ML 2023-2024

Authors: **Alberto Dicembre, Giuseppe De Marco**

Team name: **Exploding gradients**

Date: **08/07/2024**

Type of project: **A (Python)**

Objectives

- Assessing the performance of an **handcrafted MLP**, experimenting with different state-of-the-art techniques; the «DIY» approach enabled us to really grasp che concepts from the ground up, to estabilish deep knowledge on the mechanisms seen during the course.

- Finding out how different problems (**binary classification** and **non-linear regression**) require different methodologies and hyperparameter combinations, with an iterative and incremental approach that allowed us to familiarize well with the subject and deep dive into how Neural Networks work.

Our contributions

- Code consists of two Jupyter Notebooks (MONK / CUP) and some source files.
- Development of a MLP, trained with Gradient Descent by Backpropagation.
- Features:
  - **Number of layers**: 1, 2, 3, 7;
  - **Activation functions**: Sigmoid, Tanh, ReLU, Leaky ReLU, Softmax, Linear;
  - **Training algorithms**: classic Backpropagation;
  - **Batching modes**: Online, Mini-Batch, Full Batch;
  - **Initialization**: Xavier (Glorot), He;
  - **Regularization techniques**: Tikhonov (L2), Lasso Regression (L1);
  - **Stopping conditions**: Early Stopping with patience value;
  - **Learning rate schedule**: Linear learning rate decay;
  - **Momentum**: standard Momentum (no Nesterov).

MONK Results

Developing separate models for the 4 different problems/datasets has yielded the best results;
Some hyperparameters, though, were **fixed** for all 4 models (Table 1). The motivation for **Tanh** is described in [1].

N. Hidden Layers	Epochs (fixed)	Batch Size	Hidden Layer act.	Output Layer act.	Weight Initial.
1	300	1 (Online)	Tanh	Sigmoid	Xavier

Table 1: Fixed Hyperparameters for MONK

Task	Units	Eta	Alpha (Momentum)	Lambda	MSE(TR/TS)	ACC(TR/TS)(%)
MONK 1	4	0.15	0.85	/	0.00002/0.0003	100/100%
MONK 2	4	0.2	0.8	/	0.00001/0.00007	100/100%
MONK 3 (no reg.)	4	0.003	0.7	/	0.0216/0.0215	95.08/95.55%
MONK 3 (reg)	4	0.001	0.9	0.00001	0.058/0.054	93.44/97.22%

