ML 2023 Project Slides

Authors:

Chiara Checchetti (583054) - c.cecchetti@studenti.unipi.it

Nicola Emmolo (672926) - n.emmolo@studenti.unipi.it

Team name: gradient_decent

Master degree: MSc in Computer Science - Artificial Intelligence

Date: 31/01/2024

Type of project: A

OBJECTIVES

- Implementation of an MLP capable of performing binary classification and linear regression tasks
- Optimization of the various losses considered and metrics used to evaluate the ability to generalize and predict on new data (accuracy and MEE respectively).
- Analyses of the best models obtained.
- Solve the MONK problems and obtain the output of the blind test for the CUP

YOUR CONTRIBUTIONS

- Libraries: Numpy / Pandas / Matplotlib / Itertools
- Implementation
  - Multi Layer Perceptron with one hidden layer
  - Output: 1 unit for MONK, 3 units for CUP
  - Activation Functions
    - Monk: Sigmoid (hidden), TanH (output)
    - CUP: ReLU / Leaky ReLU / TanH (hidden), Identity (output)
  - Training: Minibatch Stochastic Gradient Descent and Backpropagation
  - Weights Initialization: Random Uniform
  - Bias Initialization: All 1
  - Regularization: Tikhonov Regularization (L2) + Early Stopping (only for CUP)
- Preprocessing
  - Monk: One Hot Encoding and preliminary statistical tests (to check for a normal distribution)
- Validation
  - Hold Out & K Fold Cross Validation (K = 5)

MONK RESULTS

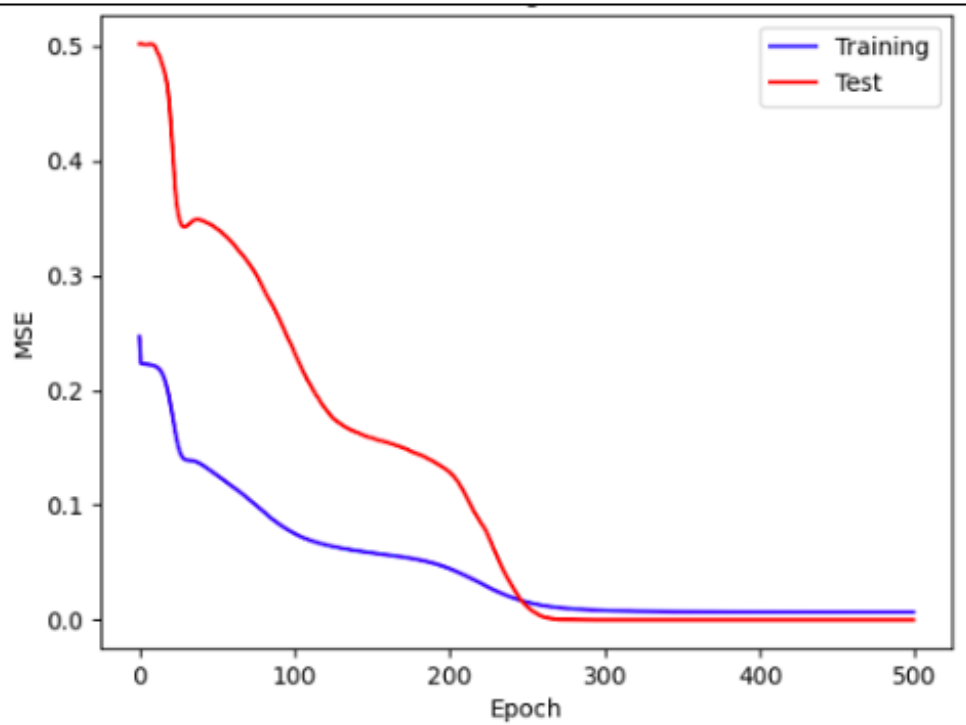
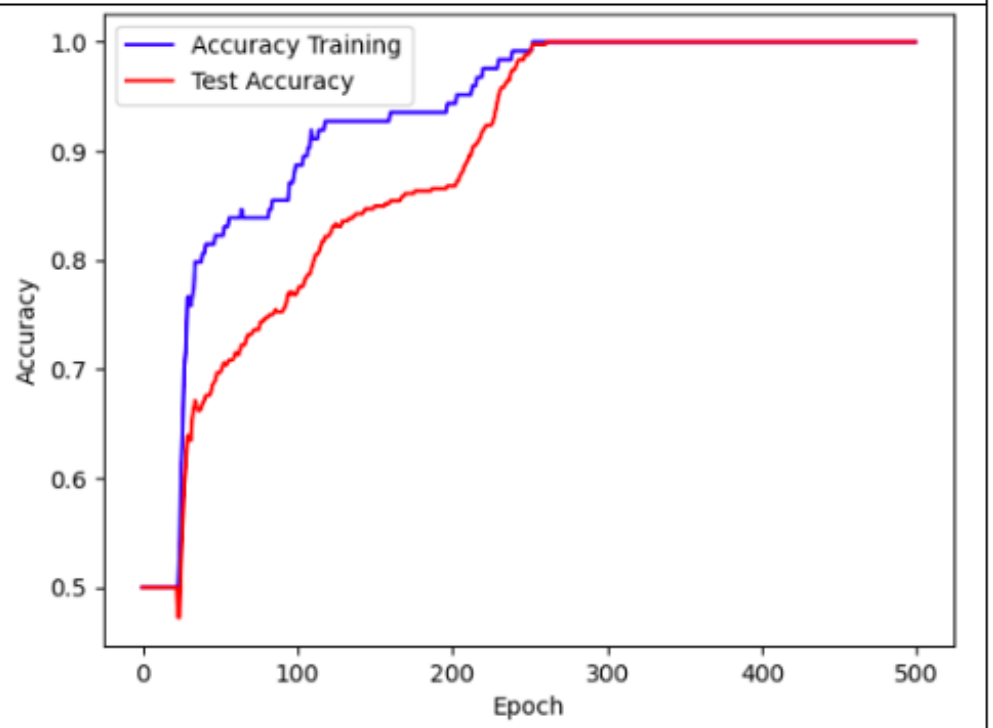
- To understand best initializations:
  - Small Grid Search, Theory, Personalized trials

