Final Project Report

Analysis of Parameter Dynamics in Image Classification Models

Saravanan Govindarajan (sg3896) and Simran Tiwari (st3400)

1. Abstract

Despite the astonishing success of image classification models, the mechanism by which their inner layers act has not been understood completely yet. This project aims to understand CNNs by studying evolution of parameters and performances (loss, accuracy) of image classification models during and after training for various initialization techniques. We study the feature relationships that the networks extracts for performing image classification. We train 126 models and make numerous plots which visualize the information we extracted by our experiments.

2. Introduction

Image classification has been studied extensively in Deep Learning. Several labeled datasets have been created for learning and many neural network architectures have been proposed to solve the image classification problem.

Despite the practical success of such models, the inner mechanism of neural networks have remained mostly unknown. This unclear performance of hidden layers in CNN models is one of the greatest drawback of such models. Treating NN as a successful black-box for computationally intensive problems prevents us from extending their applications to sensitive areas such as medical applications and autonomous vehicles. Recently, there have been some efforts to shed light on the dynamics of inner parameters of a deep network and explain the rationale of its success.

We study the dynamic internal parameters of a network during the training as well as the relation between learned features. Given the numerous networks and

datasets available, we narrow down our study to - Lenet, Resnet18, and VGG, and train them to classify images on 3 datasets- Cifar10, Fashion MNIST, and MNIST. Nevertheless, our methodology is quite general and results can be extended to other models and datasets with minimal extra effort.

More specifically, we aim to answer the following questions:

1. What percentage of the initialized parameters change after the training? Shallow (LeNet) vs deep networks (VGG, ResNet) - do they have different characteristics in this context? Does the weight initializer play a role in this case?
2. How distant (norm) are the trained parameters from initialized parameters? Does the learning optimizer affect this characteristic?
3. If we take a pretrained model and flip the sign of its weights, does the model again converge back to the same weights?
4. If we take a trained model and add a small distortion to the weights, does the model converge back to the same weights?
5. Are learnt features independent of each other or is there any correlation? Does the choice of optimizer have any impact on it?

3. Approach

3.1 Implementation Details

We trained three different neural architectures, namely Lenet, Resnet18, and VGG19, which have 3246, 11 × 106 , and 138 × 106 trainable parameters in 5, 16, and 19 layers, respectively. In order to make all models compatible with our datasets, we changed the output size of the FC layer to match the number of classes in our datasets, i.e. 10.

We train each architecture over three datasets, CIFAR10, MNIST, and Fashion MNIST. Each dataset contains 60000 samples which is partitioned to subsets of size 50000 and 10000 samples for training and testing. All three datasets we used have 10 different classes and the goal of each trained network is to classify images to one of the 10 classes. Below table denotes more details about these datasets.

Name	Image size	Description
CIFAR10	$3 * 32 * 32$	Objects including airplane, automobile, ..., truck.
MNIST	$1 * 28 * 28$	Handwritten digits $(0 - 9)$
Fashion MNIST	$1 * 28 * 28$	Clothing including T-shirt/top, Trouser, ..., Ankle boot.

Figure 1: Datasets used and their description

Figure 2: Visualization of datasets

3.2 Experiments

Our project aims to answer the following questions:

1. What percentage of the initialized parameters change after the training? Shallow (LeNet) vs deep networks (VGG, ResNet) - do they have different characteristics in this context? Does the weight initializer play a role in this case?
2. How distant (norm) are the trained parameters from initialized parameters? Does the learning optimizer affect this characteristic?
3. If we take a pretrained model and flip the sign of its weights, does the model again converge back to the same weights?
4. If we take a trained model and add a small distortion to the weights, does the model converge back to the same weights?
5. Are learnt features independent of each other or is there any correlation? Does the choice of optimizer have any impact on it?

In order to answer these questions we perform 3 experiments. s. In the first experiment, we analyse weight movements on different initialization methods . In the second experiment, we study the relation between features extracted by different models with different parameters. In the third experiment, we analyze the effect of small weight distortions on performance. Details of each experiment is explained in the following.

