An analysis of Deep Neural Networks training methods

Miquel Perelló, E-mail: miquel.perello.nieto@est.fib.upc.edu

Abstract—In this work has been analysed the performance in time and accuracy of Deep Neural Networks. It shows the differences in initialising randomly the hidden units and pre-training in an unsupervised way using the Restricted Boltzmann Machines approach. Also, if there are differences in applying the Hinton's or Bengio's algorithm. Finally if the different number of neurons per layer affects the performance of the mentioned methods.

Keywords—Artificial Neural Network, Deep Neural Network, Restricted Boltzmann Machines, Deep Boltzmann Machines

INTRODUCTION

DEEP Neural Networks has shown to outperform some of the best models on solving classification problems. They rely on the pre-training of the neurons on a unsupervised manner. It does not was until last years when Hinton saw in the Restricted Boltzmann Machines one method to train large Neural Networks in a not exponential time. With this approach we will see how the different architectures of neurons and the initial weights changes the final accuracy, although all the methods rely on the backpropagation algorithm in the main phase.

1 PRE-TRAINING WITH RBM

The use of RBM has demonstrated a huge improve in lots of applications during last years. Here we will see the differences of using and not the phase of pre-training with a Restricted Boltzman Machine algorithm.

In this section we will focus only on that part of the problem, but we will use different architectures and RBM algorithms. But our focus is only to see if there is an improve when we use it.

1.1 Time

Regarding to the computational time, it is obvious that pre-training phase will add some time to assign the initial neurons weights. But is this phase sufficiently long to enlarge the overall training? In the picture 1 we can see that random assignation is instantaneous. On the other side, pre-training with Hilton's and/or Bengio's algorithm takes from twenty to four hundred seconds (depending on the configuration).

Once the pre-training is done, in the training phase the first half has been executed with one hundred epochs, and last executions with fifty epoch. That is because the spend time was growing so much. In this comparison this is not a problem because all the executions has been paired. Here the computation times are not so different (see figure 2), except one section corresponding to the executions interval $[34, 48]$. This times are expected to be equal because the backpropagation algorithm must not be influenced by the initial weights but since these results are not cross-validated we can take the total average to take the conclusions.

Finally we can see a summary of the time spend in pre-training, training and the overall process in the figure 3. We can see in perspective that pre-training time is not the biggest problem regarding to computational time. It seems to be only outliers for the average examples. It will be convenient to create a bigger test, but it is out of the scope of this report.

1.2 Accuracy

We have seen that the amount of time seems not to be significant, and here we will see the

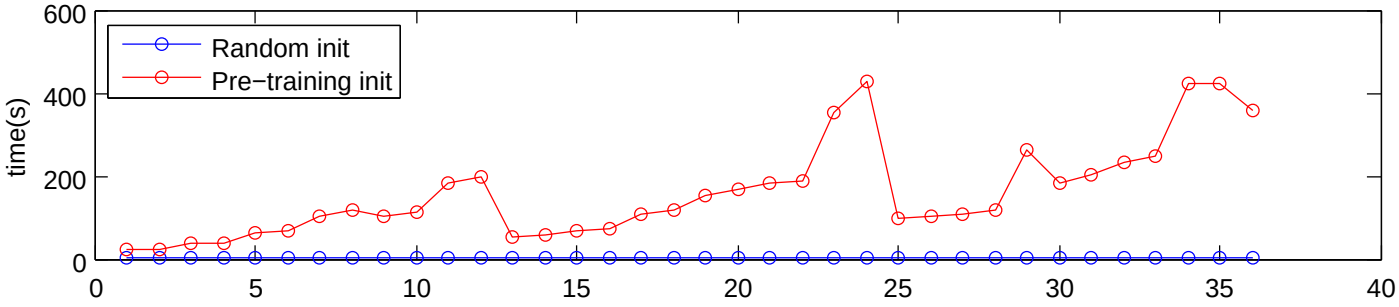

Fig. 1. **Pre-training Time** - Time in seconds of random initialisation and RBM initialisation

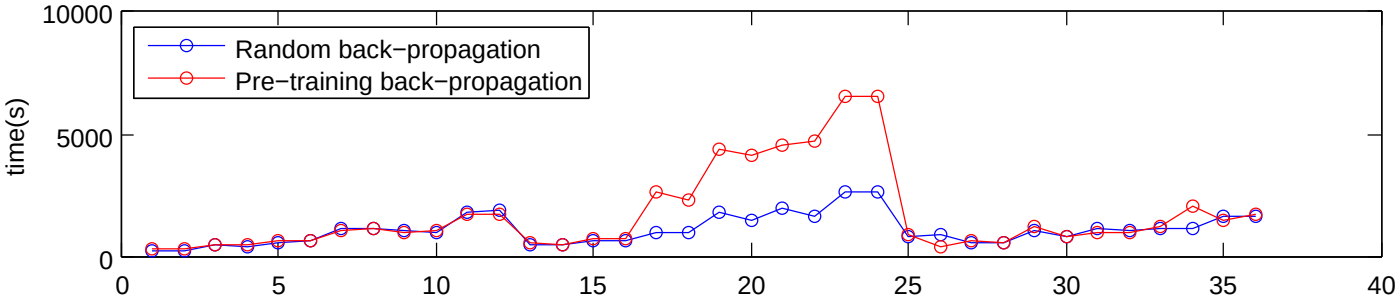

Fig. 2. **Backpropagation Time** -Time in seconds of backpropagation with each initialisation

