Université de Bordeaux
Image & Son

DLCV Lab 2_3 et 3

Report

Author :
Sofian Antri
Adrien Célérier

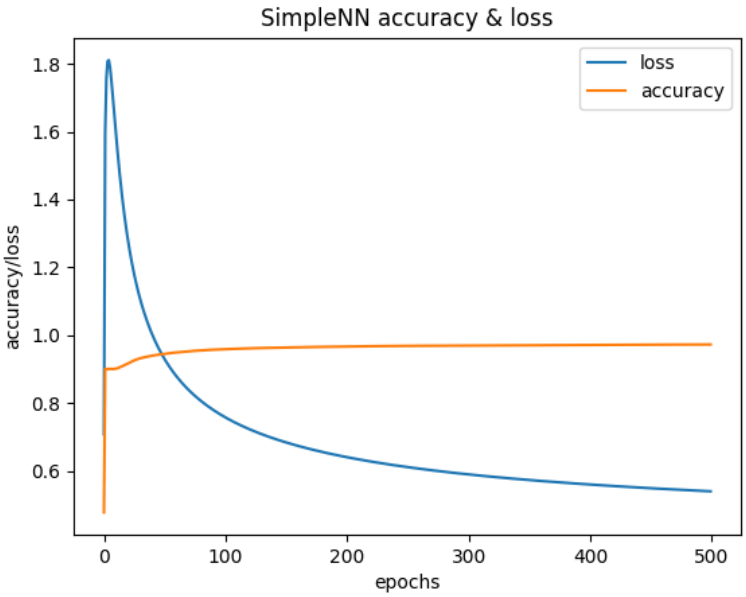
Supervisor :
Boris Mansencal

Contents

1 Multiclass Neural Network from Scratch

With the simple Neural Network, we can observe an increasing loss at the beginning (for about 5 epochs). This might be due to the fact that the AI encounters a digit never seen before. It keeps decreasing afterwards, while the accuracy keeps increasing before quickly starting to converge.

Figure 1: Accuracy & Loss of Multiclass Simple Neural Network with 500 epochs



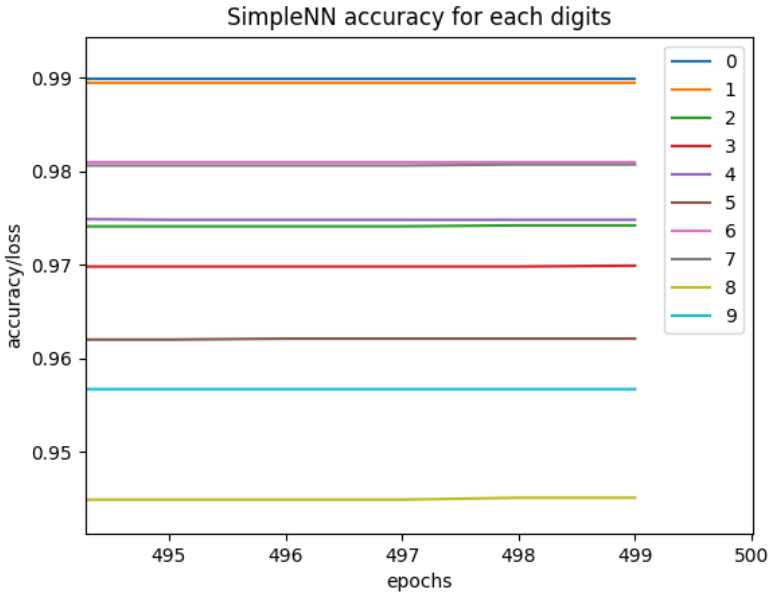
After 500 epochs, the accuracy is close to stagnant at 97,25%. The loss keeps decreasing slightly but is also quickly converging.

```
Simple NN -- Epoch :  500
Loss : 0.5390037892
Accuracy : 97.25%
Total time :  1213.333236694336
```

The accuracy for each digits goes in this order (from the best to the worst):

`0, 1, 6, 7, 4, 2, 3, 5, 9, 8.`

Figure 2: Accuracy for each digit at the end of the 500 epochs

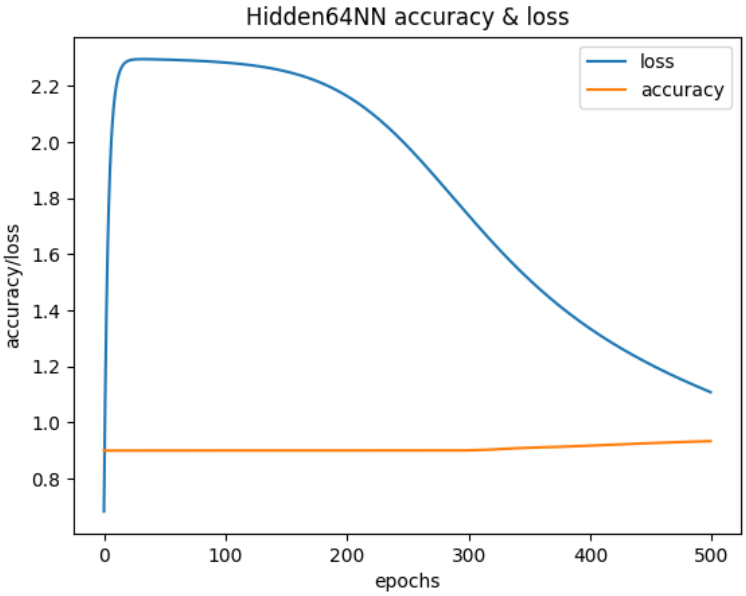


There seems to be a pattern for the different digits, with 0 and 1 being the most easily recognizable, and 8 being the worst.

