ANN implementation & Cascade Correlation analysis

MONK's Problems & ML-CUP25 Classification and Regression Tasks

Alessia Stocco, Cosimo Botticelli, Giulia Tumminello

PlusUltra

Digital Humanities, Artificial Intelligence, Physics of Complex Systems
a.stocco@studenti.unipi.it, c.botticelli1@studenti.unipi.it, g.tumminello@studenti.unipi.it

19/01/2026

Type of project: **A**

Introduction and objectives

This project presents the design and implementation of a neural network framework developed from scratch in Python which supports multiple training algorithms, regularization strategies, and validation schemes.
For both classification and regression problems different training approaches (standard backpropagation and a cascade-correlation–inspired method) were compared.
Results highlight the impact of architecture, regularization, and model selection strategies on generalization performance.

MAIN GOALS:

- To study the behavior and performance of feed-forward neural networks trained with different learning strategies.
- To analyze generalization behavior.
- To evaluate alternative training approaches beyond standard backpropagation.
- To analyze learning dynamics.

Software design & libraries

LANGUAGE AND LIBRARIES USED: Python 3.14, Numpy, Joblib, Matplotlib, Pandas, Graphviz.

The implementation of a custom neural network allowed full control over the network architecture, weight initializer and training algorithms and over loss functions and momentum.

Its main characteristics are:

- Fully-connected feed-forward neural network which allows for modular architecture.
- Support for classification and regression tasks.
- Ensemble methods were only adopted for the MONK classification tasks to reduce variance and improve robustness.
- Multiple independently trained networks were combined.
- Final predictions were obtained by averaging model outputs.

Architecture & implementation specifics

Single and multi-hidden-layer MLPs whose number of hidden units was selected via model selection.

- ACTIVATION FUNCTIONS: Sigmoid, Tanh.
- LOSS FUNCTIONS: Half MSE, MSE, MEE, MAE, Logcosh, Binary Crossentropy, Huber.
- TRAINING ALGORITHMS: Stochastic Gradient Descent (SGD), SGD with momentum, RPROP and QuickProp.
- REGULARIZATION: L2 (Tikhonov).
- OPTIMIZATION DETAILS: Mini-batch training, learning rate decay on validation plateau, early stopping based on validation error.

Implementation specifics for CC

- Number of hidden units: fixed at 20 and controlled by the max_hidden_units hyperparameter;
- Activation function: tanh to map inputs/outputs in the range [-1, 1] both in the hidden and output units;
- Training algorithms: Quickprop and Rprop to speed up convergence and Gradient Ascent to maximize the correlation score S;
- Learning mode: Full-Batch learning to ensure stable convergence statistics.
- Xavier initialization applied to weights to maintain variance across connections;
- L2 regularization to prevent overfitting;
- Stop conditions: when the validation error stops improving for a set num of epochs (patience) or The network stops adding new hidden units when the error is low enough (tolerance) or the maximum size is reached.

Preprocessing & validation schema

INPUT NORMALIZATION:

- MONK: one-hot encoding.
- CUP: min–max normalization.

OUTPUT NORMALIZATION:

- CUP: Targets scaled to symmetric ranges compatible with Tanh.

Denormalization was applied before computing final performance metrics.

Dataset shuffle both for MONK's problems and CUP.

MONK DATASET:

- Predefined training and test sets.

Test set used only once for final evaluation.

CUP DATASET:

- Explicit split into Training / Validation / Internal Test sets.

Validation set used for model selection and internal test set used for final assessment before blind test.

Validation schema & Screening phase for CC

We implemented a Hold-Out validation strategy managed by `data_manipulation.py`. The dataset was partitioned into a Training Set (85%) for weight optimization and a Validation Set (15%) specifically used to guide the constructive early stopping mechanism, while keeping an Internal Test Set completely isolated for the final model assessment.

The preliminary screening phase revealed that the constructive nature of the Cascade algorithm drastically alters the loss landscape compared to static topologies. Consequently, standard patience settings proved inadequate, requiring a calibrated two-stage search protocol (Coarse and Fine) to identify the optimal balance between learning rates and tolerance thresholds.

Hyperparameter space of CC

Model selection was conducted via a parallelized Grid Search implemented in `cascade_grid_search.py`. We explored a comprehensive hyperparameter space including learning rate, patience, tolerance, and the regularization factor.

The architecture dynamically toggled between QuickProp and RProp update rules to maximize convergence speed. The search results highlighted that while deeper networks reduced training error, they required significant L2 regularization to maintain generalization capability on the validation set.

SEARCH SPACE					
Learning rate	Patience	Tolerance	Max hidden units	Algorithm	Epochs
0.001	50	0.001	20	QuickProp	2000
0.005					
0.01	80	0.0001		Rprop	
0.05					

Table 1. Parameters set in the grid search for multi-output regression task.

