POZNAN UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTING AND TELECOMMUNICATION

DOCTORAL THESIS

BEYOND TRADITIONAL CURRICULUM LEARNING: SHAPING THE OPTIMIZATION PATH TO FIND BETTER MINIMA, FASTER

Izabela Krysińska

Supervisor
dr. hab. inż. Mikołaj Morzy, prof. PP

POZNAŃ 2025

Abstract

Training of modern deep neural networks traditionally relies on presenting data in a random, unstructured order. This thesis challenges that paradigm by systematically exploring curriculum learning (CL), a training methodology inspired by the human learning principle of starting with easy concepts before moving to more complex ones. Despite its intuitive appeal and early promise, CL has failed to become a standard tool in the machine learning practitioner's toolkit. This thesis addresses the central paradox of CL: the wide gap between its theoretical potential and its inconsistent, often disappointing, empirical results in practice. The core motivation of this work is to move beyond the collection of ad-hoc methods that characterize much of the CL literature. It aims to establish a more principled, analytical foundation for the field by systematically investigating not just *if* a curriculum works, but *why* and *how* it influences the fundamental dynamics of neural network training.

This work introduces a formal framework for curriculum learning and systematically investigates its core components — scoring function that defines data or task difficulty and pacing function that schedules the curriculum — through a series of novel methods and comprehensive empirical studies. We explore the scoring function across two primary dimensions: in the data space and in the task space. First, we propose a typicality-based static scoring, hypothesizing that prioritizing samples that are good representatives of their classes will help the model reach better minima and improve robustness. Second, we introduce VoG-Guided Learning (VGL), a data-driven curriculum that employs an adaptive scoring function based on the Variance of Gradients (VoG) to sample examples based on the model's uncertainty in real-time. Third we propose Transitional Objective Learning (TOL), a task-space curriculum that leverages a static, domain-specific hierarchy of objectives in speech recognition. We then conduct an in-depth investigation of the pacing function through the lens of batch composition, treating dynamic batch size scheduling as a primary curriculum mechanism and analyzing its effect in combination with different scoring functions.

Across a diverse range of tasks, from tabular data analysis, image classification, text classification, to speech recognition, our findings demonstrate that principled curriculum design significantly improves training outcomes. VGL consistently outperformed baselines by dynamically focusing on the most informative samples. TOL was shown to accelerate convergence by leveraging the natural hierarchy of phonemes. Additionally, our study of batch orchestration reveals that dynamic, non-monotonic pacing functions can act as a powerful implicit regularizer, effectively mitigating the generalization gap in large-batch training and outperforming simple warm-starting strategies. In summary, this thesis provides a systematic perspective on curriculum learning, moving it from a collection of disparate heuristics towards a principled discipline. By formally analyzing and empirically validating the roles of scoring and pacing, this work demonstrates that the deliberate orchestration of the training process is a powerful tool for more efficient development of higher-performing deep learning models.

Streszczenie

Trenowanie nowoczesnych, głębokich sieci neuronowych tradycyjnie opiera się na prezentowaniu danych w losowej kolejności. Niniejsza rozprawa doktorska podważa ten paradygmat, systematycznie badając uczenie programowe (curriculum learning, CL) — metodologię trenowania inspirowaną procesem uczenia u ludzi, polegającą na rozpoczynaniu od koncepcji łatwych przed przejściem do bardziej złożonych. Pomimo swojej intuicyjnej atrakcyjności i obiecujących wyników początkowych, CL nie stało się standardowym narzędziem w arsenale praktyków uczenia maszynowego. Rozprawa ta traktuje o głównym paradoksie związanym z CL: o niezgodności pomiędzy jego teoretycznym potencjałem a często rozczarowującymi, wynikami empirycznymi. Główną motywacją tej pracy jest wykroczenie poza zbiór niesystematycznych metod charakteryzujący znaczną część literatury na temat CL. Celem jest ustanowienie bardziej pryncypialnych, analitycznych podstaw dla tej dziedziny poprzez zbadanie nie tylko *czy* CL działa, ale przede wszystkim *dlaczego* i *jak* wpływa ono na fundamentalną dynamikę optymalizacji uczenia sieci neuronowych.

W pracy tej wprowadzono formalne ramy dla uczenia programowego i systematycznie zbadano jego kluczowe komponenty — funkcję określającą trudność (scoring function), która definiuje złożoność danych bądź zadań, oraz funkcję sterującą tempem (pacing function), która kontroluje czego w danym momencie model się uczy — poprzez serię nowatorskich metod i kompleksowych badań empirycznych. Funkcje określające trudność badamy w dwóch głównych wymiarach: w przestrzeni danych i w przestrzeni zadań. Po pierwsze, proponujemy statyczną ocenę trudności opartą na typowości próbek, zakładając, że priorytetyzacja próbek dobrze reprezentujących swoją klasę pomoże znaleźć lepsze minimum i wzmocni odporność na szum danych. Po drugie, wprowadzamy uczenie kierowane wariancją gradientów (VGL), program nauczania zdefiniowany w przestrzeni danych, który wykorzystuje adaptacyjną funkcję określającą trudność opartą na wariancji gradientów (VoG) do próbkowania przykładów na podstawie niepewności modelu w czasie rzeczywistym. Po trzecie, proponujemy uczenie z progresywną funkcją celu (TOL), program nauczania w przestrzeni zadań, który wykorzystuje statyczną, specyficzną dla domeny hierarchię celów w rozpoznawaniu mowy. Następnie przeprowadzamy dogłębne badanie funkcji sterującej tempem przez pryzmat kompozycji wsadów (batch composition), traktując dynamiczne harmonogramowanie rozmiaru wsadu jako główny mechanizm programu nauczania i analizując jego działanie w połączeniu z różnymi funkcjami określającymi trudność.

Nasze wyniki, uzyskane na zróżnicowanym zbiorze zadań, od analizy danych tabelarycznych, przez klasyfikację obrazów i tekstu, po rozpoznawanie mowy, pokazują, że analityczne projektowanie programu nauczania znacząco usprawnia proces trenowania. Modele trenowane z użyciem VGL konsekwentnie przewyższały dokładnością modele trenowane standardowymi metodami dzięki dynamicznemu skupianiu się na najbardziej informatywnych próbkach. Wykazano, że TOL przyspiesza zbieżność do minimum lokalnego, dzieląc złożony problem na składowe poprzez wykorzystanie naturalnej hierarchii fonemów. Dodatkowo, nasze badanie orkiestracji wsadów (batch orchestration) pokazuje, że dynamiczne, niemonotoniczne funkcje sterujące tempem nauczania

mogą działać jako czynnik regularyzacji, skutecznie poprawiając generalizacje modelu w trenowaniu z użyciem dużych wsadów i przewyższając proste strategie typu warm-starting. Podsumowując, niniejsza rozprawa dostarcza formalne ramy dla uczenia programowego, przekształcając je w kierunku zdefiniowej dyscypliny. Poprzez analizę i empiryczną walidację funkcji określających trudność i sterujących tempem, praca ta dowodzi, że świadoma orkiestracja procesu trenowania jest skutecznym narzędziem do wydajniejszego trenowania dokładniejszych głębokich sztucznych sieci neuronowych.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Aims . . . 1
- 1.3 Thesis structure . . . 2
- 1.4 Original contribution . . . 3

2 Curriculum Learning **6**
- 2.1 Introduction . . . 6
  - 2.1.1 Formal framework . . . 7
    - Stochastic Gradient Descent . . . 8
    - Data-driven CL . . . 9
    - Task-driven CL . . . 9
- 2.2 Core components . . . 10
  - 2.2.1 Scoring function . . . 10
    - Static scoring . . . 11
    - Dynamic scoring . . . 13
    - Summary . . . 17
  - 2.2.2 Pacing function . . . 19
    - Fixed pace . . . 19
    - Adaptive pace . . . 20
    - Summary . . . 21

3 Typicality scoring **22**
- 3.1 Introduction . . . 22
  - 3.1.1 Related work . . . 23
  - 3.1.2 Method . . . 24
- 3.2 Experiments . . . 25
- 3.3 Results . . . 26
- 3.4 Conclusions . . . 27

4 Variance of Gradients scoring **29**
- 4.1 Introduction . . . 30
  - 4.1.1 Related work . . . 30
  - 4.1.2 Method . . . 31
- 4.2 Experiments . . . 33
  - 4.2.1 Datasets . . . 33
  - 4.2.2 Setup . . . 36
- 4.3 Results . . . 38

4.3.1 Understanding VGL dynamics . . . 39
4.3.2 Performance analysis . . . 41
4.4 Conclusions . . . 42

5 Batch orchestration **46**
5.1 Introduction . . . 46
5.1.1 Related work . . . 48
5.1.2 Research gap . . . 50
5.2 Experiments . . . 51
5.2.1 Intra-epoch batch orchestration with typicality sampling . . . 51
5.2.2 Inter-epoch batch orchestration with VoG sampling . . . 54
5.3 Results . . . 58
5.3.1 Intra-epoch batch orchestration with typicality sampling . . . 58
5.3.2 Inter-epoch batch orchestration with VoG sampling . . . 60
Small-batch regime . . . 60
Large-batch regime . . . 61
Adaptive learning rate . . . 65
5.4 Conclusions . . . 65

6 Transitional Objective Learning **67**
6.1 Introduction . . . 68
6.1.1 Phoneme production . . . 68
6.1.2 TOL framework . . . 70
6.1.3 Connectionist Temporal Classification . . . 71
6.1.4 Related work . . . 72
6.2 Experiments . . . 73
6.2.1 Datasets . . . 73
6.2.2 Setup . . . 74
6.3 Results . . . 75
6.3.1 Phoneme Error Rate . . . 75
6.3.2 Convergence behavior . . . 77
6.3.3 Optimization stability . . . 77
6.3.4 Ablation study . . . 78
6.4 Conclusions . . . 78

7 Conclusions **80**
7.1 Research synthesis . . . 80
7.2 Revisiting the aims . . . 81
7.3 Other research paths . . . 82
7.4 Broader implications and limitations . . . 83
7.5 Future research directions . . . 83
7.6 Closing . . . 84

A Transfer hyperparameter learning **85**
A.1 Motivation . . . 85
A.2 Method . . . 86
A.3 Experiments and results . . . 87
A.3.1 Experiment 1: Measuring convergence rate . . . 87

A.3.2 Experiment 2: Performance under a limited time budget 89
A.3.3 Experiment 3: Testing the transferability of hyperparameter rankings . . . 89
A.3.4 Experiment 4: Predicting future performance with early results 90
A.3.5 Experiment 5: Establishing a baseline for early stopping 90
A.4 Conclusions 91

Bibliography **92**

Chapter 1

Introduction

1.1 Motivation

In the pursuit of building more capable and efficient artificial intelligence, the machine learning community has often drawn inspiration from the way humans learn. While we have made tremendous progress in developing complex models like deep neural networks, we often overlook one of the most fundamental aspects of human education: the curriculum. Humans and animals rarely learn from a randomly ordered sequence of examples. Instead, we learn progressively, mastering simple concepts before moving on to more complex ones. A child learns to identify letters before reading words, and words before comprehending sentences. This structured, meaningful ordering of information is the foundation of efficient learning.

This same principle is the core idea behind curriculum learning (CL), a training paradigm that aims to improve a model's convergence and generalization by presenting training examples in a structured, non-random order — typically from easy to hard. The central hypothesis is that a carefully designed curriculum can guide a model through a complex, non-convex loss landscape towards a better local minimum than it might find if presented with the entire, unordered dataset from the outset.

The intellectual roots of this idea in machine learning trace back to Elman, 1993 on "starting small". Elman demonstrated that a simple recurrent neural network could learn complex grammatical structures only if it was first trained on a simpler subset of the language. When exposed to the full, complex dataset from the beginning, the network failed to learn. This insight suggested that the sequence of learning mattered profoundly; starting with a simplified version of the problem space effectively preconditioned the model, enabling it to subsequently grasp more intricate patterns. The term "curriculum learning" was later formally coined by Bengio et al., 2009, who showed its benefits in speeding up training and improving the final performance of deep models.

1.2 Aims

Despite its intuitive appeal and early promise over 15 years ago (Bengio et al., 2009), curriculum learning has failed to become a standard tool in the practitioner's toolkit. Why does this discrepancy exist? We hypothesize that the paradigm's limited adoption stems from several factors. Firstly, empirical studies on CL have yielded mixed results, showing that a given curriculum is not universally beneficial and can even degrade performance. This lack of reliability discourages adoption, as the process of curriculum design, finding the right data ordering and pacing, is a time-consuming process with no guarantee of success. Secondly, the landscape of machine learning has evolved. Modern architectures like transformers, often trained in few-shot or transfer learn-

ing settings, may reduce the perceived need for carefully orchestrated datasets where curriculum learning was first shown to be effective.

However, we argue that these challenges do not invalidate the core principle of curriculum learning. Rather, they suggest that its most common interpretation, a rigid progression from "easy" to "hard", is incomplete. The focus of this thesis is to challenge this dogma. We propose that the true potential of curriculum learning is unlocked when we view it not as a fixed recipe, but as a meta-learning paradigm that actively controls the training process by tailoring the data or task sequence to the model's evolving capabilities. From this perspective, the goal is not always to present the easiest examples first. On the contrary, an efficient training strategy might prioritize difficult examples if they provide more informative gradients for generalization, while avoiding redundant computation on easy examples.

The main goal of this thesis is to move beyond the inconsistent approach of describing and evaluating curricula that characterize curriculum learning literature and establish a more principled, analytical foundation for the field. We aim to understand *why* and *how* a given curriculum influences a model's training process, providing actionable insights for designing effective learning strategies. To achieve this, the research is guided by the following specific aims:

A1 To establish a formal framework for curriculum learning Our first aim is to bring structure to the diverse landscape of curriculum learning research. We propose a formal framework that deconstructs any curriculum into two core components: a scoring function (which assigns a value, like difficulty or typicality, to data points or tasks) and a pacing function (which dictates the rate and manner of introducing data or tasks over time). Using this framework, we categorize existing literature into two primary domains: data-driven curricula, which reorder examples within a single task, and task-driven curricula, which sequence a series of distinct but related tasks.

A2 To systematically analyze the impact of curriculum learning on learning dynamics Having defined the formal framework, our central aim is to conduct a thorough empirical investigation into how different curricula affect the fundamental mechanisms of neural network optimization. We move beyond simply measuring final accuracy and instead analyze a suite of underlying dynamics, including:

- gradient variance, magnitude, and norm,
- gradient noise scale,
- Lipschitz constant,
- and convergence dynamic.

By correlating the choice of curriculum with its effect on these learning dynamics, we want to uncover the causal mechanisms that determine a curriculum's success or failure.

The scope of this thesis is deliberately limited to the domain of deep neural networks operating within the supervised learning paradigm.

1.3 Thesis structure

This thesis is structured into seven chapters, each building upon the previous to develop a comprehensive and principled analysis of curriculum learning.

Chapter 1 introduces the thesis by presenting the core motivation for this research, defining the specific aims of the work, and providing a high-level summary of its original contributions.

Following this, **Chapter 2** lays the theoretical groundwork for the entire thesis. It provides a comprehensive review of the field, critically assesses existing work, and introduces our formal framework for deconstructing curricula into their core components: the scoring function and the pacing function. This chapter also establishes the distinction between data-driven and task-driven curricula that we use to structure our subsequent investigations.

Chapter 3 presents our first empirical study, focusing on a data-driven curriculum that prioritizes "typical" or representative examples. We introduce a graph-based method to calculate the typicality of a sample. The findings in this chapter highlight a fundamental flaw in naive curriculum implementations: the creation of homogenous batches that introduce destabilizing gradient noise late in training, ultimately harming model performance.

Chapter 4 explores a dynamic, model-aware alternative to the static scoring function from the previous chapter. We propose and evaluate a scoring function based on the Variance of Gradients (VoG). This chapter investigates how a metric tied directly to the learning process itself can be used to rank and understand the utility of training examples as the model learns.

Chapter 5 addresses the limitations identified in Chapter 3 by focusing on the pacing function. Moving beyond simple sequential ordering, we introduce the concept of batch orchestration, a method for dynamically constructing mini-batches to balance different types of examples based on their scores (from Chapters 3 and 4). This chapter demonstrates how pacing can change gradient noise and unlock the benefits of curriculum learning.

Chapter 6 expands our scope from data-driven to task-driven curricula. We present a novel method, Transitional Objective Learning (TOL), applied to the challenging domain of automatic speech recognition. This chapter shows how the core principles of curriculum learning, starting simple and increasing complexity, can be applied not only to data but to the learning objective itself, leading to more stable and effective training.

Finally, **Chapter 7** concludes the thesis by synthesizing the key findings, discussing the broader implications of this research, and outlining several promising directions for future work.

1.4 Original contribution

The primary contributions of this thesis are aimed at transforming curriculum learning from a collection of ad-hoc methods into a structured field of study and understanding how CL impacts training dynamics. Our contributions are as follows:

C1 A formal framework for curriculum learning The first major contribution of this thesis is the establishment of a formal, unified framework that deconstructs the design of any curriculum into its fundamental components. In Chapter 2, we formally defined the two core modules of a curriculum: the scoring function, which evaluates the difficulty or suitability of training examples or tasks, and the pacing function, which governs the introduction of data or tasks over the course of training. Furthermore, we defined two distinct domains in which a curriculum can operate: the input space, leading to data-driven curricula that prioritize certain examples in the course of training, and the task space, which involves sequencing or modifying the learning tasks themselves. To validate and demonstrate the utility of this framework, we provided a comprehensive overview of past research, systematically placing prior works within our proposed framework. This effort synthesized existing knowledge, provided a common vocabulary for the field, and enabled a more structured analysis and comparison of curriculum strategies.

C2 Redefinition of "difficulty": Our second contribution is the proposal of two novel scoring functions that challenge the traditional "easy-to-hard" curriculum narrative. In Chapter 3, we introduced a static, graph-based typicality score as a data-driven measure of sample representativeness. In Chapter 4 we proposed a dynamic scoring function based on the Variance of Gradients (VoG). We are the first to repurpose VoG from its original context in explainable AI (Agarwal et al., 2022) to the domain of curriculum learning. In doing so, we also successfully transferred its application from the image domain, where it was originally demonstrated, to the text domain. We empirically proved its effectiveness as an adaptive scoring function across diverse datasets and model architectures, including modern pretrained transformers. Through these investigations, we demonstrated that an effective curriculum is not only about ordering examples by a fixed notion of difficulty, but about dynamically selecting data that guides the optimization process. This work shows that a well-designed curriculum can serve as a training accelerator, improving convergence rate and guiding the model towards better local minima.

C3 A novel task-driven curriculum for hierarchical objectives Our third contribution extends the principles of curriculum learning from the data space to the objective space itself. In Chapter 6, we proposed a novel task-driven strategy called Transitional Objective Learning (TOL). This framework leverages the inherent hierarchy of subtasks within a complex problem to construct a progressive optimization schedule that guides the model from a coarse-grained to a fine-grained objective. We demonstrated the effectiveness of TOL on the challenging task of phoneme recognition, validating its performance on three distinct datasets across three languages: Polish, English, and French. The results show that TOL consistently improves the convergence rate and leads to significantly better final performance across all tested conditions, establishing it as a powerful method for applying curriculum principles to structured prediction tasks.

C4 An analysis of curriculum-induced learning dynamics Finally, we provided a thorough empirical investigation into how and why curricula alter the fundamental mechanisms of neural network training. In Chapter 5, we systematically studied various batch orchestration strategies as pacing functions, observing their complex interactions with training dynamics. This was guided by a set of research questions aimed at understanding the relationship between batch composition, the variance of stochastic gradients, and the model's generalization gap. A hallmark of our work is its consistent focus on the underlying dynamics of learning. In the majority of the experimental chapters, we moved beyond simply measuring final accuracy to analyze a suite of optimization metrics, including gradient variance, magnitude, and norm (Chapters 5, 6), gradient noise scale (Chapter 5), convergence dynamics (Chapters 4 and 5, 6, and Appendix A), and the evolution of model uncertainty (Chapter 4). This analysis provided a clearer picture of the mechanisms by which curricula influence optimization, offering valuable insights for designing more effective and reliable training strategies.

C5 Democratization of research assets and knowledge transfer Beyond the core scientific investigations, a significant contribution of this work is our dedicated effort to democratize access to our research and facilitate its usage in the real world, practical applications. We focused on transferring the scientific knowledge and assets generated during this thesis into practical, ready-to-use resources for the broader machine learning community. To this end, we have published a curated collection of five datasets and three models on the Hugging

Face platform[1]. These assets have already been downloaded several thousand times overall, demonstrating their direct utility and impact on machine learning community globally. It shows the value of our work extends beyond academic publications and contributes directly to the tools and resources used in real-world applications.

[1] https://huggingface.co

Chapter 2

Curriculum Learning

This chapter provides a comprehensive overview of the field of curriculum learning, from its theoretical foundations to its practical implementations. We begin in Section 2.1 by establishing a formal definition of the paradigm, contrasting it with standard mini-batch Stochastic Gradient Descent (mini-batch SGD). This section introduces a mathematical framework, demonstrating how curriculum learning can be realized by modifying either the data sampling (a data-driven curriculum) or the learning objective itself (a task-driven curriculum).

Following this formal introduction, Section 2.2 deconstructs the paradigm into its two essential, modular components: the scoring function and the pacing function. In Section 2.2.1, we conduct an in-depth analysis of the scoring function, the mechanism used to quantify the difficulty or complexity of data or tasks. We explore the critical distinction between static scoring functions, which are pre-computed using human knowledge or fixed heuristics, and the dynamic scoring functions, which adapt to the model's state during training by using signals such as loss, prediction confidence, entropy, and information gain. Subsequently, in Section 2.2.2, we examine the second core component, the pacing function or scheduler. This section details the different strategies for controlling the temporal progression of the curriculum, distinguishing between fixed pace schedules (e.g., linear, exponential) and responsive adaptive pace functions that create a feedback loop with the model's own learning progress.

2.1 Introduction

Curriculum learning represents a training paradigm in machine learning that draws inspiration from human pedagogical principles, where learning progresses from simple to complex concepts in a structured manner. The fundamental premise underlying curriculum learning is that the order in which training examples are presented to a learning algorithm can significantly impact both the efficiency and effectiveness of the learning process. Formally introduced by Bengio et al., 2009, curriculum learning challenges the conventional approach of random sampling from training datasets. Instead, it proposes a controlled presentation of training examples that follows a carefully designed curriculum. This approach mirrors human learning processes, where foundational concepts are mastered before advancing to more complex material.

Curriculum learning promises several key benefits. Firstly, it promises to lead to **improved convergence rates**. By initially focusing on easier or more informative examples, models can establish basic patterns more quickly, often leading to faster convergence than standard random-shuffle training (Hacohen & Weinshall, 2019; Weinshall & Amir, 2018). CL can improve efficiency in large-scale training. For instance, some methods use a curriculum for only the initial, most

unstable phase of training to accelerate stabilization before switching to standard training (Amodei et al., 2016), while other analyses have shown that training on a smaller set of examples can achieve performance comparable to training on a massive dataset, offering potential savings in computational resources (Toneva et al., 2018).

Secondly, CL promises to provide **robustness to noisy data**. In datasets with corrupted or ambiguous labels, starting with cleaner examples helps the model build a reliable foundation before it tackles more challenging, potentially misleading data (L. Jiang et al., 2018; Lyu & Tsang, 2020; Mindermann et al., 2022). Therefore, the model should be more resilient to noisy data.

Finally, CL promises to find better local minima and, in a result, **improve the performance** of the final model (Hong et al., 2024; Huang et al., 2020; Kim & Lee, 2024; Wang et al., 2021; Xu et al., 2020; Yang et al., 2025). Many machine learning models, particularly deep neural networks, can become trapped in poor local minima when faced with complex optimization landscapes. The theoretical foundation of curriculum learning rests on the hypothesis that by starting with an effective subset of examples, the model can learn more robust foundational representations. This initial training phase effectively smooths the optimization landscape. As training progresses, the data sampling is gradually shifted to include more complex examples, refining the model's capabilities. Bengio et al., 2009 compare this process to a continuation method (Allgower & Georg, 2012), an optimization technique for finding the global minimum of a non-convex function.

The intuitive appeal of curriculum learning has driven significant research interest across domains like computer vision, natural language processing, and reinforcement learning. However, its empirical effectiveness remains a subject of active debate. While many studies report improvements in convergence speed and final model performance, the benefits are not universal. Notable work by X. Wu et al., 2020 questions the advantages of explicit curricula, arguing that observed gains might often be attributed to other factors, such as implicit regularization from starting with a smaller, more effective dataset. Their findings suggest that the benefits of CL are most visible under specific conditions, such as a limited training budget or in the presence of noisy data. This highlights a central challenge in the field: understanding the precise interplay between dataset characteristics, model architecture, and curriculum design that leads to successful outcomes.

This chapter provides a comprehensive overview of curriculum learning. We begin by establishing a formal mathematical foundation for CL, contrasting it with a standard SGD-type training. As the SGD-type training, we understand the family of SGD-based optimizers: plain SGD, mini-batch SGD, and its adaptive versions such as Adam (Adam et al., 2014) and RMSProp (Hinton, 2012). We then deconstruct CL into its two core components: the scoring function, which quantifies the difficulty of training examples or tasks, and the pacing function, which schedules the introduction of complexity. We explore various instantiations of these components, from static, predefined curricula to dynamic, model-aware approaches, and discuss their applications across different machine learning tasks.

2.1.1 Formal framework

To operationalize the concept of curriculum learning and analyze its impact on the optimization process, we establish a formal mathematical framework. This section provides such a foundation. We begin by outlining the standard training paradigm of mini-batch SGD to establish the baseline. Against this, we then formalize how a curriculum modifies the learning objective through data or task over time, providing a precise definition of a curriculum-enhanced training process. Finally, we distinguish between the two primary ways a curriculum can be implemented: by manipulating the data sampling (in the input space) or by altering the learning task itself (in the task space).

Stochastic Gradient Descent

The optimization of most large-scale machine learning models is based on mini-batch Stochastic Gradient Descent. In mini-batch SGD model parameters are updated iteratively to minimize a loss function, ℓ, based on gradients computed from small, random subsets of the training data called mini-batches. The standard update rule is:

$$\theta_t = \theta_{t-1} - \eta \nabla_\theta \ell(\theta_{t-1}; B_t) \tag{2.1}$$

where θ_t represents the model parameters at iteration t; $\eta > 0$ is the learning rate, controlling the step size; $\ell(\theta_{t-1}; B_t)$ is the loss function computed on the mini-batch B_t sampled randomly without replacement from:

$$\{B_t \subset D \mid card(B_t) = b\};\ \{B_1, B_2, \ldots, B_{\lceil n/b \rceil}\} \overset{\text{iid}}{\sim} D \tag{2.2}$$

where:

- $D = \{(\mathbf{x_i}, \mathbf{y_i})\}_{i=1}^{n}; D \sim \mathcal{D}$[1], n is the size of the whole dataset and b is the batch size
- Subsequent batches are sampled once and remain the same across epochs. For every epoch the batch sequence $\{B_1, B_2, \ldots, B_{\lceil n/b \rceil}\}$ stays the same.

The gradient estimation of the loss function calculated on the examples from batch B with respect to the model's parameters is defined as

$$\nabla_\theta J = \nabla_\theta \ell(\theta; B) = \nabla_\theta \ell(f_\theta(\mathbf{x}), \mathbf{y}) \tag{2.3}$$

where f is the model's prediction function. Gradient estimation $\nabla_\theta \ell$ is calculated by averaging the gradient of all examples in the batch B.

This formulation assumes a static optimization process in which the loss function ℓ and the assignment of samples to subsequent batches remain constant throughout training. This approach can lead to several issues:

1. Poor local minima: The optimization may converge to suboptimal solutions when faced with complex loss landscapes
2. Slow convergence: The model wastes computational resources optimizing parameters on examples that are already learned or are potentially unlearnable due to misannotation or excessive noise in the data.

Curriculum learning modifies the standard gradient descent algorithm by introducing a curriculum strategy Υ that changes updates to the model through deliberate manipulation of samples or task importance during the training. The curriculum-enhanced gradient descent is formulated as:

$$\theta_t = \theta_{t-1} - \eta \nabla_\theta J_t^\Upsilon \tag{2.4}$$

where J^Υ is the curriculum-modified objective function.

Definition 1 *Curriculum learning strategy Υ can be defined with two components: scoring function S and pacing function P, forming a curriculum learning strategy $\Upsilon = (\mathcal{S}, \mathcal{P})$.*

The modification of mini-batch SGD with Υ, can be realized at two levels: by controlling the data the model processes, or by changing the complexity of the task the model is asked to perform. This leads to two primary implementations of the curriculum framework: curriculum defined in the input space and curriculum defined in the task space.

[1] For the sake of brevity and clarity, in the remainder of the text we will refer to a sample $(\mathbf{x_i}, \mathbf{y_i})$ as sample i

Data-driven CL

A data-driven curriculum modifies the samples importance throughout the training. It can be pursued by oversampling or undersampling the examples at different phases of training. Meaning that the curriculum learning strategy Υ defined in the input space modifies the i.i.d. batch sampling defined in Equation 2.2.

Let us consider components of a data-driven curriculum strategy. The first component, scoring function $\mathcal{S} : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}$, assigns a real number $s_i = \mathcal{S}(\mathbf{x_i}, \mathbf{y_i})$ to sample i. The score s_i represents the complexity or importance of sample i.

Before we define the second component, pacing function, let us define probabilistic batch sampling, which is a core mechanism of the pacing function. For each example $(\mathbf{x_i}, \mathbf{y_i}) \in D$, let p_i denote the probability that $(\mathbf{x_i}, \mathbf{y_i})$ is included in the sampled mini-batch where $p_i \in [0, 1]\ \forall i \in \{1, 2, \dots n\}$. A minibatch $B_t \subset D$ is sampled from D according to probability vector $\mathbf{p} = (p_1, p_2, \dots p_n)$. For each element $(\mathbf{x_i}, \mathbf{y_i})$ we have:

- $(\mathbf{x_i}, \mathbf{y_i}) \in B_t$ with probability p_i
- $(\mathbf{x_i}, \mathbf{y_i}) \notin B_t$ with probability $1 - p_i$

Let Z_i be a Bernoulli random variable for each $i \in \{1, 2, \dots n\}$ such that $Z_i \sim \text{Bernoulli}(p_i)$ where:

- $Z_i = 1$ if $(\mathbf{x_i}, \mathbf{y_i}) \in B_t$
- $Z_i = 0$ if $(\mathbf{x_i}, \mathbf{y_i}) \notin B_t$

Then the sampled mini-batch B_t can be expressed as:

$$B_t = \{(\mathbf{x_i}, \mathbf{y_i}) \in D \mid Z_i = 1\} \tag{2.5}$$

For any specific minibatch $A \subset D$ probability of being sampled is equal to $P(B_t = A) = \prod_{i:(\mathbf{x_i}, \mathbf{y_i}) \in A} p_i$.

If $\forall_i\ p_i = p$, then each sample has equal probability p of inclusion, which makes the batch sampling identical to the standard batch sampling defined in Equation 2.2. We can also define deterministic batch selection if $\forall_i\ p_i \in \{0, 1\}$. Deterministic batch selection allows for excluding samples from batch selection and sampling the rest of the samples uniformly.

Having defined probabilistic batch sampling, we can finally describe pacing function $\mathcal{P} : \{(\mathbb{R}^d \times \mathbb{R}^m)\} \times \mathbb{R} \times \mathbb{R} \to \{(\mathbb{R}^d \times \mathbb{R}^m)\}$, which returns a mini-batch $B_t = \mathcal{P}(D, \mathbf{s}, t)$. The pace function $\mathcal{P}$ defines the probability vector $\mathbf{p}$ based on the scoring vector $\mathbf{s}$ and the iteration of training t. The probability vector $\mathbf{p}$ can be static, remaining the same regardless of the phase of training t, or dynamic, changing depending on the training progress.

The curriculum learning strategy defined in the input space modifies the update function (Equation 2.1) in the following way:

$$J_t^{\Upsilon} = \ell(\theta_{t-1}; B_t^{\Upsilon}) \tag{2.6}$$

Task-driven CL

The goal for finding a good parameter vector θ^* is to minimize the average loss all training examples in D:

$$\theta^* = \underset{\theta}{\text{argmin}} \sum_{(\mathbf{x}, \mathbf{y}) \in D} \ell(f_\theta(\mathbf{x}), \mathbf{y}) \tag{2.7}$$

Let a complex learning problem be decomposable into a set of subtasks $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_j, \}$. Each subtask ℓ_i has an associated difficulty score s_i assigned by the scoring function $\mathcal{S}$, where $\mathbf{s^t} = \mathcal{S}(\mathcal{L}, \sqcup)$. A task-driven curriculum learning strategy defines a schedule for mastering these subtasks:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{(x,y)\in\mathcal{D}} \mathcal{L}(f_\theta(x), y) \tag{2.8}$$

The task-driven curriculum learning strategy loss function at training step t can be represented as a weighted sum over all tasks and examples:

$$J_t^{\Upsilon} = \ell^{\Upsilon}(\theta_{t-1}; B_t) = \left[\frac{1}{j}\sum_{i=1}^{j} w_i^t \ell_i(f_{\theta_{t-1}}(\mathbf{x}), \mathbf{y})\right] \tag{2.9}$$

where ℓ_i is the loss for the i-th subtask, and $w_i^t \in \mathbb{R}^+$, is the weight assigned to that subtask i at step t. The pacing function, $\mathcal{P}$ determines these weights based on the score vector $\mathbf{s}$ and the current progress of training: $\mathbf{w^t} = \mathcal{P}(\mathbf{s}, t)$.

In both types, data-driven curriculum and task-driven curriculum, CL transforms a single optimization problem into a meaningful sequence of more manageable ones. It can be viewed as solving a sequence of optimization problems that differ in input or task space:

$$\underset{\theta}{\operatorname{argmin}}\, J(\theta) : \underset{\theta}{\operatorname{argmin}}\, J_1(\theta) \rightarrow \underset{\theta}{\operatorname{argmin}}\, J_2(\theta) \rightarrow \ldots \rightarrow \underset{\theta}{\operatorname{argmin}}\, J_T(\theta), \tag{2.10}$$

where $J(\theta)$ represents the original problem.

2.2 Core components

Designing any curriculum learning strategy requires answering two fundamental questions: first, how should the difficulty of data or tasks be measured, and second, at what rate should this difficulty be changed during training? These questions give rise to the two core components of a curriculum learning strategy: the scoring function and the pacing function.

The scoring function is responsible for assigning a numerical rating to each training example or task to quantify its difficulty, complexity, uncertainty or importance. The pacing function, in turn, uses these scores to govern the schedule of the curriculum, controlling the transition from easier to more complex material over the course of training. The combination of these two components determines the curriculum. In the following sections, we examine different approaches for designing both scoring and pacing functions.

2.2.1 Scoring function

One of the biggest challenges when designing a curriculum is to establish a meaningful measure of "difficulty" for training examples or tasks. The scoring function is the component designed to solve this problem. The function's precise form depends on whether the curriculum is defined in the input or task space.

- As defined earlier, for a data-driven curriculum learning strategy, the function $\mathcal{S} : D \rightarrow \mathbb{R}$ assigns a score to each data point $(\mathbf{x}, \mathbf{y}) \in D$, where D is the training dataset.
- For a task-driven curriculum, the function $\mathcal{S} : \mathcal{L} \rightarrow \mathbb{R}$ assigns a score to each subtask $\ell \in \mathcal{L}$, where $\mathcal{L}$ represents the set of objectives with different complexities described by $\mathcal{S}$.

