Technische Universität Berlin

Faculty of Electrical Engineering and Computer Science
Dept. of Computer Engineering and Microelectronics
Remote Sensing Image Analysis Group

Collaborative Learning Models for Classification of Remote Sensing Images with Noisy Labels

---

Bachelor of Science in Computer Science

November, 2020

Ahmet Kerem Aksoy

Matriculation Number: 391954

Supervisor: Prof. Dr. Begüm Demir

Advisors: Dr. Mahdyar Ravanbakhsh, Tristan Kreuziger

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt habe. Sämtliche benutzten Informationsquellen sowie das Gedankengut Dritter wurden im Text als solche kenntlich gemacht und im Literaturverzeichnis angeführt. Die Arbeit wurde bisher nicht veröffentlicht und keiner Prüfungsbehörde vorgelegt.

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Berlin, 17.11.2020

. . . . . .

Ahmet Kerem Aksoy

Acknowledgements

Foremost, I would like to express my deepest gratitude to my supervisor Prof. Dr. Begüm Demir for her continuous support and invaluable advice. I would also like to thank my advisors Dr. Mahdyar Ravanbakhsh and Tristan Kreuziger for their unwavering guidance and insights throughout the creation of this work. Furthermore, I would like to thank Remote Sensing Image Analysis Group at the Technical University of Berlin for providing me a workplace in their office and access to their server as well as to TU Berlin High-Performance Cluster.

Abstract

Deep neural networks require large amounts of correctly annotated training data for successful classification of remote sensing images. Unfortunately, large remote sensing archives almost always include label noise, which distorts the course of the learning process by misleading the classifier. This consequently decreases the performance of the classifier, resulting in wrong predictions. To overcome this problem, the negative effects of label noise on training must be alleviated. However, it is challenging to detect and fix label noise, especially in multi-label setting, where each instance can be associated with more than one label. Even though the current literature suggests noise handling approaches for single label data, there is no equivalent work in the multi-label setting, analyzing the adverse effects of label noise. To this end, we propose a novel method called Consensual Collaborative Multi-Label Learning (CCML) to address the challenge of classification with noisy labels in the multi-label setting. The proposed method identifies noisy samples and excludes them from back-propagation, aiming to train the classifier with clean samples. CCML also detects noisy labels in samples and fixes them through a relabeling mechanism. The comparative experiments conducted on two multi-label remote sensing archives, BigEarthNet and multi-label UC Merced Land Use, indicate that the proposed method outperforms the models that are trained on data with noisy labels, when sample-wise noise is applied in a class-balanced setting. The benefits of the proposed method become more apparent with increased noise rates. The experimental results reveal remarkable characteristics of multi-label data in the presence of noise, and provide insights to the effects of class-imbalance in multi-label classification.

Zusammenfassung

Tiefe neuronale Netze erfordern große Mengen korrekt annotierter Trainingsdaten für eine erfolgreiche Klassifizierung von Fernerkundungsbildern. Leider enthalten große Fernerkundungsarchive fast immer Störung in den Annotationen, was den Verlauf des Lernprozesses verzerrt, indem es den Klassifikator irreführt. Dies verringert folglich die Leistung des Klassifikators, was zu falschen Vorhersagen führt. Um dieses Problem zu lösen, müssen die negativen Effekte der Störungen in den Annotationen auf das Training gemindert werden. Es ist jedoch schwierig, die Störungen in den Annotationen zu erkennen und zu beheben, insbesondere in Situationen, in denen jeder Instanz mehr als einem Annotation zugeordnet werden kann. Obwohl in der aktuellen Literatur Ansätze zum Umgang mit gestörten Annotationen für Daten mit singulären Annotationen vorgeschlagen werden, gibt es keine vergleichbare Arbeit in der Einstellung mit Daten mit mehreren Annotationen, in der die nachteiligen Auswirkungen von den Störungen in den Annotationen analysiert werden. Zu diesem Zweck schlagen wir eine neuartige Methode namens Consensual Collaborative Multi-Label Learning (CCML) vor, um die Herausforderung der Klassifizierung mit gestörten Annotationen in der Einstellung mit Daten mit mehreren Annotationen anzugehen. Das vorgeschlagene Verfahren identifiziert gestörte Proben und schließt sie von der Back-propagation aus, um den Klassifikator mit sauberen Proben zu trainieren. CCML erkennt auch gestörte Annotationen in Proben und korrigiert sie durch einen Mechanismus zur Neuannotation. Die Vergleichsexperimente, die mit zwei mehrfach gekennzeichneten Fernerkundungsarchiven, BigEarthNet und UC Merced Land Use, durchgeführt wurden, zeigen, dass die vorgeschlagene Methode die Modelle übertrifft, die auf Daten mit gestörten Annotationen trainiert werden, wenn in einer Klasse stichprobenartige Störung angewendet wird. Die Vorteile des vorgeschlagenen Verfahrens werden mit erhöhten Störungsraten deutlicher. Die Ergebnisse zeigen bemerkenswerte Eigenschaften von Daten mit mehreren Annotationen bei Vorhandensein von Störung und liefern Einblicke in die Auswirkungen des Klassenungleichgewichts bei der Multi-Label-Klassifizierung.

Contents

List of Acronyms

DNN	Deep Neural Network
CCE	Categorical Cross Entropy
BCE	Binary Cross Entropy
MMD	Maximum Mean Discrepancy
MAE	Mean Absolute Error
MSE	Mean Square Error
CCML	Consensual Collaborative Multi-Label Learning
RKHS	Reproducing Kernel Hilbert Space
CPU	Central Processing Unit
GPU	Graphics Processing Unit
R-CNN	Regional-based Convolutional Neural Networks
YOLO	You Only Look Once
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
GB	Gigabyte
RAM	Random-access Memory
SGD	Stochastic Gradient Descent
RBF	Radial Basis Function
RNS	Random Noise per Sample
MLN	Mix Label Noise
NaN	Not a Number
NCS	Noisy Class Selector
NCF	Noisy Class Flipper

List of Figures

List of Tables

1 Introduction

It is hard to deny the progress of deep learning in the last decade, considering its success especially in image and speech recognition[34][20]. Deep learning has successfully tackled many previously inextricable problems, and proved its effectiveness in many tasks, surpassing existing machine learning algorithms and human expert performance. Its large application area, ranging from medical image analysis[41] to visual art processing[16], attracted many researchers to work in the field looking for solutions for broad range of real-world problems. Perhaps one of the most sensational achievements in the field, AlphaGo[53], was the first program to defeat a world champion in the game of Go. AlphaGo showed the ability and great potential of deep learning in solving complex problems and transcending human capabilities. The mentioned advancements in the field carried deep learning to the focal of the research community as well as the industry for the last couple of years.

However, prior to the last decade, using deep learning to solve real-world problems was not preferred over the long-established machine learning algorithms such as Support Vector Machine[9] or simple classifiers that work on handcrafted features. Training deep neural networks was computationally very expensive with traditional CPUs. Even simple models that consist of a handful of layers took to much time to give promising results. The standard back-propagation algorithm that was applied to a deep neural network by LeCun had considerable success, although it required 3 days of training to recognize handwritten ZIP codes[38]. With the introduction of GPUs, which are very convenient for parallel matrix/vector calculations, the computational cost of training a Deep Neural Network (DNN) was dramatically reduced. Involvement of GPUs sped up the necessary time for training, which made deep learning feasible among other machine learning methods.

Along the decrease of computation time for training, another key factor that made deep learning practical was the exponential explosion of available data over time. The internet became increasingly accessible through the beginning of 2000s to millions of people around the world, who provided the needed data to train neural networks on a large scale. The performance of DNNs has proportionally increased with the provided examples to networks, which attracted the attention of big tech companies such as Google and Facebook, who are able to collect huge amounts of data through their various services. Researchers have developed better models and techniques that can take advantage of myriad of data, which made deep learning flourish especially in image recognition.

As a result of these advancements in computation power and data availability, today, deep learning is the undisputed solution to complex problems, automating processes, and detecting patterns. Despite the staggering advancements in the field, deep learning has still a lot of potential for solving real-world problems with better techniques and data.

Deep neural networks are proven to be successful in the field of computer vision[34]. They

outperform previous state-of-the-art machine learning methods, and more importantly, they do not need specialized hand-crafted pipelines. That, along with their performance, makes deep learning models a prominent approach to deal with diverse computer vision problems. However, their success relies on the quality and quantity of the data that the neural networks are trained with. Large, clean datasets that are widely used in research and in the industry, such as ImageNet[12], allow DNNs to learn diverse features, which results in more accurate predictions. In the presence of rich, high quality annotated datasets, image recognition is not an arduous task for computers anymore. Nonetheless, collecting such high quality datasets is a challenging and expensive process. Since datasets get manually labeled by multiple expert annotators to reduce label errors, creating a high quality dataset often requires a lot of time and skilled labor. They are also insufficient for unique tasks, such as analysis of remote sensing images, which require domain-specific archives. Going through the same demanding process to obtain well-annotated datasets for different domains is not always feasible. Particularly, for classification tasks that include large number of classes or fine-grained classes, finding a clean dataset may be impossible.

On the other hand, an alternative approach to overcome the necessity of large labeled datasets is to collect data using keywords from search engines, image tags from social media and crowd sourcing, e.g. Amazon Mechanical Turk[54], which decrease the cost of annotation remarkably. Once some percentage of noise in the dataset is accepted, which is almost inevitable with the mentioned methods, it is possible to build enormous task-specific datasets. These low-effort data collection techniques however are not reliable, and they may lead to misleading and inaccurate information due to the noise that they include.

(a) Pastures, Moors and heathland, Sea and ocean

(b) Pastures, Mixed forest, Beaches, dunes, sands

Figure 1.1: Missing class label in red and extra class label assignment in blue

In [27], noise is defined as anything that obscures the relationship between the class of a sample and the features of that sample. In the literature, two types of noise are described: feature noise and label noise[15]. They both degrade the accuracy of classifiers by misguiding neural networks while training. Feature noise is anything that obscures the attainable information by affecting the observable features of image. For example, color, light, or shape variations can affect the attainable information adversely. On the other hand, label noise is defined as erroneous matching of a class with a sample. Label noise is also branched into two types in itself, which are missing class label and extra class label assignment, and both types can be observed in Figure 1.1. The former occurs when a label that must have been assigned to a sample is not assigned. The latter occurs when an assigned label does not belong to the sample. This false matching can have detrimental

effects on training, especially when a sample has at least a missing class label as well as an extra class label assignment, which is called wrong class label assignment. According to [76], noise in labels is more harmful than noise in attributes. Thus, to obtain a more robust model to noise, we choose to focus on label noise and refer to it in the rest of this work.

Too much noise often causes trouble in the training process of neural networks, because noise distorts the course of learning, which causes neural networks to produce wrong results and predictions. Moreover, while the accuracy of predictions drops, the complexity of inferred models and number of necessary samples to train the model increase. Since neural networks can easily overfit to noisy data[71][68], dealing with noise can drastically improve the performance of neural networks. In particular, deep and complex neural networks tend to memorize the noise in the data, while ignoring the overall pattern and structure. That is why, it is a challenge to build robust DNNs to severe label noise. This notion captivates many researchers to fix label noise related issues and ameliorate the ability of a network to be not affected by label noise.

Learning with label noise is a relatively new topic in computer vision, and most research in the literature that tries to overcome the negative effects of label noise concentrates on single label classification rather than multi-label classification. This is due to the fact that the vast majority of computer vision datasets consist of single label images, and there are many techniques that focus on classifying isolated objects. Single label classification methods are also easily extendable through detection algorithms to many scenarios, where identification of multiple objects are necessary. For example, R-CNN[18] and YOLO[51] are very effective techniques at selecting interest regions and repurposing CNNs to recognize multiple objects in a frame. These techniques can be adapted to various domains, using multi-class classification models that are trained on a huge variety of object types, or different specialized classifiers. There are also several problem transformation methods that can transfer developments in single label domain to multi-label domain, such as binary relevance method[50], label powerset transformation[55], and various ensemble methods[62]. For all the mentioned reasons, developing single label classification techniques that are robust to label noise has greater benefit and practical importance, and thus studied more intensively than label noise tolerant multi-label classification techniques.

However, there are many situations where singling out entities in a frame may be impossible or inconsequential. In certain domains, relative positions of objects or their relationship to each other is an important aspect in classifying them. Moreover, background information may be essential to recognize certain features in an image. It is also possible that an object in an image represents numerous classes. In such cases, it is preferred to evaluate the whole image at once — which is referred to as multi-label classification — rather than splitting it into multiple bounding boxes as in R-CNN.

In multi-label classification, there is no constraint to the number of classes an instance can be assigned to. The task is to predict the proper label set for the unseen sample. Learning from multi-label data was initially motivated by text categorization and medical imaging tasks[61]. For example, a comment on the internet could be labeled as toxic as well as insulting. Similarly, a patient may have been diagnosed with tuberculosis and pneumonia as a result of chest X-ray. Today, multi-label learning is used extensively in many different domains, including movie genre categorization[65] and ontological function characterization of long non-coding RNAs[72].

Nonetheless, multi-label data can be very hard to classify due to its complex input and output space. In multi-label datasets, each instance may be assigned to multiple classes. This complicates mapping a relationship between labels and training instances for DNNs. Especially class-imbalanced datasets introduce bias in the training process, which makes it harder to associate the correct classes with proper training instances. Furthermore, it is very hard to find clean multi-label datasets. There are many well-established, clean single label archives such as MNIST[39], CIFAR[33] and ImageNet[12] that are acknowledged benchmarks in the literature. On the other hand, there is no commonly accepted clean data in multi-label domain. Therefore, it is critical to develop multi-label classification methods that are robust to label noise. Even though noisy learning on multi-label data is researched in the literature[74][73], to the best of our knowledge, there is no substantial advancement in the field yet.

