP and other prediction algorithms in terms of Δ_α and Δ_β.

	LR	CART	RF	GBDT	SOER1	SOER2
Δ_α	0.672	0.0940	0.342	0.448	**0.0571**	—
Δ_β	0.172	0.0879	0.145	0.141	—	**0.0398**

Next, we experiment with the capability of E2QRP to jointly optimize accuracy and fairness, and to expose Pareto-optimal trade-offs between these concerns. Since Δ_α and Δ_β correspond to two distinct fairness metrics, which capture different aspects of fairness, we perform trade-off experimentation in three different forms:

- 2-dimensional optimization involving accuracy and Δ_α;
- 2-dimensional optimization involving accuracy and Δ_β; and
- 3-dimensional optimization involving accuracy, Δ_β and Δ_β.

These forms of different experiments highlight the flexibility with which E2QRP can be tailored based on the different aspect or aspects of fairness that are of interest in a particular application.

The Pareto fronts resulting from these three forms of experiments are illustrated in Fig. 3.2, Fig. 3.3, and Fig. 3.4, respectively. The results clearly demonstrate the conflicting nature of accuracy- and fairness-related metrics in this application — improvement in one concern comes at the expense of degradation in the other. E2QRP provides a novel framework that allows prediction system designers to expose the underlying space of trade-offs so that they can select the operating point that is jointly optimized in terms of

the relevant concerns, and that trades off the concerns in a way that is found to be most acceptable for the given application.

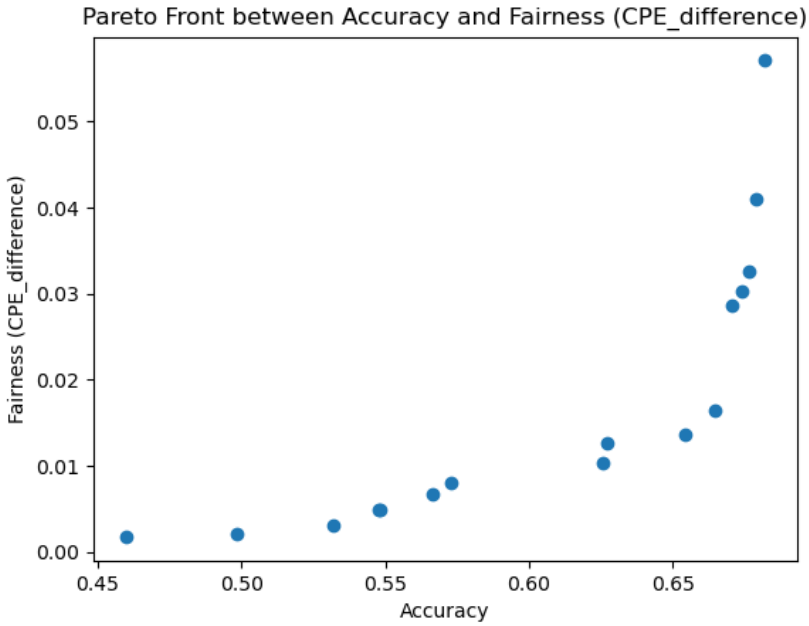


Figure 3.2: Pareto-front between accuracy and Δ_α.

3.5.5 Pareto Front Evolution

In this section we illustrate how the Pareto front derived by E2QRP evolves as generations of the underlying EA progress. We recorded the Pareto fronts obtained during the evolutionary process at the following milestones:

- after 1/5 of the total iterations, as shown in Fig. 3.5 and Fig. 3.8
- after 1/2 of the total iterations, as shown in Fig. 3.6 and Fig. 3.9
- after after all iterations are completed (the final result), as shown in Fig. 3.7 and Fig. 3.10.

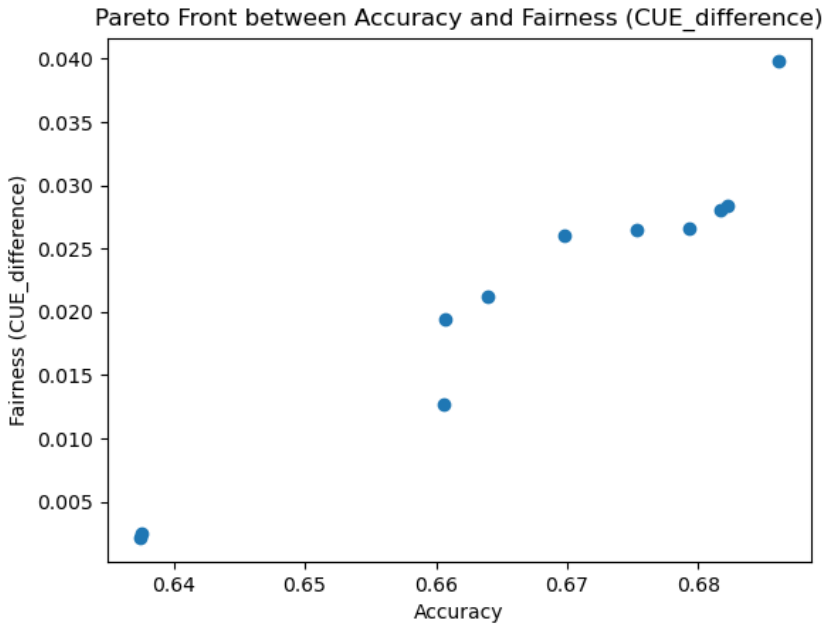


Figure 3.3: Pareto-front between accuracy and Δ_β.

From Fig. 3.5, it is clear that in the joint optimization of accuracy and Δ_α, the number of Pareto-optimal solutions found generally increases with the number of EA iterations. As iterations of the EA progress, new solutions are found to fill in empty regions near the Pareto front, and some solutions are removed because they become dominated by newly-generated solutions.

However, as seen in Fig. 3.8, the same trend is not evident in the joint optimization of accuracy and Δ_β as the Pareto-optimal solutions found in all three stages are the same. Indeed, we have verified that the Pareto fronts in the different stages illustrated all numerically equivalent. We anticipate that the Pareto fronts remain unchanged because the Δ_β fairness measure is relatively compatible with accuracy for this dataset, and the initialization process for E2QRP performs relatively well for this combination of metrics (optimization dimensions) and dataset.

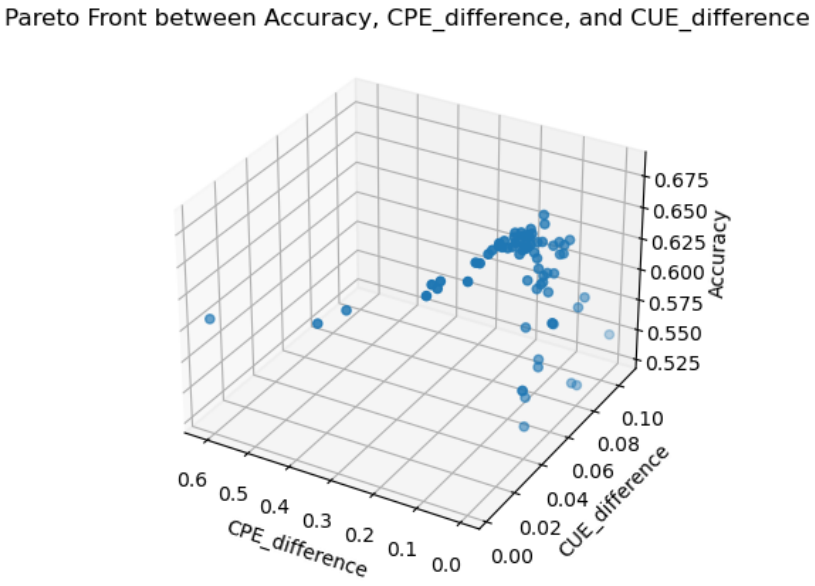


Figure 3.4: Pareto-front among accuracy, Δ_α, and Δ_β.

3.6 Conclusion

In this research, we have demonstrated the potential of evolutionary algorithms (EAs) in developing decision trees that balance the conflicting objectives of accuracy and fairness in criminal justice predictions. Our approach, called E2QRP (Evolutionary algorithm for multi-objective EQuity and RP optimization) enhances the natural interpretability of decision trees and acknowledges the ethical concerns associated with biased data in predictive modeling. This is crucial in the field of criminal justice, where decision-making has profound social implications.

E2QRP is based on a novel integration of (1) interpretable, single-decision-tree models, (2) multi-objective optimization, (3) evolutionary algorithms, and (4) quantitative fairness assessment. Our approach of tree-based encoding, combined with careful design of initialization, selection, crossover, and mutation components of the EA process, is

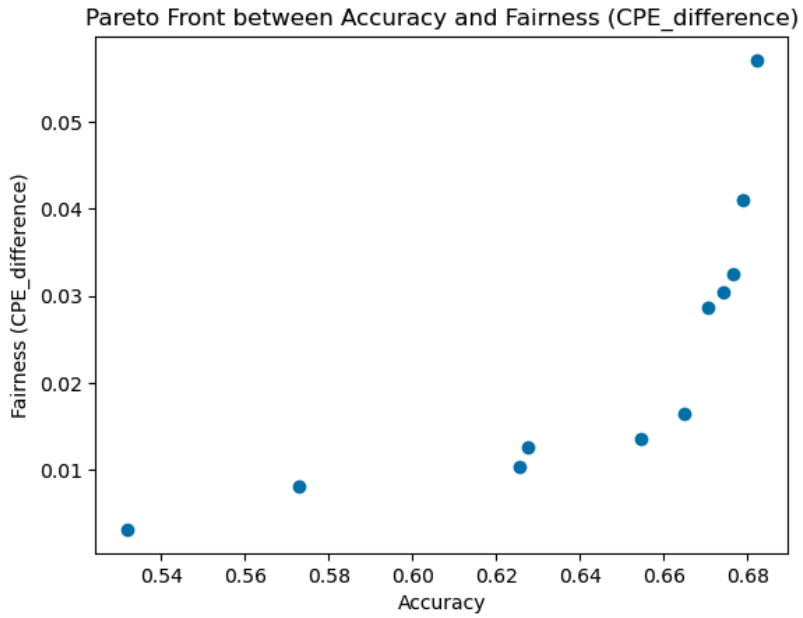


Figure 3.5: Results after 200 generations.

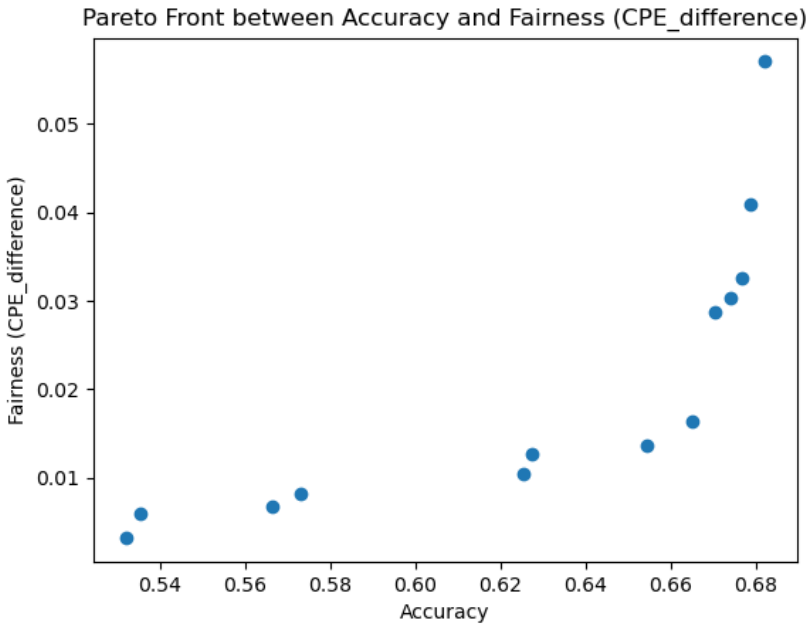


Figure 3.6: Results after 500 generations.

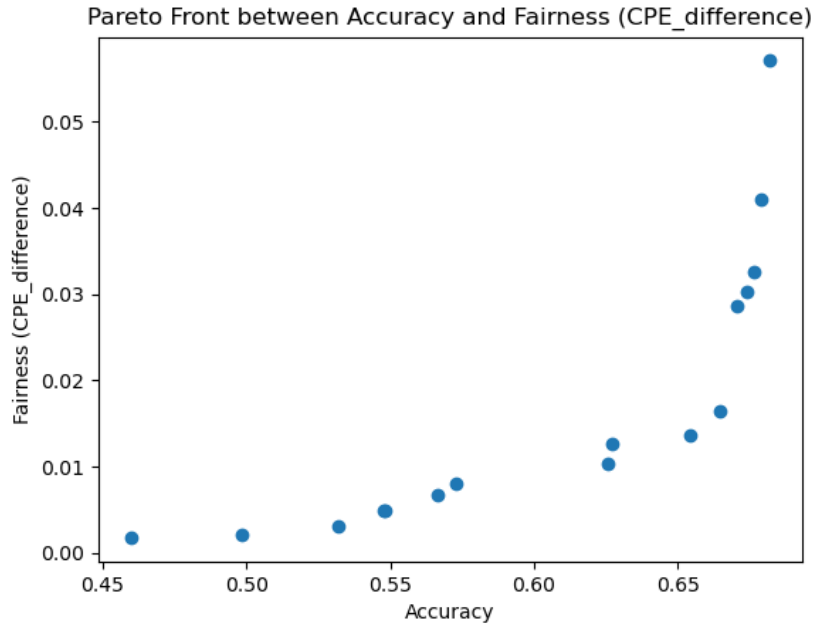


Figure 3.7: Results after 1,000 generations.

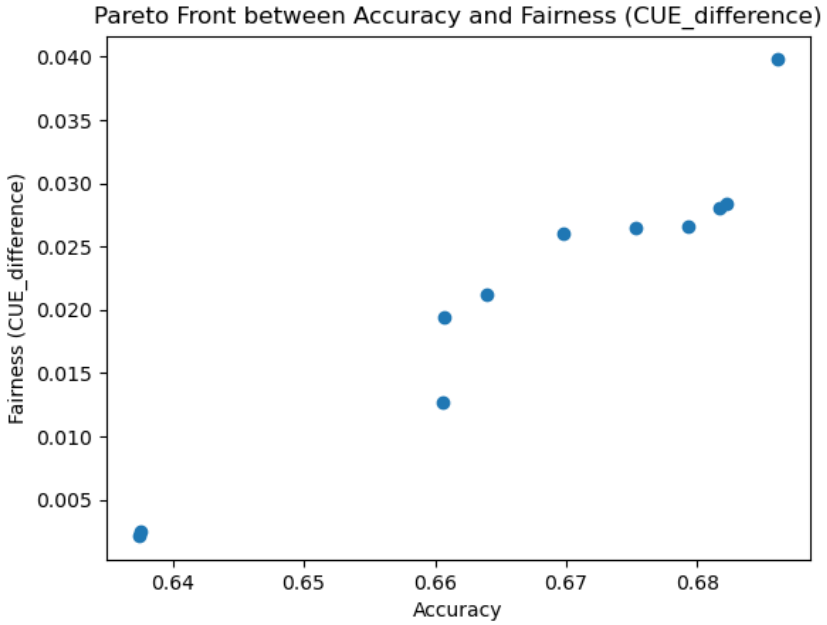


Figure 3.8: Results after 200 generations.

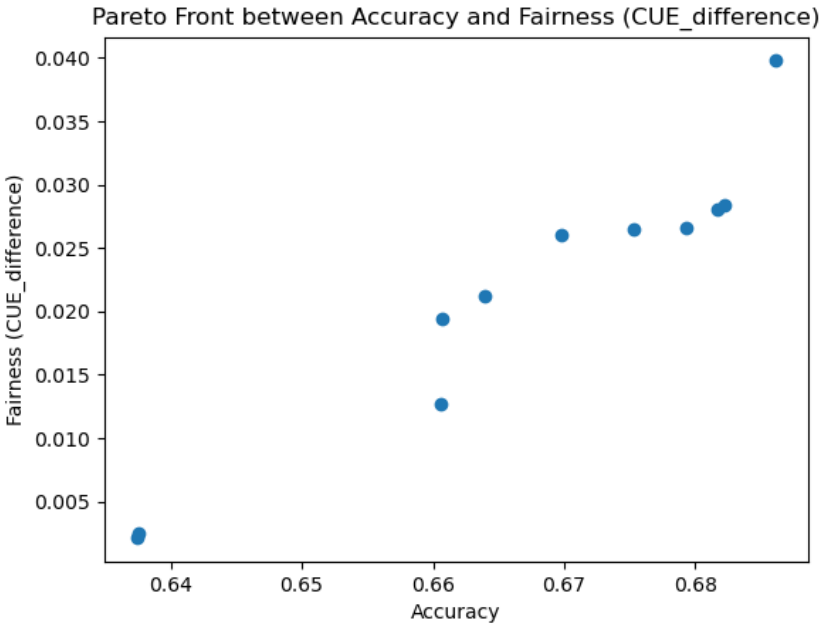


Figure 3.9: Results after 500 generations.

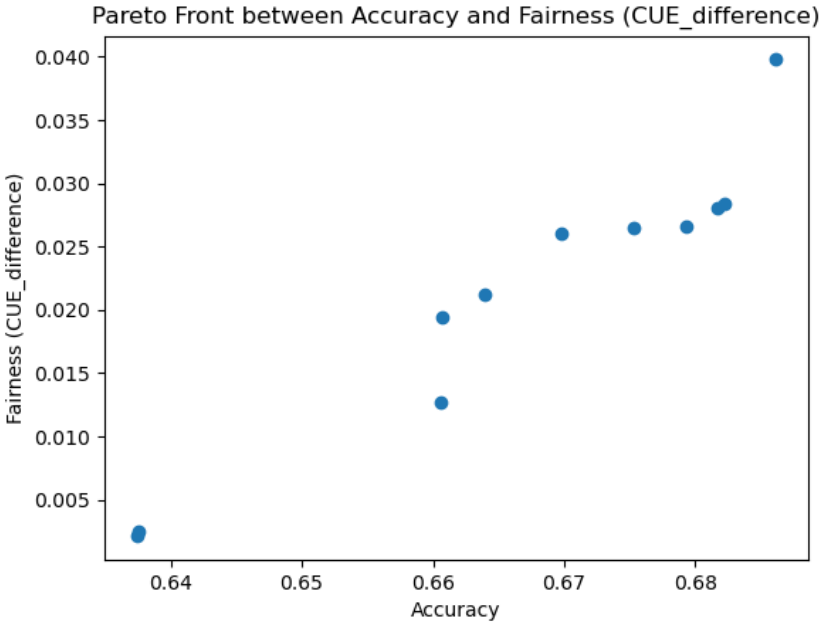


Figure 3.10: Results after 1,000 generations.

effective in cultivating a diverse set of potential solutions that are output in the form of a Pareto front, thereby allowing prediction-system designers to strategically select an operating point that best matches the desired balance between accuracy and fairness in the targeted application.

Useful directions for future research include exploring the integration of structural mutation into the EA process underlying E2QRP; currently, only semantic mutation is applied. Another useful direction is exploration into adapting E2QRP to other application areas in which fairness plays an important role alongside predictive accuracy.

Chapter 4

Towards Interpretable, Attention-Based Crime Forecasting

4.1 Chapter Overview

While the use of machine learning techniques in high stake fields, such as medical diagnosis and criminal justice, has been increasing in recent years, concerns have been raised regarding the lack of transparency and interpretability of the algorithms used. In this chapter, we propose the use of interpretable attention-based ConvLSTM models for crime forecasting application. This approach combines the power of ConvLSTM models in capturing spatio-temporal patterns with the interpretability of attention mechanisms, allowing for the identification of key geographic areas in the input data that contribute to the prediction. We demonstrate the effectiveness of this approach through experiments on real-world crime data, showing that our model demonstrates high accuracy in crime predictions while providing insightful visualization that enhances the interpretability of prediction results.

Material in this chapter was published in partial, preliminary form in [71].

4.2 Introduction

In recent years, machine learning has become a fundamental tool in many fields, from computer vision and natural language processing (NLP) to healthcare and finance. With

the ever-increasing amount of data being generated and collected, there is a growing need for algorithms that can efficiently learn from large data of different types, make accurate predictions, and aid human decisions. Recently, variants of attention-based transformer models have produced state-of-the-art results in NLP [3] and computer vision [3], due in part to their ability to capture long-range dependencies and spatial relationships. Despite the demonstrated results in these areas, the performance of transformers on time series forecasting has been lagging. This is an important limitation to address, given the wide range of applications of time series forecasting.

Spatio-temporal modeling, as a relatively new form of machine learning, has become increasingly popular in various application domains, such as traffic forecasting [72, 73, 74], climate prediction [75], and surveillance-video analysis [76]. These domains have in common the need to analyze data that varies both in space and time, and to study correlations involving variations in the data. However, although many similar real-life applications can be described with spatio-temporal modeling, the differences among areas make it hard to find a general inductive bias. Therefore, it usually takes significant domain expertise to exploit the full potential of a spatio-temporal modeling approach on a specific application area.

Crime forecasting is part of an emerging approach to law enforcement that leverages data, machine learning algorithms, and other types of predictive models to predict and prevent crime [77]. While crime forecasting has shown promising results in improving crime predictions and contributing to crime reduction [78], it has also raised concerns about potential biases and lack of transparency in the decision-making process [79]. One way to address these concerns is to use interpretable machine learning models that provide

insights into how the predictions are made [11]. However, interpreting the operation of effective crime forecasting models can be challenging due to their complex and dynamic nature.

In addition to predictive accuracy, as well as fairness, interpretability has been increasingly studied as an important aspect of machine learning models, particularly in high-stakes fields, such as healthcare [80], finance [81], and criminal justice [82], where the consequences of algorithm-supported decisions, especially incorrect decisions, can be severe. Unlike conventional machine learning models, such as logistic regression and single-decision-tree models, deep learning models are often seen as "black boxes" due to their complex nature, and it can be difficult to understand how they arrive at their decisions or predictions [83]. This lack of interpretability can lead to mistrust and weaken legitimacy. Thus, researchers have been exploring more self-interpretable models [84], as well as developing post-hoc techniques to explain black-box models [85].

