

THESIS

TAMIM DEEB
Mechanical Engineering

Gödöllő
2024



Hungarian University of Agriculture and Life Sciences

Szent István Campus

Institute of Technology

Mechanical engineering master's

**Development of a Part Recognition Tool for Physical Analysis of
eAxles**

Insider consultant: Tarr Bence

Associate professor

Insider consultant's

Institute/Department: Department of Engineering

Informatics

Outsider consultant: Lelkes János

Simulation engineer, Robert

Bosch GmbH

Created by:

Tamim Deeb

Robert Bosch GmbH (Budapest, Gyömrői út 104)

2024

INSTITUTE OF TECHNOLOGY
MECHANICAL ENGINEERING/ MASTER¹

THESIS

worksheet

Tamim Deeb (fzpyfx)

for

Title of the diploma thesis:

Development of a Part Recognition Tool for Physical Analysis of eAxles

Task reference:

- Overview and Testing of Image Recognition AI Models.
- Creation of a Learning Dataset for AI Model Training.
- Development of a Part Identification AI Model.
- Evaluation of AI Model Performance.

Contributing department: Mechanical engineering

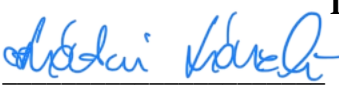
Outsider consultant: *Lelkes János, Simulation engineer, NVH and topology optimization, Transmission engineer, Robert Bosch GmbH.*

Insider consultant: *Tarr Bence, MATE, Department of Engineering Informatics*

Deadline for thesis submission: 2024 April 29

Gödöllő, 2023 November 13

I approve




(head of department)



(specialist coordinator)

Received



(student)

As the external consultant to the thesis writer, I declare that the student has attended the pre-arranged consultations.

2024 April 26



¹The appropriate training level should be retained, the other deleted.

Table of content

1	Introduction	9
2	Literature Review	10
2.1	Introduction of the Literature review	10
2.2	Image Classification in Machine Learning	10
2.2.1	Machine Learning Algorithms in Image classification	12
2.2.1.1	Support Vector Machines (SVMs)	12
2.2.1.2	Decision Trees	14
2.2.1.3	Random Forests	15
2.2.1.4	k-Nearest Neighbors (k-NN)	16
2.2.2	Feature Extraction Techniques	16
2.2.2.1	Color Features	17
2.2.2.2	Texture Features	17
2.2.2.3	Shape Features	17
2.2.2.4	Invariant Feature Transform (SIFT)	17
2.2.2.5	Histogram of Oriented Gradients (HOG)	18
2.2.2.6	Gabor Filters	19
2.3	The Rise of Deep Learning and Convolutional Neural Networks	21
2.3.1	Convolutional Neural Networks (CNNs)	21
2.3.1.1	ResNet	23
2.3.1.1.1	ResNet18	25
2.3.1.1.2	ResNet50	26
2.3.2	Advanced Techniques for Data preparation and model optimization	26
2.3.2.1	Image processing and recognition	26
2.3.2.2	Image types	27
2.3.2.2.1	Binary image	27
2.3.2.2.2	Indexed Images	27
2.3.2.3	Histogram of an image	28
2.4	Conclusion	30
3	Material and Methods	31
3.1	Linearoch: Systematizing Complex Data Structures for Enhanced Machine Learning Preparation	31
3.1.1	Copying the dataset	32

3.1.2	Submapper	32
3.1.3	Renamer	33
3.1.4	Submapper	34
3.1.5	Mover	34
3.1.6	MPD	35
3.1.7	Submapper	35
3.1.8	Resizing	35
3.1.8.1	Resizing without Padding	35
3.1.8.2	Resizing with padding	36
3.1.9	Splitter	36
3.1.10	Augmenter	36
3.1.11	Submapper	36
3.1.12	Mover	37
3.1.13	EFD	37
3.1.14	Comparer	37
3.2	Tailored Approaches for Peak Performance in Image Classification	37
4	Results	39
4.1	Systematic Exploration of Configuration Parameters Impacting Image Classification Efficacy	39
4.1.1	Data Size	39
4.1.2	Model Type	42
4.1.3	Image Size	44
4.1.4	Augmentation Timing	45
4.1.5	Padding inclusion	47
4.1.5.1	Analysis of ResNet18 Trained on 4000 Images	47
4.1.5.1.1	Aspect Ratio Preserved without Padding (348x232)	47
4.1.5.1.2	Aspect Ratio Preserved with White Padding (224x224)	47
4.1.5.2	Additional Observations from ResNet50 Trained on 10000 Images	48
4.1.5.2.1	ResNet50 without Padding, Direct Resizing to Square (299x299)	48
4.1.5.2.2	ResNet50 with White Padding (299x299)	49
4.1.6	Extent of augmentation	49
4.1.6.1	No Augmentation (0:1 ratio)	50
4.1.6.2	Mild Augmentation (1:1 ratio)	50

4.1.6.3	Moderate Augmentation (2:1 and 3:1 ratios)	50
4.1.6.4	High Augmentation (4:1, 5:1, and 6:1 ratios).....	51
4.1.7	Batch Size and Epoch Number	53
4.1.7.1	Batch Size 16.....	53
4.1.7.2	Batch Size 32.....	54
4.1.7.3	Batch Size 64.....	55
4.1.8	Comparing Model Architectures.....	55
5	Conclusion.....	58
6	Summary.....	59
7	References	63

1 Introduction

The advent of electric vehicles (EVs) represents a transformative shift in the automotive industry, driven by the imperative for sustainability and bolstered by rapid advancements in technology. Central to this transformation is the development of electric axles (eAxles), which consolidate essential drivetrain components into a singular, efficient unit. This thesis introduces the development of a Part Recognition Tool for eAxles, employing advanced image recognition technologies enhanced by artificial intelligence (AI). This tool is designed to automate the identification and categorization of eAxle components, thereby facilitating a more nuanced and efficient analysis during the developmental phases of these crucial elements. Situated at the confluence of mechanical engineering and artificial intelligence, this research explores the potential of digital technologies to revolutionize the development and analysis of mechanical components in EVs. The Part Recognition Tool aims to minimize human error and optimize the efficiency of development workflows through the automation of part recognition tasks. By streamlining these processes, the tool not only aims to enhance the speed and accuracy of component analysis but also to significantly reduce development costs and time-to-market for new eAxle designs.

The research begins with a comprehensive review of existing image recognition models and machine learning algorithms, identifying those that hold the most promise for adaptation to the complexities of eAxle components. This foundational work informs the creation of a specialized dataset, meticulously annotated to detail the diverse parts of eAxles. This dataset serves as the basis for training and testing the AI models developed in this thesis. The integration of such a dataset is critical, as it reflects real-world variability and complexity, providing a robust platform for the AI to learn and adapt effectively. In terms of methodology, this thesis employs a combination of qualitative and quantitative research techniques. Qualitative analysis involves a thorough review of scholarly literature and existing technologies, focusing on their application within mechanical engineering contexts and pinpointing the gaps this project aims to address. Quantitatively, the project entails the experimental training of machine learning models, rigorous validation of these models, and a detailed statistical analysis of their performance. The objective is to develop a model that achieves high accuracy and reliability in recognizing and categorizing eAxle parts under various conditions. This dual approach ensures that the research is grounded in solid theoretical foundations while also being tested against empirical standards.

The structure of the thesis ensures a coherent presentation of the research. It begins with this introduction, which sets the stage by outlining the research problem and its significance. Subsequent chapters delve into the detailed literature review, describe the methodology for dataset creation and model development, and present the results of model testing. The discussion interprets these results, comparing them with established technologies and drawing conclusions about their practical implications. The thesis concludes with a summary of the research contributions and proposes directions for future investigations in this vibrant field of study. The anticipated impact of this research extends beyond the academic, suggesting practical applications that could influence future designs and maintenance strategies for electric vehicles, potentially setting new industry standards for automated systems in automotive engineering.

2 Literature Review

2.1 Introduction of the Literature Review

In the vast realm of artificial intelligence, image classification plays a vital role in various applications like facial recognition, medical imaging analysis, and autonomous vehicle navigation. It enables systems to categorize images into different classes based on their content, transforming complex visuals into structured data that can be used for analysis and practical purposes. This transformation is crucial for systems to interact with and interpret the visual world effectively.

The core of this literature review centers on evaluating the robustness and efficiency of ResNet architectures, specifically the ResNet-18 and ResNet-50 models, in their application to complex and imbalanced datasets. These models are celebrated for their profound capabilities in deep learning, particularly adept at learning from the intricate and hierarchical data structures that are typical in scenarios where data is not uniformly distributed across categories.

ResNet architectures incorporate innovative features like skip connections, which are crucial for overcoming the vanishing gradient problem—a common challenge in training deep neural networks. These skip connections help gradients flow smoothly during the training process, enabling the construction of deeper networks without encountering the usual training difficulties associated with increased depth.

This review aims to shed light on three primary aspects. Firstly, it explores the structural elements of ResNet models that enhance their ability to address vanishing gradients, leading to improved training effectiveness and model performance. Secondly, it examines how these models perform on imbalanced datasets, which is essential for ensuring fair and accurate outcomes in AI applications. Lastly, it evaluates the scalability and adaptability of ResNet models, considering how well they handle changes in dataset size and complexity—a crucial factor for their practical deployment in real-world applications, where computational resources and data availability can vary significantly.

By conducting an extensive analysis of existing literature on ResNet models, this review highlights their unique capabilities in streamlining the image classification process and ensuring accurate representation of all classes in a dataset despite inherent challenges. These insights contribute to both the theoretical framework of image classification and practical applications, suggesting ways to enhance the performance and reliability of machine learning systems that deal with image-based data. This comprehensive examination not only deepens our understanding of ResNet architectures but also provides actionable insights that could inspire future innovations in the field of artificial intelligence.

2.2 Image Classification in Machine Learning

Machine learning is a specialized area within the discipline of artificial intelligence that focuses on enabling computers to acquire knowledge and improve their performance through learning. Although this perspective is basic, ever since the inception of the first computer, we have pondered

the capacity of computers to acquire knowledge in a manner similar to humans. In 1959, Samuel et al. introduced a set of methods to develop an algorithm with the goal of enabling computers to outperform inexperienced people in the game of checkers. The goal was ambitious, especially considering the restricted availability of hardware at that time. Nevertheless, this demonstrates the significance of machine learning since the inception of computers.

Learning is a highly intricate process, and there is no consensus on its definition. However, for humans, learning can be characterized functionally as alterations in behavior that occur as a result of experience or mechanistically as changes in the organism that occur as a result of experience (De Houwer et al., 2013). Computers are computational devices; therefore, we must see learning as a computer algorithm. Figure 1 illustrates the visual representation of classical programming (a) and machine learning (b). It is important to note that traditional programming focuses on determining the correct output based on provided inputs and a program. In contrast, machine learning focuses on identifying the correct program, referred to as a model, based on a set of inputs and outputs, which may be empty. The sophisticated program is now capable of suggesting novel outputs based on fresh inputs.

Fig. 1 clearly demonstrates that there are substantial differences in the understanding of learning between humans and computers. Machine-learning algorithms strive to create a model that precisely reflects the input data in order to propose new outputs. Returning to the definitions of

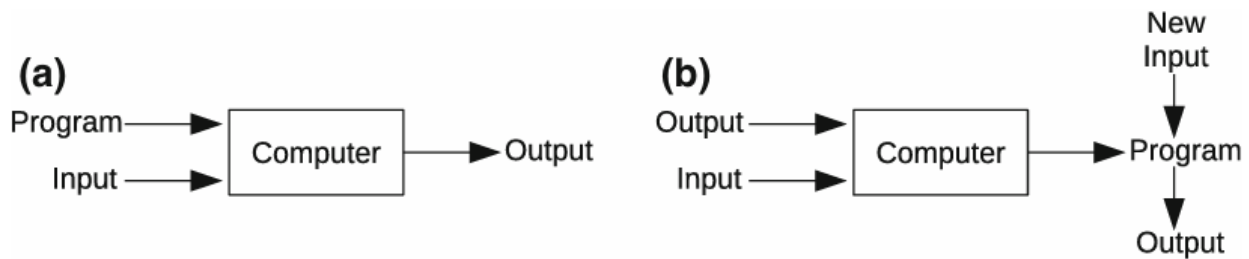


Fig. 1 Traditional programming (a) and Machine learning (b) (Duarte & Ståhl, 2019).

machine learning, we highlight two primary explanations. The initial one, introduced by Samuel et al. (1959), defines machine learning as a discipline that enables computers to learn without direct programming. It should be noted that Samuel's definition was one of the earliest proposed definitions. Almost forty years later, Mitchell (1997) presented a more formalized perspective on machine learning. According to Mitchell, a computer program can be considered to learn from experience E in relation to a specific task T and a performance measure P if its performance on T , as evaluated by P , demonstrates improvement with increasing experience E .

From the observations made so far, the process of machine learning can be categorized into four distinct steps: The process involves obtaining the training set X , selecting and executing a learning task using X , constructing a model, and evaluating the model using new inputs. These four stages can be iterated until a satisfactory P value is achieved. It is important to consider that when confronted with a machine-learning problem, the initial obstacle is determining the appropriate machine-learning algorithm to employ. Thousands of options are now accessible, and each year, hundreds of new options are presented (Domingos, 2012). The collection of potential learning

algorithms that can be used to solve a specific machine-learning problem is referred to as the hypothesis space. In order to avoid confusion amidst the vast array of options, the learning components can be categorized into three distinct groups (Domingos, 2012):

- **Representation:** An algorithm must be used to represent a task. In order to narrow down the hypothesis space, it is crucial to determine the specific form of learning that we are interested in.
- **Evaluation:** The projected outputs must be tested to determine the effectiveness of the chosen representation. Various evaluation functions can be employed depending on the specific task at hand.
- **Optimization:** After analyzing the findings of the assessment component, it is necessary to perform optimization. The objective of the learning algorithm is to optimize a specified performance metric.

The assessments of a machine learning model are conducted on previously unseen instances from the same dataset that was used for training (Duarte & Ståhl, 2019).

2.2.1 Machine Learning Algorithms in Image Classification

The creation of automated image classification algorithms has risen to prominence as a critical research area over the last several decades. These algorithms are utilized in a wide array of sectors, such as web search engines, digital libraries, geographic information systems, biomedicine, surveillance, e-commerce, sensor networks, manufacturing, and educational systems. Within these applications, image classification functions as an initial organizational tool, categorizing images in databases into categories that carry semantic meaning. This activity is central to the disciplines of pattern recognition, image processing, and computer vision, which have been heavily focused on the identification, detection, and categorization of a specific spectrum of objects or ideas within distinct application areas. Essential to this process is the extraction of features, where fundamental attributes of images like color, texture, and dimension are examined. Machine learning techniques are implemented to effectively determine the classifications of these attributes (Chen et al., 2010). This section will explore four machine-learning techniques.

2.2.1.1 Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are advanced learning systems that operate within a high-dimensional space, leveraging linear functions shaped by hypotheses (Roweis et al., 2000). These systems are trained through an optimization-focused algorithm that incorporates a bias informed by statistical learning theory. The primary objective in using SVMs for classification is to develop an effective method for identifying optimal hyperplanes that facilitate data separation in this multidimensional space, thereby enhancing generalizability. The term 'computationally efficient' refers to the capability of these algorithms to handle very large data sets efficiently (Tao et al., 2008 & Belkin et al., 2001). SVMs are typically applied to binary classification problems. Drawing from Bennett's (2000) discussion, consider a dataset comprising "1" observations, each represented by a pair $\{x_i, y_i\}$, where x_i is an N-dimensional vector and y_i is a class indicator with values of -1 or 1. The goal is to discover the best possible hyperplane, an (N-1) dimensional surface

that distinctly separates these classes. Initially, it is presumed that such a hyperplane is feasible, characterized by a normal vector w , with two parallel planes established on either side:

$$W \times x_i \geq b + 1 \text{ for } y_i = 1 \quad (1)$$

$$W \times x_i \leq b - 1 \text{ for } y_i = -1 \quad (2)$$

Here, b represents the offset of the plane from the origin. Often, separating data requires a non-linear solution plane, which can be computationally intensive to achieve through repeated optimization of the separation between two non-linear functions (Li et al., 2008 & Tao et al., 2007). To address this, the kernel trick is employed, where input data are transformed into a higher dimensional feature space using a specific kernel function. In this transformed space, the data become linearly separable. Additionally, a technique to handle errors and outliers in the input data has been developed. This involves allowing a permissible error of up to ξ in each dimension, creating what is termed a 'fuzzy margin,' and incorporating a penalty function $C(i)$ into the optimization equation, as discussed by Burges. The goal is to minimize this function.

$$1/2 \|W\|^2 + C \times (\sum \xi_i) \quad (3)$$

Subject to the constraint:

$$y_i(W \times x_i - b) + \xi_i \geq 1 \quad (4)$$

Solving this is significantly more challenging than the separable case. In the LIBSVM manual by Chang and Lin, the constraints, conditions for minimization, and the resulting decision functions are specified for each classification type. Additionally, the manual includes algorithms for addressing the necessary quadratic programming challenges.

Support Vector Machines were initially crafted solely for binary classification tasks. However, adaptations have been made to extend SVMs to handle multi-class classification problems through two principal methods, both aimed at breaking down the complex multi-class scenario into simpler binary classification tasks. The "one against all" method is the first of these approaches. In this method, individual binary classifiers are developed for each class to distinguish it from all other classes. Upon evaluating each data point, the classifier that reports the highest decision score assigns its respective class to the point (Hsu and Lin 2002). This technique involves creating N classifiers corresponding to the number of classes, each yielding a decision function. While rapid, this approach can be vulnerable to inaccuracies arising from training sets that are not perfectly balanced. An iteration of this method simplifies the process by solving a single optimization problem to extract all N decision functions, potentially reducing the need for as many support vectors as would be necessary when each binary classification is handled separately. The second approach, "one against one," involves training classifiers for every possible pair of classes. Each object is then classified based on the most frequently identified class across all classifiers for that object. To finalize the classification, a maximum-winner technique is applied. This method requires the use of $N(N-1)/2$ classifiers, marking it as more computationally intensive than the "one against all" approach. Despite its complexity, this method has been found to be better suited for addressing the nuances of multi-class classification tasks (Hsu and Lin 2002), and as such, it has been preferred for applications in SVM-based object image classification.

2.2.1.2 Decision Trees

Decision tree ensemble methods hold significant popularity in machine learning due to their ability to enhance an existing algorithm by aggregating the outputs of multiple models. Particularly when combined with decision trees, these methods excel in performance; this combination tends to surpass the accuracy of decision trees used alone, which may not always compete well with other algorithms. Given their success across various domains and their ability to rival other advanced algorithms, ensemble methods utilizing decision trees serve as an excellent foundation for developing a universal system for image classification, where no specific assumptions about the problem are made (Ouma et al., 2008).

In this context, the suggestion is to employ a specific ensemble approach involving decision trees for tackling image classification tasks. This method involves the construction of a multitude of extremely randomized trees, initially introduced in (Kim et al., 2009). These trees resemble traditional decision trees in structure (Pesaresiand & Benediktsonn, 2001) but differ significantly in their development process. Unlike the standard method, where a tree is developed top-down, selecting tests at each node to optimize a score measure, the extremely randomized approach selects tests and thresholds randomly at each node. This randomness is based on the mean values present, simplifying the recursive function used in image classification.

In practice, the system begins with a learning set comprising all examples. Tests start at internal nodes, such as comparing a pixel's value at a specified position to a threshold, to facilitate the classification. By constructing multiple such trees, ideally as many as feasible from the same learning set, the system enhances its predictive accuracy. When classifying an image, it is successively processed through all trees, and the most frequently occurring class among the outputs is assigned to the image. This method encapsulates several procedural steps of the algorithm, underscoring its robustness in handling image classification (Kumar et al., 2012).

- Build extra tree (LS) (Kumar et al., 2012):

1. If LS contains all images of the same class, then return a leaf with this class associated with it;
2. Otherwise:

I. Set $[\alpha_{k,1} < \alpha_{th}] = \text{Choose a random split (LS)}$.

II. Split LS into LS_{left} and LS_{right} according to the test $[\alpha_{k,1} < \alpha_{th}]$ and build the subtrees T_{left} build extra tree (LS_{left}) and $T_{right} = \text{build extra tree } (LS_{right})$ from these subsets.

3. Create a node with the test $[\alpha_{k,1} < \alpha_{th}]$ attach T_{left} and T_{right} as successors of this node and return the resulting tree.

- Choose a random split (LS) ((Kumar et al., 2012):

1. Select a pixel location $(K,1)$ at random;
2. Select a threshold α_{th} random according to a distribution $N(\mu_{k,1}, R_{k,1})$, where $\mu_{k,1}$ and $R_{k,1}$ are respectively the mean and standard deviation of the pixel values $\alpha_{k,1}$ in LS.

