An Empirical Study of Robust Deep Models

Haoye Cai
Stanford University
hcaiaa@stanford.edu

Kaidi Cao
Stanford University
kaidicao@cs.stanford.edu

Bingbin Liu
Stanford University
bingbin@stanford.edu

1 Introduction

Advances in deep neural networks have led to a resurgence of their use in many applications. Though deep learning has greatly improved accuracy for many tasks, its robustness as well as training efficiency issues are preventing it from deployment in many real-work scenarios. To be more specific, Deep Neural Networks can fit complex functions by learning from data, but the capability to fit the entire training set is sometimes undesirable, as the mislabeled samples in the training set might severely hurt model generalization. Mislabeled samples are hard to avoid in real world large-scale datasets. Therefore, algorithms that are robust to mislabeled samples are warranted to improve the generalization of deep neural networks when it comes to very large labeled datasets, e.g., Webvision [1] dataset. Another important research direction is trying to look for solutions for faster training. A commonly encountered problem for training deep learning model is that the process takes long (GPU) hours, and acceleration is normally done by large-scale training with distributed systems.

Therefore, in this project, we try to address both issues, and develop new training algorithms which could potentially result in fast convergence and a learned model more robust to noise and mislabelled data points. We propose a novel training strategy that first cluster the data points according to certain distance metrics, and sample points from each cluster according to our selection criterion, and then weight each sampled point based on information within the cluster. We propose two different algorithm to achieve this scheme. We show experimental results on two popular datasets and demonstrate the effectiveness of our method in both accelerating training in initial stages and increasing the model's robustness to noisy datapoints.

2 Related Work

There are three papers that we found are pretty close to our topic. [2] studies the "forgetting events" for deep learning, and further propose that a significant fraction of examples can be omitted from the training data set while still maintaining the accuracy. We are looking into the same direction to see if a better data sampler could help with generalization or training speed. [3] uses a clean meta validation set to estimate the quality of each training example. And [4] proposes to make dataset smaller by taking advantage of the internal structure of dataset. Their work has theorital proof when optimization is convex. In this project we are following a similar formulation but also trying the extend the problem to nonconvex optimization.

3 Problem Description

Our project focus on solving two problems: faster and more robust training.

For **faster training**, the goal is to use fewer number of gradient calculations during training. Formally, suppose we train a model for N epochs, and each epoch involves m gradient calculations as it walks through the whole training set. Our goal is to train the model with m' gradient calculations in each epoch where $m' < m$. This is equivalent to using a smaller set of samples per training epoch, which

Preprint. Project for CS 341.

motivates us to subsample the training set. We perform subsampling by selecting from specially designed clusters, and will describe the two variations in more detail in Section 5.

For **robust training**, we would like the model to learn the correct classification boundaries despite the presence of noise in the training set. More specifically, the model is trained with a noisy training set obtained by randomly altering the class labels of $x\%$ of the training samples. Our goal is to train the model on the altered training set while still maintain high accuracy on the test set. We experimented with different values of $x\%$, and present our findings in Section 6.

We use the clean and complete test set in both scenarios.

4 Data

We conduct our major experiments on two dataset, which we describe below.

CIFAR-10 dataset [5]: CIFAR-10 contains 32 colored images from 10 classes. The dataset is split into a training set with 50k images and a testing set with 10k images. We follow standard data split and pre-processing steps as in previous papers.

FashionMNIST [6]: to validate our method across different datasets, we conduct the same set of experiment on FashionMNIST, which is more challenging than MNIST [7] which we used before the milestone. FashionMNIST contains 28 grayscale images from 10 classes of different clothes. The dataset is split into a training set with 60k images and a testing set with 10k images. We follow standard data split and pre-processing steps as in previous papers.

5 Methods

We modify the training procedure in two steps: first as a preprocessing step (which might be online or offline), we perform clustering to identify sets of neighbors for each data point, and then during training, we perform weighted selection to help achieve different training goals.

5.1 Clustering

We propose two different algorithms for our clustering step, namely set cover and facility location, which we describe below:

Set cover problem: the set cover problem aims to find a minimum number of subsets that can cover all elements in set. Particular to our setting, given a dataset of size N, we generate N subsets, each centered at one data point and containing all the neighbors within a ball with a given radius r. The set cover problem has a well-known greedy algorithm that provides $(1 - \frac{1}{e})$-optimal solution. Our implementation (provided by Baharan) further utilize the submodularity of the problem to operate in a lazy fashion. Clustering is performed on the training set only, and is performed offline as one-time preprocessing.

