Short Paper: A Unified Edge Runtime for Multilingual NLP with Static and Distilled LLM Features

S Mahmudul Hasan
Tulane University
New Orleans, LA 70118, USA
shasan1@tulane.edu

Lu Peng
Tulane University
New Orleans, LA 70118, USA
lpeng3@tulane.edu

***Abstract*—Multilingual NLP on edge devices is often framed as a model-compression problem. However, deployed performance also depends on runtime-path organization, artifact residency, and wake overhead. We present a unified multilingual edge runtime that reuses the native tokenizer and frozen token-embedding table of a multilingual LLM as a shared, memory-mapped feature source. *Static* pools these frozen embeddings, while *Hybrid* adds lightweight contextual recovery distilled from the same LLM's contextual states without executing its full transformer backbone online. An optional entropy-routed *Cascade* provides request-level control between the two paths. We evaluate the runtime on a Jetson Orin NX across six languages and four NLP workloads using two complementary protocols. Protocol A isolates fixed-batch intrinsic pipeline cost, while Protocol B measures fixed-trace deployed behavior with lifecycle, residency, and wake effects. Relative to a routed per-language FastText deployment, *Static* and *Hybrid* improve monolingual throughput by 39.3% and 29.3%, reduce runtime energy by 32.2% and 23.7%, and lower P95 latency by 92.0% and 82.7%, respectively. Under multilingual serving, they improve throughput by 303.4% and 273.7%, reduce energy by 67.4% and 61.6%, and lower P95 latency by 99.0% and 97.8%, respectively. We also evaluate a consolidated FastText baseline that removes per-language artifact routing. Consolidation eliminates much of its wake overhead. However, *Static* retains a substantial per-request advantage, separating artifact consolidation from common-path simplification. *Static* anchors the low-cost operating point, while *Hybrid* preserves competitive quality on harder semantic tasks at substantially lower deployed cost than full multilingual LLM execution. These results show that multilingual edge efficiency depends on both model-level efficiency and deployed inference-path organization.**

***Index Terms*—multilingual NLP, edge computing, on-device inference, inference serving, runtime artifact organization, memory residency, energy efficiency**

I. Introduction

Multilingual NLP is increasingly deployed on edge devices, where latency, memory, and energy constraints must be balanced against diverse language inputs. Multilingual transformers provide strong cross-lingual representations [1], [2], [3]; however, repeated contextual-layer execution, large runtime state, and cold-start behavior can impose substantial edge cost [4], [5], [6]. Quantization and distillation reduce model size or computation [7], [8]; however, they generally preserve contextual execution as the online serving path. This leaves a complementary systems problem: how multilingual inference artifacts are organized, loaded, and accessed at runtime.

We therefore treat multilingual edge NLP as a runtime-path design problem. Our runtime consolidates inference around a frozen multilingual backbone's native tokenizer and memory-mapped embedding table. *Static* ($\mathcal{R}_{\mathrm{S}}$) pools frozen token embeddings with minimal online computation, while *Hybrid* ($\mathcal{R}_{\mathrm{H}}$) adds lightweight distilled contextual recovery without executing the full backbone. An optional entropy-routed *Cascade* ($\mathcal{R}_{\mathrm{Cas}}$) escalates uncertain requests from $\mathcal{R}_{\mathrm{S}}$ to $\mathcal{R}_{\mathrm{H}}$. This organization removes full contextual traversal from the common path while providing shared multilingual feature extraction through a single deployment artifact.

We evaluate the runtime on a Jetson Orin NX across six languages and four NLP workloads using two protocols. Protocol A isolates intrinsic fixed-batch pipeline cost, while Protocol B measures fixed-trace deployed behavior with loading, residency, wake, tail-latency, and energy effects. We compare against routed and consolidated FastText deployments and full contextual multilingual backbones, allowing us to separate artifact consolidation from per-request feature-path cost.

The results show substantial deployed gains. Relative to routed FastText, $\mathcal{R}_{\mathrm{S}}$/$\mathcal{R}_{\mathrm{H}}$ improve monolingual throughput by 39.3%/29.3%, reduce runtime energy by 32.2%/23.7%, and lower P95 latency by 92.0%/82.7%. Under multilingual serving, the corresponding improvements reach 303.4%/273.7% in throughput, 67.4%/61.6% in energy, and 99.0%/97.8% in P95 latency. Consolidating FastText removes much of its wake-path penalty, showing that artifact organization contributes substantially to deployment overhead. However, $\mathcal{R}_{\mathrm{S}}$ retains a substantial per-request advantage, with phase attribution showing that per-language tokenization accounts for much of the remaining baseline cost. $\mathcal{R}_{\mathrm{H}}$ preserves competitive quality on harder tasks, and when $\mathcal{R}_{\mathrm{S}}$ and $\mathcal{R}_{\mathrm{H}}$ diverge, $\mathcal{R}_{\mathrm{Cas}}$ recovers 94–96% of the quality gap while resolving 22–32% of requests on the cheap path.

Contributions:

- A unified multilingual edge runtime built around a shared tokenizer, memory-mapped embedding space, and lightweight

contextual recovery from frozen multilingual backbones.

- Two primary quality–efficiency operating points: *Static* for low-cost embedding lookup and *Hybrid* for contextual recovery, with an optional entropy-routed *Cascade* for deployment control.
- A protocol-separated methodology that distinguishes intrinsic fixed-batch cost from lifecycle-managed serving, including loading, residency, wake latency, tail latency, and runtime energy.
- A controlled Jetson Orin NX evaluation across six languages and four NLP workloads, separating artifact consolidation from common-path simplification and demonstrating substantial deployed efficiency gains while preserving competitive task quality.

II. Background and Related Work

Multilingual Representations. Classical representations such as Bag-of-Words and TF-IDF are inexpensive, although they provide limited semantic and contextual modeling. Dense embeddings such as FastText improve semantic structure while retaining low inference cost [9]. Transformers instead produce contextual representations through self-attention [10], and multilingual models such as mBERT and XLM-R learn shared cross-lingual spaces [1], [2]. However, online contextual execution increases latency, memory pressure, and energy on constrained edge devices [4].

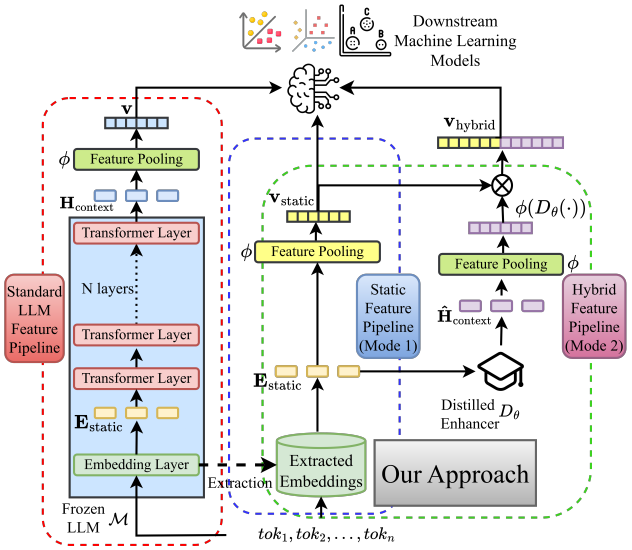
Edge-Efficient Inference. Quantization and distillation reduce model size or computation [7], [8], yet they generally retain contextual-layer execution on the online path. Systems work therefore targets complementary deployment costs. NNV12 reduces cold-start and loading overhead [5], STI trades memory for latency through sharding and pipelining [6], and EdgeMoE manages storage hierarchy and weight residency for on-device inference [11]. Together, these studies show that edge efficiency depends on runtime organization as well as model computation.

