Title: SE-kNN: A Hybrid Approximate-Exact Neighbor Method with Margin-Based Fallback

Name: Ghassan Saba

INTRODUCTION:

k-Nearest Neighbors (kNN) is a simple but competitive baseline for many classification problems. Its main limitation is inference cost: exact neighbor search scales poorly as data grows. Modern approximate nearest neighbor (ANN) libraries (e.g., Annoy) dramatically speed up lookups, but they introduce a tunable accuracy–latency trade-off that is not explicitly controlled at prediction time. In this project, I replicate the original kNN/Annoy methodology and contribute an uncertainty-aware hybrid called Selective-Exactness kNN (SE-kNN). SE-kNN uses a fast ANN query first and falls back to exact kNN only for uncertain queries, adding a principled, per-example decision that improves robustness while keeping inference fast.

AIM:

Replicate the baseline methodology (exact kNN and Annoy-accelerated kNN).

Contribute a significant method: SE-kNN, an adaptive, uncertainty-aware hybrid that escalates to exact search only when needed.

Evaluate across multiple datasets (Letter Recognition, Credit Card Default, Athlete Selection; Shuttle if available) with accuracy, macro-F1, average inference time, and fallback rate.

Ablate the uncertainty rule (distance-margin vs. vote-margin), and sweep key parameters (k, n_trees, search_k).

Provide a coarse-to-fine (KMeans) alternative when Annoy is unavailable.

Github Repo:

https://github.com/SabaGhassan/MLP_Final_Project.git

DESCRIPTION OF PAPER:

The selected methodology uses kNN for supervised classification and accelerates neighbor search with Annoy (Approximate Nearest Neighbors Oh Yeah). Annoy builds multiple random projection trees and retrieves candidates via n_trees and search_k parameters, trading speed for recall/accuracy. The paper/methodology demonstrates that ANN-based kNN can preserve most

of kNN's accuracy while drastically reducing query time, but it does not decide per-query when approximation is "safe."

My contribution extends this methodology by adding a per-query uncertainty test and exact fallback, turning a static pipeline into an adaptive one.

PROBLEM STATEMENT :

Exact kNN offers strong accuracy but does not scale large datasets make inference slow. ANN indices are fast but may silently lose accuracy on "ambiguous" queries. Practitioners need a tunable mechanism that (1) retains ANN's speed on easy queries, (2) recovers exact behavior on hard queries, and (3) exposes a transparent control for meeting accuracy or latency budgets.

CONTEXT OF THE PROBLEM:

Data characteristics:

Letter Recognition — 26 classes, medium-size, all numeric.

Credit Card Default – binary, mixed features, class imbalance.

Athlete Selection — tiny toy set (sanity check).

Shuttle – multi-class and imbalanced.

Operational constraints: real-time or near real-time inference favors ANN speed, but stakeholders often require predictable accuracy on difficult cases.

Research gap: existing baselines choose "always exact" or "always approximate," lacking a per-example decision rule.

SOLUTION:

Selective-Exactness kNN (SE-kNN)

Idea: Query Annoy first; compute an uncertainty score; if uncertain, fallback to exact kNN for that point only.

Uncertainty tests (ablations):

Distance-margin: ratio $r = d_2/(d_1+\epsilon)$. If $r < \tau \rightarrow$ fallback.

- Vote-margin: difference between top 2 neighbor vote counts. If gap $< \gamma \rightarrow$ fallback.

Knobs you can tune: τ or γ (confidence), k, Annoy's n_trees, search_k.

Alternative backend: If Annoy isn't available, use a coarse-to-fine KMeans bucket stage with the same fallback logic.

Metrics: accuracy, macro-F1 (for imbalance), avg inference time per sample, fallback rate (% queries escalated to exact).

Key observations from my runs (illustrative):

Letter Recognition: Annoy-only was fastest and slightly most accurate (~0.947). SE-kNN with small τ stays close in accuracy and speed; large τ increases fallbacks (slower) and toward exact accuracy.

Credit Card Default: SE-kNN with moderate τ (≈1.2) slightly outperformed both baselines (accuracy ~0.793) while keeping latency much lower than always-exact; fallback rate ~0.67 shows the hybrid actively protected hard queries.

