Rethinking Network Foundation Models: From Bigger Models to Better Data

Jaber Daneshamooz
UC Santa Barbara
Santa Barbara, USA

Haarika Manda
UC Santa Barbara
Santa Barbara, USA

Manni Moghimi
UC Santa Barbara
Santa Barbara, USA

Sylee Beltiukov
UC Santa Barbara
Santa Barbara, USA

Walter Willinger
Northwestern University
Evanston, USA

Arpit Gupta
UC Santa Barbara
Santa Barbara, USA

Abstract

Network foundation models (NFMs) report high accuracy on traffic tasks, yet diagnostic studies keep finding that this accuracy rests on shortcuts: artifacts of how the training data was collected. The usual response to a weak model is a bigger one, and better evaluation and better architectures both hold the training data fixed. We argue that the lever is the data. A trace's network context is the set of factors that generated it: the application, the transport, and the network conditions. A good NFM is a *minimal sufficient representation of network context*: it preserves those factors and spends the least data and model capacity needed to do so. We show, in one setting, that better data helps. A semantically deduplicated pre-training set one-tenth the size of a campus corpus matches the full corpus and beats an equal-size random sample on context preservation (how much of the network context the model keeps). Obtaining better data is a *search* over which traces to select and which conditions to generate, where selection is generation limited to what was already collected. Composing real application, transport, and network components makes controlled generation cheap and produces the bytes and headers an NFM reads as in deployment. One context-preservation objective can then decide which data to keep and how much model capacity to retain; the search and the capacity choice remain proposals. For the factors it can specify, networking can pursue minimal sufficiency more directly than most domains, because it can generate traffic through the stack it deploys and check what a representation kept against the conditions it specified.

CCS Concepts

• **Networks** → **Network performance evaluation**; • **Computing methodologies** → *Machine learning*.

Keywords

network foundation models, traffic classification, data curation, representation learning



This work is licensed under a Creative Commons Attribution 4.0 International License.
NTC-R '26, Utrecht, Netherlands
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2980-5/2026/12
https://doi.org/10.1145/3843199.3857225

ACM Reference Format:
Jaber Daneshamooz, Haarika Manda, Manni Moghimi, Sylee Beltiukov, Walter Willinger, and Arpit Gupta. 2026. Rethinking Network Foundation Models: From Bigger Models to Better Data. In *Network Traffic Classification – Reloaded (NTC-R '26), December 07–10, 2026, Utrecht, Netherlands.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3843199.3857225

1 Introduction

Several network foundation models (NFMs), encoders pre-trained on large traffic corpora and reused across tasks, report accuracy as high as 98% on encrypted-traffic classification. Yet they perform poorly once a careful study removes the shortcuts that their data preparation allowed [67]. The pattern is older than NFMs: for over a decade, machine learning (ML) for networking has produced models that work in the lab and fail in the field [52]. Recent work traced these failures to models that key on artifacts of how the data was collected instead of on how traffic behaves [35, 67]. The community now calls this a credibility crisis [60], and NFMs have inherited it whole.

The community has responded along two lines of work. The first is better *evaluation*: intrinsic evaluation inspects a pre-trained encoder's embeddings directly, independent of any downstream task [8]. The second is better *architecture*: network-native designs exclude header fields that leak the label and respect protocol structure by construction [9]. Both make progress: in its own evaluation, netFound reports an F1 score of 0.95 for recovering congestion control, cross-traffic, and queue discipline from its representation, where prior NFMs score below 0.62 [9]. But evaluation can only report what an encoder learned, and architecture can at most filter the training data by excluding fields. The shortcut diagnoses point back to the training data, yet both lines of work leave unchanged which applications, transports, and network conditions the data covers.

Before asking how to fix an NFM, we state what makes one good. We call the factors that generate a trace its *network context*: the application and how a client drives it, the transport and its congestion control, and the network conditions (capacity, latency, queue discipline, cross-traffic). A queue discipline decides which queued packet a link sends next and when to drop packets; cross-traffic is other flows' traffic on the same link. For example, a YouTube video using BBR congestion control on a 20 Mbps, 40 ms path and a file download using CUBIC on a 5 Mbps, 5 ms path differ in application, transport, and network conditions. A good NFM is a *minimal sufficient representation of network context*. Sufficiency means that the representation preserves these factors, so the two traces stay

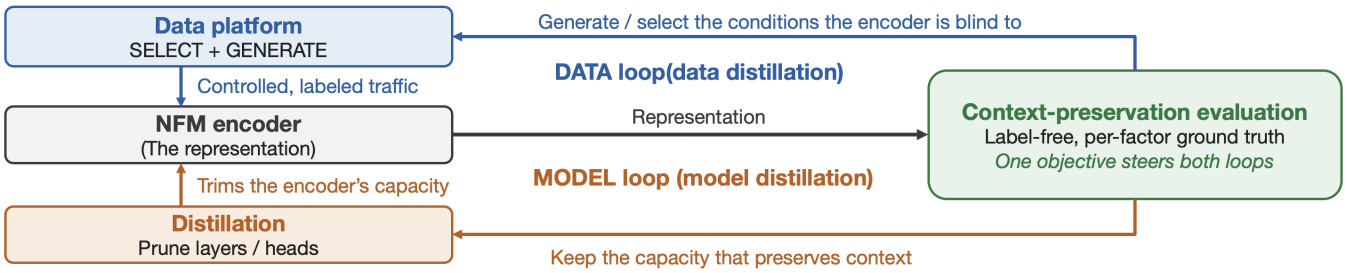


Figure 1: One objective closes two loops over data and model. A platform selects and generates controlled, labeled traffic (top); distillation trims the encoder's capacity (bottom); context preservation (how much network context the encoder keeps) steers both toward the conditions the encoder fails to keep.

distinguishable on each of these factors. Minimality means that the NFM spends the least data and model capacity that still preserve them and discards collection artifacts that lie outside the network context. A model with capacity to spare can use it to store shortcuts. Minimality also lowers compute cost, one of two barriers to deployment beyond shortcuts; the other is policy that bars access to packet payloads [9]. Minimal sufficiency judges success on the representation itself instead of on downstream accuracy numbers that shortcuts can inflate.