Multilingual Deployment. Classical multilingual embeddings commonly rely on language-specific resources [9], while models such as mBERT, LaBSE, and BGE-M3 consolidate multilingual representations into shared artifacts [1], [12], [13]. The former can introduce routing and artifact-management overhead, whereas the latter still requires online contextual execution.

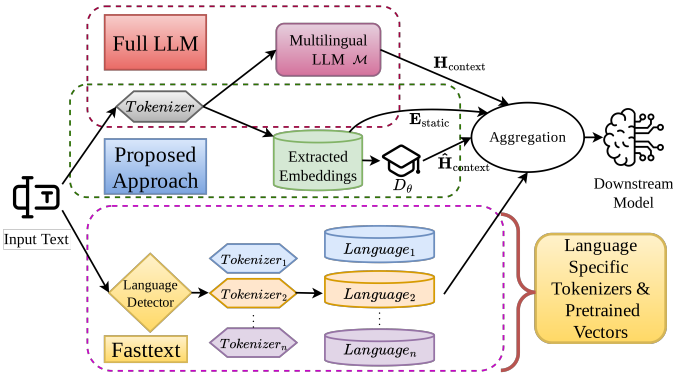
In an earlier study, we explored frozen multilingual LLM embeddings as a shared feature source for Static and Hybrid inference [14]. Here, we study their broader systems implications under lifecycle-managed deployment, including artifact consolidation, residency, wake overhead, and runtime control.

III. Unified Runtime Design

We design a multilingual edge runtime that shifts full transformer execution off the common path and consolidates inference around a shared multilingual artifact. The runtime exposes two primary modes—*Static* ($\mathcal{R}_{\mathrm{S}}$) and *Hybrid* ($\mathcal{R}_{\mathrm{H}}$)—as quality–efficiency operating points, with an optional entropy-routed *Cascade* ($\mathcal{R}_{\mathrm{Cas}}$) for deployment control.



(a) Feature extraction pathways.



(b) Deployment Packaging.

Fig. 1: Framework overview. **(a)** Feature extraction pathways, contrasting standard full-contextual execution with our embedding-first design. **(b)** Deployment packaging comparison. Unlike the routed FastText baseline, which uses language-specific memory-mapped vector resources selected through language-identification logic, our unified framework consolidates multilingual inference into a single shared tokenizer and deployable artifact.

A. Runtime Overview and Operating Modes

Notation. Let x denote an input sequence, $\mathcal{M}$ a frozen multilingual backbone, $\mathbf{E} \in \mathbb{R}^{n \times d}$ its token-embedding matrix, $\phi(\cdot)$ a pooling operator, and D_θ the contextual-recovery module. For $\mathcal{R}_{\mathrm{Cas}}$, $u(x)$ and τ denote uncertainty and routing threshold.

Fig. 1 summarizes the feature-extraction paths and deployment organization. Given x, the runtime applies the native tokenizer of $\mathcal{M}$ and retrieves $\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_n]$ from its frozen embedding table, stored as a memory-mapped feature source.

We use $\texttt{Mean}$ and $\texttt{MMM}(\mathbf{X}) = [\max_i \mathbf{x}_i; \mathrm{mean}_i \mathbf{x}_i; \min_i \mathbf{x}_i]$ pooling, where ; denotes concatenation. They yield d and $3d$ dimensions, respectively.

Static and Hybrid. The two primary feature representations are

$$\mathbf{v}_S = \phi(\mathbf{E}) \quad \text{and} \quad \mathbf{v}_H = [\phi(\mathbf{E}); \phi(D_\theta(\mathbf{E}))].$$

$\mathcal{R}_{\mathrm{S}}$ directly pools frozen embeddings, avoiding contextual backbone layers during inference. $\mathcal{R}_{\mathrm{H}}$ adds lightweight con-

textual recovery through D_θ, which maps static token embeddings $\mathbf{E}$ to approximate contextual states $D_\theta(\mathbf{E})$ without executing the full backbone. The hybrid representation has dimension $2d$ under `Mean` and $6d$ under `MMM`.

Cascade. $\mathcal{R}_{\mathrm{Cas}}$ provides request-level control between the cheap $\mathcal{R}_{\mathrm{S}}$ and richer $\mathcal{R}_{\mathrm{H}}$ paths. From the $\mathcal{R}_{\mathrm{S}}$-based classifier posterior $p_{\mathrm{cheap}}(y \mid x)$, we compute entropy $u(x) = -\sum_{y\in\mathcal{Y}} p_{\mathrm{cheap}}(y \mid x) \log p_{\mathrm{cheap}}(y \mid x)$. Requests with $u(x) \leq \tau$ exit through $\mathcal{R}_{\mathrm{S}}$; otherwise they escalate to $\mathcal{R}_{\mathrm{H}}$.

B. Artifact-Consolidated Multilingual Deployment

The runtime uses one shared multilingual path: all languages use the native tokenizer of $\mathcal{M}$ and retrieve features from its frozen embedding table, stored as a memory-mapped artifact for low-overhead access. As shown in Fig. 1b, the embeddings, D_θ, and routing state form a single deployable artifact shared by $\mathcal{R}_{\mathrm{S}}$, $\mathcal{R}_{\mathrm{H}}$, and $\mathcal{R}_{\mathrm{Cas}}$, avoiding per-language bundles and reducing dispatch overhead and artifact fragmentation.

For comparison, the routed FastText baseline ($\mathcal{R}_{\mathrm{FT}}$) performs language identification and dispatches requests to separate language-specific, memory-mapped FastText resources. Thus, the comparison captures routed per-language versus shared multilingual deployment. A consolidated FastText variant later separates artifact consolidation from per-request feature-path cost.

C. Lightweight Contextual-Recovery Module

$\mathcal{R}_{\mathrm{D}_\theta}$ denotes a distilled-only ablation using only $D_\theta(\mathbf{E})$. The $\mathcal{R}_{\mathrm{H}}$ and $\mathcal{R}_{\mathrm{D}_\theta}$ paths use D_θ to approximate contextual token states from frozen input embeddings. Given $\mathbf{E}$ and contextual states $\mathbf{H}_{\mathrm{context}}$ from $\mathcal{M}$, $D_\theta(\mathbf{E})$ predicts their token-level approximation.

Training minimizes masked mean squared error over non-padding positions:

$$\mathcal{L}_{\mathrm{MSE}} = \frac{\sum_i m_i \|\hat{\mathbf{h}}_i - \mathbf{h}_i\|_2^2}{\sum_i m_i},$$

where $\mathbf{h}_i$ is the contextual state from $\mathcal{M}$, $\hat{\mathbf{h}}_i$ its recovered approximation, and $m_i \in \{0, 1\}$ masks padding. Masking prevents padded positions from affecting the distillation objective.

D_θ is implemented as a lightweight 4-layer Transformer encoder with input down-projection, positional embeddings, stacked encoder blocks, and output projection to the contextual-state dimension.

