

# TURBOTEST: Learning When Less is Enough through Early Termination of Internet Speed Tests

Haarika Manda  
*UC Santa Barbara*

Manshi Sagar  
*IIT Delhi*

Yogesh  
*IIT Delhi*

Kartikay Singh  
*IIT Delhi*

Cindy Zhao  
*UC Santa Barbara*

Tarun Mangla  
*IIT Delhi*

Phillipa Gill  
*Google*

Elizabeth Belding  
*UC Santa Barbara*

Arpit Gupta  
*UC Santa Barbara*

## Abstract

Internet speed tests are indispensable for users, ISPs, and policymakers, but their static flooding-based design imposes growing costs: a single high-speed test can transfer hundreds of MB, and collectively, platforms like Ookla, M-Lab, and Fast.com generate petabytes of traffic each month. Reducing this burden requires deciding when a test can be stopped early without sacrificing accuracy. We frame this as an optimal stopping problem and show that existing heuristics—static thresholds, BBR pipe-full signals, or throughput stability rules from Fast.com and FastBTS—capture only a narrow slice of the achievable accuracy–savings trade-off. This paper introduces TURBOTEST, a systematic framework for speed test termination that sits atop existing platforms. The key idea is to decouple throughput prediction (Stage 1) from test termination (Stage 2): Stage 1 trains a regressor to estimate final throughput from partial measurements, while Stage 2 trains a classifier to decide when sufficient evidence has accumulated to stop. Leveraging richer transport-level features (RTT, retransmissions, congestion window) alongside throughput, TURBOTEST exposes a single tunable parameter  $\epsilon$  for accuracy tolerance and includes a fallback mechanism for high-variability cases. Evaluation on 1 million M-Lab NDT speed tests (2024–2025) shows that TURBOTEST achieves 1.8–4.4 $\times$  higher data savings than an approach based on BBR signals while reducing median error. These results demonstrate that adaptive ML-based termination can deliver accurate, efficient, and deployable speed tests at scale.

## 1 Introduction

Internet speed tests are critical tools for multiple stakeholders: users rely on them to verify service quality against advertised speeds, Internet Service Providers (ISPs) to plan capacity and diagnose performance issues, and policymakers to inform broadband policy and evaluate infrastructure investments. However, these measurements impose substantial costs that scale with network performance improvements. Their

static, flooding-based design requires saturating the network path for a fixed duration (often 10+ seconds), transferring hundreds of megabytes on high-speed links. This creates significant expenses for providers operating measurement infrastructure and for users on metered connections. With millions of tests conducted daily across platforms such as Ookla [1], M-Lab [24], and Netflix’s Fast.com [2], the aggregate traffic volume and infrastructure burden represent a substantial and growing economic cost. Reducing measurement overhead is therefore essential, not only for providers but also for preserving the societal value of speed test data.

Different providers have embraced distinct *speed test methodologies*. For example, Ookla employs multi-threaded tests that offer high fidelity at the cost of substantial overhead, while M-Lab uses a single-threaded design with BBR congestion control to saturate the link. Newer methodologies such as FastBTS [33] and Fast.com [2] aim to balance efficiency and accuracy by integrating convergence-based stopping rules. However, scaling new methodologies remains challenging due to legacy constraints, proprietary business models, and lack of standardization. This paper does *not* focus on developing yet another measurement methodology. Instead, it addresses the complementary problem of designing an *external termination layer* that sits atop existing methodologies: given an ongoing test, determine when it can be stopped early without sacrificing accuracy. This layered approach makes solutions deployable across diverse platforms without replacing existing test designs.

**The opportunity: early termination.** This problem can be framed as an *optimal stopping* problem. At each point in the test, the decision maker must weigh the potential accuracy gain from continuing the test against the savings from stopping early. The challenge is to design stopping policies that balance these competing objectives while remaining robust across heterogeneous and evolving access networks.

**Existing approaches.** Several heuristic approaches to test termination have been proposed for this problem. Static thresholds—e.g., M-Lab’s fixed 250 MB cap [24] or Cloudflare’s capped tests [7]—offer simplicity but ignore link het-

erogeneity. Transport-signal rules, such as BBR’s pipe-full indicator [12], have also been used to exploit congestion-control signals. Other ideas were not originally designed for external termination but could, in principle, be applied to it. For example, Netflix’s Fast.com implements a throughput stability heuristic (TSH) [2], and FastBTS introduced crucial interval sampling (CIS) [33]; while these are embedded in complete methodologies, the underlying logic could also be repurposed as external stopping rules. Each of these methods defines a stopping condition and exposes a tunable parameter that governs the trade-off between accuracy and efficiency.

**Limitations of heuristics.** Despite their utility, all such approaches share key limitations. First, they rely on fragile assumptions. For instance, BBR fails in high-speed links where pipe-full may never appear. Second, they draw only on narrow signal spaces (e.g. throughput time series), discarding richer TCP-level features such as RTT and congestion window dynamics that could provide more reliable evidence of convergence. Third, even when they terminate at the “right” time, their throughput estimation remains naïve and utilizes simple averages that yields biased results. As a result, these approaches capture only a narrow slice of the achievable accuracy–savings frontier.

**Our approach: TURBOTEST.** This paper introduces TURBOTEST, a systematic framework for early termination of Internet speed tests. The key idea is to decompose the problem into two coordinated subproblems: (i) *prediction*, where a regression model estimates the final throughput from partial measurements, and (ii) *termination*, where a classifier decides when to stop based on prediction accuracy and cost trade-offs. This decomposition avoids conflating two separate but not independent tasks: termination decision are conditioned on prediction outcomes, accurate predictions rely on termination to trigger only once sufficient evidence has accumulated. This coupling enables more aggressive yet reliable termination compared to prior heuristics.

TURBOTEST leverages machine learning (ML) to optimize prediction and termination independently, while coordinating them end-to-end. ML naturally incorporates complementary metrics and adapts to non-stationary network conditions that break heuristic assumptions. It can also exploit the vast repositories of *full* speed test results already collected by providers, turning existing infrastructure into a training resource. To ensure robustness in deployment, TURBOTEST includes a lightweight fallback mechanism: tests exhibiting high variability—where early termination would be unreliable—are allowed to run to completion, bounding worst-case error.

**Key contributions.** This paper makes three contributions:

- **Learning problem.** We cast external termination of speed tests as an *optimal stopping* problem, exposing the compound trade-off between accuracy and measurement cost and clarifying the implicit assumptions behind existing heuristics.

- **Design.** We develop TURBOTEST, a two-stage ML framework that decouples throughput prediction from termination, leverages richer transport-level features (e.g., RTT, retransmissions, congestion window) alongside throughput, and exposes a single tunable parameter  $\epsilon$  that encodes operator accuracy tolerance.
- **Evaluation.** We evaluate TURBOTEST on 1 million M-Lab speed tests spanning 2024–2025 across diverse access types. Our results demonstrate that TURBOTEST consistently improves the accuracy-savings frontier over state-of-the-art heuristics. At  $\epsilon = 15$ , for example, TURBOTEST achieves 91.7% data savings with 18.6% median error, outperforming BBR’s maximum of 84.9% savings at 35.4% error and CIS’s 89.9% data savings at 32.1% error (see §5.2). Importantly, it closes much of the gap to oracle bounds, showing that adaptive ML-based termination can approach theoretical limits while remaining deployable at scale.

The remainder of this paper is organized as follows. Section 2 reviews Internet speed test methodologies; Section 3 formalizes the external termination problem. Section 4 presents the design of TURBOTEST, detailing its two-stage framework for prediction and termination as well as implementation specifics. Section 5 evaluates TURBOTEST against heuristic baselines, exploring accuracy–savings trade-offs, adaptive parameterization, and robustness. Section 6 summarizes the related work and Section 7 concludes with future directions.

## 2 Early Termination of Speed Tests

### 2.1 Speed Tests in Practice

Internet speed tests are widely used to evaluate end-to-end performance and broadband quality. At a high level, these tests estimate bottleneck link throughput by saturating one or more TCP connections to strategically placed servers. Different providers adopt different strategies to achieve this goal. Tests employ different termination strategies to ensure that there is self-congestion at the bottleneck link: M-Lab uses a fixed 10-second duration, whereas Ookla [1] and Fast.com [2] terminate dynamically.

Despite methodological differences, all approaches consume significant network resources. This problem – worsened by increasing access speeds – creates strain on test provider infrastructure and incurs costs for users with data caps. For example, M-Lab reports that its global test infrastructure generated 12 PB of traffic in September 2024 alone, a  $23\times$  increase compared to the same period in 2023 [12].

### 2.2 The Early Termination Problem

Speed test overhead has been recognized as a problem, with new protocols being developed to balance accuracy with

data usage. For instance, FastBTS proposes a modified congestion control approach that saturates the bottleneck link faster [33]. Recent interest has shifted towards reducing overhead by terminating the test earlier, thus augmenting existing methods rather than replacing them. For example, M-Lab recently capped tests at 250 MB [23]. This reduces data usage but can underestimate throughput, particularly on high-speed networks. More broadly, early termination introduces an accuracy–efficiency tradeoff. Stopping sooner reduces bandwidth consumption and server load but risks underestimating the user’s true throughput, while continuing longer increases accuracy but inflates costs.

