[Re] HyperFast: Instant Classification for Tabular Data

Mohamed Amine Kina
Universität Bremen
Msc. Artificial Intelligence and Intelligent Intelligence
mkina@uni-bremen.de

1 Reproducibility Summary

This report documents a replication probe of the "HyperFast" model, specifically focusing on its claims of "instant" inference and competitive performance on small tabular datasets. Using the official pre-trained model artifact, I attempted to replicate the results reported in the paper's "Mini-test" benchmark (datasets with ≤ 1000 samples).

I successfully verified the inference speed claim: HyperFast generated predictions in 1.5–6.0 seconds per dataset, significantly faster than the 12–113 seconds required to tune a Gradient Boosting baseline. I also validated the performance claim on complex, noisy data: on the Diabetes dataset, HyperFast (73.6% accuracy) outperformed a tuned LightGBM model (72.7%).

However, I encountered a significant "reproducibility gap" on simpler datasets. On Banknote Authentication and Pendigits, HyperFast failed to reproduce the near-perfect accuracy reported in the paper (e.g., scoring 85.2% vs. the reported 100%), performing worse than a standard Logistic Regression baseline. This suggests the model may be brittle or highly sensitive to the specific random downsampling splits used in the original study.

2 Methodology

To conduct a resource-efficient probe, we selected a diverse subset of four datasets from the OpenML-CC18 "Mini-test" suite. My selection covers varied task types and feature modalities: *Diabetes* and *Banknote-authentication* (numerical binary classification), *Pendigits* (numerical multiclass), and *Credit-approval* (mixed-type binary classification).

2.1 Experimental Protocol

Data Processing: I strictly adhered to the paper's "Mini-test" protocol by performing a stratified 80/20 train–test split and then downsampling the training set to a maximum of $1{,}000$ samples and 100 features. This downsampling is critical to testing the "small data" capabilities claimed by the authors.

Preprocessing: As required for the HyperFast neural network, I applied mean imputation for numerical features, mode imputation for categorical features, and standard scaling/one-hot encoding.

2.2 Baselines

Logistic Regression: Run with default `scikit-learn` settings to establish a performance floor.

LightGBM: Used as the state-of-the-art gradient boosting baseline. To ensure a fair comparison, I utilized `RandomizedSearchCV` with a fixed iteration budget designed to complete within 5 minutes per dataset, matching the time budget allocated in the paper.

Metrics: I reported *Balanced Accuracy* to verify the paper's findings, which explicitly warn against using standard accuracy for these datasets.

3 Results

The table below compares my replication results against the values reported in Table 7 of the original paper.

Table 1: Comparison of Accuracy: Original Paper vs. Replication Results

Dataset	Log. Reg.		LightGBM		HyperFast	
	Paper	**Rep.**	**Paper**	**Rep.**	**Paper**	**Rep.**
Diabetes	66.93 ± 0.0	66.93	71.28 ± 0.33	72.70	70.91 ± 0.0	**73.63**
Pendigits	92.97 ± 0.59	91.97	96.81 ± 0.48	96.28	98.50 ± 0.38	88.03
Credit Approval	84.35 ± 0.0	83.53	87.00 ± 0.08	82.88	86.44 ± 0.54	81.55
Banknote Auth.	98.69 ± 0.0	97.39	99.84 ± 0.17	99.67	100.0 ± 0.0	85.25

3.1 Analysis

- **Validation of Generalization Potential:** On the *Diabetes* dataset, my probe supports the paper's central claim regarding zero-shot generalization. HyperFast achieved a higher balanced accuracy (73.6$\%$) than the tuned LightGBM baseline (72.7$\%$) without requiring any gradient updates. This demonstrates that the meta-learning prior can, in specific instances, effectively model unseen real-world tabular data better than traditional boosting methods.
- **Inference Efficiency:** Although I did not strictly achieve sub-second inference due to platform-specific loading overhead (1.5 s–6.2 s), HyperFast remained orders of magnitude faster than the iterative tuning required for LightGBM. For *Pendigits*, HyperFast inference concluded in ~ 1.7 s, whereas the LightGBM hyperparameter search required over 113 s to converge to a comparable solution.
- **Discrepancies on Linearly Separable Tasks:** I observed significant performance degradation on datasets where simple baselines typically excel. On *Banknote Authentication* and *Pendigits*, my replication lagged behind the original paper's reported scores by -14.8% and -10.5%, respectively. Crucially, on these tasks, even the baseline Logistic Regression outperformed HyperFast in my experiments. This divergence suggests potential sensitivity to feature preprocessing or distribution shifts that the meta-model did not robustly handle in my environment.

4 What Was Easy

- **Model Availability:** The authors provided a pre-trained model artifact (`hyperfast.ckpt`) and a `pip`-installable package. This allowed me to treat the complex meta-learning framework as a "black box" standard `scikit-learn` classifier (`HyperFastClassifier`), drastically reducing the engineering barrier to entry.
- **Inference Pipeline:** Once the environment was set up, running the model was straightforward. The API consistency with `scikit-learn` meant I could plug HyperFast directly into my existing evaluation loops without writing custom inference code.

5 What Was Difficult

- **The Reproducibility Gap:** The most challenging aspect was diagnosing the root cause of performance degradation on "trivial" datasets like *Banknote Authentication*. Despite adhering to the strict downsampling protocol ($N < 1{,}000$), my results deviated significantly from the paper's reported near-perfect scores. This discrepancy highlights a lack of documentation regarding the specific random seeds or stratified splits used in the original study. In small-data regimes, the specific selection of training examples can drastically shift the decision boundary, yet the exact indices were not provided.

- **Resource-Constrained Baseline Comparison:** Reproducing the baselines required a strategic trade-off between fairness and feasibility. While the paper utilizes extensive hyperparameter search spaces, implementing full Bayesian optimization for complex baselines (such as `SAINT` or `TabPFN`) was computationally prohibitive within a single-day workflow. Consequently, we had to rely on `LightGBM` as a representative "efficiency" rival, implementing a time-constrained search to approximate the paper's experimental conditions.
- **Implicit Preprocessing Requirements:** Although the documentation mentions data imputation, the strict requirement for handling missing values in the neural network backend was understated. Unlike robust tree-based models (e.g., `XGBoost`), which handle `NaN` values natively, the HyperFast pipeline is sensitive to input integrity. Ensuring the preprocessing pipeline was perfectly aligned for `HyperFast`—without leaking information or unfairly penalizing the baselines—required trial-and-error that went beyond the provided "Quickstart" guide.