BEST CONFIGURATIONS FOUND					
Rank	Algorithm	Hidden units	TR MEE	VL MEE	TS MEE
1	QuickProp	20	16.782978	23.525320	24.716925
2	QuickProp	20	92.172781	90.402928	90.296196
3	QuickProp	20	92.134299	90.756010	90.923462

Table 2. Ranking of the three best configurations found for CUP regression task.

Final model selection

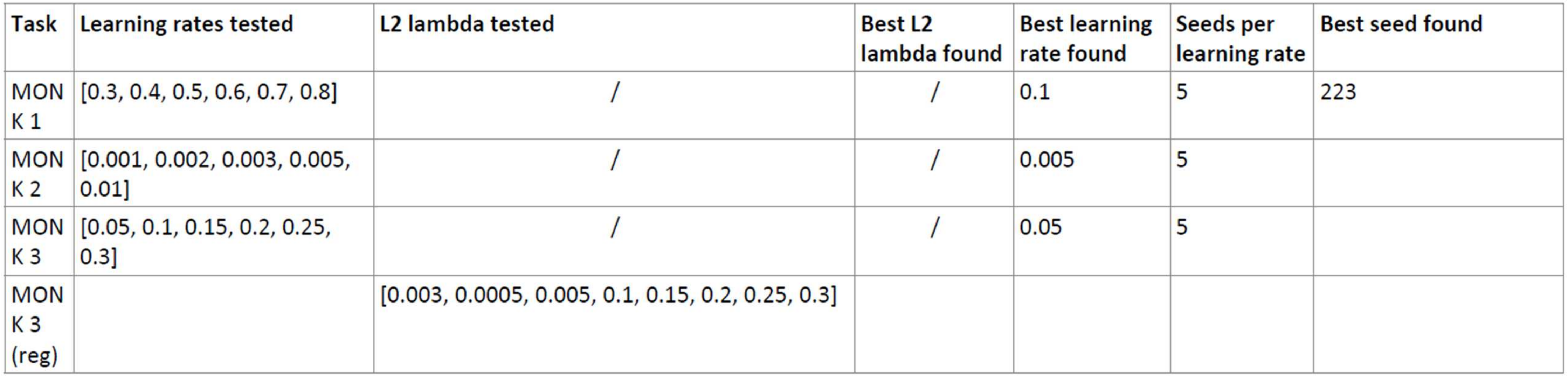
Model selection was made through GRID SEARCH which allowed for exhaustive exploration of learning rate, L2 regularization parameter, momentum and network architecture.
Average validation performance was used for comparison.

SELECTION CRITERION:
Lowest validation loss, stable learning curve and limited gap between training and validation error.

FINAL MODEL CHARACTERISTICS:

- Multi-layer architecture.
- Tanh activation.
- SGD with momentum.
- L2 regularization.
- Early stopping.

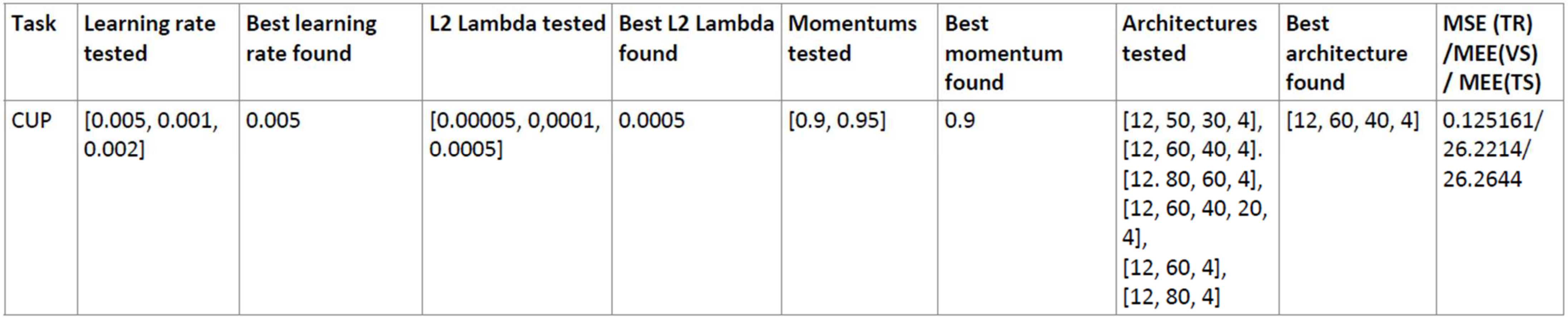
ANN grid search on Monk problems



Task	Learning rates tested	L2 lambda tested	Best L2 lambda found	Best learning rate found	Seeds per learning rate	Best seed found
MONK 1	[0.3, 0.4, 0.5, 0.6, 0.7, 0.8]	/	/	0.1	5	223
MONK 2	[0.001, 0.002, 0.003, 0.005, 0.01]	/	/	0.005	5	
MONK 3	[0.05, 0.1, 0.15, 0.2, 0.25, 0.3]	/	/	0.05	5	
MONK 3 (reg)		[0.003, 0.0005, 0.005, 0.1, 0.15, 0.2, 0.25, 0.3]				

Table 3. Grid search for the Monk tasks.

