國立臺北科技大學

資訊系碩士班

碩士學位論文

Use Efficiency-based Genetic Programming to Create Loss Function for Image Classification Tasks

研究生：周玟萱

指導教授：陳香君博士

中華民國一百一十四年五月

國立臺北科技大學

資訊系碩士班

碩士學位論文

Use Efficiency-based Genetic Programming to Create Loss Function for Image Classification Tasks

研究生：周玟萱

指導教授：陳香君博士

中華民國一百一十四年五月

「學位論文口試委員會審定書」掃描檔

審定書填寫方式以系所規定為準，但檢附在電子論文內的掃描檔須具備以下條件：

1. 含指導教授、口試委員及系所主管的完整簽名。
2. 口試委員人數正確，碩士口試委員至少 3 人、博士口試委員至少 5 人。
3. 若此頁有論文題目，題目應和書背、封面、書名頁、摘要頁的題目相符。
4. 此頁有無浮水印皆可。

審定書不用頁碼

Abstract

Title: Use Efficiency-based Genetic Programming to Create Loss Function for Image Classification Tasks

Pages: (Fill it)

School: National Taipei University of Technology

Department: Computer Science and Information Engineering

Time: May, 2025

Degree: Master of Science

Researcher: Tzu-Yu Chou

Advisor: Shiang-Jiun Chen, Ph.D.

Keyword: Image Classification, Genetic Programming, Loss Function, Deep Learning

In the recent, with the rapid rise of deep learning, image classification models have quickly become one of the most popular and widely known models. When training such models, the steps are broadly divided into the following: preparing image datasets, dividing them into training, validation and testing sets as needed, training the model, evaluating the model, and repeating these steps until the ideal result is met or the computational resources are exhausted.

A crucial function during the process of training a model is called the loss function, which calculate the difference between the predicted values and the ground-truth values. The results of the loss function can significantly influence the effectiveness of the model's training because it simply decide the direction of the adjustment to the model. However, designing a loss function often requires the assistance of experts in the related field, leading to a resource-intensive design process. Recent research has proposed using Genetic Programming (GP) to generate loss functions to avoid the necessity of hiring numerous domain experts for assistance. Nevertheless, using GP typically results in decreased computational efficiency. This paper aims to improve the method of using GP to generate loss functions by modifying certain genetic operations and introducing the concept of tournament selection.

這邊要加結果

Acknowledgements

所有對於研究提供協助之人或機構，作者都可在誌謝中表達感謝之意。

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Driven by the rapid advancement of digital transformation and smart infrastructure, the Internet of Things (IoT) has become a cornerstone of next-generation information technology. By integrating sensors, embedded devices, communication modules, and platform software, IoT enables physical-world objects to connect to networks in real time and generate massive volumes of data. This data facilitates a wide array of applications—such as smart manufacturing, intelligent transportation, remote healthcare, and smart homes—creating significant economic and societal value [1]. However, as the number of connected devices surges and deployment scenarios diversify, IoT systems face unprecedented cybersecurity challenges. These devices are often resource-constrained, infrequently updated, and difficult for users to manage. Combined with unencrypted data transmission and lack of monitoring mechanisms, they become prime targets for cyber intrusions and attacks. Therefore, how to effectively identify abnormal behaviors and hidden threats within IoT networks has become a pressing focus in the field of cybersecurity. As IoT devices become increasingly pervasive—from smart homes and wearable technologies to industrial control systems—the associated cybersecurity risks continue to grow. Due to limited resources and a lack of standardized design, these devices are frequent targets of cyberattacks. Most existing intrusion detection technologies cannot adapt quickly to emerging threats or rely heavily on large-scale labeled datasets and computationally intensive models, which limit their deployment at the network edge. This research is thus motivated by the need to design an efficient, scalable, and interpretable anomaly detection architecture that leverages semantic embedding and statistical learning to meet the dual requirements of real-time performance and high detection accuracy in IoT contexts. 其中，現今的方法會用 Word2Vec 或 Transformer 來達成這件事情，，但這個方式有著甚麼的缺陷，因此這個 S2GE-NIDS 方法被提出，藉由 MLP 以及改善 hash function 易碰撞的缺點，所以結合 link list 這個方法改善這個碰撞機率，，改善了 Hashmurmur 的碰撞可能，

To address the above problems, existing research actively explores the application of deep learning technology in network anomaly detection, especially in combination with semantic embedding, as well as complex associations and contexts in network traffic data [3-4]. However, these methods usually rely on resource-intensive architectures, such as recursive or attention-

based neural networks [5], or require pre-defined vocabularies and tedious feature engineering, which are often difficult to scale and have limited deployment flexibility in high-throughput real-time application scenarios [6]. 藉由導入語意向量，我們提出了，它的優點是很均勻分布，但她還是有些缺點，在資源小的地方容易碰裝到同一個，所以我們導入 link list 這個方法改善這個碰撞機率，To this end, this paper proposes an efficient anomaly detection framework with semantic vectors, combining hash-based embedding coding with a lightweight multi-layer perceptron (MLP) model. Specifically, we convert each packet of data into semantic tokens, and because it is hash-coded, the corresponding vector is used through the embedding table, which effectively eliminates the need to maintain a large-scale vocabulary. After that, the embedding vectors of all fields are concatenated (flattened) into a fixed-length one-dimensional semantic vector, and sent to the MLP for encoding, projected near a single semantic center. If the vector of a piece of data deviates from the center by more than the preset threshold, it is determined to be a potential anomaly, which effectively improves the ability to identify isolated anomalies [7]. 我加入了 Link list 來減少 hashmurmur 的碰撞機率，Compared with existing research, the main contribution of this study is to reduce the burden of vocabulary maintenance. Unlike methods that require maintaining a complete field/value table, this architecture uses a hash function to quickly map all combinations, combining high performance and flexibility. Secondly, it also demonstrates lightweight design and real-time reasoning capabilities. This framework only requires simple table lookup and MLP encoding, and the amount of computation and memory consumption are much lower than traditional anomaly detection. In addition, because the core design combines semantic vectors with statistical center loss, statistical methods are used to more effectively detect and isolate anomalies, and the combination of Mahalanobis distance improves the accuracy of detection in high dimensions. The structure of this paper is as follows: Chapter 2 is the relevant background knowledge about S2GE-NIDS (Structured Semantics and Generation Embedded Network Intrusion Detection System). Chapter 3 introduces the architecture and methodology of the proposed S2GE-NIDS framework, presenting each module and its rationale in detail. Chapter 4 presents the experimental setup, evaluation metrics, and results on two benchmark datasets, as well as interpretability demonstrations. Chapter 5 summarizes the main findings, limitations, and directions for future research.

