

Closed-loop Metrics-based Dataset Optimization Toward Deep Learning Models Effectiveness Enhancement

Somayeh Shahrabadi

UMinho | 2026



Universidade do Minho

Escola de Engenharia

Somayeh Shahrabadi

Closed-loop Metrics-based Dataset Optimization Toward Deep Learning Models Effectiveness Enhancement

March 2026



University of Minho

School of Engineering

Somayeh Shahrabadi

Closed-loop Metrics-based Dataset Optimization Toward Deep Learning Models

PhD Thesis

Doctoral Program in Informatics

Work carried out under the supervision of

Professor Doutor Victor Manuel Rodrigues Alves

Professor Doutor Telmo Miguel Oliveira Adão

March 2026

DECLARATION

Name: Somayeh Shahrabadi

Thesis Title: Closed-loop Metrics-based Dataset Optimization Toward Deep Learning Models

Supervisors: Victor Manuel Rodrigues Alves, Telmo Miguel Oliveira Adão

Conclusion Year: 2025

Doctoral Program Designation: Doctoral Program in Informatics

I hereby grant the University of Minho the non-exclusive and irrevocable right to archive, reproduce, communicate and/or distribute through its institutional repository, accordingly to the conditions mentioned below, the final version approved after the completion of the public defence trials and, when applicable, after confirmation by the advisor(s) of the insertion of requested amendments and corresponding homologation by the jury president, of my academic work referred to above in digital format.

I hereby authorize the University of Minho to archive more than one copy of my academic work referred to above and, without altering its content, to convert the delivered archive to any file format, media or support, for the purpose of preservation and access.

I declare that my academic work referred to above and now delivered is an original work, and if it contains material which I don't hold copyright, that I obtained previous permission from the copyright owner to grant the University of Minho the terms required by this license.

I also declare that the delivery of this document does not infringe, as far as I can know, the rights of any other person or entity.

I agree that my academic work referred to above is placed in the University of Minho institutional repository with the following status.

Atribuição-NãoComercial-SemDerivações

CC BY-NC-ND

https://creativecommons.org/licenses/by-nc-nd/4.0/

Universidade do Minho, 31/03/2026

Signature: Somayeh Shahrabadi Digitally signed by Somayeh Shahrabadi Date: 2026.03.31 13:42:39 +01'00'

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to everyone who supported me throughout this PhD journey. First and foremost, I sincerely thank my supervisors, Prof Victor Manuel Rodrigues Alves and Prof Telmo Miguel Oliveira Adão, for their guidance and valuable insights. Their feedback was essential to the progress and completion of this work.

To my friends, thank you for being there during both the challenging and joyful moments. Your support, motivation, and occasional distractions helped me stay balanced and keep going.

To my family, to whom I owe everything. I thank my father for his support. My special and heartfelt thanks go to my mother, whose strength, love, and sacrifices have carried me through life. You are my hero, Mom, and this achievement is as much yours as it is mine.

A special thanks to my sister, niece, and brother-in-law for their constant love. Most importantly, thank you for taking care of our parents and giving me the peace of mind to focus on my work without extra stress. I'm deeply grateful for all you've done.

This journey would not have been possible without the support of all these amazing people. I am truly grateful.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Universidade do Minho, 31/03/2026

Signature: Somayeh Shahrabadi

Digitally signed by Somayeh Shahrabadi
Date: 2026.03.31 13:43:38 +01'00'

RESUMO

Os avanços tecnológicos permitiram a exploração prática de conceitos de *artificial neural network* propostos na década de 1980, conduzindo a progressos significativos na visão por computador, particularmente através das *Convolutional Neural Networks* (CNNs). No entanto, o desempenho das CNNs depende fortemente da qualidade e diversidade dos dados. Neste contexto, têm sido propostas estratégias centradas nos dados para explorar todo o potencial de inferência dos modelos, abrangendo desde etapas iniciais de divisão e aumento de conjuntos de dados até procedimentos de avaliação posteriores, como a *K-Fold Cross Validation* (KFCV) . Apesar da sua adoção generalizada, algumas limitações podem ser identificadas nestas práticas padronizadas. Por um lado, os procedimentos de divisão e aumento de conjuntos de dados são tipicamente realizados sem considerar as respetivas características intrínsecas, com implicações para a representatividade inerentemente ligada a situações como padrões raros de ocorrências ou combinações específicas de primeiro plano e fundo. Por outro lado, embora técnicas baseadas em KFCV sejam geralmente eficazes, a sua natureza iterativa torna-as computacionalmente dispendiosas e temporalmente exigentes.

Motivado por estes desafios, o presente trabalho investiga a engenharia e otimização de conjuntos de dados como forma de melhorar e, simultaneamente, tornar mais eficiente o treino de CNNs. Para tal, são propostas duas abordagens complementares. A primeira, inspirada em princípios de aprendizagem não supervisionada, realiza uma análise da distribuição de características intrínsecas dos dados antes do treino, permitindo uma divisão equilibrada dos dados numa única etapa de curta duração. A segunda introduz uma estratégia em malha fechada, baseada em reforço e aplicada durante o treino, que reamostra seletivamente classes sub-representadas enquanto monitoriza métricas-chave para mitigar a sobre-amostragem e a degradação do modelo.

Para avaliar a robustez e capacidade de generalização, foram utilizados dois conjuntos de dados específicos de domínio: um relativo à classificação de variedades de folhas de videira e outro focado na deteção de defeitos em pontes. Os resultados mostram que a divisão de dados informada pelas características oferece uma alternativa eficiente aos métodos tradicionais. Além disso, a estratégia de reamostragem orientada por métricas, implementada num esquema de malha fechada, demonstrou ser eficaz na melhoria consistente do desempenho das CNNs, reforçando a representatividade das classes sem promover sobreajuste.

Palavras-chave: Aprendizagem Profunda, Divisão de Dados Informada por Características, Treino de Redes Neuronais Profundas Otimizado em Malha Fechada.

ABSTRACT

Technological advancements have enabled the practical application of artificial neural network concepts proposed in the 1980s, leading to major progress in computer vision, particularly through Convolutional Neural Networks (CNNs). However, CNN architectures alone do not guarantee successful supervised learning, as performance also strongly depends on data quality and diversity. Therefore, data-centred strategies have been proposed to explore the full potential of model inference, ranging from early-stage dataset splitting and augmentation to later evaluation procedures such as k-fold cross-validation. Despite their widespread adoption, some limitations can be identified in these standardized practices. On the one hand, dataset splitting and augmentation procedures are typically performed in a feature-blind manner, often failing to address representativeness issues in which plausible but infrequent situations are underrepresented, such as rare feature patterns or specific foreground-background combinations. On the other hand, although techniques based on KFCV are generally effective, its iterative nature makes them computationally expensive and time-consuming.

Motivated by these challenges, this work investigates dataset engineering and optimization to improve CNN training in a cost- and time-efficient manner. Two complementary approaches are proposed. The first, inspired by unsupervised learning principles, performs a pre-training feature distribution analysis to enable one-shot balanced data splitting. The second introduces a reinforcement-based closed-loop strategy applied during training, which selectively resamples underrepresented classes while monitoring key metrics to prevent oversampling and model degradation.

To assess robustness and generalization, two domain-specific datasets were used: one addressing grapevine leaf variety classification and another focusing on bridge defect detection. The results show that feature-aware pre-training data splitting provides an efficient alternative to traditional methods. Furthermore, the proposed metrics-guided resampling strategy, implemented in a closed-loop framework, demonstrated effectiveness in consistently improving CNN performance by reinforcing class-wise representativeness while preventing overfitting.

Keywords: Deep Learning, Feature-Aware Dataset Splitting, Closed-loop Deep Neural Networks Training Optimization.

TABLE OF CONTENTS

Resumo ..... **ii**

Abstract ..... **iii**

1 Introduction **1**

1.1 Motivation 6

1.2 Research Methodology 7

1.3 Research Questions and Objectives 10

1.4 Contributions 11

1.5 Structure of the Thesis 11

2 Literature Review **13**

2.1 Methodology 15

2.2 ANN's Structural Optimization 18

2.3 Hyperparameter Optimization 21

2.4 Dataset-based Optimization 22

2.4.1 Dataset Configuration Approaches 23

2.4.2 Dataset Enhancement Approaches 26

2.5 Other Relevant Optimization Approaches 35

3 Overview of the Proposed DL-oriented Optimizations **40**

3.1 Contextual Datasets 41

3.1.1 Grapevine leaves: a nature-based dataset 42

3.1.2 Bridge defects: a dataset focusing on man-made structures 43

3.1.3 External Imagery dataset 44

3.2 Exploratory Deep Learning Operations over Datasets 45

3.2.1 Preliminary experiments with the grapevine leaves dataset 45

3.2.2 Preliminary experiments with the bridge defects dataset 46

3.3 DL Workflow and the Integration of the Proposed Optimizations 48

3.4 Experimental design 51

3.4.1 Development Environment and Tools 51

3.4.2 Preliminary Benchmarking Trials for Training Configuration Fine-Tuning 52

3.4.3 Traditional dataset chunking and division methods for DL evaluation 55

3.4.4 Augmentation Techniques 56

3.4.5 Method Proposal Overview: Feature Engineering and Dataset Splitting Before Training 59

3.4.6 Method Proposal Overview: Feature-Aware Representativeness Enhancement via Closed-Loop Training 61
3.4.7 The Rationale for Selecting CNNs Over ViTs 63
4 Proposed Feature-Aware-based Dataset Splitting Optimization methods 65
4.1 Initial Data Handling and Feature Engineering 67
4.2 Feature Clustering Distance-based Image Selection 69
4.3 Feature Space Center-based Image Selection 75
4.4 Results Assessment 77
4.4.1 Learning Consistency Analysis 78
4.4.2 Assessment Using Test Set 80
4.4.3 Assessment with External Imagery Sets 81
4.4.4 Attention Mechanisms Assessment with External Imagery Sets 83
5 Proposed Metrics-based Closed-Loop Methodology 90
5.1 MbCL Approach Proposal: A Detailed Perspective 92
5.1.1 MbCL Training Process 94
5.1.2 Required Augmented Images Quantity 97
5.1.3 Validation-Based Class-Wise Stopping Criterion 98
5.2 Augmentation Techniques 99
5.2.1 Classical Augmentation 100
5.2.2 Generative Adversarial Network Augmentation 101
5.3 Results Assessment 103
5.3.1 cGAN Model Configuration: Implications and Outcomes 104
5.3.2 Model Performance Upon Test Set 109
5.3.3 Analysis Through External Imagery Data 116
6 Discussion and Conclusion 121
6.1 Research questions-oriented discussion 122
6.1.1 Addressing RQ1 122
6.1.2 Addressing RQ2 124
6.1.3 Addressing RQ3 125
6.2 Conclusions 127
6.3 Summary of Thesis Scientific Contributions 128
6.4 Possible Future Research Directions 134
REFERENCES 136

List of Abbreviations and Acronyms

A

AA	AutoAugment
AALO	Automatic Artificial Learning Optimization
Adam	Adaptive Moment Estimation
AdvAA	Adversarial AutoAugment
AI	Artificial Intelligence
ANNs	Artificial Neural Networks
AutoML	Automated Machine Learning

B

BatchNorm	Batch Normalization
BO	Bayesian Optimization

C

CA	Classical Augmentation
cDCGAN	conditional Deep Conditional Generative Adversarial Networks
cGAN	conditional Generative Adversarial Networks
CNN	Convolutional Neural Network
CwES	Class-wise Early Stopping

D

DADA	Differentiable Automatic Data Augmentation
DARTS	Differentiable ARchiTecture Search
DDAS	Direct Differentiable Augmentation Search
DeepAA	Deep AutoAugment
DL	Deep Learning
DRL	Deep Reinforcement Learning
DSRM	Design Science Research Methodology

E

EA	Evolutionary Algorithm
ENAS	Efficient Neural Architecture Search
ESE	Early Stopping Epochs

F

FAA	Fast AutoAugment
FAbDSO	Feature-Aware-based Dataset Splitting Optimization
FasterAA	Faster AutoAugment
FCDIS	Feature Clustering Distance-based Image Selection
FSCIS	Feature Space Center-based Image Selection

G

GA	Genetic Algorithm
GANs	Generative Adversarial Networks
GMM	Gaussian Mixture Models
Grad-CAM	Gradient-weighted Class Activation Mapping
GS	Grid Search

H

HBA	Hypernetwork-Based Augmentation
HDBSCAN	Hierarchical Density-based Spatial Clustering of Applications with Noise
HO	Hold Out
HOG	Histogram of Oriented Gradients

I

InceptionV3	Inception Network-Version3
InceptionResNetV2	Inception Residual Network-Version 2
IoU	Intersection Over Union

K

KFCV	K-Fold Cross Validation
KNN	K-Nearest Neighbors
KS	Kennard-Stone

L

LEMONADE	Lamarckian Evolutionary Algorithm for Multi-Objective Neural Architecture Design
LeakyReLU	Leaky Rectified Linear Unit
LR	Learning Rate
LSTM	Long Short-Term Memory

M

MbCL	Metrics-based Closed-Loop
MbCL-CA	MbCL with classical augmentation
MbCL-GA	MbCL with GAN-based augmentation
ML	Machine Learning
MLMV	Moralis Lima Martin Validation
MLP	MultiLayer Perceptron
MobileNetV2	Mobile Neural Network-Version2

N

Nadam	Nesterov-accelerated Adaptive Moment Estimation
NAS	Neural Architecture Search
NASNetMobile	Neural Architecture Search Network-Mobile
NLP	Natural Language Processing

P

PBA	Population-Based Augmentation
PBT	Population-Based Training
PCA	Principal Component Analysis
PCSs	Pearson Correlation Scores
PPO	Proximal Policy Optimization
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses

Q

DQN	Deep Q-Networks

R

RBS	R-value-Based Sampling
ResNet50	Residual Network-50
RF	Random Forest
RL	Reinforcement Learning
RMSProp	Root Mean Square Propagation
RNN	Recurrent Neural Network
RoCGAN	Robust Conditional GAN
RQ	Research Question
RLT	Regular loopless training

RS	Random Search

S

SA	SmartAugment
SAGAN	Self-Attention Generative Adversarial Networks
SGD	Stochastic Gradient Descent
SIFT	Scale-Invariant Feature Transform
SMBO	Sequential Model-based Optimization
SPXY	Sample Partitioning based on joint X-Y distance
SSA	SmartSamplingAugment
SURF	Speeded-Up Robust Features
SVM	Support Vector Machine

T

TA	TrivialAugment
TPE	Tree-structured Parzen Estimator
TPOT	Tree-Based Pipeline Optimization Tool

U

US	Uniform Search

V

VGG	Visual Geometry Group Network
ViT	Vision Transformer

X

XAI	Explainable AI
Xception	Extreme Inception
XGBoost	Extreme Gradient Boosting

Y

YOLO	You Only Look Once

LIST OF FIGURES

Figure 1. A visual representation showing the hierarchical relationship between DL, ML, and AI — adapted from [14]..... 2
Figure 2. The DSRM Process Model was adapted from [85]. 7
Figure 3. The PRISMA flow diagram for collected data. 16
Figure 4. Distribution of reviewed publications by publisher. 17
Figure 5. Yearly distribution of reviewed publications from before 2000 through 2025. 17
Figure 6. Raw data of grapevine leaves composed of six classes (phenotypes) acquired over a white background. From left to right: *Touriga Nacional, Tinto Cão, Códega, Moscatel, Tinta Roriz,* and *Rabigato*. 43
Figure 7. Different types of defects that are commonly found in bridges. 43
Figure 8. External image sets for DL models extended assessments: features a set of images for a) grapevine leaves imagery, b) bridge defects imagery. 44
Figure 9. Grad-CAM heatmaps comparison for the presence of plants in the image, with different prediction probabilities. 48
Figure 10. Dataset preparation processes. 49
Figure 11. Methodology general workflow. 50
Figure 12. Various DL architectures: a) Xception, b) NASNetMobile, c) InceptionV3, d) VGG16, e) MobileNetV2, and f) InceptionResNetV2, adopted from previously published works [303], [92],[304],[165],[306],[305]. 54
Figure 13. cGAN network architecture. 59
Figure 14. General workflow of the split process 60
Figure 15. Overview of the MbCL methodology. 63
Figure 16. Key steps of PCA. 68
Figure 17. Visualization of the centroid point of features for the class of "Absence of Joint Cover Plate" for $DTr - ValBDI$, for exemplification purposes. Blue dots represent the distribution of features, while the red cross is the global centroid. 69
Figure 18. FCDIS process for setting up training and validation subsets: a) presents the main pipeline of the process; b) depicts the pipeline's general result-oriented concept using 3 abstract clusters as examples. 71

Figure 19. Example of K-means-based clustering process plot perspective for the "Absence of Joint Cover Plate" class in $DTr - ValBDI$. a) is a chart depicting the K value determined from the Elbow technique, and b) presents the clustering results. A black star indicates the global center, and colored crosses mark the cluster centers. 72
Figure 20. Visualization of a few samples of each cluster for the "Absence of Joint Cover Plate" class from $DTr - ValBDI$. a) depicts the images from clustering results; b) shows a summary of the images collected for DTr, considering each cluster's distance and size. 74
Figure 21. FSCIS process for setting up training and validation subsets: a) presents the main pipeline of the process; b) depicts the result of the pipeline behavior. 76
Figure 22. Learning curves, comparing the behaviour of models' learning over datasets produced based on traditional techniques and FCDIS/FSCIS methods: a) Xception Nadam for $DTr, ValBDI$; b) Xception Nadam for $DTr, ValGLI$. 79
Figure 23. Visualization of the Grad-CAM evaluation approach. a) and b) show the saliency maps over $DExt_TstGLI$ and $DExt_TstBDI$, respectively. In each one, the first row corresponds to the Grad-CAM visualization, while the second row presents the ground-truth. 84
Figure 24. Attention visualization for $DExt_TstGLI$ considering the top Xception/Nadam model trained with the splitting techniques. 86
Figure 25. Attention visualization for $DExt_TstBDI$ considering the top Xception/Nadam model trained with the splitting techniques. 87
Figure 26. Overview of the dataset preparation workflow. 93
Figure 27. Overview of the methodology workflow. 94
Figure 28. Methodological steps of the MbCL training process. 95
Figure 29. The structure of the cGAN is adapted from [234]. 102
Figure 30. The learning curve of the cGAN training model upon a) $DGLI$ and b) $DBDI$. 105
Figure 31. Generated samples using cGAN. For both $DGLI$ (a) and $DBDI$ (b), the first row corresponds to real images, while the 2nd and 3rd rows are the generated samples for each class. 108
Figure 32. Samples of visual performance assessment for two classes of each dataset: a) $DExt_TstGLI$, b) $DExt_TstBDI$ 119

LIST OF TABLES

Table 1. Summary of ANN´s structural optimizations. 20
Table 2. Summary of hyperparameter optimization techniques 22
Table 3. Summary of dataset configuration approaches. 25
Table 4. Summary of data augmentation methods and their purposes. 31
Table 5. Summary report on computation time and performance of the automatic augmentation approaches. * The dataset used was CIFAR-100-C. 32
Table 6. Summary of generative dataset enhancement approaches. 34
Table 7. Overview of additional miscellaneous optimization approaches. 38
Table 8. Distribution of samples across classes in the Grapevine Leaves imagery. 42
Table 9. Distribution of samples across classes in the Bridge Defects Imagery. 43
Table 10. Accuracy results for models trained with $DTstBDI$. "Acc" refers to classification accuracy (%). ↑, ↓ and = indicate, respectively, if HO and KFCV are better, worse, or equal to the best proposed splitting techniques(FCDIS/FSCIS). 80
Table 11. Accuracy results for models trained with datasets $DTstGLI$. "Acc" refers to classification accuracy (%). ↑, ↓ and = indicate, respectively, if HO and KFCV are better, worse, or equal to the best proposed splitting techniques(FCDIS/FSCIS). 81
Table 12. Accuracy of models using the $DExt_TstBDI$. "Acc" refers to classification accuracy (%). ↑, ↓ and = indicate, respectively, if HO and KFCV are better, worse, or equal to the best proposed splitting techniques(FCDIS/FSCIS). 82
Table 13. Accuracy of models using the $DExt_TstGLI$. "Acc" refers to classification accuracy (%). ↑, ↓ and = indicate, respectively, if HO and KFCV are better, worse, or equal to the best proposed splitting techniques(FCDIS/FSCIS). 83
Table 14. Prediction results upon the $DExt_TstGLI$ for Xception Nadam. 88
Table 15. Prediction results for the proposed dataset splits for the Xception model using Nadam optimizer upon $DExt_TstBDI$. *AJCP - Absence of Joint Cover Plate; PC - Pavement Crack; PoC - Peeling on Concrete*. 88
Table 16. Best FID Score over the cGAN training steps upon $DGLI$ and $DBDI$ 107
Table 17. Results of the accuracy of classes upon the $DValGLI$ for Xception with and without closed-loop augmentation techniques using Nadam optimizer for LR=10^{-3} 110

Table 18. Results of the accuracy of classes upon the $DValBDI$ for Xception with and without closed-loop augmentation techniques using Nadam optimizer for LR=10^{-3}..... 111
Table 19. Class-wise accuracy upon the $DTstGLI$ for different architectures Xception, VGG16, ResNet50, with and without closed-loop augmentation techniques using Nadam and SGD optimizers for LR=10^{-3} and 10^{-4}. 112
Table 20. Class-wise accuracy upon the $DTstBDI$ for different architectures Xception, VGG16, ResNet50, with and without closed-loop augmentation techniques using Nadam and SGD optimizers for LR=10^{-3} and 10^{-4}. 113
Table 21. The accuracy of the models upon $DTstGLI$ for different architectures Xception, VGG16, ResNet50, with and without closed-loop augmentation techniques using Nadam and SGD optimizers for LR=10^{-3} and 10^{-4}. 115
Table 22. The accuracy of the models upon the $DTstBDI$ for different architectures Xception, VGG16, ResNet50, with and without closed-loop augmentation techniques using Nadam and SGD optimizers for LR=10^{-3} and 10^{-4}. 115
Table 23.Prediction results upon $DExt_Tst$ GLI for Xception using Nadam with LR=10^{-3}..... 117
Table 24. Prediction results upon $DExt_TstBDI$ for Xception using Nadam with LR=10^{-3}. 117
Table 25. Accuracy of the classification upon the $DExt_TstGLI$. 118
Table 26. Accuracy of the classification upon the $DExt_TstBDI$. 118

1 Introduction

Artificial Intelligence (AI) is the broad field of computer science aimed at developing approaches that approximate human capabilities in tasks such as problem-solving, decision-making, image interpretation, and language understanding, among others. It has been playing a crucial role in the development of cutting-edge solutions for various professional fields, ranging from industry [1], [2] to medicine [3], [4], encompassing several tasks of major importance, such as image classification [5], image segmentation [6], object detection [7], and language modeling [8].

Within AI (Figure 1) lies Machine Learning (ML), a subarea that enables systems to learn from data and improve performance without explicit programming [9]. ML relies on statistical methods to recognize patterns and make predictions, and it is further divided into approaches based on how data is used for training [10]. In turn, Deep Learning (DL) is a specialized subset of ML [11], which focuses on using ANNs composed of multiple connected layers and mathematical function units (neurons) aimed at grasping complex patterns in data [12]. These layers allow DL structures to learn hierarchical representations, making them highly effective for tasks like image recognition, Natural Language Processing (NLP), and autonomous systems. DL stands out because it reduces the need for manual feature extraction, as the system learns relevant features directly from raw data through training [13].

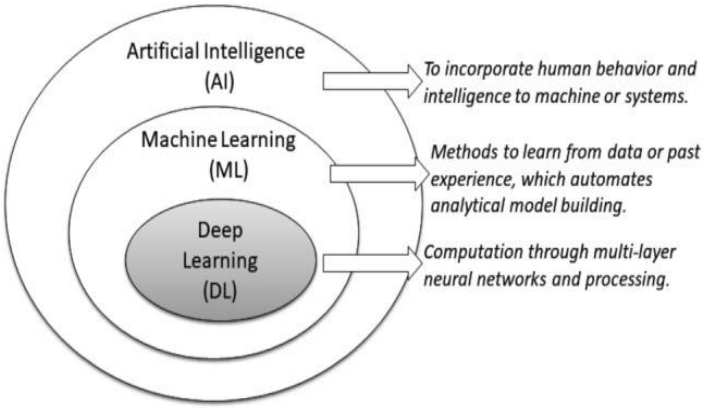


Figure 1. A visual representation showing the hierarchical relationship between DL, ML, and AI — adapted from [14].

Building on this foundation, predictive ML/DL involves four fundamental approaches: supervised learning, semi-supervised learning, unsupervised learning, and Reinforcement Learning (RL) [15]. Supervised learning consists of training models based on labeled datasets where the output is known, making it suitable for tasks like classification and regression [16], [17]. Semi-supervised learning, on the other hand, operates with a mix of labeled and unlabeled data, often leveraging the small amount of labeled data to improve the performance of models on larger unlabeled datasets [18]. This approach is particularly useful in scenarios where labeling data is expensive or time-consuming, such as medical image analysis [19], [20]. Unsupervised learning deals exclusively with unlabeled data,

aiming to identify patterns, groupings, or structures within the data [21], [22]. Techniques like clustering and dimensionality reduction fall into this category, commonly applied in customer segmentation [23], [24], [25] or exploration of data analysis [26], [27]. RL differentiates from the former ones by employing agents that learn through interaction with a constrained environment by receiving feedback in the form of rewards or penalties [28], [29]. This trial-and-error-based approach is ideal for applications like robotics [30], [31], [32], gaming [33], [34], and real-time decision-making [35], [36], [37].

Notably, the complexity of developing solutions based on these approaches is significant, requiring expert intervention in tedious and time-consuming trial-and-error experimentation and fine-tuning processes to achieve satisfactory computer-based analytical models. To mitigate such issues, strategies relying on Automatic Artificial Learning Optimization (AALO) can be applied to improve the ML pipeline model by automating crucial steps inherently associated, for example, with data preparation and model training. This automation not only helps prevent human errors by systematizing important procedures that become computer-managed, but also opens avenues for the inclusion of non-experts, thus bringing society one step closer to the democratization of AI.

The ML/DL automation towards model development consists of the following steps: i) architecture optimization, consisting of a search space-based determination of the number and structure of layers, among other required components; ii) hyperparameter tuning, and iii) data engineering [38]. These steps are crucial to achieving computational approaches for automatic data/image analysis. More specifically, while conducting image analysis using DL techniques [39], [40],[41] multiple stages are involved in the development of models to enhance their performance, from raw data organization to the interpretation of post-training results. In the case of supervised learning techniques — more aligned with this thesis work —, having labeled data of good quality is an important contribution to reaching a model with adequate performance to operate in the context to which it was designed [42]. Such a condition paves the way for the model to effectively learn the relations between the image features and assigned labels. In the following, each stage is explained:

1. **From raw data gathering to the dataset implementation — traditional splitting techniques**

ML/DL-based solutions commonly rely on contextual datasets tailored to a specific domain or task. A straightforward way to obtain such datasets is through public repositories. Another approach is to collect data directly from the environments in which the ML/DL solutions are intended to operate,

typically using specialized acquisition devices, such as RGB, multispectral, or hyperspectral sensors. These sensors not only capture spatial information but also provide multiple spectral bands that may carry relevant domain-specific features. In terms of quality, the imagery quality can vary not only due to the specification of the sensors but also because of environmental factors such as illumination, climatic conditions, and physical occlusions, among others. Even though this latter method is time-consuming and susceptible to environmental conditions, it also provides unique opportunities to capture images that can be fully compliant with the ML/DL requirements of specific contexts. Then the collected data must be manually validated and checked for any inconsistencies and lack of relevancy. This cleaning process aims to avoid negative impacts on the ML training process. The next step involves labeling the data before dividing it into three typical subsets: training, validation, and testing. The training set enables the analysis, processing, and extraction of relevant features. The validation set is used for tuning hyperparameters and preventing overfitting, as a response to the correspondence of features found in this subgroup with the ones computed in the former set. Commonly, both training and testing sets are used complementarily while the model is being trained and evaluated. Finally, the testing set reserves a group of data that the model is unaware of to assess its performance in a distinct step that usually follows the training process. Still, in the context of DL training, one of the main challenges is the dataset imbalance [43], [44], which may lead to biases with severe negative impacts on the models' inference capabilities.

2. Data balancing and augmentation

Imbalanced datasets are one of the main issues in ML/DL tasks, affecting models' performance [45], [46]. More specifically, models trained with datasets composed of classes with significant variations in the number of samples are more likely to become biased. To tackle this issue, data balancing and augmentation techniques [47], [48], [49], [50] are typically applied. One of the common approaches is traditional data augmentation [51], [52], [53] using cropping, rotation, color transformation, etc. Also, Generative Adversarial Networks (GANs) can balance the dataset by generating synthetic data [54], [55].

3. Popular DL optimizations

In DL, there are some approaches to enhance the performance of the learning process. One of them is based on dataset engineering [56]. The other noteworthy aspects are based on the model architecture [57], [58], and hyperparameter tuning [59], [60], [61] being used to lead the most

accurate and optimized model. The application of these methods can range from the early data preparation stage to the steps immediately before the start of model training.

Another recurring issue in supervised learning-based ML/DL that can be addressed from an optimization perspective is overfitting – i.e., the model performs well on the training data but behaves poorly on the validation data. One of the strategies to tackle such issues is by resorting to dataset-based optimization. More specifically, techniques such as K-Fold Cross Validation (KFCV) [62], [63] and Hold-Out (HO) can be used to perform systematized dataset splitting operations that are submitted to training sessions for the assessment of a given model's learning potential. Moreover, by optimizing hyperparameters, the Grid Search (GS) technique [64] helps to find the ML/DL configurations that increase the potential for convergence while dealing with overfitting susceptibilities. Also, training optimizer [65] play a relevant role in the scope of ML/DL optimization, as they govern the update dynamics of model parameters. Their behavior can be predefined through scheduling strategies or adaptively adjusted in response to model performance during training, to converge toward global minima effectively.

4. Models' results interpretation

The post-training results interpretation of a model is a key step for anticipating its effectiveness in performing useful inferences. Among the elements that are advisable for analysis, the plots of training and validation curves present the model's progress at the different stages of the training process. These plots help to clarify whether the model is likely to perform well or not (in case of overfitting or underfitting). A more realistic performance assessment can only, however, be done using metrics such as accuracy, F1-score, precision, and recall. Also, Intersection Over Union (IoU) [66], [67], can be used to evaluate the performance of the model by measuring the overlap between the predicted and the actual areas regarding an element of interest to be detected. Eventually, for better understanding and explaining the decision-making of the model, the Explainable AI (XAI) [68], [69], [70] can be utilized based on, for example, Gradient-weighted Class Activation Mapping (Grad-CAM) [71] in which heatmap gradients can be obtained, illustrating the areas to which the model gave more attention.

Next, the motivation and objective for this PhD work will be presented, followed by the research questions and scientific research methodology.

1.1 MOTIVATION

Digital transition is occurring at a fast pace and with a great focus on the integration of smart services across the various professional fields, to improve several aspects related to, for example, production, maintenance, and decision-making, in which AI has been playing a vital role. The implementation of ML systems requires experts who can develop and apply inference models of excellent performance, which usually involve complex steps that may need to be repeated over time for fine-tuning purposes. Therefore, researchers have been proposing DL architectures and automation strategies, providing, however, little attention to data features reasoning, which is usually left to the responsibility of DL "black-boxes".

Today, there are numerous AI architectures designed to tackle virtually every challenge. One of them, related to ML scope, is to define the proper model and fine-tune hyperparameters, in which Automated Machine Learning (AutoML) can be highlighted. Its main objective is to automate complex, time-consuming processes that require expertise [72]. These processes include data preprocessing, feature and model selection, hyperparameter adjustment, and the evaluation process [73]. They make ML more effective and accessible, so that both experts and non-experts can use it to create accurate models. According to the input data type, AutoML chooses the proper architecture and hyperparameters to be trained in. Moreover, it fine-tunes the hyperparameters using GS [74] or Random Search (RS) [75], among others. Although, according to Andrew Ng [76], it's now time to shift focus to data-centric AI, where the quality and handling of data take center stage as the main fuel for AI deployment. Data-centric involves any possible procedures upon datasets, from preparing data to enriching it. One way is to split the given dataset using traditional techniques like KFCV, which helps enhance the model's performance. In this technique, the model should be trained K times across subset arrangements, which is computationally heavy and time-consuming. The other data-centric strategies, like data augmentation, have some limitations that are worth highlighting. The representativeness of data is one that stands out. Even though the data augmentation techniques are used — among other goals — to balance datasets, they cannot produce original features, as the generated variants are based on existing data that defines the characteristics of the group. Therefore, considering a given image set, if a certain element of interest shows up in consistent conditions (e.g., uniform background and unvarying orientation), an overrepresentation may be induced, leading to biased conceptions of reality from the perspective of the AI model that learned from that data. Another issue related to generated data is the unconvincing synthetic variants that may be spawned, which,

in some contexts, can cause constraints (e.g., marketing [77] and medicine [78]). These problems can affect the performance of the models negatively.

Therefore, the purpose of this work is to consider the aforementioned challenges and to propose automated data-centric feature-based optimization techniques for computer vision tasks. More specifically, feature mining techniques are explored in two fundamental steps of a common DL pipeline: the dataset preparation stage and the training process. Rather than focusing on inter-class example balancing, this approach targets even class-wise feature distribution — both during dataset preparation and splitting, and then, throughout the closed-loop training process, wherein feature-aware representativeness is strategically enhanced until model performance begins to degrade.

1.2 Research Methodology

Recently, Design Science Research Methodology (DSRM) has been evolving from early approaches [79], [80] to more advanced conceptual frameworks [81], [82], [83], [84], being the work proposed by Peffers et. al [85] — as a derivation from the previous literature — the most comprehensive one.

In this research, the stages of DSRM (Figure 2) are used as the main framework for developing and evaluating our artifacts and answering Research Questions (RQs). Worth noting that our research type is **applied research**, aiming to solve real-world problems.

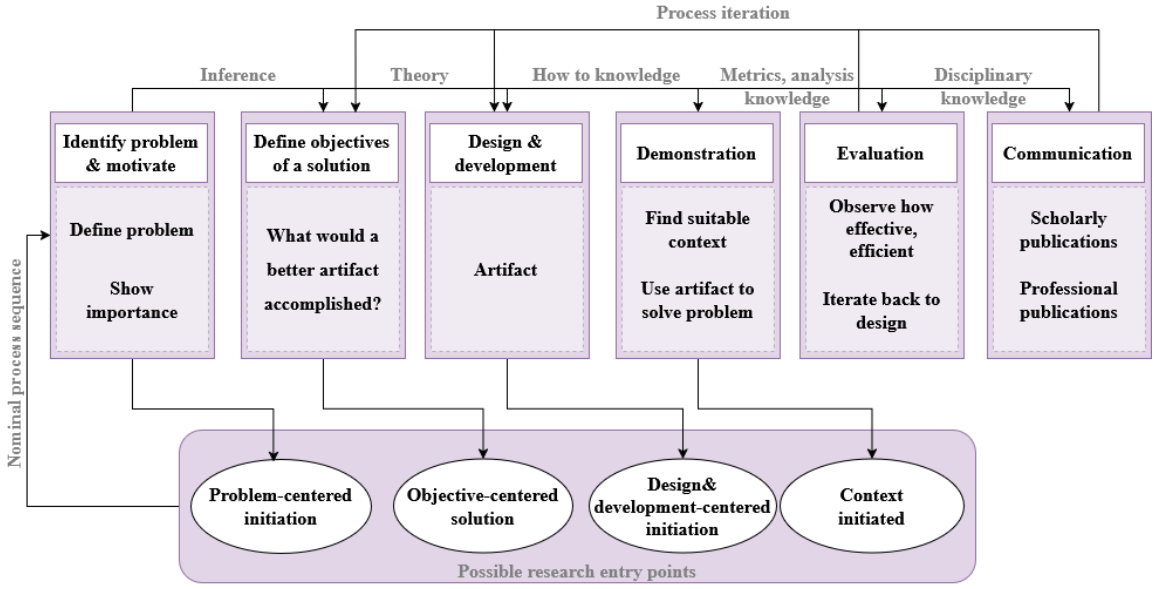


Figure 2. The DSRM Process Model was adapted from [85].

These steps have been redefined as follows to meet the specific needs of this research:

1- **Identifying the problem and motivation —** At this stage, the main problem of the research to be carried out is identified and contextualized. In this stage, the importance of the highlighted challenge(s) must be clarified.

   The key challenges addressed in this research include: (i) improving datasets and DL training by assessing feature distributions and optimizing data allocation across training, validation, and testing subsets – instead of relying on random or fold-based splitting; and (ii) evaluating the contribution of closed-loop-based sampling for the robust reinforcement of underrepresented classes whose feature imbalance potentially prevents models from reaching their full potential.

2- **Defining the objectives for a solution —** After establishing a clear and representative framework of the problem and its motivation, the objectives are outlined. There are three possible types of objectives, including qualitative, quantitative, and mixed, which should also be measurable in terms of concretization.

   This research sets mixed objectives, both quantitative and qualitative, aiming at optimizing DL training. The main ones are listed as follows:

   - To make dataset exploration more expeditious and effective, through computational/numerical/statistical image-based deep feature inspection and strategic splitting;
   - To explore feedback-driven strategies in line with DL training toward the balancing of class-wise representativeness, improving underperforming classes – and therefore overall model performance –, reducing bias, while surveilling potential aspects leading to training degradation.

   Success can be measured through: (i) the attainment of replicable processes capable of effectively tackling the identified issues of traditional/known methods; and (ii) DL metrics (accuracy, IoU, Grad-CAM-based explainability, etc.) to compare models trained with the proposed approach against the ones produced based on traditional/known procedures.

3- **Designing and development —** In this stage, the materialization of the artifact is pursued, which essentially relies on the proposed DL optimization approaches – combining innovative or literature-inspired methods, techniques, and procedures – that will increment the current state-of-the-art in alignment with the specified objectives.

   Therefore, the tasks supporting this work focus on investigating and proposing approaches to optimize DL deployment in two stages. The first occurs before training, using one-shot

strategies to achieve balanced dataset splitting by distributing images across training, validation, and testing sets based on an inspection of their (deep) features. The second operates during training, applying feature-aware augmentation/generation of examples for enhancing underrepresented classes, guided by feedback metrics obtained from checkpoint epochs in a closed-loop process that continues until signs of performance decline appear. In both cases, metric-based analysis of the resulting CNN models provides measurable references for evaluating the effectiveness of the proposed approaches.

4- **Demonstration —** The proposed approaches originate DL models that are crucial for demonstration purposes. In this stage, such models must show robustness across varying conditions in the representation of the artifact's capabilities.

   This requirement applies to both the feature-aware splitting strategy and the class-wise, metrics-based closed-loop approach, which should be capable of operating effectively on unseen test sets from involved datasets, as well as on external imagery sources that share similar context and labeling but differ in acquisition conditions.

5- **Evaluation —** Observing and measuring the performance of the artifacts is crucial to benchmarking with other well-established methods with similar goals.

   In the evaluation stage, the performance of the proposed methods relies on the use of multiple metrics. The evaluation comprises the analysis of training/validation curves, accuracy and loss values, metrics extracted from unseen test sets belonging to involved datasets and from external imagery sources that share similar context and labeling but differ in acquisition conditions, as well as spatial alignment metrics such as IoU and centroid displacement can be used to assess the precision of attention maps generated by Grad-CAM.

6- **Communication —** Scientific communication plays a crucial role in the development and validation of research. Progressively reporting issues encountered, strategies adopted to address them, and the resulting findings allow researchers to contribute to the collective advancement of knowledge while obtaining valuable external feedback. Sharing work in progress also promotes transparency, invites constructive criticism, and paves the way to collaboration and improvement.

   Typical venues for disseminating such contributions include peer-reviewed conferences and journals. Conferences provide a dynamic forum for presenting preliminary results, discussing methodologies, and receiving real-time input from peers. Journals, in contrast, are often

intended for more mature and consolidated research work. Moreover, the scope and topics covered by these venues should align with the research objectives. In the specific context of this thesis, target topics include computer vision, artificial intelligence, computational methods, numerical analysis, and related fields.

This model is iterative, and just like in an agile project development methodology, the researcher can rewind from stages 5 or 6 to stages 1 or 2. Such flexibility enables to better define and refine the problem under analysis, as well as the respective tackling solution proposal.

1.3 Research Questions and Objectives

Given that the current state of the art primarily focuses on ANN architectures for automatic ML, and dataset optimization is scarcely addressed, this work aims to develop dataset-driven ML-based techniques for computer vision tasks in AALO. To fulfill this goal, firstly, the RQs need to be identified.

RQs are essential in science, constituting the guidelines to approach the respective objectives [86], [87], [88]. In the context of this work, the RQs are the following:

- RQ1 — Can feature-aware dataset splitting improve the performance of the models and constitute a viable alternative to traditional techniques exploring learnability potential (e.g., KFCV)?
- RQ2 — Is it beneficial to augment data strategically to increase class-wise representativeness to achieve a more adequate equilibrium in terms of the feature set, rather than keeping the cardinality of the categories balanced within a given dataset?
- RQ3 — Can a self-regulated closed-loop strategy be implemented to perform dataset-related optimizations, for consistently enhancing the quality of DL models targeting classification tasks in diverse contexts (man-crafted vs. natural environments)?

To investigate and answer the mentioned questions and solve the challenges, the objectives of this research are formulated as follows:

1- To perform an extensive literature review focusing on ML-based optimization approaches and to survey the proper assessment metrics.

2- To propose and develop a novel multi-context dataset-driven splitting methodology for image classification, aiming at the close to optimal one-shot data organization, suited for enhanced DL model training.

3- To propose and develop an iterative and self-managed multi-context closed-loop approach aiming at the improvement of class-wise representativeness.
4- To assess and discuss the results of the proposed techniques, based on standard DL-oriented metrics, involving datasets from different contexts (construction industry and agriculture).

1.4 CONTRIBUTIONS

In the scope of this work, the following set of contributions covering review, research, and development stages is pursued:

1- A comprehensive literature review to investigate the current optimization methodologies in ML/DL;
2- Exploration of DL standard procedures and good practices ranging from dataset arrangement strategies to the selection of key hyperparameters and Convolutional Neural Network (CNN) architectures, considering contextual variability reflected on a couple of authoring datasets – i.e., agricultural and construction industry environments;
3- Definition of a novel methodology to optimize DL operations covering two relevant stages – dataset preparation and model training – to tackle current methods' burdensomeness and manageability drawbacks (e.g., KFCV), as well as to mitigate feature-aware representativeness unevenness;
4- At the dataset configuration stage, proposal of novel methods oriented to perform one-shot feature-aware data distribution among training, validation, and testing subsets, to enhance intra-class diversity;
5- Aligned with the model training stage, a proposal of a class-wise closed-loop optimization technique to reinforce representativeness of contextual categories, simultaneously monitoring metrics to halt sampling when training starts to degrade.

1.5 STRUCTURE OF THE THESIS

This document is organized into six chapters, including this introduction. Next, Chapter 2 consists of a preliminary literature review, describing the current approaches related to CNN learning optimization, covering optimization techniques for ANN´s structural, hyperparameter, and dataset-based, and other relevant optimization techniques. Chapter 3 presents a set of preliminary DL experiments, not only providing a context for the considered distinct operational scenarios, but mostly

to lay the groundwork for subsequent stages of the pipeline, including decisions related to hyperparameters and architectural configurations. Additionally, it provides an overview of the DL optimization methods proposed in this thesis, while it illustrates the complete workflow and details the experimental design. Chapters 4 and 5 explain in more detail the proposed feature-based dataset splitting and the Metrics-based Closed-Loop (MbCL) techniques, respectively. In both chapters, results are assessed and discussed. Finally, Chapter 6 provides a broader discussion of the results concerning the RQs and presents the conclusions, which also sum up the published scientific contributions that resulted from this thesis.

2 Literature Review

With rapid progress in ML/DL-based solutions, innovation in optimization techniques, dataset handling, and training methodologies has become critical for building high-performing models [89]. Achieving optimal performance in supervised ML requires a comprehensive understanding of how different components — the model architecture, hyperparameters, and datasets — can be effectively optimized [90]. Optimization can be performed manually or automatically [91] targeting the structure of Artificial Neural Networks(ANN) [92], the configuration of hyperparameters [93], or the quality and structure of the training data [94].

Typically, the supervised ML/DL pipeline begins with data preparation [95], involving tasks such as data collection, cleaning, labeling, splitting, augmentation, and synthetic data generation [96]. Once the data is prepared, the next crucial step involves selecting and configuring the ANN architecture — e.g., CNN for vision tasks [97]. ANN structural optimization has been a prominent topic to which recent approaches — such as Neural Architecture Search (NAS) [98], network pruning, and quantization — aim to automate and refine the design of these architectures to improve performance and efficiency. Equally important is the optimization of hyperparameters, which significantly affects training dynamics and final model performance. Automated hyperparameter tuning methods — such as Bayesian Optimization (BO) [99], Evolutionary Algorithm (EA) [100], and gradient-based approaches [101] — have emerged to ease this process, replacing labor-intensive manual trial-and-error procedures. Beyond model architecture and parameters, dataset-based optimization plays a vital role in improving learning outcomes [102]. This includes both dataset structuring — such as intelligent data splitting for balanced learning [103] — and dataset enhancement, where techniques like data augmentation [104] and GANs [105] introduce variability and boost generalization. Lastly, this chapter transversally explores optimization approaches that span multiple pipeline stages. For example, AutoML [72], [73], [74], [106], [107] for full ML workflow automation, including ANN design, hyperparameter tuning, and data preprocessing, are part of the topics encompassed. Additionally, techniques inspired by RL [108] are also addressed, as they play a role in adaptive optimization processes that align well with the scope of this thesis.

The chapter is structured as follows: The following subsection outlines the methodology employed for the literature review. Section 2.2 focuses on ANN structural optimization, covering NAS, pruning, and related techniques. Section 2.3 discusses hyperparameter optimization, highlighting both manual and automated tuning methods. Section 2.4 presents dataset-based optimization and is further divided into two subsections: Section 2.4.1 on dataset structuring, which includes strategies such as

data splitting and balancing, and Section 2.4.2 on dataset enhancement approaches, which cover techniques like data augmentation and the use of GANs to enrich training data. Finally, Section 2.5 concludes with other relevant optimization approaches, including AutoML and reinforcement learning, which provide a broader perspective on the automation of various stages in modern ML pipelines.

2.1 METHODOLOGY

This literature review was conducted under the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) guidelines, a widely accepted methodology in the scientific community that aligns with the standards of many peer-reviewed journals [3], [109], [110], [111]. To collect the proper data, we systematically searched databases, including MDPI, IEEE Xplore, ScienceDirect, ACM Digital Library, Springer, and Google Scholar. The search strategy utilized a range of keywords, both independently and in combination, such as: ("AutoML" OR "Closed-loop Optimization") AND ("Computer Vision" OR "Convolutional Neural Network") AND ("Data Augmentation" OR "Generative Adversarial Network-based Augmentation") AND ("Deep Learning" OR "Machine Learning") AND ("Dataset Learning Optimization" OR "Dataset Splitting Optimization") AND ("Explainable Artificial Intelligence" OR "Metrics Monitoring") AND ("Feature Clustering" OR "Feature Selection") AND ("Neural Architecture Search" OR "Reinforcement Learning").

Duplicate records were removed through an initial screening process. Subsequently, abstracts were reviewed to filter out irrelevant publications, and a full-text analysis was performed on the remaining articles to assess their suitability for inclusion. Based on this multi-stage filtering process, the final set of studies was selected for detailed analysis.

The reviewed works were then categorized according to three major machine learning optimization strategies: (1) hyperparameter optimization, (2) ANN´s structural optimization, and (3) dataset-based optimization. These categories represent the core focus areas where most optimization efforts have been directed to enhance model accuracy, efficiency, and generalization. However, in addition to these three strategies, the literature also reveals a growing body of work dedicated to other relevant optimization techniques. These include emerging approaches such as AutoML, RL-driven optimization, and NAS, among others.

The review process was conducted in two main stages. The first one took place at the beginning of this work, aiming to collect and synthesize the state of the art from the past five years. The findings from this initial review were subsequently shared with the scientific community through publication.

Following a strategy of continuous surveillance of the relevant sources of literature, in the second stage, the review was updated and expanded as the research progressed, aiming to include relevant scientific contributions to the state-of-the-art.

Following this methodology, illustrated in Figure 3, a total of 11,000 records were identified through database searching. After removing duplicates, 9,700 records remained. These were screened for relevance based on titles and abstracts, resulting in 1,130 full-text articles assessed for eligibility. Following a detailed evaluation, 317 records met the inclusion criteria and were included in the final analysis.

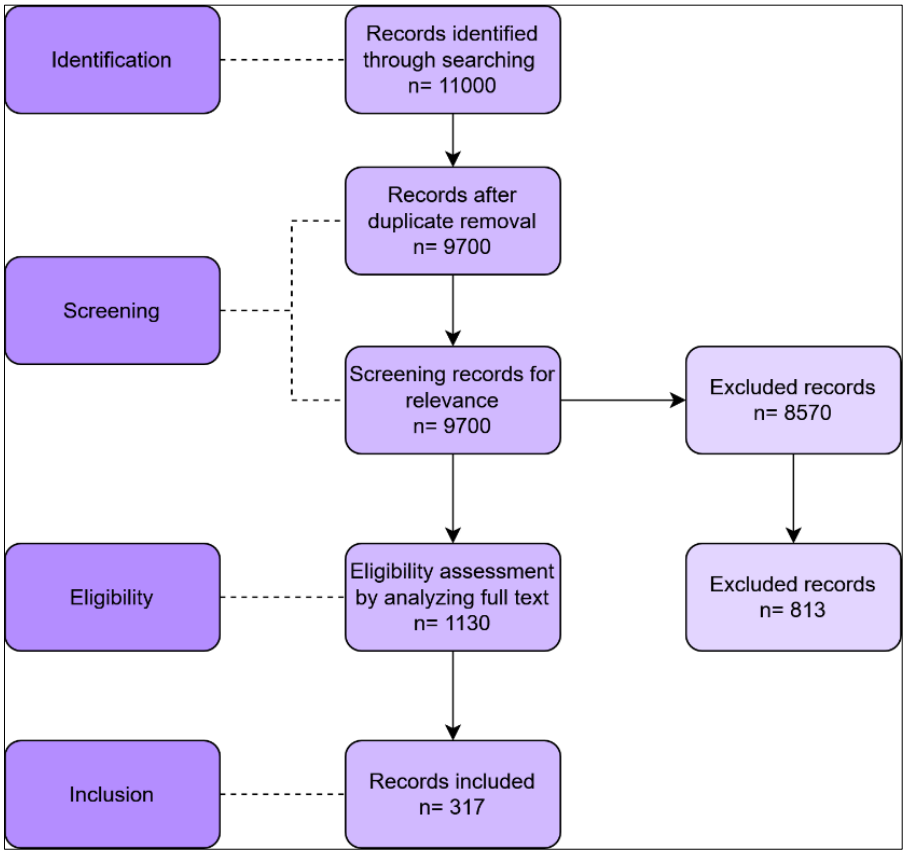


Figure 3. The PRISMA flow diagram for collected data.

Figure 4 presents the distribution of the selected publications across major scientific publishers. Among the retrieved contributions, 60% were obtained from major scientific publishers (Elsevier/ ScienceDirect, IEEE, MDPI, Springer, ACM), typically governed by established peer-review processes, whereas the remaining percentage corresponds to other sources, including — but not confined to — non-peer-reviewed preprints (e.g., arXiv).

