From Noisy Prediction to True Label: Noisy Prediction Calibration via Generative Model

HeeSun Bae [*1] **Seungjae Shin** [*1] **Byeonghu Na** [1] **JoonHo Jang** [1] **Kyungwoo Song** [2] **Il-Chul Moon** [1 3]

Abstract

Noisy labels are inevitable yet problematic in machine learning society. It ruins the generalization of a classifier by making the classifier over-fitted to noisy labels. Existing methods on noisy label have focused on modifying the classifier during the training procedure. It has two potential problems. First, these methods are not applicable to a pre-trained classifier without further access to training. Second, it is not easy to train a classifier and regularize all negative effects from noisy labels, simultaneously. We suggest a new branch of method, *Noisy Prediction Calibration* (NPC) in learning with noisy labels. Through the introduction and estimation of a new type of transition matrix via generative model, NPC corrects the noisy prediction from the pre-trained classifier to the true label as a post-processing scheme. We prove that NPC theoretically aligns with the transition matrix based methods. Yet, NPC empirically provides more accurate pathway to estimate true label, even without involvement in classifier learning. Also, NPC is applicable to any classifier trained with noisy label methods, if training instances and its predictions are available. Our method, NPC, boosts the classification performances of all baseline models on both synthetic and real-world datasets. The implemented code is available at https://github.com/BaeHeeSun/NPC.

1. Introduction

The success of deep neural networks heavily relies on large-scale datasets with annotations (Yao et al., 2020). Whereas the importance of large-size dataset is unanimous, creating such large-scale dataset is arduous and often prone to human errors in their label annotations (Cheng et al., 2021). For instance, the recent utilization of crowd-sourcing (Welinder et al., 2010) or search engines (Xiao et al., 2015) in building the datasets potentially results in the problem of *noisy label* (Han et al., 2018b; Yi & Wu, 2019; Xia et al., 2020a; Liu et al., 2020; Wang et al., 2021). The model training with noisy label could be detrimental given the fitness of over-parameterized neural network to the training dataset, because such networks are even ready to fit the mislabeled training instances, a.k.a *memorization* (Arpit et al., 2017; Zhang et al., 2021a; Xia et al., 2020a) problem.

Several studies have suggested resolutions on *memorization* from noisy labels. First, *noise-cleansing* approaches (Malach & Shalev-Shwartz, 2017; Han et al., 2018a; Tanaka et al., 2018; Han et al., 2018b; Yu et al., 2019; Han et al., 2020; Wei et al., 2020; Zheng et al., 2020; 2021; Wang et al., 2021; Kim et al., 2021) focus on segregating the clean data pairs from the corrupted dataset, based on the outputs of noisy classifier, i.e. loss, entropy, and feature alignment. Second, *noise-robust* approaches utilize explicit regularizations (Liu et al., 2020; Xia et al., 2020a) or robust loss functions (Zhang & Sabuncu, 2018; Wang et al., 2019; Ma et al., 2020) to design a robust classifier. Yet, their modeling and structures mostly originate from either heuristics or hard assumptions. Another line of researches suggest an explicit formulation on noise patterns, i.e. *transition matrix* T from a true label to a noisy one (Patrini et al., 2017; Yao et al., 2020; Zhang et al., 2021b; Xia et al., 2020b; Berthon et al., 2021). While methods with T provide rigorous formulation on the label modifications, accurate estimation of T heavily depends on the inference of true label y, which is assumed latent from observations (Yao et al., 2021).

With the unsolved challenges above, classifiers trained by existing methods are still not robust to label noises, depending on the various noise characteristics (Song et al., 2020). It necessitates the modelling of reducing the gap between the prediction of trained classifier and the true latent label. Post-processing can be effective for this objective, in that it is easily applicable to any trained classifier without access to training. Motivated by this spirit, we introduce another

[*]Equal contribution [1]Industrial and Systems Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea [2]Department of AI, University of Seoul, Seoul, Republic of Korea [3]Summary.AI, Daejeon, Republic of Korea. Correspondence to: Il-Chul Moon <icmoon@kaist.ac.kr>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

branch of solutions on the problem of noisy labels. We propose the algorithm, coined Noisy Prediction Calibration (NPC), which estimates the explicit transition from a noisy prediction to a true latent class via utilizing a deep generative model. NPC operates as a post-processing module to a black-box classifier trained on noisy labels, while previous transition models were applied during the training procedures. Therefore, NPC can expand the scalability of transition models in terms of learning time by far, and NPC inherits wider usability with its post-processing nature.

For theoretical aspect, we prove that NPC is interchangeable with the transition matrix, which generalizes the modeling framework of NPC. For empirical aspect, NPC significantly boosts the accuracy of classifiers trained with various type of noisy labels without any access to noise ratio. Moreover, NPC captures potentially noisy data instances from the learning with benchmark datasets, i.e. MNIST and Fashion-MNIST, which have been believed to be clean.

2. Problem Definition

2.1. Problem Setup

Assuming a classification task of c classes, let $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} = \{1, 2, ..., c\}$ be a input space and a label set, respectively. Given the input and the label spaces, the i.i.d. samples from the joint probability distribution P over $\mathcal{X} \times \mathcal{Y}$, $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, becomes a classification dataset. Unlike traditional supervised learning, our assumption on the noisy label dictates that only observables are $\tilde{\mathcal{D}} = \{(x_i, \tilde{y}_i)\}_{i=1}^n$, which are samples from $\tilde{P}$, potentially different from P.

The training objective of a classifier f is to minimize the true risk, $R_L(f) := \mathbb{E}_P\left[L\left(f\left(x\right), y\right)\right]$, yet the only accessible risk function is the noisy empirical risk of $\tilde{R}_L^{emp}(f) := \frac{1}{n}\sum_{i=1}^n L\left(f\left(x_i\right), \tilde{y}_i\right)$. Hence, when learning with noisy labels, the objective becomes finding a function that minimizes $R_L(f)$ via the learning procedure with $\tilde{R}_L^{emp}(f)$.

2.2. Previous Research on Noisy Label Classification

Neural networks trained with gradient descent can easily fit even random labels (Zhang et al., 2021a). To tackle this issue, various works have been introduced for learning with noisy labels. One directional approaches include the extraction of reliable clean samples (Han et al., 2018b; Yu et al., 2019; Wei et al., 2020), label modification (Tanaka et al., 2018; Yi & Wu, 2019; Zheng et al., 2020; Li et al., 2020; Zheng et al., 2021; Wang et al., 2021), introducing noise-robust losses (Zhang & Sabuncu, 2018; Wang et al., 2019), and additive regularization (Liu et al., 2020; Xia et al., 2020a). [1] These works usually hinge upon heuristics or assumptions, such as the phenomenon of learning *simple pattern* at the early stage of learning (Arpit et al., 2017). These experimentally justified heuristics require hand-picked hyper-parameters, such as noise ratio, early stopping time, and noisy indication threshold, which are critical for their performance improvements.

Other type of approaches estimate the true class probability by formulating a transition matrix, T, as follows:

$$T_{kj}(x) = p(\tilde{y} = j | y = k, x) \text{ for all } j, k = 1, ..., c \quad (1)$$

Transition matrix, T, provides a probabilistic formulation on label transition from true class y to noisy label $\tilde{y}$. Since $p(\tilde{y}|x) = \sum_{k=1}^c p(\tilde{y}|y = k, x)p(y = k|x)$, T provides a pathway to the true loss from observable $\tilde{y}$ and x as follows:

$$E_{\tilde{P}}[L(T(f(x)), \tilde{y})] = R_L(f) \quad (2)$$

The benefit of transition matrix methods is the explicit formulation of label distribution modifications. However, estimating T with a latent variable y is not an easy problem.

2.3. Previous Studies on Transition Matrix Estimation

$p(\tilde{y} = j|y = i, x)$ has a large support space if the input space has a large dimension. As an initial solution, (Patrini et al., 2017; Yao et al., 2020; Zhang et al., 2021b) suggested the transition matrix as $p(\tilde{y} = j|y = i, x) = p(\tilde{y} = j|y = i)$ assuming the instance independence. Yet, the estimation gap exists with this assumption because x influences on the transition to $\tilde{y}$, i.e. an image of *3* resembling to *5* in MNIST and its potential mislabeling (Chen et al., 2020). Various models have appeared to estimate an instance-dependent T (Chen et al., 2020; Xia et al., 2020b; Berthon et al., 2021; Yao et al., 2021), but they still have limitations since they rely on either assumption of part-dependent label noise (Xia et al., 2020b) or access to additional information on confidence score (Berthon et al., 2021).

Emphasizing the importance on estimating latent y, a recent study (Yao et al., 2021) proposes an estimation of T by maximizing the likelihood of observable variables, $p(x, \tilde{y})$, through a generative model with the latent y. They model a noisy data generative process as Figure 2a. Their model, CausalNL introduces an auxiliary latent variable of z that is another source of information to generate x, other than y so that z and y jointly generate x. Eventually, this model learns the generation process of x by Variational Autoencoder (VAE) (Kingma & Welling, 2013), and this serves the maximization of $p(x, \tilde{y})$, which leaves $p(z, x, y, \tilde{y})$ as a side-product that becomes an ingredient to formulate T. This generation is inferred via maximizing Evidence Lower Bound (ELBO) of Eq. 3.

$$\begin{aligned}\text{ELBO}(x, \tilde{y}) &= \mathbb{E}_{(z,y)\sim q_\phi(z,y|x)}[\log p_\theta(x|y, z)] \\ &+ \mathbb{E}_{y\sim q_\phi(y|x)}[\log p_\theta(\tilde{y}|y, x)] - \text{KL}(q_\phi(y|x)||p(y)) \\ &- \mathbb{E}_{y\sim q_\phi(y|x)}[\text{KL}(q_\phi(z|y, x)||p(z))]\end{aligned} \quad (3)$$

[1] A detailed description of each method is in Appendix A.1.

This estimation framework may have two shortcomings. First, the generation of x, whose data resolution can be high, would not be an appropriate objective for a noisy classification task. This model would infer the generation process with sub-optimal reconstruction, a well-known problem of VAE (Dosovitskiy & Brox, 2016). Therefore, inference on T with VAE would lead to sub-optimal, as well. Second, CausalNL assumes that there is an additional cause of x from z, so z and y jointly generate x. Given the observed x, z and y become dependent by V-structure as in Figure 2a. For classification, the correlation between z and y needs to be disentangled from z because y needs to model only the pure label information. This disentanglement can be a burden particularly given that the generation of x is not the main goal of classification with noisy label.

Deep generative models for classification with noisy labels have not yet been actively studied. However, the importance of estimating the true class y as latent motivates us to utilize a deep generative model. Having said that, there are differences between CausalNL and our model, NPC, in terms of generative model design and modeling purpose. From the perspective of generative model design, our method differs from CausalNL because we remove the unnecessary generation of x without any auxiliary latent variables. For modelling purposes, CausalNL improves the classifier learning through identifiable estimation of T. On the contrary, we introduce and estimate a new type of transition matrix, H, which is mainly utilized to post-process the outputs of classifiers trained on noisy datasets. We discuss the advantageous properties of post-processing for noisy classifiers in Section 3.1. Also, we claim that our introduced transition, H, is interchangeable with T in Section 3.3.

3. Method

This section presents our model, Noisy Prediction Calibration (NPC), by going through its theoretic formulation, probabilistic model structure, and its inference procedure. Additionally, we discuss the relation between NPC and the original transition matrix T.

