ROBUST DETERMINANTAL GENERATIVE CLASSIFIER FOR NOISY LABELS AND ADVERSARIAL ATTACKS

Anonymous authors
Paper under double-blind review

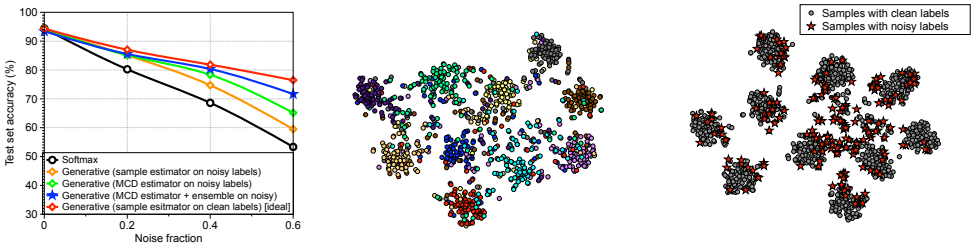
ABSTRACT

Large-scale datasets may contain significant proportions of noisy (incorrect) class labels, and it is well-known that modern deep neural networks poorly generalize from such noisy training datasets. In this paper, we propose a novel inference method, *Deep Determinantal Generative Classifier (DDGC)*, which can obtain a more robust decision boundary under any softmax neural classifier pre-trained on noisy datasets. Our main idea is inducing a generative classifier on top of hidden feature spaces of the discriminative deep model. By estimating the parameters of generative classifier using the minimum covariance determinant estimator, we significantly improve the classification accuracy, with neither re-training of the deep model nor changing its architectures. In particular, we show that DDGC not only generalizes well from noisy labels, but also is robust against adversarial perturbations due to its large margin property. Finally, we propose the ensemble version of DDGC to improve its performance, by investigating the layer-wise characteristics of generative classifier. Our extensive experimental results demonstrate the superiority of DDGC given different learning models optimized by various training techniques to handle noisy labels or adversarial examples. For instance, we improve the test accuracy of DenseNet on CIFAR-10 datasets with 60% noisy labels from 53.34% to 74.72%.

1 INTRODUCTION

Deep neural networks (DNNs) are known to generalize well when they are trained on large-scale datasets with clean label annotations. For example, DNNs have achieved state-of-the-art performance on many classification tasks, e.g., speech recognition (Amodei et al., 2016), object detection (Girshick, 2015) and image classification (He et al., 2016). However, as the scale of the training dataset increases, it becomes infeasible to obtain all class labels from domain experts. A common practice is collecting the class labels from data mining on social media (Mahajan et al., 2018) and web data (Krause et al., 2016). However, they may contain missing/noisy (incorrect) labels, and recent studies have shown that modern deep architectures may generalize poorly from the noisy datasets (Zhang et al., 2017; Arpit et al., 2017). For example, in Figure 1(a), the test set accuracy (black line) of DenseNet-100 model (Huang & Liu, 2017) trained on the CIFAR-10 dataset (Krizhevsky & Hinton, 2009) significantly decreases as the noise fraction increases.

To overcome the poor generalization issue of DNNs against noisy labels, several training strategies have been investigated in the literature (Reed et al., 2014; Patrini et al., 2017; Ma et al., 2018b; Hendrycks et al., 2018). Reed et al. (2014) proposed a bootstrapping method which trains deep models with new labels generated by a convex combination of the raw (noisy) labels and their predictions, and Ma et al. (2018b) improved the bootstrapping method by utilizing the dimensionality of subspaces during training. Patrini et al. (2017) modified the loss and posterior distribution to eliminate the influence of noisy labels, and Hendrycks et al. (2018) improved such a loss correction method by utilizing the information from data with true class labels. However, adopting such training methods might incur expensive back-and-forth costs, e.g., additional time and hyperparameter tuning. This motivates our approach of developing a more plausible inference method which can be applied to any pre-trained deep model. Nevertheless, our direction is orthogonal to the prior works, where one can also combine them for even better performance (see Table 3 in Section 3).



(a) Test set accuracy comparison by varying noise fraction

(b) Features on penultimate layer from test samples by t-SNE

(c) Features on penultimate layer from training samples by t-SNE

Figure 1: Experimental results under DenseNet-100 model and CIFAR-10 dataset. (a) Test accuracy comparison when the labels of a given proportion of training samples are flipped to other labels uniformly at random. Visualization of features on the penultimate layer using t-SNE from (b) test samples (same colors indicate same classes) and (c) training samples when the noise fraction is 20%.

Contributions. It has been observed that DNNs can learn meaningful feature patterns shared by multiple training examples even for datasets with noisy labels (Arpit et al., 2017). We also found that an induced generative classifier (Ng & Jordan, 2002) under linear discriminant analysis (LDA) assumption (with naive estimations on sample mean and covariance) built upon the hidden feature space can outperform the softmax classifier (orange line in Figure 1(a)). In Figure 1(b), the hidden features from test samples projected in a 2-dimensional space using t-SNE (Maaten & Hinton, 2008) are illustrated. Here, one can observe that all ten classes are well-separated in the embedding space, even though the model is trained under a noisy dataset. More importantly, Figure 1(c) plotting noisy training samples implies that outliers induce the class-wise multi-modal distributions in the feature space. Therefore, an LDA-like generative classifier assuming the class-wise unimodal distribution might be more robust, as a discriminative classifier is easier to be overfitted by outliers. This motivates our goal to induce a generative classifier on the pre-trained hidden features of DNNs.

To this end, we propose the so-called deep determinantal generative classifier (DDGC), based on the minimum covariance determinant (MCD) (Rousseeuw, 1984; Rousseeuw & Driessen, 1999) estimation on its parameters. While a naive sample estimator can be highly influenced by outliers, MCD estimator can improve the robustness by removing them. Note that MCD is known to have a near-optimal breakdown point (Hampel, 1971) of almost 50% in most situations, i.e., the number of outliers should be larger than that of normal samples to fool it. We further provide a new theoretical support on the larger margin property of DDGC such that not only does it generalize well from noisy labels (Durrant & Kabán, 2010), but also improves the robustness against adversarial perturbations (Pang et al., 2018). In addition, we observe that DNNs tend to have similar hidden features, regardless of whether they are trained with clean or noisy labels at early layers (Morcos et al., 2018), and DDGC built from low-level features can be often more effective. Under the observation, we finally propose an ensemble version of DDGC to incorporate all effects of low and high layers.

We demonstrate the effectiveness of DDGC using modern neural architectures, such as DenseNet (Huang & Liu, 2017) and ResNet (He et al., 2016) trained for image classification tasks including CIFAR (Krizhevsky & Hinton, 2009) and SVHN (Netzer et al., 2011). First, our methods (green and blue lines in Figure 1(a)) significantly outperform the softmax classifier, although they use the same feature representations trained by the noisy dataset. For example, we improve the test accuracy of DenseNet on CIFAR-10 datasets with 60% noisy labels from 53.34% to 74.72%. We also demonstrate that DDGC can be used to further improve various prior training methods (Reed et al., 2014; Patrini et al., 2017) which are specialized to handle the noisy environment. Finally, DDGC is shown to be robust against various adversarial attacks (Goodfellow et al., 2015; Moosavi Dezfooli et al., 2016; Carlini & Wagner, 2017). In particular, we improve the classification accuracy of ResNet against the DeepFool attack (Moosavi Dezfooli et al., 2016) on CIFAR-10 from 0.34% to 7.50%.

2 Robust inference via generative classifier

In this section, we propose a novel inference method which obtains the robust posterior distribution from any softmax neural classifier pre-trained on datasets with noisy labels. Our idea is inducing the generative classifier using fixed features from any pre-trained model. We show the advantages of our method in terms of high breakdown points (Hampel, 1971), generalization error (Durrant &

Kabán, 2010) and adversarial robustness (Pang et al., 2018). We also investigate the layer-wise characteristics of generative classifiers, and introduce an ensemble of them to improve its performance. We emphasize again that we study inference methods, not requiring any modification on pre-trained deep models. Hence, the proposed methods are easily applicable on top of any pre-trained modern neural classifiers.

2.1 Generative classifier and MCD estimator

Let $\mathbf{x}$ be an input and $y \in \{1, \cdots, C\}$ be its class label. Without loss of generality, suppose that a pre-trained softmax neural classifier is given: $P(y=c|\mathbf{x}) = \frac{\exp\left(\mathbf{w}_c^\top f(\mathbf{x}) + b_c\right)}{\sum_{c'} \exp\left(\mathbf{w}_{c'}^\top f(\mathbf{x}) + b_{c'}\right)}$, where $\mathbf{w}_c$ and b_c are the weight and the bias of the softmax classifier for class c, and $f(\cdot) \in \mathbb{R}^d$ denotes the output of the penultimate layer of DNNs. Then, without any modification on the pre-trained softmax neural classifier, we induce a generative classifier by assuming the class-conditional distribution follows the multivariate Gaussian distribution. In particular, we define C Gaussian distributions with a tied covariance $\mathbf{\Sigma}$, i.e., linear discriminant analysis (LDA) (Fisher, 1936), and a Bernoulli distribution for the class prior: $P(f(\mathbf{x})|y=c) = \mathcal{N}(f(\mathbf{x})|\mu_c, \mathbf{\Sigma}), P(y=c) = \beta_c$, where μ_c is the mean of multivariate Gaussian distribution and β_c is the normalized prior for class c. We provide an analytic justification on the LDA (i.e., tied covariance) assumption in Appendix A. Then, based on the Bayesian rule, we induce a new posterior different from the softmax one as follows:

$$P(y=c|f(\mathbf{x})) = \frac{P(y=c)\,P(f(\mathbf{x})|y=c)}{\sum_{c'} P(y=c')\,P(f(\mathbf{x})|y=c')} = \frac{\exp\left(\mu_c^\top \mathbf{\Sigma}^{-1} f(\mathbf{x}) - \frac{1}{2}\mu_c^\top \mathbf{\Sigma}^{-1}\mu_c + \log\beta_c\right)}{\sum_{c'} \exp\left(\mu_{c'}^\top \mathbf{\Sigma}^{-1} f(\mathbf{x}) - \frac{1}{2}\mu_{c'}^\top \mathbf{\Sigma}^{-1}\mu_{c'} + \log\beta_{c'}\right)}.$$

To estimate the parameters of the generative classifier, one can compute the sample class mean and covariance of training samples $\mathcal{X}_N = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$:

$$\bar{\mu}_c = \sum_{i:y_i=c} \frac{f(\mathbf{x}_i)}{N_c}, \quad \bar{\mathbf{\Sigma}} = \sum_c \sum_{i:y_i=c} \frac{(f(\mathbf{x}_i) - \bar{\mu}_c)(f(\mathbf{x}_i) - \bar{\mu}_c)^\top}{N}, \quad \bar{\beta}_c = \frac{N_c}{N}, \tag{1}$$

where N_c is the number of training samples labeled to be class c. We remark that inducing a generative classifier from a pre-trained softmax classifier under the LDA assumption was studied recently for different purposes (Lee et al., 2018b). Their main focus is on detecting abnormal samples like out-of-distribution (Lee et al., 2018a) and adversarial samples (Ma et al., 2018a). However, our goal is completely different, and even more difficult in the sense of assuming abnormality in training or/and defending it beyond detection. To tackle new challenges, we design a more advanced generative classifier as stated in below.

