Je vais pas utiliser une partie de MSP1 pour mon CdO. Mais je vais essayer d’implémenter du DP dans mon CdO. (c’est du

MSP1

Business Use Case – Deep Learning

I. Business Problem

In mid-March 2020, the French president announced a nationwide lockdown due to the covid-19 pandemic. This unprecedented measure, which began on March 17, has changed enormously the way we live, we work and we communicate.

The fact that we could not go outside accelerated digital communication. Many tools were offered quickly worldwide to tackle this problem, notably Teams, Zoom, etc. Thanks to these tools, meetings and lessons could continue as if nothing had happened. Students could even summit their homework online.

However, some subjects are more difficult than others to be done in this way. For example, to check if a student really understands and knows how to solve a mathematical problem, teachers cannot just ask students to key in the final answer. They need to see the detailed calculations that students use to derive their final answer. Therefore, teachers asked their students to scan their handwritten mathematical solutions and submit online.

We are all unsure if Covid will come back this coming winter. As such, the French Ministry of National Education wants to automate the scoring process to facilitate teachers’ workload and make sure its sustainability even after this Covid period. More concretely, it seeks for a system that can recognize students’ handwritten digits. It is decided to start with primary school students, whose calculations involve less mathematical symbols and who usually have worse handwriting for teachers to read.

What they need is to be able to scan through the PDFs or images submitted by students and the system automatically recognizes the numbers written in the detailed calculations. As such, teachers can spot calculation mistakes early on and mark them correctly.

In order to achieve this, the French Ministry of National Education seeks for our help to develop an AI system to recognize handwritten digits. However, due to the sanitary and economy crisis triggered by the Covid-19 pandemic, the Ministry wishes to minimize both the time costs and resources.

II. Proposed Technical Solutions

After studying their needs, we understand that it is a computer vision problem. Fortunately, there is a very famous dataset, MINST, to help us tackle this problem. Available for free, MNIST dataset consists of 70 thousands of already-labeled handwritten digits. With this dataset, we can tackle the problem as a supervised image classification. Since the dataset is available for free and is already labeled, it will help us to save a lot of time and resources as wished by the Ministry.

To date, we know that deep learning is achieving state-of-the-art results on recognizing and classifying images. There are many neural network architectures available to solve computer vision problems, including CNN, LeNet-5, VGG, ResNet, etc. For our problem of classifying digits, which is rather classic, we decided to use a classic architecture - LeNet-5. Once again, this will help us to gain time and resources.

A. Neural Network

Before going further to discuss different neural network architectures. Let us take a look at what a neural network is. In artificial intelligence, a neural network is a network of nodes (alias neurons) that performs a series of mathematical calculations in order to solve complicated problems.

Figure 1 shows a 4-layer neural network. The first layer is the input layer where it receives the input features for training. We do not count the input layer as a layer. Therefore, in Figure 1, there are only 4 layers, but not 5, with the first 2 layers having 5 nodes (or neurons), the third layer 3 nodes and the last layer, i.e., output layer, one node. All the layers between the input and output layers are called hidden layers. The number of nodes in the output layer depends on the problems we need to solve. For example, in our case, we need to classify digits ranging from 0 – 9. Hence, there will be 10 nodes in our output layer.

