**Aparna K**

**21BDA24**

**What is Stride, Padding & Pooling? Explain with an example.**

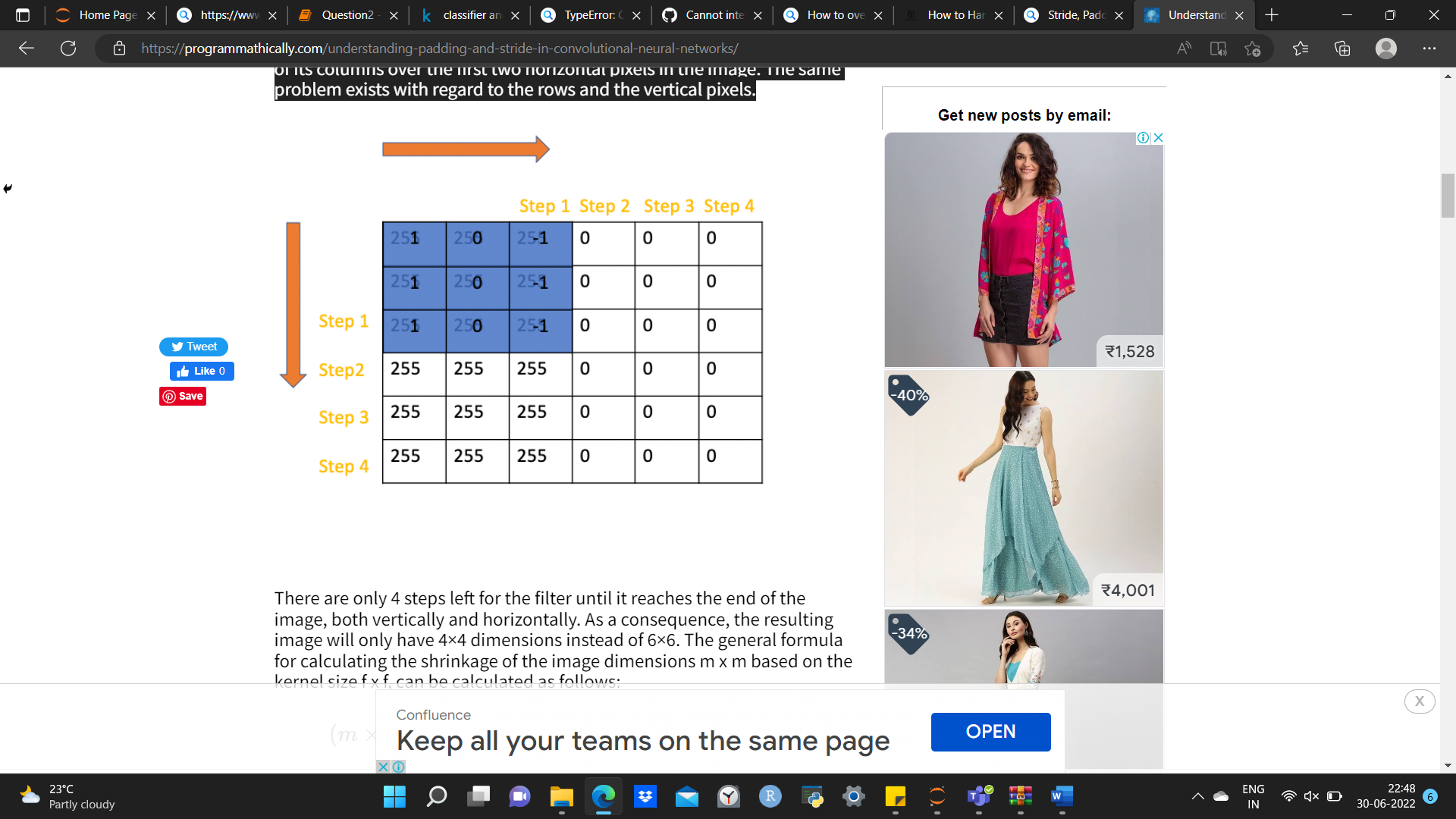
**Padding**

**Padding describes the addition of empty pixels around the edges of an image. The purpose of padding is to preserve the original size of an image when applying a convolutional filter and enable the filter to perform full convolutions on the edge pixels.**

When performing a standard convolution operation, the image shrinks by a factor equivalent to the filter size plus one. If we take an image of width and height 6, and a filter of width and height 3, the image shrinks by the following factor.

