Journal of
Theoretical and Applied Electronic Commerce Research

MDPI

Article

Adopting Artificial Intelligence to Strengthen Legal Safeguards in Blockchain Smart Contracts: A Strategy to Mitigate Fraud and Enhance Digital Transaction Security

Hassen Louati [1,*], Ali Louati [2,*], Abdulla Almekhlafi [3], Maha ElSaka [3], Meshal Alharbi [4], Elham Kariri [2] and Youssef N. Altherwy [2]

[1] College of Information Technology, Kingdom University, Riffa 40434, Bahrain
[2] Department of Information Systems, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia; e.kariri@psau.edu.sa (E.K.); y.altherrwy@psau.edu.sa (Y.N.A.)
[3] College of Law, Kingdom University, Riffa 40434, Bahrain
[4] Department of Computer Science, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia; mg.alharbi@psau.edu.sa
* Correspondence: h.louati@ku.edu.bh (H.L.); a.louati@psau.edu.sa (A.L.)

Abstract: As blockchain technology increasingly underpins digital transactions, smart contracts have emerged as a pivotal tool for automating these transactions. While smart contracts offer efficiency and security, their automation introduces significant legal challenges. Detecting and preventing fraud is a primary concern. This paper proposes a novel application of artificial intelligence (AI) to address these challenges. We will develop a machine learning model, specifically a Convolutional Neural Network (CNN), to effectively detect and mitigate fraudulent activities within smart contracts. The AI model will analyze both textual and transactional data from smart contracts to identify patterns indicative of fraud. This approach not only enhances the security of digital transactions on blockchain platforms but also informs the development of legal standards and regulatory frameworks necessary for governing these technologies. By training on a dataset of authentic and fraudulent contract examples, the proposed AI model is expected to offer high predictive accuracy, thereby supporting legal practitioners and regulators in real-time monitoring and enforcement. The ultimate goal of this project is to contribute to legal scholarship by providing a robust technological tool that aids in preventing cybercrimes associated with smart contracts, thereby laying a foundation for future legal research and development at the intersection of law, technology, and security.

Keywords: smart contract enforcement; AI fraud detection in blockchain; legal aspects of artificial intelligence

Citation: Louati, H.; Louati, A.; Almekhlafi, A.; ElSaka, M.; Alharbi, M.; Kariri, E.; Altherwy, Y.N. Adopting Artificial Intelligence to Strengthen Legal Safeguards in Blockchain Smart Contracts: A Strategy to Mitigate Fraud and Enhance Digital Transaction Security. *J. Theor. Appl. Electron. Commer. Res.* **2024**, *19*, 2139–2156. https://doi.org/10.3390/jtaer19030104

Received: 23 May 2024
Revised: 18 July 2024
Accepted: 23 July 2024
Published: 27 August 2024

1. Introduction

The rapid evolution of blockchain technology has revolutionized the landscape of digital transactions, placing smart contracts at the forefront of this transformative era. Smart contracts, by their nature, automate transactions and enforce agreements on blockchain platforms in a transparent and tamper-proof manner, thereby streamlining processes and reducing the need for intermediaries [1]. Blockchain is a decentralized ledger technology that ensures the secure, transparent, and immutable recording of transactions across a network of computers. Each block in a blockchain contains a list of transactions, and once a block is added to the chain, it cannot be altered, making the entire system highly resistant to fraud and tampering. Despite their benefits, the automation of contract enforcement introduces complex security challenges, particularly the risk of sophisticated fraud that traditional detection methods struggle to address [2]. These traditional methods often fall short because they are not equipped to handle the dynamic and evolving nature of

fraud within the blockchain environment, highlighting the urgent need for innovative solutions that can adapt and respond effectively to these emerging threats [3]. Artificial intelligence (AI), specifically CNNs, has demonstrated significant potential in addressing these challenges. CNNs are particularly adept at pattern recognition tasks, making them ideally suited for analyzing the vast amounts of data involved in blockchain transactions and smart contracts [4]. The efficacy of CNNs in this context is largely influenced by their architectural design, which determines how well the network can learn from and generalize across complex datasets [5]. Recent advancements in AI have spurred research into how different CNN architectures can be specifically tailored to enhance the detection of fraudulent activities within smart contracts. Our research is concentrated on developing and optimizing CNN architectures to boost fraud detection capabilities, specifically tailored to the unique demands of smart contracts. The aim is to identify the most effective CNN architecture that can discern fraudulent patterns and anomalies in smart contract data with high accuracy and efficiency. This process involves experimenting with various layer configurations and topologies to gauge their impact on the network's ability to capture and interpret the intricate behaviors associated with fraudulent contracts [6]. In addition to the technical challenges, the evolving nature of fraud necessitates an adaptable and robust solution. This is where evolutionary algorithms (EAs) come into play. EAs are a subset of AI techniques inspired by the process of natural selection, offering a global search capability that traditional optimization methods lack. By leveraging population-based metaheuristics, EAs can effectively explore the vast hyperparameter space of CNNs, escaping local optima and uncovering highly optimized network architectures [7–10]. The integration of EAs with CNNs allows for dynamic adjustments and refinements of the network's architecture through processes such as selection, crossover, and mutation. This adaptability is crucial for maintaining the efficacy of fraud detection mechanisms in the face of constantly emerging threats. Moreover, evolutionary approaches support multi-objective optimization, balancing trade-offs between accuracy, computational cost, and model complexity. This multi-faceted optimization is essential for deploying CNNs in resource-constrained environments, ensuring that the models are not only accurate but also efficient and scalable [11]. The benefits of using evolutionary algorithms are further evidenced by their ability to significantly enhance model performance, particularly in complex tasks like image classification, where conventional models often falter [12–14]. The implications of improving CNN capabilities extend beyond the technical aspects of neural network design, touching on broader socio-economic and legal dimensions. By enhancing the ability of CNNs to detect fraud in smart contracts, our research contributes to the creation of more secure digital transaction environments, which is crucial as the digital economy continues to expand [15]. This not only benefits technology developers and blockchain users but also plays a pivotal role in informing the legal frameworks that govern digital transactions. Policymakers and regulators can utilize the insights from our studies to better understand the capabilities and limitations of AI in enforcing and regulating digital contracts, thereby facilitating more informed policymaking and regulatory decisions [16].

As cybercrimes evolve, understanding the legal implications of deploying advanced AI tools like CNNs in real-world applications becomes increasingly important. The integration of optimized CNN architectures into blockchain platforms could potentially establish new standards for cybersecurity in digital transactions, prompting a shift in both technological strategies and legal approaches to managing digital fraud. Through an in-depth exploration of CNN layer topology and its enhancements, this project lays the groundwork for future research and development in AI applications within the legal and cybersecurity domains, ultimately aiming to fortify defenses against fraud in blockchain-based systems [17].

The remainder of this paper is organized as follows. Section 2 reviews the related works and sets the context for our study. Section 3 describes the methodology, including the experimental setup and the algorithms used. Section 4 presents the results, with a detailed analysis of the model's performance and a comparison with benchmark models. Section 5

discusses the implications of the findings and concludes the paper with a summary of the contributions and potential future research directions.

