EVIDENTIAL DEEP LEARNING FOR HIGH-CONFIDENCE SAMPLE SELECTION IN NOISY LABEL LEARNING

MARC PASCUAL ROIG

Thesis supervisor
PETIA RADEVA ((ENG)Universitat de Barcelona)

Thesis co-supervisor
BHALAJI NAGARAJAN ((ENG)Universitat de Barcelona)

Degree
Master's Degree in Artificial Intelligence

Master's thesis

School of Engineering
Universitat Rovira i Virgili (URV)

Faculty of Mathematics
Universitat de Barcelona (UB)

Barcelona School of Informatics (FIB)
Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

01/07/2025

This thesis is submitted to the Computer Science Department, Universitat Politècnica de Catalunya in fulfilment of the requirements for the master's degree in Artificial Intellignece.

Marc Pascual Roig, June 2025

Acknowledgements

I would like to express my sincere gratitude to my thesis supervisor, Prof. Petia Radeva, and co-supervisor, Dr. Bhalaji Nagarajan, for their invaluable guidance and dedication, without which this thesis would not have been possible.

Abstract

Learning with Noisy Labels (LNL) remains a remarkable challenge in deep learning, as models tend to memorize incorrect labels, especially under high-noise conditions. While many state-of-the-art methods can effectively identify clean samples when label noise is low, their performance deteriorates as the proportion of mislabeled data increases. This leads to degraded model generalization and reduced reliability in real-world scenarios.

In this thesis, we address these challenges by integrating Evidential Deep Learning (EDL) into LNL high-confidence sample selection frameworks. Our proposed framework combines uncertainty-aware modeling, self-supervised contrastive learning, and a refined clean sample selection strategy to create a robust training pipeline. The proposed method consistently outperforms existing LNL frameworks in high-noise settings and remains competitive when the noise level is low.

Contents

List of Figures

Chapter 1

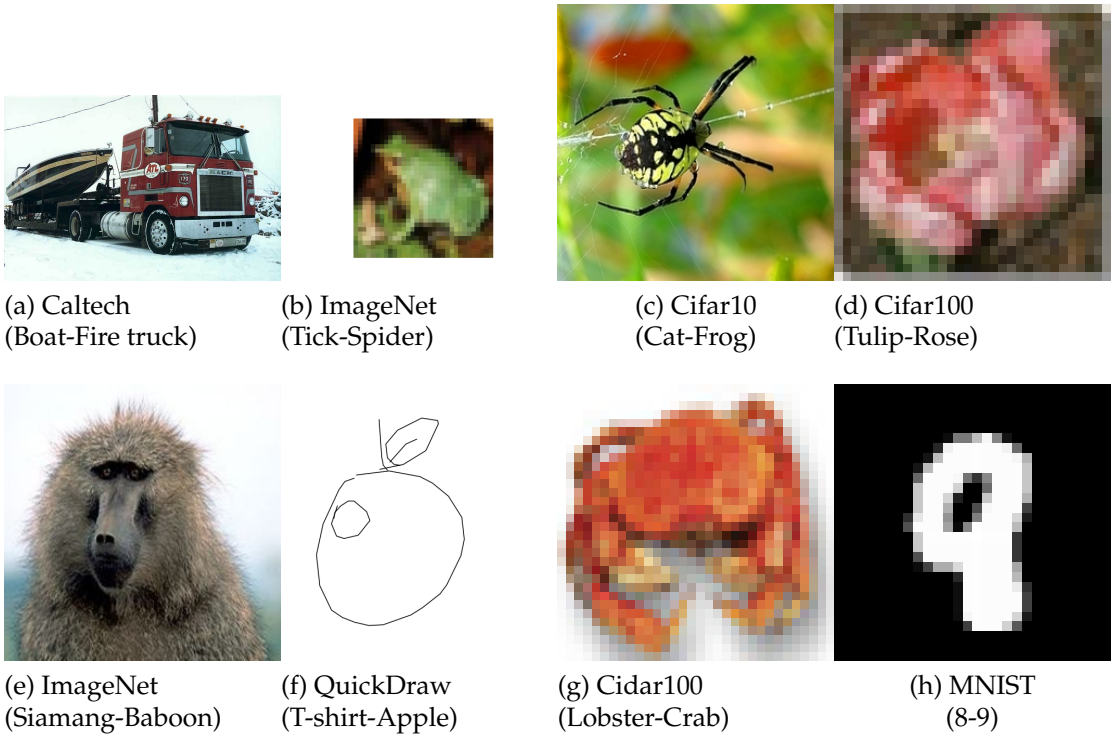
Introduction

1.1 Context and Motivation

Deep Learning model architectures are growing increasingly large and complex, achieving remarkable performance improvements across a wide range of tasks, including complex computer vision tasks (Krizhevsky et al., 2012; Redmon et al., 2016; Dosovitskiy et al., 2021; Kirillov et al., 2023; He et al., 2022; Radford et al., 2021). Much of this progress has been driven by supervised learning, which relies heavily on access to vast amounts of labeled data. As models scale, so does their need for high-quality annotations, making data collection a major bottleneck. Manual labeling is expensive and prone to human error, while automated labeling pipelines can introduce inconsistencies. Consequently, **label noise** — incorrect or ambiguous annotations — is inevitable in large-scale datasets (Frenay and Verleysen, 2014; Liang et al., 2022; Xiao et al., 2015*a*). Training with noisy labels has an detrimental impact on the generalization of models due to their over-parameterization (Allen-Zhu et al., 2019) and strong memorization capability (Zhang et al., 2021).

To continue leveraging large datasets, which inevitably contain errors, without compromising model performance and generalization, or introducing bias, the field of **Learning with Noisy Labels (LNL)** has emerged (Song et al., 2022; Liang et al., 2022). Various familry of LNL methods acknowledge and address the presence of noise in training data. One promising line of work within LNL involves high-confidence sample selection models that rely on identifying and using samples they consider clean. A notable benchmark algorithm is DivideMix (Li et al., 2020), which has demonstrated strong performance in learning with noisy labels.

[1]Source: https://labelerrors.com/



(a) Caltech (Boat-Fire truck)
(b) ImageNet (Tick-Spider)
(c) Cifar10 (Cat-Frog)
(d) Cifar100 (Tulip-Rose)
(e) ImageNet (Siamang-Baboon)
(f) QuickDraw (T-shirt-Apple)
(g) Cidar100 (Lobster-Crab)
(h) MNIST (8-9)

Figure 1.1: Examples of mislabeled images in benchmark datasets[1]. The dataset name is indicated in the subcaption, while the class labels in parentheses correspond to the incorrect label (first) and the correct label (second).

Despite meaningful advances, the problem of LNL remains far from solved. Current sample selection methods still struggle to accurately distinguish clean samples from noisy samples, especially under high noise conditions, leading to suboptimal selection. Furthermore, deep neural networks tend to memorize noisy labels (*memorization effect*) over time (Arpit et al., 2017), which can degrade generalization performance and introduce bias. Addressing these challenges requires more effective modeling of sample selection criteria and improved mechanisms to mitigate memorization effects.

Another challenge within LNL — and deep learning more broadly — is the inability to effectively express uncertainty. Neural networks are often overconfident in their predictions (Gawlikowski et al., 2022) and generally lack reliable mechanisms for uncertainty quantification (Gawlikowski et al., 2022). To address this issue, various approaches have been proposed, including ensemble methods (Lakshminarayanan et al., 2017; Sagi and Rokach, 2018; Renda et al., 2019) and Bayesian approximations (Duane et al., 1987; Gal and Ghahramani, 2016*a*,*b*).

Despite this, very few works (Malinin and Gales, 2018; Huang et al., 2022; Na-

garajan et al., 2024) have explored the integration of uncertainty estimation in LNL. These methods have shown promising results, highlighting uncertainty estimation as a valuable and promising direction for improving robustness in learning with noisy labels. A compelling approach is to incorporate uncertainty estimation using Evidential Deep Learning (EDL) (Sensoy et al., 2018; Gao et al., 2024; Ulmer et al., 2023; Guo et al., 2024). Unlike other methods, EDL handles uncertainty directly during training, where the network's outputs are modeled using Dirichlet distributions, which align with subjective opinions in Subjective Logic (Jøsang, 2016) — naturally capturing uncertainty in a principled way. Additionally, unlike ensemble methods or Bayesian neural networks, EDL does not incur additional computational cost during training.

In this thesis, we explore the potential of EDL as a powerful tool for learning with noisy labels. We argue that incorporating uncertainty can help models better handle label noise, ultimately leading to improved robustness and generalization without incurring additional training cost.

1.2 Objectives of the Thesis

Motivated by the challenges in LNL algorithms, the primary objective of this thesis is to address the *memorization effect* and *overconfident predictions* exhibited by deep neural networks, particularly within the sample selection methods of LNL framework.

To achieve this, we build upon the popular DivideMix framework by incorporating EDL to directly model uncertainty during training. Furthermore, we leverage state-of-the-art (SOTA) strategies to mitigate noise effects and improve model robustness. Following this, we list the **specific objectives** that help us develop a robust sample selection LNL framework that effectively handles noisy labels by improving uncertainty estimation in deep neural networks and reducing memorization in them.

- Gain a thorough **understanding of the fundamentals** of LNL, with a particular focus on sample selection strategies and DivideMix-based methods.
- Conduct a **comprehensive study of uncertainty estimation** in deep neural networks, including its taxonomy, with special emphasis on Dirichlet-based approaches.

- Investigate how uncertainty estimation techniques — particularly EDL — can be integrated into the DivideMix framework to **mitigate overconfident predictions**.
- Design a **sample selection-based LNL pipeline** that incorporates self-supervised learning techniques to effectively counteract the memorization of noisy labels.
- **Evaluate the performance** of the proposed method on standard benchmark datasets across noise types and ratios and compare it with existing SOTA approaches.
- **Analyze the results** both quantitatively and qualitatively to gain insights into the model's behavior, with particular emphasis on uncertainty estimation and its impact.

1.3 Contributions of the Thesis

Based on the motivations and objectives stated above, the main contributions of this thesis are as follows:

1. A novel DivideMix-like algorithm that integrates EDL to improve uncertainty modeling and robustness to label noise.
2. The use of contrastive learning techniques — including contrastive loss and pretrained self-supervised models — to reduce the memorization effect and enhance representation learning.
3. An uncertainty-aware MixMatch strategy that leverages vacuity-based weighting to better handle epistemic uncertainty[2] during training.
4. A comprehensive evaluation and analysis of the proposed framework on standard noise benchmarks, including both quantitative results and qualitative assessments of uncertainty behavior.

[2]Epistemic uncertainty refers to uncertainty arising from limited knowledge or data. It captures the model's uncertainty about its own parameters and can, in principle, be reduced with more data.

1.4 Organization of the Thesis

To ensure clarity and coherence, the thesis is organized into the following chapters:

1. **Introduction (Chapter 1):** Introduces the context, motivation, objectives, and contributions of the thesis.

2. **Theoretical Background (Chapter 2):** Presents a review of core concepts in Learning with Noisy Labels and Uncertainty Estimation, with emphasis on prior work relevant to this thesis.

3. **Methodology (Chapter 3):** Describes the methodological foundations of the thesis, detailing DivideMix and EDL.

4. **Proposed Method (Chapter 4):** Introduces our proposed DivideMix-based LNL framework, explaining the integration of EDL, vacuity-based MixMatch, and contrastive learning techniques.

5. **Validation (Chapter 5):** Details the experimental setup, benchmark datasets, evaluation metrics, and presents a comprehensive analysis of results both quantitatively and qualitatively. In addition, the sustainability of the thesis is discussed.

6. **Conclusions and Future Work (Chapter 6):** Summarizes the main findings, discusses the broader implications, and suggests directions for future research.

Chapter 2

Theoretical background

This chapter presents the theoretical foundations necessary for understanding the challenges and solutions related to learning with noisy labels and uncertainty in deep learning models. It is organized into three main sections. Section 2.1 introduces the problem of noisy labels, explores their impact on model performance, and reviews key approaches to mitigate noise, including a focus on sample selection methods like DivideMix. Section 2.2 provides an overview of uncertainty quantification in neural networks, detailing different types of uncertainty and methods to estimate them. Finally, Section 2.3 examines how uncertainty estimation techniques can be integrated with LNL frameworks to enhance robustness and reliability.

2.1 Learning with Noisy Labels

Various studies (Tanno et al., 2019*a*) have shown that traditional and widely-used regularization techniques, such as data augmentation, weight decay, dropout, and batch normalization, are insufficient to fully mitigate the negative impact of label noise (especially in high-noise scenarios), which degrades generalization performance on unseen data (Frenay and Verleysen, 2014).

In this context, Learning with Noisy Labels (LNL) has emerged as a critical area of research in deep learning (Song et al., 2022; Liang et al., 2022), with a focus on developing strategies to reduce the adverse effects of noisy annotations in supervised learning.

This chapter introduces the LNL problem, with a focus on classification tasks. The structure is as follows: we begin by defining label noise in classification datasets. We then present key approaches in LNL - specifically, robust architectures, robust

regularization, robust loss functions, loss adjustment techniques, and sample selection strategies. The goal is to provide the reader with a comprehensive understanding of LNL concepts and the main strategies employed to address label noise. Subsequently, we focus on the sample selection approach, with particular emphasis on the DivideMix family of methods, which forms the foundation of this project. We describe the general model pipeline and review relevant related work that inspired our approach.

Finally, Table 2.1 summarizes the notation used throughout this chapter.

Notation	Description
X	The data feature space
$Y, \tilde{Y}$	The true and noisy label space
$D, \tilde{D}$	The clean and noisy training data
$P_D, P_{\tilde{D}}$	The joint distributions of clean and noisy data
B_t	A set of mini-batch examples at time t
Θ_t	The parameter of a deep neural network at time t
$f(\cdot;\Theta_t)$	A deep neural network parameterized by Θ_t
ℓ	A specific loss function
R	An empirical risk
$\mathbb{E}_D$	An expectation over D
x, x_i	A data example of X
y, y_i	A true label of Y
$\tilde{y}, \tilde{y}_i$	A noisy label of $\tilde{Y}$
η	A specific learning rate
τ	A true noise rate
b	The number of mini-batch examples in B_t
c	The number of classes
$T, \hat{T}$	The true and estimated noise transition matrix

Table 2.1: Summary of notation.

2.1.1 Noise in Datasets

The concept of noise in datasets is inherently ambiguous. However, it is typically categorized into two broad types: feature noise and class (label) noise (Frenay and Verleysen, 2014). Feature noise affects the observed values of the input features - for example, Gaussian noise introduced during measurement. In contrast, label noise alters the assigned class labels. In the context of LNL, we are primarily concerned with label noise, as it tends to be the most detrimental to learning. Accordingly, throughout this chapter, we refer to label noise simply as noise.

The presence of noisy labels in large datasets is almost inevitable and stems from various sources. As outlined by Liang et al. (2022), the two primary sources of label noise are explained below:

1. Cognitive bias and ambiguity: Human annotators - particularly those involved in crowdsourcing - may label data inconsistently due to subjective biases. Furthermore, some data types (e.g., medical imagery, emotional speech, or facial aesthetics) are inherently ambiguous, making accurate labeling difficult even for experts.

2. Web-based data collection: Large-scale datasets are often created using data from web search engines or social media platforms. Labels are typically derived from surrounding textual metadata, introducing noise due to linguistic ambiguity or misleading context.

In the LNL literature, the robustness of algorithms is often analyzed in relation to both the noise rate and the noise type. The noise rate is defined as the proportion of mislabeled examples within a dataset. A higher noise rate implies a more corrupted dataset, making it harder to learn meaningful patterns and increasing the risk of overfitting to noisy labels. In typical classification benchmarks, noise rates range between 8% and 38.5% (Xiao et al., 2015*a*).

2.1.1.1 Types of Label Noise

This section describes common types of label noise considered when designing robust learning algorithms. Even when label corruption is assumed to be random, it is often influenced by relationships between data features and class labels. A detailed taxonomy is available in Frenay and Verleysen (2014). We primarily distinguish between two types of label noise: instance-independent and instance-dependent noise.

Instance-independent Label Noise. Instance-independent noise assumes that the probability of label corruption depends solely on the class label and not on the data instance itself. This assumption is often modeled via a noise transition matrix $T \in [0,1]^{c \times c}$ (Song et al., 2022), where each entry T_{ij} indicates the probability that a true label i is flipped to a corrupted label j:

$$T_{ij} = P(\tilde{y} = j \mid y = i), \tag{2.1}$$

If the dataset is noise-free, T becomes a diagonal matrix.

There are several common forms of instance-independent noise:

- **Symmetric (Uniform) Noise:** Each label is flipped uniformly to any of the other classes:
$$\forall j \neq i: \quad T_{ij} = \frac{\eta}{c-1}, \quad T_{ii} = 1 - \eta, \tag{2.2}$$
where $\eta < 1$ is the noise rate and c is the number of classes.

- **Asymmetric (Label-dependent) Noise:** Each label has a specific probability to be flipped to another class, depending on class similarity:
$$\exists j \neq i : T_{ij} > T_{ik}, \forall k \neq j; \quad T_{ii} = 1 - \eta. \tag{2.3}$$
For example, the label "dog" might be misclassified as "cat" more often than as "fish".

- **Pair Noise:** A special case of asymmetric noise, where each label is flipped to a single specific incorrect class:
$$T_{ii} = 1 - \eta; \quad T_{ij} = \eta, \quad T_{ik} = 0 \text{ for } k \neq j, i. \tag{2.4}$$

Instance-dependent Label Noise. In more realistic scenarios, label corruption may depend on both the instance's features and its true class label. Some approaches (Xiao et al., 2015*b*; Goldberger and Ben-Reuven, 2017) model this dependency using a noise adaptation layer within the neural network architecture. The transition probability is defined as:

$$T_{ij}(x) = P(\tilde{y} = j \mid y = i, x), \tag{2.5}$$

where x denotes the data features. Unlike instance-independent noise, here the mislabeling probability varies across different instances, making the modeling process more complex.

Controlled Web Label Noise. To bridge the gap between theoretical models and real-world scenarios, Liang et al. (2022) proposed controlled web label noise. This approach mimics real-world noise by collecting mislabeled data via Google image searches, using both text-to-image and image-to-image queries. After filtering out duplicates (especially those overlapping with the test set), a proportion $p \in [0, 100]$ of original training samples is replaced with the noisy data. Like symmetric noise, p is uniformly applied across all classes. This type of noise offers a balance between realism and controllability, making it suitable for benchmarking robust algorithms.

2.1.2 LNL Approaches and Methods

This section provides a concise overview of the main approaches developed to address LNL. Rather than delving into implementation details, the goal is to give the reader a broad understanding of the landscape and the core strategies commonly used to handle noisy labels in deep learning. These methods are generally categorized into five groups (Song et al., 2022): robust architecture, robust regularization, robust loss functions, loss adjustment, and sample selection.

2.1.2.1 Robust Architecture

Robust architecture methods modify DNNs to explicitly model label noise. Two main families of approaches exist.

A common strategy is to introduce a noise adaptation layer that estimates the noise transition matrix, learning the probability of clean labels being corrupted. This layer, positioned above the softmax output, helps the model adapt to corrupted labels. Variants include Webly Learning, the noise model, dropout noise model, S-model, and C-model. While these approaches improve generalization, they often struggle to identify mislabeled instances, leading to errors when noise levels are high.

An alternative direction designs dedicated architectures to handle complex noise patterns beyond label-dependent transitions. These models introduce specialized components, such as probabilistic noise modeling (Xiao et al., 2015*c*) — which uses two independent networks to estimate noise type and transition probabilities — and masking techniques (Han, Yao, Niu, Zhou, Tsang, Zhang and Sugiyama, 2018) that constrain invalid label transitions using human priors. Recent methods, like contrastive-additive noise networks (Yao et al., 2017), introduce quality embeddings to refine noise modeling, while robust generative classifiers build on pre-trained DNNs to enhance noise resilience. Despite their strengths, these architectures often lack generalization across different model types.

2.1.2.2 Robust Regularization

Traditional regularization techniques such as data augmentation, weight decay, dropout, and batch normalization improve resilience to moderate noise but degrade under high-noise settings. To overcome this, more advanced regularization strategies have emerged, often categorized into explicit and implicit forms.

Explicit regularization directly modifies the loss or optimization process. For example, bilevel learning (Jenni and Favaro, 2018) uses a clean validation set to guide optimization, while annotator confusion models (Tanno et al., 2019*b*) estimate transition probabilities via regularized EM. Pre-training (Hendrycks et al., 2019) also improves robustness by leveraging transferable features. However, these methods often involve model-specific hyperparameters requiring careful tuning.

Implicit regularization improves generalization without directly altering the loss. Adversarial training enhances robustness by requiring the model to correctly classify both clean and perturbed samples. **Label smoothing** (Lukasik et al., 2020) replaces hard labels with softened distributions, reducing overfitting. **Mixup** (Zhang et al., 2018) (Figure 2.1) linearly interpolates between training samples and their labels, encouraging smoother decision boundaries and lowering sensitivity to noise.

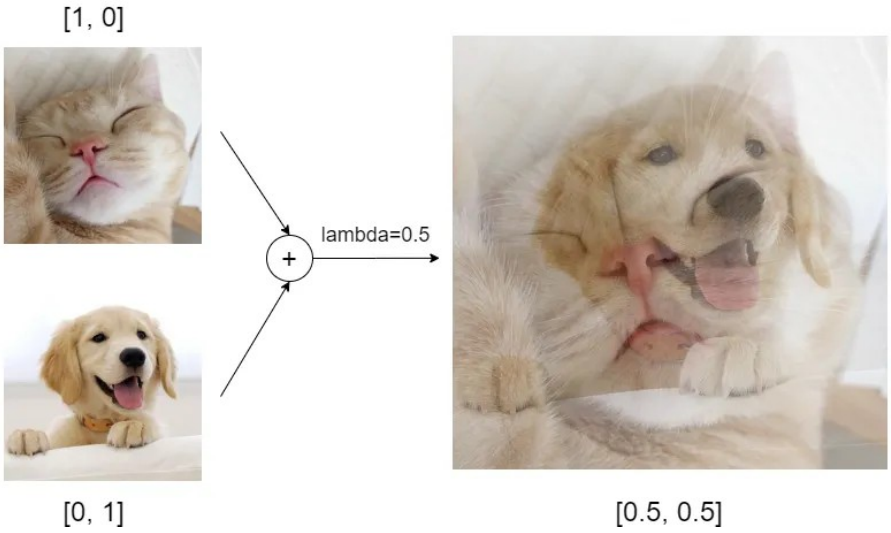


Figure 2.1: Example of Mixup augmentation.[1]

Overall, robust regularization improves noise tolerance: explicit methods offer principled control at the cost of complexity, while implicit methods improve generalization with minimal architectural changes.

