CS-439 Optimization for Machine Learning

Optimizer performances on manipulated data

Hilda Abigél Horváth (342844), Nicholas Sperry Grandhomme (341161), Medya Tekeş Mızraklı (342041)
School of Computer and Communication Sciences, EPFL

***Abstract*—The goal of this project was to compare different optimisers on corrupted images and images with corrupted labels to examine how much neural networks can memorise a large amount of data, as well as how well they can generalise on a test set afterwards. We saw that while the networks are able to fit and obtain 100% train accuracy even with fully randomized labels, the selection of optimiser and hyper-parameters can have a significant effect on the training time. However, the generalisation error is not much affected by these choices after full convergence. This project was a part of the Optimization for Machine Learning course at EPFL in 2022.**

I. Introduction

In this project we aim to compare the performance of different optimisers when used for image classification under different dataset corruptions. In Machine Learning, generalisation is very important. Ideally a model that performs well on the training set will also perform well in an unseen test set. However, Deep Learning models have an extremely high effective capacity, enough to memorise an entire dataset of hundreds of thousands of images, and so it can be easy to overfit. In this project, we explored similar randomisation tests as done in [1], and measured how long it takes for different optimisers, namely SGD, ADAM [2], and RMSprop to converge to 100% training accuracy using the CIFAR10 dataset [3] while under different corruptions using the AlexNet model architecture [4]. These corruptions include randomising the train labels at varying levels, shuffling pixels in the image, and fully corrupting every image by replacing them with Gaussian noise. We also measured how well the models generalised on the uncorrupted test set when trained with different optimisers to overfit a corrupted dataset.

II. Background

A. Dataset

For the classification task, we used the CIFAR10 dataset [3]. This dataset contains 50,000 train and 10,000 test 32×32 colour images in 10 classes. We used randomly selected 25,000 images to train the models and the full test set to evaluate the test performances.

B. Model

As the model of the experiment we choose to use the AlexNet convolutional neural network [4].

C. Randomisation tests

To truly understand the effective capacity of Deep Neural Networks, various experiments have been done to see how well they can fit any given data. A popular form of tests are randomisation tests, and these have previously been performed on the Alexnet [4], Inception [5], and MLP architectures in [1]. In this paper, the authors performed various randomisation tests when training on the CIFAR10 [3] and ImageNet [6] datasets, and found that every model was able to fit perfectly or nearly perfectly on the training data, even with completely random labels and noise. More specifically, they performed the following corruptions:

- **True labels:** Original dataset with no corruptions.
- **Partially corrupted labels:** Each label is randomized with probability p where $p \in [0.2, 0.4, 0.6, 0.8]$.
- **Random labels:** All labels are randomised.
- **Shuffled pixels:** The same pixel permutation is applied to the entire dataset.
- **Random pixels:** Each image gets a different permutation.
- **Random noise:** A Gaussian distribution with same mean and variance as the original dataset is used to generate a new dataset.

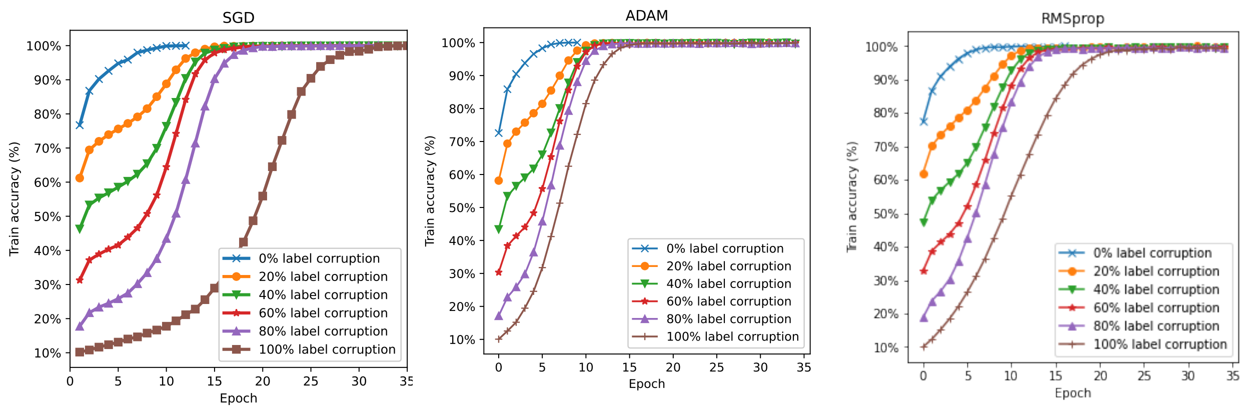
The authors found that with no regularisation methods, SGD was enough to optimise the model weights to achieve 0% training error, and adding weight decay slightly increased the error. Interestingly, AlexNet was able to fit random noise faster than random labels, which the authors say might be because with random noise there is absolutely no correlation between the labels and the images. From their observations, they also claim that once the fitting starts, it converges quickly. However this doesn't seem to be the case for some optimisers, as we will discuss later.

