Une implémentation GPU de la méthode de recherche approximative FlyHash

Arthur da Cunha[1], ***Emanuele Natale***[1], *Damien Rivet*[1] *and Aurora Rossi*[1]

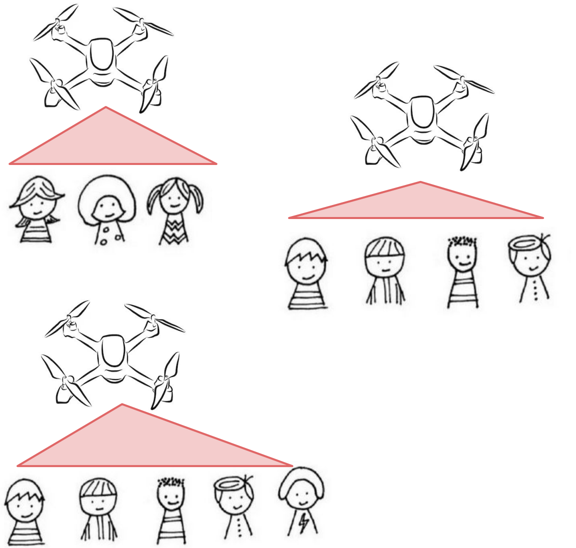
[1] COATI, I3S & INRIA d'Université Côte d'Azur

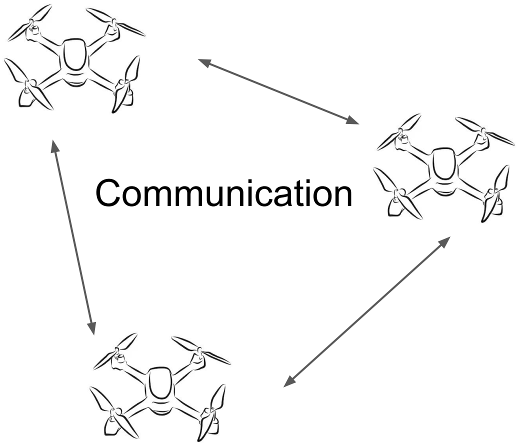
Conference on Artificial Intelligence for Defense
22 et 23 novembre 2023
Rennes - France

Application: distributed classification problem

(Ram and Sinha, 2022)

