**TECHNICAL REPORT**

NO. 26-SN01

AUG 2026

# SNAP: Semantic Normalization via Attached Probes for Multilingual Real-Time Text-to-Speech

**JaiHyuk Lee**
Independent Researcher
softguy@gmail.com

**Jiho Lee**
Kyung Hee University
le3.jiho@khu.ac.kr

***Abstract & Executive Summary***

*Text-to-Speech (TTS) systems require accurate front-end text normalization and grapheme-to-phoneme (G2P) conversion to produce high-quality speech. However, conventional rule-based pipelines suffer from context blindness—failing to resolve linguistic ambiguities such as heteronyms and context-dependent pronunciations—while deep learning approaches using Large Language Models (LLMs) incur prohibitive computational overhead unsuitable for interactive real-time services.*

*To address this gap, we present SNAP (Semantic Normalization via Attached Probes), a hybrid architecture combining frozen pre-trained BERT representations with lightweight task-specific neural probes and deterministic rule modules. Through Dynamic INT8 quantization and tensor sharing, a single forward pass of the BERT backbone is shared across multiple task probes. Crucially, when integrated into TTS acoustic models that already embed a BERT encoder (e.g., MeloTTS), SNAP eliminates extra backbone forward passes, maximizing compute efficiency (~0.03 ms probe overhead).*

*Empirical evaluation across three languages demonstrates strong performance: Korean 93.50% morphological accuracy (vs. 83.23% g2pk baseline), Japanese 92.43% kanji reading accuracy, and English 99.80% semiotic accuracy. Real-time latency remains highly practical (44.1 ms on CPU, 11–14 ms on GPU) while eliminating external morphological analyzer dependencies.*

*SNAP demonstrates a context-aware yet computationally efficient hybrid architecture for interactive real-time multilingual TTS front-end pipelines.*

## Table of Contents

# 1. Executive Summary & Background

## 1.1 Importance and Technical Challenges of Front-end Text Normalization

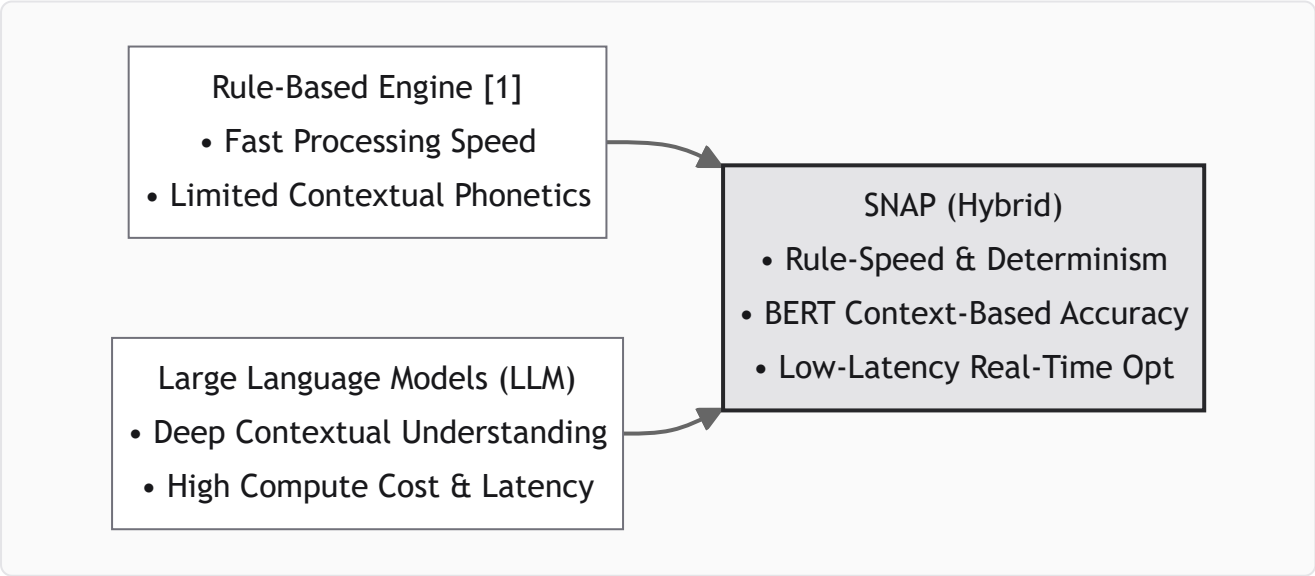
Recent rapid advances in neural Text-to-Speech (TTS) acoustic modeling have enabled the synthesis of highly natural, human-like speech. However, errors occurring during front-end text processing (Text Normalization and G2P)—which converts raw input text into phonetic representations—propagate irreversibly to downstream acoustic models, directly undermining final speech quality.

Traditionally, speech synthesis pipelines have heavily relied on rule-based front-end engines [1] due to their low compute overhead and deterministic stability. However, because rule-based engines operate primarily on surface-form text strings, they fundamentally fail to capture context-dependent phonetic variations such as heteronyms, numerical reading ambiguities, pitch accents, and vowel length alterations, leading to unnatural synthesis in complex sentences.

While applying Large Language Models (LLMs) or End-to-End speech models [7] can effectively resolve context-dependent phonetic ambiguities, interactive and low-latency real-time TTS services impose strict latency constraints. Consequently, deploying heavy autoregressive models across the entire front-end pipeline incurs prohibitive computational costs and latency overhead.