ANN grid search on CUP problem



Task	Learning rate tested	Best learning rate found	L2 Lambda tested	Best L2 Lambda found	Momentums tested	Best momentum found	Architectures tested	Best architecture found	MSE (TR) /MEE(VS) / MEE(TS)
CUP	[0.005, 0.001, 0.002]	0.005	[0.00005, 0,0001, 0.0005]	0.0005	[0.9, 0.95]	0.9	[12, 50, 30, 4], [12, 60, 40, 4]. [12. 80, 60, 4], [12, 60, 40, 20, 4], [12, 60, 4], [12, 80, 4]	[12, 60, 40, 4]	0.125161/ 26.2214/ 26.2644

Table 4. Grid search for the CUP task.

Final model assessment for CC

The final model was selected by isolating the configuration with the lowest Mean Euclidean Error (MEE) on the Validation set, prioritizing architectures with stable learning curves to ensure robust generalization.

FINAL MODEL HYPERPARAMETERS					
Learning rate	Patience	Tolerance	Max hidden units	Algorithm	#Epochs, Momentum, Mu
0.04000000000000001	70	0.001	20	QuickProp	2000, 0.0, 1.5

MEE VALUES (performance metrics calculated on the original scale)		
MEE TR	MEE VL	MEE TS
16.953521	22.744240	23.938295

Estimation of the training computing time: 16.3 min
HW resources: CPU: AMD64 Family 23 Model 24 Stepping 1, Authentic AMD Ryzen5, Cores: 8, OS: Windows 11

ANN – Monk results

Task	Architecture	Loss type	Training algorithm	Momentum	Max epochs	Batch size	patience	Actual epochs trained	L2 lambda	Decay	Activation function	Weight initializer	Half MSE (TR/ VS/TS)	Accuracy (TR /VS/TS) (%)
MONK 1	[17, 5, 1]	Half MSE	SGD	0.9	1500	1 (SGD online)	50	242	0.0	0.9	Sigmoid	Xavier	0.000364/ 0.000364/ 0.000358	100% /100% / 100%
MONK 2	[17, 8, 1]	Half MSE	SGD	0.9	2000	1 (SGD online)	50	235	0.0	0.9	Sigmoid	Xavier	0.000812/ 0.000812/ 0.000788	100% /100% / 100%
MONK 3	[17, 6, 1]	Half MSE	SGD	0.85	2000	1 (SGD online)	50	235	0.0	0.9	Sigmoid	Xavier		97.26% / 97.96% / 98.84 %
MONK 3 (reg)	[17, 6, 1]	Half MSE	SGD	0.85	2000	1 (SGD online)	50		/					

Table 5. Average predictions results for the Monk's task.

Task	Half MSE (TR/VS/TS)	Accuracy (TR/VS/TS) (%)
MONK 1	0.016839 / 0.019390 / 0.029984	98.8372% / 94.7368% / 93.5185%
MONK 2	[0.001, 0.002, 0.003, 0.005, 0.01]	5
MONK 3	[0.003, 0.0005, 0.005, 0.1, 0.15, 0.2, 0.25, 0.3]	5

Table 6. Ensemble results for the Monk's task.