One can expect that the naive sample estimator (1) can be highly influenced by outliers (i.e., training samples with noisy labels). In order to improve the robustness, we propose the so-called deep determinantal generative classifier (DDGC), which utilize the minimum covariance determinant (MCD) estimator (Rousseeuw & Driessen, 1999) to estimate its parameters. For each class c, the main idea of MCD is finding a subset $\mathcal{X}_{K_c}$ for which the determinant of the corresponding sample covariance is minimized:

$$\min_{\mathcal{X}_{K_c} \subset \mathcal{X}_{N_c}} \det\left(\widehat{\mathbf{\Sigma}}_c\right) \quad \text{subject to } |\mathcal{X}_{K_c}| = K_c, \tag{2}$$

where $\mathcal{X}_{N_c}$ is the set of training samples labeled to be class c, $\widehat{\mathbf{\Sigma}}_c$ is the sample covariance of $\mathcal{X}_{K_c}$ and $0 < K_c < N_c$ is a hyperparameter. Then, only using the samples in $\bigcup_c \mathcal{X}_{K_c}$, it estimates the parameters, i.e., $\widehat{\mu}_c, \widehat{\mathbf{\Sigma}}, \widehat{\beta}_c$, of the generative classifier, by following (1). Such a new estimator can be more robust by removing the outliers which might be widely scattered in datasets. The robustness of MCD estimator has been justified in the literature: it is known to have near-optimal breakdown points (Hampel, 1971), i.e., the smallest fraction of data points that need to be replaced by arbitrary values (i.e., outliers) to fool the estimator completely. Formally, denote $\mathcal{Y}_M$ as a set obtained by replacing M data points of set $\mathcal{Y}$ by some arbitrary values. Then, for a multivariate mean estimator $\mu = \mu(\mathcal{Y})$ from $\mathcal{Y}$, the breakdown point is defined as follows (see Appendix A for more detailed explanations including the breakdown point of covariance estimator):

$$\varepsilon^*(\mu, \mathcal{Y}) = \frac{1}{|\mathcal{Y}|} \min\left\{M \in \{1, \cdots, |\mathcal{Y}|\} : \sup_{\mathcal{Y}_M} \|\mu(\mathcal{Y}) - \mu(\mathcal{Y}_M)\| = \infty\right\}.$$

While the breakdown point of the naive sample estimator is 0%, the MCD estimator for the generative classifier under LDA assumption is known to attain its breakdown value of $\min_c \frac{\lfloor (N_c-d+1)/2 \rfloor}{N_c} \approx 50\%$ (Lopuhaa et al., 1991). Inspired by this fact, we choose the default value of K_c in (2) by $\lfloor (N_c+d+1)/2 \rfloor$.

We also establish the following theoretical support that the MCD-based generative classifier, i.e., DDGC, can have smaller errors on parameter estimations and produce a larger margin, compared to the naive sample estimator, under some assumptions for its analytic tractability.

Theorem 1. *Assume the followings:*

- *($\mathcal{A}1$) The distribution of hidden features is $P(f(\mathbf{x})|y=c) = \mathcal{N}\left(f(\mathbf{x})|\mu_c, \sigma^2\mathbf{I}\right)$ (i.e., the conditional Gaussian distribution) and that of outliers is an arbitrary distribution, which has a density function in L^1 space (i.e., its absolute value is measurable), with mean $\mu_{\texttt{out}}$ and covariance matrix $\sigma^2_{\texttt{out}}\mathbf{I}$, where $\mathbf{I} \in \mathbb{R}^{d\times d}$ is the identity matrix.*
- *($\mathcal{A}2$) $\mu_{\texttt{out}} = \frac{1}{C}\sum_c \mu_c = \mathbf{0}$, and $\frac{1}{C}\sum_c \mu_c\mu_c^T$ is a diagonal matrix.*
- *($\mathcal{A}3$) All classes have the same number of samples (i.e., $N_c = \frac{N}{C}$), the same fraction $\delta_{\texttt{out}} < 1$ of outliers, and the sample fraction $\delta_{\texttt{mcd}} = \frac{K_c}{N_c} < 1$ of samples selected by MCD estimator.*
- *($\mathcal{A}4$) The outliers are widely scattered such that $\sigma^2 \leq \sigma^2_{\texttt{out}}$.*
- *($\mathcal{A}5$) The number of outliers is not too large such that $\delta_{\texttt{out}} < 1 - \delta_{\texttt{mcd}}$ and $\delta_{\texttt{mcd}} > \frac{d}{N_c}$.*

Let $\widehat{\mu}, \widehat{\mathbf{\Sigma}}$ and $\bar{\mu}, \bar{\mathbf{\Sigma}}$ be the outputs of the MCD and sample estimators, respectively. Then, $\widehat{\mu}, \bar{\mu}, \widehat{\mathbf{\Sigma}}, \bar{\mathbf{\Sigma}}$ converge to their expectation almost surely as $N \to \infty$, and it holds that

$$\frac{B(\widehat{\mu}_c, \widehat{\mu}_{c'}, \mathbf{\Sigma}, \widehat{\mathbf{\Sigma}})}{B(\bar{\mu}_c, \bar{\mu}_{c'}, \mathbf{\Sigma}, \bar{\mathbf{\Sigma}})} \overset{a.s.}{\to} \lim_{N\to\infty} \frac{B(\widehat{\mu}_c, \widehat{\mu}_{c'}, \mathbf{\Sigma}, \widehat{\mathbf{\Sigma}})}{B(\bar{\mu}_c, \bar{\mu}_{c'}, \mathbf{\Sigma}, \bar{\mathbf{\Sigma}})} \leq \exp\left(-\frac{\|\mu_c - \mu_{c'}\|_2^2 \left(1-(1-\delta_{\texttt{out}})^2\right)}{8\sigma^2}\right) \leq 1, \tag{3}$$

$$\|\mu_c - \widehat{\mu}_c\|_1 \overset{a.s.}{\to} \lim_{N\to\infty} \|\mu_c - \widehat{\mu}_c\|_1 = 0, \qquad \|\mu_c - \bar{\mu}_c\|_1 \overset{a.s.}{\to} \lim_{N\to\infty} \|\mu_c - \bar{\mu}_c\|_1 = \delta_{\texttt{out}}\|\mu_c\|_1 \tag{4}$$

$$\frac{(\widehat{\mu}_c - \widehat{\mu}_{c'})^T \widehat{\mathbf{\Sigma}}^{-1} (\widehat{\mu}_c - \widehat{\mu}_{c'})}{(\bar{\mu}_c - \bar{\mu}_{c'})^T \bar{\mathbf{\Sigma}}^{-1} (\bar{\mu}_c - \bar{\mu}_{c'})} \overset{a.s.}{\to} \lim_{N\to\infty} \frac{(\widehat{\mu}_c - \widehat{\mu}_{c'})^T \widehat{\mathbf{\Sigma}}^{-1} (\widehat{\mu}_c - \widehat{\mu}_{c'})}{(\bar{\mu}_c - \bar{\mu}_{c'})^T \bar{\mathbf{\Sigma}}^{-1} (\bar{\mu}_c - \bar{\mu}_{c'})} = \frac{1}{(1-\delta_{\texttt{out}})^2} \geq 1. \tag{5}$$

for all c, c', where $B(\widehat{\mu}_c, \widehat{\mu}_{c'}, \mathbf{\Sigma}, \widehat{\mathbf{\Sigma}}) := \exp\left(-\frac{1}{8}\frac{[(\widehat{\mu}_c - \widehat{\mu}_{c'})^T \widehat{\mathbf{\Sigma}}^{-1} (\widehat{\mu}_c - \widehat{\mu}_{c'})]^2}{(\widehat{\mu}_c - \widehat{\mu}_{c'})^T \widehat{\mathbf{\Sigma}}^{-1} \mathbf{\Sigma} \widehat{\mathbf{\Sigma}}^{-1} (\widehat{\mu}_c - \widehat{\mu}_{c'})}\right)$.

The proof of the above theorem is given in Appendix G, where it is built upon the fact that the determinants can be expressed as the d-th degree polynomial of outlier ratio. We note that one might enforce the assumptions of the diagonal covariance matrices in $\mathcal{A}1$ and the zero-mean/uncorrelated properties of class mean in $\mathcal{A}2$ to hold under an affine translation of hidden features. In addition, the assumption in $\mathcal{A}5$ holds when N_c is large enough. Nevertheless, we think most assumptions of Theorem 1 are not necessary to claim the superiority of DDGC (they are rather from a limitation of our proof techniques) and it is an interesting future direction to explore to relax them.

First, (5) implies that the MCD estimator induces a larger margin and improves the robustness to adversarial attacks, since Pang et al. (2018) showed that the margin of a generative classifier corresponds to the Mahalanobis distance between class-conditional distributions and the optimal robustness to adversarial examples is achieved by the maximum margin. More importantly, (3) and (4) together imply that a MCD-based classifier can generalize better since the generalization error of a generative classifier is known to be bounded as follows (Durrant & Kabán, 2010):

$$P_{\mathbf{x}}\left(y^* \neq \arg\max_y P_{\widehat{\mu}_c, \widehat{\Sigma}}(y|\mathbf{x})\right) \leq \sum_c \sum_{c' \neq c} B(\widehat{\mu}_{c'}, \widehat{\mu}_c, \mathbf{\Sigma}, \widehat{\mathbf{\Sigma}}) + D\sum_c \|\mu_c - \widehat{\mu}_c\|_1,$$

for some constant $D > 0$.

Algorithm 1 (Rousseeuw & Driessen, 1999) Approximating MCD for a single Gaussian.

Input: $\mathcal{X}_{N_c} = \{\mathbf{x}_i : i = 1, \cdots, N_c\}$ and the maximum number of iterations $I_{\max}$.

Uniformly sample initial subset $\mathcal{X}_{K_c} \subset \mathcal{X}_{N_c}$, where $|\mathcal{X}_{K_c}| = \lfloor (N_c + d + 1)/2 \rfloor$.
Compute mean and covariance $\widehat{\mu}_c = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$, $\widehat{\mathbf{\Sigma}}_c = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} (f(\mathbf{x}) - \widehat{\mu}_c)(f(\mathbf{x}) - \widehat{\mu}_c)^\top$.
for $i = 1$ **to** $I_{\max}$ **do**
  Compute the Mahalanobis distance: $\alpha(\mathbf{x}) = (f(\mathbf{x}) - \widehat{\mu}_{\mathbf{c}})^\top \widehat{\mathbf{\Sigma}}_c^{-1} (f(\mathbf{x}) - \widehat{\mu}_{\mathbf{c}})$, $\forall \mathbf{x} \in \mathcal{X}_{N_c}$.
  Update $\mathcal{X}_{K_c}$ such that it includes $\lfloor (N_c + d + 1)/2 \rfloor$ samples with smallest distance $\alpha(\mathbf{x})$.
  Compute sample means and covariance, i.e., $\widehat{\mu}_c$, $\widehat{\mathbf{\Sigma}}_c$, using new subset $\mathcal{X}_{K_c}$.
  breakdown the loop if the determinant of covariance matrix is not decreasing anymore.
end for
Return $\widehat{\mu}_c$ and $\widehat{\mathbf{\Sigma}}_c$

2.2 Approximation algorithm for MCD

Even though the MCD estimator has several advantages, the optimization (2) is computationally intractable (i.e., NP-hard) to solve (Bernholt, 2006). To handle this issue, we aim for computing its approximate solution, following a similar idea to that by (Hubert & Van Driessen, 2004). We design two step scheme as follows: (a) obtain the mean and covariance, i.e., $\widehat{\mu}_c$, $\widehat{\mathbf{\Sigma}}_c$, using Algorithm 1 for each class c, and (b) compute the tied covariance by $\widehat{\mathbf{\Sigma}} = \frac{\sum_c K_c \widehat{\mathbf{\Sigma}}_c}{\sum_c K_c}$. In other words, we apply the MCD estimator for each class, and combine the individual covariances into a single one due to the tied covariance assumption of LDA. Even though finding the optimal solution of MCD estimator under a single Gaussian distribution is still intractable, Algorithm 1 can produce a local optimal solution since it monotonically decreases the determinant under any random initial subset (Rousseeuw & Driessen, 1999). We choose $I_{\max} = 2$ in our experiments as the additional iterations would not improve the results significantly.

