TP 1 - cd : Convolutional Neural Networks

CHAHINE Marilyn 21517286

MACQUET Jean 3532408

1. Let n_x, n_y, n_z be the output size of the convolutional filter of padding p stride s and kernel size k. we have

$$n_x = \frac{x-k+2p}{s}$$
$$n_y = \frac{y-k+2p}{s}$$
$$n_z = 1$$

   Because the filters are computed for each feature z and then added together to make only one image. For one convolutional filter, the number of weight is $k^2 * z + 1$ (one kernel and one biais). for a fully connected layer that produce an output size of $n_x \times n_y \times n_z$ we would have needed $\frac{(x-k+2p)(y-k+2p)}{s^2}xyz$

2. Cnn have the advantages of having less weights than fully connected layer for the same output and input size. this is convenient to treat data of large size, wich is the case for images. Furthermore, beacause the weight are shared over the neurons (the same filter pass over each input) it can captures easily capture dependencies a fully connected layer would have more difficulty to learn, like shapes, wich are related to correlation in the input neuron (the dimensions of the data).

3. Pooling is used to lower the dimension. By doing so, the network can higher level correlation inside the data, and be invariant to small perturbation of the high dimensional input.

4. with a bigger image, the convolutional layer and pooling layer can be applied, beacause they dont need a specific size of input (they only need a kernel size and a depth). On the contrary, the fully connected layer need a fixed size of input because it is made linear function of shape (input size, hidden size)

5. If we only want to focus on the central output pixel of a convolution, we can modify the filter such that it covers the whole image, so the size of the filter would correspond to the size of the image. Each pixel of the image will have a weight in this convolution, equivalently to a fully connected layer.

6. this layer would have $5 \times 5 \times B \times C \times H \times W$ parameters if we consider a padding and a stride that make the filter going through each pixel. This is beacaus eeach pixel has its own convoluttional kernel of size 5×5

7. For the first layer, if we consider a stride of one and a padding that make the kernel pass the same amout of time on each pixel of the original picutre (picture 0), the receptive field is the size of the kernel, which is $k \times k$. if we visualize the output of the firsrt layer as a picture (picture 1), the receptive filed has the shape of the kernel on the image. Thus, after a second layer of convolution, each pixel of picture 1 would have a receptive field of shape of the kernel. In picture 2, the one produced after the second kernel, the shape would be $(k + k - 1)^2 = (2k - 1)^2$

8. To preserve the same output size as the input, we need to use a stride of 1 and a padding of 2 when the kernel size is 5.

9. To downsample the feature maps by a factor of two, with a pooling kernel size of 2×2, we need a stride of 2, and no padding

10. Let's do the computation for the first convulational layer. The size is $32 \times 32 \times 3$ The conv layer has padding and stride choosen for the output size to be same as the input one. Because there are 32 of those convolution, the output size is $32 \times 32 \times 32$. to get the weight, we have the kernel weights : $5 \times 5 \times 32$. and the bias weight 32 wich make $5 \times 5 \times 32 + 32 = 2432$.

layer name	output size	number of weights
conv1	(32, 32, 32)	2 432
pool1	(16, 16, 32)	0
conv2	(16, 16, 64)	51 264
pool2	(8, 8, 64)	0
conv3	(8, 8, 64)	102 464
pool3	(4, 4, 64)	0
fc4	(1000)	1 025 000
fc5	(10)	10,010

We observe that most of the weights come from the fully connected layer

11. The model has around 1.2 million parameters for 50,000 training examples. the ratio is 24 parameters per exemple, which is not good : it could lead to overfitting

12. There a lot more parameters to learn than for the BoW and the SVM approach

13. cf notebook

14. Inside the main loop there is a condition on the loss calculation : during training, the loss is computed with backpropagation and parameters update. During evaluation, we don't do backpropagation.

15. cf notebook

16. A large learning rate can cause divergence, a small learning rate can cause slow convergence and thus underfitting. A large batch size causes fast convergence (in number of iterations) because of a smoother gradient descent, but it increases the chances of being stuck in a local minima. On the other hand, a small batch size causes a noisy gradient but help generalization.

17. Precision is 13% at the start of the first epoch. This is caused by the choice of the initialization parameters, which is random.

18. With `batch_size = 128` and `lr = 0.1` with `epochs = 20` we clearly see that the train loss is decreasing but that the test loss is exploding and that the test accuracy is plateauing. This phenomenon is typical of overfitting

19. Standardization must be done on the training and the test set. It has a a relatively good effect on learning : the test loss is still far from the train loss after 20 epochs, but the train loss converge faster. Overall, we get the same a result as before, but faster

20. The average image must only be calculate on the training examples because we are not supposed to access the test exemples : in practice we apply the same model to different test data. if we calculate the average image for the validation data, the resulting normalization will not be the same that the one on which the model has been train, and the prediction will not be good.

21. Bonus

22. As shown in Figure 1, after 20 epochs, the test loss is under 0.4 and very close to the train loss. We are neither in underfitting nor overfitting : the model is at last learning something and extends correctly to the test. This is because random cropping and horizontal flipping increase data diversity and thus improve generalization.

23. It is usable for shape recognition and object detection, when the classification that is made is stable under symetry. if we want to learn letters, numbers, or in general relative positions (like left and right hand) we cannot use symmetry

24. Basic transformations only cover limited variability like position, shifts and cannot simulate all real-world changes. We can add transformations such as lighting, rotation, scale but we cannot add element in the environment. For example, if we want to detect car, we cannot add a human next to the car for data augmentation.

