1 Experiments

Reproduced results of the previous study, train phase, and the hyperparameter tuning process are explained in chapter **??**. For all runs, scratch implementations are used with minimal Pytorch functionalities. They are performed three times and plotted with their mean and standard deviation or with their confidence interval. The same architectures are used for BP and DFA, meaning that we have only a hidden layer with 512 neurons, and reLU is used as a non-linear function. BCE is used as a loss function and, sigmoid is used for the non-linearity of the last layer. Networks are trained for twenty epochs if others are not specified, and at each epoch, train and test datasets are recreated as it is explained in chapter **??**. Weights of the networks are initialized uniformly with $\frac{1}{\sqrt{inputdim}}$ as in Pytorch. Moreover, random matrix B is initialized with the same way to have similar starting as weights.

1.1 Parity Learning with DFA

After having all the theoretical bases, the problem definition, and the experiment details, it is time to test DFA on parity learning problem with BP by using SGD. We can observe the results in 1.1 with 95% confidence interval. It is obvious that DFA performs much better than lazy methods. However, it is behind the BP. The reason is that DFA has an additional task to accomplish, which is aligning with

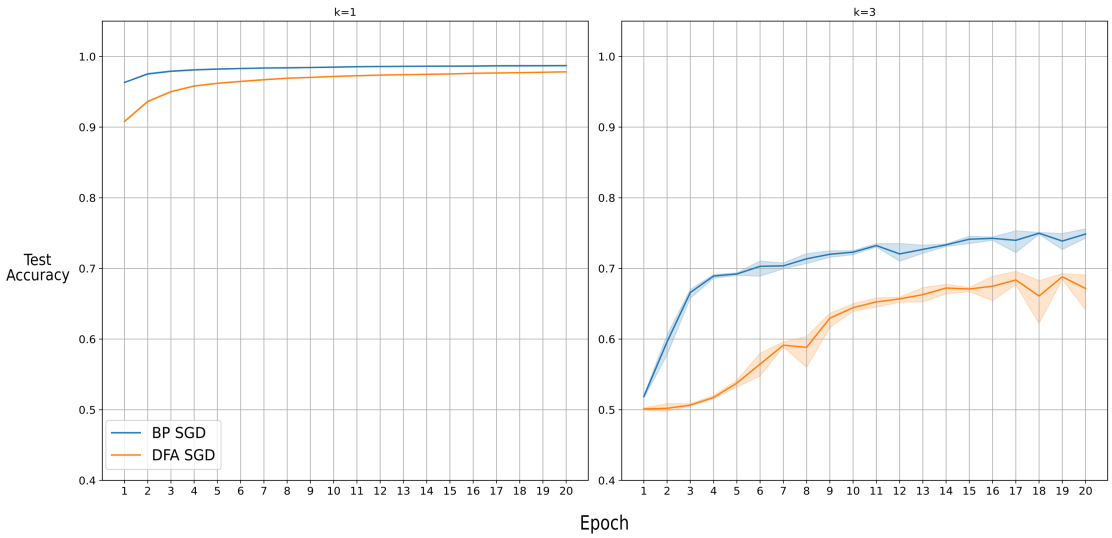

Figure 1.1: BP and DFA on parity learning problem with SGD

BP's teaching signals. This delays the convergence and causes performance lag. The thrilling question is, is there a limit for DFA to reach, and can we get a similar performance as BP by making some changes? For answering the first question, it is better to run DFA for more epochs to see if it can reach a similar performance as BP. Because with longer training, DFA will have time to align and converge. It would be convenient to test DFA with different random matrices to observe any improvement for the second question. Lastly, adaptive methods can be used to have better convergence properties both in BP and DFA.

We trained DFA for 50 epochs with a tuned learning rate to observe if it can reach a similar performance as BP. At the same time, alignment between the random matrix and the transpose of the weight matrix is plotted. This alignment is measured by using the cosine similarity.

