# Commonly Asked Questions Regarding CNNs

## 1. How to choose the size of the convolution filter or Kernel size for CNN?

During this learning process of CNN, you find different kernel sizes at different places in the code, then this question arises in one's mind that *whether there is a specific way to choose such dimensions or sizes*. So, the answer is no. In the current Deep Learning world, we are using the most popular choice that is used by every Deep Learning practitioner out there, and that is 3x3 kernel size. Now, another question strikes your mind, why only 3x3, and not 1x1, 2x2, 4x4, etc.

Basically, We divide kernel sizes into smaller and larger ones. Smaller kernel sizes consists of 1x1, 2x2, 3x3 and 4x4, whereas larger one consists of 5x5 and so on, but we use till 5x5 for 2D Convolution. In 2012, when **AlexNet** CNN architecture was introduced, it used 11x11, 5x5 like larger kernel sizes that consumed two to three weeks in training. So because of extremely longer training time consumed and expensiveness, we no longer use such large kernel sizes.

One of the reason to prefer small kernel sizes over fully connected network is that it reduces computational costs and weight sharing that ultimately leads to lesser weights for back-propagation. So then came **VGG** convolution neural networks in 2015 which replaced such large convolution layers by **3x3** convolution layers but with a lot of filters. And since then, 3x3 sized kernel has became as a popular choice. But still, ***why not 1x1, 2x2 or 4x4 as smaller sized kernel?***

1. **1x1** kernel size is only used for dimensionality reduction that aims to reduce the number of channels. It captures the interaction of input channels in just one pixel of feature map. Therefore, 1x1 was eliminated as the features extracted will be finely grained and local that too with no information from the neighboring pixels.

2. **2x2** and **4x4** are generally **not preferred** because odd-sized filters symmetrically divide the previous layer pixels around the output pixel. And if this symmetry is not present, there will be distortions across the layers which happens when using an even sized kernels, that is, 2x2 and 4x4. So, this is why we don't use 2x2 and 4x4 kernel sizes.

Therefore, ***3x3 is the optimal choice*** to be followed by practitioners until now. But it is still the most expensive parts!

**Bonus:** Further digging into it, I found an another **interesting approach** that was used in **Inception V3 CNN architecture** launched by *Google* during the *ImageNet Recognition Challenge* that **replaced 3x3 convolution layer**

**by 1x3 layer followed by 3x1 convolution layer**, which is actually splitting down the 3x3 convolutions into a series of one dimensional convolution layer. And it came out to be quite cost-friendly!!

**Calculating Parameters of Convolutional and Fully Connected Layers with Keras**

https://dingyan89.medium.com/calculating-parameters-of-convolutional-and-fully-connected-layers-with-keras-186590df36c6

**How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras**

https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/