Each D_θ is trained only on text from the sentiment and clustering training corpora. XNLI is excluded from distillation training and used only for downstream evaluation, providing a held-out test of whether the recovered contextual representation transfers beyond the training corpora. The same module supplies the contextual branch in $\mathcal{R}_{\mathrm{H}}$, the isolated branch in $\mathcal{R}_{\mathrm{D}_\theta}$, and the richer path activated by $\mathcal{R}_{\mathrm{Cas}}$.

D. Workload Heads and Training Policy

We use lightweight downstream learners: Gaussian Naïve Bayes, Random Forest, XGBoost, LightGBM, and compact MLP/ResidualMLP models for sentiment and XNLI, and `K-Means`/`K-Means++` for clustering. Each model family uses one fixed preset across methods, tasks, and languages.

TABLE I: Datasets used for sentiment, clustering, and XNLI.

Language	Sentiment	Clustering	XNLI
Arabic	LABR [17]	Sanad [18], [19]	Facebook XNLI [20]
Bengali	[21], [22]	Bangla News [23]	-
English	Amazon Reviews [24], [25]	AG News [26], [27]	Facebook XNLI [20]
Japanese	Amazon Reviews [24], [25]	Livedoor News [28]	-
Turkish	Movie Reviews [29]	Turkish News [30]	Facebook XNLI [20]
Chinese	Amazon Reviews [24], [25]	CLUE TNews [31]	Facebook XNLI [20]

IV. Experimental Methodology

We evaluate intrinsic pipeline efficiency and deployed multilingual serving across workloads, deployment artifacts, and lifecycle conditions.

A. Evaluation Questions

We study three systems questions: **RQ1: Intrinsic pipeline cost.** What are the fixed-batch latency, energy, EDP, and throughput characteristics of the compared feature paths? **RQ2: Deployed edge behavior.** How do the methods behave under fixed-trace serving with lifecycle, residency, wake, and tail-latency effects? **RQ3: Unified multilingual deployment.** Can a shared multilingual artifact reduce deployment fragmentation while preserving competitive task quality?

B. Edge Platform and Measurement Stack

Platform and execution. We use a Jetson Orin NX with an 8-core ARM Cortex-A78AE CPU, 1024-core NVIDIA Ampere GPU, 16 GB shared memory, JetPack 6.1, and Ubuntu 22.04 [15], [16]. Primary experiments use the fixed 25W power mode and run inside Docker. Downstream learners execute on the GPU; feature extraction is GPU-accelerated for $\mathcal{R}_{\mathrm{LLM}}$ and $\mathcal{R}_{\mathrm{H}}$, while $\mathcal{R}_{\mathrm{S}}$ uses CPU-side memory-mapped lookup with CPU-to-GPU transfer included in end-to-end measurements.

Measurement. Each recorded run follows an unprofiled warm-up. We collect `VDD_IN` telemetry with `tegrastats` and compute total energy as $\int P(t)\,dt$ by trapezoidal integration. Instantaneous power may vary under the fixed cap; a 40/25/15W sensitivity study is reported in Section IV-F.

C. Workloads and Language Coverage

We evaluate Arabic (ar), Bengali (bn), Chinese (zh), English (en), Japanese (ja), and Turkish (tr) on binary and ternary sentiment classification, text clustering, and XNLI (Table I).

We truncate inputs to 800 characters for sentiment, 2500 for clustering, and 1500 for XNLI; contextual pipelines process longer inputs chunk-wise and aggregate them. Sentiment and clustering use fixed stratified 75/25 train/test splits, while XNLI uses its native partitions and is evaluated on Arabic, Chinese, English, and Turkish. We use released labels without re-annotation, so cross-language quality reflects matched evaluation under the original annotations rather than culturally uniform ground truth. Protocol A covers sentiment and clustering across all evaluated configurations; Protocol B uses ternary sentiment for the primary deployed evaluation and adds XNLI and clustering in the extended study.

TABLE II: Backbones and distilled recovery modules.

Backbone	$\lvert\mathcal{M}\rvert$ (M)	$\lvert D_\theta\rvert$ (M)	η (%)	ρ
LaBSE [12]	470.927	0.921	99.804	0.91
mBERT [1]	177.853	0.921	99.482	0.85
E5-small [32]	117.654	0.724	99.385	0.92
E5-base [32]	278.044	0.921	99.669	0.90
XGLM-T [33]	564.464	1.052	99.814	0.95
BGE-M3 [13]	567.755	1.052	99.815	0.87
MiniLM [34]	117.654	0.724	99.385	0.91

TABLE III: On-disk artifact size versus language count.

Languages	$\mathcal{R}_{\mathrm{FT}}$ (GB)	$\mathcal{R}_{\mathrm{FT}}/\mathcal{R}_{\mathrm{S}}$	$\mathcal{R}_{\mathrm{FT}}/\mathcal{R}_{\mathrm{H}}$	$\mathcal{R}_{\mathrm{FT}}/\mathcal{R}_{\mathrm{LLM}}$
1	3.5	8.97×	8.54×	2.76×
3	12.6	32.31×	30.73×	9.92×
6 (deployed)	24.2	62.05×	59.02×	19.06×
30	108.0	276.92×	263.41×	85.04×
50	185.5	475.64×	452.44×	146.06×

D. Compared Deployment Artifacts

We compare routed and consolidated classical baselines, full contextual execution, and the unified runtime.

Routed FastText ($\mathcal{R}_{\mathrm{FT}}$). $\mathcal{R}_{\mathrm{FT}}$ performs language identification and dispatches requests to separate language-specific FastText resources. Each vector table is memory-mapped, avoiding text-vector loading as a confound while retaining per-language artifact routing and lifecycle.

Consolidated FastText ($\mathcal{R}_{\mathrm{FT}}^{\mathrm{trie}}$). $\mathcal{R}_{\mathrm{FT}}^{\mathrm{trie}}$ replaces the per-language tables with one memory-mapped MARISA-trie over the six-language union (11.4M tokens, 300-d). It removes language-conditional dispatch and per-language artifact lifecycle while retaining FastText embeddings and native word tokenizers, isolating the effect of artifact consolidation.

Full contextual execution ($\mathcal{R}_{\mathrm{LLM}}$). $\mathcal{R}_{\mathrm{LLM}}$ executes the multilingual backbones in Table II end-to-end in FP16, preserving full contextual-layer traversal. Lower-precision or compressed variants are complementary to the runtime-artifact comparison. The table also reports D_θ size, parameter compression $\eta = 100(1 - |D_\theta|/|\mathcal{M}|)$, and cosine-similarity fidelity ρ.

Unified runtime. We evaluate $\mathcal{R}_{\mathrm{S}}$, $\mathcal{R}_{\mathrm{H}}$, and $\mathcal{R}_{\mathrm{Cas}}$ as defined in Section III, with $\mathcal{R}_{\mathrm{D}_\theta}$ as a distilled-only ablation. Applicable methods use the same `Mean`/`MMM` pooling choices. The full backbone set in Table II is used for broad evaluation; Protocol B uses `mE5-S`, `mBERT-T`, `XGLM-T`, and `BGE-M3-S` to span the capacity range.

Artifact footprint. We measure on-disk artifact size as the supported language set grows. FastText artifact sizes are taken from the official tables, while $\mathcal{R}_{\mathrm{S}}$, $\mathcal{R}_{\mathrm{H}}$, and $\mathcal{R}_{\mathrm{LLM}}$ artifacts are measured directly. The unified artifacts remain fixed at 0.39, 0.41, and 1.27 GB, respectively.

