Catastrophic Forgetting in Shallow Neural Networks

Ankur Mali and Yevhen Tupikov

May 1, 2017

Abstract

Catastrophic forgetting is widely believed to be a serious problem faced by many Machine Learning models and algorithms primarily Neural Networks. When network is trained one new task say TASK A, then trained on another task say TASK B, many computational neural models "Forget" how to perform TASK A. Here we scrutinize the extent to which catastrophic forgetting problem is observed in modern shallow neural networks, comparing methods such as neurogenesis, replays and recent Fisher information matrix .We also examine the relationship between learning on the first task and learning on the second task on catastrophic learning. We find that catastrophic forgetting is most noticeable effect in neural networks and is more conspicuous in shallow networks compared with deeper architecture. We found that it is always better to train network with Fisher information matrix - the Fisher information matrix is consistently better at preserving old features and adapting to the new task and also have better tradeoff curve. We compare our results with newly trained networks on Task A and Task B individually and Transfer Learning approach.

1 Introduction

Biological and Machine Learning systems including neural networks are affected by widely known problem termed as catastrophic forgetting[3]. When neural networks are trained on Task one, then trained on new Task, networks "Forget" how to perform the previous task[6].For instance, a neural network trained on a convex objective will always reach the same grouping at the end of training on later task, regardless of how the network were initialized.This means that weights trained on "Task A" will get reinitialized and will converge for "Task B" while training on later task.Thus "Forgetting" previously learned weights. Whenever the neural networks can correctly predict previous label or image, this is only due to similarities between the two task. The Model for biological learning in humans indicates that available neocortical neurons learns using an algorithm which is prone to catastrophic forgetting[4]. The neocortical learning algorithm is enhanced by a virtual experience system that performs replays of memories stored in the hippocampus. This method continually reinforces tasks that have not been recently performed and helps to improve overall performance[4-6].In this research project, we investigated the extent to which catastrophic forgetting affects training procedure in common neural networks. Neuroscience literature shows evidence that the relationship between the old and new task strongly influences the outcome of the two successive learning experiences. Consequently, we examine three different types of methods using neurogenesis, replays and Fisher information matrix.We found that catastrophic forgetting is more prominent in shallow networks compared with deeper architecture.Experiments show that above three approaches slow down the forgetting process, but results are not comparable with transfer learning approach.

2 Datasets

To lower the computational costs of training the models and to simplify the analysis we selected a canonical machine learning dataset MNIST [1]. MNIST dataset contains standardized 28x28 pixel greyscale images of handwritten digits (for example see Figure 1) and is one of the first successful applications of neural networks to real-world problems. To reproduce the results of the paper on catastrophic forgetting [2] we generated a new dataset by taking images from the original MNIST dataset and permuting their pixels in the same way. Figure 1 shows an example of digit one in the original and shuffled datasets. For the experiments on learning new classes we split

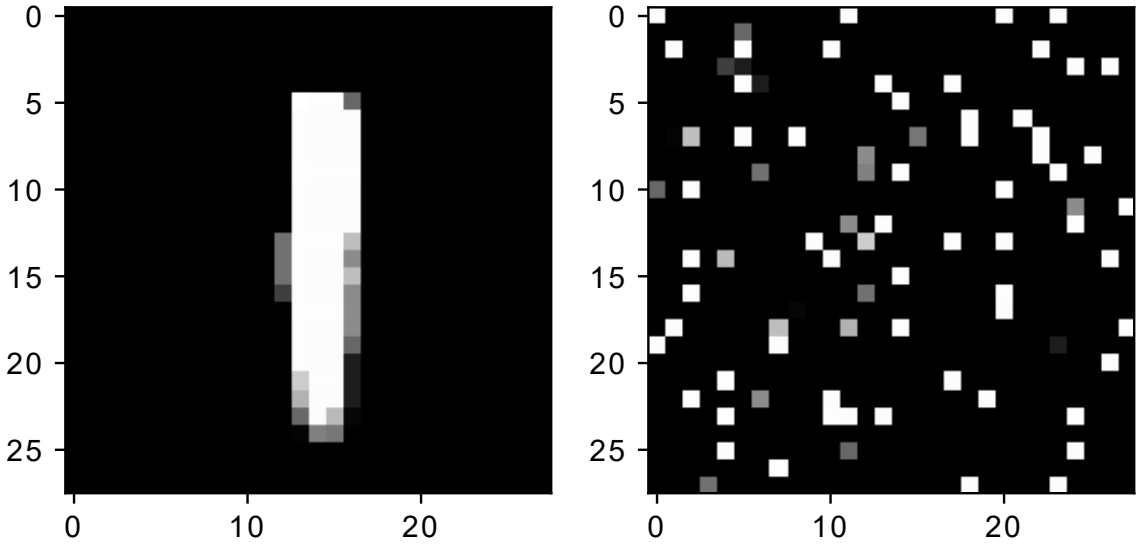

Figure 1: Samples from the used datasets. Left: example of an image of handwritten digit one from original MNIST dataset. Right: the same digit one in the shuffled dataset