III. Method

For our experiments we used the AlexNet model architecture [4] pre-trained on ImageNet, modifying the final layer to predict 10 classes. Similar to [1], we performed some randomisation tests on the model with SGD, ADAM and RMSprop optimisers. Namely, we measured how long it took to get to 100% train accuracy for the following corruptions:

- **True labels:** Original dataset with no corruptions.
- **Partially corrupted labels:** Each label is randomized with probability p, where $p \in [0.2, 0.4, 0.6, 0.8, 1]$
- **Shuffled pixels:** The same pixel permutation is applied to the entire dataset. This is done in 2 levels, the permutation is either applied along a single axis or along both axes.
- **Random noise:** A Gaussian distribution with same mean and variance as the original dataset is used to generate a new dataset.

We split the original CIFAR10 training dataset in half, for a total of 25,000 images, in order to speed up the training process. We used SGD with a learning rate of 0.01 and momentum of 0.9, ADAM with a learning rate of 0.00001, and RMSprop with a learning rate of 0.00001. All optimisers had a weight decay of $1e^{-4}$. Initially, we found that SGD was less sensitive to the hyper-parameters, and would more quickly converge to 100% train accuracy, while ADAM and RMSprop struggled more. For that reason, we did some minor adjustments to their learning rates until they were able to converge reasonably quickly. Each corruption was applied to the training set separately, and we trained the AlexNet model using the different optimisers under these different



(a) SGD (b) ADAM (c) RMSprop

Figure 1: Convergence on corruptly labelled data for different optimizers - displayed first 35 epochs

corruptions while keeping the hyper-parameters fixed as in the initial runs. A new model was retrained for each corruption. All models were trained until they reached 100% train accuracy. The models trained on the corrupted labels were then evaluated on the uncorrupted CIFAR10 test set to see how well the models generalised when trained using different optimisers.

IV. Training results

A. SGD

Overall the SGD optimiser was able to fit the corrupted labels relatively quickly. From Figure 1a, we see that convergence starts off slow, then picks up, and finally slows down once it gets close to 100% accuracy. Interestingly, for the later corruptions the plots resemble a Sigmoid function, where in the middle convergence seems to be the fastest. We see that each corruption level causes the optimiser to perform worse initially, and causes it to take longer to converge to 100% accuracy. Even though at the end we are fitting absolutely random labels with no correlation to the images, we see that the optimiser is able to change the model's weights such that it is still able to achieve 100% accuracy.
When corrupting the actual images instead of the labels, the SGD optimiser is still able to converge to 100% train accuracy, as can be seen in Figure 2a. We see that permuting the image pixels has a small effect, and the optimiser is able to achieve 100% train accuracy relatively quickly. On the other hand, we see that replacing the images with Gaussian noise causes the optimiser to struggle, taking 67 epochs to achieve 100% accuracy. In the original paper [1], the authors found that the Gaussian corruption converged faster than using random labels, but here we see this is not the case. Table I summarises our findings of the SGD optimiser. We see that for label corruption, it takes nearly twice as long to get to 100% accuracy as it does to get to 99%. On the other hand, for the image corruptions we only need a few more epochs to go from 99% to 100%.

