Handling noisy annotations in deep supervised learning

Ichraq Lemghari, Sylvie Le Hégarat, Emanuel Aldea, Jennifer Vandoni

Handling Noisy Annotations in Deep Supervised Learning

Ichraq Lemghari[a,b], Sylvie Le Hégarat-Mascle[a], Emanuel Aldea[a], and Jennifer Vandoni[b]

[a]SATIE, Paris-Saclay University, Gif-sur-Yvette, France
[b]Safran Tech, Digital Sciences & Technologies Department, Magny-Les-Hameaux, France

ABSTRACT

Non-destructive testing (NDT) is employed by companies to assess the features of a material, in order to identify some variations or anomalies in its properties without causing any damage to the original object. In this context of industrial visual inspection, the help of new technologies and especially deep supervised learning is nowadays required to reach a very high level of performance. Data labelling, that is essential to reach such performance, may be fastidious and tricky, and only experts can provide the labelling of the material possible defects. Considering classification problems, this paper addresses the issue of **handling noisy labels in datasets**. We will first present the existing works related to the problem, our general idea of how to handle it, then we will present our proposed method in detail along with the obtained results that reach more than 0.96 and 0.88 of accuracy for noisified MNIST and CIFAR-10 respectively with a 40% noise ratio. Finally, we present some potential perspectives for future works.

Keywords: Supervised learning, classification, noisy labels, set-valued decisions, partial ignorance modelling.

1. INTRODUCTION AND PROBLEM STATEMENT

Deep Learning (DL) relies heavily on large annotated training datasets.[1] However, accurate manual expert-labelling of each instance on a large scale is not feasible and often prohibitively expensive for some applications,[2] such as anomaly or defect detection. In these circumstances, three factors negatively impact the dataset creation and subsequently the learning process: the scarcity of some types of samples, the scarcity of qualified annotators, and the higher level of noise being present in the provided labels due to the difficulty of the task. Therefore, imperfect datasets presenting label noise are often employed,[3] especially in the context of NDT. The impact of this label noise is very clear on the results that are strongly degraded, because deep neural networks are known for memorizing and eventually fitting label noise.[4] In this context, our main objective is to find a way to effectively train deep neural networks with modern architectures under label noise in a supervised learning classification setting.

In this work, we formalize the problem in the context of a c-class classification. Let $\mathcal{D}$ be the *clean* (i.e., *not noisy*) dataset, which consists of a d-dimensional feature space $\mathcal{X} \subset \mathbb{R}^d$ and a true label space $\mathcal{Y} = \{0,1\}^c$, and let $\tilde{\mathcal{D}}$ be the *noisy* dataset, which consists of the same d-dimensional feature space $\mathcal{X} \subset \mathbb{R}^d$ and a noisy label space $\tilde{\mathcal{Y}} = \{0,1\}^c$. The goal of the learning step in the classification task is to learn the mapping function $f(.;\Theta) : \mathcal{X} \rightarrow [0,1]^c$ such that the parameter Θ minimizes the empirical risk $\mathcal{R}_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}[l(f(x;\Theta),y)] = \frac{1}{|\mathcal{D}|}\Sigma_{(x,y)\in\mathcal{D}} l(f(x;\Theta),y)$ where l is the considered loss function. However, since data labels are usually noisy in real-world scenarios, this risk is calculated with noisy data, and the learned classifier is obtained as: $\hat{f} = \arg\min_{f\in\mathcal{F}} \mathbb{E}_{\tilde{\mathcal{D}}}[l(f(x;\Theta),\bar{y})]$. Thus, the model can be biased by the corrupted labels, and in order to find a way to mitigate the impact of those noisy labels, one has to follow a *noise-tolerant* training process.

To do so, we will handle set-valued labels corresponding to imprecise or partially ignorant labels. More specifically, if $\Omega = \{\omega_1, \ldots, \omega_c\}$ denotes the label set (associated to $\mathcal{Y}$ which is defined in the one-hot encoding space), we denote by 2^Ω the powerset of Ω, i.e. the set of the Ω subsets: $2^\Omega = \{\emptyset, \omega_1, \omega_2, \{\omega_1,\omega_2\}, \omega_3, \{\omega_1,\omega_3\}, \{\omega_2,\omega_3\}, \ldots, \Omega\}$. We consider the belief function theory[5] in order to model the belief on 2^Ω, and a mass function: $m : 2^\Omega \rightarrow [0,1]$, $\sum_{A\subseteq\Omega} m(A) = 1$. Finally, let us call *imprecise* any decision function g with $g : \mathcal{V} \rightarrow 2^\Omega$, where $\mathcal{V}$ is the score space which is usually equal to $[0,1]^c$ (*e.g.*, an output layer providing a normalized score vector of dimensionality equal to the number of classes) but can be equal to $[0,1]^{2^c}$ if an evidential layer is used and the output is a mass function m.