The score typically reflects some notion of difficulty, complexity, importance or in general suitability for presentation at a given stage of training. Broadly, approaches to designing a scoring function fall into two main categories: static and dynamic. Static scoring functions are pre-computed and remain fixed throughout training. Their scores are typically based on domain-specific heuristics that are assumed to reflect difficulty. For instance, in computer vision tasks, difficulty might relate to image clarity, object occlusion, or scene complexity; in natural language processing, it could concern syntactic complexity, semantic ambiguity, or discourse coherence. In contrast, dynamic scoring functions are adaptive, with scores that can evolve during training. The mechanism for calculating the score is often general and domain-agnostic, but the resulting scores are highly specific to the model's state at that moment. For example, a model's prediction confidence or training error for an item can be used to dynamically assess its difficulty. This approach allows the curriculum to be tailored in real-time to the learner's specific needs. The following subsections explore the most common scoring functions within both categories, detailing their methodologies and applications across various domains.

Static scoring

We begin by examining static scoring functions, which rely on pre-computed, fixed difficulty measures. Within this category, the most intuitive approach is to leverage direct human intelligence and perception to define the curriculum. This leads to human-derived scoring methods, where the notion of difficulty is grounded in expert knowledge or the collective judgment of annotators.

Human-derived scoring Human-derived scoring represents the most intuitive approach, where domain experts manually annotate training examples with difficulty scores or where difficulty is inferred directly from the annotation process. This method leverages human expertise to capture nuanced aspects of difficulty that may be challenging to quantify algorithmically. A clear illustration of this is found in computer vision, where Soviany et al., 2020 developed a curriculum for training Generative Adversarial Networks (GANs) based on perceptual image difficulty. In their work, the difficulty scores were derived from human cognitive load, specifically, the time it took for annotators to complete a visual search task for a given image (Tudor Ionescu et al., 2016). This directly ties the curriculum's structure to the nuances of human visual processing. Beyond direct annotation of difficulty, valuable scoring functions can also be extracted from the standard labeling process itself, particularly when multiple annotators are involved. The level of agreement among these annotators can serve as a powerful proxy for a sample's ambiguity and, by extension, its difficulty. This technique has been effectively applied in speech emotion recognition, where the variance in continuous ratings (e.g., for emotional valence) or the entropy of categorical labels among annotators is used as the difficulty score (Lotfian & Busso, 2019). A sample with high variance or entropy signifies low annotator agreement, marking it as more difficult for the curriculum. Similarly, in fracture classification from X-ray images, medical knowledge such as decision trees and inconsistencies in annotations from multiple experts have been used to assign difficulty degrees to training examples, with medical knowledge-based curricula achieving up to 15% improvement in accuracy compared to random training strategies (Jiménez-Sánchez et al., 2019). However, human annotation is often expensive, time-consuming, and subject to inter-annotator variability (Snow et al., 2008). Human-defined curricula can be effective in specific domains but may not generalize well, limiting their scalability and broader application (Platanios et al., 2019).

Heuristics As an alternative to the often costly and time-intensive methods of direct human annotation, heuristic-based scoring functions determine difficulty based on intrinsic, measurable properties of the data itself. These heuristics serve as proxies for complexity and allow for the automatic, pre-computed ranking of an entire dataset without manual intervention.

The most common and widely applied heuristic in NLP is text length and lexical complexity. Bengio et al., 2009 proposed a curriculum for language modeling that started with a small vocabulary and gradually introduced rarer, more complex words. This principle extends naturally to sentence length. For example, in NLP, curricula are often designed by starting with shorter sentences and progressing to longer ones to improve training stability and performance (Kocmi & Bojar, 2017; Xu et al., 2020; X. Zhang et al., 2018; X. Zhang et al., 2019; Y. Zhou et al., 2020). Similarly, Tay et al., 2019 designed a curriculum for long-form reading comprehension that gradually increased the length of the document context provided to the model. This concept is not limited to text; in software engineering, for instance, curricula for code completion and execution tasks have used the length of code snippets, along with other heuristics, as a proxy for difficulty (Nair et al., 2024). The same heuristic has proven particularly effective in speech processing. The Deep Speech 2 system introduced "SortaGrad", a curriculum approach that uses utterance length as a difficulty metric. Training begins with shorter utterances and gradually incorporates longer ones, a strategy found to be critical in early training epochs to avoid large gradients from long utterances that could destabilize the learning process (Amodei et al., 2016).

Moving beyond simple length, other curricula leverage more direct measures of linguistic and reasoning complexity. For instance, Kocmi and Bojar, 2017 explored (Neural Machine Translation) NMT curricula based not only on sentence length but also on the syntactic structure of text. They measured text complexity with the number of clauses present in a sentence. For reading comprehension over long narratives, Tay et al., 2019 proposed a set of curriculum learning strategies based on answerability and understandability metrics. Models trained on progressively more challenging segments in terms of answerability and understandability show significant improvements in comprehension tasks.

In domains where signal quality is a primary concern, such as speech processing, the signal-to-noise ratio (SNR) provides a direct and effective heuristic for difficulty. The core principle is to treat data cleanness as a proxy for easiness, allowing a model to learn core patterns before tackling the complexities introduced by noise. For noise-robust speaker recognition, for instance, curriculum approaches have employed criteria like SNR to sort the training data. The system is first initialized on clean, high-SNR data, and then progressively more challenging, noisy samples are incorporated to build robustness against real-world degradations (Ranjan & Hansen, 2017). Similarly, for automatic speech recognition, Braun et al., 2017 introduced the "accordion annealing" (ACCAN) curriculum strategy. This method uses SNR values to create a multi-stage training schedule that starts with clean speech and gradually adds samples with increasingly higher levels of noise (i.e., lower SNR), demonstrating improved noise robustness compared to standard training

Heuristic-based curricula have also been extended to multimodal domains, such as audiovisual learning, where the complexity of the scene itself can be used as a difficulty metric. A key challenge in this area is learning to associate the correct sounds with the corresponding visual objects, especially in acoustically complex environments. To address this, Hu et al., 2020 proposed a curriculum based on the number of sound sources detected in a video. Their training starts with 'cleaner,' acoustically simple scenes containing only one or two dominant sound sources. The curriculum then gradually introduces more complex and ambiguous scenes that feature multiple, potentially overlapping, sound sources. This easy-to-hard strategy, based on a direct acoustic property of the scene, was shown to improve the model's ability to learn correct cross-modal

associations and achieve better performance.

In the context of domain adaptation, curricula are often designed around the principle of data similarity. To adapt a model from a general domain to a specific one (e.g., in NMT), a curriculum can be created by first training on examples from the source domain that are most similar to the target domain. X. Zhang et al., 2019 demonstrated this by applying the Moore-Lewis (Moore & Lewis, 2010) and cynical data selection method (Santamaría & Axelrod, 2017), where sentences from the source domain are scored based on the difference in cross-entropy between a general-domain and a target-domain language model. This method allows the curriculum to prioritize data that is most relevant to the target domain, effectively bridging the domain gap in a structured manner.

The principles of heuristic-based curricula are widely applied in computer vision. A survey by Wang et al., 2021 notes that common heuristics include image properties like resolution, blur metrics, contrast measures, or the number and size of objects present.

While these heuristic-based methods offer practical and scalable alternatives to manual annotation, their main limitation is that the chosen heuristic is only a proxy for true learning difficulty and may not well align with the model's internal learning process. This observation motivates the need for dynamic scoring functions, which we will discuss in the following section.

Dynamic scoring

In contrast to static methods that use fixed, precomputed heuristics, dynamic scoring functions re-evaluate the difficulty of data throughout the training process. This allows the curriculum to adapt to the model's evolving state, creating a more responsive and personalized learning path. The most common signal for dynamic scoring is the model's own training loss on each example, which provides a direct measure of how well the model currently understands a given data point.

Loss The core idea of a loss-based curriculum is that samples with low loss are "easy" for the current model, while those with high loss are "hard." This is arguably the most direct and intuitive method for assessing instance difficulty, as the loss function explicitly quantifies the discrepancy between the model's prediction and the ground truth. The foundational framework for this approach is Self-Paced Learning (SPL), introduced by Kumar et al., 2010. In SPL, the model begins by training only on a subset of examples that it finds easiest—those with a training loss below an initial low threshold. As the model's parameters are updated and its performance improves, this threshold (the "pace") is gradually increased, allowing more difficult, higher-loss examples to be included in the training set. This easy-to-hard progression mimics a natural learning process of mastering fundamentals before proceeding to more challenging material.

An alternative philosophy argues for focusing on the most challenging examples. In methods often termed hard example mining, the curriculum prioritizes data points with the highest loss. The rationale is that these are the examples the model is most wrong about, and focusing on them could lead to faster improvements and a more robust final model. For instance, Loshchilov and Hutter, 2015 proposed an "online batch selection" strategy where data points are ranked by their most recent loss value, and training batches are sampled with a strong bias towards these high-loss examples. This presents a direct contrast to the "easy-first" approach of classic curriculum learning.

Very often, researchers seek to combine these two philosophies, creating a curriculum that evolves from easy to hard. Huang et al., 2020 exemplify this with their "CurricularFace" loss function for deep face recognition. Their method adaptively modifies the importance of samples

within the loss calculation itself. During the initial stages of training, it assigns higher importance to easy samples (those with low loss), helping the model to build a stable foundation. As training matures, the focus gradually shifts, assigning higher importance to harder, high-loss examples to help the model learn more discriminative features.

More recent work has introduced even greater nuance, arguing that neither the easiest nor the hardest examples are optimal for training. Mindermann et al., 2022 claimed that very low-loss examples are already learned and offer little new information, while very high-loss examples might be mislabeled outliers that could corrupt the model. They propose a method for prioritizing training points that are "just right", those that the model is uncertain about but are still learnable. This represents a sophisticated evolution of loss-based curricula, moving from a simple easy-vs-hard dichotomy to a more targeted selection of the most informative examples for the model's current state.

Loss-based curricula gain their popularity because of its computational efficiency, as the loss for each instance is typically computed during the standard forward pass of the training procedure, requiring no extra overhead to generate the difficulty score. This approach is intuitive, model-agnostic and domain-agnostic, making it widely applicable across different architectures and tasks

Prediction confidence Beyond training loss, another intuitive signal for dynamic scoring is the model's own prediction confidence. Typically derived from the softmax output probabilities, confidence indicates how certain the model is about its prediction for a given sample. A high-confidence prediction suggests an "easy" example that the model understands well, while a low-confidence prediction indicates an "uncertain" or "hard" example. This duality has led to two distinct curriculum strategies in the literature: the "easy-first" approach of classic curriculum learning and the "hard-first" approach central to active learning.

Following the classic curriculum philosophy, the "easy-first" approach uses high confidence as the primary indicator of easiness. The model starts by training on examples it is most certain about, building a stable initial representation before being exposed to more ambiguous data. For example, Ao et al., 2023 propose using the model's predicted probability for the ground-truth class as a direct scoring function. In this setup, samples with higher confidence are considered easier and are prioritized in the early stages of training. This can be particularly effective for learning in noisy environments, as the model will naturally have lower confidence in potentially mislabeled examples, temporarily de-emphasizing them.

In direct contrast, the "hard-first" philosophy prioritizes the most uncertain, low-confidence examples. This strategy originates from the field of Active Learning (AL), where it is known as Uncertainty Sampling. The goal of active learning is to select the most informative data points to label next to maximally improve the model. The most informative points are typically those the model is least confident about, as they lie closest to its current decision boundary. When applied as a curriculum strategy, this "hard-first" approach continuously forces the model to confront the most challenging examples to accelerate the refinement of its predictions.

Recognizing the benefits of both strategies, some modern methods propose hybrid approaches that bridge Curriculum and Active Learning. These curricula may change their focus as training progresses. For instance, Yamada et al., 2023 propose an "active curriculum learning" strategy that begins by prioritizing easy and representative samples to establish a stable base model (the CL phase). Once the model is sufficiently trained, the strategy shifts to prioritizing uncertain, low-confidence samples to refine its decision boundaries on difficult cases (the AL phase).

The concept of confidence can also be applied more broadly than just to individual predictions. In a curriculum for Multi-Agent Reinforcement Learning, Phan et al., 2024 use a confidence-based

measure on the learning process itself. The difficulty of the task is only increased when the system's success rate is stable, determined by a high-confidence estimate of its average performance. This ensures the system has truly mastered a difficulty level before advancing, showcasing the versatility of confidence as a guiding signal for a curriculum.

Entropy As a more holistic measure of a model's predictive uncertainty, entropy is frequently used as a dynamic scoring function. While confidence scoring typically considers only the probability of the most likely class, the Shannon entropy of the model's full softmax output distribution captures the overall "peakedness" or "flatness" of its prediction. A low-entropy output, where probability is concentrated on a single class, indicates high certainty (an "easy" example). Conversely, a high-entropy output, where probability is spread across multiple classes, indicates high uncertainty (a "hard" example). As with confidence-based scoring, this measure is used to justify two distinct curriculum philosophies.

The most prominent use of entropy is in the "hard-first" strategy, which forms the basis of Uncertainty Sampling in Active Learning. The central idea is that the most uncertain examples are the most informative for improving the model. An active learning system will therefore prioritize querying the labels for unlabeled data points that yield the highest prediction entropy. When adapted to a curriculum setting, this means focusing training on the most ambiguous, high-entropy examples to most efficiently refine the model's decision boundaries.

Conversely, the classic "easy-first" curriculum approach uses low entropy as the primary indicator of easiness. In this paradigm, samples with the lowest prediction entropy are introduced first, allowing the model to build a robust understanding from the examples it is most certain about before exposing to more ambiguous cases. This use of prediction entropy as a difficulty metric is a well-established technique in the field, often cited as a standard dynamic curriculum method (Xie et al., 2024). It is particularly useful for filtering out inherently ambiguous or noisy data in the early stages of training.

The concept of entropy can also be applied in more advanced ways. In Reinforcement Learning, for instance, Satici et al., 2025 proposed a curriculum guided by the relative entropy (KL divergence) between an agent's past and current policies. In this framework, the agent is actively guided toward states where this relative entropy is high, signifying areas where its policy is changing the most and is thus most uncertain. This "hard-first" strategy uses entropy not just to measure uncertainty in a single prediction, but to quantify the dynamics of the learning process itself, showcasing the broad applicability of information-theoretic measures in curriculum design.

Information gain A more advanced class of dynamic scoring functions moves beyond measuring the model's current state (e.g., its loss or confidence) and instead attempts to estimate the information gain a training sample is likely to provide. The goal is to prioritize data points that are expected to induce the most significant or useful change in the model's internal parameters. This perspective is deeply rooted in the field of Active Learning and typically results in "hard-first" curricula that focus on the most informative, rather than the easiest, examples.

One of the most theoretically grounded approaches is Bayesian Active Learning by Disagreement (BALD), introduced by Houlsby et al., 2011. BALD formalizes sample selection as an information-theoretic problem, aiming to select data points that maximize the mutual information between the model's predictions and its parameters. In essence, it prioritizes examples where a committee of models, sampled from the Bayesian posterior, show the most disagreement. These are the points where labeling them would most reduce the model's parameter uncertainty, thereby providing the maximum possible information gain.

As Bayesian methods can be computationally intensive, a more practical proxy for information gain is the Expected Gradient Length (EGL), proposed by Settles and Craven, 2008 and linked to Bayesian by Shukla, 2022. The intuition behind EGL is that the most informative training example is the one that will cause the largest update to the model's parameters. Since the parameter update is determined by the gradient, this method prioritizes samples that are expected to produce a gradient with the largest magnitude (or norm). Like BALD, EGL is a "hard-first" strategy, as the most surprising or wrongly classified examples are the ones that typically generate the largest gradients.

While these methods often focus on finding the single most informative example at each step, modern interpretations have applied the principle of information gain in a more holistic, curriculum-like manner. For instance, in their work on building instruction-tuning datasets, Y. Chen et al., 2025 propose a method to Maximize Information Gain (MIG) over a semantic space. Instead of just selecting individual hard points, their approach iteratively selects data that provide the most new information relative to the semantic space already covered by the existing data. This ensures a balance between selecting informative examples and maintaining a diverse curriculum.

Prediction variability Beyond single-point metrics like loss or confidence, a more nuanced approach to dynamic scoring measures the variability of a model's prediction for a given sample. Again, the core intuition is that an "easy" example should elicit a stable prediction over time and under perturbation, while a "hard" or ambiguous example will cause the model's prediction to be unstable. This variability can be measured in several ways.

One approach is to track prediction stability across training time. Toneva et al., 2018 analyzed the training dynamics of individual examples and categorized them by their "forgetting events" which they defined as the number of times a model's prediction for an example flips from correct to incorrect during training. This forgetting count serves as a direct difficulty score. "Unforgettable" examples with zero forgetting events are considered the easiest, having been learned early and robustly. Conversely, samples that are frequently forgotten are considered the hardest, and the study shows these often correspond to noisy labels or inherently ambiguous cases. The study showed that removing unforgettable examples from the full training set did not hurt final performance, implying they are redundant if the rest of the more challenging data is present. Building on the related principle of tracking instability, the Active Bias method proposed by Chang et al., 2017 directly measures the variance of a model's softmax output probabilities for each sample over recent training epochs. Samples with high prediction variance are identified as unstable and are actively emphasized during training, for instance, by increasing their weight in the loss function. The authors showed that this "hard-first" curriculum, by forcing the model to focus on the samples it finds most unstable, can lead to higher final model accuracy and improved generalization compared to standard training.

Another way to measure variability is across an ensemble of models. This is the central idea behind the classic Active Learning framework of Query-by-Committee (Seung et al., 1992). In this method, an ensemble of different models is trained, and the difficulty of a data point is measured by the level of disagreement in their predictions. A high disagreement score indicates that the sample is ambiguous, lying in a region where the models are uncertain. A "hard-first" curriculum can then be designed to prioritize these high-disagreement samples, as they are the most informative for resolving ambiguity among the committee members.

External source Another curriculum design involves using an external scoring function, where one or more separate models provide the difficulty signal for the main model being trained (the

"student"). This approach allows for the transfer of knowledge from a pre-trained "teacher" or for a dedicated "mentor" to guide the learning process. The most common framework is the Transfer Teacher (Wang et al., 2021). In this setup, a teacher model is first trained on a relevant task. The knowledge from this teacher is then used to create a curriculum for the student. For instance, Weinshall et al., 2018 demonstrated that data can be sorted for a student based on the confidence of predictions of the teacher model. A variation of this approach was proposed by Xu et al., 2020, where "Cross Review" method uses an ensemble of teachers trained on different subsets of the data. An example's difficulty is then determined by how accurately it is classified in terms of loss by the teachers that were not trained on it, providing a more reliable, committee-based score.

The effectiveness of this teacher-student paradigm is supported by theoretical analysis. Hacohen and Weinshall, 2019 investigated this setup and found that a curriculum provided by a capable teacher can significantly speed up the student's convergence. Their analysis suggests that an easy-to-hard curriculum effectively modifies the optimization landscape to be steeper, allowing for larger and more productive gradient steps, while still leading the model to the same high-quality minima.

This concept of external guidance has been implemented in several other advanced frameworks. The Knowledge Distillation method (Hinton et al., 2015) can be viewed as an implicit, rich curriculum. The "soft labels" (the full probability distribution) from a powerful teacher network provide a much more nuanced training signal than hard labels, conveying the teacher's "doubts" and knowledge about inter-class similarity. This implicitly guides the student on more or less ambiguous examples. Taking this a step further, L. Jiang et al., 2018 developed MentorNet, a separate network explicitly trained to learn a curriculum for a student model. By observing the student's performance, MentorNet learns a data-weighting scheme to de-emphasize noisy or corrupt samples and guide the student's focus, proving highly effective for robust learning. Finally, state-of-the-art scoring policies like the one proposed by Mindermann et al., 2022 act as sophisticated external guides, using the student's learning dynamics to compute a "learnability" score that prioritizes the most informative examples.

Summary

The effectiveness of a curriculum learning strategy is critically dependent on its scoring function (Toneva et al., 2018). A well-chosen function that accurately maps the difficulty of data to the model's learning state can significantly accelerate convergence and improve final performance. Conversely, a poorly chosen or misaligned scoring function can offer no benefit or even harm the training process (Hacohen & Weinshall, 2019). The choice of a scoring function involves a trade-off between its accuracy as a difficulty measure, its computational cost, and its generalizability. In Table 2.1 we summarize the advantages, disadvantages, and computational overhead of the primary scoring function categories we have discussed.

As shown in Table 2.1, a key distinction between these categories is their level of specificity and adaptability. This leads to a practical trade-off between using domain-specific versus domain-agnostic scoring functions. Domain-specific functions, which are typically static heuristics like sentence length in NLP or signal-to-noise ratio in speech, can be extremely effective when the heuristic is a strong proxy for true difficulty. They are simple to implement and computationally cheap. Their primary drawback, however, is a lack of generalizability. A scoring function based on SNR is useless for a text-based task, making these methods requiring domain expertise to design. Domain-agnostic, on the other hand, are typically dynamic methods like model loss, entropy, or gradient-based scores that offer universal applicability. They can be used with any model that

Scoring function category	Advatages	Disadvantages	Computational overhead
Heuristics	Pre-computed once. Simple and intuitive.	The heuristic might not be a true proxy for difficulty for the model. Not generalizable across domains.	Negligible (one-time offline cost)
Loss	Domain-agnostic and adaptive. Directly reflects the model's current performance.	Sensitive to noise. Mislabeled examples can have very high loss and prioritized in hard-first curricula (Mindermann et al., 2022).	Low. Requires a forward pass, which is already part of the training loop.
Confidence & Entropy	Domain-agnostic and adaptive.	Can be miscalibrated (overconfident) especially in the early phases of the training (Y. Zhou et al., 2020).	Low. Requires a forward pass, which is already part of the training loop.
Prediction variability	Domain-agnostic and adaptive. Identify ambiguous examples. Robust to noise.	Requires tracking history (forgetting) or maintaining multiple models (ensembles).	Medium to high. Involves significant data logging or increased computational cost from ensembles.
Information gain	Estimates the direct impact of a sample on learning.	Can require calculating second-order statistics, gradients for all data, or complex Bayesian methods.	High to very high. Often requires extra backward passes or complex calculations.
External source	Domain-agnostic. Effective for noisy data.	Requires training and maintaining one or more separate models.	High to very high. The cost of training the teacher/mentor network can be substantial.

TABLE 2.1: Comparison of scoring function categories in curriculum learning. The table provides an overview of different families of scoring functions used to estimate sample difficulty. Each category is evaluated along three axes: its primary advantages (e.g., simplicity, robustness), disadvantages (e.g., sensitivity to noise), and associated computational overhead. This summary highlights the fundamental trade-offs between the efficiency of simpler methods and the potential effectiveness but high computational cost of more complex approaches.

has a loss function or a probabilistic output, making them the default choice for most modern CL research. By adapting to the model's specific learning trajectory, they provide a more personalized curriculum than a fixed heuristic can. Their main limitation is that they might not capture certain structural difficulties that a well-designed domain-specific heuristic can, and they often come with a higher computational cost. Domain-specific heuristics are powerful when available but the field has largely moved towards developing more sophisticated, domain-agnostic dynamic scoring functions that offer greater flexibility and adaptability across the vast range of machine learning tasks.

2.2.2 Pacing function

The pacing function represents the second fundamental component of curriculum learning framework, governing the temporal dynamics of how training examples are introduced to the learning algorithm. While scoring functions determine the relative difficulty of examples, pacing functions control the scheduling and progression through the curriculum, determining when and how quickly the model transitions from easier to more challenging examples. The design of effective pacing functions is crucial for curriculum learning success, as inappropriate pacing can lead to premature exposure to difficult examples or prolonged training on overly simple data, both of which can hinder learning performance. Pacing functions can be broadly divided into two categories: fixed and adaptive.

Fixed pace

A fixed pacing function follows a pre-defined schedule that is determined before training begins and does not change based on the model's real-time performance. The pacing function $\mathcal{P}(D, \mathbf{s}, t)$ can schedule the difficulty by defining the fraction of a dataset used at time t. Let us introduce the auxiliary function $g(t)$ that specifies the fraction of the training dataset D ordered by $\mathbf{s}$ that are included in training at the training step t.

Linear pacing The fraction of available data increases at a constant rate. This provides a smooth, gradual transition from easy to hard examples and is a common baseline. The function is defined as:

$$g(t) = g_o + (1 - g_o)\frac{t}{T} \tag{2.11}$$

Logarithmic pacing The function implements a "fast-then-slow" schedule. The data fraction increases quickly at the beginning of training and then slows down as it approaches the full dataset. This allows the model to see a wide range of easy and medium-difficulty examples relatively early, while dedicating more training time later to the harder examples. The function is defined as:

$$g(t) = g_o + (1 - g_o)\frac{\log(1 + t\alpha)}{\log(1 + T\alpha)} \tag{2.12}$$

Root pacing Similarly to the logarithmic pacing, the root pacing function includes the majority of the data in the first part of the training and then slowly adds the remaining data. The function is defined as:

$$g(t) = g_o + (1 - g_o)(\frac{t}{T})^{1/k} \tag{2.13}$$

for a root k (e.g., $k = 2$ for square root). In an empirical study, Penha and Hauff, 2019 found that root-pacing often outperformed linear pacing.

Exponential pacing In contrast to root pacing, this function implements a "slow-then-fast" schedule where the fraction of data grows exponentially. It forces the model to spend a significant amount of time mastering the easiest "core" subset of the data before rapidly introducing the remaining, more complex examples towards the end of the schedule. A properly scaled function to meet the boundary conditions is:

$$g(t) = g_o(\frac{1}{g_0})^{t/T} \tag{2.14}$$

This represents an opposite approach to example scheduling and is also analyzed in the work by Hacohen and Weinshall, 2019.

Step pacing A step pacing function defines $g(t)$ as a piecewise constant function, where the fraction of available data increases in discrete "jumps" at specific time steps. For a curriculum with K stages, the function is defined as:

$$g(t) = \begin{cases} v_1 & \text{if} \quad 0 \leq t < T/K \\ v_2 & \text{if} \quad T/K \leq t < 2T/K \\ \vdots & \vdots \\ v_K & \text{if} \quad (K-1)T/K \leq t \leq T \end{cases} \tag{2.15}$$

where v_k is the data fraction for the k-th stage. Hacohen and Weinshall, 2019 analyze this method. A well-known practical example is the "SortaGrad" method, which employs a simple single-step curriculum ($K = 2$): it trains on a small fraction of the easiest data for only the first epoch before using the full dataset for the remainder of training (Amodei et al., 2016).

Adaptive pace

Fixed pacing functions have an obvious flaw. They operate on an assumption of how a model should learn. A natural enhancement seems to lie in aligning them to the dynamics of a learning process. Adaptive pacing functions create a dynamic feedback loop, adjusting the curriculum's schedule based on the model's real-time learning progress. The idea is to introduce more difficult material only when the model demonstrates that it is "ready." This readiness (or model's capability) is measured using various signals from the training process itself. We can formalize this using a capability function, $\mathcal{C}(t)$, which provides a difficulty threshold at each training step $t \in [0, T]$, where T is the total number of training steps. An i-th sample with difficulty score s_i is included in the training set at step t only if its difficulty does not exceed the model's assumed capability, $\mathcal{C}(t)$. This creates a binary pacing function, $\mathcal{P}(D, \mathbf{s}, t)$, that "gates" each sample:

$$\mathcal{P}(D, \mathbf{s}, t) = \{(\mathbf{x_i}, \mathbf{y_i}) : s_i < \mathcal{C}(t)\} \tag{2.16}$$

The specific type of these functions uses a capability threshold $\mathcal{C}(t)$ set to the difficulty score of the hardest sample within the set $\mathcal{P}(D, \mathbf{s}, t)$. If $\forall_{t\in[0,T]}\ \forall_{s_i\in\mathbf{s}}\ s_i < \mathcal{C}(t)$, it means that all the examples are equally important throughout the training and curriculum is reduced to standard training. Several standard forms of $\mathcal{C}$ have been explored over the past years (Mindermann et al., 2022; Penha & Hauff, 2019).

The most common adaptive strategy is pacing by mastery, where the curriculum advances based on the model's competence on the current subset of data. This mastery is often measured by the average training loss. The Self-Paced Learning framework (Kumar et al., 2010) formalizes this by making the difficulty threshold (the "pace") a variable that is optimized alongside the model's

parameters. In practice, this means the threshold for including harder, higher-loss examples is only increased when the model has sufficiently converged on the current, easier set. The function is defined as follows:

$$\mathcal{P}(D, \mathbf{s}, t) = \{(\mathbf{x_i}, \mathbf{y_i}) : h(f_{\theta_t}(\mathbf{x_i}), \mathbf{y_i}) \leq K\} \tag{2.17}$$

where K is a threshold that influences the number of samples to be included into training at iteration t. If $h(f_{\theta_t}(\mathbf{x_i}), \mathbf{y_i})$ at time t is larger than K, the sample is not included in the training at time t. Small K means preference for "easy" samples with a small value of $h(\cdot)$, which can be any dynamic scoring function (e.g., confidence, loss, or uncertainty). A similar principle is used in frameworks like MentorNet, where an external mentor model monitors the student's performance and adaptively decides when to introduce more challenging or potentially noisy data (L. Jiang et al., 2018). More recent theoretical analyses have further refined this, formalizing the pace as a function that directly depends on the average loss of the currently selected easy samples (L. Jiang et al., 2015).

A higher-level approach uses more direct performance metrics, such as accuracy or validation error, to determine the model's readiness to introduce new data. This focuses on correct predictions rather than the raw loss value. A key paper by Graves et al., 2017 proposed a system where a scheduler network learns to adjust a mixture of easy and hard data distributions. This decision is guided by the student model's performance on a separate, clean validation set, thus tying the curriculum's pace directly to the model's generalization ability.

The curriculum can be advanced when the model's performance on the current data has plateaued, signaling that it has extracted most of the available information and is ready for a new challenge. For example, methods based on "Dynamic Instance Hardness" (a moving average of sample loss) first monitor the model's loss on all samples until the relative hardness rankings stabilize, and only then do they trigger a new stage of the curriculum that focuses on the dynamically identified hard examples (T. Zhou et al., 2020). Another advanced concept is to measure the "functional variance" of a network, how much the model's output function itself is changing over training epochs (Benjamin et al., 2019). A pacing function can use this signal to advance the curriculum only when the model has stopped changing, indicating it has fully learned from the current data.

This adaptive principle is particularly natural in Reinforcement Learning, where an agent's performance can be directly measured by its accumulated reward. In an RL curriculum, the difficulty of the environment or task can be dynamically tied to the agent's success. For instance, Shi et al., 2025 propose a system where a "curriculum scheduler" monitors the agent's recent reward signal. The target difficulty of the task is increased only when the agent's performance improves and is decreased if the agent begins to fail. This ensures the agent is always appropriately challenged, preventing it from getting stuck on tasks that are too hard or wasting time on tasks that are too easy.

Summary

Adaptive mechanisms reflect a broader trend in machine learning towards greater automation and meta-learning. Fixed functions provide an interpretable baseline, but the state-of-the-art lies in adaptive approaches that treat the curriculum schedule not as a static hyperparameter, but as a dynamic policy to be optimized. Key open research questions remain in this domain, focusing on how to design these adaptive schedulers to be more robust, computationally efficient, and capable of automatically balancing the exploration of new, harder data with the consolidation of existing knowledge.

Chapter 3

Typicality scoring

This chapter challenges the conventional approach to curriculum learning, which is often predicated on ordering training data by a measure of "difficulty". This paradigm can be ambiguous. Here, we explore an alternative strategy based on typicality scoring. This approach is founded on the hypothesis that a more robust and efficient learning path can be achieved by first building a model of the most representative data. The curriculum, therefore, proceeds not from easy to hard, but from typical to atypical, allowing the model to establish a strong understanding of the core concepts of each class before being exposed to more ambiguous or peripheral examples.

The work presented in this chapter is based on our publication, "Curriculum Learning Revisited: Incremental Batch Learning with Instance Typicality Ranking," which was presented at the International Conference on Artificial Neural Networks, ICANN 2021 (Krysińska, Morzy, et al., 2021).

To investigate this paradigm, this chapter is structured as follows. Section 3.1 begins with an introduction that formally defines the concept of typicality scoring and situates it within the context of related work in data selection and curriculum design, before detailing the novel graph-based typicality scoring method proposed in this thesis. Section 3.2 describes the experimental setup, including the datasets and evaluation protocols used to test our method. Section 3.3 presents the empirical results, comparing the performance of typicality-based curricula against standard baselines. Finally, Section 3.4 offers concluding remarks, summarizing the findings and discussing promising avenues for future research in representative data scoring.

3.1 Introduction

Defining the difficulty of a training instance is hard and it requires several additional assumptions on how to assess this measure. As we have seen, difficulty can be a proxy for various properties: a sample can be considered "hard" because it is noisy, contains complex features, lies near a decision boundary, or is simply underrepresented in the data. This ambiguity can be problematic, as a curriculum that prioritizes high-loss examples, for instance, may inadvertently focus on mislabeled data, hindering the learning process.

We explore an alternative paradigm for curriculum design based not on difficulty, but on typicality. Instead of asking "what samples are easy, what samples are hard?", we ask "what samples are the best representatives of their class?". A typicality-based curriculum operates on the assumption that the most effective learning path begins with examples that are central and prototypical to their class. By training on the most representative examples first, a model can build a robust, low-variance understanding of the core concepts of each category. Once this stable foundation is

established, the curriculum can then introduce more atypical, ambiguous, or boundary examples to refine the model's decision boundaries. This approach seeks to provide a cleaner learning signal by prioritizing representativeness over notions of difficulty. The main difference between classic approaches to curriculum learning and the approach shown in this chapter is the philosophy of constructing the scoring function.

3.1.1 Related work

Large datasets often contain significant redundancy, and effective training does not require processing every single example. Instead, by selecting a representative subset, it is possible to significantly reduce training time and computational cost without sacrificing final model performance (Toneva et al., 2018). The goal of instance selection is to identify a smaller, more representative subset of a large dataset to make the training process more efficient and robust. The core challenge, therefore, lies in defining what makes the subset "representative".

The notion of data representativity is itself a complex topic. As formally discussed by Clemmensen and Kjærsgaard, 2022, a representative training set is one whose statistical properties accurately reflect the real-world data distribution on which the model will eventually be deployed. They claim that representativity must be considered in the context of the specific task and model. A training dataset is considered representative for a given model if the conclusions drawn from training the model on that sample dataset are the same as the conclusions that would have been drawn if the model had been trained on the entire real-world distribution data we ultimately care about. A curriculum based on typicality scoring is a practical implementation of this principle. It operates on the hypothesis that the most central or prototypical examples of a class are the most effective for learning the core statistics of the data distribution. By starting with these examples, a model can build a stable, low-variance foundation before being exposed to more ambiguous boundary cases.

The idea of organizing a curriculum based on the typicality or representativeness of examples, rather than their difficulty, has been explored through several related lines of research. The common line for these is that a model benefits from first learning the most central or prototypical examples of a class before moving to more ambiguous or atypical cases. The primary distinction between these methods lies in how they formally define and measure "typicality".

One of the most intuitive ways to define typicality is through data clustering (Chaudhry & Sharma, 2024; Choi et al., 2019; Guo et al., 2018; Lin et al., 2025). In this approach, examples that are close to the center of a cluster in a feature space are considered typical. A recent work by Lin et al., 2025 introduces a "prototype-driven curriculum", where K-means clustering is first used to identify these central prototypes. The curriculum begins by training on these prototypical examples and only gradually expands to include more varied and complex samples from the cluster peripheries. Similarly, Chaudhry and Sharma, 2024 propose a method where each sample is scored based on the density of its local neighborhood. The assumption is that typical examples reside in high-density regions of the feature space. The curriculum then proceeds from the highest-density to the lowest-density regions, ensuring the model first learns from the most common examples.

