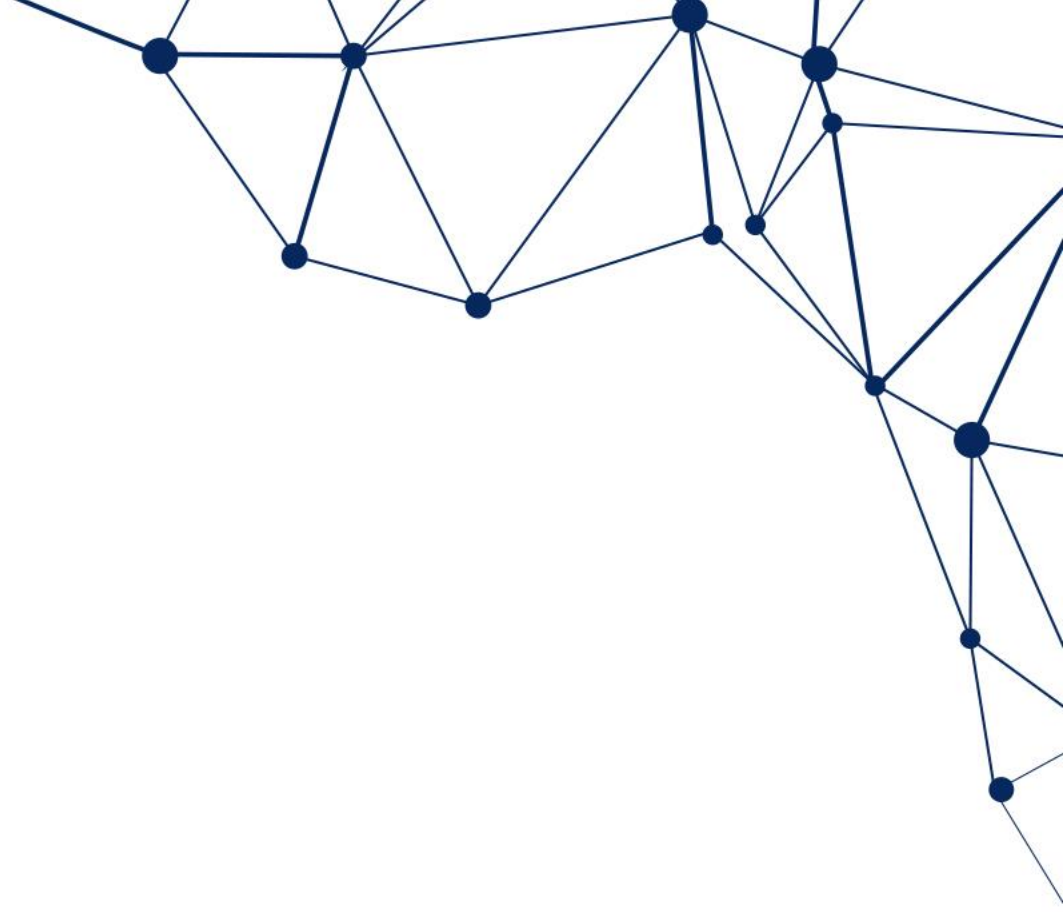
Course notes:

Convolutional Neural Networks

Image kernels



- The linking bridge between feedforward neural networks and the Convolutional Neural Networks
- They are the basic building blocks of filters such as Instagram filters
- Their main purpose is to transform images

Example image transformations

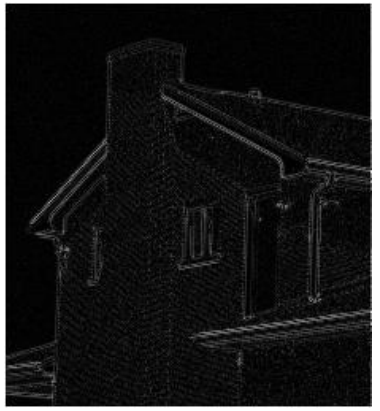
Achieved using kernels

Original

Blur

Sharpen

Edge Detection



Kernels as matrices

Kernels work by combining the values of each pixel with its neighbors in order to obtain a new, transformed value. Thus, kernels can be conveniently expressed in matrix form, visually expressing the combination of values.

Blur

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

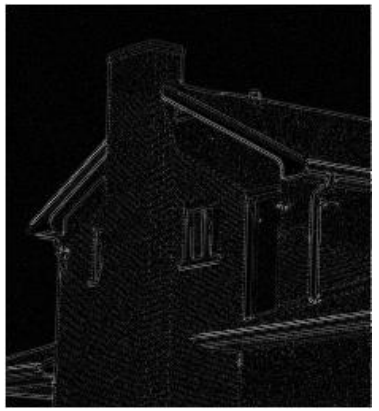
$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Sharpen

$$\begin{bmatrix}0 & -1 & 0\\ -1 & 5 & -1\\ 0 & -1 & 0\end{bmatrix}$$

Edge Detection

$$\begin{bmatrix}0 & -1 & 0\\ -1 & 4 & -1\\ 0 & -1 & 0\end{bmatrix}$$



365 DataScience

Convolution

The kernels are applied to the image through the mathematical operation of **convolution.**

(Okay, actually, this is cross-correlation, but convolution is closely related, and the two are conflated often enough)