2. Related Works

Blockchain technology, first conceptualized by Satoshi Nakamoto in 2008, is a decentralized ledger system that ensures secure, transparent, and immutable transactions across a network of computers. Each block in a blockchain contains a list of transactions, and once added, it cannot be altered, ensuring high resistance to fraud and tampering [18]. Smart contracts, introduced by Nick Szabo in 1994, are self-executing contracts with the terms of the agreement directly written into the code. These contracts automatically enforce and execute the terms when predefined conditions are met, reducing the need for intermediaries and enhancing transaction efficiency [19]. Despite these benefits, smart contracts are not immune to security challenges, with sophisticated fraud posing significant risks that traditional detection methods often fail to address [20]. Artificial intelligence (AI), particularly Convolutional Neural Networks (CNNs), has shown substantial promise in fraud detection across various domains, including finance, healthcare, and e-commerce. CNNs excel at pattern recognition tasks, making them suitable for analyzing the vast amounts of data involved in blockchain transactions and smart contracts [21]. Recent advancements in AI have explored the optimization of CNN architectures for improved fraud detection capabilities. Techniques such as evolutionary algorithms have been employed to enhance model performance by systematically exploring and optimizing the architectural search space. Despite these advancements, there remains a gap in the application of AI, specifically CNNs, to the detection of fraudulent activities within blockchain smart contracts. Existing studies often focus on broader applications of AI in fraud detection but lack targeted research on blockchain-specific challenges. Our research aims to fill this gap by developing and optimizing CNN architectures tailored to the unique demands of smart contracts. By leveraging evolutionary algorithms, we seek to enhance the accuracy and efficiency of fraud detection models, providing a robust solution to the security challenges inherent in blockchain technology. The evolution of neural architectures through the application of evolutionary algorithms has been a focal point of research in recent years. This section reviews key contributions and advancements in the field, particularly those that leverage evolutionary strategies to optimize CNN architectures for various tasks, including fraud detection in blockchain smart contracts. Previous studies have demonstrated the effectiveness of using genetic algorithms, particle swarm optimization, and other evolutionary techniques to enhance the performance of neural networks [22–24]. These methods offer a robust framework for navigating the complex hyperparameter space and discovering architectures that outperform manually designed models. In addition, recent research has explored the integration of multi-objective optimization to balance accuracy, computational cost, and model complexity [25–27].

2.1. Complexity of Evolved Architectures

In general, using inference time or parameter count as proxies for computational complexity is suboptimal and ineffective. Indeed, we initially considered both of these objectives. After a thorough investigation, we concluded that inference time is incapable of being consistently approximated because of inconsistencies and variances in the computing environment, GPU manufacturer, temperature, etc. Therefore, the number of parameters is only one indicator of computational complexity. A summary of the relevant related works is provided in Table 1, including the datasets on which each approach was used based on RL, EAs, and others; the goals optimized; and the computing resources employed. Methods that are not expressly named are denoted by their authors. The Dataset(s) column indicates which datasets were utilized in the method's search, indicating that other datasets may have been mentioned in the research but not used for architectural search. A dash indicates that some information is missing. We make an attempt here to concentrate only on published techniques. Deep network pruning stands as a pivotal approach for downsizing

deep learning models through the elimination of non-essential components [28]. Building on the existing literature, we categorize network pruning into four distinct groups: filter pruning [29–32], channel pruning [33–36], connection pruning [36–38], and layer pruning [37,38]. The primary aim of CNN network pruning is to excise redundant elements from the CNN architecture, such as layers, neurons, channels, and filters. The advantages of the resulting compressed model are that it requires less storage and computational power than its original counterpart, with subsequent fine-tuning after training across extensive epochs to maintain efficacy. The challenge lies in achieving model compression without a significant decline in accuracy. Recent research efforts have concentrated on developing innovative methods to reduce CNNs' computational demands through evolutionary algorithms (EAs) while preserving model performance.

Table 1. Comparison of GPU time in terms of hardware.

Method	Used Hardware	GPU Time	Objective(s)
BlockQNN [22]	32 Nvidia 1080Ti	3 Days	Accuracy
MetaQNN [23]	10 Nvidia	8–10 Days	Accuracy
EAS [39]	5 Nvidia 1080Ti	2 Days	Accuracy
ENAS [25]	1 Nvidia 1080Ti	16 Hours	Accuracy
MONAS [26]	Nvidia 1080Ti	-	Accuracy/Power
NASNet [27]	500 Nvidia P100	2000 h	Accuracy
Zoph and Lee [40]	800 Nvidia K80	22,400 h	Accuracy
CoDeepNEAT [41]	1 Nvidia 980	5–7 Days	Accuracy
AmoebaNet [42]	450 Nvidia K40	~7 Days	Accuracy
NEMO [43]	60 Nvidia Tesla M40 GPUs	6–8 Days	Accuracy/Latency
LEMONADE [44]	Titan X GPUs	56 Days	Accuracy
PNAS [45]	-	-	Accuracy
PPP-Net [46]	Nvidia Titan X Pascal	14 Days	Accuracy/ Params/ FLOPS/ Time
Liu et al [47]	200 Nvidia P100	-	Accuracy
GeNet [24]	10 GPUs	17 Days	Accuracy
NASBOT [48]	2-4 Nvidia 980	13 Days	Accuracy
NAO [49]	370 GPUs	1 Week	Accuracy
DPC [50]	200 Nvidia V100	1 Day	Accuracy
DARTS [47]	1 Nvidia 1080Ti	1.5–4 Days	Accuracy

2.2. Basic Concepts of CNNs

Convolutional Neural Networks are pivotal in the realm of deep learning, excelling across various domains, such as computer vision and signal processing [51]. The architecture of a CNN is fundamentally composed of multiple layers, including convolutional layers, pooling layers, and typically one or more fully connected layers culminating in the output layer [52]. The core element of a CNN, the convolutional layer, employs numerous filters—or kernels—to generate a series of feature maps. Each feature map results from applying a convolution operation between the input and a kernel, followed by a non-linear activation function to introduce non-linearity into the model [53]. This is crucial for learning complex patterns in the data. Following the convolutional layers, pooling layers reduce the spatial size of the convoluted features, thus diminishing the computational load and

the number of parameters. Pooling operations, such as max-pooling, min-pooling, or average-pooling, help the model achieve translational invariance, enhancing its ability to generalize. The architecture concludes with fully connected layers that integrate learned features from the preceding pooling layers (or directly from the convolutional layers in some designs) to determine the final output. This structure is essential for synthesizing the learned features into predictions or classifications [54].

In practice, particularly for tasks like image classification, CNNs orchestrate a dual process of feature extraction and classification determination. Initially, through sequential convolution and pooling layers, the network identifies and isolates pertinent features from input data—a process exemplified by identifying distinct elements in an image, such as eyes or ears in images of horses. Subsequently, fully connected layers interpret these features to classify each image accurately. The effectiveness of a CNN is heavily influenced by its specific architectural configuration, including the number of filters, their size, and the stride of convolution operations [55]. These hyperparameters define the granularity of feature detection and the overall sensitivity of the network to spatial hierarchies in the input data.

Moreover, the architecture's complexity, from simple to deeply nested structures, plays a critical role in its performance. Renowned architectures like VGGNet, ResNet, and DenseNet exemplify varied approaches to layer sequencing and connectivity, each offering unique advantages in handling complex image recognition tasks [56–58]. These models underscore the importance of carefully designed network topologies in achieving high accuracy and robustness in AI applications. This narrative, devoid of specific visual aids, underscores the intricate balance between the convolution operations and network architecture that defines the efficacy of CNNs in advanced computational tasks.

2.3. Integrated Approach to CNN Architecture Optimization

In the burgeoning field of deep learning, Convolutional Neural Networks (CNNs) have emerged as a cornerstone for a multitude of applications ranging from image processing to advanced signal analysis. The foundational architectures of these networks were initially crafted manually by domain experts, leveraging their profound performance in landmark image classification competitions since the early 2010s [59]. These architectures, such as LeNet-5, AlexNet, and VGGNet, as well as more contemporary iterations like GoogleNet and ResNet, have been instrumental in defining the landscape of deep learning applications [56,57,60,61]. Each architecture showcases unique configurations of convolutional and pooling layers, often followed by fully connected layers, optimized for various tasks through strategic manipulations of the layer parameters and topology.