Along with computer vision, remote sensing is an another research area that adopts multi-label learning to gain a better insight into geography, geology, and meteorology. Remote sensing is the science of information acquisition about a surface from distance without making physical contact. For example, [57] uses multi-label classification to match satellite images with their land-cover class labels, hoping to develop a scalable and accurate Earth observation image search and retrieval system.

In the recent years, the availability of remote sensing data has dramatically increased due to new launches of earth observation satellites such as Sentinel-2. However, from-satellite-retrieved remote sensing imagery often includes feature noise as a result of spatial resolution differences. Furthermore, remote sensing data contains multi-spectral bands that make the labeling process laborious and expensive. Different spectral bands must be evaluated together meticulously in order to identify various entities in a frame. This exhausting process requires expert knowledge in the field and introduces noise in labels due to its complexity. Considering the mentioned reasons, noise detection and overcoming techniques are essential for success in the field. To alleviate the negative effects of feature noise, better high-precision equipment or resolution enhancement methods may be adopted. On the other hand, increasing the number of expert annotators may be an alternative approach to overcome the adverse effects of label noise, which would be a costly solution. In this thesis, the damage of noisy labels on classification of remote sensing data is aimed to be compensated through algorithmic design.

We refer to the problem that we are trying to solve as **Classification of multi-label Remote Sensing Images with Noisy Labels**. To address the mentioned problems in noisy learning and remote sensing, we propose a collaborative learning model with an additional algorithm to identify, rank, and fix noisy samples in multi-label data, called **CCML**, **Consensual Collaborative Multi-label Learning**.

The main contributions of this work are as follows:

- We propose a collaborative training framework to identify different types of multi-label noise in data by making no prior assumptions about the noise distribution.
- The proposed method offers an algorithm to deal with noisy samples either by removing them from back-propagation or fixing their noisy labels.
- We adapt the Co-teaching paradigm to multi-label data, where samples are ranked accord-

ing to their noise levels and the ranking information is exchanged.

- The proposed method locates noisy labels of each sample and determines their type by taking advantage of group lasso.
- The proposed technique is evaluated on the remote sensing archives BigEarthNet and multi-label UC Merced Land Use with synthetic label noise.

The rest of the thesis is organized as follows:

Chapter 2 explores previous work in learning with noisy labels.

Chapter 3 explains the proposed methodology.

Chapter 4 describes the datasets, the network architecture and the experimental setup used in this work.

Chapter 5 analyzes the results of the conducted experiments and provides a deeper understanding about the proposed method.

Chapter 6 summarizes the thesis, the results, and mentions future work.

2 Related Work

Over the last few years, deep learning has thrived in image classification. However, deep learning requires an excessive amount of labeled data in order to get good results for complex tasks. This is not always feasible due to several reasons such as labeling cost and difficulty of proper annotation. Moreover, deep learning models tend to overfit to label noise in data[71]. For this reason, label noise is a common problem in image classification. It has severe negative effects on model accuracy, especially if the dataset is large and hard. In cases of an imbalanced class distribution, where some classes appear many times in the data while others appear only a few times, noise may have detrimental effects on predictions of a network. Various methods have been proposed to overcome the negative effects of this problem on training. In this chapter, we evaluate previous work from the literature by categorizing the approaches into two groups following [1]. The analyzed literature aims to develop a noise-tolerant architecture that effectively learns the underlying pattern in the data despite the present noise.

2.1 Image Classification with Label Noise Structure Modelling

The methods in this category attempt to model the noise distribution that is present in the data. The extracted noise information is then utilized in different ways to overcome the negative effects of it on the training. Some approaches remove the detected noisy samples all together from training, while others only remove the noisy labels leaving the unlabeled samples to train their network further. There are other methods that weigh down the importance of noisy samples on training seeking to degrade the possible adverse effects. Relabeling improper labels is another approach that pursues to clean the dataset from noisy instances, which impacts the accuracy of the classifier positively.

Patrini et al. propose a loss correction procedure by estimating the noise transition matrix of data[49]. The proposed method claims to fix the class-dependent label noise in the data, given the probability of each class being corrupted into another. However, as the number of classes increases, it becomes harder to estimate the noise transition matrix, making the method unscalable. In [19], the correct labels are considered as latent variables in the presence of noisy labels, and the expectation-maximization[10] algorithm is applied, to iteratively calculate the correct labels as well as the network parameters. This scheme is also extended to scenarios, where noisy labels are dependent on the features, by modeling noise via a softmax layer that connects correct labels to noisy labels. Sukhbaatar et al. explore the performance of discriminatively-trained CNNs on noisy labels using an additional fully connected layer at the end of their network to adapt the predictions to the noisy label distribution of the data[56]. In [69], data is partitioned into multiple subsets and used to train different classifiers. If all of the classifiers agree on a label from the original dataset, the label is updated with the agreed prediction. This process repeats for several stages, which gradually improves the overall performance. Nonetheless, this process depends very much on the quality of the classifiers, and it can take a lot of time when large and complex datasets are used. Dehghani et al. use a student model that is trained on noisy data along with a

teacher model that exploits the noise structure that is extracted by the student model[11]. However, this approach needs data with both clean and noisy labels, which is not always available. On the other hand, [43] proposes a self-error-correcting CNN that can work on completely noisy data. The network swaps potential noisy labels with the most probable prediction of the network while simultaneously optimizing the model parameters. Failing to distinguish hard samples from noisy ones is a common problem when fixing labels, which may result in an undesired situation where the model swaps labels in a delusional way. To avert that, additional policies are often needed. Wu et al. use semantic bootstrapping to detect noisy samples and remove them from training[66]. Removing as few samples as possible is an important aspect of the noisy sample removal process to avoid unnecessary information loss. On the other hand, [47] trains a DNN on a noisy dataset and filters the noisy labels out while keeping the samples for the training in an unsupervised fashion. This prevents loosing training samples, and thus allows DNN to learn from more examples. MentorNet[29] imposes a data-driven sample weighting curriculum on the student network to choose probable clean samples from data. Han et al. propose a learning paradigm called Co-Teaching, which we predicate our collaborative model on[22]. Under Co-Teaching, two networks train simultaneously and choose clean batches feeding to each other. Each network back-propagates the mini-batch that is chosen by its peer network to update itself. Han et al. demonstrate that Co-Teaching is an effective method when dealing with label noise. [23] bases on the principle of Co-Teaching with an additional discrepancy measurement to make the used networks diverge. They use two peer networks that learn different features of the same probability distribution. Both networks then select clean samples for each other based on their predictions to improve performances mutually. Ren et al. propose a meta-learning algorithm where a weighting factor is assigned to every training sample based on its gradient direction[52]. Although this method achieves impressive results, it needs a clean validation set, which may not always be present.

The advantage of the above mentioned label noise structure modelling methods is that they can be decoupled from the training process and the application domain. This modularity allows these approaches to be used and tested with different network architectures without demanding modifications on the framework. However, the success of the above mentioned work almost always depends on the accurate estimation of noise distribution in data, thus to the choice of the classifier. For this reason, some of the mentioned methods make prior assumptions on noise structure, which restricts the applicability and extendability of the suggested work.

2.2 Inherently Label Noise Robust Image Classification Methods

The methods in this category aim to design models that are intrinsically robust to label noise. By design, these models try to reduce the memorization and influence of noisy labels on the result, without explicitly modeling the noise distribution of data.

Ghosh et al. study different loss functions such as categorical cross-entropy (CCE), mean square error (MSE), and mean absolute error (MAE) for noisy multi-label classification, and argue that MAE is more robust to label noise compared to the others[17]. In [75], it is stated that MAE shows poor performance with more complex datasets, and proposed a set of robust loss functions that combine CCE and MAE, which is known as noise-robust alternative of CCE in the litera-

ture[17]. Other techniques modify existing loss functions by adding a regularization term to get more robust models to label noise. Regularization is already a well-known overfitting avoidance method in the literature, and its applications on noisy data yield promising results. In [30], an additional softmax layer is added to the network modeling the possible label noise statistically. Hendrycks et al. shows that pre-training improves model robustness in cases of label corruption and class-imbalance[26]. In [60], it is shown that overfitting avoidance techniques such as regularization and dropout can be partially effective when dealing with label noise. However, in these cases, label noise may still impede the classifier's quality, resulting in an accuracy drop. Moreover, since noise injection is already an overfitting prevention technique that attempts to improve generalization performance[48], regularization should be applied prudently. Too much of it can lead to scenarios where instead underfitting happens. Meta-Learning and ensemble methods are other approaches that are used in the literature against label noise. Li et al. aim to find noise-tolerant model parameters using a teacher model along with a student model to make accurate predictions by optimizing a meta-objective, which encourages the student model to give consistent results with the teacher model after introducing synthetic labels[40]. In [59], a robust alternative to AdaBoost for multi-class noisy data is offered.

The methods mentioned above focus on mostly single label data, and many of them are hard to extend or apply into multi-label domain without modifications. This is because there is only one type of noise in single label data: the wrong class label assignment. On the other hand, in multi-label datasets, there may be more than one type of noise, namely wrong class label assignments, missing class labels and extra class label assignments. They all require different types of noise avoidance techniques. Moreover, algorithms that do multi-label classification are often prone to error due to their complexity. Thus, more advanced algorithms that identify different types of noise and deal with them accordingly are needed. Some of them that are classified under Inherently Label Noise Robust Image Classification Methods are mentioned below.

Bucak et al. present a ranking-based multi-label learning framework that exploits the group lasso method to enhance accuracy with incomplete class assignments[6]. The proposed method computes an error for every assigned class against the unassigned ones for each sample. The computed errors are then combined into ranking errors for each unassigned class, which indicate the possible missing classes for the sample. Finally, all the errors for the sample are summed up in a final ranking error. The aforementioned formula is used to regularize the network in [6], giving empirically robust results against missing class labels. Durand et al. propose a modified binary cross-entropy loss, which reduces the negative effects of missing class labels on training[13]. Jain et al. try to treat the problem of missing class labels by defining a novel loss function[28]. Their loss function prioritizes the most relevant few labels over the irrelevant majority. They also provide unbiased estimates of true labels of a sample without omitting the missing labels completely. Nonetheless, these strategies are designed to handle one type of noise in the data, such as missing class labels or extra class labels, and they are unable to identify all types of noise and solve entailed problems without prior assumptions.

Experiments show that inherently label noise robust image classification methods only seem to work with random or simple cases of label noise, and lose their competency in case of more complex noise structures[60][1]. Moreover, some of these techniques such as [40] require additional training steps to fine-tune model parameters, which may be unfeasible with very large training data.

3 Proposed Remote Sensing Image Classification Method for Noisy Labels

In this chapter, the proposed Consensual Collaborative Multi-label Learning (CCML) method is presented. We first explain the properties of our method and describe the problem that we are trying to solve. We then explain the different parts of our method. The method consists of four major parts: the discrepancy module, the group lasso module, the flipping module, and the swap module. The discrepancy module allows the two networks that are used collaboratively in the method to learn different feature sets, while ensuring consistent predictions. The group lasso module identifies potential noisy classes by computing a ranking loss for every sample and penalizing noisy samples. The flipping module flips the noisy labels that are approved by the two networks, and finally, the swap module exchanges the ranking information between the networks by taking the Binary Cross Entropy (BCE) loss and the ranking loss into consideration. The pseudo code of the proposed method can be found at the end of the Chapter.

We propose a solution for all types of multi-label noise without making prior assumptions about the noise characteristics of the data. To the best of our knowledge, there is no such research yet that identifies and overcomes the adverse effects of different types of multi-label noise within the same algorithm. We use the principle of Co-Training to exchange loss information between networks and choose low loss samples from the training set. Co-Training is a semi-supervised learning technique that was first proposed in [4] in order to overcome the labeled data insufficiency. The framework has successfully improved the performance of learning from vast amounts of unlabeled data. It was originally designed to classify web pages, where two learning algorithms were trained together with two different views of the same page to make use of unlabeled data. Fundamentally, Co-Training attempts to learn different classifiers using conditionally independent feature sets. It uses each classifier's most confident predictions to construct labels for unlabeled data iteratively. It has a broad range of use cases including domain adaptation[8], multi-view spectral clustering[37], email classification[32] and so forth. However, it is shown by [35] that Co-training is only beneficial under the assumption that the used feature sets are independent and sufficient for classification. Learning from the same view of the data worsens the results more than it makes them better. Inspired by [23], we use two DNNs with the same architecture, which are enhanced with discrepancy modules to make them learn complementary feature representations while attaining the same class distribution. This improves the ability to select clean instances immensely, since two networks learn different features and correct each other by exchanging their loss information. The proposed algorithm can also identify label noise in the data, taking the predictions of both of the used networks into consideration by feeding them into the group lasso module. The group lasso module attempts to detect label noise including its type in a sample by calculating two loss terms. The first loss term gives a measure of per sample probability of having a missing class label, and the second loss term gives a measure of per sample probability of having an extra class label assignment. Then the module aggregates both loss terms into a ranking loss. Finally, the algorithm combines the BCE loss and the ranking loss

into the compound loss. Samples with larger compound loss values are more likely to be noisy. Thus, the algorithm excludes them from back-propagation by ranking them lower, expecting to achieve better results by learning from relatively clean samples. Our algorithm can also weigh down or up the noisy samples based on their noise type. For example, excluding a sample with an extra class label assignment may be more beneficial than excluding a sample with a missing class label. The group lasso module also selects a potential noisy class label per sample that can be later fixed by the flipping module, if both of the networks are convinced that the class label is noisy. The flipped label then gets incorporated into the training.