- Fixed for every MONK:
  - **Epochs**: 500 / **Batch Size**: 1 / **Momentum**: 0.9 / **Weight Initialization Limits**: [-0.2, 0.2]

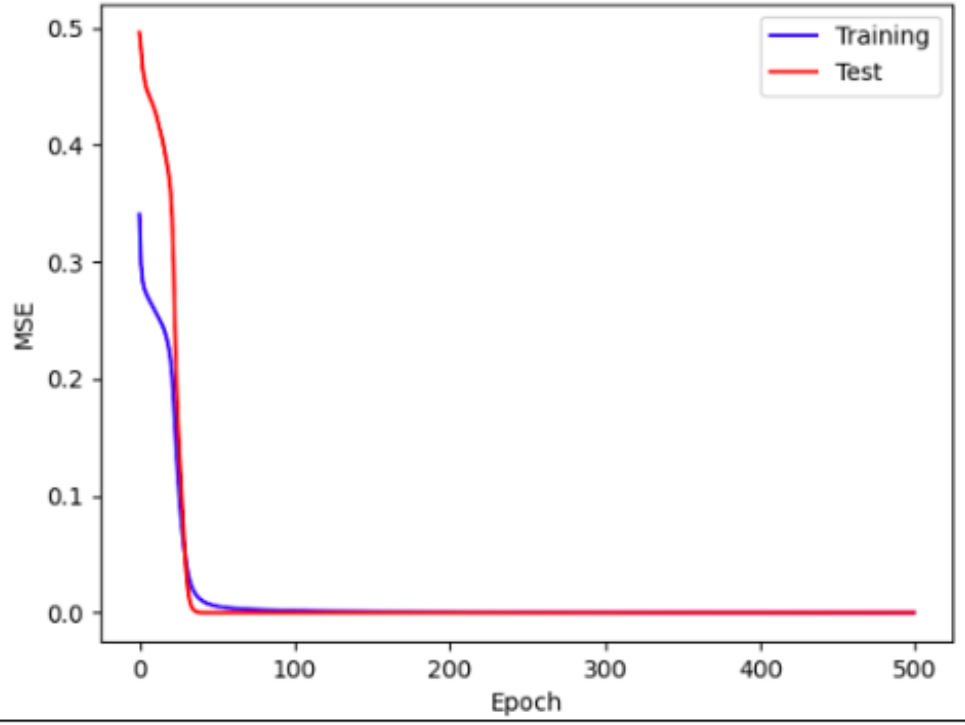
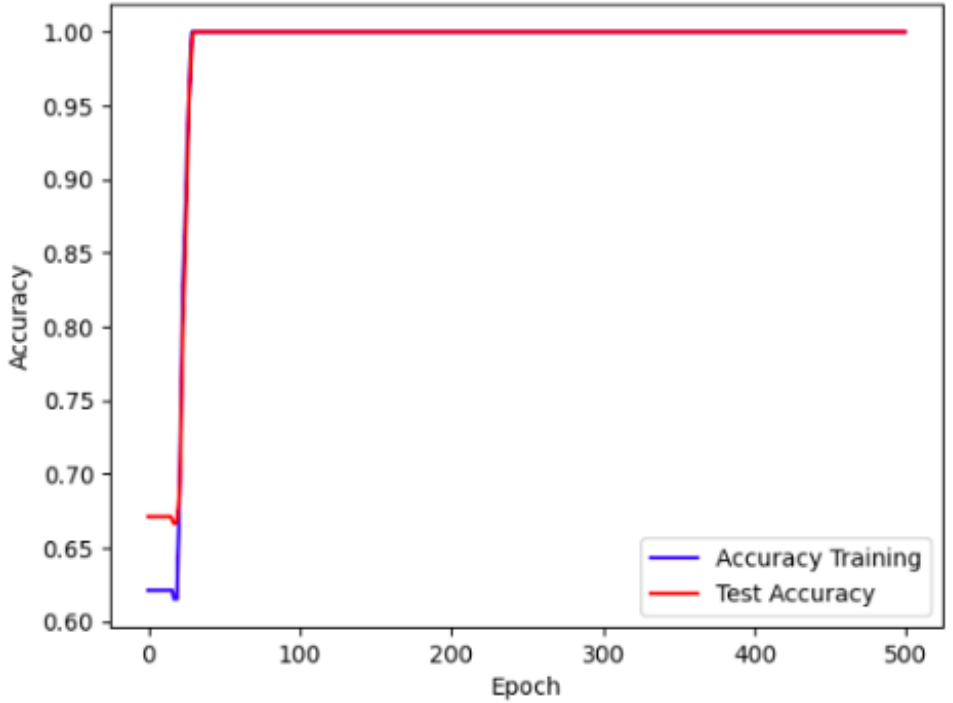
TASK	Hidden Units	Learning Rate	Lambda	MSE (TR/TS)	Accuracy (TR/TS)
MONK1	3	0.05	0	0.006632 / 0.0	100% / 100%
MONK2	4	0.4	0	0.000155 / 0.0	100% / 100%
MONK3	2	0.55	0	0.01626 / 0.06779	99.18% / 93.51%
MONK3 (reg.)	3	0.1	0.001	0.03467 / 0.02419	95.90% / 97.91%