Convolution equation

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

the result, the transformed image

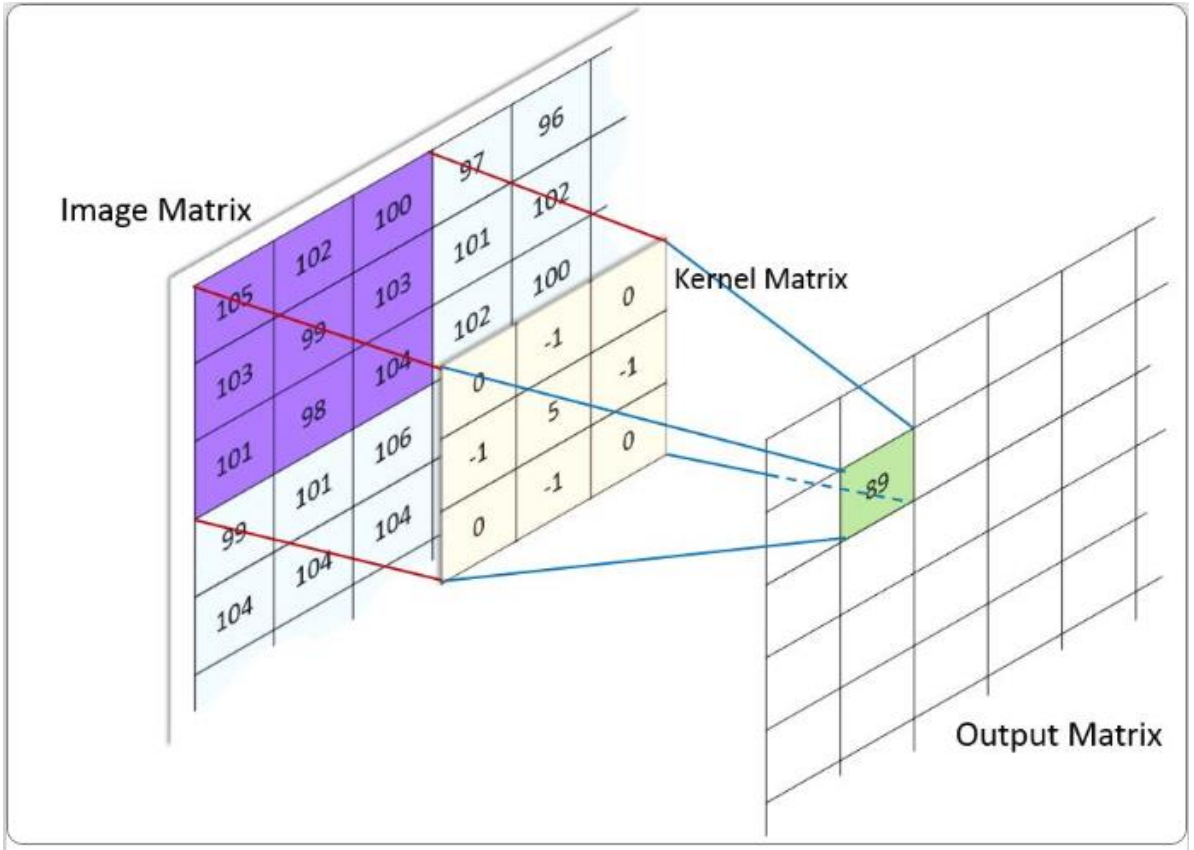
the original image

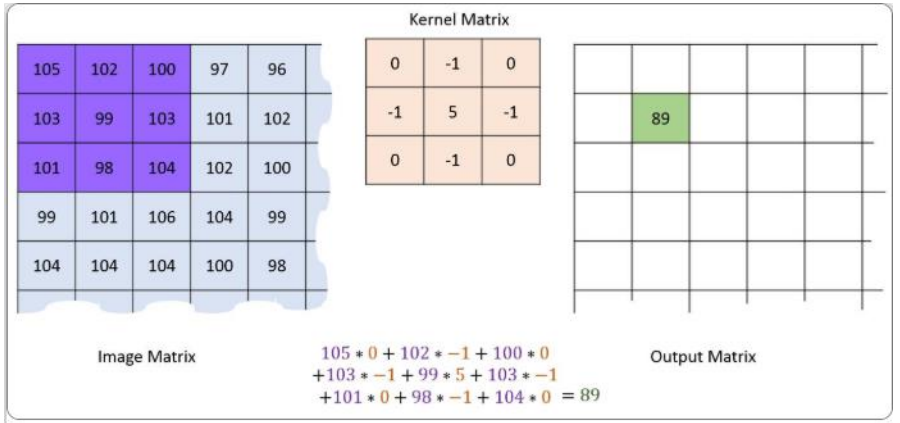
the kernel matrix

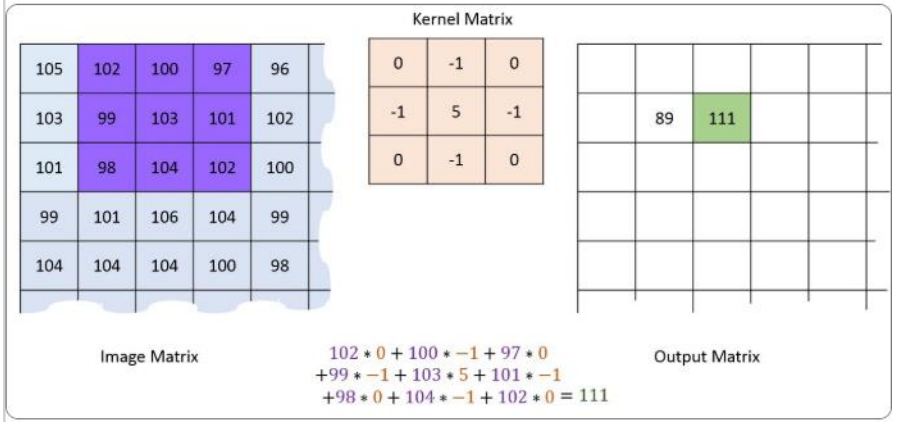
In practice, **convolution** is the operation of calculating the new value of each pixel in the image, by “sliding” the kernel over it.

