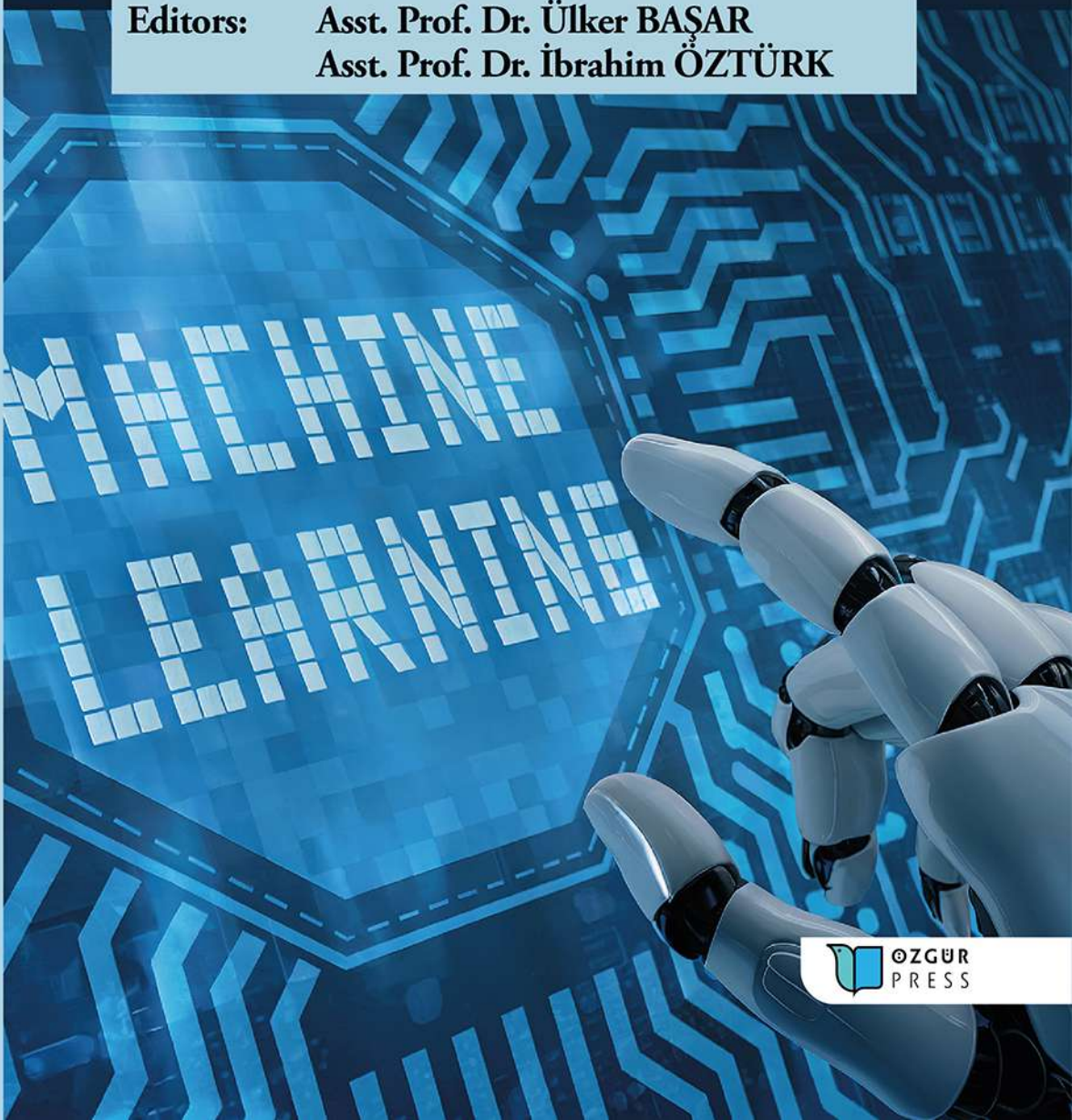
Machine Learning in Computer Science: Concepts, Hybrid Methods, and Spiking Neural Networks

Editors: Asst. Prof. Dr. Ülker BAŞAR
Asst. Prof. Dr. İbrahim ÖZTÜRK



ÖZGÜR PRESS

Machine Learning in Computer Science: Concepts, Hybrid Methods, and Spiking Neural Networks

Editors:

Asst. Prof. Dr. Ülker BAŞAR

Asst. Prof. Dr. İbrahim ÖZTÜRK

ÖZGÜR PRESS

Published by
Özgür Yayın-Dağıtım Co. Ltd.
Certificate Number: 45503

15 Temmuz Mah. 148136. Sk. No: 9 Şehitkamil/Gaziantep
+90.850 260 09 97
+90.532 289 82 15
www.ozguryayınlari.com
info@ozguryayinlari.com

Machine Learning in Computer Science: Concepts, Hybrid Methods, and Spiking Neural Networks

Editors: Asst. Prof. Dr. Ülker BAŞAR • Asst. Prof. Dr. İbrahim ÖZTÜRK

Language: English
Publication Date: 2026
Cover design by Mehmet Çakır
Cover design and image licensed under CC BY-NC 4.0
Print and digital versions typeset by Çizgi Medya Co. Ltd.

ISBN (PDF): 978-625-8998-51-1

DOI: https://doi.org/10.58830/ozgur.pub1236

Suggested citation:
Başar, Ü. (ed), Öztürk, İ. (ed) (2026). *Machine Learning in Computer Science: Concepts, Hybrid Methods, and Spiking Neural Networks.* Özgür Publications. DOI: https://doi.org/10.58830/ozgur.pub1236. License: CC-BY-NC 4.0

The full text of this book has been peer-reviewed to ensure high academic standards. For full review policies, see https://www.ozguryayinlari.com/

Preface

In recent years, machine learning has become a transformative area of research and application, not only in computer science but also across numerous disciplines such as engineering, healthcare, agriculture, music technologies, industrial automation, and decision support systems. The rapid increase in data production, advancements in computing power, and the widespread use of AI-based solutions, from everyday life to scientific research, have placed machine learning at the center of contemporary scientific production. This development has necessitated considering the field not merely as a technical area of expertise, but as a multi-layered research universe encompassing theoretical, methodological, and applied dimensions.

This book addresses machine learning not from a one-dimensional perspective, but within a holistic framework that begins with its fundamentals and encompasses current theoretical debates, hybrid models, biologically inspired learning mechanisms, and field-specific applications. The chapters are structured to represent different layers of the machine learning field, thus becoming a comprehensive reference source appealing to both readers who want to learn fundamental concepts systematically and researchers who wish to delve deeper into specific subfields.

The book's structure is designed to allow the reader to first grasp the fundamental concepts and computational infrastructure of machine learning, then move on to more advanced theoretical and methodological developments, and finally see how this accumulated knowledge is concretized in different application areas. Accordingly, the initial chapters of the book address the basic concepts of machine learning; these are followed by discussions on hybrid machine learning systems, the theoretical limitations of learning under noisy and limited data, and biologically inspired learning mechanisms. The final chapters reveal the practical implications of machine learning through field applications such as music data analysis and agricultural disease classification.

The main aim of this work is to make machine learning visible as an interdisciplinary and multi-scale research field. Each chapter in the book offers its own unique contribution while also shedding light on the evolving nature and future directions of machine learning. In this respect, the work will contribute to the literature as an instructive and guiding resource for graduate students, researchers, academics, and all readers interested in machine learning.

Contents

Chapter 1

Concepts of Machine Learning

Yousef Farhang[1]

Abstract

This chapter introduces the fundamental concepts and principles of machine learning, serving as a theoretical foundation for the subsequent chapters of the book. It provides a comprehensive overview of the main learning paradigms, including supervised, unsupervised, semi-supervised, and reinforcement learning, along with essential terminology and notations commonly used in machine learning research and applications. Key topics such as data representation, preprocessing techniques, feature engineering, and the learning process are discussed to highlight how raw data is transformed into meaningful knowledge through computational models. The chapter also explains core concepts related to model training, generalization, overfitting, and the bias–variance tradeoff, which are critical for understanding model performance and reliability.

In addition, fundamental ideas in optimization and model evaluation are presented, including cost functions, gradient-based learning, and standard performance metrics. Ethical and practical considerations, such as data bias, interpretability, and privacy, are briefly addressed to emphasize responsible use of machine learning technologies. Overall, this chapter establishes a conceptual framework that enables readers to better understand, design, and critically evaluate machine learning systems.

1.1. Introduction

Machine Learning (ML) is originated from computer science (CS) and Artificial Intelligence (AI), which addresses systems that are able to learn from data, rather than merely following the programmed instructions explicitly. Moreover, ML has a close relationship with optimization and statistics, delivering both theory and methods to the field. ML is applied to various

1 Asst. Prof. Dr. Yousef Farhang, Istanbul Esenyurt University, Faculty of Engineering and Architecture, Department of Computer Engineering, youseffarhang@esenyurt.edu.tr, 0000-0001-9348-2624.

https://doi.org/10.58830/ozgur.pub1236.c5001

computing tasks in which designing and programming rule-based, explicit algorithms is impractical. In some situations, ML, pattern recognition, and data mining are conflated.

Arthur Samuel, in 1959, defined ML as a "field of study that gives computers the ability to learn without being clearly programmed". In general, ML and data mining are confused with each other since they use the same methods and they significantly overlap. These two can be described as follows. ML is focused on prediction based on recognized properties that are learned from training data. On the other hand, data mining is concentrated on discovering the (previously) unknown properties in the data. The two areas are overlapped in many aspects: in data mining, numerous machine learning methods are used, but with a diverse goal in mind. However, ML makes use of data mining methods as "unsupervised learning" or as a preprocessing step for the improvement of learner accuracy. The present research is focused on ML.

Regarding to the tasks, ML can be divided into three types, namely supervised learning, semi-supervised learning, and unsupervised learning. Figure 2.1 demonstrates the types of ML.

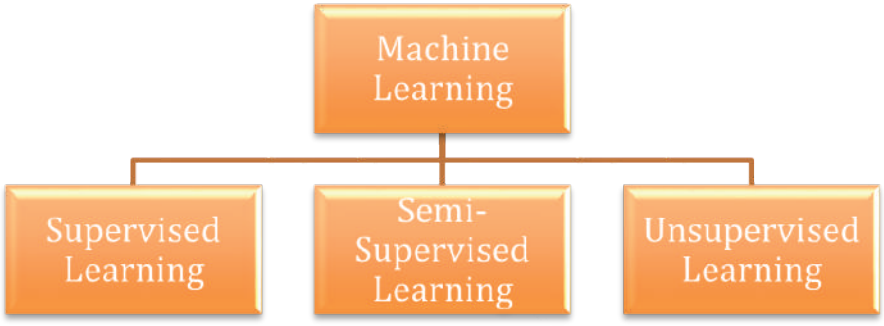


Figure 1.1. Types of Machine Learning

In the supervised learning, example inputs and desired outputs are provided by teacher, aiming at learning a general rule that maps inputs to outputs. The supervised learning is the ML task in which a function is inferred from labeled training data. One of appropriate examples of the supervised learning is classification in which labeled data are used for solving given problems. The supervised learning is the ML task of tracing a function from labeled training data that is consisted of a set of training instances. In the supervised learning, each instance is a pair that is composed of an input, which is typically a vector, and a desired output value, which is also known as supervisory signal. An algorithm that is based on supervised learning makes analysis on training data

and generates an inferred function; the function is applicable to mapping new instances. An optimal scenario helps the algorithm determine correctly the class labels for unseen instances. To this purpose, the learning algorithm should generalize from training data to unseen situations "reasonably".

In the semi-supervised learning, both labeled and unlabeled examples are combined for the generation of a proper function or classifier. The semi-supervised learning is considered as a class of the tasks and techniques of the supervised learning, which typically employs a small amount of labeled data together with a large amount of unlabeled data for training. The semi-supervised learning is situated between supervised learning (with entirely labeled training data) and unsupervised learning (with no labeled training data). Several researchers working on ML have shown that unlabeled data, when utilized together with a small amount of labeled data, are capable of producing significant improvement in the accuracy level of learning.

The performance of the unsupervised learning algorithms is based on unlabeled examples, i.e., input where there is not known desired output. In this case, the goal is discovering the structure of the data, for example using a cluster analysis, not generalizing a mapping from inputs to outputs. In the next section, the unsupervised learning is elaborated.

1.2. Supervised Learning

Supervised learning represents one of the most fundamental and widely used paradigms in machine learning, in which a learning algorithm is trained using labeled data. In this framework, each training instance consists of an input vector together with a corresponding target output, commonly referred to as a label. The primary objective of supervised learning is to learn a mapping function that accurately predicts outputs for previously unseen data.

Formally, supervised learning attempts to approximate an unknown function that maps input variables to output responses based on examples provided during the training phase. The learning process is guided by a supervisory signal that evaluates prediction errors and enables the model to iteratively adjust its internal parameters. Unlike unsupervised learning, where structure must be inferred without guidance, supervised learning benefits from explicit feedback regarding prediction correctness.

Supervised learning problems are generally categorized into two major types: classification and regression. Classification tasks involve assigning discrete labels to input data, such as identifying whether an email is spam or non-spam, diagnosing diseases based on medical records, or recognizing objects in images. Regression tasks, on the other hand, focus on predicting

continuous numerical values, including temperature forecasting, energy demand estimation, and financial trend prediction.

A wide variety of algorithms have been developed for supervised learning applications. Classical methods include linear regression, decision trees, k-nearest neighbors, and support vector machines, while modern approaches rely heavily on artificial neural networks and deep learning architectures. During training, model performance is typically evaluated using loss functions that quantify the difference between predicted outputs and true labels. One of the most important challenges in supervised learning is achieving strong generalization capability, meaning that the learned model performs well not only on training data but also on unseen datasets. Issues such as overfitting and under fitting arise when model complexity is not properly balanced with available data. Consequently, validation techniques, regularization methods, and cross-validation strategies are commonly employed to ensure reliable predictive performance. Due to its ability to learn predictive relationships directly from historical data, supervised learning has become a cornerstone of modern artificial intelligence systems and is extensively applied in areas such as medical diagnosis, speech recognition, autonomous systems, recommender systems, and intelligent decision support.

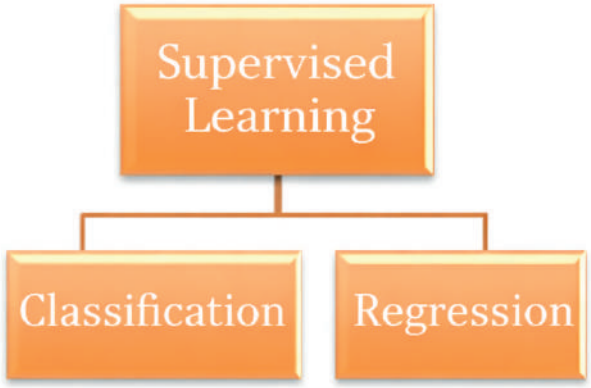


Figure 1.2. Types of the Supervised Learning

Figure 1.2. illustrates the fundamental structure of the supervised learning paradigm, where learning is performed using labeled training data. As shown in the figure, supervised learning problems are primarily categorized into two major types: classification and regression. Classification focuses on assigning discrete class labels to input data based on learned decision boundaries, while regression aims to predict continuous numerical values through functional

approximation. These two problem categories represent the core predictive tasks addressed by supervised learning algorithms. During the learning process, models utilize labeled examples to minimize prediction errors and improve performance through iterative optimization. The hierarchical structure presented in the figure highlights how supervised learning forms the foundation of predictive modeling systems widely applied in domains such as medical diagnosis, pattern recognition, financial forecasting, and intelligent decision-making systems.

1.3. Semi-Supervised Learning

Semi-supervised learning constitutes an intermediate learning paradigm that combines characteristics of both supervised and unsupervised learning. In many real-world applications, obtaining labeled data is expensive, time-consuming, or requires expert knowledge, whereas large quantities of unlabeled data are readily available. Semi-supervised learning addresses this limitation by utilizing a small set of labeled samples together with a substantially larger collection of unlabeled data during training.

The central assumption underlying semi-supervised learning is that unlabeled data contains valuable structural information about the underlying distribution of the dataset. By exploiting this structure, learning algorithms can improve classification or prediction accuracy beyond what is achievable using labeled data alone. This paradigm effectively reduces annotation costs while maintaining high model performance.

Several theoretical assumptions guide semi-supervised learning methods. The smoothness assumption states that data points located close to one another in feature space are likely to share similar labels. The cluster assumption suggests that decision boundaries should lie in low-density regions separating clusters of data. Additionally, the manifold assumption proposes that high-dimensional data often resides on lower-dimensional manifolds that can be exploited for learning.

A variety of techniques have been developed within the semi-supervised learning framework. Self-training methods iteratively label confident unlabeled samples and incorporate them into the training set. Co-training approaches employ multiple classifiers trained on different feature subsets to improve learning reliability. Graph-based methods model relationships among samples using similarity graphs, allowing label information to propagate through connected data points. More recently, pseudo-labeling and consistency-regularization techniques have gained popularity in deep learning applications.

Semi-supervised learning has demonstrated remarkable success in domains where labeled data is scarce, including medical imaging, speech processing, natural language understanding, remote sensing, and anomaly detection. By integrating supervised guidance with unsupervised structure discovery, semi-supervised learning provides a practical balance between learning accuracy and data annotation cost. Consequently, semi-supervised learning serves as a critical bridge between fully supervised and fully unsupervised paradigms, enabling scalable machine learning solutions in modern data-intensive environments.

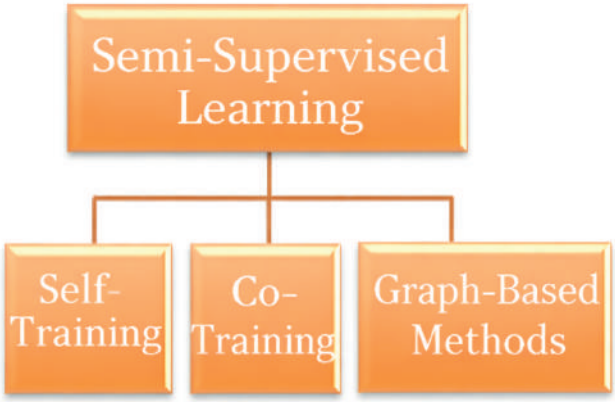


Figure 1.3. Types of the Semi-Supervised Learning

Figure 1.3. presents the conceptual framework of semi-supervised learning, which integrates both labeled and unlabeled data within the learning process. As depicted in the figure, semi-supervised learning techniques commonly include self-training, co-training, and graph-based methods. Self-training approaches iteratively expand labeled datasets by assigning pseudo-labels to confidently predicted unlabeled samples. Co-training methods employ multiple learners trained on complementary feature subsets to enhance learning reliability. Graph-based approaches model relationships among data samples using similarity graphs, enabling label information to propagate throughout the dataset. The structure illustrated in the figure demonstrates how semi-supervised learning bridges the gap between supervised and unsupervised paradigms by exploiting the intrinsic structure of unlabeled data while maintaining guidance from labeled examples. This capability significantly reduces annotation cost while improving model generalization in real-world applications where labeled data availability is limited.

1.4. Unsupervised Learning

In ML, the unsupervised learning problem is that of attempting to find the structure hidden in unlabeled data. As the examples that are provided for the learner are unlabeled, there is not error or reward signal for the evaluation of a potential solution. This makes difference between the unsupervised learning and the supervised learning and reinforcement learning. The unsupervised learning is in a close relationship with the density estimation problem in statistics. On the other hand, the unsupervised learning involves several other techniques used for summarizing and explaining the most important characteristics of data. Several methods that are used in unsupervised learning are designed based on data mining methods that are applied to preprocessing data.

The approaches based on the unsupervised learning can be divided into three types, namely dimensionality reduction, self-organizing map, and clustering algorithm. Figure 1.4. presents the types of the unsupervised learning.

Figure 1.4 Types of the Unsupervised Learning

In ML, dimensionality reduction and statistics is a process in which the number of random variables under consideration is reduced. This is separated into feature selection and feature extraction. In case of high-dimensional datasets, the dimensionality reduction is generally carried out before using a *K*-Nearest Neighbors (*K*-NN) algorithm to avoid the impacts of the dimensionality curse.

A self-organizing map (SOM) is a type of Artificial Neural Network (ANN) that is trained using the unsupervised learning in order to generate a low-dimensional, discretized representation of input space of the training samples, which is named a map. SOMs differ from other ANNs; SOMs employ a neighborhood function in order to preserve the topological properties of the input space.

Clustering refers to the task of grouping a set of objects in such a way that objects in the same cluster are more similar to each other than to those in other clusters. In the next section, clustering analysis will be fully described.

1.4.1. Dimensionality Reduction

Dimensionality reduction is an essential technique in unsupervised learning that aims to reduce the number of variables under consideration while preserving the intrinsic structure and meaningful information contained in the original dataset. In many real-world applications, datasets often contain a large number of features, some of which may be redundant, irrelevant, or noisy. The presence of such high-dimensional data increases computational complexity and may negatively affect learning performance, a phenomenon commonly referred to as the *curse of dimensionality*.

The primary objective of dimensionality reduction is to transform data from a high-dimensional space into a lower-dimensional representation that retains the most significant characteristics of the original data distribution. By reducing dimensionality, machine learning models become computationally efficient, less sensitive to noise, and more capable of generalization.

Dimensionality reduction techniques are generally categorized into two main approaches: feature selection and feature extraction. Feature selection focuses on identifying a subset of relevant original variables without modifying them, whereas feature extraction generates new variables through mathematical transformations or combinations of existing features.

Several well-established techniques have been developed for dimensionality reduction. Linear approaches such as Principal Component Analysis (PCA) aim to maximize variance preservation while minimizing information loss. Independent Component Analysis (ICA) attempts to identify statistically independent components within data. In contrast, nonlinear dimensionality reduction methods, including manifold learning techniques and auto encoders, capture complex relationships that cannot be represented through linear transformations.

Dimensionality reduction plays a critical role in visualization and exploratory data analysis by enabling high-dimensional datasets to be represented in two- or three-dimensional spaces. Furthermore, it is frequently employed as a preprocessing step prior to clustering algorithms such as K-means or hierarchical clustering in order to improve distance measurement reliability and clustering stability.

Consequently, dimensionality reduction serves as a fundamental bridge between raw high-dimensional data and efficient machine learning models, facilitating improved learning performance and interpretability in large-scale data environments.

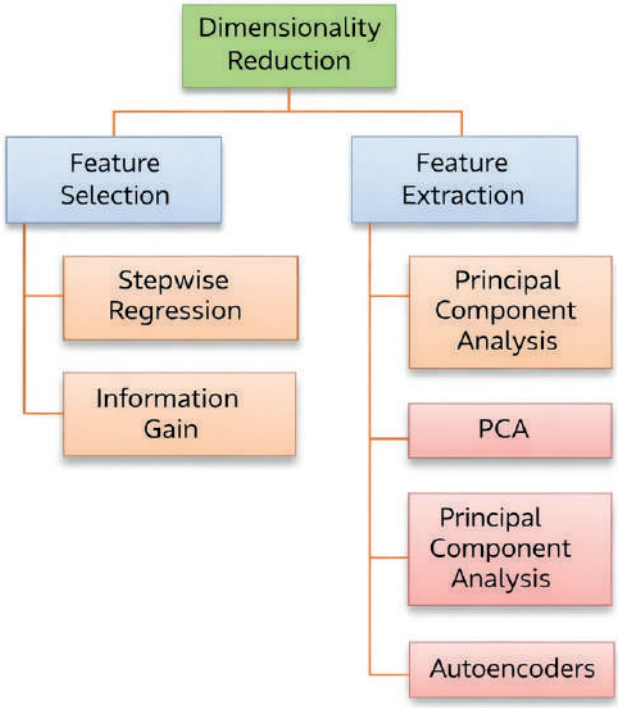


Figure 1.5. Types of the Dimensionality Reduction Techniques

Figure 1.5. illustrates the general taxonomy of dimensionality reduction techniques used in unsupervised learning. As shown in the figure, dimensionality reduction methods are broadly categorized into two principal groups: feature selection and feature extraction. Feature selection approaches aim to identify the most informative subset of original variables while eliminating redundant or irrelevant attributes that may negatively affect learning performance. Techniques such as stepwise regression and information gain analysis evaluate feature importance based on statistical or information-theoretic criteria. In contrast, feature extraction methods transform the original data into a new lower-dimensional representation through mathematical projection or nonlinear transformation. Classical approaches such as Principal Component Analysis (PCA) reduce dimensionality by maximizing variance preservation, whereas

modern techniques such as auto encoders learn compact representations using neural network architectures. Overall, dimensionality reduction improves computational efficiency, enhances visualization capability, and increases the stability of subsequent learning algorithms such as clustering and classification models.

1.4.2. Self-Organizing Map

The Self-Organizing Map (SOM) is a neural network–based unsupervised learning technique designed to produce a low-dimensional representation of high-dimensional input data while preserving the topological relationships among data samples. Originally introduced by Teuvo Kohonen, SOM provides an effective mechanism for visualization, clustering, and exploratory pattern discovery.

Unlike traditional artificial neural networks that rely on supervised learning, SOM operates through competitive learning. In this framework, neurons compete to represent input data, and only the most suitable neuron—referred to as the *winning neuron*—is activated during each training iteration. Neighboring neurons surrounding the winner are simultaneously updated according to a neighborhood function, enabling the network to maintain spatial relationships among similar data points.

The architecture of a self-organizing map typically consists of a two-dimensional grid of neurons, where each neuron is associated with a weight vector having the same dimensionality as the input data. During training, input vectors are repeatedly presented to the network, and neuron weights gradually adapt to approximate the distribution of the input space. As learning progresses, similar observations become mapped to nearby neurons, resulting in an organized representation of data structure.

One of the most important characteristics of SOM is its ability to preserve topology. This means that data points that are close in the original feature space remain close within the generated map. This property distinguishes SOM from many other clustering approaches and makes it particularly useful for analyzing complex multidimensional datasets.

Self-organizing maps are widely applied in numerous domains, including pattern recognition, image processing, bioinformatics, financial analysis, and anomaly detection. They are especially valuable when prior knowledge about class labels is unavailable and when visualization of hidden data structures is required.

In modern machine learning workflows, SOM can also serve as a preprocessing or clustering mechanism, supporting dimensionality reduction and facilitating subsequent analytical tasks. Due to its interpretability and visualization capability, SOM remains an important unsupervised learning tool despite the emergence of more advanced deep learning techniques.

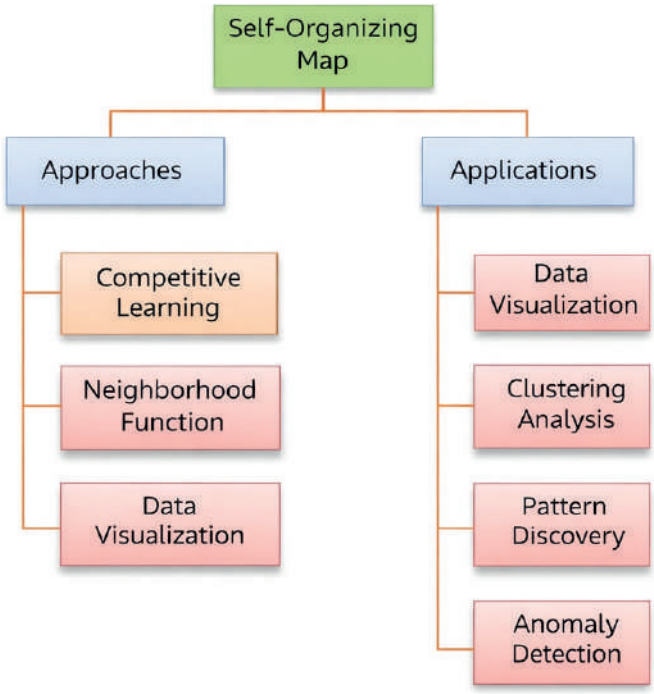


Figure 1.6. Types of the Self-Organizing Map Structure and Applications

Figure 1.6. presents the conceptual framework of the Self-Organizing Map (SOM) and highlights its fundamental operational components and application domains. The SOM learning mechanism is primarily based on competitive learning, where neurons compete to represent input patterns, followed by adaptation governed by a neighborhood function that preserves topological relationships among data samples. Through iterative learning, similar input vectors are mapped to neighboring regions of the network, enabling meaningful visualization of complex multidimensional datasets. As illustrated in the figure, SOM techniques are widely applied in data visualization, clustering analysis, pattern discovery, and anomaly detection tasks. The ability of SOM to simultaneously perform dimensionality reduction

and clustering makes it an effective exploratory analysis tool, particularly in situations where labeled data are unavailable. Consequently, SOM serves as an important bridge between neural computation and unsupervised data mining methodologies.

1.4.3. Clustering Algorithm

Cluster analysis or clustering algorithm is to group a set of objects so that objects in a group or cluster have a higher degree of similarity to each other compared to similarity to members of other clusters. Clustering is a mainly applied to exploratory data mining and this is a technique commonly used for statistical data analysis, and it is employed in several fields such as ML, information retrieval, pattern recognition, bioinformatics, and image analysis (Jain, 2010). Cluster analysis per se is not considered as an algorithm; rather it is taken onto account as a general task to be done. This task can be performed by different algorithms that significantly differ in the notion regarding to what constitutes a cluster and how to find them in an efficient way. The popular notions of clusters involve groups in which there are small distances amongst the members of cluster, dense areas of data space, intervals or certain statistical distributions.

Clustering is applied to several fields, for example information retrieval and knowledge discovery. This helps to find more quickly related information. This way, researchers can stay up to date with the latest findings in their fields. Currently, clustering has attracted the attention of many scholars as a tool for classification, decision making, information extraction, and pattern analysis.

Vladimir Estivill-Castro believes that the term "cluster" cannot be defined precisely, and this condition has led to emergence of so many clustering algorithms. The common denomination is a group of data objects. Though, different researchers make use of various cluster models for each of which various algorithms can be presented. As found by various algorithms, the notion of cluster significantly varies in its properties. To understand these "cluster models" is the most important point in understanding differences that exist among different algorithms.

Clustering analysis can be divided into several models: connectivity models, distribution models, density models, subspace models, group models, graph-based models, and centroid models. Figure 1.7 shows different types of clustering algorithm.

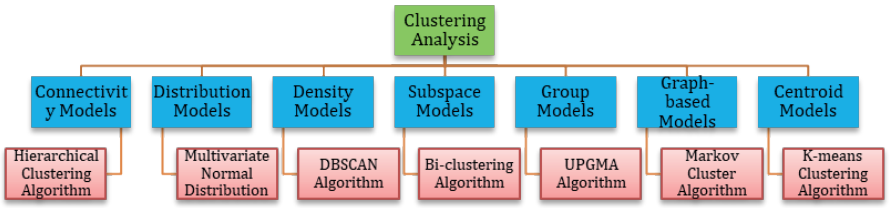


Figure 1.7. Types of Clustering Algorithm

Connectivity model is a model for clustering that is designed based on distance connectivity. One of major examples for connectivity model is hierarchical clustering algorithm. The hierarchical clustering is an algorithm of cluster analysis which attempts to build a hierarchy of clusters. Bottom up and top down are two strategies used in the hierarchical clustering.

Distribution model is a model for clustering in which clusters are modeled using statistical distributions. One of examples for distribution model is multivariate normal distributions used by the expectation-maximization algorithm.

Density model is a model for clustering in which clusters are defined as connected dense regions in data space. Two main examples for density model are Density-Based Spatial Clustering of Applications with Noise algorithm (DBSCAN algorithm) and Ordering Points to Identify the Clustering Structure algorithm (OPTICS algorithm).

Subspace model is a model for clustering in which clusters are modeled with both cluster members and relevant attributes. An example for subspace model is Bi-clustering algorithm that is also known as Co-clustering algorithm. Group model is a model for clustering in which algorithms do not provide a refined model for their results and just provide the grouping information. An example for group model is Un-Weighted Pair Group Method with Arithmetic Mean algorithm (UPGMA Algorithm).

Graph-based model is one of the models for clustering, which is a subset of nodes in a graph such that every two nodes in the subset are connected with an edge that can be considered as a prototypical form of cluster. One instance for this model is Markov Cluster Algorithm. Centroid is a model for clustering in which the similarity of two clusters is defined as the similarity of their centroids. One example for this model is *K*-means clustering algorithm that represents each cluster by a single mean vector. The *K*-means clustering algorithm is explained in the next section.

1.5. Basic Terminology and Notation

Machine learning systems rely on a set of fundamental terminologies and mathematical notations that provide a common framework for describing datasets, models, and learning processes. Understanding these concepts is essential for interpreting machine learning algorithms and evaluating their performance in practical applications.

A dataset represents a structured collection of observations used for training and evaluating learning models. Typically, datasets are organized in tabular form where rows correspond to individual samples and columns represent measurable attributes known as features. Each observation within the dataset is referred to as an instance or sample, describing a single entity such as a patient record, an image, or a sensor measurement.

A feature denotes an individual characteristic or measurable property of an instance that serves as input to the learning algorithm. Features may be numerical, categorical, ordinal, or binary depending on the nature of the problem domain. In supervised learning environments, each instance is associated with a label, representing the ground-truth output that the model attempts to predict.

Machine learning datasets are commonly divided into three subsets: the training set, validation set, and testing set. The training set is used to learn model parameters, the validation set assists in hyper parameter tuning and model selection, and the testing set provides an unbiased evaluation of generalization performance.

A model can be defined as a mathematical function that maps input space X to output space Y. Each possible configuration of model parameters represents a hypothesis, and the collection of all hypotheses forms the hypothesis space. Learning occurs through optimization of a loss function, which measures the discrepancy between predicted outputs and true values.

Standard notation frequently used in machine learning includes input vectors x_i, true outputs y_i, predicted outputs y^i, dataset D, and model parameters W. Consistent use of notation improves clarity, reproducibility, and scientific communication across machine learning research.

1.6. Data Representation and Preprocessing

Data representation and preprocessing constitute one of the most critical stages in the machine learning workflow. Real-world data is often incomplete, noisy, inconsistent, or heterogeneous, making preprocessing an essential prerequisite before applying learning algorithms.