3. If the score of this test is greater than a given threshold S_{th} return the test $[\alpha_{k,1} < \alpha_{th}]$.
4. Otherwise, return to step 1 and select a different location. If all locations have already been considered, then return the best test so far.

2.2.1.3 Random Forests

Random forests are a method used to construct a classification ensemble by growing a set of decision trees in randomly chosen subspaces of data (Breiman, 2001). Empirical evidence has demonstrated that random forest classifiers can attain a noteworthy level of accuracy when categorizing data in areas characterized by a large number of parameters and numerous classes (Breiman, 2001 & Banfield, 2007). Various techniques have been suggested for constructing random forest models using subsets of data (Breiman, 2001 & Ho, 1998). One of the most widely used methods for constructing forests in the field of forestry, as suggested by (Breiman, 2001), is to randomly choose a subset of features at each node to create branches in decision trees. Then, the bagging method is employed to generate subsets of training data for constructing individual trees. Finally, all the individual trees are combined to create a random forest model (Breiman, 2001). However, when applied to image data, which is characterized by high dimensionality, sparsity, and multi-class labels, random forests face challenges.

To address these challenges, Xu, Ye, and Nie (2012) propose two strategies. The first strategy is a feature weighting method for subspace sampling. This method computes feature weights based on the correlations between features and the class label. Features with higher weights are more likely to be selected in the construction of decision trees, thus improving their classification power. The second strategy is a tree selection method aimed at excluding "weak" trees from the random forest ensemble. This is done by evaluating the importance of each tree using out-of-bag accuracy and selecting only the top-performing trees for the final ensemble.

Xu, Ye, and Nie (2012) present their improved random forest algorithm (Fig. 2), which incorporates both the feature weighting method and the tree selection method. In this algorithm, feature weights are used to guide the sampling of features for each decision tree, and only the top 80% of trees with high out-of-bag accuracy are included in the ensemble. The experimental results support the effectiveness of the proposed method in generating better random forests for image classification tasks.

Algorithm 1. Improved Random Forest Algorithm

Input:

- D : the training data set,
- A : the feature space $\{A_1, A_2, \dots, A_M\}$,
- Y : the feature space $\{y_1, y_2, \dots, y_n\}$,
- K : the number of trees,
- m : the size of subspaces.

Output: A random forest μ **Method:**

- 1: **for** $i=1$ to K **do**
- 2: draw a bootstrap sample in-of-bag data subset IOB_i and out-of-bag data subset OOB_i from the training data set D ;
- 3: $h(IOB_i) = \text{createTree}(IOB_i)$;
- 4: use out-of-bag data subset OOB_i to calculate the out-of-bag accuracy $OOBAcc_i$ of the tree classifier $h_i(IOB_i)$ by Equation (4);
- 5: **end for**
- 6: sort all K trees classifiers in their $OOBAcc$ descending order;
- 7: select the top 80% trees with high $OOBAcc$ values and combine the 80% tree classifiers into an improved random forest μ ;

Function createTree()

- 1: create a new node η ;
 - 2: **if** stopping criteria is met **then**
 - 3: return η as a leaf node;
 - 4: **else**
 - 5: **for** $j=1$ to $j=M$ **do**
 - 6: compute the informativeness measure $\text{corr}(A_j, Y)$ by Equation (1);
 - 7: **end for**
 - 8: compute feature weights $\{w_1, w_2, \dots, w_M\}$ by Equation (3);
 - 9: use the feature weighting method to randomly select m features;
 - 10: use these m feature as candidates to generate the best split for the node to be partitioned;
 - 11: call createTree() for each split;
 - 12: **end if**
 - 13: return η ;
-

Fig. 2 Improved Random Forest Algorithm (Xu, Ye, & Nie, 2012).

2.2.1.4 k-Nearest Neighbors (k-NN)

K-Nearest Neighbors (KNN) classifiers demonstrate effective image classification when a query image closely resembles a labeled image within its class. Nearest Neighbor (NN) classifiers are particularly robust in specific image classification fields characterized by a high number of labeled images relative to class complexity. From a theoretical standpoint, NN classification approaches the Bayes optimal classifier as the sample size becomes infinitely large (Guo et al., 2002 & Vapnik, 1995). Nonetheless, NN classifiers struggle to generalize beyond the set of labeled images. This limitation becomes apparent in practical scenarios where the available number of training or labeled images is disproportionately small compared to the complexity of the class. Poor classification results often occur when there are few labeled images for classes that exhibit significant variability in object shape and appearance. In situations where images are characterized by "bag-of-features" histograms, "Image-to-Image" distance is defined as the distance between two images' descriptor distributions, which can be calculated through methods like histogram intersection, Chi-square, or KL divergence (Bian & Tao., 2010). While typical NN image classifiers evaluate the descriptor distribution for each image individually (Dalla Mura et al., 2011 & Dalla Mura et al., 2010), utilizing the descriptor distribution from the entire class (all images in class C) could enhance generalization. This approach allows for an "Image-to-Class" distance measurement by computing the KL distance between the query descriptors and the class descriptors. Even if the "Query-to-Image" KL distance is substantial for all labeled images in a specific class, such as Ballet, a small "Query-to-Class" KL distance can still facilitate accurate classification.

2.2.2 Feature Extraction Techniques

Feature extraction plays a crucial role in machine learning for image classification. It involves transforming raw data into a set of features that effectively differentiate between different classes.

Here are some commonly used techniques for feature extraction in machine learning for image classification:

2.2.2.1 Color Features

Color features utilize color histograms, color moments (mean, variance, skewness), and color coherence vectors. A color histogram represents the distribution of colors within an image and is robust to changes in orientation and scale. This technique is particularly valuable in scenarios where color serves as a significant distinguishing characteristic of the objects being classified (Gonzalez & Woods, 2002).

2.2.2.2 Texture Features

Texture features capture the visual patterns and their spatial arrangement in an image. Popular methods for extracting texture features include the Gray Level Co-occurrence Matrix (GLCM), Local Binary Patterns (LBP), Gabor filters, and Tamura features. These techniques measure properties such as contrast, coarseness, and directionality of the texture in an image (Haralick, Shanmugam, & Dinstein, 1973).

2.2.2.3 Shape Features

Shape features describe the geometric properties of an object. Techniques like edge detection (using filters like Sobel or Canny), region-based segmentation, and morphological features (such as perimeter, area, and convexity) are commonly employed. Shape features are especially important in applications where the shape of objects serves as a strong indicator of their class (Zhang & Lu, 2004).

2.2.2.4 Invariant Feature Transform (SIFT)

In the field of computer vision, the Scale-Invariant Feature Transform (SIFT) algorithm identifies and characterizes local features within an image. SIFT features facilitate accurate matching across different perspectives of the same subject, thanks to their resistance to changes in scale, orientation, and partial resistance to variations in lighting (Das & Vijaykumar, 2010). The process of extracting SIFT features involves four main steps. Initially, potential interest points in the image are identified by detecting extrema across various scales using Difference of Gaussian (DOG) filters applied to the image. Points that are in low-contrast regions or along edges are subsequently removed. Each of the remaining points is then assigned an orientation based on the gradients of the surrounding image. In the final step, local image features are calculated from the image gradients around each key point. These features are represented as 128-element vectors, defined within the 4 x 4 neighborhoods surrounding each key point (Das & Vijaykumar, (2010). In other words, the Scale Invariant Feature Transform (SIFT) method, as detailed in the Kher & Thakar, (2014), is a robust image processing technique primarily used for image matching and registration. It excels in identifying and matching key features across images that are invariant to scale, rotation, and partially to changes in illumination and 3D viewpoint. The SIFT algorithm involves four main stages: scale-space extrema detection to identify potential interest points, keypoint localization to refine positions and eliminate weak candidates, orientation assignment to ensure rotation invariance and keypoint descriptor formation to create unique fingerprints for each key point.

These descriptors enable reliable matching of features across varied conditions, proving SIFT's effectiveness in applications such as object recognition, 3D reconstruction, and video tracking. The paper highlights SIFT's distinctive capability to handle complex transformations and challenging environments, demonstrating its broad utility in computer vision tasks.

2.2.2.5 Histogram of Oriented Gradients (HOG)

In many cases, HOG descriptors can often be utilized in a similar manner to SIFT descriptors. Vo et al. (2013) specifically explore the use of HOG as a mechanism for extracting features in image classification tasks. Similar to SIFT, HOG descriptors enable an image classifier to identify matches between a given image and the images used to train the classifier. This is achieved by constructing a visual bag-of-words (BoW) model of the HOG descriptor and combining it with a machine learning algorithm like SVM. HOG descriptors are typically represented in three-dimensional space due to the way they are generated. Additionally, their size tends to be large, depending on the image resolution. Fig. 3 illustrates the HOG descriptors created from an accordion image. Unlike SIFT, HOG allows for the configuration of a parameter called the number of orientations (O), which can significantly influence the resulting features (as depicted in Fig. 3b and Fig. 3c). This flexibility enables HOG to be applied to various image detection problems.

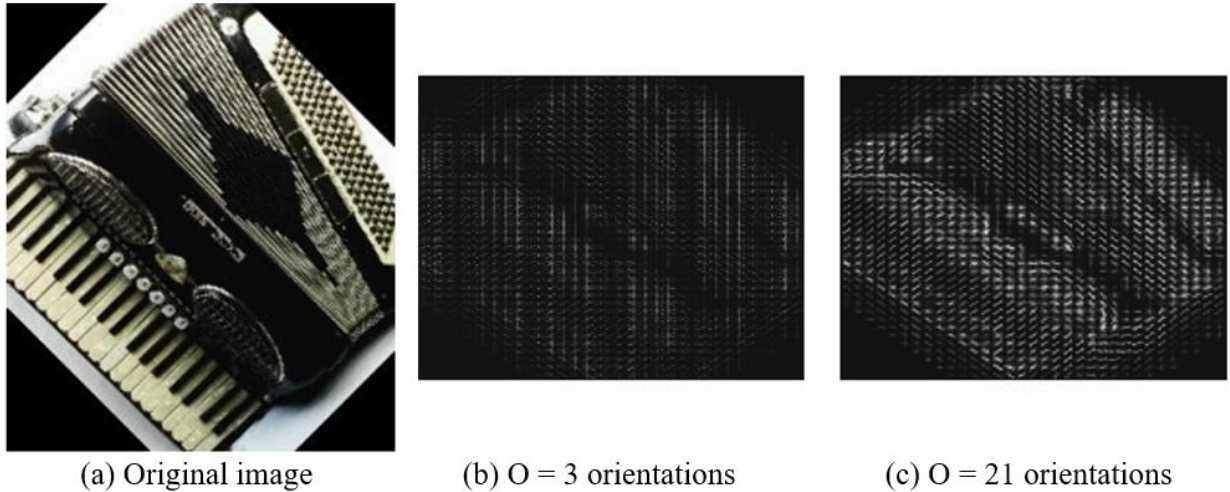


Fig. 3 Visual representation of HOG descriptors (Vo et al., 2013).

Fig. 4 outlines the steps involved in extracting HOG descriptors from an image. The process begins by dividing the image into equal-sized cells, as shown in Fig. 4(a). Within each cell, a histogram of gradient directions or edge orientations is accumulated over the pixels, as depicted in Fig. 4(b). The orientation $\theta(x, y)$ and magnitude $r(x, y)$ of a pixel (x, y) are calculated using a 1-D discrete derivations mask $[-1, 0, 1]$ and its transpose $[-1, 0, 1]^T$. The magnitude $r(x, y)$ is computed using the color channel with the highest gradient magnitude. Assuming the number of orientations O is set to 9, there will be 18 directed orientation bins allocated, representing one bin for every 20° within the range of 0° to 360° : 2 orientations (+/-) for each of the 9 undirected gradient directions (Dalal et al. 2005).

The next step in HOG is block normalization, where blocks are formed by grouping four adjacent cells together (sliding each cell), as shown in Fig. 4(c). Let vector v represent the stacking of the positive direction histogram in a block, and $\|v\|_2$ be the two-norm of v and ϵ along with a very small number (assumed to have an insignificant value). The norm (l^2 -norm) of a block is defined as (Vo et al., 2013):

$$v = \frac{v}{\sqrt{\|v\|_2^2 + \epsilon^2}} \quad (5)$$

The final step, Fig. 4(d), generates the actual descriptors. For each cell, four normalization factors are obtained as the inverses of the norms of the four blocks containing it. The cell's undirected 9-dimensional histogram is then normalized using each normalization factor separately. The resulting histograms are stacked and clipped at 0.2. This process yields a vector of length 36 (4×9), which serves as the HOG descriptor representing the cell (Vo et al., 2013).

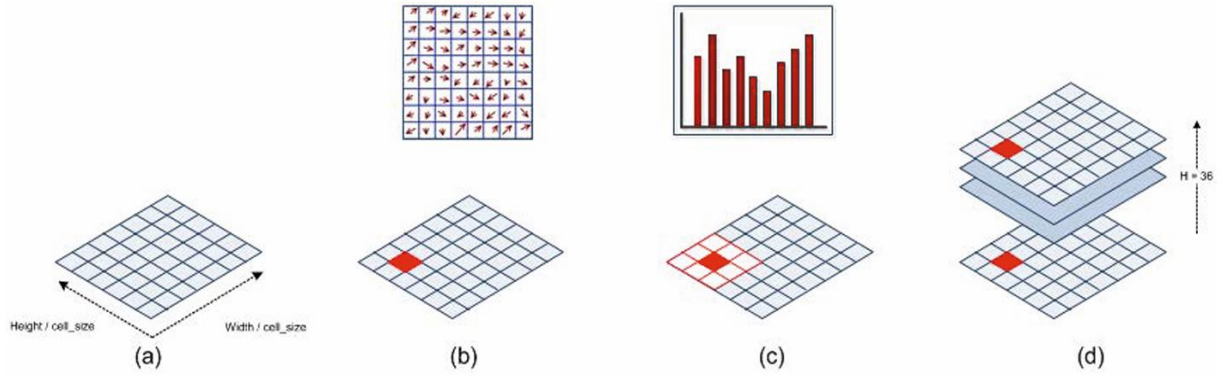


Fig. 4 Process of creating HOG descriptors (Vo et al., 2013).

2.2.2.6 Gabor Filters

A feature extractor plays a crucial role in enhancing the accuracy of a classifier by providing visually distinctive patterns. One such feature extractor is the Gabor filter, which selectively extracts patterns from a signal or data at specific frequencies. The Gabor filter is created by modulating harmonic functions with a Gaussian distribution function. The concept of the Gabor filter was initially introduced by Dennis Gabor in the 1940s and later extended to 2D filters by Daugman in the 1980s (YI & Su, 2014) (Singh & Dhir, 2012). The Gabor filter is defined by multiplying a sinusoidal wave with a Gaussian function. The real and imaginary components of the filter are generated by the cosine and sine waves, respectively. These two components can be combined into a complex number, as expressed below.

$$h = (x, y, \lambda, \phi, \sigma_x, \sigma_y) = \frac{1}{2\pi\sigma_x\sigma_y} \times e^{-\frac{1}{2}(\frac{x'^2}{\sigma_x^2} + \frac{y'^2}{\sigma_y^2})} \times e^{i(\frac{2\pi x'}{\lambda})} \quad (6)$$

Where x' and y' are:

$$x' = x \cos \phi + y \sin \phi \quad \text{and} \quad y' = y \cos \phi - x \sin \phi$$

The wavelength and orientation of a Gabor filter are typically represented by λ (lambda) and θ (theta), respectively. The parameter σ (sigma) denotes the standard deviation of the Gaussian envelope in a one-dimensional plane. By utilizing various scales and orientations, a Gabor filter bank can be created (Yi et al., 2009) (Haghighat et al., 2015). For instance, a filter bank consisting of 40 filters can be generated by employing 5 scales and 8 orientations, as illustrated in Fig. 5 (Haghighat et al., 2015).

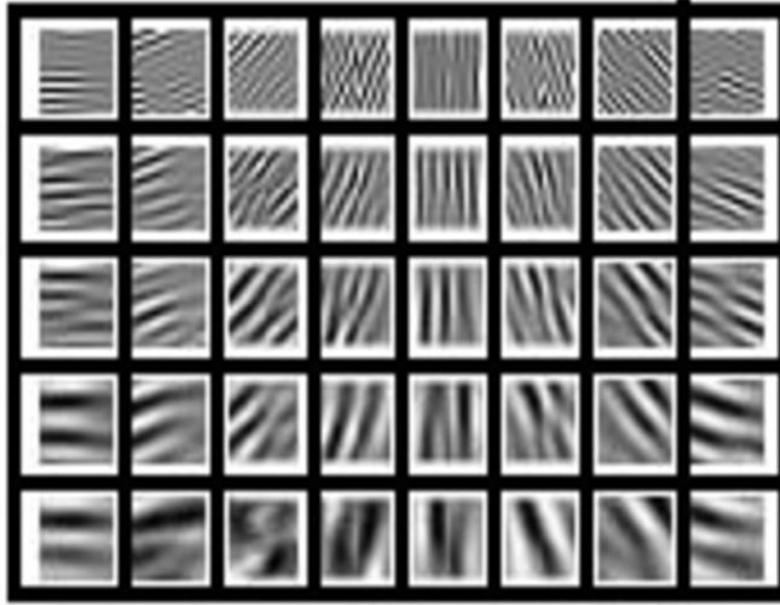


Fig. 5 Gabor Filter Bank for five Scales and eight Orientations (Tahir et al., 2016).

The filter bank can be convolved with an image, resulting in the extraction of features and the creation of a structure called a feature map. This process is performed across different orientations and scales using the following equation.

$$\text{Feature Map } (x, y) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} \text{Im}(k_1, k_2) h(x - k_1, y - k_2, \lambda, \phi, \sigma_x, \sigma_y) \quad (7)$$

The concept of feature extraction mentioned above is demonstrated in Fig. 6. In this illustration; an image undergoes convolution with a Gabor filter bank to generate a feature map for different orientations and scales. It is important to note that this process involves a significant number of computations, as indicated in reference (Wang et al., 2002).

A feature map is generated for every combination of Gabor filters for each image channel. In the case of using a single-channel image with dimensions 32x32, convolving the original image with 40 Gabor filters would result in 40 feature maps, each with a size of 32x32.



Fig. 6 Feature Maps computed by Convolution of Image and Gabor Filters (Tahir et al., 2016)

2.3 The Rise of Deep Learning and Convolutional Neural Networks

Deep learning is a subset of machine learning and has been one of the most significant advancements and areas of intense research in machine learning lately. Deep learning is a form of representation learning in which the machine acquires multiple internal representations from raw data to perform classification or regression tasks (LeCun et al., 2015). This distinguishes it from traditional machine learning algorithms that typically rely on expertly engineered features based on domain knowledge (Bengio et al., 2013). Deep learning models are constructed in a layered structure, with each layer learning a set of hidden representations. These representations are often not easily interpretable by humans. Each layer's representations are composed nonlinearly from the representations of the previous layer. This enables the model to initially learn simple representations in the early layers, such as detecting edges and strokes, which are then combined to form progressively more complex and abstract representations in subsequent layers (Zeiler et al., 2014). By learning from the representations of the preceding layer only, a general-purpose learning algorithm like back propagation (LeCun et al., 1988) can be employed to train a given network.

2.3.1 Convolutional Neural Networks (CNNs)

In 2006, Professor Geoffery Hinton and his student Ruslan Salakhutdinov made a significant contribution to the field of deep learning. They published a paper in a prestigious academic journal called Science (Hinton et al., 2006). The paper highlighted two key points. First, it emphasized the remarkable ability of artificial neural networks with multiple hidden layers to learn complex features. These networks could extract abstract and fundamental representations from the input data through training. Second, the researchers introduced a technique called "layer initialization," which used unsupervised learning algorithms to express hierarchical information in the input data. This method proved effective in reducing the challenges associated with training deep neural networks. Following this breakthrough, the significance of deep learning gained substantial attention in both academic and industrial circles. It led to remarkable advancements in areas such as speech recognition, image recognition, and natural language processing. In the early 1960s,

researchers Hubel and Wiesel made significant progress in understanding the visual cortex of cats. They introduced the concept of receptive fields (Hubel et al., 1962) and discovered how information is processed hierarchically in the visual pathway. Their groundbreaking work earned them the Nobel Prize in Physiology or Medicine. By the mid-1980s, Fukushima et al. (1982) built upon the concept of receptive fields and developed what can be considered the first implementation of Convolutional Neural Networks (CNNs). These networks mimicked the local connectivity and hierarchical structure found in biological neural networks. The idea was to break down visual patterns into subpatterns, which were then processed by cascaded feature planes. This approach proved effective, even for recognizing small objects. Building on this progress, researchers began experimenting with artificial neural networks, specifically using a multi-layer perceptron Ruck et al., (1990) instead of manually extracting features. They applied a simple stochastic gradient descent method and introduced the backpropagation algorithm for calculating error gradients, which proved highly effective (Rumelhart., 1986). LeCun et al. (1990) focused on handwritten digit recognition and introduced the gradient backpropagation algorithm to train convolutional neural network models. They demonstrated superior performance compared to existing methods using the MNIST dataset (LeCun et al., 2010). The success of the gradient backpropagation algorithm and convolutional neural networks brought new hope to the field of machine learning. It paved the way for statistical learning models and propelled the artificial neural network into a phase of rapid development. Today, convolutional neural networks are a hot research topic, particularly in the fields of speech analysis and image recognition. They represent the first successful training of multi-layer neural networks and offer significant advantages when the network input is multidimensional. Convolutional neural networks have been successfully applied to various large-scale machine learning problems, such as speech recognition, image recognition, and natural language processing, contributing to the ongoing exploration and advancement of machine learning.

