# Robust Learning for Noisy Datasets for Overparameterized Neural Networks

**Prakhar Sharma**
705728582
prakhar6sharma@gmail.com

**Niranjan Vaddi**
205525172
niranjanv@ucla.edu

**Sunchu Rohit**
505525335
sunchurohit@ucla.edu

**Yash Trivedi**
905730032
yashtrvd@ucla.edu

## Abstract

Modern day neural network architectures have huge number of parameters compared to the training set size. Training these networks to achieve best accuracy leads to memorization of data and overfitting which can have severe issues on performance when trained especially on a noisy dataset. Inspired from [1, 2, 3, 4], we propose a new training paradigm where at each training step we categorize parameters into critical and non-critical parameters and add an auxiliary variable for each training example to the network output as a regularization term. Based on our categorization of parameters, we apply different update rules to favor learning from the cleaner labels. Without extensive hyperparameter tuning, our results show that proposed approach outperforms the original works.

## 1 Introduction & Related Works

Even though deep neural networks have been highly successful across various tasks, their performance is highly dependent upon the amount and quality of the data[5]. To build such large-scale datasets, manual annotation becomes infeasible and the only way to do so is via crowd-sourcing. Unfortunately, maintaining the quality of annotations of crowd-sourced data becomes a challenge in itself. For small to medium scale data collection process, it is possible to do quality checks on the crowd-sourced workers by randomly sampling few annotations of workers but again it becomes infeasible to do this on a large scale. Hence, the noise in the final data is inevitable. This problem is exacerbated even further by recent neural network architectures where the number of parameters are much higher than the number of data points. [6] showed that such over-parameterized networks can fit any labels, even completely random ones. This poses a serious threat on the generalization performance of such networks.

Fortunately, there is a substantial amount of work done to deal with noisy labels [7, 8, 9]. Recent papers [10, 3] have shown that using first order gradient based methods such as SGD results in networks first learning the cleaner portion of the data before overfitting on the noisier counterpart. This is the reason why early-stopping works so well in practice. Despite using early-stopping, [1] showed that noisy labels can still affect the performance of the network.

The lottery ticket hypothesis [11] on the other hand shows that deep networks are likely to be over-parameterized, and only partial parameters are important for generalization. With this part of the parameters, the small and sparsified networks can be trained to generalize well. The lottery ticket hypothesis focuses on network compression and aims to find a sparsified sub-network which has competitive generalization performance compared with the original network. Motivated by the lottery ticket hypothesis, [1] proposed a method ($CDR$) on finding critical/non-critical parameters to reduce the side effect of noisy labels.

[4] analyzed the problem of learning on noisy labels using deep networks from a theoretical perspective by considering them to be in a neural tangent kernel[12] regime. They proved that network must move a lot from it's initialization parameters to overfit the data and correspondingly proposed a regularization term to prevent the network from doing so. They also showed that under mild assumptions[13], replacing the regularization term based on the initialization distance to an auxiliary variable for each training example converges to the same network. This auxiliary variable can especially be tuned to control the impact of noise in the early stages of training.

While, [1] improves the generalization of the model by training only the critical parameters, it still takes gradient of noisy labels into account, on the other hand [4] partially solves this problem but it still suffers from over-parameterization. In this report, we propose $CDR + AUX$ which even without extensive hyper-parameter tuning, outperforms both methods on most of our experiments. Our implementation comparing all methods can be found here github repo.

## 2 Methodology

In this section we first discuss about the problem setup, followed by the preliminaries regarding the neural network optimization, CDR method proposed in [1], regularization techniques to reduce the noise in gradient [2] and finally our proposed approach.

### 2.1 Notation and Problem Setup

In this work, we consider the task of multi-class classification with $c$ classes. We denote, $\mathcal{X} \in \mathbb{R}^d$ as the feature space, with $d$ being the dimensionality and $\mathcal{Y} = [c]$ as the label space. Thus, our data, denoted by $D$ can be represented as a joint distribution over $\mathcal{X} \times \mathcal{Y}$. The sample space $S = \{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^n$ consists of $n$ i.i.d samples drawn from $D$. While learning with noisy labels, our sample contains examples from a corrupted distribution $\tilde{D}$ instead of $D$. We denote this corrupted sample by $\tilde{S} = \{(\mathbf{x}_i, \tilde{\mathrm{y}}_i)\}_{i=1}^n$. Here, $\tilde{\mathrm{y}}_i$ represents the corrupted version of its underlying clean label $\mathrm{y}_i$. Therefore, the aim of our work is to develop a robust classifier that can assign clean labels to test data by learning on a training sample with noisy labels.