Mean MSE and Accuracy over 5 trials

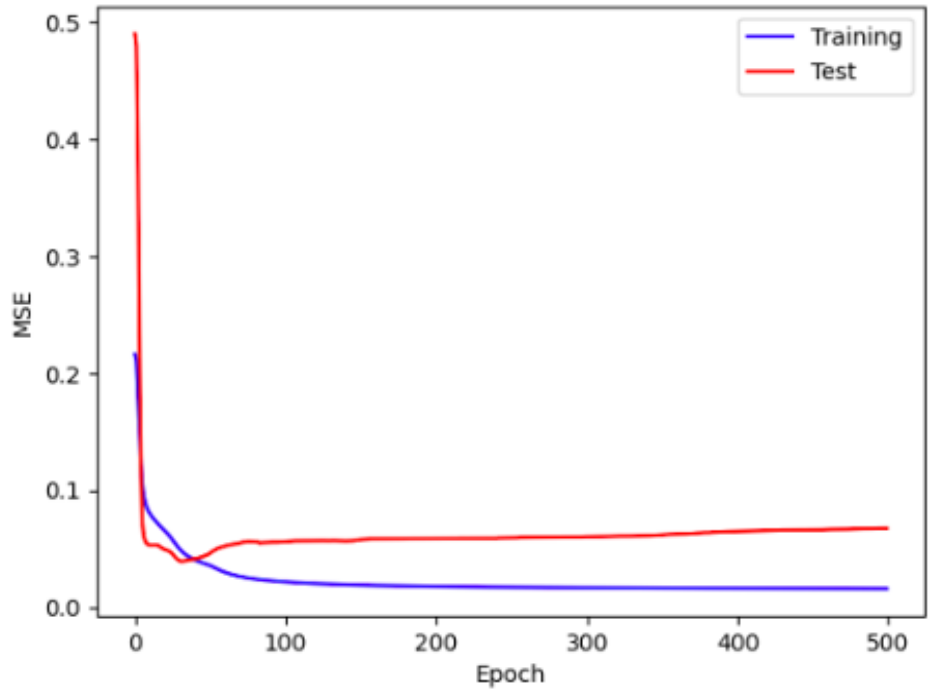
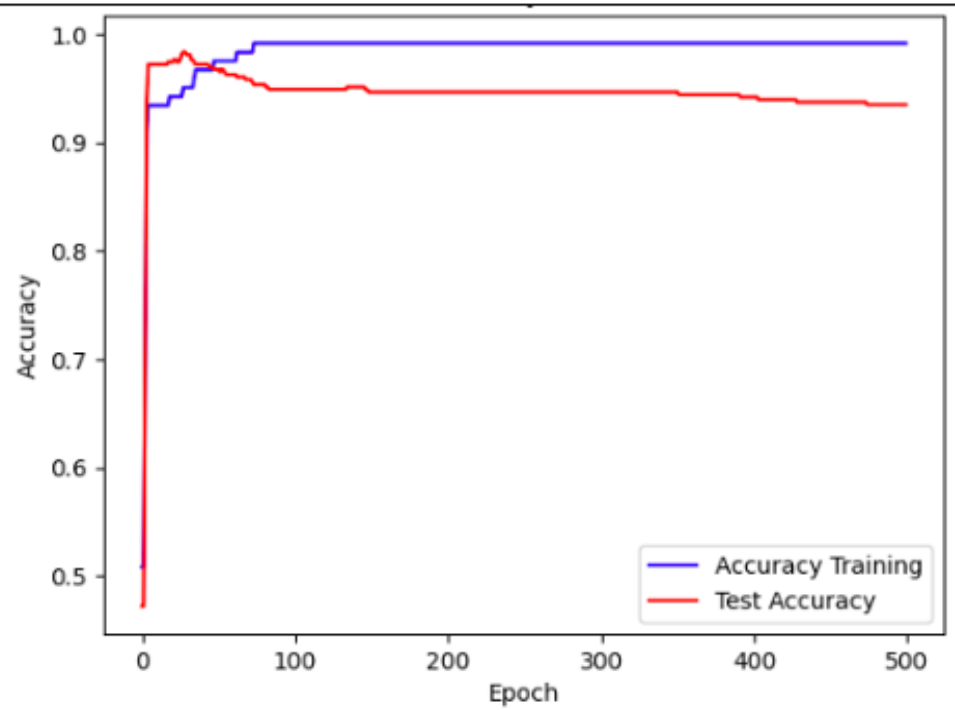
MONK RESULTS