Corruption type	Initial acc.	99% acc.	100% acc.
No corruption	76.69	9	12
20% label corruption	61.20	15	31
40% label corruption	46.26	16	35
60% label corruption	31.27	17	31
80% label corruption	17.75	19	32
100% label corruption	10.19	32	44
Noise	10.22	63	66
X-axis permutation	53.48	13	17
X- and Y-axis permutation	46.28	15	20

Table I: SGD summary - Corruption type, Accuracy after first epoch, First epoch with 99% accuracy, First epoch with 100% accuracy.

B. ADAM

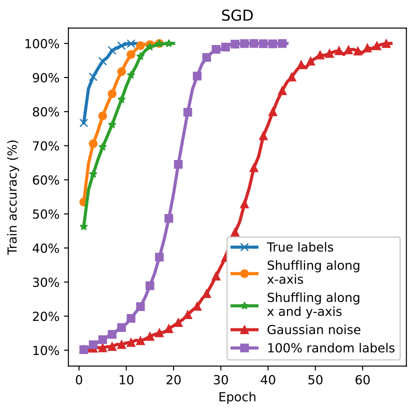
The ADAM optimiser during the training on the corrupted labelled data performed worse for each corruption probability. From 0.4 label corruption probability, the accuracy through the epochs shows similar convergence, just starting from lower accuracy as the probability is higher. Here, for the totally corrupted labels the convergence time to 99% accuracy was not significantly increased unlike with the other optimisers. However it took longer to reach 100% accuracy after reaching 99%. Also here we experienced the same trend with other optimisers in terms of the more corrupted the labels, the more epochs we need for convergence.
As for the corrupted images, ADAM performed really well on the permuted pixels and on noise-manipulated data. It converged to 100% faster than in the label-corrupted cases. However, in convergence to 99% accuracy we can see almost the same numbers in every manipulated cases, but for label corruption it took more epochs to reach the 100% after converging to 99%. It might be because, the manipulated images still have some relation to the original data, but label corruption needs the model to memorise totally the dataset. Table II summarises our findings for the Adam optimiser.

Corruption type	Initial acc.	99% acc.	100% acc.
No corruption	72.56	7	10
20% label corruption	58.14	11	32
40% label corruption	43.36	12	40
60% label corruption	30.31	12	73
80% label corruption	17.12	14	167
100% label corruption	10.06	16	200<
Noise	10.25	16	19
X-axis permutation	52.02	11	15
X- and Y-axis permutation	44.01	13	16

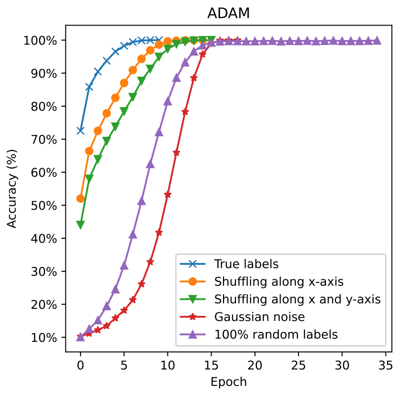
Table II: Adam summary - Corruption type, Accuracy after first epoch, First epoch with 99% accuracy, First epoch with 100% accuracy.

C. RMSprop

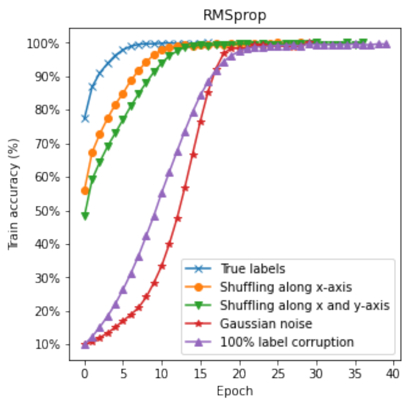
Similar to SGD and ADAM, for RMSprop optimiser, increasing the label corruption level resulted in lower initial accuracy values and longer convergence times. However the effect on convergence time is much more striking here. Although all models were able to obtain a 100% accuracy, it can be seen in Figure 1c that while the model was able to reach 100% accuracy on the original images within 17 epochs, this



(a) SGD



(b) ADAM



(c) RMSprop

Figure 2: Convergence on corrupted images for different optimizers. Note that here the x-axis is different for each optimiser. The lines end once 100% train accuracy is reached.

number increased to 213 epochs when the corruption level is gradually increased to 100%.

