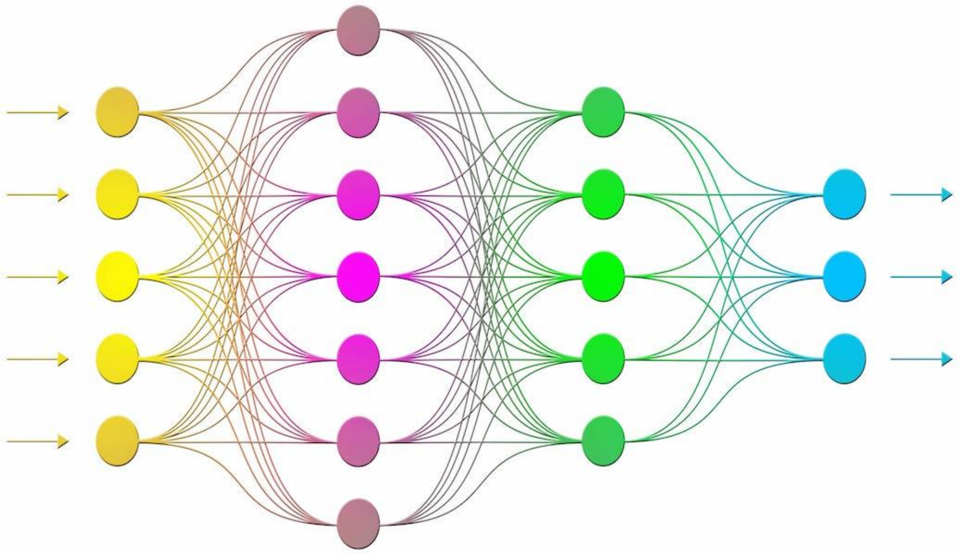
Università di Pisa

Dipartimento di Informatica

ML 2025 Project



Authors: Andrea Marcheschi (a.marcheschi6@studenti.unipi.it)
Luca Nasini (l.nasini@studenti.unipi.it)
Simone Passera (s.passera@studenti.unipi.it)

Team name: SimoLucaAndre

Master degree:

Date: 07/01/2026

Type of project: A

SUMMARY

Introduction & objectives

What we want to do:

Create, from scratch, a package to build a general neural network that can predict, with sufficient small error, new data target

Neural Network with:

- Early stopping, regularization, (Nesterov) momentum
- (Stochastic) Gradient Descent and Backpropagation algorithms
- 4 type of initialization (random, fan-in, xavier, he)
- Model validation and assessment (both with holdout, k-fold or leave-one-out)

Method - 1 (what is in the package)

Package scripts:

- main.py;
- model.py;
- model_selection.py;
- data_loaders.py;
- utils.py;
- activations.py;
- losses.py

External package used:

- numpy (for vectors);
- pandas (for reading .csv);
- matplotlib (for plots);
- pickle (for model savings)

Method – 2 (code description)

Philosophy of «model.py» code

Create a NeuralNetwork object with (dense) NeuronLayer (made of Neuron) with same activation functions. All Neuron have a bias and a list of weights.
After the weights are initialized(and biases) all layers (hiddens and output), do the training until early stopping conditions are satisfied. Training can be batch or online (or alse mini-batch).
The NeuralNetwork class permits to plot the learning curves and saves the entire neural network on a .pkl file

Philosophy of «model_selection.py» code

First select the type of search, and types of model selection/assessment.
If model selection is k-fold split data in fold else respect the given proprortions.
Then choose the hyperparameters from the search and choose the best model from lower (average) loss. Retrain the model with tr+vl data. Evaluate the model with ts data. Then you can save the entire run (best model config, weights, biases, etc.).

Method – 3 (necessary features)

Activation functions:

- linear;
- sigmoid;
- tanh;
- (leaky)ReLU;
- softplus

Loss functions:

- MSE;
- MEE;
- binary cross-entropy

Early stopping conditions:

Monitor: «val_loss»:	reset patience if new loss is less than (best loss - ε)
Monitor: «val_accuracy»:	reset patience if new accuracy is greater than (best accuracy + ε)
Monitor: «train_loss»:	set patience at 0 if new loss lesser than target loss (only for retraining)

Method – 4 (necessary features)

Momentum (α):

$$\Delta w_{tu} = \eta \delta_t o_u + \alpha \Delta \mathrm{w}_{\mathrm{old,tu}}$$

Nesterov: add α-term before computing the deltas and update weights unless this term

Regularization (λ):

$$w_{new,tu} = w_{tu} + \Delta w_{tu} - \lambda w_{tu}$$

Early stopping conditions:

Monitor: «val_loss»:	reset patience if new loss is less than (best loss - ε)
Monitor: «val_accuracy»:	reset patience if new accuracy is greater than (best accuracy + ε)
Monitor: «train_loss»:	set patience at 0 if new loss lesser than target loss (only for retraining)

Method – 3 (model novelties)

Nesterov momentum

Require: Learning rate ϵ, momentum α.
Require: Initial parameter $\boldsymbol{\theta}$, inizial velocity $\boldsymbol{v}$.
while stopping criterion not met **do**

Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ with corresponding labels $\mathbf{y}^{(i)}$.