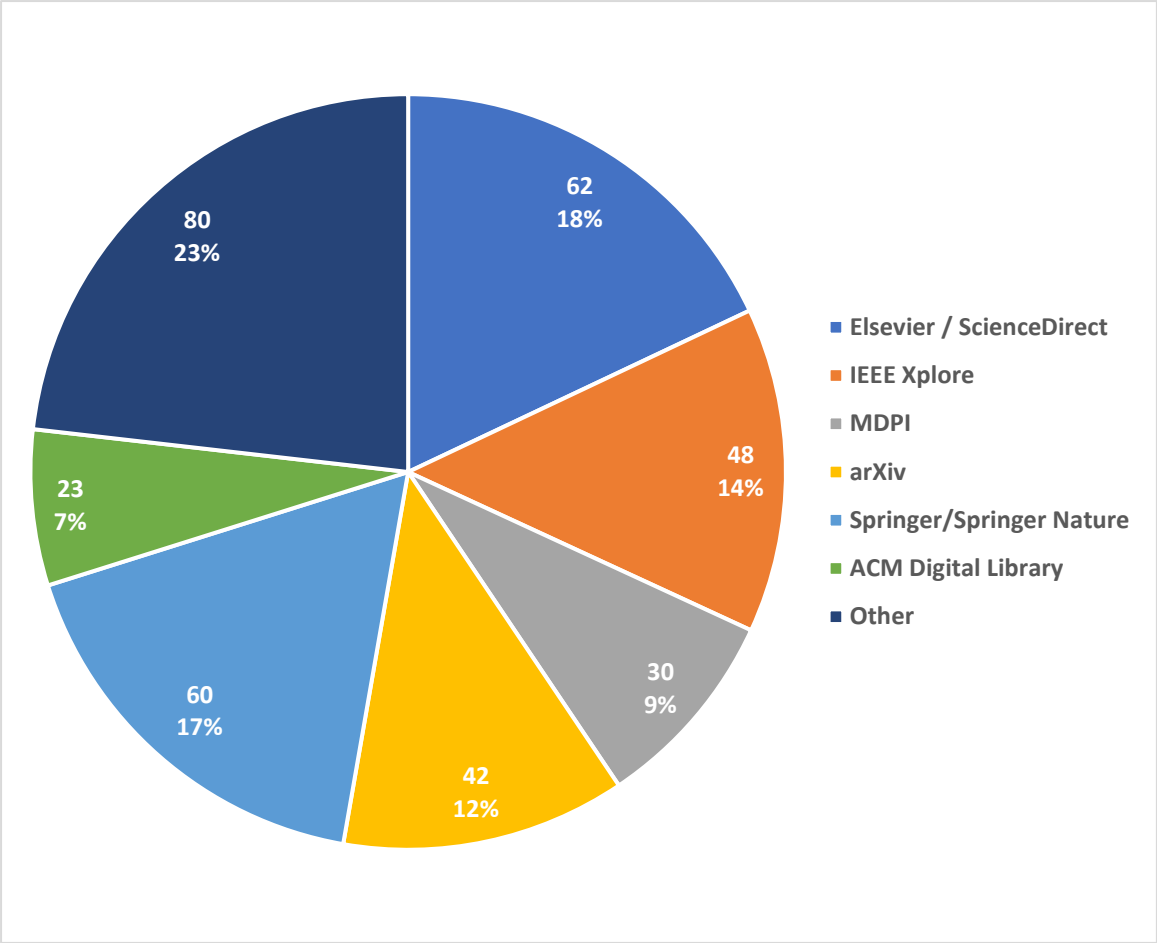


Figure 4. Distribution of reviewed publications by publisher.

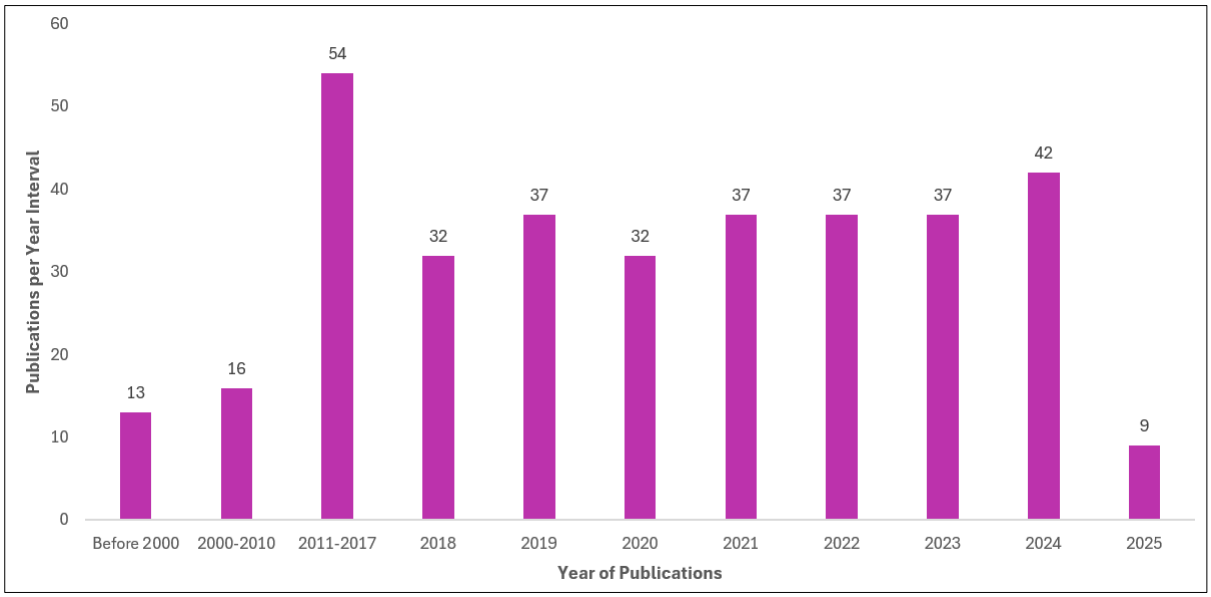


Figure 5. Yearly distribution of reviewed publications from before 2000 through 2025.

The yearly distribution of the reviewed publications (Figure 5) reflects an apparent increase in research output related to ML-based optimization techniques over the past decade. While only 13

publications were recorded before 2000 and 16 between 2000 and 2010, this lower count likely reflects the study's emphasis on more recent optimization approaches — such as AutoML, NAS, and data-centric learning — rather than a true lack of earlier research. From 2011 to 2017, the number of included publications increased to 54, and between 2018 and 2024, annual counts remained consistently high (ranging from 32 to 42 based on our literature review). This trend underscores the increasing attention that has been given to optimization methodologies within modern ML. In 2025, only 9 publications were found, as the year was still ongoing when the literature review was completed, which roughly coincided with the conclusion of this thesis.

The rest of this chapter explores advanced optimization methods in ML pipelines, focusing on three key components: CNNs, hyperparameters, and datasets. The chapter concludes by mentioning other optimization strategies, such as AutoML and RL, which contribute to automated ML pipelines.

2.2 ANN's Structural Optimization

NAS aims to discover the right model for a given ML problem, often through a large variety of candidate models and then automatically selecting the best model, which can be broken into three phases: search space, search strategy, and performance estimation strategy. The NAS has led to some remarkable results, outperforming the state-of-the-art of manually tuned models across a wide range of model sizes.

Within the scope of the architecture search space, some works have been investigated.

In [98], they proposed a system that uses a Recurrent Neural Network (RNN)-based controller, which proposes network architectures. The models are trained from scratch until convergence. If a trained model performs well, then the controller gets a reward; otherwise, a penalization is imposed. The controller update is carried out through this reinforcement learning mechanism. As this process repeats, the controller learns to minimize the expected validation error and starts proposing increasingly better architectures in the next iterations. In Google Brain's NASNet [92], there is a smaller search space compared to the original paper. The authors narrowed down the search space by hand-designing the overall macro architecture and searching for optimal blocks. More specifically, they search for reduction cells that downscale their inputs and normal cells that return a feature map of the same dimension as their input. Once these cell architectures are learned, they can be transferred to other models as well. Another work, named progressive architecture search [112] , it is based on a Sequential Model-based Optimization strategy (SMBO) [113] and uses the surrogate

model; their system required 5x fewer network evaluations as compared to NASNet [92]. Another paper from Google Brain proposes AmoebaNet [114] , an EA that does not consider reinforcement learning to search for architectures. An EA simply mutates existing architectures and kills the ones that don't perform well. Over time, better and better model architectures evolve. AmoebaNet uses the same search space as NASNet. It is very computationally expensive and uses a lot of Tensor Processing Units. A more efficient solution was developed by [115], named Efficient Neural Architecture Search (ENAS), which is an approximation of NASNet. It produces almost as good results as NASNet on a single GPU in less than a day. The speed-up comes from sharing the parameters between proposed networks instead of trashing them every single time and training from scratch. This is done by sampling subnetworks from a large parent network, where they force all subnetworks to share weights. Once they find the best-performing model, they retrain that final model architecture from scratch. Cai et al. [116] proposed an approach based on an RL-based controller to explore a search space. However, they use function-preserving transformations in their search to transfer as much as possible from previously trained models. They modified architectures in a way that uses different parametrizations to represent the same underlying function (i.e., a new layer can be added to a block in a way that doesn't change the output).

The additional layer would increase the representational power of the network and create more room for exploration without training the rest of the network from scratch. This idea originated from network morphism [117]. It is similar to a greedy layer-by-layer search approach. Instead of modifying and training only the last layer, network morphism allows modifying any layer without breaking the following ones. Another method was presented by Elsken et al. [118], named the Lamarckian Evolutionary algorithm for Multi-Objective Neural Architecture Design (LEMONADE), in which an EA for architecture search is employed, along with a morphism to modify networks. Among the networks that were sampled using morphisms, the good ones are kept, others are discarded, and the process repeats. Unlike prior work, LEMONADE tries to optimize architecture for multiple objectives, such as predictive performance and the number of parameters, at the same time. In the context of architecture search, learned weights are reused in the subsequent iterations. LEMONADE's performance is comparable to other methods in terms of computational needs. However, it is much more computationally efficient than NASNet, but still expensive when compared to other efficient methods [116], [119]. Auto-Keras, proposed by [120], also uses network morphism, wherein the search space is explored by morphing network architecture guided by a BO algorithm. In [121], the authors proposed convolutional neural fabrics, which consist of a grid that connects nodes at different layers,

scales, and channels, with a sparse local connectivity pattern. A convolutional neural fabric is essentially a macro architecture with many parallel paths on which the architecture search is performed. Theoretically, an infinitely large fabric would be a universal architecture that can implement any ANN architecture. Unlike the prior work, which used either an EA or RL to search for architectures, Differentiable ARchiTecture Search (DARTS) [119] proposed a different approach and formulated architecture search as a differentiable, continuous optimization problem. To make the search space continuous, they relax the categorical choice of a particular operation, as well as softmax over all possible operations. Once the search is over, the final architecture is discretized by choosing the operations with the highest softmax values. Their approach is reminiscent of sub-network and fabric-based search algorithms since all possible operations are present during training. The employed search space is similar to NASNet, where they search for cell architectures. Neural architecture optimization [122] also adopts a differentiable approach, wherein searches for network architectures in a continuous embedding space are performed. More specifically, this is a framework that consists of an encoder, decoder, and performance predictor. The encoder embeds topologies into a continuous network-embedding space. The performance predictor inputs a network embedding and predicts its performance. The decoder maps a network embedding back into a discrete topology representation. These three components are jointly trained. In essence, the model tries to learn what kind of topologies work better for optimizing a performance predictor in the embedding space. In [123], they proposed an approach of training a hypernetwork to predict the weights of a network in one shot instead of training all the candidates' networks from scratch. First, they train a HyperNet [124] to predict the weights of a given arbitrary network, and then, many architectures are randomly generated, from which weights are obtained using HyperNet. Finally, they pick the architecture that performs the best with those weights and train it from scratch. MetaQNN [125] is a designed CNN algorithm based on RL in which agents are trained to sequentially choose the CNN layers using Q-learning as a search strategy. They modeled the process of layer selection as a Markov Decision Process [126]. Table 1 provides a summary of the ANN´s structural optimizations applied to the CNN architecture, focusing on training performance.

Table 1. Summary of ANN´s structural optimizations.

Ref	Method	Technique	Objective
[98]	NAS	RNN	Controller for the reward/penalization of good/bad training.
[92]	NASNET	RL+ PPO	Searches for reduction and normal cells, which downscale inputs and return the feature map, respectively.

Ref	Method	Technique	Objective
[112]	Proposed	SMBO	Optimizes network architecture search sequentially.
[114]	AmoebaNet	EA	Mutates existing architectures and kills the ones that don't perform well.
[115]	ENAS	Gradient optimization	Shares the parameters between subnetworks.
[116]	EAS	Reinforcement learning	Controller for growing the network layer by searching with function-preserving transformations.
[118]	LEMONADE	EA+ Network morphism	Optimizes multi-objective architecture search.
[119]	DARTS	Continuous optimization	Relaxes the categorical choice of a particular operation, as well as softmax over all possible operations.
[120]	AutoKeras	Network morphism + BOA	Optimizes network architecture search.
[121]	Proposed	Convolutional neural fabrics	Connects nodes at different layers, scales, and channels, with a sparse local connectivity pattern.
[122]	NAONet	Continuous optimization	Searches for network architectures in a continuous embedding space using the encoder, decoder, and performance predictor.
[123]	SMASH	HyperNet	Predicts the weights of a network in one shot and uses them in the other generated networks.
[125]	MetaQNN	Reinforcement learning	Trains agents to sequentially choose the CCN layers using Q-learning.

2.3 HYPERPARAMETER OPTIMIZATION

Hyperparameters are a relevant part of an ANN's configuration, demanding careful handling that must be adjusted from challenge to challenge. Therefore, defining their values with strategy, either through manual means or resorting to techniques for semi- or fully automatic selection, is advisable. Precise selection of them will have a considerable effect on the performance of the model, leading to appropriate results. The hyperparameter optimization process involves the search space of possible values, approaching the highest efficiency of the model [127], [128].

The most important algorithms for hyperparameter optimization are RS [129], the BO algorithm [130], GS [131], and the Genetic Algorithm (GA) [132], among others, the last two being knowledge-based methods.

In [133], the authors have compared RS, Hyperband, BO, and the GA to optimize the hyperparameters for designing an AI-based roll angle estimator. They concluded that knowledge-based methods result in better performance, and among them, the GA outperforms the others. H. Alibrahim et al. [134] employed GS, BO algorithm, and GA to classify whether a customer will do the transaction

or not. In their experiments, the GA had better performance than others. Also [135], optimized the hyperparameter using the GA to predict the business process monitoring. In fact, GS is one of the popular search algorithms, and it has been an applicable algorithm to learn the hyperparameter space setting in various ML-based solutions for different fields, such as the prediction of HIV/AIDS [136], classification of microarray cancer data [127], heart disease prediction [137], prediction of the electricity generation [64], distsributed denial of service (DDoS) attacks detector [138], lifelong learning prediction [139]. The following Table presents a summary of the hyperparameter optimization techniques utilized to enhance model performance and training efficiency. The table outlines the different methods applied—such as RS and BO—mentioning their objective.

Table 2. Summary of hyperparameter optimization techniques

Ref	Technique	Objective
[131], [134] ,[136],[64],[138],[139]	GS	Exhaustively explores all possible hyperparameter combinations to identify the best set.
[129], [133]	RS	Randomly sample from the hyperparameter space to find effective configurations efficiently.
[130], [133], [134]	BO	Models the performance landscape to intelligently explore promising areas of the search space.
[133], [134], [132], [135]	GA	Uses evolutionary strategies to evolve optimal hyperparameter settings over generations.
[133]	Hyperband	Allocates computational resources dynamically to optimize configuration search efficiency.

Another key step in ML is data preparation, where creating appropriately sized and structured datasets is crucial for optimization. This involves techniques like dataset splitting, data augmentation, and generation. Recently, automating these processes has gained attention, and the next subsection discusses the latest advancements in this área.

2.4 DATASET-BASED OPTIMIZATION

Optimizing the dataset is a critical component within the broader ML workflow, given that the quality, diversity, and structural composition of the data significantly affect a model's ability to learn effectively and generalize to unseen scenarios [140], [141]. High-quality data not only enhances model accuracy but also mitigates issues such as overfitting, bias, and poor convergence. In this context, dataset-based optimization [142], [143], [144] encompasses a range of strategies aimed at refining the training data — either through interventions applied before the training phase or adaptively during

the training process itself. These strategies involve augmenting or generating data to ensure it better represents the problem space and challenges in the task.

This subsection explores two primary directions: pre-training approaches and in-training enhancements. Pre-training approaches involve techniques such as strategic data splitting and data selection, where subsets of data are leveraged to incorporate before model training begins. On the other hand, in-training approaches focus on real-time data optimization mechanisms that occur during the training process. This includes techniques such as data augmentation pipelines and GANs, which can synthesize new samples or adaptively enrich training data. These methods are particularly useful when the available data is insufficient or imbalanced, as they provide dynamic support to the model by continuously improving the quality and variability of the training input.

2.4.1 Dataset Configuration Approaches

Pre-training approaches focus on optimizing the dataset before the actual model training begins, laying a strong foundation for learning by ensuring that the initial data configuration is both effective and representative. One of the most fundamental yet impactful steps in this phase is data splitting. The way a dataset is split — typically into training, validation, and test subsets — can significantly affect model performance, generalizability, and reliability [145]. Strategic data splitting helps mitigate [146], [147] issues such as data leakage, class imbalance, and sampling bias, which, if unaddressed, can lead to misleading results or poor real-world applicability. Highlighting the importance of appropriate data splitting, beyond the traditional random splitting method (e.g., [148]), various statistical sampling techniques have also been proposed for more effective data splitting [149]. Fox et al. [150] used both the KFCV and normal split techniques to divide the heITEC LapGyn4 Gynecologic Laparoscopy Image Dataset [151] and employed them to classify the images using a CNN model and Scale-Invariant Feature Transform (SIFT) classification. Their results show better performance for the KFCV using CNN. A feature-weighted sampling method was proposed in [152] to optimize the split of the dataset. In this work, firstly, a division of the full dataset into several subsets using the modified R-value-Based Sampling (RBS) [153] was carried out. Then, the distances of each set to the whole dataset were computed to obtain a similarity score, and the one with the smallest distance was chosen. Eliane [154] employed different methods to split the dataset into training/test sets, including KFCV, Sample Partitioning based on joint X-Y distance (SPXY), Kennard-Stone (KS), and a random sampling algorithm. The use of Pearson Correlation Scores (PCSs) for comparison allowed them to rank these approaches from best to worst performance, as follows: KFCV, SPXY, and KS. An algorithm for

performing stratified KFCV with a focus on similarity-based sample splitting was developed by Farias et al. [155]. First, they select a pivot sample from each label group that is most similar to other samples within the same label group. Afterward, the algorithm identifies additional samples that are most similar to the "pivot" sample. They employed different similarity functions, including Cityblock, Chebyshev, Euclidean, cosine, and correlation, deploying different classifiers: K-Nearest Neighbors (KNN), Random Forest (RF) classifier with 100 trees; Support Vector Machine (SVM), and MultiLayer Perceptron (MLP). Their results showed the better performance of the proposed algorithm over the normal KFCV, using the correlation similarity function. It is worth mentioning that the difference between the accuracies of different similarity functions was almost 1%. Nurhopipah et al. [156] compared four dataset splitting techniques for face classification tasks: random subsampling validation, bootstrap validation, KFCV, and Moralis Lima Martin Validation (MLMV). They concluded that KFCV provides a more stable performance with higher accuracy. The KFCV method was, once again, employed in [157] to partition brain image samples for tumor diagnosis, using the SVM algorithm. In this work, different kernels of the SVM were used. In [158], KFCV, with K=10, was applied to split a vegetation dataset into training and validation subsets and, subsequently, build different classifiers – KNN, Gaussian Naive Bayes, RFs, SVM, and MLP – capable of performing vegetation physiognomic classification. Inside the cross-validation loop, they performed a univariate statistical test (ANOVA F-value) between physiognomic classes and the features in the training set to select the best-scoring features from the training set. A higher accuracy (81%) was verified for RFs.

Varma and Simon [159] used both normal and nested KFCV and showed that the nested one can be considered a nearly unbiased method. Vabalas et al. [160] used the train/test split, normal, nested, and partially nested KFCV to split the dataset, and used an SVM and a logistic regression classifier to distinguish autistic from non-autistic individuals. In their experiments, the train/test split and nested KFCV approaches produced robust performance. Kahloot et al. [161] developed a technique named “Algorithmic splitting”. They used, firstly, the Umap technique for dimension reduction and then two different clustering methods, including Hierarchical Density-based Spatial Clustering of Applications with Noise (HDBSCAN) [162] and Gaussian Mixture Models (GMM) [163]. The distances of data points to the mean were computed using Chebyshev’s inequality bound [164]. For each cluster, they calculate the mean (μ) and standard deviation (σ) of the data points' positions. Then, the algorithm defines different ranges based on μ and σ, which include a "median range" (within 1 standard deviation from μ), an "extreme range" (beyond 2 standard deviations from μ), and a "quantile range" (between median and extreme ranges). For each sub-dataset size (training,

validation, testing with sizes of 70%, 20%, and 10%, respectively), the algorithm samples data points from the defined ranges according to specified percentages and experiments with different rates of selection for each range, to train models based on Visual Geometry Group Network (VGG) [165], ResNet [166], and Inception [167]. The optimal percentages were 55%, 35%, and 10% out of the median, extreme, and quantile ranges, respectively. For comparison purposes, they also split the dataset using random sampling. They concluded that the HDBSCAN outperforms the GMM with the algorithmic split.

To split the dataset, Doan et al. [168] First, divided each class into clusters using HDBSCAN and then applied a stratification technique to select data from each cluster for the train/test set. They also used random sampling, stratified sampling, stratified cross-validation, and bootstrapping techniques as data splitting techniques and compared the results with their approach, which proved to outperform the others for all the trained models (MLP, SVM, RF, Extreme Gradient Boosting(XGBoost)).

Resampling, normal, repeated, nested, Leave-One-Out KFCV were used by [169] to split the dataset for training different ML algorithms (linear regression, Bayesian ridge regression, ridge regression, LASSO regression, KNN, CART decision trees, SVM regression – SVMR, XGBoost, gradient boosting, RFs, and extra trees), to predict profits in Olive farms. Their results presented a better performance for SVMR than the others, and the resampling technique outperformed the other splitting techniques. In specific contexts, such as agriculture, traditional dataset splitting is the preferred option to be used [170], [171], [172], even in limited dataset conditions, as can be verified in [173], which proposes the use of KFCV to split datasets to feed decision tree, RF, gradient boosting, and SVM models, aiming at tomato crop yield prediction, based on certain inputs (soil properties, applied fertilizers, and weather conditions). Optuna [174] uses HO splitting as part of its hyperparameter optimization process. The same tendency of using traditional dataset splitting can be verified in the Construction field [103].

Table 3 provides a summary of the dataset configuration approaches implemented to ensure effective data utilization and balanced representation across training, validation, and testing subsets, highlighting their key features and performance.

Table 3. Summary of dataset configuration approaches.

Ref	Technique	Objective
[103], [174]	HO	Apply standard split methods in ML tasks.

Ref	Technique	Objective
[145]	KFCV vs. Normal Split	Improve image classification accuracy using different split strategies.
[152]	Feature-weighted RBS	Optimize dataset splitting using similarity-based selection.
[154]	KFCV, SPXY, KS, Random	Compare and rank splitting methods based on performance (PCS).
[155]	Stratified KFCV with similarity functions	Enhance class-wise sample similarity within folds.
[156]	Random, Bootstrap, KFCV, MLMV	Evaluate stability and accuracy for face classification.
[157]	KFCV	Split the brain image dataset for effective tumor classification.
[158]	10-Fold KFCV	Classify vegetation physiognomic types with feature selection.
[159]	Normal vs. Nested KFCV	Assess bias in validation methods.
[160]	Train/Test, Normal, Nested, Partially Nested KFCV	Evaluate robustness in autism classification.
[161]	Algorithmic Splitting (UMAP + HDBSCAN/GMM + Chebyshev)	Create balanced splits using dimensional and statistical techniques.
[168]	Clustered Stratified Sampling	Improve stratification by incorporating clustering (HDBSCAN).
[169]	Resampling, KFCV (Normal, Nested, Repeated, LOOCV)	Predict farm profits and evaluate splitting on regression performance.
[173]	Traditional + KFCV	Predict tomato crop yield using ML with reliable data spliting.

2.4.2 Dataset Enhancement Approaches

While structuring and preparing a dataset provides a crucial foundation for effective learning, dataset enhancement approaches aim to further expand, diversify, and enrich the data to improve its impact during model training. These methods are particularly valuable in addressing challenges such as data scarcity, class imbalance, and limited variability, all of which can negatively affect a model's ability to generalize. One of the most established strategies in this domain is data augmentation, which applies controlled transformations to existing samples. Beyond transformation-based methods, generative models, particularly GANs, have emerged as powerful tools for synthesizing new data samples.

This subsection explores the current state-of-the-art approaches in both augmentation-based and generative dataset enhancement, highlighting their roles, mechanisms, and contributions to improved model performance.

Traditional augmentation-based techniques

In ML-based methodologies, traditional data augmentation serves as a foundational technique during the dataset preparation phase, particularly addressing the challenge posed by limited or imbalanced data [50]. These augmentation techniques consist of sampling over existing data and applying operations for inducing variability — such as image rotation, scaling, flipping, or adding noise. By artificially increasing the diversity and volume of training data, it is expected to improve the model's ability to generalize. However, manual implementation of these augmentations often demands significant time and effort, especially when tailored to specific tasks or domains. As a result, recent research and development efforts have shifted toward automating the augmentation process [175], [176]. These automated approaches aim to enrich datasets more efficiently, reduce human intervention, and ultimately lead to improved model performance by enabling more robust and representative training.

Cubuk et al.[177] presented AutoAugment(AA), an approach aiming at better and automatic data augmentation strategies, resorting to a search space and a search algorithm. In their search space, the policy has 5 sub-policies in a batch that consists of two operations. Each operation has two hyperparameters, which are the probability of applying the operation and the magnitude of the operation. The search algorithm consists of two components: one is a controller RNN, which is a one-layer Long Short-Term Memory (LSTM) [178] and predicts a decision using Softmax; and the other is a training algorithm, which is the Proximal Policy Optimization (PPO) algorithm [179]. Since this approach is very time-consuming, to overcome this problem, Lim et al. [180] proposed an efficient algorithm named Fast AutoAugment (FAA) that employs a more efficient density-matching-based search strategy to find a less computationally burdensome augmentation strategy. They reduced the search time by orders of magnitude, employed Bayesian hyperparameter optimization to search for optimal policies, and resorted to the Tree-structured Parzen Estimator (TPE) algorithm [181] for practical implementation. The density match between two train datasets in the search policy was their objective. Resorting to K-fold stratified shuffling [182], the proposed method splits the dataset into chunks, which comprise two datasets: data for learning the model (DM) and data for exploring the augmentation policy (DA). The model was then trained in parallel on each DM without data augmentation using an ANN model. After training, they used the BO approach [183] to explore candidates for augmentation policies on DA without using the training parameter, sampled a sequence of sub-policies from the search space, and appended the top N policies from each K-fold to an

augmentation policy list. RAY was integrated with these experiments, which is a distributed learning framework to implement their approach easily [184], [185].

Faster AutoAugment (FasterAA) [184] was an approach with a differentiable policy search pipeline that used gradient-based policy searching. For this purpose, they minimized the Wasserstein distance between distributions of the original and augmented images, and employed adversarial learning for better matching the augmented one with the distribution of the original image. Unlike AA, which was based on a fixed policy, Ho et al. [186] proposed a method named Population-Based Augmentation (PBA), which learns a schedule of augmentation policies. They applied Population-Based Training (PBT), an optimization algorithm that optimizes hyperparameters jointly with network parameters.

To develop Adversarial AutoAugment (AdvAA), Zhang et al. [187] employed the basic structure of the search space of AA [177], and to compare with AA, they slightly modified the search space by not considering the probability of each operation, from a set that involves Rotate, Invert, TranslateX/Y, Brightness, Equalize, Sharpness, AutoContrast, Color, Contrast, Solarize, Sample Pairing [188], Posterize, ShearX/Y, and Cutout [189]. In their approach, data augmentation of the batch with sample policies by multiple pre-processing steps is carried out. Then, a target network is trained to minimize the loss of a large batch (formed by multiple augmented instances of the input batch). They extracted the training losses of a target network as the reward signal. Finally, the augmentation policy network is trained with the processed reward signal, maximizing the training loss of the target network while using generative adversarial methods. In prior works [130], [136], [190] scaling learned data augmentation (Containing over 30 parameters) proved to be insufficient to be applied to larger datasets and models. So in [191], they considerably reduced the parameter space for data augmentation by removing the learned policies and probabilities in each transformation and using a parameter-free procedure by selecting a transformation with uniform probability.

Zoph et al.[192] developed a method building upon the earlier work [177], focusing on the use of auto augmentation for object detection. Their training setup closely followed the techniques outlined in [92], [98], [177], [193]. The output space of the RNN was used to structure the discrete optimization problem, and RL [98] was employed to update the weights. TrivialAugment(TA) is another automatic augmentation approach presented by [194], which simplifies the previous work and just uses a single augmentation to be applied to each image. It consists of a parameter-free tool in which an augmentation process can be randomly selected from a set of possible operations and applied to the image with a randomly chosen strength from a set of values. Focused on performance concerns,

LingChen et al. [195] presented an approach that reduced the computational cost and avoided searching for the optimal parameter by uniformly choosing augmentation operations. Negassi et al. [196] presented two automated data augmentation algorithms named SmartAugment (SA) and SmartSamplingAugment (SSA). Rather than optimizing the total number of sampled augmentations, SA optimizes the number of sampled color and geometric augmentations, as well as their magnitude. Moreover, they optimize the probability of hyperparameters for applying data augmentations. BO is used to search for the best augmentation strategy, and then the augmentations are chosen from the augmentation operation list. Unlike RandAugment [191], which allows the same augmentation to be selected multiple times for each image, SA allows each augmentation to be sampled just once for each image. In SSA, two augmentation operations are used, and the magnitude is randomly chosen from a magnitude list. Like RandAugment, positive average improvement was considered and converted into probability (the average performance improvement is computed when a certain augmentation operation is added to a random subset of augmentation). Hendrycks et al. [197] proposed a data processing-based technique for improving the image classifier's robustness and uncertainty estimates. They added limited computational resources to models, which helps ensure their resilience to unknown errors. To augment the data, they mixed some traditional augmentation techniques that outperformed the other augmentation techniques like CutMix [198], Cutout [189], Mixup [199], and AA [177]. Since the size of the search space is a challenge when using deep augmentation layers ($K \geq 5$, where K is the number of transformations), the search space can become unmanageably large. To overcome this issue, Zheng et al. [200] developed a Deep AutoAugment (DeepAA) method and built a full data augmentation by sequentially adding layers of augmentation and optimizing each augmentation layer on top of the data distribution transformed by prior layers. They used regularized gradient matching to formulate the multi-layer data augmentation search so that they maximized the cosine similarity of the gradients between the original data and augmented data with regularization.

Li et al. [201] proposed the method named Differentiable Automatic Data Augmentation (DADA) in which they followed the structure of AA but reformulated the augmentation operation application and the sub-policy selection as sampling from Bernoulli and Categorical distributions, respectively, addressing this topic from a perspective of the joint distribution. As these two distributions are not differentiable, the authors relaxed them through the Gumbel-Softmax gradient estimator [202] to optimize the differentiable approach. Due to the estimator bias, the RELAX estimator [203] the method was considered to tackle the bias of the conventional Gumbel-Softmax estimator. They also optimized

the magnitude by applying the straight-through gradient estimator [204]. Unlike AA, which uses validation accuracy as a reward, DADA minimizes the validation loss for gradient computation. In the end, they presented the joint optimization of ANN and data augmentation by employing Stochastic Gradient Descent (SGD).

Different from DADA, Liu et al. [205] proposed an efficient differentiable search algorithm, named Direct Differentiable Augmentation Search(DDAS), that does not rely on Gumbel-Softmax [202] or second-order gradient approximation [119] and exploits meta-learning with one-step training loss. Their search approach is based on a two-level hierarchy in which, first, they check whether to perform the augmentation and then determine a certain augmentation policy by sampling N operations from the candidate set, applying them consecutively to the input data. For the setting of magnitude, they followed the RandAugment setting. Taking a different direction from the works that conventionally rely on offline search augmentation methods, Lin et al [206] proposed an online hyperparameter learning for auto-augmentation. The challenge was addressed with a parametrized probability distribution that can be optimized jointly with the parameters of the network. Hypernetwork-Based Augmentation (HBA) [207] is a gradient-based algorithm that learns augmentation hyperparameters jointly with the model parameters in a single training. They employed a PBT algorithm using a weight-sharing strategy that helps them reduce memory consumption and time spent searching.

The summary of these data augmentation techniques is presented in Table 4, outlining their key objectives. Furthermore, Table 5 presents a comparative summary of the computation time and performance of various automatic data augmentation approaches across multiple benchmark datasets, including CIFAR-10, CIFAR-100, SVHN, and ImageNet. For each dataset, the table details the GPU specifications used, the total training time measured in GPU hours, and the corresponding error rates achieved by the models. From these indicators, some conclusions can be drawn. More specifically, techniques such as AA require extensive GPU time (up to 15,000 hours on ImageNet) and achieve error rates on CIFAR-10 (e.g., 2.6%) that roughly match the arithmetic mean error of the overall competing approaches. More recent methods like FasterAA and DADA significantly reduce computation time—down to fractions of GPU hours—while maintaining comparable error rates. Some other approaches, like AdvAA and UA, demonstrate improved accuracy with less computation on select datasets.

Table 4. Summary of data augmentation methods and their purposes.

Ref	Method	Technique	Objective
[177]	AA	RNN+ PPO	A controller to predict the decision and a PPO for training.
[180]	FastAA	Density matching-based search	Finds the density match between two training datasets in the search policy.
[184]	FasterAA	Gradient-based search + Adversarial learning	Minimizes the Wasserstein distance between distributions of the original and augmented images and finds better matches.
[186]	PBA	PBT	Optimizes hyperparameters and network parameters.
[187]	AdvAA	Adversarial learning	Extracts the training losses of a target network as the reward signal and trains the network to maximize the training loss.
[191]	RA	RS	It is parameter-free and randomly chooses an augmentation method from a set of augmentations. The same augmentation can be sampled several times per image.
[194]	TA	RS	It is parameter-free and randomly chooses an augmentation method from a set of augmentations.
[195]	UA	Uniform Search	Uniformly chooses the augmentation operations to find optimal parameters.
[196]	SSA	BO	Chooses the best augmentation strategy. Each strategy can be sampled once per image.
[197]	AugMix	Mixing	Combines different augmentation operations using elementwise convex combinations.
[200]	DeepAA	Multi-layer search	Sequentially adds layers of augmentation and optimizes each augmentation layer on top of the data distribution transformed by prior layers.
[201]	DADA	Gumbel-Softmax Gradient Estimator	Relaxes the distributions and optimizes them.
[205]	DDAS	Meta-learning	First, define whether to apply the augmentation and then sample the operations.
[206]	OHLAA	Parametric probability distributions	Optimizes a policy as a parametrized probability distribution jointly with the parameters of the network.
[207]	HBA	Gradient-based algorithm + PBT	Jointly learns augmentation parameters and model parameters. PBT uses the weight-sharing strategy to reduce memory usage and processing time.

Table 5. Summary report on computation time and performance of the automatic augmentation approaches. * The dataset used was CIFAR-100-C.

Ref	Method	GPU-Spec	CIFAR-10		CIFAR-100		SVHN		ImageNet	
			GPU (Hours)	Errors rate (%)	GPU (Hours)	Errors rate (%)	GPU (Hours)	Errors rate (%)	GPU (Hours)	Errors rate (%)
[177]	AA	Tesla P100	5000	2.60	NA	17.10	1000	1.10	15000	22.40
[180]	FastAA	Tesla V100	3.50	2.70	NA	17.30	1.50	1.10	450	22.40
[184]	FasterAA	V100	0.23	2.60	NA	17.80	0.061	1.20	2.30	23.50
[186]	PBA	Titan XP	5	2.60	NA	16.70	10	1.20	NA	NA
[187]	AdvAA	Tesla V100s	NA	1.90	NA	15.50	NA	NA	1280	20.60
[191]	RA	NA	NA	2.70	NA	16.70	NA	1.30	NA	22.20
[194]	TA	RTX 2080 Ti	NA	2.54	2	15.67	NA	1.10	NA	21.93
[195]	UA	NA	NA	1.90	NA	17.20	NA	NA	NA	22.30
[197]	AugMix	NA	NA	4.90	NA	35.90*	NA	NA	NA	25.20
[200]	DeepAA	NA	9	2.44	9	15.98	NA	NA	96	21.70
[201]	DADA	Titan XP	0.10	2.70	0.20	17.50	0.10	1.20	1.30	22.50
[205]	DDAS	RTX 2080Ti	0.15	2.70	0.15	16.60	NA	NA	1.20	22.30
[206]	OHLAA	NA	83.40	2.60	NA	NA	NA	NA	625	21.10
[207]	HBA	Tesla V100	0.10	2.60	NA	NA	0.10	1.10	0.90	22.10

Generative dataset enhancement approaches

Classical data augmentation methods, such as rotations, flips, scaling, and color adjustments, are well-established techniques used to artificially increase the variability in training datasets [47]. These methods have proven effective in enhancing model generalization but are typically applied uniformly across the dataset, failing to specifically address class imbalances. Advanced augmentation strategies have emerged to overcome this limitation, focusing on more sophisticated techniques like automated data augmentation and GANs [50], [208].

GAN [209] is a method for creating artificially proposed synthetic data, operating through a computationally intensive adversarial framework involving two neural networks: a generator and a discriminator. The generator produces synthetic data samples tailored to a specific context, while the discriminator evaluates these samples by estimating the likelihood that they resemble real data. The interaction between the two networks forms a dynamic learning process in which the generator progressively improves its ability to produce realistic and high-quality outputs based on the discriminator's feedback. It is applicable for numerous practical applications in various fields, e.g., signal processing [210], language processing [211], video generation [212], and image generation

[213], [214], [215], [216]. The images can be generated from low-resolution images [217], [218], [219], [220], image-to-image translation [221], [222], [223], [224], [225], and text [226], [227], [228]. GANs, generating images from an imbalanced dataset, are challenging, as Mariani et al.[229] overcome this issue by including all data from the majority and minority classes' images at once so that the generator can learn useful features from the majority classes and employ them for generating images for the minority classes.

Although the GAN-based data augmentation has already enriched the datasets, some investigations have been conducted to improve the quality of GAN results. Gong et al. combined GAN with NAS and proposed AutoGAN [230]. They defined a search space for a generator and employed an RNN to control it with parameter sharing, then used the Inception score [231] as a reward. Furthermore, they introduced a multi-level search strategy to the NAS. GANs constitute a suitable alternative for data augmentation, by allowing to generation of synthetic examples that closely resemble the original dataset. GANs have been particularly effective in augmenting minority classes, addressing class imbalances more directly. Techniques such as [232], [233] generate synthetic samples for underrepresented classes, enriching the training set and improving model performance in these classes.

Building on these advancements, a range of GAN variants have been developed further to improve the quality and diversity of generated data. Conditional Generative Adversarial Networks (cGAN) [234] introduce the class-level conditioning, enabling more class-specific data synthesis [235], [236], [237], [238]— especially applicable for augmenting specific underrepresented classes. In the study done by S.Mohammed et al. [239], a cGAN is employed for fabric defect data augmentation, designed as an image-to-image translation model. The proposed architecture features a conditional U-Net generator and a six-layer PatchGAN discriminator [240]. Taking a segmented defect mask and a fabric image as inputs, highly quality synthetic defect samples are generated. The Robust Conditional GAN (RoCGAN) technique was proposed by [241] to improve the quality of image-to-image translation tasks. It incorporates a novel loss function (latent loss) and architectural enhancement employing a dual-path generator composed of a supervised conditional generator and an unsupervised autoencoder instead of the original cGAN single path, and the discriminator remains the same as the cGAN. Their proposed approach outperformed the existing state of the art of cGAN, especially in situations involving data with intense noise. Expanding on the architectural limitations of early GANs, Deep Convolutional Generative Adversarial Networks (DCGAN) [242] introduced convolutional layers techniques — Batch Normalization (BatchNorm) and Leaky Rectified Linear Unit (leakyReLU) — improving training

reliability. Although this technique was originally designed for unsupervised learning, several advanced methods have been developed to adapt it for semi-supervised applications. This adaptation often involves incorporating label information or modifying the discriminator to support classification tasks. For example, in the study [243], wherein a DGAN-based architecture was employed, resorting to training over retinal images. More specifically, by using a small labeled dataset alongside a large number of unlabeled images, the system was able to generate realistic images and classify them with high accuracy. Another work [244] used the Conditional DCGAN (cDCGAN) for brain tumor classification, where synthetic and traditionally augmented images were used to train a custom-designed DCNN. However, cDCGAN was still limited in capturing global image structure [245], which led to the development of Self-Attention Generative Adversarial Networks (SAGAN), incorporating self-attention mechanisms to model long-range dependencies and improve visual coherence. To further enhance generation quality and control, StyleGAN [246], [247], [248] introduced a disentangled latent space and stochastic per-pixel variation, enabling fine-grained manipulation of image features and producing highly realistic outputs.

Table 6 summarizes the techniques mentioned above, highlighting their respective objectives. These methods generate realistic synthetic data across various domains and modalities, including images, videos, and signals. Key advances include class-conditioned generation, balancing minority and majority classes, and integrating neural architecture searching processes for optimized generators. Innovations like SAGAN and StyleGAN improve image quality and control.

Table 6. Summary of generative dataset enhancement approaches.

Ref	Technique	Objective
[50], [208]	Automated Data Augmentation	Uses algorithms to determine optimal augmentation policies.
[209]	GAN	Uses a generator and a discriminator to produce synthetic data.
[210], [211], [212], [213], [214], [215], [216]	Signal, Language, Video, Image Generation	GANs are applied in various fields.
[217], [218], [219], [220], [221], [222], [223], [224], [225], [226], [227], [228]	Low- to High-resolution, Image-to-Image, Text-to-Image	Different input modalities for image generation.
[229]	Balanced GAN Training	Include all data, both the majority and minority class data, for training.
[230], [231]	AutoGAN (GAN + NAS)	Combines GAN with NAS to optimize generator design.

Ref	Technique	Objective
[232], [233]	Minority Class Augmentation	GAN-based techniques focused on generating data for underrepresented classes.
[234] , [235], [236], [237], [238], [239], [240]	cGAN	Introduces class-level conditioning for targeted data synthesis.
[241]	RoCGAN	Enhanced image-to-image translation with novel latent loss and dual-path generator.
[242]	DCGAN	Introduced convolutional layers, batch normalization, and leaky ReLU to improve training stability.
[243]	SS-DCGAN	Adapted in semi-supervised learning for retinal image synthesis and classification; enables realistic images and high accuracy.
[244]	cDCGAN	Extension of DCGAN with conditional input for tasks like brain tumor classification.
[245]	SAGAN	Incorporates self-attention to capture long-range dependencies and improve the visual coherence of generated images. It overcomes the global structure limitations of prior GANs like cDCGAN.
[246], [247], [248]	StyleGAN	Uses disentangled latent space and per-pixel stochastic variation for fine-grained control and highly realistic outputs.

2.5 Other Relevant Optimization Approaches

In addition to the core optimization strategies that target model architecture, hyperparameters, and dataset management, several other approaches contribute significantly to enhancing the performance of ML systems. These methods may not fall strictly within the traditional categories but are nonetheless crucial in automating and refining the ML pipeline. One of the most transformative areas in recent years is AutoML, which automates various aspects of the ML lifecycle, including model selection, feature engineering, hyperparameter tuning, and dataset preparation. AutoML frameworks have made ML more accessible and efficient by reducing the need for expert intervention, thereby democratizing the process and enabling non-experts to build high-performing models. Notable frameworks such as Google's AutoML [249], Auto-sklearn [250], and H2O AutoML [107] have enabled researchers and practitioners alike to achieve competitive performance without deep technical backgrounds. AutoML leverages sophisticated search strategies such as BO, GA, and RL to explore vast configuration spaces. These techniques not only optimize model performance but also contribute significantly to model robustness and reproducibility by minimizing human bias in design choices.

In recent years, AutoML has advanced beyond simple pipeline automation to include more adaptive and intelligent mechanisms. One of the most notable developments is NAS [251], [252], where the architecture of DL models is automatically designed rather than manually engineered. NAS uses

methods such as RL, evolutionary strategies, and differentiable architecture search to discover novel neural network structures, many of which have achieved state-of-the-art results in image classification, NLP, and speech recognition. In [253], they review its key components, common techniques, current systems, and future research challenges, offering a useful guide for further development in the field. F.Hutter et al. [254], in their book, they explore how to automate ML tasks to make them easier and more accessible. It covers key techniques like hyperparameter tuning, neural architecture searching, and meta-learning. It also reviews popular AutoML tools such as Auto-sklearn and Tree-Based Pipeline Optimization Tool (TPOT) [255] and discusses real-world challenges and future research directions.

Additionally, RL techniques are being explored to optimize the learning process [256]. RL, traditionally known for its success in sequential decision-making problems, has recently become a powerful paradigm for solving complex optimization tasks [257], [258]. RL formulates the optimization process as a dynamic interaction between an agent and its environment [259], [260]. The agent learns an optimal policy through trial and error, receiving feedback as rewards or costs, which makes RL particularly well-suited for non-convex, high-dimensional, or combinatorial optimization landscapes. Recent advancements in RL have extended its applicability from simulated control problems to complex real-world domains such as robotics [31], finance [261], autonomous driving [262], game playing [34], healthcare [263], agriculture [264], and construction engineering [265].

Modern RL methods have been combined with and enhanced by DL, giving rise to the Deep Reinforcement Learning (DRL) designation [266]. DRL techniques are broadly categorized into value-based, policy-based, and actor-critic methods [267]. Value-based methods [268], such as Deep Q-Networks (DQN) [269], [270], approximate the expected return of actions and have been foundational in achieving human-level performance in games like Atari. Policy-based methods [271], including REINFORCE [272] and PPO [273], [274], directly model the policy distribution and are more suited to continuous action spaces. Actor-critic architectures [275], which combine the strengths of both approaches, are now the backbone of many state-of-the-art algorithms due to their improved sample efficiency and stability. Among all DL advancements, DRL has achieved notable effectiveness in a variety of domains [276], [277].

In ML-based approaches, the improvement of model performance during the training process is one of the challenging steps. This improvement not only helps to increase the efficiency of the model but also provides better compatibility with diverse data and settings. Focusing on effective data

organization and developed techniques, the models can improve their accuracy increasingly and adapt to new conditions. In the rapidly evolving field of ML, automated solutions have become essential for handling complex tasks efficiently. The effectiveness of DL models, particularly in computer vision, relies heavily on the quality and quantity of the training data. However, in real-world scenarios, datasets are often limited in size and representational contexts and suffer from class imbalances, where certain classes are significantly underrepresented. This imbalance can lead to biased models that perform poorly in minority classes [278], limiting their generalizability and robustness. To address these issues, data augmentation techniques have become a principle in training pipelines, enhancing the diversity of training data and improving model performance [279], [280].

Automated data augmentation techniques, such as [177] , have been significantly improved by automatically searching for optimal augmentation policies tailored to a specific dataset. These methods, however, remain static in their application, applying the same augmentation strategy throughout the training process.

In addition to these advancements, closed-loop strategies have emerged as a dynamic approach to data augmentation during the training process. These strategies involve continuously monitoring the model's performance and adjusting the augmentation techniques in response to the model's needs. Such as RL-based augmentation, which adapts augmentation policies based on model feedback, ensuring that the augmentation process evolves alongside the model's training [177], [184], [191].

However, despite the works on this theme, challenges remain, particularly in terms of low sample efficiency and limited generalization ability, which to address these issues, researchers are increasingly exploring data augmentation [281], [282], [283], [284], [285], [286] for RL. Grigorescu et al. [287] proposed an approach using feedback control in image processing to enhance object recognition robustness, particularly by improving the quality of binary segmented images under varying illumination. Their closed-loop control ensures reliable input for feature extraction and recognition, improving system reliability, which showcases its benefits for service robotics. Lu et al. [288] presented a system for real-time defect detection and closed-loop adjustment in robot-based additive manufacturing of carbon fiber-reinforced polymer composites. By adjusting the process parameters, including layer thickness, feed rate of filament, printing speed, and printing temperature, the approach outperforms conventional methods by enabling online defect control through process parameter adjustments.

The next table represents the summary of reviewed advanced optimization techniques, including AutoML frameworks (Google AutoML, Auto-sklearn, H2O AutoML) that automate model selection and tuning, and NAS methods that discover optimal DL architectures. RL and Deep RL are highlighted for their applications across diverse domains and in optimizing training processes.

Table 7. Overview of additional miscellaneous optimization approaches.

Ref	Technique	Objective
[249]	Google AutoML	Automation of model selection and tuning.
[250]	Auto-sklearn	Scikit-learn-based AutoML with meta-learning and BO.
[107]	H2O AutoML	Automate the end-to-end ML process to build high-performing, scalable, and interpretable models with minimal user intervention.
[251], [252], [253]	NAS	Automatically discovers DL architectures using RL.
[254]	Auto-sklearn and TPOT	Covers hyperparameter tuning, NAS, and AutoML tools like Auto-sklearn and TPOT.
[256],[257], [258],[259], [260]	RL	Applies RL to model training, viewing it as agent-environment interaction.
[31],[34], [261], [262], [263], [264], [265]	RL	Application of RL in different applications: robotics, games, finance, autonomous driving, healthcare, agriculture, and construction.
[266]	DRL	Uses DL networks to approximate complex policies or value functions in environments.
[268], [269], [270]	Value-based RL	Learns value functions to guide actions.
[271], [272], [273], [274]	Policy-based RL (e.g., REINFORCE, PPO)	Directly models and optimizes policies.
[275]	Actor-critic methods	Combines value and policy learning for better stability and sample efficiency.
[276]	DRL across domains	Solving a Rubik's Cube.
[177], [184], [191]	RL-Augment (Closed-loop)	Adjusts augmentation based on model feedback.
[287]	Closed-loop Control	Feedback-based segmentation enhances the image quality under the noise or lightning variation.
[288]	Closed-loop Control	Real-time control in 3D printing by real-time adjustment of layer settings to reduce manufacturing defects.

This chapter has explored various techniques to optimize several aspects of ML-based approaches. These methods cover architectural design, hyperparameter tuning, and data augmentation strategies. In particular, the integration of AutoML frameworks, NAS, and RL-based optimization methods demonstrates the field's shift toward greater automation and adaptability. To facilitate a clearer understanding, the relevant summary tables are presented in Tables 1-7. Moreover, the approaches mentioned above focus on the model's overall performance but do not adequately address performance at a class-specific level. This can mask underlying issues where the model performs well

on average but fails to generalize across all classes. Therefore, this study intentionally introduces imbalances in the dataset to identify and fix these biases, especially those caused by underrepresented classes that may be overlooked when focusing only on overall performance. To tackle the mentioned issues, in this work, two approaches will be presented: 1) proposing two feature-aware-based dataset splitting techniques, and 2) a closed-loop metric-based data augmentation strategy is proposed that integrates data augmentation techniques, including Classical Augmentation (CA) and GAN-based synthetic data, with a dynamic feedback system during the training process. This approach iteratively refines the augmentation techniques applied to underrepresented classes based on the model's performance at each training iteration. By intentionally unbalancing a given dataset, it is aimed to achieve balanced model performance across classes, leading to an overall improvement of metrics.

3 Overview of the Proposed DL-oriented Optimizations

Acquiring, labeling, and organizing data are among the most burdensome yet critical aspects of applying DL, more specifically, to address supervised problems. The process is not only time-consuming but also filled with intricacies. Identifying a reliable data source and ensuring diversity in the dataset for representativeness purposes, while simultaneously maintaining relevance to the intended task, are just the initial barriers. Once the data has been gathered, two critical steps follow: labeling and cleaning [289]. Accurate labeling ensures that each data instance is correctly annotated with the information the model needs to learn, aligning with the specific problem domain and ensuring that DL models can accurately interpret the data. Thus, proper labeling requires not only domain expertise but also a robust strategy to ensure consistency across the dataset [290]. Complementarily, the data cleaning process removes inconsistencies, redundancies, and noise that could negatively affect model performance [291], [292].

After the dataset has been prepared, it must then be divided into subsets for training, validation, and testing. This stage, despite being straightforward, plays a pivotal role in determining the final performance of the model. Both the ratio of splitting [293], [294] and the techniques [154], [295] employed for splitting the data can significantly influence the outcomes. An imbalanced split might lead to inadequate model generalization [296], while poorly implemented splitting methods could result in data leakage or biased performance metrics [297].

As highlighted earlier in this thesis, besides the importance of properly organizing datasets, most of the commonly used methods dedicated to that task do not inspect features for a smarter and more expeditious division. Instead, they rely on either random splitting (e.g., HO) and/or burdensome and time-consuming brute-force-based sorting approaches (e.g., KFCV). Another noteworthy issue concerns the impact of representativeness unevenness as another factor unbalancing datasets. The lower occurrence of certain conditions in the dataset may lead to models that perform well on statistically dominant features but poorly on underrepresented ones.