A convolutional neural network (CNN) is an artificial neural network designed specifically to process two-dimensional input data. It consists of multiple layers, with each layer made up of several two-dimensional planes. Each plane comprises independent neurons that are not connected within the same layer but are connected to adjacent layers. The concept of CNNs draws inspiration from early time delay neural networks (Waibel et al., 1989) and TDNNs, which simplify network training by sharing weights in the time dimension, making them suitable for processing speech and sequential signals.

CNNs adopt a weight-sharing network structure, making them more akin to biological neural networks. The capacity of the model can be adjusted by changing the depth and breadth of the network. CNNs are built on strong assumptions about natural images, such as statistical smoothness and local correlation. These assumptions enable CNNs to effectively reduce the complexity of learning within the network model. They have fewer network connections and weight parameters compared to fully connected networks of comparable size. This characteristic makes CNNs more trainable and manageable.

The diagram in Fig. 7 illustrates the structure of a simple convolutional neural network (CNN) model. This model comprises two convolution layers (C1, C2) and two sub-sampling layers (S1,

S2) that alternate. The process begins with the original input image being convoluted by three pre-trained filters, also known as convolution kernels, along with bias vectors. This convolution operation generates three feature maps in the C1 layer. Next, each feature map in the C1 layer undergoes a weighted averaging process within localized regions. This step is accompanied by the application of a nonlinear activation function. As a result, three new feature maps are obtained in the S1 layer. These feature maps represent the condensed information from the previous layer. The feature maps obtained in the S1 layer are then convoluted with three trained filters from the C2 layer. This process generates three new feature maps, which are passed through the S2 layer. Finally, the output of the S2 layer is vectorized and fed into a traditional neural network for further training.

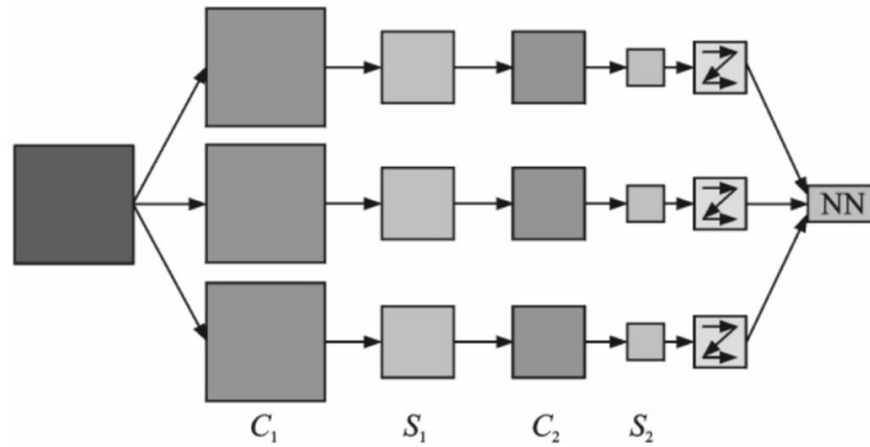


Fig. 7 Simplified convolution neural network structure (Lembaga Ilmu Pengetahuan Indonesia et al., 2017).

2.3.1.1 ResNet

Deep convolutional neural networks have spurred numerous advancements in image classification. These networks seamlessly combine features across low, mid, and high levels in a multi-layered setup, where the complexity of features can be enhanced by the depth of the network—i.e., the number of layers stacked together. There's strong evidence indicating that the depth of a network is critical for its performance. On demanding datasets like ImageNet, the most successful models are those with substantial depth, ranging from sixteen to thirty layers. Additionally, various complex tasks in visual recognition have also shown significant improvements when using these deeply layered models (He et al., 2015).

The paper of He et al. (2015) highlights that the depth of networks (i.e., the number of layers) is crucial for achieving state-of-the-art results on challenging datasets like ImageNet. However, simply increasing the depth of networks can lead to issues such as vanishing/exploding gradients, which impede the training process. Although solutions like normalized initialization and intermediate normalization layers have mitigated these issues, allowing deeper networks to start converging, another problem arises: the degradation of training accuracy as network depth increases. This degradation is notably not due to overfitting, as deeper models sometimes show higher training errors than shallower ones (He et al., 2015).

To address this, the authors introduced a framework called deep residual learning. Instead of expecting each stacked layer to directly learn the desired underlying mapping, they proposed that these layers should focus on learning a residual mapping. This means that instead of fitting the original mapping $H(x)$ directly, they aimed to fit an intermediate mapping $F(x) := H(x) - x$. By recasting the original mapping as $F(x) + x$, we hypothesize that optimizing the residual mapping is easier than optimizing the original mapping without any reference (He et al., 2015).

To implement this framework, He et al. (2015) feedforward neural networks with "shortcut connections," as shown in Fig. 8. Shortcut connections, also known as skip connections, allow bypassing one or more layers. In our case, the shortcut connections simply perform an identity mapping, and their outputs are added to the outputs of the stacked layers. These identity shortcut connections do not introduce additional parameters or computational complexity. The entire network can still be trained end-to-end using stochastic gradient descent (SGD) with backpropagation.

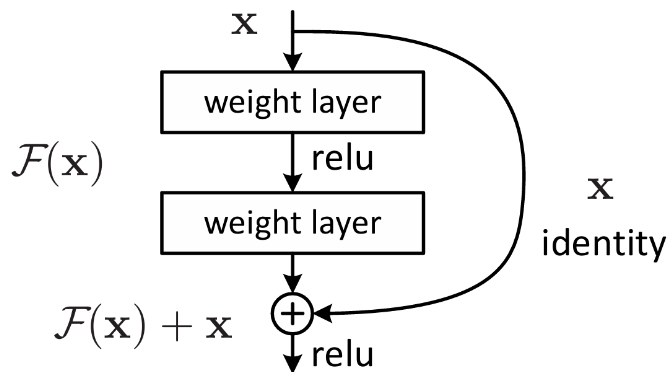


Fig. 8 Residual learning: a building block (He et al., 2015).

He et al. (2015) introduced Residual Networks (ResNets), a new architecture that allows for the training of significantly deeper neural networks by using residual mappings and shortcut connections. This design mitigates the vanishing and exploding gradient problems, enabling the training of networks with depths of up to 152 layers. ResNets demonstrated outstanding performance on the ImageNet dataset, achieving a top-5 error rate of 3.57%, and also performed well across other datasets and tasks, such as COCO for object detection and segmentation. The use of identity and projection shortcuts in these networks facilitates easier optimization and better generalization, setting a new state-of-the-art for deep learning models in image recognition.

Since the introduction of ResNet, researchers have proposed various improvements to enhance the learning capabilities of the network. One such improvement involves the use of identity mapping (He et al., 2016), which establishes direct paths between the residual units and has been found to facilitate training. Another enhancement, proposed by Zagoruyko and Komodakis (2016), suggests the utilization of wider residual units. Experimental results have demonstrated that employing wider residual units in a reasonably deep network leads to performance gains.

Additionally, merge-and-run mappings in residual units (Zhao et al., 2016) aid in the smooth flow of information within the network. These mappings have shown superior performance compared

to alternative approaches. Overall, these advancements in the structure of residual units have contributed to the continuous improvement of ResNet models.

2.3.1.1.1 ResNet18

Residual networks (ResNets) have gained significant popularity due to their impressive performance in tasks like image classification. The key factors contributing to their success are the use of residual mapping and shortcut connections, which offer better results compared to very deep plain networks while also making the training process easier. The deep residual network framework was designed to tackle the issue of degradation observed in deeper neural networks. It employs an innovative approach that incorporates a shortcut, commonly referred to as a skip connection. This skip connection streamlines the transfer of information across layers, effectively circumventing the conventional sequential layer progression typical in traditional Convolutional Neural Networks. Consequently, the network primarily concentrates on adjusting to the residual mappings, as opposed to directly learning from the base mappings. The RESNET has many variants, as shown in Fig. 9. (Venkata Sai Abhishek et al., 2022).

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig. 9 ResNet variants (Venkata Sai Abhishek et al., 2022).

RESNET18 is a neural network architecture that comprises 18 layers, with the first layer utilizing a 7x7 kernel. The network consists of four identical Convolutional Neural Network (ConvNet) layers. Each ConvNet layer is made up of two residual blocks. Each residual block consists of two weight layers, and a skip connection is established from the output of the second weight layer to the main output using a rectified linear unit (ReLU) activation function (Venkata Sai Abhishek et al., 2022).

The skip connection serves an important role in preserving information. If the output of the residual block is equivalent to the input of the ConvNet layer, an identity connection is employed. This means that the skip connection directly passes the input to the output without any alteration. However, if the input and output differ, a convolutional pooling operation is performed on the skip

connection. This pooling operation helps align the dimensions of the input and output, ensuring consistency in the overall network architecture.

In Venkata Sai Abhishek et al.'s (2022) study, the performance of the RESNET18 architecture was enhanced by adding a sequential layer that includes a linear (512, 512) layer. The output from this layer is directed through an initial ReLU activation function, followed by a dropout of 0.2 and another linear layer (512, 2). The process concludes with a LogSoftmax function, which computes the logarithm of the probabilities derived using the negative log-likelihood loss function. This modified network architecture was used to determine the accuracy of a specific image dataset, making it applicable for tasks like classification or detection.

2.3.1.1.2 ResNet50

The architecture of the Resnet-50 network is detailed in Fig. 10, comprising 50 Conv2D operations. The Resnet-50 architecture is segmented into five distinct sections: conv1, conv2_x, conv3_x, conv4_x, and conv5_x. Its structure remains generally static, with variations primarily in the number of channels, which are adjusted based on the specific requirements of the input (Liu et al., 2021).

It is important to note that the final Average Pooling layer in ResNet-50 converts each feature map into a single feature. Consequently, the size of the pooled field matches the size of the feature map. For instance, if the final output dimension is 512x7x7, then the pooled field size would be 7 (Liu et al., 2021).

Liu et al. (2021) explored the application of the ResNet-50 neural network model for the intelligent classification of rock images. The researchers employed the ResNet-50 model to classify images of rocks into seven different categories based on images captured under white light illumination. They enhanced their dataset by segmenting each image into smaller sections, leading to a substantial increase in data volume, which aids in model training and validation. The study achieved a promising classification accuracy of 94.12% on the validation set, demonstrating the effectiveness of using deep learning for this type of image recognition task.

2.3.2 Advanced Techniques for Data Preparation and Model Optimization

2.3.2.1 Image Processing and Recognition

Image processing is a fundamental aspect of working with images, enabling us to perform more advanced tasks and enhance digital photographs. It encompasses a range of techniques, including basic ones like optimizing images and reducing noise, as well as more complex techniques like segmenting images and identifying key elements. The ultimate goal is to enhance the visual appeal and coherence of the images, making them more visually pleasing and understandable.

The field of image processing introduces us to the essential components and teaches us how to apply them in various contexts. In today's world, image processing plays a crucial role in diverse applications, such as satellite monitoring of the environment and enhancing medical images for better interpretation by doctors. It addresses the significant challenge of enabling computers to comprehend and respond to the vast amount of visual data that surrounds us. Photo recognition, at

its core, empowers technology to replicate human visual understanding, allowing computers to identify and analyze objects, scenes, patterns, faces, text, and more.

2.3.2.2 Image Types

2.3.2.2.1 Binary Image

In a binary image, each pixel is assigned one of just two possible values: 1 or 0. In many software programs like MATLAB, a pixel with a value of 1 typically represents an area of interest, while a pixel with a value of 0 is considered the background. Binary images are often used alongside other types of images to highlight specific areas for processing. Fig. 10 provides a detailed view of some of these pixel values (Matlab Image Processing Toolbox User's Guide, 2023).

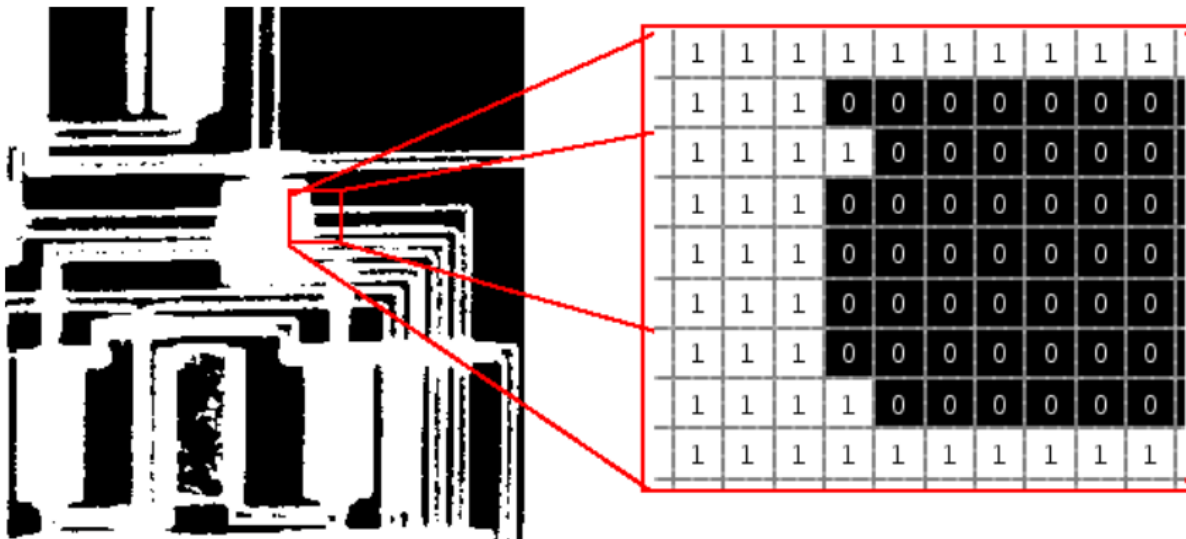


Fig. 10 Binary image (Matlab Image Processing Toolbox User's Guide, 2023).

2.3.2.2.2 Indexed Images

An indexed image is composed of two main components: an image matrix and a colormap. The colormap is a matrix of size c -by-3, where each row represents the red, green, and blue components of a specific color. These values are typically in the range of 0 to 1 (Matlab Image Processing Toolbox User's Guide, 2023).

The pixel values in the image matrix serve as indices in the colormap. This means that the color of each pixel in the indexed image is determined by finding the corresponding color in the colormap based on the pixel value. The specific mapping process depends on the data type of the image matrix (Matlab Image Processing Toolbox User's Guide, 2023):

- If the image matrix is of type single or double, the colormap usually contains integer values ranging from 1 to p , where p is the length of the colormap. The value 1 corresponds to the first row in the colormap, 2 corresponds to the second row, and so on.
- If the image matrix is of type logical, uint8, or uint16, the colormap normally contains integer values ranging from 0 to $p-1$. Here, the value 0 corresponds to the first row in the colormap, 1 corresponds to the second row, and so forth.

Typically, a colormap is stored together with an indexed image and is automatically loaded when using the `imread` function. When you read the image and colormap into your workspace as separate variables, it's important to keep track of the association between the image and the colormap. However, you have the freedom to choose any colormap you prefer, and you are not limited to using the default colormap.

Fig. 11 illustrates an indexed image, showcasing the image matrix and the colormap. In this example, the image matrix is of data type `double`, which means that a pixel value of 7 corresponds to the seventh row of the colormap.

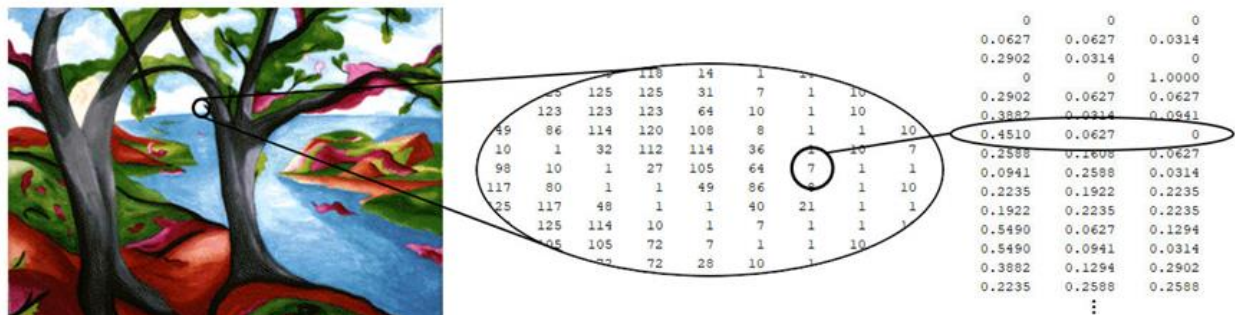


Fig. 11 indexed image (MATLAB Image Processing Toolbox User's Guide, 2023).

2.3.2.3 Histogram of an Image

Image processing is performed through image transformations, which are executed using operators. These operators take an input image and produce an output image. The goal of image enhancement is accomplished through two major approaches. The first approach involves reasoning about the statistics of the grey values in the image, while the second approach focuses on the spatial frequency content of the image (Image Processing the Fundamentals, 1999).

One fundamental concept in image processing is the histogram of an image. It is a discrete function that counts the number of pixels in the image with specific grey values. By normalizing this function to add up to 1 for all grey values, it becomes a probability density function. This probability density function provides insights into the likelihood of encountering a certain grey value within the image (Image Processing the Fundamentals, 1999).

Histogram equalization is a technique used to enhance images by making all grey values equally probable (Fig. 12, Fig. 13, Fig. 14). It achieves this by redistributing the grey values to create a more balanced representation. However, it is important to note that histogram equalization programs typically do not produce images with perfectly flat histograms. Instead, they aim to improve the overall distribution of grey values and enhance the visual quality of the image (Image Processing: The Fundamentals, 1999).

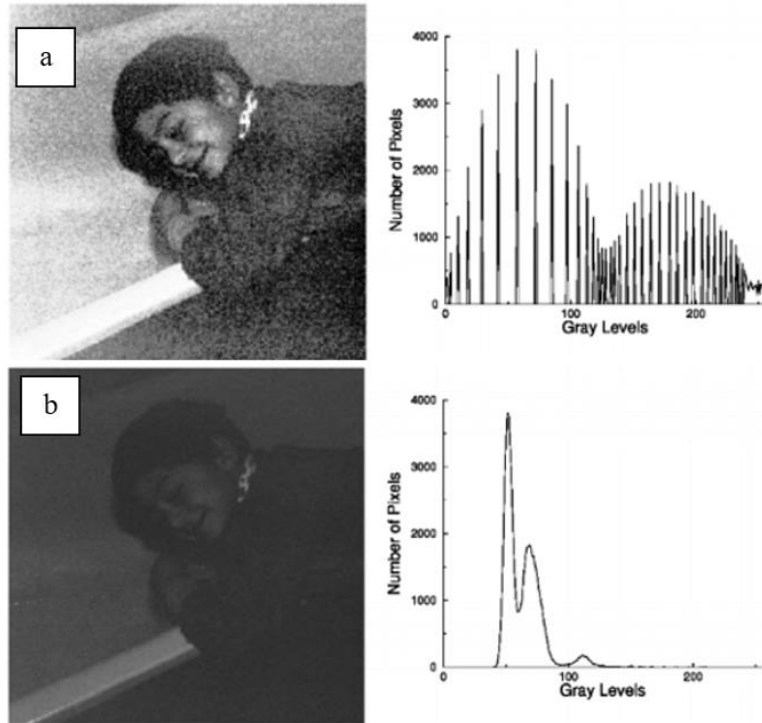


Fig. 12 (a) original image (b) after histogram equalization(Image Processing the Fundamentals, 1999).



Fig. 13 (a) Original image (b) After global histogram equalization (c) After local histogram equalization(Image Processing: The Fundamentals, 1999).