Chapter 2 Related Work

In the following sections, we will introduce the concept of metaheuristics [1, 2] . Then, we will discuss the importance of loss functions and their role in deep learning [3] models. Finally, we will explore image classification [4] and its real-life applications.

2.1 Metaheuristic

In the following section, we will discuss metaheuristic and some of the most famous metaheuristic algorithms include Evolving Algorithm (EA) [5], Genetic Algorithm (GA) [6], and Genetic Programming (GP) [7]. After that, we will introduce related research in the GP domain.

2.1.1 Overview of Metaheuristic

Metaheuristic [8] was first proposed by Glover. It can be regarded as a high-level procedure used to find solutions to optimization problems or other issues that conventional algorithms cannot solve. Within metaheuristics, an important concept is that these procedures must find a potentially optimal solution under reasonable computational costs or insufficient information. In this paper, we will primarily focus on nature-inspired metaheuristics [1]. In these methods, a common approach is to use a population-based strategy to find the optimal solution. Within a population, each individual typically represents a potential solution. We perform various operations on the individuals within the population to achieve our goal of approximating the optimal solution. A subset of specialized algorithms falls under population-based methods [9], commonly referred to as evolving algorithms (EAs). Commonly used methods within EAs include genetic algorithms (GA), genetic programming (GP) , evolutionary programming (EP) [10], and differential evolution (DE) [11]. The relationship between Metaheuristic, EA, GA, GP, EP and DE is illustrated in the figure 2.1. We will briefly discuss EAs and GAs, and then we will focus primarily on GPs in the following paragraph.

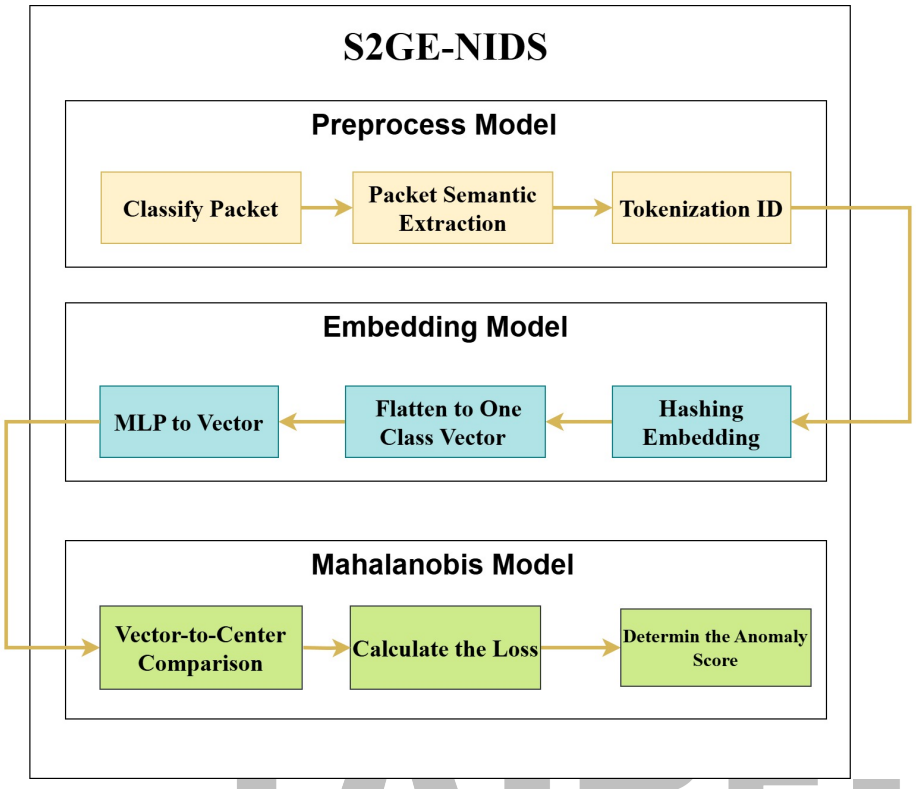

Figure 2.1 Relationship between metaheuristic, EA, GA, GP, EP, and DE

2.1.2 Evolving Algorithms

EAs can be described as algorithms inspired by the concept of ”survival of the fittest [12].” These algorithms often take cues from natural evolutionary mechanisms to create algorithms that mimic animal behavior in the search for optimal solutions. Common examples include Ant Colony Optimization (ACO) [13], GA, and Particle Swarm Optimization (PSO) [14]. The common EA procedure is shown in figure 2.2. In this type of algorithm, the typical approach is to initialize a population, which consists of several individuals. Each individual represents a potential optimal solution. After initializing the population, we usually set a number to represent how many generations we want this population to evolve. Before the start of each generation, a special function is used to calculate the fitness value of each individual, determining their level of excellence. Next, we perform selection, mutation, and crossover on the population. Selection involves choosing individuals based on their fitness to reproduce the next generation. Mutation generally refers to making random changes to an individual, while crossover combines the characteristics of two individuals to create the next generation. These steps are repeated until the predefined number of

generations is reached, or the individuals fail to achieve the expected results, leading to a forced stop. Ultimately, the best individual in the population is obtained as the solution.