Facility location: in the facility location, given a set, we would like to find a fixed size of subset, whose elements are called facilities, such that the total distance of elements in the set to the closest facility is minimized. Formally, the goal is to find a subset $F*$ of size k,

$$F* = \text{argmin}_{F:|F|=k} \sum_{i \in S \setminus F} \min_{p \in F} D(i, p)$$

Modeling clustering as the facility location problem has the following advantages. First, since the facilities are calcualted using the distances to all points in the set, it can better capture the global geometry of the vector space. Second, the facility location problem allows us to explicitly specify the number of clusters, rather than controlling indirectly by adjusting the radius r. Last and perhaps the most important, facility location is much faster to compute than set cover, which allows us to update the facilities in an online fashion after each epoch.

5.2 Choice of vector space

In our experiments, we calculate distances in two different ways, one on the feature space and one on the gradient space.

Table 1: Number of clusters

r	0	1.2	1.3	1.5	1.6	1.7	1.8	2.0
# clusters	50000	49500	49043	46078	43105	38964	34099	23346
acc@1	92.65	92.84	92.41	**92.70**	92.26	91.55	91.36	87.57
acc@5	99.81	99.84	99.80	99.80	**99.81**	99.75	99.81	99.57

Distances using Features: We use L_2 distance defined over the feature space. The choice of distance measure takes inspiration from [4] which uses the distance between gradients, with the approximation that in the convex case, feature distance can approximate gradient distance up to a constant scale.

Distances using Gradients: Since neural networks are high non-convex, directly using gradients may give better results. We therefore take gradients from the last layer, as suggested by [4].

5.3 Selection & Weighting

Denote the number of clusters as C. During training, we modify the data loader to iterate over the clusters rather than individual data points, such that there is a saving of $\frac{C}{N}$ in the number of forward/backward passes.

The data points within a cluster are given equal weight and sampled randomly. However we consider their contribution to the training batch to be different. Specifically, we use two different weighting schemes for different training aims:

- For faster training, a sample drawn from a cluster of size s_c is given a weight of s_c. This weighting can be considered as a correction, with which the data loader is effectively sampling from a distribution consistent with the entire dataset. The weights are normalized over the training batch to prevent gradients from exploding, i.e. $s'_c = \frac{s_c}{\sum_{b=[1..B]} s_b}$, where B is the batch size.
- For robust training where noisy labels present, each sample is weighted by its label confidence, which is the fraction of data points in the same cluster whose labels agree with the sampled point.

6 Experiments

Our project focuses on both faster training and more robust training, and test our algorithm on the CIFAR-10 and FashionMNIST dataset. All experiments are performed with ResNet18 [8].

6.1 Faster Learning

The experiments on faster learning can be grouped by the algorithm used, namely, set cover and facility location.

6.1.1 Set cover

We first present our results using the set cover algorithm on the similarity (distance) matrix for the entire training set. We examine the change in performance as the radius r varies.

Using feature similarity: To obtain the features, we first pretrain ResNet18 on the dataset, and extract features for each data point in the training set. Table 1 shows the change in the number of clusters as well as the top-1 and top-5 accuracy with respect to r. We noticed that the model is able to achieve competitive performance using 85 - 90% of the data in each epoch. However, the performance starts to degrade significantly as the number of clusters decreases further. These trends are also observed in the training and testing plots as shown in figure 1.

Using gradient similarity: we conduct experiments to answer two questions:

Which layer to use? We use two interpretations of "last layer": the last fully-connected layer, or the final softmax loss layer. We found that the softmax layer works better; See Table 2 for more detailed results.

Which checkpoint to use? Since the gradients depend on the model weights, choice of model checkpoints may affect the performance. Therefore, we extracted gradients with 1) the checkpoint

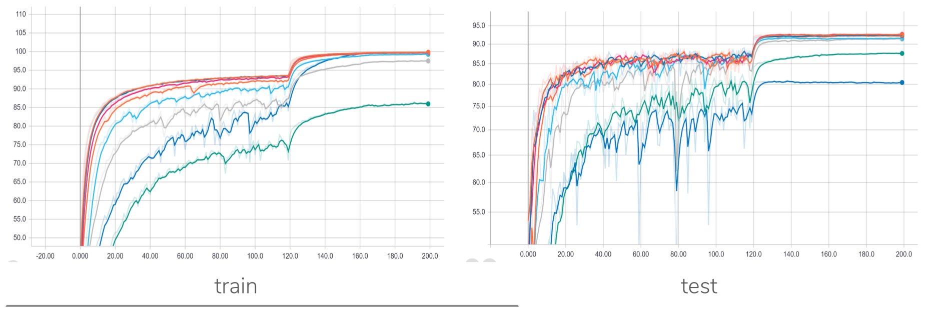

Figure 1: Curves on training and testing performance with feature similarity.