the MNIST dataset into two parts: one containing images of digits 0-9 excluding digit 5; and the second containing only images of digit 5. The choice of digit 5 was done randomly.

3 Shallow network architecture

To check whether introducing new penalty term, defined by Fisher information matrix, helps to preserve the knowledge learned in the previous task, we started with the simplest network architecture consisting of a single hidden layer. One hidden layer network consisted of one input layer with 28x28=784 neurons, hidden layer with 64 neurons and output layer with 10 neurons representing 10 classified digits. All neurons were modeled as standard rectified linear units (RELU), whose activation is a nonlinear function of input: $a_i = \max(0, x_i)$, where a_i - activation of the RELU neuron i; x_i - total input to neuron i. All network layers were modeled as fully-connected layers and therefore total input to neuron i is defined by the following expression: $x_i = \sum_j a_j w_{ij} + b_i$, where w_{ij} - weight from neuron j in the previous layer to neuron i; b_i - bias of neuron i. Network weights were optimized by the backpropagation method using Adam optimizer with learning rate = 1e-3 and mini-batch size of 10 samples. Experiments were run with custom written python code using tensorflow environment.

First, we decided to reproduce the results of the paper on catastrophic forgetting to ensure that we correctly calculate and apply Fisher information matrix. Figure 2 shows the results on original (task A) MNIST and shuffled (task B) datasets for pure stochastic gradient descent (SGD left) and gradient descent with additional Fisher penalty term (right). Pure SGD cannot keep the knowledge of the task A, since weights during learning on B change in the direction of increased performance on task B, thus leaving the region of good performance on task A. On the other hand, task B is learned quickly. The balance between performances on both tasks is achieved when lines representing accuracies intersect. For SGD it happens at 93.1% of accuracy. Weights constrained with Fisher information matrices remain in the region of good performance on task A for a long time. However, learning of task B progresses slowly since now weights cannot change in the direction of the best performance of task B. Still task B can be learned and both accuracies intersect at 94.3%. The difference of 1.2% may seem to be small, however, one needs to take into account that we used only two tasks and the simplest network architecture with one hidden layer. For example, the simplest network used in [2] contained two hidden layers, but we were able to achieve similar results with one layer only.

Then we applied the same approach in the context of our initial problem: learning on new classes. Figure 3 shows performance for pure SGD (left) and weights largely constrained (right) on dataset with digits 0-9 excluding 5 (task A) and dataset with digits 5 (task B) during learning on task B. Pure SGD again fails to preserve the knowledge of the previous information. Surprisingly,