Further author information: Ichraq Lemghari. E-mail: ichraq.lemghari@safrangroup.com

2. RELATED WORK

2.1 Noisy labels

Learning from noisy labels is a high-stakes but challenging task in modern Deep Learning (DL) applications. Several DL approaches that have been studied in this context address the problem from different perspectives.

The first type of methods, based on *robust architectures*, focuses on the architectural modifications that can model the transition between true and noisy labels[3,6] It mainly includes two strategies, which work by either adding a noise adaptation layer at the top of the Deep Neural Network (DNN) model, or developing a dedicated architecture to handle the label noise.

A second category of methods, called *robust regularization*, consists of advanced regularization techniques that specifically enforce a DNN to overfit less the false-labeled samples.[7] It can have an implicit form such as data augmentation or adversarial training[8] , or an explicit form that modifies the expected training loss such as in Bilevel learning,[9] which uses a clean validation dataset to regularize the overfitting of a model by introducing a bilevel optimization approach.

Sample selection is a third strategy whose objective is to separate the true-labeled samples from the noisy ones based on a specific metric.[10–13] Hence, only the samples selected as *clean* (i.e., *not noisy*) are used during the update of the DNN model in the training step.

Finally, a fourth type of methods, known as *loss adjustment*, proposes to adjust the loss computation on the noisy training data before updating the DNN model, so that the impact of noisy labels can be reduced.[14,15] For example, *Forward Loss Correction*[14] is a popular method, which adjusts the loss function by integrating an estimated label transition matrix T in the training in order to mimic the behavior of the noisy labels. The procedure is divided into two main steps, namely the transition matrix estimation, which summarizes the probability of one class being flipped into another one, and the forward loss correction, that adjusts the loss.

2.2 Evidential Convolutional Neural Network

In order to handle uncertainty in DL classification tasks, Evidential Convolutional Neural Networks (ECNNs)[16,17] have been proposed. ECNNs are built upon the framework of Convolutional Neural Networks (CNNs) in which the output is a probability distribution over all possible *subsets* of classes, along with different measures of uncertainty.[18]

Following the *evidential* layer that replaces the usual *softmax* layer, a so-called *utility* layer[18] is used for decision making. This layer allows for the adoption of set-valued solutions corresponding to imprecise decisions.[19,20] The basic idea behind a partial classification decision is "better imprecise than wrong", which can be formulated using the utility matrix $U \in [0,1]^{(2^c-1)\times c}$, which specifies the utility $u_{A,k}$ of any decision $A \subseteq \Omega, A \neq \emptyset$, when the true label is $k \in \Omega$. This matrix allows us to calculate the expected-utility for each decision in favor of an hypothesis $A \in 2^\Omega$ using the Hurwicz criterion:[18]

$$\mathbb{E}_{m,\nu}(A) = \nu \underline{\mathbb{E}}_{m,\nu}(A) + (1-\nu)\overline{\mathbb{E}}_{m,\nu}(A), \tag{1}$$

where m is the mass function[18] corresponding to the general case and $\nu \in [0,1]$ is a *pessimism index* that weights the lower and upper expected utilities $\underline{\mathbb{E}}_{m,\nu}(A)$ and $\overline{\mathbb{E}}_{m,\nu}(A)$:

$$\underline{\mathbb{E}}_{m,\nu}(A) = \sum_{B\subseteq\Omega} m(B) \min_{\omega_j \in B} u_{A,j}, \qquad \overline{\mathbb{E}}_{m,\nu}(A) = \sum_{B\subseteq\Omega} m(B) \max_{\omega_j \in B} u_{A,j}. \tag{2}$$

3. PROPOSED METHOD

As mentioned in the problem statement, in the presence of complex data, the task of finding a hard partitioning among the classes is particularly difficult even for expert annotators. It thus results in noisy labeled datasets, which leads to poor ML model performance. Now, labelling errors are mainly due to ambiguities between some specific classes. They are thus not completely random but rather correspond to some confusion between two or three classes. Hence, we propose to handle these confusions by considering set-valued labels, also called "imprecise" labels, e.g. $\{\omega_i, \omega_j\}$ when we doubt between ω_i and ω_j classes for a given sample.