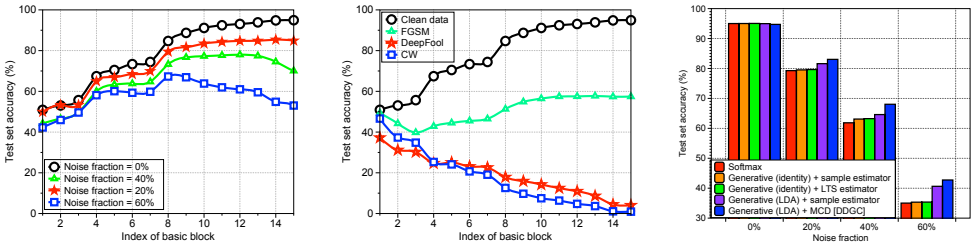
2.3 Ensemble of generative classifiers

To further improve the performance of our method, we consider the ensemble of generative classifiers not only from the penultimate features but also from other low-level features in DNNs. Formally, given training data, we extract ℓ-th hidden features of DNNs, denoted by $f_\ell(\mathbf{x}) \in \mathbb{R}^{d_\ell}$, and compute the corresponding parameters of a generative classifier (i.e., $\widehat{\mu}_{\ell,c}$ and $\widehat{\mathbf{\Sigma}}_\ell$) using the (approximated version of) MCD estimator. Then, the final posterior distribution is obtained by the sum of all posterior distributions of generative classifiers:

$$\sum_\ell \alpha_\ell P_\ell(y = c|\mathbf{x}) = \sum_\ell \alpha_\ell \frac{\exp\left(\widehat{\mu}_{\ell,c}^\top \widehat{\mathbf{\Sigma}}_\ell^{-1} f_\ell(\mathbf{x}) - 0.5 \widehat{\mu}_{\ell,c}^\top \widehat{\mathbf{\Sigma}}_\ell^{-1} \widehat{\mu}_{\ell,c} + \log \widehat{\beta}_c\right)}{\sum_{c'} \exp\left(\widehat{\mu}_{\ell,c'}^\top \widehat{\mathbf{\Sigma}}_\ell^{-1} f_\ell(\mathbf{x}) - 0.5 \widehat{\mu}_{\ell,c'}^\top \widehat{\mathbf{\Sigma}}_\ell^{-1} \widehat{\mu}_{\ell,c'} + \log \widehat{\beta}_{c'}\right)},$$

where α_ℓ is an ensemble weight at ℓ-th layer. In our experiments, we choose the weight of each layer by optimizing NLL loss over the validation set. One can expect that this simple but natural scheme can bring an extra gain in improving the performance due to ensemble effects.

To confirm that ensemble approach is indeed effective, we measure the classification accuracy of the generative classifier from different basic blocks of ResNet-34 (He et al., 2016) trained on CIFAR-10 dataset (Krizhevsky & Hinton, 2009) with various noise fractions, where the corresponding results on DenseNet models (Huang & Liu, 2017) can be found in Appendix D. For computational efficiency, the dimensions of the intermediate features are reduced using average pooling (see Section 3 for more details). First, we found that the performances of the generative classifiers from low-level features are more stable, while the accuracy of generative classifier from penultimate layer significantly decreases as the noisy fraction increases as shown in Figure 2(a). We expect that this is because the dimension (i.e., number of channels) of low-level features is usually smaller than that of high-level features. Since the breakdown point of MCD is inversely proportional to the feature dimension, the generative classifiers from low-level features can be more robust. This also coincides with the prior observation in the literature (Morcos et al., 2018) that DNNs tend to have similar hidden features at early layers, regardless of whether they train clean or noisy labels. More



(a) Generalization of generative classifiers from noisy labels

(b) Robustness of generative classifiers to adversarial attacks

(c) Generative classifiers under various assumptions

Figure 2: Experimental results under ResNet-34 model and CIFAR-10 dataset. (a) Test accuracy of generative classifiers computed at different basic blocks. (b) Test accuracy on various adversarial attacks when the model is trained on clean dataset. (c) Test accuracy of generative classifiers from penultimate features under various assumptions: identity covariance and tied covariance (LDA).

Model	Inference method	Ensemble	Noise = 0%	20%	40%	60%
DenseNet	Softmax	-	94.42	80.24	68.61	53.34
	Generative + sample	-	94.31	85.08	74.72	59.49
		✓	93.73	87.23	80.01	69.17
	Generative + MCD (DDGC)	-	94.37	85.96	78.34	66.16
		✓	93.49	**87.25**	**81.04**	**74.72**
ResNet	Softmax	-	95.01	79.28	61.85	35.02
	Generative + sample	-	94.98	81.61	64.60	40.63
		✓	94.98	87.23	78.40	61.94
	Generative + MCD (DDGC)	-	94.73	83.04	68.04	42.74
		✓	94.32	**87.25**	**80.01**	**71.06**

Table 1: Effects of ensemble method. We use the CIFAR-10 dataset with various noise fractions. All values are percentages and the best results are highlighted in bold if the gain is bigger than 1%.

importantly, we found that generative classifiers from low-level features are more robust to adversarial attacks like DeepFool (Moosavi Dezfooli et al., 2016) and CW (Carlini & Wagner, 2017), as shown in Figure 2(b). Therefore, we utilize the low-level generative classifiers as well to improve the generalization from noisy label and robustness to adversarial attacks simultaneously.

3 Experimental Results

In this section, we demonstrate the effectiveness of the proposed method using deep convolutional neural networks such as DenseNet (Huang & Liu, 2017) and ResNet (He et al., 2016) on various vision datasets: CIFAR (Krizhevsky & Hinton, 2009) and SVHN (Netzer et al., 2011). Due to the space limit, we provide the more detailed experimental setups and results in Appendix B.

3.1 Generalization from Noisy Labels

Setup. First, we evaluate the effectiveness of DDGC using deep models trained on datasets with noisy labels. We train DenseNet-100 and ResNet-34 for classifying CIFAR-10, CIFAR-100 and SVHN datasets. Following similar strategies in (Ma et al., 2018b; Vahdat, 2017), the labels of a given proportion of training samples are flipped to other class labels uniformly at random. For ensembles of generative classifiers, we induce the generative classifiers from every end of basic block of DenseNet (or ResNet), where ensemble weights of each layer are tuned on a separate validation set, which consists of 500 images (i.e., only $\leq 1\%$ of the number of training samples) with clean labels.[1] Similar to (Lee et al., 2018b), the size of feature maps on each convolutional layers is reduced by average pooling for computational efficiency: $\mathcal{F} \times \mathcal{H} \times \mathcal{W} \rightarrow \mathcal{F} \times 1$, where $\mathcal{F}$ is the number of channels and $\mathcal{H} \times \mathcal{W}$ is the spatial dimension.

[1]For fair comparisons, one might also suggest fine-tuning the softmax classifier (with fixed features) using the validation data to improve the performance. However, the 1% data is often not enough for the purpose as the number of parameters of softmax classifier is too large. We indeed report the performance of the fine-tuned softmax classifier on Table 9 in Appendix E.

Dataset	Model	Noise = 0%	Noise = 20%	Noise = 40%	Noise = 60%
		Softmax / DDGC			
CIFAR-10	ResNet	95.01 / 94.32	79.28 / **87.25**	61.85 / **80.01**	35.02 / **71.06**
	DenseNet	94.42 / 93.49	80.24 / **87.25**	68.61 / **81.04**	53.34 / **74.72**
CIFAR-100	ResNet	77.51 / 76.55	60.92 / **66.08**	44.08 / **59.72**	23.43 / **48.85**
	DenseNet	**76.41** / 73.65	57.63 / **62.19**	45.08 / **53.98**	35.83 / **45.27**
SVHN	ResNet	95.96 / 96.09	83.52 / **91.67**	72.89 / **87.16**	61.23 / **80.52**
	DenseNet	96.59 / 96.18	86.92 / **89.50**	81.91 / **85.71**	71.18 / **77.67**

Table 2: Test accuracy (%) of different models trained on noisy datasets. We use the ensemble version of DDGC, and the best results are highlighted in bold if the gain is bigger than 1%.

Dataset	Training method	Noise = 0%	Noise = 20%	Noise = 40%	Noise = 60%
		Softmax / DDGC			
CIFAR-10	Cross-entropy	94.49 / 94.35	79.49 / **86.05**	64.12 / **79.02**	60.05 / **71.38**
	Bootstrap (hard)	94.65 / 94.63	82.28 / **87.76**	75.79 / **83.35**	75.34 / **77.00**
	Bootstrap (soft)	94.48 / 94.33	81.67 / **86.72**	69.36 / **79.27**	59.99 / **71.52**
	Forward	94.32 / 94.25	85.23 / **87.31**	78.62 / **81.34**	72.30 / **74.65**
	Backward	94.39 / 94.47	76.18 / **81.62**	62.35 / **74.75**	57.02 / **63.58**
CIFAR-100	Cross-entropy	75.76 / 76.00	60.24 / **63.93**	47.52 / **53.54**	34.23 / **40.59**
	Bootstrap (hard)	75.67 / 75.62	61.84 / **65.16**	50.95 / **55.77**	38.60 / **44.17**
	Bootstrap (soft)	75.50 / 75.49	61.84 / **64.10**	46.88 / **53.15**	34.70 / **40.45**
	Forward	76.59 / 76.50	63.22 / **65.96**	51.87 / **56.96**	42.59 / **46.37**
	Backward	76.64 / 76.43	56.43 / **61.28**	38.91 / **50.34**	22.68 / **35.16**
SVHN	Cross-entropy	96.36 / 96.23	83.13 / **88.32**	64.30 / **75.25**	36.06 / **64.35**
	Bootstrap (hard)	96.20 / 96.18	86.52 / **90.37**	62.60 / **83.30**	61.56 / **75.43**
	Bootstrap (soft)	96.35 / 96.26	80.91 / **88.43**	57.21 / **77.38**	36.37 / **54.81**
	Forward	96.45 / 96.40	88.35 / **91.32**	77.87 / **87.74**	76.44 / **84.17**
	Backward	96.43 / 96.35	85.87 / **87.13**	70.81 / **76.73**	60.14 / **77.38**

Table 3: Test accuracy (%) of ResNet trained on various training methods. We use the ensemble version of DDGC, and the best results are highlighted in bold if the gain is bigger than 1%.

