Solution: Assignment 1

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

Channels are composed of mutliple instances of a single feature. For instance, in a colored image (RGB), the pixel intensity values for R corresponds to the Red channel and likewise G for Green and B for Blue.

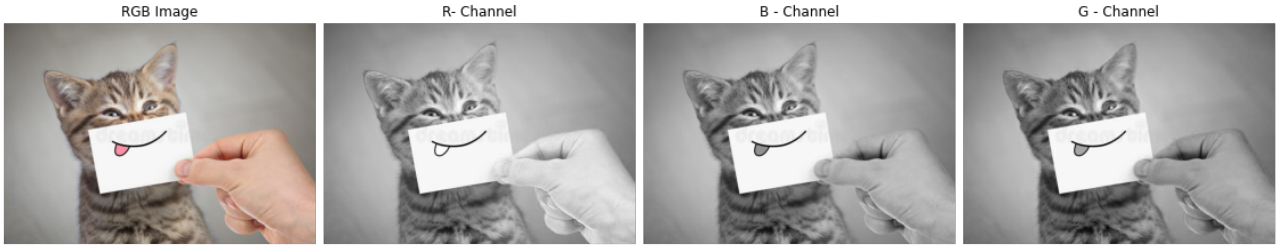


Figure 1: Image split into constituent channels

Another instance would be the composition of different emojis (i.e. the Red Emojis, Yellow Emojis and Blue Emojis), each type of emoji can be considered a emoji channel of the correponding emoji.

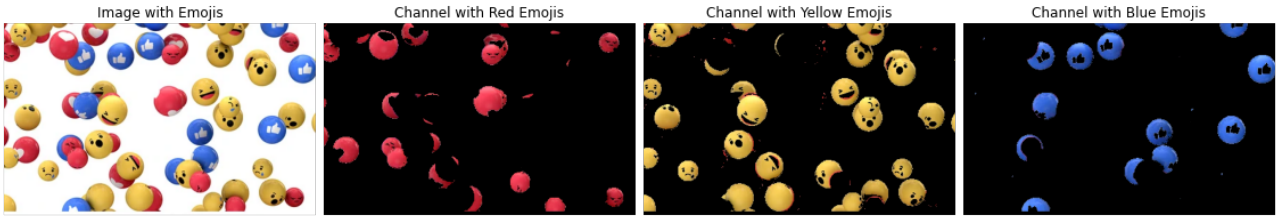


Figure 2: Emoji Channels

* Ignore the artificats created while extractions

Kernels when applied/convolved on an image extract the corresponding features, hence its also known as “Feature Extractor”. It can also be thought of as filtering out specific patterns in an image when convolved, hence they are also known as “Filters”.

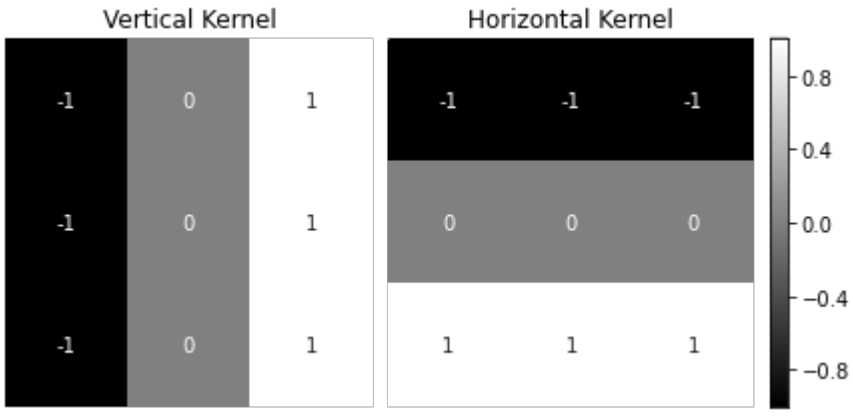


Figure 3: Vertical and Horizontal Kernels

When the above kernels are applied to an image, the components of a vertical line and that of horizontal line are extracted respectively as shown in the image below

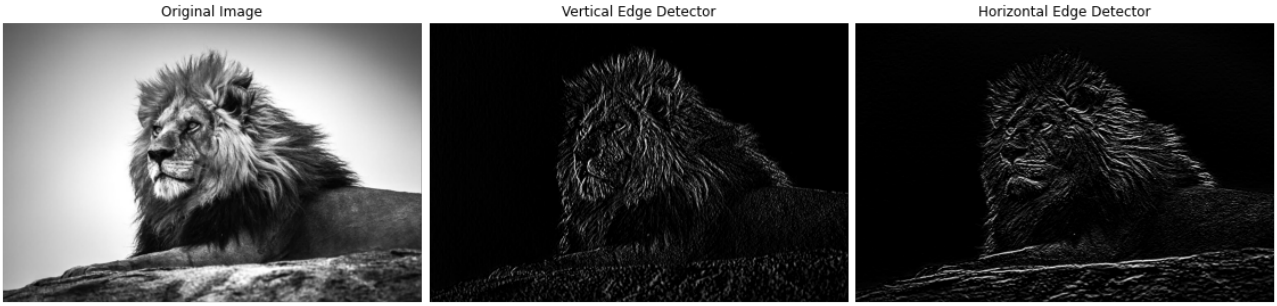


Figure 4: After applying both the kernels

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

3x3 kernels are nearly always preferred because of the following advantages,

- It can be chained to get the receptive field of a higher order kernel
- It is more efficient than any higher order kernels (such as 5x5, 7x7, etc.). For example, if 3x3 kernel is applied twice on an image of 7x7 would require 225 + 81 = 306 multiplications where as using a 5x5 kernel on the same 7x7 image would require 375 multiplications. Though the number of additions required for the 1st case is higher, addition operations are not too computationally expensive unlike multiplication operations, so using 3x3 is more computationally efficient than higher order kernels
- The use of 3x3 kernels have also been optimized to be used on GPU (by NVIDIA), so its use would result it much lesser training time.

