Deep Message Passing

Around the middle of the XVII century, a young man called Baruch Spinoza was dealing with a problem tougher than yours: finding the essence of God Himself.

CONTENTS

I. INTRODUCTION

II. LEARNING THROUGH MESSAGE PASSING

TODO ENTIRE SECTION.

In this deep inference problem, we assume that a signal with prior P^{in} is fed to a deep feedforward networks with $L+1$ layers of weights $\boldsymbol{W}^\ell \in \mathbb{R}^{N_{\ell+1}\times N_\ell}, \ell = 0,\dots,L$ and biases $\boldsymbol{b}^\ell \in \mathbb{R}^{N_{\ell+1}}$. The signal is propagated through stochastic neuron layers described by probability distributions P^ℓ conditioned on the preactivations, therefore we have the following Markov chain:

$$\boldsymbol{x}^0 \sim P^{\text{in}} \tag{II.1}$$

$$\boldsymbol{x}^{\ell+1} \sim P^{\ell+1}\left(\bullet \,|\, \boldsymbol{W}^\ell \boldsymbol{x}^\ell + \boldsymbol{b}^\ell\right) \qquad \ell = 0,\dots,L \tag{II.2}$$

Only $\boldsymbol{y} = \boldsymbol{x}^{L+1}$ is observed, and the task is to reconstruct the original signal $\boldsymbol{x}^0$. The posterior distribution $p(\boldsymbol{x}^{0:L}) = P(\boldsymbol{x}^{0:L} \,|\, \boldsymbol{x}^{L+1} = \boldsymbol{y})$ reads

$$p(\boldsymbol{x}^{0:L}) \propto \prod_{\ell=0}^{L}\prod_{k=1}^{N_{\ell+1}} P_k^{\ell+1}\left(x_k^{\ell+1} \;\middle|\; \sum_{i=1}^{N_\ell} W_{ki}^\ell x_i^\ell\right) \prod_{k=1}^{N_0} P_k^{\text{in}}(x_k^0), \tag{II.3}$$

Typical channels are given by deterministic elementwise activation function $f_\ell(z)$ (e.g. $f_\ell(z) = \text{sign}(x)$ or $f_\ell(z) = \text{relu}(z) = \max(0,z)$), combined with Gaussian additive pre-activation noise with variance σ^2. In such cases we have

$$P_k^\ell(x \,|\, z) = \int D\xi\, \delta\left(x - f_\ell(z + \sigma_k^\ell\, \xi)\right)$$

We also call $\alpha_\ell = N_{\ell+1}/N_\ell$ the layer *expansion ratio.*

III. BELIEF PROPAGATION

A. BP updates

The AMP equations have been derived from the BP equation in the Appendix. First we introduce the neuron scalar entropy functions:

$$\varphi_k^0(B,A) = \log \int \mathrm{d}x\; e^{-\frac{1}{2}A^2x^2 + Bx}\, P_k^{\text{in}}(x) \tag{III.1}$$

$$\varphi_k^\ell(B,A,\omega,V) = \log \int \mathrm{d}x\,\mathrm{d}z\; e^{-\frac{1}{2}A^2x^2 + Bx}\, P_k^\ell\,(x|z)\, e^{-\frac{(\omega-z)^2}{2V}} \qquad \ell = 1,\dots,L \tag{III.2}$$

$$\varphi_k^{L+1}(\omega,V,y) = \log \int \mathrm{d}z\; P_k^{L+1}\,(y|z)\, e^{-\frac{(\omega-z)^2}{2V}} \tag{III.3}$$

$$\psi_{ki}^\ell(H,G) = \log \int \mathrm{d}w\; e^{-\frac{1}{2}G^2w^2 + Hw}\, P_{ki}^\ell(w) \tag{III.4}$$

For convenience define $\varphi_i^{0,t} = \varphi_i^0\left(B_i^{0,t}, A_i^{0,t}\right)$ and $\varphi_i^{\ell,t} = \varphi_i^\ell\left(B_i^{\ell,t}, A_i^{\ell,t}, \omega_i^{\ell-1,t}, V_i^{\ell-1,t}\right)$ and $\varphi_i^{L+1,t} = \varphi_i^{L+1}\left(\omega_i^{L,t}, V_i^{L,t}, y_i\right)$.

Then, we can decompose the BP update rules in a forward and a backward step.

Forward pass. As the initial condition for the iterations, we set to zero the following quantities: $B_i^{\ell,t=0} = 0, A_i^{\ell,t=0} = 0$ and $g_k^{\ell,t=0} = 0$. The following iterations hold at time $t \geq 1$. In the FORWARD pass, starting from $\ell = 0$ and up to $\ell = L$, we have

$$\hat{x}^{\ell,t}_{ia\to k} = \partial_B \varphi^{\ell}_{ia\to k}\left(B^{\ell,t-1}_{ia\to k}, A^{\ell,t-1}_{ia}, \omega^{\ell-1,t}_{ia}, V^{\ell-1,t}_{ia}\right) \tag{III.5}$$

$$\Delta^{\ell+1,t}_{ia\to k} = \partial^2_B \varphi^{\ell+1,t}_{ia\to k} \tag{III.6}$$

$$m^{\ell,t}_{ki\to a} = \partial_H \psi^{\ell}_{ki}(H^{t-1}_{ki\to a}, G^{t-1}_{ki}) \tag{III.7}$$

$$\sigma^{\ell,t}_{ki\to a} = \partial^2_H \psi^{\ell}_{ki}(H^{t-1}_{ki\to a}, G^{t-1}_{ki}) \tag{III.8}$$

$$V^{\ell,t}_{ka} = \sum_i \left(\left(m^{\ell,t}_{ki\to a}\right)^2 \Delta^{\ell,t}_{ia\to k} + \Sigma^{\ell,t}_{ki\to a}(\hat{x}^{\ell,t}_{ia\to k})^2 + \sigma^{\ell,t}_{ki\to a}\Delta^{\ell,t}_{ia\to k} \right) \tag{III.9}$$

$$\omega^{\ell,t}_{ka\to i} = \sum_{i'\neq i} m^{\ell,t}_{ki\to a}\hat{x}^{\ell,t}_{ia\to k} \tag{III.10}$$

Here V^{ℓ} and ω^{ℓ} are computed as a function of the previous layer values $V^{\ell-1}$ and $\omega^{\ell-1}$.

Backward pass. In the BACKWARD sweep, starting from $\ell = L$ and down to $\ell = 0$, we have

$$g^{\ell,t}_{ka\to i} = \partial_\omega \varphi^{\ell+1}_{ka\to i}\left(B^{\ell+1,t}_{ka}, A^{\ell+1,t}_{ka}, \omega^{\ell,t}_{ka\to i}, V^{\ell,t}_{ka}\right) \tag{III.11}$$

$$\Gamma^{\ell,t}_{ka\to i} = -\partial^2_\omega \varphi^{\ell+1,t}_{ka\to i} \tag{III.12}$$

$$A^{\ell,t}_{ia} = \sum_k \left((m^{\ell,t}_{ki\to a})^2 + \sigma^{\ell,t}_{ki\to a} \right) \Gamma^{\ell,t}_{ka\to i} - \sigma^{\ell,t}_{ki\to a}\left(g^{\ell,t}_{ka\to i}\right)^2 \tag{III.13}$$

$$B^{\ell,t}_{ia\to k} = \sum_{k'\neq k} m^{\ell}_{k'i\to a} g^{\ell,t}_{k'a\to i} \tag{III.14}$$

$$G^{\ell,t}_{ki} = \sum_a \left((\hat{x}^{\ell,t}_{ia\to k})^2 + \Delta_{ia\to k} \right) \Gamma^{\ell,t}_{ka\to i} - \Delta_{ia\to k}\left(g^{\ell,t}_{ka\to i}\right)^2 \tag{III.15}$$

$$H_{ki\to a} = \sum_{a'\neq a} \hat{x}^{\ell,t}_{ia'\to k} g^{\ell,t}_{ka'\to i} \tag{III.16}$$

Notice that A^{ℓ} and B^{ℓ} are computed as as a function of the $A^{\ell+1}, B^{\ell+1}$ of the layer above, with the initial condition given by the output $\boldsymbol{x}^{L+1} = \boldsymbol{y}$ on the top layer.

