6 Convolutional neural networks

Exercise 39 — What are the parameters and hyperparameters of Conv, MaxPooling, Dense and Dropout layers. For Conv explain what's stride and padding. How to set padding to ensure that the input and output of Conv layer will have the same width and height for stride $S = 1$ and $S = 2$? (3p)

Exercise 40 — What's filter (kernel) in CNN and what determines its width, height and depth? Suppose an input to some Conv layer has size $16 \times 16 \times 20$, where the last number is depth.

1. Define some filter and find the number of connections from a single element of the feature map to the input. Explain what's locally-connected layer.
2. If all elements of the feature map share the same filter what is the total number of parameters to learn? Explain what is parameter sharing and when it might be useful. (3p)

Exercise 41 — Assume that Conv layer accepts an input of size $W_1 \times H_1 \times D_1$ and let the number of filters be K, filter width and hight be F, stride be S and padding be P. (3p)

1. What's the shape of the output $W_2 \times H_2 \times D_2$ in terms of the input shape and hyperparameters? What are constraints for hyperparameters?
2. What's the number of parameters to be set per filter and in total if we use parameter sharing?
3. How we obtain dth depth slice of size $W2 \times H2$ in the output?

Exercise 42 — What's might be the role of a filter with width and hight equal to 1? (1p)

Exercise 43 — Explain the role of MaxPooling layer. Give a practical example. (1p)

Exercise 44 — Explain why feature scaling of the input features is important. (1p)

Exercise 39

Number of parameters in a given layer is the count of "learnable" (assuming such a word exists) elements for a filter aka parameters for the filter for that layer.

Model parameters are the properties of the training data that are learnt during training by the classifier or other ml model. For example in case of some NLP task: word frequency, sentence length, noun or verb distribution per sentence, the number of specific character n-grams per word, lexical diversity, etc. Model parameters differ for each experiment and depend on the type of data and task at hand.

Model hyperparameters, on the other hand, are common for similar models and cannot be learnt during training but are set beforehand. A typical set of hyperparameters for NN include the number and size of the hidden layers, weight initialization scheme, learning rate and its decay, dropout and gradient clipping threshold, etc.

- **Conv:**

Parameters:

- Weights control the signal (or the strength of the connection) between two neurons. In other words, a weight decides how much influence the input will have on the output.
- Biases are like the intercepts added in a linear equation. It is an additional parameter which is used to adjust the output along with the weighted sum of the inputs to the neuron.

Hyperparameters:

- Filters, in image processing, a filter, kernel, convolution matrix, or mask is a small matrix used for blurring, sharpening, embossing, edge detection, and more. This is accomplished by doing a convolution between the kernel and an image. Its height, width and depth are determined only by the creator of the model, as they are hyperparameters.
- Kernel size, the kernels in the convolutional layer, are the convolutional filters. Actually no convolution is performed, but a cross-correlation. The kernel size here refers to the Width x Height of the filter mask.
- strides
- padding
- activation

• **MaxPooling:**

Hyperparameters:

- pool size
- strides
- padding

• **Dense:** normal fully connected layer in a neuronal network.

Parameters:

- weights
- biases

Hyperparameters:

- units
- activation

• **Dropout:** In short, a dropout layer ignores a set of neurons (randomly) as one can see in the picture below. This normally is used to prevent the net from overfitting.

Hyperparameters:

- rate

Padding is a term relevant to convolutional neural networks as it refers to the amount of pixels added to an image when it is being processed by the kernel of a CNN. We refer to the number of rows and columns traversed per slide as the **stride**.

Stride with which we slide the filter. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2 (or uncommonly 3 or more, though this is rare in practice) then the filters jump 2 pixels at a time as we slide them around. This will produce smaller output volumes spatially.

$$H_{\text{out}} = \left\lfloor \frac{H_{\text{in}} - F_H + 2P}{S} \right\rfloor + 1$$

For S = 1:

$$H_{\text{out}} = \left\lfloor \frac{H_{\text{in}} - F_H + 2P}{1} \right\rfloor + 1 = \lfloor H_{\text{in}} - F_H + 2P \rfloor + 1 = H_{\text{in}} - F_H + 2P + 1$$

$$P = \frac{H_{\text{out}} - H_{\text{in}} + F_H - 1}{2}$$

Cuz we want H_{out} to be equal H_{in}, we have:

$$P = \frac{F_H - 1}{2}$$

For S = 2:

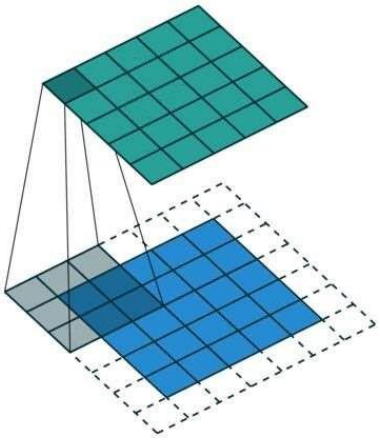
$$H_{\text{out}} = \left\lfloor \frac{H_{\text{in}} - F_H + 2P}{2} \right\rfloor + 1$$

$$2H_{out} = H_{in} - F_H + 2P + 2$$

$$P = \frac{2H_{\text{out}} - H_{\text{in}} + F_H}{2} - 1$$

Cuz we want H_{out} to be equal H_{in}, we have:

$$P = \frac{H_{\text{in}} + F_H}{2} - 1$$



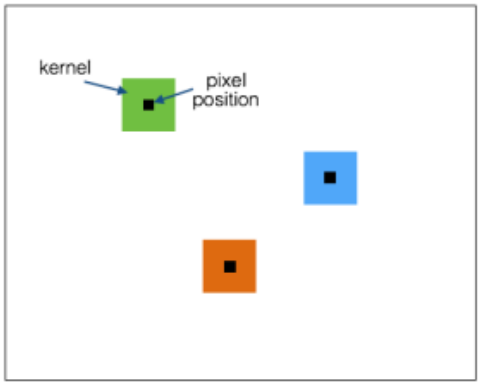
Exercise 40

In image processing, a filter, kernel, convolution matrix, or mask is a small matrix used for blurring, sharpening, embossing, edge detection, and more. This is accomplished by doing a convolution between the kernel and an image.

Its height, width and depth are determined only by the creator of the model, as they are hyperparameters.

1. Input to Conv layer has size $16 \times 16 \times 20$, if we assume that the filter size is 3×3, every single element in the Conv layer will have $3 * 3 * 20 = 180$ connections of the feature map to the input.

We can say that Convolutional layers are locally connected layers with shared weights. Let's say we want train 256 filters of size 5×5 at the current stage of the network.Here, each filter learns to detect different aspects of the image. The input image is of size, say, 128 x 128. We can fit 124 x 124 filters in that image grid. As seen from the figure below, the different colors indicate different filters:



In a convolutional layer, we just need to train 5 x 5 x 256 number of parameters. But if we are dealing with locally connected layers with unshared weights, we will be dealing with 5 x 5 x 256 x 124 x 124.

2. Let assume that we have 3 × 3 filter, S = 1 and P = 1

$$3 \times 3 \times 20 + 1 = 181$$

Parameter sharing is used in Convolutional Layers to control the number of parameters. It turns out that we can dramatically reduce the number of parameters by making one reasonable assumption: constrain the neurons in each depth slice to use the same weights and bias. If one feature is useful to compute at some spatial position (x,y), then it should also be useful to compute at a different position (x_2,y_2).
The equation above is for shared parameter. Let's see what will happens for unshared parameters.
Number of neurons/units within the conv layer:

$$16 \times 16 \times 1 = 256$$

Number of training parameters within the conv layer (unshared):

$$256 \times (3 \times 3 \times 20 + 1) = 46\,336$$

Exercise 41

1. We have then input size $W_1 \times H_1 \times D_1$, we require four hyperparameters:
   K – number of filters
   F_w – filter width
   F_H – filter high
   S – stride
   P – padding
   We produce then output size $W_2 \times H_2 \times D_2$, where :

$$W_2 = \frac{(W_1 - F + 2P)}{S + 1}$$
$$H_2 = \frac{(H_1 - F + 2P)}{S + 1}$$

As we can see width and height are computed equally by symmetry

$$D_2 = K$$

2. Number of parameters to be set per filter (parameter sharing):

$$F_w \times F_H \times D_1 + 1$$

Number of parameters in total (parameter sharing):

$$(F_w \times F_H \times D_1 + 1) \times K$$

3. We obtain the d-th depth slice in the output of size $W_2 \times H_2$ as the result of performing a valid convolution of the d-th filter over the input size with a stride of S and then offset by d-th bias.