2.1.2.3 Robust Loss Functions

This approach modifies the loss function to improve robustness against noisy labels. In theory, a well-designed loss function allows the classifier to approximate the Bayes-optimal solution despite label noise. Categorical cross-entropy (CCE),

[1]Source: Medium article on Mixup augmentation

though widely used due to its fast convergence, is highly sensitive to noise. In contrast, mean absolute error (MAE) is more noise-tolerant but struggles with complex data.

To balance robustness and generalization, several hybrid losses have been proposed. These include generalized cross-entropy (GCE) (Zhang and Sabuncu, 2018), curriculum loss (CL) (Lyu and Tsang, 2020), and active-passive loss[2] (Ma et al., 2020). Although theoretically sound, their performance often drops in real-world, high-dimensional tasks.

Recently, contrastive learning[3] has been incorporated into LNL. The goal is to learn noise-robust features using data augmentations, sample selection, or representation learning. Examples include JO-SRC (Yao et al., 2021), Selective-CL (Li et al., 2022), and noise-aware contrastive methods (Yi et al., 2022).

Chimera (Liu et al., 2023) combines Mixup augmentations with contrastive losses. UNICON (Karim et al., 2022), PSSCL (Zhang et al., 2025), BPT-PLR (Zhang et al., 2024), and PLReMix (Liu et al., 2024) all enhance standard supervised losses with unsupervised contrastive components. Pre-training the encoder (Zheltonozhskii et al., 2022) is also beneficial in these frameworks.

2.1.2.4 Loss Adjustment

Loss adjustment methods retain standard loss functions but modify their behavior during training to account for label noise. They fall into four categories.

Loss correction (Patrini et al., 2017) estimates the noise transition matrix and corrects the loss via forward or backward correction. **Loss reweighting** (Liu and Tao, 2014) assigns sample weights to reduce the influence of mislabeled data. While effective, these methods often require carefully chosen weighting schemes.

Label refurbishment (Song et al., 2019) blends noisy labels with model predictions to create pseudo-clean targets. Techniques such as bootstrapping and SELFIE improve label quality through model consistency. **Meta-learning** (Shu et al., 2019) learns optimal adjustment strategies using a small clean validation set. Examples include Meta-Weight-Net and knowledge distillation frameworks.

These methods improve training dynamics under noise but may rely on clean data, accurate noise estimation, or additional hyperparameters.

[2]The "active" loss maximizes the probability of the labeled class; the "passive" loss minimizes the probability of all other classes.

[3]Contrastive loss pulls similar samples closer in the embedding space and pushes dissimilar ones apart. In LNL, this often involves different augmentations of the same image.

2.1.2.5 Sample Selection

Sample selection improves robustness by identifying clean examples and training the model primarily on them. These approaches are grounded in the **memorization effect** — the tendency of DNNs to first learn simple patterns before overfitting to noise (Arpit et al., 2017; Song et al., 2020). As a result, early-training samples with small losses are more likely to be clean.

Multi-network learning involves training two (or more) DNNs that collaboratively select clean samples. For example, Decoupling (Malach and Shalev-Shwartz, 2018) updates each model using samples on which they disagree. The **small-loss trick** — selecting low-loss samples early — is used in MentorNet (Jiang et al., 2018), where a mentor network guides a student. Co-Teaching (Han, Yao, Yu, Niu, Xu, Hu, Tsang and Sugiyama, 2018) extends this by having both networks exchange small-loss samples during training.

Multi-round learning improves selection over time by iteratively refining labels and retraining the model, rather than relying on auxiliary networks.

Sample selection is often combined with other techniques to exploit noisy data more effectively. SELF (Nguyen et al., 2019) integrates semi-supervised learning to filter noisy samples progressively. DivideMix (Li et al., 2020) uses Mixup (Berthelot et al., 2019) to blend clean and noisy examples. RoCL (Zhou et al., 2021) combines supervised training on selected clean samples with self-supervised learning on relabeled noisy data.

2.1.3 The DivideMix Family

The DivideMix framework (Li et al., 2020) is one of the most prominent multi-network approaches within the family of sample selection algorithms. Since it is the method chosen for this project, we provide a brief explanation of the original algorithm, followed by an overview of its most impactful extensions.

DivideMix is a robust training strategy for LNL that builds on several key principles. It exploits the observation that neural networks tend to learn clean patterns before memorizing noisy labels, employing two networks in a co-teaching setup. After an initial warm-up phase, it partitions training samples into clean and noisy subsets based on their loss distributions. Clean samples are used as labeled data, while noisy ones receive pseudo-labels and are incorporated into a semi-supervised MixMatch framework, which improves robustness through data augmentation and consistency regularization.

Subsequent research has introduced several improvements to various stages of the DivideMix training pipeline. In what follows we highlight some of the most meaningful contributions. Both pretraining the model to learn the features of the samples before the Semi-Supervised Training stage and adding contrastive loss to the framework to mitigate the negative effect of labeling noisy samples as clean help to the robustness of the model.

Contrast2Divide (Zheltonozhskii et al., 2022) emphasizes the importance of the pre-training stage by incorporating self-supervised pre-training to improve feature representations. ScanMix (Sachdeva et al., 2023) also leverages a self-supervised pretraining stage with a contrastive loss to achieve a good feature representation of the samples. Then, semantic clustering is trained in parallel to the semi-supervised training. Approaches such as UniCon (Karim et al., 2022) add an unsupervised contrastive loss to counter the effect of considering noisy samples as clean. Noisy labels hinder the learning from the feature representations that happens in supervised contrastive learning. To tackle this issue some approaches such as Li et al. (2022) rely on selecting confident samples only for the contrastive learning. PlReMix (Liu et al., 2024) and BPT-PLR (Zhang et al., 2024) add a pseudo-label relaxed contrastive loss to the pipeline. Differently to ScanMix — where only the Semi-Supervised Training loss is used to split the data —, both losses — the per-sample loss and the per-sample contrastive loss — are used in the division of the data leveraging two 2-dimensional GMMs.

There are works aiming to improve the selection of clean samples. This can be achieved by refining the clean examples selected by the GMMs, using different parametric or even non-parametric models to divide the data, or leveraging some relabeling and correction strategies. Many different strategies have proven to be successful in obtaining a clean and reliable set of samples that balance the precision and recall of clean samples. LongReMix (Cordeiro et al., 2023) adds an additional stage to the traditional DivideMix pipeline that focuses on acquiring a reliable set of clean samples that is later used in the second stage. DISC (Li et al., 2023) proposes a dynamic threshold strategy for each instance, based on the momentum of each instance's memorization strength in previous epochs to select and correct noisy labeled data. Huang et al. (2022) generates prototypes for each class based on supervised contrastive learning that are used to correct the labels. BPT-PLR (Zhang et al., 2024) equalizes the number of clean samples that belong to each class in a balanced partitioning process that selects the same number of samples for each class. UniCon (Karim et al., 2022) selects the clean samples based

on the Jensen-Shannon divergence (JSD) distribution instead of the loss in addition to a class-specific threshold. In Zhao et al. (2022) clean samples are selected in a consistency-classification process after extracting and clustering the normalized features of the samples and comparing them with the centers of the clusters. Kim et al. (2024) uses soft structural labels that are more robust to outliers (noisy labels) by leveraging a reverse K-NN algorithm in the feature space to predict the class of the sample based on nearest neighbors. Miao et al. (2023) leverages the network predictions and the original noisy training labels directly, to carry out an adaptive modeling of non-parametric distributions for the individual clean/noisy samples.

2.2 Uncertainty Quantification in Neural Networks

Despite the success of DNNs over traditional models, several limitations have been identified (Gawlikowski et al., 2022): (1) they often lack interpretability and transparency, (2) they are unable to reliably distinguish between in-domain and out-of-domain samples or handle domain shifts, and (3) they typically fail to provide well-calibrated uncertainty estimates, often resulting in overconfident predictions.

Understanding the types of uncertainty that arise in neural networks and how to estimate them in practice is critical (Gawlikowski et al., 2022) — particularly in safety-critical applications, high-stakes domains, or *scenarios with limited or unreliable labeled data*.

To address this, numerous approaches (Gawlikowski et al., 2022; Abdar et al., 2021) have been proposed for estimating what is broadly referred to as *predictive uncertainty* — the uncertainty associated with a model's output prediction.

Predictive uncertainty propagates through four main stages:

- **Data acquisition:** Uncertainty may stem from inherent variability in the real world or from measurement and labeling errors.
- **Model construction:** The design and training of DNNs influence the optimization process. For example, deeper models often produce more overconfident predictions.
- **Inference:** When new inputs are provided for prediction, they may fall outside the distribution seen during training.
- **Uncertainty modeling:** Both model-related and data-related uncertainties need to be explicitly modeled and quantified.

2.2.1 Derivation of Uncertainty and Types of Uncertainty

Predictive uncertainty quantifies the confidence of a neural network's prediction for a new input x^*. It primarily arises from two sources: *aleatoric uncertainty* (data uncertainty) and *epistemic uncertainty* (model uncertainty). Some approaches also consider an additional component known as *distributional uncertainty*.

The predictive distribution over an output y^*, given input x^* and latent variable ω, is defined as:

$$p(y^*|x^*) = \int_\Omega p(y^*|\omega)\, p(\omega|x^*)\, d\omega. \tag{2.6}$$

A common approximation involves computing the maximum a posteriori (MAP) estimate:

$$y^* = \arg\max_y p(y|x^*). \tag{2.7}$$

Using a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, the predictive distribution can also be written as:

$$p(y^*|x^*) = \int_\mathcal{D} p(y^*|\mathcal{D}, x^*)\, d\mathcal{D}, \tag{2.8}$$

and the MAP estimate becomes:

$$y^* = \arg\max_y p(y|\mathcal{D}, x^*). \tag{2.9}$$

In a Bayesian framework, predictive uncertainty is decomposed into model and data uncertainties. Let θ denote the model parameters. Then, Eq. 2.8 becomes:

$$p(y^*|x^*, \mathcal{D}) = \int p(y^*|x^*, \theta)\, p(\theta|\mathcal{D})\, d\theta, \tag{2.10}$$

where:

- $p(y^*|x^*, \theta)$ represents **aleatoric uncertainty**, arising from data noise.
- $p(\theta|\mathcal{D})$ encodes **epistemic uncertainty**, or the uncertainty over the model parameters.

Aleatoric uncertainty, also known as data uncertainty, originates from inherent noise in the input data, such as sensor inaccuracies, ambiguities, or label errors. This type of uncertainty is considered irreducible — it cannot be minimized by collecting more data or improving model complexity, but must be explicitly modeled. Examples include blurred images or mislabeled samples, where even a perfect model cannot make accurate predictions.

Epistemic uncertainty, or model uncertainty, arises when the model lacks knowledge due to insufficient or unrepresentative training data, suboptimal architecture choices, or stochastic training dynamics (e.g., random initialization or data shuffling). Unlike aleatoric uncertainty, epistemic uncertainty can be reduced by collecting more informative data or improving the model design.

For more advanced modeling, a third component — **distributional uncertainty** — is introduced:

$$p(y^*|x^*, \mathcal{D}) = \int \int p(y|\mu)\, p(\mu|x^*, \theta)\, p(\theta|\mathcal{D})\, d\mu\, d\theta, \tag{2.11}$$

with the following interpretation:

- $p(y|\mu)$ models **data uncertainty**.
- $p(\mu|x^*, \theta)$ models **distributional uncertainty**.
- $p(\theta|\mathcal{D})$ captures **model uncertainty**.

Here, distributional uncertainty refers to uncertainty arising from input data that deviates from the training distribution. A typical example includes using a Dirichlet distribution to model class probabilities in classification tasks. This formulation reveals how model uncertainty propagates to distributional uncertainty, which in turn affects data uncertainty.

The integrals in Eq. 2.14 and 2.11 are generally intractable due to the complexity and high dimensionality of the posterior distribution $p(\theta|\mathcal{D})$. Even applying Bayes' theorem:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)\, p(\theta)}{p(\mathcal{D})}, \tag{2.12}$$

the marginal likelihood $p(\mathcal{D}) = \int p(\mathcal{D}|\theta)\, p(\theta)\, d\theta$ is typically infeasible to compute exactly.

2.2.1.1 Uncertainty in Data Distribution

Predictive uncertainty can also be categorized based on the relationship between the input data and the training data distribution:

- **In-Domain Uncertainty:** This occurs when the input comes from the same distribution as the training data but still yields uncertain predictions. It may arise due to aleatoric noise or limited model capacity.

- **Domain-Shift Uncertainty:** This arises when the input distribution has shifted slightly from the training distribution, such as through lighting changes, occlusions, or sensor drift. While some domain shifts can be anticipated and mitigated during training, others are more challenging to address.
- **Out-of-Domain Uncertainty:** This refers to inputs that lie entirely outside the training data distribution. For example, a classifier trained to distinguish between cats and dogs may receive an image of a bird. In such cases, predictions are highly unreliable, and uncertainty should be maximally high.

In general, epistemic uncertainty is present in all three cases due to limited model knowledge, while aleatoric uncertainty is specific to in-domain data. Understanding and modeling these uncertainty types is fundamental for building robust and trustworthy neural network models.

2.2.2 Approaches

This section provides an overview of key approaches for quantifying uncertainty in DNNs. While not exhaustive, a comprehensive survey can be found in Gawlikowski et al. (2022) and Abdar et al. (2021). The methods are grouped into four broad categories: Bayesian inference, ensemble methods, test-time data augmentation, and single-deterministic models.

2.2.2.1 Bayesian Neural Networks

Bayesian Neural Networks (BNNs) address uncertainty by treating network weights θ as random variables and placing a prior distribution over them. Given observed data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, the goal is to compute the posterior distribution over the weights using Bayes' theorem:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}. \tag{2.13}$$

The predictive distribution for a new input x^* is then given by marginalizing over this posterior:

$$p(y^*|x^*, \mathcal{D}) = \int p(y^*|x^*, \theta)\, p(\theta|\mathcal{D})\, d\theta. \tag{2.14}$$

Since this integral is analytically intractable for deep models, it is typically approximated via sampling. For instance, by drawing N samples $\theta_1, \ldots, \theta_N$ from an

approximate posterior, we estimate:

$$\hat{y}^* = \frac{1}{N} \sum_{i=1}^{N} f_{\theta_i}(x^*), \tag{2.15}$$

$$\sigma_{y^*} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(f_{\theta_i}(x^*) - \hat{y}^*\right)^2}. \tag{2.16}$$

These expressions yield estimates for the mean prediction and predictive uncertainty. The accuracy of this estimation improves with larger N due to the law of large numbers. To avoid the intractability of computing the exact posterior, a number of approximation strategies exist, including variational inference and Monte Carlo sampling.

Variational Inference. Variational inference methods approximate the true posterior $p(\theta|\mathcal{D})$ with a tractable family of distributions $q(\theta)$ by minimizing the Kullback–Leibler (KL) divergence:

$$\mathrm{KL}(q(\theta) \parallel p(\theta|\mathcal{D})).$$

A widely used example is **Monte Carlo Dropout** (MC Dropout) (Gal and Ghahramani, 2016*b*,*a*), where dropout layers are treated as stochastic Bernoulli-distributed variables. Training a neural network with dropout then becomes an instance of variational inference. During inference, multiple stochastic forward passes are performed with dropout activated, enabling uncertainty estimation without architectural changes or additional computational overhead. Gaussian dropout and other distributions can also be used in this framework.

Monte Carlo Sampling. Sampling-based methods directly approximate the posterior by generating samples from it. A prominent example is **Markov Chain Monte Carlo** (MCMC) sampling. These methods construct a Markov chain where each sample depends only on the previous one, and over time, the samples approximate the true posterior distribution. A commonly used MCMC algorithm is the Metropolis–Hastings algorithm (Duane et al., 1987). Once a collection of posterior weight samples $\{\theta_1, \theta_2, \ldots, \theta_N\}$ is obtained, uncertainty estimates are computed using the same predictive sampling method as described above. While theoretically accurate, MCMC methods are computationally expensive and scale poorly to large neural networks.

2.2.2.2 Ensemble Methods

Ensemble methods estimate uncertainty by training multiple models independently and aggregating their predictions. The ensemble predictive distribution for a new sample x^* is given by:

$$p(y^*|x^*, \mathcal{D}) \approx \frac{1}{M} \sum_{m=1}^{M} p(y^*|x^*, \theta_m),$$

where each θ_m corresponds to a different model trained with variations in initialization, data shuffling, or training subsets.

Although ensembles are often used to improve accuracy, they also provide strong predictive uncertainty estimates (Lakshminarayanan et al., 2017). Unlike BNNs, which rely on a single model that converges to a local optimum, ensembles exploit the diversity of solutions by combining multiple local optima.

The primary drawback of deep ensembles is their high computational and memory cost. However, studies such as Ovadia et al. (2019) show that even relatively small ensembles (e.g., five models) can achieve robust uncertainty estimation. To mitigate costs, techniques like **pruning** (Guo et al., 2018) and **distillation** (Buciluǎ et al., 2006) can be applied.

Model diversity can be enhanced via:

- Different random initializations and data shuffling,
- Bagging and boosting strategies,
- Data augmentation during training,
- Using varied network architectures (Renda et al., 2019).

2.2.2.3 Test-Time Data Augmentation

Test-Time Data Augmentation (TTA) is a simple yet effective strategy to estimate predictive uncertainty by introducing variability in the input space rather than in the model parameters. Instead of sampling model weights, TTA generates multiple augmented versions of the input x^* using stochastic transformations (e.g., flips, rotations, noise), and feeds them through the model:

$$\{T_1(x^*), T_2(x^*), \ldots, T_K(x^*)\} \rightarrow \{f(T_1(x^*)), \ldots, f(T_K(x^*))\}.$$

The resulting distribution of predictions is used to compute uncertainty. TTA is especially popular in domains like medical imaging (Wang et al., 2019), where data

is scarce and augmentation is already standard practice. However, the effectiveness of TTA depends heavily on the choice and number of augmentations used (Shanmugam et al., 2021), and it does not model uncertainty over the model's parameters.

2.2.2.4 Single Deterministic Methods

Single deterministic methods estimate uncertainty using a single, deterministic neural network that produces the same output for a given input. These methods typically require only a single forward pass, making them computationally efficient compared to Bayesian or ensemble-based approaches. However, they may be sensitive to architecture choices and training biases. They can be broadly divided into two categories based on how uncertainty is obtained.

Internal methods incorporate uncertainty estimation into the training process. The model is trained to output not only class predictions but also measures of uncertainty, often in the form of distributions over outputs. Examples include:

- **Dirichlet Prior Networks** (Malinin and Gales, 2018), which learn a Dirichlet distribution over class probabilities to capture both data and model uncertainty.
- **Evidential Deep Learning (EDL)** (Sensoy et al., 2018), which uses subjective logic to represent belief and uncertainty in predictions.
- **Radial Basis Function (RBF) Networks** (van Amersfoort et al., 2020), which estimate uncertainty based on the distance from known training samples — larger distances imply higher uncertainty.

On the other hand, **external methods** compute uncertainty after the model is trained, often by attaching auxiliary modules. For instance, the approaches in Raghu et al. (2019); Ramalho and Miranda (2019) train a separate neural network to predict the uncertainty associated with the outputs of the primary model. These methods avoid altering the architecture or training regime of the original network, making them easy to integrate, albeit potentially less accurate than methods that account for uncertainty during training.

2.3 Uncertainty and LNL

As mentioned in section 2.1.3, subsequent works based on DivideMix rely on incorporating information from features (via contrastive loss) or improving the sample selection process. However, some works have explored leveraging uncertainty in different ways and incorporate uncertainty in different parts of the DivideMix pipeline.

Zong et al. (2024) does not explicitly use uncertainty estimates but it incorporates the edl framework into the DivideMix pipeline. They replace the softmax layer for the edl adjustments mentioned in the previous chapter. They perform ablation studies and claim that the peak performance is achieved when the term added to the evidence is equal to 10/num. They also incorporate the regularization term given by the KL-divergence loss and find optimal values for it. The main contribution of the article is the proposal a novel fashion for splitting the data: instead of fitting the data with the loss distribution, they fit a distribution of margins (for a sample with noisy label "c", the margin is defined as the difference between the collected evidence for the label class "c" and the maximum evidence among the classes different from "c"). They achieve remarkable results, specially in the low noise rate scenario.

Other approaches explicitly use uncertainty to model and improve specific parts of the pipeline. Huang et al. (2022) proposes an Epistemic Uncertainty-aware Class-Specific Noise Modeling (EUCS), which separates noise modeling by class to account for differing loss distributions — the loss of clean samples might differ whether the sample belongs to a minority or majority class. It estimates epistemic uncertainty using MC-Dropout and combines it with loss values to better identify clean samples. Additionally, it corrects labels using a weighted mix of the original label and the model prediction. Moreover, it incorporates an Aleatoric Uncertainty-aware Learning (AUL) to model aleatoric uncertainty as logit corruption (adding Gaussian noise to the logits), which helps prevent overfitting to noisy labels in majority classes.

In Bayesian DivideMix++ (Nagarajan et al., 2024) a new mixmatch strategy is introduced: Monte-Carlo MixMatch, where the augmentations are based on the epistemic uncertainty of samples using MC dropout. This technique allows to weight uncertain samples differently during MixMatch, preventing overconfidence in noisy or ambiguous data. It performs particularly well in the high noise rate scenario.

The following works analyze uncertainty in LNL in general. Köhler et al. (2019) analyses the relationship between noisy samples and highly uncertain samples (here uncertainty is derived from MC dropout and ensemble). It highlights the counterintuitive fact that relabeling highly uncertain samples does not result in a better performance. Although the images identified as noisy are correctly relabeled, they simply are not helpful in making the network generalize and be more robust to the label noise. On the other hand, when these noisy images are relabeled with oracle relabeling[4] then the accuracy on validation set does increase accordingly. In Joo et al. (2020) a comparison of the generalization error is provided. It compares a traditional softmax layer against MC Dropout and a Dirichlet EDL Network. The evidence-based Dirichlet network achieves the best results.

