Learning to Reweight Examples for Robust Deep Learning

Manish Kumar Krishne Gowda, Purdue University, ECE Department, On Campus MS Student[1]

Abstract

Although Deep Neural Networks are very efficient in classifying incoming data based on historical trained patterns, they can easily overfit when trained with imbalanced or noisy datasets.Regularizers and example reweighting algorithms are popular solutions to these problems. But these algorithms require careful tuning of additional hyperparameters. Traditionally, validation with a correctly labelled test set is performed at the end of each training epoch. This can be computationally inefficient if the example weights are treated as hyperparameters to optimize. Alternatively, the paper proposes a new technique for training deep neural networks which can be more robust to such unevenness in the training data. Three experiments were re-implemented with comparable trial conditions to ensure the accuracy and reliability of the proposed algorithm. The results of the demonstrate the effectiveness of the algorithm as outlined in the paper.

1. Introduction

Commercial deployments of Large Language Models(LLMs) have taken the world by storm off late. This is due to their ability to provide a more concise and insightful feedbacks to user queries compared to traditional search engines. Deep Neural Networks containing billions (sometimes trillions) of parameters are at the heart of these expert systems. The parameters are adjusted in training spanning over months to provide intelligence to the models. The models are said to 'learn' in this training phase and this forms the basis of an intelligent automated response that the LLMs are designed to output.

Such a learning forms an integral part of all Machine Learning systems in general. An LLM is just a 'small' example that in recent times has grabbed the attention of many people - from layperson to scholars. It is this learning that embeds what is termed as artificial intelligence in such systems. Many learning algorithms are proposed in literature and researchers are consistently engaged in improving (i.e. optimizing) these algorithms or propose novel ones. The paper "Learning to Reweight Examples for Robust Deep Learning" proposes one such optimization to the supervised learning algorithm that is employed to train the Deep Neural Networks.

Conventional deep learning methods often fail when they encounter new data that differs from the training set. In practical scenarios, such a disparity between training and can occur due to variations in the environment, lighting, or other factors. To address this problem, the authors propose a method called "example reweighting". This "Example Reweighting" method adjusts the weights given to each training example during the learning process.

The idea behind example reweighting is to assign higher weights to examples that are difficult to learn or that have been misclassified during training, and lower weights to examples that are easier to learn or that have been classified correctly. By doing so, the neural network is made to focus more on the difficult examples and to learn a more robust representation of the data.

The learning algorithm in the paper learns to assign weights to training examples based on their gradient directions. They perform gradient descent on the mini-batch example weights which are initialized to zero. Thereby they minimize the loss on clean unbiased validation set.

The model in the paper is derived from a meta-learning objective towards an online approximation that can fit into any regular supervised training. Instead of minimizing the expected loss for the training set each input example is weighted equally and this reweighted input is used to model the dataset. There are multiple examples of automatic differentiation techniques which will be the correct starting point as it is needed to compute the gradient of the validation loss.

In the re-implementation experiments, both class imbalance and corrupted label problems are realised and it is found that the new approach significantly increases the robustness to training set biases.

2. Related work

The paper itself references a multiple of other papers to build upon their proposal. Several of them implement regularization, reweighting, hyperparameter tuning, etc using popular deep learning frameworks such as TensorFlow.

While the main paper, Internet and ChatGPT were sufficient to understand the probalistic and mathematical background of the paper, (Natarajan et al., 2013) and (Angluin & Laird, 1988) were helpful in understanding the meaning of noisy label and uniform flip. Another work, unmentioned in the main paper, (Lenc & Vedaldi, 2015) was also helpful in this regard. Additional information in these papers was not studied in detail. Other aspects of these papers were not studied in detail, as they were not crucial to replicating the main study.

The main code of the paper can be found at https://tinyurl.com/3ny7ydw2. Authors have tested the algorithm on tensorflow 1.10 and python 3. Above referenced code was reviewed to gain a deeper understanding of the algorithm and the experiment details like the use of a 40% uniform flip, experiments involving data imbalance, background flipping, and noisy label analysis, among others.

