CNNs

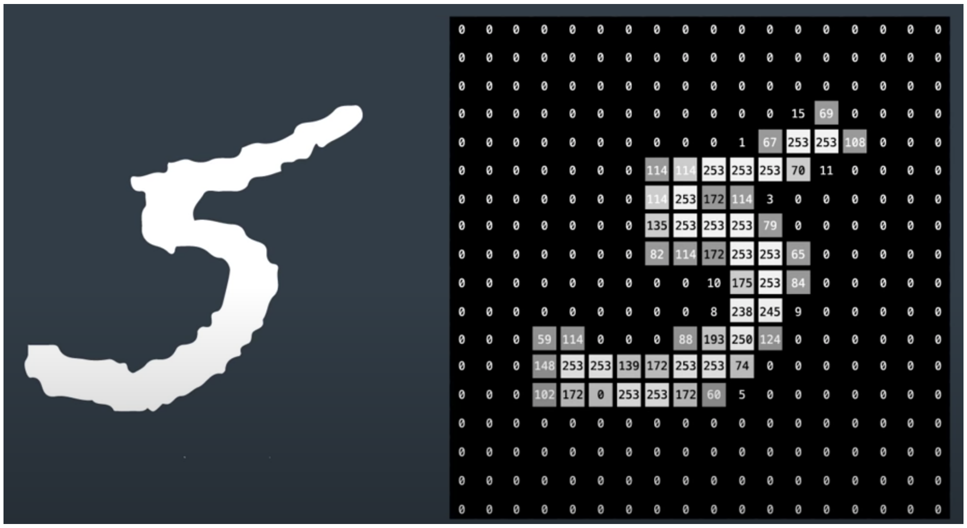
Agenda

1. Motivation
2. How computers interpret images
3. Local Connectivity
4. Convolution layer
5. Max pooling
6. Overall picture
7. Keras Example

1. Motivation

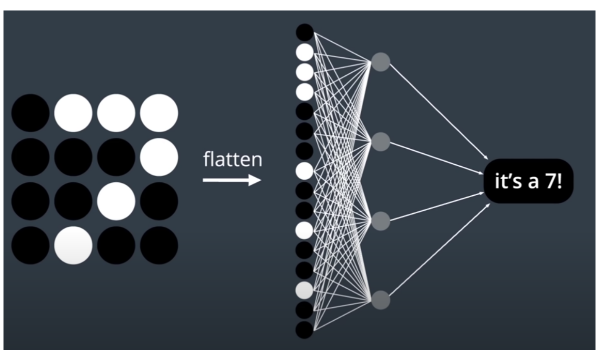
- **Self driving cars** ; Waymo https://www.youtube.com/watch?v=B8R148hFxPw
- **AlphaGo**: There are an astonishing 10 to the power of 170 possible board configurations. This makes the game of Go a googol (1.0 × 10100) times more complex than chess. https://deepmind.com/research/case-studies/alphago-the-story-so-far

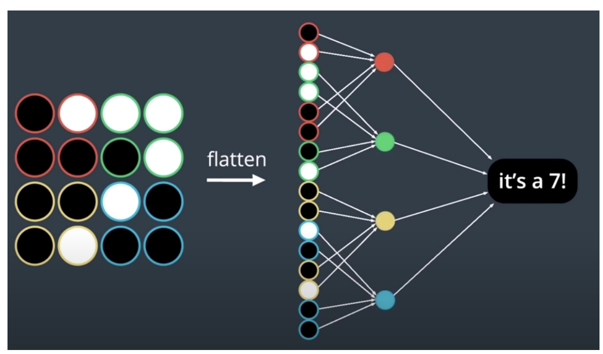
2. How Computers interpret images



For color images, it is the same concept, however we use 3 dimensions for the RGB channels (Red, Green, Blue)

3. Local Connectivity





Fully Connected layer

- In a fully connected layer, every node in a hidden layer is connected to all pixels, and is responsible for finding patterns in all pixels

Locally Connected layer

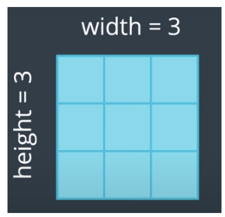
- In a locally connected layer, nodes are only assigned to a group of pixels and thus every node is responsible for finding patterns in a group of pixels that are next to each other
- Everyone of those nodes that are connected to a specific group of pixels report to the output layer to combine the patterns found seperately in each region
- Use far less weights than a fully connected layer
- Less prone to overfitting

