Generating and Validating Synthetic Kernel Traces Using Diffusion Models

Yuvraj Sehgal | Sneh Patel | Mahsa Panahandeh | Prof. Naser Ezzati-Jivan | Francois Tetroe

The Problem & Motivation

Problems:

- Modern AIOps systems require high-fidelity kernel traces for:
- Scheduling decisions, memory allocations, I/O operations (microsecond precision)
- Training diagnostic and trace-driven ML models
- Root cause analysis and MTTR reduction

Three Key Barriers:

- **Production overhead**: Tracing adds 1.5–1.6× runtime cost → infeasible for latency-sensitive services
- **Privacy constraints**: Traces contain sensitive file paths, network endpoints → violate data retention policies
- **Long-tail diversity**: Real traces miss rare failure modes valuable for training

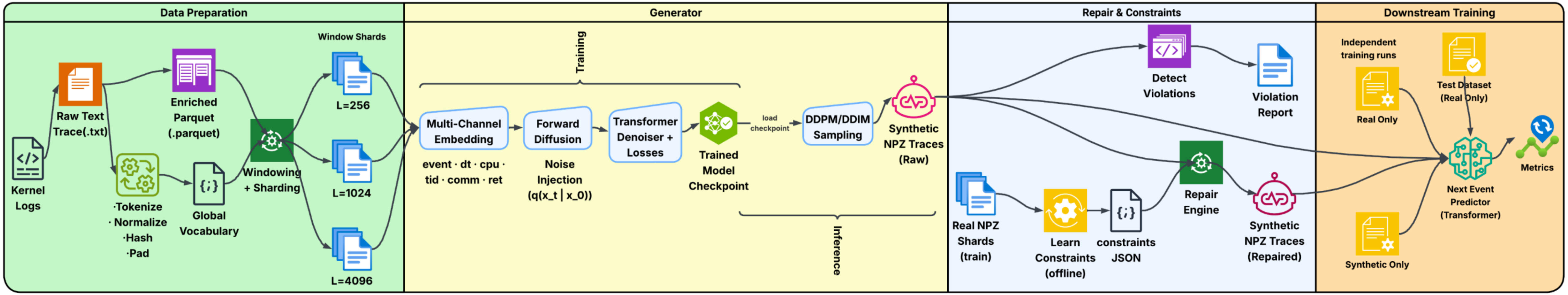
Why Existing Approaches Fall Short

Approach	Limitation	Impact
Statistical models (Markov chains)	Can't capture long-range dependencies or multi-attribute correlations	Locally valid but globally implausible
Rule-based generators	Require substantial domain expertise; don't generalize across workloads	Labor-intensive, brittle
GANs (SeqGAN, MaliGAN)	Violate chronology and event coherence even when syntactically correct	Unreliable semantic correctness

