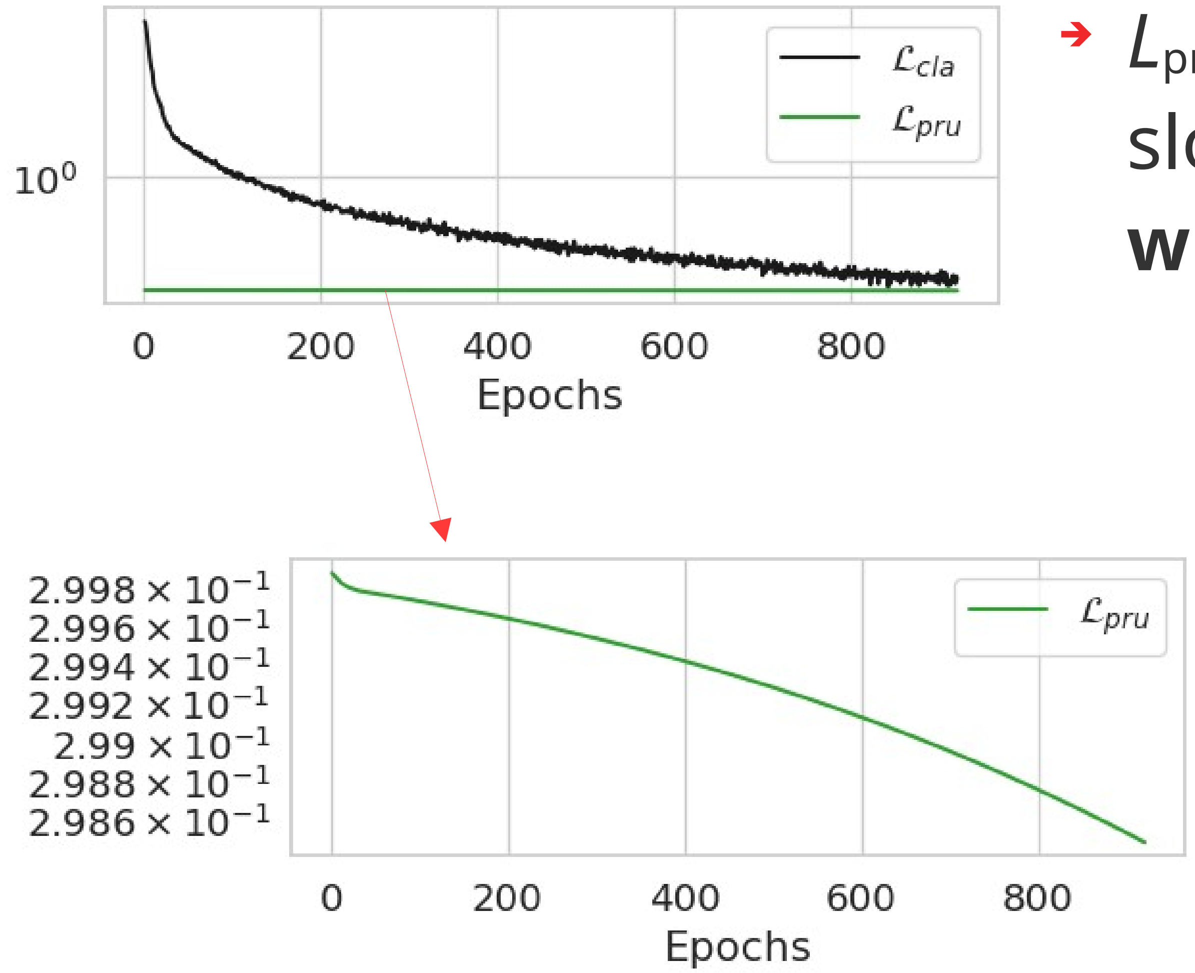
deepPP weekly meeting

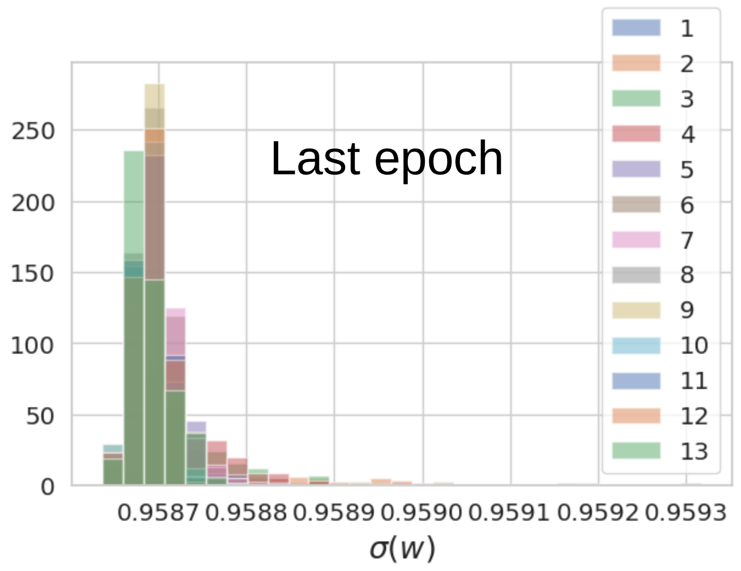
28/04/2023

deeppp

The issue with our pruning tool