3.1. Motivational Comparison with Transition Matrix

Before we present our approach, we first explain our abstract view of noisy label classification in this section. Let ψ be a set of model parameters of a classifier. Then, there are three cases of ψ: 1) $\tilde{\psi}$ which composes the function that explains the noisy labelled dataset perfectly; 2) $\hat{\psi}$ which is equivalent to classifiers trained with algorithms for learning with noisy labels; and 3) ψ^* is from the optimal classifier that best explains the true joint distribution of x and y.

Traditional methodologies for dealing with noisy labels have focused on finding ψ^* with observation of $(x,\tilde{y})$. Without any access to the true joint distribution between x and y, it results in $\hat{\psi}$. Corresponding to the definition of $\hat{\psi}$, a classifier f maps x to $\hat{y}$, a prediction at the current training. Figure 1 shows these mappings from x to y, $\tilde{y}$, and $\hat{y}$.

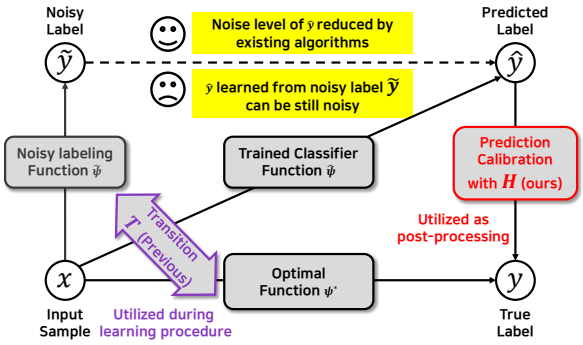


Figure 1. **Overview.** Existing methods shift the function ψ to estimate the mapping from x to y during the learning procedure. On the contrary, our method, NPC, formulates a prediction calibration procedure which reduces the gap between noisy prediction $\hat{y}$ and true label y under the post-processing framework. NPC successfully boosts the pre-trained classifier performances by leveraging the classifier predictions as noise-reduced inputs.

Previously, there was no explicit modelling on $\hat{y}$ because $\hat{y}$ was considered as a transient state converging to y. This transience is modelled with T in the previous research instead of articulating $\hat{y}$. T is merged into the learning process as Eq. 2, so T becomes a force to converge on y with only $\tilde{R}_L^{emp}$, which we represents as a purple texts in Figure 1.

We hypothesize that there could be also a chance of *calibration* at the label space of y, instead of the parameter space of ψ. We call this procedure as Noisy Prediction Calibration (NPC) because $\hat{\psi}$ is assumed to be fixed and to produce $\hat{y}$. The calibration will transform $\hat{y} \rightarrow y$ by H as a prediction adjustment, and it is formulated as Eq. 4.

$$
\begin{aligned}
p(y|x) &= \sum_{\hat{y}} p(y|\hat{y},x)p(\hat{y}|x) \\
H_{kj}(x) &= p(y=j|\hat{y}=k,x) \;\; for\; j,k=1,...,c
\end{aligned}
\tag{4}
$$

The predicted label $\hat{y}$ from the classifier trained by algorithms for managing noisy labels has the potential to reduce the noise-level compared to the original noisy label $\tilde{y}$. Then, by explicitly mentioning $\hat{y}$ from the post-processing stage, we find a novel opportunity to adjust $\hat{y}$. If there is a trained classifier f [2], the previous methods with T are not applicable to calibrate noisy labels to manage the relation from $\hat{y}$ to y. With post-processing mechanism, however, an adjustment $\hat{y} \rightarrow y$ without altering $\hat{\psi}$ can be applied to the pre-trained classifier as the red text in Figure 1.

[2]Such black-box classifiers include deep neural networks with too many parameters (Brown et al., 2020; Dosovitskiy et al., 2020) for training given a user's computing environment.

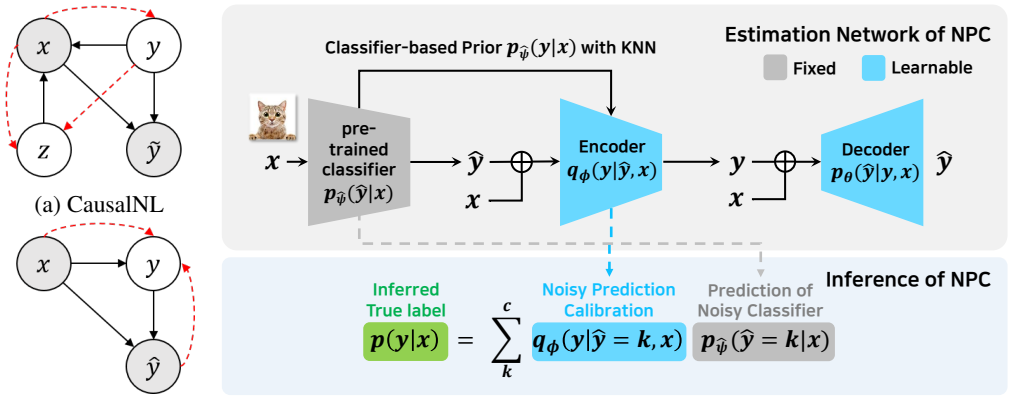


Figure 2. **Structure.** Bayesian network of (a) CausalNL and (b) NPC. Arrows with solid lines and dashed lines denote generative process and inference process, respectively; illustrations of (c) Neural Network structure (upper) and inference (under) of NPC, respectively.

The most critical difference between 1) a classifier training with noisy labels with T and 2) a noisy prediction calibration with H in post-processing is whether influencing on $\hat{\psi}$ or calibrating $\hat{y}$. Surely, the calibration quality of H depends upon the quality of $\hat{y}$ and the structure of $\hat{H}$, the estimation of H. Section 3.2 describes our probabilistic estimation on $\hat{H}$ with minimal introduction of latent variables. Then, we show whether T and H can be analytically aligned with different formulation. Section 3.3 provides a theoretic claim that T and H are potentially interchangeable with some implementable weighting variables.

3.2. NPC : Noisy Prediction Calibration Algorithm

As described in Eq. 4, we need to estimate $p(y|\hat{y},x)$ and $p(\hat{y}|x)$. Since $p(\hat{y}|x)$ can be obtained by training a classifier, we focus on the estimation of $p(y|\hat{y},x)$.

3.2.1. Probabilistic Model Structure

We model a generative process of data instances with label noise as Figure 2b. This generative process expresses the probabilistic relation of y and $\tilde{y}$ given input x.

1. Choose a latent vector $y \sim \text{Dir}(\alpha_x)$
2. Choose a noisy label $\tilde{y} \sim \text{Multi}(\pi_{x,y})$

α_x is the instance-dependent parameter of the prior probability distribution given a sample x, $\alpha_x \in \mathbb{R}_+^c$. $\text{Dir}(\alpha_x)$ is a Dirichlet distribution, a conjugate prior of the corresponding multinomial distribution. $\pi_{x,y}^k$ is the probability for selecting a class $k = 1, ..., c$ for the noisy label, $\tilde{y}$ [3]. $\text{Multi}(\pi_{x,y})$ is the multinomial distribution.

This generative process describes the following scenario of noisy labelling. Given an input x, there is a prior distribution of its true label y with a Dirichlet distribution. Then, the input content and its prior information on true label dictates a noisy label $\tilde{y}$.

According to the generative process above, the joint probability $p(y, \hat{y}, x)$ can be factorized as follows:

$$p(y, \hat{y}, x) \propto p(y|x)p(\hat{y}|y, x) \tag{5}$$

Here, as $p(y|x)$ is unknown from our training state, we approximate it with our pre-trained classifier parameterized by $\hat{\psi}$. The probabilities are defined as:

$$p_{\hat{\psi}}(y|x) = \text{Dir}(\alpha_x),\ p(\tilde{y}|y, x) = \text{Multi}(\pi_{x,y}) \tag{6}$$

3.2.2. Parameter Inference

Our main estimation target is the posterior probability $p(y|\hat{y}, x)$, which is intractable. Accordingly, we follow the variational inference (Kingma & Welling, 2013) framework to minimize the KL divergence between the inference distribution and the target distribution. We introduce a variational distribution, $q(y|\hat{y}, x)$, and we parametrize the variational and the remaining model distributions with ϕ and θ. These formulations result in the KL divergence as Eq. 7.

$$\begin{aligned}
&\text{KL}(q_\phi(y|\hat{y},x)||p(y|\hat{y},x)) \\
&= \int q_\phi(y|\hat{y},x) \log \frac{q_\phi(y|\hat{y},x)}{p(y|\hat{y},x)} dy \\
&= \int q_\phi(y|\hat{y},x) \log \frac{q_\phi(y|\hat{y},x)p(\hat{y}|x)}{p(\hat{y}|y,x)p(y|x)} dy \\
&= \log p(\hat{y}|x) - E_{y\sim q_\phi(y|\hat{y},x)}\left[\log p_\theta(\hat{y}|y,x)\right] \\
&+ \text{KL}(q_\phi(y|\hat{y},x)||p(y|x)) = \log p(\hat{y}|x) - \text{ELBO}
\end{aligned} \tag{7}$$

[3] $\pi_{x,y} \in \mathbb{R}_+^c$, $\sum_{k=1}^c \pi_{x,y}^k = 1$

With a latent variable y following the Dirichlet distribution, the direct reparameterization is difficult unlike a case with normal distribution prior (Kingma & Welling, 2013). Therefore, we utilize the reparametrization trick from (Joo et al., 2020), which decomposes Dirichlet distribution into Gamma distributions with their inverse CDF[4]. Finally, Eq. 8 presents ELBO in Eq. 7 as a objective function.

$$\begin{aligned}\text{ELBO} = &\sum_{k=1}^{c} \hat{y}^i \log \hat{y^*}^k + (1-\hat{y}^k)\log(1-\hat{y^*}^k) \\ &- \sum_{k=1}^{c} \log\Gamma(\alpha_x^k) + \sum_{k=1}^{c} \log\Gamma(\hat{\alpha}_{x,\bar{y}}^k) \\ &- \sum_{k=1}^{c} (\hat{\alpha}_{x,\bar{y}}^k - \alpha_x^k)\psi(\hat{\alpha}_{x,\bar{y}}^k)\end{aligned} \tag{8}$$

Here, $\hat{y^*}$ is the reconstruction output; Γ and ψ are the gamma and digamma function, respectively.

3.2.3. Implementation

Figure 2c shows the illustration of a neural network structure for NPC. Since we are interested in $p(y|\hat{y}, x)$, we adopted the structure of Variational Autoencoder (VAE) to generate the parameter of variational posterior distribution.

We design a prior distribution of the latent variable y to be dependent on x, which is enabled by utilizing the pre-trained classifier $\hat{\psi}$. We apply K-Nearest Neighbor (KNN) algorithm (Wang et al., 2018; Bahri et al., 2020) to samples with high confidence in the classifier output, $p_{\hat{\psi}}(\hat{y}|x)$. The most selected label from k neighbours is referred as $\bar{y}$, and we differentiate the parameter values of the prior Dirichlet distribution of y by $\bar{y}$ as Eq. 9.

$$\alpha_x^k = \begin{cases} \delta & k \neq \bar{y} \\ \delta + \rho & k = \bar{y} \end{cases} \quad for\ k = 1, ..., c \tag{9}$$

Here, δ and ρ are hyper-parameters to setup α of the Dirichlet distribution. Throughout the paper, we fixed $\delta = 1$.