Convolution

Visuals







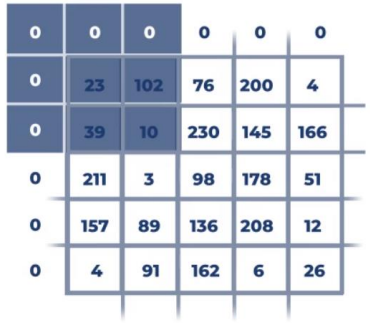
365 DataScience

Edge handling

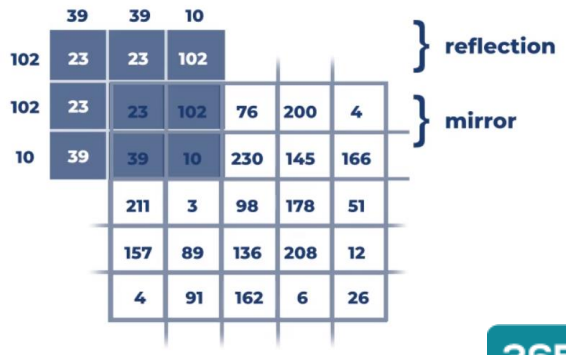
There is ambiguity what to do when the kernel “sticks out” of the image, near the edges.

In this case, there are a couple of solutions:

- Zero padding – expand the image outwards with pixel values set to zero. This is equivalent to just not using the part of the kernel sticking out.



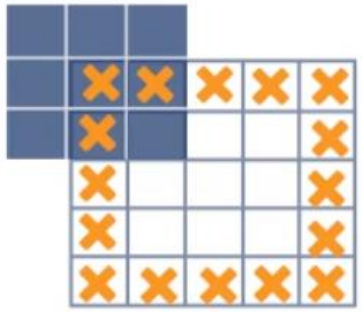
- Extend the image outwards using the values from the actual image. This is like placing a mirror at the edges.

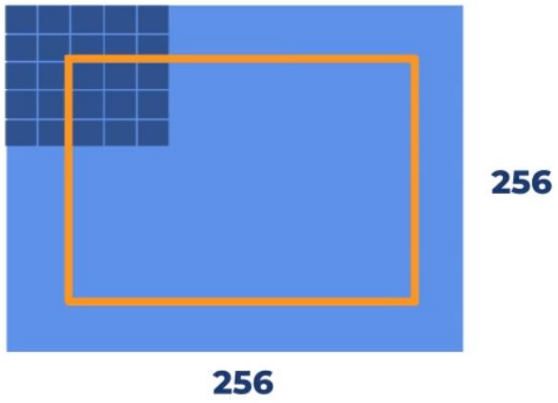


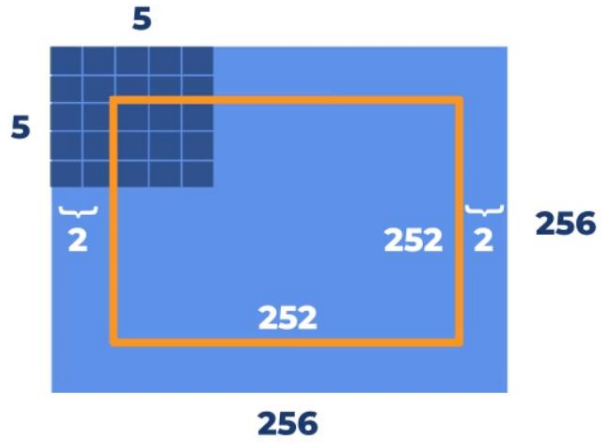
365 DataScience

Edge handling

- An out-of-the-box solution is to ignore the pixels for which the kernel sticks out. Essentially, that would trim the border of the image. This is not a big deal when dealing with big images, as if we have a 256x256 image, with 5x5 kernel, the result would be 252x252 transformed image.







From convolution to CNN

- A **convolutional layer** outputs convolutions of its inputs and some kernels
- In turn, Convolutional Neural Networks (CNNs) are deep networks that have at least one convolutional layer
- In CNNs, the point of the kernels is to transform an image in a useful way
- However, we don't manually set the kernel matrices. We just let the network find out what kernels would do the job best

CNN motivation

Why use CNNs, instead of normal feedforward neural networks?

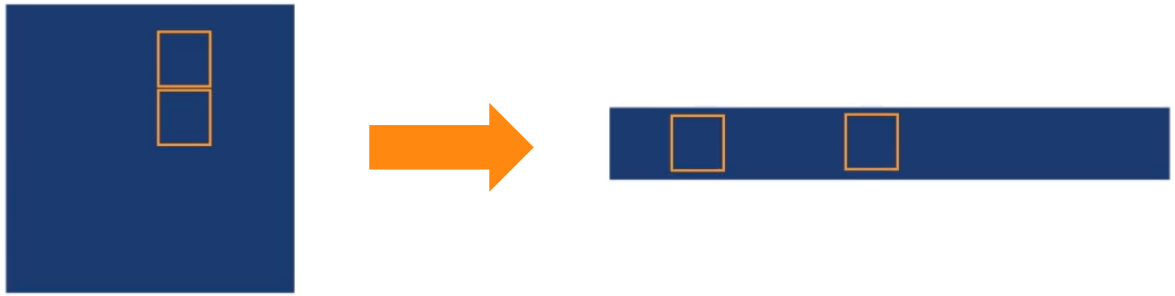
- CNNs are specialized for structured data. They preserve the spatial structure of the image since they transform a 2D input into a 2D output (the transformed image)
- Feedforward networks, in contrast, first unwrap the 2D input image into a 1D vector (row of pixels). This means that the spatial information has been lost