Table 2: Comparison of types of distances

Model	# clusters	Acc@1	Acc@5
No clustering	50000	92.90	99.81
Feature	43105	92.26	99.81
Grad@FC	42053	90.70	99.74

with best test set performance; 2) checkpoints at every 10_{th} epoch, starting from epoch 100, which contain only relatively stable gradients; 3) checkpoints at every 10_{th} epoch, starting from the beginning, which contain the entire training history.

We experimented with these three settings on gradients from both the last fully-connected layer and the softmax layer. Results from the checkpoint with the best test set performance were poor and therefore were omitted. Table 3 shows comparison for the other two settings, where we also include results from feature similarity experiments with similar number of clusters. Our experiments show that gradients from the softmax layer using checkpoints starting from 100 epochs gives the best test set accuracy.

6.1.2 Facility Location

In facility location, we similarly experiment with different ratio of training samples per epoch. As mentioned before, facility location has a major advantage on computation speed, which allows us to compute the gradients in an online fashion. Specifically, we train the model on the full dataset for 5 epochs to stabilize the model weights, after which we extract the gradients and apply the facility location algorithm on normalized gradients at the end of each epoch.

Since our results from the set cover part show that gradient similarity from the softmax layer works the best, the experiments on facility locations are all using gradient similarity. Moreover, note that for the logit o_i for each sample, the gradient of the softmax layer with respect to o_i is $p_i - y_i$ where y_i is the class label and p_i is the value after normalization. This means that the gradient of the softmax

Table 3: Comparison of choice of checkpoints, experimented with both fully-connected (FC) layer and softmax (SM) layer. For softmax gradients with stable (i.e. after 100 epochs) checkpoints, we also compare it with the feature similarity using another set of experiments with around 39K clusters. The results are shown in parentheses, where using softmax gradients significantly outperforms using features similarity.

Model	# clusters	Acc@1	Acc@5
Feature	46078	92.70	99.80
FC@stable	47794	**92.47**	99.83
FC@all	48960	92.46	99.80
Feature	43105 (38964)	92.26 (91.55)	99.81
SM@stable	43334 (39715)	**92.48** (92.15)	99.80
SM@all	42939	92.43	99.77

Table 4: Experiments on facility location: as the facility ratio varies, the number of data points that the training process effectively covers also varies. Training on 80% of the samples can get competitive testing accuracy compared to training on the full dataset.

Facility ratio	20%	40%	60%	80%	100%
Data covered	40%	60%	80%	92%	100%
Acc@1	80.79	85.17	89.16	91.76	93.08

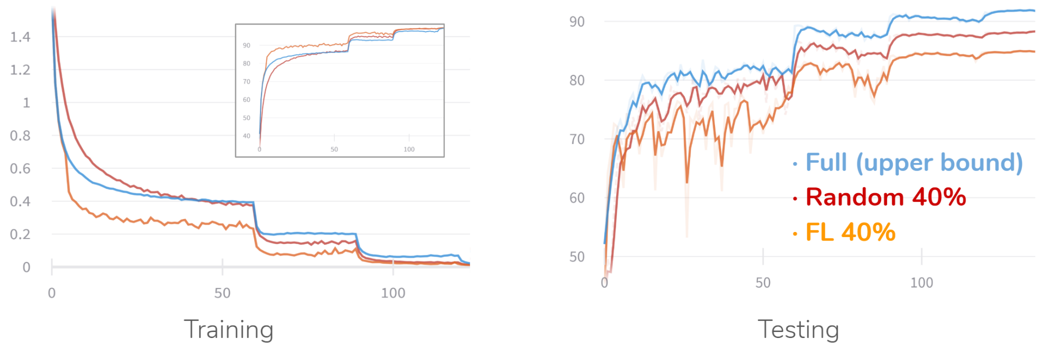

Figure 2: Loss curve (left) and testing accuracy (right) as training proceeds.

layer can be obtained without addition backpropogation, which is desirable from the performance perspective.