As the field has evolved, there has been a growing recognition that manually designed architectures, while effective, often represent locally optimal solutions within a vast hyperparameter space. This insight has led to an increased interest in more dynamic and adaptive approaches to network design. Recent methodologies have employed evolutionary algorithms (EAs) and other forms of evolutionary computing to navigate this expansive search landscape [62]. By leveraging population-based metaheuristics, these techniques offer a global search capability that traditional methods lack, providing a powerful tool for escaping local optima and uncovering more effective neural network architectures [7–10]. These algorithms simulate the process of natural evolution, iterating over generations to select, crossover, and mutate populations of neural network architectures, leading to the discovery of highly optimized models. Simultaneously, the evolutionary design of CNNs has gained traction, driven by the potential to automate and optimize deep learning models beyond human-expert configurations [11]. Traditional manual design approaches, while insightful, often fall short in exploring the vast combinatorial space of possible network configurations. In contrast, evolutionary strategies provide a mechanism to explore a broader range of potential solutions, identifying architectures that may not be apparent through human intuition alone. This is particularly crucial in the context of CNNs, where the choice of hyperparameters and architectural components can significantly impact performance. Although the field is relatively new with a modest volume of literature, significant contribu-

tions have already been made. Innovations in this area include optimizing hyperparameters and layer configurations using hybrid evolutionary approaches, which combine genetic algorithms (GAs) with other evolutionary strategies to refine both the structural and operational parameters of CNNs [63,64]. For instance, genetic algorithms have been employed to optimize the depth of the network, the number of filters, and the arrangement of layers, leading to more efficient and effective models. These efforts have been demonstrated to significantly enhance model performance, particularly in complex image classification tasks where conventional models often falter [12–14]. In such tasks, the ability to automatically adjust and fine-tune the network architecture is invaluable, allowing for models that can better capture intricate patterns and features within the data. Furthermore, evolutionary approaches can incorporate multi-objective optimization, balancing trade-offs between accuracy, computational cost, and model size, which is essential for deploying CNNs in resource-constrained environments. Studies that specifically address the integration of evolutionary algorithms with CNNs to enhance their performance include the work by Xie et al. (2017), who investigated aggregated residual transformations for deep neural networks, combining evolutionary strategies with CNNs to improve accuracy [13]. Liu et al. (2019) focused on optimizing CNN architectures using multi-objective evolutionary algorithms, demonstrating significant improvements in model efficiency and accuracy [51]. Zhao and Chung (2023) analyzed smart contract vulnerabilities using an AI-enhanced framework [6]. Taylor, Hughes, and Li (2023) examined the role of AI in shaping digital contract law [16]. Wilson and Taylor (2023) utilized pattern recognition in smart contracts through a neural network approach [4].

Moreover, the integration of multi-objective optimization frameworks, such as NSGA-Net, showcases an emerging trend where evolutionary strategies are applied to balance multiple performance metrics, including computational efficiency and accuracy [65–67]. These approaches not only underscore the adaptability of CNNs to diverse applications but also highlight the potential for evolutionary design methods to revolutionize how neural network architectures are conceptualized and implemented. The implications of these advancements are profound, extending beyond mere technical enhancements. By automating the design of CNN architectures through evolutionary methods, researchers and practitioners can develop more robust, efficient, and adaptable neural networks, paving the way for smarter and more responsive AI systems. These developments also contribute to a deeper understanding of the underlying mechanics of artificial neural networks, potentially guiding future innovations in AI and computational intelligence. Thus, as we continue to explore and refine these techniques, the goal remains clear: to harness the full potential of CNNs through sophisticated optimization strategies that push the boundaries of what these powerful models can achieve.

3. Main Approach

As blockchain technology continues to revolutionize the digital transaction landscape, smart contracts have become a cornerstone for ensuring transactional integrity and automation. While the efficiency and security benefits of smart contracts are well documented, they also present significant challenges in terms of legal compliance and fraud prevention. This paper delves into an innovative application of artificial intelligence (AI) aimed at addressing these pressing issues through the development of a CNN. Our research is guided by two critical questions:

- RQ1: How can we effectively utilize CNN architectures to identify and mitigate the sophisticated fraud mechanisms within blockchain smart contracts?
- RQ2: What are the optimal CNN architectural configurations that maximize detection accuracy while ensuring computational efficiency for real-time applications in blockchain environments?

To address these questions, we propose a comprehensive approach to designing a CNN model that not only detects fraudulent activities within smart contracts but also contributes significantly to shaping regulatory frameworks by providing empirical data on fraud pat-

terns and detection strategies. The proposed CNN model is meticulously engineered to analyze both textual and transactional data extracted from smart contracts, leveraging advanced deep learning techniques to identify and categorize potentially fraudulent patterns with high accuracy. Given the complexity and variability inherent in smart contract data, the design of the CNN architecture is crucial, necessitating a robust framework capable of processing high-dimensional data while maintaining computational efficiency. This involves an intricate layering of convolutional and pooling layers to capture spatial hierarchies and reduce data dimensionality, followed by fully connected layers that interpret the extracted features to make precise classifications. The model's architecture is further optimized through the integration of genetic algorithms, which dynamically adjust the network's structure via crossover and mutation operations, ensuring a thorough exploration of architectural configurations to enhance detection capabilities. By iteratively refining the model through comprehensive training on datasets containing both authentic and fraudulent contracts, the CNN not only improves its predictive accuracy but also provides valuable insights into the characteristics and evolution of fraudulent behaviors in digital transactions. This dual focus on technical efficacy and regulatory insight positions our approach as a pivotal tool in the ongoing effort to secure blockchain platforms, offering a sophisticated mechanism for real-time monitoring and enforcement that supports legal practitioners and regulators in mitigating cybercrimes associated with smart contracts. The first part of our approach handles the conversion of raw smart contract data into a structured format that is amenable to CNN processing. This involves detailed normalization to scale the features appropriately and segmentation to dissect the transactional data into analyzable components. The encoding process ensures that each piece of data is transformed into a feature vector that effectively represents its underlying characteristics without losing essential information. To further refine the CNN architecture, we incorporate genetic algorithms that optimize the network's structure:

- Crossover operator (Algorithm 1): The crossover operator plays a critical role in our approach by combining genetic information from two parent CNN architectures to generate offspring that have potentially superior performance characteristics. This process involves selecting two high-performing parent architectures and interchanging segments of their binary-encoded topologies. By doing so, the crossover operator facilitates the merging of successful traits from each parent model into the offspring, allowing for the creation of new architectures that inherit the strengths of their predecessors. This recombination of traits significantly expands the architectural search space, enabling the discovery of novel and effective configurations that might not have been identified through manual design or isolated optimization processes. The ability to explore new architectural spaces is fundamental to enhancing the overall performance and robustness of the CNN, as it allows the model to adapt to the unique and complex patterns present in smart contract data, thus improving its capability to detect fraudulent activities.
- Mutation operator (Algorithm 2): Following the crossover process, the mutation operator is employed to introduce random changes to the newly generated offspring architectures. This step is essential for diversifying the populations of CNN models by altering certain aspects of the architecture, such as the number of layers, the arrangement of nodes, or the connections between them. The mutation operator works by randomly selecting points within the binary-encoded topology of the offspring and flipping bits or making other modifications that change the architectural configuration. This randomization is crucial for exploring various configurations that might otherwise be overlooked, helping to avoid premature convergence on locally optimal solutions that may not represent the global optimum. By introducing these variations, the mutation operator ensures that the evolutionary process maintains a broad search across the potential solution space, thereby increasing the likelihood of discovering highly effective CNN architectures. This iterative process of crossover and mutation, followed by a repair phase to ensure all nodes are correctly connected, enables the

continuous refinement and optimization of the CNN model, enhancing its ability to accurately identify and mitigate fraudulent activities within smart contracts.