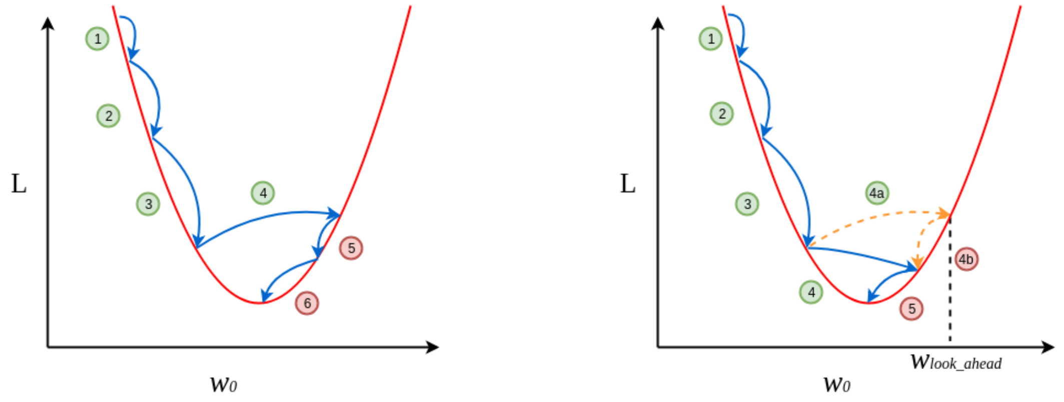
Apply interim update: $\widetilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta} + \alpha\boldsymbol{v}$

Compute gradient (at interim point): $\boldsymbol{g} \leftarrow \frac{1}{m}\nabla_{\widetilde{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \widetilde{\boldsymbol{\theta}}), \boldsymbol{y}^{(i)})$

Compute velocity update: $\boldsymbol{v} \leftarrow \alpha\boldsymbol{v} - \epsilon\boldsymbol{g}$

Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$

end while



(a) Momentum-Based Gradient Descent

(b) Nesterov Accelerated Gradient Descent

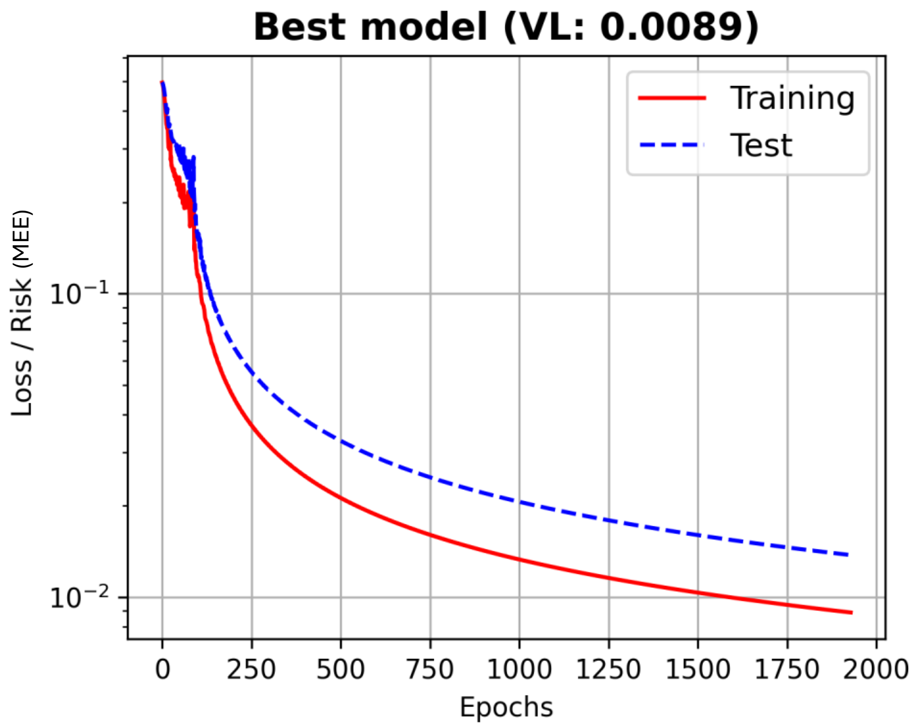
$\bigcirc \Longrightarrow \frac{\partial L}{\partial w_0} = \frac{Negative(-)}{Positive(+)}$ $\bigcirc \Longrightarrow \frac{\partial L}{\partial w_0} = \frac{Negative(-)}{Negative(-)}$