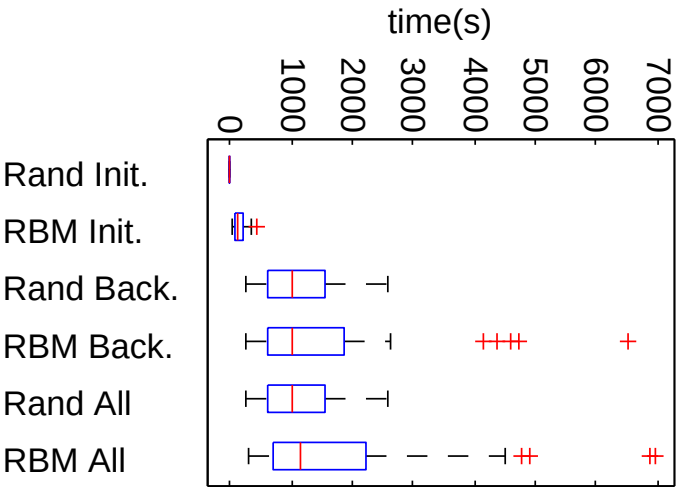

Fig. 3. **Training Time** - Summary of time spent with and without pre-training in each phase.

accuracy that each approach is able to reach.

First of all, we can see in the figure 4 the initial accuracy once the initial weights are selected. Although the pre-training phase with RBM is not so long, the results in some cases are so god. The three depressions of the plot, corresponds to the examples with thirty-two neurons in the middle layer, this seems to affect in the RBM algorithm. On the other side like is expected, the random initial weights are equally distributed.

Once the network is pre-trained, in the back-propagation phase in some examples selecting random weights performs better than pre-training (see figure 5). Concretely, the two hills where RBM performs better is when the number of neurons per layer is great or equal to five hundred twelve. On that cases the improvement is over five percent. But with less neurons the random selection is better in all the cases.

With this information we can only see that pre-training with RBM algorithms seems to perform better for large neural networks, but in this examples on average both algorithms performs similar on average. We can see that RBM makes the accuracies more variables o the other side selecting randomly the initial weights creates more homogeneous solutions (see figure 6).

2 RBM TRAINING VERSION

In this section we will analyse if there is some differences between Hinton's and Bengio's algorithm. To study both algorithms the executions where the pre-training was not selected have been removed. For that reason from the

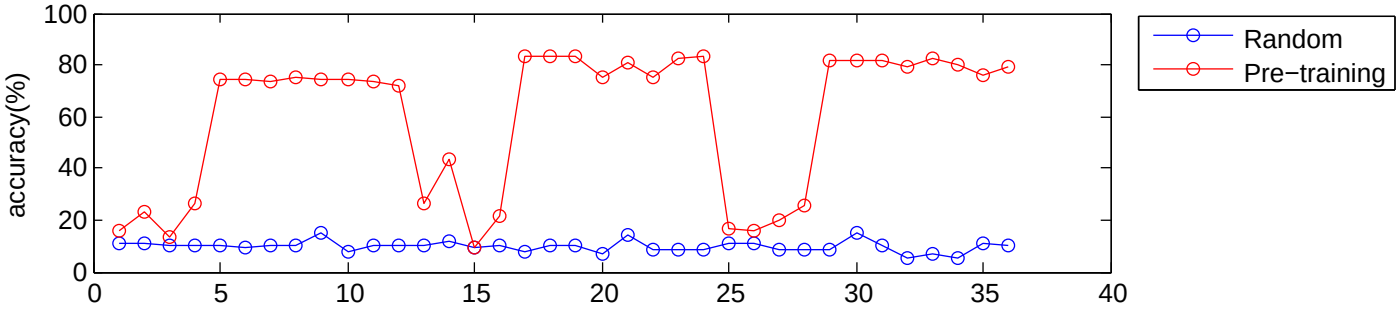

Fig. 4. **Initial Accuracy Test** - Accuracy after initial weights have been selected

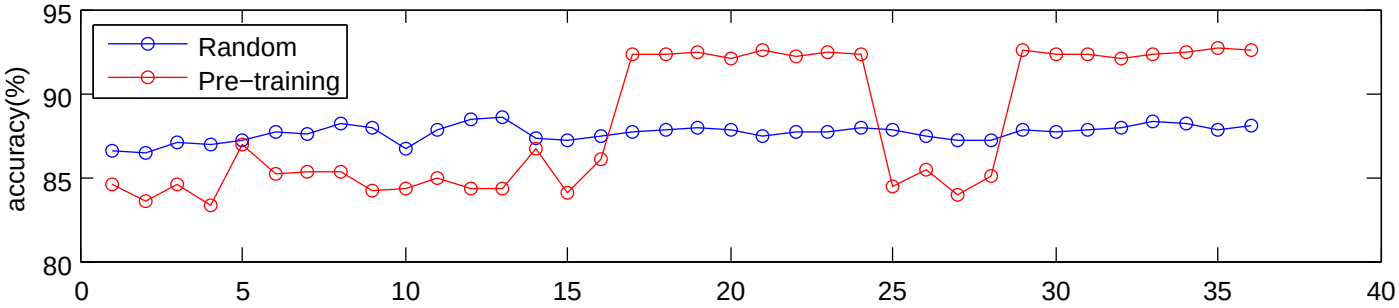

Fig. 5. **Final Accuracy Test** - Accuracy in the test data for both methods of initialisation