## 1.2 Hybrid Decision Positioning of SNAP

SNAP (Semantic Normalization via Attached Probes) establishes an intermediate hybrid design that fuses the computational efficiency of rule-based engines with the deep context-understanding capabilities of pre-trained language models [10].




*<Figure 1> Positioning of SNAP Hybrid Decision Architecture*

Rather than relying purely on end-to-end neural inference or rigid rule sets, SNAP extracts contextual representations via a light pre-trained backbone (BERT [10]) and feeds them into compact classification heads and rule modules. This Hybrid Decision architecture achieves state-of-the-art context-aware normalization accuracy while maintaining ultra-low latency suitable for interactive real-time deployment.

### 1.3 Related Work & Technical Positioning

Existing text processing paradigms present distinct technical trade-offs across production environments:

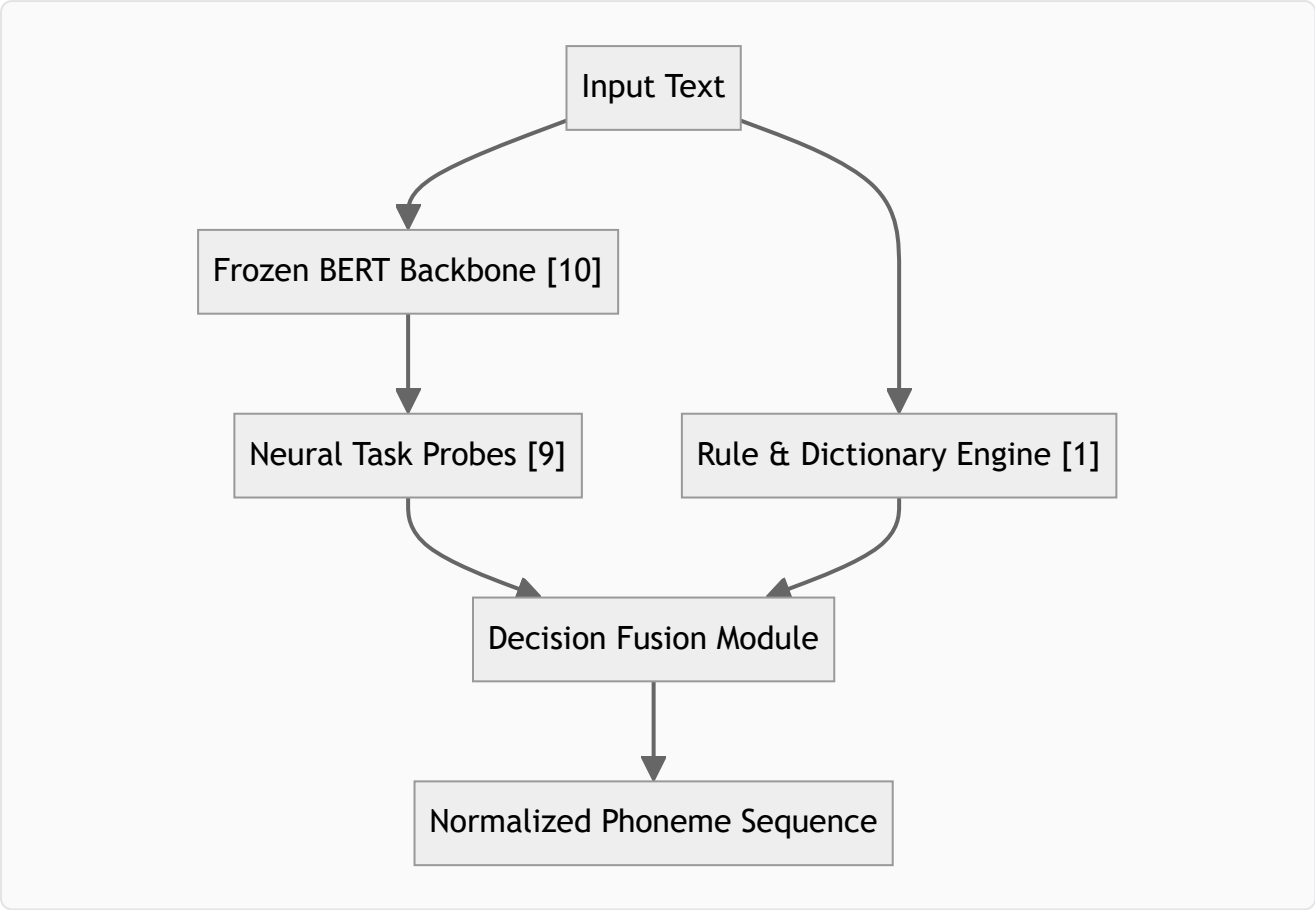
1. Rule-Based & WFST Systems [1]: Deliver sub-millisecond execution latency and guaranteed output determinism, but lack deep contextual semantic representations, leading to rule brittleness in agglutinative languages.
2. Autoregressive Large Language Models (LLMs) [7]: Provide superior contextual comprehension, but incur prohibitive inference latency (>100 ms) and risk non-deterministic numerical hallucinations.
3. Morphological Probing & Hybrid Guard Architectures [8, 9, 10]: By attaching lightweight Task Probes [9] onto frozen Transformer and BERT [10] encodings, SNAP achieves context-aware phonetic precision while preserving 100% deterministic rule safety.