While setting the prior parameters as above, we obtain the posterior distribution from the encoder. We select the soft plus function as the activation function of the encoder to make the posterior distribution parameter non-negative. The resulting posterior distribution provides a parameter sample of the posterior multinomial distribution H. We set the parameter of H to be the mode of the posterior Dirichlet distribution, which becomes $H = p(y|\hat{y}, x) = \frac{\alpha_{x,\hat{y}}^y - 1}{\sum_{y=1}^{c} \alpha_{x,\hat{y}}^y - c}$.

With the definition of H, we calibrate the noisy prediction described as in Figure 1, indicating $\hat{y} \rightarrow y$. This is an transition from the prediction of noisy classifier $(p(\hat{y}|x))$ to the true probability of label distribution $(p(y|x))$, which we will calculate as Eq. 10.

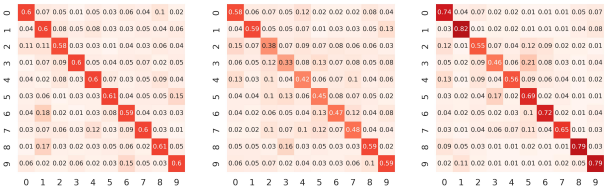
$$\begin{aligned} p(y|x) &= \sum_k p(y|\hat{y}=k, x) p(\hat{y}=k|x) \\ &\approx \sum_k q_\phi(y|\hat{y}=k, x) p(\hat{y}=k|x) \end{aligned} \tag{10}$$

3.3. Alignment of T and H

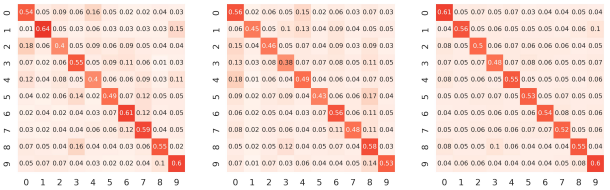
When we utilize NPC as a post-processing algorithm, the classifier $p_{\hat{\psi}}(\hat{y}|x)$ becomes a fixed model in Figure 2c. Therefore, training H does not affect ψ, the classifier parameters, unlike training T. Nevertheless, we can relate our modeling with T as stated in Proposition 3.1 (full proof in Appendix C.1). This alignment shows that our methodology provides a same pathway to correct the noisy classifier as in the previous studies on T.

Proposition 3.1. *Assume that $\hat{y}$ and y are conditionally independent given $\tilde{y}$, and $p(\hat{y} = k|x) \neq 0$ for all $k = 1, .., c$. Then, $H_{kj}(x) = \frac{p(y=j|x)}{p(\tilde{y}=k|x)} \sum_i p(\tilde{y} = k|\tilde{y} = i, x) T_{ij}(x)$ for all $j, k = 1, ..., c$.*

The assumption of $\hat{y} \perp\!\!\!\perp y|\tilde{y}$ is natural because $\hat{y}$ is conditionally independent to y when $\hat{\psi}$ is trained only with $\tilde{y}$. Proposition 3.1 shows that H can be formulated by T with weighting variables, which are observable or can be easily computed from our framework. It implies that we can also infer T from the inference procedure of H, which suggests the possibility of relating the framework of NPC to the transition matrix based approaches.



(a) True Transition (b) Forward (0.004) (c) Dualt (0.004)



(d) TVR (0.003) (e) CausalNL (0.005) (f) NPC (0.002)

Figure 3. (a) Average of the true transition matrix T of CIFAR-10 with instance dependent noise 40%, (b)-(e) the estimated transition matrix of each algorithm. Values in parentheses are the mean squared error between estimated $\hat{T}$ and true T.

Figure 3 shows (a) true transition matrix T and the empirically estimated $\hat{T}$ (b)-(e) from various algorithms for

[4] we describe the reparameterization details in Appendix B.

CIFAR-10 dataset with 40% instance dependent noise. We calculate the mean squared error between true T and estimated $\hat{T}$ from each method. It should be noted that the estimated $\hat{T}$ is an aggregated matrix from the input-wisely estimated matrix. Although the main inference target of NPC is not T but H, NPC shows lowest estimation error of T, even without any access to the noise ratio. Details of the experiment and more results of transition matrix estimation on other noise conditions are provided in Appendix C.2.

4. Experiments

In this section, we first introduce implementation details and baselines for experiments in 4.1. Afterwards, we provide quantitative performance results in 4.2 and qualitative analyses in 4.3. We also compare our method with other post-processing methods in 4.4, with CausalNL in 4.5. Finally, we provide NPC's ability as a detector of potentially noisy samples in the benchmark dataset in 4.6.

4.1. Datasets and Baselines

To verify the efficacy of our proposed method, NPC, we first experiment on three benchmark datasets: *MNIST* (LeCun, 1998), *Fashion-MNIST* (Xiao et al., 2017), and *CIFAR-10* (Krizhevsky et al., 2009). We inject synthetic label noises following the previous setups of existing works (Patrini et al., 2017; Han et al., 2018b; Xia et al., 2020a; Berthon et al., 2021; Yao et al., 2021). Four different types of artificial label noise are generated as below. More details on each noise type are provided in Appendix D.1.1. We also consider the clean setting without any noise to analyze the generalization ability of NPC.

- **Symmetric Noise (SN)** flips labels uniformly to all other classes (Patrini et al., 2017; Han et al., 2018b; Xia et al., 2020a).
- **Asymmetric Noise (ASN)** flips labels within a set of semantically similar classes (Tanaka et al., 2018; Han et al., 2018b; Xia et al., 2020a).
- **Instance Dependent Noise (IDN)** flips labels by the probability that a data instance is mislabeled. This probability is computed based on the corresponding feature of the data instance (Xia et al., 2020b; Cheng et al., 2021; Yao et al., 2021).
- **Similarity Reflected Instance Dependent Noise (SRIDN)** not only considers instance-wise similarity, but also takes into account of inter-class similarity of each data instance when flipping the labels not (Chen et al., 2020; Berthon et al., 2021).

Also, we compute experiments on two real-world datasets: *Clothing-1M* (Xiao et al., 2015) and *Food101* (Bossard et al., 2014). Details of datasets and implementations are provided in Appendix D.1.2 and D.1.3, respectively.

We demonstrate the performance of NPC as a post-processor, by applying NPC to a classifier pre-trained with baseline methods. The selected algorithms include: (1) **CE** (Cross Entropy), (2) **Joint** (Tanaka et al., 2018), (3) **Coteaching** (Han et al., 2018b), (4) **JoCoR** (Wei et al., 2020), (5) **CORES2** (Cheng et al., 2021), (6) **SCE** (Wang et al., 2019), (7) **ES** (EarlyStop), (8) **LS** (Label Smoothing) (Lukasik et al., 2020), (9) **REL** (Xia et al., 2020a), (10) **Forward** (Patrini et al., 2017), (11) **DualT** (Yao et al., 2020), (12) **TVR** (Zhang et al., 2021b), and (13) **CausalNL** (Yao et al., 2021). More details are explained in Appendix D.1.4.

4.2. Classification Performance on Noisy Datasets

Table 1 shows the experimental results on three synthetic datasets with the four types of noisy types of various noisy ratios. By applying NPC to the selected 13 baselines as a post-processing model, we have total of 351 experimental cells including the clean cases. With five replications with different seeds, we got 341 cells with statistically significant improvement by NPC.

This result demonstrates that NPC is widely applicable to any type of pre-trained classifiers; and that NPC effectively calibrates any types of noisy labels without any access to noise information, i.e. noise ratio and noise type. It is noteworthy that NPC achieves impressive performances on IDN conditions on average, and we conjecture that with its generative modelling structure, NPC successfully calibrates instance dependent noise, which is quite a realistic noisy label generation process. Moreover, we observe that there are consistent performance enhancements in the clean cases. These clean-case improvements state that NPC also improves the generalization of the applied classifier.

While the previous results originate from synthetic datasets, Table 2 shows the experimental results on *Food-101* and *Clothing1M*, which are datasets with real world noisy labels with its test dataset containing human-annotated labels. Again, NPC improves the performance of baseline classifiers on their classification accuracy with statistical significance in 13 out of 16 experimental cells.

4.3. Qualitative Analysis of $q_\phi(y|\hat{y}, x)$

Figure 4 shows t-SNE mapping views and confusion matrices of our samples from variational posterior, $q_\phi(y|\hat{y}, x)$[5]. Each dot is colored by the original classifier output $\hat{y}$ (Figure 4a) or $y^* = argmax_y P(y|x)$ calibrated with NPC (Figure 4b), respectively. Comparing two t-SNE views, we can observe that the latent variable y is well clustered to reflect the

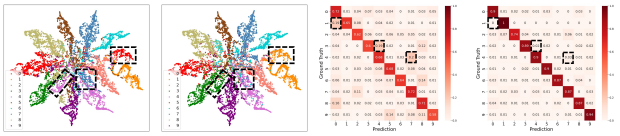
[5]The analysis is based on MNIST dataset with 40% IDN.

