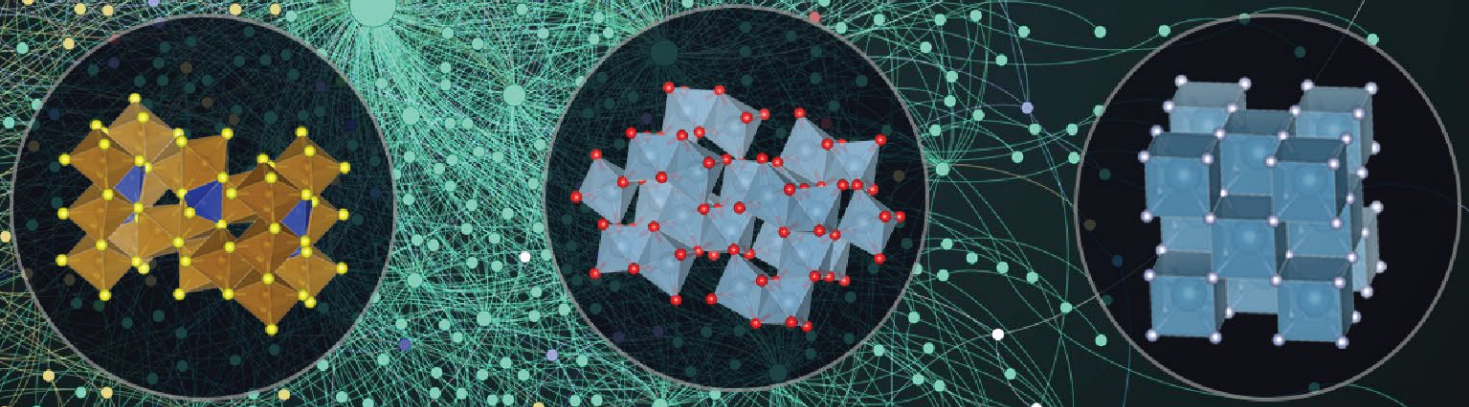
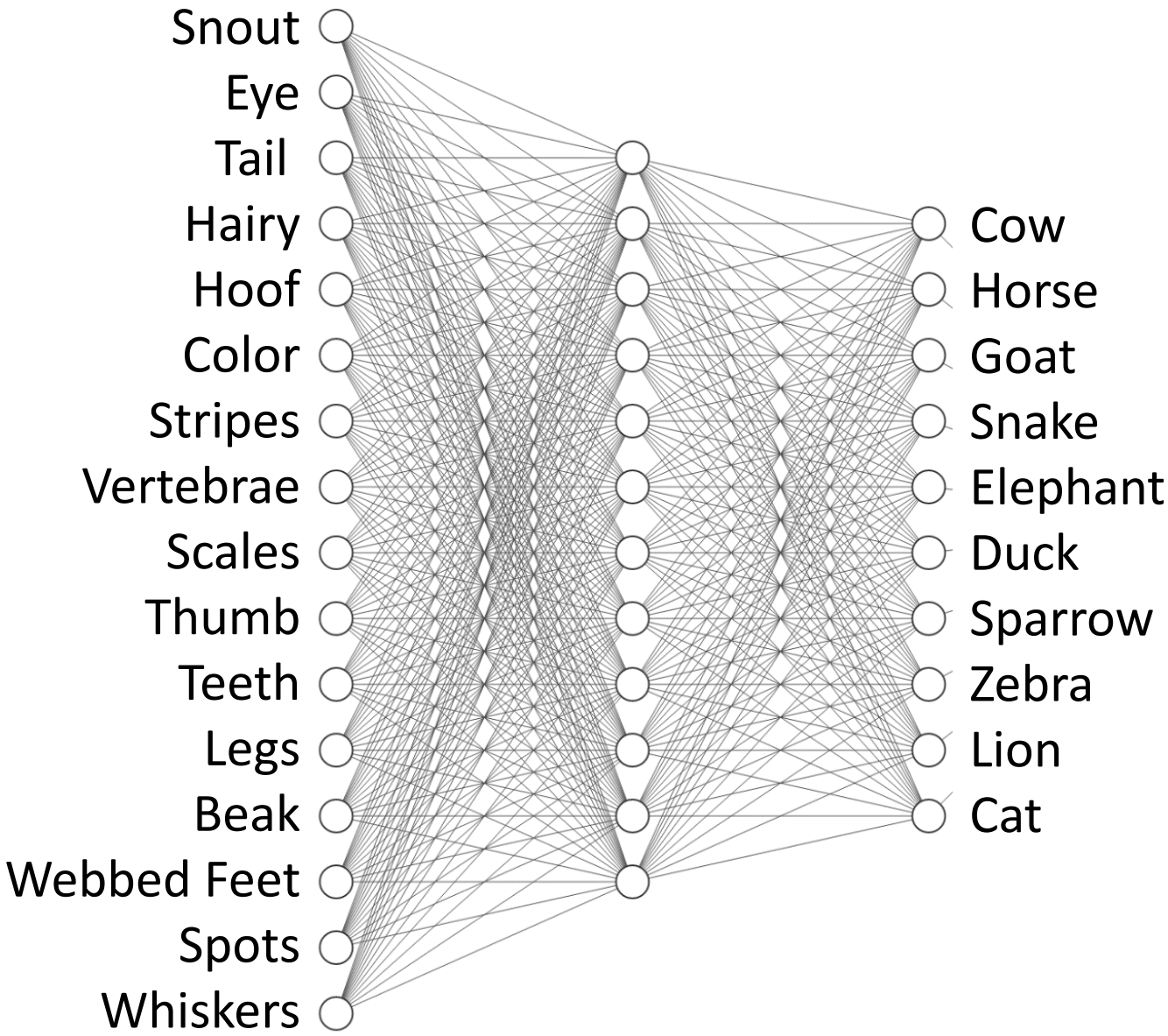
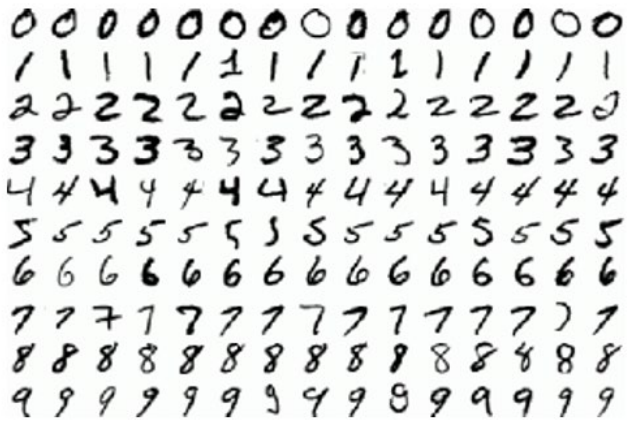
convolutional neural networks