365 DataScience

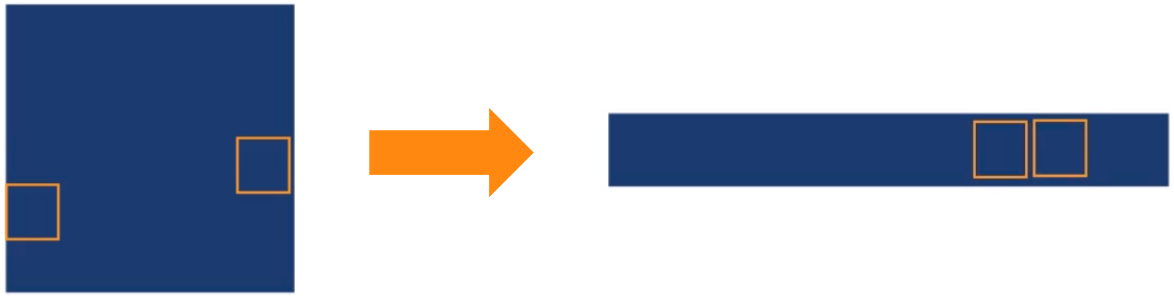
CNN motivation

Why use CNNs, instead of normal feedforward neural networks?

- Some pixels that were close to each other in the original image, are now far apart

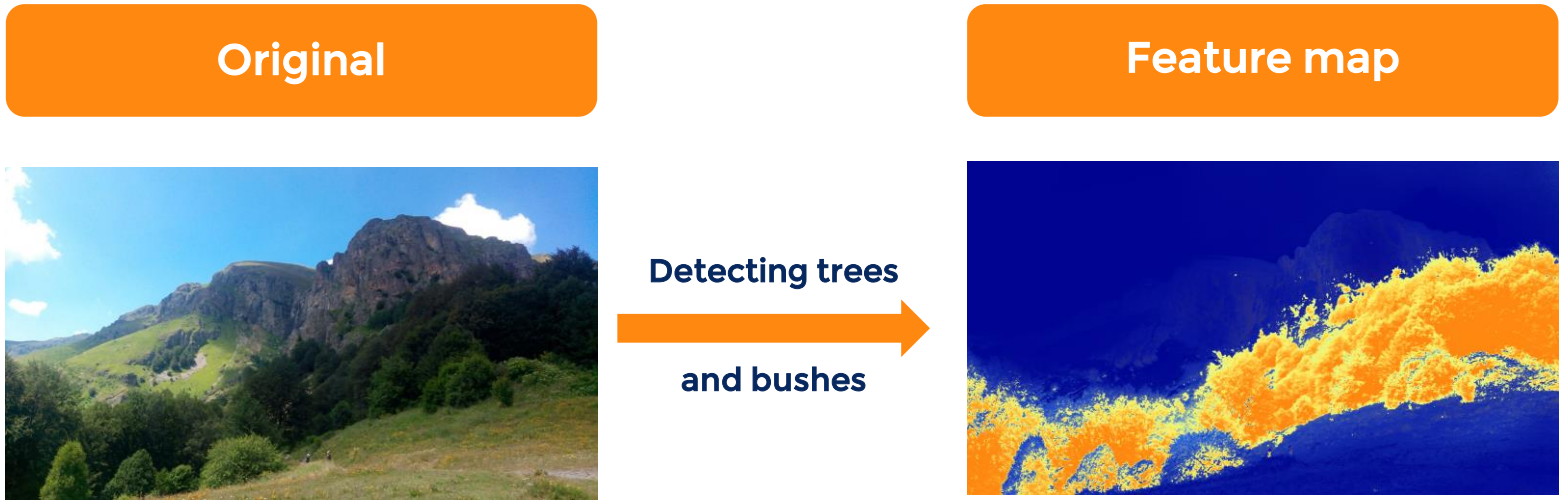


- Others that were separated, appear next to each other in the vector



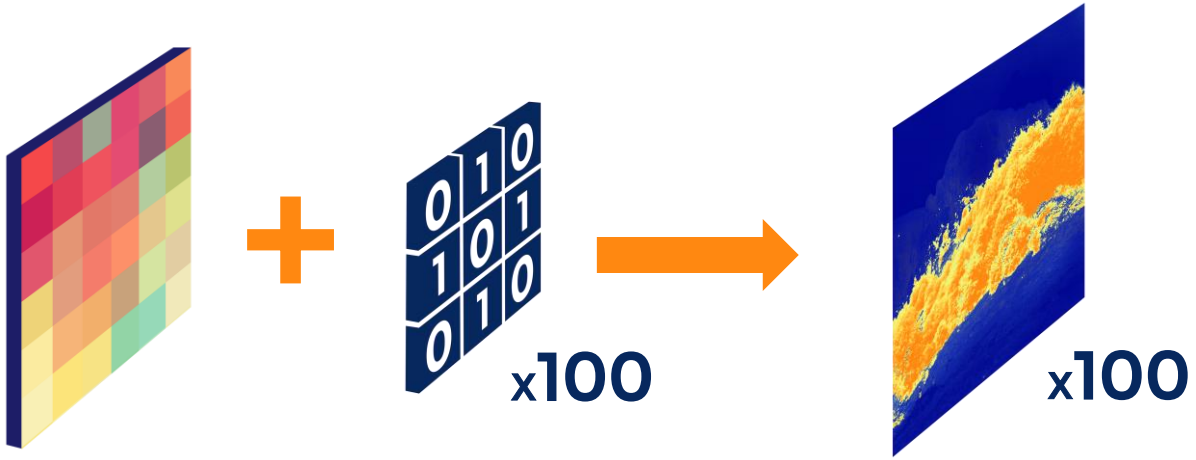
Feature maps

- In CNNs, the kernels that transform the image are usually most helpful if they are constructed as detectors. For instance, the GDE detection kernel is one such example
- These kernels are trained to search for specific patterns or features – circles, arcs, edges, vertical lines, horizontal lines etc. Thus, the resulting output is not an image, but a **feature map**
- A feature map shows how much of the corresponding feature is present at a location in the original image. It is literally a map showing the presence of features