One of the potential limitations of curricula based on typicality is the risk of selecting a set of highly redundant examples, as the most prototypical samples are often very similar to one another. This has led to research that focuses not just on typicality, but also on diversity. For example, C. Zhang et al., 2017 propose using Determinantal Point Processes (DPPs) to select mini-batches for training. DPPs are a tool that can select a subset of items that are both high-quality (e.g., typical) and diverse (i.e., dissimilar from each other). This introduces a crucial trade-off: an

effective curriculum may need to balance learning from the most representative examples with being exposed to a wide variety of non-redundant data to ensure the model generalizes well.

Another significant challenge arises from the presence of outliers and mislabeled examples. A purely density-based or centroid-based approach assumes that the most representative samples are those at the center of a data cluster. However, in real-world datasets, this is rarely the case. An outlier from one class can be located deep within the feature space of another class, near its centroid or in a high-density region. As can be observed in the visualizations from Lin et al., 2025, it is possible for examples from one class to be closer to the centroid of another class than some of that class's own members. Under a typicality curriculum, such a mislabeled example would be mistakenly prioritized early in training, potentially corrupting the model's initial understanding of the class boundary. To address this problem, we propose a typicality metric that is more robust to such noise by measuring the representativeness of an example primarily within the context of its own class, rather than based on the global data distribution alone.

3.1.2 Method

It is hard to provide a formal definition of representativeness and typicality. Intuitively, an instance is typical if it is similar to many instances of the same class. In other words, a typical instance is a good representative of its class. An instance is atypical if it is similar to many instances of other classes, so features of an atypical instance do not provide reliable information about the properties of the class. If an instance is not similar to other instances, irrespective of their class assignment, such instance is of limited importance to the training process since it does not contribute to the generalization capabilities of the learned model. We refer to such instances as nonaligned. Thus, our method tries to build a curriculum of training instances, starting from the most typical instances, then proceeding to nonaligned instances, to finish with atypical instances. The main idea is to focus most of the learning process on typical instances and downplay the importance of atypical instances.

Our method works provided that there is a function $sim(\mathbf{x_i}, \mathbf{x_j})$ which computes the similarity between samples. The choice of a particular similarity function is irrelevant from the point of view of the procedure. Let $C = \{c_1, c_2, \ldots, c_k\}$ be the set of classes, and $D = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\}$ be the set of training examples. Let

$$\overline{sim} = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} sim(\mathbf{x}_i, \mathbf{x}_j) \tag{3.1}$$

be the average instance similarity. Then the instance similarity graph is defined as a pair $\langle D, E \rangle$, where X is a set of nodes (and each node represents a single training instance), and $E = \{(\mathbf{x_i}, \mathbf{x_j}) : \mathbf{x_i}, \mathbf{x_j} \in D \wedge sim(\mathbf{x_i}, \mathbf{x_j}) \geq \overline{sim}\}$ is the set of edges. An edge exists if and only if the similarity between instances is greater than the average instance similarity.

To measure the typicality of each instance we use the topology of the instance similarity graph. We compute three centrality indexes and treat them as proxies for instance typicality. Below, we present these centrality indexes which serve as a scoring function $\mathcal{S}$ in curriculum. We thoroughly describe rationale behind using them to represent instance typicality.

Degree centrality The degree centrality of an instance $\mathbf{x_i}$ is the number of instances adjacent to $\mathbf{x_i}$ in the instance similarity graph. Formally, $\mathcal{S}_{DC}(\mathbf{x_i}) = |\{\mathbf{x_j} : (\mathbf{x_i}, \mathbf{x_j}) \in E\}|$. Degree centrality captures the overall typicality of each instance by promoting in the curriculum instances that are similar to many other instances. Thus, training the model on these instances allows to generalize

the model and cover many training instances with one selected prototype. The main drawback of degree centrality is that, similarly to some density-based methods, it is class-agnostic. So, an instance that is highly connected to instances from multiple other classes could receive a high score, making it a poor representative of its own class.

Entropy centrality Next, we introduce the entropy centrality. We use classical Shannon's information entropy (Shannon, 1948) to measure the homogeneity of class assignments of a given sample nearest neighborhood. Let $N(\mathbf{x_i}) = \{\mathbf{x_j} : (\mathbf{x_i}, \mathbf{x_j}) \in E\}$ be the nearest neighborhood of $\mathbf{x_i}$. The probability of finding class c_l in $N(\mathbf{x_i})$ is given by:

$$p(c_l, \mathbf{x_i}) = \frac{|\{\mathbf{x_j} \in N(\mathbf{x_i}) : y_j = c_l\}|}{|N(\mathbf{x_i})|} \tag{3.2}$$

The entropy centrality of an instance $\mathbf{x_i}$ is simply the entropy of the distribution of classes in the nearest neighborhood of $\mathbf{x_i}$, $\mathcal{S}_{EC}(\mathbf{x_i}) = -\sum_l p(c_l, \mathbf{x_i}) \ln p(c_l, \mathbf{x_i})$. Unfortunately, this formulation of the centrality index does not safeguard against picking very atypical instances early in the curriculum, for the same reason as the degree centrality. An instance connected only to instances of the opposite class will have zero entropy, despite being a very poor representative of its class. To circumvent this problem, we have to modify the entropy centrality formula. Let $c^*(\mathbf{x_i}) = kp(c_k, \mathbf{x_i})$ be the majority class in the nearest neighborhood of $\mathbf{x_i}$. We can define an auxiliary function

$$g(\mathbf{x_i}) = \begin{cases} +1 & if\, c^*(\mathbf{x_i}) = y_i \\ -1 & otherwise \end{cases} \tag{3.3}$$

which indicates if the class of the instance $\mathbf{x_i}$ is the same as the majority class in its nearest neighborhood. We define the class-adjusted entropy centrality as follows: $\mathcal{S}_{ECCA}(\mathbf{x_i}) = \frac{g(\mathbf{x_i})}{\mathcal{S}_{EC}(\mathbf{x_i})+1}$. According to this formulation, the class-adjusted entropy of $\mathbf{x_i}$ is minimized when all instances in the nearest neighborhood of $\mathbf{x_i}$ belong to the same class as $\mathbf{x_i}$, and maximized when they belong to the opposite class. Still, this solution has a drawback: the formula does not consider the cardinality of neighborhoods of instances.

Degree-adjusted entropy centrality In order to solve the above problem, we also experimented with a simple heuristic which combines both degree centrality and entropy centrality. The degree-adjusted entropy centrality is given by: $\mathcal{S}_{ECDA}(\mathbf{x_i}) = |\{\mathbf{x_j} \in N(\mathbf{x_i}) : y_j = y_i\}| - \mathcal{S}_{ECCA}(\mathbf{x_i})|N(\mathbf{x_i})|$. This formula takes into account both the homogeneity of the nearest neighborhood of a sample and its size.

3.2 Experiments

To empirically validate the effectiveness of the proposed typicality-based curriculum learning strategies, a series of experiments were conducted on a diverse collection of tabular datasets. This section details the datasets, model architecture, hyperparameters, and specific configurations for the baseline and curriculum-based methods used in experiments.

Experiments were carried out on 9 popular datasets from the UCI Machine Learning Repository[1], chosen to represent a variety of data types and complexities. The datasets include those with purely categorical or numerical attributes, as well as mixed-type data. A summary of the dataset statistics is provided in Table 3.1.

[1] https://archive.ics.uci.edu

Dataset	Instances	Attributes	Types
Audiology	200	69	categorical
Breast cancer	569	5	numerical
Cars	1728	21	categorical
Credit screening	690	15	mixed
Diagnosis	120	6	mixed
Hepatitis	155	19	categorical
Horse colic	300	26	mixed
House votes	435	16	categorical
Parkinson	195	22	numerical

TABLE 3.1: Summary of datasets for typicality scoring experiments. All datasets were sourced from the UCI Machine Learning Repository and were chosen to represent a diverse range of sizes, attribute counts, and data types (categorical, numerical, or mixed).

In order to isolate the effect of the data ordering and avoid over-optimizing for any single dataset, a fixed set of hyperparameters was used across all experiments. For all conditions, a simple feed-forward neural network was trained using the Adam optimizer. The network architecture consists of an input layer with a size equal to the number of attributes in the dataset, and a single output layer. For binary classification tasks, the output layer contains a single node with a sigmoid activation function. For multi-class classification, the output layer contains a node for each class with a softmax activation function. The training protocol was designed to ensure a fair comparison between the baseline and the curriculum methods. For each experimental run, the data was randomly split into a training set and a validation set, with the validation set containing 33% of the instances. This process was repeated across multiple runs to avoid any bias from a particularly favorable or unfavorable data split.

The baseline model consists of the feed-forward network architecture described above, trained for 50 epochs with a batch size of 32. The Adam optimizer was used with a learning rate of 0.001. For the baseline, training batches were sampled uniformly at random from the training set, and the data was reshuffled between epochs.

The curriculum-based models used the identical network architecture and training parameters as the baseline. The only difference was the order in which training instances were presented.

3.3 Results

This section presents the empirical evaluation of the proposed typicality-based curricula against a standard baseline. The primary goal is to determine if ordering training instances based on a typicality score offers a measurable benefit over the conventional training approach. The key metric used in the experiments is the validation set accuracy. We verify if curriculum learning is able to improve the training by comparing the baseline method to the various curriculum orderings, noting that these two approaches differ only in the order of training instances, with all other hyperparameters being the same.

To assess the statistical significance of our findings, we first apply the non-parametric Friedman test (Friedman, 1937). The null hypothesis states that there is no observable difference between the performance of the baseline and the different curriculum groups. With a significance level set to $\alpha = 0.05$, the test yields a p-value of 3.78×10^{-6}, which allows us to confidently reject the null hypothesis and conclude that there are significant differences among the methods.

Given this result, we conduct a post-hoc analysis using the Nemenyi test (Nemenyi, 1963) for pairwise comparisons to identify exactly which methods differ. Table 3.2 presents the results of this comparison between the baseline, the degree centrality curriculum ($\mathcal{S}_{DC}$), the entropy

	Baseline	$\mathcal{S}_{DC}$	$\mathcal{S}_{ECCA}$	$\mathcal{S}_{ECDA}$
Baseline	−1.0000	0.0010	0.0010	0.0503
$\mathcal{S}_{DC}$	0.0010	−1.0000	0.9000	0.1586
$\mathcal{S}_{ECCA}$	0.0010	0.9000	−1.0000	0.2980
$\mathcal{S}_{ECDA}$	0.0503	0.1586	0.2980	−1.0000

TABLE 3.2: Post-hoc analysis of curriculum learning strategies. The table displays the p-values from pairwise statistical comparisons between the baseline (uniform sampling) and three curricula: the Degree Centrality curriculum ($\mathcal{S}_{DC}$), the Entropy Centrality curriculum ($\mathcal{S}_{ECCA}$), and the Degree-adjusted Entropy curriculum ($\mathcal{S}_{ECDA}$). The p-values indicate a statistically significant difference (p ¡ 0.05) between the baseline and the curricula using either degree or entropy centrality. However, the test does not detect a statistically significant difference between the baseline and the degree-adjusted entropy curriculum.

	Baseline	$\mathcal{S}_{DC}$	$\mathcal{S}_{ECCA}$	$\mathcal{S}_{ECDA}$
Baseline	−1	163	163	150
$\mathcal{S}_{DC}$	95	−1	130	110
$\mathcal{S}_{ECCA}$	98	130	−1	117
$\mathcal{S}_{ECDA}$	110	148	138	−1

TABLE 3.3: Pairwise rank comparison of curriculum learning strategies. The table displays a "win-loss" matrix where each cell indicates the number of experiments in which the method in the row achieved a better final rank than the method in the column. The analysis shows that both the Degree Centrality ($\mathcal{S}_{DC}$) and Entropy Centrality ($\mathcal{S}_{ECCA}$) curricula underperformed the baseline.

centrality curriculum ($\mathcal{S}_{ECCA}$), and the degree-adjusted entropy curriculum ($\mathcal{S}_{ECDA}$). The p-values indicate that there is a statistically significant difference between the baseline method and the curriculum methods using either degree centrality or entropy centrality. However, the test does not detect a statistically significant difference between the baseline and the curriculum using the degree-adjusted entropy centrality.

Pairwise comparisons inform us only of the existence of a difference, but not of its direction. Therefore, we follow with rank comparisons between the groups, presented in Table 3.3. The analysis shows that the baseline outperforms the degree centrality curriculum in 163 experiments, while the degree centrality curriculum is superior in only 95. Similarly, the baseline outperforms the entropy centrality curriculum in 163 experiments, with the curriculum performing better in just 98. Since the Friedman test has already confirmed a significant difference for both of these curriculum methods, and the rank comparisons clearly show they underperform the baseline more frequently, we can conclude that for this set of experiments, both the degree centrality and entropy centrality curricula lead to worse final model performance than standard, non-curriculum-based training.

3.4 Conclusions

This chapter introduced a novel approach to curriculum learning based on the principle of typicality scoring. Moving away from the ambiguous and often problematic concept of "difficulty," this work is founded on the hypothesis that a more robust and efficient learning path can be forged by starting with the most representative examples of a class before progressing to more atypical instances.

We explored a specific region of the curriculum design space: a data-driven, static curriculum based on typicality. While we have highlighted the advantage of this approach over difficulty metric, we also identified a key limitation in existing typicality measures: their susceptibility to outliers and mislabeled examples that may reside in high-density regions or near class centroids. Our proposed class-aware centrality metrics are a first step toward mitigating this issue.

We presented a new method for quantifying typicality using a graph-based representation of the dataset and demonstrated its effectiveness. Our proposed method fits within the broader

curriculum learning framework as a data-driven curriculum with a static scoring function. The core components are implemented as follows:

- **Scoring Function**: The typicality score is derived from the topology of an instance similarity graph. We proposed and evaluated three specific, novel scoring functions based on graph centrality metrics: Degree Centrality, which measures an instance's overall similarity to the dataset; Class-Adjusted Entropy Centrality, which measures the class-purity of an instance's local neighborhood; and Degree-Adjusted Entropy Centrality, which balances both neighborhood size and purity.
- **Pacing Function**: In our experiments, these static scores were paired with a standard fixed pacing function, where the subsequent batches contain more atypical examples than previous batches. After each epoch, the difficulty level was reset; we started with the easiest batches and ended the epoch with the most atypical batch.

However, the empirical results did not support our initial hypothesis. The analysis showed that curricula based on our typicality metrics led to a statistically significant degradation in final model performance compared to the standard, non-curriculum baseline of random shuffling. No ordering strategy provided an advantage in terms of peak accuracy.

The key reason for this underperformance appears to be the detrimental interaction between a static, ordered curriculum and mini-batch stochastic gradient descent (SGD). By strictly ordering the training data, we inadvertently create highly homogenous mini-batches. Early in an epoch, the model trains on batches of low-variance, typical examples, often leading to rapid initial convergence. Conversely, the batches at the end of the epoch consist exclusively of atypical, high-variance examples. This introduces a sudden and significant amount of gradient noise late in the training process. The model, having already settled into a stable state based on the "cleaner" initial data, is unable to effectively temper this late-stage noise. Instead of refining the decision boundary, these high-variance gradients can destabilize the learned features, preventing the model from converging to a more robust final solution.

Beyond this fundamental issue of gradient variance, the process of constructing the instance similarity graph and computing centrality scores adds a considerable pre-processing cost. When combined with the negative performance impact, this suggests that a naive, single-pass typicality curriculum is not a practical alternative to a simpler, well-tuned random shuffling baseline.

In conclusion, while the principle of starting with typical examples remains appealing, this investigation highlights a critical flaw in its static implementation. The problem lies not necessarily with the scoring of examples, but with the creation of homogenous batches that introduce noisy updates at a vulnerable point in the training cycle. This recognition motivates the need for more sophisticated pacing functions that can intelligently orchestrate batches to balance typical and atypical examples, a topic we will explore in Chapter 5 and Appendix A.

Chapter 4

Variance of Gradients scoring

This chapter introduces VoG-Guided Learning (VGL), a novel data-driven curriculum learning framework designed to improve the training efficiency and final performance of deep learning models. While the principles of VGL are broadly applicable, this chapter makes a deliberate move beyond testing on well-known, sanitized benchmarks. Here, we demonstrate the robustness and utility of VGL by applying it not only to standard datasets but also to challenging, real-world problems. Specifically, we evaluate VGL on noisy, complex social media data curated to address the socially critical issue of online misinformation, thereby testing our method in an environment where performance improvements have tangible, real-world implications.

Before delving into the specifics of this research, it is important to contextualize my broader scientific activities that led to the creation of the novel datasets used in this evaluation. A significant portion of my work involved extensive contributions to the Webimmunization project[1], an interdisciplinary initiative focused on understanding and countering online misinformation. My role spanned the full research lifecycle, from foundational literature analysis to large-scale data engineering and ethical framework development, culminating in three co-authored articles.

The work began with a rigorous systematic review of psychological interventions against online misinformation (Gwiaździński et al., 2023), in which I helped devise the structure of the paper that analysed over 3,500 papers to synthesize the state of the art and identify critical gaps for future research. Subsequently, I contributed to an innovative study on information exposure, pioneering an ethical framework for using bots in social media research to analyze how initial friend selections impact exposure to pro- and anti-vaccination content (Krysińska, Wójtowicz, et al., 2021). This research directly led to the creation and public release of the "Vaccine Attitudes" dataset [2], a key benchmark used in this chapter. Finally, I played a critical data-centric role in a study published in the prestigious journal Nature (Kunst et al., 2024). This involved data engineering effort: collecting and processing a massive amount of data, approximately 50 million GB of data from the X (formerly Twitter) platform over more than a year. I then designed and managed a large-scale annotation campaign on Prolific platform to label 60,000 social media posts, resulting in the creation of the COVID-19 "Conspiracy Theories" dataset [3] published on HuggingFace platform. I developed a machine learning technique to assess individuals and larger networks susceptibility to misinformation, measured as their beliefs in conspiracy theories. The Conspiracy Theories dataset, also used as a benchmark in this chapter, was instrumental in combining AI-driven analysis of big data with psychological self-reports to identify risk factors for belief in online conspiracy theories.

[1] https://webimmunization.cm-uj.krakow.pl/en/webimmunization funded from the EEA Financial Mechanism 2014–2021. Project registration number: 2019/35/J/HS6/03498

[2] https://huggingface.co/datasets/webimmunization/COVID-19-vaccine-attitude-tweets

[3] https://huggingface.co/datasets/webimmunization/COVID-19-conspiracy-theories-annotated-tweets

This chapter is structured to present the VoG-Guided Learning framework in full detail. Section 4.1 provides an introduction, covering the related work and a formal description of the VGL method. Section 4.2 describes the experimental methodology, including the datasets and the training setup. Section 4.3 presents a comprehensive analysis of the results, evaluating VGL's impact on model dynamics and final performance. Finally, Section 4.4 offers concluding remarks, summarizing the contributions of this work and outlining promising directions for future research.

4.1 Introduction

Achieving high accuracy of the modern large machine learning models often requires extensive training on large, uncurated datasets. As we mentioned earlier, conventional training strategies treat all examples equally. In some cases it can lead the model to spend considerable time revisiting "easy" examples that have been learned already. This approach overlooks the fact that not all training examples contribute equally to model improvement and that resampling already learned instances may be redundant, especially as training progresses. As such, it seems promising to optimize the training process by focusing on the most informative examples. The problem of optimizing the training process by selecting the most informative examples is a long-standing area of research in machine learning. The underlying principle is that by focusing on the "harder" or more informative samples, we can potentially achieve better performance and faster convergence (Chang et al., 2017; Katharopoulos & Fleuret, 2018; Lapedriza et al., 2013; Mindermann et al., 2022; Toneva et al., 2018).

The need to select an effective subset of samples is a pressing problem in the current state of machine learning field development, as the problem of data redundancy is particularly pronounced in the context of massive, web-crawled datasets commonly used for training large language and vision models these days. These datasets often consist of near-duplicate examples that may not contribute significantly to learning.

The following sections review several lines of research that aim to address this issue by moving away from uniform data sampling. We also introduce a novel method for importance approximation in the context of importance sampling. Our work is based on several key research streams in this domain, including importance sampling, curriculum learning, and online batch selection.

4.1.1 Related work

A leading approach to non-uniform sampling is importance sampling, where each training example is assigned a weight, and samples are drawn with probabilities proportional to these weights.

Various metrics have been proposed to define a sample's "importance". Some methods use the training loss, with the intuition that samples with higher loss are more difficult and should be prioritized (A. H. Jiang et al., 2019; Loshchilov & Hutter, 2015). Another prominent approach is to base importance sampling on the uncertainty and variability of predictions (Coleman et al., 2019; Swayamdipta et al., 2020; Toneva et al., 2018). An interesting alternative to the standard measure of prediction variability was introduced by Jeong et al., 2019. They measured prediction variability with data augmentation. The principle here is that a model that has learned a concept robustly should be invariant to small, semantic-preserving changes in the input. As demonstrated, a sample's importance can be measured by the stability of the model's prediction across different augmentations of that sample (e.g., rotating or changing the color of an image). An example that yields a consistent prediction across all augmentations is easy, while one whose prediction changes is hard.

However, probably the most common strategy in importance sampling is using the L2-norm of the gradient with respect to the model's parameters as a proxy for the "importance" of i-th sample at time t: $w_{i,t} \propto ||\nabla \ell(\theta_{t-1}; x_i)||^2$

The intuition is that samples with larger gradient norms are the ones on which the model is more "wrong" and, therefore, can provide a stronger learning signal. Several works demonstrated that sampling more difficult examples can lead to better final performance of the model or faster convergence (Alain et al., 2015; Katharopoulos & Fleuret, 2018; Loshchilov & Hutter, 2015). The work by Katharopoulos and Fleuret, 2018 and Alain et al., 2015 explored how prioritization of examples with high gradient norm influences the training dynamics and showed that it can reduce the variance of gradients, leading to more stable and faster convergence. They also highlighted the practical challenge of the computational overhead required to calculate these gradients for the entire dataset at each epoch.

4.1.2 Method

We introduce our dynamic training methodology, which leverages the Variance of Gradients (VoG), a metric originally proposed for auditing datasets and identifying challenging examples (Agarwal et al., 2022), as a novel importance score for weighted sampling. VoG provides a measure of a model's uncertainty about a specific sample by analyzing the stability of its gradient signal. The key distinction in the VoG calculation is that the gradient is computed **with respect to the input features** (e.g., the pixels of an image), not the model's parameters. In machine learning, we usually talk about the gradient calculated with respect to the model's parameters; hence, we want to once again emphasize that VoG is the variance of gradients calculated with respect to the input features. This gradient is often used in explainability methods. It identifies parts of the input that need to change the least to affect the model's output the most. The variance of this gradient across different stages of training reveals the model's consistency in its reasoning:

- A **low VoG** score indicates that the model consistently relies on the same input features to make its decision about a sample, epoch after epoch. This suggests the model has a stable and confident understanding of the example.
- A **high VoG** score signifies that the model's focus is unstable. The input features it considers important for its prediction change significantly as training progresses. This "gradient disagreement" indicates conflict or uncertainty; the model is struggling to form a consistent explanation for its classification of the sample.

Formally, to compute the VoG score for a single input sample $\mathbf{x_i}$ with d dimensions, we first need a set of K model checkpoints, $\{\theta_1, \theta_2, \ldots, \theta_K\}$, saved at different points during training. For each checkpoint θ_t, we compute the gradient of the model's pre-softmax output (logit) for the true class, y_i, with respect to the input $\mathbf{x_i}$. This gives us a gradient matrix: $G_t = \nabla_{\mathbf{x_i}} y_i$, where y_i is a true class of $\mathbf{x_i}$. G_t matrix is the shape of $\mathbf{x_i}$ with d dimensions. Then the variance of gradients is calculated for each of d dimensions across all K checkpoints and averaged to compute a single scalar for a sample:

$$\text{VoG}(\mathbf{x_i}) = \frac{1}{d} \sum_{j=1}^{d} \sigma^2(G_1^j, G_2^j, \ldots, G_K^j) \tag{4.1}$$

VoG score represent the variance of gradients in the input space, providing a direct and intuitive measure of the model's uncertainty about an example.

FIGURE 4.1: Top-25 images with the lowest (first column) and highest (second column) VoG scores in the early (first row) and late (second row) training stage. Figure from Agarwal et al., 2022.

As shown in Agarwal et al., 2022, what a model considers "easy" changes throughout training. For instance, at the beginning of training, the model might learn that an apple is best represented as a "red, round object," making prototypical red apples the easiest examples to classify. However, as training progresses and the model's internal representation of an "apple" becomes more robust and nuanced, the easiest examples may shift to become apples of various colors against a plain background. The challenge then moves to more complex scenes or atypical examples (Figure 4.1).

By using VoG, our method dynamically tracks this shifting landscape of difficulty. It continuously adapts its focus to the examples that are most challenging for the model in its current state, rather than relying on a fixed notion of what is consistently difficult. This adaptability is key to improving overall training efficiency. This approach may seem to stand in direct contrast to the traditional curriculum learning paradigm introduced by Bengio et al., 2009. Rather than progressing from easy to difficult examples, our method prioritizes the hardest instances for the model at its current training stage, a strategy consistent with an anticurriculum. While seemingly counterintuitive, this approach effectively mirrors an advanced human learning strategy. For example, a student preparing for a difficult exam does not endlessly review mastered material. Instead, they focus their efforts on the concepts and problems they struggle with the most. In the same way, our method prioritizes examples for which the model has not yet found a stable, consistent recognition pattern. This forces the model to refine its understanding of ambiguous cases and challenging data points.

Our training algorithm, VoG Guided Learning (VGL), integrates VoG scores into a probabilistic sampling scheme. We replace static or instantaneous measures of sample difficulty, like loss or gradient norm, with a more dynamic metric - VoG. By prioritizing examples that exhibit high gradient variance, our method focuses the model's capacity on the most informative samples. The informativeness is modeled by model uncertainty and ambiguity. Additionally, this approach

acknowledges that the notion of a "difficult" example is fluid and evolves as the model learns. The difficulty of a sample is not a static property but is instead dependent on the current state of the model's learned representation.

Instead of hard pacing, which enforces presenting only examples that meet the current difficulty criterion, we allow for probabilistic sampling. We dynamically adjust the sampling probabilities at each epoch based on the VoG scores computed in the previous epoch. This creates a curriculum where the model focuses progressively on the examples it finds most challenging.

The process for each epoch t (from 1 to T) is as follows:

1. **Probabilistic sampling**: A training subset D' of size m is drawn from the full dataset D. For the first two epochs ($t \in \{1, 2\}$), this sampling is uniform. For all subsequent epochs, each sample $(\mathbf{x_i}, y_i) \in D$ is drawn with a probability p_i^t proportional to its VoG score from the previous epoch: $p_i^t \propto \text{VoG}^{t-1}(x_i)$

2. **Model training**: The model is trained for one epoch on the sampled subset D' using a standard optimization algorithm (e.g., mini-batch SGD or Adam) to update the model parameters from θ_{t-1} to θ_t.

3. **VoG score calculation**: After the training epoch is complete, we compute the gradient G_t and the corresponding VoG score for each sample $\mathbf{x_i}$ in the full dataset D using the updated model parameters θ_t. These scores will then be used to form the sampling distribution for the next epoch, $t + 1$.

This iterative process allows the model to continuously shift its attention. As the model learns and certain examples become "easier" (their VoG scores decrease), they are naturally sampled less frequently, allowing computational resources to be reallocated to newly identified or persistently difficult examples.

4.2 Experiments

To evaluate the effectiveness of the proposed VGL method, we conducted experiments on a diverse suite of benchmark datasets and model architectures. This selection allows us to test our approach across different data modalities (images and text), task complexities, and dataset characteristics. This section details the datasets used and the experimental setup.

4.2.1 Datasets

We selected four datasets to provide a comprehensive evaluation: MNIST for image classification, AG News for topic classification, and two datasets collected during the Webimmunization project for fine-grained sentiment and conspiracy theory detection in social media text. These datasets vary significantly in terms of size, task difficulty, and inherent data redundancy, providing a robust testbed for our method.

MNIST The MNIST database (L. Deng, 2012) is a canonical dataset in machine learning, consisting of 70,000 28x28 grayscale images of handwritten digits (0-9). The dataset is split into a training set of 60,000 images and a test set of 10,000 images. The task is a 10-class image classification problem. For modern neural networks, it is considered a relatively simple, "solved" task. In present-day, it is used as a benchmark for initial validation and ensuring that performance improvements are not merely artefacts of a complex task. The dataset has relatively low variance.

The images are size-normalized and centered, with the primary source of variation being different handwriting styles.

AG News For our primary text classification benchmark, we used the AG News dataset (X. Zhang et al., 2015), a large-scale collection of news articles. The task involves classifying articles into one of four distinct categories: World, Sports, Business, or Sci/Tech. The dataset contains 120,000 articles for training and 7,600 for testing.

Webimmunization COVID-19 Vaccine Attitude To evaluate our method on a challenging classification task using real-world, noisy data, we employ the Webimmunization COVID-19 Vaccine Attitude dataset. This curated collection consists of 2,564 English-language tweets focused on the public discourse surrounding COVID-19 vaccines during the initial vaccine rollout period.

The data was collected between April and May 2021 using a specialised, two-stage process designed to capture emotionally charged content. First, we queried the Twitter API v2 using a manually curated set of hashtags known to be associated with strong opinions (e.g., #getvaccinated, #mRNA, #VaccinesKill, #BillGatesVaccine). This created an initial seed set of emotionally loaded tweets. In the second stage, we collected additional tweets from the timelines of the authors who posted the seed tweets. This entire collection was then filtered by a topic classifier (Krysińska, Wójtowicz, et al., 2021) to remove any content not strictly related to vaccines. This methodology was intentionally designed to produce a dataset rich in emotional content rather than purely informational posts.

The primary task for this dataset is sentiment analysis. However, unlike standard sentiment tasks, the goal is to classify each tweet into one of three attitudes towards vaccines. Eight annotators manually annotated over 2,500 tweets. We have assumed the existence of three classes:

- **Pro-vaccine**: the tweet unequivocally suggests support for getting vaccinated against COVID-19
- **Neutral**: the tweet is mostly informative, does not show emotions vs. presented information, contains strong positive or negative emotions, but concerning politics (vaccine distribution, vaccine passports, etc.)
- **Anti-vaccine**: the tweet is clearly against vaccination and contains warnings, conspiracy theories, etc.

The reliability of the labelling process was measured using a standard metric for inter-annotator agreement. According to a popular interpretation of Fleiss' kappa (Fleiss, 1971), the annotators are in fair agreement ($\kappa = 0.3753$). The annotators were largely coherent when labelling distinctly anti-vaccine tweets, but struggled to differentiate between the pro-vaccine and neutral classes. This difficulty often arose from the lack of broader context needed to interpret an author's true intent from a short text, leaving room for divergent opinions.

This fair agreement score underscores the real-world ambiguity and complexity inherent in social media data, where perception requires a nuanced understanding of sarcasm, politics, and specific social events. Consequently, this dataset provides a unique and demanding benchmark, requiring a model to move beyond simple keyword matching and grasp the subtle contextual cues of the vaccine debate.

Webimmunization COVID-19 Conspiracy Theories To test our method's ability to identify specific narratives within a noisy social media environment, we use the Webimmunization COVID-19 Conspiracy Theories dataset. This large-scale collection of tweets is designed for the Natural Language Understanding task of determining whether a given tweet expresses support or not for a specific conspiracy theory related to the COVID-19 pandemic.

The dataset was constructed from an initial pool of over 40 million tweets collected via the Twitter API between November 2019 and December 2021. First we defined six COVID-19 conspiracy theories. To build the dataset, we employed a cosine similarity method to select tweets from the initial pool that were semantically similar to these six defined narratives. This process resulted in a candidate set of 60,000 tweets. After filtering out non-English content and tweets containing only mentions, links, or emojis, the final dataset consists of 53,734 tweets for analysis.

The annotation was performed by 1,151 unique US-based annotators recruited through the Prolific platform [4], with each tweet being evaluated by three different annotators. The task for the annotators was to determine if a tweet expressed support for one of the six pre-defined popular COVID-19 conspiracy theories (CTs) (Kunst et al., 2024):

- CT 1: **Deliberate strategy to create economic instability or benefit large corporations.** The coronavirus or the government's response to it is a deliberate strategy to create economic instability or to benefit large corporations over small businesses.
- CT 2: **Public was intentionally misled about the true nature of the virus and prevention.** The public is being intentionally misled about the true nature of the Coronavirus, its risks, or the efficacy of certain treatments or prevention methods.
- CT 3: **Human made and bioweapon.** The Coronavirus was created intentionally, made by humans, or as a bioweapons.
- CT 4: **Governments and politicians spread misinformation.** Politicians or government agencies are intentionally spreading false information, or they have some other motive for the way they are responding to the coronavirus.
- CT 5: **The Chinese intentionally spread the virus.** The Chinese government intentionally created or spread the coronavirus to harm other countries.
- CT 6: **Vaccines are unsafe.** The coronavirus vaccine is either unsafe or part of a larger plot to control people or reduce the population.

The dataset provides a binary label for each tweet and conspiracy theory pair, where *1* indicates that the tweet supports the conspiracy, and *0* indicates it does not.

This dataset is noted for being highly imbalanced and having rather low inter-annotator agreement, which makes it a uniquely challenging benchmark. The agreement, measured with Krippendorff's alpha (Krippendorff, 2018) and PABAK (G. Chen et al., 2009), varied significantly across the different conspiracy theories. For instance, agreement was highest for more concrete and explicit theories like "Human made and bioweapons" and "The Chinese intentionally spread the virus", while it was lower for more nuanced theories, that allowed for different interprtations.

