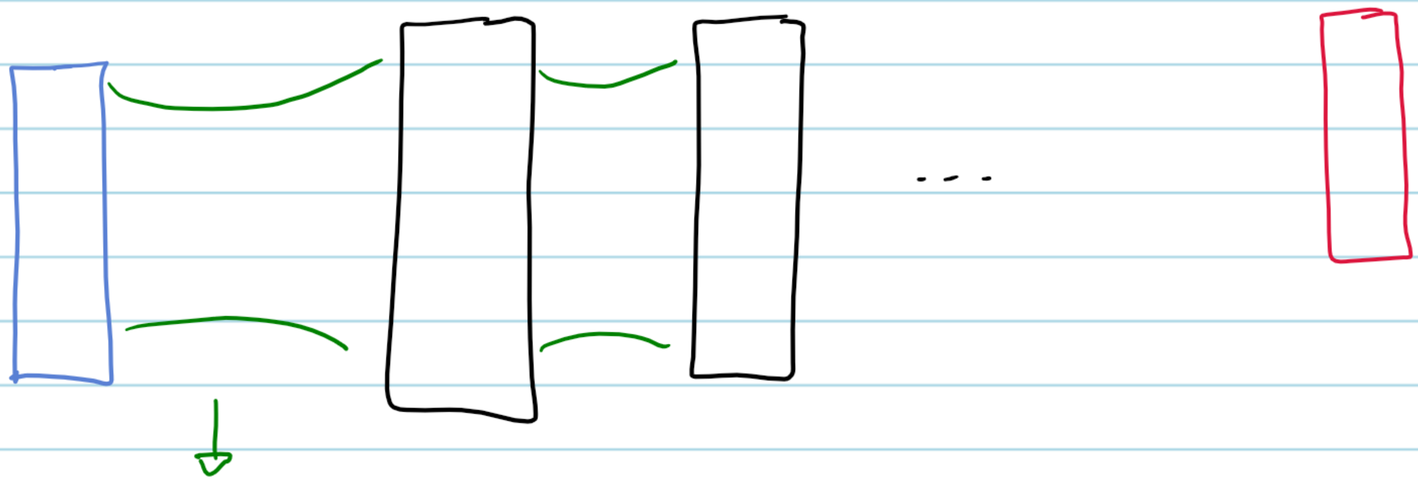
شبکه عصبی کانولوشنال

نمای شبکه عصبی تا اینجا یادگرفتیم.



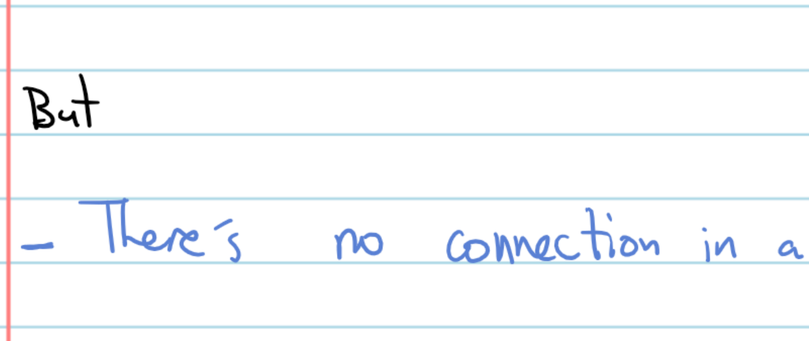
Fully connected:

- All the nodes between the two layers are connected.

But

- There's no connection in a layer.
- There's no self-connection.
- It's rather a dense graph → Too many connections.

Consider the example of an image.



How many parameters? $\sim 10^5$

What if the input image was 100×100?

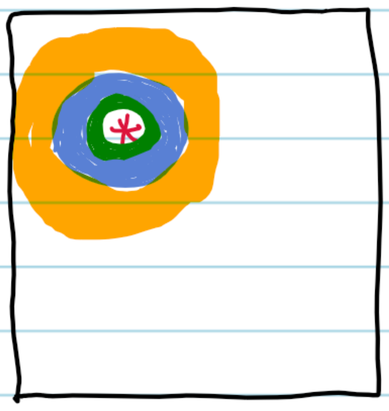
or a Cube $(32 \times 32 \times 32)$?

$\sim 10^6$

→ Too many parameters mean

* Long training process
* Overfitting

Some of you dealt with image data.