As for the 64-sized hidden layer Neural Network, we have a loss that increases, and then stays high for a much larger number of epochs, before finally going down.
The accuracy, however, stays constant until the epoch 280 (approximately), before finally going up. We can also notice that the accuracy for each digit stays constant during this time.

```
64-sized hidden layer NN -- Epoch :  500
Loss : 1.1084138134
Accuracy : 93.37%
Total time :  1400.590366601944
```

Figure 3: Accuracy & Loss of Multiclass Hidden layer of 64-sized Neural Network with 500 epochs

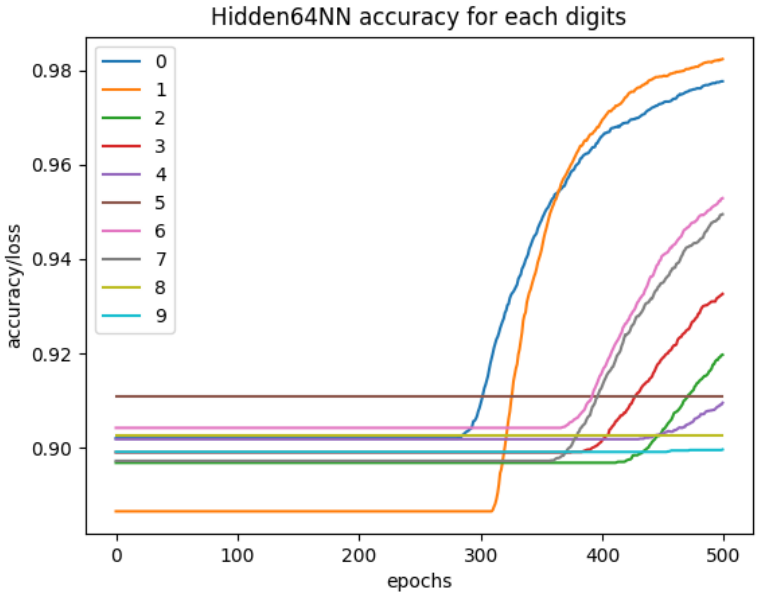


We could probably say that the results would get better if we went further than 500 epochs, because the curves weren't done converging. We also notice that this one takes a bit more time to process.

The accuracy for each digits goes in this order (from the best to the worst):

`1, 0, 6, 7, 3, 2, 5, 4, 8, 9.`

Figure 4: Accuracy for each digit at the end of the 500 epochs



We can notice some slight differences between these results and the Simple Neural Network's one, but we also have to keep in mind that these result aren't complete since, as we mentioned earlier, the curves weren't done converging.

2 Simple Neural Network with Keras

The **Keras** version follows an epoch-by-epoch verbose output.
When using the *sigmoid* activation function, the final accuracy reaches around 99.24% after 300 epochs in under 144 seconds, so we can easily say that the **Keras** version is much more precise and much quicker than the "from-scratch" one.

```
Loss : 0.0237078574
Accuracy : 99.24%
Total time : 143.97s
```

However, with different activation functions, we need to take care of eventual overfitting.
This is the case for the *relu* function that starts having an erratic behaviour from the 117th epoch, ending after 400 periods with an accuracy around 10%.

```
Epoch 117/300
469/469 [==============================] - 1s 1ms/step - loss: 1.5225
- accuracy: 0.9013 - val_loss: 1.5116 - val_accuracy: 0.9020
Epoch 118/300
469/469 [==============================] - 1s 1ms/step - loss: 10.8688
- accuracy: 0.2875 - val_loss: 13.7548 - val_accuracy: 0.0980
Epoch 119/300
469/469 [==============================] - 1s 1ms/step - loss: 13.7439
- accuracy: 0.0987 -     val_loss: 13.7548 - val_accuracy: 0.0980
```

3 Hidden layer Neural Network with Keras

When using an additional hidden layer of size 64 on the *sigmoid* activation function, a reasonable upgrade of accuracy is noticeable while the consumed time is almost doubled. This is interesting when considering whichever we want to prioritize performance or This makes sense considering the amount of "work" produced by the neurons is way more important.

```
Loss : 0.0209621247
Accuracy : 99.38%
Total time : 256.58s
```

As opposed to the simple layer of neurons, the *relu* activation function reaches a skyhigh accuracy of 99.75% with even less time than the abovementioned one.
This underlines the fact that results can dramatically vary depending on the structure of the Neural Network even with similar architecture.

```
Loss : 0.0095428843
Accuracy : 99.75%
Total time : 244.73s
```

4 Multiclass Neural Network with Keras

As well as the "from-scratch" multiclass Neural Network implementation, the main difference compared to the 2-digits version is the accuracy that is now a bit lower.
But using **Keras**, it is now interesting to compare different architectures in terms of performance.
On the same batch size, we can obtain different results depending on the optimizer being used.
Here, with the **Adam** and **SGD** optimizers, results are pretty close and show no big differences.

```
Model with Adam optimizer on 100 epochs :
Loss : 0.0913420692
Accuracy : 88.98%
Total time : 73.10s

Model with SGD optimizer on 100 epochs :
Loss : 0.0915403962
Accuracy : 89.01%
Total time : 73.62s
```

But once we add a hidden 64-sized layer of neurons to our model, the differences are much more noteworthy.

```
Model with Adam optimizer on 100 epochs :
Loss : 0.0253374465
Accuracy : 97.01%
Total time : 121.88s

Model with SGD optimizer on 100 epochs :
Loss : 0.1396678090
Accuracy : 82.95%
Total time : 100.96s
```