Considering the identified issues, this chapter provides an overview of proposed methods to address them. It begins by contextualizing the raw data used for experimental purposes, explaining the rationale behind its selection, and describing the processes undertaken to make it suitable for use. Afterwards, the main workflow adopted to address the objectives of this research and tackle the previously outlined challenges is presented. In the end, the experimental setup for this work is detailed.

3.1 Contextual Datasets

In this study, two distinct image datasets have been carefully gathered and constructed, each designed to represent different contextual domains and offer unique perspectives for analysis. The first dataset

focuses on a nature-based context, specifically involving images of leaves. This dataset captures the rich diversity of natural features such as shape, texture, color, and patterns inherent to various types of grapevine leaves. The collection process for this dataset involved sourcing images from real vineyard environments. The second dataset, in contrast, focuses on man-made infrastructure with anomalies, more specifically, bridges showing signs of degradation.

To formally describe the datasets, the notation $D_s^{i,n}$ is defined, where:

- D represents the dataset;
- i represents a specific dataset chosen from a set of datasets $I = \{Bridge\ Defects\ Imagery\ (BDI), Grapevine\ Leaves\ Imagery\ (GLI)\}$;
- s represents the specific sets $S = \{Training - Validation\ (Tr - Val),\ train\ (Tr),\ Validation\ (Val),\ Test\ (Tst), and\ external\ test\ (Ext_Tst)\}$
- n represents the number of samples within the dataset.

3.1.1 GRAPEVINE LEAVES: A NATURE-BASED DATASET

The first imagery group, described in [298], with the context of viticulture, is composed of six different varieties (Figure 6), summing up a total of 480 images ($D^{GLI,480}$). The varieties involved in this imagery group are the following: *Touriga Nacional, Tinto Cão, Códega, Moscatel, Tinta Roriz,* and *Rabigato*. In terms of acquisition procedure, weekly, between May 4th to July 31st of 2017, one leaf was picked from the same two previously selected plants of each grapevine variety, put on top of a white sheet of paper (background), and photographed (in the field), without any artificial lighting, using a Canon EOS 600D, equipped with a 50mm f/1.4 objective lens (Canon, Ota, Tokyo, Japan). For the sake of variability, data collected on cloudy days and in sunny late afternoons were also included. It is worth noting that domain experts supported leaf acquisition and labelling. The following table presents the distribution of samples per class, with 80 images assigned to each class.

Table 8. Distribution of samples across classes in the Grapevine Leaves imagery.

Class name	Códega	Moscatel	Rabigato	Tinta Roriz	Tinto Cão	Touriga Nacional
Size	80	80	80	80	80	80



Figure 6. Raw data of grapevine leaves composed of six classes (phenotypes) acquired over a white background. From left to right: *Touriga Nacional, Tinto Cão, Códega, Moscatel, Tinta Roriz,* and *Rabigato*.

3.1.2 BRIDGE DEFECTS: A DATASET FOCUSING ON MAN-MADE STRUCTURES

The second imagery group regards the context of bridge defects. This collection was created by capturing photographs of various bridges using a handheld camera and a mobile phone, resulting in a set of images with a diverse range of sizes and resolutions. Afterward, each image was manually labeled using a tool called LabelImg [299] and attributed to one of the six groups representing defect types: Paint Deterioration, Plant, Absence of Joint Cover Plate, Cracks, Pavement Crack, and Peeling on Concrete (Figure 7). In total, 2400 images were labeled. However, a pruning operation to remove unusable data had to be carried out, resulting in a final set of 1872 images ($D^{BDI,1872}$).



Figure 7. Different types of defects that are commonly found in bridges.

The class distribution of this dataset is summarized in Table 9. The Pavement Crack class contains the largest number of samples (461 images), whereas the Peeling on Concrete class is the smallest, with only 101 images. Overall, the dataset exhibits a pronounced class imbalance.

Table 9. Distribution of samples across classes in the Bridge Defects Imagery.

Class name	Absence of Joint Cover Plate	Cracks	Paint Deterioration	Pavement Crack	Peeling on Concrete	Plant
Size	149	461	406	523	101	232

The inclusion of both balanced and imbalanced datasets enables a more realistic and comprehensive evaluation of the data-splitting techniques under comparison.

3.1.3 External Imagery dataset

In addition to the main imagery dataset, another imagery set is incorporated into the evaluation process to ensure a thorough and robust assessment of the proposed methodology. This supplementary data is carefully selected to extend the diversity of the dataset, involving a broader range of features, contexts, and variations that test the methodology under more challenging and varied scenarios, ensuring its generalizability and reliability. Two other grapevine and bridge defects external datasets were included in this study (Figure 8), envisaging an extended assessment that focuses on the following aspects: a) the model prediction consistency over data that is characteristically different from the one used for training; b) CNNs' attention maps-based metrics for a richer interpretation of the trained models.

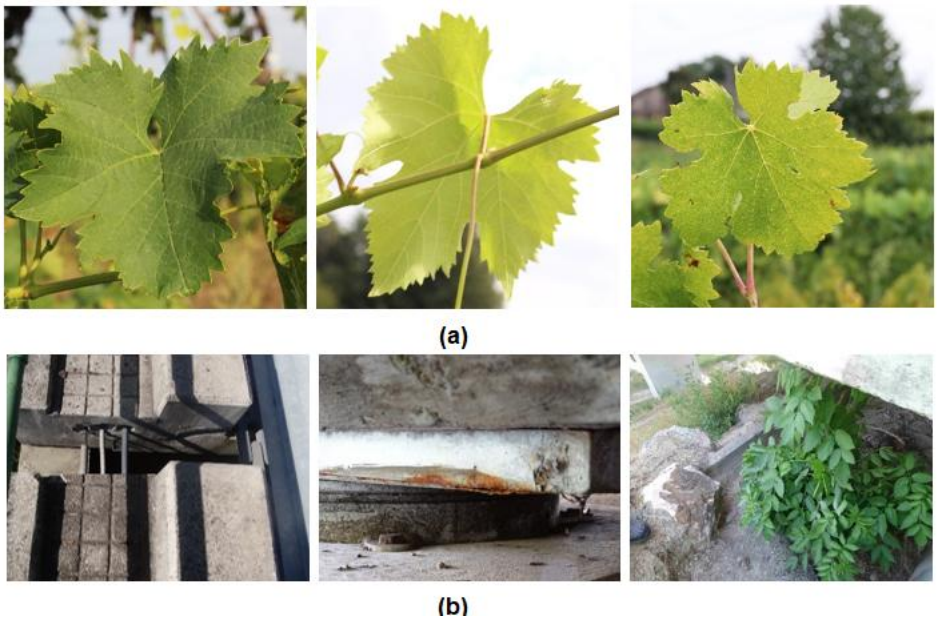


Figure 8. External image sets for DL models extended assessments: features a set of images for a) grapevine leaves imagery, b) bridge defects imagery.

Within the vineyard context, another dataset of six grapevine varieties (Figure 8a), based on imagery captured in the natural background – *Códega, Moscatel, Rabigato, Tinta Roriz, Tinto Cão, and Touriga Nacional* – was considered. It resulted from a campaign that took place from July 17th and September 20th of 2019, in intervals of two weeks, to collect 888 images of leaves, slightly unbalanced in terms of class distribution. As in the previous grapevine group, lightning conditions were mostly influenced by clean solar incidence between 1 PM and 3 PM. However, both cloudy and sunny late afternoon illumination conditions were also considered to complement the imagery variability. The acquisition equipment was the

same as that previously described for the six-class grapevine dataset, i.e., a Canon EOS 600D camera. In the scope of this study, a pruning operation was carried out to ensure that the labels/classes of both grapevine datasets involved match with each other, and a total of 10 samples per class were randomly considered to compose the $D^{GLI,60}$.

The creation of the external assessment set for the bridge defects (Figure 8b) followed a similar methodology outlined in the first collected set of this domain. Utilizing the same equipment and conditions, images were captured with a handheld camera and a mobile phone to cover the six defect classes from the bridge: Paint Deterioration, Plant, Absence of Joint Cover Plate, Cracks, Pavement Crack, and Peeling on Concrete. Like in the latter external grapevine dataset, 10 images per class were randomly selected, setting up the $D^{BDI,60}$ used for extended assessments in this study.

3.2 Exploratory Deep Learning Operations over Datasets

In this subsection, the application of traditional DL techniques over both of the previously introduced datasets is presented, with twofold intentions: (i) to preliminarily revisit some of the commonly employed procedures used to train DL models for classification tasks, and (ii) to provide a glimpse into the internal consistency and representativeness of the datasets through their early-stage exploration using conventional training configurations. First, the exploration of conventional training configurations for D^{GLI} will be discussed, followed by the one for D^{BDI}.

3.2.1 Preliminary experiments with the grapevine leaves dataset

The D^{GLI} was selected and paired with the Extreme Inception (Xception) CNN architecture for the first exploration. Xception was chosen due to its proven efficiency in extracting deep hierarchical features while maintaining a relatively lightweight structure through depthwise separable convolutions [300], [301]. Training and validation procedures were configured to run for up to 1000 epochs, enabling an in-depth view of learning behaviors and potential convergence trends.

A standardized training setup was adopted across all runs. The batch size was fixed at 24, as higher values were observed to cause memory leaks during experimentation. The number of steps per epoch was computed by dividing the total number of training images by the batch size, and similarly, validation steps were determined by dividing the number of validation images by the same batch size. All input images were resized to a target shape of 300 × 300 × 3 to align with the network's input layer specifications. Categorical cross-entropy was used as the loss function to support the multi-class classification scenario.

For optimization, Nesterov-accelerated Adaptive Moment Estimation (Nadam) was employed. Dataset splitting was performed consistently across all experiments: 10 images per class were allocated to a dedicated test set — reserved exclusively for final inference performance evaluation — while the remaining data was split into 65% for training and 35% for validation. To tackle overfitting and training inefficiency, an early stopping monitor was integrated, allowing the training process to halt if no improvements were observed in validation performance for a given number of consecutive epochs. Before training, several preliminary grid-search trials were conducted, with a particular focus on optimizing the Learning Rate (LR). These trials yielded promising results, achieving precision rates of 95% and 98% on two image sets: one comprising leaf images from six grapevine varieties captured against a white background, and the other containing leaf images from 12 grapevine varieties photographed in natural environments. The experiments consistently showed that an LR of 10^{-4} led to better convergence and model accuracy. As a result, this value was selected as one of the baselines for subsequent exploratory training.

3.2.2 PRELIMINARY EXPERIMENTS WITH THE BRIDGE DEFECTS DATASET

Another set of exploration experiments was conducted using the D^{BDI}, focusing on the benchmarking of several CNN architectures through transfer learning techniques. Pre-trained models from ImageNet served as a foundational starting point, allowing for the reusability of general-purpose visual features while adapting the final classification layers to fit the D^{BDI} specific task. The selected architectures—Mobile Neural Network-Version2 (MobileNetV2), Xception, Inception Network-Version3 (InceptionV3), Neural Architecture Search Network-Mobile (NASNetMobile), Inception Residual Network-Version 2 (InceptionResNetV2), and VGG16 — represent a diverse range of model complexities and design philosophies, from lightweight and mobile-efficient models to deep and highly expressive ones.

Given that these architectures were originally trained for ImageNet's 1000-class classification task, structural modifications were required to tailor them for the D^{BDI}, which comprises six defect categories. These modifications primarily involved replacing the top classification layers with new dense layers configured for a six-class softmax output. To maintain consistency across experiments and to ensure comparability of results, all models were adapted to accept input images of shape $128 \times 128 \times 3$.

To thoroughly evaluate architecture performance and the influence of optimization strategies, a systematic experimental pipeline was established. The pipeline was designed in stages, where models achieving higher performance in earlier tests were funnelled into subsequent experiments for further tuning. In this process, multiple optimizers were tested, including SGD and adaptive methods like Adaptive Moment Estimation (Adam), Nadam, and Root Mean Square Propagation (RMSprop). These optimizers

were paired with different initial LRs — 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, and 10^{-5} — to assess their effect on learning stability, convergence speed, and final performance.

In the context of SGD, a momentum of 0.90 was used to accelerate gradient descent in the relevant direction and mitigate oscillations. Recognizing that a static LR may not suit all phases of training, the second phase of the experiment explored LR decay strategies. These included time-based, step-based, and exponential decay policies, implemented over a prolonged training window of 100 epochs, which allowed for observing the impact of LR adjustments on deeper convergence and generalization behavior.

To ensure training sessions remained efficient and prevented unnecessary computation when a model ceased to improve, an early stopping mechanism was introduced. This callback monitored validation loss and interrupted training after 20 epochs without improvement. Furthermore, a dropout rate of 20% was applied to the dense layer of all models to mitigate overfitting, especially given the relatively modest size of the D^{BDI} and the complexity of some architectures.

Accuracy was adopted as the primary — but not single — performance metric throughout the experiments to evaluate and compare the models. In the initial phase, models were trained using four different optimizers — SGD, Adam, Nadam, and RMSprop — across a range of LRs: 1, 10^{-1}, 10^{-2}, 10^{-3},10^{-4}, and 10^{-5}. Each model underwent a 20-epoch training session with constant decay. Observations from this round revealed that extreme LR values — 1 and 10^{-5} — led to instability or very low accuracy, resulting in their exclusion from subsequent tests.

To further investigate learning potential, a 100-epoch training session was conducted under the same conditions, with early stopping enabled to terminate unproductive training. Analysis of the learning curves showed that models with higher Early Stopping Epochs (ESE) generally demonstrated more stable training, as evidenced by smoother and better-separated accuracy and loss lines. This phase confirmed that LRs above 10^{-3} were less stable, leading to the selection of 10^{-3} and 10^{-4} for continued evaluation.

The experimentation then expanded to include different LR decay schedulers — time-based, step-based, and exponential — applied to the SGD optimizer with the filtered LRs. These decays contributed to improved training stability and reduced validation loss. Based on performance, models achieving accuracy above 87% and reaching the full 100-epoch training threshold (ESE = 100) were selected as preferred classifiers.

To complement models' explainability, Grad-CAM visualizations were generated for these selected models. The results revealed a contrast in attention behavior: while InceptionResNetV2 and InceptionV3 exhibited more dispersed focus areas, other models effectively localized defects, highlighting regions of interest more precisely. Notably, a clear relationship was observed between classification confidence and

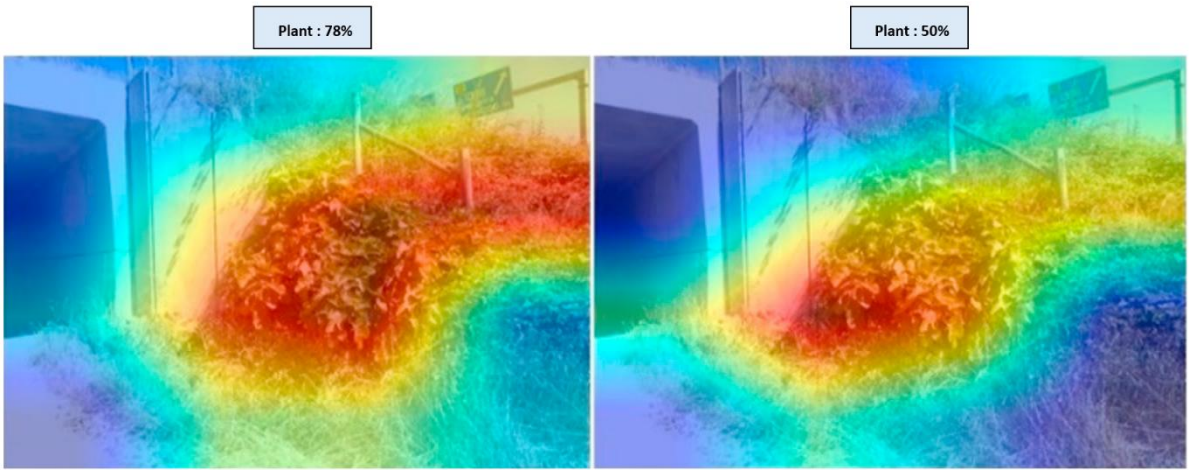


Figure 9. Grad-CAM heatmaps comparison for the presence of plants in the image, with different prediction probabilities.

the sharpness of the Grad-CAM heatmaps—higher confidence predictions tended to produce more focused, background-independent attention regions, reinforcing the validity of the classification decisions made by those models (Figure 9).

3.3 DL Workflow and the Integration of the Proposed Optimizations

The development of an ML model for supervised learning begins with a foundational and critical step: the division of the original dataset into distinct subsets designated for training, validation, and testing. This splitting process constitutes the backbone of the entire workflow, ensuring that the data is appropriately structured for the subsequent stages of model training, refinement, and evaluation. As illustrated in Figure 10, the workflow in this study begins by preparing the dataset through an initial splitting of the raw data. The original dataset is divided into two primary subsets: one for training and validation, comprising 80% of the data, and another for testing, which makes up the remaining 20%. This initial split ensures that a separate dataset is reserved for final evaluations, enabling unbiased assessments of the model's generalizability. Following this initial split, the training-validation subset undergoes further division to create separate training and validation subsets. This step employs both conventional techniques — KFCV and HO — and the Feature-Aware-based Dataset Splitting Optimization (FAbDSO) methods, which are explained in detail in Chapter 4 and demonstrate the related results.

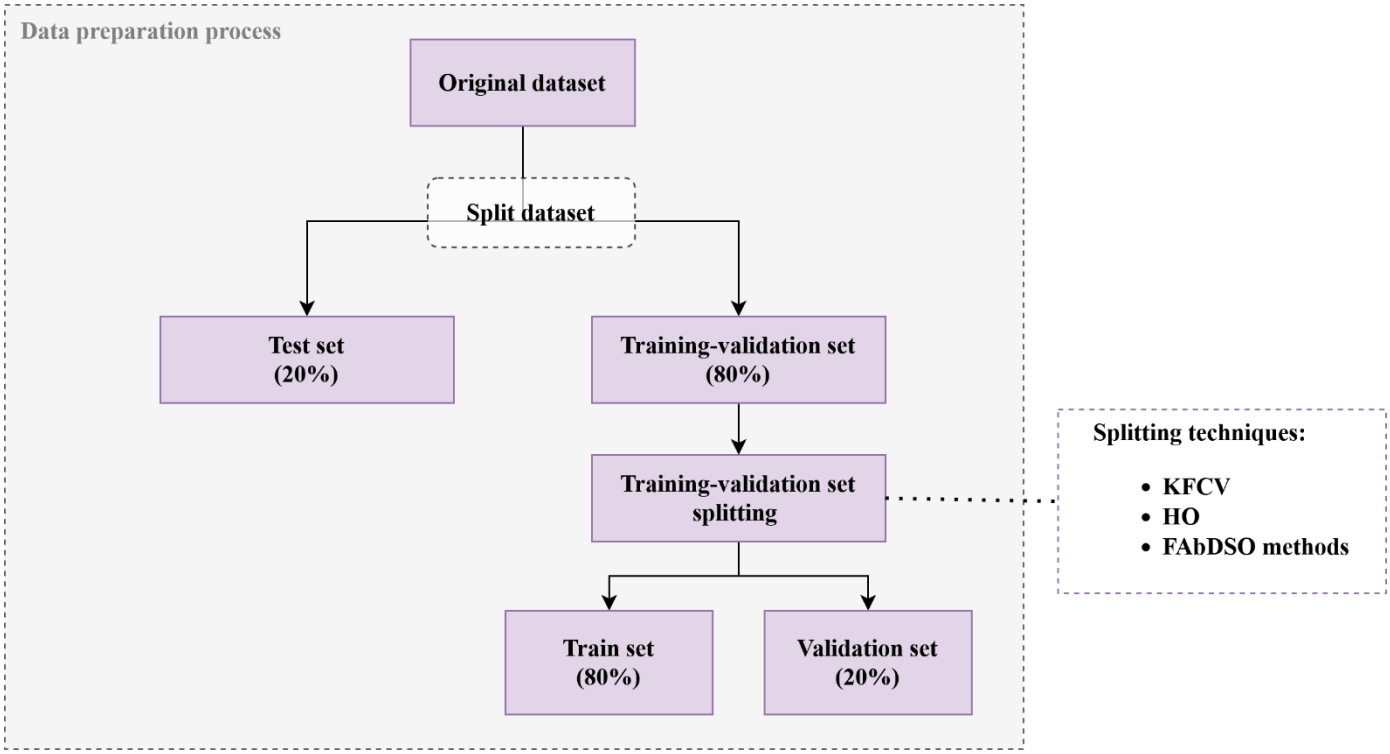


Figure 10. Dataset preparation processes.

These techniques are pivotal in optimizing data reorganization between the training and validation subsets, ensuring that the model is trained effectively while providing reliable feedback during the validation process. After the dataset has been appropriately split using the FAbDSO method, the model training stage begins, wherein an MbCL technique is proposed (Figure 11). It introduces a dynamic and iterative approach to improving the model's performance, as it will be comprehensively presented in Chapter 5. MbCL is designed to enhance the learning process by continuously integrating feedback, thereby refining the model's ability using the metrics-based validation performance.

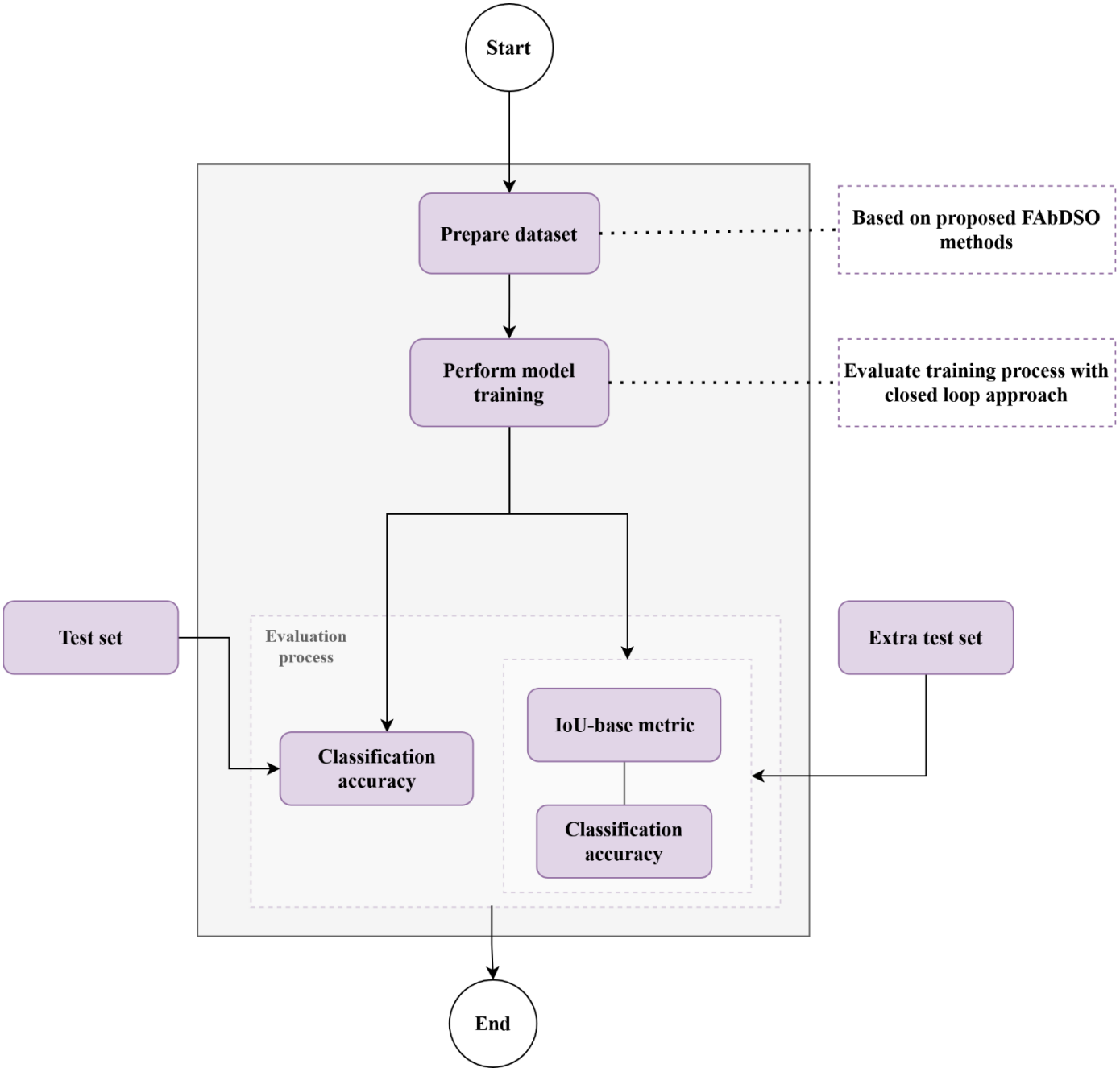


Figure 11. Methodology general workflow.

Once the training stage is completed, the evaluation step is initiated. To assess the performance of the resulting models, a suite of commonly used metrics is applied. These metrics include accuracy, which measures the overall correctness of the predictions, and IoU, which evaluates the overlap between predicted and the ground-truth data. The detailed characterization and definitions of these metrics are provided in the following:

- **Accuracy** – A fundamental metric in the field of ML, measuring the proportion of correctly predicted instances out of the total number of instances. It is usually used to evaluate how well

the model classifies and predicts classes over a testing subset. More specifically, it provides an overall assessment of a model's correctness (Eq. 1).

$$\text{Accuracy} = \frac{\text{TP}}{\text{TP+TN+FP+FN}} \quad (1)$$

where True Positives (TP) and True Negatives (TN) represent, respectively, the number of instances that were correctly classified as positive and negative by the model; in turn, False Positives (FP) and False Negatives (FN) correspond to the opposite.

- **IoU** – A spatial overlap metric commonly used in tasks involving object detection and segmentation. It quantifies the degree of overlap between the predicted regions and the ground-truth regions. Specifically, IoU calculates the ratio of the intersection area between the predicted and actual regions to the union area of those regions. IoU allows for the capture of the spatial alignment and precision of the model's output related to the true object regions. To compute it, the most distinguished attention area of CNN's Grad-CAM is used as a bounding box predictor (Eq. 2).

$$\text{IoU} = \frac{\text{OverlapArea}}{\text{UnionArea}} \quad (2)$$

where OverlapArea and UnionArea represent, respectively, the intersection and the union between ground-truth and predicted bounding boxes of a given object of interest.

3.4 EXPERIMENTAL DESIGN

Establishing a robust experimental setup and design for evaluating the proposed methodology is critical to ensure meaningful comparisons and reproducibility. This subsection describes the development environment and traditional splitting techniques used as benchmarks, as well as the rationale behind selecting CNNs over Vision Transformer (ViT) for experimental purposes.

3.4.1 DEVELOPMENT ENVIRONMENT AND TOOLS

To ensure effective training and evaluation of the DL models, an appropriate computational environment was established. The operations related to the DL models' training and evaluation took place on a computer composed of the following components:

- Processor: An Intel(R) Core (TM) i7-8700 CPU 3.20 GHz 3.19 GHz (Intel Co., Santa Clara, California, USA);

- Random Access Memory (RAM): 16.0 GB (Corsair, Fremont, California, USA);
- Graphic Card: NVIDIA GeForce GTX 1080, 16.0 GB (NVIDIA Co., Santa Clara, California, USA);
- Storage: 500 SSD (Samsung Electronics, Suwon, South Korea);
- Operating System: Windows 10 Pro (Microsoft Co., Redmond, Washington, USA).

The experiments were conducted in a configured software environment to guarantee consistency and reproducibility. Python was chosen as the programming language due to its extensive libraries and active community support in the field of ML and DL. For implementing and evaluating models, TensorFlow served as the primary framework. TensorFlow's dynamic computation graph and ease of integration with various modules made it ideal for iterative and feature-aware dataset-splitting processes.

To facilitate preprocessing, feature extraction, and traditional dataset splitting, libraries such as NumPy, Pandas, and scikit-learn were employed. Scikit-learn, in particular, was employed in applying feature engineering techniques like Principal Component Analysis (PCA). Visualization and tracking of results were performed using Matplotlib, Seaborn, and TensorBoard, ensuring a transparent and interpretable experimental pipeline.

The study also incorporated PyTorch for GAN-based data augmentation. Its robust support for GANs was leveraged to enrich underrepresented classes, thus addressing class imbalance issues detected during validation.

3.4.2 Preliminary Benchmarking Trials for Training Configuration Fine-Tuning

To determine the appropriate architecture and hyperparameter configurations for the defined objectives, a comprehensive exploratory study was conducted — as addressed in Section 3.2 — highlighting a set of relevant factors and indicators that were considered along this work.

Among them, there are comparisons between various DL architectures, including MobileNetV2 [302] , Xception [303], InceptionV3 [304], NASNetMobile [92], VGG16 [165], and InceptionResNetV2 [305]. Figure 12 displays these six DL architectures, highlighting their diverse design approaches.

Also, different optimizers were incorporated for comparison purposes, such as Nadam, Adam, RMSprop, and SGD, crossed with various LRs ranging from 1 to 10^{-5}, which were submitted to 100-epoch training sessions. The key findings that resulted from these trials are summarized as follows:

- Model Performance — Among the evaluated architectures — MobileNetV2, Xception, InceptionV3, NASNetMobile, VGG16, and InceptionResNetV2 — Xception, VGG16, and demonstrated superior stability in learning curves and performance, making them the most reliable choices for this work.
- Feature Localization (Grad-CAM Analysis) — Grad-CAM revealed that models like InceptionResNetV2 and InceptionV3 often focused on irrelevant areas outside the regions of interest, reducing their reliability. Models with higher certainty rates, such as Xception and VGG16, displayed sharper and more focused colormaps concentrated on the target area rather than noisy backgrounds.
- Optimizer Performance— Adaptive optimizers, including Nadam and Adam, consistently outperformed SGD, even when paired with LR decay schedulers, reinforcing their selection for robust training.
- LR Sensitivity— 10^{-3} and 10^{-4} produced the most effective training results, with 10^{-3} seeming to offer a balance between convergence speed and stability.

These insights led to filter architectures for Xception and VGG16, given their apparent stability and performance in similar tasks. To further evaluate the efficacy of one of the proposed methodologies, ResNet50 was additionally incorporated.

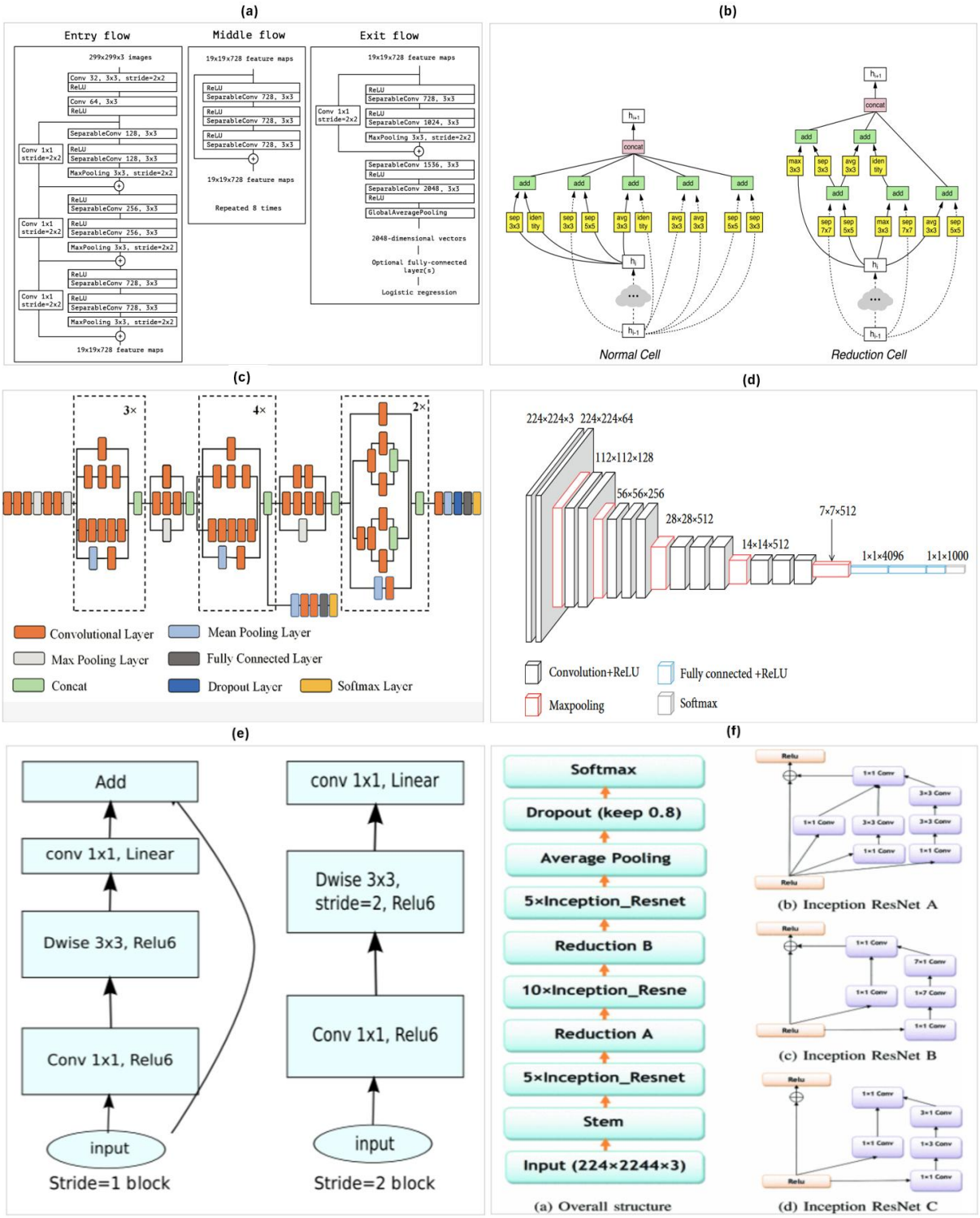


Figure 12. Various DL architectures: a) Xception, b) NASNetMobile, c) InceptionV3, d) VGG16, e) MobileNetV2, and f) InceptionResNetV2, adopted from previously published works [303], [92],[304],[165],[306],[305].

Building upon the findings from the previous study stage, a standardized training configuration was adopted to ensure consistency across all experiments. The training setup involved the following:

- Image dimensioning: All images were resized to dimensions of 128×128×3 to balance computational efficiency with the preservation of salient deep features.
- Optimizers — Nadam and SGD were utilized with initial LRs of 10^{-3} and 10^{-4}. Both LRs were used to gain practical experience and evaluate their effectiveness within the context of the proposed methodologies.
- Batch size — A fixed batch size of 32 (adjusted to imagery characteristics in terms of size and definition) was chosen to balance computational efficiency and gradient stability, as it was the maximum size suitable for the available hardware.
- Epochs — A maximum of 100 training epochs was allowed to ensure enough space for the model to try to converge while avoiding excessive overfitting and unnecessary computational overhead. An early stopping criterion consisting of the verification of learning stagnation for 20 consecutive epochs with a sensitivity of 10^{-4} regarding validation metric analysis was considered, as those seemed to be suitable values to accommodate learning variations and adaptability intervals in line with models' training.
- Regularization — After preliminarily assessing dropout rates of 0.2 and 0.5, the former value was adopted, since the latter, despite effectively reducing overfitting, also interfered negatively with the model´s learning abilities, in relative terms.

This study, supported by a consistent training framework and carefully selected architectures and optimizers, establishes a reliable foundation for evaluating the proposed methodology. Regarding the dataset-splitting techniques, the experimental results of the proposed methods are compared with those of traditional splitting techniques. The next subsection provides a detailed explanation of the traditional dataset-splitting methods used for this comparison.

3.4.3 TRADITIONAL DATASET CHUNKING AND DIVISION METHODS FOR DL EVALUATION

Two widely used dataset splitting methods, KFCV and HO, were selected as benchmarks to compare with the proposed feature-aware techniques. These conventional approaches are foundational in DL research and serve as standard practices for dataset splitting.

- **KFCV**

  KFCV is a comprehensive method that divides the dataset into k equally sized subsets (folds). Each fold serves as the validation set once, while the remaining $k - 1$ folds are used for

training. This cycle repeats k times, providing a robust assessment of model performance by averaging the results across all iterations. While KFCV minimizes the bias associated with a single data split, it is computationally intensive, particularly for large datasets, as it requires multiple training cycles. This computational demand can hinder experimentation when rapid iterations are needed.

- **HO**

  The HO method involves a simple split of the dataset into distinct subsets for training, validation, and testing. While computationally efficient, HO is sensitive to the representativeness of the split. Imbalanced classes or uneven feature distributions can lead to suboptimal performance, as the model might overfit or fail to generalize effectively.

These traditional techniques, while widely adopted, have inherent limitations such as inefficiency in addressing feature distribution imbalances and class representation issues. The proposed feature-aware methods, which consider feature diversity and class distributions, aim to overcome these challenges.

3.4.4 AUGMENTATION TECHNIQUES

Data augmentation is a fundamental technique for addressing data imbalance and improving model generalization, particularly in scenarios involving limited datasets. This research utilizes two augmentation strategies within the MbCL framework: (i) traditional augmentation methods and (ii) generative augmentation using cGANs. These techniques complement each other to enhance model robustness and adaptivity, especially in the context of underrepresented classes in real-scene datasets.

Traditional Augmentation Strategies

Traditional image augmentation is a fundamental technique in computer vision that artificially expands the training dataset by applying label-preserving transformations to existing images, without the need to collect new images. These methods are especially valuable when dealing with limited, imbalanced, or biased datasets — challenges frequently encountered in real-world applications [50], [307]. Below are some of the most widely used transformations, which serve as the foundation for traditional image augmentation pipelines:

- Geometric Transformations — Include operations such as flipping, rotation, translation, and scaling. These modifications simulate different perspectives or orientations of objects while

preserving the class label. For instance, rotating or flipping an object in a scene preserves its semantic meaning.

- Cropping and Padding — Consists of random cropping that introduces partial views of objects, which encourages the model to learn local features. Padding maintains spatial resolution after transformations, ensuring input dimensionality remains consistent.
- Color Space Adjustments — Involves techniques such as brightness, contrast, hue, and saturation changes to simulate variations in lighting conditions, camera sensors, or environments. These adjustments force models to rely more on shape and structure than colour alone.
- Noise Injection — Consists of adding Gaussian, salt-and-pepper, or speckle noise improves robustness against sensor imperfections or low-quality inputs.
- Blur and Sharpening — Involves slight blurring or sharpening, mimics real-world distortions, helping models learn more generalizable edge and texture features.
- Cutout and Random Erasing — Encompasses masking random patches of an image to simulate occlusion or missing data. This encourages the model to focus on the most discriminative parts of the object.

Traditional augmentation is computationally inexpensive and easy to integrate into training pipelines. However, these methods are rule-based and may not sufficiently address complex data limitations such as class imbalance, intra-class variability, or rare pattern representation. Furthermore, excessive or misapplied augmentation can lead to label noise or hinder learning, especially if semantic content is exaggeratedly altered. Hence, while traditional augmentation paves the way for a general foundation for models, it is increasingly supplemented or replaced by learned or generative augmentation strategies in modern DL pipelines.

Conditional Generative Adversarial Network Augmentation

One of the key limitations of traditional GANs is the lack of control over the content or attributes of the generated data [308], [309], [310]. Since the generator in a standard GAN maps random noise to data without any guidance, the outputs tend to be diverse but arbitrary. This makes such models less suitable for applications that require structured, task-specific, or semantically meaningful outputs [311], [312]. cGANs [234] as an extension of the original GANs [313] address this limitation by incorporating additional information as class labels to condition the generation process. cGANs introduced by Mirza et al. in 2014, aim to control the data generation process by conditioning both the generator and the discriminator on supplementary information (class labels). Like standard GANs, cGANs consist of two primary components:

- Generator (G)— Learns to generate data instances x from random noise z and an additional condition y to produce data that appears indistinguishable from real data, conditioned on y.
- Discriminator (D)— Evaluates whether a given instance x conditioned on x is real (from the dataset) or fake (from the generator).

The generator function can be expressed as $G(z|y)$ and the discriminator function as $D(x|y)$. The objective function of a cGAN is a modified version of the original GAN loss:

$$\min_G \max_D \mathbb{E}_{x\sim p_{Data}(x)} [\log D(x|y)] + \mathbb{E}_{z\sim p_z(z)}\big[\log\big(1 - D(G(z|y)|y)\big)\big] \qquad (3)$$

Where:

- G is the generator, which tries to produce realistic data conditioned on y,
- D is the discriminator, which tries to distinguish real samples from generated ones,
- $x\sim p_{Data}(x)$ represents real data samples,
- $z\sim p_z(z)$ denotes random noise vectors sampled from a known distribution (e.g., Gaussian),
- y is the condition provided to both the generator and the discriminator.

The symbol $\mathbb{E}$ stands for expectation, which refers to the average value of the expressions inside the brackets. Specifically:

- $\mathbb{E}_{x\sim p_{Data}(x)}[\log D(x|y)]$ is the average log-likelihood that the discriminator assigns to real samples x conditioned on y,
- $\mathbb{E}_{z\sim p_z(z)}\big[\log\big(1 - D(G(z|y)|y)\big)\big]$ is the average log-likelihood that the discriminator correctly identifies generated samples $G(z|y)$ as fake, again conditioned on y.

In this formula, $x\sim p_{Data}(x)$ represents the true data distribution, meaning the distribution from which the real samples x are drawn. Since this distribution is generally unknown, we rely on a finite dataset of real examples that are assumed to come from p_{Data}. The generator aims to produce samples that mimic this distribution as closely as possible, conditioned on y. Meanwhile, $p_z(z)$ is the distribution of the noise input z , often chosen to be a simple distribution like Gaussian or uniform.

In image-based cGANs, the conditioning information y — such as class labels — must be effectively integrated into both the generator and the discriminator. The way this condition is injected plays a crucial

role in determining the generated outputs' relevance, quality, and fidelity. One practical strategy to incorporate this conditioning information in DL models is concatenation [234], a widely used method. In this approach, the condition y is appended directly to the input vectors of both the generator and the discriminator. For instance, if y is a one-hot encoded class label, it is concatenated with the noise vector z to form a single composite input to the generator. On the discriminator side, the same condition is concatenated with the image (real or generated) to help the model learn whether the image is not only realistic but also contextually consistent with y.

The following figure illustrates the typical architecture of a cGAN. The system consists of two adversarial components: the Generator and the Discriminator, both of which are explicitly conditioned on external information (class labels).

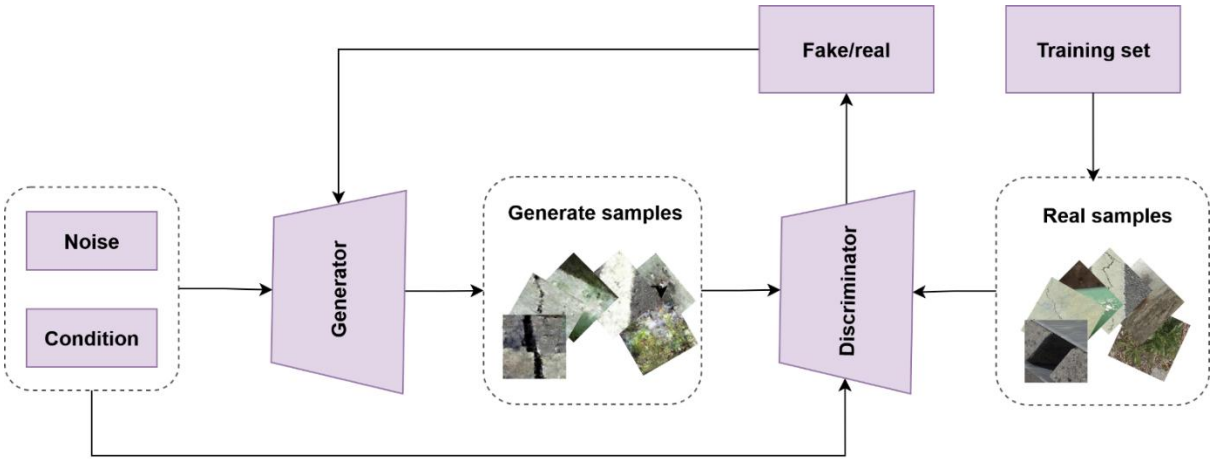


Figure 13. cGAN network architecture.

In the generator, a random noise vector, sampled from a known prior distribution (e.g., Gaussian), is concatenated with labels. This combined vector is fed into a series of layers. The generator's role is to produce synthetic data that closely resembles the real data corresponding to the condition. The discriminator, on the other hand, receives as input both the real samples and the same labels. The discriminator then outputs a probability score indicating whether the data, given the label, is real or fake (Figure 13).

3.4.5 Method Proposal Overview: Feature Engineering and Dataset Splitting Before Training

The data split approach employed in this study integrates two essential and interdependent steps: feature engineering and dataset splitting. These steps work jointly to ensure that the resulting subsets are not only representative of the original dataset but also structured to optimize the training, validation, and

testing phases of model development. A visual representation of the general workflow encompassing these steps is provided in Figure 14.

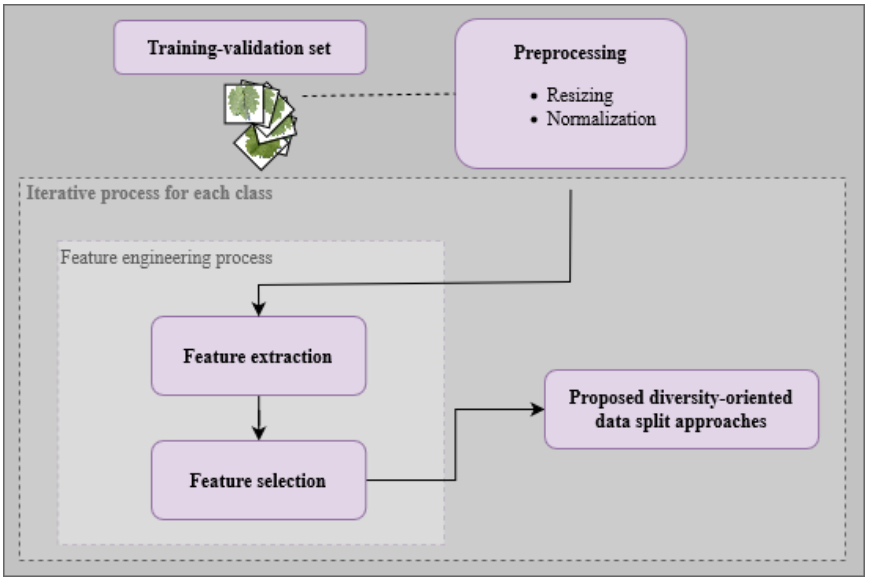


Figure 14. General workflow of the split process

Feature engineering serves as the first step in ML [314] and computer vision [315], where critical characteristics of the data are identified, extracted, and analyzed. This process involves transforming raw data into meaningful features that can enhance the dataset's utility for ML models. The process of feature extraction and dimensionality reduction plays a pivotal role in enhancing the performance of ML models [316], [317], particularly in tasks where data complexity is high.

Traditional feature extraction techniques, while useful in structured or simpler datasets, often fall short when applied to complex, high-dimensional data such as imagery. Methods like the Histogram of Oriented Gradients (HOG) [318], SIFT [319], and Speeded-Up Robust Features (SURF) [320] have been instrumental in the past, extracting edges, textures, and key points to represent patterns in an image. However, these traditional techniques are often limited by their reliance on manual engineering and their difficulty in capturing the complexity and variability of real-world data [321].

DL has revolutionized this aspect by providing an automated, highly effective alternative that overcomes many limitations associated with traditional methods [322], [323], [324]. One of its most significant advantages is its ability to learn hierarchical features directly from raw input data [325]. This hierarchical learning process eliminates the need for labour-intensive manual feature engineering, ensuring that the extracted features are not only relevant but also optimized for the task at hand [326].

Another key benefit is the transferability of DL features [327], [328]. Models pre-trained on large datasets like ImageNet have been shown to extract general-purpose features that can be fine-tuned for

specific tasks, making them a powerful starting point for feature extraction in domains with limited labeled data.

In this study, DL feature extraction was chosen to leverage these advantages. Specifically, the VGG16 CNN architecture was employed as the primary feature extraction tool. VGG16 is a well-established DL model known for its ability to recognize hierarchical patterns in visual data [329]. It employs a deep stack of convolutional layers followed by fully connected layers, which helps it capture fine-grained features in images. By leveraging the pre-trained weights of VGG16, the model could benefit from a broad understanding of visual patterns, reducing the need for training from scratch and speeding up the overall feature extraction process.

Once the features were extracted, the next step was to optimize their representation and reduce the dimensionality of the data using PCA [330]. PCA is a well-known statistical technique [331], [332] that transforms the data into a new set of orthogonal components, ordered by the amount of variance they explain in the dataset. The core idea behind PCA is to reduce the number of dimensions in the data while preserving as much of its variability as possible.

After feature engineering, traditional dataset-splitting methods like random or stratified sampling often led to unbalanced feature distributions, negatively affecting model performance and evaluation. To address this, two novel one-shot, feature-aware splitting techniques are proposed. These methods automatically create balanced training and validation subsets based on feature distributions. The detailed explanation of these methods will be presented in Chapter 4.

3.4.6 Method Proposal Overview: Feature-Aware Representativeness Enhancement via Closed-Loop Training

The proposed MbCL methodology is designed to enhance the effectiveness and adaptability of DL models by introducing a feedback-centric, iterative training process. Unlike conventional model development pipelines that rely heavily on static datasets and generalized hyperparameter tuning, MbCL adopts a dynamic, metrics-guided framework that integrates validation-driven feedback at the core of its learning process. The goal of this approach is to improve both overall accuracy and class-specific performance by addressing data imbalance and underrepresentation through targeted intervention.

At the core of MbCL lies a continuous evaluation and refinement loop, initiated after the completion of an initial training stage. In this stage, the trained model is evaluated using a performance metric such as

overall accuracy. Beyond these global metrics, the methodology also emphasizes class-wise validation, examining performance at the individual class level to identify outliers in the model's predictive capabilities.

When certain classes are observed to consistently underperform — defined through thresholds set on validation accuracy — these classes are flagged as outliers. Instead of proceeding with a generic retraining approach, MbCL introduces a targeted augmentation mechanism aimed at enriching the training data for these specific classes. This mechanism consists of two complementary strategies: conventional data augmentation and GAN-based augmentation.

Conventional data augmentation involves applying well-established image transformations such as rotations, scaling, translations, flips, and brightness adjustments. Also, GAN-based augmentation is employed to synthesize entirely new data samples. GANs, which leverage a dual-network training approach involving a generator and a discriminator, are capable of producing high-quality, contextually relevant images that match the complexity of real-world data. By training the GAN specifically on the underperforming classes, the generated samples closely represent the challenging features that the model previously struggled to learn. This not only augments the training dataset with novel instances but also mitigates the dependency on manual data collection, offering a scalable and automated solution to data scarcity.

Such generated samples are reintegrated into the training dataset, and the model undergoes another round of training. This process is iterative, with each cycle including re-evaluation based on the same set of validation metrics. Underperforming classes are continually monitored, and if necessary, re-augmented in subsequent iterations. This loop continues until the validation performance of all classes converges toward a predefined threshold of uniformity or meets the early-stopping criterion, which ensures computational efficiency by halting training once further improvements become slight (Figure 15).

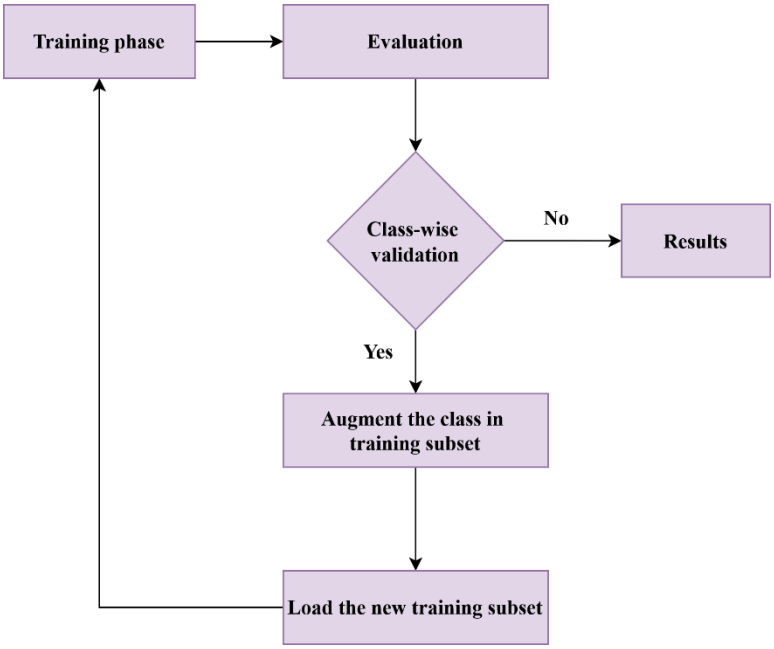


Figure 15. Overview of the MbCL methodology.