ANN – Monk learning curves

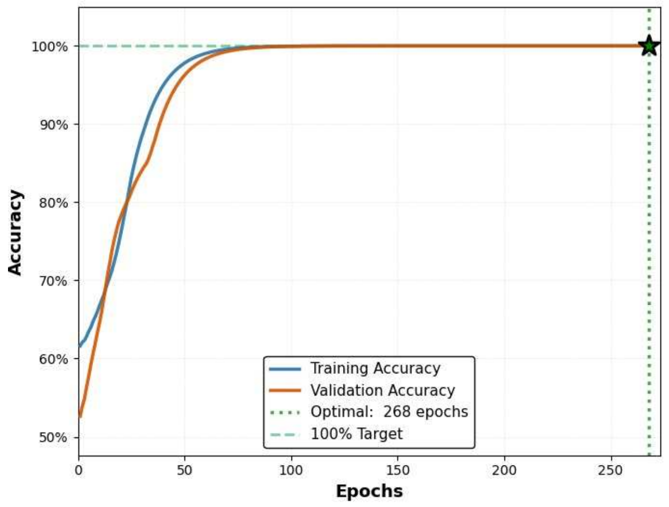
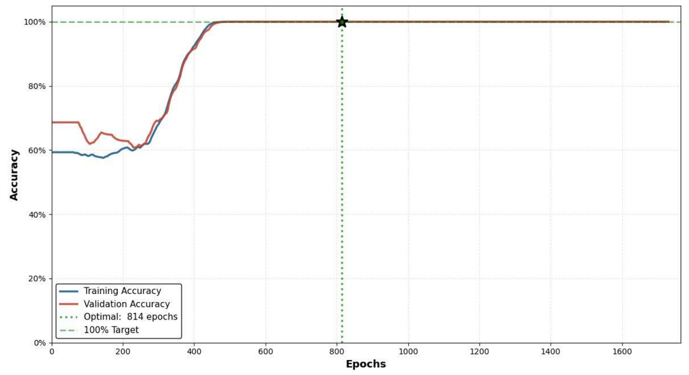
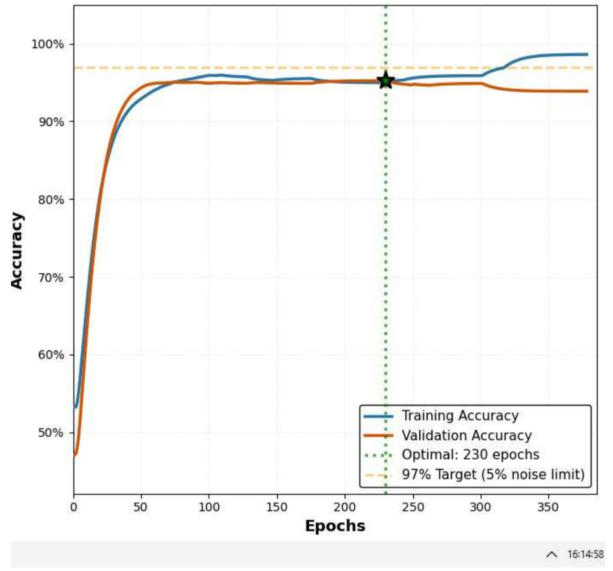
MONK 1	MONK 2	MONK 3
		
100% Accuracy	97% Accuracy	97% Accuracy

Table 7. Plots of the accuracy obtained on Monk's problems.

ANN – Monk confusion matrices

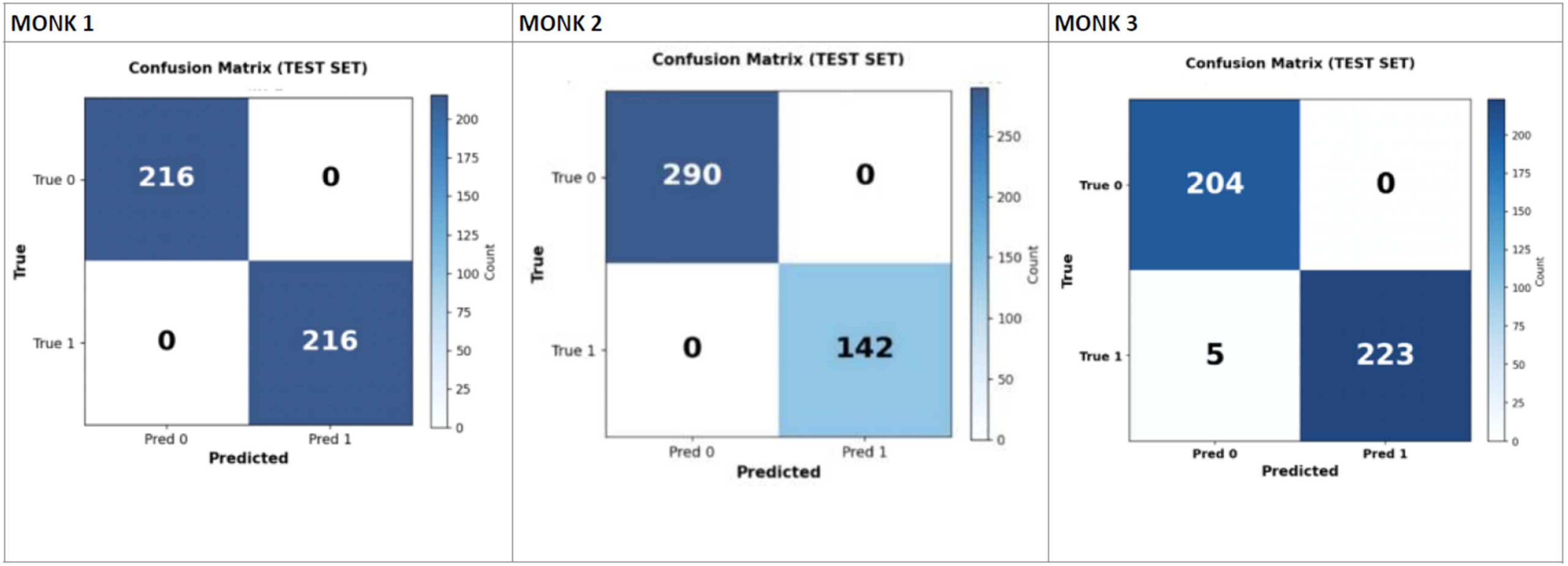


Figure 1. Confusion matrixes for the three MONK's benchmarks.