Table 2: MONK hyperparameters and results, averaged over 10 executions

MONK 1 Plots

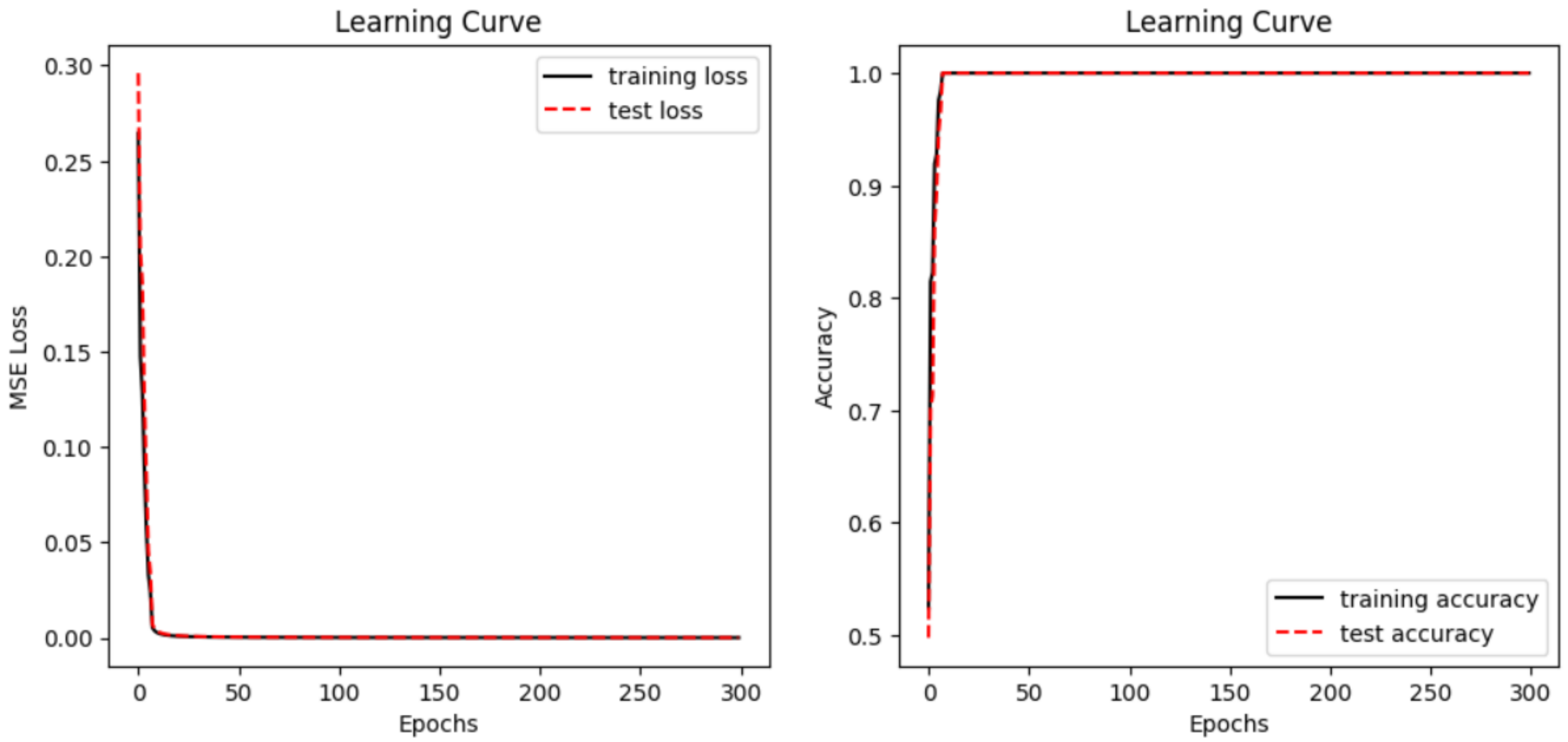