In this context, starting with our dataset $\bar{\mathcal{D}}$ containing noisy labels, we derive a two-step strategy that relies on: (1) A set-valued classifier able to **identify** the noisy samples; (2) A method to **correct** our model trained with noisy samples. The interactions between these two steps, that are detailed in the two following subsections, are summarized on Fig. 1. More specifically, the overall training procedure can be described in Algorithm 1.

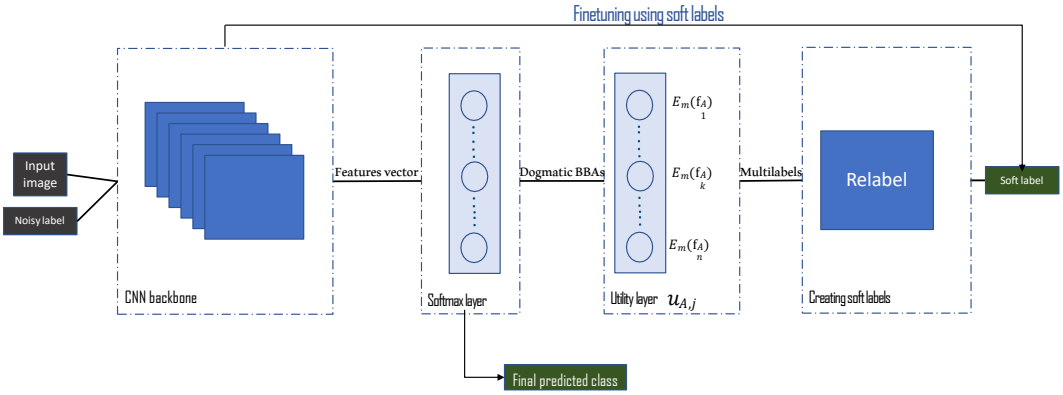


Figure 1: Scheme of the proposed method.

Algorithm 1: The overall training algorithm for dealing with noisy labels

Data: Training set $\bar{D}_N = \{x_i, \bar{y}_i\}, i \in [\![1, N]\!]$ possibly having noisy labels; Untrained CNN classifier
Result: Trained CNN classifier
1 Train the set-valued CNN classifier on the noisy dataset $\bar{D}_N$;
2 Use the result set-valued classification to create soft labels for the training dataset using Eq. (5);
3 Use the new dataset with soft labels as a training dataset for the classifier model that is initialized with the model trained on the original noisy dataset without the utility layer.

3.1 Set-valued convolutional neural network classifier

In this section, we describe the proposed classifier that is derived from the ECNN classifier presented in Section 2.2. We first present the overall structure of the model and then explain the role of each part.

3.1.1 The architecture

As illustrated in Figure 1, our model involves mainly three components:

- a **CNN backbone**, which is used to extract some relevant latent features for the classification by propagating the input samples through its different hidden layers, and which consists mainly in convolution and pooling operations.
- a **Softmax layer**, that specifically converts the feature vectors into dogmatic Basic Belief Assignments (BBAs, cf. Section 3.1.2) which quantify the probability distribution of the sample class.
- a **Utility layer**, which calculates for each sample the expected-utility vector defined on the $2^{\Omega} \setminus \{\emptyset\}$, based on the considered utility matrix U.

3.1.2 From Non Dogmatic to Bayesian BBA

In the ECNN, the hypotheses $A \subseteq \Omega$ with a non-null mass value (i.e., $m(A) > 0$) , called the focal sets, are the singleton hypotheses, namely $\omega_1, \omega_2, \ldots, \omega_c$, and the hypothesis representing the complete ignorance, namely Ω. Now, using Eq. (1), we can show that introducing some mass on Ω promotes decisions in favor of singleton classes or Ω. For illustration, Figure 2 shows, for a 3-class problem ($\Omega = \{\omega_1, \omega_2, \omega_3\}$), how the different decision areas evolve when we increase the mass on Ω (from left to right), and we clearly notice that the areas

corresponding to decisions in favor of the hypotheses with cardinality equal to 2 ($\{\omega_i, \omega_j\}, (i,j) \in \{1,2,3\}, i \neq j$) decrease with $m(\Omega)$ until they completely disappear when $m(\Omega) = 0.3$.