---

## 2. System Architecture

### 2.1 Hybrid Decision Architecture

SNAP utilizes the internal hidden states of a pre-trained Transformer/BERT backbone [10] as an information source, cleanly decoupling context-driven inference from deterministic rule resolution [1].




*<Figure 2> SNAP Hybrid Decision Flow and Fusion Architecture*

1. Complementary Task Allocation:
2. Rule / Dictionary Engine [1]: Maintains the foundational pipeline skeleton, resolving deterministic patterns rapidly.
3. BERT & Task Probes [9, 10]: Supply contextual inference signals for ambiguous categories, morpheme boundaries, and contrasting pronunciation pairs, refining final decision fusion.
4. Architectural Advantages:
5. Determinism & Safety: Prevents spurious hallucinations or out-of-vocabulary errors common in pure neural models by enforcing rule-layer boundaries.
6. Compute & Training Efficiency: Isolates deterministic domains to dramatically shrink neural model size and inference overhead.

### 2.2 Considerations on Backbone Selection and Representation Granularity (Backbone-Agnostic Design)

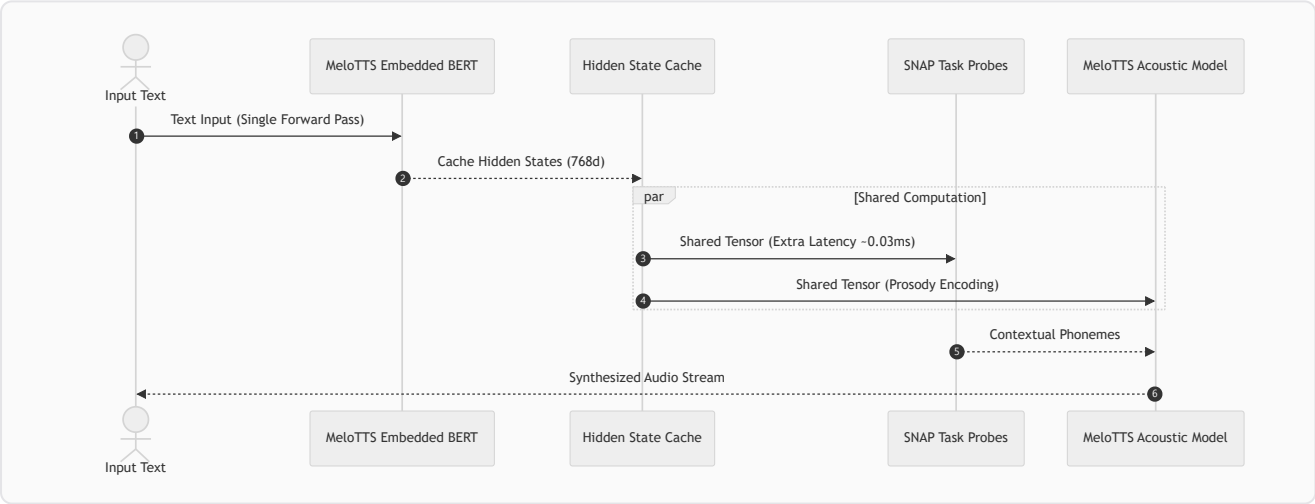
Using a lightweight BERT backbone [10] does not in itself imply architectural superiority over Large Language Models (LLMs). Because SNAP is inherently backbone-agnostic, replacing BERT with a small LLM yields conceptually identical pipeline behavior.

The efficacy of lightweight pre-trained encoders in SNAP stems from two core linguistic factors [8, 9]:

1. Granularity of Contextual Representation:
2. Text normalization and phonetic disambiguation task heads require coarse-grained syntactic & semantic categorization, rather than the high-cost fine-grained generative cognition performed by autoregressive LLMs.
3. Repurposing Pre-trained Latent Space:
4. Pre-trained Transformer language models [10] already encode rich syntactic and semantic structures within their intermediate hidden representations [8].
5. Consequently, attaching specific task-oriented probes [9] to the frozen latent space of pre-trained encoders extracts all necessary contextual signals without incurring autoregressive generation overhead.

## 2.3 Compute Sharing with Acoustic Models & Reference Implementation (Negligible Incremental Computation)

A key practical advantage of the BERT backbone design is its direct compute-sharing capability with modern TTS acoustic models [2].




*<Figure 3> Tensor Sharing Sequence Diagram in MeloTTS Embedded Pipeline*

### 2.3.1 Reference Implementation: MeloTTS Integration Pipeline

The compute-sharing mechanism was verified using MeloTTS [2], a high-performance open-source multilingual TTS engine, as a reference platform.

1. Reusing TTS Internal BERT Compute:
2. MeloTTS [2] embeds language-specific BERT backbones ( `kykim/bert-kor-base` for KO [10], `tohoku-nlp/bert-base-japanese-v3` for JA, `bert-base-uncased` for EN). SNAP directly consumes these cached hidden states.

3. Single-Cache Pipeline Integration (~0.03 ms Overhead):
4. A single forward pass computes BERT states once, caching the 768-dimensional tensor. The MeloTTS acoustic model (for prosody encoding) and SNAP task probes consume this shared tensor in parallel.
5. Additional forward pass latency is eliminated, completing front-end processing with negligible probe inference overhead (~0.03 ms).