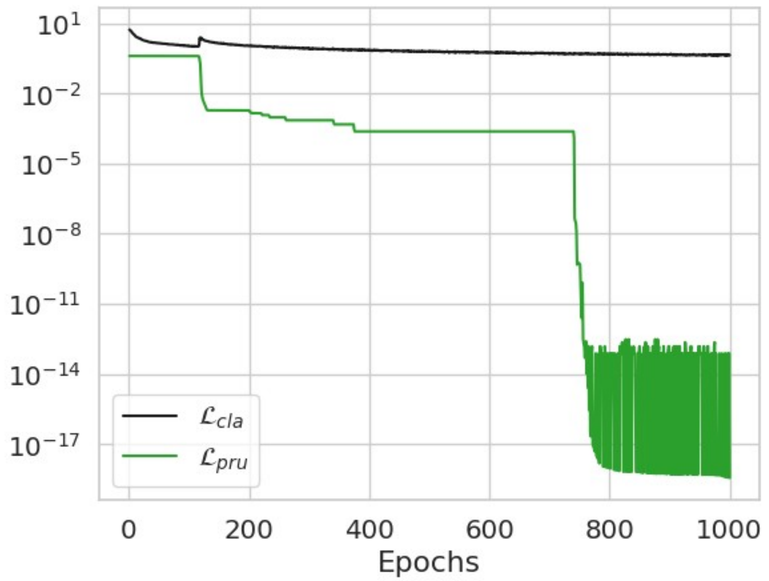


- L_{pru} is deacreasing so slowly that the **pruning will never take place**

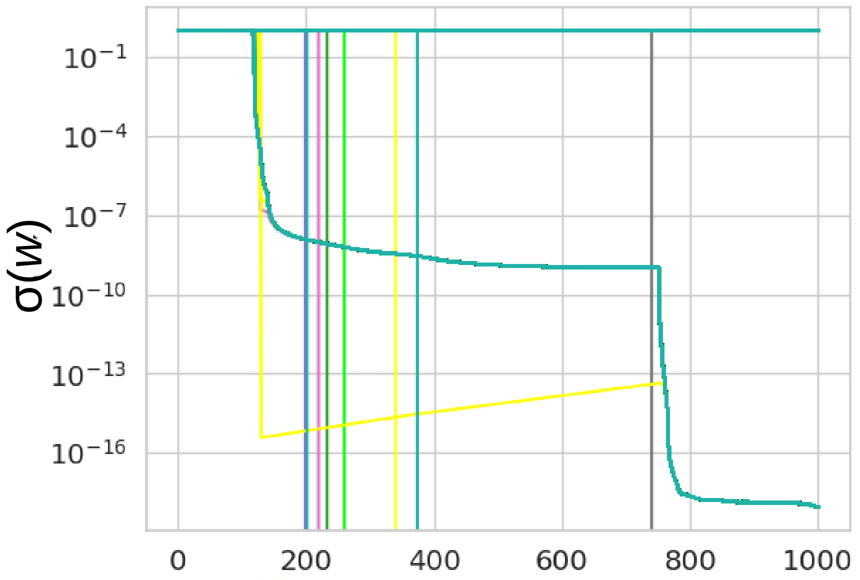