To illustrate further the evolution of these decision areas, we draw some samples in the different decision areas defined when $m(\Omega) = 0$ (Fig. 2-a), and then we analyse the distribution of their predicted label in 2^Ω (i.e. possibly imprecise based on the Hurwitz criterion) when $m(\Omega)$ increases. Figure 3 shows some distribution examples versus $m(\Omega)$ for the samples initially in $\{\omega_1, \omega_2\}$ decision area. We notice that, when $m(\Omega)$ increases, the samples are redistributed between the two singleton classes ω_1 and ω_2, and the Ω class, with a different final behaviour against ν, i.e. either completely redistributed in the two singleton classes when $\nu = 0.2$ or in Ω when $\nu = 0.5$. Furthermore, according to some distribution results not shown in this paper, the samples which are initially in a singleton class can be redistributed only in Ω when this latter increases (for $\nu \geq 0.5$).

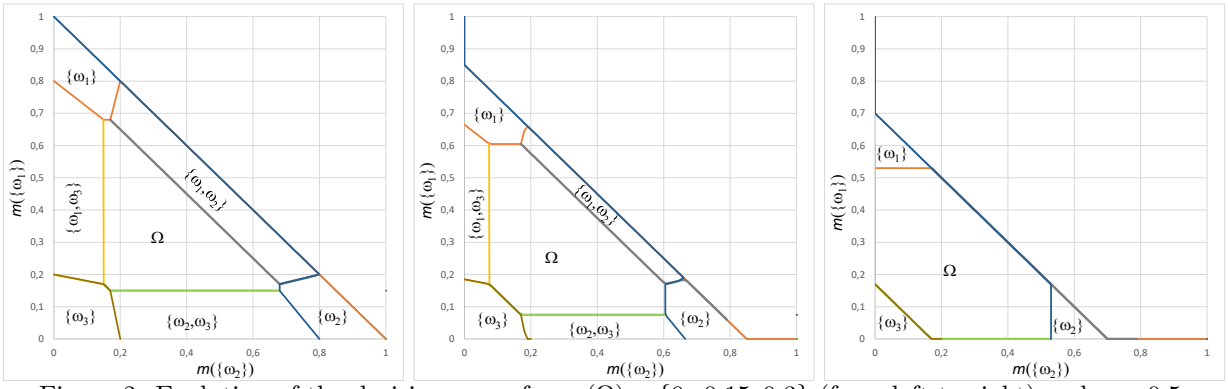


Figure 2: Evolution of the decision areas for $m(\Omega) \in \{0., 0.15, 0.3\}$ (from left to right) and $\nu = 0.5$.

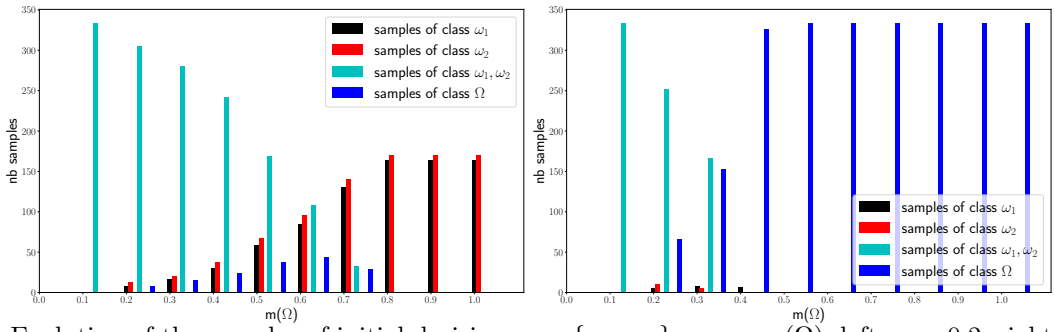


Figure 3: Evolution of the samples of initial decision area $\{\omega_1, \omega_2\}$ versus $m(\Omega)$; left: $\nu = 0.2$, right: $\nu = 0.5$.

Now, recall that we assume that confusions mostly occur when two classes are mislabeled as each other and our objective with set-valued classification is to assign classes with a cardinality of *two* for the uncertain samples, i.e. optimally the noisy samples. Then, relaxing the constraint of decision among singleton classes, we nevertheless aim at keeping the predicted imprecise classes with as low cardinality as possible. Thus, based on the previous analysis, in this work, we decide to set $m(\Omega) = 0$ and only keep singletons as focal sets, i.e. to handle only *Bayesian* BBAs.

3.1.3 Utility layer

The *utility* layer output is the expected-utility vector of size equal to $2^c - 1$. To compute it, we connect any unit j of the *softmax* layer to each output unit representing a hypothesis $A \subseteq \Omega$ with a weight equal to $u_{A,j}$, which specifies the utility of assigning the input sample to $A \subseteq \Omega$ when the true label is $\omega_j \in \Omega$.