3.4.7 The Rationale for Selecting CNNs Over ViTs

The choice of CNNs as the primary architecture for experimentation was driven by several practical and theoretical considerations. CNNs have long been regarded as the de facto standard for computer vision tasks due to their computational efficiency, hierarchical feature extraction capabilities, and extensive optimization in diverse datasets. While ViT [333] has emerged as a significant breakthrough in the field of computer vision [334], [335], CNNs remain the go-to choice for a variety of reasons, particularly in the context of this study:

- Efficiency — Compared to ViT, CNNs require fewer computational resources for training [336]. ViT is more suited for significantly larger datasets and, therefore, requires longer training times to perform effectively due to its reliance on self-attention mechanisms, which scale quadratically with the number of image patches. Conversely, this study's focus is on exploring techniques more adapted to short/medium-sized datasets (i.e., groups of data involving some hundreds to a few thousands of images and a relatively short number of classes), requiring rapid training cycles, in which CNNs appear to have higher efficiency.
- Dataset size considerations — ViT excels in scenarios involving extremely large datasets [336] where their self-attention mechanisms can uncover global dependencies. However, for medium-sized datasets or small ones, CNNs provide excellent performance without requiring extensive data augmentation or pretraining.

- Compatibility with proposed feature-aware methods — The proposed dataset-splitting methods in this study heavily rely on feature extraction for tasks such as clustering and centroid-based selection. CNNs are well-suited to this task due to their ability to capture spatial hierarchies and extract meaningful features efficiently. In contrast, ViT is powerful in extracting global features and information [337], [338].

Given the scope of this study and the focus on dataset-splitting techniques, CNNs were more aligned with the research objectives, ensuring effective feature extraction without the need for extensive pre-training or massive datasets.

In the following chapter, the FAbDSO methods are detailed, presenting two novel strategies aimed at improving data splitting for DL tasks. These methods are designed to address limitations found in conventional splitting techniques by incorporating feature-level information and class distribution awareness into the dataset-splitting process. The theoretical foundation, implementation steps, and practical advantages of FAbDSO are presented, highlighting its potential to enhance training stability.

4 Proposed Feature-Aware-based Dataset Splitting Optimization methods

The success of any DL/ML model depends fundamentally on the quality and quantity of the dataset used during the training, validation, and testing stages, while its distribution is also important for the dissemination of core characteristics that can be both learned and validated. A properly built dataset serves as the foundation for robust model development, as it directly impacts the model's ability to learn meaningful patterns, generalize to unseen data, and deliver accurate predictions. High-quality data ensures relevance, accuracy, and diversity in the information provided to the model, while adequate quantity allows the model to capture complex variations and reduce the likelihood of overfitting. Consequently, the process of dataset preparation and splitting is not merely a procedural step but a critical component of the ML/DL workflow that determines the overall success of the model.

Having said that, a major challenge associated with the aforementioned process lies in effectively dividing the data into subsets for training, validation, and testing in a manner that preserves the integrity and representativeness of the dataset. Training subsets must provide sufficient data to enable the model to identify underlying patterns and relationships, while validation subsets are essential for fine-tuning model hyperparameters and preventing overfitting. Testing subsets, on the other hand, must remain entirely separate and unseen during the training process to offer an impartial evaluation of the model's performance on imagery that the model does not know, i.e., unseen data. Conversely, improperly splitting the dataset can lead to underperforming inference capabilities, biased results, unreliable evaluations, and, consequently, a model that fails to generalize effectively in real-world scenarios. Among the perceived challenges is the need to mitigate potential overfitting, which affects models of higher complexity more severely [339]. Therefore, developing and employing strategies to find the proper balance between generalization capabilities and biases contributes to the achievement of more suitable inference capabilities. One of these strategies may consist of finding adequate training data arrangements to ensure harmonization, configured by a balanced distribution of examples per class/label, representativeness, and diversity, among other aspects. In that regard, the data-splitting process for setting up datasets is crucial for the achievement of reliable and consistent models [340]. To address these challenges, several approaches to dataset splitting have been developed [341], each with its strengths and limitations. One of the most commonly used methods is the HO approach, where the dataset is divided into fixed proportions. This straightforward technique is practical and effective when the dataset is sufficiently large to maintain representativeness across all subsets. Another widely used method is KFCV, which is a multiple chunk-based splitting approach that ensures the whole training and validation data is utilized with different arrangements, providing a more comprehensive evaluation of the model's performance and reducing the

impact of a poorly chosen split. Traditional dataset-splitting approaches present, however, some limitations. Both HO and KFCV may lack the resources to explore the model's inference potential, as they rely on blind data-splitting techniques. Besides, and especially valid for KFCV, the multiple chunk-based splitting consists of a trial-and-error-like pipeline that implies the use of the computational resources for longer time periods. Therefore, there is room for optimizations capable of tackling the mentioned drawbacks, for example, at the dataset level.

This chapter explores the details of introducing and evaluating two novel one-shot dataset-splitting techniques designed to overcome the inherent limitations of conventional methods. These approaches emphasize the creation of feature-based balanced datasets, moving away from the trial-and-error nature of traditional splitting strategies. These techniques lay a lightweight, feature-aware exploration framework, designed to automatically and efficiently organize data into subsets with consistent, even, and uniform feature distributions across training, validation, and testing subsets. By achieving such a balance, the proposed techniques seek to maximize the convergence efficiency of DL models and provide more representative evaluations of their performance while reducing the needed time for computational resource allocation. As such, the potentials of feature-based dataset-splitting techniques as a robust alternative to conventional methodologies are explored, particularly in scenarios involving limited or imbalanced datasets.

4.1 Initial Data Handling and Feature Engineering

Data splitting and feature engineering are of great importance for the success of DL operations. DL-based feature extraction is highly robust to variations in data [342]. Changes in scale, orientation, and illumination, which might confound traditional methods, are effectively managed through convolutional operations and pooling layers. This robustness is particularly valuable in real-world scenarios, where datasets are often noisy and diverse. Moreover, DL methods excel at dimensionality reduction, transforming high-dimensional input data into compact, discriminative representations while preserving essential information. This not only improves computational efficiency but also reduces the risk of overfitting by discarding irrelevant or redundant features. After feature extraction, the well-known dimensionality reduction technique [343] — PCA — was employed. To implement PCA effectively, a systematic approach was adopted, starting with the goal of retaining components that explained at least 99% of the total variance in the dataset to ensure that the vast majority of the information in the original data was preserved, while components with minimal explanatory power — such as those representing noise or irrelevant variations — were discarded. Before applying PCA, the dataset was standardized (Figure 16)

to ensure that all features were on a comparable scale. Standardization transforms the data such that each feature has a mean of zero and a standard deviation of one. This step is crucial because PCA is sensitive to the scale of the input data; features with larger numerical ranges could otherwise dominate the variance calculation, skewing the resulting principal components. By standardizing the data, each feature contributes equally to the PCA process, allowing the transformation to better reflect the underlying structure of the dataset.

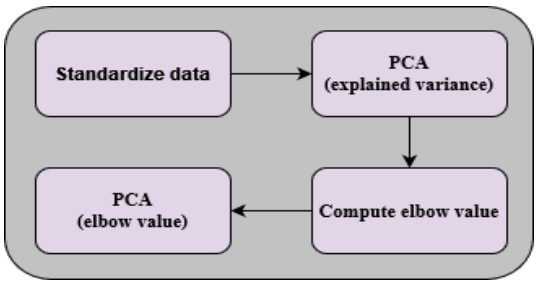


Figure 16. Key steps of PCA.

After standardization, the next step involved determining the optimal number of components to retain for the dimensionality reduction. This was achieved by calculating the explained variance ratio for each principal component. The explained variance ratio indicates the proportion of the total variance in the data that is captured by each component. To identify the optimal number of components, the elbow method was employed [344], [345]. This method involves plotting the cumulative explained variance and identifying the elbow value or the point at which the addition of more components results in a minimal increase in the explained variance. This point marks the optimal number of components to retain, as it balances the trade-off between dimensionality reduction and the preservation of key information.

Once the elbow value was identified, the final step in the PCA process was the actual dimensionality reduction. Only the principal components that were most informative were retained. This reduction led to a new feature space that was more compact, retaining the essential information from the original dataset while eliminating unnecessary dimensions.

In the proposed dataset split approach, one of the key parameters is computing the centroid points of the features, which play a pivotal role in organizing and structuring the dataset. By calculating the centroid of the data, we obtain a representative center of the feature distribution, which serves as an essential parameter for the subsequent stages of the dataset-splitting process. An illustrative example of this centroid-based process is shown in Figure 17, where a distinctive red cross denotes the centroid.

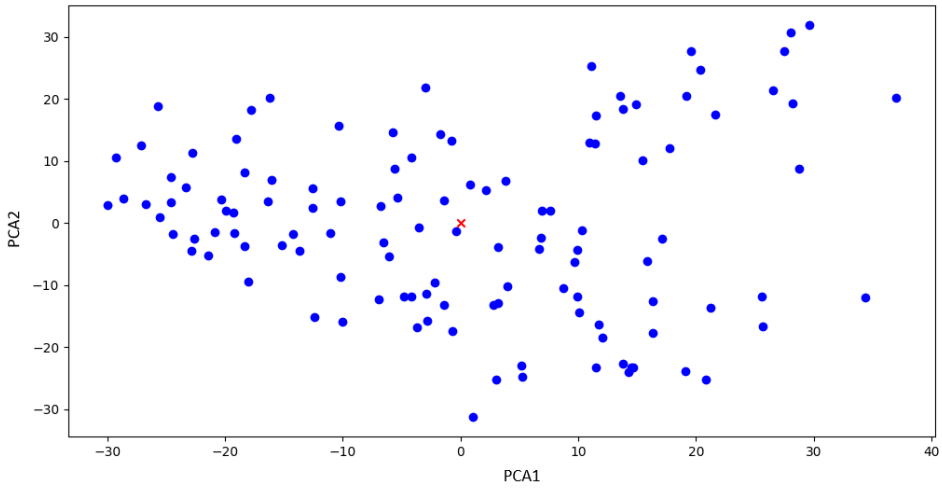


Figure 17. Visualization of the centroid point of features for the class of "Absence of Joint Cover Plate" for D_{Tr-Val}^{BDI}, for exemplification purposes. Blue dots represent the distribution of features, while the red cross is the global centroid.

Two systematic, feature-aware methodologies are proposed for dataset splitting, aiming to create diversity-oriented dataset structures. These methods — Feature Clustering Distance-based Image Selection (FCDIS) and Feature Space Center-based Image Selection (FSCIS) — address the limitations of conventional dataset splitting techniques by incorporating the feature distribution and diversity of the data into the process. Both methods are designed to balance feature representation across the subsets, ensuring that the training and validation datasets are representative of the original dataset's complexity and diversity. Both methods follow a consistent initial step, which involves splitting the D_{Tr-Val} into two subsets: the D_{Tr} and the D_{Val}. Specifically, 80% of the D_{Tr-Val} is allocated to D_{Tr}, while the remaining 20% is reserved for D_{Val}.

4.2 Feature Clustering Distance-based Image Selection

This approach employs clustering techniques to organize the dataset based on feature similarity. By grouping data points that share common patterns or attributes within the feature space, FCDIS identifies representative samples for each cluster. The selection process is distance-aware, ensuring that images from distinct feature groups are proportionally distributed across the training and validation subsets.

The FCDIS method begins with a fundamental step: organizing the extracted features into structured groups using clustering techniques. This process lies in the K-means clustering algorithm, a widely used unsupervised learning method that groups data points based on feature similarity. This clustering step is crucial because it allows the dataset to be divided into subsets that encapsulate distinct and meaningful patterns within each class, providing a foundation for the subsequent splitting process.

In the context of the FCDIS method, features are first extracted from the dataset, ensuring that the representation of each data sample reflects its unique characteristics. These extracted features are then passed to the K-means algorithm. K-means works iteratively to partition the data into K clusters, where K represents the desired number of groups. Each cluster is defined by its centroid, which represents the average position of all points within that cluster. The algorithm minimizes the sum of squared distances between each data point and its respective cluster centroid, ensuring that data points within a cluster are as similar as possible while maximizing the dissimilarity between different clusters.

The clustering step serves as a key mechanism for feature-aware data organization, as it helps to identify patterns within the dataset. Once the clusters are formed, the FCDIS method proceeds to assign data samples to either the D_{Tr}, or D_{Val} based on their distance from the class center. To provide clarity and a better understanding of the FCDIS methodology, Figure 18 offers an illustrative depiction of the clustering and splitting process. The figure includes an overview of the splitting process (Figure 18a), accompanied by a plot highlighting how the clusters contribute to the formation of D_{Tr} and D_{Val} (Figure 18b).

A critical aspect of the K-means algorithm's effectiveness lies in the careful selection of the K parameter, which defines the number of clusters. Choosing an inappropriate value for K can lead to either under-segmentation (where clusters are too large and lack specificity) or over-segmentation (where clusters are too small and fragmented). To address this, the precomputed elbow value determination step is employed to determine the optimal number of clusters (Figure 19a). Fine-tuning K in this manner ensures that the clusters effectively represent the feature space, capturing meaningful patterns while avoiding redundancy or overfitting [345], [346].

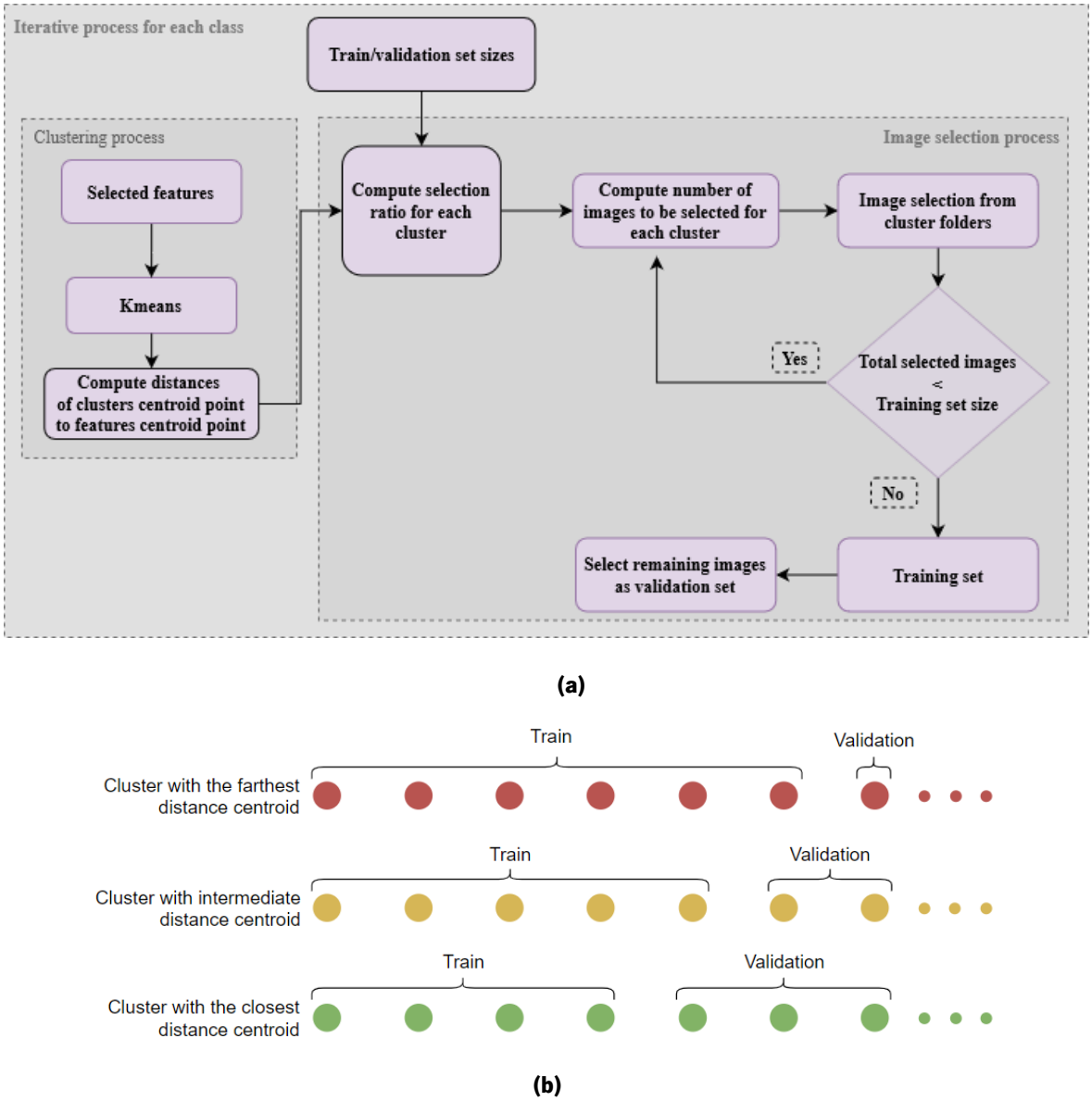


Figure 18. FCDIS process for setting up training and validation subsets: a) presents the main pipeline of the process; b) depicts the pipeline's general result-oriented concept using 3 abstract clusters as examples.

An iterative process (Figure 18a) is utilized within each class to systematically select images for inclusion in the training subset, ensuring a balanced and representative sampling of the feature space. Once the feature space has been segmented into clusters, the distance ratio becomes the guiding principle for image selection. This step ensures that the data points allocated to the training subset are not only representative of their clusters but also take into account their positional relationships within the broader feature space.

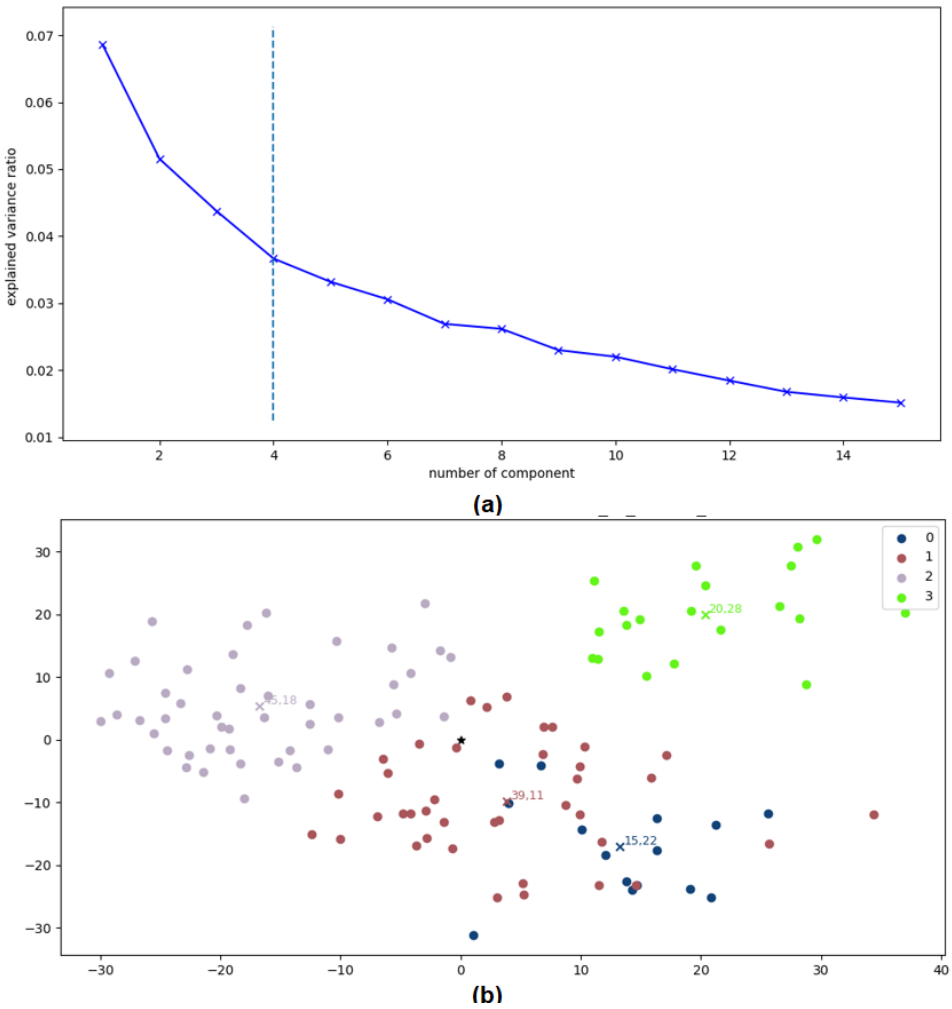


Figure 19. Example of K-means-based clustering process plot perspective for the "Absence of Joint Cover Plate" class in D_{Tr-Val}^{BDI}. a) is a chart depicting the K value determined from the Elbow technique, and b) presents the clustering results. A black star indicates the global center, and colored crosses mark the cluster centers.

To compute the distance ratio, the center of each cluster is compared to the global centroid of the entire feature space. The global centroid represents the central point of the overall distribution of features and serves as a reference for evaluating the spatial positioning of each cluster (Figure 19b). In Figure 19b, the global center point is represented by a distinctive black star and cluster center, depicted with a colored cross that matches the hue of the data points within that specific cluster. For each cluster, the distance from its centroid to the global centroid is calculated. The latter center corresponds to the global one, denoted as C , which is the mean of the top-2 principal components in the reduced feature space. This distance is computed using the Euclidean distance formula:

$$Distance(c_i, C) = \sqrt{(c_{ix} - C_x)^2 + (c_{iy} - C_y)^2} \quad (4)$$

where (c_{ix}, c_{iy}) are the coordinates of the i-th cluster centre, and correspond to(C_x, C_y) the coordinates of the global centre.

Clusters positioned farther away from the global centroid are prioritized in the selection process, as they likely represent more distinctive or less frequent patterns in the data. A metric-based distance is used to determine the number of images selected from each cluster, according to the following formula:

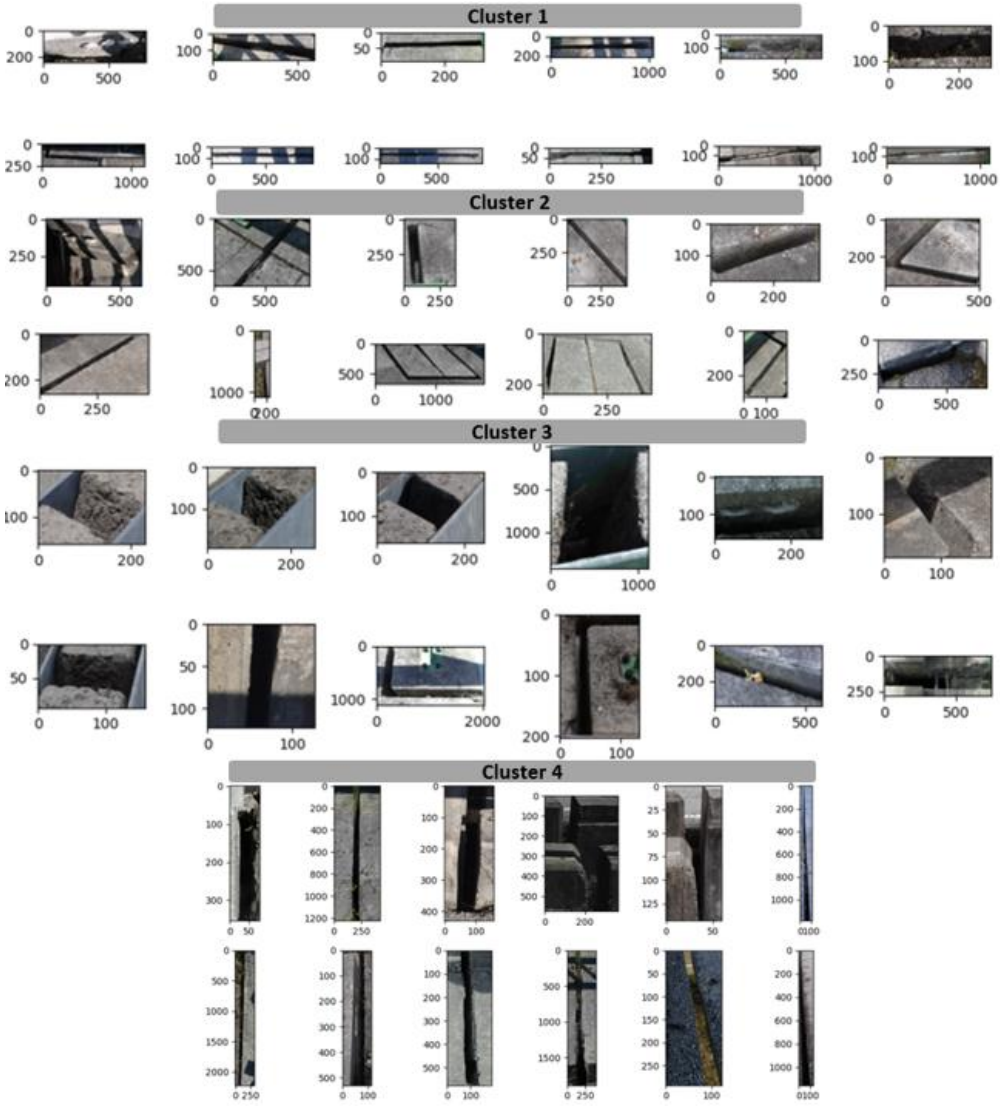
$$Image\ Selection\ Size(c_i,) = \frac{Distance(c_i,C)}{\sum_{j=1}^{n} Distance(c_j,C)} \qquad (5)$$

where c_i denotes the i-th cluster, C is the global centroid, and n is the total number of clusters.

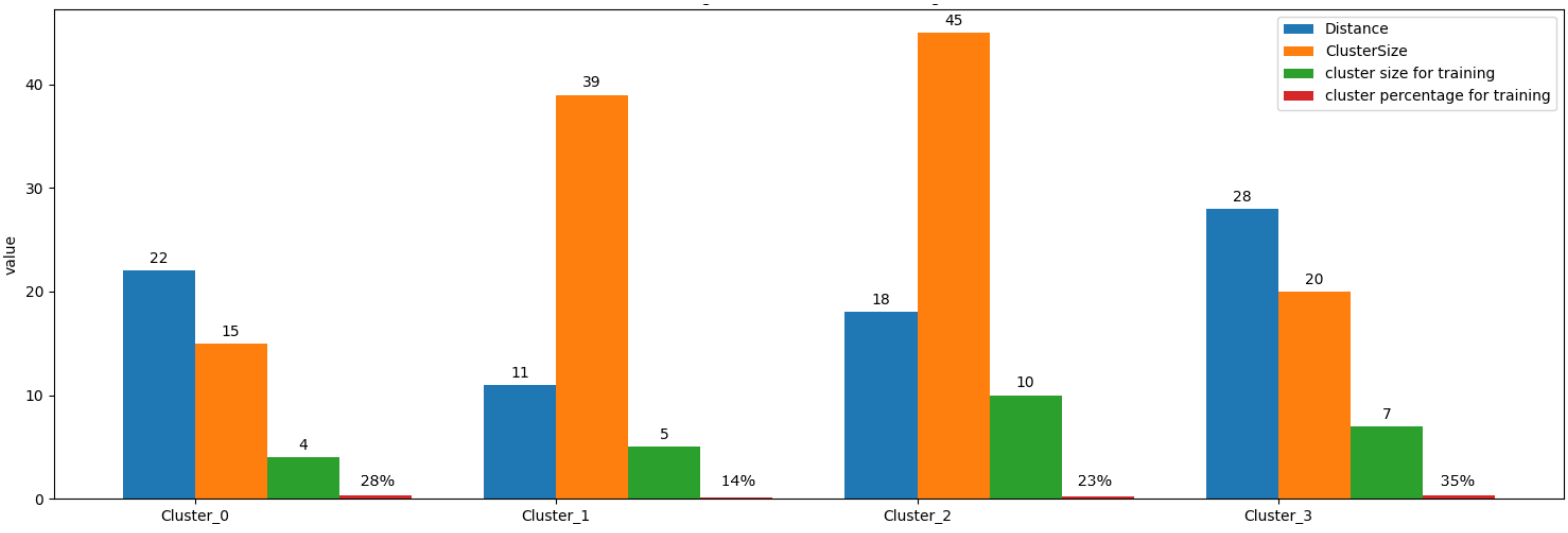
The FCDIS approach produces results that are both intuitive and insightful, as demonstrated in Figure 20, which uses the "Absence of Joint Cover Plate" of D^{BDI} as an illustrative example. A detailed examination of the clustering outcomes reveals a similarity among images grouped within the same cluster (Figure 20a).

Further insights are provided in Figure 20b, which summarizes the process of constructing the D_{Tr} by selecting images based on the distance and size characteristics of each cluster. For each cluster, the figure highlights several key metrics:

- Distance — The distance from the cluster center to the global center. Clusters with larger distances often embody less frequent features, which are particularly important for diversifying the training dataset. These clusters are given higher priority in the image selection process, ensuring that the training data reflects the full variability present in the feature space.
- Cluster size — Provides another layer of information, indicating the number of images contained within each cluster.
- Cluster size for training set — The number of images selected from each cluster for the training set is carefully determined using a distance ratio metric. This calculation ensures that clusters farther from the global center are weighted more heavily, allowing them to contribute a greater proportion of their data to the training subset.
- Cluster percentage for training set – This proportional selection strategy is further clarified by the percentage of each cluster's images that are chosen for training. By adjusting this percentage according to the cluster's distance and size, the FCDIS approach achieves a balanced yet diverse representation of the dataset.



(a)



(b)

Figure 20. Visualization of a few samples of each cluster for the "Absence of Joint Cover Plate" class from D_{Tr-Val}^{BDI}. a) depicts the images from clustering results; b) shows a summary of the images collected for D_{Tr}, considering each cluster's distance and size.

4.3 FEATURE SPACE CENTER-BASED IMAGE SELECTION

The FSCIS method introduces a novel strategy for dataset splitting, aiming to achieve a balanced and representative features distribution. The central idea of this approach is to systematically partition a given distance list into multiple sub-lists, enabling the selection of images located at the median positions within each sublist. This method ensures that the selected training data reflects the diversity and feature distribution of the entire dataset while mitigating biases that might arise from random or unstructured selection methods. The process (Figure 21a) begins by computing a sorted list of distances from each image's feature representation to the global center of the entire feature space. The global center serves as a reference point for measuring the relative positions of data points within the feature space. This distance-based organization lays the foundation for creating sub-lists (Figure 21b) that capture different ranges of feature distribution.

To accomplish this, the following pivotal variables must be considered: the designated size of the D_{Tr}– 80% of the D_{Tr-Val}, as proposed in this work — a sorted list of distances between each image's features and the global center of the entire feature space. The processing steps are the following: (i) images are first sorted by their distance to the feature space centroid; (ii) subgroups are then created; (iii) examples for D_{Tr} are iteratively selected based on the median element of each created subgroup until a percentage of 80% of the D_{Tr-Val} is reached; and, finally, (iv) the remaining images are assigned to D_{Val}. Black vertical lines illustrate the referred distance to the features' space centroid, white squares represent the not yet assigned images, and orange squares depict the images for D_{Tr} , and, finally, purple squares represent the images that are assigned for D_{Val}.

These variables are integrated into a structured image selection process designed to achieve a balanced and representative training dataset. This process involves a series of computations that ensure an even distribution of training data across the spectrum of feature distances relative to the global center. The first step in this process involves calculating a factor number that determines how the main distance list will be divided into evenly weighted subgroups. The goal is to create subgroups that, as closely as possible, contain the same number of elements. This division ensures that each subgroup captures a proportionate slice of the feature space, avoiding biased representation.

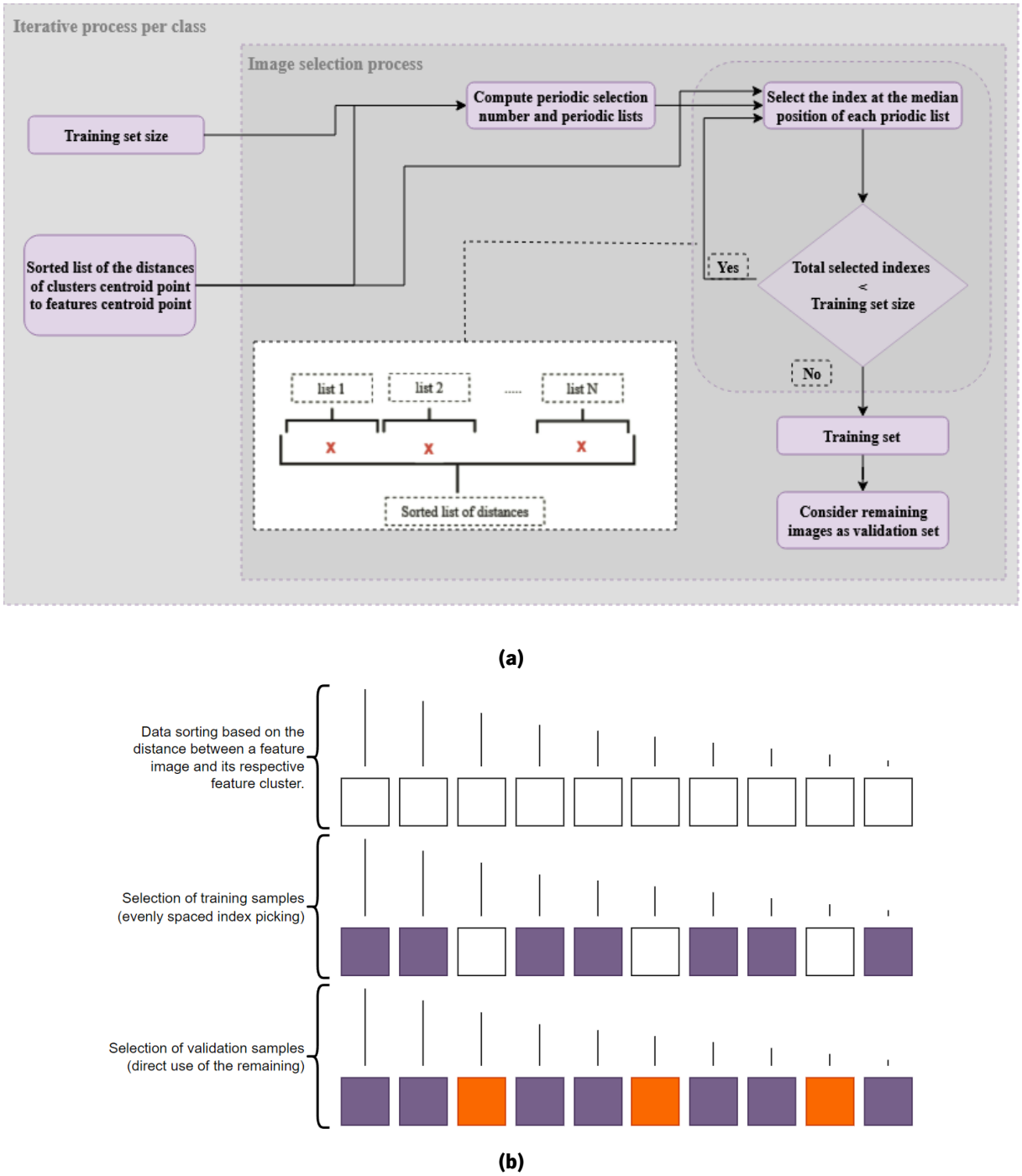


Figure 21. FSCIS process for setting up training and validation subsets: a) presents the main pipeline of the process; b) depicts the result of the pipeline behavior.

The computation for this factor is defined mathematically (as shown in Eq.6), ensuring a systematic and reproducible partitioning of the sorted list. Once the subgroups are formed, the next step is to identify the proper positions or indexes for image selection. Within each subgroup, the process focuses on locating the median-positioned image, which serves as a representative for that specific range of distances. This median selection is guided by the principle that centrally positioned elements in each subgroup offer a balanced representation of the features within that range. For each subgroup, the image at the median

index is selected and included in the training dataset (D_{Tr}). This selection process is repeated iteratively across all subgroups until the total number of images required for D_{Tr} is reached, ensuring that the training dataset is evenly sampled from the full spectrum of feature distances. This systematic sampling minimizes the risk of overemphasizing specific regions of the feature space while neglecting others, thereby promoting diversity and balance in the training subset. After the formation of D_{Tr} , the remaining images that were not selected during the median-picking operation are allocated to the validation subset (D_{Val}). This ensures that D_{Val} complements D_{Tr} , preserving the overall integrity and representativeness of the dataset while keeping the two subsets distinct.

$$\text{NumberOfGroups} = \left\lceil \frac{size\ of\ D_{Tr-Val}}{size\ of\ D_{Tr}} \right\rceil \quad (6)$$

4.4 Results Assessment

To assess the efficacy of the proposed feature-aware dataset-splitting methods in comparison to traditional dataset-splitting approaches, an extensive series of experiments was conducted. This evaluation utilized two distinct datasets, carefully chosen to test the robustness of the proposed methods across varied contexts. The experimental setup leveraged the Xception architecture combined with two optimizers: Nadam and SGD. This dual-optimizer approach was employed to ensure a balanced evaluation of optimization strategies and their interaction with dataset-splitting techniques.

The resulting models from these experiments underwent detailed analysis across three critical dimensions. The first area of focus was training consistency, assessed through the inspection of training plots that captured loss and accuracy metrics over 100 epochs. By carefully examining these plots, insights were gained into the stability of the learning process, the smoothness of convergence, and the early detection of overfitting. The second evaluation criterion was classification accuracy, which served as the primary quantitative metric for comparing the performance of different dataset-splitting methods. Accuracy on the test subset was recorded, providing clear benchmarks for model generalization. To deepen the interpretative aspect of the evaluation, activation analysis was conducted using Grad-CAM. This technique visualized the regions within the input data that the model prioritized during its decision-making process, offering a layer of transparency into the model's reasoning. The focus of this analysis was on the sharpness and specificity of the activation maps.

4.4.1 LEARNING CONSISTENCY ANALYSIS

After training the models, an in-depth analysis of the learning curves provided valuable insights into the behavior of the different dataset-splitting approaches. Notably, the training and validation accuracy and loss curves exhibited observable differences between models trained on datasets split using the proposed FCDIS and FSCIS methods, compared to those split through traditional dataset-splitting techniques.

The training and validation lines for FCDIS- and FSCIS-derived datasets appeared smoother and more stable, reflecting a consistent optimization process and reduced variability across epochs. By contrast, models trained on datasets split from traditional methods exhibited more pronounced fluctuations and inconsistencies in their learning curves, indicating potential issues such as class imbalance or feature representation gaps.

Figure 22 distinctly illustrates these observations through a set of training and validation accuracy-loss plots. These plots correspond to a training process of 100 epochs performed over datasets from two distinct imagery collections: D^{BDI} and D^{GLI} . Figure 22a highlights the learning curve for KFCV10-fold0, where training concluded at epoch 71. Similarly, Figure 22b displays the curve for KFCV10-fold7, where early stopping occurred at epoch 48.

These variations highlight the implementation of an early stopping callback mechanism, a vital aspect of the training configuration discussed in the prior section. The very early stopping observed in the KFCV10-fold0 and KFCV10-fold7 strongly indicates the effectiveness of the datasets prepared using FCDIS and FSCIS approaches.

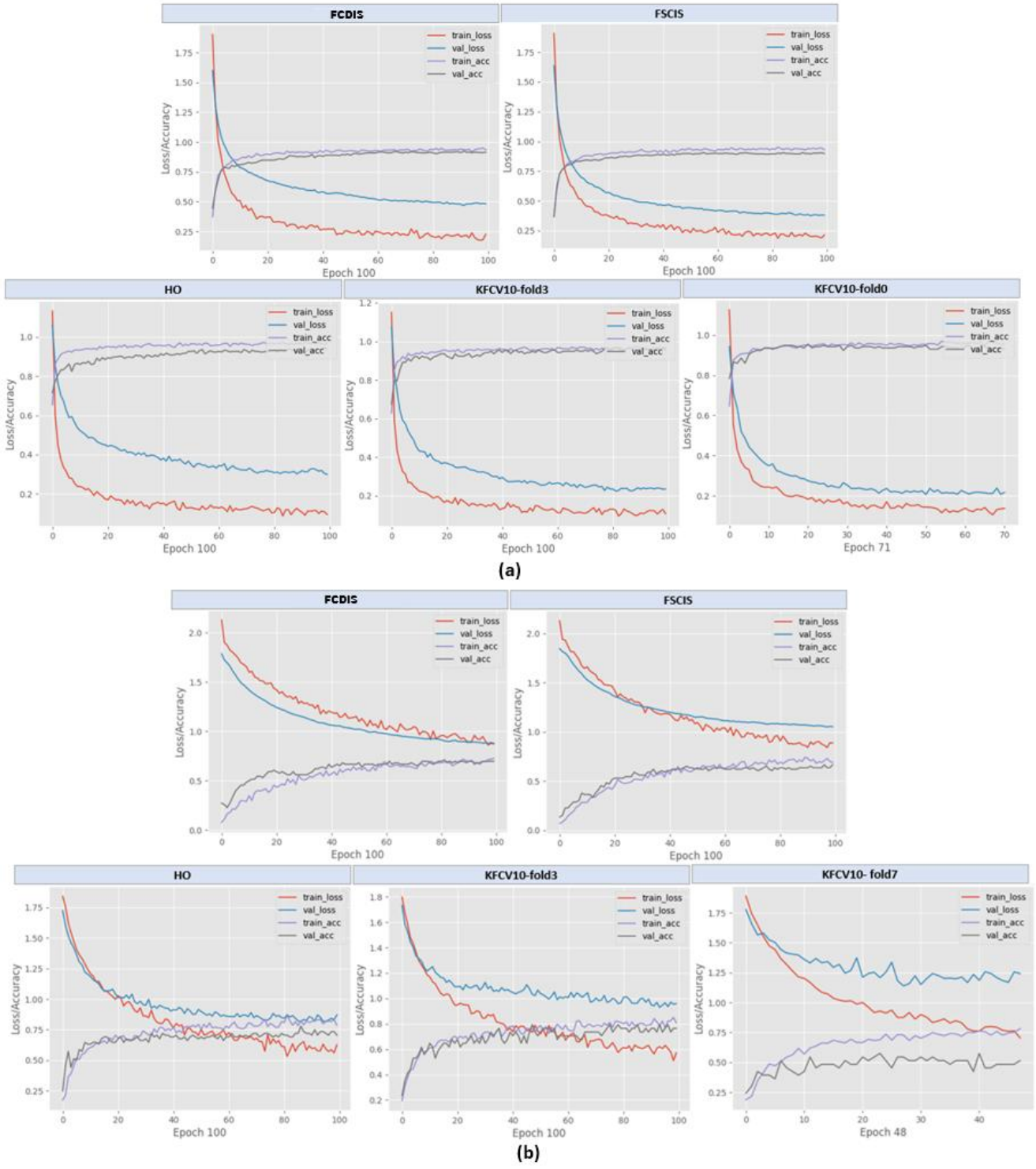


Figure 22. Learning curves, comparing the behaviour of models' learning over datasets produced based on traditional techniques and FCDIS/FSCIS methods: a) Xception Nadam for $D_{Tr,Val}^{BDI}$; b) Xception Nadam for $D_{Tr,Val}^{GLI}$.

The relevance of such training progress control mechanisms has been previously substantiated (Chapter 3), which emphasized their role in achieving optimized model performance.

4.4.2 ASSESSMENT USING TEST SET

Based on D^{BDI} and D^{GLI} , experiments were conducted to evaluate the effectiveness of the proposed feature-aware dataset splitting techniques, FCDIS and FSCIS, against traditional methods (HO and KFCV10).

The evaluation was performed using dedicated D_{Tst}^{BDI} and D_{Tst}^{GLI} reserved exclusively for assessing model performance. These testing datasets were structured to ensure unbiased evaluation and consistent comparison across all splitting methods. The results of these experiments, detailed in Tables 10 and 11, reveal notable patterns in model performance. The proposed FCDIS and FSCIS approaches consistently outperformed traditional splitting techniques, achieving higher accuracy and enhancing learning stability. A significant observation was that models trained on datasets derived from FCDIS and FSCIS completed the full 100 epochs of training, while several models trained using traditional methods — particularly those using KFCV10 and HO — terminated early due to the implementation of an early stopping mechanism.

Table 10. Accuracy results for models trained with D_{Tst}^{BDI}. "Acc" refers to classification accuracy (%). ↑, ↓ and = indicate, respectively, if HO and KFCV are better, worse, or equal to the best proposed splitting techniques(FCDIS/FSCIS).

Dataset Splitting Approaches		Xception			
		Nadam		SGD	
		Acc	ESE	Acc	ESE
FCDIS		**90%**	100	**86%**	100
FSCIS		88%	100	81%	100
HO		89%↓	100	82%↓	100
KFCV10	Fold 0	89%↓	71	**86%(=)**	100
	Fold 1	88%↓	100	84%↓	100
	Fold 2	87%↓	100	82%↓	100
	Fold 3	**90%(=)**	100	81%↓	100
	Fold 4	88%↓	100	82%↓	100
	Fold 5	88%↓	100	85%↓	100
	Fold 6	89%↓	100	83%↓	100
	Fold 7	89%↓	100	82%↓	100
	Fold 8	88%↓	89	84%↓	100
	Fold 9	88%↓	100	83%↓	100
	Mean KFCV10	88%↓		73%↓	

On the bridge defect dataset (D_{Tst}^{BDI}), as shown in Table 10, models trained with FCDIS and FSCIS demonstrated superior accuracy compared to those trained with HO and surpassed the majority of KFCV10 folds.

Similarly, Table 11 illustrates the performance of the grapevine variety dataset (D_{Tst}^{GLI}), where FCDIS and FSCIS once again achieved better results than the traditional approaches.

Table 11. Accuracy results for models trained with datasets D_{Tst}^{GLI}. "Acc" refers to classification accuracy (%). ↑, ↓ and = indicate, respectively, if HO and KFCV are better, worse, or equal to the best proposed splitting techniques(FCDIS/FSCIS).

Dataset Splitting Approaches		Xception			
		Nadam		SGD	
		Acc	ESE	Acc	ESE
FCDIS		70%	100	67%	100
FSCIS		**76%**	100	**74%**	100
HO		75%↓	100	69%↓	100
KFCV10	Fold 0	70%↓	100	67%↓	100
	Fold 1	65%↓	100	70%↓	100
	Fold 2	70%↓	98	68%↓	100
	Fold 3	74%↓	100	71%↓	100
	Fold 4	68%↓	94	68%↓	100
	Fold 5	68%↓	100	65%↓	100
	Fold 6	65%↓	92	67%↓	100
	Fold 7	61%↓	48	60%↓	100
	Fold 8	65%↓	84	67%↓	100
	Fold 9	67%↓	100	71%↓	100
	Mean KFCV10	67%↓		67%↓	

In addition to dataset splitting methods, the impact of optimizers on model performance was also examined. The experiments employed Nadam and SGD optimizers to train models across all splitting approaches. The results revealed that models trained with Nadam consistently outperformed those trained with SGD. Nadam's adaptive nature, which integrates the benefits of Nesterov momentum and Adam optimization, contributed to more effective convergence and superior generalization. This was particularly evident for models trained on datasets derived from FCDIS and FSCIS, where the synergy between advanced splitting techniques and an adaptive optimizer like Nadam produced optimal results.

4.4.3 Assessment with External Imagery Sets

An extended evaluation was conducted to further assess the robustness and generalization capabilities of the trained models. This evaluation utilized external testing datasets, $D_{Ext_Tst}^{BDI}$ and $D_{Ext_Tst}^{GLI}$. These datasets were designed to align directly with the class structures of the original training datasets while introducing additional variability to test the models' adaptability (as detailed explained in Chapter 3). The performance of the trained models on these external datasets is detailed in Table 12 for $D_{Ext_Tst}^{BDI}$ and Table 13 for $D_{Ext_Tst}^{GLI}$.

Table 12. Accuracy of models using the $D_{Ext_Tst}^{BDI}$. "Acc" refers to classification accuracy (%). ↑, ↓ and = indicate, respectively, if HO and KFCV are better, worse, or equal to the best proposed splitting techniques(FCDIS/FSCIS).

Dataset Split Approaches		Xception	
		Nadam	SGD
		Acc	Acc
FCDIS		**75%**	68%
FSCIS		**75%**	**72%**
HO		72%↓	70%↓
KFCV10	Fold 0	72%↓	73%↓
	Fold 1	72%↓	**77%↑**
	Fold 2	77%↓	70%↓
	Fold 3	73%↓	73%↓
	Fold 4	68%↓	75%↑
	Fold 5	**77%↑**	67%↓
	Fold 6	75%↓	75%↑
	Fold 7	75%↑	70%↓
	Fold 8	73%↓	70%↓
	Fold 9	72%↓	68%↓
	Mean KFCV10	73%↓	72%(=)

The results highlight the differences in generalization between models trained using the proposed splitting methodologies (FCDIS and FSCIS) and those trained using traditional techniques such as HO and KFCV10.

In the case of $D_{Ext_Tst}^{BDI}$, as shown in Table 12, the models trained with the proposed FCDIS and FSCIS splitting methods consistently outperformed those trained with HO. For example, the FCDIS- and FSCIS-based Xception/Nadam models demonstrated superior accuracy, either matching or surpassing 8 out of the 10 KFCV10-based models.

For the FSCIS-based Xception/SGD models, the performance, while still superior to HO and half of the KFCV10-based models, was less pronounced compared to the Nadam-optimized models. This highlights the interplay between dataset-splitting strategies and optimizer selection, with Nadam's adaptive LR and momentum-based updates appearing to be better suited to the data characteristics introduced by FSCIS.

The evaluation of $D_{Ext_Tst}^{GLI}$, detailed in Table 13, revealed similar trends. The FCDIS-based models outperformed the HO approach and the majority of the KFCV10-based models, significantly surpassing them across both Xception/Nadam and Xception/SGD configurations. However, the performance of FSCIS-based models on $D_{Ext_Tst}^{GLI}$ was more variable. While they consistently outperformed HO, their advantage over KFCV10-based models was less consistent. The Xception/Nadam model achieved moderate success, while the Xception/SGD model outperformed only a few KFCV10 counterparts. This

suggests that while FSCIS remains a strong candidate for dataset splitting, its performance may depend on specific characteristics of the dataset and the chosen optimizer.

Table 13. Accuracy of models using the $D_{Ext_Tst}^{GLI}$. "Acc" refers to classification accuracy (%). ↑, ↓ and = indicate, respectively, if HO and KFCV are better, worse, or equal to the best proposed splitting techniques(FCDIS/FSCIS).

Dataset Split Approaches		Xception	
		Nadam	SGD
		Acc	Acc
FCDIS		**32%**	**27%**
FSCIS		25%	22%
HO		22%↓	20%↓
KFCV10	Fold 0	25%↓	25%↓
	Fold 1	25%↓	22%↓
	Fold 2	25%↓	28%↑
	Fold 3	23%↓	28%↑
	Fold 4	32%↓	28%↑
	Fold 5	23%↓	**27%(=)**
	Fold 6	23%↓	23%↓
	Fold 7	25%↓	25%↓
	Fold 8	27%↓	23%↓
	Fold 9	25%↓	30%↑
	Mean KFCV10	25%↓	26%↓

4.4.4 Attention Mechanisms Assessment with External Imagery Sets

Grad-CAM-based IoU provides critical insights into the spatial accuracy and localization precision of model predictions. By highlighting the regions of an image that the model focuses on while making decisions, Grad-CAM enhances interpretability, aligning with the objectives of XAI. This approach is pivotal for understanding the inner workings of DL models and identifying avenues for performance improvement. To facilitate a detailed Grad-CAM analysis, a series of preparatory steps was implemented. First, the Grad-CAM-generated areas were identified for each sample image. These areas, marked in red, represent the most salient regions as determined by the model during its decision-making process. Next, corresponding ground-truth regions were annotated for all images involved in the assessment. This step was crucial for establishing a reliable reference point for comparison. Figure 23 illustrates an example of Grad-CAM-generated attention areas side by side with manually annotated ground-truth masks, providing a clear visual representation of the regions under consideration.