This leads to a new decision problem: given partial measurements collected during an ongoing test, can we stop without significantly degrading the accuracy of the reported throughput? An effective early termination solution must balance accuracy and data usage, generalizing across heterogeneous access types (e.g., cable, fiber, cellular) and dynamic network conditions. This motivates the rule-based heuristics we discuss next (§2.3) and the need for a systematic framework that can push the accuracy–savings frontier outward.

### 2.3 Existing Early Termination Approaches

Several efforts have explored rule-based heuristics for terminating speed tests. These heuristics embody simple decision rules that map partial throughput observations to a stopping decision. We group them into two broad classes: static thresholds and dynamic convergence-based rules.

**Static thresholds.** The simplest approach is to terminate after transferring a fixed amount of data, such as 10 MB, 100 MB, or 1 GB. The rationale is straightforward: larger transfers improve accuracy in estimating bottleneck throughput, while smaller transfers reduce measurement cost. However, such thresholds are oblivious to network heterogeneity: a 10 MB transfer may suffice for a 25 Mbps connection but be grossly inadequate for a 1 Gbps link. Thus static thresholds, while attractive for their simplicity, are poor at controlling error across diverse conditions due to their lack of adaptivity [12].

**BBR pipe-full termination.** A more dynamic heuristic leverages the TCP BBR congestion control algorithm [12]. BBR estimates bottleneck bandwidth by tracking the maximum delivery rate and registering a “pipe-full” signal once it detects the link is saturated. This pipe-full signal can be used to stop the test after a specified number of pipe-full events (e.g., 3, 5, or 7). A smaller value terminates more aggressively, saving bandwidth but risking underestimation, while larger values improve accuracy at the cost of higher bandwidth consumption. Although this heuristic adapts to congestion dynamics, it relies on a narrow transport-layer signal and struggles in high-speed tests where pipe-full events occur late or not at all.

**Throughput stability heuristic (TSH).** TSH is a dynamic heuristic originated by Netflix’s Fast.com [2], which is a complete test design rather than an explicit early termination so-

lution. The key idea is to monitor throughput over time and terminate the test once the throughput remains within a small tolerance or threshold. This method can be adapted for early termination more generally: a test could be stopped once throughput fluctuations fall within the tolerance window. Two parameters govern this tradeoff: the tolerance level and the stability window length. Smaller tolerances and longer windows yield higher accuracy but longer tests, while looser tolerances and shorter windows enable earlier termination with greater error risk. TSH is thus more adaptive than static thresholds, but remains vulnerable to variability from bursts, which may delay termination.

**Crucial interval sampling (CIS).** CIS is a dynamic heuristic originated by the FastBTS methodology [33]. Its central idea is the notion of a *crucial interval*: a narrow range in which most throughput samples concentrate. As a test stabilizes, consecutive crucial intervals become increasingly similar, and a connection is deemed “converged” once their similarity exceeds a threshold. While FastBTS embeds CIS into a new test design, it can also be adapted for early termination: a test can be stopped once the difference between consecutive crucial intervals falls below a tunable bound. The threshold, denoted by  $\beta$ , acts as knob to tune the accuracy–efficiency tradeoff. Like TSH, however, CIS remains sensitive to short-term variability and relies solely on throughput-based signals.

## 3 TURBOTEST’S Motivation

**Early termination as optimal stopping.** Consider a speed test that runs for  $T$  seconds and produces a sequence of samples  $x_i$  (a vector of network measurements such as throughput and RTT). Let the true throughput  $y_{\text{true}}$  be the throughput estimate computed from the full sequence. The early termination problem is to determine a stopping time  $\tau \leq T$  and a prediction function  $f$  such that the reported throughput  $\hat{y} = f(x_1, \dots, x_\tau)$  approximates  $y_{\text{true}}$  while minimizing  $\tau$ , which in turn minimizes data transfer. This naturally fits the classical optimal stopping framework: at each time step  $t$ , the algorithm observes the prefix  $x_1, \dots, x_t$  and decides whether to stop ( $\tau = t$ ) or continue ( $\tau > t$ ). The central objective is to balance accuracy – ensuring the estimation error,  $|\hat{y} - y_{\text{true}}|$ , is within  $\epsilon$  of the true throughput – and efficiency, i.e., minimizing data transferred before termination.

**Limitations of existing approaches.** Heuristic solutions to the early termination problem are fundamentally limited in three ways. First, they rely on fragile assumptions about network behavior that do not hold uniformly in practice. For example, BBR often fails on very high-throughput tests (e.g., >400 Mbps), since the test can complete before enough pipe-full signals are observed [12]. This is particularly problematic considering that high throughput speed tests transfer a larger amount of data, leading to excess operator costs. CIS makes assumptions about smooth convergence that can lead

to premature termination during transient bursts or wireless variability.

Second, heuristics exploit only a narrow slice of the signal space, typically the throughput time series or a single transport-layer variable, while ignoring subtler patterns across other fields (e.g., RTT, retransmissions, congestion window). These signals are readily-available through the Linux `tcp_info` struct [18] and could provide earlier and more reliable evidence of convergence.

Third, heuristics conflate two intrinsically distinct learning problems: regression (estimating the final throughput) and classification (deciding when to stop). Even when a heuristic chooses the “right” stopping time, naive throughput estimation can yield large errors (e.g., CIS or BBR stopping at the right point but reporting a biased aggregate). In reality, regression and classification are tightly coupled: smarter regression reduces error at the stopping point and provides stronger predictive signals for deciding whether to stop.

Taken together, these limitations highlight why existing heuristics struggle to balance accuracy and efficiency across diverse access networks. They either stop too early, stop too late, or misestimate throughput even when they stop at the right time. This motivates the need for a systematic framework that leverages richer signals and explicitly decouples prediction from stopping, as we develop in TURBOTEST.

**The case for decoupling termination and prediction.** The optimal stopping formulation naturally decomposes into two distinct subproblems: *termination* (when to stop) and *prediction* (estimating the final throughput). These subproblems have different objectives and error sensitivities. In particular, prediction is a regression task, whereas termination is a classification task with asymmetric risks – prematurely stopping may be far more costly than stopping a little late. Decoupling the two lets us design predictors and stopping policies independently, improving modularity and extensibility. At the same time, the two tasks are not independent: stopping policies can be made more aggressive by conditioning them on the predictor outputs to enable more aggressive yet accurate termination strategies.

**The case for machine learning.** Early termination is inherently a multi-objective problem, requiring simultaneous trade-offs between accuracy and efficiency that existing single-objective heuristics cannot capture. A machine learning approach naturally encodes these trade-offs in the loss function, giving providers explicit knobs to balance cost against accuracy. Moreover, when trained on large and diverse datasets, ML models can generalize across heterogeneous network conditions without relying on narrow assumptions such as throughput convergence or stationarity. This is particularly feasible here: providers already collect millions of complete tests spanning diverse environments, offering ample training data. Finally, ML can leverage a richer signal space such as those available from `tcp_info` structure of the Linux kernel stack. A joint consideration of multiple signals such as RTT

and congestion window behavior could provide earlier, more reliable evidence of termination.

**Takeaway.** Unlike classic optimal stopping problems, speed tests are non-stationary and highly variable across between access mediums and network conditions. Heuristics that rely on single features or stationary assumptions cannot robustly balance accuracy and efficiency. TURBOTEST addresses this by (i) explicitly decoupling termination from prediction, (ii) leveraging richer, multi-dimensional features that capture more subtle patterns in transport dynamics, and (iii) exposing tunable parameters so providers can balance accuracy and overhead across a wide range of conditions. By doing so, TURBOTEST is designed to shift the accuracy-savings frontier outward, achieving higher efficiency at any accuracy budget compared to existing heuristics.

## 4 TURBOTEST: Design and Implementation

TURBOTEST is a data-driven framework for early termination that leverages diverse network signals and provides tunable parameters to balance accuracy and efficiency. The key insight is that early termination can be decomposed into two coordinated tasks: (i) a *termination task*, where a classifier  $\pi_\phi$  decides whether the current data suffices to stop, and (ii) a *prediction task*, where a regression model  $h_\theta$  estimates the final throughput from partial measurements. During inference, the termination task precedes prediction. However, during training, the classifier ( $\pi_\phi$ ) is conditioned on the prediction hypothesis ( $h_\theta$ ). This coupling enables aggressive yet accurate termination: regression accuracy sharpens stopping decisions, while classification ensures regression is invoked only when appropriate.

Figure 1 illustrates the overall workflow. During training, we generate truncated partial sequences from each complete speed test. **Stage 1** trains a regression model on these sequences; we then compute an *oracle stopping time*  $t^*$  – the earliest point where prediction error falls within tolerance  $\epsilon$ ; and **Stage 2** trains a stopping classifier to reproduce these oracle decisions, learning when partial data suffices. At inference, the **Stage 2** classifier operates online as measurements arrive; once it signals termination, the **Stage 1** regressor predicts the final throughput.