Table 4 shows different facility ratios and their corresponding performances on the test set. The second row "Data covered" keeps track of the number of data points the model sees during the stable training stage, which helps to understand how the facilities are updated at each epoch.

As a baseline, we run experiments where the model is trained with a fixed ratio of data points sampled at the beginning of training. With the facility ratio set as 40%, the comparison between facility location (orange) and the random baseline (red) is shown in figure 2, where we also include the results for training on the full dataset (blue). As can be seen from the training loss on the left, with the same number of gradient calculation and model parameter updates in each epoch, facility location has better training behavior than the random baseline, with a more rapidly optimized loss curve and training accuracy. This means our facility location algorithm is indeed effective in selecting the most representative examples from the dataset. Unfortunately, facility location seems to overfit to the training data, since in the testing accuracy curve on the right, facility location has better performance at the very beginning, but is soon surpassed by the random baseline. The same trend is also observed with a different scheduling scheme where the learning rate is kept constant. Nevertheless, facility location was proven useful with robust training, which we will introduce in the next subsection.

Regarding the hyperparameters, we experimented with different batch size, learning rate scheduling, choice of optimizer and order of presenting training samples. The best setting we found is to train on batchs of size 32, using SGD with a starting learning rate of 0.1 and decreasing by 0.3 at epoch 60, 90, 120, 160 and stops at epoch 200. Moreover, maintaining the ordering given by the facility location algorithm worked better than random shuffling.

6.2 Robust Learning

In addition to the testing setting for faster learning, we also test the robustness of the proposed algorithm. To do so, we follow the setting in Jiang et al. [9], generating a corrupted CIFAR-10 dataset by randomly flipping a portion of the labels in the training set to another uncorrelated label.

We follow the simple data augmentation in [8] for training: 4 pixels are padded on each side, and a 32×32 crop is randomly sampled from the padded image or its horizontal flip. We use ResNet-32 [8] as our base network, and use stochastic gradient descend with momentum of 0.9, weight decay of 2×10^{-4} for training. The model is trained with a batch size of 128 for 200 epochs. For fair

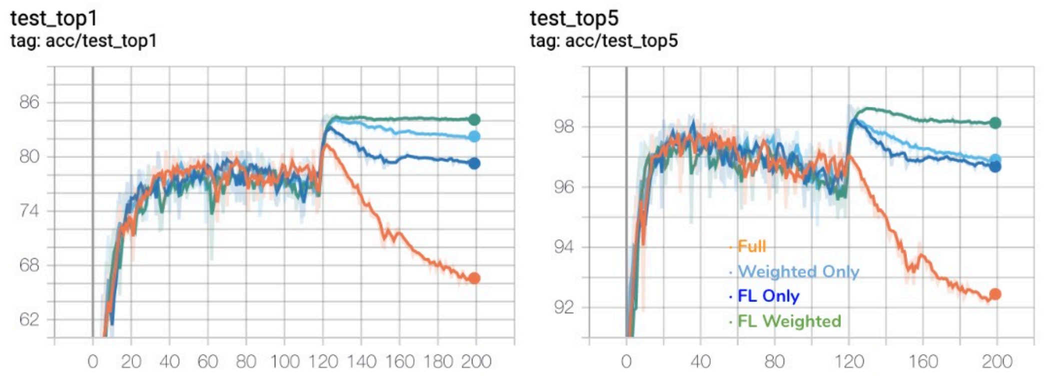

Figure 3: Training curve (test top1/top5 accuracy) of ResNet-32 trained with different schemes on noisy CIFAR-10. The x-axis is number of epochs and the y-axis is the test accuracy. The noisy ratio is 0.4.

comparison, we use an initial learning rate of 0.1, then decay by 0.01 at the 120th epoch and again at the 160th epoch.

Here we test on four settings and we enumerate them as follows,

1. Full: we train on the full noisy dataset with the pure Empirical Risk Minimization (ERM) setting. Each data point is weighted equally.
2. FL only: We calculate Facility Location of the noisy dataset at learning rate decay at 120th epoch, and use subsampled points for training after that. Each FL data point is treated equally.
3. Weighted only: We train on the full noisy dataset, but weight each data point by the number of data points in its unit ball with radiust r after the learning rate decay at 120th epoch.
4. FL Weighted: We combine the methods described in FL only and Weighted only.