B. Approximate Message Passing

Forward pass. As the initial condition for the iterations, we set to zero the following quantities: $B^{\ell,t=0}_i = 0, A^{\ell,t=0}_i = 0$ and $g^{\ell,t=0}_k = 0$. The following iterations hold at time $t \geq 1$. In the FORWARD pass, starting from $\ell = 0$ and up to $\ell = L$, we have

$$\hat{x}^{\ell,t}_{ia} = \partial_B \varphi^{\ell,t^-}_{ia} \tag{III.17}$$

$$\Delta^{\ell,t}_{ia} = \partial^2_B \varphi^{\ell,t^-}_{ia} \tag{III.18}$$

$$m^{\ell,t}_{ki} = \partial_H \psi^{\ell,t^-}_{ki} \tag{III.19}$$

$$\sigma^{\ell,t}_{ki} = \partial^2_H \psi^{\ell,t^-}_{ki} \tag{III.20}$$

$$V^{\ell,t}_{ka} = \sum_i \left(\left(m^{\ell,t}_{ki}\right)^2 \Delta^{\ell,t}_{ia} + \sigma^{\ell,t}_{ki}(\hat{x}^{\ell,t}_{ia})^2 + \sigma^{\ell,t}_{ki}\Delta^{\ell,t}_{ia} \right) \tag{III.21}$$

$$\omega^{\ell,t}_{ka} = \sum_i m^{\ell,t}_{ki}\hat{x}^{\ell,t}_{ia} + TODO : onsagsize(x)er \tag{III.22}$$

Here V^{ℓ} and ω^{ℓ} are computed as a function of the previous layer values $V^{\ell-1}$ and $\omega^{\ell-1}$.

Backward pass. In the BACKWARD sweep, starting from $\ell = L$ and up to $\ell = 0$, we have

$$g_{ka}^{\ell,t} = \partial_\omega \varphi_{ka}^{\ell+1,t} \tag{III.23}$$
$$\Gamma_{ka}^{\ell,t} = -\partial_\omega^2 \varphi_{ka}^{\ell+1,t} \tag{III.24}$$
$$A_{ia}^{\ell,t} = \sum_k \left((m_{ki}^{\ell,t})^2 + \sigma_{ki}^{\ell,t} \right) \Gamma_{ka}^{\ell,t} - \sigma_{ki}^{\ell,t} \left(g_{ka}^{\ell,t} \right)^2 \tag{III.25}$$
$$B_{ia}^{\ell,t} = \sum_k m_{ki}^{\ell} g_{ka}^{\ell,t} + TODO : onsager \tag{III.26}$$
$$G_{ki}^{\ell,t} = \sum_a \left((\hat{x}_{ia}^{\ell,t})^2 + \Delta_{ia} \right) \Gamma_{ka}^{\ell,t} - \Delta_{ia} \left(g_{ka}^{\ell,t} \right)^2 \tag{III.27}$$
$$H_{ki} = \sum_a \hat{x}_{ia}^{\ell,t} g_{ka}^{\ell,t} + TODO : onsager \tag{III.28}$$

Notice that A^ℓ and B^ℓ are computed as as a function of the $A^{\ell+1}, B^{\ell+1}$ of the layer above, with the initial condition given by the output $\boldsymbol{x}^{L+1} = \boldsymbol{y}$ on the top layer.

IV. ARGMAX LAYER

In order to perform multi-class classification, we have perform an argmax operation. Call z_k, for $k = 1, \dots, K$, the Gaussian random variables output of the last layer of the network in correspondence of some input $\boldsymbol{x}$. Assuming the correct label is class k, the effective partition function corresponding to the output constraint reads

$$Z_{k^*} = \int \prod_k dz_k \, \mathcal{N}(z_k; \omega_k, V_k) \prod_{k \neq k^*} \theta(z_{k^*} - z_k) \tag{IV.1}$$
$$= \int dz_{k^*} \, \mathcal{N}(z_{k^*}; \omega_{k^*}, V_{k^*}) \prod_{k \neq k^*} H\left(-\frac{z_{k^*} - \omega_k}{\sqrt{V_k}} \right) \tag{IV.2}$$

This last integral is intractable, therefore we have to resort to approximations.

A. Approach 1: Jensen Inequality

Using the Jensen inequality we obtain

$$\phi_{k^*} = \log Z_{k^*} = \log \mathbb{E}_{z \sim \mathcal{N}(\omega_{k^*}, V_{k^*})} \prod_{k \neq k^*} H\left(-\frac{z - \omega_k}{\sqrt{V_k}} \right) \tag{IV.3}$$
$$\geq \sum_{k \neq k^*} \mathbb{E}_{z \sim \mathcal{N}(\omega_{k^*}, V_{k^*})} \log H\left(-\frac{z - \omega_k}{\sqrt{V_k}} \right) \tag{IV.4}$$

Reparameterizing the expections we have

$$\tilde{\phi}_{k^*} = \sum_{k \neq k^*} \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)} \log H\left(-\frac{\omega_{k^*} + \epsilon \sqrt{V_{k^*}} - \omega_k}{\sqrt{V_k}} \right) \tag{IV.5}$$

The derivative $\partial_{\omega_k} \tilde{\phi}_{k^*}$ and $\partial^2_{\omega_k} \tilde{\phi}_{k^*}$ that we need can then be estimated by sampling (once?) ϵ.

B. Approach 2: Jensen again

A further simplification is obtained by applying Jensen inequality again to (IV.5) but in the opposite direction. We have the new effective free energy

$$\tilde{\phi}_{k^*} = \sum_{k \neq k^*} \log \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)} H\left(-\frac{\omega_{k^*} + \epsilon\sqrt{V_{k^*}} - \omega_k}{\sqrt{V_k}} \right) \tag{IV.6}$$

$$= \sum_{k \neq k^*} \log H\left(-\frac{\omega_{k^*} - \omega_k}{\sqrt{V_k + V_{k^*}}} \right) \tag{IV.7}$$

This gives, for $k \neq k^*$

$$\partial_{\omega_k} \tilde{\phi}_{k^*} = \begin{cases} -\frac{1}{\sqrt{V_k + V_{k^*}}} GH\left(-\frac{\omega_{k^*} - \omega_k}{\sqrt{V_k + V_{k^*}}} \right) & k \neq k^* \\ \sum_{k' \neq k^*} \frac{1}{\sqrt{V_{k'} + V_{k^*}}} GH\left(-\frac{\omega_{k^*} - \omega_k}{\sqrt{V_{k'} + V_{k^*}}} \right) & k = k^* \end{cases} \tag{IV.8}$$

Notice that $\partial_{\omega_{k^*}} \tilde{\phi}_{k^*} = -\sum_{k \neq k^*} \partial_{\omega_k} \tilde{\phi}_{k^*}$. In last formulas we used the definition

$$GH(x) = \frac{G(x)}{H(x)} = \frac{\sqrt{2/\pi}}{\mathrm{erfcx}(x/2)} \tag{IV.9}$$

V. NUMERICAL RESULTS

In this section we study multi-layer perceptrons (MLPs) on a binary classification task on the Fashion-MNIST dataset (divided arbitrarily in two classes: the first/last 5 classes in the original dataset respectively represent the first/second class in the binary classification task). We perform experiments on the whole dataset and with a MLP with two hidden layers of size 101 (apart from the sections in which we modify the architecture or the dataset size, or otherwise stated).

NB: in all figures the title reports "MNIST" while it should be "Fashion-MNIST".

M and P are the same parameter (the dataset size).

A. Experiments on Fashion-MNIST

We compare the BP family with binary-SGD without biases and without additional parameters for batch normalization. In order to keep the pre-activations of each hidden layer normalized we rescale them by $\frac{1}{\sqrt{N_{\text{in}}}}$ where N_{in} is the size of the previous hidden layer (or the input size in the case of the preactivations afferent to the first hidden layer).

1. Good parameters: batch-size 1 and batch-size 128

We report that it is possible to find values of the hyper-parameters such that the BP family has final train/test error comparable with SGD. In our experiments this holds for generic dataset sizes and batch-sizes, and for generic MLPs (generic depth / hidden layer size).

In summary, up to now the best hyper-parameters (apart from P, batch-size and ρ that has to be tuned similarly to the learning rate for SGD) are:

- $\psi = 0.8$
- $\epsilon init = 1$
- $maxiters = 1$ $(r = 0)$

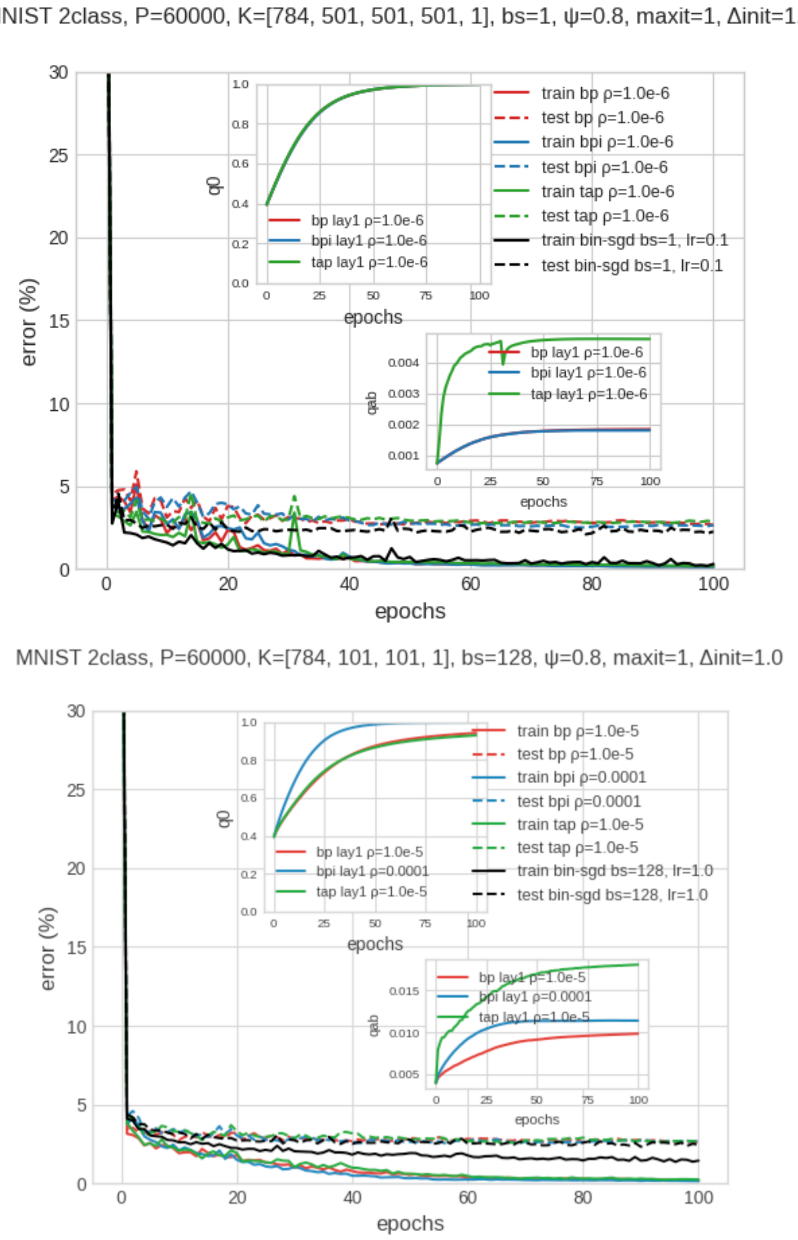

Figure V.1. Left panel: MLP with 3 hidden layers with 501 hidden units each, batch-size=1 on the Fashion-MNIST dataset. Right panel: MLP with 2 hidden layers with 101 hidden units each, batch-size=128 on the Fashion-MNIST dataset. Here we have selected some "good" values of the parameters. **NB: in all figures the upper inset is the self-overlap of the first layer, while the lower inset is the mean overlap of the couples of sub-perceptrons in the first layer.**

2. Varying batch-size: computational performance

Here we vary only the batchsize (per completezza farlo per un buon set degli altri iperparametri), in order to compare the performance (time) of BP with binary-SGD (both on GPUs).

The command to reproduce the experiments in this section is:

run_experiment(9; M=Int(6e4), batchsize=batchsize, usecuda=true, gpu_id=0, ρ=1+1e-5, ψ=0.5, lay=lay, epochs=100)

with batchsize={1,16,128,1024} and lay={:bp, :bpi, :tap}.

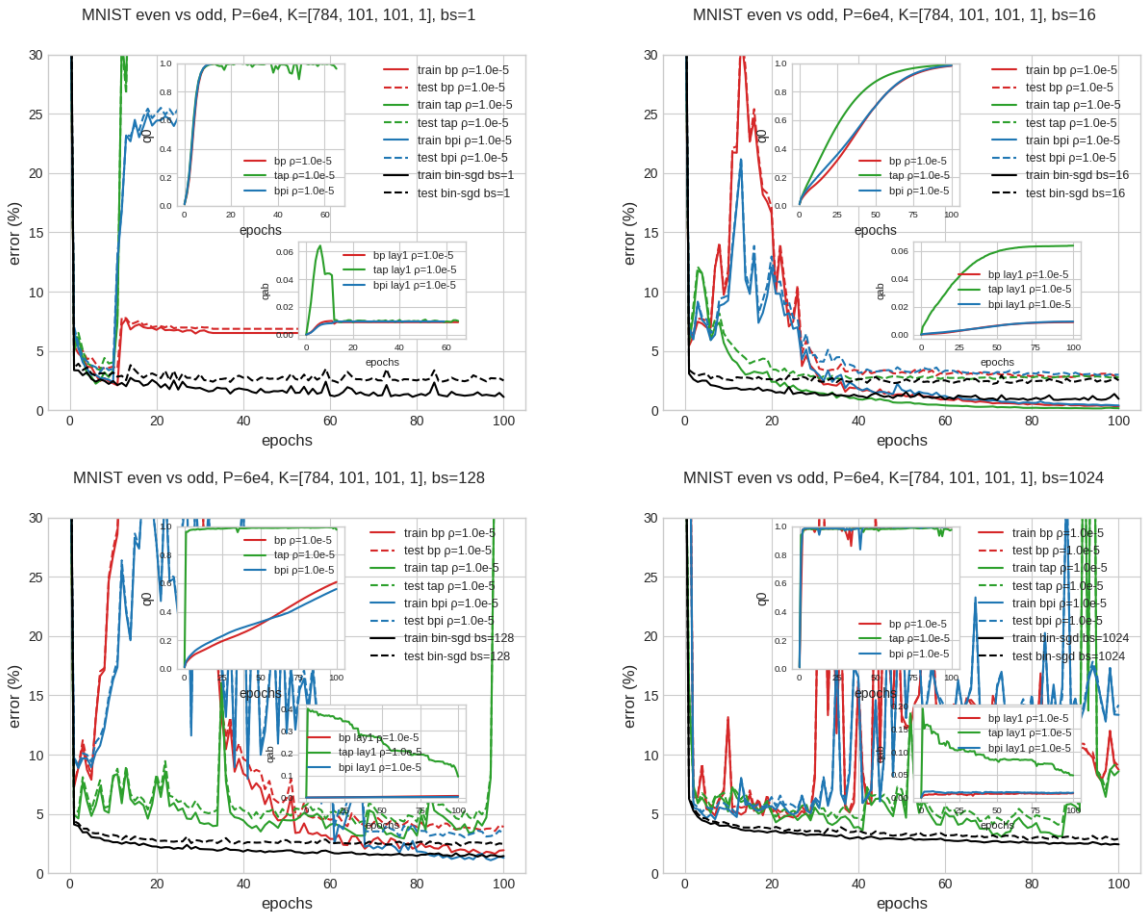

Figure V.2. Comparison of BP, TAP, BPI, SGD varying the batchsize (upper left: bs=1; upper right: bs=16, lower left: bs=128; lower right=1024). The parameter $\rho - 1$ is fixed in all experiments to 10^{-5}.

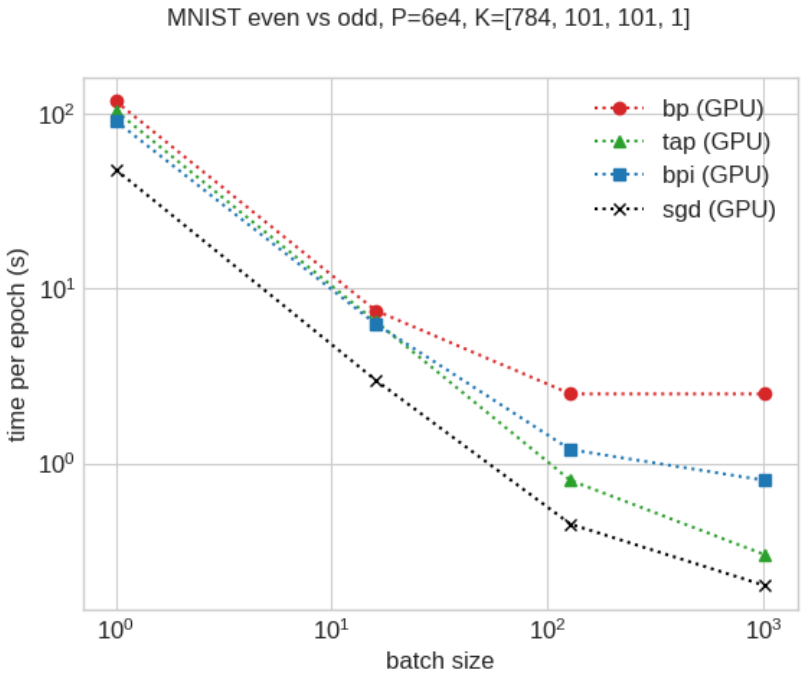

Figure V.3. Algorithms time scaling with the batchsize. The reported time refers to one epoch for each algorithm.

B. Experiments on Fashion-MNIST: parameters

Here we stick to batchsize 128 and we vary the other hyperparameters. However, we expect that some of the parameters scale with the batchsize (in particular we expect $\rho - 1 \propto \frac{\text{batchsize}}{\text{data set size}}$).

The command to reproduce the experiments is along the lines of:

run_experiment(9; usecuda=true, gpu_id=0, epochs=100, lay={:bp, :bpi, :tap}, batchsize=128, ρ=p[2], ψ=0.5, M=Int(6e4), maxiters=1, r=0., K=[28*28,101,101,1])

1. *Varying ρ*

ρs = [-1e-1, -1e-5, 0., 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1] .+ 1.

For $bs = 128$ we choose $\rho - 1 = 10^{-4}$. The command is the same as the one in the first paragraph of section B except for ρ.

