mathematics

MDPI

Review

Understanding Machine Learning Principles: Learning, Inference, Generalization, and Computational Learning Theory

Ke-Lin Du [1], Rengong Zhang [2], Bingchun Jiang [1,*], Jie Zeng [3] and Jiabin Lu [4]

[1] School of Mechanical and Electrical Engineering, Guangdong University of Science and Technology, Dongguan 523668, China; kldu@gdust.edu.cn
[2] Zhejiang Yugong Information Technology Co., Ltd., Changhe Road 475, Hangzhou 310002, China; zrg@ygwjg.com
[3] Shenzhen Feng Xing Tai Bao Technology Co., Ltd., Shenzhen 518063, China; zj@fxtb.cc
[4] Faculty of Electromechanical Engineering, Guangdong University of Technology, Guangzhou 510006, China; lujiabin@gdut.edu.cn
* Correspondence: jiangbingchun@gdust.edu.cn

Abstract: Machine learning has become indispensable across various domains, yet understanding its theoretical underpinnings remains challenging for many practitioners and researchers. Despite the availability of numerous resources, there is a need for a cohesive tutorial that integrates foundational principles with state-of-the-art theories. This paper addresses the fundamental concepts and theories of machine learning, with an emphasis on neural networks, serving as both a foundational exploration and a tutorial. It begins by introducing essential concepts in machine learning, including various learning and inference methods, followed by criterion functions, robust learning, discussions on learning and generalization, model selection, bias–variance trade-off, and the role of neural networks as universal approximators. Subsequently, the paper delves into computational learning theory, with probably approximately correct (PAC) learning theory forming its cornerstone. Key concepts such as the VC-dimension, Rademacher complexity, and empirical risk minimization principle are introduced as tools for establishing generalization error bounds in trained models. The fundamental theorem of learning theory establishes the relationship between PAC learnability, Vapnik–Chervonenkis (VC)-dimension, and the empirical risk minimization principle. Additionally, the paper discusses the no-free-lunch theorem, another pivotal result in computational learning theory. By laying a rigorous theoretical foundation, this paper provides a comprehensive tutorial for understanding the principles underpinning machine learning.

Keywords: PAC learning theory; empirical risk minimization; computational learning theory; model selection; universal approximation; Turing-complete

MSC: 68Q32; 68T99

check for updates

Academic Editors: Evgeny Nikulchev and Ravil Muhamedyev

Received: 31 December 2024
Revised: 20 January 2025
Accepted: 26 January 2025
Published: 29 January 2025

Citation: Du, K.-L.; Zhang, R.; Jiang, B.; Zeng, J.; Lu, J. Understanding Machine Learning Principles: Learning, Inference, Generalization, and Computational Learning Theory. *Mathematics* **2025**, *13*, 451. https://doi.org/10.3390/math13030451

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Machine learning is now the most successful and mainstream artificial intelligence (AI) approach. It is the most widely investigated and used approach in almost all disciplines and fields. There are numerous machine learning approaches and models for selection and integration, and the wide applications of machine learning have introduced far-reaching impacts on human kind.

Artificial neural networks (ANNs), encompassing multilayer perceptrons (MLPs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs), have

gained substantial popularity and success. Inspired by biological neural systems, the development of ANNs can be traced back to early works such as McCulloch (1943) [1] and Rosenblatt (1958) [2]. The effectiveness of ANNs is based on representation theorems that guarantee their capacity to approximate complex functions with high accuracy when properly parameterized.

Spiking neural networks (SNNs) [3] provide a biologically inspired alternative to traditional ANNs, using components like dendrites and synapses to model neural dynamics. SNNs excel in unsupervised learning with mechanisms like spike-timing dependent plasticity (STDP) and Hebbian learning, enabling distributed and online learning. Their sparse activity reduces computational demands, making them ideal for edge intelligence and neuromorphic chip designs.

While ANNs are suited for computationally intensive tasks, SNNs are energy-efficient and effective for binary coding and sparse models. Brain-inspired computing [4], driven by SNNs, advances AI by improving models, algorithms, hardware, and applications. Neuromorphic computing, a core aspect of brain-inspired computing, uses very large scale integration (VLSI) technology to replicate biological systems.

The event-driven dynamics of SNNs enhance energy efficiency and biological plausibility, offering an alternative to ANN-based deep learning. As brain-inspired computing evolves with deep learning, it emerges as a promising, energy-efficient AI paradigm.

Machine learning always goes hand-in-hand with statistics, and the border between the two is blurred. Likewise, the boundary between machine learning and neural networks is also blurred. Neural networks are usually treated as models that model the human or animal neural systems, but many neural network models do not actually align with the actual neural systems. Thus, neural networks can be described as those models that can be represented by graphs and adaptively adjust their model parameters. Neural networking is a field of machine learning. Some traditional machine learning models, such as support vector machine (SVM) [5], clustering [6], k-nearest neighbors (k-NN) [7], and ensemble learning [3], can also be represented by graphs and adaptively adjusted. Traditionally, statistical models such as Gaussian process, principal component analysis (PCA), logistic regression, canonical correlation analysis (CCA), and nonnegative matrix factorization (NMF) can all be represented by graphs, and the model parameters can be solved by adaptation. Therefore, machine learning, statistics, and neural networks are sometimes deemed synonyms in the AI or machine learning community. The present AI era started with the successful implementations of deep learning, which belongs to neural networks.

The literature on machine learning is extensive, covering various models, their improvements, applications, and numerous survey and tutorial papers on specific areas. However, there remains a notable scarcity of works focusing on the fundamental principles and theoretical foundations of machine learning. Our thorough literature review reveals that no comprehensive tutorial or survey has systematically covered the foundational principles of machine learning in the last two decades. To the best of our knowledge, this paper is the first such comprehensive survey in twenty years, specifically dedicated to the key principles and foundational theories of the field, with a particular emphasis on neural networks.

This paper presents the core theories of machine learning, with a focus on neural networks, and explores various learning and inference methods, as well as classical and computational learning theories. While practical problem-solving applications are not the primary focus, the paper positions machine learning as a universal statistical and mathematical framework that models human brain functions across multiple levels, encompassing learning, generalization, and inference for tasks such as classification, regression, memory, and optimization.

The research papers referenced in this work were collected through a systematic process informed by the extensive academic expertise of the first author, who has authored three textbooks on machine learning since 2006 [3,7,8]. Over the years, the authors have diligently reviewed every issue of leading journals, including *IEEE Transactions on Neural Networks and Learning Systems*, *Neural Networks*, *Journal of Machine Learning Research*, *Neurocomputing*, *Machine Learning*, *Neural Computation*, *IEEE Transactions on Cybernetics*, *Artificial Intelligence Review*, and other top-tier machine learning publications. In addition, the authors consistently monitor proceedings from prominent conferences such as ICML, NeurIPS, AAAI, CVPR, IJCAI, and ICLR to identify influential and foundational contributions to the field. For this work, over one thousand papers were systematically collected and rigorously analyzed. Only those offering original contributions, significant theoretical advancements, or the latest developments were selected. These papers were then categorized by topic, forming the basis of this paper, which seeks to provide readers with a comprehensive overview of the machine learning discipline.

The structure of this paper is as follows. The first part focuses on outlining the fundamental methods and concepts related to learning and inference. In Section 2, we introduce many learning and inference methods that are related logical and scientific reasoning. In Section 3, many common criterion functions for machine learning are introduced, and those criterion functions for robust learning are also introduced. Section 4 treats a wide range of topics regarding learning and generalization. Section 5 is on model selection by cross-validation or using complexity criteria. Section 6 discusses the bias and variance trade-off. Section 7 deals with overparameterization and double descent. Section 8 discusses the universal approximation capability of various neural network models. The second part of this paper relates to computational learning theory. In Section 9, an introduction to computational learning theory is presented, where the no-free-lunch theory is introduced. From Sections 10–13, PAC learning, the VC-dimension, Rademacher complexity, and the empirical risk-minimization principle, respectively, are introduced. Finally, Section 14 concludes this paper by presenting a few directions for future investigation.

2. Learning and Inference Methods

Learning plays a critical role in the survival and evolution of living organisms, and it represents a key function of neural networks. Learning rules serve as algorithms designed to identify appropriate weights $\mathbf{W}$ and/or other parameters for the network. The process of training a neural network involves solving a nonlinear optimization problem aimed at identifying the parameter set that minimizes a cost function based on provided examples. This parameter estimation process is commonly referred to as a learning or training algorithm.

Neural networks are generally trained over multiple epochs. An epoch refers to a full cycle where the network processes all the training examples using the learning algorithm exactly once. Upon completion of training, the neural network has captured a complex mapping and demonstrates generalization capabilities. To manage the training process, a stopping criterion is established to determine when to halt the process. The algorithm's complexity is often represented as $O(m)$, where m refers to the order, representing the number of floating-point operations required.

Transduction, in both logic and statistical inference, refers to reasoning from observed specific instances (training data) to new, specific instances (test data). In contrast, induction involves generalizing from the training data to form general principles or rules that can be applied to new cases. Machine learning can be broadly categorized into inductive learning and transductive learning. Inductive learning is aimed at the standard machine learning objective of accurately classifying the entire input space. On the other hand,

transductive learning targets a predefined set of unlabeled data, with the goal of labeling this particular set.

Inductive inference involves estimating a model function by examining the relationship between the data and the entire hypothesis space. This model is then used to predict output values for examples that are not part of the training set. Numerous machine learning techniques, including SVMs, neural networks, and neuro-fuzzy models, fall into this category.

In contrast, transductive learning, or transductive inference, focuses on predicting model functions specifically for test cases by incorporating supplementary information from the training dataset related to the new instances [9].

2.1. Scientific Reasoning

Scientific reasoning can generally be divided into three categories: deduction, induction, and abduction. Deduction is concerned with conclusions that are necessarily true, induction deals with conclusions that are likely true, and abduction involves conclusions that are plausibly true, though not certain.

Deductive reasoning moves from a cause to derive its effects or consequences. It involves analyzing premises, which consist of a general rule and a specific case, resulting in a necessary truth. Inductive reasoning, on the other hand, allows for inferring potential causes based on a given consequence, formulating a rule based on a specific instance and its result, though the conclusion is only likely, not guaranteed. Abductive reasoning, nevertheless, infers a cause based on a rule and an observation (result), producing a conclusion that is plausible but hypothetical.

Syllogistic reasoning serves as a formal framework for deduction. In syllogistic inference, the conclusion logically follows from both the major and minor premises. Reasoning can progress through successive syllogistic inferences, using transitive closure (deduction), generalization (induction), and experimental validation (abduction). These reasoning forms are interconnected, and their varying degrees of truth can be understood as different levels of belief [10]. The differences in these beliefs stem from how the premises are converted in syllogistic processing.

2.1.1. Deductive Reasoning

In deductive reasoning (also referred to as top-down logic), a conclusion is derived by applying universally accepted principles within a closed system of discourse. The reasoning process progressively narrows the scope of consideration, leaving only the conclusion. The outcome of a deductive argument is always certain.

One of the primary forms of deductive reasoning is the law of detachment, also called affirming the antecedent or modus ponens, which is a Latin term meaning "the way that affirms by affirming". In propositional logic, modus ponens is a rule of inference that states: "If P implies Q ($P \rightarrow Q$), and P is asserted as true, then Q must also be true".

The law of syllogism involves combining two conditional statements to form a conclusion by joining the hypothesis of one statement with the conclusion of another. For example, if $P \rightarrow Q$ and $Q \rightarrow R$, then it follows that $P \rightarrow R$.

In propositional logic, modus tollens (Latin for "the way that denies by denying") is a valid form of reasoning that denies the consequent. It follows from the principle that if a statement holds true, its contrapositive must also hold.

The contrapositive law asserts that, in a conditional statement, if the conclusion is false, the hypothesis must also be false. For example, from $P \rightarrow Q$, if $\neg Q$ is true, then $\neg P$ must also be true.

Any application of modus tollens can be transformed into an application of modus ponens or into a transposition of the premise, which represents a material implication. Similarly, each instance of modus ponens can be transformed into modus tollens or a transpositional form.

2.1.2. Inductive Reasoning

In inductive reasoning (often referred to as bottom-up logic), conclusions are drawn by generalizing or extrapolating from specific instances to broader principles, which inherently involves epistemic uncertainty. However, it is important to distinguish inductive reasoning from mathematical induction, which is a form of deductive logic used in mathematical proofs.

Inductive learning is a subset of supervised learning methods, where the goal is to derive a hypothesis $h(\mathbf{x}_i)$ that approximates $f(\mathbf{x}_i)$ for all i, given a set of input–output pairs $\mathbf{x}_i, f(\mathbf{x}_i)$. In inductive learning, the task is to create a concept that can accurately categorize most positive instances and exclude negative ones, based on a collection of training examples. This process necessitates a sufficient quantity of training instances to effectively develop a generalizable concept.

2.1.3. Abductive Reasoning

Abductive reasoning, sometimes referred to as abduction, abductive inference, or retroduction, is a type of logical reasoning that begins with an observed phenomenon and aims to identify the most straightforward and plausible explanation by tentatively accepting a hypothesis. The outcomes of the hypothesis can be subjected to empirical verification. Unlike deductive reasoning, the premises do not guarantee the conclusion in abduction, which makes it a process of inferring the most probable explanation.

The scientific approach to answering questions is depicted in Figure 1. This method consists of three distinct phases [11]. In the first phase, labeled *question in search of answers*, the analyst generates potential hypotheses as possible answers. The second phase, labeled *hypothesis in search of evidence*, involves testing these hypotheses to find supporting evidence. In the third phase, *evidentiary assessment of hypotheses*, the likelihood of each hypothesis is assessed based on the gathered evidence. As new evidence is uncovered, it may lead to revised answers to the question, prompting the discovery of additional evidence and the reassessment of probabilities. The process incorporates abductive, deductive, and inductive reasoning. Abductive reasoning is used in generating hypotheses, suggesting what might be true. Hypothesis-driven evidence discovery employs deductive reasoning, showing what must be true. Hypothesis assessment utilizes inductive reasoning to determine what is likely to be true.

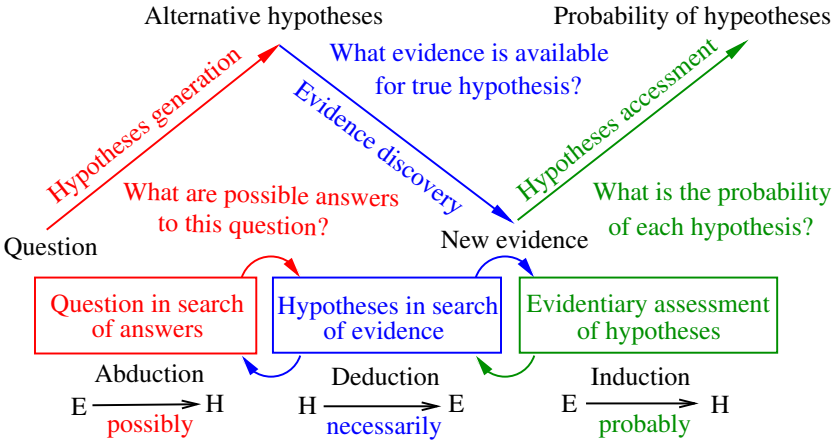


Figure 1. The process of reasoning used to answer a question.

2.1.4. Analogical Reasoning

In contrast to inductive learning, analogical learning can be based on just a single instance. For example, by knowing that the plural form of `fungus` is `fungi`, one can deduce that the plural of `bacillus` should be `bacilli`.

2.1.5. Case-Based Reasoning

Case-based reasoning, along with knowledge generalization, is a primary method that utilizes prior experiences. The focus of case-based reasoning is on retrieving the most relevant case(s) from a database of past experiences to assist with learning in the current task.

2.1.6. Ontologies

Ontologies are structured frameworks that categorize data objects into classes and define the relationships between these classes. They allow for flexible classification, where an object can belong to multiple classes simultaneously. The most basic form of ontology involves classification, where each class has a single parent class. The assignment of an object to a class is governed by rules, which are programmed into the system, defining the relationships of classes, subclasses, and superclasses.

2.2. Supervised, Unsupervised, and Reinforcement Learning

Learning techniques are commonly categorized into three types: supervised, unsupervised, and reinforcement learning. In these methods, $\mathbf{x}_p$ and $\mathbf{y}_p$ represent the input and output for the pth instance in the training data, $\hat{\mathbf{y}}_p$ represents the predicted output by the neural network for the pth input, and E denotes the error function. Statistically, unsupervised learning aims to model the probability density function (pdf) of the input data, $p(\mathbf{x})$, whereas supervised learning focuses on the conditional pdf, $p(\mathbf{y}|\mathbf{x})$. Supervised learning finds widespread use in areas like classification, regression, modeling, control systems, signal processing, and optimization. In contrast, unsupervised learning is typically employed in tasks such as vector quantization, clustering, feature extraction, signal encoding, and data analysis. Reinforcement learning, however, is predominantly used in control systems and various artificial intelligence applications

2.2.1. Supervised Learning

Supervised learning involves learning a mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$ from a provided dataset $\{(\mathbf{x}_i, y_i), i = 1, 2, \ldots, N\}$, where each input instance $\mathbf{x}_i \in \mathcal{X}$ is associated with a corresponding label $y_i \in \mathcal{Y}$.

In this learning paradigm, the parameters of the model are adjusted by comparing the actual output of the network with the desired target output. This process functions as a closed-loop system, where the feedback signal is represented by the error. The error metric quantifies the disparity between the output generated by the network and the expected result from the training data, guiding the learning process. Typically, the error is measured using the mean squared error (MSE) as follows:

$$E = \frac{1}{N} \sum_{p=1}^{N} \left\| \mathbf{y}_p - \hat{\mathbf{y}}_p \right\|^2, \tag{1}$$

where N denotes the total number of training patterns, $\mathbf{y}_p$ is the actual output for the pth training instance, and $\hat{\mathbf{y}}_p$ is the network's predicted output for the same instance. The error E is recalculated at the end of each epoch. The learning process concludes once E falls below a predefined threshold or when a failure condition is triggered.

To minimize E and drive it towards zero, gradient descent is commonly used. This method guarantees convergence to a local minimum within a region near the initial network parameter values. The most commonly used gradient-descent algorithms are the least mean squares (LMS) and backpropagation (BP) algorithms. Second-order methods, on the other hand, involve calculating the Hessian matrix for optimization.

2.2.2. Unsupervised Learning

Unsupervised learning operates without predefined target values, aiming to uncover meaningful patterns or structures in input data by leveraging relationships between features. This approach reduces data dimensionality or volume, and is particularly suited for biological systems due to its reliance on intuitive mechanisms such as neural competition and cooperation, without requiring external supervision.

A stopping criterion is crucial to control the learning process, ensuring it halts appropriately and does not continuously adapt to evolving patterns outside the training set. Common unsupervised learning methods include Hebbian learning, competitive learning, and the self-organizing map (SOM), with these approaches typically converging more slowly to stable states.

- Hebbian learning: A local process involving two neurons and a synapse, where weight adjustment is proportional to the correlation observed between pre- and postsynaptic activities. It underpins models for PCA and associative memory.
- Competitive learning: Neurons compete to be the most responsive to a given input. The SOM, a form of competitive learning, relates closely to clustering techniques.
- Boltzmann machine: employs stochastic training via simulated annealing, inspired by thermodynamics, to learn in an unsupervised manner.

These methods highlight unsupervised learning's capacity to extract structure and adapt in complex environments.

2.2.3. Reinforcement Learning

Reinforcement learning encompasses computational techniques that guide an artificial agent, whether a physical or simulated robot, to select actions that maximize its cumulative expected reward over time [12]. The difference computed for this purpose, known as the reward prediction error, has been found to closely align with the activity of dopamine-releasing neurons in the substantia nigra of nonhuman primates [13].

This type of learning is a variation of supervised learning, where the exact desired outcome is not provided. Instead of detailed guidance, the agent receives feedback on whether its actions were successful or not. This approach is considered more aligned with human cognition, where fully specified correct answers are often unavailable to both the learner and the teacher. In reinforcement learning, the agent is evaluated based on whether the actual outcome is close to the expected result. The agent is rewarded for successful actions and punished for unsuccessful ones, with no need for explicit derivative computations. However, this makes reinforcement learning a slower process. In control systems, if the controller produces a correct output in response to an input, it is considered a good result; otherwise, it is deemed bad. The binary evaluation of this output, known as external reinforcement, serves as the error signal for the learning process.

2.3. Semi-Supervised Learning and Active Learning

In domains such as bioinformatics, text categorization, web and text mining, spam detection, face recognition, and video indexing, vast amounts of unlabeled data can be collected quickly and inexpensively. However, manually labeling these data are often labor-intensive, costly, and susceptible to errors. When the availability of labeled samples

is limited, the inclusion of unlabeled samples can help mitigate the risk of performance degradation caused by overfitting.

Semi-supervised learning aims to leverage a vast amount of unlabeled data alongside a small set of labeled examples to enhance generalization. Some methods in this domain rely on assumptions connecting the probability distribution $P(x)$ with the conditional distribution $P(Y = 1|X = x)$. This learning paradigm shares similarities with transductive learning. Semi-supervised learning is grounded in two central principles: the cluster assumption [9] and the manifold assumption [14]. The cluster assumption posits that points within the same cluster tend to have identical labels, with transductive SVM [9] serving as a significant illustration.

Universum data consist of unlabeled examples that do not belong to any of the classes of interest in a classification task. A contradiction occurs when two functions from the same equivalence class produce differing outputs on a Universum sample. Unlike semi-supervised learning and transductive learning [9], Universum learning operates with Universum data originating from a distribution distinct from that of the labeled training data. This approach balances the need to explain training samples (using large margin hyperplanes) and minimize contradictions in Universum data.

In active learning, also known as pool-based active learning, the learner selectively queries the most informative examples for labeling to optimize generalization while minimizing labeling effort [15]. Given a pool of unlabeled samples, active learning selects data points that enhance model performance. Its effectiveness depends on the model architecture and data characteristics. To address model misspecification, methods often rely on the conditional expectation of generalization error and weight training samples by importance [16]. Reinforcement learning can be considered a type of active learning, where a query mechanism actively seeks labels for some of the unlabeled data.

Active learning can be applied in situations such as web searching, email filtering, and relevance feedback systems for databases or websites. The first two applications typically involve induction, aiming to build a classifier that generalizes well to unseen examples. The third scenario illustrates transduction [9], where the learner's performance is evaluated on a set of instances within the same database, instead of using an independent test set.

Query-by-committee is a widely used pool-based active learning approach for classification [17] and regression [18]. It builds a committee of learners from labeled data, often using bootstrapping or diverse algorithms, and selects unlabeled samples where committee disagreement is highest for labeling. Query-by-committee also employs a prior over hypotheses and processes unlabeled data streams, deciding label requests. For data sampled uniformly from a unit sphere in $\mathbb{R}^d$ with labels aligned to a homogeneous linear separator, it achieves generalization error ϵ with $O(\frac{d}{\epsilon}\log(\frac{1}{\epsilon}))$ samples and $O(d\log(\frac{1}{\epsilon}))$ labels, significantly improving on supervised learning sample complexity $O(\frac{d}{\epsilon})$. The method involves random sampling from intermediate hypothesis spaces, with update complexity growing polynomially as the number of updates grows.

An information-theoretic method for active data selection is proposed in [19]. In [20], a two-stage sampling method is presented to reduce both bias and variance, leading to the development of two active learning strategies.

In batch-mode active learning, as outlined in [21], a set of informative examples is selected for labeling in each iteration. The key aspect of this approach is minimizing redundancy among the chosen examples, ensuring that each example contributes distinct information to update the model. The method chooses unlabeled examples that can effectively reduce the Fisher information in the classification model [21].

2.4. Transfer Learning

Domain refers to the state of the world at a given time. Transfer learning, or meta-analysis in statistics [22], models rare events by leveraging data from related cases to improve performance in a target domain using information from a source domain with a different distribution. This concept, based on case-based and analogical learning, suggests that fewer labeled examples are needed compared to independent task learning [23].

When single-source domain knowledge is insufficient, multiple source domains with richer transferable information are used, where multisource domain adaptation considers the differing contributions of each source to the target [22]. Transfer learning, which can involve inductive (e.g., multitask), transductive (e.g., domain adaptation), and unsupervised (e.g., clustering) cases, transfers knowledge from labeled sources to an unlabeled target, involving both single and multiple sources/targets.

To address source–target discrepancies, features are transformed into a latent space, with deep learning methods such as pretrained CNNs [24] extracting robust representations for visual domain adaptation. Transfer learning can be homogeneous, where source and target domains share the same instance space but differ in distribution, or heterogeneous, where domains are from distinct but related spaces. It also includes instance transfer, feature representation transfer, parameter transfer, and relational knowledge transfer.

Knowledge transfer has two main directions [25]: representation transfer [26], which abstracts task characteristics for future problems by collecting shared knowledge across tasks, and parameter transfer [27], which directly transfers parameters between tasks. Strategic knowledge transfer, especially in multiagent reasoning frameworks like intra-agent transfer [28] and ad hoc teamwork [29,30], enables agents to cooperate with new teammates by selecting the best pre-learned policy.

Two challenges are tackled in [31]: transferring learned responses across varying opponent strategies, and reducing game-solving costs. Q-mixing addresses the transfer problem for value-based policies by averaging Q-values, while the mixed-oracles and mixed-opponents algorithms minimize costs by reusing responses and combining prior opponents into novel policies.