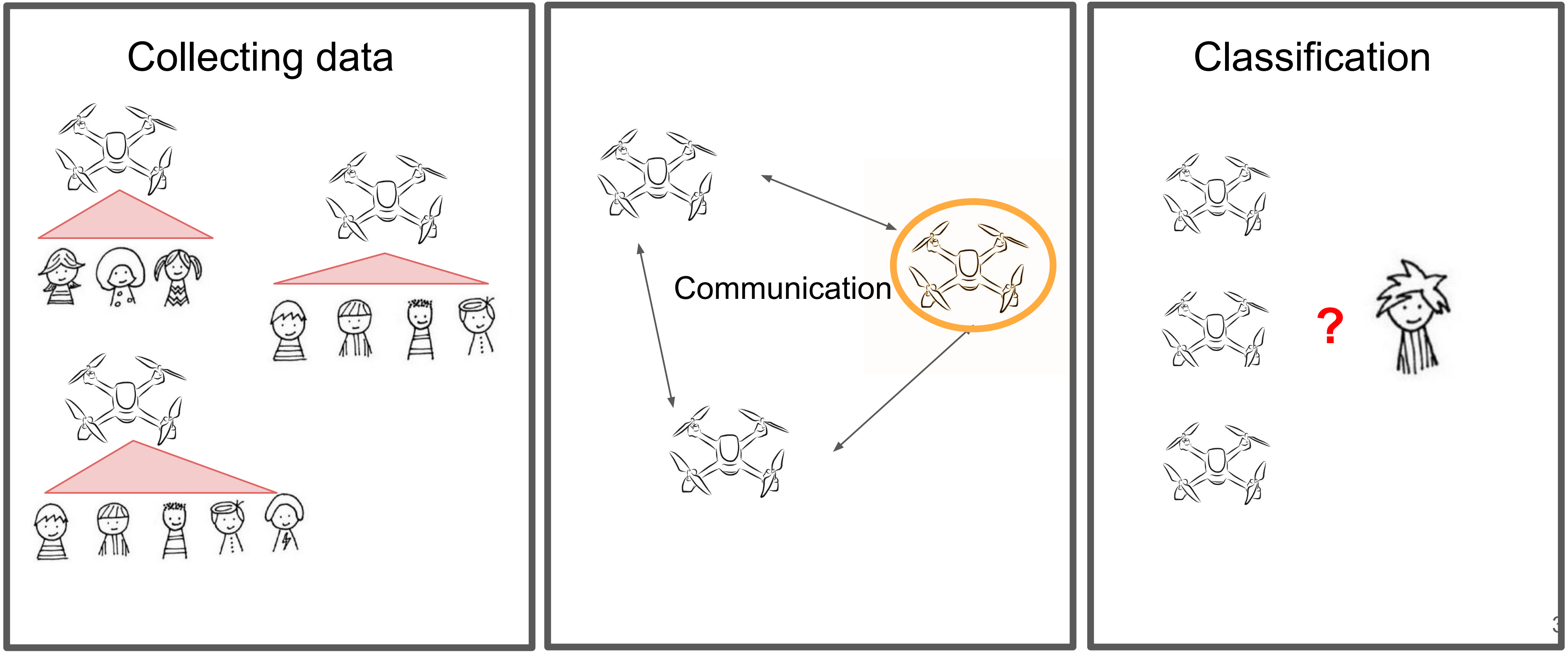
Collecting data



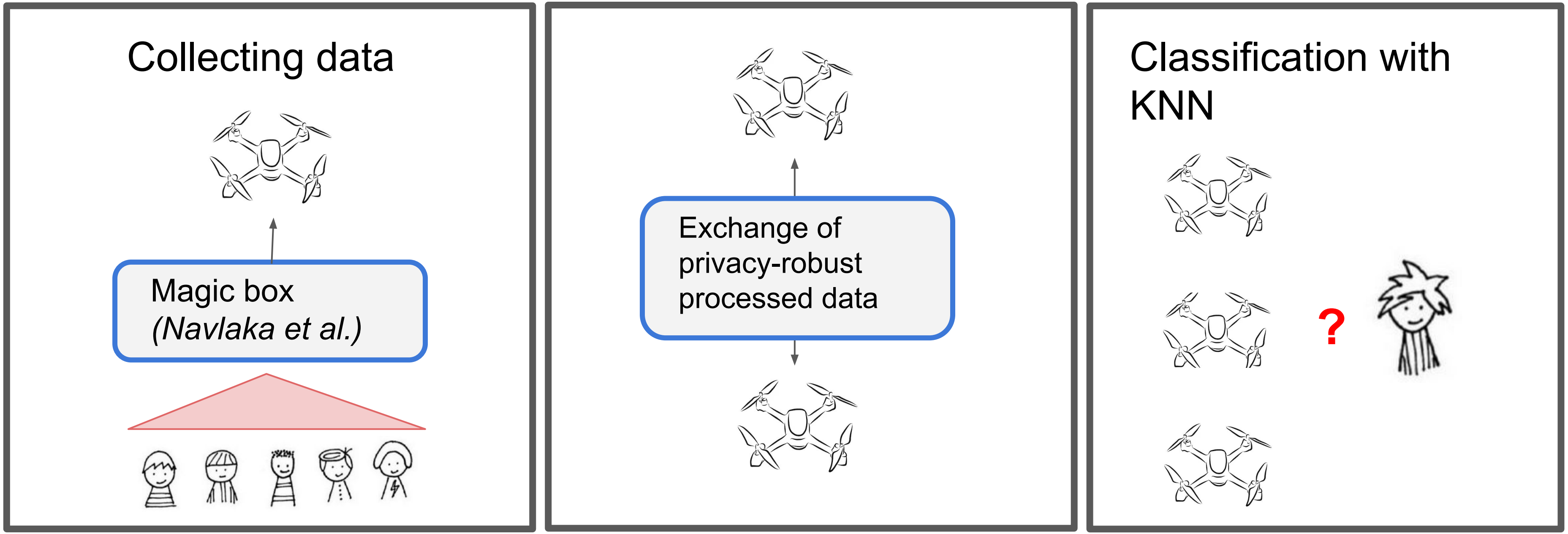


Classification

What happens if the data is sensitive and one of the agents is an eavesdropper?