MONK results

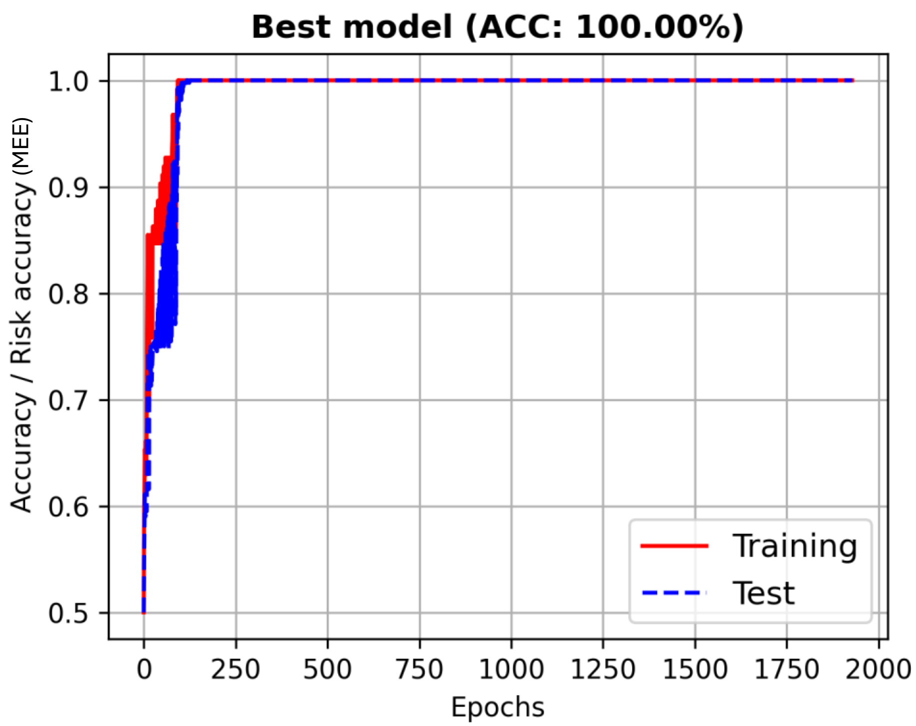
Input units: 17 (from one-hot-encoding)
Output units: 1
Activation funtion: ReLU (hidden layer), sigmoid (output unit)
Initialization: fan-in

Task	Units	η	λ	α	MEE (TR/TS)	Accuracy (TR/TS)
MONK1	4	0.0452	-	-	0.0089 / 0.0138	100% / 100%
MONK2	4	0.0480	-	-	0.0071 / 0.0076	100% / 100%
MONK3 (no reg.)	5	0.0143	-	-	0.0282 / 0.0779	97.54% / 92.13%
MONK3	4	0.0172	0.0011	-	0.0728 / 0.1002	99.18% / 95.37%

MONK 1 - plots

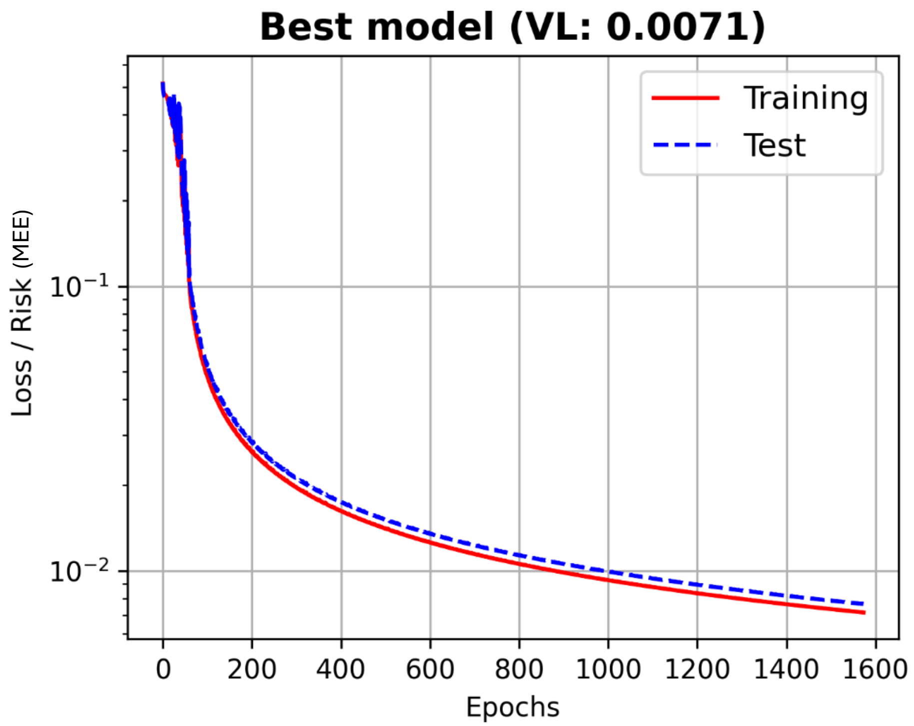


Best model (VL: 0.0089)
Training
Test
Loss / Risk (MEE)
10^{-1}
10^{-2}
0
250
500
750
1000
1250
1500
1750
2000
Epochs

