Introduction to Artificial Intelligence (CS470): Assignment 2

Gabriel Schwab

April 12, 2023

1 Shallow Convolution Neural Networks

1.1 Convolution and Average Pooling using NumPy

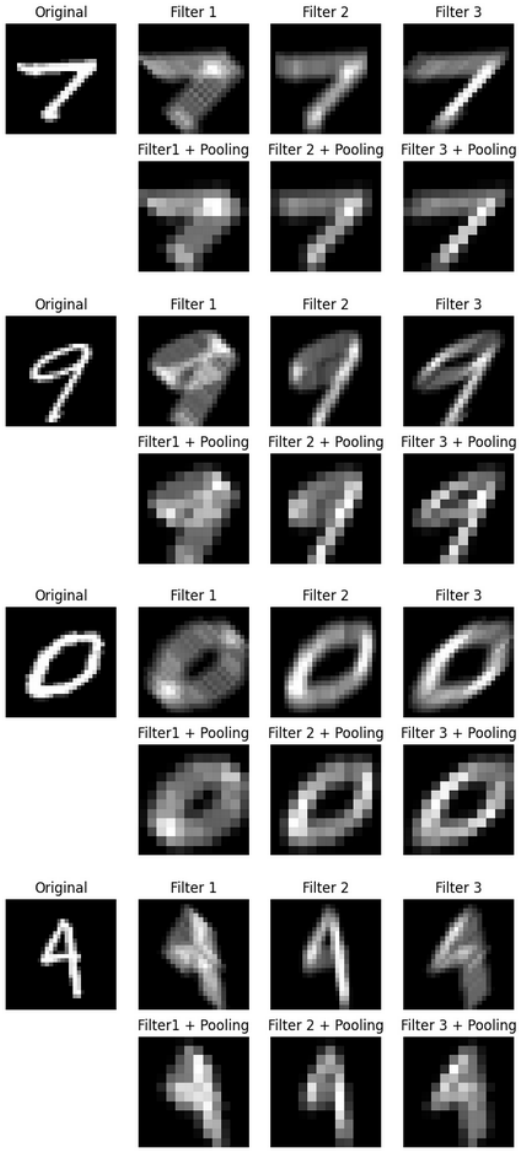

Figure 1: Visualization result of convolution with NumPy

The images from the convolution without average pooling filter the shapes while allowing to distinguish the shapes from the numbers. We notice that with average pooling, the numbers on the resulting images are coarser and appear as pixelated. This is the effect of max pooling which allows to extract more homogeneous features and not sharp details like edges.

1.2 Convolution and Average Pooling using PyTorch

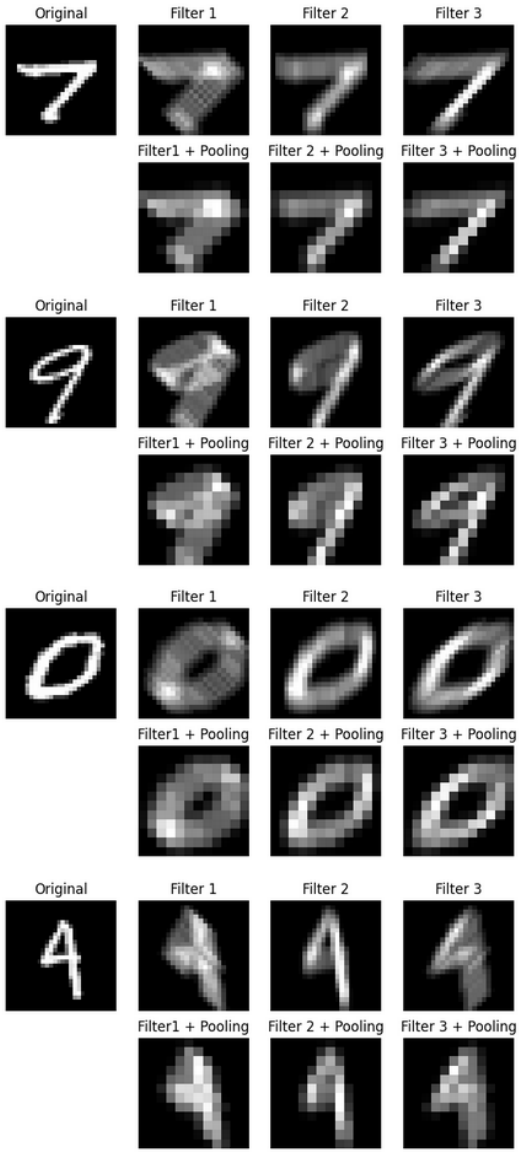

Figure 2: Visualization result of convolution with PyTorch

At a glance, there is no difference between the convolution and maxpooling results obtained with numpy and those with PyTorch.