Position. *A good network foundation model is a minimal sufficient representation of network context. For the factors it can specify, networking can do what most domains cannot: instead of relying on the data it happened to collect, it can generate traffic under known conditions and check, factor by factor, what the representation kept.*

Meanwhile, the default recipe for a better NFM is a bigger model trained on more traffic, following the broader ML trend toward ever-larger training runs [20]. Recent designs grow the encoder (netFound alone spans 53–663M parameters [9]), add mixture-of-experts layers for capacity [69], or borrow architectures from language and vision [42, 66]. This recipe assumes that the training data is a given: an input one collects once and then works with, as ImageNet is for vision [18]. In networking, this assumption fits badly. No web-scale traffic corpus exists to scale into [40], and a model trained on traffic from a handful of collection conditions can learn those conditions as shortcuts. The assumption is also unnecessary. A network trace comes from a stack one can run (real application drivers, real congestion-control code, an emulated bottleneck link), so one can generate traces for any factor one can specify, such as a congestion-control algorithm or a link's capacity.

We therefore treat the training data as the lever for better NFMs and search over it: we *select* the informative examples from the data that exists and *generate* the conditions missing from the collection. Selection is a special case of generation that can keep only what was collected (§5). Generation can go further, and networking generates traffic through the same stack it deploys, so the bytes and headers an NFM reads are produced as in deployment (§4). Generation is now practical, because composing existing components can produce a labeled dataset for the cost of writing a specification. Generated traffic is realistic only insofar as the emulation matches deployment, and an emulated topology is not the wild Internet. Some factors are latent and hard to specify (e.g., a user's in-session behavior), and finding which data is missing may itself be costly. Addressing these limits is part of our agenda. One objective, *context preservation* (how much of the network context a representation keeps), can then decide what data to keep, what to generate, and how much model capacity to retain (Figure 1; §6). We use NFMs as the running case, but the argument applies to any traffic encoder whose embedding one can probe, such as a backbone trained with contrastive or supervised learning. Work that applies off-the-shelf large language models (LLMs) to networking [61] is out of scope; netFound already notes that text-based LLMs fit raw traffic poorly [9].

That data quality matters is established in ML [1, 53], and closed-loop data collection (e.g., active learning) is not new either. This paper brings both ideas to networking and makes three claims; we demonstrate one and argue for two.

- **What defines success.** We adapt minimal sufficiency [13] to network context and argue that model evaluation should enforce it instead of in-distribution accuracy (§2).
- **A premise, demonstrated.** A semantically deduplicated pretraining set (near-duplicate flows removed) one-tenth the size of a full corpus matches the full corpus on context preservation and beats an equal-size random sample (§3). This single-setting result suggests that data is a lever; it does not validate any generation method.
- **How to pull the lever.** Obtaining better data is a search over what to select and what to generate. Networking can run this search because composing real components makes controlled generation cheap (§4, §5). The choice of model capacity can also be checked against ground truth (§6). The search, its generation half, and the capacity choice remain proposals.

2 A Good NFM Preserves Network Context

An NFM is a compressor. It maps raw traffic to a learned, low-dimensional representation (an *embedding*), and every task built on the NFM sees the traffic only through that embedding. A task cannot recover what the compression discarded, so the compression's quality caps every task that uses it. Shortcuts can also inflate a single task's accuracy [35, 67], so we judge quality on the representation itself.

Sufficiency: preserve the context that generated the trace. The network context of §1 groups its factors into three layers that

we can name and, as §4 shows, largely control. They are the application and how a client drives it (e.g., video resolution, call mode), the transport and its congestion-control algorithm (e.g., TCP with CUBIC or BBR), and the network conditions (capacity, latency, queue discipline, cross-traffic). A representation is *sufficient* if it preserves the information needed to recover these factors, as far as the trace makes them recoverable at all. A *foundation* model must preserve all three layers, because different downstream tasks depend on different layers (e.g., traffic classification on the application, troubleshooting on the network conditions). A representation that keeps only one task's signal cannot serve the other tasks; it is the non-generalizing model that the field keeps rediscovering. Because we can set these factors as ground truth, sufficiency becomes *measurable*: one can check, factor by factor, what the embedding preserved, and a simple linear classifier gives a lower bound on it [8].

Finer context. Each layer holds finer factors. The application layer also includes the provider, the client's setup (e.g., device, browser), in-session state (e.g., the game in a cloud-gaming session [59] or the activity in a VR session [45]), and how the application adapts to the network (e.g., bitrate switching). Network conditions also span wired and wireless access and upstream paths outside the capture network (e.g., to a video server), beyond the emulated link we control (§4). Campus measurement nodes such as PINOT [6] capture some of these network-side factors today (e.g., wired and wireless campus access); modeling them fully is future work.