Figure 1: Results on MONK 1 Dataset

MONK 2 Plots

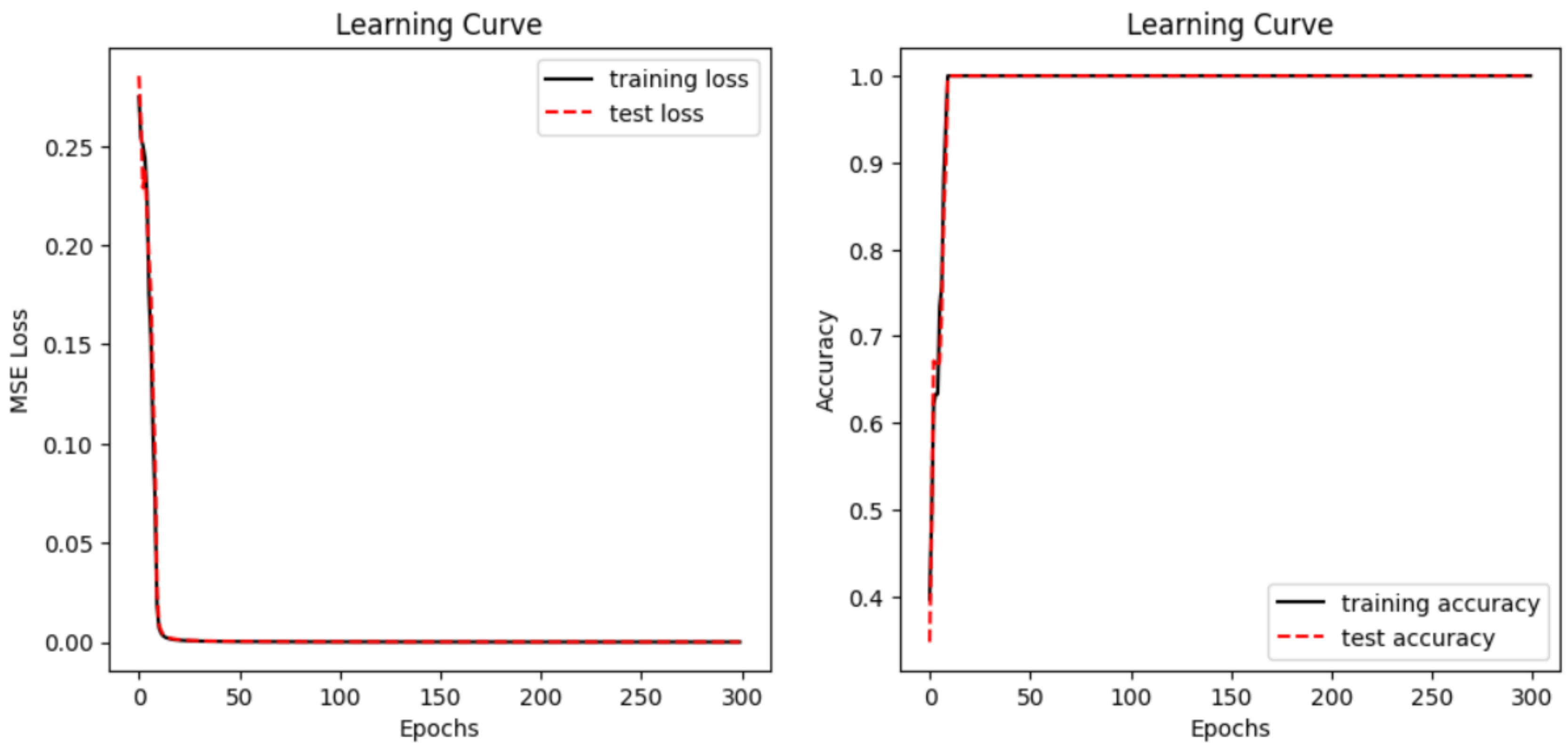


Figure 2: Results on MONK 2 Dataset