### 2.2 Neural Network Optimization - SGD

Optimization is essential for training neural networks. Among various optimization methods available, Stochastic Gradient Descent (SGD) is the most popular one nowadays. Suppose we have a classifier with $m$ parameters to be trained. We shall denote $\mathcal{W} \in \mathbb{R}^m$ as the set of all parameters. Let $L : \mathbb{R}^c \times \mathcal{Y} \to \mathbb{R}_+$ be a surrogate loss function, e.g., the cross entropy loss. Along with an additional regularization term, e.g., the $l_2$ regularizer, the objective function to be minimized will look like:

$$\min L(\mathcal{W}; S) = \min \frac{1}{n} \sum_{i=1}^{n} L(\mathcal{W}; (\mathbf{x}_i, \mathrm{y}_i)) + \lambda ||\mathcal{W}||_2^2$$

where $\lambda \in \mathbb{R}_+$ is the regularization parameter. At every iteration, $\mathcal{W}$ will be updated as per:

$$\mathcal{W}(t+1) \leftarrow \mathcal{W}(t) - \eta \left( \frac{\partial L(\mathcal{W}(t); \tilde{S}^*)}{\partial \mathcal{W}(t)} + \lambda(\mathcal{W}(t)) \right)$$

where $\eta > 0$ is the learning rate, $\mathcal{W}(t)$ is the set of parameters at $t$-th iteration, and $\tilde{S}^*$ is a subset randomly sampled from $\tilde{S}$. With SGD, $\lambda$ is equivalent to the weight decay coefficient in the training process [14].

### 2.3 CDR

According to lottery ticket hypothesis, deep neural networks are overparameterized and only partial parameters are sufficient for generalization. Inspired by this hypothesis, [1] proposed to identify the parameters which are critical to memorize clean labels. Since, neural networks learn the easier/clean examples first, hence during the initial phase of training, the parameters receiving higher gradients can be thought of as critical parameters. The problem with this approach is that if we only use the gradient information, we ignore the value of the parameter $\mathrm{w}_i$. However, if the value of the parameter is close to zero, it is inactivated, hence it is also non-critical for optimality. Therefore a combination of the value of the parameters and gradient w.r.t the parameters is a better criterion for criticality. For a parameter $\mathrm{w}_i \in \mathcal{W}$, its gradient is $\nabla L(\mathrm{w}_i; S)$ and the proposed judgement criteria is $g_i = |\nabla L(\mathrm{w}_i; S) \times \mathrm{w}_i|$. If the value of $g_i$ is large, $\mathrm{w}_i$ is viewed as a critical parameter. Else, $\mathrm{w}_i$ is classified as a non-critical parameter i.e. it is not important for fitting clean labels.

If the noise rate of the dataset is high, the number of clean labels is small, so the number of critical parameters required for memorizing clean labels is small. Therefore the number of the critical parameters has a negative correlation with the noise rate. Let $\tau$ denote the noise rate. Then the number of critical parameters $m_c$ can be defined as $m_c = (1-\tau)m$. The critical and non-critical parameters are determined according by numerical sorting of $g_i$. The critical and non-critical parameters are denoted by $\mathcal{W}_c$ and $\mathcal{W}_n$ respectively. Two different update strategies are performed for two types of the parameters.

**Robust positive update**: For critical parameters $\mathcal{W}_c$, we use the gradients derived from the objective function and weight decay. The update rule is:

$$\mathcal{W}_c(t+1) \leftarrow \mathcal{W}_c(t) - \eta \left( (1-\tau) \frac{\partial L(\mathcal{W}_c(t); \tilde{S}^*)}{\partial \mathcal{W}_c(t)} + \lambda \mathcal{W}_c(t) \right)$$

where $\tilde{S}^*$ is a subset randomly sampled from $\tilde{S}$. Though we have only noisy training data, deep networks will first memorize training data with clean labels. As can be seen in the above equation, the gradient decay coefficient is set to $1-\tau$ , which can prevent over-confident descent steps in the training process.

