CNN Steps

1. Feature map:

* Reduces to a smaller map with feature information in place. This feature information is different if convoluted with a different feature detector.
* Different feature detector/filter/kernel gives different feature map (blur, emboss, sharpen, etc.)
* This process of extracting a feature map is known as convolution.
* Input Image feature detector = feature map

1. Remove linearity. Increase nonlinearity using ReLU.
2. Pooling/Down Sampling – [q: why we need it? a: to allow more flexibility]

* Introduces a property called spatial invariance.
* Usually when a CNN learns the feature during the training, it tends to find the feature according to the feature map at a particular place in a particular way. If the conditions in the test images are different like the lighting condition, the subject location or is bit distorted, etc. the CNN will fail to recognize it.
* It is for this reason that spatial invariance needs to be introduced and this can be done via Pooling.
* This is applied on the feature map resulting in pooled feature map with a reduced size.
* This reduces overfitting, how?
  + As the size of the final map (i.e., pooled feature map) is reduced the number of nodes required to pass on to the CNN is reduced. This results in eliminating the chances of overfitting our test data as the pooling allows us to get rid of the other unimportant information and keeps the spatial feature with spatial invariance to generalize more on the test data (irrespective of conditions in the test image). Hence overriding gets eliminated.
* Types of pooling: max pooling, mean pooling, sum pooling, etc.

1. Flattening

* To flatten the pooled feature map as an input for the ANN in the next stage.

1. Full Connection

* Fully connected NN
* Why is it needed?
  + Leveraging the features extracted and optimized from CNN.
  + Features from the feature map is flattened and passed as an attribute to the fully connected neural network. This further optimizes the parameters for and does better generalization as we know what the ANN does.
  + Also, the feature detectors are adjusted. Incase our feature detector is incorrect.
* Softmax function is used for the output node in Fully connected NN; SoftMax function helps the output nodes to be aware of the winning votes at the output layer. What Softmax function does is it squashes the value at the output nodes between 0 and 1 and makes sure the output layer add up to give a resultant value of 1 (to abide by the rule of probability), since the values at the output necessarily does not guarantee that their addition would yield 1 as the output. E.g., 80% dog and 45% cat wouldn’t make any sense if you look it from the probability POV.
* Cross Entropy – for minimal error it takes care of the gradient so that the optimization can go on in a considerable way (explained in a very layman’s term)

Convolution Layers, ReLU Layers & Pooling Layers

The different convolved Images in the convolution layer are the result of different filters that were used to get these convolved images.

ReLU operation is applied before the Pooling layer. This is essential because the ReLU helps in normalizing the features in the convoluted images. ReLU converts every negative value (or features in the convolved image) to 0. This is important because it may mess up the pooling and negative numbers might lead to unwanted calculations\*. To avoid this, we use ReLU as our choice of activation function. More importantly it brings no linearity in the model like all the other activation functions.

\*The activation functions help us in fitting the non-linear problems. Without activation function out model will only output linear outcomes. It’s more like moulding the outcome of neurons (mx + c) in a way that it changes its slope and offset. If this is implemented on all the neuron’s value (i.e., mx + c) we get a curved line of non-linear shape. This is how a non-linearity is introduced in a neural network. This was a very general explanation of activation functions.

As we know Pooling is responsible for down-sampling our normalized feature map.

Pooling if done for all the feature maps from convolution layer (one feature map 🡪 one pooling window) hence getting the pooled feature map and the no.of these are same as that of no.of normalized feature map.

Further these maps from the Pooling layer and then forwarded to the convolution.