Machine learning data appears in multiple formats, including numerical data, categorical attributes, textual information, and image-based representations. Numerical data consists of measurable quantities such as temperature or income values, whereas categorical data represents qualitative properties such as gender or geographic location. Since many learning algorithms operate on numerical inputs, categorical variables must be encoded using techniques such as label encoding or one-hot encoding. Textual data requires specialized preprocessing steps including tokenization, stop-word removal, and vectorization methods such as Bag-of-Words or TF–IDF representations. Similarly, image data is represented as multidimensional pixel matrices and typically undergoes normalization, resizing, and augmentation procedures prior to learning.

Data cleaning is another essential preprocessing step aimed at removing duplicate records, correcting inconsistencies, and detecting outliers. Additionally, handling missing values plays a crucial role in maintaining dataset integrity. Common strategies include deletion methods, statistical imputation, and model-based estimation.

Feature scaling techniques such as normalization and standardization ensure that variables contribute equally during model training. Without proper scaling, features with larger numerical ranges may dominate optimization processes, leading to unstable learning behavior. Effective preprocessing significantly improves model accuracy, accelerates convergence, and enhances generalization capability, making it a foundational component of modern machine learning systems.

1.7. Feature Engineering

Feature engineering refers to the process of transforming raw data into meaningful representations that improve machine learning performance. It is widely recognized that the success of a learning system often depends more on feature quality than on algorithmic complexity.

Feature engineering encompasses three primary operations: feature selection, feature extraction, and feature scaling. Feature selection aims to identify the most informative variables while eliminating redundant or irrelevant attributes that may introduce noise or increase computational cost. Common selection approaches include filter methods based on statistical measures, wrapper methods relying on predictive performance, and embedded techniques integrated within learning algorithms. Feature extraction, in contrast, generates new features by transforming original variables into compact representations. Techniques such as Principal Component Analysis, signal transformation

methods, and deep neural embedding's enable efficient representation of complex data structures.

Feature scaling ensures numerical consistency among variables, particularly in algorithms sensitive to distance calculations or gradient optimization. Methods such as Min–Max scaling, standardization, and robust scaling are widely applied to stabilize learning processes. Well-designed feature engineering improves interpretability, reduces overfitting risk, and enhances predictive accuracy. In many real-world applications, carefully engineered features enable simpler models to outperform more complex algorithms applied to poorly prepared data.

1.8. Conclusion

Machine learning has emerged as one of the most transformative technologies of modern computational science, enabling intelligent systems to learn from data and make informed decisions without explicit programming. This book chapter presented a comprehensive introduction to the fundamental concepts underlying machine learning, establishing a solid theoretical foundation for understanding advanced learning models and methodologies. The discussion began with an overview of machine learning principles and learning paradigms, including supervised, unsupervised, semi-supervised, and reinforcement learning approaches. These paradigms demonstrate how different learning strategies address diverse problem domains depending on data availability and learning objectives.

Essential terminology and notations were introduced to provide a common framework for describing datasets, models, hypotheses, and evaluation procedures. Proper understanding of these concepts is crucial for interpreting machine learning algorithms and conducting scientifically valid experiments. The chapter further emphasized the importance of data representation and preprocessing, highlighting how raw data must be transformed into structured and meaningful formats before learning can occur. Feature engineering techniques, including feature selection, extraction, and scaling, were discussed as key mechanisms for improving model efficiency and predictive performance.

The learning process was examined through model training, generalization capability, and challenges such as overfitting and under fitting. The bias–variance tradeoff was presented as a central principle governing model complexity and prediction reliability. In addition, systematic model evaluation techniques and optimization concepts were explored to demonstrate how learning algorithms achieve optimal performance through iterative improvement.

Finally, ethical and practical considerations—including data bias, interpretability, and privacy protection—were addressed to underline the responsibility associated with deploying machine learning systems in real-world environments. In conclusion, mastering the foundational concepts of machine learning is essential for researchers, engineers, and practitioners seeking to design robust, efficient, and trustworthy intelligent systems. The principles introduced in this chapter provide the conceptual basis for more advanced topics such as deep learning, hybrid intelligent systems, optimization strategies, and real-world machine learning applications discussed in subsequent chapters of this book.

References

Mitchell, T. M. (1997). Machine Learning. New York, USA: McGraw-Hill.

Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer, New York.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT Press.

Alpaydin, E. (2020). Introduction to Machine Learning (4th ed.). MIT Press.

Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.

Geron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media.

Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction (2nd ed.). MIT Press.

Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques (3rd ed.). Morgan Kaufmann.

Kelleher, J. D., Namee, B. M., & D'Arcy, A. (2020). Fundamentals of Machine Learning for Predictive Data Analytics. MIT Press.

Zhou, Z.-H. (2021). Machine Learning. Springer Nature.

Farhang, Y. (2016) Hybrid Optimization for K-means Clustering Learning Enhancement.

Farhang, Y. (2017) Face Extraction from Image Based on K-means Clustering Algorithms.

Farhang, Y. (2017) Development of the Meta-Heuristic PSOGA with K-means Algorithm.

Chapter 2

Hybrid Methods of Machine Learning: Taxonomies, Architectures, and Optimization

Ülker Başar[1]

Abstract

Although machine learning models have found a wide range of applications in the literature, being used in pattern discovery, predictive modeling, and decision-making processes in complex data spaces, single algorithms do not possess a universal theoretical and algorithmic superiority across all problem spaces, as mathematically emphasized by the "No Free Lunch" theorem (Wolpert and Macready, 1997). Therefore, hybrid machine learning approaches, which aim to increase generalization power, reduce the risk of getting stuck in local minima, and enhance model robustness by integrating the mathematical, statistical, and computational advantages of different learning paradigms, have been gaining increasing attention in recent years.

This book chapter is structured as a comprehensive literature review addressing the theoretical development of hybrid machine learning methods through the lens of taxonomies, computational architectures, and optimization strategies. The study systematically synthesizes leading and current work in the field, delving deeply into the variance and bias-reduction effects of ensemble learning approaches (bagging, boosting, stacking) and the conceptual integration of symbolic and sub-symbolic methods. Furthermore, sequential, parallel, and hierarchical hybrid architectures are analyzed with respect to information transfer among component models, deep feature fusion mechanisms, and coupling levels. In addition, the integration of metaheuristic algorithms, such as Genetic Algorithms (GA) and Particle Swarm Optimization (PSO), with machine learning is discussed, with approaches proposed in the literature for hyperparameter space exploration, dynamic feature selection, and loss function optimization. This chapter aims to go beyond a purely performance-oriented review and provide a contemporary theoretical reference that highlights how hybrid systems overcome fundamental limitations, including the balance between bias and variance, interpretability and verification, and computational complexity.

1 Asst. Prof. Dr., Istanbul Esenyurt University, Faculty of Business and Administrative Sciences, Department of Management Information Systems, ulkerbasar@esenyurt.edu.tr, 0009-0000-6720-4161.

https://doi.org/10.58830/ozgur.pub1236.c5003

1. Introduction

Machine learning models have revolutionized the literature by finding wide application in pattern discovery, predictive modeling, and decision-making processes in complex data spaces. However, none of these data-driven singular algorithms possesses universal theoretical and algorithmic superiority across all problem spaces. This fundamental limitation is mathematically confirmed by the "No Free Lunch" theorem (by Wolpert and Macready, 1997). While deep learning networks or traditional statistical models can achieve high accuracy on certain data architectures, the conceptual and computational limitations of these paradigms become apparent when faced with noisy data, limited sample sizes, and non-linear (non-convex) NP-hard optimization scenarios.

To overcome these theoretical bottlenecks, Hybrid Machine Learning (HML) approaches, which synergistically synthesize the mathematical, statistical, and algorithmic advantages of different learning paradigms, have become central to modern computer science. The primary goal of hybrid systems is not merely to provide an empirical performance increase; the main objective is to enhance generalization capacity, minimize the risk of getting stuck in local minima during optimization, and maximize model robustness. In doing so, the variance and bias dilemmas of predictive models are balanced, while the "black box" nature of data-driven models is made more transparent and verifiable through rule-based or meta-heuristic approaches.

This section presents the theoretical evolution of hybrid machine learning methods through a systematic literature synthesis, focusing on taxonomies, computational architectures, and optimization strategies. To achieve this fundamental goal, the following sections of the introduction will sequentially: construct the theoretical foundation by examining the practical implications of the "No Free Lunch" theorem; discuss the increasing complexity of modern data spaces and the violated assumptions of singular models; then, explain why hybrid approaches are a necessity to overcome these bottlenecks through four main integration categories; and finally, detail the theoretical contribution and scope of this study to the literature.

1.1. Pedagogical Interpretation of the "No Free Lunch" Theorem: The Impossibility of a Single Superior Model

The No Free Lunch (NFL) theorems formalize a striking truth about the nature of optimization and learning algorithms: when performance is averaged across all possible problems, no algorithm can be superior to another (Wolpert & Macready, 1997). In other words, when no prior knowledge of the problem distribution is available, there is no universally "best" learner or optimizer. In

this section, we offer a pedagogical interpretation of this theorem, treating it not as a limitation in machine learning practice but as a guide explaining the necessity of hybrid models.

At the heart of the theorem lies the "averaging principle": the expected performance of an algorithm on the entire possible problem space is the same as that of any other algorithm. This is similar to choosing a toolbox without knowing what the job is; a hammer is not "better" than a screwdriver, they are simply designed for different tasks (Wolpert & Macready, 1997). In machine learning, algorithms are effective for specific problem structures (e.g., linearity, low dimensionality, convexity, etc.). Therefore, high performance achieved by an algorithm on a particular benchmark dataset cannot be interpreted as its universal superiority; rather, it demonstrates the compatibility between the algorithm's assumptions and the dataset's structure. This highlights the challenges of attempting to solve problems from different domains with the same algorithm (Kotary et al., 2021).

Based on this theoretical framework, we can say that successful real-world applications stem not from an intrinsic superiority of algorithms, but from domain knowledge and constraints successfully adapted to the problem structure. For example, the hybrid model (Stacked AutoEncoder + PSO + Softmax) developed by (Bülbül and Işık, 2024) for predicting survival after a heart attack is successful thanks to a preprocessing (SAE) and optimization (PSO) strategy that takes into account the high-dimensional and noisy nature of medical data, while a pure Softmax classifier may be insufficient for the same problem. Similarly, Cross-Entropy (CE) based methods developed to solve the hybrid pre-coding problem in millimeter-wave communication systems (Li et al., 2019; Gao et al., 2017) do not carry either the computational overhead of a pure optimization algorithm (full search) or the inadequacy of a pure learning model in modeling hardware constraints. These examples demonstrate the practical echo of the NFL theorem: Success lies in the fit between the algorithm and the problem, i.e., often in hybridization.

This pedagogical interpretation transforms the NFL theorem from a mere "impossibility" into a "roadmap" for the machine learning designer. It teaches us that, instead of searching for the single best algorithm, we should analyze the problem's constraints and design hybrid architectures that combine the strengths of different algorithms. This understanding leads us to the question, which we will examine in the next subsection, of why hybrid models have become a necessity, not just a preference, in the face of the ever-increasing complexity of today's data.

1.2. The Increasing Nature of Complexity: Violated Assumptions and the Limitations of Singular Models

In the previous section, the pedagogical interpretation of the "No Free Lunch" (NFL) theorem demonstrated the futility of seeking a universally superior model. However, this theoretical framework raises a far more pressing question in practice: By what concrete mechanisms do today's real-world problems make the failure of singular models inevitable? The answer lies in the increasing complexity of modern data and problems. This chapter examines how this complexity systematically violates the fundamental assumptions of classical machine learning models and how these violations sharpen the limitations of singular approaches. This analysis will provide the fundamental rationale for why hybrid architectures have become a necessity rather than a preference.

1.2.1. Assumptions in Violation: The Anatomy of Complexity

To understand why singular models fail, we must first dissect the specific assumptions that modern data systematically violates. The complexity of real-world problems manifests across interrelated yet distinct dimensions. Each of these dimensions weakens or completely invalidates the assumptions that enable the success of individual models.

Most machine learning models assume that training and test data come from the same distribution (IID assumption). As the NFL theorem points out, no algorithm can establish universal superiority without prior knowledge of the problem distribution (Wolpert & Macready, 1997). In practice, this means that data distributions in fields such as finance, healthcare, or engineering can change over time (concept drift) or enter unexpected regimes. Relying on a single model risks a sharp drop in performance in the face of distributional shift or contextual patterns not represented in the training set (Wolpert & Macready, 1997; Badran et al., 2020; Labbani et al., 2023).

Single models typically assume an immutable data structure and carry fixed inductive biases (e.g., a specific kernel structure, a predetermined architecture, or a fixed feature preprocessing process) that conform to this structure. As problem complexity increases, the fit between these fixed biases and the actual data generation mechanisms deteriorates, increasing the risk of model misspecification (Kotary et al., 2021; Bülbül & Işık, 2024). This can lead to overfitting to a narrow family of problems or weaken the ability to generalize (Gao et al., 2017; Ding et al., 2011).

Purely data-driven models tend to ignore physics, governance, or operational constraints critical to robust performance, such as hardware constraints in

millimeter-wave communication systems (Li et al., 2019), safety frameworks in clinical decision-making processes (Bülbül & Işık, 2024), or conservation laws in physical processes. NFL-based arguments show that learning without domain knowledge can explore unrealistic regions of the solution space and fail in practice (Li et al., 2019).

In many applied fields, such as fraud detection, biomedical prediction, or industrial anomaly detection, data are unbalanced, with few samples for some classes, or highly heterogeneous across subdomains (Bülbül & Işık, 2024). As NFL points out, the same algorithm may perform well in some subpopulations while performing poorly in others; this contradicts the idea of a universally superior model (Kumar et al., 2023).

Real-world deployments impose strict limitations on training time, energy consumption, and data throughput. Models that demand extensive training resources or massive labeled datasets often become impractical under these constraints (Bülbül, 2023).

1.2.2. Limitations of Singular Models: Consequences of Assumption Violations

These violations are not merely theoretical; they manifest as concrete limitations that cripple the performance of singular models in practice. These assumption violations bring with them a number of fundamental limitations that singular models encounter in complex systems:

Singular models encoding a single inferential bias fall short of optimality when problems exhibit structures that vary across tasks or regimes (Labbani et al., 2023). Hybrid approaches accommodate this diversity by designing components tailored to sub-problems, using ontologies, prior knowledge, and modular design (Li et al., 2019).

As highlighted by the NFL, a performance advantage in one problem class does not transfer to all other classes without prior knowledge of the problem distribution. This explains why hybrid models incorporating domain knowledge (physics, ontologies, regulatory constraints) generalize better to heterogeneous data (Wolpert & Macready, 1997; Labbani et al., 2023).

Complex systems often require balancing multiple criteria such as accuracy, energy efficiency, latency, and interpretability. Single-model optimization for a single objective falls short when trade-offs change. Hybrid architectures frequently integrate multi-objective optimization and hardware awareness to navigate such trade-offs (Zhou & Xie, 2025).

When models need to operate in safety, critical or high risk areas, explainability and uncertainty reasoning are vital. Hybrid approaches that combine uncertainty methods (e.g., Dempster-Shafer theory) or interpretable surrogate models with ML components provide more robust decision support compared to opaque single-model systems (Hu et al., 2019).

1.2.3. The Rise of Hybrid Architectures as a Solution

Having diagnosed the limitations, we now turn to the cure: how hybrid architectures are specifically designed to overcome each of these shortcomings. The limitations listed above illustrate that single models alone are insufficient. Hybrid machine learning (HML) offers a range of strategies that address these breached assumptions and overcome the limitations:

By breaking problems down into subtasks and assigning an appropriate model or ontology to each subtask, modular hybrids reduce the burden of a single inferential bias and increase the ability to adapt to domain-specific structures (Labbani et al., 2023). Physics-based or knowledge-driven hybrids require known relationships and constraints. This increases model robustness even when data is sparse or noisy (Das & Winter, 2016). Replacing expensive evaluations with surrogate estimators and incorporating hardware constraints into the search space helps navigate large and complex problem spaces while respecting practical limits (Eslami et al., 2022). Hybrid methods often utilize evolutionary strategies, PSO, and GA/SA hybrids to address the combinatorial explosion of possible designs and mitigate the risk of suboptimal fixed choices (Eslami et al., 2022). Methods that combine evidence-theoretic fusion or ensemble approaches with ML reduce uncertainty and provide more robust estimates under data scarcity or conflicting signals (Kumar et al., 2023; Hu et al., 2019).

1.2.4. Implications for Research and Application

Based on this analysis, some practical guidelines can be derived for researchers and practitioners who want to design robust hybrid systems:

- *Start with Problem-Aware Prior Knowledge and Modular Design:* Create a task decomposition compatible with domain knowledge and identify areas where ML will add value (representation learning, prediction), while also defining areas where prior knowledge should restrict or guide learning (Labbani et al., 2023).
- *Use Surrogate Models and Hardware-Aware Goals to Manage Complexity:* Use surrogate estimators to efficiently eliminate candidates when exploring large design spaces; explicitly encode hardware or domain

constraints into optimization goals to avoid impractical or suboptimal designs (Eslami et al., 2022).

- *Encourage Multi-Objective Optimization and Uncertainty Management:* Integrate multiple objectives and uncertainty reasoning (e.g., evidential methods) to balance competing demands (accuracy, energy, delay, interpretability) across tasks and enhance reliability (Gao et al., 2017; Hu et al., 2019).
- Prioritize *Diverse Benchmarks and Cross-Domain Validation:* To counteract NFL influences, evaluate hybrids on representative distributions and multiple benchmarks. This helps ensure robustness across a broad range of problems and highlights the role of prior knowledge in generalization (Wolpert & Macready, 1997; Bulbul & Isik, 2024).

In summary, the multidimensional complexity of today's data and problems systematically violates the fundamental assumptions of singular models, rendering them inadequate in practice. This diagnosis leads to an unequivocal prescription: hybridization. Hybrid machine learning offers the conceptual and methodological tools necessary to repair these violations, overcome limitations, and generate robust, reliable, and efficient solutions to the complex demands of the real world. With this foundation laid, the following chapters will shift from the 'why' to the 'how,' providing a detailed taxonomy of hybrid architectures and the optimization strategies that power them.

2. Taxonomy of Hybrid Methods

Hybrid machine learning (HML) is not simply a random combination of different algorithms, but a structured integration process serving a specific theoretical purpose. To systematically understand the successful applications presented in the literature, it is necessary to categorize these systems along "integration axes." In this section, HML systems will be examined using a three-dimensional taxonomy defined by what is being combined (Integration Target), why it is being combined (Driving Objective), and where this combination is located in the architecture (Architectural Locus).

2.1. Why Hybrid? A Bias-Variance Perspective

The previous chapter dissected the multidimensional complexity of modern data, revealing how it systematically violates the assumptions of singular models. But how does this violation translate into model failure? The answer lies in the fundamental decomposition of prediction error, a concept formalized in the bias-variance trade-off. This section argues that the primary reason hybrid architectures succeed where singular models fail is their unique ability to

navigate, and often transcend, this trade-off. By combining multiple learning paradigms, hybrid systems can simultaneously address the two primary sources of error bias and variance that singular models are forced to balance against each other.

Any predictive model's error on unseen data can be decomposed into three fundamental components: bias, variance, and irreducible noise. Bias measures the error introduced by approximating a real-world, often highly complex problem with a simplified model. A linear model, for instance, exhibits high bias when tasked with learning a non-linear relationship; it simply lacks the capacity to capture the underlying pattern, leading to systematic underfitting. Variance, on the other hand, measures the model's sensitivity to the peculiarities of a particular training dataset. A deep, unpruned decision tree has high variance; it may perfectly memorize the training data, including its noise, but will fail to generalize to new samples, a classic case of overfitting (Gao et al., 2017). The classical bias-variance trade-off posits that decreasing one often comes at the cost of increasing the other. The art of building a singular model lies in finding the delicate balance that minimizes total error.

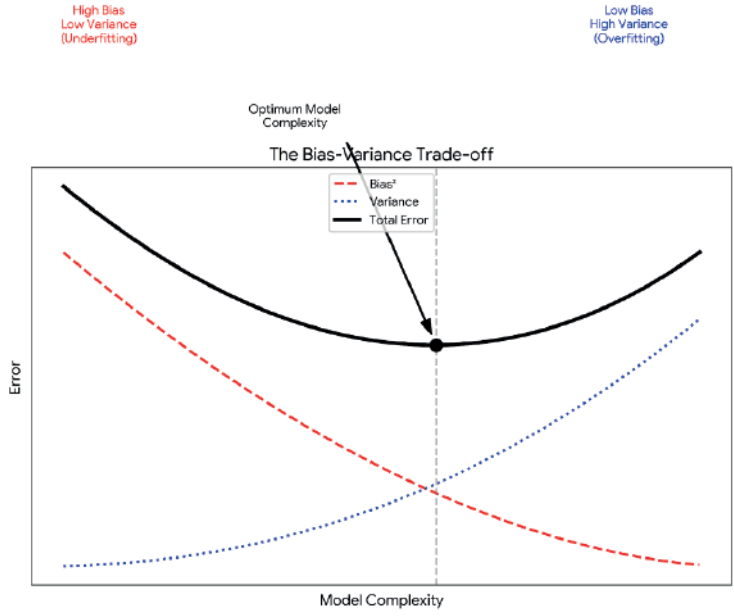


Figure 2.1. The Bias-Variance Trade-off

A singular model is defined by a fixed inductive bias and a specific capacity, whether it be the depth of a tree, the degree of a polynomial, or the number

of layers in a neural network. This fixed capacity anchors it to a specific point on the bias-variance spectrum. A low-capacity model, such as a shallow tree, will exhibit high bias, failing to capture intricate patterns in complex, high-dimensional data, as in the medical study described by (Bülbül and Işık, 2024). Conversely, a high-capacity model, such as a deep neural network, while possessing low bias, becomes susceptible to high variance. It risks overfitting to noise and spurious correlations in the training data, a critical flaw in safety-critical domains such as industrial anomaly detection (Terbuch et al., 2023). The singular model is thus trapped in a zero-sum game: it can only improve one source of error at the expense of the other.

Hybrid architectures, particularly ensemble methods, offer a way out of this trap by decoupling the management of bias and variance. Instead of relying on a single model, they combine multiple models, each with its own inductive bias, to achieve a balance that no individual component can achieve.

Bagging (e.g., Random Forests) primarily targets variance. It trains numerous high-variance models (like deep decision trees) on bootstrap samples of the data and averages their predictions. While each individual tree may overfit and be highly sensitive to its specific training sample, the aggregated prediction smoothes out this volatility, drastically reducing variance without a significant increase in bias (Gao et al., 2017). This makes bagging particularly effective for complex, high-dimensional data where overfitting is a primary concern.

Boosting (e.g., AdaBoost, Gradient Boosting), in contrast, focuses on reducing bias. It sequentially trains a series of weak, high-bias learners (such as shallow trees), with each new model concentrating on the mistakes of its predecessor. This iterative process gradually builds a powerful, low-bias ensemble from simple components. This approach is well-suited for problems with intricate underlying structures that a single simple model cannot capture.

Stacking (Stacked Generalization) takes a different tack by combining heterogeneous models. It trains a diverse set of base learners (e.g., an SVM, a decision tree, and a neural network) and then uses a meta-learner to learn the optimal way to combine their predictions. The hybrid model proposed by (Bülbül and Işık, 2024) which uses a Stacked Autoencoder for feature learning (a form of representation bias reduction) and a PSO-optimized classifier can be viewed through this lens, where different components handle different aspects of the error.

Table 2.1. Comparison of Key Ensemble Methods

Feature	Bagging	Boosting	Stacking
Primary Target	Variance Reduction	Bias Reduction	Both (through model diversity)
Model Type	Homogeneous (typically of the same type)	Homogeneous (typically of the same type)	Heterogeneous (different types)
Training Method	Parallel (independent)	Sequential (dependent)	Parallel (base) + Sequential (meta)
Example Algorithm	Random Forest	AdaBoost, Gradient Boosting	Blending, Super Learner
Bias Effect	Invariant (fixed)	Decreases	Combines the strengths of the models
Variance Effect	Decreases	May increase if not controlled	Balanced

Beyond ensemble methods, hybrid systems also leverage metaheuristic optimization to directly search for model configurations that strike the optimal bias-variance balance for a given task. A neural network's capacity and thus its position on the bias-variance spectrum is determined by its architecture (number of layers, neurons) and hyperparameters (learning rate, regularization). Fixed, manually chosen configurations are unlikely to be optimal.

Metaheuristics such as Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) provide a powerful mechanism for navigating this complex configuration space. They can optimize a network's weights (Ding et al., 2011), its architecture or even the placement of hyperparameters within the network (Akl et al., 2019). By treating model design as an optimization problem, these hybrid approaches can discover architectures that are complex enough to capture the true signal (low bias) yet regularized enough to exhibit low variance (resistant to noise). As (Bülbül, 2023) demonstrates, a Gray Wolf Optimizer can effectively tune a Multi-Layer Perceptron, implicitly guiding it towards a more favorable error profile than a default or manually-tuned version could achieve.

In essence, the bias-variance perspective reveals why hybridization is not merely a pragmatic trick but a theoretically grounded necessity. Singular models are constrained by a zero-sum game between bias and variance. Hybrid systems, whether through ensemble methods like bagging and boosting or through metaheuristic-driven architecture search, break these constraints. They

manage the two sources of error more independently, building models that can be both complex enough to learn intricate patterns and robust enough to generalize to new data. Having established the why, the following section shifts focus to the what, providing a taxonomy that categorizes hybrid methods based on what they integrate, why they integrate it, and where in the learning pipeline this integration occurs.

2.2. A Taxonomy of Hybrid Methods

Having established the theoretical necessity (Section 1.1), the practical urgency (Section 1.2), and the mechanistic advantage (Section 2.1) of hybrid machine learning, a crucial question emerges: How do we systematically categorize the vast and diverse landscape of hybrid approaches? The literature presents a multitude of hybrid designs, each tailored to specific problems and paradigms. Without a coherent taxonomy, comparing these methods, understanding their trade-offs, and selecting the appropriate architecture for a new problem becomes a daunting task. This section proposes a taxonomy organized along three primary axes; the integration target (what is being hybridized), the driving objective (why the hybridization is performed), and the architectural locus (where in the learning pipeline the hybridization occurs). This framework, synthesized from representative works in the field, provides a conceptual map to navigate the hybrid ML landscape.

The proposed taxonomy rests on three fundamental questions about any hybrid system:

1) What is being hybridized? : Does the system combine different learning paradigms (e.g., symbolic and sub-symbolic), learning with optimization, or machine learning with domain knowledge?
2) Why is the hybrid used? : Is the goal to improve predictive accuracy, enforce physical constraints, accelerate computation, enhance interpretability, or enable learning under limited data?
3) Where does hybridization occur?: Is the integration at the data level, the model level (parallel, sequential, or hierarchical), or the optimization level (e.g., metaheuristic-guided search)?

The following subsections elaborate on these axes with concrete examples from the literature, highlighting how different combinations yield distinct families of hybrid methods.

2.2.1. Optimization-Learning Hybrids (Learning to Optimize)

This family of hybrids integrates machine learning components directly into traditional optimization pipelines. The core idea is to use data-driven models to predict intermediate information or heuristics that can accelerate combinatorial solvers, such as branching and cutting rules in Mixed-Integer Programming (MIP) solvers (Kotary et al., 2021).

- *Integration Target:* Machine learning models (e.g., neural networks) + combinatorial optimization solvers (e.g., MIP, constraint programming).
- *Driving Objective:* To obtain fast, approximate solutions to otherwise intractable combinatorial problems and to enable structural inference by leveraging learned priors for solver components. The goal is to combine the pattern-recognition capabilities of ML with the rigorous constraint satisfaction of optimization.
- *Architectural Locus:* The ML component is embedded within the solver's loop, guiding search, pruning, or heuristic selection. This creates a feedback loop where the solver's performance informs the learning process.
- *Example:* (Kotary et al., 2021) survey end-to-end constrained optimization learning frameworks, where a model learns to predict optimal solutions or useful heuristics, which are then refined by a differentiable optimizer. In wireless communications, (Li et al., 2019) and (Gao et al., 2017) use learning-augmented Cross-Entropy methods to guide discrete decisions in hybrid precoding, achieving near-optimal spectral efficiency with drastically reduced computational cost.

2.2.2. Hardware-Aware and Wireless System Hybrids

A specialized but highly impactful class of hybrids emerges from engineering domains where learning must operate under strict physical and hardware constraints. These systems jointly design neural architectures and optimization methods to solve problems in which the solution space is constrained by real-world limitations.

1) *Integration Target:* Machine learning modules (for prediction or feature extraction) + hardware-aware optimization algorithms (e.g., Cross-Entropy, heuristic search) + physical system models.
2) *Driving Objective:* To reduce hardware complexity and computational burden while achieving high performance (e.g., spectral efficiency). The

hybrid must respect constraints such as low-resolution ADCs, limited RF chains, or power budgets.

3) *Architectural Locus:* End-to-end ML modules are embedded within physical-layer optimization loops. Stochastic optimization techniques (such as CE) are often used to guide discrete decisions that are not differentiable.

4) *Example:* (Li et al., 2019) tackle hybrid precoding in mmWave MIMO systems. Their approach uses a CE-inspired method to search for optimal precoder/combiner configurations under hardware constraints, achieving superior spectral efficiency compared to both pure optimization (exhaustive search is too costly) and pure learning (which may ignore hardware limits).

2.2.3. Quantum-Classical Hybrid Learning Systems

The emergence of quantum computing has given rise to a novel class of hybrids that integrate parameterized quantum circuits (PQCs) with classical architectures. These systems aim to leverage the potential advantages of quantum computation for specific learning tasks while relying on classical methods for control and optimization.

1) *Integration Target:* Parameterized quantum circuits (PQCs) + classical neural networks or optimization algorithms + architecture search mechanisms.

2) *Driving Objective:* To exploit potential learning advantages of PQCs for certain tasks (e.g., reinforcement learning, chemistry simulations) while mitigating the effects of quantum noise, decoherence, and limited qubit counts. Architecture search aims to balance learning performance against quantum resource constraints.

3) *Architectural Locus:* An outer optimization/search loop (classical) guides the design of the PQC, with inner evaluations performed on quantum hardware or simulators. This often resembles Neural Architecture Search (NAS) but with quantum-specific considerations.

4) *Example:* Ding and Spector (2023) introduce MEAS-PQC (Multi-Objective Evolutionary Architecture Search for Parameterized Quantum Circuits), a framework that evolves PQC architectures to optimize for both task performance and robustness to quantum noise, explicitly acknowledging the hardware reality of near-term quantum devices.

2.2.4. Neural Architecture Search (NAS) and Hyperparameter Optimization Hybrids

This category addresses the meta-problem of designing the learning system itself. Hybrid approaches here combine search strategies (evolutionary, Bayesian, gradient-based) with surrogate models and performance predictors to navigate the combinatorial explosion of possible architectures and hyperparameters.

1) *Integration Target:* Search algorithms (e.g., evolutionary strategies, PSO, Bayesian optimization) + performance estimation methods (e.g., surrogate models, weight sharing) + the architecture/hyperparameter space itself.
2) *Driving Objective:* To automate the discovery of high-performing neural network architectures and their associated hyperparameters, reducing the need for manual, expert-driven design and improving sample efficiency.
3) *Architectural Locus:* The hybridization occurs in the search loop. A search strategy proposes candidate architectures, a surrogate model or fast evaluator estimates their performance, and the results guide the next generation of candidates.
4) *Example:* (Eslami et al., 2022) propose learning a unified latent space for NAS, leveraging structural and symbolic information to improve search efficiency. (Bülbül, 2023) uses a Gray Wolf Optimizer to tune a Multi-Layer Perceptron for medical diagnosis, while (Akl et al., 2019) explore the novel concept of optimizing not just hyperparameter values but also their positions within a deep network.

2.2.5. Data-Driven and Physics/Medical-Informed Hybrids

In domains like healthcare and engineering, pure data-driven models often struggle due to data scarcity, noise, or the need for safety-critical guarantees. This family of hybrids explicitly incorporates domain knowledge, such as physical laws, clinical guidelines, or ontologies, to constrain and guide the learning process.

1) *Integration Target:* Data-driven models (e.g., autoencoders, neural networks) + domain knowledge representations (e.g., physics-based equations, ontologies, expert rules, Key Performance Indicators) + optimization algorithms (e.g., PSO for tuning).
2) *Driving Objective:* To leverage the pattern recognition strengths of ML while ensuring that predictions remain consistent with known

domain constraints, thereby improving robustness, interpretability, and generalization, especially under limited data regimes.

3) *Architectural Locus:* Modular pipelines where learned representations are combined with knowledge-derived features, or where domain constraints are encoded into the loss function or optimization process.

4) *Example:* (Bülbül and Işık, 2024) combine Stacked Autoencoders for representation learning with PSO-optimized classifiers for survival prediction, implicitly using medical domain knowledge to structure the problem. (Das and Winter, 2016) develop a hybrid knowledge-driven framework for GPS trajectory classification that integrates ontological rules with data-driven components. (Losada and Terranova, 2024) bridge pharmacology and neural networks using Neural Ordinary Differential Equations (Neural ODEs) that embed pharmacological knowledge into the learning process.

2.2.6. Hybrid Time-Series Anomaly Detection and Fault Diagnosis

Industrial and cyber-physical systems generate multivariate time-series data that is often high-dimensional, noisy, and non-stationary. Hybrid approaches in this domain fuse multiple techniques to robustly detect anomalies and diagnose faults.

1) *Integration Target:* Generative models (e.g., autoencoders, variational models) + evolutionary algorithms (e.g., GA for hyperparameter tuning) + domain-informed indicators (e.g., Key Performance Indicators).

2) *Driving Objective:* To exploit the redundancy and feature extraction capabilities of neural models while leveraging domain knowledge (KPI-based indicators) and evolutionary search to customize architectures for specific industrial settings. The goal is to improve detection performance while reducing false positives.

3) *Architectural Locus:* Ensemble or hybrid pipelines that combine generative feature learning with evolutionary optimization for architecture and hyperparameter selection. Domain knowledge is often injected at the feature level.

4) *Example:* (Terbuch et al., 2023) propose a hybrid pipeline for detecting anomalies in multivariate industrial time-series that fuses KPI-driven indicators with neural models and GA-based hyperparameter tuning. This combination allows the system to capture both expected (rule-based) and unexpected (data-driven) anomalies.