Model	MNIST		Fashion-MNIST									CIFAR-10								
	Clean	IDN	Clean	SN		ASN		IDN		SRIDN		Clean	SN		ASN		IDN		SRIDN	
	-	40%	-	20%	80%	20%	40%	20%	40%	20%	40%	-	20%	80%	20%	40%	20%	40%	20%	40%
CE	97.8	66.3	87.1	74.0	27.0	81.0	77.3	68.4	52.1	81.0	67.3	86.9	73.1	15.1	80.2	71.4	72.9	53.9	72.6	61.8
w/ NPC	**98.2**	**89.0**	**88.4**	**84.0**	**35.8**	**85.9**	**86.2**	**82.5**	**74.5**	**81.8**	**69.4**	**89.0**	**80.8**	**17.0**	**84.7**	**78.8**	**80.9**	**59.9**	**74.3**	**64.3**
Joint	93.0	93.6	82.8	82.0	**6.0**	82.1	82.3	82.7	82.4	80.6	74.6	83.0	78.9	**8.3**	81.5	76.8	80.4	64.5	70.6	62.2
w/ NPC	**94.0**	**94.6**	**83.6**	**82.7**	6.0	**82.9**	**82.9**	**83.4**	**83.0**	**81.1**	**75.5**	**84.4**	**80.2**	8.3	**83.0**	**77.7**	**80.7**	**69.1**	**72.0**	**63.6**
Coteaching	98.0	87.5	87.0	82.5	64.2	88.2	**73.6**	81.8	75.4	84.0	75.0	88.5	82.5	29.7	86.5	76.6	81.5	75.2	75.3	66.6
w/ NPC	**98.3**	**90.6**	**88.3**	**85.8**	**66.0**	**88.5**	**73.6**	**85.1**	**78.7**	**84.2**	**75.3**	**89.2**	**85.3**	**32.1**	**87.1**	**76.8**	**84.8**	**78.5**	**76.1**	**67.2**
JoCoR	97.8	93.3	88.7	86.0	27.6	88.9	79.4	86.3	83.2	81.9	71.3	89.1	83.6	24.8	82.6	73.3	82.8	75.3	75.2	66.1
w/ NPC	**98.3**	**96.1**	**89.8**	**88.0**	**31.5**	**89.2**	**82.7**	**88.0**	**85.7**	**82.2**	**72.3**	**89.3**	**86.0**	**27.0**	**85.1**	**79.0**	**85.8**	**80.1**	**75.9**	**66.7**
CORES2	97.0	48.8	87.2	74.6	8.9	77.6	74.3	80.0	58.1	81.3	71.2	87.1	70.1	**31.2**	79.0	71.2	70.3	50.9	72.8	62.0
w/ NPC	**98.0**	**67.2**	**88.5**	**84.3**	**10.2**	**82.5**	**81.0**	**84.0**	**69.6**	**82.2**	**74.9**	**88.2**	**80.4**	30.7	**84.2**	**80.4**	**80.4**	**65.6**	**74.2**	**64.1**
SCE	97.7	66.6	87.0	74.0	27.0	82.0	77.4	68.3	52.0	81.1	67.5	86.9	73.1	15.1	80.2	71.4	72.9	53.9	72.6	61.8
w/ NPC	**98.2**	**88.7**	**88.3**	**83.7**	**35.5**	**86.4**	**86.7**	**82.0**	**75.2**	**81.8**	**69.7**	**87.4**	**75.0**	**15.2**	**81.5**	**75.2**	**75.4**	**55.6**	**72.9**	**62.5**
Early Stop	96.5	73.3	87.5	83.6	49.5	84.1	76.6	79.5	55.4	83.3	72.6	83.0	79.1	18.0	80.9	70.6	77.1	62.5	71.4	60.6
w/ NPC	**97.9**	**90.8**	**88.7**	**85.9**	**62.9**	**87.6**	**87.1**	**84.3**	**75.3**	**84.0**	**76.0**	**84.0**	**82.5**	**18.2**	**81.2**	**72.0**	**79.4**	**65.1**	**72.1**	**63.0**
LS	97.8	66.2	87.5	73.9	27.8	81.5	77.0	69.0	52.5	81.1	67.5	86.9	73.1	15.1	80.2	71.4	72.9	53.9	72.6	61.8
w/ NPC	**98.2**	**88.6**	**88.6**	**83.7**	**35.2**	**86.0**	**86.4**	**82.2**	**74.7**	**81.6**	**69.5**	**89.0**	**80.8**	**15.5**	**84.7**	**78.8**	**80.9**	**59.9**	**74.3**	**64.3**
REL	**98.0**	90.7	**88.1**	84.6	70.1	82.8	76.2	**84.6**	75.5	**83.7**	78.1	80.7	74.9	21.2	72.8	69.9	75.5	**51.8**	69.3	63.8
w/ NPC	97.9	**95.5**	86.9	**85.0**	**70.3**	**85.3**	**83.0**	83.8	**80.1**	82.9	**78.3**	**83.4**	**78.6**	**26.0**	**75.9**	**76.1**	**78.5**	51.2	**70.7**	**64.2**
Forward	98.0	67.9	88.5	77.4	24.3	83.3	79.2	75.2	56.9	82.4	69.5	85.3	71.8	16.9	78.2	70.1	70.2	54.5	73.2	63.5
w/ NPC	**98.4**	**91.1**	**89.6**	**85.3**	**33.0**	**87.2**	**86.8**	**86.8**	**80.5**	**83.3**	**73.7**	**88.7**	**81.5**	**17.2**	**83.8**	**74.5**	**80.3**	**63.3**	**74.8**	**65.0**
DualT	96.7	94.3	86.3	84.5	**10.0**	86.9	83.1	85.1	68.5	82.7	73.2	84.3	79.3	7.6	80.6	77.1	78.6	71.2	68.7	63.1
w/ NPC	**97.8**	**96.6**	**88.2**	**85.9**	**10.0**	**87.6**	**84.3**	**86.3**	**72.3**	**83.4**	**74.9**	**86.0**	**83.0**	**8.4**	**83.0**	**77.5**	**81.0**	**77.3**	**70.1**	**64.0**
TVR	97.7	64.4	87.0	72.6	24.9	80.6	76.4	66.3	51.7	81.4	67.7	86.7	71.9	15.2	78.5	71.2	72.3	53.6	72.2	62.2
w/ NPC	**98.1**	**84.5**	**88.3**	**82.3**	**31.9**	**84.9**	**85.3**	**79.8**	**73.6**	**82.1**	**70.3**	**88.3**	**80.8**	**15.7**	**84.1**	**76.5**	**80.8**	**60.7**	**74.5**	**64.5**
CausalNL	98.1	85.2	88.1	84.0	51.5	88.8	87.4	83.4	75.2	82.0	71.2	89.6	79.9	17.0	84.6	74.8	79.9	60.4	74.6	63.5
w/ NPC	**98.6**	**94.5**	**89.4**	**87.0**	**58.9**	**89.3**	**88.7**	**87.6**	**83.3**	**83.3**	**74.1**	**89.7**	**81.2**	**18.8**	**85.0**	**74.8**	**81.2**	**71.9**	**75.3**	**63.9**

Table 1. Test accuracies for MNIST, Fashion MNIST and CIFAR-10 datasets with their labels corrupted by four types of noisy label conditions. We demonstrate averaged performances computed by baselines and the post-hoc performances after applying NPC. The experimental results are averaged values over five trials. **Bolded** text denotes the one with better performance. For MNIST, experiment results on other noise conditions are provided in Appendix D.2.

Method	Food-101		Clothing1M	
	w.o/ NPC	w/ NPC	w.o/ NPC	w/ NPC
CE	78.37	**80.21**±0.2	68.14	**70.83**±0.1
Early Stop	73.22	**76.80**±0.3	67.07	**70.21**±0.1
SCE	75.23	**78.26**±0.3	67.77	**70.36**±0.1
REL	78.96	78.95±0.4	62.53	**64.83**±0.1
Forward	83.76	83.77±0.3	66.86	**70.02**±0.1
DualT	57.46	**61.82**±0.7	70.18	69.99±0.4
TVR	77.34	**79.37**±0.1	67.18	**69.44**±0.1
CausalNL	86.08	**86.29**±0.0	68.31	**69.90**±0.2

Table 2. Classification accuracy on *Food-101* and *Clothing1M*. w.o. means without and w. means with. Experiments are replicated over 5 times. **Bolded** text denotes improved performance with statistical significance.



(a) $\hat{y}$ (b) y^* (c) $\hat{y} \rightarrow y$ (d) $y^* \rightarrow y$

Figure 4. t-SNE mapping view of our latent $q_\phi(y|\hat{y}, x)$ (a, b) and corresponding confusion matrix (c, d). Colors of t-SNE views and columns of the confusion matrix represent $\hat{y}$ and y^*, respectively.

class information of a sample.

Figure 4c and 4d illustrates the confusion matrix, each comparing the prediction of a classifier and the result after calibration with NPC to the true label. Comparing the two figures, we can see the intrusion cases of $\hat{y}$ in a cluster of y, which indicates the misclassified result from the original classifier[6], become fewer with the calibrated y^* from NPC. In other words, with darker diagonal colours, we analyze that y^* is more similar to true y than $\hat{y}$.

By relating t-SNE figures and the confusion matrix together, we can implicitly find out the meaning of color change of dots from Figure 4a to Figure 4b. Black dashed boxes show visible differences between $\hat{y}$ and y^*. For example, red dots ($\hat{y} = 0$) of the right side in Figure 4a is wrongly classified as the confusion matrix in Figure 4c. On the contrary, the labels

[6]Another thing to focus is that the average of the diagonal terms of the confusion matrix is larger than 0.6, indicating that $\hat{y}$ is at least better than the original noisy label, $\tilde{y}$.

of those samples have changed to orange color ($y^* = 1$), its true labels. These calibrations of classifier outputs are the reason behind the performance gain in Table 1 and 2.

4.4. NPC as a Post-processor

Our method, NPC, modifies predictions from the classifier after training. In this section, we first analyze the efficacy of NPC compared to other post-processing methods (4.4.1). Second, we discuss the similarities and differences between label correction framework and post-processing (4.4.2), We further discuss the empirical convergence of NPC from the iterative application (in the Appendix D.5).

4.4.1. Comparison with Post-processors

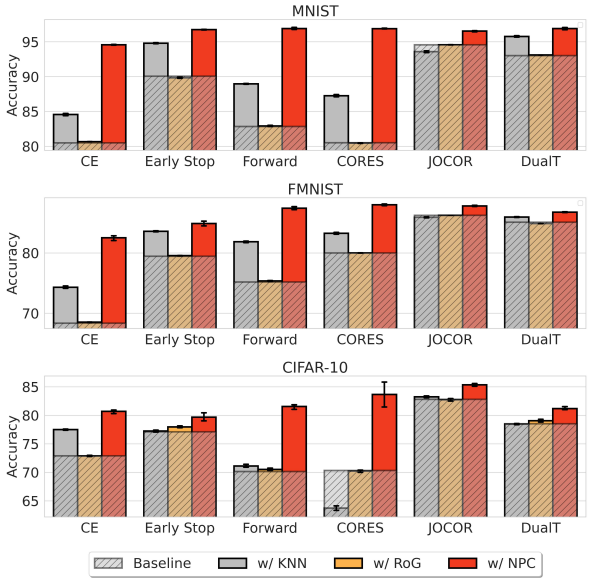


Figure 5. Classification accuracy comparison applied on various baseline methods. Results are averaged over five trials with standard deviations. We denote the baseline as the algorithm utilized for training classifier.

In this paper, we do not recognize the methods which optimize or re-train classifier parameters during the inference, as post-processors. In other words, we treat the the methods as a post-processors only when they manage the relationship between input features and labels, without modifying the classifier parameters.

KNN Prior KNN prior, which we utilize as a prior for inference of our posterior, could be recognized as a post-processor itself. Therefore, we compare the performance of KNN with NPC.

RoG (Lee et al., 2019) is a well-known post processing method for noisy label classification, as well. RoG assumes that samples with wrong labels are mostly outliers, so RoG removes outliers from the measured mahalanobis-distance (Lee et al., 2018), hypothesizing a classifier to be more robust to noisy labels.

Figure 5 indicates that NPC performs better than KNN and RoG on several baselines. Particularly, there is a significant gap between KNN and NPC, which emphasizes the importance of modeling posterior distribution via generative model, other than the prior setting on y. We also report the result on Food-101 and Clothing1M in Appendix D.3.

4.4.2. Label Correction and Post-processing

Label correction method, such as Joint (Tanaka et al., 2018), LRT (Zheng et al., 2020) and MLC (Zheng et al., 2021) have similarity with post-processing methods in that both utilizes the prediction of a given classifier to correct the noisy label. They are different, however, considering the classifier training: label correction methods iteratively correct the labels and train the classifier, which requires access to the classifier parameters. Post-processing methods are different from them because it does not entail classifier training. Having said that, by replacing noisy label $\tilde{y}$ with prediction $\hat{y}$ of a pre-trained classifier and training a new classifier, existing methods can be utilized as post-processor. Therefore, we focus on label correction methods and the utility of these methods as post-processor.

Method	Label Correction				Post-processing			
Noise	Joint	LRT	MLC	CauseNL	LRT*	MLC*	CauseNL*	NPC
SN	80.0±0.6	82.9±0.2	71.1±1.9	77.2±1.5	82.7±0.1	82.2±1.9	83.5±0.5	**85.3**±0.3
IDN	78.6±1.3	82.5±0.2	72.2±2.6	78.4±1.7	82.9±0.2	82.1±0.4	83.3±0.5	**84.8**±0.1

Table 3. Test accuracy on CIFAR-10 with different noise types. Noise ratio are 20%. Experiments are repeated over 5 times. For post-processing models, we utilize the prediction trained with Co-teaching method for fair comparison with CausalNL.

Table 3 shows the model performance of label correction methods and their applicability as post-processors. Baselines without and with an asterisk mean the original version and the post-processor of models, respectively. NPC shows best performance with statistical significance.