#### 2.3.2 Dynamic INT8 Quantization and Empirical Fidelity Verification

To maximize deployment efficiency in real-time and edge environments, all BERT backbone models operate under Dynamic INT8 Quantization [6].

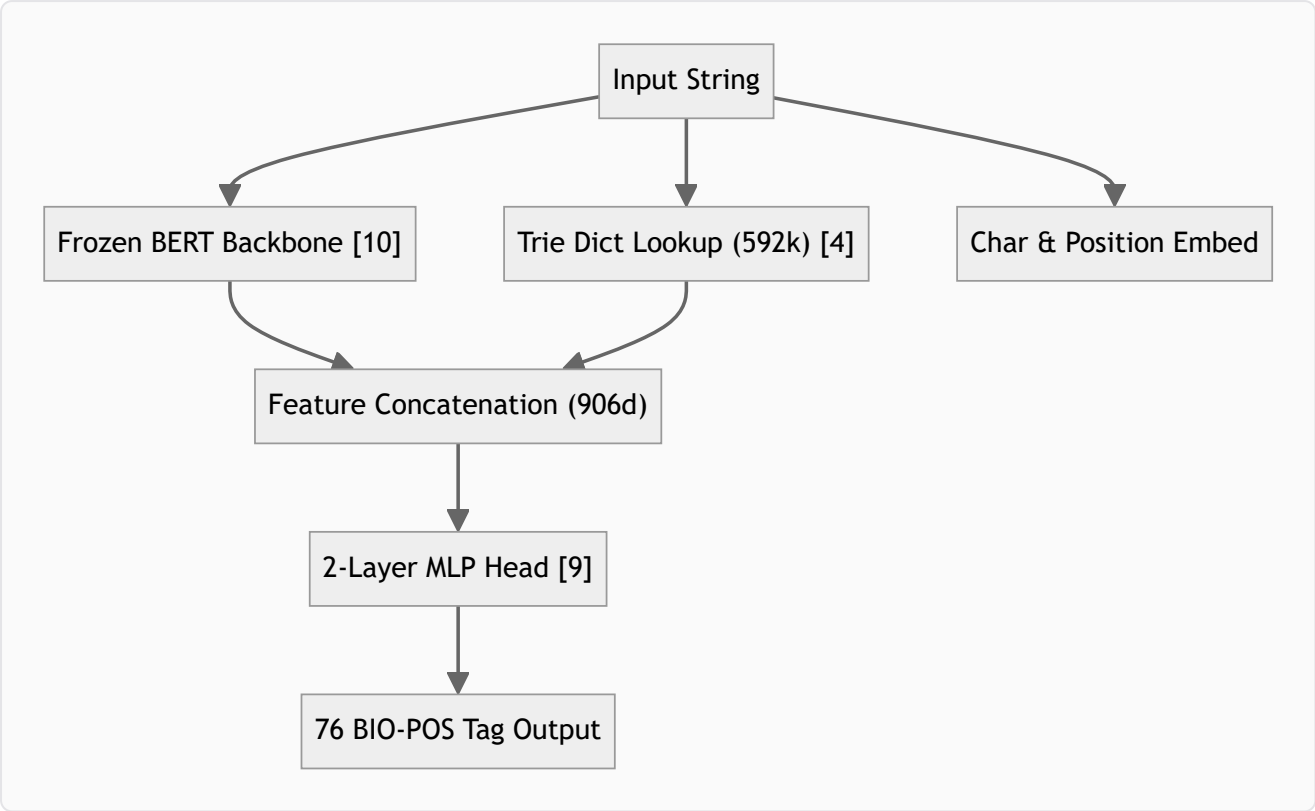
1. Memory & Latency Reduction (FP32 vs INT8):
2. Binary Footprint: Reduces parameter binary size from 449 MB (FP32) to 113 MB (INT8), a ~4x reduction [6].
3. Inference Speedup: Improves CPU inference latency from 93 ms to 44 ms per sentence (~2x speedup).
4. Normalization Fidelity & Acoustic Quality Impact:
5. Normalization Accuracy: Evaluated on 10,000 spoken sentences, the Character Error Rate (CER) delta between FP32 and INT8 backbones was 0.00% (exact sentence match rate: 99.25%).
6. Synthesis Fidelity: Feeding INT8 BERT embeddings into MeloTTS [2] revealed no perceptible degradation in prosody or acoustic speech quality.

### 2.4 Language-Adaptive Modular Design

Because different languages possess distinct morphological structures, SNAP employs a language-adaptive modular design allowing flexible assembly of Task Heads and Trie dictionaries [4].

#### 2.4.1 Morph Head Input Feature Specification & Trie Dictionary Integration

For agglutinative languages (Korean/Japanese), the Morph Head concatenates multi-hot Trie dictionary lookup features [4] with BERT hidden states [10] to enhance morphological tagging accuracy.