Minimality: discard the rest. A good representation also throws away what does not matter for the network context, such as collection artifacts (e.g., IP addresses or TTL values specific to where a trace was captured). Machine learning calls this goal the information bottleneck [55]: keep as little of the input as possible while keeping everything needed to predict a target. A summary that meets this goal is a minimal *sufficient statistic*. Prior work turns the goal into a training objective [2, 13]; we set the target to network context. We borrow the information bottleneck only as a guide for choosing what to optimize. Whether it also describes what happens inside trained networks is a separate, open question: studies disagree on whether deep networks go through a "compression phase" during training, and their results depend on the activation function and on how information is measured [4, 25, 50]. Minimality matters here because of shortcuts. A shortcut is a correlation that a model keeps but should not, such as a collection artifact that happens to track the label. A minimal representation has less spare capacity in which to store such correlations.

The objective. Putting the two requirements together, a good NFM *preserves as much network context as possible while using as little data and model capacity as possible*: a minimal sufficient representation of network context. This is the context-preservation objective of §1. It replaces three success criteria that the field uses implicitly. More data and bigger models are unsustainable, and higher in-distribution accuracy (accuracy on test data drawn from the same collection as the training data) can be inflated by shortcuts. The objective itself is not our contribution: it adapts prior work on intrinsic evaluation of NFMs [8] and on minimal sufficiency [13]. Our measurement in §3 applies it factor by factor.

Measuring context preservation. We measure what an embedding preserves with *probes*: simple linear classifiers trained on embeddings from the frozen NFM (its weights held fixed). Each per-factor probe predicts one factor's known setting (e.g., which congestion-control algorithm produced a trace), and one more probe predicts the full context at once. We call a probe's F_1 score (the harmonic mean of precision and recall) its *context* F_1. A model that keeps only the application scores high on the application probe and low on the others. The search we propose in §5 therefore targets the *weakest factor* (the one with the lowest per-factor F_1), so a factor the embedding lost cannot hide behind a high overall score. A high score shows only that a factor is easy to read out of the embedding; it does not prove that the encoder uses that factor [31]. To test whether the encoder relies on a factor, one can vary that factor alone and measure how far the embedding moves. Context F_1 rewards only sufficiency; minimality is enforced separately, by the objective's limits on data and model capacity. The rest of the paper asks how to *obtain* the data and the model that meet a given context-preservation objective.

3 The Data-Quality Premise

Before proposing how to obtain better data, we first establish that better data is worth obtaining. In one realistic NFM setting (one corpus, one model family, one curation method), data *quality* mattered more than quantity: a tenth of the corpus chosen by deduplication matched the whole corpus on overall context F_1. This deliberately narrow result supports our *premise* that better data can improve NFMs. It does not validate a data-generation method; generating data is the position we advance afterward.

Setup. We use netFound [9]. The full corpus holds 500k flows from 15-minute packet traces at a campus network's border router. From it we draw two 50k subsets, each exactly one-tenth of the corpus: one by random sampling, and one by semantic deduplication (SemDeDup [1]), which groups flows by their embeddings from a publicly available netFound checkpoint and drops near-duplicate flows within each group. Because the two subsets have the same size, any difference between them comes from *how* the flows were chosen. We pre-train netFound on each subset and compare both with netFound trained on the full corpus, using standard pre-training metrics and the context F_1 of §2. For the latter, we use NetReplica [15] to generate evaluation datasets of about 100 flows each that vary congestion control (BBR or CUBIC), queue discipline (drop-tail FIFO, which drops packets only when the buffer is full, or CoDel, which drops packets when queueing delay stays high), and cross-traffic (background or competing flows), and we train linear probes (logistic regression) on each model's frozen embeddings to recover those settings. No such labels enter pre-training, so a probe measures only what the encoder already kept; as §2 notes, this is a lower bound [31]. Here the probe serves only as a yardstick; §5 proposes using the same setup to choose training data.

Result. Table 1 reports the outcome. The deduplicated 50k set matches the full 500k corpus on context preservation (context F_1 of 0.974 for both): here, the right data did as well as ten times more data, consistent with broader results on curated data in machine learning [30, 41, 53]. It also beats the equal-size random subset on context preservation (0.974 vs. 0.947) and on every pre-training

Table 1: Curated data matches ten times more data. A 50k semantically deduplicated set matches the full 500k corpus on context F_1 (0.974) at one-tenth the size and beats a random 50k subset on every metric. Context F_1: one probe over all evaluation contexts (§2); MLM: masked-language modeling; dashes: pre-training metrics not available for the full-corpus model. Single run; seed variance not measured.

Pre-training set	Eval loss ↓	MLM acc. ↑	Perplexity ↓	Context F_1 ↑
Full (500k)	—	—	—	0.974
Random (50k)	5.811	0.581	334.0	0.947
SemDeDup (50k)	**5.213**	**0.605**	**183.6**	**0.974**

metric (all score predicting masked-out tokens of held-out traffic): evaluation loss (5.213 vs. 5.811), masked-language-modeling (MLM) accuracy (0.605 vs. 0.581), and perplexity (183.6 vs. 334.0). The per-factor probes for congestion control (0.710 vs. 0.688) and queue discipline (0.872 vs. 0.800) also favor the deduplicated set. The search of §5 would target the weakest factor; congestion control alone (0.710) already sits well below the overall 0.974. These are single-run numbers on one corpus and one model family: we draw no scaling law from them and do not claim that one factor is inherently harder to preserve. We did not test larger curated sets, and deduplication may change how often popular and rare traffic appear in the training data, which tasks such as traffic fingerprinting depend on.

Preliminary evidence from public NFMs. The same kind of probe shows that several public NFMs keep little of congestion control and cross-traffic. An intrinsic evaluation [8] trained probes on the embeddings of YaTC [66], ET-BERT [42], and NetMamba [57] to tell traces with a changed congestion-control algorithm or cross-traffic apart from a baseline in which nothing changed. Their F_1 scores fall between 0.24 and 0.48; guessing would score about 0.5 on these two-way tests. The models do better on queue discipline (0.51–0.70). netFound does better still (0.60 on congestion control, 0.78 on cross-traffic), though these per-factor tests differ from the evaluation behind its 0.95 in §1.