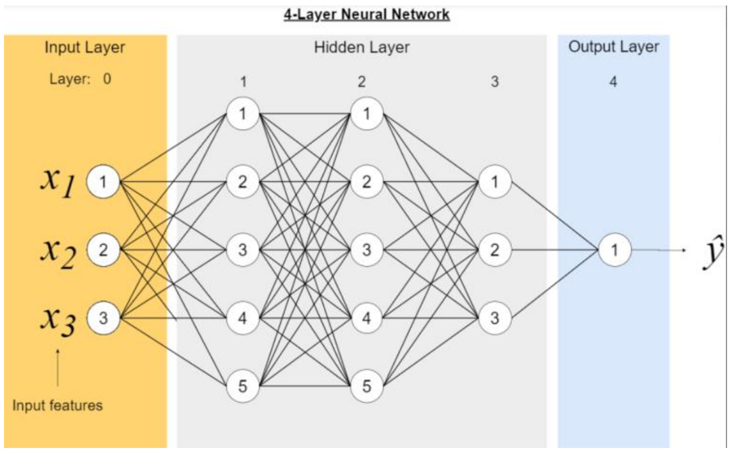


Fig. 1 Fully Connected 4-Layer Neural Network

To illustrate how it works, we use a simplified 2-layer neural network as shown in Figure 2 below. For the moment, let us consider the neurons are black boxes. All input features will go through each node (black box) of the hidden layer. Then each node in the hidden layer will turn the inputs into a numerical result, which will then be passed to each node (only one in this example) of the output layer, which in turn will output a prediction.

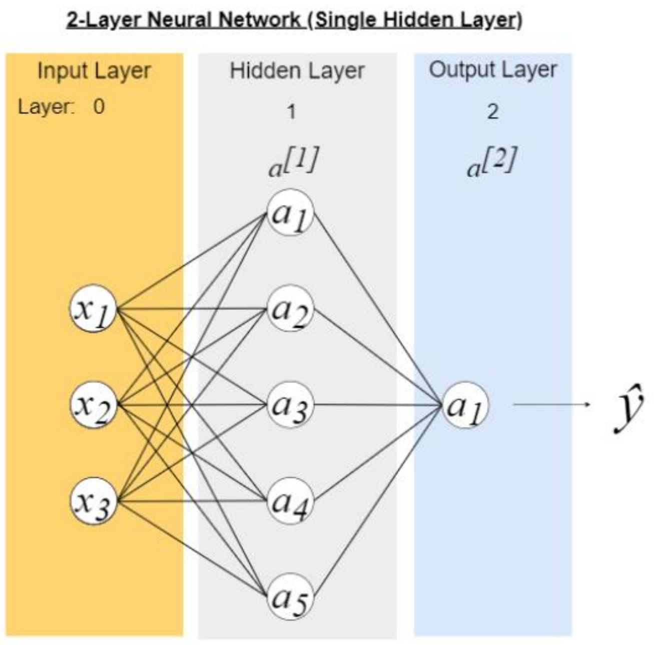
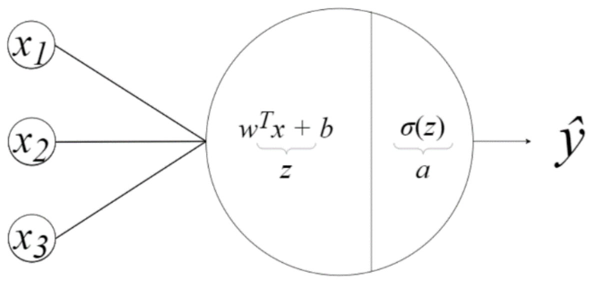


Fig. 2 Fully Connected 2-Layer Neural Network

So, what exactly happens inside the black boxes? Imagine a neural network that has only one neuron as shown in Figure 3.



x: input features
w: weight
b: bias
σ: activation function
$\hat{y}$: prediction

Fig. 3 Neuron Zoom in

The neuron gets x as inputs, transforms x with w and b:

$$z = w^{\mathrm{T}}x + b$$

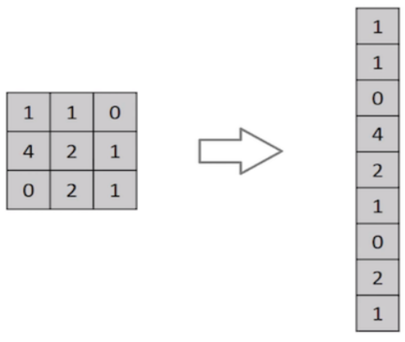
After getting z, the neuron will obtain an output a by using an activation function, in this case, sigmoid.

$$a = \sigma(z)$$

Since this is a single neuron neural network, a will be equal to the prediction, $\hat{y}$.