Corruption type	Initial acc.	99% acc.	100% acc.
No corruption	77.39	8	17
20% label corruption	62.03	13	49
40% label corruption	47.22	15	65
60% label corruption	32.73	16	69
80% label corruption	18.79	16	112
100% label corruption	10.00	25	213
Noise	9.94	23	30
X-axis permutation	55.98	13	29
X- and Y-axis permutation	48.22	16	37

Table III: RMSprop summary - Corruption type, Accuracy after first epoch, First epoch with 99% accuracy, First epoch with 100% accuracy.

From Table III we can see that while the first epoch at which the model obtains a 99% accuracy is more or less similar for different levels of corruption, the drastic change happens in the number of epochs spend to get 100% accuracy.

As can be seen in Figure 2c, corrupting the actual images through shuffling of x-axis, shuffling of x and y-axis and replacing with Gaussian noise had a milder effect on convergence times with RMSprop. All models were able to reach 100% accuracy in 29, 37 and 30 epochs, respectively. Similar to both SGD and ADAM, initially the convergence rate of random labels is faster than of the Gaussian noise with RMSprop. However towards the end Gaussian noise takes over random label as it did with ADAM.

D. Comparison

For the corrupted labels, in terms of getting to 100% train accuracy, SGD performed best as it was able to converge to 100% accuracy each time the quickest with a higher learning rate. On the other hand, RMSprop and ADAM were much faster at converging to 99% train accuracy compared to SGD, but then struggled to fully converge to 100% and would hover at a high 99.8/99.9% train accuracy for a longer time for the larger label corruptions.

When fitting shuffled pixels all models converged more or less similarly. However when fitting random noise, we found that ADAM performed best, being able to achieve 100% accuracy within 20 epochs. As expected RMSprop performed slightly worse than ADAM and took slightly longer to converge to 100%accuracy. On the other hand, SGD took much longer to fit, taking nearly 70 epochs which was not the case for any of the other treatments applied.

V. Generalization results

Finally, we wanted to find out if different optimisers could help generalise better on the unseen, uncorrupted test set. Each model trained on the varying levels of label corruption with different optimiser was evaluated on the uncorrupted test set and the expectation was to see increasing generalisation error with increasing corruption level. Looking at Figure 3, we can see that while our expectations are confirmed, there is practically no difference on generalisation when using different optimisers, as they all seem to result in the same generalisation error at each corruption level. We do see that although RMSprop struggled to converge to 100% train accuracy, it still achieves a slightly lower generalisation error.

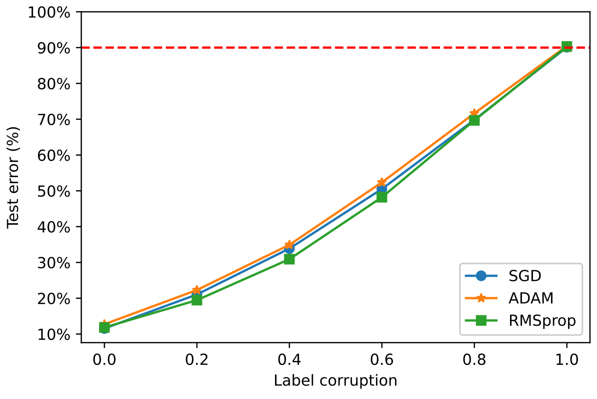


Figure 3: Generalisation errors of different optimisers on the uncorrupted test set. The red line represents the highest error in this case, i.e. just guessing the classes.

VI. Conclusion

To conclude, inspired by the experiments done in [1], we have compared the performance of different optimisers on images with corrupted labels and corrupted images through a series of systematic experiments. In the end we saw that while all the optimisers were able to fit the given data and obtain a 100% train accuracy even under extreme corruptions, the selection of optimiser and the hyperparameters can have a significant effect on the training time. However, the generalisation error is not effected by the choice of optimiser and increases with the level of corruption.

Possible future work can be experimenting with other parameters for Gaussian noise, or try other densities. Also other label corruption methods and optimizers can be also examined.

References

[1] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM*, 64(3):107–115, feb 2021.

[2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[3] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[5] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.