4. Convolution Layer

Now lets formalize the conceptual idea that we introduced in the previous section

4.1 Convolution Window

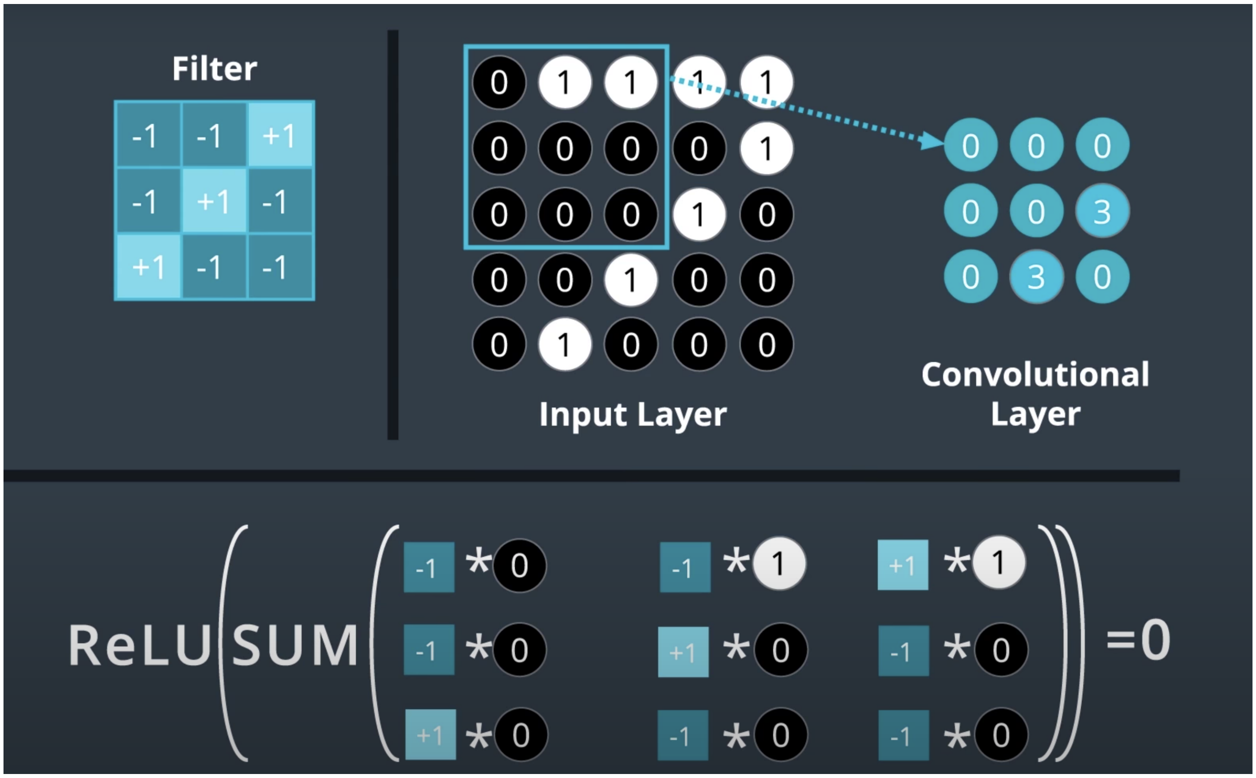
We first define a convolution window of a specific width and height,we then slide this window over the image horizontally and vertically as follows