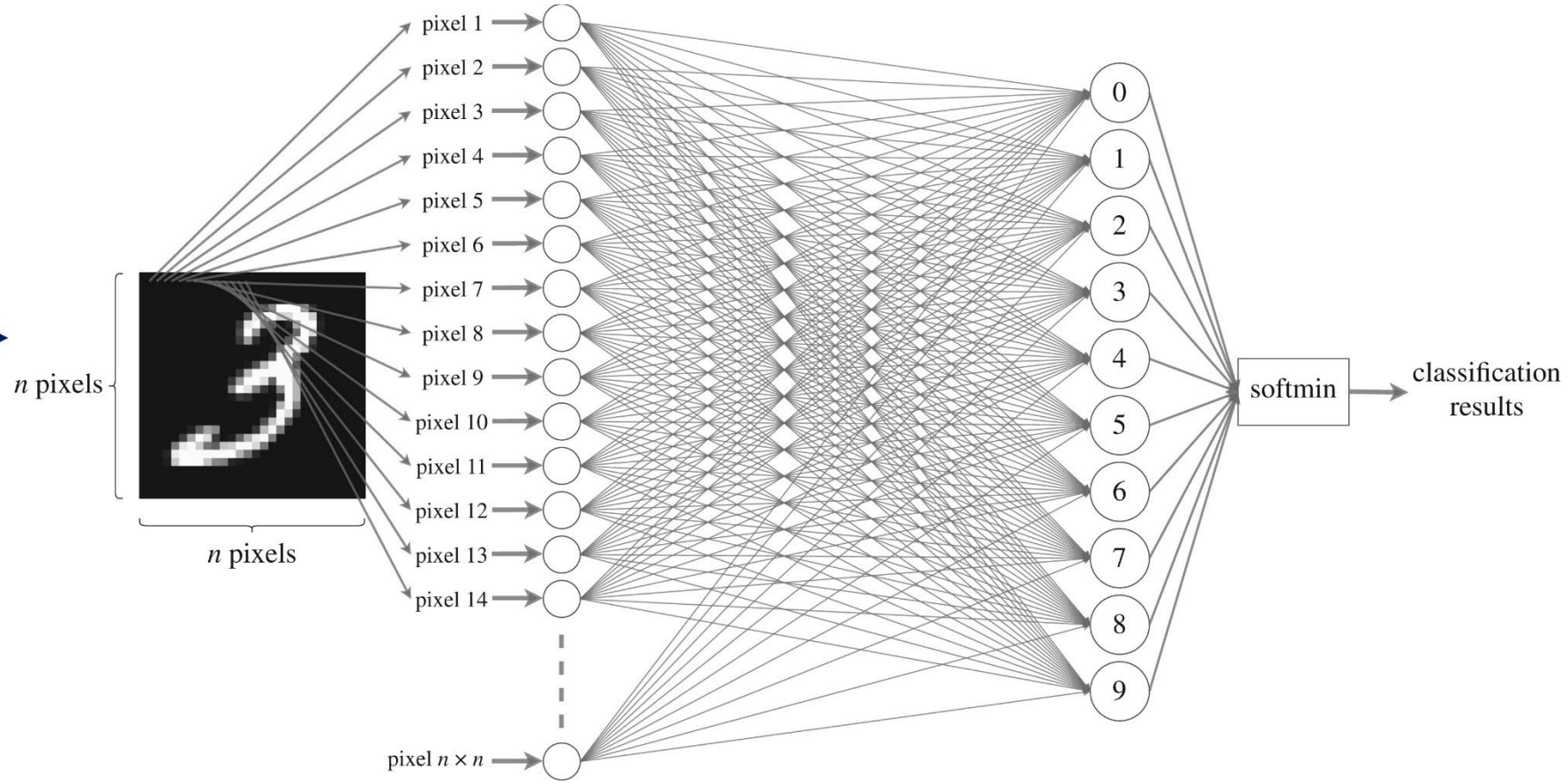
Vanilla neural networks do well to find patterns in the features provided



