# Corrupted CIFAR Datasets Classification

Dekai Meng
Department of Computing and Software
McMaster University
400376431

Hongshan Shang
Department of Computing and Software
McMaster University
400384172

https://github.com/dekaimeng/Corrupted-CIFAR-Datasets-Classification

***Abstract*—Cifar-10 and Cifar-100 datasets are two image collections, which are widely used in the machine learning area. The size of Cifar-10 contains 60,000 thirty by thirty colour images in 10 classes. On the other hand, Cifar-100 is similar, but it has 100 classes containing 600 images per class.**

**Deep neural networks(DNNs) currently stand on a high-performance role in image classification. In supervised image classification, we always learn the model with a clean training set. But normally, we cannot guarantee the dataset we train is always high-quality with less noisy labels, which will result in the low performance of prediction and classification accuracy, to be specific, our DNNs will be easily overfitted by the data with noisy labels. Unfortunately, that is extremely time-consuming and inefficient if we obtain a high-quality dataset with manual labels.**

**To deal with those cases, we reference the paper "Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks" [1] and "An Introduction to Confident Learning: Finding and Learning with Label Errors in Datasets" [2]. Those papers introduced a new framework in the Deep Learning area, called CleanLab, which can be used to justify and characterize the noisy labels and learn with that by the confident learning method.**

## I. Introduction

The measurement of justifying the deep neural network mainly depends on the prediction accuracy, but that will be strongly affected by the quality of training data. As our data is not clean and contains various meaningless data, the model's accuracy will be greatly reduced. Thus, the priority of improving the model performance is to handle the noisy label in our training dataset.

Our method is based on the framework for machine learning in python library, called CleanLab, the main idea is to use the multiple cross-validation method based on the CleanLab to identify the possible noisy sample [1], then we delete a part of noisy samples from our datasets and retrain the CNN model based on our clean dataset to achieve the accuracy [2].

To achieve the highest performance for the CleanLab, we need to identify the noisy label based on sample similarity. Firstly, we need to train a similarity model, then sample a batch of black samples with high confidence and calculate the similarity between the black and white samples. The next step is to identify the black samples whose similarity to the while one is greater than the threshold and can match on the while samples as many times. These sets of black samples might be the noisy labels' data.

We had summarized three possible methods. After testing three methods called SCE loss, GMM with the CleanLab and the CNN cross-validation with the CleanLab, we found that CNN cross-validation will be the best option to increase the accuracy.

## II. CleanLab Framework in Python

The CleanLab is a useful python package in the machine learning area, which can locate and process the noisy labels [2], then we can change or delete them after we get clear information about the possible noisy labels. We can understand the CleanLab as a framework for Confident Learning, which will be introduced later.

There are two notions of confidence in this package, one is confident examples, and another one is confident errors. Confident examples that we are confident are labelled correctly, and the confident errors are the examples we are confident are labelled wrong.

The input of CleanLab is a matrix of out-of-sample predicted probabilities for each sample in each class, and an array of noisy labels for each sample, which is easier to understand and use compared to many of the approaches frameworks Northcutt, C. G. [3].

In our method, we mainly used the function called get_noise_indices to find and clean the noisy labels in our dataset.

Python example code:

```
From cleanlab.pruning import
    get_noise_indices
get_noise_indices(y, probas, prune_method
    ='both', frac_noise=1, n_jobs=1)
```

Import and process the get_noise_indices function (Hint: for the frac_noise parameter, the best value we tried from range (0,1] is 1, which means return all 'confident' estimated noise indices)

We will import the get_noise_indices function from CleanLab, then we can use that to detect the noisy labels, to get two sets of data which indicate data and labels that are not noisy.

## III. The Previous Method We Researched

### *A. Symmetric Cross-Entropy(SCE) Loss Method*

The first time I got in touch with the method is from the paper called 'Symmetric Cross-Entropy for Robust Learning with Noisy Labels', which is simpler and easier to use compared with the others, we only need to modify a little from the previous cross-entropy loss [4]. Unfortunately, after our test, the accuracy was not considerable, even worse than the cross-entropy loss to train. On the other hand, this method only works in one dataset, but the accuracy will be lower than using the noisy label to train the model directly. Thus, we give up this method because that will not help our prediction accuracy have a dramatic increase.

### *B. Gaussian Mixture Modelling(GMM)+CleanLab*

Clustering in machine learning is known as a set of data points that need to be grouped into several parts or clusters based on their similarity [5], The Gaussian Mixture Modelling is one of the methods in clustering to discover the underlying groups of data, which is used to the data classification into different categories based on the probability distribution [6].