Verification of contributions from each technique. We first evaluate the performance of generative classifiers with various assumptions: identity covariance (Euclidean) and tied covariance (LDA). In the case of identity covariance, we also apply a robust estimator called the least trimmed square (LTS) estimator (Rousseeuw, 1984) which finds a K-subset with smallest error and computes the sample mean from it, i.e., $\min_{\widehat{\mu}} \sum_{i=1}^{K} (\|\mathbf{x}_i - \widehat{\mu}\|_2^2)$. As shown in Figure 2(c), the generative classifiers with LDA assumption (blue and purple bars) generalize better than the generative classifiers with identity covariance (orange and green bars) well from noisy labels. Table 1 validates the contributions of the proposed techniques by incrementally applying our techniques to see the improvement from adding each component one by one. One can note that the generative classifier on features extracted from the penultimate layer outperforms the softmax classifier without the MCD estimator or ensemble method, while it still provides a comparable classification accuracy when the model is trained on clean dataset (i.e., noise = 0%). On top of that, by utilizing the MCD estimator, the classification accuracy is further improved compared to that employs only the naive sample estimator. This implies that the proposed method can improve the performance without any information of clean labels. In addition, the ensemble method significantly improves the classification accuracy. Finally, Table 2 reports the classification accuracy for all networks and datasets, where the proposed method significantly outperforms the softmax classifier for all tested cases.

Compatibility and comparison with the state-of-art training methods. We compare the performance of the standard softmax classifier and DDGC when they are combined with other various training methods for noisy environment in Table 3. Here, we use ResNet-44 for all experiments,[2] For comparison, we consider the following training methods: Hard/soft bootstrapping (Reed et al., 2014), and forward/backward (Patrini et al., 2017), where more detailed explanations about training methods and network architectures are given in Appendix B. Table 3 shows the classification accuracy of softmax classifier and the ensemble version of DDGC. Note that DDGC always improves the classification accuracy compared to the softmax classifier, where the gains due to ours are more significant than those due to other special training methods.

[2]For all experiments, we follow the experimental setting in Ma et al. (2018b).

Dataset	Adversarial attacks	Noise = 0%	Noise = 20%	Noise = 40%	Noise = 60%
		Softmax / DDGC			
CIFAR-10	FGSM	57.04 / **58.09**	23.05 / **55.16**	23.56 / **56.55**	19.52 / **54.06**
	DeepFool	0.34 / **7.50**	0.06 / **55.94**	0.02 / **64.96**	0.00 / **63.88**
	CW	0.01 / **3.29**	0.47 / **52.21**	0.43 / **59.01**	0.06 / **61.90**
CIFAR-100	FGSM	31.72 / 32.40	18.16 / **32.85**	13.83 / **34.51**	9.34 / **36.41**
	DeepFool	0.45 / 1.40	0.18 / **10.47**	0.15 / **20.82**	0.07 / **30.55**
	CW	0.05 / **4.73**	0.30 / **31.42**	0.25 / **41.59**	0.03 / **43.31**
SVHN	FGSM	74.18 / **75.38**	22.27 / **75.26**	22.98 / **73.30**	24.71 / **66.83**
	DeepFool	1.17 / 1.82	0.04 / **71.14**	0.01 / **73.33**	0.01 / **66.30**
	CW	0.02 / **3.78**	0.03 / **68.55**	0.04 / **69.18**	0.01 / **62.20**

Table 4: Test accuracy (%) of ResNet on adversarial attacks. We use the ensemble version of DDGC, and the best results are highlighted in bold if the gain is bigger than 1%.

Dataset	Training method	FGSM	DeepFool	CW
		Softmax / DDGC		
CIFAR-10	Cross-entropy	57.04 / **58.09**	0.34 / **7.50**	0.01 / **3.29**
	Adversarial training	74.55 / 74.56	2.85 / **5.69**	0.11 / **1.32**
CIFAR-100	Cross-entropy	31.72 / 32.40	0.45 / 1.40	0.05 / **4.73**
	Adversarial training	47.31 / **49.47**	1.10 / 1.71	0.22 / **4.93**
SVHN	Cross-entropy	74.18 / **75.38**	1.17 / 1.82	0.02 / **3.78**
	Adversarial training	84.90 / 85.53	3.73 / **5.18**	0.52 / **2.42**

Table 5: Test accuracy (%) of ResNet trained on clean datasets under adversarial training. We use ensemble version of DDGC, and best results are highlighted in bold if the gain is bigger than 1%.

3.2 Robustness against adversarial attacks

Setup. We also evaluate if the proposed method can improve the robustness on adversarial attacks (Szegedy et al., 2014). It is well-known that the adversarial (visually imperceptible) perturbation to clean inputs can induce the DNNs to make incorrect predictions at test time. This undesirable property of DNNs has raised major security concerns. To verify that DDGC can improve the robustness to adversarial attacks, we train DenseNet-100 and ResNet-34 for classifying CIFAR-10, CIFAR-100 and SVHN datasets, and generate the adversarial examples using FGSM (Goodfellow et al., 2015), DeepFool (Moosavi Dezfooli et al., 2016) and CW (Carlini & Wagner, 2017) attacks, where the detailed explanations can be found in Appendix B.

Robustness against adversarial attacks. Table 4 shows the classification accuracy of ResNet-34 on adversarial examples, and more results on DenseNet-100 can be found in Appendix F. One can note that DDGC significantly improves the robustness against adversarial attacks since the generative classifiers from low-level features are robust as shown in Figure 2(b). Based on our observations, the adversarial samples are generated in a way that mainly fools the upper layers of DNNs, and thus both clean and adversarial samples produce similar hidden features at lower layers. In Table 5, we also apply DDGC to the "robust" learning models optimized by adversarial training methods (Goodfellow et al., 2015): generating FGSM samples and optimize the cross-entropy loss by treating them as additional training examples. Even though adversarial training requires an extra computational cost, the improvements of robustness on multi-step attack methods such as CW and DeepFool are not significant compared to DDGC. We also consider adaptive attacks based on FGSM against DDGC, and the results show that it is robust against such adaptive attacks (see Appendix C), while proposing stronger adaptive attacks to evaluate its robustness is an interesting future direction.

4 Conclusion

We propose a new inference method, easily applicable to any softmax neural classifier pre-trained on datasets with noisy labels. Our main idea is defining the generative classifier on top of fixed features from the pre-trained model. Such "deep generative classifiers" have been largely dismissed for fully-supervised classification settings as they are often substantially outperformed by discriminative deep classifiers (e.g., softmax classifiers). In contrast to this common belief, we show that it is possible to formulate a simple generative classifier that is more robust without sacrificing the discriminative performance. We expect that our work would bring a refreshing angle for other related tasks, e.g., adaptive attacks, memorization (Zhang et al., 2017) and interpretability (Morcos et al., 2018).

REFERENCES

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *ICML*, 2016.

Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *ICML*, 2017.

Thorsten Bernholt. Robust estimators are hard to compute. Technical report, Technical Report/Universität Dortmund, SFB 475 Komplexitätsreduktion in Multivariaten Datenstrukturen, 2006.

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM workshop on AISec*, 2017.

Robert J Durrant and Ata Kabán. Compressed fisher linear discriminant analysis: Classification of randomly projected data. In *ACM SIGKDD*, 2010.

Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 1936.

Ross Girshick. Fast r-cnn. In *ICCV*, 2015.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. Countering adversarial images using input transformations. In *ICLR*, 2018.

Frank R Hampel. A general qualitative definition of robustness. *The Annals of Mathematical Statistics*, 1971.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In *NIPS*, 2018.

Gao Huang and Zhuang Liu. Densely connected convolutional networks. In *CVPR*, 2017.

Mia Hubert and Katrien Van Driessen. Fast and robust discriminant analysis. *Computational Statistics & Data Analysis*, 2004.

Jonathan Krause, Benjamin Sapp, Andrew Howard, Howard Zhou, Alexander Toshev, Tom Duerig, James Philbin, and Li Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. In *ECCV*, 2016.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Julia A Lasserre, Christopher M Bishop, and Thomas P Minka. Principled hybrids of generative and discriminative models. In *CVPR*, 2006.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *ICLR*, 2018a.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NIPS*, 2018b.

Hendrik P Lopuhaa, Peter J Rousseeuw, et al. Breakdown points of affine equivariant estimators of multivariate location and covariance matrices. *The Annals of Statistics*, 1991.

Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Michael E Houle, Grant Schoenebeck, Dawn Song, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. In *ICLR*, 2018a.

Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah M Erfani, Shu-Tao Xia, Sudanthi Wijewickrema, and James Bailey. Dimensionality-driven learning with noisy labels. In *ICML*, 2018b.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 2008.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. *arXiv preprint arXiv:1805.00932*, 2018.

Seyed Mohsen Moosavi Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, 2016.

Ari S Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. In *NIPS*, 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop*, 2011.

Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *NIPS*, 2002.

Tianyu Pang, Chao Du, and Jun Zhu. Max-mahalanobis linear discriminant analysis networks. In *ICML*, 2018.

Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*, 2017.

Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

Peter J Rousseeuw. Least median of squares regression. *Journal of the American statistical association*, 1984.

Peter J Rousseeuw and Katrien Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 1999.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

Arash Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *NIPS*, 2017.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

A PRELIMINARIES

Gaussian discriminant analysis. In this section, we describe the basic concepts of the discriminative and generative classifier (Ng & Jordan, 2002). Formally, denote the random variable of the input and label as $\mathbf{x}$ and $y = \{1, \cdots, C\}$, respectively. For the classification task, the discriminative classifier directly defines a posterior distribution $P(y|\mathbf{x})$, i.e., learning a direct mapping between input $\mathbf{x}$ and label y. A popular model for discriminative classifier is softmax classifier which defines the posterior distribution as follows: $P(y = c|\mathbf{x}) = \frac{\exp\left(\mathbf{w}_c^\top \mathbf{x} + b_c\right)}{\sum_{c'} \exp\left(\mathbf{w}_{c'}^\top \mathbf{x} + b_{c'}\right)}$, where $\mathbf{w}_c$ and b_c are weights and bias for a class c, respectively. In contrast to the discriminative classifier, the generative classifier defines the class conditional distribution $P(\mathbf{x}|y)$ and class prior $P(y)$ in order to indirectly define the posterior distribution by specifying the joint distribution $P(\mathbf{x}, y) = P(y) P(\mathbf{x}|y)$. Gaussian discriminant analysis (GDA) is a popular method to define the generative classifier by assuming that the class conditional distribution follows the multivariate Gaussian distribution and the class prior follows Bernoulli distribution: $P(\mathbf{x}|y = c) = \mathcal{N}(\mathbf{x}|\mu_c, \mathbf{\Sigma}_c), P(y = c) = \frac{\beta_c}{\sum_{c'} \beta_{c'}}$, where μ_c and $\mathbf{\Sigma}_c$ are the mean and covariance of multivariate Gaussian distribution, and β_c is the unnormalized prior for class c. This classifier has been studied in various machine learning areas (e.g., semi-supervised learning (Lasserre et al., 2006) and incremental learning (Lee et al., 2018b)).