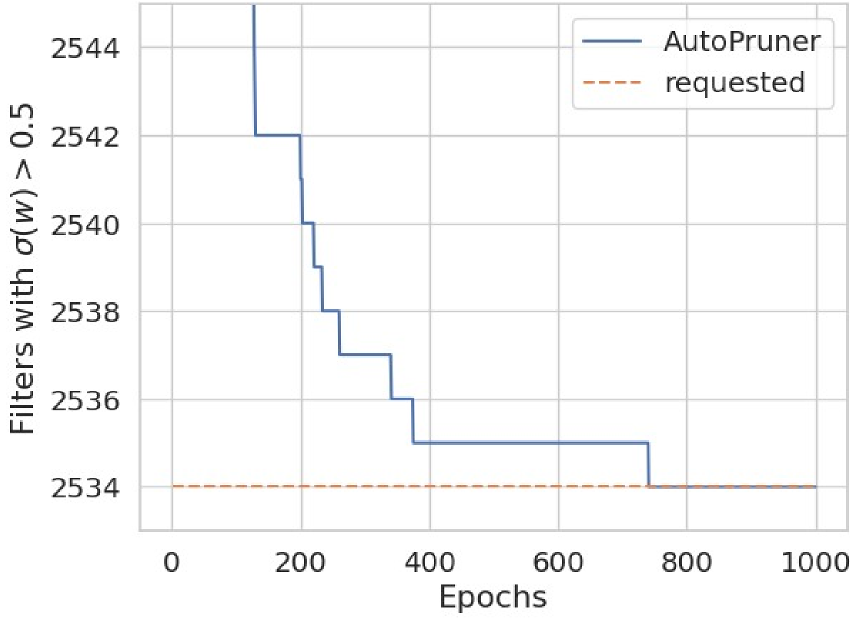
A proposed workaround

- $L_{tot} = L_{cla} + \mu * \lambda * L_{pru}$
  - ➔ λ=λ*10 every 20 epochs, from λ=1 to $\lambda=10^7$