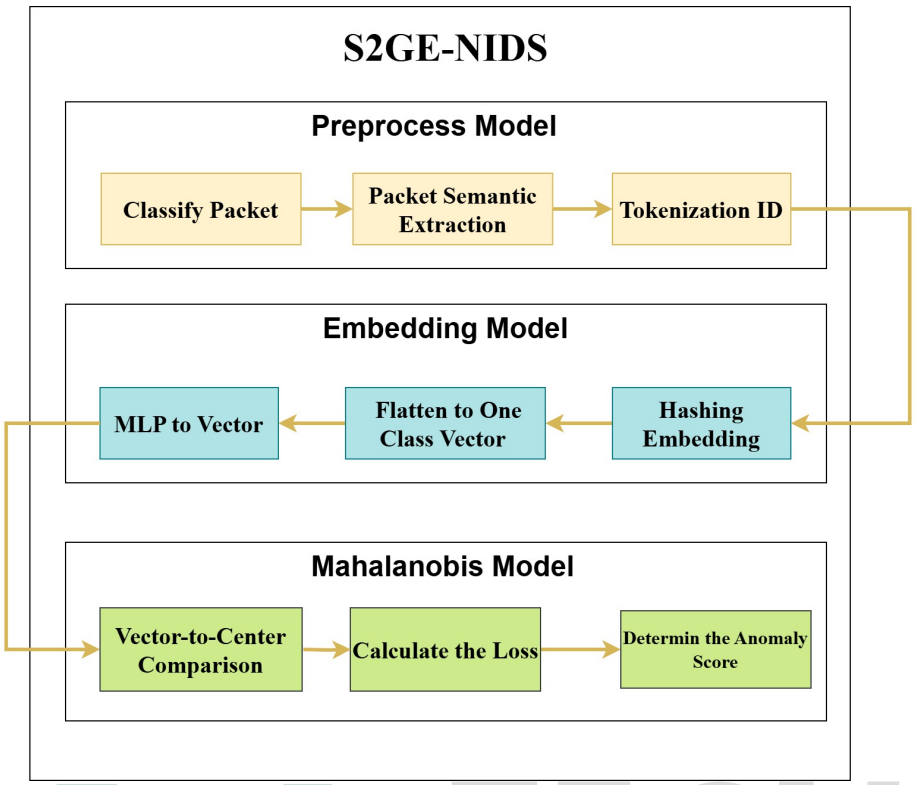

Figure 2.2 Flowchart for EA

2.1.3 Genetic Algorithms & Genetic Programming

GA is considered a type of EA, utilizing the representation of each solution in the form of chromosomes to perform selection, mutation, and crossover. GP is regarded as a special case of GAs, but due to its versatility and practicality, it is seen as a reliable solution. Unlike GAs, GP typically uses a more robust encoding method to represent chromosomes. For instance, GAs often use strings to represent chromosomes, which can present potential issues that need to be addressed, such as the priority of operations for each symbol or the validity of the equation itself. As shown in the figure 2.3, if we want to express the function $(x + y)\div(5 * x)$ in GA, the chromosome representation of this function will be $x + y \div 5 * x$. Without the proper parentheses, it can easily cause confusion. In contrast, GP usually represents chromosomes as trees, a method that can employ predefined traversal techniques to avoid these issues. As illustrated in the figure 2.3, the same function represented by GA can be expressed through a tree structure in GP. By specifying

the use of inorder traversal, we can obtain the same function. It is worth noting that in GAs and GP, we can leverage their inherent ability to evolve automatically to find a reasonable optimal solution without having an in-depth understanding of the knowledge domain related to the problem we aim to solve. In the following paragraphs, we will introduce related research.

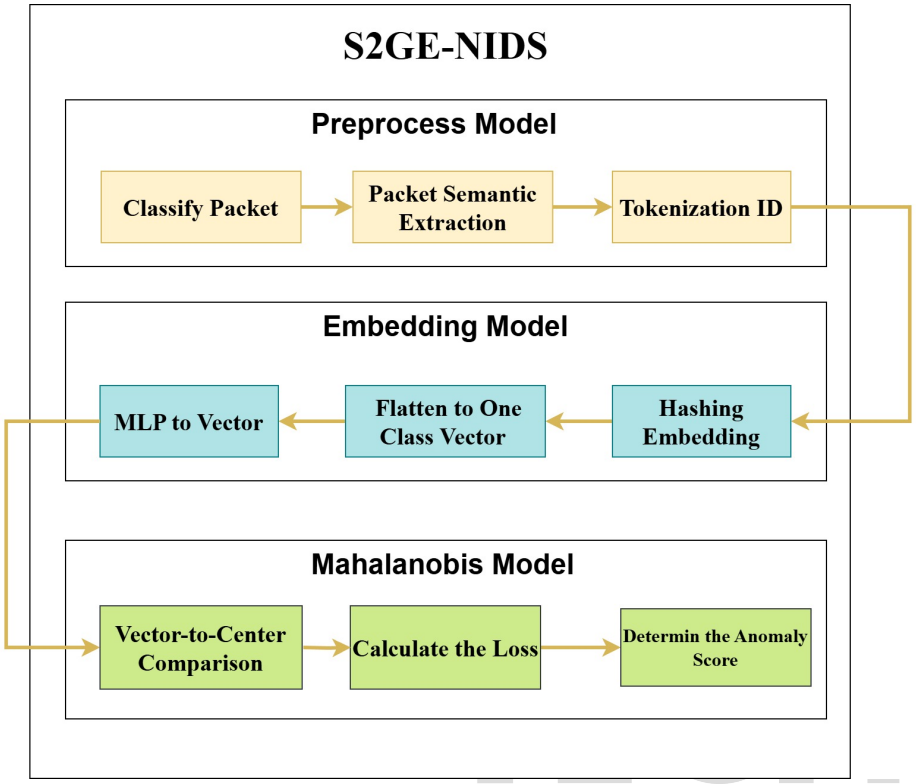

Figure 2.3 Different chromosome representation between GA and GP

In [15], Co-Reyes et al. proposed a meta-learning reinforcement learning algorithm. They used computational graphs to represent algorithms. By doing so, algorithms could be identified, calculated, and optimized through Reinforcement Learning (RL) [16]. Notably, in this study, algorithms were represented as directed acyclic graphs (DAG) [17] of nodes. Within the DAG , all nodes were classified into three categories: input nodes, parameter nodes, and operation nodes. Once the algorithms were represented as DAGs, they could be placed into RL for training and evaluation. In this proposed algorithm, they employed the concept of regularized evolution [18] to evolve a population formed by several randomized and known algorithms. The method was as follows: initialize the population with algorithms, evaluate each algorithm in the population and record their performance, then in a loop, repeatedly use a sample tournament [19] to select algorithms, perform mutations on the algorithms by the mutator they designed, and evaluate them

again until the loop ends. During the evaluation process, they trained and assessed the algorithms using RL, continuously testing the performance of each algorithm in different training environments. They also utilized normalized training performance to avoid numerical biases caused by varying environments. Through this approach, the study successfully created two algorithms, named DQNClipped and DQNReg, which outperformed classical control tasks.

