On-Device Multi-Type Disfluency Detection with Sub-Millisecond Inference on Apple Silicon

Nazar Kozak

Kozak Technologies Inc, Los Angeles, CA 90036, USA

Abstract

Published multi-type disfluency detection systems achieve their best results with 300M+ parameter server-class backbones, leaving speech-therapy applications without a concrete reference for the detection performance and inference latency achievable on a smartphone.

We present *DisfluoSDK*, a multi-type disfluency classifier running entirely on-device on Apple Silicon. On SEP-28K (20,131 clips, episode-grouped 5-fold cross-validation) a 617K-parameter CNN achieves macro-F1 0.382 (1.2 MB CoreML) and an adapted ResNet-18 achieves 0.404 (11.2M parameters, 21 MB)—occupying an otherwise unpopulated region of the accuracy–efficiency Pareto frontier where on-device deployment is feasible.

A four-way CoreML compute-unit sweep across four hardware generations (M1 Max, A19 Pro, A18, A15; 16,000+ timed trials) shows that the Neural Engine delivers sub-millisecond mean inference across all tested devices (CNN 0.225–0.635 ms), providing ample real-time headroom for speech processing on any in-service iPhone. The sweep also surfaces a desktop/mobile CoreML scheduler divergence in GPU routing with a direct consequence for deployment practice. PyTorch-to-CoreML export fidelity is numerically verified on 500 test-fold spectrograms (cell-level agreement 99.96%/100.00%, ΔF1 ≤ 0.003).

As an auxiliary empirical result, voice-stress features show no practically meaningful linear association with any disfluency type across 14,645 clips ($|r| < 0.05$, all Cohen-negligible), supporting the architectural separation of

⋆Preprint submitted to Computer Speech & Language (Elsevier), April 12, 2026. This manuscript has not been peer-reviewed. Please cite as a preprint.

Email address: nzrkzk@gmail.com (Nazar Kozak)

stress and disfluency modules.

Keywords: speech disfluency detection, stuttering, on-device inference, CoreML, Apple Neural Engine, voice stress analysis, SEP-28K, mobile speech processing

---

1. Introduction

Developmental stuttering affects approximately 5% of children, with about 1% continuing into adulthood [1]. Continuous objective monitoring could support therapy between clinical visits, but the highest-accuracy approaches in the recent literature [2] use 300M+ parameter wav2vec 2.0 backbones that require server-side inference—precluding deployment on a user's personal device, where audio is processed privately and no network round-trip is incurred.

On-device speech research has concentrated on ASR and keyword spotting [3]; disfluency-specific detection has remained server-based [4]. The practical consequence is that mobile speech-therapy applications have no published reference point for how much detection performance is recoverable at smartphone-scale model sizes, or what latency budget those models incur on real hardware. This paper provides that reference point.

1.1. Scope and Contributions

This paper is a *deployment-oriented systems and empirical-analysis* contribution, not a new-architecture paper. We do not claim to advance macro-F1 above the state of the art; Bayerl et al.'s wav2vec 2.0 system [2] has higher raw numbers. The goal is instead to quantify what detection performance is available under the constraints that matter for a deployable mobile application: model size, on-device latency, and inference correctness across real hardware.

Concretely, we contribute:

1. **A quantified accuracy–efficiency Pareto frontier for on-device multi-type disfluency detection.** We train a 617K-parameter CNN and an 11.2M-parameter ResNet-18, export both to CoreML, and jointly report macro-F1, model size, and on-device latency across a four-way compute-unit sweep on four hardware generations—providing practitioners with a principled design-space map from 1.2 MB to 21 MB.

Table 1: What closest prior work covers vs. this paper. "Multi-type" = all five SEP-28K disfluency categories. "Stress indep." = empirically tested.