In this chapter, we propose a novel attention-based ConvLSTM model for crime forecasting. ConvLSTM, which stands for convolutional long short-term memory, is a type of recurrent neural network model that has been designed for spatio-temporal modeling [75], and provides an important foundation for the contributions in this chapter; ConvLSTM concepts are discussed further in Section 4.3.

We evaluate the performance of our proposed new ConvLSTM model using the Moving MNIST dataset [86] and a publicly available dataset of crime incidents in Baltimore City, and we compare the performance with other state-of-the-art models. We visualize the attention maps of our proposed model, which provides important insight into the operation of the model. In addition, we explore the fairness of our proposed model in terms of

minimizing bias and promoting equitable outcomes for all populations. The results of our study have important implications for the development of spatio-temporal models that are interpretable. By addressing the challenges of interpretability and fairness, we aim to help pave the way for the adoption of our proposed attention-based ConvLSTM methods in real-world public safety applications and other domains where transparency and accountability are critical.

The remainder of this chapter is organized as follows. In Section 4.3, we survey literature that is related to machine learning for public safety and more specifically, to the ConvLSTM model. In Section 4.4, we propose a general method to apply the attention mechanism to ConvLSTM. In Section 4.5, we conduct a series of experiments to analyze the performance and interpretability of our proposed model. In Section 4.6, we summarize our the findings in our chapter, and present directions for future work.

4.3 Background and Related Work

Spatio-Temporal Models. Spatio-temporal prediction has become a popular research area in machine learning, with applications in various domains such as weather forecasting, traffic prediction, and crime analysis. Even before the emergence of machine learning models in these applications, powerful conventional models have existed to predict phenomena that can exhibit patterns across time and space. For instance, the spatio-temporal autoregressive integrated moving average (STARIMA) [87] model combines a spatial autoregressive model with a temporal autoregressive integrated moving average model to capture spatio-temporal dependencies in the data.

Spatio-Temporal Deep Neural Networks. More recently, machine learning techniques, particularly deep learning models, such as convolutional neural network (CNN) models, have demonstrated great promise in learning complex spatio-temporal patterns and achieving high accuracy in prediction tasks, including image classification and facial recognition [88]. In particular, the convolutional block has shown great potential in learning spatial representations from input data. Perhaps the most natural thinking to apply CNNs to spatio-temporal prediction is to augment a 2D spatial CNN with a a temporal dimension [89]. A 3D-CNN designed in this way can leverage the ability to extract temporal information through its 3D convolution and 3D pooling layers. 3D-CNNs have been applied to various spatio-temporal prediction tasks such as human action recognition [90] and medical image analysis [91].

Although 3D-CNNs are an important step toward incorporating temporal dynamics, their ability to capture long-range dependencies is limited. One possible reason for this is that spatial correlation is typically stronger with adjacent entities, but time dependency may not be. For example, there are sharp semantic discontinuities between weekdays and weekends, and between the last day of a financial quarter and the first day of the next. To model connections between data from different time steps, a new type of deep learning model, called Recurrent Neural Networks (RNNs), was developed. RNNs are specifically designed for time series forecasting [92]. RNN models that are currently in use typically include gated state or gated memory modules to decide whether to pass information forward or "forget" it.

Long Short-Term Memory (LSTM) [93] is a type of RNN architecture that is designed to handle the issue of vanishing gradients in standard RNNs, and has been

shown to outperform ordinary RNNs in many areas [94]. The vanishing gradients problem arises when the gradients propagated through the network during backpropagation become very small, which can make it difficult for the network to learn long-term dependencies. The LSTM architecture was introduced to help address the vanishing gradients problem. ConvLSTM[75] is a variant of LSTM that has been developed to facilitate effective spatio-temporal modeling. ConvLSTM is a variant of LSTM that replaces the original LSTM blocks with convolutional layers. Unlike other models that treat the spatial and temporal dimensions independently, ConvLSTM models allow the model to capture complex patterns and dependencies by combining the strengths of LSTMs and CNNs.

Attention-based Models. Transformers have become one of the most popular types of machine learning models since they emerged [95]. Since their introduction, transformers have been adopted rapidly in major application areas of machine learning, including NLP [96] and computer vision [3]. However, transformers have exhibited limitations in forecasting time series due to temporal information loss [97]. To address this limitation, our work applies self-attention [95] on CNN and RNN structures to improve the feature extraction process. To aid in capturing global spatial information, Han concatenated self-attention modules after the convolutional feature maps of the Self-Attention Generative Adversarial Network (SAGAN) [98]. To the best of the authors' knowledge, existing work to apply attention blocks to ConvLSTMs have applied the blocks only to hidden states [99]. In contrast, our work also applies self-attention blocks after the outputs of convolutional layers in a ConvLSTM so that the model can better capture spatial correlation.

Interpretability. Attention-based models not only have the potential to significantly improve inference accuracy, but they are also useful in terms of interpretability. Unlike

CNNs, for which additional techniques, such as Gradient-weighted Class Activation Mapping (Grad-CAM) [85] are needed for interpretation, attention maps themselves can usually be treated as a kind of post-hoc interpretability. Samira developed two information flow methods, called attention roll-out and attention flow, to precisely quantify the attention weights while traversing different model layers [100]. Hila generalized interpretability assessment to computing the Layer-wise Relevance Propagation (LRP) score of visual transformers and showed that attention maps had better consistency with human intuition [8].

Despite the success of the models described above, important challenges remain in spatio-temporal prediction in ensuring the fairness and equity of applied models, and in integrating them effectively with domain knowledge for important new application areas for the models. Our focus this chapter is on the application area of crime forecasting, an area where fairness and equity are critical concerns, and where bias can have severe consequences on human lives. Public safety researchers have proposed several methods to ensure fairness in spatio-temporal prediction models, such as counterfactual analysis and adversarial training. There are also corresponding techniques to help explain the mechanisms underlying black-box models. However, several of the techniques proposed or applied in the public safety literature are highly application-specific and not applicable to our context of crime forecasting. However, one technique, called attention rollout, is relevant to our work. We apply attention rollout to aid in the visualization of the proposed ConvLSTM model's attention during forecasting and to compare with raw attention maps. More details are discussed in Section 4.4.

In summary, machine learning techniques have shown great potential in spatio-

temporal prediction, with applications in various domains such as weather forecasting, traffic prediction, and crime analysis. While deep learning models such as deep CNNs have achieved state-of-the-art results in major application areas, there is still a need for models that are both accurate and interpretable, particularly in domains such as public safety, where transparency and accountability are critical. Addressing the challenges of interpretability in spatio-temporal prediction models is critical for their adoption in such domains. This chapter develops new methods, based on concepts of ConvLSTMs, to address these challenges.

4.4 Proposed Learning Model

In this section, we propose a general form of attention-based ConvLSTM model that is well-suited to the problem of crime forecasting and related public safety applications. We refer to this type of model as ABC-LSTM (Attention-Based Convolutional LSTM).

Although self-attention models perform well in many areas, the capacity of transformers to capture temporal correlations depends heavily on the positional encoding, which leads to some controversy in the use of transformers for time series forecasting applications. A distinguishing characteristic of our proposed ABC-LSTM approach is that we do not *replace* CNN blocks or any LSTM modules with attention blocks. Instead, we *concatenate* self-attention layers at the outputs of selected blocks to aid in the feature extraction process.

4.4.1 A General Framework of Attention-based ConvLSTM

We first formulate the ABC-LSTM approach, which starts with the original ConvLSTM structure, and modifies the structure by concatenating two modules at intermediate outputs within the model. For further details on fundamental LSTM concepts on which our ABC-LSTM formulation builds, we refer the reader to [101].

We present the ABC-LSTM approach in the context of single-channel, two-dimensional inputs, such as grayscale images or data laid out along a two-dimensional spatial grid. However, the model can readily be extended to handle multi-channel data, single-dimensional data, and multi-dimensional data with more than two dimensions. Details of these extensions are beyond the scope of this chapter.

The ABC-LSTM approach can be specified formally as follows.

$$i_m = \sigma(SA_x(W_{xi} \star \mathcal{X}_m) + SA_h(W_{hi} \star \mathcal{H}_{m-1}) + W_{ci} \circ \mathcal{C}_{m-1} + b_i) \tag{4.1}$$

$$f_m = \sigma(SA_x(W_{xf} \star \mathcal{X}_m) + SA_h(W_{hf} \star \mathcal{H}_{m-1} + W_{cf} \circ \mathcal{C}_{m-1} + b_f) \tag{4.2}$$

$$\mathcal{C}_m = f_m \circ \mathcal{C}_{m-1} + i_m \circ \tanh(SA_x(W_{xc} \star \mathcal{X}_m) + SA_x(W_{hc} \star \mathcal{H}_{m-1}) + b_c) \tag{4.3}$$

$$o_m = \sigma(SA_x(W_{xo} \star \mathcal{X}_m) + SA_h(W_{ho} \star \mathcal{H}_{m-1}) + W_{co} \circ \mathcal{C}_m + b_o) \tag{4.4}$$

$$\mathcal{H}_m = o_m \circ \tanh(\mathcal{C}_m) \tag{4.5}$$

Here, $\mathcal{X}_m$ represent the inputs to the model, $\mathcal{H}_m$ denote hidden states, and $\mathcal{C}_m$ denote cell outputs. H_0 and C_0 are boundary values that we set to zero-vectors. The equations above for i_m, f_m, C_m, o_m, H_m hold for $m = 1, 2, \ldots t$. The symbol t represents the batch size, which is the window-length for data across which the ABC-LSTM model makes predictions. The symbols $\star$ and $\circ$ represent the convolution operator and Hadamard product, respectively. The symbols i_m and f_m represent the LSTM input gate and forget gate, which control information accumulation and discarding of information, respectively. The symbol o_m represents the output gate, which determines the propagation of the cell output c_m and hidden state h_m, and then controls the hidden states to be passed to next time step. A symbol of the form SA_x represents attention module x applied to convolutional layers and hidden states. The symbols σ and tanh represents the sigmoid and tanh functions. Here Fig. 4.1 shows one entire ABC-ConvLSTM block, Each symbol of the form W_{ab} represents a matrix of weights associated with layer input a and component type b. Here component types $b = f, c, i, o$ correspond to the forget gate, cell output, input gate, and output gate, respectively. For example, W_{xf} corresponds to the weights associated with layer input X and the forget gate.

Self-attention. Here we use a conventional self-attention module, which calculates query components $Q_H = Z_q H_m$ and $Q_X = Z_q X_m$ for $m = 1, 2, \ldots, t$, as well as key components $K_H = Z_k H_m$ and $K_X = Z_k X_m$, and value components $V_H = Z_v H_m$ and $V_X = Z_v X_m$. Here, for each m, $H_m, X_m \in \mathbb{R}^2$, where $\mathbb{R}$ denotes the set of real numbers. Also, $Z_q, Z_k \in \mathbb{R}$, and $Z_v \in \mathbb{R}$ are in the form of learned 1×1 convolutions.

We can selectively apply the self-attention block to the cell output and hidden states. As shown in Fig. 4.2, Z_q, Z_k and Z_v are applied as 1×1 convolutions to the feature maps

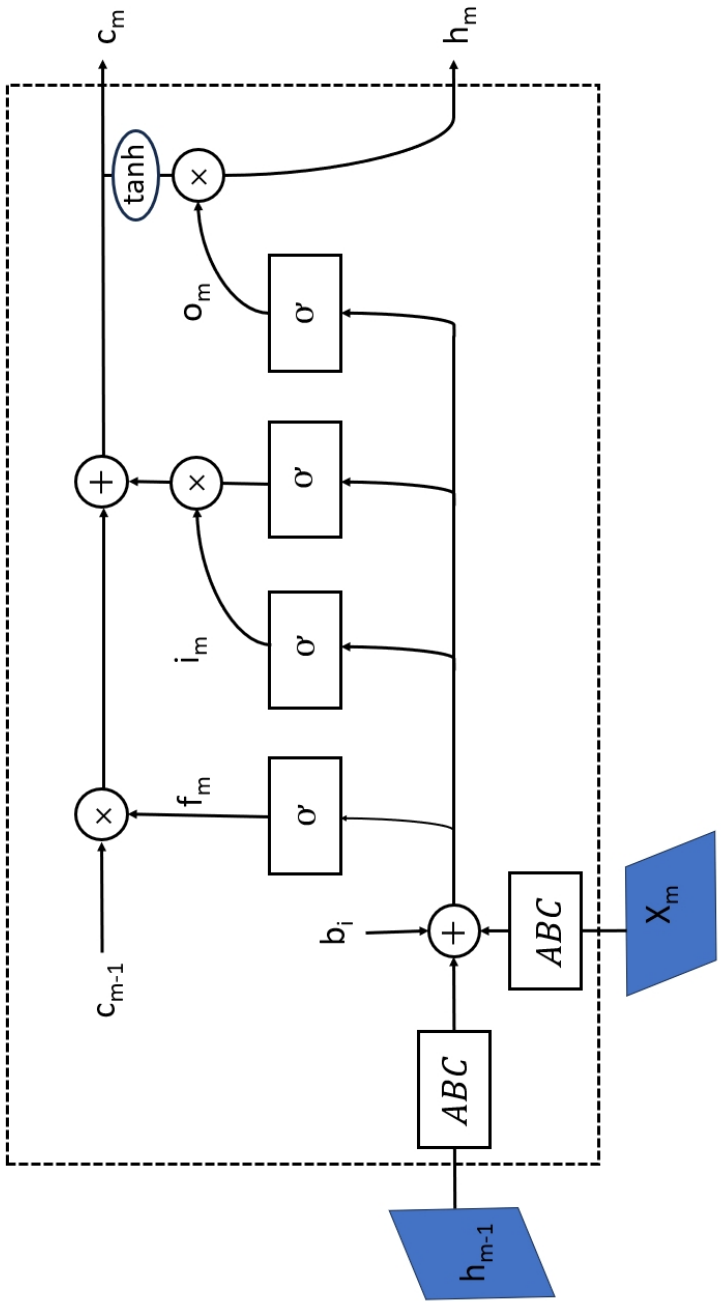


Figure 4.1: Illustration of a single ABC-ConvLSTM block.

of hidden states and inputs. After calculating self-attention, a residual connection is added to avoid a vanishing gradient.

4.4.2 Post-hoc Interpretability of Attention Models

As motivated in Section 2.2, a critical problem that we need to address when developing machine learning models for crime forecasting and other public safety problems is interpretability. Attention-based models are useful to promote interpretability since we can map information from feature maps back to the input through a simple process of propagation.

The concept of attention flow is used to help system designers visualize and understand the propagation of information in a machine learning model. Attention flow uses a graph model, called the *information flow graph*. A flow maximization algorithm, such as the Ford-Fulkerson algorithm, can be applied to the information flow graph to derive the maximum level of information propagation from a source vertex to a source vertex in the information flow graph.

In our work, the information flow graph can be expressed as an ordered pair $G = (V, E)$, where V is a set of vertices representing individual neurons in the ABC-LSTM model, and E is the set of connections between neurons. For more background on attention flow concepts and the information flow graph, we refer the reader to [100]. In Section 4.5, we present results from using attention flow concepts to visualize operation of the ABC-LSTM model.

We plan to release our implementation of ABC-LSTM for crime forecasting in open source form along with the publication of this chapter if it is accepted.

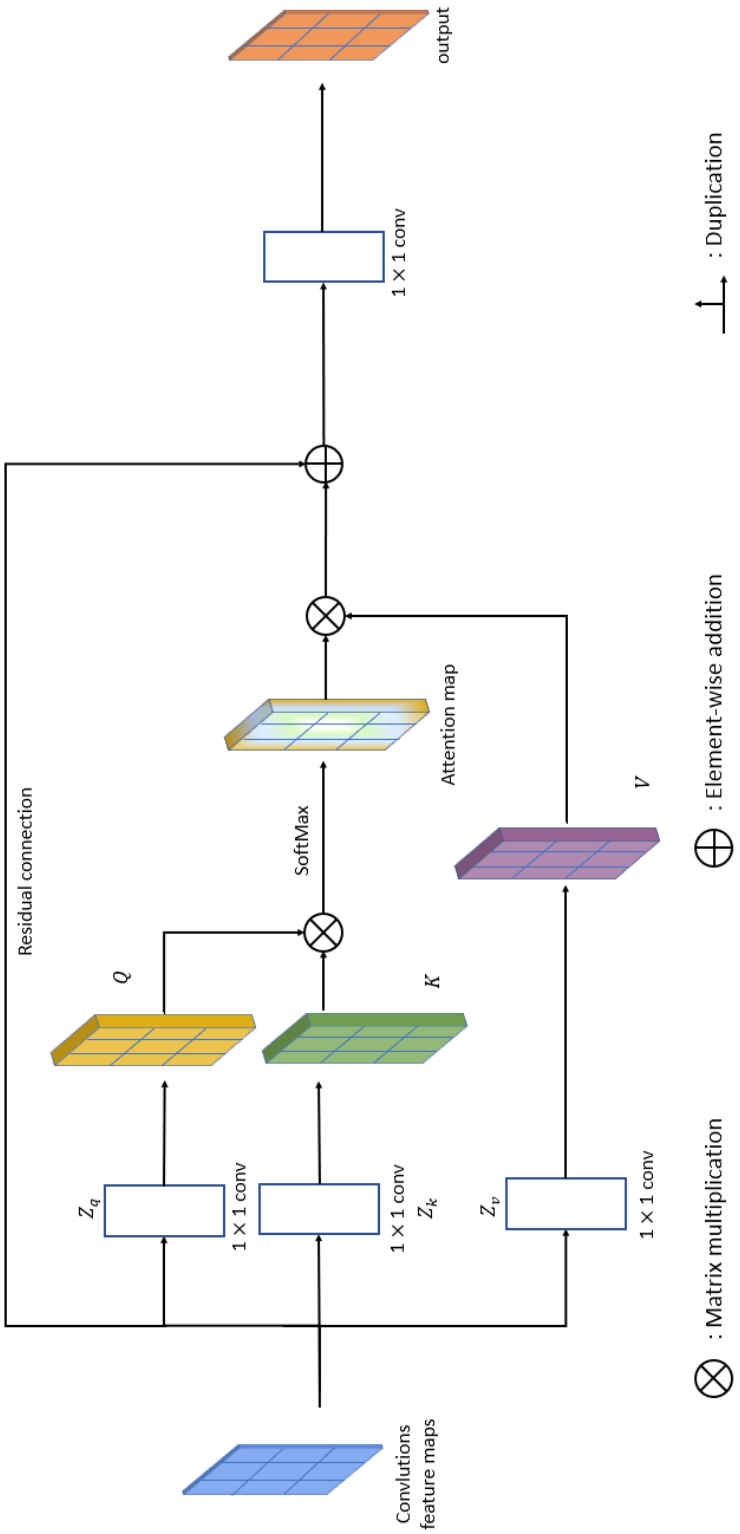


Figure 4.2: Illustration of a self-attention module in ABC-LSTM.

4.5 Experiments

In this section, we evaluate our proposed ABC-LSTM model on two relevant datasets and compare the performance with baselines taken from prior work, including the work reported in STARIMA[87] and ConvLSTM[75]. The datasets that we use are the Moving MNIST Dataset and a crime incident dataset from the Baltimore Police Department. For both datasets, we use the same model parameters: mini batch size $\rho_m = 4$, number of hidden channels $\rho_h = 32$, number of attention channels $\rho_h = 32$, number of epochs $\rho_e = 4000$, and learning rate $\rho_l = 0.001$. We trained the model using the ADAM optimizer based on the mean squared error. For each of the two datasets that we used, we used the following breakdown of the data into training, validation (hyperparameter optimization), and testing: 60%, 10%, and 30%.

4.5.1 Moving MNIST Dataset

Moving MNIST is a popular dataset in the area of spatio-temporal forecasting. The Moving MNIST dataset consists of a sequence of 20 frames, each of which contains two handwritten digits of size 64×64 (gray scale) that move across the image plane. The digits move in different directions and at varying speeds, making the dataset challenging for spatio-temporal modeling. Fig. 4.3 shows an example of our proposed model applied to the problem of predicting the next 5 frames given a sequence of 5 frames in the Moving MNIST dataset.

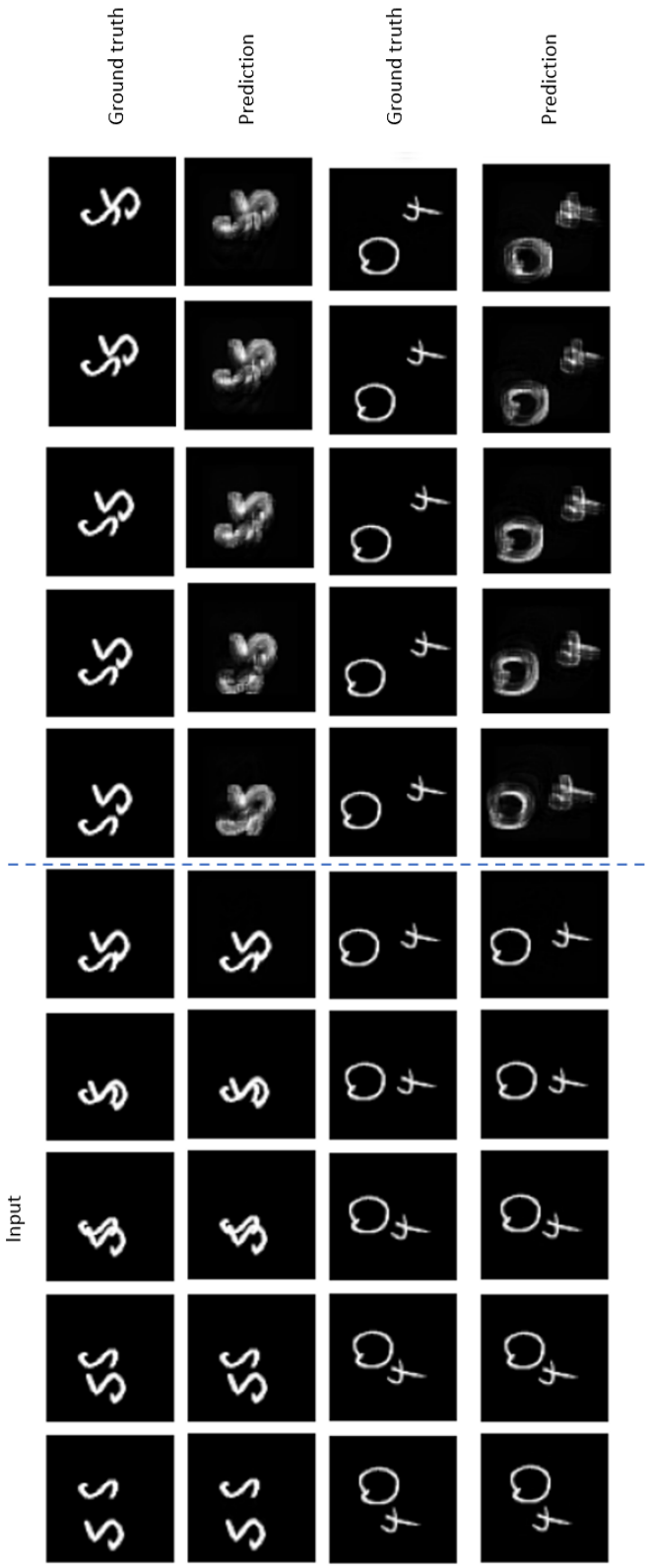


Figure 4.3: Example of prediction and ground truth of Moving MNIST dataset.