Exercise 42

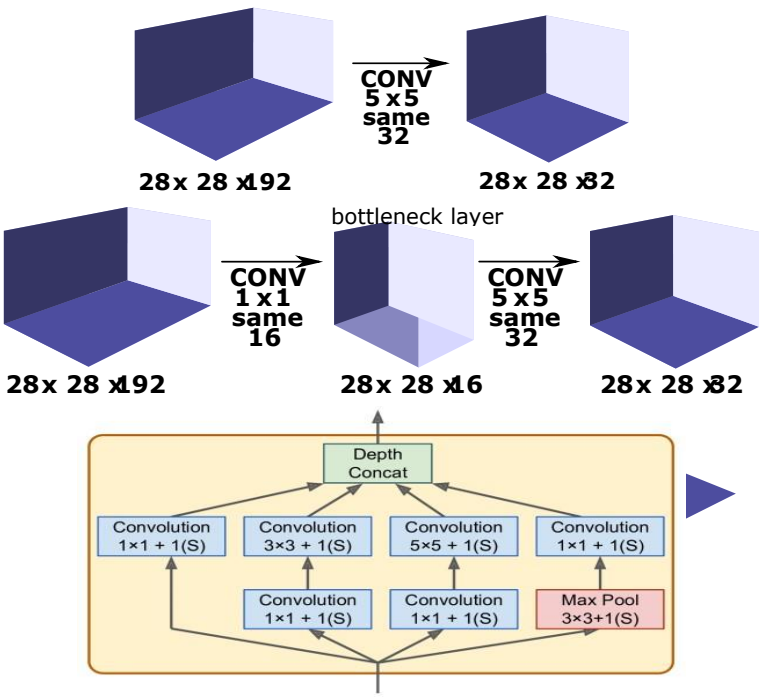
Suppose that I have a conv layer which outputs an (N, F, H, W) shaped tensor where:

- N is the batch size
- F is the number of convolutional filters
- H, W are the spatial dimensions

Suppose the input is fed into a conv layer with F_1 1x1 filters, zero padding and stride 1. Then the output of this 1x1 conv layer will have shape (N, F_1, H, W).

So 1x1 conv filters can be used to change the dimensionality in the filter space. If $F_1 > F$ then we are increasing dimensionality, if $F_1 < F$ we are decreasing dimensionality, in the filter dimension.

A 1x1 convolution simply maps an input pixel with all it's channels to an output pixel, not looking at anything around itself. It is often used to reduce the number of depth channels, since it is often very slow to multiply volumes with extremely large depths.



$$28 \times 28 \times 32 \times 5 \times 5 \times 192 = 120\,422\,400 \text{ multiplications}$$

$$28 \times 28 \times 16 \times 1 \times 1 \times 192 + 28 \times 28 \times 32 \times 5 \times 5 \times 16 = 12\,443\,648 \text{ multiplications}$$

Exercise 43

Max pooling is a type of operation that is typically added to CNN following individual convolutional layers. When added to a model, max pooling reduces the dimensionality of images by reducing the number of pixels in the output from the previous convolutional layer.

Each convolutional layer has some number of filters that we define with a specified dimension and that these filters convolve our image input channels. When a filter convolves a given input, it then gives us an output. This output is a matrix of pixels with the values that were computed during the convolutions that occurred on our image. We call these output channels. For example, let's recall that we have 26×26 output channel produced by using 3×3 filter:

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.4	0.6	0.7	0.5	0.4	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.3	0.6	1.2	1.4	1.6	1.6	1.6	1.6	1.9	1.9	2.2	2.3	2.1	2.0	1.7	0.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.5	1.2	1.8	2.6	2.7	3.0	3.0	3.0	3.0	3.4	3.5	3.8	4.0	3.7	3.6	3.2	2.3	1.5	0.5	0.1	0.0	0.0	0.0	0.0	0.0	0.0
1.1	2.1	3.2	4.2	4.4	4.7	4.7	4.5	4.2	4.0	3.8	3.9	3.9	4.1	4.5	4.7	4.1	3.1	1.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0
1.1	2.0	3.1	3.6	3.3	3.2	3.2	3.1	2.9	2.7	2.5	2.5	2.5	2.7	3.0	3.9	4.4	4.1	2.9	1.4	0.3	0.0	0.0	0.0	0.0	0.0
0.9	1.4	2.1	2.2	1.8	1.7	1.7	1.5	1.1	0.8	0.5	0.5	0.5	0.8	1.3	2.4	3.7	4.5	4.0	2.4	1.0	0.0	0.0	0.0	0.0	0.0
0.1	0.3	0.3	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	1.3	2.8	4.2	4.7	2.8	1.6	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	1.2	2.9	3.9	5.1	3.1	2.2	0.1	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.4	1.0	1.3	1.6	1.9	2.4	3.7	4.4	5.2	3.8	2.5	0.7	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.5	1.1	1.7	2.3	2.7	3.0	3.4	3.7	4.6	4.9	5.2	4.1	2.5	1.2	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.1	0.7	1.3	1.9	2.6	3.2	4.0	4.4	4.8	4.4	4.2	4.5	4.8	5.2	4.5	2.7	1.6	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.4	1.0	1.8	2.6	3.3	3.8	3.9	3.8	3.6	3.4	3.0	2.9	3.6	4.1	5.0	3.8	2.5	1.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.8	1.7	3.0	3.5	3.7	3.3	3.0	2.5	2.2	1.9	1.3	1.3	2.4	3.3	4.8	3.4	2.3	0.6	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.9	2.0	2.7	3.2	2.6	1.8	1.3	0.7	0.4	0.1	0.0	0.4	2.2	3.3	4.6	3.0	2.0	0.2	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.7	1.4	1.6	1.7	0.7	0.2	0.0	0.0	0.0	0.0	0.0	0.8	2.5	3.7	4.2	2.6	1.5	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.1	0.5	0.2	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.7	1.7	3.3	4.0	3.6	2.2	0.8	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.3	2.3	4.0	3.9	2.8	1.6	0.2	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	2.3	3.1	4.5	3.4	2.0	0.8	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	2.6	3.4	3.8	2.5	1.2	0.2	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2	2.0	2.8	2.4	1.5	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.3	2.0	1.3	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