Each point is assigned this properties

Density in the green zone → f_1

" in the blue " → f_2

" " " Orange ". → f_3

Model: $F(f_1, f_2, f_3)$

e.g. $w_1 f_1 + w_2 f_2 + w_3 f_3$

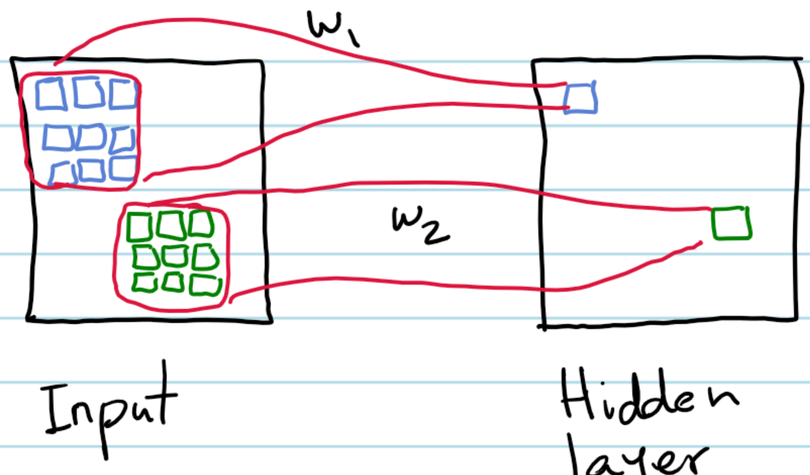
There are two assumptions behind this approach:

* Locality: What happens to a sample is mostly affected by its neighbours.
* Translational invariance: The way our model depends on f_1, f_2, f_3 is the same for all points.

We don't expect it to depend on f_1 for sample 1 with w_1 & for sample $\overline{w}_1$.

We want to use the same intuition/assumptions to simplify the NN.

Imagine that we keep the hidden layer of the same shape as the input image



This indicates that the blue node in the hidden layer is only connected to the blue spots in the input & the green node only to the green spots.

Locality

Question: How many params in the first layer?

$$n_w^{[1]} \times n_H^{[1]} \times 9 + n_w^{[1]} \times n_H^{[1]} = 10\ n_w^{[1]} \times n_H^{[1]}$$

$\times 9$: Each point connected to 9 points.

$+ n_w^{[1]} \times n_H^{[1]}$: Bias

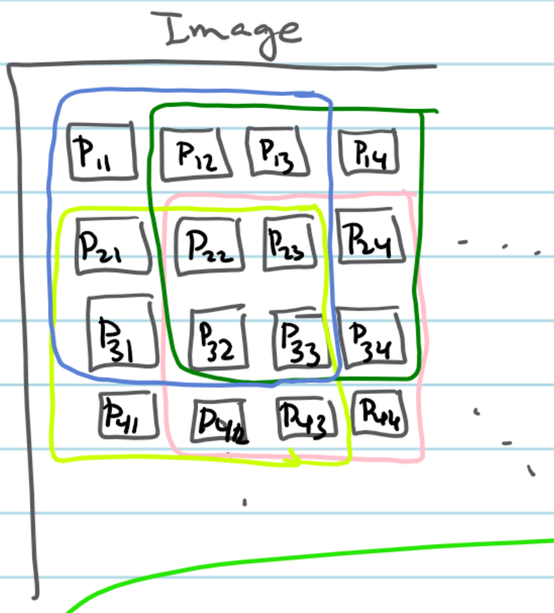
This is indep. of the input image size. ($32 \to 1000$ pixel)

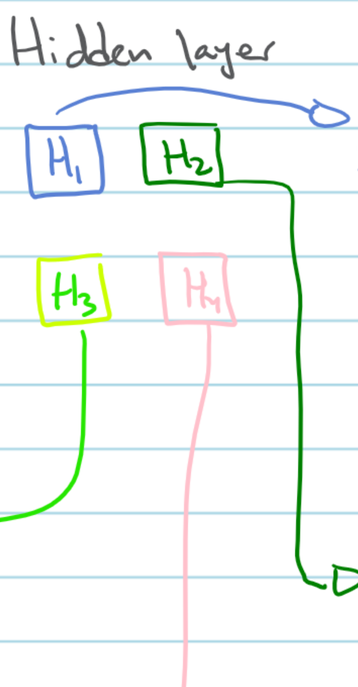
But we go even further:

Translational Invariance : It is natural to expect the weights for the blue node to be the same as the one for the green node.

params $\to$ 10 (roughly)

Let's make this notion more precise:





$$H_1 = W_1 P_{11} + W_2 P_{12} + W_3 P_{13} + W_4 P_{21} + W_5 P_{22} + W_6 P_{23} + W_7 P_{31} + W_8 P_{32} + W_9 P_{33}$$