Athlete Selection: all methods achieved 100%; SE-kNN didn't hurt speed much for small τ, confirming low overhead on easy data.

Value of the contribution: SE-kNN adds adaptive computation and a transparent accuracy-latency control, generalizing beyond any single ANN library. Even when accuracy does not increase, the method provides robust, controllable behavior that is practically valuable.

Background

kNN delivers strong accuracy but suffers at query time; ANN methods (Annoy/FAISS/HNSW) trade accuracy for speed. Prior work rarely makes a per-query decision about when approximation is safe. This motivates our Selective-Exactness kNN (ANN first; exact fallback on uncertainty).

Reference: https://doi.org/10.48550/arXiv.2004.04523

Explanation: Classic kNN majority vote of the k closest labeled points.

Dataset/Input: 4 datasets and 4 files found in the github repo

Weakness: Single metric can hide latency/efficiency; must pair with time & fallback rate.

Implement paper code:

kNN_Annoy.ipynb file found in github repo

Contribution Code:

KNN_Annoy_SEkNN_Contribution.ipynb found in github repo

Results:

3 .png files and seknn_results.csv found in github repo

Observations:

AthleteSelection (tiny, easy) All methods hit 100% accuracy.

Annoy-only is fastest; exact is slightly slower; SE-kNN is in between until τ gets large.

Table shows fallback rates of 0.20 for $\tau \in \{1.05, 1.10, 1.20\}$ and 0.60 for $\tau=1.5$ (which is why that last green dot is slower). This dataset is trivially separable at k=5 — use Annoy-only or SE-kNN with a small τ (≈1.05–1.1) just to keep the hybrid logic without paying latency.

CC_default[Default] (realistic, imbalanced) Best accuracy observed is SE-kNN at $\tau \approx 1.2$ (~0.79293), slightly higher than both Annoy-only (~0.7928) and exact (~0.7926).

That best point comes with fallback rate ≈ 0.67 and increased latency (green dot to the right).

$\tau \approx 1.1$ gives ~0.7924 with ~0.46 fallback rate and noticeably lower time than $\tau=1.2$. Here the hybrid adds value: with moderate τ you recover (and even slightly exceed) the best single-model accuracy while keeping latency much lower than "always exact."

letter-recognition (26 classes, large) Annoy-only is both fastest and slightly more accurate (0.9474) than exact (0.9400).

As τ increases, SE-kNN falls back more (fallback 0.29 → 0.84 as τ: 1.05 → 1.5) and its accuracy slides toward the exact model (0.9458 → 0.9416) while getting slower — exactly what we expect. Why can Annoy beat exact? Approximate neighbors can act like a mild regularizer on noisy local neighborhoods; it's possible for ANN to generalize a bit better on some splits. There's no rule that exact must always have the best test accuracy. Prefer Annoy-only here or SE-kNN with very small τ (≈1.05) for a guardrail for ambiguous queries without sacrificing much speed/accuracy.

Think of τ as a speed ↔ confidence knob.

Smaller τ (≈1.05): few fallbacks, speed close to Annoy-only, accuracy close to Annoy-only.

Larger τ (≥1.2): many fallbacks, accuracy drifts toward exact, latency rises.

A good default across datasets is $\tau \approx 1.1$: gets real guardrails with moderate cost.

SLA (e.g., average inference ≤ X ms), the largest τ keeps the fallback rate under a budget (e.g., ≤40–50%). That's precisely what SE-kNN tune.

Why this is a solid methodology contribution I've turned the original kNN+Annoy pipeline into an uncertainty-aware, adaptive computation method that:

Exposes a clear trade-off control (τ),

Improves robustness on tougher data (CC_default),

And never forces to pick one extreme (always exact vs. always approximate).

Conclusion and Future Direction :

This project reproduced the baseline kNN/Annoy pipeline and added a significant, practical contribution: Selective-Exactness kNN (SE-kNN)—an uncertainty-aware hybrid that queries a fast ANN index first and falls back to exact kNN only when a margin test flags the query as ambiguous. Across datasets, SE-kNN provided a transparent accuracy-latency trade-off: on easy/separable data, ANN-only was both fastest and strong in accuracy; on harder/imbalanced data, a moderate threshold (τ) improved robustness versus ANN-only while remaining much faster than always-exact. Reporting accuracy, macro-F1, average inference time, and fallback rate gave a complete, decision-oriented view of model behavior.