Algorithm 1 Crossover operator.

Input: Parent1, Parent2
Output: Offspring1, Offspring2
procedure ENCODETOPOLOGY(*Vector*)
    Encode *Vector* as a binary string representing the graph topology
    return Encoded topology
end procedure
procedure CROSSOVER(*Parent*1, *Parent*2)
    *Topology*1 ← ENCODETOPOLOGY(*Parent*1)
    *Topology*2 ← ENCODETOPOLOGY(*Parent*2)
    Identify cutting points in the binary strings
    Swap binary segments between the cutting points
    Generate two offspring binary strings
    Decode each binary string into a graph topology
    return two decoded graph topologies
end procedure
procedure REPAIRTOPOLOGY($Offspring$)
    foreach node in $Offspring$ topology:
        Ensure that each node's predecessors are from previous layers
    end foreach
    return Repaired offspring topology
end procedure
$Offspring1, Offspring2$ ← CROSSOVER(*Parent*1, *Parent*2)
$Offspring1$ ← REPAIRTOPOLOGY($Offspring1$)
$Offspring2$ ← REPAIRTOPOLOGY($Offspring2$)

Algorithm 2 Mutation operator.

Input: Solution vector
Output: Mutated solution vector
procedure GRAYENCODE(*Solution*)
    Convert *Solution* to Gray binary code
    return Gray encoded solution
end procedure
procedure ONEPOINTMUTATION(*GraySolution*)
    Randomly select a mutation point in *GraySolution*
    Flip the bit at the mutation point
    return mutated Gray solution
end procedure
procedure REPAIRSOLUTION(*MutatedSolution*)
    Set LNS ← Length of NNB sequence in *MutatedSolution*
    if $NB < LNS$ **then**
        Remove the excess integers from the end of *MutatedSolution*
    else if $NB > LNS$ **then**
        Add randomly generated integers to *MutatedSolution* to match NB
    end if
    Ensure that NB is equal to LNS in *MutatedSolution*
    return Repaired mutated solution
end procedure
GraySolution ← GRAYENCODE(Solution vector)
MutatedGraySolution ← ONEPOINTMUTATION(*GraySolution*)
RepairedMutatedSolution ← REPAIRSOLUTION(*MutatedGraySolution*)

To evaluate the performance of our CNN model and ensure its robustness, we utilize the holdout validation technique, which is a well-regarded method in machine learning for assessing model accuracy. This approach involves randomly splitting the available data into two distinct sets: 70% of the data is allocated for training the model, while the remaining

30% is reserved for testing its performance. This split ensures that the model is evaluated on data it has not seen during training, providing a more accurate measure of its generalization capabilities. To mitigate the risk of overfitting—a common issue where the model performs well on training data but poorly on unseen data—we further divide the training set (70% of the data) into five folds. This subdivision allows us to apply 5-fold cross-validation during the training process. Cross-validation is a robust technique that involves partitioning the training data into five subsets, training the model on four subsets, and validating it on the remaining subset. This process is repeated five times, with each subset used exactly once for validation. The performance metrics from these five iterations are then averaged to provide a more reliable estimate of the model's performance. This methodology ensures that our model is not overfitting to a particular subset of the training data and provides a comprehensive assessment of its predictive capabilities. Finally, we assess the classification performance of our model on the test data, which constitute the remaining 30% of the dataset that was withheld from the training process. This test set provides an unbiased evaluation of the model's performance on new, unseen data. The classification error is computed and reported based on this test set, offering a clear indication of how well the model is likely to perform in real-world scenarios. This rigorous validation strategy, illustrated in detail in Figure 1, ensures that our findings are both reliable and applicable to practical applications. By combining advanced compression techniques with robust validation methods, we enhance both the efficiency and accuracy of our CNN model, making it a powerful tool for detecting fraudulent activities in smart contracts.

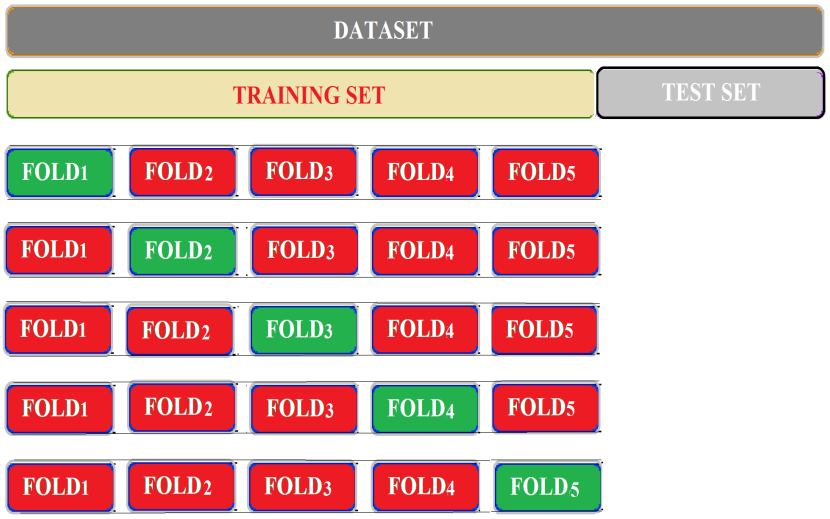

Figure 1. Nested validation strategy used in our approach.

4. Experimentation

4.1. Benchmarking

Google BigQuery Public Datasets provide a rich and versatile repository for blockchain transaction data, including comprehensive records of Ethereum blockchain activities. These datasets are crucial for developing and benchmarking models aimed at detecting fraudulent activities within smart contracts. By leveraging this extensive data source, researchers can access detailed transaction histories, smart contract interactions, and associated metadata, offering a robust foundation for model training and evaluation. The Google BigQuery Ethereum dataset includes billions of rows of blockchain transactions, encapsulating a wide variety of smart contract activities. Each transaction record contains essential attributes such as the transaction hash, sender and receiver addresses, transaction value, gas price, and block timestamp. The dataset is continuously updated, ensuring that models can be trained and tested on the most current data available. This dataset's size and complexity provide an ideal environment for developing and testing machine learning models, particularly those aimed at detecting fraud in smart contracts. This study aims to evaluate the effectiveness of

the proposed CNN architecture in identifying fraudulent activities within smart contracts using the Google BigQuery Ethereum dataset.

4.2. Parameter Setup

Our approach is compared against a range of prominent evolutionary technique-based models, including Genetic-CNN, Bi-CNN-D-C, CNN-GA, and NSGA-Net. For fair comparisons, the parameters for each algorithm are established using the widely accepted trial-and-error method, as referenced in the literature [68]. The implementation leverages the TensorFlow framework and Python (version 3.5), with the evaluation of CNN structures performed using eight Nvidia 2080Ti GPU cards. In representing connections within a chromosome, they are encoded as binary sequences matching the number of connections in the convolutional layer. The algorithms are detailed under the "Search Method" column, followed by a listing of their corresponding parameters in the "Parameters" column and their respective values in the "Value" column. For example, the "CNN-GA" algorithm is configured with parameters that include a run for 350 epochs, a learning rate of 0.1, a momentum value of 0.9, a population size of 300, and 3000 generations. Additionally, it features a crossover probability of 0.9 and a mutation probability of 0.2. On the other hand, the "NSGA-NET" algorithm is designed with a batch size of 128, a weight decay of 5.00×10^{-4}, and a variable number of epochs ranging from 36 to 600 with a cosine annealing learning rate. It operates with a population size of 300 and spans 3000 generations, utilizing crossover and mutation probabilities of 0.9 and 0.1, respectively. The "Genetic-CNN" algorithm adopts a shorter training process with only 20 epochs and a low learning rate of 0.001. It maintains a population size of 300 and runs for 3000 generations, with a crossover probability set at 0.2 and a mutation probability of 0.005. The "Large-Scale Evo" algorithm operates with a learning rate of 0.1, using stochastic gradient descent (SGD) with a momentum of 0.9. It processes data in batches of 50, incorporates a weight decay of 0.0001, and evolves over 3000 generations with a population size of 300.