The six categories outlined above, while diverse, share a common theme: they are all motivated by the desire to leverage complementary strengths; data-driven learning for pattern abstraction, optimization techniques for search and control, and domain-specific priors for robustness and feasibility to achieve performance unattainable by a single paradigm. Several works explicitly discuss architecture search or optimization as a core component (Ding & Spector, 2023; Eslami et al., 2022; Bülbül, 2023), while others emphasize end-to-end integration with constraints or hardware considerations (Kotary et al., 2021; Li et al., 2019; Bülbül & Işık, 2024).

This taxonomy provides a high-level map of the hybrid ML landscape, categorizing approaches by what they integrate and why. However, understanding what is integrated is only half the story. The next section delves deeper into the how, examining the recurring architectural patterns; sequential, parallel, and hierarchical that define the flow of information and control within these hybrid systems. By mapping the categories from this taxonomy onto the architectural patterns of Section 2.3, we can gain a comprehensive understanding of how hybrid systems are constructed and why they succeed.

Table 2.2. A Taxonomy of Hybrid Machine Learning Methods

Kategori	**Integration Target (What?)**	**Driving Objective (Why?)**	**Architectural Locus (Where?)**	**Representative Works**
Optimization-Learning Hybrids	ML models + combinatorial solvers	Fast, approximate solutions to intractable problems; learned heuristics	ML embedded in solver loop (branching, cutting, search guidance)	Kotary et al. (2021)
Hardware-Aware Hybrids	ML + hardware-aware optimization (CE)	Respect physical constraints (ADC, RF chains); reduce complexity	ML in physical-layer optimization loops	Li et al. (2019); Gao et al. (2017)
Quantum-Classical Hybrids	PQCs + classical architectures + NAS	Leverage quantum advantage while mitigating noise; resource-aware design	Outer classical search loop + inner quantum evaluation	Ding & Spector (2023)

NAS & Hyperparameter Hybrids	Search algorithms + surrogate models + architecture space	Automate model design; improve sample efficiency	Meta-level search loop with performance prediction	Eslami et al. (2022); Bülbül (2023); Akl et al. (2019)
Physics/Medical-Informed Hybrids	Data-driven models + domain knowledge (physics, ontologies, KPIs)	Ensure consistency with known constraints; improve robustness under limited data	Modular pipelines; knowledge in features, loss, or constraints	Bülbül & Işık (2024); Das & Winter (2016); Losada & Terranova (2024)
Anomaly Detection Hybrids	Generative models + evolutionary search + domain indicators	Customize architectures for industrial settings; fuse rule-based and data-driven signals	Ensemble/ hybrid pipelines with feature fusion	Terbuch et al. (2023);

2.3. Architectural Patterns and Information Flow

The taxonomy presented in Section 2.2 provides a high-level map of hybrid machine learning, categorizing approaches by what they integrate and why. However, understanding what is integrated is only half the story. The how the architectural blueprint that governs the interaction and information flow between components is equally critical to a hybrid system's success. Different architectural patterns dictate how errors propagate, how knowledge is fused, and how the system scales with complexity. This section delves into these architectural blueprints, examining three fundamental patterns: sequential, parallel, and hierarchical architectures. By mapping the categories from our taxonomy onto these patterns, we gain a comprehensive understanding of how hybrid systems are constructed and why certain architectures are better suited for specific tasks.

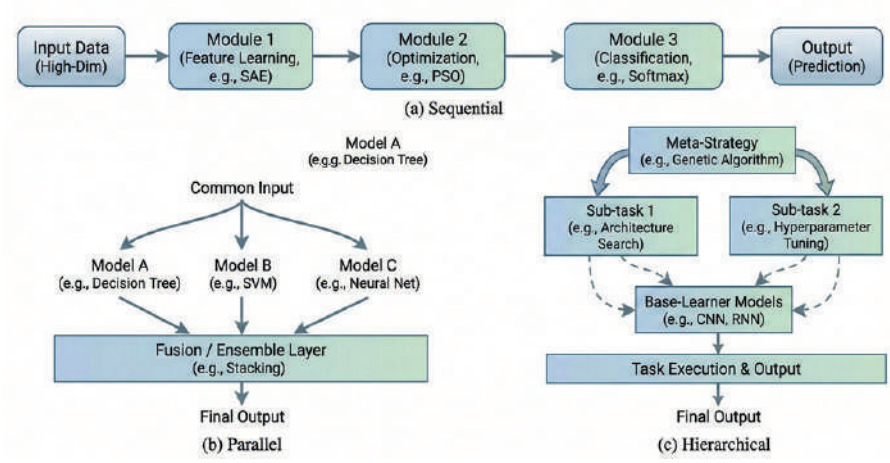


Figure 2.2. Modern Architectural Patterns in Hybrid Machine Learning

2.3.1. Sequential Architectures: Pipelines of Processing

In a sequential hybrid architecture, components are arranged in a chain, where the output of one module serves as the input to the next. This creates a processing pipeline, with each stage responsible for a specific transformation of the data or intermediate representations. This pattern is intuitive and modular, allowing each component to be developed, tested, and optimized independently for its specific subtask.

A classic example from the healthcare domain is the hybrid model proposed by (Bülbül and Işık, 2024) for survival prediction. Their architecture operates sequentially: first, a Stacked Autoencoder (SAE) denoises and reduces the dimensionality of the raw medical data; second, a Particle Swarm Optimization (PSO) algorithm tunes the hyperparameters of a classifier; and finally, a Softmax classifier produces the prediction. Each stage refines the output of the previous one, culminating in a final decision.

Similarly, in the optimization-learning hybrids surveyed by (Kotary et al., 2021), a machine learning model is often used as a sequential preprocessor for a traditional solver. The ML component predicts useful heuristics or warm start points, which are then fed into a constrained optimizer to refine the solution. This pipeline leverages the pattern-recognition strengths of ML to accelerate the rigorous but slower optimization process.

The primary advantage of sequential architectures is their clarity and modularity. However, they suffer from a critical vulnerability: error propagation. Any error or bias introduced at an early stage is compounded and propagated

through the pipeline, potentially corrupting the final output. This makes the design of each stage, especially the first, a matter of utmost importance.

2.3.2. Parallel Architectures: Ensemble and Consensus

Parallel architectures take a different approach: multiple, often independent, models process the same input simultaneously, and their outputs are combined through a fusion mechanism to produce a final decision. This pattern is the foundation of ensemble learning and is particularly effective at reducing variance and improving robustness.

The most prominent examples of parallel architectures are bagging methods like Random Forests. As discussed in Section 2.1, a Random Forest trains numerous high-variance decision trees in parallel on bootstrap samples of the data. Each tree produces its own prediction, and the final output is obtained by averaging (for regression) or voting (for classification). The independence of the trees ensures that their errors are uncorrelated, and averaging smooths out this volatility, leading to a low-variance ensemble (Gao et al., 2017).

Parallel architectures are not limited to homogeneous ensembles. Heterogeneous systems also benefit from this pattern. For instance, in the anomaly detection framework proposed by (Terbuch et al., 2023), a knowledge-driven module (based on Key Performance Indicators) and a data-driven neural model operate in parallel. Their outputs are then fused to produce a more comprehensive and robust anomaly score than either could provide on its own. This is a classic example of late fusion, where decisions from diverse, parallel pathways are combined.

The key to a successful parallel architecture is diversity. If the parallel models are highly correlated, the ensemble's error reduction is minimal. Ensuring diversity through techniques such as bootstrap sampling, feature randomness, or the use of fundamentally different model types (heterogeneous ensembles) is crucial for maximizing the benefits of parallelism.

2.3.3. Hierarchical and Multi-stage Architectures: Divide and Conquer

Hierarchical architectures address complexity by decomposing a problem into smaller, more manageable sub-problems, often organized in a tree-like or multi-layered structure. Specialized models are assigned to each sub-problem, and their outputs are progressively combined to form a global solution. This "divide and conquer" strategy is particularly effective for tasks with inherent hierarchical structure, such as image recognition (objects composed of parts) or natural language understanding (sentences composed of phrases).

A distinct form of hierarchy is the outer-inner loop architecture, common in meta-learning and Neural Architecture Search (NAS). In these systems, an outer optimization loop (e.g., an evolutionary algorithm) proposes candidate configurations or architectures, while an inner loop evaluates them by training a model. (Ding and Spector's, 2023) MEAS-PQC framework for quantum circuits exemplifies this pattern: the outer loop uses a multi-objective evolutionary algorithm to search for optimal Parameterized Quantum Circuit (PQC) architectures, and the inner loop evaluates each candidate's performance and noise resilience on quantum hardware or simulators. The outer and inner loops operate at different timescales and levels of abstraction, creating a clear hierarchy.

Similarly, in surrogate-assisted NAS, as discussed by (Eslami et al., 2022), a surrogate model sits in the middle of the hierarchy, predicting the performance of candidate architectures to avoid expensive full training. This creates a three-level hierarchy: the search strategy (outer loop), the surrogate predictor (middle layer), and the actual architecture evaluation (inner loop, used only for validation).

The primary advantage of hierarchical architectures is their scalability and ability to impose structure on complex problems. However, they come with increased design complexity. The decomposition into sub-problems must be meaningful, and the coordination between levels (e.g., how information flows from the inner loop to the outer loop) must be carefully managed to avoid bottlenecks or information loss.

2.3.4. Levels of Fusion: Early, Late, and Hybrid Integration

A concept closely related to architectural patterns is the level at which information from different components is fused. This fusion level significantly affects the system's flexibility, robustness, and susceptibility to error propagation.

Early fusion refers to the integration of data or features at the input level. In a multimodal system, for instance, image pixels and text embeddings might be concatenated before being fed into a single neural network. In sequential architectures, the output of the first module (a transformed representation of the raw data) serves as input to the next, thereby enabling an early form of feature fusion.

Late fusion combines the decisions or predictions of multiple models after they have been generated independently. This is the hallmark of parallel architectures, such as the voting mechanism in Random Forests or the meta-learner in stacking. Late fusion is generally more robust to errors in individual

models, as a mistake in one component does not directly corrupt the internal representations of others.

Hybrid fusion combines elements of both. A hierarchical system might fuse features across multiple levels or combine intermediate representations from one branch with the final decision of another branch. For example, a deep neural network for image classification fuses low-level features (edges) early on, mid-level features (shapes) in middle layers, and high-level features (object parts) in later layers, culminating in a final class decision. This is a form of hierarchical fusion intrinsic to deep learning itself.

Each architectural pattern: sequential, parallel, and hierarchical, offers distinct trade-offs in terms of modularity, error propagation, computational cost, and suitability for different problem types. Table 2.3 summarizes these key characteristics, providing a quick reference for selecting an appropriate architecture for a given hybrid task.

Table 2.3. Comparison of Hybrid Architectural Patterns

Feature	Sequential	Parallel	Hierarchical
Information Flow	Linear chain	Concurrent, independent	Tree-like, layered
Primary Advantage	Modularity, simplicity	Robustness (variance reduction)	Scalability, handling complexity
Primary Disadvantage	Error propagation	Computational cost, need for diversity	Design complexity
Fusion Level	Early (feature-level)	Late (decision-level)	Hybrid (multi-level)
Error Propagation	High (cascading)	Low (errors isolated)	Medium (depends on hierarchy)
Typical Use Case	Preprocessing + classifier pipelines (Bülbül & Işık, 2024)	Ensemble methods, heterogeneous model fusion (Gao et al., 2017; Terbuch et al., 2023)	NAS, meta-learning, divide-and-conquer (Ding & Spector, 2023; Eslami et al., 2022)

The architectural patterns discussed in this section, (sequential, parallel, and hierarchical) provide the structural blueprints for hybrid machine learning systems. They govern how components interact, how information flows, and how errors propagate. By understanding these patterns, a designer can make informed decisions about the overall organization of a hybrid system, choosing the pattern that best aligns with the problem's structure and the desired trade-offs between robustness, complexity, and computational cost.

However, an architecture is only a skeleton. To bring it to life, it must be coupled with effective optimization strategies that tune its components, search for optimal configurations, and guide the learning process. The next section delves into these strategies, exploring how metaheuristics, surrogate models, and hardware-aware optimization algorithms power the hybrid systems we have now classified and architected.

2.4. Optimization Strategies

The architectural patterns discussed in Section 2.3 (sequential, parallel, and hierarchical) provide the structural blueprints for hybrid machine learning systems. However, a blueprint alone does not constitute a building; it must be brought to life by the right construction machinery. In hybrid ML, this machinery is optimization. Optimization strategies are the engines that train model components, search for optimal architectures, tune hyperparameters, and enforce compliance with hardware and domain constraints. This section delves into the core optimization algorithms that power hybrid systems, examining how they navigate complex, often non-convex search spaces to find configurations that balance performance, efficiency, and robustness. We will explore four families of optimization strategies: Cross-Entropy methods, evolutionary algorithms and swarm intelligence, surrogate-assisted optimization, and hardware-aware techniques. By understanding these engines, we complete our picture of how hybrid systems are designed, built, and deployed.

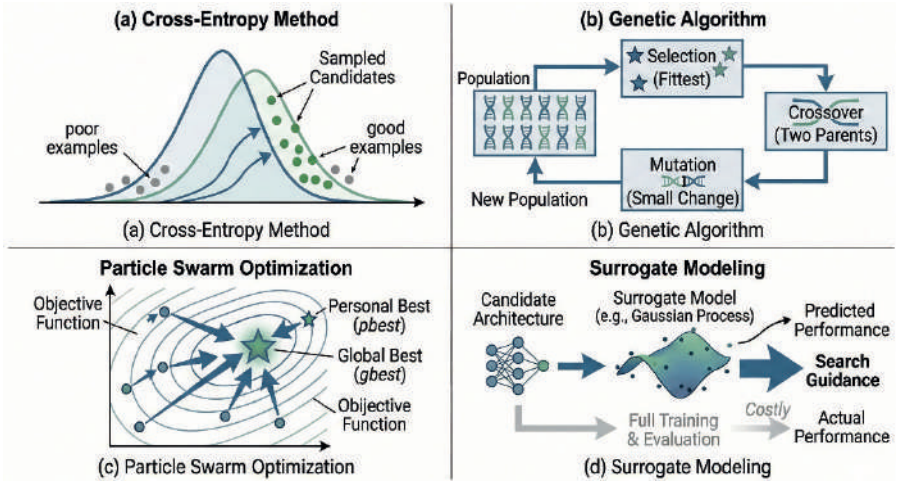


Figure 2.3. Optimization Strategies in Hybrid Machine Learning

2.4.1. Cross-Entropy Methods: Probabilistic Search in Discrete Spaces

The Cross-Entropy (CE) method, originally developed for rare-event simulation, has proven to be a powerful tool for combinatorial optimization, particularly in hybrid systems where decisions are discrete and the search space is vast. CE operates by iteratively sampling candidate solutions from a parameterized probability distribution, evaluating them, and then updating the distribution parameters to favor the generation of high-quality samples in subsequent iterations.

This "sample-evaluate-update" loop makes CE exceptionally well-suited for problems where gradient information is unavailable or meaningless, such as selecting optimal precoder configurations in wireless communications. Li et al. (2019) and Gao et al. (2017) employ CE-inspired methods to tackle the hybrid precoding problem in mmWave MIMO systems. Here, the goal is to choose a discrete set of precoders and combiners that maximize spectral efficiency under strict hardware constraints (e.g., low-resolution ADCs, limited RF chains). The CE method guides this search by maintaining a probability distribution over the discrete decision space, iteratively biasing it towards configurations that yield high spectral efficiency. This approach achieves near-optimal performance with a fraction of the computational cost of exhaustive search, demonstrating the power of probabilistic search in hybrid system design.

In our taxonomy, CE methods are a cornerstone of hardware-aware hybrids. Architecturally, they often function as an outer optimization loop, making them a natural fit for the hierarchical patterns discussed in Section 2.3.3, where an outer algorithm guides the configuration of an inner system.

2.4.2. Evolutionary Algorithms and Swarm Intelligence: Population-Based Metaheuristics

Drawing inspiration from natural evolution and collective behavior, population-based metaheuristics offer another powerful class of optimization strategies for hybrid ML. Unlike CE, which maintains a probability distribution, these methods maintain a population of candidate solutions and iteratively improve them through competition and cooperation.

Genetic Algorithms (GA) mimic the process of natural selection. A population of candidate solutions (e.g., neural network architectures or hyperparameter sets) is evolved over generations. Individuals are evaluated using a fitness function (e.g., validation accuracy). The fittest individuals are selected to breed a new generation through crossover (combining parts of two

solutions) and mutation (randomly altering a solution). Ding et al. (2011) demonstrate the use of GA for optimizing neural network weights, while Akl et al. (2019) extend this concept to optimize not only hyperparameter values but also their positions within a deep network, a problem with a combinatorial explosion of possibilities. Terbuch et al. (2023) also employ GA within their hybrid anomaly detection pipeline to fine-tune model hyperparameters for specific industrial contexts.

Particle Swarm Optimization (PSO) is inspired by the social behavior of bird flocks or fish schools. Each "particle" in the swarm represents a candidate solution and moves through the search space. Its movement is guided by its own best-known position (personal best) and the swarm's best-known position (global best). This simple yet effective mechanism allows the swarm to converge on optimal regions. Bülbül (2023) utilizes a variant of PSO—the Gray Wolf Optimizer—to tune a Multi-Layer Perceptron for medical diagnosis, demonstrating the algorithm's ability to navigate the hyperparameter space and improve predictive performance. Zarei and Dzalilov (2009) combine PSO with Simulated Annealing (SA) in a hybrid approach to optimize both the architecture and parameters of back-propagation neural networks.

Both GA and PSO are well-suited for the outer-loop optimization tasks in hierarchical architectures. They explore the high-level design space (architectures, hyperparameters), while inner loops (e.g., gradient descent) handle low-level weight learning. This division of labor is a quintessential example of hybrid optimization in action.

2.4.3. Surrogate Models and NAS Predictors: Accelerating Search with Approximate Evaluations

A significant challenge in many hybrid systems, particularly in Neural Architecture Search (NAS), is the astronomical cost of evaluating each candidate. Training a deep neural network from scratch to convergence can take days or even weeks, making a brute-force search over thousands of architectures infeasible. Surrogate models offer a powerful solution by acting as fast, approximate predictors of candidate performance, dramatically improving sample efficiency.

A surrogate model is itself a machine learning model (e.g., a regression model, a graph neural network) trained to predict the performance of an architecture given its representation. Once trained, the surrogate can be used within the optimization loop to rapidly screen thousands of candidates, and only the most promising ones are selected for full training and evaluation. This creates a three-tiered hierarchy: the search strategy (e.g., an evolutionary

algorithm) proposes candidates; the surrogate model predicts their performance; and only the top candidates proceed to the expensive inner-loop training for final validation.

Eslami et al. (2022) provide a comprehensive overview of this approach in their work on learning a unified latent space for NAS. They leverage structural and symbolic information about architectures to build powerful surrogate predictors. This allows the search process to explore a much wider range of designs than would be possible with direct evaluation alone, ultimately leading to the discovery of higher-performing architectures. The use of surrogates is a prime example of a hybrid strategy at the meta-level, combining search algorithms with predictive models to overcome a fundamental computational bottleneck.

2.4.4. Hardware-Aware Optimization: Navigating the Accuracy-Efficiency Trade-Off

In the era of edge computing and Internet of Things (IoT), a model's predictive accuracy is only one facet of its performance. Its inference latency, energy consumption, and memory footprint are equally critical for real-world deployment on resource-constrained devices. Hardware-aware optimization explicitly incorporates these physical and operational constraints into the search process, transforming a single-objective problem (maximizing accuracy) into a multi-objective one (maximizing accuracy, minimizing latency, minimizing energy).

The hybrid precoding work of Li et al. (2019) is a quintessential example of hardware-aware optimization. The optimization objective is not just spectral efficiency, but spectral efficiency achievable under the constraints of low-resolution ADCs and limited RF chains. The Cross-Entropy method they employ is guided by these hardware limits, ensuring that the generated precoder configurations are not only high-performing but also physically realizable.

In the quantum domain, Ding and Spector (2023) tackle an even more complex hardware-aware problem. Their MEAS-PQC framework performs multi-objective evolutionary architecture search for Parameterized Quantum Circuits (PQCs), optimizing for both task performance (e.g., accuracy in a reinforcement learning task) and robustness to quantum noise. Quantum noise is a fundamental hardware constraint of near-term devices, and by explicitly including it as an optimization objective, they ensure that the discovered circuits are not just theoretically sound but also practically viable.

Hardware-aware optimization often requires multi-objective versions of the algorithms discussed above, such as multi-objective GA (NSGA-II) or

multi-objective PSO. These algorithms maintain a population of solutions that represent different trade-offs along the Pareto frontier, providing the designer with a set of optimal choices given the specific deployment constraints.

Table 2.4. Comparison of Optimization Strategies in Hybrid ML

Strategy	Working Principle	Strengths	Weaknesses	Typical Use	Related Taxonomy	Related Architecture
Cross-Entropy (CE)	Probabilistic sampling, distribution update	Effective in discrete spaces, simple	May struggle in continuous spaces	Hybrid pre-coding, feature selection	Hardware-Aware	Hierarchical (outer loop)
Evolutionary Algorithm (GA)	Population, selection, crossover, mutation	Global search, gradient-free, flexible	Slow convergence, many parameters	NAS, hyperparameter optimization	NAS & Hyperparameter	Hierarchical (outer loop)
Particle Swarm (PSO)	Population-based search, movement based on personal/ swarm best	Fast convergence, simple implementation	Risk of getting stuck in local minima	Weight optimization, hyperparameter tuning	NAS & Hyperparameter	Hierarchical (outer loop)
Surrogate Models	Fast proxy model that predicts performance	Sample efficiency, cost reduction	Dependent on surrogate model accuracy	NAS, architecture search	NAS & Hyperparameter	Hierarchical (intermediate layer)
Hardware-Aware Optimization	Multi-objective optimization (accuracy + constraints)	Realistic, deployable solutions	Complex objective function	Edge AI, quantum circuit design	Hardware-Aware, Quantum-Classical	Parallel/ Hierarchical

Optimization is the engine that powers hybrid machine learning. From the probabilistic search of Cross-Entropy methods to the population-based evolution of GAs and PSO, from the efficiency gains of surrogate models to the realism of hardware-aware constraints, these strategies provide the means to navigate the complex, high-dimensional spaces in which hybrid systems operate. Together with the taxonomies of Section 2.2 and the architectural patterns of Section 2.3, they form a comprehensive toolkit for understanding, designing, and deploying hybrid models. With this foundation firmly established, the following chapter will demonstrate these concepts in action, presenting detailed case studies from healthcare, communications, and industrial monitoring that illustrate how theoretical principles translate into practical success.

3. Case Studies in Engineering and Healthcare

In previous sections, we examined why hybrid machine learning is necessary (NFL, complexity), how it works (bias-variance), what it combines (taxonomy), where it combines (architectural patterns), and which engine

it runs on (optimization). Now, we will bring all these concepts together through four key case studies selected from the literature.

3.1. Case 1: Survival Prediction in Healthcare (Bülbül & Işık, 2024)

Predicting survival after a heart attack is a challenging problem due to the high-dimensionality, noise, and limited sample size of medical data. Classical classifiers (e.g., Softmax) either overfit to such data or fail to capture complex patterns (high bias). Furthermore, medical datasets are often unbalanced; the number of surviving patients may exceed or fall short of the number of non-surviving patients. These challenges limit the chances of success for a single model.

The hybrid model proposed by Bulbul and Isik (2024) follows a three-stage sequential architecture:

1) *Stacked AutoEncoder (SAE):* Encodes the high-dimensional raw data into a lower-dimensional and noise-free representation space (latent space).
2) *Particle Swarm Optimization (PSO):* Optimizes the hyperparameters of a classifier (Softmax) that will operate on the representations generated by SAE.
3) *Softmax Classifier:* Using the optimized hyperparameters, generates the final prediction (survival or non-survival) using the SAE output.

This case falls under the "Data-Driven and Physics/Medical-Informed Hybrids" category in our taxonomy. SAE is a data-driven model for representation learning. PSO, on the other hand, is used as an optimization algorithm to improve the classifier's performance. Although it doesn't directly use medical rules (ontology), the representation learned by SAE can be interpreted as an implicit prior knowledge of the structure of medical data.

The model has a sequential architecture. The information flow is linear:

Raw data → SAE → PSO-optimized hyperparameters → Softmax → Prediction.

The critical point is that the SAE's output serves as input to the PSO and Softmax. In the event of error propagation, a corruption in the SAE's representation will affect all subsequent steps. Therefore, proper SAE training is vital.

Optimization occurs at two levels:

1) *Inner loop:* SAE and Softmax are trained using traditional gradient descent (backpropagation).
2) *Outer loop:* PSO optimizes the hyperparameters (e.g., learning rate, regularization coefficient) of the Softmax classifier. This is an example of hierarchical optimization (meta-heuristic in the outer loop, gradient-based learning in the inner loop). PSO's population-based structure allows it to perform a global search in the hyperparameter space, discovering configurations that cannot be found with manual tuning or grid search.

The study by Bulbul and Isik (2024) demonstrates that the hybrid model achieves significantly higher accuracy in survival prediction compared to a standalone Softmax classifier or a standard machine learning pipeline. This case is valuable in showing how sequential architectures and population-based optimization can be successfully applied in high-risk areas such as healthcare. The main lesson learned: In complex, high-dimensional, and noisy data, combining representation learning with hyperparameter optimization can deliver performance levels unattainable by single models.

3.2. Case 2: Hybrid Pre-coding in mmWave Communication (Li et al., 2019; Gao et al., 2017)

Complex precoding techniques are necessary to achieve high spectral efficiency in millimeter-wave (mmWave) multiple-input multiple-output (MIMO) systems. However, due to hardware limitations (low-resolution ADCs, limited RF chains), fully digital precoding is impractical. Hybrid precoding aims to overcome these limitations by combining analog and digital components. However, finding the optimal combination of analog and digital preencoders requires navigating a discrete and extremely large search space. Exhaustive search is computationally impossible, and pure machine learning models are insufficient to account for the hardware limitations.

Li et al. (2019) and Gao et al. (2017) proposed a hybrid approach based on Cross-Entropy (CE) to solve this problem. The CE method evaluates preencoder candidates sampled from a probabilistic distribution, selects the best performing ones, and updates the distribution toward these best candidates. This process iteratively converges toward configurations that provide high spectral efficiency while respecting hardware constraints.

This case is a prototypical example of the "Hardware-Aware and Wireless System Hybrids" category in our taxonomy. It involves an optimization problem where machine learning (CE's probabilistic model) and hardware constraints (ADC resolution, RF chain count) are intertwined. The goal is

not only to maximize spectral efficiency but also to do so under physical constraints.

The model can be thought of as a hierarchical outer-loop/inner-loop architecture:

1) *Outer loop:* The CE algorithm navigates the search space by updating the probability distribution.
2) *Inner loop:* Each sampled candidate preencoder configuration is evaluated with a channel capacity calculation function (or simulation). While this inner loop is not as costly as training a machine learning model, it still introduces a computational overhead. Thanks to CE's efficiency, the number of these evaluations is minimized.

The optimization strategy is the Cross-Entropy (CE) method. In this context, CE operates as a gradient-free optimization algorithm. Its effectiveness in a discrete decision space (which preencoder to choose) and its probabilistic nature strike a good balance between exploration and exploitation. Furthermore, the method provides hardware-aware optimization by directly incorporating hardware constraints into the search space (e.g., allowing only specific ADC configurations).

Li and Gao's work demonstrates that the CE-based hybrid approach reduces computational complexity several-fold while achieving spectral efficiency close to that of the full search method. This case proves how effective probabilistic search methods are compared to pure optimization or pure learning methods, especially for discrete optimization problems under hardware constraints. The main lesson learned: Integrating real-world constraints (physics, hardware) into the optimization process produces solutions that are not only theoretically optimal but also practically feasible.

3.3. Case 3: Anomaly Detection in Industrial Time Series (Terbuch et al., 2023)

In industrial systems (e.g., power plants, production lines), anomaly detection is critical for predicting equipment failures and ensuring safety. Multivariate time-series data from these systems are often high-dimensional, noisy, and non-stationary. Furthermore, the boundaries between normal operating conditions (nominal behavior) and abnormal states can be blurred. A single model (e.g., only an autoencoder or only a rule-based system) is insufficient to simultaneously detect both expected (rule-based) and unexpected (data-driven) anomalies.

The hybrid pipeline proposed by Terbuch et al. (2023) combines three different information sources:

1) *KPI (Key Performance Indicator) Guided Modules:* Expert-defined rules or indicators that reflect the physical information and operational limits of the system.
2) *Data-Driven Neural Models:* Models, such as autoencoders, that learn normal patterns in the data and detect deviations.
3) *Hyperparameter Optimization with Genetic Algorithm (GA):* A meta-heuristic algorithm that optimizes both the threshold values of the KPI modules and the hyperparameters of the neural models according to a specific industrial context.

This case falls under the "Hybrid Time-Series Anomaly Detection and Fault Diagnosis" category in our taxonomy. It is also closely related to the "Data-Driven and Physics/Medical-Informed Hybrids" category because it combines domain information (KPIs) with data-driven learning (neural models). The use of GA also connects it to the NAS & Hyperparameter Hybrids category. This case is a rich example demonstrating how multiple hybrid categories can overlap.

The model exhibits a hybrid architecture:

- KPI modules and neural models operate in parallel; both process the same input (raw time series) and generate their own anomaly scores.
- These scores are then combined in a fusion layer to create a final anomaly decision (late fusion).
- GA, on the other hand, operates outside this parallel structure as a hierarchical outer loop, optimizing both the KPI thresholds and the hyperparameters of the neural models.

The optimization strategy is based on a Genetic Algorithm (GA). The GA performs a population-based search without any gradient information about the search space (KPI thresholds, neural model hyperparameters). Each candidate solution (a set of thresholds and hyperparameters) is evaluated (fitness) based on the anomaly detection performance of the hybrid model on a validation dataset. The population is evolved with selection, crossover, and mutation operators. This demonstrates how population-based meta-heuristic optimization can be used to simultaneously tune multiple components of a hybrid system.

The study by Terbuch et al. (2023) shows that combining KPI-driven rules with data-driven models significantly improves anomaly detection performance compared to using either approach alone. GA-based optimization provides flexibility by enabling this combined system to adapt to different industrial contexts. The main lesson learned: In complex industrial problems, parallel architectures that combine domain knowledge (rules) with data-driven learning (neural models) offer a more robust and comprehensive solution to both expected and unexpected anomalies.

3.4. Case 4: Searching for a Multipurpose Evolutionary Architecture for Quantum Circuits (Ding & Spector, 2023)

Parameterized Quantum Circuits (PQCs) are a promising approach to harnessing the potential of quantum computers. However, designing an effective PQC architecture is far more challenging than searching for architectures for classical neural networks. Quantum circuits are subject to unique hardware constraints such as quantum noise, decoherence, and a limited number of qubits. The performance of a PQC should be measured not only by how well it learns the task but also by how noise-tolerant it is on the current quantum hardware. This renders traditional NAS methods, which optimize for a single objective (e.g., accuracy alone), insufficient.

Ding and Spector (2023) proposed a framework, MEAS-PQC (Multi-Objective Evolutionary Architecture Search for Parameterized Quantum Circuits), to address this challenge. MEAS-PQC searches for PQC architectures using a multi-objective evolutionary algorithm. The objective functions include not only task performance (e.g., success in a classification or reinforcement learning task) but also the circuit's robustness to quantum noise. In this way, quantum circuits that are both high-performance and noise-tolerant, i.e., "useful," are discovered.

This case is one of the most current examples of the "Quantum-Classical Hybrid Learning Systems" category in our taxonomy. It also has strong links to the NAS & Hyperparameter Hybrids category due to its use of evolutionary algorithms, and to the Hardware-Aware Hybrids category due to its hardware-awareness (quantum noise). This case demonstrates how advanced hybrid systems can encompass multiple taxonomic categories.

MEAS-PQC has a clear hierarchical outer-loop/inner-loop architecture:

1) *Outer loop:* A multi-objective evolutionary algorithm (e.g., NSGA-II) manages a population of PQC architectures. Each individual (architecture) represents a set of quantum gates and their connections.

2) *Inner loop:* Each candidate PQC architecture is evaluated on a quantum simulator or real quantum hardware. This evaluation calculates both task performance (e.g., accuracy achieved at the end of a training cycle) and performance under a noise model (robustness).

The outer loop generates new generations of candidates based on the results of these multi-objective evaluations.

The optimization strategy is based on a multi-objective evolutionary algorithm (MOEA). This approach identifies multiple optimal solutions on the Pareto frontier rather than a single optimal solution. Each point on the Pareto frontier represents a different trade-off between two objectives (performance and noise stability). This gives quantum researchers the flexibility to choose the most suitable circuit based on the noise level of the hardware at their disposal and the importance of the task. MOEAs are an ideal choice for such problems because they are gradient-free, population-based, and multi-objective.

Ding and Spector's (2023) MEAS-PQC study emphasizes the importance of multi-objective optimization in the field of quantum machine learning. While conventional single-objective NAS methods may produce high-performance yet noise-fragile circuits, the circuits discovered by MEAS-PQC can operate more reliably on noisy real-world quantum processors. The main lesson derived is that in computational paradigms with novel and unique constraints, such as quantum computing, multi-objective, hardware-aware, and evolutionary optimization strategies provide far more valuable and practical solutions than purely performance-oriented approaches.

The four case studies presented in this section (ranging from healthcare and communications to industrial monitoring and quantum computing) demonstrate the power and versatility of hybrid machine learning. Each case tackles a unique set of challenges by combining different learning paradigms, architectural patterns, and optimization strategies. Table 3.1 summarizes these case studies within the unified framework developed throughout this book, linking them to the taxonomy, architectural patterns, and optimization strategies discussed in Chapter 2. This synthesis underscores that while the application domains differ widely, the underlying principles of hybridization remain consistent.