Fig. 14 (a) original image (b) after global histogram globalization(Image Processing The Fundamentals, 1999)

2.4 Conclusion

This literature review has thoroughly examined the effectiveness and adaptability of ResNet architectures, specifically ResNet-18 and ResNet-50, in the domain of image classification when dealing with complex and imbalanced datasets. These models have demonstrated impressive capabilities in handling intricate data structures, which is crucial for achieving unbiased and accurate results in various AI applications. The review emphasized the innovative features of ResNet architectures, such as skip connections, which play a vital role in addressing the vanishing gradient problem and enabling the training of deeper networks.

Moreover, the review delved into the performance of these architectures in scenarios with imbalanced datasets and variable computational resources. The insights gathered indicate that ResNet models, with their deep learning capabilities, are not only theoretically significant but also adaptable to real-world challenges. They exhibit scalability and maintain high performance even when faced with changes in dataset size and complexity.

Looking ahead, I continued to explore and expand upon the advancements in ResNet models and their underlying principles across several operational areas, rigorously testing them under a wide variety of aspects and conditions to thoroughly assess their versatility and robustness.

3 Material and Methods

3.1 Linearoch: Systematizing Complex Data Structures for Enhanced Machine Learning Preparation

The proper preparation and organization of data are of paramount importance for the effective development and performance of predictive models. Efficient data management improves the overall efficiency of the machine learning process.

Introducing the "Linearoch" script marks a significant milestone in this domain, as it has been specifically designed to streamline the handling of intricate data structures. This script automates the tasks involved in managing, preprocessing, and organizing extensive and heterogeneous datasets, thereby simplifying the process of preparing data for machine learning applications.

The "Linearoch" script addresses a range of common challenges encountered in data management, such as the need to reorganize folder structures, rename files while preserving hierarchical relationships, and handle various types of images with meticulous care. Through the automation of these tasks, the script not only reduces the potential for human error but also significantly decreases the time and effort typically associated with data preparation. The core functionality of Linearoch lies in its ability to transform unstructured or semi-structured data repositories into well-organized formats that are easily accessible and ready for further processing and analysis. Linearoch plays a pivotal role in enabling the effective utilization of data for building reliable and scalable machine learning models.

I encountered a complex system folder comprising numerous folders and subfolders (Fig. a15), each containing various types of parts and diverse surface characteristics (Fig. b15).

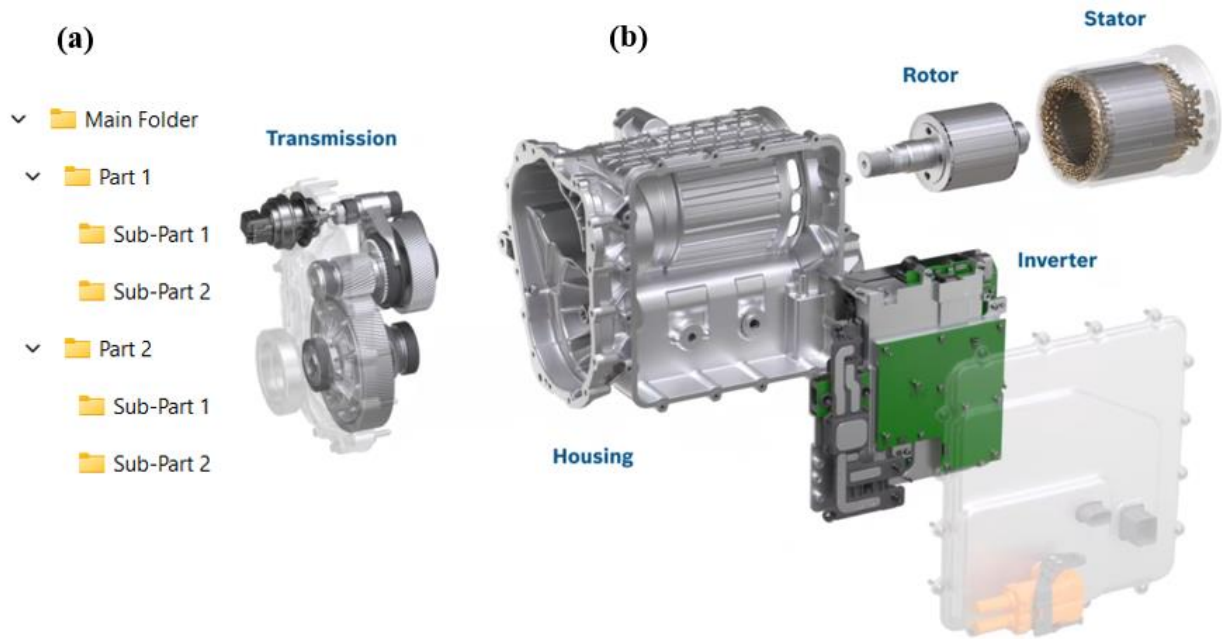


Fig. 15 (a) Complex system folder (b) eAxle exploded view (Bosch Mobility Solutions, 2024).

In addition, there is a source folder where all the photos for the sample are saved in an unsorted manner. This folder contains photos, videos, folders, and thumbs files. The data we have varies from sample to sample, resulting in an imbalance in the amount of data for each class. This imbalance presents challenges during the data preparation stage. To address this, I developed a script called "Linearoch" that automates the handling of the classified data, ensuring it is ready for the training step.

The Linearoch script consists of 14 main functions, with two switchable functions for different configurations, built upon 21 subfunctions (Fig. 16). A Master Function is included to control the flow of the script, making it easier to track the functions and facilitate debugging.

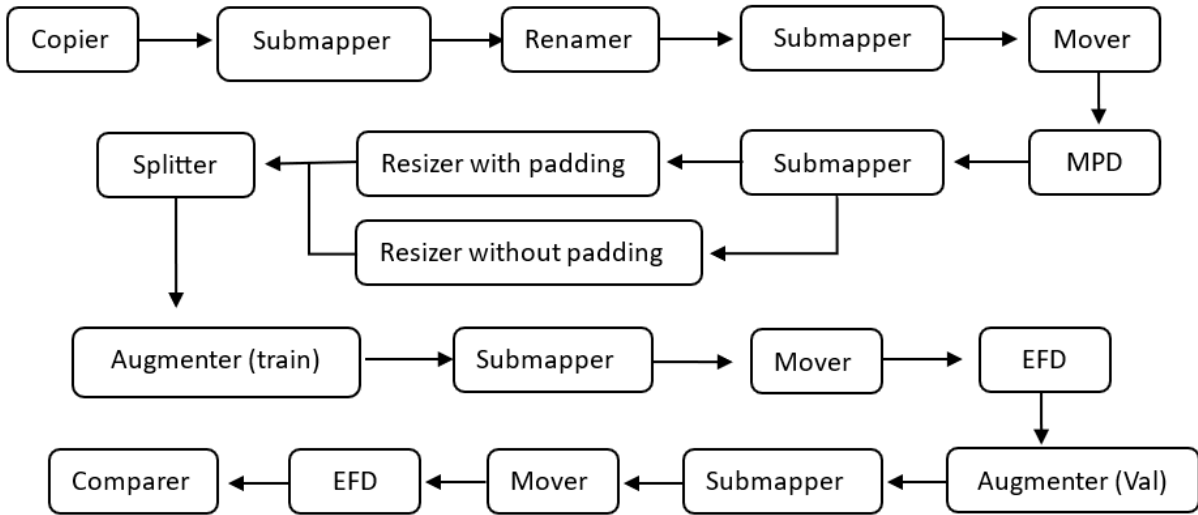


Fig. 16 Workflow of Linearoch script.

3.1.1 Copying the Dataset

The script performs a crucial operation of creating an exact copy of the original dataset and saving it in a designated location called "AI_database." During this process, specific files (e.g., 'Thumbs.db') that may cause issues during the augmentation step are excluded. This operation is essential for preserving the integrity of the original data, as it allows for manipulations and transformations to be conducted on the duplicate without the risk of altering or damaging the source files (Fig. 17). By specifying the source folder, destination base, and an optional new name for the copied folder, this function facilitates a seamless transition of the dataset from its original state to a workable copy. This step sets the stage for subsequent processing steps, such as renaming, resizing, and augmenting the images, by providing a reliable and preserved dataset for further operations.

3.1.2 Submapper

The "Submapper" function plays a key role in the script by listing the subdirectories within a specified directory. This operation is crucial for preparing the data for various processing tasks, such as renaming, moving, or copying. The function collects the paths of all subfolders (partial paths) and sorts them based on their length. This sorting process is essential for the subsequent

step, which involves the renaming function. By sorting the subfolders' paths, the function ensures that there are no errors during the renaming process. This sorting process is particularly important because it prevents mistakes that could occur if the renaming function were to change the name of a parent folder. Such a change would alter the directory of the child folder, leading to errors when attempting to rename the child folder.

The output of the Submapper function will be a list of subfolder paths sorted as follows:
['Part 1/Sub-Part 1', 'Part 1/Sub-Part 2', 'Part 2/Sub-Part 1', 'Part 2/Sub-Part 2'].

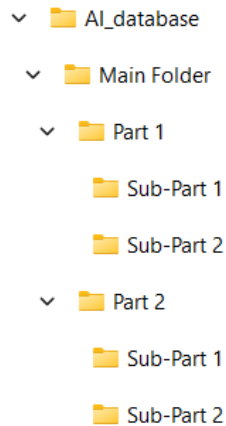


Fig. 17 Copied original data set structure.

3.1.3 Renamer

The “Renamer” function is tasked with systematically renaming subdirectories within a specified base directory by appending the name of each directory's parent as a prefix. It will iterate through the Submapper list from the previous step, then extract and append the parent directory's name to the “Main folder” directory's name; therefore, it transforms the directories to include a hierarchical context. This function preserves and clarifies the structural relationships within the directory tree to contain the whole paths, thereby enhancing navigability to run ‘os.rename’ for the renaming operation, which directly modifies the filesystem to reflect these changes (Fig. 18).

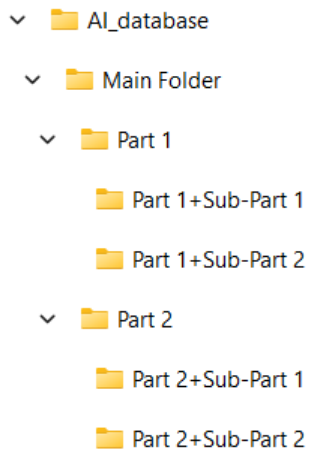


Fig. 18 After renaming function.

3.1.4 Submapper

After modifying the names of the subfolders to include the parent folder's name, it is necessary to update the list of subfolder names to reflect the changes. As a result, the Submapper function needs to be called again.

The output of this subsequent call to the Submapper function will be as follows:

```
['Part 1/Part 1+Sub-Part 1', 'Part 1/Part 1+Sub-Part 2', 'Part 2/Part 2+Sub-Part 1', 'Part 2/Part 2+Sub-Part 2']
```

It is worth noting how the Renamer function has modified the names of the subfolders to include the parent folder's name as a prefix.

3.1.5 Mover

This function orchestrates the movement of the subfolders, which are specified in the list created in the previous step, to a single destination called the "Main Folder" as shown in Fig. 19, located alongside their respective parent folders. The function combines the main folder's path with the partial path of each subfolder, and then the "move_folder" function is called sequentially to execute the actual transfer. By using this function, each folder retains its original name while being relocated to a new centralized location. This step is crucial for data preprocessing pipelines that require organizing data into specific directory structures for machine learning training. After executing this script, the "Main Folder" will contain only one layer of folders, as depicted in the accompanying image.

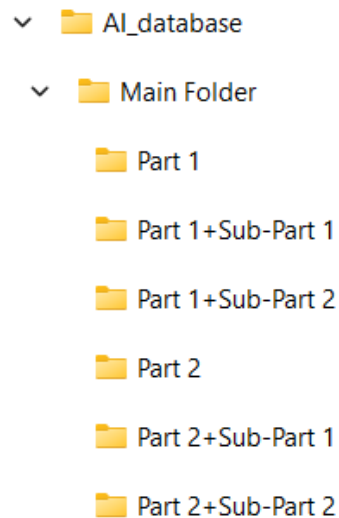


Fig. 19 Mover function result.

The purpose of having a one-layer folder structure is crucial because performing augmentation runs on sub-subfolders can be challenging. Hence, the augmentation process is designed to operate exclusively on the one-layer folder system. Additionally, these folders will serve as the names of the classes in the dataset.

3.1.6 MPD

The Minimum Photos Determiner (MPD) function enables the determination of a minimum number of photos per class. This step is crucial for maintaining a balanced dataset, which is vital when preparing data for machine learning. Sufficient data in each category is essential for training effective models. The MPD function plays a key role in this process by identifying and removing sparsely populated folders, ensuring that each class has an adequate number of photos.

3.1.7 Submapper

The Submapper must be called again to refresh the list of folders after moving the subfolders to a different directory. This preparation is necessary for the next step, resizing, because we need resizing to occur on all the folders in the 'Main Folder' after moving and deleting the smaller ones.

3.1.8 Resizing

3.1.8.1 Resizing without Padding

This function is engineered to resize images while maintaining their aspect ratios. Capable of processing multiple images simultaneously, it offers heightened efficiency and reduces time expenditure. Furthermore, it operates across several cores concurrently.

Fig. 20 compares the performance of the conventional resizing method with that of the parallel resizing method:

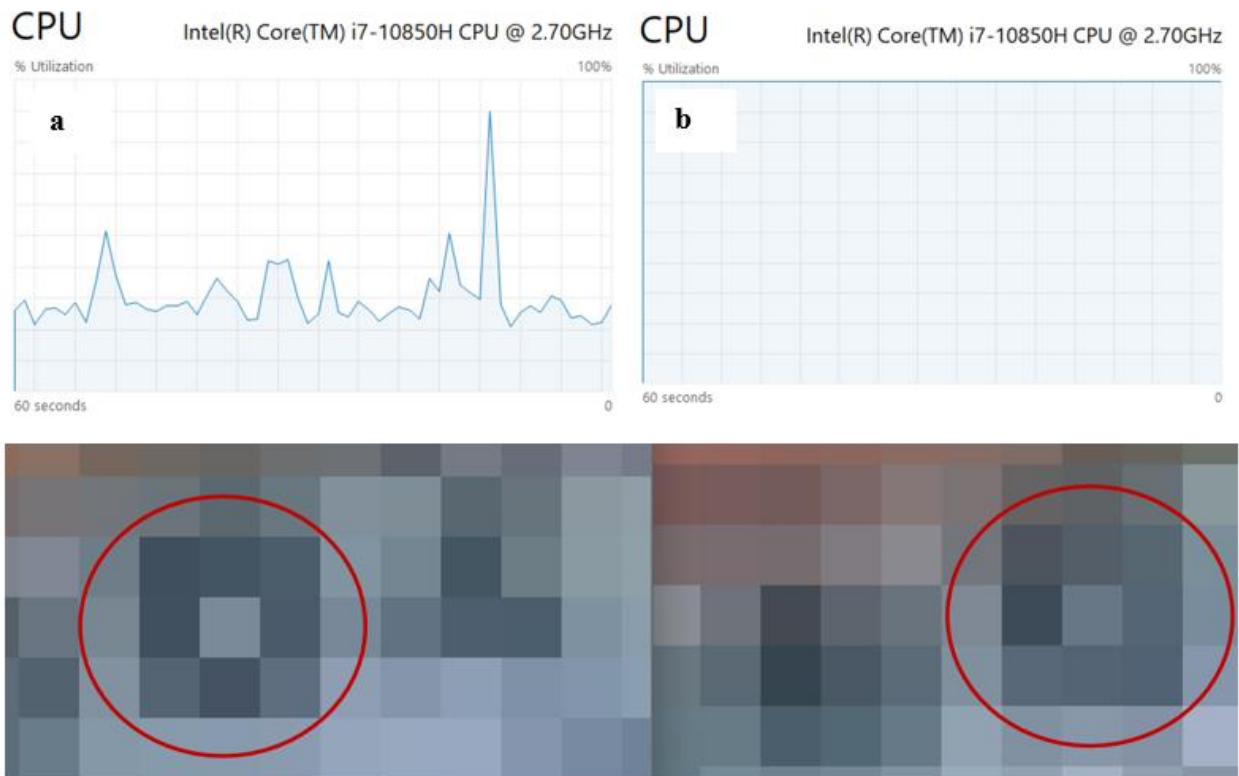


Fig. 20 (a) Conventional resizing method (b) Parallel resizing method.

While there are some differences in the pixel values, as shown in Fig. 20, these do not significantly impact the accuracy results of the AI model.

3.1.8.2 Resizing with padding

The idea behind this method is to train the model on its designed photo dimensions, rather than on rectangular photos, and without stretching the images into square shapes. One proposed solution was to add backgrounds to the images during resizing. This function operates by first calculating the scaling ratio necessary to fit the image within the target size without distortion. It then resizes the image using this ratio to ensure the dimensions are proportionally scaled. After resizing, the function creates a new image canvas with the target dimensions, filled with a specified padding color (default is white).

The resized image is then centered on this canvas, filling any excess space with the padding color. This method preserves the aspect ratio of the photos without cropping them to fit the model. However, it reduces the number of useful pixels since many pixels will lack information, requiring more photos to train the model to disregard these backgrounds.

3.1.9 Splitter

The splitter function is designed to organize a collection of files within a specified directory into training and validation sets, adhering to a predefined ratio. It operates by first identifying all files within the subfolders of the base directory, shuffling them to ensure randomness, and then dividing them according to the specified validation-to-training ratio.

After the split, the files are relocated to new 'train' and 'val' directories, which mirror the structure of the original base directory.

3.1.10 Augmenter

During this stage, the function utilizes image augmentation techniques on each image in the subfolders, thereby expanding the training dataset. The modifications include operations such as rotation, flipping, zooming, distortion, brightness adjustment, and contrast enhancement. Initially, the function identifies the maximum number of images in any subfolder to establish a benchmark for augmentation across all subfolders, ensuring dataset balance. It then iterates through each subfolder, applying a calculated number of augmentation operations to each image. This calculation is based on the initial image count in that subfolder relative to the subfolder with the most images. This relative scaling of augmentation ensures that subfolders with fewer images receive more augmented versions to proportionally match the larger ones.

3.1.11 Submapper

After applying the augmentation, all the augmented photos will be placed inside an output folder. To relocate these photos to their original folders alongside their original counterparts, the Submapper function must be called again to refresh the list of subfolder directories.

3.1.12 Mover

After refreshing the list, it is possible to run the Mover function, which will merge the partial directories of the output folders with their parent folders, facilitating the transfer.

3.1.13 EFD

In the Empty Folders Deleter (EFD) step of the script, the function is utilized to identify and remove any subfolders that do not contain any files or subdirectories. This function is crucial for maintaining an efficient and clutter-free file system, particularly after data has been moved or reorganized, which can often leave behind empty directories. It operates by traversing the directory tree from the innermost to the outermost directories (using a depth-first search approach), checking each folder for its contents. If a folder is found to be empty, it is immediately removed from the file system.

The augmentation should be run on the 'train' and 'val' folders individually, which means that all the previous four steps (Augmenter, Submapper, Mover, EFD) must be run twice.

3.1.14 Comparer

This function is designed to effectively synchronize directory structures between two specified directories. This synchronization is crucial for maintaining consistency and alignment, especially in environments where data is dispersed across multiple locations, such as training and validation datasets for machine learning models. The function operates by initially listing all subfolders in both directories. It then identifies and removes any subfolders that lack a corresponding counterpart in the other directory, ensuring each location mirrors the other accurately.

The "Linearoch" script represents a notable advancement in the mechanization of data processing for machine learning applications. This tool improves the efficiency and efficacy of data preparation by organizing the workflow for complicated data structures. It also offers a level of adaptability that was not possible before. Linearoch facilitates the easy integration of new components or projects into the machine-learning process. Users only need to place their photographs in the specified directories, and Linearoch takes care of the rest, including organizing and purifying the data, as well as executing advanced enhancements. This guarantees that the dataset is prepared and optimized for training, eliminating the necessity for manual intervention. Automation not only conserves significant time and resources but also enables the expansion of machine learning procedures and the exploration of new datasets with minimal preparation. Linearoch is an exceptional tool in the field of artificial intelligence, offering a powerful solution for seamlessly integrating and preparing various forms of data for future AI projects or expansions.

3.2 Tailored Approaches for Peak Performance in Image Classification

Image classification has come a long way in the realm of artificial intelligence, playing a vital role in various industries. Thanks to the adoption of deep learning technologies, significant progress has been made, greatly improving accuracy and efficiency in practical applications. This progress has been made possible by the development of sophisticated tools and scripts that streamline the training and testing processes, enabling more effective implementations. In this thesis, I present a Python script specifically designed to optimize performance in image classification tasks.

Leveraging the powerful PyTorch library, this script offers a comprehensive set of tools that support every stage of the machine learning journey, from data preparation to model training and performance evaluation.