A semiparametric additive framework is proposed in [32] to predict target models by transferring parameter information from source models. Using K-fold cross-validation, this approach assigns data-driven weights for adaptive transfer, ensuring optimality and robustness even in misspecified settings, while addressing negative transfer without requiring prior source model knowledge.

Error bounds for a single-source parameter transfer in high-dimensional linear models are established in [33], extending to multi-source learning with minimax optimality under L_q-regularization in [34]. A multi-source framework for generalized linear models is introduced in [35], including methods to identify transferable sources. This framework is applied to Gaussian graphical models in [36], with edge detection procedures ensuring false discovery rate control. These methods rely on parameter similarity, auxiliary source models, and effective tuning parameters.

Transfer learning can harm performance when source models are unrelated to the target, known as negative transfer [37]. Transfer learning for linear models [34] and generalized linear models [35] address this by constructing auxiliary source models based on parameter similarity.

Domain generalization [38] tackles prediction on future unlabeled data with varying distributions. Ref. [39] frames it as a supervised learning problem by augmenting the feature space with marginal distributions, building on [40]. They propose two statistical models: one for arbitrary dependencies, and another for independent and identi-

cally distributed data, alongside a distribution-free kernel machine for consistent learning across frameworks.

Learning to learn (LTL), or meta-learning, uses labeled data from multiple tasks to design a meta-learner that selects optimal algorithms for future tasks [41]. Well-adapted hypothesis classes improve sample efficiency [42]. Knowledge transfer to new tasks has been quantified through structures like feature representations [43,44], priors on predictors [45], and dictionaries [46]. While both LTL and domain generalization aim for task generalization, LTL relies on labeled data and achieves Bayes risk, whereas domain generalization works with unlabeled data. Active learning variants of domain generalization were explored by [23,47], and focused on identifying a common feature space across tasks.

2.5. Other Learning Methods

2.5.1. Ordinal Regression and Ranking

Regression learns the relationship between input variables and continuous output values. Categorical data includes ordinal, with natural ordering, and nominal, with no ordering (e.g., hair color is nominal, service quality assessment is ordinal). Ordinal regression predicts ordinal variables, treating it as a multiclass problem with ordinal constraints, often using numeric values for ordinal scales [48].

Ranking problems, such as those in search engines and online advertising, maintain ordinal relationships through pointwise, pairwise, and listwise methods [49]. Pointwise methods predict relevance scores, pairwise methods (e.g., rankSVM [50]) classify preference pairs, and listwise methods (e.g., LambdaMART [51]) optimize ranking lists. Gradient-boosted decision trees (GBDT) [52] and LambdaMART [51] excel in web search ranking tasks.

Preference learning focuses on binary preference relations [53], ranking input points based on comparisons.

2.5.2. Manifold Learning

Representation learning enables machines to autonomously derive task-relevant representations from raw data, often as unsupervised preprocessing methods like PCA and cluster analysis. Manifold learning focuses on creating low-dimensional representations that preserve the intrinsic geometric structure of data, ensuring close data points in the original space remain close after reduction. Sparse coding [54] adds a sparsity constraint to these representations, while multilinear subspace learning derives low-dimensional representations directly from tensor-based models of multidimensional data.

In deep learning, hierarchical representations are formed, with higher-level features built from lower-level ones. Many dimensionality reduction techniques stem from manifold learning, including:

- Locally Linear Embedding (LLE) [55]: captures global structures of nonlinear manifolds, such as face image datasets.
- Laplacian eigenmaps [56]: maintains local structures using a graph-based representation of data.
- Orthogonal neighborhood-preserving projections [57]: a linear extension of LLE that preserves local geometric relationships.

These methods highlight the versatility of manifold learning in representing complex data geometries.

2.5.3. Multi-Task Learning

Domain refers to the input space $\mathcal{X}$ and its marginal distribution, while tasks refer to the output space $\mathcal{Y}$ and the conditional distribution of $\mathcal{Y}$ given $\mathcal{X}$. Typically, $\mathcal{X}$ and $\mathcal{Y}$

remain consistent across problems, with the joint distribution P_{XY} describing domains and tasks interchangeably. Multi-task learning improves task learning by leveraging distribution similarities [58–60], whereas domain generalization focuses on generalizing to new tasks.

Multiagent learning shares similarities with multi-task learning, where addressing each strategic situation is comparable to solving a distinct task, with the strategies of opponents representing distributions over a common set of tasks [61,62].

The multi-task learning community categorizes knowledge into task-relevant knowledge, specific to a task [63,64], and domain-relevant knowledge, shared across tasks [58,65–67]. Some approaches bridge these, such as using task-specific knowledge as a curriculum [68]. In task-relevant learning, a key method is abstracting irrelevant state information [63,64], with [31] focusing on learning responses to specific opponent policies.

2.5.4. Imitation Learning

Imitation learning, where an agent learns from demonstrations, used in sequential decision-making tasks without predefined reward functions, is applied when designing rewards is difficult or uncertain. It finds optimal policies based on expert demonstrations, with two main approaches: behavioral cloning [69], which suffers from error accumulation, and inverse reinforcement learning [70], which infers a cost function, often through adversarial learning [71]. To enhance sample efficiency, self-supervised representation learning can generate training signals from available data to enhance efficiency [72].

Earlier works frame imitation as independent and identically distributed classification [73] or no-regret online learning [74]. An online imitation learner for stochastic environments [75] eliminates the need for environment resets or stationary dynamics. This approach enables continual learning with finite error bounds, querying only when necessary, with a rapid decrease in query frequency and bounds on unlikely events.

Imitation-active learning ensemble (IALE) [76] proposes a batch-mode, pool-based imitation learning strategy integrating expert heuristics like uncertainty, diversity, and query-by-committee. Using DAgger [74], IALE encodes states and expert actions, balancing exploration and exploitation while distilling expert knowledge into an active learning sampling strategy.

2.5.5. Curriculum Learning

The idea of using learning progress as a reward originates from [77], and is linked to intrinsic motivation [78]. Curriculum learning, introduced by [79], has been applied in contexts like semi-supervised learning, where label propagation benefits from ordering samples from simple to hard [80]. Challenges in curriculum learning include ordering subtasks by difficulty, defining a "mastery" threshold [81], and balancing task difficulty to avoid forgetting [81]. The teacher aims to train the student for a final task using minimal steps, modeled as solving a partially observable Markov decision process (POMDP) [82], with reinforcement and supervised learning formulations. Exploration bonuses [83] help mitigate sparse rewards in Student algorithms.

Deep reinforcement learning has excelled in domains like video games [84], but random exploration's sample complexity grows exponentially with the number of steps required [85]. Curriculum learning [79,81] addresses this by sequencing tasks by difficulty. Recent work includes [86], which proposed a two-agent system for goal generation, and [87], who used generative adversarial networks (GANs) for goal state creation. However, these methods do not explicitly improve learning of the final task.

Teacher–student curriculum learning [88] automates curriculum generation by having the teacher select tasks based on learning progress, prioritizing those with steep positive

slopes and addressing forgetting with negative slopes. It outperforms handcrafted curricula in supervised (decimal addition with long short-term memory (LSTM)) and reinforcement learning (Minecraft navigation) tasks [88]. A similar approach is presented by [89] for supervised sequence learning and some reinforcement learning tasks.

2.5.6. Multiview Learning

Human perception is inherently multimodal, involving sight, hearing, touch, smell, and taste. Similarly, data are often collected from multiple sources, each providing unique, complementary information. For multi-view data, consistency across sources is crucial [90]. Multi-view (or multimodal) learning integrates complementary data from various sources to enhance performance.

Three primary multi-view learning methods are co-training, multiple kernel learning, and subspace learning. Co-training maximizes mutual agreement between two data views [91]. CCA [92] adapts PCA to maximize the correlation between two views in a shared subspace, with multiset CCA [93] and tensor decomposition [94] further enhancing joint analysis. Multiple kernel learning [95] combines kernels from different modalities to improve model performance. Subspace learning discovers a shared latent subspace across views.

Multitask learning leverages domain-specific knowledge across tasks to boost generalization [58], benefiting from additional training signals that act as an inductive bias, akin to multimodal learning.

2.5.7. Multilabel Learning

Multilabel learning encompasses multilabel classification and multilabel ranking tasks. In multilabel classification [96], the goal is to learn a function that assigns multiple labels to each data instance. On the other hand, multilabel ranking involves assigning a real value to each instance, representing its relevance to a label, and generating an ordered ranking based on these values.

A multilabel problem with q labels can be broken down into multiple single-label tasks using one of three common strategies: binary relevance, label powerset, and pairwise methods. Binary relevance applies a one-vs.-all approach, creating q binary classification tasks, though it may struggle with correlated labels. The label powerset method converts the problem into a multiclass classification with 2^q possible labels, accounting for correlations but potentially increasing label space. The pairwise method trains $q(q-1)/2$ classifiers for label pairs, combining them through majority voting.

2.5.8. Multiple-Instance Learning

Multiple-instance learning [97] extends traditional supervised learning, where examples are "bags" of feature vectors. The bag label is determined by a function applied to the labels of its individual instances, often using Boolean OR. A theoretical analysis and PAC-learning algorithm for this approach are presented in [98].

In this framework, bags are labeled positive or negative, with the instances inside lacking individual labels. The goal is to build a classifier that predicts unseen bag labels. Multiple-instance learning is applied in drug activity prediction, as well as image categorization and annotation, particularly in region-based tasks.

Formally, let $\mathcal{X}$ represent the space of instances and $\mathcal{Y}$ the set of possible labels. The goal of multiple-instance learning is to learn a mapping $f : 2^{\mathcal{X}} \to \{-1, +1\}$ from a collection of pairs $\{(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2), \ldots, (\mathbf{X}_m, y_m)\}$, where each $\mathbf{X}_i \subseteq \mathcal{X}$ is a set of instances $\{\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \ldots, \mathbf{x}_{n_i}^{(i)}\}$, with $\mathbf{x}_j^{(i)} \in \mathcal{X}$ for $j \in \{1, \ldots, n_i\}$, and $y_i \in \{-1, +1\}$ is the label of the corresponding bag $\mathbf{X}_i$.

2.5.9. Parametric, Semiparametric, and Nonparametric Classifications

Pattern classification methods for numerical inputs are divided into three categories: parametric, semiparametric, and nonparametric. Parametric and semiparametric classifiers rely on prior knowledge of the data's structure. Parametric methods assume the probability density function (pdf) is known, with parameters estimated from the data, allowing good performance with smaller sample sizes. Parametric methods, such as SVM and logistic regression, are effective when the pdf is well-understood. Semiparametric methods involve models with a fixed, but typically larger, number of parameters that do not grow with sample size.

Nonparametric methods estimate the pdf without prior assumptions on its form, requiring larger sample sizes than parametric methods. These approaches are more flexible, as parametric models are limited by their fixed structure. A common example of a nonparametric method is the *k*-NN classifier.

Methods like density estimation using neural networks or SVMs also belong to nonparametric techniques. The Parzen window method [99] is a nonparametric technique used to estimate the pdf for a finite set of patterns; however, it can be computationally expensive, due to the large number of kernels needed for its representation. Another efficient nonparametric approach is the decision tree, such as C5.0 (http://www.rulequest.com/see5-info.html, accessed on 29 December 2024). This method uses a hierarchical structure and follows a divide-and-conquer strategy, making it suitable for both classification and regression tasks in supervised learning.

2.5.10. Learning from Imbalanced Data

Imbalanced data poses a challenge in machine learning, where one class dominates and the other is underrepresented. This imbalance causes the classifier's decision boundary to favor the majority class, even though the minority class may be more important for the task.

To address this, solutions fall into two categories. Preprocessing methods, such as oversampling, undersampling, and synthetic minority oversampling technique (SMOTE) [100], balance the dataset by adjusting the number of instances in the minority or majority class. These techniques are independent of the classifier used. Algorithm-based approaches modify the learning algorithm itself, using methods like error cost algorithms [101] and class-boundary-alignment [102] to adjust how imbalanced data are handled, often by penalizing misclassifications of the minority class more heavily.

The one-class classification problem, also referred to as anomaly, outlier, or novelty detection, involves identifying patterns that belong exclusively to a target class, with all other patterns treated as nontarget.

2.5.11. Zero-Shot Learning

Zero-shot learning classifies objects from unseen classes using auxiliary information like attributes or word vectors to connect seen and unseen classes [103,104]. It relies on knowledge transfer, where models trained on seen classes generalize to unseen ones. Traditional zero-shot learning assumes only unseen classes in test data, but real-world scenarios include both seen and unseen classes [104]. Generalized zero-shot learning handles both, though it often biases toward seen classes [103]. Methods like direct attribute prediction [104] perform poorly, while frameworks [105] combine zero-shot, generalized zero-shot, and few-shot learning. Generative models [106] synthesize unseen class data, turning generalized zero-shot learning into a supervised task, though imbalanced accuracy remains [106].

Meta-learning supports few-shot learning by focusing on tasks rather than individual data points. It learns meta-knowledge across training episodes, enabling rapid adaptation to new tasks. The consistent meta-regularization method [107] enhances this meta-knowledge, improving model performance.

3. Criterion Functions

MSE is a commonly used metric that penalizes larger errors more heavily, and is optimal for maximum likelihood estimation when feature vectors follow a Gaussian distribution [108]. Alternative metrics, such as mean absolute error or maximum absolute error, may be preferred in specific cases.

A notable alternative to MSE is the logarithmic error function, derived from the Kullback–Leibler divergence. This function offers advantages over MSE [109]. For the tanh activation function, the error for pattern p is defined as

$$E_p(\mathbf{W}) = \frac{1}{2}\sum_{i=1}^{J_M}\left[(1+y_{p,i})\ln\left(\frac{1+y_{p,i}}{1+\hat{y}_{p,i}}\right)+(1-y_{p,i})\ln\left(\frac{1-y_{p,i}}{1-\hat{y}_{p,i}}\right)\right], \tag{2}$$

where $y_{p,i} \in (-1,1)$ is the output of the ith node in the output layer (the Mth layer), and J_M is the number of nodes in the output layer.

For logistic activation functions, it is expressed as [110]

$$E_p(\mathbf{W}) = \frac{1}{2}\sum_{i=1}^{J_M}\left[y_{p,i}\ln\left(\frac{y_{p,i}}{\hat{y}_{p,i}}\right)+(1-y_{p,i})\ln\left(\frac{1-y_{p,i}}{1-\hat{y}_{p,i}}\right)\right], \tag{3}$$

with $y_{p,i} \in (0,1)$. Here, $y_{p,i}$, $\hat{y}_{p,i}$, $1-y_{p,i}$, and $1-\hat{y}_{p,i}$ represent probabilities. These error functions are strictly positive unless $y_{p,i} = \hat{y}_{p,i}$ for all $i = 1,\ldots,J_M$. A simplified version, omitting constants, is provided in [111],

$$E_p(\mathbf{W}) = -\frac{1}{2}\sum_{i=1}^{J_M}\left[y_{p,i}\ln\hat{y}_{p,i}+(1-y_{p,i})\ln(1-\hat{y}_{p,i})\right]. \tag{4}$$

Training neural networks is computationally hard (NP-complete) [112,113], with no algorithm guaranteeing optimal solutions in polynomial time. A single neuron using the logistic function and the MSE criterion may exhibit up to $\left(\frac{N}{J_1}\right)^{J_1}$ local minima, for N training patterns and an input dimension of J_1 [114]. In contrast, the entropic error function creates a convex error landscape with a single minimum, significantly reducing local minima [109,111].

Entropy-based error functions in backpropagation algorithms address flat spots in the error surface without increasing computational costs, shortening training time and lowering local minima density [110]. They are particularly effective for probabilistic data, enabling the learning of hypothesis probabilities from neuron outputs [109,111].

In classification, using hard 0/1 target vectors with backpropagation risks weight saturation and overfitting. Generalization depends more on weight magnitudes than hidden node count [115], and constraining weights helps mitigate overfitting [115].

The cross-entropy loss function, grounded in the maximum likelihood principle, is expressed as

$$E_{\mathrm{CE}} = -\sum_{p=1}^{N}\sum_{c=1}^{C} t_p^c \ln(y_p^c), \tag{5}$$

where C is the number of classes, $t_p^c \in \{0,1\}$ is the target value of the pth pattern in the cth class, and y_p^c is the predicted probability of the pth pattern belonging to the cth class.

Cross-entropy loss enhances convergence and reduces local minima due to its steep gradient [110,111]. Unlike the mean squared error, which exaggerates small output errors, cross-entropy effectively estimates probabilities for rare events [111,116,117]. Networks trained with MSE or cross-entropy approximate posterior probabilities, yielding Bayesian-optimal results in large datasets, though flat regions in the weight space may hinder misclassification minimization.

MSE, assuming Gaussian-distributed targets, is less suitable for discrete labels. However, with sufficient samples and a 1-out-of-C coding scheme, MSE-trained networks can approximate posterior probabilities [116]. Cross-entropy and entropy-based losses are preferred for classification, optimizing probabilistic outputs more effectively. Classification-specific error functions [118] focus on misclassification, propagating gradients only for errors, thereby reducing weight adjustments and mitigating overfitting and saturation risks.

The MSE criterion can be extended using the Minkowski-r metric [119],

$$E_p = \frac{1}{r} \sum_{i=1}^{J_M} \left| \hat{y}_{p,i} - y_{p,i} \right|^r. \tag{6}$$

For $r = 1$, this becomes the city block metric, while $r = 2$ corresponds to MSE. Smaller r values reduce sensitivity to outliers, while larger r values emphasize significant deviations, benefiting low-noise or well-clustered data.

A generalized error function, adaptable via a real-valued parameter, is proposed in [120]. Other robust criteria, such as correntropy [121], handle non-Gaussian noise, while robust statistics [122] and regularization [123] offer further adaptations.

In regression, square loss is common for both training and testing, while classification uses hinge or logistic loss for training and 0-1 loss for testing. Despite the similar performance between square and cross-entropy loss across architectures, datasets, and domains, cross-entropy remains preferred for training modern neural networks [124].

Robust Learning

When training data contains noise or outliers, traditional algorithms may fail due to the disproportionate impact of outliers on MSE, which can lead the network to overfit these points [122]. The Student-t distribution, with heavier tails, is less sensitive to such deviations. Robust statistics [122] provide methods like the M-estimator, which modifies maximum likelihood to handle unknown underlying distributions by replacing squared error with specialized loss functions, reducing the influence of outliers.

The cost function for robust learning algorithms is given by

$$E_r = \sum_{i=1}^{N} \sigma(\epsilon_i; \beta), \tag{7}$$

where $\sigma(\cdot)$ represents a symmetric loss function that achieves its unique minimum at zero, $\beta > 0$ is the cutoff parameter serving as a scale estimator, ϵ_i is the error associated with the ith training sample, and N is the total number of samples in the dataset.

Common loss functions include the logistic function [122], Huber's function [122], Talwar's function [125], and Hampel's tanh estimator [126]. The logistic function is given by

$$\sigma(\epsilon_i; \beta) = \frac{\beta}{2} \ln\left(1 + \frac{\epsilon_i^2}{\beta}\right), \tag{8}$$

and Huber's function is

$$\sigma(\epsilon_i;\beta) = \begin{cases} \frac{1}{2}\epsilon_i^2, & \text{if } |\epsilon_i| \le \beta \\ \beta|\epsilon_i| - \frac{1}{2}\beta^2, & \text{if } |\epsilon_i| > \beta \end{cases}. \tag{9}$$

Using the gradient-descent optimization method, weight updates are computed as

$$\Delta w_{jk} = -\eta \frac{\partial E_r}{\partial w_{jk}} = -\eta \sum_{i=1}^{N} \varphi(\epsilon_i;\beta) \frac{\partial \epsilon_i}{\partial w_{jk}}, \tag{10}$$

where η denotes the learning rate, and $\varphi(\cdot)$, referred to as the influence function, is expressed as

$$\varphi(\epsilon_i;\beta) = \frac{\partial \sigma(\epsilon_i;\beta)}{\partial \epsilon_i}. \tag{11}$$

For the standard MSE function, $\sigma(\epsilon_i) = \frac{1}{2}\epsilon_i^2$ and the corresponding influence function is $\varphi(\epsilon_i;\beta) = \epsilon_i$. To reduce the impact of large errors, robust loss functions are designed so that $\varphi(\epsilon_i;\beta)$ behaves sublinearly.

The τ-estimator [127] is a variant of the M-estimator that integrates an adaptive, bounded influence function $\varphi(\cdot)$. This function is a weighted average of a robust function $\varphi_1(\cdot)$ and an efficient function $\varphi_2(\cdot)$. This estimator provides a high breakdown point while maintaining efficiency under Gaussian errors.

When initial weights are poorly chosen, loss functions may fail to detect outliers effectively, and selecting the parameter β can be challenging. One approach to define β is using the median of absolute deviations (MAD),

$$\beta = c \times \text{median}(|\epsilon_i - \text{median}(\epsilon_i)|), \tag{12}$$

where $c \approx 1.4826$ [122]. Other methods involve utilizing the median of all errors or designating a specific percentage of data points as outliers [122,126].

The C-loss function, or correntropy-induced loss, is a robust method for outlier handling [128]. It is bounded, smooth, differentiable, nonconvex, and consistent with Bayesian principles. The C-loss behaves similarly to the L_2-norm for small errors, approximates the L_1-norm for moderate errors, and closely resembles the L_0-norm for large errors.

4. Learning and Generalization

Learning reconstructs a hypersurface from examples, while generalization estimates values on that surface for unseen points, corresponding to nonlinear curve fitting, interpolation, and extrapolation.

Training neural networks aims to develop statistical models that capture the data's generative process rather than replicate the training data exactly. The mapping reconstruction problem is well-posed if each input has a unique, continuous output, but learning is often an ill-posed inverse problem. Regularization stabilizes the solution with nonnegative functional constraints [123,129].

Overfitting happens when a model achieves high performance on training data but generalizes poorly to new, unseen data, often due to excessive parameters, examples, or epochs. Simpler networks with smoother mappings typically generalize better. Generalization depends on dataset size, problem complexity, and network architecture.

The training set size N must adequately represent the problem, with $N \gg \frac{N_w}{N_y}$ for effective generalization, where N_w is the number of weights, and N_y is the number of outputs [130]. Feature extraction reduces dimensionality, mitigating the curse of dimensionality and enhancing generalization.

4.1. Generalization Error

The generalization error of a trained neural network consists of two components: an approximation error, due to the finite number of parameters and inherent noise in the training data, and an estimation error, caused by the limited data points [131]. For a feedforward network consisting of J_1 input nodes and one output node, the generalization error can be bounded by the hypothesis parameters, N_P, and the number of training examples, N [131],

$$O\left(\frac{1}{N_P}\right) + O\left(\left[\frac{N_P J_1 \ln(N_P N) - \ln \delta}{N}\right]^{1/2}\right), \quad \text{with probability } p > 1 - \delta, \tag{13}$$

where $\delta \in (0,1)$ represents the confidence level, and N_P is the number of parameters. The first term reflects the approximation error, while the second term relates to the estimation error.

As N_P grows, the approximation error reduces, but the estimation error increases due to overfitting. The optimal model size balancing both errors is $N_P \propto N^{\frac{1}{3}}$ [131]. The generalization error for feedforward networks is expected to be $O\left(\frac{1}{N_P}\right)$, consistent with MLPs using sigmoidal functions [132].

Standard gradient methods [133] do not overfit on separable data. After T iterations on a dataset of size N, empirical risk and generalization error both decline at a rate of $O(1/\gamma^2 T)$ until T approaches N, where γ is the margin. Beyond this point, the generalization error stabilizes at $O(1/\gamma^2 N)$. Non-asymptotic bounds on margin violations are also provided in [133].

4.2. Generalization by Stopping Criterion

Generalization can be managed by halting training before reaching the absolute minimum, thereby preventing overfitting. Neural networks trained iteratively incorporate features hierarchically, and stopping at the right moment avoids fitting high-frequency noise, while training error decreases monotonically, generalization error initially declines to a minimum before rising, emphasizing the importance of early stopping [134]. Cross-validation is commonly used to identify the optimal stopping point, with slower criteria offering marginal generalization improvements at the expense of longer training times [134].

For three-layer MLPs, satisfactory generalization occurs when $N > 30N_w$, where N_w is the number of weights [135]. If $N < N_w$, early stopping mitigates overfitting, although risks remain if $N < 30N_w$. Cross-validation aids in selecting the stopping point in such scenarios.

Advanced methods, like the signal-to-noise ratio figure (SNRF) criterion, automatically detect overfitting based on training error alone [136]. Moreover, optimal stopping rules refine error bounds in iterative regularization, with early stopping shown to be equivalent to Tikhonov regularization when aligned with its penalties. Cross-validation-based rules effectively address unknown parameters, achieving comparable generalization errors to traditional methods [137].

4.3. Generalization by Regularization

Regularization enhances generalization by assuming the target function is smooth, where small input variations yield minor output changes. Regularization reduces variance by increasing bias, following the bias–variance trade-off [138]. Effective regularizers minimize variance while limiting bias impact [139].

A constraint term E_c is added to the training cost function E, forming the total cost function,

$$E_T = E + \lambda_c E_c, \tag{14}$$

where $\lambda_c > 0$ balances error minimization and smoothness. Unlike early stopping, regularization applies to both iterative gradient-based methods and direct optimization techniques, such as singular value decomposition (SVD).