TASK	MSE	ACCURACY %
MONK1		

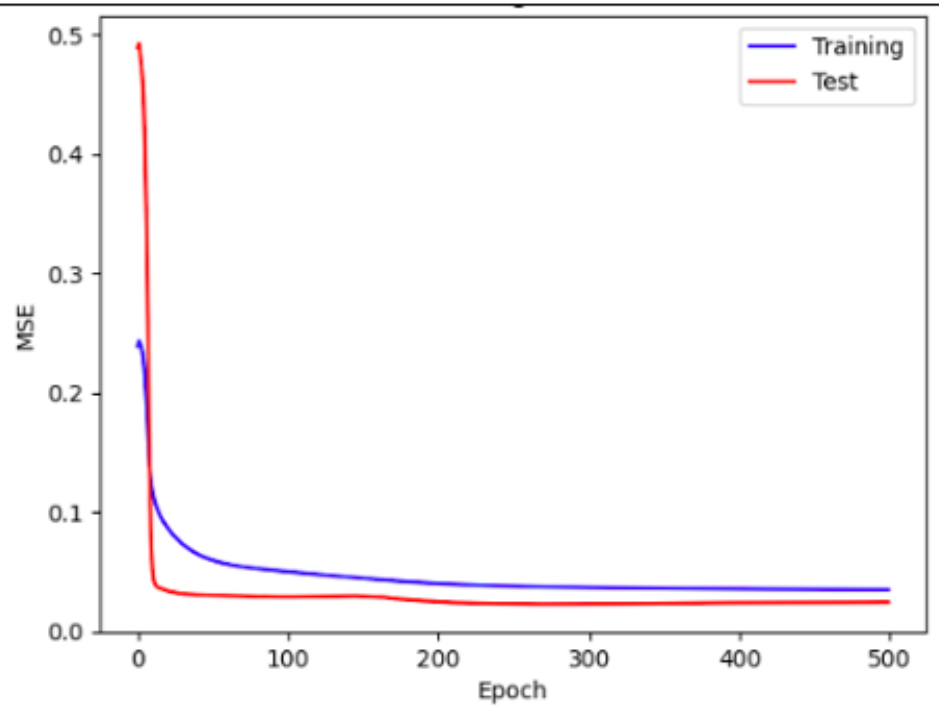
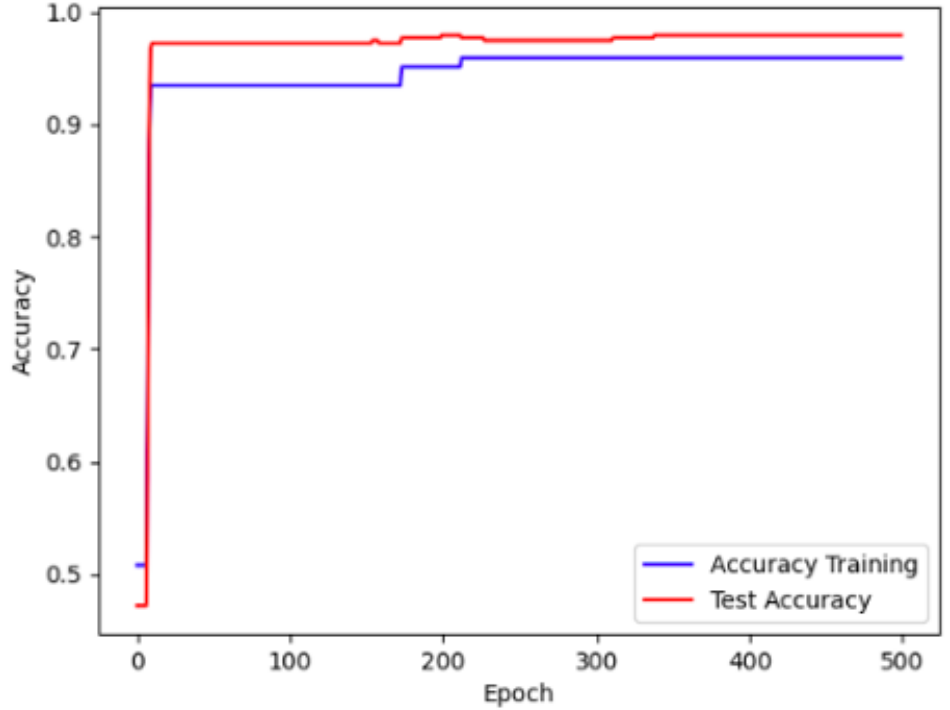
MONK RESULTS

TASK	MSE	ACCURACY %
MONK2		

MONK RESULTS

TASK	MSE	ACCURACY %
MONK3 no reg.		

MONK RESULTS

TASK	MSE	ACCURACY %
MONK3 reg.		

MONK RESULTS: LIST OF THE HYPERPARAMETERS

- ***hidden_size***: number of units of the hidden layer
- ***learning_rate***: learning rate (eta)
- ***epochs***: the number of epochs
- ***batch_size***: the mini-batch size (if 1 then online, else batch)
- ***momentum***: the momentum coefficent (alpha)
- ***lambda_reg***: the regularization coefficent (for L2)
- ***w_init_limit***: the range for the initial weights

CUP VALIDATION SCHEMA: DATA SPLITTING

- 80% Training+Validation=Development Set (800 examples)
  - K-Fold (K=5)
    - 4 Fold for Training (640 examples), 1 Fold for Validation (160 examples)
  - Hold-Out
    - 80% Training (640 examples), 20% Validation (160 examples)
- 20% Internal Test (200 examples)
- Retraining on Development Set

CUP VALIDATION SCHEMA: MODEL SELECTION

- Screening Phase
  - High number of hidden units (universal theorem)
  - Initially using a coarse grid search (Hold-Out Validation), then with a finer grid search based on the results obtained with the first one