[4] https://www.prolific.com

4.2.2 Setup

Our experimental design aims to provide a clear comparison between standard uniform sampling and our proposed VGL method across the four diverse datasets. For each dataset, we defined a suitable model architecture and conducted a rigorous set of experiments.

```python
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(1, 16, 5, 1)
        self.conv2 = nn.Conv2d(16, 32, 5, 1)
        self.fc1 = nn.Linear(512, 64)
        self.fc2 = nn.Linear(64, 10)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.max_pool2d(x, 2)
        x = F.relu(self.conv2(x))
        x = F.max_pool2d(x, 2)
        x = torch.flatten(x, 1)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

Listing 4.1: Convolutional Neural Network architecture used for experiments on MNIST dataset

```python
class MLP(nn.Module):
    """Simple MLP with bag-of-words representation"""
    def __init__(
            self,
            vocab_size,
            embed_dim=128,
            hidden_dim=64,
            num_classes=2,
            dropout=0.3
        ):
        super(MLP, self).__init__()
        self.embedding = nn.Embedding(vocab_size, embed_dim, padding_idx=0)
        self.fc1 = nn.Linear(embed_dim, hidden_dim)
        self.fc2 = nn.Linear(hidden_dim, num_classes)
        self.dropout = nn.Dropout(dropout)

    def forward(self, x):
        x = self.embedding(x)
        x = x.mean(dim=1)
        x = F.relu(self.fc1(x))
        x = self.dropout(x)
        x = self.fc2(x)
        return x
```

Listing 4.2: Multi Layer Perceptron architecture used for experiments on AG News dataset

To ensure a fair comparison, we selected model architectures appropriate for the complexity of each task:

- For MNIST, we used a simple Convolutional Neural Network (CNN) (Listing 4.2.2).
- For AG News, we employed a simple Multi-Layer Perceptron (MLP) (Listing 4.2.2).

Model	Learning rate	Optimizer	Weight decay	Batch size
CNN	1×10^{-3}	Adam	0.001	16
MLP	1×10^{-3}	Adam	-	32
RoBERTa base	5×10^{-5}	SGD	0.01	8

TABLE 4.1: The table lists the hyperparameters used for each model during the experiments.

Dataset	Max gradients	% of data per epoch	Gradients computation frequency
MNIST	all	10%	every epoch
AG News	all	50%	every epoch
Vaccine Attitudes	10	20%	every epoch
Conspiracy Theories	3	50%	every third epoch

TABLE 4.2: VGL configuration for scalability across datasets. The table details the specific configurations used to make the VGL method tractable across datasets of varying sizes. The settings were adapted to balance performance with computational and memory constraints. For the larger datasets, a more memory-efficient sliding window was used, estimating VoG scores from the last 10 gradients for Vaccine Attitudes and the last 3 gradients for Conspiracy Theories. At every epoch, a fraction of the dataset was used, which was empirically determined to be the best performing one for each case. The fractions were 10% for MNIST, 50% for AG News, 20% for Vaccine Attitudes, and 50% for Conspiracy Theories. The frequency of gradient computation was reduced for the largest dataset. For the Conspiracy Theories dataset, this was done every third epoch.

- For the more complex Webimmunization Vaccine Attitude and Conspiracy Theories datasets, we utilized RoBERTa base (Y. Liu et al., 2019), a pre-trained transformer model, with a classification head. We used a version trained on Twitter data [5] to work on text representations aligned to our domain.

For each of the four experimental setups, we performed a hyperparameter grid search to find an optimal configuration, over optimizers (SGD, Adam), learning rates (1×10^{-5}, 5×10^{-5}, 1×10^{-4}, 1×10^{-3}) and weight decays (0, 1×10^{-3}, 1×10^{-2}). Table 4.1 shows the chosen hyperparameters for each model. To ensure the robustness of our results, every experiment was repeated 10 times with different random seeds, and we report the mean and standard deviation of the evaluation metrics across these runs.

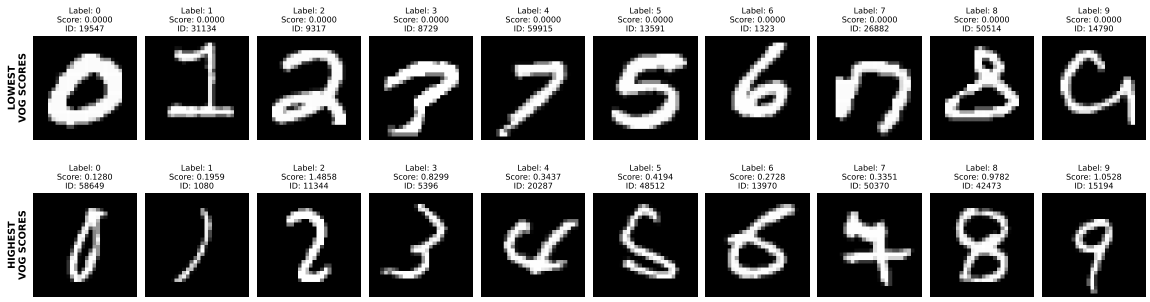
Each experiment consists of a direct comparison between two distinct training implementations, which differ only in their data sampling strategy at each epoch.

Baseline (uniform sampling) This procedure follows a standard training approach but uses only a fraction of the full training dataset during the epoch. At the beginning of each epoch, a subset of m unique examples is sampled uniformly at random from the total training dataset of size n. The model is then trained for one epoch on this subset, lasting for $m/|B|$ iterations, where $|B|$ is the batch size.

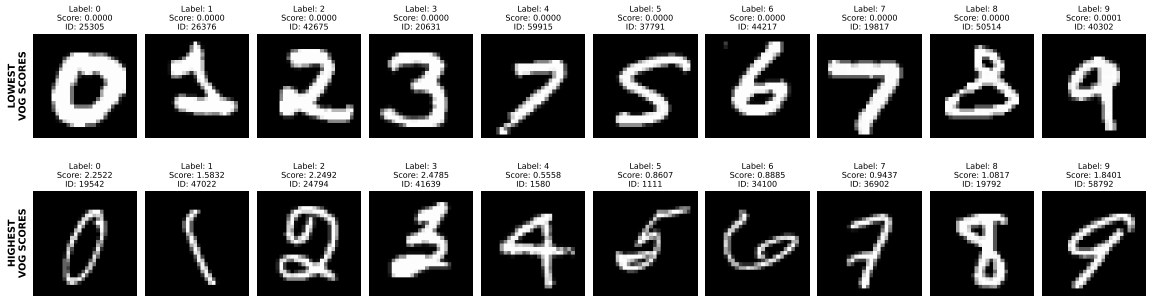
VoG Guided Learning This implementation incorporates our dynamic curriculum. At each epoch, a subset of m unique examples is drawn from the training set using a VoG-based probability distribution, where the probability of sampling each example is directly proportional to its VoG score. Samples with higher VoG scores, indicating greater uncertainty, are more likely to be selected for training.

The calculation of VoG scores is performed iteratively. After each training epoch, gradients are computed for all examples in the training set using the updated model checkpoint. These gradients are then used to update the VoG scores for the next epoch's sampling phase. As the VoG score requires at least two gradient snapshots to compute variance, the training procedure for both the baseline and our proposed method is identical for the first two epochs, with both using uniform

[5] https://huggingface.co/cardiffnlp/twitter-roberta-base



(a) Training examples with the lowest (first row) and highest (second row) VoG scores after epoch 5



(b) Training examples with the lowest (first row) and highest (second row) VoG scores after final epoch

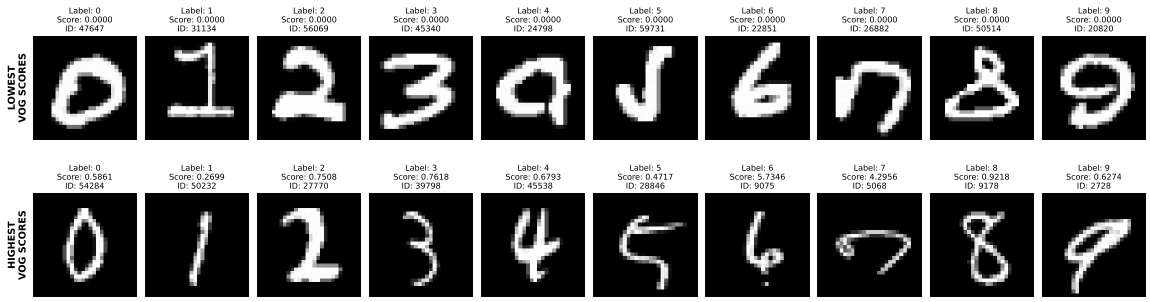
FIGURE 4.2: Training examples with the lowest and highest VoG scores sampled from epochs 5, and 20 during the standard training.

sampling. This allows us to collect the initial gradients required to start the VoG-based sampling from the third epoch onwards.

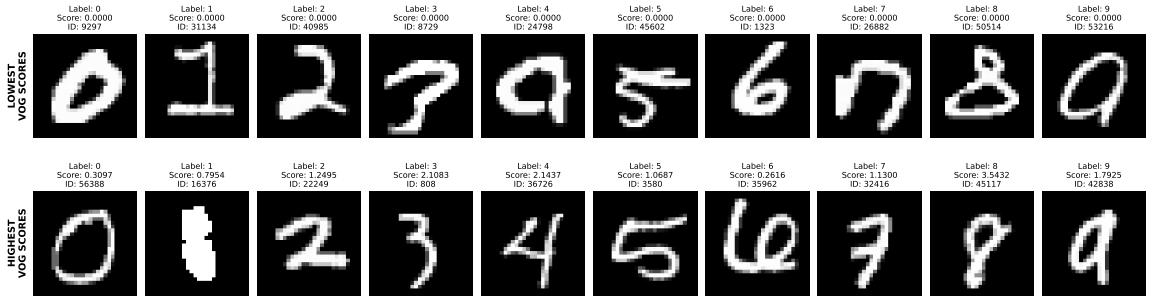
Recognizing that storing all historical gradients is not scalable for modern large models and datasets, we implemented two strategies for VoG computation. For smaller datasets, MNIST and AG News, the VoG score for each sample was calculated using the full history of gradients from all preceding epochs. For the larger datasets, Webimmunization Vaccine Attitude and Conspiracy Theories, we used a more memory-efficient sliding window approach. The VoG scores were estimated using only the gradients from the last $k = 10$ epochs for Vaccine Attiudes dataset and $k = 3$ epochs for Conspiracy Theories dataset. This makes the approach tractable while still capturing the recent dynamics of model uncertainty.

4.3 Results

In this section, we present the results of our experiments designed to evaluate the performance and behavior of our proposed VGL method compared to a standard uniform sampling baseline. Our analysis is structured in two parts to provide a comprehensive understanding of our method's impact. We begin with a qualitative deep dive into the training dynamics on the MNIST dataset, illustrating how VGL actively re-prioritizes examples and reshapes the learning process over time. Following this intuitive analysis, we present the main quantitative performance results across all four benchmark datasets, demonstrating VGL's effectiveness in improving final model performance and data efficiency.



(a) Training examples with the lowest (first row) and highest (second row) VoG scores after epoch 5



(b) Training examples with the lowest (first row) and highest (second row) VoG scores after final epoch

FIGURE 4.3: Training examples with the lowest and highest VoG scores sampled from epochs 5 and 20 during the VGL training.

4.3.1 Understanding VGL dynamics

We begin our analysis with a qualitative analysis on the MNIST dataset to build an intuition for how VGL alters the training process. We aim to visualize what the models trained with uniform sampling and VGL perceive as "easy" and "hard" and how this perception evolves. To do this, in Figures 4.2 and 4.3 we present a selection of training examples with high and low VoG scores sampled from the two different moments in training: the beginning of training and after the final epoch. For each training method, the figure displays ten examples with the lowest VoG scores (one per class, representing the "easiest" sample in the class) and ten examples with the highest final VoG scores (one per class, representing the "hardest" sample in the class).

An analysis of the results obtained from the final epoch for the model trained with uniform sampling reveals a predictable outcome: the difficulty scores are highly interpretable and align with human perception. The row of easiest examples consists mostly of clean, prototypical digits that are unambiguous. The only exception is the representativeness of class '4' that looks more like a '7'. Conversely, the row of the most difficult examples contains digits that sometimes might be hard for a human to decipher, featuring unusual strokes or ambiguous shapes. In contrast, the difficulty scores returned by the model trained with VGL after the final epoch are less immediately interpretable. While the "hardest" examples are still visually complex, the "easiest" examples are often not the clean, prototypical digits we might expect. For instance, the representatives of the classes 3, 4, 7 and 9 with the lowest VoG scores are ambiguous and hard to decipher. The representative of class 4 has a rounded shape that does not look like a typical '4' digit; similarly representative of class 7 has a shape that does not resemble any digit; and the representative of 9 looks more like a '0' digit. This counterintuitive result demonstrates the core mechanism of VGL. By persistently focusing training capacity on the most challenging examples throughout training,

VGL forces the model to master them. Consequently, examples that were once difficult for the model have become simple by the end of training, fundamentally reshaping the model's internal definition of difficulty.

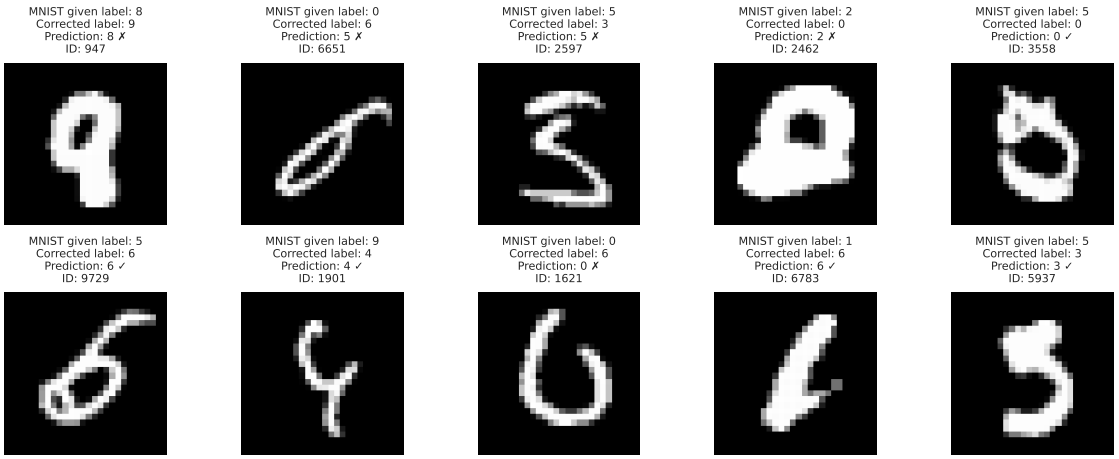


FIGURE 4.4: Pervasive label errors analysis. The predictions are obtained from one of the instances of models trained with the VGL method.

This observation, however, raises a critical question regarding generalization: does VGL's intense focus on atypical or difficult examples lead to overfitting on label noise, essentially causing the model to memorize flawed patterns? To investigate this possibility, we analyzed the models' robustness using known mislabeled examples from the labelerrors.com MNIST collection[6]. We ran predictions with both our final models (Figure 4.4) on these examples and measured accuracy against the human-provided, corrected labels. The results show that both the uniform sampling and VGL models achieve approximately 50% accuracy against the corrected labels. Furthermore, their predictions were almost identical, differing for only two examples (IDs: 1621 and 6783). This important finding suggests that while VGL dramatically alters the learning process, in this setting, it does not come at the cost of reduced robustness. For this task, VGL appears to be neutral with respect to label noise, neither introducing a vulnerability nor providing an improvement in resilience.

VoG ranking stability To quantitatively verify the observations from our visual analysis, we next investigated the stability of the sample difficulty rankings over time for both training methods. If VGL truly creates a more dynamic learning environment, decreasing the difficulty of the examples with initially highest VoG scores, we would expect the ranking of examples by their VoG score to change more significantly from one epoch to the next compared to the baseline. To measure this, we computed the VoG scores for all training examples at each epoch and then calculated Spearman's rank correlation coefficient ρ between the rankings of every pair of epochs. This coefficient measures the similarity of the ordinal ranking of samples, providing a measure of how much the model's perception of difficulty shifts during training.

Our analysis confirms our hypothesis. The mean correlation between epoch rankings for the model trained with uniform sampling was $\rho = 0.012$. In contrast, the VGL method yielded a lower mean correlation of $\rho = 0.008$. This lower value for VGL indicates a greater decorrelation between the difficulty rankings over time, quantitatively demonstrating that the set of "hardest" examples is being reshuffled much more actively. This supports our earlier conclusion: VGL does

[6] https://labelerrors.com

not simply focus on a static set of hard examples but dynamically alters the difficulty landscape by prioritizing challenging samples and, in the process of mastering them, effectively making them easier for the model in subsequent stages of training.

VoG mean score As the final piece of our qualitative analysis, we provide evidence that VGL not only reshuffles the difficulty ranking but actively reduces the overall difficulty of the dataset for the model over time. To do this, we tracked the mean VoG score, averaged across all training samples, at each epoch for both the baseline and VGL methods.

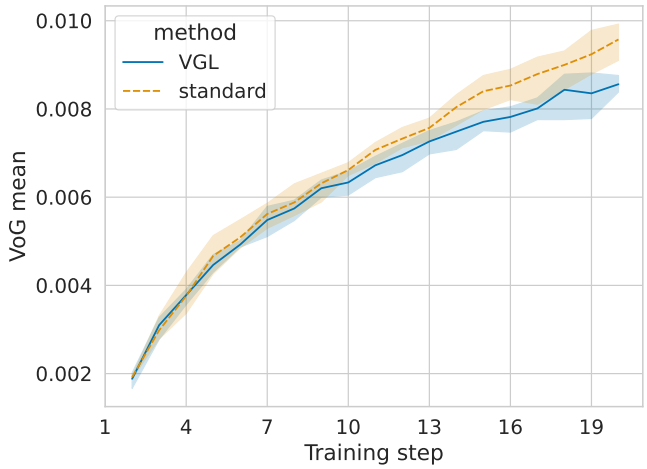


FIGURE 4.5: Evolution of the mean VoG score during training. This chart compares the mean VoG score for the VGL method against a standard training baseline over the course of training. The scores are averaged across all training samples at each training step. The results show that while the mean VoG score increases for both methods, VGL maintains a slightly lower mean VoG in the later stages, suggesting it reaches a more stable state.

The results, visualized in Figure 4.5, support our hypothesis. Throughout the training process, the mean VoG score was consistently lower for the VGL method compared to uniform sampling. Additionally, the difference between the mean scores becomes more pronounced as training progresses. This demonstrates that while the baseline model's ability to resolve difficult examples may slow down or plateau, VGL maintains its effectiveness, continually driving down the model's uncertainty across the dataset. By the final epoch, the divergence is clear: the model trained with VGL achieves a final mean VoG score of 0.00856, whereas the model trained with uniform sampling ends with a higher score of 0.00958. This confirms that VGL's strategy of prioritizing high-variance examples is not just a reallocation of focus, but a genuinely more efficient learning process that leads to a greater overall reduction in sample difficulty by the end of training.

4.3.2 Performance analysis

Having established how VGL alters training dynamics in the previous section, we now turn to the primary question: Does this approach lead to tangible improvements in model performance? In this section, we present the main quantitative results from our experiments, comparing the final test set accuracy of models trained with VGL against the uniform sampling baseline across all four datasets.

The final performance metrics, summarized in Table 4.3 show a consistent and positive improvement for the VGL method across every task. The most significant gain was observed on the MNIST dataset, where VGL achieved a relative error reduction of 14.02%. On the more complex text classification tasks, VGL also demonstrated clear benefits, yielding error reductions of 1.27%

TABLE 4.3: Performance comparison of VGL vs. baseline with statistical validation. This table summarizes the final average accuracy of the VGL method compared to the baseline across four datasets. All results are averaged over 10 independent runs.

Dataset	Baseline accuracy	VGL accuracy	Error reduction	t (p-value)
MNIST	$0.96654 \pm .0010$	$0.97123 \pm .0016$	14.02%	7.59 (5×10^{-7})
AG News	$0.64891 \pm .0039$	$0.65337 \pm .0036$	1.27%	2.53 (2×10^{-2})
Vaccine Attitude	$0.82124 \pm .0036$	$0.82876 \pm .0020$	4.20%	5.46 (3×10^{-5})
Conspiracy Theories	$0.72062 \pm .0020$	$0.72250 \pm .0012$	0.67%	2.44 (2×10^{-2})

on AG News, 4.20% on the Vaccine Attitude dataset, and 0.67% on the Conspiracy Theories dataset.

To validate these findings, we conducted a Student's t-test on the results from the 10 independent runs for each experiment. The improvements achieved by VGL were found to be statistically significant across all four datasets. The low p-values for MNIST (5×10^{-7}) and Vaccine Attitude (3×10^{-5}), along with still significant results for AG News (2×10^{-2}) and Conspiracy Theories (2×10^{-2}), confirm that the performance gains are not due to random chance.

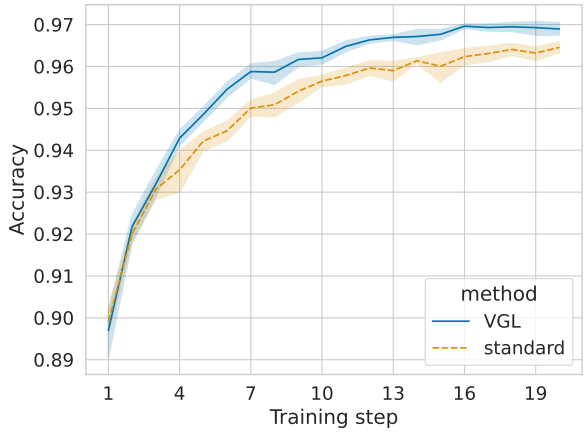


FIGURE 4.6: Test accuracy over training epochs on the MNIST dataset. The plot compares our VGL method against the uniform sampling baseline. Curves show the mean accuracy over 10 runs; shaded regions represent the standard deviation. The VGL method achieves a higher final accuracy and demonstrates faster convergence.

These aggregate performance gains are further illustrated by the learning curves shown in Figures 4.6, 4.7, 4.8, and 4.9, which plot the test set accuracy at every epoch for each dataset. In these figures, we observe that the VGL learning curve trends above the baseline curve. This demonstrates greater data efficiency, often reaching higher performance levels earlier in the training process than the uniform sampling baseline. These results provide strong quantitative evidence that the dynamic curriculum created by VGL effectively translates into improved final model performance across a variety of domains and task complexities.

4.4 Conclusions

This chapter introduced VoG-Guided Learning (VGL), a novel training methodology designed to improve the efficiency and performance of deep learning models. The preceding sections have provided an empirical evaluation of VGL, demonstrating its effectiveness across a diverse range of tasks and datasets. In these concluding remarks, we situate VGL within the broader curriculum

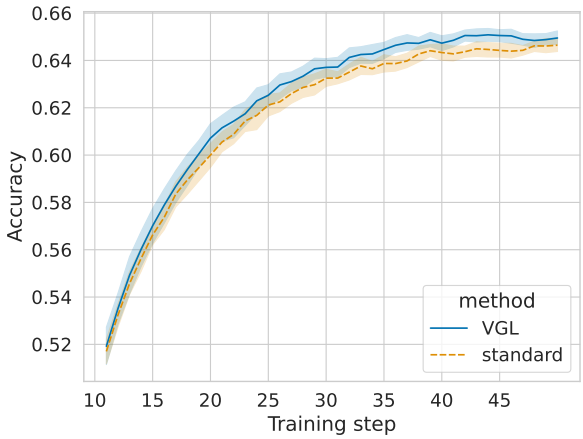


FIGURE 4.7: Test accuracy over training epochs on the AG News dataset. The plot compares our VGL method against the uniform sampling baseline. Curves show the mean accuracy over 10 runs; shaded regions represent the standard deviation. VGL consistently outperforms the baseline throughout the training process. The plot ignores the first 10 epochs for better final result visibility.

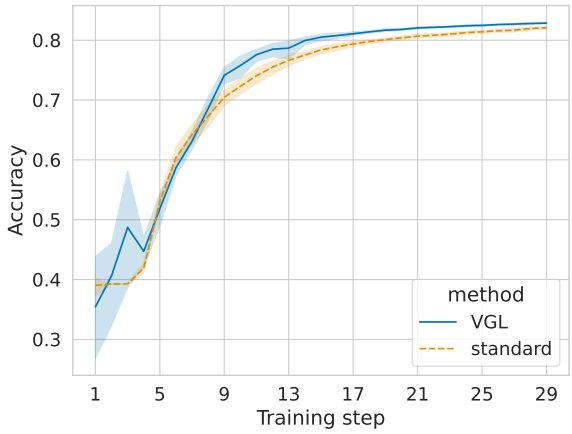


FIGURE 4.8: Test accuracy over training epochs on the Vaccine Attitude dataset. The plot compares our VGL method against the uniform sampling baseline. Curves show the mean accuracy over 10 runs; shaded regions represent the standard deviation. VGL shows a clear and statistically significant performance improvement over the baseline.

learning framework to clearly describe its unique characteristics, evaluate its performance, and outline promising avenues for future research.

VGL is a form of a data-driven curriculum, as it operates by resampling the training data rather than manipulating the learning task itself. It implements the core components of a CL system in the following way:

- **Scoring Function**: Unlike methods that rely on static or heuristic-based difficulty metrics, VGL employs a dynamic scoring function that adapts to the model state during training. The "difficulty" of each sample is measured by its Variance of Gradients (VoG). The metric is not a fixed property but re-evaluated at each epoch based on the current state of the model. The VoG scores are highly interconnected with the concept of uncertainty and ambiguity.

- **Pacing Function**: VGL utilizes an implicit and adaptive pacing function. There is no predefined schedule for the curriculum. Instead, the pacing occurs organically as the VoG scores evolve. As the model masters certain difficult examples, their VoG scores naturally decrease, causing them to be sampled less frequently and incorporate new examples into the

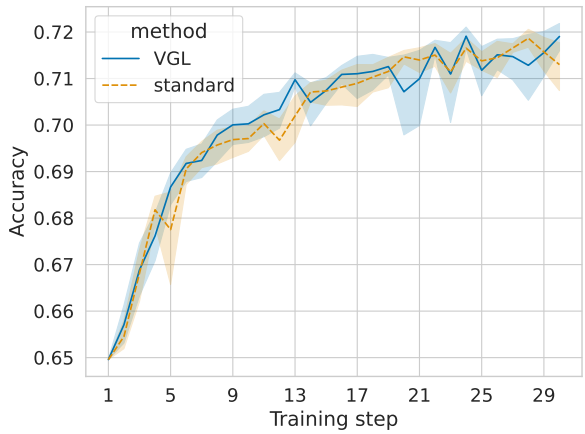


FIGURE 4.9: Test accuracy over training epochs on the Conspiracy Theories dataset. The plot compares our VGL method against the uniform sampling baseline. Curves show the mean accuracy over 10 runs; shaded regions represent the standard deviation. The VGL curve is consistently above the baseline for the first part of the training, indicating more efficient learning. However, it is getting unstable in the second part of the training, but still, the VGL checkpoints with the highest accuracies saved over each run outperform the baseline checkpoints with the highest accuracies saved over each run.

training. The curriculum's pace is therefore directly coupled to the model's own learning progress (self-paced learning).

The experimental results presented in this chapter demonstrate that VGL successfully fulfills several of the key promises of curriculum learning. The framework was shown to provide:

- **Performance improvement** VGL consistently outperformed the uniform sampling baseline across all four benchmark datasets, with statistically significant gains.
- **Convergence improvement** The learning curves revealed that VGL often converges faster, achieving higher performance with the same number of training iterations. This gain in efficiency stems from VGL's ability to dynamically identify and focus on choosing an effective subset of training examples at each epoch, a principle that resonates with recent findings on data-efficient training (Paul et al., 2021). Our approach specifically aligns with the work of Toneva et al., 2018, who demonstrated that "based on forgetting dynamics, a significant fraction of examples can be omitted from the training data set while still maintaining state-of-the-art generalization performance". Similarly, in VGL, we can omit a significant fraction of examples during each epoch — namely, the low-VoG, "easy" samples which are analogous to the "unforgettable" examples identified by Toneva et al., 2018 and not only maintain performance but actively improve convergence effectiveness by concentrating the model's capacity on the most informative, high-variance samples. Our qualitative analysis showed that VGL does not merely focus on a static set of hard examples but actively reduces their difficulty, leading to a lower aggregate VoG score (lower uncertainty) across the dataset by the end of training.
- **Robustness** We showed that the VGL maintain the same level of robustness as the baseline method.

This work has systematically explored a specific and powerful region of the curriculum learning design space: a data-driven curriculum with dynamic, model-dependent scoring and an implicit, adaptive pacing function. While we have demonstrated the efficacy of this approach, this opens

up several exciting avenues for future work. The most pressing challenge is scalability; computing gradients for the entire dataset with a large sliding window that seems to affect performance gains for each epoch is infeasible for very large-scale problems. Future research could explore more efficient approximations of the VoG score or methods for estimating it on a smaller subset of data. Furthermore, the sliding window used for our RoBERTa models was of a fixed size ($k = 10$); developing an adaptive window that adjusts its size based on the learning phase, perhaps wider at the beginning and narrower as the model converges, could yield further improvements. We also propose exploring dynamic sampling strategies. For example, to improve stability on challenging datasets such as Conspiracy Theories, where we observed some instability in the final training phase, one could implement a hybrid schedule that progressively moves from VGL's targeted sampling back towards uniform sampling. This would ensure the model revisits the entire data distribution after the most difficult examples have been mastered. The VGL method was also evaluated in training under large-batch regime in Chapter 5.

Chapter 5

Batch orchestration

This chapter transforms the philosophy behind the mini-batch construction from a simple computational necessity into a strategic tool for optimizing deep learning models training. We move beyond the traditional focus on batch size to conduct a holistic investigation into the interplay between three key levers: the size of the batch, the composition of samples within it, and the sequencing of batches throughout training. The central thesis of this chapter is that by intelligently manipulating these components, we can directly influence the dynamics of the optimization process, managing the trade-off between exploration and stability, mitigating the generalization gap, and ultimately achieving superior model performance.

The work presented in this chapter is based on our publication, "Curriculum Learning Revisited: Incremental Batch Learning with Instance Typicality Ranking," which was presented at the International Conference on Artificial Neural Networks, ICANN 2021 (Krysińska, Morzy, et al., 2021).

This chapter is structured to systematically explore the batch composition and scheduling problem. We begin by reviewing the related work to identify a critical Research Gap (Section 5.1). Section 5.2 details our two suites of experiments. The first one explores inner-epoch batch orchestration with a static, typicality-based curriculum. The second one explores inter-epoch batch orchestration with a dynamic, VoG-based curriculum. In Section 5.3, we present a comprehensive analysis of the results, and finally, Section 5.4 offers our conclusions, synthesizing the findings and outlining future directions for batch orchestration research.

5.1 Introduction

The remarkable success of modern deep learning is linked to the development of effective optimization algorithms capable of navigating the high-dimensional, non-convex loss landscapes characteristic of deep neural networks. At the heart of training wide range of models lies a family of iterative optimization methods collectively known as stochastic gradient descent (SGD) based methods.

The fundamental form of the SGD-based model update was defined in Equation 2.1. Let us use simplified notation:

$$G_t = \nabla_\theta J_t \tag{5.1}$$

The updates are performed until convergence to a local minimum. This iterative process has proven to be the cornerstone of contemporary machine learning, enabling the training of models with billions of parameters on massive datasets.

The computational efficiency of any SGD-based method hinges on the strategy used to compute the stochastic gradient. This choice presents a fundamental trilemma, forcing a trade-off between computational efficiency, memory requirements, and the quality of the gradient estimate. The three primary paradigms are:

1. **Gradient Descent**: At one extreme, the gradient is computed using the loss from each sample from training dataset D; $G = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \ell(f_\theta(\mathbf{x_i}), y_i)$. This yields the true gradient of the loss function, resulting in a stable and direct path towards a local minimum. However, for the large-scale datasets common in deep learning, this approach is computationally prohibitive, as it requires loading and processing the entire dataset for a single parameter update, exceeding the memory and processing capacity of modern hardware.

2. **SGD**: At the opposite extreme, the gradient is estimated using just a single, randomly selected training example per step (a batch size of $b = 1$); $G = \nabla_\theta \ell(f_\theta(\mathbf{x_i}), y_i)$. This method is exceptionally memory-efficient. However, this approach is computationally inefficient as it is not well-suited for the parallel processing capabilities of modern (Graphics Processing Units) GPUs. The high overhead from frequent, individual updates means that processing the entire dataset can take longer than the previous method.

3. **Mini-batch SGD**: The de facto standard in deep learning, mini-batch SGD, represents a pragmatic compromise. In this approach, the gradient is estimated by averaging the gradients over a small subset, or "mini-batch," of b training examples $G = \frac{1}{b}\sum_{i=1}^{b} \nabla_\theta \ell(f_\theta(\mathbf{x_i}), y_i)$. Mini-batch SGD characterizes the strengths of the other two methods: it remains computationally and memory-efficient for large-scale problems. The primary driver of its computational efficiency is its ability to leverage the immense parallelism of GPUs optimized for matrix operations, making the processing of a batch of dozens or hundreds of examples nearly as fast as processing a single one. This dramatically accelerates training.

The hardware limitations (memory and computation throughput) are less of our concern in this chapter. Although important, we want to shift our focus to effect on the convergence rate and final model performance. The characteristic that differentiates all three SGD-based methods is batch size: $|D|$, 1, and b for GD, SGD and mini-batch SGD, respectively. The size of batches is greatly connected to the variance of gradient estimations. Gradient variance refers to how much the gradients $\mathbf{G} = \{G_1, G_2, \dots G_k\}$ computed from subsequent batches vary from each other and from the true gradient of the full dataset.

$$Var(\mathbf{G}) = \mathbb{E}\big[\mathbf{G}^2\big] - \mathbb{E}\big[\mathbf{G}\big]^2 \tag{5.2}$$

It is a measure of the noise introduced by the sampling process. The most basic property of a batch that influences this variance is its size, with the variance of the gradient estimate being inversely proportional to the number of samples in the batch. $Var(\mathbf{G}) \propto 1/b$. Doubling the batch size, for instance, halves the variance. This simple relationship creates a complex optimization challenge and the need to find a "sweet spot". A batch must be small enough to retain sufficient gradient variance, a form of beneficial noise that helps the optimizer explore the loss landscape and escape sharp local minima, yet large enough to provide a stable learning signal for efficient convergence and to leverage computational resources effectively (Keskar et al., 2016; Masters & Luschi, 2018; McCandlish et al., 2018; Summers & Dinneen, 2020).

5.1.1 Related work

Generalization gap. Therefore, our intuition might suggest that maximizing the batch size, constrained only by hardware limitations, is the optimal strategy. This approach would seemingly lead to more accurate gradient estimates, smoother convergence paths, and faster wall-clock time. However, a significant body of research (Keskar et al., 2016; Masters & Luschi, 2018; Summers & Dinneen, 2020) challenges this intuition, revealing a critical link between batch size, the geometry of the loss landscape, and the generalization performance of the final model. The leading hypothesis to explain this gap relates to the geometry of the loss landscape. The training process for a deep network can be visualized as navigating a high-dimensional landscape of loss values, seeking its lowest points (minima). This landscape is complex and non-convex, containing many such minima. These minima can be characterized by their "sharpness" or "flatness." A sharp minimum is a narrow valley, where a small perturbation in the model's weights leads to a large increase in the loss. A flat minimum is a wide basin, where the loss remains low even with small changes to the weights. Models trained with large batches tend to converge to sharp minima, which are characterized by high curvature and often exhibit poor generalization to unseen data. Conversely, the higher gradient noise inherent in smaller batches can act as a form of implicit regularization (Hoffer et al., 2017; Qian & Klabjan, 2020). This noise helps the optimization process escape the basins of attraction of these sharp minima, guiding it towards flat minima. These flat regions are associated with better generalization performance, as the model's predictions are less sensitive to small perturbations in the parameters. This phenomenon has led to the concept of a critical batch size. Below this threshold, the stochastic noise is sufficient to facilitate exploration and drive the iterates toward flatter, more generalizable solutions. Above it, the gradient signal dominates, the noise becomes insufficient to escape the initial basin of attraction, and the optimizer is more likely to settle in a sharp, poorly generalizing minimum, often resulting in a noticeable drop in test accuracy.

Loss landscape exploration. The exploratory behavior of the optimizer is also profoundly influenced by the batch size. Empirical and theoretical results indicate that large-batch methods, with their low-variance gradients, tend to descend into the minima located within the basin of attraction of the initial parameters (Keskar et al., 2016). They are effective at exploiting the local landscape. In contrast, the high variance of small-batch gradients can drive the iterates across the loss surface, enabling them to traverse greater distances in the parameter space and explore minima that are far from the initialization point. This suggests a fundamental trade-off in the early phases of training: one must choose between the rapid convergence towards a nearby solution offered by large batches and the broad exploration of the loss landscape facilitated by the stochasticity of small batches. Prioritizing this exploration early on appears crucial for locating wider, more robust minima, at the cost of a less direct convergence path.

Vary, dynamic and adaptive batch sizes. To harness the benefits of both small and large batch training, researchers have explored strategies that dynamically vary the batch size during optimization (Byrd et al., 2012; Friedlander & Schmidt, 2012; McCandlish et al., 2018). One effective approach is batch size scheduling, often referred to as "warm-starting" or "piggybacking" (Keskar et al., 2016). In these methods, training starts with small batch sizes to leverage the beneficial exploratory effects of gradient noise in the early stages. As training progresses and the optimizer nears a promising basin of attraction, the batch size is increased. This allows for a more stable and rapid convergence in the later stages, effectively combining the exploration of small

batches with the computational efficiency of large ones.

A more sophisticated paradigm involves adaptive batch size methods. The insights from the previous paragraphs suggest that a single, fixed batch size is inherently suboptimal, as the ideal balance between exploration and exploitation changes throughout training. At the core of many adaptive strategies is the goal of maintaining a consistent level of gradient noise. This is quantified by the gradient noise scale $\mathcal{B}$, which measures the ratio of gradient variance to gradient magnitude. A simplified formulation defines it as the ratio of the trace of the gradient covariance matrix Σ, to the squared norm of the true gradient G:

$$\mathcal{B} = \frac{\mathrm{tr}(\Sigma)}{|G|^2} \tag{5.3}$$

A high value of $\mathcal{B}$ indicates that the learning process is in a noisy regime, where the effects of stochasticity, rather than the true gradient signal, dominate the training dynamics. Crucially, Keskar et al., 2016 shows that the noise scale is not static. It tends to increase as the model converges and the loss decreases, and it is typically larger for more difficult learning problems and complex datasets where gradients from different mini-batches vary more significantly. The predictable fluctuation of the noise scale during training provides a strong rationale for adaptive approaches. These techniques adjust the batch size on the fly based on training dynamics. For instance, adapting the batch size to maintain a target gradient noise scale has shown considerable promise, potentially reducing the total number of optimization steps required for convergence. These methods seek to automate the search for an optimal batch size, removing it as a critical hyperparameter that must be manually tuned.

Non-uniform sampling startegies. The standard mini-batch SGD algorithm, and the variance reduction analysis above, rests on an assumption that mini-batches are constructed by drawing samples i.i.d. from the training data. As we described in Chapter 4 this random sampling strategy is understood to be suboptimal. It treats all data points as equally informative and ignores the structure within the dataset. Consequently, random sampling can lead to inefficiencies, such as creating batches with highly redundant information or batches that are unrepresentative of the true data distribution, a problem that is particularly relevant in settings with significant class imbalance (Mao et al., 2022). This realization has catalyzed creation of various batch composition strategies which posit that the construction and sequencing of mini-batches are not implementation details but first-class levers for optimizing the training process. A growing body of research demonstrates that moving beyond random sampling can yield substantial improvements in model performance, convergence speed, and generalization. These efforts can be broadly categorized along two axes:

1. **Batch composition.** These methods focus on intelligently selecting the contents of each individual mini-batch to improve the quality of the gradient signal. One of the the composition methods is Diversity Sampling which addresses the issue of redundancy within a batch. For instance, methods based on Determinantal Point Processes (DPPs) (C. Zhang et al., 2017) sample diverse sets of data points by modeling repulsive interactions, ensuring that each batch provides a broader view of the data manifold. This has been shown to further reduce gradient variance and lead to more interpretable features, especially on imbalanced datasets. More recent proposals, such as the Batch Transformer (Her et al., 2024), even employ attention mechanisms within a batch to weigh the contributions of different samples, explicitly aiming to extract more trustworthy information and mitigate overfitting to noisy examples.

2. **Batch sequencing.** These methods focus on structuring the order in which batches are presented to the model over the course of training. Even simple heuristics can have a significant impact. In Natural Language Processing, for example, sorting the corpus by sentence length before forming batches is a common strategy to reduce computational waste from padding, but its effect on final model quality is complex and interacts with the choice of optimizer (Amodei et al., 2016; Kocmi & Bojar, 2017). For imbalanced classification tasks, structured shuffling strategies that explicitly balance classes within mini-batches consistently outperform random shuffling, ensuring that minority classes receive adequate attention throughout training. The interaction between batch sequencing and other training components can be surprisingly complex and lead to unexpected instabilities. For example, D. X. Wu et al., 2023 have shown that the interplay between the sample ordering strategy and Batch Normalization (Ioffe & Szegedy, 2015) is critical. It has been demonstrated that combining Single Shuffle (shuffling the dataset only once per epoch) with Batch Normalization can lead to training instability, slower convergence, and even divergence. This issue is not observed when using Random Reshuffle (where data is reshuffled before every epoch or batch). The root cause is that Batch Normalization is not permutation-invariant; its per-batch statistics depend on the specific samples within that batch. A fixed shuffling order causes the model to consistently optimize a "distorted risk", while random reshuffling averages out these distortions, leading to more stable training. The study of data ordering has become a central research question in the era of Large Language Models, where the sequence of training data has been shown to have a measurable and significant effect on model memorization and generalization capabilities (Kim & Lee, 2024; Yang et al., 2025).

5.1.2 Research gap

The insights from batch curation naturally lead to a more holistic view of the training process, elevating the focus from individual batches to the entire sequence of data presented to the optimizer. This perspective aligns with the curriculum learning paradigm. The existing landscape of intelligent batching strategies tend to be siloed, focusing either on the fine-grained composition of a single batch (e.g., importance or diversity sampling) or on the high-level sequencing samples or pre-defined data shards (e.g., traditional curriculum learning strategies or reinforcement learning task scheduling). There is a gap in the literature: a lack of studied methods that simultaneously address both the fine-grained composition of batches from individual data points and the strategic sequencing of these constructed batches.

In this chapter, we conduct a suite of experiments designed to bridge this gap and shed light on batch orchestration by studying the interplay between batch composition and sequencing. We investigate how composing batches with varying sizes interacts with sequencing strategies based on difficulty scoring from curriculum learning: typicality scoring and VoG scoring described in the previous chapters.

Our goal is not to propose a single optimal orchestration algorithm, but rather to provide an empirical analysis. By systematically evaluating these combined strategies across various tasks, including tabular data classification, image classification and text classification, we aim to identify key principles and trade-offs. The results of these experiments will offer guidance on how to co-design composition and sequencing methods, paving the way for more sophisticated and adaptive batch orchestration frameworks in the future.

5.2 Experiments

5.2.1 Intra-epoch batch orchestration with typicality sampling

In this section, we describe our first suite of experiments, designed to systematically investigate the interplay between batch composition and sequencing using a data-driven curriculum with a typicality-based scoring function. Our goal is to understand how deliberate, structured batch composition affects model training.

The methods proposed in this experiment suite can be framed as a practical implementation of a continuation method for non-convex optimization. Continuation methods, in principle, solve a difficult optimization problem by first addressing a smoothed or simplified version of it and then gradually deforming this simpler problem into the target one. This process creates a path in the parameter space that is intended to guide the optimizer towards a more robust and potentially better local minimum of the final, complex objective function. We can establish the analogy by mapping the coponents of the methods onto the formal structure of a continuation method. A continuation method involves a sequence of objective functions, $\{J_0, J_1, \dots J_T\}$ where J_0 is the easiest to optimize and J_T is the true target objective (Equation 2.7).

For the experiments detailed here, the entire training dataset is first pre-sorted according to a typicality score (degree-adjusted entropy centrality described in Section 3.1.2), creating a static curriculum that ranks examples from the most typical to the most atypical. This fixed, ordered dataset serves as the foundation for four distinct curriculum learning strategies, each designed to test a specific hypothesis about batch orchestration.

1. **Single Shuffle** The baseline strategy without curriculum. This is the most common strategy used in practice. It involves creating a single, fixed random permutation of the training dataset at the very beginning of training. The model then iterates through this same fixed sequence of non-overlapping mini-batches in every subsequent epoch. The name "Single Shuffle" is borrowed from D. X. Wu et al., 2023.

2. **Typicality-ordered SGD**: As a high-variance training variant, we train a model using SGD with a batch size of one. The model processes the samples sequentially one by one, strictly following the pre-defined typicality order from the most to the least typical sample in each epoch.

3. **Typicality-ordered mini-batch SGD**: The typicality-ordered dataset is partitioned into non-overlapping, sequential mini-batches of a fixed size b. The model processes these batches in order. This strategy intentionally creates batches where samples are highly homogeneous in terms of their typicality level (the first batch contains only the most typical examples, the next batch contains slightly less typical examples, and so on; the last batch contains the most atypical examples).

4. **Batch repetition**: This method aims to minimize the negative impact of atypical examples by forcing the model to master the patterns from typical instances first. The training set is divided into fixed-size batches exactly as in the mini-batch SGD. However, the training process involves repeating each batch j times before moving to the next. For example, the first batch (containing the most typical examples) is used for j consecutive updates, then the second batch is used for j updates, and so on. This repetition effectively forces the model to converge more deeply towards the parameter space defined by the typical instances before being exposed to the potentially disruptive gradients from more atypical data. The

ordering of instances is preserved between epochs. We present a batch repetition strategy as a continuation method (Allgower & Georg, 2012). A continuation method simplifies the problem at first and then gradually increases the complexity. In our proposed method, the training begins by repeatedly optimizing on the first batch, which, by design, contains the most typical examples. The act of performing j consecutive updates on this single batch constrains the optimization to a highly simplified objective function. This initial objective, defined by the low-variance gradients from typical data, can be considered the "smoothed" function, J_0. It presents a stable and simplified view of the loss landscape, allowing the model to find a strong initial basin of attraction corresponding to the core patterns in the data. The transition from one batch (B_t) to the next (B_{t+1}) represents a discrete step in the continuation schedule. When the training process moves from the first batch to the second, the underlying objective function being optimized is implicitly changed. The data distribution shifts to include slightly more atypical examples, thus introducing more complexity and variance into the loss landscape. By repeating this new batch j times, the model is given sufficient iterations to converge to a local minimum of this new intermediate objective, before the complexity is increased again. Therefore, batch repetition emulates a continuation method by decomposing the global optimization problem into a structured sequence of simpler, more focused sub-problems. By forcing convergence on progressively more atypical data, it guides the model parameters through a series of stable intermediate solutions, mitigating the disruptive influence of noisy examples and increasing the likelihood of settling in a high-quality minimum of the final objective function.

5. **Increasing batch size**: Inspired by the concept of spaced repetition in human learning Dempster, 1989, this strategy dynamically grows the batch size within an epoch to progressively incorporate less typical data. The process is as follows:

   - The first batch consists of the b most typical instances.
   - Each subsequent batch includes all the instances from the previous batch plus the next m unseen instances from the typicality-ordered curriculum.
   - The batch sizes therefore grow sequentially: b, $b+m$, $b+2m$, $\ldots$, n until the final batch encompasses all n training instances. The ordering of instances is preserved between epochs.

   The strategy can also be rigorously interpreted as a continuation method. This approach actualizes the core principle of transitioning from a simplified to a complex optimization problem by systematically expanding the data distribution considered at each step, thereby creating a smooth path for the optimizer to follow. Training starts with a small batch of size b containing only the most typical instances. The empirical loss computed over this small, homogenous subset represents the initial, highly simplified objective function, J_0. Similarly to batch repetition strategy, by restricting the optimizer's view to these core examples, this objective has minimal gradient variance and isolates the most fundamental patterns in the data, providing a stable starting point for optimization. Each new batch is formed by augmenting the previous batch with m new, progressively less typical instances. This process creates a sequence of nested datasets. Consequently, the objective function at step t, J_t, which is defined over a batch of size $b + tm$ is a direct and gentle evolution of the objective from the previous step. This gradual incorporation of new data provides a smooth deformation of the loss landscape. Unlike methods that switch between disjoint sets of data, this cumulative approach helps optimizer in building continuous optimization path, as each

subsequent objective function is a subtle perturbation of the one before it. The batch size itself acts as the continuation parameter, controlling the complexity and diversity of the data influencing the gradients. The epoch finishes with a final batch that encompasses all n training instances. The objective function associated with this final step, J_T, is precisely the target empirical loss over the entire dataset (true gradient). The model arrives at this final, most complex optimization problem not from a random state, but after being guided through a series of intermediate stages of increasing difficulty.

This experimental design allows us to empirically evaluate four core questions:

RQ1 First, we want to assess the effect of ordered, high-variance training by evaluating the final performance of the ordered SGD strategy, which combines a minimal batch size with a typicality-based data sequence. We will compare ordered SGD with the baseline, standard mini-batch SGD with uniform sampling. We will assess the method against the number of processed samples, which means that in one epoch ordered SGD will perform more model updates than mini-batch SGD. In this comparison, we are not interested in method computational efficiency but rather in the final performance of the model after convergence.

RQ2 Second, we investigate how batch composition based on typicality affects model performance. Our homogeneous mini-batch strategy, which groups similar examples together, stands in direct opposition to diversity sampling methods like Determinantal Point Processes(DPPs) (C. Zhang et al., 2017), which aim to diversify batches to reduce gradient variance. We hypothesize that our approach, by design, will instead increase gradient variance, potentially promoting better exploration of the loss landscape. This provides a lens through which to re-examine existing findings. While the DPP paper notes that its performance benefits are greatest with smaller batches and highly class-imbalanced datasets, this effect is not explicitly attributed to gradient variance. We suspect their variance-reducing method requires the additional noise from small batches to effectively escape poor local minima, which explains the improved performance in that specific regime reported in their paper. The method might be a good choice under a tight computational budget, but it might be worse than the i.i.d batch sampling under a tight wall-clock time budget due to poor parallelization of computations.

RQ3 Third, we want to test the batch repetition strategy against the baseline mini-batch SGD. We hypothesize that allowing the model to converge on a smaller dataset consisting typical examples helps to find a better local minima and improves generalization. The variance of gradients will fluctuate during the training. In episodes in which a single batch is repeating variance will be stable and low, but it will increase significantly, after j repetitions, when new batch is introduced. We suspect that the moments with increased variance help the model escape poor local minima.

RQ4 Finally, with the increasing batch size strategy, we want to check if we can downplay the effect of atypical examples by combining them with a growing set of more representative samples when computing the gradient. The gradient signal from atypical examples should be diminished by the more consistent signal from typical ones. On one hand, this strategy maintains high gradient variance with small batches of typical examples early on; on the other, it decreases variance with large batches containing both typical and atypical examples later. This approach is similar to warm-starting, described earlier, where we prioritize loss

landscape exploration in early stages of training, and as the training progresses we allow for more stable, local gradient updates. The central question is: does this strategy allow the model to escape poor minima early on but at the same time keep the training smooth enough to converge faster than the baseline and shrink the generalization gap?

The research questions **RQ3** and **RQ4** are grounded in theory on continuation methods and implicit regularization of gradient variance, however they stand in direct opposition to the findings of the Ash and Adams, 2020. Ash and Adams, 2020 conclude that warm-starting, continuing to train an already converged model on new data, can be detrimental because the model gets stuck in a sharp local minimum and fails to find a better, more generalizable solution. Their proposed fix is to explicitly "shrink and perturb" the model's weights to force it out of this poor minimum. Our hypotheses for **RQ3** and **RQ4** challenge this conclusion. Instead of requiring an external regularizer we propose that curriculum can inherently guide the optimizer towards a better solution. For **RQ3**, the strategy is a series of micro warm-starts. The model converges on one small, typical batch, then is warm-started on the next, slightly more atypical batch. According to Ash and Adams, 2020, this should be problematic, as the model could get trapped in a sequence of narrow minima specific to each batch. We hypothesize the opposite: that the periodic "shock" of introducing a new batch, which sharply increases gradient variance, acts as an implicit regularizer, naturally guiding the model out of poor minima and allowing it to find a superior final solution. For **RQ4**, we propose a smoother form of warm-starting. The model begins with typical data and gradually incorporates more complex, atypical data. Ash and Adams, 2020 suggest this gradual continuation is insufficient to escape the initial basin of attraction. Our hypothesis, however, is that this smooth transition of the objective function creates a continuous optimization path. The high variance from small batches at the beginning of each training provides the necessary exploration, while the growing batch size later provides stability. This method is designed to find a good minimum without the need for an explicit "shrink and perturb" step, directly questioning the idea that a simple continuation of training is inherently flawed.

Experiments setup The experiments were conducted on the same diverse collection of nine tabular datasets from the UCI Machine Learning Repository as those used in our analysis of typicality scoring. These datasets were chosen to represent a variety of sizes, attribute counts, and data types, including categorical, numerical, and mixed-type data.

To isolate the specific effects of the batch orchestration strategies, we employed a consistent experimental setup across all conditions. A simple feed-forward neural network was trained using the Adam optimizer with a fixed learning rate of 0.001 for 50 epochs. The only variable between the different experimental runs was the strategy used to compose and sequence the training batches.

For a comprehensive description of the individual dataset statistics, model architecture, and detailed training protocol, please refer to the Section 3.2.

5.2.2 Inter-epoch batch orchestration with VoG sampling

Having explored batching strategies within a static curriculum defined by typicality, we now turn our attention to a more adaptive paradigm. In this second suite of experiments, we investigate VGL-based batch orchestration, where the data ordering is not fixed but is dynamically guided by the model's own state during training. Instead of relying on a pre-computed metric, we leverage the Variance of Gradients (VoG) score from our VoG-Guided Learning framework, described in Chapter 4 to sequence the training data.

Our primary objective is to study the interplay between this dynamic, uncertainty-driven sample ordering and the choice of batch size, with a particular focus on the large-batch training regime. While large-batch training is known to accelerate wall-clock time, it often suffers from a "generalization gap," leading to models that converge to sharp, poorly-generalizing minima. We hypothesize that an intelligent data ordering, guided by real-time VoG scores, can mitigate these adverse effects. By composing large batches with an informative mix of high-variance samples, we aim to improve the quality of the gradient signal, encouraging exploration and guiding the optimizer towards flatter, more robust solutions, even at large batch sizes. The following experiments are designed to test this hypothesis by evaluating different VGL-based ordering strategies under various batch size configurations.

To systematically investigate the interplay between data ordering and batch size, we will evaluate different sampling strategies under three distinct and progressively complex batch size orchestration schedules. These schedules define how the batch size, b_e, changes as a function of the current epoch, e.

1. **Fixed large batch regime**: This schedule serves as our primary baseline for studying the generalization gap. The batch size is fixed to a single, large value, for the entire duration of training. This represents the common approach of maximizing hardware utilization but often leads to challenges in generalization.

2. **Linear batch size increase**: As previous research concluded that fixed large batch regime often leads to poor generalization we want to test increasing batch size schedule. This schedule implements the "warm-starting" or "gradual increase" strategy, designed to capture the exploratory benefits of small batches early in training and the stability of large batches later on. The batch size starts at a small value, b_min, and increases linearly with each epoch until it reaches a predefined maximum, b_{max}, at the final epoch, E.

$$b_e = b_{min} + \frac{e}{E}(b_{max} - b_{min}) \tag{5.4}$$

   This schedule should ensure loss landscape exploration at the beginning of training and at the same time benefits from larger batch sizes later in training.

3. **Cyclical exponential batch size increase**: Previous schedule is liable to be trapped in poor local minima at the end of training. Therefore, this schedule introduces a periodic "shock" to the training process, designed to help the optimizer escape poor local minima. The batch size grows exponentially from a minimum, b_{min}, to a maximum, b_{max}, over a fixed period of E epochs. After each period, the batch size abruptly resets to b_{min} and the cycle begins again. The batch size for a given epoch e is determined by its position within the current cycle, $e' = e \bmod E$.

$$b_e = b_{min}\left(\frac{b_{max}}{b_{min}}\right)^{\frac{e'}{E}} \tag{5.5}$$

We propose testing the above batch orchestration regimes with the following sampling methods:

1. **Single Shuffle**: The same strategy as described in the first suite of experiments. The strategy involves shuffling the training data only once at the very beginning. The model then trains on this same fixed order of data in every following epoch.

2. **Random Reshuffle**: Random Reshuffle is a name of strategy proposed by D. X. Wu et al., 2023 as an alternative to Single Shuffle. At the beginning of each epoch, the entire training dataset is randomly shuffled to create a new, unique permutation. The model then iterates

through this newly shuffled sequence in non-overlapping mini-batches. This ensures the model sees the data in a different random order every epoch, which is the standard approach for preventing cyclic learning dynamics.

3. **VoG-ordered mini-batch SGD**: This method applies a dynamic, VoG-based curriculum to the batching process. After each training epoch, the entire dataset is re-sorted based on the newly computed VoG scores of each sample, from the lowest score (easiest) to the highest (hardest). This VoG-ordered dataset is then partitioned into non-overlapping, sequential mini-batches of a fixed size, b. The model processes these batches in order during the next epoch. This strategy intentionally creates batches where samples are highly homogeneous in terms of their VoG scores. The data order is re-evaluated and changes after every single epoch.

4. **VGL**: This is our proposed method, as described in the Chapter 4. VGL performs weighted sampling with replacement to form each mini-batch. After an epoch, the VoG score for every sample is used to create a probability distribution. For each step in the next epoch, a mini-batch is constructed by drawing b samples from the entire dataset according to this VoG-weighted distribution. Because the sampling is done with replacement, a high-VoG sample may appear in multiple different batches within a single epoch, while a low-VoG sample may not be selected at all.

With our data sampling strategies and batch size schedules defined, our experiments aim to answer one central question: *How does the effectiveness of different data ordering strategies change when subjected to different batch size orchestration schedules?* We investigate this through three distinct experimental setups, each designed to probe a specific aspect of this interplay.

RQ1 Which data ordering strategies are most effective at mitigating the generalization gap commonly associated with large-batch training? As we stated before, the prevailing theory is that the low gradient variance from large batches is insufficient to help the optimizer escape these sharp basins of attraction. Our experiment challenges the notion that the only solution is algorithmic (e.g., sharpness-aware optimizers) or simply using smaller batches. We hypothesize that the problem can also be addressed at the data level. We posit that a strategically composed large batch can re-introduce beneficial noise, not by shrinking its size, but by curating its contents. Specifically, we hypothesize that both our VoG-ordered mini-batch SGD (which creates high variance between batches) and our VGL method (which prioritizes high-variance examples within batches) can increase the overall gradient noise scale, even with a large batch size. This acts as a data-driven implicit regularizer, forcing the optimizer towards flatter, more robust solutions and thereby shrinking the generalization gap.

RQ2 Can we close the generalization gap by combining VGL with an increasing batch size schedule? While the idea of increasing batch size is established, and is often used as a "warm-start" in real-world applications, the interaction with structured data ordering is not well understood. We hypothesize that a curriculum-aware strategy like VGL will be synergistic with this schedule. VGL will ensure that the initial, small-batch exploratory phase is maximally effective by focusing the limited batch capacity on the most uncertain examples. This "intelligent exploration" should allow the model to find a superior, wider basin of attraction early on, which is then more effectively exploited in the later, large-batch phase, leading to both better generalization and faster wall-clock convergence.

RQ3 Which data ordering strategies best leverage the periodic "shock" of the batch size reset to escape poor local minima and achieve superior final performance? We hypothesize that this periodic "shock" will destabilize the optimizer just enough to escape suboptimal local minima that it may have settled into during the previous cycle's large-batch phase. This approach stands in direct contrast to works like Ash and Adams, 2020, which suggest that such "warm-starting" is harmful and requires explicit weight perturbations to work. We posit the opposite: that a well-designed data curriculum, particularly when orchestrated by VGL, can harness the increased variance from the batch size reset to naturally guide the optimizer to re-explore the landscape, ultimately finding a more robust and generalizable solution without any external intervention.

Experiments setup To investigate our research questions, we designed a controlled set of experiments on the MNIST dataset. For all experiments, we used a consistent model architecture and optimizer to isolate the effects of the different batch orchestration strategies. Each configuration, representing a pair of a batch schedule and a sampling method, was run 10 times with different random seeds, and the results were averaged.