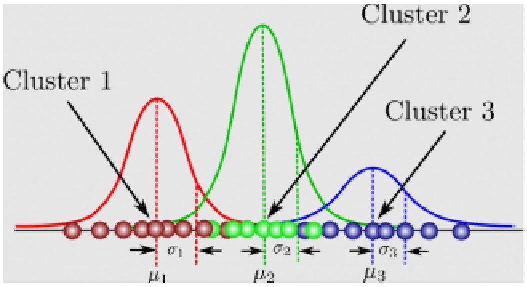


Fig. 1. Gaussian mixture models [6].

We can use the convolutional part of the pretrained resnet18 to be a feature extractor, then put the extracted features into the GMM for unsupervised training. The probabilities of each class in each sample will be output by GMM, and the value of probabilities can be the input of our CleanLab. Then we focus on the CleanLab from the paper. It is mainly based on the confidence learning theory. After we get the predicted probability for each sample in each category, we need to do cross-validation, which means that the predicted sample cannot be in our training set. In other words, we must separate a part of the training data set as the prediction data and another part as training. Then we can use k-fold cross-validation. (Hint: Each flod will have one process of training and prediction, to be specific, 5 flod will contain 5 times training and 5 times predicting).

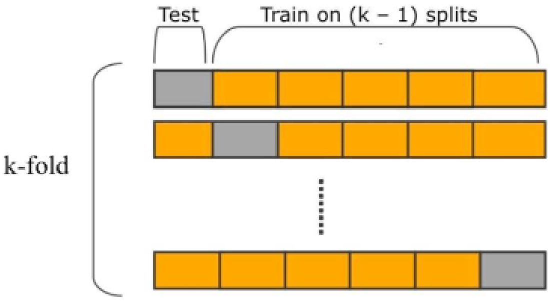


Fig. 2. 5 cross-validation [7].

If we can get a feature for the prediction and process the clustering which is similar to unsupervised learning, then we do not need to separate it every single time. Because unsupervised learning only needs to put all samples inside and predict that probability, then input that directly to the CleanLab.

**Algorithm 1** Construct the GMM and test the noisy index

```
(Hint: We will find a good performance of the toy dataset, but for our real dataset, that does not work.)
$x, y$ ← Generate isotropic Gaussian blobs for clustering.
nClass = The unique elements of data.
centers ← [] {Initialize the cluster centers}
loop
    $i$ of the data {The average of all points labeled $i$ is used as a center of this point, and this category of marks is used as a center of category $i$}
    centers[$i$, :] ← Arithmetic mean of the data along the specified axis.
end loop
gmm ← GussianMixture(nClass, $y$) {Get the Gaussian Mixture}
psx ← predict the probability(gmm)
```

After we processed this method, we found that because the GMM is unsupervised learning, which is not specified which class, means we cannot guarantee the consistency of the class. That will cause many mistakes if we input that in the CleanLab, we solve that by defining an initial cluster center [8]. Another problem is whether the features can be classified after extraction directly with resnet18 and pretrain. We put those features into a logistic regression and found the rate of classification is lower than expected, which means we cannot train the classifier by these features directly. Thus, we need to optimize the CNN, or the extracted features still cannot be used for classification.

**Algorithm 2** Find the rate of classification by logistic regression

```
kflod ← number
kf ← divides all the samples in $k$ groups of samples
loop
    in generate indices to split data into training and test set
    xtrain and ytrain ← obtain the value by the train index
    xtest ← obtain the value by test index
    model ← Logistic_regression_model(xtrain, ytrain)
    psx ← model.predict_probability(xtest)
end loop
```

For the methods of getting and comparing the real and predicted noisy index for both GMM and Logistic Regression, we applied the cross-validation method, which get the probability of prediction of $n * m$ and use the probability of prediction to the get_noisy_indices function and find the indices of array elements that are non-zero, grouped by element. Then get and compare the predicted noisy Index and real noisy index.

Those methods we researched are mainly for helping us know how to achieve the best performance for the CleanLab. The prediction accuracy is still our goal. So, after careful consideration, this new method is only retained as a process and ideas, but not incorporated into the test code.

## IV. The Main Idea of Our Method

Finally, we used resnet [1] to train our noisy data with a cross-validation method and use it to predict the probability value of each data belonging to each category probability. To be specific, for the Cifar-10 dataset, if we have several n samples, we should output an $n * 10$ matrix, each row represents a sample and the number of 10 represent the probability of each category respectively.