The coefficients $u_{A,j}$ of the utility matrix $U \in [0,1]^{(2^c-1)\times c}$ are defined as an Ordered Weighted Average (OWA),[21] and thus only depend on the cardinality of the considered imprecise class A. In,[22] they are obtained

by maximizing the entropy of the OWA operator, given an imprecision tolerance degree γ.
In this work, we rather propose an estimation of $u_{A,j} \in [0,1]$ in order to make the noisy samples be classified in an imprecise class of cardinality 2 (the noisy label and the true one) and the not noisy samples be classified in a singleton class (cardinality 1). Since we mostly focus on imprecise classes of cardinality 2, $\forall A$ such that $|A| = 1$ we set $u_{A,j} = 1$, $\forall A$ such that $|A| > 2$ we set $u_{A,j}$ to a low value, and we set to a same value, denoted by $u_2 \in (0,1)$, all the utilities $u_{A,j}$ $\forall A$ such that $|A| = 2$. Then, the estimation of u_2 is as follows.
Based on Eq. (1), and (2), for any BBA, having as only focal sets the singletons ω_i and Ω

$$\forall i \in [\![1,c]\!], \mathbb{E}_m(\{\omega_i\}) = m(\{\omega_i\}) + (1-\nu)\, m(\Omega), \tag{3}$$

$$\forall (i,j) \in [\![1,c]\!]^2, i \neq j, \mathbb{E}_m(\{\omega_i,\omega_j\}) = u_2 [m(\{\omega_i\}) + m(\{\omega_j\}) + (1-\nu)\, m(\Omega)]. \tag{4}$$

Then, we propose to plot the distributions of the quantity $\alpha_{1,2} = \frac{m(\{\omega_i\})+(1-\nu)m(\Omega)}{m(\{\omega_i\})+m(\{\omega_j\})+(1-\nu)m(\Omega)}$, which allows us to distinguish the clean samples correctly labelled ω_i from the noisy samples wrongly labelled ω_j with true label ω_i. Indeed, we aim to set the u_2 value such that it allows us to separate the *clean* samples from the *noisy* ones by classifying the first ones in singleton classes.

3.2 Training with soft labels

As mentioned before, the main objective of the set-valued classifier is to identify the noisy samples and assign them new class sets with cardinality greater than one. Indeed, we aim to train again our classifier model after having replaced the noisy labels of these samples, that correspond to a false class, by soft labels associated to a subset of classes which optimally includes also the true class. A possible way to fulfill this objective is to create some *soft* labels as follows.

First, let us recall that in the simple case of Bayesian BBAs, the mass function is equal to the belief function and to the plausibility function, and that the mass function restriction to singletons is also equal to the pignistic probability and to the contour function. Then, for any sample i classified as belonging to $A \subseteq \Omega$, its *soft* label is defined as:

$$\forall i \in [\![1,n]\!], y_i^{soft} := \begin{cases} y_i^{soft}(\omega_j) = \frac{m(\omega_j)}{\sum_{\omega_k \in A} m(\omega_k)} & \text{if } \omega_j \in A, \\ y_i^{soft}(\omega_j) = 0 & \text{otherwise.} \end{cases} \tag{5}$$

4. EXPERIMENTAL RESULTS

We tested our method on the MNIST[23] and CIFAR-10[24] datasets.

4.1 Simulating realistic label corruption

Before carrying out the experiments, we artificially integrate noise in the datasets by noisifying samples in a manner that mimics the mistakes that can occur in a realistic dataset for similar classes, namely the classes $1 \longleftrightarrow 7$ for MNIST and $CAT(3) \longleftrightarrow DOG(5)$ for CIFAR-10, with varying ratios of noise. Figure 4 shows some examples of confusing samples in MNIST and CIFAR-10 respectively.

(a) One (b) One (c) Seven (d) Seven (e) Cat (f) Cat (g) Dog (h) Dog

Figure 4: Examples of confusing samples for MNIST$(1-7)$ and CIFAR-10(Cat-Dog)

With this aim, we first train a CNN backbone with a *softmax* layer on the original dataset without noise in order to get the class score vectors for each sample. Then, for a specific ratio of noise $n \in [0,1]$, we noisify the proportion $n \times N$ of the samples that have the smallest scores for their ground-truth classes, i.e. samples that are hard for the model to predict. The noise is introduced symmetrically between the classes in question, i.e. labels of samples of classes $1/CAT$ are replaced by $7/DOG$ and vice versa.