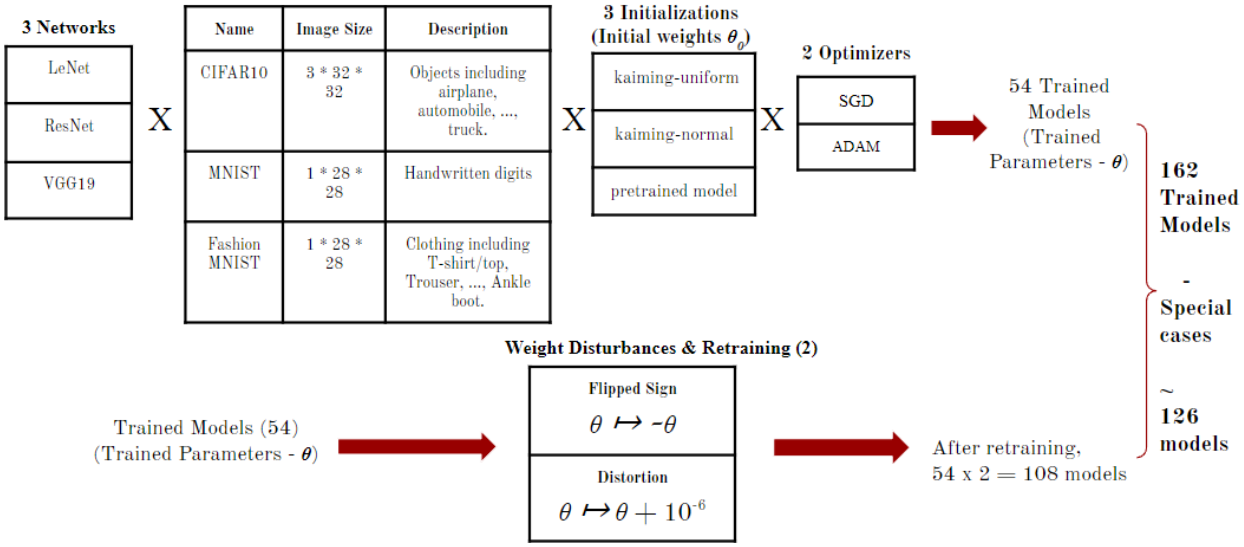


Figure 3: Visualization of experimental architecture

3.2.1 Experiment 1 - Analyse weight movements on different initialization methods

In this experiment we track the evolution of the parameter θ. For a model with fixed initialization and optimizer let θi denotes the vector of trainable parameters for i = 0, 1, ..., 20. At epochs 0, 5, 10, 15, 20 we plot the distance of θi with the previous checkpoints, i.e. ||θ5- θ0 ||, ||θ10 - θ5 ||, ||θ15 - θ10 ||, ||θ20 - θ15 ||. Moreover, we tracked the portion of parameters that remain fixed and those who change during the training and plotted their relative norm and frequencies. We also plot the histogram of both fixed and changing parameters at the initial and final step.

3.2.2 Experiment 2 - Analyse relationship between features

Analyze correlations between features (θi, θj) where $i \neq j$ in the trained feature layer - second last layer of the CNN will be computed and compared. Similarly, pairwise correlations between feature layers from different models was also

compared. This is to evaluate if the models learn the same features irrespective of the weight initialization, learning optimizer etc. Correlation can also give us a sense if the models with different initializations learn features in the same order .

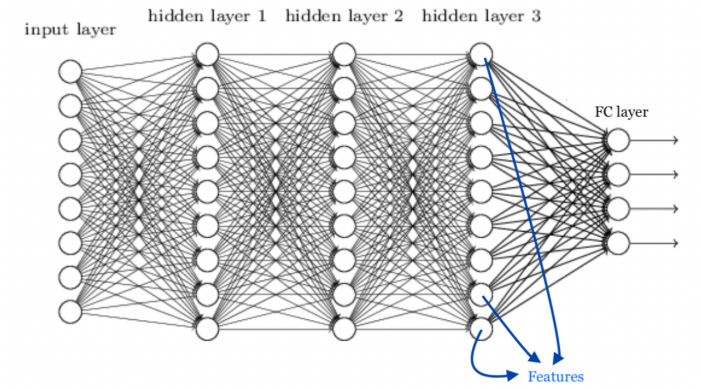


Figure 4: visualization of network architecture from Experiment 3

3.2.3 Experiment 3 - Analyse effect of small weight distortions on performance

Wel perform weight perturbations such as sign flipping, adding small noise etc. on the learnt parameters, and retrain the network with the perturbed weights. We evaluate if the new weights converge to the original weight values. We use 2 methods of distorting θ:

1. Θ → Θ + 10^−6 × 1
2. Θ → - Θ

4. Results

We perform the 3 aforementioned experiments to answer our questions. Using Google Colab, we train 126 models and make numerous plots which visualize the information we extracted by our experiments. We summarize our observations and results in this section and leave further analysis to future works. Due to limited space we only show a few of the plots in this report.

1. Resnet tends to keep over 60% parameters unchanged regardless of dataset, optimizer, initial point whereas LeNet and VGG had >90% parameters changed. We think it is due to lazy training phenomenon.