Experimental Setup

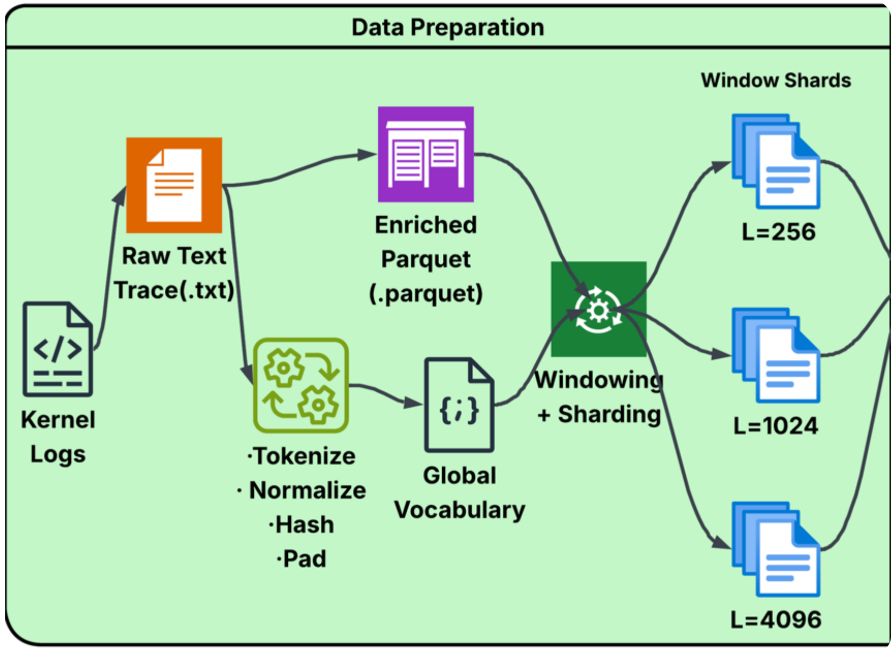
4 Stages

- Data Preparation
- Generation
- Repair and Constraints
- Downstream Training



Data Preprocessing

Channel	Collection Method	ID Assignment	Special Tokens	Vocab Size
event	Scan all traces, count frequency	Sort by frequency (0 → most common)	—	384
dt	Scan all traces	log(1+ Δt)	—	—
comm	Extract process names from Parquet	Sort by frequency, start at ID 2	<PAD>=0, <UNK>=1	123
ret	Extract return values, keep Top-K	Assign Top-1024 IDs from 2	<PAD>=0, <UNK>=1	1026
tid	Raw thread IDs	Hash to buckets: `tid % 256`	—	256 buckets
cpu	CPU core IDs	Direct encoding (0–3)	—	4



Diffusion Model Architecture (DDPM)

- **Core Idea**
  - Learn data distribution by **denoising noise → data**
  - Train to reverse a gradual **Gaussian noising process**
- **Forward (Noising) Process**
  - Add noise over T steps

$$x_t = \sqrt{\alpha_t}\, h_0 + \sqrt{1-\alpha_t}\, \epsilon, \epsilon \sim \mathcal{N}(0, I)$$

- **Reverse (Denoising) Process**
  - Neural network ϵ_θ predicts noise

$$\hat{\epsilon} = \epsilon_\theta(x_t, t)$$

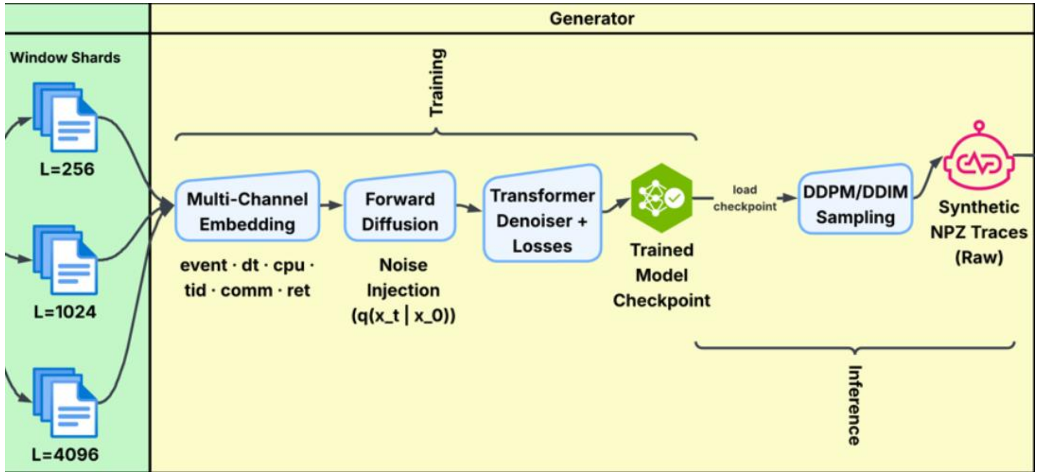
- Recover clean signal

$$\hat{x}_0 = \frac{x_t - \sqrt{1-\alpha_t}\, \hat{\epsilon}}{\sqrt{\alpha_t}}$$

- **Model Architecture**
  - Input: noisy sample x_t +timestep t
  - Backbone: **U-Net / Transformer**
  - Timestep embedding conditions the network
- **Training Objective**

$$\mathcal{L} = \mathbb{E}[\| \epsilon - \epsilon_\theta(x_t, t) \|^2]$$

- **Sampling**
  - Start from pure noise $x_T \sim \mathcal{N}(0, I)$
  - Iteratively denoise $T \to 0$



Repairing Synthetic Data

- Generative Model
  - Can be semantically incorrect
- Fix:
  - Invalid transitions
  - Temporal violations
  - Attribute inconsistencies
- 4 Classes of constraints from real shards:
  - **Event transitions**:
    - a directed graph G=(V,E), where $(e\_i,e\_j) \in E$ if e_j follows e_i in real traces
  - **Temporal bounds**:
    - min & max inter-event deltas per event type
  - **CPU affinity**:
    - allowed CPU sets per event type
  - **Attribute validity:**
    - Allowed values for tid, comm, and ret conditioned on event type.