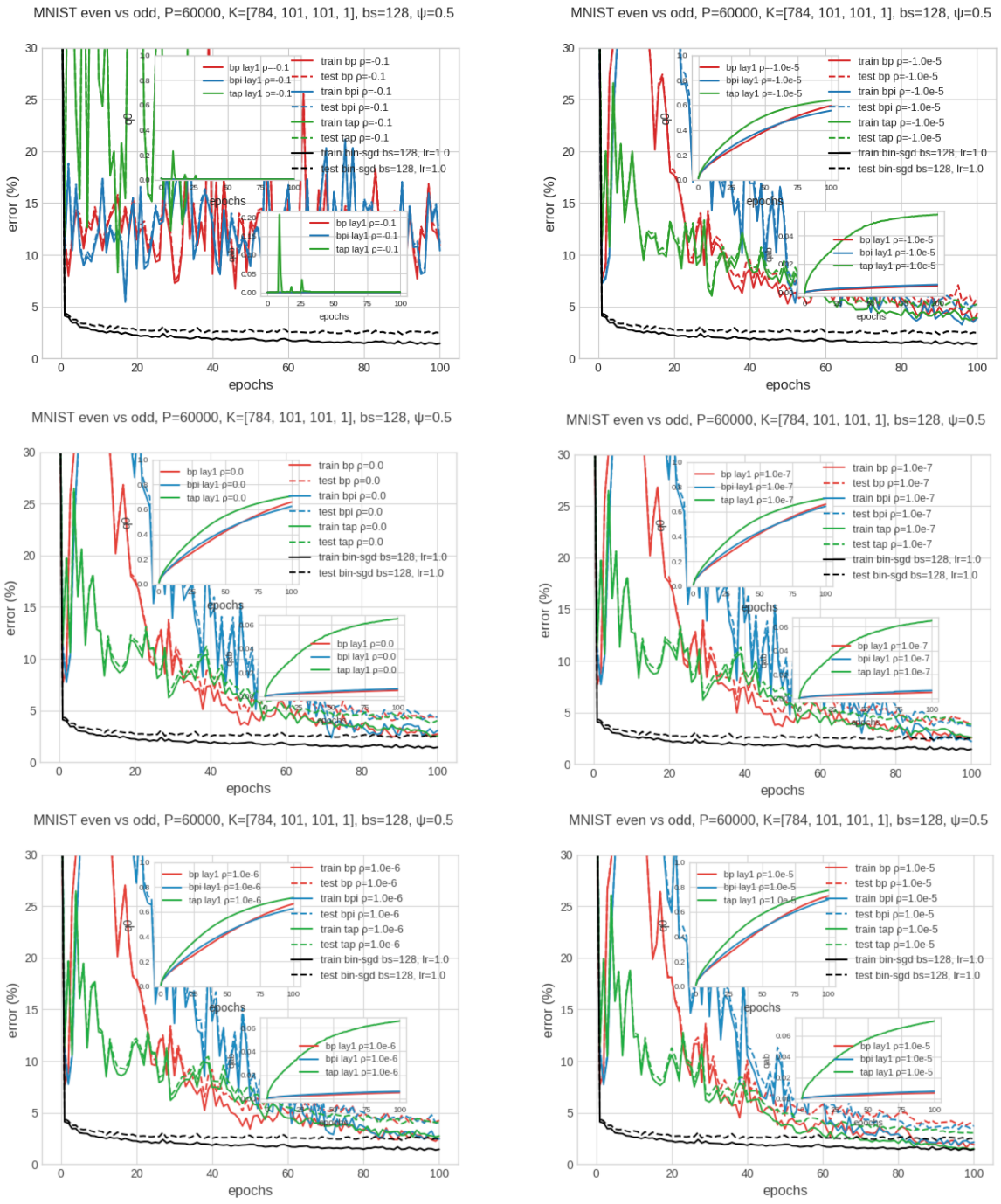

Figure V.4. MLP with 2 hidden layers with 101 hidden units each, batch-size=128 on the Fashion-MNIST dataset. We vary the parameter ρ.

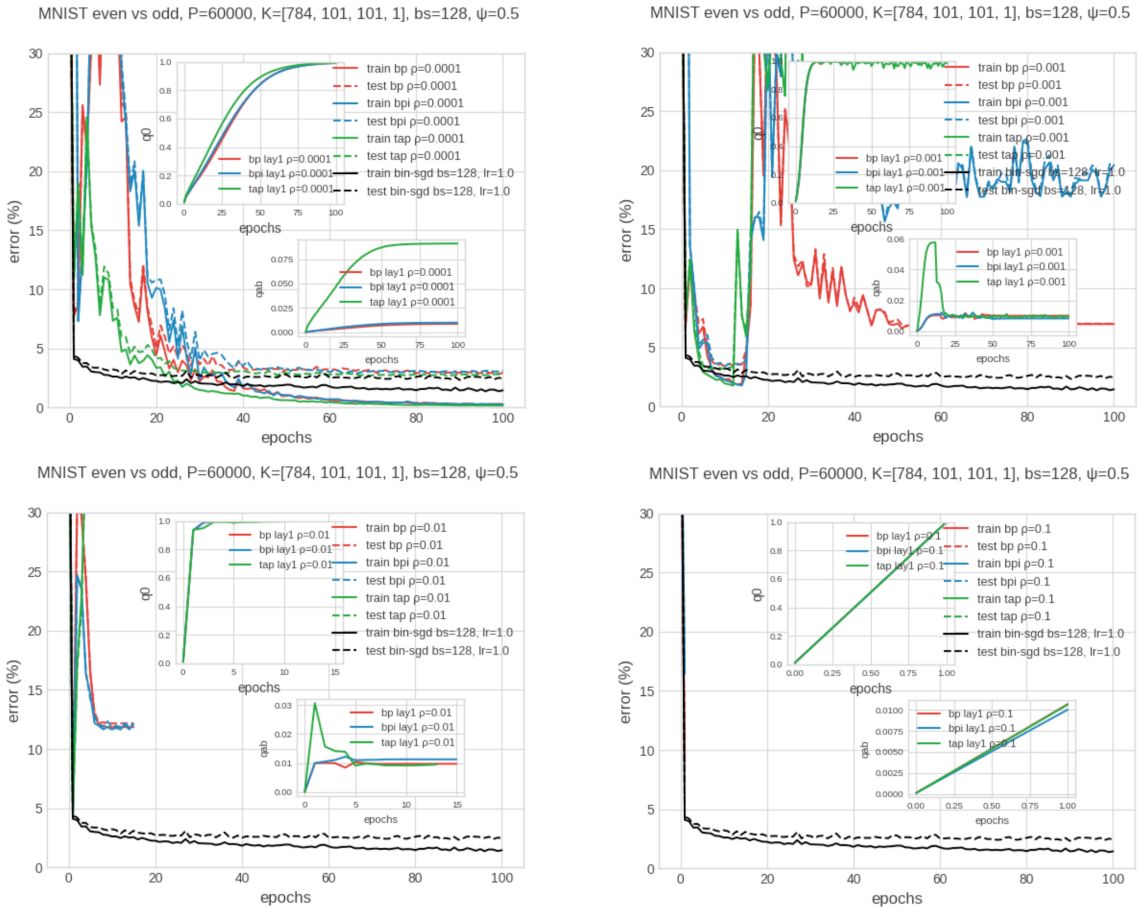

Figure V.5. MLP with 2 hidden layers with 101 hidden units each, batch-size=128 on the Fashion-MNIST dataset. We vary the parameter ρ.

2. *Varying damping* ψ

ψs = [0:0.2:1;] ∪ [0.9, 0.99, 0.999, 0.9999]

For $bs = 128$ and $\rho - 1 = 10^{-4}$ we choose $\psi = 0.8$. The command is the same as the one in the first paragraph of section B except for ρ and ψ.