4.2 Set-valued classification

4.2.1 Learning parameters

Common to all experiments, the loss function used is cross entropy. The training dataset is divided into 20% for validation and 80% for training and the test dataset is not noisified. Adam optimizer is used during the training. For MNIST, a convolutional network with two convolutional hidden layers, initialized with Kaiming method, is used with a batch of size 128. Training is performed during 50 epochs with a learning rate of 0.001. For CIFAR-10, we employ a ResNet-34[25] pretrained on ImageNet with a batch size of 256. Training is performed during 10 epochs with a learning rate of 10^{-5}.

4.2.2 Utility matrix estimation

Starting from the noisy dataset, we first estimate the values of the utility matrix needed for the set-valued classification, based on the estimation process presented in Section 3.1.3. By plotting the distribution of $\alpha_{1,2}$ for the different classes, we notice that for the noisified classes, there is a difference between the distribution of the noisy and not noisy samples, which gives an approximation interval of the value u_2 (cf. Section 3.1.3). Figures 5 and 6 represent this distribution in the case of the noisified MNIST and CIFAR-10 datasets with different ratios of noise. Specifically, we empirically find that u_2 belongs to the range $[0.55, 0.6]$ with a default value about 0.57. Hence, in the following, we employ a utility matrix where $u_2 = 0.57$, while $\forall A \subseteq \Omega$ such that $|A| > 2, u_{A,j} = 0.1$ (low value to avoid decisions in favor to classes of cardinality greater than two).

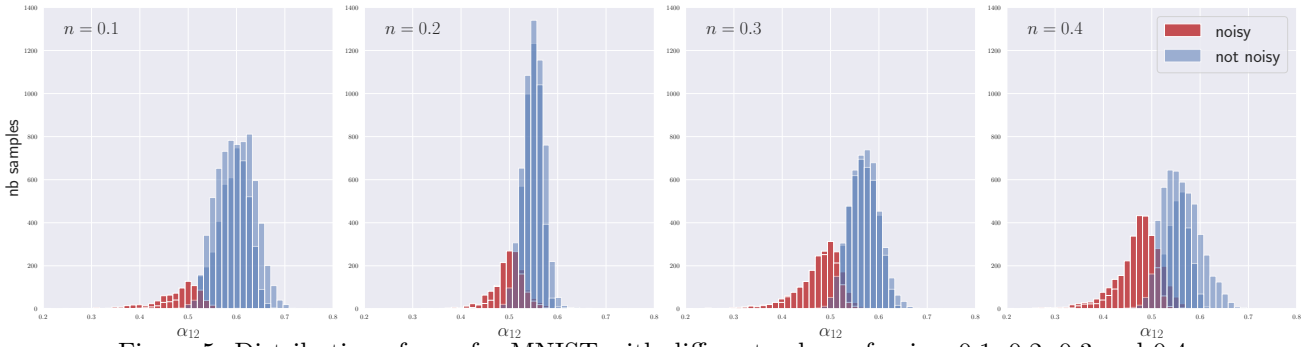


Figure 5: Distribution of $\alpha_{1,2}$ for MNIST with different values of noise: 0.1, 0.2, 0.3 and 0.4

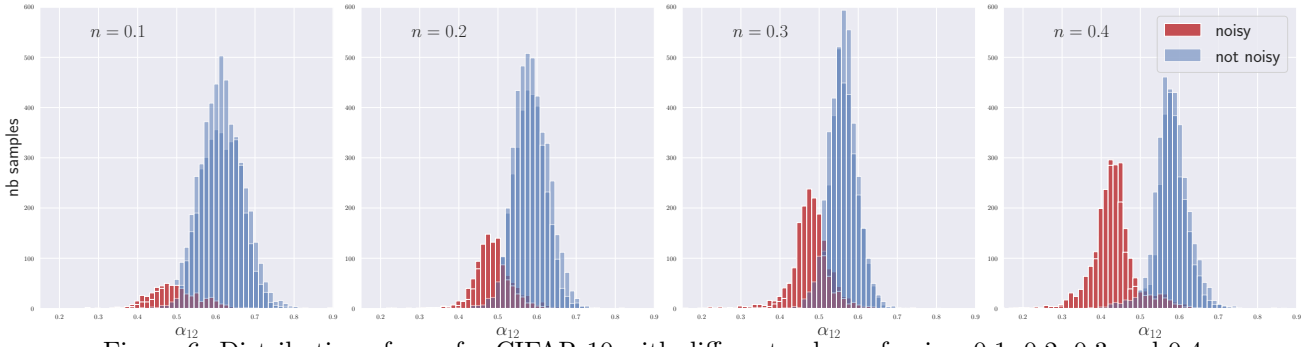
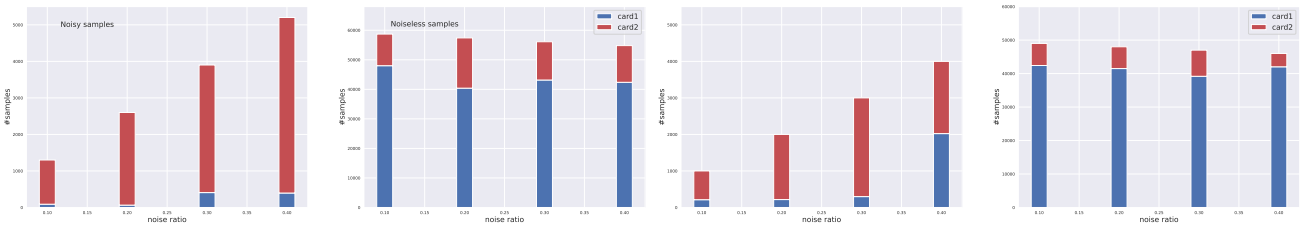


