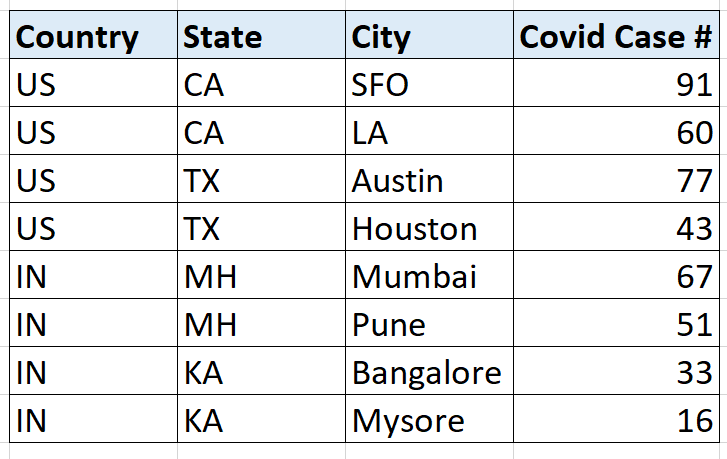
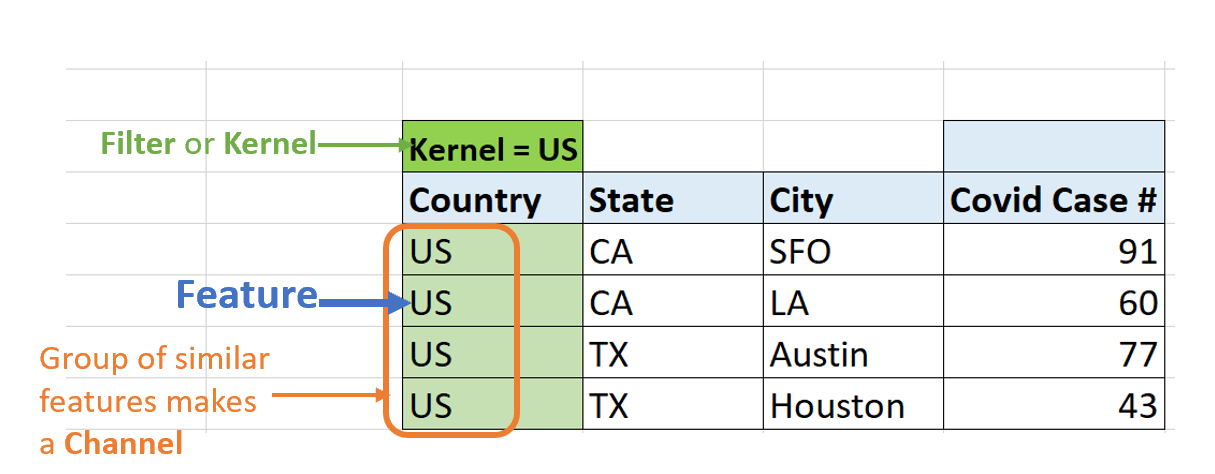
1. What are Channels and Kernels (according to EVA)?

***Kernels and Channels (intuitive explanation)–***

Kernels or feature extractor or filter, are all same. We can consider kernel as a filter to extract similar type of features or objects or information. For example, in dataset below-

[](https://camo.githubusercontent.com/ecc5407a71934e27616b410bbbc90789f19adfd4/68747470733a2f2f7265732e636c6f7564696e6172792e636f6d2f73732d64612f696d6167652f75706c6f61642f76313539353030313436312f696d616765325f6c7a6677737a2e706e67)

Here, we can consider, US and IN as two features (feature can be considered as a unique property). We can have two filters or kernels, which can help to find similar records with similar country features and the output will be-

[](https://camo.githubusercontent.com/b62951e31471da514b5c6e91d532593ca7783848/68747470733a2f2f7265732e636c6f7564696e6172792e636f6d2f73732d64612f696d6167652f75706c6f61642f76313539353030313436372f696d616765335f6b77747a64652e706e67)

Similarly, we can have filters/ kernels state = CA or TX and then for city. Records with similar feature like country (US) makes a channel.

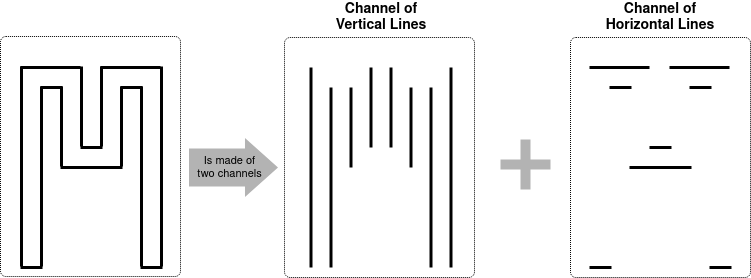
***In the case of CNNs-***

* Image is input. We use a 3x3 (generally) matrix of numbers (picked randomly and then refined while training the model) to detect features from the image.
* Usually many such 3x3 matrix are "convolved" (multiplied over pixels in every possible 3x3 area on the image) over the image.
* This convolution operation outputs a new image with as many channels (explained below) as we intended. One kernel's output is called as 1 channel. The number of channels in the output will be equal to the number of kernels we use
* Although the above points mention every kernel to be a 3x3 matrix, the truth is that it is a "Collection of N 3x3 matrices". This collection is what is referred to as a "3x3 Matrix" in the above points.
* N is equal to the number of channels in the input image
* For example, if our image has the 3 usual channels of R, G & B, then the 3x3 kernel we are talking about is a set of 3 "3x3 matrices". Each matrix convolves over 1 channel and finally adds the results to combine and create 1 output channel

***Channels -***

Now, if we consider output of kernel (Country = US), we get four records in the above example. These similar features resulting out of application of a filter makes a channel.