We can go beyond “vanilla” neural networks to do more advanced tasks

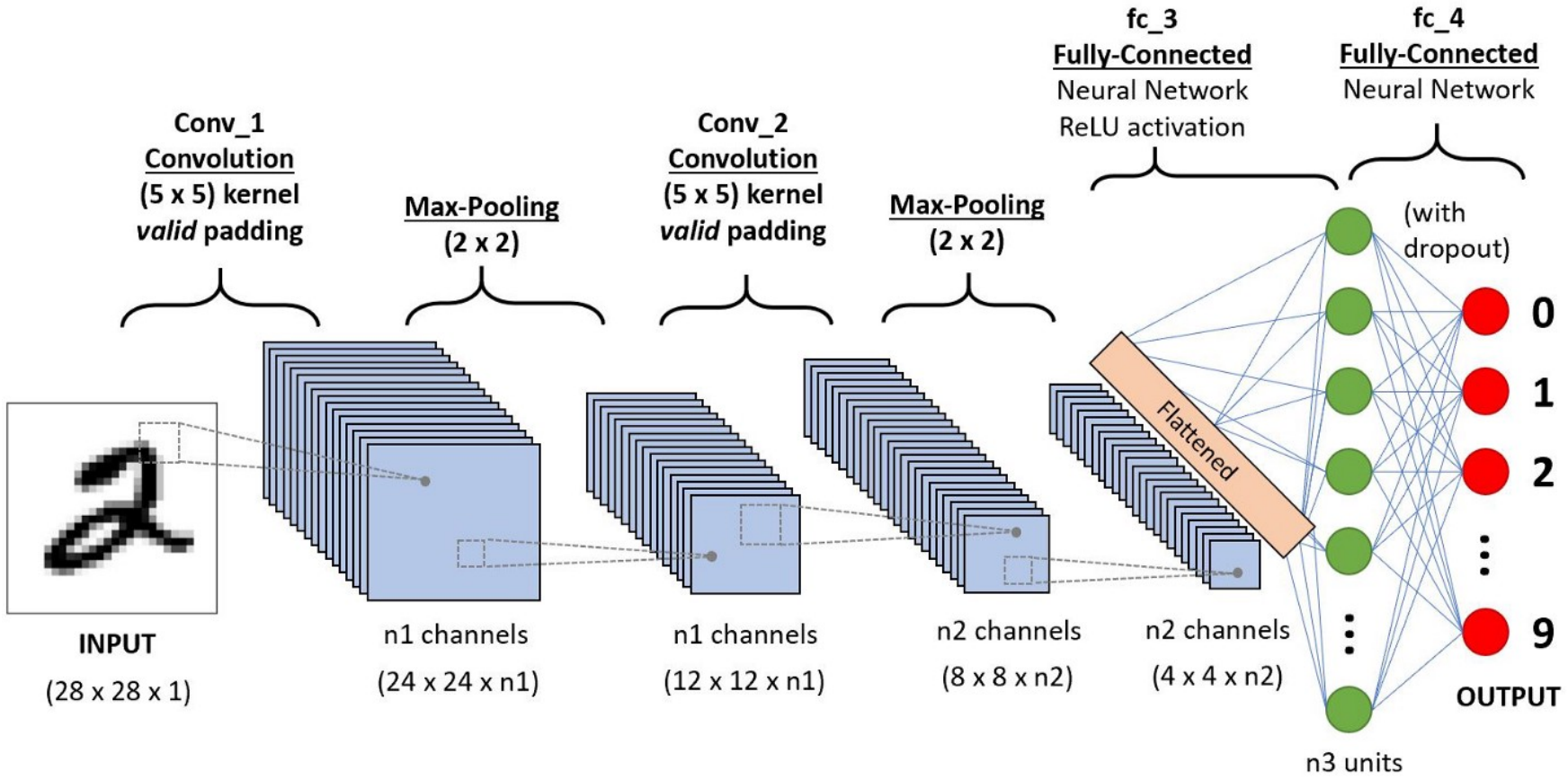


Data & Labels



https://royalsocietypublishing.org/doi/10.1098/rsta.2019.0163

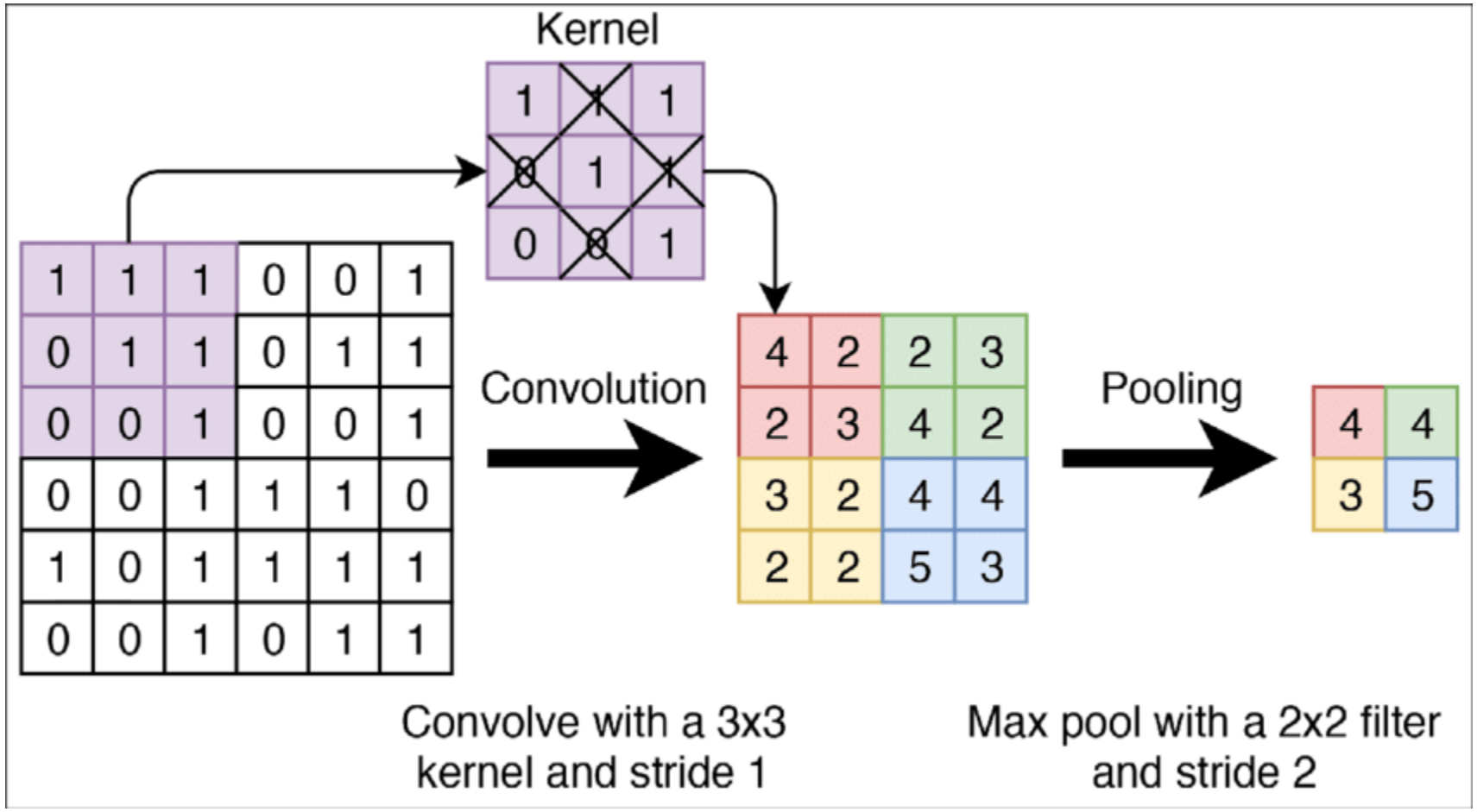
Convolutional neural networks do really well at learning from image data



Key assumption: we shouldn't ignore how pixels are spatially located!

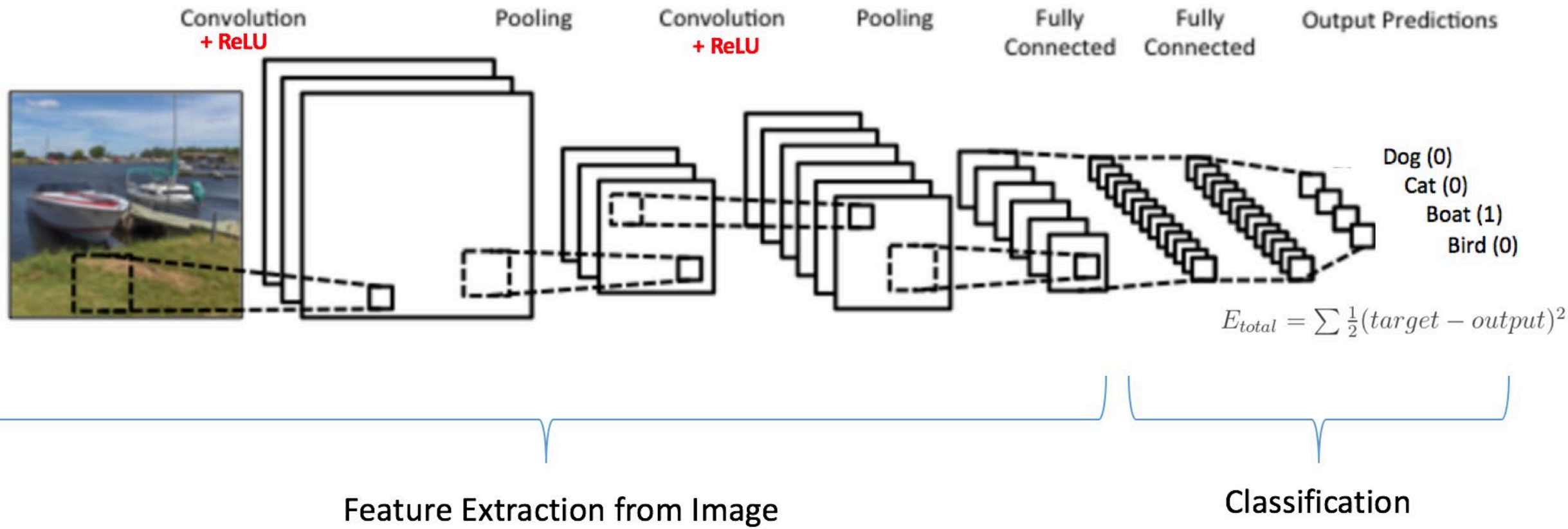
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

CNNs are made up of convolutions and pooling layers



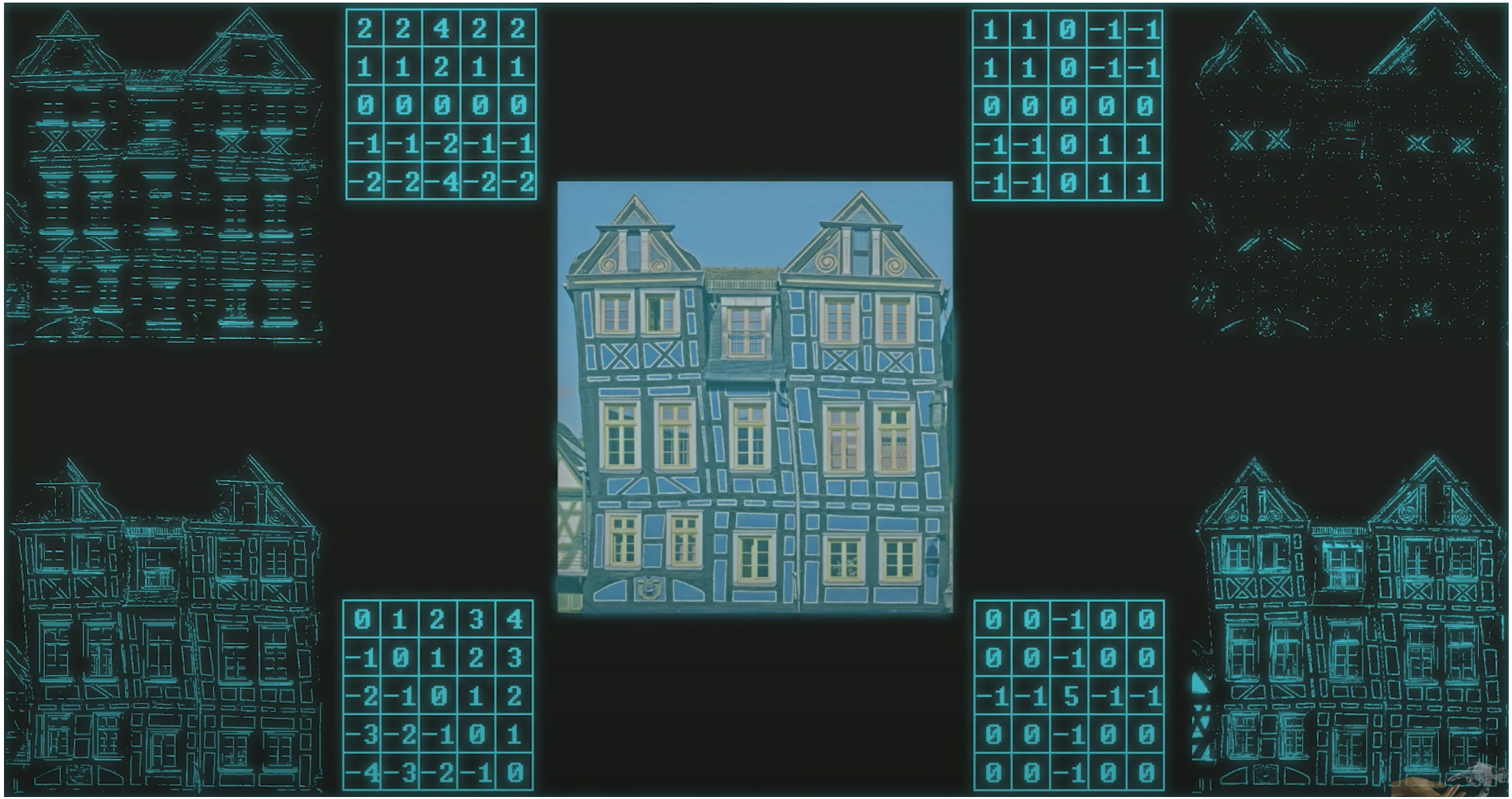
Verschoof-van der Vaart and Lambers, *J. Comp. App. in Arch.,* (2019)