### 4.1 Stage 1: Speed Estimation (Regression)

**Problem setup.** The first stage of TURBOTEST aims to predict the final throughput  $y_{\text{true}}$  of a test given only partial observations. This regression task matters because any early termination will be judged by the accuracy of its predicted throughput: even if the stopping decision is correct, poor prediction can still yield large errors. A reliable predictor therefore reduces error regardless of when the test halts and provides the foundation for the stopping policy in Stage 2.

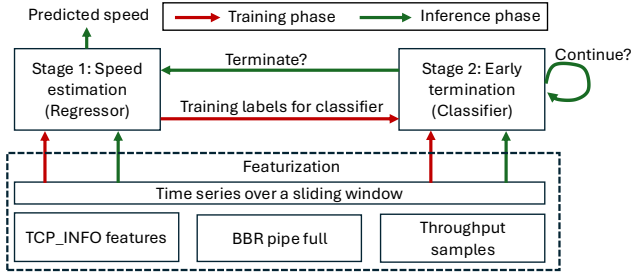


Figure 1: TURBOTEST workflow.

**Feature space design.** A key design question is which features best capture the evolution of a speed test. For instance, because the TSH, CIS, and BBR-based schemes<sup>1</sup> rely solely on the throughput time series, they miss subtler cues about convergence, particularly under variable conditions (e.g., wireless links or cross traffic). We instead consider three classes of readily available signals: (i) throughput samples, (ii) the pipe-full signal from TCP BBR, and (iii) congestion-control-agnostic transport-layer metrics from the `tcp_info` struct in the Linux kernel. The rationale is that richer feature combinations expose patterns not visible to any single signal.

**Model architecture.** We evaluated several models for mapping features to throughput. Linear regression offers interpretability but cannot capture nonlinear dynamics. Tree-based ensembles such as XGBoost handle heterogeneous features, are robust to outliers, and yield interpretable feature importances. Neural networks (feed-forward nets, Transformers) can exploit temporal dependencies. We default to XGBoost for its strong performance on mixed-scale tabular data, fast training, and resilience to missing or sparse inputs – a critical property for `tcp_info` features, where low-throughput tests may send no bytes for hundreds of milliseconds. We also experimented with neural network architectures, using feed-forward nets as lightweight baselines and Transformers for long-range sequence modeling (see §5.5 for comparison).

**Training objective.** Another crucial decision is the choice of training objective. Relative-error losses (e.g.,  $L_{\text{rel}}(y, \hat{y}) = \frac{|y - \hat{y}|}{|y| + \gamma}$ ) emphasize proportional accuracy but can produce unstable gradients as  $y \rightarrow 0$ . In contrast, absolute-error losses such as squared error ( $L_{\text{sq}}(y, \hat{y}) = (y - \hat{y})^2$ ) yield stable optimization and prioritize accuracy at high speeds, leading to greater data savings. Hybrid objectives that combine relative and absolute error are possible but add complexity. We therefore use Mean Squared Error (MSE) for its simplicity and efficiency.<sup>2</sup> In principle, the regressor could be specialized by throughput regime. For example, one variant could be trained for low-throughput tests (<25 Mbps) and another for high-throughput tests (>400 Mbps). However, for simplicity and

ease of deployment, we use a single unified regressor across all tests.

## 4.2 Stage 2: Early Termination (Classification)

**Problem setup.** Stage 2 of TURBOTEST decides whether sufficient evidence has accumulated to terminate the test at time  $t$ . This is naturally framed as a classification problem: given features from the partial sequence, the policy must predict whether additional measurements would materially change the throughput estimate. Unlike Stage 1, which focuses on the *accuracy* of the estimate, Stage 2 focuses on the *sufficiency* of the available information.

**Label construction.** To ensure stopping decisions are grounded in prediction quality rather than arbitrary thresholds, we derive oracle labels from Stage 1. For each test  $i$ , we define the oracle stopping time  $t_i^*$  as the earliest point at which the regression prediction error falls within the operator-specified tolerance  $\epsilon$ . Samples at  $t \geq t_i^*$  are labeled as positive (safe to stop), while earlier samples are labeled as negative (must continue). This approach ties classification directly to achievable accuracy, avoiding the brittle parameterization that limits existing heuristics.

**Feature design.** A central question is what features the classifier should consume. One option is to base stopping decisions entirely on Stage 1 predictions, mirroring heuristics driven solely by throughput stability. Another is to ingest the same raw features as Stage 1. A third is to collapse prediction and classification into a single joint model. We adopt the second option because it avoids invoking the regressor at every step (lighter than option 1) while preserving modularity between the two tasks.

**Model architecture.** We evaluated several alternatives for the classifier, including logistic regression, tree-based ensembles (e.g., XGBoost), and Transformers. We default to Transformers because they empirically yield the best results – likely because they are designed to capture long-range temporal context and can detect subtle signs of convergence or instability in partial feature sequences.

**Loss function.** We train the stopping classifier using standard binary cross-entropy loss. This choice reflects the nature of the task: the model must decide between two outcomes – whether the current partial sequence is sufficient to stop or not. Binary cross-entropy directly penalizes misclassification, aligns naturally with probabilistic outputs, and provides stable gradients for training.

**Training vs. inference.** An important distinction arises between training and inference. During training, Stage 1 precedes Stage 2: the regression model produces oracle stopping times used as labels for the termination classifier. At inference, however, the order reverses: Stage 2 runs online to decide when to stop the test, after which Stage 1 reports the throughput at the chosen stopping point. This inversion highlights the synergy of the two stages—regression informs classifica-

<sup>1</sup>The pipe-full signal is only used for the termination decision.

<sup>2</sup>A limitation is that MSE de-emphasizes relative errors at low speeds.

tion during training, while classification governs regression at runtime.

### 4.3 TURBOTEST Implementation

We implement TURBOTEST as a two-stage pipeline consisting of a throughput regressor (Stage 1) and a stopping classifier (Stage 2). Unless otherwise specified, we use an XGBoost regressor in Stage 1 and a Transformer classifier in Stage 2, as validated through ablation studies (§5.5).

**Features.** The feature set includes: (i) throughput samples (instantaneous and cumulative average), (ii) the number of TCP BBR pipe-full signals, and (iii) a subset of metrics from `tcp_info`. While `tcp_info` provides a rich set of metrics [18], we focus on congestion window size, bytes in flight, RTT, retransmissions, and duplicate ACKs, as these directly capture throughput, delay, and loss dynamics. NDT already records these metrics at a 10 ms granularity, but we observed that the sampling intervals are not exact and vary across samples. To ensure uniform sequence length and reduce processing cost, we resample these metrics to 100 ms granularity, computing the mean and standard deviation within each window. This yields 13 features per 100 ms interval – a 10-second test is represented as a 1300-dimensional feature vector.

**Model parameters.** We now describe the model parameters of the regressor and classifier:

- *Stage 1 (Regression).* We employ XGBoost with depth 7, 1,500 trees, and a learning rate of 0.03, trained using mean squared error (MSE). Accounting for our sliding-window training technique, the dataset includes nearly 15 million samples, whose diversity helps mitigate overfitting to specific speed tests.
- *Stage 2 (Classification).* We use a transformer model with 8 layers, hidden dimension 128, 8 attention heads, and dropout 0.1, trained with binary cross-entropy loss, the Adam optimizer, learning rate  $10^{-3}$ , and batch size 4,096. We intentionally keep the transformer comparatively lightweight to enable fast inference in deployment scenarios.

**Partial sequence construction.** For model training and inference, we extract partial sequences from the test feature vector. While features are extracted at 100 ms granularity, TURBOTEST makes termination (and prediction) decisions at 500 ms strides. This design allows us to amortize the model inference overheads, and ensures real-time feasibility. For the transformer-based classifier, at time  $t$ , we use the entire feature history up to  $t$ . In contrast, the XGBoost-based regressor considers only the most recent two seconds. This is because excessive padding, especially in the early phases of a test, tends to confuse the model, whereas a two second window provides reasonable temporal context. For  $t < 2$  seconds, we pad the feature vector by duplicating features from the latest 100 ms window, thereby reducing noise from input sparsity.

**Deployment parameter ( $\epsilon$ ).** The only operator-facing parameter is  $\epsilon$ , which specifies the acceptable error tolerance and is used to define the ground-truth labels for the classifier. For example, with  $\epsilon = 20$ , the pre-trained regression model identifies the earliest stopping point that keeps the prediction error within 20%. All subsequent points are labeled as *terminate*, while earlier points are labeled as *continue*. We evaluate across  $\epsilon \in \{5, 10, 15, 20, 25, 30, 35\}$ , and later extend to an RTT-adaptive setting where  $\epsilon$  varies at runtime.

**Inference workflow.** At runtime, each new measurement window is encoded into features and passed to Stage 2. If the classifier outputs *continue*, the test proceeds to the next window. If it outputs *stop*, the regressor is invoked to produce the final throughput estimate, which is returned as the test result. This preserves the two-stage coupling while ensuring that regression is executed only once per terminated test.