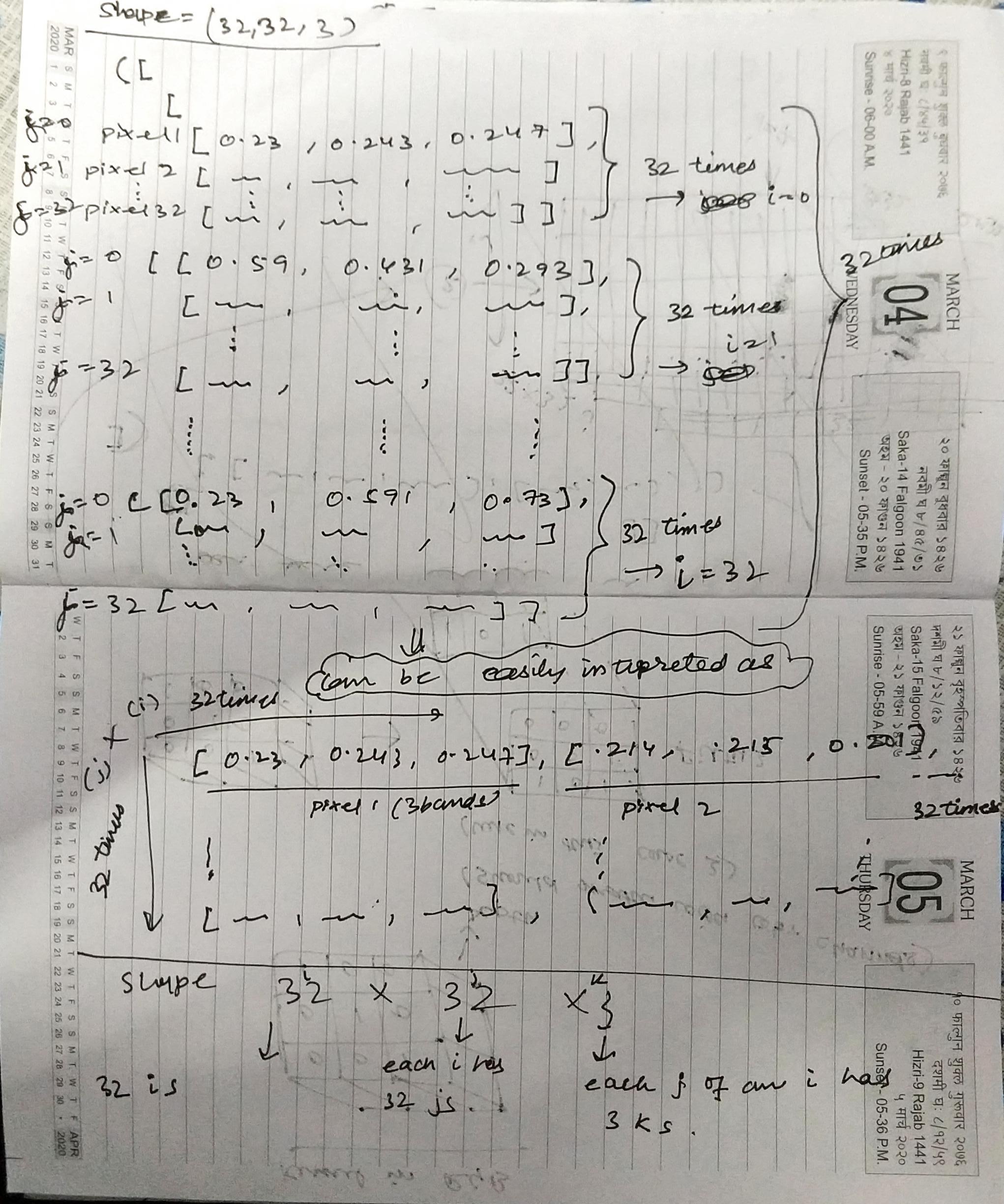
The Filters now get applied to all the pooled feature map one at a time (one pooled feature map \* n kernels/filters) goes on for all the pooled maps. Then the ReLU is applied to all these convolved images and pooling happens after that. This is an iterative process.