### A. K-Flod Cross-validation Method

The K-Fold cross-validation method is a kind of technique in the machine learning area, which is a type of cross-validation. In this method, we can split the original data into several $k$ subsamples with equal sizes, which can be used for the training data. And we can use the number of $k - 1$ subsamples for each fold to estimate the model and serve the validation sample using the rest $k$th subsample. Then we just repeat those processes by $k$ times to achieve that every subsample was processed as the validation data and the result of our model will be averaged across folds [9].

**Algorithm 3** Train the cross-validation

```
Input: trainDataLoaders, validateDataLoaders for cross-validation
loop
    $k$ in length(trainDataloaders).
    trainDataLoader[$k$] $\leftarrow$ trainDataloaders[$k$]
    validDataLoader[$k$] $\leftarrow$ validateDataLoaders[$k$]
    model $\leftarrow$ Classifier(resnet18)
    trainer $\leftarrow$ general trainer
    trainer.fit(trainDataLoader, validDataLoader) {train the model}
    model $\leftarrow$ trainer.model {predict the probability on validation set}
    Evaluate the model {Sets the model in evaluation mode}

    loop
        validDataLoader
        collect the probabilities of the validation data
    end loop
end loop
```

In addition, the $k$-fold cross-validation can also be extended by splitting the original data into a subset that undergoes the $k$-fold cross-validation process described above and into a subset that is used for evaluating the final model's performance, where this final data subset is often called the test data.

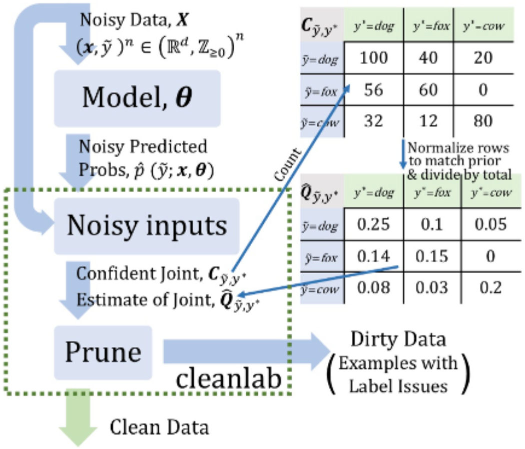
The $k$-fold cross-validation can also apply to different types of models such as logistic regression and linear regression models. The advantage of this method is effectively avoiding overfitting because all the data will be processed in training and prediction. The disadvantage is that the number of split $k$ is tricky, which will affect the balance of the bias and variance.

### B. Confident Learning

We learned confident learning in the paper called "An introduction to confident learning: finding and learning with label errors in datasets" [2]. This idea is based on a principled framework for identifying noisy labels, characterizing meaningless labels and applying that to achieve learning with noisy labels. There are many advantages of confident learning. The biggest advantage is that it can help us to find the mislabeled sample. And that is model-independent, which allows the use of arbitrary models, but unlike many noisy learning methods that are strongly coupled to the model and training process [10].

On the other hand, the prediction accuracy has been a dramatic increase compared with the other recent approach. The result of Confident Learning in the paper for Cifar10 dataset showed a better performance (30% higher in high noise dataset, and over 10% on average) than Google's top-performing Co-Teaching, MentorNet, Mix-up.. which shows in the table below [2].

For the structure of Confident Learning, that is based on training with confidence by pruning noisy data, counting to estimate noise and ranking examples.


Noisy Data, $\boldsymbol{X}$
$(\boldsymbol{x}, \tilde{y})^n \in (\mathbb{R}^d, \mathbb{Z}_{\geq 0})^n$
Model, $\boldsymbol{\theta}$
Noisy Predicted
Probs, $\hat{p}(\tilde{y}; \boldsymbol{x}, \boldsymbol{\theta})$
Noisy inputs
Confident Joint, $\boldsymbol{C}_{\tilde{y},y^*}$
Estimate of Joint, $\hat{\boldsymbol{Q}}_{\tilde{y},y^*}$
Prune
cleanlab
Dirty Data
(Examples with Label Issues)
Clean Data
Count
Normalize rows to match prior & divide by total

| $\boldsymbol{C}_{\tilde{y},y^*}$ | $y^*$=dog | $y^*$=fox | $y^*$=cow |
|---|---|---|---|
| $\tilde{y}$=dog | 100 | 40 | 20 |
| $\tilde{y}$=fox | 56 | 60 | 0 |
| $\tilde{y}$=cow | 32 | 12 | 80 |

| $\hat{\boldsymbol{Q}}_{\tilde{y},y^*}$ | $y^*$=dog | $y^*$=fox | $y^*$=cow |
|---|---|---|---|
| $\tilde{y}$=dog | 0.25 | 0.1 | 0.05 |
| $\tilde{y}$=fox | 0.14 | 0.15 | 0 |
| $\tilde{y}$=cow | 0.08 | 0.03 | 0.2 |