Network-pruning techniques, like weight decay, enhance generalization by pruning near-zero weights while retaining significant ones [140,141]. Biases are excluded from penalties to ensure unbiased target mean estimates.

Early stopping also acts as a regularization method, especially with the MSE function [139]. Here, the term $\frac{1}{\eta t}$, where η denotes the learning rate and t is the iteration index, serves as the regularization parameter, with effective non-zero weights increasing during training.

Introducing jitter to input data enhances generalization by serving as a form of smoothing regularization, with the noise variance acting as the regularization parameter [139,141]. This technique helps prevent overfitting in large networks that lack sufficient constraints. Noise injection [140] establishes an equivalence between ridge regularization and Gaussian noise augmentation, linking regularization strength to noise variance. Incorporating noise into training, as proposed by [142], creates an infinite supply of examples, enhancing generalization, accelerating backpropagation, and reducing local minima risks.

Another approach proposed by [143] reduced model complexity by encoding weights with shorter bit lengths, balancing squared error minimization with reduced weight information via Gaussian noise adjustment. Weight sharing ties multiple weights to a single parameter, reducing parameters and improving generalization [144]. Soft weight-sharing, introduced in [145], uses regularization to determine which weights to tie. Additionally, dropout introduces implicit regularization by randomly deactivating intermediate neurons, akin to enforcing sparsity regularization.

Implicit regularization describes how optimization algorithms shape generalization through their preferred solutions. Gradient descent induces L_2-norm regularization but is limited to specific settings. Mirror descent, a generalization of gradient descent, provides a unified framework for controlling implicit regularization in regression and classification [146].

In infinite-dimensional linear models with convex, differentiable loss functions, implicit regularization has been analyzed through high-probability bounds on the excess risk of batch gradient descent iterates [147]. This work shows that optimization itself regularizes learning without explicit penalties, complementing earlier studies addressing statistical and optimization challenges in deep learning [148–150].

Unsupervised pretraining acts as a regularizer for deep learning algorithms [151], with theoretical analysis interpreting it as part of Tikhonov-regularized batch learning methods [152]. The method jointly optimizes predictors and neural network parameters to produce Tikhonov matrices. Learning meaningful matrices through pretraining enhances stability and improves generalization as sample size increases [152].

4.4. Data Augmentation

Data augmentation enhances supervised and self-supervised learning by improving generalization. While traditionally linked to generating data from the same distribution [153,154], recent techniques like randomized masking [155] and cutout [156] challenge this notion by significantly altering distributions [155]. Adding Gaussian noise aligns with Tikhonov regularization [140] and vicinal risk minimization [157,158], linking augmentation to explicit regularization, particularly in underparameterized models, where it reduces variance and mitigates noise overfitting.

In [159], a framework is proposed to analyze augmentation's effects on generalization for underparameterized and overparameterized linear models, drawing from overparame-

terized learning theory [160,161]. Two mechanisms are identified: modifying eigenvalue proportions in the data covariance matrix, and amplifying the spectrum via ridge regression. These explain generalization differences across model complexities and tasks.

In underparameterized models, augmentation reduces variance with minimal bias, enhancing generalization. Overparameterized regression, however, often faces significant bias and distributional shifts, degrading performance. These biases are mitigated under 0-1 loss, improving classification tasks [159].

Randomized augmentation techniques, including random masking [155], cutout [156], noise injection [140], and group-invariant augmentations [153], are examined in [159]. Noise injection induces constant spectral shifts, while masking, cutout, and distribution-preserving augmentations isotropize the data spectrum. Random feature rotation [159] achieves lower bias than least-squares estimators with variance reduction comparable to ridge regression. Stochastic data augmentation, implemented on-the-fly via augmented stochastic gradient descent (SGD) [153,154], addresses augmented empirical risk minimization.

Distinct robustness to augmentation hyperparameters between regression and classification tasks, as well as differences between precomputed [162,163] and online augmentation methods, is highlighted in [159].

Augmentations producing identically distributed samples generally enhance generalization. A group-theoretic framework introduced in [153] emphasizes variance reduction for minimally disruptive augmentations in underparameterized models. Building on this, the analysis of out-of-distribution (OOD) augmentations shows that their spectral effects and biases can either enhance or impair generalization in overparameterized models [159].

A Markov process framework linking Bayes-optimal classifiers to augmentation-dependent kernel classifiers is proposed in [154], identifying data-dependent regularization. Deterministic linear augmentations expand the training span and induce regularization in regression [163], while local augmentations reduce network rugosity [164]. Permutation-style augmentations in two-layer convolutional networks enhance feature learning in classification tasks [162]. Random mask [155], cutout [156], and random rotation augmentation [159] achieve comparable generalization errors across hyperparameters, often outperforming ridge regularization, with random mask [155] removing covariates.

4.5. Dropout

Dropout, introduced by [165], mitigates overfitting in deep networks by randomly removing units and connections during training. This prevents fragile co-adaptations, enhancing generalization. Dropout effectively samples from numerous thinned networks, approximating their combined effect at test time with a single network using scaled weights. Retaining units with a fixed probability p adds noise, acting as regularization. Empirical studies suggest optimal performance by dropping 20% of input units and 50% of hidden units.

Dropout can be viewed as a stochastic regularization method, with a deterministic variant achieved by marginalizing the noise. In the case of linear regression, this regularization is a modified version of L_2 regularization [165]. An extension of dropout, called DropConnect [166], goes further by randomly zeroing elements of the weight matrix $\mathbf{W}$, rather than dropping neuron outputs.

Dropout prevents overfitting by randomly masking neurons during training, with each neuron deactivated based on a Bernoulli distribution. The method is often linked to explicit regularization [167–173].

Several theories explain the success of dropout. Ref. [174] likens it to sexual reproduction in evolution, where offspring inherit a combination of genes, suggesting dropout is most effective at $p = 0.5$. It controls network complexity by limiting weight co-adaptation,

enabling simpler functions in deeper layers [175]. Ref. [168] interprets dropout as an ensemble method combining various network topologies. Ref. [176] frames dropout as approximate Bayesian inference in deep Gaussian processes, minimizing Kullback–Leibler divergence between the approximate and true posteriors.

Dropout with rectified linear unit (ReLU) activation and quadratic loss differs from weight decay regularization [177], preventing co-adaptation of weights. Dropout's penalty grows exponentially with depth, whereas weight decay grows linearly. Additionally, dropout is less sensitive to feature, output, and weight rescaling, suggesting no isolated local minima.

Spectral dropout [178] regularizes by eliminating weak and noisy Fourier coefficients, outperforming traditional techniques. This approach can be seen as a convolutional layer using a fixed basis function for decorrelation.

Uncertainty estimation methods include Bayesian approximations, ensemble approaches, and parametric models. Monte Carlo dropout [176] is a popular Bayesian approximation, with a more efficient variant, last-layer dropout [179]. Parametric uncertainty Monte Carlo [179] uses dropout samples from a parametric model to aggregate predictions for Gaussian parameters. Ref. [176] reinterpret dropout as Bayesian variational inference approximating a deep Gaussian process prior.

Wasserstein dropout [180] is a non-parametric approach for regression, modeling heteroscedastic noise and aleatoric uncertainty by generating sub-network distributions through dropout and minimizing the Wasserstein distance between predicted and true data distributions.

Standard dropout's fixed random activation does not replicate the adaptive neuron activation of the human cerebral cortex. Inspired by gene theory, an adaptive dropout method [181] uses a variational autoencoder to dynamically adjust dropout probabilities, preserving critical features like CNN edges while deactivating less essential neurons. This approach, building on [165], enhances robustness, reduces overfitting, and improves training efficiency by leveraging diverse sub-networks.

Non-asymptotic bounds for gradient descent with dropout in linear regression show that additional randomness persists with fixed learning rates. The gap between dropout and L_2-penalization vanishes in Ruppert–Polyak averages. A simplified dropout model converges to the least-squares estimator without regularization, providing insights into strategies for extending to randomized optimization methods [182].

For two-layer models, marginalizing dropout noise introduces a nuclear norm penalty in matrix factorization [169] and linear neural networks, which generalizes to L_2-path regularizers in deep linear [171] and shallow ReLU networks [167]. Dropout generalization bounds [166,183,184] associate dropout rates with a reduction in Rademacher complexity, while bounds on complexity for shallow ReLU networks are derived in [167]. PAC-Bayes bounds [170] highlight trade-offs between dropout rates.

Exponential convergence with marginalized dropout in shallow linear networks is shown [172], while dropout in gradient descent on logistic loss for shallow ReLU networks is analyzed [185], with misclassification error rates derived in the lazy regime for overparameterized networks.

In generalized linear models, dropout training is shown to be the minimax solution to a zero-sum game where an adversary corrupts covariates with a multiplicative errors-in-variables model. The least favorable distribution is dropout noise, with covariates deleted with probability δ, providing out-of-sample loss guarantees for perturbed data distributions, and offering a method for selecting δ along with a parallelizable multi-level Monte Carlo algorithm for faster dropout training [186].

4.6. Fault Tolerance and Generalization

Noise introduction during training enhances both generalization and fault tolerance: input noise improves generalization [140], and synaptic noise strengthens fault tolerance [187]. These properties are interconnected; improving one often enhances the other [188,189]. Reducing weight magnitudes increases both fault tolerance [188,190] and generalization [190,191]. VC-dimension theory explains how redundancy improves these attributes [192].

Fault tolerance relies on distributing learning across neurons, a task not inherently optimized by the BP algorithm [189]. Weight perturbations during training improve fault tolerance in MLPs [187,189], similarly to how input noise enhances generalization. Lower saliency, derived from the Hessian matrix, indicates higher fault tolerance [187]. Metrics for MSE degradation under perturbations help refine understanding of noise immunity and generalization [190].

Online node fault injection, where hidden nodes output zero randomly during training, enhances fault tolerance in MLPs, with convergence proofs [193]. This method incorporates terms for MSE, regularization, and weight decay. Fault injection studies in RBF networks show that weight noise (additive or multiplicative) does not improve fault tolerance, although additive input noise does improve generalization by resembling Tikhonov regularization [194]. Stuck-at faults, where a node's output is fixed, are tolerated by well-trained networks, demonstrating robustness [188,195].

4.7. Sparsity Versus Stability

Stability plays a crucial role in determining the generalization performance of an algorithm [196]. Sparsity and stability are both highly desirable characteristics for learning algorithms, as they both contribute to enhanced generalization. However, these two properties are inherently contradictory, as demonstrated by the no-free-lunch theorem [197,198]. Specifically, a sparse algorithm cannot also exhibit stability, and vice versa. Sparse algorithms may have multiple optimal solutions, making them ill-posed. For sparse algorithms, uniform stability is inherently constrained by a nonzero lower bound, reinforcing the conclusion that sparsity and stability are fundamentally incompatible. As a result, a balance must be struck between sparsity and stability when designing learning algorithms.

In [197,198], it is shown that an L_1-regularized regression (LASSO, least absolute shrinkage and selection operator) lacks stability, while L_2-regularized regression is stable but not sparse. Algorithms that promote sparsity include L_1-norm SVM, LASSO, sparse PCA, and deep belief networks.

5. Model Selection

Model selection identifies the simplest model that balances data fit and generalization, a key model quality measure. Selection strategies include cross-validation, complexity-based criteria, regularization, and pruning/growing. Generalization error can be estimated through cross-validation or bootstrapping. Cross-validation evaluates models of varying complexities on validation sets but is resource-intensive, while complexity-based criteria avoid validation sets yet remain computationally demanding. Regularization efficiently penalizes complexity, but may yield suboptimal representations. Pruning and growing, often combined with regularization, rely on assumptions that can restrict model flexibility.

5.1. Occam's Razor

Occam's razor, attributed to William of Occam, advocates for simplicity in model selection: entities should not be multiplied beyond necessity. It suggests that simpler

models are preferable when they fit the data similarly to more complex models, particularly in noisy environments, as they often generalize better.

While often seen as a principle of simplicity, Occam's razor implies that simpler models generalize better when training performance is equivalent. However, overfitting or poor generalization is argued to be more closely related to the number of models evaluated by the learner, rather than to model complexity [199]. Complex models typically require more evaluations, creating a confounding relationship between complexity and generalization.

Empirical evidence from [200] supports this view, showing that: (1) models chosen from larger candidate pools compared to those from smaller pools, provided their complexity remains unchanged; and (2) more complex models are not inherently more likely to overfit than simpler ones, as long as the size of the candidate pool being evaluated stays fixed.

5.2. Cross-Validation

Cross-validation, a common model selection method [201], splits the dataset into a training set and a validation set (typically 10–20% of the data). Leave-one-out cross-validation uses a single sample for validation, while providing accurate performance estimates [134], cross-validation is computationally expensive due to repeated training, despite reducing variance in estimates.

Consider a dataset partitioned into n subsets, with $\mathcal{D}_i$ and $\overline{\mathcal{D}}_i$ representing the training and testing subsets, respectively, for the ith partition. During cross-validation, the model is trained n times to minimize the log-likelihood,

$$E_{\mathrm{CV}} = -\frac{1}{n}\sum_{i=1}^{n} \ln\left(L\left(\widehat{\mathbf{W}}\left(\overline{\mathcal{D}}_i\right)\middle|\mathcal{D}_i\right)\right), \tag{15}$$

where $\widehat{\mathbf{W}}\left(\overline{\mathcal{D}}_i\right)$ are the maximum likelihood estimates from the test set $\overline{\mathcal{D}}_i$, and $L\left(\widehat{\mathbf{W}}\left(\overline{\mathcal{D}}_i\right)\middle|\mathcal{D}_i\right)$ is the likelihood evaluated on the training set $\mathcal{D}_i$.

Validation uses a separate dataset to estimate generalization error, helping avoid overfitting and identifying the best model. Cross-validation, especially for large training sets (N), supports generalization [202].

In K-fold cross-validation, the dataset $\mathcal{D}$ is divided into K partitions, where one partition serves as the validation set while the remaining partitions are used for training. The error is averaged across folds, and variance is influenced by data distribution [203]. Nearly unbiased estimators are obtained for smooth loss functions and absolute error loss [204], with a practical 25% data test split [204]. K-fold cross-validation can introduce bias in least-squares regression, but K-fold penalization corrects this bias [205]. A K value greater than 5 stabilizes variance, with $K = 5$ commonly recommended [206].

Monte Carlo cross-validation replaces K with B random splits, with variance estimators studied [205]. Leave-many-out cross-validation outperforms leave-one-out, particularly for small samples, although resampling reduces variance in both [207]. Cross-validation can be inconsistent for model selection [208], and the moment approximation estimator [204] generally outperforms the Nadeau–Bengio estimator [209], though the latter is simpler.

Cross-validation and bootstrapping differ in resampling methods: bootstrapping resamples with replacement, while cross-validation does so without. Cross-validation estimates generalization error, while bootstrapping estimates error bars and confidence intervals. In small sample sizes, cross-validation has higher variance than bootstrapping, despite being more accurate and less biased [207].

5.3. Complexity Criteria

An effective generalization strategy involves compact network development through parsimonious model selection, using methods like final prediction error (FPE) [210], Akaike information criterion (AIC) [211], Bayesian information criterion (BIC) [212], and minimum description length (MDL) principle [213]. These combine training error measurement with a penalty for model complexity.

Model order is typically determined by minimizing the Kullback–Leibler (KL) divergence between the true and model pdfs, or equivalently, maximizing the relative KL information. Asymptotic approximations of this lead to minimizing the AIC, which is formulated by optimizing an asymptotically unbiased estimator of the relative KL information, I. The BIC rule, similarly derived, is a penalized maximum likelihood method [214].

AIC minimizes prediction error, while BIC focuses on model selection for inference. The formulations for AIC and BIC are given by

$$E_{\mathrm{AIC}} = -\frac{1}{N}\ln\left(L_N\left(\widehat{\mathbf{W}}_N\right)\right) + \frac{N_P}{N}, \tag{16}$$

$$E_{\mathrm{BIC}} = -\frac{1}{N}\ln\left(L_N\left(\widehat{\mathbf{W}}_N\right)\right) + \frac{N_P}{2N}\ln N, \tag{17}$$

where $L_N\left(\widehat{\mathbf{W}}_N\right)$ is the likelihood for a dataset of size N with parameters $\widehat{\mathbf{W}}_N$, and N_P is the number of model parameters.

These criteria can be reformulated in terms of empirical risk [214],

$$AIC(N_P) = R_{\mathrm{emp}}(N_P) + \frac{2N_P}{N}\hat{\sigma}^2, \tag{18}$$

$$BIC(N_P) = R_{\mathrm{emp}}(N_P) + \frac{N_P}{N}\hat{\sigma}^2\ln N, \tag{19}$$

where $\hat{\sigma}^2$ is the estimated noise variance and $R_{\mathrm{emp}}(N_P)$ is the empirical risk,

$$R_{\mathrm{emp}}(N_P) = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(\mathbf{x}_i, N_P))^2. \tag{20}$$

For a linear model with N_P parameters, the noise variance is estimated by

$$\hat{\sigma}^2 = \frac{N}{N - N_P}\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2. \tag{21}$$

This leads to the final prediction error (FPE) form of AIC [215],

$$\mathrm{FPE}(N_P) = \frac{1 + \frac{N_P}{N}}{1 - \frac{N_P}{N}}R_{\mathrm{emp}}(N_P). \tag{22}$$

The MDL principle formalizes Occam's razor by seeking the most concise dataset description with minimal symbols [213,216]. The model's description length includes the code cost for inputs, the model cost, and the reconstruction error, with the optimal model minimizing this total length. MDL applies in unsupervised learning methods like competitive learning and PCA [143], where it encourages good generalization by encoding weights with shorter bit-lengths. The MDL criterion is similar to a Bayesian approximation, with BIC derived from coding arguments related to MDL.

Generalization error Err combines training error err and optimism OP in an estimate [217], with complexity criteria like BIC used to estimate OP.

6. Bias and Variance

The generalization error comprises two components: the squared *bias* and the *variance* [138]. Supervised learning algorithms commonly encounter the bias–variance trade-off [138], where the goals of minimizing bias and variance are often in conflict, requiring a balance between the two.

Let $f(\mathbf{x}; \hat{\mathbf{w}})$ represent the optimal model within the model space, where $\hat{\mathbf{w}}$ is independent of the training dataset. The bias and variance are expressed as [139]

$$\text{bias} = E_{\mathcal{S}}[f(\mathbf{x})] - f(\mathbf{x}; \hat{\mathbf{w}}), \quad \text{var} = E_{\mathcal{S}}\left[\left(f(\mathbf{x}) - E_{\mathcal{S}}[f(\mathbf{x})]\right)^2\right], \tag{23}$$

where $f(\mathbf{x})$ is the target function, and $E_{\mathcal{S}}$ is the expectation over all possible training datasets. Bias arises from an inappropriate level of model complexity when assuming an infinite number of training samples, while variance is caused by the finite size of the training data.

The generalization error can be expressed as

$$E_{\mathcal{S}}\left[\left(f(\mathbf{x}) - f(\mathbf{x}, \hat{\mathbf{w}})\right)^2\right] = E_{\mathcal{S}}\left[\left(\left(f(\mathbf{x}) - E_{\mathcal{S}}[f(\mathbf{x})]\right) + \left(E_{\mathcal{S}}[f(\mathbf{x})] - f(\mathbf{x}, \hat{\mathbf{w}})\right)\right)^2\right] \tag{24}$$

$$= (\text{Bias})^2 + \text{Var}. \tag{25}$$

A network with too few parameters may underfit (high bias, low variance), while one with too many parameters may overfit (low bias, high variance). Balancing bias and variance is essential for optimal generalization, achievable through model complexity selection or regularization. Nonparametric methods aim to reduce variance. Increasing hidden units reduces bias but raises variance.

For three-layer feedforward networks containing N_P hidden sigmoidal units, the bias is bounded by $O\left(\frac{1}{N_P}\right)$, while the variance is bounded by $O\left(\frac{N_P J_1 \ln N}{N}\right)$ [132]. Here, N denotes the training set size and J_1 is the feature vector dimensionality. A larger N_P reduces bias, but may lead to overfitting, highlighting the bias–variance trade-off.

Unbiasedness in model selection is important, but controlling variance is crucial, as excessive variance can lead to overfitting [218], mitigated by regularizing the selection criterion [219].

Variance and bias reduction are vital in Monte Carlo simulation, with control variates and importance sampling as key techniques. The control variate method reduces variance by incorporating an auxiliary variate with a known mean [220]. Coefficients for control variates are typically estimated using least squares [221], and adding more control variates accelerates convergence [222], while regularized least squares enhances precision [223].

7. Overparameterization and Double Descent

SVM generalization error in overparameterized settings has been analyzed using the convex Gaussian min-max theorem for precise asymptotics [224–226]. Upper bounds for max-margin SVMs in overparameterized linear discriminant models emphasize anisotropy for better generalization [227]. Overparameterization in linear models with Gaussian features turns all training points into support vectors [161], with least-squares minimum-norm interpolation outperforming hard-margin SVM solutions under 0-1 loss. Overparameterized models generalize better in classification than regression under milder conditions.

Overparameterized models, including deep neural networks, achieve near-zero training loss and strong generalization, creating a 'generalization paradox' [148,228–230]. This is linked to double descent [228,229,231], as illustrated in Figure 2, where test error decreases, rises near the interpolation threshold, and declines again with increased overparameterization. The generalization paradox contrasts the classical bias–variance regime (first descent)

with the modern regime (second descent) [115,148,228,230], with VC-dimension potentially differing from the number of parameters [9,232,233].

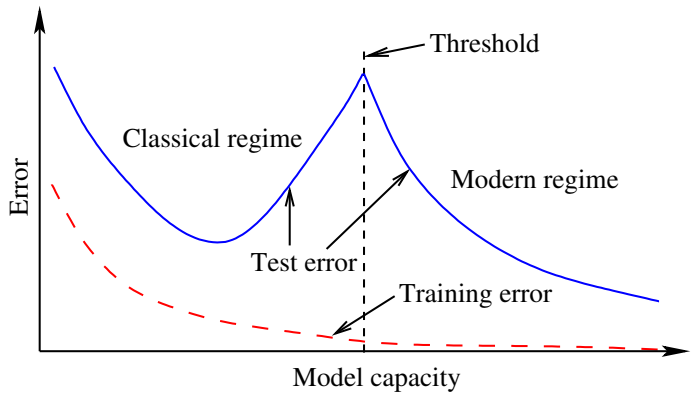


Figure 2. Illustration of double descent curve.

Double descent highlights overparameterization's benefits, where effective interpolation relies on feature families aligning critical signal directions while absorbing noise, emphasizing the complex relationship between overparameterization and generalization. For deep ReLU networks, the adoption of ReLU function makes many neuron become inactive and output zero, and a large portion of the parameters do not work. This properly explains the double descent phenomenon in deep learning [234].

Test error variance in two-layer networks has been analyzed via analysis of variance (ANOVA), decomposing contributions from initialization, label noise, and training data [231], diverging from traditional theory [235] that attributes variance solely to the training set. Interaction between samples and initialization dominates variance, with algorithms contributing to implicit regularization, reducing complexity and enhancing generalization [231]. Overparameterized models exhibit implicit regularization, imposing structural constraints [236,237].

Theoretical studies show benign overfitting in regression tasks, with empirical risk minimization leading to good generalization despite noisy data [160,238–240]. Generalization error in two-layer networks has been studied via ANOVA, with interaction effects dominating variance under Gaussian initialization [241]. Variance is unimodal with respect to parameterization, influenced significantly by label noise and initialization [242,243].

Efforts to reconcile classical theory with modern overparameterized models reflect nonparametric statistics principles [244], such as kernel smoothing, which uses many parameters for strong performance. This aligns with the "overparametrize-then-regularize" strategy in machine learning, akin to techniques like basis pursuit and Lasso [54].

In overparameterized settings, SGD can lead to benign overfitting without explicit regularization. Sharp excess risk bounds for constant-stepsize SGD in linear regression link variance to an SGD-specific effective dimension and bias to the initial iterate's alignment with the covariance structure [245]. Tail averaging in SGD improves risk bounds, and algorithmic regularization in unregularized SGD is contrasted with ordinary least squares and ridge regression.

Three mechanisms for controlling generalization include maintaining a small feature set relative to training size, minimizing VC-dimension for linearly separable data, and maximizing data compression efficiency when encoding the training set [233]. The double descent phenomenon in deep learning, where large networks achieve perfect training fit and strong generalization, challenges classical VC-theory, which requires finite VC-dimension for generalization [148,228,230,246–249].

Weight initialization, normalization, decay, and epochs influence VC-dimension, known as SGD's 'implicit bias'. Generalization is controlled by weight norms [236,237,246,250], with gradient descent maximizing the margin on linearly separable data, in line with VC-theoretical explanations [237]. High-dimensional data are controlled by weight norms [236,237,249–253], while low-dimensional data benefits from data compression or MDL. VC-theory concepts, including radius-margin and leave-one-out bounds, provide insights into generalization for second descent, with weight norms governing generalization [249].

Double descent is explained within the VC-theoretical framework using VC-dimension and structural risk minimization (SRM), with generalization curves aligning with classical VC bounds. This analysis provides insights into generalization for noisy or high-dimensional data and extends to transfer learning in pre-trained networks [254]. A conceptual interpretation of double descent is presented using VC-bound (56) [254], where minimizing this bound via data-dependent SRM is linked to fat-shattering dimension for large-margin hyperplanes [255]. During second descent, as the number of parameters increases, VC-dimension is controlled by the weight norm and decreases for larger networks [254]. However, results from noisy, high-dimensional datasets indicate that second descent does not always enhance generalization. In fact, memorizing noisy data can increase the VC-dimension which, in turn, leads to a higher VC-bound. The VC-bound-based theoretical explanation of double descent is further validated for classifiers such as SVMs, least squares, and MLPs [256].