Table 3.1. Summary of Case Studies

Case	Domain	Core Challenge	Taxonomy Category	Architectural Pattern	Optimization Strategy
Bülbül & Işık (2024)	Healthcare (Medical Prediction)	High dimensionality, noise, limited data	Data-Driven & Medical-Informed	Sequential	PSO (hyperparameter optimization) + Gradient Descent (SAE training)
Li et al. (2019); Gao et al. (2017)	Communicati-ons (mmWave MIMO)	Discrete decision space, hardware constraints	Hardware-Aware	Hierarchical (outer-loop CE)	Cross-Entropy (CE)
Terbuch et al. (2023)	Industrial Anomaly Detection	Rule-based + data-driven fusion, context adaptation	Anomaly Detection + Domain-Informed	Parallel (KPI + Neural) + Hierarchical (GA outer loop)	Genetic Algorithm (GA)
Ding & Spector (2023)	Quantum Computing	Quantum noise, multi-objective trade-off	Quantum-Classical + NAS + Hardware-Aware	Hierarchical (outer-loop MOEA)	Multi-Objective Evolutionary Algorithm (MOEA)

These case studies illuminate the journey from theoretical principles to practical solutions. They show that the "No Free Lunch" theorem is not a dead end, but a starting point for creative, problem-aware design. They demonstrate that the increasing complexity of real-world data demands architectures that go beyond a single model. They reveal that managing bias and variance, integrating diverse components, and navigating constrained search spaces are not abstract exercises, but essential skills for building systems that work in the real world. With this practical understanding in hand, the final chapter of this book looks to the future, exploring emerging trends and open challenges in the rapidly evolving field of hybrid machine learning.

4. Conclusion and Future Directions

Throughout this chapter, we have embarked on a comprehensive journey through the landscape of hybrid machine learning. Starting from the fundamental limitations imposed by the No Free Lunch theorem, we dissected the increasing complexity of real-world data and the consequent violations of the assumptions of singular models. We then built a unified framework encompassing taxonomies, architectural patterns, and optimization strategies, grounding each concept in concrete examples from the literature. Finally, we witnessed this framework in action through four diverse case studies. This concluding chapter synthesizes the key insights from our exploration, outlines

the emerging trends shaping the future of hybrid ML, and identifies the open challenges that await researchers and practitioners in this dynamic field.

4.1. Synthesis of Key Findings

The central thesis of this work, rooted in the No Free Lunch theorem (Wolpert & Macready, 1997), is that the pursuit of a single, universally superior model is a futile endeavor. Success in machine learning is not an inherent property of an algorithm but a measure of its compatibility with the specific structure and constraints of a given problem. The increasing complexity of modern data—its high dimensionality, heterogeneity, non-stationarity, and the ubiquity of physical constraints—systematically violates the core assumptions (IID, fixed inductive biases) upon which singular models rely, rendering them inadequate in practice (see Section 1.2).

Hybrid machine learning emerges not merely as a performance-enhancing trick but as a theoretically grounded necessity to address this inadequacy. By integrating multiple learning paradigms, hybrid systems transcend the classical bias-variance trade-off. As demonstrated in Section 2.1, ensemble methods like bagging and boosting, and metaheuristic-driven architecture searches allow for the simultaneous management of bias and variance, a feat unattainable by any singular model. Our proposed taxonomy (Section 2.2) provides a structured way to navigate the vast space of hybrid designs, categorizing them by what they integrate (e.g., learning with optimization, data with domain knowledge), why they integrate it, and where in the pipeline this integration occurs.

Furthermore, the architectural blueprints and optimization strategies examined in Sections 2.3 and 2.4 reveal that the how is just as critical as the what. Whether through sequential pipelines (Bülbül & Işık, 2024), parallel ensembles (Gao et al., 2017), hierarchical outer-inner loops (Ding & Spector, 2023), or the use of gradient-free optimizers like Cross-Entropy and Genetic Algorithms (Li et al., 2019; Terbuch et al., 2023), the success of a hybrid system hinges on a thoughtful design that aligns its structure with the problem's demands. The case studies in Chapter 3 vividly illustrate this principle, showing how different combinations of these elements lead to robust, efficient, and deployable solutions across healthcare, communications, industrial monitoring, and quantum computing.

4.2. Emerging Trends and Open Challenges

The field of hybrid machine learning is far from static; it is continuously shaped by new computational paradigms, societal needs, and theoretical advancements. Several key trends are poised to define its future trajectory.

Just as AutoML aims to automate the design of single models, the next frontier is the automated discovery of hybrid systems. The taxonomy presented in this book could serve as a foundational ontology for such systems, in which a meta-learner could analyze a problem's characteristics and propose not just a single algorithm but an optimal combination of paradigms, an architectural pattern, and an optimization strategy. The primary challenge lies in the astronomical size of this hybrid design space, necessitating advances in meta-learning and efficient surrogate-assisted search (Eslami et al., 2022).

LLMs are evolving from text generators into general-purpose reasoning engines. A powerful hybrid trend involves using LLMs as controllers or orchestrators that decompose complex user queries into subtasks, invoke specialized tools (e.g., symbolic solvers, physical simulators, traditional ML models), and synthesize the results. This neuro-symbolic approach (Section 1.1) promises to combine the flexible understanding of LLMs with the precision and verifiability of classical systems. Open challenges include mitigating LLM hallucinations and ensuring the reliability of the overall hybrid pipeline.

Federated learning (FL) inherently faces a hybrid challenge: balancing a global model with the diverse, non-IID data distributions across local clients. Future FL systems will likely be hybrid themselves, employing personalized local models that share knowledge through meta-learning or transfer learning while maintaining privacy. Architecturally, this points to sophisticated parallel and hierarchical designs in which client-level and server-level optimization loops interact.

As ML permeates high-stakes domains, the demand for interpretable models grows. Hybrid systems offer a promising path forward by, for instance, using a symbolic rule-based system to provide explanations for a neural network's decisions, or by incorporating uncertainty-aware methods such as evidential learning (Hu et al., 2019). The key challenge is to design these systems so that the explanatory component is both faithful to the underlying model and comprehensible to the end-user, without excessively sacrificing predictive performance.

The environmental impact of large-scale ML training and deployment is a growing concern. This reinforces the need for hardware-aware optimization (Li et al., 2019) and multi-objective approaches that explicitly trade off

accuracy against energy consumption and latency (Ding & Spector, 2023). Future research will focus on developing even more efficient optimization algorithms and architectural patterns that push the Pareto frontier of the accuracy-efficiency trade-off, making hybrid ML a key enabler of green and sustainable artificial intelligence.

4.3. Final Remarks

The journey through hybrid machine learning reveals a fundamental truth: in a world of infinite problem complexity, the quest for a single, all-powerful algorithm is a mirage. The real power lies not in any one method, but in the thoughtful and creative synthesis of multiple paradigms. Hybrid ML is not merely a collection of techniques; it is a mindset—an approach that views each problem as a unique puzzle, to be solved by assembling the right combination of tools, architectures, and optimization strategies.

As we look to the future, the boundaries between learning, reasoning, and optimization will continue to blur. The integration of data-driven models with symbolic knowledge, of classical algorithms with quantum components, and of global objectives with local, hardware-aware constraints will define the next generation of intelligent systems. We hope this book serves as both a foundational reference and an inspiration for researchers and practitioners to explore this rich and rewarding landscape, building hybrid solutions that are not only more powerful, but also more robust, interpretable, and aligned with the complex needs of our world.

References

Akl, A., El-Henawy, I., Salah, A., & Li, K. (2019). Optimizing deep neural networks hyperparameter positions and values. Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology, 37(5), 6665–6681. https://doi.org/10.3233/JIFS-190033.

Badran, M. F., Md Sahar, N., Sari, S., & Taujuddin, N. S. A. M. (2020). Intrusion-detection system based on hybrid models: Review paper. IOP Conference Series: Materials Science and Engineering, 917(1), 012059. https://doi.org/10.1088/1757-899X/917/1/012059.

Bülbül, M. A. (2023). Urinary bladder inflammation prediction with the Gray Wolf Optimization algorithm and multi-layer perceptron-based hybrid architecture. Bitlis Eren Üniversitesi Fen Bilimleri Dergisi, 12(4), 1185–1194. https://doi.org/10.17798/bitlisfen.1360049.

Bülbül, M. A., & Işık, M. F. (2024). Survival prediction of patients after heart attack and breast cancer surgery with a hybrid model built with particle swarm optimization, stacked autoencoders, and the softmax classifier. Biomimetics, 9(5), 304. https://doi.org/10.3390/biomimetics9050304.

Das, R. D., & Winter, S. (2016). Detecting urban transport modes using a hybrid knowledge-driven framework from GPS trajectory. ISPRS International Journal of Geo-Information, 5(11), 207. https://doi.org/10.3390/ijgi5110207.

Ding, L., & Spector, L. (2023). Multi-objective evolutionary architecture search for parameterized quantum circuits. Entropy, 25(1), 93. https://doi.org/10.3390/e25010093.

Ding, S., Xu, X., Zhu, H., Wang, J., & Jin, F. (2011). Studies on optimization algorithms for some artificial neural networks based on genetic algorithm (GA). Journal of Computers, 6(5), 939–946. https://doi.org/10.4304/jcp.6.5.939-946.

Eslami, S., Monsefi, R., & Akbari, M. (2022). Learning a unified latent space for NAS: Toward leveraging structural and symbolic information. IEEE Access, 10, 102945–102956. https://doi.org/10.1109/ACCESS.2022.3208591.

Gao, X., Dai, L., Sun, Y., Han, S., & Chih-Lin, I. (2017). Machine learning inspired energy-efficient hybrid precoding for mmWave massive MIMO systems. In 2017 IEEE International Conference on Communications (ICC) (pp. 1–6). IEEE. https://doi.org/10.1109/ICC.2017.7997065.

Hu, D., Dong, W., Lu, X., et al. (2019). Evidential MACE prediction of acute coronary syndrome using electronic health records. BMC Medical Informatics and Decision Making, 19(Suppl 2), 61. https://doi.org/10.1186/s12911-019-0754-7.

Kotary, J., Fioretto, F., Van Hentenryck, P., & Wilder, B. (2021). End-to-end constrained optimization learning: A survey. In Proceedings of the Thir-

tieth International Joint Conference on Artificial Intelligence (IJCAI-21) (pp. 4475–4482). International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2021/610.

Kumar, B. K., Bilgaiyan, S., & Mishra, B. S. P. (2023). Software effort estimation through ensembling of base models in machine learning using a voting estimator. International Journal of Advanced Computer Science and Applications, 14(2). https://doi.org/10.14569/IJACSA.2023.0140222.

Labbani Narsis, O., Dujardin, E., & Nicolle, C. (2023). Objective-driven modular and hybrid approach combining machine learning and ontology. In 2023 15th International Congress on Advanced Applied Informatics Winter (IIAI-AAI-Winter) (pp. 300–304). IEEE. https://doi.org/10.1109/IIAI-AAI-Winter61682.2023.00062.

Li, L., Ren, H., Li, X., Chen, W., & Han, Z. (2019). Machine learning-based spectrum efficiency hybrid precoding with lens array and low-resolution ADCs. IEEE Access. Advance online publication. https://doi.org/10.1109/ACCESS.2019.2937209.

Losada, I. B., & Terranova, N. (2024). Bridging pharmacology and neural networks: A deep dive into neural ordinary differential equations. CPT: Pharmacometrics & Systems Pharmacology, 13(8), 1289–1296. https://doi.org/10.1002/psp4.13149.

Terbuch, A., O'Leary, P., Khalilimotlaghkasmaei, N., et al. (2023). Detecting anomalous multivariate time-series via hybrid machine learning. IEEE Transactions on Instrumentation and Measurement, 72, Article 2503711. https://doi.org/10.1109/TIM.2023.3236354.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1(1), 67–82. https://doi.org/10.1109/4235.585893.

Zhou, W., & Xie, Z. (2025). Enhancing sealing performance predictions: A comprehensive study of XGBoost and polynomial regression models with advanced optimization techniques. Materials, 18(10), 2392. https://doi.org/10.3390/ma18102392.

Chapter 3

Theoretical Limits of Machine Learning under Noisy and Limited Data

Gökhan Halimoğlu[1]

Abstract

Machine learning has achieved remarkable success across many scientific and engineering domains. However, the increasing reliance on data driven methods has also raised important questions regarding the theoretical limits of learning systems operating under noisy and limited data conditions. While modern algorithms often demonstrate impressive predictive performance, their reliability depends strongly on the statistical assumptions and structural properties of the data used during training.

This chapter examines the fundamental limitations of machine learning when observations are affected by measurement noise, limited sample size, and violations of classical statistical assumptions. Particular attention is given to the physical and mathematical structure of noise, including electronic fluctuations, quantization effects, drift phenomena, and heavy tailed variability. These characteristics challenge simplified modeling assumptions frequently adopted in machine learning, such as independent and identically distributed observations or Gaussian noise structures.

The discussion further explores how noise and data scarcity influence generalization behavior. Highly flexible models may achieve low training error while memorizing noise rather than learning meaningful signal patterns. Such phenomena highlight the importance of understanding the interaction between model complexity, data availability, and uncertainty in the data generating process. In addition, the chapter analyzes potential risks associated with simulation based data augmentation and the generation of synthetic datasets. When the statistical structure of simulated data does not accurately reflect the underlying system, models may learn artifacts of the simulation process instead of genuine physical relationships.

1 Dr. Gökhan Halimoğlu, Istanbul Kultur University, Vocational School, Department of Air Conditioning and Refrigeration Technology, g.halimoglu@iku.edu.tr, 0000-0002-0419-7723.

https://doi.org/10.58830/ozgur.pub1236.c5004

Finally, the role of physical priors and domain knowledge in stabilizing learning systems is discussed. Incorporating structural constraints derived from known physical principles can reduce the effective hypothesis space and improve robustness in noisy environments. From this perspective, machine learning should be viewed not as a universal replacement for theoretical modeling, but as a complementary tool within a broader scientific methodology.

2.1. Why Do Limits Matter?

Machine learning has advanced rapidly over the past decade and has produced impressive results across many application domains. However, a large portion of the literature primarily focuses on model performance, while giving less systematic attention to limitations, assumptions, and failure conditions. This tendency creates an implicit success bias: published studies often emphasize high accuracy scores and improved metrics, whereas the conditions under which a model loses reliability receive comparatively limited discussion. In contrast, real world problems differ substantially from controlled benchmark datasets and idealized experimental settings.

In real systems, data are often limited. This limitation is not only related to sample size. Physical and operational constraints such as measurement duration, hardware capacity, energy budget, cost, and accessibility also shape the data generation process. In fields such as particle detection, biomedical sensing, industrial monitoring, or financial time series analysis, data are not abstract mathematical objects; they are physical outputs produced under specific conditions. Therefore, the assumption that "more data" is always available or feasible is often unrealistic in practice.

Similarly, in many machine learning models, noise is treated as an unwanted error term. Yet in numerous physical systems, noise is an inherent component of the measurement process. Thermal fluctuations, microscopic electronic behavior, quantization effects, environmental variations, and long term drift all contribute to the observed signal. In this sense, noise is not merely a disturbance to be removed, but part of the system's natural dynamics. Models that ignore the structural characteristics of noise may achieve strong statistical performance while producing physically misleading interpretations.

This raises a fundamental question: when does machine learning fail, and why? Increasing model complexity does not automatically guarantee better generalization. When data are limited, when distributional assumptions are violated, or when noise carries structural properties, a model may learn

variations instead of the underlying signal. At this point, the distinction between training performance and physical validity becomes critical.

The aim of this chapter is to examine the limits of machine learning within a systematic framework. By analyzing the mathematical and physical nature of noise, the statistical consequences of limited data, the violation of distributional assumptions, and the risks of simulation based approaches, we seek to highlight the need for more cautious, physically informed, and methodologically robust learning strategies.

2.1.1. Structural Origins of Success Bias in Machine Learning Research

As emphasized in the introductory discussion, there exists a noticeable imbalance between the visibility of performance gains and the visibility of model limitations in machine learning research. This imbalance does not arise solely from individual research choices; it is also shaped by the structural characteristics of scientific production and evaluation. Within academic publishing, novelty, measurable improvement, and statistical performance gains often serve as primary evaluation criteria. As a result, demonstrating superior metrics can become more central than systematically analyzing the conditions under which a model becomes unstable or unreliable.

Model comparisons are frequently conducted on widely accepted benchmark datasets. Over time, these datasets evolve into reference standards, and models are ranked within this fixed experimental framework. While such comparisons are useful, they assess model behavior under a specific distributional setting. The response of the same model to distributional shifts or alternative data generation mechanisms is often left unexplored. Consequently, performance may be interpreted as an intrinsic property of the model rather than a context dependent outcome.

A similar issue arises in hyperparameter optimization and architectural refinement. Small numerical improvements are commonly reported as evidence of methodological progress. However, it is not always clear whether these gains reflect genuine structural advancement or dataset specific variation. When data are limited, even modest differences in evaluation metrics may fall within the range of statistical uncertainty. Without careful analysis, such differences can be overstated.

Success bias also influences the visibility of negative results. Instances of failed generalization, unstable behavior, or sensitivity to minor perturbations are less frequently reported. This creates a gap between the controlled scenarios represented in the literature and the variability encountered in real

systems. Models that perform well under optimized conditions may behave unpredictably when confronted with constraints that were not present during development.

This gap becomes particularly significant in domains where data generation is governed by physical and operational constraints. Performance evaluation is often carried out under the implicit assumption that sufficient and stable data are available. In many real world systems, however, data are inherently constrained in both quantity and structure. Recognizing the structural roots of success bias therefore leads naturally to a more fundamental question: why is data in real systems almost always limited, and how do these limitations shape the learning process?

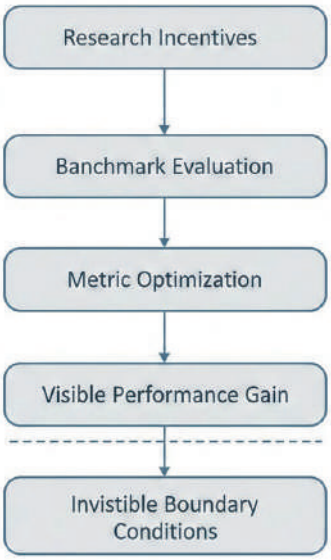


Figure 2.1. Structural Drivers of Success Bias in Machine Learning Research.

This conceptual diagram illustrates how structural elements within academic research and evaluation processes can collectively contribute to success bias. Rather than emerging solely from individual reporting choices, performance visibility is shaped by benchmark centered evaluation, metric optimization practices, and selective reporting mechanisms. The dashed boundary emphasizes the often overlooked conditions under which model reliability deteriorates, highlighting the gap between visible performance gains and unexamined boundary constraints.

2.1.2. Noise as a Physical Reality Rather Than an Undesired Error

In many machine learning formulations, the error term is treated as an abstract and idealized random component. The observed output is typically expressed as:

$$y = f(x) + \varepsilon$$

In this representation, ε is often assumed to be zero mean, independent, and drawn from a specific distribution, most commonly Gaussian. This assumption simplifies mathematical analysis and provides a convenient theoretical foundation for learning algorithms. However, in real systems, noise rarely follows such an idealized structure.

If one has worked with physical measurement systems, it becomes evident that noise is an intrinsic part of the measurement chain. Thermal fluctuations in electronic circuits, quantization limits in sensors, micro variations in environmental conditions, and long term drift effects are naturally embedded in the observed signal. Yet this phenomenon is not limited to physics based systems. In economic time series, market microstructure effects introduce inherent volatility. In financial data, liquidity driven jumps are part of the system's behavior. In biomedical measurements, inter individual physiological variability shapes the recorded signal. In social sciences, sampling uncertainty and contextual variability influence observed outcomes. These variations are often labeled as "error," but they are, in fact, structural features of the data generating process.

Noise, therefore, should not be interpreted as an external disturbance simply added to an otherwise clean system. It is frequently a direct consequence of the internal dynamics of the process under investigation. This distinction is not merely conceptual; it has methodological implications. When noise depends on measurement conditions, time, or system parameters, it cannot be adequately modeled as a simple independent random variable. In such cases, the error term may be more appropriately represented as:

$$\varepsilon = g(x, t, \theta)$$

indicating that noise may interact with observed variables, temporal structure, or intrinsic system parameters. Under these circumstances, the classical assumption of independent error becomes insufficient.

Recognizing noise as a physical or structural component of the system alters how learning outcomes should be interpreted. Methods that rely on simplified distributional assumptions may generate a misleading sense of robustness in environments where noise carries structure. A model may achieve low training error, yet still fail to capture the underlying system behavior in a physically meaningful way. This issue becomes particularly pronounced in systems characterized by temporal dependence, cross channel correlations, or resolution limits, where the boundary between signal and noise is not sharply defined.

Moreover, noise is not always devoid of information. Variations observed in measurement systems can encode indirect knowledge about system stability, environmental conditions, or operational regimes. Aggressive preprocessing steps designed to suppress noise may therefore remove patterns that are informative for modeling. The objective should not be the elimination of noise, but the accurate characterization of its structure.

Viewing noise as a structural property of the data generating mechanism, rather than as a purely unwanted perturbation, provides a more realistic foundation for modeling. Otherwise, learning algorithms risk optimizing idealized assumptions instead of representing the actual system. This perspective naturally leads to a deeper examination of the mathematical properties of noise and the limitations of common distributional assumptions, which will be discussed in the following section.

2.1.3. Structural Limits and Assumption Breakdown

The discussions presented in this section indicate that failures in machine learning systems are rarely accidental. Learning algorithms produce meaningful and reliable outcomes only under specific statistical and structural assumptions. When these assumptions weaken or are violated, performance metrics may cease to reflect the actual behavior of the model. In particular, when independence, distributional stability, or bounded variation assumptions break down, generalization capacity may deteriorate even if model complexity increases.

This effect becomes more pronounced in limited data regimes. Instead of capturing the underlying structure, the model may internalize incidental variation, widening the gap between apparent training success and physical interpretability. In such cases, failure does not arise from a purely technical

deficiency but from a misalignment between the model and the data generating process. The learning procedure remains confined within abstract statistical assumptions rather than representing the true dynamics of the system.

For this reason, understanding the limits of machine learning cannot be reduced to a discussion of algorithmic capacity alone. The decisive factor lies in the nature of variability observed in the data. Noise is frequently modeled as small and symmetric perturbations; however, in many physical and operational systems it exhibits structural properties, temporal dependence, or heavy tailed behavior. When these characteristics are ignored, the discrepancy between apparent predictive success and genuine system behavior may widen.

At this point, the discussion naturally leads to a more fundamental question: What is the mathematical and physical nature of noise? A substantial portion of the limitations observed in learning systems originates from the structural character of uncertainty within the data. The next chapter therefore examines noise in greater depth, addressing electronic noise mechanisms, quantization effects, drift and systematic bias, the implicit assumption of noise free modeling, the limitations of Gaussian approximations, the role of heavy tailed distributions, and the distinction between outliers and intrinsic noise. This technical analysis provides a deeper explanation for the failure mechanisms outlined above.

2.2. Mathematical and Physical Nature of Noise

The limitations discussed in the previous section ultimately converge on a common source: the structural character of uncertainty embedded in data. If machine learning systems fail under noisy and limited conditions, it becomes essential to examine the mathematical and physical foundations of that noise itself. Noise is not merely a statistical residual; it reflects measurable processes, resolution constraints, electronic fluctuations, discretization effects, and systematic biases inherent to real systems.

Importantly, such uncertainty does not arise from a single homogeneous mechanism. Different physical and operational processes generate distinct forms of variability, each carrying its own statistical structure and theoretical implications. Understanding the theoretical limits of learning therefore requires moving beyond abstract error terms and toward a structured analysis of these heterogeneous noise sources as physical and mathematical phenomena.

2.2.1.Electronic Noise

Electronic noise represents one of the most fundamental sources of uncertainty in physical measurement systems. It originates from microscopic

processes inherent to electronic components and cannot be eliminated entirely, only characterized and controlled.

Thermal noise, for instance, arises from the random motion of charge carriers within resistive elements. Its power spectral density is proportional to temperature and bandwidth, indicating that uncertainty is intrinsically linked to physical conditions rather than modeling imperfections. Similarly, shot noise emerges from the discrete nature of electric charge and becomes particularly relevant in low current or high sensitivity systems.

In many measurement environments, the observed signal can be expressed as:

$$y(t) = s(t) + \varepsilon_{el}(t)$$

Where $\varepsilon_{el}(t)$ reflects electronic fluctuations. While often approximated as Gaussian for analytical convenience, the distributional form depends on operating conditions, bandwidth constraints, and device architecture. Thus, electronic noise is not an abstract modeling artifact but a physically grounded stochastic process.

For machine learning systems trained on such measurements, ignoring the physical structure of electronic noise may lead to overconfidence in apparent signal patterns that partially originate from instrumentation effects.

2.2.2. Quantization Noise

Quantization noise arises from the discretization of continuous signals during analog to digital conversion. Every digital system imposes finite resolution, meaning that measured values are mapped onto discrete levels.

If x denotes a continuous signal and $Q(x)$ its quantized representation, the quantization error can be written as:

$$\varepsilon(q) = Q(x) - \chi$$

Under high resolution assumptions, this error is often modeled as uniformly distributed within a bounded interval. However, this approximation relies on specific regularity conditions. In low resolution systems or in signals with structured patterns, quantization effects may correlate with the signal itself, violating independence assumptions.

From a learning perspective, quantization introduces systematic discretization artifacts. Models trained on discretized data may inadvertently learn resolution induced patterns rather than intrinsic system behavior.

2.2.3. Drift and Systematic Bias

Unlike random fluctuations, drift reflects slow temporal variation in system parameters. It may originate from temperature changes, hardware aging, calibration shifts, or environmental transitions.

A simple representation may be written as:

$$y(t) = s(t) + \varepsilon(t) + d(t)$$

where $d(t)$ represents a slowly varying drift component.

Systematic bias differs from stochastic noise in that its expectation is non zero:

$$\text{Å}[\varepsilon] \neq 0$$

In such cases, the assumption of unbiased error collapses. For learning systems, drift and bias can produce misleading trends, causing models to interpret structural shifts as meaningful predictive signals.

2.2.4. The Implicit "Noise Free" Assumption in Modeling

Many learning formulations implicitly assume that noise is small, independent, and removable. Even when not stated explicitly, optimization objectives often treat the residual term as symmetric and statistically well behaved.

This "noise free" assumption becomes embedded in loss functions, regularization strategies, and validation procedures. However, when noise carries structure temporal dependence, cross channel correlation, or heavy tailed deviations such simplifications distort model evaluation.

The theoretical limits of learning are therefore partly determined by how closely the assumed noise model matches the physical reality of the data generating mechanism.

2.2.5. Why the Gaussian Assumption Often Fails

Gaussian noise assumptions are mathematically convenient due to their stability under linear transformations and their central role in classical statistical theory. Many generalization results rely implicitly on finite variance and concentration inequalities that hold under sub Gaussian conditions.

Yet empirical data in numerous domains exhibit skewness, kurtosis excess, or tail behavior inconsistent with Gaussian decay. In systems affected by rare but significant perturbations, the probability of extreme deviations may decay polynomially rather than exponentially.

When variance is unstable or dominated by rare events, classical concentration results weaken. Consequently, empirical risk minimization may not approximate expected risk as reliably as theory suggests.

2.2.6. Heavy Tailed Distributions

Heavy tailed distributions describe situations in which extreme deviations occur more frequently than predicted by Gaussian models. In contrast to exponential decay in normal distributions, heavy tailed behavior often follows polynomial decay of the form as:

$$P\left(|x| > x\right) \sim x^{-\alpha}, \alpha > 0$$

When the tail index $\alpha \leq 2$, the variance may be infinite; when $\alpha \leq 1$, even the mean may be undefined. Under such conditions, classical statistical assumptions finite variance, stable concentration around the mean, rapid convergence of sample averages no longer hold in their standard form.

In learning systems, heavy tailed variability can significantly alter optimization dynamics and generalization behavior. Rare but high magnitude deviations may dominate gradients, distort empirical risk estimates, or produce unstable parameter updates. Consequently, theoretical guarantees derived under sub Gaussian or bounded noise assumptions may substantially underestimate real world variability.

Heavy tails therefore do not merely represent statistical anomalies; they redefine the scale at which uncertainty operates.

2.2.7. Outliers Are Not Necessarily Noise

Extreme observations are often treated as undesirable perturbations and removed through filtering or trimming procedures. However, not every outlier is noise. An outlier may reflect:

- A rare but valid operating regime
- A transition between system states
- A structural anomaly
- The emergence of a new distributional phase

Automatically classifying extreme values as measurement error risks discarding meaningful information. In systems characterized by regime shifts, heavy tailed dynamics, or structural instability, extreme points may contain precisely the signals that reveal underlying constraints.

The distinction between intrinsic noise and structurally meaningful deviation is therefore not purely statistical; it is contextual and system dependent. Learning algorithms that indiscriminately suppress outliers may improve short term metrics while obscuring deeper structural behavior.stability, extreme points may contain precisely the signals that reveal underlying constraints.

2.2.8. Structural Implications for Learning Limits

The phenomena discussed throughout this section electronic fluctuations, quantization effects, drift, systematic bias, heavy tailed variability, and the ambiguous status of outliers collectively demonstrate that uncertainty in real systems is heterogeneous and structured. Noise does not arise from a single, uniform mechanism, nor does it consistently conform to classical assumptions of independence, Gaussian decay, or bounded variance.

When machine learning models rely on simplified noise formulations, theoretical guarantees regarding stability and generalization may rest on fragile premises. Mischaracterizing the structure of uncertainty does not merely affect parameter estimation; it alters the interpretation of performance metrics and the credibility of generalization bounds.

Within the broader theme of *Theoretical Limits of Machine Learning under Noisy and Limited Data*, these observations clarify a central point: the limits of learning are inseparable from the structure of noise. To model uncertainty inaccurately is to misinterpret the theoretical boundaries of machine learning itself.

2.3. Violation of the IID Assumption

Many theoretical guarantees in machine learning rely on a fundamental statistical assumption: that the observed data are independent and identically distributed (IID). Under this assumption, each observation is treated as a random sample drawn independently from the same underlying probability distribution. In other words, every data point is assumed to carry the same statistical characteristics and to be unaffected by the presence or order of other observations.

The IID framework provides a convenient foundation for theoretical analysis. Many classical results in statistical learning theory including convergence guarantees, risk bounds, and generalization theorems are derived under the assumption that training samples represent unbiased realizations of a stable data generating process. When this condition holds, empirical observations can be interpreted as reliable approximations of the underlying distribution, allowing models to infer patterns that extend beyond the observed dataset.

However, the IID assumption is fundamentally an idealization. In many practical contexts, the process that generates the data is influenced by physical mechanisms, environmental factors, or operational constraints. These influences introduce dependencies and structural variations that challenge the assumption that observations are independent realizations from a single stationary distribution.

As a result, the IID framework should not be interpreted as a universal description of real world data, but rather as a mathematical simplification that facilitates theoretical development.

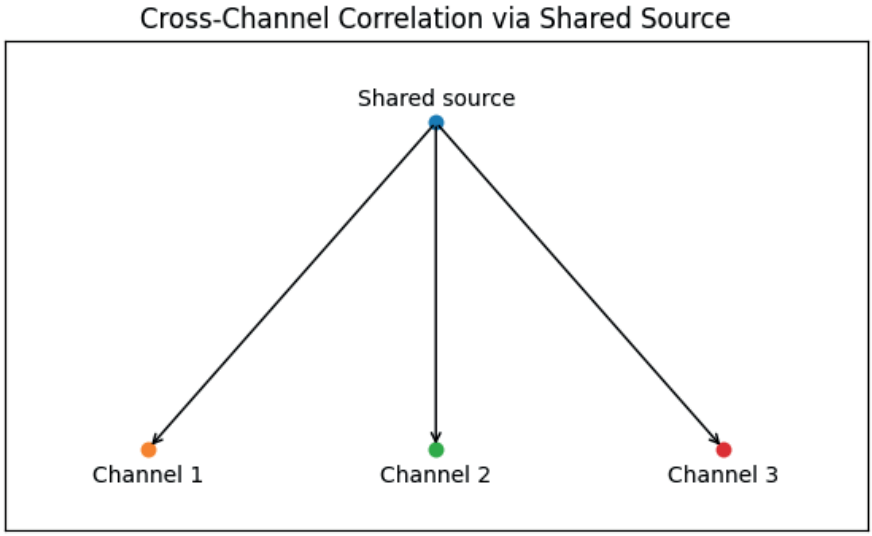


Figure 2.2. Illustration of cross channel correlation caused by a shared physical or environmental source. When multiple measurement channels respond to the same underlying process, their outputs become statistically dependent, violating the independence assumption frequently used in machine learning models.

2.3.1. Hidden Assumptions in Machine Learning Algorithms

Although the IID assumption is often explicitly stated in theoretical analyses, it also appears implicitly in many practical machine learning workflows. Training procedures, cross validation strategies, and performance evaluation metrics typically rely on the premise that the training and testing data originate from the same statistical distribution

For instance, common validation techniques assume that randomly partitioning a dataset into training and testing subsets provides an unbiased estimate of model performance. This assumption holds only when the underlying data are IID. If the data exhibit temporal structure, environmental drift, or hidden correlations, the random partitioning process may produce training and test sets that share similar dependencies. In such situations, the measured performance may overestimate the true predictive capability of the model.

Furthermore, many theoretical generalization guarantees rely on the idea that the empirical distribution observed during training approximates the true data distribution. When the IID assumption is violated, this approximation may break down. The learned model may perform well on the training data

yet fail to maintain similar performance when applied to data collected under slightly different conditions.

Consequently, the IID assumption plays a deeper role than simply simplifying mathematical derivations. It shapes how machine learning systems are evaluated, interpreted, and deployed.

2.3.2. Temporal Dependence

One of the most common sources of IID violation arises from temporal dependence. In many real world systems, data are collected sequentially over time rather than sampled independently from a static distribution. In such cases, successive observations may be influenced by the previous state of the system.

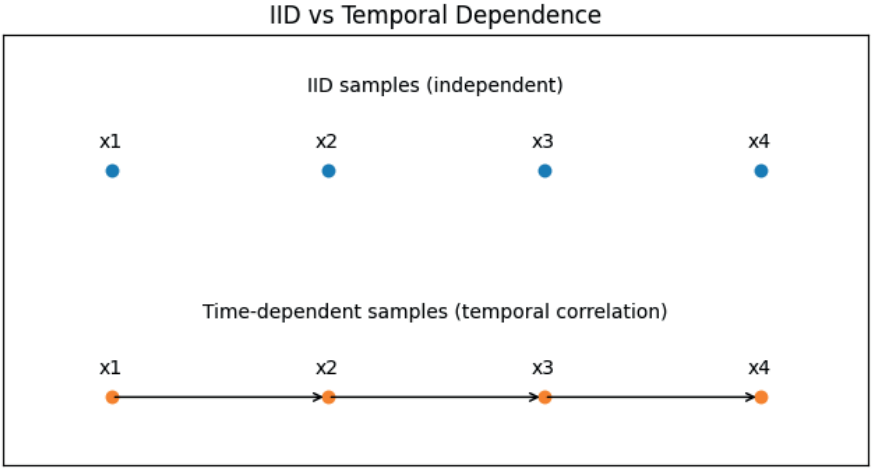


Figure 2.3. Illustration of the IID assumption versus temporally dependent observations. While many machine learning methods assume that samples are independent and identically distributed, real world measurements are often sequentially correlated, violating the independence assumption.