4.5. NPC as a Generative Model

4.5.1. What CausalNL and NPC focus from X

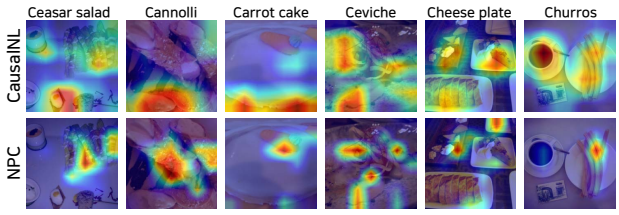


Figure 6. GradCAM results on Food-101. CausalNL tends to capture the whole image, while NPC focuses on the class-related features. Texts on each Column denotes the label of each image.

To understand what information each model focuses on from X to predict Y, we utilize GradCAM (Selvaraju et al., 2017). Figure 6 shows that NPC focuses on the class-related features while CausalNL also pays attention to other parts, which is driven by the generation of X. For example, to classify an image of *Churros* correctly, NPC mainly understands the part of features largely related to the *Churros* itself, while CausalNL captures both the *Churros* part and the *Chocolate* part to generate the image itself also well.

4.5.2. Failure Case Analysis

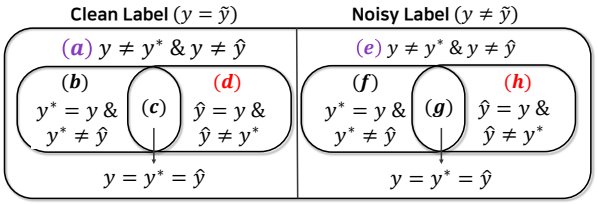


Figure 7. Venn diagram by relation between true class (y), noisy label ($\tilde{y}$), classifier model prediction ($\hat{y}$) and model prediction after post-processing (y^*)

Figure 7 is a Venn-Diagram to illustrate the count on gains and losses from post-processing. An analysis on four cases are required to evaluate the performance of a post-processor; 1) miss from classifier and miss from post-processor ($(a), (e)$), 2) miss from classifier and hit from post-processor ($(b), (f)$), 3) hit from classifier and hit from post-processor ($(c), (g)$), and 4) hit from classifier and miss from post-processor ($(d), (h)$). Reducing case 4) as small as possible while increasing case 2) would indicate the best case for post-processors. Table 4 shows the results on NPC and CausalNL*, which is the post-processor version of CausalNL.

	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
NPC	8	89	39799	86	9035	949	15	19
CausalNL*	39	58	32459	7426	2446	7538	31	3

Table 4. The number of samples on each region for Venn diagram in Figure 7. Results from CIFAR-10 with Symmetric Noise (20%)

Computing $\{(b)+(f)\}-\{(d)+(h)\}$, NPC and CausalNL* each shows 933, 167 counts, making NPC 5.59 times more effective than CausalNL*. Comparing cases on NPC and CausalNL* explains the difference of modeling nature between two methods. With NPC managing the relation of noisy prediction and the true label given X, NPC becomes a cautious corrector. However, CausalNL explores wider range of Y with X generation included in training objective, making CausalNL* a risk taking corrector.

4.6. NPC Identifies Potential Noises in Benchmarks

In this section, we provide the detection results of potentially noisy instances from the learning of NPC on benchmark datasets. We first capture all test samples whose label annotations and predictions from NPC are different. From them, we select 50 samples with highest prediction confidences and show 20 human-picked samples with marks of original annotations and predictions. From MNIST, we found out that images, which were captured by NPC, could be written differently from the original intention of annotators. Moreover, the samples from Fashion-MNIST are observed to have a possibility to be wrongly labeled, e.g. confusing features between Sneaker and Ankle boot.

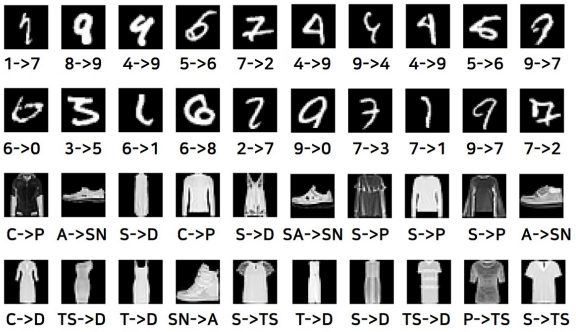


Figure 8. The selected samples of MNIST (upper two rows) and FMNIST (under two rows) from the set of instances whose annotations and predictions from NPC are different. The marks below images denote (label $\rightarrow$ prediction). The abbreviation for Fashion-MNIST means as follows: {C: Coat, P: Pullover, A: Ankle boot, SN: Sneaker, D: Dress, SA: Sandal, S: Shirt,TS: T-shirt/top}

5. Conclusion

In this paper, motivated by the possible failure of the classifier trained with noisy labels, we provide a novel method, Noisy Prediction Calibration (NPC), to calibrate noisy prediction from a classifier to a true label. By explicitly modelling the relation between the output of a classifier and the true label, NPC opens up new possibilities for formulating noisy label problems. With the provided experiments on several types of noise settings, NPC proves its effectiveness to improve the robustness of a classifier's prediction in various situations. Also, we believe that our methodological framework, which calibrates the prediction of given classifier, can be actively utilized on other fields of machine learning, e.g., long tailed recognition, domain adaptation.

Acknowledgements

This research was supported by AI Technology Development for Commonsense Extraction, Reasoning, and Inference from Heterogeneous Data (IITP) funded by the Ministry of Science and ICT (2022-0-00077).

References

Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pp. 233–242. PMLR, 2017.

Bahri, D., Jiang, H., and Gupta, M. Deep k-nn for noisy labels. In *International Conference on Machine Learning*, pp. 540–550. PMLR, 2020.

Berthon, A., Han, B., Niu, G., Liu, T., and Sugiyama, M. Confidence scores make instance-dependent label-noise learning possible. In *International Conference on Machine Learning*, pp. 825–836. PMLR, 2021.

Bossard, L., Guillaumin, M., and Van Gool, L. Food-101 – mining discriminative components with random forests. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T. (eds.), *Computer Vision – ECCV 2014*, pp. 446–461, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10599-4.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Chen, P., Ye, J., Chen, G., Zhao, J., and Heng, P.-A. Beyond class-conditional assumption: A primary attempt to combat instance-dependent label noise. *arXiv preprint arXiv:2012.05458*, 2020.

Cheng, H., Zhu, Z., Li, X., Gong, Y., Sun, X., and Liu, Y. Learning with instance-dependent label noise: A sample sieve approach. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=2VXyy9mIyU3.

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

Dosovitskiy, A. and Brox, T. Generating images with perceptual similarity metrics based on deep networks. *Advances in neural information processing systems*, 29, 2016.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Ghosh, A., Kumar, H., and Sastry, P. Robust loss functions under label noise for deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Han, B., Yao, J., Niu, G., Zhou, M., Tsang, I., Zhang, Y., and Sugiyama, M. Masking: A new perspective of noisy supervision. *Advances in neural information processing systems*, 31, 2018a.

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I. W., and Sugiyama, M. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*, 2018b.

Han, B., Niu, G., Yu, X., Yao, Q., Xu, M., Tsang, I., and Sugiyama, M. Sigua: Forgetting may make learning with noisy labels more robust. In *International Conference on Machine Learning*, pp. 4006–4016. PMLR, 2020.

Joo, W., Lee, W., Park, S., and Moon, I. Dirichlet variational autoencoder. *Pattern Recognit.*, 107:107514, 2020. doi: 10.1016/j.patcog.2020.107514. URL https://doi.org/10.1016/j.patcog.2020.107514.

Kim, T., Ko, J., Cho, s., Choi, J., and Yun, S.-Y. Fine samples for learning with noisy labels. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 24137–24149. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/ca91c5464e73d3066825362c3093a45f-Paper.pdf.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

LeCun, Y. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Lee, K., Lee, K., Lee, H., and Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.

Lee, K., Yun, S., Lee, K., Lee, H., Li, B., and Shin, J. Robust inference via generative classifiers for handling noisy labels. In *International Conference on Machine Learning*, pp. 3763–3772. PMLR, 2019.

Li, J., Socher, R., and Hoi, S. C. H. Dividemix: Learning with noisy labels as semi-supervised learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

OpenReview.net, 2020. URL https://openreview.net/forum?id=HJgExaVtwr.

Liu, S., Niles-Weed, J., Razavian, N., and Fernandez-Granda, C. Early-learning regularization prevents memorization of noisy labels. *Advances in Neural Information Processing Systems*, 33, 2020.

Lukasik, M., Bhojanapalli, S., Menon, A., and Kumar, S. Does label smoothing mitigate label noise? In *International Conference on Machine Learning*, pp. 6448–6458. PMLR, 2020.

Ma, X., Huang, H., Wang, Y., Romano, S., Erfani, S., and Bailey, J. Normalized loss functions for deep learning with noisy labels. In *International Conference on Machine Learning*, pp. 6543–6553. PMLR, 2020.

Malach, E. and Shalev-Shwartz, S. Decoupling "when to update" from "how to update". In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 960–970, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/58d4d1e7b1e97b258c9ed0b37e02d087-Abstract.html.

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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1944–1952, 2017.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

Song, H., Kim, M., Park, D., Shin, Y., and Lee, J.-G. Learning from noisy labels with deep neural networks: A survey. *arXiv preprint arXiv:2007.08199*, 2020.

Tanaka, D., Ikami, D., Yamasaki, T., and Aizawa, K. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5552–5560, 2018.

Wang, X., Hua, Y., Kodirov, E., Clifton, D. A., and Robertson, N. M. Proselflc: Progressive self label correction for training robust deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 752–761, 2021.

Wang, Y., Jha, S., and Chaudhuri, K. Analyzing the robustness of nearest neighbors to adversarial examples. In *International Conference on Machine Learning*, pp. 5133–5142. PMLR, 2018.

Wang, Y., Ma, X., Chen, Z., Luo, Y., Yi, J., and Bailey, J. Symmetric cross entropy for robust learning with noisy labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 322–330, 2019.

Wei, H., Feng, L., Chen, X., and An, B. Combating noisy labels by agreement: A joint training method with co-regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13726–13735, 2020.

Welinder, P., Branson, S., Perona, P., and Belongie, S. The multidimensional wisdom of crowds. In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper/2010/file/0f9cafd014db7a619ddb4276af0d692c-Paper.pdf.

Xia, X., Liu, T., Han, B., Gong, C., Wang, N., Ge, Z., and Chang, Y. Robust early-learning: Hindering the memorization of noisy labels. In *International Conference on Learning Representations*, 2020a.

Xia, X., Liu, T., Han, B., Wang, N., Gong, M., Liu, H., Niu, G., Tao, D., and Sugiyama, M. Part-dependent label noise: Towards instance-dependent label noise. *Advances in Neural Information Processing Systems*, 33, 2020b.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Xiao, T., Xia, T., Yang, Y., Huang, C., and Wang, X. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2691–2699, 2015.

Yao, Y., Liu, T., Han, B., Gong, M., Deng, J., Niu, G., and Sugiyama, M. Dual t: Reducing estimation error for transition matrix in label-noise learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7260–7271. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/512c5cad6c37edb98ae91c8a76c3a291-Paper.pdf.

Yao, Y., Liu, T., Gong, M., Han, B., Niu, G., and Zhang, K. Instance-dependent label-noise learning under a structural causal model. *Advances in Neural Information Processing Systems*, 34, 2021.

Yi, K. and Wu, J. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7017–7025, 2019.

Yu, X., Han, B., Yao, J., Niu, G., Tsang, I., and Sugiyama, M. How does disagreement help generalization against label corruption? In *International Conference on Machine Learning*, pp. 7164–7173. PMLR, 2019.