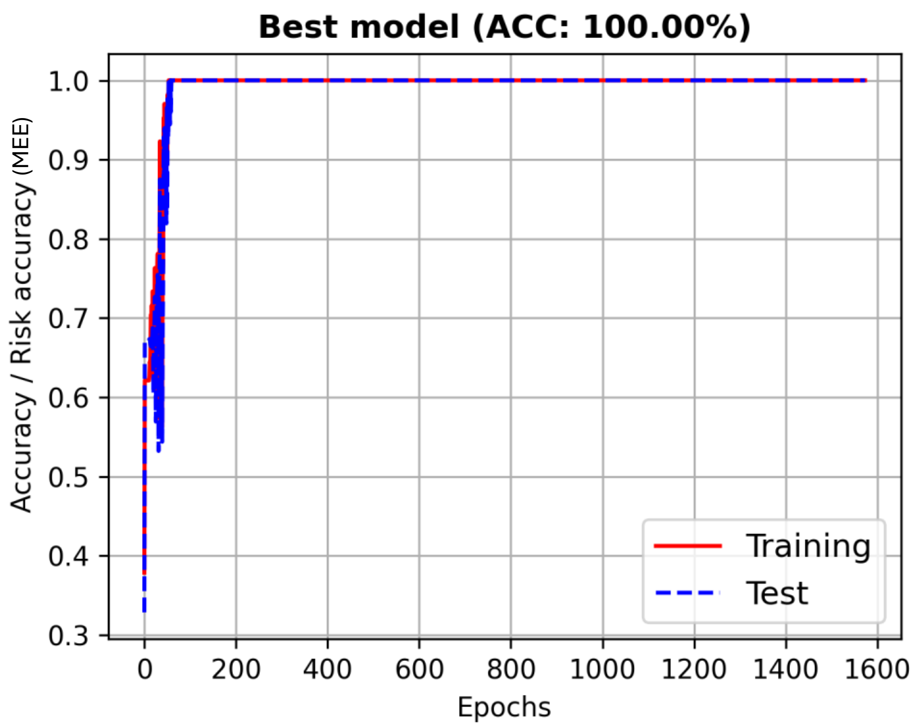


Best model (ACC: 100.00%)
Accuracy / Risk accuracy (MEE)
1.0
0.9
0.8
0.7
0.6
0.5
Training
Test
0
250
500
750
1000
1250
1500
1750
2000
Epochs

