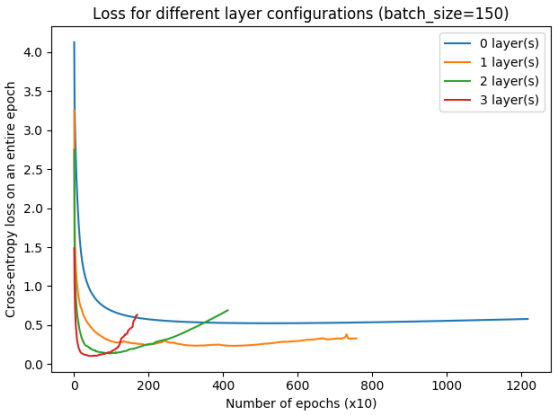
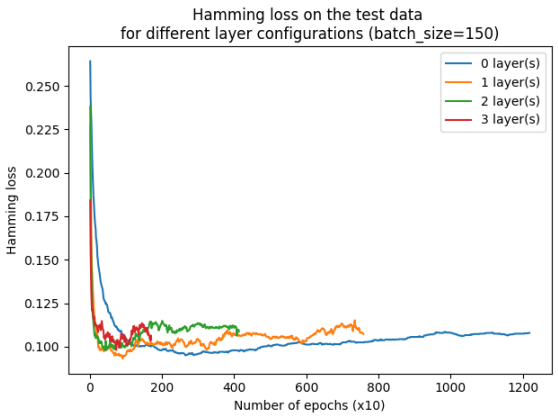
# MACHINE LEARNING (INF554)

Romain Fouilland – 1st assignement

## NETWORK STRUCTURE

First, I focused on the impact of the **number of hidden layers**. On the first figure, we can see that the cross-entropy (which is being optimised) decreases when we add layers. However, we can see that with 2 and 3 layers, we overfit very quickly. Besides, when we look at the hamming loss on the test data (second figure), we can see that 1 layer actually yields the most accurate network.



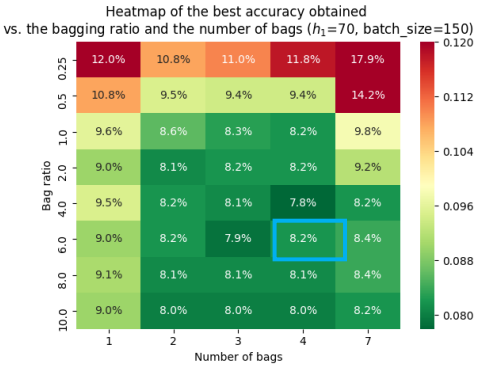




*Conclusion*: I decided to take **1 hidden layer** and focus on **reducing the overfitting**.

## BAGGING

To reduce the overfitting, I decided to implement a method used for random forests: **bootstrap aggregating**. On the third figure, we can see the best hamming loss obtained over the training depending on the number of bags and the ratio of data used in each bag (the rows are drawn randomly with replacement).

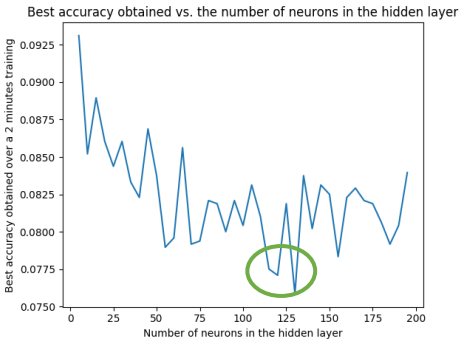
*Conclusion:* using **4 bags of 4x** the number of instances helps reduce the overfit and reach a high level of accuracy.




## NUMBER

I then tried to determine the optimum **number of neurons** for the hidden layer. On the fourth picture, we can see the best hamming loss obtained on the train data vs. the number of neurons. Even though there is no clear optimum, it seems to be a bit better to have between 110 and 130 neurons.

*Conclusion:* I took **115 neurons**. However, it varies a lot around that value so it's not really relevant to pick a precise value (because we shouldn't overfit the parameter on the test data).




## OTHER

I eventually plotted the accuracy as a function of the **batch size** (between 5 and 200) and the **learning rate** (between $10^{-5}$ and $10^{-3}$).

*Conclusion:* The best accuracy was reached for a batch_size of 100 and a learning rate of $10^{-4}$.

I also did some attemps I eventually didn't use:

- Scaling the data: reduced the accuracy a bit.
- Bootstrap aggregating by features: reduced the accuracy (the model didn't seem to learn well).
- Dropout: little effect once bootstrap aggregating was used.
- Randomizing the order of the instances/batches : little effect once bootstrap aggregating was used.
- Trying to return 0.5 / 0.5 (to still be a proba) when 2 classes were to be found. It didn't work well but we thus always miss one of the two classes of 1.2% of the test data.

*Conclusion*: With 1 hidden layer of 130 neurons, 4 bags of 4x instances, batch size of 70 and a learning rate of $10^{-4}$, I get a Hamming Loss of **7.77% after 110s and 8.02% at the end** (small overfitting).

*Benchmarks*:

- OneVsRestClassifier: 9.92% in 977ms
- ClassifierChain: 9.54% in 993ms
- KNeighborsClassifier: 10.56% in 43ms
- DecisionTreeClasifier: 14.56% in 325ms