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, 2021a.

Zhang, Y., Niu, G., and Sugiyama, M. Learning noise transition matrix from only noisy labels via total variation regularization. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12501–12512. PMLR, 18–24 Jul 2021b. URL https://proceedings.mlr.press/v139/zhang21n.html.

Zhang, Z. and Sabuncu, M. R. Generalized cross entropy loss for training deep neural networks with noisy labels. In *32nd Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

Zheng, G., Awadallah, A. H., and Dumais, S. Meta label correction for noisy label learning. *AAAI 2021*, 2021.

Zheng, S., Wu, P., Goswami, A., Goswami, M., Metaxas, D., and Chen, C. Error-bounded correction of noisy labels. In *International Conference on Machine Learning*, pp. 11447–11457. PMLR, 2020.

A. Previous Researches

Here, we explain the details on the previous researches for learning with noisy labels which were not included in the main paper.

A.1. Previous Researches for Learning With Noisy Labels

A.1.1. Extraction of Reliable Clean Samples

(Han et al., 2018b) is one of the studies which assumed that samples with small losses are to be clean. The problem of small loss criteria is that, however, the deep neural network will overfit to small loss samples in earlier learning stage, and it will cause learning bias. To prevent this problem, (Han et al., 2018b) use two identical structured networks with different initial point. Then it chooses small loss samples from each network and exchanges them with peer network for updating the parameters. With the increase of training epochs, however, two networks have converged to a consensus gradually. Therefore, (Yu et al., 2019) has been proposed. In this study, a sample can only be selected when its output from two different networks disagree and it has small loss. Although it solves the problem of two different networks' convergence to a same point, it selects very few examples to train classifiers, especially when the noise ratio is high. This phenomenon is reported in (Wei et al., 2020), and they relieve these limitations by updating two networks together, making the result of two networks become closer to true labels and peer network's.

A.1.2. Label Modification

(Tanaka et al., 2018) jointly optimizes both model parameter and label data, initialize network parameters and train with modified labels again. (Yi & Wu, 2019) adopts label probability distributions to supervise network learning and to update these distributions through back-propagation end-to-end in each epoch. (Li et al., 2020) removes labels of samples with large loss, considering the samples as unlabeled, and applying semi-supervised learning strategies. (Zheng et al., 2020) takes the likelihood ratio between the classifier's confidence on noisy label and its confidence on its own label prediction as its threshold to configure clean labeled dataset, and corrects the label into the prediction for samples with low likelihood ratio iteratively. (Zheng et al., 2021) uses meta learning and reweights samples depending on the cleanness of its label. (Wang et al., 2021) analyzes several types of label modification approaches and suggests label correction regularization hyperparameter depending both on learning time stage and confidence of a sample.

A.1.3. Robust Loss and Regularization

(Ghosh et al., 2017) propose that the mean absolute loss (MAE) is more robust to the noisy label. However, it can cause underfitting, meaning a classifier converge to a bad sub-optimal. (Zhang & Sabuncu, 2018) combine the advantages of the mean absolute loss (MAE) and the cross entropy loss (CE) to obtain a better loss function and presents a theoretical analysis of the proposed loss functions in the context of noisy labels. (Wang et al., 2019) analyze CE, propose that CE fail to learn all classes uniformly well when learning with noisy labels. They suggest that adding reverse cross entropy term to original cross entropy term makes more robust loss to noisy label.

(Liu et al., 2020) is based on the phenomenon that the deep neural networks trained with noisy labels make progress during the early learning stage before memorization occurs. Therefore, they added a regularization term that seeks to maximize the inner product between the model output and the targets, with the target same as the weighted results of previous epochs. (Xia et al., 2020a) distinguishes critical parameters from non-critical parameters by gradient value and use only critical parameters for fitting true labels, hoping to solve overparameterization problem.

A.2. Explanation on Recent Instance-dependent Transition Matrix Studies

A.2.1. PDN

(Xia et al., 2020b) approximates $T(x)$ by a weighted combination of *part-dependent* transition matrices, $\{P^k\}_{k=1}^r$. This estimation is elaborated as follows:

$$T_{i,j}(x) = \sum_{k=1}^{r} h_k(x) P_{i,j}^k \tag{11}$$

Here, $h_k(x)$ is a input-dependent scalar weight for P^k. However, this estimation requires another estimation of P^k relying on *anchor point*s, whose discrete selection becomes heuristic, i.e. selecting an instance with a confident output from noisy

classifier $\tilde{f}$.

A.2.2. CSIDN

(Berthon et al., 2021) introduces a new method based on *confidence score*, $p(y=i|\tilde{y}=i,x)$, and this method assumes that the confidence score is given. With observed $p(y=i|\tilde{y}=i,x)$, T is transformed as Eq. 12.

$$T_{i,j}(x) = \begin{cases} P(y=i|\tilde{y}=i,x)\frac{P(\tilde{y}=i|x)}{P(y=i|x)} & if\ j=i \\ P(\tilde{y}=j|\tilde{y}\neq i, y=i)(1-T_{i,i}(x)) & if\ j\neq i \end{cases} \tag{12}$$

Whereas this method provides an input-dependent T, an assumption of observing $p(y=i|\tilde{y}=i,x)$ may be unrealistic and potentially annotation intensive.

B. Reparameterization Trick for Dirichlet Distribution

Following (Joo et al., 2020), we consider the dirichlet distribution as a composition of gamma random variables as follows:

$$Dirichlet(x;\alpha) = \frac{\Gamma(\sum \alpha_k)}{\prod \Gamma(\alpha_k)} \prod x_k^{\alpha_k - 1},\ Gamma(x;\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} \tag{13}$$

where $\alpha_k, \alpha, \beta > 0$. In other words, if there are K independent random variables $X_k \sim \text{Gamma}(\alpha_k, \beta)$ with $\alpha_k, \beta > 0$ for $k = 1, \cdots, K$, we have $Y \sim \text{Dirichlet}(\alpha)$ where $Y_k = X_k / \sum X_i$. Note that β should be same for all X_k. In this way, we can calculate the kl divergence of two dirichlet distributions as kl divergence of two multi-gamma distributions, where $X = (X_1, \cdots, X_K) \sim \text{MultiGamma}(\alpha, \beta \cdot \mathbf{1}_K)$ represents vector of K independent gamma random variables $X_k \sim \text{Gamma}(\alpha_k, \beta)$ with $\alpha_k, \beta > 0$ for $k = 1, \cdots, K$.

Note that the derivative of a Gamma-like function $\frac{\Gamma(\alpha)}{\beta^\alpha}$ can be derived as follows:

$$\frac{d}{d\alpha}\frac{\Gamma(\alpha)}{\beta^\alpha} = \beta^{-\alpha}(\Gamma'(\alpha) - \Gamma(\alpha)\log\beta) = \int_0^\infty x^{\alpha-1} e^{-\beta x} \log x\, dx\ . \tag{14}$$

Using Eq. 14, the KL divergence can be written as follows:

$$\begin{aligned}
&\text{KL}(Q||P) = \int_{\mathcal{D}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})}\, d\mathbf{x} = \int_0^\infty \cdots \int_0^\infty \prod \text{Gamma}(\hat{\alpha}_k, \beta) \log \frac{\beta^{\sum \hat{\alpha}_k} \prod \Gamma^{-1}(\hat{\alpha}_k) e^{-\beta \sum x_k} \prod x_k^{\hat{\alpha}_k - 1}}{\beta^{\sum \alpha_k} \prod \Gamma^{-1}(\alpha_k) e^{-\beta \sum x_k} \prod x_k^{\alpha_k - 1}}\, d\mathbf{x} \\
&= \int_0^\infty \cdots \int_0^\infty \prod \text{Gamma}(\hat{\alpha}_k, \beta) \times \Big[\sum (\hat{\alpha}_k - \alpha_k) \log\beta + \sum \log\Gamma(\alpha_k) - \sum \log\Gamma(\hat{\alpha}_k) + \sum (\hat{\alpha}_k - \alpha_k) \log x_k \Big]\, d\mathbf{x} \\
&= \sum (\hat{\alpha}_k - \alpha_k) \log\beta + \sum \log\Gamma(\alpha_k) - \sum \log\Gamma(\hat{\alpha}_k) \\
&\quad + \int_0^\infty \cdots \int_0^\infty \frac{\beta^{\hat{\alpha}_k}}{\prod \Gamma(\hat{\alpha}_k)} e^{-\beta \sum x_k} \prod x_k^{\hat{\alpha}_k - 1} \big(\sum (\hat{\alpha}_k - \alpha_k) \log x_k \big)\, d\mathbf{x} \\
&= \sum (\hat{\alpha}_k - \alpha_k) \log\beta + \sum \log\Gamma(\alpha_k) - \sum \log\Gamma(\hat{\alpha}_k) + \sum (\hat{\alpha}_k - \alpha_k) \beta^{\hat{\alpha}_k} \Gamma^{-1}(\hat{\alpha}_k) \beta^{-\hat{\alpha}_k} \big(\Gamma'(\hat{\alpha}_k) - \Gamma(\hat{\alpha}_k) \log\beta \big) \\
&= \sum (\hat{\alpha}_k - \alpha_k) \log\beta + \sum \log\Gamma(\alpha_k) - \sum \log\Gamma(\hat{\alpha}_k) + \sum (\hat{\alpha}_k - \alpha_k) (\psi(\hat{\alpha}_k) - \log\beta) \\
&= \sum \log\Gamma(\alpha_k) - \sum \log\Gamma(\hat{\alpha}_k) + \sum (\hat{\alpha}_k - \alpha_k) \psi(\hat{\alpha}_k)
\end{aligned} \tag{15}$$

Now we can calculate the kl divergence of two multi-gamma distributions as follows:

$$\text{KL}(Q||P) = \sum \log\Gamma(\alpha_k) - \sum \log\Gamma(\hat{\alpha}_k) + \sum (\hat{\alpha}_k - \alpha_k) \psi(\hat{\alpha}_k)\ , \tag{16}$$

where $P = \text{MultiGamma}(\alpha, \beta \cdot \mathbf{1}_K)$ and $Q = \text{MultiGamma}(\hat{\alpha}, \beta \cdot \mathbf{1}_K)$.

C. Alignment of T and H

C.1. Proof of Proposition 3.1

Proposition C.1. *Assume that $\widehat{y}$ and y are conditional independent given $\widetilde{y}$, and $p(\widehat{y}=k|x) \neq 0$ for all $k=1,..,c$. Then, $H_{kj}(x) = \frac{p(y=j|x)}{p(\widehat{y}=k|x)} \sum_i p(\widehat{y}=k|\widetilde{y}=i,x) T_{ij}(x)$ for all $j,k=1,...,c$.*

Proof.

$$
\begin{aligned}
p(y=j|\hat{y}=k,x) &= \sum_i p(y=j,\tilde{y}=i|\hat{y}=k,x) && (17)\\
&= \sum_i \frac{p(y=j,\tilde{y}=i,\hat{y}=k|x)}{p(\hat{y}=k|x)} && (18)\\
&= \sum_i \frac{p(\hat{y}=k|\tilde{y}=i,y=j,x)p(\tilde{y}=i|y=j,x)p(y=j|x)}{p(\hat{y}=k|x)} && (19)\\
&= \frac{p(y=j|x)}{p(\tilde{y}=i|x)} \sum_i p(\hat{y}=k|\tilde{y}=i,x)p(\tilde{y}=i|y=j,x) \quad (\because \widehat{y} \perp\!\!\!\perp y|\widetilde{y}) && (20)\\
& && (21)
\end{aligned}
$$

Therefore, $H_{kj}(x) = \frac{p(y=j|x)}{p(\widehat{y}=k|x)} \sum_i p(\widehat{y}=k|\widetilde{y}=i,x) T_{ij}(x)$ for all $j,k=1,...,c$.