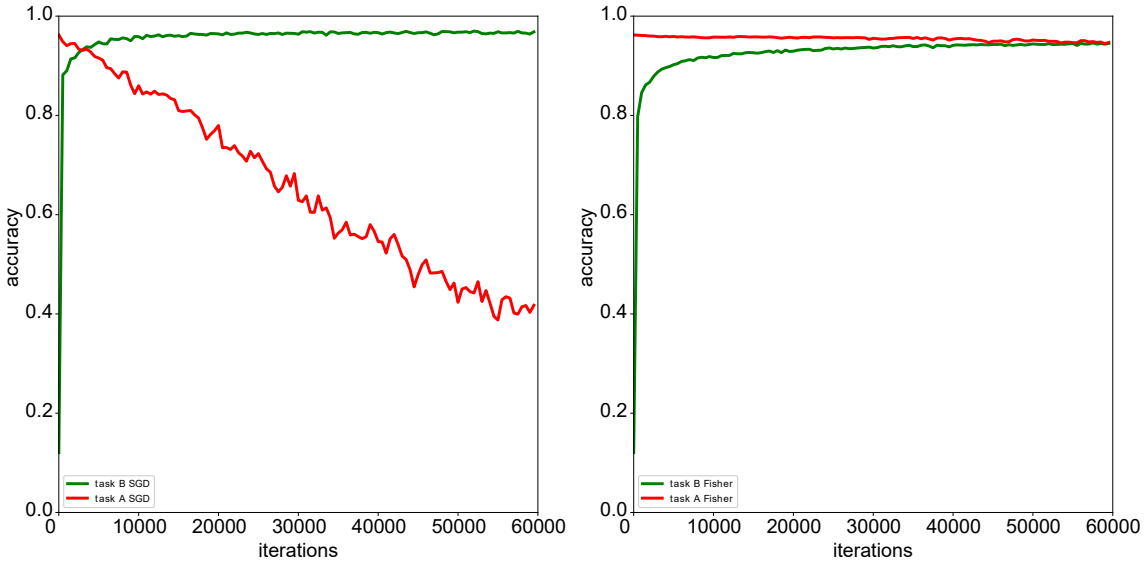

Figure 2: Penalizing weights keeps the knowledge of the previous task. Left: pure stochastic gradient descent results in continuous decline of performance on previous task A, while task B is being learned. Right: constraining weights with Fisher matrices preserves the knowledge of the previous task A. Task B can still be learned.

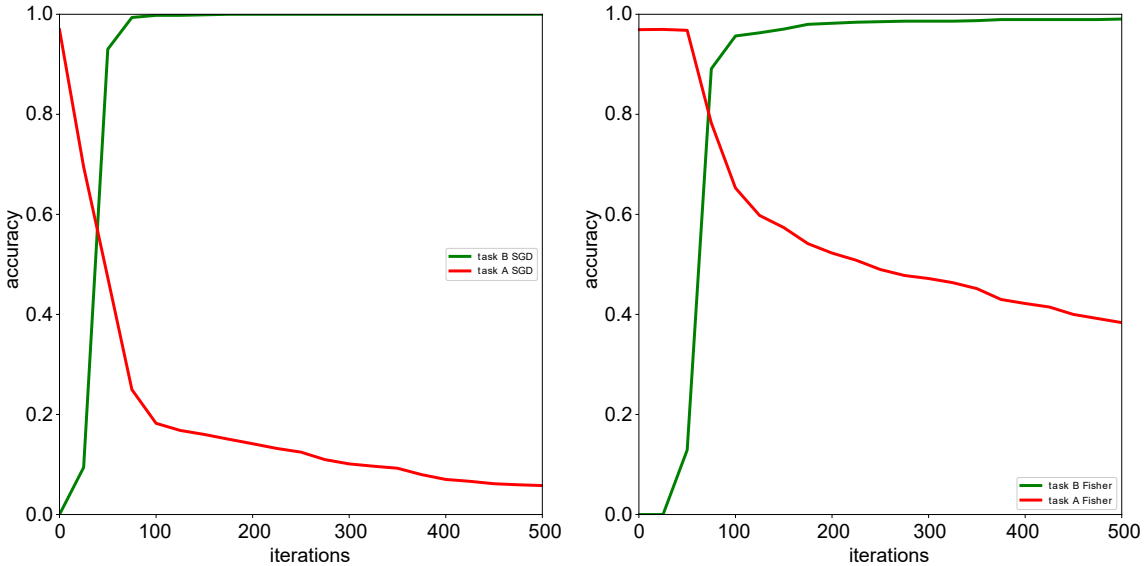

Figure 3: Penalizing weights doesn't help to keep the knowledge of the previous classes. Left: pure SGD results in fast forgetting of the previously learned classes (task A), while new class (task B) is being learned. Right: very large constrain of $\lambda = 10^6$ is able to keep the knowledge of the previously learned classes only for a short time. When performance on new class takes off, performance on the previous classes goes down