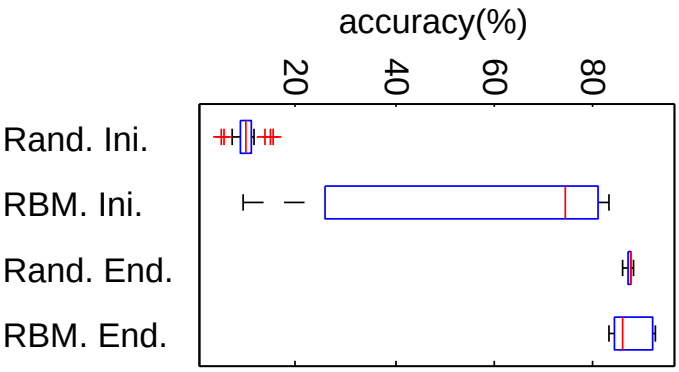

Fig. 6. **Pre-training accuracy** - Summary of the accuracy results with and without pre-training phase

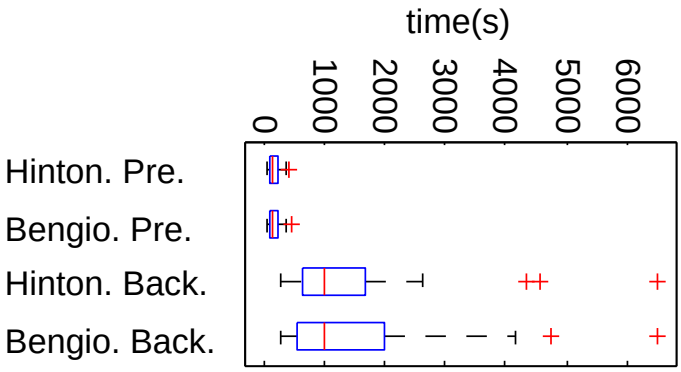

Fig. 7. **RBM algorithms time** - Summary of the time for Hilton's and Bengio's algorithm

original seventy two executed examples we will see only the results of the half thirty six.

Execution times for both algorithms are so similar, there is not evidence to choose one as the best (see figure 7). On both cases the pre-training time is about one hundred seconds, while the training phase spends sixteen minutes ($\approx 1000s$).

Regarding to the accuracy, they performs equal in all the situations. It is not possible to select one or another in the executions performed in this report. It can be seen in figures 8 with the final accuracies and in the figure 9 with a summary of pre-training and final accuracies.

3 Architectures

The number of layers and neurons in an Artificial Neural Network is always something that needs a lot of effort to select. In this case we do not want to search the best model to fit our dataset. Instead, we only want to see if there is some differences regarding of the architecture,

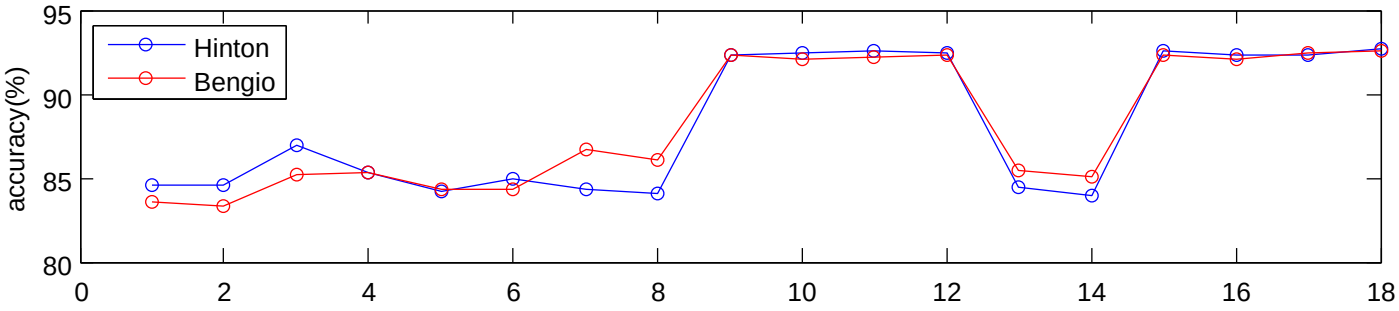

Fig. 8. **RBF test accuracy** Differences in Hilton's and Bengio's rbm version

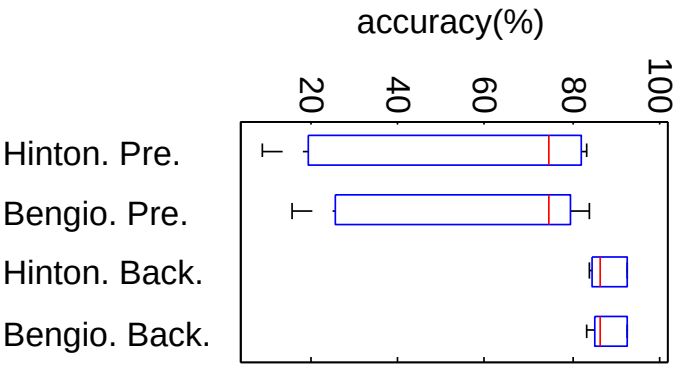

Fig. 9. **RBM algorithms accuracy** - Summary of the accuracy for Hilton's and Bengio's algorithm

but related to the pre-training method and algorithm.

The different architectures to be tested have been all the combinations of these layers and neurons:

1) First hidden layer : [32, 512, 1024]
2) Second hidden layer: [32, 512, 1024]
3) Third hidden layer : [1024, 2048]

After all the executions we can see that the time grows with the number of neurons (see figure 10). There is an interval that seems to grow in time. It can be shown in the appendix 4 from the execution thirty four to forty eight. Just after that part, the times fall once again. The first reason is that the second layer starts with thirty two neurons again, but additionally the number of epoch was decreased to fifty for the large amount of time it was taking.