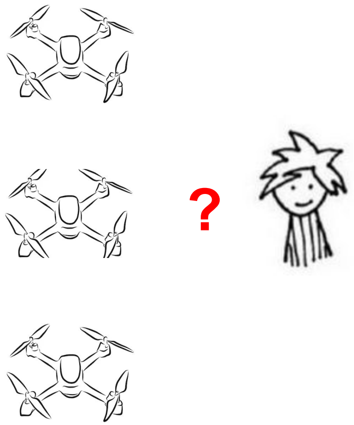
Solution



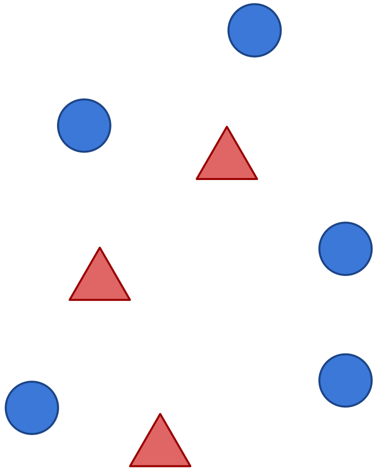
OUR CONTRIBUTION: make the magic box more efficient

Preliminaries

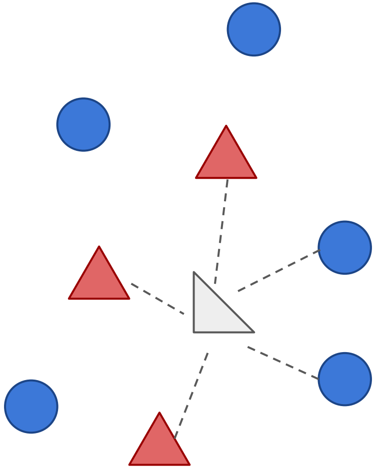
Classification with KNN



Labeled data

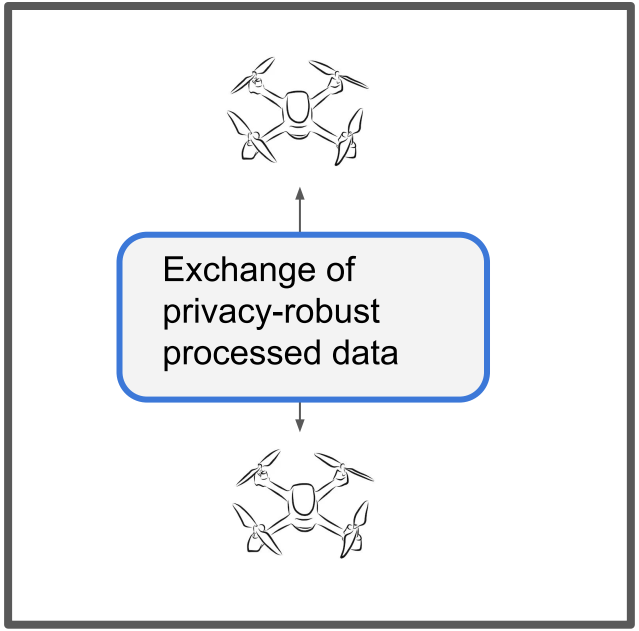


New point to classify



Majority of neighbors

Preliminaries



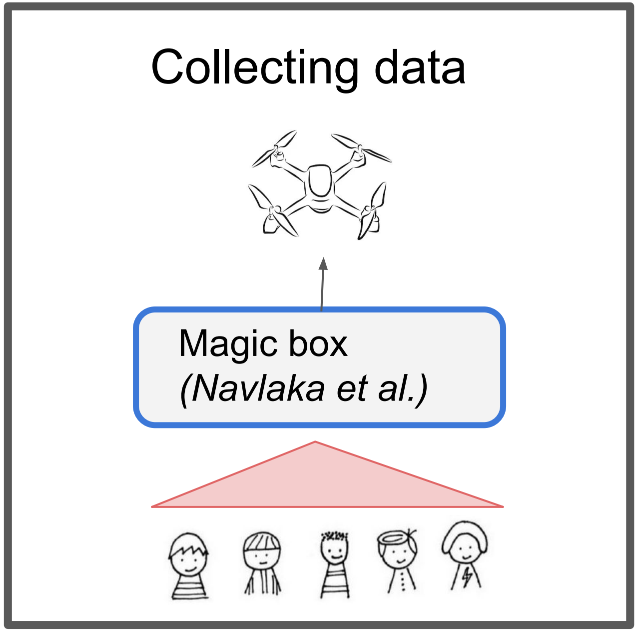