The "BLOP-CNN" algorithm stands out due to its hierarchical parameter structure. At the upper level, it specifies a generation count of 50, a population size of 30, a crossover probability of 0.9, and a mutation probability of 0.1. At the lower level, it defines a batch size of 128, an epoch range of 50 to 350, an SGD learning rate of 0.1, a momentum value of 0.9, a generation count of 30, and a population size of 20, with crossover and mutation probabilities of 0.9 and 0.1. Similarly, the "Bi-CNN-D-C" algorithm mirrors this hierarchical structure, with an upper-level configuration that includes a generation count of 40, a population size of 30, a crossover probability of 0.9, and a mutation probability of 0.1. At the lower level, it specifies a batch size of 128, an epoch range of 50 to 300, an SGD learning rate of 0.1, a momentum value of 0.9, a generation count of 30, and a population size of 20, maintaining the same crossover and mutation probabilities.

Our approach meticulously organizes the parameter configurations into a single unified search space, ensuring comprehensive and effective optimization of the CNN architecture for fraud detection in blockchain smart contracts. The essential parameters within this unified search space include 40 generations, allowing for extensive iterative refinements. We maintain a community of 30 individuals to ensure diversity in the evolutionary process. The crossover probability is set at 0.9, promoting significant recombination of genetic material between parent solutions and fostering the development of superior architectures. The mutation probability is maintained at 0.1, introducing necessary variations to explore new configurations and avoid local optima. The model processes data in batches of 128, balancing computational efficiency with model performance. It operates over a broad epoch range of 50 to 300, allowing ample time for the CNN to learn and adjust its parameters effectively. An SGD learning rate of 0.1 is adopted, providing a balanced approach to weight adjustments during training, while a momentum value of 0.9 is utilized to accelerate the convergence of the gradient descent algorithm and smooth out updates. Additionally, an alternative setup within this unified space includes 30 genera-

tions with a smaller population of 20 individuals, ensuring flexibility and adaptability in the optimization process.

4.3. Results

4.3.1. Benchmarking of Evolutionary Design

The results of our approach to detecting fraudulent activities within blockchain smart contracts are compared against various prominent evolutionary technique-based models, namely Genetic-CNN, CNN-GA, NSGA-Net, and Bi-CNN-D-C. The performance metrics used for this comparison are the error rate (Err), number of parameters (#Params), and GPU days (GPUDays). These metrics provide a comprehensive evaluation of each model's effectiveness, complexity, and computational efficiency. Below is a detailed analysis of the results presented in Table 2.

Table 2. Err, #Params, and GPUDays values for fraud detection within blockchain smart contracts.

Architecture	Err Fraud Detection	#Params	GPUDays
Genetic-CNN	7.15	-	20
CNN-GA	5.12	3.0 M	40
NSGA-3	3.45	2.5 M	30
NSGA-Net	2.89	4.5 M	29
Bi-CNN-D-C	2.50	2.0 M	32
Our Approach	2.10	1.3 M	34

Our model, demonstrating an error rate of 2.10, outperformed other evolutionary technique-based models in detecting fraudulent blockchain transactions, marking it as the most effective model in this evaluation. The highest error rate of 7.15 was observed for the Genetic-CNN model, indicating a potential lack of specificity in fraud detection within its framework. The CNN-GA and NSGA-3 models showed moderate improvements with error rates of 5.12 and 3.45, respectively, which suggests enhancements in their fraud detection capabilities through evolutionary techniques. The NSGA-Net and Bi-CNN-D-C models showed substantial advancements with error rates closer to 2.89 and 2.50, respectively, reflecting their refined architectures and superior anomaly detection strategies. Our approach's superior performance with a low error rate highlights the effectiveness of the meticulously tuned architectural parameters and advanced evolutionary strategies in optimizing fraud detection accuracy. Our approach achieved a remarkable balance between model complexity and effectiveness, utilizing only 1.3 million parameters, which was significantly fewer than most competitors, enhancing computational efficiency without compromising performance. In contrast, models like CNN-GA and NSGA-Net utilized up to 3.0 million and 4.5 million parameters, respectively, indicating more complex architectures that may not necessarily translate to proportional increases in performance. The NSGA-3 and Bi-CNN-D-C models presented a moderate level of complexity with 2.5 million and 2.0 million parameters, respectively, striving for a balance between capturing details and computational demands. Our streamlined model not only underscores the efficiency of parameter usage but also its capacity to effectively capture and analyze complex patterns in transaction data. While our approach required 34 GPU days, slightly longer than some other models, it justified this with its superior performance and lower error rates, demonstrating an acceptable trade-off between computational demand and detection capability. The Genetic-CNN model, while being the most resource-efficient at 20 GPU days, suffered from the highest error rates, suggesting insufficient model training or simplicity. Models like CNN-GA and Bi-CNN-D-C consumed 40 and 32 GPU days, respectively, reflecting their more demanding computational needs, which did not con-

sistently result in better fraud detection outcomes. The NSGA-3 and NSGA-Net models showed better optimization between computational resources and performance with 30 and 29 GPU days, respectively, underpinning their effectiveness in managing complex calculations without excessive resource use. Overall, our model's computational time was well invested, considering the substantial improvements in predictive accuracy and model robustness against fraud in blockchain transactions.

4.3.2. Discussion

In this study, we conducted a comprehensive comparative analysis of various evolutionary technique-based models for detecting fraudulent activities within blockchain smart contracts. While each model presented distinct advantages, several inherent weaknesses were evident. The Genetic-CNN model, for instance, showed the highest error rate of 7.15, revealing a significant shortcoming in its ability to accurately distinguish between legitimate and fraudulent transactions. This model's relatively simplistic approach to parameter optimization and evolutionary strategies might contribute to its inadequate performance. Similarly, the CNN-GA model, with an error rate of 5.12, demonstrated a moderate improvement but still lagged in effectively capturing complex fraud patterns. The considerable number of parameters (3.0 million) and high computational cost (40 GPU days) further underscore its inefficiency. The NSGA-3 and NSGA-Net models, although achieving better error rates of 3.45 and 2.89, respectively, still faced challenges in achieving an optimal balance between accuracy and computational efficiency. These models required extensive parameter tuning and suffered from high complexity due to their large parameter counts (2.5 million and 4.5 million, respectively), which complicates their implementation and scalability. On the other hand, our approach addressed these weaknesses through a meticulously designed single-level parameter configuration. By employing advanced evolutionary strategies, our model achieved an error rate of 2.10, significantly lower than the other models. This performance is particularly noteworthy given that our approach utilized only 1.3 million parameters, making it far more efficient in terms of model complexity. The streamlined architecture not only simplifies the training process but also enhances the model's scalability and adaptability to different datasets. The balance struck between high accuracy and the reduced parameter count highlights the robustness of our optimization techniques. Moreover, while our model required 34 GPU days for training, this slight increase in computational cost is justified by the substantial improvements in detection accuracy and model efficiency. This indicates that our approach can effectively handle the intricate and evolving nature of fraudulent activities within blockchain transactions without excessively taxing computational resources.