The optimal generalization strategy depends on training data characteristics, with VC-bounds guiding the selection of an optimal approach. Empirical studies suggest second descent is effective for high-dimensional data, though the reasons remain unclear [254]. In under-parameterized networks, the VC-dimension increases linearly with the number of features N_f, which aligns with first descent. In contrast, for over-parameterized networks, the VC-dimension is influenced by the weight norm, leading to second descent. As N_f grows, double descent curves emerge.

In transfer learning, pre-trained deep networks extract features for simpler domain-specific networks. VC-theory explains this in two steps: a pre-trained network learns N_f non-linear features using general data, and a linear classifier is trained on these features with domain-specific data. Since the non-linear mapping does not depend on domain-specific data, VC-bound (56) applies to step 2 for generalization.

8. Neural Networks as Universal Machines

Neural networks excel at approximating complex functions. Feedforward neural networks can approximate any continuous function under suitable conditions, while RNNs with sigmoidal activations are Turing equivalent [257], meaning they can replicate any function computable by a Turing machine, matching the computational power of digital computers.

Reservoir computation models, a subclass of RNNs, have fixed inputs and dynamic coupling weights, with only the static readout being trainable, avoiding gradient propagation issues. These models serve as universal approximators for dynamic filters with time-invariant fading memory. Simple cycle reservoirs, with equal weight ring connectivity and binary input-to-reservoir weights, can universally approximate any unrestricted linear reservoir system and time-invariant fading memory filter over uniformly bounded input streams [258].

Differentiable neural computers enhance neural networks with external memory. The reservoir memory machine [259], an echo state network with external memory and convex optimization training, outperforms finite state machines and solves benchmark

tasks for differentiable neural computers more effectively than traditional recurrent models, including deep networks.

Universal memcomputing machines (UMMs) integrate memory for both storage and processing, achieving Turing-completeness [260]. They are capable of simulating both liquid-state machines (LSMs) and quantum computers, making them liquid-complete and quantum-complete [261], thus offering a unifying framework for diverse computational models.

8.1. Boolean Function Approximation

Feedforward neural networks with binary neurons model Boolean functions by processing variables in binary $(0, 1)$ or bipolar $(-1, +1)$ formats. For J_1 Boolean variables, 2^{J_1} combinations exist, yielding $2^{2^{J_1}}$ distinct Boolean functions. A linear threshold gate (LTG) separates two classes, and the function counting theorem [262,263] quantifies the linearly separable dichotomies for m points in a general position in $\mathcal{R}^n$, defining a single LTG's classification capacity.

Theorem 1 (Function counting theorem)**.** *The count of distinct dichotomies that are linearly separable for m points, in general, position within* $\mathbb{R}^n$*, is expressed as*

$$C(m,n) = \begin{cases} 2\sum_{i=0}^{n} \binom{m-1}{i}, & \text{if } m > n+1, \\ 2^m, & \text{if } m \leq n+1, \end{cases} \tag{26}$$

A set of m points in $\mathbb{R}^n$ is in *general position* if any subset of m or fewer points is linearly independent.

For m points, the total number of possible dichotomies is 2^m. Assuming equiprobable dichotomies, the likelihood of an LTG with n inputs successfully separating m points in a general position is

$$P(m,n) = \frac{C(m,n)}{2^m} = \begin{cases} \frac{2}{2^m}\sum_{i=0}^{n} \binom{m-1}{i}, & \text{if } m > n+1, \\ 1, & \text{if } m \leq n+1. \end{cases} \tag{27}$$

The probability of a linear dichotomy, $P(m,n)$, equals 1 when $\frac{m}{n+1} \leq 1$. For $1 < \frac{m}{n+1} < 2$, P approaches 1 as $n \to \infty$. At the critical ratio $\frac{m}{n+1} = 2$, P drops to $\frac{1}{2}$. The value $m = 2(n+1)$ is often employed to assess the statistical capacity of a single LTG. As $m/(n+1)$ further increases, P approaches 0.

A three-layer feedforward LTG network with a (J_1-2^{J_1}-1) structure can represent any Boolean function with J_1 inputs [264,265]. For a general function $f : \mathbb{R}^{J_1} \to \{0, 1\}$ defined over N arbitrary points in $\mathbb{R}^{J_1}$, the minimum number of hidden nodes required is estimated as $O\left(\frac{N}{J_1 \log_2 \frac{N}{J_1}}\right)$ for $N \geq 3J_1$ and $J_1 \to \infty$ [109,263]. When N points are, in general, position, this reduces to $\frac{N}{2J_1}$ as $J_1 \to \infty$ [263]. Multi-hidden-layer networks are generally more size-efficient than single-hidden-layer designs [263].

Binary Radial Basis Function

In three-layer feedforward networks, a network using binary or generalized binary RBF activation functions in the hidden layer and LTG output neurons is called a binary or generalized binary RBF network, capable of modeling Boolean functions.

A generalized binary RBF neuron is defined by a center $\mathbf{c} \in \mathbb{R}^n$ and a non-negative radius $r \geq 0$, with an activation function $\phi : \mathbb{R}^n \to \{0, 1\}$, defined as

$$\phi(\mathbf{x}) = \begin{cases} 1, & \|\mathbf{x} - \mathbf{c}\|_{\mathbf{A}} \leq r \\ 0, & \text{otherwise} \end{cases}, \tag{28}$$

where $\mathbf{A}$ is a real, symmetric, positive-definite matrix, and $\|\cdot\|_{\mathbf{A}}$ represents the Euclidean norm influenced by $\mathbf{A}$. If $\mathbf{A}$ is the identity matrix, the neuron acts as a binary RBF neuron.

A generalized binary RBF neuron, with greater computational power than LTGs, can compute any Boolean function of an LTG, allowing it to replace an LTG processing binary inputs without changing the network's computational power [266].

8.2. Linear Separability and Nonlinear Separability

Definition 1 (Linear separability)**.** *Consider a set $\mathcal{X}$ consisting of N patterns $\mathbf{x}_i$ in a J_1-dimensional space, with each pattern assigned to either class $\mathcal{C}_1$ or $\mathcal{C}_2$. The problem is termed* linearly separable *if there is a hyperplane capable of distinguishing all points of class $\mathcal{C}_1$ from those of class $\mathcal{C}_2$.*

An LTG can represent a linearly separable dichotomy using a linear separating hyperplane, defined by

$$\mathbf{w}^T\mathbf{x} + w_0 = 0, \tag{29}$$

where $\mathbf{w} \in \mathbb{R}^{J_1}$ and w_0 represents the bias term. For any given pattern, if $\mathbf{w}^T\mathbf{x} + w_0 > 0$, the point is assigned to $\mathcal{C}_1$; if $\mathbf{w}^T\mathbf{x} + w_0 < 0$, it is assigned to $\mathcal{C}_2$.

Definition 2 (φ-separation)**.** *A dichotomy $\{\mathcal{C}_1, \mathcal{C}_2\}$ of a set $\mathcal{X}$ is called* φ-separable *if there exists a transformation $\varphi : \mathbb{R}^{J_1} \to \mathbb{R}^{J_2}$ that maps the data into a space where a separating hyperplane exists [262], such that*

$$\mathbf{w}^T\varphi(\mathbf{x}) = 0 \tag{30}$$

with $\mathbf{w}^T\varphi(\mathbf{x}) > 0$ for points $\mathbf{x} \in \mathcal{C}_1$ and $\mathbf{w}^T\varphi(\mathbf{x}) < 0$ for points $\mathbf{x} \in \mathcal{C}_2$. In this case, $\mathbf{w}$ is a vector in $\mathbb{R}^{J_2}$.

A dichotomy that is not linearly separable can become separable when mapped to a higher-dimensional space. This is illustrated in Figure 3, where two initially inseparable dichotomies become φ-separable after transformation. The figure also illustrates linearly separable and nonlinearly separable classes in a two-dimensional space.

For nonlinearly separable problems, the linear term in an LTG can be substituted by higher-order polynomial terms, creating a polynomial threshold gate. The function counting theorem still holds for polynomial threshold gates, provided the m points are, in general, position in the transformed φ-space.

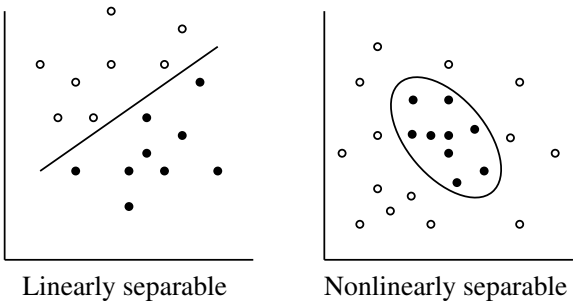


Figure 3. Classification examples in a 2D space illustrating linearly separable, and linearly inseparable/nonlinearly separable cases, with dots and circles representing different classes.

Higher-order neurons, or Σ-Π units, improve linear neuron models by adding nonlinearity through monomials—products of input variables—in the hidden layer, enhancing the model's ability to capture complex relationships.

8.3. Universal Function Approximation

Kolmogorov's theorem, or the Kolmogorov–Arnold superposition theorem, resolving Hilbert's 13th problem, states that any continuous function defined on a compact set can be represented through compositions and superpositions of univariate functions [267]. The Kolmogorov–Arnold representation decomposes a multivariate function into interior and outer functions, akin to a two-hidden-layer network [268,269]. Kolmogorov's theorem guarantees that a feedforward network with two hidden layers and a sufficient number of hidden units is capable of approximating any continuous function to any desired degree of accuracy [270].

Theorem 2 (Kolmogorov)**.** *Any continuous real-valued function $f(x_1, \ldots, x_n)$, where $n \geq 2$ and the domain is $[0,1]^n$, can be represented by*

$$f(x_1, \ldots, x_n) = \sum_{j=1}^{2n+1} h_j \left(\sum_{i=1}^{n} \psi_{ij}(x_i) \right), \tag{31}$$

where h_j and ψ_{ij} are continuous, and ψ_{ij} are monotonically increasing functions, independent of f.

Hecht-Nielsen extended this result, showing neural networks can approximate complex functions with simpler, single-variable components [270].

Theorem 3 (Hecht-Nielsen)**.** *Any continuous function $f : [0,1]^n \to \mathbb{R}^m$ can be approximated with arbitrary accuracy by a feedforward neural network consisting of n input nodes, $2n+1$ hidden neurons, and m output nodes.*

The Kolmogorov two-hidden-layer network model can exactly represent continuous, discontinuous bounded, and all unbounded multivariate functions, depending on whether the activation function in the second hidden layer is continuous, discontinuous bounded, or unbounded [268]. Recent adaptations [269,271] applied this to deep ReLU networks, enabling them to approximate the outer function while most layers focus on the interior function.

The Weierstrass theorem states that any continuous real-valued function of multiple variables can be approximated with arbitrary precision by a polynomial. The Stone–Weierstrass theorem [272], extending Weierstrass' result, underpins the approximation capabilities of dynamic system models.

Establishing appropriate activation functions and universal approximation properties has been foundational for neural networks. The universal approximation theorem shows that a neural network with a single hidden layer is capable of approximating any continuous function defined on compact sets, making it suitable for tasks like regression and classification.

The approximation capabilities of fully connected feedforward neural networks, or MLPs, were rigorously analyzed before 2000. Key results include proofs for networks with continuous and discriminatory activation functions [273], nonconstant, bounded, monotone continuous functions [274], bounded generalized sigmoid functions with relaxed continuity [275], bounded and nonconstant functions [276], and locally bounded, piecewise continuous, nonpolynomial functions [277].

RNNs can be broadly categorized into globally recurrent networks and locally recurrent networks with global feedforward structures, both of which can serve as universal approximators for dynamical systems [278,279]. Echo state networks, as a special form of RNN, are universal approximators under L^p criteria [280].

Extensions of the universal approximation theorem to single hidden-layer MLPs incorporate hypercomplex-valued networks, where inputs, outputs, and parameters are hypercomplex numbers. Examples include networks based on complex numbers, quaternions [281], and Clifford algebra, such as hyperbolic-valued [282,283], tessarine-valued [284], complex-valued [285], and vector-valued networks [286,287].

Vector-valued networks treat vector arrays as unified entities, with hypercomplex-valued networks forming a subclass defined over algebras with a multiplicative identity, including complex, quaternion, hyperbolic, tessarine, and Clifford-valued networks.

Extensions also demonstrate that neural networks with dropout retain universal approximation properties [288]. Additionally, other feedforward networks, such as functional link neural networks [289,290] and broad learning systems [291,292], also possess universal approximation capabilities.

Geometric deep learning models can approximate continuous target functions on compact sets, with curvature-dependent bounds on diameter and model depth [293]. However, locally defined models may fail to approximate continuous functions between non-degenerate compact manifolds [293]. Universal approximation has also been established for hyperbolic feedforward networks [294] and deep Kalman filter architectures [295].

Deep sets [296] act as universal approximators for continuous set functions, with their expressive capacity governed by latent space dimensionality [296,297]. These dimensionality conditions, proven optimal in [297], emphasize the critical role of latent space size. Deep sets implement the Janossy pooling paradigm, foundational to modern set-learning methods [297]. Despite advancements, open questions remain about Janossy pooling's broader applicability, even as curriculum learning [79] and exploration bonuses [83] address specific training challenges.

Self-attention provides an alternative, aggregating information via input-dependent weights to support relational reasoning by processing element pairs. Originally developed for natural language processing [298], self-attention has been adapted to sets [299] and graph learning [300], serving as a k-ary Janossy pooling instance alongside deep sets [301].

Capacity of a Neural Network Architecture

The capacity of a neural network, defined as the binary logarithm of the total functions it can realize by adjusting synaptic weights, establishes an upper bound on bits (information) that can be derived from training data.

For fully connected networks with L layers of sizes $n_1, n_2, \ldots, n_L$, capacity is approximated in [302] by

$$C(n_1, \ldots, n_L) = \sum_{k=1}^{L-1} \min(n_1, \ldots, n_k) n_k n_{k+1}, \tag{32}$$

with smaller layers acting as bottlenecks. Techniques like multiplexing, enrichment, and stacking, along with bounds on finite set capacities, identify optimal architectures under constraints, introducing structural regularization.

While shallow networks compute more functions, deep networks yield more structured functions [302]. It is summarized in [234] that both deep and shallow models can achieve the same performance, which is validated by the no-free-lunch theorem [8].

For ReLU networks, a single unit's capacity scales as $n^2(1 + o(1))$, where n is the input dimension [303], and multilayer networks follow a similar trend. The VC-dimension of ReLU networks grows super-linearly with depth L [304].

8.4. Turing Machines

Turing-completeness, essential for neural networks learning algorithms from examples, means a network can simulate any Turing machine with unbounded memory. Neural networks are generally Turing-complete, with RNNs proven to be so even with bounded resources [257], leveraging internal representations. Extensions like neural Turing machines [305] maintain this completeness.

Self-attention architectures, such as the Transformer [298], dominate sequence-processing tasks like GPT-3 [306]. The capabilities of the Transformer are underpinned by its attention mechanism. Turing-completeness of the Transformer with hard attention has been demonstrated due to its ability to compute and access internal dense representations, with minimal elements for this completeness identified [307]. Unlike the fixed precision assumption in [308], fixed-precision Transformers are not Turing-complete, with differences between hard and soft attention highlighted [307].

The computational power of neural networks dates back to [1] for comparing neurons to Boolean formulae, and [309] for linking neural networks to finite automata. Turing-completeness of finite neural networks with linear connections was first demonstrated by [257,257]. However, practical learning requires more than Turing-completeness, prompting interest in enhancing RNNs with external memory.

Recent trends favor non-recurrent architectures like the Transformer, which excels in language tasks [298]. Enhancements for better generalization across input lengths have been proposed [308], and studies have explored the Transformer encoder as a language recognizer with limitations identified [310] and a universal approximator of continuous functions over strings [311].

Turing Machine Computations

Definition 3 (Turing machine)**.** *A (deterministic) Turing machine is formally defined as a tuple* $\mathcal{M} = (\mathcal{Q}, \Sigma, \delta, q_{init}, \mathcal{F})$*, where* $\mathcal{Q}$ *represents a finite state set,* Σ *denotes a finite input alphabet,* $\delta : \mathcal{Q} \times \Sigma \to \mathcal{Q} \times \Sigma \times \{1, -1\}$ *specifies the state transition function,* $q_{init} \in \mathcal{Q}$ *is the starting state, and* $\mathcal{F} \subseteq \mathcal{Q}$ *represents the set of halting states.*

The machine operates on a single tape with infinitely many cells extending to the right. A special blank symbol $\# \in \Sigma$ marks empty positions on the tape. The machine uses a single head that moves left or right, reading and writing symbols from Σ.

For the Turing-completeness of sequence-to-sequence (seq-to-seq) neural network architectures, let $\mathcal{N}$ denote such a class, with $\mathcal{L}_{\mathcal{N}}$ as the set of languages recognized by its language recognizers. Turing-completeness for $\mathcal{N}$ is defined as follows.

Definition 4 (Turing-complete)**.** *A sequence-to-sequence neural network class* $\mathcal{N}$ *is considered Turing-complete if the language set* $\mathcal{L}_{\mathcal{N}}$ *includes all languages that are decidable, meaning those that can be recognized by a Turing machine.*

One can formalize the Transformer architecture, abstracting away specific parameter or function choices.

Theorem 4 (Turing-completeness of Transformer networks [307])**.** *The Transformer network architecture, when equipped with positional encodings, is capable of universal computation and thus Turing-complete, even under restricted conditions where positional embeddings* $pos(n)$ *for* $n \in \mathbb{N}$ *are limited to* $n, 1/n$*, and* $1/n^2$*, with a single encoder layer and three decoder layers.*

Transformers can recognize all recursively enumerable (semi-decidable) languages, extending beyond decidable ones. The undecidability of some problems in probabilistic

language modeling with RNNs [312] suggests heuristic solutions, relying on the Turing-completeness of RNNs [257].

8.5. Winner-Takes-All

The winner-takes-all (WTA) competition, seen in artificial and biological systems, is a highly efficient computational module, outperforming threshold and sigmoidal gates [313]. A quadratic lower bound for computing WTA in feedforward circuits with threshold gates is established in [313], and circuits with a single soft WTA gate can approximate arbitrary continuous functions [313].

Theorem 5 (Maass, 1 [313]). *For a winner-takes-all (WTA) with $n \geq 3$ inputs, any feedforward circuit C composed of threshold gates with arbitrary weights requires at least $\binom{n}{2} + n$ threshold gates.*

Theorem 6 (Maass, 2 [313]). *A two-layer feedforward circuit C with m binary or analog inputs and a binary output, constructed from threshold gates, can be emulated by a circuit utilizing a single k-winner-takes-all (k-WTA) gate, which operates on n weighted sums of the input variables with positive weights, except for a measure-zero set $S \subseteq \mathbb{R}^m$.*

Any Boolean function $f : \{0,1\}^m \to \{0,1\}$ can similarly be realized by a single k-WTA gate that processes weighted sums of the input bits. When C is a polynomial-size circuit with integer weights constrained by a polynomial in m, n can also be polynomially bounded in m, ensuring the weights are natural numbers and the circuit size remains polynomial in m.

For inputs $(x_1, \ldots, x_n)$ with real values, a soft-WTA gate produces outputs $(r_1, \ldots, r_n)$, where r_i reflects the relative position of x_i in the input order. These soft-WTA functions mimic cortical circuits with lateral inhibition and, and when used individually as gates, they can approximate any continuous function with arbitrary precision.

Theorem 7 (Maass, 3 [313]). *Consider a continuous function $h : D \to [0,1]$ defined on a bounded, closed domain $D \subseteq \mathbb{R}^m$. For any given $\epsilon > 0$ and any function g meeting these conditions, there exist natural numbers k, n, biases $\alpha_0^j \in \mathbb{R}$, and coefficients $\alpha_i^j \geq 0$ for $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$, such that the soft-WTA gate soft-WTA$_{n,k}^g$, when applied to the sums $\sum_{i=1}^m \alpha_i^j z_i + \alpha_0^j$ for $j = 1, 2, \ldots, n$, computes a function $f : D \to [0,1]$ that satisfies the condition $|f(\mathbf{z}) - h(\mathbf{z})| < \epsilon$ for all $\mathbf{z} \in D$. Consequently, circuits containing a single soft-WTA gate acting on positively weighted sums of input variables can universally approximate any continuous function.*

9. Introduction to Computational Learning Theory

Machine learning algorithms predict an unknown model using training data from various hypotheses, with learning theory providing probabilistic performance bounds due to finite data. Computational learning theory explores how to achieve optimal generalization in supervised learning, utilizing frameworks such as VC-theory [314] and probably approximately correct (PAC) learning [315], which are both distribution-free and nonparametric.

VC-theory [314], also known as statistical learning theory, estimates dependencies using the empirical risk minimization (ERM) principle. It connects uniform convergence in function classes to capacity, measured by the VC-dimension [314], which quantifies classification complexity. VC-theory underpins SVMs [316] and provides model optimism bounds.

PAC learning [315] identifies hypotheses approximating target concepts with high probability. Linked to ERM, it minimizes empirical error to improve approximations. PAC principles support methods like Boosting [317], and neural network generalization is analyzed via VC-dimension.

Kolmogorov complexity, though foundational, is impractical to compute. The MDL serves as an approximation. Vapnik and Chervonenkis introduced measures like VC-entropy, the growth function, and VC-dimension [9]. Additionally, the fat-shattering dimension [318], which generalizes VC-dimension to real-valued settings, relates to Rademacher complexity [319].

Minimax label complexity assesses the worst-case label requests needed for active learning to achieve a target error rate under noise models. Distribution-free bounds on minimax label complexity for general hypothesis classes are detailed in [320].

Classical statistics assumes a true model exists to explain the observed data, while VC-learning theory, grounded in risk minimization, does not require such an assumption. These concepts converge in practical machine learning. For example, least squares (LS) minimization, often used for function estimation, can be derived from maximum likelihood under Gaussian noise assumptions and framed within a risk-minimization framework. SVMs, originally developed within VC-theory, were later adapted for function approximation and regularization [235]. VC-theory asserts that generalization from finite samples is possible even when exact function approximation is not achievable [321], unlike regularization approaches that may not guarantee good generalization in finite sample settings.

Function approximation is concerned with estimating an unknown target function in regression tasks or predicting the posterior probability $P(y|\mathbf{x})$ in classification problems. In contrast, VC-theory focuses on selecting the function that minimizes prediction risk and ensures generalization, accounting for the (unknown) input distribution. The margin concept, introduced in VC-theory and later adopted as a form of regularization in SVM, does not naturally extend to the broader regularization framework. Ultimately, SRM and SVM can be viewed as specific instances within the larger regularization paradigm.

No-Free-Lunch Theorem

Prior to the no-free-lunch theorem by Wolpert [8,322], it was believed that some algorithms could consistently outperform others across all search tasks. Researchers sought to identify these universally efficient algorithms. The theorem refutes this, asserting that no algorithm is universally superior. Averaged over all discrete functions, all search algorithms perform equivalently, akin to random enumeration.

Theorem 8 (No-free-lunch theorem)**.** *Let $\mathcal{F}$ denote the set of all functions, with $\mathcal{F}_1$ being a particular subset. If algorithm A_1 outperforms algorithm A_2 on average over $\mathcal{F}_1$, then A_2 must outperform A_1 on the complement of $\mathcal{F}_1$, i.e., $\mathcal{F} - \mathcal{F}_1$.*

Algorithm performance depends on prior knowledge about the cost function, making evaluation meaningless without specifying assumptions. Real-world problems often involve priors like smoothness, symmetry, or independent and identically distributed samples, typically encoded by restricting the hypothesis class. While neural networks are effective for classification, they may not always be the most efficient solution, with other methods sometimes outperforming them.

The no-free-lunch theorem extends to areas like coding techniques, early stopping strategies [323], overfitting prevention, and noise prediction [324], emphasizing that no method suits all situations. In line with this, the limitations of leave-one-out cross-validation were demonstrated in [325], while for simple problems, strict leave-one-out cross-validation can still yield reliable results [326]. Nevertheless, statistical tests generally outperform cross-validation for model selection, both in linear and nonlinear settings [327].

10. Probably Approximately Correct (PAC) Learning

The PAC learning framework focuses on learning a target function, or concept, by selecting a suitable function from a hypothesis space, which consists of mappings from inputs to $\{0,1\}$. The objective is to identify a function that approximates the target concept. In supervised learning, PAC learning aims to produce a classifier that, with high probability $(1-\delta)$, achieves an error rate not exceeding ϵ, making it *probably approximately correct*. That is, PAC theory provides bounds on the generalization error with a probability larger than a certain threshold. A key challenge in PAC learning is understanding sample complexity.