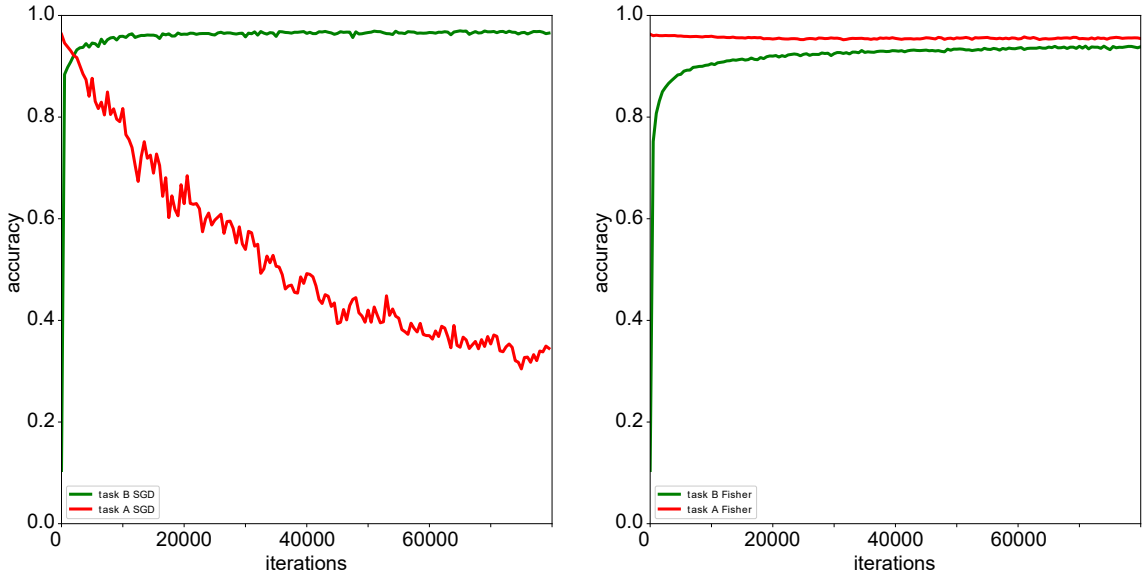

Figure 4: Penalizing weights keeps the knowledge of the previous task in deeper network with 2 hidden layers. Left: pure stochastic gradient descent results in continuous decline of performance on previous task A, while task B is being learned. Right: constraining weights with Fisher matrices preserves the knowledge of the previous task A. Task B can still be learned.

no matter how strongly we penalized the weights, performance on the previous classes also went down when new class was being learned. Initially strong constrain prevents the weights from changing and performance on task A remains good. At the same time, performance on task B keeps very low. Finally, the weights corresponding to the smallest values of Fisher information matrices (close to zero and zero) start to change and task A performance goes down. We concluded that shallow architecture cannot handle learning of new classes and decided to experiment with deeper architectures to see if that can solve the issue.

4 Deeper network architectures

Next step was to try neural network with two layers. In our experiments we added one additional hidden layer with 64 neurons. With this architecture we obtained similar results on original and shuffled MNIST datasets. Figure 4 shows that SGD fails to keep the knowledge of task A, which can be rescued by adding Fisher penalty term. Unfortunately, 2-hidden layer network results on the task of learning new class was similar to the ones obtained with a single hidden layer (see Figure 5). Again, even strongly constrained weights cannot keep the knowledge of the previously learned classes. We saw similar results with the network containing 4 hidden layers with 64 neurons in each of them. Therefore we think the obtained result is general and we are facing a fundamental issue that prevents the network to learn all classes.

5 Transfer Learning

Transfer learning is about leveraging the knowledge we gained by building a machine learning model for a particular task and applying that knowledge to an another task. It takes a trained neural network and leverages the features extracted by that network on an input data and then uses previously learned knowledge for another task[7]. We carried out experiments for shallow and deeper architecture using simple neural networks and also experimented with Convolution model. Better performance for the simple neural network is observed using two layers achieving 94 accuracy, and Convolution model has shown better result compared with simple feed forward networks giving 99.2. Table 1 shows experimented model results.Transfer learning has shown good results in simulation and knowledge sharing scenarios.However, Transfer learning requires a significant amount of data with deeper architecture to work better.In practice, we found that Transfer learning works better than above-proposed method since weights are not allowed to changed in the