Time series data provide a clear example of this phenomenon. Measurements collected from sensors, financial markets, monitoring systems, or experimental apparatus often exhibit temporal correlations. These correlations may appear as trends, seasonal cycles, periodic oscillations, or slow drift in system behavior.

When temporal dependence is present, the assumption that each observation is independent of the previous one no longer holds. Instead, observations become linked through underlying system dynamics. The statistical properties

of the data may therefore evolve gradually over time, reflecting changes in system conditions or measurement environments.

For machine learning algorithms trained under the assumption of independence, temporal dependence can introduce significant challenges. Models may inadvertently learn patterns that are specific to a particular time interval rather than capturing stable relationships that persist across different periods.

2.3.3. Cross Channel Correlations

Another important source of IID violation emerges in systems where multiple measurement channels are recorded simultaneously. In complex observational environment such as sensor networks, particle detectors, industrial monitoring systems, or biomedical instrumentatio different channels often respond to the same underlying physical processes.

For example, in multi sensor measurement systems, environmental disturbances such as temperature changes or electromagnetic interference may affect multiple channels simultaneously, introducing correlations across observations.

Because these channels share common environmental influences or system dynamics, their outputs may become correlated. For example, a temperature fluctuation affecting a measurement device may simultaneously influence multiple sensors. Similarly, variations in power supply, mechanical vibrations, or electromagnetic interference may propagate across different components of an experimental setup.

When such cross channel dependencies exist, the assumption that individual observations are independent becomes questionable. Even if each channel produces a distinct signal, the underlying noise sources or environmental influences may introduce shared variability across measurements.

For machine learning models, ignoring these dependencies can lead to misleading conclusions. Apparent predictive relationships between variables may arise not from meaningful causal connections, but from shared background influences that simultaneously affect multiple channels.

2.3.4. Environmental Influences

Beyond temporal and cross channel dependencies, external environmental factors can also introduce significant deviations from the IID assumption. Real world data collection rarely occurs under perfectly controlled and unchanging conditions. Instead, measurements are often influenced by slowly varying

environmental parameters such as temperature fluctuations, humidity changes, hardware aging, calibration shifts, or operational adjustments.

These factors can gradually alter the statistical properties of the observed data. For example, sensor sensitivity may drift over time as components degrade, or measurement noise may increase due to environmental disturbances. In such cases, the distribution from which the data are drawn is no longer stationary.

Distributional shifts of this kind create additional challenges for machine learning models. A model trained under one set of environmental conditions may encounter systematically different data when deployed in another context. If the learning algorithm assumes that all observations originate from a single stable distribution, it may struggle to adapt to these changes.

Understanding the influence of environmental variability is therefore essential when evaluating the reliability of machine learning systems applied to real world data.

2.3.5. Summary

Taken together, these factors demonstrate that the IID assumption represents a theoretical idealization rather than a universal property of empirical data. Temporal dependencies, cross channel correlations, and environmental influences introduce structural relationships that violate the assumption of independent and identically distributed observations.

When machine learning models are trained under conditions where independence or distributional stability does not hold, the theoretical guarantees derived from statistical learning theory may become fragile. Performance estimates obtained under IID assumptions may no longer accurately reflect how the model will behave in realistic operational environments.

Recognizing and analyzing these deviations is therefore crucial for understanding the practical limits of machine learning. In noisy and data constrained systems, the reliability of learning algorithms depends not only on model architecture or optimization strategies, but also on how closely the underlying data generating process aligns with the assumptions upon which the learning framework is built.

2.4. Impact of Noise on Generalization

A central objective of machine learning is the ability to generalize beyond the data used during training. Models are typically evaluated using performance metrics measured on a training set, and improvements in these metrics are

often interpreted as evidence of successful learning. However, in the presence of noise, such interpretations may become misleading.

Training error measures how well a model fits the observed data, but it does not necessarily reflect how accurately the model captures the underlying structure of the system. When noise is present in the data, a learning algorithm may reduce training error by adapting to random fluctuations rather than identifying stable patterns. As a result, low training error does not always imply strong predictive performance on new observations.

Noise therefore introduces a fundamental limitation on generalization. Even when a model has sufficient expressive capacity, the information contained in noisy observations may be insufficient to reliably distinguish signal from random variation. In such cases, the effective amount of usable information in the dataset becomes smaller than the nominal sample size.

This limitation becomes particularly visible in the phenomenon known as *noise memorization*. Highly flexible models, including modern deep neural networks, are capable of fitting datasets even when labels contain substantial randomness. Empirical studies have demonstrated that deep networks can eventually achieve near perfect training accuracy even when the labels in the dataset are partially randomized. In such cases, the model is not learning meaningful structure but rather memorizing the idiosyncratic patterns present in the training data.

In such situations, increasing model complexity does not necessarily improve the model's ability to capture the underlying signal. Instead, the model may begin to adapt to random fluctuations present in the training data, effectively memorizing noise rather than learning the true structure of the system. This effect is illustrated conceptually in Figure 2.4.

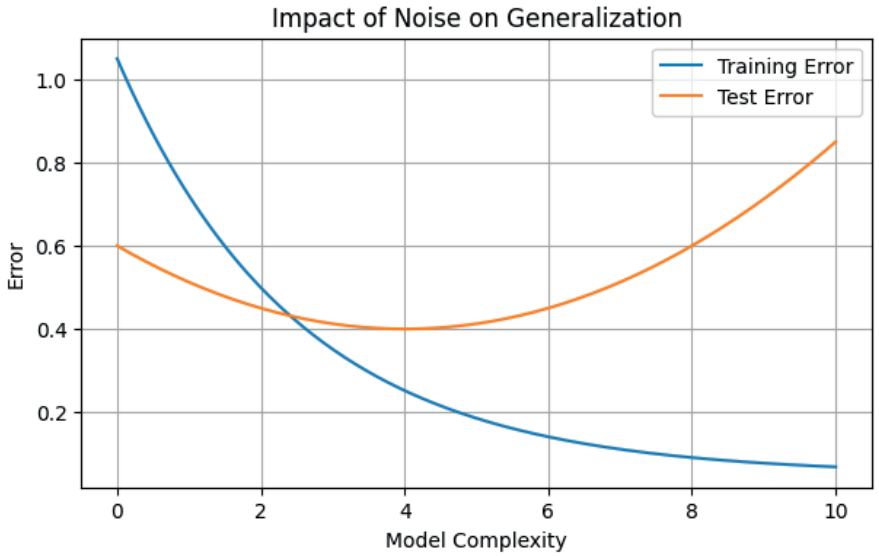


Figure 2.4. Conceptual illustration of the relationship between model complexity and generalization error in the presence of noise. While training error decreases as model capacity increases, test error may eventually rise when the model begins to memorize noise rather than capturing the underlying signal.

This behavior is often referred to as *noise fitting*. Instead of capturing the underlying data generating process, the model adapts to irregularities introduced by noise. While such models may appear highly accurate during training, their predictions can become unstable or unreliable when applied to new data.

Understanding the interaction between noise and generalization is therefore essential for interpreting model performance. In noisy environments, improvements in training accuracy must be evaluated cautiously, as they may reflect the model's ability to fit noise rather than its ability to learn the true structure of the system.

2.5. Theoretical Risks of Simulation and Data Augmentation

When real world data are limited, a common strategy is to generate additional samples through simulation or data augmentation techniques. The underlying idea is that artificially increasing the size of the training dataset can improve model performance and stabilize the learning process. However, this approach introduces its own theoretical challenges.

A key requirement for synthetic data generation is that the simulated samples preserve the statistical properties of the original data distribution. Methods such as Kernel Density Estimation (KDE) or Empirical Cumulative Distribution Functions (ECDF) attempt to approximate the observed distribution and generate additional samples consistent with that approximation. While such approaches can be useful under certain conditions, they rely on the assumption that the estimated distribution accurately reflects the true underlying process.

In practice, this assumption may be fragile. If the simulation procedure fails to reproduce the correct variance structure or dependency patterns present in the real data, the generated samples may introduce artificial regularities or distortions. In such situations, the model may learn features of the simulation process rather than characteristics of the original system.

Another potential issue arises when the generated data lack sufficient diversity. In generative modeling literature, similar phenomena are sometimes described as *mode collapse*, where simulated samples concentrate around limited regions of the distribution and fail to represent the full range of variability observed in the real data.

When these effects occur, the learning algorithm may become increasingly adapted to the properties of the synthetic dataset. Instead of improving generalization, the model may effectively learn the structure of the simulation procedure itself. This risk becomes particularly relevant when the original dataset is small, as the simulation model is then estimated from limited information.

Consequently, while simulation and data augmentation can be powerful tools, their use requires careful consideration of the assumptions underlying the generation process. Without preserving the true statistical and structural characteristics of the data, synthetic samples may inadvertently amplify the very limitations they are intended to mitigate.

2.6. Integrating Physical Priors into Learning

The discussions in the previous sections have emphasized that learning under noisy and limited data is fundamentally constrained by the structure of uncertainty in the observed system. When data are scarce and noise is present, the flexibility of machine learning models can become a source of instability rather than an advantage. In such situations, incorporating prior knowledge about the system can play a critical role in stabilizing the learning process.

In many scientific and engineering applications, the systems being modeled are governed by well established physical principles. Measurement processes,

conservation laws, and structural relationships often impose constraints on how variables can interact. However, standard machine learning approaches are typically designed to operate with minimal assumptions about the data generating mechanism, relying primarily on statistical patterns extracted from observations.

While this data driven flexibility can be useful in settings where little domain knowledge is available, it may also allow models to learn relationships that are inconsistent with the known behavior of the system. When training data are limited or noisy, the absence of structural constraints can lead the learning algorithm to interpret random fluctuations as meaningful patterns.

Introducing physical priors into the learning process provides a way to restrict the space of admissible solutions. Instead of allowing the model to explore all mathematically possible mappings between inputs and outputs, prior knowledge constrains the hypothesis space to configurations that remain consistent with the underlying physical system.

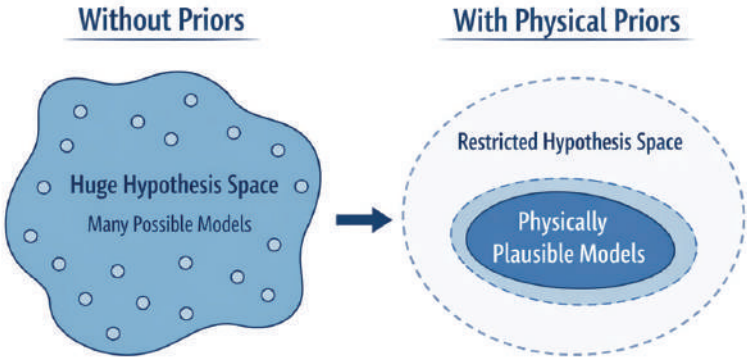


Figure 2.5. Conceptual illustration of the effect of physical priors on the hypothesis space of machine learning models. Without structural constraints, the learning algorithm can explore a very large set of possible mappings between inputs and outputs. Incorporating physical priors restricts this hypothesis space to solutions that remain consistent with the known structure of the system, thereby reducing the risk of fitting noise or spurious correlations.

This restriction acts as a form of regularization, reducing the tendency of the model to overfit noise or spurious correlations.

Another important consideration concerns the choice of input features. In purely data driven models, algorithms may exploit correlations that arise accidentally within the training dataset. Such correlations may appear predictive but often lack a meaningful causal or physical interpretation. As a result, models built on such features may perform well during training but fail to remain stable when conditions change.

Incorporating physically meaningful features can therefore improve the robustness of learning systems. When the model relies on variables that reflect genuine properties of the system, its predictions become less sensitive to incidental variations and measurement noise.

Physical knowledge can be integrated into machine learning models in several ways. Some constraints are hard constraints, meaning that the model must satisfy them exactly. Examples include conservation laws, symmetry relations, or structural dependencies that cannot be violated. Other constraints are soft constraints, which allow limited deviations but penalize violations during training. Such constraints are typically implemented through additional regularization terms in the loss function.

By introducing these forms of prior knowledge, the effective flexibility of the learning algorithm is reduced. Although this reduction may appear to limit the expressive capacity of the model, it often leads to more reliable learning in practice. When the hypothesis space is constrained by physically meaningful relationships, the model becomes less prone to fitting noise and more likely to capture stable patterns in the data.

From this perspective, incorporating physical priors should not be viewed as restricting machine learning models, but rather as guiding them toward physically plausible solutions. Especially in scientific and engineering contexts, combining data driven methods with domain knowledge can provide a more robust foundation for learning under noisy and limited data conditions.

2.7. Discussion: Where Should Machine Learning Stop?

The rapid success of machine learning across many domains has led to an increasing tendency to frame diverse problems as learning tasks. Advances in computational power, the availability of large datasets, and the impressive performance of modern algorithms have reinforced the perception that sufficiently complex models can extract meaningful patterns from almost any type of data. While this perspective has driven significant progress, it also

raises an important methodological question: where should the application of machine learning appropriately stop?

Not every analytical problem is naturally suited to a machine learning formulation.

This distinction can be illustrated conceptually in Figure 2.6.

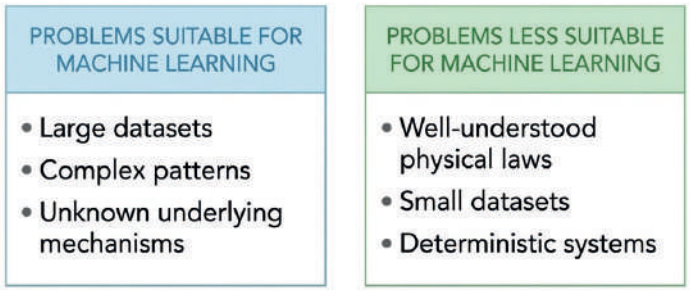


Figure 2.6. Conceptual illustration of problem types with respect to the suitability of machine learning methods. Machine learning is particularly effective in settings characterized by large datasets, complex patterns, and unknown mechanisms, whereas problems governed by well understood physical laws or limited data may be more appropriately addressed using theoretical or statistical approaches.

In some cases, the underlying mechanisms governing a system are already well understood through theoretical models, physical laws, or established statistical frameworks. When such knowledge exists, replacing these models with purely data driven approaches may introduce unnecessary complexity and reduce interpretability without providing meaningful gains in predictive capability.

A related misconception arises when machine learning is implicitly treated as a substitute for statistical reasoning. Although modern learning algorithms build upon statistical principles, they do not replace the need for careful modeling assumptions, uncertainty analysis, or hypothesis testing. Statistical thinking remains essential for interpreting model outputs, evaluating robustness, and understanding the limitations of empirical results.

These issues become particularly visible in the academic literature, where methodological pitfalls occasionally appear in studies that rely heavily on predictive performance metrics while paying less attention to the underlying assumptions of the learning process. High accuracy on benchmark datasets may not always translate into reliable understanding of the system being studied.

Benchmark performance therefore does not necessarily imply scientific validity. A model may perform well on a dataset while still failing to capture the causal mechanisms governing the underlying system. When noise, limited data, or distributional shifts are present, models can appear successful while capturing artifacts of the data rather than genuine structural relationships.

Recognizing these limitations does not diminish the value of machine learning. Instead, it highlights the importance of applying learning methods within an appropriate methodological framework. Machine learning is most powerful when used as a complementary tool one that operates alongside theoretical knowledge, statistical reasoning, and domain expertise.

From this perspective, the central challenge is not whether machine learning should be used, but how and where it should be integrated into the broader process of scientific and analytical inquiry. Careful consideration of model assumptions, data limitations, and domain knowledge is therefore essential for ensuring that machine learning contributes meaningfully to understanding complex systems rather than merely optimizing predictive performance.

References

Arpit, D., et al. (2017). A closer look at memorization in deep networks. ICML.

Belkin, M., et al. (2019). Reconciling modern machine learning and bias–variance trade-off. PNAS.

Breiman, L. (2001). Statistical modeling: The two cultures. Statistical Science.

Clauset, A., Shalizi, C., & Newman, M. (2009). Power law distributions in empirical data. SIAM Review, 51(4), 661–703.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

Karniadakis, G., et al. (2021). Physics informed machine learning. Nature Reviews Physics.

Kay, S. M. (1993). Fundamentals of statistical signal processing: Estimation theory. Prentice Hall.

Lipton, Z. C., & Steinhardt, J. (2019). Troubling trends in machine learning scholarship. Queue, 17(1), 45–77.

Middleton, D. (1999). Non-Gaussian noise models. IEEE Transactions on Information Theory, 45(4), 1129–1149.

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). Foundations of machine learning. MIT Press.

Papoulis, A., & Pillai, S. (2002). Probability, random variables and stochastic processes. McGraw-Hill.

Pineau, J., et al. (2021). Improving reproducibility in machine learning research. JMLR, 22(164), 1–20.

Raissi, M., Perdikaris, P., & Karniadakis, G. (2019). Physics informed neural networks. Journal of Computational Physics, 378.

Rudin, C. (2019). Stop explaining black box ML models. Nature Machine Intelligence.

Sculley, D., et al. (2018). Winner's curse? On pace, progress, and empirical rigor. ICLR Workshop.

Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding machine learning. Cambridge University Press.

Shorten, C., & Khoshgoftaar, T. (2019). A survey on image data augmentation. Journal of Big Data, 6(1).

Taleb, N. N. (2020). Statistical consequences of fat tails. STEM Academic Press.

Willard, J., et al. (2020). Integrating physics based modeling with ML. Nature Reviews Earth & Environment.

Zhang, C., et al. (2017). Understanding deep learning requires rethinking generalization. ICLR.

Chapter 4

Temporal and Homeostatic Mechanisms of Synaptic Plasticity[1]

Ibrahim Ozturk[2]

David M. Halliday[3]

Abstract

This chapter provides a comprehensive examination of the biologically plausible synaptic plasticity mechanisms fundamental to learning in Spiking Neural Networks (SNNs). Because SNNs, recognized as the third generation of neural network models, encode information through the precise temporal dynamics of discrete spikes, learning is intrinsically dependent upon the continuous modulation of synaptic weights and delays. We initially review the mechanics of chemical synaptic transmission and classical Hebbian plasticity, which are operationalized biologically through Long-Term Potentiation (LTP) and Long-Term Depression (LTD).

The analysis subsequently advances to Spike-Timing Dependent Plasticity (STDP), a critical unsupervised mechanism wherein the precise relative timing of pre-synaptic and post-synaptic action potentials dictates both the magnitude and direction of synaptic weight modifications. To mitigate the inherent risk of runaway network excitation driven by these localized STDP rules, the chapter elucidates the essential regulatory role of homeostatic plasticity. Specifically, it details how activity-dependent synaptic scaling globally adjusts neuronal excitability to maintain stable, target firing rates without disrupting the relative weight distributions established by STDP.

1 This chapter is based on the PhD thesis: 'Learning spatio-temporal spike train encodings with ReSuMe, DelReSuMe, and Reward-modulated Spike-timing Dependent Plasticity in Spiking Neural Networks,' by Ibrahim Ozturk, 2017, University of York. Copyright 2017 by I. Ozturk.

2 Faculty of Engineering and Natural Sciences, Department of Electrical and Electronics Engineering, Osmaniye Korkut Ata University, ibrahimozturk@osmaniye.edu.tr, 0000-0003-3149-0527. Faculty of Technology, Department of Electronics and Communication Engineering, Karadeniz Technical University, ibrahimozturk@osmaniye.edu.tr, 0000-0003-3149-0527.

3 Department of Electronic Engineering, University of York, david.halliday@york.ac.uk, 0000-0001-9957-0983.

https://doi.org/10.58830/ozgur.pub1236.c5005

Finally, the integration of dopamine-modulated plasticity is examined, illustrating how global neuromodulatory reward signals act as reward-prediction errors to consolidate local STDP-induced changes. This mechanism bridges the gap between unsupervised Hebbian timing rules and complex, goal-oriented reinforcement learning. Collectively, these temporal and homeostatic principles establish a robust theoretical foundation for the development of more advanced learning algorithms and supervised mapping techniques in artificial networks.

1. Introduction

Spiking Neural Networks (SNNs) rely on neuron models that closely mimic biological brains, allowing many natural neurobiological principles to be directly applied to their architecture. In SNNs, information is carried through connections between a sending (pre-synaptic) node and a receiving (post-synaptic) node. Each connection typically involves a synaptic weight (which determines the impact of the signal) and a synaptic delay (which dictates the travel time of the signal).

In biological terms, the process of "learning" is achieved by regulating these connections over time, which is a phenomenon known as synaptic plasticity. If an artificial network is intended to learn via supervised, unsupervised, or reinforcement strategies, its synapses must incorporate mechanisms to dynamically facilitate or depress these connection weights.

This chapter explores these fundamental, biologically plausible synaptic plasticity approaches, which will serve as the foundation for the advanced learning algorithms proposed later in the text. The chapter is organized into the following parts:

- **Section 1.2 (The Synapse):** Introduces the biological background of neuronal junctions (synapses) and explains their basic artificial interpretation and transmission models.
- **Section 1.3 (Hebbian Plasticity):** Discusses classical Hebbian learning, an unsupervised mechanism which dictates that correlated activation between neurons strengthens their connection. It also details the biological manifestations of this rule through Long-Term Potentiation (LTP) and Long-Term Depression (LTD).
- **Section 1.4 (Spike-Timing Dependent Plasticity - STDP):** Explores STDP, a crucial extension of Hebbian plasticity where the magnitude and direction of weight changes depend heavily on the *precise relative timing* of pre-synaptic and post-synaptic spikes. This section details STDP's mathematical models, implementations, and asymmetric/symmetric variants.

- **Section 1.5 (Homeostatic Plasticity):** Examines the mechanisms necessary to maintain stable neuronal functionality and prevent runaway network activity during learning. It details how neurons regulate their own excitability through synaptic scaling to keep firing rates within a target range.
- **Section 1.6 (Dopamine Modulated Plasticity):** Describes the biological background of neuromodulators, specifically dopamine, and how they consolidate the synaptic changes proposed by local STDP rules. This provides the biological inspiration for bridging unsupervised STDP with goal-oriented Reinforcement Learning.

2. Activity-Dependent Plasticity

Artificial Neural Networks have evolved significantly, with SNNs recognized as the third generation of neural network models. Unlike previous generations that use continuous analogue signals, SNNs use discrete "spikes" to carry information, mimicking the natural computation and biological plausibility of the brain. In these networks, information is processed and encoded in the precise timing between neuron firings (Gerstner & Kistler, 2002).

The fundamental building block of this communication is the synapse. A connection between a pre-synaptic (sending) node and a post-synaptic (receiving) node carries at least two dynamics: a synaptic weight, which determines the impact of the signal on the post-synaptic potential, and a synaptic delay, which dictates the travel time of the signal. In biological terms, "learning" is the process of regulating these synaptic efficacies, which is a phenomenon known as synaptic plasticity. If a network is intended to learn via supervised, unsupervised, or reinforcement strategies, the artificial synapses must include a mechanism for weight dynamics to facilitate or depress connections over time.

3. The Synapse and Synaptic Transmission

In the mammalian nervous system, synapses are specialized junctions, which are generally chemical rather than electrical contacts that allow information to flow from the axon terminals of one neuron to the dendrites of another (see Figure 1). This transmission occurs via the diffusion of neurotransmitters (such as acetylcholine) across the synaptic cleft, which activates ion channels in the target neuron as illustrated in Figure 1.

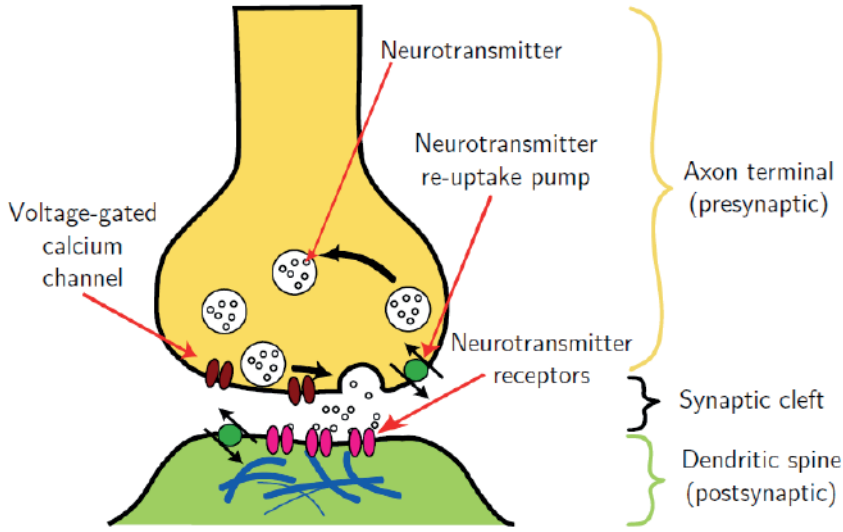


Figure 1. Biological structure of a chemical synapse, illustrating the axon terminal, neurotransmitters, and the synaptic cleft. Source: From "Learning spatio-temporal spike train encodings with ReSuMe, DelReSuMe, and Reward-modulated Spike-timing Dependent Plasticity in Spiking Neural Networks," by I. Ozturk, 2017, University of York. Copyright 2017 by I. Ozturk.

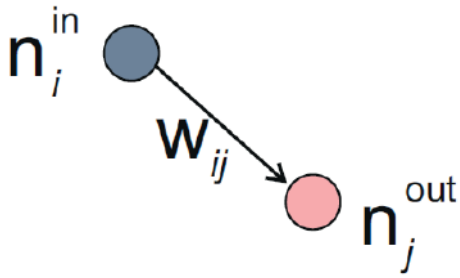


Figure 2. Artificial synapse model: bridging biological junctions to SNN abstractions via pre-synaptic/post-synaptic dynamics and synaptic weighting. Source: From "Learning spatio-temporal spike train encodings with ReSuMe, DelReSuMe, and Reward-modulated Spike-timing Dependent Plasticity in Spiking Neural Networks," by I. Ozturk, 2017, University of York. Copyright 2017 by I. Ozturk.

Depending on the synapse type, the post-synaptic potential (PSP) can be modified in two ways:

- **Excitatory Post-Synaptic Potential (EPSP):** Increases the electric polarization of the membrane, moving the neuron closer to its firing threshold.

- **Inhibitory Post-Synaptic Potential (IPSP):** Changes the charge negatively, lowering the membrane potential further from the firing threshold.

In SNN models, the input current $I_j(t)$ to a post-synaptic neuron j is commonly expressed as a weighted sum of pre-synaptic currents: $I_j(t) = \sum_{i=1}^{N_{\text{synapses}}} w_{ij}(t) \cdot f\left(t - t_f^i\right)$ where w_{ij} is the synaptic strength, t_f^i is the firing time of the pre-synaptic neuron, and f is a synaptic current function (often modeled as a Dirac delta or alpha function).

4. Hebbian Plasticity

In 1949, Donald O. Hebb proposed a groundbreaking unsupervised mechanism for synaptic plasticity. Hebb postulated that when an axon of cell A persistently takes part in firing cell B, a growth process or metabolic change occurs that increases A's efficiency in firing B (Hebb, 1949). In the context of artificial neural networks, simultaneous activation of pre- and post-synaptic cells increases the weight between them, while separate activation decreases it. The basic mathematical formulation for Hebbian learning is: $\Delta w_{ij} = \eta x_i x_j$ where Δw_{ij} is the change in synaptic weight, η is a small, positive learning rate, and x_i, x_j are the activations of the respective neurons.

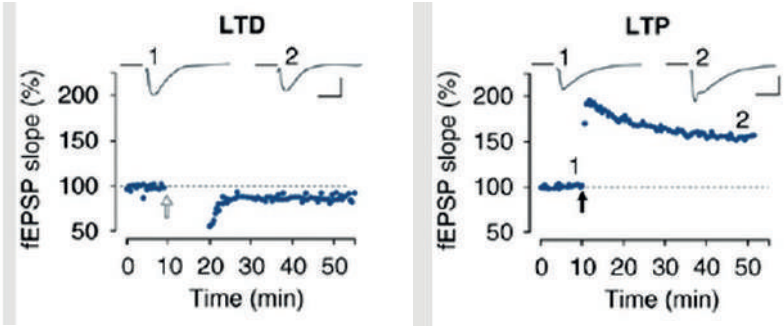


Figure 3. NMDAR-dependent synaptic plasticity in the hippocampal CA1 region. Sample experimental traces illustrating the induction of Long-Term Potentiation (LTP) and Long-Term Depression (LTD) as biological correlates for Hebbian and anti-Hebbian learning rules.Source: From "Learning spatio-temporal spike train encodings with ReSuMe, DelReSuMe, and Reward-modulated Spike-timing Dependent Plasticity in Spiking Neural Networks," by I. Ozturk, 2017, University of York. Copyright 2017 by I. Ozturk.

Decades later, biological experiments confirmed Long-Term Potentiation (LTP), a long-lasting increase in synaptic strength following rapid, successive stimulation (Bliss & Gardner-Medwin, 1973; Citri & Malenkas, 2008). Conversely, Long-Term Depression (LTD) was discovered as an activity-

dependent decrease in synaptic strength. Because some forms of LTD do not require pre-synaptic activity to be depressed, this reverse process is often defined as "anti-Hebbian" learning (Lynch et al., 1977) as sample experiments demonstrating these biological correlates for LTP and LTD are shown in Figure 3.

5. Spike-Timing Dependent Plasticity (STDP)

While traditional Hebbian plasticity relies heavily on rate-based coding, biological observations have demonstrated that the *precise relative firing times* of pre- and post-synaptic neurons dictate synaptic modification (Bi & Poo, 1998). This mechanism is known as Spike-Timing Dependent Plasticity (STDP).

There are two main shapes of the STDP form, illustrated as symmetric in Figure 4 and asymmetric in Figure 5. Symmetric STDP relies solely on the time difference between presynaptic and postsynaptic firing times, whereas asymmetric STDP also depends heavily on the specific temporal order of those firing times.

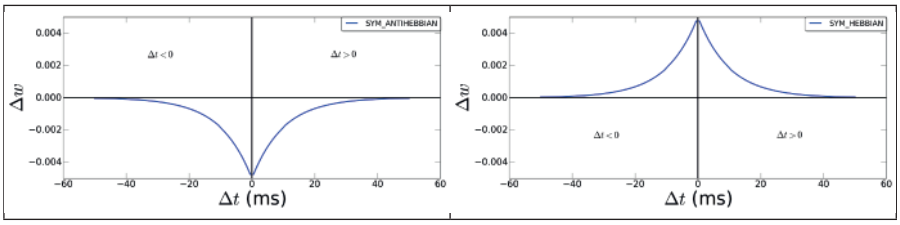


Figure 4. Symmetric-STDP. The left hand plot shows symmetric anti-Hebbian-STDP window. The right hand plot shows symmetric Hebbian-STDP window.Source: From "Learning spatio-temporal spike train encodings with ReSuMe, DelReSuMe, and Reward-modulated Spike-timing Dependent Plasticity in Spiking Neural Networks," by I. Ozturk, 2017, University of York. Copyright 2017 by I. Ozturk.

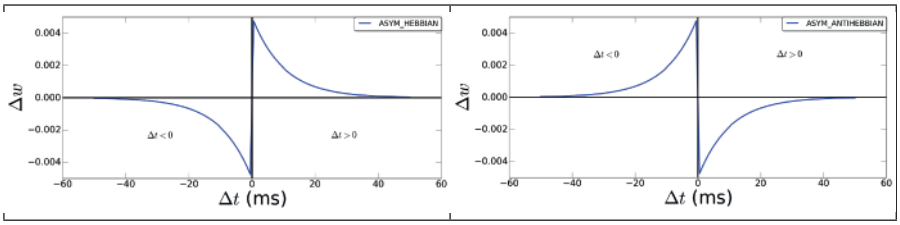


Figure 5. Asymmetric-STDP. Regardless of the order of pre- and post-synaptic firing times, STDP function W (Δt) modify the weights in the same manner for Δt < 0 and Δt > 0. The left hand plot shows asymmetric anti-Hebbian-STDP window. The right hand plot shows asymmetric Hebbian-STDP window.Source: From "Learning spatio-temporal spike train encodings with ReSuMe, DelReSuMe, and Reward-modulated Spike-timing Dependent Plasticity in Spiking Neural Networks," by I. Ozturk, 2017, University of York. Copyright 2017 by I. Ozturk.

a. The Asymmetric Learning Window

STDP operates within a critical temporal window, typically 20-25 ms (Song et al., 2000). The exact interval, defined as $\Delta t = t_j^{\text{post}} - t_i^{\text{pre}}$, determines the magnitude and direction of the weight change.

- **Potentiation (LTP):** If a post-synaptic neuron fires shortly after a pre-synaptic spike ($\Delta t \geq 0$), the synapse is strengthened.
- **Depression (LTD):** If the pre-synaptic spike arrives after the post-synaptic neuron has fired ($\Delta t < 0$), the synapse is weakened.

The standard STDP modification function, $W\left(\Delta t\right)$, is phenomenologically modeled using exponential decay constants (τ_{pre} and τ_{post}):

$$W\left(\Delta t\right) = \begin{cases} +F_{\text{pre}}\left(w\right) exp\left(-\Delta t / \tau_{\text{pre}}\right) \text{ if } \Delta t \geq 0 \\ -F_{\text{post}}\left(w\right) exp\left(+\Delta t / \tau_{\text{post}}\right) \text{ if } \Delta t < 0 \end{cases}$$

where $F_{pre}\left(w\right)$ and $F_{post}\left(w\right)$ control the amplitude of the learning window based on the current weight (Gerstner & Kistler, 2002).

b. Weight Dependence

The dependence of synaptic changes on the current weight (w_{ij}) is modulated using a non-negative exponent μ.