Definition 5 (PAC learnability)**.** *Let $\mathcal{C}_n$ and $\mathcal{H}_n$, for $n \geq 1$, represent sets of target concepts and hypotheses over $\{0,1\}^n$, where $\mathcal{C}_n \subseteq \mathcal{H}_n$. If a polynomial-time algorithm can achieve low error with high confidence for all concepts in $\mathcal{C} = \{\mathcal{C}_n\}$ using $\mathcal{H} = \{\mathcal{H}_n\}$ with sufficient data, $\mathcal{C}$ is* PAC learnable *by $\mathcal{H}$.*

If the hypothesis space $\mathcal{H}$ contains $|\mathcal{H}|$ classifiers, and a classifier labels a random example correctly with probability less than $1-\epsilon$, then the probability of correctly labeling m examples is bounded by $P \leq (1-\epsilon)^m$, which decays rapidly for typical values of ϵ and m.

If an error rate above ϵ is unacceptable, evaluating m training examples allows the selection of $k \leq |\mathcal{H}|$ classifiers with no mistakes. The likelihood that the chosen k classifiers accurately classify the m examples satisfies the following inequality:

$$P \leq k(1-\epsilon)^m \leq |\mathcal{H}|(1-\epsilon)^m < |\mathcal{H}|e^{-m\epsilon}, \tag{33}$$

where $1-\epsilon < e^{-\epsilon}$.

To ensure the probability is below the failure threshold δ, the following inequality must hold:

$$|\mathcal{H}|e^{-m\epsilon} \leq \delta. \tag{34}$$

This leads to the bound derived in [328],

$$m > \frac{1}{\epsilon}\left(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\right). \tag{35}$$

Equation (35) illustrates that m grows linearly with $1/\epsilon$ and is relatively unaffected by δ. This result represents a worst-case scenario, assuming $k = |\mathcal{H}|$, which could be a conservative estimate.

A class is deemed not PAC learnable if satisfying the (ϵ,δ)-criteria requires an impractically large number of training examples.

Example 1. *General Boolean functions are not PAC learnable. When dealing with n Boolean variables, the instance space consists of 2^n possible examples. Consequently, the number of possible subsets is 2^{2^n}, meaning the size of the hypothesis space is $|\mathcal{H}| = 2^{2^n}$. Using Equation (35), the following inequality for the required number of examples m is derived:*

$$m > \frac{1}{\epsilon}\left(2^n \ln 2 + \ln\frac{1}{\delta}\right).$$

This lower bound increases exponentially with n, demonstrating that a general Boolean function is not PAC learnable.

PAC-Bayesian theory [329,330] analyzes the generalization of models like linear classifiers, SVMs, and neural networks by modeling predictions as samples from a posterior

distribution with bounds on average risk. While effective for randomized classifiers, applying these bounds to deterministic models requires derandomization, addressed in [331] through disintegrating PAC-Bayesian bounds [332,333]. These bounds offer guarantees for individual hypotheses, simplify optimization, and provide insights for algorithm design.

Sample Complexity

Definition 6 (Sample complexity of a learning algorithm)**.** *The sample complexity, denoted $m_{\mathcal{H}}$, refers to the minimum number of samples required for a learning algorithm to learn a concept class $\mathcal{C}$ using a hypothesis class $\mathcal{H}$, ensuring that the approximation error is within ϵ with probability at least $1-\delta$.*

For any consistent algorithm learning $\mathcal{C}$ with hypothesis class $\mathcal{H}$, the sample complexity is bounded by [328,334]:

$$m_{\mathcal{H}}(\epsilon,\delta) \leq \frac{1}{\epsilon(1-\sqrt{\epsilon})}\left(2d\ln\frac{6}{\epsilon}+\ln\frac{2}{\delta}\right), \quad \forall\, 0<\delta<1, \tag{36}$$

where d is the VC-dimension of $\mathcal{H}$. This bound ensures that with a confidence level of at least $1-\delta$, the algorithm produces a hypothesis whose error does not exceed ϵ, and the sample size scales linearly with the VC-dimension.

The sample complexity is upper-bounded in terms of the hypothesis space size $|\mathcal{H}|$ by [328],

$$m_{\mathcal{H}}(\epsilon,\delta) \leq \frac{1}{\epsilon}\left(\ln|\mathcal{H}|+\ln\frac{1}{\delta}\right). \tag{37}$$

For Boolean hypothesis spaces, the second PAC bound is typically more advantageous, whereas for infinite hypothesis spaces over real-valued attributes, only the first bound can be applied. The PAC framework provides upper bounds on the number of samples required for learning, with linear threshold concepts like perceptrons being PAC learnable over both Boolean and real-valued input domains [328].

The PAC framework and VC-dimension can estimate sample sizes for neural networks, including sigmoidal neurons [335] and linear threshold gates [336]. Sample sizes scale with ϵ and δ, typically as $N = O\left(\frac{N_w}{\epsilon}\right)$ [337].

Sample complexity estimates in [338] help control empirical deviations from expected cost functions and apply to matrix factorization techniques [54], such as PCA, NMF, and clustering methods [6], with bounds proportional to $\left(\frac{\log(N)}{N}\right)^{1/2}$. A majority-vote classifier trained on independent datasets mitigates the logarithmic growth of sample complexity [339].

11. Vapnik–Chervonenkis Dimension

The VC-dimension, a combinatorial measure of a neural network's expressive capacity, extends Cover's concept of capacity [262]. It is key in VC-theory for estimating the training sample size needed for effective generalization.

Definition 7 (VC-dimension)**.** *A set $\mathcal{S} \subseteq \mathcal{X}$ is shattered by a neural network class $\mathcal{N}$ if, for every function $f : \mathcal{S} \to \{0,1\}$, there exists a corresponding function in $\mathcal{N}$. The VC-dimension of $\mathcal{N}$ is the size of the largest shattered set,*

$$\dim_{\mathrm{VC}}(\mathcal{N}) = \max\{|\mathcal{S}|\,|\,\mathcal{S} \subseteq \mathcal{X} \text{ is shattered by } \mathcal{N}\}.$$

The VC-dimension d of a function family $\{f(\boldsymbol{\alpha})\}$ measures the maximum number of points that the classifier family can shatter, though not all sets of size d may necessarily be

shattered. For instance, a neural network with the function $f(\mathbf{x}, \mathbf{w}, \theta) = \text{sgn}(\mathbf{w}^T\mathbf{x} + \theta)$ is capable of shattering up to three points, giving it a VC-dimension of 3.

For linear classifiers, the VC-dimension equals the number of parameters in the model. The Boolean VC-dimension of a neural network, $\dim_{\text{BVC}}(\mathcal{N})$, refers to the VC-dimension of the Boolean functions it can compute.

To demonstrate that $\dim_{\text{VC}}(\mathcal{H}) = d$, one must show that a set of size d can be shattered, while no set of size $d + 1$ can. Neural networks with LTGs have a VC-dimension of $O(N_w \log N_w)$ [340], when having N_w weights. Networks with real-valued outputs have extended VC-dimensions [340].

For feedforward networks, those with threshold activations have a VC-dimension of $O(N_w \ln N_w)$ [336], and those with logistic activations have a VC-dimension of $O(N_w^2)$ [341]. When dealing with higher-order neurons involving k monomials over n variables, the VC-dimension is at least $nk + 1$ [342].

Consider a J_1-J_2-1 feedforward network with LTG output, where the hidden neurons are LTGs, binary RBF, and generalized binary RBF neurons in $\mathcal{N}_1$, $\mathcal{N}_2$, and $\mathcal{N}_3$, respectively. The VC-dimensions of these networks satisfy [266]

$$\dim_{\text{BVC}}(\mathcal{N}_1) = \dim_{\text{BVC}}(\mathcal{N}_2) \leq \dim_{\text{BVC}}(\mathcal{N}_3). \tag{38}$$

For $J_1 \geq 3$ and $J_2 \leq \frac{2^{J_1+1}}{J_1^2+J_1+2}$, the lower bound for these networks is [266,343]

$$\dim_{\text{BVC}}(\mathcal{N}_1) = J_1 J_2 + 1. \tag{39}$$

The VC-dimension estimates of a classifier set, though analytically impractical, can be experimentally derived by fitting formulas to error frequencies on artificial datasets of varying sizes [344]. Sampling variability was mitigated using a nonuniform design for better model complexity control and improved VC-generalization bounds [345]. An objective function is introduced for VC-dimension estimation [344], later optimized for regression model selection, showing consistency and competitive performance against established methods like AIC, BIC, and cross-validation [346].

Fully connected networks have VC-dimensions proportional to $WL \log(W/L)$, where W and L are the number of weights and depth, respectively [304,347]. Equivariant networks, such as CNNs, show lower VC-dimensions due to reduced dimensionality in equivariant matrices [348]. VC-dimension also depends on training methods like SGD [232,233].

CNNs exhibit translation equivariance [24], while group CNNs extend this to rotations, scales, and matrix symmetries [349]. Invariant kernels reduce sample complexity by the group size [350], with larger group volumes improving generalization [351]. The VC-dimension on group orbits relates to optimal sample complexity [352]. For simple architectures, precise VC-dimension estimates exist [353], though infinite groups with specific kernels can lead to infinite VC-dimension, even with group invariance.

Machine Teaching and Teaching Dimension

Machine teaching complements machine learning by selecting training data to optimize the learning process. Unlike machine learning, where a target concept c^* is inferred from random examples, machine teaching involves a teacher designing a labeled dataset $\mathcal{T}(c^*)$ to ensure the learner reconstructs c^*. Applications include explainable AI, trustworthy AI, and pedagogy.

Teaching dimension is a theoretical concept that determines the smallest training set size necessary for a learner to effectively approximate a target model. This concept has been explored in works such as [354,355]. In this setting, the teacher possesses knowledge of both the target model and the learners algorithm. The teacher's task is to select a training

set that helps the learner approximate the target model accurately. Importantly, the training examples do not need to be independent and identically distributed, and the teacher has the freedom to select any points from the input space.

For a finite hypothesis space $\mathcal{H}$, the relationship between the VC-dimension and teaching dimension is given by [354] as follows:

$$\dim_{\mathrm{VC}}(\mathcal{H}) = \log(|\mathcal{H}|) \leq \dim_{\mathrm{TD}}(\mathcal{H}) \leq \dim_{\mathrm{VC}}(\mathcal{H}) + |\mathcal{H}| - 2^{\dim_{\mathrm{VC}}(\mathcal{H})}. \tag{40}$$

Initially introduced for version-space learners, the concept of teaching dimension has since been expanded to machine learning models that select hypotheses via optimization. Studies such as [356] examine its application to models like ridge regression, SVMs, and logistic regression, identifying optimal training sets for these cases.

A learner L infers a hidden concept using priors $P(c)$ and c-conditional likelihoods $P(x|c)$, where $c \in C$ represents concepts and $x \in \mathcal{X}$ represents observations. L is a maximum a posteriori (MAP)-learner if it maximizes the posterior probability, or a maximum likelihood estimation (MLE)-learner if it maximizes the c-conditional likelihood [357]. Sampling modes depend on whether $\mathcal{S}$ is ordered or unordered, and with or without replacement.

For a target concept $c^* \in C$, a teacher aims to find the smallest set of observations ensuring L selects c^*, defining the MAP- and MLE-teaching dimensions [357]. These dimensions exhibit monotonicity, relate across sampling modes, and are characterized graph-theoretically for 0,1-labeled examples. The MLE-teaching dimension is at most one greater than the MAP-teaching dimension, bounded by combinatorial measures like the antichain number and VC-dimension, and computable in polynomial time.

12. Rademacher Complexity

In contrast to the VC-dimension, a global measure of complexity, Rademacher complexity [358], ref. [359] and other data-dependent metrics [255] offer more refined estimates by considering the data distribution, leading to tighter bounds and more accurate generalization estimates. Rademacher complexity, which measures a hypothesis space's ability to fit random labels for a training sample of size N, provides "average-case" bounds and is generally more effective than VC-dimension. Its precision, however, depends on the distribution approximation and empirical evaluation cost. In general, Rademacher complexity offers more precise generalization error bounds than VC-dimension and cover numbers.

Definition 8 (Rademacher complexity [358]). *Consider a family of functions $\mathcal{F}$ defined on a probability space $(\mathcal{Z}, \mathcal{P})$, with N independent training samples $\mathbf{z}_1, \ldots, \mathbf{z}_N$ drawn from $\mathcal{P}$. Let $\sigma_1, \ldots, \sigma_N$ represent independent Rademacher variables, each taking values ± 1 with equal likelihood. The Rademacher complexity of $\mathcal{F}$ is then defined as*

$$R_N\mathcal{F} = \sup_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} \sigma_i f(\mathbf{z}_i). \tag{41}$$

The expected Rademacher complexity is $E[R_N\mathcal{F}]$, and the empirical Rademacher complexity is

$$E_\sigma[R_N\mathcal{F}] = E[R_N\mathcal{F}|\mathbf{z}_1, \ldots, \mathbf{z}_N]. \tag{42}$$

The above formulation defines the global Rademacher complexity, which estimates the complexity of a function class and is used to derive generalization error bounds in supervised learning.

Theorem 9 (Generalization error bound from Rademacher complexity [359]). *For $\delta > 0$, consider a hypothesis $\hat{f}$ learned from N examples. With probability at least $1 - \delta$,*

$$E\left[\hat{f}\right] \leq \inf_{f \in \mathcal{F}} E[f] + 4R_N\mathcal{F} + \sqrt{\frac{2\log(2/\delta)}{N}}. \tag{43}$$

The bound in Theorem 9 converges at a rate of $O(\sqrt{1/N})$.

Local Rademacher complexity focuses on the subset of the hypothesis class likely selected by the learning algorithm, defined by the intersection of the hypothesis class with a ball around the target function,

$$E_\sigma\left[R_N\left\{f \in \mathcal{F} : E\left[f^2\right] \leq r\right\}\right] \quad \text{or} \quad E\left[R_N\left\{f \in \mathcal{F} : E\left[f^2\right] \leq r\right\}\right]. \tag{44}$$

The Rademacher complexity of a set of functions is upper-bounded by the growth function $G(N)$ and VC-dimension d [359,360],

$$R_N\mathcal{F} \leq \sqrt{\frac{2\ln G(N)}{N}}, \quad R_N\mathcal{F} \leq c\sqrt{\frac{d}{N}}, \tag{45}$$

where c is a constant. Thus, Rademacher complexity is of order $O(\sqrt{d/N})$.

Local Rademacher complexity, dependent on r, excludes high-variance functions, is always less than or equal to the global complexity, and offers tighter error bounds when the variance-expectation relationship is satisfied [361].

Theorem 10 (Generalization error bound using local Rademacher complexity [361]). *For a function $\hat{f}$ learned from N samples, where $E\left[f^2\right] \leq r$ for all $f \in \mathcal{F}$, and for $\delta > 0$, the following holds with probability at least $1 - \delta$,*

$$E[\hat{f}] \leq \inf_{f \in \mathcal{F}} E[f] + 8LR_N(\mathcal{F}, r) + \sqrt{\frac{8r\log(2/\delta)}{N}} + \frac{3\log(2/\delta)}{N}. \tag{46}$$

By selecting a smaller subset $\mathcal{F}' \subseteq \mathcal{F}$ that minimizes variance while ensuring $\hat{f} \in \mathcal{F}'$, the error bound in Theorem 10 achieves a faster convergence rate of $O\left(\frac{\log N}{N}\right)$ compared to Theorem 9. After determining the local Rademacher complexity, the error difference $E(\hat{f}) - \inf_{f \in \mathcal{F}} E[f]$ can be bounded using the fixed point of the local Rademacher complexity of $\mathcal{F}$.

Direct computation of Rademacher complexity is challenging. Dudley's entropy integral [362] relates covering numbers to Rademacher complexities, a result extended to local Rademacher complexity [363].

In ERM-based multi-label learning, the Rademacher complexity is constrained by the trace norm of the multi-label predictors, providing a rationale for applying trace norm regularization in multi-label problems [364]. The generalization of RBF networks has been analyzed using local Rademacher complexities with L_1-metric capacity, yielding improved error bounds [365].

Annealed VC-entropy [366] provides a bound for generalization error in terms of empirical error. The connection between VC-entropy and Rademacher complexity [360] allows the use of Rademacher complexity in Vapnik's general bound, yielding faster convergence rates from $O(N^{-1/2})$ to $O(N^{-1})$, surpassing the $O(N^{-1/2})$ rate with Rademacher complexity alone.

Local VC-entropy [367], a refined adaptation of VC-complexity, provides a generalization bound for binary classifiers while reducing computational demands compared to local Rademacher complexity. It quantifies the number of functions in $\mathcal{F}$, defined by $\mathcal{D}_N$, capable of classifying at least one configuration of $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_N)^T \in \mathcal{S}$ within a Hamming

distance of less than Nr. By eliminating improbable functions, this approach enhances Vapnik's results. The connection between local VC-entropy and Rademacher complexity is established through the determination of their admissible ranges [367], with further insights gained via extensions to the geometric framework [360].

13. Empirical Risk Minimization Principle

Consider N independent and identically distributed samples $(\mathbf{x}_i, y_i)$ drawn from an unknown distribution $p(\mathbf{x}, y)$. A machine defines mappings $\mathbf{x} \mapsto f(\mathbf{x}, \boldsymbol{\alpha})$, where $\boldsymbol{\alpha}$ are adjustable parameters. Upon selecting $\boldsymbol{\alpha}$, the machine becomes a trained machine.

The expected risk, representing the trained machine's generalization error, is defined as

$$R(\boldsymbol{\alpha}) = \int L(y, f(\mathbf{x}, \boldsymbol{\alpha}))\, \mathrm{d}p(\mathbf{x}, y), \tag{47}$$

Here, $L(y, f(\mathbf{x}, \boldsymbol{\alpha}))$ quantifies the discrepancy between the true output y and the model prediction $f(\mathbf{x}, \boldsymbol{\alpha})$, with its form tailored to the specific task, such as

$$L(y, f(\mathbf{x}, \boldsymbol{\alpha})) = \begin{cases} 0, & y = f(\mathbf{x}, \boldsymbol{\alpha}) \\ 1, & y \neq f(\mathbf{x}, \boldsymbol{\alpha}) \end{cases} \quad \text{(for classification)}, \tag{48}$$

$$L(y, f(\mathbf{x}, \boldsymbol{\alpha})) = (y - f(\mathbf{x}, \boldsymbol{\alpha}))^2 \quad \text{(for regression)}, \tag{49}$$

$$L(p(\mathbf{x}, \boldsymbol{\alpha})) = -\ln p(\mathbf{x}, \boldsymbol{\alpha}) \quad \text{(for density estimation)}. \tag{50}$$

The empirical risk $R_{\text{emp}}(\boldsymbol{\alpha})$ represents the average loss on the training set and is defined as

$$R_{\text{emp}}(\boldsymbol{\alpha}) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i, \boldsymbol{\alpha})). \tag{51}$$

The ERM principle minimizes the empirical risk (51) to approximate the true loss (47) with respect to model parameters. Common in statistics and machine learning, ERM supports models such as SVMs, linear regression, and logistic regression.

For a 0-1 loss function, the VC-bound [316] holds with confidence $1 - \delta$,

$$R(\boldsymbol{\alpha}) \leq R_{\text{emp}}(\boldsymbol{\alpha}) + \sqrt{\frac{d\left(\ln \frac{2N}{d} + 1\right) - \ln \frac{\delta}{4}}{N}}, \tag{52}$$

where d is the model's VC-dimension. The second term, the VC-confidence increases with d, and lowering d tightens the true error bound. VC-confidence depends on the function class, while empirical and true risks depend on the chosen function during training.

For regression problems, a useful form of the VC-bound is given by [9]

$$R(d) \leq R_{\text{emp}}(d) \left(1 - \sqrt{q - q \ln q + \frac{\ln N}{2N}}\right)^{-1}, \tag{53}$$

where $q = \frac{d}{N}$, with d as the VC-dimension and N as the training set size. Equation (53) is a specific instance of the general VC-bound [316], obtained by assigning suitable values to theoretical constants.

The SRM principle [316] extends ERM by incorporating VC-dimension to minimize the upper bound of true risk, improving generalization by selecting the function with the lowest worst-case risk. Empirical comparisons of AIC, BIC, and SRM in regression tasks [368] show SRM outperforms AIC and performs similarly to BIC in predictive accuracy. Excess risk bounds for loss functions that are unbounded, such as log loss and squared loss,

are derived in [369], focusing on estimators like η-generalized Bayesian, MDL, and ERM, particularly for heavy-tailed distributions.

13.1. Generalization Error by VC-Theory

VC-theory enforces regularization via a hierarchy of admissible models, known as a structure, which ranks models based on their complexity, measured by the VC-dimension. For a training sample containing N samples, the uniform convergence bound for classification is given by [9,232,233]

$$R_{\text{test}} \leq R_{\text{emp}} + \frac{1}{2}\sqrt{\epsilon}, \tag{54}$$

where

$$\epsilon = \frac{a_1}{N}\left(d\left[\ln\left(\frac{a_2 N}{d}\right)+1\right] - \ln\frac{\delta}{4}\right). \tag{55}$$

VC-theory also establishes a uniform relative convergence bound [9,232,233],

$$R_{\text{test}} \leq R_{\text{emp}} + \frac{\epsilon}{2}\left(1+\sqrt{1+\frac{4R_{\text{emp}}}{\epsilon}}\right). \tag{56}$$

VC-bounds (54) and (56) hold with probability $1-\delta$ for all models, including the one minimizing the training error (R_{emp}), with δ typically set as $\delta = \min\left(\frac{4}{\sqrt{N}}, 1\right)$ [232,233]. Bound (56) includes the training error and a confidence interval that depends on R_{emp} and ϵ (55), reflecting the model's capacity.

The constants $a_1 \in [0,4]$ and $a_2 \in [0,2]$ in (55) are usually set to $a_1 = 4$ and $a_2 = 2$ in classical VC-theory [9,233], though these worst-case values may lead to looser bounds for test error estimation [232]. For double descent modeling, $a_1 = 3$ and $a_2 = 1$ are preferred [254].

The uniform relative convergence bound (56) is more accurate than bound (54), particularly when R_{emp} is small (or zero) in the second descent mode [254]. When R_{emp} is zero, bound (54) gives $O(\sqrt{d/N})$, while (56) provides a sharper bound of $O(d/N)$.

VC-theory asserts that finite VC-dimension is both necessary and sufficient for generalization [9,233], and similar conditions apply to distribution-dependent complexity measures like VC-entropy, annealed VC-entropy, and Rademacher complexity [370].

13.2. Generalization Error by Rademacher Bound

Rademacher bounds on test error for classification, similar to the VC-bound (54), are widely discussed [359]. For binary functions $\mathcal{F}$ mapping $\mathbf{x}$ to $[-1,+1]$, with inputs $\mathbf{x}$ drawn from distribution $\mathcal{P}_x$, the following bound holds for n training samples, with probability $1-\delta$:

$$R_{\text{test}} \leq R_{\text{emp}} + aR_N\mathcal{F}(\mathcal{P}_x) + \sqrt{\frac{\ln(1/\delta)}{2N}}, \tag{57}$$

where $R_N\mathcal{F}(\mathcal{P}_x)$ represents the Rademacher complexity, and a denotes a theoretical constant (typically $a = 1$ or 2). Estimating Rademacher complexity for arbitrary function sets is computationally challenging, requiring either analytical estimation or a relation to VC-dimension.

Both the VC-bound (54) and Rademacher bound (57) provide the same excess error bound $R_{\text{test}} - R_{\text{emp}} < O(\sqrt{d/N})$, confirming their conceptual equivalence [371]. However, VC-bound (56) offers a more accurate estimate. A bound on Rademacher complexity for linear functions with bounded weights is provided [359,372]. It also aligns with the radius-margin bound in VC-theory for large-margin classifiers [359].

13.3. Fundamental Theorem of Learning Theory

The foundation of learning theory establishes PAC learnability for binary classifiers based on their VC-dimension. A hypothesis class is considered PAC learnable if and only if its VC-dimension is finite, which also dictates the sample complexity required for PAC learning. Uniform convergence of empirical error to the true error across all inputs guarantees that any training algorithm with low training error is PAC learnable. In cases where learnability holds, uniform convergence is assured, and the ERM principle can be applied.

Theorem 11 (Fundamental theorem of learning theory [373])**.** *Let $\mathcal{H}$ be a class of hypotheses, where each function maps from a domain $\mathcal{X}$ to $\{0,1\}$, and the loss function is the 0-1 loss. The following statements are equivalent:*

- *$\mathcal{H}$ exhibits uniform convergence.*
- *Any ERM rule can agnostically learn $\mathcal{H}$.*
- *$\mathcal{H}$ is agnostically PAC learnable.*
- *$\mathcal{H}$ has a finite VC-dimension.*

Agnostic PAC learning broadens the concept of PAC learnability to situations where the realizability assumption might not be valid. If realizability is assumed, agnostic PAC learning leads to the same results as PAC learning. Without realizability, a learner cannot guarantee an arbitrarily low error, but it can still perform well if its error is close to the best achievable by any predictor from $\mathcal{H}$.

When the VC-dimension of $\mathcal{H}$ is infinite, PAC learnability is not guaranteed. In contrast, a finite VC-dimension ensures PAC learnability, establishing VC-dimension as a crucial element in determining PAC learnability. The concept of VC-dimension also allows for the analysis of learnability in continuous domains and determines the sample complexity, as shown in Equation (36). The sample complexity is proportional to the VC-dimension, with further insights on this topic available in [373].

The SRM framework aims to select a hypothesis that minimizes the upper bound on the true risk. When applied to countable hypothesis classes, SRM leads to the MDL approach. In the SRM framework, prior knowledge is incorporated by encoding preferences for hypotheses in $\mathcal{H} = \bigcup_{n\in\mathbb{N}} \mathcal{H}_n$, where weights are assigned to each subclass $\mathcal{H}_n$.