**Negative update**: For non-critical parameters $\mathcal{W}_n$, only weight decay is used to perform the update. The update rule is:

$$\mathcal{W}_n(t+1) \leftarrow \mathcal{W}_n(t) - \eta\lambda\mathcal{W}_n(t)$$

Robust positive update uses the gradients to update the critical parameters, which helps deep networks memorize clean labels. Non-critical parameters tend to overfit noisy labels, so their gradients are misleading for generalization. Thus, only the weight decay is used to update them. The weight decay will penalize their values to be zero and help generalization [15]. As they are deactivated, they will not contribute to the memorization or generalization. The use of two update rules helps to reduce the side effect of noisy labels and thus enhance the memorization of clean labels.

## 2.4 Regularization methods - AUX, RDI

[2] proposed regularization methods named AUX, RDI and proved that gradient descent training with either of these 2 methods leads to generalization guarantee on clean data distribution despite being trained using noisy labels. Let $f(\mathcal{W}, \cdot)$ be the neural network to be trained. Firstly, let's consider the case of a scalar target and a single output network. Given a noisily labeled training set $\tilde{S}$, a direct unregularized training method would involve minimizing an objective function like $L(\mathcal{W}; \tilde{S}) = \frac{1}{2}\sum_{i=1}^{n}(f(\mathcal{W}, \mathbf{x}_i) - \tilde{y_i})^2$. To prevent overfitting, [2] proposed the following simple modifications in the regularization terms of the objective function:

- **Method 1: Regularization using Distance to Initialization (RDI).** Randomly generate the initial parameters $\mathcal{W}(0)$, and minimize the following regularization objective:

$$L_\lambda^{\mathrm{RDI}}(\mathcal{W}; \tilde{S}) = \frac{1}{2}\sum_{i=1}^{n}(f(\mathcal{W}, \mathbf{x}_i) - \tilde{y_i})^2 + \frac{\lambda^2}{2}||\mathcal{W} - \mathcal{W}(0)||^2$$

- **Method 2: adding an AUXiliary variable for each training example (AUX).** An auxiliary training parameter $b_i \in \mathbb{R}$ is added for each traning example and the following objective is minimized:

$$L_\lambda^{\mathrm{AUX}}(\mathcal{W}, b; \tilde{S}) = \frac{1}{2}\sum_{i=1}^{n}(f(\mathcal{W}, \mathbf{x}_i) + \lambda b_i - \tilde{y_i})^2$$

where $b = (b_1, b_2 ..., b_n)^T \in \mathbb{R}^n$ is initialized to be 0.

To extend these to multiple outputs, suppose that $\{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}^c$, and the neural net $f(\mathcal{W}, \mathbf{x})$ has $c$ outputs, then the objectives would be:

$$L_\lambda^{\mathrm{RDI}}(\mathcal{W}; \tilde{S}) = \frac{1}{2}\sum_{i=1}^{n} ||f(\mathcal{W}, \mathbf{x}_i) - \tilde{\mathrm{y}_i}||^2 + \frac{\lambda^2}{2}||\mathcal{W} - \mathcal{W}_0||^2$$

$$L_\lambda^{\mathrm{AUX}}(\mathcal{W}, B; \tilde{S}) = \frac{1}{2}\sum_{i=1}^{n} ||f(\mathcal{W}, \mathbf{x}_i) + \lambda b_i - \tilde{\mathrm{y}_i}||^2, \text{ where } B = (b_1, b_2, ..., b_n) \in \mathbb{R}^{c \times n}$$

Note that the theoretical results from [2] do not apply to networks that are not in the NTK regime or to loss functions other than $l_2$ loss, however their experiments show that the proposed regularization methods are still very effective even in these scenarios.

### 2.5 Our proposed method: CDR+AUX

As discussed in the previous section, CDR improves the generalization by training only the critical parameters but the gradient used to train these critical parameters is dependent upon all the training examples. In the initial training phase when model hasn't memorized the noisy labels yet, auxiliary variables can help absorb the noise in gradient. Moreover, by properly adjusting lambda value, one can define a curriculum for the training. Thus we propose to combine both CDR and AUX in this report.