Data cleansing algorithms' general procedure is to train the classifier first on noisy data, detecting the noisy samples, and then to filter them out and train again[5]. Some algorithms repeat this procedure until the algorithm cannot find noise in the dataset any more. This can take a lot of time, especially if the dataset is complex and large. An alternative is to construct two separate algorithms: the first detecting and filtering noisy instances in data, and the second classifying the data[5]. On the contrary, our algorithm identifies potential noisy instances during the training with the help of the group lasso module, and relabels the noisy labels. Thus, the misleading effect of noisy samples on our classifier is getting minimized. Our algorithm does not need multiple iterations or two separate algorithms to filter and classify data.

The proposed method is architecture-independent, meaning it can be implemented using any architecture. Since every DNN has different ability to extract key features, discrepancy module's position can differ from network to network. Another distinguishing property of the proposed algorithm is its modularity. The flipping module as well as the swap module can be completely decoupled from the used classification approach and from each other, which makes them reusable with any classification algorithm.

3.1 Problem Statement

We consider a multi-label classification problem with a training set $\mathcal{D} = \{(x_1, y_1), ..., (x_n, y_n)\}$, where x_i denotes the i^{th} training instance that is labeled by a multi hot vector $y_i = (y_i^1, ..., y_i^m) \in \{0,1\}^m$ representing the corresponding label over m classes, where $y_i^k = 1$ if $k \in y_i$, and $y_i^k = 0$ if $k \notin y_i$. We use two DNNs with the same architecture that are represented as f and g with parameters θ and $\hat{\theta}$, respectively. We employ Binary Cross Entropy as our classification loss for each network, which are denoted by L_f and L_g as shown below:

$$L_f(x_i) = -\sum_{k=1}^{m} y_i^k \cdot \log\left(f_\theta(x_i^k)\right) + (1 - y_i^k) \cdot \log\left(1 - f_\theta(x_i^k)\right) \tag{3.1}$$

$$L_g(x_i) = -\sum_{k=1}^{m} y_i^k \cdot \log(g_{\hat{\theta}}(x_i^k)) + (1 - y_i^k) \cdot \log(1 - g_{\hat{\theta}}(x_i^k)) \ , \tag{3.2}$$

where $f_\theta(x_i^k)$ and $g_{\hat{\theta}}(x_i^k)$ are the normalized predictions of the corresponding networks for class k in sample i.

BCE loss is a measure of probability error of network predictions. It is a convenient way to calculate error in answering binary choice questions such as yes/no or 1/0. Thus, we consider each class assignment a binary classification problem and calculate separate BCE losses, which

are then summed up. For the normalization process, we use the sigmoid function, which converts the raw (non-normalized) predictions of the networks into probability scores between zero and one. The sigmoid function is a popular choice in multi-label classification tasks, because it produces probabilities that are independent from each other and not constrained to sum to one. Evaluating class predictions in a not mutually exclusive way allows the networks to choose multiple classes at the same time.

When using collaborative training, it is critical that the networks are complementary to one another and diverge as much as possible while predicting the same class distribution. The networks must be able to fix each other's mistakes in the training process by exchanging information. If they do not learn diverse features on the same dataset, they will not be able to enhance the predictions of each other mutually. To achieve the desired diversity, we embed a discrepancy module in our framework and include its products, L_D and L_C, in the final loss. The details of the discrepancy module can be found in Section 3.2. We also choose a certain number of low loss samples from each mini-batch to update the weights of the networks. The samples are determined following a sample selection policy that is explained in Section 3.5.

So the final losses that are minimized by f and g are as follow:

$$Loss_f^b = \frac{\sum_{i=1}^{R} L_f(x_i)}{R} + \lambda_2 \cdot L_C - \lambda_3 \cdot L_D \tag{3.3}$$

$$Loss_g^b = \frac{\sum_{i=1}^{R} L_g(x_i)}{R} + \lambda_2 \cdot L_C - \lambda_3 \cdot L_D \ , \tag{3.4}$$

where b represents the mini-batch, and R represents the number of the chosen low loss samples. λ_2 and λ_3 represent the weights for L_C and L_D.

3.2 Discrepancy Module of the Proposed Method

The discrepancy module consists of two components: the discrepancy and the consistency component. The former makes sure that the networks learn distinct feature sets, and it is inserted in-between the chosen layers of each network. The layers to be chosen depend on the network architecture, and the discrepancy component can be placed anywhere in the network. However, since the first layers of deep learning models are responsible for extracting primitive key features of input data such as lines and shapes, it is not recommended to place the component after the half of the layers in the network. The discussion on the position of the discrepancy component can be found in the chapter 4. On the other hand, the consistency component makes sure that the final predictions of the networks are similar. The networks should learn different features while predicting the same class distribution. Hence, we plant the consistency component at the end of the networks to reduce the discrepancy and achieve comparable predictions.

The inserted **discrepancy component** gets the logits of the layers that come before the component as input and calculates the discrepancy loss L_D. It is to be underlined that the discrepancy module is placed between the two networks. Thus, the input is the logits of both of the networks. The calculated loss is written as:

$$L_D = M(\hat{F}_i, \hat{G}_i) \ , \tag{3.5}$$

where M is the choice of discrepancy measurement, and $\hat{F}_i$ and $\hat{G}_i$ represent the logits of the layers that come before the component. Formally, $\hat{F}_i = f_{\theta(1:l)}(X_i)$ and $\hat{G}_i = g_{\hat{\theta}(1:l)}(X_i)$. l denotes the layer where the discrepancy component is placed, and $\theta(1:l)$ and $\hat{\theta}(1:l)$ stand for the parameters of networks till the layer l. X_i represents a batch of samples.

The **consistency component** is placed after the last layer of the networks, and it gets the logits of the last layer as input:

$$L_C = M(F_i, G_i) \ , \tag{3.6}$$

where F_i and G_i denote the logits of the last layer of the networks, analogous to 3.5. The flowchart of the discrepancy module is displayed in Figure 3.1.

As the discrepancy measurement M, any statistical distance function that measures the difference between two probability distributions can be used. Distance measures have a wide range of application area such as information theory, neuroscience and machine learning. In machine learning, they are commonly used as loss functions that measure the distance between the model predictions and the labels. They are also elementary for many categorization algorithms[42][3]. While Kullback-Leibler divergence[36] is a popular and effective choice of measure in machine learning, it is distribution-wise asymmetric. To overcome the asymmetry problem, Jensen-Shannon[44] divergence metric is developed. However, Kullback-Leibler measure as well as Jensen-Shannon metric fail to capture the topology of distributions, which makes them undesirable when the geometry of the underlying space is a crucial factor in disentangling distributions[14]. On the other hand, algorithms such as Maximum Mean Discrepancy (MMD)[21] and Wasserstein metric[63] are able to assess the topology of the feature space. This property makes them convenient for diverging two distributions from one another.

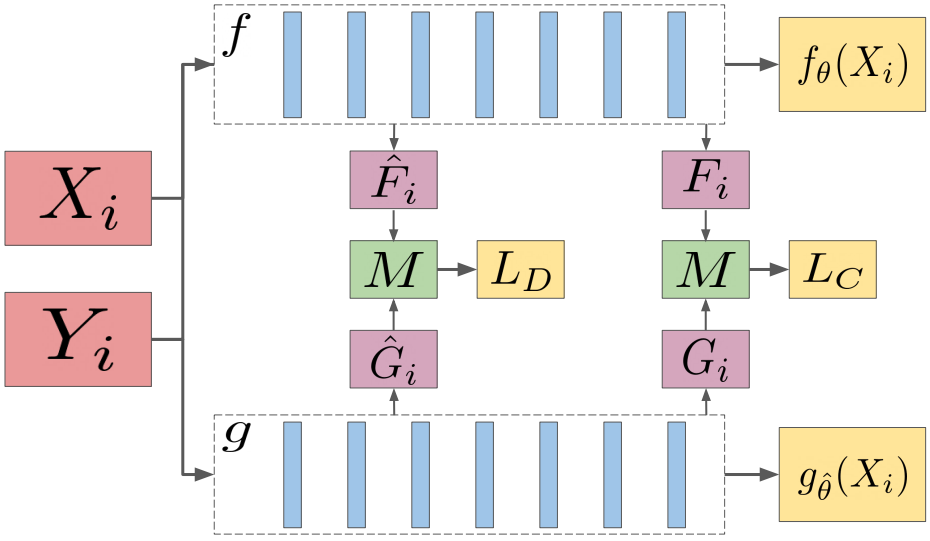


Figure 3.1: Flowchart of the Discrepancy Module. f and g represent the networks; red boxes are the inputs and the yellow boxes are the outputs. $\hat{F}_i$ and $\hat{G}_i$ stand for the intermediate logits of the networks. F_i and G_i stand for the logits of the last layers. M is the discrepancy component.

MMD is used in [23] to disentangle probability distributions. We follow their choice of measure in our method. Wasserstein metric would be also a very good candidate for the discrepancy measurement, since it takes the topology of the feature space into account. However, its cubic complexity $O(n^3 \log(n))$ makes it inadequate for our algorithm. Along the MMD, we explore the Jensen-Shannon divergence[44] as well, which is a widely used symmetric measure. The performance comparison can be found in Chapter 4.

Maximum Mean Discrepancy is a distance measure between two distributions with a quadratic computational cost, and it is defined as the distance between the mean embeddings of the distributions in Reproducing Kernel Hilbert Space (RKHS)[2]. In other words, if the mean values of the functional mappings of the distributions to a high dimensional space is close, the distance between the distributions is small.

MMD to measure the difference between two distributions p and q can be formally defined as shown below:

$$MMD(p,q) = \|\mu_p - \mu_q\|_H \ . \tag{3.7}$$

Above, μ_p and μ_q denote the expected values (means) of the distributions p and q; H represents the RKHS, and $\|\|_H$ represents its norm.

Avoiding the explicit mapping of distributions with the kernel trick, the formula of the MMD can be written as:

$$MMD(p,q) = \frac{1}{m^2}\left(\sum_{i=1}^{m}\sum_{t=1}^{m} k(x_i^p, x_t^p) - 2\cdot \sum_{i=1}^{m}\sum_{t=1}^{m} k(x_i^p, x_t^q) + \sum_{i=1}^{m}\sum_{t=1}^{m} k(x_i^q, x_t^q) \right) \ , \tag{3.8}$$

where x_i^p and x_i^q are samples from respective distributions.

Following [23], we have employed the radial basis function kernel[64] in our experiments:

$$k(s_1, s_2) = \exp\left(-\frac{\|s_1 - s_2\|^2}{\sigma} \right) \ . \tag{3.9}$$

The value of σ is crucial in MMD's ability to diverge two networks, and a discussion on the matter can be found in Chapter 4.

3.3 Group Lasso Module of the Proposed Method

Binary Cross Entropy is a widely accepted measure of prediction accuracy in multi-label learning. We follow the literature and use BCE to minimize the respective losses of our networks. However, when training data contains label noise, the networks can overfit to the training data and perform poorly in the test phase. Thus, an additional mechanism is necessary to avoid being misguided by noisy training samples. The additional mechanism can have many forms such as regularization, noisy sample exclusion or noise correction. Bucak, et al.[6] use a regularization technique to deal with incompletely multi-label data, where they selectively penalize noisy samples by giving every sample a ranking loss. The ranking loss per sample is calculated through an aggregation of multiple ranking errors for every unassigned class. They exploit the group lasso

technique to combine these errors in ranking assigned classes against unassigned ones. They aim thereby to detect missing class labels in each sample and build a more robust model to incomplete class assignments. The group lasso was introduced by [70] as an extension to regular lasso to group points of interest together for accurate prediction in regression. It is effective, because it groups variables together to be included or excluded wholly, as opposed to lasso, which only selects variables individually. Inspired by [6], we develop a ranking error function that deals with all possible types of noises — missing class label, extra class label assignment, wrong class label assignment — without prior assumptions. The existing ranking methodology for missing class labels in [6] is extended to detect extra class label assignments and wrong class label assignments as well. However, we do not use our ranking error function as a regularizer. Instead, we use it, along with the BCE loss, to detect noisy samples within a mini-batch to be excluded from back-propagation. The exclusion of noisy samples from back-propagation prevents overfitting to erroneous training data. The motivation behind using group lasso is the fact that it is able to locate noisy classes in samples creating the opportunity to fix them. It also provides information about label noise type in samples. We first define the ranking error function as follows:

$$E_{k,l}(x_i) = \max\Big(0, 2\cdot\big(f_l(x_i) - f_k(x_i)\big) + 1\Big) \ , \tag{3.10}$$

where k represents a class index labeled by 1 and l represents a class index labeled by 0 for sample x_i. $f_k(x_i)$ and $f_l(x_i)$ denote the predictions of the classes k and l, respectively. The ranking error function gives a measure of potential noise in a class combination, which can be observed in Table 3.1. In case of a clean prediction, the ranking error is equal to 0. On the other hand when there is label noise, the ranking error function returns a positive result.

The key idea behind the ranking error function is that if $k \in y_i$ and $l \notin y_i$, then $f_k(x_i) > f_l(x_i)$. However, the trustworthiness of the ranking error function depends very much on the prediction accuracy of the model. If one wants to decide whether the class combination includes noise, using $E_{k,l}(x_i)$ in the beginning of the training process may lead to misleading results when the networks are unstable.

Table 3.1: Ranking Error Function Values

	f_l	f_k	E
clean	0	1	0
miss	0	0	+1
extra	1	1	+1
wrong	1	0	+3

To find the potential noise rate and noise type in a sample, losses from class combinations are gathered into two components using the group lasso technique. The first component calculates an aggregated loss for the sample taking missing class labels into consideration, and the second component calculates an aggregated loss for extra class label assignments. This approach allows our algorithm to rank noisy samples according to their noise rate as well as their noise type by adjusting the factors (α and β) of the two losses. The combined ranking loss per sample is defined as follows:

$$Lasso_f(x_i) = \alpha \cdot \sum_{l=a+1}^{m} \sqrt{\sum_{k=1}^{a} E_{k,l}^2(x_i)} \; + \; \beta \cdot \sum_{k=b+1}^{m} \sqrt{\sum_{l=1}^{b} E_{k,l}^2(x_i)} \quad , \tag{3.11}$$

where a and b denote the number of assigned labels and unassigned ones in a sample, respectively. $Lasso_g(x_i)$ is computed analogous to $Lasso_f(x_i)$.