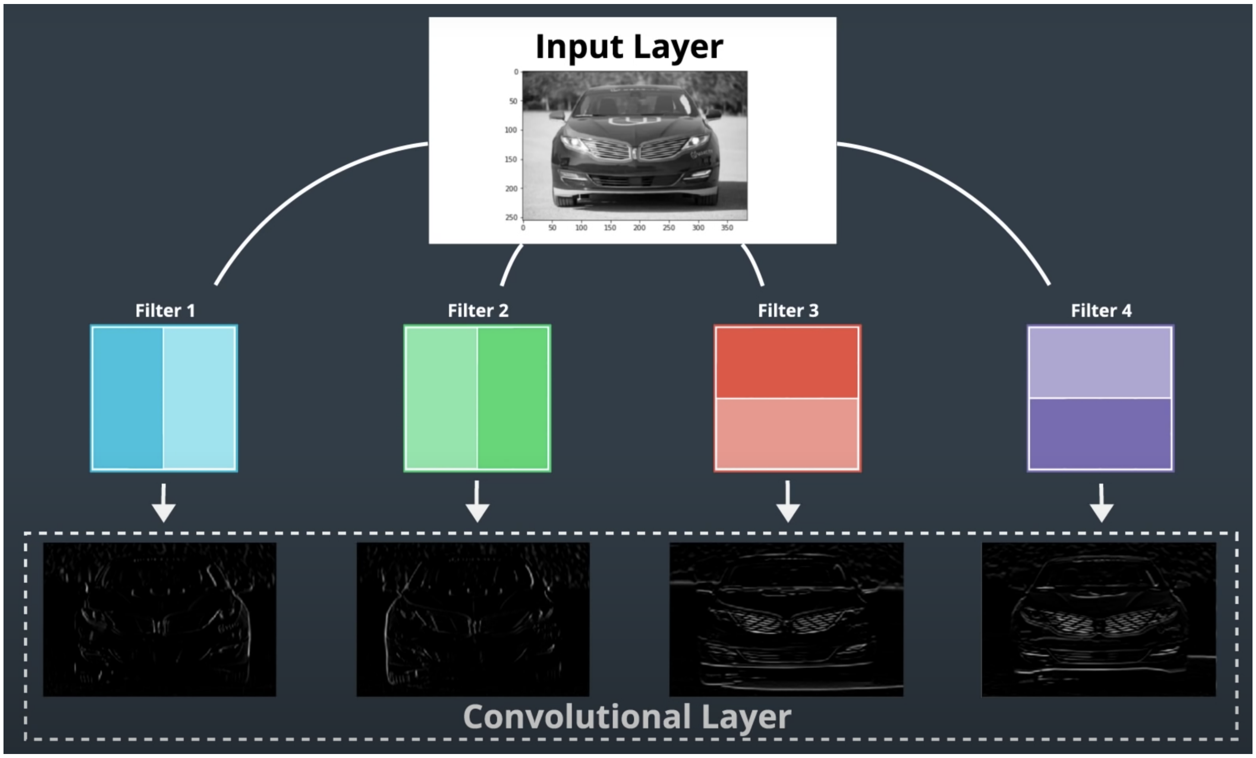
4.2 Filter

In the convolution window, we will have numbers that simulates exciting the neurons for certain patterns. In the below example, we are looking for diagonal lines in the image.

4.3 Slide & Compute



when we do the same operation but with different filters, in essense, what is happening is that we are picking up different patterns in the image



and then the more convolution layers you have, the more complicated patterns are detected by the network.

For example;

- the first layer might be picking up vertical and horizontal lines
- and then the second layer might be picking up circles, boxes, triangles
- and then the third layer might be picking up fingers, eyes, mouth, etc.
- etc.

So what does the network learn here ?

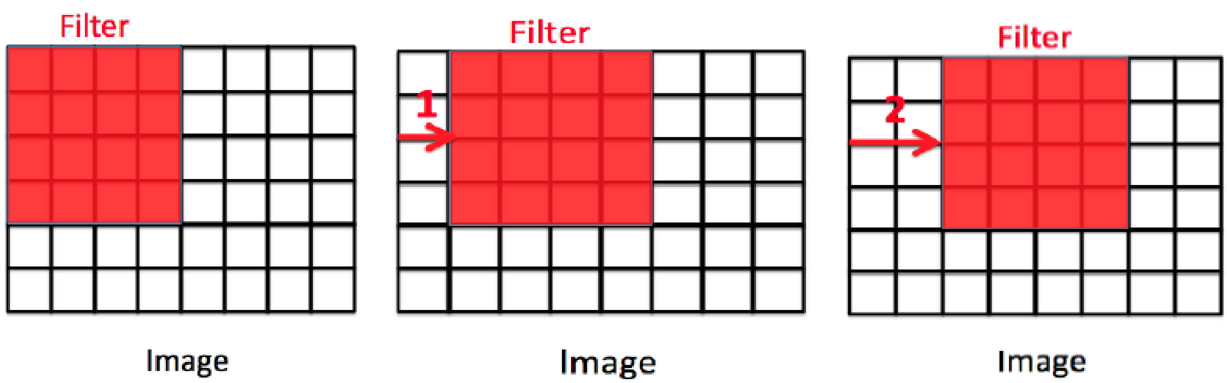
The values inside the filters are weights that are going to be learned by the network. We are not directing the network to learn simple concepts first and then combine them into more complicated concepts.