The script's architecture is meticulously crafted to accommodate a flexible and adaptable workflow, catering to diverse project requirements and dataset intricacies. This flexibility is achieved through a user-friendly interface that allows for easy customization of the machine learning process. Starting with a robust argument parsing module, the script empowers users to define crucial training parameters, such as batch size, learning rate, and the number of epochs. These parameters can be fine-tuned to suit different training scenarios and objectives, offering the flexibility needed to optimize performance across various computing environments.

Furthermore, the script provides users with the ability to choose from a range of model architectures. Options include simpler and more efficient models like ResNet18, as well as more complex and computationally demanding ones like ResNet50. This feature enables users to strike an optimal balance between computational load and model performance, taking into account available hardware resources and specific project requirements. Such configurability is essential for tailoring the system to deliver the best possible results within the constraints and objectives of each unique use case.

A critical aspect of the script's effectiveness lies in its advanced data management system. It not only facilitates efficient loading and preprocessing of image data but also incorporates advanced techniques like data augmentation. Data augmentation plays a crucial role in enhancing the model's ability to generalize from training data to real-world scenarios, ensuring robustness in AI applications. Additionally, the script leverages mixed-precision training, utilizing PyTorch's autocast and GradScaler features. This approach optimizes GPU utilization and accelerates the training process without compromising the accuracy of the model.

Moreover, the script adopts the F1 score as the primary metric for evaluating model performance. The F1 score, which combines precision and recall, is particularly well-suited for scenarios where a balanced measure of model effectiveness is essential, such as datasets with imbalanced classes. By considering both precision and recall, this metric promotes a holistic assessment of the model's performance.

The training process is further enriched by incorporating periodic checkpoints and extensive logging capabilities through TensorBoard. These features enable continuous monitoring and in-depth analysis of the model's progress and performance, providing valuable insights to guide further optimizations. By visualizing training metrics in detail, TensorBoard helps identify trends and issues in real time, facilitating timely adjustments to the training process.

In summary, the script developed as part of this thesis is not just a tool but the culmination of meticulous research, rigorous testing, and continuous refinement. It represents the most effective and innovative practices in the field of machine learning for image classification. The result is a highly adaptable, efficient, and powerful methodology that significantly enhances the capabilities of image classification systems. It demonstrates the profound impact that tailored, cutting-edge approaches can have on the advancement of artificial intelligence.

4 Results

In this section of the thesis, we dive into the practical evaluation of the "Tailored Approaches for Peak Performance in Image Classification" script. Our main focus is to understand how different configurations and methodologies affect the effectiveness and efficiency of the image classification process. The experiments were carefully designed to systematically test the influence of various parameters. These parameters include factors like the size of the dataset, the dimensions of the images, the type of model used, when to perform augmentation, how to group the classes, whether to include padding and how much data augmentation to apply. Each of these factors plays a critical role in optimizing the performance and accuracy of the machine learning models that were used.

The methodological approach employed in the experiments focused on systematically identifying optimal settings and methods to enhance the training and prediction accuracy of the model. Several factors were investigated, including the size of the dataset and image dimensions, the selection of different model architectures (ResNet18 and ResNet50), the timing of image augmentation (before or after dataset splitting), the inclusion of padding in images, and the extent of augmentation. These choices were made to improve the robustness and accuracy of the classification models by refining the training process, reducing overfitting, and enhancing generalization capabilities. The detailed results from these experiments provide insights into the best configurations and methodologies for achieving high performance in image classification tasks, validating the effectiveness of the approach and offering practical guidance for configuring image classification systems.

Building upon the foundational methodologies we discussed earlier, the next section takes a deep dive into a careful evaluation of specific configuration parameters that are crucial for optimizing image classification models. Through this thorough analysis, we aim to shed light on the subtle yet significant influence of each variable, including factors like the amount of data available, the size and dimensions of the images, the choice of neural network architecture, and the preprocessing techniques applied. Our objective is to methodically measure the impact of these parameters on both the effectiveness and efficiency of the models. By doing so, we establish a solid empirical basis that can guide strategic decision-making when it comes to deploying image classification systems.

4.1 Systematic Exploration of Configuration Parameters Impacting Image Classification Efficacy

This section conducts a detailed examination of key configuration variables influencing the performance of image classification models.

4.1.1 Data Size

When examining the impact of data size on the performance of image classification models, a trend emerges where increased dataset volume correlates with enhanced model accuracy and generalization capability. The dataset of 4000 images, while showing substantial improvement over training epochs, reached a plateau in validation accuracy at 88% (Fig. 21). This plateau suggests a limitation in the model's ability to learn further from the available data. Real-world application of this model yielded a correct classification rate of approximately 13.3%, a value that

represents an average derived from observed performance on different test samples.

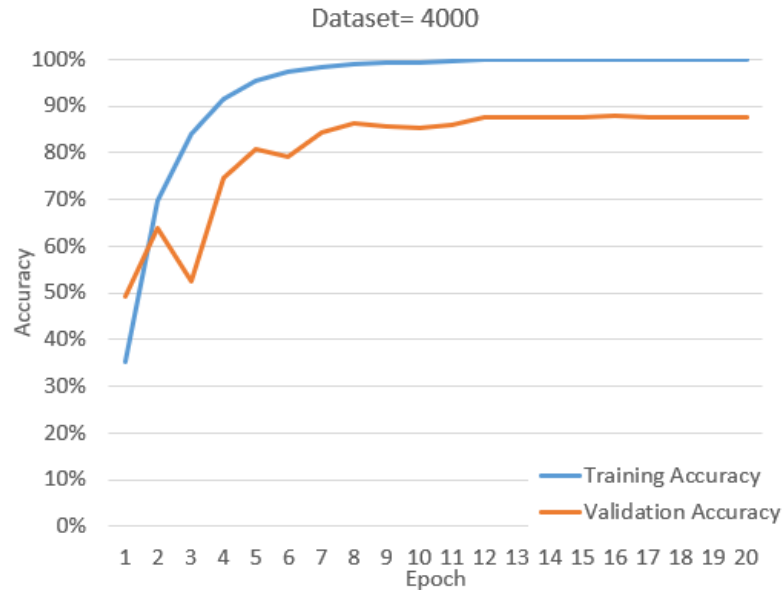


Fig. 21 Model performance (4000 images).

Expanding the dataset to 7000 images resulted in a notable increase in validation accuracy, achieving a peak of 92% (Fig. 22). This improvement in the model's validation performance translated into a higher correct classification rate, averaging 27.55% when tested on independent samples. The increased data volume provides the model with a richer set of features and examples to learn from, thereby improving its predictive accuracy.

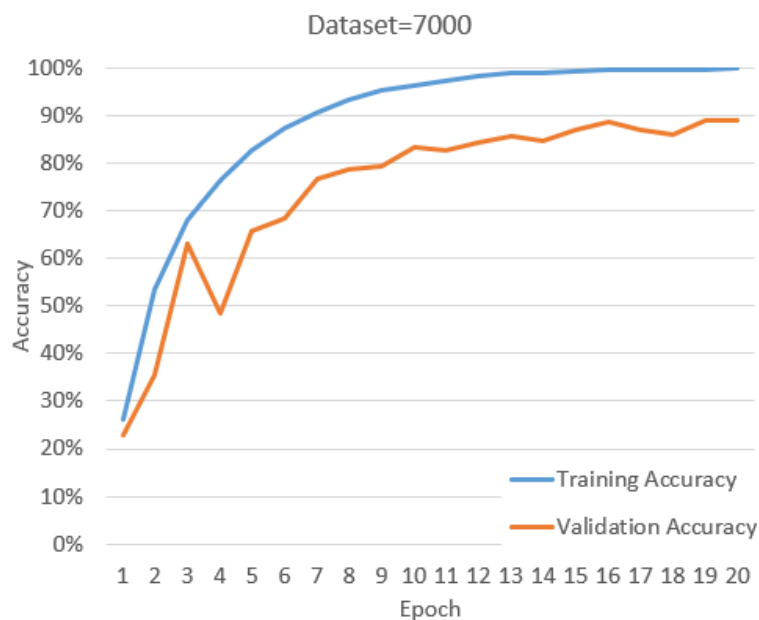


Fig. 22 Model performance (7000 images).

The dataset comprising 10000 images further underscored this relationship. The model maintained a high validation accuracy of 92% (Fig. 23), consistent with the smaller 7000 image dataset. However, the correct classification rate on new samples significantly increased, averaging 39.5%. This notable increase implies that the additional data points contributed to a more robust representation of the underlying patterns within the image classes, allowing the model to better generalize and thus perform more effectively on unseen data.

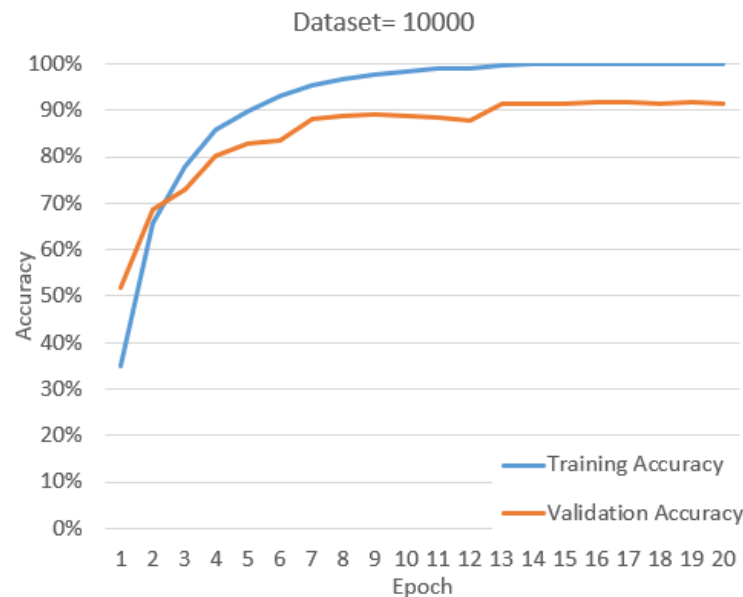


Fig.23 Model performance (10000 images).

The consistency of high validation accuracy across the larger datasets indicates that the models have adequately learned the distinguishing features of each class. However, the varying correct classification rates point to the fact that accuracy on a validation set within the same distribution as the training set may not fully represent a model's performance in real-world scenarios, where the data may deviate from the training distribution. These findings emphasize the importance of not only training with ample data but also ensuring diversity within the dataset to bridge the gap between validation performance and practical applicability (Fig. 24).

The relationship between data size and model performance is indeed not strictly linear, and this phenomenon is evident in the progression of real-world classification rates across the models trained on datasets of 4000, 7000, and 10000 photos. As the amount of data increases, there is an initial surge in performance; however, this rate of improvement begins to taper off, indicating a point of diminishing returns. This implies that each additional image contributes less to the model's ability to generalize as the dataset grows larger. When examining the transition from a dataset of 4000 to 7000 images, the increase in correct classification rates is quite pronounced. Yet, the subsequent increase from 7000 to 10000 images, while still leading to better performance, does not yield as significant an improvement as the previous increment. This suggests the model is approaching a saturation point where additional data has a progressively smaller impact on performance.

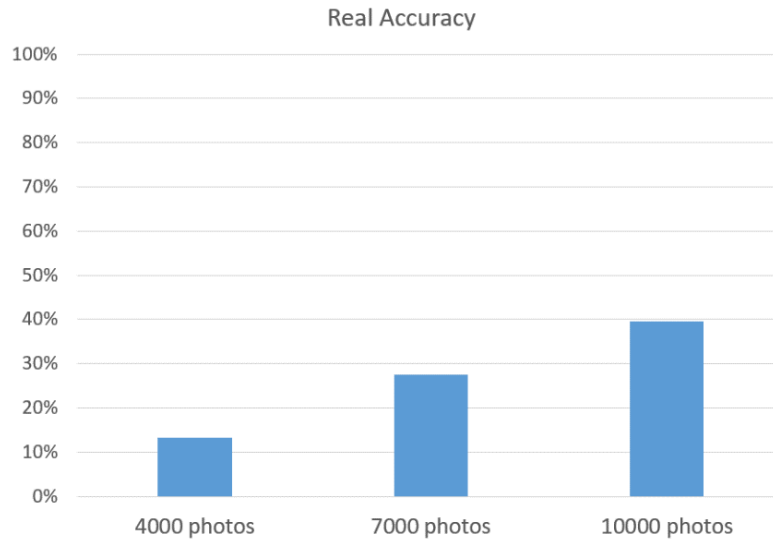


Fig. 24 Real Accuracy.

The saturation point is influenced by several factors, such as the inherent complexity of the image data and the diversity encapsulated within the dataset. Complex images containing intricate patterns or subtle variations require more data to capture all the nuances necessary for accurate classification. Conversely, datasets with a high degree of diversity in terms of classes, backgrounds, lighting conditions, and perspectives offer a broader range of features for the model to learn from, which can prolong the linear portion of the performance curve before plateauing.

It's also important to consider that as datasets grow, they may include more redundant or less informative images, which have a lesser effect on enhancing the model's robustness. Identifying the optimal dataset size thus becomes a balance between acquiring sufficient data to capture the diversity of the problem space and avoiding unnecessary data that does not contribute to model improvement. This balance is crucial for efficient model training, ensuring resources are not expended on processing data that yields minimal returns in performance enhancement.

4.1.2 Model Type

Comparing the ResNet18 and ResNet50 models provides insight into how different architectures perform with varying dataset sizes. On a dataset of 7000 photos, ResNet18 achieved a peak validation accuracy of 89% by the end of the training epochs (Fig. 25). When deployed in real-world scenarios, this model demonstrated an average correct classification rate of 27% across different sample sets.

On the same dataset, ResNet50 showed a slightly lower peak validation accuracy of 86% (Fig. 26). However, in practical applications, it managed to yield a higher correct classification rate, ranging from 23.4% to 31.7%, which indicates that the increased complexity of the architecture may provide better generalizability under certain conditions.

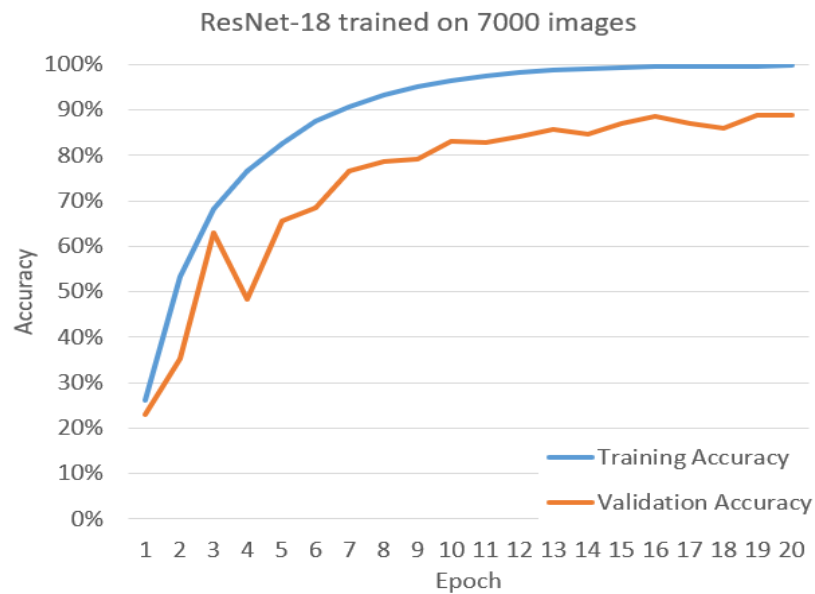


Fig. 25 ResNet-18 trained on 7000 images.

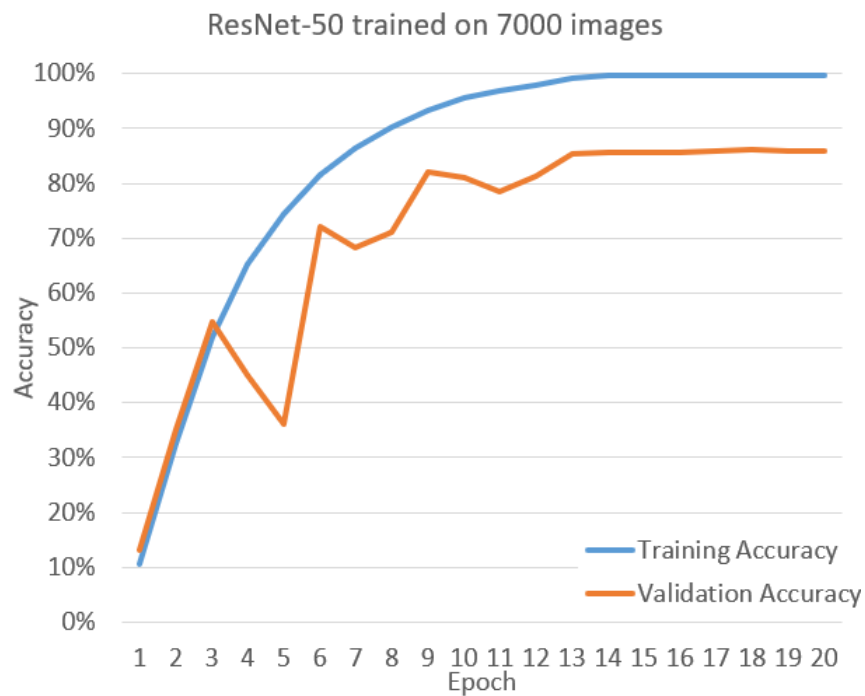


Fig. 26 ResNet-50 trained on 7000 images.

For the 10000-photo dataset, the pattern continues. ResNet18 shows strong learning, reaching a validation accuracy of 93% with real-world classification rates between 32% and 48%. ResNet50 also peaks at a 90% validation accuracy but with a real-world performance ranging from 31% to

47% (Fig. 27). These findings highlight that ResNet50 does not always outperform ResNet18 significantly despite its increased complexity.

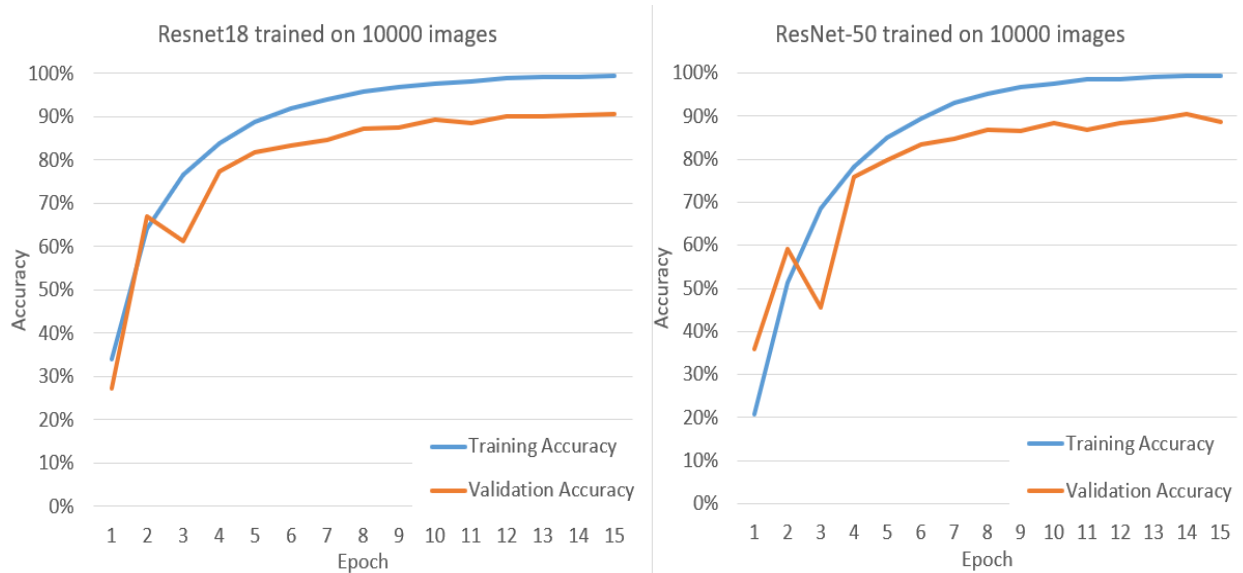


Fig. 27 ResNet-18 and ResNet-50 trained on 10000 images.

In conclusion, the choice of model architecture should be driven by the specific requirements of the task and the computational resources at hand. While ResNet50 may have a higher potential for learning complex features, ResNet18 provides a computationally efficient alternative without compromising much on performance, proving to be a suitable choice for datasets of this size. These insights into the behavior of different architectures are instrumental in guiding the selection process for future image classification tasks.

4.1.3 Image Size

The impact of image size on the effectiveness of convolutional neural network models, specifically ResNet18 and ResNet50, is a topic of significant interest in the field of image classification. By comparing the classification rates of two different image resolutions (224,224) and (299,299), one can discern the influence of image resolution on model performance.

