Before doing the following, I found by hand an approximation of the **best possible values** for the 3 tested parameters. I decided to set them as **default values** in order to refine them with the following protocol:

|  |  |  |
| --- | --- | --- |
| **First round**: Vary number of layers, learning rates, number of epochs **(with fixed default values, [in brackets])** | | |
| Layers **[4]** | Learning rates **[0.1]** | Epochs **[600]** |
|  |  |  |
| 🡪 Number of layers = 4 | 🡪 Learning rate ∈ [0.01, 1] | 🡪 Number of epochs ∈ [250, 1200] |

|  |  |  |
| --- | --- | --- |
| **Second round**: Fix number of layers = 4 and try to improve learning rates in [0.01, 1] and number of epochs > 500. Since all running times are far less than 120 secondes, I only look at the losses. | | **Final results**: Learning rate = 0.5  Number of layers = 4  Number of epochs = 1000 |
| Learning rates **[0.5]** | Epochs **[750]** |  |
|  |  |
| 🡪 Learning rate = 0.5 seems gives the best result | 🡪 Any number of epochs >200 seems relevant. No overfitting under <1200 at least. Lets take 400 | Execution time : 112 seconds  Hamming loss : ≈8.6% |

|  |  |
| --- | --- |
| **Comparison** with SKLearn methods | |
| Execution times | Hamming losses |
|  |  |
| Execution time of the NN is way higher but the hamming loss is better. | |