Feature maps

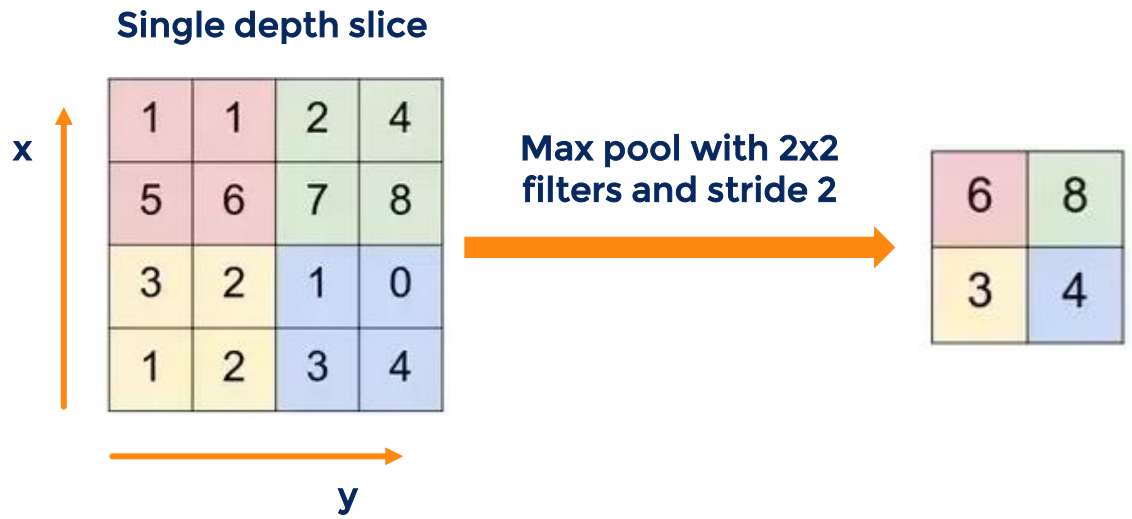
- A single such feature map is not very useful. That's why a convolutional layer contains many kernels (hyperparameter) that produce many different feature maps



- Thus, a convolutional layer consists of N kernels, each with dimensions M x M x D. Both N and M are hyperparameters of our network. However, the depth of the kernel D is determined to be the same as the depth of the input. So, for a grayscale input, D would be 1; for a color image D would 3; and for an input consisting of 50 feature maps constructed by a previous conv. Layer, D would be 50

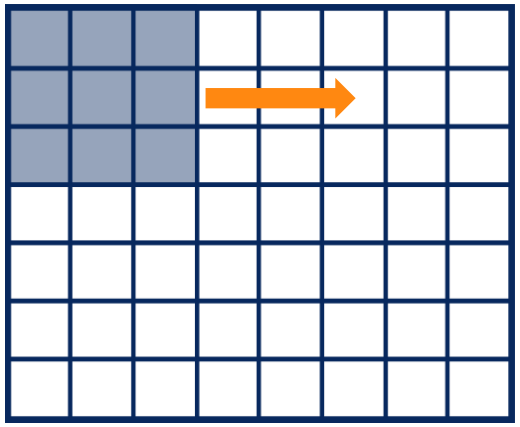
Pooling

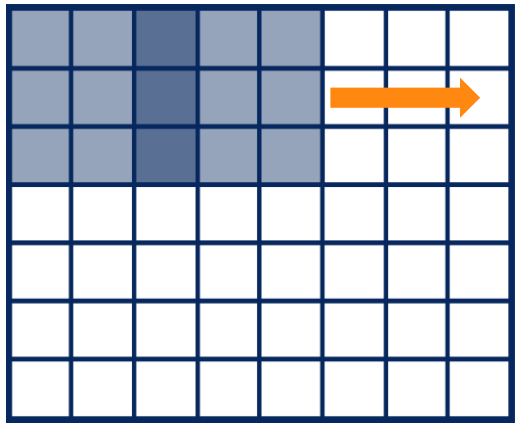
- Besides the convolutional layers, there is a second ingredient in the making of CNNs – pooling layers.
- The main purpose of pooling layers is to condense the feature maps to a smaller size. Thus, usually pooling layers are situated right after convolutional layers.
- The most popular pooling is 2 by 2 MaxPooling with stride 2. It partitions the feature map into 2x2 regions, and extracts the maximum from each region.

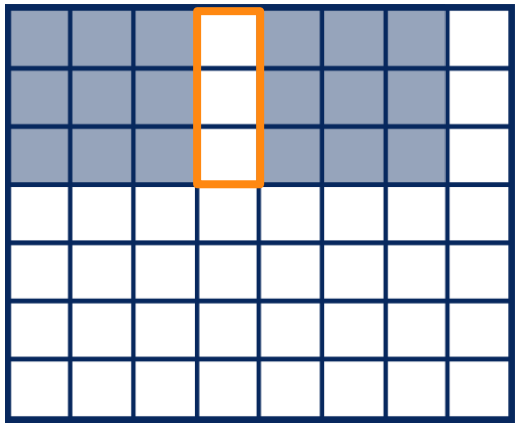


Stride

- Stride refers to the amount of pixels the kernel moves.
- For example, in the previous maxpool example, the stride was 2, as the regions have been calculated for each 2 pixels.