For ResNet18, the model designed to handle (224,224) images showed a correct classification rate of 30% on sample 1 and 43% on sample 2. When the image size was increased to (299,299), which offers more pixel data and potentially more detailed features for the model to learn, the correct classification rate improved to 32% for sample 1 and 48% for sample 2. This improvement suggests that the additional resolution can be beneficial, possibly due to the model's ability to discern finer details that contribute to the accuracy of classification (Fig. 28).

Similarly, the ResNet50 model experienced a gain in performance with the larger image size. For sample 1, the classification rate increased from 28% with (224,224) images to 31% with (299,299) images. For sample 2, the increase was from 39% to 47%. These results align with those observed

for ResNet18, reinforcing the notion that higher image resolutions can enhance model performance.

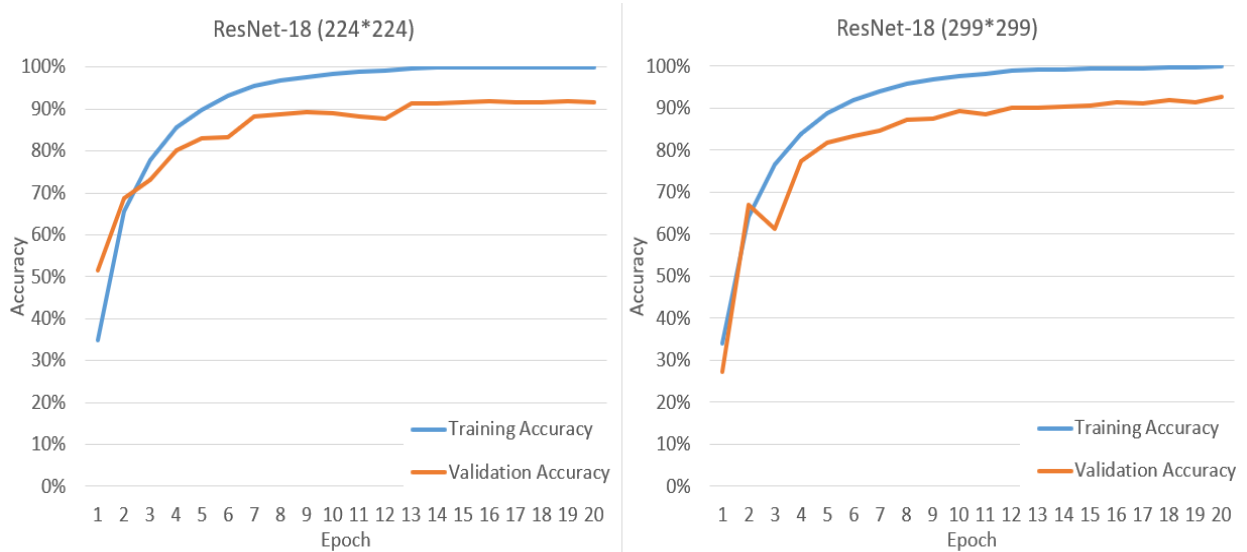


Fig. 28 ResNet-18 (224,224) and ResNet-50 (299,299).

It is important to note that while higher resolutions generally lead to better classification rates, the increase is not overly dramatic. This may indicate that beyond a certain resolution, the added detail does not significantly contribute to model performance, particularly if the model is initially designed for a lower resolution. Moreover, the computational cost of processing higher-resolution images is substantially greater, which must be weighed against the incremental gains in accuracy.

In summary, while larger image sizes can provide a performance boost, the trade-off between improved accuracy and increased computational demand must be carefully considered. For applications where fine-grained detail is crucial for accurate classification, higher resolutions may be justified. However, for more general purposes, the standard (224,224) size might offer a more balanced approach, especially when resource optimization is a priority.

4.1.4 Augmentation Timing

The sequence of augmentation and dataset splitting plays a pivotal role in the training of neural networks. Augmentation before splitting can inflate validation accuracy due to the augmented images in the training set resembling those in the validation set, creating an overestimation of the model's ability to generalize (Fig. 29). On the other hand, augmentation after splitting ensures that the validation set remains a more accurate gauge of real-world performance, as it better simulates the introduction of novel images that the model has not encountered during training.

In the given experiment, applying augmentation after the split presents a more reliable progression of validation accuracy, starting from a high of 40% and then showing variation, with the highest accuracy plateauing around 76% (Fig. 30). This variance and the lower peak suggest that the model is learning from a more diverse and less overlapping dataset, which can result in a better measure of true generalizability.

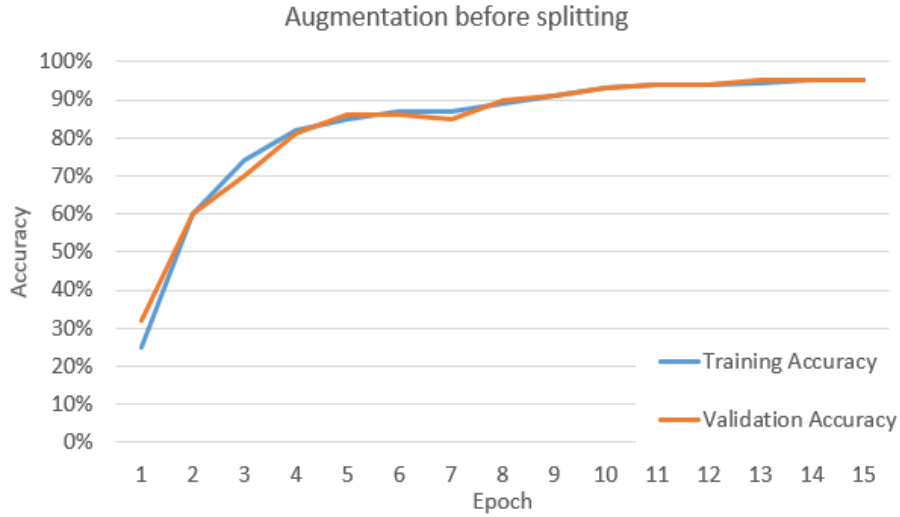


Fig. 29 Augmentation before splitting.

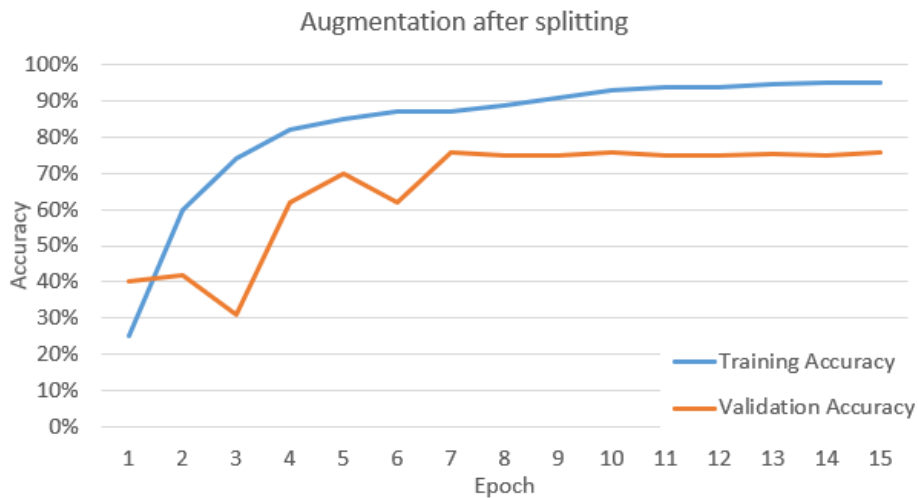


Fig. 30 Augmentation after splitting.

The initial higher validation accuracy when augmenting before splitting is potentially misleading due to the similarity between training and validation images. This could lead to the false conclusion that the model is performing well on unseen data when, in reality, it has learned to recognize the augmented features that appear in both sets.

The recommendation from these findings is to split the dataset before augmentation. This practice allows for a more honest assessment of a model's performance, preventing information leakage between training and validation datasets and ensuring that the validation set acts as a stand-in for entirely new data. Thus, augmentation after splitting contributes to building a model that generalizes well to unseen images, which is a critical requirement for real-world applications.

4.1.5 Padding inclusion

The adaptation of image preprocessing strategies, specifically the application of padding during resizing, plays a crucial role in aligning the dataset with the architectural requisites of Convolutional Neural Networks (CNNs) such as ResNet18 and ResNet50. This section delves into the impact of padding on image classification accuracy by comparing various padding approaches against resizing that maintains the original aspect ratio without padding.

4.1.5.1 Analysis of ResNet18 Trained on 4000 Images

4.1.5.1.1 Aspect Ratio Preserved without Padding (348x232)

This method directly resizes images to fit one dimension of the expected input size while preserving the original aspect ratio, leading to non-square dimensions. Despite the preservation of natural image proportions, this approach results in the lowest performance, with validation accuracy peaking at 83% and significant fluctuations in loss across epochs (Fig. 31), suggesting instability and inefficiency in model learning with a real-world accuracy of only 3.5% on a scanned sample.

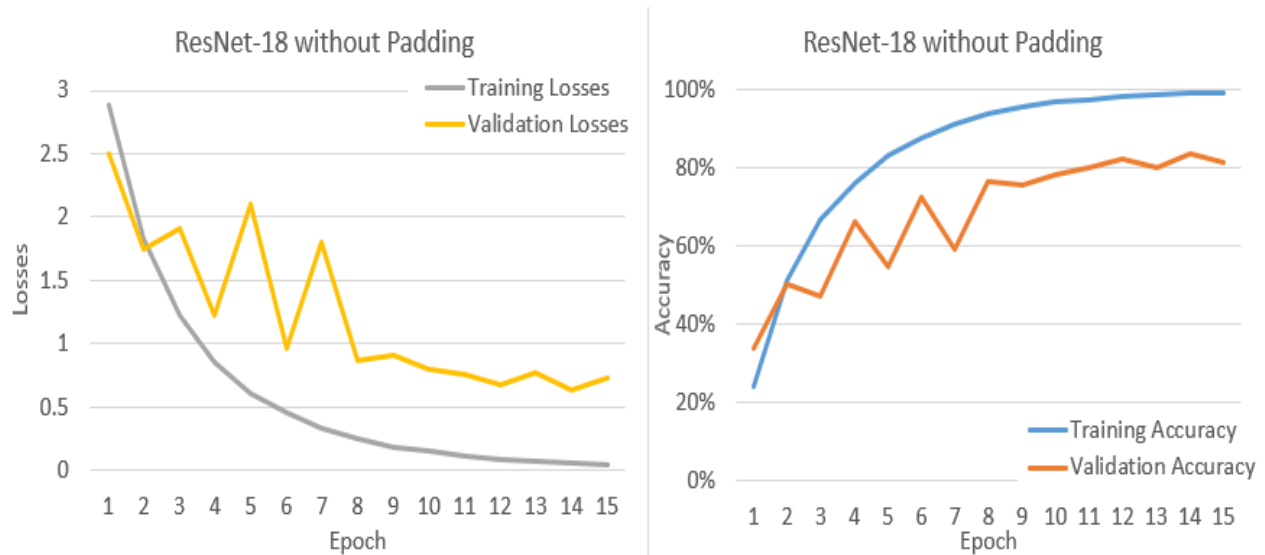


Fig. 31 ResNet-18 Without Padding.

4.1.5.1.2 Aspect Ratio Preserved with White Padding (224x224)

By resizing images to 224x224 pixels and filling the excess space with white padding to maintain the aspect ratio, there is a notable improvement in model performance (Fig. 32). This method stabilizes the validation accuracy, achieving a consistent upper range of 88% toward later epochs. The white padding minimizes distraction, allowing CNN to better focus on the informative parts of the image, thereby enhancing the learning process.

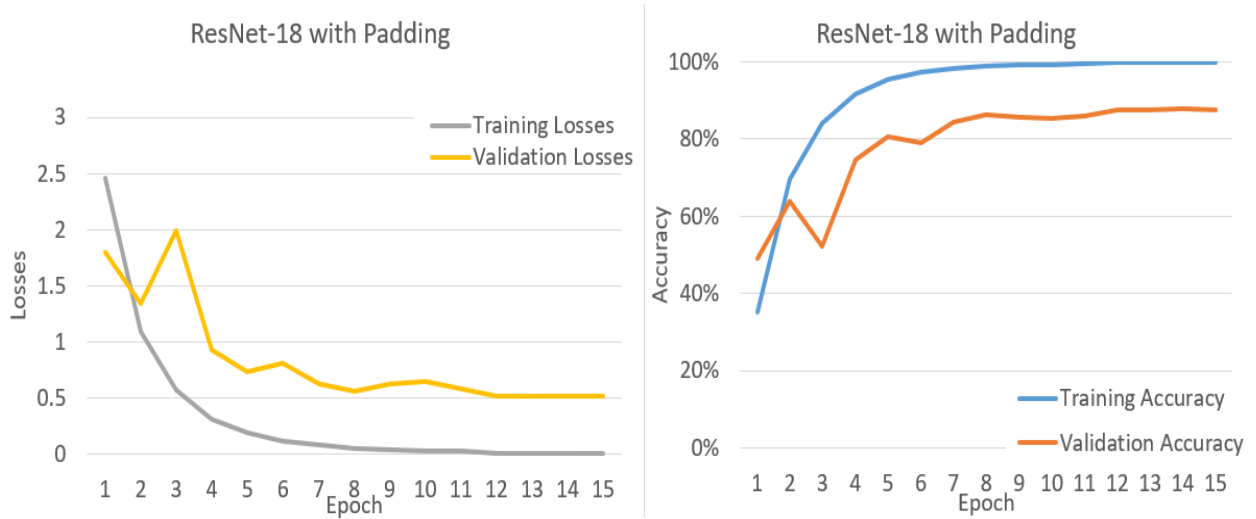


Fig. 32 ResNet-18 With Padding.

4.1.5.2 Additional Observations from ResNet50 Trained on 10000 Images

4.1.5.2.1 ResNet50 without Padding, Direct Resizing to Square (299x299)

When images are resized directly to 299x299 pixels to meet the model's input specification without applying padding, the model achieves an 89% accuracy. This approach aligns with the architectural design of the network, supporting efficient training and high accuracy without the need for additional preprocessing strategies (Fig. 33).

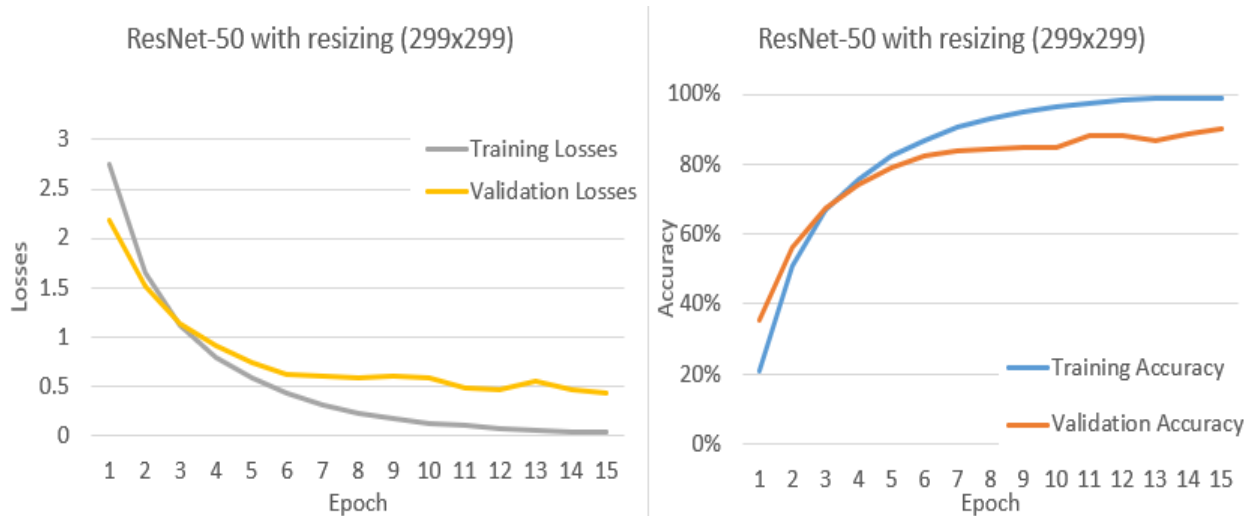


Fig. 33 ResNet-50 Without Padding.

4.1.5.2.2 ResNet50 with White Padding (299x299)

Incorporating white padding at this dimension further illustrates the effectiveness of this preprocessing method, which is particularly evident in the stability of model performance across different training scenarios and datasets. The white background ensures that all images conform uniformly to the network's expectations, facilitating better generalization and robustness in model predictions across diverse data samples.

The comparison reveals that while maintaining the original aspect ratio without padding leads to poorer model performance due to the introduction of potentially misleading spatial configurations, the use of white padding under the same conditions enhances performance. This outcome underscores the importance of considering both the model architecture and the characteristics of the input data when designing preprocessing pipelines. The consistent application of suitable padding not only aids in achieving higher accuracy but also ensures the stability of the model across different epochs and datasets. Additionally, resizing the image to a square shape without padding (299x299) proved effective, demonstrating that this approach can adapt well to the model's requirements, further enhancing the processing efficacy for CNNs.

The training progression showed initial high losses with gradual improvements in accuracy and validation scores as the epochs advanced. This trajectory indicates that while the model could adapt to the improperly scaled inputs, its efficiency was hampered, likely due to the distorted input data.

With white padding (224x224), the model displayed a more stable and consistent improvement across epochs, with lower initial losses and higher accuracy, reflecting the benefits of this preprocessing strategy.

For ResNet-50 with square resizing (299x299), the model training showed robust performance across metrics, with validation accuracies reaching up to 89.92%, highlighting that for certain architectures, directly resizing to the expected input dimensions without padding can be effective, particularly when the resizing does not severely distort the image content.

These experiments elucidate the critical impact of image preprocessing techniques on the performance of convolutional neural networks. They demonstrate that while padding can significantly influence model training and performance, the choice of padding strategy—whether color-consistent or random—can also play a crucial role. Moreover, direct resizing to square dimensions can sometimes be just as effective, depending on the model and dataset characteristics, suggesting a need for tailored preprocessing approaches based on specific model requirements and data attributes.

4.1.6 Extent of Augmentation

The experimental findings clearly illustrate the profound impact of varying levels of augmentation on model training and real-world performance. The augmentation-to-original photo ratio was systematically increased from 0 to 6, providing a detailed view of how model robustness and generalization to new data can be influenced by image diversity within the training set.

4.1.6.1 No Augmentation (0:1 ratio)

This configuration yielded a maximum training accuracy of 89.28% and a validation accuracy that reached up to 78.95% (Fig. 34). In real-world applications, the model only achieved a 12% accuracy, highlighting a significant overfitting problem due to the lack of variety in the training examples.

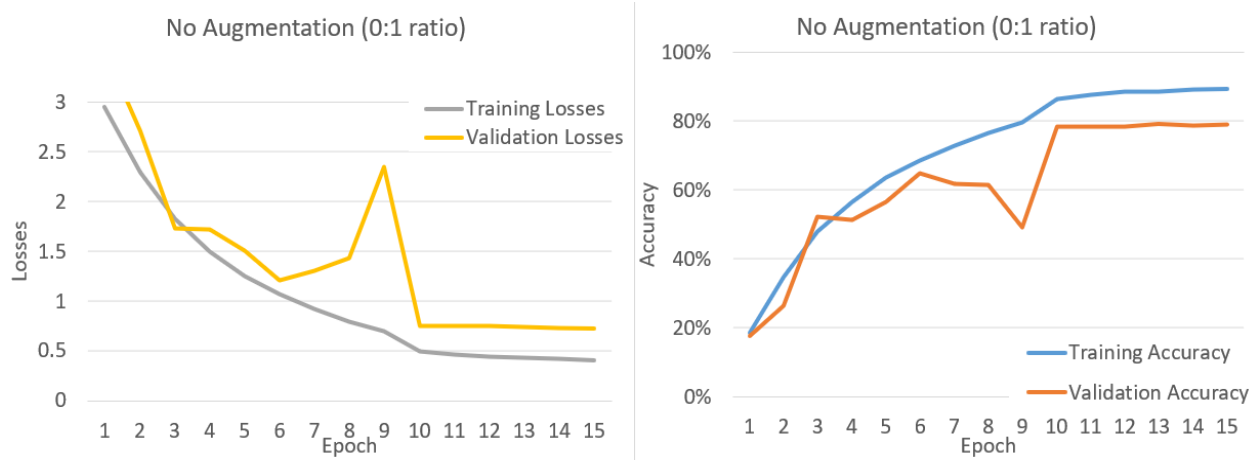


Fig. 34 No Augmentation (0:1 ratio).