	Lea 2021	Bayerl 2022	Bayerl 2023	Sheikh 2022	**This work**
Multi-type (5-class)	✓	✓	✓	survey	✓
On-device deployed	–	–	–	–	✓
Sub-ms inference	–	–	–	–	✓
Size <25 MB	∼	–	–	–	✓
Stress indep. tested	–	–	–	–	✓
Adaptive F0 baseline	–	–	–	–	✓
Cross-device NE sweep	–	–	–	–	✓

We report this frontier on SEP-28K together with a verified on-device deployment path.

2. **Deployment validation across four Apple Silicon generations.** We benchmark across M1 Max, A19 Pro, A18, and A15 with a correctness-verified harness. The sweep reveals a desktop/mobile scheduler divergence with practical consequences for `CPU_AND_GPU` routing, and quantifies latency across devices spanning 2022–2025.
3. **A dataset-scale empirical test of stress-disfluency linear independence on SEP-28K.** On 14,645 clips with valid F0, we compute four voice-stress features against five disfluency labels (20 Bonferroni-corrected tests), report 95% CIs and two-sample Cohen's d, and find no practically meaningful linear association ($|r| < 0.05$, all Cohen-negligible)—providing empirical grounding for the architectural decision to treat these as independent modules.

Table 1 summarizes the gap with respect to the closest prior systems. Our moderate absolute F1 numbers (0.382–0.404) quantify the cost of the on-device constraint; they are not intended to contest server-class baselines on accuracy terms.

2. Related Work

2.1. Stuttering Event Detection

The SEP-28K dataset [5] established a benchmark for stuttering event detection from podcast audio, providing 28,177 annotated 3-second clips across

five disfluency types. Lea et al. [5] reported per-type F1 using a ConvLSTM with mel-filterbank features augmented with F0 and articulatory features, under a fixed random train/test split. Bayerl et al. [6] subsequently showed that four podcast hosts account for 59% of SEP-28K clips and that random splits allow speaker overlap that inflates results; their speaker-independent SVM baseline is substantially lower. In later work, Bayerl et al. [2] used wav2vec 2.0 with multi-task learning on SEP-28K-E, achieving higher scores at the cost of a 300M+ parameter backbone. Sheikh et al. [7] provide a comprehensive survey. More recent work has explored rule-based prolongation detection across corpora including SEP-28K [8], reporting high single-class accuracy but without addressing the full five-class problem or on-device deployment. None of these works reports on-device latency, a CoreML/TFLite/ONNX model size, or a published deployment path.

2.2. On-Device Speech Processing

On-device inference has matured with engines such as CoreML [9], TensorFlow Lite [10] and ONNX Runtime [11]. While ASR has been deployed on-device [3], disfluency-specific detection has remained server-based. Published work on disfluency classification has not, however, reported on-device multi-type inference or sub-millisecond latency measurements.

Prior latency characterization work has focused on large speech models (ASR, keyword spotting) and has not examined the compute-unit trade-offs specific to small (<15M parameter) classification models on Apple Silicon. A known challenge for on-device deployment is the gap between training-time (Python/PyTorch) and inference-time (Swift/CoreML) feature extraction; we address this via a numerically verified export pipeline and a per-device sanity check harness described in Section 6.4.

2.3. Voice Stress and Disfluency

Voice-stress markers—jitter, shimmer and F0 variability—have been studied as correlates of psychological stress [12, 13]. Their relationship to disfluency events in stuttering populations has not been examined systematically at dataset scale in the published literature we surveyed. Prior multimodal disfluency/stress systems typically assume either tight coupling or full independence between the two signals without empirical validation; Section 5 fills this gap.

3. Dataset and Preprocessing

We use SEP-28K [5], which contains 28,177 three-second clips from podcasts featuring people who stutter. Each clip is annotated by three raters for five disfluency types: Prolongation, Block, Sound Repetition, Word Repetition and Interjection. After filtering clips flagged for poor audio quality, music or absence of speech (majority vote $\geq 2/3$), and matching with downloadable audio (258 of 385 episodes), we retain **20,131 clips** from five shows. Binary labels per disfluency type are established by majority vote ($\geq 2/3$ annotators). Class prevalence: Prolongation 9.9%, Block 12.3%, Sound Rep. 8.9%, Word Rep. 11.5%, Interjection 23.3%.