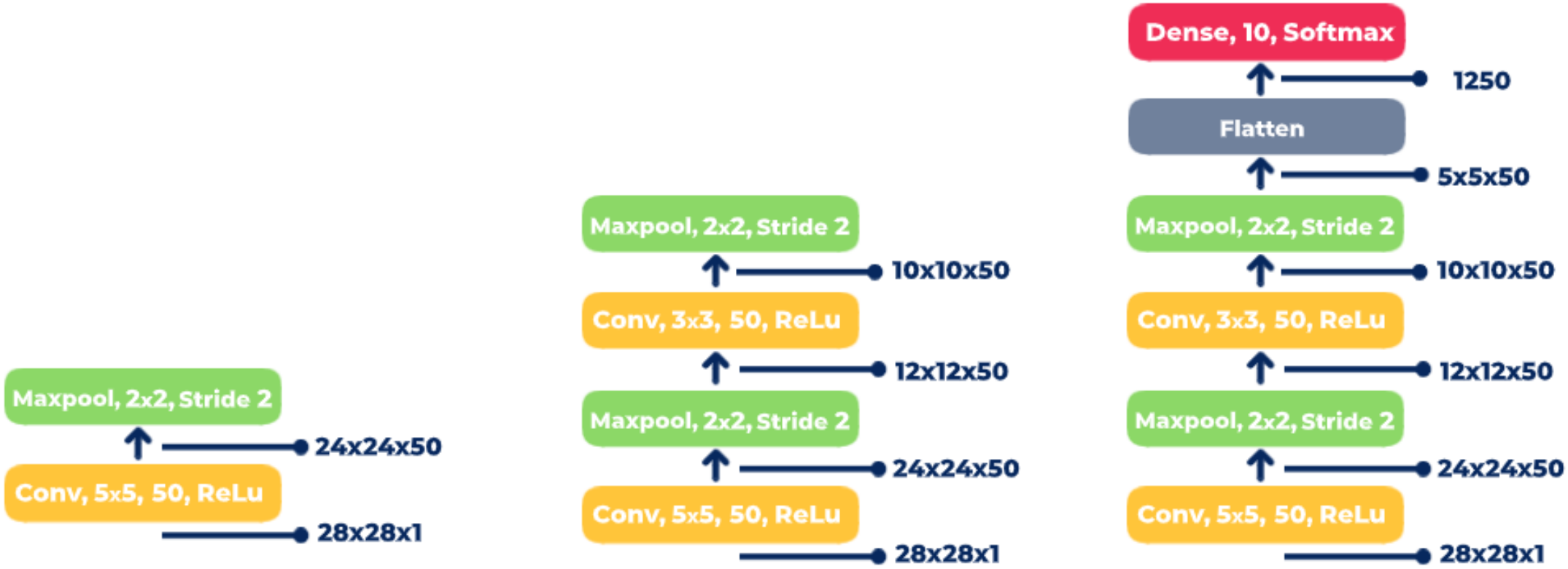






Example dimension transformation

Transformed by a single CNN



365 DataScience

Common techniques

To improve the network's performance

When considering the performance of our model, there are some techniques we can employ in order to prevent overfitting, or simply to increase the accuracy. These are not intended only for CNNs, but all kinds of networks.

Popular ones are:

- **L^2 regularization**
- **Weight decay**
- **Dropout**
- **Data augmentation**

Common techniques

L^2 regularization

Regularization, in general, is the technique of adding factors to the loss function, in order to control what the network is learning.

L^2 regularization specifically, adds this factor to the loss:

L^2 regularization equation

$$L = L_0 + \lambda \sum w_i^2$$

Weights of the network

Hyperparameter to control the scale of the effect

Common techniques

L^2 regularization

This discourages the model from learning a very complex solution, thus limiting overfitting.

Due to this factor :

The update rule

$$w_{new} = (1 - \lambda\eta)w_{old} - \eta\nabla_w L_0(w)$$

Learning rate

Common techniques

Weight decay

Weight decay is similar to L^2 regularization, however it changes the update rule directly, instead of doing that indirectly through the loss function

The update rule
Weight Decay

$$w_{new} = (1 - \lambda)w_{old} - \eta\nabla_w L_0(w)$$

The update rule
L^2 regularizatin

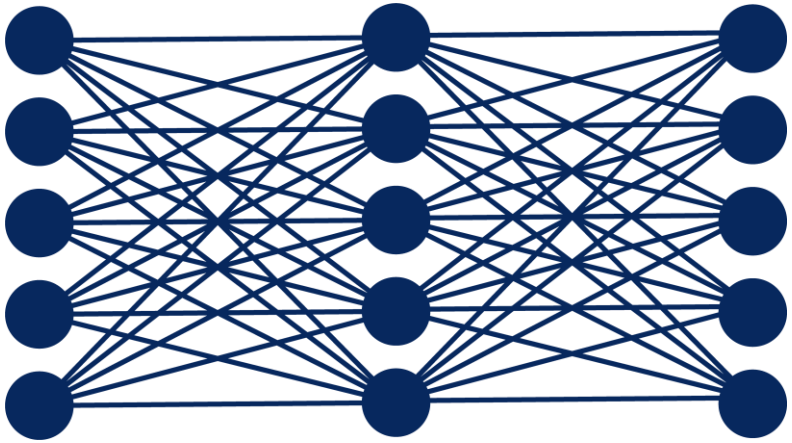
$$w_{new} = (1 - \lambda\eta)w_{old} - \eta\nabla_w L_0(w)$$

