A Fast Implementation of Max-Pooling

Yanhua Huang

Oct 2019

For a $S \times S$ feature map with filter $K * K$ and stride 1, the time complexity of the Brute Force Algorithm is $\mathcal{O}(S^2K^2)$. Max-heap (or priority queue) provides a $\mathcal{O}(S^2(K + \log K))$ solution, while the monotone priority queue gives a better one, just $\mathcal{O}(S^2)$.

The monotone priority queue maintains a monotonic sequence in a double-ended queue. Given a fixed size monotone priority queue, before enqueue the new value a, it needs to remove values in both sides that are less than a or not in the current window. For one dimensional situation, the time complexity is $\mathcal{O}(S)$ because the number of operations of each item is 2.

Let's return to the $\mathcal{O}(S^2)$ solution of max-pooling. First, we can get a matrix A where A_{ij} is the maximum value of the raw window with the tail index ij. Then apply the same process to A in the column direction. The time and space complexities of both steps above are $\mathcal{O}(S^2)$. Actually, the monotone priority queue reduces the time complexity but increase the space complexity, so it is inefficient for memory expensive computation.