E. Evaluation Setup and Protocols

We use the lightweight learners and fixed presets defined in Section III-D; task-quality distributions aggregate over downstream learners and are reported as box plots.

Protocol A: Fixed-Batch Intrinsic Cost. We run the raw-text-to-prediction pipeline at batch size 64 for each method–task–language setting. We aggregate applicable backbones, pooling choices, and downstream learners, reporting energy/inference, EDP/inference, and throughput without lifecycle effects.

TABLE IV: Protocol B trace configuration.

Parameter	Value
Requests	400
Batch sizes	50 requests each, 1–8 (1800 texts)
Request delays	340×0s, 40×2s, 20×25s (580s idle)
Cooldown	15s
Unload interval	1s

Protocol B: Fixed-Trace Deployed Runtime. Under Protocol B, we evaluate all methods on the same pre-generated replay trace combining bursts, idle periods, on-demand loading, and a 15s cooldown. Monolingual runs use language-specific traces, while multilingual runs merge all six languages ($n = 3060$). The primary evaluation uses ternary sentiment; the extended study reuses the trace for XNLI and clustering. For $\mathcal{R}_{\mathrm{Cas}}$, $\mathcal{R}_{\mathrm{S}}$ is loaded first and $\mathcal{R}_{\mathrm{H}}$ only on escalation. Table IV summarizes the lifecycle configuration.

F. Extended Single-Configuration Study

The primary evaluation spans multiple backbones, pooling choices, and downstream learners. The extended study instead fixes `BGE-M3-S` with `MMM` pooling and varies one factor at a time.

Task coverage. We replay the Protocol B trace for multilingual XNLI and clustering across $\mathcal{R}_{\mathrm{FT}}$, $\mathcal{R}_{\mathrm{FT}}^{\mathrm{trie}}$, $\mathcal{R}_{\mathrm{S}}$, $\mathcal{R}_{\mathrm{H}}$, and $\mathcal{R}_{\mathrm{LLM}}$ using length-representative corpus subsets.

Sensitivity and attribution. We evaluate three additional factors. First, we vary cooldown over $\{5,\ 15,\ 60\text{s},\ \text{always-resident}\}$ using a trace with varied idle intervals; because this trace differs from the primary replay, we report only relative changes. Second, we repeat a 48-cell evaluation at 40W, 25W, and 15W. Third, we instrument tokenization, embedding, feature extraction, and downstream prediction to attribute end-to-end latency.

G. Metrics and Implementation

We report weighted F1 for sentiment and XNLI and V-measure for clustering. Protocol A reports energy/inference, EDP/inference, and throughput; Protocol B reports throughput, P95 latency, runtime energy, RSS, residency/wake metrics, and $\mathcal{R}_{\mathrm{Cas}}$ cheap-exit behavior. Task quality is summarized with box plots, while system results report Q1/Median/Q3 or medians across configurations; extended single-configuration runs remain separate observations.

The pipeline uses Python 3.12 and Bash automation, with PyTorch/Hugging Face for contextual extraction, NumPy for embedding-first processing, and scikit-learn, PyTorch, XGBoost, and LightGBM for downstream models.

V. Evaluation Results

We evaluate intrinsic pipeline cost (Protocol A), deployed behavior under lifecycle management (Protocol B), multilingual deployment, and task-quality tradeoffs.

A. RQ1: Intrinsic Pipeline Cost

Protocol A isolates raw-text-to-prediction cost under fixed-batch execution without deployment lifecycle effects. Table V summarizes monolingual and multilingual efficiency.

TABLE V: Protocol A intrinsic efficiency (Q1/Median/Q3; brackets vs. $\mathcal{R}_{\mathrm{FT}}$).

Metric	Method	Mono. Classification	Mono. Clustering	Multi. Classification
Energy (J/inf) ↓	$\mathcal{R}_{\mathrm{FT}}$	0.171 / 0.221 / 0.443 [1.00×]	0.255 / 0.262 / 1.141 [1.00×]	0.789 / 0.793 / 0.801 [1.00×]
	$\mathcal{R}_{\mathrm{LLM}}$	0.183 / 0.299 / 0.631 [0.74×]	0.294 / 1.331 / 2.326 [0.20×]	0.150 / 0.334 / 0.937 [2.38×]
	$\mathcal{R}_{\mathrm{S}}$	0.005 / 0.006 / 0.007 [38.34×]	0.008 / 0.036 / 0.047 [7.23×]	0.006 / 0.007 / 0.008 [121.13×]
	$\mathcal{R}_{\mathrm{H}}$	0.110 / 0.117 / 0.125 [1.88×]	0.134 / 0.182 / 0.219 [1.44×]	0.093 / 0.100 / 0.106 [7.91×]
	$\mathcal{R}_{\mathrm{D}_\theta}$	0.108 / 0.114 / 0.117 [1.94×]	0.122 / 0.156 / 0.185 [1.68×]	0.092 / 0.096 / 0.101 [8.24×]
EDP (J·ms/inf) ↓	$\mathcal{R}_{\mathrm{FT}}$	4.578 / 7.608 / 30.711 [1.00×]	10.087 / 10.541 / 203.669 [1.00×]	96.886 / 98.140 / 99.634 [1.00×]
	$\mathcal{R}_{\mathrm{LLM}}$	3.258 / 8.262 / 35.392 [0.92×]	8.293 / 146.976 / 454.367 [0.07×]	2.409 / 9.604 / 72.355 [10.22×]
	$\mathcal{R}_{\mathrm{S}}$	0.003 / 0.005 / 0.008 [1496.47×]	0.009 / 0.200 / 0.329 [52.63×]	0.005 / 0.007 / 0.009 [14865.91×]
	$\mathcal{R}_{\mathrm{H}}$	1.426 / 1.607 / 1.796 [4.73×]	1.991 / 3.823 / 5.117 [2.76×]	0.938 / 1.076 / 1.238 [91.17×]
	$\mathcal{R}_{\mathrm{D}_\theta}$	1.353 / 1.474 / 1.566 [5.16×]	1.660 / 2.605 / 3.725 [4.05×]	0.876 / 0.964 / 1.058 [101.84×]
Throughput (inf/s) ↑	$\mathcal{R}_{\mathrm{FT}}$	14.421 / 29.069 / 37.282 [1.00×]	5.603 / 24.842 / 25.327 [1.00×]	8.025 / 8.082 / 8.149 [1.00×]
	$\mathcal{R}_{\mathrm{LLM}}$	17.825 / 36.460 / 54.372 [1.25×]	5.146 / 9.231 / 35.905 [0.37×]	12.943 / 34.769 / 61.795 [4.30×]
	$\mathcal{R}_{\mathrm{S}}$	933.592 / 1133.474 / 1408.402 [38.99×]	142.165 / 181.654 / 930.337 [7.31×]	848.155 / 994.282 / 1163.588 [123.03×]
	$\mathcal{R}_{\mathrm{H}}$	69.429 / 72.975 / 77.050 [2.51×]	41.612 / 48.544 / 66.455 [1.95×]	86.713 / 92.762 / 99.163 [11.48×]
	$\mathcal{R}_{\mathrm{D}_\theta}$	74.783 / 77.224 / 80.256 [2.66×]	48.403 / 58.875 / 74.733 [2.37×]	95.101 / 100.006 / 104.550 [12.37×]

Monolingual. For classification, $\mathcal{R}_{\mathrm{S}}$ reduces median energy and EDP by $38.34\times$ and $1496.47\times$ relative to $\mathcal{R}_{\mathrm{FT}}$ and increases throughput by $38.99\times$. $\mathcal{R}_{\mathrm{H}}$ occupies the intermediate-cost contextual point. Longer clustering inputs narrow the absolute gap but preserve the ordering: $\mathcal{R}_{\mathrm{S}}$ reduces energy and EDP by $7.23\times$ and $52.63\times$ and improves throughput by $7.31\times$.