From figure 1.2, we can see that DFA can reach a similar performance as BP trained with SGD. This result approves our comments about the additional task DFA has and why it takes longer to achieve the same performance. On the right side of the plot, we can examine the alignment between the random matrix and the transpose of the weight matrix. At the beginning of training, the similarity is low. However, with advancing steps, we can see that alignment becomes higher, similar to the

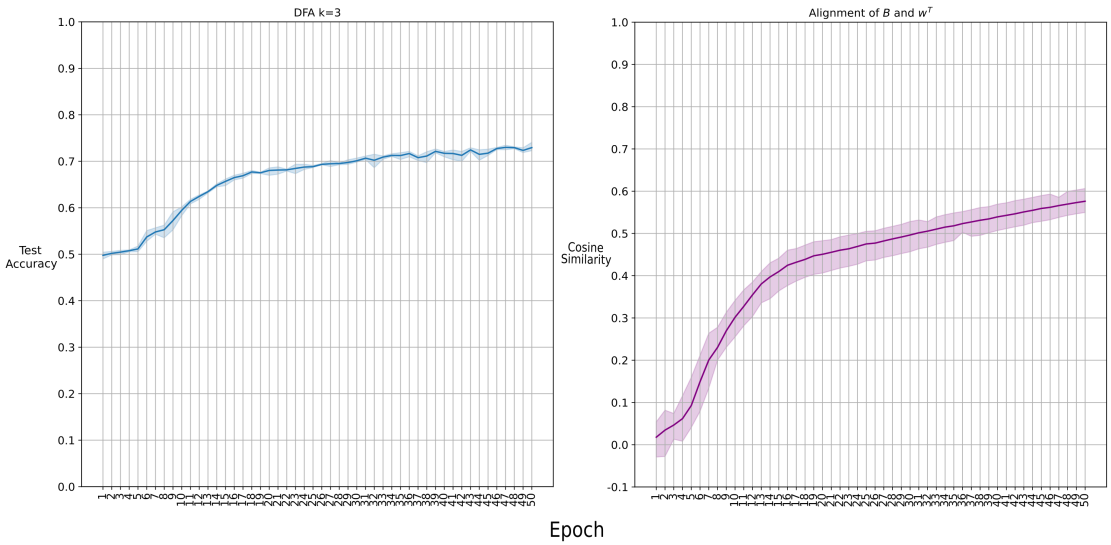

Figure 1.2: BP and DFA on parity learning problem with SGD

performance. It shows that the network aligns with the BP's teaching signals. In other words, the network learns how to learn by using the random matrix. After having a similar performance from DFA with SGD, it is intriguing to test if we can achieve similar performance within the same epoch number. For this purpose, the first improvement attempt will be related to random matrices. Using different random matrices may influence the performance of DFA. Some of them might align better with BP's teaching signals. On the other hand, it is interesting to observe if we can learn with any random matrix.

Among uniform random matrix, three different random matrices are tested. They are initialized as the following: **standard uniform** is default Pytorch initialization that is uniformly distributed from 0 to 1. **Gaussian** is initialized normally with μ and σ are equal to each other that is $\frac{1}{\sqrt{inputdim}}$. Lastly, **standard gaussian** is initialized with $\mu = 0$ and $\sigma = 1$.

From figure 1.3, we can observe that DFA can learn with any random matrices. However, it is essential to specify that learning rates for each random matrix are tuned and drastically different from each other. Apart from standard uniform, the rest of the random matrices achieved similar performances, but they are still behind the BP. However, thanks to these results, we can see that DFA is highly