CC – Monk results

TASKS	**#eta, patience, tolerance, max units, algorithm, l2_lambda, epochs, mom, (mu, eta plus, eta minus)**	**MSE (TR/TS)**	**ACCURACY (TR/TS)**
MONK 1	0.2, 60, 0.001, 6, Rprop, 0.0, 2000, 0.0, (1.2, 0.5)	0.000290/0.001123	100%/100%
MONK 2	0.24, 60, 0.001, 6, QuickProp, 0.001, 2000, 0.0 (1.5)	0.000150/0.000160	100%/100%
MONK 3	0.01, 30, 0.03, 2, 0.1, QuickProp, 2000, 0.0 (1.5)	0.062535/0.054864	91%/96%

Table 8. Average prediction results obtained for the MONK's tasks.

CC - Monk-1 learning curves

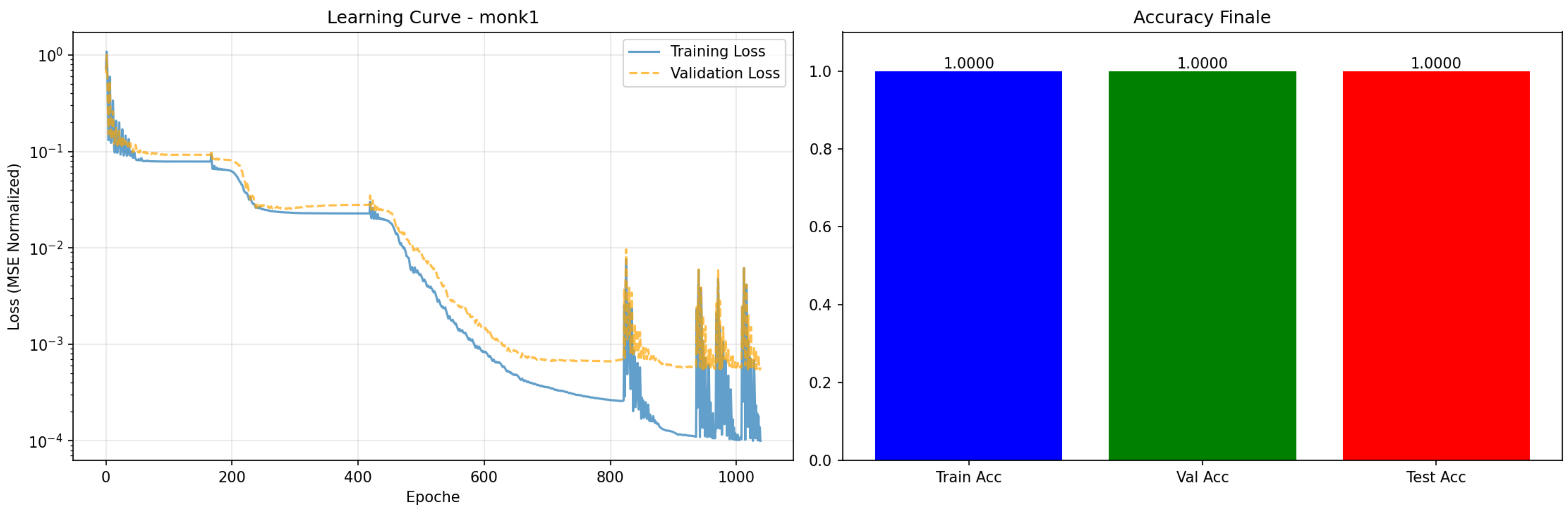


Figure 2. Plot of the Learning Curve and accuracy obtained on Monk 1 with Cascade Correlation.

