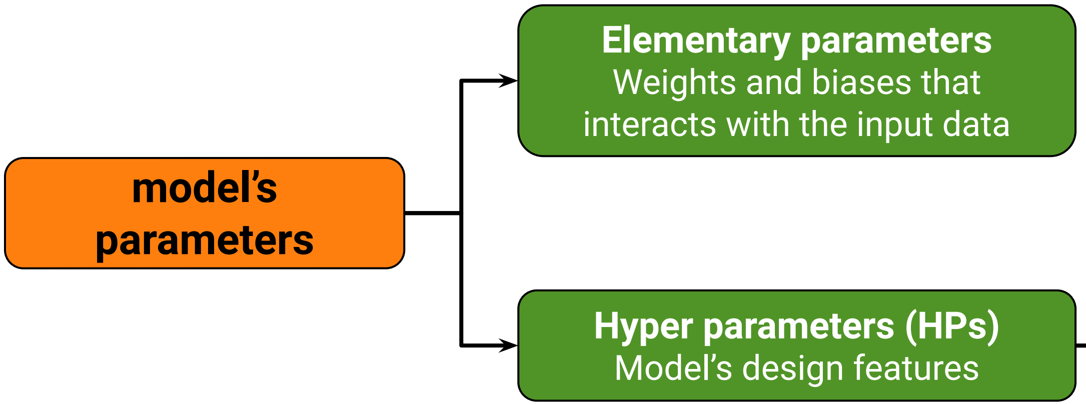
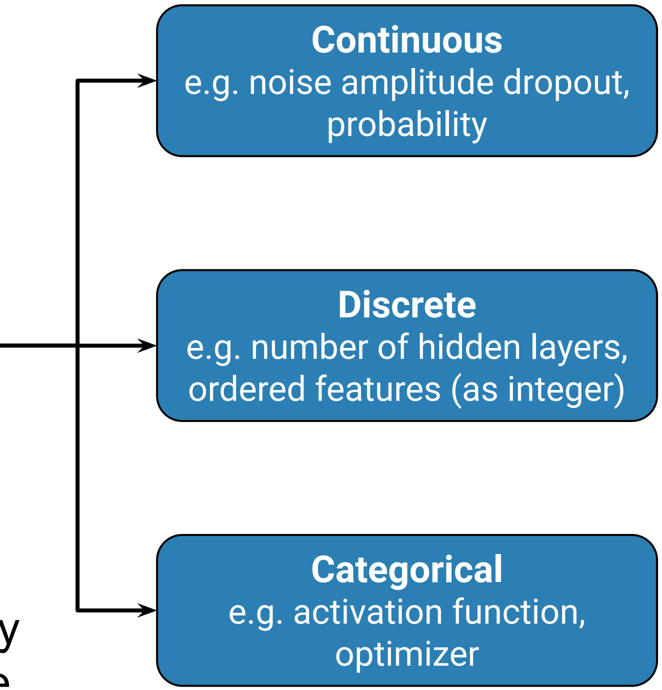
Online Hyper-Parameters Optimization

Stefano Fioravanti (matr. 1806588)





A common assumption is that the **HPs affect the training trajectory**, so they have to be fixed during the whole training.
For this reason the common HP optimizations rely on fully training multiple configurations and then find the best one.

The studied approach takes in account only the **continuous HPs** and assumes that if the largest updates occur at the beginning of the training, and then they become small enough, both the **parameter sets can be tuned simultaneously**.

Proposed method

The samples are splitted in 3 datasets: **training set** (T1), **validation set** (T2) and **test set**. The algorithm alternates a pre-defined number of T1 steps (updating the elementary parameters) with one T1-T2 step (tuning also the hyper parameters).

The paper focuses on the optimization of:

- **L2 regularizer (λ)**: the additive loss penalty, computed as

$$\Omega(\lambda, \theta) = \sum_j 10^{\lambda_j} \frac{\theta_j^2}{2}; \;\; \lambda \in [-5.5; -2.5]$$

- **Input noise (n0)**: the Gaussian noise's standard deviation added to the input data

$$x_{in} = x + \mathcal{N}(0, n_0); \;\; n_0 \in [0.0; 0.8]$$

Hyper-parameters' gradient

To update the HPs, the gradient wrt the cost function is required.
The gradient computation is the main part of the algorithm, and it depends on how the HPs are implemented.