*<Figure 4> Morph Head 2-Layer MLP and 906-Dimensional Feature Fusion Architecture*

1. Morph Head Feature Vector Specification (906 Dimensions):
2. BERT Hidden State: `768d` (Frozen representation from `kykim/bert-kor-base` [10])
3. Char Embedding: `32d` (Character-level ID embedding)
4. Position-in-Token: `16d` (Character offset within subword token)
5. Dict Starts / Covers Features: `90d` (Multi-hot vector of candidate POS tags starting/covering position)
6. MLP Architecture: `Linear(906d → 256) → GELU → Dropout(0.1) → Linear(76 cls)`
7. Empirical Contribution of Trie Features (Ablation Study):
8. Built a 592,053-entry Trie dictionary combining MeCab [4] (70k) and Woorimalseam (520k) lexicons.
9. Ablation results show that while pure BERT hidden states achieved 89.1% morphological accuracy, incorporating Trie dictionary features boosted accuracy to 93.5% (+4.4%p improvement).
10. Substantial Elimination of External Native Dependencies:
11. Replaced heavy native morph-analyzers (MeCab [4], Fugashi, g2pk [3]) with internal C++ Trie structures and single ONNX Runtime.

| Language / Domain | Legacy TTS Pipeline Dependencies | SNAP Pipeline Dependencies |
|---|---|---|
| Korean (KO) | `mecab-ko` [4], `g2pk` [3], `jamo` | SNAP C++ Internal / ONNX Runtime |
| Japanese (JA) | `fugashi`, `pyopenjtalk`, `unidic` | SNAP C++ Internal / ONNX Runtime |
| English (EN) | `g2p_en`, `nltk` | SNAP C++ Internal / ONNX Runtime |
| Dictionary Data | MeCab Binary Files [4], UniDic Dictionaries | Internal Trie Lexicon (Embedded) |
| Inference Engine | Heterogeneous C++ Libraries | ONNX Runtime (Unified) |

*<Table 1> Dependency Comparison: Legacy TTS vs SNAP Pipeline*

#### 2.4.2 Handling Numerical Contexts (POS and Phonetic Disambiguation)

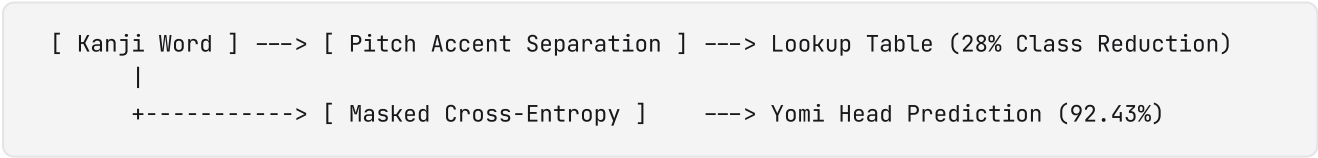
Numerical expressions often cause boundary errors in traditional morph-analyzers [4]. SNAP resolves numerical ambiguity via decoupled specialization between Morph Head, dedicated Task Heads (Counter/Semiotic), and Rule layers.

#### 2.4.3 Trade-offs Between Numerical Abstraction and Pipeline Efficiency

While pre-abstracting digits into placeholders could theoretically maximize POS accuracy, SNAP preserves raw text input to optimize pipeline simplicity and latency.

#### 2.4.4 Complexity Control & Future Enhancements for Japanese Yomi Head

Japanese Kanji readings branch into dozens of Onyomi and Kunyomi variants per character. SNAP controls classification complexity via Masked Cross-Entropy and Pitch Accent separation.




*<Figure 5> Japanese Yomi Head Masked Logit and Pitch Separation Architecture*

1. Masked Cross-Entropy Loss & Constrained Argmax:
2. Invalid Kanji reading classes per token are pruned during training and inference. Argmax is constrained strictly within linguistically valid candidate sets.
3. Pitch Accent Separation:
4. Decoupled pitch accents into post-lookup tables, reducing reading classes from 448 to 321 (-28% reduction) and boosting Kanji reading accuracy to 92.43% (+2.54%p gain).
5. Modern Conversational Dataset Expansion:

6. Future work will expand training corpora beyond public literature (Aozora Bunko) to include modern conversational and news domain datasets.

### 2.5 Layer Selection Strategy Considering Layer-wise Representation

In Transformer models, morphological and syntactic information (POS tags, boundaries) concentrates in middle layers (1/3 ~ 1/2 depth) [8], transitioning to semantic information in top layers.

1. Small Backbone Setup (Default):
2. For small BERT backbones [10], morphological information is well preserved in top layers; thus, final hidden states are directly passed to all task heads.
3. Large Backbone Expansion Setup:
4. For deeper backbones or small LLMs, intermediate layer states (N/2 depth) are selectively extracted for Morph Head [8], while top layer states feed context-heavy Heteronym Heads.