In this paper, we focus on the special case of GDA, also known as the linear discriminant analysis (LDA). In addition to Gaussian assumption, LDA further assumes that all classes share the same covariance matrix, i.e., $\mathbf{\Sigma}_c = \mathbf{\Sigma}$. Since the quadratic term is canceled out with this assumption, the posterior distribution of generative classifier can be represented as follows:

$$P(y = c|\mathbf{x}) = \frac{P(y = c) P(\mathbf{x}|y = c)}{\sum_{c'} P(y = c') P(\mathbf{x}|y = c')} = \frac{\exp\left(\mu_c^\top \mathbf{\Sigma}^{-1} \mathbf{x} - \frac{1}{2} \mu_c^\top \mathbf{\Sigma}^{-1} \mu_c + \log \beta_c\right)}{\sum_{c'} \exp\left(\mu_{c'}^\top \mathbf{\Sigma}^{-1} \mathbf{x} - \frac{1}{2} \mu_{c'}^\top \mathbf{\Sigma}^{-1} \mu_{c'} + \log \beta_{c'}\right)}.$$

One can note that the above form of posterior distribution is equivalent to the softmax classifier by considering $\mu_c^\top \mathbf{\Sigma}^{-1}$ and $-\frac{1}{2} \mu_c^\top \mathbf{\Sigma}^{-1} \mu_c + \log \beta_c$ as its weight and bias, respectively. This implies that $\mathbf{x}$ might be fitted in Gaussian distribution during training a softmax classifier.

Breakdown points. The robustness of MCD estimator can be explained by the fact that it has high breakdown points (Hampel, 1971). Specifically, the breakdown point of an estimator measures the smallest fraction of observations that need to be replaced by arbitrary values to carry the estimate beyond all bounds. Formally, denote $\mathcal{Y}_M$ as a set obtained by replacing M data points of set $\mathcal{Y}$ by some arbitrary values. Then, for a multivariate mean estimator $\mu = \mu(\mathcal{Y})$ from $\mathcal{Y}$, the breakdown point is defined as follows (see Appendix A for more detailed explanations including the breakdown point of covariance estimator):

$$\varepsilon^*(\mu, \mathcal{Y}) = \frac{1}{|\mathcal{Y}|} \min \left\{ M \in \{1, \cdots, |\mathcal{Y}|\} : \sup_{\mathcal{Y}_M} \|\mu(\mathcal{Y}) - \mu(\mathcal{Y}_M)\| = \infty \right\}.$$

For a multivariate estimator of covariance $\mathbf{\Sigma}$, we have

$$\varepsilon^*(\mathbf{\Sigma}, \mathcal{Y}) = \frac{1}{|\mathcal{Y}|} \min\{M \in \{1, \cdots, |\mathcal{Y}|\} : \sup_M \max_i \{|\log \lambda_i(\mathbf{\Sigma}(\mathcal{Y})) - \log \lambda_i(\mathbf{\Sigma}(\mathcal{Y}_M))|\}\},$$

where the $k-$th largest eigenvalue of a general $n \times n$ matrix is denoted by $\lambda_k(\mathbf{\Sigma})$, $k = 1, \cdots, n$ such that $\lambda_1(\mathbf{\Sigma}) \leq \lambda_2(\mathbf{\Sigma}) \leq \cdots \leq \lambda_n(\Sigma)$. This implies that we consider a covariance estimator to be broken whenever any of the eigenvalues can become arbitrary large or arbitrary close to 0.

B EXPERIMENTAL SETUP

In this section, we describe detailed explanation about all the experiments described in Section 3.

Detailed model architecture and datasets. We consider two state-of-the-art neural network architectures: DenseNet (Huang & Liu, 2017) and ResNet (He et al., 2016). For DenseNet, our model follows the same setup as in Huang & Liu (2017): 100 layers, growth rate $k = 12$ and dropout rate 0. Also, we use ResNet with 34 and 44 layers, filters = 64 and dropout rate 0.[3] The softmax classifier

[3]ResNet architecture is available at https://github.com/kuangliu/pytorch-cifar.

is used, and each model is trained by minimizing the cross-entropy loss. We train DenseNet and ResNet for classifying CIFAR-10 (or 100) and SVHN datasets: the former consists of 50,000 training and 10,000 test images with 10 (or 100) image classes, and the latter consists of 73,257 training and 26,032 test images with 10 digits.[4] By following the experimental setup of Ma et al. (2018b), All networks were trained using SGD with momentum 0.9, weight decay 10^{-4} and an initial learning rate of 0.1. The learning rate is divided by 10 after epochs 40 and 80 for CIFAR-10/SVHN (120 epochs in total), and after epochs 80, 120 and 160 for CIFAR-100 (200 epochs in total). For our method, we extract the hidden features at $\{34, 46, 56, 67, 79, 89, 99\}$-th layers and $\{3, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33\}$-th layers for DenseNet and ResNet, respectively.

Training method for noisy label learning. We consider the following training methods for noisy label learning:

(a) **Hard bootstrapping** (Reed et al., 2014): Training with new labels generated by a convex combination (the hard version) of the noisy labels and their predicted labels.

(b) **Soft bootstrapping** (Reed et al., 2014): Training with new labels generated by a convex combination (the soft version) of the noisy labels and their predictions.

(c) **Backward** (Patrini et al., 2017): Training via loss correction by multiplying the cross-entropy loss by a noise-aware correction matrix.

(d) **Forward** (Patrini et al., 2017): Training with label correction by multiplying the network prediction by a noise-aware correction matrix.

(e) **Cross-entropy**: the conventional approach of training with cross-entropy loss.

Adversarial attacks. In this paper, we consider the following attack methods: fast gradient sign method (FGSM) (Goodfellow et al., 2015), DeepFool (Moosavi Dezfooli et al., 2016) and Carlini-Wagner (CW) (Carlini & Wagner, 2017). The FGSM directly perturbs normal input in the direction of the loss gradient. Formally, non-targeted adversarial examples are constructed as

$$\mathbf{x}_{adv} = \mathbf{x} + \varepsilon_{FGSM} \text{sign}\left(\bigtriangledown_{\mathbf{x}} \ell(y^*, P(y|\mathbf{x}))\right),$$

where ε_{FGSM} is a magnitude of noise, y^* is the prediction of classifier and ℓ is a loss function to measure the distance between the prediction and the ground truth. DeepFool works by finding the closest adversarial examples with geometric formulas. CW is an optimization-based method which arguably the most effective method. Formally, non-targeted adversarial examples are constructed as

$$\arg\min_{\mathbf{x}_{adv}} \ \lambda d(\mathbf{x}, \mathbf{x}_{adv}) - \ell(y^*, P(y|\mathbf{x}_{adv})),$$

where λ is penalty parameter and $d(\cdot, \cdot)$ is a metric to quantify the distance between an original image and its adversarial counterpart. However, compared to FGSM and BIM, this method is much slower in practice. For all experiments, L_2 distance is used as a constraint. We used the library from (Guo et al., 2018) for generating adversarial samples.[5] Table 6 and 7 report the statistics of adversarial attacks including the L_∞ mean perturbation and classification accuracy on adversarial attacks.

C ADAPTIVE ATTACKS AGAINST GENERATIVE CLASSIFIER

In this section, we evaluate the robustness of our method by generating adversarial samples which may fool the generative classifiers in a white-box setting, i.e., one can access to the parameters of the generative classifiers. Here, we remark that accessing the parameters of the generative classifiers, i.e., sample means and covariance, is not mild assumption since the information about training data is required to compute them. To attack our method, we generate an adversarial examples using FGSM method as follows:

$$\mathbf{x}_{adv} = \mathbf{x} + \varepsilon_{FGSM} \text{sign}\left(\bigtriangledown_{\mathbf{x}} \ell(y^*, P(y|\mathbf{x}))\right),$$

where ε_{FGSM} is a magnitude of noise, y^* and ℓ is a loss function, i.e., cross-entropy loss, to measure the distance between the prediction and the ground truth. We test two different scenarios using

[4] We do not use the extra SVHN dataset for training.

[5] The code is available at https://github.com/facebookresearch/adversarial_image_defenses.

Dataset	Data type	Noise = 0%		Noise = 20%		Noise = 40%		Noise = 60%	
		L_∞	Acc.	L_∞	Acc.	L_∞	Acc.	L_∞	Acc.
CIFAR-10	Clean	0	95.01%	0	79.28%	0	61.85%	0	35.02%
	FGSM	0.05	57.04%	0.05	23.05%	0.05	23.56%	0.05	19.52%
	DeepFool	0.82	0.34%	0.20	0.06%	0.08	0.02%	0.05	0.00%
	CW	0.09	0.01%	0.03	0.47%	0.02	0.43%	0.01	0.06%
CIFAR-100	Clean	0	77.51%	0	60.92%	0	44.08%	0	23.43%
	FGSM	0.05	31.72%	0.05	18.16%	0.05	13.83%	0.05	9.34%
	DeepFool	2.09	0.45%	1.00	0.18%	0.60	0.15%	0.31	0.07%
	CW	0.06	0.05%	0.03	0.30%	0.02	0.25%	0.01	0.03%
SVHN	Clean	0	96.10%	0	83.52%	0	72.89%	0	61.23%
	FGSM	0.05	74.18%	0.05	22.27%	0.05	22.98%	0.05	24.71%
	DeepFool	2.32	1.17%	0.26	0.04%	0.17	0.01%	0.16	0.01%
	CW	0.16	0.02%	0.05	0.03%	0.04	0.04%	0.03	0.01%

Table 6: The L_∞ mean perturbation and classification accuracy of ResNet-34 on clean and adversarial samples.

Dataset	Data type	Noise = 0%		Noise = 20%		Noise = 40%		Noise = 60%	
		L_∞	Acc.	L_∞	Acc.	L_∞	Acc.	L_∞	Acc.
CIFAR-10	Clean	0	94.42%	0	80.24%	0	68.61%	0	53.33%
	FGSM	0.05	32.63%	0.05	22.75%	0.05	22.33%	0.05	19.57%
	DeepFool	0.34	0.09%	0.09	0.01%	0.06	0.01%	0.05	0.02%
	CW	0.06	0.08%	0.04	0.42%	0.02	0.34%	0.02	0.09%
CIFAR-100	Clean	0	76.41%	0	57.63%	0	45.08%	0	35.83%
	FGSM	0.05	18.88%	0.05	12.20%	0.05	10.09%	0.05	8.51%
	DeepFool	1.14	0.27%	0.46	0.09%	0.31	0.02%	0.25	0.05%
	CW	0.03	0.22%	0.02	0.21%	0.01	0.10%	0.01	0.16%
SVHN	Clean	0	96.59%	0	86.91%	0	81.90%	0	71.18%
	FGSM	0.05	70.05%	0.05	51.69%	0.05	43.37%	0.05	31.08%
	DeepFool	1.04	0.21%	0.35	0.09%	0.25	0.08%	0.17	0.05%
	CW	0.13	0.05%	0.07	0.10%	0.06	0.10%	0.05	0.02%

Table 7: The L_∞ mean perturbation and classification accuracy of DenseNet-100 on clean and adversarial samples.

