那我幫你整理一份 **O(1)/Row Sliding Window Counter vs. CMS Sliding Window** 的對照表，方便你直接用在論文裡。

**Table — Comparison of O(1)/Row Sliding Window Counter and CMS Sliding Window**

| **Feature / Aspect** | **O(1)/Row Sliding Window Counter** | **CMS (Count-Min Sketch) Sliding Window** |
| --- | --- | --- |
| **Counting Method** | Exact count using Counter per minute bucket (srcip, dstip, pair) | Approximate count using d×w integer matrix per bucket |
| **Key Storage** | Stores actual keys in hash maps (exact values) | Stores hashed indices, values are merged and collisions cause overcount |
| **Update Complexity** | **O(1)** per row (remove expired, query current count, insert) | O(d) per key (update d hash positions) |
| **Query Complexity** | **O(1)** (direct lookup in Counter) | **O(d × B)** (B = number of buckets in window) |
| **Memory Usage** | O(K × B) where K = number of unique keys in window, B = number of buckets | O(w × d × B), fixed regardless of key count |
| **Accuracy** | 100% exact | Approximate, error depends on hash collisions |
| **Window Maintenance** | Removes oldest bucket and decrements counts in aggregate counters | Removes oldest CMS bucket without decrement (no exact removal of counts) |
| **Suitability** | Best for exact, real-time metrics in high-speed logs where keys are not astronomically large | Best for approximate counting of extremely high-cardinality keys when memory is constrained |
| **Use in Current Project** | ✅ Preferred — because we need accurate short-window counts for feature engineering | ❌ Not used — extra complexity, query slower, not necessary for current dataset |

我建議你後面畫流程圖的時候，可以直接做 **左右對照圖**：

* 左邊畫 O(1)/列計數：deque + Counter 實時更新、移除過期桶
* 右邊畫 CMS 計數：deque + CMS matrices 更新、多桶查詢累加

如果你要，我可以直接用這份表加上剛剛的流程步驟，幫你畫成 **雙流程對照圖**。  
這樣附錄就同時有「步驟」、「對照表」、「圖」，引用起來非常完整。

要我直接幫你畫這張雙流程對照圖嗎？