[4]In oracle relabeling the noisy labels are replaced with the ground true labels.

Chapter 3

Methodology

In this chapter, we describe the methodological approach used to address the problem of learning under noisy supervision and quantifying uncertainty in deep learning models. The proposed pipeline is inspired by the DivideMix algorithm and EDL. In Section 3.1 we describe the motives supporting our methodological choices. Section 3.2 details the DivideMix algorithm while Section 3.3 provides an in-depth explanation of EDL and its interpretation of neural network outputs as Dirichlet distributions to capture both belief and uncertainty. The implementation details are found in Chapter 4.

3.1 Motivation of the method

The DivideMix algorithm was selected as the foundation of our training methodology due to its strong empirical performance and adaptability: the framework allows for straightforward integration of additional techniques such as robust loss functions, uncertainty estimation modules, and advanced data augmentations and robust regularization techniques. This makes it particularly suitable for our setting, where we aim to enhance noise robustness while also incorporating uncertainty quantification through EDL.

EDL was chosen for uncertainty quantification due to its ability to internally model both prediction and uncertainty within a single forward pass. Unlike ensemble methods or BNNs, which require multiple forward passes or multiple training procedures, EDL directly interprets network outputs as parameters of a Dirichlet distribution, allowing for both a principled and efficient uncertainty estimation and mitigation of overconfident predictions. Moreover, EDL does not impose additional computational overhead or require additional data, making it a practical

and theoretically grounded choice for enhancing the robustness of models trained under noisy labels.

3.2 DivideMix algorithm

As introduced in Section 2.1.3, DivideMix utilizes two networks (f_0 and f_1) that collaboratively train one another through multi-network learning. Training consists of two phases: a warm-up phase followed by semi-supervised learning (see Algorithm 1 and Figure 3.1).

Algorithm 1 DivideMix Algorithm

Require: Noisy dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, number of epochs E, warm-up epochs E_{warm}, batch size n, models f_0, f_1.

for $e = 1$ to E **do**
  if $e \leq E_{warm}$ **then**
    Train f_0 and f_1 independently using standard cross-entropy on D
  else
    for $i = 0, 1$ **do**
      Compute per-sample losses on D using f_i
      Fit a two-component GMM to the loss distribution
      Divide dataset into $\mathcal{D}_c^i$ (clean) and $\mathcal{D}_n^i$ (noisy) using Eq. 3.1
    end for
    for $i = 0, 1$ **do**
      for $j = 1$ to $|\mathcal{D}_c^{1-i}|/n$ **do**
        Sample n examples from $\mathcal{D}_c^{1-i}$ as C and from $\mathcal{D}_n^{1-i}$ as N
        Generate pseudo-labels for $C \cup N$ (Eq. 3.3 and 3.2)
        Apply MixMatch to $C \cup N$ (Eq. 3.4)
        Train f_i on the mixed batch using loss in Eq. 3.8
      end for
    end for
  end if
end for
return Trained models f_0, f_1

The two networks are initially trained with noisy labels during a warm-up phase, based on the premise that deep neural networks first learn general patterns before memorizing noise.

Following the warm-up, semi-supervised training begins. In this stage, each model is evaluated to compute the per-sample cross-entropy losses across the dataset.

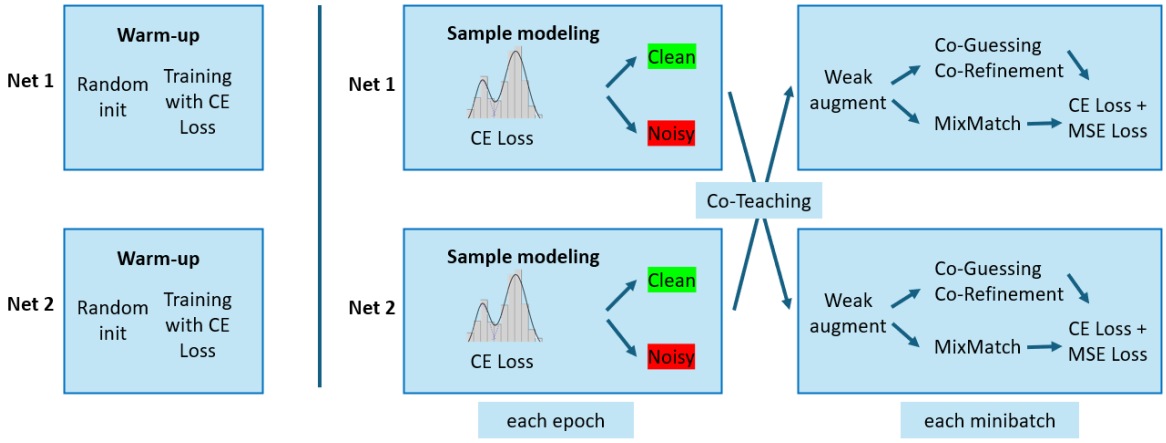


Figure 3.1: DivideMix pipeline.

These losses are modeled using a two-component Gaussian Mixture Model[1] (GMM) (Figure 3.2) to separate clean and noisy samples:

$$\begin{aligned} \textbf{Clean set:} \quad & \mathcal{D}_c = \{(x_i, y_i) \mid p_i > \tau\}, \\ \textbf{Noisy set:} \quad & \mathcal{D}_n = D \setminus \mathcal{D}_c, \end{aligned} \tag{3.1}$$

where p_i denotes the posterior probability that x_i belongs to the low-loss Gaussian.

A co-teaching strategy is then applied, where each model uses the clean and noisy partitions identified by the other network.

Clean and noisy samples are subsequently augmented using **MixMatch** (Berthelot et al., 2019), which includes MixUp and pseudo-labeling. Mini-batches are constructed to include balanced proportions of clean and noisy examples.

Now we denote by $\mathcal{D} = \mathcal{D}_c \cup \mathcal{D}_n$ the training batch composed by clean and noisy samples. For clean samples ($\mathcal{D}_c$), **label refinement** is performed using a combination of the original label and the average predictions across two augmentations — termed **co-refinement**:

$$\mathbf{y}_x = (1 - w_x)\frac{1}{2}\left(\text{softmax}(f(x_1)) + \text{softmax}(f(x_2))\right) + w_x c_x, \tag{3.2}$$

[1] A Gaussian Mixture Model (GMM) is a probabilistic model that assumes data is generated from a mixture of several Gaussian distributions with unknown parameters. In our case, two Gaussians are used: one representing clean samples and the other representing noisy samples. GMMs are commonly used for clustering, where each component models a distinct group in the data.

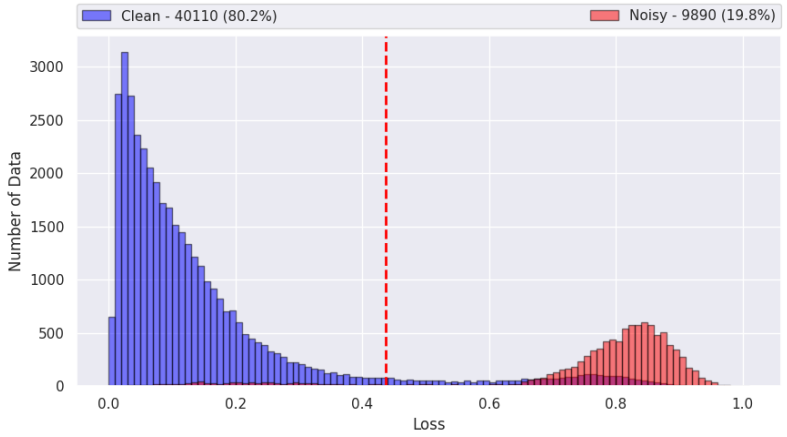


Figure 3.2: Histogram of per-sample loss: blue bins correspond to clean samples, red bins to noisy samples. The vertical line indicates the GMM-based threshold.

where x_1 and x_2 are two augmentations of sample x, c_x is its label, and w_x is the probability of x belonging to the low-loss gaussian according to the GMM. Finally, f is the trained network.

For noisy samples ($\mathcal{D}_n$), pseudo-labels are produced through **co-guessing** by averaging the predictions of both networks across two augmentations:

$$\mathbf{y}_u = \frac{1}{4}\left(\text{softmax}(f_0(u_1)) + \text{softmax}(f_0(u_2)) + \text{softmax}(f_1(u_1)) + \text{softmax}(f_1(u_2))\right), \tag{3.3}$$

where u_1 and u_2 are two augmentations of sample u.

Finally, both the co-guessed and co-refined labels go through a temperature sharpening step:

$$\mathbf{y}_T = \mathbf{y}^{1/T}.$$

MixUp is applied to blend clean and noisy samples indistinguishably. Let z_i be a sample of the training batch (either clean or noisy) and z'_i an augmentation of the sample; y_i its pseudo label (obtained with co-guessing or co-refinement), and Per a permutation over all the training batch. Then, mixed-up samples are generated as the linear interpolation of 2 random samples as follows:

$$\tilde{z}_i = \lambda z'_i + (1-\lambda) z'_{Per(i)}, \quad \tilde{y}_i = \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_{Per(i)}, \tag{3.4}$$

with $\lambda = max(\text{Beta}(\alpha,\alpha), 1-\text{Beta}(\alpha,\alpha))$.

We denote by $\tilde{x}$ and $\mathbf{y}_{\tilde{x}}$ the mixed-up clean samples and by $\tilde{u}$ and $\mathbf{y}_{\tilde{u}}$ the mixed-up noisy samples obtained from Eq. 3.4. And let $p_{\tilde{x}} = softmax(f(\tilde{x}))$ and $p_{\tilde{u}} = softmax(f(\tilde{u}))$.

The final loss combines supervised, unsupervised, and prior regularization terms of the mixed-up samples and the target labels.

$$\mathcal{L}_x = \text{CE}(\mathbf{p}_{\tilde{x}}, \tilde{y}_x), \tag{3.5}$$

$$\mathcal{L}_u = \text{MSE}(\mathbf{p}_{\tilde{u}}, \tilde{y}_u), \tag{3.6}$$

$$\mathcal{L}_{\text{prior}} = \sum_{c=1}^{C} \pi_c \log\left(\frac{\pi_c}{\bar{p}_c}\right), \tag{3.7}$$

$$\mathcal{L} = \mathcal{L}_x + \lambda \mathcal{L}_u + \mathcal{L}_{\text{prior}}, \tag{3.8}$$

where π is a prior uniform distribution and $\bar{p}$ are the model predictions on the mixed batch.

3.3 Evidential Deep Learning

Evidential Deep Learning (EDL) (Sensoy et al., 2018; Gao et al., 2024; Ulmer et al., 2023; Guo et al., 2024) is a principled framework for modeling uncertainty in neural networks by interpreting their outputs as subjective opinions over class probabilities. Standard deep learning models typically use softmax activation to produce a probability distribution over class labels. While softmax outputs can be interpreted as confidence scores, they often fail to reflect true uncertainty, especially in the presence of out-of-distribution samples — modeling the probabilities with a softmax assumes that the sample belongs to the labeled classes — or adversarial examples; and tend to inflate the probability of the predicted class.

EDL addresses this shortcoming by modeling predictions as parameters of a Dirichlet distribution, thereby explicitly capturing both the mean prediction and the uncertainty associated with it.

EDL draws from **Subjective Logic** (Jøsang, 2016), a type of probabilistic logic that explicitly models degrees of belief and uncertainty. Subjective logic is closely related to **Dempster-Shafer Theory (DST)** (Sentz and Ferson, 2002), a generalization of Bayesian probability theory that allows for reasoning under uncertainty by assigning belief masses to subsets of outcomes rather than to singletons.

In Subjective Logic, a **multinomial opinion** about a variable x in a finite domain

$$\mathcal{X} = \{x_1, x_2, \ldots, x_k\}$$

is represented as a triplet:

$$\omega_{\mathcal{X}} = (b_{\mathcal{X}},\, u_{\mathcal{X}},\, a_{\mathcal{X}})$$

where:

- $b_{\mathcal{X}}$ is the *belief mass distribution* over the elements of the domain, i.e., a vector

  $$b_{\mathcal{X}} = [b_1, b_2, \ldots, b_k]$$

  with $b_i \geq 0$ for all i, and $\sum_{i=1}^{k} b_i \leq 1$.

- $u_{\mathcal{X}}$ is the *uncertainty mass*, a scalar in the range $[0, 1]$, representing the total uncommitted belief mass.

- $a_{\mathcal{X}}$ is the *base rate distribution* (or prior), a probability distribution over $\mathcal{X}$, with

  $$a_{\mathcal{X}} = [a_1, a_2, \ldots, a_k], \quad \sum_{i=1}^{k} a_i = 1.$$

These components must satisfy the constraint:

$$\sum_{i=1}^{k} b_i + u_{\mathcal{X}} = 1. \tag{3.9}$$

The **projected probability distribution** $P(x_i)$ derived from the opinion is given by:

$$P(x_i) = b_i + u_{\mathcal{X}} \cdot a_i \tag{3.10}$$

This expresses that in the presence of uncertainty, the belief is distributed proportionally according to the base rates.

Given the base rates, there is a bijective correspondence between **multinomial opinions** in Subjective Logic and the **Dirichlet distribution**, which allows for seamless transitions between probabilistic reasoning and opinion-based reasoning.

A Dirichlet distribution over a categorical domain $\mathcal{X} = \{x_1, x_2, \ldots, x_k\}$ is denoted by:

$$\text{Dir}(\boldsymbol{\alpha}) \quad \text{with} \quad \boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_k), \quad \alpha_i > 0.$$

The probability density function (PDF) of the Dirichlet distribution is defined as:

$$f(\boldsymbol{p} \mid \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{k} p_i^{\alpha_i - 1} \quad \text{for } \boldsymbol{p} \in \Delta^{k-1},$$

where Δ^{k-1} is the $(k-1)$-dimensional probability simplex, and $B(\boldsymbol{\alpha})$ is the k-dimensional beta function.

The sum of the Dirichlet parameters is called the **Dirichlet strength**:

$$S = \sum_{i=1}^{k} \alpha_i. \tag{3.11}$$

Then, the correspondence between a Dirichlet Distribution and a multinomial opinion is given by:

$$\alpha_i = S \cdot b_i + a_i \tag{3.12}$$

$$b_i = \frac{\alpha_i - a_i}{S}, \tag{3.13}$$

This correspondence allows opinions to be interpreted probabilistically and a Dirichlet distribution to be interpreted as a mutinomial opinion.

In the context of a classification DNN we can actually think that the outputs of the network yield a Dirichlet Distribution and therefore we can treat them as multinomial opinions of subjective logic. In this case we have believes about the sample belonging to the classes of the dataset: "b_i" represents the belief mass assigned to the class "i". The evidence for each class can be represented as in Figure 3.3.

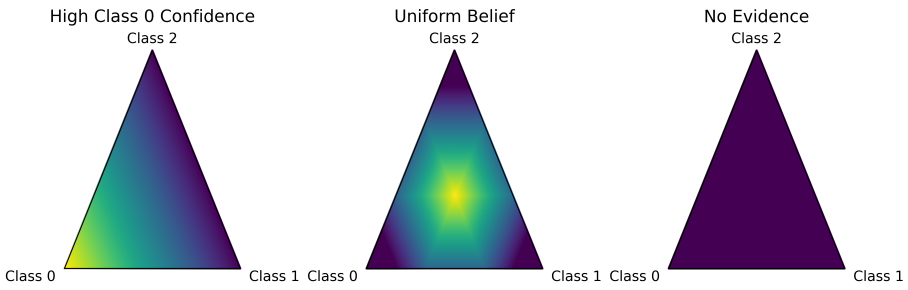


Figure 3.3: Examples of the evidence distribution across three classes. The second image represents vagueness (ambiguity between classes), while the third image represents vacuity (lack of evidence).

It is common to use a non-negative activation function (a) to the raw outputs of the network plus a positive scalar (W) representing the prior weight (base rate).

Let f denote a classification neural network and x a sample. Then the evidence and alpha parameters from the Dirichlet distribution are:

$$e = a[f(x)], \tag{3.14}$$

$$\alpha_i = e_i + W \tag{3.15}$$

Typically, $W \in \{1, \frac{10}{C}\}$ and $a \in \{softplus, exponential, ReLu\}$.

The belief mass assigned to each class is:

$$b_i = \frac{e_i}{S}, \quad \text{where} \quad S = \sum_{i=1}^{C} \alpha_i. \tag{3.16}$$

and the uncertainty is given by $u = 1 - \sum_i b_i$. It yields:

$$u = \frac{CW}{S}. \tag{3.17}$$

3.3.1 Uncertainty Quantification

There are several metrics quantifying the uncertainty of the predictions in the framework of EDL (Guo et al., 2024). Uncertainty can be measured from two perspectives: (1) distributional, using the Dirichlet distribution and its derived quantities such as entropy and mutual information, and (2) subjective logic, using concepts like vacuity and dissonance.

Entropy measures the expected uncertainty of the predicted categorical distribution — and therefore it is an estimate of the aleatoric uncertainty —, averaged over the Dirichlet distribution:

$$\text{Entropy}(x) = \mathbb{E}_{p \sim \text{Dir}(\boldsymbol{\alpha})} \left[-\sum_{i=1}^{C} p_i \log p_i \right] \tag{3.18}$$

In the case of the Dirichlet distribution the entropy takes the closed form (Ulmer et al., 2023):

$$\text{Entropy}(x) = -\sum_{i=1}^{C} \frac{\alpha_i}{\alpha_0} \left(\psi(\alpha_i + 1) - \psi(\alpha_0 + 1) \right) \tag{3.19}$$

where $\psi(\cdot)$ is the digamma function, α_i are the Dirichlet parameters, and $S = \sum_{i=1}^{C} \alpha_i$ is the Dirichlet strength.

Mutual Information captures the reducible part of predictive uncertainty and quantifies the expected reduction in entropy if the true model parameters were known. Therefore, it estimates the epistemic uncertainty. It is defined as:

$$\text{Mutual Information}(x) = H\left(\mathbb{E}[P(y|x)]\right) - \mathbb{E}\left[H(P(y|x))\right] \tag{3.20}$$

$$\text{Mutual Information}(x) = -\sum_{i=1}^{C} p_i \log p_i - \text{Entropy}(x). \tag{3.21}$$

Vacuity represents the uncertainty arising from a lack of evidence (i.e. it is the uncertainty of the subjective opinion):

$$\text{Vacuity}(x) = \frac{C}{S}. \tag{3.22}$$

High-vacuity is assigned to out-of-domain samples.

Dissonance is not related with a specific type of uncertainty. Instead, it quantifies the internal conflict among beliefs in different classes and is computed using the belief masses b_i:

$$\text{Dissonance}(x) = \sum_{i=1}^{C} \left(b_i \sum_{j \neq i} \frac{b_j \, \text{Bal}(j,i)}{\sum_{k \neq i} b_k} \right) \tag{3.23}$$

where the balance function between two classes i and j is defined as:

$$\text{Bal}(i,j) = 1 - \left| \frac{b_i - b_j}{b_i + b_j} \right| \tag{3.24}$$

Hyperopinion EDL (Li et al., 2024) is capable of computing **Vagueness**. Vagueness is related to vacuity and reflects the model's lack of confidence in any specific prediction. It increases when the model has seen little class-specific information.

Chapter 4

Proposed Method

In this chapter, we present our proposed Learning with Noisy Labels framework. Building upon the foundations of DivideMix-like architectures and recent advances in uncertainty-aware learning, our method integrates multiple components aimed at enhancing robustness and performance in the presence of label noise. These include Evidential Deep Learning for Uncertainty Modeling (Section 4.3), Contrastive Learning for improved feature representations and mitigation of memorization (Section 4.6), and a refined sample selection strategy that leverages both confidence margins and feature-space consistency (Section 4.4). We also propose an uncertainty-guided MixMatch (Section 4.5) for a better exploitation of uncertain samples. The following sections detail the architecture, providing overview of individual modules that compose our overall approach and the overall training procedure.

4.1 Overview of Our Method

This section provides a high-level description of our proposed framework. In Figure 4.1, we illustrate the overall pipeline of our framework, followed by a formal description in Algorithm 2.

Several key stages are designed to ensure robust training under label noise. We begin with a **self-supervised pretraining phase**, which provides a strong initialization for the model's backbone. This is followed by a **warm-up stage** tailored to the EDL framework, where the model begins learning to classify while avoiding early overfitting to noisy samples.

In the **semi-supervised learning phase**, we iteratively identify clean samples at each epoch using a dual distribution strategy based on the margin of evidence

and the distance to class prototypes in the feature space. Additionally, this clean set is then further refined to mitigate class imbalance.

In the co-teaching mechanism the clean and noisy sets are exchanged between networks. Pseudo-labels are assigned using co-guessing and co-refinement techniques, estimated from Dirichlet-based class probabilities, and we introduce a vacuity-aware MixMatch strategy to effectively use uncertain samples. Finally, we enhance representation learning by integrating a contrastive loss, which encourages better class separation in the feature space, and incorporate an EDL loss to promote uncertainty-awareness during training.

The detailed components of the approach are explained in the subsequent sections.

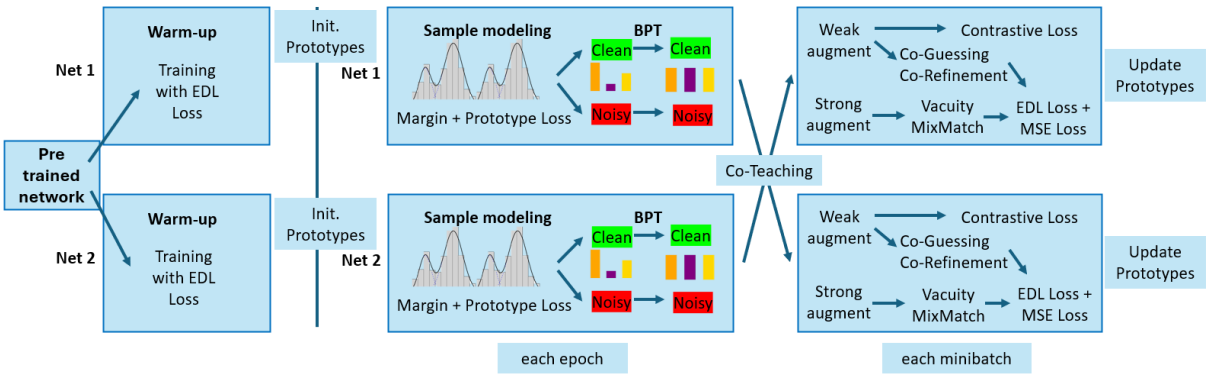


Figure 4.1: Our framework uses co-teaching of two models (Net-1 and Net-2), initialized with self-supervised pretraining. At each epoch, samples are split into clean and noisy sets using a dual criterion of evidence margin and prototype proximity, followed by class-balancing. The networks exchange these splits (Co-Divide), apply co-guessing and co-refinement based on Dirichlet distributions, and train using a vacuity-aware MixMatch strategy with EDL and contrastive loss.

