A meta-learning approach to tune hyperparameters in Neural Networks

Micaela Verucchi(*), Giorgia Franchini(*), Matteo Spallanzani(*), Marko Bertogna(*) and Luca Zanni(*)

(*) Department of Physics, Informatics and Mathematics, Modena (Italy)

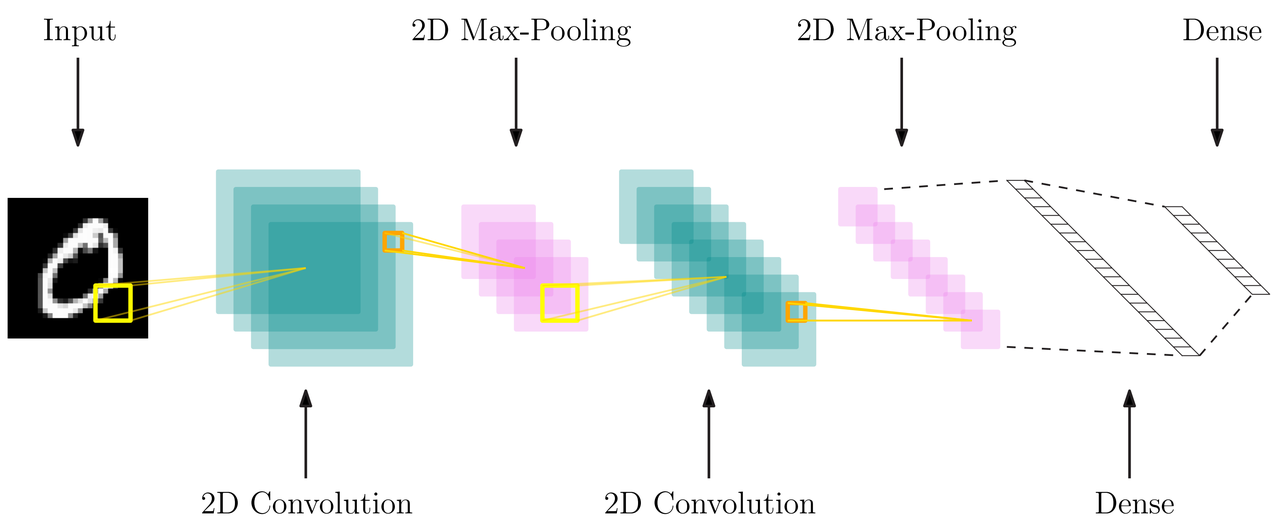
Problem statement

- Hyperparameters are a fundamental component of most machine learning algorithms; these parameters strongly influence both the space of approximating functions and the way this space is explored during the training phase.
- Hyperparameters tuning is particularly critical for Deep Learning, as it affects both the convergence time of the training procedure and the network's accuracy.
- Hyperparameters have usually been tuned through careful handmade procedures. Nowadays, optimized search algorithms which goal is *learning to learn* are emerging: this research field deals with the so-called **meta-learning** approach.

- Naive strategy: grid search ("brute force").
- Our strategy: probabilistic change of the learning rate, driven by past measurements of the test error.

1. Start with a heuristic guess of the learning rate, then train the network.
2. Randomly select an increase/decrease of the learning rate, then train the network again.
3. If the new test error improves, then increase the probability to repeat the choice; if the new test error worsens, then decrease the probability to repeat the choice.
4. Return to step 2.

CNN experimental test architecture



Network architecture:

- Input: 28x28x1 (MNIST image [5])
- 2D Convolution: 5x5x32, stride 1x1, "same" padding, ReLU
- 2D Max-Pooling: 2x2, stride 2x2
- 2D Convolution: 5x5x64, stride 1x1, "same" padding, ReLU
- 2D Max-Pooling: 2x2, stride 2x2
- Fully connected: 1024 units, ReLU
- Dropout: $p_{keep} = 0.5$
- Output: 10 units (class probabilities)

Training parameters:

- Mini-batch size: 50
- Loss function: *cross entropy*
- Optimizer: *Adam*
- Early stopping