That happens because the network figured out in the solution space that the best way to find a minimum error is using this strategy.

Essentially the network is find the best strategy for what to be learned in every layer of the deep neural network.

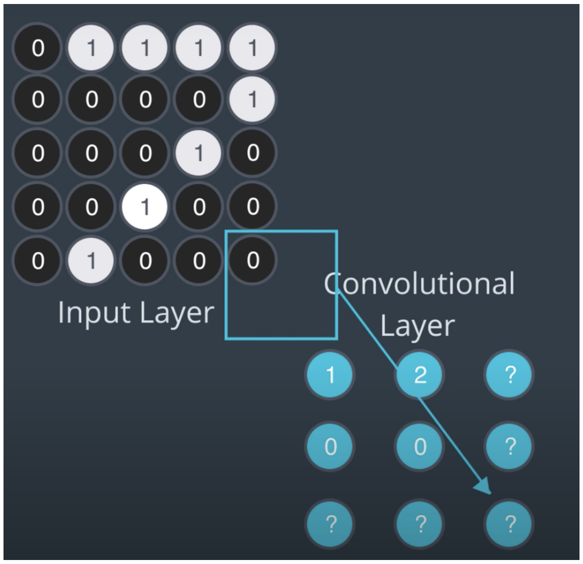
4.4 Stride

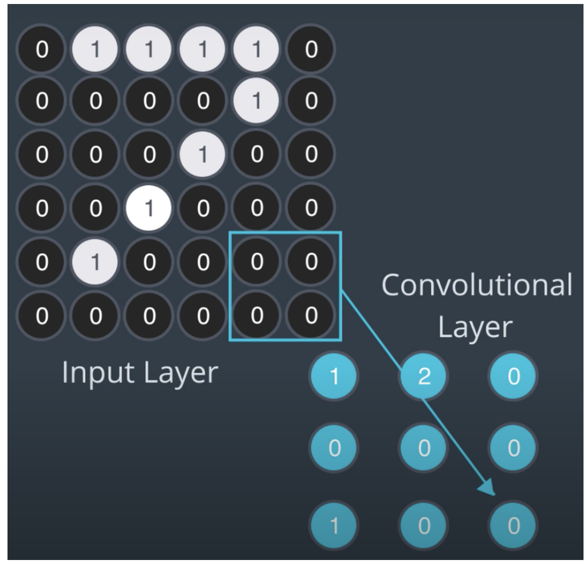
The stride determines the movement of the filter. For example does it moves by +1 step or +2 steps, +3 steps, etc. The bigger the stride is smaller the result is.



4.5 Padding

Padding determines how to handle situations where the convolution window is going outside of the boundaries of the image.





Here we ignore the parts outside the image and would be losing some information

The other method is to introduce zero values around the borders of the image to include in the convolution window as it is sliding

Padding

- `"VALID"` = without padding:

```
inputs:         1  2  3  4  5  6  7  8  9  10 11 (12 13)
               |________________|                dropped
                              |_________________|
```

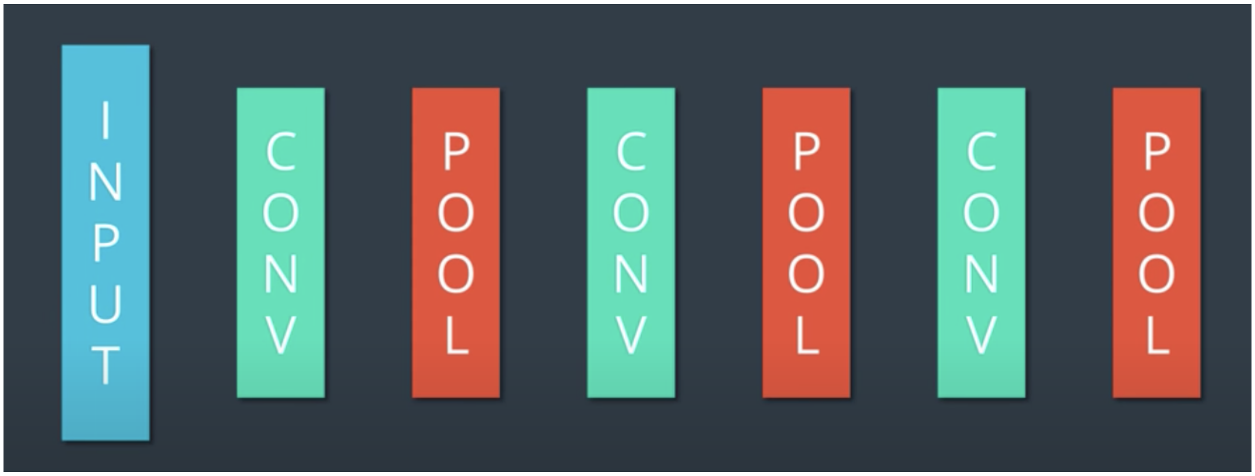
- `"SAME"` = with zero padding:

```
            pad|                                      |pad
inputs:      0 |1  2  3  4  5  6  7  8  9  10 11 12 13|0  0
            |________________|
                           |_________________|
                                          |________________|
```