Algorithm 2 Algorithm of Our Proposed Framework.

Require: Noisy dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, number of epochs E, warm-up epochs E_{warm}, batch size n, models f_0, f_1, pretrained weights w_i.

Initialize f_0 and f_1 with w_i
for $e = 1$ to E **do**
  if $e \leq E_{warm}$ **then**
    Train f_0 and f_1 independently using standard cross-entropy on D
    Initialize Prototypes for each class
  else
    for $i = 0, 1$ **do**
      Compute margins on D using f_i (Eq. 4.6)
      Compute Prototype loss on D using f_i (Eq. 4.8)
      Fit a 2 component 2D GMM with margins and loss (Eq. 4.11)
      Divide dataset into $\mathcal{D}_c^i$ (clean) and $\mathcal{D}_n^i$ (noisy) (Eq. 3.1)
      Refine $\mathcal{D}_c^i$ and $\mathcal{D}_n^i$ with BPT.
    end for
    for $i = 0, 1$ **do**
      for $j = 1$ to $(|\mathcal{D}_c^{1-i}| + |\mathcal{D}_n^{1-i}|)/n$ **do**
        Sample n examples from $\mathcal{D}_c^{1-i}$ (C) and n from $\mathcal{D}_n^{1-i}$ (N)
        Generate pseudo-labels for C and N (Eq. 3.3, 3.2)
        Apply Vacuity-MixMatch to $C \cup N$ (Eq. 4.14)
        Compute PLR Loss
        Compute EDL Loss
        Train f_i using loss in Eq. 4.15
      end for
    end for
    Update Prototypes
  end if
end for
return Trained models f_0, f_1

4.2 Self-supervised Pretraining

During the warm-up stage, training is fully supervised without distinguishing between clean and noisy samples. While neural networks initially learn general patterns, they begin to memorize noisy labels — a phenomenon that becomes more pronounced under high-noise conditions.

To improve generalization during the warmup stage, we initialize each of our models with self-supervised pretrained weights, following the works of Contrast2Divide (Zheltonozhskii et al., 2022). Pretraining helps the model learn high-level semantic representations and reduces the risk of overfitting to noisy labels, particularly

in high-noise settings during the warmup stage. Since these weights are obtained through unsupervised contrastive learning, they are inherently robust to label noise.

In particular, we use the **SimCLR** pre-trained model (Figure 4.2). SimCLR is a self-supervised learning framework[1] that learns visual representations by contrasting different augmented views of the same image. For each image in a batch, two augmented versions are generated, forming a positive pair. The goal of the model is to bring the representations of these positive pairs closer together in the feature space. At the same time, the model treats augmented views from different images in the batch as negative examples and pushes their representations apart. This encourages the model to learn features that distinguish between different images.

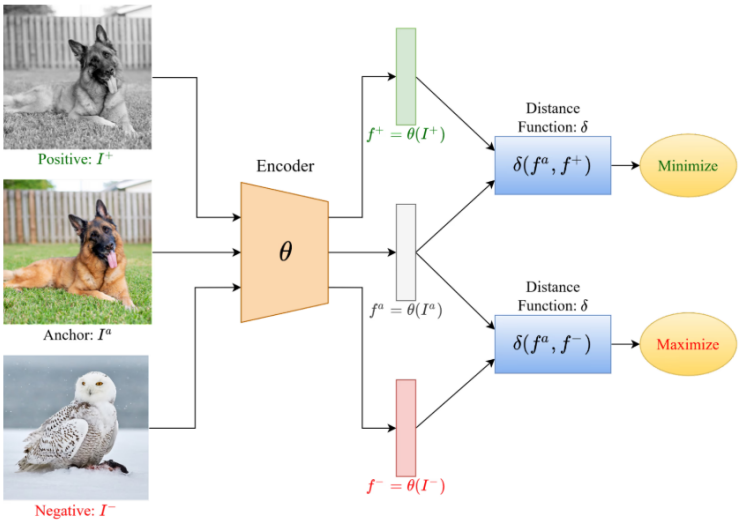


Figure 4.2: SimCLR loss pipeline (extracted from (V7 Labs, 2022)).

The training objective of SimCLR is a contrastive loss (Eq. 4.1) that maximizes similarity between positive pairs while minimizing similarity with all negative examples in the batch:

$$\mathcal{L}_{SimCLR} = \frac{-1}{2N}\sum_{i=1}^{N}\log\left(\frac{\exp\left(\frac{\mathbf{z}_i^\top \mathbf{z}_{i+N}}{\tau}\right)}{\sum_{j=1,j\neq i}^{2N}\exp\left(\frac{\mathbf{z}_i^\top \mathbf{z}_j}{\tau}\right)}\right) + \frac{-1}{2N}\sum_{i=N+1}^{2N}\log\left(\frac{\exp\left(\frac{\mathbf{z}_i^\top \mathbf{z}_{i-N}}{\tau}\right)}{\sum_{j=1,j\neq i}^{2N}\exp\left(\frac{\mathbf{z}_i^\top \mathbf{z}_j}{\tau}\right)}\right) \tag{4.1}$$

where:

[1] https://github.com/vturrisi/solo-learn

- $\mathbf{z}_i$ and $\mathbf{z}_{i+N}$ are L2-normalized feature vectors of two different augmentations of the i-th sample in a batch of size N.
- τ is the temperature hyperparameter.

4.3 EDL Implementation

Our implementation of Evidential Deep Learning follows the design discussed in Section 3.3. We replace the softmax output layer with an activation function a, and introduce a class prior of $10/C$, where C is the number of classes in the dataset.

The evidence, e, and Dirichlet parameters are computed as:

$$e = a[f(x)], \tag{4.2}$$

$$\alpha_i = e_i + \frac{10}{C}, \tag{4.3}$$

where f is the neural network, a is a non-negative activation function and α_i are the Dirichlet parameters (explained in Section 3.3).

Co-Refinement and Co-Guessing steps of our proposed method are modified to use the Dirichlet probabilities instead of the softmax probabilities:

$$\mathbf{y}_x = (1 - w_x)\frac{1}{2}\left(\alpha(x_{11}) + \alpha(x_{12})\right) + w_x c_x, \tag{4.4}$$

$$\mathbf{y}_u = \frac{1}{4}\left(\alpha(u_{11}) + \alpha(u_{12}) + \alpha(u_{21}) + \alpha(u_{22})\right), \tag{4.5}$$

where α represents the Dirichlet strength. The rest of the MixMatch is explained in Section 4.5.

The Dirichlet probabilities (Eq. 3.10) and the EDL loss are integrated throughout the various stages of our framework to promote uncertainty-aware learning. During the warmup stage, the models output Dirichlet distributions over classes instead of conventional softmax probabilities. The training at this stage is guided by the EDL loss, which replaces the standard cross-entropy loss.

In the semi-supervised stage, we incorporate the EDL framework consistently. For the supervised component (i.e., clean samples selected by the sample selection process), we use the Dirichlet probabilities as the models output and optimize them using the EDL loss. For the unsupervised component (i.e., noisy samples), we again rely on Dirichlet-based outputs, but employ the Mean Squared Error

(MSE) loss to align the predicted distributions with pseudo-labels. This loss helps regularize the learning process without forcing overconfident predictions.

4.4 Sample Selection

4.4.1 Modeling

Sample modeling each epoch is performed by leveraging information from two different distributions, namely, *Margins* and a *Prototype loss*.

4.4.1.1 Margins

Instead of modeling sample distributions using the prediction loss — as done in the original DivideMix and similar algorithms — we model it based on *margins*, inspired from the works of Zong et al. (2024). Margins are derived from the EDL evidence scores. Given the total evidence $\mathbf{e}$ output by the model for each class, the margin of a sample i is defined as the difference between the evidence assigned to the noisy-label class and the highest evidence assigned to any other class:

$$\text{margin}_i = e_{y_i} - \max_{j \neq y_i} e_j, \tag{4.6}$$

where e_{y_i} is the evidence for true class label and e_j denotes evidence for class j.

A high margin indicates strong evidence in favor of the labeled class in comparison to the other classes, providing a more interpretable and direct measure of confidence. In contrast, a low loss value — though often associated with clean samples — can be harder to interpret and does not necessarily imply that there is no conflict between competing class predictions; conversely, high loss may also arise from samples that are inherently harder to learn, not just mislabeled ones.

4.4.1.2 Prototype Loss

In addition to the evidence-based margins distribution, incorporating feature-space information offers a complementary perspective into sample selection. To this end, we include a Prototype Loss, inspired from BPT-PLR (Zhang et al., 2024), which measures the similarity between a sample's embedding and the learned prototype of its assigned class. These prototypes represent the average feature representation of each class and serve as anchors in the embedding space.

Let $\mathbf{q}_i \in \mathbb{R}^d$ be the normalized embedding of the i-th sample and $\mathbf{p}_j \in \mathbb{R}^d$ be the j-th prototype vector. The logits for sample i with respect to all prototypes are computed as:

$$\text{logits\_proto}_i = \left[\frac{\mathbf{q}_i^\top \mathbf{p}_1}{\tau}, \frac{\mathbf{q}_i^\top \mathbf{p}_2}{\tau}, \ldots, \frac{\mathbf{q}_i^\top \mathbf{p}_K}{\tau} \right], \tag{4.7}$$

where the prototype $\mathbf{p}_j$ of class j is initialized as the average feature vector of all samples predicted as class j, and then updated with momentum at each epoch.

The Prototype Loss is thus defined as:

$$\mathcal{L}_{\text{proto},i} = \text{CE}(\text{logits\_proto}_i, \text{label}(i)) \tag{4.8}$$

By evaluating how close a sample lies to its corresponding class prototype, the model gains another signal to assess sample selection. Clean samples are expected to cluster tightly around their class prototypes, while noisy samples tend to deviate.

4.4.1.3 2-D Gaussian Mixture Model

Finally, sample modeling consists of a 2-dimensional GMM (instead of a 1D GMM used widely) where both the margins and the prototype loss are fed to the GMM. To ensure comparability and stabilize the GMM training, both margins and prototype losses are normalized using min-max normalization.

$$\text{margin}_i^{\text{norm}} = 1 - \frac{\text{margin}_i - \min(\text{margin})}{\max(\text{margin}) - \min(\text{margin})}, \tag{4.9}$$

$$\mathcal{L}_{\text{proto},i}^{\text{norm}} = \frac{\mathcal{L}_{\text{proto},i} - \min(\mathcal{L}_{\text{proto}})}{\max(\mathcal{L}_{\text{proto}}) - \min(\mathcal{L}_{\text{proto}})}. \tag{4.10}$$

$$\text{GMM} = \text{GMM}\left(\left[1 - \text{margin}_i^{\text{norm}}, \mathcal{L}_{\text{proto},i}^{\text{norm}} \right] \right) \tag{4.11}$$

The posterior probabilities from the 2D GMM are then used to divide the dataset into clean and noisy subsets, with Eq. 3.1:

$$\begin{aligned} &\textbf{Clean set:} \quad \mathcal{D}_c = \{(x_i, y_i) \mid p_i > \tau\}, \\ &\textbf{Noisy set:} \quad \mathcal{D}_n = D \setminus \mathcal{D}_c, \end{aligned} \tag{4.12}$$

where τ is the threshold parameter, and p_i is the probability that sample i belongs to the low-value Gaussian component.

4.4.2 Balanced Partition Strategy (BPT)

A common issue in clean sample selection is class imbalance, which arises because the loss (or margin-based selection in our case) methods tend to favor certain classes over others. For instance, classes that are harder to learn or more easily confused with others often exhibit higher losses or lower evidence margins, causing their samples to be underrepresented in the selected clean set. This imbalance can bias the learning process since the clean samples are the ones that guide the training in sample selection models like ours.

To address this, we adopt the balanced partition process (Zhang et al., 2024). Clean samples selected from the loss modeling (Eq. 4.12) are further balanced across classes. If N samples are labeled as clean after GMM evaluation, the top N/C (where C is the number of classes) samples with the highest clean probabilities are selected for each class to mitigate class imbalance.

4.5 Vacuity-based MixMatch

We further leverage EDL's ability to quantify uncertainty by integrating vacuity — which represents uncertainty due to a lack of evidence — into the MixUp augmentations, following Bayesian DivideMix++ (Nagarajan et al., 2024), where Monte-Carlo based MixMatch was proposed. Vacuity captures epistemic (model) uncertainty, which is inherently reducible through additional learning. In our framework, samples with higher vacuity are given greater importance during MixUp training, based on the assumption that the model can still extract meaningful information from them. This encourages the network to accumulate more evidence for these uncertain samples, thereby improving its confidence over time.

In the standard MixMatch framework, the interpolation coefficient λ is sampled from a Beta distribution (see Eq. 3.4). We adopt an uncertainty-aware weighting strategy to emphasize uncertain samples more effectively. To incorporate this, we normalize the vacuity values across all samples using min-max normalization:

$$\text{weights}_{\text{vacuity}} = \frac{v - \min(v)}{\max(v) - \min(v)} \tag{4.13}$$

where v denotes the vacuity values of the samples (see Section 3.3.1). These normalized weights are then used to adjust the interpolation coefficient λ as follows:

$$\lambda_{vacuity} = 0.5 \cdot \left(1 + \lambda \cdot \text{weights}_{\text{vacuity}}\right), \tag{4.14}$$

where λ is as in Eq. 3.4.

This formulation increases the influence of low-evidence samples in the MixMatch procedure, encouraging the model to learn more robust representations from ambiguous data.

4.6 Contrastive Learning

Contrastive learning is integrated into our framework to enhance feature representations and mitigate the negative effects of noisy labels. During the warmup stage, the model may begin memorizing noisy samples, which can degrade overall learning. Moreover, the clean set obtained with the GMM process (Eq.4.12) may contain noisy samples even in low-noise setting. By leveraging contrastive learning, we encourage the model to learn more discriminative and robust features, helping to separate true class representations even in the presence of label noise.

We adopt the PLR approach (Zhang et al., 2024) to incorporate feature-based learning effectively. Unlike standard contrastive learning methods such as SimCLR, special care is taken in the selection of negative examples to avoid pushing apart samples that are likely to belong to the same class. Specifically, for a given sample i, any sample j in the minibatch is excluded from the negative set if the top-k predicted classes for i and j overlap. This prevents conflicting samples — which may be mislabeled or ambiguous — from being treated as negatives in the contrastive loss (i.e., excluded from the denominator in Eq. 4.1). For instance, if two samples are labeled with the same class, they are not used as negative examples of each other.

By excluding these potentially related samples from negative pairs, we reduce the risk of forcing the model to learn misleading feature boundaries. This refinement helps the model focus on meaningful differences in the feature space, improving class separation and overall robustness to noisy labels.

4.7 General Improvements

Additionally, we incorporate two general improvements to our proposed framework. Following the wide-spread adaption of data augmentation policies in works such as UNICON (Karim et al., 2022) and BPT-PLR (Zhang et al., 2024), we combine weak and strong augmentations in the training stage, instead of a single data augmentation scheme, so as to improve the performance of the Semi-supervised

Learning stage. A batch of data in the training step consists of 50% clean samples and 50% noisy samples. We perform a pair of weakly-augmented views and a pair of strongly-augmented view for each sample — the weak augmentations are used in the Co-Refinement and Co-Guessing processes (Eq. 3.3, Eq. 3.2) whereas the strong augmentations are used in the MixMatch algorithm (Eq. 3.4).

In the original DivideMix algorithm, the data is iterated only until the clean dataset is exhausted, making some data not seen at some epochs. We incorporate a modification in this process, where in, if either the clean set or the noisy set is exhausted first, we keep sampling from it until both sets are exhausted.

In summary, our framework has two improvements regarding data augmentation and training iterations compared to the original DivideMix algorithm — (1) an enhanced data augmentation policy and (2) more iterations are conducted in training to ensure all data is seen at each epoch. These strategies are adapted from widely used strategies in SOTA methods.

4.8 Training Objective

The total training loss $\mathcal{L}$ combines four components: a supervised EDL loss applied on clean samples, an unsupervised consistency loss applied on noisy samples, a contrastive loss term based on the PLR method, and a prior regularization term that promotes balanced class distributions.

$$\mathcal{L} = \mathcal{L}_{\text{sup}} + \lambda_{\text{unsup}}\mathcal{L}_{\text{unsup}} + \lambda_{\text{contr}}\mathcal{L}_{\text{contr}} + \lambda_{\text{prior}}\mathcal{L}_{\text{prior}}, \tag{4.15}$$

where λ_{unsup}, λ_{contr}, λ_{prior} are the weights of the unsupervised, contrastive and prior terms, respectively.

4.8.1 Supervised Loss

The supervised loss, $\mathcal{L}_{\text{sup}}$, is applied on clean labeled samples and combines a negative log-likelihood loss adapted for evidential deep learning with a regularization term based on the Kullback-Leibler (KL) divergence (Gao et al., 2024):

$$\mathcal{L}_{\text{sup}} = \mathcal{L}_{\text{nll-edl}} + \eta\mathcal{L}_{\text{kl}}, \tag{4.16}$$

where η controls the regularization strength.

Negative Log-Likelihood Loss:

$$\mathcal{L}_{\text{nll-edl}} = \frac{1}{|D|} \sum_{(x,y)\in D} \sum_{i\in\mathcal{X}} y_i \left(\log S - \log \alpha_i\right), \tag{4.17}$$

with

- D: clean labeled dataset,
- y_i: one-hot encoded ground truth label for class i,
- α_i: Dirichlet parameters predicted by the model,
- $S = \sum_i \alpha_i$: total evidence.

This encourages assigning high evidence α_i to the correct class.

KL Divergence Regularization:

$$\begin{aligned}\mathcal{L}_{\text{kl}} = \text{KL}[D(p \mid \tilde{\alpha}) \| D(p \mid \mathbf{1})] = \log\left(\frac{\Gamma\left(\sum_{j=1}^{K} \tilde{\alpha}_j\right)}{\Gamma(K)} \prod_{j=1}^{K} \frac{\Gamma(1)}{\Gamma(\tilde{\alpha}_j)}\right) \\ + \sum_{j=1}^{K} (\tilde{\alpha}_j - 1)\left[\psi(\tilde{\alpha}_j) - \psi\left(\sum_{k=1}^{K} \tilde{\alpha}_k\right)\right],\end{aligned} \tag{4.18}$$

where $\Gamma(\cdot)$ is the Gamma function and $\psi(\cdot)$ is the Digamma function.

The modified parameters $\tilde{\alpha}_j$ are:

$$\tilde{\alpha}_j = (\alpha_j - W)(1 - y_j) + 1, \tag{4.19}$$

which penalizes evidence from non-target classes and encourages confident predictions.

4.8.2 Unsupervised Consistency Loss

The unsupervised loss, $\mathcal{L}_{\text{unsup}}$, is applied on noisy samples to encourage consistency between model predictions $\mathbf{p}_u$ and pseudo-labels $\tilde{y}_u$:

$$\mathcal{L}_{\text{unsup}} = \text{MSE}(\mathbf{p}_u, \tilde{y}_u), \tag{4.20}$$

where MSE is the mean squared error between predicted probabilities and pseudo-labels.

4.8.3 Contrastive Loss

The self-supervised contrastive loss, $\mathcal{L}_{\text{contr}}$, encourages learning robust and discriminative features by contrasting positive and negative pairs in the PLR fashion, helping to separate different samples in the feature space and avoiding separating conflictive samples:

$$\begin{aligned}\mathcal{L}_{contr} = &- \frac{1}{2N} \sum_{i=1}^{N} \log \frac{\exp\left(\frac{\mathbf{z}_i^\top \mathbf{z}_{i+N}}{\tau}\right)}{\sum\limits_{\substack{j=1 \\ j \neq i \\ j \notin \mathcal{C}_k(i)}}^{2N} \exp\left(\frac{\mathbf{z}_i^\top \mathbf{z}_j}{\tau}\right)} \\ &+ \frac{1}{2N} \sum_{i=N+1}^{2N} \log \frac{\exp\left(\frac{\mathbf{z}_i^\top \mathbf{z}_{i-N}}{\tau}\right)}{\sum\limits_{\substack{j=1 \\ j \neq i \\ j \notin \mathcal{C}_k(i)}}^{2N} \exp\left(\frac{\mathbf{z}_i^\top \mathbf{z}_j}{\tau}\right)}\end{aligned}, \tag{4.21}$$

where:

- $\mathbf{z}_i$ and $\mathbf{z}_{i+N}$ are L2-normalized feature vectors of two different augmentations of the i-th sample in a batch of size N.
- τ is the temperature hyperparameter.
- $\mathcal{C}_k(i)$ is the set of samples in the minibatch whose top k predicted classes do not overlap with the top k predictions of i.

4.8.4 Prior Regularization Loss

The prior regularization loss $\mathcal{L}_{\text{prior}}$ promotes balanced class distributions by penalizing divergence between the estimated class prior $\bar{p}_c$ and the target class prior π_c. In this work, the target prior π_c is assumed to be a uniform distribution over all classes:

$$\mathcal{L}_{\text{prior}} = \sum_{c=1}^{C} \pi_c \log\left(\frac{\pi_c}{\bar{p}_c}\right), \quad \text{with} \quad \pi_c = \frac{1}{C}, \tag{4.22}$$

where C is the number of classes.