Differential privacy techniques to prevent extraction of information

Let ε be a positive real number and A be a randomized algorithm that takes as input a dataset,
then A is said to provide **ε-differential privacy**, if for all dataset D_1 and D_2 that differ on a single element and all S subset of the image of A:

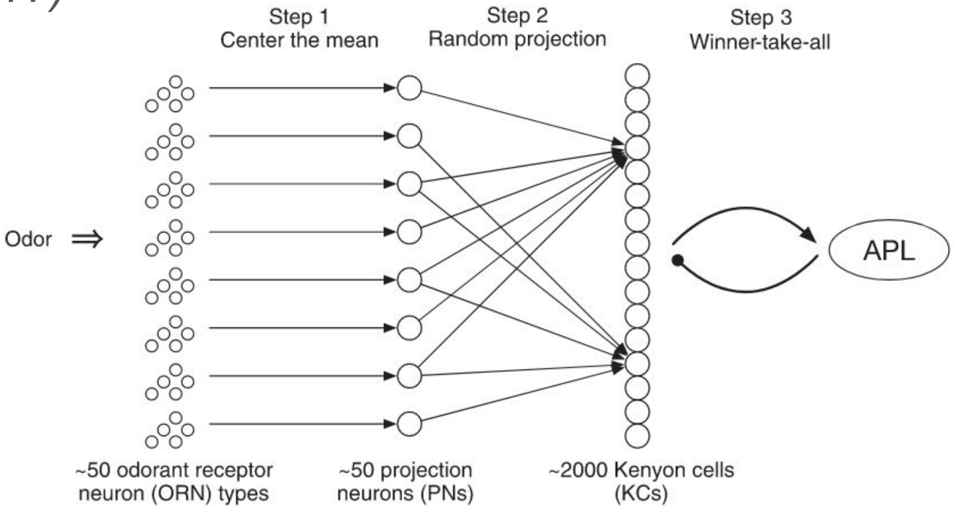
$$\Pr[A(D_1) \in S] \,/\, \Pr[A(D_2) \in S] \leq \exp(\varepsilon)$$

