Report

Name	Sec	B.N	Code
Sara Gamal Gerges	1	20	9210455
Eman Ibraheem	1	14	9210265

Execution Time Comparison

For 6 images 1440*1880 and batch =2

Kernel	Mask Type	10×10 Mask,	3×3 Mask	4×4 Mask
Kernel1	Constant	16.789	2.6525	3.4674
Kernel1	Global	24.466	2.7942	4.1809
Kernel2	Constant	36.249	3.4750	5.0076
Kernel2	Global	42.665	4.2065	5.561
PyTorch	-	107.89	88.8259	100.277

Speedup vs PyTorch

Method	Speedup (10x10 Mask)	Speedup (3x3 Mask)
Kernel1 Constant	6.42×	33.48×
Kernel1 Global	4.41×	31.79×
Kernel2 Constant	2.97×	25.57×
Kernel2 Global	2.53×	21.11×

For 8 image 5000*5000 and batch 8

Kernel	Mask Type	10×10 Mask,	3×3 Mask	4×4 Mask
Kernel1	Constant	269.8	56.4	74.103
Kernel1	Global	348.4	59.54	90.55
Kernel2	Constant	385.6	65.92	70.88
Kernel2	Global	442.76	78.7	88.6
PyTorch	-	87.04	135.1389	140.9