6-3+1=4

The reason for the shrinking image is that a 3×3 filter cannot slide all three of its columns over the first two horizontal pixels in the image. The same problem exists with regard to the rows and the vertical pixels.



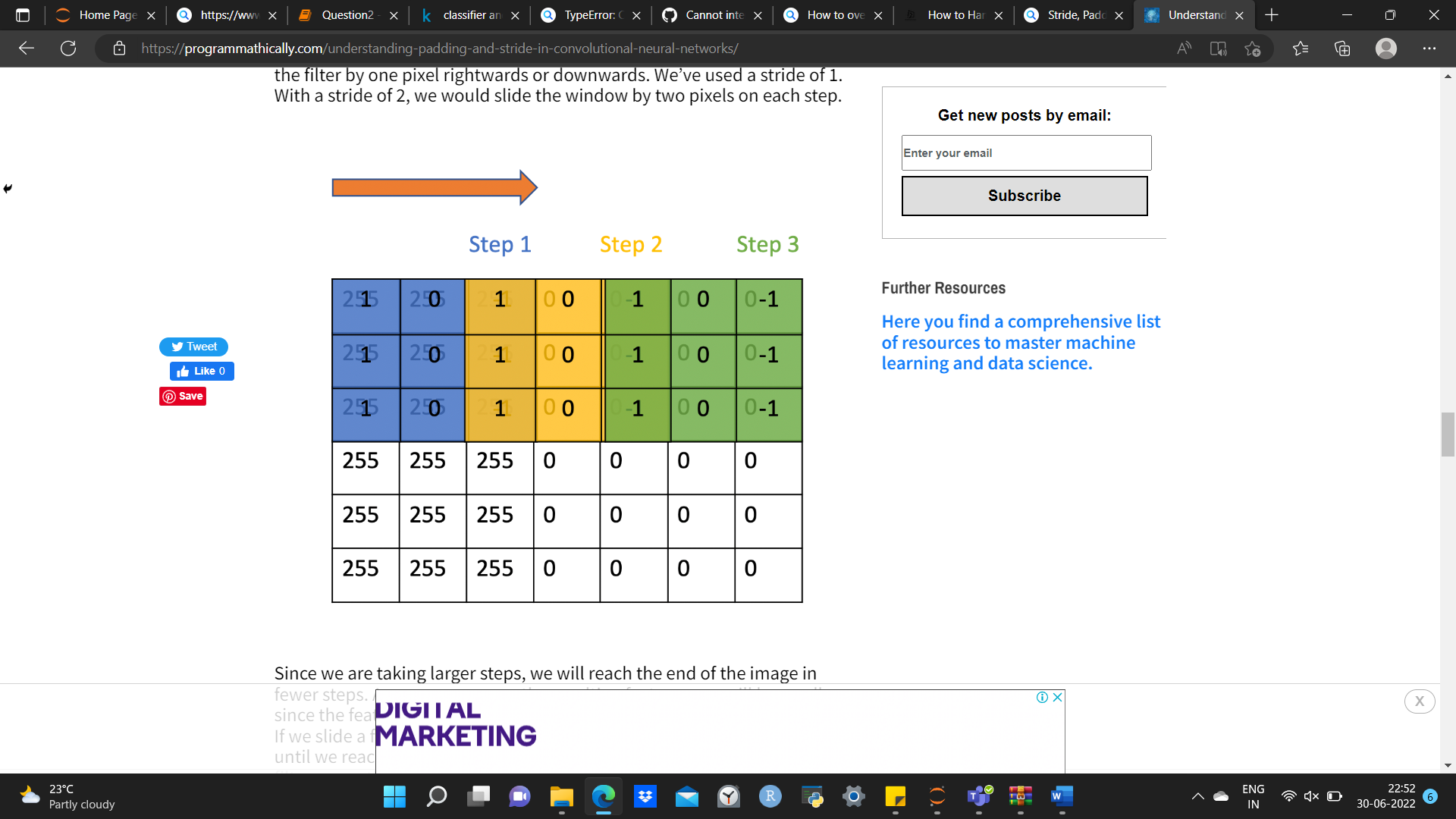
There are only 4 steps left for the filter until it reaches the end of the image, both vertically and horizontally. As a consequence, the resulting image will only have 4×4 dimensions instead of 6×6. The general formula for calculating the shrinkage of the image dimensions m x m based on the kernel size f x f, can be calculated as follows:

(m\times m) \* (f\times f) = (m-f+1)\*(m-f+1)

(m×m)∗(f×f)=(m−f+1)∗(m−f+1)

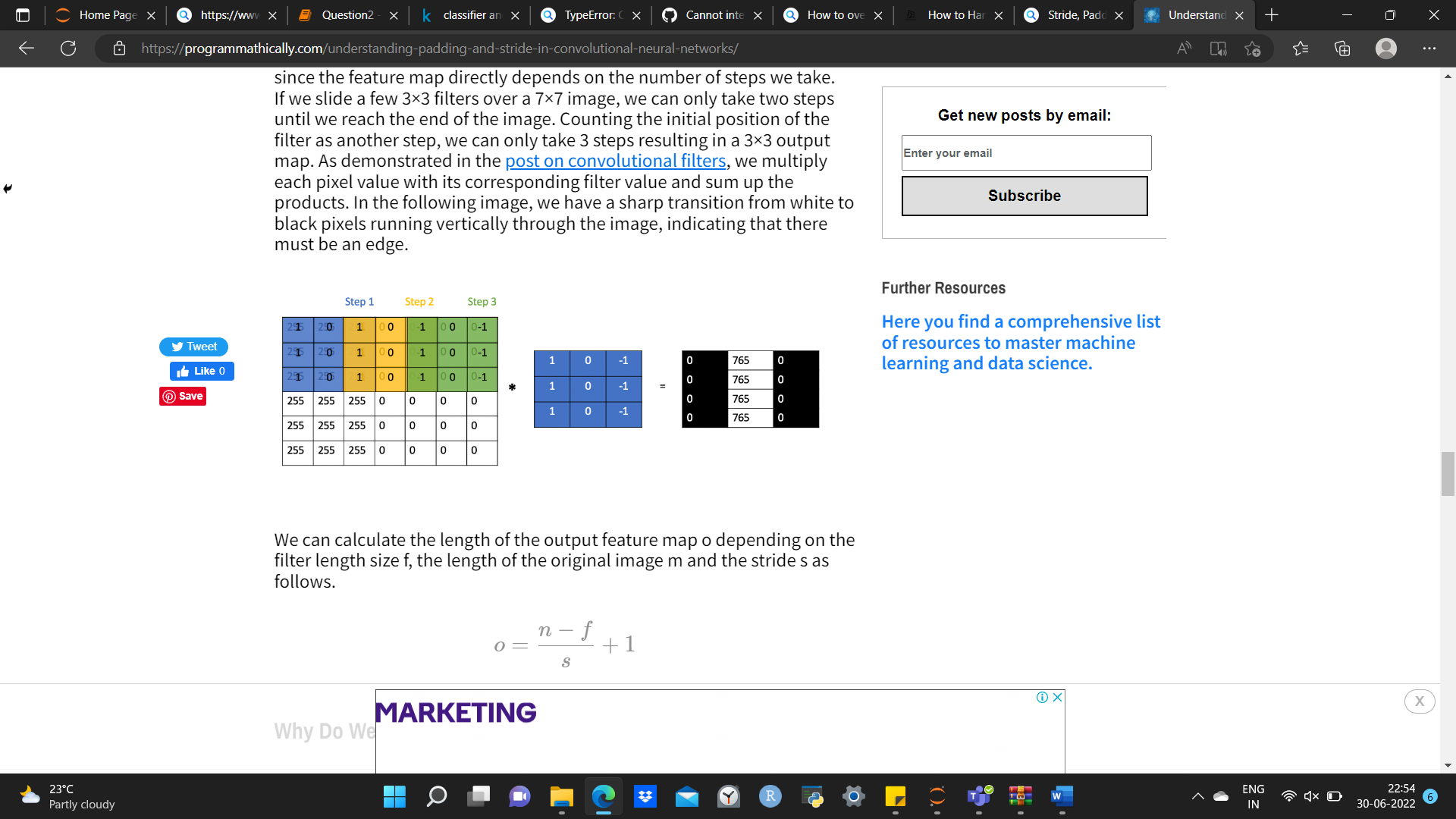
**Stride**

Stride in the context of convolutional neural networks describes the process of increasing the step size by which you slide a filter over an input image. With a stride of 2, you advance the filter by two pixels at each step**.**The stride simply describes the step size when sliding the convolutional filter over the input image. In the previous examples, we’ve always slid the filter by one pixel rightwards or downwards. We’ve used a stride of 1.  
With a stride of 2, we would slide the window by two pixels on each step.



Since we are taking larger steps, we will reach the end of the image in fewer steps. As a consequence, the resulting feature map will be smaller since the feature map directly depends on the number of steps we take.  
If we slide a few 3×3 filters over a 7×7 image, we can only take two steps until we reach the end of the image. Counting the initial position of the filter as another step, we can only take 3 steps resulting in a 3×3 output map. As demonstrated in the Since we are taking larger steps, we will reach the end of the image in fewer steps. As a consequence, the resulting feature map will be smaller since the feature map directly depends on the number of steps we take.

If we slide a few 3×3 filters over a 7×7 image, we can only take two steps until we reach the end of the image. Counting the initial position of the filter as another step, we can only take 3 steps resulting in a 3×3 output map. As demonstrated in the post on convolutional filters, we multiply each pixel value with its corresponding filter value and sum up the products. In the following image, we have a sharp transition from white to black pixels running vertically through the image, indicating that there must be an edge., we multiply each pixel value with its corresponding filter value and sum up the products. In the following image, we have a sharp transition from white to black pixels running vertically through the image, indicating that there must be an edge.



We can calculate the length of the output feature map o depending on the filter length size f, the length of the original image m and the stride s as follows.

o = \frac{n-f}{s} + 1*o*=*sn*−*f*​+1

**Pooling**

**Pooling in convolutional neural networks is a technique for generalizing features extracted by convolutional filters and helping the network recognize features independent of their location in the image.**