This works, **BUT** does not explain WHY and HOW

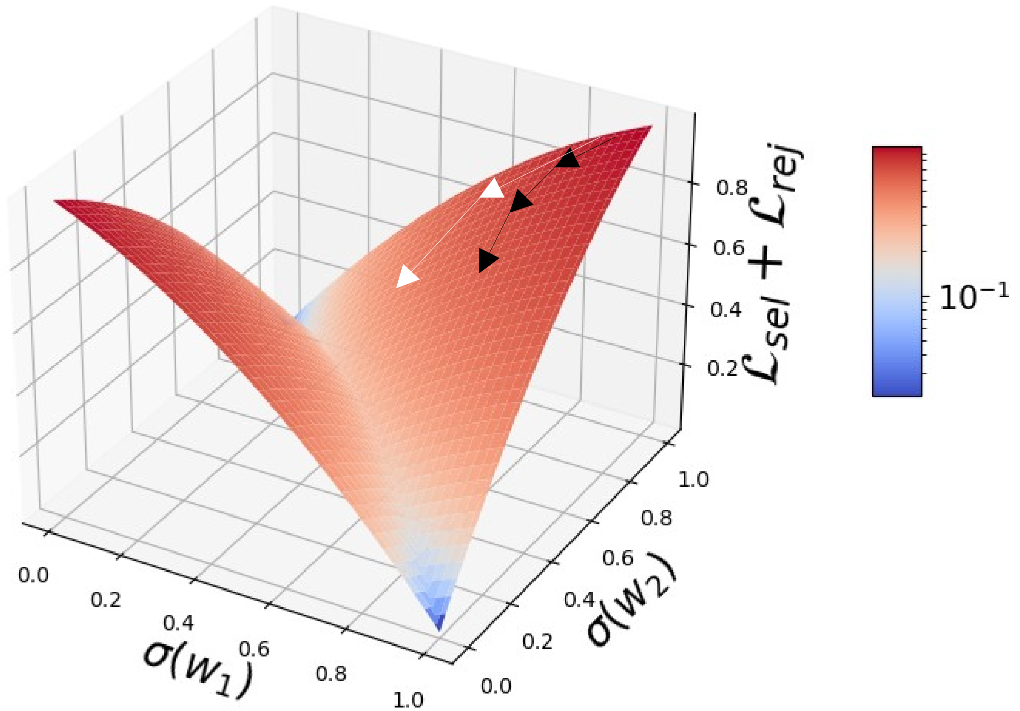


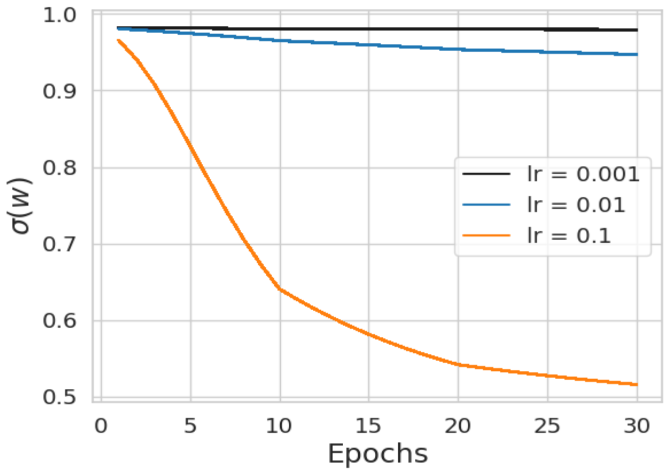


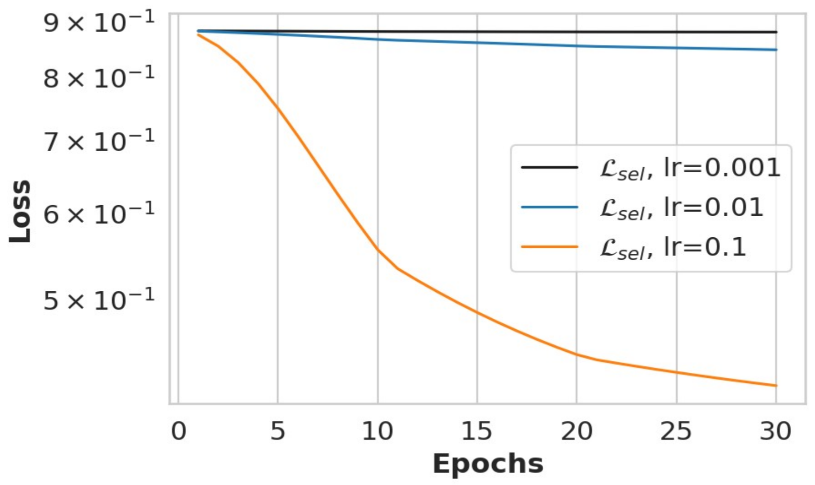
Studying the problem in depth

The problem seems to be related to the **learning rate**: if the learning rate is too low L_{pru} will decrease too slowly

- Increasing the learning rate may be a possible solution







Collateral effects

As shown by Alessia, increasing the learning rate

- has detrimental consequences on the accuracy of the model
- may result in the nodes switched on again