COARSE

HYPERPARAMETER TYPE	RANGE
hidden_size	(40, 50) step 1
learning_rate	(0.05, 0.15) step 0.01
epochs	(400, 800) step 100
batch_size	[200, 400, 600, len_data]
momentum	(0.1, 0.2) step 0.01
lambda_reg	[0.000001, 0.00001, 0.0001]
w_init_limit	[[-0.5, 0.5], [-0.6, 0.6], [-0.7,0.7]]

FINE

HYPERPARAMETER TYPE	RANGE
hidden_size	[47]
learning_rate	(0.01, 0.1) step 0.001
epochs	(500, 600) step 100
batch_size	[200]
momentum	(0.1, 0.2) step 0.01
lambda_reg	[0.0001]
w_init_limit	[[-0.7,0.7]]

CUP VALIDATION SCHEMA: CUP RESULTS

- Coarse Grid Search with Hold Out
  - 10 hours using MSI 12th Gen Intel(R) Core(TM) i7-12650H 2.30 GHz
  - Number of combinations: 239580
  - Initial activation function: ReLU (hidden), Identity (output)
  - Patience for early stopping: 20
- With 5 best model we understand the range for the Fine Grid Search (combinations: 2000), and we:
  - Compare Hold-Out and K-fold
    - Similar results but K-Fold is better with smaller dataset
  - Compare ReLU / LeakyReLU / TanH on hidden layer
    - ReLU and Leaky ReLU seems better than TanH

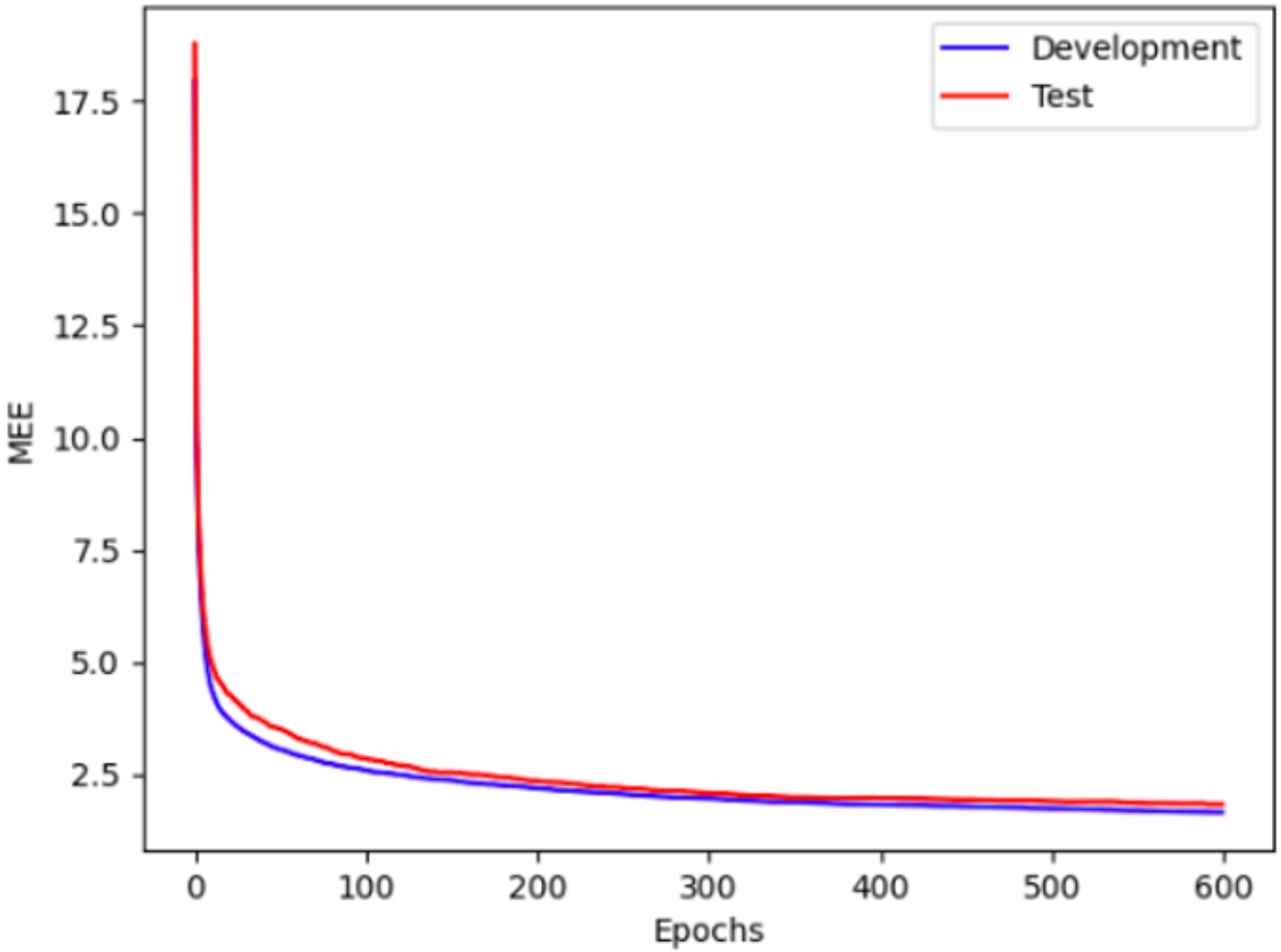
CUP RESULTS

- Comparison of different combinations of hyperparameters to check the differences and similarities of the results (plots and hyperparameters used are in the appendix)
- Final Model with K-Fold and ReLU (hidden)
  - Select best 10 model on Fine Grid Search, basing on validation error
  - Compare their plot
  - Compare their test error