## 3 Experimental Setup

To verify effectiveness of our proposed method, we ran experiments on manually corrupted version of MNIST [16] . MNIST has $28 \times 28$ grayscale images of 10 classes including 60,000 training images and 10,000 test images. We consider four types of synthetic label noise, i.e., symmetric noise, asymmetric noise, pairflip noise and instance-dependent noise (abbreviated as instance noise). The noise rates $\tau$ are set to 20%, 40% and 70%. For each setting, we compared the performances of 4 methods presented in the pervious section - SGD, CDR, AUX, and CDR+AUX (our proposed method). The details of the noise setting are described as follows:

- Symmetric noise: this kind of label noise is generated by flipping labels in each class uniformly to incorrect labels of other classes
- Asymmetric noise: this kind of label nosie is generated by flipping labels within a set of similar classes for e.g. flipping $2 \rightarrow 7, 3 \rightarrow 8, 5 \leftrightarrow 6$
- Pairflip noise: this noise flips each class to its adjacent class.
- Instance noise: this noise is quite realistic, where the probability that an instance is mislabeled depends on its features.

We trained a LeNet [16] on MNIST with a batch size of 32. For all the training, we used an SGD optimizer with momentum 0.9 and and a weight decay of $10^{-3}$ when required. The initial learning rate was set to $10^{-2}$. Our the codes were implemented in Pytorch 1.6.0 with CUDA 11.1 and ran on NVIDIA Tesla K80 GPUs.

## 4 Results

In the following table we present the test accuracies achieved by each of the 4 models when trained on datasets with different noise types (Instance, Pairflip, Asymmetric, Symmetric) and noise rates (20%, 40%, 60%). All the accuracies presented below are on the test dataset which does not contain any noise.

From the table we can infer that in general, CDR+AUX is performing better than the other 3 models. Since symmetric noise is the easiest type of noise to deal with, we see that all the models perform well when trained on this noise type even for noise rate as high as 70%. At 70% noise, the Pairflip

| Instance | 20% | 40% | 70% |
|---|---|---|---|
| SGD | 98.14 | 92.46 | 29.23 |
| CDR | **98.45** | 93.96 | 30.89 |
| AUX | 98.27 | 94.21 | 27.86 |
| CDR+AUX | **98.45** | **94.46** | **31.03** |

| Pairflip | 20% | 40% | 70% |
|---|---|---|---|
| SGD | 98.97 | 95.34 | 11.52 |
| CDR | 99.05 | 97.91 | 11.52 |
| AUX | 99.14 | 97.73 | **14.29** |
| CDR+AUX | **99.16** | **98.32** | 11.52 |

| Asymmetric | 20% | 40% | 70% |
|---|---|---|---|
| SGD | 99.05 | 97.56 | 63.86 |
| CDR | 99.18 | 98.85 | 63.67 |
| AUX | **99.15** | 98.76 | 62.33 |
| CDR+AUX | 99.07 | **98.91** | **71.64** |

| Symmetric | 20% | 40% | 70% |
|---|---|---|---|
| SGD | 98.74 | 98.4 | 96.06 |
| CDR | 98.88 | 98.55 | 96.63 |
| AUX | 98.84 | 98.59 | **96.8** |
| CDR+AUX | **98.89** | **98.63** | 96.56 |

Table 1: Test accuracy across different noise types, rates and method

dataset is most challenging. In this case, we see that all models perform very poorly with accuracies a bit higher than 10% (which is just better than random guessing). When the models are trained in the Asymmetric noise setting with 70% noise rate, CDR+AUX outperforms all other models by a accuracy of around 10%. In the Instance Noise type setting, CDR+AUX performs better than the other 3 models for all the 3 noise rates.

Now we check if the auxiliary variables are really helping in training the model.