Figure 6: Distribution of $\alpha_{1,2}$ for CIFAR-10 with different values of noise: 0.1, 0.2, 0.3 and 0.4

4.2.3 Set-valued classification

The set-valued classifier enables us to obtain multi-labels classification. The distribution of the obtained multi-labels among noisy and not noisy samples is represented in Figures 7a and 7b for MNIST and CIFAR-10 respectively, for a noise ratio varying as $\{0.1, 0.2, 0.3, 0.4\}$. The model is mostly able to identify the noisy samples, especially when the noise ratio is small, by assigning them to some sets of classes of cardinality two, while the majority of the not noisy samples were assigned to singleton classes.



(a) (b)

Figure 7: Distribution of multi-labels (red) with noise (noisy samples on the left/not noisy on the right) for (a) MNIST and (b) CIFAR-10

4.3 Final experimental results and comparison with the related works

After obtaining the set-valued classification, the dataset is relabelled with *soft* labels as explained in Section 3.2 and the model is finetuned to give the final classification results, that we present in this section. In order to assess the given results, Table 1 presents the evolution of the accuracy against the noise ratio for three methods, namely training the considered CNN model on the original noisy dataset, training using the T-forward correction procedure[14] and finally the proposed method.

Method/Dataset	MNIST				CIFAR-10			
	$n = 0.1$	$n = 0.2$	$n = 0.3$	$n = 0.4$	$n = 0.1$	$n = 0.2$	$n = 0.3$	$n = 0.4$
CNN model	0.9717	0.9585	0.9395	0.9232	0.9241,	0.9141	0.8925	0.8701
T-forward[14]	0.9743	0.9605	0.9523	0.9477	0.9298	0.9204	0.9048	0.8781
Our method	**0.9897**	**0.9830**	**0.9731**	**0.9644**	**0.9411**	**0.9346**	**0.9131**	**0.8852**

Table 1: Accuracy values for different ratios of symmetric noise for MNIST and CIFAR-10.

As we can see, for both CIFAR-10 and MNIST datasets, our method was able to reach higher values of accuracy than the T-forward procedure. The improvement is clearly robust to the noise ratio introduced in the dataset, and is more relevant when using MNIST dataset since it is more easy to handle. These results show that identifying the noisy samples helps more the model to adjust its classification results than using an estimated transition matrix to correct the loss function because the transition matrix models the class ambiguities without taking into account the dependence to the samples.

5. CONCLUSIONS AND FUTURE WORKS

In this paper, we have proposed a framework for dealing with noisy labels in datasets, using a set-valued (imprecise) classification. We derived a new classifier that combines a CNN with a utility layer in order to relabel the noisy samples in the dataset based on their predicted multi-labels.
The performance of the method has been assessed in case of using the MNIST and CIFAR-10 datasets with different ratios of noise, and the accuracy was consistently better than the training using a CNN model and the popular T-forward correction method.

Several perspectives remain open. Firstly, recall that ECNNs integrate the uncertainty by placing a given amount of mass on Ω, without considering subsets of classes. Thus, in this first work, we end up using Bayesian BBAs because using the $(c + 1)$ mass values does not reflect pair-wise class ambiguities since Ω is the only compound focal element.
Hence, in future work, to benefit from the powerful modeling of belief functions, we aim to define an evidential layer whose output is a BBA with focal elements of intermediate cardinality values between one (singleton classes) and $|\Omega|$, which feeds the utility layer. Secondly, we aim to integrate this utility layer in the training by adjusting the loss function to depend also on the utilities.