CC - Monk-2 learning curves

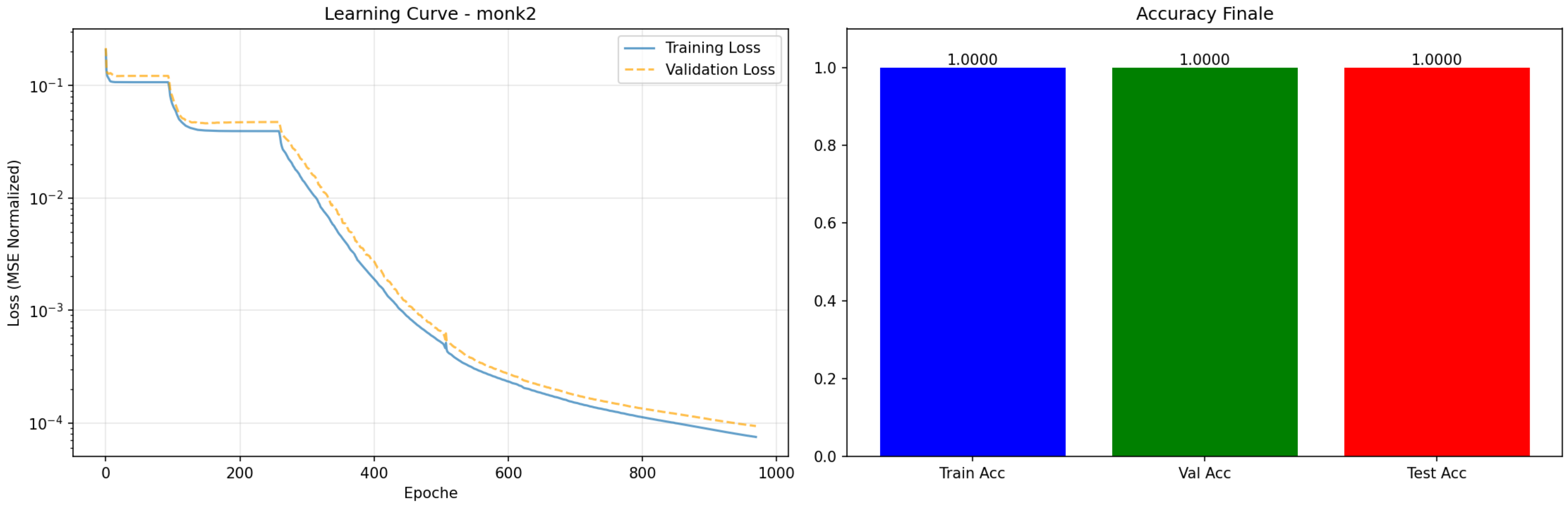


Figure 3. Plot of the Learning Curve and accuracy obtained on Monk 2 with Cascade Correlation.

CC - Monk-3 learning curves

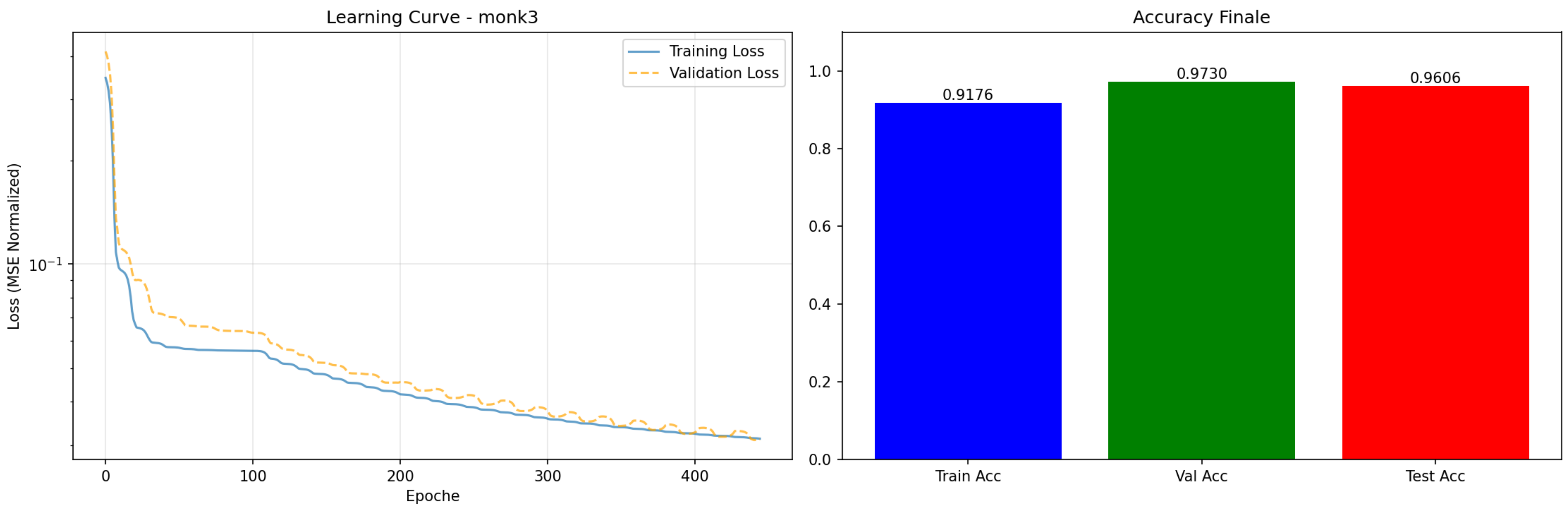
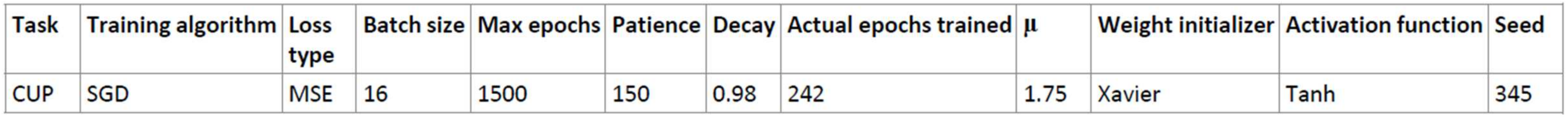


Figure 4. Plot of the Learning Curve and accuracy obtained on Monk 3 with Cascade Correlation.