Fig. 3. The confident learning process and examples of the confident joint and estimated joint distribution between noisy (given) labels and uncorrupted (unknown) labels. $\bar{y}$ denotes an observed noisy label and $y*$ denotes a latent uncorrupted label [2].

The confident joint is defined as:

$$\boldsymbol{C}_{\tilde{y},y^*}[i][j] := \left|\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j}\right|$$

where

$$\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} := \{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j,$$
$$j = \underset{l \in [m]: \hat{p}(\tilde{y}=l; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_l}{\arg\max} \hat{p}(\tilde{y}=l; \boldsymbol{x}, \boldsymbol{\theta})\}$$

TABLE I
THE CONFIDENT LEARNING IMPROVES THE STATE-OF-THE-ART IN MACHINE LEARNING WITH NOISY LABELS BY 30% FOR THE HIGH NOISY DATASETS [2].

| NOISE | | 0.2 | | | | 0.4 | | | | 0.7 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPARSITY | **AVG** | 0 | 0.2 | 0.4 | 0.6 | 0 | 0.2 | 0.4 | 0.6 | 0 | 0.2 | 0.4 | 0.6 |
| **CL:** $C_{\text{CONFUSION}}$ | 0.662 | 0.854 | 0.854 | 0.863 | 0.857 | 0.806 | 0.796 | 0.802 | 0.798 | 0.332 | 0.363 | 0.328 | 0.291 |
| **CL:** $C_{\tilde{y},y^*}$ | 0.673 | 0.848 | 0.858 | 0.862 | 0.861 | **0.815** | **0.810** | **0.816** | 0.815 | 0.340 | 0.398 | 0.282 | **0.372** |
| **CL: OPT** | **0.696** | 0.860 | 0.859 | 0.865 | **0.862** | 0.810 | 0.801 | 0.814 | **0.825** | **0.468** | **0.420** | **0.399** | 0.371 |
| **SCE-LOSS** | 0.615 | **0.872** | **0.875** | **0.888** | 0.844 | 0.763 | 0.741 | 0.649 | 0.583 | 0.330 | 0.287 | 0.309 | 0.240 |
| **MIXUP** | 0.622 | 0.856 | 0.868 | 0.870 | 0.843 | 0.761 | 0.754 | 0.686 | 0.598 | 0.322 | 0.313 | 0.323 | 0.269 |
| **MENTORNET** | 0.590 | 0.849 | 0.851 | 0.832 | 0.834 | 0.644 | 0.642 | 0.624 | 0.615 | 0.300 | 0.316 | 0.293 | 0.279 |
| **Co-TEACHING** | 0.569 | 0.812 | 0.813 | 0.814 | 0.806 | 0.629 | 0.616 | 0.609 | 0.581 | 0.305 | 0.302 | 0.277 | 0.260 |
| **S-MODEL** | 0.556 | 0.800 | 0.800 | 0.797 | 0.791 | 0.586 | 0.612 | 0.591 | 0.575 | 0.284 | 0.285 | 0.279 | 0.273 |
| **REED** | 0.560 | 0.781 | 0.789 | 0.808 | 0.793 | 0.605 | 0.604 | 0.612 | 0.586 | 0.290 | 0.294 | 0.291 | 0.268 |
| **BASELINE** | 0.554 | 0.784 | 0.792 | 0.790 | 0.782 | 0.602 | 0.608 | 0.596 | 0.573 | 0.270 | 0.297 | 0.282 | 0.268 |

And the threshold $t_j$ is defined as the excepted self-confident for each class.

$$t_j = \frac{1}{|\boldsymbol{X}_{\tilde{y}=j}|} \sum_{\boldsymbol{x}\in \boldsymbol{X}_{\tilde{y}=j}} \hat{p}(\tilde{y}=j;\boldsymbol{x},\boldsymbol{\theta}) \qquad [2]$$

In order to achieve the Confident Learning with CleanLab, there are three main steps in total:

1) Estimate the joint distribution of noisy labels and the clean labels, to be specific, we will use a cross-validation method to process our dataset and come up with a confusion matrix.