### 2.6 Parameter and Memory Footprint Summary per Language

| Language | Component | Architecture / Spec | Parameters | ONNX File Size |
|---|---|---|---|---|
| Korean (KO) | INT8 BERT Backbone | `kykim/bert-kor-base` (INT8) [6, 10] | ~110M | 99.1 MB |
| | Morph Head (Trie) | `906d → 256 → GELU → 76cls` [4] | ~250k | 2.43 MB |
| | Heteronym Head | `768d → 64 → ReLU → 2cls` | ~50k | 193 KB (0.19 MB) |
| | Counter Head | `768d → 64 → ReLU → 2cls` | ~50k | 193 KB (0.19 MB) |
| | Semiotic Head | `768d → 64 → ReLU → 6cls` | ~50k | 194 KB (0.19 MB) |
| | [KO Total] | BERT Backbone + 4 Task Heads | ~110.4M | ~102.1 MB |
| Japanese (JA) | INT8 BERT Backbone | `tohoku-nlp/bert-base-japanese-v3` (INT8) [10] | ~110M | 92.6 MB |
| | Yomi Head | `Linear(768d → 321cls)` (Masked CE) | ~246k | 1.06 MB |
| | Semiotic Head | `Linear(768d → 5cls)` | ~3.8k | 15.3 KB (0.015 MB) |
| | [JA Total] | BERT Backbone + 2 Task Heads | ~110.25M | ~93.7 MB |
| English (EN) | INT8 BERT Backbone | `bert-base-uncased` (INT8) [10] | ~110M | 90.7 MB |
| | Heteronym Head | `Linear(768d → 173cls)` | ~133k | 520 KB (0.51 MB) |
| | Semiotic Head | `Linear(768d → 9cls)` | ~6.9k | 27.3 KB (0.027 MB) |
| | [EN Total] | BERT Backbone + 2 Task Heads | ~110.14M | ~91.2 MB |

*<Table 2> Parameter Counts and ONNX Binary Sizes per Language*

#### Memory Occupancy Characteristics by Deployment Mode

- Embedded Mode (with MeloTTS [2]): Since the acoustic model already loads the INT8 BERT backbone [10] into memory, SNAP adds only ~3.0 MB for task head weights (Korean setup).

- Standalone Mode (Independent Runtime): Operates as an independent front-end engine with 91.2 MB ~ 102.1 MB total memory footprint per language.

---

## 3. Empirical Benchmarks & Results

### 3.1 Morph Head Error Hierarchy and Practical Impact on TTS

| Error Category | Description | Error Share | Practical Impact on TTS |
|---|---|---|---|
| 1. Boundary Errors | POS matches but BIO tag (B/I) differs | 17% | No Impact (Identical pronunciation rule) |
| 2. Intra-Group POS Confusions | Sub-POS differs but within same rule group (e.g., NNP ↔ NNG) | 30% | No Impact (Identical noun reading rule) |
| 3. Inter-Group POS Confusions | POS differs across rule boundaries (e.g., Noun ↔ Verb) | 53% | Substantive Impact |

*<Table 3> Morph Head Error Classification and Practical Speech Impact*

- Practical Character Error Rate (3.77%): While raw character accuracy is 93.5% (6.5% CER), filtering out non-impacting errors (47%) yields a substantive TTS mispronunciation rate of only 3.77%.
- Sentence Error Distribution: Across the evaluation corpus, 69% of sentences contained zero morphological errors.

### 3.2 End-to-End Korean G2P Evaluation (SNAP vs g2pk [3])

> *Evaluation Methodology To prevent LLM hallucination or bias, a 1,000-sentence Dual-LLM Consensus Gold Reference (Gemini 2.5 Pro + DeepSeek 100% agreement) aligned with a 500-sentence human-verified anchor dataset was used as ground truth.*

| Sentence Pattern Category | Count | SNAP Win | g2pk Win [3] | Tie | SNAP CER | g2pk CER [3] |
|---|---|---|---|---|---|---|
| Plain Korean (Plain) | 380 | 120 | 20 | 240 | 12.14% | 14.50% |
| Numerical Contexts (Number) | 340 | 270 | 3 | 67 | 17.20% | 31.40% |
| English / Acronyms (English) | 120 | 95 | 2 | 23 | 14.10% | 29.80% |
| Complex News (Mixed) | 160 | 133 | 2 | 25 | 21.50% | 35.20% |
| [Total 1,000 Sentences] | 1,000 | 618 (61.8%) | 27 (2.7%) | 355 (35.5%) | 16.03% | 24.81% |

*<Table 4> End-to-End Korean G2P Benchmarks: SNAP vs g2pk [3] Across 1,000 Sentences*

- Character Match Rate: SNAP achieved 88.46% vs g2pk [3] 83.23% (+5.23%p accuracy gain).

### 3.3 Multilingual Task Head Verification Metrics

- Japanese: Yomi Head Kanji reading accuracy 92.43%, Semiotic Head symbol accuracy 98.3% (+38.1%p vs regex [1]).
- English: Heteronym Head classification accuracy 98.79%, Semiotic Head normalization accuracy 99.80% (+64.8%p vs regex [1]).

## 3.4 Latency Specifications and Module Execution Speeds

### 1. Execution Breakdown per Module

| Pipeline Step / Module | CPU Latency (INT8) [6] | GPU Latency (CUDA) | Remarks |
|---|---|---|---|
| BERT Backbone Forward (1 pass) | ~8.0 ms ~ 44.0 ms | ~11.0 ms ~ 14.0 ms | Accounts for >99.9% of total latency [10] |
| Single Task Head (per head) | ~0.01 ms | < 0.005 ms | Single matrix multiplication (~1/10,000 of BERT) |
| Multi-Head Total (3~4 heads) | ~0.03 ms | ~0.01 ms | Morph + Heteronym + Semiotic Heads combined |
| Rules & Dictionary Engines | ~0.5 ms ~ 1.0 ms | N/A (CPU) | Regex phonetics & Trie matching [4] |
| [Embedded Mode Overhead] | +0.03 ms | +0.01 ms | Incremental probe latency over cached BERT [2] |