Constraint-based distance metrics. We quantify synthetic trace validity using four distance metrics. *Transition distance* measures invalid event pairs:

$$D_{\text{trans}}(\hat{X}) = 1 - \frac{1}{|\hat{X}| - 1} \sum_t \mathbb{I}\left[(\hat{e}_t, \hat{e}_{t+1}) \in \mathcal{G}\right].$$

Temporal distance measures timing violations:

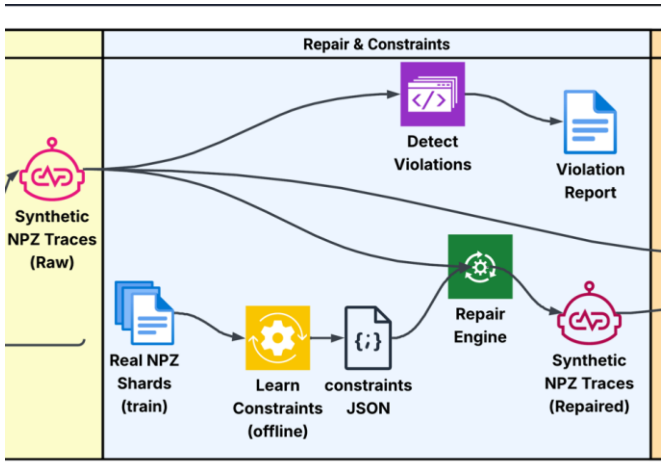
$$D_{\text{time}}(\hat{X}) = \frac{1}{|\hat{X}|} \sum_t \mathbb{I}\left[\Delta t_t \notin [\min_e, \max_e]\right].$$

CPU affinity distance measures invalid CPU assignments:

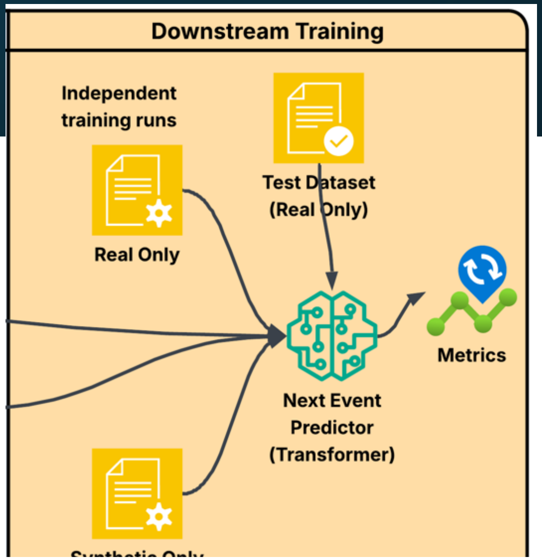
$$D_{\text{cpu}}(\hat{X}) = \frac{1}{|\hat{X}|} \sum_t \mathbb{I}\left[\hat{cpu}_t \notin C_{\hat{e}_t}\right].$$

Attribute validity distance aggregates categorical violations:

$$D_{\text{attr}}(\hat{X}) = \frac{1}{|\hat{X}|} \sum_t \mathbb{I}\left[\exists a \in \mathcal{A} : \hat{a}_t \notin \mathcal{V}^{(a)}_{\hat{e}_t}\right].$$



Downstream Task – Next Event Prediction



- What We're Testing:
  - Task: Next-event prediction (384-way classification)
  - Input: Sequence of 128 kernel events
  - Goal: Predict what event happens next
  - Test Set: Real data only (never seen before)
- Model Architecture:
  - Transformer encoder (4 layers, 8 heads, d_model=256)
  - Multi-channel inputs: Event type, timing, CPU, thread ID, command, return values
  - Training: 20 epochs with early stopping (patience=5)
- Metrics:
  - Primary: macro F1
  - Secondary: weighted F1, accuracy, and Top-K accuracy.