   Then we need to make the count sum the same as the total number of manually labeled samples.

   The last step is to estimate the joint distribution of noisy labels and the real labels [2].
2) Find and filter out the samples with noisy labels into a clean dataset.
3) Retrain the clean dataset to achieve prediction accuracy.

## V. THE ROLE OF EACH FILE

Our package contains 2 folders and 11 python files in total.

The Datasets folder is the new dataset we defined that aims to protect the original data. We will introduce this folder more specifically in the 8$^{\text{th}}$ in part 6 (The difficulty we met and the solution).

The resulting folder contains all our results. It includes our history result and the final result with the graphs.

There are 10 new python files and the loader_for_CIFAR python file we keep unchanged.

S1CleanlabDemo.py:

This is an approach we have tried before and rejected. This file aims to verify the effect of CleanLab based on a toy dataset.

S2computeProbas.py:

Predict the probability value of each sample belonging to each class in the training set. This part mainly uses the pre-trained resnet18 as the classifier and uses 5-fold cross-validation for training and prediction.

S2computeProbasAll.py:

We run all the tasks of cifer10 and cifer100 together. Automatically run S2computeProbas in batches to reduce the number of experimental operations.

S3trainClearedData.py:

Load the predicted value of the previous step, determine possible noise labels based on CleanLab, and remove these noisy samples. The model is retrained on the new samples and tested on the validation set. In this step, we will propose the noise, calculate the noisy rate and produce the final accuracy.

S3trainClearedDataAll.py:

We run all the tasks of cifer10 and cifer100 together. Automatically run S3trainClearedData in batches to reduce the number of experimental operations.

S4trainNoisyData.py:

Train and test models based on noisy labels.

S5animalProcessing.py:

Uses the pre-trained resnet18 as the classifier to predict the probability value of each sample belonging to each class in the training set. Same work as S2computeProbas.py.

S6trainClearedanimal.py:

Eliminate samples. Retrain the model based on the newly constituted samples. Same work as S3trainClearedData.py.

network.py:

network structure. Define the number of classes, input image channels, classifier name and pretrained model weights.

trainers.py:

This is the main training part, which initializes some training parameters and then trains the trainDataLoader.

Plot.py:

This function is used to generate the resulting graph.

## VI. THE DIFFICULTY WE MET AND THE SOLUTION

During the repeated testing process, we encountered many difficulties

1) The CleanLab predicted as much normal data as the noisy data, to solve this problem, we used k-fold cross-validations to improve the probability of the model's prediction.
2) During the algorithm validation process, we need to deal with the time-consuming problem. To solve this problem, we first generate small-scale data and then randomly modify their label. Based on this noisy data, we tested and verified the algorithm and found the feasibility of the algorithm.

3) The training time for the model random initialization is pretty long, so we load the pre-trained to accelerate the convergence of the model.
4) Because we cannot reuse the dataset which was used for the clean dataset, we redefined a new dataset class and data loader. However, there were still other problems, such as forgetting to add the transform and implement the data augmentations, so that the model is very easy to be overfitting during training.
5) The accuracy of the training model is not considerable, we changed to a better classification model called densenet121, this model can fully combine the features of multiple levels.
6) The model code redundancy is a big problem for us. We searched on the website and designed the trainer class and classifier class, which reduce the code redundancy and increase the readability.
7) There are two problems we met during our processing of the training dataset that affects our accuracy result. The overfitting and the accuracy will fluctuate during the training, which the paper has not discussed yet, and as we know so far, the CleanLab doesn't have a specific method to deal with those problems. To deal with that, we tried to increase the number of multi-fold cross-validation may increase accuracy, which improves the reliability of probabilities from model prediction.
8) We also researched the batch size during our coding. We observed that if the batch size is small, the curve of loss will float a lot, but if we increase the batch size to be larger, that will be unacceptable to our machine for the training set. Thus, the value of batch size in a balanced situation is important, which can guarantee that the model is optimized in a meaningful direction every time and has the stochastic gradient descent feature to avoid the problem of overfitting.
   For the training dataset, we found that we cannot output directly by the data loader. Because the training data outputted by the data loader has been processed by data augmentation, which is no longer the original image anymore and affects our results. Thus, we define a new dataset when we need to use our original data, and the new data set that we defined is allowed to achieve the augmentations during the training.

## VII. The Improvement We Made Compared with the Original Methods

For future improvements, we found a better classification model: DenseNet during our research to improve our prediction accuracy.