Here we present the results of the proposed algorithms when trained on noisy CIFAR-10: Figure 3 shows the training curves on test accuracy with noise ratio 0.4 and Table 5 shows the summarized top1 accuracy of different methods under noise ratio 0.2 and 0.4. We can see from Figure 3 that after decaying the learning rate at 120th epoch, the baseline method "Full" degrades fast while our proposed two methods degrade much slower and the combined "FL Weighted" achieves the highest accuracy. In Table 5 we also confirm that our proposed methods yield higher accuracy under noise ratio of either 0.2 or 0.4.Hence we conclude that both the facility location and the re-weighting scheme could help to increase the generalization of the training process when there exists corrupted labels. Notably, our final result is on par with state-of-the-art results [10].

Table 5: Top-1 validation error of ResNet-32 trained with different schemes on noisy CIFAR-10. The best scores are highlighted.

	Top1 Acc	
Noise Ratio	0.2	0.4
Full	87.22	83.20
FL only	88.39	83.64
Weighted only	88.69	84.18
FL Weighted	**89.72**	**85.16**

Figure 4 and Figure 5 show ablation study results on radius r and FL ratio, which are the only two hyper-parameters in the proposed algorithm that is subject to be tuned. We see that $r = 2.0$ and FL raito $= 0.4$ work the best.

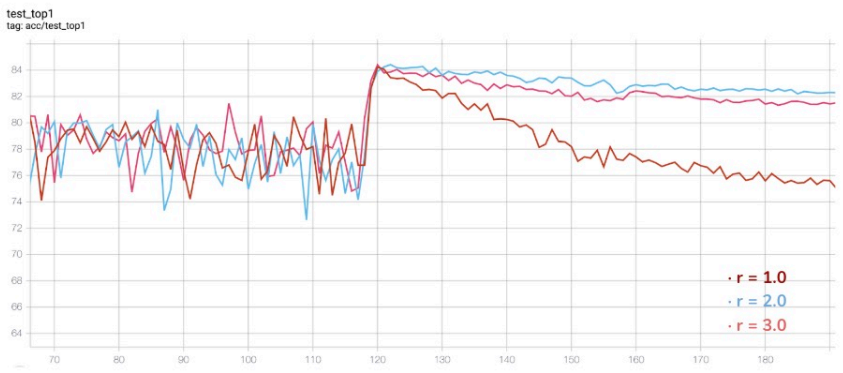

Figure 4: Ablation study on the influence of radius r. We observe that $r = 2.0$ works best.

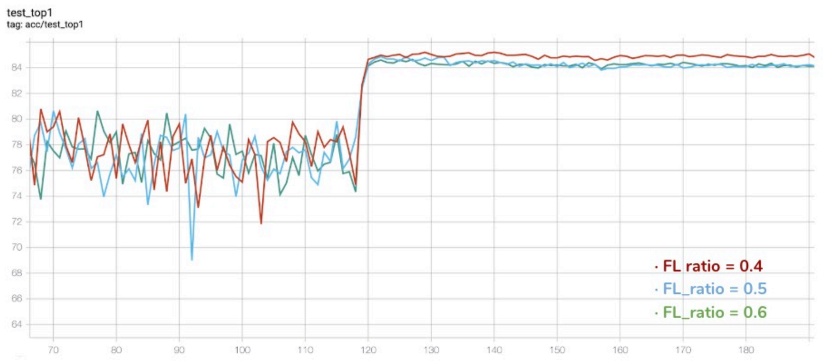

Figure 5: Ablation study on the influence of FL ratio. We observe that 0.4 works best.

Table 6: Number of clusters and accuracy with respect to radius size.

r	0	0.2	0.4	0.5	0.6	1.0	1.5
# clusters	60000	59607	50020	37267	24203	4276	797
acc@1	**93.12**	**93.13**	92.97	**93.06**	92.60	92.08	91.85

6.3 Experiments on FashionMNIST

To validate our method across different datasets, we also conduct the same set of experiments on FashionMNIST dataset. We follow the same scheme for data processing, the same method set up for both faster and robust training and the same hyperparameters in each case.

6.3.1 Faster Learning

For experiments on faster learning, we replicate experiments for both Set Cover and Facility Location.

For Set Cover, Table 6 shows the change in the number of clusters with respect to different radius sizes as well as the top-1 accuracy. We noticed that the model is able to achieve competitive performance, while the performance starts to degrade as the number of clusters decreases significantly. These results are consistent with our findings in the CIFAR-10 experiments.