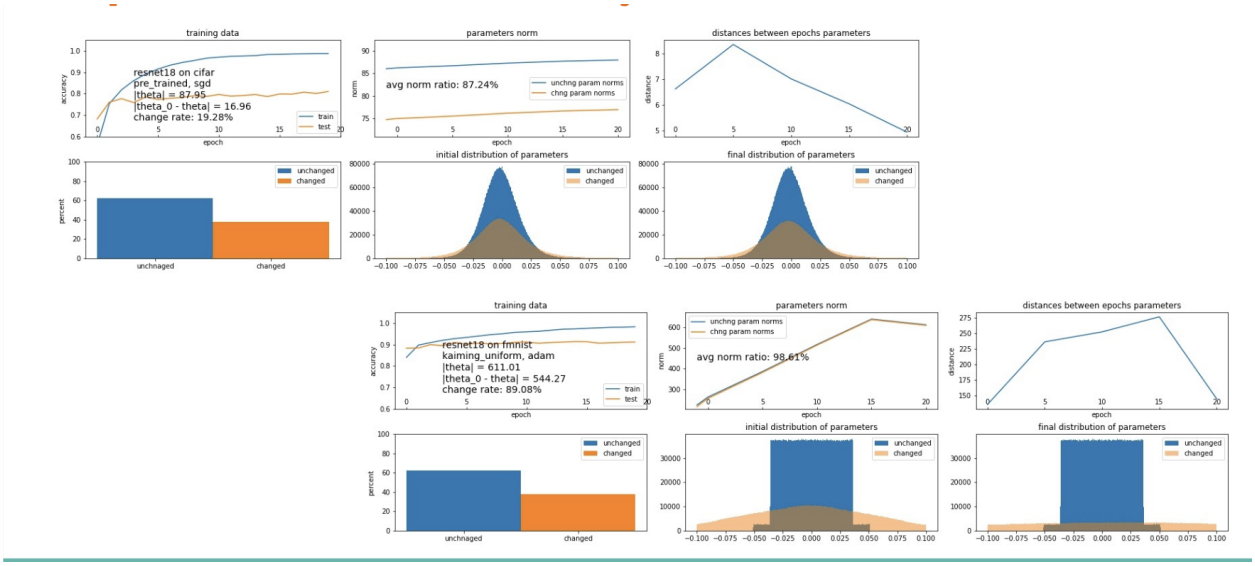
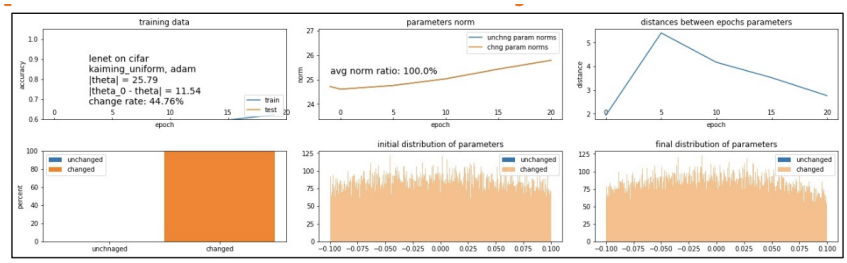


Figure 5: A sample visualization of part 1 analysis



Lenet on Cifar:

100% Parameters changed

VGG on Cifar:

>90% Parameters changed

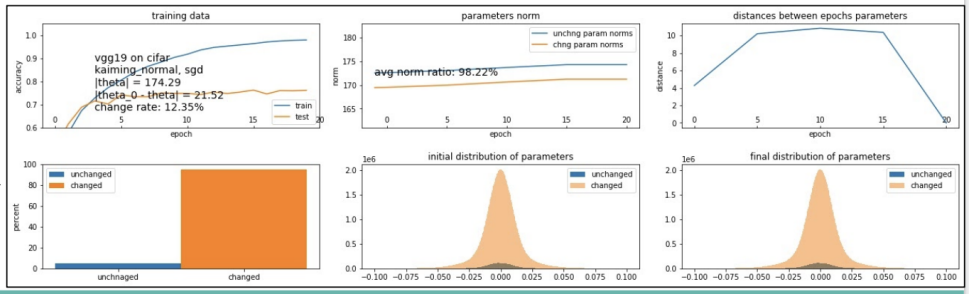


Figure 6: A sample visualization of 1 analysis

2. SGD finds a close local minima for huge networks.

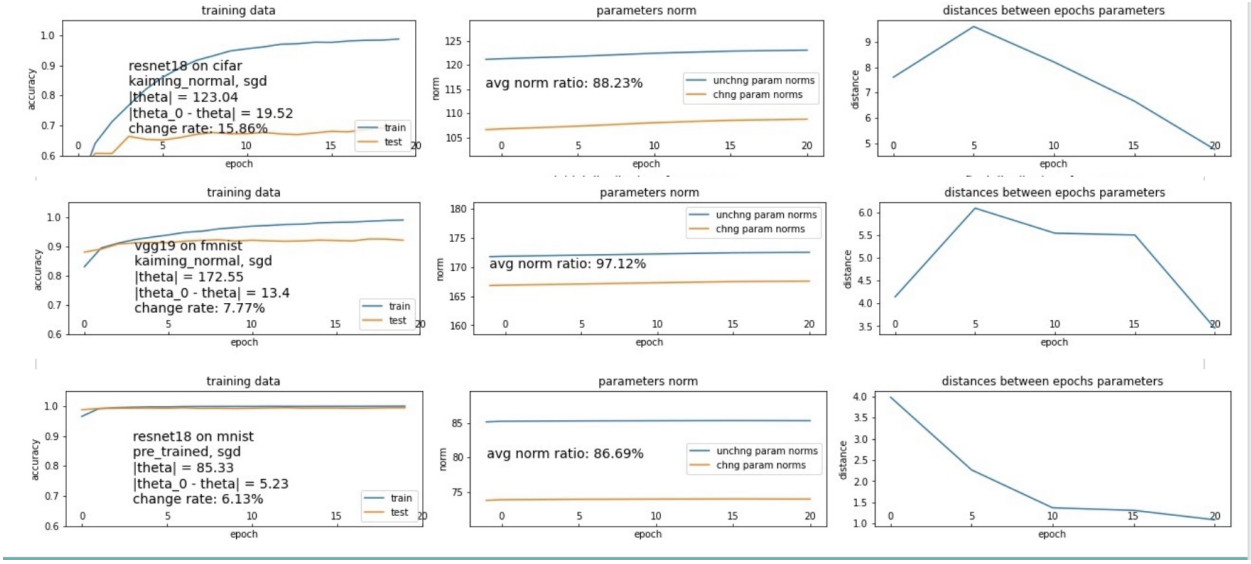


Figure 7: A sample visualization of part 2 analysis

3. We observed that when we train a same model with SGD and ADAM, we saw a very strong correlation between features trained with ADAM.

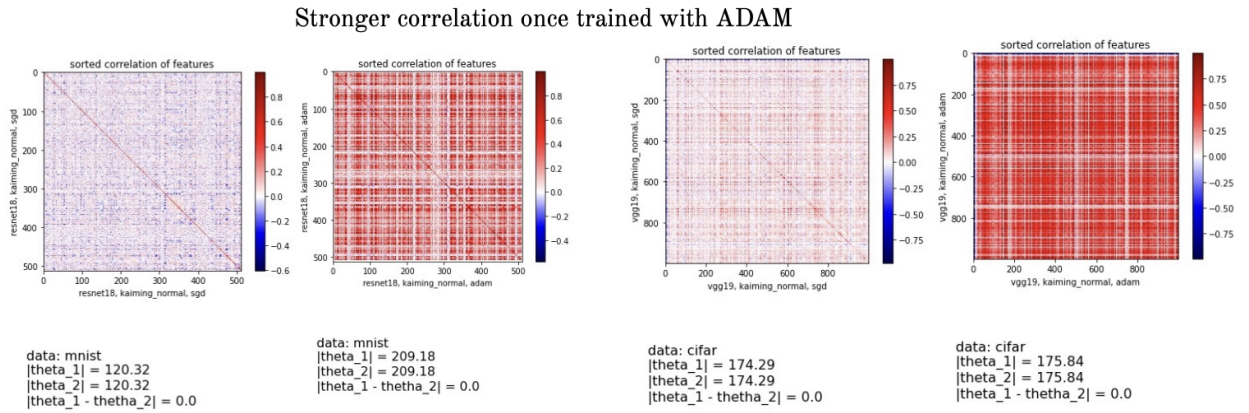


Figure 8: A sample visualization of part 3 analysis

4. Coming to the distortion effect, it seems that distortion has very minimal effect on the training process. They more or less converge to the same features even with weight distortions. Here, we are looking at plot between models features and it's features trained with model distortions. Also, when we compared the norm between the parameters of both the models after training, they were very close suggesting that it has minimal effects.