Config	Training Data	Purpose
Real-Only	100% real	Baseline performance
Combined (50/50) (Unrepaired)	100% synthetic + repair	Can synthetic replace real?
Combined (50/50) (Repaired)	50% real + 50% synthetic	Can augmentation help?

RQ1 - When Can Synthetic Traces Safely Augment Real Data?

Table 2: RQ1: Performance trade-offs when doubling the training dataset size using synthetic data. We compare training on real data (Real-only) with training on data composed of 50% real and 50% synthetic traces (Combined). ΔF1 reports the change in macro-F1 score introduced by synthetic augmentation across workloads and context lengths.

Benchmark	L=256			L=1024			L=4096		
	Real	Combined	ΔF1	Real	Combined	ΔF1	Real	Combined	ΔF1
ffmpeg	69.9%	32.0%	-37.9%	82.9%	60.1%	-22.8%	81.5%	64.4%	-17.1%
iozone	64.0%	19.9%	-44.1%	67.7%	34.8%	-32.9%	69.3%	40.8%	-28.5%
pybench	70.6%	41.8%	-28.8%	89.6%	69.7%	-19.9%	88.6%	*78.3%*	*-10.3%*
scimark2	72.0%	40.6%	-31.4%	88.5%	68.0%	-20.5%	89.8%	**87.2%**	**-2.6%**
stream	68.5%	17.6%	-50.9%	70.5%	40.7%	-29.8%	69.7%	44.9%	-24.8%
unpack-linux	63.4%	27.8%	-35.6%	69.1%	44.3%	-24.8%	—	43.8%	—
Average	**68.1%**	**30.0%**	**-38.1%**	**78.0%**	**52.9%**	**-25.1%**	**79.8%**	**59.9%**	**-17.7%**

Table 3: RQ1 (Secondary Metrics): Weighted F1, accuracy, and Top-K accuracy for the Combined (50% real + 50% synthetic) configuration across workloads and context lengths.

Benchmark	L=256				L=1024				L=4096			
	F1-W	Acc	Top-5	Top-10	F1-W	Acc	Top-5	Top-10	F1-W	Acc	Top-5	Top-10
ffmpeg	85.9%	86.6%	95.8%	97.4%	91.9%	92.1%	98.6%	99.2%	93.8%	93.9%	99.4%	99.7%
iozone	84.4%	84.7%	95.2%	96.9%	89.6%	89.7%	98.2%	99.1%	92.8%	92.9%	99.3%	99.6%
pybench	87.4%	87.8%	95.2%	96.6%	94.2%	94.3%	98.6%	99.2%	96.1%	96.2%	99.6%	99.8%
scimark2	87.0%	87.5%	95.1%	96.5%	93.8%	93.8%	98.5%	99.1%	96.9%	97.0%	99.7%	99.8%
stream	84.0%	84.5%	98.0%	98.5%	88.3%	88.4%	99.2%	99.5%	89.8%	89.9%	99.6%	99.8%
unpack-linux	85.3%	85.6%	95.1%	96.8%	90.5%	90.6%	98.2%	99.0%	92.9%	93.0%	99.3%	99.7%
Average	**85.7%**	**86.1%**	**95.7%**	**97.1%**	**91.4%**	**91.5%**	**98.5%**	**99.2%**	**93.7%**	**93.8%**	**99.5%**	**99.7%**

RQ2 - Does Constraint-Guided Repair Help?

Table 4: RQ2: Effect of constraint-guided repair across benchmarks and context lengths. We compare Combined (No Repair) and Combined (Repaired) configurations. ΔF1 reports the change in macro-F1 score introduced by applying constraint-guided repair.