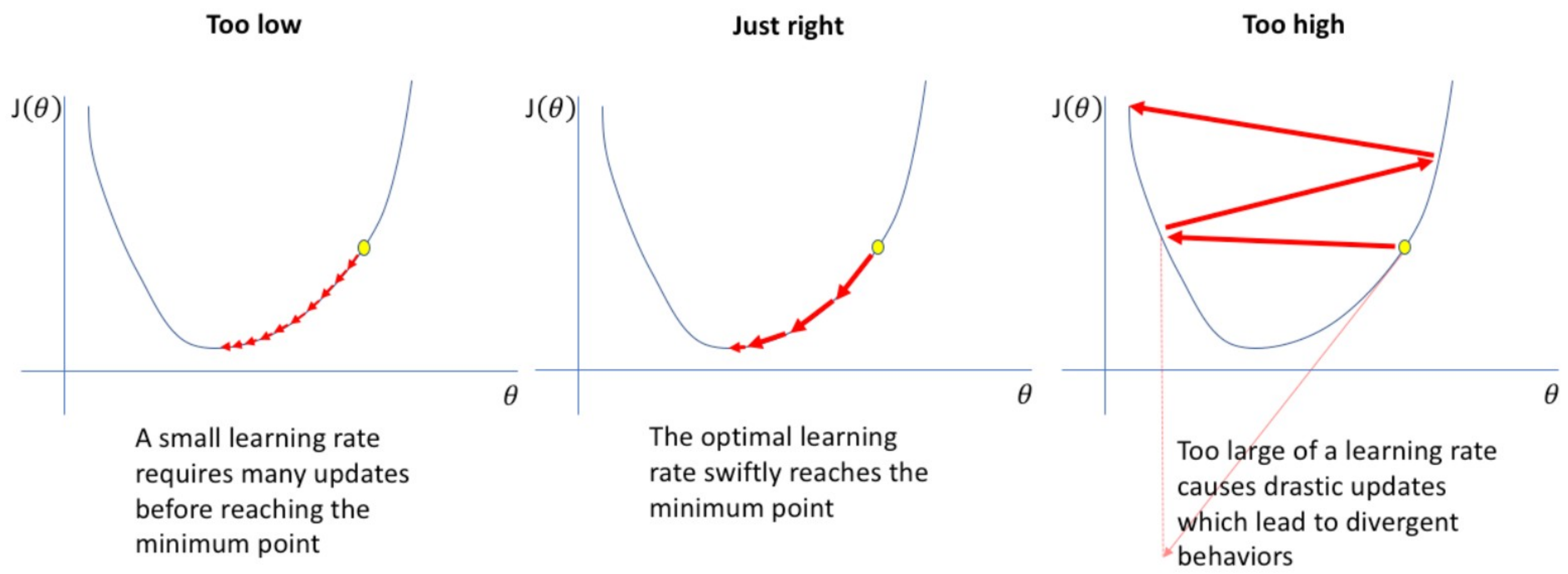
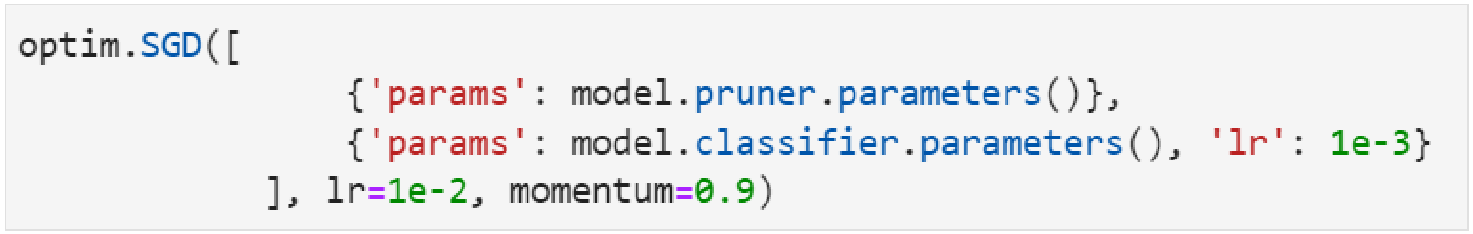