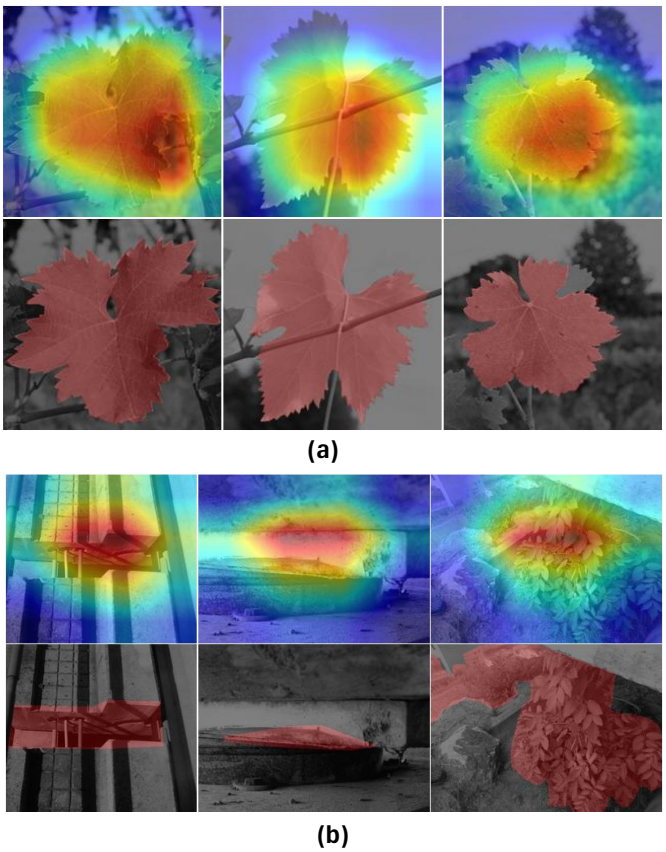


Figure 23. Visualization of the Grad-CAM evaluation approach. a) and b) show the saliency maps over $D_{Ext_Tst}^{GLI}$ and $D_{Ext_Tst}^{BDI}$, respectively. In each one, the first row corresponds to the Grad-CAM visualization, while the second row presents the ground-truth.

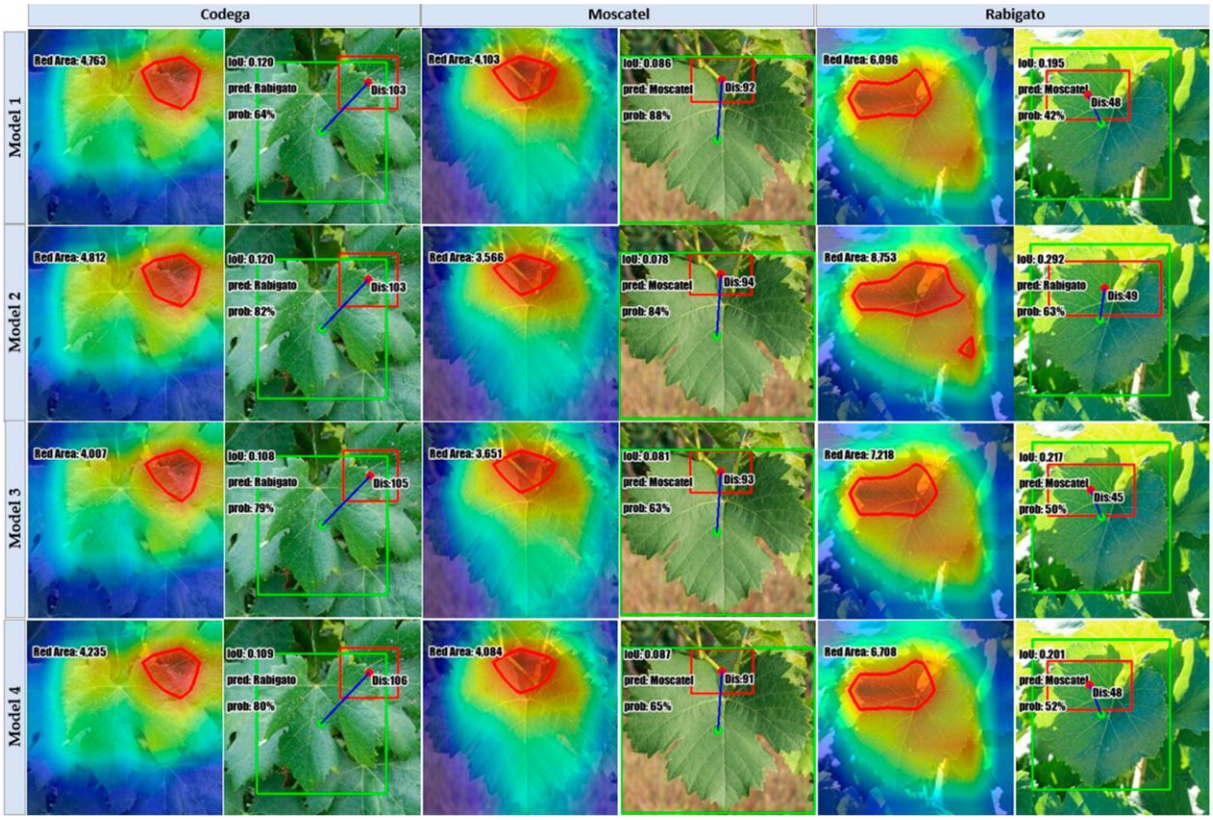
The analysis involved computing several key parameters for each image. These included:

- Displacement Between Centers (Dis) — The distance between the centroids of the Grad-CAM-generated bounding box and the annotated ground-truth region was calculated, offering a quantitative measure of spatial alignment.
- Classification Accuracy (Acc) — This parameter validated the correctness of the model's predictions, confirming whether the focused attention areas contributed to accurate classifications.
- Intersection over Union (IoU) — The measured overlap between the Grad-CAM-generated bounding box and the annotated ground-truth region, quantifying the model's localization precision.
- Grad-CAM Size (GCS) — The size of the salient region identified by Grad-CAM was recorded to assess the extent of the model's focus within an image.

The Grad-CAM analysis was conducted using the top-performing models identified in the earlier classification evaluations. These models, trained using FSCIS, FCDIS, HO, and the best-performing

KFCV10-based approach, were applied to both external test datasets: $D^{GLI}_{Ext_Tst}$ and $D^{BDI}_{Ext_Tst}$. Figures 24 and 25 present representative outcomes of the Grad-CAM analysis for these datasets, respectively. Each figure includes examples of Grad-CAM overlays on test images, accompanied by text summarizing the key parameters (Dis, Acc, IoU, and GCS) for each model. In the context of $D^{GLI}_{Ext_Tst}$, the FSCIS- and FCDIS-based models exhibited superior spatial precision and consistency. Their Grad-CAM overlays showed highly localized attention on relevant areas, with smaller centroid displacements and higher IoU scores compared to models trained with traditional HO and KFCV10 approaches. Similarly, for $D^{BDI}_{Ext_Tst}$, the FSCIS-based model emerged as a standout performer. Its Grad-CAM outputs are consistently aligned with ground-truth annotations, achieving higher IoU scores and minimal centroid displacement. While the FCDIS-based model also demonstrated strong results, its performance was slightly less consistent compared to FSCIS. Models trained with HO and KFCV10 exhibited broader attention areas and lower localization accuracy, indicating limitations in their training dataset's representativeness.

In both Figures 24 and 25, Model 1, Model 2, Model 3, and Model 4 correspond, respectively, to the models trained with $D^{GLI}_{Ext_Tst}$ and $D^{BDI}_{Ext_Tst}$ variants split using FCDIS, FSCIS, HO, and KFCV10. In the first column, the red rectangle represents the most distinguished attention area determined through Grad-CAM, while the green rectangle corresponds to the hand-crafted ground-truth.



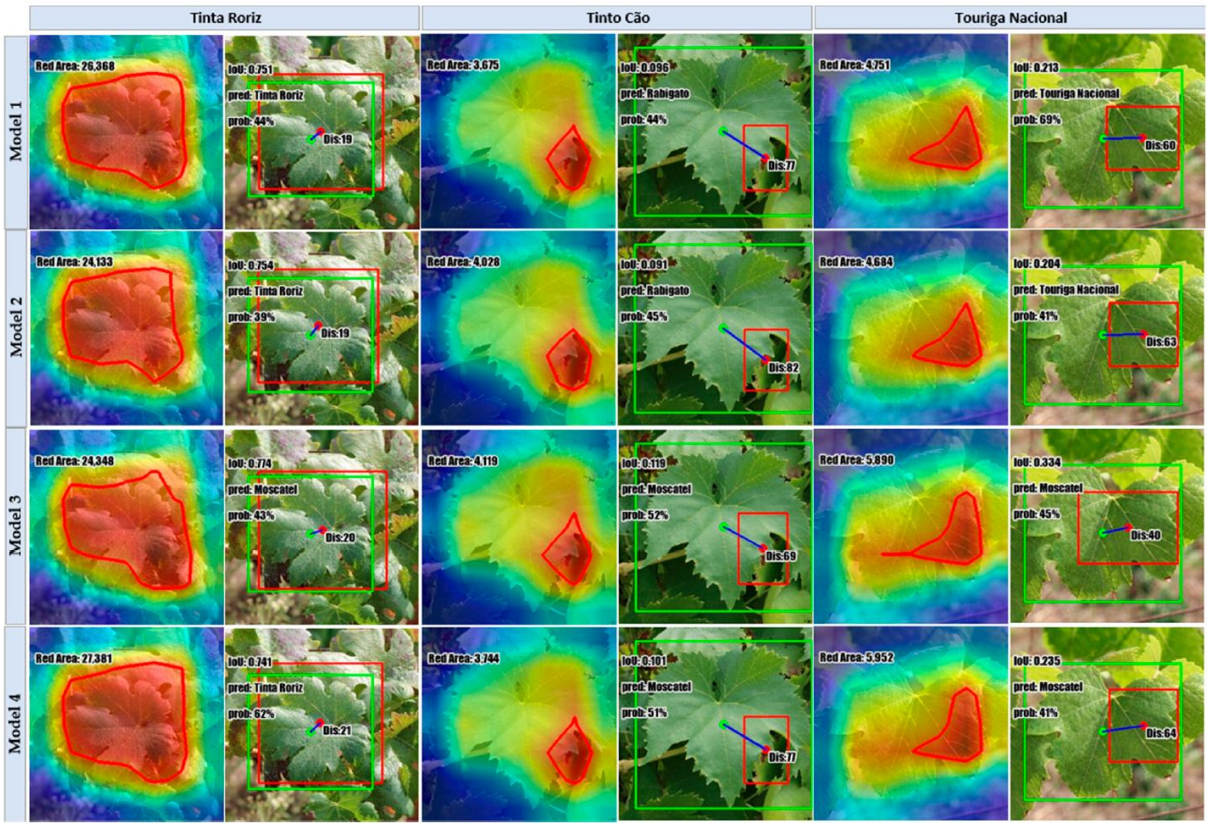
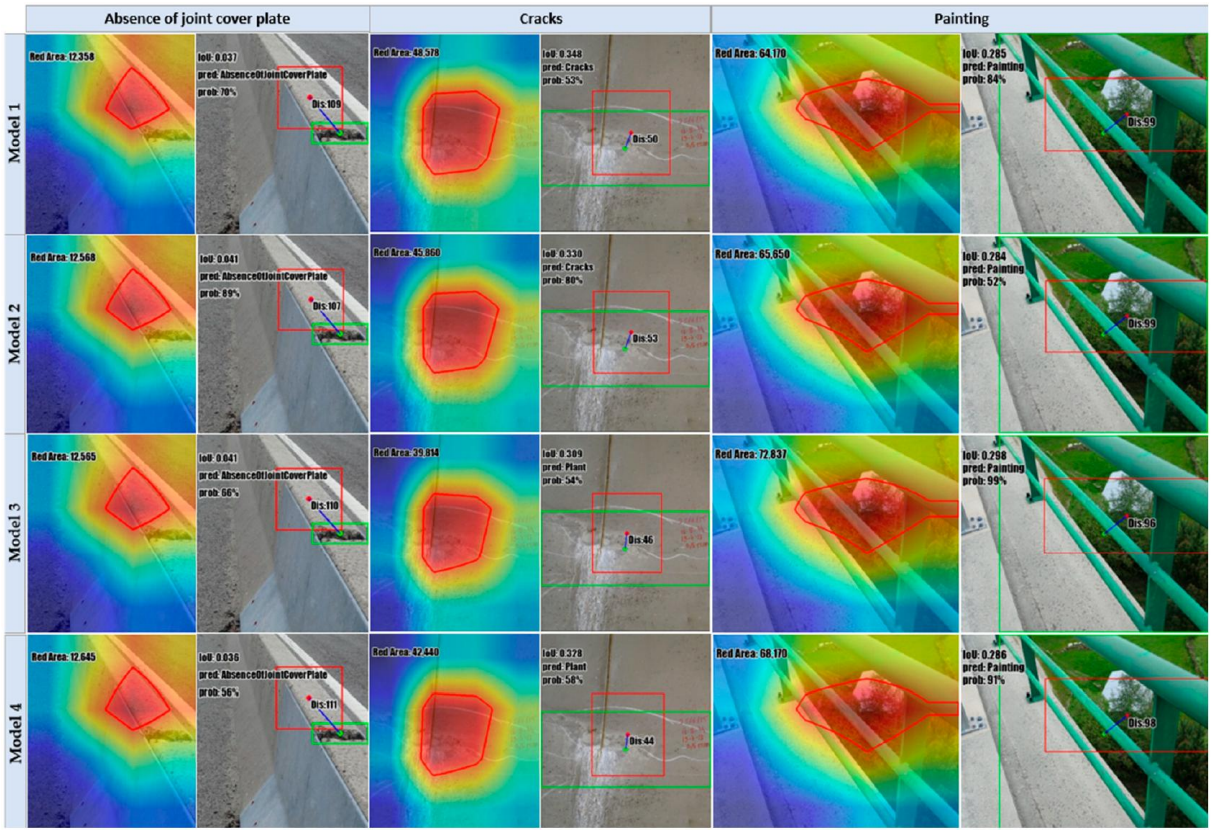


Figure 24. Attention visualization for $D_{Ext_Tst}^{GLI}$ considering the top Xception/Nadam model trained with the splitting techniques.



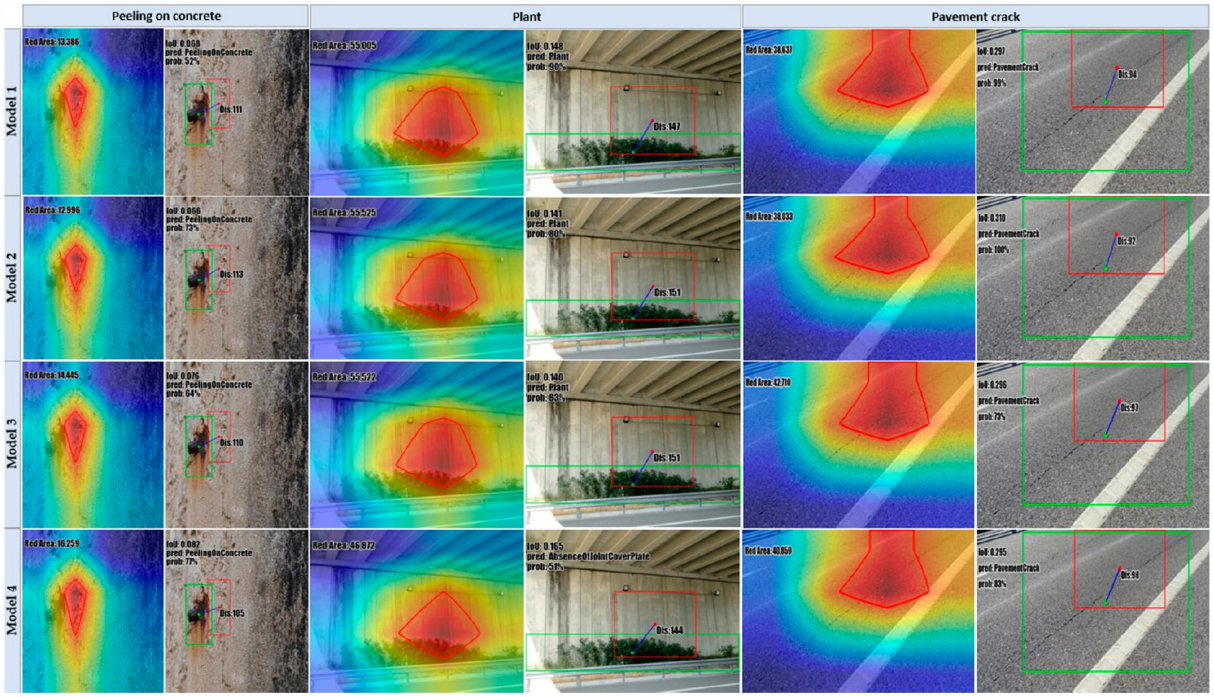


Figure 25. Attention visualization for $D_{Ext_Tst}^{BDI}$ considering the top Xception/Nadam model trained with the splitting techniques.

To analyze the models' performance from a broader perspective, mean values of the key parameters were calculated for each class within both datasets, including mean values of accuracy (μAcc), IoU (μIoU), Grad-CAM size (μGCS), and displacement (μDis). This generalization provides an insightful overview of the models' capabilities across various categories, avoiding the noise that can arise from individual image-level analyses. The results, summarized in Tables 14 and 15, present the performance metrics for the Xception/Nadam models, providing a clear comparative view of the effectiveness of the proposed dataset-splitting approaches — FCDIS and FSCIS — in relation to traditional techniques. Table 14 illustrates the mean μAcc, μIoU, μGCS, and μDis for the $D_{Ext_Tst}^{GLI}$. Among the analyzed approaches, the FCDIS-based model emerged as the top performer, achieving a μAcc of 38%, a μIoU of 19%, a μGCS of 7220px, and a μDis of 68px. These metrics highlight the FCDIS's ability to effectively focus the model's attention, enabling precise localization and robust classification outcomes. The FSCIS-based model also demonstrated strong results, outperforming the traditional HO and KFCV10-based methods by a significant margin.

Similarly, Table 15 presents the results for the $D_{Ext_Tst}^{BDI}$. The FSCIS- and FCDIS-based models exhibited nearly identical performance, with both methods achieving a μAcc of 75% and a μIoU of 18%. The μGCS for FSCIS and FCDIS were 37,604px and 37,538px, respectively, with minor differences in displacement (μDis = 135px for FSCIS and μDis = 134px for FCDIS). These results reaffirm the robustness of the

proposed dataset-splitting methodologies, as both approaches consistently outperformed the traditional HO and KFCV10 techniques.

Table 14. Prediction results upon the $D_{Ext_Tst}^{GLI}$ for Xception Nadam.

Class	FCDIS				FSCIS				HO				KFCV10 - fold3			
	µAcc	µIoU	µGCS	µDis	µAcc	µIoU	µGCS	µDis	µAcc	µIoU	µGCS	µDis	µAcc	µIoU	µGCS	µDis
Codega	0%	12%	4393	85	0%	13%	4880	82	0%	10%	3541	87	0%	11%	4139	86
Moscatel	90%	17%	7349	69	50%	16%	6311	72	70%	14%	5396	75	0%	15%	4996	70
Rabigato	60%	19%	7678	67	80%	19%	7478	70	50%	15%	6099	76	70%	15%	6000	72
Tinta Roriz	20%	24%	8231	63	10%	23%	7731	62	10%	24%	7956	62	0%	24%	8526	62
Tinto Cao	0%	21%	7961	58	0%	23%	8911	58	0%	16%	6467	62	10%	15%	6415	65
Touriga Nacional	20%	23%	7707	67	10%	21%	7652	70	0%	28%	8784	63	0%	22%	7915	70
Total mean	**38%**	**19%**	7220	68	**25%**	**19%**	7160	69	22%	18%	6374	71	13%	17%	6332	71

Table 15. Prediction results for the proposed dataset splits for the Xception model using Nadam optimizer upon $D_{Ext_Tst}^{BDI}$. *AJCP - Absence of Joint Cover Plate; PC - Pavement Crack; PoC - Peeling on Concrete.*

Class	FCDIS				FSCIS				HO				KFCV10 - fold3			
	µAcc	µIoU	µGCS	µDis	µAcc	µIoU	µGCS	µDis	µAcc	µIoU	µGCS	µDis	µAcc	µIoU	µGCS	µDis
AJCP	90%	18%	42969	139	90%	20%	48068	137	80%	19%	46502	140	80%	18%	45516	141
Cracks	70%	21%	39203	126	70%	21%	38771	129	40%	18%	38429	132	60%	20%	37225	127
Painting	50%	20%	36071	124	50%	18%	32077	117	90%	20%	33226	107	70%	21%	36871	118
PC	90%	16%	27457	149	90%	16%	26962	148	90%	15%	28107	147	100%	15%	28070	147
PoC	50%	17%	30665	125	50%	17%	30473	126	50%	17%	31801	125	50%	18%	31065	124
Plant	100%	17%	49259	144	100%	16%	48878	146	50%	16%	48549	145	80%	17%	49576	145
Total mean	**75%**	**18%**	37604	135	**75%**	**18%**	37538	134	67%	18%	37769	133	73%	18%	38054	134

To sum up, this chapter introduced two dataset–splitting strategy proposals: FCDIS and FSCIS. These strategies were designed to overcome the limitations of traditional approaches by leveraging automatic feature-aware clustering and centroid-informed sampling. These methods were evaluated using DL models trained on datasets derived from two distinct imagery collections.

One of the key outcomes of this study was the superior training stability observed in models trained on datasets split using FCDIS and FSCIS. These models exhibited smoother and more consistent learning curves compared to those trained on datasets derived from traditional methods, indicating the suitability of the proposed splitting approaches for DL tasks. Early stopping, employed to monitor training progress, further highlighted the efficiency of the proposed methods by ensuring that models achieved optimal performance in a timely manner without overfitting.

In terms of classification accuracy, the models trained on datasets split using FCDIS and FSCIS consistently outperformed their counterparts trained with traditional splitting methods. Across both the first test datasets (D_{Tst}^{BDI} and D_{Tst}^{GLI}) and external test datasets ($D_{Ext_Tst}^{BDI}$ and $D_{Ext_Tst}^{BDI}$), the proposed methods achieved higher accuracy, better IoU scores, and more precise attention localization. Notably, the use of Grad-CAM analyses demonstrated the spatial precision of the models, revealing that the proposed methods enabled models to focus more effectively on relevant regions within the imagery, leading to more accurate predictions.

Furthermore, a deeper analysis of key metrics such as classification accuracy, IoU, Grad-CAM size, and displacement between predicted and ground-truth centroids revealed that higher IoU scores correlated with higher classification accuracy. This finding underscores the importance of precise attention localization in improving overall model performance, emphasizing the critical role of spatial awareness in DL tasks.

Overall, the results of this study highlight the transformative potential of FCDIS and FSCIS in addressing challenges associated with dataset splitting, particularly in scenarios involving complex and high-dimensional datasets. These methodologies have been shown to be promising in the improvement of training efficiency and model performance.

While this chapter focused on dataset structuring before training, the following chapter introduces a complementary methodology — MbCL — that shifts the attention toward continuous, performance-guided datasets during the learning process. By integrating evaluation metrics into an iterative feedback loop, MbCL provides a dynamic technique for further improving data quality and model outcomes, completing the proposed data-centric learning pipeline.

5 Proposed Metrics-based Closed-Loop Methodology

In DL model development, effective training depends on adaptive and dynamic approaches that not only evaluate the model's performance in real-time but also promote adjustments based on an iterative feedback mechanism to optimize learning processes. The development of robust and accurate DL models heavily relies on the quality of training processes, which implies resorting to appropriate datasets [347]. When confronted with challenges like class imbalance, underrepresentation, and noisy data [348], [349], the models often end up under their performance potential. On the other hand, the traditional development of DL models does not consider dataset-oriented adjustments during the training process, with eventual penalties for their generalization [350], [351]. In contrast, a closed-loop methodology offers a dynamic, adaptive framework by integrating real-time performance evaluations, enabling informed adjustments that enhance model accuracy and robustness [287], [288].

Worth mentioning that in real-world scenarios, datasets are often limited in size and representativeness symmetries, leading to biased models with poor generalization, especially for minority classes[278]. Traditional data augmentation techniques [279] , such as rotations, flips, and color adjustments, have been widely used to increase dataset diversity, but they typically apply transformations uniformly without addressing feature imbalances. More advanced methods like AA [177] automate the augmentation process but remain static, consistently applying the same policies throughout training. GANs offer an alternative by creating synthetic samples, effectively enhancing minority class representation and improving overall model robustness. Beyond static augmentation strategies, dynamic closed-loop approaches, such as RL-Augment, have been introduced to adapt augmentation policies during training. DRL [266] has also gained attention for optimizing augmentation processes but faces challenges related to sample efficiency and generalization. Despite these advancements, many existing methods focus on improving overall model performance rather than dataset-oriented class-specific improvements, sometimes leading to over-representation of dominant classes and potential oversampling issues [352]. This imbalance can negatively affect the model's learning process and limit its real-world applicability.

To address these limitations, this chapter introduces a proposed MbCL methodology designed to optimize model training through iterative validation-based feedback. Unlike conventional workflows that primarily depend on pre-trained yet adjustable configurations — such as hyperparameter tuning and architectural structure adjustments — alongside static datasets, this approach actively utilizes insights from specific performance metrics relying on a feedback mechanism that monitors class-wise validation and applies outlier detection techniques to identify underperforming classes. Next, such underperforming classes are strategically augmented to enhance their representativeness in the training process, ensuring

a more balanced and equitable learning process. Augmentations of flagged classes are inspired by two techniques: conventional data augmentation and GAN-based augmentation. Conventional augmentation involves applying transformations such as rotations, translations, scaling, and flipping to generate variations of existing data. These transformations are particularly effective in increasing the dataset's diversity without altering the fundamental characteristics of the outlier class. On the other hand, GAN-based augmentation is employed to synthesize entirely new data samples. By using the adversarial training process of GANs, the methodology creates realistic and contextually relevant images that encapsulate the complexities of the underperforming classes. This expansion provides the model with richer representations, enabling it to learn more effectively even in the presence of complex or noisy datasets. Additionally, the use of GANs reduces the reliance on extensive manual data collection efforts, offering a scalable solution for dataset enrichment. Then these augmented samples are integrated into subsequent training cycles, allowing the model to refine its understanding of the problematic classes. This iterative process continues until validation metrics indicate that the performance of the outlier classes aligns with the overall accuracy of the model using an early-stopping criterion. This aims not only to improve the performance of underrepresented classes but also to enhance the model's overall generalization capability, making it more robust in real-world applications.

5.1 MbCL Approach Proposal: A Detailed Perspective

The MbCL approach consists of an iterative method aimed at enhancing the model training process by leveraging continuous feedback and evaluation, as well as by addressing data representativeness in line with the training. More specifically, this methodology incorporates a loop management strategy that allows for real-time dataset-target adjustments during the training process, responsive to class-wise performance outcomes. This feedback-driven approach ensures that representativeness deficiencies are identified and addressed before the models' training conclusion, intending to attain a more effective model convergence and an overall improved performance.

Focusing on dataset arrangement, MbCL integrates several crucial processes, including feature extraction and dimensionality reduction, which are designed to ensure an effective inspection of the data. Following this, the FCDIS technique — highlighted in Chapter 4 — is applied to facilitate the splitting of the dataset, enabling an optimal allocation of images between the training and validation sets.

First, the dataset undergoes a sequential splitting process to ensure proper distribution for model evaluation and training. Initially, the dataset is divided into two primary subsets: 20% for the test set and 80% for the training-validation set (Figure 10). This initial split ensures that a significant portion of the data

is reserved for testing the model's final performance while retaining the majority for training and validation tasks. Subsequently, the training-validation subset (D_{Tr-Val}) is further partitioned using the FCDIS method (Figure 26). This advanced splitting technique employs feature clustering to ensure a balanced and representative distribution of the data. Through this process, the D_{Tr-Val} is split into 20% for D_{Val} and 80% for D_{Tr}.

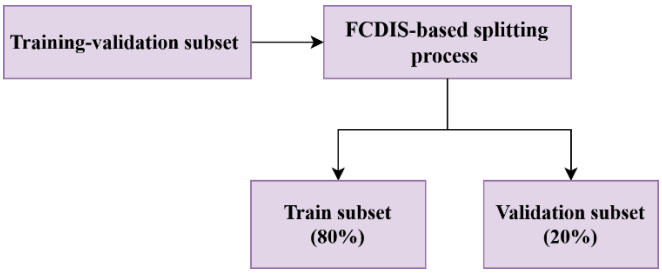


Figure 26. Overview of the dataset preparation workflow.

The training process is structured as a closed-loop system, emphasizing adaptability and continuous refinement to achieve optimal model performance. Central to this process is the monitoring of key performance metrics, with accuracy serving as a primary indicator of the model's learning progress and effectiveness.

At each iteration, the validation-based model's overall accuracy and class-wise accuracy are systematically tracked to assess its generalization ability and predictive performance. When deviations from the expected model's overall performance are detected, specific adjustments are made to the training and validation subsets. These adjustments involve augmenting the data to enhance diversity and representation. This dynamic process ensures that the learning trajectory stays aligned with the desired objectives, aiming to advance an under-progress model to be more robust and capable of generalizing. After the training stage, the model undergoes an applied evaluation to validate its performance. Two key datasets are employed in this evaluation process: D_{Tst} (test set) and $D_{\mathrm{Ext_Tst}}$ (external test set). This comprehensive training and evaluation workflow is visually summarized in Figure 27, illustrating the interplay between iterative refinement during training and thorough post-training assessments.

The following subsections provide a detailed explanation of the methodologies and strategies implemented during the MbCL training phase, focusing on the closed-loop process, how many images need to be generated per class, and a criterion for efficient augmentation management to avoid redundancy.

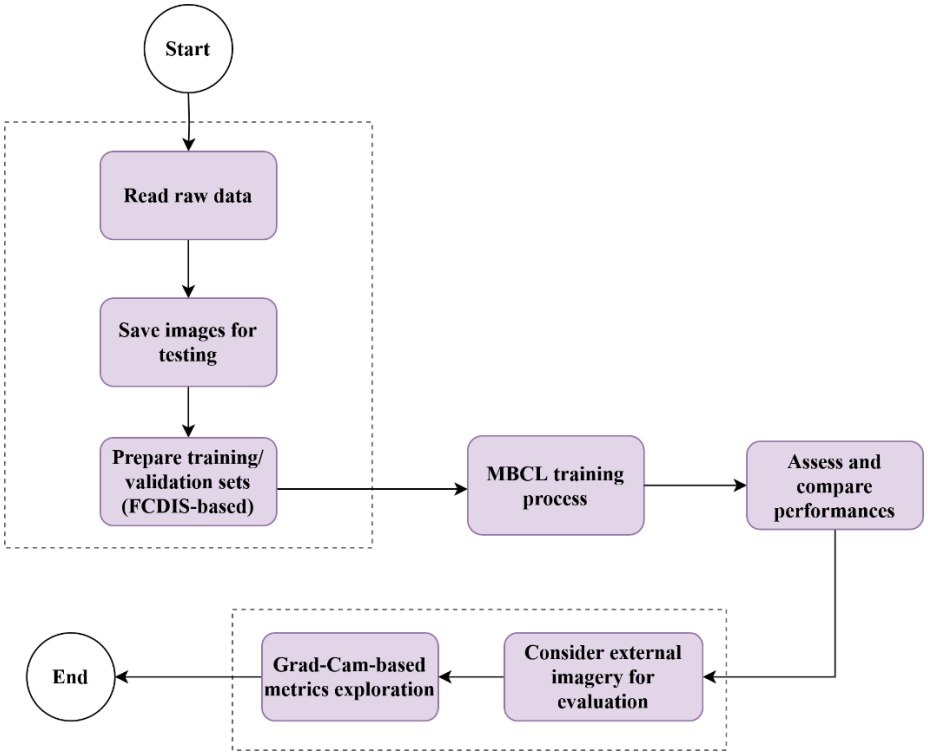


Figure 27. Overview of the methodology workflow.

5.1.1 MbCL Training Process

The MbCL training process consists of an iterative and adaptive methodology specifically designed to refine and optimize model performance through continuous monitoring, feedback, and adjustment. This approach ensures that the model not only learns effectively from the training data but also addresses deficiencies in a structured manner, ultimately achieving high robustness and accuracy across all classes. Unlike static training paradigms, the closed-loop methodology emphasizes the dynamic evolution of the model, incorporating regular checkpoints for performance evaluation and targeted interventions to maintain a balanced learning process. The various stages of this methodology are interconnected, forming a workflow, as illustrated in Figure 28.

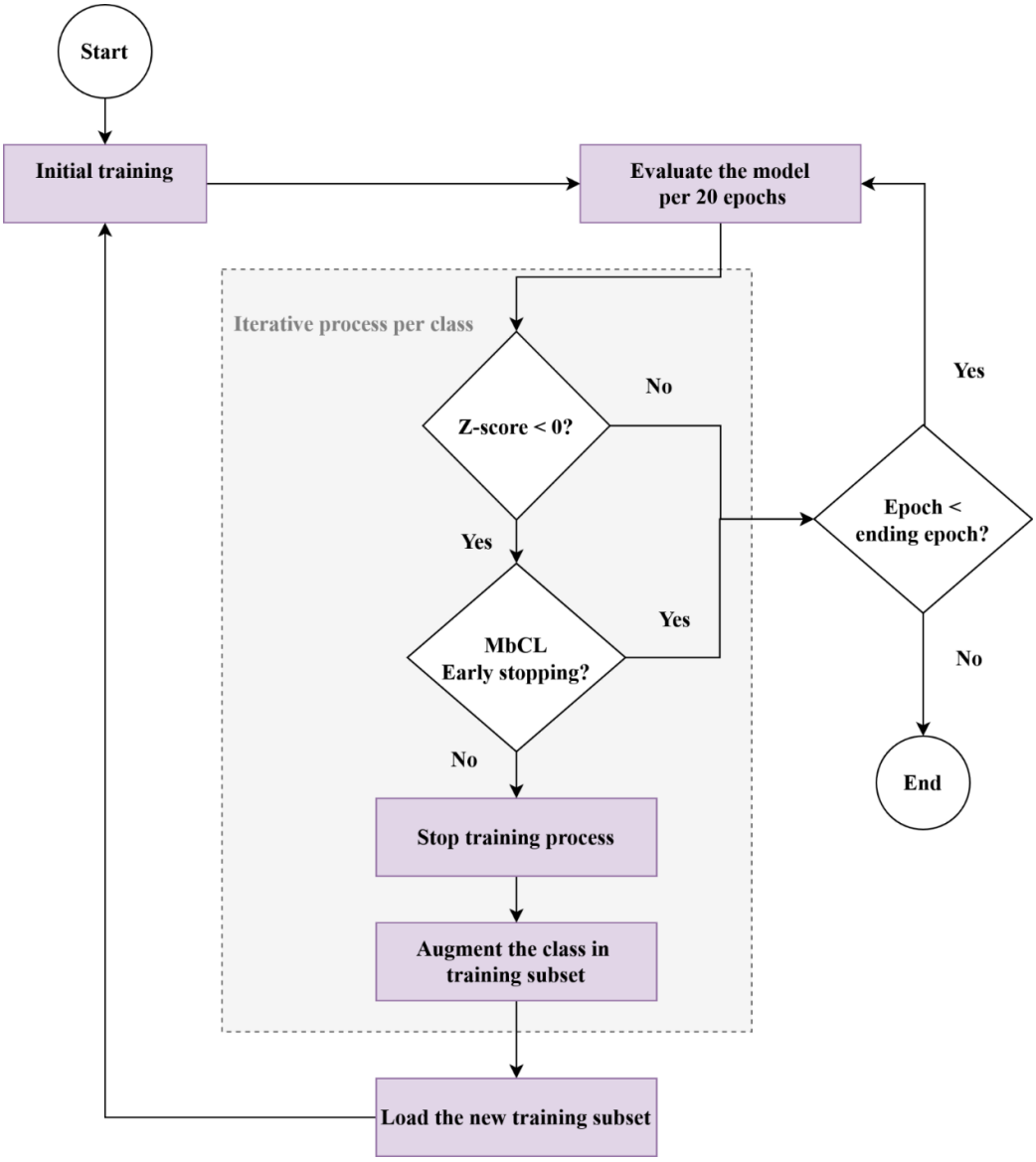


Figure 28. Methodological steps of the MbCL training process.

The process begins with the initialization of the model and the systematic feeding of the D_{Tr} and D_{Val} into the system. During this stage, the model adjusts its internal parameters to minimize a predefined loss function. This foundational phase sets the stage for effective learning by enabling the model to identify patterns and relationships within the data. However, the process does not end with mere parameter adjustment; it extends into a cycle of evaluation and refinement to ensure the model's development aligns with the desired outcomes.

To monitor the model's progress, class-wise accuracy assessments are conducted at predefined intervals of 20 epochs. This interval is strategically chosen to align with the early stopping criterion, a mechanism designed to halt training when no significant improvement is observed in the validation performance over successive intervals. Also, synchronizing the accuracy evaluations with the Class-wise

Early Stopping (CwES) mechanism ensures a consistent and streamlined evaluation process while preventing unnecessary computational effort. These checkpoints are critical for providing detailed insights into the model's performance and for identifying potential weaknesses or areas requiring intervention.

At every 20-epoch checkpoint, a class-wise accuracy analysis is performed to ensure balanced learning across all classes. For each class i, the accuracy A_i is calculated using the formula:

$$A_i = \frac{TP_i}{TP_i+FP_i+FN_i+TN_i} \quad (7)$$

Where TP_i, FP_i, FN_i, and TN_i represent the true positives, false positives, false negatives, and true negatives, respectively, for the class i. This formula captures the proportion of correctly classified instances for each class within the validation subset, providing a granular view of the model's performance on a per-class basis.

To evaluate the overall effectiveness of the model, the mean accuracy (μ) across all classes is computed:

$$\mu = \frac{1}{C}\sum_1^C A_i \quad (8)$$

Where c represents the total number of classes. The mean accuracy serves as a baseline for assessing individual class accuracies, allowing the identification of outliers or underperforming classes. However, mean accuracy alone does not reveal the consistency of the model's performance across all classes. To address this, the standard deviation (σ) of class accuracies is calculated:

$$\sigma = \sqrt{\frac{1}{C}\sum_1^C (A_i - \mu)^2} \quad (9)$$

This standard deviation quantifies the variability of class accuracies around the mean, highlighting disparities in the model's performance.

The next step involves calculating the Z-score for each class i, a metric that normalizes the accuracy deviations relative to the mean and standard deviation. The Z-score is calculated as:

$$Z_i = \frac{(A_i-\mu)}{\sigma} \quad (10)$$

The Z-score provides a precise measure of how much a class's accuracy deviates from the average performance. A Z-score-based threshold ($Thr_{Z-score}$) is established to define acceptable performance levels. Classes with $Z_i < Thr_{Z-score}$ — wherein $Thr_{Z-score} = 0$ — Indicating accuracies below the mean are flagged as underperforming.

Augmentation strategies are applied for each flagged class to enhance the training dataset. This augmentation involves generating new and diverse data for the underperforming class, ensuring efficient resource allocation, and focusing efforts on the areas where improvement is most needed.

5.1.2 REQUIRED AUGMENTED IMAGES QUANTITY

In ML, particularly in the context of image classification, the relationship between the number of training images (N) and the performance of the model ($P(N)$) follows a Power-Law decay function [353], [354], [355]. This relationship reflects the diminishing returns effect, where increasing the number of training samples continues to improve performance but at a progressively slower rate. The Power-Law decay function is expressed mathematically as:

$$P(N) = P_{\infty} - \frac{C}{N^{\alpha}} \quad (11)$$

Here:

- P_{∞} represents the maximum achievable performance for a given task, acting as an asymptotic limit that the model approaches as N becomes very large.
- c is a constant that scales the influence of the sample size on performance.
- α is a positive constant specific to the task, governing the rate of decay and indicating how quickly the performance curve flattens as N increases.

Understanding this relationship is crucial for determining how much additional data is required to achieve a target performance level. Given the computational and resource constraints in generating or acquiring more data, practitioners often employ a linear approximation derived from the Power-Law decay to estimate the required number of additional samples. This approximation is formulated as:

$$\Delta N \approx \frac{N \times (P_{target} - P_{current})}{K} \quad (12)$$

Where:

- ΔN is the number of new samples needed to achieve the desired improvement in performance.
- N is the current number of training samples.
- P_{target} is the target performance level that the model aims to achieve.
- $P_{current}$ is the current performance of the model.

- K is a constant representing the current iteration in the MbCL process, capturing the dynamics of the training cycle.

This formula provides a practical means to estimate how much augmentation or data acquisition is necessary to bridge the performance gap (P_{target}- $P_{current}$) in the overall model. The closed-loop strategy, which tracks the performance of each class, allows for further refinement of this equation at the class level. To address imbalances in performance, the augmentation process targets underperforming classes. For each class i, the performance gap is calculated using the class-specific accuracy (A_i), leading to an updated formulation:

$$\Delta N \approx \frac{N \times (P_{target} - A_i)}{K} \quad (13)$$

Where A_i represents the current accuracy of the class i. This approach ensures that augmentation is focused on the classes that contribute most significantly to the overall performance gap. By generating ΔN additional samples for these underperforming classes, the strategy seeks to correct the distribution of data in the dataset (ending with balancing or unbalancing the dataset) and improve the feature-based representation for each class. Although this targeted augmentation introduces a temporary imbalance in dataset proportions, it aligns with the broader goal of creating a more robust model by enhancing the learning of underperforming classes, thereby raising their accuracy and improving the overall model performance.

5.1.3 VALIDATION-BASED CLASS-WISE STOPPING CRITERION

The MbCL approach operates on the principle of continuous monitoring and feedback adjustment to achieve optimal results. Central to this approach is the technique of validation-based CwES criterion, which serves to discontinue the training process before reaching a predefined endpoint. The CwES is particularly crucial for preventing overfitting. Additionally, CwES aims to reduce computational costs by avoiding unnecessary iterations.

In the MbCL process, the model is trained on the D_{Tr} while its performance is periodically evaluated on the D_{Val} . The decision to stop the training process is made based on the performance of individual classes. Specifically, the accuracy of each class on the validation subset ($A_{i,val}$) is closely monitored throughout the training. If the performance of any class fails to improve over a series of consecutive iterations, the adjustment and augmentation for this specific class will be terminated. This ensures that

the model does not expend resources trying to improve beyond what is realistically achievable for that class. This CwES criterion is determined by a predefined parameter as patience (L), which defines the number of consecutive iterations allowed without improvement before finishing the adjustment. The stopping condition can be expressed mathematically as:

$$t_{stop} = t \quad \text{if} \quad t - t_{best} \geq L \tag{14}$$

Where:

- t represents the current iteration of the closed-loop process.
- t_{best} denotes the iteration at which the best class accuracy ($A_{i,val}^{best}$) was observed.

Whenever a new best accuracy is observed for a given class, the condition is updated as follows:

$$\text{If } A_{i,val}(t) > A_{i,val}^{best} \text{ , then } A_{i,val}^{best} = A_{i,val}(t) \text{ and } t_{best} = t \tag{15}$$

This update ensures that the process always tracks the most recent instance of optimal performance for each class. If the accuracy does not improve within the next L closed-loop iterations after t_{best} , the stopping condition (t_{stop}=t) is met, and the closed-loop process for the class i is terminated.

5.2 AUGMENTATION TECHNIQUES

In the MbCL process, the refinement of a model relies heavily on the strategic application of data augmentation techniques. These techniques serve to artificially expand the D_{Tr} , especially for underrepresented classes, thereby ensuring the model achieves a more balanced and robust performance across all classes. The augmentation process incorporates both classical methods and GAN-based approaches, each contributing uniquely to the enhancement of model performance. This process of addressing underperforming classes in a dataset involves a strategic and data-driven augmentation approach, leveraging insights from prior clustering analysis (Chapter 4). This method ensures that augmentation efforts are focused on the areas where they are most needed, optimizing resource allocation and enhancing model performance. The process employs the application of the FCDIS methodology, which organizes the dataset into meaningful clusters to facilitate targeted interventions.

When a specific class i is identified as underperforming — based on evaluation metrics, Z-scores — attention is immediately shifted to the clusters within that class. Each class i is pre-divided into A_{ij} clusters through the application of the FCDIS method — refer to Chapter 4 —, which groups similar images based on feature similarity. To strategically guide the augmentation process, the sizes of the individual clusters

within the flagged class are analyzed. For each cluster j in A_{ij}, its size S_{ij} — the number of images it contains — is compared against the overall average size of all clusters in the class, denoted as μ_i. The augmentation process prioritizes clusters where $S_{ij} < \mu_i$, focusing on those that are smaller and, consequently, underrepresented. Once the clusters for augmentation are identified, variations of the images within these clusters are generated using the augmentation techniques.

5.2.1 CLASSICAL AUGMENTATION

CA is a fundamental technique employed to improve the representation of classes with outlier performance by generating additional examples through systematic transformations. The augmentation process is designed to introduce diverse variations into the dataset, particularly for underrepresented clusters, thereby enhancing the model's learning capabilities and overall performance.

The space of augmentation techniques is represented as $\Phi = \{\phi_1, \phi_2, \ldots, \phi_k\}$, where each $\phi_k \in \Phi$ corresponds to a specific augmentation operation. These techniques can have adjustable parameters to create a wide range of transformations. Below is a detailed breakdown of the augmentation techniques and their associated parameters:

- Flipping– Horizontal Flip (ϕ_1), Vertical Flip (ϕ_2).
- Rotation (ϕ_3)– ϕ_3^1: $90°$ clockwise, ϕ_3^2: $-90°$ counter-clockwise, ϕ_3^3: $180°$.
- Brightness Adjustment (ϕ_4) – $\alpha \in [0.5,1.5]$.
- Contrast Adjustment (ϕ_5)– $\beta \in [-50,50]$.
- Scaling (ϕ_6) – $s_1, s_2 \in [0.8,1.2]$
- Translation (ϕ_7) – $t_x, t_y \in [-10,10]$.
- Random Rotation (ϕ_8) – $\theta \in [-45°, 45°]$.
- Gaussian Blur (ϕ_9) – Kernel size (5,5).
- Random Shearing (ϕ_{10}) – $s_x, s_y \in$ [–0.2,0.2].
- Random Zoom (ϕ_{11}) – $z_1, z_2 \in$ [0.8,1.5]
- Add Noise (ϕ_{12}) – Noise intensity $n \in$ [–25,25]

For each class i identified as underperforming, attention is directed to its clusters j, which are previously defined through the FCDIS clustering technique. Augmentation is applied to smaller clusters (S_{ij}) — those with sizes less than the average cluster size (μ_i) — to enrich their representation in the dataset.

These augmented images $I^{p}_{i,j,k}$ are subsequently added to the training subset D_{Tr} , where:

- i represents the class,
- j denotes the cluster,
- k indicates the specific augmentation technique applied,
- p refers to the parameter value used for the technique.

These augmentations boost the representation of underperforming classes and clusters.

To maintain the effectiveness of the iterative refinement process, redundancy in augmentation techniques is avoided. A tracking mechanism is implemented for each class, maintaining a list of all augmentation techniques and their specific parameters that have already been applied. Before initiating a new augmentation cycle, the selected technique is checked against the list of previously applied techniques for the given class. If the technique has already been used, an alternative transformation is selected, ensuring that each iteration introduces novel variations. By preventing the reuse of the same augmentation parameters, the process ensures a continuous infusion of fresh data variations into the D_{Tr} . This approach avoids dependency on specific transformations and maintains a diverse and enriched dataset.

5.2.2 Generative Adversarial Network Augmentation

An advanced strategy for augmenting data employs GANs [313] , with particular emphasis on cGANs [234]. Unlike standard GANs, cGANs incorporate class labels into the training process, enabling the generator to produce targeted examples aligned with specific class distributions. This refinement makes cGANs especially suitable for addressing class imbalances and enhancing the representation of underrepresented categories in datasets.

The cGAN framework operates as an adversarial system, comprising a generator and a discriminator engaged in a competitive interplay (Figure 29). The generator receives both random noise and class labels as inputs, synthesizing data that conforms to the specified label. In parallel, the discriminator is tasked with evaluating the fidelity of the generated data by analyzing it, alongside its associated label. Its objective is to distinguish between real samples from the original dataset and synthetic samples generated by the model. Through this adversarial dynamic, the generator iteratively improves, producing data that increasingly mirrors the distribution of real-world samples.

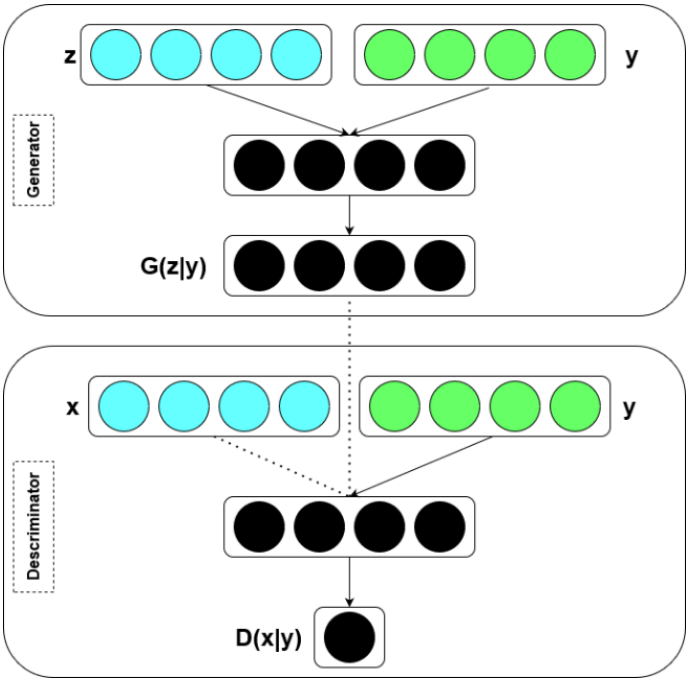


Figure 29. The structure of the cGAN is adapted from [234].

The discriminator architecture is designed for optimal accuracy and efficiency. Class labels are embedded into a 100-dimensional vector and concatenated with the input image data. This combined input is processed through a series of convolutional layers, starting with six input channels and progressively increasing feature maps from 64 to 512. LeakyReLU activations ensure smooth gradient propagation, essential for robust training. BatchNorm is employed throughout the architecture to stabilize training by normalizing feature distributions and mitigating internal covariate shifts.

As the discriminator processes data, its output feature maps are flattened and passed through a dropout layer with a rate of 0.4 to reduce overfitting. Finally, a fully connected layer with a sigmoid activation function generates the probability that the input data is real, allowing the model to effectively classify inputs.

The generator architecture complements the discriminator, emphasizing the creation of high-quality synthetic images. It employs a series of transposed convolutional layers to upsample noise inputs into realistic images. Each layer incorporates BatchNorm for gradient stability and Rectified Linear Unit (ReLU) activations to introduce non-linearity. The generator's final layer utilizes a hyperbolic tangent (tanh) activation function to scale pixel values.

The cGAN training process is controlled by a binary cross-entropy loss function, which serves as a performance metric for both the generator and discriminator. The generator's objective is to minimize the discriminator's ability to distinguish between real and synthetic data, while the discriminator aims to maximize its classification accuracy. This adversarial optimization leads to a dynamic balance where the

generator continually improves its synthetic outputs, and the discriminator becomes more adept at identifying discrepancies. The integration of class labels into the generator and discriminator architecture enables targeted data augmentation.

Building upon the methodologies outlined in prior sections, following subsequent discussion centers on a comprehensive analysis of the outcomes achieved through the implementation of the proposed models. The initial focus is on presenting the experimental setup and results specific to the GAN, encompassing both its configuration and the observed outputs. Following the successful application of these augmentation methods, attention turned to assessing the performance of various DL models. These models were trained and tested using the proposed MbCL approach, which integrates iterative feedback and optimization processes to enhance learning outcomes. The next section represents the comparative analysis of results, highlighting how the integration of CA, cGAN, and the MbCL approach impacted the classification outcomes.