Natarajan dimension generalizes the concept of VC-dimension to multiclass predictor classes, and the multiclass fundamental theorem is derived using Natarajan dimension [373].

14. Conclusions and Future Directions

The topics of machine learning is very wide, and in this paper, we have only introduced the fundamentals of machine learning that are related to neural networks. The contents are sufficient to support the understanding of the discipline of neural networks and deep learning. Interested readers can turn to some other review papers on related machine learning topics, such as kernel methods and SVMs [5], compressed sensing and matrix factorization [54], MLP and learning methods [374], and clustering [6], reinforcement learning, and ensemble learning, to name a few.

While this survey focuses on the foundational principles and theories of machine learning, several important topics were not fully covered due to space constraints. Challenges such as noisy data, overlapping data, highly dynamic data, and high-dimensional data, which directly impact learning and generalization, are critical in practical machine learning applications, but are primarily associated with data preprocessing and feature extraction, topics typically addressed within statistics. Additionally, emerging areas such as explain-

ability, interpretability, uncertainty quantification, and robust learning are gaining significant importance in developing trustworthy machine learning systems. Although these topics are essential for the application of machine learning models, they were not the central focus of this paper. Similarly, while recent advancements such as transfer learning, deep reinforcement learning, and transformer-based architectures significantly influence learning, inference, and generalization, they were mentioned briefly within the paper, but not explored in-depth due to the broad scope and theoretical focus of this survey. We encourage readers to explore these critical areas in other specialized literature.

14.1. Future Directions

Deep learning, often synonymous with modern machine learning, has ushered in the era of AI and transformative advancements. Below, we outline several fundamental issues that merit future investigation, many of which are central to the ongoing development of deep learning.

14.1.1. Analyzing Transfer Learning

Transfer learning has emerged as a dynamic and increasingly influential field in modern AI research. A wide range of transfer learning methods have been proposed, such as reweighting instances, adaptating features, adaptating classifiers, fine-tuning deep networks, and using adversarial strategies, which go beyond the conventional semi-supervised and unsupervised frameworks [22]. It has become the cornerstone of modern deep learning, operating on the heuristic of analogical learning [234].

Despite its significance, the theoretical foundations of transfer learning remain underdeveloped. Most theoretical work assumes a similarity between the source and target domains, yet this similarity is difficult to quantify, and the resulting theoretical frameworks often lack rigorous validation. Given its central role in deep learning, it is crucial to establish a deeper understanding of the principles and mechanisms underlying transfer learning, moving beyond reliance on empirical observations.

Future research should focus on uncovering the fundamental principles of transfer learning, developing metrics to evaluate domain similarity, and creating robust theoretical models to validate its effectiveness and applicability.

14.1.2. Explaining Double Descent

Understanding how neural networks learn remains one of the core challenges in machine learning. Neural network weights evolve from random initialization to effectively perform specific tasks. Despite their empirical successes, a comprehensive theoretical framework for neural networks is still lacking. The double descent phenomenon, observed in overparameterized models within deep learning, contradicts traditional theory, which suggests that models with too many parameters would overfit and generalize poorly. A central question is why neural networks can generalize well to unseen samples, despite their large number of parameters, which should theoretically lead to overfitting [230,375].

Various explanations have been proposed, but they often yield conflicting results. One explanation is that most of the nodes in a neural network remain inactive, with only a subset of the parameters being active, thus preventing overfitting. This idea is analogous to the way the human brain operates [234]. Additionally, training techniques like dropout and architectures like ResNet, which resemble ensemble learning, have been shown to improve generalization [234]. However, while these heuristics offer insights, they remain inadequate without a rigorous theoretical foundation.

Addressing this gap in understanding would not only improve the interpretability of neural networks, but also provide a principled approach to designing architectures that can leverage these insights effectively.

14.1.3. Exploring SGD in Deep Learning Setting

SGD is fundamental to modern machine learning, forming the basis for many optimization methods. Its simplicity and low computational complexity make it ideal for large-scale tasks, ensuring its continued importance in the field. By iteratively updating parameters using small data batches, SGD achieves efficient convergence and scalability across diverse applications, from linear regression to deep learning. SGD is still being actively investigated. Despite its success, challenges remain, particularly in deep learning. For instance, the behavior of SGD with dropout—a regularization technique—is not fully understood, complicating theoretical analysis. Future research may address these gaps, enhancing optimization strategies and ensuring SGD's relevance as machine learning evolves.

14.1.4. Understanding Data Augmentation

While data augmentation is widely recognized as a crucial technique for preventing overfitting, particularly in situations where datasets are small or labels are scarce, its implementation often lacks a consistent theoretical foundation. Practitioners apply data augmentation heuristically, relying on domain-specific insights or empirical validation to guide their choices. Despite its effectiveness, there is limited understanding of the underlying mechanisms that drive its success or the precise conditions under which it provides optimal benefits [376,377].

14.1.5. Delving into Transformers

Transformers, driven by the self-attention mechanism introduced by Vaswani et al. [298], have transformed modern AI, replacing recurrence in sequence processing to excel in tasks like language generation, image analysis, and speech recognition. Models like GPT-3 [306] leverage this innovation to achieve state-of-the-art performance in NLP by supporting long-range dependency modeling and efficient parallelization. Beyond NLP, Transformers have impacted vision (e.g., Vision Transformers for image classification) and bioinformatics, advancing protein analysis and genomics.

Key challenges include scaling for extreme data while addressing computational costs. Efficient variants, such as sparse Transformers, tackle resource efficiency, while methods like pruning and knowledge distillation adapt Transformers for low-resource and real-time applications. Questions remain on theoretical performance limits, minimal architectures for tasks, and generalization guarantees.

Integrating multimodal data (e.g., text, images, and audio) represents a promising but complex frontier. Ongoing research into scalability, efficiency, and multimodal learning ensures Transformers remain central to AI's evolution across diverse domains.

Author Contributions: The authors collaborated closely and contributed significantly to all aspects of the work. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analyzed in this study.

Conflicts of Interest: Author Rengong Zhang was employed by Zhejiang Yugong Information Technology Co., Ltd. Author Jie Zeng was employed by Shenzhen Feng Xing Tai Bao Technology Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

ADMM	alternating direction method of multipliers
AI	artificial intelligence
AIC	Akaike information criterion
ANOVA	analysis of variance
ANN	artificial neural network
BIC	Bayesian information criterion
BP	backpropagation
CCA	canonical correlation analysis
CNN	convolutional neural network
ERM	empirical risk minimization
FPE	final prediction error
GAN	generative adversarial network
GBDT	gradient-boosted decision trees
IALE	imitation-active learning ensemble
KL	Kullback–Leibler
k-NN	*k*-nearest neighbors
LASSO	least absolute shrinkage and selection operator
LLE	locally linear embedding
LMS	least mean squares
LS	least squares
LSM	liquid-state machines
LSTM	long short-term memory
LTG	linear threshold gate
LTL	learning to learn
MAD	median of absolute deviations
MAP	maximum a posteriori
MDL	minimum description length
MLE	maximum likelihood estimation
MLP	multilayer perceptrons
MSE	mean squared error
NMF	nonnegative matrix factorization
OOD	out-of-distribution
PAC	probably approximately correct
PCA	principal component analysis
pdf	probability density function
POMDP	partially observable Markov decision process
RBF	radial basis function
ReLU	rectified linear unit
RNN	recurrent neural network
SGD	stochastic gradient descent
SMOTE	synthetic minority oversampling technique
SNN	spiking neural network
SNRF	signal-to-noise ratio figure
SOM	self-organizing map
SRM	structural risk minimization
STDP	spike-timing dependent plasticity
SVD	singular value decomposition
SVM	support vector machine
UMM	universal memcomputing machines
VC	Vapnik–Chervonenkis
VLSI	very large scale integration
WTA	winner-takes-all

References

1. McCulloch, W.S.; Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **1943**, *5*, 115–133. [CrossRef]
2. Rosenblatt, F. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **1958**, *65*, 386–408. [CrossRef] [PubMed]
3. Du, K.-L.; Swamy, M.N.S. *Neural Networks and Statistical Learning*, 2nd ed.; Springer: London, UK, 2019.
4. Li, G.; Deng, L.; Tang, H.; Pan, G.; Tian, Y.; Roy, K.; Maass, W. Brain-inspired computing: A systematic survey and future trends. *Proc. IEEE* **2024**, *112*, 544–584. [CrossRef]
5. Du, K.-L.; Jiang, B.; Lu, J.; Hua, J.; Swamy, M.N.S. Exploring kernel machines and support vector machines: Principles, techniques, and future directions. *Mathematics* **2024**, *12*, 3935. [CrossRef]
6. Du, K.-L. Clustering: A neural network approach. *Neural Netw.* **2010**, *23*, 89–107. [CrossRef]
7. Du, K.-L.; Swamy, M.N.S. *Neural Networks in a Softcomputing Framework*; Springer: London, UK, 2006.
8. Du, K.-L.; Swamy, M.N.S. *Search and Optimization by Metaheuristics: Techniques and Algorithms Inspired by Nature*; Springer: New York, NY, USA, 2016.
9. Vapnik, V.N. *Statistical Learning Theory*; Wiley: New York, NY, USA, 1998.
10. Sarbo, J.J.; Cozijn, R. Belief in reasoning. *Cogn. Syst. Res.* **2019**, *55*, 245–256. [CrossRef]
11. Tecuci, G.; Kaiser, L.; Marcu, D.; Uttamsingh, C.; Boicu, M. Evidence-based reasoning in intelligence analysis: Structured methodology and system. *Comput. Sci. Eng.* **2018**, *20*, 9–21. [CrossRef]
12. Barto, A.G.; Sutton, R.S.; Anderson, C.W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man Cybern.* **1983**, *13*, 834–846. [CrossRef]
13. Schultz, W. Predictive reward signal of dopamine neurons. *J. Neurophysiol.* **1998**, *80*, 1–27. [CrossRef]
14. Belkin, M.; Niyogi, P.; Sindhwani, V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.* **2006**, *7*, 2399–2434.
15. Fedorov, V.V. *Theory of Optimal Experiments*; Academic Press: San Diego, CA, USA, 1972.
16. Sugiyama, M.; Nakajima, S. Pool-based active learning in approximate linear regression. *Mach. Learn.* **2009**, *75*, 249–274. [CrossRef]
17. Freund, Y.; Seung, H.S.; Shamir, E.; Tishby, N. Selective sampling using the query by committee algorithm. *Mach. Learn.* **1997**, *28*, 133–168. [CrossRef]
18. Wu, D. Pool-based sequential active learning for regression. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 1348–1359. [CrossRef] [PubMed]
19. MacKay, D. Information-based objective functions for active data selection. *Neural Comput.* **1992**, *4*, 590–604. [CrossRef]
20. Sugiyama, M.; Ogawa, H. Incremental active learning for optimal generalization. *Neural Comput.* **2000**, *12*, 2909–2940. [CrossRef]
21. Hoi, S.C.H.; Jin, R.; Lyu, M.R. Batch mode active learning with applications to text categorization and image retrieval. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 1233–1248. [CrossRef]
22. Zhang, L.; Gao, X. Transfer Adaptation Learning: A Decade Survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 23–44. [CrossRef]
23. Yang, L.; Hanneke, S.; Carbonell, J. A theory of transfer learning with applications to active learning. *Mach. Learn.* **2013**, *90*, 161–189. [CrossRef]
24. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*; Curran Associates, Inc.: New York, NY, USA, 2012; pp. 1097–1105.
25. Lampinen, A.K.; Ganguli, S. An analytic theory of generalization dynamics and transfer learning in deep linear networks. In Proceedings of the 7th International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
26. Ammar, H.B.; Eaton, E.; Luna, J.M.; Ruvolo, P. Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning. In Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI), Buenos Aires, Argentina, 25–31 July 2015; pp. 3345–3349.
27. Taylor, M.E.; Stone, P.; Liu, Y. Value functions for RL-based behavior transfer: A comparative study. In Proceedings of the 20th National Conference on Artificial Intelligence, AAAI, Pittsburgh, PA, USA, 9 July 2005; pp. 880–885.
28. Silva, F.; Costa, A. A survey on transfer learning for multiagent reinforcement learning systems. *J. Artif. Intell. Res.* **2019**, *64*, 645–703. [CrossRef]
29. Bard, N.; Foerster, J.N.; Chandar, S.; Burch, N.; Lanctot, M.; Song, H.F.; Parisotto, E.; Dumoulin, V.; Moitra, S.; Hughes, E.; et al. The Hanabi challenge: A new frontier for AI research. *Artif. Intell.* **2020**, *280*, 103216. [CrossRef]
30. Barrett, S.; Stone, P. Cooperating with unknown teammates in complex domains: A robot soccer case study of ad hoc teamwork. In Proceedings of the 29th AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; pp. 2010–2016.
31. Smith, M.O.; Anthony, T.; Wellman, M.P. Strategic Knowledge Transfer. *J. Mach. Learn. Res.* **2023**, *24*, 1–96.
32. Hu, X.; Zhang, X. Optimal parameter-transfer learning by semiparametric model averaging. *J. Mach. Learn. Res.* **2023**, *24*, 1–53.

33. Bastani, H. Predicting with proxies: Transfer learning in highdimension. *Manag. Sci.* **2021**, *67*, 2964–2984. [CrossRef]
34. Li, S.; Cai, T.T.; Li, H. Transfer learning for high-dimensional linear regression: Prediction, estimation and minimax optimality. *J. Roy. Stat. Soc. B* **2021**, *84*, 149–173. [CrossRef]
35. Tian, Y.; Feng, Y. Transfer learning under high-dimensional generalized linear models. *J. Am. Stat. Assoc.* **2023**, *118*, 2684–2697. [CrossRef]
36. Li, S.; Cai, T.T.; Li, H. Transfer learning in large-scale Gaussian graphical models with false discovery rate control. *J. Am. Stat. Assoc.* **2023**, *118*, 2171–2183. [CrossRef]
37. Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2009**, *22*, 1345–1359. [CrossRef]
38. Muandet, K.; Balduzzi, D.; Scholkopf, B. Domain generalization via invariant feature representation. In Proceedings of the International Conference on Machine Learning (ICML), Atlanta, GA, USA, 16–21 June 2013; Volume 28, pp. I.10–I.18.
39. Blanchard, G.; Deshmukh, A.A.; Dogan, U.; Lee, G.; Scott, C. Domain generalization by marginal transfer learning. *J. Mach. Learn. Res.* **2021**, *22*, 1–55.
40. Blanchard, G.; Lee, G.; Scott, C. Generalizing from several related classification tasks to a new unlabeled sample. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2011; Volume 24, pp. 2178–2186.
41. Thrun, S. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 1996; pp. 640–646.
42. Baxter, J. A model of inductive bias learning. *J. Artif. Intell. Res.* **2000**, *12*, 149–198. [CrossRef]
43. Denevi, G.; Ciliberto, C.; Stamos, D.; Pontil, M. Incremental learning-to-learn with statistical guarantees. In Proceedings of the Uncertainty in Artificial Intelligence (UAI), Monterey, CA, USA, 6–10 August 2018; pp. 457–466.
44. Maurer, A. Transfer bounds for linear feature learning. *Mach. Learn.* **2009**, *75*, 327–350. [CrossRef]
45. Pentina, A.; Lampert, C. A PAC-Bayesian bound for lifelong learning. In Proceedings of the International Conference on Machine Learning (ICML), Beijing, China, 21–26 June 2014; Volume 32, pp. 991–999.
46. Maurer, A.; Pontil, M.; Romera-Paredes, B. Sparse coding for multitask and transfer learning. In Proceedings of the International Conference on Machine Learning (ICML), Atlanta, GA, USA, 16–21 June 2013; Volume 28, pp. 343–351.
47. Li, Y.; Liang, Y. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2018; pp. 8168–8177.
48. McCullagh, P. Regression models for ordinal data. *J. Roy. Stat. Soc. B* **1980**, *42*, 109–142. [CrossRef]
49. Chapelle, O.; Chang, Y. Yahoo! learning to rank challenge overview. In Proceedings of the JMLR Workshop and Conference Proceedings: Workshop on Yahoo! Learning to Rank Challenge, San Francisco, CA, USA, 28 June 2011; Volume 14, pp. 1–24.
50. Herbrich, R.; Graepel, T.; Obermayer, K. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*; Bartlett, P.J., Scholkopf, B., Schuurmans, D., Smola, A.J., Eds.; MIT Press: Cambridge, MA, USA, 2000; pp. 115–132.
51. Burges, C.J.C. *From RankNet to LambdaRank to LambdaMART: An Overview*; Technical Report MSR-TR-2010-82; Microsoft Research: Redmond, WA, USA, 2010.
52. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
53. Freund, Y.; Iyer, R.; Schapire, R.E.; Singer, Y. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.* **2003**, *4*, 933–969.
54. Du, K.-L.; Swamy, M.N.S.; Wang, Z.-Q.; Mow, W.H. Matrix factorization techniques in machine learning, signal processing and statistics. *Mathematics* **2023**, *11*, 2674. [CrossRef]
55. Roweis, S.T.; Saul, L.K. Nonlinear dimensionality reduction by locally linear embedding. *Science* **2000**, *290*, 2323–2326. [CrossRef]
56. Belkin, M.; Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.* **2003**, *15*, 1373–1396. [CrossRef]
57. Kokiopoulou, E.; Saad, Y. Orthogonal neighborhood preserving projections: A projection-based dimensionality reduction technique. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 2143–2156. [CrossRef]
58. Caruana, R. Multitask learning. *Mach. Learn.* **1997**, *28*, 41–75. [CrossRef]
59. Evgeniou, T.; Michelli, C.A.; Pontil, M. Learning multiple tasks with kernel methods. *J. Mach. Learn. Res.* **2005**, *6*, 615–637.
60. Yang, X.; Kim, S.; Xing, E.P. Heterogeneous multitask learning with joint sparsity constraints. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2009; pp. 2151–2159.
61. Kaelbling, L.P. Learning to achieve goals. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Chambery, France, 28 August–3 September 1993; pp. 1094–1099.
62. Ruder, S. An overview of multi-task learning in deep neural networks. *arXiv* **2017**, arXiv:1706.05098.
63. Jong, N.K.; Stone, P. State abstraction discovery from irrelevant state variables. In Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI), Edinburgh, Scotland, UK, 30 July–5 August 2005; pp. 752–757.
64. Walsh, T.J.; Li, L.; Littman, M.L. Transferring state abstractions between MDPs. In Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning, Pittsburgh, PA, USA, 25 June 2006.
65. Foster, D.; Dayan, P. Structure in the space of value functions. *Mach. Learn.* **2002**, *49*, 325–346. [CrossRef]

66. Konidaris, G.; Barto, A. Autonomous shaping: Knowledge transfer in reinforcement learning. In Proceedings of the 23rd International Conference on Machine Learning (ICML), Pittsburgh, PA, USA, 25–29 June 2006; pp. 489–496.
67. Snel, M.; Whiteson, S. Learning potential functions and their representations for multi-task reinforcement learning. In Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Paris, France, 5–9 May 2014; pp. 637–681.
68. Czarnecki, W.; Jayakumar, S.; Jaderberg, M.; Hasenclever, L.; Teh, Y.W.; Heess, N.; Osindero, S.; Pascanu, R. Mix & match agent curricula for reinforcement learning. In Proceedings of the 35th International Conference on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018; pp. 1087–1095.
69. Pomerleau, D.A. *Alvinn: An Autonomous Land Vehicle in a Neural Network*; Technical Report; Carnegie-Mellon University: Pittsburgh, PA, USA, 1989.
70. Arora, S.; Doshi, P. A survey of inverse reinforcement learning: Challenges, methods, and progress. *Artif. Intell.* **2021**, *297*, 103500. [CrossRef]
71. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2014; Volume 27, pp. 2672–2680.
72. Grill, J.-B.; Strub, F.; Altche, F.; Tallec, C.; Richemond, P.; Buchatskaya, E.; Doersch, C.; Avila Pires, B.; Guo, Z.; Gheshlaghi Azar, M.; et al. Bootstrap your own latent: A new approach to self-supervised learning. In Proceedings of the Conference on Neural Information Processing Systems, Vitual, 6–12 December 2020; pp. 21271–21284.
73. Syed, U.; Schapire, R.E. A reduction from apprenticeship learning to classification. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2010; Volume 23, pp. 2253–2261.
74. Ross, S.; Gordon, G.; Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS), Ft. Lauderdale, FL, USA, 11–13 April 2011; pp. 627–635.
75. Cohen, M.K.; Hutter, M.; Nanda, N. Fully General Online Imitation Learning. *J. Mach. Learn. Res.* **2022**, *23*, 1–30.
76. Loffler, C.; Mutschler, C. IALE: Imitating active learner ensembles. *J. Mach. Learn. Res.* **2022**, *23*, 1–29.
77. Schmidhuber, J. Curious model-building control systems. In Proceedings of the IEEE International Joint Conference on Neural Networks, Seoul, Republic of Korea, 17–21 June 1991; pp. 1458–1463.
78. Schmidhuber, J. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Trans. Auton. Ment. Develop.* **2010**, *2*, 230–247. [CrossRef]
79. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum learning. In Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; pp. 41–48.
80. Gong, C.; Tao, D.; Liu, W.; Liu, L.; Yang, J. Label propagation via teaching-to-learn and learning-to-teach. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 1452–1465. [CrossRef]
81. Zaremba, W.; Sutskever, I. Learning to execute. *arXiv* **2014**, arXiv:1410.4615.
82. Kaelbling, L.P.; Littman, M.L.; Cassandra, A.R. Planning and acting in partially observable stochastic domains. *Artif. Intell.* **1998**, *101*, 99–134. [CrossRef]
83. Bellemare, M.; Srinivasan, S.; Ostrovski, G.; Schaul, T.; Saxton, D.; Munos, R. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2016; pp. 1471–1479.
84. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
85. Langford, J. Efficient exploration in reinforcement learning. In *Encyclopedia Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 309–311.
86. Sukhbaatar, S.; Lin, Z.; Kostrikov, I.; Synnaeve, G.; Szlam, A.; Fergus, R. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv* **2017**, arXiv:1703.05407.
87. Held, D.; Geng, X.; Florensa, C.; Abbeel, P. Automatic goal generation for reinforcement learning agents. *arXiv* **2017**, arXiv:1705.06366.
88. Matiisen, T.; Oliver, A.; Cohen, T.; Schulman, J. Teacher–student curriculum learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 3732–3740. [CrossRef] [PubMed]
89. Graves, A.; Bellemare, M.G.; Menick, J.; Munos, R.; Kavukcuoglu, K. Automated curriculum learning for neural networks. *arXiv* **2017**, arXiv:1704.03003.
90. Du, K.-L.; Swamy, M.N.S. *Wireless Communication Systems*; Cambridge University Press: Cambridge, UK, 2010.
91. Dasgupta, S.; Littman, M.; McAllester, D. PAC generalization bounds for co-training. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2002; Volume 14, pp. 375–382.
92. Hotelling, H. Relations between two sets of variates. *Biometrika* **1936**, *28*, 321–377. [CrossRef]
93. Kettenring, J. Canonical analysis of several sets of variables. *Biometrika* **1971**, *58*, 433–451. [CrossRef]