In [20], Akhmedova and Körber proposed a genetic programming-based method to find a better function for the image classification training task. They encoded the loss functions into trees, with each tree considered an individual. All individuals were aggregated into a population. In this study, the population was evolved across multiple generations. Before the start of each generation, all individuals were evaluated. Each individual had a probability of undergoing crossover with an individual from a special external archive to create a new individual; otherwise, it would perform the crossover with another individual. Following this, two probabilistic decisions were made. If successful, the new individual could perform subtree mutation or one-point mutation, respectively. At the end of each generation, the fitness value of all individuals was re-evaluated. A certain number of low-scoring individuals were eliminated to the special external archive mentioned earlier, maintaining the stability of the population's size. After these steps, a new generation began, continuing until a predefined number of generations was reached. By employing this method, this study successfully evolved an outstanding individual within the population, creating a function that could train an image classification model more effectively compared to Cross Entropy (CE) [21].

2.2 Loss Function

In this section, we will first introduce deep learning model and explain how does it work. Secondly, we will discuss the importance of loss function and present some related researches.

2.2.1 Deep Learning Model

In recent years, due to the significant increase in the computational speed of GPUs, training deep learning models to solve a wide variety of problems has become widely regarded as a fea-

sible and popular solution. The process of training a deep learning model is often divided into several steps: collecting the data needed to train the model, and adding ground truth values to the data according to the model's requirements. Ground truth can be thought of as the ideal answers we hope to achieve after the model's computation. Once the data collection is complete, these data sets are collectively referred to as the dataset. Typically, the dataset is divided into training sets, validation sets, and testing sets. Data from the training set are used to train the model, the validation set is used to evaluate the model's effectiveness after each training loop, and the testing set is used to calculate the model's final score and determine whether the model successfully achieves its designed purpose. It is worth noting that the testing set data should not be seen during training and validation phases. After dividing the dataset, we decide on the model's architecture. Common architectures include Fully Connected Networks (FCN) [22] and Convolutional Neural Networks (CNN) [23], etc. Then the training process can begin. During training, we compare the model's output with the ground truth of the training data and use a loss function to calculate the difference between the output and the ground truth. This guides the adjustment of the model's parameters. Therefore, the loss function significantly determines the effectiveness of model training, and this aspect will be explained further later. After adjusting the model, the next round of training begins, continuing until the training is deemed ineffective or a predetermined maximum number of iterations is reached.

2.2.2 Loss Function In Deep Learning Model

In section 2.2.1, we discussed the role of loss functions in deep learning models. Here, we would like to introduce some commonly used loss functions along with their advantages and disadvantages. Mean-Square Error (MSE) [24] is a loss function used for solving regression problems. It calculates the difference between the actual and predicted values, squares them, and then takes the average of these squared differences to obtain the final loss value. However, a notable disadvantage of MSE is that it amplifies the impact of outliers exponentially. Cross-Entropy (CE), unlike MSE, leverages probability concepts to help compute the error in classification tasks. In other words, CE measures the difference between the predicted probability and the true probability to calculate the error. CE can also be adapted to different types of classification tasks to better

suit their requirements, with common variants including Binary CE Loss [25] and Multiclass CE Loss [26].

From the above paragraphs, it seems that using a single type of loss function to train all models is considered impractical. Therefore, researchers design different loss functions based on specific needs to calculate more appropriate loss values for the model. As the introduction above indicates, designing a loss function requires an in-depth understanding of the model and a clear grasp of how to define and calculate the loss value. Consequently, designing a loss function, similar to adjusting hyperparameters or model architecture, is considered as a challenging task that needs a deep understanding of the domain for effective design and adjustment.

In [27], Gonzalez and Miikkulainen proposed a meta-learning approach to create a loss function called Genetic Loss-function Optimization (GLO). Through their research, the authors experimentally found a loss function named de novo, created by GLO, outperforms the well-known CE loss in standard image classification tasks. Additionally, because GLO enables training to be completed in fewer steps, this method also enhances the speed and efficiency of the training process.

In [20], Akhmedova and Körber utilize GP to create a loss function that outperforms CE in image classification. The method described in the paper led to the creation of a loss function named Next Generation Loss (NGL). When trained with the Inception model, NGL outperforms CE on datasets such as CIFAR-10 [28], CIFAR-100 [29], and Fashion-MNIST [30]. Additionally, it demonstrates excellent performance on larger datasets like [31]. Furthermore, NGL is also effective in improving model performance in segmentation downstream tasks.

From the above paragraphs, we can observe that numerous studies indicate that in image classification tasks, if a well-designed loss function is developed, it has the potential to outperform the most famous and widely regarded as the most effective CE loss function. However, as mentioned earlier, designing a loss function is usually considered a resource-intensive task. Therefore, the two studies discussed above have begun to use genetic programming to enable computers to automatically design a loss function that can effectively collaborate with a model in a specific domain and significantly improve the model's performance.

2.3 Image Classification

Image classification is a technique in computer vision that enables computers to identify what object present in an image. This technology is often used with image localization [32]. Image localization determines the location of objects within an image, using bounding box [33] to enclose the areas where the objects are found. Additionally, object detection [34] is a similar technique to image classification and image localization. Unlike image classification and image localization, which focus on identifying one label per image, object detection can identify multiple labels in a single image. The difference between image classification, image localization and object detection is illustrated in the figure 2.4. We can see that image classification is capable of recognizing which label the entire image belongs to, whereas image localization identifies which specific regions of the image belong to which labels. Object detection is a more complex technique, which can identify multiple labels and also mark the object by using bounding boxes.

In image classification, the most common types are binary classification [35], multiclass classification [36] and multilabel classification [37]. As the name implies, in binary classification, there are only two possible outcomes when identifying an image. For instance, if the model's labels are cat and dog, the prediction can only be either a cat or a dog. Conversely, multiclass classification means that the image can belong to multiple possible classes instead of two classes. Using the previous example, the prediction in multiclass classification could be a cat, dog, elephant, snake, or other different animals, rather than choosing between just two labels. It is important to note that the final prediction in both binary and multiclass classification will result in only one label. If an image needs to have multiple labels, a different type of image classification model is required, such as a multilabel model.