5. **Pooling Layers**

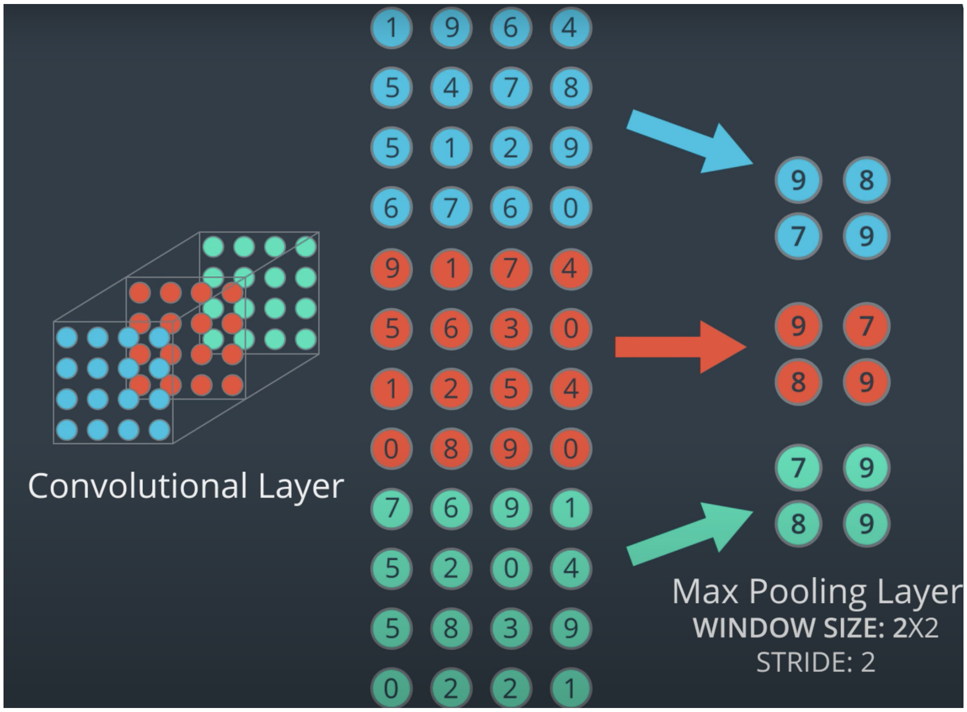
As we add more convolution layers to our network, the complexity of the network will grow and thus we would be risking overfitting and thus we need a measure to reduce the dimension of the network. That is why we use pooling layers.

Where do you apply pooling in a network architecture ?