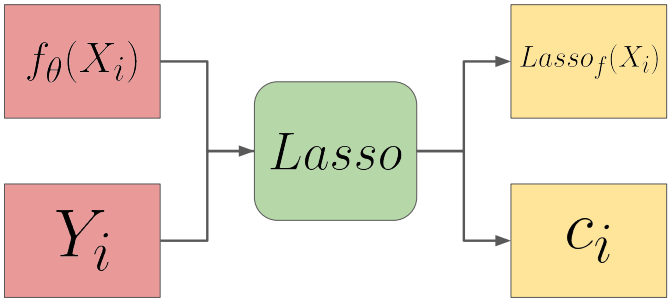


Figure 3.2: Flowchart of the Group Lasso Module. It gets the normalized predictions and labels as input, and outputs the ranking loss and potential noisy classes. The calculation for the network g is analogous to the calculation for the network f.

We also save ranking errors per class before summing them up into the ranking loss per sample, which allows noisy labels to be located. The located labels are thereafter transferred into the flipping module to be fixed. There may be more than one noisy label in a sample, whether it is a missing or an extra label. However, locating each of them correctly is an arduous task, and when done wrong, may lead to detrimental effects on training. Thus, we select one label per sample with the highest ranking error to be transferred to the flipping process.

At the end, the group lasso module returns a ranking loss and a potential noisy class for every sample in a mini-batch, as can be seen in Figure 3.2. The returned ranking losses are added to the BCE loss by some factor to be used to determine the noisy samples. The potential noisy classes are sent to the flipping module.

3.4 Flipping Module of the Proposed Method

The flipping module consists of two components: Noisy class selector (NCS) and Noisy class flipper (NCF). **NCS** takes the previously calculated ranking losses, potential noisy classes and an additional flipping percentage argument as well, as input. The flipping percentage argument dictates the number of classes that get chosen to be sent to NCF. When it comes to relabeling potential noisy labels, the difficult part is distinguishing hard samples from noisy samples. In order to overcome this hardship, the component first compares the chosen classes by the two networks and eliminates the ones that are not common, which prevents relabeling false positives — classes that are in fact not noisy. After both networks agree upon the noisy classes, the ranking losses of these noisy classes from the two networks are summed up. The component then selects the samples with the biggest ranking losses according to the value of the flipping percentage.

Let $l_i = (l_i^1, l_i^2, ..., l_i^n)$ and $\hat{l}_i = (\hat{l}_i^1, \hat{l}_i^2, ..., \hat{l}_i^n)$ be ranking losses, and $c_i = (c_i^1, c_i^2, ..., c_i^n)$ and $\hat{c}_i = (\hat{c}_i^1, \hat{c}_i^2, ..., \hat{c}_i^n)$ be the chosen potential noisy classes for every sample for the mini-batch i with size n. The noisy classes C_i to be proceeded to NCF are chosen as follows.

$$I_i = \left\{ \frac{(l_i^k + \hat{l}_i^k)}{2} \middle| k \in \{1, ..., n\}, c_i^k = \hat{c}_i^k \right\}, \tag{3.12}$$

$$C_i = \{ c_i^a = \hat{c}_i^a | a \in D_i \} \ , \tag{3.13}$$

where D_i represents the set of indices of k largest elements in I_i.

The task of **NCF** component is to flip the labels of the forwarded noisy classes by the previous component. Flipping means that if a noisy class is labeled with 0, NCF turns the label into 1, and if the noisy class is labeled with 1, NCF turns the label into 0. After the flipping is done, an additional group lasso component (Section 3.3) recalculates the ranking losses for the mini-batch with flipped labels. The flipped labels are also used for Binary Cross Entropy loss calculation. An example scenario for the flipping module's working mechanism can be found in Figure 3.3.

As mentioned earlier, the algorithm's success in finding and fixing the noisy labels depends on the accuracy of the network predictions, which are in general erroneous in the beginning of the training phase. Thus, if the label flipping module starts flipping noisy labels when the networks are unstable, it may flip the labels of hard but clean samples. To avoid that, the flipping module is initiated after the network is somewhat stable. The value of this parameter can be set looking at the learning curve of the networks. It is also important to point out that the number of labels that are flipped is significant for the module's success. As the networks learn from each other's mistakes, the number of consented labels may increase, and a high flipping percentage may have negative effects rather than positive.

It should be underlined that the main goal of the flipping module is not cleaning the existing dataset. Fixes that are made by the module are not permanent and do not modify the actual dataset labels. The algorithm rather takes advantage of the group lasso calculation by fixing the located noisy labels as safe as possible during the training and using the fixed labels for the back-propagation.

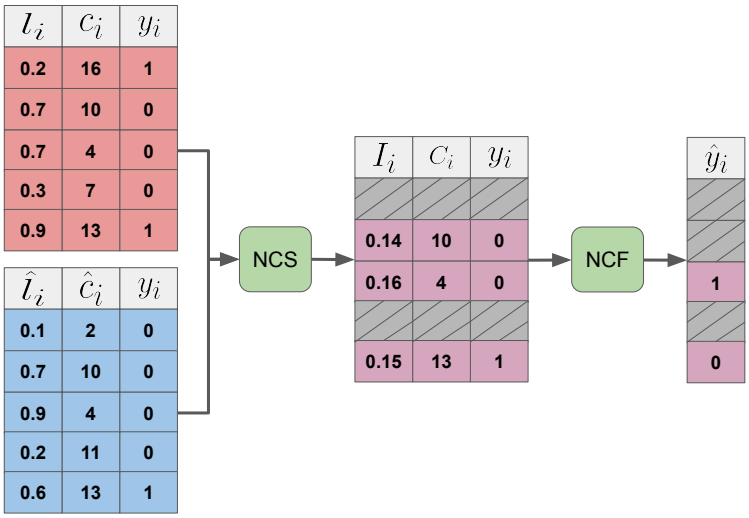


Figure 3.3: Diagram of the Flipping Approach. l_i and $\hat{l}_i$ represent the ranking losses. c_i and $\hat{c}_i$ imply the potential noisy class indexes. NCS outputs the consented noisy labels C_i and sums up the ranking losses from both of the networks. NCF gets the labels y_i as input and flips a percentage of the consented noisy labels with the largest ranking losses. In this example the percentage of flipping is set to 2/3.

3.5 Swap Module of the Proposed Method

The swap module adds the ranking losses, which are calculated by the equation 3.12, to the BCE loss, and selects R elements that produce the lowest losses for each network. The ranking information of lowest R samples are exchanged between the two networks, where the network f uses R lowest loss samples that are found by the network g to update its weights, and vice versa.

$$B_i^f = \frac{\sum_{i=1}^{R} L_f(x_i) + \alpha \cdot Lasso_f(x_i)}{R} \quad , \quad x_i \in m_g^R \tag{3.14}$$

$$B_i^g = \frac{\sum_{i=1}^{R} L_g(x_i) + \alpha \cdot Lasso_g(x_i)}{R} \quad , \quad x_i \in m_f^R \quad , \tag{3.15}$$

where B_i^f and B_i^g are the compound losses for each network, and m_f^R and m_g^R are the set of images that results in the R lowest losses for the networks f and g respectively. α is the trade off parameter representing the strength of $Lasso_f$ and $Lasso_g$ in the low loss sample selection policy. A visualization of the process is illustrated in Figure 3.4.

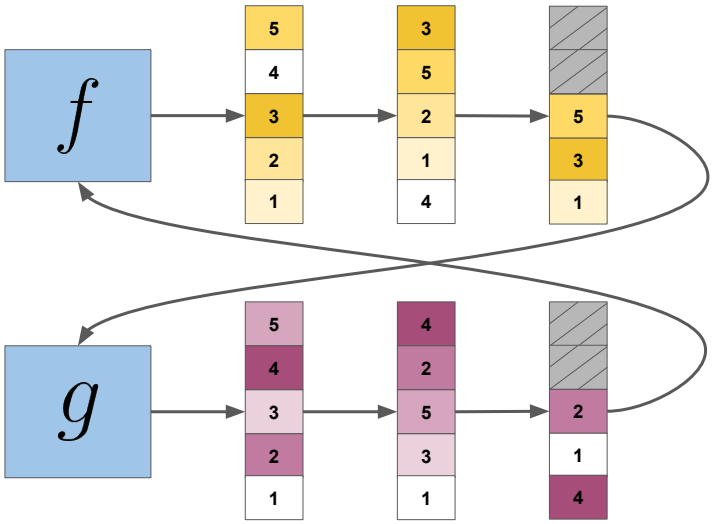


Figure 3.4: Flowchart of the Swapping Policy. The two networks exchange the ranking information. Each network then updates its weights using the samples with smaller loss values that are found by the opposite network.

Algorithm 1: Consensual Collaborative Multi-label Learning

```
for epoch = 1,...,E do
    for mini-batch = 1,...,B do
        Calculate the ranking loss;
        if flipping = True then
            Flip the noisy classes;
            Recalculate the ranking loss;
        end
        Calculate the BCE loss;
        Sum the BCE loss and the ranking loss;
        Get low loss samples;
        Swap the ranking information;
        Update θ and θ̂;
    end
end
```

4 Dataset Description and Design of Experiments

In this chapter, we first describe the datasets that are used to measure the performance of the proposed method. We then present the two different network architectures that are employed in the tests. Thereafter, the considered label noise approaches for the experiments are explained. Finally, we detail the experimental setting including the optimal values for the hyperparameters of the proposed method, considered label noise approaches and evaluation metrics.

4.1 Dataset Description

We perform experiments on two different remote sensing archives, namely BigEarthNet[57] and UC Merced Land Use dataset[67], to assess the performance of the proposed method and do a well-rounded analysis. As the title of the thesis states, we focus on remote sensing imagery in the multi-label setting. The behavior of the proposed method is illustrated in different scenarios by using different datasets. This provides meaningful insights to the proposed method. We choose BigEarthNet and UC Merced Land Use archives for testing purposes.

BigEarthNet is a large scale multi-label remote sensing archive, consisting of 125 Sentinel-2 image tiles. The image tiles were collected by two twin satellites, called Sentinel-2A and Sentinel-2B, which acquire optical imagery over land, and coastal waters. To construct BigEarthNet, the collected image tiles were atmospherically corrected, and divided into non-overlapping 590,326 image patches, each annotated by multiple land-cover classes provided by CORINE Land Cover Map (CLC) of 2018. Each image patch is composed of twelve spectral bands, four of which are 10m (120×120 pixels), six are 20m (60×60 pixels) and two are 60m (20×20 pixels). In our experiments, we use the 10m and 20m bands excluding the two 60m bands that are not related to land cover classes. The 20m bands are resized to 120×120 pixels using bicubic interpolation for coherent input dimensions. The channels are normalized according to their specific means and standard deviations that are provided by the archive.

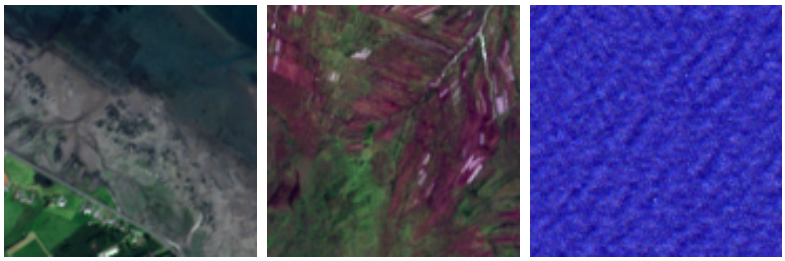
BigEarthNet was initially annotated with 43 land cover classes, some of which were challenging to be accurately described[58]. Therefore, the archive is updated with an alternative class-nomenclature consisting of 19 classes. Ten classes from the original label set are kept in the new label set, while twenty classes that have similar land cover features and spectral patterns are grouped into nine new classes. Eleven classes that are related to land use or require additional information to be recognized are eliminated from the label set all together. The new class-nomenclature increases the description quality of BigEarthNet's semantic context. Furthermore, it alleviates the class-imbalance problem that was present in the original label set, where the class distribution of data was highly imbalanced. The most common class had 217,119 samples, and the most uncommon class had 328 samples. The new class-nomenclature achieves a much more balanced archive with the removal of the rare classes.

Class-imbalance refers to the problem where classes in a dataset are not represented equally. Generally, classes that have more samples are learned easier and faster in classification because the model sees more examples of the class. As a consequence, samples with more common classes receive smaller loss values, whereas samples with less common classes receive greater loss values. On the other hand, loss values in a balanced dataset, where every class has approximately the same number of samples, are often proportional. The proposed method works best when the training set is balanced as much as possible, because it excludes the samples with greater loss values from back-propagation, which may coincide with the samples that have less common classes in a class-imbalanced dataset. If these samples are constantly excluded from back-propagation, they may never be learned by the model which may lead to poor test performance. Thus, we choose a subset of BigEarthNet to use in the experiments, and we reduce the class-imbalance further by introducing a 12 class-nomenclature. BigEarthNet images were collected between June 2017 and May 2018 over ten European countries, and the archive is divided into subsets each representing a country. Among them, the Ireland subset is chosen to be subjected to the experiments. A set of example images from the subset is displayed in Figure 4.1. Table 4.1 shows the number of training, validation and test samples in the Ireland subset of BigEarthNet using the 19 class-nomenclature. To obtain a more balanced dataset, classes that have less training samples than a defined threshold are eliminated. The eliminated classes from Table 4.1 are Industrial Commercial Units, Permanent Crops, Agro-foresty Areas, Natural Grassland and Sparsely Vegetated Areas, Beaches, Dunes, Sands, Coastal Wetlands and Inland Waters. To display the negative effects of class-imbalance on multi-label classification with noisy labels, we include results on the Ireland subset using 43, 19 and 12 class-nomenclatures in Chapter 5.