MONK 3 No reg. Plots

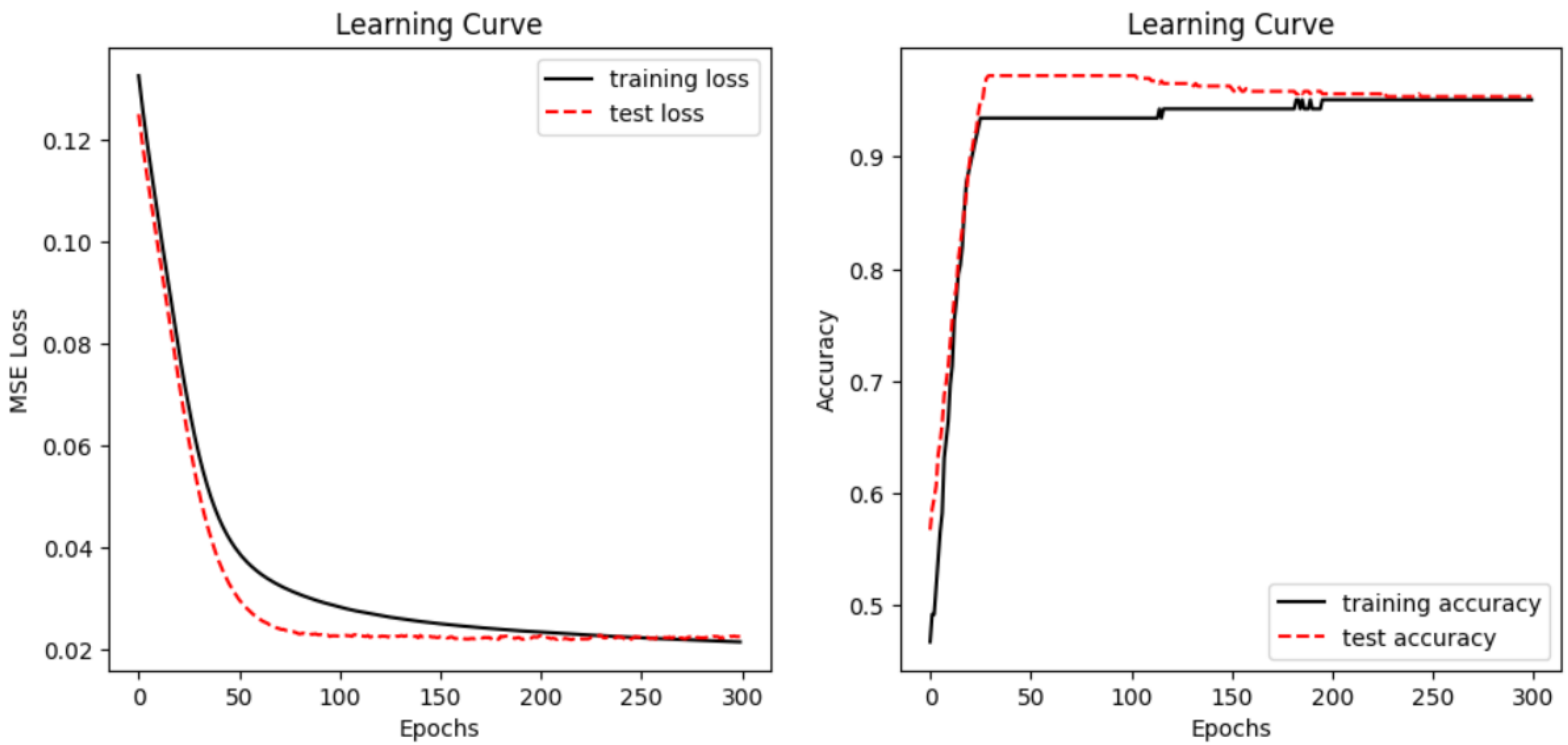
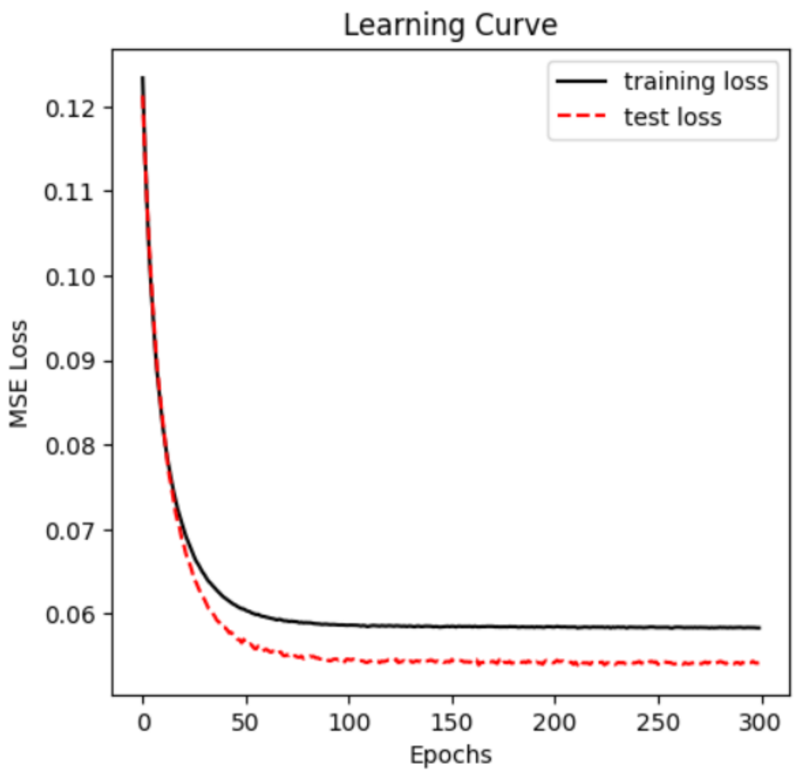