4.1.6.2 Mild Augmentation (1:1 ratio)

Introducing an equal number of original and augmented images slightly improved the real-world accuracy to 24%. This setup demonstrates the initial benefits of augmentation, reducing overfitting by providing varied perspectives and background modifications to the model during training.

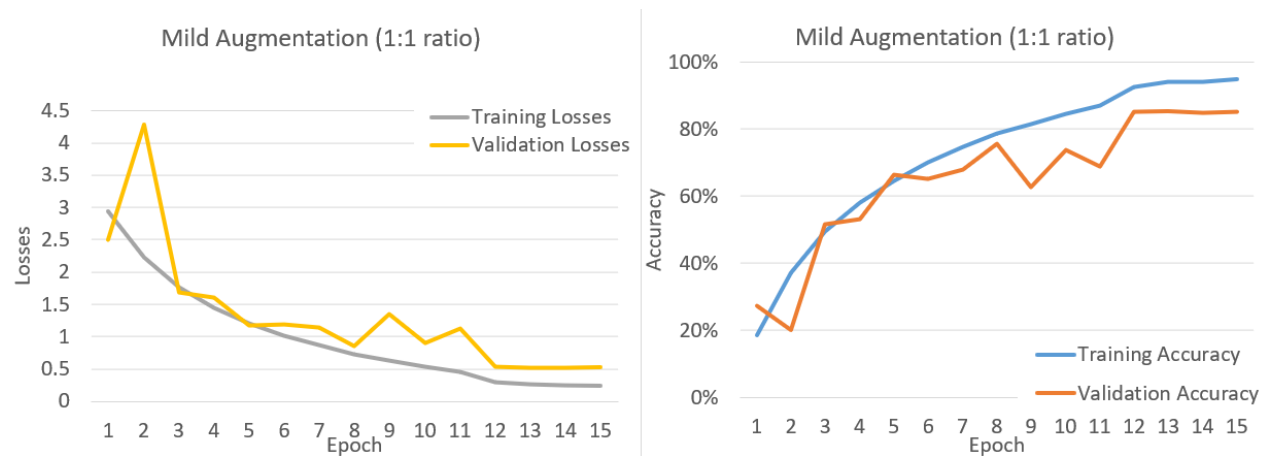


Fig. 35 Mild Augmentation (1:1 ratio).

4.1.6.3 Moderate Augmentation (2:1 and 3:1 ratios)

With a doubling and tripling of augmented images compared to the original ones, real-world accuracy improved to 33% and 40%, respectively. These configurations indicate a better

generalization capability, likely due to the model's increased exposure to diverse training data, which better simulate real-world scenarios.

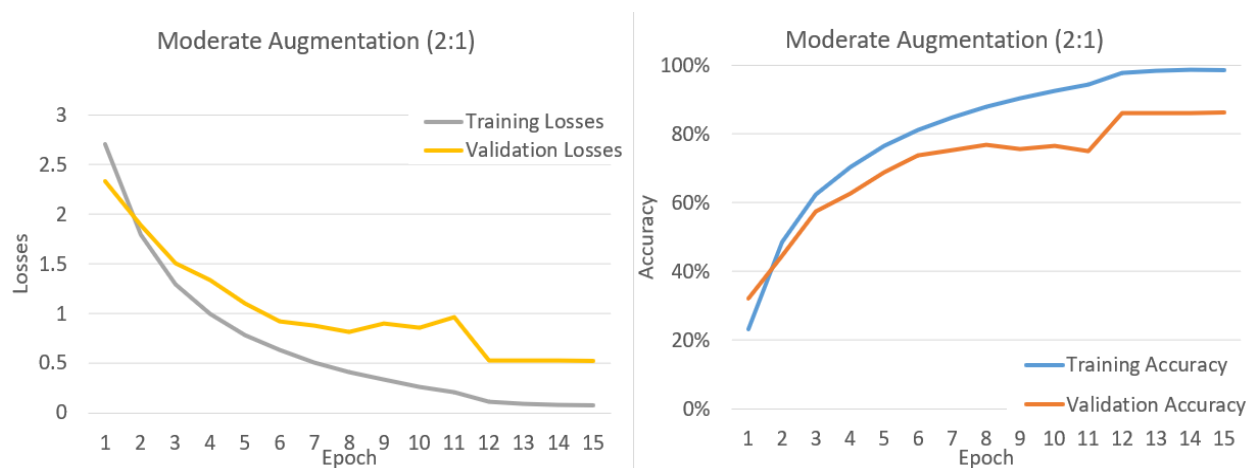


Fig. 36 Moderate Augmentation (2:1 ratio).

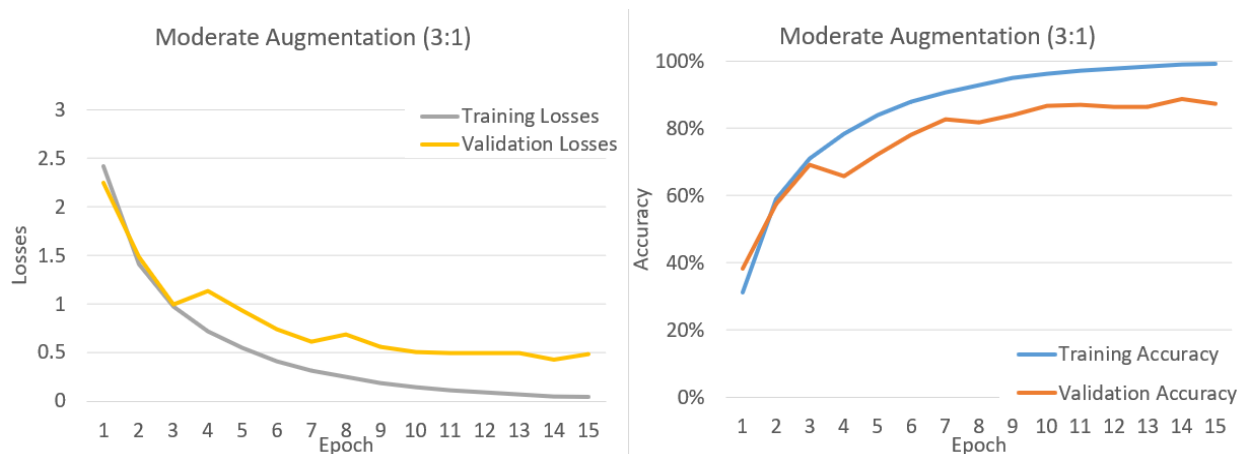


Fig. 37 Moderate Augmentation (3:1 ratio).

4.1.6.4 High Augmentation (4:1, 5:1, and 6:1 ratios)

Continuing to increase the ratio of augmented images shows a more nuanced result. At a 4:1 ratio, real-world accuracy peaks at 36%, but further increasing the augmentation does not yield proportionate gains, with real-world accuracies at 42% and 41% for 5:1 and 6:1 ratios, respectively. This plateau suggests that beyond a certain point, additional augmentation may not provide significant benefits and could potentially introduce noise or irrelevant variations that the model must learn to ignore.

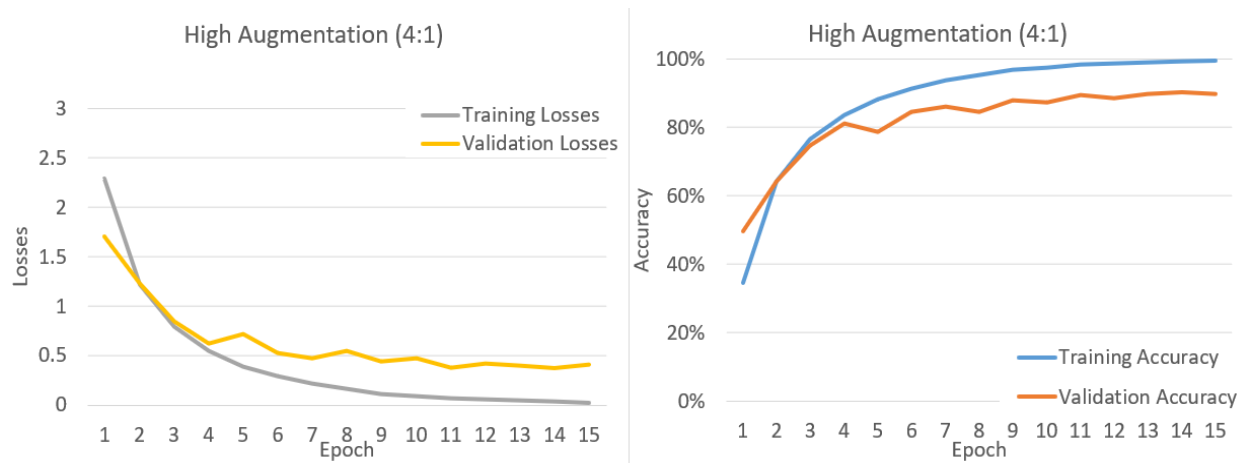


Fig. 38 High Augmentation (4:1 ratio).

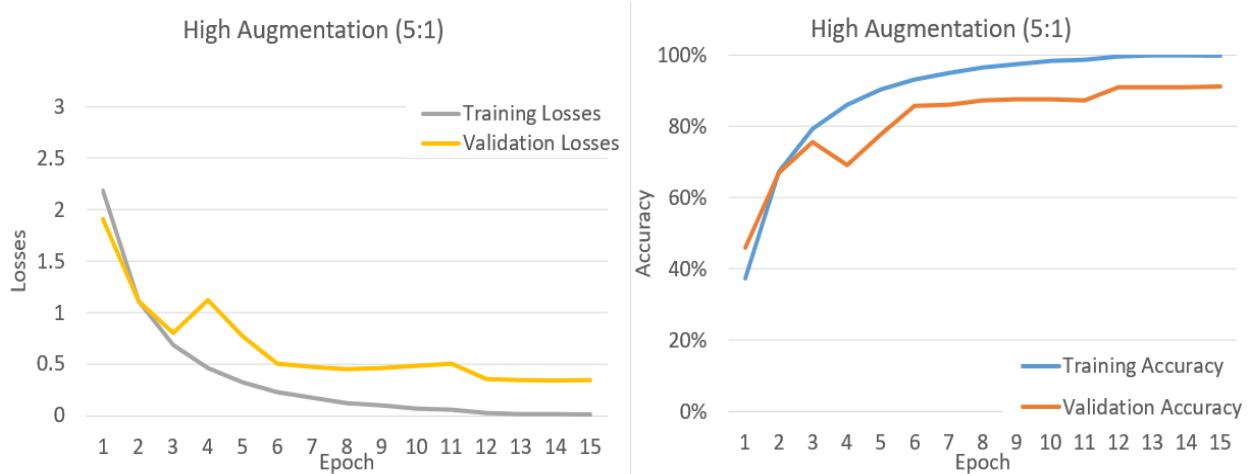


Fig. 39 High Augmentation (5:1 ratio).

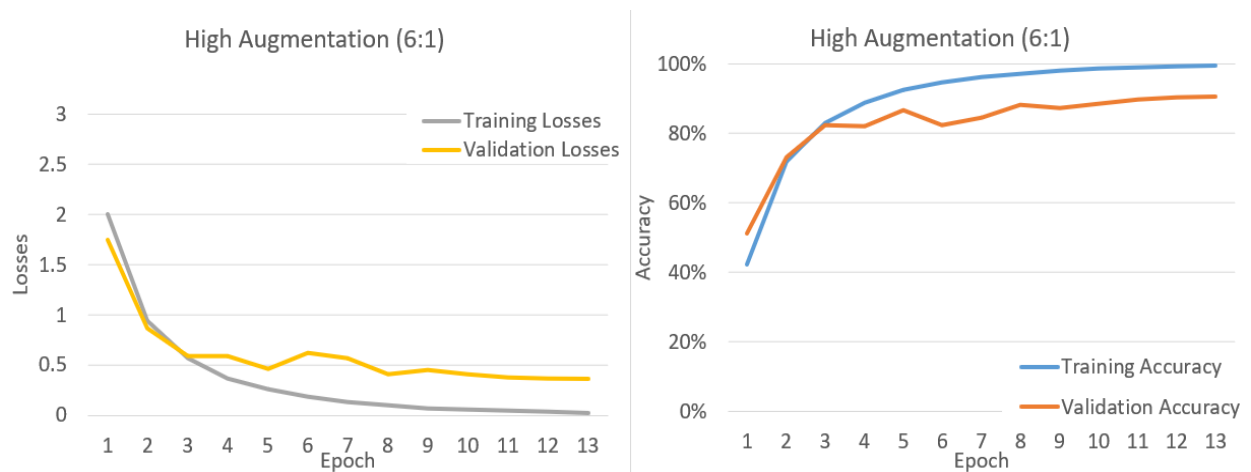


Fig. 40 High Augmentation (6:1 ratio).

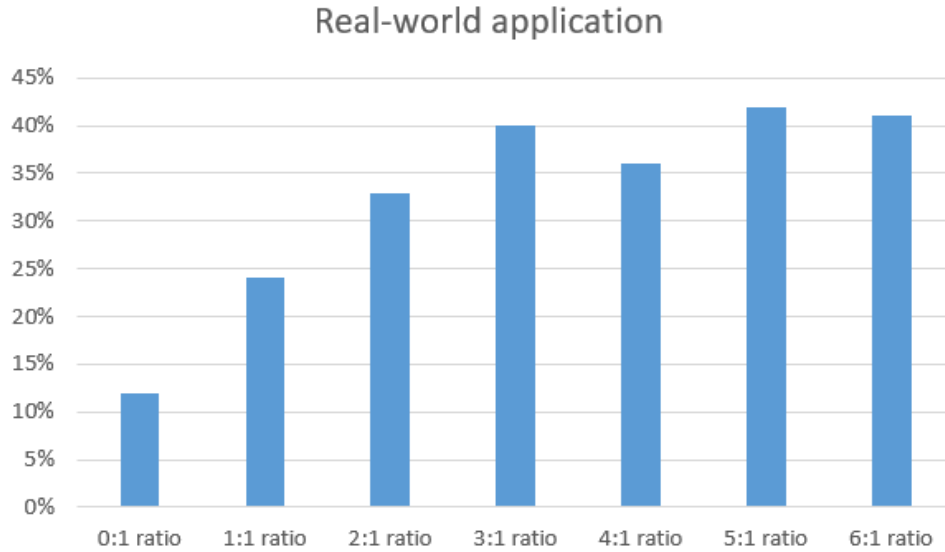


Fig. 41 Real-World application.

These results underscore the critical balance required in selecting the extent of augmentation. While augmentation is crucial for enhancing model robustness and accuracy, too much can lead to diminishing returns or even detrimental effects on the model's ability to generalize from training to real-world application. Effective augmentation strategies must, therefore be tailored to the specific characteristics of the dataset and the capacity of the model to handle complex variations within input data.

4.1.7 Batch Size and Epoch Number

The examination of how batch size influences model training for image classification has uncovered subtle impacts on performance metrics in various setups. Here is a comprehensive summary of the observed trends and their implications.

4.1.7.1 Batch Size 16

In the case of using a batch size of 16, the model starts with a lower proportion of correct classifications, suggesting an initial struggle, but shows a gradual improvement over the epochs, achieving its highest number of true classifications by the tenth epoch (Fig. 42). This trend suggests that smaller batch sizes may enable more frequent updates to the model's weights, allowing for finer adjustments and better navigation through the loss landscape, which potentially enhances steady learning improvements.

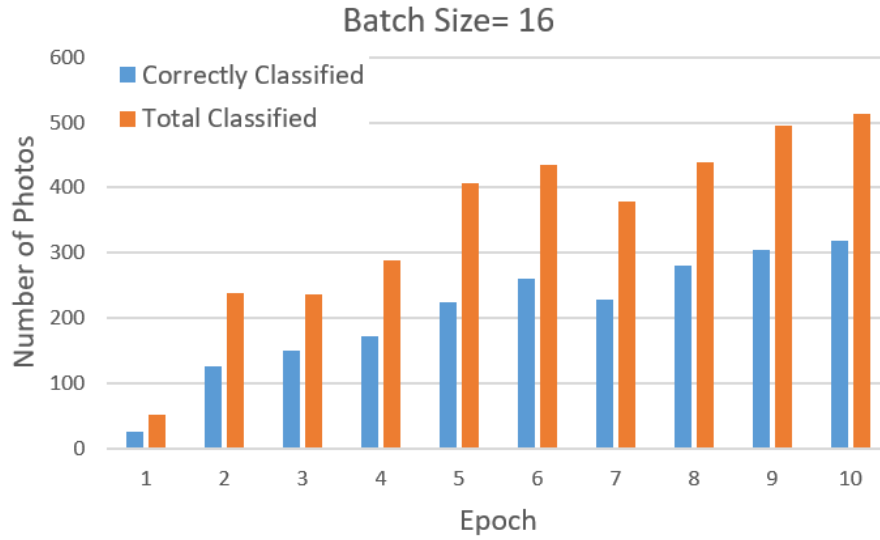


Fig. 42 Models performance (Batch Size = 16).

4.1.7.2 Batch Size 32

For the batch size of 32, the model kicks off with a notably high accuracy, but the performance shows significant fluctuations throughout the training epochs (Fig. 43). Despite a strong start, this variability might indicate that while the batch size allows for relatively stable updates, it may not be as effective at capturing fine-grained variations in the data, leading to inconsistent performance across epochs.

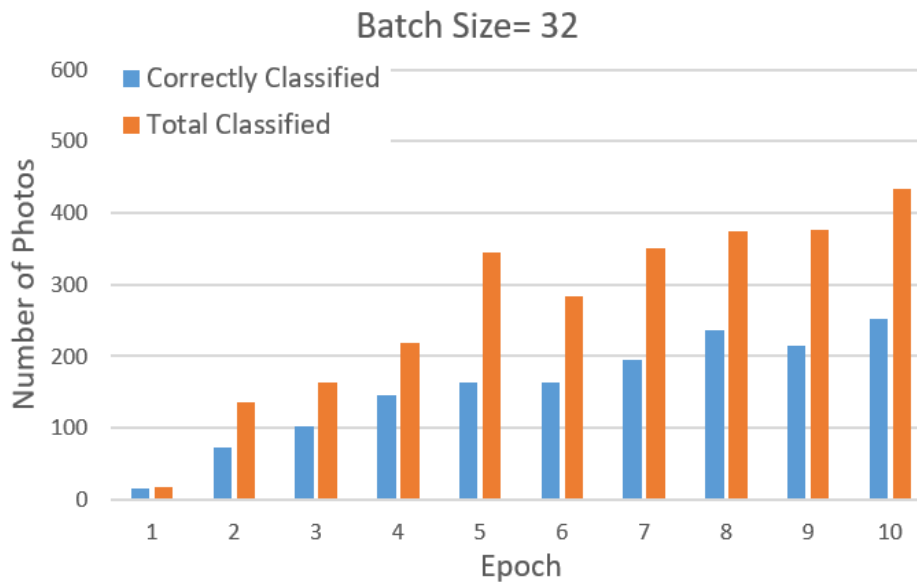


Fig. 43 Models performance (Batch Size =32).

4.1.7.3 Batch Size 64

Using a batch size of 64 exhibits a robust initial performance with a higher number of correct classifications early on but also displays fluctuations and a decline in performance as training progresses (Fig. 44). This pattern suggests that larger batch sizes may complicate the optimization process, possibly due to slower convergence rates or difficulties in escaping local minima, which can result in overfitting or insufficient generalization capabilities, particularly highlighted by the later decrease in performance.

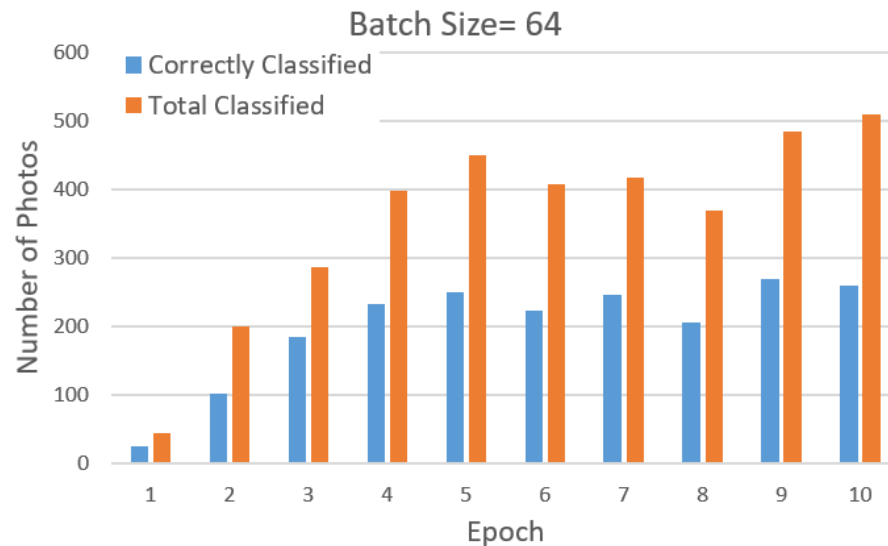


Fig. 44 Models performance (Batch Size = 64).