UC Merced Land Use is the other dataset that a set of experiments were conducted on. The archive is composed of 2100 aerial ortho images that were retrieved from the United States Geological Survey National Map. There are 100 images in the RGB color space for each of the 21 classes in the archive. Each image measures 256x256 pixels and has a spatial resolution of 30m. Although the dataset is called UC Merced Land Use, it includes also land cover classes such as beach, forest and river. To explain the difference, land cover refers to the physical land type such as bare soil, water or urban infrastructure, whereas land use indicates how people use land e.g. agricultural and residential purposes. Figure 4.2 shows example images from UC Merced Land Use dataset. The archive is originally constructed as a single label archive. For this reason, it is not possible to use it in the experiments. Fortunately, the authors of [7] created a multi-label version of UC Merced Land Use where each image is labeled with one to seven labels based on visual inspection. This decreases the number of distinctive classes in the dataset from 21 to 17, while keeping the number of total images at a constant of 2100. The number of training, validation and test samples in multi-label UC Merced Land Use dataset can be found in Table 4.2.

Table 4.1: Number of samples per class in training, validation and test sets of the Ireland subset of BigEarthNet using 19 class-nomenclature.

Class	Train	Val	Test
Urban Fabric	818	369	371
Industrial Commercial Units	193	79	69
Arable Lands	2860	1404	1397
Permanent Crops	6	0	0
Pastures	5723	2725	2739
Complex Cultivation Patterns	510	240	226
Land Principally Occupied by Agriculture, with Significant Areas of Natural Vegetation	896	420	414
Agro-foresty Areas	0	0	0
Broad-leaved Forest	410	205	199
Coniferous Forest	1156	545	524
Mixed Forest	588	273	296
Natural Grassland and Sparsely Vegetated Areas	124	48	55
Moors, Heathland and Sclerophyllous Vegetation	467	217	219
Transitional Woodland, Shrub	708	345	369
Beaches, Dunes, Sands	63	7	15
Inland Wetlands	811	400	415
Coastal Wetlands	37	7	12
Inland Waters	165	64	75
Marine Waters	2087	894	900



(a) Pastures, Intertidal flats, Sea and ocean

(b) Peatbogs

(c) Sea and ocean

(d) Pastures, Moors and heathland, Sea and ocean

(e) Pastures

(f) Pastures, Mixed forest

Figure 4.1: Example images with labels from BigEarthNet

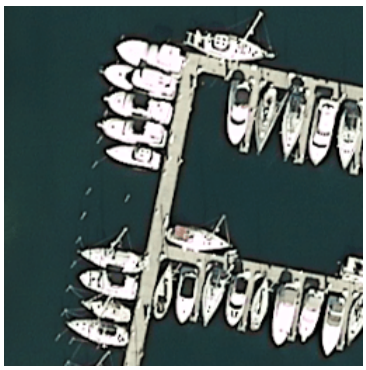
Table 4.2: Number of samples per class in training, validation and test sets of multi-label UC Merced Land Use dataset.

Class	Train	Val	Test
Airplane	70	15	15
Bare Soil	506	103	109
Buildings	482	102	107
Cars	631	125	130
Chaparral	77	21	17
Court	73	16	16
Dock	70	15	15
Field	73	15	15
Grass	683	145	147
Mobile home	70	17	15
Pavement	917	189	194
Sand	210	43	41
Sea	70	15	15
Ship	70	16	16
Tanks	70	15	15
Trees	702	155	152
Water	142	31	30

(a) Airplane, Grass, Pavement

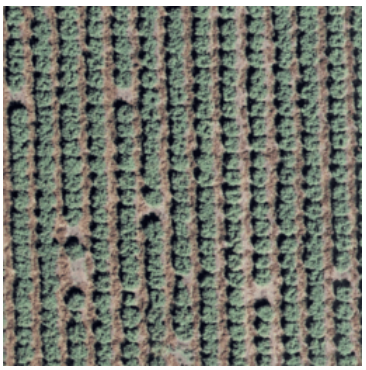
(b) Bare soil, Buildings, Grass

(c) Cars, Grass, Pavement



(d) Dock, Ship, Water

(e) Sand, Sea



(f) Trees

Figure 4.2: Example images with labels from multi-label UC Merced Land Use dataset

4.2 Network Architecture

We employ ResNet50V2[25] and a shallow convolutional neural network architecture called SCNN, following the setup in [57] to train and test our model. SCNN consists of three convolutional layers. The first two layers have 32 filters with 5×5 filter size. The last layer has 64 filters with 3×3 filter size. All three layers have zero paddings, and they are followed by the activation function Rectified Linear Unit (ReLU). After each convolutional layer, a maximum pooling layer with 2×2 window size and 2×2 strides is inserted. A fully connected layer and a classification layer are added to the end of the network to output the network predictions. A shallow network, such as SCNN, is chosen to show the effectiveness of the proposed method even in shallow structures. A more popular and well established network architecture, such as ResNet50V2, is also incorporated into the experiments to create accountability and comparability.

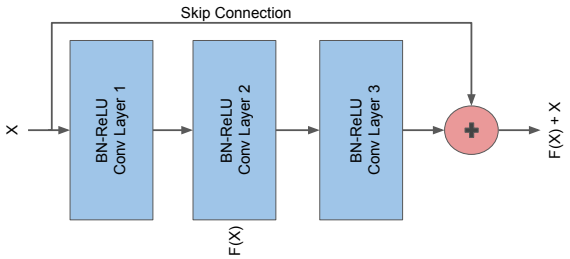


Figure 4.3: Residual Block

Table 4.3: Structure of the Network

Layer	Input Image
1	5×5 conv, 32 ReLU
	2×2 max-pool, 2×2 stride
2	5×5 conv, 32 ReLU
	2×2 max-pool, 2×2 stride
3	3×3 conv, 64 ReLU
	2×2 max-pool, 2×2 stride
4	Fully Connected Layer

ResNet[24] was originally introduced to create deeper networks addressing the notorious vanishing gradients problem. Calculated gradients are back-propagated through the network in conventional DNNs. However, if the DNN is too deep, gradients may vanish along the way and cannot reach to the initial layers. This problem causes neural networks to saturate early and sometimes diverge from the optimal accuracy. In order to avoid that, the authors of [24] suggested a new network architecture that consists of many residual blocks. A residual block is a stack of convolutional blocks that is embellished with a skip connection. A skip connection is a structure, where the input of the residual block is transmitted to the output function bypassing the in-between layers. The skip connection avoids gradients being zero by copying the input of the current residual block to the following residual block. This allows building very deep neural networks without worrying about the vanishing gradient problem. ResnetV2[25] is an improvement on ResNet that adds additonal non-linearities to the network. There are different versions of ResNet each having different amounts of layers. We choose ResNet50 that has 50 layers.

4.3 Experimental Setup

We conduct extensive experiments to evaluate our method, using a comparative analysis on the base model and the proposed method. The base model consists of two deep neural networks that run parallel without exchanging loss information or using any other component of the proposed method. Different rates of label noise are applied to both of the models. Then, a quantitative comparison between the accuracies of them is documented. In this section, first, the tuning of the hyperparameters is described, and the optimal values for them are revealed. Next, the considered label noise approaches for the experiments are explained. Finally, the evaluation metrics used in the comparative analysis are described. The results of the experiments can be found in Chapter 5.

4.3.1 Hyperparameter Optimization of the Proposed Method

Optimization of the hyperparameters of a deep neural network is crucial for its performance. Hyperparameters control the learning process. They cannot be learned via an optimizer during back-propagation, in contrast to regular parameters. They often affect the speed and quality of the training process. Thus, the accuracy of the network can suffer quite badly without inferring the right values for the hyperparameters. Hyperparameter optimization aims to find the tuple of parameters that minimizes the predefined loss function on the given input. However, it is not an easy task to tune these parameters. It often requires an exhaustive search that grows exponentially with the number of unknown parameters and a lot of time and computation power. The optimal hyperparameters for the proposed method are found after extensive experiments.

The usual hyperparameters for a deep neural network include batch size, number of epochs and learning rate of optimizer. Prediction threshold can be added to the list in the multi-label classification setting. Our choice of optimizer for the experiments is Adam[31]. Adam is an extension to stochastic gradient descent algorithm, where the estimate of the error that is used to update the network weights is calculated on a mini-batch of samples. However, in contrast to SGD, Adam calculates individual learning rates for different network parameters, thus achieving better results. The learning rate for the optimizer is set to 0.001 to avoid a suboptimal solution while keeping a high convergence rate. The chosen batch size depends on the dataset that is trained. Batch size is the total number of training examples that are used to calculate gradient in a single iteration. DNNs converge quicker with smaller batches at the cost of noise in the training process[45][46]. Bigger batches make networks converge slowly, yet give a more accurate estimation of the error gradient. We fix the batch size for BigEarthNet to 256. On the other hand, 32 is determined as batch size for multi-label UC Merced Land Use archive because UC Merced Land Use is a resolution wise higher quality dataset with 256×256 pixels compared to BigEarthNet, which includes 120×120 pixels. Therefore, it takes more memory. We reduce the training time as well as converge safer by avoiding noise with large batch sizes. We report the results of training after 20, 30 and 100 epochs depending on the dataset, model and experiment type. We choose the prediction threshold for our model as 0.5, which means that the normalized predictions under 0.5 are evaluated as 0 and the ones over the threshold are evaluated as 1.

In addition to the parameters mentioned above, there are method-specific hyperparameters that need tuning. Finding the right values for them is essential for the proposed method's success. However, our model requires tuning of copious hyperparameters. This complicates the process,

as finding the right combination of parameters is extremely challenging. Thus, after we set the batch size, epoch, learning rate, and threshold, we focus on finding the right tuple of method-specific hyperparameters for the proposed method, and then conduct extensive practical as well as theoretical experiments. Below, we inspect the hyperparameters successively and explain the reasoning behind the choices of optimal values.

Within the proposed approach's swap module, the networks swap the calculated low loss sample information with each other. According to this information, each network chooses a certain amount of low loss samples of the opposite network and updates its weights using them. The goal of the process is to remove the noisy samples from back-propagation. Without knowing the exact noise amount and distribution in the labels, optimizing this parameter is very hard. Excluding too many samples from back-propagation degrades the accuracy of the predictions. Networks starve and cannot learn diverse features using enough samples. On the other hand, excluding too few samples also affects the accuracy adversely because the network overfits to the noise present in the data. According to our observations, exchanging 75% of the low loss sample information at each iteration is the optimal solution to this tradeoff.

The discrepancy measure is a deciding factor in the success of the proposed method. The network's ability to learn diverse features from the data depends on the discrepancy measurement's quality and accuracy. We considered three discrepancy measurements to apply in our experiments: Wasserstein, Jensen-Shannon, and MMD. Although in theory, Wasserstein metric is a plausible choice, it is eliminated from the options because of its large time complexity. Comparing MMD and Jensen-Shannon through a set of tests reveals the superior performance of MMD. The fact that MMD is more capable of assessing the topology of feature spaces than Jensen Shannon also supports our findings. Thus, MMD is chosen as the discrepancy measurement to be used in the experiments.

MMD takes advantage of the kernel trick to analyze the relations in the feature space. In all our experiments, we employ the RBF kernel as our similarity function in the following form:

$$k(s_1, s_2) = \exp\left(-\frac{\|s_1 - s_2\|^2}{\sigma}\right) . \tag{4.1}$$

The optimal value of the σ in the RBF kernel is determined as 10.000. We observed that low sigma values were ineffective for measuring the distance between the logits of the two networks used in the proposed method. Using the RBF kernel, the discrepancy module first maximizes the diversity to learn diverse features, and then minimizes it to learn the same class distributions. The module achieves this by adding the respective losses, namely L_C and L_D, to the final loss terms of the networks, weighting them by λ_2 and λ_3 as follows:

$$Loss_f^b = \frac{\sum_{i=1}^{R} L_f(x_i)}{R} + \lambda_2 \cdot L_C - \lambda_3 \cdot L_D \tag{4.2}$$

$$Loss_g^b = \frac{\sum_{i=1}^{R} L_g(x_i)}{R} + \lambda_2 \cdot L_C - \lambda_3 \cdot L_D \ , \tag{4.3}$$

The value of λ_2 is set to 0.25, and the value of λ_3 is set to 0.5. The diversity component of the module is placed approximately within the second quarter of the convolutional layers. Forcing networks to diverge in an early stage and converging them thereafter teach the networks distinct

feature sets while keeping the predictions consistent.

An additional mechanism along with the BCE loss is necessary to identify the potential noisy samples. For this purpose, a ranking loss for every sample is calculated and added to the BCE loss by a factor that is set to 0.2 as the result of our experiments. There are also parameters α and β that control the effects of two different noise types on the calculation of the ranking loss. A group lasso module with a bigger α concentrates more on finding the missing class labels, whereas a bigger β gives a heavier weight to detecting the extra class label assignments. An extra class label assignment is more harmful than a missing class label in our setting, where every sample is annotated with the minority of the present classes. Thus, we set α to 0.5 and β to 1.0.