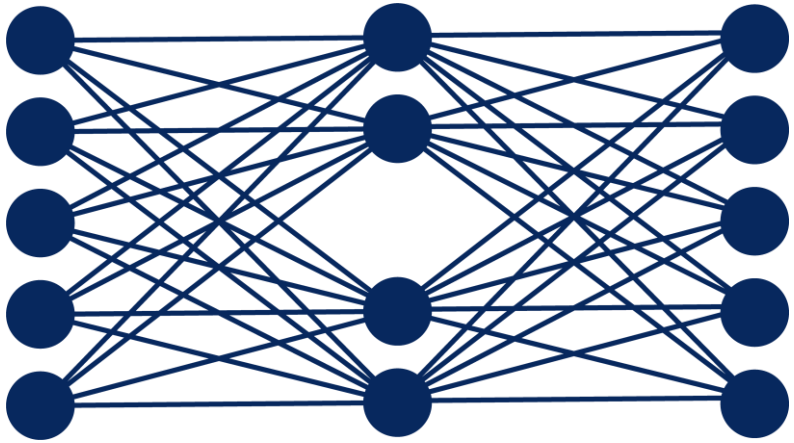
- The only difference to the L^2 regularization update rule is the missing learning rate in the brackets
- Thus, for optimizers with static learning rate (e.g. SGD), weight decay and L^2 regularization are equivalent
- However, for optimizers that use adaptive learning rate, the effects of L^2 regularization would be different in the beginning and end. In contrast, weight decay would have constant effect no matter the optimizer

Common techniques

Dropout

- Dropout consists of randomly setting a portion of the neurons in a layer to zero. This creates some form of redundancy in the layer and helps with the generalization of the network

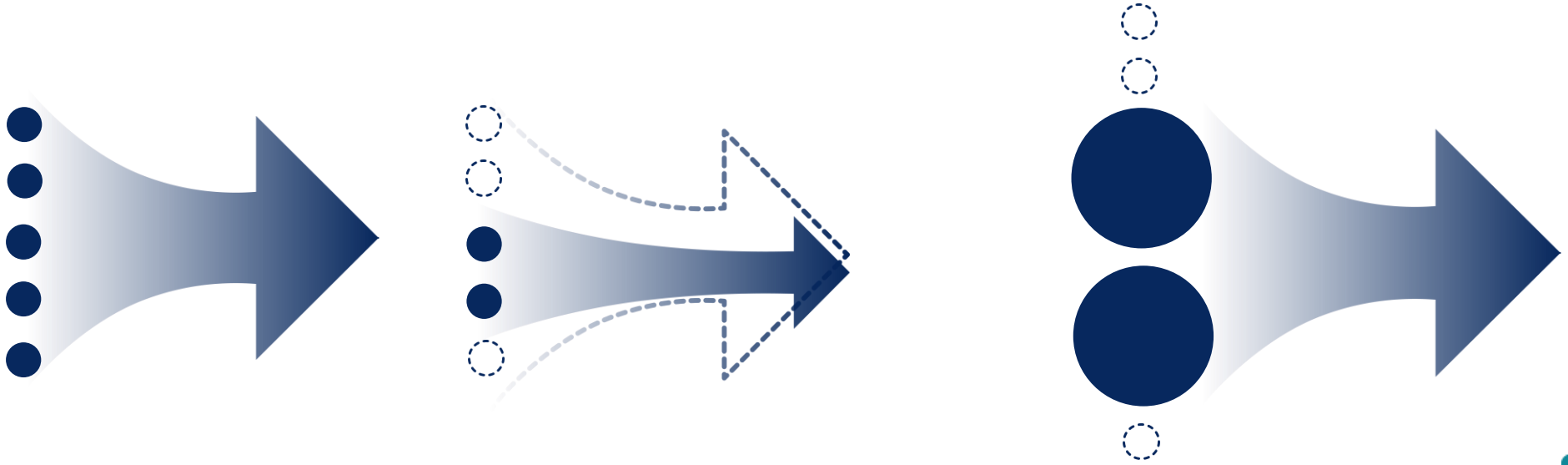




Common techniques

Dropout

- Dropout is present only during training. During testing and operational use of the model, all neurons are present
- To work properly, the remaining outputs of the given layer should be scaled up. If the portion of neurons to be dropped is p $(0 < p < 1)$, then the scaling factor is $\frac{1}{p}$



Common techniques

Data augmentation

- Data augmentation is used when the data we have available does not include examples of all classes we would like our model to learn

- For example, if we want to classify images of cats, ideally our dataset should include pictures of cats in different poses. In the case our dataset contains only cats facing to the right, we can correct that with data augmentation