The process of training an image classification model is generally similar to what is described in section 2.2.1 on Deep Learning. In image classification, commonly used datasets include: ImageNet [38], CIFAR-10, CIFAR-100, and MNIST [39]. It's worth mentioning that we often perform preprocessing on images. Common preprocessing techniques include image cropping, image resizing, and image normalization. Image cropping is a technique that removes the unnecessary parts of the image. Through image cropping, the irrelevant parts of the image will be removed.

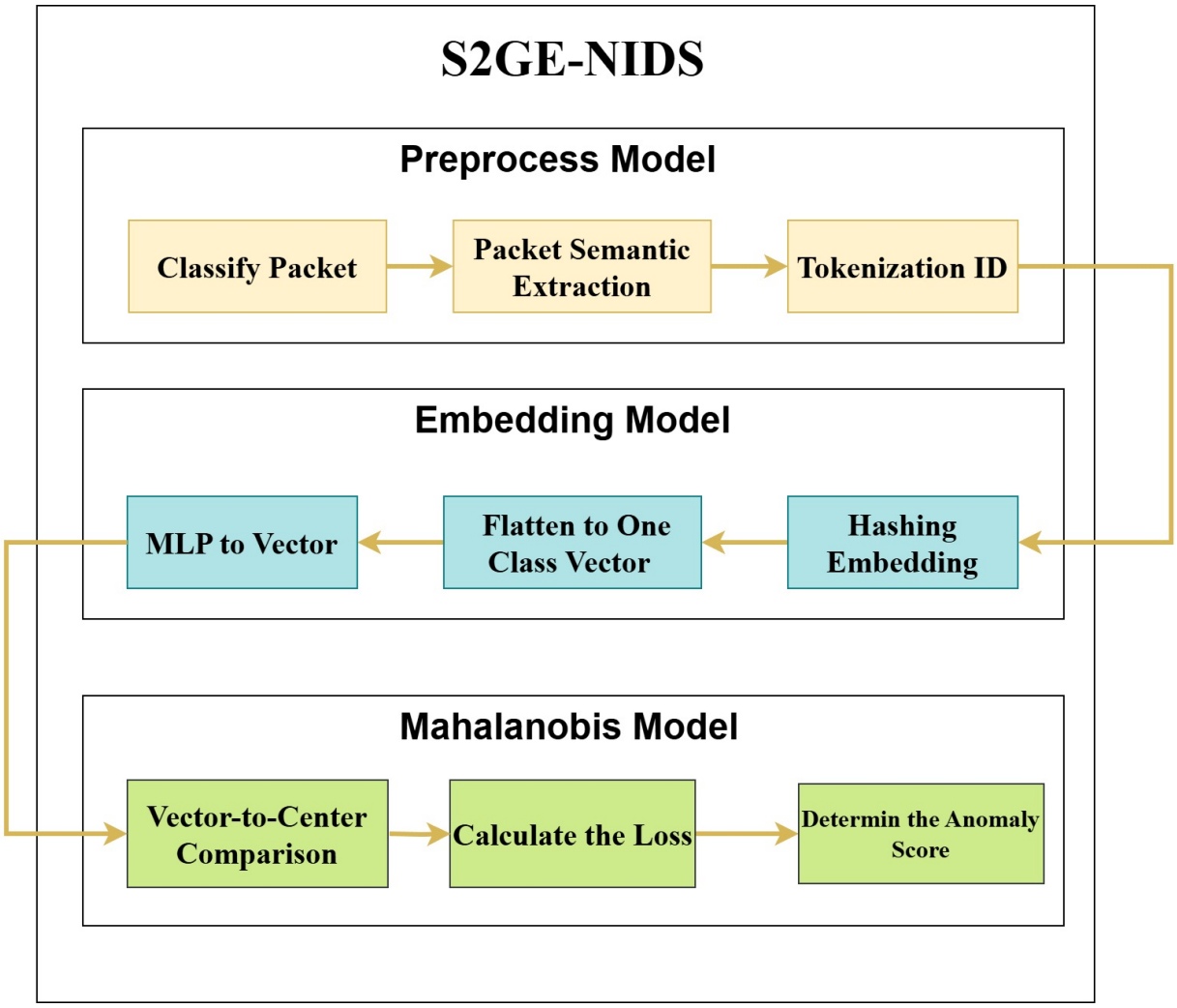

Figure 2.4 Difference between image classification, image localization and object detection

As shown in the figure 2.5, the pink area in the original picture is removed to prevent model from learning the useless information. Image resizing ensures that all images are of the same size, which helps improve computational speed. However, there is a potential risk of degrading the model's performance by using image resizing [40]. Hence, the size of the resized images is typically determined based on actual requirements. Normalizing images also helps the model learn from each image more effectively. After preprocessing, we usually use pretrained models instead of designing a new model architecture from scratch. This approach enhances computational efficiency and results. Common image classification models used include InceptionV3 [41], ResNet [42], and others.