* In case of DNNs, Channels are on a "layer" in an image. Channel will contain one single feature from the whole image e.g. in an RGB image, the "Red Channel" contains all the red pixels from the images, the "Blue Channel" contains all the blue pixels for the image and the "Green Channel" contains all the green pixels from the image
* When each channel is laid on top of each other, we get the whole image
* The example given here talks about "Red Pixel" as a feature contained in one channel, but the feature need not be color, it could be something else too e.g. we can think of the below image as being made up of two channels: 1 channel of two vertical lines and another channel of two horizontal lines

[](https://camo.githubusercontent.com/75607282e9203500d5420e2084f9ab9e9995bf54/68747470733a2f2f7265732e636c6f7564696e6172792e636f6d2f73732d64612f696d6167652f75706c6f61642f76313539353030333539392f466f725f41737369676e6d656e745f6c6d786174662e706e67)

2. Why should we (nearly) always use 3x3 kernels?

***We should use odd-numbered size kernel as-***

* An odd-numbered kernel has a center and surrounding cell unlike an even-numbered kernel
* This property gives the model flexibility to learn many patterns better than an even numbered kernel
* An identity kernel has a "1" in the middle and zeroes all around. This would be difficult to achieve in an even-numbered kernel

***Okay, odd-numbered kernels are better but why not 5x5 or 7x7?***

* Because finally what matters in a convolutional neural network is how much of the image has been "SEEN" by the kernel and thereby learnt its features.
* "How much has it seen" is what is called as "Receptive Field" of layer. The receptive field achieved using a 5x5 kernel can be achieved using 2 - 3x3 kernels with much lesser parameters to learn (5x5 = 25 parameter to learn, while in case of 3x3 kernels, 9 \* 2 = 18 to learn, less than 5x5).
* The same holds for 7x7 kernels (receptive field can be achieved using 3 3x3 kernels.
* Therefore, 3x3 kernels are the most efficient odd-numbered kernels parameter-wise. And, most GPUs nowadays have been optimized for 3x3 kernel usage
* Also, if the kernel size if bigger, it will be difficult to learn small features in images.

3. How many times to we need to perform 3x3 convolutions operations to reach close to 1x1 from 199x199 (type each layer output like 199x199 > 197x197...)

The total times we will need to perform 3x3 convolutions to reach close to 1x1 from 199x199 is - **99**

Read the below table from LEFT to RIGHT:

199x199 > 197x197 > 195x195 > 193x193 > 191x191 > 189x189 > 187x187 > 185x185 > 183x183 > 181x181 > 179x179 > 177x177 > 175x175 > 173x173 > 171x171 > 169x169 > 167x167 > 165x165 > 163x163 > 161x161 > 159x159 > 157x157 > 155x155 > 153x153 > 151x151 > 149x149 > 147x147 > 145x145 > 143x143 > 141x141 > 139x139 > 137x137 > 135x135 > 133x133 > 131x131 > 129x129 > 127x127 > 125x125 > 123x123 > 121x121 > 119x119 > 117x117 > 115x115 > 113x113 > 111x111 > 109x109 > 107x107 > 105x105 > 103x103 > 101x101 > 99x99 > 97x97 > 95x95 > 93x93 > 91x91 > 89x89 > 87x87 > 85x85 > 83x83 > 81x81 > 79x79 > 77x77 > 75x75 > 73x73 > 71x71 > 69x69 > 67x67 > 65x65 > 63x63 > 61x61 > 59x59 > 57x57 > 55x55 > 53x53 > 51x51 > 49x49 > 47x47 > 45x45 > 43x43 > 41x41 > 39x39 > 37x37 > 35x35 > 33x33 > 31x31 > 29x29 > 27x27 > 25x25 > 23x23 > 21x21 > 19x19 > 17x17 > 15x15 > 13x13 > 11x11 > 9x9 > 7x7 > 5x5 > 3x3 > 1x1

4. How are kernels initialized?

* Kernels are initialized by random values between [0 and 1).
* There are different distributions of random numbers that can be used, but most popular ones seem to be normal and uniform distributions.
* Kernels shouldn’t be initialized by similar values or all 0. If kernels are initialized by similar values, they will produce same output and won’t be able to learn different features during back propagation.
* Nice explanation [link (Links to an external site.)](https://stats.stackexchange.com/questions/200513/how-to-initialize-the-elements-of-the-filter-matrix)

5. What happens during the training of a DNN?

During a training for a DNN, following steps takes place-

* Kernels (3x3 matrices with random numbers) "convolve" over image (multiply with pixels in every 3x3 area possible on the image) and output new images with different number of channels (that wouldn’t make sense to the human eye). The number of channels in the output are decided by the person creating the model based on his/her experience
* Point above describes operations on one layer in the convolutional neural network. There could be many such layers as decided by the person who is creating the model
* The last layer however outputs the required number of predictions e.g. if we are creating a model to distinguish an image as either "Cat Image" Or "Dog Image", the last layer should output 2 values giving us the probability of the image being a "Cat Image" or a "Dog Image"
* The output probabilities are then compared to the real probabilities (say 1 & 0, if the input is a Dog Image & the first number is the probability of the image being a dog image
* The above comparison is calculated as one final number. Whatever formula / function calculates this "comparison number" is called the loss function
* The single objective of the neural network model is to reduce this loss function to as less as possible.
* This reduction is done by an algorithm called as "Back Propagation"
* Back propagation calculates the derivative of the output in each layer with respect to the input in each layer. And in this process keeps adjusting the "weights" (the numbers inside every kernel in that layer) such that the next iteration in calculating the predictions should reduce the loss function output
* This step is conducted again and again until the predictions reach a satisfactory level of accuracy