4.5.2 Baltimore City Crime Dataset

We also performed experiments with the Baltimore City Crime Dataset [102]. The public crime data provided in this dataset is based on General Offense Reports extracted from the Baltimore Police Department's Records Management System. We used the Part I Crime data, which include the location and characteristics of major (Part I) violent crimes, such as homicide, robbery, and aggravated assault, as well as property crimes, such as burglary, vehicle theft, and larceny. The dataset is updated weekly; the specific time period that we used in our experiments is from December 2010 to February 2023. After removing crime records outside of the city limits based on the longitude and latitude information, we used the dataset containing a total of $539{,}781$ crime incidents over $4{,}550$ days.

We segment the original map in which crimes are stored to a 64×64 grid and map the crime events into the grid based on longitude and latitude data. For fair comparison with conventional models, such as STARIMA, we use the crime records by themselves — that is, without other contextual factors, such as census-based neighborhoods characteristics and other risk and protective factors of crime. However, our model is readily capable of taking arbitrary contextual factors into consideration. Further discussion on the use of such contextual factors is beyond the scope of this chapter.

A potential application of the type of prediction performed in these experiments is for police to identify possible hot spots for crime in advance, thereby aiding decisions into where to deploy resources. We examine the mean squared error (MSE) from predictions derived by alternative predictive models, and we also examine the prediction trade-off curve (PTC) produced by each model experimented with[103].

The PTC helps to assess the utility of a crime prediction model in guiding resource allocation decisions for public safety agencies. The PTC is a curve that is plotted with the ratio of police patrolling coverage — in terms of the the percentage of geographic area with the highest amount of forecasted crime — on the horizontal axis, and the ratio of crime committed in the forecasted area on the vertical axis. For example, if the the point $(0.05, 0.2)$ falls on the PTC, it means that the top 5% of geographic area within the city in terms of the amount of forecasted crime accounts for 20% of the actual crime reported in the city.

4.5.3 Results

Experimental results are presented in Table 4.1 and Fig. 4.4. From the results in Table 4.1, we see that the best performance is produced by the ABC-LSTM model with self-attention on both the inputs and hidden states — that is, by the ABC-LSTM-xh model. The superior performance by the ABC-LSTM-xh model is observed on both datasets. Compared to ConvLSTM, the ABC-LSTM-xh model provides better performance on the Moving MNIST dataset by 9.69% and the Baltimore City Crime Dataset by 3.06%. Compared to STARIMA [87], the improvement provided by ABC-LSTM-xh on the two datasets is more impressive — 21.11% and 6.75%, respectively.

For all five of the predictive models studied in Table 4.1, PTCs are illustrated in Fig. 4.4. By comparing the curves in the figure, we can see that the proposed ABC-LSTM-xh model is the most effective model for guiding resource allocation decisions towards crime hot spots.

Table 4.1: Mean squared error (MSE) of spatio-temporal models over both datasets considered in our experiments. Here, ABC-LSTM-z represents the ABC-LSTM method in conjunction with SA_z.

	ARIMA [87]	ConvLSTM	ABC-LSTM-x	ABC-LSTM-h	ABC-LSTM-xh
MNIST	77.26	67.49	63.36	62.25	**60.95**
Bal	1520.68	1462.74	1431.17	1425.93	**1418.05**

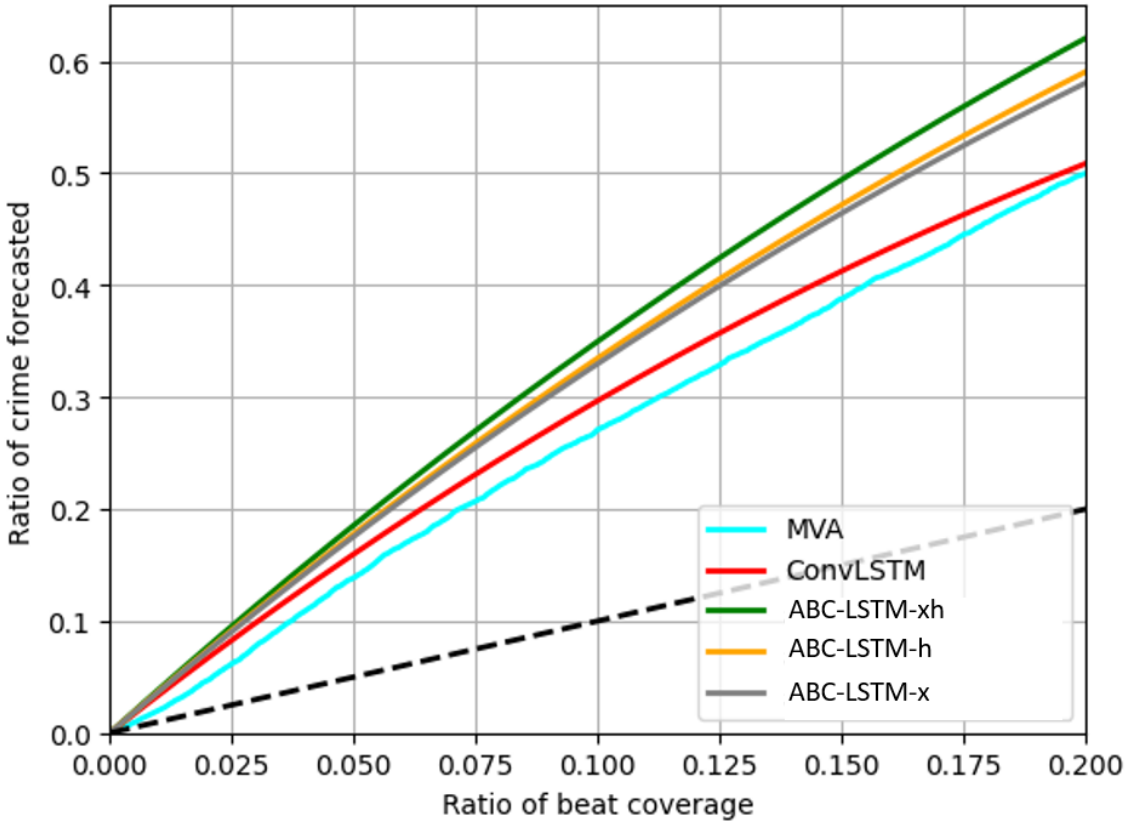


Figure 4.4: Prediction trade-off curves (PTCs) for spatio-temporal models applied to the Baltimore City Crime Dataset. The Part 1 component of the dataset is used with a temporal evaluation window of 13 years. The dotted line corresponds to a naive baseline that represents randomly sending police officers to different regions. MVA represents STARIMA.

4.5.4 Visualization of Feature Maps

Effective visualization of feature maps can significantly enhance the interpretability of a machine learning model, which can in turn help people to better understand the model and improve the model's legitimacy and trustworthiness. Both interpretability and trustworthiness are critical factors in the adoption and general acceptance of machine-learning-driven tools for public safety applications.

Our use of attention-flow-based visualization methods with the ABC-LSTM model helps to improve its interpretability. Fig. 4.5(b) illustrates the attention flow map of the ABC-LSTM model. The illustration is derived based on input provided from the Baltimore City Crime Dataset. The illustration highlights regions with high concentration of crime, and is useful to provide intuition about patterns that the model focuses on in its analysis.

Fig. 4.5 illustrates the ABC-LSTM model's attention over the input map, and quantifies the contribution of each grid cell to the final output crime prediction, where attention flow map has better concentration on the cells with higher crime density. The illustration shows that unlike moving-average-based models, our ABC-LSTM model will also spread a certain extent of the attention to neighboring cells when the "heat" (predicted crime intensity) changes gradually at cell borders, consistent with spatio-temporal dependencies in crime risk.

4.6 Discussion and Conclusion

In summary, our work makes the following contributions: Overall,

- We proposed a general method to apply attention block to ConvLSTM model and

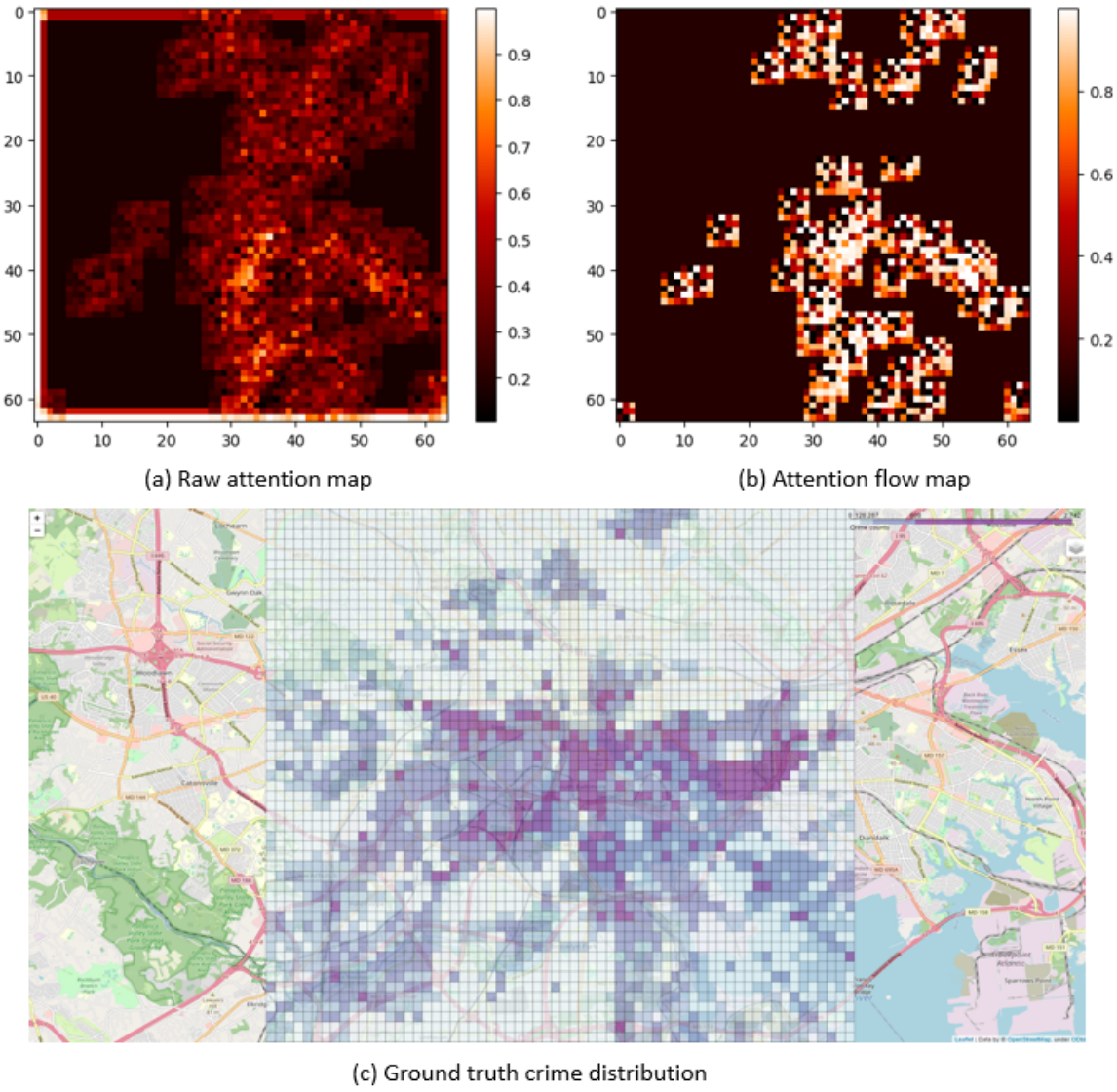


Figure 4.5: Attention flow visualization for operation of the ABC-LSTM model on the Baltimore City Crime Dataset.

can be easily extended to other CNN/RNN based models and it shows superior performance on real-life spatio-temporal forecasting applications;

- The combination of attention block and other models also benefits from improved interpretability providing more insight into the spatial patterns of contribution to forecasted crime risk.

By introducing attention mechanism to ConvLSTM and other spatio-temporal deep learning models, we can retain the temporal information, such as the order of time series data, while capturing global spatial dependencies without complex preprocessing. Our proposed model shows fairly robust results both on benchmark datasets and practical fields. Also, with certain supplementary algorithms, we can retrieve a rather intuitive attention map comparing to the raw attention maps. It is also worth noting that enhanced interpretability in the form of visualization can further help with iterative model improvement. For future work, one promising direction is to investigate how we can leverage attention maps along with domain-specific knowledge to optimize models and enhance interpretability further for practical use in the field.

Chapter 5

Decidable variable-rate dataflow for heterogeneous signal processing systems

5.1 Chapter Overview

Dynamic dataflow models of computation have become widely used through their adoption to popular programming frameworks such as TensorFlow and GNU Radio. Although dynamic dataflow models offer more programming freedom, they lack analyzability compared to their static counterparts (such as synchronous dataflow).

In this chapter we advocate the use of a boundedly dynamic dataflow model of computation, VR-PRUNE, that remains analyzable but still offers more programming freedom than a fully static dataflow model.

The chapter presents the VR-PRUNE model of computation and runtime, and illustrates its applicability to practical signal processing applications by two use cases: an adaptive convolutional neural network, and a predistortion filter for wireless communications. By runtime experiments on two heterogeneous computing platforms we show that VR-PRUNE is both flexible and efficient.

Material in this chapter was published in partial, preliminary form in [104].

5.2 Introduction

In the recent years, dataflow has been used as the underlying model of computation (MoC) in TensorFlow for machine learning applications [105], and in GNU Radio for radio communications [106]. Dataflow MoCs provide an abstract and efficient for way describing various signal processing applications. Under dataflow, an application is described as a graph that consists of nodes and directed edges that interconnect the nodes. The nodes, called *actors*, perform computations, whereas the edges act as order-preserving communication channels between actors. The data that is communicated over the edges is packaged into *tokens* of fixed size. An actor can trigger (*fire*) its computations when its input edges have a sufficient number of tokens available – this token quantity is called the *input token rate*. Respectively, during its firing the actor produces a number of tokens to each of its output channels. The channel-specific number of tokens produced is called *output token rate*.

In one of the most studied dataflow MoCs, Synchronous Dataflow (SDF) [107], the data quantities produced and consumed by each actor need to be fixed and known at application design time. This restriction however enables a variety of compile-time optimizations, as well as full consistency checks at design time for, e.g., ensuring that the application cannot deadlock [108] or cause communication buffer overflows. If the dataflow MoC allows such a full consistency check in finite time, the MoC is called *decidable*. Consequently, SDF has been used for specifying a variety of complex digital signal processing systems [109] [110] [111].

Despite the advantages of fully static dataflow MoCs, there are situations where

fixed token rates are too restrictive. Examples of such a case are adaptive signal processing applications where the application behavior changes at runtime in response to changes in the environment. An example of such an application, used in this chapter, is a neural network for video analysis that can adjust its inference frequency based on video motion detection.

In order to properly support such dynamic behavior, a dynamic dataflow MoC is needed. Consequently, actors' token rates can vary at runtime in response to dynamically varying computation needs. As a drawback, such flexibility restricts application analyzability. An example of a dynamic dataflow model is Boolean dataflow [112] that is not a decidable MoC.

Between the extremes of fully static and fully dynamic dataflow, a variety of boundedly dynamic dataflow MoCs exist, of which a well-known example is Well-Behaved Dataflow (WBDF) [113]. Recently, Boutellier et al. presented the PRUNE MoC and framework [114], which similar to WBDF allows dynamic application behavior to be expressed within dataflow subgraphs that follow a constrained graph topology. A different branch of work is variable-rate dataflow (VRDF) [115] that allows port token rates to change arbitrarily within predefined limits.

In this chapter, we introduce a novel dataflow MoC that is a hybrid between the PRUNE MoC [114] and the VRDF MoC [115]: VR-PRUNE keeps the decidability of PRUNE, but adds to it support for variable token rates. *Here we show that VR-PRUNE increases MoC flexibility over the PRUNE MoC without having a negative impact on run-time efficiency*. The proposed VR-PRUNE addresses the inherent conflict between analyzability and expressiveness in a different way: the VR-PRUNE MoC enables describ-

ing conditional execution, however regulated by a set of formal rules and design patterns, which ensure that graph inconsistencies can be detected at design time. Some early results related to the proposed MoC were presented in our previous work [116], while in this chapter the VR-PRUNE MoC is formalized (as far as the available space allows), and the MoC's run-time efficiency is illustrated by experiments.

5.3 Background and Related Work

In this section, we talk about the development of related dataflow models and briefly present the PRUNE MoC [114] that acts as the basis of the proposed VR-PRUNE MoC.

In the dataflow programming concept, applications are expressed as directed *graphs*. The application graph consists of nodes, which are called *actors* that are interconnected by *edges*. Edges carry data that is encapsulated within *tokens*; for example, in an image processing application, a single token may encapsulate the data related to one image pixel, or the data of a complete frame, depending on the application.

The interconnection between an actor and an edge is called a *port*. A port that is connected to an edge, which departs from an actor is called an *output port*, and respectively a port that is connected to an incoming edge is called an *input port* of an actor.

Another well-known model for describing information flow in dynamic discrete systems is Petri nets [117]. The main advantage of dataflow graphs over Petri nets is their compactness and convenience in mapping real-life applications to graph-based models [118]. However, a significant class of dataflow models can be transformed to Petri nets, thus enabling application of analytic methods that have been designed for Petri nets,

to dataflow models [119].

Most of the differences between existing dataflow MoCs associate with rules and restrictions related to token rates. *Homogeneous* dataflow MoCs require that every port of each actor has strictly a token rate of one, whereas the synchronous dataflow (SDF) [107] MoC allows a fixed positive integer token rate for each port. Examples of design frameworks that are based on SDF are PREESM [120] and StreamIt [121, 122]. Cyclo-static dataflow (CSDF) [123], on the other hand, enables token rates to change in fixed, periodic cycles. SDF (and its homogeneous variant) and CSDF are regarded as *static* dataflow MoCs, as their token rates are completely predetermined at application design time. Consequently, the dataflow graphs are analyzable at compile-time, enabling formal proofs for absence of deadlocks and bounded memory.

Somewhat more flexibility can be added to the SDF MoC by adding *scenarios*, different SDF graph topologies that are at runtime switched, e.g., based on a finite state machine. The baseline work in this direction is SADF [124], and recently also full software design frameworks such as HOPES+ [125] have been built around the SADF concept.

Dynamic dataflow MoCs, of which *dataflow process networks* (DPN) [126] is one of the most well-known examples, allow port token rates to change arbitrarily at application run time, and therefore possibilities for graph consistency verification at design time are very limited. In the past decade, dynamic dataflow around the CAL language [127] and its subvariant RVC-CAL [128] has triggered the development of several design frameworks such as Tÿcho [129], Orcc [130] and SHeD [131].

PRUNE, where an application is expressed as a graph $G = (A, F)$, where A is a set of actors and F is a set of FIFO channels. A FIFO channel is connected to an actor

through an actor *port*; each actor $a \in A$ can have any number of input and output ports. We use the following notation: $a = parent(p)$ denotes an actor a contains port p, and p_{a1}^{+} stands for output port 1 of actor a. The superscript $+/-$ corresponds to output/input port direction, respectively.

For FIFO $f \in F$, we say ports p_a^+ and p_b^- are *connected* if $fifo(p_a^+) = fifo(p_b^-)$, where actors a and b are referred as the source and sink of the same FIFO. In the PRUNE MoC, an output port p^+ can be connected to multiple FIFOs, but every input port p^- has only one FIFO connected to it and each FIFO has unique source and sink ports.

There are three types of ports in the PRUNE MoC: the control (input/output) port, the static regular port (SRP) and the dynamic regular port (DRP). SRPs have a fixed token consumption/production rate, whereas DRPs have two fixed token rates, which are respectively denoted as *active token rate* and *inactive token rate* of p, also abbreviated as $atr(p)$ and $itr(p)$. The token rate determines how many tokens the actor consumes (for input ports) from its associated edge, or how many tokens the actor produces (for output ports) to its associated edge upon one *firing*. Morever, each FIFO has a single, positive-integer token rate, denoted as $fiforate(f)$, and we restrict that $atr(p) = fiforate(fifo(p))$, in other words the active port rate must strictly equal to the token rate of the connected FIFO. By this *symmetric-rate dataflow* behavior, PRUNE requires the token consumption rate equal the production rate for every FIFO. Even though this makes PRUNE more restricted than SDF with respect to token rate conversion, PRUNE on the other hand allows dynamic token rates that are not supported by SDF. Meanwhile, the delay of FIFOs along with the symmetric-rate behavior can lead to unaligned FIFO accesses.