Based on the result of the other related research paper (ex. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications), compared with classical ResNet, DenseNet models we found may bring up better features of the image and reduce the overfitting, which is more advanced and can perform better in ImageNet.

Secondly, We observed that if the batch size is small, the curve of loss will float a lot, but if we decrease the batch size to be smaller, that will be unacceptable to our machine for the training set. Currently, we set the batch size to 128 which is better. Thus, the value of batch size in a balanced situation is important.

## VIII. Final Result

After repeated trying we got the final accuracy of Cifar 10, Cifar 100 and animal 10 datasets are shown below:

### A. Test Result for the Cifar10 Dataset

1) Task 1: 82.5%

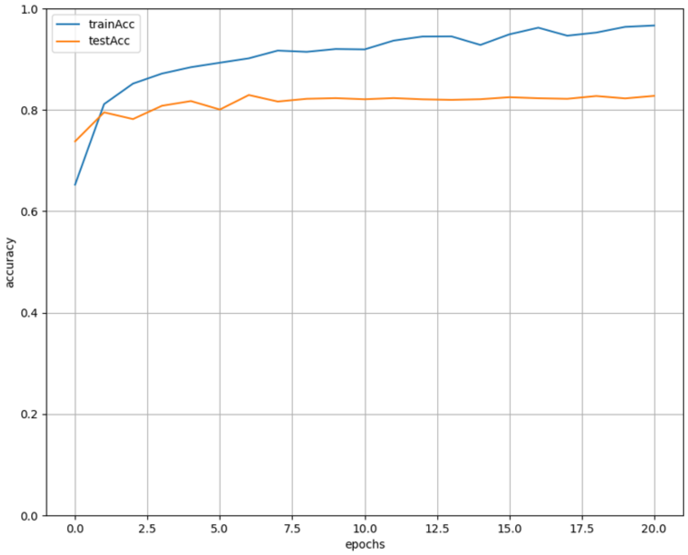


Fig. 4. Cifar10 dataset task1.

2) Task2: 64.3%

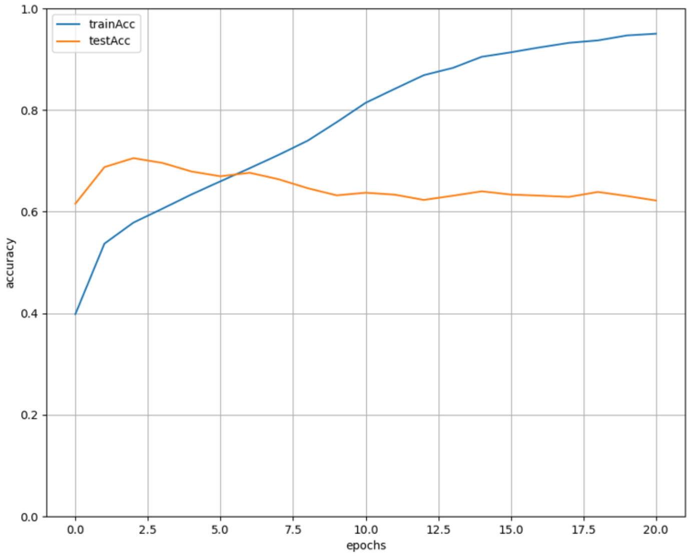


Fig. 5. Cifar10 dataset task2.

3) Task3: 80.2%

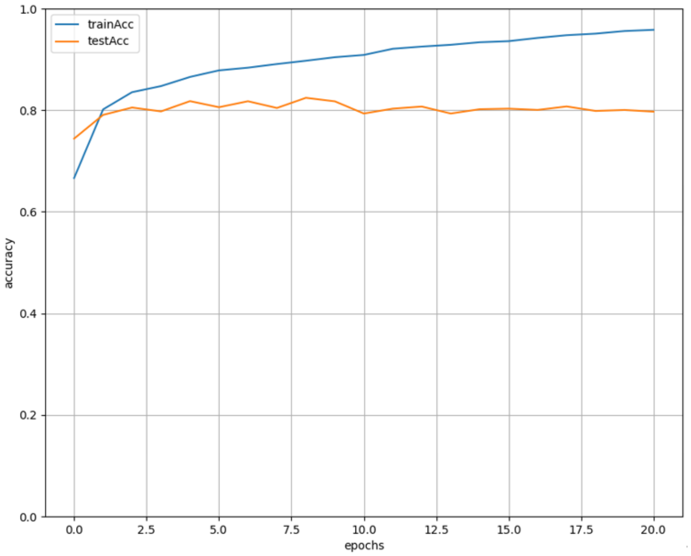


Fig. 6. Cifar10 dataset task3.