Chapter 5

Validation

In this chapter, we first define the evaluation methodology and benchmark datasets (Section 5.1) followed by the configuration of hyperparameters of our proposed framework (Section 5.2). In Section 5.3 we present our results and compare them with SOTA methods. We then provide an in-depth analysis of the sample selection procedure and the evolution of uncertainty (Section 5.5) in our models. Later, we describe the overall results in a broader perspective (Section 5.6) and discuss the limitations of our model (Section 5.7). Finally, we end the chapter by providing a brief analysis on the sustainability of the thesis (Section 5.8).

5.1 Evaluation and Datasets

In this work, we evaluate our method using the **CIFAR-100** (Krizhevsky, 2009) and **CIFAR-100N**[1] datasets, both of which are widely used benchmarks to study learning with label noise.

The CIFAR-100 dataset is a subset of the 80 million tiny images dataset and contains 60,000 color images of size 32×32. The dataset is evenly divided into 100 classes, with 600 images per class. Of these, 500 images per class (50,000 in total) are allocated for training, and the remaining 100 images per class (10,000 in total) are used for testing.

In our experiments, we use the CIFAR-100 dataset with synthetic label noise to evaluate robustness in controlled scenarios. We consider the following noise types:

- **Symmetric noise**: Each class label has a fixed probability of being flipped uniformly to any of the other classes, creating a symmetric and balanced noise pattern across all classes. We conduct evaluations under symmetric

[1] https://paperswithcode.com/dataset/cifar-100n

noise with noise rates of 20%, 50%, and 80%, following literature (Li et al., 2020; Cordeiro et al., 2023; Karim et al., 2022).

- **Instance-dependent noise (IDN)**: Labels are flipped according to a noise transition matrix that depends on the content of each individual image. Specifically, for each image, a flip probability vector is generated using a learned linear transformation of the image's pixel features. These probabilities are sampled from a truncated normal distribution centered around the target noise level (20%, 40%, and 60%), ensuring a diverse range of instance-dependent flip rates. The final transition probabilities are used to randomly sample a new (potentially incorrect) label, making the noise pattern more realistic and correlated with the visual features of each sample.

These controlled settings enable a thorough assessment of the model's robustness to both structured and realistic noise patterns.

To complement the synthetic noise experiments, we also evaluate on CIFAR-100N, a variant of CIFAR-100 with **real-world human annotation noise**. In this dataset, training labels are corrupted by replacing ground-truth labels with annotations collected from non-expert human annotators via Amazon Mechanical Turk. The noise rate is **40.20%**. The test set remains clean and identical to that of CIFAR-100.

We evaluate model performance from two perspectives:

- **Test accuracy**: Classification accuracy on the clean CIFAR-100 test set is used to measure generalization under noisy training conditions. It serves as the primary metric for assessing the model's performance.
- **Clean sample identification**: During training, we monitor the precision and recall of clean sample selection, providing insight into the model's ability to effectively separate clean from noisy labels. We use it to complement **Test accuracy** when necessary.

5.2 Experimental Setup

We utilize the training schedule of DISC (Li et al., 2023): 200 epochs, where the learning rate is reduced by a factor of 10 at epoch 80 and 160. The rest of training hyperparameters are shown in Table 5.1.

Table 5.1: Training hyperparameters.

Hyperparameter	Value
λ_{contr}	1
λ_{prior}	1
Alpha (beta distribution)	4
Temperature	0.5
GMM threshold (τ)	0.5
Num. epochs	200
Batch Size	64
Learning rate (epoch 0 -79)	0.02
Learning rate (epoch 80 -159)	0.002
Learning rate (epoch 160-200)	0.0002
Optimizer	SGD
Momentum	0.9
Weight decay	5e-4

5.2.1 Model Architecture

We adopt a **Pre-activation ResNet18 (PreResNet18)** architecture as the backbone for all our experiments. This design follows the residual network structure proposed by He et al. (2015), with the modification of pre-activation in each residual block — where batch normalization and ReLU activation precede the convolutional operations. The network comprises four stages of residual blocks, progressively increasing the number of channels while reducing spatial resolution. A global average pooling layer is applied to the final feature maps, followed by a fully connected layer that produces logits for classification over the target classes.

To support flexible learning objectives, the network architecture is extended to include a multi-head forward mechanism. Alongside the standard classification head, a projection head is introduced, implemented as a two-layer MLP with ReLU activation. This head maps the final feature vector to a lower-dimensional representation space, which is subsequently L2-normalized. This projection is used for contrastive representation learning and prototype-based classification (explained in section 4.6). We choose this architecture since it is a standard and widely adopted choice for experiments on CIFAR-10 and CIFAR-100, particularly in the context of LNL and therefore it allows us to fairly compare with other models (Li et al., 2020; Cordeiro et al., 2023; Li et al., 2023; Nagarajan et al., 2024; Zong et al., 2024). Following our proposed pipeline, we initialize two identical models.

5.2.2 Self-supervised Pretraining Setup

We leverage pretrained weights from the *Solo-learn* library (da Costa et al., 2022), a modular and comprehensive framework that provides standardized implementations of SOTA self-supervised learning (SSL) methods (no label information is used). It includes a repository of pretrained models with SimCLR, BYOL, MoCoV2+, DINO, and others, trained on standard datasets like CIFAR-10, CIFAR-100, and ImageNet-100 with strong performance in both online and offline linear evaluation tasks. All models are trained for **1000 epochs** using a batch size of 256, an SGD optimizer with LARS, and a temperature parameter $\tau = 0.5$. The training pipeline utilizes Nvidia DALI for faster data loading and mixed-precision training for improved efficiency. As stated before, the pretrained SSL weights for our framework are trained using **SimCLR** method.

We made two consequent modifications due to the usage of pretrained weights following the findings of Bayesian DivideMix++ (Nagarajan et al., 2024). First, we adjust the warm-up phase of our model to **5 epochs** (Warm up typically lasts 30 epochs for CIFAR-100 in DivideMix). This decision is motivated by the fact that pretrained models already provide meaningful representations, allowing the model to distinguish clean from noisy samples earlier in training; and to avoid unlearning the self-supervised weights completely or overfitting to noisy labels.

Second, we strengthen the unsupervised weight during training and we adopt the weights from Nagarajan et al. (2024) (Table 5.2) - which are remarkably higher than the ones used in models than do not use pretrained weights.

Table 5.2: Unsupervised weights.

Noise	20%	40%	40.2% (CIFAR-100N)	50%	60%	80%
λ	25	150	150	150	150	500

5.2.3 EDL and PLR Loss hyperparameters

Based on preliminary experiments to tune the EDL architecture, we select the *exponential* activation function and set the KL-divergence regularization weight to $\eta = 0.01$. Table 5.3 summarizes the full set of hyperparameters and architectural choices used in our EDL implementation. Additionally, Table 5.4 details the hyperparameters adopted for the PLR loss.

Table 5.3: EDL hyperparameters.

Parameter	Value
Activation function	Exponential
W	10/C
EDL Loss	Negative Log-likelihood + KL div
η	0.01

Table 5.4: PLR (SimCLR-style) contrastive learning hyperparameters. The number of overlapping classes k for a sample to be excluded as a negative example is 3 until epoch 40, 2 until epoch 70 and 1 from there, following the BPT-PLR implementation (Zhang et al., 2024).

Parameter	Value
τ	0.05
Size of feature space	128
k (num of overlapping classes)	3,2,1

5.3 Results

Table 5.5 shows the results under symmetric noise at various noise rates (20%, 50%, and 80%). Table 5.6 presents results under instance-dependent noise (20%, 40%, and 60%). Finally, Table 5.7 reports performance on the CIFAR-100N dataset, which contains real-world label noise.

We primarily base our performance analysis on the symmetric noise setting (Table 5.5), as relatively few competing methods report results under instance-dependent noise or on the CIFAR-100N dataset.

Our method delivers strong performance across all noise levels, especially under high-noise conditions. At high-noise settings (60% and 80%), it outperforms all baselines, surpassing the closest competitors in different noise setting by 2.91% (Bayesian DivideMix++ (Nagarajan et al., 2024), which also leverages pretrained weights and uncertainty) and by 7.24% (SplitNet (Kim et al., 2025)). In medium noise-rate settings (40% and 50%), our model also outperforms the competitors, but the gap is closed. Finally, although our approach slightly trails SOTA methods under low noise (20%), it remains competitive.

On the real-world CIFAR-100N benchmark (Table 5.7), our model achieves results comparable to the SOTA. The comparison with Li et al. (2020); Zong et al. (2024) is somewhat unfair because our experiments use a ResNet-18 backbone,

Table 5.5: Test Accuracy (%) on CIFAR-100 under Symmetric Noise. *Best* indicates the highest accuracy achieved during training, while *Last* is the average accuracy over the final 10 epochs.

Methods	20% Sym Noise		50% Sym Noise		80% Sym Noise	
	Best	Last	Best	Last	Best	Last
DivideMix Li et al. (2020)	77.30	76.90	74.60	74.20	60.20	59.60
LongReMix Cordeiro et al. (2023)	78.61	78.10	75.87	75.84	62.24	61.60
BPT-PLR Zhang et al. (2024)	78.85	78.66	78.02	77.77	69.31	69.06
DPC Zong et al. (2024)	**81.0**	-	78.5	-	66.4	-
CCLM Tatjer et al. (2024)	70.2	-	76.6	-	67.1	-
ULC Huang et al. (2022)	77.3	77.1	74.9	74.3	61.2	60.8
Bayesian DivideMix++ Nagarajan et al. (2024)	80.02	**79.56**	78.31	77.71	70.01	69.55
Ours	79.03	78.74	**78.72**	**78.43**	**72.92**	**72.64**
Difference	−1.97		+0.20		+2.91	

while they rely on a larger ResNet-34, which typically yields better accuracy. Comparing it with CCLM (Tatjer et al., 2024) with the same backbone architecture, our model yields better results and is more consistent through the last epochs of training.

5.4 Selection of Hyperparameters

5.4.1 Tuning of EDL

In order to tune the EDL activation function and loss, we test the following values for $\eta : 0.001, 0.01, 0.1$; and we test *softplus* and *exp* as activation functions.

5.4.1.1 Experimental Setup

All subsequent experiments are carried out with the original DivideMix algorithm using the experimental setup explained in section 5.2 for 100 epochs instead of 200. For the best-performing models of each method, we do provide the results on the complete training procedure (200 epochs) in Table 5.10. The EDL implementation is aligned with section 4.3. Since no pretrained weights are used, the warmup is

Table 5.6: Test Accuracy (%) on CIFAR-100 under Instance-dependent Noise.

Methods	20% IDN		40% IDN		60% IDN	
	Best	Last	Best	Last	Best	Last
DISC Li et al. (2023)	80.12	-	78.44	-	69.57	-
SL Kim et al. (2024)	**80.94**	-	78.60	-	-	-
SplitNet Kim et al. (2025)	80.45	-	76.97	-	70.20	-
Ours	78.85	78.69	**78.84**	78.61	**77.44**	77.24
Difference	–2.09		+0.24		+7.24	

Table 5.7: Test Accuracy (%) on CIFAR-100N. Methods marked with * use a different backbone network (ResNet-34) than ours.

Methods	Best	Last
DivideMix* Li et al. (2020)	71.13	-
DPC* Zong et al. (2024)	**71.42**	-
CCLM Tatjer et al. (2024)	70.6	69.7
Ours	70.78	70.55

kept to 30 epochs and the unsupervised weights are those used in PSSCL (Zhang et al., 2025) (Table 5.8).

5.4.1.2 Results

Tables 5.9 and 5.11 show the results of tuning the EDL architecture, focusing on different activation functions and KL-divergence loss weights. Table 5.10 presents a comparison of the final chosen EDL architecture against the standard CE method.

We start by comparing the overall performance trends of EDL against the standard Cross-Entropy (CE) loss. Notably, all EDL variants outperform the CE baseline under low (20%) and moderate (50%) noise levels, although the performance gap narrows as noise increases. Interestingly, under the high noise setting (80%), CE surpasses all EDL configurations.

The 80% noise scenario offers intriguing insights. Despite EDL models initially attaining substantially higher precision and recall for clean samples post warm-up, CE eventually outperforms EDL during semi-supervised training. We hypothesize

Table 5.8: Training hyperparameters.

Sym noise	20%	50%	80%
λ	0	25	25

Table 5.9: Test Accuracy (%) on CIFAR-100 under Symmetric Noise after 100 epochs. Tuning of KL-divergence weight and the distribution used to split the data. We denote unsuccessful experiments with a '-'.

Methods	Activation	Weight	20% Sym	50% Sym	80% Sym
CE	Softmax	-	75.40	74.63	**63.76**
EDL	Softplus	0.001	-	-	-
EDL	Softplus	0.01	**78.27**	74.58	54.85
EDL	Softplus	0.1	77.11	75.77	61.63
EDL	Exp	0.001	76.38	75.36	61.71
EDL	Exp	0.01	76.21	**75.80**	63.40
EDL	Exp	0.1	-	-	-

that EDL's evidential learning framework struggles to "unlearn" or discard mislabeled clean samples, limiting its ability to adapt when the noise rate is extremely high and the clean set becomes less reliable.

A further distinction is observed in the precision and recall metrics for clean sample identification (Table 5.11). The EDL methods consistently achieve higher recall without compromising precision, especially at lower noise levels. This suggests EDL's advantage in effectively retaining clean samples during training.

Regarding activation functions, Softplus shows superior performance in low-noise scenarios, with gains of nearly 2% percentage points over other alternatives. The elevated recall achieved with Softplus likely contributes to this advantage. Conversely, the Exponential activation demonstrates greater robustness and consistency across varying noise rates, performing better in higher noise settings.

Concerning the KL-divergence weight, smaller values favor performance in low-noise conditions, while increased weighting benefits high-noise scenarios by

Table 5.10: Test Accuracy (%) on CIFAR-100 under Symmetric Noise. Comparison of EDL Loss and CE Loss.

Methods	Activation	Weight	20% Sym	50% Sym	80% Sym
CE	Softmax	-	78.36	76.93	**66.02**
EDL	Exp	0.01	**79.09**	**77.76**	64.64

Table 5.11: Precision and Recall of clean labels (%) and accuracy of the unlabeled samples (%) on CIFAR-100 under Symmetric Noise (20% and 80%) at epoch 30 and 100.

Methods	20% Sym			80% Sym		
	Precision	Recall	Acc	Precision	Recall	Acc
CE (epoch 30)	94.67	82.87	25.09	65.97	48.49	20.64
CE (epoch 100)	99.79	80.42	38.94	83.09	85.40	56.05
EDL Softplus (epoch 30)	98.09	90.77	42.22	70.55	69.06	27.62
EDL Softplus (epoch 100)	99.13	93.48	54.91	75.59	89.41	54.06
EDL Exp (epoch 30)	98.42	86.09	33.06	72.12	57.20	23.04
EDL Exp (epoch 100)	99.69	85.38	43.09	76.26	89.53	52.06

strengthening regularization and mitigating label noise impact. Two configurations stand out: (1) Softplus with a KL weight of 0.01 excels in low-noise settings, and (2) Exponential with the same weight offers consistent performance across all noise levels, outperforming CE except at 80% noise where it trails slightly.

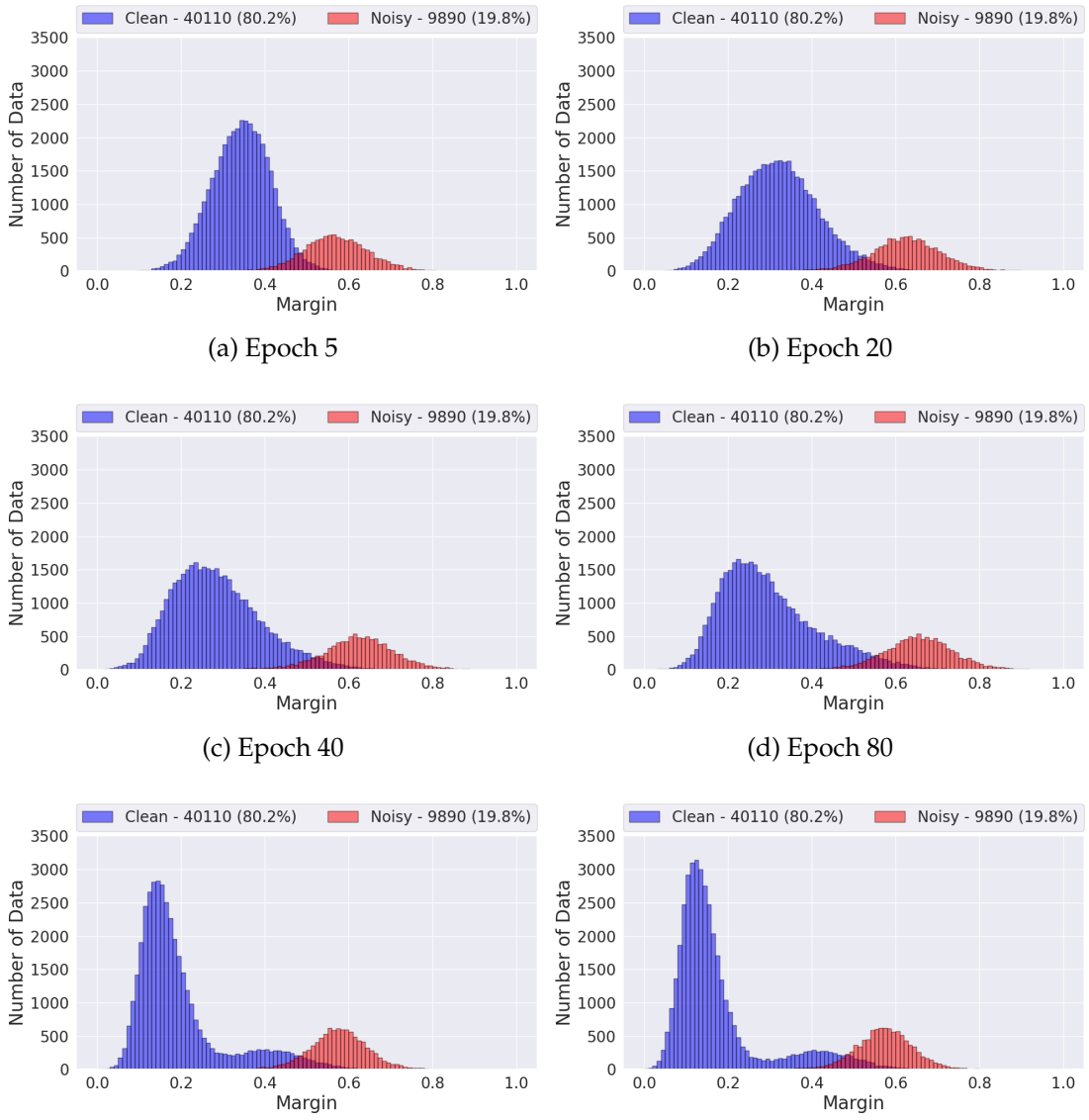
In summary, the EDL approach is effective at obtaining a higher-quality clean set after the warm-up phase, excelling in both precision and recall. However, the CE approach demonstrates a greater ability to recover during semi-supervised training even when starting with a noisier clean set, as evidenced by the 80% noise scenario. Nevertheless, EDL methods clearly outperform CE under moderate noise levels (20% and 50%), though careful tuning of the KL-divergence weight remains crucial to achieving optimal results.

5.5 Analysis

In this section, we conduct a comprehensive analysis of the training dynamics and model behavior. We consider two noise regimes: 20% symmetric noise (representing a low-noise scenario) and 80% symmetric noise (representing a high-noise scenario). The analysis focuses on three key aspects: (1) sample modeling, which explores how margin distributions and prototype loss distributions evolve; (2) vacuity, which measures the model's epistemic uncertainty; and (3) dissonance, which captures internal conflict in belief over classes. Together, these perspectives provide a detailed understanding of the learning process under varying levels of label noise.

Throughout this section, we use the metrics obtained from one (network 1) of the two networks trained in our model.

5.5.1 Sample modeling



(a) Epoch 5

(b) Epoch 20

(c) Epoch 40

(d) Epoch 80

(e) Epoch 140

(f) Epoch 200

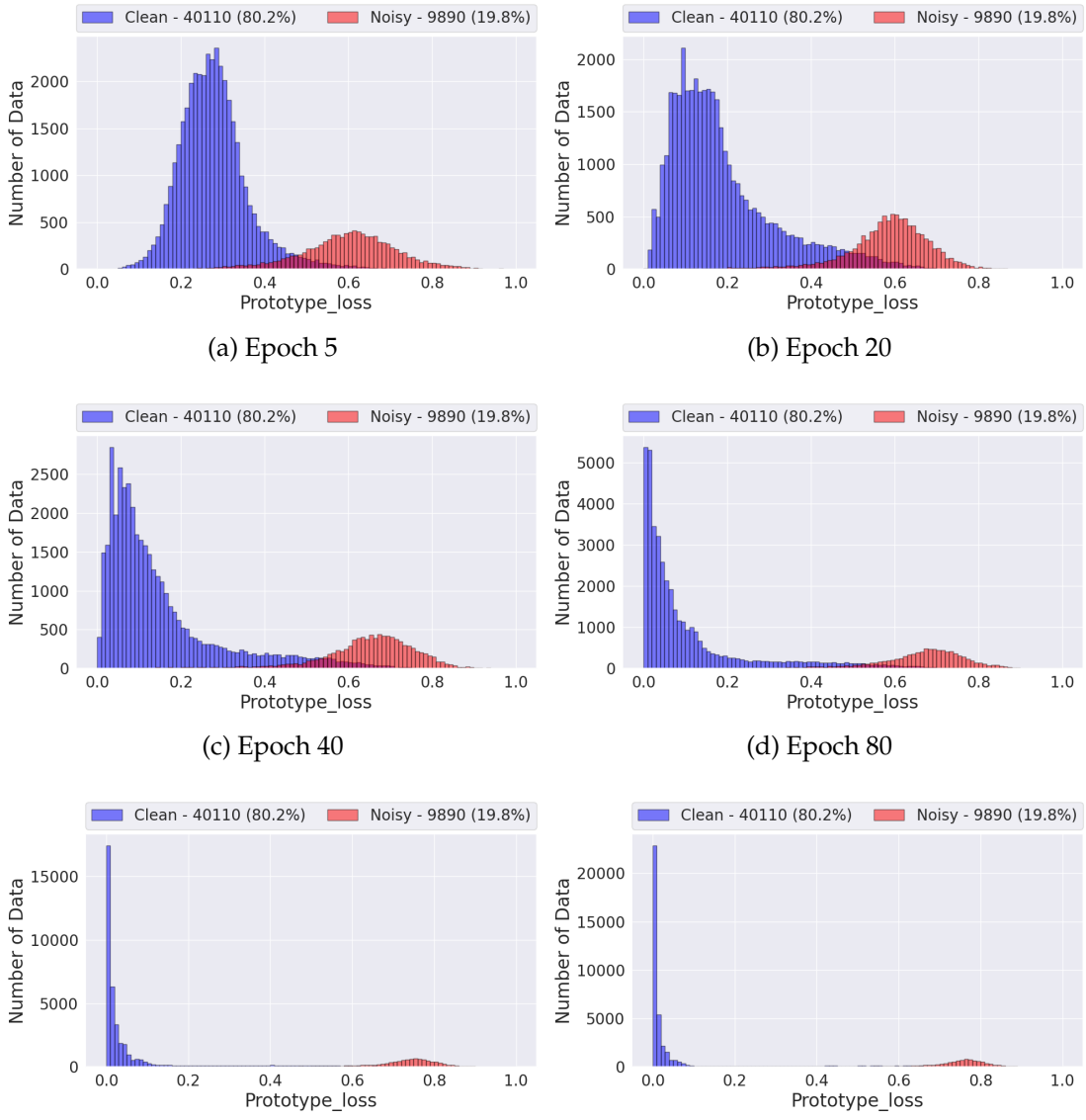
Figure 5.1: Margins histograms at various training epochs (20% symmetric noise).