Two PRUNE actors are $adjacent$ if each of them has at least one port connected over

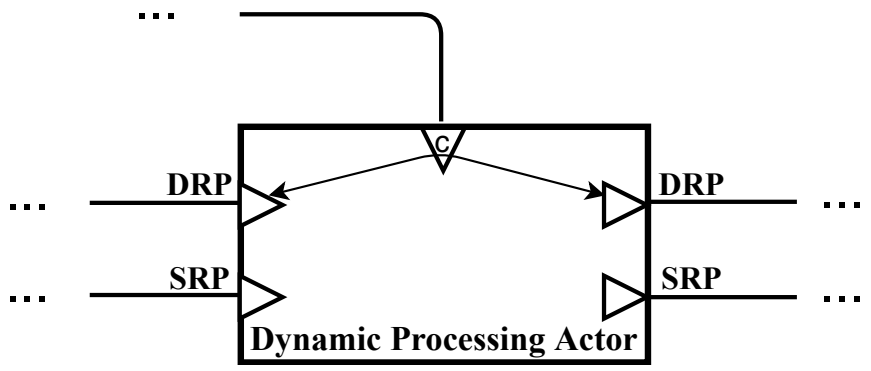


Figure 5.1: An illustrative example of a dynamic processing actor.

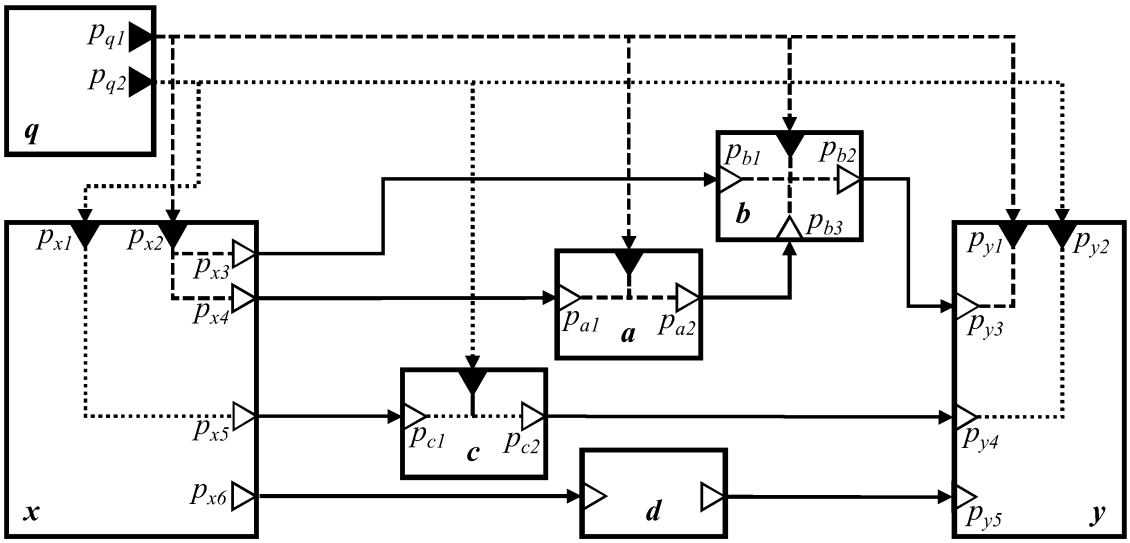
Table 5.1: **Comparison to related dataflow models and languages.**

Work	Deci-dable	Dynamic	High-performance	Variable-rate
SADF [124]	+	+	-	+(-)
BDF [132]	-	+	-	-
DAL [133]	-	+(-)	+	+(-)
RVC-CAL [130]	-	+	+	+
StreamIt [122]	+	-	+	-
PRUNE [114]	+	+	+	-
VRDF [115]	+	+	-	+
VR-PRUNE	+	+	+	+

the same FIFO. A *chain* $S = (a_1, a_2, ..., a_n)$ is a set of actors for which $\forall i = 1, 2, ..., n$, a_i and a_{i+1} are adjacent, and S connects a_1 and a_n.

5.4 The proposed VR-PRUNE MoC

The proposed VR-PRUNE MoC is a generalization of the PRUNE MoC in that VR-PRUNE allows token rates of certain ports to dynamically vary within predefined limits, similar to VRDF [115]. In the following subsections, we identify the new structures and required changes compared to the PRUNE MoC to enable this variable-token rate behavior.



T	p_{x3}	p_{x4}	p_{x5}	p_{a1}	p_{a2}	p_{b1}	p_{b2}	p_{b3}	p_{c1}	p_{c2}	p_{y3}	p_{y4}	*rate*
p_{q1}	1	1	0	1	1	1	1	1	0	0	1	0	0..2
p_{q2}	0	0	1	0	0	0	0	0	1	1	0	1	0..3

Figure 5.2: **VR-PRUNE subgraph example.** An example of control relationships between a configuration actor (q), dynamic actors (x, y), and DPAs (a, b, c) expressed equivalently by a control table, and graphically. Dashed actor-actor connections visualize edges between control ports, whereas dashed connections within actors illustrate control relationships according to the control table T. Note: the *rate* column refers to the range of valid atr values (defined by lrl and url) emitted by the output control port.

5.4.1 VR-PRUNE actor types

The varying token rate feature requires a new type of actor, the *dynamic processing actor*, that is introduced among other actor types that are identical to those in PRUNE.

a. Static Processing Actor: Static processing actors (SPA) can only have SRP ports, and therefore during every firing of an SPA all ports p must have the token rate $atr(p)$.

b. Configuration Actor: A configuration actor (example: q in Fig. 5.2) can have one or multiple control output ports which must be connected to the control input port of a dynamic actor or a dynamic processing actor. The control output ports must be SRPs with a token rate of unity. Besides, a configuration actor may have zero or more input or output

data ports of type SRP.

c. Dynamic Actor: A dynamic actor has at least one DRP, one control input port and any number of SRPs. All the DRPs have to be either input ports, or output ports. Every time a dynamic actor performs a firing, a token is consumed from each of its control input ports. *In contrast to PRUNE, values of tokens originating from the control input port set the token rate of each DRP p to any integer value between 0 and $atr(p)$.*

Fig. 5.2 shows a VR-PRUNE subgraph with one configuration actor q and two dynamic actors, x and y. The figure illustrates how the token values originating from black-colored input control ports p_{x1} and p_{x2} associate with the DRPs p_{x3}, p_{x4} and p_{x5} in a one-to-many relationship. For actor x that has a DRP p, $cport(p)$ is the input control port of x that is associated with p. Drawing an example from Fig. 5.2, $cport(p_{x3}) = p_{x2}$.

d. Dynamic Processing Actors (DPA): DPAs are similar to dynamic actors, but differ in the sense that a DPA is required to have at least one input DRP and at least one output DRP, as well as at least one control input port. Also, a DPA can have any number of SRPs. Similar to dynamic actors, the control input port sets the token rates of DRPs to any integer value between 0 and $atr(p)$.

Fig. 5.1 provides an illustration of a DPA: the control input port sets the token rates of a pair of DRPs (one on the input side, the other on the output side). When the actor performs a firing, the value of a single token coming from the control port sets the token rate of both DRPs. In Fig. 5.2, a, b and c are DPAs. Out of these, b has two input DRPs (p_{b1} and p_{b3}) and one output DRP (p_{b2}), all of which are controlled by the single input control port of b.

Fig. 5.3 on the other hand, shows a VR-PRUNE subgraph with a configuration actor

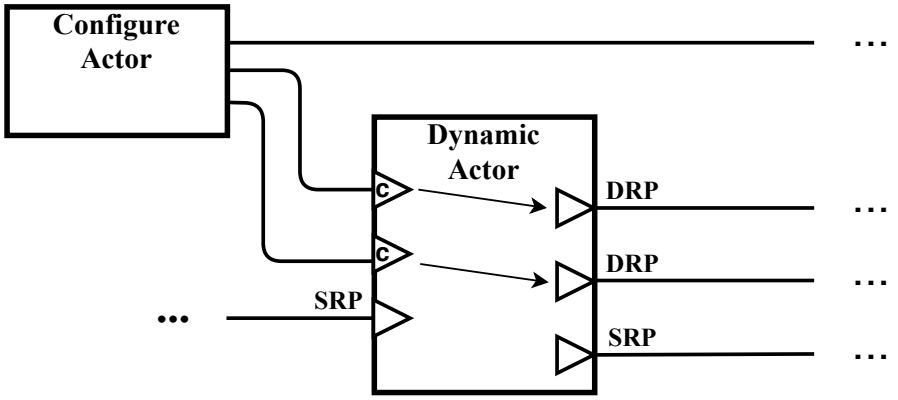


Figure 5.3: Configure actors and dynamic actors, configure actors control two DRPs by a control table.

and a dynamic actor. The port with notation 'c' is considered as control input port. Two DRPs are both on the output side, and are configured by two consecutive inputs consumed by control input port respectively.

5.4.2 VR-PRUNE design rules

Similar to PRUNE, VR-PRUNE imposes a small set of concrete design rules to ensure that graph topologies leading to inconsistent dataflow behavior are avoided. This is achieved by five *design rules* that are identical to those in PRUNE, except for the *connecting subchain* rule (3).

1. Linked port control rule: A linked pair of DRPs $\{p_x, p_y\}$ must be controlled by the same control output port.

2. Balanced delay rule: If a control output port p_c controls two control input ports p_x, p_y, then $delay(p_c, p_x) = delay(p_c, p_y)$.

3. Connecting subchain rule: If $\{p_x, p_y\}$ are DRPs linked by the chain $S = (a_1, a_2, ..., a_n)$, then (1) $a_i, i = 1, 2, ..., n$ must all be actors of type SPA or DPA; (2)

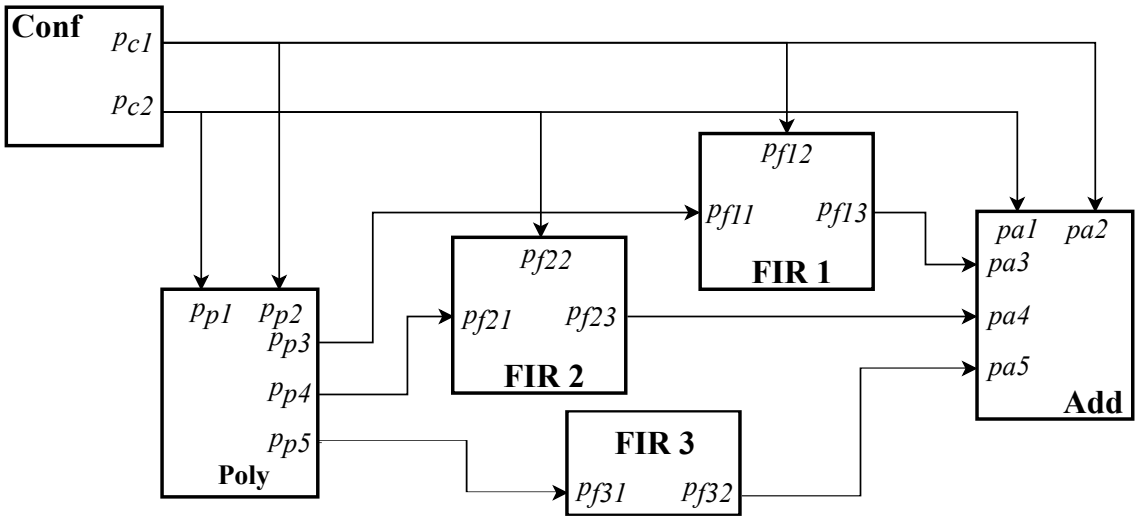


Figure 5.4: A VR-PRUNE dynamic processing graph.

every connecting subchain to which $a_i \in S$ belongs, must be associated with the two dynamic actors x and y.

4. Single-sided dynamism rule: A dynamic actor can only have either input DRPs or output DRPs, not both.

5. Encapsulation rule: If $\{p_x, p_y\}$ are linked DRPs whose parents are x and y, $S = (a_1, a_2, ..., a_n)$ is a chain, and $k \notin S$ is an actor that connects to $a_i, i = 1, 2, ..., n$, then k must belong to a chain that connects x and y.

5.4.3 VR-PRUNE subgraphs and control tables

The VR-PRUNE design rules require that dynamic token rates occur within a specific type of subgraph, the Dynamic Processing Graph (DPG). A DPG consists of one configure actor, two dynamic actors and any number of DPAs or SPAs. Below, an example of a DPG is described.

Fig. 5.4 shows an example of a VR-PRUNE DPG, where FIR1 and FIR2 are dynamic

processing actors, p_{f11}, p_{f13} and p_{f21}, p_{f23} are DRPs, controlled by p_{f12}, p_{f22} respectively. The Poly actor is a dynamic actor with DRPs on the output side and the Add actor is also a dynamic actor with DRPs on the input side. Conf is a configuration actor and FIR3 is a static processing actor.

The values of tokens produced by the configuration actor's control output ports can be illustrated by a *control table*. In the VR-PRUNE MoC, variable token rates are restricted to subgraphs called Dynamic Processing Graphs (DPGs) that define graph-level structure for ensuring analyzability. DPG types may impose additional restrictions to actor types, actor port counts, or connectivity between actors.

The complete VR-PRUNE application graph G can consist of any number of interconnected DPGs: dynamic actors and configuration actors of a DPG are allowed to connect outside the DPG over SRPs without restrictions. In fact, using the concept of hierarchical dataflow graphs (e.g. [134]), a DPG could be represented as a composite actor with static token rate ports. Formal presentation of hierarchical graphs is however limited outside the scope of this work as clustering of actors may change the graph semantics and cause deadlock [134]. In the rest of this chapter the analysis concentrates on internal behavior of individual DPGs.

Fig. 5.2 shows an example of a DPG, where q is a configuration actor, a, b and c are DPAs, x and y are DAs, and d is an SPA. Each DPG is associated with a *control table* T that unambiguously defines 1) the control relationships between output control ports of configuration actors, and DRPs of DAs and DPAs, and 2) sets limits for variable token rates by means of output control port specific lrls and urls.

The control table of Fig. 5.2 is shown below the DPG — it is a matrix with dimen-

sions $h \times (w + 1)$, where h and w equal the number of output control ports and DRPs, respectively. In the control table, a value of '1' indicates a control relationship between the corresponding output control port (row) and DRP (column), whereas a value of '0' indicates that the control port and DRP are not related. Since each DRP is required to be controlled by exactly one output control port, a valid control table must have a column sum of '1' for each DRP column. The lrl and url values are defined per output control port in the last column of the control table, and apply to all DRPs that are associated ('1') with that row.

Finally, we define the meaning of a *complete cycle* related to a DPG in the spirit of [132]: assuming that a DPG is consistent, we define a complete cycle of a DPG as a sequence of actor executions that returns the DPG to its original state. The execution of a complete cycle S of actors of the form $S = q_1, q_2, ..., q_m, a_1, a_2, ..., a_n$, where the q_i's are the control actors of the DPG, and the a_i's are the SPAs together with the active DAs and DPAs of the DPG, as determined by the execution sequence $q_1, q_2, ..., q_m$. Here, an *active* DA or DPA a means that (1) a has input tokens on all control ports, and (2) a is configured by the incoming control tokens so that at least one DRP of a will have a nonzero rate on the next actor firing. To be valid, a complete cycle must also satisfy the condition that there is no FIFO buffer underflow within the edges of the DPG when executing S.

5.4.4 VR-PRUNE MoC summary

The novelty of VR-PRUNE over PRUNE is the support for token rates that vary within predefined limits. Whereas in PRUNE DRP ports had two token rates, *itr* and *atr*, VR-

Table 5.2: Platforms used for experiments.

Tag	GPPs	GPU	Operating System
i7-940MX	Intel i7-6700HQ @ 2.60 GHz	NVidia GeForce 940MX	Ubuntu 18.04, g++ 7.0.0
i7-GTX1080	Intel i7-8700K @ 3.70GHz	NVidia Geforce GTX1080	Ubuntu 18.04, g++ 7.0.0

PRUNE allows these ports to have any integer token rate between *itr* and *atr*. To realize this, VR-PRUNE introduces a new actor type, the DPA, as well as a modification to the *connecting subchain* design rule. Unfortunately, due to strict space limits, it is not possible to show decidability proofs of the VR-PRUNE MoC here.

In the next section, we will show by experiments with the VR-PRUNE runtime that this added design and MoC freedom of VR-PRUNE does not add computational overhead in the context of heterogeneous target platforms.

5.5 Experimental results

In this section, we present two application use cases for VR-PRUNE and measure the efficiency of the VR-PRUNE runtime. Similar to PRUNE, VR-PRUNE is designed for heterogeneous systems that use both a GPU and multiple CPU cores for computations. The considered target platforms are shown in Table 5.2. The designer provides actor descriptions in C or OpenCL, based on which the VR-PRUNE compiler generates a toplevel file that realizes inter-actor communication.

Table 5.3: **Adaptive digital predistortion throughput** in complex float Megasamples/s, as a function of enabled filter branches (%), on the i7 CPU and on the N2 CPU. Higher is better.

Framework	20%	40%	60%	80%	100%
PRUNE i7	**90.77**	45.75	30.50	22.60	18.29
VPRF i7 (prop.)	88.44	**46.86**	**31.62**	**23.63**	**18.68**
PRUNE N2	17.88	9.53	6.50	4.88	4.11
VPRF N2 (prop.)	**18.25**	**9.62**	**6.77**	**5.20**	**4.29**

Table 5.4: Adaptive digital predistortion lines of code per actor.

Framework	*s*	*q*	*x*	*f*	*y*	*t*	mean
PRUNE	59	64	139	44	141	48	**83**
VR-PRUNE	59	68	103	64	106	48	**75**

5.5.1 Adaptive digital predistortion

Digital predistortion is a signal processing approach for compensating non-linear effects of a wireless transmitter's power amplifier [135]. The signal processing for predistortion is computationally very demanding, and therefore solutions [136] that trade-off predistortion bandwidth against computational effort are useful for saving power when possible, e.g. due to varying interference conditions.

Fig. 5.5 shows such an adaptive predistortion filter in two implementations: a) is a VR-PRUNE implementation, and b) is the conventional PRUNE implementation introduced in [114]. Both implementations describe the same 10-tap parallel Hammerstein filter structure [137], with adaptive functionality such that individual filter branches can be enabled or disabled on-the-fly at run time. In Fig. 5.5, s_1 and s_2 are source actors that provide samples from the transmitter baseband side; x is a dynamic actor that computes polynomial basis functions and distributes the samples to parallel FIR filter branches f; y

is a dynamic actor that implements a summation function for combining the filter branch outputs and compensates for I/Q imbalance; t_1 and t_2 finally act as output actors towards the power amplifier. Adaptive processing is controlled by the configuration actor q that based on external input can enable/disable filter branches $f_1 ... f_4$ and $f_7 ... f_{10}$ in the PRUNE implementation. In the VR-PRUNE implementation the same effect is accomplished by variable token rates to/from the actor f. In the PRUNE implementation the FIR filter actors are SPAs, whereas the VR-PRUNE FIR actor f is a DPA.

It can be seen that the VR-PRUNE implementation (Fig. 5.5a) compactly captures the adaptive predistortion functionality within 8 actors and 11 FIFOs, whereas the PRUNE implementation (Fig. 5.5b) requires 17 actors and 46 FIFOs (double-line arrows stand for 2×FIFOs, one for the I channel, and one for the Q channel). This reduction of graph elements comes from the fact that VR-PRUNE is able to capture the functionality of actors $f_1 ... f_{10}$ within a single actor f, and consequently reduces the number of connecting FIFOs from 40 down to 4. Based on the number of architectural elements, VR-PRUNE therefore reduces the predistortion model complexity by 70% compared to PRUNE. This reduction ratio depends on the structure of the original dataflow graph, more specifically on the count of f (filter) actors.

One potential fallacy in evaluating model complexity reduction as described above is that actor and edge counts can often be reduced by just lumping actors together. For example, an entire SDF graph can be implemented as a single actor, thereby reducing the graph size to 1 actor and 0 edges. To show that this is not the case here, Table 5.4 shows the lines of code per actor for the Fig. 5.5 graphs. It can be seen, that on average, the VPRF implementation requires fewer lines per code than the PRUNE implementation.

Looking at the dynamic actors x and y, it can be seen that the PRUNE implementations of these actors have around 30% more code than their VPRF counterparts, which is related to inter-actor communication: initiating and terminating inter-actor data transfer requires two lines of code per FIFO. Since the PRUNE x and y actors have 20 FIFOs towards the f actor each, whereas the VR-PRUNE x and y have only 2, the difference is obvious. In contrast, the VPRF f actor has 20 lines of code more than the PRUNE counterpart – this is due to the fact that the single f actor stores all the filter coefficients that in the PRUNE implementation are distributed over $f_1 ... f_{10}$.

Finally, Table 5.3 shows that the VPRF implementation (Fig. 5.5a) does not cause computational overhead compared to the conventional PRUNE implementation (Fig. 5.5b), which is shown for both i7 and N2 platforms. It is worthwhile to point out that reducing the number of actors as between Fig. 5.5b and Fig. 5.5a evidently decreases potential for parallel execution. Therefore, if the underlying execution platform has spare cores, it is not advisable to collapse all actors $f_1 ... f_{10}$ into a single actor, but instead consider distributing the computation effort evenly over available compute resources.

5.5.2 Dynamic-update predistortion filter

The dynamic-update predistortion filter (DU-DPD) is a filter for wireless transmitters that in real-time filters the baseband signal to be transmitted. The VR-PRUNE implementation of the DU-DPD has an online learning capability, which allows updating the filter coefficients periodically to accommodate with varying RF interference.

As Fig. 5.6 shows, actors Src, $Learn$ and $Update$ form the varying-token rate

learning chain, where Src and $Update$ are dynamic actors, and $Learn$ is a DPA. This dynamic-token rate feature allows any number of samples between 1 and X to be used as training data to acquire new filter coefficient values. The actor $Filter$ on the other hand operates in real time on the GPU at a constant token rate of N

Fig. 5.7 shows the execution time of the whole DU-DPD graph for one iteration as a function of *kilo*samples used (X) for training in the $Learn$ actor. Here, the minimum value of X was 1, and the maximum value was 10; intermediate sample rates were varied with a stepping of 1. The figure shows that in this applications the variable sample rate feature allows reducing the number of samples used for learning in situations when there is little or no RF interference, with significant computation time savings.