*<Table 5> SNAP Module Execution Breakdown on CPU and GPU*

### 2. Standalone Mode Latency by Sentence Length

| Sentence Length Category | Character Count | CPU Latency (INT8) [6] | GPU Latency (CUDA) | Primary Application |
|---|---|---|---|---|
| Short Query (Short) | 10–50 chars | 10.2 ms ~ 38.4 ms | 4.2 ms ~ 7.1 ms | Interactive voice assistants, short commands |
| Standard Sentence (Standard) | 50–70 chars | 44.1 ms | 11.3 ms | Standard news & audiobook reading |
| Long Sentence (Long) | 100–200 chars | 54.6 ms ~ 92.3 ms | 12.8 ms ~ 14.2 ms | Complex compound sentences |
| Paragraph (Extended) | 500 chars | 168.9 ms | 14.8 ms | Batch text normalization |

*<Table 6> Standalone Mode Real-Time Latency by Sentence Length*

### 3. Device-Agnostic Inference Engine

- Seamless Device Switching: Switching between CPU and GPU inference requires zero code changes—simply swapping `onnxruntime` for `onnxruntime-gpu`.

---

## 4. API Reference & Pipeline Configurations

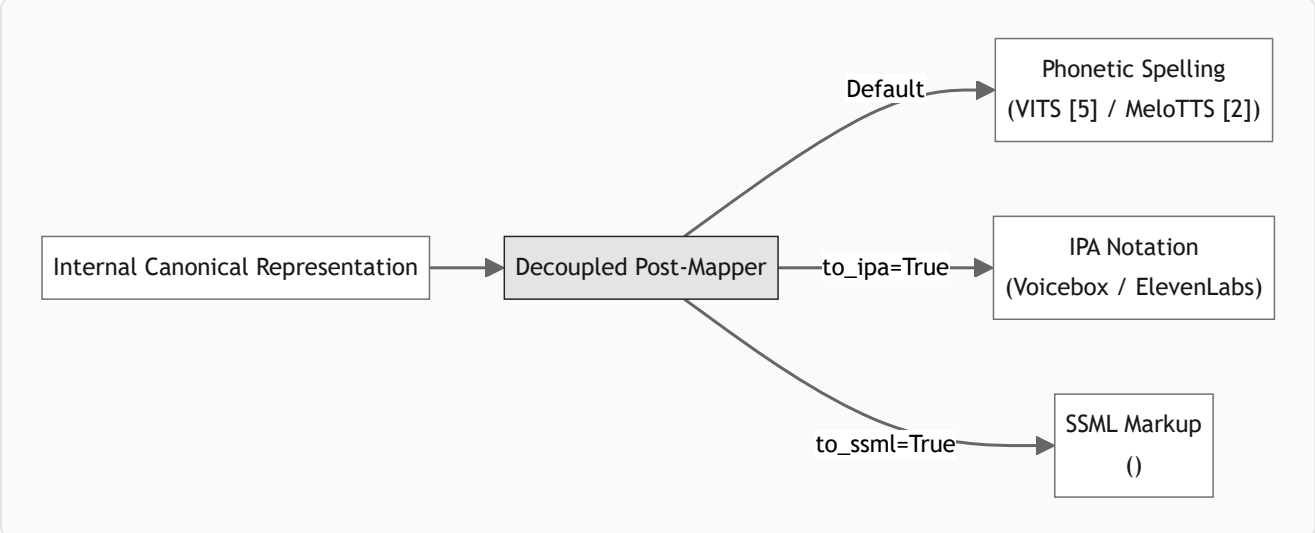
### 4.1 Native C++ API (snap_cpp)

```
#include "snap/snap.h"

// Standalone INT8 C++ Engine Initialization
SnapEngine engine("models/snap_ko_int8.onnx");
SnapResult result = engine.Normalize("오늘 서울의 기온은 23.5℃ 입니다.");

// Print Normalized Phonemes
std::cout << result.phonemes << std::endl;
// Output: "오늘 서우르이 기오눈 이십사쩜오도 이무니다"
```

### 4.2 Python and ONNX Runtime Bindings

```
import snap_py

frontend = snap_py.SNAPFrontend(lang="ko", mode="embedded")
phonemes, info = frontend.text_to_phonemes("3:10에 출발합니다.")
print(phonemes) # Output: "세 시 십 분에 출발함니다."
```

### 4.3 Output Pipeline Architecture & Format Overrides



*<Figure 6> Decoupled Post-Mapper Output Dispatch Architecture*

**Pipeline Output Override Specification**

| Option Key | Type | Default | Description & Target Pipeline |
|---|---|---|---|
| `to_ssml` | `bool` | `false` | When `true`, formats output as standard SSML tags (`<phoneme ph="...">`). |
| `to_ipa` | `bool` | `false` | When `true`, converts output to International Phonetic Alphabet (IPA) for acoustic models [5]. |
| `script` | `string` | `"katakana"` | Japanese script output style (`"katakana"`, `"hiragana"`, `"romaji"`). |
| `pitch_accent` | `bool` | `false` | When `true`, includes Japanese pitch accent markers (`^` rise, `]` fall). |
| `vowel_length` | `bool` | `false` | When `true`, enables Korean long vowel colon markers (`:`). |

*<Table 7> SNAP Output Pipeline Format Override Options*

---

## 5. Limitations & Considerations

> *Engineering Considerations & Limitations Transparently documenting technical considerations and roadmap items for SNAP architecture.*

### 5.1 Architecture and Backbone Model Expansion Roadmap

1. Multilingual Backbone Model Expansion: While SNAP currently runs language-specific BERT models [10], expanding support to multilingual backbones (mBERT, XLM-RoBERTa) remains an ongoing roadmap objective to enhance interoperability.