Another unofficial PyTorch implementation is available at https://tinyurl.com/ym8zn47w. Here, the researcher has implemented the MNIST data imbalance experiment indicated in the paper and has been successful in reproducing the results. In accordance with the paper, a training dataset with an imbalanced distribution of classes was generated using class '4' and '9' of the MNIST dataset, with the '9' class being the predominant class. It was observed that the model achieves an accuracy of over 90% on the balanced test data, even when the training data has a dominant class proportion of 0.995. Result snippet is shown in Figure 1

This PyTorch implementation was used as the primary reference for this project and additional experiments were developed on top of this implementation

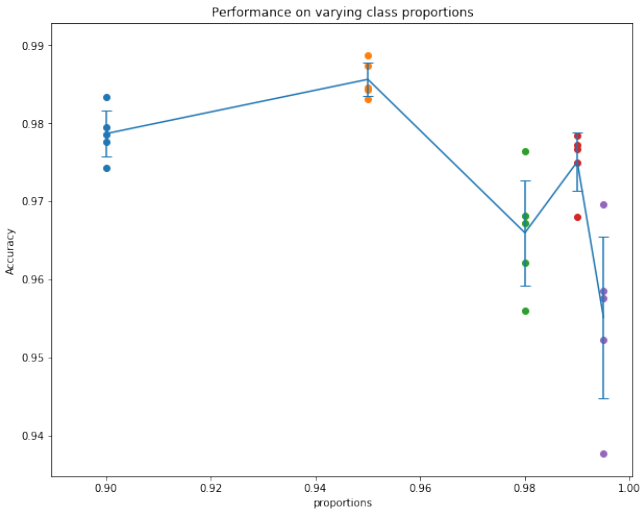

Figure 1. MNIST data imbalanced experiment

3. Method

While traditional reweighting methods use functions of the cost value of each example, current work propose a new meta-learning algorithm that learns to assign weights to training examples based on their gradient directions. To determine the example weights, firstly the example weights are initialised to zero. Later, they update the example weights by performing a meta gradient descent step on the existing mini-batch. These example weights are used to minimise the loss on a clean unbiased validation set.

Without delving deep into the mathematical derivations presented in the original paper, I reference the key results of a critical step in the algorithm, i.e. computing the gradients of the validation loss wrt the local perturbation ϵ.

$$\frac{\partial \mathbb{E}[f^v(\theta_{t+1}(\epsilon))\ |_{\epsilon_{i,t}=0}]}{\partial \epsilon_{i,t}} = \frac{-1}{m}\sum_{j=1}^{m}\sum_{l=1}^{L}(\tilde{z}^v_{j,l-1}{}^{\mathsf{T}})(z_{i,\tilde{l-1}})(g^v_{j,l}{}^{\mathsf{T}} g_{i,l}) \quad (1)$$

where we have:

1. parameters for each layer $\theta = (\theta_l)_{l=1}^{L}$
2. z_l is the pre-activation
3. $\tilde{z_l}$ is the post-activation
4. g_l is the gradient of loss wrt z_l

As per Eq 1, the meta-gradient on ϵ is composed of the sum of the products of two terms:

1. $z^{\mathsf{T}} z^v$: This computes the similarity between the training and validation inputs to the layer
2. $g^{\mathsf{T}} g^v$: This computes the similarity between the training and validation gradient directions

The computation graph in Fig 2 summarizes the workflow involved in training the deep learning models

3.1. Code Structure

The main algorithm is clearly spelt out in the original paper. This is the core of the paper and is shown in

The summary of the algorithm is :

1. Initialize the weights for each example in the training data. The weights are initially set to be equal for all examples.

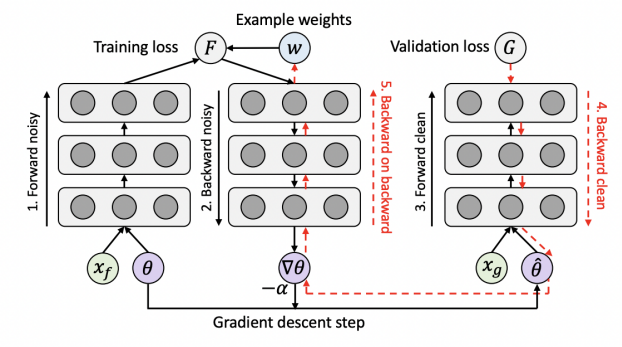

Figure 2. Computation Graph

2. Train a deep learning model on the training data with the initial weights.

3. Step 3: Evaluate the performance of the model on a validation set. If the model's performance is good enough, stop training. Otherwise, go to the next step.