What this result supports. The deduplication result shows only that data is a lever worth pulling for NFMs; it does not show *how* to obtain the right data. Deduplication is selection, the easy special case of generation: it prunes flows the corpus already has and cannot add a condition the corpus lacks (e.g., a CoDel queue in a corpus captured only behind FIFO queues). Reaching such conditions requires generating them, cheaply enough to search over many candidates; §4 and §5 take this up.

4 The Generation Enabler

Adding the conditions a corpus lacks requires generating them, and the search of §5 needs that generation to be cheap and faithful. We argue that *disaggregation* (building an experiment from separate, reusable components) now makes it cheap, at least for the factors one can specify.

Composition of real components. Pramana [17] is a research backend that turns one experiment specification into labeled data by composing reusable components (Figure 2). An application driver runs real workflows, such as streaming a video, in a real client [16]. A substrate sets the bottleneck link's capacity and queue discipline and replays recorded cross-traffic through it [14, 15, 28]. Linux `tc/netem` sets link rate, delay, and topology [39], and a controller launches the components [7]. Each component runs production code instead of a model of it, so we expect fewer artifacts than a monolithic simulator adds (e.g., a TCP model that departs from the Linux stack). Components can still interact, for example by competing for the host's CPU [29], so we must measure how closely the composed setup matches deployment traces instead of assuming it. Every capture arrives with its full context as a ground-truth label, because the specification itself supplies the label. NetReplica, one of these components, generated the evaluation traces of §3.

Why disaggregation, and why now. Composing such experiments was impractical a decade ago, when researchers stitched experiments together by hand from separate tools and ad hoc scripts. Disaggregation changed the interface: the community has built and hardened these components separately, and because they stay separate and reusable, a new experiment should cost a specification instead of an engineering project. This drop in cost is recent.

Cheap generation is the search's precondition. The search of §5 repeats one step many times: generate data under a candidate specification, then measure how much context the encoder preserves. The search is tractable only if each step is cheap and faithful; cheap generation cuts the data cost of each step, and the specification supplies its labels. Networking can therefore attempt this search more readily than domains without a cheap, faithful generator for their deployment data.

Where generated traffic differs. Controllable, labeled generators are not unique to networking. Robotics creates training data through simulation and domain randomization [46, 56], graphics renders labeled scenes with pixel-exact ground truth [22, 26], and causal representation learning recovers factors from a known generative process [51]. Those fields generate a stand-in for deployment data and pay a sim-to-real or photorealism gap. We run the same protocol and application software as deployment, so the bytes and headers an NFM reads are produced the same way. Our *realism gap*, how far generated traces depart from deployment traces, lies instead in *timing and throughput*. Calibrated emulators match real paths' throughput and delay to within 17% on average [63], and record-and-replay reproduces web page-load times with a median error of 12.4% [47]. Container-based emulation stayed accurate at 100 Mbps but not at 1 Gbps in Mininet-HiFi's evaluation, once the host ran short of CPU [29]. Offloads, tunnels, and multi-hop routes can still change the packet-level dynamics of traffic, so the setup must be tuned to match deployment. Replayed cross-traffic comes from recorded traces and does not react to the workflows it competes with, and an emulated topology, even a multi-hop one, is not the wild Internet. The gap also affects evaluation. An encoder trained and scored only on generated traces could preserve the generator's regularities (e.g., the same host stack in every run) instead of network context, so probes should also run on deployment traces whose context is known. Quantifying and bounding this gap is itself part of the agenda. We claim only that it is smaller and cheaper to close than a sim-to-real gap; it cannot be eliminated.

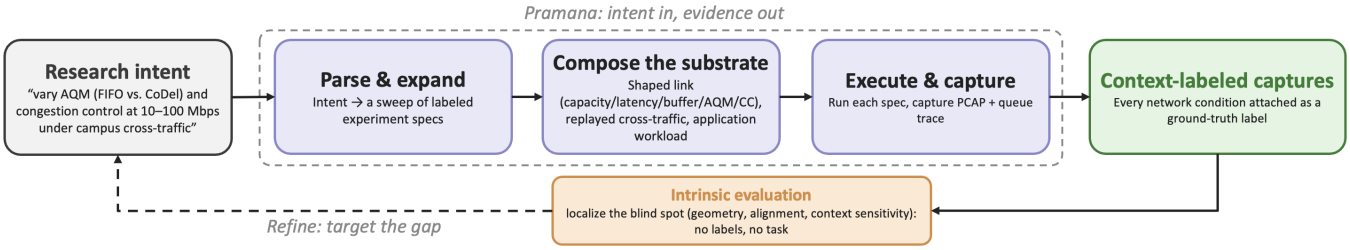


Figure 2: Disaggregation closes the data loop. Pramana expands a research intent into experiment specifications and composes the components each one needs (network conditions, replayed cross-traffic, application workflows). It runs each specification, capturing packet and queue traces in which *every specified network condition is a ground-truth label.* An intrinsic evaluation that needs no task labels finds the encoder's weakest factor, and the next intent targets it (dashed: the proposed refinement step). The researcher writes the first intent and reads the evaluation; the backend runs the steps in between.

5 A Guided Search

With better data as the lever (§3) and generation now cheap (§4), we cast obtaining data as a *search* over what to select and what to generate.