We analyze how our model handles sample modeling. Specifically, we examine the evolution of the two distributions used to model the data within the GMM: the inverse of the margins (Eq. 4.6) and the prototype loss (Eq. 4.8). For simplicity, we

refer to the distribution of the inverse margins simply as the margin distribution throughout the analysis.

In the analysis we distinguish between the true clean and noisy samples based on a post-hoc analysis, meaning this separation is known only after training and not available to the model during training. This retrospective labeling allows us to better understand how the model behaves with respect to clean versus noisy data.

In Figures 5.1 and 5.2 we show the distributions in the 20% symmetric noise setting.



(a) Epoch 5

(b) Epoch 20

(c) Epoch 40

(d) Epoch 80

(e) Epoch 140

(f) Epoch 200

Figure 5.2: Prototype loss histograms at various training epochs (20% symmetric noise).

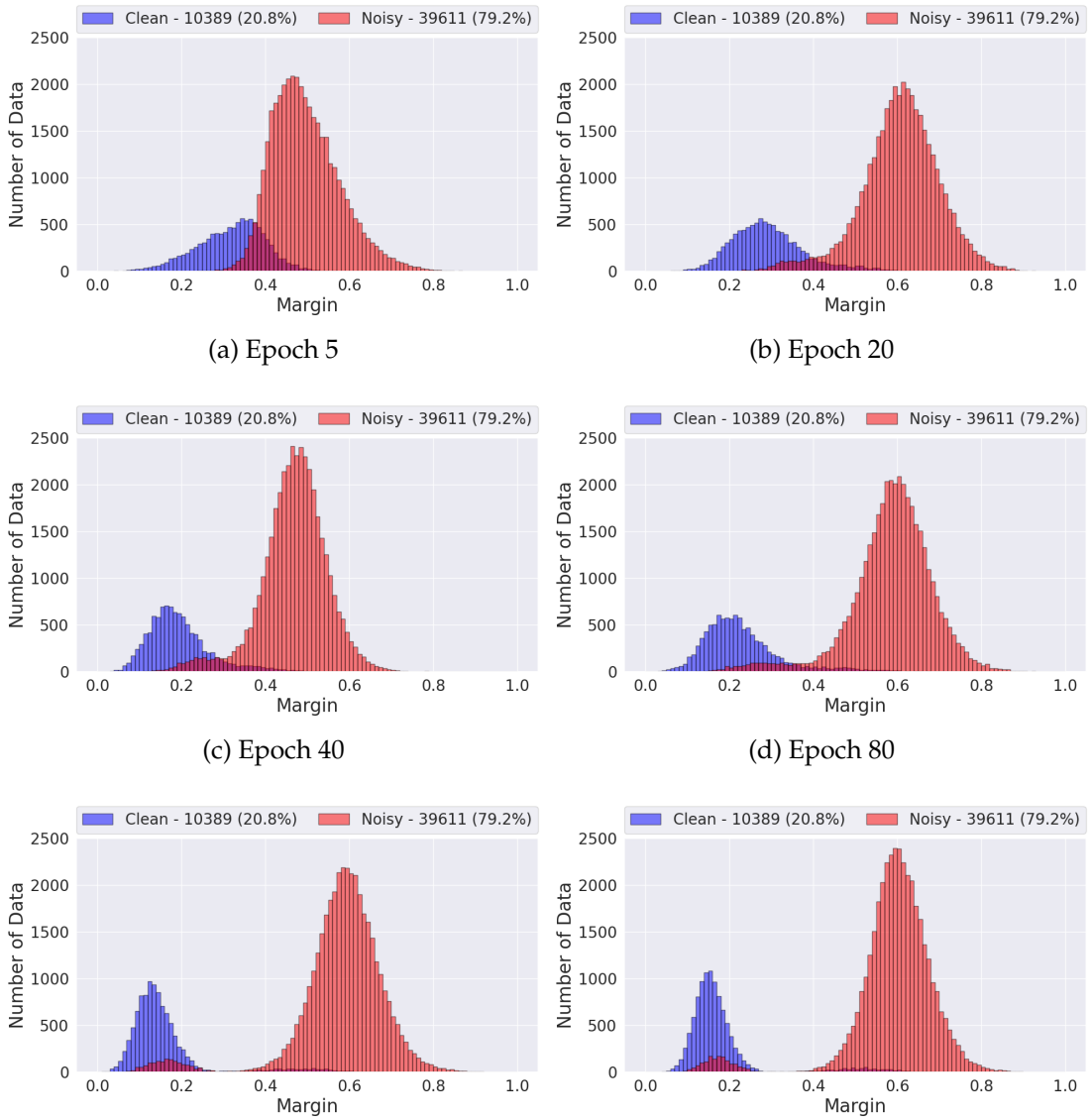
Under low noise, the distributions exhibit a strong Gaussian-like shape from the beginning. Both the margin distributions and prototype loss distributions show a clear separation between clean and noisy samples just after the warm-up phase. This early structure reflects the model's ability to effectively identify and prioritize clean samples for supervised learning.

It is also noticeable that clean samples not selected for supervised training — due to imperfect recall in the filtering process — exhibit a characteristic behavior. In the low-noise setting, these unselected clean samples form a tail-like extension to the right of the main low-loss Gaussian peak in the prototype loss distribution. This tail becomes more pronounced over time, indicating a gradual divergence between clean samples that are included in supervised training and those consistently excluded.

There are also important distinctions between the margin and prototype loss distributions themselves. While both reflect clean-noisy separation early in training, they diverge more noticeably in later stages. The prototype loss distribution for clean samples tends to collapse sharply toward the minimum value, forming a narrow, peaked distribution — suggesting increasingly confident and consistent prototype-based representations. In contrast, the margin distribution for clean samples remains broader, reflecting the variability in the margin scores across classes and samples. This broader spread is likely due to the pairwise nature of margin computations, which are sensitive to local decision boundaries and intra-class variation; while the computation of prototypes is conditioned on the features of each sample independently.

Finally, as training progresses, the gap between the two Gaussian-like modes — corresponding to clean and noisy samples — becomes more pronounced. This separation is primarily driven by the supervised component of the loss, which explicitly encourages the model to minimize prototype loss and maximize margins for clean samples. In contrast, the unsupervised loss component, being based on the model's own predictions rather than labels, does not exert the same pressure, and thus contributes less to this sharpening of the distributions.

In Figures 5.3 and 5.4 we show the evolution of the distributions over the course of training for 80% symmetric noise.



(a) Epoch 5

(b) Epoch 20

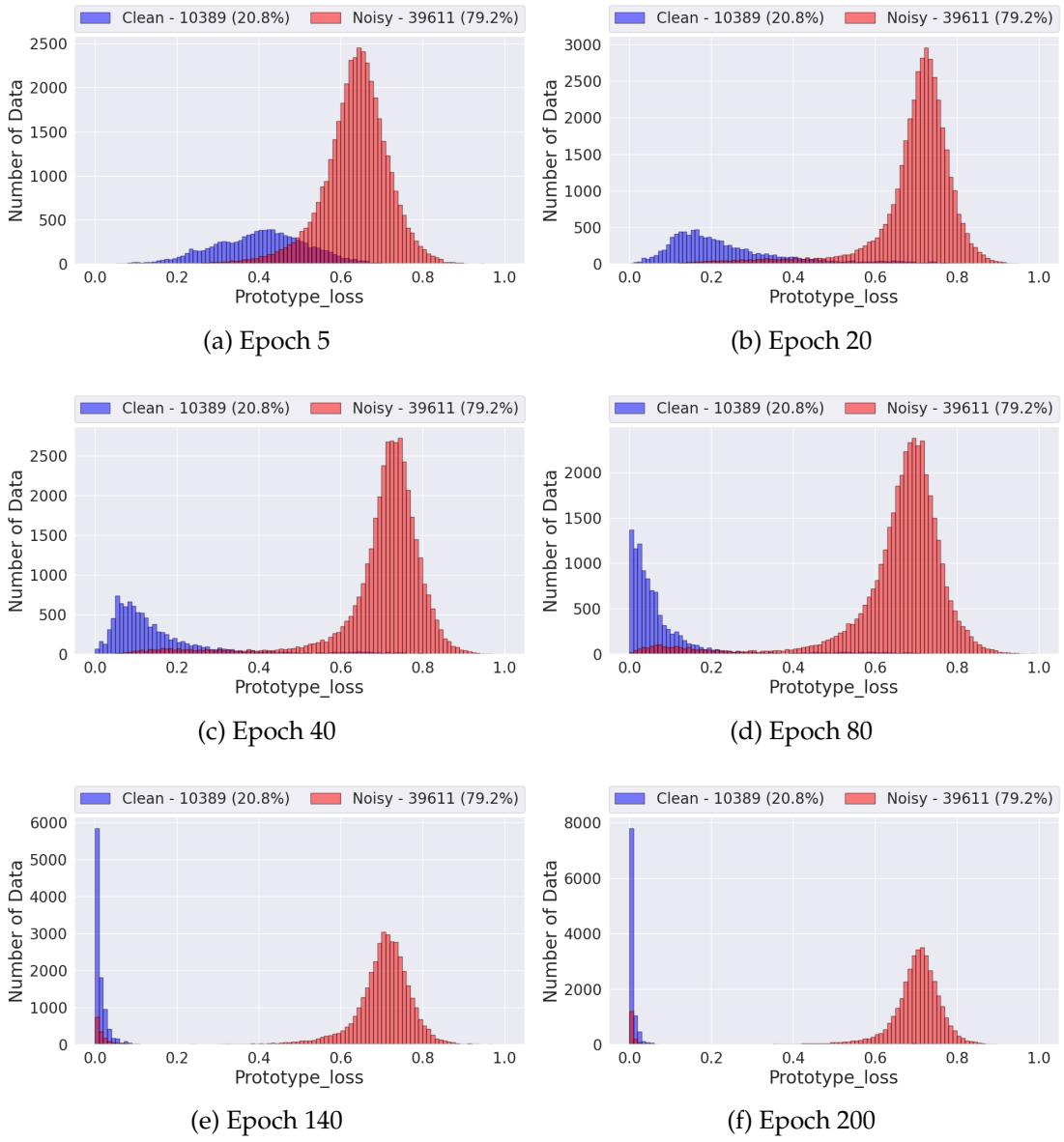
(c) Epoch 40

(d) Epoch 80

(e) Epoch 140

(f) Epoch 200

Figure 5.3: Margins histograms at various training epochs (80% symmetric noise).



(a) Epoch 5

(b) Epoch 20

(c) Epoch 40

(d) Epoch 80

(e) Epoch 140

(f) Epoch 200

Figure 5.4: Prototype loss histograms at various training epochs (80% symmetric noise).

When comparing training under low and high noise conditions, clear differences emerge in the evolution of both the margin and prototype loss distributions, particularly during the early stages of training.

Opposite to the low-noise setting, under high noise conditions, the distributions take longer to develop a clear structure. In the early epochs, the shapes are more diffuse and deviate from a well-defined Gaussian form, particularly the clean samples. Although clean samples still tend to occupy the lower prototype loss and low margin regions, the distributions do not separate as distinctly as in the low-

noise setting. Another key difference is the onset of overfitting and memorization: noisy samples that are incorrectly selected as clean are initially positioned near the boundary of the low-loss region (i.e., the right side of the lower Gaussian), but as training progresses, the model begins to fit these mislabeled instances. These noisy samples are gradually pulled toward the mean of the lower prototype loss and margin region, ultimately overlapping with the clean distribution in both prototype loss and margin space.

The differences in behavior between the margin distribution and the prototype loss distribution are similar to those observed under 20% symmetric noise, where the prototype distribution develops a prominent peak in the low-loss region during the final training stages, while the margin distribution continues to resemble the two-Gaussian pattern.

5.5.2 Uncertainty

In this section, we analyze the behavior of uncertainty throughout the training pipeline of our model. Leveraging EDL, we quantify uncertainty using two complementary measures: *vacuity* (Eq. 3.22), which captures the model's lack of evidence and reflects epistemic uncertainty, and *dissonance* (Eq. 3.23), which quantifies internal conflict among competing class hypotheses.

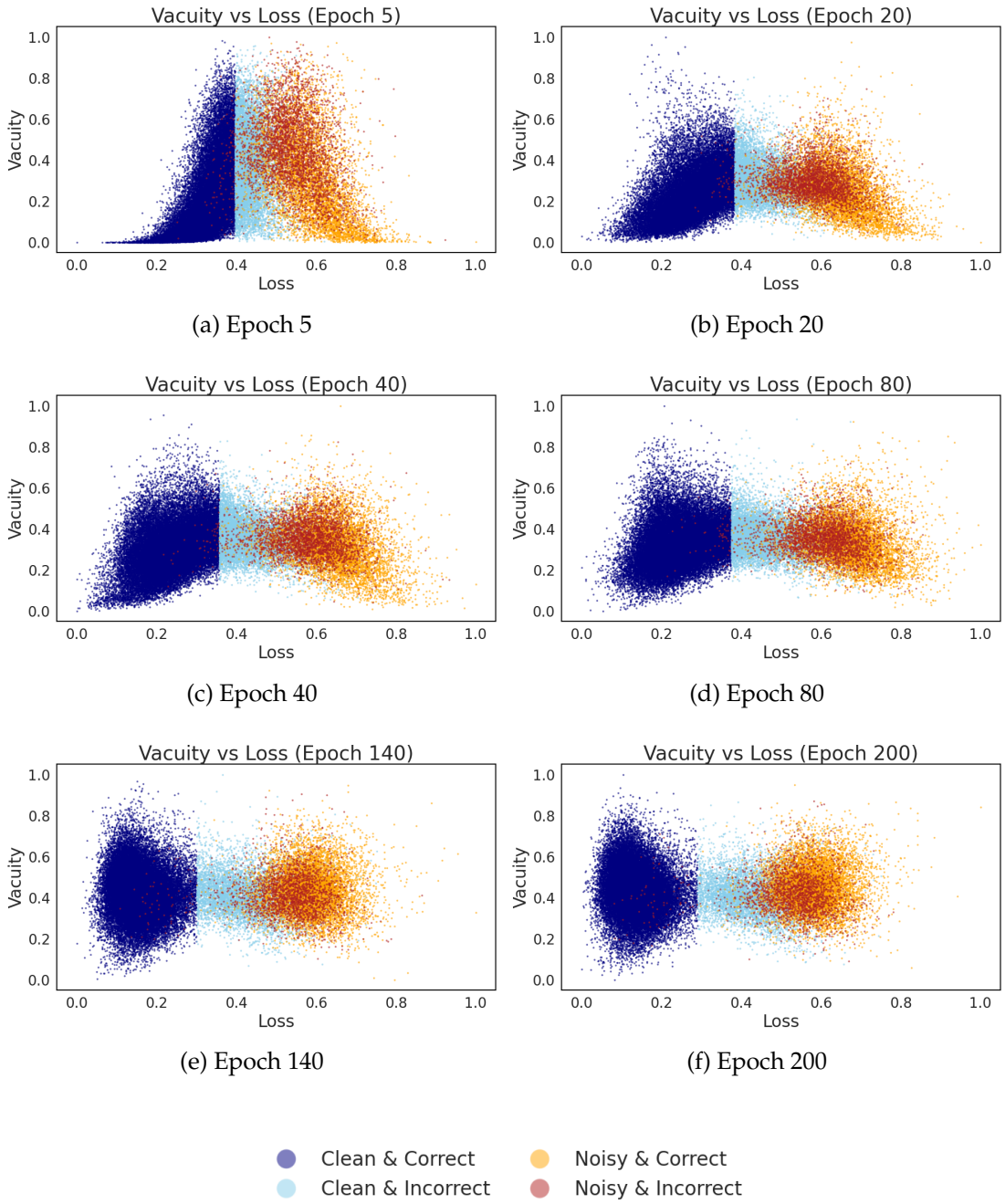
It is important to emphasize that neither vacuity nor dissonance conveys meaningful information in isolation. Their values must be interpreted in relation to the broader sample distribution, as absolute values may vary across tasks, training stages, or model confidence levels. To enable consistent comparisons, we apply min–max normalization to each uncertainty measure at every epoch, ensuring that the relative structure and spread of the data are preserved and visually interpretable.

To uncover how uncertainty evolves under noisy supervision, we track both vacuity and dissonance throughout training under different noise settings. This is done via post-hoc scatter plots in the uncertainty–loss space, where we distinguish between correctly and incorrectly predicted samples. Using the ground-truth labels (unavailable during training), we identify four groups: correctly predicted clean samples (samples for which the model prediction is correct and the noisy label matches the true label), incorrectly predicted clean samples (samples with correct labels that the model misclassifies), correctly predicted noisy samples (samples with incorrect labels where the model still predicts the true class), and incorrectly

predicted noisy samples (samples with incorrect labels that the model also misclassifies). We emphasize that prediction correctness in this context is assessed using the ground-truth labels, not the noisy labels used during training. This allows us to analyze how uncertainty correlates with model predictions and supervision quality over time.

5.5.2.1 Vacuity

We analyze the evolution of vacuity in relation to the loss throughout training under both 20% and 80% symmetric noise settings. In Figures 5.5 and 5.6 the results for 20% and 80% symmetric noise are shown, respectively.



(a) Epoch 5

(b) Epoch 20

(c) Epoch 40

(d) Epoch 80

(e) Epoch 140

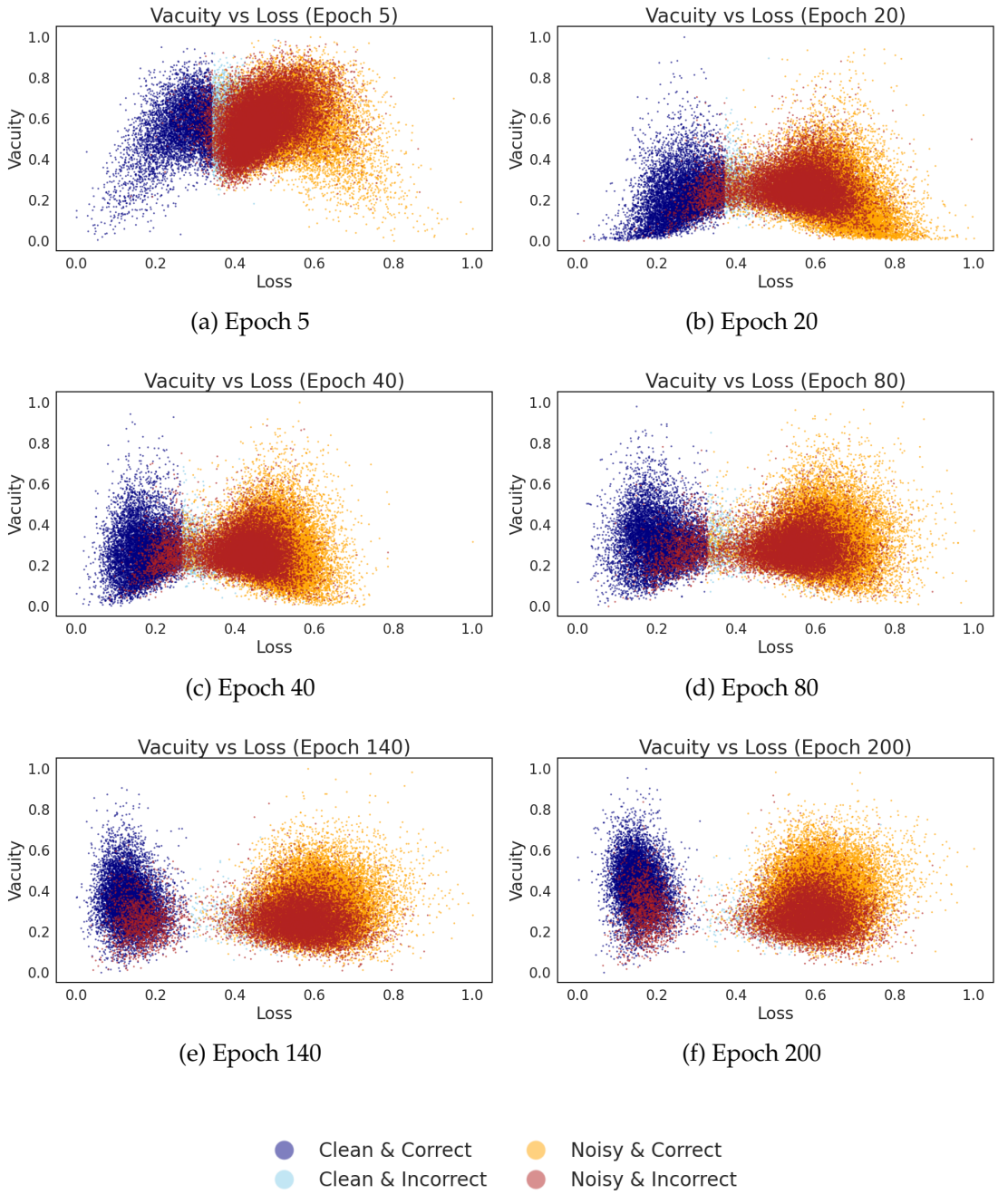
(f) Epoch 200

(g) Legend

Figure 5.5: Vacuity at various training epochs (20% symmetric noise).