This left-to-right process is called forward propagation. However, for a neural network to learn to predict, we will need to adjust w and b by using backward propagation. Backward propagation goes from right to left. It finds the gradient by taking the derivative of the functions with respect to w and b. Then it uses this gradient to adjust the values of w and b, which is called gradient descent. We loop through this forward and backward propagation until we find a satisfactory prediction.

B. Convolutional Neural Networks (CNN)



Flattening of a 3x3 image matrix into a 9x1 vector

Convolutional Neural Network (CNN) is a deep neural network class, most commonly used to analyze visual imagery.

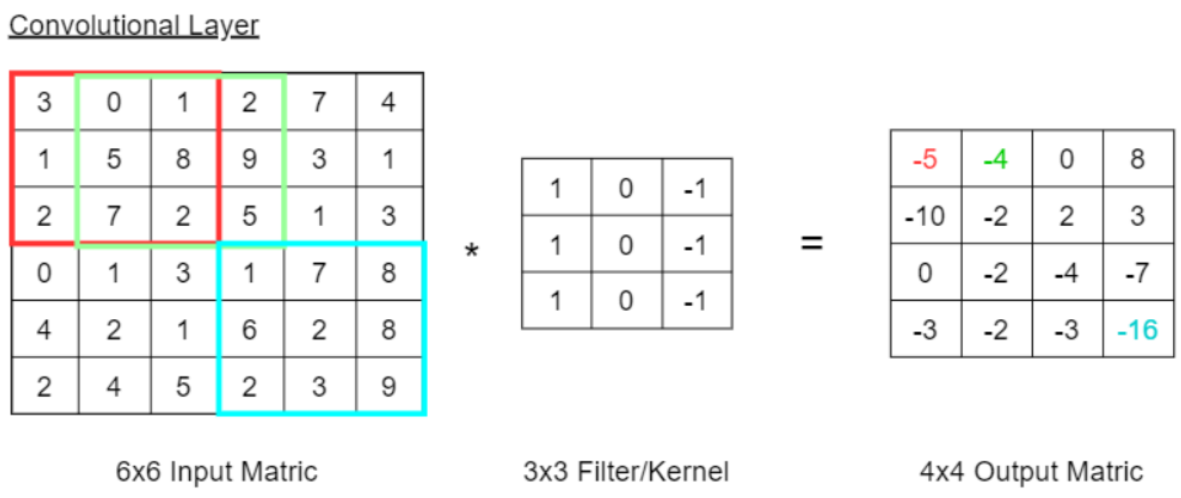
The classic neural network explained above requires flattening the image before fitting to a fully connected layer. However, doing so, it cannot capture the spatial dependencies in an image. CNN uses relevant filters (also known as kernel) to capture these and perform a better prediction with less parameters.

Convolutional Layer

In a convolutional layer, we use a filter to do convolutional operation on the input features. In the example below, we use a 3x3 filter on a 6x6 input matrix.

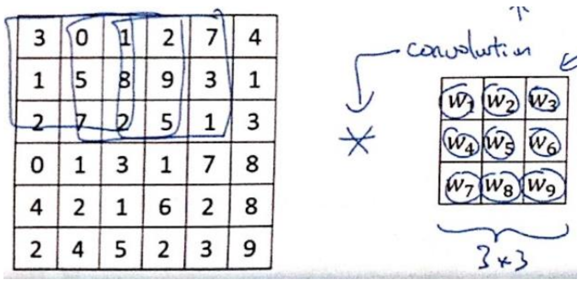
We hover the filter on the input matrix, starting from the top left corner where the red square is. Then multiplying the overlapping entries and sum them. For example, for the red square, it will be