Selection and generation are one search. A better dataset can come from keeping the informative examples one already has or from generating the cases that are missing. We call the first *exogenous selection* (deduplication as in §3, coreset selection, coverage-based sampling [19]) and the second *endogenous generation* (synthesizing an unseen condition, application mix, or path). Exogenous selection removes redundancy but *cannot add a condition the corpus lacks*, and networking has no web-scale corpus to fall back on; endogenous generation can add it. Selection is a special case of generation: if we restrict the generator to emit only examples already in the corpus, with no transformation, generation reduces to selecting existing examples (*selection* $\subset$ *generation*).

Cheap generation makes choice the constraint. One can produce most contexts one can specify, such as an application, a transport, and a path's capacity and latency, up to a realism gap (§4). Generation is not solved, though: latent factors, such as how users behave inside a session, are hard to specify. Still, the main constraint is now *choice*: picking the contexts to add and the traces to keep, and deciding when to stop. Finding which context is missing has its own cost, and how expensive this search is remains an open question. The space of candidate specifications is vast, and sampling it blindly is little better than scraping the web: it wastes runs on contexts the encoder already preserves.

The search, made concrete. We propose to cast selection and generation as actions in one space; this search drives the data loop of Figure 1. A candidate c is a specification (§4) that sets the application, the congestion-control algorithm (e.g., CUBIC or BBR), and network conditions such as capacity and queue discipline (e.g., FIFO or CoDel). An action a is either $\textsc{Select}(B)$, keeping a block B of the current data (a cluster, a slice, or a condition already present), or $\textsc{Generate}(c)$, creating new traces under c. We score every action the same way, by the context preservation it adds per unit cost: $S(a) = \Delta\textsc{cp}(a) \,/\, \textsc{cost}(a)$. Here $\Delta\textsc{cp}(a)$ is the action's gain in the targeted factor's F_1 (§2), and $\textsc{cost}(a)$ counts data volume and compute. Because a full retrain per candidate is expensive, we would estimate $\Delta\textsc{cp}(a)$ with a quick per-factor probe after a short additional pre-training run, and check it against an occasional full retrain. The loop is greedy. It finds the encoder's weakest factor, takes the highest-S action for it, and re-measures, stopping when the best marginal S falls below a threshold. This greedy policy is a first cut; escaping local optima is future work. We have not yet tested this procedure. How to weigh data volume against compute in $\textsc{cost}(a)$, how short the pre-training run can be, and how closely its estimate tracks a full retrain are open design choices.

One turn of the loop. Suppose the per-factor probes show that queue discipline is the encoder's weakest factor: it cannot separate drop-tail FIFO from CoDel. Exogenous selection cannot help if the corpus holds only FIFO paths: no subset adds the missing contrast. Endogenous generation can. $\textsc{Generate}(c)$ would run a specification that holds the application, the cross-traffic, and the link capacity fixed and switches only the queue discipline, returning paired traces labeled by discipline. We would continue pre-training the encoder on them and re-measure the queue-discipline F_1. If it rises, the turn paid off. A score of the same form later governs how much model capacity to keep (§6).

Why other domains rely on proxies. A community without a faithful generator of its own data can only rank or filter the data it already has, using proxies for quality such as quality classifiers, similarity heuristics, and synthetic priors (§7). Communities that can generate, such as robotics and graphics, pay a transfer gap because their generated data stands in for deployment data (§4). Networking pays a smaller gap, *if* generation is faithful and cheap.

6 Proposal and Limits

We propose a method built from two loops (Figure 1): one chooses the training data, the other trims model capacity, and one context-preservation objective drives both.

Data distillation. The first loop runs the guided search of §5 under the context-preservation objective to build the smallest dataset on which a model still preserves network context. It keeps the informative part of what exists (exogenous selection) and generates the contexts that are missing (endogenous generation). Deduplication (§3) is a proof of concept for selection; generation in the loop, targeting the contexts a model measurably fails to preserve, remains untested.

Model distillation, verified. The second loop decides which model capacity to keep. Model distillation reliably preserves *downstream task accuracy* [32, 38, 49], but task accuracy does not measure context preservation: public NFMs that report high classification accuracy keep little of congestion control and cross-traffic (§3). A distilled model (the student) can fail to match the original (the teacher) even when it has the capacity to [54]. Compression silently degrades rare cases, shifted conditions, and knowledge-heavy tasks while top-line accuracy holds [33, 34, 36, 62, 64]. Whether a minimal sufficient subnetwork lives inside a large model also remains contested. Some studies find such subnetworks [11, 21]; others find that the pruned architecture, retrained from scratch, does as well [44], though not at scale [23]. The broader field checks what a compressed model preserves only indirectly, by comparing embeddings (e.g., with centered kernel alignment, CKA) or probing for properties that were not controlled. Networking can instead probe for the known factors that generated each trace: controlled generation yields paired traces that change one factor at a time, so distillation can be checked against those factors instead of done blind. Model actions mirror data actions: we score each candidate layer, head, or block by $S_{\text{model}} = \Delta\text{CP}/\Delta\text{FLOPs}$ (context preservation lost per unit of compute saved by pruning it) and prune the lowest-scoring first, so data and model become two coordinated loops under one objective.

What we show and what remains. We provide evidence for one claim: better-curated data preserves context at a fraction of the volume (§3). Generation served there only as the yardstick. We propose the rest: the guided search, including generating training data for the contexts a model misses, and the model-distillation loop, all still untested. We also argue, without yet measuring it, that cheap generation makes the search tractable (§4).