In all plots, two distinct clusters — or clouds — emerge in the loss–vacuity space: one corresponding to samples selected as clean (i.e., included in supervised updates), and the other comprising the remaining unselected (potentially noisy) samples. The two clusters get separated as training progresses as the loss of labeled

and unlabeled data separates as explained in Section 5.5.1. However, there is no sharp separation between these two groups in terms of vacuity values. Both clouds span similar vacuity ranges, although the clean sample cloud typically exhibits a slightly wider spread, indicating more variability in uncertainty across this group.



(a) Epoch 5

(b) Epoch 20

(c) Epoch 40

(d) Epoch 80

(e) Epoch 140

(f) Epoch 200

(g) Legend

Figure 5.6: Vacuity at various training epochs (80% symmetric noise).

Similarly to 20% symmetric noise, in the 80% symmetric noise we clearly distinguish two clouds in the loss-vacuity space. A more interesting pattern emerges when focusing on the behavior of vacuity within these clouds. As training progresses, we observe that correctly predicted samples within the noisy cloud (i.e., samples that the model predicts correctly despite their noisy label) tend to show increasing vacuity values. This behavior suggests that the model, rather than collapsing toward incorrect supervision, is gradually discarding misleading evidence associated with the noisy label, while still accumulating evidence in favor of the predicted class. This is also observed in the 20% noise setting, altohugh with less clearance.

Conversely, in the clean cloud of samples under high noise, we observe that the truly noisy (i.e., mislabeled as clean) samples tend to exhibit lower vacuity than their correctly labeled counterparts. A likely explanation is that these samples accumulate more (albeit misleading) evidence from multiple classes (such as the ground-truth class), leading to lower vacuity despite being fundamentally incorrect.

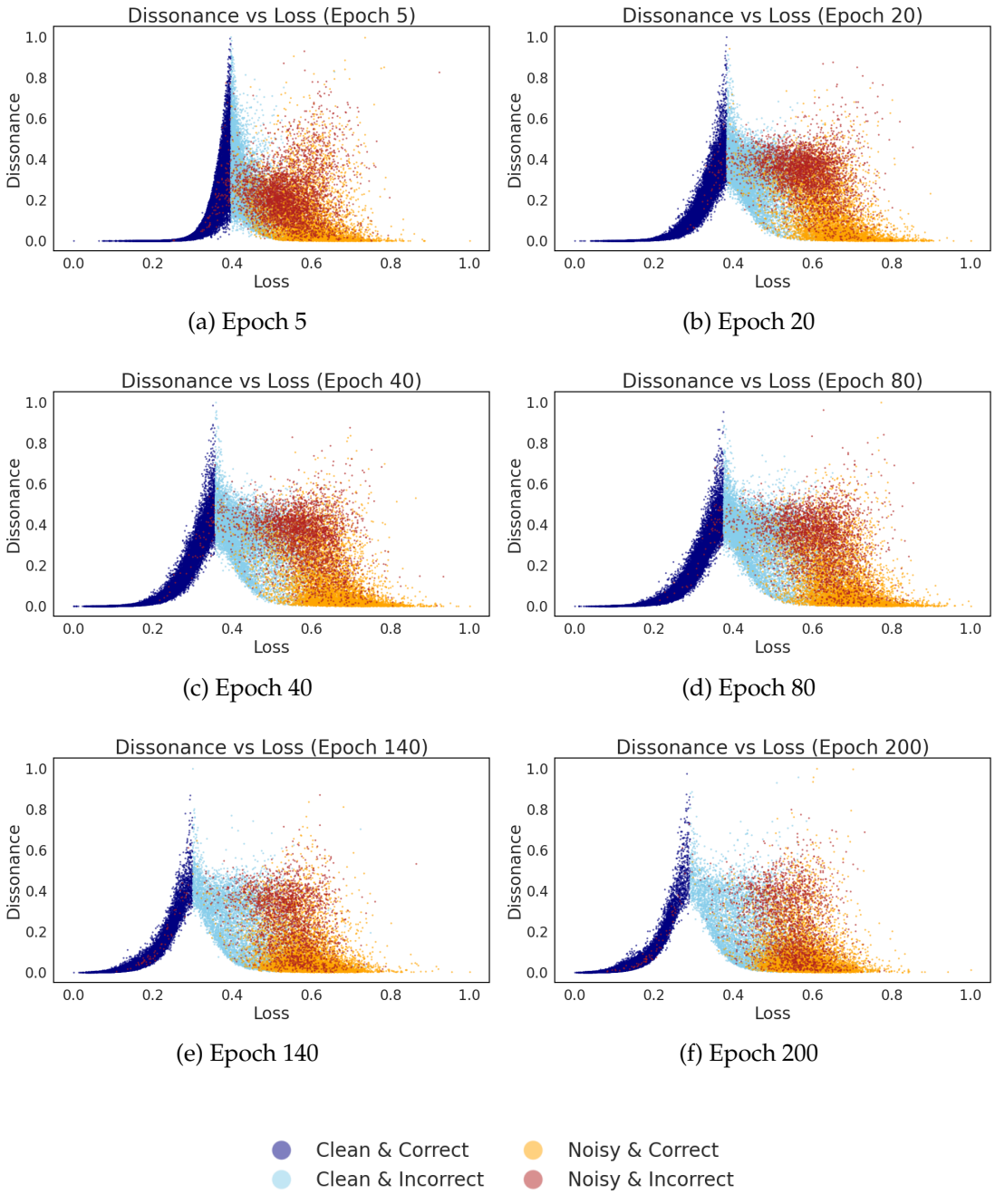
5.5.2.2 Dissonance

Figures 5.7 and 5.8 illustrate the evolution of dissonance across different training epochs under 20% and 80% symmetric noise conditions, respectively.

We now analyze these trends in detail. Dissonance, which captures the degree of conflict among competing class predictions, provides a complementary perspective to vacuity by reflecting how evenly distributed the model's belief is over different classes.

We analyse how dissonance propagates through the clean and noisy set of samples. In the clean set, a clear trend emerges: dissonance becomes increasingly tied to the sample loss. Samples with low loss consistently exhibit low dissonance, indicating confident, unambiguous predictions. As the loss increases, so does the dissonance, reflecting a rising level of uncertainty and internal disagreement in the model's predictions. This behavior is expected, as higher losses generally correspond to more difficult or misclassified examples where multiple classes may be competing for dominance in the output distribution. In the intermediate loss range, we observe a peak in dissonance, indicating conflict among several class (one of which is the label class). At the high-loss end, dissonance can go in either direction: it may remain high (if competing classes excluding the label class confuse the model) or drop again (if the model becomes confident in a class different

from the noisy label class, which can or cannot be the ground-true class). Similarly to the vacuity behavior (Section 5.5.2.1), in the first stages there are not low-loss and low-dissonance noisy samples, indicating that the model has not memorized them yet.



(a) Epoch 5

(b) Epoch 20

(c) Epoch 40

(d) Epoch 80

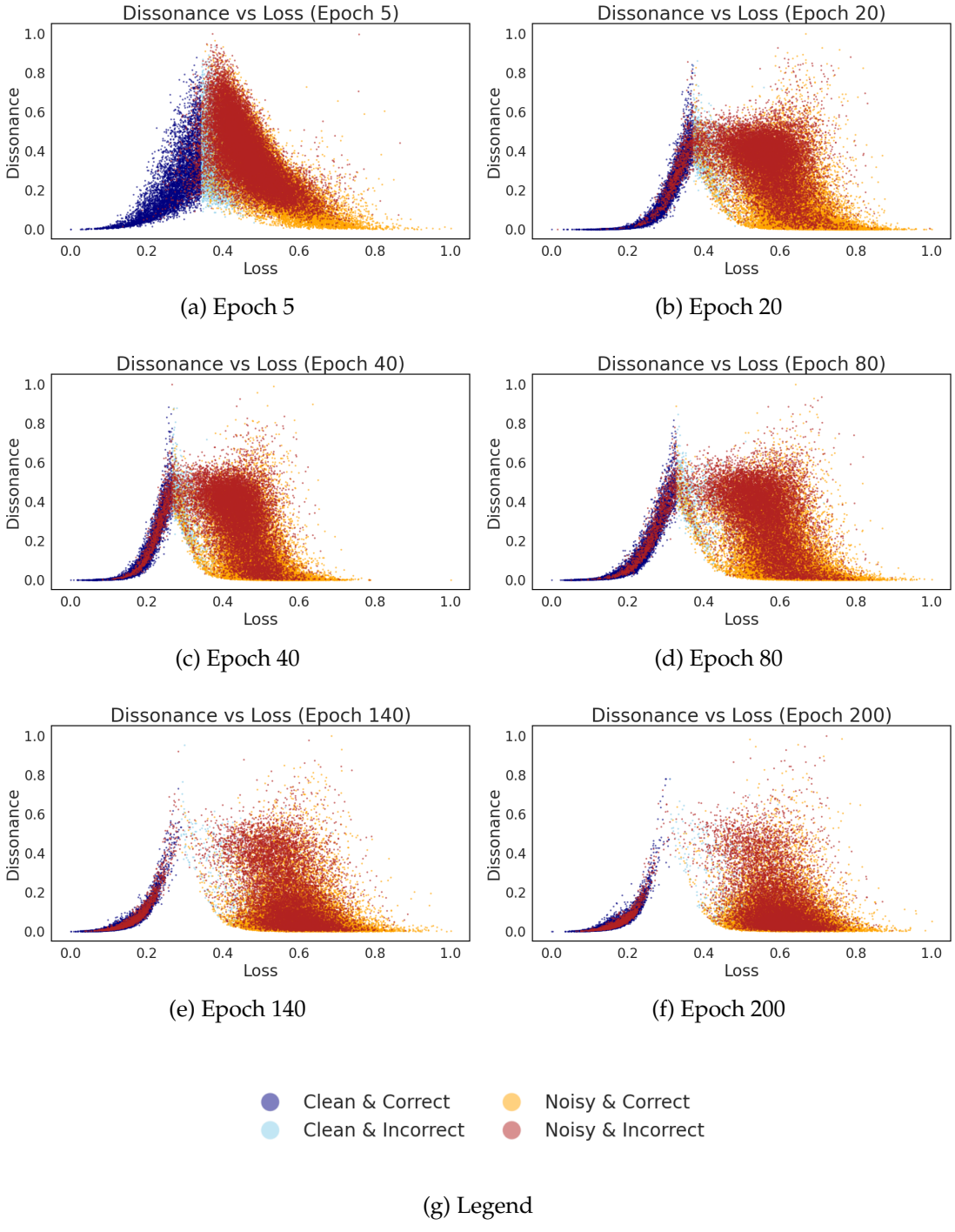
(e) Epoch 140

(f) Epoch 200

(g) Legend

Figure 5.7: Dissonance at various training epochs (20% symmetric noise).

An interesting behavior in the noisy sample cluster is that regions of low dissonance at early training stages tend to be populated by noisy samples that the network is predicting correctly — i.e., the prediction matches the ground-true label, not the noisy one.



(a) Epoch 5

(b) Epoch 20

(c) Epoch 40

(d) Epoch 80

(e) Epoch 140

(f) Epoch 200

(g) Legend

Figure 5.8: Dissonance at various training epochs (80% symmetric noise).

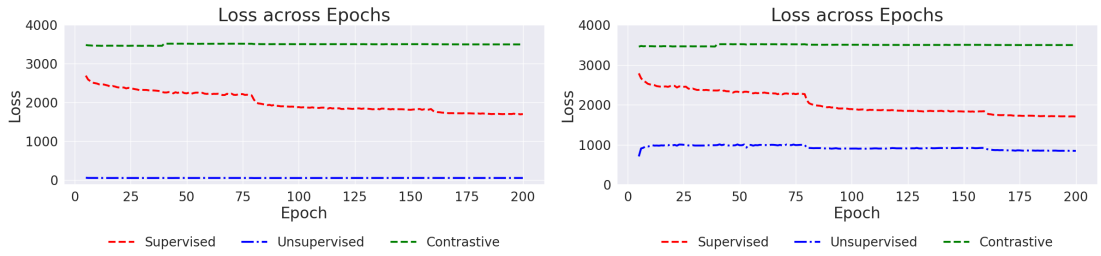
The evolution of dissonance in the high-noise configuration is similar to the low-noise scenario. However, we notice that the distinction between clean and noisy samples in the low-loss cloud is lost as the model is consistently trained with mislabeled data.

One key difference with vacuity is that dissonance does not provide a reliable signal to distinguish correctly labeled from mislabeled samples within the clean set in the high-noise setting. The clean and mislabeled samples in this group exhibit overlapping dissonance patterns form very early stages, limiting the metric's utility in this context. On the other hand, as in the 20% noise setting, in the high-loss cluster low-dissonance samples tend to be correctly predicted by the network, although this pattern is less pronounced than in the low-noise setting.

Overall, dissonance serves as a complementary indicator to vacuity — they show very different behaviors — and offers another insight to the evolution of training, memorization of noisy samples and uncertainty.

5.5.3 Loss curves

In Figure 5.9 we show the evolution of the different loss components (supervised, unsupervised and Contrastive).



(a) 20% Sym

(b) 80% Sym

Figure 5.9: Curves of different loss components under 20% and 80% symmetric noise.

The supervised loss decreases steadily over the epochs, with two noticeable drops that coincide with learning rate reductions (at epochs 80 and 160), reflecting the network's progressive learning of clean samples. In contrast, the unsupervised and contrastive loss components remain relatively stable throughout training. In the low-noise setting, the unsupervised component is weighted lightly, thus contributing minimally to training. On the other hand, under 80% symmetric noise,

the unsupervised loss has a larger impact and shows a slight decrease over time. The contrastive loss remains mostly stable and even increases around epoch 40. This rise is likely due to the adjustment made at that point to the number of overlapping classes required for using samples as negative examples (Section 5.2). Additionally, the lack of a consistent decline in contrastive loss may be explained by the use of pretrained weights obtained through contrastive pretraining. Nevertheless, as demonstrated in Section 5.5.4, the inclusion of contrastive learning is beneficial to overall performance.

5.5.4 Ablation Studies

Table 5.12 presents the impact of key components of our method under 80% symmetric noise. Removing the contrastive loss or Vacuity MixMatch slightly degrades performance, confirming their positive contribution. The variant without pretrained weights show a more meaningful drop, highlighting the importance of contrastive pretraining for robustness under high noise as a strong initialization for our model.

Table 5.12: Ablation Study: Best Test Accuracy (%) on CIFAR-100 with 80% Symmetric Noise

Methods	**80% Sym Noise**
w/o pretrained	66.30
w/o Contrastive loss	72.57
w/o Vacuity MixMatch	72.79
ours	72.92

5.6 Overall results

Our proposed method demonstrates strong and consistent performance across a variety of label noise scenarios, achieving SOTA or highly competitive results in both synthetic and real-world settings. Its performance is particularly notable under high noise rates, where we managed to narrow the performance gap between low and high noise settings.

Compared to models that do not leverage uncertainty — such as Li et al. (2020); Zhang et al. (2024); Li et al. (2023) — our method shows a noticeable improvement in medium - and especially high-noise settings. While uncertainty-aware models

(Zong et al., 2024; Huang et al., 2022) tend to improve performance in low-noise regimes, they still struggle when the noise level increases. Conversely, models incorporating contrastive learning (Zhang et al., 2024; Nagarajan et al., 2024) generally perform better under high noise. Our approach, which combines uncertainty estimation with self-supervised contrastive learning, outperforms both categories, especially in high-noise scenarios.

This trend holds across both symmetric and instance-dependent noise: our method excels under high and medium noise rates, though the performance gap narrows at medium noise levels. In low-noise settings, it remains competitive, although some other models may outperform it.

On the CIFAR-100N benchmark, our method achieves performance comparable to SOTA approaches that leverage deeper neural networks (Zong et al., 2024), and improves upon models that use the same backbone architecture (Tatjer et al., 2024). This highlights the effectiveness of our approach in real-world noisy label scenarios.

5.7 Limitations

In this section, we outline some limitations of our proposed model. While it demonstrates strong and consistent performance across a variety of scenarios, the following aspects remain constraints:

- As observed in Section 5.3, some SOTA methods outperform our model in low-noise settings. The incorporation of EDL and self-supervised contrastive learning makes our method more conservative in selecting clean samples. While this conservatism is advantageous in medium and high noise conditions, its advantages are not observed when there are less noisy samples.

- Although our EDL-based approach improves clean sample selection compared to standard cross-entropy (CE) loss models, it can still exhibit overfitting to noisy labels in high-noise scenarios, as shown in Section 5.5.

- While EDL integrates uncertainty-awareness throughout training, the practical utilization of these uncertainty measures remains somewhat constrained. Our Vacuity-based MixMatch method (Section 4.5) offers one way to incorporate uncertainty. However, as discussed in Section 5.5, uncertainty measures by themselves provide limited insights and present challenges when attempting to directly incorporate them into the training process.

- Although our model is trained for 200 epochs — compared to 400 epochs used by methods like BPT-PLR (Zhang et al., 2024) — it requires a pretraining stage with self-supervised contrastive learning. This pretraining can extend the total training time if a pretrained backbone is not already available, making the overall pipeline longer in practice.

5.8 Sustainability

This section outlines the sustainability considerations of the thesis across three dimensions: environmental, economic, and social. We estimate the environmental impact based on the carbon footprint. The economic aspect is limited to hardware usage, as no direct financial costs were involved. Finally, we reflect on the social implications, highlighting both the personal learning outcomes and the potential benefits of uncertainty-aware learning for broader AI applications.

5.8.1 Environmental impact

The experiments conducted throughout this thesis were executed primarily on a single NVIDIA GeForce RTX 2080 Ti GPU. This graphics card is part of NVIDIA's Turing architecture and features a typical power draw (Thermal Design Power, TDP) of approximately 250 watts under full load[2].

The experiments were run consistently over a period of approximately five months (February - June), and we estimate that the GPU was active two-thirds of that time (2400 hours). Therefore, the total energy consumption estimated is $250 \times 2400 = 600$ kWh.

To estimate the environmental impact, we convert the electricity usage into equivalent CO_2 emissions based on the carbon intensity of electricity generation in Spain. According to recent data, Spain's average carbon intensity is approximately **136 g CO_2/kWh**[3].

$$\text{CO}_2 \text{ emissions} = 600 \text{ kWh} \times 0.136 \text{ kg CO}_2/\text{kWh} = \mathbf{81.6 \text{ kg CO}_2}.$$

This value provides a rough estimate of the carbon footprint associated with the computational workload of the thesis.

[2]https://www.techpowerup.com/gpu-specs/geforce-rtx-2080-ti.c3305
[3]https://www.ree.es/es/datos/generacion/no-renovables-detalle-emisiones-CO2

5.8.2 Economic Impact

The economic impact during the development phase of this thesis has primarily been related to computational resources. Specifically, training and evaluating the proposed model involved the use of a single GPU. The main cost can be attributed to the electricity consumption associated with running multiple experiments across different configurations and datasets. No additional financial expenditure was incurred, as the work was carried out in an academic setting without remuneration. Therefore, the economic footprint corresponds solely to the amortization and operational costs of the hardware infrastructure used throughout the research.

The proposed research may offer economic benefits by improving the robustness and generalization of models trained with noisy data. By combining LNL techniques with principled Uncertainty Quantification, our method can reduce the reliance on costly manual relabeling or extensive data-cleaning pipelines. This leads to lower annotation costs and fewer errors during deployment, potentially reducing operational losses and post-deployment failures in real-world applications. These advantages make uncertainty-aware LNL methods particularly attractive for cost-effective AI system development. Especially, the EDL methodology allows for this uncertainty awareness without additional training expenses.

5.8.3 Social Impact

On a personal level, the thesis had a meaningful educational and professional impact. It enabled a deeper understanding of the technical aspects of the AI methods and algorithms involved in the thesis and offered valuable experience in managing and completing a long-term research project and navigating its associated challenges.

From a broader perspective, we believe that the project has the potential to benefit various social actors involved in data labeling, model deployment, and end-user applications. By addressing noisy labels through principled uncertainty estimation, the proposed approach may reduce the human effort required for data cleaning and annotation, and mostly mitigate the propagation of annotation errors. This, in turn, encourages the use of large datasets even when label noise is present. We also observed that incorporating uncertainty-aware predictions is beneficial within the LNL field and advocate for the broader adoption of uncertainty modeling, both in this area and across other domains of machine learning.

5.9 Ethical Considerations and Bias

The use of machine learning systems trained on noisy data introduces significant ethical challenges, particularly regarding bias, fairness, and accountability. In real-world scenarios, label noise is often instance-dependent and reflects the subjective judgments or societal biases of human annotators (Frenay and Verleysen, 2014; Liang et al., 2022). Consequently, models trained without appropriate noise handling may internalize and amplify these biases, potentially leading to unfair or discriminatory outcomes.

LNL methods — including our proposed approach — are inherently designed to combat the effects of mislabeled data and, by extension, reduce the propagation of annotation-induced bias. However, they are not immune to bias themselves. For example, the sample selection process in LNL methods inherently involves discrimination between "clean" and "noisy" samples, which can introduce additional bias if not handled carefully.

To address this, our method incorporates self-supervised contrastive learning, allowing the network to also learn from samples that are not explicitly labeled as clean. This helps mitigate selection bias and reduces reliance on potentially flawed labels during early training. Furthermore, self-supervised pretraining stabilizes learning and lowers the likelihood of memorizing noisy or biased samples.

Our model also integrates principled uncertainty estimation through EDL, which promotes cautious predictions in uncertain regions of the data. Beyond robustness, uncertainty awareness offers a degree of explainability by identifying which samples the model finds harder to learn or less confident about, or which samples the model has ambiguous evidence about. It provides insight into underlying dataset imbalances or sources of bias. In this way, uncertainty modeling can serve as a diagnostic tool, helping developers identify problematic regions in the data and refine curation or annotation practices accordingly.