On these examples the accuracy grows in two sections, these are when the minimum number of neurons per layer is bigger than thirty. And these growth is only for the executions that made the pre-training. In the figure 11 there

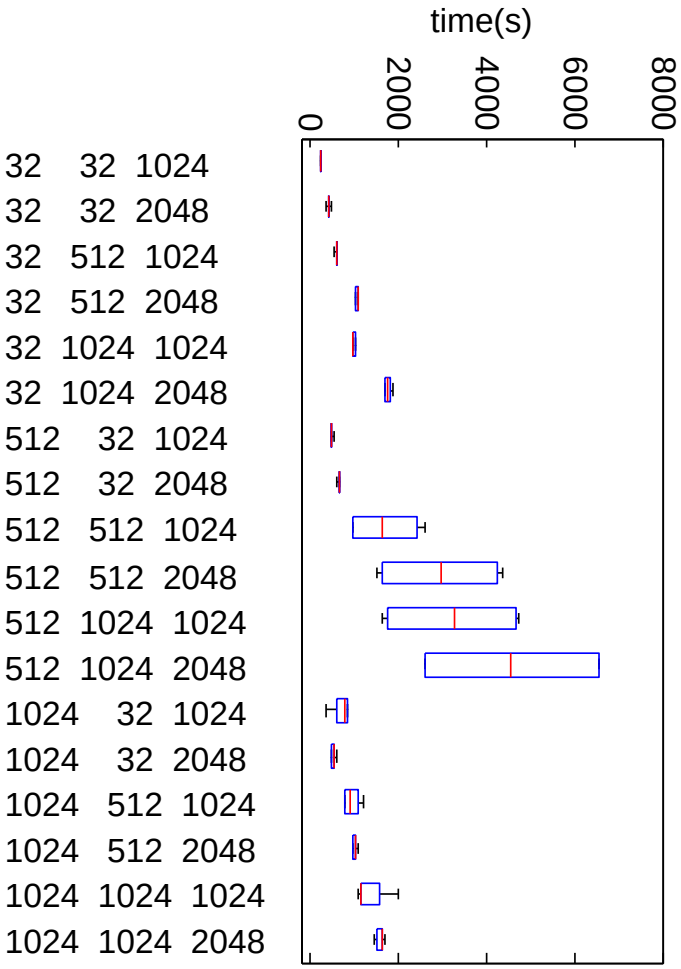

Fig. 10. **Time of all architectures** - all execution time for each architecture

are all the executions in boxplots (both with random and RBM pre-training).

4 Conclusion

We have seen that Artificial Neural Networks are able to outperform the best results if the number of neurons and hidden layer grows.

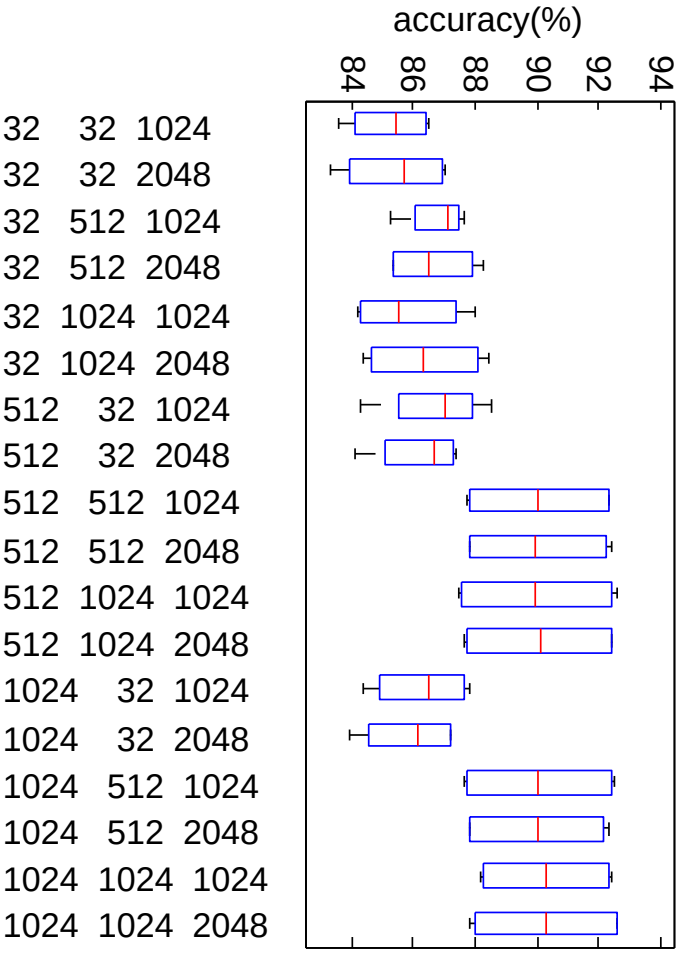

Fig. 11. **Accuracy of all architectures** - all accuracy the results for each architecture

The pre-training and training time grows with the number of neurons, but there is a trade-off where the time and the accuracy can be optimal.

Regarding to the Hilton's and Bengio's algorithm, we can not extract any conclusion in this work, since there is not evidence on any of them.

REFERENCES

[1] Salakhutdinov, Ruslan, and Geoffrey E. Hinton. "Deep boltzmann machines." Proceedings of the international conference on artificial intelligence and statistics. Vol. 5. No. 2. Cambridge, MA: MIT Press, 2009.