## 5 Evaluation

We structure our analysis around five operator-relevant questions: (1) Does TURBOTEST consistently extend the accuracy-savings Pareto frontier compared to heuristics such as BBR and CIS (Section 5.2)? (2) Which classes of tests – defined jointly by throughput tier and RTT – benefit most from an ML-based termination strategy (Section 5.3)? (3) How can adaptive parameterization, tuned by grouping strategies such as RTT- or speed-based bins, further improve the balance between accuracy and efficiency (Section 5.4)? (4) Which design components of TURBOTEST are most responsible for its performance gains (Section 5.5)? and (5) How well does TURBOTEST generalize and scale in practice (Section 5.6)?

### 5.1 Experimental Setup

**Datasets.** For evaluation, we use approximately 1 million download speed tests from M-Lab’s global measurement platform between April 2024 and March 2025.<sup>3</sup> To avoid skew from M-Lab’s early termination policy, we exclude all tests pre-terminated by the platform’s 250 MB data transfer cap<sup>4</sup>. The data is split into three disjoint subsets: (i) *training set* of 800k tests (Apr 2024–Jan 2025), balanced across speed tiers. Tiers are defined using thresholds at [25, 100, 200, 400] Mbps, aligned with policy definitions in the US where links below 25 Mbps and 100 Mbps are classified as unserved and under-served, respectively. A balanced sampling ensures adequate representation of >400 Mbps links, which are fewer but dominate bandwidth overhead. (ii) *test set* comprising over 1 million tests (July 2024–Jan 2025), randomly sampled from the natural distribution of speed tests and used for main evaluation. (iii) *robustness set* of 133k tests (Feb–Mar 2025),

<sup>3</sup>While we focus on NDT due to its publicly available test data, our methodology naturally extends to other tests.

<sup>4</sup>Currently, M-Lab randomly applies the data cap on a fraction of their tests.

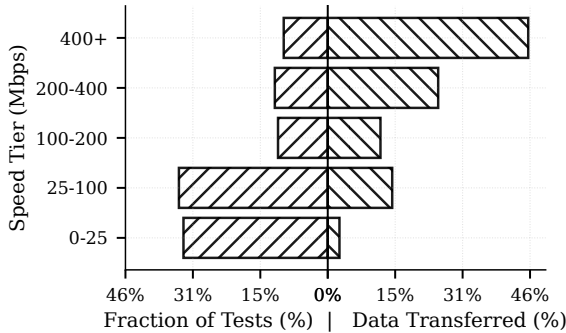


Figure 2: Distribution of tests across different speed tiers. The left bars show the fraction of total tests in each group, while the right bars show the fraction of total data transferred.

used to assess robustness to concept drifts. The random sampling for the latter two datasets preserves the natural distribution of speeds and RTTs in the dataset. Furthermore, random sampling ensures a geographically diverse set of speed tests, spanning 46 countries across 6 continents.

Figure 2 shows the breakdown of tests across speed tiers, reporting both the fraction of data points and their cumulative contribution to total bandwidth overhead. The imbalance highlights that higher speed tiers, despite having fewer tests, contribute disproportionately to overall bandwidth consumption. As an example, the 400+ Mbps tier includes roughly  $4\times$  fewer tests than the 0–25 Mbps tier, yet it contributes about  $10\times$  more traffic volume, highlighting the outsized role of high-speed tiers in driving measurement costs. Our splitting strategy ensures the model learns from a balanced training distribution while evaluations reflect real-world conditions, avoiding contamination between training and testing.

**Baselines.** We compare TURBOTEST against three classes of dynamic early termination strategies, each exposing a tunable parameter to balance relative error against data savings. Static threshold approaches are excluded, as prior work has shown them to be both ineffective and inflexible in managing the trade-off between data overhead and accuracy [12].

**BBR Pipe-Full (BBR).** The BBR heuristic terminates a speed test once the congestion control algorithm declares the connection “pipe-full” [12]. We vary the termination threshold by requiring a minimum of  $\{1, 2, 3, 5, 7\}$  pipe-full signals before stopping, where larger values reduce premature termination but lower savings. We refer to these configurations as *BBR pipe-X*, where  $X$  is the required number of pipe-full signals (e.g., *BBR pipe-7*, for 7 pipe-full signals till termination).

**Crucial Interval Sampling (CIS).** CIS terminates a test once the throughput distribution stabilizes relative to a reference interval [33]. We tune the similarity threshold  $\beta$  over  $\{0.6, 0.8, 0.85, 0.9, 0.95, 1.0\}$ , with higher  $\beta$  enforcing stricter similarity and improving accuracy at the cost of savings.

**Throughput Stability Heuristic (TSH).** This Fast.com-style

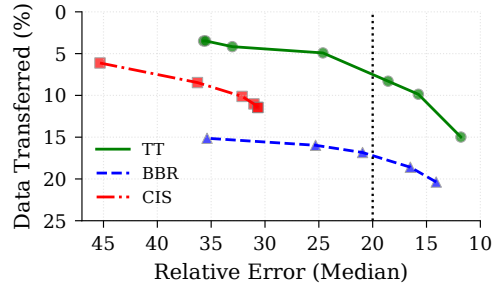


Figure 3: Pareto frontiers of TURBOTEST, BBR, and CIS.

heuristic terminates a test once throughput variation stabilizes over a sliding window. We adjust the stability threshold over  $\{20\%, 30\%, 40\%, 50\%\}$ , where smaller thresholds yield more accurate but longer tests. Given that we observe TSH’s data savings to be much smaller compared to our previously mentioned baselines, we leave this analysis to Appendix A.

**Success metrics.** We evaluate each method along two complementary dimensions: accuracy and efficiency. For a given test, let  $T$  be the true throughput from a full-length run,  $T^{\text{early}}$  the throughput estimate at termination,  $B$  the bytes transferred in a full run, and  $B^{\text{early}}$  the bytes transferred until termination.

**Data transfer** is defined as  $\frac{B^{\text{early}}}{B}$ , where smaller values indicate greater efficiency from early termination. Unless otherwise noted, we report this metric as *cumulative data transferred*,  $\frac{\sum_i B_i^{\text{early}}}{\sum_i B_i}$ , rather than as per-test averages. This choice reflects the operator’s perspective: what ultimately matters is the aggregate reduction in bandwidth consumption and server-side costs. The cumulative view thus captures the real-world benefit of a stopping policy at scale.

**Relative error** is defined as  $E_{\text{rel}} = |T - T^{\text{early}}|/T$ , capturing the accuracy of throughput estimation under early stopping. We use relative error rather than absolute error because speed tests span several orders of magnitude in throughput (e.g., sub-10 Mbps vs. 1 Gbps tiers). An absolute error of 10 Mbps may be negligible in a high-speed tier but catastrophic in a low-speed tier; normalizing by  $T$  ensures comparability across tiers. Unless otherwise noted, we report the *median* relative error across tests, which reflects typical accuracy.

## 5.2 TURBOTEST vs. Existing Strategies

We first ask whether TURBOTEST extends the accuracy–savings frontier beyond existing early termination strategies. Each method exposes a tunable parameter that trades accuracy for efficiency:  $\epsilon$  in TURBOTEST, pipe-full counts in BBR, and similarity threshold  $\beta$  in CIS. Because these parameters span different operating points, comparing strategies with their default or “fixed” settings alone is uninformative. Instead, we compare their *Pareto frontiers* in the two-dimensional space of accuracy (median relative error) and efficiency (cumulative

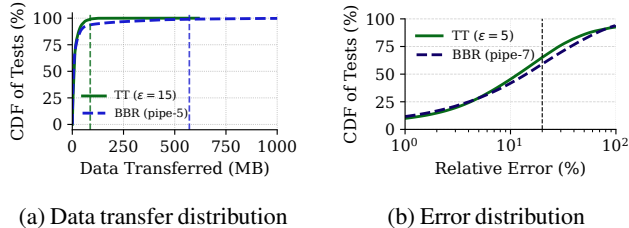


Figure 4: Distribution of data transfer and relative errors across samples for parameters that satisfy the median error  $< 20\%$  constraint.

data transfer). Figure 3 shows these frontiers for TURBOTEST, BBR, and CIS. The X-axis shows median relative error. The Y-axis reports cumulative data transfer as a percentage of the 172.3 TB required if all 1 million tests in the set ran to completion. The lines denote the frontier for each method with markers indicating parameter settings.

TURBOTEST dominates across the entire frontier. At  $\epsilon = 20$ , it achieves 95.1% savings (4.9% transferred) with 24.6% error, compared to BBR’s maximum of 84.9% savings (15.1% transferred) at 35% error. Across the sweep of  $\epsilon$ , TURBOTEST spans 85–96% savings with 12–36% error, while BBR covers a similar error range (14–35%) but never exceeds 85% savings. CIS falls in between: its default yields 31.0% error and 89% savings, while reducing  $\beta$  increases data savings only at the cost of sharply increasing error.