3x1 + 1x1 + 2x1 + 0x0 + 5x0 + 7x0 + 1x(-1) + 8x(-1) + 2x(-1) = -5

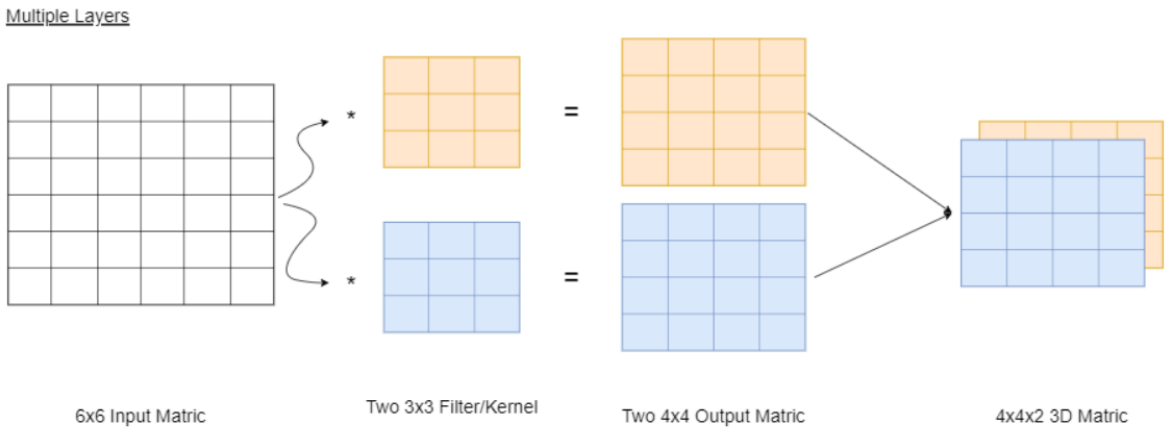


We fill the top left entry of the output matrix with the answer, -5. Then we shift the filter one step to the right, where the green square is, and repeat the operation. When it reaches to the right side, it will then go back to the left side but shift one step down, and so on and so forth, until it reaches to the bottom right corner, where the blue square is. The output matrix will be 4x4 (= 6 – 3 + 1).

Instead of handcrafting the filters, we can also use deep learning to learn the best weight of the filters.



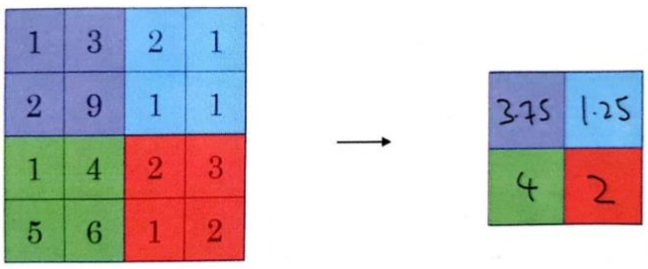
You can also use more than one filter. The number of output matrices will equal to the number of filters. And they will be concatenated to form a 3D matrix as shown on the graph below.



Pooling

Another kind of layer in CNN is pooling layer. Pooling is used to reduce the spatial size of the convolved feature. It helps to reduce computational power by reducing dimensionality.

There are two types of pooling: max and average pooling. On the right is an example of an average pooling with a 2x2 filter and a stride length of 2. Similar to the convolutional layer, the filter hovers over the top left corner. But this time, it returns an average of the four numbers over which it hovers. In max



pooling, it will returns the maximum value among the four numbers instead. Then it shifts two steps to the right (due to stride length = 2) and repeats the same operation, and so on and so forth, until it reaches the red square.