APPENDIX A

These are some of the results that has been used to understand all explained in this work. It is here for your convenience in looking most in detail.

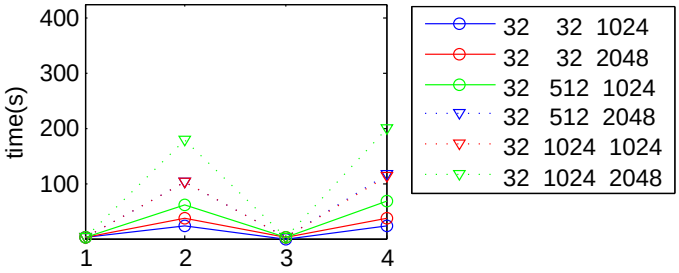

Fig. 12. **Time 32 neurons first layer**

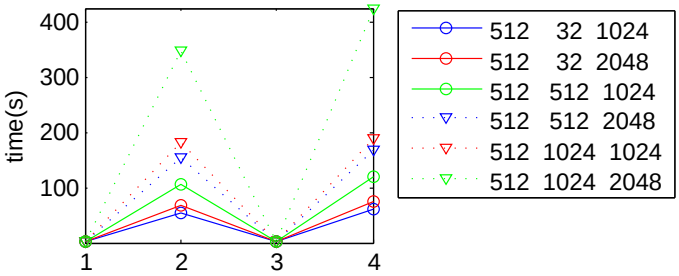

Fig. 13. **Time 512 neurons first layer**

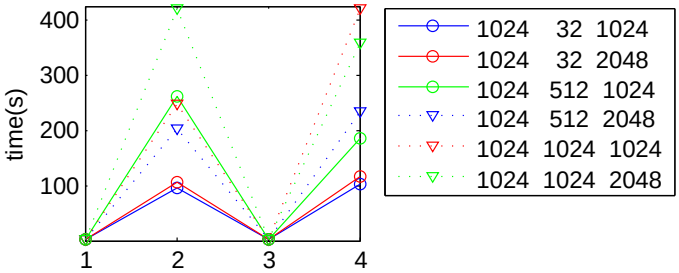

Fig. 14. **Time 1024 neurons first layer**

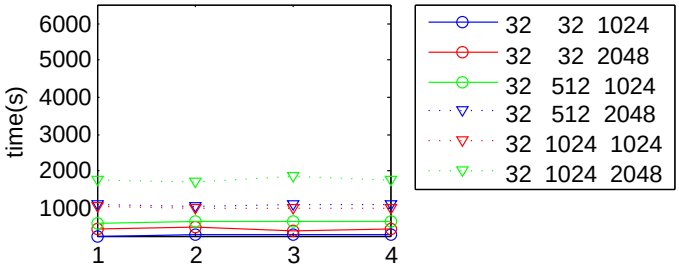

Fig. 15. **Time 32 neurons first layer**

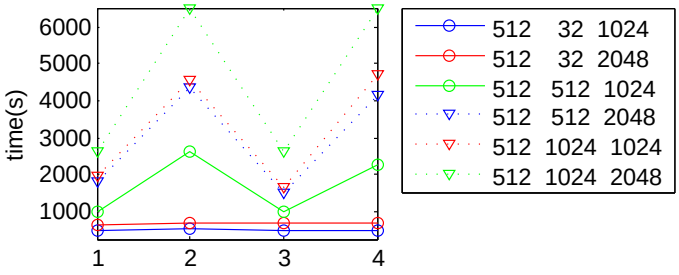

Fig. 16. **Time 512 neurons first layer**

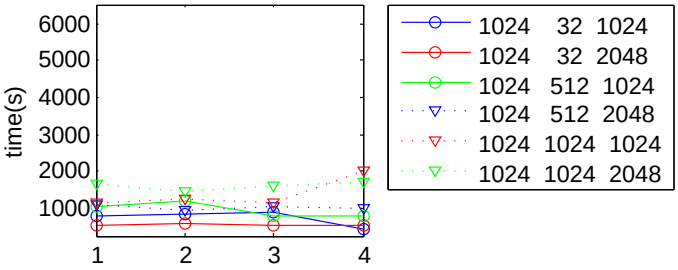