Too low
Just right
Too high
J(θ)
θ
A small learning rate requires many updates before reaching the minimum point
The optimal learning rate swiftly reaches the minimum point
Too large of a learning rate causes drastic updates which lead to divergent behaviors

Image source: https://www.jeremyjordan.me/nn-learning-rate/

Possible solutions

- Begin with a high learning rate and decrease it once the pruning has started
  - BUT: this may lead to an incomplete pruning (as shown by Alessia)
- Try to use two different learning rates for L_{cla} and L_{pru}

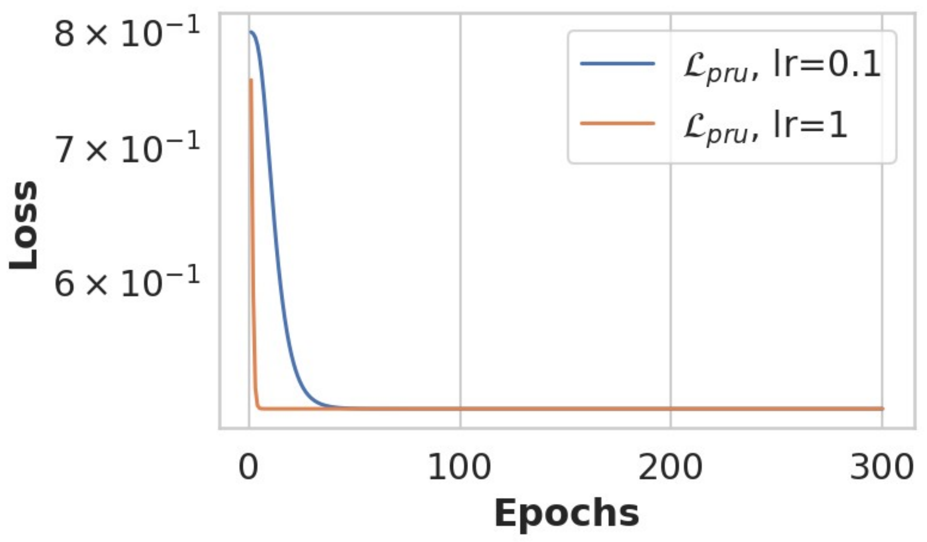


```
optim.SGD([
                {'params': model.pruner.parameters()},
                {'params': model.classifier.parameters(), 'lr': 1e-3}
            ], lr=1e-2, momentum=0.9)
```

Source: https://pytorch.org/docs/0.3.0/optim.html#per-parameter-options

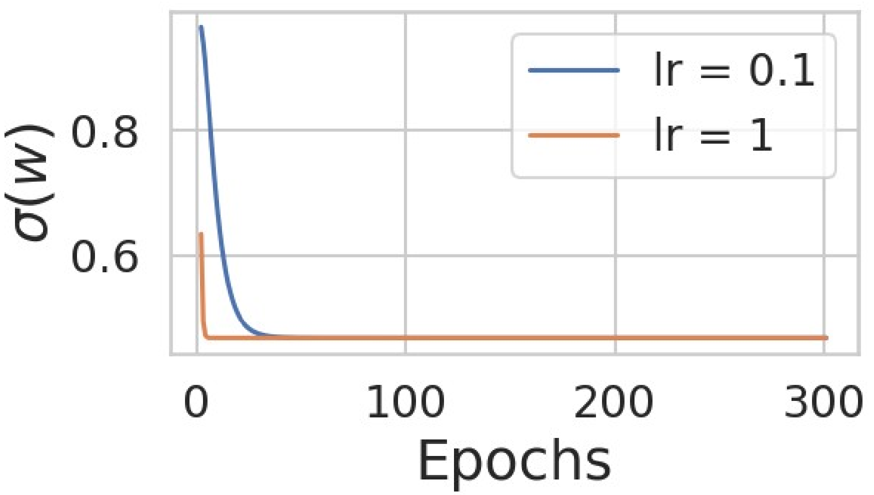
Further problems

In order to determine the most suitable learning rate for the pruning to occur, trainings with $L_{tot} = L_{sel} + L_{rej}$ and different values of learning rates have been performed