4. Calculate a weight update for each example in the training data. The weight update is based on how much the example contributes to the model's error on the validation set. Examples that contribute more to the error will have larger weight updates.

5. Update the weights for each example in the training data using the weight updates calculated in step 4.

6. Repeat steps 2-5 until the model's performance on the validation set is good enough or a maximum number of iterations is reached.

7. Use the final weights for each example in the training data to train a new deep learning model.

4. Experiment

There are two contradicting ideas in training loss based approaches. In noisy label problems, authors prefer examples with smaller training losses as they are more likely to be clean images. In class imbalance problems, they prefer examples with higher training loss since they are more likely to be the minority class. In cases when the training set is both imbalanced and noisy, these existing methods would have the wrong model assumptions. In this context, it is common to construct a dataset with two parts - one relatively small but very accurately labeled, and another massive but coarsely labeled. The proposed example weighting method minimizes the loss of a set of unbiased clean validation examples.

As previously mentioned, three experiments conducted in the paper have been re-implemented. A variant of data

Algorithm 1 Learning to Reweight Examples using Automatic Differentiation

Require: $\theta_0, \mathcal{D}_f, \mathcal{D}_g, n, m$
Ensure: θ_T
for $t = 0 \dots T-1$ **do**
  $\{X_f, y_f\} \leftarrow \text{SampleMiniBatch}(\mathcal{D}_f, n)$
  $\{X_g, y_g\} \leftarrow \text{SampleMiniBatch}(\mathcal{D}_g, m)$
  $\hat{y}_f \leftarrow \text{Forward}(X_f, y_f, \theta_t)$
  $\epsilon \leftarrow 0; l_f \leftarrow \sum_{i=1}^{n} \epsilon_i C(y_{f,i}, \hat{y}_{f,i})$
  $\nabla\theta_t \leftarrow \text{BackwardAD}(l_f, \theta_t)$
  $\hat{\theta}_t \leftarrow \theta_t - \alpha\nabla\theta_t$
  $\hat{y}_g \leftarrow \text{Forward}(X_g, y_g, \hat{\theta}_t)$
  $l_g \leftarrow \frac{1}{m}\sum_{i=1}^{m} C(y_{g,i}, \hat{y}_{g,i})$
  $\nabla\epsilon \leftarrow \text{BackwardAD}(l_g, \epsilon)$
  $\tilde{w} \leftarrow \max(-\nabla\epsilon, 0); w \leftarrow \frac{\tilde{w}}{\sum_j \tilde{w} + \delta(\sum_j \tilde{w})}$
  $\hat{l}_f \leftarrow \sum_{i=1}^{n} w_i C(y_i, \hat{y}_{f,i})$
  $\nabla\theta_t \leftarrow \text{BackwardAD}(\hat{l}_f, \theta_t)$
  $\theta_{t+1} \leftarrow \text{OptimizerStep}(\theta_t, \nabla\theta_t)$
end for

Figure 3. Algorithm

imbalance has already been implemented by an independent researcher as mentioned in section **??**. Hence a subset of the second set of experiments (i.e. noisy label experiments) conducted by the original authors are taken for re-implementation in this project. Two settings of label noise were studied in the original paper viz. UNIFORMFLIP and BACKGROUNDFLIP on CIFAR benchmarks. I constructed three experiments based on the above combination, details are elaborated in the subsequent sections of the paper. All the experiments were conducted using google colab. The code can be found at https://github.com/mkrishne/MA50024_Final_Project

"To facilitate the training process, the experiments were conducted in batches of iterations (not to be confused with the batch size of the image dataloader in PyTorch) and the results were saved in CSV files. This approach served as an effective workaround to the problem of losing progress and efforts due to session termination on colab (note that colab is for interactive use)"

The three experiments conducted are :

1. CIFAR BACKGROUNDFLIP
2. MNIST BACKGROUNDFLIP
3. CIFAR UNIFORMFLIP

Each experiment consists of two test runs - one on the baseline model and the other on the proposed reweighted model. Results of these experiments are discussed in the relevant subsections.