Each 16 kHz mono WAV clip is converted to a log-mel spectrogram with 128 mel bands, a 1024-sample FFT and a 512-sample hop, yielding tensors of shape $(1, 128, 94)$. Per-sample normalization (zero mean, unit variance) is applied at training time. The entire mel pipeline is re-implemented in Swift using Accelerate/vDSP so that training-time and on-device features are numerically consistent.

4. Method

4.1. Model Architectures

To characterize the accuracy–efficiency Pareto frontier, we intentionally evaluate two models at very different scales: a custom lightweight CNN and a transfer-learned ResNet-18. This pairing brackets the design space between a task-specific sub-1M-parameter architecture and the smallest standard ImageNet backbone, while remaining within what a practitioner would consider for real-time on-device deployment. Comparisons with architectures such as MobileNetV3 or EfficientNet-Lite are left as future work.

DisfluencyCNN. A 4-block convolutional network. Each block: two 3×3 convolutions with batch normalization, ReLU, 2×2 max pooling and spatial dropout. Channels $32 \to 64 \to 128 \to 256$, followed by adaptive average pooling and a classifier head ($256 \to 128 \to 5$). Parameters: 616,549. CoreML size: 1.2 MB.

ResNet-18 adapted. A ResNet-18 [14] pre-trained on ImageNet, with the input convolution modified from 3 to 1 channel (by averaging pre-trained weights) and the head replaced with $512 \to 128 \to 5$. Parameters: 11,236,549. CoreML size: 21 MB.

Both networks produce five logits for multi-label prediction via sigmoid activation and per-class thresholding.

4.2. Training Protocol

We use binary cross-entropy loss with per-class positive weights set to the ratio of negative to positive samples, AdamW with weight decay 10^{-4} and a cosine-annealing learning-rate schedule. Training runs for up to 30 epochs with early stopping (patience 7). Data augmentation follows SpecAugment [15]: time masking (12 frames $\approx 0.4\,\mathrm{s}$), frequency masking (20 mel bands) and random gain ($\pm 20\%$).

4.3. Episode-Grouped Cross-Validation

We employ 5-fold stratified group cross-validation (`StratifiedGroupKFold`, seed 42) grouped by podcast episode, so that all clips from a single recording session remain in the same fold. This substantially reduces, though does not fully eliminate, data leakage via shared speaker characteristics, since the same speaker may appear across multiple episodes [6]. Our protocol is stricter than Lea et al.'s random split but less strict than full speaker-independent evaluation; we make this choice explicit so that our numbers can be placed between the two prior protocols.

4.4. Per-Class Threshold Optimization

Rather than a fixed 0.5 threshold, we optimize per-class thresholds on the validation fold of each split to maximize F1—important given that class imbalance shifts the operating point. Thresholds are frozen before test-fold metrics are computed and are embedded as model metadata at CoreML export time.

4.5. On-Device Deployment

Trained models are exported to CoreML (`.mlpackage`) via TorchScript tracing and `coremltools` 9.0, targeting iOS 17+. At runtime the Swift SDK performs mel-spectrogram extraction, model inference and per-class thresholding without contacting any network endpoint. The SDK is implemented as a zero-dependency Swift Package relying exclusively on AVFoundation, Accelerate and Foundation.

5. Voice-Stress Analysis and Independence

DisfluoSDK includes a per-speaker adaptive stress-scoring module that runs alongside disfluency detection. This section describes its implementation and then presents a dataset-scale empirical test of the linear independence between stress features and disfluency labels—an auxiliary result that

informs the system's modular architecture by establishing that the two signal streams are not redundant at this scale.