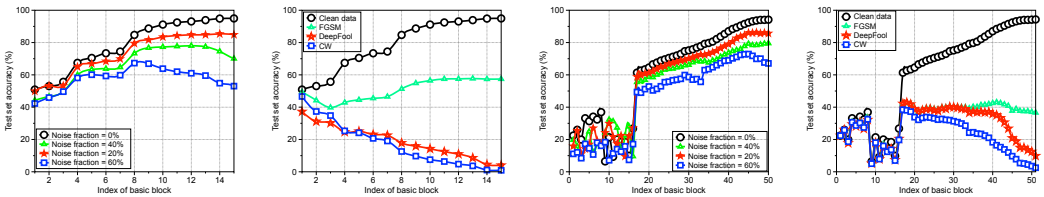
ResNet with 34 layers trained on CIFAR-10 dataset. In the first scenario, we generate the adversarial images by targeting the posterior distribution of generative classifier from a penultimate layer of DNNs: $P\left(y=c|f(\mathbf{x})\right) = \frac{\exp\left(\mu_{c,L}^\top \mathbf{\Sigma}_L^{-1} f(\mathbf{x}) - \frac{1}{2}\mu_{c,L}^\top \mathbf{\Sigma}_L^{-1}\mu_{c,L} + \log\beta_c\right)}{\sum_{c'}\exp\left(\mu_{c',L}^\top \mathbf{\Sigma}_L^{-1} f(\mathbf{x}) - \frac{1}{2}\mu_{c',L}^\top \mathbf{\Sigma}_L^{-1}\mu_{c',L} + \log\beta_{c'}\right)}$. In the second scenario, we attack the ensemble of posterior distributions:

$$P\left(y=c|f(\mathbf{x})\right) = \sum_\ell \alpha_\ell \frac{\exp\left(\widehat{\mu}_{\ell,c}^\top \widehat{\mathbf{\Sigma}}_\ell^{-1} f_\ell(\mathbf{x}) - 0.5\widehat{\mu}_{\ell,c}^\top \widehat{\mathbf{\Sigma}}_\ell^{-1}\widehat{\mu}_{\ell,c} + \log\widehat{\beta}_c\right)}{\sum_{c'}\exp\left(\widehat{\mu}_{\ell,c'}^\top \widehat{\mathbf{\Sigma}}_\ell^{-1} f_\ell(\mathbf{x}) - 0.5\widehat{\mu}_{\ell,c'}^\top \widehat{\mathbf{\Sigma}}_\ell^{-1}\widehat{\mu}_{\ell,c'} + \log\widehat{\beta}_{c'}\right)}.$$

Table 8 shows the the classification accuracy on FGSM samples generated by targeting various posterior distributions. We remark that adaptive attacks on generative classifiers are not successfully generating the adversarial samples, i.e., the proposed generative classifier is more robust to adaptive attacks. We found that this is because the posterior distributions from generative classifier are very confident, i.e., predicting the true answer with 100% confidence. Therefore, FGSM-based adaptive attacks are failed since the gradients of cross-entropy loss with respect to input consist of zero ele-

Dataset	Perturbation	Softmax	DDGC	DDGC + ensemble
CIFAR-10	$\varepsilon = 0$ (clean)	95.00%	94.99%	94.99%
	$\varepsilon = 0.01$	56.90%	94.19%	92.59%
	$\varepsilon = 0.1$	17.77%	93.57%	91.47%

Table 8: The classification accuracy on FGSM samples attacking various posterior distributions.



(a) Generalization from noisy labels (b) Robustness to adversarial attacks (c) Generalization from noisy labels (d) Robustness to adversarial attacks

Figure 3: Layer-wise characteristics of generative classifiers from (a)/(b) ResNet-34 and (c)/(d) DenseNet-100 trained on the CIFAR-10 dataset.

ments. Designing more advanced adaptive attacks such optimization-based method is an interesting future direction to explore.

D LAYER-WISE CHARACTERISTICS OF GENERATIVE CLASSIFIERS

Figure 3 shows the classification accuracy of the generative classifiers from different basic blocks of ResNet-34 (He et al., 2016) and DenseNet-101 (Huang & Liu, 2017). One can note that the generative classifiers from DenseNet and ResNet have different patterns due to the architecture design. In the case of DenseNet, we found that it produces meaning features after 20-th basic blocks. However, we remark that the performances of the generative classifiers from low-level features (from 20 to 40-th layers) of DenseNet are still more robust to adversarial attacks and noisy labels. Because of that, the ensemble of generative classifiers on DenseNet also improves the generalization from noisy labels and robustness to adversarial attacks.

E FINE-TUNING SOFTMAX LAYER USING VALIDATION SET

In this paper, we utilize a validation set, which consists of 500 images (i.e., only $\leq 1\%$ of the number of training samples) with clean labels. For fair comparisons, one might also suggest fine-tuning the softmax classifier (with fixed features) using the validation improve the classification accuracy. We indeed measure the classification accuracy of softmax classifier after fine-tuning in Table 9. One can note that the proposed method still outperforms softmax classifiers. Since 1% data is not enough compared to the number of softmax classifiers, fine-tuning can decrease the performance in the case of softmax classifier, while DDGC is still working well (it only trains the ensemble weights).

F MORE EXPERIMENTAL RESULTS ON ADVERSARIAL ATTACKS

In this section, we provide more experimental results on adversarial attacks using DenseNet. First, Table 10 shows the classification accuracy on adversarial attacks when we train DenseNet on noisy datsets. One can note that the proposed method outperforms softmax classifier for all tested cases. We also apply our method to the models optimized by adversarial training method. As shown in Table 11, the proposed method further improves the robustness.

Model	Inference Method	Noisy Dataset								
		CIFAR-10			CIFAR-100			SVHN		
		20%	40%	60%	20%	40%	60%	20%	40%	60%
ResNet	Softmax	79.28	61.85	35.02	60.92	44.08	23.43	83.52	72.89	61.23
	Softmax + fine-tuning (val)	80.48	69.42	54.16	51.15	35.17	20.41	85.10	81.93	75.06
	Generative + MCD	83.04	68.04	42.74	63.20	51.71	33.96	89.60	78.88	66.96
	Generative + MCD + ensemble (val)	**87.25**	**80.01**	**71.06**	**66.08**	**59.72**	**48.85**	**91.67**	**87.16**	**80.52**
DenseNet	Softmax	80.24	68.61	53.34	57.63	45.08	35.83	86.92	81.91	71.18
	Softmax + fine-tuning (val)	82.47	74.38	63.25	48.69	35.97	27.73	86.83	76.39	68.13
	Generative + MCD	85.96	78.34	66.16	58.22	48.54	37.44	88.64	84.08	74.47
	Generative + MCD + ensemble (val)	**87.25**	**81.04**	**74.72**	**62.19**	**53.98**	**45.27**	**89.50**	**85.71**	**77.67**

Table 9: Comparison with softmax classifier fine-tuned with validation data. All values are percentages and the best results are highlighted in bold if gain is bigger than 1%.

Dataset	Adversarial attacks	Noise = 0%	Noise = 20%	Noise = 40%	Noise = 60%
		Softmax / DDGC			
CIFAR-10	FGSM	32.63 / **40.01**	22.75 / **53.06**	22.33 / **54.54**	19.57 / **52.94**
	DeepFool	0.09 / **20.36**	0.01 / **65.34**	0.01 / **64.25**	0.02 / **62.28**
	CW	0.08 / **8.47**	0.42 / **49.27**	0.34 / **54.65**	0.09 / **55.85**
CIFAR-100	FGSM	18.88 / **23.87**	12.20 / **28.42**	10.09 / **26.46**	8.51 / **18.26**
	DeepFool	0.27 / **2.79**	0.09 / **18.01**	0.02 / **22.74**	0.05 / **20.25**
	CW	0.22 / **13.66**	0.21 / **34.63**	0.11 / **34.75**	0.16 / **29.83**
SVHN	FGSM	70.05 / **75.74**	51.69 / **71.11**	43.37 / **64.40**	31.08 / **50.61**
	DeepFool	0.21 / **19.93**	0.09 / **56.99**	0.08 / **55.03**	0.05 / **40.59**
	CW	0.05 / **24.54**	0.10 / **51.20**	0.11 / **46.29**	0.02 / **30.45**

Table 10: Test accuracy (%) of DenseNet on adversarial attacks. We use the ensemble version of DDGC, and the best results are highlighted in bold if gain is bigger than 1%.

Dataset	Training method	FGSM	DeepFool	CW
		Softmax / DDGC		
CIFAR-10	Cross-entropy	32.63 / **40.01**	0.09 / **20.36**	0.08 / **8.47**
	Adversarial training	66.12 / **68.18**	0.55 / **11.96**	0.07 / **5.81**
CIFAR-100	Cross-entropy	18.88 / **23.87**	0.27 / **2.79**	0.22 / **13.66**
	Adversarial training	36.67 / **43.26**	0.75 / **2.27**	0.07 / **12.77**
SVHN	Cross-entropy	70.05 / **75.74**	0.21 / **19.93**	0.05 / **24.54**
	Adversarial training	85.23 / 85.95	1.49 / **4.46**	0.62 / **9.11**

Table 11: Test accuracy (%) on adversarial examples. We use DenseNet trained on CIFAR and SVHN with clean labels, and ensemble version of DDGC. The best results are highlighted in bold if gain is bigger than 1%.

G PROOF OF THEOREM 1

In this section, we present a proof of Theorem 1. We start with the following lemma, which shows the properties of convergences of the sample and MCD estimators as the total number of training samples N goes to infinity.

Lemma 1. *Suppose we have N number of d-dimensional random samples $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$, where $\delta_{\texttt{out}} < 1$ is the fraction of outliers from an arbitrary distribution, which has a density function in L^1 space (i.e. its absolute value is measurable), with zero mean and finite covariance matrix $\sigma_{\texttt{out}}^2 \mathbf{I}$ and the others from $\mathcal{N}\left(\mu, \sigma^2 \mathbf{I}\right)$. Let $\bar{\mu}$ and $\bar{\mathbf{\Sigma}}$ be the mean and covariance matrix of sample estimator, and let $\widehat{\mu}$ and $\widehat{\mathbf{\Sigma}}$ be the mean and covariance matrix of MCD estimator which selects the fraction $\frac{d}{N} < \delta_{\texttt{mcd}} < 1$ of samples from $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ to optimize its objective (2). Then the sample estimator converges to below almost surely as $N \to \infty$,*

$$\bar{\mu} \overset{a.s.}{\to} (1 - \delta_{\texttt{out}})\, \mu, \quad \bar{\mathbf{\Sigma}} \overset{a.s.}{\to} \left((1 - \delta_{\texttt{out}})\, \sigma^2 + \delta_{\texttt{out}} \sigma_{\texttt{out}}^2\right) \mathbf{I} + \delta_{\texttt{out}} (1 - \delta_{\texttt{out}})\, \mu\mu^T.$$

Also, if $\delta_{\texttt{mcd}} \leq 1 - \delta_{\texttt{out}}$ and $\sigma^2 \leq \sigma_{\texttt{out}}^2$, the MCD estimator converges to below almost surely as $N \to \infty$.

$$\widehat{\mu} \overset{a.s.}{\to} \mu, \qquad \widehat{\mathbf{\Sigma}} \overset{a.s.}{\to} \sigma^2 \mathbf{I}.$$

Proof. To obatin $\bar{\mu}$ and $\bar{\mathbf{\Sigma}}$, consider a random variable z that is from a mixture distribution of two distributions, $\mathcal{N}\left(\mu, \sigma^2 \mathbf{I}\right)$ and outlier distribution, where its pdf $p(z)$ is consisted of $(1 - \delta_{\texttt{out}})\, \mathcal{N}(\mu, \sigma^2 \mathbf{I}) + \delta_{\texttt{out}} (\mathbf{0}, \sigma_{\texttt{out}}^2 \mathbf{I})$. Then, one can write

$$\mathbb{E}[z] = (1 - \delta_{\texttt{out}})\, \mu \quad \text{and} \quad \text{Var}[z] = (1 - \delta_{\texttt{out}})\, \sigma^2 \mathbf{I} + \delta_{\texttt{out}} \sigma_{\texttt{out}}^2 \mathbf{I} + \frac{(N - M)M}{N^2} \mu\mu^T.$$

Since $\|x + y\| \leq \|x\| + \|y\|$, $p(z)$ is also L_1 measurable. By the Strong Law of Large Numbers[6], $\bar{\mu} \overset{a.s.}{\to} \mathbb{E}[z]$ *as* $N \to \infty$, and one can easily induce that $\bar{\mathbf{\Sigma}} \overset{a.s.}{\to} \text{Var}[z]$ *as* $N \to \infty$.