□

The assumption of $\hat{y} \perp\!\!\!\perp y|\tilde{y}$ is natural because $\hat{y}$ is conditionally independent to y when $\hat{\psi}$ is trained only with $\tilde{y}$. Proposition 3.1 shows that H can be formulated by T with weighting variables, which are observable or can be easily computed from our framework. It implies that we can also infer T from the inference procedure of H, which reduces the framework of NPC to the transition-based approaches.

C.2. Experiment on the Relation between T and H

C.2.1. Experiment Details

For $p(\hat{y}|x)$, we use same network as in Section 4. For finding $p(y|x)$, we use our model output, which is $\sum_{k=1}^{c} p(y|\hat{y}=k,x)p(\hat{y}=k|x)$. What is left is then $p(\hat{y}|\tilde{y},x)$ term. Since all variables for the probability calculation is given as datasets, we can make a function that approximates this probability with another deep neural network. This network has similar structure with the networks used in Section 4. The only difference is $\tilde{y}$ is to be concatenated with feature of x. For other implementation details, such as batch size, learning rate, we utilize same condition as in Section 4.

In the concept of instance-dependent noisy label generation model, each transition matrix of all different samples is assumed not to be identical. As a method to merge instance-wise different transition matrices as a single representative value, we approximate the class-wise transition matrix with monte-carlo estimation, which is calculated as Eq. 22.

$$
\begin{aligned}
T_{ij} &= P(\tilde{y}=j|y=i)\\
&= \sum_x P(\tilde{y}=j,x|y=i)\\
&= \sum_x P(\tilde{y}=j|y=i,x)P(x|y=i)\\
&\approx \frac{1}{N_i} \sum_x P(\tilde{y}=j|y=i,x),\ \textit{where } N_i = \textit{total number of samples of class } i
\end{aligned} \tag{22}
$$

C.2.2. Additional Results on Transition Matrix Estimation

We report MSE between the transition matrix approximated from each algorithm and the true transition matrix in Table 5 for CIFAR-10 dataset with various noisy label conditions. The result shows that NPC approximates the transition matrix as good as the traditional algorithms.

Noise	Ratio	Forward	DualT	TVR	CausalNL	NPC
SN	20	0.0018	0.0012	0.0018	0.0035	0.0015
	80	0.0004	0.0576	0.0005	0.0195	0.0005
IDN	20	0.0038	0.0021	0.0024	0.0034	0.0035
	40	0.0041	0.0043	0.0032	0.0049	0.0018

Table 5. MSE between the approximated transition matrix and the true one.

D. Experiment

Here, we manage dataset explanation, noise label generation process, experiment settings and additional result.

D.1. Implementation Details and Baseline Description

D.1.1. Synthetic Noisy Label Generation Process

MNIST (LeCun, 1998) and *Fashion-MNIST* (Xiao et al., 2017) are both 28×28 grayscale image datasets with 10 classes, which include 60,000 training samples and 10,000 test samples. *CIFAR-10* (Krizhevsky et al., 2009) is $32 \times 32 \times 3$ color image dataset with 10 classes, which includes 50,000 training samples and 10,000 test samples. Since these datasets are assumed to have no noisy labels, we manage four types of noisy label for noisy label injection. Here, we explain details of those noisy label generation processes. Since we explain enough for symmetric noise in Section 4, we pass explanation on it.

Asymmetric Noise (ASN) (Han et al., 2018b; Tanaka et al., 2018; Xia et al., 2020a) For this type of noise, we flipped label class as below, following the previous researches.

- MNIST : $2 \Rightarrow 7, 3 \Rightarrow 8, 5 \Leftrightarrow 6$
- FMNIST : $T-shirt \Rightarrow Shirt, Pullover \Rightarrow Coat, Sandals \Rightarrow Sneaker$
- CIFAR-10 : $Truck \Rightarrow Automobile, Bird \Rightarrow Airplane, Deer \Rightarrow Horse, Cat \Leftrightarrow Dog$

Instance Dependent Noise (IDN) We followed noise generation process as utilized at (Xia et al., 2020b; Cheng et al., 2021; Yao et al., 2021).

Algorithm 1 Instance Dependent Noise Generation Process

Require: Clean samples $(x_i, y_i)_{i=1}^n$;, Noise rate τ
1: Sample instance flip rates $q \in \mathbb{R}^n$ from the truncated normal distribution $N(\tau, 0.1^2, [0, 1])$;
2: Independently sample $w_1, w_2, ..., w_c$ from the standard normal distribution $N(0, 1^2)$;
3: **for** $i = 1, 2, ..., n$ **do**
4: $p = x_i \times w_{y_i}$;
5: $p_{y_i} = -\inf$;
6: $p = q_i \times softmax(p)$;
7: $p_{y_i} = 1 - q_i$;
8: Randomly choose a label from the label space according to the possibilities p as noisy label $\bar{y}_i$;
9: **end for**
Output: Noisy samples $(x_i, \bar{y}_i)_{i=1}^n$

Similarity Reflected Instance Dependent Noise (SRIDN) We followed noise generation process as utilized at (Chen et al., 2020; Berthon et al., 2021).

We only perform normalization for CIFAR-10.

Algorithm 2 Similarity Reflected Instance Dependent Noise Generation Process

Require: Clean samples $(x_i, y_i)_{i=1}^n$;, Noise rate τ
1: Train a classifier;
2: Get output from a classifier $f_i \in \mathbb{R}^c$ for all $i = 1, ..., n$ and sort by $f_i^{y_i}$;
3: Set $N_{noisy} = 0$;
4: **while** $N_{noisy} < n \times \tau$ **do**
5: From the least confident sample, choose noisy label $\bar{y}_i = max_{j \neq y_i, j \in 1,...,c} f_i^j$;
6: $N_{noisy} = N_{noisy} + 1$;
7: **end while**
Output: Noisy samples $(x_i, \bar{y}_i)_{i=1}^n$

D.1.2. Real Datasets with Noisy Labels

Food101 (Bossard et al., 2014) is color image datasets with 101 food categories, each category having 1,000 samples each. For each class, 250 images are annotated by humans, consisting of test dataset and 750 images are provided with real-world label noise. *Clothing-1M* (Xiao et al., 2015) is another real-world noisy label dataset collected from several online shopping websites. The data contains 1 million training images with 14 classes. We use the total 75,000 human annotated images only for testing. All datasets have been widely used in the previous researches (Wei et al., 2020; Xia et al., 2020a; Cheng et al., 2021; Yao et al., 2021). For Food101 and Clothing1M, we resize the image to 256×256, crop the middle 224×224 as input, and perform normalization.

D.1.3. Implementation Details

For fair comparison, we implement same network structures for all baseline methods. In details, we use network with two convolution layers for MNIST and Fashion-MNIST, 9-layered convolutional neural network for CIFAR-10 as in (Han et al., 2018b; Yu et al., 2019). For real-world dataset, Food101 and Clothing-1M, we use a ResNet-50 network pre-trained on ImageNet (Deng et al., 2009).

CNN on MNIST & Fashion-MNIST	CNN on CIFAR-10
28×28	$32 \times 32 \times 3$
	3×3 conv, 128LRELU
	3×3 conv, 128LRELU
	3×3 conv, 128LRELU
	2×2 max-pool, stride 2
	dropout, $p = 0.25$
3×3 conv, 8RELU	3×3 conv, 256LRELU
3×3 conv, 16Tanh	3×3 conv, 256LRELU
dense $16 \times 28 \times 28 \rightarrow 28 \times 28$	3×3 conv, 256LRELU
dense $28 \times 28 \rightarrow 256$	2×2 max-pool, stride 2
	dropout, $p = 0.25$
	3×3 conv, 512LRELU
	3×3 conv, 256LRELU
	3×3 conv, 128LRELU
	avg-pool
dense $256 \rightarrow 10$	dense $128 \rightarrow 10$

Table 6. Convolutional Neural Network Structure for MNIST, Fashion-MNIST and CIFAR-10 datasets.

We train all synthetic datasets with batch size 128 and all real-world datasets with batch size 32. For all datasets, we use Adam optimizer with learning rate of 10^{-3} and no learning rate decay is applied. All methods are implemented by PyTorch.

D.1.4. Baseline Description

We compare the proposed method with the following approaches.

- **CE** It trains the deep neural networks with the cross entropy loss on noisy datasets.
- **Joint** (Tanaka et al., 2018) It jointly optimizes the network parameters and the sample labels. We set α and β as 1.0 and 0.5 respectively.
- **Coteaching** (Han et al., 2018b) It trains two different networks and the samples with small loss are only fed to the other network for learning.
- **JoCoR** (Wei et al., 2020) It also trains two networks and selects the samples, for which the sum of the losses from two networks is small, as clean samples. Following authors of the original paper, we set λ as 0.5, and set other hyperparameter settings as cited on the paper.
- **CORES2** (Cheng et al., 2021) Regarding the prediction output of a network, which is trained on noisy dataset, as a confidence of each instance, it selects data instances with high confidence as clean instances. We follow hyperparameter settings as cited on the paper.
- **SCE** (Wang et al., 2019) It trains network based on the specialized loss function, which is sum of cross entropy and reverse cross entropy. We follow hyperparameter settings as cited on the paper.
- **ES (EarlyStop)** Splitting a part of noisy training dataset as validation dataset, it only learns network until the validation performance continues to decrease.
- **LS (Label Smoothing)** (Lukasik et al., 2020) It trains network based on the noisy dataset where each label is expressed as a smoothed label by a given label smoothing factor rather than one-hot encoding. We set smoothing factor as 0.1.
- **REL** (Xia et al., 2020a) To avoid the over-parameterization of the network, it only trains critical parameters, whose gradients are high, of network.
- **Forward** (Patrini et al., 2017) It trains network based on the loss function, which is modified by the estimated transition probability matrix.
- **DualT** (Yao et al., 2020) In order to reduce the estimation error of the transition probability matrix, the corresponding matrix is expressed as a product of two matrices, and each matrix is estimated.
- **TVR** (Zhang et al., 2021b) To solve the problem of infinitely many possible solutions of the transition probability matrix, it minimizes the total variance distance.
- **CausalNL** (Yao et al., 2021) We already explained on this research previously in Section 2.3.

D.2. Performance on MNIST Dataset

Due to space issue, we show test accuracy on MNIST dataset with several noise conditions here. In most noise settings, our proposed method increases model performance, calibrating noisy prediction to true label.