Convolutions and pooling extract features automatically!



https://www.kdnuggets.com/2016/11/intuitive-explanation-convolutional-neural-networks.html/3

Low level filters do a good job of finding edges, corners, shapes, colors, and gradients

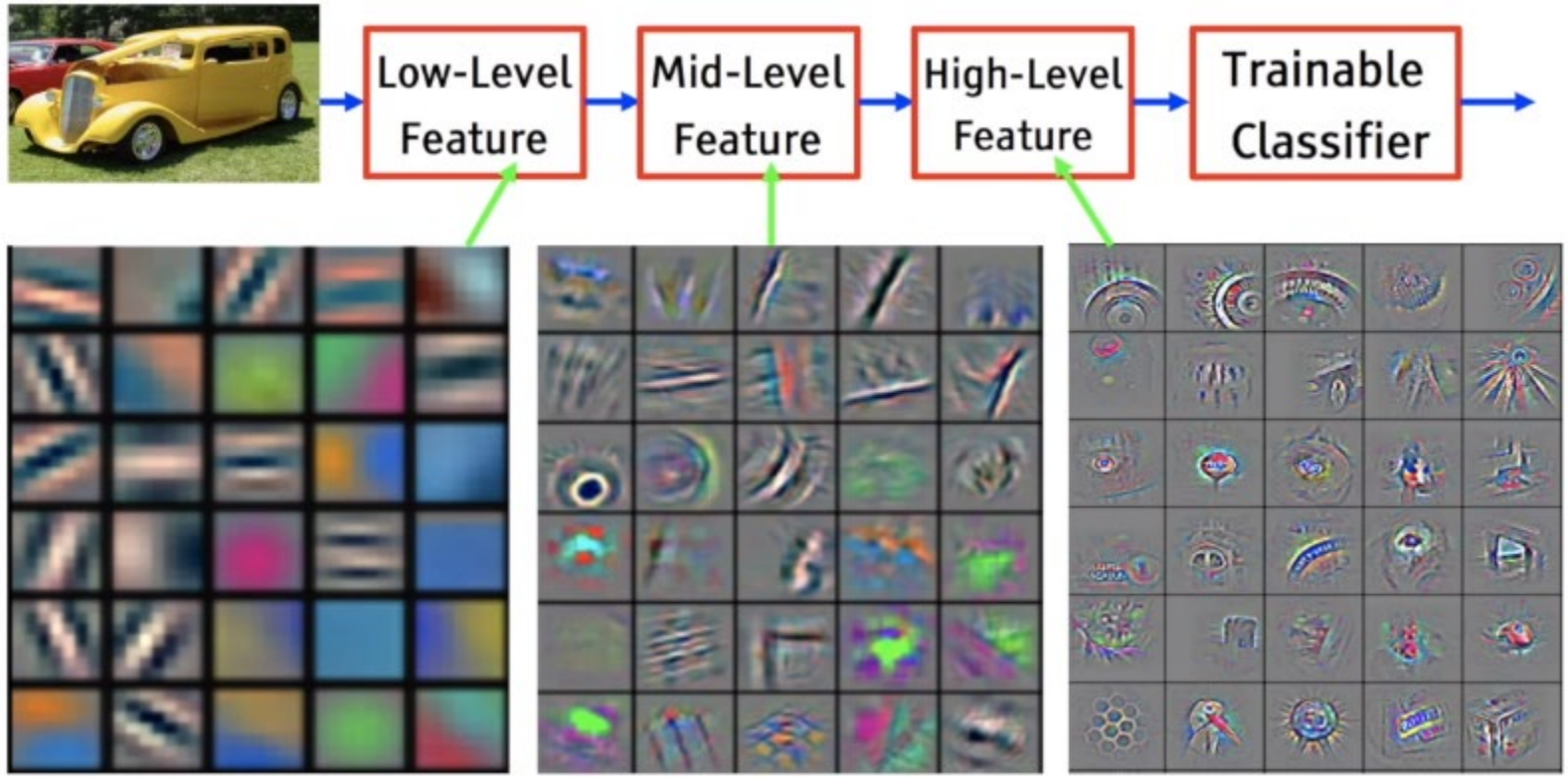


Futurology, “Convolutional Neural Networks Explained (CNN Visualized)” YouTube

Low level filters do a good job of finding edges, corners, shapes, colors, and gradients