Figure 3: Results on MONK 3 Dataset with no regularization

MONK 3 Reg. Plots



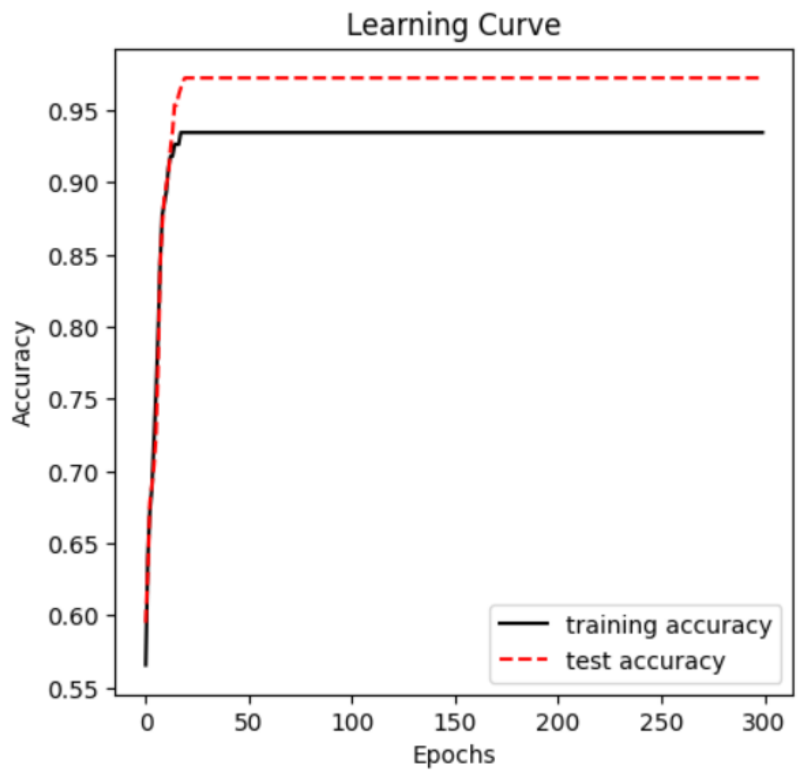


Figure 3: Results on MONK 3 Dataset with regularization

CUP Validation Schema

- Data was split in the following way: 80% train (including a 10% Early Stopping Validation), 20% internal test.
- Model selection was performed through multiple **Grid Searches** (by usage of **5-fold cross validation** on the train split):
  - Often preceded by preliminary tests;
  - This iteratively, increasing the number of layers.
- Based on the Grid Searches results, we selected the **best 5 models**, according to their Validation Error (MEE) Mean calculated over the 5 folds.
- Then, the final results on the *Internal* Test Set and the predictions for the *Blind* Test Set were obtained after **retraining** the final model on the whole training set (including the Early Stopping portion).

CUP Validation Schema: Model Selection

- In CUP we have some **fixed hyperparameters** as well (Table 3). We have decided to fix them after preliminary trials, or for efficiency reasons in case of the number of epochs.
- The Model Selection was performed by fixing the (shuffled) data split, to eliminate the difference introduced by using a different partitioning each time (even though models have very low variance, as we will discuss later).

Epochs (max.)	ES patience	Hidden Layer act.	Output Layer act.	Weight Initial.	Lasso Reg.
500	30	Tanh	Linear	He	No

Table 3: Fixed Hyperparameters for CUP

CUP Validation: Grid Searches

N. Layers	N. Neurons	Eta	Batch Size	Momentum	Reg (L2)	LR Decay Steps*	LR Decay Value	Best Validation Mean
1	100, 200, 300, 400	0.1 0.01 0.001 0.0001	1, 2, 8, 128, full	0.6, 0.8, 0.9, 0.95, 0.97	No, $1 * 10^{-8}$, $5 * 10^{-8}$, $1 * 10^{-5}$, $1 * 10^{-3}$	No, 100	5, 10, 25, 50, 100	0.72
2	50, 100, 150, 200	0.001, 0.005, 0.0001	1, 8	0.5, 0.6, 0.7, 0.8, 0.9, 0.95	No, $1 * 10^{-8}$	No, 100	5, 10, 25, 50, 100	0.61
3	100, 150, 200	0.0001	1, 8	0.5, 0.6, 0.8, 0.9	No, $1 * 10^{-8}$	No	No	0.57
7	70, 100, 150	0.001, 0.0005	8	0.5, 0.6	No, $1 * 10^{-8}$	No	No	0.71

Table 4: Grid Searches description for CUP

* Note that when LR Decay Steps is disabled, LR Decay Value is ignored as well

CUP Results: Top 5 Models

N°	N. Layers	N. Neurons	Eta	Batch Size	Momentum	Reg. (L2)	Linear Learning rate Decay	TR/VL
1	3	200	0.0001	1	0.8	$1 * 10^{-8}$	No	0.244/0.577
2	3	200	0.0001	1	0.6	$1 * 10^{-8}$	No	0.262/0.600
3	2	200	0.0001	1	0.9	No	No	0.256/0.610
4	3	200	0.0001	8	0.9	No	No	0.260/0.617
5	3	200	0.0001	1	0.5	No	No	0.259/0.633

Table 5: Top 5 models for CUP. The reported scores are the Training and Validation MEE errors, averaged over the k-fold validation.