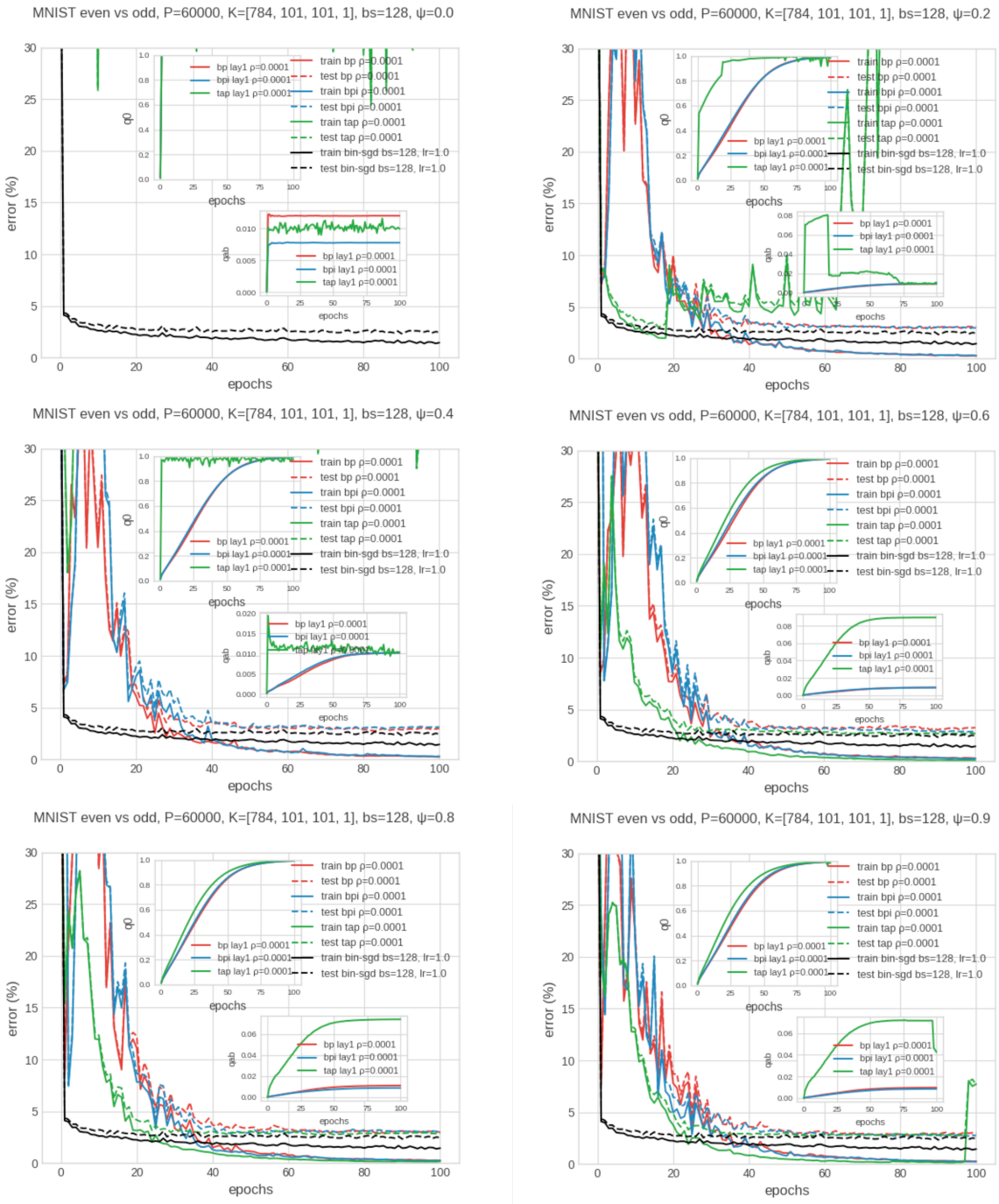

Figure V.6. MLP with 2 hidden layers with 101 hidden units each, batch-size=128 on the Fashion-MNIST dataset. We vary the parameter ψ.

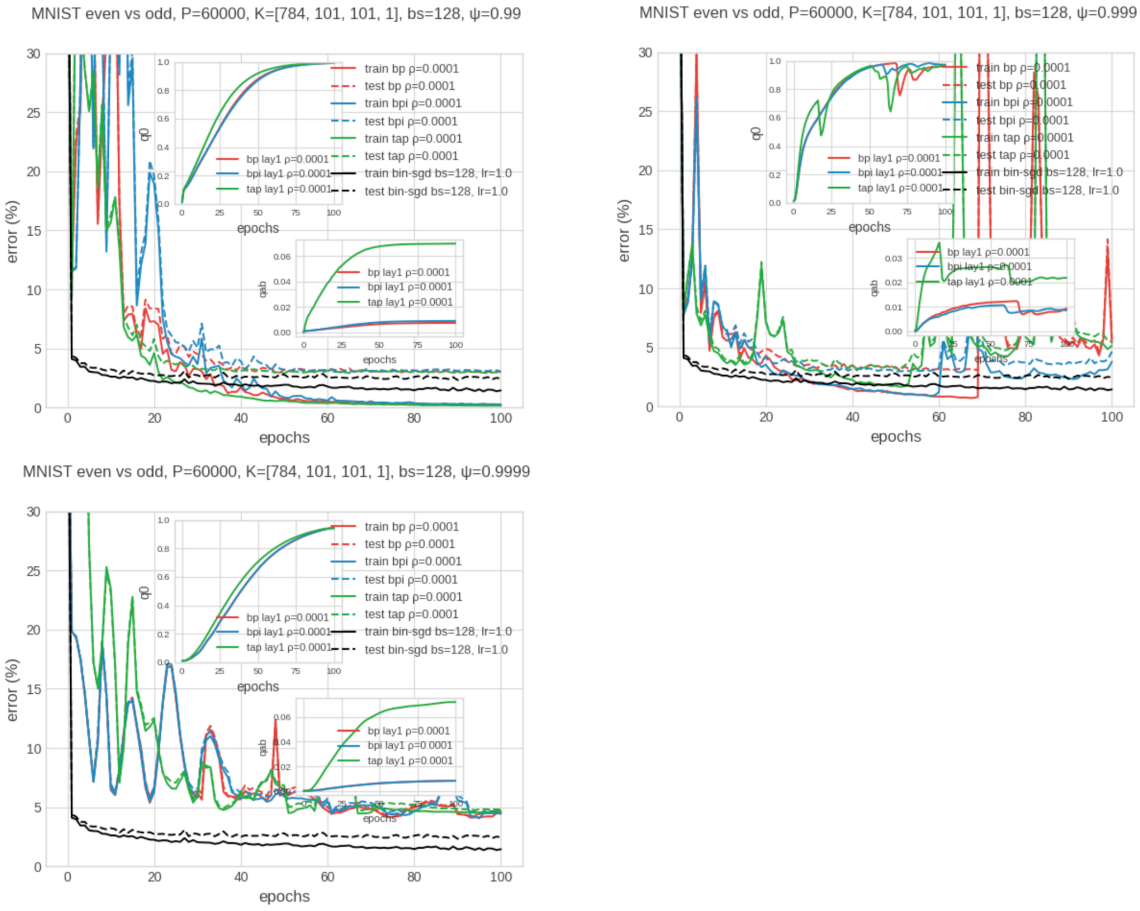

Figure V.7. MLP with 2 hidden layers with 101 hidden units each, batch-size=128 on the Fashion-MNIST dataset. We vary the parameter ψ.

3. Varying initial weights

ϵinits = [0., 1e-3, 1e-2, 1e-1, 5e-1, 1e0]

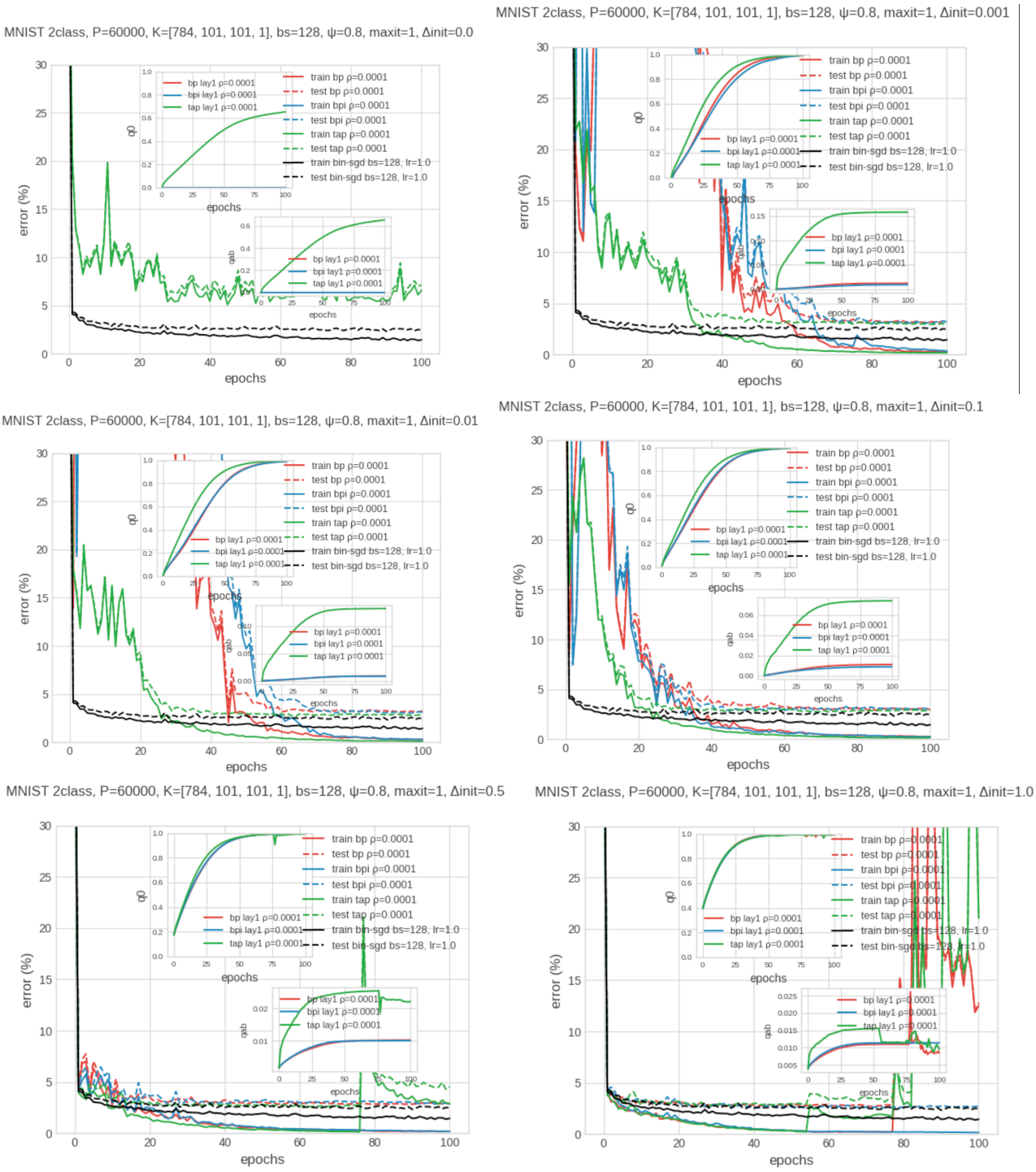

Figure V.8. MLP with 2 hidden layers with 101 hidden units each, batch-size=128 on the Fashion-MNIST dataset. We vary the parameter ϵinit.