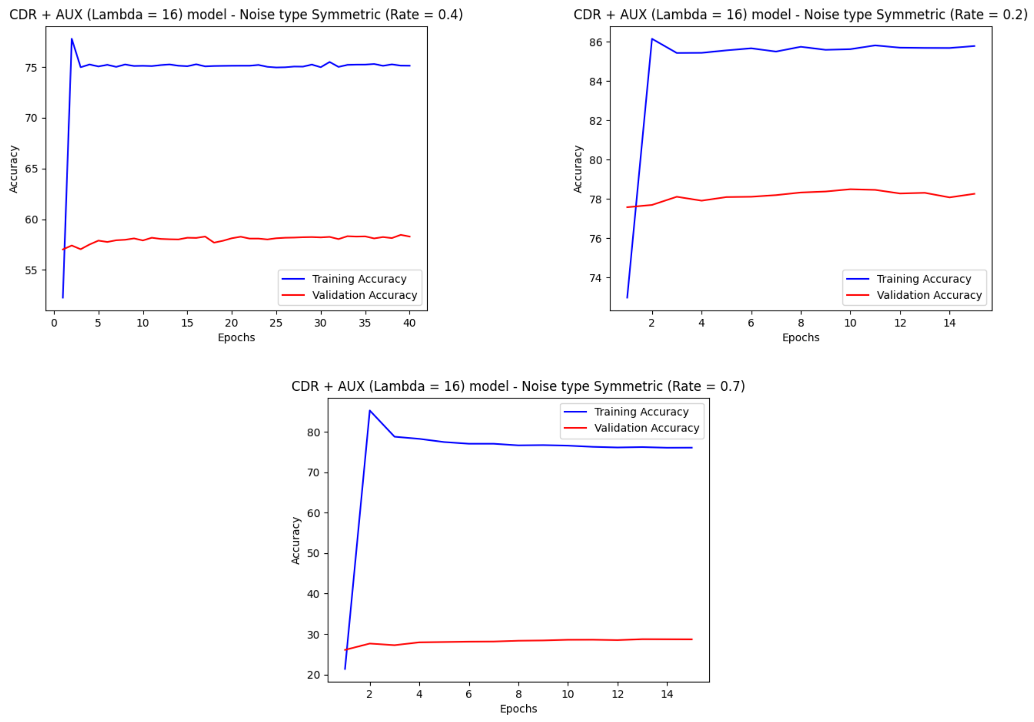

Figure 1: Plots of Training and Validation Accuracy vs epochs

From the plots in Figure 1 we can conclude that Training accuracy is significantly better than the ideal accuracy (1 - $\tau$) (for all the noise rates) but the validation accuracy is close to the ideal performance. Thus, implying that the auxiliary variables are indeed absorbing the noise in training set but at the same time they also allow model to learn from clean data points. Note that we use auxiliary variables in the training set and not in validation or testing.

Now we analyze how changing $\lambda$ affects the performance of the CDR+AUX model for different noise rates.