where the probability is taken over the randomness used by the algorithm

The magic box: FlyHash and FlyBloomFilter

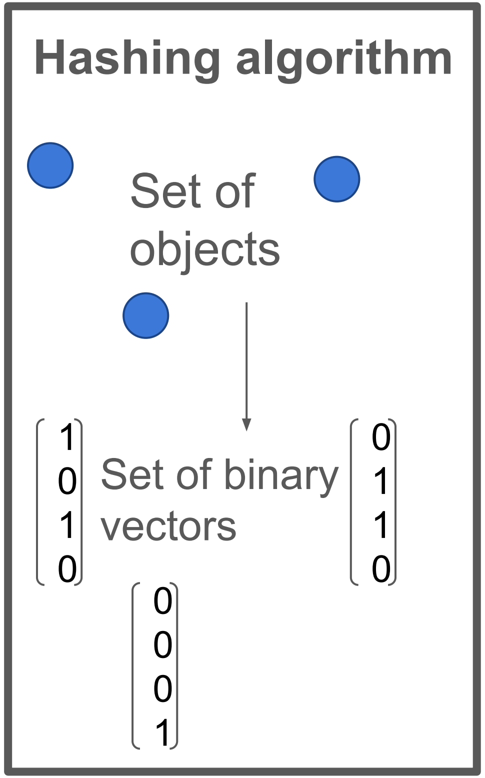


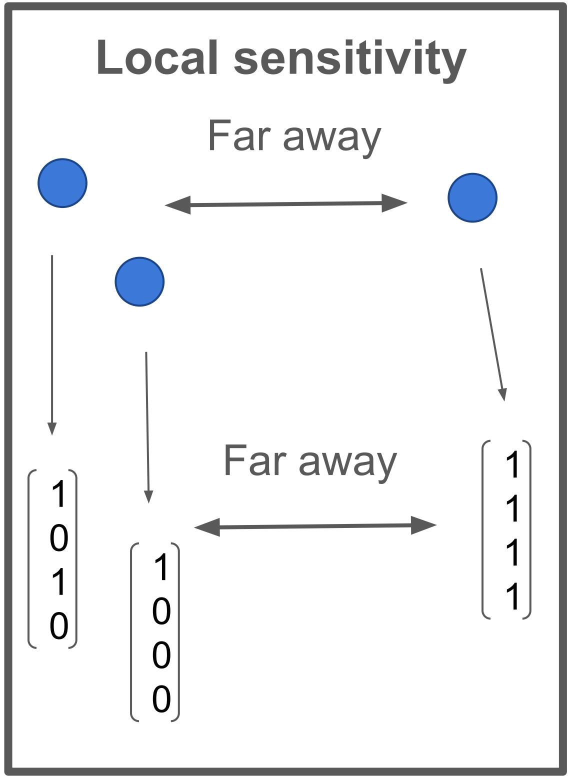
FlyHash: bio-inspired hashing algorithm *(Dasgupta et al., 2017)*



FlyBoomFilter: FlyHash-based data structure for novelty detection (Bloom filter) *(Dasgupta et al., 2018)*

FlyHash





WTA = binary vector of k-greatest entries

Pseudocode

Algorithm 1 FlyHash

Input: $X \in \mathbb{R}^d$, $M \in \{S \in \{0,1\}^{N \times d}$: each row of S contains s ones$\}$, $k \in [1, N]$
Output: $X \in \{0,1\}^N$
  $A = M \times X$
  return WTA(A, k)

FlyBloomFilter *(Dasgupta et al. 2018)*

Traditional Bloom Filter

Hash( ) = [1, 3]
Hash( ) = [2, 4]

Hash( ) = [1, 5]

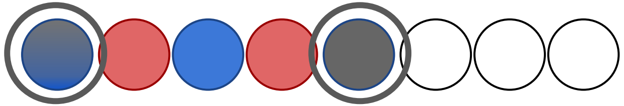
[0 0 0 0 1 1 1 1] bits

Novelty = **NO**

FlyBloomFilter

Hash( ) = [1, 3]
Hash( ) = [2, 4]

Hash( ) = [1, 5]