At the aggressive end of the frontier (approximately 36% error), TURBOTEST transfers just 6 TB compared to 10.5 TB for CIS (45.3% error) and 26.1 TB (35.4% error) for BBR—a 1.8–4.4 $\times$  reduction. At the conservative end ( $\epsilon = 5$ ), it transfers 25.8 TB with 11.8% error, versus 35.1 TB for BBR at 14.1% error. Detailed metrics for each termination methodology are enumerated in Table 1 of Appendix A.

We now focus on an operationally meaningful target: achieving a median error below 20% when applying our chosen method on the complete test dataset. From Figure 3, only TURBOTEST with  $\epsilon \leq 15$  and BBR with more than five pipe-full signals qualify. CIS in its configuration with the best accuracy, has a median relative error of 30.7%, which exceeds our 20% defined median relative error threshold. Hence, we exclude it from our analysis moving forward. Using the most aggressive parameters (i.e. tending to stop earlier) with median relative error less than 20%, we observe that TURBOTEST transfers about 14.3 TB, nearly 2.25 $\times$  less than BBR (32 TB). Put in perspective, this  $\sim 92\%$  savings (8% transferred) would reduce M-Lab’s monthly overhead from 12 PB to under 1 PB—a 12 $\times$  reduction from the data transferred over fixed-duration tests.

Figure 4a further compares these configurations by plotting the CDF of data transferred per test. While BBR occasionally transfers less data than TURBOTEST per test, the differences are marginal and disappear once we move to the upper per-

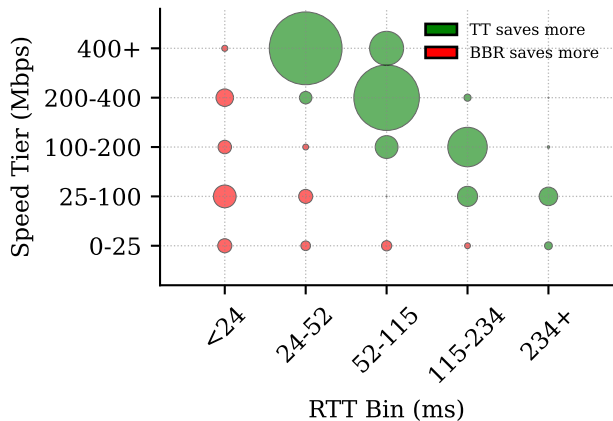


Figure 5: Delta in data transfer between TURBOTEST and BBR across all speed-tier  $\times$  RTT groupings. Point size indicates the magnitude of the difference in data transferred. Green denotes cases where TURBOTEST transfers less, while red denotes cases where BBR transfers less.

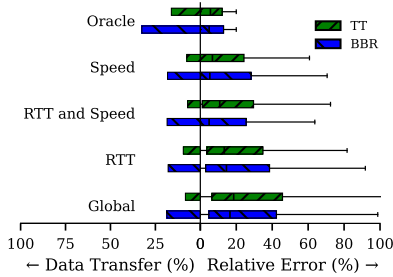
centiles. Specifically, at the 99th percentile (as visualized by the vertical marking for TT and BBR, respectively) we find that BBR pipe-5 transfers  $>550$  MB for the top 1% of tests whereas the top 1% of tests for the most aggressive TURBOTEST model ( $\epsilon=15$ ) transfers only 87 MB (6.3 $\times$  less).

Figure 4b examines error distributions in these settings when selecting the most conservative (tending to stop late) configurations of TURBOTEST ( $\epsilon=5$ ) and BBR-pipe 7. Both schemes are heavy-tailed: while both TURBOTEST and BBR meet the 20% bound at the median, neither sustains it at higher quantiles. This motivates the need for more adaptive approaches that can trim the error tail, which we explore next in Section 5.3.

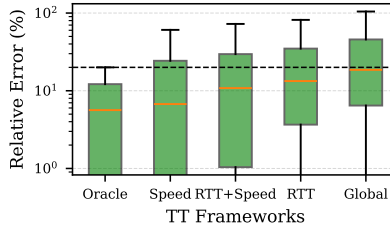
### 5.3 Deep Dive: Who Benefits Most?

Our previous analysis shows the overall strength of TURBOTEST, but we observe that BBR can outperform TURBOTEST for certain subsets of tests. This motivates two questions: (i) what characterizes the cases where heuristics perform better, and (ii) which types of tests benefit most from an ML-based approach? To this end, we build on the 20% error case study from Section 5.2, decomposing the dataset by speed tiers and RTT bins. For RTT, we use thresholds at [24, 52, 115, 234] ms, which approximately correspond to the 25<sup>th</sup>, 50<sup>th</sup>, 75<sup>th</sup>, and 90<sup>th</sup> percentiles of our dataset. This decomposition enables a systematic comparison of strategies across heterogeneous network conditions and reveals where learning-based methods deliver the largest advantages.

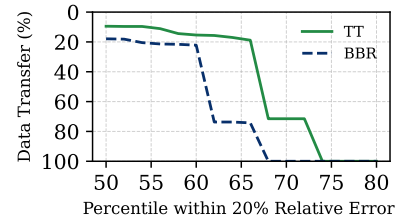
Figure 5 summarizes the results using a matrix visualization across all 25 speed tier  $\times$  RTT bin combinations. Each cell reports the relative advantage of TURBOTEST versus BBR when both are tuned to their most aggressive parameter



(a) Comparison of cumulative data transferred (left) and relative error (right) across different strategies



(b) Distribution of relative error for different TURBOTEST frameworks (20% relative error line is marked for clarity)



(c) Data transfer across RTT-aware termination strategies at varying relative error (20%) percentiles

Figure 6: Adaptive parameterization strategies. The label TT refers to TURBOTEST.

that still satisfies the median error  $<20\%$  constraint identified earlier. The size of each bubble is proportional to the absolute difference in cumulative data transferred, while color encodes the winner: green indicates that TURBOTEST transfers less data, red indicates BBR transfers less. We observe that TURBOTEST outperforms BBR in high-speed tiers, which explains its overall superior data savings, as these tiers contribute the most to total data transfer (see Figure 2). We attribute these gains to the use of a balanced training dataset, which provides better opportunities to learn termination behavior for relatively scarce but critical high-speed test cases. Moreover, TURBOTEST also performs better for tests with higher RTTs, underscoring ML’s superior ability to make effective use of fewer samples in challenging high-latency conditions.

**Takeaway.** Overall, this deep dive demonstrates that TURBOTEST not only dominates heuristics in aggregate but also offers the best accuracy-savings trade-off across most operational scenarios. Its largest gains occur in precisely those tests—high throughput, moderate RTT—that contribute most to aggregate data volume.

## 5.4 Adaptive Parameterization

**Grouping strategies for adaptive parameterization.** Thus far, we have considered global settings or coarse group-wise tuning (by speed tier or RTT bin). A practical question remains: *how should parameters be selected to suppress tail errors while retaining efficiency?* We formalize this as a constrained selection problem and evaluate five strategies, all under the same rule: within each strategy’s grouping scope, we sweep each method’s control knob (TURBOTEST  $\epsilon$  and BBR pipe-full count) and pick the *most aggressive* setting that *keeps the group’s median relative error below 20%*; if no setting satisfies the constraint for a group, that group does not terminate early. Concretely: (1) *Global strategy*: one group (the entire test set); a single parameter is applied to all tests (as in Section 5.3). (2) *Speed-only strategy*: groups are throughput tiers; we select one parameter per tier. (3) *RTT-only strat-*

*egy*: groups are RTT bins; we select one parameter per bin. (4) *RTT+Speed strategy*: groups are all (tier, RTT) pairs; we select one parameter per pair. (5) *Oracle strategy*: groups degenerate to single tests; for each test we choose the most aggressive setting that would keep that test’s relative error  $\leq 20\%$ , otherwise we do not terminate it. The Oracle represents a theoretical upper bound on what grouping can achieve. Detailed information on the specific method chosen for each speed/RTT class is in Appendix A.

Figure 6a summarizes the results using a composite plot: for each strategy, we show cumulative data transferred (bars on left) alongside the distribution of relative errors (box plots on right) for TURBOTEST and BBR. Several observations emerge. First, both Speed-only and RTT-only strategies substantially reduce tail errors compared to the Global strategy, while RTT+Speed offers even greater flexibility. This is further validated in Figure 6b, which highlights how adaptive parametrization in the case of TURBOTEST reduces the median relative error over the simplistic "global" strategy. Unsurprisingly, the Oracle strategy provides the best possible trade-off, reflecting an unattainable upper bound in realizing low relative error for the tail. Second, across all comparisons, TURBOTEST dominates BBR by transmitting  $2\times$  less data while maintaining comparable tail errors under equivalent accuracy constraints.

The practical implication lies in the choice between Speed-only and RTT-only grouping. While Speed-only adaptation achieves improvements, it is difficult to deploy because the throughput tier cannot be reliably inferred in the first few hundred milliseconds of a test. RTT-only grouping, by contrast, is practical: RTT can be measured immediately at runtime and provides a strong, deployable basis for adaptation. Thus, RTT-aware parameterization emerges as the most effective middle ground between the simplicity of global tuning and the theoretical optimality of the Oracle.