5.3 Results Assessment

This section begins by detailing the evaluation and results derived from the configuration of the cGAN model, aiming to enhance class representation and improve overall model performance. The configuration of the cGAN involved fine-tuning its architecture and training parameters to produce realistic, class-specific synthetic data, leveraging its ability to incorporate class labels during the data generation process. The effectiveness of the cGAN's augmentation capabilities was assessed through a comprehensive analysis, focusing on the quality and utility of the generated images in improving training outcomes.

Following this, the MbCL models are evaluated and compared directly to those trained with a traditional, loopless DL approach. Comparative analysis considers the performance of these models on D_{Val} and the D_{Tst}, offering insights into their efficacy under controlled evaluation conditions. The accuracy metric is employed to provide a significant understanding of the advantages of the closed-loop methodology over conventional training paradigms.

To further assess model robustness, an additional evaluation is conducted using an D_{Ext_Tst}. The results are analyzed to determine how well the MbCL-trained models adapt to unseen data compared to their loopless counterparts. This includes an examination of how the incorporation of augmented data from both classical and cGAN-based techniques contributes to performance consistency across varying datasets. In these evaluations, the following training strategies are encompassed:

- Regular loopless training (RLT) — Referring to models traditionally trained, without the MbCL strategy;
- MbCL with classical augmentation (MbCL-CA) — Models trained under the MbCL strategy, wherein classes with feature-based underrepresentation are enhanced with classical augmentation strategies;
- MbCL with GAN-based augmentation (MbCL-GA) — Models trained under the MbCL strategy, wherein classes with feature-based underrepresentation are enhanced with images generated by a cGAN model.

5.3.1 cGAN Model Configuration: Implications and Outcomes

Utilizing the D_{Tr}^{GLI} and D_{Tr}^{BDI} , a cGAN was trained to generate high-quality synthetic images aimed at enhancing dataset diversity. The training process was carefully designed to ensure stable convergence and meaningful data generation, employing a batch size of 256 and the Adam optimizer with an initial LR of 10^{-4}. To maximize efficiency and facilitate monitoring, the model was trained for 30,000 training steps, with both the generator and discriminator models being saved every 100 training steps. This checkpointing system helped track model performance over time, reducing the risk of forgetting important information and allowing a backup option if the model started to fail.

Figure 30 presents the learning curves for both datasets, D^{GLI} and D^{BDI}, offering insights into the training dynamics of the cGAN. As is commonly observed in GAN-based models, prolonged training periods can lead to instability, often manifesting as mode collapse, where the generator produces repetitive or unrealistic samples. This phenomenon severely limits the model's ability to generate diverse and high-quality images. Specifically, for D^{GLI} (Figure 30a), instability emerges around the 10,000-epoch mark, as evident from the erratic fluctuations in validation loss and accuracy. Such instability suggests that the generator may have started overfitting to the training data, learning to reproduce specific patterns rather than generalizing meaningful features across different samples. This overfitting compromises the model's ability to generate novel, high-quality images and requires early stopping or adaptive LR adjustments to prevent further degradation.

In contrast, the training behavior for D^{BDI} (Figure 30b) exhibits a more stable trajectory throughout the entire 30,000 training steps. Unlike D^{GLI} , where excessive training leads to degradation, the cGAN model for D^{BDI} maintains a relatively smooth learning curve, indicating effective generalization without significant signs of instability or collapse. This suggests that the dataset's inherent structure and class distributions

contribute to a more balanced training process, where the adversarial dynamics between the generator and discriminator remain stable over extended iterations.

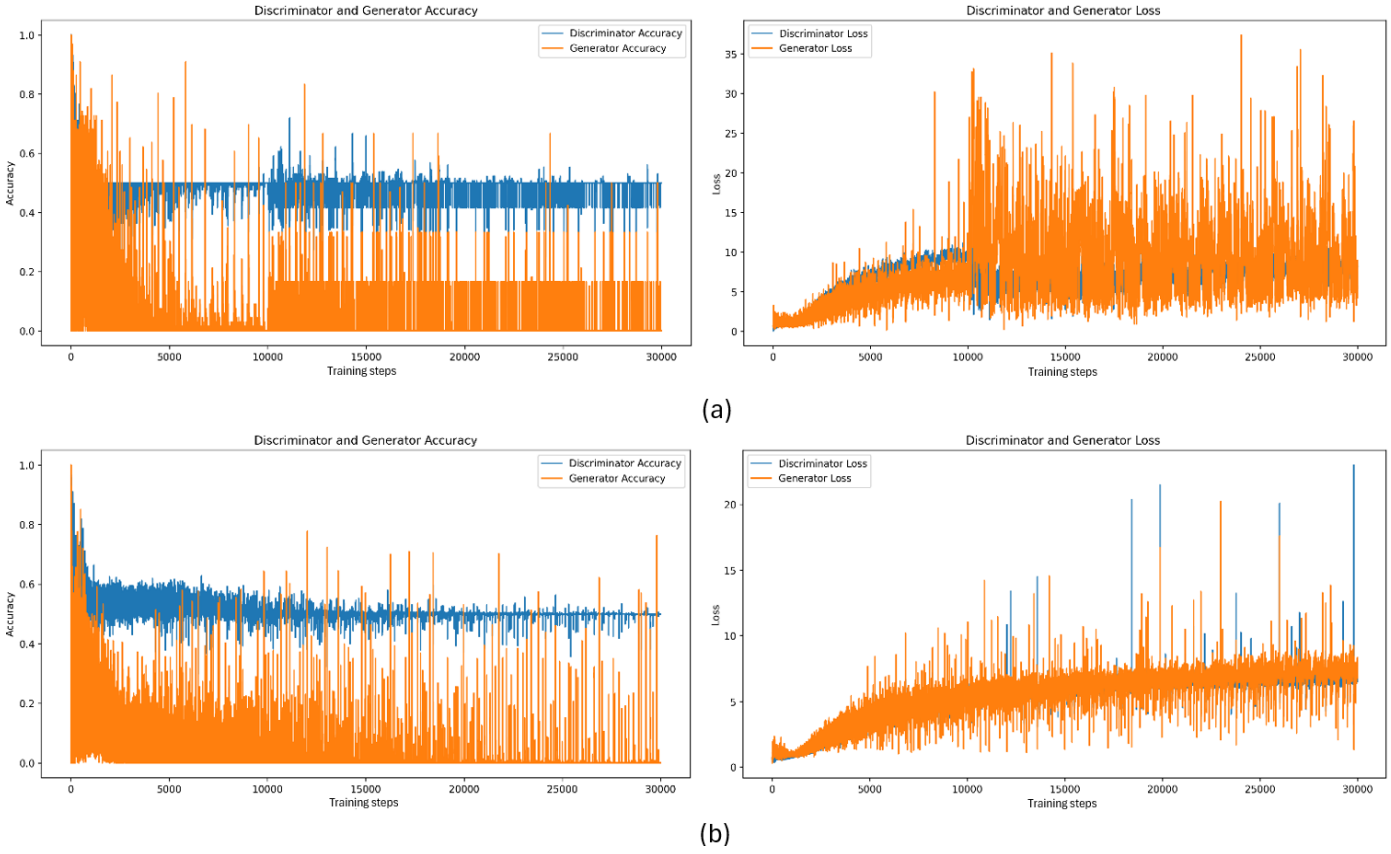


Figure 30. The learning curve of the cGAN training model upon a) D^{GLI} and b) D^{BDI}.

To further quantify the quality of the images generated, the Fréchet Inception Distance (FID) [356] was computed for different training models saved at various stages. FID is a well-established metric for evaluating GAN-generated images by comparing the feature distributions of real and synthetic data. This metric leverages a pre-trained Inception CNN to extract high-level feature representations, comparing them in the form of multivariate Gaussian distributions. The mathematical formulation of FID captures both the mean and covariance differences between the real and generated distributions, providing a comprehensive measure of visual quality and diversity. A lower FID score indicates higher similarity between synthetic and real images, implying superior generation performance.

Despite its advantages, FID alone does not fully capture the perceptual quality of generated images, particularly when fine-grained details and structural consistency are crucial. Therefore, in addition to FID, qualitative evaluations through visual inspections of generated samples were conducted at various training milestones. These inspections helped assess factors such as texture coherence, structural alignment, and artifacts' absence, further confirming the model's effectiveness. Notably, while the cGAN was trained on

D^{GLI} exhibited increasing artifacts and mode collapse symptoms beyond 10,000 training steps, the model trained on D^{BDI} maintained visual consistency throughout the entire training duration.

This metric evaluates the similarity between two sets of data by comparing their multivariate Gaussian distributions — one representing real images and the other representing generated images. It does so by computing the statistical distance between these distributions using the following formula:

$$FIDscore = || \mu r - \mu g ||^2 + Tr\left(\Sigma r + \Sigma g - 2\sqrt{\Sigma r \Sigma g}\right) \quad (16)$$

Where:

- μ_r and μ_g are the mean feature vectors of the real and generated images, respectively.
- Σ_r and Σ_g are the covariance matrices of the real and generated image features.
- Tr denotes the trace of the matrix.

In the development of the proposed MbCL-GA, a key consideration was to balance precision with computational efficiency in model evaluation. To achieve this, the FID score was computed at regular intervals of every 100 training steps. This periodic evaluation ensured that the generative performance of the model was consistently monitored while maintaining a reasonable computational cost, especially considering the setup in use (see subsection 5.2.2).

For the D^{GLI}dataset, after the model reached 10,000 training steps, instability in the learning curve became apparent, suggesting that the model was starting to diverge or overfit. The observed volatility indicated that continuing the training past this point could lead to the inclusion of an unreliable or poorly performing model in the MbCL-GA process. As a result, FID scores were only evaluated up to this epoch to avoid integrating an unstable model. This precautionary approach was critical to maintaining the integrity of the MbCL-GA framework. As shown in Table 16, the lowest FID score for D^{GLI} was recorded at epoch 1,100, which represented the optimal balance between generative performance and stability. Therefore, the model corresponding to this epoch was selected for integration into the MbCL-GA framework, ensuring that only the best-performing and most stable model was used in the final process.

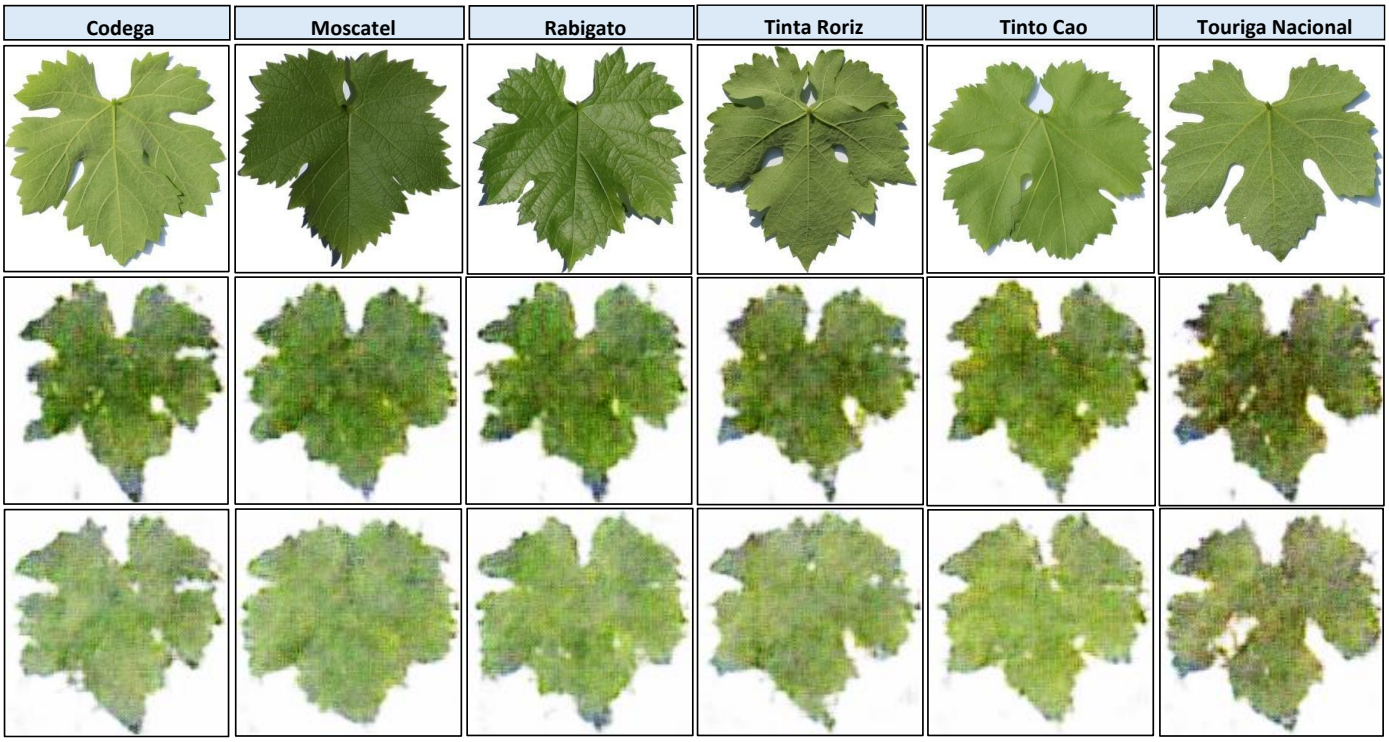
On the other hand, the D^{BDI} exhibited a stable learning curve throughout the entire training process, with no signs of divergence or instability. This stability allowed for continuous evaluation of the model's FID score without the need for early termination. The model was evaluated up to its final epoch, which facilitated a more comprehensive selection process. The FID score for D^{BDI}showed consistent improvement as training progressed, with the minimum FID score being reached at epoch 28,000 (Table 16). Given the steady learning curve and the lowest FID score at this point, the model associated with

epoch 28,000 was selected for integration into the MbCL-GA framework, as it represented the best overall performance for this dataset.

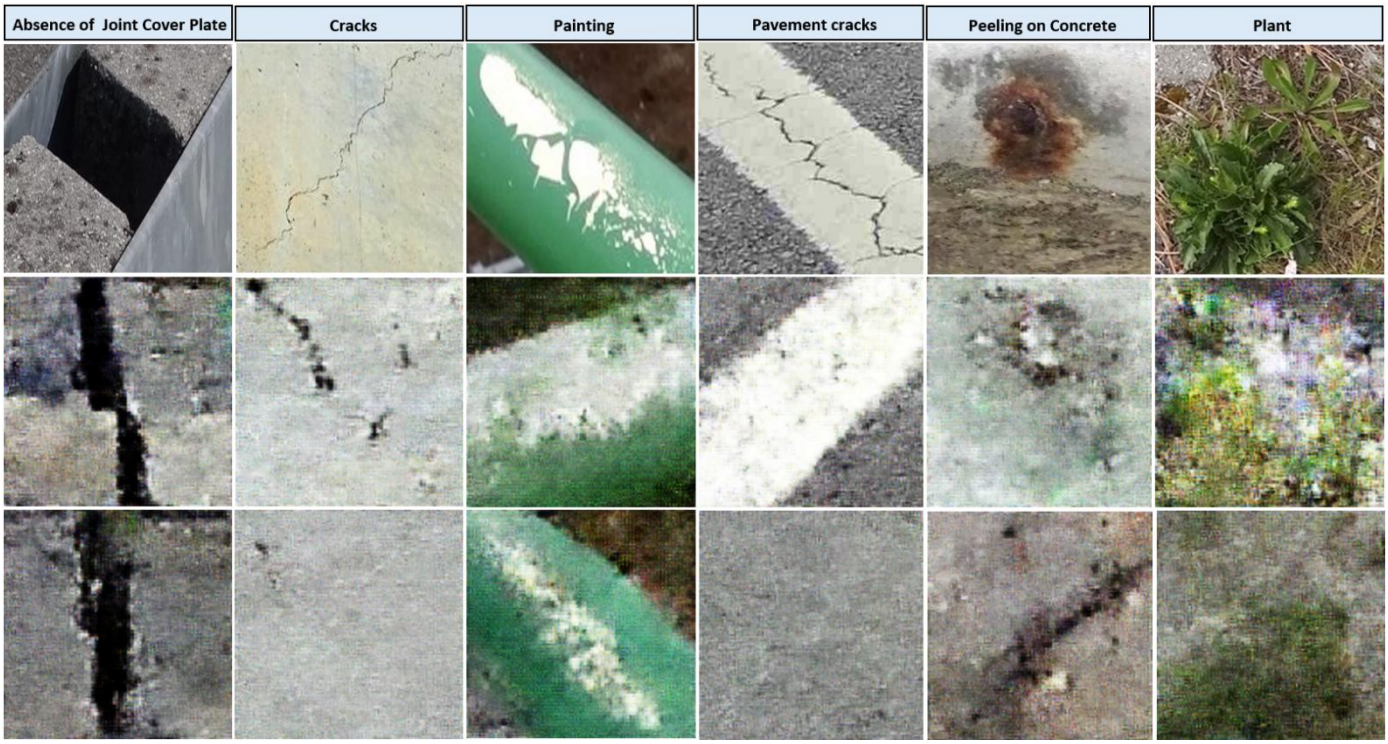
Table 16. Best FID Score over the cGAN training steps upon D^{GLI} and D^{BDI}.

Dataset	FID	Epoch
D^{BDI}	355	28000
D^{GLI}	376	1100

Figure 31 provides a visual representation of the generated class-specific samples for both the D^{GLI} and D^{BDI} showcasing the ability of the cGAN model to produce high-quality, class-specific outputs. These samples are critical for evaluating the model's performance in generating realistic and coherent images that align with the targeted classes. In the case of D^{GLI}, the generated samples (Figure 31a) are carefully crafted to reflect the inherent features and nuances of the dataset's classes, ensuring that each class maintains its unique characteristics while remaining visually consistent. For the D^{BDI}, Figure 31b highlights the framework's ability to generate class-specific samples with similar attention to detail.



(a)



(b)

Figure 31. Generated samples using cGAN. For both D^{GLI} (a) and D^{BDI} (b), the first row corresponds to real images, while the 2nd and 3rd rows are the generated samples for each class.

5.3.2 Model Performance Upon Test Set

This subsection provides a comprehensive examination of the performance of different trained models on D_{Val} and D_{Tst}. The core objective of this analysis is to compare the performance of three distinct approaches: RLT, MbCL-CA, and MbCL-GA. These methods were implemented with the goal of class-wise enhancing model performance. The RLT halting training either after a fixed number of epochs or upon meeting an early stopping criterion, where the early stopping condition is defined as $P = 20$. This means that the training process would stop if the model's performance on a D_{Val} did not improve after 20 consecutive epochs. While RLT represents a straightforward and relatively simple approach, it lacks the flexibility to adaptively respond to class-wise imbalances or the underperformance of specific classes during training.

In contrast, both MbCL-CA and MbCL-GA follow an iterative training procedure, designed to continuously refine and improve the model's performance on underrepresented classes. Both methods adjust the learning process over multiple cycles, re-weighting the importance of underperforming classes by either generating new samples (as with MbCL-GA) or augmenting existing data (as with MbCL-CA) to address class imbalance. This iterative approach allows these models to focus specifically on difficult or underrepresented classes throughout the training process, thus addressing a common challenge in ML: the difficulty of models performing well in minority classes.

To prevent these iterative methods from running indefinitely without achieving meaningful improvements, an additional stopping criterion, CwES, was introduced. This criterion ensures that the loop-based augmentation for a given class terminates when no performance improvement is observed over five consecutive iterations. The introduction of this criterion prevents the use of unnecessary computational resources and ensures that the model halts its training immediately after the point where significant improvements can no longer be achieved. This feature of MbCL-CA and MbCL-GA is a crucial difference compared to RLT, which only halts training based on overall performance or predefined epoch limits.

During the training and evaluation of both MbCL-CA and MbCL-GA, several performance metrics were tracked. These include the per-class accuracy, which provides insight into how well the model performs on each class separately, as well as the identification of outlier classes — i.e., the ones that consistently make the model underperform. Additionally, overall metrics such as validation loss and validation accuracy were recorded. These metrics represent the performance of the model as a whole and were tracked up until the model reached the predefined maximum number of epochs. The models that achieved the highest

accuracies, concerning both specific classes and overall results, are highlighted in the comparative tables presented further.

Table 17. Results of the accuracy of classes upon the D_{Val}^{GLI} for Xception with and without closed-loop augmentation techniques using Nadam optimizer for LR=10^{-3}.

Approach	MbCL Iteration	Classes (Val accuracy)							Val loss	Val accuracy
		Codega	***Moscatel***	***Rabigato***	***Tinta Roriz***	***Tinto Cao***	***Touriga Nacional***	**Outliers**		
MbCL-CA	0	64%	55%	27%	55%	46%	82%	*Rabigato*	1.31	55%
	1	64%	73%	73%	64%	36%	73%	*Tinto Cao*	1.13	64%
	2	64%	73%	73%	64%	55%	82%	*Tinto Cao*	1.03	68%
	3	64%	82%	73%	64%	55%	91%	*Tinto Cao*	0.98	71%
	4	64%	82%	73%	64%	55%	91%	*Tinto Cao*	0.95	71%
	5	64%	82%	73%	55%	55%	91%	*Tinta Roriz, Tinto Cao*	0.93	70%
	6	73%	82%	73%	55%	55%	91%	*Tinta Roriz, Tinto Cao*	0.93	71%
	7	**73%**	82%	**73%**	55%	46%	82%	*Tinta Roriz, Tinto Cao*	0.93	**68%**
MbCL-GA	0	73%	55%	18%	46%	73%	82%	*Rabigato, Tinta Roriz*	1.305	58%
	1	64%	73%	36%	46%	64%	82%	*Rabigato, Tinta Roriz*	1.153	61%
	2	64%	82%	46%	55%	64%	82%	*Rabigato, Tinta Roriz*	1.051	65%
	3	64%	82%	55%	55%	64%	82%	*Rabigato, Tinta Roriz*	1.009	67%
	4	64%	82%	55%	55%	64%	82%	*Rabigato, Tinta Roriz*	0.963	67%
	5	64%	82%	64%	55%	64%	82%	*Tinta Roriz*	0.946	68%
	6	64%	82%	64%	55%	64%	82%	*Tinta Roriz*	0.927	68%
	7	64%	82%	64%	45%	64%	82%	*Tinta Roriz*	0.911	67%
RLT		55%	82%	64%	55%	46%	82%	N/A	1.02	64%

Table 18. Results of the accuracy of classes upon the D_{Val}^{BDI} for Xception with and without closed-loop augmentation techniques using Nadam optimizer for LR=10^{-3}.

Approach	MbCL Iteration	Classes (Val accuracy)							Val loss	Val accuracy
		Absence of Joint Cover Plate	Cracks	Painting	Pavement Crack	Peeling on Concrete	Plant	Outliers		
MbCL-CA	0	92%	92%	99%	94%	69%	98%	Peeling on Concrete	0.22	94%
	1	88%	93%	99%	96%	69%	98%	Peeling on Concrete	0.16	94%
	2	88%	95%	99%	95%	69%	98%	Peeling on Concrete	0.14	94%
	3	92%	95%	99%	98%	69%	98%	Peeling on Concrete	0.13	95%
	4	92%	93%	99%	98%	69%	98%	Peeling on Concrete	0.13	95%
	5	88%	93%	98%	98%	**75%**	98%	Peeling on Concrete	0.12	95%
MbCL-GA	0	88%	92%	99%	93%	69%	98%	Peeling on Concrete	0.234	93%
	1	88%	95%	99%	94%	69%	98%	Peeling on Concrete	0.169	94%
	2	92%	93%	99%	96%	69%	98%	Peeling on Concrete	0.145	95%
	3	92%	92%	99%	96%	69%	98%	Peeling on Concrete	0.14	94%
	4	92%	92%	99%	96%	69%	100%	Peeling on Concrete	0.133	95%
	5	96%	92%	98%	98%	69%	100%	Peeling on Concrete	0.13	95%
RLT		100%	96%	98%	99%	69%	100%	N/A	0.12	**97%**

These tables provide a clear visualization of how each approach is performed under varying conditions. Tables 17 and 18 present detailed comparisons between RLT, MbCL-CA, and MbCL-GA for the D_{Val}^{GLI} and D_{Val}^{BDI}, respectively. In these tables, the results of training with the Xception architecture, Nadam optimizer, and an LR of 10^{-3} are shown. From these tables, it becomes evident that the closed-loop performance of the models may differ significantly, particularly in how outliers (i.e., the minority or underperforming classes) are handled. The iterative approaches, MbCL-CA and MbCL-GA, often show improvements in outlier classes, with their performance increasing across iterations, as reflected by respective accuracies. Such a tendency highlights the effectiveness of these methods in ensuring that classes with initial poorer performances end up contributing to the model's learning process. In contrast, RLT does not demonstrate this kind of iterative refinement, and as such, it may struggle to adequately handle outliers, as the dataset remains static during the whole training process.

Moving on to Tables 19 and 20, these display the class-wise performance results for models trained on D_{Tst}^{GLI} and D_{Tst}^{BDI}, respectively. Such evaluations are critical for understanding how well the models generalize to new, unseen data. The goal is not only to compare the three strategies (RLT, MbCL-CA, and MbCL-GA) but also to evaluate the consistency of the MbCL approach when integrated with various CNN architectures, including Xception, VGG16, and ResNet50. In addition, different optimizers (Nadam and SGD) and LRs (10^{-3} and 10^{-4}) were tested to assess how these factors interact with the MbCL-CA and MbCL-GA strategies.

In both Tables 19 and 20, the best-performing configuration is highlighted in bold, allowing for a direct comparison of how different model setups impact classification performance per class. This breakdown shows, by class, which sets combining architecture/optimizer/LR/dataset enhancement approach are the most effective. Hence, it also allows identifying classes requiring additional optimization or augmentation due to lower predictive accuracy.

Table 19. Class-wise accuracy upon the D_{Tst}^{GLI} for different architectures Xception, VGG16, ResNet50, with and without closed-loop augmentation techniques using Nadam and SGD optimizers for LR=10^{-3} and 10^{-4}.

Models	Optimizer	LR	Approach	Classes Accuracy					
				Codega	***Moscatel***	***Rabigato***	***Tinta Roriz***	***Tinto Cao***	***Touriga Nacional***
Xception	**Nadam**	**10^{-3}**	**MbCL-CA**	**100%**	59%	75%	59%	75%	93%
			MbCL-GA	**100%**	59%	75%	53%	62%	88%
			RLT	**100%**	59%	82%	56%	64%	88%
		10^{-4}	**MbCL-CA**	88%	58%	64%	53%	75%	72%
			MbCL-GA	78%	52%	64%	44%	57%	76%
			RLT	56%	42%	56%	50%	80%	45%
	SGD	**10^{-3}**	**MbCL-CA**	**100%**	50%	75%	56%	64%	88%
			MbCL-GA	**100%**	55%	82%	56%	60%	88%
			RLT	**100%**	57%	82%	53%	67%	82%
		10^{-4}	**MbCL-CA**	80%	42%	73%	43%	62%	76%
			MbCL-GA	88%	50%	67%	41%	54%	62%
			RLT	50%	62%	35%	56%	50%	50%
VGG	**Nadam**	**10^{-3}**	**MbCL-CA**	75%	58%	61%	39%	80%	**100%**
			MbCL-GA	80%	64%	80%	53%	68%	64%
			RLT	60%	73%	67%	39%	56%	48%
		10^{-4}	**MbCL-CA**	0%	**100%**	23%	0%	38%	**100%**
			MbCL-GA	36%	47%	29%	40%	44%	72%
			RLT	0%	86%	0%	0%	67%	20%

Models	Optimizer	LR	Approach	Classes Accuracy					
				Codega	*Moscatel*	*Rabigato*	*Tinta Roriz*	*Tinto Cao*	*Touriga Nacional*
ResNet50	SGD	10^{-3}	**MbCL-CA**	67%	60%	62%	36%	55%	88%
			MbCL-GA	0%	78%	56%	42%	61%	58%
			RLT	67%	75%	50%	39%	59%	45%
		10^{-4}	**MbCL-CA**	30%	57%	30%	35%	50%	**100%**
			MbCL-GA	29%	46%	38%	**100%**	43%	30%
			RLT	0%	0%	0%	**100%**	0%	17%
	Nadam	10^{-3}	**MbCL-CA**	33%	28%	21%	83%	0%	60%
			MbCL-GA	0%	27%	23%	33%	12%	27%
			RLT	0%	0%	0%	0%	24%	19%
		10^{-4}	**MbCL-CA**	23%	24%	0%	20%	30%	**100%**
			MbCL-GA	0%	24%	0%	29%	27%	35%
			RLT	0%	0%	0%	0%	0%	17%
	SGD	10^{-3}	**MbCL-CA**	0%	0%	0%	0%	17%	62%
			MbCL-GA	0%	0%	0%	22%	27%	60%
			RLT	0%	0%	0%	0%	17%	0%
		10^{-4}	**MbCL-CA**	0%	0%	8%	0%	31%	20%
			MbCL-GA	0%	50%	23%	30%	31%	55%
			RLT	0%	0%	0%	0%	0%	17%

Table 20. Class-wise accuracy upon the D_{Tst}^{BDI} for different architectures Xception, VGG16, ResNet50, with and without closed-loop augmentation techniques using Nadam and SGD optimizers for LR=10^{-3} and 10^{-4}.

Models	Optimizer	LR	Approach	Classes Accuracy					
				Absence of Joint Cover Plate	Cracks	Painting	Pavement Crack	Peeling on Concrete	Plant
Xception	Nadam	10^{-3}	**MbCL-CA**	91%	98%	**99%**	93%	90%	96%
			MbCL-GA	97%	99%	**99%**	92%	86%	97%
			RLT	97%	95%	**99%**	92%	95%	97%
		10^{-4}	**MbCL-CA**	91%	98%	**99%**	93%	90%	96%
			MbCL-GA	97%	99%	**99%**	92%	86%	97%
			RLT	97%	95%	**99%**	92%	95%	97%
	SGD	10^{-3}	**MbCL-CA**	97%	92%	**99%**	93%	86%	98%
			MbCL-GA	97%	92%	**99%**	94%	90%	98%
			RLT	**100%**	93%	98%	94%	95%	98%
		10^{-4}	**MbCL-CA**	**100%**	84%	93%	98%	70%	97%

Models	Optimizer	LR	Approach	Classes Accuracy					
				Absence of Joint Cover Plate	Cracks	Painting	Pavement Crack	Peeling on Concrete	Plant
		10^{-4}	**MbCL-GA**	87%	90%	96%	96%	70%	98%
			RLT	96%	80%	89%	98%	**100%**	97%
VGG	Nadam	10^{-3}	**MbCL-CA**	86%	90%	94%	93%	32%	97%
			MbCL-GA	79%	88%	83%	91%	40%	94%
			RLT	92%	79%	80%	88%	73%	92%
		10^{-4}	**MbCL-CA**	78%	66%	82%	94%	36%	86%
			MbCL-GA	66%	67%	76%	86%	36%	96%
			RLT	0%	61%	71%	65%	0%	84%
	SGD	10^{-3}	**MbCL-CA**	87%	**100%**	73%	50%	26%	92%
			MbCL-GA	73%	91%	75%	81%	25%	**100%**
			RLT	**100%**	86%	65%	54%	40%	88%
		10^{-4}	**MbCL-CA**	77%	69%	76%	78%	60%	75%
			MbCL-GA	58%	67%	73%	78%	32%	90%
			RLT	0%	69%	63%	56%	0%	80%
ResNet50	Nadam	10^{-3}	**MbCL-CA**	86%	0%	57%	38%	12%	70%
			MbCL-GA	73%	0%	52%	46%	22%	72%
			RLT	**100%**	0%	77%	30%	0%	43%
		10^{-4}	**MbCL-CA**	**100%**	25%	36%	65%	0%	45%
			MbCL-GA	52%	57%	52%	54%	0%	56%
			RLT	0%	0%	0%	28%	0%	52%
	SGD	10^{-3}	**MbCL-CA**	0%	0%	0%	**100%**	5%	**100%**
			MbCL-GA	33%	0%	0%	28%	0%	**100%**
			RLT	0%	0%	0%	27%	0%	55%
		10^{-4}	**MbCL-CA**	0%	0%	0%	28%	0%	40%
			MbCL-GA	0%	0%	0%	30%	0%	31%
			RLT	0%	0%	0%	27%	0%	**100%**

In Table 21 for the D_{Tst}^{GLI} , MbCL-CA achieved the best performance, with an accuracy of 74%. This suggests that MbCL-CA was particularly effective in handling the D_{Tst}^{GLI} , likely due to its ability to augment the class distribution during training and improve the model's performance on underrepresented classes.

On the other hand, in Table 22 for the D_{Tst}^{BDI} , MbCL-GA emerged as the superior approach, reaching an accuracy of 96%. This result demonstrates the effectiveness of MbCL-GA in improving model

performance when applied to more complex or challenging datasets, like D_{Tst}^{BDI}. The ability of MbCL-GA to generate new samples through class-wise generative augmentation likely helped boost the accuracy for this particular dataset, where class imbalance has been more pronounced.

Table 21. The accuracy of the models upon D_{Tst}^{GLI} for different architectures Xception, VGG16, ResNet50, with and without closed-loop augmentation techniques using Nadam and SGD optimizers for LR=10^{-3} and 10^{-4}.

Models	Optimizer	LR	MbCL-CA	MbCL-GA	RTL
Xception	Nadam	10^{-3}	**74%**	69%	71%
		10^{-4}	**65%**	61%	49%
	SGD	10^{-3}	68%	**70%**	69%
		10^{-4}	**60%**	57%	50%
VGG	Nadam	10^{-3}	**64%**	**64%**	52%
		10^{-4}	35%	**46%**	26%
	SGD	10^{-3}	**58%**	56%	50%
		10^{-4}	**37%**	**37%**	18%
ResNet50	Nadam	10^{-3}	**32%**	25%	20%
		10^{-4}	25%	**29%**	17%
	SGD	10^{-3}	21%	**27%**	17%
		10^{-4}	21%	**32%**	17%

Table 22. The accuracy of the models upon the D_{Tst}^{BDI} for different architectures Xception, VGG16, ResNet50, with and without closed-loop augmentation techniques using Nadam and SGD optimizers for LR=10^{-3} and 10^{-4}.

Models	Optimizer	LR	MbCL-CA	MbCL-GA	RTL
Xception	Nadam	10^{-3}	95%	**96%**	95%
		10^{-4}	95%	**96%**	95%
	SGD	10^{-3}	95%	95%	**96%**
		10^{-4}	92%	**93%**	90%
VGG	Nadam	10^{-3}	84%	**85%**	84%
		10^{-4}	**77%**	75%	67%
	SGD	10^{-3}	62%	**74%**	65%
		10^{-4}	**74%**	71%	63%
ResNet50	Nadam	10^{-3}	38%	**51%**	36%
		10^{-4}	40%	**54%**	31%
	SGD	10^{-3}	8%	28%	**30%**
		10^{-4}	**31%**	30%	28%

The results from both the D_{Val} and D_{Tst} highlight the advantages of MbCL-CA and MbCL-GA over RLT, especially in the context of class imbalances. The iterative, class-focused nature of these methods allows for consistent improvements in performance, particularly concerning challenging classes, which is reflected in the enhanced validation accuracy and test accuracy across both datasets.

5.3.3 Analysis Through External Imagery Data

To conduct a more thorough evaluation of the model results, additional analyses were carried out for both D^{GLI} and D^{BDI} , utilizing an external test set. This external set contained classes that matched those found in the original datasets but were acquired under distinct conditions and specifications. The goal of this external validation was to assess the generalizability of the models trained on D^{GLI} and D^{BDI}, ensuring that the results obtained were not limited to the specific data used for training. To enhance the understanding of the models' performance, a variety of evaluation metrics were employed beyond accuracy. These additional metrics incorporated spatial-related attention indicators, contributing to a more holistic view of how well the models performed not only in terms of classification accuracy but also in terms of the spatial regions they focused on when performing inferences.

One key tool in this analysis was Grad-CAM, which was used to generate attention heatmaps. These heatmaps indicate which regions of the input image were most influential in the model's decision-making process. The Grad-CAM technique allows for visualizing which parts of an image the model "attends" to when making predictions, offering valuable insights into its interpretability and highlighting potential weaknesses or biases in its decision-making process.

Alongside Grad-CAM, the IoU metric was also incorporated into the analysis. The IoU metric is derived from the bounding box computations within Grad-CAM, measuring the overlap between the predicted attention regions and ground-truth regions of interest. A higher IoU indicates that the model's attention is more accurately aligned with the target areas.

Furthermore, the analysis also included a measurement of the distance between the centers of the predicted and ground-truth regions. This additional metric provides complementary information about the model's accuracy in terms of spatial positioning. By calculating the distance between the centers of the predicted attention and the actual target regions.

To summarize these spatial-related metrics, the following indicators were recorded for each class within the external test set:

- Mean accuracy (μAcc) — Provides the average classification accuracy for each class.
- Mean IoU (μIoU) — Measures the average intersection-over-union of the predicted attention regions and ground-truth areas.
- Mean Grad-CAM size (μGCS) — Indicates the average size of the attention regions identified by the Grad-CAM method, giving insight into the focus areas of the model's predictions.

- Mean distance between the centers of both predicted and ground-truth regions (μDis) – This represents the average spatial deviation between the predicted attention and the actual ground-truth regions in the images.

The results from these metrics are detailed in Tables 23 and 24, which present the performance evaluations for the D^{GLI} and D^{BDI} , respectively. These tables highlight variations in both predictive accuracy and spatial alignment across different models. In Table 23, for the $D^{\mathrm{GLI}}_{Ext_Tst}$, models using the MbCL-CA strategy demonstrated more focused attention on objects located closer to the center of the ground-truth areas. This suggests that the MbCL-CA method is more effective at correctly centering its focus around the core regions of interest, which may be indicative of its ability to more effectively handle class imbalances in the central regions of the image.

Table 23.Prediction results upon $D^{\mathrm{GLI}}_{Ext_Tst}$ for Xception using Nadam with LR=10^{-3}.

Class	RLT				MbCL-CA				MbCL-GA			
	μAcc	μIoU	μGCS	μDis	μAcc	μIoU	μGCS	μDis	μAcc	μIoU	μGCS	μDis
Codega	28%	30%	20524	84	27%	32%	19895	84	20%	33%	12032	89
Moscatel	29%	30%	22823	86	29%	31%	22136	86	25%	32%	14851	89
Rabigato	24%	32%	19874	88	23%	33%	19307	87	23%	33%	18642	88
Tinta Roriz	27%	32%	24863	81	27%	33%	24187	80	27%	33%	22827	82
Tinto Cao	28%	29%	22317	76	27%	31%	21774	75	22%	33%	15714	78
Touriga Nacional	24%	32%	22015	81	24%	33%	21437	80	19%	34%	15176	90
Total mean	**27%**	31%	22069	83	26%	**32%**	**21456**	**82**	23%	33%	16540	86

Conversely, for the $D^{BDI}_{Ext_Tst}$ (shown in Table 24), the MbCL-GA strategy outperformed others, achieving a 7% improvement in μIoU compared to alternative methods. Additionally, MbCL-GA demonstrated the smallest distance difference between the predicted and ground-truth regions' centers, suggesting that it exhibited more precise spatial attention.

Table 24. Prediction results upon $D^{BDI}_{Ext_Tst}$ for Xception using Nadam with LR=10^{-3}.

Class	RLT				MbCL-CA				MbCL-GA			
	μAcc	μIoU	μGCS	μDis	μAcc	μIoU	μGCS	μDis	μAcc	μIoU	μGCS	μDis
Absence of Joint Cover Plate	16%	60%	155117	201	16%	62%	151809	201	16%	70%	165720	189
Cracks	13%	50%	44754	176	14%	51%	43387	173	17%	57%	54852	156
Painting	26%	58%	113609	209	26%	59%	111003	205	26%	68%	117826	209
Pavement Crack	15%	54%	54565	226	15%	55%	53161	224	15%	63%	45058	215
Peeling on Concrete	9%	52%	101922	201	9%	53%	98534	200	13%	60%	103238	187
Plant	32%	58%	142342	167	31%	59%	138554	171	24%	64%	107522	169
Total mean	**19%**	55%	102051	197	**19%**	57%	99408	196	**19%**	**64%**	**99036**	**187**

This result highlights the ability of MbCL-GA to more accurately align its predictions with the true regions of interest, improving its overall spatial precision. Afterwards, the classification accuracies of 12 models trained using various architectures (such as Xception, VGG16, and ResNet50) and evaluated with both $D_{Ext_Tst}^{GLI}$ and $D_{Ext_Tst}^{BDI}$ are summarized in Tables 25 and 26. In Table 25, for the $D_{Ext_Tst}^{GLI}$, the MbCL-GA strategy demonstrated the highest effectiveness, achieving an accuracy of 32%. Although this accuracy may appear lower compared to others, it still indicates significant improvement relative to traditional approaches, particularly when considering the model's ability to focus on spatial aspects of the image.

Table 25. Accuracy of the classification upon the $D_{Ext_Tst}^{GLI}$.

Models	Optimizer	LR	MbCL-CA	MbCL-GA	RLT
Xception	Nadam	10^{-3}	30%	27%	**32%**
		10^{-4}	17%	**22%**	20%
	SGD	10^{-3}	23%	**32%**	25%
		10^{-4}	20%	**23%**	18%
VGG	Nadam	10^{-3}	22%	23%	**27%**
		10^{-4}	12%	18%	**20%**
	SGD	10^{-3}	**28%**	**28%**	23%
		10^{-4}	12%	**32%**	17%
ResNet50	Nadam	10^{-3}	15%	**27%**	20%
		10^{-4}	17%	**22%**	17%
	SGD	10^{-3}	17%	**32%**	17%
		10^{-4}	15%	**23%**	17%

Table 26. Accuracy of the classification upon the $D_{Ext_Tst}^{BDI}$.

Models	Optimizer	LR	MbCL-CA	MbCL-GA	RLT
Xception	Nadam	10^{-3}	68%	**72%**	70%
		10^{-4}	**73%**	72%	68%
	SGD	10^{-3}	**73%**	70%	70%
		10^{-4}	**72%**	67%	68%
VGG	Nadam	10^{-3}	**43%**	38%	37%
		10^{-4}	**38%**	32%	30%
	SGD	10^{-3}	**37%**	35%	**37%**
		10^{-4}	**40%**	35%	28%
ResNet50	Nadam	10^{-3}	**40%**	28%	30%
		10^{-4}	20%	33%	**23%**
	SGD	10^{-3}	20%	**27%**	22%
		10^{-4}	23%	**30%**	17%

In contrast, for $D_{Ext_Tst}^{BDI}$ (as shown in Table 26), the MbCL-CA strategy significantly outperformed all others, achieving an accuracy of 73%. This high accuracy suggests that MbCL-CA was better suited for handling the class distribution and spatial characteristics of the $D_{Ext_Tst}^{BDI}$, indicating its success in both improving classification accuracy and enhancing the spatial focus on relevant regions.

Figure 32 provides a sample of the previously discussed indicators, highlighting an in-depth comparison of model performance across different classes. The figure consists of two distinct representations per class, each highlighting key spatial and predictive metrics that contribute to the overall evaluation of model effectiveness.

The first one focuses on multiple performance indicators, including IoU, Prediction (Pred), Prediction Probability (Acc), and Displacement Between Centers (Dis). The second one illustrates the Grad-CAM-based heatmap region size (GCS). This visualization reveals how much of the image the model considers important when carrying out an inference. A larger Grad-CAM heatmap region indicates broader attention coverage.

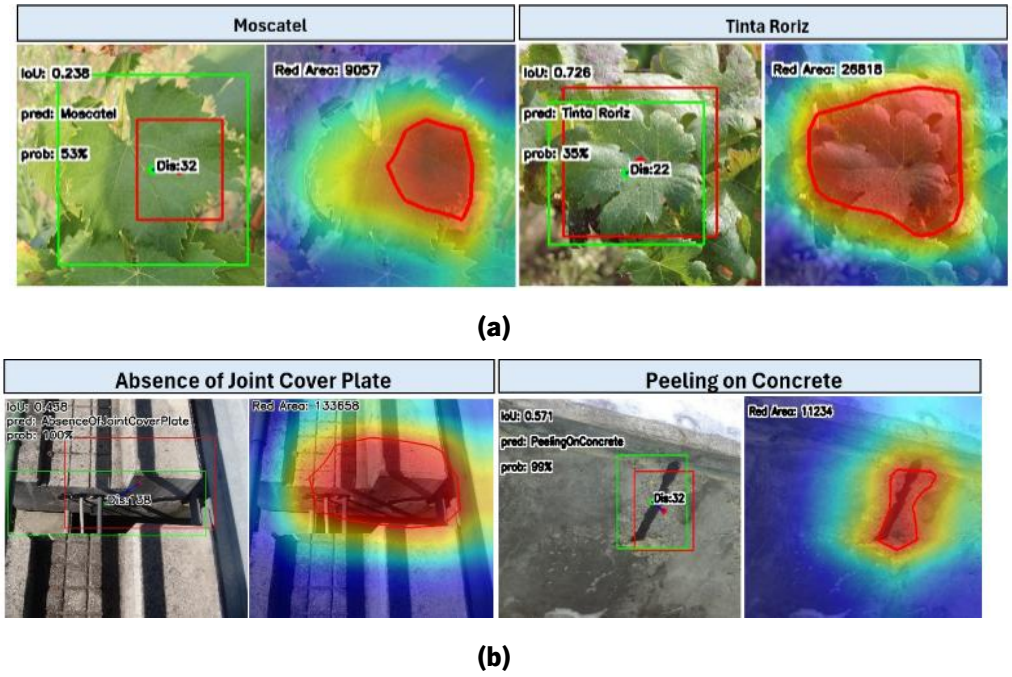


Figure 32. Samples of visual performance assessment for two classes of each dataset: a) $D_{Ext_Tst}^{GLI}$, b) $D_{Ext_Tst}^{BDI}$.

This chapter introduced MbCL methodology, an adaptive training approach designed to refine model performance through ongoing, data-driven feedback mechanisms systematically focused on individual class behavior. Unlike conventional training procedures that treat all classes equally, regardless of their distribution or performance, MbCL actively monitors and responds to class-level discrepancies during the training process, aiming to ensure more balanced learning outcomes, particularly in datasets affected by class imbalance or underrepresented categories.

The core of the MbCL methodology lies in its iterative evaluation framework, which incorporates checkpointing every 20 epochs. At each checkpoint, the model performs a detailed per-class accuracy analysis, identifying classes whose performance deviates significantly from the rest. These deviations are flagged using a defined outlier detection criterion, which considers accuracy across checkpoints. This dynamic monitoring process ensures that the classes that most contribute to lowering models' performances are addressed under a representation reinforcement perspective. Once an underperforming class is identified, the methodology triggers a targeted data augmentation intervention, selecting one of two augmentation strategies (MbCL-CA and MbCL-GA).

Together, these augmentation strategies operate within a closed-loop system, where the model is continually re-evaluated and re-trained as the number of examples increases for the targeted classes. This feedback loop not only adjusts representational imbalances as they occur, but also promotes consistent model performance across classes´ distributions, enhancing robustness and generalizability.

Furthermore, the perspective posed in this chapter not only demonstrates the effectiveness of the MbCL-CA and MbCL-GA strategies in improving classification accuracy but also highlights their ability to enhance spatial alignment and attention precision, as demonstrated in the assessments involving Grad-CAM.

When considered alongside the FAbDSO techniques introduced in the previous Chapter 4, the MbCL methodology demonstrates a strategic complement to the optimization pipeline. The next chapter discusses the experimental results with respect to the research question, focusing on how the proposed methods affect performance and generalization.

6 Discussion and Conclusion

This chapter is dedicated to discussing the results obtained for the main proposed frameworks – the feature-aware-based data splitting methods and the MbCL methodology – while addressing the RQs posed at the beginning of this work. For each RQ, the contributions of the proposed methodologies to addressing key challenges in DL workflows—such as dataset imbalance, feature representation, and generalization—are analysed. In addition, a critical reflection is presented regarding the implications, limitations, and outcomes derived from the findings. At the end, the main conclusions are drawn, a summary of the peer-reviewed scientific contributions resulting from this work is presented, and potential directions for future research are outlined.

6.1 RESEARCH QUESTIONS-ORIENTED DISCUSSION

In this subsection, each RQ posed at the beginning of this document is revisited and discussed, considering the challenges addressed, the methodologies proposed, and the experimental results obtained throughout the work.

6.1.1 ADDRESSING RQ1

One of the challenges of this work was to explore whether feature-aware dataset splitting could improve DL model performance, specifically addressed through RQ1:

> **Can feature-aware dataset splitting improve the performance of the models and constitute a viable alternative to traditional techniques exploring learnability potential (e.g., KFCV)?**

To address this RQ, two feature-aware dataset splitting strategies — FCDIS and FSCIS — were proposed, implemented, and assessed in Chapter 4. The proposed methods demonstrated strong evidence of contributing to the development of models with enhanced inference capabilities compared to traditional splitting techniques, such as HO and KFCV10. More specifically, evaluated across two distinct domains — D^{BDI} and D^{GLI} —, the proposed methods yielded notable improvements in training stability, classification accuracy, and spatial precision of model predictions.

A primary observation lies in the improved consistency of training and validation metrics across models trained using FCDIS and FSCIS approaches (explored in Chapter 4). As indicated by the learning curves (Chapter 4, Figure 22), these models exhibit smoother trajectories and reduced oscillations when compared with those trained on datasets split by HO or KFCV10 lead. This behaviour suggests that the proposed strategies lead to potentially improved models, benefiting from a more balanced distribution of

features and class representations among training and validation subsets. Such balance, in turn, promotes a more stable learning process and, eventually, contributes to the mitigation of overfitting.

The accuracy results further confirm the advantages of the proposed methods. As highlighted in Chapter 4, Tables 10 and 11, the first observation is the better performance of the KFCV over the HO, aligning with conclusions from [150]. As a second observation, notably, the proposed approaches in this study, which are based on clustering and non-random, ratio- and structured-driven selection, consistently surpassed HO and outperformed most of the KFCV10-derived models, particularly when paired with the Nadam optimizer. This is consistent with research from [152], which proved the importance of the similarity-based techniques over the traditional ones (HO, KFCV). Unlike existing clustering-based splitting techniques such as [155], [168], which often rely on random sampling within clusters, our method deterministically selects samples based on defined ratios and ensures even distribution across folds. This design reduces both redundancy and randomness, contributing to more stable and representative training and test sets.

Generalization capacity was also validated using external datasets, $D_{Ext_Tst}^{GLI}$ and $D_{Ext_Tst}^{BDI}$, which presented variations in acquisition conditions while preserving class identities. Results presented in Chapter 4, Tables 12 and 13, clearly demonstrate that the models trained with FCDIS and FSCIS maintained high performance when confronted with unseen data, further confirming the robustness of the proposed dataset splitting techniques.

Grad-CAM-based visualizations and spatial metrics were incorporated to extend the interpretability of these models and to deepen the performance assessment beyond accuracy. Through the analysis of attention heatmaps (Figures 24 and 25 in Chapter 4) and quantified via metrics such as IoU, GCS, and Dis, it became evident that models trained using the proposed splitting methods exhibited improved spatial precision in their attention. Specifically, the results in Tables 14 and 15 showed that FCDIS and FSCIS-trained models had higher IoU values, lower Dis values, and more compact Grad-CAM regions, indicating a tighter focus on objective areas. This spatial focus was not only consistent but also predictable: models demonstrating higher attention accuracy tended to yield higher classification performance. This relationship between attention-based localization and predictive success highlights the potential of integrating spatial indicators as complementary performance metrics.

However, direct comparisons with existing studies are not feasible, as the experimental settings differ significantly. Factors such as dataset size and structure, class distribution, problem domain, model selection, and evaluation metrics vary across studies, making it inappropriate to draw definitive conclusions

from such comparisons. For example, some studies target classification tasks with class imbalance [168], while others evaluate regression problems or work with large-scale benchmark datasets. Despite these differences, the consistent performance gains observed across our experiments support the broader conclusion found in the literature: structure-aware, similarity-based splitting methods tend to outperform traditional random or stratified approaches. The proposed methods, by avoiding randomness and enforcing controlled sample selection, offer additional advantages in terms of reproducibility and generalizability — especially in scenarios with limited or complex data.

6.1.2 Addressing RQ2

Exploring whether targeted, class-aware data augmentation can enhance the representativeness of underrepresented classes, thereby improving overall model performance while promoting feature-based class-wise balance, through RQ2:

> **Is it beneficial to augment data strategically to increase class-wise representativeness to achieve a more adequate equilibrium in terms of the feature set, rather than keeping the cardinality of the categories balanced within a given dataset?**

To address RQ2, the augmentation techniques within the MbCL methodology were developed in two forms: MbCL-CA (utilizing classical augmentation) and MbCL-GA (employing cGANs). These strategies introduced augmentations selectively and dynamically, focusing on classes with weaker performance during training (explored in Chapter 5).