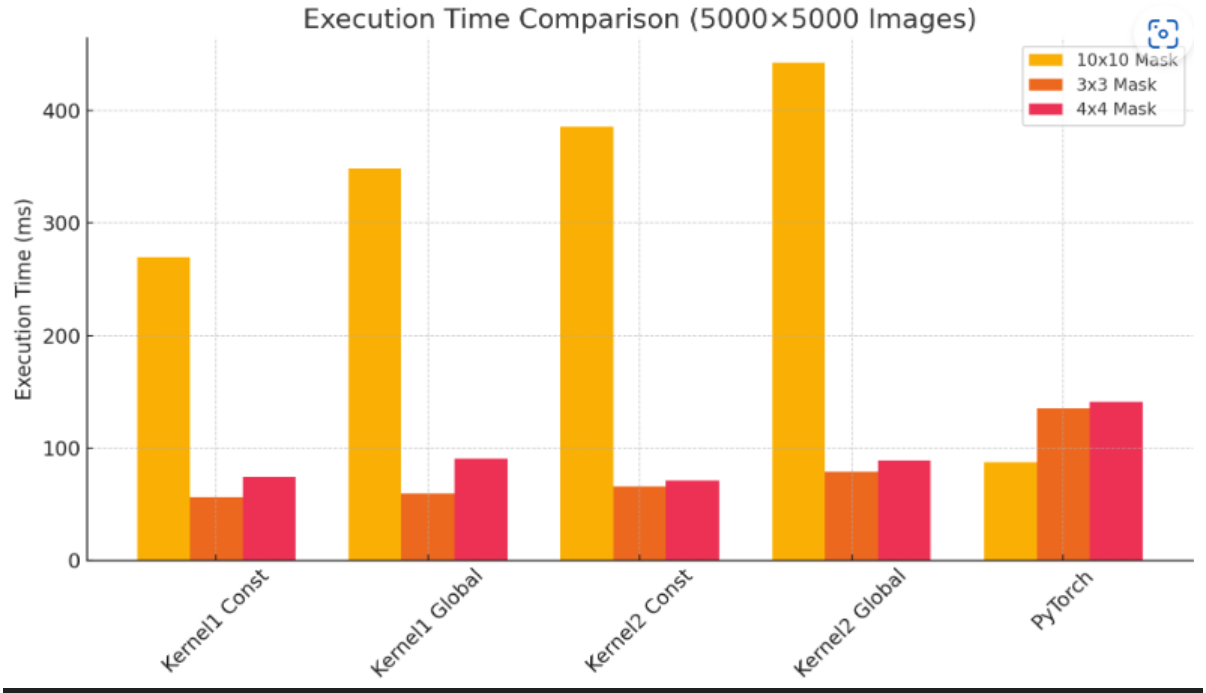
Notes and conclusions

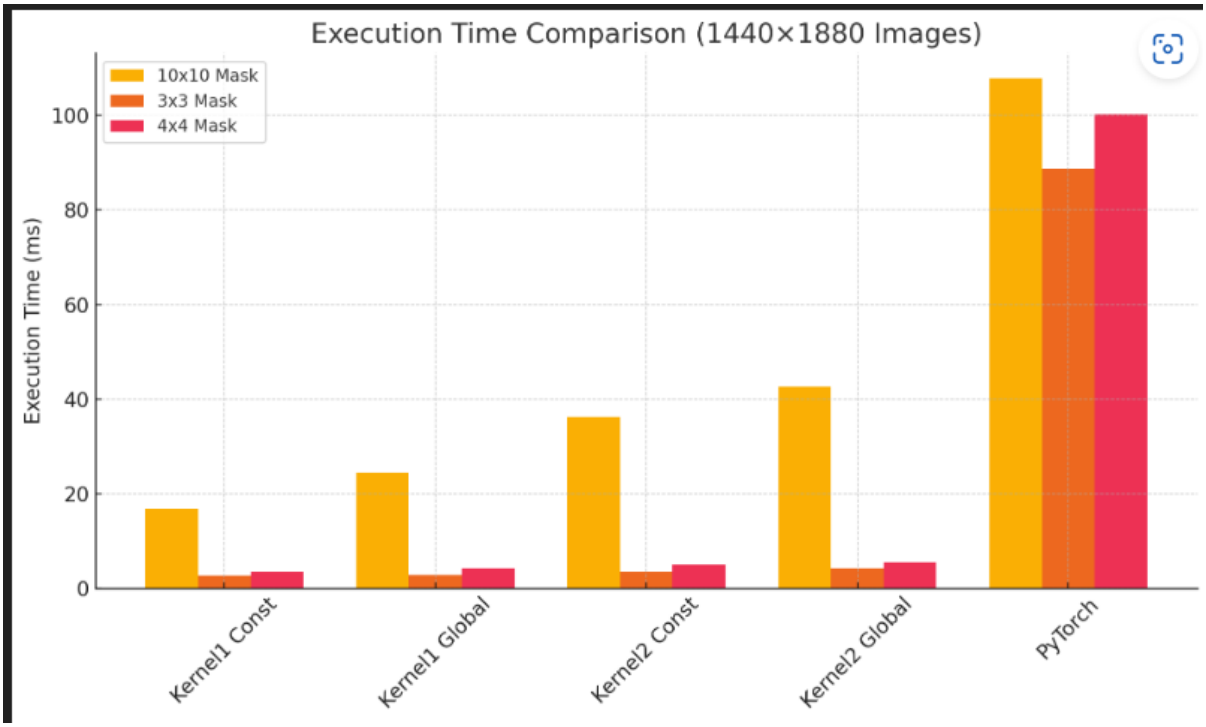
- **Using constant memory** results in **significantly faster execution** compared to using global memory, especially if mask size is bigger because the overhead of accessing global memory becomes more significant for larger mask.

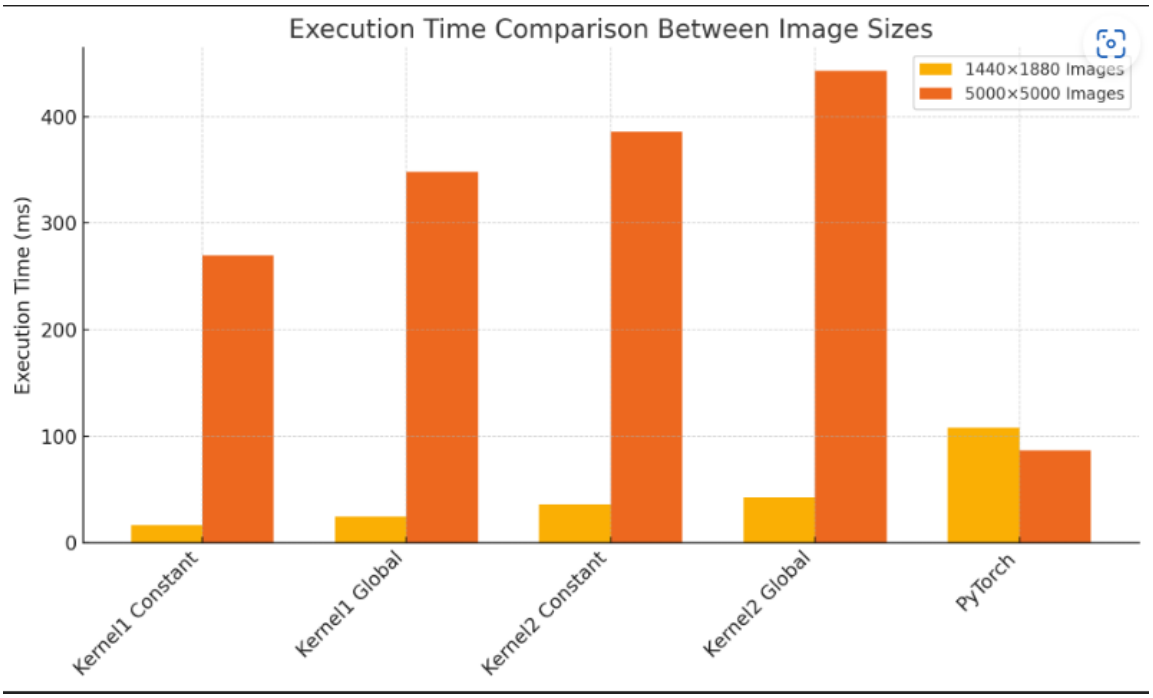
- For **smaller images**, PyTorch is slower compared to all custom kernels.
- For **larger images**, PyTorch sometimes outperforms custom kernels with certain mask sizes (notably 10×10 masks).
- Increasing the mask size increases the time for all kernels.

- Kernel1 (No tiling) is sometimes faster than Kernel2 (Input Tiling) due to:
    - Overhead of managing tiles.
    - Caching benefits in Kernel1.
    - Tiling overhead is not worth it unless memory bandwidth becomes a bottleneck.
- Image Size Effects:
    - Moderate images (1440x1880): Kernel1 performs better.
    - Large images (5000x5000): Kernel2 shows advantages due to better handling of memory access patterns.

Graphical Comparison






Execution Time Comparison Between Image Sizes
1440×1880 Images
5000×5000 Images
Execution Time (ms)
0
100
200
300
400
Kernel1 Constant
Kernel1 Global
Kernel2 Constant
Kernel2 Global
PyTorch