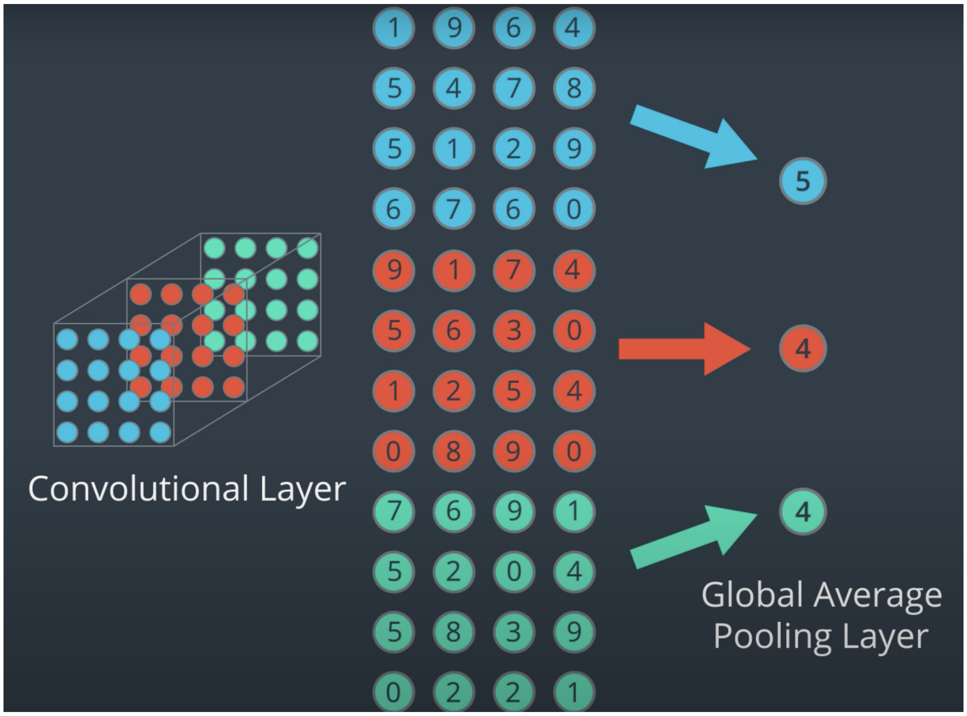
5.1 **Max Pooling**

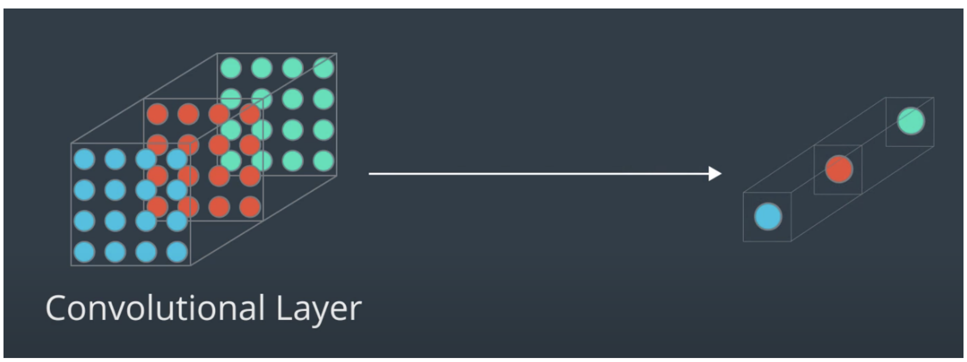
Max pooling is an dimension reduction operation that is done by sliding a window of a specific size and a certain stride over the the convolution output and then taking the maximum of the values inside that window.



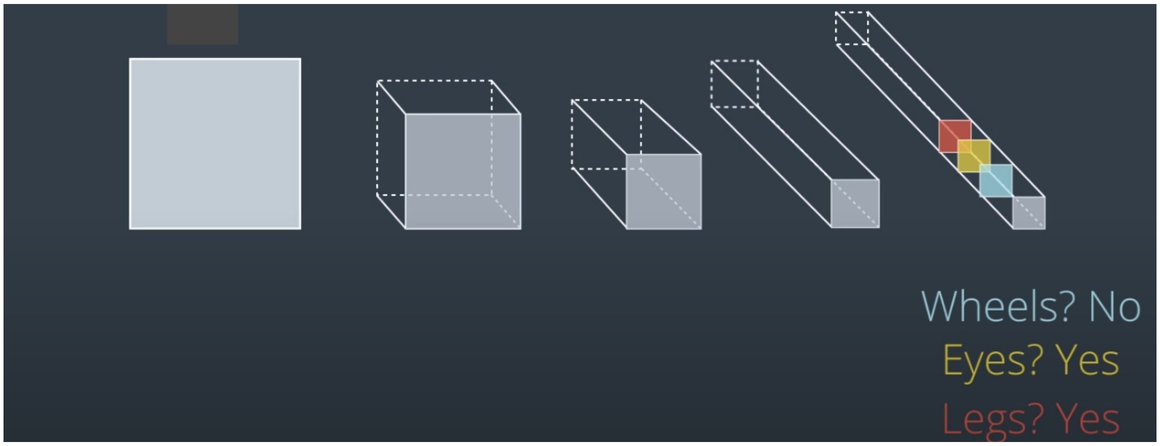
5.2 **Global Pooling**

Global pooling is a more extreme measure of reducing dimensions and that is done by taking the global average of the all values inside convolution layers





6. Overall Picture



7. **Keras Example**

https://keras.io/examples/vision/mnist_convnet/

Check the code