Independently of the value of the learning rate, the pruner loss gets stuck into a flat valley, does not reach the global minimum and the pruning does not occur

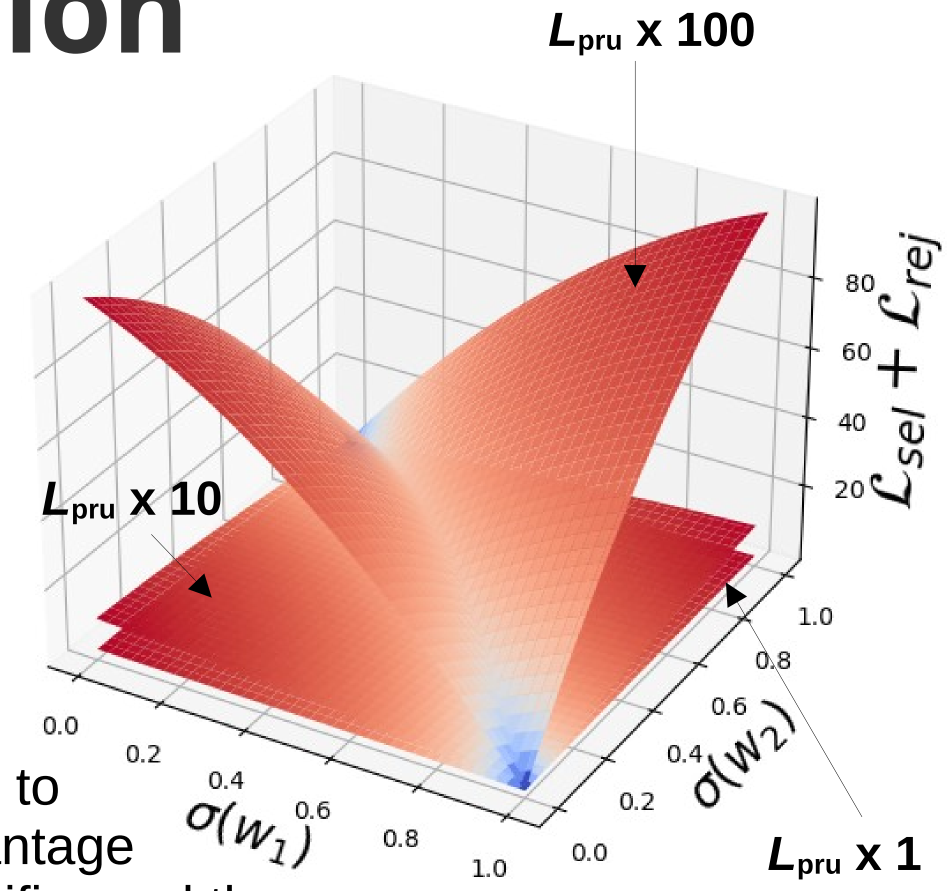
The value of the learning rate only determines how fast L_{pru} freezes



Alternative solution

- Increase the slope of L_{pru}

The effect of introducing a coefficient that multiplies the L_{pru} term is to **increase its slope**, which is why the workaround was actually working



The introduction of a coeffient to increase the slope of the pruning loss could represent an alternative solution, equivalent in principle to increasing the learning rate but with the advantage of using the same learaning rate for the classifier and the pruner.

Ideas for defining λ:

- inversely proportional to the variation of L_{pru}

- related somehow to the size of the net (since this problem appears when dealing with a big network such as VGG16)

Next step

- Look for alternative solutions or already developed learning strategies to deal with the problem of a stuck loss
- For the FTAG group and Hbb/cc tagger task force:
  - Repeat the study done for Hbb tagging also for single *b*-tag
  - Vary the size of the network (still need to think of how)