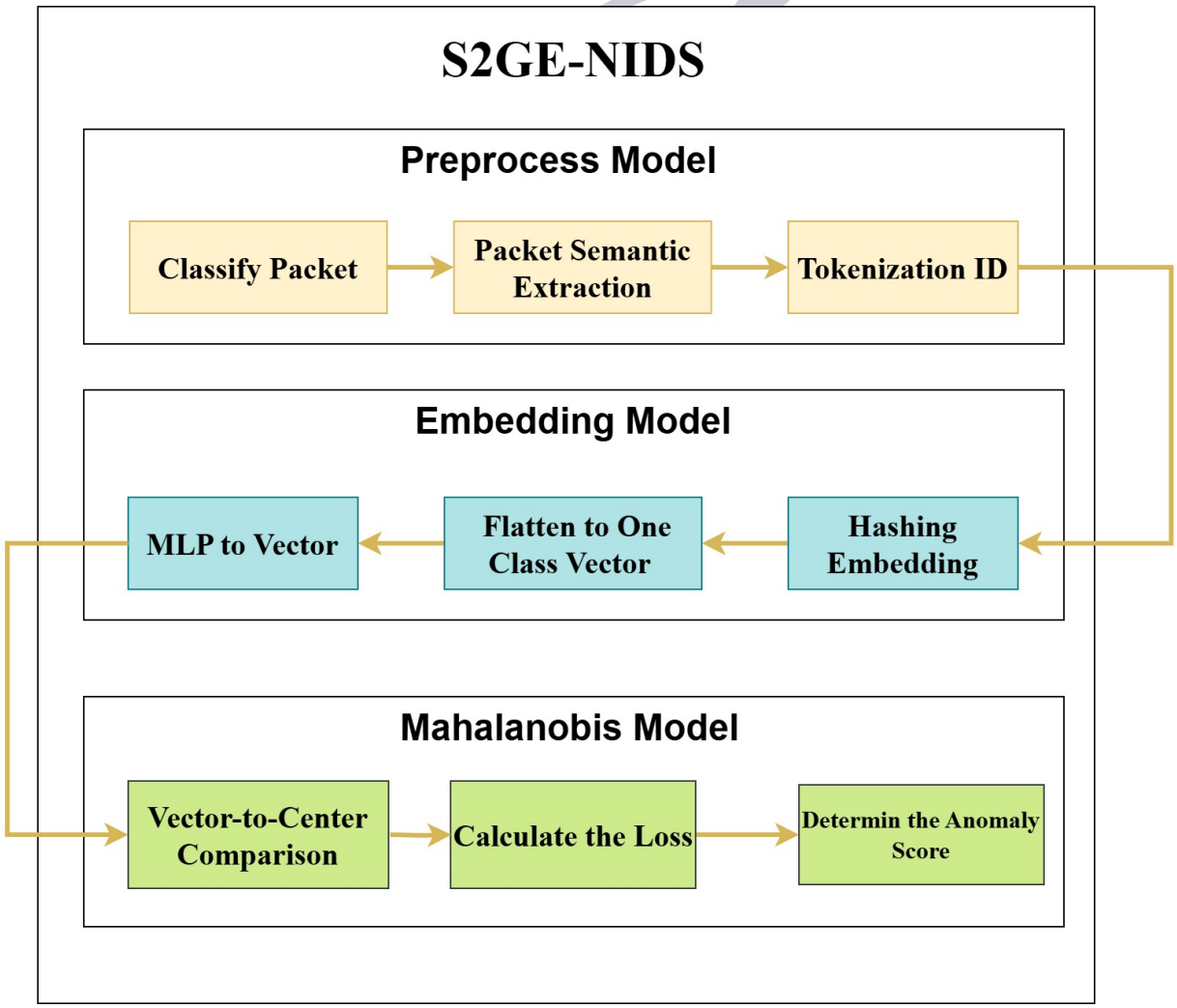

Figure 2.5 The process of image cropping

In the real world, image classification can be applied in many areas. For example, it can be

used in medical imaging to identify tumors in X-rays. It can also be employed in autonomous vehicles to quickly recognize objects surrounding the vehicle. Facial recognition is another application of image classification, used for company access control and as a method to unlock personal devices such as phones or computers.

Chapter 3 Methodology

In this session, we will introduce the S2GE-NIDS (structured semantics and generation embedded network intrusion detection system) architecture and details its operational workflow, clearly delineating each step from semantic tokenization through anomaly detection and decision-making processes.

3.1 Architecture

S2GE-NiDS is presented as Figure 3.1 including Preprocess model, embedding model, and Mahalanobis model.

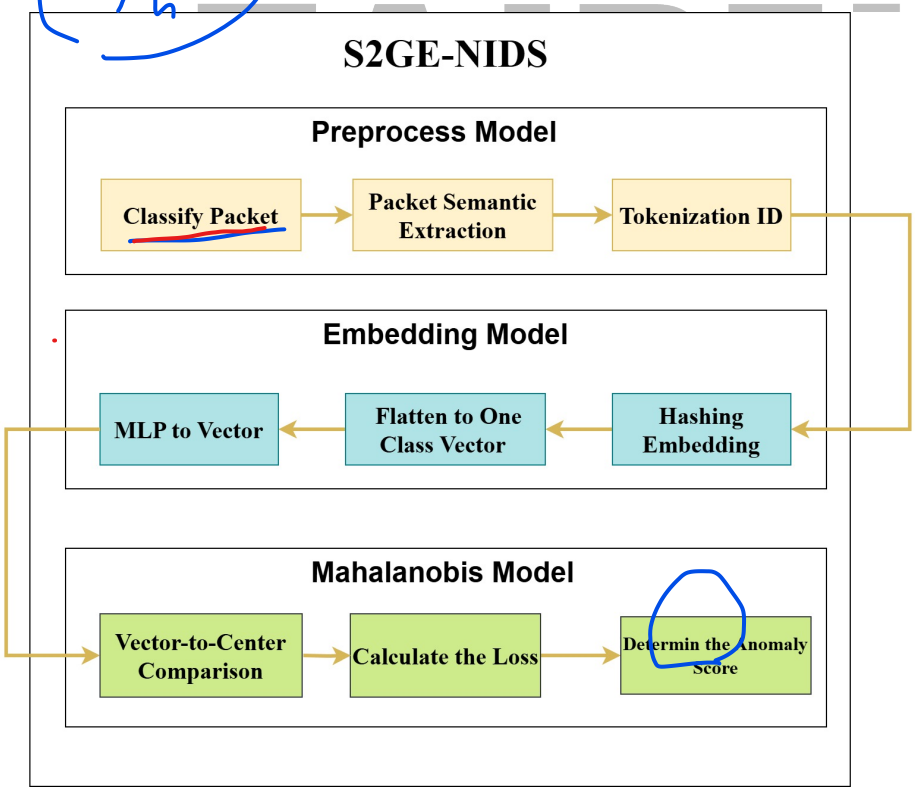

Figure 3.1 Architecture of S2GE-NIDS

In the preprocessing stage, features are first extracted from the data packets and converted into text and numeric tokens. To avoid high duplication of text features during embedding, the Embedding Model uses a non-encrypted MurmurHash3 function. This helps distribute the features evenly across the embedding table. To further reduce the chance of hash collisions, the

model applies a modulo operation to the hash value. The result is used as the position to insert the feature into the embedding table. This method lowers the possibility of different tokens being mapped to the same location, improving the accuracy and efficiency of feature embedding. Then, the vectors of different features are flattened into a single one-dimensional vector. This vector is passed through an MLP model to extract the semantic feature vector. Finally, in the Mahalanobis distance stage, the distance between each vector and the center point is calculated. Points that exceed a certain threshold are considered anomalies, and metrics such as the F1 score are computed.