Futurology, “Convolutional Neural Networks Explained (CNN Visualized)” YouTube

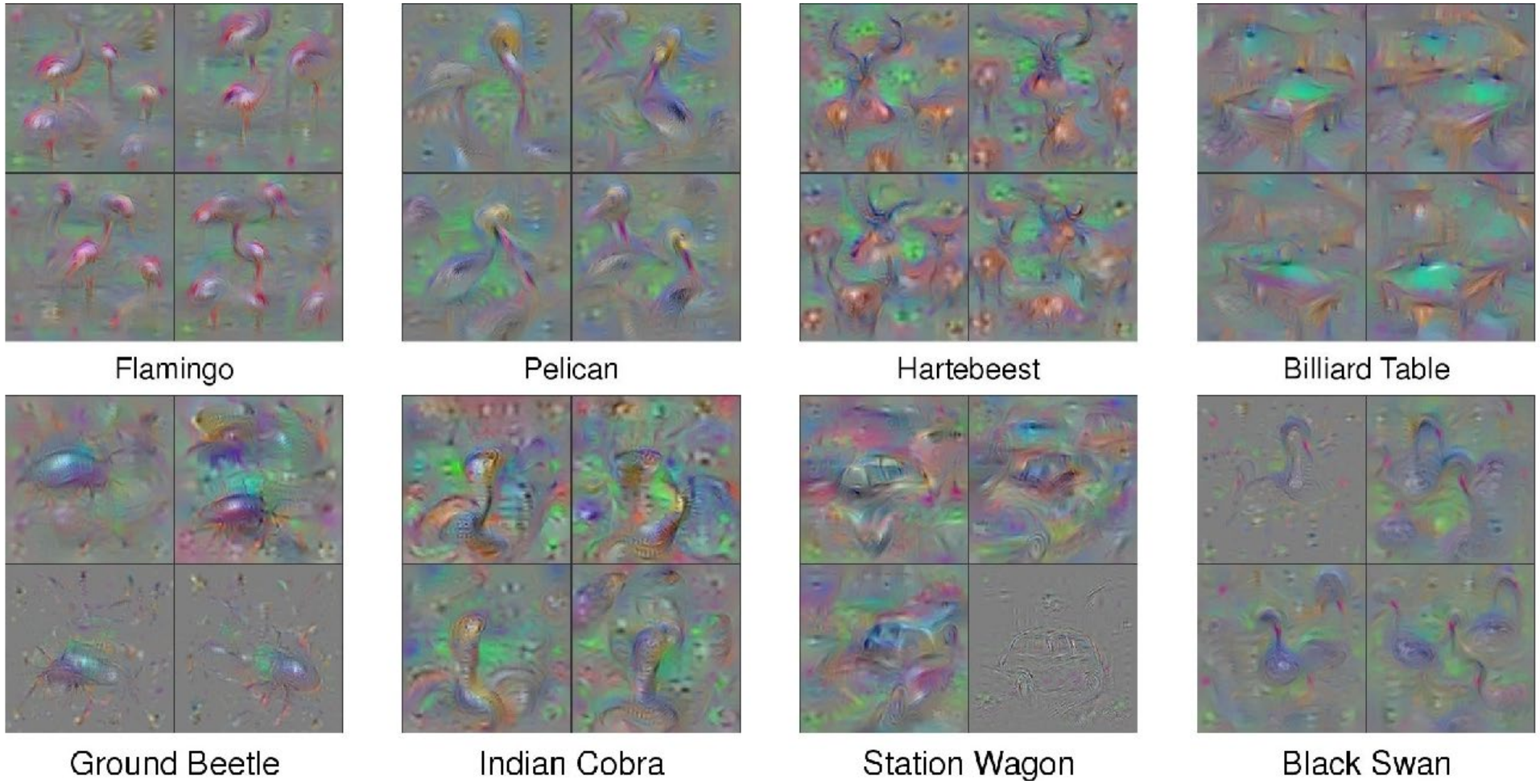
Filters are learned at different scales and change in complexity



Simple features → Complex shapes similar to real world

https://www.kdnuggets.com/2016/11/intuitive-explanation-convolutional-neural-networks.html/3

Filters help us learn the feature "essence" of an object



Preferred inputs for different class categories

Yosinski et al, *Computer Science*, (2015)

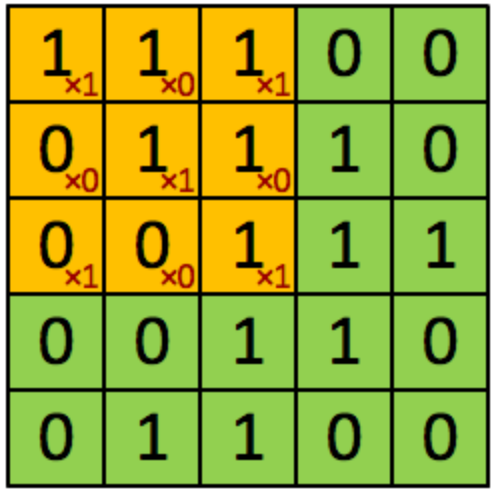
CNNs capture spatial dependence through the use of filters

We need to reduce image to a computationally tractable size without losing information