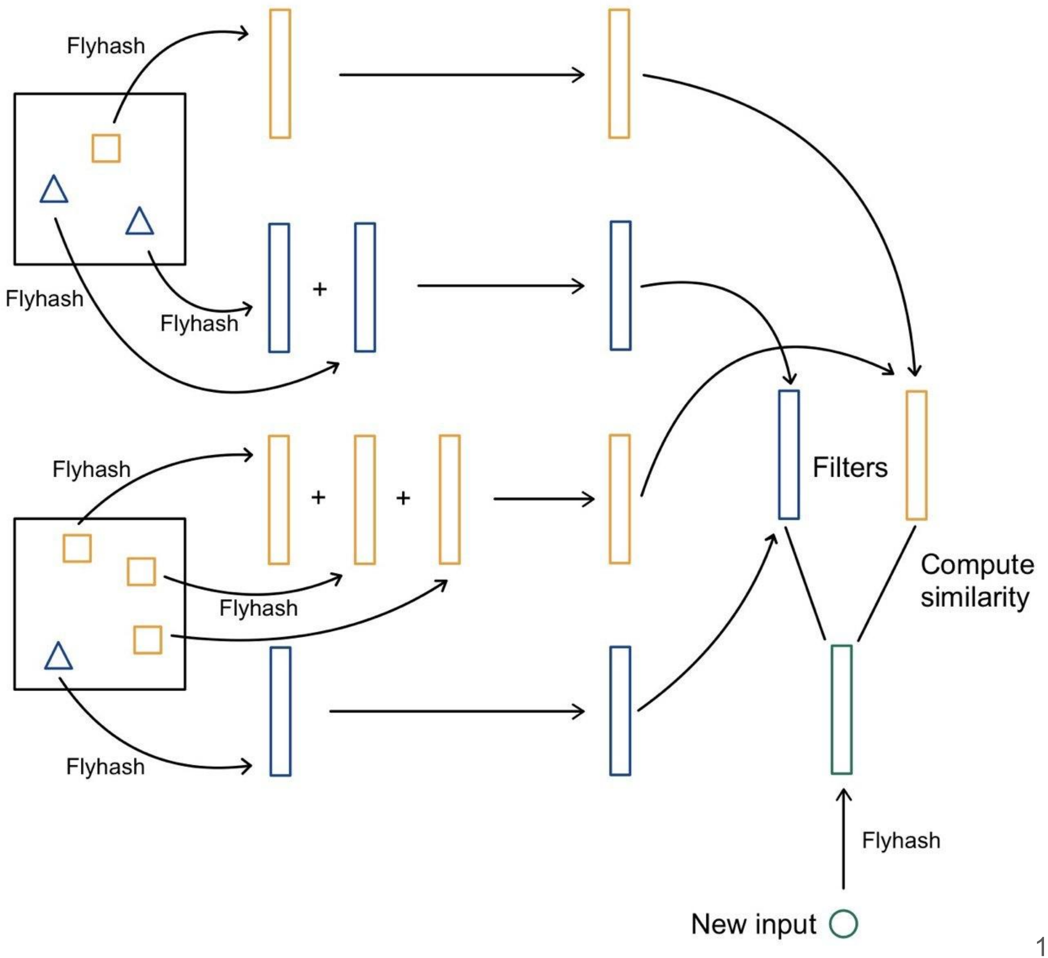


Kenyon cells

Novelty = **0.5**

The big picture

FlyNN scheme to perform approximate k-Nearest Neighbors classification using FlyHash



Our contribution: parallel version of FlyHash

Algorithm 1 FlyHash

Input: $X \in \mathbb{R}^d$, $M \in \{S \in \{0,1\}^{N \times d}$: each row of S contains s ones$\}$, $k \in [1, N]$

Output: $X \in \{0,1\}^N$

$A = M \times X$

return WTA(A, k)

Projection is GPU friendly

Is **Winner Takes All** GPU friendly?

Branchless k-Winner Takes All

Algorithm 2 Winner-take-all (WTA). Functions preceded or followed by a dot (Julia's broadcasting operator) are applied element-wise.

Input: $X \in \mathbb{R}^N$, $k \in [1, N]$
Output: $X \in \{0, 1\}^N$

$lb = \texttt{minimum}(X,\ \text{dims} = 1). - \varepsilon$
$ub = \texttt{maximum}(X,\ \text{dims} = 1). + \varepsilon$
$mid = (lb. + ub)./2$
for _ in 1 : 64 **do**
  $tot = \texttt{count}(X. > mid,\ \text{dims} = 1)$
  $lb = \textbf{ifelse}.(tot. > k, mid, lb)$
  $ub = \textbf{ifelse}.(tot. < k, mid, ub)$
  $mid = (lb. + ub)./2$
end for
return $X. > mid$

Conclusions

We provided a **parallel version of the FlyBloomFilter** which allows to run neuromorphic classification efficiently on edge devices, e.g. drone swarms

Future directions

- empirical validation: test the algorithm on the field
- improve algorithm: there is no solid theoretical guarantee that there are not better ways to implement a distributed KNN

THANK YOU!