As mentioned in Chapter 3, it is crucial to choose the right time to start flipping potential noisy labels because the networks are not stable in a very early stage of the training, and thus the predictions are not accurate. Therefore, the early initiation of the flipping module may cause erroneous flips. To prevent this, we start the flipping process after 90% of epochs is done. Furthermore, the flipping module flips labels when the two networks agree that the label is noisy. As the networks learn from each other and get more stable, the number of agreed labels may increase. However, the networks may agree on the labels of hard classes instead of noisy ones, and flipping too much of them would decrease the performance of the model. The flipping module flips only the 5% of the agreed classes every iteration after it is initiated to avoid this problem.

The hyperparameter tuning and the initial tests were conducted on the Technical University of Berlin High-Performance Cluster where an NVIDIA Tesla P-100 GPU with 16 GB of RAM was used. However, the model training and experiments were conducted on a GPU with a bigger 32 GB RAM, called Tesla V100, on RSiM cluster.

The code[1] for this work is written in Python 3.7.6 using Tensorflow 2.2.0. We additionally used an open-source library called Noisifier to introduce label noise to the datasets during the training.

4.3.2 Considered Label Noise Approaches

To verify the effectiveness of the proposed method, we add synthetic noise to the labels of BigEarthNet and multi-label UC Merced Land Use dataset, using two different noise approaches that are designed for the experiments. Both of the techniques get a batch of multi hot binary encoded labels as input. The first technique is called Random Noise per Sample (RNS). It chooses random samples from each mini-batch by a predefined percentage that is called sample rate. Then, it chooses random labels of the chosen samples by another percentage, namely class rate, and flips them. RNS adds noise to certain samples while disregarding the others. The second technique is called Mix Label Noise (MLN). It randomly chooses a percentage of all the negative labels in the mini-batch and flips them to positive labels introducing extra class noise to the dataset. By the same percentage that is called sample rate, the method randomly chooses positive labels and flips them to negative labels adding missing class noise. MLN introduces different types of label noise proportional to the distribution of classes in the batch. If most entries in the batch are 0, the technique adds more extra class noise to the batch and vice versa. A visual explanation of the considered label noise approaches is displayed in Figure 4.4.

[1]The code for this work is available at https://gitlab.tubit.tu-berlin.de/rsim/CCML

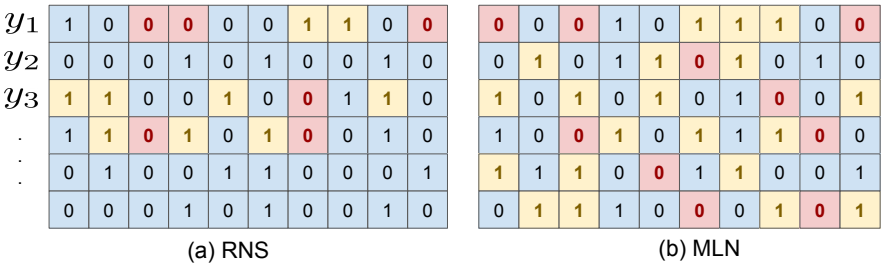


Figure 4.4: The two label noise approaches used in the experiments. (a) shows Random Noise per Sample with a sample rate of 0.5 and a class rate of 0.5. Three samples out of six are chosen, and half of their labels are randomly noised. (b) shows Mix Label Noise with a sample rate of 0.5. Over the whole batch, half of the positive labels and half of the negative labels are chosen and noised. The yellow boxes represent the negative ground truth labels that are flipped, and the red boxes represent the positive ground truth labels that are flipped.

In the experiments, two different settings are used for the above mentioned label noise approaches. In the first setting, the random selection of the samples and labels to be noised is not-deterministic (it changes every epoch). At every epoch, the noise distribution in the labels change. We do that to show the robustness of the proposed method even for complicated and dynamically changing noise in the labels. We refer to this setting in Chapter 5 as not-deterministic noise. In the second setting, the same set of labels are randomly selected every epoch, and label noise is added to them. This approach is referred to deterministic noise in Chapter 5.

4.3.3 Evaluation Metrics

The performance of the proposed method is evaluated through an empirical study using various evaluation metrics. In this subsection, the evaluation metrics used in the experiments are described.

To test a classifier's ability to generalize to real world data, the classifier runs on unseen data and attempts to predict the correct label set for each sample. It is imperative to utilize the proper evaluation metrics to assess the accuracy of its predictions. Failing to choose the right metrics may be misleading about the classifier's performance, especially in multi-label setting, where the prediction accuracies of multiple classes per sample are involved in the evaluation. In single label classification, every sample is matched with only one label. On the other hand, multi-label classifiers predict a set of labels for each instance from the dataset, and therefore, the prediction can be partially correct, fully correct or fully incorrect. This complicates the evaluation process of multi-label classifiers, requiring more advanced methods to capture the true performance of the classifier.

In a classification scenario, the predictions can be categorized into four groups: True Negative (TN), False Positive (FP), False Negative (FN) and True Positive (TP). They are often described in a 2×2 grid called confusion matrix as shown in Table 4.4, where each cell contains the number

of predictions that fall into its group.

Table 4.4: Confusion Matrix

	Negative Prediciton	Positive Prediction
Ground Truth Negative Label	True Negative (TN)	False Positive (FP)
Ground Truth Positive Label	False Negative (FN)	True Positive (TP)

The first column of the confusion matrix represents the negative predictions while the second column represents the positive predictions. Ground truth negative labels are located in the first row, and the ground truth positive labels are on the second row. True predictions are represented by the upper left cell and the lower right cell. The confusion matrix allows a detailed analysis of the performance of the classifier. All the evaluation metrics used in this work are derived from the confusion matrix.

A common choice of evaluation metric of a classifier's performance is the classification accuracy. Classification accuracy is the number of correct predictions divided by the total number of predictions.

$$Classification\ Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad . \tag{4.4}$$

Although classification accuracy is simple and intuitive, it yields misleading results when dataset is class-imbalanced. The classifier may always predict the over-represented classes and still get very good classification accuracy scores because the under-represented classes are very scarce in number, and predicting them wrong does not influence the overall classification accuracy significantly. This obfuscates the real performance of the classifier leading to thinking that the classifier is successful, when in fact, it is not. Moreover, in multi-label classification, usually multi hot vector notation is adopted, where generally most entries are zero representing the classes that are not annotated for the instance. With the ground truth labels and predictions as in Figure 4.5, the classification accuracy will be 60%, even though the predictions are completely wrong.

Figure 4.5: Label prediction example

This example illustrates that classification accuracy is not enough to assess the performance of a multi-label classifier and may be misleading in certain scenarios. Therefore, we use more advanced evaluation metrics to test the proposed method.

The first evaluation metric we employ is precision, which gives a measure of what proportion of positive predictions was actually correct. Formally, it is

$$Precision = \frac{TP}{TP+FP} \quad . \tag{4.5}$$

The second evaluation metric we utilize is recall, which answers the following question: What proportion of actual positives was identified correctly? Formally, it is defined as

$$Recall = \frac{TP}{TP + FN} \quad . \tag{4.6}$$

Precision and recall are good measures for multi-label classifiers because they together asses the relevancy of the positive predictions. Higher precision means that most of the positive predictions were in fact correct, and higher recall implies that most of the positive ground truth labels were predicted correctly. To evaluate the performance of a classifier accurately, precision and recall must be examined together. Ideally, both of them must be as high as possible and proportional. However, they often compete with each other. Improving precision typically reduces recall and vice versa. Thus, we choose F1 as our third evaluation metric, which combines precision and recall by calculating the harmonic mean of them:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad . \tag{4.7}$$

F1 is usually the most trustworthy evaluation metric in classification tasks.

There are different ways of calculating the above mentioned metrics. One way is computing the metric scores independently for each class and then taking the averages. This approach is called macro averaging, and it treats all classes equally. An alternative approach is called micro averaging, where the metric scores are calculated over all the labels and then averaged. We provide the micro averaged results in Chapter 5. We also calculate the class-based performances using the F1 scores to reveal the effects of the proposed method on under-represented and over-represented classes.

5 Experimental Results

Following the settings mentioned in the previous chapter, several experiments are conducted to assess the performance of the proposed method. The results and their analysis are presented in this chapter under two sections. The first section analyzes the results obtained using different label sets for the Ireland subset of BigEarthNet including the 43 class-nomenclature, the 19 class-nomenclature and the deducted 12 class-nomenclature. The second section includes the results and analysis of multi-label UC Merced Land Use dataset. The experimental results are reported in a comparative style where six different noise rates (0%, 10%, 20%, 30%, 40%, 50%) were tested for each setting using the base model as well as the proposed method. Note that the obtained recall, precision, and F1 scores are reported in percentages. The better score in each setting is highlighted, comparing the performances of the base model and the proposed method. The most representative experiments in terms of the analysis of the proposed method are reported. Since the proposed method includes two networks that run simultaneously, we get two different sets of scores at the end of each experiment for both of the networks. The network with the best F1 validation score is chosen and tested on the test sets of the respective datasets. In this chapter, the class-based F1 scores are also reported as comparative plots to give a better intuition about the performance and behaviour of the proposed method. The results provide meaningful insights into the proposed method, and reveal remarkable characteristics of the used datasets and considered label noise approaches.

5.1 Results on the Ireland Subset of BigEarthNet

The test results for the base model and the proposed method on the Ireland subset of BigEarthNet using the 43 class-nomenclature are summarized in Table 5.1. SCNN is chosen as the network structure and run for twenty epochs. A pattern that is observable under this setting is that the precision scores of the base model as well as the proposed method are significantly higher than the recall scores. Moreover, this pattern continues in the following experiments, which reveals that the networks, whether using the proposed method or not, are inclined to do negative predictions. This can be explained with the overwhelming majority of the negative ground truth labels compared to the positive ones. Since 95% of the BigEarthNet images in the 43 class-nomenclature are labeled by at most 5 classes, the networks learn more negative labels than they do positives. This causes a bias in the training in favour of predicting negative labels.

The proposed method obtains slightly better F1 scores than the base model using 43 class-nomenclature for all rates of label noise except 50%. The reason for this increase is the improvement on the recall score of the proposed method compared to the base model. Since precision and recall are often in tension, the precision scores of the proposed method in return decreases in this scenario. This displays the proposed method's ability to predict more positive ground truth labels correctly. The performance of both models do not decrease significantly with the increased noise rates. This trend is also observable, with few exceptions, throughout the rest of the experiments. This is because the number of samples for the over-represented classes are sufficient for

Table 5.1: Test results on the Ireland subset of BigEarthnet using 43 class-nomenclature for 20 epochs on SCNN with not-deterministic RNS

	Precision		Recall		F1	
	Base	CCML	Base	CCML	Base	CCML
0%	76.4	**82.4**	66.2	**66.7**	70.9	**73.6**
10%	**88.5**	85.0	64.2	**66.0**	**74.3**	**74.3**
20%	**84.1**	81.8	65.2	**67.0**	73.4	**73.6**
30%	**93.2**	78.7	55.9	**67.3**	69.8	**72.6**
40%	**85.2**	81.7	56.8	**60.6**	68.1	**69.5**
50%	**83.4**	79.6	56.0	**56.9**	**66.9**	66.3

the networks to predict them correctly, despite the present label noise. Since the over-represented classes are high in number, their influence on the micro averaged scores are proportionally high. On the other hand, the under-represented classes do not have much effect on the micro averaged scores because they are low in number.

In contrast to the above mentioned results, the base model outperforms the proposed method in Table 5.2. The base model obtains better results with ResNet50 architecture using the 19 class-nomenclature except for the 50% label noise scenario. This can be explained by the relative robustness of ResNet50 (which uses regularization techniques) compared to SCNN to label noise, and the improved label set. Moreover, the base module uses the whole training set, which gives it the opportunity to learn from diverse samples. On the other hand, the proposed method excludes the 25% of the training samples from back-propagation hoping to obtain improved accuracy. However, excluding samples deteriorates the model's ability to generalize to the test set failing to detect the noisy samples in this setting. With a high rate of noise (50%), where half of the samples have noisy labels, the proposed method performs about 25% (in terms of F1 score) better.

Table 5.2: Test results on the Ireland subset of BigEarthnet using 19 class-nomenclature for 30 epochs on ResNet50 with deterministic RNS

	Precision		Recall		F1	
	Base	CCML	Base	CCML	Base	CCML
0%	84.2	**90.1**	**65.8**	61.3	**73.8**	72.9
10%	87.8	**88.1**	**64.8**	59.8	**74.6**	71.2
20%	90.2	**92.7**	**64.7**	59.4	**75.3**	72.4
30%	85.6	**91.8**	**67.1**	55.3	**75.1**	68.9
40%	**92.2**	89.3	59.7	**60.3**	**72.4**	72.0
50%	20.4	**65.0**	**74.2**	52.0	31.9	**57.6**

When the class-based validation scores at the end of the 30th epoch for the 19 class-nomenclature are examined, it is observed that the scores of the under-represented classes include zeros, which indicates the absence of true positive predictions. Also NaN (Not a Number) values are observed in these classes' precision scores, which signals the absence of positive predictions all together. This is partly due to the inclination of the networks to do mostly negative predictions as explained above. However, the biggest reason for this situation is the class-imbalance problem. In case of

a class-imbalance problem, the model predicts the over-represented class too often, while ignoring the under-represented classes completely. To verify this argument, the class-based validation scores using the same setting for the deducted 12 class-nomenclature are checked, and neither zero nor NaN values are detected.

In a class-imbalanced dataset, since there are more samples that belong to the over-represented classes in the training set, they are learned better, and their loss values decrease quickly. Contrarily, the under-represented classes' loss values do not decrease in a similar fashion. In this case, the proposed method may lead to worse results than the base model do, because it may exclude the samples with low loss values from the back-propagation considering them noisy, when in fact they are just under-represented. This can evolve into an ongoing trend, where the model cannot ever learn the under-represented classes due to their constant exclusion. Thus, utilizing the proposed method using a class-balanced dataset would give the best results.

