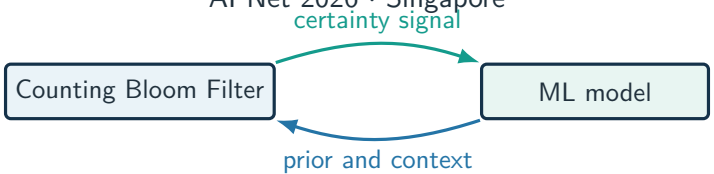
Learning Filters with Certainty

Bloom filters can help ML—and ML can help Bloom filters

Yuval Banoun Daniel Sadoc Menasché Ori Rottenstreich

Technion · UFRJ

APNet 2026 · Singapore



The central idea is a two-way interaction

Filters help learning

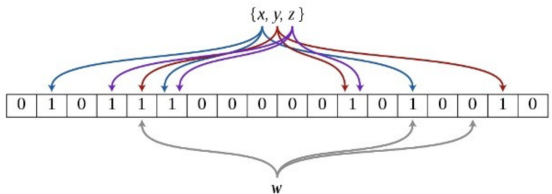
- Reject obvious non-members before inference.
- Provide a safe fallback for model false negatives.
- Expose counter values as a certainty signal.

Learning helps filters

- Provide a query-specific membership prior.
- Combine item features with counter evidence.
- Enable early decisions and lower query cost.

The roles are complementary: **CBFs contribute structural evidence; ML contributes context and priors.**

A Bloom filter compresses membership into one bit



- Compact set representation using k hash functions.
- A zero bit certifies **negative**.
- All-one positions imply **positive**—but may be a collision.

The limitation

Every positive answer looks equally convincing.

Source: Bloom (1970); figure adapted from the authors' APNet deck.

Counting Bloom filters turn collisions into evidence

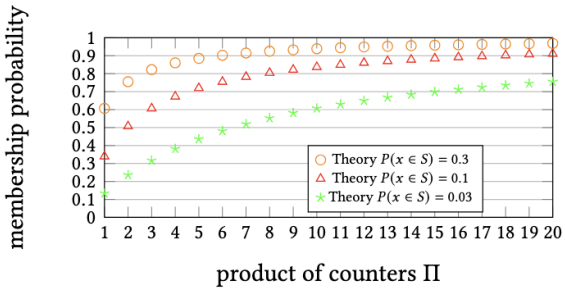
- Replace each bit by a counter.
- Insertions increment; deletions decrement.
- For query x, observe $V(x) = (c_1, \ldots, c_k)$.
- Summarize the evidence by $\Pi(x) = \prod_{j=1}^{k} c_j$.
- Larger counter products can indicate stronger support for membership.

Posterior membership probability

$$\Pr(x \in S \mid V(x); p) = \frac{m^k \Pi(x) p}{m^k \Pi(x) p + (nk)^k (1-p)}.$$

p: prior membership probability; m: counters; $n = |S|$.

The counters matter most when membership is rare



Example

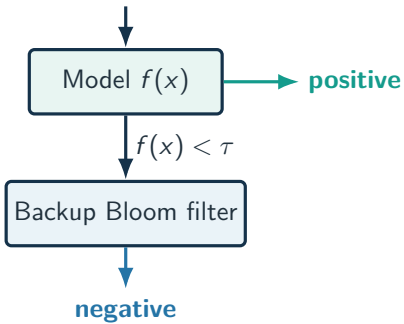
$n = 1000,\ m = 5000,\ k = 3.$

- With $p = 0.03$, increasing Π from 1 to 8 raises the posterior by about $4.1\times$.
- With $p = 0.30$, the same increase is about $1.5\times$.

Low-score queries are precisely where counter evidence is most useful.

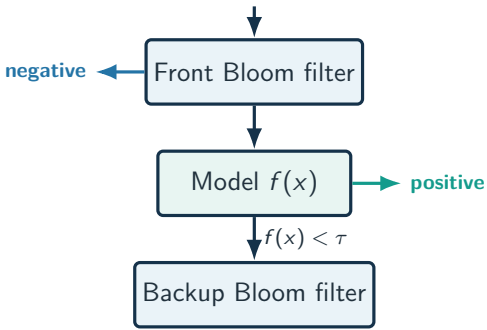
Existing learned filters already combine two safeguards

Learned Bloom filter



The filter protects members that the model scores below τ.

Sandwiched learned Bloom filter



The front filter removes many non-members before model inference.