4. Varying dataset size

Ms = [Int(1e2), Int(1e3), Int(1e4), Int(6e4)]
bs = [Int(1e0), Int(1e1), Int(1e2), Int(6e2)]
We fix the ratio $\frac{\text{dataset size}}{\text{batch size}} = \frac{M}{b} = 10^2$.

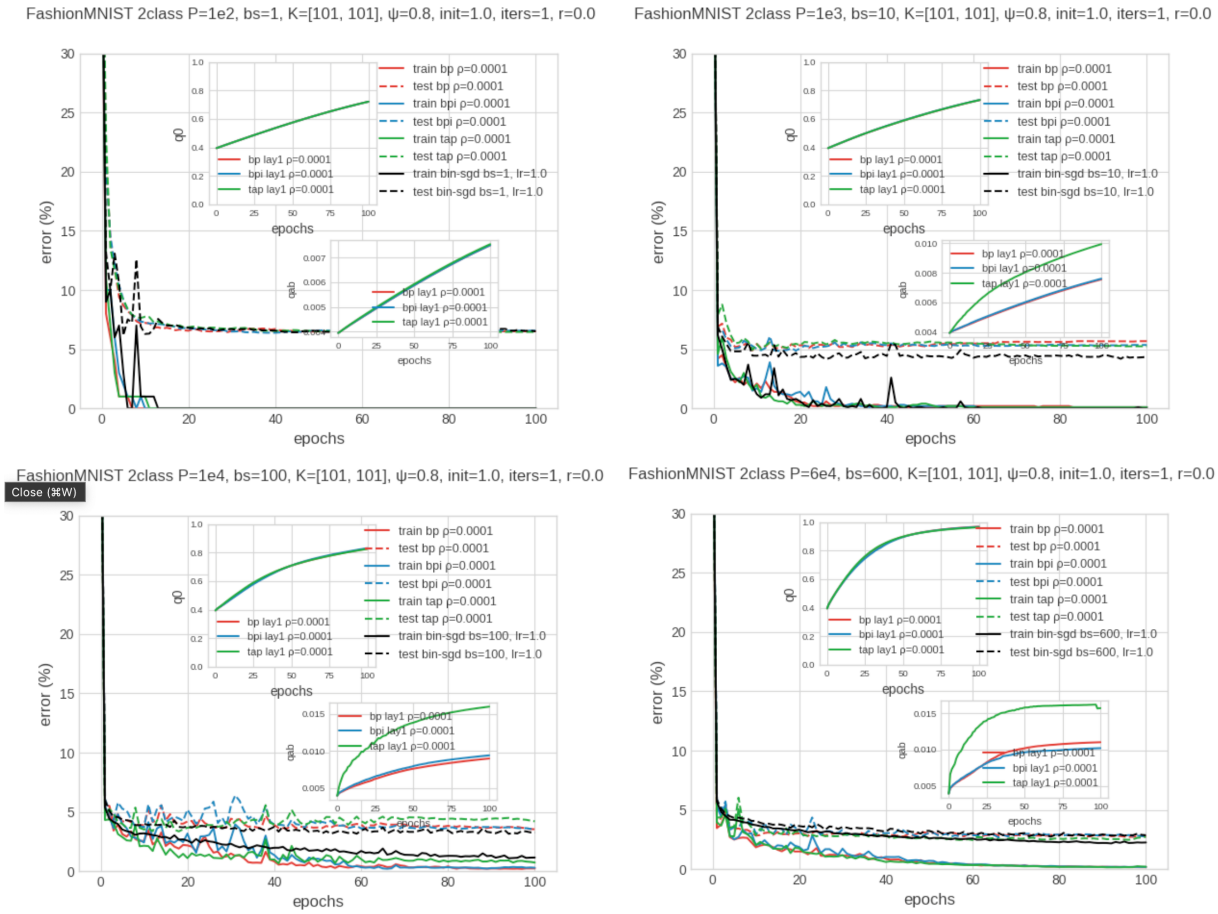

Figure V.9. MLP with 2 hidden layers with 101 hidden units each, batch-size=128 on the Fashion-MNIST dataset. We vary the parameter M (with $\frac{M}{b} = 10^2$).

5. *Varying maxiters*

maxiterss = [1, 10, 50, 100]. Here also time is interesting. It is not very favorable in terms of time compared to SGD to choose $maxiters > 1$, however it is interesting how many iterations are necessary for convergence.

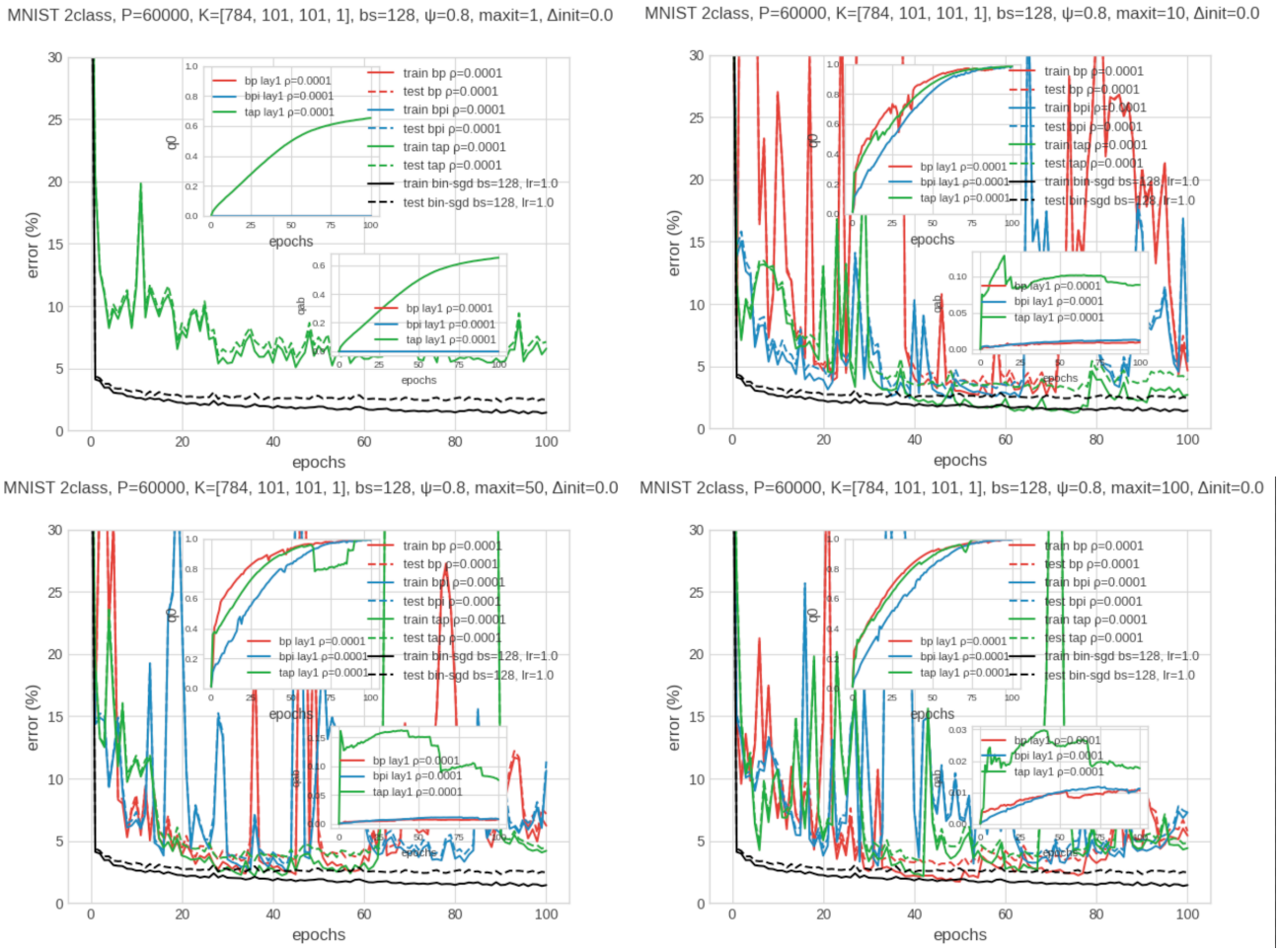

Figure V.10. MLP with 2 hidden layers with 101 hidden units each, batch-size=128 on the Fashion-MNIST dataset. We vary the parameter *maxiters*.