**Taming the tails.** Finally, we evaluate how well different schemes can contain *tail errors*, i.e., performance beyond the median, under increasingly strict accuracy constraints. Fig-

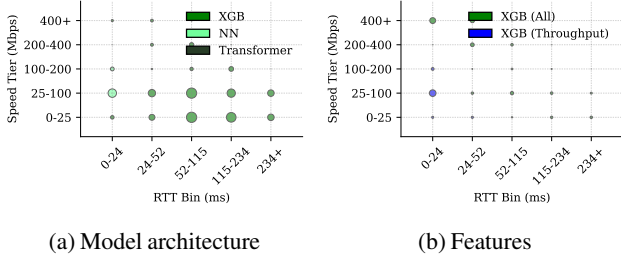


Figure 7: Delta in data transfer across regressors. Point size indicates the magnitude of the difference in data transferred, while color indicates the best performing method.

Figure 6c compares our RTT-aware TURBOTEST framework with BBR when progressively tightening the error requirement from the median to higher quantiles. The evaluation procedure is as follows: for each method, we first ensure that the average (median) test is within a 20% relative error bound. We then select the best possible configuration for data savings under this constraint and measure how savings degrade as the percentile error requirement becomes stricter. The results show a clear separation. At the median (50<sup>th</sup> percentile), both methods can be tuned to satisfy the 20% constraint, but TURBOTEST consistently achieves higher data savings for the same error bound. When looking at the raw bytes transferred for this region, this translates to  $2\times$  less data transferred than BBR. As we move to the 62<sup>th</sup>–66<sup>th</sup> percentiles, TURBOTEST’s RTT-aware parameterization sustains at less than 20% data transferred, whereas BBR’s increases to 60% or more. More specifically, even in regimes where operators may enforce higher accuracy across a larger fraction of tests, TURBOTEST transfers between  $1.4\times$  and  $4.7\times$  less data than BBR. This demonstrates that adaptive use of richer features allows TURBOTEST to preserve efficiency while curbing errors in the bulk of the distribution.

However, the figure also highlights the fundamental limits of early termination. At the 74<sup>th</sup> percentile, no method—including TURBOTEST—is able to achieve safe early termination, with data savings collapsing to 0% (100% transferred). Roughly one-quarter of tests remain resistant to early termination without inflating prediction error. These hard cases are typically low-throughput flows with high RTT, where variability persists for most of the test duration. This finding underscores both the progress and the boundaries of current approaches: while adaptive ML-based parameterization can dramatically expand the accuracy-savings frontier, a significant fraction of speed tests remain inherently difficult to truncate early, reflecting the complexity of network dynamics in challenging environments.

**Takeaway.** Adaptive parameterization, particularly RTT-aware strategies, provides operators with a practical mechanism to curb tail errors while retaining most of the efficiency benefits of TURBOTEST. While oracle-level adaptivity is in-

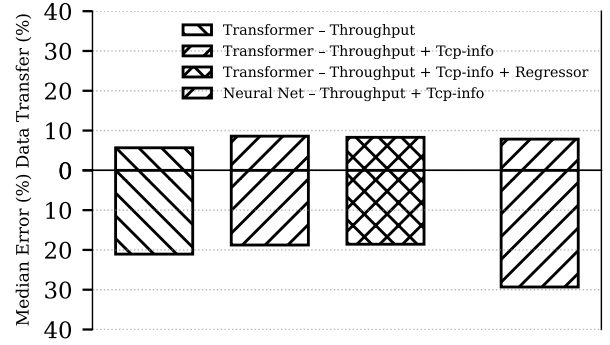


Figure 8: Classifier performance under a fixed XGBoost (TCPInfo) regressor.

feasible, RTT-aware grouping demonstrates that careful parameterization can deliver robust performance at scale.

## 5.5 Ablation Study

Next, we validate that TURBOTEST’s dual-stage ML design is justified. We systematically swap regressors (XGBoost, Neural Net, Transformer), classifiers, and feature subsets (throughput-only, TCPInfo-only, hybrids), as summarized in Figures 7 and 8. We exclude XGBoost from the classifier analysis because its architecture is not easily amenable to training with full length sequences. This results from issues fitting our entire classification dataset into memory at once.

**Regressor (stage 1).** The results are consistent: for stage 1 regression, XGBoost offers the strongest predictive accuracy across throughput and TCPInfo features, while stage 2 classification with a Transformer yields the best stopping decisions and savings-accuracy balance. In Figure 7, the ideal stopping point for each regressor configuration is found, i.e. the earliest possible timestamps with a relative error less than or equal to 20%. We then visualize the delta in the data transferred for all test classes (speed tier+RTT) through this constraint. Figure 7a illustrates that, in a majority of classes, with the exception of the 0-24ms RTT – 25-200Mbps tests, XGboost outperforms the other regressors with higher data savings due to typically earlier stopping points. This is especially evident in the median latency low throughput tiers, through the higher deltas. This indicates that XGboost is able to capture dynamics that other models miss. Recent research has shown that Gradient boosted decision trees excel in handling heavy-tailed distributions and skews, which are inherent in network data [22]. Figure 7b shows that adding in TCP-info features only marginally helps reduce the ideal stopping point/time, with the delta between the throughput and TCP-info variants remaining small.

**Classifier (stage 2).** Figure 8 compares transformer-based

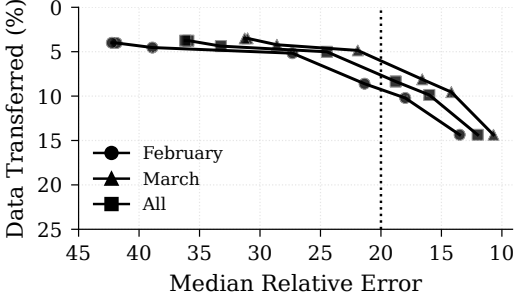


Figure 9: Pareto frontiers for the new distribution in February and March.

classifiers at  $\epsilon = 15$ . All three variants achieve broadly similar performance, with the throughput-only model showing a modestly higher median relative error (21%) compared to those augmented with TCP-info or regressor outputs. The addition of TCP-info slightly improves stopping accuracy, though the incremental gains over throughput alone are relatively small. By contrast, our end-to-end neural network variant transfers less data but exhibits substantially higher error (29%), underscoring that it fails to reliably capture stopping dynamics. TCP-info features offer only marginal improvements over throughput alone, so the main benefit comes from using a Transformer classifier itself rather than the exact feature mix.

**Takeaway.** These ablations demonstrate that both stages are essential. XGBoost in Stage 1 provides robust regression accuracy, especially in skewed and heavy-tailed network conditions, while the Transformer in Stage 2 delivers strong stopping decisions. Using more elaborate feature sets (e.g., TCPInfo) offers only marginal gains, while applying the same model type across both stages (e.g., Transformer-only) reduces performance. Together, these results confirm that TURBOTEST’s two-stage design is well-founded and critical for achieving its accuracy–savings frontier.

## 5.6 Robustness and Overheads

**Robustness to concept drift.** To assess robustness under temporal drift, we evaluate all  $\epsilon$  configurations of TURBOTEST on the *robustness set*—133k speed tests collected in February and March 2025—while the model was trained exclusively on April 2024–January 2025 data. Figure 9 plots Pareto frontiers for February and March alongside the frontier for the 2024–2025 training period, allowing us to directly visualize how accuracy–savings trade-offs evolve over time. Overall, we observe only mild drift: median relative error shifts by less than 2% across the full robustness set. However, February exhibits larger deviations than March, with roughly 4% higher median error at  $\epsilon = 15$ . Inspecting the data reveals that the February set contains more low-throughput, high-RTT tests, concentrated in the 90<sup>th</sup> percentile RTT bin. This suggests that TURBOTEST remains consistent in which regimes

it performs well (stable, higher-throughput connections) and where it struggles (sparse, high-latency flows), underscoring the need for periodic retraining to maintain accuracy as the distribution of test conditions evolves.

**Training overhead.** We measure TURBOTEST’s offline training cost to evaluate training across all  $\epsilon$  configurations. Stage 1 is  $\epsilon$ -independent (fit XGBoost once on the full training set), while Stage 2 trains a transformer (classifier) per  $\epsilon$ . On a 4×A100 node (256 GB RAM) with a 64-core AMD CPU, Stage 1 on 800k tests takes 14 minutes and Stage 2 (5 epochs) takes ~50 minutes, totaling 64 minutes per  $\epsilon$  (5.8 hours for seven). Since training is offline and can be parallelized, this can make the effective training time to 1.06 hours, making full retraining practical at deployment scale.

**Runtime overhead.** A key question for deployment is whether TURBOTEST can operate in real time without incurring prohibitive inference costs. In particular, the use of a Transformer-based classifier for stopping decisions may appear impractical for online measurement pipelines. To assess feasibility, we measure the wall-clock inference time of both stages of the pipeline: the XGBoost regressor (Stage 1) and the Transformer classifier (Stage 2). Our measurements report the latency from the arrival of a new TCPInfo snapshot to the model’s output, excluding TCPInfo preprocessing time since that step is implementation-dependent and typically amortized.