max pooling is added after a convolutional layer. This is the output from the convolution operation and is the input to the max pooling operation.

After the max pooling operation, we have the following 13×13 output channel:

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.3	0.6	0.7	0.4	0.0	0.0	0.0	0.0	0.0
1.2	2.6	3.0	3.0	3.4	3.8	4.0	3.6	2.3	0.5	0.0	0.0	0.0
2.1	4.2	4.7	4.7	4.2	3.9	4.1	4.7	4.4	2.9	0.3	0.0	0.0
1.4	2.2	1.8	1.7	1.1	0.5	0.8	2.4	4.5	4.7	1.6	0.0	0.0
0.0	0.0	0.0	0.0	0.1	1.0	1.6	2.4	4.4	5.2	2.5	0.0	0.0
0.0	0.0	0.1	1.3	2.6	4.0	4.8	4.4	4.9	5.2	2.7	0.0	0.0
0.0	0.0	1.7	3.5	3.8	3.9	3.6	3.0	4.1	5.0	2.5	0.0	0.0
0.0	0.0	2.0	3.2	2.6	1.3	0.4	0.8	3.7	4.6	2.0	0.0	0.0
0.0	0.0	0.5	0.5	0.0	0.0	0.0	2.3	4.0	3.6	0.8	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.9	3.4	4.5	2.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	1.2	2.8	2.4	0.3	0.0	0.0	0.0

Exercise 44

Machine learning algorithm just sees number — if there is a vast difference in the range say few ranging in thousands and few ranging in the tens, and it makes the underlying assumption that higher ranging numbers have superiority of some sort. So these more significant number starts playing a more decisive role while training the model.

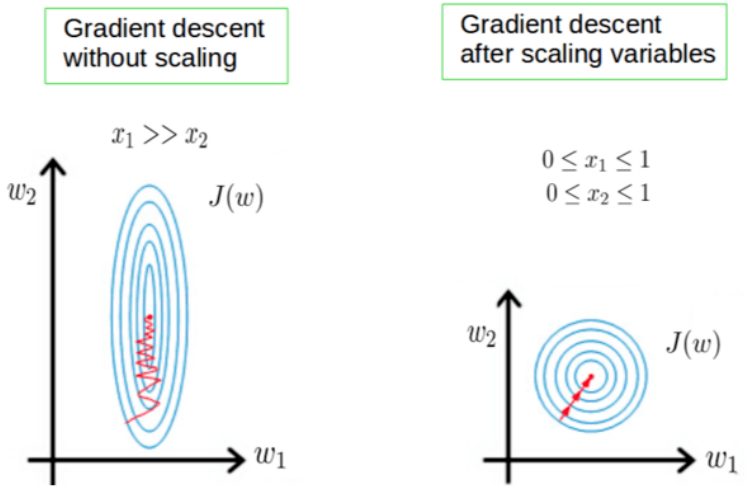
The machine learning algorithm works on numbers and does not know what that number represents. A weight of 10 grams and a price of 10 dollars represents completely two different things — which is a no brainer for humans, but for a model as a feature, it treats both as same.

Suppose we have two features of weight and price, as in the below table. The "Weight" cannot have a meaningful comparison with the "Price." So the assumption algorithm makes that since "Weight" > "Price," thus "Weight," is more important than "Price."

Name	Weight	Price
Orange	15	1
Apple	18	3
Banana	12	2
Grape	10	5

So these more significant number starts playing a more decisive role while training the model. Thus feature scaling is needed to bring every feature in the same footing without any upfront importance. Interestingly, if we convert the weight to "Kilograms," then "Price" becomes dominant.

Another reason why feature scaling is applied is that few algorithms like Neural network gradient descent converge much faster with feature scaling than without it.



One more reason is saturation, like in the case of sigmoid activation in Neural Network, scaling would help not to saturate too fast.