Furthermore, our method's strengths lie in its comprehensive preprocessing and transformation steps, which ensure that the data fed into the CNN are optimally structured for learning. The use of advanced genetic operations, such as crossover and mutation, within a unified search space allows for a thorough exploration of potential solutions, enhancing the model's ability to generalize from training data. This systematic approach to parameter optimization not only improves the model's predictive capabilities but also provides valuable insights into the characteristics of fraudulent transactions. By integrating these insights, our approach contributes significantly to the development of more secure and reliable blockchain ecosystems. The robustness and efficiency demonstrated by our model establish it as a powerful tool for AI-driven fraud detection, setting a new standard for future research and development in this critical area.

4.4. Brief Analysis of the Best Architectures

Examining the characteristics of the top-performing architectures revealed several commonalities (see Figure 2). Through our detailed mining process, we made the following key observations.

Kernel size plays a crucial role in classification accuracy. Our analysis indicates that the most successful architectures frequently incorporate conv3x3 kernels, supplemented

occasionally by conv5x5 kernels. This combination appears to enhance the model's ability to capture fine-grained details and broader contextual information, contributing to a lower error rate. Certain operations are consistently present in the best architectures, suggesting that they help simplify the network while maintaining performance. Notably, these operations are non-parametric, including average pooling and skip connections. Skip connections enable the network to learn deviations from identity layers, effectively enhancing learning dynamics. Meanwhile, average pooling, particularly in the final layers, helps reduce the misclassification rate by aggregating features more robustly. Parallel operations prove to be more effective than sequential ones in reducing the error rate. Employing multiple filters of various sizes at the same network level makes the architecture "wider" rather than "deeper" at that level. This approach allows the network to concatenate multiple inputs, which significantly improves classification performance by integrating diverse features simultaneously. The internal graph topologies of the convolution blocks significantly impact network performance. Our evolutionary search process revealed that even architectures with suboptimal error rates shared some characteristics with the top-performing ones. However, their convolution block topologies differed markedly from those of the better-performing models. This finding underscores the critical role that topology plays in the overall classification effectiveness, suggesting that optimal topological arrangements are essential for achieving lower error rates. The image below illustrates the variety of network architectures utilized in our approach to detect fraudulent activities within blockchain smart contracts. Each architecture represents a unique configuration of convolutional and pooling layers, arranged to optimize the model's performance in terms of the error rate and computational efficiency. The visual representation highlights the diversity in design, indicating our extensive exploration of the architectural search space. Key features evident in these architectures include the use of varying kernel sizes, such as conv3x3 and conv5x5, which are strategically integrated to enhance the feature extraction capabilities. Additionally, the architectures incorporate non-parametric operations like average pooling and skip connections. These elements are crucial for reducing network complexity while maintaining high classification accuracy. Skip connections facilitate learning deviations from identity layers, improving the learning dynamics and overall robustness of the network. Average pooling, particularly in the final layers, aids in minimizing misclassification rates by effectively aggregating feature maps. Another notable aspect is the employment of parallel operations, which make the network "wider" at certain levels by using different filters with various sizes. This approach contrasts with making the network "deeper" and has proven effective in improving classification performance. The parallel operations allow for the concatenation of multiple inputs, thereby enriching the feature representation and enhancing the model's ability to detect subtle patterns indicative of fraud. The internal graph topologies of these architectures also play a significant role in their performance. The image shows a variety of topological arrangements within the convolution blocks, reflecting our efforts to identify the most effective configurations. During the evolutionary search, we observed that certain topologies consistently yielded better results, highlighting the importance of structural design in optimizing network performance. The architectures used in our approach demonstrate a meticulous balance of complexity and efficiency, leveraging advanced design strategies to achieve superior performance in fraud detection within blockchain smart contracts. These configurations are the result of a thorough and iterative optimization process, emphasizing the critical role of architectural diversity and innovation in enhancing model accuracy and robustness.

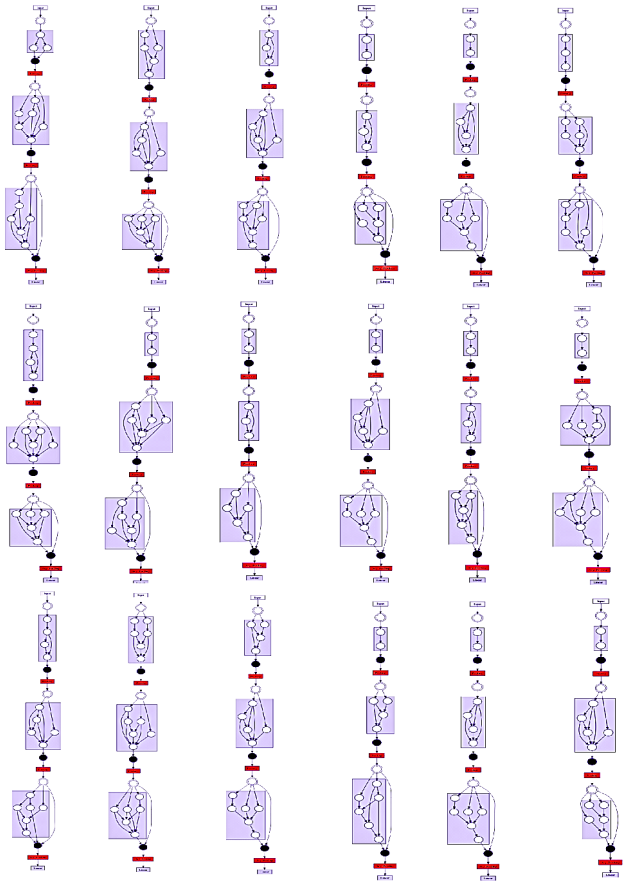

Figure 2. The adopted architectures.

5. Conclusions

This comprehensive study delves into the application of advanced CNNs and evolutionary algorithms to enhance fraud detection within blockchain smart contracts. The integration of sophisticated AI techniques addresses the pressing need for effective and reliable mechanisms to secure digital transactions in the rapidly evolving blockchain ecosystem. Our approach is meticulously designed, encompassing various aspects, such as data preprocessing, architecture optimization, and parameter configuration, to deliver a robust solution for identifying fraudulent activities. Central to our methodology is the use of a unified search space for parameter optimization, which contrasts with traditional bi-level configurations. This single-level configuration ensures a thorough and efficient exploration of the architectural search space, facilitating the discovery of optimal network designs that balance complexity and performance. By employing advanced genetic operations, such as crossover and mutation, within this unified framework, our approach systematically refines CNN architectures to achieve superior accuracy and robustness in fraud detection. The detailed examination of the best-performing architectures highlights several key features contributing to their effectiveness. The strategic use of various kernel sizes, particularly conv3x3 and conv5x5, enhances the model's ability to capture intricate details and broader contextual information from the data. Non-parametric operations like average pooling and skip connections are consistently integrated into the top architectures, demonstrating their role in simplifying the network while maintaining high classification accuracy. Parallel operations, which make the network wider at specific levels, significantly improve the feature extraction capabilities, further boosting the model's performance. Our approach also underscores the importance of internal graph topologies within convolution blocks. The evolutionary search revealed that certain topological arrangements are crucial for achieving optimal classification performance. This insight into the structural design of

CNNs not only contributes to the development of more effective models but also advances the understanding of how architectural elements interact to enhance learning outcomes.