這樣一遍系統偵測就算結束

3.2 Preprocess Model

In the preprocessing phase, we need to preprocess the data in order to obtain the semantic vectors for starting the experiments. This model is essential because it transforms raw input into a suitable format for further analysis and model training.

3.2.1 Classify Data Packet

The first step is to select and filter data files for subsequent analysis. In this work, network traffic data is collected and stored in the CSV (Comma-Separated Values) format, which is a widely used and flexible tabular data format for representing structured information. CSV files allow for efficient storage, easy parsing, and compatibility with mainstream data analysis tools such as pandas and NumPy in Python. During this stage, only relevant CSV files containing the required packet features are selected from the raw dataset, while irrelevant or malformed files are filtered out. This ensures that the input data for the anomaly detection system is both standardized and of high quality. By focusing on valid CSV files, we can automate subsequent steps such as feature extraction and semantic tokenization, reducing preprocessing errors and improving the reproducibility of experimental analysis

THus, the classifed data had reduce .... can be the input for the next pahse called feature extraction.

3.2.2 Packet Semantic Extraction

After the data is cleaned and organized, the first step is to extract meaningful features from the network packet data to better capture the characteristics of each packet. For example, we focus on key fields such as Destination Port, Flow Duration, and Protocol Type, which are widely used in previous studies to detect abnormal network behavior [42]. In practice, we use Python tools to read each CSV file and select these important features as the main input for the S2GE-NIDS model. By focusing only on these key values, we can make the data cleaner and easier for the anomaly detection system to use.

3.2.3 Tokenization ID

After extracting the features mentioned above, the next step is to perform tokenization. Each data entry contains multiple fields, such as Port, Protocol, and SrcIP. Tokenization is done by combining the name of each field with its corresponding value to form a string. For example, each token follows the format "field name + field value"

3.3 Embedding Model

We use a simple hash function called MurmurHash3 to avoid repeated text features in the embedding process,. This helps spread the features more evenly in the embedding table.To further reduce the chance of different tokens landing in the same spot, we use a modulo operation on the hash value. This gives us a specific position in the table to store each feature. This method makes the embedding more accurate and efficient.Next, all the feature vectors are combined into one long, flat vector. This vector is then passed through an MLP model to get the final semantic feature vector. The following is an explanation of Hash embedding, flattening and MLP.

3.3.1 Hash Embedding

Hash Embedding is a hashing of non-encrypted function [26]. This model mainly encodes the tokenized values using MurmurHash3. In our experiment, this method is suitable for use in IoT environments. We take the remainder of the smallest three-digit prime number from the hash value to determine the row and column in the embedding table. We will take the remainder of the smallest prime number 三位數中最小的質數 233 in the hash code as the column and row number of the corresponding table. For example, the hash code of PORT field name is 4283257230. Then take mod 233 and the remainder is 56, port number is 405, after the above operation, the number is 7, which is the column and row number of the corresponding table. which we use as the row. List the embedding column [-0.982, -0.301, -0.555, 2.061, 0.045, -0.618, -0.786, 0.573], which is an 8-dimensional vector. 一開始是隨機的，

3.3.2 Flatten

Flatten will string the tokenized data into a single vector through the vectors after the embedding column. For example, Destinazation port 405 is [-0.982, -0.301, -0.555, 2.061, 0.045, -0.618, -0.786, 0.573] andXXX 經過 tokenization 變成 [-0.024, 0.494, 0.754, -0.78, -1.002, 0.069, -0.52, -1.336] ,XXX [-1.042, -0.116, 0.542, -0.987, 1.001, 0.086, 0.699, -0.903] [-0.982, -0.301, -0.555, 2.061, 0.045, -0.618, -0.786, 0.573, -0.024, 0.494, 0.754, -0.78, -1.002, 0.069, -0.52, -1.336, -1.042, -0.116, 0.542, -0.987, 1.001, 0.086, 0.699, -0.903]

3.3.3 MLP

A multi-layer perception (MLP) consists of multiple fully connected layers with nonlinear activation functions, which enables it to learn complex associations between features and automatically extract high-level semantic patterns from the input data. The output of the MLP is called a semantic vector, which encapsulates the basic normal and abnormal behaviors present in the data packet [1]. Taking the above as an example, the semantic vector z is [-0.982, -0.301, -0.555, 2.061, 0.045, -0.618, -0.786, 0.573, -0.024, 0.494, 0.754, -0.78, -1.002, 0.069, -0.52, -1.336, -

1.042, -0.116, 0.542, -0.987, 1.001, 0.086, 0.699, -0.903]. Step 1: Weight matrix multiplication. The first layer of the MLP will have a weight matrix W. Assume this is an 8 X 8 conversion.

3.4 Mahalanobis Distance Model

To obtain abnormal isolated points in high dimensions, we use Mahalanobis distance [47] as the abnormal score judgment, combined with the distance between the semantic vector and the central area, so that the feature data is automatically gathered in the central area, and any deviation from the central area can be accurately judged as a potential anomaly.

3.4.1 Vector-to-Center Comparison

To enhance anomaly detection, S2GE-NIDS introduces a center loss mechanism. During training, all semantic vectors corresponding to ”normal” samples are aggregated to calculate a center point □ ，

• Taking into account the variability and correlation of each feature, more accurately detect abnormal samples that are ”off-center “

- Taking into account the variability and correlation of each feature, we can more accurately detect abnormal samples that are “off-center”.

$$D_M(z) = \sqrt{(z-c)^T \Sigma^{-1} (z-c)} \tag{3.1}$$

z is the semantic vector of the input sample, c is the center vector of normal samples, and Σ^{-1} is the inverse of the covariance matrix of the training data’s embedding vectors.

Chapter 4 Result & Analysis

Chapter 5 Conclusion & Future Work

5.1 Conclusion

5.2 Future Work

References

[1] Xin-She Yang. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[2] Xin-She Yang. "Metaheuristic optimization". In: *Scholarpedia* 6.8 (2011), p. 11472.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[4] Waseem Rawat and Zenghui Wang. "Deep convolutional neural networks for image classification: A comprehensive review". In: *Neural computation* 29.9 (2017), pp. 2352–2449.

