

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× |

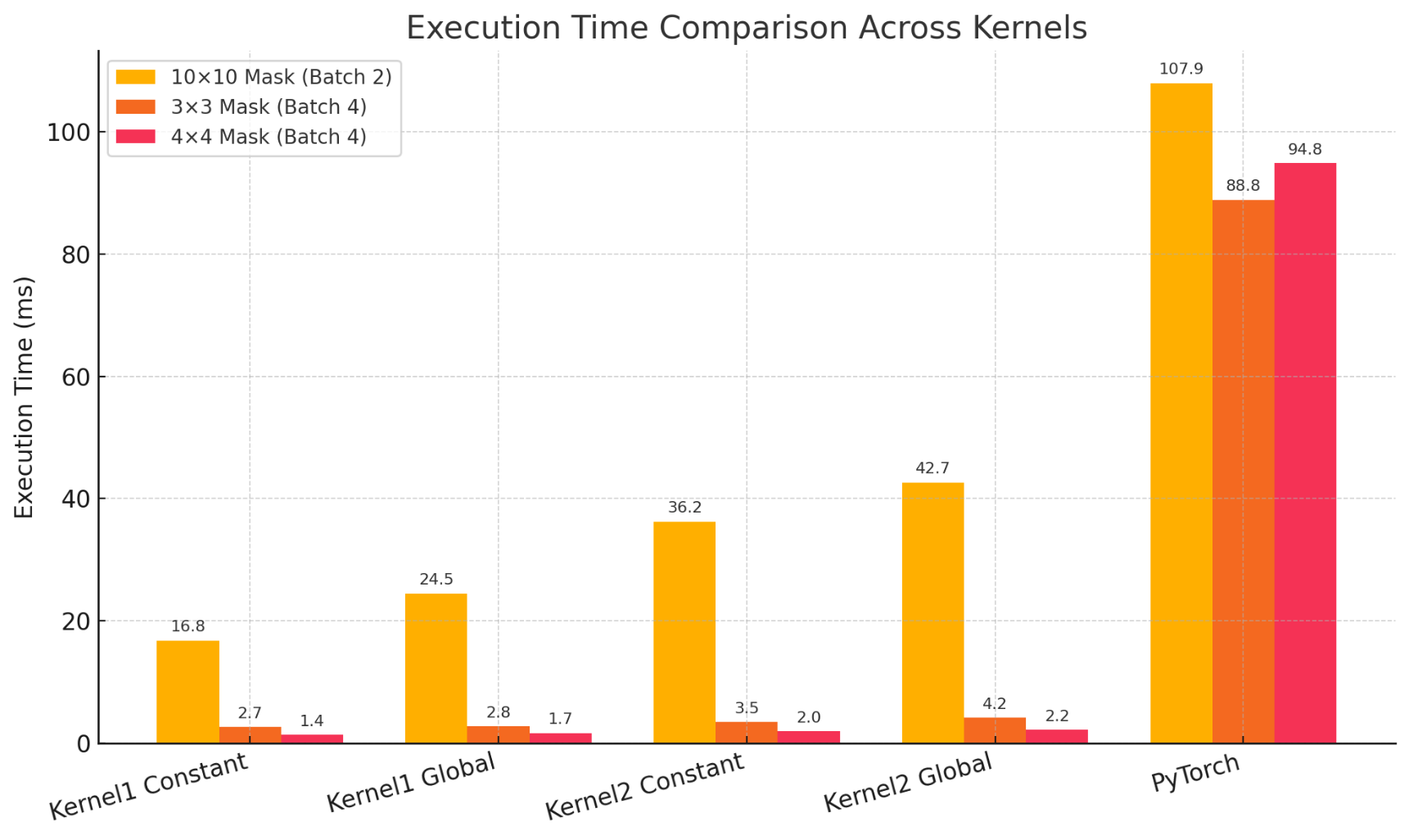
For 8 image 5000\*5000 and batch 8

| **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.
* **PyTorch’s built-in convolution is much slower** than custom kernels.
* **Increasing the mask size increases the time for all kernels and pytroch.**

## Why kernel 1(no tilling) sometimes faster than kernel 2(input tilling)?

* Overhead of Managing Tiles
* Caching in kernel 1 is making it faster
* Tiling is Worth It Only When Global Memory is a Bottleneck, but we are using processing **moderate-sized images** (like 1440x1080) with **small batches (2 or 4)**, you're **not saturating memory bandwidth**.
* When bigger images was used kernel 2 was sometimes better than kernel 1