Each of these observations underscores the nuanced impact of batch size on the dynamics of training deep learning models for image classification, with smaller batches generally favoring more detailed learning and larger batches potentially offering quicker initial learning but at the risk of instability and inefficiencies in long-term training outcomes.

4.1.8 Comparing Model Architectures

In my project, which involves tackling the complexities of classifying a varied and imbalanced dataset, ResNet architectures—specifically ResNet-18 and ResNet-50—have been identified as highly suitable. These models excel in efficiently learning from complex and hierarchical data structures that often characterize imbalanced datasets. The incorporation of skip connections within these models mitigates the problem of vanishing gradients, which is prevalent in deep neural networks, particularly when training on datasets with uneven class representation.

The design of ResNet models facilitates effective learning even when data points per class are limited compared to other deep networks. This capability is critical in scenarios like mine, where data distribution across classes is not uniform. ResNet's skip connections simplify the flow of gradients during training, enabling the construction of deeper network architectures without the typical training difficulties associated with such depth. This characteristic allows for robust

performance, even with a relatively small number of images in some classes, by extracting meaningful features that aid in the classification across diverse classes.

ResNet also offers considerable flexibility and scalability, which are advantageous for my project as it evolves. Starting with a simpler model such as ResNet-18 for preliminary experiments and advancing to ResNet-50 as the dataset expands or becomes more complex offers a practical approach to managing computational resources while achieving high accuracy.

The profound feature extraction capabilities of ResNet prove beneficial for image classification tasks. Models like ResNet-50, capable of discerning intricate patterns and subtle differences between images, are essential for ensuring precise classification across multiple, diverse classes. This depth allows the model to perform effectively, even in situations where some classes are significantly underrepresented, thus supporting better generalization and reducing bias towards more frequently represented classes.

ResNet's established track record in handling image classification tasks further underscores its suitability for this project. It has been extensively tested and validated in both academic and industrial environments, demonstrating high performance and reliability.

The widespread adoption of ResNet in a variety of image-related applications highlights its robustness and the confidence that researchers and developers have in its capabilities.

Overall, the combination of deep learning efficiency, robust feature extraction, model scalability, and ease of the implementation process makes ResNet a compelling choice for this project. Its architecture is adept at handling the challenges posed by an imbalanced dataset, ensuring that all classes are accurately represented and contributing to the overall effectiveness of the classification system.

Table.1 Overall evaluation

		Model Accuracy					
		ResNet-18			ResNet-50		
		Training	Validation	Real-world	Training	Validation	Real-world
Data Size	4000	99.00%	87.00%	13.00%	-	-	-
	7000	99.00%	89.00%	27.00%	-	-	-
	10000	99.00%	91.00%	39.00%	-	-	-
Image Size	224	99.00%	92.00%	43.00%	99.00%	87.00%	39.00%
	299	99.00%	93.00%	48.00%	99.00%	88.00%	47.00%
Padding	with	99.00%	87.00%	-	-	-	-
	without	99.00%	81.00%	-	99.00%	90.00%	-

Augmentation	0:1	90.00%	80.00%	12.00%	-	-	-
	1:1	93.00%	85.00%	24.00%	-	-	-
	2:1	98.00%	84.00%	33.00%	-	-	-
	3:1	99.00%	85.00%	40.00%	-	-	-
	4:1	99.00%	90.00%	36.00%	-	-	-
	5:1	99.00%	91.00%	42.00%	-	-	-
	6:1	99.00%	92.00%	41.00%	-	-	-
Batch Size	Batch Size 16	Epoch 1	25/50	-	-	-	
		Epoch 2	125/230	-	-	-	
		Epoch 3	150/230	-	-	-	
		Epoch 4	175/290	-	-	-	
		Epoch 5	220/405	-	-	-	
		Epoch 6	255/420	-	-	-	
		Epoch 7	225/380	-	-	-	
		Epoch 8	280/420	-	-	-	
		Epoch 9	302/495	-	-	-	
		Epoch 10	310/510	-	-	-	
	Batch Size 32	Epoch 1	14/15	-	-	-	
		Epoch 2	75/135	-	-	-	
		Epoch 3	100/155	-	-	-	
		Epoch 4	140/215	-	-	-	
		Epoch 5	155/330	-	-	-	
		Epoch 6	160/290	-	-	-	
		Epoch 7	190/345	-	-	-	
		Epoch 8	230/370	-	-	-	
		Epoch 9	210/375	-	-	-	
		Epoch 10	250/430	-	-	-	
	Batch Size 64	Epoch 1	20/40	-	-	-	
		Epoch 2	100/200	-	-	-	
		Epoch 3	190/290	-	-	-	
		Epoch 4	225/400	-	-	-	
		Epoch 5	250/440	-	-	-	
		Epoch 6	220/405	-	-	-	
		Epoch 7	240/410	-	-	-	
		Epoch 8	205/370	-	-	-	
		Epoch 9	270/480	-	-	-	
		Epoch 10	250/510	-	-	-	

5 Conclusion

The research undertaken in this thesis represents a significant stride in the field of automotive engineering, particularly in the enhancement of electric vehicle (EV) technologies through the development of a Part Recognition Tool for eAxles. This tool, powered by sophisticated image recognition technology and artificial intelligence, automates the identification and categorization of eAxle components, thereby advancing the efficiency and accuracy of physical analyses in EV development.

Central to the success of this initiative was the in-depth exploration and adaptation of ResNet architectures, which are renowned for their robust performance in handling complex and imbalanced datasets. The tailored AI model developed through this research demonstrated a promising initial real-world application accuracy of 55%. Strategies for enhancing this accuracy, including the augmentation of training datasets and the optimization of augmentation techniques, were explored, with projections indicating gradual improvements in performance.

Furthermore, the introduction of the "Linearoch" system played a pivotal role in the project by automating the preparation and organization of extensive datasets. Linearoch significantly streamlined the data management process, reducing the potential for human error and enhancing the overall efficiency of the machine learning workflow. This system not only supported the effective training of the Part Recognition Tool but also demonstrated the potential for broader applications in managing data for AI-driven projects.

The integration of the Part Recognition Tool into the automotive industry is anticipated to revolutionize the development process for eAxles, reducing time-to-market and production costs while improving the quality of the output. The practical implications of this tool extend beyond academic interest, offering substantial benefits to industry practitioners by enhancing the precision of component analyses and supporting faster design iterations.

This thesis not only contributes to the academic and practical knowledge in the field of AI applications in mechanical engineering but also sets a precedent for the future integration of AI tools in automotive engineering and other industrial applications. The research highlighted the importance of maintaining rigorous standards of accuracy and reliability in the deployment of AI tools, ensuring that these technologies meet the complex demands of real-world applications.

In conclusion, the development of the Part Recognition Tool for eAxles represents a significant advancement in the application of artificial intelligence in the automotive sector. The successful implementation of this tool could serve as a model for similar innovations across different sectors, potentially leading to more widespread adoption of AI in complex industrial analyses. Future research will focus on refining this tool and exploring other AI models that could further enhance its capabilities, continuing to push the boundaries of what is possible in the automation of vehicle system analyses.

6 Summary

In the landscape of automotive engineering, the emergence of electric vehicles (EVs) represents a significant technological and environmental milestone. Central to the advancement of EV technology is the development and optimization of electric axles (eAxles), which integrate key drivetrain components into compact, efficient units. This thesis presented the development of an advanced Part Recognition Tool designed to enhance the physical analysis of eAxles through the utilization of sophisticated image recognition technologies powered by artificial intelligence (AI). The tool automates the identification and categorization of eAxle components, facilitating more accurate and efficient developmental analyses.

The research included a detailed review of existing AI models and image recognition techniques, with a focus on ResNet architectures known for their deep learning capabilities and robust performance on complex datasets. This review provided foundational knowledge for the development of a specialized AI tool tailored for the intricacies of eAxle parts, highlighting the importance of selecting appropriate models that can handle the specific challenges posed by the unique geometries and configurations of eAxle components.

A critical component of this thesis was the development and implementation of the "Linearoch" system, a sophisticated script designed to systematize and streamline the data preparation process for machine learning. Linearoch automates the organization, preprocessing, and management of complex data structures, significantly enhancing the efficiency of preparing large and heterogeneous datasets. By automating tasks such as reorganizing folder structures, renaming files while preserving hierarchical relationships, and handling various types of image files, Linearoch minimizes human error and reduces the time and effort typically associated with manual data preparation. Its core functionality transforms unstructured or semi-structured data repositories into well-organized formats that are readily accessible and optimized for further processing and analysis. This structured systematization is essential for improving the performance of machine learning models, ensuring that the data provided to these models is of high quality, well-annotated, and consistently formatted. The introduction of Linearoch in this research not only supported the effective training of the Part Recognition Tool but also set a precedent for future applications, offering a robust framework for data management in complex AI-driven projects.

The creation of a comprehensive dataset specifically designed for the training and testing of the AI model was detailed. This dataset included a wide array of eAxle images, meticulously annotated to ensure precise model learning. The complexity and variety embedded within this dataset aimed to mirror real-world conditions, thus preparing the model to accurately identify and categorize eAxle parts across different makes and models of eAxles.

The core of the thesis involved the development of the AI model using the ResNet architectures. Modifications and optimizations were applied to these architectures to better suit the specific needs of eAxle analysis. The performance of the AI model was rigorously evaluated through a series of

tests designed to assess its accuracy, reliability, and scalability, validating the effectiveness of the tool in practical applications.

In real-world applications, the Part Recognition Tool achieved an initial accuracy of 55%. Several strategies to enhance this accuracy were detailed, such as increasing the size of the training dataset, optimizing the number of augmentations, and continuously refining the AI model based on ongoing testing and feedback. These enhancements are expected to gradually improve the tool's accuracy, making it even more effective in practical settings.

Moreover, the thesis explored the practical implications of the Part Recognition Tool within the automotive industry. The integration of this tool into the eAxle development process is projected to significantly enhance the efficiency and accuracy of part analysis, which could lead to faster design iterations, reduced costs, and improved overall vehicle performance. The potential industry impact of such a tool underscores its value, not only as an academic endeavor but also as a significant technological advancement.

The thesis concluded by summarizing the research outcomes and reiterating the potential of the Part Recognition Tool to transform the eAxle development process. It also outlined future research directions, including the exploration of other AI models that could further enhance the capabilities of the tool. The conclusion reaffirmed the significance of the tool in advancing not only the field of automotive engineering but also the broader application of AI in complex industrial analyses.

This comprehensive summary encapsulates the development, implementation, and evaluation of a Part Recognition Tool for eAxles, highlighting its importance and potential impact on the automotive industry. The integration of advanced AI technologies within the automotive sector promises to enhance the precision and efficiency of vehicle development, paving the way for more innovative and sustainable vehicle technologies.

DECLARATION

on authenticity and public assess of master's thesis

Student's name: Tamim Deeb

Student's Neptun ID: FZPYFX

Title of the document: Development of a Part Recognition Tool for Physical Analysis of eAxles

Year of publication: 2024

Department: Mechanical Engineering

I declare that the submitted final master's thesis is my own, original individual creation. Any parts taken from another author's work are clearly marked, and listed in the table of contents.

If the statements above are not true, I acknowledge that the Final examination board excludes me from participation in the final exam, and I am only allowed to take the final exam if I submit another master's thesis

Viewing and printing my submitted work in a PDF format is permitted. However, the modification of my submitted work shall not be permitted.

I acknowledge that the rules on Intellectual Property Management of Hungarian University of Agriculture and Life Sciences shall apply to my work as an intellectual property.

I acknowledge that the electronic version of my work has been uploaded to the repository system of the Hungarian University of Agriculture and Life Sciences.

Place and date: Gödöllő, 2024 April 26



Student's signature

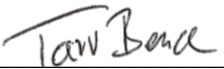
STATEMENT ON CONSULTATION PRACTICES

As a supervisor of Tamim Deeb FZPYFX, I here declare that the master's thesis has been reviewed by me, the student was informed about the requirements of literary sources management and its legal and ethical rules.

I **recommend** the master's thesis to be defended in a final exam.

The document contains state secrets or professional secrets: yes **no**

Place and date: Gödöllő, 2024 April 26



Internal supervisor

7 References

- Banfield, R. E., Hall, L. O., Bowyer, K. W., & Kegelmeyer, W. P. (2007). A comparison of decision tree ensemble creation techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1), 173-180.
- Belkin, M., & Niyogi, P. (2001). Laplacian Eigenmaps and spectral techniques for embedding and clustering. In *Proc. Adv. NIPS* (Vol. 14, pp. 585–591).
- Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 1798-1828.
- Bian, W., & Tao, D. (2010). Biased discriminant Euclidean embedding for content-based image retrieval. *IEEE Transactions on Image Processing*, 19(2), 545–554.
- Bosch Mobility Solutions. (n.d.). eAxle. Retrieved April 25, 2024, from <https://www.bosch-mobility.com/en/solutions/electric-motors/eaxle/>
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5-32.
- Chen, J., Shan, S., Zhao, G., Chen, X., & Gao, W. (2010). WLD: A robust descriptor based on Weber's law. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9), 1705–1720.
- Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005* (Vol. 1, pp. 886–893).
- Dalla Mura, M., Benediktsson, J. A., Waske, B., & Bruzzone, L. (2010). Morphological attribute profiles for the analysis of very high resolution images. *IEEE Transactions on Geoscience and Remote Sensing*, 48(10), 3747–3762.
- Dalla Mura, M., Villa, A., Benediktsson, J. A., Chanussot, J., & Bruzzone, L. (2011). Classification of hyperspectral images by using extended morphological attribute profiles and independent component analysis. *IEEE Geoscience and Remote Sensing Letters*, 8(3), 542–546.
- Das, V., & Vijaykumar, R. (2010). Image Object Classification Using Scale Invariant Feature Transform Descriptor with Support Vector Machine Classifier with Histogram Intersection Kernel. In *CCIS* (Vol. 101).
- De Houwer, J., Barnes-Holmes, D., & Moors, A. (2013). What is learning? On the nature and merits of a functional definition of learning. *Psychonomic Bulletin & Review*, 20(4), 631–642.
- Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10), 78–87.
- Duarte, D., & Ståhl, N. (2019). Machine Learning: A Concise Overview. In *Studies in Big Data* (Vol. 46, pp. 27–58). Springer Science and Business Media Deutschland GmbH. https://doi.org/10.1007/978-3-319-97556-6_3

- Fukushima, K., & Miyake, S. (1982). Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, 15(6), 455-469.
- Guo, G., Jain, A. K., Ma, W., & Zhang, H. (2002). Learning similarity measure for natural image retrieval with relevance feedback. *IEEE Transactions on Neural Networks*, 13(4), 811–820.
- Haghighat, M., Zonouz, S., & Abdel-Mottaleb, M. (2015). CloudID: Trustworthy cloud-based and cross-enterprise biometric identification. *Expert Systems with Applications*, 42(21), 7905-7916.
- Haralick, R. M., Shanmugam, K., & Dinstein, I. (1973). Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6), 610-621.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)* (pp. 630–645).
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. Retrieved from <http://image-net.org/challenges/LSVRC/2015/>
- Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 504-507.
- Ho, T. K. (1998). The Random Subspace Method for Constructing Decision Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8), 832-844.
- Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction of functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1), 106-154.
- Image Processing The Fundamentals. (1999).
- Kher, H. R., & Thakar, V. K. (2014). Scale-invariant feature transform-based image matching and registration. *Proceedings - 2014 5th International Conference on Signal and Image Processing, ICSIP 2014*, 50–55. <https://doi.org/10.1109/ICSIP.2014.12>
- Kim, M., Madden, M., & Warner, T. A. (2009). Forest type mapping using object-specific texture measures from multispectral IKONOS imagery: Segmentation quality and image classification issues. *Photogrammetric Engineering & Remote Sensing*, 75(7), 819–829.
- Kumar, S., Khan, Z., & Jain, A. (2012). A Review of Content Based Image Classification using Machine Learning Approach. In *International Journal of Advanced Computer Research*.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.
- LeCun, Y., et al. (1990). Handwork digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems*. (pp. 396-404), Colorado, USA.
- LeCun, Y., Bengio, Y., & Hinton, G. (1988). A theoretical framework for back-propagation. In *The Connectionist Models Summer School* (Vol. 1, pp. 21–28).
- LeCun, Y., & Cortes, C. (2010). MNIST handwritten digit database. Retrieved from <http://yann.lecun.com/exdb/mnist>

- Li, J., Allinson, N., Tao, D., & Li, X. (2006). Multitraining support vector machine for image retrieval. *IEEE Transactions on Image Processing*, 15(11), 3597–3601.
- Liu, J., Du, W., Zhou, C., & Qin, Z. (2021). Rock image intelligent classification and recognition based on Resnet-50 model. *Journal of Physics: Conference Series*, 2076(1). <https://doi.org/10.1088/1742-6596/2076/1/012011>
- Lembaga Ilmu Pengetahuan Indonesia, Institute of Electrical and Electronics Engineers. Indonesia Section, & Institute of Electrical and Electronics Engineers. (2017). 2017 ICRAMET proceeding: 2017 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET). Jakarta, Indonesia, October 23-24, 2017.
- Matlab. (2023). Image Processing Toolbox User's Guide (R2023b). MathWorks. Retrieved from <https://www.mathworks.com>
- Mitchell, T. M. (1997). *Machine learning* (1st ed.). New York, NY, USA: McGraw-Hill Inc.
- Ouma, Y. O., Tetuko, J., & Tateishi, R. (2008). Analysis of co-occurrence and discrete wavelet transform textures for differentiation of forest and nonforest vegetation in very high-resolution optical-sensor imagery. *International Journal of Remote Sensing*, 29(12), 3417–3456.
- Pesaresi, M., & Benediktson, J. A. (2001). A new approach for the morphological segmentation of high-resolution satellite imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 39(2), 309–320.
- Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(22), 2323–2326.
- Ruck, D. W., Rogers, S. K., & Kabrisky, M. (1990). Feature selection using a multilayer perceptron. *Journal of Neural Network Computing*, 2(2), 40-48.
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533-538.
- Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3), 210–229.
- Singh, S., & Dhir, R. (2012). Recognition of Handwritten Gurmukhi Numeral using Gabor Filters. *International Journal of Computer Applications*, 47(1), 7-11.
- Tao, D., Tang, X., & Li, X. (2008). Which components are important for interactive image searching? *IEEE Transactions on Circuits Systems and Video Technology*, 18(1), 3–11.
- Tao, D., Tang, X., Li, X., & Wu, X. (2007). Asymmetric bagging and random subspace for support vector machines based relevance feedback in image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7), 1088–1099.
- Tahir, S., Rizvi, H., Cabodi, G., & Gusmao, P. (2016). Gabor Filter-based Image Representation for Object Classification.

- Tou, J. Y., Tay, Y. H., & Lau, P. Y. (2009). A Comparative Study for Texture Classification Techniques on Wood Species Recognition Problem. In Proc. Fifth International Conference on Natural Computation (pp. 8-12), Tianjin.
- Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. New York: Springer-Verlag.
- Venkata Sai Abhishek, A., Rao Gurralla, V., Sahoo, L., Scholar, R., & Professor, S. (2022). Resnet18 Model With Sequential Layer For Computing Accuracy On Image Classification Dataset. Retrieved from www.ijert.org
- Vo, T., Tran, D., Ma, W., & Nguyen, K. (2013). LNCS 8228 - Improved HOG Descriptors in Image Classification with CP Decomposition.
- Waibel, A., et al. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(3), 328-339.
- Wang, X., Ding, X., & Liu, C. (2002). Optimized Gabor filter-based feature extraction for character recognition. In Proc. 16th International Conference on Pattern Recognition (Vol. 4, pp. 223-226).
- Yi, J., & Su, F. (2014). Histogram of Log-Gabor Magnitude Patterns for face recognition. In Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 519-523), Florence.
- Xu, Y., & Nie, S. (2012). An improved random forest classifier for image classification. Details on publication not provided.
- Zhang, D., & Lu, G. (2004). Review of shape representation and description techniques. *Pattern Recognition*, 37(1), 1-19.
- Zhao, L., Wang, J., Li, X., Tu, Z., & Zeng, W. (2016). On the connection of deep fusion to ensembling. Retrieved from <https://www.microsoft.com/en-us/research/publication/connection-deep-fusion-ensembling/>.
- Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European Conference on Computer Vision* (pp. 818–833). Springer.
- Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. In *British Machine Vision Conference (BMVC)* (pp. 1–15).