```
class MLP(nn.Module):
    def __init__(self):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(28 * 28, 64)
        self.fc2 = nn.Linear(64, 10)

    def forward(self, x):
        x = torch.flatten(x, 1)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

Listing 5.1: Multi-Layer Perceptron Architecture used for experiments on MNIST dataset

We used a simple Multi-Layer Perceptron (MLP) for all experiments, with an architecture consisting of a single hidden layer of 64 units with a ReLU activation function, and a 10-unit output layer for classification (Listing 5.2.2). To ensure that any observed performance differences were attributable to the batching strategies rather than complex optimization dynamics, we used a plain SGD optimizer across all conditions. Each training run lasted for a maximum of 100 epochs. We implemented an early stopping mechanism to halt training if the test set accuracy did not improve for 20 consecutive epochs, ensuring that we compare models at their point of optimal generalization.

For the VGL strategy, which uses sampling with replacement, one "epoch" is defined as performing a number of gradient updates equal to the number of batches in the full training dataset (i.e., n/b updates, where n is the training set size). This ensures a fair comparison of wall-clock time and computational effort. For all VGL experiments, VoG scores were calculated using the full history of gradients from all preceding epochs.

The four sampling methods (Single Shuffle, Random Reshuffle, VoG-ordered mini-batch SGD, and VGL) were evaluated under the three distinct batch size schedules with the following hyperparameters:

1. **Fixed large batch regime:**
    - Batch size: 512
    - Learning rate: $\eta = 0.1$

2. **Linear batch size increase:**
    - Initial batch size: $b_{min} = 4$
    - Maximum batch size: $b_{max} = 1024$
    - Learning rate: $\eta = 0.01$
    - Note: Due to early stopping, the maximum batch size of 1024 may not be reached in every run.
3. **Cyclical exponential batch size increase:**
    - Cycle period: $E = 15$ epochs
    - Initial batch size: $b_{min} = 4$
    - Maximum batch size: $b_{max} = 512$
    - Learning rate: $\eta = 0.01$
    - Note: Due to early stopping, the training can stop at any moment in the cycle.

Differences between the experiment suits. The section detailed two distinct suites of experiments designed to probe the effects of batch orchestration. While both suites explore curriculum learning, they differ fundamentally in their approach to defining and scheduling difficulty. The first suite of experiments uses typicality scoring, a static, heuristic-based metric, to create a fixed data curriculum. The second suite, uses VoG scoring to create a dynamic, model-dependent curriculum that adapts over time. However, a more subtle and more important distinction lies in the timescale of orchestration. The first suite of experiments explores **intra-epoch orchestration**, where the batch composition is manipulated at a fine-grained level within each epoch. For example, the increasing batch size strategy in that suite modifies the batch after every single model update, and this sequence is repeated in each epoch. In contrast, the experiments detailed in the second suite investigate **inter-epoch orchestration**. The batch size and sampling strategy are determined at the beginning of an epoch and remain fixed for its entire duration. Any changes, such as an increase in batch size according to a schedule, are applied only between epochs. This design allows us to separately analyze the effects of both fine-grained, step-by-step curriculum control and coarser, epoch-by-epoch strategic adjustments.

5.3 Results

5.3.1 Intra-epoch batch orchestration with typicality sampling

This section presents the empirical results for the first suite of experiments, which investigated four distinct batch orchestration strategies built upon a static, typicality-based data curriculum. To evaluate the effectiveness of each strategy against the standard mini-batch SGD baseline with uniform sampling (Single Shuffle), we use validation set accuracy as the primary performance metric.

We employ the non-parametric Friedman test to determine if the observed differences in model performance are statistically significant, with a significance level set at $\alpha = 0.05$. For a more granular comparison, we also report the number of experimental runs (out of 790 total across all datasets and typicality based scoring functions) where each curriculum strategy performed better (minimal data accuracy set to 0.01) or worse than the baseline. The aggregated rank comparison data is summarized in Table 5.1.

TABLE 5.1: Pairwise rank comparison of curriculum learning strategies. The table displays a "win-loss" matrix where each cell indicates the number of experiments in which the method in the row achieved a better final rank than the method in the column.

Method	# baseline outperforms method	# baseline underperforms method	p-value
Typicality-ordered SGD	313	467	1.18×10^{-6}
Typicality-ordered mini-batch SGD	476	303	3.78×10^{-6}
Batch repetition	195	493	1.8×10^{-5}
Increasing batch size	234	556	1×10^{-17}

RQ1 Typicality-ordered SGD. Our first objective was to assess the effect of a high-variance, ordered training process by evaluating the Ordered SGD strategy (batch size of one) against the baseline. The Friedman test yields a p-value of 1.18×10^{-6}, allowing us to confidently reject the null hypothesis. Across the experiments, the typicality-ordered SGD was superior to the baseline in 467 runs and inferior in 313 runs.

RQ2 Typicality-ordered mini-batch SGD. Second, we investigated our typicality-ordered mini-batch SGD, which intentionally groups similar-typicality examples together, standing in opposition to diversity sampling methods. The Friedman test yields a p-value of 3.78×10^{-6}, which again allows us to reject the null hypothesis. However, in this case, the results show that the strategy is significantly worse than the baseline, outperforming it in only 303 runs while underperforming in 476 runs. This finding refutes our initial hypothesis that increasing variance via batch homogeneity would be beneficial. A likely explanation is that while the variance between batches is high, the gradient signal within each batch is uninformative. The abrupt, high-magnitude changes between these homogeneous batches may create an optimization path that is too noisy and unstable for the optimizer to follow effectively, preventing it from settling in a good minimum.

RQ3 Batch repetition Third, we tested the Batch repetition strategy, hypothesizing that allowing the model to converge on typical examples before moving to atypical ones would improve generalization. The results strongly support this hypothesis. The Friedman test yields a p-value of 1.8×10^{-5}, and the strategy proved superior in 493 experimental runs compared to only 195 where it was inferior. This suggests that the proposed mechanism is highly effective. By forcing the model to repeatedly process a single, typical batch, we anchor the parameters in a stable, well-defined region of the loss landscape. The subsequent introduction of a new, slightly more atypical batch then acts as a regularizer, providing the necessary perturbation to escape a poor local minimum without destabilizing the entire training trajectory.

RQ4 Increasing batch size Finally, we evaluated the Increasing batch size strategy, which acts as a form of continuation method. This strategy yielded the most positive results. The Friedman test returns a p-value of 1×10^{-17}, providing the evidence to reject the null hypothesis. The method outperformed the baseline in 556 runs and underperformed in only 234. This strongly validates the hypothesis that blending exploration and exploitation within an epoch is a superior strategy. The initial small batches facilitate a broad exploration of the loss landscape. As the batch size grows, the training process smoothly transitions to a more stable, low-variance regime, allowing for rapid convergence within the wide, generalizable basin of attraction found during the

exploratory phase. This confirms that a dynamic batch size schedule, can effectively shrink the generalization gap.

5.3.2 Inter-epoch batch orchestration with VoG sampling

Small-batch regime

Before we analyze the effectiveness of the batch orchestration schedules, we first establish a performance baseline by evaluating the standard small-batch training regime. As we explained before, training with small batches often leads to models with a smaller generalization gap and superior final performance. However, this benefit comes at a significant cost: the training process is inefficient in terms of GPU utilization and requires a vast number of model updates, resulting in long wall-clock training times. Thus, we first quantify the best possible performance achievable under this paradigm, providing an accuracy target. The subsequent sections will then investigate whether our batch orchestration curricula can approach or match this high-performance baseline while operating in more computationally efficient, larger-batch regimes.

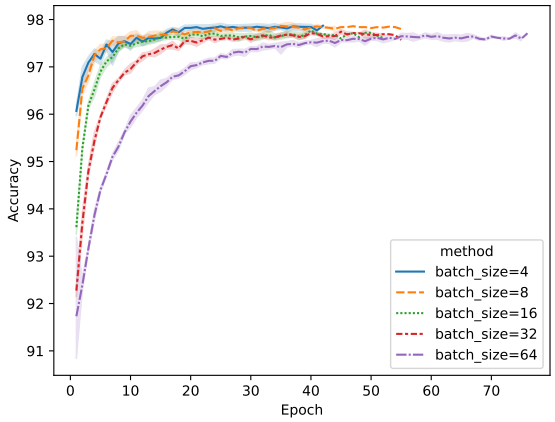


FIGURE 5.1: Test accuracy over epochs in the small-batch regime. The plot compares the performance of five different fixed, small batch sizes. Curves show the mean accuracy over 10 runs, with shaded regions representing the standard deviation. The results illustrate the generalization gap: the smallest batch sizes (4 and 8) achieve the highest final accuracy and converge in the fewest epochs. Performance slightly degrades as the batch size increases.

	$b = 4$	$b = 8$	$b = 16$	$b = 32$	$b = 64$
Accuracy	97.88%	97.89%	97.78%	97.75%	97.71%
Epochs until convergence	39.6	40	41	46.6	70
Model updates until convergence	594,000	300,000	153,000	86,000	65,625

TABLE 5.2: Performance comparison of models trained with different small batch sizes. The table reports the mean peak accuracy (%), epochs until convergence, and total model updates until convergence, averaged over 10 independent runs on the MNIST dataset.

The results of our experiment, shown in Figure 5.1 and the Table **??**, confirm the expected trend. The highest final accuracies are achieved with the smallest batch sizes. A batch size of 8 yielded the best performance, reaching a peak accuracy of 97.89% , closely followed by a batch

size of 4 at 97.88%. We observe a clear trend where increasing the batch size, even within this small-batch regime, leads to a slight but consistent degradation in the final model accuracy. This is also the consequence of fixed learning rate, $\eta = 0.01$. The learning curves show that the smallest batch sizes also converged in the fewest epochs. Based on these results, we establish a performance target of 97.9% accuracy. The central challenge for the batch orchestration strategies explored in the following sections will be to match this high level of generalization while significantly improving upon the computational and sample efficiency.

Large-batch regime

	Large batch regime	**Increasing batch size**	**Cyclical batch size increase**
Accuracy			
Single Shuffle	97.74%	97.32%	97.82%
Random Reshuffle	97.73%	97.28%	97.83%
VoG-ordered mini-batch SGD	97.69%	97.32%	97.80%
VGL	97.70%	97.55%	97.78%
Epochs until convergence			
Single Shuffle	47.2	56.3	50.9
Random Reshuffle	51	50.3	49.4
VoG-ordered mini-batch SGD	32.2	62.8	50.2
VGL	39.2	52	48.4
Model updates until convergence			
Single Shuffle	5,569	24,277	103,726
Random Reshuffle	6,018	23,580	99,787
VoG-ordered mini-batch SGD	3,799	24,790	101,638
VGL	4,625	23,784	100,788

TABLE 5.3: Performance comparison of four data ordering strategies across the three batch size orchestration schedules. The table reports the mean peak accuracy (%), epochs until convergence, and total model updates until convergence, averaged over 10 independent runs on the MNIST dataset. The curriculum-based methods (VoG-ordered mini-batch SGD and VGL) demonstrate superior sample efficiency in the larg batch regime, converging in significantly fewer epochs and total model updates with a small drop in accuracy.

RQ1 Fixed large-batch regime Our first experiment sought to determine which data ordering strategies could best mitigate the generalization gap associated with large-batch training. We hypothesized that the VGL and VoG-ordered methods could re-introduce beneficial gradient noise, helping the optimizer find better, more generalizable solutions compared to standard shuffling techniques.

The results, however, do not support this hypothesis in terms of final model accuracy (Table **??**). While Single Shuffle achieved the highest average peak accuracy, the differences between all four methods are negligible. A t-test comparing each method against the Single Shuffle baseline confirms this observation, yielding p-values of 0.93, 0.73, and 0.64 for Random Reshuffle, VGL, and VoG-ordered mini-batch SGD, respectively. As all p-values are well above the standard 0.05 significance threshold, we conclude that there is no statistically significant difference in the final performance of the four methods in this large-batch regime.