MONK 2 - plots

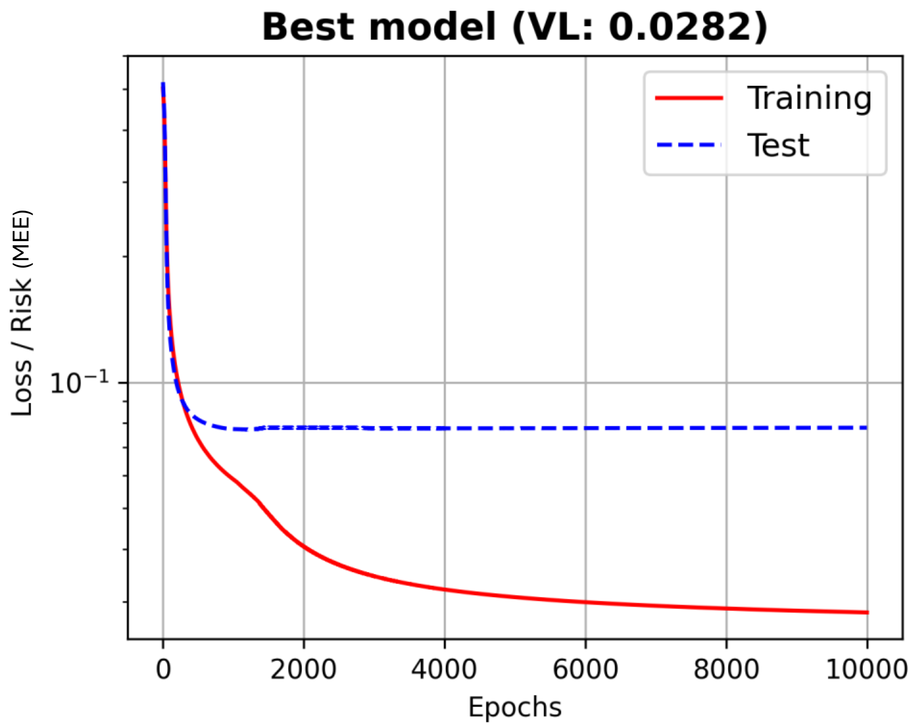


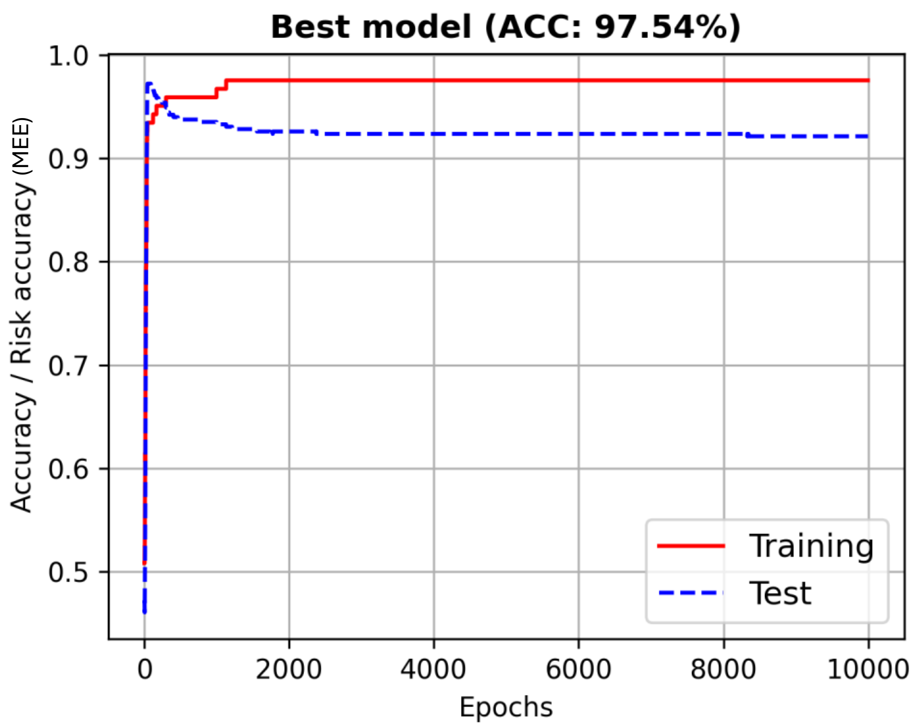
Best model (VL: 0.0071)
Training
Test
Loss / Risk (MEE)
10^{-1}
10^{-2}
0
200
400
600
800
1000
1200
1400
1600
Epochs



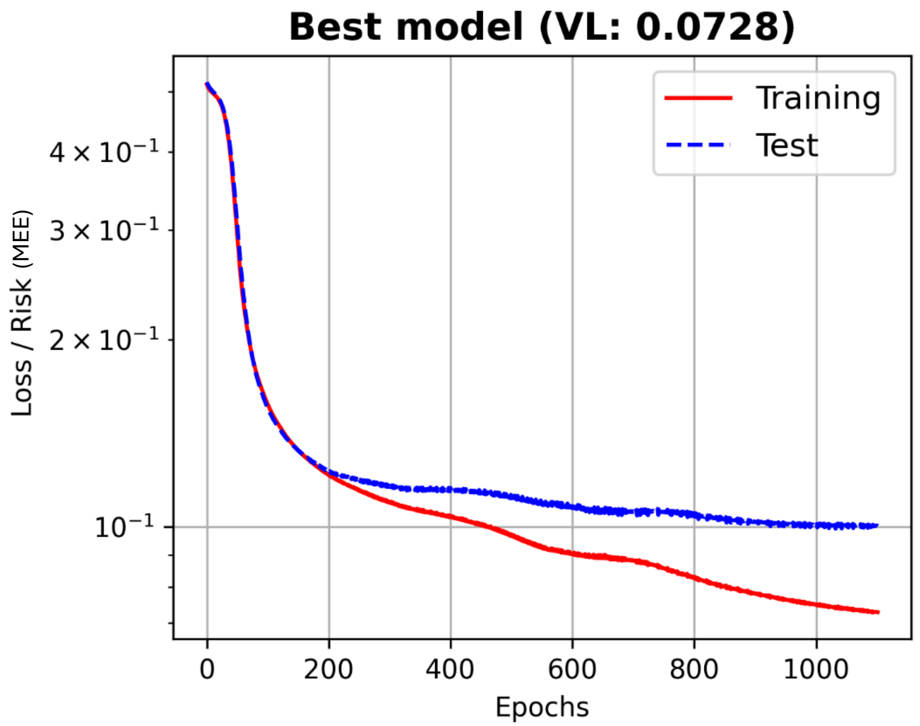
Best model (ACC: 100.00%)
Accuracy / Risk accuracy (MEE)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
Training
Test
0
200
400
600
800
1000
1200
1400
1600
Epochs