- **Additive STDP ($\mu = 0$):** Modifies weights independently of their current strength. In this model, $F_{pre}\left(w_{ij}\right) = A_{pre}$ and $F_{post}\left(w_{ij}\right) = A_{post}$.
- **Multiplicative STDP ($\mu = 1$):** Scales the weight change linearly based on the current weight, creating different equilibrium dynamics. Additive STDP is most commonly utilized for reliable network training.

c. Implementation via Multi-Spike Interactions

Simulating STDP by directly summing all pairs of spikes mathematically is computationally inefficient and physiologically implausible, as a biological neuron cannot remember all its individual firing times. Instead, models implement continuous pre-synaptic and post-synaptic traces (a_{pre} and a_{post}). Each spike leaves a trace that decays exponentially over time (Masquelier et al., 2008).

- When a pre-synaptic spike arrives, a_{pre} is incremented by A_{pre}, and the weight is decreased based on the current value of the post-synaptic trace a_{post}.

- When a post-synaptic spike arrives, a_{post} is incremented by A_{post}, and the weight is increased based on the current value of the pre-synaptic trace a_{pre}.

d. Weight Bounds

To prevent runaway excitation, where highly active neurons cause their synaptic weights to increase uncontrollably, synaptic weights must be bounded. Hard boundaries are applied to clip weights so they cannot exceed a maximum value (w_{max}) or fall below a minimum value (w_{min}).

e. Homeostatic Plasticity: The Balancing Act

While STDP drives learning through local timing mechanisms, the network must maintain stable overall functionality and firing rates. To achieve this, neocortical circuits employ homeostatic plasticity, an activity-dependent mechanism where neurons detect changes in their own firing rates and regulate their own excitability (Turrigiano, 2008).

6. Synaptic Scaling

Synaptic scaling globally scales a neuron's synaptic strengths multiplicatively to maintain target neuronal firing rates without destroying the relative weight distribution established by STDP. A simplified mathematical representation of this is:

$$\frac{dw_{ij}(t)}{dt} = \beta w_{ij}(t)\left[N_{\text{des}} - N_{\text{act}}\right]$$

where β is a scaling factor, N_{des} is the desired number of post-synaptic spikes, and N_{act} is the actual number of spikes (van Rossum et al., 2000).

However, continuous up-and-down scaling can cause instability, conflicting with the primary STDP learning mechanism. To solve this, scaling is often restricted to a permissible range ($N_{\text{min}}^{\text{des}} < N_{\text{act}} < N_{\text{max}}^{\text{des}}$):

$$\frac{dw_{ij}(t)}{dt} = \begin{cases} \beta w_{ij}(t)\left[N_{\text{max}}^{\text{des}} - N_{\text{act}}\right] & \text{if } N_{\text{act}} > N_{\text{max}}^{\text{des}} \\ \beta w_{ij}(t)\left[N_{\text{min}}^{\text{des}} - N_{\text{act}}\right] & \text{if } N_{\text{act}} < N_{\text{min}}^{\text{des}} \end{cases}$$

If the neuronal activity remains within this range, homeostatic scaling is inactive, and the local STDP mechanism is entirely responsible for learning. If

activity falls outside the boundaries, the scaling mechanism acts as a regularizer, adjusting all of a neuron's synapses globally to restore stability.

7. Dopamine-Modulated Plasticity

Beyond local (STDP) and global (Homeostatic) mechanisms, learning in biological systems is heavily influenced by neuromodulators like dopamine, which is closely associated with reward-driven reinforcement learning. Experimental evidence shows that subcortical dopamine signals act as a reward-prediction error (Schultz, 1998), functionally similar to Temporal Difference (TD) learning. By using a neuromodulator to consolidate the synaptic changes proposed by STDP as illustrated by the schematic of the reward-modulated STDP learning rule in Figure 6, networks can be trained through a hybrid semi-supervised approach, bridging the gap between unsupervised Hebbian mechanisms and complex, goal-oriented behavioral learning (Fremaux et al., 2010; Ozturk & Halliday, 2016). Practical applications of these reward-driven algorithms extend beyond theoretical models to training agents in specific environmental challenges, such as spatial navigation. For instance, utilizing knowledge-based reinforcement learning in a maze task demonstrates how reward mechanisms can significantly optimize an agent's trial-and-error exploration to successfully reach a target destination (Ozturk & Halliday, 2014).

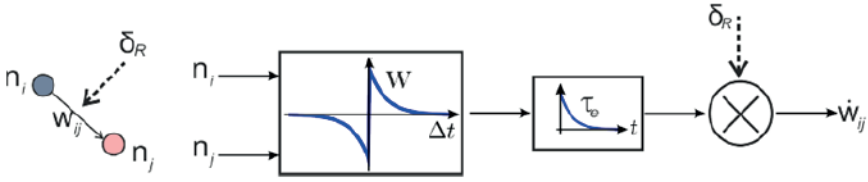


Figure 6. Schematic of the reward-modulated STDP learning rule. Block diagram illustrating the interaction between the local STDP coincidence window and a global Temporal Difference (TD) reward signal (δ) in governing synaptic weight modulation (Δw).Source: From "Learning spatio-temporal spike train encodings with ReSuMe, DelReSuMe, and Reward-modulated Spike-timing Dependent Plasticity in Spiking Neural Networks," by I. Ozturk, 2017, University of York. Copyright 2017 by I. Ozturk.

8. Conclusion

This chapter has presented the fundamental mechanisms of synaptic inputs and transmission, emphasizing that learning in Spiking Neural Networks (SNNs) is a highly complex process because information is intrinsically encoded in the time domain. To understand how artificial networks can mimic this

natural computation, various synaptic plasticities were detailed alongside their biological backgrounds.

We initially explored classical Hebbian plasticity before advancing to Spike-Timing Dependent Plasticity. STDP captures the essence of unsupervised learning in the nervous system by relying on the precise relative timing of pre-synaptic and post-synaptic spikes, and its additive form serves as the foundational learning method for training SNNs.

Beyond local STDP rules, the chapter discussed homeostatic plasticity, specifically synaptic activity-dependent scaling which is as a necessary regulatory process. This mechanism plays a crucial role in maintaining an optimal and stable level of firing activity across the network, ensuring that neurons do not become over-activated or entirely silenced during the learning process. Finally, the introduction of dopamine-modulated plasticity established a biological basis for linking these local, unsupervised timing mechanisms with global, reward-driven reinforcement signals.

Together, these temporal and homeostatic principles provide a robust, biologically plausible foundation. These concepts will serve as the core building blocks for developing the more advanced Reinforcement Learning algorithms and supervised mapping techniques explored in subsequent chapters.

References

Bi, G. Q., & Poo, M. M. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. Journal of Neuroscience, 18(24), 10464–10472.

Bliss, T. V., & Gardner-Medwin, A. R. (1973). Long-lasting potentiation of synaptic transmission in the dentate area of the unanaesthetized rabbit following stimulation of the perforant path. Journal of Physiology, 232(2), 357–374.

Citri, A., & Malenkas, R. C. (2008). Synaptic plasticity: Multiple forms, functions, and mechanisms. Neuropsychopharmacology, 33, 18–41.

Fremaux, N., Sprekeler, H., & Gerstner, W. (2010). Functional requirements for reward-modulated spike-timing-dependent plasticity. The Journal of Neuroscience, 30(40), 13326–13337.

Gerstner, W., & Kistler, W. M. (2002). Spiking neuron models: Single neurons, populations, plasticity. Cambridge University Press.

Hebb, D. O. (1949). The organization of behavior: A neuropsychological theory. Wiley and Sons.

Lynch, G. S., Dunwiddie, T., & Gribkoff, V. (1977). Heterosynaptic depression: A postsynaptic correlate of long-term potentiation. Nature, 266(5604), 737–739.

Masquelier, T., Guyonneau, R., & Thorpe, S. J. (2008). Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. PLoS ONE, 3(1), Article e1377.

Ozturk, I., & Halliday, D. M. (2014). Knowledge based reinforcement learning in maze navigation task [Poster presentation]. YDS 2014.

Ozturk, I., & Halliday, D. M. (2016, December). Mapping spatio-temporally encoded patterns by reward-modulated STDP in spiking neurons. In Proceedings of the International Conference on Computational Intelligence (IEEE SSCI 2016). Athens, Greece.

Schultz, W. (1998). Predictive reward signal of dopamine neurons. Journal of Neurophysiology, 80(1), 1–27.

Song, S., Miller, K., & Abbott, L. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. Nature Neuroscience, 3(9), 919–926.

Turrigiano, G. G. (2008). The self-tuning neuron: Synaptic scaling of excitatory synapses. Cell, 135(3), 422–435.

Van Rossum, M. C., Bi, G. Q., & Turrigiano, G. G. (2000). Stable Hebbian learning from spike timing-dependent plasticity. Journal of Neuroscience, 20(23), 8812–8821.

Chapter 5

Key Applications of Machine Learning in Music Data

Nigar Tuğbagül Altan Gülgün[1]

Abstract

Music has held an important place in human life throughout history and has continued to exist as a significant form of expression in cultural, social, and artistic contexts. With the advancement of technology, the ways in which music is produced, distributed, and consumed have undergone substantial transformation. In particular, with the widespread adoption of digital technologies, music listening platforms have evolved, and a vast number of musical works have become accessible in digital environments. In addition, the use of digital tools and software in music production processes has increasingly expanded.

The large volume of music data available in digital environments has made processes such as analyzing and classifying these data, discovering new music, and developing recommendation systems based on listener preferences increasingly important. In this context, machine learning methods play a significant role in the processing and interpretation of music data. Especially on digital music platforms where Western music is predominantly represented, numerous studies in the literature focus on topics such as the identification of musical genres, the analysis of song popularity levels, and the automatic classification of musical works.

In this section, the structure and representation methods of music data are examined from the perspective of machine learning (ML). Furthermore, recent research and example studies on the application of machine learning techniques in music analysis are reviewed. In this regard, the section aims to provide guidance for both researchers and practitioners by presenting a theoretical conceptual framework as well as practical examples.

1 Asst. Prof. Dr. Nigar Tuğbagül Altan Gülgün, Istanbul Okan University, Faculty of Engineering and Natural Sciences, Department of Computer Engineering, tugbagul.altan@okan.edu.tr, 0000-0002-1597-7231.

https://doi.org/10.58830/ozgur.pub1236.c5006

In conclusion, this section serves as a concise resource that provides both theoretical knowledge and practical guidance for researchers and practitioners aiming to achieve specific research objectives using music data. By demonstrating how machine learning methods can be effectively applied in music research and applications, it also establishes a fundamental reference framework for future studies.

1. Introduction

This chapter explores the application of machine learning techniques to music data. It begins with an overview of historical development of machine learning (ML) and what music is. In the subsequent sections, existing music datasets, the characteristics of music data, and several studies are discussed. The chapter then examines existing music datasets, the inherent characteristics of musical data, and selected studies from the literature that have applied machine learning methods to various music-related tasks.

Research on processing and analyzing music data using machine learning continues to advance rapidly, driven by the growing availability of digital music resources and computational tools. In this context, the chapter is intended to serve as a comprehensive guide for researchers and practitioners seeking to apply machine learning techniques in the field of music analysis.

1.1. Born of ML and ML

In the 1940s, Warren McCulloch and Walter Pitts presented some of the earliest foundational work on neural networks. Their research introduced key system components, including linear threshold decision elements, and provided a logical formalization for representing various forms of behavior and memory. In 1947, they further described a network architecture framework associated with geometric transformations, contributing to the conceptual development of computational models of neural activity (Minsky and Papert,1969).

Frank Rosenblatt, widely regarded as a pioneer of early perceptron research, formulated three fundamental questions concerning information processing in biological systems. These questions addressed: (1) how information about the physical world is sensed or detected by a biological system; (2) in what form this information is stored or remembered; and (3) how information retained in storage or memory influences recognition and behavior. These questions were presented at the Cornell Aeronautical Laboratory in 1958 (Rosenblatt, 1958). Rosenblatt also illustrated the organization of a perceptron, which is represented in Figure 1.

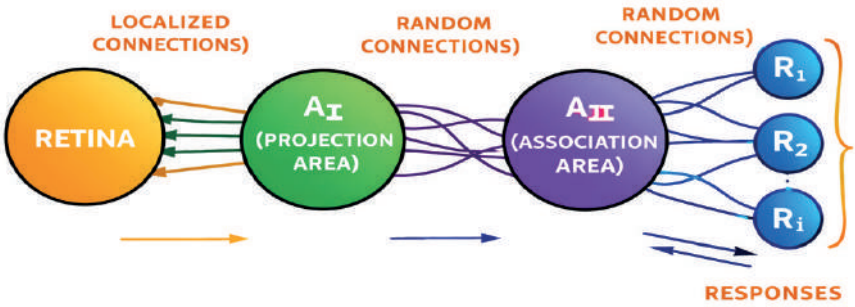


Figure 1. The organization of a typical photoperceptron

Figure 1 illustrates the organization of a typical photoperceptron—a perceptron designed to respond to optical patterns as stimuli—proposed by Frank Rosenblatt. In this model, Rosenblatt explained the interaction among units connected within a simple perceptron architecture.

In 1962, Rosenblatt further developed and demonstrated several theoretical foundations of the machines he referred to as perceptrons. However, the perceptron—considered the simplest form of a learning machine—was later critically analyzed by Marvin Minsky and Seymour Papert in 1969 (Minsky and Papert,1969). Their work emphasized the limitations of single-layer perceptrons and discussed aspects of parallel computation. They also presented a diagram illustrating their analysis, which is shown in Figure 2 to represent a fundamental concept of parallel computation.

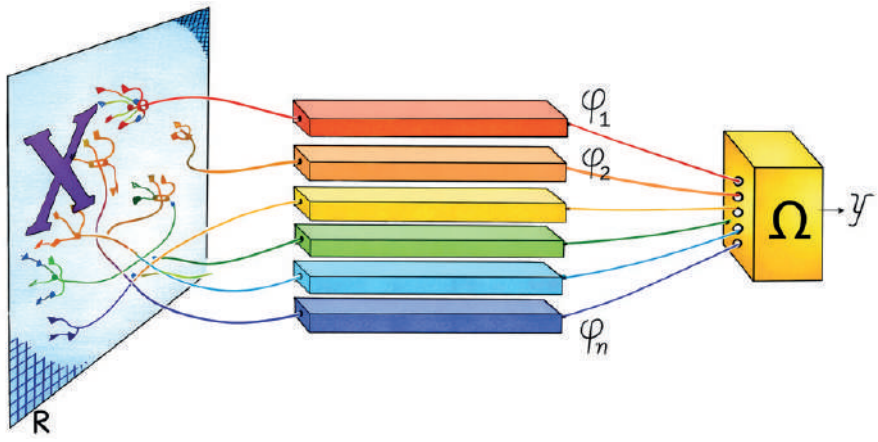


Figure 2. The simplest concept of parallel computation.

First, the set of functions $\varphi_1(X)$, $\varphi_2(X)$,..., $\varphi_n(X)$ is computed independently. The resulting values are then combine through a function Ω, which takes n arguments, to produce the final value of ψ.

Between 1950 and 1980, the development of artificial intelligence shifted direction due to limitations in computational power and data storage capacity. During the 1990s and early 2000s, the field increasingly adopted data-driven modeling approaches grounded in strong mathematical principles, particularly statistical learning theory, convex optimization, and feature engineering. Between 2012 and 2017, deep learning emerged as a highly influential paradigm, replacing manual feature engineering with end-to-end learning architectures capable of automatically extracting hierarchical representations from data. Since 2017, many research institutions and organizations have focused on developing models based on self-supervised learning, scaling laws, and multimodal architectures.

Today, machine learning (ML) methods are generally categorized into several primary learning paradigms: supervised learning, unsupervised learning, semi-supervised learning, self-supervised learning, and reinforcement learning. In supervised learning, algorithms are trained using labeled datasets. In contrast, unsupervised learning aims to discover patterns or structures in data without labeled outputs. Semi-supervised learning combines both labeled and unlabeled data to improve model performance. Reinforcement learning enables systems to learn optimal behaviors through interaction with an environment and feedback in the form of rewards or penalties. Self-supervised learning leverages inherent structures within unlabeled data to generate supervisory signals for model training. The major categories of ML are illustrated in Figure 3.

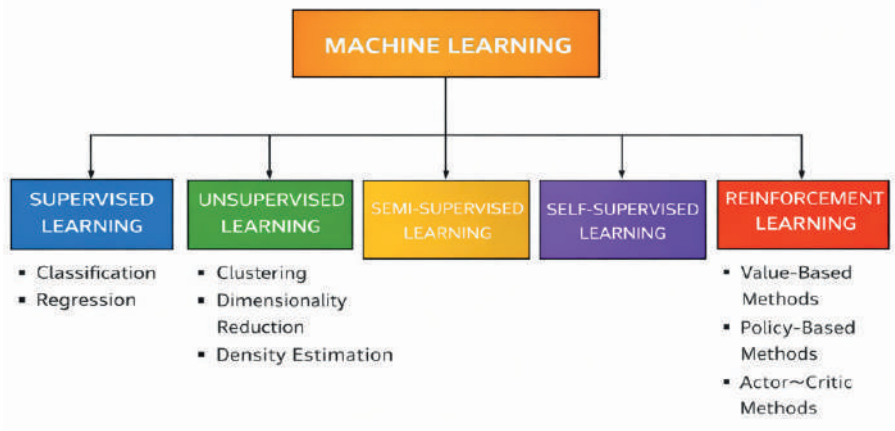


Figure 3. The major categories of ML

Figure 3 illustrates the general categories of machine learning (ML). Each category can be selected depending on the specific objectives and characteristics of the data. For example, supervised learning approaches are generally more suitable for labeled datasets such as tabular data, time-series data, images with annotations, text with labels, and audio with ground-truth information. In contrast, unsupervised learning methods are often preferred for analyzing high-dimensional tabular data, image and text embeddings, and sensor data, where explicit labels are not available.

In this chapter, we examine different types of music data that can be modeled and discuss various machine learning approaches used to process these data types across a range of applications and analytical objectives.

1.2. What is Music?

The ontological question "What is music?" has intrigued scholars across a wide range of disciplines for decades, including philosophy, musicology, sociology, cultural anthropology, history, archaeology, theology, mythography, biology, the humanities, cognitive science, neuroscience, and cultural studies, among others. According to widely recognized dictionaries—such as the Concise Oxford Dictionary, the Penguin Dictionary of Music, the Compact Oxford English Dictionary, and the Merriam-Webster Dictionary—the term music originates from the Greek word mousiké (μουσική), meaning "the art of the nine Muses." Music is commonly defined as the art or science of combining or arranging vocal or instrumental sounds to produce aesthetically pleasing compositions. Such compositions are typically characterized by beauty, form, harmony, melody, rhythm, unity, continuity, and emotional expression (Kokkidou, 2022).

Music is a highly multifunctional medium of communication and interaction, serving as a system through which individuals and communities engage with each other and with their environment. Across cultures, music fulfills a wide range of social, cultural, and emotional functions in both individual and collective contexts. These functions are accompanied by considerable variability in musical structures, practices, and phenomena, as well as by sets of rules, norms, and values that govern participation and shape music's role in human societies (Regelski 2020, Hargreaves et all. 2005). In essence, music serves as a vehicle for communication both among individuals and between individuals and society.

May Kokkidou conducted the Music Definition Project to explore the fundamental nature of music (Kokkidou, 2022). The study involved extensive collaboration with postgraduate music students, many of whom were practicing

music teachers. The participant group comprised thirty-six women and twenty-one men, aged 18 to 49 years, with approximately 60% aged 30 or younger. This research prompted new and challenging questions, such as: "What is music for?" and "What does music do?" These questions highlight the importance of systematically investigating music to guide scientific research in the field.

Research on music spans a wide spectrum, from music perception and psychology to emotional responses to music and computational approaches using machine learning.

In this chapter, music data types for machine learning (ML), the key components of ML applied to music data, and numerous studies employing ML in music analysis are discussed. The aim is to provide foundational knowledge and practical guidance for researchers or practitioners who wish to learn or implement machine learning techniques for tasks such as classification, clustering, prediction, or achieving specific objectives using music data.

2. Music Data Types for ML

In 1965, at the age of 17, Ray Kurzweil developed one of the earliest computer programs capable of composing music in the style of classical composers. This program, written in Fortran, was demonstrated on the television show "What's My Secret?" and gained wider recognition following his appearance on the CBS program I've Got a Secret. Although Kurzweil's system did not employ neural networks, it is regarded as an early example of a symbolic AI system, relying on pattern-based rule systems for music composition.

In 1977, research on automated musical sound processing was conducted at the University of Michigan and published in the journal Automatic Music Transcription. The study utilized a low-pass filter at the Nyquist frequency and sampled musical signals at an effective rate of 4,000 samples per second to input music into a computer. The dataset consisted of flute recordings captured on reel-to-reel tape, which were subsequently converted into digital files using an analog-to-digital converter. Due to the memory limitations of computers at the time, each file was restricted to a duration of 60 seconds (Benetos et all., 2019).

2.1. Forms of Music Data

Music data can take several forms, including audio signals (e.g., waveforms, spectrograms), symbolic representations (e.g., MIDI files, sheet music), and metadata (e.g., genre, artist, tempo, mood, lyrics). Each type of music data

requires distinct machine learning techniques tailored to its structure and characteristics.

2.1.1. Audio signals (waveforms, spectrograms)

The two most common methods for visualizing audio data are waveforms and spectrograms, both of which are widely used by machine learning models to analyze and interpret sound, including speech. Raw audio files are typically stored in formats such as WAV, MP3, or FLAC. In machine learning, raw audio is often represented as a time-series of amplitudes, with typical applications including genre classification and instrument recognition.

Audio representations in the time–frequency domain, such as Short-Time Fourier Transform (STFT), Mel-spectrograms, or Mel-Frequency Cepstral Coefficients (MFCCs), are displayed as 2D arrays (time × frequency). These representations are particularly useful for applications such as music transcription, emotion recognition, and other tasks that require analysis of both temporal and spectral characteristics of sound.

The piano-roll is a simplified representation that encodes only the pitches of musical notes and their temporal arrangement. It is typically presented as a two-dimensional graph, with pitch displayed along the vertical axis and time along the horizontal axis. Figure 4 illustrates three complementary representations of the opening of W. A. Mozart's *Piano Sonata No. 5, KV. 283, II. Andante*: a piano-roll (top), a spectrogram (middle), and a waveform (bottom).

The **piano-roll** provides a symbolic depiction of the music, showing the precise pitch and duration of each note, which facilitates analysis of melodic, harmonic, and rhythmic structures. In contrast, the **spectrogram** represents the frequency content of the sound over time, with intensity or color indicating amplitude, allowing detailed examination of harmonic content, timbre, and articulation. Finally, the **waveform** displays the raw audio signal, with amplitude plotted over time, making it particularly useful for observing overall dynamics and temporal variations in loudness (Pricop and Iftene, 2024).

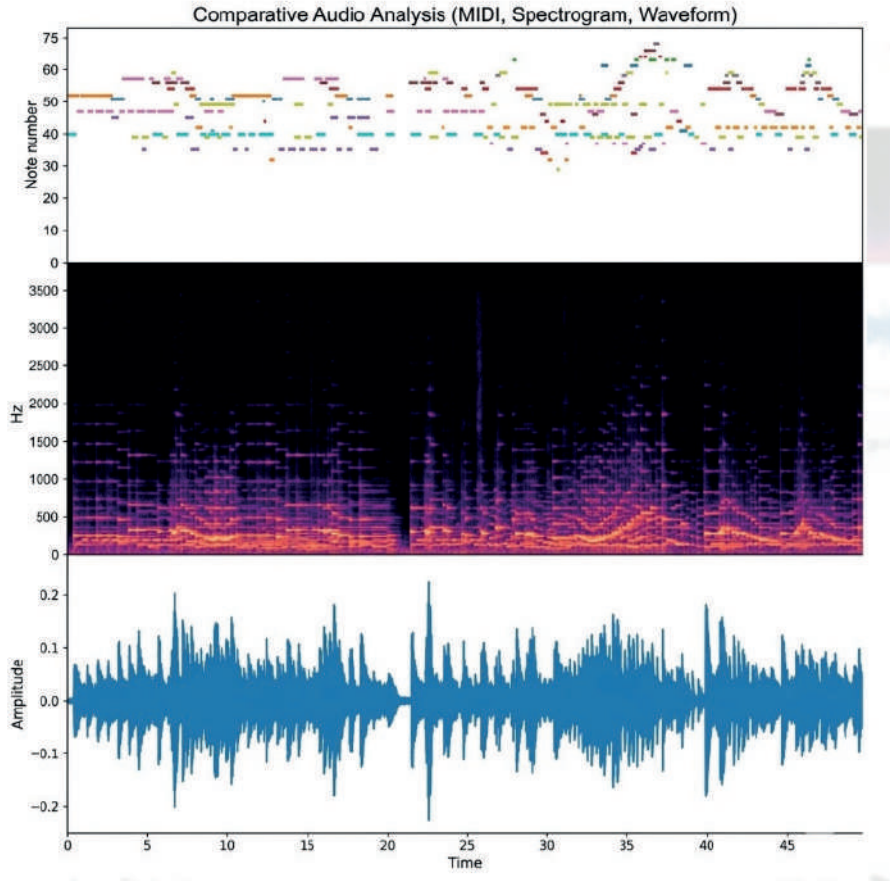


Figure 4. Three different representations of the beginning of W. A. Mozart Piano Sonata No. 5, KV. 283, II. Andante: piano-roll, spectrogram, waveform (from top to bottom, respectively) (Pricop and Iftene, 2024)

2.1.2. Symbolic data (MIDI, sheet music)

A collection of musical events, or notes, constitutes symbolic music data, which typically includes information about timing (onset, offset, duration), pitch, volume, and instrumentation. Various formats are used to store music files digitally, with one of the most widely recognized being the Standard MIDI (Musical Instrument Digital Interface) format. Introduced in August 1982, MIDI is a protocol designed to facilitate communication between keyboards and synthesizers, based on the Western musical interval system. In this system, the smallest interval, the semitone, corresponds to a single step in MIDI (Gedik, 2013; Plasser and Widmer,2023).

Standard MIDI Files are composed of discrete events, each consisting of two key components: a MIDI time, which specifies the timing of the event, and a MIDI message, which conveys the musical action (e.g., note on, note off, control change).

Standard MIDI Files contain events. Standard MIDI Files contain two components:

"a MIDI time, and a MIDI message".

The list of midi events: 00 90 40 40 00 90 43 40 81 00 80 43 00 00 90 45 40 81 00 80 45 00 00 80 40 00 00 90 3C 40 00 90 47 40 81 00 80 47 00 00 90 48 40 81 00 80 48 00 00 80 3C 40 for an example of a score in Figure 5 (Airy, S., Parr, J. M. 2001; de Oliveira, H., Oliveira, R., 2017; Heckroth, 1998).

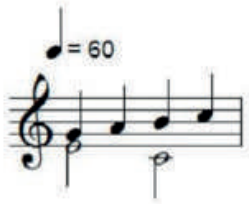


Figure 5. An example of a score

MIDI files are often difficult for humans to read and edit. Additionally, bit-level errors can corrupt an entire file, and transferring files between different applications can be challenging. One of the primary applications of machine learning for music data is automatic music transcription, which is essential for processing and analyzing musical information.

In (de Oliveira and Oliveira, 2017), the authors present an automatic piano transcription system that converts polyphonic audio recordings into musical scores using a machine learning model. The system transcribes audio recordings (WAV files) from the MIDI Aligned Piano Sounds (MAPS) database into standard music notation. For this purpose, the corresponding MusicXML files are used to represent the musical scores.

The Automatic Music Transcription Pipeline described in the study is illustrated in Figure 6. This work is a seminal study demonstrating how audio signals can be converted into symbolic music data. The transcription process begins with multipitch detection in MIDI format, followed by MIDI

quantization, which converts the detected pitches into a structured musical score representation (Benetos et al., 2019).

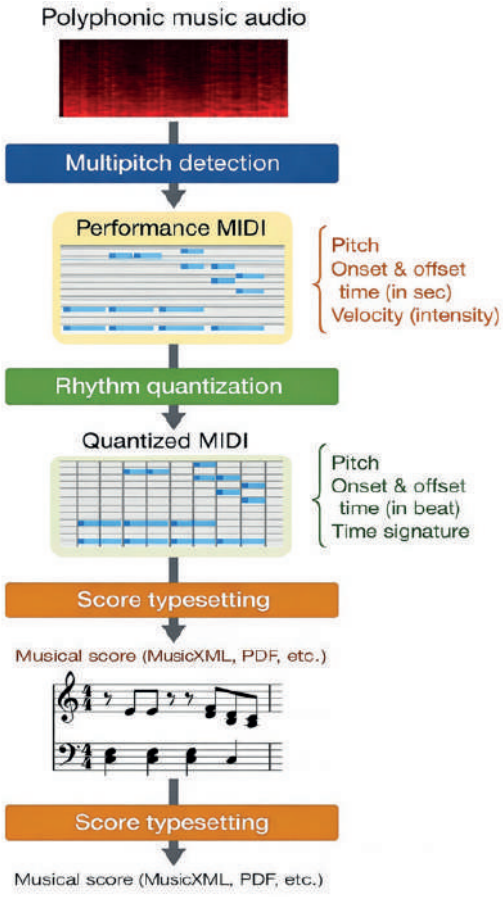


Figure 6. The Automatic Music Transcription Pipeline

MusicXML is an open, flexible, and platform-independent XML-based file format with a hierarchical, tag-based structure that is also human-readable. It is specifically designed for representing symbolic Western music notation. A MusicXML file can store a wide range of musical information, including notes (pitch, duration), rests, chords, measures and bar lines, clefs, key and time signatures, dynamics, articulations, slurs, ties, as well as structural elements such as voices, instrument definitions, tempo markings, repeats, and codas.

Importantly, MusicXML encodes musical score information rather than audio, meaning it specifies what to play rather than the actual sound waveform. Other examples of XML-based music markup languages include ChordML,

MCML (Music Contents Markup Language), EMNML (Extensible Music Notation Markup Language), MML (Music Markup Language), MNML (Music Notation Markup Language), MEI (The Music Encoding Initiative), MX-IEEE 1599, WEDELMUSIC [13] and Lakh MIDI Dataset (Raffel, 2016).

MusicXML is widely supported by various applications, including sheet music software, music editors, educational programs, and music databases. An example of the MusicXML format is illustrated in Figure 7. The MusicXML format is easy to understand and can be readily converted or translated into other music notation formats.

```
<?xml version="1.0" encoding="UTF-8"?>
<score-partwise version="3.1">
  <part-list>
    <score-part id="P1">
      <part-name>Music</part-name>
    </score-part>
  </part-list>

  <part id="P1">
    <measure number="1">
      <attributes>
        <divisions>1</divisions>
        <key><fifths>0</fifths></key>
        <time><beats>4</beats><beat-type>4</beat-type></time>
        <clef><sign>G</sign><line>2</line></clef>
      </attributes>

      <note>
        <pitch><step>C</step><octave>4</octave></pitch>
        <duration>4</duration>
        <type>whole</type>
      </note>

    </measure>
  </part>
</score-partwise>
```

Figure 7. Example of MusicXML format

2.1.3. Metadata (Genre, artist, tempo, mood, lyrics)

Lyrics is textual format and most ML application with it are listed as Lyrics is Lyrics is textual format, genre classification, hit song prediction, lyric generation, identifying common topics in songs. While music metadata type with genre, artist and/ or tempo formats is represented numerical/ sequential, user interaction data type with play counts, ratings is numerical/ sequential. On the other hand, most application in Ml for both is recommendation systems. MIDI (Musical Instrument Digital Interface) is a technical standard that enables electronic musical instruments, computers, and other digital devices to communicate and synchronize with each other. Importantly, MIDI does not transmit audio; instead, it conveys digital instructions specifying how music should be played.

In Music Information Retrieval (MIR) and computational musicology, "music data" exists at multiple levels, ranging from raw acoustic signals to symbolic representations and user-context metadata. The optimal choice of music data depends on the research objectives.

For example:

Playlist recommendation often uses the AotM-2011 dataset (McFee and Lanckriet, 2012).

Singing voice separation employs several datasets:

MIR-1K, designed specifically for singing voice separation, contains 1,000 song clips and has been widely used in multiple studies (Hsu et al., 2011; Hsu et al., 2015).

ccMixter (Liutkus and Rafii, 2017), a community-driven collection of shared and open music.

MUSDB18, which includes 150 full-length tracks (~10 hours) across multiple genres, along with isolated stems for drums, bass, vocals, and other instruments. MUSDB18 is organized into a training set ("train") of 100 songs and a test set ("test") of 50 songs.

In (Prabhakar and Lee, 2023), three datasets—GTZAN (Tzanetakis and Cook, 2002), ISMIR, and MagnaTagATune (collected via the TagATune game and music from the Magnatune label—were evaluated using a combination of Bidirectional Long Short-Term Memory (BLSTM) networks and a Graph Convolutional Network, achieving an accuracy of 93.51%.

A substantial body of research in the literature focuses on machine learning applications using digital music platforms, which predominantly feature

Western music. Among these, the GTZAN Genre Dataset is one of the most widely used. It consists of 1,000 audio recordings evenly distributed across 10 music genres, with each recording having a duration of 30 seconds. Due to its structured format and balanced genre distribution, the GTZAN dataset has been extensively employed for tasks such as music genre classification, audio feature extraction, and algorithm benchmarking. As a result, a large number of studies applying various classification algorithms to the GTZAN dataset are available in the literature.

In recent years, the widely used music streaming platform Spotify has made portions of its music-related data publicly accessible, facilitating academic research. Similar to many large-scale music datasets, the Spotify dataset predominantly features Western music and does not include Turkish-language compositions. The dataset contains 42,305 songs across 15 genres, including Trap, Techno, Techhouse, Trance, Psytrance, Dark Trap, Drum and Bass (DnB), Hardstyle, Underground Rap, Trap Metal, Emo, Rap, RnB, Pop, and Hip-hop.

This dataset has been employed in numerous studies for classification and prediction tasks. For example, some research has focused on predicting the popularity scores of songs based on musical features and metadata (Rosenblatt, 1958). Additionally, several studies have explored genre classification and feature-based clustering using this dataset (Kokkidou, 2022; Regelski, BBB; Hargreaves, 2005; Benetos et al., 2019; Gedik, 2013; Plasser and Widmer, G., 2023; Shibata et al., 2021). Despite the abundance of research conducted using Western music datasets, studies focusing on the makam classification of Turkish classical music remain limited in the literature. One of the major resources containing Turkish music is the Dunya CompMusic Dataset, developed within the framework of the CompMusic Project. This dataset contains audio recordings, musical scores, metadata, and annotations specifically prepared for computational analysis. It includes approximately 6,500 audio recordings corresponding to around 412 hours of music. In addition, the dataset contains 2,200 musical scores accompanied by detailed metadata, including the title of the piece, makam information, and instrument data (McFee and Lanckriet, 2012).