5.5.3 Adaptive Convolutional Neural Network

The dataflow graph of our neural network use case is depicted in Fig. 5.8; it consists of two GPU-accelerated convolution layers (L1Conv, L2Conv) followed by a GPU-accelerated dense layer, as well as two more dense layers that have been combined into a single actor (L3Relu-L5). The application is adaptive in the sense that it allows dynamically enabling and disabling convolutional neural network inference for individual frames, e.g. based on inter-frame motion detection.

In the dataflow sense, the image data related to one frame (image) forms a single token. For accelerating GPU processing, $N = 24$ frames are processed in parallel, but for each of the 24 frames, the inference can be disabled, which makes the token rate vary in steps of 1 between values 0 and 24. Fig. 5.8 shows the varying-token rates of ports by X

(= [0, 24]) and fixed rates by N (= 24).

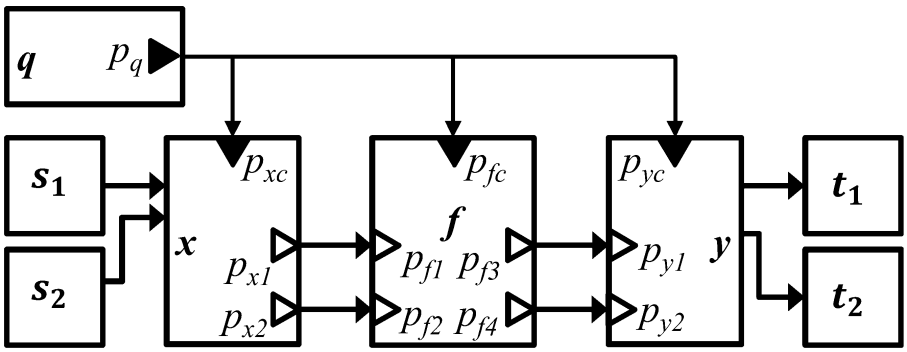
Fig. 5.9 shows the processing time of the whole graph for a sequence of 384 frames on two platforms (Table 5.2). The CNN inference (on/off) for each frame was randomly varied at runtime achieving the average percentage of frames processed for 0%, 12.5%, 25.0%, 50.0% and 100%, measuring the execution time time for each percentage value.

For this use case the conditional CNN inference feature was implemented under conventional PRUNE using the *conditional statements* approach (See [116]) and by the variable token rate feature of VR-PRUNE. The results show that the variable token rate feature of VR-PRUNE does not impose any overhead compared to conventional PRUNE – in contrast, VR-PRUNE is computationally slightly more efficient than conventional PRUNE due to the fact that variable data rate processing allows omitting data transfer for skipped frames.

5.6 Conclusion

We have introduced VR-PRUNE, a Model of Computation and runtime for signal processing on heterogeneous platforms. Experimental results show that VR-PRUNE is both more flexible and efficient than its predecessor PRUNE.VR-PRUNE MoC has great capability for decribing practical signal processing application and expressiveness for the dynamic process, like DU-DPD application. According to the DNN experiment, we can see VR-PRUNE model has better analysis efficiency than previous presented DAL framework.

a) VR-PRUNE graph of adaptive digital predistortion.



T	p_{x1}	p_{x2}	p_{f1}	p_{f2}	p_{f3}	p_{f4}	p_{y1}	p_{y2}	*rate*
p_q	1	1	1	1	1	1	1	1	1..5

b) PRUNE graph of adaptive digital predistortion.

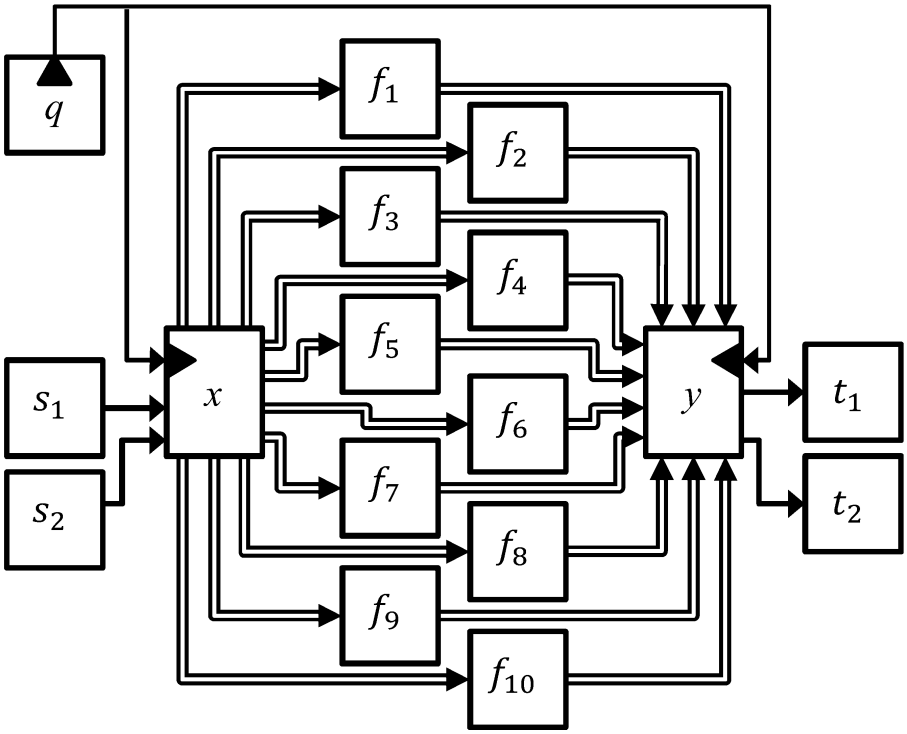


Figure 5.5: **The adaptive digital predistortion application.** Subfigures: a) VR-PRUNE graph, b) PRUNE graph. In Subfig. b) double-line arrows depict 2×FIFOs.

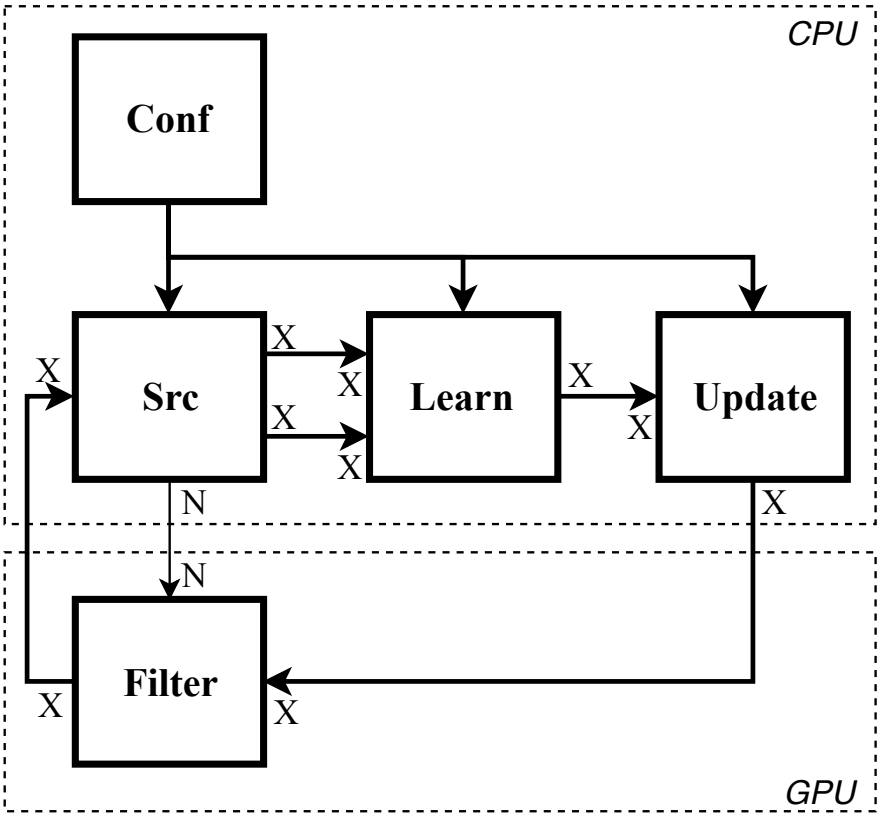


Figure 5.6: VR-PRUNE model for the DU-DPD use case.

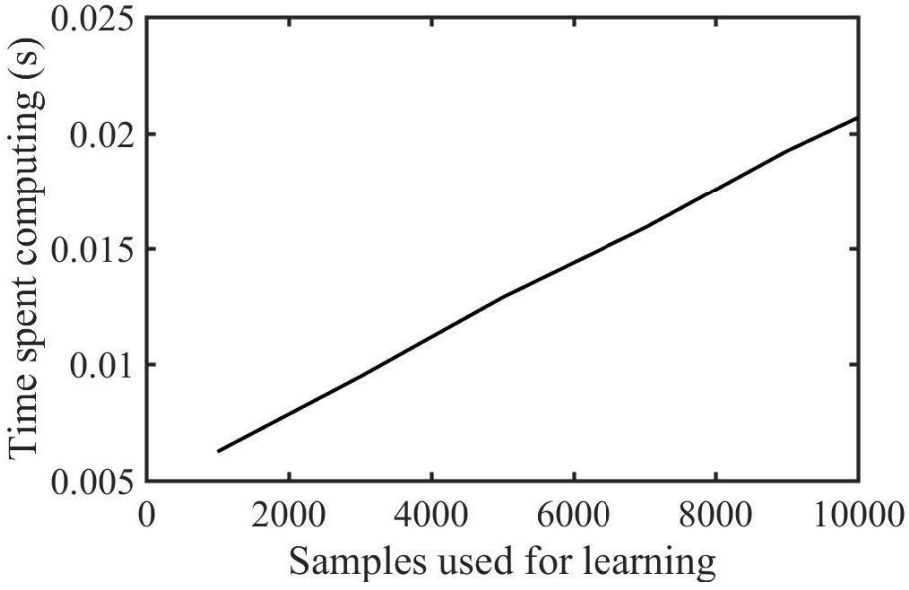


Figure 5.7: Time spent in DU-DPD under VR-PRUNE on i7-GTX1080, as a function of samples used for learning.

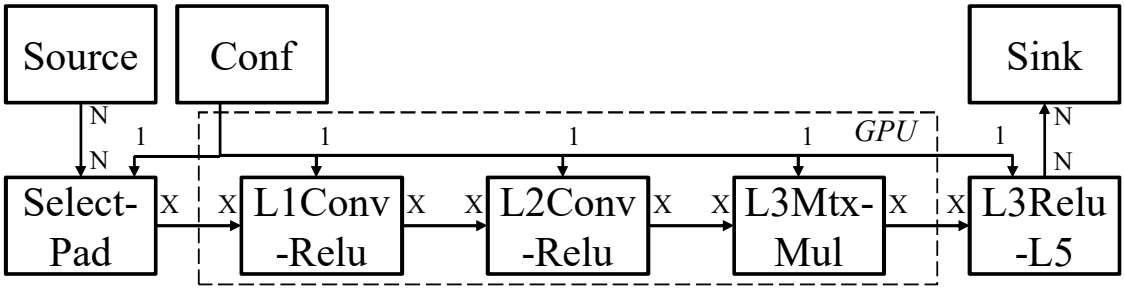


Figure 5.8: The adaptive CNN application.

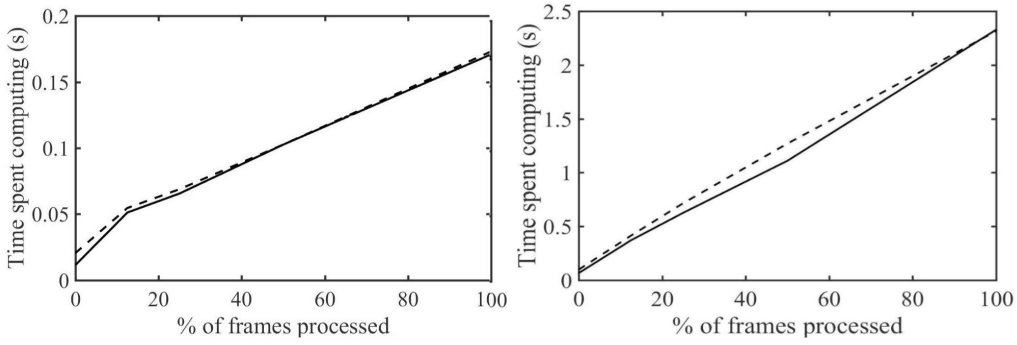


Figure 5.9: Time spent in the CNN application as a function of % of frames processed: PRUNE (dashed line) vs. VR-PRUNE (solid line) on i7-GTX1080 (left) and i7-940MX (right).

Chapter 6

Dynamically Reconfigurable Perception using Dataflow Parameterization of Channel Attention

6.1 Chapter Overview

Despite the increasing performance of deep learning models for intelligent perception tasks, the size and complexity of state-of-the-art models have become increasingly large, which makes them difficult to deploy in resource-constrained edge computing environments. Moreover, the conditions and constraints under which these models operate may be impossible to fully anticipate at design time, and may undergo significant changes during inference. In this chapter, we address these challenges by developing a new parameterized design approach for image-based perception that enables efficient and dynamic reconfiguration of convolutions using channel attention. Compared to switching among sets of multiple complete neural network models, the proposed reconfiguration approach is much more streamlined in terms of resource requirements, while providing a high level of adaptability to handle unpredictable and dynamically-varying operational scenarios. The efficiency and adaptability of the proposed approach are demonstrated on ResNet50 integrated with the Convolutional Block Attention Module (CBAM) concept.

Material in this chapter was published in partial, preliminary form in [138].

6.2 Introduction

Deep Neural Network (DNN) models have dominated many machine learning applications in recent years. However, the increasing size of DNN models brings about major challenges for model deployment on resource-constrained devices. The convolution operator is one of the most general and powerful tools to extract features from multiple kinds of input. Increasingly large convolutional neural networks (CNNs) are employed to extract increasingly complex and high-level features in the input data [139, 140]. As networks become deeper, the kernels can be optimized to produce relatively larger activation responses when presented patterns corresponding to larger receptive fields (i.e., more effective pixels) and high frequency details.

Convolution layers are widely favored in the field of machine learning due to their remarkable suitability for analyzing data structured in grid-like or sequential patterns, as seen in images and audio. Despite the impressive expressive power of CNNs, for a modern DNN, such as ResNet50 [141], the convolutional layers account for over 80% of the total computation cost. The massive computation requirement hampers inference on edge devices due to limitations in memory and energy capacity. In this chapter, we seek to address these problems by developing a new parameterized design approach for deep learning models that enables efficient and dynamic reconfiguration of convolutions using channel attention. Compared to switching among sets of multiple complete neural network models, the reconfiguration approach developed in this chapter is much more streamlined in terms of resource requirements, while providing a high level of adaptability to handle unpredictable and dynamically-varying operational scenarios.

Channel attention is applied to CNNs to selectively emphasize specific channels in feature maps from the network's intermediate representations. The attention is designed to help the network focus on meaningful features during the learning process, enhancing its performance. In this chapter, we use attention as an indicator to guide model pruning. In this context, a smaller channel attention score implies a poorer contribution to the final output from the given channel, and a greater priority for removing the channel during inference under resource-constrained operation.

Model pruning and hardware acceleration are two major methods to solve efficiency problems in DNN systems. Dataflow modeling approaches can help with both aspects [142, 143]. In this chapter, we introduce a novel dataflow modeling approach for integration of channel attention into DNN design optimization. In particular, we apply dynamic dataflow methods to manage the intermediate outputs in pre-trained DNN models. This approach allows for rapid adjustment of different model variants—having different operational trade-offs—without the need for changing overall model architecture or retraining. Since the objective is to enable rapid adaptation to changing application requirements, architecture modification and retraining have prohibitive overhead (due to inefficient resource usage); our dataflow-based adaptation avoids such overhead.

The conditions and constraints under which DNN models operate may be impossible to fully anticipate at design time, and may undergo significant changes at run-time. This chapter develops integrated methods involving channel attention and dataflow modeling to control the adaptation of DNN models in a manner that provides optimized performance subject to dynamic changes in operational conditions and constraints. The methods that we develop have the following advantages:

• efficient adaptation in response to dynamically-varying application requirements or to carry out a pre-defined schedule of different operating modes;

• utilizing variable rate dataflow methods to systematically integrate channel attention with DNN functionality, which enables efficient and reliable adaptation of model inference;

• integration of variable-rate dataflow modeling with the Open Neural Network Exchange (ONNX) [144], which is a widely-used open-source tool for representing and exchanging deep learning models across different machine learning frameworks and platforms.

The remainder of this chapter is organized as follows. In Section 6.3, we survey literature that is related to channel pruning and dataflow modeling of DNN models. In Section 6.4, we propose the method to prune ResNet + CBAM (Convolutional Block Attention Module) models based on channel attention. In Section 6.5, we conduct a series of experiments to analyze the performance of our proposed pruning methods, and demonstrate their effectiveness. We also discuss limitations of the proposed methods in Section 6.5. In Section 6.6, we summarize our findings in this chapter, and present useful directions for future work.

6.3 Background and Related Work

In this section, we summarize recent works about the development of channel attention techniques, how such techniques help with model pruning, and general methods to work with DNN models as dataflow graphs.

6.3.1 Channel Attention

The original aim of channel attention is to assign relevance scores to channels within convolutional layers, and facilitate the exploration of inter-channel relationships[145]. Early channel attention approaches were used in so-called Squeeze-and-Excitation (SE) Networks, which employed global average-pooled features to compute channel-wise attention[146].

Woo et al., in their development of the CBAM [147] concept, advanced channel attention methods by introducing both spatial and channel attention, thus enhancing the expressive power of attention mechanisms within CNNs. They also explored the combination of average pooling and max pooling to aggregate features. Building on these developments, Wang introduced Efficient Channel Attention (ECA) [148], which provides a local cross-channel interaction strategy without dimensionality reduction. ECA has a lower computational cost compared to other attention mechanisms.

Additionally, Misra et al. [149] explored triplet attention, which captures the importance of features across dimensions, for example channel attention and spatial attention, in a tensor. Qin et al. [150] reevaluated the extraction of global information, emphasizing compression as a pivotal perspective. Their analysis focused on the frequency domain, especially global average pooling. They demonstrated that global average pooling could be understood as a specific instance of the discrete cosine transform (DCT).

6.3.2 Pruning CNN Channels

Channel pruning is a vital technique to reduce the computational and memory requirements of a CNN without significantly sacrificing its performance. Liu introduced the concept of *network slimming* [151], which hinges on the application of L1-norm regularization to scaling factors in batch normalization layers, enabling the precise pruning of channels from deep, pre-trained CNNs.

Building upon this foundation, Liu et al. designed the Channel Pruning guided by Spatial and Channel Attention (CPSCA) approach [152], which attains better inference accuracy than other state-of-the-art pruning methods at similar pruning ratios. Their technique relies on channel scale, a statistical metric derived from element-wise averaging of the channel attention map to measure the channel importance.

Unlike other methods that rely solely on the CNN output for channel pruning, CNN interpretability theory involves guiding channel pruning through mask layers [153]. A mask layer refers to a CNN layer that applies a binary mask to the feature maps of a CNN. This mask selectively turns off certain channels, effectively allowing the network to focus on more relevant features for its tasks. A class-wise mask can be derived for each channel to configure the contribution of the channel in identifying each class in a multi-class classification system.

Additionally, Chen designed a channel-wise gate to dynamically estimate the conditional accuracy change due to a given pruning operation [154]. This estimation approach was used to incrementally prune channels during the training process. The dynamic channel pruning strategy developed by Chen facilitates the formation of a stable network

structure during training so that a separate fine-tuning process is not needed after pruning.

6.3.3 Dataflow-based Modeling of DNNs

In this work, we apply dataflow as a methodology for modeling and optimization of DNN applications. For background on dataflow-based methods for signal and information processing (SIP) applications, see [155, 156]. In this context, a dataflow graph (DFG) provides a representation of a SIP application as a directed graph. Vertices in the graph, called *actors*, represent functional modules of arbitrary complexity, and edges represent queues that buffer data as it passes from the output of one actor to the input of another.

To model a DNN with a DFG[157], an important issue is the *granularity* of the dataflow representation — in particular, the complexity of the DNN modules that are represented by individual actors. A common approach is to encapsulate each DNN layer as a distinct actor (e.g., see [114, 142]).

To address the coupling of channels within multi-branch neural networks, Narshana et al. propose the Backwards Graph-based Saliency Computation (BGSC) algorithm [158]. This algorithm jointly considers coupled elements of Distributed Feature Channels (DFCs) to determine their saliencies. The saliencies in turn serve as crucial metrics for determining which parts of the network to prune, leading to modifications in the data flow graphs of the associated subnetworks.

Various methods have been developed that apply dataflow concepts to hardware acceleration of DNNs. For example, a hardware design called sparse periodic systolic (SPS) dataflow prunes DNN models in a way that results in periodic pattern-based sparsity [159].

The approach is shown to lower latency and power dissipation of CNNs on FPGA devices. In the realm of sparse dataflow mapping for various types of DNN operators, such as convolutions and recurrent layers, Kwon et al. have developed a strategy that involves enhancing multipliers and adders with miniature switches and interconnecting them via a novel reconfigurable interconnect capable of supporting arbitrarily-sized neurons [160]. The Efficient Inference Engine (EIE) [161] is an accelerator that performs customized sparse matrix-vector multiplication and handles weight sharing with no loss of efficiency. EIE capitalizes on dynamic input vector sparsity, static weight sparsity, relative indexing, weight sharing, and very narrow weights (4 bits).

A variety of methods have been developed to incorporate dynamic communication patterns between actors into dataflow modeling. Such work is relevant to our objective of supporting adaptive models whose dataflows can be modified dynamically to streamline their operation in relation to run-time changes in an application's operating requirements. For example, Variable Rate Dataflow (VRDF) is a form of dynamic dataflow that facilitates dynamic workload balancing in real-time applications [115]. Methods have been developed for efficient memory management of VRDF-based implementations [115].