6. *Varying r*

rs = [0:0.2:1.2;] (for maxiters=10). Da rifare.

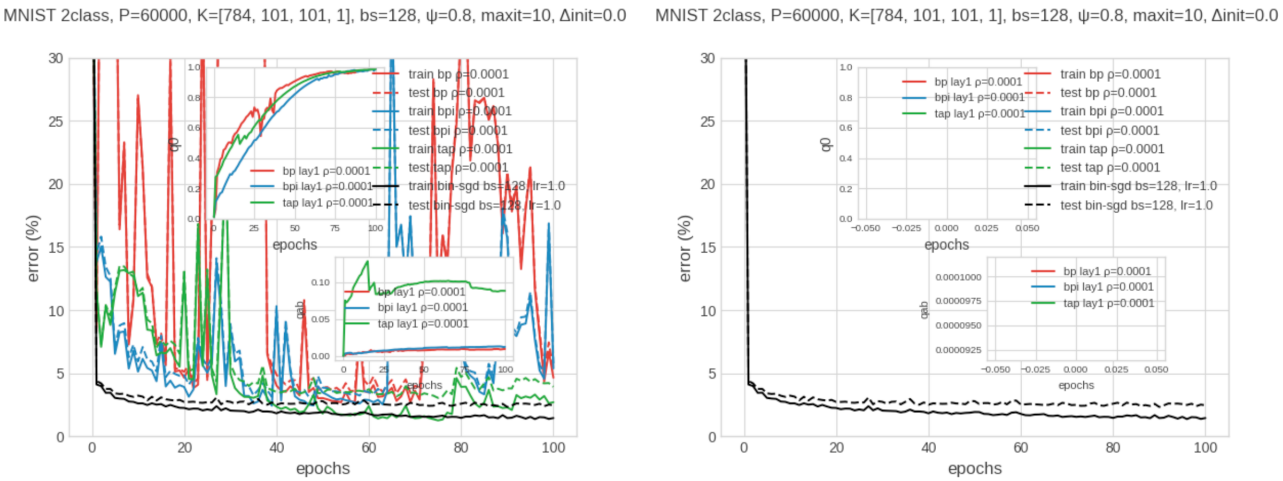

Figure V.11. MLP with 2 hidden layers with 101 hidden units each, batch-size=128 on the Fashion-MNIST dataset. We fix $maxiters = 10$ (just to be faster) and we vary the parameter r. I vari valori di $r \neq 0$ non hanno proprio funzionato (sembra siano tutti NaN) - gli altri parametri vanno cambiati.

7. *Varying the number of layers and the size of the hidden layer*

We fix some reasonable values of the hyperparameters (see previous experiments, in particular: $P = 6e4$, $bs = 128$, $\psi = 0.8$, $\epsilon init = 1$, $maxiters = 1$, $r = 0$) and want to check if the algorithm converges (and todo the time scaling).

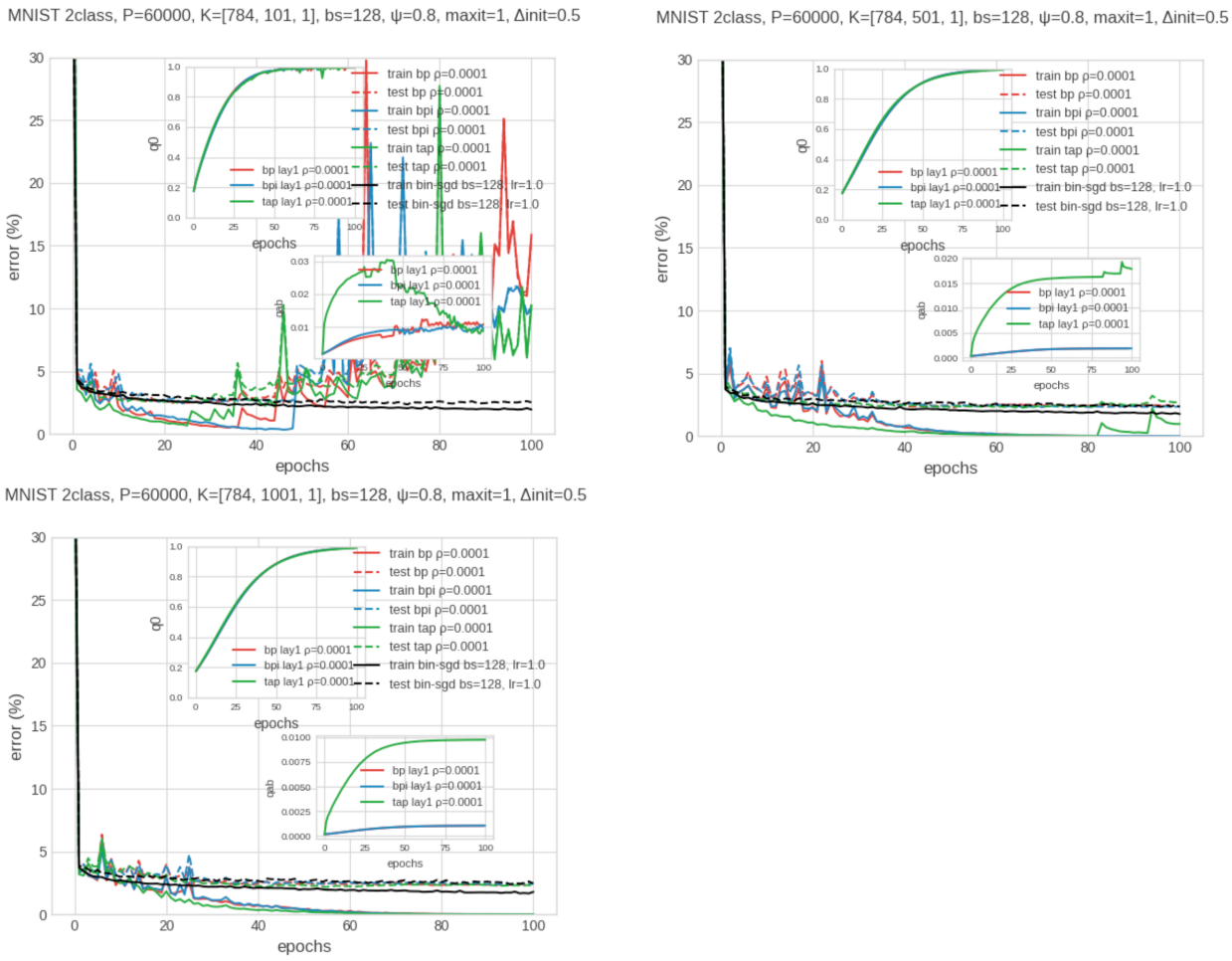

Figure V.12. MLP with 1 hidden layer with 101/501/1001 hidden units each, batch-size=128 on the Fashion-MNIST dataset.