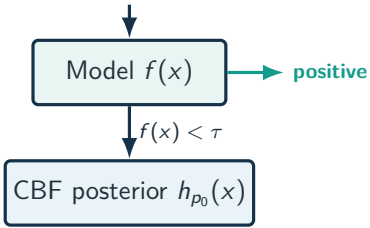
Our four models progressively tighten the feedback loop

Architecture	Front stage	ML role	CBF role
1. Learned CBF	None	Early positive decision	Posterior using population prior p_0
2. Asymmetric	Standard BF	Learned prior $p_1 = f(x)$	Final posterior decision
3. Early decision	Standard BF	Early positive or learned prior	Certainty for low-score cases
4. Symmetric MAP	Same CBF	Joint score $p_2 = f(x, V(x))$	Screening, features, and posterior

From Model 1 to Model 4, counter information moves from a **fallback test** to an explicit **input to learning**.

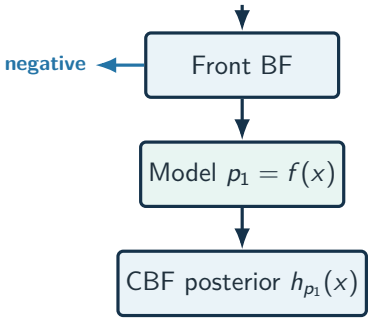
Models 1 and 2: certainty upgrades the final decision

Model 1: learned CBF



- Simplest drop-in replacement.
- No front filter; potentially higher FPR.

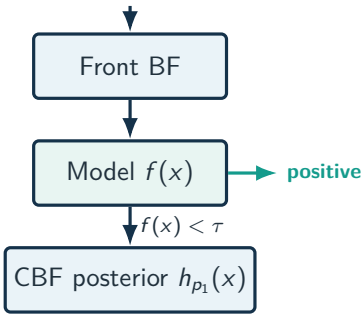
Model 2: asymmetric sandwich



- Front BF suppresses non-member traffic.
- ML score becomes the CBF prior.

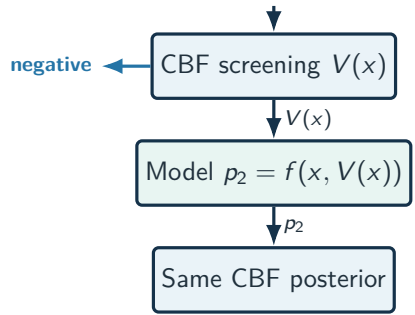
Models 3 and 4: the interaction becomes richer

Model 3: early decision



- Skips the CBF for confident positives.
- Uses counters for the low-prior region.

Model 4: symmetric MAP



- Reuses one CBF, reducing memory.
- Most expressive—but stages are dependent.

The analysis makes the sources of false positives explicit

Model 1

$$\mathrm{FPR}_1 = \underbrace{\varepsilon_f}_{\text{model FP}} + (1-\varepsilon_f)\underbrace{\Pr(h_{p_0}=1 \mid x \notin S, f<\tau)}_{\text{CBF posterior FP}}.$$

Model 2

$$\mathrm{FPR}_2 = \underbrace{\varepsilon_B}_{\text{front-filter FP}} \Pr(h_{p_1}=1 \mid x \notin S, B_1=1).$$

False-positive decompositions derived in the paper.

- A front filter multiplies down the surviving non-member mass.
- A learned prior changes which positive counter patterns are accepted.
- For Model 4, dependence must be modeled explicitly: the same CBF screens and informs the model.

No architecture dominates on every resource

	Model 1	Model 2	Model 3	Model 4
Front prefilter	–	BF	BF	same CBF
Filter structures	1	2	2	1
Early ML positive	yes	no	yes	yes
ML uses counters	no	no	no	yes
Learned score is a prior	no	yes	yes	yes
Best fit	simplest	low FPR	low latency	tight memory
Main caveat	no prefilter	two filters	two filters	dependence

The design choice is a joint decision over accuracy, latency, memory, and coupling.

Dynamic sets expose the next research questions

Operational mismatch

The CBF can absorb insertions and deletions online, but the learned model becomes stale until retraining.

Practical direction: update the CBF continuously; retrain the oracle periodically or after detected degradation.

Open design questions

- How should memory be split across filters and models?
- Should τ adapt to downstream evidence and costs?
- When should the learned oracle be retrained?
- Which other sketches expose useful uncertainty?

Takeaway: do not discard the filter's internal signal

1. A standard Bloom filter returns a bit; a CBF can return a **degree of certainty**.
2. The CBF can help ML through **screening, safe fallback, and counter features**.
3. ML can help the CBF through **query-specific priors, context, and early decisions**.
4. The four architectures expose a spectrum from **loose composition** to **joint inference**.

The useful object is not only the membership answer—it is the **information produced on the way to that answer**.