Multilingual. The routed $\mathcal{R}_{\mathrm{FT}}$ path becomes substantially more expensive under multilingual execution, reaching 98.140 J·ms/inf median EDP and 8.08 inf/s. $\mathcal{R}_{\mathrm{S}}$ reduces median energy to 0.007 J/inf and reaches 994.28 inf/s, while $\mathcal{R}_{\mathrm{H}}$ reaches 92.76 inf/s with 1.076 J·ms/inf EDP. The shared feature path therefore provides an intrinsic advantage even without lifecycle effects.

B. RQ2: Deployed Edge Behavior

Protocol B evaluates the same feature paths under fixed-trace serving with loading, residency, wake, and tail-latency effects.

1) Monolingual Runtime and Lifecycle: Table VI combines core deployed metrics with lifecycle behavior for the same monolingual replay.

Relative to routed $\mathcal{R}_{\mathrm{FT}}$, $\mathcal{R}_{\mathrm{S}}$ increases median throughput by 39.3%, reduces runtime energy by 32.2%, lowers P95 latency by 92.0%, and reduces RSS by 30.3%. $\mathcal{R}_{\mathrm{H}}$ improves throughput by 29.3%, energy by 23.7%, and P95 latency by 82.7% while retaining contextual recovery. $\mathcal{R}_{\mathrm{LLM}}$ lowers latency but incurs the largest memory footprint.

Lifecycle behavior reinforces this separation. Wake P95 follows the same ordering: 1.04 s for $\mathcal{R}_{\mathrm{S}}$, about 2 s for $\mathcal{R}_{\mathrm{H}}$ and $\mathcal{R}_{\mathrm{LLM}}$, and 12.61 s for routed $\mathcal{R}_{\mathrm{FT}}$. $\mathcal{R}_{\mathrm{S}}$ also minimizes median loaded time at 419.2 s. Varying cooldown from 5 s to always-resident preserves the $\mathcal{R}_{\mathrm{S}} < \mathcal{R}_{\mathrm{H}} < \mathcal{R}_{\mathrm{FT}}$ tail-latency ordering, indicating that the result does not depend on the primary 15 s cooldown.

2) Deployed Phase Attribution: Table VII decomposes warm-request latency by pipeline stage.

Tokenization dominates routed $\mathcal{R}_{\mathrm{FT}}$, accounting for 61.9 of 72.9 ms per median request, while embedding lookup costs only 4.2 ms. The shared multilingual tokenizer used by the unified paths costs 1.3–1.5 ms. $\mathcal{R}_{\mathrm{S}}$ remains lightweight throughout the pipeline, while $\mathcal{R}_{\mathrm{H}}$ adds contextual recovery and $\mathcal{R}_{\mathrm{LLM}}$ spends most of its request time in contextual feature extraction. Within $\mathcal{R}_{\mathrm{S}}$, embedding cost primarily follows embedding width, ranging from 0.9 ms for MiniLM to 13.4 ms for BGE-M3.

3) Power-Envelope Sensitivity: Repeating the deployed evaluation at 40W, 25W, and 15W changes median latency, energy, RSS, and average power by less than 2%, with wake latency similarly stable. Average board draw remains about 6–7 W except for $\mathcal{R}_{\mathrm{LLM}}$ on long clustering inputs. The evaluated pipelines therefore remain primarily tokenizer- and memory-bound across these power settings.

4) Runtime Control with $\mathcal{R}_{\mathrm{Cas}}$: At deployed $\tau = 0.5$, $\mathcal{R}_{\mathrm{Cas}}$ resolves 26.1% of monolingual requests through $\mathcal{R}_{\mathrm{S}}$, reaching 2.47 text/s, 4174.05 J, 378.92 ms/text P95 latency, and 1768.62 MB RSS. Under multilingual serving, it resolves 13.0% cheaply and reaches 2.45 text/s, 4223.22 J, 384.22 ms/text P95, and 1840.59 MB RSS. Dynamic activation of $\mathcal{R}_{\mathrm{H}}$ keeps tail latency above always-on $\mathcal{R}_{\mathrm{H}}$ under this lifecycle-managed trace, so $\mathcal{R}_{\mathrm{Cas}}$ primarily provides request-level control rather than a direct tail-latency improvement.

Table VIII shows how routing behaves as the branch-quality gap changes.

Ternary sentiment provides little quality separation between $\mathcal{R}_{\mathrm{S}}$ and $\mathcal{R}_{\mathrm{H}}$, so weighted F1 remains nearly flat across intermediate thresholds. When the branches diverge more clearly, routing becomes more useful: on XNLI and clustering, $\mathcal{R}_{\mathrm{Cas}}$ recovers 94% and 96% of the $\mathcal{R}_{\mathrm{S}}$–$\mathcal{R}_{\mathrm{H}}$ quality gap while resolving 32% and 22% of requests through the cheap path. Its value therefore depends on both branch-quality separation and the cost of activating the richer path.

C. RQ3: Unified Multilingual Deployment

RQ3 evaluates the effect of replacing routed language-specific artifacts with a shared multilingual deployment. Table IX reports the primary ternary-sentiment evaluation and the fixed-configuration XNLI and clustering study. Because the panels use different aggregation schemes, comparisons should remain within each panel.

Under routed multilingual sentiment serving, $\mathcal{R}_{\mathrm{FT}}$ reaches 0.68 text/s with 12.15 s P95 latency. $\mathcal{R}_{\mathrm{S}}$ increases throughput by 303.4%, reduces P95 latency by 99.0%, energy by 67.4%,

TABLE VI: Protocol B monolingual runtime and lifecycle behavior. Core metrics are configuration medians; lifecycle metrics report Q1/Median/Q3. All methods incur 22 load/unload events.

	(a) Core runtime metrics				*(b) Lifecycle metrics*		
Method	**Text/s ↑**	**Energy (J) ↓**	**P95 (ms/text) ↓**	**RSS (MB) ↓**	**Loaded (s)**	**Sleep ratio**	**Wake P95 (s)**
$\mathcal{R}_{\mathrm{FT}}$	1.97	5377.26	1485.64	1196.80	420.6 / 431.5 / 496.8	0.489 / 0.508 / 0.532	12.33 / 12.61 / 13.02
$\mathcal{R}_{\mathrm{LLM}}$	2.37 [+20.1%]	4831.80 [-10.1%]	264.15 [-82.2%]	1814.34 [+51.6%]	456.5 / 505.4 / 529.1	0.327 / 0.344 / 0.352	1.40 / 2.01 / 2.60
$\mathcal{R}_{\mathrm{S}}$	2.74 [+39.3%]	3644.11 [-32.2%]	118.90 [-92.0%]	834.22 [-30.3%]	418.2 / 419.2 / 419.6	0.357 / 0.361 / 0.362	0.81 / 1.04 / 1.05
$\mathcal{R}_{\mathrm{H}}$	2.55 [+29.3%]	4101.17 [-23.7%]	256.37 [-82.7%]	1544.07 [+29.0%]	441.7 / 445.5 / 452.2	0.356 / 0.367 / 0.373	1.65 / 2.07 / 2.10

TABLE VII: Median warm-request phase latency (ms).