Learnings:

Adaptive computation works: A simple margin-based fallback turns a fixed ANN pipeline into an uncertainty-aware method that can meet accuracy or latency targets by tuning a single knob (τ or γ).

ANN can sometimes beat exact kNN: On well-separated, mid-size data (e.g., Letter Recognition), Annoy-only was both fastest and slightly more accurate than exact kNN—likely due to mild regularization of noisy neighborhoods.

Hybrid helps on tougher/imbalanced data: On Credit Card Default, SE-kNN with a moderate τ slightly improved accuracy over both baselines while keeping latency much lower than always-exact—showing the value of selective fallback.

Fallback rate is the key control signal: As τ increases, fallback rate rises, accuracy moves toward exact kNN, and latency increases. Small τ (≈1.05–1.10) often gives strong speed with minimal accuracy change.

Metrics matter: Reporting accuracy + macro-F1 + avg inference time + fallback rate gives a complete picture; macro-F1 is especially informative on imbalanced sets.

Engineering hygiene counts: Scaling features, guarding against invalid stratification (tiny classes), and robust loaders prevent brittle results.

Results Discussion :

Letter Recognition (26 classes):

Annoy-only achieved the best accuracy (≈0.947) and the lowest latency.

SE-kNN with small τ stayed close in accuracy while adding a safety net for ambiguous points; large τ increased fallbacks and drifted toward exact kNN accuracy with higher cost.

Takeaway: Prefer Annoy-only or SE-kNN with τ≈1.05–1.10 as a guardrail.

Credit Card Default (binary, imbalanced):

SE-kNN at a moderate τ (≈1.2) slightly outperformed both exact and Annoy-only in accuracy, with a fallback rate ~0.67 and moderate latency.

Takeaway: Hybridization is beneficial on harder data; τ can be tuned to a latency/accuracy budget.

Athlete Selection (tiny toy):

All methods hit 100%; Annoy-only or small-τ SE-kNN avoids unnecessary overhead.

Takeaway: Use hybrid logic with conservative τ to retain speed while keeping a safety valve.

Limitations:

Threshold tuning: Choosing τ (or γ for vote-margin) requires validation; the best τ can be dataset-dependent.

Worst-case latency: If many queries are ambiguous, the fallback rate grows and latency approaches exact kNN.

Distance metric & scaling: Results assume Euclidean space with standardized features; other metrics or poor scaling can degrade margins.

Static index: Annoy requires rebuilds for dynamic datasets; updates aren't incremental.

Memory footprint: kNN/ANN store full training vectors; large datasets may need compression/quantization.

Class imbalance sensitivity: Even with macro-F1, rare classes may need specialized handling (e.g., class-aware k or reweighting).

Future Extension :

Auto-tune τ / γ: Learn a threshold to hit a target fallback rate or target accuracy (e.g., binary search on validation or conformal risk control for calibrated guarantees).

Dynamic-k and tiered fallback: Increase k only when the margin is small; escalate through tiers (ANN → local exact in bucket → global exact).

Alternative uncertainty signals: Try vote-margin (already added), entropy of neighbor labels, or distance spread among top-k.

Backend variants: Swap Annoy for HNSW or FAISS (IVF/PQ/HNSW); keep the same selective-exactness logic and compare curves.

Coarse-to-fine at scale: Use KMeans/IVF buckets as the first stage; learn bucket sizes to balance recall vs. compute.

Budget-aware serving: Select τ per deployment to satisfy an SLA (e.g., $\leq$X ms average) while maximizing accuracy.

Metric learning: Apply LMNN or modern deep metric learning to improve neighborhood quality before ANN + SE-kNN.

Streaming & drift: Add periodic re-indexing or mini-batch updates; monitor margin distributions to detect concept drift.

Explainability: Log influential neighbors and margins per prediction; surface them for auditability and error analysis.

References:

https://doi.org/10.48550/arXiv.2004.04523