To confirm our observations we calculate the Mean Square Error (MSE) between the results generated by the two methods. We obtain the following error which is very small and allows us to say that there is almost no difference between the two methods.

```
Average MSE: (2.3516035169895075e-15, 3.4800341422477088e-15)
```

Figure 3: Average MSE for convolution and convolution + maxpooling between NumPy and PyTorch implementation

2 Convolutional Neural Networks

2.1 A CNN with MaxPooling layers

Layer Number	Parameters Calculation and Result
Input	0
1	(3 * 3 * 1 + 1) * 32 = 320
2	0
3	(3 * 3 * 32 + 1) * 64 = 18,496
4	0
5	0
6	((64*22*22)*64 + 64) = 1,982,528
7	(64 + 1) * 32 = 2,080
8	(32 + 1) * 10 = 330
Total	2,003,754

Table 1: Parameter calculation and result for each layer in the CNN

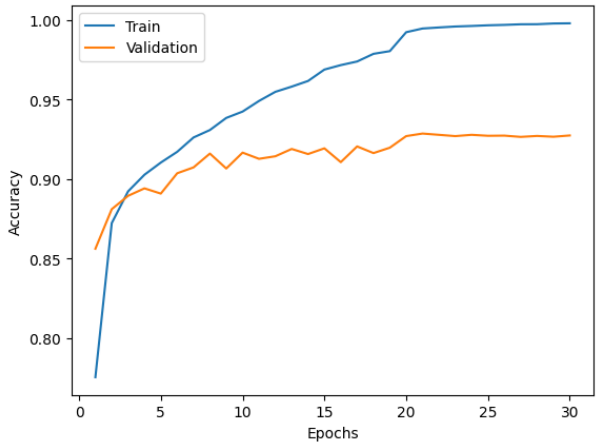

Figure 4: Graph of training and validation accuracies over 30 epochs

```
Accuracy on the 10000 test images: 92.35 %
```

Figure 5: Test accuracy on the test images

2.2 Prevention of Overfitting

In order to prevent overfitting, I tried to implement all the solutions suggested. After several tests, I could conclude that the best solution is the dropout with a probability of 0.5. This technique is particularly effective in the case of fully-connected linear layers, and not between convolutions, as shown in this paper about dropout [1].

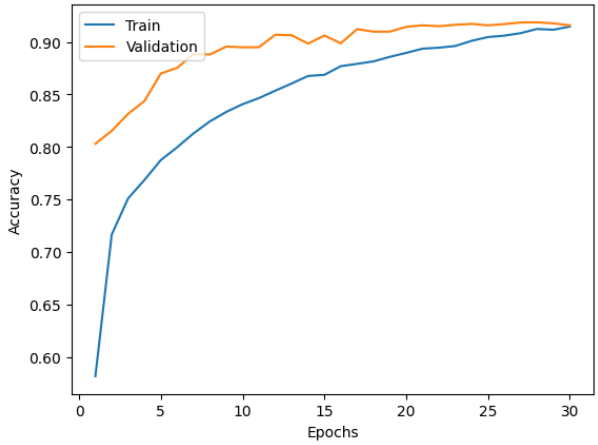

Figure 6: Graph of training and validation accuracy over 3O epochs with dropout

By only using this solutions we observe a significant improvement of overfiting as well as accuracy and test data set which reach 91.60%. We can see that the accuracy on validation test data set is even higher than the one for train data set.

We can even more improve this overfiting prevention by applying weights regularization after non-linear activation in the convolution fully-connected layers as suggested by this paper [2]. By combining this two solutions I was able to get the 92.77% of accuracy in test data set. I tried other solutions like data augmentation but the accuracy score on the data set test was lower.

3 Comparison of MLP and CNN

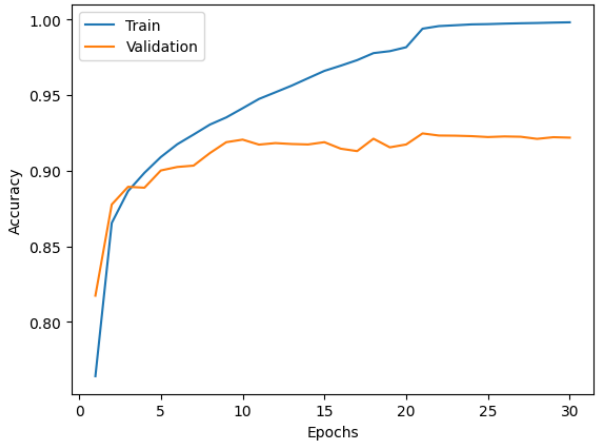

Figure 7: Graph of training and validation accuracy over 30 epochs of CNN

We observe that the CNN is more accurate (92.32% on test data set) than MLP (88.70% on test data set). This can be explained because convolutional neural network is better and more efficient fore image classification than basic mutli-layer perceptron. Moreover the CNN architecture is more complex and has more layers which contribute to improve his effiency. However, we notice on the

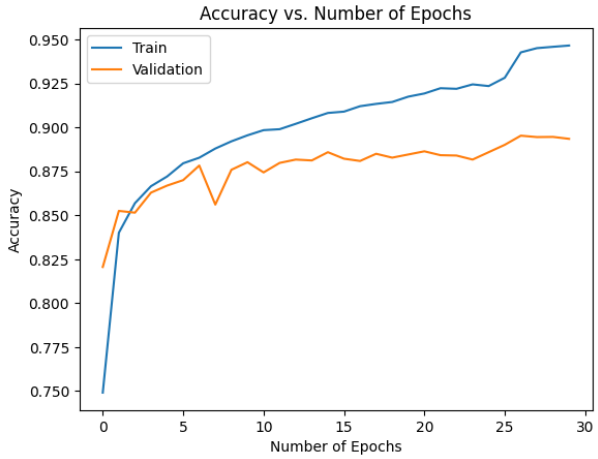

Figure 8: Graph of training and validation accuracy over 30 epochs of MLP

graphics that the MLP is less overfiting on the training data set. This is probably because he has a very basic architecture with only 2 layers. But this may let us think that there is a lot of optimizations that could be performed on the CNN that would even more improve his efficiency.

References

[1] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[2] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456, 2015.