Similarly, we can obtain $\widehat{\mu}$ and $\widehat{\mathbf{\Sigma}}$ as below. Consider a event E_p be a set of the MCD estimators contain outliers with the fraction p. Then, we have a random variable Z_p again, which is from a mixture distribution of two distributions, where its pdf $p(Z_p)$ is consisted of $(1 - p)\mathcal{N}(\mu, \sigma^2 \mathbf{I}) + p(\mathbf{0}, \sigma_{\texttt{out}}^2 \mathbf{I})$. Then,

$$\mathbb{E}[Z_p] = (1 - p)\mu \quad \text{and} \quad \text{Var}[Z_p] = (1 - p)\sigma^2 \mathbf{I} + p\sigma_{\texttt{out}}^2 \mathbf{I} + p(1 - p)\mu\mu^T.$$

Therefore, for the all MCD estimator $\widehat{\mu},\ \widehat{\mathbf{\Sigma}} \in E_p$, we have

$$\widehat{\mu} \overset{a.s.}{\to} \mathbb{E}[Z_p] \qquad \text{and} \qquad \widehat{\mathbf{\Sigma}} \overset{a.s.}{\to} \text{Var}[Z_p] \qquad as \quad N \to \infty.$$

By definition (2), the objective of the MCD estimator corresponds to their determinant of the covariance matrix. Note that the determinant is continuous funciton. Then, by the Continuous Mapping Theorem[7],

$$det\left(\widehat{\mathbf{\Sigma}}\right) \overset{a.s.}{\to} \lim_{N \to \infty} det\left(\widehat{\mathbf{\Sigma}}\right) = det\left(\text{Var}[Z_p]\right), \qquad as\ N \to \infty,$$

and,

$$\begin{aligned} \det\left(\text{Var}[Z_p]\right) &= \det\left((1 - p)\sigma^2 \mathbf{I} + p\sigma_{\texttt{out}}^2 \mathbf{I} + p(1 - p)\mu\mu^T\right) \\ &= \left((1 - p)\sigma^2 + p\sigma_{\texttt{out}}^2\right)^{d-1} \left((1 - p)\sigma^2 + p\sigma_{\texttt{out}}^2 + p(1 - p)\mu^T\mu\right). \end{aligned}$$

For simplicity, consider a polynomial function of p as follows:

$$f(p) = \left((1 - p)\sigma^2 + p\sigma_{\texttt{out}}^2 + p(1 - p)\mu^T\mu\right) \left((1 - p)\sigma^2 + p\sigma_{\texttt{out}}^2\right)^{d-1}.$$

[6] https://en.wikipedia.org/wiki/Law_of_large_numbers#Strong_law
[7] https://en.wikipedia.org/wiki/Continuous_mapping_theorem

Since $\sigma^2 \leq \sigma_{\texttt{out}}^2$, we have

$$\begin{aligned} f(p) &\geq \left((1-p)\sigma^2 + p\sigma^2 + p(1-p)\mu^T\mu\right)\left((1-p)\sigma^2 + p\sigma^2\right)^{d-1} \\ &\geq \left((1-p)\sigma^2 + p\sigma^2\right)\left((1-p)\sigma^2 + p\sigma^2\right)^{d-1} = f(0). \end{aligned}$$

Note that

$$\min_{\mathcal{X}_K \subset \mathcal{X}_N} \det(\widehat{\mathbf{\Sigma}}) \overset{a.s.}{\to} \min_{E_p} \det\left(\mathrm{Var}[Z_p]\right), .$$

Then, for $\delta_{\texttt{mcd}} \leq 1 - \delta_{\texttt{out}}$, the MCD estimators in E_0 converges to a minimum determinant of the covariance matrix of Z_p among E_p. Therefore, as N goes to infinity,

$$\min_{\mathcal{X}_K \subset \mathcal{X}_N} \det(\widehat{\mathbf{\Sigma}}) \overset{a.s.}{\to} \min_{E_p} \det\left(\mathrm{Var}[Z_p]\right) = \det\left(\mathrm{Var}[Z_0]\right),$$

and,

$$\widehat{\mu} \overset{a.s.}{\to} \mathbb{E}[Z_0] = \mu \quad \text{and} \quad \widehat{\mathbf{\Sigma}} \overset{a.s.}{\to} \mathrm{Var}[Z_0] = \sigma^2\mathbf{I}.$$

□

Now, we start to prove Theorem 1 with Lemma 1.

First, to show (4), we use Lemma 1 and the assumptions $\mathcal{A}3$ and $\mathcal{A}5$. Then, we have$\bar{\mu}_c \overset{a.s.}{\to} (1-\delta_{\texttt{out}})\,\mu_c$ and $\widehat{\mu}_c \overset{a.s.}{\to} \mu_c$. One can easily induce that

$$\lim_{N\to\infty} \|\mu_c - \widehat{\mu}_c\|_1 = 0, \qquad \lim_{N\to\infty} \|\mu_c - \bar{\mu}_c\|_1 = \delta_{\texttt{out}}\|\mu_c\|_1.$$

Second, to show (5), consider C number of Gaussian distributions $\mathcal{N}\left(\mu_c, \sigma^2\mathbf{I}\right)$ with $\frac{1}{C}\sum_c \mu_c = \mathbf{0}$, by the assumptions $\mathcal{A}1$ and $\mathcal{A}2$. To obtain tied covariance matrix $\bar{\mathbf{\Sigma}}$ and $\widehat{\mathbf{\Sigma}}$, we gather $\bar{\mathbf{\Sigma}}_c$ and $\widehat{\mathbf{\Sigma}}_c$ for each class c as follow,

$$\bar{\mathbf{\Sigma}} = \frac{\sum_c N_c\bar{\mathbf{\Sigma}}_c}{\sum_c N_c} = \frac{\sum_c \bar{\mathbf{\Sigma}}_c}{C}, \quad \widehat{\mathbf{\Sigma}} = \frac{\sum_c K_c\widehat{\mathbf{\Sigma}}_c}{\sum_c K_c} = \frac{\sum_c \widehat{\mathbf{\Sigma}}_c}{C}.$$

Then by Lemma 1, one can write

$$\begin{aligned} \lim_{N\to\infty} \bar{\mathbf{\Sigma}} &= \left((1-\delta_{\texttt{out}})\,\sigma^2 + \delta_{\texttt{out}}\sigma_{\texttt{out}}^2\right)\mathbf{I} + \delta_{\texttt{out}}\,(1-\delta_{\texttt{out}})\,\frac{1}{C}\sum_c \mu_c\mu_c^T, \\ \lim_{N\to\infty} \widehat{\mathbf{\Sigma}} &= \sigma^2\mathbf{I}. \end{aligned}$$

By the assumption $\mathcal{A}2$, $\frac{1}{C}\sum_c \mu_c\mu_c^T = \mathbf{D}$ where $\mathbf{D}$ is a diagonal matrix. Then $\lim_{N\to\infty}\bar{\mathbf{\Sigma}}$ is a diagonal matrix, and

$$\lim_{N\to\infty} \bar{\mathbf{\Sigma}}^{-1} = \left(\left((1-\delta_{\texttt{out}})\,\sigma^2 + \delta_{\texttt{out}}\sigma_{\texttt{out}}^2\right)\mathbf{I} + \delta_{\texttt{out}}\,(1-\delta_{\texttt{out}})\,\mathbf{D}\right)^{-1}.$$

Note that

$$\begin{aligned} \lim_{N\to\infty} (\bar{\mu}_i - \bar{\mu}_c)^T(\bar{\mu}_i - \bar{\mu}_c) &= (1-\delta_{\texttt{out}})^2\,(\mu_i - \mu_c)(\mu_i - \mu_c)^T, \\ \lim_{N\to\infty} (\widehat{\mu}_i - \widehat{\mu}_c)^T(\widehat{\mu}_i - \widehat{\mu}_c) &= (\mu_i - \mu_c)(\mu_i - \mu_c)^T, \end{aligned}$$

and,

$$\begin{aligned} \lim_{N\to\infty} (\bar{\mu}_c - \bar{\mu}_{c'})^T\bar{\mathbf{\Sigma}}^{-1}(\bar{\mu}_c - \bar{\mu}_{c'}) &= \lim_{N\to\infty} tr\left(\bar{\mathbf{\Sigma}}^{-1}(\bar{\mu}_c - \bar{\mu}_{c'})(\bar{\mu}_c - \bar{\mu}_{c'})^T\right) \\ &= \sum_i \frac{(1-\delta_{\texttt{out}})^2\,(\mu_{ci} - \mu_{c'i})^2}{(1-\delta_{\texttt{out}})\,\sigma^2 + \delta_{\texttt{out}}\sigma_{\texttt{out}}^2 + \delta_{\texttt{out}}\,(1-\delta_{\texttt{out}})\,D_i}, \end{aligned}$$

where $\mathbf{D} = \text{diag}\,(D_1, ..., D_d)$. By the assumption $\mathcal{A}4$, for $\forall\, i$,

$$(1-\delta_{\texttt{out}})\,\sigma^2 + \delta_{\texttt{out}}\sigma_{\texttt{out}}^2 + \delta_{\texttt{out}}\,(1-\delta_{\texttt{out}})\,D_i \geq \sigma^2.$$