The Dunya CompMusic platform is not merely an archive for listening; it is a structured academic resource developed to support research in Music Information Retrieval (MIR) and computational musicology. Within the CompMusic/Dunya ecosystem, several specialized sub-datasets have been created to address different research objectives.

One of the most notable of these is the SymbTr Turkish Makam Music Symbolic Dataset, a symbolic dataset specifically designed for the analysis of Turkish makam music. The dataset comprises 2,200 musical pieces, representing approximately 150 different makams, nearly 100 rhythmic patterns (usul), and around 50 musical forms. In total, it contains approximately 865,000 musical notes, corresponding to roughly 80 hours of nominal playback duration (Hsu, 2011).

The dataset provides musical data in multiple formats, including Text, MusicXML, PDF, MIDI, and MU2, enabling diverse analytical approaches and computational experiments. Detailed information regarding the dataset's characteristics and structure is available on the official dataset website.

2.2. ML in Music

A search in Scopus using the keywords "Machine Learning" and "Music", limited to Computer Science and restricted to articles, abstracts, and keywords from 2015 to 2026, yielded 3,025 documents. A similar search with the keywords "Deep," "Learning," and "Music" returned 3,451 documents, while a search using "Machine," "Learning," "Music" resulted in 3,396 documents. These numbers indicate that interest in the application of machine learning to music has increased steadily over the years.

The question "Can machines think?" (Koul, 2019) is widely considered one of the fundamental questions that led to the emergence of Artificial Intelligence (AI) as a field of study. The capability of a machine to imitate intelligent human behavior is referred to as **Artificial Intelligence (AI)**. AI encompasses computer systems or sets of algorithms designed to perform tasks that typically require human intelligence, such as learning from data, recognizing speech or images, making decisions, solving problems, and understanding natural language (Samuel, 1959; Mycka and Mańdziuk, 2025).).

Machine Learning (ML) is a subfield of artificial intelligence that enables computers to learn patterns from data and improve their performance without being explicitly programmed for every task. Machine learning algorithms analyze data, identify underlying patterns, and use these patterns to make predictions or decisions.

Deep Learning (DL) is a specialized subfield of machine learning and representation learning that relies on artificial neural networks with multiple processing layers. These networks learn hierarchical representations of data, allowing models to capture complex patterns and multiple levels of abstraction (Samuel, 1959).

Figure 8 represents the relationship between AI, ML, and DL.

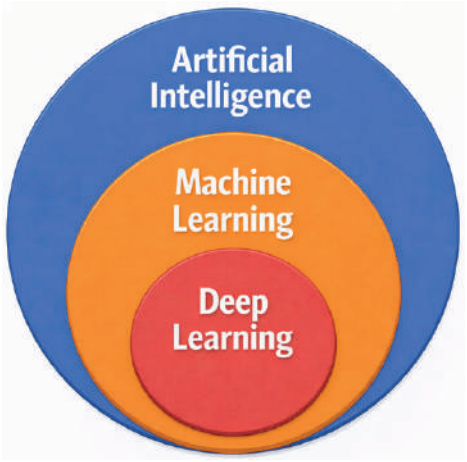


Figure 8. The Relationship between AI, ML and DL

From Figure 8, Machine learning is a subset of artificial intelligence (AI), while deep learning—which includes techniques such as Natural Language Processing (NLP), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs)—is a specialized subset of machine learning. Both machine learning and deep learning can be broadly categorized into two main types: supervised learning, which relies on labeled data, and unsupervised learning, which identifies patterns from unlabeled data.

2.3. Basic Steps in Machine Learning

The basic steps involved in developing a machine learning solution are illustrated below:

Problem Definition: Clearly define the task to be addressed and determine the most suitable type of machine learning approach, such as supervised, unsupervised, or reinforcement learning.

Data Collection: Gather relevant data from reliable sources. In music-related applications, this may include audio files, symbolic representations (e.g., MIDI), and associated metadata.

Data Preprocessing – Prepare the dataset by handling missing values, normalizing or standardizing features, and partitioning the data into training, validation, and test sets.

Feature Extraction and Engineering – Convert raw data into informative features suitable for machine learning algorithms. For instance, Mel-Frequency Cepstral Coefficients (MFCCs) are commonly employed for audio analysis.

Model Selection – Choose an appropriate machine learning algorithm based on the problem and dataset characteristics. Typical options include Support Vector Machines (SVM), Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN).

Model Training – Train the selected model using the training dataset and optimize relevant hyperparameters to enhance predictive performance.

Model Evaluation – Evaluate the trained model on unseen data using appropriate performance metrics, such as accuracy, F1-score, or Root Mean Square Error (RMSE), and refine the model as necessary.

Deployment – Integrate the validated model into real-world applications, such as music recommendation systems, emotion recognition tools, or genre classification platforms.

Monitoring and Maintenance – Continuously monitor model performance in the operational environment and retrain or update the model as new data becomes available to ensure sustained effectiveness.

Figure 9 illustrates the fundamental steps, along with brief descriptions, that must be considered when developing a machine learning solution.

Figure 9. Basic steps in ML

2.4. Music Data Types for ML

Music Information Retrieval (MIR) is a research and technological field dedicated to analyzing, understanding, and extracting information from music data using a variety of techniques, including signal processing, machine learning, and data science. Within MIR, common tasks include genre classification, beat and tempo tracking, beat detection, key and chord recognition, instrument

recognition, and music similarity, clustering, etc. Figure 10 provides a summary of the various types of music data utilized in machine learning, along with the machine learning techniques most commonly applied to each type.

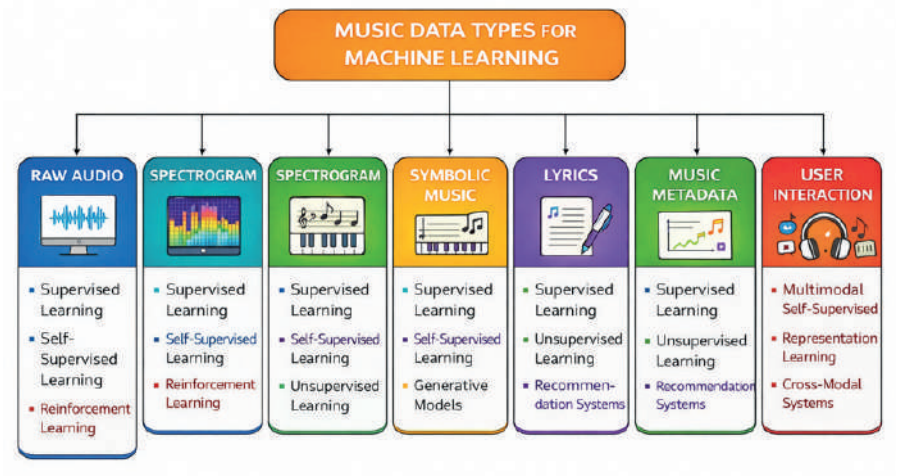


Figure 10. Music Data Types for ML

2.5. Key Applications of ML in Music

In the music domain, machine learning (ML) methods are frequently applied to music data to achieve specific objectives. ML supports a wide range of applications, which can be structured into a taxonomy of music data types and corresponding tasks. Key applications include:

- Music classification for listener recommendations or predicting user preferences for new music.
- lyric analysis, automatic music generation, including composition of original music or emotion-based music generation aimed at evoking specific emotional responses.
- Genre categorization and style modeling, as well as mood detection.
- Automatic music transcription, converting audio recordings into symbolic music notation.
- Score following, which tracks a live performance in relation to a written score.

These objectives can involve partial or fully integrated ML applications, depending on the complexity of the task.

There are several motivations for researchers to study music classification. For instance, songs can be categorized based on the emotional responses they evoke using emotion-based algorithms, or organized to help listeners discover similar music that matches their preferences. In other words, classification enables the identification of potential new favorite songs in a time-efficient manner.

The literature includes numerous studies on music classification, each pursuing different objectives and employing a variety of dataset types.

Machin

77e learning has had a profound impact on music research, providing innovative approaches to music generation, style modeling, and genre classification. Advances in machine learning models have made it possible to create music that emulates human compositional techniques, opening new opportunities for creativity and artistic innovation (Tang et al., 2018).

Table 1 summarizes selected studies from the literature, including the machine learning approaches employed, their research objectives, and their applications to various types of music data.

Table 1. Several representative studies from the literature, highlighting the types of ML, used, their objectives.

The Used Keywords of Related Researches	**Objectives**	**References**
Convolutional Neural Networks (CNN) Support Vector Machines (SVMs)	Categorize music into genres, moods, or other attributes based on audio features	Purwins et al., 2019; Puig, 2019; Schedl, 2019; Briot et al., 2017; Costa et al., 2017; Senac et al., 2022; Lu et al., 2022; Martín-Gutiérrez et al., 2022
Deep Learning (Music Deep Learning)	Automatic generator	He and Zhan, 2025
Deep Learning (Music Deep Learning)	Singing voice detection	Monir et al., 2022
Deep Learning (Music Deep Learning)	Instrument recognition	Choi et al., 2017
Deep Learning (Music Deep Learning)	Music emotion recognition, classification	Han et al., 2022; Zhao et al., 2018
Deep Learning (Music Deep Lear+ning)	Transcription	Sturm, 2016

Deep Learning (DL) - Convolutional Neural Network (CNN)	The Live emotion based music recommendation	Pooja et al., 2026
Deep Learning (DL)	Artists classification; Music classification	Mycka and Mańdziuk,2026
Deep Belief Networks (DBNs) Recurrent Neural Networks (RNNs) Variational Autoencoders (VAEs) Long Short-Term Memory (LSTM) Networks and Transformers	Music transcription	Rafii et al., 2018; Monir et al., 2022; Choi et al., 2017; Han et al., 2022; Sturm et al., 2016; Casini and Roccetti,20182018
Deep Neural Networks (DNNs) Machine Learning (ML) Long Short-Term Memory (LSTM)	Advanced music generation	Pricop and Iftene, 2024
Long Short-Term Memory (LSTM) Recurrent neural network (RNN) Markov Chains	Automatic music generation	Cretu et al., 2022
Deep Learning (Music Deep Learning) Long Short-Term Memory (LSTM)	Music generation	Chou and Peng, 2019
Support Vector Machine (SVM) Random Forest (RF) Artificial Neural Network (ANN)	Music classification	Dai and Huang, 2022
Artificial Neural Network (ANN) Convolutional Neural Network (CNN)	Classify the music based on certain instruments	Shreevathsa et al., 2020
Machine Learning (ML)	Automatic evaluation system for piano music performance	Xiao, 2026
Machine Learning (ML)	Cultural music classification	Abebe et al., 2026
Machine Learning (ML)	The Music Emotion Recognition (MER)	Louro et al., 2026
Unsupervised Disentanglement Strategies	Controllable Music Generation	Ibáñez-Martínez et al., 2026

3. Conclusion

In recent years, machine learning techniques have increasingly shaped the analysis and processing of musical data. The rapid expansion of digital music platforms and the availability of large-scale datasets have enabled researchers to employ advanced computational methods across a wide range of music-related tasks. In particular, machine learning has been extensively applied in areas such as Music Information Retrieval (MIR), classification of artists, genres and moods, recommendation systems, automatic music generation and music emotion recognition, etc.

By leveraging large-scale musical datasets and sophisticated computational models, machine learning methods facilitate the systematic analysis and interpretation of musical information. These approaches enable the extraction of meaningful features from audio signals, symbolic music representations, and associated metadata. Such capabilities support diverse analytical tasks, including music genre recognition, mood detection, popularity prediction, and automatic transcription.

Accordingly, machine learning has become a critical tool for both academic research and practical applications in music analysis and technology. Its adoption has contributed to the development of more efficient music analysis systems and intelligent digital music services, advancing the capabilities of computational approaches in understanding and interacting with music.

Research on the application of machine learning techniques to music data continues to advance rapidly. It is hoped that this chapter will serve as a useful resource for researchers seeking to pursue studies in this area.

References

Abebe, M., Endashew, L., Heikkonen, J., Kanth, R., & Mohapatra, S. K. (2026). Decoding cultural music classification with machine learning and segment length analysis. *SN Computer Science*. https://doi.org/10.1007/s42979-026-04747-6 Benetos, E., Dixon, S., Duan, Z., & Ewert, S. (2019). Automatic music transcription: An overview. *IEEE Signal Processing Magazine, 36*(1), 20–30. https://doi.org/10.1109/MSP.2018.2869928

Airy, S., & Parr, J. M. (2001). MIDI, music and me: Students' perspectives on composing with MIDI. *Music Education Research, 3*(1), 41–49. https://doi.org/10.1080/14613800020029941

Bishop, G. (n.d.). *Music datasets for machine learning*. *Medium*. Retrieved March 8, 2026, from https://gail-bishop.medium.com/music-datasets-for-machine-learning-a6cd8d707340

Briot, J.-P., Hadjeres, G., & Pachet, F.-D. (2017). Deep learning techniques for music generation—A survey. *arXiv*. https://arxiv.org/abs/1709.01620

Casini, L., Marfia, G., & Roccetti, M. (2018). Some reflections on the potential and limitations of deep learning for automated music generation. In *IEEE 29th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)* (pp. 27–31).

Choi, K., Fazekas, G., Cho, K., & Sandler, M. (2017). A tutorial on deep learning for music information retrieval. *arXiv*. https://arxiv.org/abs/1709.04396

Costa, Y. M. G., Oliveira, L. S., & Silla, C. N. (2017). An evaluation of convolutional neural networks for music classification using spectrograms. *Applied Soft Computing, 52*, 28–38.

Cretu, B.-A., Vranceanu, A., Cojocariu, A., Simionescu, C., & Iftene, A. (2022, August 8–10). *Music generation using neural nets*. In *Proceedings of the International Conference on Innovations in Intelligent Systems and Applications (IEEE INISTA 2022)* (pp. 1–6). Biarritz, France: IEEE.

Dai, Z., & Huang, X. (2022). Electronic dance music classification based on machine learning methods. In *2022 International Conference on Electronics and Devices, Computational Science (ICEDCS)* (pp. 351–354). IEEE. https://doi.org/10.1109/ICEDCS57360.2022.00084

de Oliveira, H., & Oliveira, R. (2017). Understanding MIDI: A painless tutorial on MIDI format. *arXiv*. https://doi.org/10.48550/arXiv.1705.05322

Essinger, S. D., & Rosen, G. L. (2021). *An introduction to machine learning for students in secondary education*. Drexel University, Department of Electrical & Computer Engineering. https://exampleurl.edu

Gedik, A. (2013). Türk müziği icralarının analizi için bir MIDI projesi. *Yedi, 10*, 81–92. https://izlik.org/JA45NM54LS

Gray, B., Geis, R., O'Connell, T., Babyn, P., Koff, D., & Shabana, W., et al. (2018). Canadian Association of Radiologists white paper on artificial intelligence in radiology. *Canadian Association of Radiologists Journal, 69*, 120–135. https://doi.org/10.1016/j.carj.2018.02.002

Han, D., Kong, Y., Han, J., & Wang, G. (2022). A survey of music emotion recognition. *Frontiers of Computer Science, 16*(6), 1–11.

Hargreaves, D. J., MacDonald, R., & Miell, D. (2005). How do people communicate using music? In D. Miell, R. MacDonald, & D. J. Hargreaves (Eds.), *Musical communication* (pp. 1–26). Oxford University Press.

He, Y., & Zhan, Y. (2025). Application of machine learning in automatic generation of intangible cultural heritage music scores. In *Proceedings of the International Conference on Artificial Intelligence and Computer Engineering* (pp. 2502–2505). IEEE. https://doi.org/10.1109/ICAACE65325.2025.11019496

Heckroth, J. (1998). *A tutorial on MIDI and wavetable music synthesis* (Application Note AN27). Crystal Semiconductor Products Division, Cirrus Logic, Inc.

Honi, P. (2023). *Exploring the synergy between generative AI, data and analytics in the modern age*. TechRxiv. https://doi.org/10.36227/techrxiv.24045792

Hsu, C.-L., Jang, J. S. R., & Wang, D. (2011). A trend estimation algorithm for singing pitch detection in musical recordings. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Hsu, C.-L., Jang, J. S. R., & Wang, D. (2010). On the improvement of singing voice separation for monaural recordings using the MIR-1K dataset. *IEEE Transactions on Audio, Speech, and Language Processing, 18*(2), 310–319.

Hsu, C.-L., Wang, D., Jang, J. S. R., & Hu, K. (2011). A tandem algorithm for singing pitch extraction and voice separation from music accompaniment. *IEEE Transactions on Audio, Speech, and Language Processing*.

Ibáñez-Martínez, L., Nkama, C., Poltronieri, A., Serra, X., & Rocamora, M. (n.d.). *Evaluating disentangled representations for controllable music generation*. Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain.

Kokkidou, M. (2022). *Music definition and music education: Many perspectives, many voices, many questions.*

Koul, A. (2019). What is AI? In *Practical deep learning for cloud, mobile, and edge: Real-world AI and computer vision projects using Python, Keras, and TensorFlow* (pp. 6–7). O'Reilly Media.

Liutkus, A., Fitzgerald, D., & Rafii, Z. (2015). Scalable audio separation with light kernel additive modelling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 76–80).

Louro, P. L., et al. (2026). MERGE: A bimodal audio-lyrics dataset for static music emotion recognition. *IEEE Transactions on Affective Computing*. https://doi.org/10.1109/TAFFC.2026.3668037

Merriam, A. P. (1964). *The anthropology of music*. Northwestern University Press.

Merriam-Webster. (n.d.). *Artificial intelligence*. In *Merriam-Webster.com dictionary*. Retrieved March 7, 2026, from https://www.merriam-webster.com/dictionary/artificial%20intelligence

Martín-Gutiérrez, D., Penaloza, G. H., Belmonte-Hernandez, A., & Garcia, F. A. (2020). A multimodal end-to-end deep learning architecture for music popularity prediction. *IEEE Access, 8*, 39361–39374. https://doi.org/10.1109/ACCESS.2020.2979145

McFee, B., & Lanckriet, G. R. (2012). Hypergraph models of playlist dialects. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)* (pp. 343–348).

Minsky, M., & Papert, S. (1969). *Perceptrons: An introduction to computational geometry*. MIT Press.

Monir, R., Kostrzewa, D., & Mrozek, D. (2022). Singing voice detection: A survey. *Entropy, 24*(1), 114.

Mycka, J., & Mańdziuk, J. (2025). Artificial intelligence in music: Recent trends and challenges. *Neural Computing and Applications, 37*, 801–839. https://doi.org/10.1007/s00521-024-10555-x

Mycka, J., & Mańdziuk, J. (2026). Classification of piano performers with deep learning models. *Journal of Computational Science*. https://doi.org/10.1016/j.jocs.2026.102804

Plasser, M., Peter, S., & Widmer, G. (2023). Discrete diffusion probabilistic models for symbolic music generation. *arXiv*. https://doi.org/10.48550/arXiv.2305.09489

Pooja, M., Kavyashree, H. L., Pratham M., & Meghana, G. (2026). Live emotion based music recommender system. In *Proceedings of the 2026 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)* (pp. 1–6). IEEE. https://doi.org/10.1109/IITCEE67948.2026.11394620

Pricop, T.-C., & Iftene, A. (2024). Music generation with machine learning and deep neural networks. *Procedia Computer Science, 246*, 1855–1864. https://doi.org/10.1016/j.procs.2024.09.692

Puig, J. P. (2019). *Deep neural networks for music and audio tagging* (Doctoral dissertation, Universitat Pompeu Fabra, Barcelona, Spain).

Raffel, C. (2016). *Learning-based methods for comparing sequences, with applications to audio-to-MIDI alignment and matching* (Doctoral dissertation). Columbia University.

Rafii, Z., Liutkus, A., Stöter, F.-R., Mimilakis, S. I., FitzGerald, D., & Pardo, B. (2018). An overview of lead and accompaniment separation in music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 26*(8), 1307–1335.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65*(6), 386–408. https://doi.org/10.1037/h0042519

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development, 3*(3), 210–229. https://doi.org/10.1147/rd.33.0210

Schedl, M. (2019). Deep learning in music recommendation systems. *Frontiers in Applied Mathematics and Statistics, 5*, 44.

Shreevathsa, P. K., Harshith, M., R. M. A., & Ashwini. (2020). Music instrument recognition using machine learning algorithms. In *International Conference on Computation, Automation and Knowledge Management (ICCAKM)* (pp. 161–166). IEEE.

Sturm, B. L., Santos, J. F., Ben-Tal, O., & Korshunova, I. (2016). *Music transcription modelling and composition using deep learning*. arXiv. https://arxiv.org/abs/1604.08723

Tang, A., Tam, R., Cadrin-Chênevert, A., Guest, W., Chong, J., Barfett, J., Chepelev, L., Cairns, R., Mitchell, J., Cicero, M., Gaudreau Poudrette, M., Jaremko, J., Reinhold, C., Gallix, B., Gray, B., Geis, R., O'Connell, T., Babyn, P., Koff, D., & Shabana, W. (2018). Canadian Association of Radiologists white paper on artificial intelligence in radiology. *Canadian Association of Radiologists Journal, 69*, 120–135. https://doi.org/10.1016/j.carj.2018.02.002

The MagnaTagATune Dataset. (n.d.). *MagnaTagATune dataset*. Retrieved March 6, 2026, from https://mirg.city.ac.uk/codeapps/the-magnatagatune-dataset

Tudor-Constantin Pricop, & Iftene, A. (2024). Music generation with machine learning and deep neural networks. *Procedia Computer Science, 246*, 1855–1864. https://doi.org/10.1016/j.procs.2024.09.692

Tzanetakis, G., & Cook, P. (2002). *GTZAN genre dataset*. Retrieved March 8, 2026, from http://marsyas.info/downloads/datasets.html

URL-1: https://gail-bishop.medium.com/music-datasets-for-machine-learning-a6cd8d707340 (Date Accessed: 22 March 2026).

URL-2: https://zenodo.org/records/3532216, (Date Accessed: 02 Mart 2026).

URL-3: https://ismir2004.ismir.net, (Date Accessed: 06 March 2026).

URL-4:http://marsyas.info/downloads/datasets.html, (Date Accessed: 06 March 2026).

URL-5: https://www.merriam (Date Accessed: 06 March 2026).

Xiao, X. (2026). Design of an automatic evaluation system for piano music performance based on machine learning algorithms. In *Proceedings of the 2025 International Conference on Artificial Intelligence, Virtual Reality and Interaction Design* (pp. 583–587). Association for Computing Machinery. https://doi.org/10.1145/3777730.3777825.

Chapter 6

ConvNeXt-Based Deep Feature Engineering and Machine Learning Approach with Explainable Artificial Intelligence for Guava Fruit Disease Classification

Havva Gül Koçer[1]

Birkan Büyükarıkan[2]

Esin Ayşe Zaimoğlu[3]

Abstract

Guava (*Psidium guajava*) is an important part of tropical agriculture with its high nutritional value and economic return. However, diseases in this fruit cause serious losses in productivity and product quality. The manual and subjective nature of traditional diagnostic methods is insufficient to meet the speed and accuracy requirements of modern agriculture. Therefore, the aim of the study is to identify guava fruit using a model that combines ConvNeXt and machine learning. In this study, ConvNeXt-Tiny, ConvNeXt-Small, ConvNeXt-Base, ConvNeXt-Large and ConvNeXt-XLarge architectures were used as feature extractors. The features extracted from these variants were classified using Support Vector Machines (SVM), Random Forest (RF) and Logistic Regression (LR) algorithms. In addition, different numbers of features (50, 100 and 200) were selected from each ConvNeXt variant by SelectKBest method and new feature vectors were created by combining these features. The classification performances of these feature vectors were also evaluated with SVM, RF, and LR. According to the experimental results, the ConvNeXt-XLarge-SVM model achieved an accuracy of 0.997. The CS200-ConvNeXt-SVM model after feature selection and feature fusion achieved

1 Lecturer Dr. Havva Gül Koçer, Selçuk University, hgul@selcuk.edu.tr, 0000-0003-4083-722X.

2 Assist. Prof. Dr. Birkan Büyükarıkan, Isparta University of Applied Sciences, birkanbuyukarikan@isparta.edu.tr, 0000-0002-9703-9678.

3 Assist. Prof. Dr. Esin Ayşe Zaimoğlu, Sakarya University, esinzaimoglu@sakarya.edu.tr, 0000-0001-9688-9868.

https://doi.org/10.58830/ozgur.pub1236.c5007

1.000 in all performance metrics. Contextual Importance and Utility (CIU), one of the explainable artificial intelligence methods, was used to identify the features that contribute to the decision mechanism of the proposed model. The contributions of the features determined by the CIU method for the CS200-ConvNeXt-SVM model were analyzed on a class basis, and the top 10 features with the highest contribution were identified for each class. It was found that the F570 feature contributed the most in the Healthy guava class, the F625 feature contributed the most in the Anthracnose class, and the F626 feature contributed the most in the Fruit fly class. In addition, evaluations were conducted across different datasets to assess the generalization performance of the proposed model. The findings suggest that combining ConvNeXt variants and machine learning algorithms is an effective approach for guava fruit disease classification.

1. Introduction

The ongoing growth of the world population, along with the prediction that food demand will double by 2050, underscores the importance of sustainable, efficient agricultural practices. Therefore, increasing food demand necessitates strategies to maintain product quality and productivity (Hunter et al., 2017; Mirvakhabova et al., 2018). Agricultural production, which extends beyond meeting basic nutritional requirements to serve as a significant economic driver, includes guava (*Psidium guajava*) as a strategically important crop due to its high nutritional value, industrial versatility, and substantial value-added potential (Joseph and Priya, 2011). The global guava market is expected to reach billions of dollars in volume by 2033. This market continues to increase its commercial value every day, driven by its steady growth trend (Ahmed et al., 2025). Guava fruit, which is cultivated in tropical and subtropical climates, is eaten fresh because of its nutritional value and is also utilized for making fruit juice and jam (Amin et al., 2024; Kılcı and Koklu, 2024). Guava fruit, which is a super fruit because of its four times more vitamin C content than that of orange and its antioxidant properties, is attacked by a variety of diseases, resulting in a considerable loss of yield (Paramesha et al., 2025).

In traditional agricultural practices, disease diagnosis largely relies on farmers or agrarian experts physically observing the field with the naked eye. However, this manual monitoring process, which requires expertise and relies on human factors, falls short of meeting the speed and accuracy requirements of modern agriculture due to its limited applicability across large areas, its time-consuming nature, and its subjectivity, which depends on the observer's experience (Ahmed et al., 2025). Furthermore, it leads to incorrect or delayed diagnoses in the early stages of diseases with similar visual symptoms, triggering irreversible yield losses. This situation not only increases production costs

and environmental pollution but also brings serious commercial obstacles, such as product rejection in export markets due to pesticide residues (Arnal Barbedo, 2013; Paramesha et al., 2025).

The need to improve the accuracy and speed of diagnosis has led to the prominence of digital technologies in agricultural disease management, thus increasing the popularity of Precision Agriculture and Smart Farming practices (Mohanty et al., 2016). In this regard, Convolutional Neural Networks (CNNs), which are deep learning (DL) models, have proven to be more effective in fruit disease image classification than conventional image processing techniques because of their capability to automatically and efficiently extract features from images (Krizhevsky et al., 2012). Studies on guava have typically used well-known classical CNN architectures such as VGGNet, GoogLeNet, ResNet, DenseNet, and EfficientNet, which have achieved high classification performance (Doutoum et al., 2023; Kaur et al., 2024; Mostafa et al., 2022). However, due to its superior performance, the ConvNeXt architecture has recently been widely adopted across many fields, including agriculture (Kanimalar and Karthikeyan, 2025; KP and Gowrishankar, 2023).

ConvNeXt is a DL model that unifies the structural efficiency of CNN-based architectures and the powerful representation learning of Transformer-based architectures. The ConvNeXt model is available at different scales, including Tiny, Small, Base, Large, and XLarge. The main difference between these scales is their depth and capacity (Liu et al., 2022). This allows ConvNeXt to be easily adapted to different datasets and computational requirements. However, end-to-end training of ConvNeXt architectures may increase the risk of overfitting in agricultural datasets due to high computational costs and the need for large-scale labeled data. Therefore, the classification of features extracted from ConvNeXt architectures using machine learning (ML) algorithms is widely used (An et al., 2024; Gao et al., 2023; Solmaz and Tasci, 2025).

The aim of this study is to classify the diseases in guava fruit using an approach that combines ConvNeXt variants and ML algorithms. This present study used a publicly available dataset containing the classes Anthracnose, Fruit Flies, and Healthy. The features were extracted using the dataset, where the ConvNeXt-Tiny, ConvNeXt-Small, ConvNeXt-Base, ConvNeXt-Large, and ConvNeXt-XLarge architectures were utilized, followed by classification using the Support Vector Machine (SVM), Random Forest (RF), and Logistic Regression (LR) algorithms. Moreover, the features extracted using the ConvNeXt variants were also selected using the SelectKBest method. The selected features were combined based on their number, and feature vectors

of varying sizes were created. These vectors were evaluated using SVM, RF, and LR classifiers. Furthermore, to increase transparency into the decision-making mechanism of the high-performing model, an explainable artificial intelligence (XAI) analysis was conducted using the Contextual Importance and Utility (CIU) method. In addition, the behavior of the proposed model on different datasets was also examined. The main contributions of this study to the literature are summarized below:

(i) This study presented an approach for classifying guava fruit diseases using different scales (Tiny, Small, Base, Large, and XLarge) of the ConvNeXt architecture and evaluating the obtained features using ML algorithms.

(ii) Using guava fruit, 50, 100, and 200-dimensional feature vectors were generated from features obtained from ConvNeXt architectures by applying the SelectKBest method, and the effect of these vectors on the performance of ML algorithms was investigated.

(iii) To increase the transparency of the model showing the high performance, a contextual analysis of the decision mechanism was performed using the CIU method.

(iv) The proposed model was evaluated on different datasets to demonstrate its generalization capability.

2. Related Work

The earliest approaches in the literature for detecting guava diseases rely on basic image processing techniques and classical ML algorithms. The main characteristic of studies during this period is that extracting meaningful information from raw images relies on handcrafted features that require human intervention and domain knowledge. For example, Almutiry et al. (2021) classified features extracted using Local Binary Pattern (LBP) and Principal Component Analysis (PCA) with a cubic-SVM model, achieving 98.6% accuracy. Similarly, Almadhor et al. (2021) achieved 99% accuracy using LBP and Bagged Trees. Asim et al. (2023) classified tissue features extracted from guava leaves using the Instant Base Identifier model, achieving an accuracy of 87.125%.

On the other hand, researchers preferred CNN models to overcome the limitations of hand-crafted features. In this context, AlexNet, DenseNet, and ResNet architectures were widely used. Mostafa et al. (2022) classified guava diseases using AlexNet, SqueezeNet, GoogLeNet, ResNet-50, and ResNet-101, achieving the highest accuracy of 97.74% with ResNet-101. Similarly, Tewari et al. (2024) achieved 99% accuracy with the DenseNet169

model on a dataset with four disease classes. Hashan et al. (2024) achieved 93% accuracy using the AlexNet model on a dataset of 612 images. Kaur et al. (2024) reported that, when using Adam optimization with their DenseNet-based model, they achieved 98.76% accuracy.

The goal of disease detection systems is to integrate this technology into easily accessible mobile devices or drone systems with limited processor and battery capacity. In this context, lightweight models that balance parameter count and accuracy stand out for guava damage classification. Doutoum et al. (2023) achieved 94.93% accuracy in their study classifying guava leaf diseases using the EfficientNet-B3 model. In another study, Nath Nandi et al. (2022) reduced the size of the GoogleNet model to 0.143 MB using model quantization techniques and achieved approximately 97% accuracy. Similarly, Mustak Un Nobi et al. (2023) achieved 98% accuracy with their proposed MobileNet-based model. They also evaluated the model's robustness using XAI, employing Grad-CAM. On the other hand, combining the powerful feature extraction capabilities of CNN models with ML algorithms, Kılcı and Koklu (2024) classified features extracted using SqueezeNet with the Gradient Boosting algorithm, achieving an accuracy of 95.6% with their proposed model. In another study, Kılcı and Koklu (2025) combined InceptionV3 with an SVM and achieved 99.74% accuracy with their proposed model.

These studies in the field of guava fruit and leaf disease classification have demonstrated high accuracy with known CNN architectures. However, modern architectural structures such as ConvNeXt have not been sufficiently investigated for the classification of these diseases in the current literature. Furthermore, explainability analysis of which features the model's decision processes are based on has been largely neglected. Therefore, the primary motivation of this study is to extract features of diseases observed in guava fruit using ConvNeXt-based architectures, classify these features using ML, and evaluate the features that affect the model's performance using XAI.

3. Materials and Methods

3.1. Dataset Details

This study used the Guava Disease Dataset downloaded from the Mendeley Data platform. This dataset consists of three classes such as Anthracnose, Fruit Flies, and Healthy guava fruit. The images in the dataset were captured in July in orchards in the cities of Rajshahi and Pabna, Bangladesh, and were labeled and verified by a plant pathologist. Furthermore, all images are 512 × 512 pixels in size and in PNG format. Preprocessing techniques such as unsharp masking and Contrast-Limited Adaptive Histogram Equalization

were applied to improve the image quality of the Guava Disease Dataset. This dataset, consisting of 3784 images, has been divided into training, validation, and test sets at ratios of 70%, 20%, and 10%, respectively (Amin et al., 2024). In this study, the existing data division was preserved, and no re-division process was applied. All images in the dataset were resized to 224 × 224. The distribution of the number of images per class in the dataset is shown in Table 1.