For the **L2 regularizer**, the gradient is

$$\nabla_\lambda C_2 = -\eta_1 \nabla_\theta C_2 \cdot \theta$$

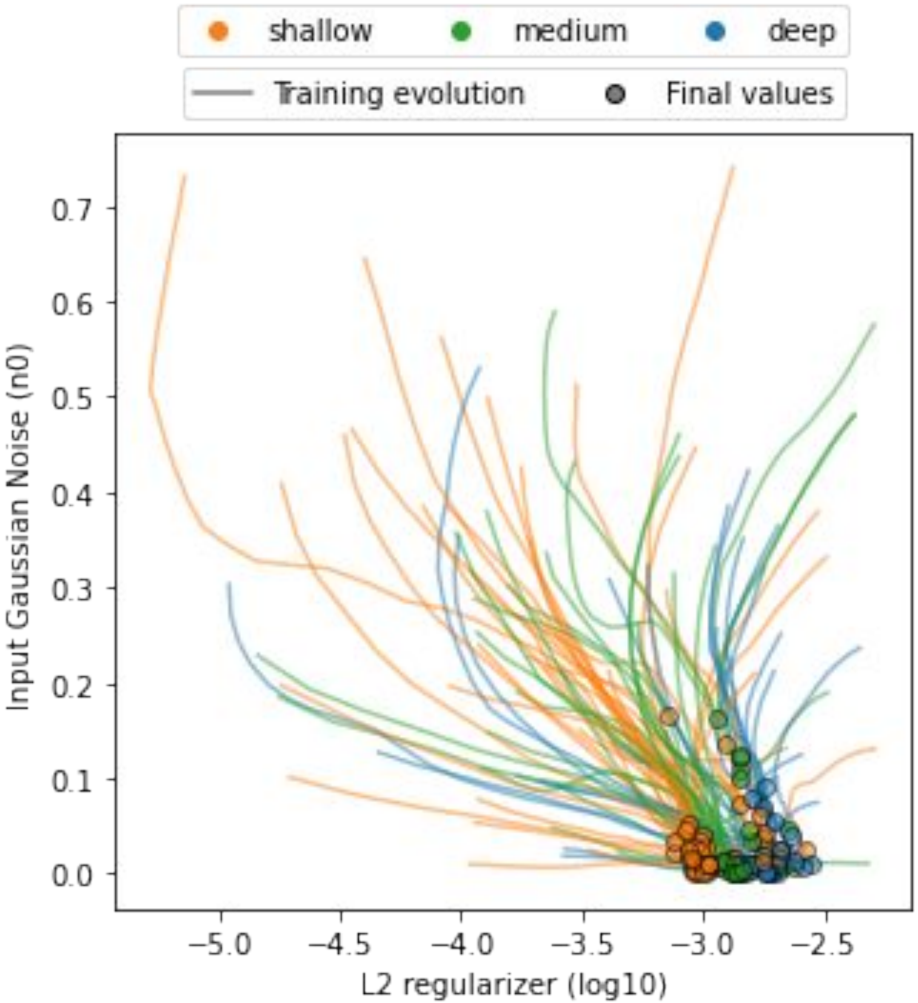
For the **input noise n0**, the gradient can be computed through the backward propagation

$$\nabla_{n_0} C_2 = \mathcal{N}(0, n_0) w_0 \nabla_{w_0} C_2$$

Efficiency

Training the model multiple times, with different initial configurations, it's possible to see how the algorithm tunes the HPs into the **same optimal region**.

Different model sizes have slightly different optimal regions, this behavior highlights the difficulty of the manual tuning.