Limits. The premise rests on one campus corpus, one model family, and one curation method, and we have not yet run downstream or out-of-distribution tests to complement context preservation. The residual realism gap of emulation is real, and we have not yet quantified it. The information-bottleneck view is a lens whose theory for trained networks is contested [50]. We also leave uncertainty quantification and downstream explainability open, though per-factor context preservation already names which conditions a representation makes readable, a partial answer to *what did the model learn*.

7 Related Work

Network foundation models and their diagnosis. NFMs adapt language and vision pre-training to traffic: ET-BERT treats packets as language [42], YaTC treats them as images [66], PcapEncoder trains a language model to extract features from protocol headers [67], and netFound builds a network-native encoder [9]. A diagnostic line shows that these models often exploit shortcuts instead of traffic structure. Trustee traces a trained model's decisions to collection artifacts [35]. The Sugar study finds that reported accuracy owes much to shortcuts left by data preparation, with PcapEncoder the one model whose representation holds up [67]. A credibility-crisis argument generalizes the concern [60], and intrinsic evaluation probes what a pre-trained encoder learned [8]. These efforts diagnose or redesign the model, and architecture work filters the data by excluding leaky fields, but the contexts the training data covers stay fixed.

Data-centric learning and distillation. Machine learning repeatedly finds that data quality beats quantity, through deduplication [1, 30, 41], data pruning [53], dataset distillation [58], and curated-data models across NLP, vision, speech, and time series [5, 10, 24, 27, 48, 68]. Model distillation retains *task accuracy* [32, 38, 49] but can lose context the teacher preserved (§6). These methods curate or compress using proxies for quality, because their data carry no labels for the factors that generated each example. Networking can check what context survives on generated traces whose factors are known (§4).

Controllable data generation. Networking has controlled data-collection platforms: netUnicorn separates intent from execution [7], kernel emulation shapes links [39], NetGent automates real application workflows [16], and NetReplica sets up bottleneck links under replayed cross-traffic [14, 15]. Learned generators instead synthesize traffic statistically with GANs [43, 65], diffusion [37], or state-space models [12], or predict counterfactuals on fixed traces [3]. They do not run the deployment stack or target what a model fails to preserve, and they stay near their training distribution. Labeled generators in robotics and graphics pay a larger transfer gap (§4).

How this paper differs. Data-centric learning and closed-loop data collection are established in machine learning. We propose their networking instantiation: a generator that shares the deployment stack, steered by one context-preservation objective over both data and model. netUnicorn [7] assembles *labeled* data for a chosen task, whereas context preservation needs no task labels. We also propose using intrinsic evaluation [8], so far a diagnostic, to choose data. Architectural gains such as netFound's are complementary: a better encoder still learns only the contexts its training data covers.

8 Conclusion

We began from one narrow result: in the one setting we tested, a well-chosen tenth of the data matched the full corpus on context preservation. A representation's quality is the context it preserves, so obtaining better data is a search: keep the informative part of what exists and generate the missing contexts one can specify. Controlled generation, made cheap by composing real components, can make that search practical, and one context-preservation objective can decide both the data a model learns from and the capacity it keeps. The argument concerns the data, so it applies beyond any one model family. Networking keeps rediscovering that its models fail in the field. Because it can generate traffic through the stack it deploys, it can aim to build minimal sufficient NFMs and check what they preserve against ground truth. Our call is to make the data better before making the model bigger.

Acknowledgments

We thank the anonymous reviewers for their feedback. This work was supported by NSF awards 2443777 and 2323229 and by research gifts from Cisco and Google. This research used resources of NERSC, a DOE Office of Science User Facility, under Contract No. DE-AC02-05CH11231 (award DDR-ERCAP0029768).

References

[1] Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S. Morcos. 2023. SemDeDup: Data-Efficient Learning at Web-Scale Through Semantic Deduplication. *arXiv:2303.09540* (2023).

[2] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. 2017. Deep Variational Information Bottleneck. In *ICLR*.

[3] Abdullah Alomar, Pouya Hamadanian, Arash Nasr-Esfahany, Anish Agarwal, Mohammad Alizadeh, and Devavrat Shah. 2023. CausalSim: A Causal Framework for Unbiased Trace-Driven Simulation. In *USENIX NSDI*.

[4] Rana Ali Amjad and Bernhard C. Geiger. 2020. Learning Representations for Neural Network-Based Classification Using the Information Bottleneck Principle. *IEEE TPAMI* (2020).

[5] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. 2024. Chronos: Learning the Language of Time Series. *TMLR* (2024).

[6] Roman (Sylee) Beltiukov, Sanjay Chandrasekaran, Arpit Gupta, and Walter Willinger. 2023. PINOT: Programmable Infrastructure for Networking. In *ACM/IRTF ANRW*.

[7] Roman (Sylee) Beltiukov, Wenbo Guo, Arpit Gupta, and Walter Willinger. 2023. In Search of netUnicorn: A Data-Collection Platform to Develop Generalizable ML Models for Network Security Problems. In *ACM CCS*.

[8] Roman (Sylee) Beltiukov, Satyandra Guthula, Wenbo Guo, Walter Willinger, and Arpit Gupta. 2025. Demystifying Network Foundation Models. In *NeurIPS*.

[9] Roman (Sylee) Beltiukov, Satyandra Guthula, Haarika Manda, Jaber Daneshamooz, Wenbo Guo, Walter Willinger, Arpit Gupta, and Inder Monga. 2023. netFound: Principled Design for Network Foundation Models. *arXiv:2310.17025* (2023).

[10] Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. 2024. AlpaGasus: Training a Better Alpaca with Fewer Data. In *ICLR*.

[11] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The Lottery Ticket Hypothesis for Pre-trained BERT Networks. In *NeurIPS*.