Table 1. Distribution of images across classes in the Guava Disease Dataset

Class	Training set	Validation set	Test set
Anthracnose	1080	308	156
Fruit Flies	918	262	132
Healthy Fruits	649	185	94
Total number of images	2647	755	382

3.2. ConvNeXt-Based Deep Feature Extraction and Classification

3.2.1. ConvNeXt architecture

ConvNeXt, proposed by Liu et al. (2022), is a CNN architecture designed based on the principles of Vision Transformer (ViT) architectures. In the ConvNeXt architecture, 7×7 depthwise convolutions, Layer Normalization, the GELU activation function, and inverted bottleneck structures are used instead of the small-kernel convolutions commonly used in classical CNNs. Furthermore, some design elements from the ResNet architecture have been simplified, with layer normalization preferred over batch normalization. These architectural adjustments have both increased ConvNeXt's training efficiency and strengthened its generalizability across different scales (Liu et al., 2022; Noda and Hashimoto, 2025).

Depending on resource constraints, the ConvNeXt architecture has different variants according to model capacity: Tiny, Small, Base, Large and XLarge (Liu et al., 2022). In scenarios where hardware capacity is limited, ConvNeXt-Tiny and ConvNeXt-Small stand out as efficient options that lighten the computational load by optimizing the number of parameters (Lekkala et al., 2025; Roshanzadeh et al., 2025; Shabrina et al., 2023). The ConvNeXt-Base and ConvNeXt-Large variants offer medium model capacity, providing a trade-off between performance and computational cost (Nan et al., 2025). ConvNeXt-XLarge is an architecture with the highest parameter capacity.

3.2.2. Classification algorithms

SVM is a robust supervised learning algorithm that aims to find the optimal decision hyperplane that maximizes the separation between classes (Cortes and Vapnik, 1995). This algorithm positions itself to maximize the margin, a gap between training data, thereby increasing the model's generalization ability and minimizing overfitting. SVM's reliance on the principle of structural risk minimization and its resilience to outliers make it particularly effective for small- and medium-sized datasets (Hosmer Jr et al., 2013).

LR is a fundamental statistical classification algorithm used to model the relationship between classes when the dependent variable is categorical. Unlike linear regression, this algorithm uses a sigmoid function that limits the probability that the dependent variable belongs to a specific class to values between 0 and 1 (Cox, 1958). The algorithm optimizes the coefficients using the maximum likelihood estimation method and delivers high-performance results even on high-dimensional datasets (Hosmer Jr et al., 2013).

RF is a classification algorithm based on ensemble learning, where multiple decision trees are randomly split and trained, and their predictions are combined to produce a single, stable result. This algorithm builds many decision trees during training and classifies using randomly selected feature subsets at each node. The predictions from individual trees are combined using majority voting to make the final classification decision. The algorithm uses both random sampling from the dataset and selecting a random subset of features at each node when constructing each tree (Breiman, 2001). This randomness mechanism minimizes the risk of overfitting by reducing the correlation between trees and increases prediction accuracy by reducing variance (Hastie, 2009).

3.3. Proposed Approach

In this study, an approach combining feature extraction, feature selection, and ML algorithms with ConvNeXt variants is proposed for the classification of diseases observed in guava fruit. The proposed approach consists of two stages, as shown in Figure 1.

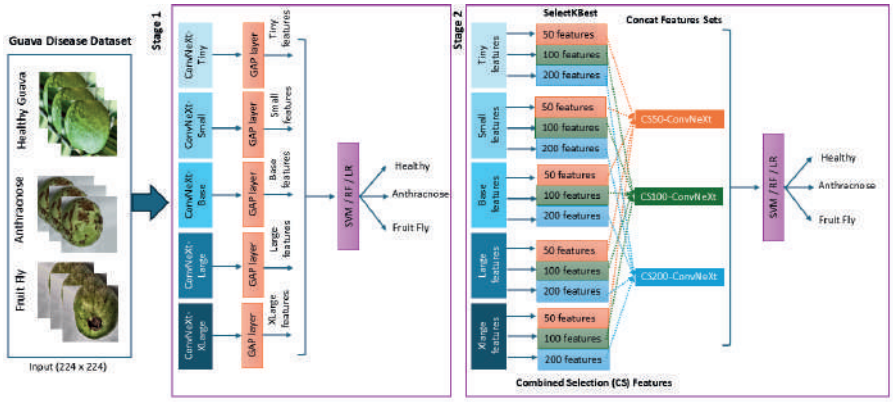


Figure 1. Diagram of the proposed approach

In the first stage, ConvNeXt-Tiny, ConvNeXt-Small, ConvNeXt-Base, ConvNeXt-Large, and ConvNeXt-XLarge architectures were initialized with pre-trained weights from the ImageNet-1k dataset. Global Average Pooling (GAP) was applied to the output of the last convolutional layer of each ConvNeXt variant, and fixed-size feature vectors were obtained. Accordingly, 768-dimensional feature vectors were obtained from the ConvNeXt-Tiny and ConvNeXt-Small architectures, 1024-dimensional feature vectors from the ConvNeXt-Base architecture, 1536-dimensional feature vectors from the ConvNeXt-Large architecture, and 2048-dimensional feature vectors from the ConvNeXt-XLarge architecture. These feature vectors were classified using SVM, RF, and LR algorithms.

In the second stage, the Mutual Information-based SelectKBest feature selection method was applied to these features obtained from each ConvNeXt architecture. The first K distinctive features were selected from each feature extractor. Feature selection was performed only on the training data, and data leakage was prevented by applying the same selective transformations to the validation and test data. In this context, 50, 100, and 200 features were selected from the feature sets extracted by each ConvNeXt architecture in the study. Then, these features were combined based on the number of selections to form feature sets (named Combined Selection (CS) feature count). Accordingly, using all ConvNeXt variants, there are 250 features in the CS50-ConvNeXt feature set, 500 features in the CS100-ConvNeXt feature set, and 1000 features in the CS200-ConvNeXt feature set. Table 2 lists the hyperparameters used in training the ML models and their values. The hyperparameters used in the study were determined based on the validation set performance. Finally, the

CIU method was used to increase the transparency of the model's decision-making mechanism, achieving the highest performance.

All experiments in the study were performed using the Python programming language in the Google Colaboratory environment, and the models were evaluated using the Keras library on v5e-1 TPU. The scikit-learn library was used for training the ML models.

Table 2. Hyperparameters and their values used in ML algorithms

Model	**Parameter**	**Value**
SVM	Kernel	rbf
	C	10
	Gamma	scale
RF	n_estimators	500
	max_depth	None
	random_state	42
	n_jobs	-1
LR	max_iter	5000
	solver	lbfgs
	n_jobs	-1

3.4. Performance Metrics

In this study, accuracy, precision, recall, F1-score, and Cohen's Kappa were used to evaluate the classification performance of the proposed models quantitatively. These metrics are calculated using True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) values. The metrics module of the Scikit-learn library was used to calculate these metrics in this study.

Accuracy is the ratio of correctly predicted examples to the total number of examples and indicates the model's overall success. The accuracy metric is calculated using the mathematical expression given in Equation 1.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Precision indicates how many of the predicted positives are actually positive. The mathematical expression for the precision metric is given in Equation 2.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Recall measures the proportion of true positives correctly captured by the model. The mathematical expression for calculating the recall metric is given in Equation 3.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

The F1-score is the harmonic mean of precision and recall. This metric reflects the model's balanced performance on imbalanced datasets. The F1-score is calculated using the mathematical expression given in Equation 4.

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

Cohen's Kappa measures the difference between observed accuracy and the accuracy expected by chance, evaluating the consistency between the classifier and the actual labels. The mathematical expression used to calculate Cohen's Kappa is given in Equation 5. Here, P_o represents the observed accuracy, and P_e represents the expected chance agreement.

$$\text{Cohen's Kappa} = \frac{P_o - P_e}{1 - P_e} \tag{5}$$

4. Experimental Results and Discussion

4.1. Classification of Features Extracted from ConvNeXt Variants Using Machine Learning Algorithms

The performance results obtained with SVM, RF, and LR classifiers on the Guava Disease Dataset, using features extracted from ConvNeXt variants, are listed in Table 3. According to the findings, all performance metrics increased as the ConvNeXt scale increased. This indicates that deeper, higher-capacity ConvNeXt architectures can represent image discriminative features more effectively. SVM-based models achieved the highest performance across all

ConvNeXt variants. The classification performed with the ConvNeXt-XLarge-SVM model achieved 0.997 accuracy, 0.996 precision, 0.998 recall, 0.997 F1-score, and 0.996 Cohen's Kappa values, respectively.

LR-based models showed a slight performance drop compared to SVM. However, the ConvNeXt-Base-LR and ConvNeXt-XLarge-LR models produced quite competitive results. In contrast, RF-based models achieved the lowest performance results across all scales compared to the other proposed models.

Table 3. Classification performance of ConvNeXt-based features with ML classifiers

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Cohen's Kappa**
ConvNeXt-Tiny-SVM	0.940	0.934	0.936	0.935	0.908
ConvNeXt-Tiny-RF	0.874	0.882	0.865	0.872	0.806
ConvNeXt-Tiny-LR	0.914	0.906	0.904	0.905	0.868
ConvNeXt-Small-SVM	0.969	0.968	0.964	0.966	0.952
ConvNeXt-Small-RF	0.840	0.849	0.827	0.835	0.753
ConvNeXt-Small-LR	0.942	0.941	0.937	0.939	0.912
ConvNeXt-Base-SVM	0.984	0.985	0.983	0.984	0.976
ConvNeXt-Base-RF	0.895	0.898	0.890	0.893	0.839
ConvNeXt-Base-LR	0.979	0.974	0.980	0.977	0.968
ConvNeXt-Large-SVM	0.974	0.972	0.975	0.973	0.960
ConvNeXt-Large-RF	0.893	0.897	0.884	0.889	0.835
ConvNeXt-Large-LR	0.976	0.979	0.975	0.977	0.964
ConvNeXt-XLarge-SVM	**0.997**	**0.996**	**0.998**	**0.997**	**0.996**
ConvNeXt-XLarge-RF	0.945	0.947	0.944	0.945	0.916
ConvNeXt-XLarge-LR	0.992	0.994	0.991	0.992	0.988

The best results are shown in bold font.

The confusion matrices for evaluating features extracted using the ConvNeXt-XLarge architecture with different classifiers are shown in Figure 2. The ConvNeXt-XLarge-SVM model (Figure 2a) correctly predicted almost all classes. The ConvNeXt-XLarge-RF model (Figure 2b) mostly misclassified the classes. The ConvNeXt-XLarge-LR model (Figure 2c) showed limited confusion in some classes.

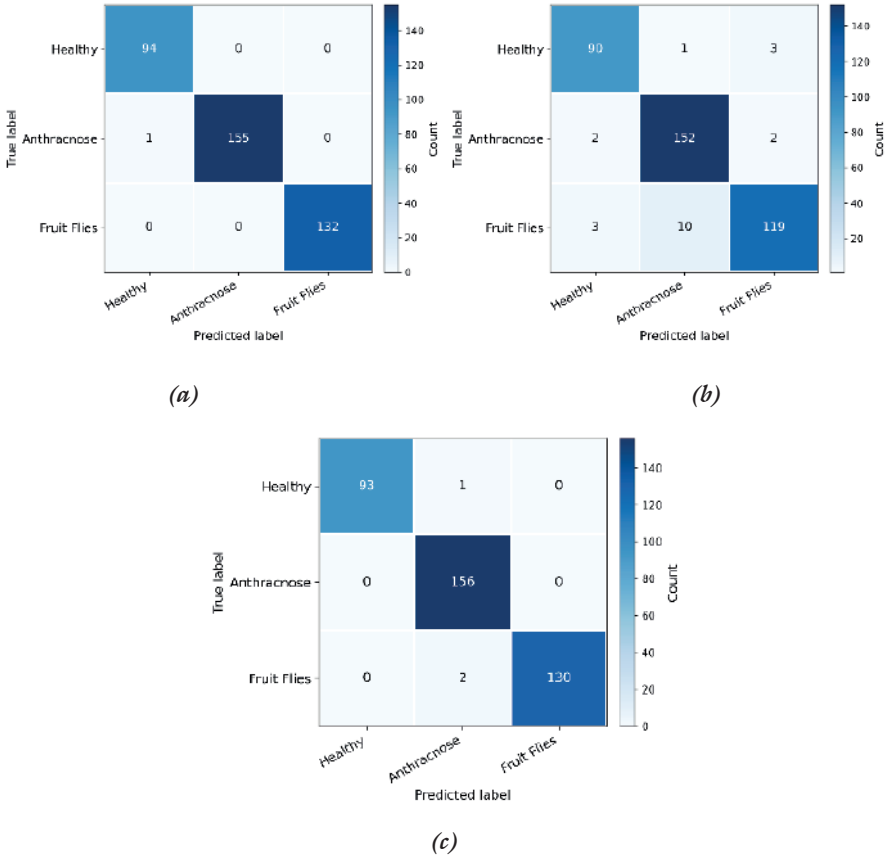


Figure 2. Confusion matrices for ConvNeXt-XLarge-based models, (a) ConvNeXt-XLarge-SVM, (b) ConvNeXt-XLarge-RF, (c) ConvNeXt-XLarge-LR

4.2. Combining Selected Features from ConvNeXt Variants and Classifying Them with Machine Learning Algorithms

The performance results of classifying the vectors obtained by combining the subsets determined by applying feature selection to ConvNeXt-based deep features using SVM, RF, and LR are reported in Table 4. In this context, 50, 100, and 200 features were selected from each ConvNeXt variants. According to the experimental results, the CS200-ConvNeXt-SVM model achieved the highest performance among all models. This model reached a value of 1.000 in all accuracy, precision, recall, F1-score, and Cohen's Kappa metrics.

When examining SVM-based models, it was observed that increasing the number of features from 250 to 1000 raised the accuracy rate from 0.982 to 1.000. LR-based models ranked second among the groups, with performance

increasing steadily as the number of features increased. In this context, the accuracy of 0.950 in the CS50-ConvNeXt-LR model increased to 0.984 in the CS200-ConvNeXt-LR model. In contrast, the RF algorithm performed worse than other classifiers, and a slight downward trend in accuracy was observed as the number of features increased. However, the fact that Cohen's Kappa values remained above 0.85 in all SVM and LR-based models indicates that the classification successes achieved were not random. In other words, it reveals a high level of consistency between the classifier outputs and the actual labels.

Table 4. Classification performance of ML classifiers using combined feature vectors after feature selection

Model	**Sum Feature counts**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Cohen's Kappa**
CS50-ConvNeXt-SVM	250	0.982	0.982	0.981	0.981	0.972
CS100-ConvNeXt-SVM	500	0.997	0.998	0.997	0.998	0.996
CS200-ConvNeXt-SVM	1000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
CS50-ConvNeXt-RF	250	0.921	0.923	0.920	0.921	0.880
CS100-ConvNeXt-RF	500	0.914	0.914	0.909	0.911	0.867
CS200-ConvNeXt-RF	1000	0.908	0.909	0.904	0.906	0.859
CS50-ConvNeXt-LR	250	0.950	0.947	0.951	0.949	0.924
CS100-ConvNeXt-LR	500	0.971	0.968	0.971	0.969	0.956
CS200-ConvNeXt-LR	1000	0.984	0.984	0.984	0.984	0.976

The best results are shown in bold font.

The CS50-ConvNeXt-SVM model (Figure 3a) made a limited number of misclassifications between classes. In the CS100-ConvNeXt-SVM model (Figure 3b), misclassifications have been significantly reduced, and the distinction between classes has improved markedly. Here, the nearly error-free classification of the Anthracnose class demonstrates that the selected features represent disease-specific visual patterns more effectively. However, there is a limited level of error in the fruit fly class. Finally, the confusion matrix for the CS200-ConvNeXt-SVM model in Figure 3c demonstrates perfect classification performance, with all classes correctly identified.

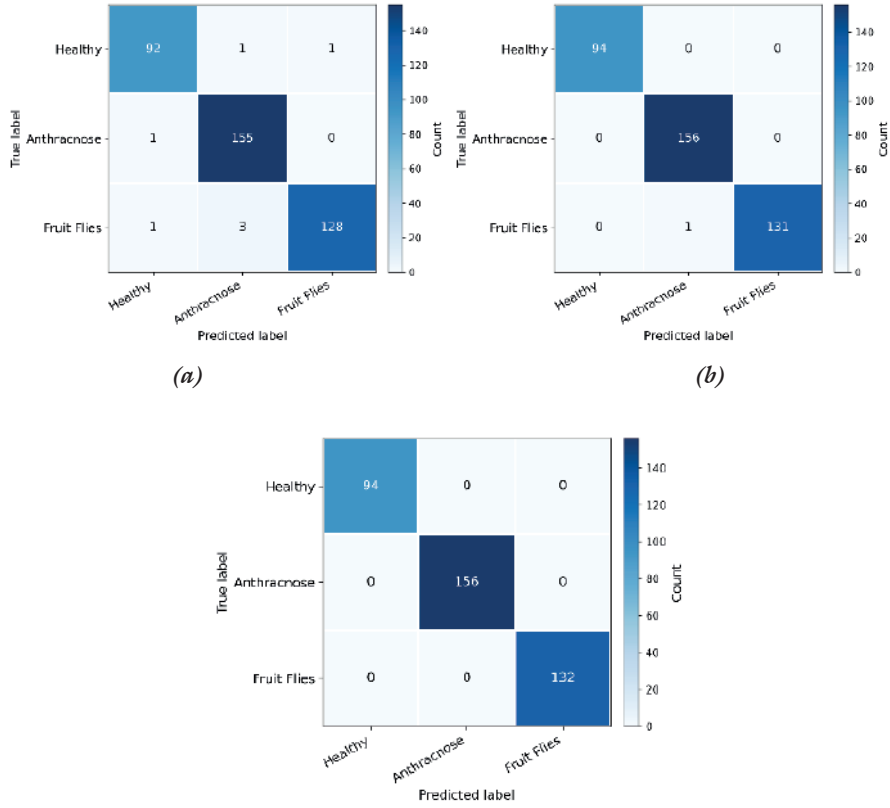


Figure 3. Confusion matrices of models with different feature counts, (a) CS50-ConvNeXt-SVM, (b) CS100-ConvNeXt-SVM, (c) CS200-ConvNeXt-SVM

4.3. Explainable Artificial Intelligence Evaluation of the Proposed Model

Figure 4 presents the class-based average CIU values calculated from correctly classified examples in the CS200-ConvNeXt-SVM model. The top 10 features contributing most to the average CI value for each class have been identified.

Upon examining the results, the F570 feature stood out in the Healthy guava class, with both high average Contextual Importance (CI) and Contextual Utility (CU) values. The F570 was followed by the F861, F952, F925, F896, F761, F529, F449, F520, and F852 features, respectively, based on their average CI values. On the other hand, the features that provide the highest contribution differ in the Anthracnose and Fruit fly classes. In other

words, the proposed model makes decisions based on discriminative and class-specific feature representations for each class. The highest contribution in the Anthracnose class was obtained from the F625 feature, and the highest contribution in the Fruit fly class was obtained from the F626 feature.

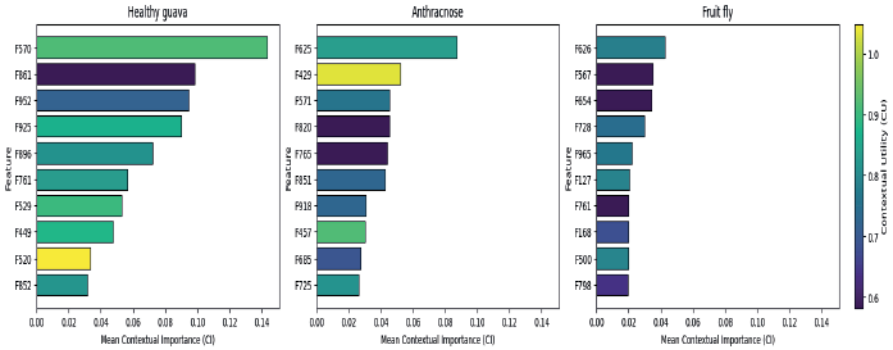


Figure 4. CIU analysis results for the CS200-ConvNeXt-SVM model

4.4. Cross-Dataset Evaluation of the Proposed Model

To evaluate the generalization performance of the proposed model against different data distributions, the results obtained on various publicly available guava datasets are presented in Table 5. The datasets evaluated differ in terms of the number of classes, the number of images, and the content. Both datasets contain images of guava leaves and fruits. The datasets were divided into 70% training and 30% testing ratios. The proposed model achieved high performance in datasets with a relatively high number of images, while it exhibited relatively low performance in smaller datasets. These findings reveal that the model's performance varies depending on the data volume and that it can generalize better in larger datasets.

Table 5. Performance results of the proposed model on different datasets

Dataset	Type	Number of images and classes	Accuracy	Precision	Recall	F1-score
Guava Disease Dataset (Rajbongshi et al., 2022)	Leaf and fruit	4899 images and 5 classes	0.994	0.995	0.994	0.994
Guava Disease Detection Dataset (Shihab et al., 2024)	Leaf and fruit	606 images and 6 classes	0.737	0.746	0.757	0.749

4.5. Discussion

As seen in the studies summarized in Table 6, CNN-based architectures have been predominantly used for the classification of guava fruit and leaf diseases. However, since these studies used different datasets, numbers of classes, and image types (guava fruit or guava leaf), the comparison was limited to the methods used.

Mostafa et al. (2022) achieved 97.74% accuracy with the ResNet-101 model on a dataset of 2889 images across 5 classes. Tewari et al. (2024) obtained 99% accuracy in classifying a 4-class guava disease dataset comprising 681 images using the DenseNet169 model. Hashan et al. (2024) achieved a 93% accuracy rate in classifying guava fruit diseases using the AlexNet model. Kaur et al. (2024) achieved an accuracy rate of 98.76% with their proposed model. Doutoum et al. (2023) achieved 94.93% accuracy in classifying guava leaf diseases using the EfficientNet-B3 model. Nath Nandi et al. (2022) achieved approximately 97% accuracy with their proposed approach. Although most of these studies report high classification accuracies, analyses of the explainability of model decisions remain limited. Mustak Un Nobi et al. (2023) achieved 98% accuracy with their proposed model, which incorporates a Grad-CAM-based XAI approach.

On the other hand, hybrid approaches also show noteworthy performance results in detecting guava diseases. In this context, Kılcı and Koklu (2024) achieved 95.6% accuracy using SqueezeNet and Gradient Boosting models on a dataset of 3784 images across 3 classes. Kılcı and Koklu (2025) reached an accuracy rate of 0.9974 on the same dataset using the InceptionV3-SVM model.

In this study, the ConvNeXt-XLarge-SVM model achieved an accuracy rate of 0.997, while the CS200-ConvNeXt-SVM model, which used 1000 features, achieved an accuracy of 1.000. Furthermore, the features influencing model decisions were analyzed using the CIU method.

Table 6. Comparison of studies in the literature on the classification of guava diseases

Reference	**Dataset structure: number of images**	**Method**	**Use of explainable models**	**Highest accuracy value**
Mostafa et al. (2022)	2889 images and 5 classes	ResNet-101	-	97.74%
Tewari et al. (2024)	681 images and 4 classes	DenseNet169	-	99%
Hashan et al. (2024)	612 images and 5 classes	AlexNet	-	93%
Kaur et al. (2024)	527 images and 5 classes	DenseNet	-	98.76%
Doutoum et al. (2023)	1834 images and 5 classes	EfficientNet-B3	-	94.93%
Nath Nandi et al. (2022)	8525 images and 6 classes	GoogleNet	-	97%
Mustak Un Nobi et al. (2023)	Dataset 1: 1842 images and 5 classes Dataset 2: 527 images and 5 classes	MobileNet	Grad-CAM	98%
Kılcı and Koklu (2024)	3784 images and 3 classes	SqueezeNet and Gradient Boosting	-	95.6%
Kılcı and Koklu (2025)	3784 images and 3 classes	InceptionV3 and SVM	-	0.9974
This study	3784 images and 3 classes	ConvNeXt-XLarge-SVM	-	0.997
This study	3784 images and 3 classes	CS200-ConvNeXt-SVM	CIU	1.000

4.6. Limitations of the Study

The proposed ConvNeXt-based ML approach achieves high performance in classifying guava fruit diseases but also has limitations. These limitations are explained below:

(i) ConvNeXt variants were used only as feature extractors in this study; no end-to-end training evaluations were performed.

(ii) ML algorithms such as SVM, RF, and LR were used in the study, and the hyperparameters of these algorithms were kept fixed.

(iii) This study used a single dataset. The fact that this dataset contains three different classes may limit the model's performance in more complex and multi-class scenarios.

(iv) The performance of the proposed model is evaluated over the training, validation and testing partitions in the 70:20:10 % defined in the original dataset. Strategies such as K-fold or nested cross-validation were not applied in this study.

5. Conclusions and Future Works

In this study, features obtained from ConvNeXt variants (Tiny, Small, Base, Large, XLarge) were classified using ML algorithms (SVM, RF and LR) for the classification of diseases observed in guava fruit. The Guava Disease Dataset, comprising 3784 images across three classes (healthy, anthracnose, fruit fly), was used in the study.

According to the experimental results, the ConvNeXt-XLarge-SVM model achieved the highest performance among the models evaluated. This model achieved values of 0.997, 0.996, 0.998, 0.997, and 0.996 for accuracy, precision, recall, F1-score, and Cohen's Kappa, respectively. On the other hand, the CS200-ConvNeXt-SVM model achieved a value of 1.000 in all metrics. Furthermore, the contribution of the features selected using the CIU method to the CS200-ConvNeXt-SVM model was evaluated on a class-by-class basis. It was determined that the highest contribution in the Healthy guava class came from feature F570, in the Anthracnose class from feature F625, and in the Fruit fly class from feature F626. Furthermore, evaluations across different datasets demonstrated that the model performs well under variable datasets. Future studies may explore the integration of the ViT architectures with various ML algorithms, along with cross-validation strategies and hyperparameter optimization.

Dataset

The Guava Disease Dataset utilized in this study is publicly available on the Mendeley Data repository, and the citation is provided in the reference list. The dataset can be accessed at: https://data.mendeley.com/datasets/bkdkc4n835/1 [07.08.2025]

Rajbongshi et al. (2022). The Guava Disease Dataset [Data set]. Mendeley Data. https://data.mendeley.com/datasets/x84p2g3k6z/1 [25.12.2025]

Shihab et al. (2024). Guava Disease Detection Dataset [Data set]. Mendeley Data. https://data.mendeley.com/datasets/fspx44mwfp/1 [25.12.2025]

References

Ahmed, M., Ahmed, F., Naz, N. S., Mazhar, T., Khan, M. A., Khan, M. A., . . . Abbas, M. (2025). Automated Guava Disease Detection Using Transfer Learning With ResNet-101. *Food Science & Nutrition*, *13*(12), e71348.

Almadhor, A., Rauf, H. T., Lali, M. I. U., Damaševičius, R., Alouffi, B., Alharbi, A. (2021). AI-driven framework for recognition of guava plant diseases through machine learning from DSLR camera sensor based high resolution imagery. *Sensors*, *21*(11), 3830.

Almutiry, O., Ayaz, M., Sadad, T., Lali, I. U., Mahmood, A., Hassan, N. U., Dhahri, H. (2021). A novel framework for multi-classification of guava disease. *Computers, Materials & Continua*, *69*(2), 1915–1926.

Amin, M. A., Mahmud, M. I., Rahman, A. B. P., Mst Aktarina, Mamun, M. A. A. (2024). *Guava Fruit Disease Dataset* Version (V1). https://data.mendeley.com/datasets/bkdkc4n835/1

An, Y., Yi, Y., Han, X., Wu, L., Su, C., Liu, B., . . . Li, Y. (2024). A hybrid attention-guided ConvNeXt-GRU network for action recognition. *Engineering Applications of Artificial Intelligence*, *133*, 108243.

Arnal Barbedo, J. G. (2013). Digital image processing techniques for detecting, quantifying and classifying plant diseases. *SpringerPlus*, *2*(1), 1–12. https://doi.org/https://doi.org/10.1186/2193-1801-2-660

Asim, M., Ullah, S., Razzaq, A., Qadri, S. (2023). Varietal discrimination of guava (Psidium guajava) leaves using multi features analysis. *International Journal of Food Properties*, *26*(1), 179–196.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Cortes, C., Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *20*(2), 215–232.

Doutoum, A., Eryiğit, R., Tuğrul, B. (2023). Classification of guava leaf disease using deep learning. *World Scientific and Engineering Academy and Society (WSEAS)*, *20*(38).

Gao, D., Li, P., Wang, M., Liang, Y., Liu, S., Zhou, J., . . . Zhang, Y. (2023). CSF-GTNet: A novel multi-dimensional feature fusion network based on Convnext-GeLU-BiLSTM for EEG-signals-enabled fatigue driving detection. *IEEE Journal of Biomedical and Health Informatics*, *28*(5), 2558–2568.

Hashan, A. M., Rahman, S. M. T., Avinash, K., Ul Islam, R. M. R., Dey, S. (2024). Guava fruit disease identification based on improved convolutional neural network. *International Journal of Electrical & Computer Engineering (2088-8708)*, *14*(2).

Hastie, T. (2009). The elements of statistical learning: data mining, inference, and prediction. In: springer.

Hosmer Jr, D. W., Lemeshow, S., Sturdivant, R. X. (2013). *Applied logistic regression*. John Wiley & Sons.

Hunter, M. C., Smith, R. G., Schipanski, M. E., Atwood, L. W., Mortensen, D. A. (2017). Agriculture in 2050: recalibrating targets for sustainable intensification. *Bioscience*, *67*(4), 386–391.

Joseph, B., Priya, M. (2011). Review on nutritional, medicinal and pharmacological properties of guava (Psidium guajava Linn.). *International Journal of pharma and bio sciences*, *2*(1), 53–69.

Kanimalar, C., Karthikeyan, M. (2025). Banana disease detection using Swin Transformer and ConvNeXt-Tiny architecture with Spotted Hyena Optimizer for optimized classification performance. *Engineering Research Express*, *7*(4), 045249.

Kaur, A., Kukreja, V., Upadhyay, D., Aeri, M., Sharma, R. (2024). Multi-class Guava Disease Classification using an Efficient and Fine-Tuned DenseNet model. 2024 IEEE 9th International Conference for Convergence in Technology (I2CT),

Kılcı, O., Koklu, M. (2024, 13–14 Dec 2024). *Classification Of Guava Diseases Using Features Extracted From Squeezenet With Adaboost And Gradient Boosting* 4th International Conference on Frontiers in Academic Research (ICFAR), Konya, Turkey.

Kılcı, O., Koklu, M. (2025). Guava fruit disease classification using deep learning and machine learning models. *Research in Agricultural Sciences*, *56*(3), 217–226.

KP, A. R., Gowrishankar, S. (2023). Convnext-based mango leaf disease detection: Differentiating pathogens and pests for improved accuracy. *International Journal of Advanced Computer Science and Applications*, *14*(6).

Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.

Lekkala, C. H., Bharatula, S., Sonika, C., Kannaiah, K. (2025). ConvNeXt-Small: An Automated Deep Learning Method for Detecting Papaya Leaf Disease Using the BDPapayaLeaf Dataset. 2025 7th International Conference on Intelligent Sustainable Systems (ICISS),

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., Xie, S. (2022). A convnet for the 2020s. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition,

Mirvakhabova, L., Pukalchik, M., Matveev, S., Tregubova, P., Oseledets, I. (2018). Field heterogeneity detection based on the modified FastICA RGB-image processing. Journal of Physics: Conference Series,

Mohanty, S. P., Hughes, D. P., Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in plant science*, *7*, 1419.

Mostafa, A. M., Kumar, S. A., Meraj, T., Rauf, H. T., Alnuaim, A. A., Alkhayyal, M. A. (2022). Guava disease detection using deep convolutional neural networks: A case study of guava plants. *Applied Sciences*, *12*(1), 239.

Mustak Un Nobi, M., Rifat, M., Mridha, M., Alfarhood, S., Safran, M., Che, D. (2023). Gld-det: Guava leaf disease detection in real-time using lightweight deep learning approach based on mobilenet. *Agronomy*, *13*(9), 2240.

Nan, B., Liu, F., Qian, X., Song, W. (2025). A Visual Self-attention Mechanism Facial Expression Recognition Network beyond Convnext. *arXiv preprint arXiv:2504.09077*.

Nath Nandi, R., Haque Palash, A., Siddique, N., Zilani, M. G. (2022). Device-friendly Guava fruit and leaf disease detection using deep learning. *arXiv e-prints*, arXiv: 2209.12557.

Noda, R., Hashimoto, T. (2025). Emotion Estimation Using ConvNeXt Based on Feature Extraction from Imaged MFCC. 2025 7th International Conference on Robotics and Computer Vision (ICRCV),

Paramesha, K., Jalapur, S., Hanok, S., Puttegowda, K., Manjunatha, G., Kumara, B. (2025). Machine Learning and Deep Learning Approaches for Guava Disease Detection. *SN Computer Science*, *6*(4), 361.

Rajbongshi, A., Sazzad, S., Shakil, R., Akter, B., Sara, U. (2022). *Guava Leaves and Fruits Dataset for Guava Disease Recognition through Machine Learning and Deep Learning* Version (V1). https://doi.org/10.17632/x84p2g3k6z.1

Roshanzadeh, F., Barati, H., Barati, A. (2025). Lightweight Intrusion Detection System Using a Hybrid CNN and ConvNeXt-Tiny Model for Internet of Things Networks. *arXiv preprint arXiv:2509.06202*.

Shabrina, N. H., Gunawan, D., Ham, M. F., Harahap, A. S. (2023). Papillary Thyroid Cancer Histopathological Image Classification Using Pretrained ConvNeXt Tiny and Grad-CAM Interpretation. 2023 IEEE 11th Joint International Information Technology and Artificial Intelligence Conference (ITAIC),

Shihab, M. M. R., Saim, N. I., Mojumdar, M. U. (2024). *Image Dataset for Detection and classification of Diseases of Guava Fruits and Leaves* Version (V1). https://doi.org/10.17632/fspx44mwfp.1

Solmaz, Ö. A., Tasci, B. (2025). ViSwNeXtNet Deep Patch-Wise Ensemble of Vision Transformers and ConvNeXt for Robust Binary Histopathology Classification. *Diagnostics*, *15*(12), 1507.

Tewari, V., Azeem, N. A., Sharma, S. (2024). Automatic guava disease detection using different deep learning approaches. *Multimedia Tools and Applications*, *83*(4), 9973–9996.

Machine Learning in Computer Science: Concepts, Hybrid Methods, and Spiking Neural Networks

Editors:
Asst. Prof. Dr. Ülker BAŞAR
Asst. Prof. Dr. İbrahim ÖZTÜRK

ÖZGÜR PRESS