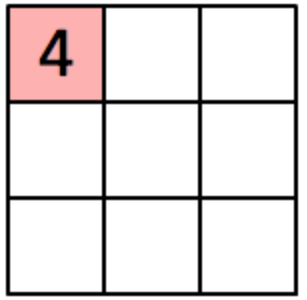
Original input 5X5

Special 3X3 filter/kernel used for convolution

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$



Image

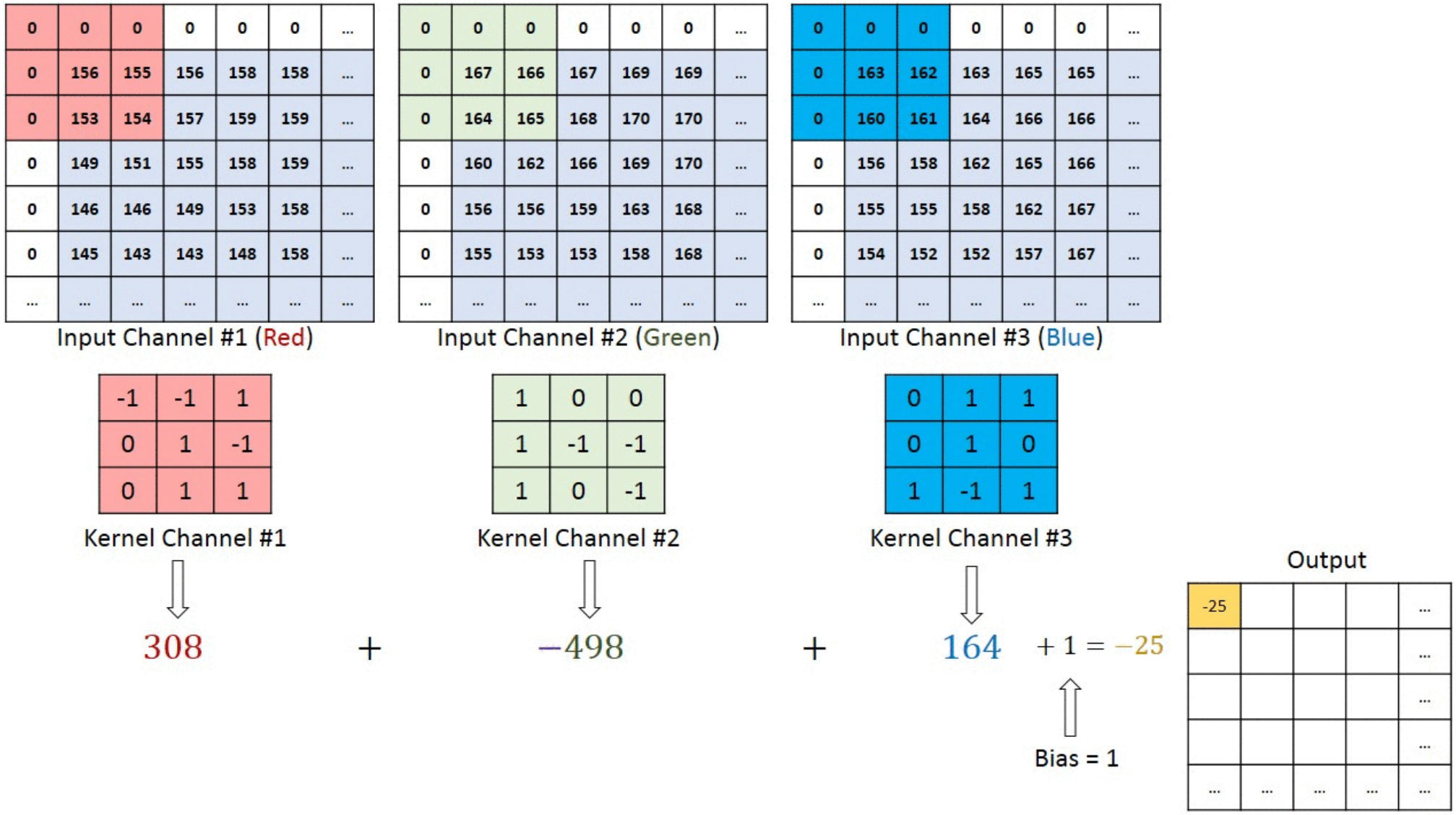


Convolved Feature

Convolved feature is 3X3 because we had 9 kernel shifts

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

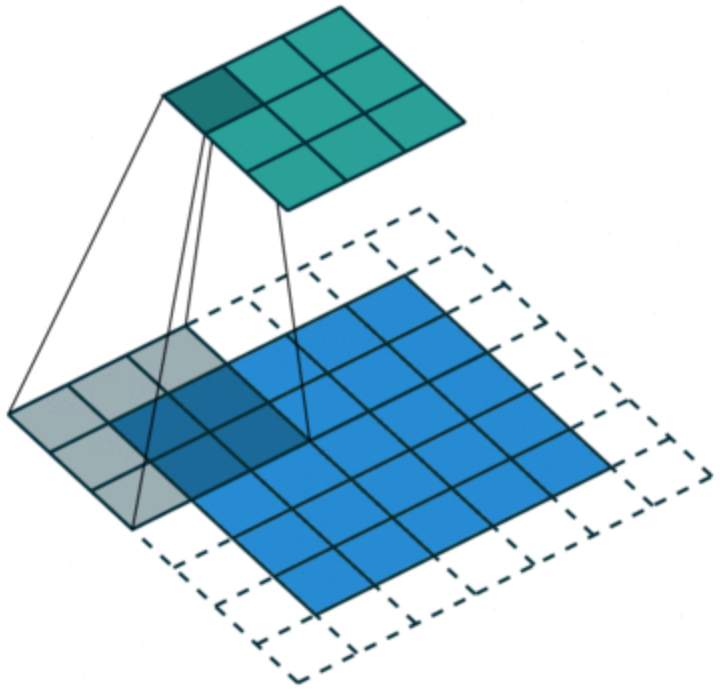
If our input has depth > 1, we use filter with same depth and sum to single channel output



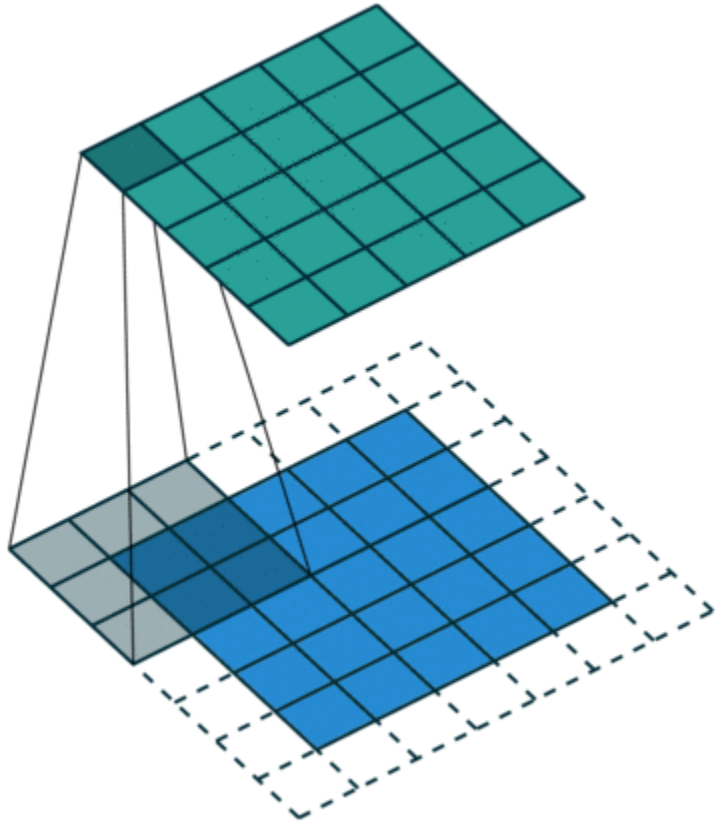
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Convolutions can take place over different “strides” with different padding

“Valid padding” reduces dimensionality



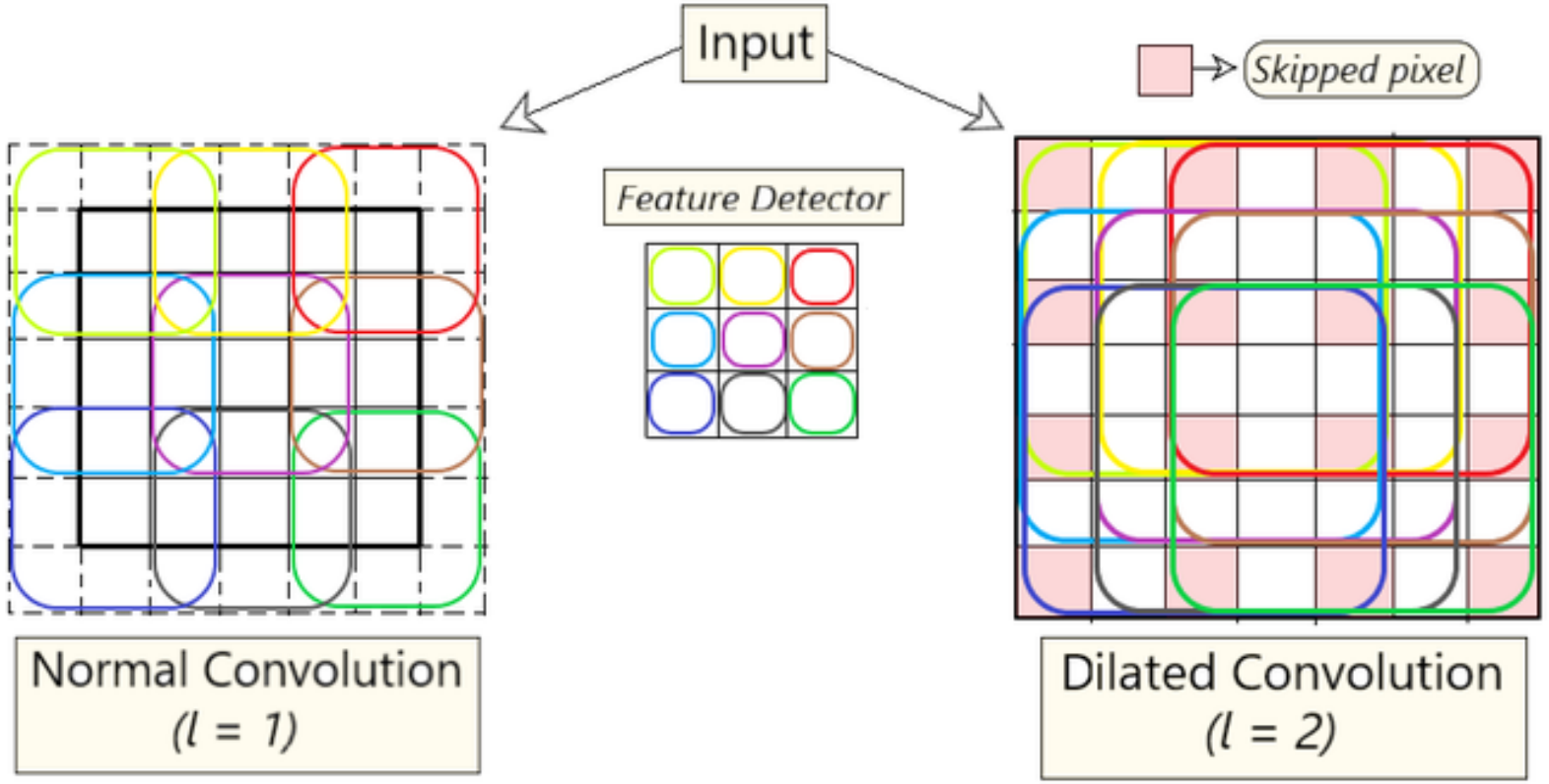
“Same padding” keeps it same or increases