ANN – CUP results



Task	Training algorithm	Loss type	Batch size	Max epochs	Patience	Decay	Actual epochs trained	μ	Weight initializer	Activation function	Seed
CUP	SGD	MSE	16	1500	150	0.98	242	1.75	Xavier	Tanh	345

Table 9. Average predictions results for the CUP's task.

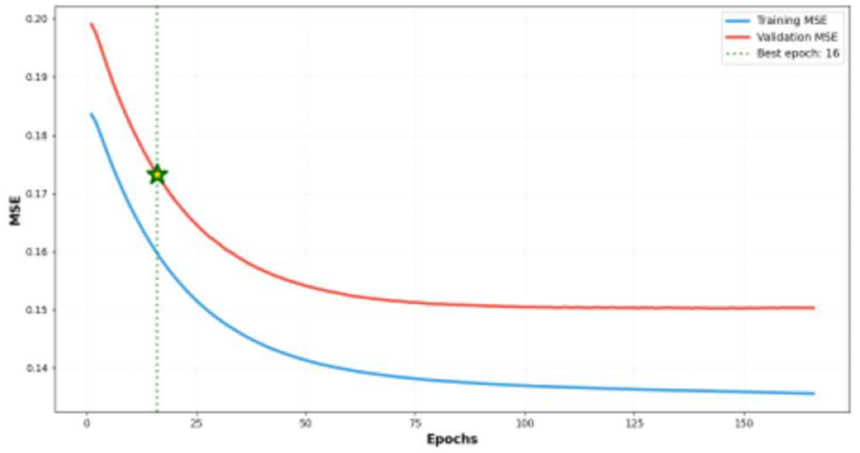
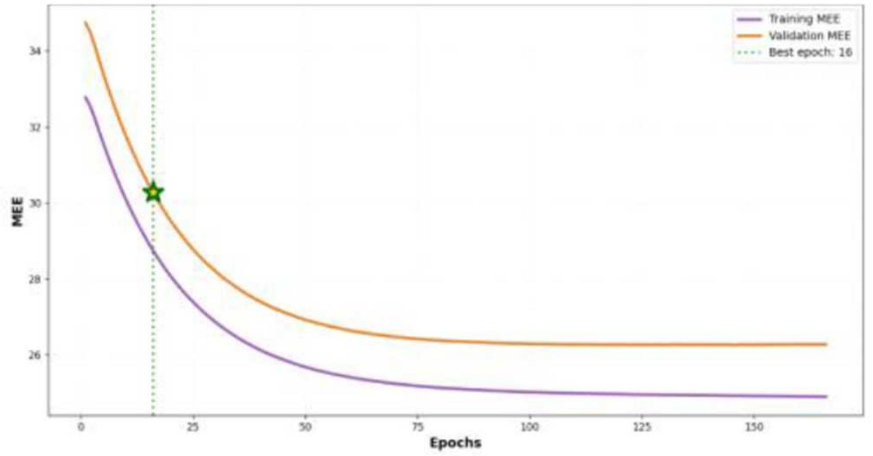
MSE	MEE
	

Figure 5. Plot of MSE loss (TR/VS), MEE loss (TR/VS) the CUP's task.