For Stage 1, we simulate speed test execution by feeding TCPInfo snapshots in 500 ms increments and running inference across varying batch sizes that mimic the workload of M-Lab measurement servers (from a handful of concurrent tests up to nearly 1,000). The regressor consistently produces predictions within 10 ms, averaging 6.3 ms, with only mild increases as batch size grows. For Stage 2, we run the Transformer classifier in tandem with Stage 1 after warm-up. Once in steady state, classification decisions are produced within 14 ms on average, with stable latency across batch sizes; occasional outliers (up to 133 ms) stem from scheduling delays rather than the model itself.

**Takeaway.** Overall, these results demonstrate that TURBOTEST can comfortably meet real-time constraints: both prediction and stopping decisions are returned an order of magnitude faster than the 500 ms decision interval. While further runtime optimizations could reduce latency even more, the current design already establishes the feasibility of deploying TURBOTEST in production environments.

## 6 Related Work

**Lightweight speed tests.** Various speed testing approaches have evolved that aim to balance accuracy and efficiency. Early work relied on UDP probing with crafted packets to estimate available bandwidth [13, 16, 29, 30]. For instance, Pathload uses periodic packet trains [16], while Pathchirp introduces exponentially spaced bursts [29]. These methods

face challenges from fluid traffic assumptions, interrupt coalescing, differential treatment of UDP traffic, and timestamp inaccuracies [17], challenges that worsen in high-speed networks. Orthogonal to active speed testing, prior work has explored passive bandwidth estimation by piggybacking on background user traffic (e.g., pre-staged ads) [25, 26] or synthesizing estimates from network telemetry [35].

Recent efforts in active measurement include FastBTS, which modifies the probing and termination for TCP-based tests. Gill et al. [12] design a heuristic that leverages transport signals for early termination. The most closely related work comes from Maier et al. [21] and Arifuzzaman et al. [3], who apply ML for early termination. Maier et al. [21] restrict the problem to a fixed-point binary decision, while Arifuzzaman et al. [3] consider both regression and classification, but with fixed-input models that rely solely on throughput signals and are evaluated only in emulated conditions. In contrast, we propose a general, data-driven approach extensively validated on real-world network data.

**Optimal stopping problem.** Optimal stopping is a classic problem with applications in finance, industry, and health-care. Traditional solutions are based on modeling, leveraging stochastic processes [28] and control theory [31]. However, due to the non-stationary nature of network conditions, data-driven approaches are more relevant. The closest ML-based line of work is early time-series classification [5, 14], which seeks to classify a sequence as early as possible. Techniques include combining a predictor (a classifier in this case) with heuristic termination rules (e.g., confidence thresholds or stability checks) [11], identifying discriminative shapelets [32], or incorporating the cost of delay into the optimization objective [8]. A key distinction is that our setting involves the regression task, where confidence-based stopping is less natural and more challenging.

**Improving speed tests.** Prior studies have analyzed the impact of factors such as test protocol, network conditions, and server infrastructure on speed test accuracy [10, 19, 20, 34]. Another line of work contextualizes these factors into the crowdsourced data before using it, especially for policy purposes [4, 6, 27]. These efforts are complementary to ours, which focuses on improving the efficiency of tests.

## 7 Future Work and Conclusion

Our evaluation on 1 million M-Lab NDT tests shows that TURBOTEST achieves 1.8-4.4 $\times$  higher data savings than M-Lab’s BBR-based approach while reducing median error, shifting the Pareto frontier outward and approaching oracle bounds. These results highlight the limits of existing heuristics—static thresholds, BBR pipe-full signals, and throughput stability rules—that capture only a narrow slice of the accuracy–savings trade-off. By decoupling prediction from termination, leveraging richer transport-level features, and exposing a single tunable parameter  $\epsilon$ , TURBOTEST delivers

accurate, efficient, and robust termination policies suitable for deployment at scale.

Several avenues exist for future work. Alternate loss functions could mitigate mean-squared error’s bias toward high-speed tiers, and more advanced architectures (e.g., autoregressive transformers) may improve accuracy, especially at the tail. For termination, reinforcement learning could be used to learn policies without explicit labels but would likely require substantially larger datasets. Time-series models geared towards unsupervised learning may be explored to further improve tail performance [15]. Beyond modeling, the methodology generalizes naturally: while we focused on download tests due to their higher data overhead, the same approach applies to upload tests. Similarly, while our evaluation used NDT because of its public data, the framework is extensible to other speed test applications, including multi-connection tests or tools that report throughput using different aggregation rules. Future work may include comparing TURBOTEST variants in controlled environments [9]. Importantly, our approach is agnostic to what a speed test should report – a complementary question outside the scope of this work.

This work provides the first principled learning formulation for speed test optimization, opening opportunities for adaptive measurement frameworks that generalize across heterogeneous networks while reducing the resource cost of a critical public service. In short, TURBOTEST shows that less can indeed be enough.

## 8 Acknowledgment

We thank the anonymous reviewers and our shepherd, Zahaib Akhtar, for their constructive feedback and suggestions. We also acknowledge Sylee Beltiukov for their contributions to architecting TURBOTEST’s production-ready implementation. This work was supported by the National Science Foundation (Graduate Research Fellowship Grant No. 2139319, CAREER Award No. 2443777, and CNS Award No. 2323229), research gifts from Cisco and Google, and by the Anusandhan National Research Foundation (ANRF) under Grant No. ANRF-ECRG-2024-005839. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 using NERSC award NERSC DDR-ERCAP0029768.

## References

- [1] Speedtest by Ookla. [speedtest.net](https://www.speedtest.net), 2024. Accessed: Nov. 2024.
- [2] Internet Speed Test - Fast.com. [fast.com](https://www.fast.com), 2025. Accessed: May. 2025.

- [3] Md Arifuzzaman and Engin Arslan. Swift and accurate end-to-end throughput measurements for high-speed networks. In *The Network Traffic Measurement and Analysis Conference (TMA)*, 2022.
- [4] Steven Bauer, David D. Clark, and William Lehr. Understanding Broadband Speed Measurements. In *Proceedings of the 38th Research Conference on Communication, Information and Internet Policy (TPRC)*, 2010.
- [5] Alexis Bondu, Youssef Achenchabe, Albert Bifet, Fabrice Clérot, Antoine Cornuéjols, Joao Gama, Georges Hébrail, Vincent Lemaire, and Pierre-François Marteau. Open challenges for machine learning based early decision-making research. In *ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) Explorations Newsletter*, 2022.
- [6] David Clark and Sara Wedeman. Measurement, Meaning and Purpose: Exploring the M-Lab NDT Dataset. *SSRN Electronic Journal*, 2021.
- [7] Internet Speed Test - Measure Network Performance - Cloudflare. [speed.cloudflare.com/](https://speed.cloudflare.com/), 2025.
- [8] Asma Dachraoui, Alexis Bondu, and Antoine Cornuéjols. Early classification of time series as a non myopic sequential decision making problem. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2015.
- [9] Jaber Daneshamooz, Satyandra Guthula, Jessica Nguyen, William Chen, Sanjay Chandrasekaran, Ankit Gupta, Arpit Gupta, and Walter Willinger. Netreplica: Toward a programmable substrate for last-mile data generation. arXiv preprint arXiv:2507.13476, 2025. arXiv:2507.13476 [cs.NI].
- [10] Nick Feamster and Jason Livingood. Measuring internet speed: current challenges and future recommendations. *Communications of the ACM*, 2020.
- [11] Mohamed F Ghalwash, Vladan Radosavljevic, and Zoran Obradovic. Utilizing temporal patterns for estimating uncertainty in interpretable early decision making. In *ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) International Conference on Knowledge Discovery and Data Mining*, 2014.
- [12] Phillipa Gill, Cristina Leon, Neal Cardwell, and Christophe Diot. The case for leveraging transport signals to improve internet speed test efficiency. In *Special Interest Group on Data Communications (SIGCOMM) Computer Communication Review (CCR)*, 2025.
- [13] Emanuele Goldoni, Giuseppe Rossi, and Alberto Torelli. Assolo, a new method for available bandwidth estimation. In *2009 Fourth International Conference on Internet Monitoring and Protection*, 2009.
- [14] Ashish Gupta, Hari Prabhat Gupta, Bhaskar Biswas, and Tanima Dutta. Approaches and applications of early classification of time series: A review. *IEEE Transactions on Artificial Intelligence*, 1(1):47–61, 2020.
- [15] Satyandra Guthula, Jaber Daneshamooz, Charles Fleming, Ashish Kundu, Walter Willinger, and Arpit Gupta. Netburst: Event-centric forecasting of bursty, intermittent time series. arXiv preprint arXiv:2510.22397, 2025. arXiv:2510.22397 [cs.NI].
- [16] Manish Jain. Pathload: A measurement tool for end-to-end available bandwidth. In *Proceedings of Passive and Active Monitoring (PAM) Workshop*, 2002.
- [17] Li Lao, Constantine Dovrolis, and MY Sanadidi. The probe gap model can underestimate the available bandwidth of multihop paths. *ACM Special Interest Group on Data Communications (SIGCOMM) Computer Communication Review*, 2006.
- [18] M-Lab. TCP Info. <https://www.measurementlab.net/tests/tcp-info/>.
- [19] Kyle MacMillan, Tarun Mangla, Marc Richardson, and Nick Feamster. Best practices for collecting speed test data. *Available at SSRN 4189044*, 2022.
- [20] Kyle MacMillan, Tarun Mangla, James Saxon, Nicole P Marwell, and Nick Feamster. A comparative analysis of Ookla speedtest and measurement labs network diagnostic test (NDT7). In *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2023.
- [21] Christian Maier, Peter Dorfinger, Jia Lei Du, Sven Gschweidl, and Johannes Lusak. Reducing consumed data volume in bandwidth measurements via a machine learning approach. In *Network Traffic Measurement and Analysis Conference (TMA)*, 2019.
- [22] Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad, Benjamin Feuer, Chinmay Hegde, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.
- [23] MLab. Introducing Data Transfer Limits to NDT. <https://www.measurementlab.net/blog/short-ndt/>.
- [24] Measurement Lab: Test Your Speed. <https://speed.measurementlab.net/>, 2024.
- [25] Alamin Mohammed, Theo Karagioules, Emir Halepovic, Shangyue Zhu, and Aaron Striegel. On the harmful effects of active network probing. In *IEEE International Conference on Computer Communications and Networks (ICCCN)*, 2023.