5.1. YIN-Based Pitch Detection

F0 is extracted using the YIN algorithm [16]. Our implementation is built on Apple's Accelerate framework (vDSP): precompute cumulative energy, compute autocorrelation via `vDSP_dotpr`, form the difference function $d(\tau) = E_{\text{head}} + E_{\text{tail}} - 2 \cdot \text{autocorr}(\tau)$, and apply the cumulative mean normalized difference function (CMNDF) with threshold $\tau_{\text{th}} = 0.15$. Frame size: 1024 samples (64 ms at 16 kHz); hop: 512 samples. This achieves **98.1% F0 detection** across SEP-28K versus 12.1% with naive autocorrelation—a prerequisite for meaningful mobile voice-stress scoring.

5.2. Per-Speaker Adaptive Baseline

We implement per-speaker adaptive baseline calibration:

- **Calibration** (first 10 voiced frames): Welford's online algorithm [17] maintains running mean and variance for jitter, shimmer, F0 std and speech rate.
- **Tracking:** an exponential moving average ($\alpha = 0.02$) provides drift adaptation without catastrophic re-scaling.
- **Scoring:** z-score $\rightarrow$ sigmoid (steepness = 1.5) $\rightarrow$ weighted composite, blended with raw scores in proportion to calibration confidence.

5.3. Empirical Independence from Disfluency

Of the 20,131 SEP-28K clips retained after download, 14,645 (72.8%) yield valid F0 estimates; the remaining 5,486 clips (27.2%) are excluded from the independence analysis because they are unvoiced (silence or noise segments where YIN returns no reliable fundamental frequency), for which jitter, shimmer and F0-std are undefined. We compute Pearson and point-biserial correlations between four voice-stress features and five disfluency labels across $N = 14{,}645$ clips with valid F0, for a total of 20 tests. Results appear in Table 2.

Although four Prolongation correlations reach statistical significance at our very large N, *every effect size is negligible by Cohen's convention* ($|r| < 0.10$). The strongest observation—stress $\times$ prolongation, $r = -0.050$—explains only 0.25% of variance ($r^2 = 0.0025$); using Fisher's z-transform, the 95% CI

Table 2: Pearson correlation (r) between voice-stress features and disfluency labels ($N=$ 14,645). *significant after Bonferroni correction ($\alpha=0.05/20$). All effect sizes are negligible by Cohen's convention ($|r|<0.10$).

Feature	Pro	Blk	SRep	WRep	Intj
Jitter	$-.028^*$	$-.006$	.008	.013	.001
Shimmer	$-.030^*$	.002	.019	.013	$-.002$
F0 Std	$-.028^*$	$-.007$	$-.009$	.021	.001
Stress	$-.050^*$	$-.013$	$-.007$	.025	.004

for this correlation is $[-0.066, -0.034]$, with an upper absolute bound well inside the Cohen-negligible range. At $N=14{,}645$ the study has >99% statistical power to detect any true correlation of magnitude $|\rho| \geq 0.05$ at $\alpha=0.05$; that the observed correlations are this small even under such power is itself informative. As a complementary two-sample test, we split clips into *any-disfluency* ($N = 7{,}793$) versus *fluent* ($N = 6{,}852$) subsets and compute Cohen's d for each stress feature: jitter $d = -0.001$, shimmer $d = -0.010$, F0 std $d = -0.006$, composite stress $d = -0.034$—all negligible by Cohen's convention ($|d| < 0.2$) and consistent with the correlation analysis. Figure 1 visualizes the full 4×5 correlation matrix. We therefore did not observe any practically meaningful linear association between voice-stress features and disfluency labels in SEP-28K. This result is consistent with clinical theory: the acoustic voice-stress correlates we measure (jitter, shimmer, F0 variability) reflect moment-to-moment laryngeal tension within a 3-second clip, whereas psychological stress is a sustained affective state that manifests across minutes or hours of speech; the two constructs operate at different timescales, so a near-zero clip-level correlation is the expected null outcome, not a surprising one. This supports—but does not conclusively prove—the architectural decision to treat stress analysis and disfluency detection as statistically independent modules in multimodal mobile systems. Non-linear or conditional relationships are not ruled out by this marginal analysis and are a direction for future work.

