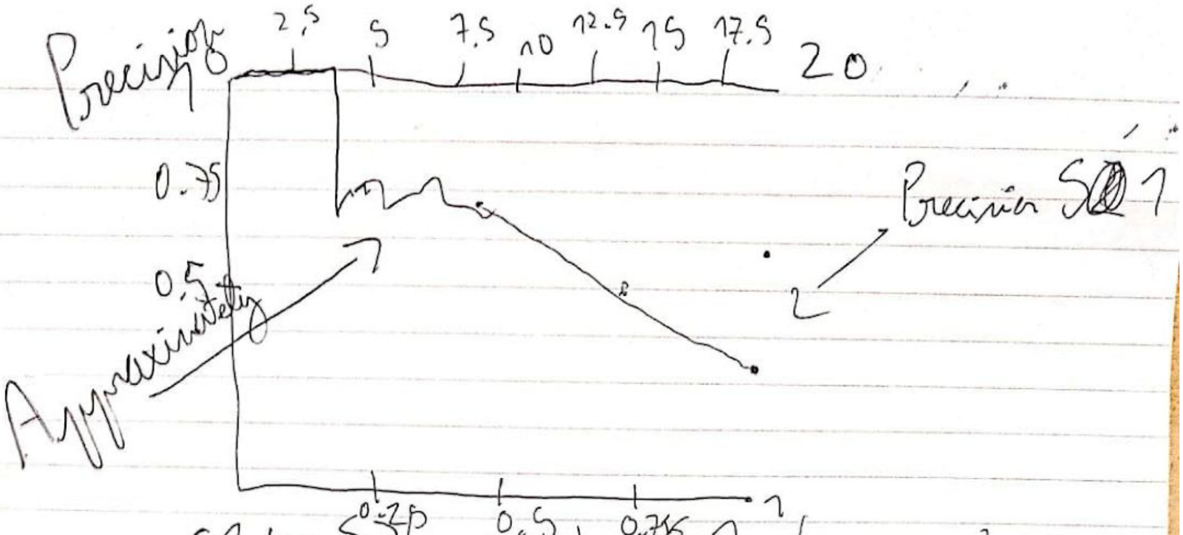
a) The sliding window approach works by having a window slide over the image at various scales (for the window), and for each window checking whether the window contains an object.
b) Image = 1000x1000, sliding window = 100x100. Stride = 5: thereby, 200 windows for the X axis, 200 for the Y-axis. E.g. for stride 5, 200x200 = 40000 windows total.
For stride 10: 1000/10 = 100 -> 100x100 = 10000 windows total
For stride 20: 1000/20 = 50 -> 50x50 = 250 windows in total.
c) Pros: less windows to consider, means less windows in total, computationally much more efficient.
Cons: we can check for less windows meaning we “skip” some sections of the images. If objects occur on a very local level (e.g. small background objects) this means that by having a skipping step of 20 pixels, we might miss a 15 by 15 pixel object.
d) By segmenting the image beforehand and then only consider different image segments as window candidates, as those are ideally likely to contain objects.

e) Precision for S1: 5/20 = 0.25
Recall for s1: 5/10 = 0.5
Precision for S2: 5/20
Recall = 5/10 = 0.5
f) To draw a precision &recall graph we record the p@k and r@k for
g) Conclusion precision recall graph:



Rank	S1 Relevance	S2 Relevance	S1 Precision	S1 Recall	S2 Precision	S2 Recall
1	✓	✗	1/1	1/10	0/1	0/10
2	✓	✗	2/2	2/10	0/2	0/10
3	✗	✗	2/3	2/10	0/3	0/10
4	✓	✗	3/4	3/10	0/4	0/10
5	✗	✗	3/5	3/10	0/5	0/10
6	✓	✗	4/6	4/10	0/6	0/10
7	✓	✓	5/7	5/10	1/7	1/10
8	✓	✗	6/8	6/10	1/8	1/10
9	✗	✓	6/9	6/10	2/9	2/10
10	✗	✗	6/10	6/10	2/10	2/10
11	✗	✓	6/11	6/10	3/11	3/10
12	✗	✓	6/12	6/10	4/12	4/10
13	✗	✓	6/13	6/10	5/13	5/10
14	✗	✗	6/14	6/10	5/14	5/10
15	✗	✓	6/15	6/10	6/15	6/10
16	✗	✗	6/16	6/10	6/16	6/10
17	✗	✗	6/17	6/10	6/17	6/10
18	✗	✗	6/18	6/10	6/18	6/10
19	✗	✗	6/19	6/10	6/19	6/10
20	✗	✗	6/20	6/10		6/10

This question took me forever and i didnt have enough time to finish it! Ideally for both systems I'd had plotted the precision recall curves by taking precision for the x, and recall for the y coordinate, and do that for both systems, but i wasnt able to do so within the allotted time.

h) a Convolutional layers work by convolving a filter over the input image
convolutional layer uses less parameters than a linear layer whilst being able to exploit the spatial structure of an image. Fully connected layers have more parameters and are essentially just a projection matrix.

i) input = 500 x 500 x 25. Filters = 100, kernels = 7x7.
Parameters = (7 x 7 x 25 + 1) * 100 = 122600 parameters (inclduing the bias term)

j) (7 x 7 x 100 + 1) * 100 = 490100 parameters.

k)
for 3x3, stride 1, output is:
[12, 12, 12, 12]
For 2x2, stride 2, output is:
[11, 1,
3, 12]