Animations of convolutions https://github.com/vdumoulin/conv_arithmetic

“Dilated” convolution allows us to sample kernel over a larger area

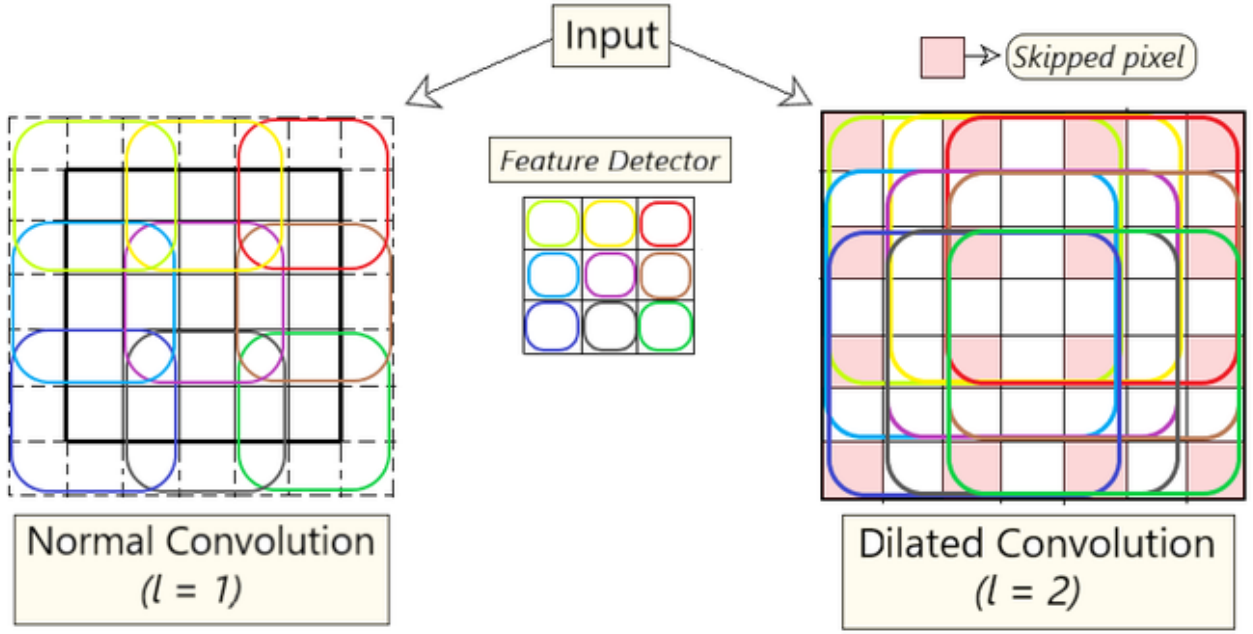
An additional parameter l (dilation factor) tells how much the input is expanded. In other words, based on the value of this parameter, (l-1) pixels are skipped in the kernel.



https://www.geeksforgeeks.org/dilated-convolution/

Dilated convolutions offer some advantages

Larger receptive field (no loss of coverage)
Computationally efficient (larger coverage at same cost)
Less memory usage since it can skip the pooling step
No loss of resolution (dilation instead of pooling)
Order of data maintained