MONK 3 (no reg.) - plots

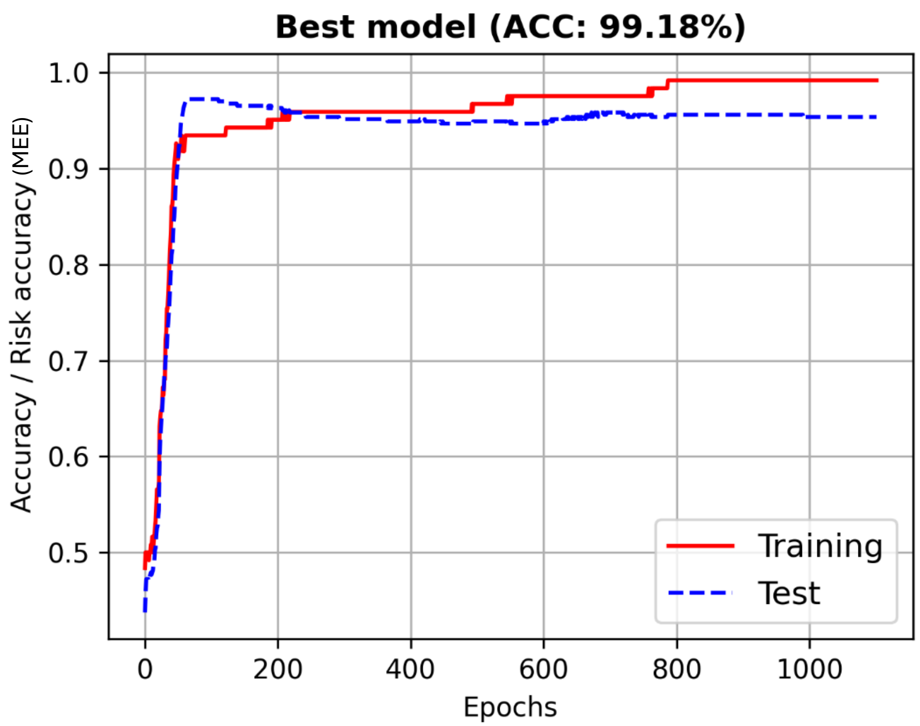




MONK 3 - plots



Best model (VL: 0.0728)
Training
Test
4×10^{-1}
3×10^{-1}
2×10^{-1}
10^{-1}
Loss / Risk (MEE)
0
200
400
600
800
1000
Epochs



Best model (ACC: 99.18%)
1.0
0.9
0.8
0.7
0.6
0.5
Accuracy / Risk accuracy (MEE)
Training
Test
0
200
400
600
800
1000
Epochs

CUP Validation Schema: data splitting

MODEL ASSESSMENT:

Hold-out technique to split the entire dataset in Selection Set (Training + Validation, 80% of Dataset) and Test Set (20% of Dataset)

MODEL SELECTION:

Hold-out technique, with 50% TR and 30% VL, (or **K-fold CV** technique, with k=5) to perform the model selection phase on the Selection Set

Hold-out or 5-fold-CV TR + VL (80%)	Hold-out Internal TS (20%)

After model selection ⇒ retraining on the entire Selection Set (Training + Validation)
After model assessment ⇒ final retraining on the entire Dataset (Training + Validation + Internal Test)

CUP Validation schema: model selection

In order to choose the best hyperparameter configuration, the model selection phase was performed:

Coarse-grained Grid Search → **Fine-grained Grid Search** → **Random Search**

- **Coarse-grained Grid Search**: Focus on testing different architectures (wide range of values for the hyperparameters)
- **Fine-grained Grid Search**: Restricting the search to the most promising ranges of the hyperparameters
- **Random Search**: To finely explore the promising hyperparameter space

Coarse-grained and Fine-grained Grid Search: Selection Set was split using the hold-out technique (VL Set used to monitor Early Stopping and to select the best configuration, i.e. lowest validation error)

Random Search: 5-fold CV on the Selection Set (ensure robust performance assessment)

CUP Validation schema: model selection

Coarse-grained Grid Search

Architecture	Activation functions	η	α	λ	Batch size	Initialization method	Preprocessing
12*,50,50,50,4* 12*,75,75,75,4* 12*,100,100,100,4* 12*,128,128,128,4*	Hidden: ReLU Output: Linear	0.001 0.0001 0.00001	0.1 0.3 0.5 0.8 (Nesterov=False)	1e-02 1e-04 1e-06 1e-08	full	random (range [-0.7,0.7])	Rescaling

CUP Validation schema: model selection

Fine-grained Grid Search

Architecture	Activation functions	η	α	λ	Batch size	Initialization method	Preprocessing
12*,45,45,45,4* 12*,50,50,50,4* 12*,55,55,55,4* 12*,68,68,68,4*	Hidden: ReLU Output: Linear	1e-05 3e-05 5e-05	0.1 0.3 0.5 0.8 (Nesterov=False)	1e-04 1e-06 1e-08	full	random (range [-0.7,0.7])	Rescaling

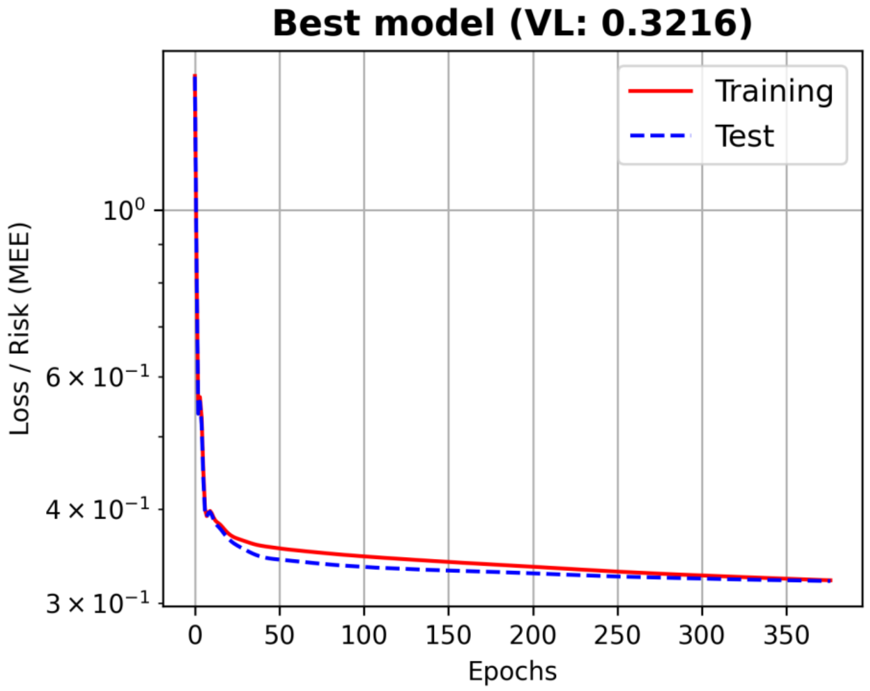
CUP Validation schema: model selection

Random Search

Architecture	Activation functions	η	α	λ	Batch size	Initialization method	Preprocessing
12*,40,40,40,4* 12*,50,50,50,4*	Hidden: ReLU Output: Linear	2.5e-05 3.5e-05	0.7 0.9 (Nesterov=True)	0.5e-08 1.5e-08	full	He (range [-0.7,0.7])	Rescaling