It is clearly shown from the figure that tasks 1 and 3 have better results. The accuracy of task2 will be relatively lower. It may be caused by many reasons. The noisy ratio is the most critical reason.

In our test, we did not determine the noisy rate of each noise label directly, However, because our method involves removing the noise first, we can calculate the noisy rate using our train data.

During our operation, as shown in the table below

| | Task1 | Task2 | Task3 |
|---|---|---|---|
| Num of raw train data | 50000 | 50000 | 50000 |
| Num of cleared train data | 28572 | 17329 | 39554 |
| Noisy rate | 42.856% | 65.342% | 20.892% |

From this table we can see that the first line is the number of raw train data. This is the total amount of data at the beginning.

The second line is the number of cleared train data. Here is how much clean data remains after our method has been used to clean the data.

The last line is the noisy rate after calculation using the previous two data.

Example: At the beginning, the total number of data is fifteen thousand and the data left after cleaning is 28572. Then the processed noise is equal to 50000-28572 and we get 21428. Then by dividing the result of the original data we can get the noise rate. In this example $21428/50000 = 0.42856$. So the noise rate in this example is 43%.

The noise rate for task two and three are 65% and 21% respectively.

Our method is not specifically used to distinguish the noise ratio, and we cannot perfectly clear the noise 100%, so we cannot guarantee the accuracy of the noise rate calculated by this method.

### B. Test Result for Cifar 100 Dataset

1) Task1: 51.6%

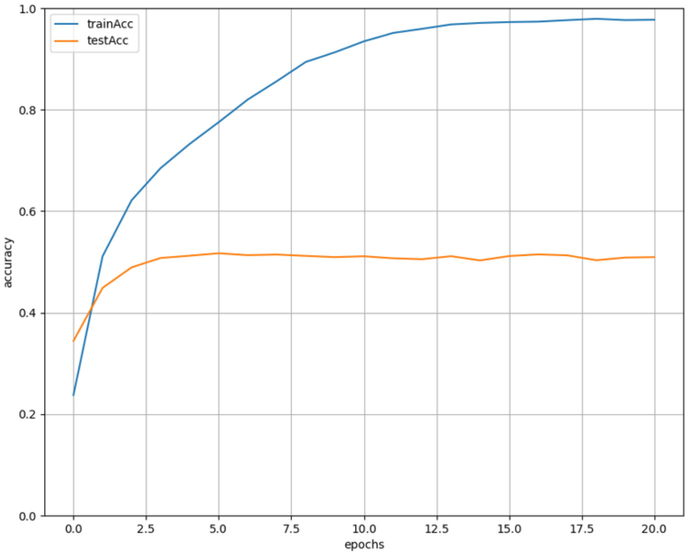


Fig. 7. Cifar100 dataset task1.

2) Task2: 26.4%

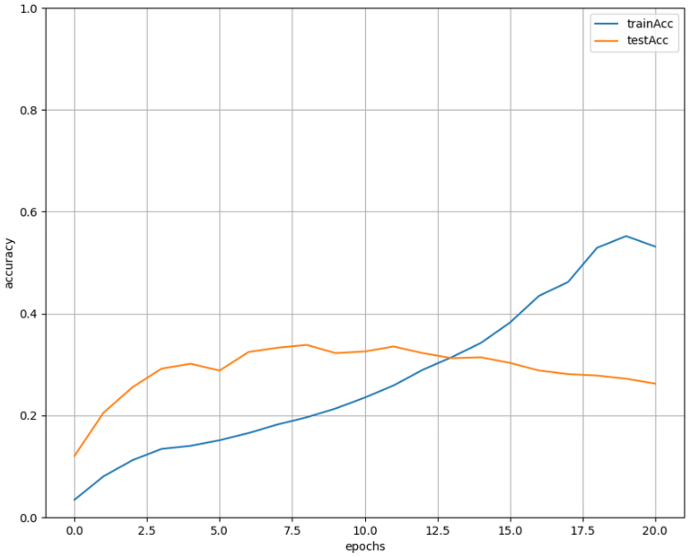


Fig. 8. Cifar100 dataset task2.

| | Task1 | Task2 |
|---|---|---|
| Num of raw train data | 50000 | 50000 |
| Num of cleared train data | 20697 | 21008 |
| Noisy rate | 58.606% | 57.984% |