Model	Clean	SN		ASN		IDN		SRIDN	
	-	20	80	20	40	20	40	20	40
CE	97.8	86.3	29.7	89.2	81.3	80.5	66.3	82.9	65.4
w/NPC	**98.2**	**95.3**	**36.6**	**96.3**	**92.3**	**94.6**	**89.1**	**85.2**	**70.3**
Joint	93.0	93.9	**22.0**	93.8	93.2	93.1	93.6	87.1	81.1
w/NPC	**94.5**	**95.5**	21.9	**95.2**	**94.9**	**94.8**	**95.5**	**89.1**	**84.4**
Coteaching	98.0	91.8	69.6	97.9	97.5	90.7	87.5	89.0	81.3
w/NPC	**98.3**	**95.0**	**74.8**	**98.2**	**97.9**	**94.9**	**93.3**	**91.0**	**84.4**
JoCoR	97.8	95.0	**36.6**	**97.5**	**95.3**	94.6	93.3	90.2	82.2
w/NPC	**98.3**	**96.4**	36.3	97.3	94.6	**96.9**	**96.1**	**91.9**	**85.2**
CORES2	97.0	75.3	16.6	88.5	**9.8**	80.5	48.8	86.5	68.5
w/NPC	**97.9**	**86.9**	**23.9**	**94.5**	**9.8**	**90.7**	**67.4**	**89.4**	**78.5**
SCE	97.7	85.7	30.5	89.4	81.3	80.6	66.6	83.3	65.8
w/NPC	**98.2**	**95.1**	**38.5**	**96.2**	**92.5**	**94.5**	**88.8**	**85.3**	**70.7**
Early Stop	96.5	95.9	57.8	94.9	84.4	90.1	73.3	85.8	71.4
w/NPC	**97.9**	**97.1**	**75.0**	**97.4**	**89.9**	**96.6**	**90.6**	**88.6**	**76.5**
LS	97.8	85.8	32.6	88.9	82.4	80.6	66.2	84.3	65.3
w/NPC	**98.2**	**95.1**	**40.6**	**96.0**	**92.8**	**94.4**	**88.7**	**86.2**	**70.8**
REL	**98.0**	96.6	79.4	93.8	94.2	95.5	90.7	90.4	86.2
w/NPC	97.9	**96.9**	**84.9**	**96.8**	**96.4**	**96.8**	**95.5**	**91.7**	**88.7**
Forward	98.0	88.1	27.8	91.8	82.1	82.9	67.9	84.5	67.7
w/NPC	**98.4**	**95.6**	**39.4**	**97.9**	**96.9**	**96.1**	**91.1**	**86.9**	**74.4**
DualT	96.7	93.6	**9.8**	95.9	89.7	93.0	94.3	86.6	72.8
w/NPC	**97.8**	**96.3**	**9.8**	**97.7**	**93.1**	**95.9**	**96.5**	**89.3**	**77.3**
TVR	97.7	84.7	31.8	86.7	80.0	75.2	64.4	83.7	66.3
w/NPC	**98.1**	**94.4**	**38.9**	**95.4**	**91.0**	**91.6**	**84.5**	**85.7**	**71.3**
CausalNL	98.1	94.1	57.1	**98.3**	97.4	92.7	85.2	84.5	68.9
w/NPC	**98.4**	**96.9**	**63.5**	**98.3**	**98.1**	**97.0**	**94.5**	**89.0**	**75.9**

Table 7. MNIST test accuracy on all noise conditions trained by several baseline classifiers and after post-processed with NPC. **Bolded** with accuracy gain. Similar to the main paper, all experiments are replicated over five times with different random seeds. Reported results are mean value.

D.3. Comparison with Post-processors: Results on Real Dataset

We compare the test accuracy of NPC and other post-processing method applicable after finishing training a classifier on Food-101 and Clothing-1M. We show NPC works best of all at Table 8.

Dataset	Food101				Clothing1M			
Method	Classifier	KNN	RoG	NPC	Classifier	KNN	RoG	NPC
CE	78.37	67.25±0.2	78.38±0.1	**80.21**±0.2	68.14	69.38±0.1	68.05±0.1	**70.83**±0.1
Early Stop	73.22	68.37±0.2	75.87±0.2	**76.80**±0.3	67.07	68.76±0.0	68.25±0.1	**70.21**±0.1
SCE	75.23	64.72±0.3	77.49±0.1	**78.26**±0.3	67.77	69.34±0.1	67.69±0.2	**70.36**±0.1
REL	78.96	78.90±0.1	**83.79**±0.1	78.95±0.4	62.53	63.49±0.0	66.12±0.3	**64.83**±0.1
Forward	83.76	83.68±0.0	82.02±0.1	83.77±0.3	66.86	69.40±0.1	66.71±0.2	**70.02**±0.1
DualT	57.46	52.87±0.2	60.82±0.1	**61.82**±0.7	70.18	69.05±0.1	69.46±0.1	69.99±0.4
TVR	77.34	65.98±0.3	77.20±0.1	**79.37**±0.1	67.18	68.27±0.2	67.55±0.1	**69.44**±0.1
CausalNL	86.08	85.14±0.1	85.93±0.1	**86.29**±0.0	68.31	68.08±0.1	68.37±0.1	**69.90**±0.2

Table 8. Test accuracy after training the classifier with cross entropy loss, training KNN algorithm on representations, applying RoG, and applying NPC for *Food101* (Food) and *Clothing-1M* (Clothing).

D.4. Flexibility of Prior Modeling for NPC

As mentioned in Section 3.2.3 of the main paper, we designed a prior distribution of the latent variable y depending on x as Eq. 9 using predictions from KNN algorithm. However, the shape of a prior distribution does not have to be one-specific dimension sharp and all other dimensions flat. In this section, we report experiment results with more flexible modeling of a prior distribution as Eq. 23. Here, unlike the main page, we define $\bar{Y}$ as the subset of the label set with $|\bar{Y}|$ possible to be larger than 1 and p as the prediction probability from KNN.

$$\alpha_x^k = \begin{cases} \delta & k \neq \bar{y} \\ \delta + \rho \times p(\bar{y}) & k = \bar{y} \end{cases} \quad \text{for } \bar{y} \in \bar{Y},\ k = 1, ..., c \tag{23}$$

Increasing flexibility of the prior may result in either accuracy increase or decrease; it can implicitly model information on the relations of classes by specifying the probability of a sample being assigned to each class, or inject noise and lose class information. We experiment the difference of a prior distribution for CIFAR-10 dataset with several noise conditions. Table 9 reports the test accuracy of NPC with its original prior and with the flexible version. We find out that TOP2 prior tends to work better than the original one with more severe noises or ASN or IDN noise condition. Since the prior distribution represents the information given from the input (x) for classification, we see the possibility of model development by modeling improved prior.

Model	**Clean**	**SN**		**ASN**		**IDN**		**SRIDN**	
	-	20	80	20	40	20	40	20	40
CE	**89.0**	**80.8**	17.0	84.7	78.8	**80.9**	59.9	**74.3**	**64.3**
w/ TOP2	80.0	79.2	**18.0**	**85.3**	**80.4**	80.3	**64.6**	74.1	63.4
Joint	**84.4**	**80.2**	8.3	**83.0**	**77.7**	80.7	69.1	**72.0**	**63.6**
w/ TOP2	83.9	80.0	**8.4**	82.4	77.3	**81.1**	**77.0**	71.3	62.8
Coteaching	89.2	**85.3**	**32.1**	87.1	76.8	**84.8**	**78.5**	**76.1**	**67.2**
w/ TOP2	**89.3**	83.9	32.0	**87.3**	**77.4**	83.8	**78.5**	75.9	67.0
JoCoR	**89.3**	**86.0**	27.0	**85.1**	79.0	**85.8**	80.1	75.9	66.7
w/ TOP2	**89.3**	85.3	**27.4**	85.0	**80.0**	85.5	**80.5**	**76.1**	**66.9**
SCE	87.4	75.0	15.2	81.5	75.2	75.4	55.6	72.9	62.5
w/ TOP2	**89.0**	**79.2**	**18.0**	**85.3**	**80.4**	**80.3**	**64.6**	**74.1**	**63.4**
Early Stop	**84.0**	82.5	**18.2**	81.2	72.0	79.4	65.1	**72.1**	**63.0**
w/ TOP2	78.4	**82.7**	17.9	**81.3**	**72.5**	**79.7**	**65.7**	71.7	62.8
LS	**89.0**	**80.8**	15.5	84.7	78.8	**80.9**	59.9	**74.3**	**64.3**
w/ TOP2	**89.0**	79.2	**18.0**	**85.3**	**80.4**	80.3	**64.6**	74.1	63.4
REL	**83.4**	78.6	**26.0**	**75.9**	**76.1**	**78.5**	**51.2**	**70.7**	64.2
w/ TOP2	81.6	77.0	25.6	74.6	75.6	77.0	51.1	70.2	**64.3**
Forward	**88.7**	81.5	17.2	83.8	74.5	80.3	63.3	**74.8**	65.0
w/ TOP2	86.9	**82.8**	**20.8**	**85.2**	**78.3**	**83.0**	**76.4**	73.9	**65.3**
DualT	**86.0**	83.0	**8.4**	**83.0**	77.5	81.0	**77.3**	**70.1**	**64.0**
w/ TOP2	**86.0**	**83.1**	8.3	82.0	**78.1**	**81.1**	**77.3**	70.0	63.8
TVR	88.3	**80.8**	15.7	84.1	76.5	**80.8**	60.7	**74.5**	**64.5**
w/ TOP2	**88.4**	78.5	**17.0**	**84.3**	**79.3**	79.7	**65.0**	74.3	63.6
CausalNL	89.7	81.2	18.8	85.0	74.8	81.2	71.9	75.3	63.9
w/ TOP2	**90.6**	**81.4**	**19.0**	**85.7**	**75.4**	**81.5**	**72.5**	**75.8**	**64.0**

Table 9. Test accuracy for CIFAR-10 datasets with various noisy label types. We demonstrate average performances with NPC of one-dimension sharp prior and its smoothed version (w/TOP2). TOP2 represents the dimension with largest probability and the second-best one become sharp with its value. The experimental results are averaged value over five trials. **Bolded** text denotes better performance.

D.5. Repetitive Application of NPC

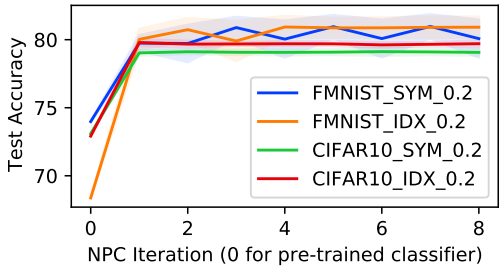


Figure 9. Iterative learning of NPC on CE for several datasets.

NPC calibrates the classifier prediction toward the true label as a post-processor. Having said that, It arises a new question : With the repetitive application of NPC, would the performance of given model gradually improve and finally torch the 100% test accuracy, or the performance would converge to some extent? If it converges, when will it do? We implemented this experimental setting by designing the repetitive application of NPC, which iteratively utilizes calibrated prediction of NPC from previous iterations. Figure 9 shows that the NPC performance converges after the first deployment. It implies that NPC with a single iteration has already utilized enough information from the classifier to model latent true label.

D.6. Extensive Comparison between CausalNL and NPC

Ratio	CausalNL	NPC
20%	81.81	**82.91** $\pm$ 0.1
40%	77.01	**78.83** $\pm$ 0.4

Table 10. Test accuracy on CIFAR-10 with IDN noise

On the main page, we reported model accuracy of CausalNL lower than the one reported in the original paper (Yao et al., 2021). We analyzed why this gap happened, found out that 1) the difference of backbone structure (9-layer CNN for ours, Resnet34 for the original paper) and 2) the difference of data normalization have affected the performance. we experiment CausalNL model under the same condition as the original authors did, and reproduce the performance as table 10. Still, NPC shows better performance than CausalNL.

D.7. Time Complexity Comparison on T and H

Our next question is: How long does it take on estimation of T and H to converge? We experimented it on MNIST Instance dependent 20 % noise label dataset, and report the running time of each method on Table 11.

Model	Time
Forward	636.4425
DualT	3004.905
TVR	468.7727
CausalNL	4165
NPC	28.2169

Table 11. Model and Time spent to converge