CC - Final model learning curve

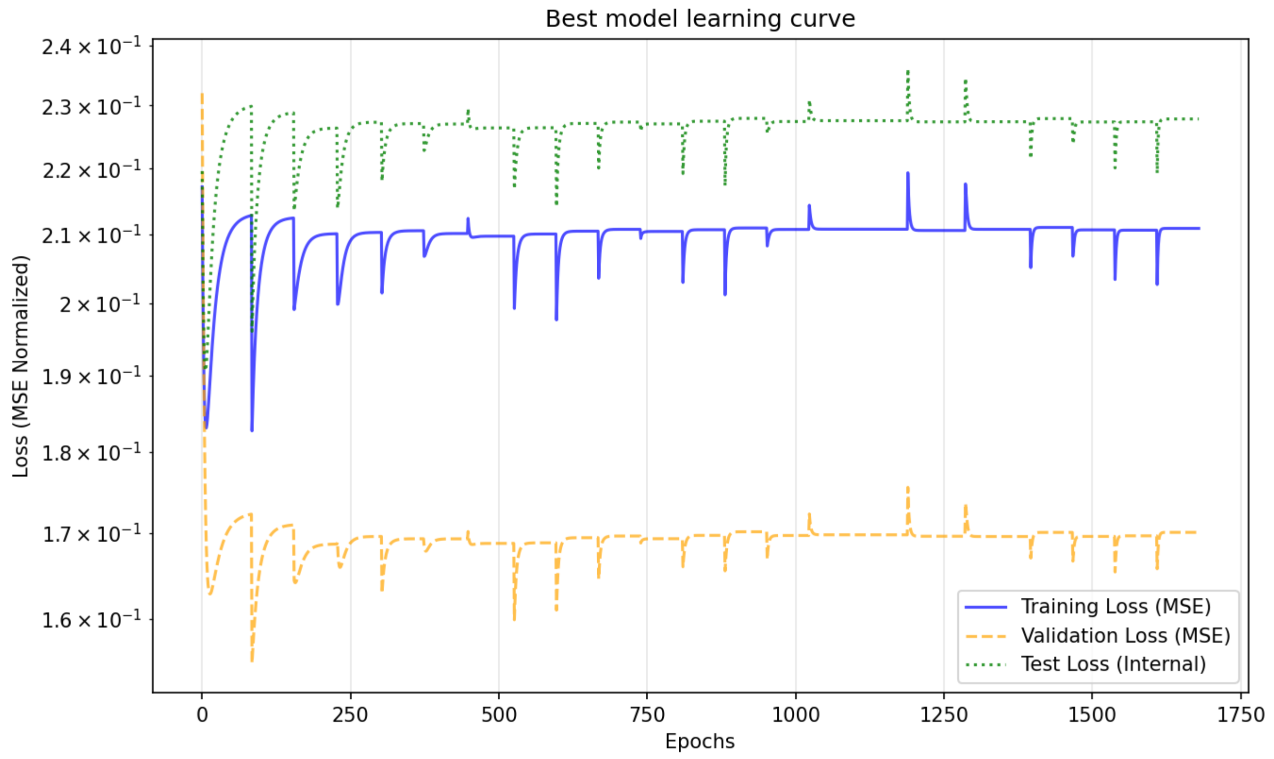


Figure 6. Learning curves obtained on CUP regression task with Cascade Correlation.

Comparison of training approaches

Standard backpropagation (SGD + momentum):

- Stable convergence.
- Strong baseline performance.

Cascade-correlation–inspired approach:

- Different convergence dynamics.
- Competitive validation performance.
- Higher training cost in many configurations.

Both approaches achieve comparable generalization when properly regularized.

Discussion and conclusions

Regularization is essential for stable generalization on CUP.
Larger architectures improve expressiveness but increase overfitting risk.
Random grid search explores the hyperparameter space more efficiently than exhaustive grid search.
The Cascade Correlation approach identified regularization as the single most critical factor for the Cascade-Correlation architecture on the CUP dataset; without it, the aggressive unit-addition strategy tends to overfit the training noise. Furthermore, while RProp demonstrated faster convergence in early stages, QuickProp provided superior stability as the network depth increased.

Blind Test Results: PlusUltra_ML-CUP25-TS.csv

Bibliography

Barron, J. T. (2019). A General and Adaptive Robust Loss Function. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4331-4340.

Borchani, H., Varando, G., Bielza, C., & Larrañaga, P. (2015). A survey on multi-output regression. *WIREs Data Mining and Knowledge Discovery*, 5(5), 216-233.

Elsebach, R. (1994). Evaluation of forecasts in AR models with outliers. *OR Spectrum*, 16, 41-45.

Fahlman, S. E. (1988). An Empirical Study of Learning Speed in Back-Propagation Networks. *Technical Report CMU-CS-88-1623*, Carnegie Mellon University.

Fahlman, S. E., & Lebiere, C. (1990). *The Cascade-Correlation Learning Architecture*. Advances in Neural Information Processing Systems 2 (NIPS 1989), pp. 524-532. Morgan Kaufmann.

Ghosh, A., Kumar, H., & Sastry, P. S. (2017). Robust Loss Functions under Label Noise for Deep Neural Networks.

Huber, P. J. (1964). Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1), 73-101.

Qi, J., Du, J., Siniscalchi, S. M., Ma, X., & Lee, C.-H. (2020). On Mean Absolute Error for Deep Neural Network Based Vector-to-Vector Regression.

Bibliography

Riedmiller, M., & Braun, H. (1993). A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. *IEEE International Conference on Neural Networks (ICNN)*, pp. 586-591.
Sun, Q., Zhou, W.-X., & Fan, J. (2020). Adaptive Huber Regression. *Journal of the American Statistical Association*, 115(529), 254-265.
Volarić, T., Ljubić, H., Vasić, D., & Rozić, R. (2025). Should MSE Remain the Default Benchmarking Loss Function for Long-Term Time Series Forecasting? *SN Computer Science*, 6:804.
Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 785–794.
Grover, A., & Ermon, S. (2019). Uncertainty Autoencoders: Learning Compressed Representations via Variational Information Maximization. *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics.*