Nonetheless, while these strategies contribute toward fairer and more trustworthy AI systems, they are not complete solutions. Ethical deployment still requires careful dataset auditing, impact analysis, and domain-specific considerations that go beyond technical safeguards.

Chapter 6

Conclusions

In this chapter, we perform a brief hindsight of the thesis to explain the overall and specific conclusions together with the degree of compliance with the proposed objectives. In addition, we provide some insights about the future lines.

6.1 Conclusions

The initial objectives of this thesis were outlined in Section 1.2. In this section, we reflect on the extent to which those objectives were achieved, the challenges encountered, and the overall outcomes of the work conducted.

The central goal of this thesis was to design a DivideMix-inspired learning framework that integrates Evidential Deep Learning to enhance uncertainty awareness and improve robustness to noisy labels. Specifically, we aimed to address the memorization of noisy labels during training and the overconfidence often exhibited by deep neural networks. Not only were we successful in achieving this goal, but we extended it by incorporating self-supervised learning components that further enhanced the model's performance (Chapter 4). The final proposed model demonstrates strong results in both high-noise and low-noise regimes (Section 5.3), outperforming competing methods by a large margin in the higher noise configurations.

In pursuing this goal, we also provided a thorough conceptual and empirical overview of LNL (Section 2.1), drawing from SOTA methods to inform and guide our design choices. In parallel, we investigated the field of uncertainty quantification in deep learning (Section 2.2), with a specific emphasis on EDL (Section 3.3). Throughout the thesis, particular importance was given to understanding the uncertainty measures provided by EDL — namely, vacuity and dissonance — and to demonstrating their relevance in the training dynamics.

From a practical standpoint, we successfully integrated EDL into the DivideMix framework at multiple stages of the training pipeline. This enabled us to leverage its uncertainty estimates both for improving sample selection and for enhancing model interpretability. In addition, we were able to successfully leverage vacuity as an enhancing component inside the training pipeline. Furthermore, we incorporated self-supervised contrastive learning (Section 4.2 and 4.6) at two key stages: pretraining and semi-supervised. This contributed to improved feature learning and greater robustness to noisy labels.

Moreover, we strengthened our sample selection strategy by combining information from both prediction distributions (via confidence margins) and feature representations (via prototype loss) (Section 4.4).

The proposed method was validated on the CIFAR-100 benchmark across various noise types — symmetric, instance-dependent, and real-world — and multiple noise rates. The model showed consistently strong and stable performance, excelling in high-noise scenarios, including both symmetric and instance-dependent noise configurations.

Importantly, we also provided an in-depth analysis of training dynamics as one of our main objectives, particularly the memorization of noisy labels, using both uncertainty and distribution-based evaluations. This allowed for a richer and more meaningful interpretation of our results and offered deeper insights into the inner dynamics and effects of mislabeled samples in training (Section 5.5).

While this work has made meaningful contributions to the intersection of uncertainty estimation and sample selection methods in LNL, there is still room for improvement. In particular, enhancing the recall of clean samples in low-noise scenarios and further reducing the impact of noisy labels in high-noise settings. Nonetheless, the objectives outlined at the beginning of this thesis have been thoroughly addressed, and the methodologies chosen have prove to successfully tackle the LNL problem.

6.2 Future Work

An appealing future direction for this thesis is to explore a more explicit use of the uncertainty measures produced by our model. The analysis presented in Section 5.5 suggests that these measures could be further leveraged to mitigate the memorization of noisy labels. In particular, it may be possible to relabel noisy

samples that exhibit low dissonance at certain stages of training—especially when the model's predictions align with the ground-truth labels for those samples.

However, this strategy should be approached with caution, as prior work (Köhler et al., 2019) has shown that such samples can sometimes be redundant when added to the clean set. Nonetheless, gaining a deeper understanding of which noisy samples are most valuable when relabeled could provide meaningful insights. This may be beneficial, for example, in reducing the need for manual labeling if informative samples can be reliably identified and incorporated into the clean set, while less informative or misleading ones can be safely discarded.

Bibliography

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., Makarenkov, V. and Nahavandi, S. (2021), 'A review of uncertainty quantification in deep learning: Techniques, applications and challenges', *Information Fusion* **76**, 243–297.
URL: *http://dx.doi.org/10.1016/j.inffus.2021.05.008*

Allen-Zhu, Z., Li, Y. and Liang, Y. (2019), 'Learning and generalization in overparameterized neural networks, going beyond two layers', *Advances in neural information processing systems* **32**.

Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y. and Lacoste-Julien, S. (2017), 'A closer look at memorization in deep networks'.
URL: *https://arxiv.org/abs/1706.05394*

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A. and Raffel, C. (2019), 'Mixmatch: A holistic approach to semi-supervised learning'.
URL: *https://arxiv.org/abs/1905.02249*

Buciluǎ, C., Caruana, R. and Niculescu-Mizil, A. (2006), Model compression, *in* 'Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining', pp. 535–541.

Cordeiro, F. R., Sachdeva, R., Belagiannis, V., Reid, I. and Carneiro, G. (2023), 'Longremix: Robust learning with high confidence samples in a noisy label environment', *Pattern recognition* **133**, 109013.

da Costa, V. G. T., Fini, E., Nabi, M., Sebe, N. and Ricci, E. (2022), 'solo-learn: A library of self-supervised methods for visual representation learning', *Journal of Machine Learning Research* **23**(56), 1–6.
URL: *http://jmlr.org/papers/v23/21-1155.html*

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. and Houlsby, N. (2021), 'An image is worth 16x16 words: Transformers for image recognition at scale'.
URL: *https://arxiv.org/abs/2010.11929*

Duane, S., Kennedy, A. D., Pendleton, B. J. and Roweth, D. (1987), 'Hybrid monte carlo', *Physics letters B* **195**(2), 216–222.

Frenay, B. and Verleysen, M. (2014), 'Classification in the presence of label noise: A survey', *IEEE Transactions on Neural Networks and Learning Systems* **25**(5), 845–869.

Gal, Y. and Ghahramani, Z. (2016*a*), 'Bayesian convolutional neural networks with bernoulli approximate variational inference'.
URL: *https://arxiv.org/abs/1506.02158*

Gal, Y. and Ghahramani, Z. (2016*b*), 'Dropout as a bayesian approximation: Representing model uncertainty in deep learning'.
URL: *https://arxiv.org/abs/1506.02142*

Gao, J., Chen, M., Xiang, L. and Xu, C. (2024), 'A comprehensive survey on evidential deep learning and its applications'.
URL: *https://arxiv.org/abs/2409.04720*

Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., Shahzad, M., Yang, W., Bamler, R. and Zhu, X. X. (2022), 'A survey of uncertainty in deep neural networks'.
URL: *https://arxiv.org/abs/2107.03342*

Goldberger, J. and Ben-Reuven, E. (2017), Training deep neural-networks using a noise adaptation layer, *in* 'International Conference on Learning Representations'.
URL: *https://openreview.net/forum?id=H12GRgcxg*

Guo, H., Liu, H., Li, R., Wu, C., Guo, Y. and Xu, M. (2018), 'Margin & diversity based ordering ensemble pruning', *Neurocomputing* **275**, 237–246.

Guo, Z., Wan, Z., Zhang, Q., Zhao, X., Zhang, Q., Kaplan, L. M., Jøsang, A., Jeong, D. H., Chen, F. and Cho, J.-H. (2024), 'A survey on uncertainty reasoning and quantification in belief theory and its application to deep learning', *Information Fusion* **101**, 101987.
URL: *https://www.sciencedirect.com/science/article/pii/S1566253523003032*

Han, B., Yao, J., Niu, G., Zhou, M., Tsang, I., Zhang, Y. and Sugiyama, M. (2018), 'Masking: A new perspective of noisy supervision'.
URL: *https://arxiv.org/abs/1805.08193*

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I. and Sugiyama, M. (2018), 'Co-teaching: Robust training of deep neural networks with extremely noisy labels'.
URL: *https://arxiv.org/abs/1804.06872*

He, K., Chen, X., Xie, S., Li, Y., Dollár, P. and Girshick, R. (2022), Masked autoencoders are scalable vision learners, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 16000–16009.

He, K., Zhang, X., Ren, S. and Sun, J. (2015), 'Deep residual learning for image recognition'.
URL: *https://arxiv.org/abs/1512.03385*

Hendrycks, D., Lee, K. and Mazeika, M. (2019), 'Using pre-training can improve model robustness and uncertainty'.
URL: *https://arxiv.org/abs/1901.09960*

Huang, B., Lin, Y. and Xu, C. (2022), 'Contrastive label correction for noisy label learning', *Information Sciences* **611**, 173–184.
URL: *https://www.sciencedirect.com/science/article/pii/S002002552200946X*

Huang et al., Y. (2022), Uncertainty-aware learning against label noise on imbalanced datasets, *in* 'Proceedings of the AAAI conference on artificial intelligence', Vol. 36, pp. 6960–6969.

Jenni, S. and Favaro, P. (2018), 'Deep bilevel learning'.
URL: *https://arxiv.org/abs/1809.01465*

Jiang, L., Zhou, Z., Leung, T., Li, L.-J. and Fei-Fei, L. (2018), 'Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels'.
URL: *https://arxiv.org/abs/1712.05055*

Joo, T., Chung, U. and Seo, M.-G. (2020), Being Bayesian about categorical probability, *in* H. D. III and A. Singh, eds, 'Proceedings of the 37th International Conference on Machine Learning', Vol. 119 of *Proceedings of Machine Learning Research*, PMLR, pp. 4950–4961.
URL: *https://proceedings.mlr.press/v119/joo20a.html*

Jøsang, A. (2016), *Subjective logic*, Vol. 3, Springer.

Karim, N., Rizve, M. N., Rahnavard, N., Mian, A. and Shah, M. (2022), Unicon: Combating label noise through uniform selection and contrastive learning, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 9676–9686.

Kim, D., Ryoo, K., Cho, H. and Kim, S. (2025), 'Splitnet: learnable clean-noisy label splitting for learning with noisy labels', *International Journal of Computer Vision* **133**(2), 549–566.

Kim, N.-r., Lee, J.-S. and Lee, J.-H. (2024), Learning with structural labels for learning with noisy labels, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 27610–27620.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P. and Girshick, R. (2023), 'Segment anything'.
URL: *https://arxiv.org/abs/2304.02643*

Köhler, J. M., Autenrieth, M. and Beluch, W. H. (2019), Uncertainty based detection and relabeling of noisy image labels., *in* 'CVPR workshops', pp. 33–37.

Krizhevsky, A. (2009), 'CIFAR-10 and CIFAR-100 datasets', https://www.cs.toronto.edu/~kriz/cifar.html. Accessed: 2023-01-13.

Krizhevsky, A., Sutskever, I. and Hinton, G. (2012), 'Imagenet classification with deep convolutional neural networks', *Neural Information Processing Systems* **25**.

Lakshminarayanan, B., Pritzel, A. and Blundell, C. (2017), 'Simple and scalable predictive uncertainty estimation using deep ensembles', *Advances in neural information processing systems* **30**.

Li, C., Li, K., Ou, Y., Kaplan, L. M., Jøsang, A., Cho, J.-H., Jeong, D. H. and Chen, F. (2024), 'Hyper evidential deep learning to quantify composite classification uncertainty', *ArXiv* **abs/2404.10980**.
URL: *https://api.semanticscholar.org/CorpusID:269187924*

Li, J., Socher, R. and Hoi, S. C. H. (2020), 'Dividemix: Learning with noisy labels as semi-supervised learning'.
URL: *https://arxiv.org/abs/2002.07394*

Li, S., Xia, X., Ge, S. and Liu, T. (2022), Selective-supervised contrastive learning with noisy labels, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 316–325.

Li, Y., Han, H., Shan, S. and Chen, X. (2023), Disc: Learning from noisy labels via dynamic instance-specific selection and correction, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 24070–24079.

Liang, X., Liu, X. and Yao, L. (2022), 'Review–a survey of learning from noisy labels', *ECS Sensors Plus* **1**(2), 021401.
URL: *https://dx.doi.org/10.1149/2754-2726/ac75f5*

Liu, T. and Tao, D. (2014), 'Classification with noisy labels by importance reweighting', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **38**.

Liu, X., Zhou, B., Yue, Z. and Cheng, C. (2024), 'Plremix: Combating noisy labels with pseudo-label relaxed contrastive representation learning'.
URL: *https://arxiv.org/abs/2402.17589*

Liu, Z., Zhang, X., He, J., Fu, D., Samaras, D., Tan, R., Wang, X. and Wang, S. (2023), 'Chimera: Learning with noisy labels by contrasting mixed-up augmentations'.
URL: *https://arxiv.org/abs/2310.05183*

Lukasik, M., Bhojanapalli, S., Menon, A. K. and Kumar, S. (2020), 'Does label smoothing mitigate label noise?'.
URL: *https://arxiv.org/abs/2003.02819*

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

Ma, X., Huang, H., Wang, Y., Romano, S., Erfani, S. and Bailey, J. (2020), 'Normalized loss functions for deep learning with noisy labels'.
URL: *https://arxiv.org/abs/2006.13554*

Malach, E. and Shalev-Shwartz, S. (2018), 'Decoupling "when to update" from "how to update"'.
URL: *https://arxiv.org/abs/1706.02613*

Malinin, A. and Gales, M. (2018), 'Predictive uncertainty estimation via prior networks'.
URL: *https://arxiv.org/abs/1802.10501*

Miao, Q., Wu, X., Xu, C., Zuo, W. and Meng, Z. (2023), 'On better detecting and leveraging noisy samples for learning with severe label noise', *Pattern Recognition* **136**, 109210.

Nagarajan, B., Marques, R., Aguilar, E. and Radeva, P. (2024), 'Bayesian dividemix++ for enhanced learning with noisy labels', *Neural Networks* **172**, 106122.
URL: *https://www.sciencedirect.com/science/article/pii/S0893608024000364*

Nguyen, D. T., Mummadi, C. K., Ngo, T. P. N., Nguyen, T. H. P., Beggel, L. and Brox, T. (2019), 'Self: Learning to filter noisy labels with self-ensembling'.
URL: *https://arxiv.org/abs/1910.01842*

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B. and Snoek, J. (2019), 'Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift', *Advances in neural information processing systems* **32**.

Patrini, G., Rozza, A., Menon, A., Nock, R. and Qu, L. (2017), 'Making deep neural networks robust to label noise: a loss correction approach'.
URL: *https://arxiv.org/abs/1609.03683*

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G. and Sutskever, I. (2021), 'Learning transferable visual models from natural language supervision'.
URL: *https://arxiv.org/abs/2103.00020*

Raghu, M., Blumer, K., Sayres, R., Obermeyer, Z., Kleinberg, R., Mullainathan, S. and Kleinberg, J. (2019), 'Direct uncertainty prediction for medical second opinions'.
URL: *https://arxiv.org/abs/1807.01771*

Ramalho, T. and Miranda, M. (2019), 'Density estimation in representation space to predict model uncertainty'.
URL: *https://arxiv.org/abs/1908.07235*

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016), You only look once: Unified, real-time object detection, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 779–788.

Renda, A., Barsacchi, M., Bechini, A. and Marcelloni, F. (2019), 'Comparing ensemble strategies for deep learning: An application to facial expression recognition', *Expert Systems with Applications* **136**, 1–11.

Sachdeva, R., Cordeiro, F. R., Belagiannis, V., Reid, I. and Carneiro, G. (2023), 'Scanmix: Learning from severe label noise via semantic clustering and semi-supervised learning', *Pattern recognition* **134**, 109121.

Sagi, O. and Rokach, L. (2018), 'Ensemble learning: A survey', *Wiley interdisciplinary reviews: data mining and knowledge discovery* **8**(4), e1249.

Sensoy, M., Kaplan, L. and Kandemir, M. (2018), 'Evidential deep learning to quantify classification uncertainty'.
URL: *https://arxiv.org/abs/1806.01768*

Sentz, K. and Ferson, S. (2002), 'Combination of evidence in dempster-shafer theory'.

Shanmugam, D., Blalock, D., Balakrishnan, G. and Guttag, J. (2021), Better aggregation in test-time augmentation, *in* 'Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)', pp. 1214–1223.

Shu, J., Xie, Q., Yi, L., Zhao, Q., Zhou, S., Xu, Z. and Meng, D. (2019), 'Meta-weight-net: Learning an explicit mapping for sample weighting'.
URL: *https://arxiv.org/abs/1902.07379*

Song, H., Kim, M. and Lee, J.-G. (2019), SELFIE: Refurbishing unclean samples for robust deep learning, *in* K. Chaudhuri and R. Salakhutdinov, eds, 'Proceedings of the 36th International Conference on Machine Learning', Vol. 97 of *Proceedings of Machine Learning Research*, PMLR, pp. 5907–5915.
URL: *https://proceedings.mlr.press/v97/song19b.html*

Song, H., Kim, M., Park, D. and Lee, J.-G. (2020), 'How does early stopping help generalization against label noise?'.
URL: *https://arxiv.org/abs/1911.08059*

Song, H., Kim, M., Park, D., Shin, Y. and Lee, J.-G. (2022), 'Learning from noisy labels with deep neural networks: A survey'.
URL: *https://arxiv.org/abs/2007.08199*

Tanno, R., Saeedi, A., Sankaranarayanan, S., Alexander, D. C. and Silberman, N. (2019*a*), 'Learning from noisy labels by regularized estimation of annotator confusion'.
URL: *https://arxiv.org/abs/1902.03680*

Tanno, R., Saeedi, A., Sankaranarayanan, S., Alexander, D. C. and Silberman, N. (2019*b*), 'Learning from noisy labels by regularized estimation of annotator confusion'.
URL: *https://arxiv.org/abs/1902.03680*

Tatjer, A., Nagarajan, B., Marques, R. and Radeva, P. (2024), 'Decoding class dynamics in learning with noisy labels', *Pattern Recognition Letters* **184**, 239–245.
URL: *https://www.sciencedirect.com/science/article/pii/S0167865524001132*

Ulmer, D., Hardmeier, C. and Frellsen, J. (2023), 'Prior and posterior networks: A survey on evidential deep learning methods for uncertainty estimation'.
URL: *https://arxiv.org/abs/2110.03051*

V7 Labs (2022), 'The beginner's guide to contrastive learning'. Accessed: 2025-06-16.
URL: *https://www.v7labs.com/blog/contrastive-learning-guide*

van Amersfoort, J., Smith, L., Teh, Y. W. and Gal, Y. (2020), 'Uncertainty estimation using a single deep deterministic neural network'.
URL: *https://arxiv.org/abs/2003.02037*

Wang, G., Li, W., Ourselin, S. and Vercauteren, T. (2019), Automatic brain tumor segmentation using convolutional neural networks with test-time augmentation, *in* 'Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 4th International Workshop, BrainLes 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Revised Selected Papers, Part II 4', Springer, pp. 61–72.

Xiao, T., Xia, T., Yang, Y., Huang, C. and Wang, X. (2015*a*), Learning from massive noisy labeled data for image classification, *in* '2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 2691–2699.

Xiao, T., Xia, T., Yang, Y., Huang, C. and Wang, X. (2015*b*), Learning from massive noisy labeled data for image classification, *in* '2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 2691–2699.

Xiao, T., Xia, T., Yang, Y., Huang, C. and Wang, X. (2015*c*), Learning from massive noisy labeled data for image classification, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)'.

Yao, J., Wang, J., Tsang, I., Zhang, Y., Sun, J., Zhang, C. and Zhang, R. (2017), 'Deep learning from noisy image labels with quality embedding'.
URL: *https://arxiv.org/abs/1711.00583*

Yao, Y., Sun, Z., Zhang, C., Shen, F., Wu, Q., Zhang, J. and Tang, Z. (2021), Jo-src: A contrastive approach for combating noisy labels, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 5192–5201.

Yi, L., Liu, S., She, Q., McLeod, A. I. and Wang, B. (2022), On learning contrastive representations for learning with noisy labels, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 16682–16691.

Zhang, C., Bengio, S., Hardt, M., Recht, B. and Vinyals, O. (2021), 'Understanding deep learning (still) requires rethinking generalization', *Communications of the ACM* **64**(3), 107–115.

Zhang, H., Cisse, M., Dauphin, Y. N. and Lopez-Paz, D. (2018), 'mixup: Beyond empirical risk minimization'.
URL: *https://arxiv.org/abs/1710.09412*

Zhang, Q., Jin, G., Zhu, Y., Wei, H. and Chen, Q. (2024), 'Bpt-plr: A balanced partitioning and training framework with pseudo-label relaxed contrastive loss for noisy label learning', *Entropy* **26**(7).
URL: *https://www.mdpi.com/1099-4300/26/7/589*

Zhang, Q., Zhu, Y., Cordeiro, F. R. and Chen, Q. (2025), 'Psscl: A progressive sample selection framework with contrastive loss designed for noisy labels', *Pattern Recognition* **161**, 111284.
URL: *https://www.sciencedirect.com/science/article/pii/S0031320324010355*

Zhang, Z. and Sabuncu, M. R. (2018), 'Generalized cross entropy loss for training deep neural networks with noisy labels'.
URL: *https://arxiv.org/abs/1805.07836*

Zhao, G., Li, G., Qin, Y., Liu, F. and Yu, Y. (2022), Centrality and consistency: two-stage clean samples identification for learning with instance-dependent noisy labels, *in* 'European Conference on Computer Vision', Springer, pp. 21–37.

Zheltonozhskii, E., Baskin, C., Mendelson, A., Bronstein, A. M. and Litany, O. (2022), Contrast to divide: Self-supervised pre-training for learning with noisy labels, *in* '2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)', IEEE, p. 387–397.
URL: *http://dx.doi.org/10.1109/WACV51458.2022.00046*

Zhou, T., Wang, S. and Bilmes, J. (2021), Robust curriculum learning: from clean label detection to noisy label self-correction, *in* 'International Conference on Learning Representations'.
URL: *https://openreview.net/forum?id=lmTWnm3coJJ*

Zong, C.-C., Wang, Y.-W., Xie, M.-K. and Huang, S.-J. (2024), 'Dirichlet-based prediction calibration for learning with noisy labels', *Proceedings of the AAAI Conference on Artificial Intelligence* **38**(15), 17254–17262.
URL: *http://dx.doi.org/10.1609/aaai.v38i15.29672*