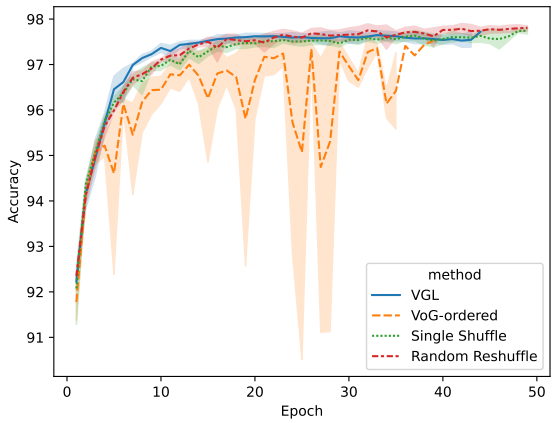


FIGURE 5.2: Test accuracy over epochs for four data orchestration strategies in the Fixed large-batch regime. The plot compares the performance of VGL, VoG-ordered mini-batch SGD, Single Shuffle, and Random Reshuffle. Curves represent the mean accuracy over 10 independent runs, with shaded regions indicating the standard deviation.

While the peak accuracies are similar, the training dynamics shown in the provided chart tell a more nuanced story. The curriculum-based methods (VoG-ordered mini-batch SGD and VGL) demonstrate superior sample efficiency, converging in significantly fewer epochs and total model updates with a small drop in accuracy compared to non-curriculum solutions. The Single Shuffle and Random Reshuffle methods exhibit stable and consistent learning curves. In contrast, the VoG-ordered method, which sequences homogeneous batches from easy to hard, is extremely unstable. Its accuracy fluctuates throughout training, with a very large variance between runs. This confirms that the strategy does indeed introduce significant variance, but this variance manifests as training instability rather than a clear improvement in generalization. The VGL method appears to be superior in the first 15 epochs, improving convergence rates compared to standard training, but did not achieve higher final accuracy. The training dynamics in the form of accuracy on the test set over time is shown in Figure 5.2.

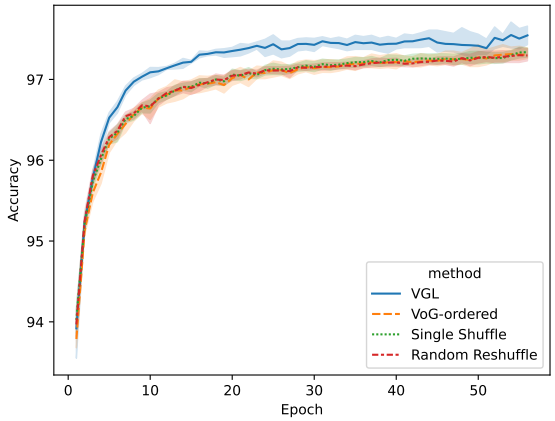


FIGURE 5.3: Test accuracy over epochs using the linear batch size increase schedule. The plot compares VGL, VoG-ordered, Single Shuffle, and Random Reshuffle. Curves show the mean accuracy over 10 runs; shaded regions represent the standard deviation. The VGL strategy achieves a statistically significant improvement in final accuracy and maintains a consistent performance advantage over the baseline methods throughout the training process.

To understand the training dynamics behind the accuracy results, we analyzed the gradient noise scale $\mathcal{B}$ for each of the four methods, as shown in the Figure 5.4. We initially hypothesized that the VGL and VoG-ordered methods would increase the $\mathcal{B}$, re-injecting a beneficial level of

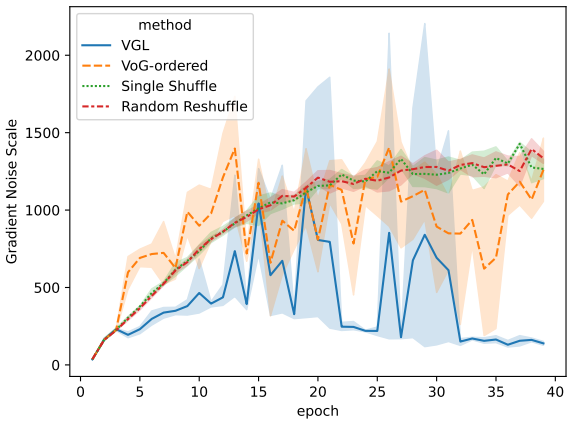


FIGURE 5.4: Gradient Noise Scale $\mathcal{B}$ over epochs in the fixed large-batch regime. Curves represent the mean $\mathcal{B}$ over 10 independent runs, with shaded regions indicating the standard deviation.

noise into the large-batch training process. The results both confirm and challenge this hypothesis in interesting ways.

- VoG instability: The Single Shuffle and Random Reshuffle methods behave as expected, showing a steady increase in gradient noise scale as the model converges and the gradient magnitude shrinks. The VoG-based sampling methods (VoG-ordered mini-batch SGD and VGL), in contrast, exhibit an extremely erratic scale that aligns perfectly with the VoG-ordered strategy's unstable accuracy curve. This confirms that its strategy of sequencing homogeneous batches does indeed inject a significant variance, but this noise is highly destabilizing rather than beneficial.

- Surprisingly low $\mathcal{B}$ of VGL: The most striking result comes from the VGL method. Contrary to our hypothesis, VGL produces a $\mathcal{B}$ that is lower than all other methods. After an initial rise and high fluctuations, the $\mathcal{B}$ for VGL drops and remains in a low-noise regime for the remainder of training.

VGL constructs batches that yield a more potent, high-magnitude gradient. Because the sampled training examples are the most informative fraction of the whole training dataset the signal is stronger and more directionally correct. Its magnitude is large relative to its variance, resulting in a lower gradient noise scale. This leads to the more stable and rapid initial convergence we observed in the accuracy plot, demonstrating that VGL improves training not by adding noise, but by improving the quality of the gradient itself.

RQ2 Linear batch size increase This experiment tested how different data ordering strategies interact with a linear batch size increase schedule. We hypothesized that VGL would be synergistic with this schedule, using the initial small-batch phase for exploration focused on the most informative samples to find a better basin of attraction, which would then be efficiently exploited in the later large-batch phase. The results presented in Table **??** strongly support this hypothesis, demonstrating a clear and statistically significant benefit for the VGL method. VGL emerges as the superior strategy, outperforming all baselines. A t-test confirms that the improvement of VGL over the Single Shuffle baseline is statistically significant ($p = 0.0018$). In contrast, the other methods show no significant difference from the baseline. The learning curves presented in Figure 5.3 chart visually confirm VGL's superiority. From the early epochs, the VGL curve separates

from and remains consistently above the other three methods, which are clustered together. This indicates that VGL uses small-batch regime period more efficiently than other methods through uncertainty-based sampling and maintain its performance advantage throughout training.

An important finding emerges when comparing the absolute performance across the two experimental regimes. Contrary to our initial hypothesis and the common "warm-start" intuition, the linear batch size increase schedule did not lead to superior final accuracies. On the contrary, for all four data ordering strategies, the peak accuracies achieved were consistently lower than those obtained in the fixed large-batch regime. This result provides another empirical support for research which finds that simple warm-starting can be suboptimal. The likely cause is that while the initial small-batch phase effectively explores the loss landscape, the subsequent gradual increase in batch size and corresponding decrease in gradient noise cause the optimizer to converge too greedily to the first sharp minimum it finds within a promising basin. Without an additional mechanism to force re-exploration, the model gets "stuck" and fails to find a better, more generalizable solution. We suspect that the issue might be resolved with more intelligent learning rate scheduling. We kept the same learning rate of $\eta = 0.01$ throughout the training. In **RQ1** we found that the large-batch regime is more efficient with a 10 times larger learning rate. Thus, we conduct ablation study to study effect of adaptive learning rate schedule. The results are describe in the Section 5.3.2. The results confirms that simply transitioning from an exploratory to a stable training phase is not sufficient, and highlights the potential need for explicit regularization and other hyperparameter tuning to make such schedules effective.

RQ3 Cyclic exponential batch size increase This final experiment tested which data ordering strategies could best leverage the "shock" of a cyclical batch size schedule. We hypothesized that the periodic reset from a large to a small batch size would act as an implicit regularizer, helping the optimizer escape poor local minima, and that a curriculum-aware strategy like VGL would harness this effect most effectively. The results indicate that while the cyclical schedule itself is a powerful mechanism, no single data ordering strategy provided a statistically significant advantage in terms of final peak accuracy (Table **??**). The performance of all four methods is similar, the VGL method stands as the worst. T-tests comparing each strategy against the Single Shuffle baseline confirm this, yielding high p-values (0.89, 0.44, and 0.74 for Random Reshuffle, VoG-ordered mini-batch SGD and VGL respectively), indicating no statistically significant differences.

The learning curves in the figure 5.5, however, reveal the powerful effect of the cyclical schedule.

- Visible "shocks": The chart in Figure 5.5 clearly shows periodic dips in accuracy for all four methods around epochs 15, 30, and 45. These moments correspond exactly to the batch size reset and visually confirm that the shock effectively perturbs the training process. The perturbation for VGL method after epoch 15 is minimal which suggests that VGL does not benefit from rapid batch size change like other methods.

- Progressive improvement: After each shock, all methods quickly recover and climb to a new, slightly higher accuracy plateau. This suggests that the cycle of exploration (at small batch sizes) followed by exploitation (at large batch sizes) is an effective strategy for navigating the loss landscape.

- VGL's initial advantage: As in the previous experiments, VGL exhibits a superior convergence rate during the first cycle (epochs 0-15). However, this advantage is neutralized in middle part of the training, where the performance of all four methods becomes indistinguishable. In the last two cycles VGL becomes inferior to other methods.

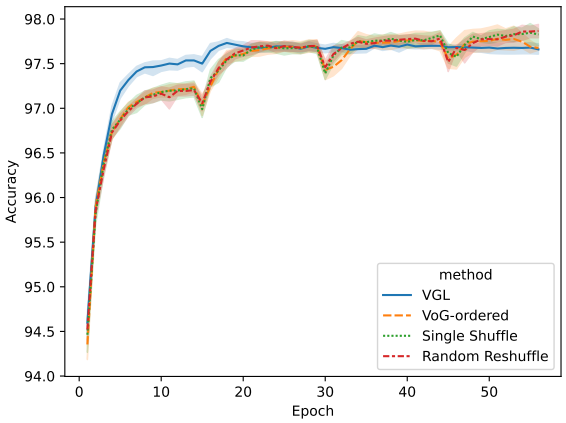


FIGURE 5.5: Test accuracy over epochs using the cyclical exponential batch size schedule. The plot compares VGL, VoG-ordered, Single Shuffle, and Random Reshuffle. The periodic dips in accuracy for all methods at epochs 15, 30, and 45 correspond to the batch size reset, visually confirming the "shock" effect. While VGL shows the fastest convergence in the initial cycle, all four methods leverage the cyclical schedule to ultimately reach a similar, high level of final performance.

The results suggest that the cyclical batch size schedule is a dominant regularizer that overrides the more subtle effects of the underlying data ordering. While our hypothesis that VGL would find a superior final solution was not confirmed, the experiment demonstrates that this periodic drastic batch size decrease is a potent technique in itself, enabling all tested methods to reach a high level of performance.

Adaptive learning rate

To further investigate why increasing batch size strategy underperformed, we conducted an ablation study focusing on the learning rate. Motivated by the fact that the learning rate should be increased in tandem with the batch size, we paired the linear batch size increase schedule with a linearly increasing learning rate (from $\eta = 0.01$ to $\eta = 0.1$ over 30 epochs).

The results showed a marked improvement in performance, with Single Shuffle achieving 97.77% and Random Reshuffle achieving 97.70% accuracy. This finding is significant, as it suggests that the previous underperformance was not caused by the dynamic batch sizing itself, but by the use of a mismatched, fixed learning rate. By co-adapting the learning rate and batch size, we can successfully harness the benefits of a warm-start schedule, validating the principle that these two components must be orchestrated together for optimal performance.

5.4 Conclusions

This chapter has systematically investigated the mechanism of batch orchestration, framing it as a powerful, data-driven implementation of curriculum learning. The central objective was to demonstrate that dynamic, adaptive batch size scheduling can serve as an effective remedy for the challenges of modern training regimes. The experiments explored both fine-grained intra-epoch schedules and broader inter-epoch schedules, treating the pacing component of a curriculum as the primary mechanism of study. To test the interaction with data ordering, these schedules were additionally combined with the scoring functions (static typicality and dynamic VoG scoring) that were described in previous chapters.

By positioning these methods within the formal CL framework, we can analyze their contributions. The primary focus of this chapter was the pacing function, which was implemented directly through various batch size schedules, from monotonic increases to non-monotonic cycles, operating at different timescales.

The experimental results demonstrate how these different pacing strategies fulfill the key promises of curriculum learning:

- **Performance improvement.** The choice of pacing function proved critical for addressing the generalization gap. When combined with typicality scoring intra-epoch batch scheduling grounded in continuation method theory, particularly batch repetition and increasing bach size, proved to be effective in closing generalization gap. In trainings under large-batch regime, a simple linear batch size increase was found to be insufficient on its own. However, the more sophisticated schedules (cyclic exponential batch increase) were highly effective and guided the optimizer to wider, more robust minima.

- **Convergence rate improvement.** Our inter-epoch experiments revealed two distinct ways in which batch orchestration can improve training efficiency. First, we showed that even within the computationally efficient large-batch regime, VGL can significantly speed up convergence. By prioritizing the most informative examples, VGL consistently achieved a steeper initial learning curve, reaching its optimal performance in demonstrably fewer epochs and model updates than standard shuffling baselines. Second, and more significantly, the cyclical exponential batch size schedule proved that an intelligent pacing function can bridge the gap between small-batch performance and large-batch efficiency. This schedule successfully narrowed the generalization gap, achieving a final accuracy (97.8%) slightly worse than best-performing small-batch methods. Crucially, it accomplished this with far greater sample efficiency, requiring approximately 100,000 model updates to converg, a substantial improvement over the 154,000 updates needed by a similarly performing small-batch model (with a batch size of 16). This provides compelling evidence that curriculum learning, specifically through the design of the pacing function via batch orchestration, can significantly improve convergence rates with minimal, if any, degradation in final model accuracy.

The interaction between batch composition, shuffling strategies, and regularization techniques creates complex optimization landscapes that determine both convergence behavior and final performance. This work has explored a region of the curriculum learning design space, focusing on curricula where the pacing function, implemented via batch size scheduling at multiple timescales, is the primary driver of performance. Most significantly, this research demonstrates that training dynamics are far more liable than previously understood. The ability to influence convergence behavior, escape poor minima, and improve generalization through strategic batch orchestration opens new avenues for optimization that complement architectural and algorithmic advances. As models continue scaling, these insights become increasingly critical for efficient and effective training strategies.

Chapter 6

Transitional Objective Learning

This chapter introduces Transitional Objective Learning (TOL), a novel curriculum learning framework designed to improve the training of models by leveraging inherent task hierarchies. This chapter presents a specific, practical implementation of a task-space curriculum and provides a comprehensive empirical evaluation of its effectiveness. The work detailed here demonstrates how structuring the learning process from coarse-grained to fine-grained objectives can lead to significant improvements in model performance, convergence speed, and optimization stability.

Before delving into the specifics of this research, it is important to contextualize my broader scientific activities that contributed to this work. Beyond the core research on curriculum learning presented in this thesis, I contributed to the Multibridge project [1], an initiative aimed at strengthening the collaboration between academia and the non-academic sector. Building on the findings of the Across-domain Investigations in Multilingualism (ADIM)[2] project, which studied the acquisition of third languages (L3), Multibridge focuses on transferring scientific knowledge into practical, real-world applications.

My primary scientific achievement within this initiative was centered on fostering open science and democratizing access to key resources for speech technology. This was accomplished through two main contributions. First, I processed and publicly released the LnNor corpus , a large, multilingual spoken dataset, on the Hugging Face platform [3], making it accessible to the global research community. This dataset was instrumental in the experiments conducted in this chapter. Second, to further bridge the gap between scientific research and practical application, I trained a high-performing phoneme recognition model on this data and published it as an open-source model on Hugging Face [4].

The chapter is structured to present the Transitional Objective Learning framework in full detail. Section 6.1 provides a detailed introduction, beginning with the necessary background on phoneme production and the Connectionist Temporal Classification loss, before formally defining the TOL framework and situating it within the context of related works. Section 6.2 describes the experimental methodology, including the datasets and the training setup. Section 6.3 presents a comprehensive analysis of the results, evaluating TOL's impact on the final Phoneme Error Rate, convergence behavior, and optimization stability, supported by an ablation study. Finally, 6.4 offers concluding remarks, summarizing the contributions of this work and outlining promising directions for future research.

[1] https://huggingface.co/MultiBridge

[2] https://adim.web.amu.edu.pl/en

[3] https://huggingface.co/datasets/MultiBridge/LnNor funded by EEA Financial Mechanism and Norwegian Financial Mechanism

[4] https://huggingface.co/MultiBridge/wav2vec-LnNor-IPA-ft

6.1 Introduction

Training contemporary machine learning models, especially those featuring deep architectures and vast numbers of parameters, presents significant challenges related to navigating complex optimization landscapes, demanding extensive computational resources, and requiring massive datasets to achieve robust generalization. These challenges are particularly amplified in complex domains like Automatic Speech Recognition (ASR). ASR systems must not only manage the difficulties of training large-scale models but also contend with high degrees of variability in the input audio signal, stemming from diverse speakers, accents, background noise, and channel effects, while learning the mapping from continuous acoustic waveforms to discrete symbolic outputs like phonemes or words.

To address these training challenges, this work explores the paradigm of curriculum learning, which structures the learning process by progressing from simpler to more complex tasks. Specifically, we introduce Transitional Objective Learning (TOL), a novel curriculum learning strategy defined in the task space. TOL leverages the inherent hierarchical structure of the target outputs to create a curriculum of learning objectives that guides a model from coarse-grained concepts to fine-grained details. Before detailing the TOL methodology, it is essential to first understand the natural hierarchy of speech sounds that makes ASR an ideal domain for this approach.

6.1.1 Phoneme production

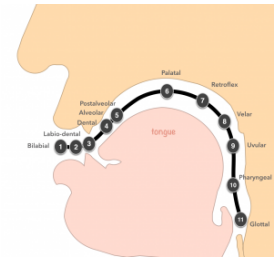
While many state-of-the-art ASR systems, such as OpenAI's Whisper (Radford et al., 2023), operate directly at the word level, phoneme recognition remains a crucial subtask with vital applications where a fine-grained analysis of speech sounds is necessary. In these domains, understanding how something was said is as important as what was said. For instance, in computer-assisted language learning, phoneme-level models are indispensable for providing diagnostic feedback on pronunciation errors (Olejnik et al., 2023), as word-level transcription often fails to capture the subtle phonetic mistakes made by learners. Similarly, in speech pathology, phoneme recognition systems serve as objective tools for the assessment and treatment of speech disorders (Lilley et al., 2017; Nagao et al., 2012), enabling a detailed analysis of specific articulation errors. Furthermore, phoneme-based models are instrumental in developing ASR for low-resource languages (Siminyu et al., 2021); by leveraging shared phonetic inventories, a multilingual phoneme recognizer can be used to transfer acoustic knowledge and bootstrap a system for a new language with limited training data. These applications, which demand a level of granularity that word-level models cannot provide, underscore the continued importance of phoneme recognition as a fundamental and impactful subtask of ASR.

Phonemes are not an unstructured set of symbols. Rather, they can be organized in a hierarchical system based on the physical properties of their production. This structure provides a natural basis for creating a meaningful curriculum. Consonants and vowels, the two primary categories of speech sounds, are defined by distinct sets of features.

Consonants are primarily classified along three main articulatory dimensions:

- **Voice**: Voice dimension describes whether the vocal cords vibrate during the production of the sound. For example, the sound /b/ is voiced, while its counterpart /p/ is voiceless. The cells in the IPA chart presented in Figure 6.1(b) contain pairs of phonemes from which the left phoneme is voiceless and the right phoneme is voiced.

- **Manner**: Manner refers to how the airstream is manipulated within the vocal tract. It ranges from a complete stoppage of air in plosives like /t/, to a continuous, turbulent airflow



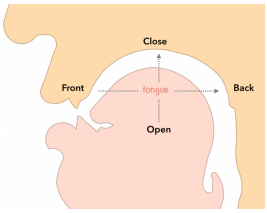
(a)

CONSONANTS (PULMONIC) © 2015 IPA

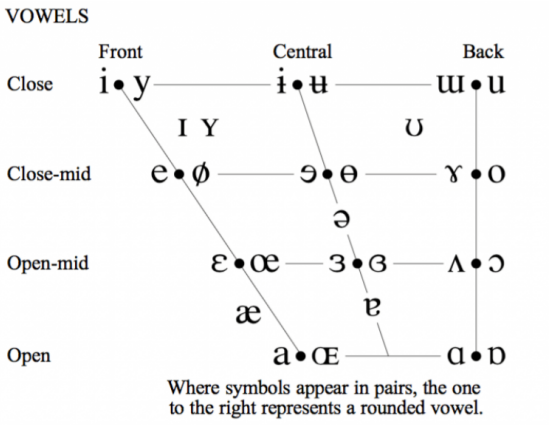
	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k g	q ɢ		ʔ
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ		
Trill	ʙ			r					ʀ		
Tap or Flap		ⱱ		ɾ		ɽ					
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative				ɬ ɮ							
Approximant		ʋ		ɹ		ɻ	j	ɰ			
Lateral approximant				l		ɭ	ʎ	ʟ			

Symbols to the right in a cell are voiced, to the left are voiceless. Shaded areas denote articulations judged impossible.

(b)



(c)



(d)

FIGURE 6.1: (a) A sagittal view of the human head, illustrating the primary places of articulation for consonant sounds. It provides an anatomical reference for the locations listed across the top of that chart. (b) The standard chart of pulmonic consonants from IPA. This chart systematically organizes consonant sounds based on two key phonetic dimensions: the place of articulation (represented in the columns) and the manner of articulation (represented in the rows). Within many cells, symbols appear in pairs. The symbol on the left represents a voiceless consonant (produced without vocal fold vibration), while the symbol on the right represents a voiced consonant (produced with vocal fold vibration). (c) A schematic diagram that illustrates the relationship between the tongue's position in the mouth and the dimensions of the IPA vowel chart. It simplifies the complex movements of the tongue into two primary axes (d) The IPA chart for vowels organized in a quadrilateral shape known as a vowel trapezium. This chart maps vowel sounds based on the position of the tongue in the oral cavity during their production. Where symbols appear in pairs, the one on the right represents a rounded vowel (produced with rounded lips), and the one on the left represents an unrounded vowel. Source: https://www.happyhourspanish.com/ipa-chart-language-learners

in fricatives like /s/, to air being redirected through the nasal cavity in nasals like /m/. Manner categories were presented in the first column of the IPA consonant chart (Figure 6.1(b))

- **Place**: Place describes where in the vocal tract the primary constriction occurs. This can be at the lips for bilabial sounds like /m/, just behind the teeth for alveolar sounds like /t/, or at the back of the mouth for velar sounds like /k/. Place categories were presented in the first row of the IPA consonant chart (Figure 6.1(b))

Vowels, in contrast, are produced with a relatively open vocal tract and are defined by the position and shape of the tongue and lips. Their primary characteristics are determined by tongue height (how open or closed the mouth is), tongue backness (how front or back the tongue is in the mouth) and roundness as presented in the IPA chart in Figure 6.1(d).

These phonetic features create a natural hierarchy. A very general distinction can be made between vowels and consonants. Within consonants, a broad distinction can be made between voiced and voiceless sounds. Further down the hierarchy are finer-grained distinctions based on

manner and place of articulation. This multi-level structure means that the complex task of identifying dozens of specific phonemes can be decomposed into a sequence of simpler, related subtasks. A model could first learn to distinguish broad phonetic categories before being asked to master the differences between phonetically similar sounds.

6.1.2 TOL framework

So far, we have explored data-driven curricula which focus on scheduling the training data based on some notion of difficulty. Transitional Objective Learning (TOL) is a dynamic, task-driven curriculum framework that, instead of reordering data instances, manipulates the learning task itself over the course of training. TOL is designed for complex problems where the final learning objective can be decomposed into a sequence of related, simpler sub-tasks, particularly those with a natural hierarchy. The core mechanism of TOL involves utilizing a set of distinct loss functions, where each loss function represents the learning objective at a different level of granularity or abstraction. The framework then dynamically transitions the model's focus between these objectives as training progresses, guiding the model from coarse-grained to fine-grained target representations. This is achieved by defining the total loss at a given training time t as a weighted sum of the individual loss functions ℓ_i:

$$J_t^{\Upsilon} = \ell_{\Upsilon}(\theta_{t-1}; B_t) = \left[\frac{1}{j}\sum_{i=1}^{j} w_i^t \ell_i(f_\theta(\mathbf{x}), \mathbf{y})\right] \tag{6.1}$$

In this formula, w_i^t represents the weight of the task i in the training iteration t. These weights must be non-negative and sum to 1 at each epoch. The weights are determined by a pacing mechanism $\mathbf{w^t} = \mathcal{P}(\mathbf{s^t}, t)$ that relies on an importance scores $\mathbf{s^t} = \mathcal{S}(\mathcal{L}, \sqcup)$, where $\mathcal{L}$ is a set of subtasks. Examples of possible importance scoring functions include:

- *linear schedule*: $\mathbf{s^t} = \mathbf{a^t} + \mathbf{b}$, where $\mathbf{a}$ determines the rate of importance increase/descrease,
- *exponential schedule*: $\mathbf{s^t} = \mathbf{b} \times e^{\mathbf{a^t}}$,
- *logarithmic schedule*: $\mathbf{s^t} = \mathbf{a^t} \times \log(t+1) + \mathbf{b}$,
- *sigmoid schedule*: $\mathbf{s^t} = \frac{1}{1+e^{-t}}$,
- *piecewise schedule*: different functions for different epoch ranges.

The weights $\mathbf{w}$ are calculated by applying the Softmax function to the importance scores $\mathbf{s^t}$ at each epoch t:

$$w_i^t = \frac{e^{s_i^t}}{\sum_{j=1}^{n} e^{s_j^t}} \tag{6.2}$$

which guarantees that all weight functions are always strictly positive (since the exponential function is always positive), and that the weights all sum to 1.

The key novelty of TOL lies in the explicit, dynamic transition between different learning objectives over the course of training. In contrast to hierarchical multitask methods that use auxiliary losses simultaneously to learn intermediate representations, TOL explicitly schedules the learning process to shift the model's focus from an easier, coarse-grained task to the more challenging, fine-grained target task.

6.1.3 Connectionist Temporal Classification

Connectionist Temporal Classification (CTC) loss was originally introduced by Graves et al., 2006 to address sequence labeling challenges in Automatic Speech Recognition, where the alignment between the typically long input sequence (e.g., audio frames $\mathbf{x} = x_1, \ldots, x_T$) and the much shorter target label sequence (e.g., phonemes or characters $\mathbf{y} = y_1, \ldots, y_e : e < T$) is unknown and variable. Standard frame-level training requires pre-determined alignments, which are often unavailable or hard to obtain accurately. CTC solves this by defining a conditional probability of the target sequence $\mathbf{y}$ given the input $\mathbf{x}$, denoted as $P(\mathbf{y}|\mathbf{x})$, which marginalizes over all possible alignments. To achieve this, the underlying neural network first produces a probability distribution $P(p_t|\mathbf{x})$ for each input time step $t \in [1, T]$ over an extended alphabet that includes all target labels plus a special blank symbol ϵ. A sequence of these frame-wise predictions forms a path $\mathbf{p} = p_1, \ldots, p_T$. CTC then introduces a many-to-one mapping function $\mathcal{M}$, which transforms these paths into target sequences by first collapsing consecutive identical labels and then removing all blank symbols (e.g., $\mathcal{M}(\text{AA}\epsilon\text{ABB}) = \text{AAB}$). The probability of the target sequence $\mathbf{y}$ is then calculated by summing the probabilities of all possible paths p that map to $\mathbf{y}$ via this function, i.e., $P(\mathbf{y}|\mathbf{x}) = \sum_{p \in \mathcal{M}^{-1}(\mathbf{y})} P(p|\mathbf{x})$, where the probability of a single path is typically $P(p|\mathbf{x}) = \prod_{t=1}^{T} P(p_t|\mathbf{x})$.

The inclusion of the blank label ϵ is crucial: it allows the model to emit "nothing" at certain time steps, providing flexibility in aligning sequences of different lengths, and it separates identical consecutive target labels, enabling the model to correctly predict repetitions (e.g., mapping AϵA to AA). This summation over a potentially exponential number of paths is computed efficiently using a dynamic programming approach analogous to the forward-backward algorithm in Hidden Markov Models. Finally, the CTC loss function is the negative log probability of the correct target sequence, $L_{\text{CTC}} = -\log P(\mathbf{y}|\mathbf{x})$. Minimizing this loss trains the network end-to-end to output high probabilities for label sequences that correctly collapse to the target transcription, without requiring explicit frame-level annotations during training.

A well-known challenge in training models with CTC is the emergence of a "peaky behavior" in the model's outputs (Bluche et al., 2016; Zeyer et al., 2021). This phenomenon is characterized by the model's output probability distribution being dominated by the blank token (ϵ) at most time steps, with sharp, localized "peaks" of high probability for the actual target labels. This behavior is not specific to a particular model architecture like RNNs, but is an inherent property of the CTC training criterion itself. The emergence of this behavior can be understood as a two-stage process. In the initial phase of training, often called the "suppression phase" (H. Li & Wang, 2020), the model learns to predict the blank token almost exclusively. This occurs because the blank token is the "dominant label" in the CTC alignment graph. The blank token is present between each character, so the model is more likely to assign a higher posterior probability to the blank token than any other. This leads to an early local minimum where predicting blanks exclusively results in a loss decrease and flat loss landscape. As a consequence, gradient magnitudes remain small, the model receives weak training signals for actual phoneme distinctions during these initial stages, and learning meaningful label alignments is delayed (Bluche et al., 2016). As training progresses, the posterior probability for blank token is close to one, the model enters the "peaking phase"(H. Li & Wang, 2020), where it starts to assign higher probabilities to actual target labels at specific, highly localized time steps, resulting in the characteristic "peaky" behavior of CTC models. This transition is crucial for the model to learn accurate alignments and make meaningful predictions (Zeyer et al., 2021).

The "peaky behaviour" limits the model's usability in finding phoneme or word boundaries

in audio. Our proposed method, TOL, is designed to mitigate this undesirable blank-dominance phase. TOL introduces a curriculum that begins with a coarse-grained classification task, such as distinguishing between broad phonetic categories (e.g., vowels, voiced consonants, and voiceless consonants), before transitioning to the fine-grained phoneme recognition task. The hypothesis is that this initial coarse-grained objective balances the influence of the blank token. Because the broad phonetic categories are more general and acoustically distinct, the model can learn to recognize them more easily. This provides a stronger and more distributed training signal early on, preventing the model from collapsing into the simple local minimum of exclusively predicting blanks and thereby facilitating a smoother, faster transition to learning meaningful alignments.

6.1.4 Related work

Curriculum learning emerged as a technique for enhancing the performance and robustness of ASR systems, particularly in challenging scenarios like recognizing speech in noisy environments (Braun et al., 2017) or dysarthric speech (W. Lee et al., 2025). In the context of noise-robust ASR, Braun et al., 2017 introduced a data-driven curriculum learning strategy. This approach begins with training examples at low signal-to-noise ratio (SNR) values and progressively includes examples with higher SNR values. Additionally, they utilized per-epoch noise mixing to generate noisy training examples online during training, enabling dynamic changes in the SNR of the training data. These methods collectively improved the noise robustness of ASR systems, demonstrating significant reductions in word error rates under various noisy conditions.

For dysarthric speech recognition, W. Lee et al., 2025 proposed Dynamic Phoneme-level Contrastive Learning (DyPCL), which incorporates a dynamic data-driven curriculum learning strategy. This method progressively transitions from easy negative examples to more challenging ones based on phonetic similarity, determined by phoneme distance metrics. By organizing negative examples by difficulty levels, DyPCL effectively addresses the variability in dysarthric speech, leading to substantial improvements in recognition accuracy across different levels of dysarthria severity.

Building on these concepts, hierarchical multi-task learning approaches were proposed to explicitly exploit the structure of speech. Sanabria and Metze, 2018 introduced a hierarchical multitask learning framework in which multiple CTC losses at various granularities (from characters up to word-level units) encourage the model to form useful intermediate representations of speech. The architecture of the model consists of a shared encoder and auxiliary components providing task-specific CTC loss functions. The shared encoder learns simultaneously multiple tasks by forming representations useful for recognizing character, subword and final word labels. This approach experimentally proves the importance of using the hierarchy of speech characteristics to guide learning. The idea of hierarchical multitask learning is further extended by J. Lee and Watanabe, 2021, who proposed intermediate loss regularization to relax the conditional independence assumption of CTC by conditioning later predictions on intermediate ones. Such methods provide a natural way to gradually guide the model from learning basic acoustic distinctions toward more nuanced phoneme-level classifications. In a related vein, Goyal and Ghosh, 2020 proposed a hierarchical class-based curriculum loss for multi-label classification, integrating hierarchical constraints directly into the loss function. This approach assigns non-uniform weights to labels based on their position within the hierarchy. It enforces a constraint such that the loss of a child node must exceed that of its corresponding parent node, effectively realizing a dynamic class-based curriculum learning strategy. When the parent's loss surpasses the child's, the model prioritizes minimizing the parent's loss. In other words, this class-based CL loss forces the model to first correctly recog-

nize general concepts (vowel, consonant, silence), and only after learning these concepts the loss steers the model in the direction of more fine-grained labels.

Our work builds on these findings by proposing the Transitional Objective Learning and applying the framework to modern, pretrained transformer-based models. In contrast to prior hierarchical multitask methods that use multiple auxiliary CTC losses to learn intermediate representations simultaneously (J. Lee & Watanabe, 2021; Sanabria & Metze, 2018), or methods that incorporate hierarchical constraints into the loss (Goyal & Ghosh, 2020), our method explicitly schedules the learning process via weighted loss functions that transition the model's focus from an easier coarse-grained task to a more challenging fine-grained task. In other words, TOL uses explicit, dynamic transition between different learning objectives, represented by distinct loss functions, over the course of training. TOL does not require any ordering of training instances, and can be used with all popular optimizers.

6.2 Experiments

6.2.1 Datasets

We evaluate the proposed framework on three phoneme recognition datasets: TIMIT[5], LnNor[6], and Vibravox[7]. Each dataset presents different phonetic inventories and speaker conditions, allowing us to assess the robustness and generalizability of our approach.

The **TIMIT** Acoustic-Phonetic Continuous Speech Corpus (Garofolo et al., 1993) is a widely utilized benchmark in phoneme recognition research. It comprises recordings from 630 speakers across eight major American English dialect regions, each delivering ten phonetically rich sentences. The corpus includes manual phoneme annotations.

The **LnNor** corpus (Wrembel et al., 2024a, 2024b) provides approximately 101 hours of multilingual speech recordings from 231 speakers captured using head-worn microphones, primarily designed to study cross-linguistic influence and variation in the acquisition of Norwegian as a third language (L3/Ln) by L1 Polish learners, alongside native Polish, English, and Norwegian controls. The recordings encompass a variety of elicitation tasks in Polish, English, and Norwegian, including reading (words, sentences, texts), descriptive tasks (picture descriptions), and narrative tasks (story/video retelling). Annotations include rich speaker metadata (demographics, language background/proficiency based on LHQ, etc.), recording details, and orthographic transcriptions, with word alignment available for parts of the corpus to facilitate detailed linguistic analysis. Phoneme annotations in LnNor are generated through a semi-automatic process, combining forced alignment techniques with manual verification to ensure annotation quality. Due to the extensive size of the dataset, only the split with Polish speech has been used for the experiments.