Backbone	**Mode**	**Tokenize**	**Embed**	**Feature**	**Predict**	**Total**
FastText	$\mathcal{R}_{\mathrm{FT}}$	**61.9**	4.2	69.3	6.2	72.9
BGE-M3	$\mathcal{R}_{\mathrm{S}}$	1.4	13.4	14.5	6.9	22.1
	$\mathcal{R}_{\mathrm{H}}$	1.5	47.1	48.1	8.1	56.2
	$\mathcal{R}_{\mathrm{LLM}}$	1.3	**72.1**	73.7	5.8	78.7
MiniLM	$\mathcal{R}_{\mathrm{S}}$	1.4	0.9	2.3	7.0	**9.3**

TABLE VIII: Cascade routing operating points.

(a) Ternary sentiment threshold frontier

τ	**Cheap-exit (%)**	**Weighted F1**
0.2	22.7	0.7742
0.5	48.8	0.7750
0.8	84.8	0.7747
1.0	100.0	0.7684

(b) Calibrated operating points

Task	$\mathcal{R}_{\mathrm{S}}$–$\mathcal{R}_{\mathrm{H}}$ **gap**	**Cheap-exit (%)**	**Gap recovered (%)**
Ternary sentiment	∼0.01 F1	64	≥100
XNLI	0.057 F1	32	94
Clustering	0.128 V-measure	22	96

and RSS by 72.0%. $\mathcal{R}_{\mathrm{H}}$ improves throughput by 273.7% while reducing P95 latency by 97.8% and energy by 61.6%. $\mathcal{R}_{\mathrm{LLM}}$ also avoids the routed artifact path, although it remains heavier than $\mathcal{R}_{\mathrm{S}}$ in energy and memory.

The fixed-configuration XNLI and clustering runs preserve the throughput advantage of the unified paths. Consolidating FastText into $\mathcal{R}_{\mathrm{FT}}^{\mathrm{trie}}$ sharply reduces the routed wake tail, lowering P95 from about 65.6 s to 0.587 s on XNLI and 3.35 s on clustering. $\mathcal{R}_{\mathrm{S}}$ still delivers the highest throughput on both tasks. These results separate two effects: artifact consolidation repairs much of the routed lifecycle path, while common-path simplification provides an additional per-request efficiency advantage.

Interpretation. Across Protocols A and B, shared multilingual execution reduces the cost associated with language-specific routing and fragmented deployment artifacts. The consolidated $\mathcal{R}_{\mathrm{FT}}^{\mathrm{trie}}$ control confirms that artifact organization explains part of the routed baseline's overhead, while the remaining differences arise from the feature path itself.

D. Quality Retention and Runtime Tradeoffs

We next examine whether the runtime operating points retain useful downstream quality.

1) Monolingual Task Quality: Figure 2 shows that quality is similar across methods on sentiment: $\mathcal{R}_{\mathrm{H}}$ reaches 0.90 median F1 on binary sentiment and 0.74 on ternary sentiment, close to $\mathcal{R}_{\mathrm{LLM}}$ at 0.91 and 0.75. Context becomes more important on harder tasks. For clustering, $\mathcal{R}_{\mathrm{H}}$ reaches 0.42 V-measure versus 0.31 for $\mathcal{R}_{\mathrm{S}}$ and 0.46 for $\mathcal{R}_{\mathrm{LLM}}$; on XNLI, it reaches

TABLE IX: Protocol B multilingual deployed runtime. Brackets are relative to $\mathcal{R}_{\mathrm{FT}}$ within each panel. Panel (b) reports single observed points on length-representative corpus subsets.

(a) Ternary sentiment — configuration medians

Method	**Throughput ↑**	**P95 (ms/text) ↓**	**Energy (J) ↓**	**RSS (MB) ↓**
$\mathcal{R}_{\mathrm{FT}}$	0.68	12153.75	11173.69	3005.56
$\mathcal{R}_{\mathrm{LLM}}$	2.34 [+244.0%]	316.68 [-97.4%]	4998.60 [-55.3%]	1776.57 [-40.9%]
$\mathcal{R}_{\mathrm{S}}$	2.74 [+303.4%]	123.92 [-99.0%]	3641.24 [-67.4%]	841.72 [-72.0%]
$\mathcal{R}_{\mathrm{H}}$	2.54 [+273.7%]	263.90 [-97.8%]	4295.06 [-61.6%]	1614.55 [-46.3%]

(b) XNLI and clustering — fixed `BGE-M3-S/MMM` configuration

Task	**Method**	**Throughput ↑**	**P95 (ms/text) ↓**
XNLI	$\mathcal{R}_{\mathrm{FT}}$	0.85	65586
	$\mathcal{R}_{\mathrm{FT}}^{\mathrm{trie}}$	2.57 [+202.4%]	587 [-99.1%]
	$\mathcal{R}_{\mathrm{S}}$	2.77 [+225.9%]	2670 [-95.9%]
	$\mathcal{R}_{\mathrm{H}}$	2.34 [+175.3%]	5700 [-91.3%]
	$\mathcal{R}_{\mathrm{LLM}}$	2.36 [+177.6%]	4028 [-93.9%]
Clustering	$\mathcal{R}_{\mathrm{FT}}$	0.73	65611
	$\mathcal{R}_{\mathrm{FT}}^{\mathrm{trie}}$	1.74 [+138.4%]	3352 [-94.9%]
	$\mathcal{R}_{\mathrm{S}}$	2.75 [+276.7%]	2671 [-95.9%]
	$\mathcal{R}_{\mathrm{H}}$	2.35 [+221.9%]	5725 [-91.3%]
	$\mathcal{R}_{\mathrm{LLM}}$	1.39 [+90.4%]	4393 [-93.3%]

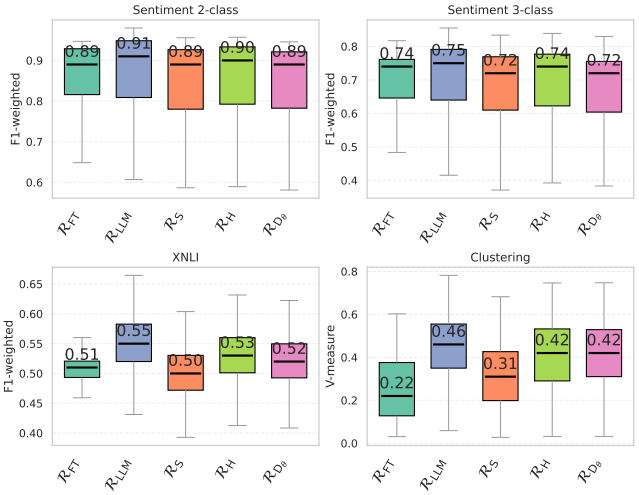


Fig. 2: Monolingual task-level comparison across feature pipelines (binary sentiment, ternary sentiment, clustering, and XNLI). Medians are annotated above the center lines.