Algorithm

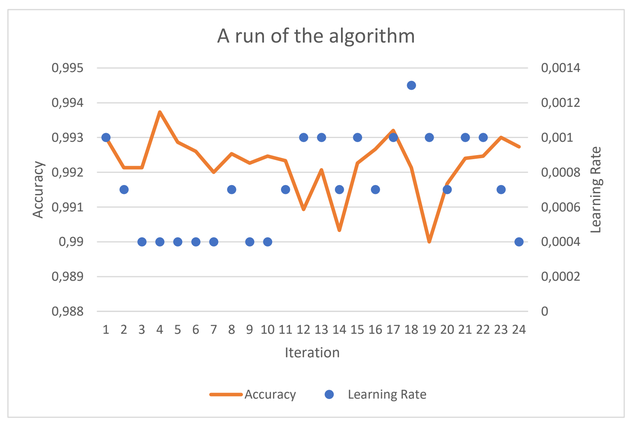
Algorithm 1 AdjustLR

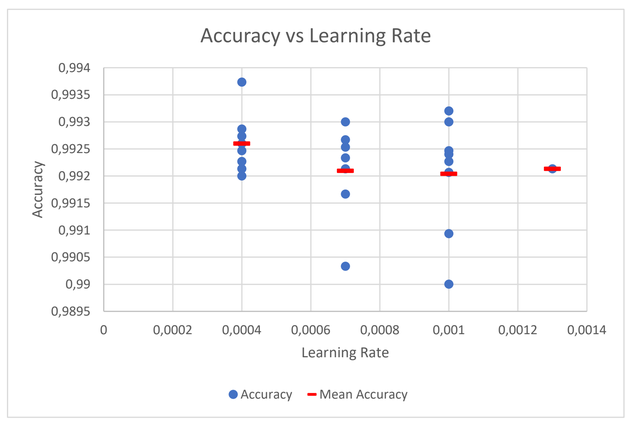
Input: $\lambda_{in}, p_1, p_2, p_3$
Output: λ_{out}, c
Sample $u \sim U[0,1]$
if $u < p_1$ **then**
  $\lambda_{out} = \lambda_{in} + \epsilon$
  $c = 1$
else if $u < p_1 + p_2$ **then**
  $\lambda_{out} = \lambda_{in}$
  $c = 2$
else
  $\lambda_{out} = \lambda_{in} - \epsilon$
  $c = 3$
end if

Algorithm 2 TuneLearningRate

Set $i = 1, noprog = 0, i_{max}, noprog_{max} \in \mathbb{N}$
Set $\lambda_0, \epsilon, p_\epsilon \in \mathbb{R}^+$
$acc_{old} = Train\_NN(\lambda_0)$
Set $p_1 = p_2 = p_3 = 1/3$
$\lambda_1, c = AdjustLR(\lambda_0, p_1, p_2, p_3)$
while $i < i_{max}$ **and** $noprog < noprog_{max}$ **do**
  $acc_{new} = Train\_NN(\lambda_i)$
  if $acc_{new} \geq acc_{old}$ **then**
    $p_c = p_c + p_\epsilon$
    $p_{\bar{c}} = p_{\bar{c}} - p_\epsilon/2$
    $\lambda_{best} = \lambda_i$
    $noprog = 0$
  else
    $p_c = p_c - p_\epsilon$
    $p_{\bar{c}} = p_{\bar{c}} + p_\epsilon/2$
    $noprog = noprog + 1$
  end if
  $\lambda_{i+1}, c = AdjustLR(\lambda_i, p_1, p_2, p_3)$
  $acc_{old} = acc_{new}$
  $i = i + 1$
end while

Experimental results





Perspective work

- Apply this methodology to other hyperparameters: mini-batch size, optimizer, regularization technique. This exploration will benefit of 25000 hours of HPC resources that we have been granted by the CINECA consortium [6].
- Apply our algorithm to train a CNN that should solve behavioural cloning (steering angle prediction) and be deployed on an autonomous F1/10th toy car [7].
- Apply our algorithm to train a CNN that should solve image segmentation on both a classification task (semantic segmentation) and a regression task (depth estimation), and should be deployed on crossroads cameras in the Modena Automotive Smart Area (MASA) [8].

References

[1] *Meta-learning approach to neural network optimization*, P. Kordìk, J. Koutnìk, J. Drchal, O. Kovarik, M. Cepek, M. Snorek, Neural Networks 23, pp. 568–582 (2010)
[2] *Learning to learn by gradient descent by gradient descent*, M. Andrychowicz1, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, N. de Freitas, arXiv:1606.04474v2 (2016)
[3] *Adam: A method for stochastic optimization*, D. Kingma, J. Ba, arXiv:1412.6980 (2014)
[4] *Optimization methods for large-scale machine learning*, L Bottou, FE Curtis, J Nocedal, arXiv:1606.04838 (2016)
[5] *yann.lecun.com/exdb/mnist/*
[6] *www.cineca.it/en*
[7] *f1tenth.org*
[8] *https://class-project.eu/*