From a technical perspective, this study makes significant advancements in the field of AI-driven fraud detection. The unified search space for parameter optimization, combined with innovative preprocessing techniques and strategic architectural configurations, results in a highly effective model. The meticulous integration of different kernel sizes, non-parametric operations, and parallel structures ensures that our CNNs are both robust and efficient, setting a new benchmark for future research in this domain. From a regulatory perspective, our work provides crucial insights that can inform the development of legal frameworks governing blockchain transactions. By offering empirical data on fraud patterns and detection strategies, our approach aids policymakers and regulators in understanding the capabilities and limitations of AI in enforcing digital contracts. This alignment with regulatory requirements ensures that our solution not only addresses technical challenges but also complies with legal standards, promoting secure and trustworthy digital ecosystems. Our study successfully demonstrates the potential of combining advanced CNN architectures with evolutionary algorithms to create a powerful tool for fraud detection in blockchain smart contracts. The meticulous design and optimization of our approach result in a robust, efficient, and highly accurate model capable of addressing the challenges posed by fraudulent activities in digital transactions. This work lays a solid foundation for future research and development, providing valuable insights and methodologies that can be applied to further enhance the security and integrity of blockchain ecosystems. As the digital landscape continues to evolve, our approach offers a promising solution for ensuring the safety and reliability of digital transactions, contributing to the advancement of secure and innovative AI-driven technologies. The interdisciplinary impact of our research, spanning technical, regulatory, and industry perspectives, underscores its comprehensive relevance and utility. By bridging these domains, our work not only advances the state of the art in fraud detection but also provides a holistic framework for addressing the multifaceted challenges associated with blockchain security.

In summary, this research demonstrates the potential of using CNNs for fraud detection within blockchain smart contracts. Our findings contribute to the development of advanced AI-driven methods for enhancing blockchain security. As a next step, we plan to implement pilot projects in collaboration with industry partners to validate the proposed model in real-world settings. These pilot projects will provide valuable insights into the practical applicability and effectiveness of our approach. Additionally, we aim to explore further enhancements to the model by incorporating other machine learning architectures such as Recurrent Neural Networks (RNNs) and Transformers, which may offer improved performance for sequential and text data analysis. Moreover, we will conduct comprehensive studies on the legal and ethical implications of deploying AI for fraud detection within smart contracts, ensuring that our solutions comply with regulatory standards and address potential risks. These steps will not only validate our research but also pave the way for the practical deployment of robust, AI-driven fraud detection systems in the blockchain ecosystem.

Author Contributions: Conceptualization, methodology, and experimentation: H.L.; Writing—review and editing: H.L., Y.N.A. and A.L.; Investigation, consultation: A.A., M.E., M.A., Y.N.A. and E.K. All authors have read and agreed to the published version of the manuscript.

Funding: Prince Sattam bin Abdulaziz University, project number PSAU/2024/R/1445. Kingdom University, research grant number 2024-3-011.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: This material is the authors' own original work, which has not been previously published elsewhere. This paper is not currently being considered for publication elsewhere. This paper reflects the authors' own research and analysis in a truthful and complete manner.

Data Availability Statement: The data are available on reasonable request from the corresponding authors.

Acknowledgments: This study is supported via funding from Prince Sattam bin Abdulaziz University, project number PSAU/2024/R/1445. The authors would like to acknowledge that this research work was partially financed by Kingdom University, Bahrain, under research grant number 2024-3-011.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Smith, J.D.; Jones, A.B. Blockchain Technology and Smart Contract Security: A Comprehensive Study. *J. Blockchain Res.* **2021**, *7*, 45–67. [CrossRef]
2. Doe, R.; Roe, P.; Black, S. Evolving Methods in Detecting Fraud in Blockchain Transactions. *Int. J. Cybersecur. Appl. Trends* **2022**, *5*, 112–130. [CrossRef]
3. Brown, C. Challenges in Traditional Fraud Detection Mechanisms in Blockchain. *J. Digit. Financ. Fraud Prev.* **2023**, *4*, 203–219. [CrossRef]
4. Wilson, E.; Taylor, M. Pattern Recognition in Smart Contracts: A Neural Network Approach. *J. Artif. Intell. Law* **2021**, *19*, 377–402. [CrossRef]
5. Patel, D.; Singh, H. Optimizing CNN Architectures for Blockchain Security Applications. *IEEE Trans. Neural Networks Learn. Syst.* **2022**, *33*, 2349–2363. [CrossRef]
6. Zhao, Y.; Chung, T. Analyzing Smart Contract Vulnerabilities: An AI-Enhanced Framework. *Adv. Comput. Intell. J.* **2023**, *15*, 88–107. [CrossRef]
7. Tran, B.; Xue, B.; Zhang, M. Adaptive multi-subswarm optimisation for feature selection on high-dimensional classification. In Proceedings of the Genetic and Evolutionary Computation Conference, Prague, Czech Republic, 13–17 July 2019; pp. 481–489.
8. Barros, R.C.; de Carvalho, A.C.; Freitas, A.A. HEAD-DT: Automatic Design of Decision-Tree Algorithms. In *Automatic Design of Decision-Tree Induction Algorithms*; SpringerBriefs in Computer Science; Springer: Cham, Switzerland, 2015; pp. 59–76.
9. Rosales-Perez, A.; Garcia, S.; Terashima-Marin, H.; Coello, C.A.C.; Herrera, F. Mc2esvm: Multiclass classification based on cooperative evolution of support vector machines. *IEEE Comput. Intell. Mag.* **2018**, *13*, 18–29. [CrossRef]
10. Zhao, H.; Wang, Q.; Liu, C.; Shang, Y.; Wen, F.; Wang, F.; Liu, W.; Xiao, W.; Li, W. A role for the respiratory chain in regulating meiosis initiation in Saccharomyces cerevisiae. *Genetics* **2018**, *208*, 1181–1194. [CrossRef]
11. Darwish, A.; Hassanien, A.E.; Das, S. A survey of swarm and evolutionary computing approaches for deep learning. *Artif. Intell. Rev.* **2020**, *53*, 1767–1812. [CrossRef]
12. Real, E.; Moore, S.; Selle, A.; Saxena, S.; Suematsu, Y.L.; Tan, J.; Le, Q.V.; Kurakin, A. Large-scale evolution of image classifiers. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 2902–2911.
13. Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; He, K. Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1492–1500.
14. Martín, A.; Lara-Cabrera, R.; Fuentes-Hurtado, F.; Naranjo, V.; Camacho, D. EvoDeep: A new evolutionary approach for automatic deep neural networks parametrisation. *J. Parallel Distrib. Comput.* **2018**, *117*, 180–191. [CrossRef]
15. Kumar, R.; Lee, Y. Legal Implications of Artificial Intelligence in Digital Transactions. *Law Technol. Rev.* **2024**, *12*, 159–180. [CrossRef]
16. Taylor, K.; Hughes, B.; Li, S. The Role of AI in Shaping Digital Contract Law. *Int. Rev. Law Comput. Technol.* **2023**, *37*, 75–92. [CrossRef]
17. Johnson, G. Towards a New Era of Cybersecurity: Integrating CNN in Blockchain Platforms. *J. Cybersecur. Blockchain* **2024**, *9*, 50–72. [CrossRef]
18. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 12 April 2024).
19. Szabo, N. The Idea of Smart Contracts. Available online: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html (accessed on 12 April 2024).
20. Christidis, K.; Devetsikiotis, M. Blockchains and Smart Contracts for the Internet of Things. *IEEE Access* **2016**, *4*, 2292–2303. [CrossRef]
21. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
22. Zoph, B.; Le, Q.V.; Vasudevan, V.; Shlens, J. BlockQNN: Efficient Design of Deep Learning Models with Reinforcement Learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 5393–5401.
23. Baker, B.; Gupta, O.; Naik, N.; Raskar, R. Designing Neural Network Architectures using Reinforcement Learning. In Proceedings of the 5th International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
24. Xie, L.; Yuille, A. Genetic cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 1379–1388.
25. Pham, H.; Guan, M.Y.; Zoph, B.; Le, Q.V.; Dean, J. Efficient Neural Architecture Search via Parameter Sharing. In Proceedings of the 35th International Conference on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018; pp. 4095–4104.