CUP Final Model

The final model was selected based on achieving the lowest validation error combined with a stable learning curve.



Architecture	Activation functions	η	α	$\ln\lambda$	Batch size	Initialization method	Preprocessing
12*, 44, 41,48, 4*	Hidden: ReLU Output: Linear	2.690e-05	0.895 (Nesterov=True)	-18.04	full	He (range [-0.7,0.7])	Rescaling

CUP Final Model – results

MEE		
VL	TR+VL	TS
35.908436	33.987997	32.912784

```
"training": {
  "epochs": 10000,
  "search_type": "grid",
  "number_random_trials": 50,
  "learning_rate": {
    "eta": 0.00002689944986613886,
    "min_rate": 100,
    "decay_factor": 0.0001
  },
  "momentum": 0.8947475896692705,
  "nesterov": true,
  "regularization": 1.4675420469884381e-8,
```

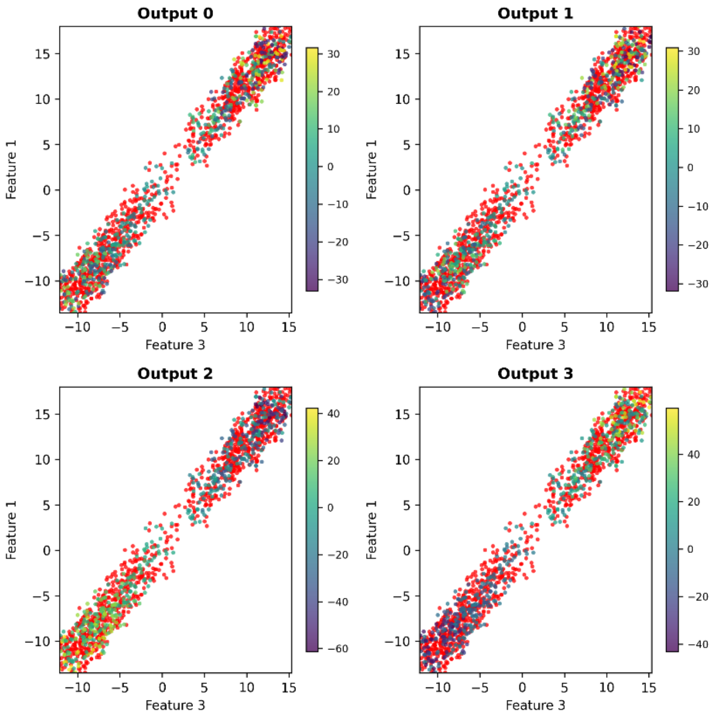
Discussion

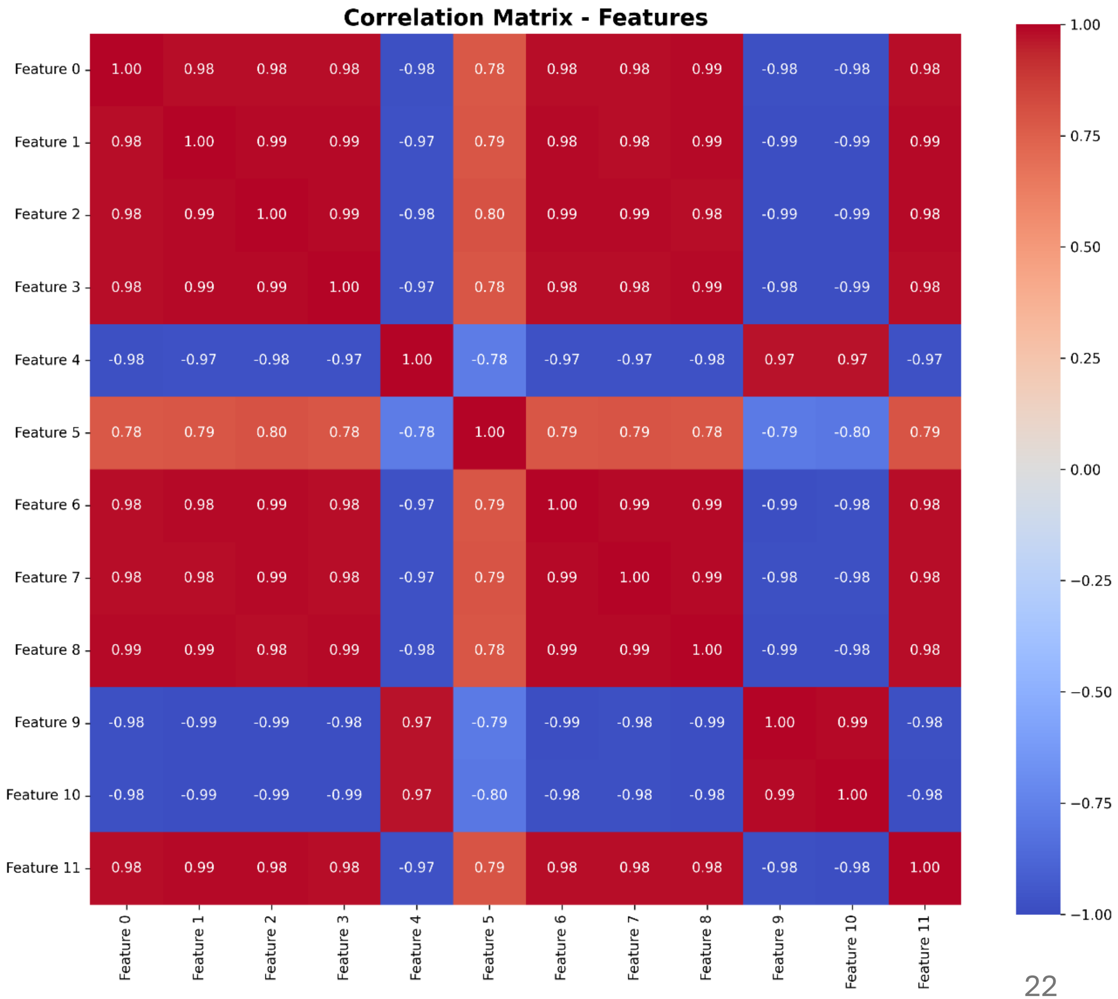
Total runs:	a few thousand (including bad ones)
Total runtime:	~3/4hrs per PC (coarse+fine search)
Most important hyperparameters:	Architecture and Learning Rate, followed by Regularization and Momentum

- Only a small range of lr is good (otherwise divergence/instability/very slow learning)
- Random search allows to explore this range quite well