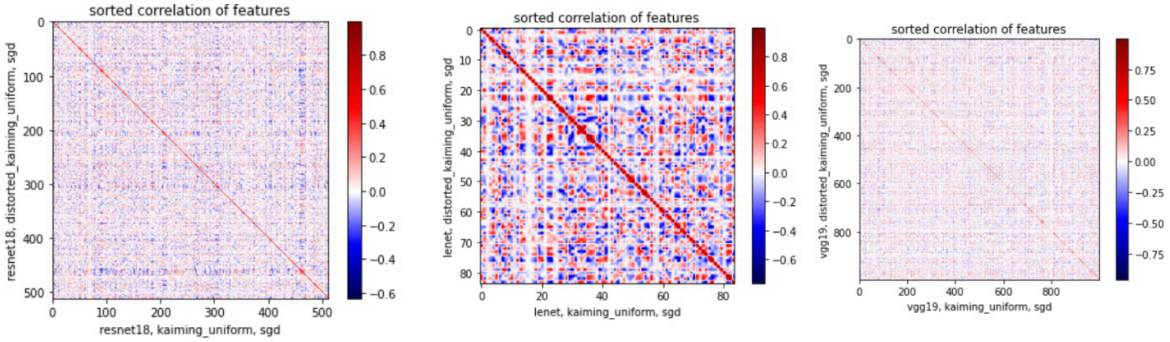


Figure 9: A sample visualization of part 4 analysis

5. Whereas, when we flipped the weights and retrained the models, we observed that it forced the local minima to be very far from the initial point. This was obvious when we compared the norm distances between the original model and it's flipped weights model.

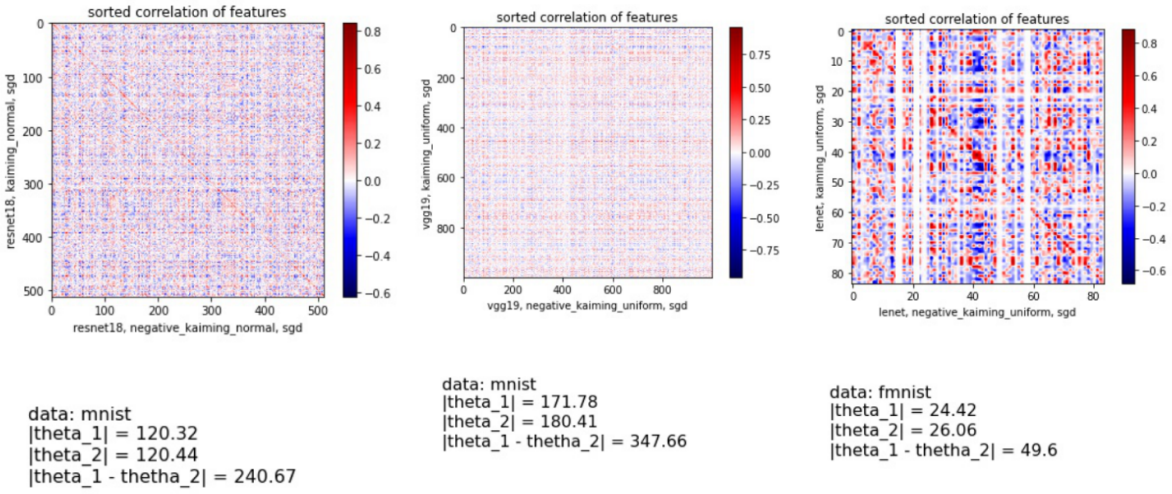


Figure 10: A sample visualization of part 5 analysis

5. Conclusion

Our experiments and analysis of the results lead to the following conclusions:

1. Huge architectures trained by ADAM optimizer, the extracted features have stronger positive correlations among each other as compared to trained models with SGD. This didn't happen for the small network Lenet.

2. 60% of the parameters in ResNet tends to remain unchanged as a result of lazy training
3. Small distortion doesn't usually change the model and features too much. They more or less come back to the same initial model (distance of vector of parameters and very high correction of features)
4. Flipping the sign of a trained parameter and retraining it forced the network to find a local minima far away from a trained one.

6. References

1. Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In International Conference on Machine Learning, pages 531–540. PMLR, 2018.
2. Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. arXiv preprint arXiv:1812.07956, 2018.
3. Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In International Conference on Machine Learning, pages 1019–1028. PMLR, 2017.
4. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision, pages 1026–1034, 2015.
5. 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 (CVPR), June 2016.
6. Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. arXiv preprint arXiv:1609.04836, 2016.
7. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25:1097–1105, 2012.
8. Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In Proceedings of the IEEE, pages 2278–2324, 1998.
9. Michael A Nielsen. Neural networks and deep learning, volume 25.

Determination press San Francisco, CA, 2015.

10. Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision, pages 618–626, 2017.