For Facility Location, Table 7 shows different facility ratios and the portion of training data covered as well as their corresponding top1 accuracy on the test set. From the table we can see that training on fewer samples with facility location has small performance compromise, while the training process becomes much faster (fewer data points, so fewer gradient calculations), similar to CIFAR-10 results.

Table 7: Experiments on facility location: the number of data points that the training process effectively covers is slightly higher than the facility ratios, and is fewer compared with CIFAR-10 experiments. Training on fewer the samples can get competitive testing accuracy compared to training on the full dataset.

Facility ratio	20%	40%	60%	80%	100%
Data covered	35%	50%	75%	90%	100%
Acc@1	90.68	90.71	91.17	91.45	93.12

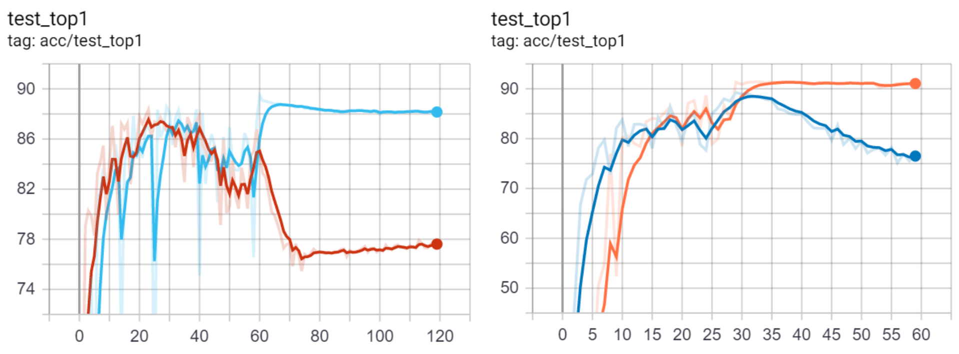

Figure 6: Training curve (test top1 accuracy) of ResNet-32 trained with different schemes on noisy FashionMNIST with noise ratio 0.2 (left) and 0.4 (right). The x-axis is number of epochs and the y-axis is the test accuracy.

6.3.2 Robust Learning

For experiments on robust learning, we replicate the experiment for our best setting on CIFAR-10 dataset, i.e. “FL Weighted” which uses Facility Location to select points and weight each point by the density within a unit ball, under the noise ratio of 0.2 and 0.4. We follow the same corrupted data generation process. We also run a baseline trained on full data for comparison.

Figure 6 shows the training curves on test accuracy with noise ratio 0.2 and 0.4. We can see from the figure that after learning rate decay, the test accuracy increases for our method while decreases significantly for the baseline trained on full noisy data. Table 8 shows the summarized top1 accuracy of different methods under noise ratio 0.2 and 0.4 and we can see that our proposed methods yield higher accuracy under both setting. The results shown are consistent with what we discover on the CIFAR-10 dataset. Hence we conclude that our model generalizes across different datasets.

7 Conclusion and Future Work

Our project focuses on finding methods to make the training of deep learning models faster and more robust. We propose to model the problem with the aid of the facility location algorithm, which is proven to be effective from experiments on CIFAR-10 and FashionMNIST, especially for robust training. As for future work, since facility location exhibits better training behavior at the early stage of training, we may consider using curriculum training, where we train the model with the most important subset of samples first, and increase the size of the subset as training progresses.

Table 8: Top-1 validation error of ResNet-32 trained with different schemes on noisy FashionMNIST. The best scores are highlighted.

	Top1 Acc	
Noise Ratio	0.2	0.4
Full	88.35	89.26
FL Weighted	**89.47**	**91.42**

Acknowledgement

We would like to thank the teaching team of CS341 for their support and valuable feedback. Special thanks to our mentor Baharan, who has given us much guidance and help throughout the course.

References

[1] Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. Webvision database: Visual learning and understanding from web data. *arXiv preprint arXiv:1708.02862*, 2017.

[2] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJlxm30cKm.

[3] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4334–4343, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/ren18a.html.

[4] Zeyuan Allen Zhu, Yang Yuan, and Karthik Sridharan. Exploiting the structure: Stochastic gradient methods using raw clusters. *CoRR*, abs/1602.02151, 2016. URL http://arxiv.org/abs/1602.02151.

[5] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[6] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[7] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. *arXiv preprint arXiv:1712.05055*, 2017.

[10] Pengfei Chen, Benben Liao, Guangyong Chen, and Shengyu Zhang. Understanding and utilizing deep neural networks trained with noisy labels. *arXiv preprint arXiv:1905.05040*, 2019.