4.1. CIFAR BACKGROUNDFLIP Experiment

Uniform flip noise label setting implies the labels for each class in the dataset would flip to a single background class. For example, the original CIFAR-10 dataset contains the following classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. So a BACKGROUNDFLIP 40% implies 40% of all the images would flip to a single background class, say class "cat". Thus out of the 50000 images, 20000 images would be class "cat" itself. Hence the authors rightly say BACKGROUNDFLIP is a blend of label imbalance and label noise since the background class dominates the label distribution. Thus BACKGROUNDFLIP experiment would also serve as a candidate for understanding the first set of experiments (i.e. data imbalance experiments). It is for this reason, two BACKGROUNDFLIP experiments were run with CIFAR and MNIST.

The experiment details are provided in Figure 4

	ORIGINAL	MINE
CNN Model	ResNet-32	WideResNet-28-10
Dataset tyoe	CIFAR-10 & CIFAR-100	CIFAR-10
Training dataset size	50000	50000
Validation set size	100	100
mini-batch size	100	10
Noise ratio	40%	40%
Background Class	3	3
learning rate	0.1 (reduce to 0.001 eventually)	0.1 (reduce to 0.001 eventually)
Iteration count	80K on both baseline and reweighted	80K on baseline; 45K on reweighted

Figure 4. CIFAR BACKGROUNDFLIP Experiment

4.1.1. Results

The results of CIFAR UNIFORMFLIP Experiment are shown in Figure 5

It is clearly seen that the reweighted model clearly outperforms the baseline model. The accuracy of the baseline is only 30% at 40K iterations compared to 50% for the reweighted model. Note that there is no reference results for comparison in the original paper as the authors have used a Resnet-32 instead of a WideResnet-28-10. Hence only a baseline vs reweighted performance comparison is feasible.

It is also seen that the baseline accuracy drastically fluctuates indicating that WideResnet is a poor baseline for the given noisy dataset. Reweighting model on that other hand has accuracy in an increasing trend. Hence we can conclude that the Reweighting algo helps to stabilise the training accuracy.

The reweighting algorithm can be best appreciated in this experiment as we can see that the accuracy of the reweighted model is three times that of the baseline

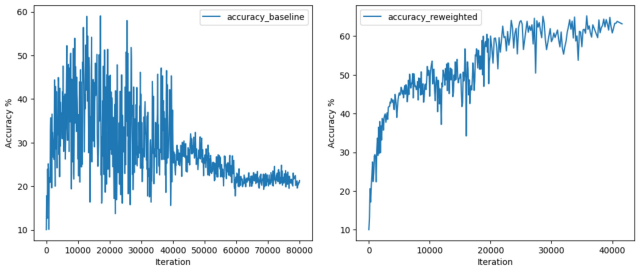

Figure 5. CIFAR UNIFORMFLIP Experiment Results

4.2. MNIST BACKGROUNDFLIP Experiment

Due to BACKGROUNDFLIP being a better representative of complex input patterns, another experiment was run with MNIST dataset on a standard lenet model. This experiment is new and not part of the original paper. This was included as it would further strengthen the validity and reliability of the implemented code. Moreover, due to the faster training time, MNIST and Lenet were used for a preliminary validation of the implementation, prior to conducting the more computationally intensive CIFAR BACKGROUNDFLIP Experiment

The experiment details are provided in Figure 6

	MINE
CNN Model	Standard LeNet
Dataset tyoe	MNIST
Training dataset size	50000
Validation set size	100
mini-batch size	25 for baseline
Noise ratio	40%
learning rate	0.1 (reduce to 0.001 eventually)
Iteration count	50K on both baseline and reweighted

Figure 6. MNIST BACKGROUNDFLIP Experiment

4.2.1. Results

The results are slightly comparable to that of the CIFAR BACKGROUNDFLIP experiments, although different observations can be drawn. Here as well, the baseline model accuracy fluctuates drastically and only marginally stabilises after 40K iteration when the learning rate reduces.

The reweighted model not only outperforms the baseline, but also manages to achieve a stable loss of around 65% at 50K iteration count.