The validation stage showed that MbCL-CA and MbCL-GA contribute to enhancing model performance on both datasets. For D_{Val}^{GLI} (Table 17), improving overall accuracy and reducing validation loss. Notably, underrepresented classes such as *Codega* and *Rabigato* achieved accuracy gains of 18% and 9%, respectively, confirming the method's effectiveness in addressing class imbalance. In D_{Val}^{BDI} (Table 18) While overall accuracy did not improve substantially, class-specific gains (e.g., a 6% increase for "Peeling on Concrete" with MbCL-CA) were observed. This suggests that classical augmentation techniques are particularly helpful in cases where specific classes have limited data, supporting previous findings that simple image transformations can improve model robustness in data-scarce scenarios [177], [191]. The comparative success of GA aligns with the literature on the benefits of generative models for data augmentation in imbalanced classification tasks. Foundational works such as Wasserstein GAN [209] and

the comprehensive overview by Creswell et al. [210] provide theoretical and practical insights into the stability and effectiveness of GANs for data generation. More specifically, approaches focusing on balancing data distributions, such as the BAGAN method introduced by [229], demonstrate how GAN-based augmentation can significantly improve classification performance on underrepresented classes. cGAN frameworks further extend this utility by enabling class-specific augmentation, as shown in recent works [234], [235], [236]. This body of literature supports the observed improvements in accuracy for underrepresented classes in the dataset of this work, confirming the effectiveness of generative augmentation in mitigating class imbalance [229], [232], [233], [234], [243]. Unlike classical augmentation, GA introduces novel and diverse samples that may capture more complex feature variations, which is crucial for enhancing minority class recognition. Both methods are effective strategies to address class imbalance and improve performance, as also demonstrated in recent automated augmentation frameworks [229], [230], [233], [235].

While a direct comparison with recent automated augmentation frameworks was not performed, the effectiveness of CA and GA in mitigating class imbalance aligns with findings in the literature that demonstrate similar improvements using automated data augmentation strategies [177], [180], [191], [194], [229], [233], [235].

6.1.3 Addressing RQ3

Exploring whether a self-regulated, feedback-driven training loop, as proposed in the MbCL framework (Chapter 5), can serve as an effective dataset-level optimization strategy to improve model learning and generalization, through RQ3:

> **Can a self-regulated closed-loop strategy be implemented to perform dataset-related optimizations, for consistently enhancing the quality of DL models targeting classification tasks in diverse contexts (man-crafted vs. natural environments)?**

To address RQ3, the MbCL method introduced a closed-loop training approach that adjusts data augmentation dynamically based on class-wise performance metrics. Unlike traditional loopless training (RLT), MbCL uses feedback metrics (e.g., per-class accuracy trends) to guide augmentation decisions throughout training.

The experimental evaluation of the proposed MbCL methodology — through both classical augmentation (MbCL-CA) and generative augmentation (MbCL-GA) — demonstrates its overall effectiveness compared to the RLT approach. This improvement is consistently observed across both datasets: D^{BDI} and D^{GLI}. On the test datasets (D_{Tst}^{GLI} and D_{Tst}^{BDI}), MbCL-based models outperformed RLT in approximately 85% of test scenarios, indicating improved generalization and robustness. Regarding the performance variation by architecture, MbCL-CA reached the best results with Xception, while MbCL-GA outperformed others when combined with ResNet50. VGG responded moderately well in tests involving both augmentation strategies. In the D_{Tst}^{BDI}, even though some RLT models outperformed MbCL, most of the improvements resulted from the employment of MbCL-GA, highlighting its strength in modeling complex and diverse visual features, which confirms prior findings, i.e., the capacity of MbCL-GA in enhancing underrepresented classes [249], [254].

Grad-CAM-based tests on external datasets ($D_{\mathrm{Ext_Tst}}^{GLI}$ and $D_{\mathrm{Ext_Tst}}^{BDI}$) revealed enhanced spatial awareness and interpretability. While classification accuracy in D^{GLI} showed modest gains, both MbCL-CA and MbCL-GA increased IoU and centroid alignment, consistent with literature emphasizing augmentation's impact on model attention [251], [255]. On D^{BDI}, MbCL-GA achieved a 64% IoU and 187px centroid displacement, representing the best localization performance, comparable to results from attention-aware approaches[260] . When analyzing the top-performing models, MbCL-GA accounted for 75% of the best results on the D^{GLI}, while MbCL-CA reached a total of 67% on the D^{BDI}, reflecting the complementary strengths of each augmentation strategy across different data domains. Unlike standard GAN-based methods or AutoML augmentation pipelines [250], [253], MbCL is lightweight and selective, using real-time, class-specific generation through cGANs and feedback-driven selection — which contrasts with the computationally heavy brute-force search occurring in NAS [251], [252].

Though inspired by RL concepts[256], [260], MbCL avoids complex reward shaping and tuning. Instead, it relies on class-wise performance signals to adjust augmentation strategies — aligning with what has been reported in recent literature for more interpretable and scalable RL-inspired training pipelines in vision and biomedical ML [257], [258], [263]. This makes MbCL a practical option for researchers or practitioners working in domains with limited computational budgets or dynamic data distributions. To prevent excessive augmentation (i.e., assessable through the model's convergence turning point), MbCL integrates early stopping and checkpoint monitoring, a strategy also adopted in model regularization frameworks [253]. External validation confirmed the effectiveness in tackling oversampling issues.

Overall, the development and testing of MbCL (Chapter 5) confirm the practicality of feedback-regulated, selective augmentation, particularly in diverse domains. The consistency of performance gains supports its use in continuous dataset optimization and sustainable model training workflows.

6.2 CONCLUSIONS

The growing demand for reliable DL models in visual classification tasks has brought persistent challenges concerning dataset quality, feature representativeness, and class distribution. These limitations often result in models that are either biased to specific patterns and, therefore, less skilled in effectively inferring classes characterized by certain underrepresented or scarce conditions. To address these issues, this work was guided by three central RQs, focusing on the role of feature-aware data splitting, the strategic use of augmentation to enhance class-wise information, and the potential of self-regulated closed-loop systems for iterative dataset refinement.

In response to these challenges, a comprehensive framework was developed and experimentally validated. It introduced two innovative strategies designed to improve the robustness, fairness, and interpretability of DL models trained on balanced/imbalanced datasets. The feature-aware dataset splitting (FCDIS and FSCIS) methods, along with the MbCL technique, which incorporates both classical augmentation (MbCL-CA) and generative augmentation (MbCL-GA), provided improvements in class-wise performance, spatial precision, and overall model generalization.

The FCDIS and FSCIS strategies enhance the data-splitting process by ensuring that both common and underrepresented classes are well-represented across training, validation, and testing splits. This dataset organization not only results in smoother training and validation curves but also potentially reduces model overfitting tendencies, which are commonly observed when training on poorly distributed datasets. These strategies contribute to more stable and reliable model convergence, with reduced oscillations in the training curves and improvements in classification accuracy. This study's findings also emphasized the synergy between optimizer selection and dataset structuring. The Nadam optimizer, in particular, exhibited enhanced performance in conjunction with FCDIS and FSCIS, underscoring the value of combining intelligent data splitting with adaptive optimization techniques. The results demonstrate that even with the challenges of class imbalance, DL models can achieve high generalization ability if the training process is carefully tuned concerning both data augmentation and feature-aware dataset splitting.

In addition, the MbCL technique stands out for its lightweight, feedback-guided augmentation methods, applied in line with the training process. By focusing on class-specific performance feedback, MbCL-CA and MbCL-GA selectively enhance underrepresented classes. This makes MbCL particularly suitable for

real-time, scalable applications where resource constraints are a concern. Importantly, the approach does not rely on traditional augmentation pipelines but instead focuses on the dynamic generation of images to address class imbalances during the training process. One of the key strengths of both MbCL strategies is their ability to improve spatial localization in model predictions. The inclusion of spatial metrics such as IoU and Dis allowed for an explainability-based deeper understanding of the model's ability to focus on relevant areas during inference. In turn, it was possible to highlight that MbCL-trained models demonstrated superior attention to key features in both the D^{GLI}, D^{BDI}, as evidenced by the improved performance of metrics related to spatial accuracy.

In conclusion, the combination of these two strategies — FCDIS/FSCIS for data splitting and MbCL for class-wise learning enhancement — offers a comprehensive solution to improve model accuracy, generalization, and interpretability, particularly in real-world scenarios where data imbalance and variation are common.

6.3 Summary of Thesis Scientific Contributions

This thesis resulted in a set of key scientific outputs. Therefore, the peer-reviewed publications that encompass the main research findings are highlighted in this subsection, linking them to the relevant chapters where the work is elaborated. Details such as publication venues, indexing status, and impact metrics are included where applicable to demonstrate these contributions' academic quality and impact. The main scientific contributions are presented through the following publications:

- **"Automatic Optimization-Based Methods in Machine Learning: A Systematic Review".**
  - ❖ Venue: this paper was published as a conference chapter in the *Proceedings of the 2023 Intelligent Systems Conference* (*IntelliSys,* ISSN 2367-3370), Springer Lecture Notes in Networks and Systems, Volume 2 (April 2024), edited by Springer. doi: https://doi.org/10.1007/978-3-031-47724-9_21.
  - ❖ Abstract: Machine Learning (ML) is a subfield of Artificial Intelligence (AI) that has been applied to various fields ranging from industrial to medical sectors, to perform miscellaneous Computer Vision tasks such as image classification, image segmentation, object detection, and language modeling. Notwithstanding, having a suitable model with practical applicability requires performing appropriate structural operations upon datasets, building adequate CNN architectures from scratch or resorting to the ones available in the state-of-the-art, and, either

way, parameterizing them to improve machine learning skills, usually, in a trial-and-error fashion. Aligned with this context, despite the several semi-/fully automatic approaches that can be found in the literature, (e.g., grid search for hyperparameter fine-tuning, auto-Machine Learning for self-configurable model development, and automatic methods for data arrangement and augmentation), which are often integrated with combination to establish automatic pipelines for the effective implementation of solutions powered by AI, surveys documenting such topic seem to be scarce. Therefore, the main goal of this work is to present an updated yet extensive literature review focusing on this class of approaches, considering the importance of their role from the perspective of ML Optimization.

- ❖ Thesis-paper alignment: this literature review paper provides a comprehensive overview of the state-of-the-art of the current ML-based optimization techniques. This work encompasses a great part of the theoretical foundation addressed in this thesis and situates the proposed approaches within the current research landscape, which is detailed in Chapter 2.
- ❖ Indexing and metrics: as a part of *Lecture Notes in Networks and Systems Springer* Book Series, several major indexing services such as DBLP, EI Compendex, INSPEC, and SCOPUS are involved.

- **"Benchmarking Deep Learning models and hyperparameters for Bridge Defects Classification".**
  - ❖ Venue: this paper was published in the CENTERIS 2022 conference proceedings, released by Elsevier in the *Procedia Computer Science* series, Volume 219 (July 2023). Edited in Elsevier. doi: https://doi.org/10.1016/j.procs.2023.01.299.
  - ❖ Abstract: Deep learning (DL) is becoming increasingly popular in numerous application fields within the current Fourth Industrial Revolution (4IR) era. This is mainly due to its capability for providing accurate predictions and reliable consistency in decision-making. Bridge engineering focused on structure monitoring and inspection is a crucial activity for disaster prevention. Therefore, it is an application field wherein synergies between professional knowledge and sophisticated machine-based analytics strategies can be established and even drive time-effective interventions. This paper presents a comparison of DL models used to detect defects in bridges, resorting to the following architectures: MobileNetV2, Xception, InceptionV3, NASNetMobile, VGG16, and InceptionResNetV2. Different optimizers (e.g., Nadam, Adam, RMSprop, and SGD) crossed with distinct LRs (e.g., 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, and 10^{-5}) were employed. VGG16, Xception, and NASNetMobile showed the most stable learning

curves. Moreover, Grad-CAM overlapping images clarifies that InceptionResNetV2 and InceptionV3 models seek features outside the areas of interest (defects). Comparing optimizers' performance, the adaptive ones outperform SGD with decay schedulers for LRs.

- ❖ Thesis-paper alignment: this paper provided an opportunity to explore one of the datasets used in this study (BDI) and to benchmark a series of DL architectures and hyperparameters. These experiments primarily support the rationale behind the DL-oriented optimization strategies proposed in Chapter 3, with relevance for subsequent Chapters 4 and 5.
- ❖ Index and metrics: *Procedia Computer Science* is indexed by Scopus, the Web of Science Conference Proceedings Citation Index, Inspec, and Engineering Village and has a CiteScore of 4.1.

- **"Advancing Digital Ampelography: Automated Classification of Grapevine Varieties".**
  - ❖ Venue: this paper was published in Computers *and Electronics in Agriculture* (*COMPAG,* ISSN 1681-699), Vol. 229 (February 2025). Edited in Elsevier. doi: https://doi.org/10.1016/j.compag.2024.109675.
  - ❖ Abstract: Within the winemaking value-chain, certification, and authenticity bear paramount significance. Both rely upon the meticulous classification of grapevine varieties on which historically ampelography – an intricate practice for identifying grapevines, primarily based on leaf traits – serves as the linchpin. Nevertheless, the growing shortage of skilled practitioners demands the modernization and preservation of this specialized knowledge. This paper delves into the convergence of ampelography, artificial intelligence (AI), and advanced techniques. Specifically, it addresses *Vitis vinifera* L. phenotypic recognition through leaf image analysis, highlighting the vital intersection of tradition and technology. The main objective is to present a comprehensive framework that enables the optimization of a Xception neural network, focusing on a few key dimensions: i) comparison between stochastic and adaptive optimizers; ii) comparison between splitting/augmentation order strategies; iii) assessment of different rates for data augmentation; iv) evaluation of learnability, considering the noise influence of having different background conditions; v) assessment of phenotype variability influence; and vi) evaluation of cross-validation-like configurations as inference improvers. Through an in-depth exploration of these critical dimensions, this paper underscores the pivotal role that AI-driven ampelography can have in

supporting quality and accurate wine certification and authenticity. Indeed, the essence of preserving and expanding ampelographic expertise in the digital era takes centre stage. Notably, this research prioritizes specific methodologies, shedding light on the potential prowess of Xception-based models. The results are promising, with precision rates of 95% and 98% achieved for two imagery sets included in this study: one consisting of images of leaves from six grapevine classes/varieties captured against a white background, and another including images of leaves from 12 grapevine classes/varieties photographed in natural environments. In essence, this paper presents not only a pivotal solution for the scarcity of skilled ampelographers, but also a promising path for sustaining the wine industry's added value through robust certification and authenticity. As the digital era unfolds, the framework laid out in this paper invites the integration of other grapevine datasets to further enrich and empower AI-driven ampelography.

- ❖ Thesis-paper alignment: this paper provided an opportunity to explore one of the datasets used in this study (GLI) and to examine and compare a range of DL processes, including various data conditions and arrangements. These analyses supported the rationale for the DL-oriented optimization strategies proposed in Chapter 3 and laid a foundation for the developments presented in Chapters 4 and 5.
- ❖ Index and metrics: *COMPAG* is indexed by Scopus and Science Citation Index Expanded (SCIE), among others. Its CiteScore is 15.1. In the 2024 Journal Citation Report (JCR, Clarivate) evaluation, an impact factor of 8.9 was attributed. This journal is also Q1 in Computer Science Applications.

- **"Automatic Optimization of Deep Learning Training through Feature-Aware-Based Dataset Splitting"**
  - ❖ Venue: this paper was published in the peer-reviewed journal *Algorithms* (ISSN 1999-4893), Vol. 17(3):106 (February 2024). Edited by Multidisciplinary Digital Publishing Institute (MDPI). doi: https://doi.org/10.3390/a17030106.
  - ❖ Abstract: The proliferation of classification-capable artificial intelligence (AI) across a wide range of domains (e.g., agriculture, construction, etc.) has been allowed to optimize and complement several tasks, typically operationalized by humans. The computational training that allows providing such support is frequently hindered by various challenges related to datasets, including the scarcity of examples and imbalanced class distributions, which have detrimental effects on the production of accurate models. For a proper approach to these

challenges, strategies smarter than the traditional brute force-based K-fold cross-validation or the naivety of hold-out are required, with the following main goals in mind: (1) carrying out one-shot, close-to-optimal data arrangements, accelerating conventional training optimization; and (2) aiming at maximizing the capacity of inference models to its fullest extent while relieving computational burden. To that end, in this paper, two image-based feature-aware dataset splitting approaches are proposed, hypothesizing a contribution towards attaining classification models that are closer to their full inference potential. Both rely on strategic image harvesting: while one of them hinges on weighted random selection out of a feature-based clusters set, the other involves a balanced picking process from a sorted list that stores data features' distances to the centroid of a whole feature space. Comparative tests on datasets related to grapevine leaves phenotyping and bridge defects showcase promising results, highlighting a viable alternative to K-fold cross-validation and hold-out methods.

- ❖ Thesis-paper alignment: this work is a direct result of Chapter 4, which represents the details of introducing and evaluating two novel one-shot dataset-splitting techniques (FCDIS and FSCIS) designed to overcome the limitations of conventional methods (e.g., KFCV). Some of the contributions of Chapter 6 are also reflected in this paper.
- ❖ Index and metrics: Algorithms is indexed in Web of Science (ESCI), Scopus, Ei Compendex, Inspec, PubMed, and DOAJ, with a 2024 JCR Impact Factor of 2.1 and a CiteScore of 4.5. It is also Q2 in the fields of Computational Mathematics and Computational Theory, and Mathematics and Numerical Analysis.

- **"Unbalancing Datasets to Enhance CNN Models Learnability: A Class-Wise Metrics-Based Closed-Loop Strategy Proposal".**

  - ❖ Venue: this paper was published in the peer-reviewed journal IEEE Access (ISSN 2169-3536), Volume 13 (pp. 57485-57503). Edited by the Institute of Electrical and Electronics Engineers (IEEE). doi: https://doi.org/10.1109/ACCESS.2025.3554395.
  - ❖ Abstract: Developing deep learning models often involves working with balanced datasets. However, in real-world scenarios, class equilibrium is rarely observed. To address this, oversampling and undersampling techniques are commonly employed, with appropriate monitoring and countermeasures to mitigate the risk of overfitting. Yet, even when datasets are balanced, they can still introduce biases, leading to better classification performance for

certain classes over others. One significant cause of such biases is class underrepresentation, which is challenging to control and often arises from the acquisition of uneven feature distributions influenced by endogenous factors and environmental conditions. In such cases, random data augmentation can inadvertently amplify these discrepancies, exacerbating the issue of underrepresentation. To address this challenge, this paper proposes a metrics-based closed-loop (MbCL) approach that strategically unbalances the dataset to enhance class-wise performance and model generalization in line with the training process. The proposed method iteratively adjusts and improves model performance across different classes by employing data augmentation techniques, including both classical and GAN-based methods, to mitigate classes' underrepresentation. To compare regular (linear) and MbCL-based approaches, a couple of contextually distinct datasets were considered to broaden consistency analysis across some diversity. Using these datasets, 72 models with varying configurations – including different CNN architectures, initial LRs, and optimizers – were initially trained and then evaluated against imagery test sets. In this first assessment, MbCL methods matched or outperformed linear training in approximately 85% of the cases. Additionally, all the models were tested on an external image set acquired under different contextual conditions but with matching labels to the training datasets. The results showed that the proposed MbCL methods consistently outperformed linear training, achieving top-accuracies of 32% and 72% for both the encompassed dataset-related contexts, therefore, also indicating that classic and GAN-based augmentation approaches had a positive impact on inference performances as resampling strategies. Furthermore, complementary indicators, including gradient-weighted class activation mapping (Grad-CAM) and intersection over union (IoU), were analyzed to explore the relationship between performance and visualizable features.

- Thesis-paper alignment: this publication is directly related to the content presented in Chapter 5, which introduces the proposed MbCL methodology. The approach is designed to optimize model training through iterative validation-based feedback, enhancing both convergence and generalization. Some of the contributions of Chapter 6 are also reflected in this paper.
- Index and metrics: IEEE Access is indexed to important services such as Directory of Open Access Journals (DOAJ), EBSCOhost, Ei Compendex, Web of Science, and Scopus. With a 2024 JCR Impact Factor of 3.6 and a Cite Score of 9, IEEE Access is Q2 in multiple scientific areas, including Computer Science and Engineering (miscellaneous).

6.4 Possible Future Research Directions

While the results of this research demonstrate the potential of the proposed methods, several areas remain for exploration and improvement. These improvements focus on scalability, efficiency, and expanding the applicability of the proposed splitting techniques and the MbCL method, as well as exploring synergies with emerging approaches.

First, scalability and robustness can be evaluated by testing larger and more diverse datasets. Expanding the scope of experimentation can be useful to assess whether the benefits observed in this study — particularly in terms of class-aware learning and spatial precision — can still be observed when scaling up to datasets with increased complexity, variety, and volume.

Another area for improvement lies in optimizing the computational efficiency of cGANs. Although GANs can significantly improve the representativeness of underrepresented classes, their computational demands are non-trivial. Fine-tuning cGAN architectures to make them more efficient — whether through pruning, quantization, or other optimization techniques — could reduce the resource burden associated with their operations. Alternatively, to circumvent hardware limitations, offloading the training responsibilities to cloud-based services with access to high-performance computing resources can also be a viable solution. This would ensure that the proposed methods maintain their performance advantages without placing excessive demands on local computational resources. Exploring other generative approaches based on, for example, diffusion models, and benchmarking them with GANs would also be a viable research line to assess not only computation and time performance but also the realism of the generated data for incorporation in the MbCL pipelines.

In addition to optimizing the core components, there is a potential to enhance its performance through the exploration of additional feature extraction methods. Leveraging advanced feature extraction techniques, such as those used in object detection architectures like YOLO (You Only Look Once) or MobileNet, could also be interesting to assess capabilities in providing richer and more discriminative features during the learning process, seeking overall improvements to the proposed methods. By incorporating more advanced feature extraction, the proposed methods could adapt more effectively to a wider range of visual patterns and tasks.

Building on the notion of augmentation, an exciting area for future research involves exploring different augmentation techniques to compare their efficacy against MbCL-CA and MbCL-GA. Understanding how other state-of-the-art augmentation methods impact class-wise inferences and generalization capabilities could provide valuable insights into how to further refine the MbCL technique. A promising approach to

investigate is AutoAugment, which automates the selection of augmentation policies based on performance feedback. Adapting AutoAugment for class-wise policy augmentation within MbCL could establish a more unified framework for dataset-specific and task-specific improvements. Further expanding the scope of augmentation techniques can also involve investigating the integration of ViTs, which are rapidly gaining attention as powerful alternatives to CNNs, especially in contexts involving larger-scale data or more complex relationships within the image data.

Exploring how MbCL can be applied to ViTs would be an interesting research line, especially involving significantly extensive datasets. This line could not only improve the MbCL flexibility in terms of supported data volumes but also enable its use across a broader spectrum of domains/contexts and in combination with an extended set of DL-oriented supervised learning approaches, including modern ones (e.g., ViTs).

REFERENCES

[1] G. Pinto, Z. Wang, A. Roy, T. Hong, and A. Capozzoli, "Transfer learning for smart buildings: A critical review of algorithms, applications, and future perspectives," *Advances in Applied Energy*, vol. 5, Feb. 2022, doi: 10.1016/j.adapen.2022.100084.

[2] S. Shahrabadi *et al.*, "Digital Tools for Aircraft Maintenance: Prototyping Location-Aware AOI for Engine Assessment and Cable Routing Solutions," in *2023 International Conference on Graphics and Interaction (ICGI)*, 2023, pp. 1–11. doi: 10.1109/ICGI60907.2023.10452715.

[3] H. E. Kim, A. Cosa-Linan, N. Santhanam, M. Jannesari, M. E. Maros, and T. Ganslandt, "Transfer learning for medical image classification: a literature review," Dec. 01, 2022, *BioMed Central Ltd*. doi: 10.1186/s12880-022-00793-7.

[4] S. Shahrabadi *et al.*, "Image-based Lung Analysis in the Context of Digital Pathology: a Brief Review," *Procedia Comput. Sci.*, vol. 239, pp. 2168–2175, 2024, doi: https://doi.org/10.1016/j.procs.2024.06.405.

[5] Q. Gao, T. Long, and Z. Zhou, "Mineral identification based on natural feature-oriented image processing and multi-label image classification," *Expert Syst. Appl.*, vol. 238, Mar. 2024, doi: 10.1016/j.eswa.2023.122111.

[6] Y. J. Yoo, G. Yang, K. Park, Y. Hyun, and S. Jeong, "Extendable machine tool wear monitoring process using image segmentation based deep learning model and automatic detection of depth of cut line," *Eng. Appl. Artif. Intell.*, vol. 135, Sep. 2024, doi: 10.1016/j.engappai.2024.108570.

[7] A. L. Wozniak, S. Segura, and R. Mazo, "Robustness Assessment of AI-Based 2D Object Detection Systems: A Method and Lessons Learned from Two Industrial Cases," *Electronics (Switzerland)*, vol. 13, no. 7, Apr. 2024, doi: 10.3390/electronics13071368.

[8] J. Su *et al.*, "Large Language Models for Forecasting and Anomaly Detection: A Systematic Literature Review," Feb. 2024, [Online]. Available: http://arxiv.org/abs/2402.10350

[9] S. A. Prome, N. A. Ragavan, M. R. Islam, D. Asirvatham, and A. J. Jegathesan, "Deception detection using machine learning (ML) and deep learning (DL) techniques: A systematic review," *Natural Language Processing Journal*, vol. 6, p. 100057, Mar. 2024, doi: 10.1016/j.nlp.2024.100057.

[10] R. Pugliese, S. Regondi, and R. Marini, "Machine learning-based approach: Global trends, research directions, and regulatory standpoints," Dec. 01, 2021, *KeAi Communications Co.* doi: 10.1016/j.dsm.2021.12.002.

[11] N. Sharma, R. Sharma, and N. Jindal, "Machine Learning and Deep Learning Applications-A Vision," *Global Transitions Proceedings*, vol. 2, no. 1, pp. 24–28, Jun. 2021, doi: 10.1016/j.gltp.2021.01.004.

[12] A. Hernández-Blanco, B. Herrera-Flores, D. Tomás, and B. Navarro-Colorado, "A Systematic Review of Deep Learning Approaches to Educational Data Mining," 2019, *Hindawi Limited*. doi: 10.1155/2019/1306039.

[13] C. Shen, "A Transdisciplinary Review of Deep Learning Research and Its Relevance for Water Resources Scientists," Nov. 01, 2018, *Blackwell Publishing Ltd*. doi: 10.1029/2018WR022643.

[14] I. H. Sarker, “Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions,” Nov. 01, 2021, *Springer*. doi: 10.1007/s42979-021-00815-1.

[15] T. Talaei Khoei, H. Ould Slimane, and N. Kaabouch, “Deep learning: systematic review, models, challenges, and research directions,” Nov. 01, 2023, *Springer Science and Business Media Deutschland GmbH*. doi: 10.1007/s00521-023-08957-4.

[16] İ. Yazici, I. Shayea, and J. Din, “A survey of applications of artificial intelligence and machine learning in future mobile networks-enabled systems,” Aug. 01, 2023, *Elsevier B.V.* doi: 10.1016/j.jestch.2023.101455.

[17] G. Obaido *et al.*, “Supervised machine learning in drug discovery and development: Algorithms, applications, challenges, and prospects,” *Machine Learning with Applications*, vol. 17, p. 100576, Sep. 2024, doi: 10.1016/j.mlwa.2024.100576.

[18] J. E. van Engelen and H. H. Hoos, “A survey on semi-supervised learning,” *Mach. Learn.*, vol. 109, no. 2, pp. 373–440, Feb. 2020, doi: 10.1007/s10994-019-05855-6.

[19] P. Singh *et al.*, “Shifting to machine supervision: annotation-efficient semi and self-supervised learning for automatic medical image segmentation and classification,” *Sci. Rep.*, vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-61822-9.

[20] K. Liu, J. Liu, and S. Liu, “Enhanced Semi-Supervised Medical Image Classification Based on Dynamic Sample Reweighting and Pseudo-Label Guided Contrastive Learning (DSRPGC),” *Mathematics*, vol. 12, no. 22, Nov. 2024, doi: 10.3390/math12223572.

[21] J. Wang and F. Biljecki, “Unsupervised machine learning in urban studies: A systematic review of applications,” *Cities*, vol. 129, Oct. 2022, doi: 10.1016/j.cities.2022.103925.

[22] A. F. Vermeulen, “Unsupervised Learning: Using Unlabeled Data,” in *Industrial Machine Learning: Using Artificial Intelligence as a Transformational Disruptor*, Berkeley, CA: Apress, 2020, pp. 181–206. doi: 10.1007/978-1-4842-5316-8_6.

[23] R. van Leeuwen and G. Koole, “Data-driven market segmentation in hospitality using unsupervised machine learning,” *Machine Learning with Applications*, vol. 10, p. 100414, Dec. 2022, doi: 10.1016/j.mlwa.2022.100414.

[24] N. Hicham and S. Karim, “Analysis of Unsupervised Machine Learning Techniques for an Efficient Customer Segmentation using Clustering Ensemble and Spectral Clustering,” *International Journal of Advanced Computer Science and Applications*, vol. 13, Jun. 2022, doi: 10.14569/IJACSA.2022.0131016.

[25] S. M. Hemadharshini, R. Kanchana Devi, S. Rajakumari, and R. Adline Freeda, “E-commerce Customer Segmentation by Unsupervised Learning,” in *Soft Computing and Signal Processing*, V. S. Reddy, J. Wang, and K. T. V Reddy, Eds., Singapore: Springer Nature Singapore, 2024, pp. 285–296. doi: https://doi.org/10.1007/978-981-99-8628-6_25.

[26] N. Verbeeck, R. M. Caprioli, and R. Van de Plas, “Unsupervised machine learning for exploratory data analysis in imaging mass spectrometry,” May 01, 2020, *John Wiley and Sons Inc.* doi: 10.1002/mas.21602.

[27] M. Â. L. Moreira *et al.*, “Defense Perception in the Geopolitical Scope: An exploratory study through unsupervised machine learning,” in *Procedia Computer Science*, Elsevier B.V., 2023, pp. 689–696. doi: 10.1016/j.procs.2023.08.039.

[28] B. Memarian and T. Doleck, “A scoping review of reinforcement learning in education,” *Computers and Education Open*, vol. 6, p. 100175, Jun. 2024, doi: 10.1016/j.caeo.2024.100175.

[29] M. L. and M. A. W. Kaelbling Leslie Pack and Littman, “An Introduction to Reinforcement Learning,” in *The Biology and Technology of Intelligent Autonomous Agents*, L. Steels, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 90–127. doi: https://doi.org/10.1007/978-3-642-79629-6_5.

[30] J. Kober Jens and Peters, “Reinforcement Learning in Robotics: A Survey,” in *Learning Motor Skills: From Algorithms to Robot Experiments*, Cham: Springer International Publishing, 2014, pp. 9–67. doi: 10.1007/978-3-319-03194-1_2.

[31] T. Zhang and H. Mo, “Reinforcement learning for robot research: A comprehensive review and open issues,” 2021, *SAGE Publications Inc.* doi: 10.1177/17298814211007305.

[32] T. Toner, M. Saez, D. M. Tilbury, and K. Barton, “Opportunities and challenges in applying reinforcement learning to robotic manipulation: An industrial case study,” *Manuf. Lett.*, vol. 35, pp. 1019–1030, 2023, doi: https://doi.org/10.1016/j.mfglet.2023.08.055.

[33] I. Szita, “Reinforcement Learning in Games,” in *Reinforcement Learning: State-of-the-Art*, M. Wiering Marco and van Otterlo, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 539–577. doi: 10.1007/978-3-642-27645-3_17.

[34] K. Souchleris, G. K. Sidiropoulos, and G. A. Papakostas, “Reinforcement Learning in Game Industry—Review, Prospects and Challenges,” Feb. 01, 2023, *MDPI*. doi: 10.3390/app13042443.

[35] G. Lalitha, B. Rajalakshmi, M. Gupta, Praveen, V. Kumar, and A. A. Alwan, “Reinforcement Learning for Real-Time Decision-Making in autonomous air combat systems,” in *2023 10th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, 2023, pp. 1593–1598. doi: 10.1109/UPCON59197.2023.10434820.

[36] E. Skordilis and R. Moghaddass, “A deep reinforcement learning approach for real-time sensor-driven decision making and predictive analytics,” *Comput. Ind. Eng.*, vol. 147, Sep. 2020, doi: 10.1016/j.cie.2020.106600.

[37] Z. Pei, A. M. Rojas-Arevalo, F. J. de Haan, N. Lipovetzky, and E. A. Moallemi, “Reinforcement learning for decision-making under deep uncertainty,” *J. Environ. Manage.*, vol. 359, May 2024, doi: 10.1016/j.jenvman.2024.120968.

[38] J. Waring, C. Lindvall, and R. Umeton, “Automated machine learning: Review of the state-of-the-art and opportunities for healthcare,” Apr. 01, 2020, *Elsevier B.V.* doi: 10.1016/j.artmed.2020.101822.

[39] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” May 27, 2015, *Nature Publishing Group*. doi: 10.1038/nature14539.

[40] N. A. Saputra, L. S. Riza, A. Setiawan, and I. Hamidah, “A Systematic Review for Classification and Selection of Deep Learning Methods,” *Decision Analytics Journal*, vol. 12, Sep. 2024, doi: 10.1016/j.dajour.2024.100489.

[41] M. D. Zakir Hossain, F. Sohel, M. F. Shiratuddin, and H. Laga, “A comprehensive survey of deep learning for image captioning,” Nov. 30, 2019, *Association for Computing Machinery*. doi: 10.1145/3295748.

[42] T. Jiang, J. L. Gradus, and A. J. Rosellini, “Supervised Machine Learning: A Brief Primer,” *Behav. Ther.*, vol. 51, no. 5, pp. 675–687, Sep. 2020, doi: 10.1016/j.beth.2020.05.002.

[43] S. Rezvani and X. Wang, “A broad review on class imbalance learning techniques,” Aug. 01, 2023, *Elsevier Ltd*. doi: 10.1016/j.asoc.2023.110415.

[44] H. He and E. A. Garcia, “Learning from imbalanced data,” *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009, doi: 10.1109/TKDE.2008.239.

[45] P. Mooijman, C. Catal, B. Tekinerdogan, A. Lommen, and M. Blokland, “The effects of data balancing approaches: A case study,” *Appl. Soft Comput.*, vol. 132, Jan. 2023, doi: 10.1016/j.asoc.2022.109853.

[46] W. Chen, K. Yang, Z. Yu, Y. Shi, and C. L. P. Chen, “A survey on imbalanced learning: latest research, applications and future directions,” *Artif. Intell. Rev.*, vol. 57, no. 6, Jun. 2024, doi: 10.1007/s10462-024-10759-6.

[47] A. Mumuni and F. Mumuni, “Data augmentation: A comprehensive survey of modern approaches,” Dec. 01, 2022, *Elsevier B.V.* doi: 10.1016/j.array.2022.100258.

[48] K. Medvedieva, T. Tosi, E. Barbierato, and A. Gatti, “Balancing the Scale: Data Augmentation Techniques for Improved Supervised Learning in Cyberattack Detection,” *Eng*, vol. 5, no. 3, pp. 2170–2205, Sep. 2024, doi: 10.3390/eng5030114.

[49] C. Shorten, T. M. Khoshgoftaar, and B. Furht, “Text Data Augmentation for Deep Learning,” *J. Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00492-0.

[50] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” *J. Big Data*, vol. 6, no. 1, Dec. 2019, doi: 10.1186/s40537-019-0197-0.

[51] R. Zhang, B. Zhou, C. Lu, and M. Ma, “The Performance Research of the Data Augmentation Method for Image Classification,” *Math. Probl. Eng.*, vol. 2022, 2022, doi: 10.1155/2022/2964829.

[52] E. K. Kim, H. Lee, J. Y. Kim, and S. Kim, “Data augmentation method by applying color perturbation of inverse PSNR and geometric transformations for object recognition based on deep learning,” *Applied Sciences (Switzerland)*, vol. 10, no. 11, Jun. 2020, doi: 10.3390/app10113755.

[53] Y. Lu, D. Chen, E. Olaniyi, and Y. Huang, “Generative adversarial networks (GANs) for image augmentation in agriculture: A systematic review,” *Comput. Electron. Agric.*, vol. 200, Sep. 2022, doi: 10.1016/j.compag.2022.107208.

[54] C. Abou Akar, R. Massih, A. Yaghi, J. Khalil, M. Kamradt, and A. Makhoul, “Generative Adversarial Network Applications in Industry 4.0: A Review,” *Int. J. Comput. Vis.*, vol. 132, pp. 1–60, Oct. 2024, doi: 10.1007/s11263-023-01966-9.

[55] S. Sabnam and S. Rajagopal, “Application of generative adversarial networks in image, face reconstruction and medical imaging: challenges and the current progress,” *Comput. Methods Biomech. Biomed. Eng. Imaging Vis.*, vol. 12, no. 1, 2024, doi: 10.1080/21681163.2024.2330524.

[56] S. D. Asgari, E. Mohammadi, A. Makui, and M. Jafari, “Data-driven robust optimization based on position-regulated support vector clustering,” *J. Comput. Sci.*, vol. 76, Mar. 2024, doi: 10.1016/j.jocs.2024.102210.

[57] K. Ukaoha and E. Igodan, “Architecture Optimization Model for the Deep Neural Network,” *International Journal of Intelligent Computing and Information Sciences*, vol. 19, no. 2, pp. 1–16, Oct. 2019, doi: 10.21608/ijicis.2019.96101.

[58] Suhartono, N. Suhermi, and D. D. Prastyo, “Design of Experiment to Optimize the Architecture of Deep Learning for Nonlinear Time Series Forecasting,” in *Procedia Computer Science*, Elsevier B.V., 2018, pp. 269–276. doi: 10.1016/j.procs.2018.10.528.

[59] M. A. K. Raiaan *et al.*, “A systematic review of hyperparameter optimization techniques in Convolutional Neural Networks,” Jun. 01, 2024, *Elsevier Inc.* doi: 10.1016/j.dajour.2024.100470.

[60] A. A. Chowdhury, A. Das, K. K. S. Hoque, and D. Karmaker, “A Comparative Study of Hyperparameter Optimization Techniques for Deep Learning,” 2022, pp. 509–521. doi: 10.1007/978-981-19-0332-8_38.

[61] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, “Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization,” *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019, doi: https://doi.org/10.11989/JEST.1674-862X.80904120.

[62] Z. Nematzadeh, R. Ibrahim, and A. Selamat, “Comparative studies on breast cancer classifications with k-fold cross validations using machine learning techniques,” in *2015 10th Asian Control Conference (ASCC)*, 2015, pp. 1–6. doi: 10.1109/ASCC.2015.7244654.

[63] K. Pal and B. V. Patel, “Data Classification with k-fold Cross Validation and Holdout Accuracy Estimation Methods with 5 Different Machine Learning Techniques,” in *Proceedings of the 4th International Conference on Computing Methodologies and Communication, ICCMC 2020*, Institute of Electrical and Electronics Engineers Inc., Mar. 2020, pp. 83–87. doi: 10.1109/ICCMC48092.2020.ICCMC-00016.

[64] Z. Qu, J. Xu, Z. Wang, R. Chi, and H. Liu, “Prediction of electricity generation from a combined cycle power plant based on a stacking ensemble and its hyperparameter optimization with a grid-search method,” *Energy*, vol. 227, Jul. 2021, doi: 10.1016/j.energy.2021.120309.

[65] P. Zhou, J. Feng, C. Ma, C. Xiong, S. Hoi, and E. Weinan, “Towards theoretically understanding why SGD generalizes better than ADAM in deep learning,” in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, in NIPS ’20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[66] L. Zhu, Z. Xie, L. Liu, B. Tao, and W. Tao, “IoU-uniform R-CNN: Breaking through the limitations of RPN,” *Pattern Recognit.*, vol. 112, Apr. 2021, doi: 10.1016/j.patcog.2021.107816.

[67] K. L. Radke *et al.*, “Adaptive IoU Thresholding for Improving Small Object Detection: A Proof-of-Concept Study of Hand Erosions Classification of Patients with Rheumatic Arthritis on X-ray Images,” *Diagnostics*, vol. 13, no. 1, Jan. 2023, doi: 10.3390/diagnostics13010104.

[68] S. Ali *et al.*, “Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence,” *Information Fusion*, vol. 99, Nov. 2023, doi: 10.1016/j.inffus.2023.101805.

[69] W. Saeed and C. Omlin, “Explainable AI (XAI): A systematic meta-survey of current challenges and future opportunities,” *Knowl. Based. Syst.*, vol. 263, Mar. 2023, doi: 10.1016/j.knosys.2023.110273.

[70] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, “Explainable ai: A review of machine learning interpretability methods,” Jan. 01, 2021, *MDPI AG*. doi: 10.3390/e23010018.

[71] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization,” Oct. 2016, doi: 10.1007/s11263-019-01228-7.

[72] M. Bahri, F. Salutari, A. Putina, and M. Sozio, “AutoML: state of the art with a focus on anomaly detection, challenges, and research directions,” Aug. 01, 2022, *Springer Science and Business Media Deutschland GmbH*. doi: 10.1007/s41060-022-00309-0.

[73] I. Salehin *et al.*, “AutoML: A systematic review on automated machine learning with neural architecture search,” *Journal of Information and Intelligence*, vol. 2, no. 1, pp. 52–81, Jan. 2024, doi: 10.1016/j.jiixd.2023.10.002.

[74] R. Barbudo, S. Ventura, and J. R. Romero, “Eight years of AutoML: categorisation, review and trends,” Dec. 01, 2023, *Springer Science and Business Media Deutschland GmbH*. doi: 10.1007/s10115-023-01935-1.

[75] J. Bergstra, J. B. Ca, and Y. B. Ca, “Random Search for Hyper-Parameter Optimization Yoshua Bengio,” 2012. [Online]. Available: http://scikit-learn.sourceforge.net.

[76] S. Brown, “Why it’s time for data-centric artificial intelligence,” Jun. 07, 2022. [Online]. Available: https://mitsloan.mit.edu/ideas-made-to-matter/why-its-time-data-centric-artificial-intelligence

[77] B. Hüttenrauch, “Limitations of data augmentation and outlook,” in *Targeting Using Augmented Data in Database Marketing: Decision Factors for Evaluating External Sources*, Wiesbaden: Springer Fachmedien Wiesbaden, 2016, pp. 279–290. doi: 10.1007/978-3-658-14577-4_8.

[78] E. Goceri, “Medical image data augmentation: techniques, comparisons and interpretations,” *Artif. Intell. Rev.*, vol. 56, no. 11, pp. 12561–12605, Nov. 2023, doi: 10.1007/s10462-023-10453-z.

[79] S. T. March and G. F. Smith, “Design and natural science research on information technology,” *Decis. Support Syst.*, vol. 15, no. 4, pp. 251–266, 1995, doi: 10.1016/0167-9236(94)00041-2.

[80] V. Vaishnavi and B. Kuechler, “Design Science Research in Information Systems,” *Association for Information Systems*, Oct. 2004.

[81] H. Takeda, P. Veerkamp, T. Tomiyama, and H. Yoshikawa, “Modeling Design Processes,” 1990. doi: DOI:https://doi.org/10.1609/aimag.v11i4.855.

[82] S. A. Carlsson, “Towards an Information Systems Design Research Framework: A Critical Realist Perspective,” 2006.

[83] A. Hevner *et al.*, “Design Science in Information Systems Research,” *Management Information Systems Quarterly*, vol. 28, p. 75, Jun. 2004.

[84] J. G. Walls, G. R. Widmeyer, and O. Sawy, “Assessing information system design theory in perspective: How useful was our 1992 initial rendition?,” *Journal of Information Technology Theory and Application*, vol. 6, pp. 43–58, Jun. 2004.

[85] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A design science research methodology for information systems research,” *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, Dec. 2007, doi: 10.2753/MIS0742-1222240302.

[86] C. A. Mertler, R. A. Vannatta, and K. N. LaVenia, *Advanced and Multivariate Statistical Methods: Practical Application and Interpretation*. Taylor and Francis, 2021. doi: 10.4324/9781003047223.

[87] J. A. Maxwell, *Qualitative Research Design: An Interactive Approach*, 3rd ed., vol. 41. 2013.

[88] J. Creswell and T. Guetterman, “Educational research: Planning, conducting, and evaluating quantitative and qualitative research,” 2019.

[89] D. Patil, N. L. Rane, P. Desai, and J. Rane, “Machine learning and deep learning: Methods, techniques, applications, challenges, and future research opportunities,” in *Trustworthy Artificial Intelligence in Industry and Society*, D. Patil, N. L. Rane, P. Desai, and J. Rane, Eds., Deep Science Publishing, 2024, pp. 28–81. doi: 10.70593/978-81-981367-4-9_2.

[90] A. Ebadi, M. Kaur, and Q. Liu, “Hyperparameter optimization and neural architecture search algorithms for graph Neural Networks in cheminformatics,” *Comput. Mater. Sci.*, vol. 254, p. 113904, 2025, doi: https://doi.org/10.1016/j.commatsci.2025.113904.

[91] Y. J. Cruz, A. Villalonga, F. Castaño, M. Rivas, and R. E. Haber, “Automated machine learning methodology for optimizing production processes in small and medium-sized enterprises,” *Operations Research Perspectives*, vol. 12, p. 100308, 2024, doi: https://doi.org/10.1016/j.orp.2024.100308.

[92] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning Transferable Architectures for Scalable Image Recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, Dec. 2018, pp. 8697–8710. doi: 10.1109/CVPR.2018.00907.

[93] F. Feurer Matthias and Hutter, “Hyperparameter Optimization,” in *Automated Machine Learning: Methods, Systems, Challenges*, L. and V. J. Hutter Frank and Kotthoff, Ed., Cham: Springer International Publishing, 2019, pp. 3–33. doi: 10.1007/978-3-030-05318-5_1.

[94] J. Kim, W. Jung, J. An, H. J. Oh, and J. Park, “Self-optimization of training dataset improves forecasting of cyanobacterial bloom by machine learning,” *Science of The Total Environment*, vol. 866, p. 161398, 2023, doi: https://doi.org/10.1016/j.scitotenv.2023.161398.

[95] D. H. Hagos *et al.*, “Scalable Artificial Intelligence for Earth Observation Data Using Hopsworks,” *Remote Sens. (Basel).*, vol. 14, no. 8, 2022, doi: 10.3390/rs14081889.

[96] P. A. Apellániz, J. Parras, and S. Zazo, “Improving Synthetic Data Generation Through Federated Learning in Scarce and Heterogeneous Data Scenarios,” *Big Data and Cognitive Computing*, vol. 9, no. 2, 2025, doi: 10.3390/bdcc9020018.

[97] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, “A review of convolutional neural networks in computer vision,” *Artif. Intell. Rev.*, vol. 57, no. 4, p. 99, 2024, doi: 10.1007/s10462-024-10721-6.

[98] B. Zoph and Q. V Le, “Neural Architecture Search with Reinforcement Learning,” 2017. [Online]. Available: https://arxiv.org/abs/1611.01578

[99] J. Mockus, “Bayesian global optimizationBayesian Global Optimization,” in *Encyclopedia of Optimization*, P. M. Floudas Christodoulos A. and Pardalos, Ed., Boston, MA: Springer US, 2001, pp. 123–127. doi: 10.1007/0-306-48332-7_24.

[100] T. Bartz-Beielstein, J. Branke, J. Mehnen, and O. Mersmann, “Evolutionary Algorithms,” *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 4, Jun. 2014, doi: 10.1002/widm.1124.

[101] Y. Lecun, E. Bottou, Y. Bengio, and P. Haffner, “Gradient-Based Learning Applied to Document Recognition,” 1998.

[102] E. Elyan, C. F. Moreno-Garcia, and C. Jayne, “CDSMOTE: class decomposition and synthetic minority class oversampling technique for imbalanced-data classification,” *Neural Comput. Appl.*, vol. 33, no. 7, pp. 2839–2851, Apr. 2021, doi: 10.1007/s00521-020-05130-z.

[103] Q. Nguyen *et al.*, “Influence of Data Splitting on Performance of Machine Learning Models in Prediction of Shear Strength of Soil,” *Math. Probl. Eng.*, vol. 2021, Jun. 2021, doi: 10.1155/2021/4832864.

[104] B. Gautam and M. Arashpour, “Advanced data augmentation techniques to enhance instance segmentation dataset for construction and demolition waste management,” *Waste Management*, vol. 200, p. 114744, 2025, doi: https://doi.org/10.1016/j.wasman.2025.114744.

[105] P. Sharma, M. Kumar, H. K. Sharma, and S. M. Biju, “Generative adversarial networks (GANs): Introduction, Taxonomy, Variants, Limitations, and Applications,” *Multimed. Tools Appl.*, vol. 83, no. 41, pp. 88811–88858, 2024, doi: 10.1007/s11042-024-18767-y.

[106] E. Bisong, “Google AutoML: Cloud Vision,” 2019, pp. 581–598. doi: 10.1007/978-1-4842-4470-8_42.

[107] E. LeDell and S. Poirier, “H2O AutoML: Scalable Automatic Machine Learning,” 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:221338558

[108] M. L. and M. A. W. Kaelbling Leslie Pack and Littman, “An Introduction to Reinforcement Learning,” in *The Biology and Technology of Intelligent Autonomous Agents*, L. Steels, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 90–127. doi: DOI:10.4018/978-1-60960-165-2.ch004.

[109] B. Sekeroglu, R. Abiyev, A. Ilhan, M. Arslan, and J. B. Idoko, “Systematic Literature Review on Machine Learning and Student Performance Prediction: Critical Gaps and Possible Remedies,” *Applied Sciences*, vol. 11, no. 22, 2021, doi: 10.3390/app112210907.

[110] K. Kolasa, B. Admassu, M. Hołownia-Voloskova, K. J. Kędzior, J.-E. Poirrier, and S. Perni, “Systematic reviews of machine learning in healthcare: a literature review,” *Expert Rev. Pharmacoecon. Outcomes Res.*, vol. 24, no. 1, pp. 63–115, 2024, doi: 10.1080/14737167.2023.2279107.

[111] A. Brnabic and L. M. Hess, “Systematic literature review of machine learning methods used in the analysis of real-world data for patient-provider decision making,” *BMC Med. Inform. Decis. Mak.*, vol. 21, no. 1, p. 54, 2021, doi: 10.1186/s12911-021-01403-2.

[112] C. Liu *et al.*, “Progressive Neural Architecture Search,” in *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I*, Berlin, Heidelberg: Springer-Verlag, 2018, pp. 19–35. doi: 10.1007/978-3-030-01246-5_2.

[113] H. H. and L.-B. K. Hutter Frank and Hoos, “Sequential Model-Based Optimization for General Algorithm Configuration,” in *Learning and Intelligent Optimization*, C. A. C. Coello, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 507–523.

[114] E. Real, A. Aggarwal, Y. Huang, and Q. V Le, “Regularized evolution for image classifier architecture search,” in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, in AAAI’19/IAAI’19/EAAI’19. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33014780.

[115] H. Pham, M. Y. Guan, B. Zoph, Q. V Le, and J. Dean, “Efficient Neural Architecture Search via Parameter Sharing,” 2018. [Online]. Available: https://arxiv.org/abs/1802.03268

[116] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, “Efficient architecture search by network transformation,” in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, in AAAI’18/IAAI’18/EAAI’18. AAAI Press, 2018.

[117] T. Wei, C. Wang, Y. Rui, and C. W. Chen, “Network Morphism,” 2016. [Online]. Available: https://arxiv.org/abs/1603.01670

[118] T. Elsken, J. H. Metzen, and F. Hutter, “Efficient Multi-objective Neural Architecture Search via Lamarckian Evolution,” Apr. 2018, [Online]. Available: http://arxiv.org/abs/1804.09081

[119] H. Liu, K. Simonyan, and Y. Yang, “DARTS: Differentiable Architecture Search,” in *proseeding of conference paper at ICLR*, Jun. 2018. doi: https://doi.org/10.48550/arXiv.1806.09055.

[120] H. Jin, Q. Song, and X. Hu, “Auto-Keras: An Efficient Neural Architecture Search System,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data MiningJuly*, 2019, pp. 1946–1956. doi: https://doi.org/10.1145/3292500.3330648.