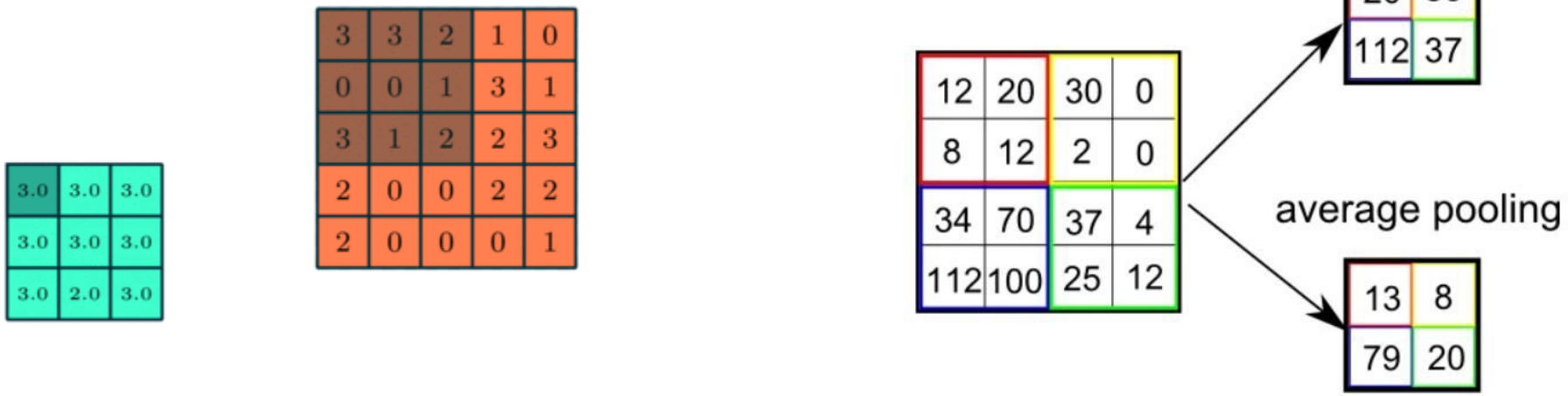


https://www.geeksforgeeks.org/dilated-convolution/

Pooling also reduces dimensionality while also extracting dominant features

Max pooling or average pooling possible

Both use a kernel, but instead of summing, we return max value or average

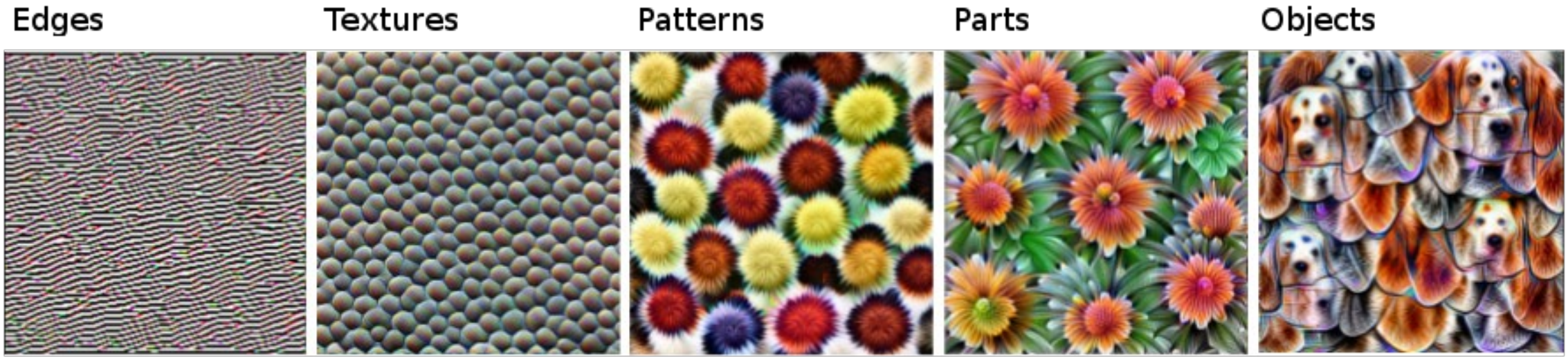


Max pooling also performs noise suppression

Pooling prevents overfitting and speeds up calculations

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

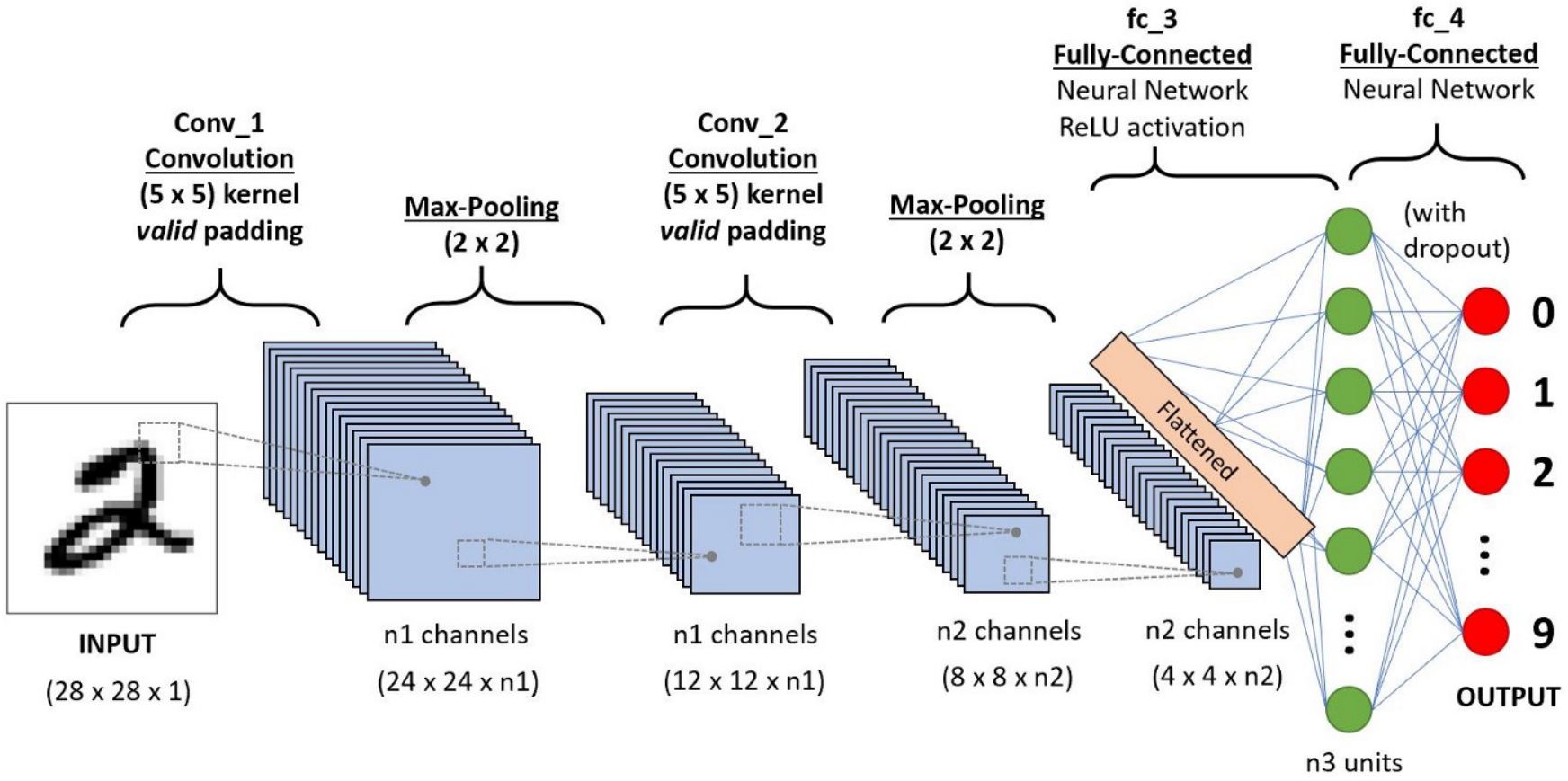
Convolution and pooling layers are repeated to build up abstraction in network



More abstract kernels are made via combination of simple filters

https://christophm.github.io/interpretable-ml-book/cnn-features.html

Convolutions/Pooling learn features, but we need to correlate these with target



https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

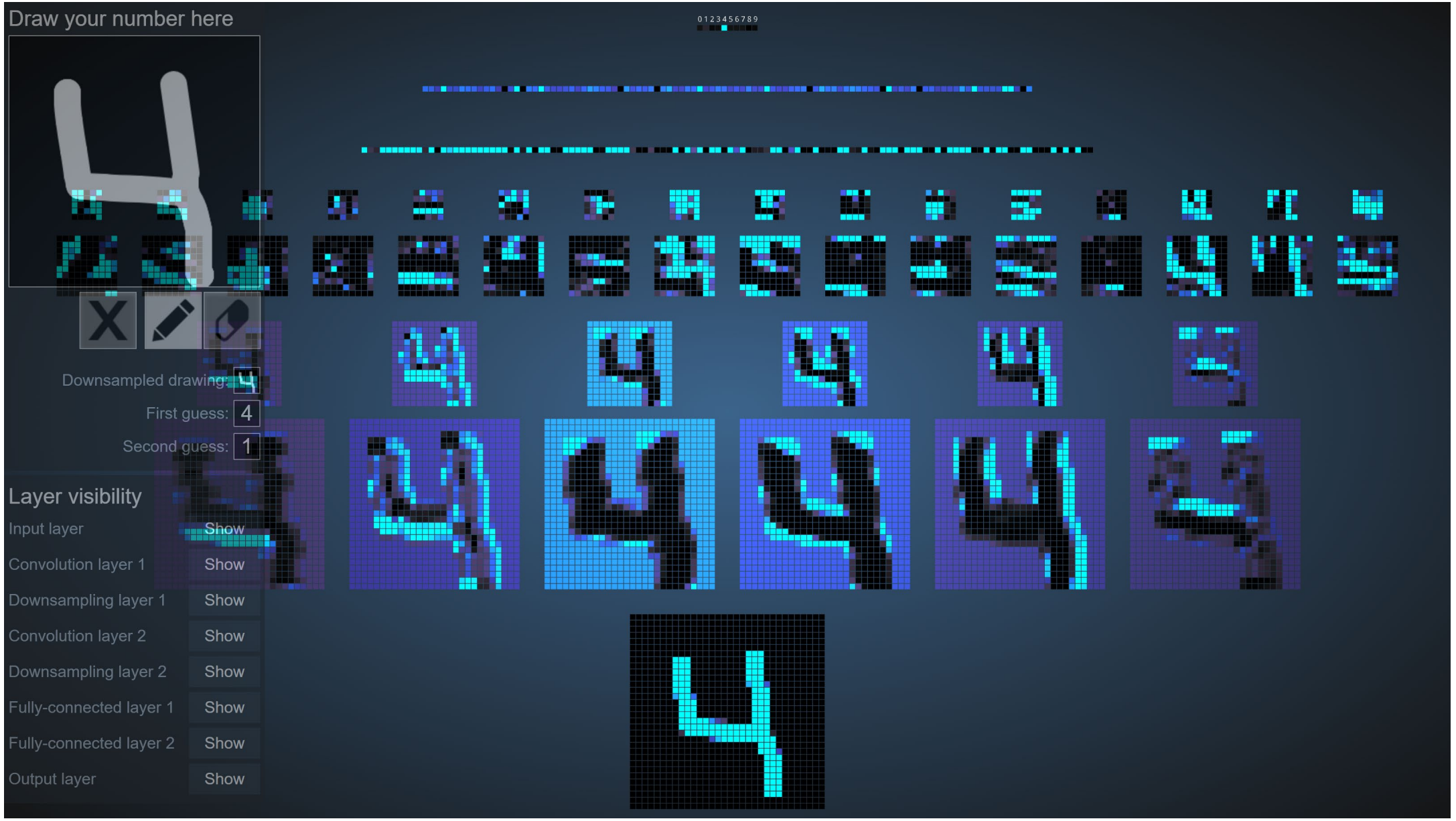
CNNs can be broken down into steps

Step 1: We initialize all filters and parameters / weights with random values
Step 2: The network takes a training image as input, goes through the forward propagation step (convolution, ReLU and pooling operations along with forward propagation in the Fully Connected layer) and finds the output probabilities for each class.
Step 3: Calculate the total error at the output layer
Step 4: Use Backpropagation to calculate the gradients of the error with respect to all weights in the network and use gradient descent to update all filter values / weights and parameter values to minimize the output error.*
Step 5: Repeat steps 2-4 with all images in the training set.

*Parameters like number of filters, filter sizes, architecture of the network etc. have all been fixed before Step 1 and do not change during training process – only the values of the filter matrix and connection weights get updated.

https://www.kdnuggets.com/2016/11/intuitive-explanation-convolutional-neural-networks.html/3

We can visualize CNNs in action!



https://adamharley.com/nn_vis/

recurrent neural networks