Comparing Table 5.3 with Table 5.4 show that deterministic RNS deteriorates the model's ability to generalize to the test data more than not-deterministic RNS does, because the label noise created by not-deterministic RNS changes randomly every epoch. This means that the noise introduced to the labels are not persistent, and the labels are free of noise for most of the epochs. The model thus finds the opportunity to learn each label when the label does not include noise. This compensates the misleading effect created by label noise, and the model learns from samples in spite of label noise. On the other hand, deterministic RNS creates samples with constant label noise, causing the model to overfit to the invariable label noise.

Table 5.3: Test results on the Ireland subset of BigEarthnet using 12 class-nomenclature for 100 epochs on ResNet50 with deterministic RNS

	Precision		Recall		F1	
	Base	CCML	Base	CCML	Base	CCML
0%	81.1	**87.7**	**73.1**	66.6	**76.9**	75.6
10%	84.1	**91.6**	**72.3**	69.4	77.7	**78.9**
20%	81.0	**89.8**	**71.4**	70.4	75.8	**78.9**
30%	83.6	**90.1**	68.9	**69.9**	75.5	**78.7**
40%	65.0	**89.5**	**73.1**	68.0	68.8	**77.3**
50%	33.8	**55.8**	**69.4**	67.3	45.2	**60.8**

As mentioned in Chapter 3, the proposed method's main idea is that it excludes the noisy samples from back-propagation. As can be observed in Table 5.3, using deterministic noise, the proposed method successfully detects the noisy samples and excludes them from back-propagation. The proposed method thereby outperforms the base model by a significant margin, especially with higher noise rates such as 40% and 50%. While the F1 scores of the base model decreases substantially with the increasing noise rate as observed in Table 5.3, the proposed method's accuracy stays stable, and the proposed method obtains in average 5% better F1 scores.
Table 5.4 reveals that the proposed method loses its superiority with high rates of noise using not deterministic label noise, because not-deterministic noise does not affect the performance of the base model adversely.

The test results on the Ireland subset of BigEearthNet with the 12 class-nomenclature using de-

Table 5.4: Test results on the Ireland subset of BigEarthnet using 12 class-nomenclature for 100 epochs on ResNet50 with not-deterministic RNS

	Precision		Recall		F1	
	Base	CCML	Base	CCML	Base	CCML
0%	82.5	**87.9**	**73.1**	69.8	77.5	**77.7**
10%	83.6	**86.3**	68.9	**69.9**	75.5	**77.2**
20%	88.0	**89.5**	67.6	**69.9**	76.4	**78.5**
30%	**91.9**	87.0	65.9	**70.0**	76.7	**77.5**
40%	**88.9**	84.8	**70.0**	69.2	**78.3**	76.2
50%	**91.8**	76.1	**66.6**	55.3	**77.2**	63.9

terministic MLN can be observed in Table 5.5. Despite RNS, where label noise is added sample wise, MLN adds noise to the whole training labels randomly. This complicates the noisy sample exclusion process for the proposed method, as most samples statistically have some degree of noise in their labels, especially with higher noise rates, which hurts the F1 scores of the proposed method.

Table 5.5: Test results on the Ireland subset of BigEarthnet using 12 class-nomenclature for 100 epochs on ResNet50 with deterministic MLN

	Precision		Recall		F1	
	Base	CCML	Base	CCML	Base	CCML
0%	84.6	**87.1**	70.6	**70.8**	77.0	**78.1**
10%	82.0	**91.5**	**69.0**	68.5	74.8	**78.3**
20%	**92.5**	91.1	**69.1**	68.3	**79.0**	78.0
30%	**92.0**	88.1	67.0	**67.2**	**77.5**	76.2
40%	**94.3**	74.9	58.8	**66.2**	**72.3**	70.2
50%	**23.8**	07.4	**79.2**	17.8	**36.6**	10.5

Inspecting the class-based F1 scores in Figure 5.1 reveals that both the base model and the proposed method learn the over-represented classes better than they learn the under-represented classes in the given setting. The classes with the highest number of training samples — Arable Land, Pastures and Marine Waters — have the highest F1 scores over all the noise rates. Even high rates of noise, such as 40% and 50%, do not hurt the F1 scores of these classes substantially. This is because they still have a lot of training samples to be learned from, despite high rates of noise. On the other hand, the most under-represented class in the subset, Broad-leaved forest, with 410 training samples receives a low F1 score averaging 15% over all noise rates.

The proposed method outperforms the base model in the majority of classes, especially with high noise rates while obtaining competitive results against the base model in some classes such as Mixed Forest and Transitional Woodland, Shrub. Both models perform poorly in Complex Cultivation Patterns with an average of 0.06 F1 score. The base model almost always performs better than the proposed method with 0% label noise. However, with increased noise rate, the proposed method keeps the F1 scores relatively high.

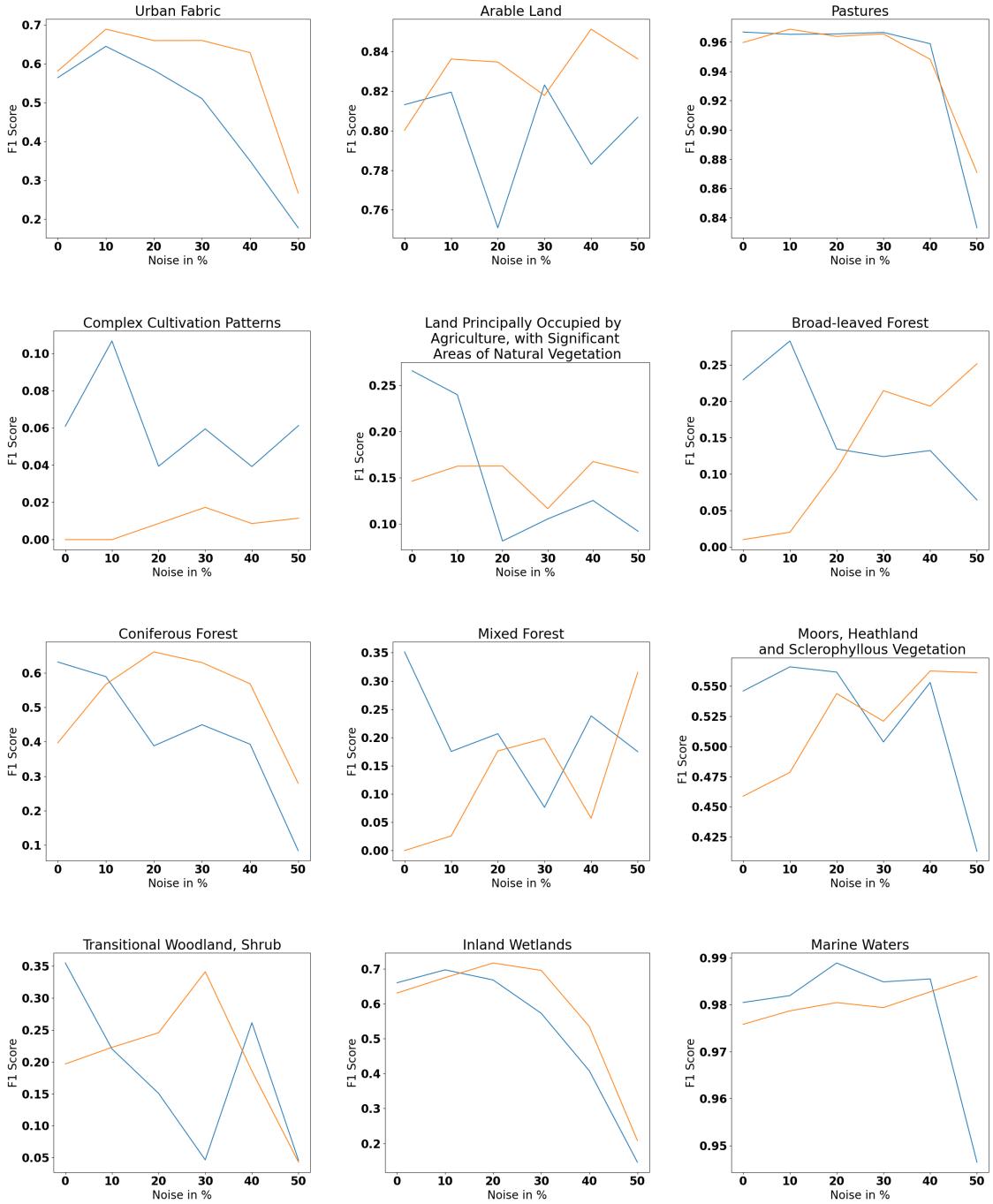


Figure 5.1: Class-based test results of the proposed method on the Ireland subset of BigEarthnet using 12 class-nomenclature for 100 epochs on ResNet50 with deterministic RNS. The blue line represents the base model, and the orange line represents the proposed method.

5.2 Results on multi-label UC Merced Land Use dataset

In this section, we provide the test results for multi-label UC Merced Land Use dataset using two different deterministic label noise approaches. Table 5.6 showcases the results using RNS, and Table 5.7 illustrate the results using MLN. Both experiments are conducted using ResNet50 architecture running for 100 epochs.

Similar to the test results in Table 5.3, deterministic RNS deteriorates the learning process of the base model severely in Table 5.6. On the other hand, the proposed method keeps the learning process relatively stable and achieves considerably higher F1 scores with 40% and 50% noise rates. It is evident that the proposed method achieves a certain degree of robustness and stability against high rates of label noise.

Table 5.6: Test results on multi-label UC Merced Land Use dataset for 100 epochs on ResNet50 with deterministic RNS

	Precision		Recall		F1	
	Base	CCML	Base	CCML	Base	CCML
0%	**79.4**	67.8	**68.9**	66.5	**73.7**	67.1
10%	**79.8**	69.8	61.7	**68.4**	**69.5**	69.0
20%	69.9	**71.9**	62.3	**63.3**	65.7	**67.3**
30%	**75.8**	73.1	**71.4**	65.5	**73.4**	69.0
40%	58.7	**70.2**	61.4	**67.3**	59.7	**68.6**
50%	32.3	**54.6**	44.9	**67.5**	37.5	**60.3**

Table 5.7: Test results on multi-label UC Merced Land Use dataset for 100 epochs on ResNet50 with deterministic MLN

	Precision		Recall		F1	
	Base	CCML	Base	CCML	Base	CCML
0%	68.1	**69.7**	54.8	**64.3**	60.5	**66.7**
10%	67.7	**78.0**	36.3	**67.3**	47.1	**72.1**
20%	73.4	**73.6**	67.4	**69.8**	70.1	**71.5**
30%	52.1	**70.1**	35.9	**55.9**	42.2	**62.1**
40%	**69.8**	56.4	54.6	**66.5**	**61.1**	61.0
50%	**14.8**	14.4	**43.2**	35.0	**22.0**	20.4

A comparison between Table 5.3 and Table 5.6 reveals that multi-label UC Merced Land Use dataset achieves lower F1 scores compared to the Ireland subset of BigEarthNet using the same training setting. The reason for that is multi-label UC Merced Land Use dataset includes 1470 training samples while the Ireland subset of BigEarthNet includes 8197 training samples. The large amount of training samples makes it easier to learn the underlying class distribution of the Ireland subset.

The results in Table 5.7 demonstrate significant fluctuation in the performance of the base model over different noise rates. It is observed that MLN is more harmful than RNS to the consistency of the scores obtained on multi-label UC Merced Land Use dataset, which is again explained by

the low number of training samples in the dataset. The proposed method successfully stabilizes the accuracy of the predictions and outperforms the base model significantly.

For the analysis of the class-based F1 scores obtained from multi-label UC Merced Land Use dataset, three representative classes are chosen and illustrated in Figure 5.2. The class-based scores are noted following the experimental setup used in Table 5.6. The chosen Airplane class is the most under-represented class in multi-label UC Merced Land Use dataset with 70 training samples. Pavement is the most over-represented class in the dataset with 917 training samples. Buildings has 482 training samples, which corresponds to an average number of samples in the dataset. The performance of the base model is very irregular as seen in Figure 5.2. The F1 scores of the base model for the class Airplane include unreasonable fluctuations, which negatively affect the micro-averaged F1 scores in Table 5.6. On the other hand, the proposed method's scores are stable, even though they do not smoothly increase. Similar to Figure 5.1, the F1 scores of the most over-represented class in the dataset are the highest among the representative classes. Nonetheless, the proposed method keeps the F1 scores of the most under-represented class considerably high.

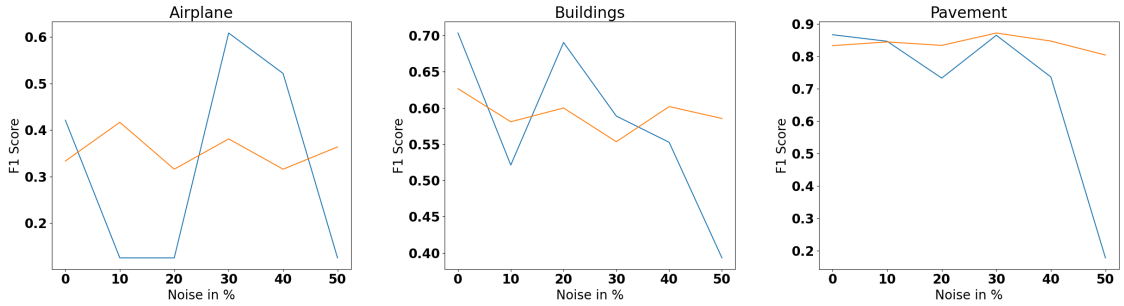


Figure 5.2: Chosen class-based test results of the proposed method on multi-label UC Merced Land Use dataset for 100 epochs on ResNet50 with deterministic RNS. The blue line represents the base model, and the orange line represents the proposed method.