CUP RESULTS

- *hidden_size*: 47
- *learning_rate*: 0.08
- *epochs*: 600
- *batch_size*: 200
- *momentum*: 0.11
- *lambda_reg*: 1e-05
- *w_init_limit*: [-0.7, 0.7]

- Development MEE
  - 1.6379876671343174
- Internal Test MEE
  - 1.8067147539347013
- Validation MEE
  - 1.788410259793571



DISCUSSION

- What we found significant / What we learned
  - Understand better the theoretical aspect creating the network from scratch
  - Understand the importance of validation and test part
  - How a small change on hyperparameter results in different solutions
  - How to avoid problems with small datasets
    - MONK: Learning rate small and momentum high
    - MONK: Online better than Batch
  - CUP: Using Regularization, still exploit Early Stopping (due to high chances of overfitting)

CONCLUSION

- Compute the predictions for blind test, and save on file "gradient_decent_ML-CUP23-TS.csv"
- According to our Internal Test, the expected loss on Blind Test is around 1.80671475 (MEE)

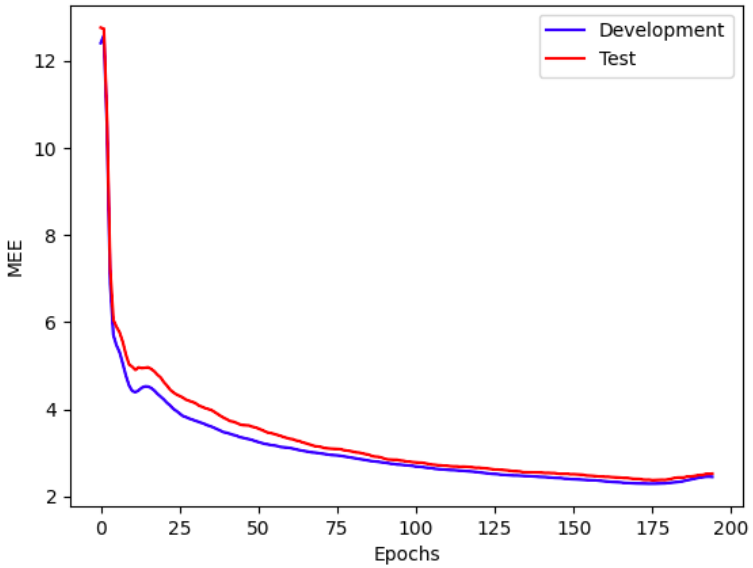
BIBLIOGRAPHY

- A. Micheli: "Derivation of the Back-propagation learning based algorithm"
- Monk Dataset
  - https://archive.ics.uci.edu/dataset/70/monk+s+problems
- A. Geron: "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow 3e: Concepts, Tools, and Techniques to Build Intelligent Systems"

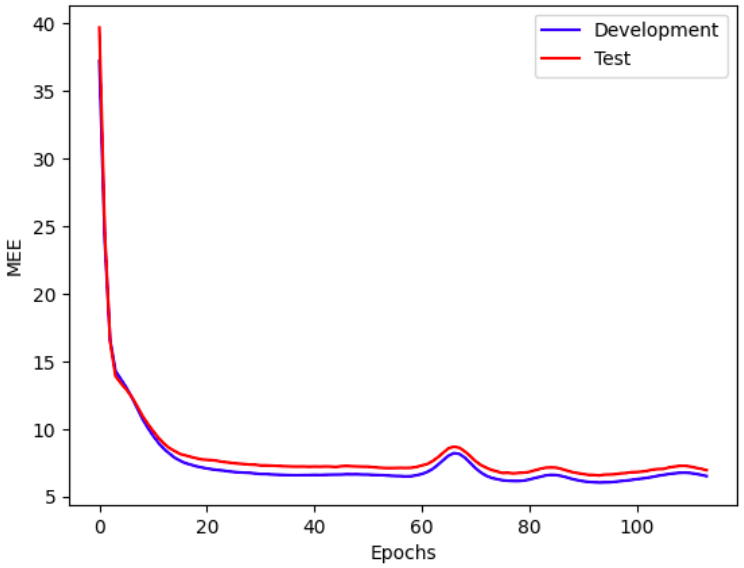
APPENDIX

MODEL	*hidden_size*	*learning_rate*	*epochs*	*batch_size*	*momentum*	*lambda_reg*	*w_init_limit*
1	100	0.1	600	200	0.1	0.0001	[-0.7, 0.7]
2	4	0.1	600	200	0.1	0.0001	[-0.7, 0.7]
3	30	0.9	600	200	0.1	0.0001	[-0.7, 0.7]
4	30	0.1	600	1	0.1	0.0001	[-0.7, 0.7]
5	30	0.1	600	200	0.9	0.0001	[-0.7, 0.7]
6	30	0.2	600	200	0.1	0.0001	[-0.1, 0.1]
7	30	0.1	600	16	0.1	0.001	[-0.5, 0.5]
8	30	0.1	600	4	0.1	0.0001	[-0.5, 0.5]
9	30	0.2	600	200	0.1	0.001	[-0.20, 0.20]
10	10	0.05	500	200	0.5	0.0001	[-0.1, 0.1]