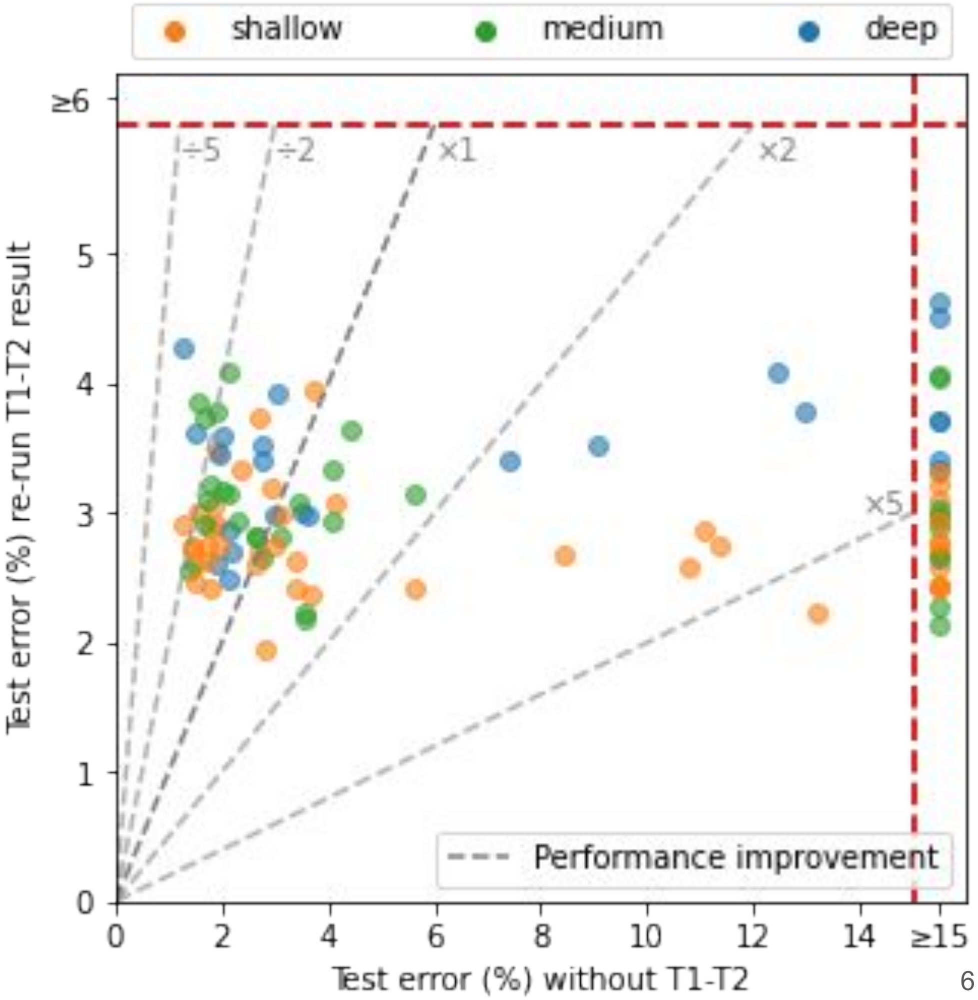


Performance #1

The achievement of the optimal region can be seen also studying the run performances.

58% of the runs achieve good performances whether with the initial configuration or the optimized one.

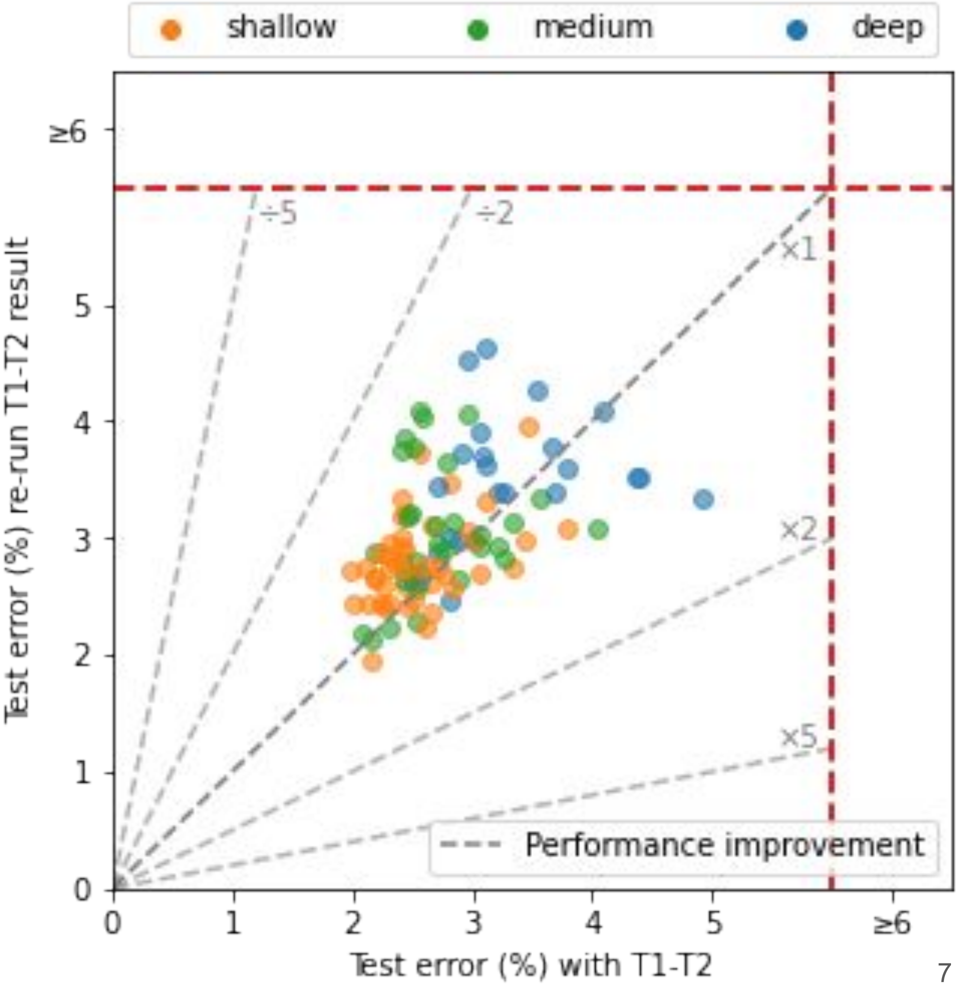
20% of the runs **improve** their performances drastically thanks to the algorithm.