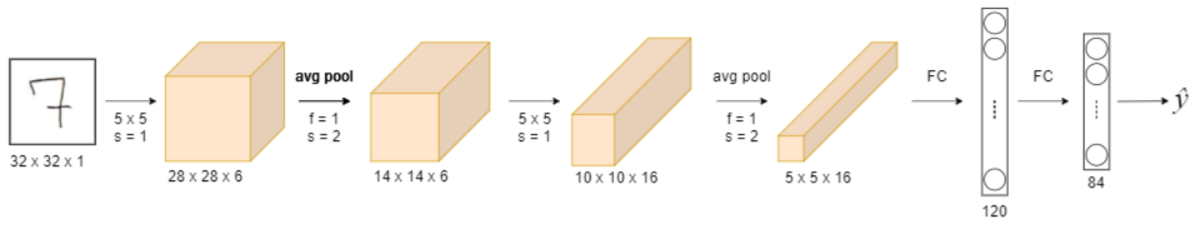
Compared to average pooling, max pooling removes noises to the maximum because it only keeps the maximum value. In practice, max pooling is used more often than average pooling.

C. LeNet-5

LeNet-5 is one of the classic CNN architectures, and a relatively simple one. This architecture is famous for classifying handwritten digits, which is ideal for our business problem. We believe that this architecture will give the most efficient results, in terms of computational power, time and accuracy.

LeNet-5 consists of:

1. Convolutional layer (5x5 filter, stride=2)
2. Average pooling layer (2x2 filter, stride=2)
3. Convolutional layer (5x5 filter, stride=2)
4. Average pooling layer (2x2 filter, stride=2)
5. Fully connected layer (120 nodes)
6. Fully connected layer (84 nodes)



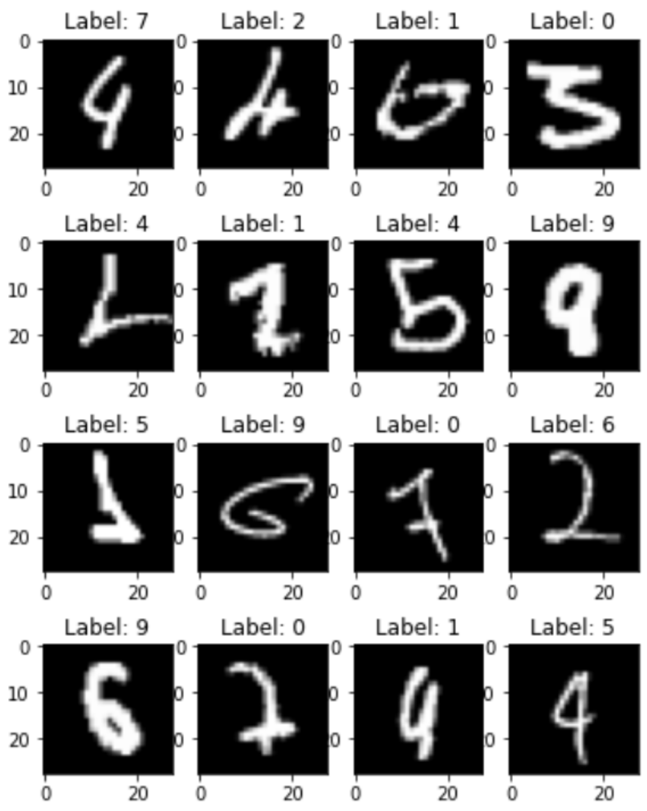
To improve the performance, we will make a few adjustments to the architecture:

1. Use max pooling instead of average pooling.
2. Use ReLU activation instead of Tanh which was used in the LeNet-5 paper.
3. Omit the fifth layer, fully connected layer (120 nodes). We obtained a better result without this layer.

III. Solutions Implementation

Please see the details of the technical implementation in this GitHub repository.

To conclude, we reached a very satisfactory accuracy of 99.1%. When we took a closer look at theose wrongly predicted images, we found that those are mostly images that were labeled wrongly. Considering this, our client, the French Ministry of National Education, very happily accepted our model and will have a nationwide implementation to all primary schools in order to better prepare the possible return of the Covid-19 in the coming winter.


Label: 7
Label: 2
Label: 1
Label: 0
Label: 4
Label: 1
Label: 4
Label: 9
Label: 5
Label: 9
Label: 0
Label: 6
Label: 9
Label: 0
Label: 1
Label: 5