Benchmark	L=256				L=1024				L=4096			
	No Rep.	Repaired	ΔF1	Rel.	No Rep.	Repaired	ΔF1	Rel.	No Rep.	Repaired	ΔF1	Rel.
ffmpeg	33.2%	32.0%	-1.2%	-3.6%	60.2%	60.1%	-0.1%	-0.2%	65.6%	64.4%	*-1.2%*	-1.8%
iozone	19.5%	19.9%	**+0.4%**	**+2.0%**	35.0%	34.8%	-0.2%	-0.6%	41.3%	40.8%	*-0.5%*	-1.2%
pybench	40.1%	41.8%	**+1.6%**	**+4.1%**	69.7%	69.7%	+0.0%	+0.0%	78.0%	78.3%	+0.3%	+0.3%
scimark2	38.9%	40.6%	**+1.7%**	**+4.3%**	67.7%	68.0%	+0.3%	+0.4%	87.0%	87.2%	+0.2%	+0.3%
stream	17.2%	17.6%	+0.3%	*+1.8%*	39.5%	40.7%	**+1.2%**	**+3.1%**	44.2%	44.9%	*+0.7%*	*+1.7%*
unpack-linux	27.4%	27.8%	+0.4%	*+1.4%*	43.9%	44.3%	+0.4%	*+1.0%*	58.0%	43.8%	-14.2%*	-24.6%*
Average	**29.4%**	**30.0%**	**+0.5%**	**+1.5%**	**52.7%**	**52.9%**	**+0.3%**	**+0.6%**	**62.4%**	**59.9%**	**-2.5%**	**-4.2%**

*Anomaly in unpack-linux L=4096; isolated outlier likely due to dataset or trace-specific irregularities.

RQ3 - How does increasing diffusion model context length improve synthetic data quality?

Table 5: RQ3: Effect of diffusion model context length on synthetic data quality. All results use the Combined (Repaired) configuration. ΔF1 denotes the absolute macro-F1 change from $L = 256$ to $L = 4096$, and Rel. Gain the corresponding relative improvement.

Benchmark	L=256	L=1024	L=4096	ΔF1 (256→4096)	Rel. Gain
ffmpeg	32.0%	60.1%	64.4%	*+32.3%*	*+101%*
iozone	19.9%	34.8%	40.8%	+20.9%	+105%
pybench	41.8%	69.7%	78.3%	*+36.5%*	*+87%*
scimark2	40.6%	68.0%	87.2%	**+46.6%**	**+115%**
stream	17.6%	40.7%	44.9%	+27.4%	+156%
unpack-linux	27.8%	44.3%	43.8%	+16.0%	+57%
Average	**30.0%**	**52.9%**	**59.9%**	**+29.9%**	**+104%**

RQ4 - Ablation study

Table 6: RQ4: Cross-model ablation results (macro-F1 %). Rows correspond to diffusion model feature sets and columns to downstream predictor features. All results use Combined (Repaired) with $L = 4096$. Bold indicates the best configuration per benchmark; *italic* indicates within 1% of best.

Benchmark	Diffusion Model	event	event+dt	event+dt+cpu+tid	all 6
ffmpeg	Base (2 ch)	60.6%	**61.8%**	—	—
	System (4 ch)	60.8%	*61.7%*	60.5%	—
	Full (6 ch)	60.8%	60.9%	59.7%	58.9%
pybench	Base (2 ch)	**71.3%**	70.6%	—	—
	System (4 ch)	70.3%	70.9%	*71.0%*	—
	Full (6 ch)	70.0%	*71.2%*	*71.2%*	70.6%
scimark2	Base (2 ch)	67.9%	68.5%	—	—
	System (4 ch)	67.8%	65.5%	67.0%	—
	Full (6 ch)	67.5%	68.9%	*68.8%*	**69.4%**

Discussion and Implications

Model Viability

Diffusion models can generate realistic system traces without explicit determinism

Performance degrades mainly when hidden external state dominates behavior

Design Implications

Temporal context is the primary driver of realism

Rich feature engineering provides diminishing returns

Simpler inputs with longer context are preferable

Learning & Repair

Models implicitly learn many system constraints at scale

Explicit repair mechanisms are most useful under uncertainty or limited context

System Integration

Suitable for fuzz testing and robustness evaluation

Enables privacy-preserving trace sharing

Effective for rare-event amplification and dataset balancing

Thank you