Performance #2

Comparing the model trained with T1-T2 optimization and the ones trained with the optimized configuration, the performances lie always close to the main diagonal (so they are always similar).

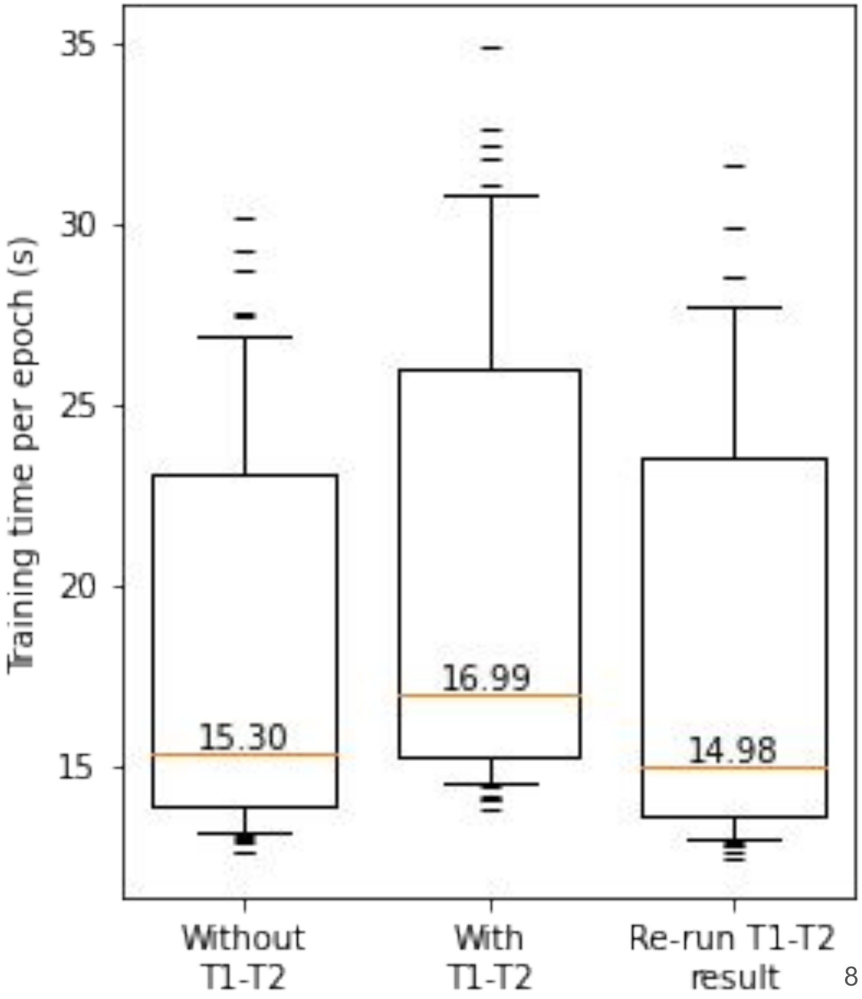
Therefore, **training again the model with the optimized configuration is not required** and the algorithm can be considered faster.



Time consumption

The algorithm requires more time than the normal one, since it has, mainly, to compute the HPs' gradients.

Taking in account the proposed HPs, the training time has increased by 2s (12%).



Conclusion & Future works

The algorithm **works well on continuous HPs**, even if, in this way, many design features are cut out from the optimization.

To apply the algorithm it's necessary to compute the gradient for the backpropagation, this step makes the algorithm **not easy to generalize** and apply on different model. Once the gradient has been computed the algorithm can be easily applied, and it has a **small impact on computation** and training time.

A possible improvement is to merge this algorithm with other ones that take in account discrete and categorical HPs.

Thank you for your attention