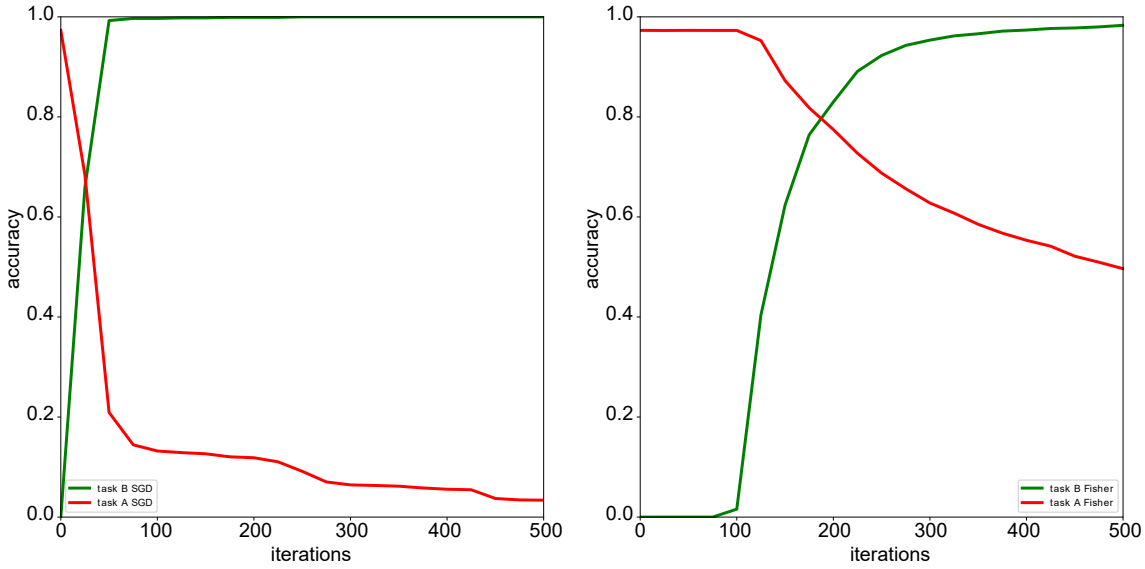

Figure 5: Penalizing weights doesn't help to keep the knowledge of the previous classes in the deeper network with 2 hidden layers. Left: pure SGD results in fast forgetting of the previously learned classes (task A), while new class (task B) is being learned. Right: very large constrain of $\lambda = 10^6$ is able to keep the knowledge of the previously learned classes only for a short time. When performance on new class takes off, performance on the previous classes goes down

Method	Simple Neural Networks	Convolution Neural Networks
Neurogensis	80.345	88.345
Replays	84.54	94.45
Transfer Learning	94	99.2

Table 1: Method results on Second Task.

process.Transfer learning is the mechanism to avoid catastrophic forgetting but cannot solve this problem.In our experiments, we found that transfer learning is inefficient when used with shallow networks.For Transfer learning, to work, there should be proximity in the two task.

6 Possible explanation and conclusions

As we saw, the approach of constraining weights using Fisher information matrices works very well with learning new tasks of similar complexity, but completely fails when we tried to learn new classes. We believe the problem is that output neuron representing new class is not able to learn when to activate and when to remain silent. During the first training, neuron representing digit 5 always remained inactive. In other words this neuron learns to be inactive for any type of input. Therefore in case of strongly constrained weights in the beginning of the second training, learning is delayed: neuron slowly finds the way to change its state to active. Unfortunately, the way for the neuron to be active interferes with the previous way for it to be inactive, which corrupts the performance on previous classes. Output neuron simply doesn't learn anything useful in the first training, while in the second training it only learns to be active all the time. Thus we conclude that constraining the weights using Fisher information matrices is not an efficient approach for transfer learning. Other methods like transfer learning are widely used to avoid catastrophic forgetting.Since we don't allow weights to change and only fine tuning the pre-trained model we can achieve better performance.In our experiments, we found that we can achieve 99.2 accuracy on the Second Task using 2 layer deeper Convolution neural network.Unfortunately Transfer learning is not useful in shallow networks and depended on large architecture and data for training. Transfer learning tries to avoid catastrophic forgetting instead of solving the problem.In future we would try to build better weights restriction function which would help to avoid catastrophic forgetting.

References

[1] http://yann.lecun.com/exdb/mnist/

[2] J. Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks", PNAS 2017, 114 (13), 3521-3526

[3] McCloskey, M. and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In Bower, G. H. (ed.), The Psychology of Learning and Motivation, Vol. 24, pp.109–164. Academic Press, San Diego, CA, 1989

[4] McClelland, J. L., McNaughton, B. L., and O'Reilly,R. C. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. Psychological Review,102:419–457, 1995

[5] Hinton, Geoffrey E., Srivastava, Nitish, Krizhevsky,Alex, Sutskever, Ilya, and Salakhutdinov, Rus- lan. Improving neural networks by preventing coadaptation of feature detectors. Technical report,arXiv:1207.0580, 2012.

[6] An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks,arxiv:1312.6211,2013.

[7] Transfer Learning Handbook University of Wisconsin, Madison WI, USA,2009 ftp://ftp.cs.wisc.edu/machine-learning/shavlik-group/torrey.handbook09.pdf