[121] S. Saxena and J. Verbeek, “Convolutional neural fabrics,” in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, in NIPS’16. Red Hook, NY, USA: Curran Associates Inc., 2016, pp. 4060–4068.

[122] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, “Neural architecture optimization,” in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, in NIPS’18. Red Hook, NY, USA: Curran Associates Inc., 2018, pp. 7827–7838.

[123] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, “SMASH: One-Shot Model Architecture Search through HyperNetworks,” Aug. 2017. [Online]. Available: http://arxiv.org/abs/1708.05344

[124] D. Ha, A. Dai, and Q. V. Le, “HyperNetworks,” Sep. 2016. [Online]. Available: http://arxiv.org/abs/1609.09106

[125] B. Baker, O. Gupta, N. Naik, and R. Raskar, “Designing Neural Network Architectures using Reinforcement Learning,” Nov. 2016, doi: https://doi.org/10.48550/arXiv.1611.02167.

[126] P. J. Schweitzer and B. Gavish, “An Optimality Principle for Markovian Decision Processes,” *Journal of Mathematics Analysis and Application*, vol. 54, pp. 173–84, 1976.

[127] B. H. Shekar and G. Dagnew, “Grid search-based hyperparameter tuning and classification of microarray cancer data,” in *2019 2nd International Conference on Advanced Computational and Communication Paradigms, ICACCP 2019*, Institute of Electrical and Electronics Engineers Inc., Feb. 2019. doi: 10.1109/ICACCP.2019.8882943.

[128] B. Bischl *et al.*, “Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges,” *WIREs Data Mining and Knowledge Discovery*, vol. 13, no. 2, p. e1484, 2023, doi: https://doi.org/10.1002/widm.1484.

[129] J. Bergstra, J. B. Ca, and Y. B. Ca, “Random Search for Hyper-Parameter Optimization,” 2012. [Online]. Available: http://scikit-learn.sourceforge.net.

[130] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, “Taking the human out of the loop: A review of Bayesian optimization,” Jan. 01, 2016, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/JPROC.2015.2494218.

[131] F. Pedregosa FABIANPEDREGOSA *et al.*, “Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot,” 2011. [Online]. Available: http://scikit-learn.sourceforge.net.

[132] M. Mitchell, “Genetic algorithms: An overview,” *Complexity*, vol. 1, no. 1, pp. 31–39, 1995, doi: https://doi.org/10.1002/cplx.6130010108.

[133] S. Blume, T. Benedens, and D. Schramm, “Hyperparameter optimization techniques for designing software sensors based on artificial neural networks,” *Sensors*, vol. 21, no. 24, Dec. 2021, doi: 10.3390/s21248435.

[134] Alibrahim Hussain and A.Ludwig Simone, “Hyperparameter Optimization: Comparing GeneticAlgorithm against Grid Search and BayesianOptimization,” IEEE, 2021.

[135] C. Di Francescomarino *et al.*, “Genetic algorithms for hyperparameter optimization in predictive business process monitoring,” *Inf. Syst.*, vol. 74, pp. 67–83, May 2018, doi: 10.1016/j.is.2018.01.003.

[136] D. M. Belete and M. D. Huchaiah, “Grid search in hyperparameter optimization of machine learning models for prediction of HIV/AIDS test results,” *International Journal of Computers and Applications*, vol. 44, no. 9, pp. 875–886, 2022, doi: 10.1080/1206212X.2021.1974663.

[137] I. M. El-Hasnony, O. M. Elzeki, A. Alshehri, and H. Salem, “Multi-Label Active Learning-Based Machine Learning Model for Heart Disease Prediction,” *Sensors*, vol. 22, no. 3, Feb. 2022, doi: 10.3390/s22031184.

[138] O. R. Sanchez, M. Repello, A. Carrega, and R. Bolla, “Evaluating ML-based DDoS Detection with Grid Search Hyperparameter Optimization,” in *Proceedings of the 2021 IEEE Conference on Network Softwarization: Accelerating Network Softwarization in the Cognitive Age, NetSoft* , Institute of Electrical and Electronics Engineers Inc., Jun. 2021, pp. 402–408. doi: 10.1109/NetSoft51509.2021.9492633.

[139] C. Kim and T. Park, “Predicting Determinants of Lifelong Learning Intention Using Gradient Boosting Machine (GBM) with Grid Search,” *Sustainability (Switzerland)*, vol. 14, no. 9, May 2022, doi: 10.3390/su14095256.

[140] S. Abbasi, H. Lan, J. Choupan, N. Sheikh-Bahaei, G. Pandey, and B. Varghese, “Deep learning for the harmonization of structural MRI scans: a survey,” Dec. 01, 2024, *BioMed Central Ltd*. doi: 10.1186/s12938-024-01280-6.

[141] A. Tatar, M. Haghighi, and A. Zeinijahromi, “Experiments on image data augmentation techniques for geological rock type classification with convolutional neural networks,” *Journal of Rock Mechanics and Geotechnical Engineering*, vol. 17, no. 1, pp. 106–125, 2025, doi: https://doi.org/10.1016/j.jrmge.2024.02.015.

[142] G. M. Weiss, “Learning When Training Data are Costly: The Effect of Class Distribution on Tree Induction,” 2003.

[143] W. Ng and M. Dash, “An Evaluation of Progressive Sampling for Imbalanced Data Sets,” in *Sixth IEEE International Conference on Data Mining - Workshops (ICDMW’06)*, 2006, pp. 657–661. doi: 10.1109/ICDMW.2006.28.

[144] F. Provost, D. Jensen, and T. Oates, “Efficient progressive sampling,” in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, in KDD ’99. New York, NY, USA: Association for Computing Machinery, 1999, pp. 23–32. doi: 10.1145/312129.312188.

[145] A. Rácz, D. Bajusz, and K. Héberger, “Effect of Dataset Size and Train/Test Split Ratios in QSAR/QSPR Multiclass Classification,” *Molecules*, vol. 26, no. 4, 2021, doi: 10.3390/molecules26041111.

[146] J. Xie, L. Sun, and Y. F. Zhao, “On the Data Quality and Imbalance in Machine Learning-based Design and Manufacturing—A Systematic Review,” *Engineering*, vol. 45, pp. 105–131, 2025, doi: https://doi.org/10.1016/j.eng.2024.04.024.

[147] P. Rouzrokh *et al.*, “Mitigating Bias in Radiology Machine Learning: 1. Data Handling,” *Radiol. Artif. Intell.*, vol. 4, no. 5, Sep. 2022, doi: 10.1148/ryai.210290.

[148] S. Huang, W. Liu, F. Qi, and K. Yang, “Development and validation of a deep learning algorithm for the recognition of plant disease,” in *Proceedings - 21st IEEE International Conference on High Performance Computing and Communications, 17th IEEE International Conference on Smart City and 5th IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2019*, Institute of Electrical and Electronics Engineers Inc., Aug. 2019, pp. 1951–1957. doi: 10.1109/HPCC/SmartCity/DSS.2019.00269.

[149] Z. Reitermanová, “Data Splitting,” WDS’10 Proceedings of Contributed Papers, Ed., 2010.

[150] M. Fox and K. Schoeffmann, “The Impact of Dataset Splits on Classification Performance in Medical Videos,” in *ICMR 2022 - Proceedings of the 2022 International Conference on*

Multimedia Retrieval, Association for Computing Machinery, Inc, Jun. 2022, pp. 6–10. doi: 10.1145/3512527.3531424.

[151] A. Leibetseder *et al.*, "Lapgyn4: a dataset for 4 automatic content analysis problems in the domain of laparoscopic gynecology," Oct. 2018, pp. 357–362. doi: 10.1145/3204949.3208127.

[152] H. Shin and S. Oh, "Feature-Weighted Sampling for Proper Evaluation of Classification Models," *Applied Sciences*, vol. 11, no. 5, Feb. 2021, doi: 10.3390/app11052039.

[153] D. Kang and S. Oh, "Balanced training/test set sampling for proper evaluation of classification models," *Intelligent Data Analysis*, vol. 24, pp. 5–18, Oct. 2020, doi: 10.3233/IDA-194477.

[154] D. Eliane Birba, "A Comparative study of data splitting algorithms for machine learning model selection," 2020.

[155] F. Farias, T. Ludermir, and C. Bastos-Filho, "Similarity Based Stratified Splitting: an approach to train better classifiers," Jun. 2020. doi: 10.48550/arXiv.2010.06099.

[156] A. Nurhopipah and U. Hasanah, "Dataset Splitting Techniques Comparison For Face Classification on CCTV Images," *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, vol. 14, no. 4, p. 341, Oct. 2020, doi: 10.22146/ijccs.58092.

[157] M. J. Lakshmi and S. Nagaraja Rao, "Effect Of K Fold Cross Validation on Mri Brain Images Using Support Vector Machine Algorithm," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 7, no. 6S4, pp. 2277–3878, 2019.

[158] R. C. Sharma, K. Hara, and H. Hirayama, "A Machine Learning and Cross-Validation Approach for the Discrimination of Vegetation Physiognomic Types Using Satellite Based Multispectral and Multitemporal Data," *Scientifica (Cairo).*, vol. 2017, 2017, doi: 10.1155/2017/9806479.

[159] S. Varma and R. Simon, "Bias in error estimation when using cross-validation for model selection," *BMC Bioinformatics*, vol. 7, Feb. 2006, doi: 10.1186/1471-2105-7-91.

[160] A. Vabalas, E. Gowen, E. Poliakoff, and A. J. Casson, "Machine learning algorithm validation with a limited sample size," *PLoS One*, vol. 14, no. 11, Nov. 2019, doi: 10.1371/journal.pone.0224365.

[161] K. M. Kahloot and P. Ekler, "Algorithmic Splitting: A Method for Dataset Preparation," *IEEE Access*, vol. 9, pp. 125229–125237, 2021, doi: 10.1109/ACCESS.2021.3110745.

[162] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *The Journal of Open Source Software*, vol. 2, no. 11, p. 205, Mar. 2017, doi: 10.21105/joss.00205.

[163] X. He, D. Cai, Y. Shao, H. Bao, and J. Han, "Laplacian regularized Gaussian mixture model for data clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 9, pp. 1406–1418, 2011, doi: 10.1109/TKDE.2010.259.

[164] B. G. Amidan, T. A. Ferryman, and S. K. Cooley, "Data outlier detection using the Chebyshev theorem," in *2005 IEEE Aerospace Conference*, 2005, pp. 3814–3819. doi: 10.1109/AERO.2005.1559688.

[165] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2015. [Online]. Available: https://arxiv.org/abs/1409.1556

[166] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[167] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826. doi: 10.1109/CVPR.2016.308.

[168] Q. H. Doan, S. H. Mai, Q. T. Do, and D. K. Thai, "A cluster-based data splitting method for small sample and class imbalance problems in impact damage classification," *Appl. Soft Comput.*, vol. 120, May 2022, doi: 10.1016/j.asoc.2022.108628.

[169] P. Christias and M. Mocanu, "A machine learning framework for olive farms profit prediction," *Water (Switzerland)*, vol. 13, no. 23, Dec. 2021, doi: 10.3390/w13233461.

[170] S. K. S. Durai and M. D. Shamili, "Smart farming using Machine Learning and Deep Learning techniques," *Decision Analytics Journal*, vol. 3, p. 100041, Jun. 2022, doi: 10.1016/j.dajour.2022.100041.

[171] K. Alibabaei *et al.*, "A Review of the Challenges of Using Deep Learning Algorithms to Support Decision-Making in Agricultural Activities," Feb. 01, 2022, *MDPI*. doi: 10.3390/rs14030638.

[172] M. S. Rao, A. Singh, N. V. S. Reddy, and D. U. Acharya, "Crop prediction using machine learning," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Jan. 2022. doi: 10.1088/1742-6596/2161/1/012033.

[173] M. F. R. Suescún, "Machine learning approaches for tomato crop yield prediction in precision agriculture," Universidade Nova de Lisboa, Lisbon,Portugal, 2021.

[174] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, in KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 2623–2631. doi: 10.1145/3292500.3330701.

[175] J. Ma, C. Hu, P. Zhou, F. Jin, X. Wang, and H. Huang, "Review of Image Augmentation Used in Deep Learning-Based Material Microscopic Image Segmentation," *Applied Sciences*, vol. 13, no. 11, 2023, doi: 10.3390/app13116478.

[176] K. Faryna, J. van der Laak, and G. Litjens, "Automatic data augmentation to improve generalization of deep learning in H&E stained histopathology," *Comput. Biol. Med.*, vol. 170, p. 108018, 2024, doi: https://doi.org/10.1016/j.compbiomed.2024.108018.

[177] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V Le, "AutoAugment: Learning Augmentation Strategies From Data," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 113–123. doi: 10.1109/CVPR.2019.00020.

[178] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[179] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Jul. 2017, doi: https://doi.org/10.48550/arXiv.1707.06347.

[180] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim, "Fast AutoAugment," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA: Curran Associates Inc., 2019.

[181] J. Bergstra, B. Kégl, Y. Bengio, R. Bardenet, and Y. Bengio, "Algorithms for Hyper-Parameter Optimization," in *25th Annual Conference on Neural Information Processing Systems (NIPS)*, 2011.

[182] M. Shahrokh Esfahani and E. R. Dougherty, "Effect of separate sampling on classification accuracy," *Bioinformatics*, vol. 30, no. 2, pp. 242–250, Jan. 2014, doi: 10.1093/bioinformatics/btt662.

[183] D. R. Jones, "A Taxonomy of Global Optimization Methods Based on Response Surfaces," *Journal of Global Optimization*, vol. 21, no. 4, pp. 345–383, Dec. 2001, doi: 10.1023/A:1012771025575.

[184] R. Hataya, J. Zdenek, K. Yoshizoe, and H. Nakayama, “Faster AutoAugment: Learning Augmentation Strategies Using Backpropagation,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Science and Business Media Deutschland GmbH, 2020, pp. 1–16. doi: 10.1007/978-3-030-58595-2_1.

[185] P. Moritz *et al.*, “Ray: a distributed framework for emerging AI applications,” in *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, in OSDI’18. USA: USENIX Association, 2018, pp. 561–577.

[186] D. Ho, E. Liang, I. Stoica, P. Abbeel, and X. Chen, “Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules,” 2019. [Online]. Available: https://arxiv.org/abs/1905.05393

[187] X. Zhang, Q. Wang, J. Zhang, and Z. Zhong, “Adversarial AutoAugment,” Dec. 2019, doi: https://doi.org/10.48550/arXiv.1912.11188.

[188] H. Inoue, “Data Augmentation by Pairing Samples for Images Classification,” Jan. 2018, doi: https://doi.org/10.48550/arXiv.1801.02929.

[189] T. DeVries and G. W. Taylor, “Improved Regularization of Convolutional Neural Networks with Cutout,” Aug. 2017, [Online]. Available: http://arxiv.org/abs/1708.04552

[190] D. Baymurzina, E. Golikov, and M. Burtsev, “A review of neural architecture search,” *Neurocomputing*, vol. 474, pp. 82–93, Feb. 2022, doi: 10.1016/j.neucom.2021.12.014.

[191] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V Le, “Randaugment: Practical automated data augmentation with a reduced search space,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 3008–3017. doi: 10.1109/CVPRW50498.2020.00359.

[192] E. D. and G. G. and L. T.-Y. and S. J. and L. Q. V Zoph Barret and Cubuk, “Learning Data Augmentation Strategies for Object Detection,” in *Computer Vision – ECCV 2020*, H. and B. T. and F. J.-M. Vedaldi Andrea and Bischof, Ed., Cham: Springer International Publishing, 2020, pp. 566–583.

[193] E. D. Cubuk, B. Zoph, S. S. Schoenholz, and Q. V. Le, “Intriguing Properties of Adversarial Examples,” Nov. 2017, doi: https://doi.org/10.48550/arXiv.1711.02846.

[194] S. G. Müller and F. Hutter, “TrivialAugment: Tuning-free Yet State-of-the-Art Data Augmentation,” 2021. [Online]. Available: https://arxiv.org/abs/2103.10158

[195] T. C. LingChen, A. Khonsari, A. Lashkari, M. R. Nazari, J. S. Sambee, and M. A. Nascimento, “UniformAugment: A Search-free Probabilistic Data Augmentation Approach,” Mar. 2020, doi: https://doi.org/10.48550/arXiv.2003.14348.

[196] M. Negassi, D. Wagner, and A. Reiterer, “Smart(Sampling)Augment: Optimal and Efficient Data Augmentation for Semantic Segmentation,” *the Special Issue Algorithms for Biomedical Image Analysis and Processing*, Oct. 2021, doi: https://doi.org/10.3390/a15050165.

[197] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, “AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty,” Dec. 2019, doi: https://doi.org/10.48550/arXiv.1912.02781.

[198] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, “CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, May 2019, pp. 6023–6032.

[199] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond Empirical Risk Minimization,” Oct. 2017, doi: https://doi.org/10.48550/arXiv.1710.09412.

[200] Y. Zheng, Z. Zhang, S. Yan, and M. Zhang, “Deep AutoAugment,” Mar. 2022, doi: https://doi.org/10.48550/arXiv.2203.06172.

[201] G. and W. Y. and H. T. and R. N. M. and Y. Y. Li Yonggang and Hu, “Differentiable Automatic Data Augmentation,” in *Computer Vision – ECCV 2020*, H. and B. T. and F. J.-M. Vedaldi Andrea and Bischof, Ed., Cham: Springer International Publishing, 2020, pp. 580–595.

[202] E. Jang, S. Gu, and B. Poole, “Categorical Reparameterization with Gumbel-Softmax,” Nov. 2016, doi: https://doi.org/10.48550/arXiv.1611.01144.

[203] W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud, “Backpropagation through the Void: Optimizing control variates for black-box gradient estimation,” Oct. 2017, doi: https://doi.org/10.48550/arXiv.1711.00123.

[204] Y. Bengio, N. Léonard, and A. Courville, “Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation,” Aug. 2013, [Online]. Available: http://arxiv.org/abs/1308.3432

[205] A. Liu, Z. Huang, Z. Huang, and N. Wang, “ Direct Differentiable Augmentation Search ,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* , Los Alamitos, CA, USA: IEEE Computer Society, Oct. 2021, pp. 12199–12208. doi: 10.1109/ICCV48922.2021.01200.

[206] C. Lin *et al.*, “Online Hyper-Parameter Learning for Auto-Augmentation Strategy,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6578–6587. doi: 10.1109/ICCV.2019.00668.

[207] C.-Y. Chen and C.-H. Chang, “Hypernetwork-Based Augmentation,” Jun. 2020, doi: https://doi.org/10.48550/arXiv.2006.06320.

[208] S. Yang, W. Xiao, M. Zhang, S. Guo, J. Zhao, and F. Shen, “Image Data Augmentation for Deep Learning: A Survey,” Apr. 2022, [Online]. Available: http://arxiv.org/abs/2204.08610

[209] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein generative adversarial networks,” in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, in ICML’17. JMLR.org, 2017, pp. 214–223.

[210] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, “Generative Adversarial Networks: An Overview,” Jan. 01, 2018, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/MSP.2017.2765202.

[211] D. Evholt and O. Larsson, “Generative Adversarial Networks and Natural Language Processing for Macroeconomic Forecasting,” Royal Institute of Technology School of Engineering Sciences KTH SCI, 2020. [Online]. Available: www.kth.se/sci

[212] N. Aldausari, A. Sowmya, N. Marcus, and G. Mohammadi, “Video Generative Adversarial Networks: A Review,” Mar. 01, 2023, *Association for Computing Machinery*. doi: 10.1145/3487891.

[213] P. Shende, M. Pawar, and S. Kakde, *A Brief Review on: MRI Images Reconstruction using GAN*. 2019.

[214] H. Alqahtani, M. Kavakli-Thorne, and G. Kumar, “Applications of Generative Adversarial Networks (GANs): An Updated Review,” *Archives of Computational Methods in Engineering*, vol. 28, no. 2, pp. 525–552, Mar. 2021, doi: 10.1007/s11831-019-09388-y.

[215] S. Karthika and M. Durgadevi, “Generative Adversarial Network (GAN): A general review on different variants of GAN and applications,” in *Proceedings of the 6th International Conference on Communication and Electronics Systems, ICCES 2021*, Institute of Electrical and Electronics Engineers Inc., Jul. 2021. doi: 10.1109/ICCES51350.2021.9489160.

[216] X. Wang, H. Guo, S. Hu, M.-C. Chang, and S. Lyu, “GAN-generated Faces Detection: A Survey and New Perspectives (2022),” Feb. 2022, [Online]. Available: http://arxiv.org/abs/2202.07145

[217] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive Growing of GANs for Improved Quality, Stability, and Variation,” Oct. 2017, [Online]. Available: http://arxiv.org/abs/1710.10196

[218] A. Bulat, J. Yang, and G. Tzimiropoulos, “To Learn Image Super-Resolution, Use a GAN to Learn How to Do Image Degradation First,” in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., Cham: Springer International Publishing, 2018, pp. 187–202.

[219] C. Ledig *et al.*, “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 105–114. doi: 10.1109/CVPR.2017.19.

[220] K. and W. S. and G. J. and L. Y. and D. C. and Q. Y. and L. C. C. Wang Xintao and Yu, “ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks,” in *Computer Vision – ECCV 2018 Workshops*, S. Leal-Taixé Laura and Roth, Ed., Cham: Springer International Publishing, 2019, pp. 63–79. doi: 10.1007/978-3-030-11021-5_5.

[221] L. Ma, X. Jia, S. Georgoulis, T. Tuytelaars, and L. Van Gool, “Exemplar Guided Unsupervised Image-to-Image Translation with Semantic Consistency,” May 2018, [Online]. Available: http://arxiv.org/abs/1805.11145

[222] D. Chen, D. Xu, H. Li, N. Sebe, and X. Wang, “Group Consistent Similarity Learning via Deep CRF for Person Re-identification,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8649–8658. doi: 10.1109/CVPR.2018.00902.

[223] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2242–2251. doi: 10.1109/ICCV.2017.244.

[224] M.-Y. and B. S. and K. J. Huang Xun and Liu, “Multimodal Unsupervised Image-to-Image Translation,” in *Computer Vision – ECCV 2018*, M. and S. C. and W. Y. Ferrari Vittorio and Hebert, Ed., Cham: Springer International Publishing, 2018, pp. 179–196.

[225] X. and Y. Z. and L. T. and L. G. Yu Xiaoming and Cai, “SingleGAN: Image-to-Image Translation by a Single-Generator Network Using Multiple Generative Adversarial Learning,” in *Computer Vision – ACCV 2018*, H. and M. G. and S. K. Jawahar C.V. and Li, Ed., Cham: Springer International Publishing, 2019, pp. 341–356.

[226] H. Zhang *et al.*, “StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1947–1962, 2019, doi: 10.1109/TPAMI.2018.2856256.

[227] H. Zhang *et al.*, “StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5908–5916. doi: 10.1109/ICCV.2017.629.

[228] T. Xu *et al.*, “AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1316–1324. doi: 10.1109/CVPR.2018.00143.

[229] G. Mariani, F. Scheidegger, R. Istrate, C. Bekas, and C. Malossi, “BAGAN: Data Augmentation with Balancing GAN,” Mar. 2018, [Online]. Available: http://arxiv.org/abs/1803.09655

[230] X. Gong, S. Chang, Y. Jiang, and Z. Wang, “AutoGAN: Neural Architecture Search for Generative Adversarial Networks,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 3223–3233. doi: 10.1109/ICCV.2019.00332.

[231] T. Salimans *et al.*, "Improved Techniques for Training GANs," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf

[232] B. Zhu, X. Pan, S. vanden Broucke, and J. Xiao, "A GAN-based hybrid sampling method for imbalanced customer classification," *Inf. Sci. (N. Y).*, vol. 609, pp. 1397–1411, Sep. 2022, doi: 10.1016/j.ins.2022.07.145.

[233] A. Sharma, P. Kumar Singh, and R. Chandra, "SMOTified-GAN for class imbalanced pattern classification problems", doi: 10.1109/ACCESS.2017.DOI.

[234] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," Nov. 2014, [Online]. Available: http://arxiv.org/abs/1411.1784

[235] S. Gandhi, H. Rana, and N. Bhatt, "Conditional GANs in Image-to-Image Translation: Improving Accuracy and Contextual Relevance in Diverse Datasets," in *Procedia Computer Science*, Elsevier B.V., 2025, pp. 954–963. doi: 10.1016/j.procs.2025.01.056.

[236] P. J. Navarro, E. Salas, F. Rosique, B. Ruszczak, and M. Egea-Cortines, "Conditional generative adversarial network for segmentation and image reconstruction of grape berries in variable illumination conditions," *Expert Syst. Appl.*, vol. 287, Aug. 2025, doi: 10.1016/j.eswa.2025.128076.

[237] M. Gao, "Conditional generative adversarial networks in computer vision: an introduction and outlook," in *2nd International Conference on Artificial Intelligence, Automation, and High-Performance Computing (AIAHPC 2022)*, L. Zhu, Ed., SPIE, 2022, p. 1234804. doi: 10.1117/12.2641644.

[238] V. Nguyen, T. F. Y. Vicente, M. Zhao, M. Hoai, and D. Samaras, "Shadow Detection with Conditional Generative Adversarial Networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4520–4528. doi: 10.1109/ICCV.2017.483.

[239] S. S. Mohammed and H. G. Clarke, "Conditional image-to-image translation generative adversarial network (cGAN) for fabric defect data augmentation," *Neural Comput. Appl.*, vol. 36, no. 32, pp. 20231–20244, 2024, doi: 10.1007/s00521-024-10179-1.

[240] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5967–5976. doi: 10.1109/CVPR.2017.632.

[241] G. G. Chrysos, J. Kossaifi, and S. Zafeiriou, "RoCGAN: Robust Conditional GAN," *Int. J. Comput. Vis.*, vol. 128, no. 10, pp. 2665–2683, 2020, doi: 10.1007/s11263-020-01348-5.

[242] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," Nov. 2015, [Online]. Available: http://arxiv.org/abs/1511.06434

[243] A. Diaz-Pinto, A. Colomer, V. Naranjo, S. Morales, Y. Xu, and A. F. Frangi, "Retinal Image Synthesis and Semi-Supervised Learning for Glaucoma Assessment," *IEEE Trans. Med. Imaging*, vol. 38, no. 9, pp. 2211–2218, Sep. 2019, doi: 10.1109/TMI.2019.2903434.

[244] E. P. Onakpojeruo, M. T. Mustapha, D. U. Ozsahin, and I. Ozsahin, "A Comparative Analysis of the Novel Conditional Deep Convolutional Neural Network Model, Using Conditional Deep Convolutional Generative Adversarial Network-Generated Synthetic and Augmented Brain Tumor Datasets for Image Classification," *Brain Sci.*, vol. 14, no. 6, 2024, doi: 10.3390/brainsci14060559.

[245] H. Zhang, I. Goodfellow, G. Brain, D. Metaxas, and A. Odena, “Self-Attention Generative Adversarial Networks,” 2018, doi: 10.48550/arXiv.1805.08318.

[246] T. Karras *et al.*, “Alias-Free Generative Adversarial Networks,” 2021. [Online]. Available: https://arxiv.org/abs/2106.12423

[247] T. Karras, S. Laine, and T. Aila, “A Style-Based Generator Architecture for Generative Adversarial Networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 12, pp. 4217–4228, 2021, doi: 10.1109/TPAMI.2020.2970919.

[248] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and Improving the Image Quality of StyleGAN,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 8107–8116. doi: 10.1109/CVPR42600.2020.00813.

[249] E. Bisong, *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*. 2019. doi: 10.1007/978-1-4842-4470-8.

[250] A. and E. K. and S. J. T. and B. M. and H. F. Feurer Matthias and Klein, “Auto-sklearn: Efficient and Robust Automated Machine Learning,” in *Automated Machine Learning: Methods, Systems, Challenges*, L. and V. J. Hutter Frank and Kotthoff, Ed., Cham: Springer International Publishing, 2019, pp. 113–134. doi: 10.1007/978-3-030-05318-5_6.

[251] C. White *et al.*, “Neural Architecture Search: Insights from 1000 Papers,” Jan. 2023, [Online]. Available: http://arxiv.org/abs/2301.08727

[252] T. Elsken, J. H. Metzen, and F. Hutter, “Neural architecture search: a survey,” *J. Mach. Learn. Res.*, vol. 20, no. 1, pp. 1997–2017, Jan. 2019.

[253] M. Baratchi *et al.*, “Automated machine learning: past, present and future,” *Artif. Intell. Rev.*, vol. 57, no. 5, p. 122, 2024, doi: 10.1007/s10462-024-10726-1.

[254] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning: Methods, Systems, Challenges*, 1st ed. Springer Publishing Company, Incorporated, 2019.

[255] R. J. and A. P. C. and L. N. A. and K. L. C. and M. J. H. Olson Randal S. and Urbanowicz, “Automating Biomedical Data Science Through Tree-Based Pipeline Optimization,” in *Applications of Evolutionary Computation*, P. Squillero Giovanni and Burelli, Ed., Cham: Springer International Publishing, 2016, pp. 123–137.

[256] M. Wiering and M. Otterlo, *Reinforcement Learning: State-of-the-Art*, 1st ed., vol. 12. in Adaptation, Learning, and Optimization, vol. 12. Berlin, Heidelberg: Springer Berlin, Heidelberg, 2012. doi: 10.1007/978-3-642-27645-3.

[257] B. Memarian and T. Doleck, “A scoping review of reinforcement learning in education,” *Computers and Education Open*, vol. 6, p. 100175, 2024, doi: https://doi.org/10.1016/j.caeo.2024.100175.

[258] P. Burggräf, F. Steinberg, B. Heinbach, and M. Bamberg, “Reinforcement Learning for Process Time Optimization in an Assembly Process Utilizing an Industry 4.0 Demonstration Cell,” *Procedia CIRP*, vol. 107, pp. 1095–1100, 2022, doi: https://doi.org/10.1016/j.procir.2022.05.114.

[259] Á. Martínez-Tenor, A. Cruz-Martín, and J.-A. Fernández-Madrigal, “Teaching machine learning in robotics interactively: the case of reinforcement learning with Lego® Mindstorms,” *Interactive Learning Environments*, vol. 27, no. 3, pp. 293–306, 2019, doi: 10.1080/10494820.2018.1525411.

[260] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, “Curriculum learning for reinforcement learning domains: a framework and survey,” *J. Mach. Learn. Res.*, vol. 21, no. 1, Jan. 2020.

[261] B. Hambly, R. Xu, and H. Yang, “Recent advances in reinforcement learning in finance,” *Math. Financ.*, vol. 33, no. 3, pp. 437–503, 2023, doi: https://doi.org/10.1111/mafi.12382.

[262] K. L. Voogd, J. P. Allamaa, J. Alonso-Mora, and T. D. Son, “Reinforcement Learning from Simulation to Real World Autonomous Driving using Digital Twin,” *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 1510–1515, 2023, doi: https://doi.org/10.1016/j.ifacol.2023.10.1846.

[263] P. Jayaraman, J. Desman, M. Sabounchi, G. N. Nadkarni, and A. Sakhuja, “A Primer on Reinforcement Learning in Medicine for Clinicians,” *NPJ Digit. Med.*, vol. 7, no. 1, p. 337, 2024, doi: 10.1038/s41746-024-01316-0.

[264] J. Balderas, D. Chen, Y. Huang, L. Wang, and R.-C. Li, “A comparative study of deep reinforcement learning for crop production management,” *Smart Agricultural Technology*, vol. 10, p. 100853, 2025, doi: https://doi.org/10.1016/j.atech.2025.100853.

[265] N. N. C. Pushpamali, D. Agdas, and T. M. Rose, “Strategic Decision Making in Construction Supply Chains: A Comparison of Reverse Logistics Strategies,” *Front. Built Environ.*, vol. Volume 6, 2020, doi: 10.3389/fbuil.2020.593372.

[266] V. Mnih *et al.*, “Playing Atari with Deep Reinforcement Learning,” Dec. 2013, [Online]. Available: http://arxiv.org/abs/1312.5602

[267] J. Liang *et al.*, “A Review of Multi-Agent Reinforcement Learning Algorithms,” *Electronics (Basel).*, vol. 14, no. 4, 2025, doi: 10.3390/electronics14040820.

[268] X. Wu, R. Li, Z. He, T. Yu, and C. Cheng, “A value-based deep reinforcement learning model with human expertise in optimal treatment of sepsis,” *NPJ Digit. Med.*, vol. 6, no. 1, p. 15, 2023, doi: 10.1038/s41746-023-00755-5.

[269] M. Sewak, “Deep Q Network (DQN), Double DQN, and Dueling DQN,” in *Deep Reinforcement Learning*, Springer Singapore, 2019, pp. 95–108. doi: 10.1007/978-981-13-8285-7_8.

[270] C. Cassady, “Reinforcement Learning with Deep Q-Networks (DQN),” *SSRN Electronic Journal*, vol. 16, pp. 575–582, Jun. 2016.

[271] E. Toledo, J. Buys, and J. Shock, “Policy-based Reinforcement Learning for Generalisation in Interactive Text-based Environments,” in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, A. Vlachos and I. Augenstein, Eds., Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 1230–1242. doi: 10.18653/v1/2023.eacl-main.88.

[272] R. J. Williams, “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning,” *Mach. Learn.*, vol. 8, no. 3–4, pp. 229–256, May 1992, doi: 10.1007/BF00992696.

[273] U. Saha, A. Jawad, S. Shahria, and A. B. M. H.-U. Rashid, “Proximal policy optimization-based reinforcement learning approach for DC-DC boost converter control: A comparative evaluation against traditional control techniques,” *Heliyon*, vol. 10, no. 18, p. e37823, 2024, doi: https://doi.org/10.1016/j.heliyon.2024.e37823.

[274] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” Jul. 2017, [Online]. Available: http://arxiv.org/abs/1707.06347

[275] O. Dogru, K. Velswamy, and B. Huang, “Actor–Critic Reinforcement Learning and Application in Developing Computer-Vision-Based Interface Tracking,” *Engineering*, vol. 7, no. 9, pp. 1248–1261, 2021, doi: https://doi.org/10.1016/j.eng.2021.04.027.

[276] T. Li, W. Xi, M. Fang, J. Xu, and M. Q.-H. Meng, “Learning to Solve a Rubik’s Cube with a Dexterous Hand,” in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2019, pp. 1387–1393. doi: 10.1109/ROBIO49542.2019.8961560.

[277] P. Pinto Santos, F. AntónioAnt, A. Chaves Saraiva de Melo JoséJos, J. Alberto Rodrigues Pereira Sardinha, P. Romano Supervisor, and A. Chaves Saraiva de Melo, "Traffic Light Control using Deep Reinforcement Learning," 2021. Accessed: Jun. 22, 2025. [Online]. Available: https://scholar.tecnico.ulisboa.pt/records/pKvsT-t336HfnpAJhdB7RfENMgsejFl_KUxk

[278] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *J. Big Data*, vol. 6, no. 1, p. 27, 2019, doi: 10.1186/s40537-019-0192-5.

[279] N. Romero, M. Gutoski, L. Hattori, and H. S. Lopes, "The Effect of Data Augmentation on the Performance of Convolutional Neural Networks," Associacao Brasileira de Inteligencia Computacional - ABRICOM, Aug. 2019, pp. 1–12. doi: 10.21528/cbic2017-51.

[280] T. Kumar, A. Mileo, R. Brennan, and M. Bendechache, "Image Data Augmentation Approaches: A Comprehensive Survey and Future directions," Jan. 2023, doi: 10.1109/ACCESS.2024.3470122.

[281] I. Kostrikov, D. Yarats, and R. Fergus, "Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels," 2021. Accessed: Jun. 22, 2025. [Online]. Available: https://arxiv.org/abs/2004.13649

[282] J. Hu, Y. Jiang, and P. Weng, "Revisiting Data Augmentation in Deep Reinforcement Learning," Feb. 2024, [Online]. Available: http://arxiv.org/abs/2402.12181

[283] R. Raileanu, M. Goldstein, D. Yarats, I. Kostrikov, and R. Fergus, "Automatic Data Augmentation for Generalization in Reinforcement Learning," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. W. Vaughan, Eds., Curran Associates, Inc., 2021, pp. 5402–5415. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/2b38c2df6a49b97f706ec9148ce48d86-Paper.pdf

[284] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman, "Data-Efficient Reinforcement Learning with Momentum Predictive Representations," Jun. 2020. doi: 10.48550/arXiv.2007.05929.

[285] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, in NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[286] G. Ma, Z. Wang, Z. Yuan, X. Wang, B. Yuan, and D. Tao, "A Comprehensive Survey of Data Augmentation in Visual Reinforcement Learning," *Int. J. Comput. Vis.*, vol. 133, no. 10, pp. 7368–7405, 2025, doi: 10.1007/s11263-025-02472-w.

[287] S. M. Grigorescu, D. Ristic-Durrant, S. K. Vuppala, and A. Graeser, "Closed-loop control in image processing for improvement of object recognition," in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2008. doi: 10.3182/20080706-5-KR-1001.2132.

[288] L. Lu, J. Hou, S. Yuan, X. Yao, Y. Li, and J. Zhu, "Deep learning-assisted real-time defect detection and closed-loop adjustment for additive manufacturing of continuous fiber-reinforced polymer composites," *Robot. Comput. Integr. Manuf.*, vol. 79, Feb. 2022, doi: 10.1016/j.rcim.2022.102431.

[289] S. E. Whang, Y. Roh, H. Song, and J. G. Lee, "Data collection and quality challenges in deep learning: a data-centric AI perspective," *VLDB Journal*, vol. 32, no. 4, pp. 791–813, Jul. 2023, doi: 10.1007/s00778-022-00775-9.

[290] C. Sager, C. Janiesch, and P. Zschech, "A survey of image labelling for computer vision applications," *Journal of Business Analytics*, vol. 4, no. 2, pp. 91–110, 2021, doi: 10.1080/2573234X.2021.1908861.

[291] I. F. Ilyas and T. Rekatsinas, “Machine Learning and Data Cleaning: Which Serves the Other?,” *J. Data and Information Quality*, vol. 14, no. 3, Jul. 2022, doi: 10.1145/3506712.

[292] I. F. Ilyas and X. Chu, *Data Cleaning*. New York, NY, USA: Association for Computing Machinery, 2019.

[293] M. K. Uçar, M. Nour, H. Sindi, and K. Polat, “The Effect of Training and Testing Process on Machine Learning in Biomedical Datasets,” *Math. Probl. Eng.*, vol. 2020, 2020, doi: 10.1155/2020/2836236.

[294] I. Muraina, “Ideal dataset splitting ratios in machine learning algorithms: General concerns for data scientists and data analysts,” Jun. 2022.

[295] Y. Xu and R. Goodacre, “On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning,” *J. Anal. Test.*, vol. 2, no. 3, pp. 249–262, Jul. 2018, doi: 10.1007/s41664-018-0068-2.

[296] J. M. Johnson and T. M. Khoshgoftaar, “Survey on deep learning with class imbalance,” *J. Big Data*, vol. 6, no. 1, Dec. 2019, doi: 10.1186/s40537-019-0192-5.

[297] A. Luque, A. Carrasco, A. Martín, and A. de las Heras, “The impact of class imbalance in classification performance metrics based on the binary confusion matrix,” *Pattern Recognit.*, vol. 91, pp. 216–231, Jul. 2019, doi: 10.1016/j.patcog.2019.02.023.

[298] T. Adão *et al.*, “Digital ampelographer: A CNN based preliminary approach,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 2019, pp. 258–271. doi: 10.1007/978-3-030-30241-2_23.

[299] Tzutalin, “LabelImg: Image annotation tool.” Accessed: Nov. 20, 2023. [Online]. Available: https://github.com/tzutalin/labelImg

[300] B. Chen, X. Liu, Y. Zheng, G. Zhao, and Y.-Q. Shi, “A Robust GAN-Generated Face Detection Method Based on Dual-Color Spaces and an Improved Xception,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 6, pp. 3527–3538, 2022, doi: 10.1109/TCSVT.2021.3116679.

[301] K. Shaheed *et al.*, “DS-CNN: A pre-trained Xception model based on depth-wise separable convolutional neural network for finger vein recognition,” *Expert Syst. Appl.*, vol. 191, p. 116288, 2022, doi: https://doi.org/10.1016/j.eswa.2021.116288.

[302] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “ MobileNetV2: Inverted Residuals and Linear Bottlenecks ,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* , Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474.

[303] F. Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1800–1807. doi: 10.1109/CVPR.2017.195.

[304] L. Ali, F. Alnajjar, H. Al Jassmi, M. Gochoo, W. Khan, and M. A. Serhani, “Performance evaluation of deep CNN-based crack detection and localization techniques for concrete structures,” *Sensors*, vol. 21, no. 5, pp. 1–22, Mar. 2021, doi: 10.3390/s21051688.

[305] S. E. Boudjellal, N.-E. Boukezzoula, and A. Boudjelal, “Deep learning model based on inceptionResnet-v2 for Finger vein recognition,” in *2022 International Conference of Advanced Technology in Electronic and Electrical Engineering (ICATEEE)*, 2022, pp. 1–5. doi: 10.1109/ICATEEE57445.2022.10093753.

[306] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," Jan. 2018, [Online]. Available: http://arxiv.org/abs/1801.04381

[307] M. Xu, S. Yoon, A. Fuentes, and D. S. Park, "A Comprehensive Survey of Image Augmentation Techniques for Deep Learning," *Pattern Recognit.*, vol. 137, p. 109347, 2023, doi: https://doi.org/10.1016/j.patcog.2023.109347.

[308] M. Bazzaz and S. Cooper, "Controllable Game Level Generation: Assessing the Effect of Negative Examples in GAN Models," 2024. [Online]. Available: https://arxiv.org/abs/2410.23108

[309] T. Kaneko, K. Hiramatsu, and K. Kashino, "Generative Attribute Controller with Conditional Filtered Generative Adversarial Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7006–7015. doi: 10.1109/CVPR.2017.741.

[310] T. Salimans *et al.*, "Improved Techniques for Training GANs," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf

[311] D. Wu, S. Zhuo, J. Ji, W. Chen, S. Qu, and L. Ma, "Application of conditional generative adversarial network to multi-step car-following modelling."

[312] A. Odena, C. Olah, and J. Shlens, "Conditional Image Synthesis With Auxiliary Classifier GANs," 2017. [Online]. Available: https://arxiv.org/abs/1610.09585

[313] I. Goodfellow *et al.*, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, Oct. 2020, doi: 10.1145/3422622.

[314] A. Mehmood, J. Ko, H. Kim, and J. Kim, "Optimizing Image Enhancement: Feature Engineering for Improved Classification in AI-Assisted Artificial Retinas," *Sensors*, vol. 24, no. 9, May 2024, doi: 10.3390/s24092678.

[315] X. Zhaosheng, A. Suzana, L. Zhongming, X. Xiuhong, and X. Zhongqi, "Image feature extraction algorithm based on visual information," *Journal of Intelligent Systems*, vol. 32, no. 1, Jan. 2023, doi: 10.1515/jisys-2023-0111.

[316] W. Jia, M. Sun, J. Lian, and S. Hou, "Feature dimensionality reduction: a review," *Complex and Intelligent Systems*, vol. 8, no. 3, pp. 2663–2693, Jun. 2022, doi: 10.1007/s40747-021-00637-x.

[317] S. Velliangiri, S. Alagumuthukrishnan, and S. I. Thankumar Joseph, "A Review of Dimensionality Reduction Techniques for Efficient Computation," in *Procedia Computer Science*, Elsevier B.V., 2019, pp. 104–111. doi: 10.1016/j.procs.2020.01.079.

[318] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, pp. 886–893 vol. 1. doi: 10.1109/CVPR.2005.177.

[319] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999, pp. 1150–1157 vol.2. doi: 10.1109/ICCV.1999.790410.

[320] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, Jun. 2008, doi: 10.1016/j.cviu.2007.09.014.

[321] N. Manakitsa, G. S. Maraslidis, L. Moysis, and G. F. Fragulis, "A Review of Machine Learning and Deep Learning for Object Detection, Semantic Segmentation, and Human Action Recognition in

Machine and Robotic Vision," Feb. 01, 2024, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/technologies12020015.

[322] O. Fink, Q. Wang, M. Svensén, P. Dersin, W. J. Lee, and M. Ducoffe, "Potential, challenges and future directions for deep learning in prognostics and health management applications," *Eng. Appl. Artif. Intell.*, vol. 92, Jun. 2020, doi: 10.1016/j.engappai.2020.103678.

[323] P. W. Tien, S. Wei, J. Darkwa, C. Wood, and J. K. Calautit, "Machine Learning and Deep Learning Methods for Enhancing Building Energy Efficiency and Indoor Environmental Quality – A Review," Nov. 01, 2022, *Elsevier B.V.* doi: 10.1016/j.egyai.2022.100198.

[324] J. Valente, J. António, C. Mora, and S. Jardim, "Developments in Image Processing Using Deep Learning and Reinforcement Learning," Oct. 01, 2023, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/jimaging9100207.

[325] H. ZainEldin *et al.*, "Silent no more: a comprehensive review of artificial intelligence, deep learning, and machine learning in facilitating deaf and mute communication," *Artif. Intell. Rev.*, vol. 57, no. 7, Jul. 2024, doi: 10.1007/s10462-024-10816-0.

[326] M. Li, Y. Jiang, Y. Zhang, and H. Zhu, "Medical image analysis using deep learning algorithms," *Front. Public Health*, vol. 11, 2023, doi: 10.3389/fpubh.2023.1273253.

[327] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *J. Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00444-8.

[328] M. Iman, H. R. Arabnia, and K. Rasheed, "A Review of Deep Transfer Learning and Recent Advancements," Apr. 01, 2023, *MDPI*. doi: 10.3390/technologies11020040.

[329] Q. Guan *et al.*, "Deep convolutional neural network VGG-16 model for differential diagnosing of papillary thyroid carcinomas in cytological images: A pilot study," *J. Cancer*, vol. 10, no. 20, pp. 4876–4882, 2019, doi: 10.7150/jca.28769.

[330] K. Pearson, " LIII. On lines and planes of closest fit to systems of points in space ," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, Nov. 1901, doi: 10.1080/14786440109462720.

[331] Z. Zhang, Y. Luo, X. Deng, and W. Luo, "Digital image technology based on PCA and SVM for detection and recognition of foreign bodies in lyophilized powder," *Technology and Health Care*, vol. 28, no. S1, pp. S197–S205, 2020, doi: 10.3233/THC-209020.

[332] I. T. Jollife and J. Cadima, "Principal component analysis: A review and recent developments," Apr. 13, 2016, *Royal Society of London*. doi: 10.1098/rsta.2015.0202.

[333] A. Dosovitskiy *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," Oct. 2020, [Online]. Available: http://arxiv.org/abs/2010.11929

[334] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in Vision: A Survey," Jan. 2021, doi: 10.1145/3505244.

[335] K. Han *et al.*, "A Survey on Visual Transformer," Dec. 2020, doi: 10.1109/TPAMI.2022.3152247.

[336] S. Takahashi *et al.*, "Comparison of Vision Transformers and Convolutional Neural Networks in Medical Image Analysis: A Systematic Review," Dec. 01, 2024, *Springer*. doi: 10.1007/s10916-024-02105-8.

[337] Z. Li, J. Hu, K. Wu, J. Miao, Z. Zhao, and J. Wu, "Local feature acquisition and global context understanding network for very high-resolution land cover classification," *Sci. Rep.*, vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-63363-7.

[338] J. Maurício, I. Domingues, and J. Bernardino, “Comparing Vision Transformers and Convolutional Neural Networks for Image Classification: A Literature Review,” May 01, 2023, *MDPI*. doi: 10.3390/app13095521.

[339] C. Xu, P. Coen-Pirani, and X. Jiang, “Empirical Study of Overfitting in Deep Learning for Predicting Breast Cancer Metastasis,” *Cancers (Basel).*, vol. 15, no. 7, Apr. 2023, doi: 10.3390/cancers15071969.

[340] J. Mathur, B. Baruah, and P. Tiwari, “Prediction of bio-oil yield during pyrolysis of lignocellulosic biomass using machine learning algorithms,” *Can. J. Chem. Eng.*, vol. 101, no. 5, pp. 2457–2471, 2023, doi: https://doi.org/10.1002/cjce.24674.

[341] O. A. Montesinos López, A. Montesinos López, and J. Crossa, *Multivariate Statistical Machine Learning Methods for Genomic Prediction*. Springer International Publishing, 2022. doi: 10.1007/978-3-030-89010-0.

[342] R. Archana and P. S. E. Jeevaraj, “Deep learning models for digital image processing: a review,” *Artif. Intell. Rev.*, vol. 57, no. 1, Jan. 2024, doi: 10.1007/s10462-023-10631-z.

[343] D. Arous, S. Schrunner, I. Hanson, N. Frederike Jeppesen Edin, and E. Malinen, “Principal component-based image segmentation: a new approach to outline in vitro cell colonies,” *Comput. Methods Biomech. Biomed. Eng. Imaging Vis.*, vol. 11, no. 1, pp. 18–30, 2023, doi: 10.1080/21681163.2022.2035822.

[344] C. Shi, B. Wei, S. Wei, W. Wang, H. Liu, and J. Liu, “A quantitative discriminant method of elbow point for the optimal number of clusters in clustering algorithm,” *EURASIP J. Wirel. Commun. Netw.*, vol. 2021, no. 1, p. 31, 2021, doi: 10.1186/s13638-021-01910-w.

[345] M. A. Syakur, B. K. Khotimah, E. M. S. Rochman, and B. D. Satoto, “Integration K-Means Clustering Method and Elbow Method for Identification of the Best Customer Profile Cluster,” in *IOP Conference Series: Materials Science and Engineering*, Institute of Physics Publishing, Apr. 2018. doi: 10.1088/1757-899X/336/1/012017.

[346] D. Marutho, S. Hendra Handaka, E. Wijaya, and Muljono, “The Determination of Cluster Number at k-Mean Using Elbow Method and Purity Evaluation on Headline News,” in *2018 International Seminar on Application for Technology of Information and Communication*, 2018, pp. 533–538. doi: 10.1109/ISEMANTIC.2018.8549751.

[347] L. Alzubaidi *et al.*, “A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications,” *J. Big Data*, vol. 10, no. 1, Dec. 2023, doi: 10.1186/s40537-023-00727-2.

[348] Z. Zhao, T. Cui, S. Ding, J. Li, and A. G. Bellotti, “Resampling Techniques Study on Class Imbalance Problem in Credit Risk Prediction,” *Mathematics*, vol. 12, no. 5, Mar. 2024, doi: 10.3390/math12050701.

[349] M. Salmi, D. Atif, D. Oliva, A. Abraham, and S. Ventura, “Handling imbalanced medical datasets: review of a decade of research,” *Artif. Intell. Rev.*, vol. 57, no. 10, Oct. 2024, doi: 10.1007/s10462-024-10884-2.

[350] J. P. Gygi, S. H. Kleinstein, and L. Guan, “Predictive overfitting in immunological applications: Pitfalls and solutions,” 2023, *Taylor and Francis Ltd.* doi: 10.1080/21645515.2023.2251830.

[351] X. Ying, “An Overview of Overfitting and its Solutions,” in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Mar. 2019. doi: 10.1088/1742-6596/1168/2/022022.

[352] A. S. Tarawneh, A. B. Hassanat, G. A. Altarawneh, and A. Almuhaimeed, “Stop Oversampling for Class Imbalance Learning: A Review,” *IEEE Access*, vol. 10, pp. 47643–47660, 2022, doi: 10.1109/ACCESS.2022.3169512.

[353] S. I. Kumamoto and T. Kamihigashi, “Power laws in stochastic processes for social phenomena: An introductory review,” Mar. 01, 2018, *Frontiers Media SA*. doi: 10.3389/fphy.2018.00020.

[354] A. Atangana and S. Jain, “The role of power decay, exponential decay and Mittag-Leffler function’s waiting time distribution: Application of cancer spread,” *Physica A: Statistical Mechanics and its Applications*, vol. 512, pp. 330–351, Dec. 2018, doi: 10.1016/j.physa.2018.08.033.

[355] A. Clauset, C. R. Shalizi, and M. E. J. Newman, “Power-Law Distributions in Empirical Data,” *SIAM Review*, vol. 51, no. 4, pp. 661–703, Nov. 2009, doi: 10.1137/070710111.

[356] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “GANs trained by a two time-scale update rule converge to a local nash equilibrium,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, in NIPS’17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6629–6640.