The **Vibravox** dataset (Hauret et al., 2024) offers over 45 hours of French speech recordings per sensor from 188 participants, specifically created to advance research into speech processing using body-conduction audio sensors. The corpus contains recordings captured simultaneously from five different body-conduction sensors (two in-ear, two bone conduction, one laryngophone) and a reference airborne microphone, under diverse acoustic conditions including quiet and controlled noisy environments simulated via a 3D spatializer. Annotations provided include details about the recording conditions and linguistic transcriptions (phonetic). Phoneme annotations were initially generated using an automatic phonemizer based on the International Phonetic Alphabet (IPA), followed by manual corrections for words with ambiguous or incorrect transcriptions.

[5] https://huggingface.co/datasets/speech31/timit_english_ipa
[6] https://huggingface.co/datasets/MultiBridge/LnNor
[7] https://huggingface.co/datasets/Cnam-LMSSC/vibravox

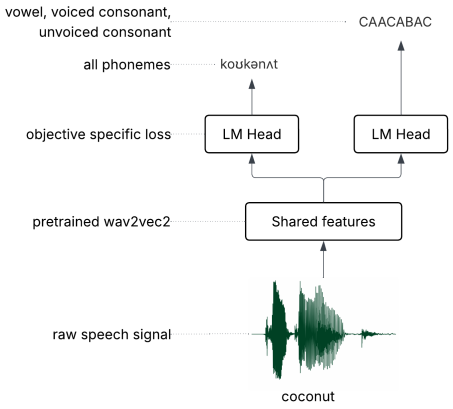


FIGURE 6.2: Two-level hierarchy TOL implementation for phoneme recognition of the word "coconut" with pre-trained wav2vec2 model. Two language modeling heads implement two distinct loss functions.

6.2.2 Setup

For each dataset, we use a pre-trained wav2vec 2.0 model as the base acoustic model, selected to match the language of the dataset:

- TIMIT: wav2vec2-base[8], pre-trained on English speech,
- LnNor: wav2vec2-XLSR[9], a cross-lingual model trained on multiple languages,
- Vibravox: wav2vec2-voxpopuli-fr[10], pre-trained on French speech.

The wav2vec 2.0 model is a self-supervised framework designed to learn speech representations directly from raw audio, incorporating three principal components in its architecture. A feature encoder composed of temporal convolutional layers processes the input waveform, transforming it into latent speech representations while reducing temporal resolution and capturing local acoustic features. These latent representations serve as input to a transformer-based (Vaswani et al., 2023) context network, which models long-range dependencies to construct contextualized representations of the speech signal. Parallel to the transformer, a quantization module discretizes the feature encoder's output using a Gumbel-Softmax approach; this is integral to the self-supervised pre-training task where parts of the latent representations are masked, and the model learns to predict the appropriate quantized representation for these masked segments based on context, similar to masked language modeling in BERT (Devlin et al., 2019). To effectively adapt the pre-trained model for tasks such as phoneme recognition, the feature encoder is typically frozen during fine-tuning, and the remaining parts of the network with a language modeling head are trained using the CTC loss function.

To evaluate the effectiveness of the proposed TOL framework, we conducted a series of experiments comparing standard fine-tuning of pre-trained wav2vec 2.0 models against fine-tuning augmented with TOL. For each dataset, two training regimes were employed: (1) standard training with the CTC loss and (2) training using the TOL framework. Figure 6.2 presents the implementation of the TOL framework.

In the TOL regime, we structured the auxiliary losses hierarchically across two levels of granularity. At the first level, the model learned to distinguish among four broad token classes: vowels,

[8]https://huggingface.co/facebook/wav2vec2-base
[9]https://huggingface.co/facebook/wav2vec2-large-xlsr-53
[10]https://huggingface.co/facebook/wav2vec2-base-fr-voxpopuli-v2

Model	Learning rate	Optimizer	Weight decay	Batch size
wav2vec2-base	1×10^{-5}	Adam	0.001	16
wav2vec2-xlsr	1×10^{-4}	Adam	0.0001	8
wav2vec2-voxpopuli-fr	1×10^{-5}	Adam	0.0001	32

TABLE 6.1: Hyperparameters

voiced consonants, voiceless consonants, and the blank token. At the second level, the model learned to predict specific phonemes and blank token, with the number of phoneme classes determined by the dataset: 48 for TIMIT, 48 for LnNor, and 33 for Vibravox. The auxiliary classification of vowel/voiced consonant/voiceless consonant tokens served as a coarse-grained task, while the fine-grained phoneme recognition task served as the final objective. The transitional dynamic between the coarse-grained and fine-grained objectives was implemented via a sigmoid-based schedule ($\mathbf{w}(\mathbf{t}) = \frac{1}{1+e^{-t}}$) to gradually increase the weight of the fine-grained phoneme loss over training epochs, while simultaneously decreasing the weight of the coarse-grained auxiliary loss using reverted sigmoid schedule ($1 - \mathbf{w}(\mathbf{t})$).

For each dataset and training regime, we generated 10 independent instances of the model, each initialized with different random seeds. All models were trained while monitoring the Phoneme Error Rate and validation loss across epochs. Results were averaged across these runs. The hyperparameters for each model-dataset pair were selected via a grid search. This ensured that comparisons between standard training and TOL were made under optimally tuned settings for standard training. The grid search was performed over:

- Optimizers: SGD and Adam.
- Learning rate: 10^{-5}, 10^{-4}, 10^{-3}, 5×10^{-3}, 10^{-2}, 5×10^{-2}
- Weight decay: 0, 10^{-4}, 10^{-3}, 10^{-2}

The best configurations (Table 6.1), evaluated by the lowest validation loss, were selected for the final comparison.

6.3 Results

In this section, we report and analyze the performance of TOL across English, French and Polish languages. The evaluation focuses on three aspects: Phoneme Error Rate, training convergence and optimization stability. Additionally, we investigate the impact of the linguistically informed hierarchy embedded in TOL through an ablation study.

6.3.1 Phoneme Error Rate

Dataset	Baseline PER	TOL PER	Error Red.	t (p-value)
TIMIT	$0.049 \pm .003$	$0.044 \pm .002$	10.2%	3.61 (.003)
LnNor	$0.084 \pm .006$	$0.076 \pm .005$	9.5%	2.90 (.009)
Vibravox	$0.063 \pm .012$	$0.054 \pm .003$	14.3%	2.29 (.042)

TABLE 6.2: Average PER for Baseline vs. TOL with t-test statistics

Table 6.2 shows the final PER achieved by baseline wav2vec 2.0 models trained with standard CTC loss and the same models trained with TOL. Across all studied datasets, TOL consistently

outperforms the standard approach, achieving relative error reductions between 9.5% and 14.3%. Independent samples t-tests confirm that these improvements are statistically significant across all datasets: TIMIT ($t = 3.61$, $p = .003$), LnNor ($t = 2.90$, $p = .009$), and Vibravox ($t = 2.29$, $p = .042$).

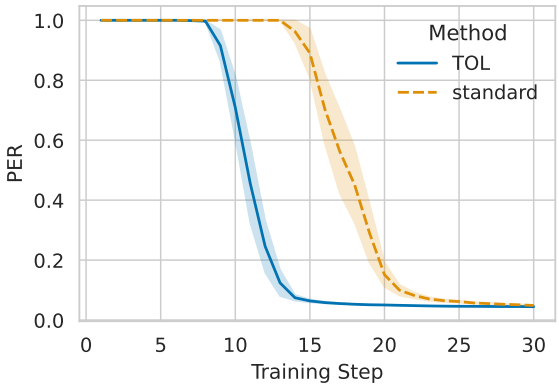


FIGURE 6.3: Phoneme Error Rate (PER) on the TIMIT validation dataset over training steps. Here, the training step is the whole epoch. The chart compares the convergence behavior of Transitional Objective Learning (TOL) against a standard training baseline. The results clearly show that the TOL curriculum leads to significantly faster convergence, achieving a low PER in approximately half the epochs required by the standard approach.

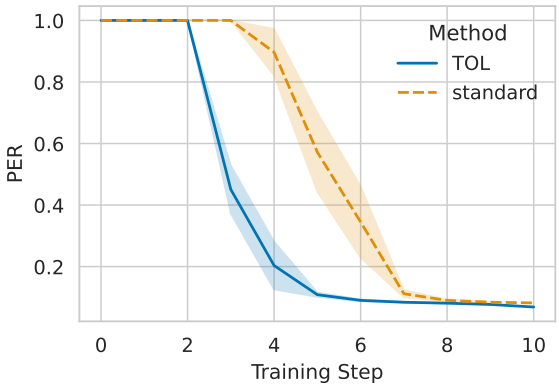


FIGURE 6.4: PER on the LnNor validation dataset (Polish split). The chart compares the Phoneme Error Rate (PER) over the course of training for the Transitional Objective Learning (TOL) method against a standard baseline. The results clearly demonstrate that the TOL curriculum leads to substantially faster convergence, reaching a low error rate in approximately half the epochs required by the baseline.

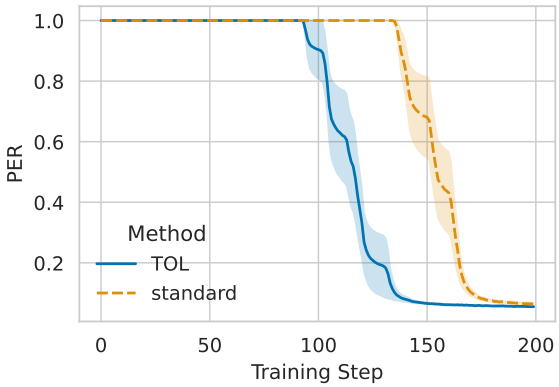


FIGURE 6.5: PER on the Vibravox validation dataset (speech$_n oisysplit$).$Thischartcomparestheconvergenceof Transitional Objective L$

6.3.2 Convergence behavior

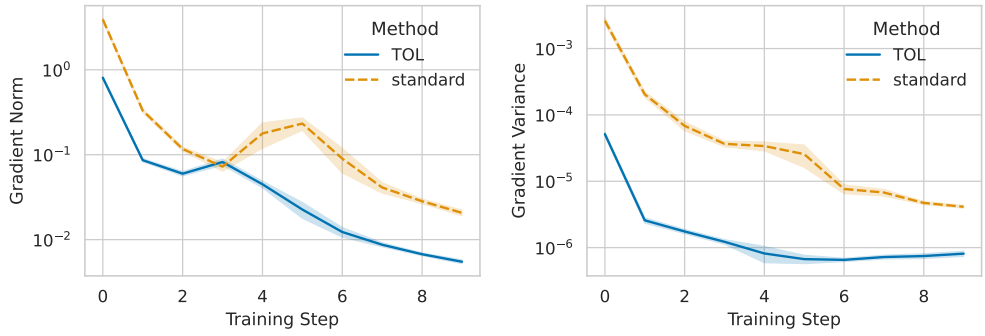
CTC models are designed to handle sequence alignment without explicit frame-level labeling. To achieve this, the blank token was introduced, allowing the model to output nothing at certain time steps, which helps in aligning input and output sequences of differing lengths. However, during the initial stages of training, models default to predicting the blank token across most time steps. This behavior arises because in CTC the blank token is present between each character, so the model is more likely to assign a higher posterior probability to the blank token than any other. This leads to an early local minimum where predicting blanks exclusively results in a loss decrease and flat loss landscape. As a consequence, gradient magnitudes remain small, the model receives weak training signals for actual phoneme distinctions during these initial stages, and learning meaningful label alignments is delayed (Bluche et al., 2016). This phenomenon is sometimes referred to as the "suppression phase" (H. Li & Wang, 2020). As training progresses, the posterior probability for blank token is close to one, the model enters the "peaking phase" (H. Li & Wang, 2020), where it starts to assign higher probabilities to actual target labels at specific time steps, resulting in the characteristic "peaky" behavior of CTC models. This transition is crucial for the model to learn accurate alignments and make meaningful predictions (Zeyer et al., 2021).

To compare the convergence behaviour and final performance of TOL with standard training, we monitored validation PER over training epochs. Figures 6.3, 6.4, and 6.5 present the evolution of validation PER over training steps for the TIMIT, LnNor, and Vibravox datasets, respectively. On TIMIT (Figure 6.3), models trained with TOL achieve a lower PER more rapidly, converging by epoch 15, whereas standard training requires around 30 epochs to reach a similar level. On LnNor (Figure 6.4), TOL again outpaces the baseline, with convergence observed by epoch 6, compared to epoch 9 for the standard model. For Vibravox (Figure 6.5), the difference is more pronounced: TOL stabilizes performance by approximately epoch 140, while the baseline continues to exhibit fluctuations until around epoch 185.

TOL helps to shorten the blank-token-dominant phase by guiding the model toward "peaking phase" predictions earlier in training. This is achieved through the initial emphasis on coarse-grained phonetic categories (e.g., vowels vs. voiced consonants vs. voiceless consonants), which are easier to learn and align than the full fine-grained phoneme inventory. As training progresses, TOL gradually shifts focus to the finer-grained phoneme prediction task, enabling the model to build upon its foundational knowledge. TOL acts as an alignment facilitator, helping the model transition out of the blank-dominated regime earlier, leading to faster convergence and more reliable performance gains.

6.3.3 Optimization stability

To assess the impact of TOL on optimization dynamics, we analyze two key indicators of training stability: gradient norm and gradient variance. These metrics provide insights into how smoothly the model traverses the loss landscape. The norm of the gradient vector across all model parameters quantifies the magnitude of the update signal at a given training step. A large norm indicates a steep slope in the loss landscape suggesting large parameter updates (potentially causing instability or "exploding gradients"), whereas a small norm signifies a flatter region, which might indicate convergence is near or that the optimization is suffering from "vanishing gradients". Gradient variance reflects the consistency of the gradient direction estimated from different subsets of the training data. High variance implies noisy or conflicting update directions between batches, potentially hindering convergence, while low variance suggests a more stable and consistent optimization path.



(a) (b)

FIGURE 6.6: Optimization dynamics on the LnNor dataset (Polish split). The figure compares the training dynamics of Transitional Objective Learning (TOL) against a standard baseline by plotting (a) the Gradient Norm and (b) the Gradient Variance at each training step. The results demonstrate that the TOL curriculum leads to a more stable optimization process, maintaining both a significantly lower gradient norm and a lower gradient variance compared to the standard approach.

Figures 6.6(a) and 6.6(b) illustrate that both the gradient norm and gradient variance are consistently lower in the TOL training regime compared to standard training. This suggests that the gradual transition of loss objectives in TOL contributes to a more tempered and stable optimization process. The stability likely contributes to the improved convergence characteristics and performance observed in our experiments. In particular, the gradient variance is an order of magnitude smaller when using TOL. This small variance of gradients across mini batches of training data is, in our opinion, the main reason for the substantially improved performance of the TOL framework.

6.3.4 Ablation study

To determine whether TOL's effectiveness stems from its hierarchy rather than merely providing a regularizing effect, we conducted an ablation study. We constructed an auxiliary loss identical in form to the tested TOL setup, but with phonemes randomly assigned to three arbitrary classes instead of linguistically motivated categories (vowels, voiced consonants, voiceless consonants). We compared the performance of this randomized-hierarchy model with both standard training and TOL setup. Results show that the random grouping consistently underperforms both standard training and TOL setup, with average PER of 0.206 for Vibravox, 0.092 for LnNor, and 0.053 for TIMIT, indicating that the gains from TOL are not due to auxiliary loss alone. Instead, they result from the meaningful structure of the hierarchy, which plays a crucial role in enhancing model performance.

6.4 Conclusions

This chapter introduced Transitional Objective Learning, a novel framework designed to enhance the training of complex machine learning models by leveraging the principles of curriculum learning. By situating TOL within the formal CL framework, we can depict its components and evaluate its contributions to the field. TOL is a form of task-space curriculum that manipulates the learning objective itself rather than reordering data instances. It operates on the core CL components as follows:

- **Scoring Function** TOL uses a static, heuristic-based scoring function. The "difficulty" is assigned not to individual data points, but to the learning objectives themselves. This

work applied TOL to the specific domain of ASR by leveraging a natural hierarchy inherent to the task of phoneme recognition. Based on the linguistic characteristics of phonemes, distinguishing broad phonetic categories (e.g., vowels vs. consonants) is defined as an "easy" task, while recognizing specific, fine-grained phonemes is a "hard" task.

- **Pacing Function** TOL uses a fixed pacing function to schedule the transition between these tasks. The total loss is a weighted sum of the coarse-grained and fine-grained loss functions, and the weights are controlled by a pre-defined schedule (e.g., a sigmoid function) that is a function of the training epoch. This gradually shifts the model's focus from the easy objective to the hard one over time.

The experimental results presented in this chapter demonstrate that TOL successfully fulfills several of the key promises of curriculum learning.

- **Performance improvement** Across diverse phoneme recognition tasks in English, Polish, and French, TOL consistently achieved statistically significant reductions in Phoneme Error Rate compared to standard training.
- **Convergence improvement** Models trained with TOL converged to their optimal performance significantly faster than the baseline models. For instance, on the TIMIT dataset, the TOL model converged in approximately half the number of epochs required by the standard approach. Additionally, by analyzing the training dynamics, we observed that TOL leads to a more stable optimization process. Both the gradient norm and, most notably, the gradient variance were consistently lower when training with TOL, which we attribute as a primary reason for its improved performance.

This work has systematically explored a specific region of the curriculum learning design space: a task-driven curriculum with static scoring and a fixed, continuous pacing function. While we have demonstrated the efficacy of this approach, this opens up several exciting research avenues for future studies. One clear direction is to explore more complex hierarchies by introducing additional intermediate tasks and corresponding loss functions, such as differentiating vowels based on their front/back position before moving to the final phoneme set. Another promising area is the development of an adaptive pacing function for TOL. Instead of a pre-defined schedule, the transition between objectives could be triggered by the model's real-time competence, for example, by monitoring the validation loss or the stability of the model's representations for the current coarse-grained task. Integrating such adaptive mechanisms could lead to an even more efficient and robust learning framework, further validating the power of structured, hierarchical curricula.

Chapter 7

Conclusions

7.1 Research synthesis

This thesis was motivated by a paradox in the field of machine learning: the wide gap between the intuitive appeal of curriculum learning and its inconsistent influence on model performance in practice. While the principle of learning from a structured sequence of examples, mastering simple concepts before moving to complex ones, is a basis of human pedagogy, its translation into a standard tool for training artificial neural networks faced many challenges. The central aim of this work, therefore, has been to move beyond the collection of ad-hoc methods that characterize much of the CL literature and establish a more principled, analytical foundation for the field. The research presented here represents a systematic effort to understand not just *if* a curriculum works, but *why* and *how* it influences the fundamental dynamics of neural network optimization. This thesis was structured to build this understanding progressively, with each chapter addressing a key component of the curriculum design problem.

The investigation began in Chapter 2 with the establishment of a formal framework, which served as the theoretical base for all subsequent empirical work. This framework deconstructed the abstract concept of a curriculum into two concrete, modular components: a scoring function, which assigns a measure of suitability (e.g., difficulty, typically, informativeness) to data or tasks, and a pacing function, which governs the rate and manner of their introduction during training. By providing this common vocabulary and distinguishing between data-driven and task-driven curricula, we enabled a structured analysis of curriculum learning strategies. Chapter 3 served as a critical diagnostic study into a common failure mode of naive CL implementations. The investigation into a static curriculum based on typicality scoring revealed a counterintuitive and detrimental outcome: ordering data from most to least typical within the epoch resulted in worse performance than standard random shuffling. The analysis identified the root cause as the creation of highly homogenous mini-batches, which introduced destabilizing, high-variance gradient noise late in each training epoch, ultimately harming the model's ability to converge to a robust solution. This CL failure was a pivotal illustration of mixed results often reported in the CL literature and motivated a shift away from static notions of difficulty and batch composition. In direct response to this finding, Chapter 4 proposed and validated a successful alternative: a dynamic, model-aware scoring function based on the Variance of Gradients (VoG). This marked a significant conceptual shift, redefining "difficulty" not as a fixed property of the data, but as an adaptive measure of the model's own uncertainty or informativeness. By prioritizing examples for which the model's reasoning was most unstable, VoG-Guided Learning (VGL) demonstrated consistent improvements in both convergence speed and final performance across diverse tasks, establishing the superiority of

dynamic, model-aware scoring. In Chapter 5 the investigation returned to the critical component of pacing. This chapter conducted an analysis of batch orchestration strategies, reframing the pacing function as a powerful tool for directly manipulating the dynamics of optimization. The experiments provided an explanation for the failure observed in Chapter 3, while also demonstrating that more sophisticated pacing schedules, such as those involving cyclical batch size changes, can effectively mitigate common optimization challenges like the generalization gap associated with large-batch training. Finally, having thoroughly explored the design space of data-driven curricula, Chapter 6 expanded the scope of the investigation to demonstrate the broad applicability of CL principles. It introduced Transitional Objective Learning (TOL), a novel task-driven curriculum that applies the "start simple" paradigm not to the data, but to the learning objective itself. By guiding the model through a hierarchy of tasks from coarse-grained to fine-grained, TOL achieved significant performance gains in the challenging domain of automatic speech recognition.

7.2 Revisiting the aims

This section revisits the specific aims and original contributions outlined in the Chapter 1. The Table 7.1 provides a concise summary, mapping each stated objective to the chapters and the key findings that substantiate it.

Aim/ Contribution	Key findings	Chapters
A1 / **C1**	Establishment a formal framework for CL; deconstruction of CL into scoring and pacing functions; distinction between data-driven and task-driven curricula.	2
A2 / **C2**	Redefinition of "difficulty" with novel scoring functions. Failure of static typicality scoring. Success of dynamic scoring in improving convergence and performance by prioritizing informative, high-uncertainty examples.	3, 4
A2 / **C4**	Analysis of curriculum-induced learning dynamics. Batch orchestration as a pacing function can mitigate the generalization gap. Cyclical batch schedules act as a powerful regularizer. VGL improves gradient quality, not just adds noise.	4, 5, 6
A2 / **C3**	A novel task-driven curriculum, TOL, that improves PER, convergence, and stability in phoneme recognition by creating a curriculum of objectives.	6
- / **C5**	Democratization of research assets. Publication of 5 datasets and 3 models on Hugging Face, enabling reproducibility and community impact.	4, 6

TABLE 7.1: Synthesis of research aims and contributions. The table provides a summary mapping each of the thesis's stated aims and original contributions to the corresponding chapter(s) where they were investigated. It highlights the key findings that substantiate each contribution.

The first aim (**A1**) of this thesis was to bring structure to the diverse and often disconnected landscape of curriculum learning research. This objective was met in Chapter 2 through the establishment of a formal, unified framework that deconstructs the design of any curriculum into its fundamental components. By formally defining the scoring function and the pacing function, this framework provided an essential analytical lens through which all subsequent investigations were conducted. This formalization was not only a literature review; it was the creation of a scientific toolset. It provided a common vocabulary that enabled a structured comparison of methods, from static, heuristic-based approaches to dynamic, model-aware strategies. This framework directly

fulfilled the first major aim of the thesis: synthesizing existing knowledge and enabling a more rigorous and principled analysis of curriculum strategies.

A central aim (**A2**) of this thesis was to move beyond simply measuring final accuracy and conduct a thorough empirical investigation into the causal mechanisms that determine a curriculum's success or failure. This was achieved through a multi-chapter analysis of a suite of optimization metrics that revealed *how* and *why* different curricula alter the fundamental mechanisms of neural network optimization. This deep dive into learning dynamics was progressively developed in Chapters 4, 5 and 6. In Chapter 4, with VoG-Guided Learning (VGL), we redefined "difficulty" as a dynamic measure of the evolution of model uncertainty. The success of VGL was evidenced not just by higher accuracy but also by its impact on convergence dynamics; it consistently reached optimal performance faster than the baseline by actively reducing the overall uncertainty across the dataset, as measured by the mean VoG score. This established that an effective curriculum could directly accelerate the learning process by focusing on the most informative examples. Chapter 5 extended this analysis by framing the pacing function as a tool for the direct manipulation of optimization dynamics, with a focus on gradient variance and the gradient noise scale. This chapter provided an insight that VGL improves training not by injecting more noise, but by improving the quality of the gradient signal itself, resulting in a surprisingly lower gradient noise scale. Furthermore, the analysis of cyclical batch size schedules showed how intelligent pacing could act as a powerful regularizer to escape poor local minima and mitigate the generalization gap associated with large-batch training. Finally, Chapter 6 demonstrated that this principled analysis of learning dynamics could be generalized from the data space to the task space. The success of Transitional Objective Learning (TOL) was substantiated not only by a lower final error rate but by its profound effect on optimization rate and stability. The analysis revealed that the TOL curriculum led to a more stable training process, characterized by a consistently lower gradient norm and an order-of-magnitude lower gradient variance compared to the standard baseline. Additionally, TOL allowed to move from "suppression phase" to "peaking phase" (phenomena associated with connectionist temporal classification) faster than standard methods. This provided clear evidence that a curriculum of objectives could create a smoother, faster, and more reliable optimization path. Together, these chapters fulfill **A2** of the thesis. By moving beyond the model's final accuracy to analyze a suite of underlying metrics, these chapters provided insights into how curricula function as powerful meta-learning strategies for controlling the optimization process.

7.3 Other research paths

Beyond the direct improvements to model training, the principles of accelerated convergence demonstrated in this thesis were also explored as a potential solution to the computationally expensive problem of hyperparameter optimization. The core idea, which we term Transfer Hyperparameter Learning, was to leverage a fast, curriculum-based training run as a proxy to predict the final performance of a longer, standard training run, thereby accelerating the evaluation of different hyperparameter configurations. While this research path was ultimately secondary to the main contributions of this thesis, a full description of the methodology, experimental setup, and a detailed discussion of the promising but nuanced results are provided for the interested reader in Appendix A.

7.4 Broader implications and limitations

The findings of this thesis carry several broader implications for the theory and practice of machine learning, while also acknowledging the inherent limitations of the research. The results presented in this work advocates for a paradigm shift in how curriculum learning is conceptualized. It should be viewed not as a static, data pre-processing technique aimed at finding a single "perfect" data order, but rather as a dynamic meta-learning strategy that adapts to the model state for controlling the optimization process. The success of dynamic methods like VGL and cyclical batch scheduling suggests that the most promising path for practitioners lies not in the search for a universal curriculum, but in the design of mechanisms that actively tailor the data or task sequence in response to the model's evolving state. The detailed analysis of learning dynamics provides actionable insights that challenge common assumptions.

However, the applicability of curriculum learning is bounded by several important limitations. The most significant practical challenge is the computational overhead associated with constructing curricula. As discussed in Chapter 4, the need to compute and store input gradients for the entire training dataset at the end of each epoch presents a substantial barrier to scalability for industrial-scale problems. While strategies like using a sliding window were employed to make the approach tractable, this remains a key area for future optimization. Furthermore, the scope of the investigation was limited to supervised deep neural network training within the domains of computer vision, natural language processing, and speech recognition. While the core principles are likely generalizable, the specific efficacy of the proposed methods (VGL, TOL, and the batch orchestration strategies) in other machine learning paradigms, such as reinforcement learning, generative modeling, or unsupervised and self-supervised learning, remains an open and important question for future research. Finally, while this work explored several pacing functions, their design often introduces new hyperparameters that themselves require careful tuning. For example, the cycle length and batch size range in the cyclical schedule from Chapter 5, or the specific shape of the sigmoid transition in TOL (Chapter 6), are critical design choices. This thesis stops short of proposing fully automated, hyperparameter-free pacing functions, representing a frontier for future work in making advanced CL methods more robust and accessible for machine learning practitioners.

7.5 Future research directions

The conclusions and limitations of this thesis naturally give rise to several promising avenues for future research, aimed at building upon the contributions made here and addressing the open questions that remain. A primary direction for future work is to directly tackle the identified limitations. The most pressing of these is the development of scalable curriculum learning strategies. Another exciting avenue lies at the intersection of the insights from Chapters 5 and 6: the development of adaptive pacing for task-driven curricula. The implementation of TOL in this thesis relied on a fixed, pre-defined sigmoid schedule to transition between objectives. A natural extension would be to replace this with an adaptive pacing function. For instance, the transition from the coarse-grained to the fine-grained objective could be triggered dynamically based on the model's real-time competence, such as when the validation loss on the coarser task plateaus. This would create a truly self-paced, task-driven learning framework. Furthermore, the ablation study in Chapter 5, which highlighted the critical need to co-adapt the learning rate with the batch size, points toward a broader research agenda on the synergistic orchestration of hyperparameters. Future work should investigate the complex interplay between the data curriculum, batch

size schedules, and learning rate schedules. Developing algorithms that can dynamically manage all three components of optimization could lead to fully automated and highly efficient training strategies.

7.6 Closing

This thesis has endeavored to demonstrate that by moving from heuristic-based approaches to a principled, analytical study of learning dynamics, curriculum learning can be transformed into a powerful optimization tool. The central finding is that through the deliberate and adaptive control of what a model learns and when, we can create more efficient, robust, and ultimately more accurate machine learning systems. The final contribution of this work, the democratization of the datasets and models developed during this research via public release, is offered not only as a deliverable but also as a commitment to open and reproducible science that is essential for continuing the journey toward a better understanding of deep neural network training optimization.

Appendix A

Transfer hyperparameter learning

This appendix details a research direction that was explored during the course of this PhD but was ultimately abandoned in favor of the primary research paths detailed in the main body of this thesis. The work centered on addressing the significant computational burden of hyperparameter optimization by leveraging the principles of curriculum learning.

A.1 Motivation

The majority of deep neural networks, if not all, depend on a wide range of hyperparameter choices such as network's architecture, batch size, regularization and optimization. The configuration is often hand-tuned in the course of evaluating a model on a particular dataset as scientists would rather rely on their knowledge and intuition. Since the configuration is a result of skewed, biased and hard-to-describe in a systematic way process, the performance of hand-tuned models might be incidental. In other words, the model's reported performance may not be a reliable indicator of its true capabilities. Such a result often fails to generalize to new, unseen data, and the lack of a systematic search process means that a significantly better-performing configuration may have been easily missed. This makes the findings difficult to reproduce on different datasets, undermining the scientific validity of the results. Consequently, over the past few decades, scientists developed a range of automated hyperparameter search methods: grid search, Hyperband L. Li et al., 2017, Fabolas Klein et al., 2017, Population-based training Jaderberg et al., 2017, just to name a few.

Despite recent success in developing efficient, automated hyperparameter optimization methods, there are still some issues which have not been directly tackled. The upsurge in the variety and complexity of machine learning methods studied today might even require a whole new approach to hyperparameter optimization. The main reason behind that is the scale of conducting hyperparameter optimization on a complex model and massive datasets which exceed the computational possibilities we have. It is a result of two factors:

1. the large size of a parameter search space (especially in the case of continuous space), and
2. the high cost of a single model evaluation.

In the result, many solutions, such as models trained on ImageNet J. Deng et al., 2009 or CelebA Z. Liu et al., 2018, still remain unstudied in terms of hyperparameter optimization because of the expense of training an individual model on these datasets. This, in turn, undermines the reproducibility and reliability of conducted studies as manual tuning is far less reproducible than automated tuning. Benchmarks can be reliable only if we can conduct a fair comparison of models. However, a fair comparison requires the same level of hyperparameters optimization for each model.

This raises demand for novel hyperparameter approaches development that would search a space of hyperparameters more efficiently.

As the main issue of using automated hyperparameter optimization for large datasets is the long time needed to evaluate an individual model configuration, we should focus our efforts on minimizing the time of an individual model configuration evaluation. We showed that curriculum learning strategies can greatly accelerate convergence especially at the beginning of the training. Thus, one of the potential solutions to address this problem is the usage of curriculum learning.

This appendix tries to tackle some of the aforementioned hyperparameter optimization challenges. In Chapters 4, 5, 6 we showed that well-defined curriculum learning strategy can accelerate convergence. The idea is then to limit the time needed to evaluate an individual hyperparameter configuration by applying curriculum learning. We hypothesize that we can speed up hyperparameter space search using curriculum learning by assuming that better loss on the model trained with curriculum learning (given a hyperparameter configuration) leads to better loss on the traditionally trained model (given the same hyperparameter configuration). We name this process transfer hyperparameter learning.

A.2 Method

The central premise of transfer hyperparameter learning is to use a curriculum to rapidly evaluate the quality of a given hyperparameter configuration. To operationalize this, we must first identify the curriculum strategy that offers the most significant acceleration in convergence, as this will provide the greatest time savings. Therefore, our methodology begins by quantitatively comparing the convergence rates of different training strategies.

To formalize the notion of "convergence rate," we propose using the Lipschitz constant of the learning curve as a proxy. In mathematics, the Lipschitz constant bounds the maximum rate at which a function's output can change in response to changes in its input. It is therefore a principled measure of a function's sensitivity or steepness. First, let us define Lipschitz continuity. A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is globally Lipschitz continous on $\chi \subseteq \mathbb{R}^n$ if there exists a positive value K such that:

$$|f(x) - f(y)| \leq K|x - y|, \forall x, y \in \chi \tag{A.1}$$

The smallest K is known as the Lipschitz constant of the function f.

In our analysis, we treat the model's accuracy on the test set as a function of the training epoch, t. Thus, $f(\theta^t)$ denotes accuracy of model θ at epoch t. The input space is the set of training epochs, $\mathbf{t}$, and the output space is the model's performance. We utilized the Lipschitz constant to assess the speed and rapidness of model convergence, calculating K for each training configuration by finding the maximum observed rate of change in accuracy between any two points in the training process:

$$K = max_{i,j \in \mathbf{t}} \frac{|f(\theta^{(i)}) - f(\theta^{(j)})|}{|i - j|} \tag{A.2}$$

In this context, a higher calculated Lipschitz constant corresponds to a steeper learning curve and thus a more rapid convergence rate. The curriculum strategy that consistently yields the highest value for K will be selected as the most suitable candidate for accelerating the hyperparameter search in our main experiment.

Let $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_n]$; $\lambda \sim \mathcal{H}$ be a set of model's hyperparameters drawn from hyperparameter set space $\mathcal{H}$. Let $f(\theta^{\lambda(i)})$ be the performance of model θ trained with hyperparameter set λ in a standard training mode at epoch i. Let $f(\theta^{\Upsilon\lambda(i)})$ be the performance of model θ trained with

hyperparameter set λ and curriculum learning strategy Υ. There are two research questions we want to ask:

RQ1 Is the accelerated convergence offered by a curriculum learning strategy, Υ, a universal property across the hyperparameter space $\mathcal{H}$? Specifically, for a given, limited training budget (e.g., a small number of epochs, i), does the performance of a curriculum-trained model consistently exceed that of a standard-trained model, such that $f(\theta^{\Upsilon\lambda(i)}) > f(\theta^{\lambda(i)})$ for a given hyperparameter configuration $\lambda \sim \mathcal{H}$?

RQ2 Most critically, can the performance ranking of hyperparameter configurations observed under a time-constrained, curriculum-based evaluation serve as a reliable proxy for the performance ranking under a full, standard training regime? Formally, if we have two hyperparameter sets, λ^m and λ^n, and observe that a curriculum-trained model performs better with the first set after a small number of epochs i: $f(\theta^{\Upsilon\lambda^{\mathbf{m}}(i)}) > f(\theta^{\Upsilon\lambda^{\mathbf{n}}(i)})$, does this imply that a fully-trained standard model will also perform better with that same configuration λ: $f(\theta^{\lambda^{\mathbf{m}}(j)}) > f(\theta^{\lambda^{\mathbf{n}}(j)})$ where $i + j$ is the peak performance epoch? In other words, is the implication:

$$f(\theta^{\Upsilon\lambda^{\mathbf{m}}(i)}) > f(\theta^{\Upsilon\lambda^{\mathbf{n}}(i)}) \Rightarrow \ f(\theta^{\lambda^{\mathbf{m}}(i+j)}) > f(\theta^{\lambda^{\mathbf{n}}(i+j)}) \tag{A.3}$$

true?