- Small value of Learning Rate due to absence of 1/batch_size prefactor

Possible improvements - 1

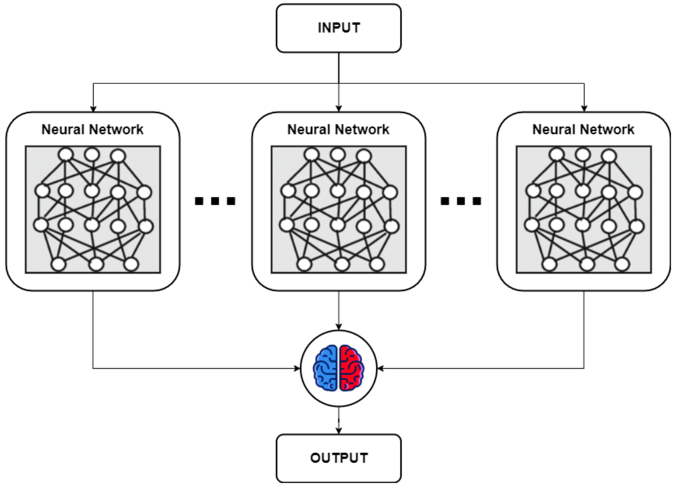
Some of the features or target CUP data are highly (anti)correlated with other features.
It would be useful to develop by scratch some form of PCA to do a better preprocess.



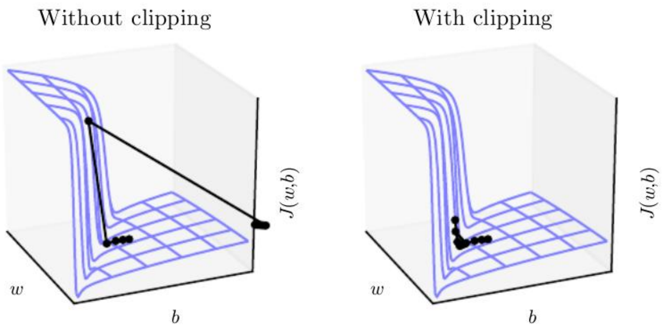


Possible improvements - 2

Ensembling



Gradient clipping



Multithreading scripts

Conclusion

What we learnt

- Build a NN from scratch using only numerical libraries (i.e. Numpy) without higher level ones (i.e. Scikit, Pytorch, ...)
- Cooperate (also between people from different curriculum)
- Read books and papers, don't trust the internet!

References

- Datta, Leonid. «A Survey on Activation Functions and their relation with Xavier and He Normal Initialization». arXiv, 2020. *DOI.org (Datacite)*, https://doi.org/10.48550/ARXIV.2004.06632.

- Shlens, Jonathon. «A Tutorial on Principal Component Analysis». arXiv, 2014. *DOI.org (Datacite)*, https://doi.org/10.48550/ARXIV.1404.1100.

- Ian Goodfellow, Yoshua Bengio, & Aaron Courville (2016). *Deep Learning*. MIT Press.

Thank you for your time! ☺