- [26] Alamin Mohammed, Theo Karagioules, Emir Halepovic, Shangyue Zhu, and Aaron Striegel. Repurpose: A case for versatile network measurement. In *IEEE International Conference on Communications (ICC)*, 2023.
- [27] Udit Paul, Jiamo Liu, Mengyang Gu, Arpit Gupta, and Elizabeth M. Belding. The importance of contextualization of crowdsourced active speed test measurements. In *ACM Internet Measurement Conference (IMC)*, 2022.
- [28] Goran Peskir and Albert N. Shiryaev. *Optimal Stopping and Free-Boundary Problems*. Lectures in Mathematics. ETH Zürich. Birkhäuser, 2006.
- [29] Vinay Ribeiro, Rudolf Riedi, Richard Baraniuk, Jiri Navratil, and Les Cottrell. pathchirp: Efficient available bandwidth estimation for network paths. In *Passive and Active Monitoring (PAM) workshop*, 2003.
- [30] Jacob Strauss, Dina Katabi, and Frans Kaashoek. A measurement study of available bandwidth estimation tools. In *ACM Internet Measurement Conference (IMC)*, 2003.
- [31] John N Tsitsiklis and Benjamin Van Roy. Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives. In *IEEE Transactions on Automatic Control*, 2002.
- [32] Wenhe Yan, Guiling Li, Zongda Wu, Senzhang Wang, and Philip S Yu. Extracting diverse-shapelets for early classification on time series. In *World Wide Web*, 2020.
- [33] Xinlei Yang, Xianlong Wang, Zhenhua Li, Yunhao Liu, Feng Qian, Liangyi Gong, Rui Miao, and Tianyin Xu. Fast and Light Bandwidth Testing for Internet Users. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI'21)*, pages 1011–1026. USENIX Association, April 2021.
- [34] Zesen Zhang, Jiting Shen, and Ricky KP Mok. Empirical characterization of Ookla’s speed test platform: analyzing server deployment, policy impact, and user coverage. In *2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC)*, 2024.
- [35] Shangyue Zhu, Alamin Mohammed, and Aaron Striegel. A Frame-Aggregation-Based Approach for Link Congestion Prediction in WiFi Video Streaming. In *International Conference on Computer Communications and Networks (ICCCN)*, 2020.

## A Appendix

### A.1 TurboTest vs Existing Strategies

Method	Data Transferred (TB / %)	Median Rel. Err. (%)
TT ( $\epsilon = 5$ )	25.8 / 15.0%	11.8
TT ( $\epsilon = 10$ )	17.0 / 9.9%	15.8
TT ( $\epsilon = 15$ )	14.3 / 8.3%	18.6
TT ( $\epsilon = 20$ )	8.5 / 4.9%	24.6
TT ( $\epsilon = 25$ )	6.0 / 3.5%	35.7
TT ( $\epsilon = 30$ )	6.0 / 3.5%	35.5
TT ( $\epsilon = 35$ )	7.2 / 4.2%	33.0
BBR-pipe 1	26.1 / 15.1%	35.4
BBR-pipe 2	27.5 / 16.0%	25.3
BBR-pipe 3	29.0 / 16.8%	20.9
BBR-pipe 5	32.0 / 18.6%	16.5
BBR-pipe 7	35.1 / 20.4%	14.1
CIS $\beta = 0.6$	10.5 / 6.1%	45.3
CIS $\beta = 0.80$	14.6 / 8.5%	36.3
CIS $\beta = 0.85$	17.4 / 10.1%	32.1
CIS $\beta = 0.90$	19.0 / 11.0%	31.0
CIS $\beta = 0.95$	19.7 / 11.4%	30.7
CIS $\beta = 1.0$	19.7 / 11.5%	30.7
No Termination	172.3 / 100.0%	—

Table 1: Median Relative Error (%) and Data transferred (in Terabytes and as a fraction of the data transferred with no early termination strategy) across termination methodologies for the 1 million M-lab NDT Tests.

### A.2 Analysis of Throughput Stability Heuristic (TSH)

We apply TSH on our test dataset of 1 million samples and calculate metrics such as Median Relative Error and Data Transfer as visualized in Table 2. As one can see, by increasing the stability threshold, the amount of data being transferred decreases at the cost of relative error. The least amount of data (greatest data savings) is transferred with the stability threshold value of 50 at 34.9%. However, TT ( $\epsilon=5$ ) for comparison, which is the most conservative TT configuration, has a data transfer value which is much lower at 15%. One can conclude, that TSH excels when the operator’s primary concern is prediction accuracy, with data savings being a secondary factor.

Stability Threshold	Median Relative Error (%)	Data Transfer (%)	Data Transferred (TB)
20	0	70.7	121.8
30	0	52.9	91.1
40	0.91	41.7	71.9
50	2.73	34.9	60.3

Table 2: Median relative error and data transfer for TSH configurations.

### A.3 Configurations for Adaptive Parameterization

Tables 3 and 4 report the best configurations of TURBOTEST, BBR, and CIS when stratified by speed tier and RTT. For each subset of tests (e.g., 0–25 Mbps low-throughput tests), we select the parameter setting for each method that yields the maximum data savings while keeping the overall median relative error below 20%. This customization allows us to explore scenarios in which different methods are preferable depending on the test type. An empty cell indicates that no parameter setting for that method satisfied the 20% median relative error threshold. As shown, all methods struggle to balance relative error and data transfer in the 0–25 Mbps (low-throughput) tier, whereas CIS also performs poorly in several higher-throughput tiers. Similarly, we also notice the pattern of all methods struggling to terminate in high RTT scenarios (>234ms).

Table 5 presents the TURBOTEST configurations after jointly stratifying the dataset by speed tier and RTT. Some bins are marked as having "No tests" because there are few or no tests in these relatively rare categories. This pattern reflects the common empirical tendency for higher-throughput tests to also exhibit lower latency. For brevity, we omit the analogous tables for BBR and CIS.

Table 3: Best configuration for speed-tier strategy.

	0–25	25–100	100–200	200–400	400+
ML	—	( $\epsilon=10$ )	( $\epsilon=30$ )	( $\epsilon=30$ )	( $\epsilon=30$ )
BBR	—	pipe-3	pipe-2	pipe-1	pipe-2
CIS	—	—	—	—	$\beta=0.80$

Table 4: Best configuration for RTT strategy.

Method	<24ms	24–52ms	52–115ms	115–234ms	234ms+
TT	$\epsilon=15$	$\epsilon=15$	$\epsilon=15$	$\epsilon=5$	—
BBR	pipe-5	pipe-2	pipe-3	pipe-7	—
CIS	$\beta=0.80$	$\beta=0.90$	—	—	—

Table 5: Best TT configuration for RTT+speed strategy.

Speed (Mbps)	<24 ms	24–52 ms	52–115 ms	115–234 ms	$\geq 234$ ms
0–25	—	$\epsilon=5$	$\epsilon=5$	—	—
25–100	$\epsilon=5$	$\epsilon=15$	$\epsilon=10$	$\epsilon=10$	$\epsilon=5$
100–200	$\epsilon=35$	$\epsilon=30$	$\epsilon=30$	$\epsilon=25$	—
200–400	$\epsilon=25$	$\epsilon=30$	$\epsilon=30$	$\epsilon=35$	$\epsilon=5$
400+	$\epsilon=25$	$\epsilon=30$	$\epsilon=30$	No tests	$\epsilon=10$