[12] Andrew Chu, Xi Jiang, Shinan Liu, Arjun Nitin Bhagoji, Francesco Bronzino, Paul Schmitt, and Nick Feamster. 2025. NetSSM: Multi-Flow and State-Aware Network Trace Generation using State-Space Models. *arXiv:2503.22663* (2025).

[13] Milan Cvitkovic and Günther Koliander. 2019. Minimal Achievable Sufficient Statistic Learning. In *ICML*.

[14] Jaber Daneshamooz, Satyandra Guthula, Jessica Nguyen, William Chen, Sanjay Chandrasekaran, Ankit Gupta, Arpit Gupta, and Walter Willinger. 2026. NetReplica: A Programmable Substrate for Bottleneck-Centric Network Data Generation. *arXiv:2507.13476* (2026).

[15] Jaber Daneshamooz, Satyandra Guthula, Jessica Nguyen, William Chen, Ankit Gupta, Walter Willinger, and Arpit Gupta. 2026. NetReplica: A Bottleneck-Regime Substrate for Reproducible Network Data Generation. In *ACM CoNEXT Workshop on Network Traffic Classification – Reloaded (NTC-R)*.

[16] Jaber Daneshamooz, Eugene Vuong, Laasya Koduru, Sanjay Chandrasekaran, and Arpit Gupta. 2025. NetGent: Agent-Based Automation of Network Application Workflows. *arXiv:2509.00625* (2025).

[17] Jaber Daneshamooz, Eugene Vuong, Alagappan Ramanathan, Manni Moghimi, Haarika Manda, Satyam Kumar, Snithik Thode, Satyandra Guthula, Roman (Sylee) Beltiukov, Dongsu Han, et al. 2026. Pramana: A Composable, Domain-Specific Backend for Empirical Networking Research. In *ACM HotNets*.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE CVPR*.

[19] Alexander Dietmüller, Romain Jacob, and Laurent Vanbever. 2024. On Sample Selection for Continual Learning: A Video Streaming Case Study. *ACM SIGCOMM CCR* (2024).

[20] Epoch AI. 2024. Training Compute of Notable AI Systems Has Doubled Every Six Months Since 2010. with major processing by Our World in Data, https://ourworldindata.org/data-insights/since-2010-the-training-computation-of-notable-ai-systems-has-doubled-every-six-months.

[21] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. 2019. Stabilizing the Lottery Ticket Hypothesis. *arXiv:1903.01611* (2019).

[22] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. 2016. Virtual Worlds as Proxy for Multi-Object Tracking Analysis. In *CVPR*.

[23] Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The State of Sparsity in Deep Neural Networks. *arXiv:1902.09574* (2019).

[24] Sanchit Gandhi, Patrick von Platen, and Alexander M. Rush. 2023. Distil-Whisper: Robust Knowledge Distillation via Large-Scale Pseudo Labelling. *arXiv:2311.00430* (2023).

[25] Ziv Goldfeld, Ewout van den Berg, Kristjan Greenewald, Igor Melnyk, Nam Nguyen, Brian Kingsbury, and Yury Polyanskiy. 2019. Estimating Information Flow in Deep Neural Networks. In *ICML*.

[26] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J. Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam H. Laradji, Hsueh-Ti Derek Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Öztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. 2022. Kubric: A Scalable Dataset Generator. In *CVPR*.

[27] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. 2023. Textbooks Are All You Need. *arXiv:2306.11644* (2023).

[28] Satyandra Guthula, Jaber Daneshamooz, Charles Fleming, Kesheng Wu, Walter Willinger, and Arpit Gupta. 2026. NetBurst: Event-Centric Forecasting of Bursty, Intermittent Time Series. *arXiv:2510.22397* (2026).

[29] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz, and Nick McKeown. 2012. Reproducible Network Experiments Using Container-Based Emulation. In *ACM CoNEXT*.

[30] Nan He, Weichen Xiong, Hanwen Liu, Yi Liao, Lei Ding, Kai Zhang, Guohua Tang, Xiao Han, and Yang Wei. 2024. SoftDedup: An Efficient Data Reweighting Method for Speeding Up Language Model Pre-training. In *ACL*.

[31] John Hewitt and Percy Liang. 2019. Designing and Interpreting Probes with Control Tasks. In *EMNLP*.

[32] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. *arXiv:1503.02531* (2015).

[33] Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. 2019. What Do Compressed Deep Neural Networks Forget? *arXiv:1911.05248* (2019).

[34] Sara Hooker, Nyalleng Moorosi, Gregory Clark, Samy Bengio, and Emily Denton. 2020. Characterising Bias in Compressed Models. In *ICML Workshop*.

[35] Arthur Selle Jacobs, Roman (Sylee) Beltiukov, Walter Willinger, Ronaldo A. Ferreira, Arpit Gupta, and Lisandro Z. Granville. 2022. AI/ML for Network Security: The Emperor Has No Clothes. In *ACM CCS*.

[36] Ajay Jaiswal, Zhe Gan, Xianzhi Du, Bowen Zhang, Zhangyang Wang, and Yinfei Yang. 2024. Compressing LLMs: The Truth Is Rarely Pure and Never Simple. In *ICLR*.

[37] Xi Jiang, Shinan Liu, Aaron Gember-Jacobson, Arjun Nitin Bhagoji, Paul Schmitt, Francesco Bronzino, and Nick Feamster. 2024. NetDiffusion: Network Data Augmentation Through Protocol-Constrained Traffic Generation. *Proc. ACM Meas. Anal. Comput. Syst. (SIGMETRICS)* (2024).

[38] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for Natural Language Understanding. In *Findings of EMNLP*.