25. We can add lighting transformation for example.

26. As shown in Figure 1,the loss is decreasing in a similar way that in the previous case, but the accuracy at the end seem less noisy. The accuracy after 20 epochs is : 78.99%. It's better that the previous one. The test loss is closer to the train loss, but the test loss is still increasing at the end : it seems like it tends to slightly overfit.

27. At the begining, the learning rate is big, allowing rapid exploration. At the end, the learning rate gets finer, allowing the model to train more slowly and increase its precision instead of going at the same rate and miss the right minima

28. Bonus

29. As shown in Figure 1, after 20 epochs, we get an accuracy of 80.22% wich is better than without dropout. It is also clear that the test loss is more controlled Dropout reduces the overfitting we had at the end the decreasing of the test loss is more stable. The model converges more slowly but achieves better generalization : test accuracy and train accuracy stay really close to each other.

30. Regularization is term referring to methods that prevent the model from overfitting.

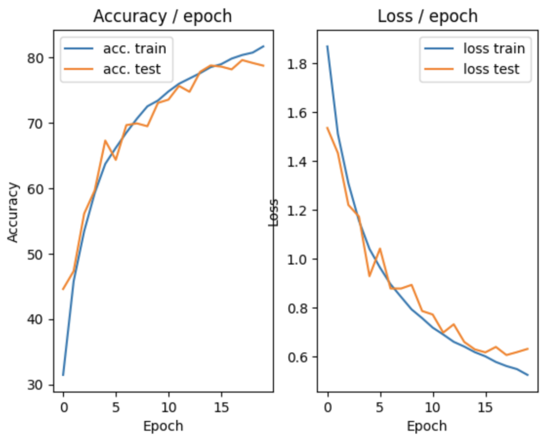
31. Dropout could be interpreted as a way to "forget" some of the thing that where learn. If we take the example of a human reasoning, we always say learning is not remembering : it is not good to remember anything because it can prevent us to generalize to thing we do not know.

32. The dropout rate determines the ratio of neurons that are randomly deactivated. A high dropout rate will lead to underfitting, and a small rate could lead to overfitting.

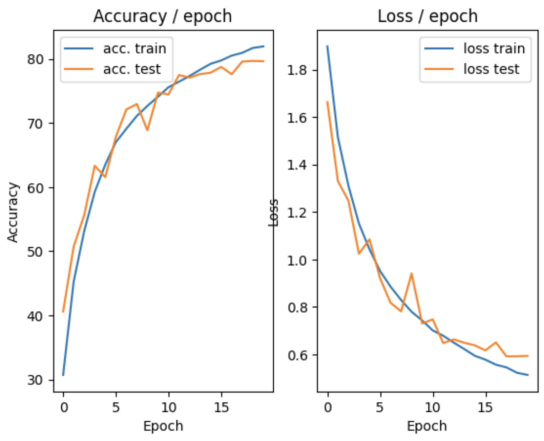
33. Dropout is not used during test because the model must use its entire learning capacity to predict. It is only used during training.

34. As shown in Figure 1, the accuracy obtained after 20 epochs in the final experiment with batch normalization added after each convolution layer is 78% which is less than before, but observing the evolution of the loss we can see it is steeper the loss decrease faster. Batch normalization normalizes activations within each batch, reducing internal shifts. It has the same effect that the normalization we made on the train data (but this time batch-wise) : it increases the learning speed.

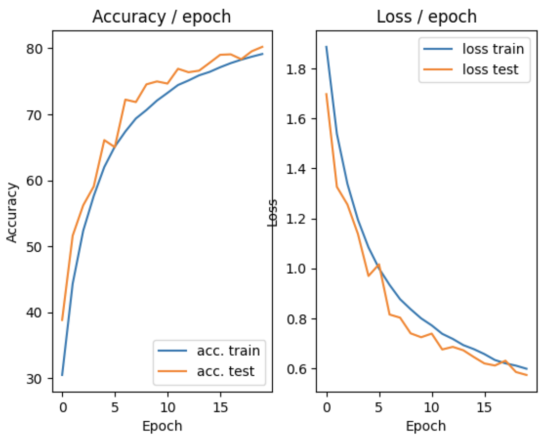
Figure 1: **Accuracy and Loss of the Different Implementations**



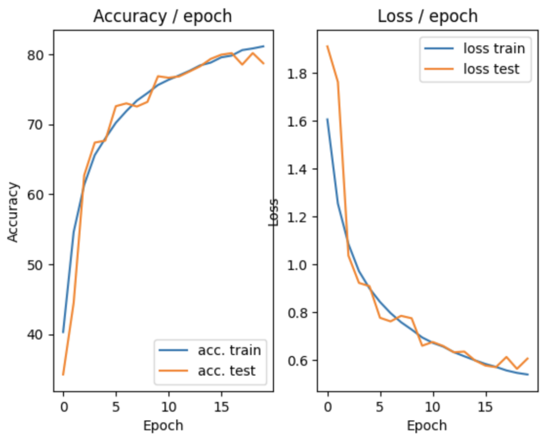
(a) Part 2: Data augmentation



(b) Part 3: Lr scheduler



(c) Part 4: Dropout



(d) Part 5: Batch normalization