CUP Results: the Final model (Ensemble)

- We decided to make use of the *Ensembling* technique to obtain a unique model from the best 5, which turned out to be an overall **better predictor** (both score and smoothness-wise) than the single best.
  - The comparison was made by training the models with a 5-fold CV on the whole training set (including the 10% Early Stopping partition), using the training error found over the grid search as a *stopping criterion*.
  - As an experiment, we also decided to include the best single model (without the use of Early Stopping) in the comparison, training it for a fixed number of 2000 epochs.

Model	TR	VL	Variance
Best single (ES)	0.235	0.560	0.0046
Best single (No ES)	0.175	0.538	0.0277
Ensemble (Top 5)	0.213	0.519	0.0002

Table 6: Comparison of the best model against the ensemble. The reported scores are the Training and Validation MEE errors, averaged over the k-fold validation, and their variance.

CUP Results: Model Assessment

- Once we estabilished the ensemble model as the final best one, we made the final assessment: a **final retraining** of it over the Training Set, and its evaluation on the Internal Test Set.
- Such evaluation was performed by **averaging** over 5 executions, to reduce the bias component and be able to calculate a variance value.

TR	TS	Variance
0.218	0.405	$1.8 * 10^{-5}$

Table 7: Results of the final model on CUP Dataset, averaged over 5 executions.