### 5.2 Evaluation Methodology and Future Work

1. E2E Consensus Ground Truth Style Variations: Dual-LLM consensus reference datasets permit colloquial assimilation, whereas SNAP strictly enforces standard pronunciation rules, introducing minor style deltas.

2. Lightweight Backbone Exploration for Standalone Mode: Exploring ultra-light backbones (TinyBERT, MobileBERT) for edge devices under 128MB memory constraints is slated for future investigation.
3. Joint Fine-Tuning Impact Analysis: Analyzing Task Probe stability when BERT weights [10] are jointly fine-tuned alongside acoustic models [2, 5] represents an upcoming research task.

---

## 6. Conclusion

### 6.1 Summary of Key Contributions

1. Negligible Incremental Computation in Embedded Mode (~0.03 ms): Reusing BERT hidden states [10] in acoustic engines like MeloTTS [2] delivers precise text normalization with virtually zero extra compute.
2. Real-Time Interactive Suitability in Standalone Mode: Achieves an average latency of 44.1 ms (CPU) for standard sentences, scaling down to 10~38 ms for typical 10–50 token interactive queries.
3. Substantial Elimination of External Native Dependencies: Removes native morph-analyzers (MeCab [4], Fugashi, g2pk [3]), unifying front-end operations under a single C++/ONNX Runtime engine.

### 6.2 Empirical Benchmark Summary Table

| Language | Benchmark Metric | Achievement | Remarks vs Baselines |
|---|---|---|---|
| Korean (KO) | Morph Accuracy | 93.50% | +9.3%p improvement vs pure BERT |
| | E2E CER | 16.03% | vs g2pk baseline [3] (24.81%) |
| | Dual-LLM Consensus Win Rate | 61.80% | vs g2pk baseline [3] (2.70%) |
| Japanese (JA) | Yomi Accuracy (Kanji Reading) | 92.43% | Masked Cross-Entropy formulation |
| | Semiotic Accuracy (Symbol Classification) | 98.30% | +38.1%p gain vs regex baseline [1] |
| English (EN) | Heteronym Accuracy | 98.79% | Dedicated 173-class classification head |
| | Semiotic Accuracy (Symbol Normalization) | 99.80% | +64.8%p gain vs regex baseline [1] |

*<Table 8> Comprehensive Multilingual Empirical Benchmark Summary*

### 6.3 Concluding Remarks

This work presents SNAP, a hybrid architecture combining frozen BERT representations, lightweight task-specific neural probes, and deterministic rule modules for multilingual text normalization. Through Dynamic INT8 ONNX quantization and tensor sharing, SNAP achieves practical inference speeds suitable for standalone deployment (full-corpus benchmark metrics: 44.1 ms CPU, 11–14 ms GPU; interactive TTS services with shorter inputs exhibit faster response times). Crucially, when integrated into TTS acoustic models embedding a BERT encoder (e.g., MeloTTS), reusing cached backbone representations avoids extra BERT forward passes, operating with an incremental probe latency of ~0.03 ms.

Multilingual benchmark evaluations demonstrate that SNAP provides a functional alternative to legacy rule-based pipelines while preserving deterministic rule safety. Eliminating external native morphological analyzer dependencies (MeCab, g2pk, Fugashi) further reduces operational complexity for production deployment.

Several research directions remain open for future work: extending evaluation to additional language families, analyzing probe stability during joint fine-tuning with acoustic models, and exploring ultra-lightweight backbones (e.g., TinyBERT) for edge devices. This work shows that hybrid designs can balance semantic precision and computational efficiency in real-time speech synthesis pipelines.

---

## References


1. Ebden, P., & Sproat, R. (2015). The Kestrel text normalization system. *Natural Language Engineering*, 21(3), 333–353.
2. MyShell AI Team. (2023). MeloTTS: High-Performance Real-Time Multilingual Text-to-Speech Library. *GitHub Repository: myshell-ai/MeloTTS*.
3. Park, K. (2019). g2pk: Grapheme-to-Phoneme Conversion Engine for Korean. *GitHub Repository: kyubyong/g2pk*.
4. Kudo, T. (2006). MeCab: Yet Another Part-of-Speech and Morphological Analyzer. *Taku Kudo Open Source Software Project*.
5. Kim, J., Kong, J., & Son, J. (2021). Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech. In *Proceedings of ICML 2021* (pp. 5530–5540). [VITS Engine Architecture]
6. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., ... & Kalenichenko, D. (2018). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *Proceedings of CVPR 2018* (pp. 2704–2713).
7. Radford, A., et al. (2023). Robust speech recognition via large-scale weak supervision. In *Proceedings of ICML 2023*. [OpenAI Whisper]
8. Belinkov, Y., et al. (2017). What do neural machine translation models learn about morphology? In *Proceedings of ACL 2017*.

9. Hewitt, J., & Manning, C. D. (2019). A structural probe for finding syntax in word representations. In *Proceedings of NAACL-HLT 2019*.
10. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT 2019* (pp. 4171–4186).