REFERENCES

[1] Daniely, A. and Granot, E., "Generalization bounds for neural networks via approximate description length," *Advances in Neural Information Processing Systems* **32** (2019).

[2] Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., and Sugiyama, M., "Co-teaching: Robust training of deep neural networks with extremely noisy labels," *Advances in neural information processing systems* **31** (2018).

[3] Xiao, T., Xia, T., Yang, Y., Huang, C., and Wang, X., "Learning from massive noisy labeled data for image classification," in [*Proc. of the IEEE conf. on CVPR*], 2691–2699 (2015).

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

[5] Shafer, G., [*A mathematical theory of evidence*], vol. 42, Princeton university press (1976).

[6] Sukhbaatar, S., Bruna, J., Paluri, M., Bourdev, L., and Fergus, R., "Training convolutional networks with noisy labels," (2015).

[7] Tanno, R., Saeedi, A., Sankaranarayanan, S., Alexander, D. C., and Silberman, N., "Learning from noisy labels by regularized estimation of annotator confusion," in [*Proc. of the IEEE/CVF conference on CVPR*], 11244–11253 (2019).

[8] Goodfellow, I. J., Shlens, J., and Szegedy, C., "Explaining and harnessing adversarial examples," (2015).

[9] Jenni, S. and Favaro, P., "Deep bilevel learning," in [*Proc. of ECCV*], 618–633 (2018).

[10] Lyu, Y. and Tsang, I. W., "Curriculum loss: Robust learning and generalization against label corruption," (2019).

[11] Malach, E. and Shalev-Shwartz, S., "Decoupling 'when to update' from 'how to update'," *Advances in neural information processing systems* **30** (2017).

[12] Song, H., Kim, M., and Lee, J.-G., "Selfie: Refurbishing unclean samples for robust deep learning," in [*Int. Conf. on Machine Learning*], 5907–5915, PMLR (2019).

[13] Song, H., Kim, M., Park, D., Shin, Y., and Lee, J.-G., "Robust learning by self-transition for handling noisy labels," in [*Proc. of the ACM SIGKDD Conf. on Knowledge Discovery & Data Mining*], 1490–1500 (2021).

[14] Patrini, G., Rozza, A., Krishna Menon, A., Nock, R., and Qu, L., "Making deep neural networks robust to label noise: A loss correction approach," in [*Proc. of the IEEE conf. on CVPR*], 1944–1952 (2017).

[15] Hendrycks, D., Mazeika, M., Wilson, D., and Gimpel, K., "Using trusted data to train deep networks on labels corrupted by severe noise," *Advances in neural information processing systems* **31** (2018).

[16] Denoeux, T., "A neural network classifier based on dempster-shafer theory," *IEEE Trans. on Systems, Man, and Cybernetics-Part A: Systems and Humans* **30**(2), 131–150 (2000).

[17] Denoeux, T., Kanjanatarakul, O., and Sriboonchitta, S., "A new evidential k-nearest neighbor rule based on contextual discounting with partially supervised learning," *International Journal of Approximate Reasoning* **113**, 287–302 (2019).

[18] Tong, Z., Xu, P., and Denoeux, T., "An evidential classifier based on dempster-shafer theory and deep learning," *Neurocomputing* **450**, 275–293 (2021).

[19] Chen, X.-l., Wang, P.-h., Hao, Y.-s., and Zhao, M., "Evidential knn-based condition monitoring and early warning method with applications in power plant," *Neurocomputing* **315**, 18–32 (2018).

[20] Guettari, N., Capelle-Laizé, A. S., and Carré, P., "Blind image steganalysis based on evidential k-nearest neighbors," in [*2016 IEEE Int. Conf. on Image Processing*], 2742–2746, IEEE (2016).

[21] Yager, R. R., "On ordered weighted averaging aggregation operators in multicriteria decisionmaking," *IEEE Trans. on systems, Man, and Cybernetics* **18**(1), 183–190 (1988).

[22] Ma, L. and Denoeux, T., "Partial classification in the belief function framework," *Knowledge-Based Systems* **214**, 106742 (2021).

[23] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., "Gradient-based learning applied to document recognition," *Proc. of the IEEE* **86**(11), 2278–2324 (1998).

[24] Krizhevsky, A., Hinton, G., et al., "Learning multiple layers of features from tiny images," (2009).

[25] He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition," in [*Proc. of the IEEE conf. on CVPR*], 770–778 (2016).