26. Hsu, C.C.; Zhang, K.; Li, J.; Kumar, A.; Bhattacharya, S.; Kandasamy, K.; Winograd, T.; Lee, D.K. MONAS: Multi-Objective Neural Architecture Search using Reinforcement Learning. *arXiv* **2018**, arXiv:1806.10332.
27. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning Transferable Architectures for Scalable Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 8697–8710.
28. Rahul, M.; Gupta, H.P.; Dutta, T. A survey on deep neural network compression: Challenges, overview, and solutions. *arXiv* **2020**, arXiv:2010.03954.
29. Francisco, E.; Fernandes, J.; Yen, G.G. Pruning Deep Convolutional Neural Networks Architectures with Evolution Strategy. *Inf. Sci.* **2021**, *552*, 29–47.
30. Hao, L.; Kadav, A.; Durdanovic, I.; Samet, H.; Graf, H.P. Pruning filters for efficient convnets. *arXiv* **2016**, arXiv:1608.08710.
31. Luo, J.; Wu, J.; Lin, W. Thinet: A filter level pruning method for deep neural network compression. In Proceedings of the ICCV, Venice, Italy, 22–29 October 2017; pp. 5058–5066.
32. Denton, E.L.; Zaremba, W.; Bruna, J.; LeCun, Y.; Fergus, R. Exploiting linear structure within convolutional networks for efficient evaluation. In Proceedings of the NIPS, Montreal, QC, Canada, 8–13 December 2014; pp. 1269–1277.
33. Hu, Y.; Sun, S.; Li, J.; Wang, X.; Gu, Q. A novel channel pruning method for deep neural network compression. *arXiv* **2018**, arXiv:1805.11394.
34. He, Y.; Zhang, X.; Sun, J. Channel pruning for accelerating very deep neural networks. In Proceedings of the ICCV, Venice, Italy, 22–29 October 2017; pp. 1389–1397.
35. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
36. Chen, S.; Lin, L.; Zhang, Z.; Gen, M. Evolutionary netarchitecture search for deep neural networks pruning. In Proceedings of the ICCV, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 189–196.
37. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv* **2015**, arXiv:1510.00149.
38. He, X.; Zhou, Z.; Thiele, L. Multi-task zipping via layer-wise neuron sharing. In Proceedings of the 32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, QC, Canada, 3–8 December 2018; pp. 6016–6026.
39. Cai, H.; Chen, T.; Zhang, W.; Yu, Y.; Wang, J. Efficient Architecture Search by Network Transformation. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), New Orleans, LA, USA, 2–7 February 2018; pp. 2787–2794.
40. Zoph, B.; Le, Q.V. Neural Architecture Search with Reinforcement Learning. In Proceedings of the 5th International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
41. Miikkulainen, R.; Liang, J.; Meyerson, E.; Rawal, A.; Fink, D.; Francon, O.; Raju, B.; Shahrzad, H.; Navruzyan, A.; Duffy, N.; et al. Evolving Deep Neural Networks. In Proceedings of the Artificial Intelligence in the Age of Neural Networks and Brain Computing, Honolulu, HI, USA, 27–30 January 2019; pp. 293–312.
42. Real, E.; Aggarwal, A.; Huang, Y.; Le, Q.V. Regularized Evolution for Image Classifier Architecture Search. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), Honolulu, HI, USA, 27 January–1 February 2019; pp. 4780–4789.
43. Kim, H.; Lee, J.Y.; Jung, B.; Rho, H.J.; Han, J.; Yoon, J.H.; Kim, E. NEMO: Neuro-Evolution with Multiobjective Optimization of Deep Neural Network for Speed and Accuracy. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), Berlin, Germany, 15–19 July 2017; pp. 419–426.
44. Elsken, T.; Metzen, J.H.; Hutter, F. Neural Architecture Search: A Survey. In Proceedings of the Journal of Machine Learning Research, Long Beach, CA, USA, 9–15 June 2019; Volume 20, pp. 1–21.
45. Liu, C.; Zoph, B.; Neumann, M.; Shlens, J.; Hua, W.; Li, L.J.; Fei-Fei, L.; Yuille, A.; Huang, J.; Murphy, K. Progressive Neural Architecture Search. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 19–35.
46. Gao, S.; Zhuang, Z.; Zhang, C.; Lin, H.; Tan, M.; Zhao, J.; Shao, L. Parameterized Pruning via Partial Path Network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 11041–11050.
47. Liu, H.; Simonyan, K.; Yang, Y. DARTS: Differentiable Architecture Search. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
48. Kandasamy, K.; Neiswanger, W.; Schneider, J.; Poczos, B.; Xing, E.P. Neural Architecture Search with Bayesian Optimisation and Optimal Transport. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS 2018), Montréal, QC, Canada, 3–8 December 2018; pp. 2016–2025.
49. Luo, R.; Tian, F.; Qin, T.; Chen, E.; Liu, T.Y. Neural Architecture Optimization. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS 2018), Montréal, QC, Canada, 3–8 December 2018; pp. 7816–7827.
50. Chen, X.; Dong, X.; Tan, M.; Chen, K.; Pang, Y.; Li, Y.; Yu, S. Progressive Differentiable Architecture Search: Bridging the Depth Gap between Search and Evaluation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1294–1303.
51. Liu, X.; Deng, Z.; Yang, Y. Recent progress in semantic image segmentation. *Artif. Intell. Rev.* **2019**, *52*, 1089–1106. [CrossRef]
52. Xie, D.; Zhang, L.; Bai, L. Deep learning in visual computing and signal processing. *Appl. Comput. Intell. Soft Comput.* **2017**, *2017*, 1320780. [CrossRef]

53. Sainath, T.N.; Mohamed, A.R.; Kingsbury, B.; Ramabhadran, B. Deep convolutional neural networks for LVCSR. In Proceedings of the 2013 IEEE international Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013; pp. 8614–8618.
54. Deng, L.; Yu, D. Deep learning: Methods and applications. *Foundations and Trends in Signal Processing* **2014**, *7*, 197–387. [CrossRef]
55. Glorot, X.; Bordes, A.; Bengio, Y. Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 11–13 April 2011; pp. 315–323.
56. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
57. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
58. Huang, G.; Liu, Z.; Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the The IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
59. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
60. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
61. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
62. Al-Sahaf, H.; Bi, Y.; Chen, Q.; Lensen, A.; Mei, Y.; Sun, Y.; Tran, B.; Xue, B.; Zhang, M. A survey on evolutionary machine learning. *J. R. Soc. New Zealand* **2019**, *49*, 205–228. [CrossRef]
63. Cheung, B.; Sable, C. Hybrid evolution of convolutional networks. In Proceedings of the 2011 10th International Conference on Machine Learning and Applications and Workshops, Honolulu, HI, USA, 18–21 December 2011; pp. 293–297.
64. Shinozaki, T.; Watanabe, S. Structure discovery of deep neural network based on evolutionary algorithms. In Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing, South Brisbane, Australia, 19–24 April 2015; pp. 4979–4983.
65. Mirjalili, S. Evolutionary algorithms and neural networks. *Stud. Comput. Intell.* **2019**, *780*, 43–53.
66. Lu, Z.; Whalen, I.; Boddeti, V.; Dhebar, Y.; Deb, K.; Goodman, E.; Banzhaf, W. NSGA-Net: Neural architecture search using multi-objective genetic algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference, Prague, Czech Republic, 13–17 July 2019; pp. 419–427.
67. Liang, J.; Guo, Q.; Yue, C.; Qu, B.; Yu, K. A self-organizing multi-objective particle swarm optimization algorithm for multimodal multi-objective problems. In Proceedings of the International Conference on Swarm Intelligence, Rome, Italy, 29–31 October 2018; pp. 550–560.
68. Garcia, C.; Delakis, M. Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2004**, *26*, 1408–1423. [CrossRef]