94. Tucker, L.R. The extension of factor analysis to three-dimensional matrices. In *Contributions to Mathematical Psychology*; Holt, Rinehardt & Winston: New York, NY, USA, 1964; pp. 109–127.
95. Lanckriet, G.R.G.; Cristianini, N.; Bartlett, P.; El Ghaoui, L.; Jordan, M.I. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.* **2004**, *5*, 27–72.
96. Zhang, M.-L.; Zhou, Z.-H. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recogn.* **2007**, *40*, 2038–2048. [CrossRef]
97. Dietterich, T.G.; Lathrop, R.H.; Lozano-Perez, T. Solving the multiple instance problem with axis-parallel rectangles. *Artif. Intell.* **1997**, *89*, 31–71. [CrossRef]
98. Sabato, S.; Tishby, N. Multi-instance learning with any hypothesis class. *J. Mach. Learn. Res.* **2012**, *13*, 2999–3039.
99. Parzen, E. On estimation of a probability density function and mode. *Ann. Math. Stat.* **1962**, *33*, 1065–1076. [CrossRef]
100. Chawla, N.; Bowyer, K.; Kegelmeyer, W. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
101. Lin, Y.; Lee, Y.; Wahba, G. Support vector machines for classification in nonstandard situations. *Mach. Learn.* **2002**, *46*, 191–202. [CrossRef]
102. Wu, G.; Cheng, E. Class-boundary alignment for imbalanced dataset learning. In Proceedings of the ICML 2003 Workshop Learning Imbalanced Data Sets II, Washington, DC, USA, 21 August 2003; pp. 49–56.
103. Chao, W.L.; Changpinyo, S.; Gong, B.; Sha, F. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 11–14 October 2016; pp. 52–68.
104. Lampert, C.H.; Nickisch, H.; Harmeling, S. Learning to detect unseen object classes by between-class attribute transfer. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Miami, FL, USA, 20–26 June 2009; pp. 951–958.
105. Rahman, S.; Khan, S.; Porikli, F. A unified approach for conventional zero-shot, generalized zero-shot, and few-shot learning. *IEEE Trans. Image Process.* **2018**, *27*, 5652–5667. [CrossRef] [PubMed]
106. Wang, D.; Li, Y.; Lin, Y.; Zhuang, Y. Relational knowledge transfer for zero-shot learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; pp. 1–7.
107. Tian, P.; Li, W.; Gao, Y. Consistent meta-regularization for better meta-knowledge in few-shot learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 7277–7288. [CrossRef] [PubMed]
108. Rumelhart, D.E.; Durbin, R.; Golden, R.; Chauvin, Y. Backpropagation: The basic theory. In *Backpropagation: Theory, Architecture, and Applications*; Chauvin, Y., Rumelhart, D.E., Eds.; Lawrence Erlbaum: Hillsdale, NJ, USA, 1995; pp. 1–34.
109. Baum, E.B.; Wilczek, F. Supervised learning of probability distributions by neural networks. In *Neural Information Processing Systems*; Anderson, D.Z., Ed.; American Institute Physics: New York, NY, USA, 1988; pp. 52–61.
110. Matsuoka, K.; Yi, J. Backpropagation based on the logarithmic error function and elimination of local minima. In Proceedings of the International Joint Conference on Neural Networks, Seattle, WA, USA, 8–12 July 1991; pp. 1117–1122.
111. Solla, S.A.; Levin, E.; Fleisher, M. Accelerated learning in layered neural networks. *Complex Syst.* **1988**, *2*, 625–640.
112. Blum, A.L.; Rivest, R.L. Training a 3-node neural network is NP-complete. *Neural Netw.* **1992**, *5*, 117–127. [CrossRef]
113. Sima, J. Back-propagation is not efficient. *Neural Netw.* **1996**, *9*, 1017–1023. [CrossRef]
114. Auer, P.; Herbster, M.; Warmuth, M.K. Exponentially many local minima for single neurons. In *Advances in Neural Information Processing Systems*; Touretzky, D.S., Mozer, M.C., Hasselmo, M.E., Eds.; MIT Press: Cambridge, MA, USA, 1996; Volume 8, pp. 316–322.
115. Bartlett, P.L. The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. *IEEE Trans. Inf. Theory* **1998**, *44*, 525–536. [CrossRef]
116. Gish, H. A probabilistic approach to the understanding and training of neural network classifiers. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Albuquerque, NM, USA, 3–6 April 1990; pp. 1361–1364.
117. Hinton, G.E. Connectionist learning procedure. *Artif. Intell.* **1989**, *40*, 185–234. [CrossRef]
118. Rimer, M.; Martinez, T. Classification-based objective functions. *Mach. Learn.* **2006**, *63*, 183–205. [CrossRef]
119. Hanson, S.J.; Burr, D.J. Minkowski back-propagation: Learning in connectionist models with non-Euclidean error signals. In *Neural Information Processing Systems*; Anderson, D.Z., Ed.; American Institute Physics: New York, NY, USA, 1988; pp. 348–357.
120. Silva, L.M.; de Sa, J.M.; Alexandre, L.A. Data classification with multilayer perceptrons using a generalized error function. *Neural Netw.* **2008**, *21*, 1302–1310. [CrossRef] [PubMed]
121. Liu, W.; Pokharel, P.P.; Principe, J.C. Correntropy: Properties and applications in non-Gaussian signal processing. *IEEE Trans. Signal Process.* **2007**, *55*, 5286–5298. [CrossRef]
122. Huber, P.J. *Robust Statistics*; Wiley: New York, NY, USA, 1981.
123. Poggio, T.; Girosi, F. Networks for approximation and learning. *Proc. IEEE* **1990**, *78*, 1481–1497. [CrossRef]

124. Hui, L.; Belkin, M. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. In Proceedings of the 9th International Conference on Learning Representations (ICLR), Virtually, 3–7 May 2021.
125. Cichocki, A.; Unbehauen, R. *Neural Networks for Optimization and Signal Processing*; Wiley: New York, NY, USA, 1992.
126. Chen, D.S.; Jain, R.C. A robust backpropagation learning algorithm for function approximation. *IEEE Trans. Neural Netw.* **1994**, *5*, 467–479. [CrossRef]
127. Tabatabai, M.A.; Argyros, I.K. Robust estimation and testing for general nonlinear regression models. *Appl. Math. Comput.* **1993**, *58*, 85–101. [CrossRef]
128. Singh, A.; Pokharel, R.; Principe, J.C. The C-loss function for pattern classification. *Pattern Recogn.* **2014**, *47*, 441–453. [CrossRef]
129. Tikhonov, A.N. On solving incorrectly posed problems and method of regularization. *Dokl. Akad. Nauk USSR* **1963**, *151*, 501–504.
130. Widrow, B.; Lehr, M.A. 30 years of adaptive neural networks: Perceptron, Madaline, and backpropagation. *Proc. IEEE* **1990**, *78*, 1415–1442. [CrossRef]
131. Niyogi, P.; Girosi, F. Generalization bounds for function approximation from scattered noisy data. *Adv. Comput. Math.* **1999**, *10*, 51–80. [CrossRef]
132. Barron, A.R. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. Inf. Theory* **1993**, *39*, 930–945. [CrossRef]
133. Shamir, O. Gradient methods never overfit on separable data. *J. Mach. Learn. Res.* **2021**, *22*, 1–20.
134. Prechelt, L. Automatic early stopping using cross validation: Quantifying the criteria. *Neural Netw.* **1998**, *11*, 761–767. [CrossRef] [PubMed]
135. Amari, S.; Murata, N.; Muller, K.R.; Finke, M.; Yang, H. Statistical theory of overtraining: Is cross-validation asymptotically effective? In *Advances in Neural Information Processing Systems*; Touretzky, D.S., Mozer, M.C., Hasselmo, M.E., Eds.; MIT Press: Cambridge, MA, USA, 1996; Volume 8, pp. 176–182.
136. Liu, Y.; Starzyk, J.A.; Zhu, Z. Optimized approximation algorithm in neural networks without overfitting. *IEEE Trans. Neural Netw.* **2008**, *19*, 983–995.
137. Hu, T.; Lei, Y. Early stopping for iterative regularization with general loss functions. *J. Mach. Learn. Res.* **2022**, *23*, 1–36.
138. Geman, S.; Bienenstock, E.; Doursat, R. Neural networks and the bias/variance dilemma. *Neural Comput.* **1992**, *4*, 1–58. [CrossRef]
139. Bishop, C.M. *Neural Networks for Pattern Recognition*; Oxford Press: New York, NY, USA, 1995.
140. Bishop, C.M. Training with noise is equivalent to Tikhonov regularization. *Neural Comput.* **1995**, *7*, 108–116. [CrossRef]
141. Reed, R.; Marks, R.J., II; Oh, S. Similarities of error regularization, sigmoid gain scaling, target smoothing, and training with jitter. *IEEE Trans. Neural Netw.* **1995**, *6*, 529–538. [CrossRef]
142. Holmstrom, L.; Koistinen, P. Using additive noise in back-propagation training. *IEEE Trans. Neural Netw.* **1992**, *3*, 24–38. [CrossRef]
143. Hinton, G.E.; van Camp, D. Keeping neural networks simple by minimizing the description length of the weights. In Proceedings of the 6th Annual ACM Conference on Computational Learning Theory, Santa Cruz, CA, USA, 26–28 July 1993; pp. 5–13.
144. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*; Rumelhart, D.E., McClelland, J.L., Eds.; MIT Press: Cambridge, MA, USA, 1986; Volume 1, pp. 318–362.
145. Nowlan, S.J.; Hinton, G.E. Simplifying neural networks by soft weight-sharing. *Neural Comput.* **1992**, *4*, 473–493. [CrossRef]
146. Sun, H.; Gatmiry, K.; Ahn, K.; Azizan, N. A unified approach to controlling implicit regularization via mirror descent. *J. Mach. Learn. Res.* **2023**, *24*, 1–58.
147. Stankewitz, B.; Mucke, N.; Rosasco, L. From inexact optimization to learning via gradient concentration. *Comput. Optim. Appl.* **2023**, *84*, 265–294. [CrossRef]
148. Neyshabur, B.; Tomioka, R.; Srebro, N. In search of the real inductive bias: On the role of implicit regularization in deep learning. In Proceedings of International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 April 2015.
149. Rosasco, L.; Villa, S. Learning with incremental iterative regularization. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Redhook, NY, USA, 2015; Volume 28.
150. Wei, Y.T.; Yang, F.; Wainwright, M.J. Early stopping for kernel boosting algorithms: A general analysis with localized complexities. *IEEE Trans. Inf. Theory* **2019**, *65*, 6685–6703. [CrossRef]
151. Erhan, D.; Bengio, Y.; Courville, A.; Manzagol, P.-A.; Vincent, P.; Bengio, S. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.* **2010**, *11*, 625–660.
152. Yao, Y.; Yu, B.; Gong, C.; Liu, T. Understanding how pretraining regularizes deep learning algorithms. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *34*, 5828–5840. [CrossRef]
153. Chen, S.; Dobriban, E.; Lee, J.H. A group-theoretic framework for data augmentation. *J. Mach. Learn. Res.* **2020**, *21*, 1–71.
154. Dao, T.; Gu, A.; Ratner, A.; Smith, V.; De Sa, C.; Re, C. A kernel theory of modern data augmentation. In Proceedings of the International Conference on Machine Learning (ICML), Long Beach, CA, USA, 9–15 June 2019; pp. 1528–1537.

155. He, K.; Chen, X.; Xie, S.; Li, Y.; Dollar, P.; Girshick, R. Masked autoencoders are scalable vision learners. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 19–24 June 2022; pp. 16000–16009.
156. DeVries, T.; Taylor, G.W. Improved regularization of convolutional neural networks with cutout. *arXiv* **2017**, arXiv:1708.04552.
157. Chapelle, O.; Weston, J.; Bottou, L.; Vapnik, V. Vicinal risk minimization. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2001; pp. 416–422.
158. Zhang, J.; Cho, K. Query-efficient imitation learning for end-to-end simulated driving. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; Volume 31.
159. Lin, C.-H.; Kaushik, C.; Dyer, E.L.; Muthukumar, V. The good, the bad and the ugly sides of data augmentation: An implicit spectral regularization perspective. *J. Mach. Learn. Res.* **2024**, *25*, 1–85.
160. Bartlett, P.L.; Long, P.M.; Lugosi, G.; Tsigler, A. Benign overfitting in linear regression. *Proc. Natl. Acad. Sci. USA* **2020**, *117*, 30063–30070. [CrossRef]
161. Muthukumar, V.; Narang, A.; Subramanian, V.; Belkin, M.; Hsu, D.; Sahai, A. Classification vs regression in overparameterized regimes: Does the loss function matter? *J. Mach. Learn. Res.* **2021**, *22*, 1–69.
162. Shen, R.; Bubeck, S.; Gunasekar, S. Data augmentation as feature manipulation: A story of desert cows and grass cows. *arXiv* **2022**, arXiv:2203.01572.
163. Wu, D.; Xu, J. On the optimal weighted 2 regularization in overparameterized linear regression. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2020; pp. 10112–10123.
164. LeJeune, D.; Balestriero, R.; Javadi, H.; Baraniuk, R.G. Implicit rugosity regularization via data augmentation. *arXiv* **2019**, arXiv:1905.11639.
165. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
166. Wan, L.; Zeiler, M.; Zhang, S.; LeCun, Y.; Fergus, R. Regularization of neural networks using dropconnect. In Proceedings of the 30th International Conference on Machine Learning (ICML), Atlanta, GA, USA, 16–21 June 2013; pp. 1058–1066.
167. Arora, R.; Bartlett, P.; Mianjy, P.; Srebro, N. Dropout: Explicit forms and capacity control. In Proceedings of the 38th International Conference on Machine Learning (ICML), Virtually, 18–24 July 2021; pp. 351–361.
168. Baldi, P.; Sadowski, P. Understanding dropout. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2013; Volume 27, pp. 2814–2822.
169. Cavazza, J.; Morerio, P.; Haeffele, B.; Lane, C.; Murino, V.; Vidal, R. Dropout as a low-rank regularizer for matrix factorization. In Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS), Grimma, Germany, 9–11 April 2018; Volume 84, pp. 435–444.
170. McAllester, D. A PAC-Bayesian tutorial with a dropout bound. *arXiv* **2013**, arXiv:1307.2118.
171. Mianjy, P.; Arora, R. On dropout and nuclear norm regularization. In Proceedings of the 36th International Conference on Machine Learning (ICML), Long Beach, CA, USA, 9–15 June 2019; pp. 4575–4584.
172. Senen-Cerda, A.; Sanders, J. Asymptotic convergence rate of dropout on shallow linear neural networks. In Proceedings of the ACM Measurement and Analysis of Computing Systems, Rome, Italy, 5–8 April 2022; Volume 6, pp. 32:1–32:53.
173. Wager, S.; Wang, S.; Liang, P. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems 26*; Curran Associates, Inc.: New York, NY, USA, 2013; pp. 351–359.
174. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.R. Improving neural networks by preventing co-adaptation of feature detectors. *Comput. Sci.* **2012**, *3*, 212–223.
175. Hinton, G.E. Dropout: A Simple and Effective Way to Improve Neural Networks. 2012. Available online: https://videolectures.net (accessed on 14 December 2024).
176. Gal, Y.; Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of the 33rd International Conference on Machine Learning (ICML), New York, NY, USA, 19–24 June 2016; Volume 48, pp. 1050–1059.
177. Helmbold, D.P.; Long, P.M. Surprising properties of dropout in deep networks. *J. Mach. Learn. Res.* **2018**, *18*, 1–28.
178. Khan, S.H.; Hayat, M.; Porikli, F. Regularization of deep neural networks with spectral dropout. *Neural Netw.* **2019**, *110*, 82–90. [CrossRef]
179. Kendall, A.; Gal, Y. What uncertainties do we need in Bayesian deep learning for computer vision? In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; pp. 5574–5584.
180. Sicking, J.; Akila, M.; Pintz, M.; Wirtz, T.; Wrobel, S.; Fischer, A. Wasserstein dropout. *Mach. Learn.* **2024**, *113*, 3161–3204. [CrossRef]
181. Li, H.; Weng, J.; Mao, Y.; Wang, Y.; Zhan, Y.; Cai, Q.; Gu, W. Adaptive dropout method based on biological principles. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 4267–4276. [CrossRef]

182. Clara, G.; Langer, S.; Schmidt-Hieber, J. Dropout regularization versus ℓ_2-penalization in the linear model. *J. Mach. Learn. Res.* **2024**, *25*, 1–48.
183. Gao, W.; Zhou, Z.-H. Dropout Rademacher complexity of deep neural networks. *Sci. China Inf. Sci.* **2016**, *59*, 072104. [CrossRef]
184. Zhai, K.; Wang, H. Adaptive dropout with Rademacher complexity regularization. In Proceedings of the 6th International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
185. Mianjy, P.; Arora, R. On convergence and generalization of dropout training. In *Advances in Neural Information Processing Systems 33*; Curran Associates, Inc.: New York, NY, USA, 2020; pp. 21151–21161.
186. Blanchet, J.; Kang, Y.; Olea, J.L.M.; Nguyen, V.A.; Zhang, X. Dropout training is distributionally robust optimal. *J. Mach. Learn. Res.* **2023**, *24*, 1–60.
187. Murray, A.F.; Edwards, P.J. Synaptic weight noise euring MLP training: Enhanced MLP performance and fault tolerance resulting from synaptic weight noise during training. *IEEE Trans. Neural Netw.* **1994**, *5*, 792–802. [CrossRef] [PubMed]
188. Chiu, C.; Mehrotra, K.; Mohan, C.K.; Ranka, S. Modifying training algorithms for improved fault tolerance. In Proceedings of the IEEE International Conference on Neural Networks, Orlando, FL, USA, 26–29 June 1994; Volume 4, pp. 333–338.
189. Edwards, P.J.; Murray, A.F. Towards optimally distributed computation. *Neural Comput.* **1998**, *10*, 997–1015. [CrossRef] [PubMed]
190. Bernier, J.L.; Ortega, J.; Ros, E.; Rojas, I.; Prieto, A. A quantitative study of fault tolerance, noise immunity, and generalization ability of MLPs. *Neural Comput.* **2000**, *12*, 2941–2964. [CrossRef]
191. Krogh, A.; Hertz, J.A. A simple weight decay improves generalization. In Proceedings of the Neural Information Processing Systems (NIPS) Conference, Denver, CO, USA, 7–10 December 1992; Morgan Kaufmann: San Mateo, CA, USA, 1992; pp. 950–957.
192. Phatak, D.S. Relationship between fault tolerance, generalization and the Vapnik-Cervonenkis (VC) dimension of feedforward ANNs. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Washington, DC, USA, 10–14 July 1999; Volume 1, pp. 705–709.
193. Sum, J.P.-F.; Leung, C.-S.; Ho, K.I.-J. On-line node fault injection training algorithm for MLP networks: Objective function and convergence analysis. *IEEE Trans. Neural Netw. Learn. Syst.* **2012**, *23*, 211–222. [CrossRef] [PubMed]
194. Ho, K.I.-J.; Leung, C.-S.; Sum, J. Convergence and objective functions of some fault/noise-injection-based online learning algorithms for RBF networks. *IEEE Trans. Neural Netw.* **2010**, *21*, 938–947. [CrossRef]
195. Xiao, Y.; Feng, R.-B.; Leung, C.-S.; Sum, J. Objective function and learning algorithm for the general node fault situation. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *27*, 863–874. [CrossRef]
196. Bousquet, O.; Elisseeff, A. Stability and Generalization. *J. Mach. Learn. Res.* **2002**, *2*, 499–526.
197. Xu, H.; Caramanis, C.; Mannor, S. Robust regression and Lasso. *IEEE Trans. Inf. Theory* **2010**, *56*, 3561–3574. [CrossRef]
198. Xu, H.; Caramanis, C.; Mannor, S. Sparse algorithms are not stable: A no-free-lunch theorem. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 187–193.
199. Domingos, P. The role of Occam's razor in knowledge discovery. *Data Min. Knowl. Disc.* **1999**, *3*, 409–425. [CrossRef]
200. Zahalka, J.; Zelezny, F. An experimental test of Occam's razor in classification. *Mach. Learn.* **2011**, *82*, 475–481. [CrossRef]
201. Janssen, P.; Stoica, P.; Soderstrom, T.; Eykhoff, P. Model structure selection for multivariable systems by cross-validation. *Int. J. Contr.* **1988**, *47*, 1737–1758. [CrossRef]
202. Stone, M. Cross-validatory choice and assessment of statistical predictions. *J. R. Stat. Soc. B* **1974**, *36*, 111–147. [CrossRef]
203. Bengio, Y.; Grandvalet, Y. No unbiased estimator of the variance of *K*-fold cross-validation. *J. Mach. Learn. Res.* **2004**, *5*, 1089–1105.
204. Markatou, M.; Tian, H.; Biswas, S.; Hripcsak, G. Analysis of variance of cross-validation estimators of the generalization error. *J. Mach. Learn. Res.* **2005**, *6*, 1127–1168.
205. Arlot, S.; Lerasle, M. Choice of V for V-fold cross-validation in least-squares density estimation. *J. Mach. Learn. Res.* **2016**, *17*, 1–50.
206. Breiman, L.; Spector, P. Submodel selection and evaluation in regression: The X-random case. *Int. Statist. Rev.* **1992**, *60*, 291–319. [CrossRef]
207. Plutowski, M.E.P. *Survey: Cross-Validation in Theory and in Practice*; Research Report; Department of Computational Science Research, David Sarnoff Research Center: Princeton, NJ, USA, 1996.
208. Shao, J. Linear model selection by cross-validation. *J. Am. Stat. Assoc.* **1993**, *88*, 486–494. [CrossRef]
209. Nadeau, C.; Bengio, Y. Inference for the generalization error. *Mach. Learn.* **2003**, *52*, 239–281. [CrossRef]
210. Akaike, H. Fitting autoregressive models for prediction. *Ann. Inst. Stat. Math.* **1969**, *21*, 425–439. [CrossRef]
211. Akaike, H. A new look at the statistical model identification. *IEEE Trans. Auto. Contr.* **1974**, *19*, 716–723. [CrossRef]
212. Schwarz, G. Estimating the dimension of a model. *Ann. Statist.* **1978**, *6*, 461–464. [CrossRef]
213. Rissanen, J. Modeling by shortest data description. *Automatica* **1978**, *14*, 465–477. [CrossRef]
214. Stoica, P.; Selen, Y. A review of information criterion rules. *IEEE Signal Process. Mag.* **2004**, *21*, 36–47. [CrossRef]
215. Akaike, H. Statistical prediction information. *Ann. Inst. Stat. Math.* **1970**, *22*, 203–217. [CrossRef]
216. Rissanen, J. Hypothesis selection and testing by the MDL principle. *Comput. J.* **1999**, *42*, 260–269. [CrossRef]
217. Ghodsi, A.; Schuurmans, D. Automatic basis selection techniques for RBF networks. *Neural Netw.* **2003**, *16*, 809–816. [CrossRef]

218. Cawley, G.C.; Talbot, N.L.C. On over-fitting in model selection and subsequent selection bias in performance evaluation. *J. Mach. Learn. Res.* **2010**, *11*, 2079–2107.
219. Cawley, G.C.; Talbot, N.L.C. Preventing over-fitting during model selection via Bayesian regularisation of the hyper-parameters. *J. Mach. Learn. Res.* **2007**, *8*, 841–861.
220. Nelson, B.L. Control variate remedies. *Oper. Res.* **1990**, *38*, 974–992. [CrossRef]
221. Glynn, P.W.; Szechtman, R. Some new perspectives on the method of control variates. In *Monte Carlo and Quasi-Monte Carlo Methods 2000*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 27–49.
222. Portier, F.; Segers, J. Monte Carlo integration with a growing number of control variates. *J. Appl. Prob.* **2018**, *56*, 1168–1186. [CrossRef]
223. South, L.F.; Oates, C.J.; Mira, A.; Drovandi, C. Regularized zero-variance control variates. *Bayes. Anal.* **2023**, *18*, 865–888. [CrossRef]
224. Deng, Z.; Kammoun, A.; Thrampoulidis, C. A model of double descent for high-dimensional binary linear classification. *arXiv* **2019**, arXiv:1911.05822. [CrossRef]
225. Kini, G.R.; Thrampoulidis, C. Analytic study of double descent in binary classification: The impact of loss. In Proceedings of the 2020 IEEE International Symposium on Information Theory (ISIT), Virtually, 12–17 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 2527–2532.
226. Montanari, A.; Ruan, F.; Sohn, Y.; Yan, J. The generalization error of max-margin linear classifiers: High-dimensional asymptotics in the overparametrized regime. *arXiv* **2019**, arXiv:1911.01544.
227. Chatterji, N.S.; Long, P.M. Finite-sample analysis of interpolating linear classifiers in the overparameterized regime. *arXiv* **2020**, arXiv:2004.12019.
228. Belkin, M.; Hsu, D.; Ma, S.; Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proc. Natl. Acad. Sci. USA* **2019**, *116*, 15849–15854. [CrossRef]
229. Geiger, M.; Spigler, S.; dAscoli, S.; Sagun, L.; Baity-Jesi, M.; Biroli, G.; Wyart, M. Jamming transition as a paradigm to understand the loss landscape of deep neural networks. *Phys. Rev. E* **2019**, *100*, 012115. [CrossRef]
230. Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; Vinyals, O. Understanding deep learning requires rethinking generalization. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
231. Lin, L.; Dobriban, E. What causes the test error? Going beyond bias-variance via ANOVA. *J. Mach. Learn. Res.* **2021**, *22*, 1–82.
232. Cherkassky, V.; Mulier, F. *Learning from Data: Concepts, Theory, and Methods*, 2nd ed.; Wiley: New York, NY, USA, 2007.
233. Vapnik, V.N. *The Nature of Statistical Learning Theory*, 2nd ed.; Springer: New York, NY, USA, 2000.
234. Du, K.-L. Several misconceptions and misuses of deep neural networks and deep learning. In Proceedings of the 2023 International Congress on Communications, Networking, and Information Systems (CNIS 2023), Guilin, China, 25–27 March 2023; CCIS 1893; Springer: Berlin/Heidelberg, Germany, 2023; pp. 155–171.
235. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed.; Springer: Berlin, Germany, 2005.
236. Ji, Z.; Telgarsky, M. The implicit bias of gradient descent on nonseparable data. In Proceedings of the 32nd Conference on Learning Theory (COLT), Phoenix, AZ, USA, 5–8 July 2019; Volume 99, pp. 1772–1798.
237. Soudry, D.; Hoffer, E.; Nacson, M.S.; Gunasekar, S.; Srebro, N. The implicit bias of gradient descent on separable data. *J. Mach. Learn. Res.* **2018**, *19*, 1–57.
238. Belkin, M.; Hsu, D.; Xu, J. Two models of double descent for weak features. *SIAM J. Math. Data Sci.* **2020**, *2*, 1167–1180. [CrossRef]
239. Hastie, T.; Montanari, A.; Rosset, S.; Tibshirani, R.J. Surprises in high-dimensional ridgeless least squares interpolation. *Ann. Stat.* **2022**, *50*, 949–986. [CrossRef] [PubMed]
240. Mei, S.; Montanari, A. The generalization error of random features regression: Precise asymptotics and double descent curve. *arXiv* **2019**, arXiv:1908.05355. [CrossRef]
241. Adlam, B.; Pennington, J. Understanding double descent requires A fine-grained bias-variance decomposition. In *Advances in Neural Information Processing Systems*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H., Eds.; Curran Associates, Inc.: New York, NY, USA, 2020; Volume 33, pp. 11022–11032.
242. d'Ascoli, S.; Refinetti, M.; Biroli, G.; Krzakala, F. Double trouble in double descent: Bias and variance(s) in the lazy regime. In Proceedings of the 37th International Conference on Machine Learning (ICML), Virtually, 13–18 July 2020; Volume 119, pp. 2280–2290.
243. Yang, Z.; Yu, Y.; You, C.; Steinhardt, J.; Ma, Y. Rethinking bias–variance trade-off for generalization of neural networks. In Proceedings of the International Conference on Machine Learning, Virtually, 13–18 July 2020; pp. 10767–10777.
244. Ibragimov, I.A.; HasMinskii, R.Z. *Statistical Estimation: Asymptotic Theory*; Springer: Berlin/Heidelberg, Germany, 2013; Volume 16.
245. Zou, D.; Wu, J.; Braverman, V.; Gu, Q.; Kakade, S.M. Benign overfitting of constant-stepsize SGD for linear regression. *J. Mach. Learn. Res.* **2023**, *24*, 1–58.