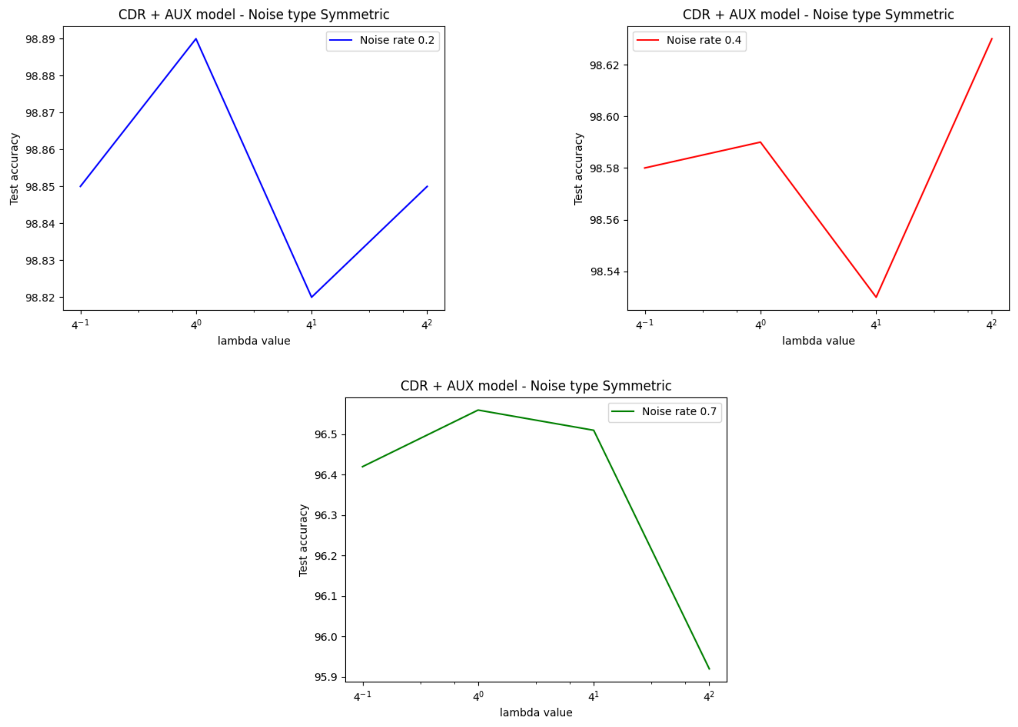

Figure 2: Plots of Test Accuracy vs $\lambda$

The general trend is that initially, the test accuracy increases on increasing $\lambda$, then the test accuracy goes down. This is because when we increase $\lambda$ initially, we are making the model more robust to noisy labels. If we keep increasing $\lambda$ further we would get better training results, but the model memorizes the training set better when $\lambda$ is relatively high, this implies that during test time (where we do not use auxiliary variables), the model would not be able to give good predictions as the output from the auxiliary variable dominates the neural network's output during training.

We have also tried RDI and CDR+RDI methods, but we weren't able to achieve good accuracy using these methods. We suspect this could be because the network is not wide enough to be in NTK regime.

## 5 Conclusion & Future Work

In this work, we successfully show that segregating the set of all parameters into critical & non-critical (CDR), along with adding an auxiliary variable for each training example (AUX) performs better than CDR or AUX alone. This is validated across all noise types and noise rates for the MNIST dataset.

There is an average increase in test accuracy of $1.44$ percentage points over SGD, 0.76 over CDR and 0.89 over AUX. MNIST is much less complex than CIFAR, which is why almost all algorithms obtain very high acccuracy on MNIST. Future work in this domain might include training the proposed methods on larger and more computationally complex datasets, like the CIFAR-10 and CIFAR-100. Performing the experiments on CIFAR might give a more accurate representation of the effectiveness of implementing CDR + AUX over just CDR or AUX alone.

Another possible direction for future work can be using ResNet instead of LeNet. As we were limited by computational resources, we could only train our network using LeNet, due to which our network fails to fall in the NTK regime. This might explain our inability to achieve good results with RDI. As was pointed out earlier, the theoretical results from [2] do not apply to networks that are not in the NTK regime. Thus, using ResNet on datasets like CIFAR-10 or 100 will be a very good judge of the robustness of our proposed method.

## References

[1] Xiaobo Xia, Tongliang Liu, Bo Han, Chen Gong, Nannan Wang, Zongyuan Ge, and Yi Chang. Robust early-learning: Hindering the memorization of noisy labels. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[2] Wei Hu, Zhiyuan Li, and Dingli Yu. Simple and effective regularization methods for training on noisily labeled data with generalization guarantee. In *International Conference on Learning Representations*, 2020.

[3] Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. *ArXiv*, abs/1903.11680, 2020.

[4] Wei Hu, Zhiyuan Li, and Dingli Yu. Understanding generalization of deep neural networks trained with noisy labels. *ArXiv*, abs/1905.11368, 2019.

[5] Bo Han, Jiangchao Yao, Gang Niu, Mingyuan Zhou, Ivor Wai-Hung Tsang, Ya Zhang, and Masashi Sugiyama. Masking: A new perspective of noisy supervision. In *NeurIPS*, 2018.

[6] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *ArXiv*, abs/1611.03530, 2017.

[7] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(3):447–461, 2016.

[8] Zhilu Zhang and Mert Rory Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *NeurIPS*, 2018.

[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. In *ICML*, 2018.

[10] Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron C. Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. *ArXiv*, abs/1706.05394, 2017.

[11] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv: Learning*, 2019.

[12] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: convergence and generalization in neural networks (invited paper). *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 2018.

[13] Sanjeev Arora, Simon Shaolei Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *NeurIPS*, 2019.

[14] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[15] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 254–263. PMLR, 10–15 Jul 2018.

[16] Corinna Cortes Yann LeCun and Christopher J.C. Burges. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998.