0.53 F1 versus 0.50 and 0.55, respectively. Thus, $\mathcal{R}_{\mathrm{H}}$ recovers much of the contextual quality lost by the static path.

2) Multilingual Task Quality: Figure 3 shows the same pattern under multilingual evaluation. $\mathcal{R}_{\mathrm{H}}$ reaches median F1 of 0.88 on binary sentiment, 0.72 on ternary sentiment, and 0.54 on XNLI, remaining close to $\mathcal{R}_{\mathrm{LLM}}$ at 0.89, 0.73, and 0.55. Because XNLI is excluded from D_θ training, this result also measures held-out task transfer. The $\mathcal{R}_{\mathrm{D}_\theta}$ ablation improves over $\mathcal{R}_{\mathrm{S}}$ on multilingual XNLI, showing that the distilled branch contributes contextual information even without XNLI supervision.

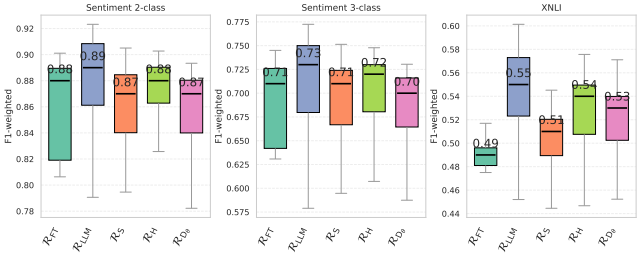


Fig. 3: Multilingual task-level comparison under the shared deployment artifact. Medians are annotated above the center lines.

Generality across backbones. Across all seven contextual backbones, the overall $\mathcal{R}_{\mathrm{FT}} \lesssim \mathcal{R}_{\mathrm{S}} \lesssim \mathcal{R}_{\mathrm{H}} \lesssim \mathcal{R}_{\mathrm{LLM}}$ quality ordering remains consistent. The $\mathcal{R}_{\mathrm{S}}$–$\mathcal{R}_{\mathrm{H}}$ gap varies by backbone, becoming largest on clustering for sentence-embedding backbones, which also explains when $\mathcal{R}_{\mathrm{Cas}}$ has useful quality headroom.

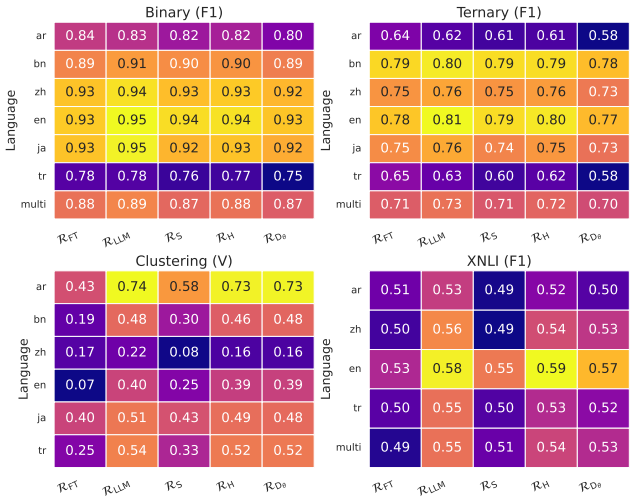


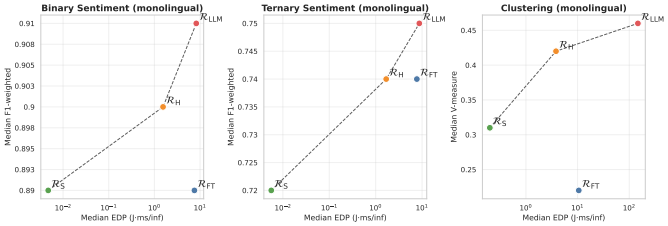
Fig. 4: Per-language performance evaluating single-language learners alongside a consolidated model ('multi'). $\mathcal{R}_{\mathrm{H}}$ consistently bridges the gap between static baselines and the heavy $\mathcal{R}_{\mathrm{LLM}}$ upper bound.

3) Per-Language Behavior and Ablation: Figure 4 shows similar trends across individual languages. For example, on Arabic clustering, $\mathcal{R}_{\mathrm{H}}$ reaches 0.73 V-measure versus 0.74 for $\mathcal{R}_{\mathrm{LLM}}$, 0.58 for $\mathcal{R}_{\mathrm{S}}$, and 0.43 for $\mathcal{R}_{\mathrm{FT}}$. The distilled-only $\mathcal{R}_{\mathrm{D}_\theta}$ ablation provides limited gains on simpler sentiment tasks but captures much of the contextual improvement on harder semantic tasks, while the fused $\mathcal{R}_{\mathrm{H}}$ representation remains the more consistent operating point.

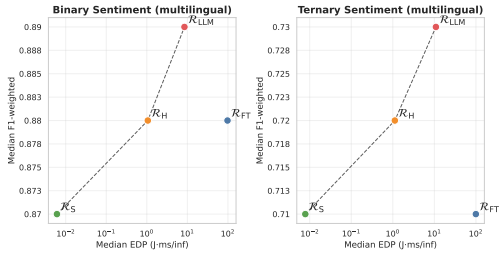
4) Quality–Efficiency Frontier: Figure 5 combines task quality with Protocol A EDP. $\mathcal{R}_{\mathrm{S}}$ anchors the low-cost region, $\mathcal{R}_{\mathrm{LLM}}$ the contextual-quality ceiling, and $\mathcal{R}_{\mathrm{H}}$ occupies the intermediate region, often approaching $\mathcal{R}_{\mathrm{LLM}}$ quality at substantially lower cost. The routed $\mathcal{R}_{\mathrm{FT}}$ baseline is frequently dominated in both quality and efficiency.

VI. Discussion

Runtime operating points. The results show that multilingual edge NLP depends on runtime organization as well as model-level efficiency. $\mathcal{R}_{\mathrm{S}}$ provides the lowest-cost path by avoiding



(a) Monolingual setting.



(b) Multilingual setting.

Fig. 5: Protocol A quality–efficiency frontiers. $\mathcal{R}_{\mathrm{S}}$ anchors the low-cost extreme, $\mathcal{R}_{\mathrm{LLM}}$ defines the heavy contextual ceiling, and $\mathcal{R}_{\mathrm{H}}$ occupies the balanced space. The $\mathcal{R}_{\mathrm{FT}}$ baseline is frequently dominated.

online contextual execution, while $\mathcal{R}_{\mathrm{H}}$ recovers contextual signal at bounded additional cost. $\mathcal{R}_{\mathrm{Cas}}$ adds request-level control when the two paths differ in quality: on XNLI and clustering, it recovers 94–96% of the $\mathcal{R}_{\mathrm{S}}$–$\mathcal{R}_{\mathrm{H}}$ gap while resolving 22–32% of requests through $\mathcal{R}_{\mathrm{S}}$. Under our lifecycle-managed trace, however, dynamic $\mathcal{R}_{\mathrm{H}}$ activation limits its tail-latency benefit. Its deployed value therefore depends on both branch-quality separation and residency policy.