[5] Heinz Mühlenbein, Martina Gorges-Schleuter, and Ottmar Krämer. "Evolution algorithms in combinatorial optimization". In: *Parallel computing* 7.1 (1988), pp. 65–85.

[6] Manoj Kumar et al. "Genetic algorithm: Review and application". In: *Available at SSRN 3529843* (2010).

[7] Michael O' Neill. "Riccardo Poli, William B. Langdon, Nicholas F. McPhee: A Field Guide to Genetic Programming". In: *Genetic Programming and Evolvable Machines* 10.2 (2009), pp. 229–230.

[8] Fred Glover. "Future paths for integer programming and links to artificial intelligence". In: *Computers & operations research* 13.5 (1986), pp. 533–549.

[9] Wikipedia contributors. *Metaheuristic — Wikipedia, The Free Encyclopedia*. [Online; accessed 15-December-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Metaheuristic&oldid=1255593755.

[10] Xin Yao, Yong Liu, and Guangming Lin. "Evolutionary programming made faster". In: *IEEE Transactions on Evolutionary computation* 3.2 (1999), pp. 82–102.

[11] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. "Differential evolution: A survey of the state-of-the-art". In: *IEEE transactions on evolutionary computation* 15.1 (2010), pp. 4–31.

[12] Diane B Paul. "The selection of the "Survival of the Fittest" ". In: *Journal of the History of Biology* 21 (1988), pp. 411–424.

[13] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. "Ant colony optimization". In: *IEEE computational intelligence magazine* 1.4 (2006), pp. 28–39.

[14] James Kennedy and Russell Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95-international conference on neural networks*. Vol. 4. ieee. 1995, pp. 1942–1948.

[15] John D. Co-Reyes et al. *Evolving Reinforcement Learning Algorithms*. 2022. arXiv: 2101.03958 [cs.LG]. URL: https://arxiv.org/abs/2101.03958.

[16] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. "Reinforcement learning: A survey". In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285.

[17] Veronika Thost and Jie Chen. *Directed Acyclic Graph Neural Networks*. 2021. arXiv: 2101.07965 [cs.LG]. URL: https://arxiv.org/abs/2101.07965.

[18] Esteban Real et al. "Regularized evolution for image classifier architecture search". In: *Proceedings of the aaai conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 4780–4789.

[19] David E Goldberg and Kalyanmoy Deb. "A comparative analysis of selection schemes used in genetic algorithms". In: *Foundations of genetic algorithms*. Vol. 1. Elsevier, 1991, pp. 69–93.

[20] Shakhnaz Akhmedova and Nils Körber. *Next Generation Loss Function for Image Classification*. 2024. arXiv: 2404.12948 [cs.CV]. URL: https://arxiv.org/abs/2404.12948.

[21] Zhilu Zhang and Mert Sabuncu. "Generalized cross entropy loss for training deep neural networks with noisy labels". In: *Advances in neural information processing systems* 31 (2018).

[22] Michael Iliadis, Leonidas Spinoulas, and Aggelos K Katsaggelos. "Deep fully-connected networks for video compressive sensing". In: *Digital Signal Processing* 72 (2018), pp. 9–18.

[23] Jianxin Wu. "Introduction to convolutional neural networks". In: *National Key Lab for Novel Software Technology. Nanjing University. China* 5.23 (2017), p. 495.

[24] Zhou Wang and Alan C Bovik. "Mean squared error: Love it or leave it? A new look at signal fidelity measures". In: *IEEE signal processing magazine* 26.1 (2009), pp. 98–117.

[25] Usha Ruby and Vamsidhar Yendapalli. "Binary cross entropy with deep learning technique for image classification". In: *Int. J. Adv. Trends Comput. Sci. Eng* 9.10 (2020).

[26] Alexis Plaquet and Hervé Bredin. "Powerset multi-class cross entropy loss for neural speaker diarization". In: *arXiv preprint arXiv:2310.13025* (2023).

[27] Santiago Gonzalez and Risto Miikkulainen. *Improved Training Speed, Accuracy, and Data Utilization Through Loss Function Optimization*. 2020. arXiv: 1905.11528 [cs.LG]. URL: https://arxiv.org/abs/1905.11528.

[28] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. "CIFAR-10 (Canadian Institute for Advanced Research)". In: (). URL: http://www.cs.toronto.edu/~kriz/cifar.html.

[29] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. "CIFAR-100 (Canadian Institute for Advanced Research)". In: (). URL: http://www.cs.toronto.edu/~kriz/cifar.html.

[30] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 2017. arXiv: 1708.07747 [cs.LG]. URL: https://arxiv.org/abs/1708.07747.

[31] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[32] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. “Fast image-based localization using direct 2d-to-3d matching”. In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 667–674.

[33] Yihui He et al. *Bounding Box Regression with Uncertainty for Accurate Object Detection*. 2019. arXiv: 1809.08545 [cs.CV]. URL: https://arxiv.org/abs/1809.08545.

[34] Zhong-Qiu Zhao et al. “Object detection with deep learning: A review”. In: *IEEE transactions on neural networks and learning systems* 30.11 (2019), pp. 3212–3232.

[35] Bohan Zhuang et al. “Structured binary neural networks for accurate image classification and semantic segmentation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 413–422.

[36] Venkatesh N Murthy et al. “Deep decision network for multi-class image classification”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2240–2248.

[37] Grigorios Tsoumakas and Ioannis Katakis. “Multi-label classification: An overview”. In: *Data Warehousing and Mining: Concepts, Methodologies, Tools, and Applications* (2008), pp. 64–74.

[38] Jia Deng et al. “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[39] Li Deng. “The mnist database of handwritten digit images for machine learning research”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[40] Sergio Saponara and Abdussalam Elhanashi. “Impact of Image Resizing on Deep Learning Detectors for Training Time and Model Performance”. In: Jan. 2022, pp. 10–17. ISBN: 978-3-030-95497-0. DOI: 10.1007/978-3-030-95498-7_2.

[41] Christian Szegedy et al. *Rethinking the Inception Architecture for Computer Vision*. 2015. arXiv: 1512.00567 [cs.CV]. URL: https://arxiv.org/abs/1512.00567.

[42] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.