[39] Bob Lantz, Brandon Heller, and Nick McKeown. 2010. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks. In *ACM HotNets*.

[40] Franck Le, Mudhakar Srivatsa, Raghu K. Ganti, and Vyas Sekar. 2022. Rethinking Data-Driven Networking with Foundation Models: Challenges and Opportunities. In *ACM HotNets*.

[41] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating Training Data Makes Language Models Better. In *ACL*.

[42] Xinjie Lin, Gang Xiong, Gaopeng Gou, Zhen Li, Junzheng Shi, and Jing Yu. 2022. ET-BERT: A Contextualized Datagram Representation with Pre-training Transformers for Encrypted Traffic Classification. In *ACM WWW*.

[43] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. 2020. Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions. In *ACM IMC*.

[44] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2019. Rethinking the Value of Network Pruning. In *ICLR*.

[45] Minzhao Lyu, Rahul Dev Tripathi, and Vijay Sivaraman. 2023. MetaVRadar: Measuring Metaverse Virtual Reality Network Activity. *Proc. ACM Meas. Anal. Comput. Syst. (SIGMETRICS)* (2023).

[46] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. 2021. Isaac Gym: High Performance GPU-Based Physics Simulation for Robot Learning. In *NeurIPS Datasets and Benchmarks*.

[47] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: Accurate Record-and-Replay for HTTP. In *USENIX ATC*.

[48] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. 2024. DINOv2: Learning Robust Visual Features without Supervision. *TMLR* (2024).

[49] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. *arXiv:1910.01108* (2019).
[50] Andrew M. Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D. Tracey, and David D. Cox. 2018. On the Information Bottleneck Theory of Deep Learning. In *ICLR*.
[51] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. 2021. Toward Causal Representation Learning. *Proc. IEEE* (2021).
[52] Robin Sommer and Vern Paxson. 2010. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *IEEE S&P*.
[53] Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S. Morcos. 2022. Beyond Neural Scaling Laws: Beating Power Law Scaling via Data Pruning. In *NeurIPS*.
[54] Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A. Alemi, and Andrew Gordon Wilson. 2021. Does Knowledge Distillation Really Work?. In *NeurIPS*.
[55] Naftali Tishby and Noga Zaslavsky. 2015. Deep Learning and the Information Bottleneck Principle. In *IEEE ITW*.
[56] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 2017. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In *IEEE/RSJ IROS*.
[57] Tongze Wang, Xiaohui Xie, Wenduo Wang, Chuyi Wang, Youjian Zhao, and Yong Cui. 2024. NetMamba: Efficient Network Traffic Classification via Pre-training Unidirectional Mamba. In *IEEE ICNP*.
[58] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. 2018. Dataset Distillation. *arXiv:1811.10959* (2018).
[59] Yifan Wang, Minzhao Lyu, and Vijay Sivaraman. 2025. Games Are Not Equal: Classifying Cloud Gaming Contexts for Effective User Experience Measurement. In *ACM IMC*.
[60] Walter Willinger, Ronaldo A. Ferreira, Arpit Gupta, Roman (Sylee) Beltiukov, Satyandra Guthula, Lisandro Z. Granville, and Arthur Selle Jacobs. 2025. When Something Looks Too Good to Be True, It Usually Is! AI Is Causing a Credibility Crisis in Networking. *ACM SIGCOMM CCR* (2025).
[61] Duo Wu, Xianda Wang, Yaqi Qiao, Zhi Wang, Junchen Jiang, Shuguang Cui, and Fangxin Wang. 2024. NetLLM: Adapting Large Language Models for Networking. In *ACM SIGCOMM*.
[62] Canwen Xu, Wangchunshu Zhou, Tao Ge, Ke Xu, Julian McAuley, and Furu Wei. 2021. Beyond Preserved Accuracy: Evaluating Loyalty and Robustness of BERT Compression. In *EMNLP*.
[63] Francis Y. Yan, Jestin Ma, Greg D. Hill, Deepti Raghavan, Riad S. Wahby, Philip Levis, and Keith Winstein. 2018. Pantheon: The Training Ground for Internet Congestion-Control Research. In *USENIX ATC*.
[64] Lu Yin, Ajay Jaiswal, Shiwei Liu, Souvik Kundu, and Zhangyang Wang. 2024. Junk DNA Hypothesis: Pruning Small Pre-Trained Weights Irreversibly and Monotonically Impairs "Difficult" Downstream Tasks in LLMs. *arXiv:2310.02277* (2024).
[65] Yucheng Yin, Zinan Lin, Minhao Jin, Giulia Fanti, and Vyas Sekar. 2022. Practical GAN-based Synthetic IP Header Trace Generation using NetShare. In *ACM SIGCOMM*.
[66] Ruijie Zhao, Mingwei Zhan, Xianwen Deng, Yanhao Wang, Yijun Wang, Guan Gui, and Zhi Xue. 2023. Yet Another Traffic Classifier: A Masked Autoencoder Based Traffic Transformer with Multi-Level Flow Representation. In *AAAI*.
[67] Yuqi Zhao, Giovanni Dettori, Matteo Boffa, Luca Vassio, and Marco Mellia. 2025. The Sweet Danger of Sugar: Debunking Representation Learning for Encrypted Traffic Classification. In *ACM SIGCOMM*.
[68] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. LIMA: Less Is More for Alignment. In *NeurIPS*.
[69] Jiajun Zhou, Changhui Sun, Wentao Fu, Meng Shen, Shanqing Yu, and Qi Xuan. 2026. Traffic-MoE: A Sparse Foundation Model for Network Traffic Security Analysis. *arXiv:2601.00357* (2026).