**Extensive Vision AI (EVA 7) – Assignment\_1**

1. What are Channels and Kernels?

**Channels:**

In order to explain the concept of channels, let’s consider multiple examples of the image below.

In the image below, music is being played by the band however, thinking in terms of a channel, the music is a composition/combination of individual instruments being played, like drums, guitar, piano, vocals etc., Hence in this example, we can consider each instrument as a channel. For example, we may have drums channel, guitar channel, piano channel.

A group of people playing instruments

Description automatically generated with medium confidence

**Kernels:**

While the channels can be considered as individual components that in combination of other components(channels) can make up say an image or a sentence, kernels, also called as feature extractors or filters are the guys who are responsible to extract the particular feature related to a particular channel. For example, one kernel works to extract only red color passing when scanning through an image(convolution) or one kernel may extract only horizontal edges or vertical edges etc.

For example, we have a 4x4 image (purple color), the kernel is the dark blue (3x3) that convolves through the image to extract, say green pixels for example. The output is a 2x2 image.

A picture containing diagram

Description automatically generated

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

**Symmetry:**A kernel with odd size (3x3, 5x5, 7x7) is always preferred as it offers an axis of symmetry at the center of the kernel matrix. This helps to capture the symmetric information.

**Resource optimization:**Consider an example where we have an image of size 5x5 and would like to convolve it so that the output image is 1x1 or the receptive field is 5x5. We could use kernels of size 3x3 and convolve it twice or we could use a 5x5 kernel and convolve it once. If we observe a 3x3 kernel would have 9 parameters convolved twice (5x5 -> 3x3 -> 1x1) hence 18 operations compared to 5x5 (5x5 -> 1x1) with 25 parameters (operations). 3x3 has an optimized computation cost.

**GPU Optimized:** not to mention, today’s GPU’s from NVidia are highly optimized to perform 3x3 convolutions.

1. 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...)

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

number of layers is 99

1. How are kernels initialized?

The weights of the kernels can be initialized using many techniques like random or gaussian or even more advanced techniques. During the training, the gradients of the weights are calculated using the process called back-propagation like chain rule to calculate the gradients. Different optimizers like ADAM, SGD or RMS prop then update the weights based on the functions of the calculated gradients.

The basic objective of appropriate kernel initialization is to avoid vanishing gradient problem or exploding gradient problem that primarily is caused due to too large weight initialization or to small weight initialization.  This could lead to either divergence or slow training.

1. What happens during the training of a DNN?

A Deep neural network has a various forms depending on the type of architecture and applications, like multilayer perceptron, Convolution neural network, RNN etc.,

At the core of the DNN architecture is operations like activation, drop outs, convolutions, max pooling etc., During the training process, there is a forward pass where in the weights are multiplied by the previous activation outputs and this operations moves sequentially in from beginning layer to last layer. Once the final prediction is made, the loss is calculated and based on the loss, the gradients are calculated which is a derivative of loss with respect to weights. Various Optimization algorithms exists such as Stochastic Gradient Descent with Momentum, Nesterov Accelerated Gradient, AdaGrad, RMSProp and Adam Optimizer, which make sure that we reach the optimal value with the least epochs. Further the weights are updated using the gradients for all the batches or training data resulting in 1 epoch. This iterative process is continued for multiple iterations, epochs till we find that the loss or accuracy metric does not change any further.

A picture containing text, watch

Description automatically generated

Ref:

[https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78 (Links to an external site.)](https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78)

[https://medium.com/usf-msds/deep-learning-best-practices-1-weight-initialization-14e5c0295b94 (Links to an external site.)](https://medium.com/usf-msds/deep-learning-best-practices-1-weight-initialization-14e5c0295b94)

<https://pathmind.com/wiki/neural-network>