VR-PRUNE[162] is dataflow modeling method together with a software framework that incorporates support for dynamic dataflow. The VR-PRUNE framework has been developed with a distinct emphasis on supporting high-performance processing on heterogeneous platforms. A distinguishing characteristic of VR-PRUNE modeling is its enforcement of certain kinds of symmetry in dynamic dataflow behavior. Such symmetry properties are satisfied in a wide variety of useful SIP applications. These properties lead to possibilities for more efficient implementation, and are useful to represent explicitly in

the model when they are satisfied.

In this chapter, we build upon VR-PRUNE modeling concepts to provide network adaptation capabilities based on channel attention. Moreover, we introduce ONNX compatibility in the VR-PRUNE software framework, which enables execution of ONNX DNN models that incorporate our novel methods for adaptive, attention-based network implementation.

6.4 Methods

In our approach to attention-guided model adaptation, we represent the given DNN as a dataflow graph G. Initially, each layer in the DNN is modeled as a distinct actor in G. However, at design time, we transform G by selecting a subset of layers Ω, and replacing the actor $A(L)$ associated with each layer $Lin\Omega$ by a subgraph consisting of layers $\delta(L,1), \delta(L,2), \ldots, \delta(L, lcount(L))$, where $lcount(L)$ gives the number of layers in the decomposition of L, and each layer $\delta(L,i)$ is again encapsulated by a distinct actor, which we denote by $A(\delta(L,i))$.

Intuitively, the transformation process described above can be viewed as a decomposition of each layer in Ω into a set of lower-complexity layers, which we refer to as *sublayers* of the original graph G. We denote the transformed version of G, with all layers in Ω replaced by their corresponding sublayers, as $T(G)$. In our approach to adaptive, attention guided DNN design, we further transform $T(G)$ by integrating dataflow structures that can selectively disable subsets of sublayers throughout $T(G)$ based on dynamically-varying input conditions or operating constraints that are monitored at run-time. This

second phase of transformation results in a new, dynamic dataflow graph denoted $U(G)$, which is the final graph that is to be employed in the targeted deployment.

In the remainder of this section, we elaborate on details of the process of deriving $U(G)$ from G.

In this chapter, we consider only multilayer perceptron (MLP) layers and convolutional layers as candidates for decomposition in the derivation of $T(G)$ from G. However, we envision that the approach can be extended to include other types of layers in the decomposition process. Indeed, extension to other types of layers in an interesting direction for future work.

The methods of this section are demonstrated by applying them to CBAM + ResNet50 as the network backbone (i.e., as the original DNN G). Experimental results from our demonstration using CBAM + ResNet50 are presented in Section 6.5. The methods of this section are general and can be applied to a wide variety of other DNNs; they are not limited to the system that we use in our experiments.

6.4.1 Restrictions on the Decomposition Process

In our design method, the decomposition of each layer in Ω is performed in a channel-wise fashion. As described previously, the decomposition process is restricted to two types of layers — MLP and convolution. We impose the following additional restrictions in the decomposition process.

1. The decomposition is performed such that the sublayers derived from a given layer $L \in \Omega$ are all layers of the same type as L.

2. The decomposition is performed such that $lcount(L)$ is the same for all $L \in \Omega$. That is, there is a positive-integer parameter γ such that $lcount(L) = \gamma$ for all $L \in \Omega$.

3. In the channel-wise decomposition of a given layer $L \in \Omega$, the number of input channels that is assigned to each sublayer is the same for all sublayers of the transformed network $T(G)$. We denote this constant number of channels per sublayer as χ. Thus, for all pairs (L, i), the number of channels in $\delta(L, i)$ is χ.

4. For convolutional layers that are decomposed, the number of output channels that is assigned to each sublayer is also the same for all sublayers of the transformed network $T(G)$. We denote this constant number of output channels per convolutional sublayer as β. Thus, for all convolutional pairs (L, i), the number of output channels in $\delta(L, i)$ is β.

5. For a given layer $L \in \Omega$, the partitioning of channels into sublayers is performed sequentially, based on the way in which the channels are ordered. That is, the first χ channels are grouped into the same sublayer, channels indexed from $(\chi + 1)$ to 2χ (assuming that the indexing starts at 1) are grouped into another sublayer, and so on.

The five restrictions listed above are imposed in the developments of this chapter to help simplify the process of constructing decomposed implementations. However, we assume that all five of the restrictions can be removed to significantly enlarge the design space of the decomposition process. Generalizing the methods of this chapter to work when some or all of the above-listed restrictions are eliminated is an interesting direction for future work.

6.4.2 Decomposing MLP Layers

In the approach developed in this chapter, the parameters Ω, γ, and χ are assumed to be specified by the designer. They could, for example, be derived through a generalized hyperparameter tuning process of the network. More direct methods for optimizing these parameters is another interesting direction for further study.

Suppose we are decomposing a layer J having type MLP. Let $w = (w[0], w[1], \ldots, w[n-1])$ denote the linear weight vector of the MLP, b denote the bias, and n denote the total number of channels ($n = \chi \times \gamma$). Then the computation for layer J can be expressed as

$$MLP(x) = b + \sum_{i=1}^{n} x[i]w[i], \tag{6.1}$$

where $x = (x[1], x[2], \ldots, x[n])$ is the input frame.

In our approach, this computation is decomposed as the sum of γ smaller MLPs, which correspond to the sublayers. For $j = 1, 2, \ldots, \gamma$, the computation for sublayer $\delta(J, j)$ can be expressed, using a minor abuse of notation, as

$$MLP_{J,j}(x) = \frac{b}{\gamma} + \sum_{i=(j-1)\chi+1}^{j\chi} x[i]w[i]. \tag{6.2}$$

In the decomposed form, the MLP for layer J is computed as

$$MLP(x) = \sum_{i=1}^{\gamma} MLP_{J,i}(x). \tag{6.3}$$

An illustration of a decomposed MLP layer with $\gamma = 3$ is given in Fig. 6.1.

As shown in Fig. 6.1, the regular input and MLP layer L are evenly divided into 3

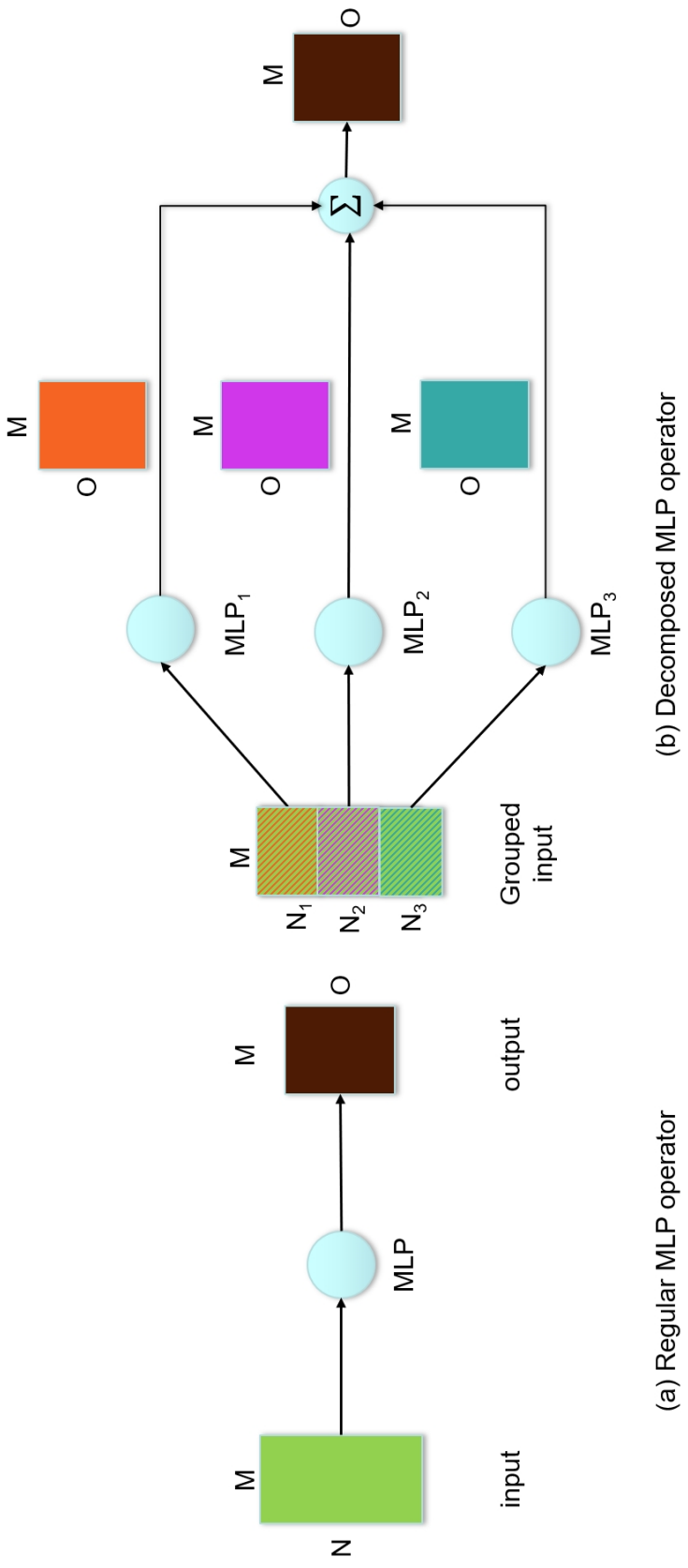


Figure 6.1: Illustration of a decomposed MLP operator.

groups, and the output of L is the summation of the 3 outputs from the groups.

6.4.3 Decomposing Convolutional Layers

The decomposition process for convolutional layers proceeds in a manner similar to the process described in Section 6.4.2 for MLP layers. Consider a convolutional layer M in G, and suppose that M operates on a an input X with dimensions $C \times H \times W$, and produces an output with dimensions $K \times H_{out} \times W_{out}$. Here, C and K give the number of channels in the input and the number of filters in the output, respectively, and H, H_{out}, W and W_{out} give the height and width of the input and output to M.

The original 2D kernel Ker is of size $K \times C \times p \times q)$, and moves over the input data, performing element-wise multiplication. Here, p and q are additional kernel-specific parameters. To decompose the convolution operation, we construct the sublayers from kernels $Ker_1, Ker_2, \ldots, Ker_\gamma$, where each Ker_i is of size $\beta \times \chi \times p \times q)$. A decomposed convolutional layer is illustrated in Fig. 6.2.

As shown in Fig. 6.2, the kernel is separated into 3 components in the channel dimension, where each component encapsulates $C/\gamma = C/3.$ channels. The separated kernels share the same stride and other parameters, but the learned weights are different. The final output of the decomposed layer is a concatenation of the outputs from the three layers instead of the summation.

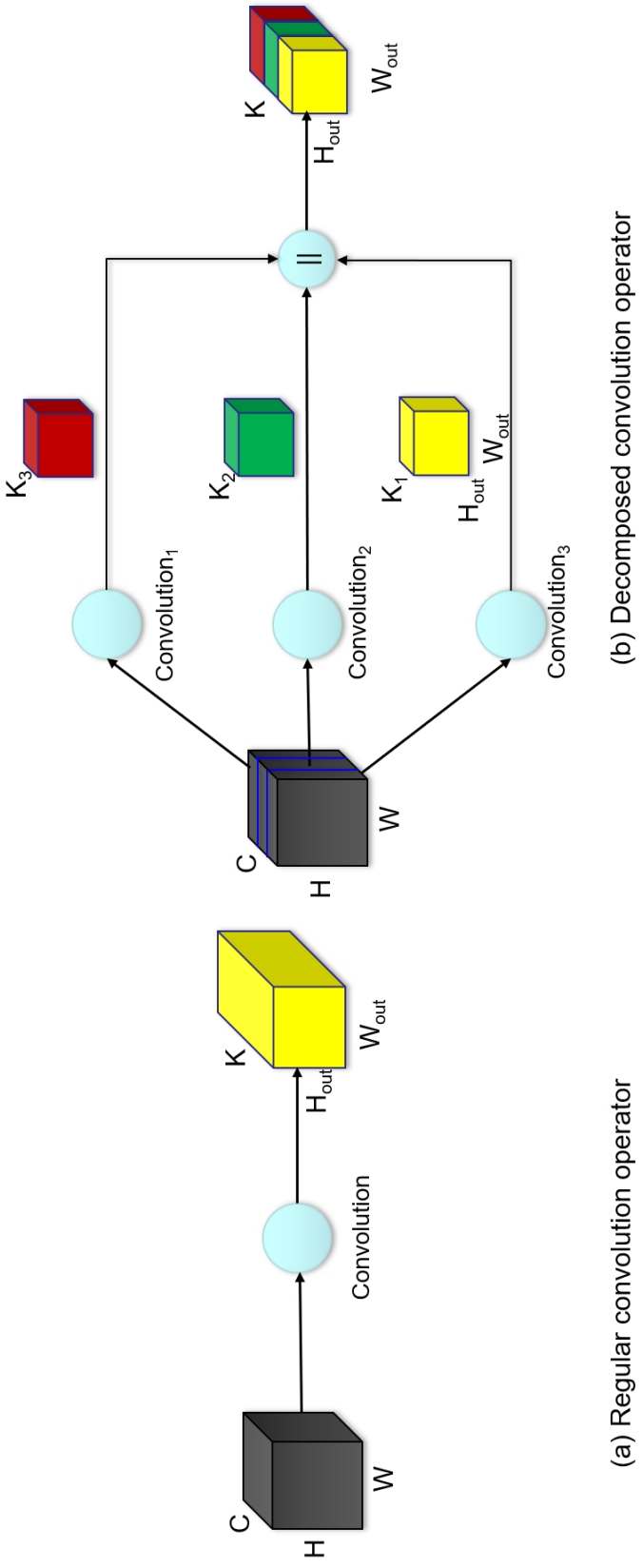


Figure 6.2: Illustration of a decomposed convolutional layer.

6.4.4 Decomposition Algorithm

Algorithm 2 sketches the process by which the given DNN model G is transformed into its decomposed form $T(G)$, and then further transformed by disabling selected sublayers. The selective disabling is performed to trade-off inference accuracy and speed based on guidance from the system designer. The guidance is provided throughout the input τ to the algorithm. The designer can experiment with different values of τ for a given deployment to select a setting that achieves the most favorable trade-off in relation to the constraints and requirements of the deployment. In Algorithm 2, $|\Omega|$ represents the number of elements in the set Ω.

In Algorithm 2, the transformation process is optimized using training and validation datasets D_t and D_v that are supplied as inputs.

The transformed graph U, which is produced by Algorithm 2, can be applied in an adaptive inference system by augmenting it with functionality to dynamically monitor attention scores and adjust the set of disabled sublayers based on recently-measured attention values. In such a deployment, U can be viewed as the initial model that is deployed, and forms the baseline for subsequent adaptations. In Section 6.4.5, we discuss further techniques that can be used to employ our proposed network decomposition technique in adaptive deployments.

6.4.5 Dataflow Mapping

DNN models that are decomposed using the methods described in this section can be represented naturally using dataflow concepts. Such dataflow representations are useful for

deriving efficient implementations of the decomposed models and enabling the dynamic disabling of selected sublayers in the models. The dynamic selection of disabled sublayers can be used to systematically trade off interference speed and accuracy based on time-varying operational requirements or resource availability.

To incorporate dataflow modeling with the decomposition methods developed in this chapter, we apply the VR-PRUNE model of computation [162]. In VR-PRUNE, as in other forms of dataflow, a signal and information processing application is modeled as a directed graph $\Gamma = (B, E)$, where the set of vertices B represents the set of functional modules (actors) in the application, and the edges in E represent first-in, first-out (FIFO) communication channels between actors.

Ports in dataflow graphs refer to interfaces between actors and their incident edges. Ports in VR-PRUNE can be categorized into three different types: each port is either a (input or output) *control port*, *static regular port* (*SRP*) or *dynamic regular port* (*DRP*). An SRP ρ_s has a fixed token consumption/production rate $atr(\rho_s)$, which is set at design time. The token rate associated with a dataflow port ρ and a given execution f of the enclosing actor α is : (1) the number of data values consumed by α through ρ during f if ρ is an input port or (2) the number of data values produced by α through ρ during f if ρ is an output port.

A DRP ρ_d in VR-PRUNE has a variable token rate. As a design rule of VR-PRUNE, this token rate must satisfy the following constraint

$$lrl(\rho_d) \leq atr(\rho_d) \leq url(\rho_d), \tag{6.4}$$

where the terms lrl, atr, and url stand for lower rate limit, active token rate, and upper rate limit, respectively; these three quantities are always non-negative-integer valued. The values lrl and url are values that are fixed at design time, whereas atr may vary at run-time within the constraint imposed by Equation 6.4.

A control port ρ_c in VR-PRUNE must have a fixed token rate of 1.

VR-PRUNE actors can be classified into three types: static processing actors (SPAs), dynamic actors (DAs), and dynamic processing actors (DPAs). The ports of an SPA can only be of the type SRP, and therefore an SPA operates as a synchronous dataflow (SDF) [107] actor.

The dynamic attributes of VR-PRUNE manifest in the DAs and DPAs of a VR-PRUNE graph. Unlike SPAs, DAs and DPAs have DRPs. In the dataflow-based DNN design approach developed in this chapter, DAs and DPAs are employed to provide dynamic enabling and disabling of sublayers based on dynamically-computed attention scores.

Fig. 6.3 provides an illustration of a DPA. The control input port sets the token rates of a pair of DRPs (one on the input side, and the other on the output side). When the actor executes, the value of a single token coming from the control port c sets the token rates of both DRPs. In our decomposed-DNN modeling approach, we apply DPAs to decompose layers that may be dynamically reconfigured (by changing the set of sublayers that is active), and we use SPAs to describe the other functional modules in the decomposed system. The set of dynamically-enabled sublayers that is active at any given time is determined by configuration actors, which can receive input from dynamically-determined attention scores.

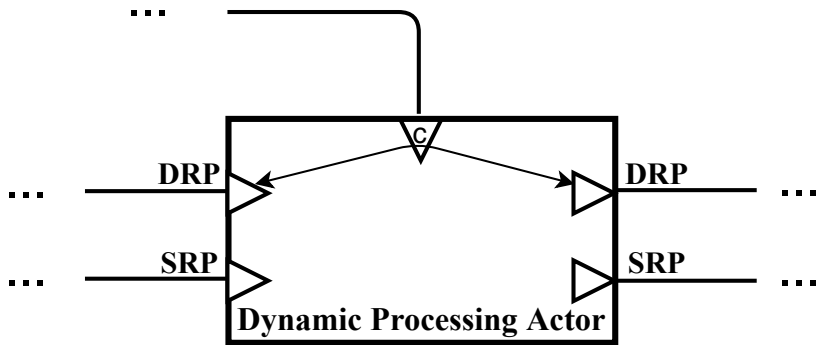


Figure 6.3: An illustrative example of a dynamic processing actor.

To enable the importing of DNN models that have been written in TensorFlow[163] or PyTorch[164] to VR-PRUNE, we have extended the VR-PRUNE runtime framework (VPRF) with ONNX[144] support. In particular, our new version of the VPRF framework includes a VR-PRUNE graph generator built on the `libonnx` project [165]. The VR-PRUNE graph generator parses an ONNX model specification and automatically generates a VR-PRUNE graph from the DNN model derived from the parsing results. The VPRF application designer can specify — as input to the VR-PRUNE graph generator — which sections of the graph are to be dynamically reconfigurable. As part of its synthesis process, the VR-PRUNE graph generator inserts dynamic actors and control actors to achieve the specified functionality for dynamic reconfiguration.

Fig. 6.4 illustrates the interplay between a decomposed layer and the configuration actor that is responsible for enabling and disabling the associated sublayers. This example highlights the role of configuration actors in determining the token rates for DRPs, thereby controlling the dynamic enabling/disabling of selected sublayers, which is central to the adaptability of our transformed network architecture.

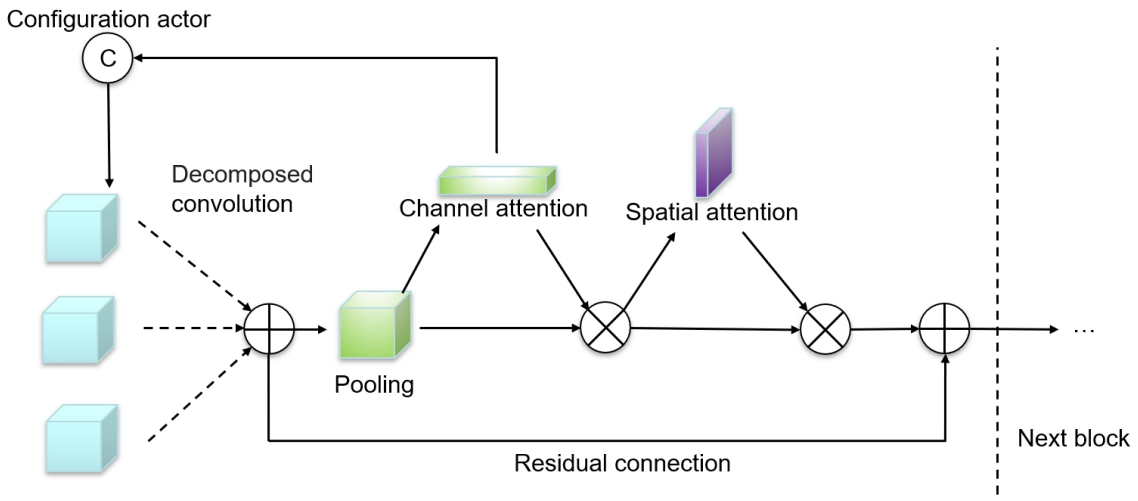


Figure 6.4: An illustration of how the dynamic control of sublayers is implemented using VR-PRUNE.