CUP Results: Plots

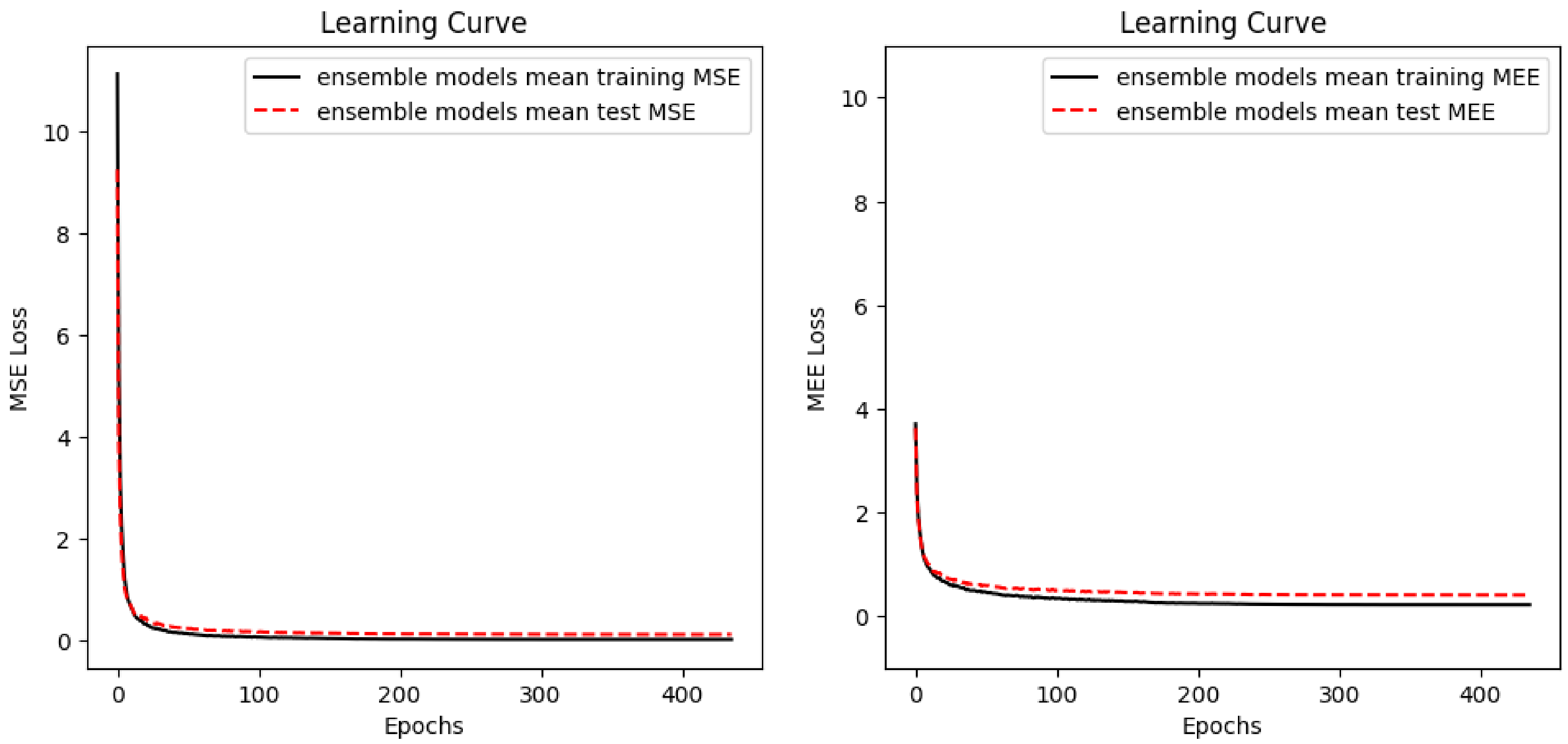


Figure 4: Learning curves of the final model on CUP Dataset, averaged over 5 executions.

Discussion: Some implementation details

- Early Stopping
  - We have implemented early stopping so that it would stop the training after detecting a rise in validation error for 30 epochs (patience) in a row.
  - Also, as we mentioned previously, we use a **dedicated** data split to validate the early stopping error threshold. This value is then saved and used on the final retraining: it represents the training error after which the training has to be stopped, to avoid overfitting.
- L1-L2 Regularizations:
  - We implemented L1 and L2 regularization such that the penalty term is not multiplied by the learning rate on the weights update, so that the two hyperparameters could be completely **independent** from each other (that could be useful for the grid search).

Discussion: Complexity and Overfitting

- One aspect we found mesmerizing is that we never actually observed a **strong overfitting** trend on the CUP dataset.
- In order to prevent overfitting, we implemented **Linear lr decay**, **early stopping** and tested two different kinds of Penalty-Term Regularization: **L1** and **L2** (which is also part of the two best single models);
- Early Stopping patience is often reached, triggering early stopping, but we found that:
    - Disabling the Early Stopping and prolonging the training for a high number of epochs actually improved results (as we show in the best models comparison, *slide* *13*);
    - In general, training for more epochs always **increased** performances (even though by very small amounts after a certain point);
    - The best models proved to be the more **complex** ones. Because of computational demands reasons, models with more than 3 layers haven't been thoroughly tested, but they would have possibly improved performances even more.

Discussion: Hyperparameters

- Another noteworthy aspect is the behaviour of the activation function: we initially opted for ReLU with the He initialization (as suggested in [2]), but actually found that **TanH** performed a little better.

- Other than that, other observations relate to some of the hyperparameters:
  - **Low minibatch sizes** (and Online) led to best results on these problems, at the expense of a higher computational time and, for some models, irregular learning curves;
  - **Momentum** proved to be very useful at speeding the training and help the error converge towards the minimum; however it requires particular caution because can make the curve irregular.
  - **Linear lr Decay** was implemented and inserted in the validation procedure but none of the best models ended up including it.
  - **L2 regularization** performs always better than L1 so the latter was excluded from grid searches.

Discussion: Ideas for improvement

- One aspect that wasn’t initially considered in the development of the code for the architecture is the possibility of having layers with **different numbers of neurons**: if incorporated, it would have been a possibility for more experimentation with the hyperparameters;

- Another interesting addition we thought about would be the refinement of the **Ensembling** technique: we could calculate the predicted output using a weighted average, based on the validation mee of the single models instead of the simple average;

- **Nesterov** accelerated momentum would have made for an alternative to the momentum implementation we used;

Conclusions

- As a conclusion, this project made for a really good exercise for chiselling the theoretical (and practical) components that being an AI student requires.

- We have learned that tuning an MLP can be a demanding process and the outcomes can never be predicted apriori, because different hyperparameters can have unpredictable behaviours on different models.

- The elaborate required time and patience, at times it felt disorientating but at the end it was a very rewarding experience.

Blind test results: Exploding_gradients_ML-CUP23-TS.csv

Appendix: Time and Hardware specification

- Final model training time: ~ 8.8 seconds per epoch (the training initially includes the fitting of 5 models at the same time, then the computational load lowers when they begin early stopping). With the final early stopping happening on the 420th epoch, the final training took ~**40 minutes**.
  - Hardware: Intel Core i7-11800H @2.30 GHz
  - The Grid Search computation was **parallelized** via Multiprocessing.

Bibliography

[1]: Glorot, X. ; Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, in Proceedings of Machine Learning Research.*

[2]: 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