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

It requires a total of 99 convolutions to reach 1x1 from 199x199 when convolving with a 3x3 kernels and a stride of 1.

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

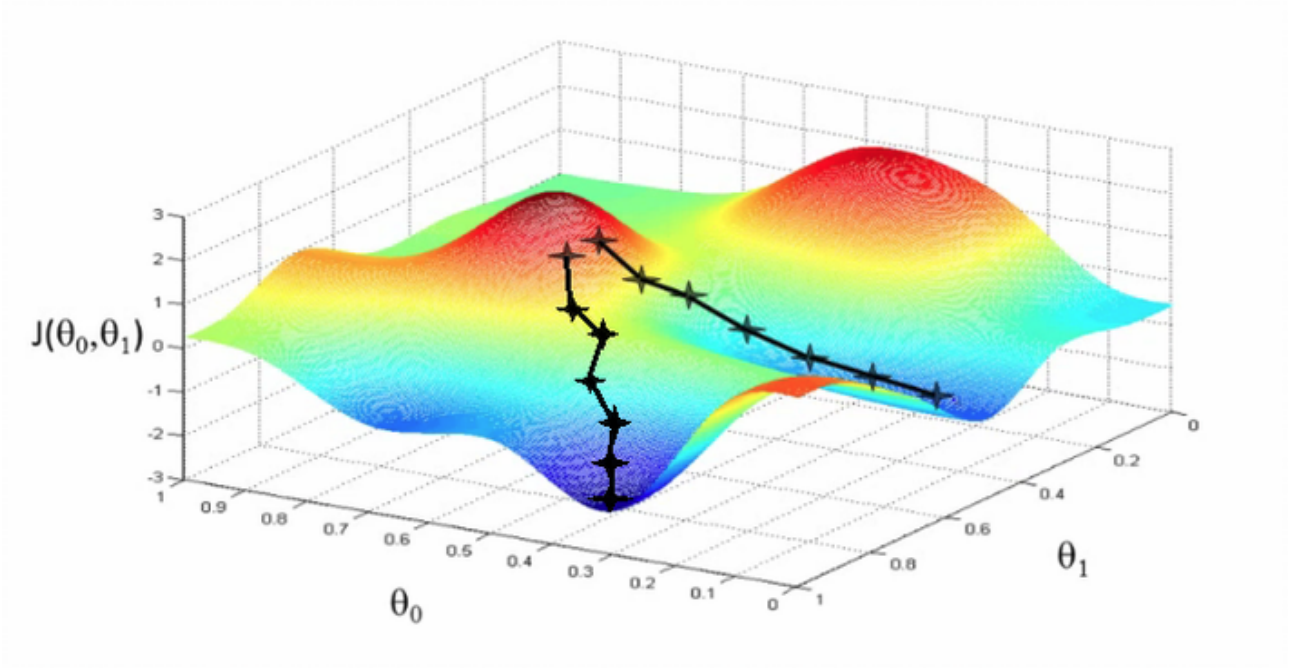
4. How are kernels initialized?

Kernels can be initalized using different methods. One of the easiest ways is to _use random values_ (without any underlying distribution). The drawback of this is if the random values are either too low or too large would results in the vanishing and exploding gradient problems (as too low values would indicated low activation value passed to the subsequent layer and so on thus resulting in vanishing the effect of that neuron, the converse applies to for larger values). An alternative could be pick random values from a normal distribution with the mean centered at 0 and a defined variance (_Xavier initialization_ and _He Normal initialization_).

5. What happens during the training of a DNN?

The training process of network involves tweaking the initalized weights such that the results match (to the best it can) with the ground truth. In neural nets this starts by computing the initial values based on the initialized weights to get the output. This output is then validated with the actual values (ground truth) and the error/loss is computed (using the chosen loss function such as MSE (Mean Squared Error) for regression or LogLoss for classification etc. or any suitable error metric depending on the usecase).

The objective of learning/training the network is to reduce this loss/error and thus an optimization function is chosen (such an Stochastic Gradient Descent (SGD), Adaptive Gradient (ADAGrad) etc.), which help to converg at a minima (minimum loss). The process of getting to this minima involves the forward pass, i.e. using the initialized/previous weights and making a prediction and backpropogation, i.e. passing the information regarding the error back to the previous layers so that they can update their weights according (which would help reduce the error).



The image depicts gradients descent, with the Cost Function J (Θ_0, Θ_1), defined on the parameters Θ_0 and Θ_1 . As iteration progresses, we see that the loss (or error/cost) decreases and steadily reaches the minima. Caveat: One of the problems could be that the network would reach a local minima of the function and to aviod this the learning rate could be tweaked and/or multiple runs with different initalization could be tried out so that the global minima could be found atleast on of the runs.