At the final stage the output from the pooled layers is flattened and forwarded to the ANN.

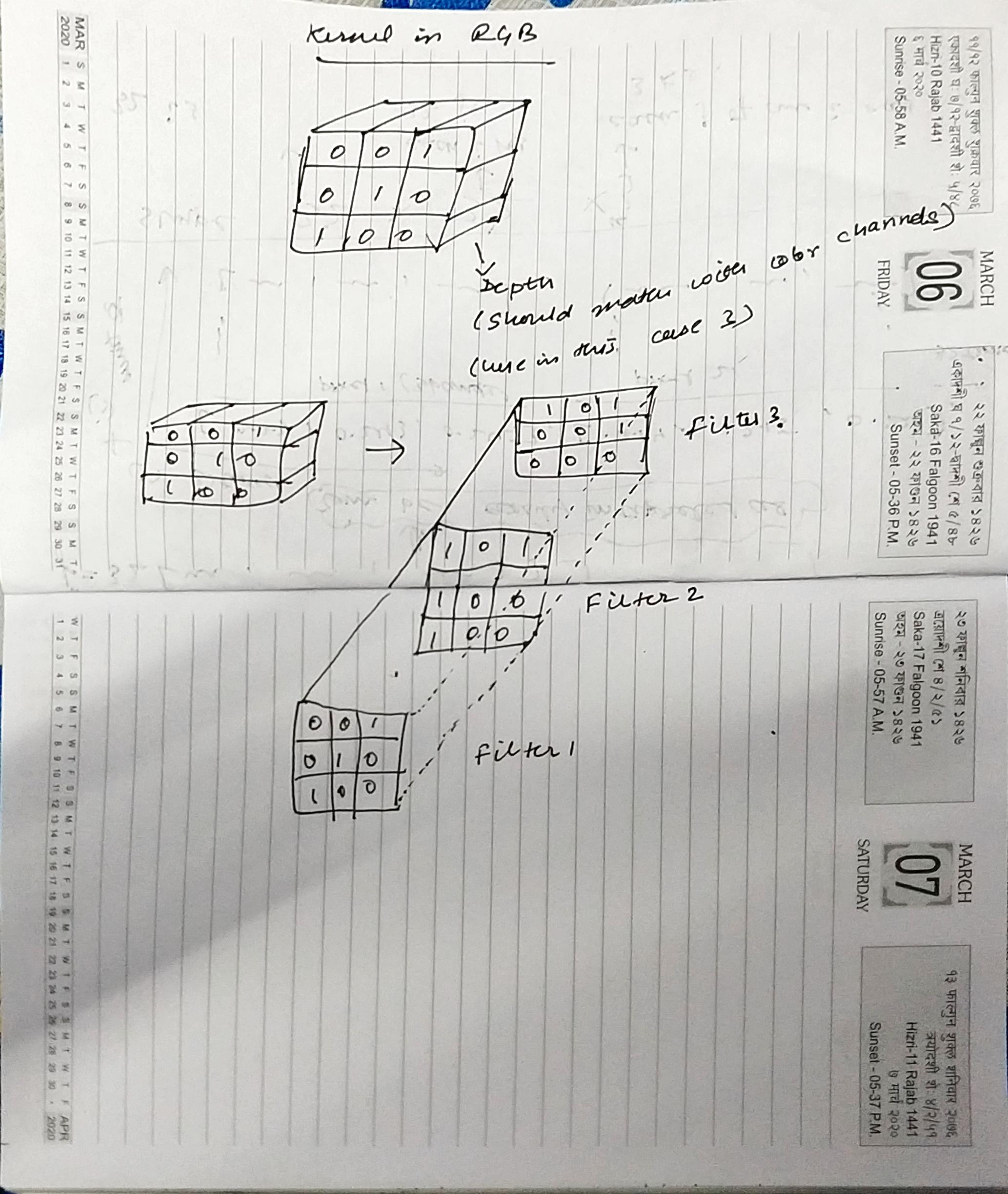
The ANN learns the positions which are important for any class/category (i.e., what important nodes are responsible for classifying an object and takes the vote for the predictions. The highest number of votes among the categories become the predicted output y^).