Artifact consolidation and baselines. Artifact organization is a substantial deployment cost. Consolidating routed $\mathcal{R}_{\mathrm{FT}}$ into $\mathcal{R}_{\mathrm{FT}}^{\mathrm{trie}}$ removes much of its wake and memory overhead, confirming that fragmentation explains a substantial share of its deployed cost. However, $\mathcal{R}_{\mathrm{S}}$ still leads after consolidation, while phase attribution identifies per-language tokenization as a remaining source of classical-path latency. The gains therefore separate into artifact consolidation and common-path simplification. $\mathcal{R}_{\mathrm{LLM}}$ provides the full contextual comparison in FP16; lower precision may narrow absolute gaps without removing contextual execution from the online path.

Scope and interpretation. The runtime targets multilingual edge services with changing language mixes or shared infrastructure; fixed-language deployments with permanently resident resources may favor specialized implementations. System measurements come from one Jetson Orin NX, so absolute magnitudes remain device- and workload-specific. The extended XNLI, clustering, cooldown, power, and phase-attribution studies use a fixed `BGE-M3-S/MMM` configuration and are reported separately from configuration-level medians. Broader device classes and request patterns remain future work. Finally, task labels come directly from the released datasets, so cross-language quality results characterize relative feature-path behavior under the original annotations.

VII. Conclusion

We presented a unified multilingual edge runtime built around a shared tokenizer, memory-mapped embedding space, and lightweight contextual recovery. Across six languages and four NLP workloads, the Static path provides the lowest deployed cost, while the Hybrid path recovers much of the quality of full contextual execution at substantially lower runtime cost. Comparisons with routed and consolidated FastText deployments show that these gains arise from two complementary effects: artifact consolidation and common-path simplification. The results also show that runtime control can selectively trade efficiency for richer contextual processing when task quality benefits from it. Overall, multilingual edge NLP efficiency depends not only on model-level optimization, but also on how inference artifacts, feature paths, and lifecycle behavior are organized at deployment time.

Acknowledgment

This work is supported in part by an NSF Grant 2618897. We appreciate the computational resources provided by the Louisiana Optical Network Infrastructure (LONI) and valuable suggestions from anonymous reviewers.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[2] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," *arXiv preprint arXiv:1911.02116*, 2019.

[3] M. M. Rahman, Y. Watanobe, S. R. Naqvi, and L. Peng, "Intuitivegraphllm: Intuitive graph-based text representation with large language model," in *Proceedings of the AAAI Symposium Series*, vol. 8, no. 1, 2026, pp. 531–540.

[4] S. Laskaridis, K. Katevas, L. Minto, and H. Haddadi, "Melting point: Mobile evaluation of language transformers," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, 2024, pp. 890–907.

[5] R. Yi, T. Cao, A. Zhou, X. Ma, S. Wang, and M. Xu, "Boosting dnn cold inference on edge devices," in *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services*, 2023, pp. 516–529.

[6] L. Guo, W. Choe, and F. X. Lin, "Sti: Turbocharge nlp inference at the edge via elastic pipelining," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 2023, pp. 791–803.

[7] E. Frantar, S. Ashkboos, T. Hoefler, and D.-A. Alistarh, "Optq: Accurate post-training quantization for generative pre-trained transformers," in *11th International Conference on Learning Representations*, 2023.

[8] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[9] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[11] R. Yi, L. Guo, S. Wei, A. Zhou, S. Wang, and M. Xu, "Edgemoe: Empowering sparse large language models on mobile devices," *IEEE Transactions on Mobile Computing*, 2025.

[12] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang, "Language-agnostic bert sentence embedding," in *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: Long papers)*, 2022, pp. 878–891.

[13] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu, "Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation," *arXiv preprint arXiv:2402.03216*, 2024.

[14] S. M. Hasan and L. Peng, "The edge-first feature extractor: Enabling efficient multilingual nlp with static and distilled llm features," in *Proceedings of the 41st ACM/SIGAPP Symposium on Applied Computing*, 2026, pp. 720–721.

[15] P. SK, S. A. Kesanapalli, and Y. Simmhan, "Characterizing the performance of accelerated jetson edge devices for training deep learning models," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 6, no. 3, pp. 1–26, 2022.

[16] S. P. Baller, A. Jindal, M. Chadha, and M. Gerndt, "Deepedgebench: Benchmarking deep neural networks on edge devices," in *2021 IEEE international conference on cloud engineering (IC2E)*. IEEE, 2021, pp. 20–30.

[17] M. Aly and A. Atiya, "Labr: A large scale arabic book reviews dataset," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2013, pp. 494–498.

[18] O. Einea, A. Elnagar, and R. Al Debsi, "Sanad: Single-label arabic news articles dataset for automatic text categorization," *Data in brief*, vol. 25, p. 104076, 2019.

[19] H. Hermassi, "Arabic news articles dataset," https://www.kaggle.com/datasets/haithemhermessi/sanad-dataset, n.d., accessed: 2025-09-13.

[20] A. Conneau, R. Rinott, G. Lample, A. Williams, S. Bowman, H. Schwenk, and V. Stoyanov, "Xnli: Evaluating cross-lingual sentence representations," in *Proceedings of the 2018 conference on empirical methods in natural language processing*, 2018, pp. 2475–2485.

[21] J. Biswas and M. N. Hasan, "Bangla sentiment dataset," 06 2025.

[22] S. Sazzed and S. Jayarathna, "A sentiment classification in bengali and machine translated english corpus," in *2019 IEEE 20th international conference on information reuse and integration for data science (IRI)*. IEEE, 2019, pp. 107–114.

[23] Z. A. Nazi, "Bangla newspaper dataset," 2020. [Online]. Available: https://www.kaggle.com/dsv/1576225

[24] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "Mteb: Massive text embedding benchmark," *arXiv preprint arXiv:2210.07316*, 2022. [Online]. Available: https://arxiv.org/abs/2210.07316

[25] P. Keung, Y. Lu, G. Szarvas, and N. A. Smith, "The multilingual amazon reviews corpus," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020. [Online]. Available: https://arxiv.org/abs/2010.02573

[26] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*, vol. 28, 2015. [Online]. Available: https://papers.nips.cc/paper/2015

[27] SetFit / Hugging Face Datasets, "Setfit ag news dataset," https://huggingface.co/datasets/SetFit/agnews, 2023, accessed: 2025-09-13.

[28] shunk031, "Livedoor news corpus," https://huggingface.co/datasets/shunk031/livedoor-news-corpus, n.d., accessed: 2025-09-13.

[29] S. Yıldırım and B. Diri, "Turkish movie sentiment analysis dataset," https://www.kaggle.com/datasets/mustfkeskin/turkish-movie-sentiment-analysis-dataset, n.d., accessed: 2025-09-13.

[30] S. Yıldırım and T. Yıldız, "A comparison of different approaches to document representation in turkish language," *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, vol. 22, no. 2, pp. 569–576, 2018.

[31] L. Xu, H. Hu, X. Zhang, L. Li, C. Cao, Y. Li, Y. Xu, K. Sun, D. Yu, C. Yu *et al.*, "Clue: A chinese language understanding evaluation benchmark," *arXiv preprint arXiv:2004.05986*, 2020.

[32] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei, "Multilingual e5 text embeddings: A technical report," *arXiv preprint arXiv:2402.05672*, 2024.

[33] X. V. Lin, T. Mihaylov, M. Artetxe, T. Wang, S. Chen, D. Simig, M. Ott, N. Goyal, S. Bhosale, J. Du *et al.*, "Few-shot learning with multilingual generative language models," in *Proceedings of the 2022 conference on empirical methods in natural language processing*, 2022, pp. 9019–9052.

[34] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.