6.5 Experimental results

Our experiments are conducted using a VR-PRUNE-based prototype implementation of the ResNet50 + CBAM model [147] with decomposition into sublayers and attention-guided sublayer disabling as developed in Section 6.4. We use $lcount = 4$ for the decomposition process. We conduct experiments on the ImageNet-1K dataset [166]. Processing includes a resize of 256×256 followed by single-crop evaluation with a size of 224×224 at validation/test time. We use the original ResNet50 + CBAM model as a baseline model in our experiments, and evaluate the performance of decomposed and partially-disabled models compared to this baseline. The processing platforms employed in our experiments are summarized in Table 6.1.

As additional baseline configurations, we consider the disabling of entire blocks (among a total of 4) of the ResNet-50 backbone model. We consider instances of this baseline disabling method involving 1, 2 and 3 disabled blocks, respectively. Before

evaluating each of these baseline-disabling instances, we fine-tune the modified model on the training data for 50 additional epochs. This baseline approach to disabling and network reconfiguration is selected as a straightforward alternative for manipulating network structure to trade off the accuracy and computational requirements of inference.

Table 6.2 shows experimental results involving the baseline in which entire blocks in ResNet-50 are disabled. The abbreviations `@1` and `@5` stand for rank-1 and rank-5 accuracy, respectively. Table 6.3 shows the performance of the proposed decomposition and selective-disabling approach with different numbers of disabled sublayers per decomposed layer L_i. In both tables, the original ResNet-50 + CBAM model is represented in the first row of data. From the data in the two tables, we see that disabling sublayers provides significant improvements in execution time, but not as much improvement as disabling entire blocks. However, the loss in accuracy for disabling sublayers is significantly less than the accuracy that is given up when disabling blocks. Thus, we can see that the proposed decomposition method leads to novel operational trade-offs compared to the block-based disabling baseline.

Limitations. Although we have shown the potential of VR-PRUNE to dynamically configure machine learning models and to save on computations, the implementation of fine-grained channel pruning requires further investigation. The main reasons for this are: (1) The limitation of the backbone model itself; for ResNet50 + CBAM, there are altogether 117 convolution operators in the graph, and the cumulative execution time of these is 34% of the total. Thus, the overall computational efficiency provided by disabling sublayers is limited to impact at most one-third of the overall computation. (2) The decomposition process generates extra actors for a dataflow model. For example, in our prototype

Table 6.1: **Platforms used for experiments.**

Tag	CPU	GPU	Operating System
Plat1	Intel Core i7-12700H 14-Core Processor: 4.70 GHz	NVIDIA RTX A2000	Ubuntu 22.04 Linux
Plat2	AMD Ryzen 5 3600 6-Core Processor: 3.60 GHz	NVIDIA RTX 2060 Super	Ubuntu 18.04 Linux
RB5	Qualcomm Kryo 585	Qualcomm Adreno 650	Ubuntu 18.04 Linux

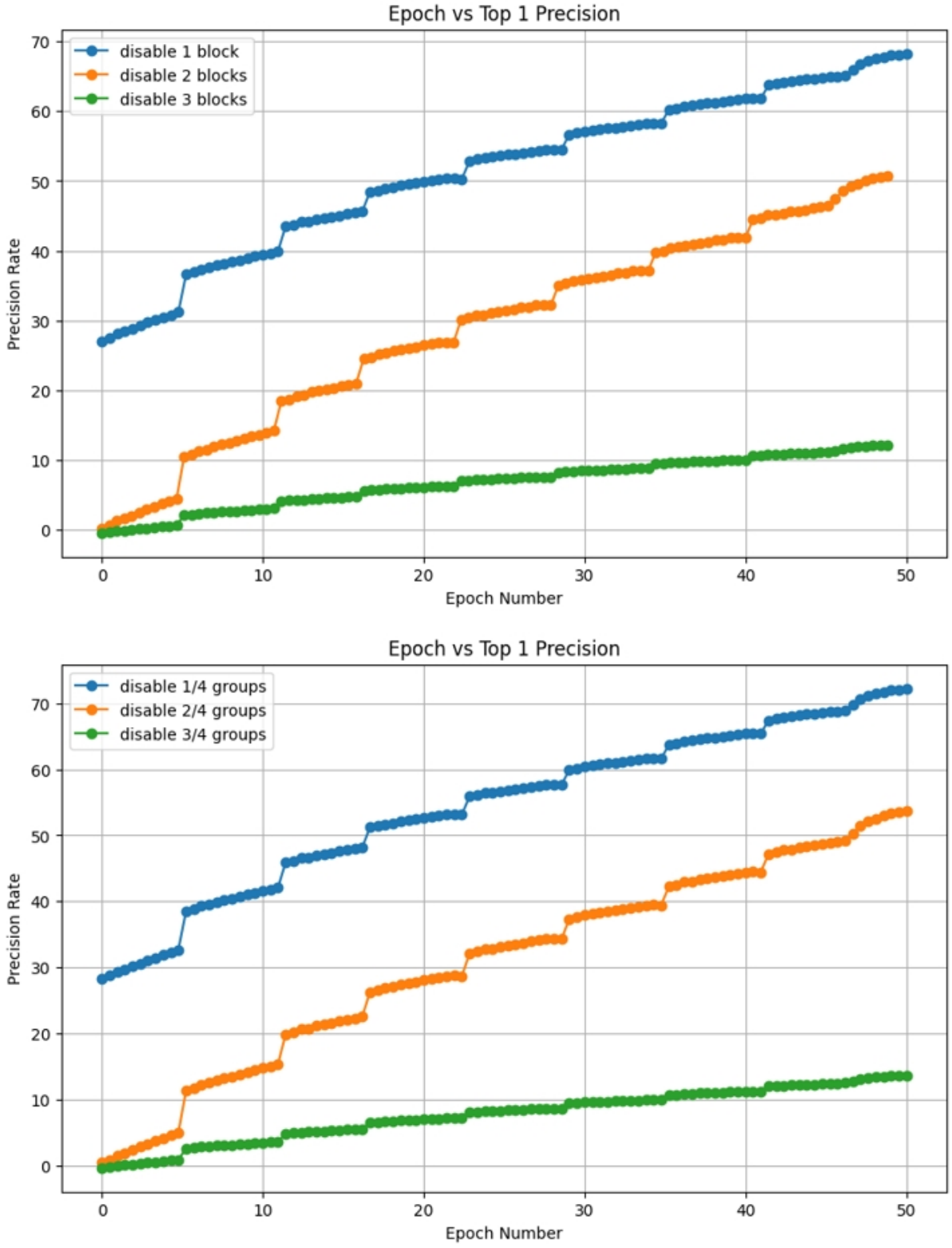


Figure 6.5: Top1 performance of partially disabled models after 50 epochs of fine tuning.

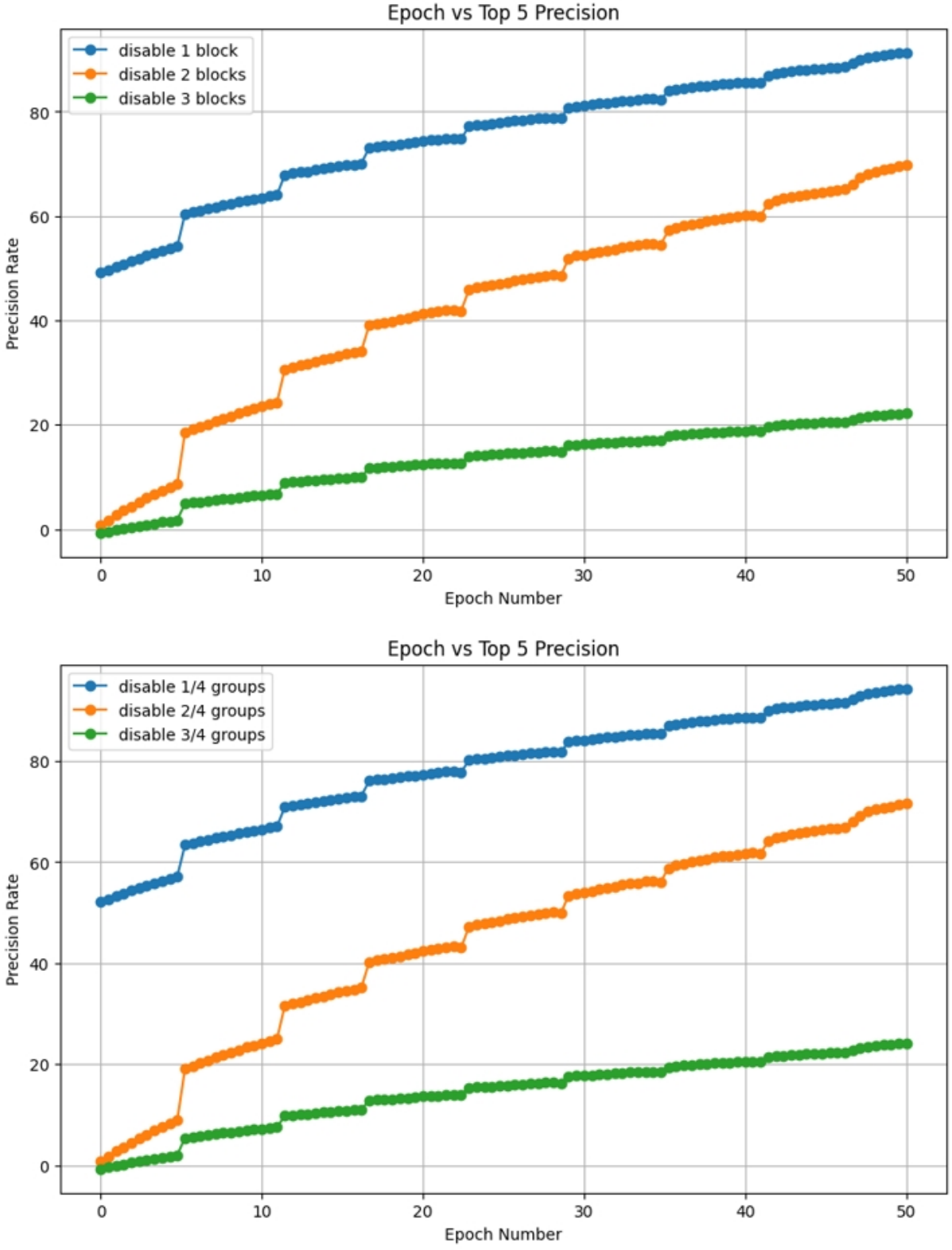


Figure 6.6: Top5 performance of partially disabled models after 50 epochs of fine tuning.

Table 6.2: Pruning of ResNet50 + CBAM at the block level.

	Plat1	Plat2	RB5	Accuracy
Res50 + CBAM	1.49s	1.72s	3.07s	@1 87.745 @5 97.672
Disable 1 block	1.32s	1.57s	2.68s	@1 69.476 @5 86.948
Disable 2 blocks	1.01s	1.25s	2.16s	@1 51.698 @5 70.716
Disable 3 blocks	0.70s	0.98s	1.73s	@1 12.246 @5 21.972

Table 6.3: Channel pruning.

	Plat1	Plat2	RB5	Accuracy
Res50 + CBAM	1.42s	1.72s	3.07s	@1 87.745 @5 97.672
Disable 1 sublayer	1.31s	1.67s	2.95s	@1 72.080 @5 91.907
Disable 2 sublayers	1.23s	1.58s	2.73s	@1 55.185 @5 72.627
Disable 3 sublayers	1.14s	1.52s	2.55s	@1 13.684 @5 23.284

implementation, it generates 96 additional Constant actors, 48 Convolution actors, and 4 additional Concat nodes, where Concat actors are implemented in ONNX [144] as inefficient memory-to-memory-copies.

6.6 Conclusion

This chapter has developed new methods for pruning deep neural networks (DNNs) to trade-off some amount of accuracy loss for reduced computational requirements, and also to enable dynamic control of accuracy/computational-requirements trade-offs in response to time-varying operational constraints and conditions. The proposed methods are based on decomposing selected DNN layers into collections of lower-complexity layers (sublayers), and applying attention techniques to select subsets of sublayers to disable at design time or runtime. The VR-PRUNE framework for dataflow modeling is applied and extended to prototype the proposed methods and provide structured methods to achieve runtime sublayer-disabling functionality efficiently and reliably. The chapter has described several

interesting directions for future work in terms of generalizing the proposed methods. We envision that these generalizations have the potential to significantly enrich the design space of model decomposition, and enhance the novel trade-offs that are provided by decomposed implementations.

Algorithm 2 Proposed network decomposition algorithm.

Input:

G: dataflow graph model of the original DNN.
$\Omega = \{L_1, L_2, \ldots, L_p\}$, where $p = |\Omega|$.
$lcount$: The number of sublayers in each decomposed layer.
D_v: The validation dataset.
D_t: The training dataset.
τ: the number of channels to be disabled.

Output:

The decomposed graph with selected groups disabled.

Initialization: Convert $L_1, L_2, \ldots, L_p$ in G to decomposed layers to form $T(G)$.
$U \leftarrow T(G)$
$i = 1$
Exercise $T(G)$ on D_v and record the resulting attention scores $\{w(c)\}$ for all channels c in all sublayers within $T(G)$.
while $i \leq p$ **do**
    $j = 1$
    while $j \leq lcount$ **do**
        Set the sublayer attention score $W_{i,j}$ to be the average of $w(c)$ over all output channels c within sublayer $\delta(L_i, j)$.
        $j \leftarrow j + 1$
    end while
    $i \leftarrow i + 1$
end while
Rank the values $\{W_{i,j}\}$ into a sorted list V.
Set ω to be the τth smallest value in V.
$m = 1$
while $m \leq p$ **do**
    $n = 1$
    while $n \leq lcount$ **do**
        if $W_{m,n} \leq \omega$ **then**
            Disable the sublayer $\delta(L_m, n)$ in U and modify the connections of operators in U accordingly.
        end if
        $n \leftarrow n + 1$
    end while
    $m \leftarrow m + 1$
end while
Fine tune U using D_t.
Return U as the decomposed, partially-disabled graph.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis has made significant contributions to the field of machine learning and deep learning for criminology and criminal justice (CCJ) and embedded computer vision applications. The contributions have included aspects pertaining to the interpretability, fairness, and efficiency of machine learning models for these domains.

The development of the Evolutionary Algorithm Decision Tree for Crime prediction (EADTC) algorithm represents a novel approach to enhancing the effectiveness of single-decision-tree models without compromising accuracy compared to less interpretable types of machine learning models. EADTC offers a robust alternative that avoids heavy reliance on domain-specific feature engineering and simplifies hyper-parameter tuning and data processing.

The combination of EADTC and multi-objective optimization, as represented by the Evolutionary algorithm for multi-objective EQuity and Recidivism-Prediction optimization (E2QRP), provides flexible and competitive performance on both accuracy and fairness metrics, and allows prediction system designers to strategically select operating points (trade-offs between accuracy and fairness) that best match the requirements of the targeted applications.

The integrated application of attention mechanisms and convolutional long short-

term memory (ConvLSTM) models marks a significant step forward in retaining temporal information while capturing global spatial dependencies in crime prediction applications. We have demonstrated that this method not only yields robust performance across various relevant datasets but also enhances model interpretability through intuitive visualization of spatial contribution patterns. Such advances facilitate iterative model improvement and offer promising directions for leveraging knowledge about crime patterns and geographic characteristics in optimizing prediction models.

The introduction of the Variable Rate PSM Runtime Environment (VR-PRUNE), which includes a novel model of computation and embedded software framework, provides enhanced flexibility and efficiency in signal and information processing across heterogeneous platforms. This graph-based design framework provides reliable and efficient methods for dynamic reconfiguration of trade-offs between accuracy and computational requirements in deep neural networks (DNNs), as well as in other important types of subsystems in signal and information processing applications.

Our integrated methods for applying channel attention and dataflow modeling to control the adaptation of DNN models provides optimized performance subject to dynamic changes in operational conditions and constraints. Our developed framework for dataflow- and attention-based design optimization provides important advantages including efficient adaptation to optimize operational trade-offs, such as those between energy efficiency and knowledge extraction accuracy, and integration of variable-rate dataflow modeling with the Open Neural Network Exchange (ONNX) [144], which is a widely-used open-source tool for representing and exchanging deep learning models across different machine learning frameworks and platforms.

7.2 Future Work

Moving forward, this thesis opens up multiple pathways for future research. For the EADTC framework, advancing the formalization of model interpretability and further enhancing the transparency of machine learning models are crucial steps. Another interesting direction is investigating the adaptation of EADTC to other relevant application domains, and the study of ideas underlying EADTC as a foundation for general evolutionary algorithm techniques for decision tree optimization.

Interesting directions for further work that builds upon the E2QRP framework also include exploring the methods in a more general context. We envision that such further work would include experimenting with other application areas to understand which aspects of E2QRP work well as a foundation for general multi-objective optimization, and which need to be left adaptable so that they can be customized as needed.

In the area of attention mechanisms and spatio-temporal forecasting, exploring the integration of attention maps with domain-specific knowledge to refine model optimization and interpretability represents a promising direction for future work. Contributions in this direction could unlock new potential for enhancing performance and user trust in practical machine learning applications.

Building on our contributions involving VR-PRUNE and model pruning techniques, future work could focus on generalizing the proposed methods to broaden the design space of model decomposition. Also, investigating structured approaches to enable runtime sublayer-disabling functionality more efficiently will be useful in providing further enhancements to run-time adaptation in embedded model-based inferencing systems and

other types of signal and information processing systems.

Bibliography

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al., "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[2] Haifeng Zheng, Feng Lin, Xinxin Feng, and Youjia Chen, "A hybrid deep learning model with attention-based conv-lstm networks for short-term traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 6910–6920, 2020.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al., "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[4] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang, "Xai—explainable artificial intelligence," *Science robotics*, vol. 4, no. 37, pp. eaay7120, 2019.

[5] Wei-Yin Loh, "Classification and regression trees," *Wiley interdisciplinary reviews: data mining and knowledge discovery*, vol. 1, no. 1, pp. 14–23, 2011.

[6] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant, *Applied logistic regression*, John Wiley & Sons, 2013.

[7] Jerome H Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[8] Hila Chefer, Shir Gur, and Lior Wolf, "Transformer interpretability beyond attention visualization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 782–791.

[9] Christoph Molnar, *Interpretable machine learning*, Lulu. com, 2020.

[10] Benjamin Timon, "Non-profiled deep learning-based side-channel attacks with sensitivity analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 107–131, 2019.

[11] Yujunrong Ma, Kiminori Nakamura, Eung-Joo Lee, and Shuvra S Bhattacharyya, "Eadtc: An approach to interpretable and accurate crime prediction," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2022, pp. 170–177.

[12] Zhongzhi Xu, Jian Zhang, Qingpeng Zhang, and Paul Siu Fai Yip, "Explainable learning for disease risk prediction based on comorbidity networks," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2019, pp. 814–818.

[13] Andrew L Beam and Isaac S Kohane, "Big data and machine learning in health care," *Jama*, vol. 319, no. 13, pp. 1317–1318, 2018.

[14] Richard Berk, Drougas Berk, and Drougas, *Machine learning risk assessments in criminal justice settings*, Springer, 2019.

[15] Jon Kleinberg, Himabindu Lakkaraju, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan, "Human decisions and machine predictions," *The quarterly journal of economics*, vol. 133, no. 1, pp. 237–293, 2018.

[16] Cade Metz and Adam Satariano, "An algorithm that grants freedom, or takes it away," *The New York Times*, vol. 6, 2020.

[17] Cynthia Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[18] Lei Jimmy Ba and Rich Caruana, "Do deep nets really need to be deep?," *arXiv preprint arXiv:1312.6184*, 2013.

[19] Zhengping Che, Sanjay Purushotham, Robinder Khemani, and Yan Liu, "Distilling knowledge from deep networks with applications to healthcare domain," *arXiv preprint arXiv:1512.03542*, 2015.

[20] Akshay Sujatha Ravindran, Manuel Cestari, Christopher Malaya, Isaac John, Gerard E Francisco, Charles Layne, and José L Contreras Vidal, "Interpretable deep learning models for single trial prediction of balance loss," in *2020 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2020, pp. 268–273.

[21] Berk Ustun, Stefano Traca, and Cynthia Rudin, "Supersparse linear integer models for interpretable classification," *arXiv preprint arXiv:1306.6677*, 2013.

[22] Jiaming Zeng, Berk Ustun, and Cynthia Rudin, "Interpretable classification models for recidivism prediction," *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, vol. 180, no. 3, pp. 689–722, 2017.

[23] Leo Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[24] Richard Berk and Justin Bleich, "Forecasts of violence to inform sentencing decisions," *Journal of Quantitative Criminology*, vol. 30, no. 1, pp. 79–96, 2014.

[25] T. Back, U. Hammel, and H-P Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 3–17, 1997.

[26] Supriyo Chakraborty, Richard Tomsett, Ramya Raghavendra, Daniel Harborne, Moustafa Alzantot, Federico Cerutti, Mani Srivastava, Alun Preece, Simon Julier, Raghuveer M Rao, et al., "Interpretability of deep learning models: A survey

of results," in *2017 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, Internet of people and smart city innovation (smartworld/SCALCOM/UIC/ATC/CBDcom/IOP/SCI)*. IEEE, 2017, pp. 1–6.

[27] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone, *Classification and regression trees*, Routledge, 2017.

[28] Richard A Berk, Susan B Sorenson, and Geoffrey Barnes, "Forecasting domestic violence: A machine learning approach to help inform arraignment decisions," *Journal of empirical legal studies*, vol. 13, no. 1, pp. 94–115, 2016.

[29] Abdarahmane Traoré and Moulay A Akhloufi, "Violence detection in videos using deep recurrent and convolutional neural networks," in *2020 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2020, pp. 154–159.

[30] Xiaolan Wu and Tony H Grubesic, "Identifying irregularly shaped crime hot-spots using a multiobjective evolutionary algorithm," *Journal of geographical systems*, vol. 12, no. 4, pp. 409–433, 2010.

[31] Peter J Angeline, Gregory M Saunders, and Jordan B Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE transactions on Neural Networks*, vol. 5, no. 1, pp. 54–65, 1994.

[32] Tzyy-Chyang Lu, "Cnn convolutional layer optimisation based on quantum evolutionary algorithm," *Connection Science*, pp. 1–13, 2020.