RGB image intuitions:

RGB image array breakdown

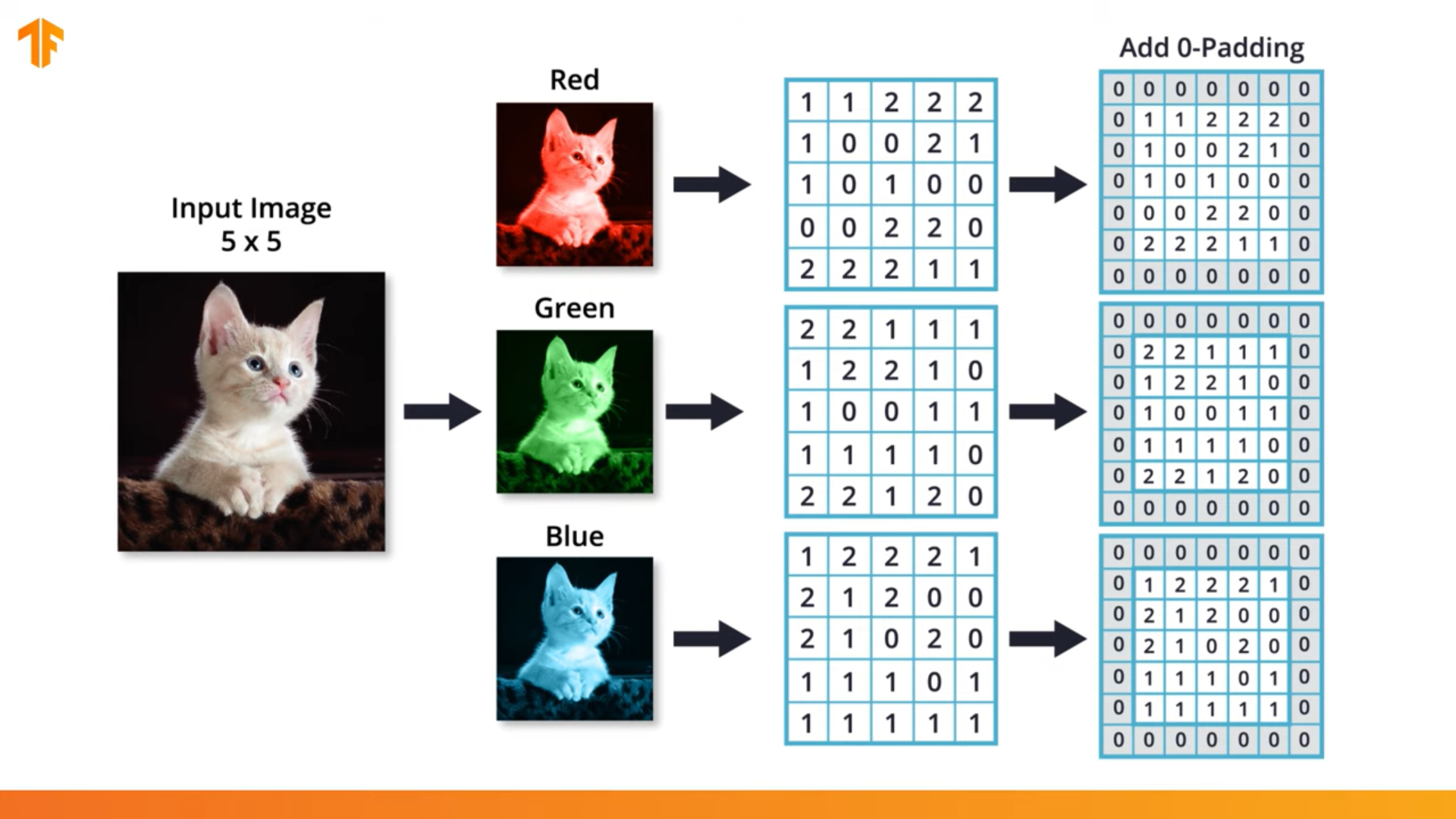