Fig. 17. **Time 1024 neurons first layer**

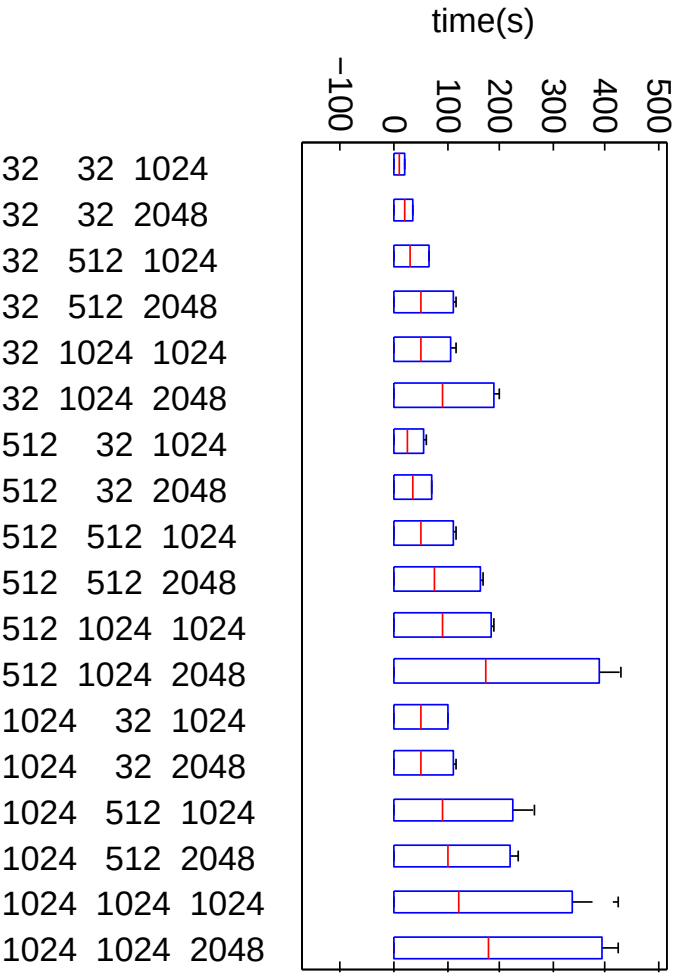

Fig. 18. **Summary of Initalization times**

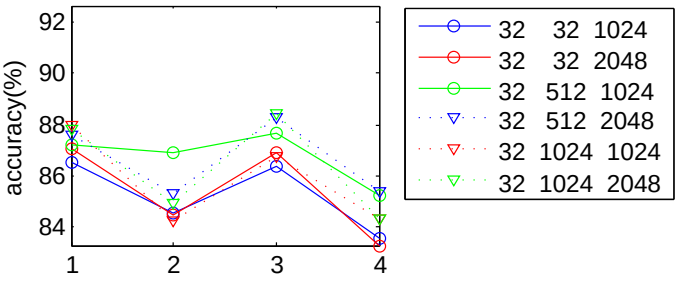

Fig. 19. **Accuracy 32 neurons first layer**

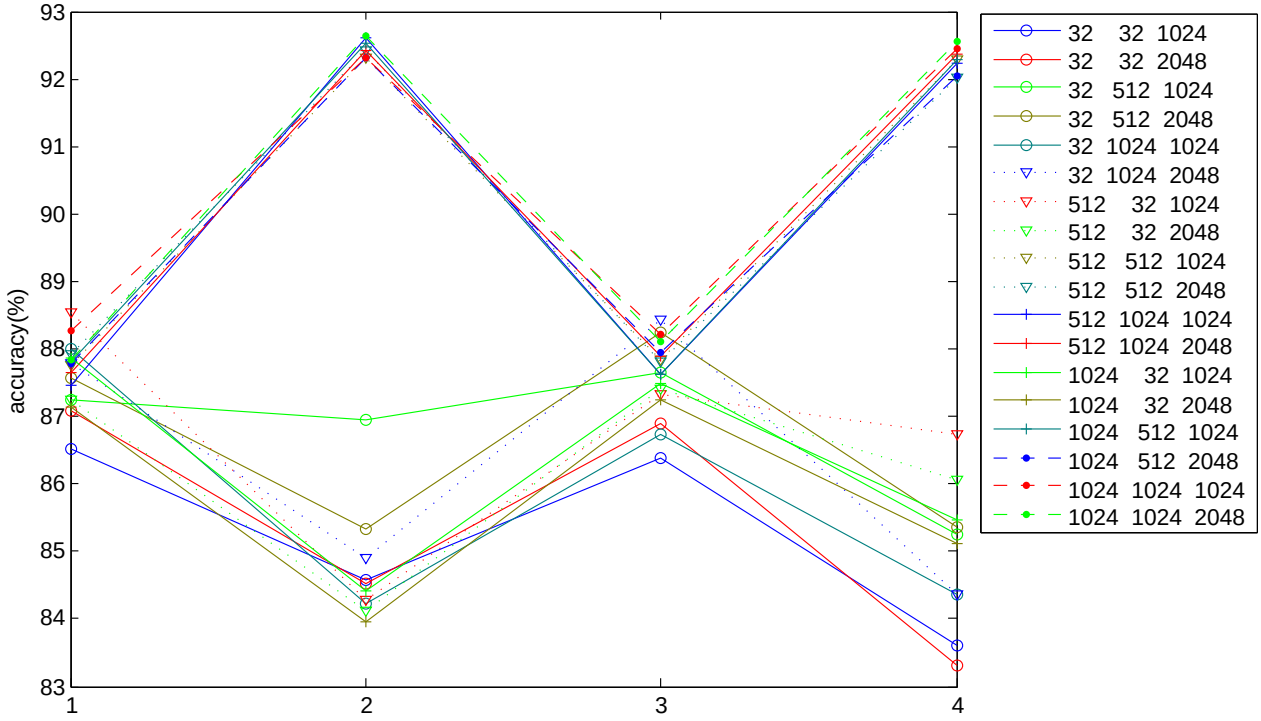

Fig. 22. **Accuracy 32 neurons first layer** -

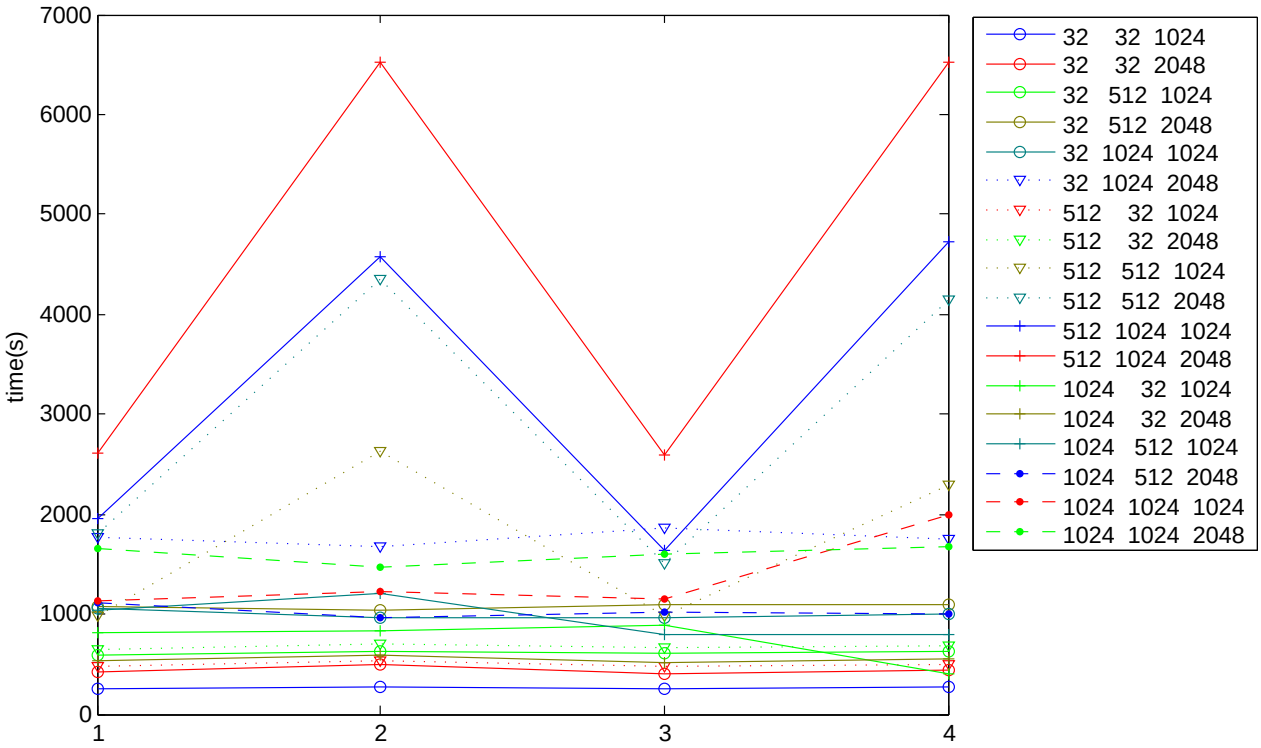

Fig. 23. **Accuracy 32 neurons first layer** -

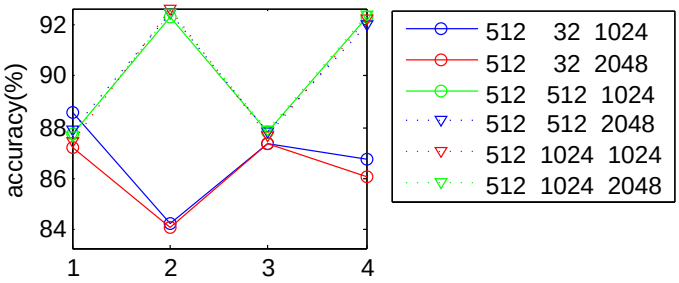

Fig. 20. **Accuracy 512 neurons first layer**

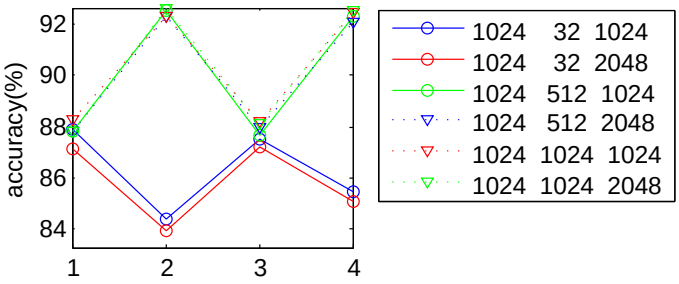

Fig. 21. **Accuracy 1024 neurons first layer**

APPENDIX B

This table contains all the execution results where: *hid1*, *pen1* and *pen2* are the number of hidden units in the first, second and third layer respectively; *rbm* is one for Hinton's version and two for Bengio's version; *PR* indicates with zero that initial weights are randomly selected and one when they are pre-trained; *epoch* is the number of epochs in the back-propagation phase; *numtr* and *numte* are the number of training and test data; *tinit* and *tback* is the time spend in initialising the weights and in the back-propagation phase respectively; *acctr* and *accte* are the maximum accuracy obtained in the test and training phases.