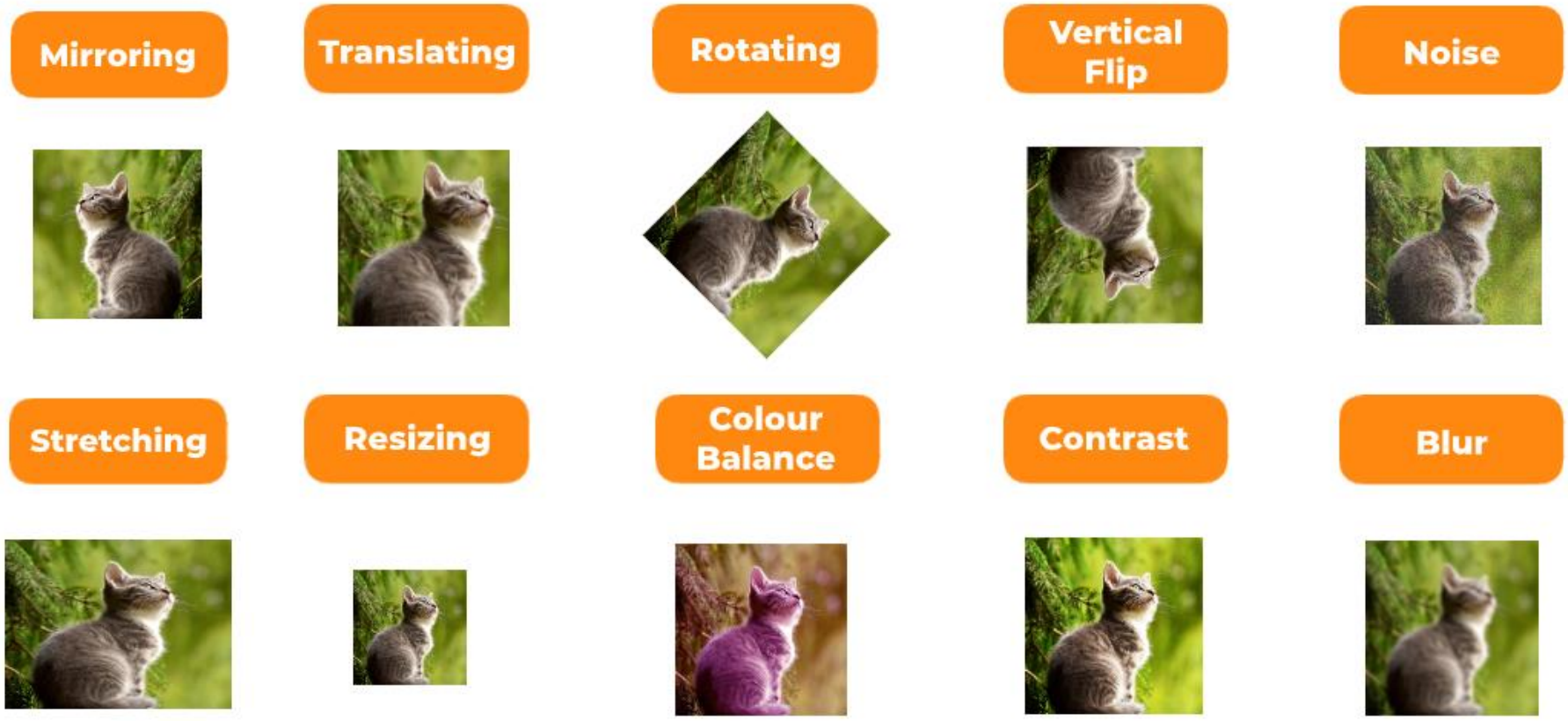




Common techniques

Data augmentation

- Data augmentation itself is the technique of transforming the data to artificially create more examples. This includes mirroring the image, translating, stretching, scaling etc.



Popular CNN architectures

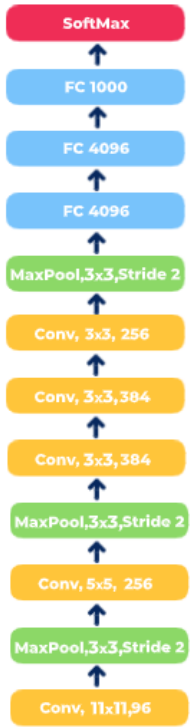
As a final note, let's take a look at some of the popular CNN architectures created by the professionals in this field.

The ones we will discuss are:

- **AlexNet (2012)** – CNN success
- **VGG (2014)** – more layers
- **GoogLeNet (2014)**- computational efficiency
- **ResNet (2015)** – revolution of depth

Popular CNN architectures
AlexNet

AlexNet was a relatively straightforward CNN, with 5 convolutional layers, 3 maxpool layers and 3 dense layers. It was one of the first easy to produce networks that had success and it started the spree of CNN research



5x Conv

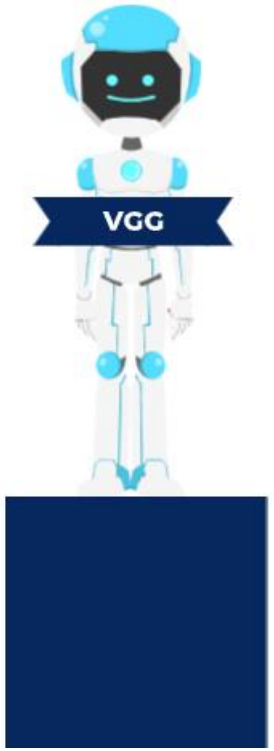
3x MaxPool

3x Dense

365 DataScience

Popular CNN architectures
VGG

VGG added more layers than AlexNet, which improved the results drastically. The trick VGG employed was to make all convolutional layers with kernels of minimum size – 3x3. This allowed for more layers to be stacked with fewer overall parameters

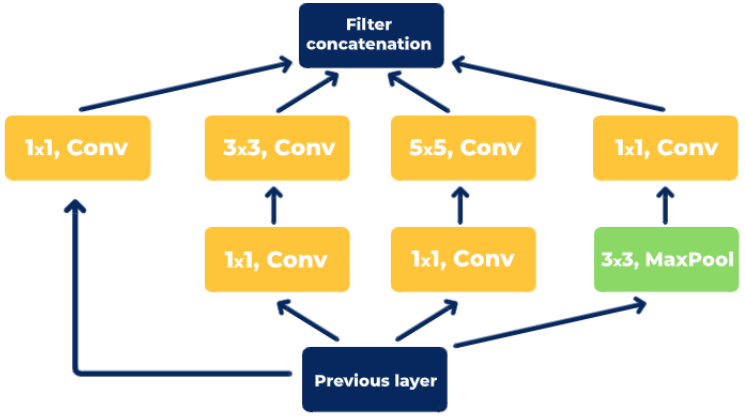


3x3 Conv

365 DataScience

Popular CNN architectures
GoogleNet

This architecture was all about computational efficiency. The team at Google designed the so called Inception module, and the whole network consisted of stacked Inception modules. The Inception module incorporated parallel layers and a 1x1 conv bottleneck layer to reduce the number of parameters and operations



365 DataScience

Popular CNN architectures
ResNet

The number of layers in the ResNet architecture skyrocketed from 22 (GoogleNet) to 152! This was achieved thanks to the residual blocks. These consisted of 2 convolutional layers in which the input was summed with the output. Thus, the network needs to learn only how to change the input, not the output itself.