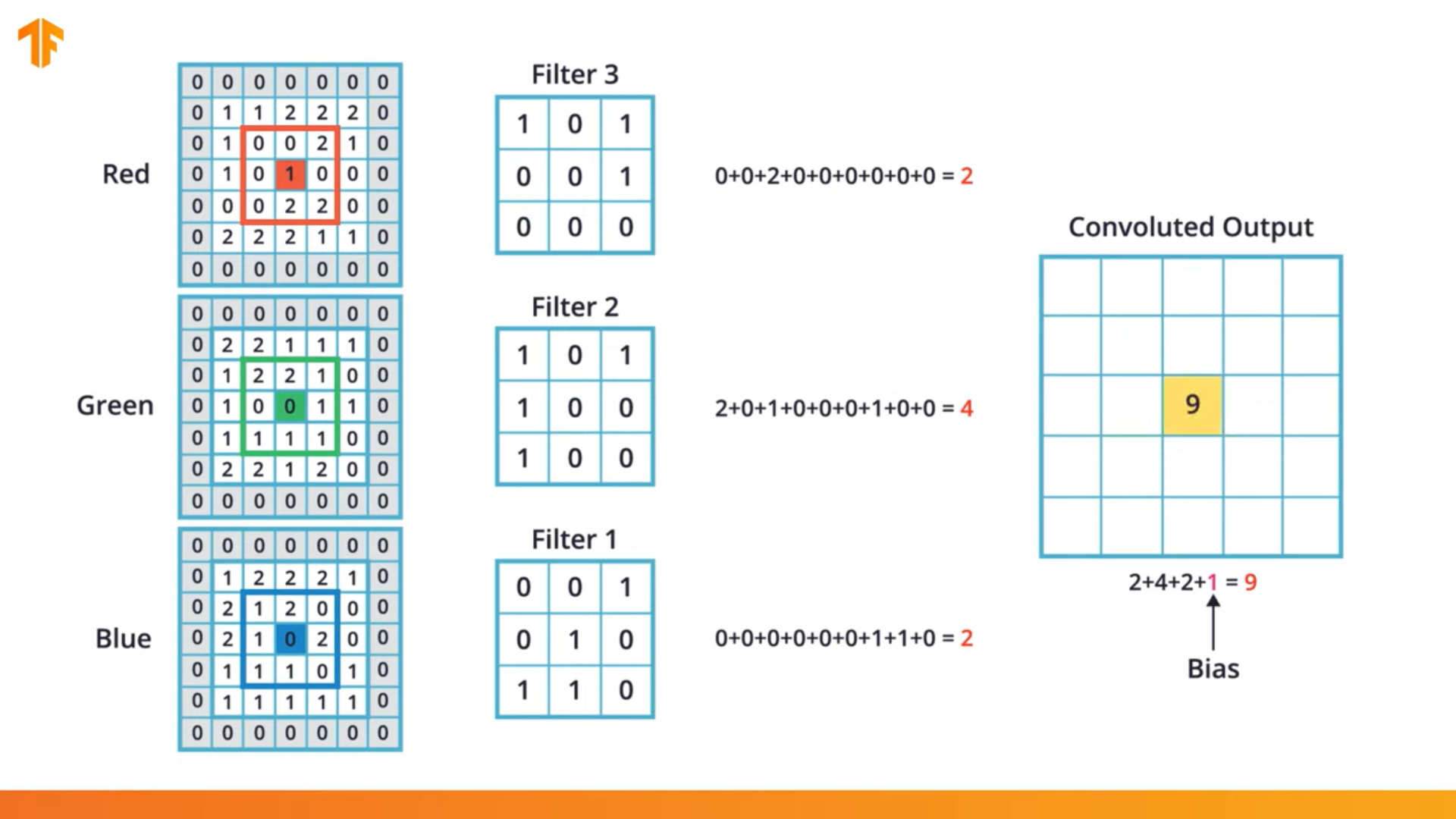
Kernel in RGB



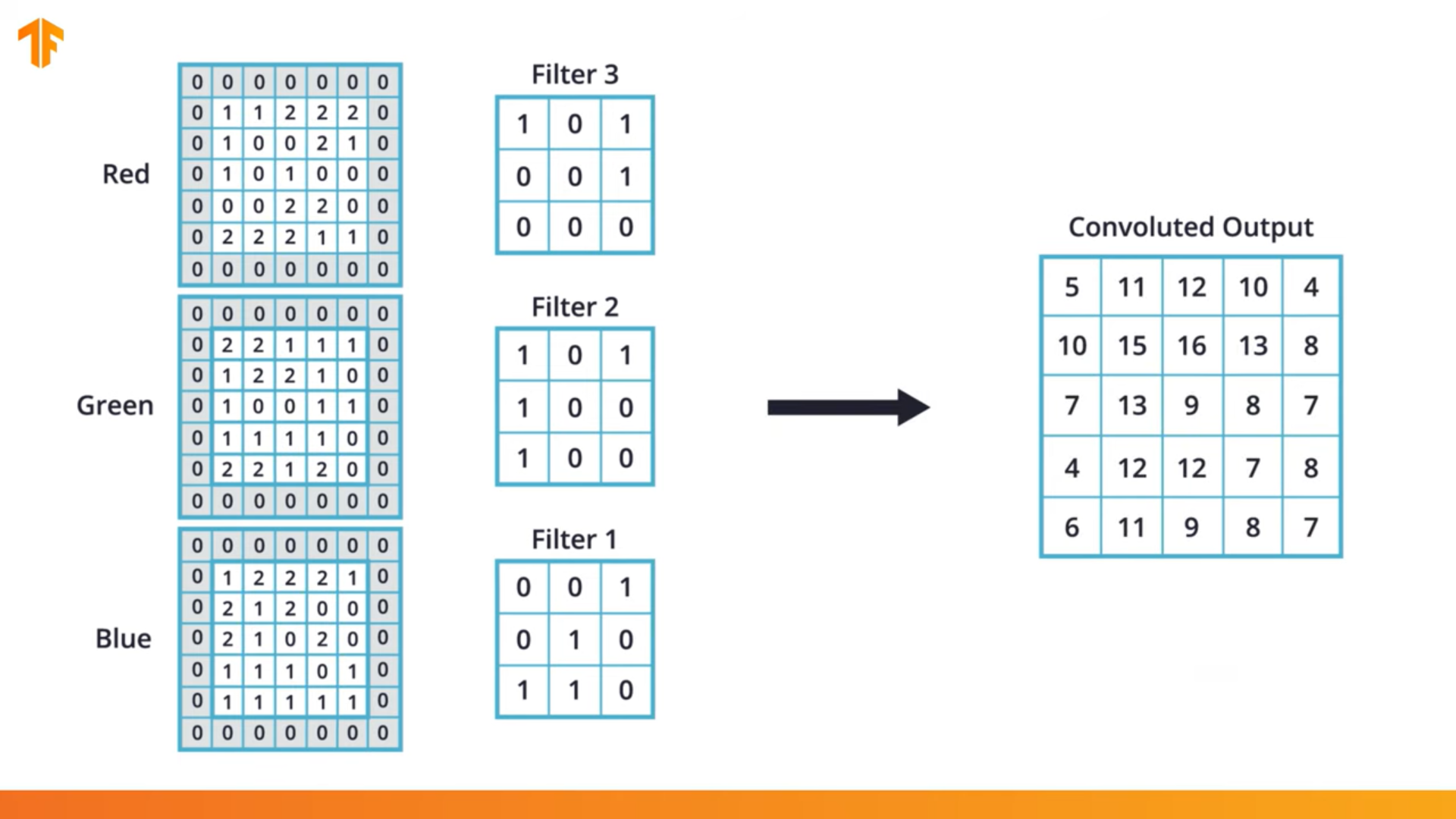
RGB images in a convolution:

Applying the above filters



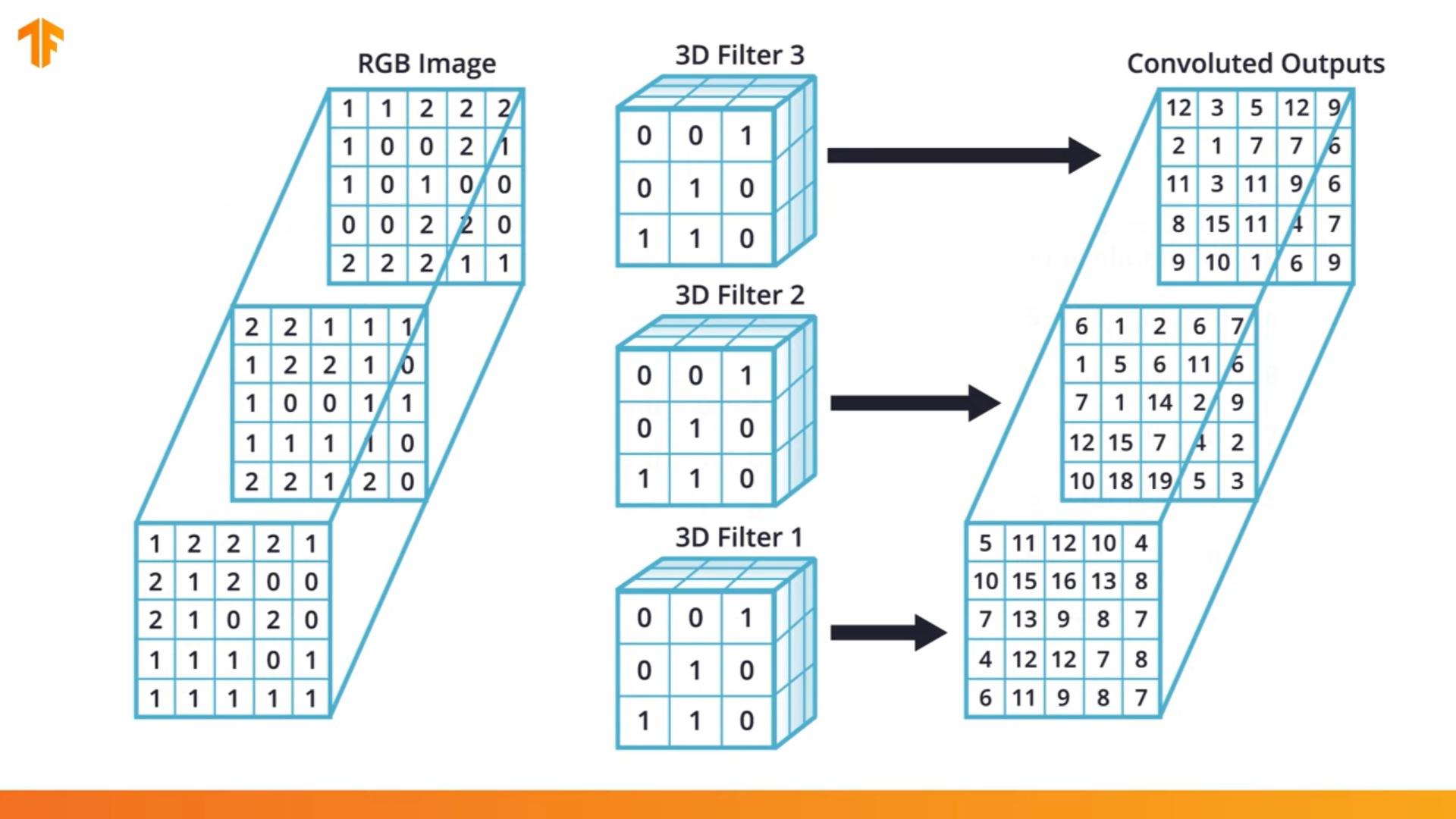


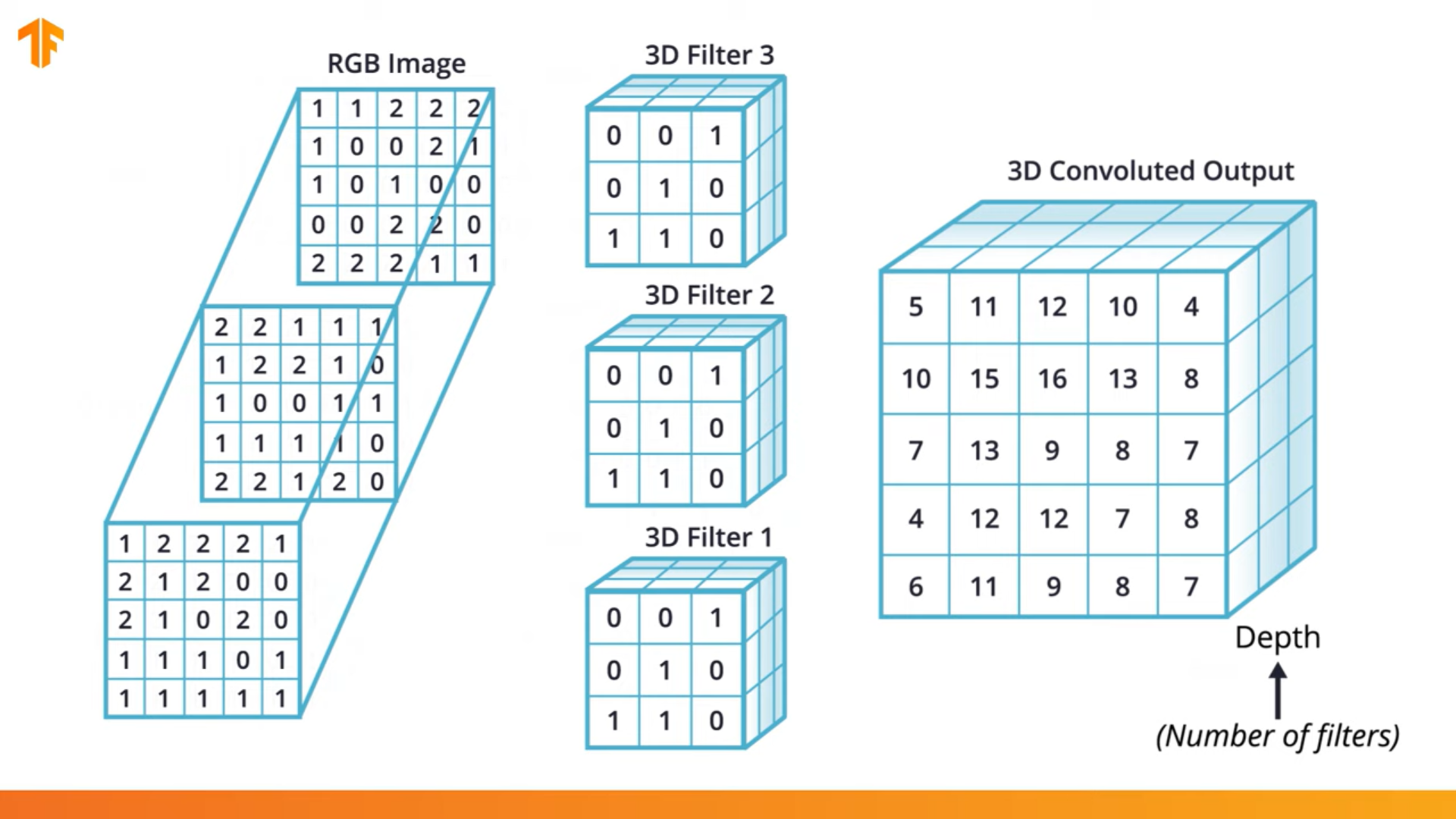
Same above operations for all pixels

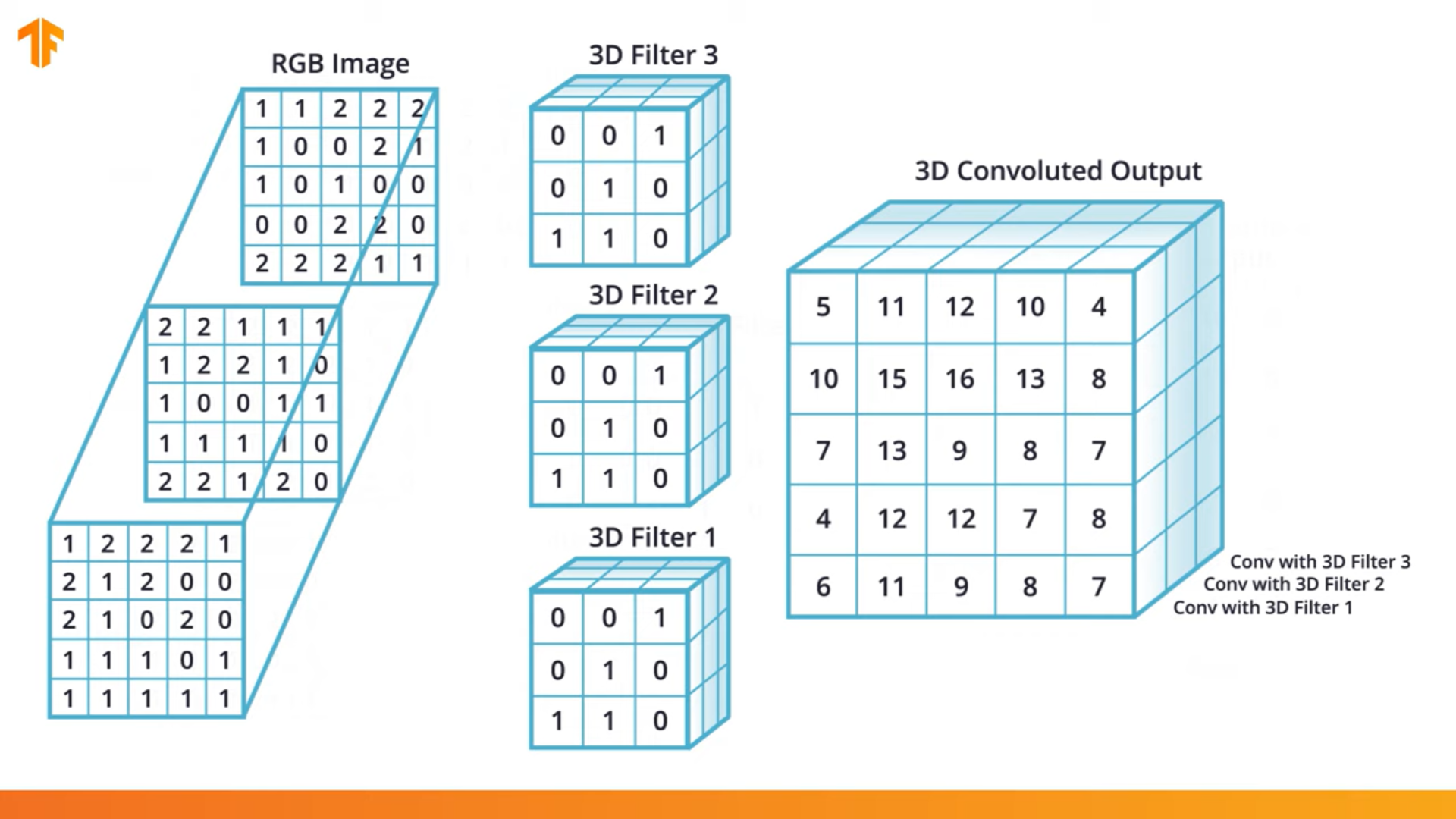


Here we got the convoluted image with