TABLE 1: Summary of all executions

N	hid1	pen1	pen2	v	PR	epo	str	ste	tinit	tback	iactr	iacte	eactr	eacte
1	32	32	1024	1	0	100	1000	10000	0	245	11	11	100	87
2	32	32	1024	1	1	100	1000	10000	22	279	16	16	97	85
3	32	32	1024	2	0	100	1000	10000	0	245	11	11	100	86
4	32	32	1024	2	1	100	1000	10000	23	268	24	23	100	84
5	32	32	2048	1	0	100	1000	10000	0	430	10	10	97	87
6	32	32	2048	1	1	100	1000	10000	36	495	12	13	100	85
7	32	32	2048	2	0	100	1000	10000	0	394	10	10	100	87
8	32	32	2048	2	1	100	1000	10000	37	440	27	26	96	83
9	32	512	1024	1	0	100	1000	10000	0	582	11	10	97	87
10	32	512	1024	1	1	100	1000	10000	61	628	76	75	100	87
11	32	512	1024	2	0	100	1000	10000	0	611	10	10	100	88
12	32	512	1024	2	1	100	1000	10000	68	632	77	74	100	85
13	32	512	2048	1	0	100	1000	10000	0	1085	10	10	100	88
14	32	512	2048	1	1	100	1000	10000	103	1044	75	73	100	85
15	32	512	2048	2	0	100	1000	10000	0	1090	9	11	100	88
16	32	512	2048	2	1	100	1000	10000	117	1087	78	75	100	85
17	32	1024	1024	1	0	100	1000	10000	0	1062	15	15	100	88
18	32	1024	1024	1	1	100	1000	10000	102	971	78	75	100	84
19	32	1024	1024	2	0	100	1000	10000	0	961	7	8	95	87
20	32	1024	1024	2	1	100	1000	10000	115	1010	77	74	100	84
21	32	1024	2048	1	0	100	1000	10000	0	1775	10	10	100	88
22	32	1024	2048	1	1	100	1000	10000	181	1678	76	73	100	85
23	32	1024	2048	2	0	100	1000	10000	0	1868	11	10	100	88
24	32	1024	2048	2	1	100	1000	10000	199	1746	74	72	100	84
25	512	32	1024	1	0	100	1000	10000	0	483	10	10	100	89
26	512	32	1024	1	1	100	1000	10000	55	529	27	26	99	84
27	512	32	1024	2	0	100	1000	10000	0	482	12	12	100	87
28	512	32	1024	2	1	100	1000	10000	60	499	44	44	100	87
29	512	32	2048	1	0	100	1000	10000	0	640	9	9	100	87
30	512	32	2048	1	1	100	1000	10000	67	695	8	9	95	84
31	512	32	2048	2	0	100	1000	10000	0	662	10	10	100	87
32	512	32	2048	2	1	100	1000	10000	74	679	21	22	99	86
33	512	512	1024	1	0	100	1000	10000	0	989	7	7	100	88
34	512	512	1024	1	1	100	1000	10000	106	2627	83	83	100	92
35	512	512	1024	2	0	100	1000	10000	0	981	10	10	100	88
36	512	512	1024	2	1	100	1000	10000	120	2286	86	84	100	92
37	512	512	2048	1	0	100	1000	10000	0	1800	10	10	100	88
38	512	512	2048	1	1	100	1000	10000	154	4357	84	83	100	92

39	512	512	2048	2	0	100	1000	10000	0	1499	7	7	100	88
40	512	512	2048	2	1	100	1000	10000	167	4154	78	76	100	92
41	512	1024	1024	1	0	100	1000	10000	0	1959	14	14	100	87
42	512	1024	1024	1	1	100	1000	10000	182	4581	81	81	100	93
43	512	1024	1024	2	0	100	1000	10000	0	1632	10	8	100	88
44	512	1024	1024	2	1	100	1000	10000	188	4726	77	75	100	92
45	512	1024	2048	1	0	100	1000	10000	0	2603	9	9	100	88
46	512	1024	2048	1	1	100	1000	10000	350	6529	84	82	100	92
47	512	1024	2048	2	0	100	1000	10000	0	2597	9	9	100	88
48	512	1024	2048	2	1	100	1000	10000	428	6522	85	83	100	92
49	1024	32	1024	1	0	100	1000	10000	0	806	12	11	100	88
50	1024	32	1024	1	1	100	1000	10000	97	834	18	17	97	84
51	1024	32	1024	2	0	100	1000	10000	0	888	11	11	99	87
52	1024	32	1024	2	1	50	1000	10000	104	402	16	16	98	85
53	1024	32	2048	1	0	50	1000	10000	0	533	8	8	97	87
54	1024	32	2048	1	1	50	1000	10000	106	594	21	20	91	84
55	1024	32	2048	2	0	50	1000	10000	0	515	9	9	97	87
56	1024	32	2048	2	1	50	1000	10000	117	551	25	26	93	85
57	1024	512	1024	1	0	50	1000	10000	0	1040	9	9	100	88
58	1024	512	1024	1	1	50	1000	10000	263	1208	82	82	100	93
59	1024	512	1024	2	0	50	1000	10000	0	801	16	15	98	88
60	1024	512	1024	2	1	50	1000	10000	185	803	83	82	100	92
61	1024	512	2048	1	0	50	1000	10000	0	1108	10	10	100	88
62	1024	512	2048	1	1	50	1000	10000	205	959	81	81	100	92
63	1024	512	2048	2	0	50	1000	10000	0	1026	5	5	96	88
64	1024	512	2048	2	1	50	1000	10000	233	999	82	80	100	92
65	1024	1024	1024	1	0	50	1000	10000	0	1127	7	7	98	88
66	1024	1024	1024	1	1	50	1000	10000	248	1226	83	82	100	92
67	1024	1024	1024	2	0	50	1000	10000	0	1156	6	5	100	88
68	1024	1024	1024	2	1	50	1000	10000	423	2002	82	80	100	92
69	1024	1024	2048	1	0	50	1000	10000	0	1662	12	11	100	88
70	1024	1024	2048	1	1	50	1000	10000	421	1467	76	76	100	93
71	1024	1024	2048	2	0	50	1000	10000	0	1602	10	10	100	88
72	1024	1024	2048	2	1	50	1000	10000	359	1684	81	79	100	93