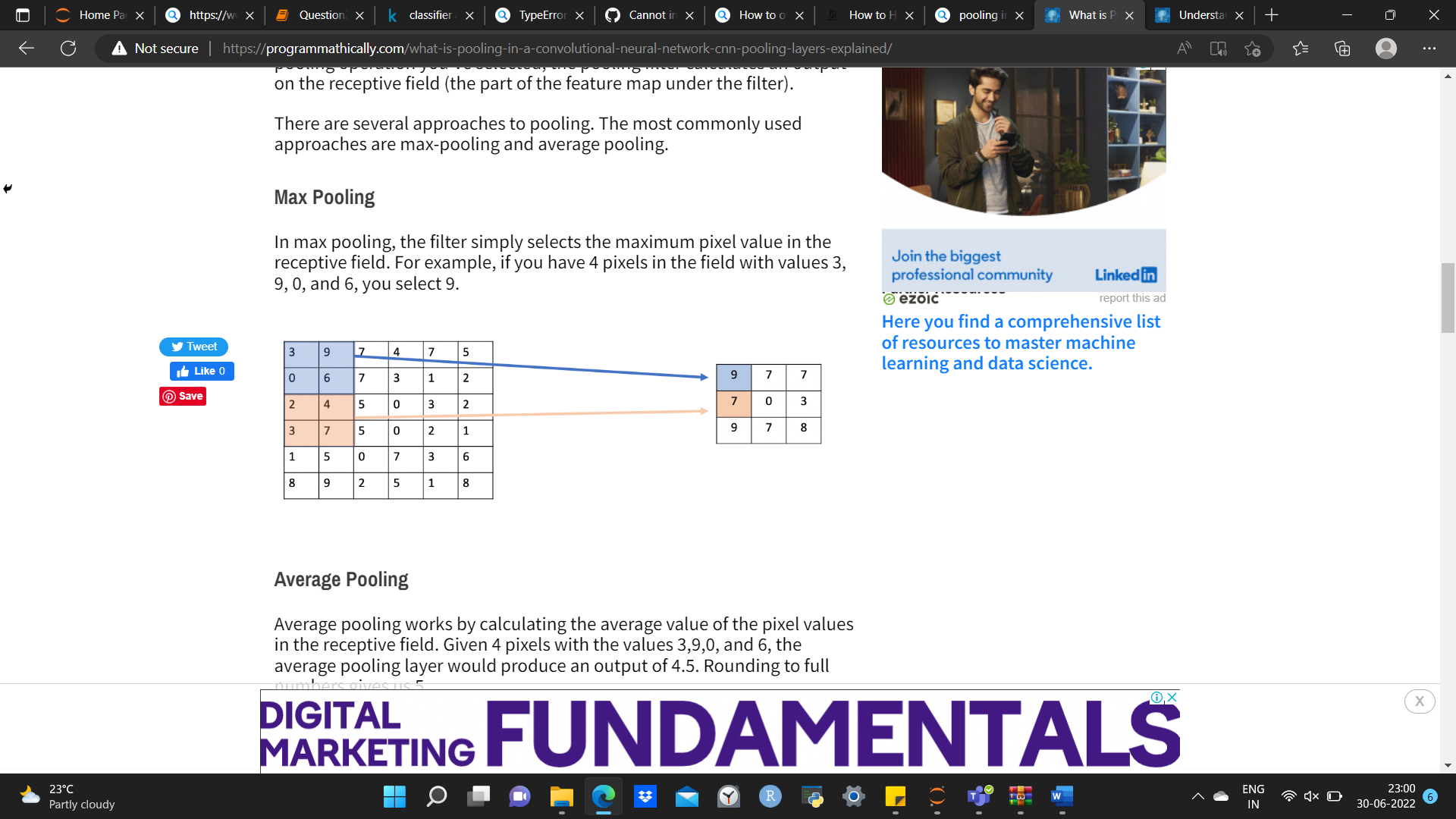
**How Does Pooling Work?**

The basic procedure of pooling is very similar to the convolution operation. You select a filter and slide it over the output feature map of the preceding convolutional layer. The most commonly used filter size is 2×2 and it is slid over the input using a stride of 2. Based on the type of pooling operation you’ve selected, the pooling filter calculates an output on the receptive field (the part of the feature map under the filter).

There are several approaches to pooling. The most commonly used approaches are max-pooling and average pooling.