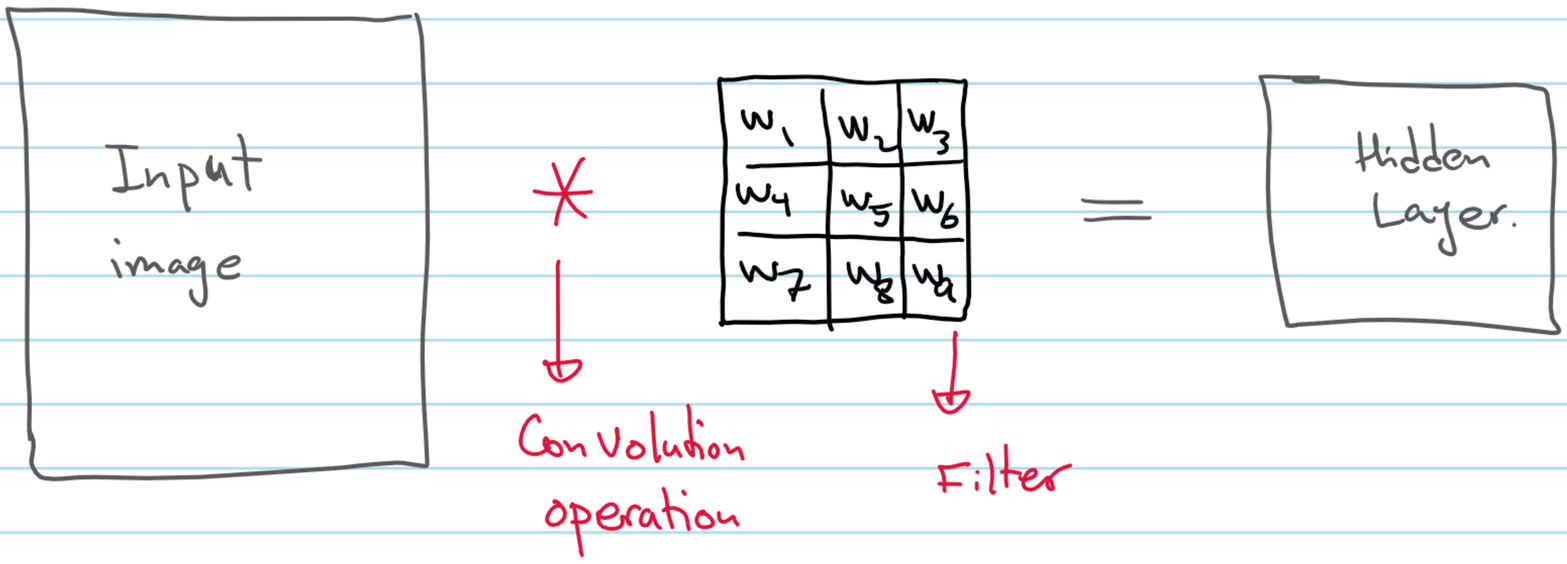
$$H_2 = W_1 P_{12} + W_2 P_{13} + W_3 P_{14} + W_4 P_{22} + W_5 P_{23} + W_6 P_{24} + W_7 P_{32} + W_8 P_{33} + W_9 P_{34}$$

$$H_3 = W_1 P_{21} + W_2 P_{22} + W_3 P_{23} + W_4 P_{31} + W_5 P_{32} + W_6 P_{33} + W_7 P_{41} + W_8 P_{42} + W_9 P_{43}$$

H_4 ?

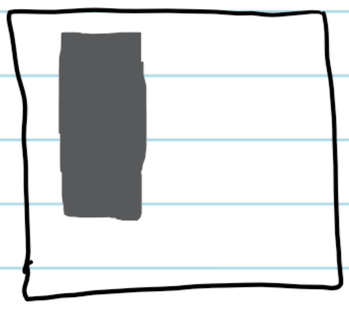
You can see that with 9 weight ($W_1 \dots W_9$) we can transform the input image into the first hidden layer. And this number is indep. of size of the input or hidden layer.

This is similar to the convolution operation



Examples of convolution & filters:

Edge detection



The first step to detection of an object is to detect its edges.

1	1	1
0	0	0
-1	-1	-1

Horizontal

1	0	-1
1	0	-1
1	0	-1

Vertical

→ See the jupyter nb.

- Filter size
- Padding
- Stride

→ Size of the Hidden layer

- Pooling → Coarse-graining
- FC