### C. Test Result for Animal-10N Dataset

The noisy rate:

| | |
|---|---|
| Num of raw train data | 50000 |
| Num of cleared train data | 41153 |
| Noisy rate | 17.694% |

Accuracy: 80.1%

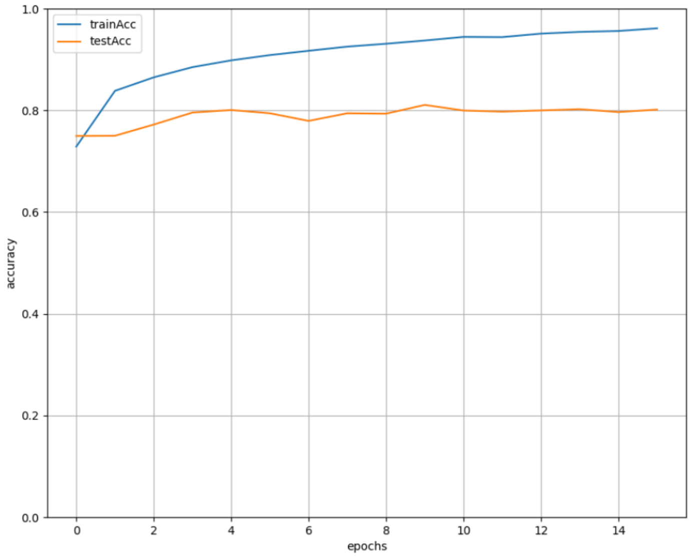


Fig. 9. Animal-10N dataset.

## IX. Conclusion

This project designed and implemented the algorithm to improve the classification performance for the Cifer10, Cifer100, and Animal-10N databases with a huge number of noisy labels. During the process, we made two attempts which are the method of Symmetric Cross-Entropy (SCE) loss method, and the Gaussian Mixture Modelling (GMM) + CleanLab method which is not relatively adopted for CleanLab. And finally, we selected the CNN cross-validation + CleanLab method.

For the training model, we finally chose the DenseNet model and the ResNet model for cross-validation. Throughout the process, we encountered various challenges and reported our solutions. And finally, we got better performance results with personal improvements based on the original article. The prediction accuracy has a dramatic increase with the calculation of the noisy rate for all the tasks in the Cifar10, Cifar100 datasets and the animal-10N dataset(Details in Final result part above). This solution may still have room for improvement(ex. a better classification model or larger batch size) to achieve better performance in the future.

## References

[1] C. G. Northcutt, A. Athalye, and J. Mueller, “Pervasive label errors in test sets destabilize machine learning benchmarks,” *arXiv preprint arXiv:2103.14749*, 2021.

[2] A. Athalye, T. Wu, N. Subrahmanya, and J. Mueller, “An introduction to confident learning: Finding and learning with label errors in datasets,” L7, November 2019. [Online]. Available: https://l7.curtisnorthcutt.com/confident-learning

[3] C. G. Northcutt, “Announcing cleanlab: A python package for ml and deep learning on datasets with label errors,” L7, November 2019. [Online]. Available: https://l7.curtisnorthcutt.com/cleanlab-python-package

[4] S. Gupta and A. Gupta, “Dealing with noise problem in machine learning data-SETS: A systematic review,” Procedia Computer Science, January 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919318575

[5] tufan_gupta2000, “Gaussian mixture model,” GeeksforGeeks, November 2021. [Online]. Available: https://www.geeksforgeeks.org/gaussian-mixture-model/

[6] A. Kumar, “Gaussian mixture models: What are they & when to use?” Data Analytics, September 2021. [Online]. Available: https://vitalflux.com/gaussian-mixture-models-what-are-they-when-to-use/

[7] S. Sossi Alaoui, Y. Farhaoui, and B. Aksasse, “Classification algorithms in data mining,” *Int. J. Tomogr. Simul.*, vol. 31, pp. 34–44, 2018.

[8] S. Mishra, “Unsupervised learning and data clustering,” Medium, May 2017. [Online]. Available: https://towardsdatascience.com/unsupervised-learning-and-data-clustering-eeecb78b422a

[9] J. Brownlee, “A gentle introduction to k-fold cross-validation,” Machine Learning Mastery, May 2018. [Online]. Available: https://machinelearningmastery.com/k-fold-cross-validation/

[10] C. Northcutt, L. Jiang, and I. Chuang, “Confident learning: Estimating uncertainty in dataset labels,” *J. Artif. Intell. Res.*, vol. 70, pp. 1373–1411, 2021.