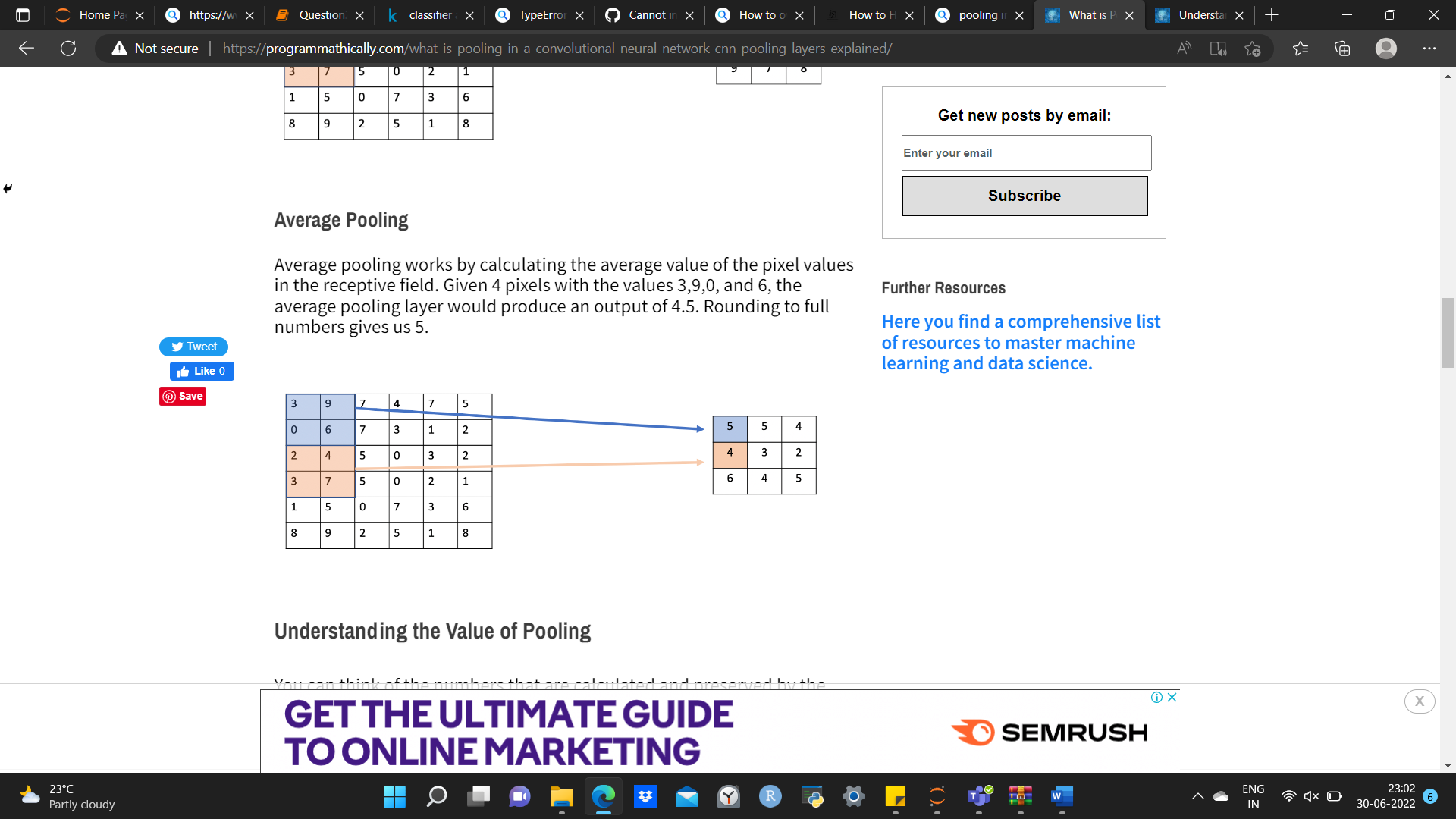
Max Pooling

In max pooling, the filter simply selects the maximum pixel value in the receptive field. For example, if you have 4 pixels in the field with values 3, 9, 0, and 6, you select 9.



Average Pooling

Average pooling works by calculating the average value of the pixel values in the receptive field. Given 4 pixels with the values 3,9,0, and 6, the average pooling layer would produce an output of 4.5. Rounding to full numbers gives us 5.



**What is overfitting? How to overcome overfitting in an ML model?**

**Overfitting:** A statistical model is said to be overfitted when the model does not make accurate predictions on testing data. When a model gets trained with so much data, it starts learning from the noise and inaccurate data entries in our data set. And when testing with test data results in High variance. Then the model does not categorize the data correctly, because of too many details and noise. The causes of overfitting are the non-parametric and non-linear methods because these types of machine learning algorithms have more freedom in building the model based on the dataset and therefore they can really build unrealistic models. A solution to avoid overfitting is using a linear algorithm if we have linear data or using the parameters like the maximal depth if we are using decision trees.

In a nutshell, Overfitting is a problem where the evaluation of machine learning algorithms on training data is different from unseen data.

**Reasons for Overfitting are as follows:**

High variance and low bias

The model is too complex

The size of the training data

**How to overcome overfitting?**

There are a number of techniques that machine learning researchers can use to mitigate overfitting. These include :

**Cross-validation**

This is done by splitting your dataset into ‘test’ data and ‘train’ data. Build the model using the ‘train’ set. The ‘test’ set is used for in-time validation. This way you know what the expected output is and you will easily be able to judge the accuracy of your model.

**Regularization**

This is a form of regression, that regularizes or shrinks the coefficient estimates towards zero. This technique discourages learning a more complex model.

**Early stopping**

When training a learner with an iterative method, you stop the training process before the final iteration. This prevents the model from memorizing the dataset.

**Pruning**

This technique applies to decision trees.

Pre-pruning: Stop ‘growing’ the tree earlier before it perfectly classifies the training set.

Post-pruning: Allows the tree to ‘grow’, perfectly classify the training set and then post prune the tree.

**Dropout**

This is a technique where randomly selected neurons are ignored during training.