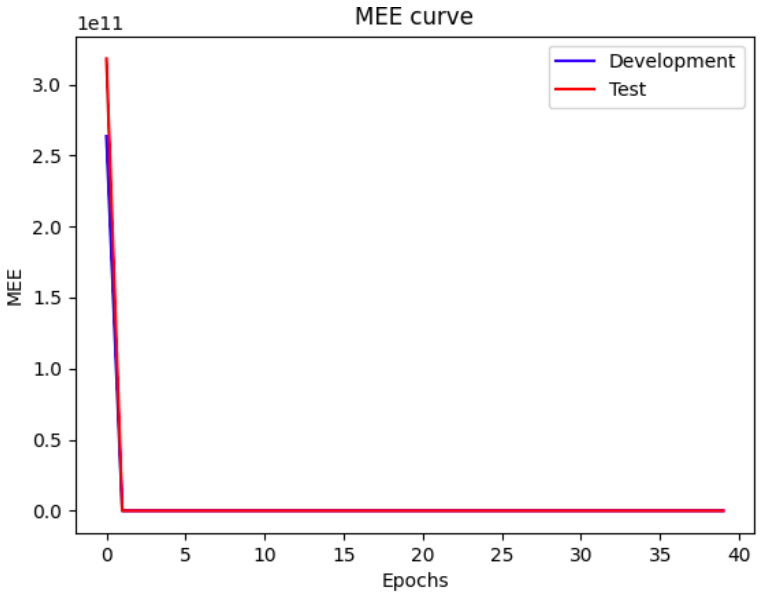
Model 1



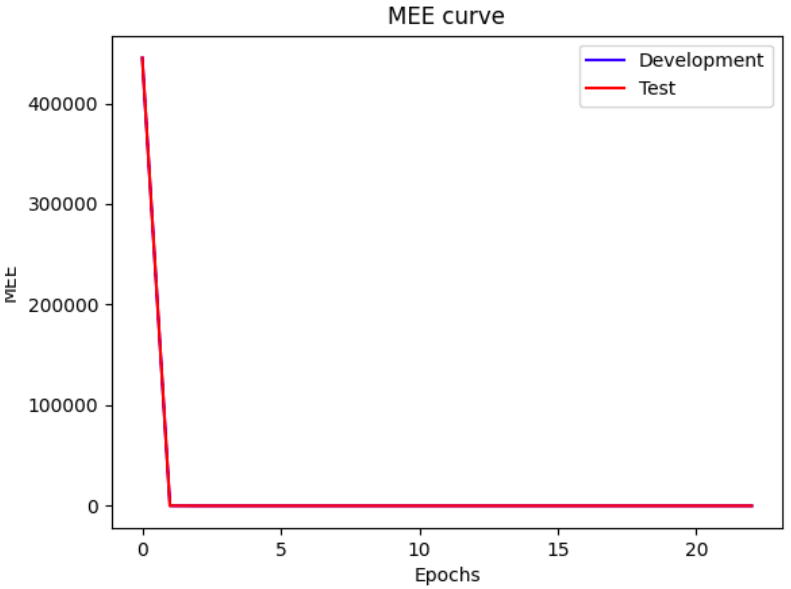
Model 2



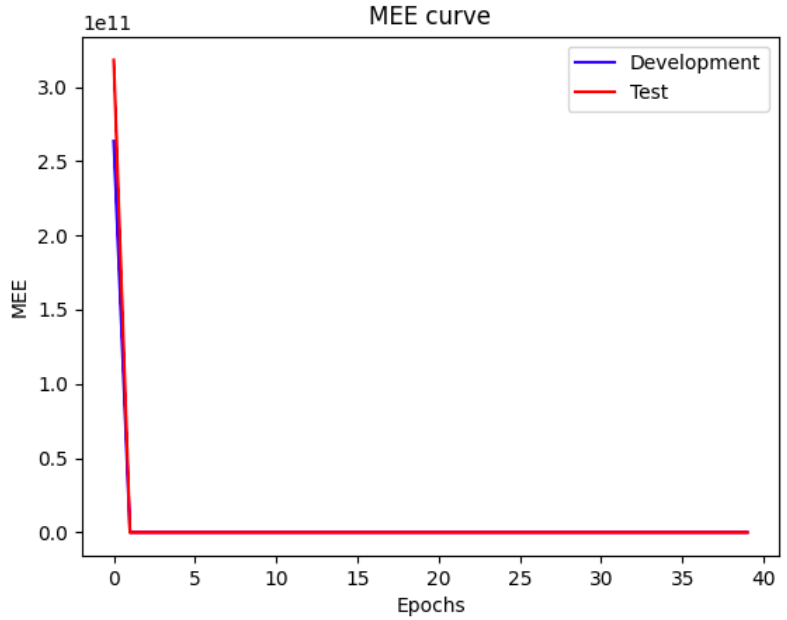
Model 3



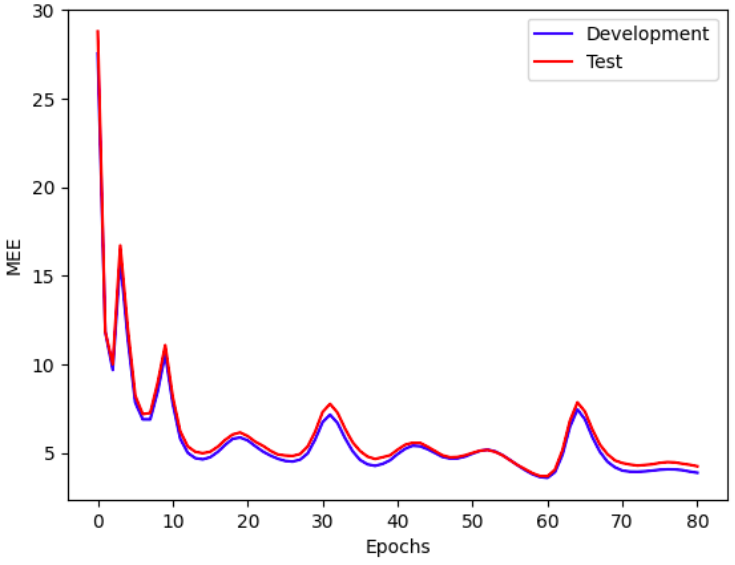