Before conducting experiments, we formulate the following hypotheses that will help us answer the above research questions:

Hypothesis 1 (H1) *Curriculum-based training mode yields better results under limited time constraints than the standard training for majority of hyperparameter configurations:*

$$\forall_{\lambda \in \mathcal{H}'} \ f(\theta^{\Upsilon\lambda(i)}) > f(\theta^{\lambda(i)})$$

Hypothesis 2 (H2) *If configuration λ^m lead to better performance than configuration λ^n when training a model with curriculum learning strategy Υ with a limited number of epochs i then it would also be better in standard training:*

$$f(\theta^{\Upsilon\lambda^{\mathbf{m}}(i)}) > f(\theta^{\Upsilon\lambda^{\mathbf{n}}(i)}) \Rightarrow \ f(\theta^{\lambda^{\mathbf{m}}(i)}) > f(\theta^{\lambda^{\mathbf{n}}(i)})$$

Hypothesis 3 (H3) *There is such an i and j that:*

$$f(\theta^{\Upsilon\lambda^{\mathbf{m}}(i)}) > f(\theta^{\Upsilon\lambda^{\mathbf{n}}(i)}) \Rightarrow \ f(\theta^{\lambda^{\mathbf{m}}(i+j)}) > f(\theta^{\lambda^{\mathbf{n}}(i+j)})$$

A.3 Experiments and results

The experiments were conducted on the four tabular datasets: Parkinson Dua and Graff, 2017d, House Votes Dua and Graff, 2017c, CarsDua and Graff, 2017a, and Credit Screening Dua and Graff, 2017b. Each training configuration was repeated 50 times to ensure statistical robustness.

A.3.1 Experiment 1: Measuring convergence rate

Our first experiment aims to identify the curriculum strategy with the most rapid convergence rate, as this is the best candidate for accelerating the hyperparameter search. We tested three training setups described in Chapter 5, Section 5.2.1:

- single shuffle,

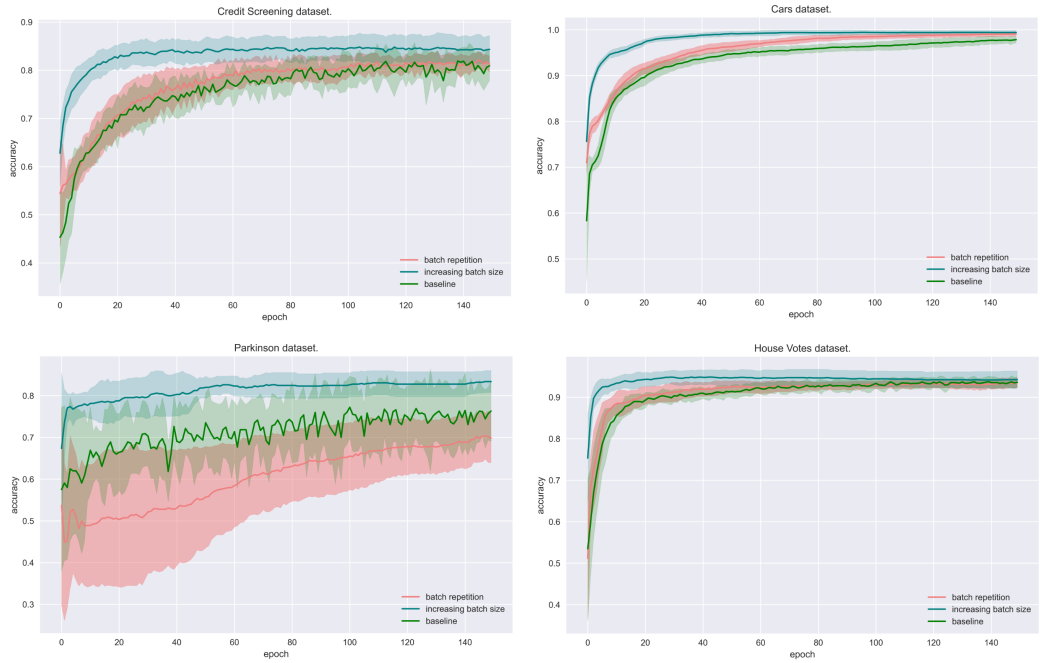


FIGURE A.1: Learning curves comparing batch orchestration strategies across four tabular datasets. The figure displays the validation accuracy over 150 epochs for three training strategies: the baseline (green), batch repetition (red), and increasing batch size (blue). Each panel corresponds to a different dataset: Credit Screening, Cars, Parkinson, and House Votes. The results visually confirm the findings from our Lipschitz constant analysis: the increasing batch size strategy consistently demonstrates the fastest convergence and achieves the highest final accuracy across all four datasets.

- batch repetition on typicality-ordered data (with each batch repeated $j = 3$),
- increasing batch size (with initial batch size $b = 32$, and step $m = 4$) on typicality-ordered data.

K	Single Shuffle	Batch repetition	Increasing batch size
Parkinson	0.6424	0.7852	**0.8455**
House Votes	0.2802	0.2976	**0.4289**
Cars	0.3144	0.3287	**0.6531**
Credit Screening	0.2938	0.2770	**0.7268**

TABLE A.1

Each experiment was run for 150 epochs. We quantified the convergence rate using the Lipschitz constant of the learning curve, where a higher value indicates a more rapid improvement in model performance. The Table A.1 presents the average Lipschitz constants computed for each method on the four datasets. The results clearly show that the most rapid changes were reported for the increasing batch size method, which yielded the highest Lipschitz constant in all four settings. The batch repetition method also outperformed the baseline in three of the four datasets. Although the final performance benefits marginally from the curriculum after long training, this analysis confirms that the most significant improvement is visible in the first stages of the training. The results confirm that a curriculum with a proper pacing function can indeed improve performance under a limited time budget. Based on these results, we selected the increasing batch size strategy as the curriculum strategy Υ for all subsequent experiments.

A.3.2 Experiment 2: Performance under a limited time budget

This experiment was designed to test Hypothesis 1, that the curriculum-based training is superior to standard training under tight time constraints. We conducted a pairwise comparison of the average model accuracy for the increasing batch size curriculum versus the standard training. We evaluated the performance at three early stopping points: after $i \in \{5, 10, 15\}$ epochs. For each duration, we calculated the percentage of the 20 hyperparameter configurations for which the curriculum-trained model outperformed the standard model.

Win %	$i = 5$	$i = 10$	$i = 15$
Parkinson	70	70	75
House Votes	85	85	70
Cars	100	90	85
Credit Screening	100	100	90

TABLE A.2: Curriculum win percentage under limited time budgets. The table shows the percentage of 20 hyperparameter configurations for which the curriculum model (increasing batch size) outperformed the standard baseline at three early stopping points (5, 10, and 15 epochs). The results demonstrate the robust advantage of the curriculum strategy, which is most pronounced at the earliest stages of training.

The results summarized in Table A.2 provide support for our hypothesis. The curriculum-based approach demonstrated a clear and consistent advantage, outperforming the standard training method for a majority of hyperparameter configurations at all time thresholds. The performance gain is most pronounced at the earliest evaluation points. This evidence strongly supports Hypothesis 1, confirming that under a tight time budget, the curriculum strategy is a more robust and efficient choice for achieving higher performance, largely independent of the specific hyperparameter settings.

A.3.3 Experiment 3: Testing the transferability of hyperparameter rankings

This experiment tests the central premise of transfer hyperparameter learning (Hypothesis 2): whether the performance ranking of hyperparameters is preserved between the curriculum and standard training modes. To measure this relationship, we calculated the Spearman's rank correlation coefficient ρ between the performance of the 20 hyperparameter configurations when trained with the curriculum versus with standard training. We created rankings across four datasets and four time budgets i.

The results, summarized in the Table A.3, show a significant and high correlation between the training modes for three out of the four datasets, providing strong evidence in favor of our hypothesis. Specifically, after 30 epochs, we observed high correlation coefficients of 0.825, 0.851, and 0.784 for the Parkinson, Cars, and Credit Screening datasets, respectively. In contrast, we did not observe a meaningful correlation for the House Votes dataset. We suspect the reason for this discrepancy is the relatively small impact of the curriculum on this specific dataset. Based

ρ	$i = 5$	$i = 10$	$i = 15$	$i = 30$
Parkinson	0.269	0.485	0.436	0.825
House Votes	0.306	0.316	0.235	0.255
Cars	0.849	0.834	0.83	0.851
Credit Screening	0.683	0.854	0.782	0.784

TABLE A.3: Correlation of hyperparameter rankings. The table shows the Spearman's rank correlation coefficient ρ between the performance rankings of 20 hyperparameter configurations, comparing a short curriculum-based training run against a standard training run at four time steps (5, 10, 15, and 30 epochs). The results show a strong correlation for the Parkinson, Cars, and Credit Screening datasets, supporting the hypothesis that hyperparameter rankings are transferable. However, no meaningful correlation was observed for the House Votes dataset.

on the results, we can conclude that the resulting hyperparameter rankings are transferable to the standard training regime.

A.3.4 Experiment 4: Predicting future performance with early results

This experiment investigates the predictive power of early curriculum performance on the later performance of standard training, directly testing Hypothesis 3. The goal is to determine if a short, computationally cheap training run with a curriculum can be used to reliably forecast the outcome of a much longer, more expensive run using standard training. To achieve this, we calculated the Spearman's rank correlation coefficient ρ between the performance rankings of the 20 hyperparameter configurations from a curriculum run at an early epoch i and a standard training run at a later epoch $i + j$.

ρ	$i = 5, j = 25$	$i = 10, j = 20$	$i = 15, j = 15$
Parkinson	0.81	0.857	0.659
House Votes	0.477	0.434	0.368
Cars	0.843	0.874	0.871
Credit Screening	0.817	0.809	0.799

TABLE A.4: The table shows the Spearman's rank correlation coefficient ρ between the hyperparameter rankings of a short curriculum run (at epoch i) and a longer standard run (at epoch $i + j$). The high correlation for most datasets validates that early performance in a curriculum is a strong predictor of final performance in a standard training regime.

The results, summarized in the Table A.4, reveal a high correlation between the rankings in the majority of experimental settings, providing support for Hypothesis 3. For the Parkinson, Cars, and Credit Screening datasets, the performance of the curriculum-trained models after only 5 or 10 epochs is a strong predictor of the final performance of the standard-trained models after 30 epochs, with correlation coefficients consistently exceeding 0.8. While the correlation for the House Votes dataset was more moderate, the overall trend demonstrates that a short run with an effective curriculum can be used to reliably forecast the outcome of a much longer hyperparameter search, validating the core predictive power of the transfer hyperparameter learning concept.

A.3.5 Experiment 5: Establishing a baseline for early stopping

Finally, to serve as a control, we measured how well a short run of standard training can predict the final hyperparameter ranking of a longer standard training run. This experiment establishes a baseline for predictive power, allowing us to determine if the curriculum-based approach offers a genuine advantage over simply running a standard hyperparameter search for a shorter amount of time. To do this, we calculated the Spearman's rank correlation coefficient ρ between the performance rankings at an early epoch i and later at epoch 30.

ρ	$i = 5, j = 25$	$i = 10, j = 20$	$i = 15, j = 15$
Parkinson	0.632	0.833	0.824
House Votes	0.804	0.81	0.864
Cars	0.918	0.938	0.962
Credit Screening	0.64	0.793	0.945

TABLE A.5: The table shows the Spearman's rank correlation coefficient ρ between the hyperparameter rankings of a short standard run (at epoch i) and a long standard run (at epoch 30). The results establish a strong baseline, showing that the performance ranking within standard training begins to stabilize and become highly predictive after 10-15 epochs.

The results, presented in the Table A.5, show that the predictive power of standard training is initially moderate but becomes very high as the training process matures. As expected, the

correlation increases as the initial evaluation epoch i gets closer to the final epoch. By the time the standard model has trained initially for 10 epochs, its own performance ranking is already a strong predictor of the ranking at epoch 30 for most datasets. When comparing these results to the predictive power of the curriculum (Experiment 4), we observe a nuanced trade-off. At the very earliest checkpoints ($i = 5$), the curriculum provides a better forecast for some datasets, but this advantage diminishes as training continues. This suggests that while our curriculum-based transfer hyperparameter learning offers a distinct benefit for extremely short evaluation budgets, a simple early-stopping approach with standard training becomes a competitive baseline if the evaluation budget allows.

A.4 Conclusions

In conclusion, this exploratory study provides promising evidence for the concept of transfer hyperparameter learning. For the majority of the datasets tested, we found a strong and significant correlation between the hyperparameter rankings generated by a short, curriculum-accelerated training run and the final rankings from a much longer, standard training run. This suggests that for complex learning problems, an effective curriculum can indeed serve as a reliable proxy to reduce the time required for hyperparameter optimization.

However, our findings also highlight a critical limitation: the utility of this method is highly contingent on the convergence characteristics of the dataset in question. The House Votes dataset serves as a clear counterexample. On this dataset, the baseline model achieved convergence very rapidly, typically within 20-30 epochs. Consequently, using a 15-epoch curriculum run to predict the ranking at epoch 30 provides little advantage, as a standard run of a similar length is already a strong predictor of its own final outcome. This undermines the usefulness of the curriculum for this case, and suggests that if we were to compare rankings for epochs higher than 30 for the rest of the datasets, the correlation would likely degrade even further.

Therefore, while transfer hyperparameter learning shows considerable potential for accelerating the search on complex problems with slow convergence, its benefits are marginal for simpler tasks where a standard early-stopping baseline is already an efficient and effective strategy.

Bibliography

Adam, K. D. B. J., et al. (2014). A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, *1412*(6).

Agarwal, C., D'souza, D., & Hooker, S. (2022). Estimating example difficulty using variance of gradients. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10368–10378.

Alain, G., Lamb, A., Sankar, C., Courville, A., & Bengio, Y. (2015). Variance reduction in sgd by distributed importance sampling. *arXiv preprint arXiv:1511.06481*.

Allgower, E. L., & Georg, K. (2012). *Numerical continuation methods: An introduction* (Vol. 13). Springer Science & Business Media.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. *International conference on machine learning*, 173–182.

Ao, S., Rueger, S., & Siddharthan, A. (2023). Confidence-aware calibration and scoring functions for curriculum learning. *Fifteenth International Conference on Machine Vision (ICMV 2022)*, *12701*, 558–567.

Ash, J., & Adams, R. P. (2020). On warm-starting neural network training. *Advances in neural information processing systems*, *33*, 3884–3894.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. *Proceedings of the 26th annual international conference on machine learning*, 41–48.

Benjamin, A. S., Rolnick, D., & Kording, K. (2019). Measuring and regularizing networks in function space. https://arxiv.org/abs/1805.08289

Bluche, T., Kermorvant, C., Ney, H., & Louradour, J. (2016). La ctc et son intrigant label «blank»: Étude comparative de méthodes d'entrainement de réseaux de neurones pour la reconnaissance d'écriture. *CORIA-CIFED*, 39–54.

Braun, S., Neil, D., & Liu, S.-C. (2017). A curriculum learning method for improved noise robustness in automatic speech recognition. *2017 25th European Signal Processing Conference (EUSIPCO)*, 548–552.

Byrd, R. H., Chin, G. M., Nocedal, J., & Wu, Y. (2012). Sample size selection in optimization methods for machine learning. *Mathematical programming*, *134*(1), 127–155.

Chang, H.-S., Learned-Miller, E., & McCallum, A. (2017). Active bias: Training more accurate neural networks by emphasizing high variance samples. *Advances in Neural Information Processing Systems*, *30*.

Chaudhry, S., & Sharma, A. (2024). Data distribution-based curriculum learning. *IEEE Access*.

Chen, G., Faris, P., Hemmelgarn, B., Walker, R. L., & Quan, H. (2009). Measuring agreement of administrative data with chart data using prevalence unadjusted and adjusted kappa. *BMC medical research methodology*, *9*(1), 5.

Chen, Y., Li, Y., Hu, K., Ma, Z., Ye, H., & Chen, K. (2025). Mig: Automatic data selection for instruction tuning by maximizing information gain in semantic space. *arXiv preprint arXiv:2504.13835*.

Choi, J., Jeong, M., Kim, T., & Kim, C. (2019). Pseudo-labeling curriculum for unsupervised domain adaptation. *arXiv preprint arXiv:1908.00262*.

Clemmensen, L. H., & Kjærsgaard, R. D. (2022). Data representativity for machine learning and ai systems. *arXiv preprint arXiv:2203.04706*.

Coleman, C., Yeh, C., Mussmann, S., Mirzasoleiman, B., Bailis, P., Liang, P., Leskovec, J., & Zaharia, M. (2019). Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*.

Dempster, F. N. (1989). Spacing effects and their implications for theory and practice. *Educational Psychology Review*, *1*(4), 309–330.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*, 248–255.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, *29*(6), 141–142.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. https://arxiv.org/abs/1810.04805

Dua, D., & Graff, C. (2017a). Cars UCI machine learning repository. http://archive.ics.uci.edu/ml

Dua, D., & Graff, C. (2017b). Credit screening UCI machine learning repository. http://archive.ics.uci.edu/ml

Dua, D., & Graff, C. (2017c). House votes UCI machine learning repository. http://archive.ics.uci.edu/ml

Dua, D., & Graff, C. (2017d). Parkinson UCI machine learning repository. http://archive.ics.uci.edu/ml

Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, *48*(1), 71–99.

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological bulletin*, *76*(5), 378.

Friedlander, M. P., & Schmidt, M. (2012). Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, *34*(3), A1380–A1405.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, *32*(200), 675–701.

Garofolo, J. S., Lamel, L. F., Fisher, W. M., Pallett, D. S., Dahlgren, N. L., Zue, V., & Fiscus, J. G. (1993). Timit acoustic-phonetic continuous speech corpus. *(No Title)*.

Goyal, P., & Ghosh, S. (2020). Hierarchical class-based curriculum loss. *arXiv preprint arXiv:2006.03629*.

Graves, A., Bellemare, M. G., Menick, J., Munos, R., & Kavukcuoglu, K. (2017). Automated curriculum learning for neural networks. https://arxiv.org/abs/1704.03003

Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. *Proceedings of the 23rd international conference on Machine learning*, 369–376.

Guo, S., Huang, W., Zhang, H., Zhuang, C., Dong, D., Scott, M. R., & Huang, D. (2018). Curriculumnet: Weakly supervised learning from large-scale web images. *Proceedings of the European conference on computer vision (ECCV)*, 135–150.

Gwiaździński, P., Gundersen, A. B., Piksa, M., Krysińska, I., Kunst, J. R., Noworyta, K., Olejniuk, A., Morzy, M., Rygula, R., Wójtowicz, T., et al. (2023). Psychological interventions

countering misinformation in social media: A scoping review. *Frontiers in psychiatry*, *13*, 974782.

Hacohen, G., & Weinshall, D. (2019). On the power of curriculum learning in training deep networks. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning* (pp. 2535–2544). PMLR. https://proceedings.mlr.press/v97/hacohen19a.html

Hauret, J., Olivier, M., Joubaud, T., Langrenne, C., Poirée, S., Zimpfer, V., & Bavu, É. (2024). Vibravox: A Dataset of French Speech Captured with Body-conduction Audio Sensors. https://arxiv.org/abs/2407.11828

Her, M. B., Jeong, J., Song, H., & Han, J.-H. (2024). Batch transformer: Look for attention in batch. https://arxiv.org/abs/2407.04218

Hinton, G. (2012). Rmsprop: Divide the gradient by a running average of its recent magnitude. https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hoffer, E., Hubara, I., & Soudry, D. (2017). Train longer, generalize better: Closing the generalization gap in large batch training of neural networks. *Advances in neural information processing systems*, *30*.

Hong, G. Z., Cui, Y., Fuxman, A., Chan, S., & Luo, E. (2024). Why fine-grained labels in pretraining benefit generalization? https://arxiv.org/abs/2410.23129

Houlsby, N., Huszár, F., Ghahramani, Z., & Lengyel, M. (2011). Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*.

Hu, D., Wang, Z., Xiong, H., Wang, D., Nie, F., & Dou, D. (2020). Curriculum audiovisual learning. *arXiv preprint arXiv:2001.09414*.

Huang, Y., Wang, Y., Tai, Y., Liu, X., Shen, P., Li, S., Li, J., & Huang, F. (2020). Curricularface: Adaptive curriculum learning loss for deep face recognition. *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5901–5910.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. https://arxiv.org/abs/1502.03167

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., et al. (2017). Population based training of neural networks. *arXiv preprint arXiv:1711.09846*.

Jeong, J., Lee, S., Kim, J., & Kwak, N. (2019). Consistency-based semi-supervised learning for object detection. *Advances in neural information processing systems*, *32*.

Jiang, A. H., Wong, D. L. .-., Zhou, G., Andersen, D. G., Dean, J., Ganger, G. R., Joshi, G., Kaminksy, M., Kozuch, M., Lipton, Z. C., & Pillai, P. (2019). Accelerating deep learning by focusing on the biggest losers. https://arxiv.org/abs/1910.00762

Jiang, L., Meng, D., Zhao, Q., Shan, S., & Hauptmann, A. G. (2015). Self-paced curriculum learning. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2694–2700.

Jiang, L., Zhou, Z., Leung, T., Li, L.-J., & Fei-Fei, L. (2018). Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. *International conference on machine learning*, 2304–2313.

Jiménez-Sánchez, A., Mateus, D., Kirchhoff, S., Kirchhoff, C., Biberthaler, P., Navab, N., González Ballester, M. A., & Piella, G. (2019). Medical-based deep curriculum learning for improved fracture classification. *Medical Image Computing and Computer Assisted Intervention–*

MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part VI 22, 694–702.

Katharopoulos, A., & Fleuret, F. (2018). Not all samples are created equal: Deep learning with importance sampling. *International conference on machine learning*, 2525–2534.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Kim, J., & Lee, J. (2024). Strategic data ordering: Enhancing large language model performance through curriculum learning. https://arxiv.org/abs/2405.07490

Klein, A., Falkner, S., Bartels, S., Hennig, P., & Hutter, F. (2017). Fast bayesian optimization of machine learning hyperparameters on large datasets. *Artificial intelligence and statistics*, 528–536.

Kocmi, T., & Bojar, O. (2017). Curriculum learning and minibatch bucketing in neural machine translation. *arXiv preprint arXiv:1707.09533*.

Krippendorff, K. (2018). *Content analysis: An introduction to its methodology*. Sage publications.

Krysińska, I., Morzy, M., & Kajdanowicz, T. (2021). Curriculum learning revisited: Incremental batch learning with instance typicality ranking. *Artificial Neural Networks and Machine Learning – ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part IV*, 279–291. https://doi.org/10.1007/978-3-030-86380-7˙23

Krysińska, I., Wójtowicz, T., Olejniuk, A., Morzy, M., & Piasecki, J. (2021). Be careful who you follow: The impact of the initial set of friends on covid-19 vaccine tweets. *Proceedings of the 2021 Workshop on Open Challenges in Online Social Networks*, 1–8.

Kumar, M., Packer, B., & Koller, D. (2010). Self-paced learning for latent variable models. *Advances in neural information processing systems*, *23*.

Kunst, J. R., Gundersen, A. B., Krysińska, I., Piasecki, J., Wójtowicz, T., Rygula, R., van der Linden, S., & Morzy, M. (2024). Leveraging artificial intelligence to identify the psychological factors associated with conspiracy theory beliefs online. *Nature Communications*, *15*(1), 7497.

Lapedriza, A., Pirsiavash, H., Bylinskii, Z., & Torralba, A. (2013). Are all training examples equally valuable? *arXiv preprint arXiv:1311.6510*.

Lee, J., & Watanabe, S. (2021). Intermediate loss regularization for ctc-based speech recognition. *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6224–6228.

Lee, W., Im, S., Do, H., Kim, Y., Ok, J., & Lee, G. G. (2025). Dypcl: Dynamic phoneme-level contrastive learning for dysarthric speech recognition. *arXiv preprint arXiv:2501.19010*.

Li, H., & Wang, W. (2020). Reinterpreting ctc training as iterative fitting. *Pattern Recognition*, *105*, 107392.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, *18*(1), 6765–6816.

Lilley, J., Ratnagiri, M., & Bunnell, H. T. (2017). Prediction of speech delay from acoustic measurements. *Interspeech 2017*, 1859–1863. https://doi.org/10.21437/Interspeech.2017-1740

Lin, J., Wu, C.-E., Li, H., Zhang, J., Hu, Y. H., & Morgado, P. (2025). From prototypes to general distributions: An efficient curriculum for masked image modeling. *Proceedings of the Computer Vision and Pattern Recognition Conference*, 20028–20038.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Liu, Z., Luo, P., Wang, X., & Tang, X. (2018). Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, *15*(2018), 11.

Loshchilov, I., & Hutter, F. (2015). Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*.

Lotfian, R., & Busso, C. (2019). Curriculum learning for speech emotion recognition from crowdsourced labels. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *27*(4), 815–826.

Lyu, Y., & Tsang, I. W. (2020). Curriculum loss: Robust learning and generalization against label corruption. https://arxiv.org/abs/1905.10045

Mao, Y., Gupta, V., Wang, K., Liao, W.-k., Choudhary, A., & Agrawal, A. (2022). To shuffle or not to shuffle: Mini-batch shuffling strategies for multi-class imbalanced classification. *2022 International Conference on Computational Science and Computational Intelligence (CSCI)*, 298–301.

Masters, D., & Luschi, C. (2018). Revisiting small batch training for deep neural networks. https://arxiv.org/abs/1804.07612

McCandlish, S., Kaplan, J., Amodei, D., & Team, O. D. (2018). An empirical model of large-batch training. https://arxiv.org/abs/1812.06162

Mindermann, S., Brauner, J. M., Razzak, M. T., Sharma, M., Kirsch, A., Xu, W., Höltgen, B., Gomez, A. N., Morisot, A., Farquhar, S., et al. (2022). Prioritized training on points that are learnable, worth learning, and not yet learnt. *International Conference on Machine Learning*, 15630–15649.

Moore, R. C., & Lewis, W. (2010). Intelligent selection of language model training data. In J. Hajič, S. Carberry, S. Clark, & J. Nivre (Eds.), *Proceedings of the ACL 2010 conference short papers* (pp. 220–224). Association for Computational Linguistics. https://aclanthology.org/P10-2041/

Nagao, K., Paullin, M., Livinsky, V., Polikoff, J. B., Vallino, L. D., Morlet, T. G., Schanen, N. C., & Bunnell, H. T. (2012). Speech production-perception relationships in children with speech delay. *INTERSPEECH*, 1127–1130.

Naır, M., Yamani, K., Lhadj, L. S., & Baghdadi, R. (2024). Curriculum learning for small code language models. *arXiv preprint arXiv:2407.10194*.

Nemenyi, P. B. (1963). *Distribution-free multiple comparisons.* Princeton University.

Olejnik, F., Stachowiak, R., Krysińska, I., & Morzy, M. (2023). Aaron earned an iron urn: Speech-to-ipa models improve diagnostic of pronunciation. *Wojciechowski A.(Ed.), Lipiński P.(Ed.)., Progress in Polish Artificial Intelligence Research 4, Seria: Monografie Politechniki Łódzkiej Nr. 2437, Wydawnictwo Politechniki Łódzkiej, Łódź 2023, ISBN 978-83-66741-92-8, doi: 10.34658/9788366741928*.

Paul, M., Ganguli, S., & Dziugaite, G. K. (2021). Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems*, *34*, 20596–20607.

Penha, G., & Hauff, C. (2019). Curriculum learning strategies for ir: An empirical study on conversation response ranking. https://arxiv.org/abs/1912.08555

Phan, T., Driscoll, J., Romberg, J., & Koenig, S. (2024). Confidence-based curriculum learning for multi-agent path finding. *arXiv preprint arXiv:2401.05860*.

Platanios, E. A., Stretcu, O., Neubig, G., Poczos, B., & Mitchell, T. M. (2019). Competence-based curriculum learning for neural machine translation. *arXiv preprint arXiv:1903.09848*.

Qian, X., & Klabjan, D. (2020). The impact of the mini-batch size on the variance of gradients in stochastic gradient descent. *arXiv preprint arXiv:2004.13146*.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2023). Robust speech recognition via large-scale weak supervision. *International conference on machine learning*, 28492–28518.

Ranjan, S., & Hansen, J. H. (2017). Curriculum learning based approaches for noise robust speaker recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *26*(1), 197–210.

Sanabria, R., & Metze, F. (2018). Hierarchical multitask learning with ctc. *2018 IEEE Spoken Language Technology Workshop (SLT)*, 485–490.

Santamaría, L., & Axelrod, A. (2017). Data selection with cluster-based language difference models and cynical selection. In S. Sakti & M. Utiyama (Eds.), *Proceedings of the 14th international conference on spoken language translation* (pp. 137–145). International Workshop on Spoken Language Translation. https://aclanthology.org/2017.iwslt-1.19/

Satici, M. Y., Wang, J., & Roberts, D. L. (2025). Autonomous curriculum design via relative entropy based task modifications. *arXiv preprint arXiv:2502.21166*.

Settles, B., & Craven, M. (2008). An analysis of active learning strategies for sequence labeling tasks. In M. Lapata & H. T. Ng (Eds.), *Proceedings of the 2008 conference on empirical methods in natural language processing* (pp. 1070–1079). Association for Computational Linguistics. https://aclanthology.org/D08-1112/

Seung, H. S., Opper, M., & Sompolinsky, H. (1992). Query by committee. *Proceedings of the fifth annual workshop on Computational learning theory*, 287–294.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, *27*(3), 379–423.

Shi, T., Wu, Y., Song, L., Zhou, T., & Zhao, J. (2025). Efficient reinforcement finetuning via adaptive curriculum learning. https://arxiv.org/abs/2504.05520

Shukla, M. (2022). Bayesian uncertainty and expected gradient length-regression: Two sides of the same coin? *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2367–2376.

Siminyu, K., Li, X., Anastasopoulos, A., Mortensen, D. R., Marlo, M. R., & Neubig, G. (2021). Phoneme recognition through fine tuning of phonetic representations: A case study on luhya language varieties. *Interspeech 2021*, 271–275. https://doi.org/10.21437/Interspeech.2021-1434

Snow, R., O'connor, B., Jurafsky, D., & Ng, A. Y. (2008). Cheap and fast–but is it good? evaluating non-expert annotations for natural language tasks. *Proceedings of the 2008 conference on empirical methods in natural language processing*, 254–263.

Soviany, P., Ardei, C., Ionescu, R. T., & Leordeanu, M. (2020). Image difficulty curriculum for generative adversarial networks (cugan). *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 3463–3472.

Summers, C., & Dinneen, M. J. (2020). Four things everyone should know to improve batch normalization. https://arxiv.org/abs/1906.03548

Swayamdipta, S., Schwartz, R., Lourie, N., Wang, Y., Hajishirzi, H., Smith, N. A., & Choi, Y. (2020). Dataset cartography: Mapping and diagnosing datasets with training dynamics. *arXiv preprint arXiv:2009.10795*.

Tay, Y., Wang, S., Tuan, L. A., Fu, J., Phan, M. C., Yuan, X., Rao, J., Hui, S. C., & Zhang, A. (2019). Simple and effective curriculum pointer-generator networks for reading comprehension over long narratives. *arXiv preprint arXiv:1905.10847*.

Toneva, M., Sordoni, A., Combes, R. T. d., Trischler, A., Bengio, Y., & Gordon, G. J. (2018). An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*.

Tudor Ionescu, R., Alexe, B., Leordeanu, M., Popescu, M., Papadopoulos, D. P., & Ferrari, V. (2016). How hard can it be? estimating the difficulty of visual search in an image. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2157–2166.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). Attention is all you need. https://arxiv.org/abs/1706.03762

Wang, X., Chen, Y., & Zhu, W. (2021). A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, *44*(9), 4555–4576.

Weinshall, D., & Amir, D. (2018). Theory of curriculum learning, with convex loss functions. https://arxiv.org/abs/1812.03472

Weinshall, D., Cohen, G., & Amir, D. (2018). Curriculum learning by transfer learning: Theory and experiments with deep networks. *International conference on machine learning*, 5238–5246.

Wrembel, M., Hwaszcz, K., Pludra, A., Skałba, A., Weckwerth, J., Malarski, K., Cal, Z., Kędzierska, H., Czarnecki-Verner, T., Balas, A., Kaźmierski, K., Żychliński, S., & Gruszecka, J. (2024a). The LnNor corpus: A spoken multilingual corpus of non-native and native norwegian, english and polish (part 1) [CLARIN-PL digital repository]. http://hdl.handle.net/11321/931

Wrembel, M., Hwaszcz, K., Pludra, A., Skałba, A., Weckwerth, J., Malarski, K., Cal, Z., Kędzierska, H., Czarnecki-Verner, T., Balas, A., Kaźmierski, K., Żychliński, S., & Gruszecka, J. (2024b). The LnNor corpus: A spoken multilingual corpus of non-native and native norwegian, english and polish (part 2) [CLARIN-PL digital repository]. http://hdl.handle.net/11321/932

Wu, D. X., Yun, C., & Sra, S. (2023). On the training instability of shuffling sgd with batch normalization. *International conference on machine learning*, 37787–37845.

Wu, X., Dyer, E., & Neyshabur, B. (2020). When do curricula work? *arXiv preprint arXiv:2012.03107*.

Xie, T., Zhu, J., Ma, G., Lin, M., Chen, W., Yang, W., & Liu, S. (2024). Structural-entropy-based sample selection for efficient and effective learning. *arXiv preprint arXiv:2410.02268*.

Xu, B., Zhang, L., Mao, Z., Wang, Q., Xie, H., & Zhang, Y. (2020). Curriculum learning for natural language understanding. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 6095–6104). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.542

Yamada, Y., Ueno, S., Oshita, T., Nakatsuka, S., & Kato, K. (2023). Acl: Active curriculum learning to reducing label efforts. *Sixteenth International Conference on Quality Control by Artificial Vision*, *12749*, 141–148.

Yang, H., Li, H., Yang, M., Chen, X., & Gong, M. (2025). Estimating the effects of sample training orders for large language models without retraining. *arXiv preprint arXiv:2505.22042*.

Zeyer, A., Schlüter, R., & Ney, H. (2021). Why does ctc result in peaky behavior? *arXiv preprint arXiv:2105.14849*.

Zhang, C., Kjellstrom, H., & Mandt, S. (2017). Determinantal point processes for mini-batch diversification. *arXiv preprint arXiv:1705.00607*.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper˙files/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf

Zhang, X., Kumar, G., Khayrallah, H., Murray, K., Gwinnup, J., Martindale, M. J., McNamee, P., Duh, K., & Carpuat, M. (2018). An empirical exploration of curriculum learning for neural machine translation. *arXiv preprint arXiv:1811.00739*.

Zhang, X., Shapiro, P., Kumar, G., McNamee, P., Carpuat, M., & Duh, K. (2019). Curriculum learning for domain adaptation in neural machine translation. *arXiv preprint arXiv:1905.05816*.

Zhou, T., Wang, S., & Bilmes, J. (2020). Curriculum learning by dynamic instance hardness. *Advances in Neural Information Processing Systems*, 8602–8613.

Zhou, Y., Yang, B., Wong, D. F., Wan, Y., & Chao, L. S. (2020). Uncertainty-aware curriculum learning for neural machine translation. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 6934–6944). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.620