Then we have

$$\begin{aligned}
\lim_{N\to\infty}(\bar{\mu}_c - \bar{\mu}_{c'})^T\bar{\mathbf{\Sigma}}^{-1}(\bar{\mu}_c - \bar{\mu}_{c'}) &= \lim_{N\to\infty} tr\left(\bar{\mathbf{\Sigma}}^{-1}(\bar{\mu}_c - \bar{\mu}_{c'})(\bar{\mu}_c - \bar{\mu}_{c'})^T\right) \\
&= \sum_i \frac{(1-\delta_{\texttt{out}})^2\,(\mu_{ci} - \mu_{c'i})^2}{(1-\delta_{\texttt{out}})\,\sigma^2 + \delta_{\texttt{out}}\sigma_{\texttt{out}}^2 + \delta_{\texttt{out}}\,(1-\delta_{\texttt{out}})\,D_i} \leq \lim_{N\to\infty}\frac{(\bar{\mu}_i - \bar{\mu}_c)^T(\bar{\mu}_i - \bar{\mu}_c)}{\sigma^2}, \\
\lim_{N\to\infty}(\widehat{\mu}_i - \widehat{\mu}_c)^T\widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_i - \widehat{\mu}_c) &= \lim_{N\to\infty} tr\left(\widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_i - \widehat{\mu}_c)(\widehat{\mu}_i - \widehat{\mu}_c)^T\right) = \frac{(\mu_i - \mu_c)^T(\mu_i - \mu_c)}{\sigma^2}.
\end{aligned}$$

Therefore,

$$\lim_{N\to\infty}\frac{(\bar{\mu}_c - \bar{\mu}_{c'})^T\bar{\mathbf{\Sigma}}^{-1}(\bar{\mu}_c - \bar{\mu}_{c'})}{(\widehat{\mu}_c - \widehat{\mu}_{c'})^T\widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_c - \widehat{\mu}_{c'})} \leq \lim_{N\to\infty}\frac{\frac{(\bar{\mu}_i - \bar{\mu}_c)^T(\bar{\mu}_i - \bar{\mu}_c)}{\sigma^2}}{\frac{(\widehat{\mu}_i - \widehat{\mu}_c)^T(\widehat{\mu}_i - \widehat{\mu}_c)}{\sigma^2}} = (1-\delta_{\texttt{out}})^2 \leq 1.$$

Third, to show (3), we define estimated error $B(\widehat{\mu}_c, \widehat{\mu}_{c'}, \mathbf{\Sigma}, \widehat{\mathbf{\Sigma}})$ as follow,

$$B(\widehat{\mu}_c, \widehat{\mu}_{c'}, \mathbf{\Sigma}, \widehat{\mathbf{\Sigma}}) := \exp\left(-\frac{1}{8}\frac{[(\widehat{\mu}_c - \widehat{\mu}_{c'})^T\widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_c - \widehat{\mu}_{c'})]^2}{(\widehat{\mu}_{c'} - \widehat{\mu}_c)^T\widehat{\mathbf{\Sigma}}^{-1}\mathbf{\Sigma}\widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_{c'} - \widehat{\mu}_c)}\right).$$

Note that

$$\frac{[(\widehat{\mu}_c - \widehat{\mu}_{c'})^T\widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_c - \widehat{\mu}_{c'})]^2}{(\widehat{\mu}_{c'} - \widehat{\mu}_c)^T\widehat{\mathbf{\Sigma}}^{-1}\mathbf{\Sigma}\widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_{c'} - \widehat{\mu}_c)} = \frac{tr\left(\widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_c - \widehat{\mu}_{c'})(\widehat{\mu}_c - \widehat{\mu}_{c'})^T\right)^2}{tr\left(\widehat{\mathbf{\Sigma}}^{-1}\mathbf{\Sigma}\widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_c - \widehat{\mu}_{c'})(\widehat{\mu}_c - \widehat{\mu}_{c'})^T\right)}.$$

Then,

$$\begin{aligned}
\lim_{N\to\infty} -8\log(B(\bar{\mu}_c, \bar{\mu}_{c'}, \mathbf{\Sigma}, \bar{\mathbf{\Sigma}})) &= \lim_{N\to\infty}\frac{tr\left(\bar{\mathbf{\Sigma}}^{-1}(\bar{\mu}_c - \bar{\mu}_{c'})(\bar{\mu}_c - \bar{\mu}_{c'})^T\right)^2}{tr\left(\bar{\mathbf{\Sigma}}^{-1}\sigma^2\mathbf{I}\bar{\mathbf{\Sigma}}^{-1}(\bar{\mu}_c - \bar{\mu}_{c'})(\bar{\mu}_c - \bar{\mu}_{c'})^T\right)} \\
&= \frac{(1-\delta_{\texttt{out}})^2}{\sigma^2}\frac{\left(\sum_i \frac{(\mu_{ci} - \mu_{c'i})^2}{(1-\delta_{\texttt{out}})\sigma^2 + \delta_{\texttt{out}}\sigma_{\texttt{out}}^2 + \delta_{\texttt{out}}(1-\delta_{\texttt{out}})D_i}\right)^2}{\sum_i \frac{(\mu_{ci} - \mu_{c'i})^2}{((1-\delta_{\texttt{out}})\sigma^2 + \delta_{\texttt{out}}\sigma_{\texttt{out}}^2 + \delta_{\texttt{out}}(1-\delta_{\texttt{out}})D_i)^2}}, \\
\lim_{N\to\infty} -8\log(B(\widehat{\mu}_c, \widehat{\mu}_{c'}, \mathbf{\Sigma}, \widehat{\mathbf{\Sigma}})) &= \frac{tr\left((\sigma^2\mathbf{I})^{-1}(\mu_c - \mu_{c'})(\mu_c - \mu_{c'})^T\right)^2}{tr\left((\sigma^2\mathbf{I})^{-1}\sigma^2\mathbf{I}(\sigma^2\mathbf{I})^{-1}(\mu_c - \mu_{c'})(\mu_c - \widehat{\mu}_{c'})^T\right)} \\
&= \frac{(\mu_c - \mu_{c'})^T(\mu_c - \mu_{c'})}{\sigma^2}.
\end{aligned}$$

By the Cauchy-Schwarz inequality[8],

$$\begin{aligned}
\sum_i(\mu_{ci} - \mu_{c'i})^2\sum_i\left(\frac{\mu_{ci} - \mu_{c'i}}{(1-\delta_{\texttt{out}})\,\sigma^2 + \delta_{\texttt{out}}\sigma_{\texttt{out}}^2 + \delta_{\texttt{out}}\,(1-\delta_{\texttt{out}})\,D_i}\right)^2 \\
\geq \left(\sum_i\frac{(\mu_{ci} - \mu_{c'i})^2}{(1-\delta_{\texttt{out}})\,\sigma^2 + \delta_{\texttt{out}}\sigma_{\texttt{out}}^2 + \delta_{\texttt{out}}\,(1-\delta_{\texttt{out}})\,D_i}\right)^2.
\end{aligned}$$

Then we get

$$\lim_{N\to\infty} -8\log(B(\widehat{\mu}_c, \widehat{\mu}_{c'}, \mathbf{\Sigma}, \widehat{\mathbf{\Sigma}})) \geq \lim_{N\to\infty} -8\log(B(\bar{\mu}_c, \bar{\mu}_{c'}, \mathbf{\Sigma}, \bar{\mathbf{\Sigma}})),$$

[8] https://en.wikipedia.org/wiki/Cauchy-schwarz_inequality

and

$$\lim_{N\to\infty} -8\log\frac{B(\widehat{\mu}_c,\widehat{\mu}_{c'},\mathbf{\Sigma},\widehat{\mathbf{\Sigma}})}{B(\bar{\mu}_c,\bar{\mu}_{c'},\mathbf{\Sigma},\bar{\mathbf{\Sigma}})} \geq \frac{(\mu_c-\mu_{c'})^T(\mu_c-\mu_{c'})}{\sigma^2}\left(1-(1-\delta_{\texttt{out}})^2\right).$$

It implies $\lim_{N\to\infty} B(\widehat{\mu}_c,\widehat{\mu}_{c'},\mathbf{\Sigma},\widehat{\mathbf{\Sigma}}) \leq \lim_{N\to\infty} B(\bar{\mu}_c,\bar{\mu}_{c'},\mathbf{\Sigma},\bar{\mathbf{\Sigma}})$, and

$$\lim_{N\to\infty}\frac{B(\widehat{\mu}_c,\widehat{\mu}_{c'},\mathbf{\Sigma},\widehat{\mathbf{\Sigma}})}{B(\bar{\mu}_c,\bar{\mu}_{c'},\mathbf{\Sigma},\bar{\mathbf{\Sigma}})} \leq \exp\left(-\frac{1}{8}\frac{(\mu_c-\mu_{c'})^T(\mu_c-\mu_{c'})}{\sigma^2}\left(1-(1-\delta_{\texttt{out}})^2\right)\right) \leq 1.$$

On the other hand, we have

$$\bar{\mu_c} \overset{a.s.}{\to} \lim_{N\to\infty}\bar{\mu_c}, \qquad \widehat{\mu_c} \overset{a.s.}{\to} \lim_{N\to\infty}\widehat{\mu_c}$$
$$\bar{\mathbf{\Sigma}_c} \overset{a.s.}{\to} \lim_{N\to\infty}\bar{\mathbf{\Sigma}_c}, \qquad \widehat{\mathbf{\Sigma}_c} \overset{a.s.}{\to} \lim_{N\to\infty}\widehat{\mathbf{\Sigma}_c}.$$

by Lemma 1. Then by the Continuous Mapping Theorem[9],

$$\bar{\mathbf{\Sigma}} \overset{a.s.}{\to} \lim_{N\to\infty}\bar{\mathbf{\Sigma}}, \qquad \widehat{\mathbf{\Sigma}} \overset{a.s.}{\to} \lim_{N\to\infty}\widehat{\mathbf{\Sigma}},$$

and,

$$\|\mu_c-\widehat{\mu}_c\|_1 \overset{a.s.}{\to} \lim_{N\to\infty}\|\mu_c-\widehat{\mu}_c\|_1, \qquad \|\mu_c-\bar{\mu}_c\|_1 \overset{a.s.}{\to} \lim_{N\to\infty}\|\mu_c-\bar{\mu}_c\|_1,$$
$$\frac{(\bar{\mu}_c-\bar{\mu}_{c'})^T\bar{\mathbf{\Sigma}}^{-1}(\bar{\mu}_c-\bar{\mu}_{c'})}{(\widehat{\mu}_c-\widehat{\mu}_{c'})^T\widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_c-\widehat{\mu}_{c'})} \overset{a.s.}{\to} \lim_{N\to\infty}\frac{(\bar{\mu}_c-\bar{\mu}_{c'})^T\bar{\mathbf{\Sigma}}^{-1}(\bar{\mu}_c-\bar{\mu}_{c'})}{(\widehat{\mu}_c-\widehat{\mu}_{c'})^T\widehat{\mathbf{\Sigma}}^{-1}(\widehat{\mu}_c-\widehat{\mu}_{c'})},$$
$$\frac{B(\widehat{\mu}_c,\widehat{\mu}_{c'},\mathbf{\Sigma},\widehat{\mathbf{\Sigma}})}{B(\bar{\mu}_c,\bar{\mu}_{c'},\mathbf{\Sigma},\bar{\mathbf{\Sigma}})} \overset{a.s.}{\to} \lim_{N\to\infty}\frac{B(\widehat{\mu}_c,\widehat{\mu}_{c'},\mathbf{\Sigma},\widehat{\mathbf{\Sigma}})}{B(\bar{\mu}_c,\bar{\mu}_{c'},\mathbf{\Sigma},\bar{\mathbf{\Sigma}})}.$$

This completes the proof of Theorem 1.

[9] https://en.wikipedia.org/wiki/Continuous_mapping_theorem