6 Conclusion and Discussion

In this thesis, we propose a novel method called Consensual Collaborative Multi-label Learning to overcome the negative effects of multi-label noise in the context of scene classification of remote sensing images. The proposed method aims to detect noisy samples and exclude them from back-propagation to reduce the misleading effects of noisy labels on training. In addition, the proposed method detects the individual noisy labels in the training set and fixes them following a flipping policy. The performance of the proposed method is tested in diverse settings using two remote sensing datasets with different label noise approaches. The results of the experiments validate the efficacy of the proposed method in certain settings where deterministic label noise is applied to relatively class-balanced datasets, such as the 12 class-nomenclature of the Ireland subset of BigEarthNet.

To the best of our knowledge, this thesis is the first work in the literature that attempts to ameliorate the harmful effects of all types of multi-label noise including missing class label, extra class label assignment and wrong class label assignment. We believe that this work exposes some important facts about the classification of multi-label images in the presence of label noise, and we hope that our work sheds some light onto the field.

With the limited computation power and time that were available during the experimental phase, the model specific hyperparameters are optimized as good as possible. However, we believe that the performance of the proposed method can be vastly improved with better tuned model specific hyperparameters. The inclination of the model to predict mostly negative classes can be fixed using a lower prediction threshold or more appropriate hyperparameters. Furthermore, the proposed method works best using class-balanced multi-label datasets. For future work, we plan to test our method on the whole BigEarthNet, which is subject to significant class-imbalance. It is often hard to obtain a perfectly class-balanced dataset in the multi-label setting. Nonetheless, we consider approximating the number of samples for each class using data augmentation techniques to overcome this problem. Another considered solution to this problem is introducing a class-weighted loss function to obtain more balanced class-based accuracy. We also plan to extend this work to the computer vision field using appropriate large scale multi-label datasets.

The flipping module of the proposed method is initiated after a certain number of epochs within the algorithm. As the network predictions get more and more accurate during the training, their quality increases, and the probability of flipping a clean label erroneously thus decreases. However, by doing that, the chance to fix more noisy samples from the initial epochs disappears. To overcome this limitation, pre-trained models or pre-set weights can be used as a starting point. This allows class flipping to be started from the beginning of the training process. Even though, we do not investigate this option, it is worth mentioning, and it may lead to better results.

In the future, the considered label noise approaches will be extended. We believe that using different label noise structures can give more meaningful results and better insights to the proposed

method. The experimental results show that the base model learns the over-represented classes very well in spite of label noise for most of the settings. We aim to deteriorate the network's ability to learn the over-represented classes in the presence of noise either by undersampling these classes or using class-based noise, and then repeating the tests. Doing that, we believe that the performance of the proposed method can be observed better.

Bibliography

[1] Görkem Algan and Ilkay Ulusoy. "Image classification with deep learning in the presence of noisy labels: A survey". In: *arXiv preprint arXiv:1912.05170* (2019).

[2] Mauricio A Alvarez, Lorenzo Rosasco, and Neil D Lawrence. "Kernels for vector-valued functions: A review". In: *arXiv preprint arXiv:1106.6251* (2011).

[3] Brigitte Bigi. "Using Kullback-Leibler distance for text categorization". In: *European Conference on Information Retrieval*. Springer. 2003, pp. 305–319.

[4] Avrim Blum and Tom Mitchell. "Combining labeled and unlabeled data with co-training". In: *Proceedings of the eleventh annual conference on Computational learning theory*. 1998, pp. 92–100.

[5] Carla E Brodley and Mark A Friedl. "Identifying mislabeled training data". In: *Journal of artificial intelligence research* 11 (1999), pp. 131–167.

[6] Serhat Selcuk Bucak, Rong Jin, and Anil K Jain. "Multi-label learning with incomplete class assignments". In: *CVPR 2011*. IEEE. 2011, pp. 2801–2808.

[7] B. Chaudhuri et al. "Multilabel Remote Sensing Image Retrieval Using a Semisupervised Graph-Theoretic Method". In: *IEEE Transactions on Geoscience and Remote Sensing* 56.2 (2018), pp. 1144–1158. DOI: 10.1109/TGRS.2017.2760909.

[8] Minmin Chen, Kilian Q Weinberger, and John Blitzer. "Co-training for domain adaptation". In: *Advances in neural information processing systems*. 2011, pp. 2456–2464.

[9] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[10] Alexander Philip Dawid and Allan M Skene. "Maximum likelihood estimation of observer error-rates using the EM algorithm". In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 28.1 (1979), pp. 20–28.

[11] Mostafa Dehghani et al. "Fidelity-weighted learning". In: *arXiv preprint arXiv:1711.02799* (2017).

[12] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[13] Thibaut Durand, Nazanin Mehrasa, and Greg Mori. "Learning a deep convnet for multi-label classification with partial labels". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 647–657.

[14] Jean Feydy et al. "Interpolating between optimal transport and MMD using Sinkhorn divergences". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. 2019, pp. 2681–2690.

[15] B. Frenay and M. Verleysen. "Classification in the Presence of Label Noise: A Survey". In: *IEEE Transactions on Neural Networks and Learning Systems* 25.5 (2014), pp. 845–869.

[16] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. “A neural algorithm of artistic style”. In: *arXiv preprint arXiv:1508.06576* (2015).

[17] Aritra Ghosh, Himanshu Kumar, and PS Sastry. “Robust loss functions under label noise for deep neural networks”. In: *arXiv preprint arXiv:1712.09482* (2017).

[18] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[19] Jacob Goldberger and Ehud Ben-Reuven. “Training deep neural-networks using a noise adaptation layer”. In: (2016).

[20] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. “Speech recognition with deep recurrent neural networks”. In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 6645–6649.

[21] Arthur Gretton et al. “A kernel two-sample test”. In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 723–773.

[22] Bo Han et al. “Co-teaching: Robust training of deep neural networks with extremely noisy labels”. In: *Advances in neural information processing systems*. 2018, pp. 8527–8537.

[23] Yan Han et al. “Learning from Noisy Labels via Discrepant Collaborative Training”. In: *The IEEE Winter Conference on Applications of Computer Vision*. 2020, pp. 3169–3178.

[24] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[25] Kaiming He et al. “Identity mappings in deep residual networks”. In: *European conference on computer vision*. Springer. 2016, pp. 630–645.

[26] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. “Using pre-training can improve model robustness and uncertainty”. In: *arXiv preprint arXiv:1901.09960* (2019).

[27] Ray J Hickey. “Noise modelling and evaluating learning from examples”. In: *Artificial Intelligence* 82.1-2 (1996), pp. 157–179.

[28] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. “Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 935–944.

[29] Lu Jiang et al. “Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels”. In: *International Conference on Machine Learning*. 2018, pp. 2304–2313.

[30] Ishan Jindal, Matthew Nokleby, and Xuewen Chen. “Learning deep networks from noisy labels with dropout regularization”. In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE. 2016, pp. 967–972.

[31] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[32] Svetlana Kiritchenko and Stan Matwin. “Email classification with co-training”. In: *Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research*. Citeseer. 2001, p. 8.

[33] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[35] Mark-A Krogel and Tobias Scheffer. “Multi-relational learning, text mining, and semi-supervised learning for functional genomics”. In: *Machine Learning* 57.1-2 (2004), pp. 61–81.

[36] Solomon Kullback and Richard A Leibler. “On information and sufficiency”. In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[37] Abhishek Kumar and Hal Daumé. “A co-training approach for multi-view spectral clustering”. In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 393–400.

[38] Yann LeCun et al. “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4 (1989), pp. 541–551.

[39] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[40] Junnan Li et al. “Learning to learn from noisy labeled data”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5051–5059.

[41] Geert Litjens et al. “A survey on deep learning in medical image analysis”. In: *Medical image analysis* 42 (2017), pp. 60–88.

[42] Ce Liu and Hueng-Yeung Shum. “Kullback-leibler boosting”. In: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* Vol. 1. IEEE. 2003, pp. I–I.

[43] Xin Liu et al. “Self-error-correcting convolutional neural network for learning with noisy labels”. In: *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*. IEEE. 2017, pp. 111–117.

[44] Christopher Manning and Hinrich Schutze. *Foundations of statistical natural language processing*. MIT press, 1999.

[45] Dominic Masters and Carlo Luschi. “Revisiting small batch training for deep neural networks”. In: *arXiv preprint arXiv:1804.07612* (2018).

[46] Sam McCandlish et al. “An empirical model of large-batch training”. In: *arXiv preprint arXiv:1812.06162* (2018).

[47] Duc Tam Nguyen et al. “Robust Learning Under Label Noise With Iterative Noise-Filtering”. In: *arXiv preprint arXiv:1906.00216* (2019).

[48] Hyeonwoo Noh et al. “Regularizing deep neural networks by noise: Its interpretation and optimization”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 5109–5118.

[49] Giorgio Patrini et al. “Making deep neural networks robust to label noise: A loss correction approach”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1944–1952.

[50] Jesse Read et al. “Classifier chains for multi-label classification”. In: *Machine learning* 85.3 (2011), p. 333.

[51] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[52] Mengye Ren et al. “Learning to reweight examples for robust deep learning”. In: *arXiv preprint arXiv:1803.09050* (2018).

[53] David Silver et al. “Mastering the game of go without human knowledge”. In: *nature* 550.7676 (2017), pp. 354–359.

[54] Alexander Sorokin and David Forsyth. “Utility data annotation with amazon mechanical turk”. In: *2008 IEEE computer society conference on computer vision and pattern recognition workshops*. IEEE. 2008, pp. 1–8.

[55] Newton SpolaôR et al. “A comparison of multi-label feature selection methods using the problem transformation approach”. In: *Electronic Notes in Theoretical Computer Science* 292 (2013), pp. 135–151.

[56] Sainbayar Sukhbaatar et al. “Training convolutional networks with noisy labels”. In: *arXiv preprint arXiv:1406.2080* (2014).

[57] Gencer Sumbul et al. “Bigearthnet: A large-scale benchmark archive for remote sensing image understanding”. In: *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*. IEEE. 2019, pp. 5901–5904.

[58] Gencer Sumbul et al. “BigEarthNet Dataset with A New Class-Nomenclature for Remote Sensing Image Understanding”. In: *arXiv* (2020), arXiv–2001.

[59] Bo Sun et al. “A robust multi-class AdaBoost algorithm for mislabeled noisy data”. In: *Knowledge-Based Systems* 102 (2016), pp. 87–102.

[60] Choh Man Teng. “Evaluating Noise Correction”. In: *PRICAI 2000 Topics in Artificial Intelligence*. Ed. by Riichiro Mizoguchi and John Slaney. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 188–198. ISBN: 978-3-540-44533-3.

[61] Grigorios Tsoumakas and Ioannis Katakis. “Multi-label classification: An overview”. In: *International Journal of Data Warehousing and Mining (IJDWM)* 3.3 (2007), pp. 1–13.

[62] Grigorios Tsoumakas and Ioannis Vlahavas. “Random k-labelsets: An ensemble method for multilabel classification”. In: *European conference on machine learning*. Springer. 2007, pp. 406–417.

[63] SS Vallender. “Calculation of the Wasserstein distance between probability distributions on the line”. In: *Theory of Probability & Its Applications* 18.4 (1974), pp. 784–786.

[64] Jean-Philippe Vert, Koji Tsuda, and Bernhard Schölkopf. “A primer on kernel methods”. In: *Kernel methods in computational biology* 47 (2004), pp. 35–70.

[65] JÃ´natas Wehrmann and Rodrigo C. Barros. “Movie genre classification: A multi-label approach based on convolutions through time”. In: *Applied Soft Computing* 61 (2017), pp. 973 –982. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2017.08.029. URL: http://www.sciencedirect.com/science/article/pii/S1568494617305112.

[66] Xiang Wu et al. “A light cnn for deep face representation with noisy labels”. In: *IEEE Transactions on Information Forensics and Security* 13.11 (2018), pp. 2884–2896.

[67] Yi Yang and Shawn Newsam. “Bag-of-visual-words and spatial extensions for land-use classification”. In: *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*. 2010, pp. 270–279.

[68] Kun Yi and Jianxin Wu. “Probabilistic end-to-end noise correction for learning with noisy labels”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 7017–7025.

[69] Bodi Yuan et al. “Iterative cross learning on noisy labels”. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, pp. 757–765.

[70] Ming Yuan and Yi Lin. “Model selection and estimation in regression with grouped variables”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.1 (2006), pp. 49–67.

[71] Chiyuan Zhang et al. “Understanding deep learning requires rethinking generalization”. In: *arXiv preprint arXiv:1611.03530* (2016).

[72] Jingpu Zhang et al. “Ontological function annotation of long non-coding RNAs through hierarchical multi-label classification”. In: *Bioinformatics* 34.10 (Dec. 2017), pp. 1750–1757. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btx833. eprint: https://academic.oup.com/bioinformatics/article-pdf/34/10/1750/25118236/btx833.pdf. URL: https://doi.org/10.1093/bioinformatics/btx833.

[73] Min-Ling Zhang and Zhi-Hua Zhou. “A review on multi-label learning algorithms”. In: *IEEE transactions on knowledge and data engineering* 26.8 (2013), pp. 1819–1837.

[74] Min-Ling Zhang and Zhi-Hua Zhou. “ML-KNN: A lazy learning approach to multi-label learning”. In: *Pattern recognition* 40.7 (2007), pp. 2038–2048.

[75] Zhilu Zhang and Mert Sabuncu. “Generalized cross entropy loss for training deep neural networks with noisy labels”. In: *Advances in neural information processing systems*. 2018, pp. 8778–8788.

[76] Xingquan Zhu and Xindong Wu. “Class noise vs. attribute noise: A quantitative study”. In: *Artificial intelligence review* 22.3 (2004), pp. 177–210.