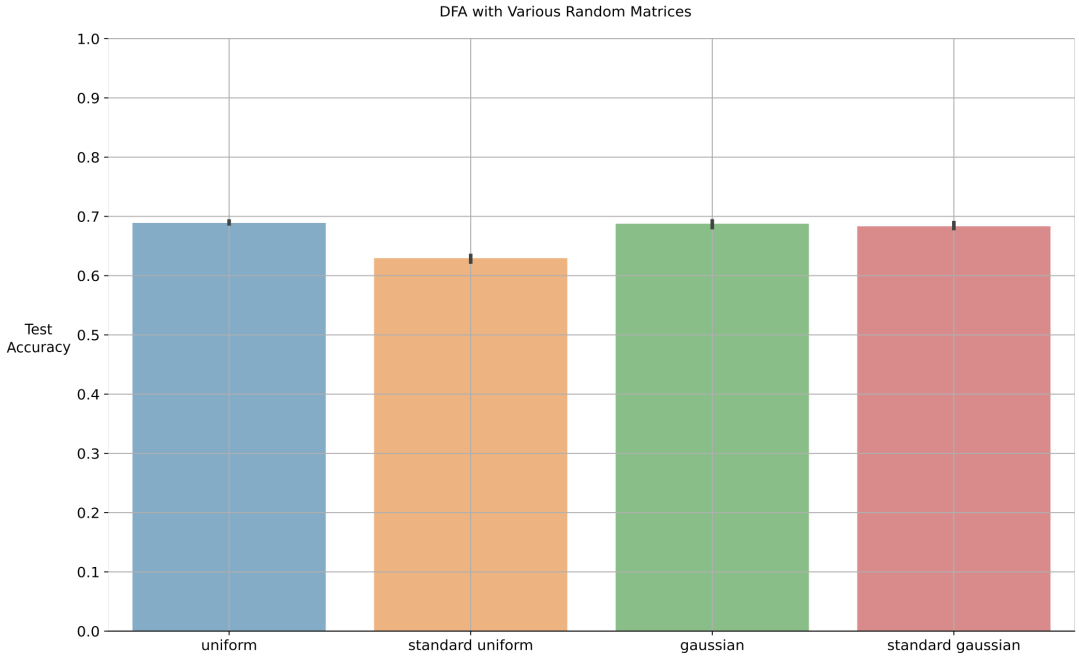

Figure 1.3: DFA on parity learning problem with various random matrices

sensitive to the learning rate. Because during the tuning phase, small learning rates did not converge within the specified epoch number. On the other hand, high learning rates demonstrated overfitting. Therefore, since adaptive methods have better convergence properties, they may increase the performance of DFA as they did in BP. For the rest of DFA experiments, random matrix is initialized uniformly since there is no significant improvement with other initializations.

Following the previous deduction, various adaptive methods are tested on the parity learning problem for BP and DFA. Their learning rates are tuned, as it is explained in the previous chapter. For the experiments they are run three times, and their final test accuracies are plotted. The results are presented in figure **??**.

As it is expected, adaptive methods improves the final test accuracy significantly for both BP and DFA. On average, DFA still behind the BP, but with RMSProp and Adadelta, the gap is much smaller than it is with plain SGD. In other words, we can say that some adaptive methods help DFA more than BP with higher standard deviation. With the last experiment, thanks to adaptive methods we could close

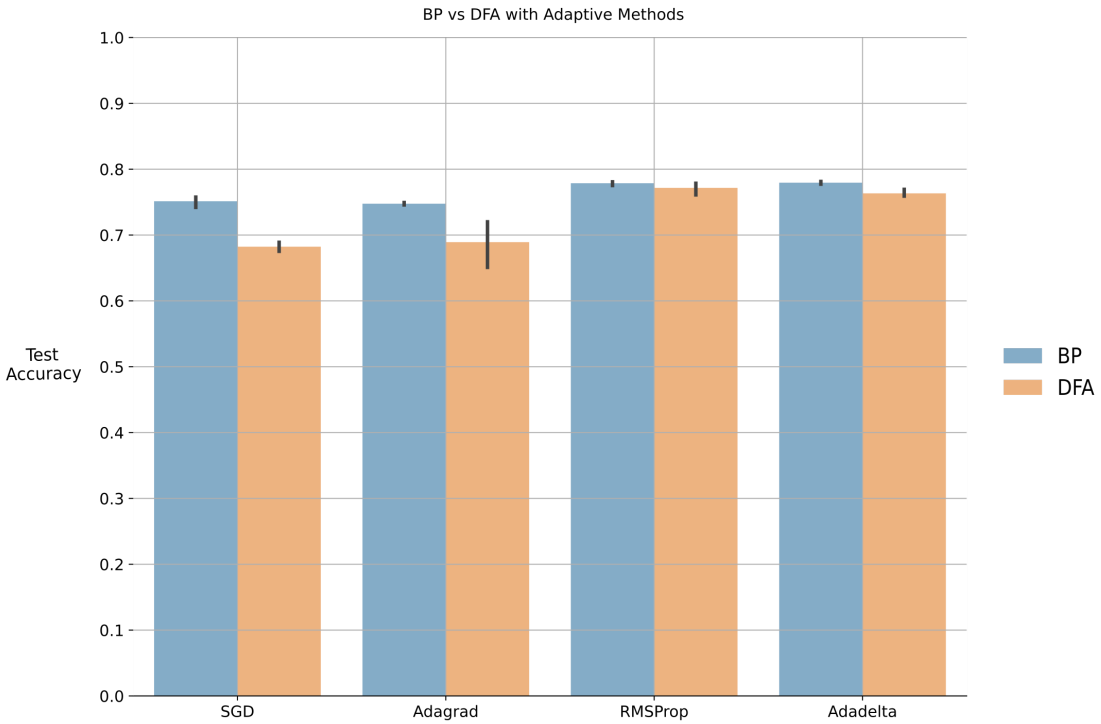

Figure 1.4: DFA on parity learning problem with various random matrices

the gap between BP and DFA on parity learning problem.