the same length and height as that of RBG images

Here convolution with the 3D kernel on RGB gives the convolution with one dimension with same height and width of the RGB image.

In CNN as we work with more than one filters, here also we will be using more than one 3D filters.

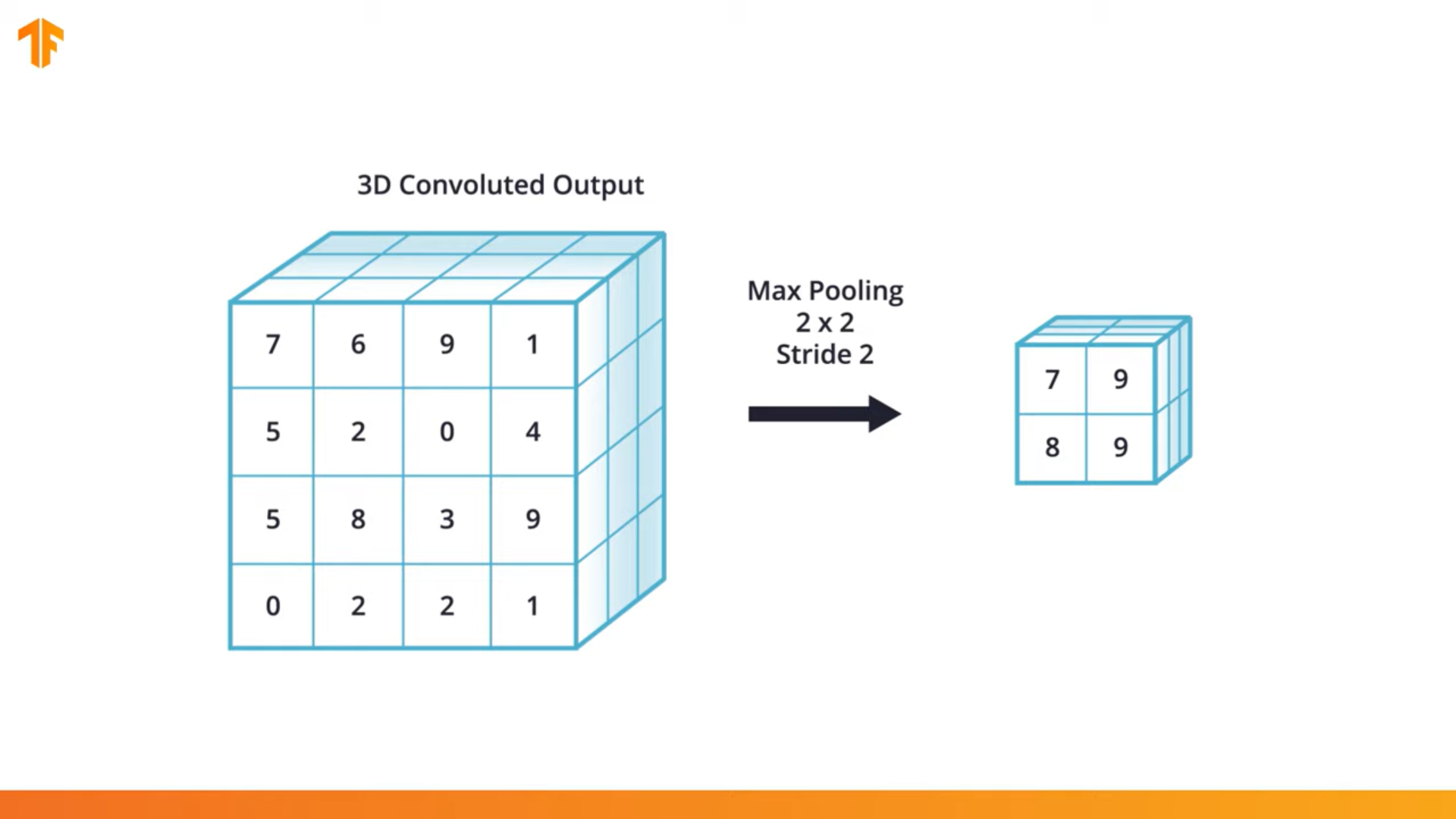




The values in kernel are also updated in order to minimize the loss function.

Max Pooling





Now the respective code makes more sense