246. Bartlett, P.L.; Foster, D.J.; Telgarsky, M.J. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: New York, NY, USA, 2017; Volume 30, pp. 6240–6249.
247. Bengio, Y. Learning deep architectures for AI. *FNT Mach. Learn.* **2009**, *2*, 1–127. [CrossRef]
248. Dinh, L.; Pascanu, R.; Bengio, S.; Bengio, Y. Sharp minima can generalize for deep nets. In Proceedings of the 34th International Conference on Machine Learning (IMCL), Sydney, Australia, 6–11 August 2017; Volume 70, pp. 1019–1028.
249. Oneto, L.; Ridella, S.; Anguita, D. Do we really need a new theory to understand over-parameterization? *Neurocomputing* **2023**, *543*, 126227. [CrossRef]
250. Chuang, C.-Y.; Mroueh, Y.; Greenewald, K.; Torralba, A.; Jegelka, S. Measuring generalization with optimal transport. In *Advances in Neural Information Processing Systems*; Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P.S., Vaughan, J.W., Eds.; Curran Associates, Inc.: New York, NY, USA, 2021; Volume 34, pp. 8294–8306.
251. Neyshabur, B.; Bhojanapalli, S.; Mcallester, D.; Srebro, N. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: New York, NY, USA, 2017; Volume 30.
252. Neyshabur, B.; Li, Z.; Bhojanapalli, S.; LeCun, Y.; Srebro, N. The role of over-parametrization in generalization of neural networks. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
253. Neyshabur, B.; Tomioka, R.; Srebro, N. Norm-based capacity control in neural networks. In Proceedings of the 28th Conference on Learning Theory, PMLR, Paris, France, 3–6 July 2015; Grnwald, P., Hazan, E., Kale, S., Eds.; Volume 40, pp. 1376–1401.
254. Cherkassky, V.; Lee, E.H. To understand double descent, we need to understand VC theory. *Neural Netw.* **2024**, *169*, 242–256. [CrossRef]
255. Shawe-Taylor, J.; Bartlett, P.L.; Williamson, R.C.; Anthony, M. Structural risk minimization over data-dependent hierarchies. *IEEE Trans. Inf. Theory* **1998**, *44*, 1926–1940. [CrossRef]
256. Lee, E.H. and Cherkassky, V. Understanding Double Descent Using VC-Theoretical Framework. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 18838–18847. [CrossRef]
257. Siegelmann, H.T.; Sontag, E.D. On the computational power of neural nets. In Proceedings of the Conference on Computational Learning Theory (COLT), Pittsburgh, PA, USA, 24–26 July 1992; pp. 440–449.
258. Li, B.; Fong, R.S.; Tino, P. Simple cycle reservoirs are universal. *J. Mach. Learn. Res.* **2024**, *25*, 1–28.
259. Paassen, B.; Schulz, A.; Stewart, T.C.; Hammer, B. Reservoir memory machines as neural computers. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 2575–2585. [CrossRef] [PubMed]
260. Traversa, F.L.; Ramella, C.; Bonani, F.; Di Ventra, M. Memcomputing NP-complete problems in polynomial time using polynomial resources and collective states. *Sci. Adv.* **2015**, *1*, e1500031. [CrossRef] [PubMed]
261. Pei, Y.R.; Traversa, F.L.; Di Ventra, M. On the universality of memcomputing machines. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 1610–1620. [CrossRef]
262. Cover, T.M. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Trans. Electron. Comput.* **1965**, *14*, 326–334. [CrossRef]
263. Hassoun, M.H. *Fundamentals of Artificial Neural Networks*; MIT Press: Cambridge, MA, USA, 1995.
264. Denker, J.S.; Schwartz, D.; Wittner, B.; Solla, S.A.; Howard, R.; Jackel, L.; Hopfield, J. Large automatic learning, rule extraction, and generalization. *Complex Syst.* **1987**, *1*, 877–922.
265. Muller, B.; Reinhardt, J.; Strickland, M. *Neural Networks: An Introduction*, 2nd ed.; Springer: Berlin, Germany, 1995.
266. Friedrichs, F.; Schmitt, M. On the power of Boolean computations in generalized RBF neural networks. *Neurocomputing* **2005**, *63*, 483–498. [CrossRef]
267. Kolmogorov, A.N. On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. *Dokl. Akad. Nauk USSR* **1957**, *114*, 953–956.
268. Ismayilova, A.; Ismailov, V.E. On the Kolmogorov neural networks. *Neural Netw.* **2024**, *176*, 106333. [CrossRef]
269. Schmidt-Hieber, J. The Kolmogorov–Arnold representation theorem revisited. *Neural Netw.* **2021**, *137*, 119–126. [CrossRef]
270. Hecht-Nielsen, R. Kolmogorov's mapping neural network existence theorem. In Proceedings of the 1st IEEE International Conference on Neural Networks, San Diego, CA, USA, 21–24 June 1987; Volume 3, pp. 11–14.
271. Shen, Z.; Yang, H.; Zhang, S. Deep network approximation characterized by number of neurons. *Commun. Comput. Phys.* **2020**, *28*, 1768–1811. [CrossRef]
272. Royden, H.L. *Real Analysis*, 2nd ed.; Macmillan: New York, NY, USA, 1968.
273. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Contr. Signals Syst.* **1989**, *2*, 303–314. [CrossRef]
274. Funahashi, K.-I. On the approximate realization of continuous mappings by neural networks. *Neural Netw.* **1989**, *2*, 183–192. [CrossRef]
275. Chen, T.; Chen, H.; Liu, R.-W. A constructive proof and an extension of cybenko's approximation theorem. In *Computing Science and Statistics*; Page, C., LePage, R., Eds.; Springer: New York, NY, USA, 1992; pp. 163–168.

276. Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Netw.* **1991**, *4*, 251–257. [CrossRef]
277. Leshno, M.; Lin, V.Y.; Pinkus, A.; Schocken, S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Netw.* **1993**, *6*, 861–867. [CrossRef]
278. Li, L.K. Approximation theory and recurrent networks. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Baltimore, MD, USA, 7–11 June 1992; pp. 266–271.
279. Li, X.; Yu, W. Dynamic system identification via recurrent multilayer perceptrons. *Inf. Sci.* **2002**, *147*, 45–63. [CrossRef]
280. Gonon, L.; Ortega, J.-P. Fading memory echo state networks are universal. *Neural Netw.* **2021**, *138*, 10–13. [CrossRef]
281. Arena, P.; Fortuna, L.; Muscato, G.; Xibilia, M. Multilayer perceptrons to approximate quaternion valued functions. *Neural Netw.* **1997**, *10*, 335–342. [CrossRef]
282. Buchholz, S.; Sommer, G. A hyperbolic multilayer perceptron. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Como, Italy, 24–27 July 2000; IEEE: Piscataway, NJ, USA, 2000; Volume 2, pp. 129–133.
283. Buchholz, S.; Sommer, G. Clifford algebra multilayer perceptrons. In *Geometric Computing with Clifford Algebras: Theoretical Foundations and Applications in Computer Vision and Robotics*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 315–334.
284. Carniello, R.A.F.; Vital, W.L.; Valle, M.E. Universal approximation theorem for tessarine-valued neural networks. In Proceedings of the Encontro Nacional de Inteligência Artificial e Computacional (ENIAC), Sao Carlos, Sao Paulo, Brazil, 25–29 2021; pp. 233–243.
285. Voigtlaender, F. The universal approximation theorem for complex-valued neural networks. *Appl. Comput. Harmon. Anal.* **2023**, *64*, 33–61. [CrossRef]
286. Valle, M.E.; Vital, W.L.; Vieira, G. Universal approximation theorem for vector-and hypercomplex-valued neural networks. *Neural Netw.* **2024**, *180*, 106632 [CrossRef]
287. Valle, M.E. Understanding vector-valued neural networks and their relationship with real and hypercomplex-valued neural networks: Incorporating intercorrelation between features into neural networks [Hypercomplex signal and image processing]. *IEEE Signal Process. Mag.* **2024**, *41*, 49–58. [CrossRef]
288. Manita, O.A.; Peletier, M.A.; Portegies, J.W.; Sanders, J.; Senen-Cerda, A. Universal approximation in dropout neural networks. *J. Mach. Learn. Res.* **2022**, *23*, 1–46.
289. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366. [CrossRef]
290. Klassen, M.; Pao, Y.; Chen, V. Characteristics of the functional link net: A higher order delta rule net. In Proceedings of the IEEE International Conference on Neural Networks (ICNN), San Diego, CA, USA, 18–22 July 1988; pp. 507–513.
291. Chen, C.L.P.; Liu, Z.L. Broad learning system: An effective and efficient incremental learning system without the need for deep architecture. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 10–24. [CrossRef] [PubMed]
292. Chen, C.L.P.; Liu, Z.; Feng, S. Universal approximation capability of broad learning system and its structural variations. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 1191–1204 [CrossRef]
293. Kratsios, A.; Papon, L. Universal approximation theorems for differentiable geometric deep learning. *J. Mach. Learn. Res.* **2022**, *23*, 1–73.
294. Ganea, O.; Becigneul, G.; Hofmann, T. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: New York, NY, USA, 2018; Volume 31, pp. 5345–5355.
295. Krishnan, R.G.; Shalit, U.; Sontag, D. Deep Kalman filters. In Proceedings of the NeurIPS—Advances in Approximate Bayesian Inference, Montreal, QC, Canada, 11 December 2015.
296. Zaheer, M.; Kottur, S.; Ravanbakhsh, S.; Poczos, B.; Salakhutdinov, R.R.; Smola, A.J. Deep sets. In *Advances in Neural Information Processing Systems (NeurIPS)*; Curran Associates, Inc.: New York, NY, USA, 2017; Volume 30, pp. 3391–3401.
297. Wagstaff, E.; Fuchs, F.B.; Engelcke, M.; Osborne, M.A.; Posner, I. Universal approximation of functions on sets. *J. Mach. Learn. Res.* **2022**, *23*, 1–56
298. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*; Curran Associates, Inc.: New York, NY, USA, 2017; Volume 30, pp. 5998–6008.
299. Lee, J.; Lee, Y.; Kim, J.; Kosiorek, A.R.; Choi, S.; Teh, Y.W. Set Transformer: A framework for attention-based permutation-invariant neural networks. In Proceedings of the International Conference on Machine Learning (ICML), Long Beach, CA, USA, 9–15 June 2019; Volume 97, pp. 3744–3753.
300. Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. In Proceedings of the International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
301. Murphy, R.L.; Srinivasan, B.; Rao, V.; Ribeiro, B. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
302. Baldi, P.; Vershynin, R. The capacity of feedforward neural networks. *Neural Netw.* **2019**, *116*, 288–311. [CrossRef]

303. Baldi, P.; Vershynin, R. On neuronal capacity. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2018; pp. 7740–7749.
304. Bartlett, P.L.; Harvey, N.; Liaw, C.; Mehrabian, A. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *J. Mach. Learn. Res.* **2019**, *20*, 1–17.
305. Graves, A.; Wayne, G.; Danihelka, I. Neural Turing machines. *arXiv* **2014**, arXiv:1410.5401.
306. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2020; pp. 1877–1901.
307. Perez, J.; Barcelo, P.; Marinkovic, J. Attention is Turing complete. *J. Mach. Learn. Res.* **2021**, *22*, 1–35.
308. Dehghani, M.; Gouws, S.; Vinyals, O.; Uszkoreit, J.; Kaiser, L. Universal transformers. *arXiv* **2018**, arXiv:1807.03819.
309. Kleene, S.C. Representation of events in nerve nets and finite automata. In *Automata Studies*; Shannon, C., McCarthy, J., Eds.; Princeton University Press: Princeton, NJ, USA, 1956; pp. 3–42.
310. Hahn, M. Theoretical limitations of self-attention in neural sequence models. *arXiv* **2019**, arXiv:1906.06755. [CrossRef]
311. Yun, C.; Bhojanapalli, S.; Rawat, A.S.; Reddi, S.J.; Kumar, S. Are transformers universal approximators of sequence-to-sequence functions? In Proceedings of the International Conference on Learning Representations (ICLR), Virtually, 26–30 April 2020.
312. Chen, Y.; Gilroy, S.; Maletti, A.; May, J.; Knight, K. Recurrent neural networks as weighted language recognizers. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT), New Orleans, LA, USA, 1–6 June 2018; pp. 2261–2271.
313. Maass, W. On the computational power of winner-take-all. *Neural Comput.* **2000**, *12*, 2519–2535. [CrossRef] [PubMed]
314. Vapnik, V.N.; Chervonenkis, A.J. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Its Appl.* **1971**, *16*, 264–280. [CrossRef]
315. Valiant, P. A theory of the learnable. *Commun. ACM* **1984**, *27*, 1134–1142. [CrossRef]
316. Vapnik, V.N. *The Nature of Statistical Learning Theory*; Springer: New York, NY, USA, 1995.
317. Schapire, R.E. The strength of weak learnability. *Mach. Learn.* **1990**, *5*, 197–227. [CrossRef]
318. Bartlett, P.L.; Long, P.M.; Williamson, R.C. Fat-shattering and the learnability of real-valued functions. In Proceedings of the 7th Annual Conference on Computational Learning Theory, New Brunswick, NJ, USA, 10–12 July 1994; pp. 299–310.
319. Mendelson, S. Rademacher averages and phase transitions in Glivenko-Cantelli classes. *IEEE Trans. Inf. Theory* **2002**, *48*, 251–263. [CrossRef]
320. Hanneke, S.; Yang, L. Minimax analysis of active learning. *J. Mach. Learn. Res.* **2015**, *16*, 3487–3602.
321. Cherkassky, V.; Ma, Y. Another look at statistical learning theory and regularization. *Neural Netw.* **2009**, *22*, 958–969. [CrossRef]
322. Wolpert, D.H.; Macready, W.G. *No Free Lunch Theorems for Search*; SFI-TR-95-02-010; Santa Fe Institute: Santa Fe, NM, USA, 1995.
323. Cataltepe, Z.; Abu-Mostafa, Y.S.; Magdon-Ismail, M. No free lunch for early stropping. *Neural Comput.* **1999**, *11*, 995–1009. [CrossRef]
324. Magdon-Ismail, M. No free lunch for noise prediction. *Neural Comput.* **2000**, *12*, 547–564. [CrossRef] [PubMed]
325. Zhu, H. No free lunch for cross validation. *Neural Comput.* **1996**, *8*, 1421–1426. [CrossRef]
326. Goutte, C. Note on free lunches and cross-validation. *Neural Comput.* **1997**, *9*, 1245–1249. [CrossRef]
327. Rivals, I.; Personnaz, L. On cross-validation for model selection. *Neural Comput.* **1999**, *11*, 863–870. [CrossRef] [PubMed]
328. Haussler, D. Probably approximately correct learning. In Proceedings of the 8th National Conference on Artificial Intelligence, Boston, MA, USA, 29 July–3 August 1990; Volume 2, pp. 1101–1108.
329. McAllester, D. Some PAC-Bayesian theorems. In Proceedings of the Annual Conference on Computational Learning Theory (COLT), Tuscaloosa, AL, USA, 23–26 July 1998; ACM: New York, NY, USA, 1998; pp. 230–234.
330. Shawe-Taylor, J.; Williamson, R. A PAC analysis of a bayesian estimator. In Proceedings of the Annual Conference on Computational Learning Theory (COLT), Nashville, TN, USA, 29 July–1 August 1997; ACM: New York, NY, USA, 1997; pp. 2–9.
331. Viallard, P.; Germain, P.; Habrard, A.; Morvant, E. A general framework for the practical disintegration of PAC-Bayesian bounds. *Mach. Learn.* **2024**, *113*, 519–604. [CrossRef]
332. Blanchard, G.; Fleuret, F. Occam's hammer. In Proceedings of the Annual Conference on Learning Theory (COLT), Angeles, CA, USA, 25–28 June 2007; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4539, pp. 112–126.
333. Catoni, O. PAC-Bayesian supervised classification: The thermodynamics of statistical learning. *arXiv* **2007**, arXiv:0712.0248.
334. Anthony, M.; Biggs, N. *Computational Learning Theory*; Cambridge University Press: Cambridge, UK, 1992.
335. Shawe-Taylor, J. Sample sizes for sigmoidal neural networks. In Proceedings of the 8th Annual Conference on Computational Learning Theory, Santa Cruz, CA, USA, 17–19 July 1995; pp. 258–264.
336. Baum, E.B.; Haussler, D. What size net gives valid generalization? *Neural Comput.* **1989**, *1*, 151–160. [CrossRef]
337. Haykin, S. *Neural Networks: A Comprehensive Foundation*, 2nd ed.; Prentice Hall: Upper Saddle River, NJ, USA, 1999.
338. Gribonval, R.; Jenatton, R.; Bach, F.; Kleinsteuber, M.; Seibert, M. Sample complexity of dictionary learning and other matrix factorizations. *IEEE Trans. Inf. Theory* **2015**, *61*, 3469–3486. [CrossRef]

339. Simon, H.U. An almost optimal PAC algorithm. In Proceedings of the 28th Conference on Learning Theory (COLT), Paris, France, 6–9 July 2015; pp. 1–12.
340. Bartlett, P.L.; Maass, W. Vapnik-Chervonenkis dimension of neural nets. In *The Handbook of Brain Theory and Neural Networks*, 2nd ed.; Arbib, M.A., Ed.; MIT Press: Cambridge, MA, USA, 2003; pp. 1188–1192.
341. Koiran, P.; Sontag, E.D. Neural networks with quadratic VC dimension. In *Advances in Neural Information Processing Systems*; Touretzky, D.S., Mozer, M.C., Hasselmo, M.E., Eds.; MIT Press: Cambridge, MA, USA, 1996; Volume 8, pp. 197–203.
342. Schmitt, M. On the capabilities of higher-order neurons: A radial basis function approach. *Neural Comput.* **2005**, *17*, 715–729. [CrossRef]
343. Bartlett, P.L. Lower bounds on the Vapnik-Chervonenkis dimension of multi-layer threshold networks. In Proceedings of the 6th Annual ACM Conference on Computational Learning Theory (COLT), Louisville, KY, USA, 10–13 July 1993; ACM Press: New York, NY, USA, 1993; pp. 144–150.
344. Vapnik, V.; Levin, E.; Le Cun, Y. Measuring the VC-dimension of a learning machine. *Neural Comput.* **1994**, *6*, 851–876. [CrossRef]
345. Shao, X.; Cherkassky, V.; Li, W. Measuring the VC-dimension using optimized experimental design. *Neural Comput.* **2000**, *12*, 1969–1986. [CrossRef] [PubMed]
346. Mpoudeu, M.; Clarke, B. Model selection via the VC dimension. *J. Mach. Learn. Res.* **2019**, *20*, 1–26.
347. Anthony, M.; Bartlett, P.L. *Neural Network Learning: Theoretical Foundations*; Cambridge University Press: Cambridge, CA, USA, 1999; Volume 9.
348. Elesedy, B.; Zaidi, S. Provably strict generalisation benefit for equivariant models. In Proceedings of the International Conference on Machine Learning (ICML), Virtually, 18–24 July 2021; pp. 2959–2969.
349. Cohen, T.; Welling, M. Group equivariant convolutional networks. In Proceedings of the 33rd International Conference on Machine Learning (ICML), New York, NY, USA, 19–24 June 2016; Volume 48, pp. 2990–2999.
350. Bietti, A.; Venturi, L.; Bruna, J. On the sample complexity of learning under geometric stability. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2021; Volume 34, pp. 18673–18684.
351. Sannai, A.; Imaizumi, M.; Kawano, M. Improved generalization bounds of group invariant/equivariant deep networks via quotient feature spaces. In *Uncertainty Inartificial Intelligence*; PMLR; Elsevier: Amsterdam, The Netherlands, 2021; pp. 771–780.
352. Shao, H.; Montasser, O.; Blum, A. A theory of pac learnability under transformation invariances. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 13989–14001.
353. Philipp Christian Petersen, Anna Sepliarskaia VC dimensions of group convolutional neural networks. *Neural Netw.* **2024**, *169*, 462–474. [CrossRef]
354. Goldman, S.; Kearns, M. On the complexity of teaching. *J. Comput. Syst. Sci.* **1995**, *50*, 20–31. [CrossRef]
355. Shinohara, A.; Miyano, S. Teachability in computational learning. *New Gener. Comput.* **1991**, *8*, 337–348. [CrossRef]
356. Liu, J.; Zhu, X. The teaching dimension of linear learners. *J. Mach. Learn. Res.* **2016**, *17*, 1–25.
357. Simon, H.U.; Telle, J.A. MAP- and MLE-based teaching. *J. Mach. Learn. Res.* **2024**, *25*, 1–34.
358. Koltchinskii, V. Rademacher penalties and structural risk minimization. *IEEE Trans. Inf. Theory* **2001**, *47*, 1902–1914. [CrossRef]
359. Bartlett, P.L.; Mendelson, S. Rademacher and Gaussian complexities: Risk bounds and structural results. *J. Mach. Learn. Res.* **2002**, *3*, 463–482.
360. Anguita, D.; Ghio, A.; Oneto, L.; Ridella, S. A deep connection between the Vapnik-Chervonenkis entropy and the Rademacher complexity. *IEEE Trans. Neural Netw. Learn. Syst.* **2014**, *25*, 2202–2211. [CrossRef] [PubMed]
361. Bartlett, P.L.; Bousquet, O.; Mendelson, S. Local Rademacher complexities. *Ann. Stat.* **2005**, *33*, 1497–1537. [CrossRef]
362. Dudley, R. The sizes of compact subsets of Hilbert space and continuity of Gaussian processes. *J. Funct. Anal.* **1967**, *1*, 290–330. [CrossRef]
363. Mendelson, S. A few notes on statistical learning theory. In *Advanced Lectures on Machine Learning*; Mendelson, S., Smola, A., Eds.; LNCS Volume 2600; Springer: Berlin, Germany, 2003; pp. 1–40.
364. Yu, H.-F.; Jain, P.; Dhillon, I.S. Large-scale multi-label learning with missing labels. In Proceedings of the 21st International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 1–9.
365. Lei, Y.; Ding, L.; Zhang, W. Generalization performance of radial basis function networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 551–564.
366. Vapnik, V.N. *Estimation of Dependences Based on Empirical Data*; Springer: New York, NY, USA, 1982.
367. Oneto, L.; Anguita, D.; Ridella, S. A local Vapnik-Chervonenkis complexity. *Neural Netw.* **2016**, *82*, 62–75. [CrossRef]
368. Cherkassky, V.; Ma, Y. Comparison of model selection for regression. *Neural Comput.* **2003**, *15*, 1691–1714. [CrossRef]
369. Grunwald, P.D.; Mehta, N.A. Fast rates for general unbounded loss functions: From ERM to generalized Bayes. *J. Mach. Learn. Res.* **2020**, *21*, 1–80.
370. Vovk, V.; Papadopoulos, H.; Gammerman, A. (Eds.) *Measures of complexity: Festschrift for Alexey Chervonenkis*; Springer International Publishing: Cham, Switzerland, 2015.

371. V'yugin, V.V. VC dimension, fat-shattering dimension, rademacher averages, and their applications. In *Measures of Complexity*; Vovk, V., Papadopoulos, H., Gammerman, A., Eds.; Springer: Cham, Switzerland, 2015; pp. 57–74.
372. Koltchinskii, V.; Panchenko, D. Empirical margin distributions and bounding the generalization error of combined classifiers. *Ann. Stat.* **2002**, *30*, 1–50. [CrossRef]
373. Shalev-Shwartz, S.; Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*; Cambridge University Press: New York, NY, USA, 2014.
374. Du, K.-L.; Leung, C.-S.; Mow, W.H.; Swamy, M.N.S. Perceptron: Learning, generalization, model Selection, fault tolerance, and role in the deep learning era. *Mathematics* **2022**, *10*, 4730. [CrossRef]
375. Kawaguchi, K.; Kaelbling, L.P.; Bengio, Y. Generalization in deep learning. *arXiv* **2017**, arXiv:1710.05468.
376. Balestriero, R.; Bottou, L.; LeCun, Y. The effects of regularization and data augmentation are class dependent. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: New York, NY, USA, 2022; Volume 35, pp. 37878–37891.
377. Ratner, A.J.; Ehrenberg, H.; Hussain, Z.; Dunnmon, J.; Re, C. Learning to compose domain-specific transformations for data augmentation. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Redhook, NY, USA, 2017; Volume 30.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.