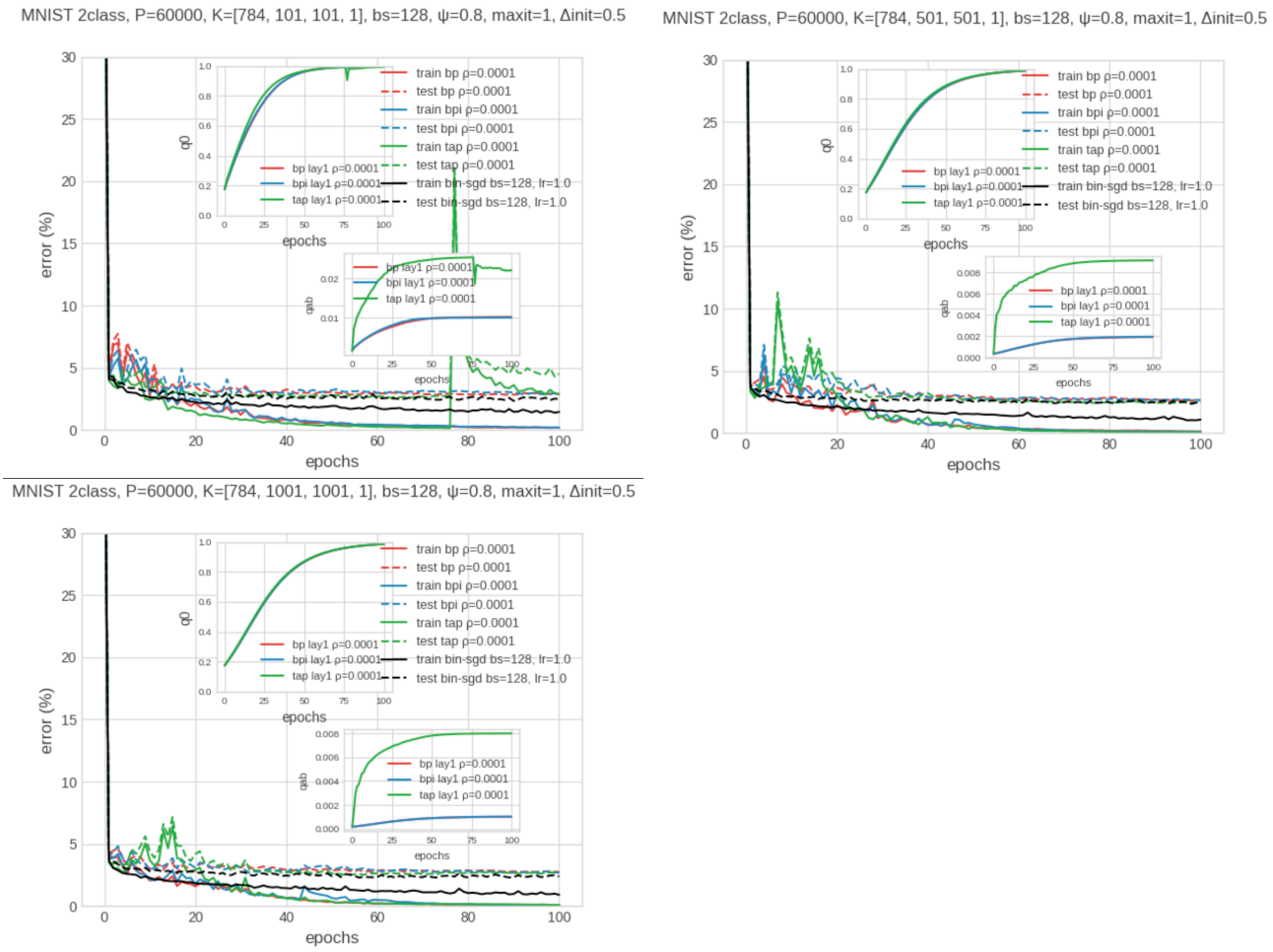

Figure V.13. MLP with 2 hidden layers with 101/501/1001 hidden units each, batch-size=128 on the Fashion-MNIST dataset.

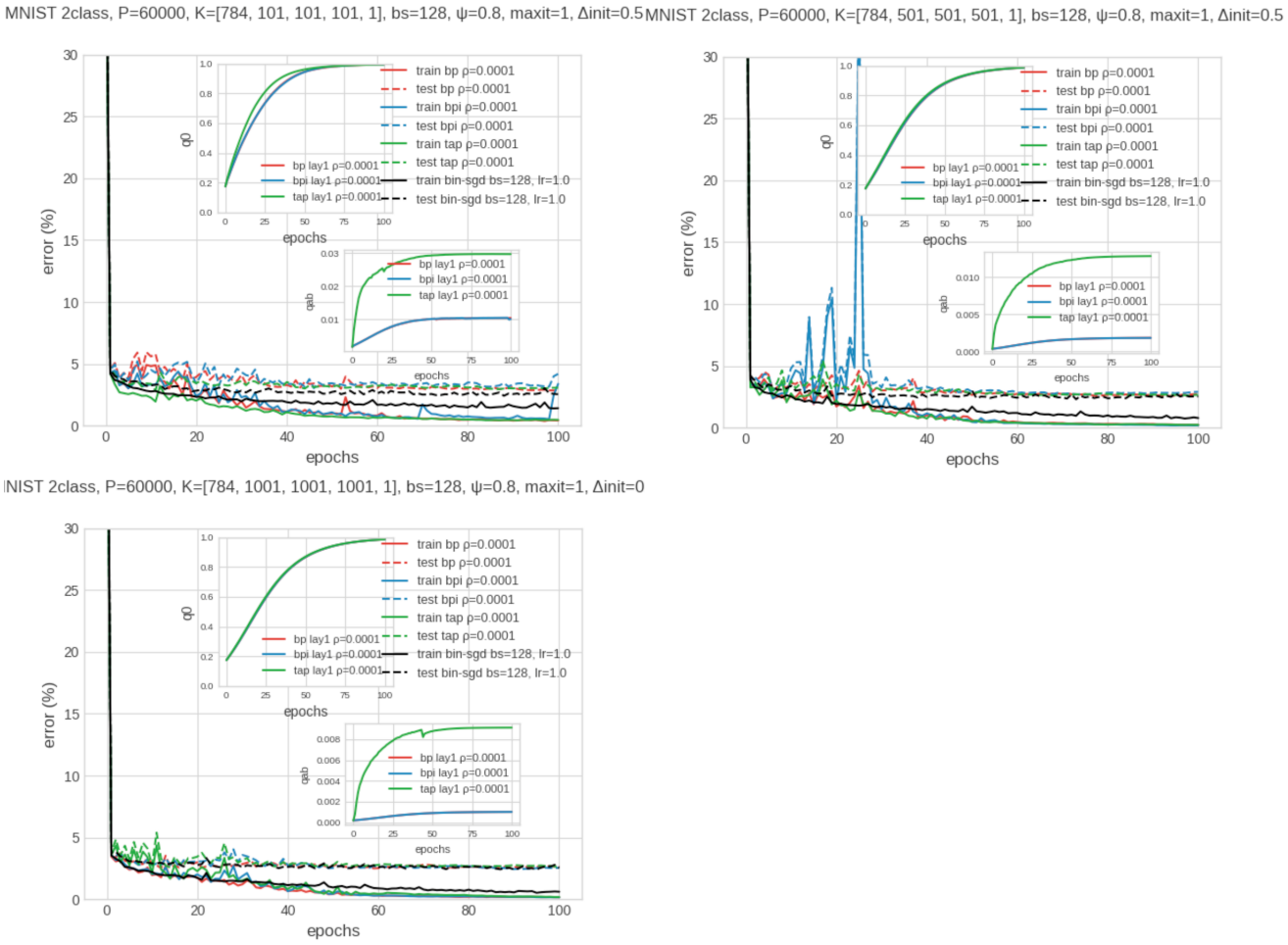

Figure V.14. MLP with 3 hidden layers with 101/501/1001 hidden units each, batch-size=128 on the Fashion-MNIST dataset.

C. Experiments on RFM

Here we present results with the Random Features Model, concerning in particular the permutation symmetry. In order to investigate the role of the permutation symmetry we present results on the fully connected committee machine (1 hidden layer network learning only the first weight layer, with the weights of the second layer fixed to all ones).

The teacher in the latent space is a perceptron (check).

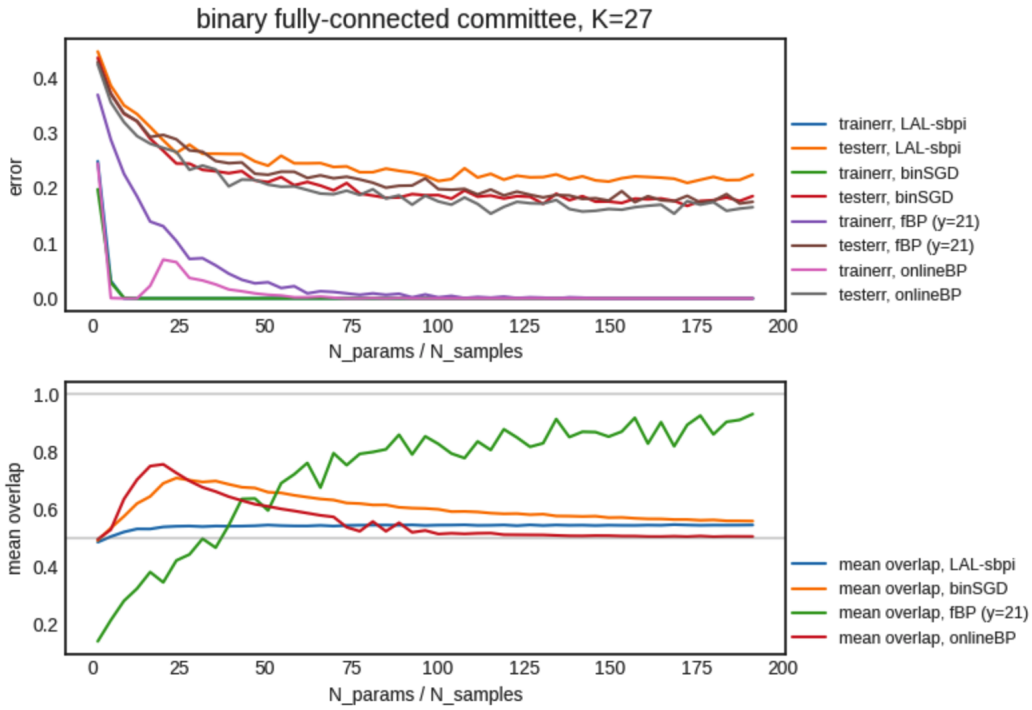

Figure V.15. Overlap varying N in the RFM, fully connected committee machine with various algorithms.