The results of MNIST BACKGROUNDFLIP Experiment are shown in Figure 7

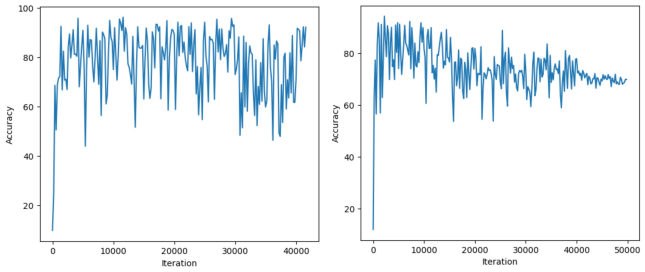

Figure 7. MNIST BACKGROUNDFLIP Experiment Results

4.3. CIFAR UNIFORMFLIP Experiment

Uniform flip noise label setting implies the labels for each class in the dataset would be flipped to one of the other nine classes with equal probability. In CIFAR-10 case for example, an image of a bird with the original label ”bird” could be flipped to any of the other nine labels with probability 1/9 each. 40% noise label setting implies 40% of the images uniformly flip to other 9 classes.

The experiment details are provided in Figure 8

	ORIGINAL	MINE
CNN Model	WideResNet-28-10	WideResNet-28-10
Dataset tyoe	CIFAR-10 & CIFAR-100	CIFAR-10
Training dataset size	50000	50000
Validation set size	1000	200
mini-batch size	100	10
Noise ratio	40%	40%
learning rate	0.1 (reduce to 0.001 eventually)	0.1 (reduce to 0.001 eventually)
Iteration count	60K on both baseline and reweighted	60K on baseline; 11K on reweighted

Figure 8. CIFAR UNIFORMFLIP Experiment

4.3.1. Results

The results of CIFAR UNIFORMFLIP Experiment are shown in Figure 9 For the baseline, it was found that that the accuracy reaches 68%. We can see a rise in the accuracy at iteration 40K due to decrease in the learning rate as per the algorithm. This is consistent with the results of the original paper as they had reported an accuracy of 67.97 ± 0.62 for CIFAR UNIFORMFLIP under 40% noise ratio using a WideResNet-28-10 model.

The reweighted model was executed for a limited number of 11,000 iterations due to time constraints. Notably, this number of iterations is the culmination of two days of intense efforts. It is observed that the reweighted model outperforms the baseline models after a certain number of iterations as the weights are updated in a progressive fashion.

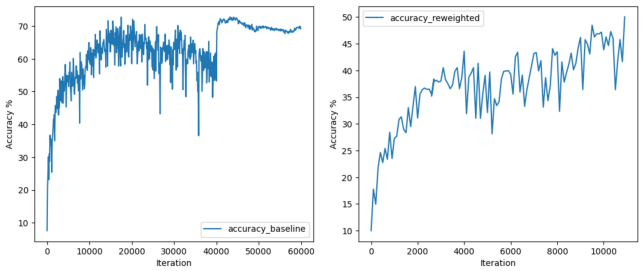

Figure 9. CIFAR UNIFORMFLIP Experiment Results

5. Conclusion

The experiments were conducted using different deep learning architectures, including ResNet and Lenet, and compared the performance of reweighted models against baseline models without reweighting.

Despite the computational cost of the algorithm, it was successfully replicated with a high degree of accuracy, closely reproducing the results reported in the original paper. The technique of example reweighting is demonstrated to be effective in improving the performance of various models on a range of datasets. Overall, the algorithm aims to improve the robustness of deep learning models by assigning more importance to difficult examples, which can help to mitigate the effects of imbalanced or noisy datasets.

Further results from these experiments can be used to demonstrate the potential of reweighting as a tool for addressing a range of related problems in deep learning, such as adversarial attacks and domain adaptation. Overall, this paper provides important insights into the challenges of training deep learning models under realistic conditions, and offers a promising avenue for improving the robustness of these models in the future.

6. Acknowledgement

I would like to thank Ruqi Bai, TA for ML50024 and ChatGPT, a language model developed by OpenAI, for its assistance in providing explanations and clarifications during the project

References

Angluin, D. and Laird, P. Learning from noisy examples. *Machine Learning*, 2:343–370, 1988.

Lenc, K. and Vedaldi, A. Understanding image representations by measuring their equivariance and equivalence, 2015.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. Learning with noisy labels. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.