Model 4



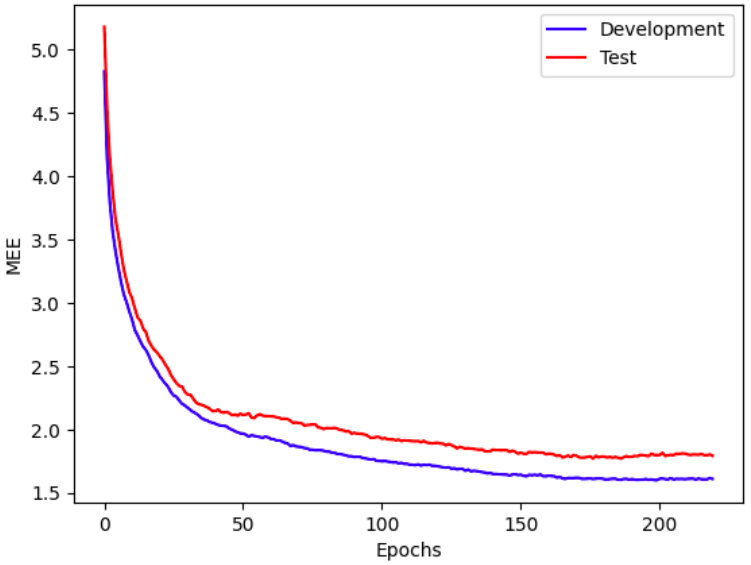
Model 5



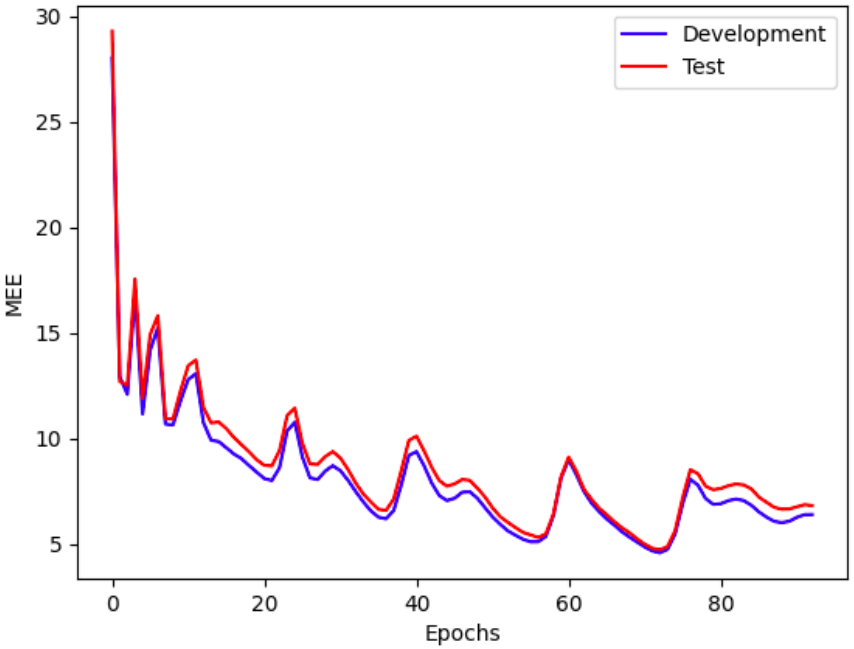
Model 6



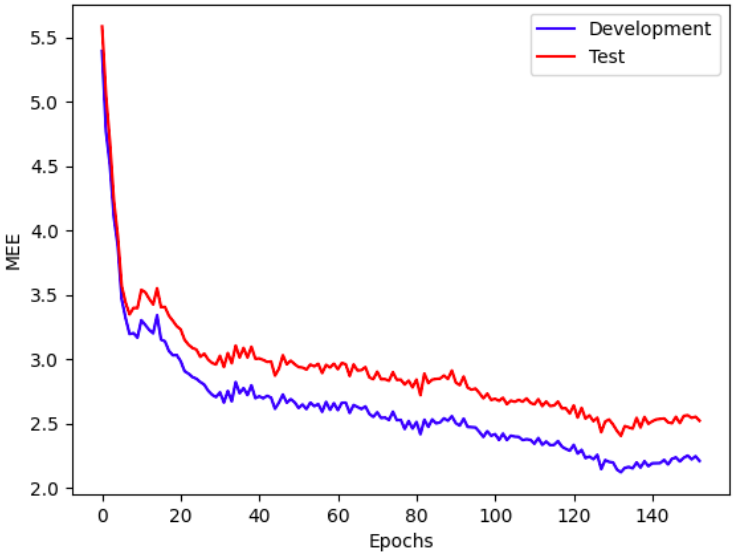
Model 7



Model 10



Model 8



Model 9