[33] Zhyar Rzgar K Rostam and Sozan Abdullah Mahmood, "Classification of brainwave signals based on hybrid deep learning and an evolutionary algorithm," *arXiv preprint arXiv:1912.07361*, 2019.

[34] Behzad Soleimani Neysiani, Nasim Soltani, Reza Mofidi, and Mohammad Hossein Nadimi-Shahraki, "Improve performance of association rule-based collaborative filtering recommendation systems using genetic algorithm," *International Journal of Information Technology and Computer Science*, vol. 11, no. 2, pp. 48–55, 2019.

[35] Qiangfu Zhao and Mitsuyoshi Shirasaka, "A study on evolutionary design of binary decision trees," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, 1999, vol. 3, pp. 1988–1993.

[36] Dariusz Jankowski and Konrad Jackowski, "Evolutionary algorithm for decision tree induction," in *IFIP International Conference on Computer Information Systems and Industrial Management*, 2015, pp. 23–32.

[37] Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq, "Algorithmic decision making and the cost of fairness," in *Proceedings of the 23rd acm sigkdd international conference on know·ledge discovery and data mining*, 2017, pp. 797–806.

[38] Alexandra Chouldechova, "Fair prediction with disparate impact: A study of bias in recidivism prediction instruments," *Big data*, vol. 5, no. 2, pp. 153–163, 2017.

[39] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan, "Inherent trade-offs in the fair determination of risk scores," *arXiv preprint arXiv:1609.05807*, 2016.

[40] Patrick A Langan and David Joshua Levin, *Recidivism of prisoners released in 1994*, US Department of Justice, Office of Justice Programs, Bureau of Justice ..., 2002.

[41] Saskia Le Cessie and Johannes C Van Houwelingen, "Ridge estimators in logistic regression," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 41, no. 1, pp. 191–201, 1992.

[42] Andreas Holzinger, Bernd Malle, Peter Kieseberg, and Peter Roth, "Towards the augmented pathologist: Challenges of explainable-ai in digital pathology," *arXiv preprint arXiv:1712.06657*, 2017.

[43] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad, "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1721–1730.

[44] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al., "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020.

[45] W James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu, "Definitions, methods, and applications in interpretable machine learning," *Proceedings of the National Academy of Sciences*, vol. 116, no. 44, pp. 22071–22080, 2019.

[46] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner, "Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks," *ProPublica*, May 2016.

[47] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian, "Certifying and removing disparate impact," *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 259–268, 2015.

[48] Danielle Ensign, Sorelle A Friedler, Scott Neville, Carlos Scheidegger, and Suresh Venkatasubramanian, "Runaway feedback loops in predictive policing," in *Conference on fairness, accountability and transparency*. PMLR, 2018, pp. 160–171.

[49] Richard Berk, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth, "Fairness in criminal justice risk assessments: The state of the art," *Sociological Methods & Research*, vol. 50, no. 1, pp. 3–44, 2021.

[50] Y. Jin and Bernhard Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 38, no. 3, pp. 397–415, 2008.

[51] Sam Corbett-Davies and Sharad Goel, "The measure and mismeasure of fairness: A critical review of fair machine learning," *arXiv preprint arXiv:1808.00023*, 2018.

[52] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan, "Algorithmic fairness," in *ACM conference on Economics and Computation*. ACM, 2018.

[53] Solon Barocas and Andrew D Selbst, "Big data's disparate impact," *California Law Review*, vol. 104, pp. 671, 2016.

[54] Julia Dressel and Hany Farid, "The accuracy, fairness, and limits of predicting recidivism," *Science advances*, vol. 4, no. 1, pp. eaao5580, 2018.

[55] Andrew D Selbst, Danah Boyd, Sorelle A Friedler, Suresh Venkatasubramanian, and Janet Vertesi, "Fairness and abstraction in sociotechnical systems," in *Proceedings of the conference on fairness, accountability, and transparency*, 2019, pp. 59–68.

[56] Moritz Hardt, Eric Price, and Nati Srebro, "Equality of opportunity in supervised learning," *Advances in neural information processing systems*, vol. 29, 2016.

[57] Zachary C Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.," *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[58] Mattia Carletti, Chiara Masiero, Alessandro Beghi, and Gian Antonio Susto, "Explainable machine learning in industry 4.0: Evaluating feature importance in anomaly detection to enable root cause analysis," in *2019 IEEE international conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2019, pp. 21–26.

[59] R Timothy Marler and Jasbir S Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[60] C. A. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Springer, 2007.

[61] Eckart Zitzler and Lothar Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[62] Y. Hua, Q. Liu, K. Hao, and Y. Jin, "A survey of evolutionary algorithms for multi-objective optimization problems with irregular pareto fronts," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 2, pp. 303–318, 2021.

[63] Ricardo Guedes, Vasco Furtado, and Tarcísio Pequeno, "Multi-objective evolutionary algorithms and multiagent models for optimizing police dispatch," in *2015 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2015, pp. 37–42.

[64] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[65] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner Gardner, Marc Parizeau, and Christian Gagné, "Deap: Evolutionary algorithms made easy," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2171–2175, 2012.

[66] Stephan Dreiseitl and Lucila Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," *Journal of biomedical informatics*, vol. 35, no. 5-6, pp. 352–359, 2002.

[67] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen, *Classification and Regression Trees*, Wadsworth, 1984.

[68] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[69] J.H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[70] C. Segura, C. A. C. Coello, G. Miranda, and C. León, "Using multi-objective evolutionary algorithms for single-objective constrained and unconstrained optimization," *Annals of Operations Research*, vol. 240, pp. 217–250, 2016.

[71] Yujunrong Ma, Xiaowen Qi, Kiminori Nakamura, and Shuvra S Bhattacharyya, "Towards interpretable, attention-based crime forecasting," in *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2023, pp. 2366–2372.

[72] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi, "Gman: A graph multi-attention network for traffic prediction," in *Proceedings of the AAAI conference on artificial intelligence*, 2020, vol. 34, pp. 1234–1241.

[73] Manuel Muñoz Sánchez, Emilia Silvas, Denis Pogosov, and Decebal Constantin Mocanu, "A hybrid framework combining vehicle system knowledge with machine learning methods for improved highway trajectory prediction," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2020, pp. 444–450.

[74] Aniekan E Essien, Ilias Petrounias, Pedro Sampaio, and Sandra Sampaio, "Deeppresimm: integrating deep learning with microsimulation for traffic prediction," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2019, pp. 4257–4262.

[75] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Advances in neural information processing systems*, vol. 28, 2015.

[76] Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann, "Video transformer network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3163–3172.

[77] Albert Meijer and Martijn Wessels, "Predictive policing: Review of benefits and drawbacks," *International Journal of Public Administration*, vol. 42, no. 12, pp. 1031–1039, 2019.

[78] George O Mohler, Martin B Short, Sean Malinowski, Mark Johnson, George E Tita, Andrea L Bertozzi, and P Jeffrey Brantingham, "Randomized controlled field trials of predictive policing," *Journal of the American statistical association*, vol. 110, no. 512, pp. 1399–1411, 2015.

[79] Bruno Lepri, Nuria Oliver, Emmanuel Letouzé, Alex Pentland, and Patrick Vinck, "Fair, transparent, and accountable algorithmic decision-making processes: The premise, the proposed solutions, and the open challenges," *Philosophy & Technology*, vol. 31, pp. 611–627, 2018.

[80] Muhammad Aurangzeb Ahmad, Carly Eckert, and Ankur Teredesai, "Interpretable machine learning in healthcare," in *Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics*, 2018, pp. 559–560.

[81] In Lee and Yong Jae Shin, "Machine learning for enterprises: Applications, algorithm selection, and challenges," *Business Horizons*, vol. 63, no. 2, pp. 157–170, 2020.

[82] Ashley Deeks, "The judicial demand for explainable artificial intelligence," *Columbia Law Review*, vol. 119, no. 7, pp. 1829–1850, 2019.

[83] Amina Adadi and Mohammed Berrada, "Peeking inside the black-box: a survey on explainable artificial intelligence (xai)," *IEEE access*, vol. 6, pp. 52138–52160, 2018.

[84] Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey E Hinton, "Neural additive models: Interpretable machine learning with neural nets," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4699–4711, 2021.

[85] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[86] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov, "Unsupervised learning of video representations using lstms," in *International conference on machine learning*. PMLR, 2015, pp. 843–852.

[87] Yiannis Kamarianakis and Poulicos Prastacos, "Forecasting traffic flow conditions in an urban network: Comparison of multivariate and univariate approaches," *Transportation Research Record*, vol. 1857, no. 1, pp. 74–84, 2003.

[88] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back, "Face recognition: A convolutional neural-network approach," *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997.

[89] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[90] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2012.

[91] Mohammad Hesam Hesamian, Wenjing Jia, Xiangjian He, and Paul Kennedy, "Deep learning techniques for medical image segmentation: achievements and challenges," *Journal of digital imaging*, vol. 32, pp. 582–596, 2019.

[92] Larry R Medsker and LC Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, pp. 64–67, 2001.

[93] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[94] Apeksha Shewalkar, Deepika Nyavanandi, and Simone A Ludwig, "Performance evaluation of deep neural networks applied to speech recognition: Rnn, lstm and gru," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 9, no. 4, pp. 235–245, 2019.

[95] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[96] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[97] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu, "Are transformers effective for time series forecasting?," *arXiv preprint arXiv:2205.13504*, 2022.

[98] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena, "Self-attention generative adversarial networks," in *International conference on machine learning*. PMLR, 2019, pp. 7354–7363.

[99] Zhihui Lin, Maomao Li, Zhuobin Zheng, Yangyang Cheng, and Chun Yuan, "Self-attention convlstm for spatiotemporal prediction," in *Proceedings of the AAAI conference on artificial intelligence*, 2020, vol. 34, pp. 11531–11538.

[100] Samira Abnar and Willem Zuidema, "Quantifying attention flow in transformers," *arXiv preprint arXiv:2005.00928*, 2020.

[101] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[102] "Crime stats," 2023, Accessed on April 17, 2023.

[103] Dylan J Fitzpatrick, Wilpen L Gorr, and Daniel B Neill, "Policing chronic and temporary hot spots of violent crime: A controlled field experiment," *arXiv preprint arXiv:2011.06019*, 2020.

[104] Yujunrong Ma, Jiahao Wu, Shuvra S Bhattacharyya, and Jani Boutellier, "Decidable variable-rate dataflow for heterogeneous signal processing systems," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 1683–1687.

[105] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[106] George F Zaki, William Plishker, Shuvra S Bhattacharyya, Charles Clancy, and John Kuykendall, "Integration of dataflow-based heterogeneous multiprocessor scheduling techniques in gnu radio," *Journal of Signal Processing Systems*, vol. 70, no. 2, pp. 177–191, 2013.

[107] Edward Ashford Lee and David G Messerschmitt, "Static scheduling of synchronous data flow programs for digital signal processing," *IEEE Transactions on Computers*, vol. 100, no. 1, pp. 24–35, 1987.

[108] Edward A Lee and Stephen Neuendorffer, "Concurrent models of computation for embedded software," *IEE Proceedings-Computers and Digital Techniques*, vol. 152, no. 2, pp. 239–250, 2005.

[109] Sebastian Ritz, Matthias Pankert, V Zivojinovic, and Heinrich Meyr, "Optimum vectorization of scalable synchronous dataflow graphs," in *Proceedings of International Conference on Application Specific Array Processors*. IEEE, 1993, pp. 285–296.

[110] Bishnupriya Bhattacharya and Shuvra S Bhattacharyya, "Quasi-static scheduling of reconfigurable dataflow graphs for DSP systems," in *Proceedings of International Workshop on Rapid System Prototyping.* IEEE, 2000, pp. 84–89.

[111] Joseph Buck, Soonhoi Ha, Edward A. Lee, and David G. Messerschmitt, *Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems*, pp. 527–543, Kluwer Academic Publishers, USA, 2001.

[112] Joseph Tobin Buck and Edward A Lee, "Scheduling dynamic dataflow graphs with bounded memory using the token flow model," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1993, vol. 1, pp. 429–432.

[113] GR Gao, R Govindarajan, and Prakash Panangaden, "Well-behaved dataflow programs for DSP computation," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1992, vol. 5, pp. 561–564.

[114] Jani Boutellier, Jiahao Wu, Heikki Huttunen, and Shuvra S Bhattacharyya, "PRUNE: Dynamic and decidable dataflow for signal processing on heterogeneous platforms," *IEEE Transactions on Signal Processing*, vol. 66, no. 3, pp. 654–665, 2017.

[115] Maarten H Wiggers, Marco JG Bekooij, and Gerard JM Smit, "Buffer capacity computation for throughput constrained streaming applications with data-dependent inter-task communication," in *IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2008, pp. 183–194.

[116] Jani Boutellier and Shuvra S Bhattacharyya, "Low-power heterogeneous computing via adaptive execution of dataflow actors," in *IEEE International Workshop on Signal Processing Systems (SiPS)*. IEEE, 2017, pp. 1–6.

[117] James L Peterson, "Petri nets," *ACM Computing Surveys (CSUR)*, vol. 9, no. 3, pp. 223–252, 1977.

[118] Krishna M. Kavi, Bill P. Buckles, and U. Narayan Bhat, "A formal definition of data flow graph models," *IEEE Transactions on Computers*, vol. 35, no. 11, pp. 940–948, 1986.

[119] Krishna M. Kavi, Bill P. Buckles, and U. Narayan Bhat, "Isomorphisms between Petri nets and dataflow graphs," *IEEE Transactions on Software Engineering*, , no. 10, pp. 1127–1134, 1987.

[120] Maxime Pelcat, Karol Desnos, Julien Heulot, Clément Guy, Jean-François Nezan, and Slaheddine Aridhi, "PREESM: A dataflow-based rapid prototyping framework for simplifying multicore DSP programming," in *Proc. European Embedded Design in Education and Research Conference (EDERC)*. IEEE, 2014, pp. 36–40.

[121] W. Thies, M. Karczmarek, and S. Amarasinghe, "StreamIt: A language for streaming applications," in *Compiler Construction*, R. Nigel Horspool, Ed., vol. 2304 of *Lecture Notes in Computer Science*, pp. 179–196. Springer, 2002.

[122] H. P. Huynh, A. Hagiescu, O. Z. Liang, W.-F. Wong, and R. S. M. Goh, "Mapping streaming applications onto GPU systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2374–2385, 2014.

[123] G. Bilsen, M. Engels, R. Lauwereins, and J. Peperstraete, "Cyclo-static dataflow," *IEEE Transactions on Signal Processing*, vol. 44, no. 2, pp. 397–408, Feb 1996.

[124] Sander Stuijk, Marc Geilen, Bart Theelen, and Twan Basten, "Scenario-aware dataflow: Modeling, analysis and implementation of dynamic applications," in *Proc. International Conference on Embedded Computer Systems (SAMOS)*. IEEE, 2011, pp. 404–411.

[125] Eunjin Jeong, Dowhan Jeong, and Soonhoi Ha, "Dataflow model–based software synthesis framework for parallel and distributed embedded systems," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 26, no. 5, pp. 1–38, 2021.

[126] E. A. Lee and T. M. Parks, "Dataflow process networks," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 773–801, 1995.

[127] J. Eker and J. W. Janneck, "CAL language report," Tech. Rep. UCB/ERL M03/48, UC Berkeley, 2003.

[128] M. Mattavelli, I. Amer, and M. Raulet, "The Reconfigurable Video Coding standard [standards in a nutshell]," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 159–167, 2010.

[129] Gustav Cedersjö and Jörn W Janneck, "Tÿcho: a framework for compiling stream programs," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 6, pp. 1–25, 2019.

[130] H. Yviquel, A. Lorence, K. Jerbi, G. Cocherel, A. Sanchez, and M. Raulet, "Orcc: Multimedia development made easy," in *Proc. ACM International Conference on Multimedia*, 2013, pp. 863–866.

[131] Omair Rafique and Klaus Schneider, "SHeD: A framework for automatic software synthesis of heterogeneous dataflow process networks," in *Proc. Euromicro Conference on Digital System Design (DSD)*. IEEE, 2020, pp. 1–10.

[132] J. T. Buck and E. A. Lee, "Scheduling dynamic dataflow graphs with bounded memory using the token flow model," in *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, April 1993, vol. 1, pp. 429–432 vol.1.

[133] L. Schor, A. Tretter, T. Scherer, and L. Thiele, "Exploiting the parallelism of heterogeneous systems using dataflow graphs on top of OpenCL," in *Proc. IEEE Symposium on Embedded Systems for Real-time Multimedia (ESTIMedia)*, 2013, pp. 41–50.

[134] Jose Luis Pino, Shuvra S Bhattacharyya, and Edward A Lee, "A hierarchical multiprocessor scheduling system for DSP applications," in *Proc. Asilomar Conference on Signals, Systems and Computers*. IEEE, 1996, vol. 1, pp. 122–126.

[135] Pablo Pascual Campo, Vesa Lampu, Alexandre Meirhaeghe, Jani Boutellier, Lauri Anttila, and Mikko Valkama, "Digital predistortion for 5G small cell: GPU implementation and RF measurements," *Journal of Signal Processing Systems*, pp. 1–12, 2019.

[136] Mona Aghababaeetafreshi, Dani Korpi, Matias Koskela, Pekka Jääskeläinen, Mikko Valkama, and Jarmo Takala, "Software defined radio implementation of a digital self-interference cancellation method for inband full-duplex radio using mobile processors," *Journal of Signal Processing Systems*, vol. 90, no. 10, pp. 1297–1309, 2018.

[137] Lauri Anttila, Peter Handel, and Mikko Valkama, "Joint mitigation of power amplifier and I/Q modulator impairments in broadband direct-conversion transmitters," *IEEE Transactions on Microwave Theory and Techniques*, vol. 58, no. 4, pp. 730–739, 2010.

[138] Yujunrong Ma, Kshitij Nikhal, Jani Boutellier, Benjamin Riggan, and Shuvra S Bhattacharyya, "Dynamically reconfigurable perception using dataflow parameterization of channel attention," in *2023 57th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2023, pp. 747–754.

[139] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[140] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, pp. 1–74, 2021.

[141] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[142] R. Xie, H. Huttunen, S. Lin, S. S. Bhattacharyya, and J. Takala, "Resource-constrained implementation and optimization of a deep neural network for vehicle classification," in *Proceedings of the European Signal Processing Conference*, Budapest, Hungary, August 2016, pp. 1862–1866.

[143] Y.-H. Chen, J. Emer, and V. Sze, "Using dataflow to optimize energy efficiency of deep neural network accelerators," *IEEE Micro*, vol. 37, no. 3, pp. 12–21, 2017.

[144] ONNX developers, "Open neural network exchange," `https://github.com/onnx/onnx`, 2023, Version: 1.15.0.

[145] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu, "Image super-resolution using very deep residual channel attention networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 286–301.

[146] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[147] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[148] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu, "Eca-net: Efficient channel attention for deep convolutional neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11534–11542.

[149] Diganta Misra, Trikay Nalamada, Ajay Uppili Arasanipalai, and Qibin Hou, "Rotate to attend: Convolutional triplet attention module," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, pp. 3139–3148.

[150] Zequn Qin, Pengyi Zhang, Fei Wu, and Xi Li, "Fcanet: Frequency channel attention networks," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 783–792.

[151] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2736–2744.

[152] Mengran Liu, Weiwei Fang, Xiaodong Ma, Wenyuan Xu, Naixue Xiong, and Yi Ding, "Channel pruning guided by spatial and channel attention for dnns in intelligent edge computing," *Applied Soft Computing*, vol. 110, pp. 107636, 2021.

[153] Yuxin Zhang, Mingbao Lin, Chia-Wen Lin, Jie Chen, Yongjian Wu, Yonghong Tian, and Rongrong Ji, "Carrying out cnn channel pruning in a white box," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[154] Zhiqiang Chen, Ting-Bing Xu, Changde Du, Cheng-Lin Liu, and Huiguang He, "Dynamical channel pruning by conditional accuracy change for deep neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 2, pp. 799–813, 2020.

[155] E. A. Lee and T. M. Parks, "Dataflow process networks," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 773–799, 1995.

[156] S. S. Bhattacharyya, E. Deprettere, R. Leupers, and J. Takala, Eds., *Handbook of Signal Processing Systems*, Springer, third edition, 2019.

[157] Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A Ying, Anurag Mukkara, Rangharajan Venkatesan, Brucek Khailany, Stephen W Keckler, and Joel Emer, "Timeloop: A systematic approach to dnn accelerator evaluation," in *2019 IEEE international symposium on performance analysis of systems and software (ISPASS)*. IEEE, 2019, pp. 304–315.

[158] Tanay Narshana, Chaitanya Murti, and Chiranjib Bhattacharyya, "Dfpc: Data flow driven pruning of coupled channels without data.," in *The Eleventh International Conference on Learning Representations*, 2022.

[159] Jung Hwan Heo, Arash Fayyazi, Amirhossein Esmaili, and Massoud Pedram, "Sparse periodic systolic dataflow for lowering latency and power dissipation of convolutional neural network accelerators," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 2022, pp. 1–6.

[160] Hyoukjun Kwon, Ananda Samajdar, and Tushar Krishna, "Maeri: Enabling flexible dataflow mapping over dnn accelerators via reconfigurable interconnects," *ACM SIGPLAN Notices*, vol. 53, no. 2, pp. 461–475, 2018.

[161] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally, "Eie: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 243–254, 2016.

[162] Jani Boutellier, Yujunrong Ma, Jiahao Wu, Mir Khan, and Shuvra S Bhattacharyya, "Vr-prune: Decidable variable-rate dataflow for signal processing systems," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1819–1833, 2022.

[163] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al., "Tensorflow: Large-scale machine learning on heterogeneous systems," `https://www.tensorflow.org/`, 2022, Version 2.9.0.

[164] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., "Pytorch," `https://pytorch.org/`, 2022, Version 1.11.0.

[165] Libonnx developers, "Libonnx: a lightweight, portable pure c99 onnx inference engine," `https://github.com/xboot/libonnx`, 2022.

[166] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 248–255.