6. Experiments and Results

6.1. Disfluency Detection

Table 3 reports per-type results at a default 0.5 threshold; Table 4 reports results with per-class threshold optimization. Per-fold stability is tight: CNN

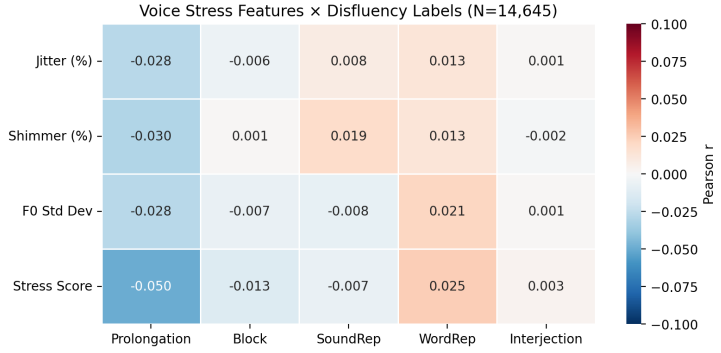

Figure 1: Correlation heatmap between four voice-stress features and five disfluency labels on SEP-28K ($N = 14{,}645$). All 20 cells lie inside the Cohen-negligible range ($|r| < 0.10$); the single darkest cell is $r = -0.050$ for composite stress × Prolongation.

Table 3: Per-type results, 5-fold episode-grouped CV, threshold = 0.5.

	CNN				ResNet-18			
Type	P	R	F1	AUC	P	R	F1	AUC
Pro	.276	.737	.401	.841	.266	.718	.388	.829
Blk	.201	.555	.295	.671	.196	.531	.287	.658
SRep	.222	.630	.328	.777	.233	.648	.342	.790
WRep	.174	.643	.273	.669	.190	.686	.298	.717
Intj	.411	.638	.500	.738	.461	.674	.548	.792
Macro	**.257**	**.640**	**.360**	—	**.269**	**.651**	**.373**	—

macro-F1 0.359±0.011 across the five folds, ResNet-18 0.377±0.014, indicating the training protocol is consistent and the mean values are reliable estimates rather than artifacts of a single favorable split. Threshold optimization improves macro-F1 by 6.1% (CNN) and 8.3% (ResNet-18). Consistent across both architectures, Interjection is the easiest class (F1 $\approx$ 0.50–0.56) and Block/Word Rep. are the hardest (F1 $\approx$ 0.28–0.32)—an ordering that mirrors published inter-annotator agreement on SEP-28K [5] and suggests the residual error floor is partially irreducible with respect to the current label quality.

Table 4: Results with optimized per-class thresholds.

	CNN		ResNet-18	
Type	F1	Thresh	F1	Thresh
Pro	.461	0.69	.453	0.70
Blk	.299	0.53	.291	0.55
SRep	.368	0.67	.385	0.71
WRep	.279	0.55	.324	0.66
Intj	.505	0.52	.564	0.63
Macro	**.382**	—	**.404**	—

Table 5: Comparison on SEP-28K. Evaluation protocols differ. "Device" indicates a published on-device deployment path.

Method	Protocol	Pro	Blk	SR	WR	Intj	Params	Device
Lea [5]	Rand.	.685	.559	.632	.604	.713	∼2M	No
SVM [6]	Spkr-ind.	.460	.360	.460	.510	.710	N/A (SVM)	No
Bayerl [2]	Spkr-ind.	.530	.320	.530	.640	.770	300M+	No
CNN (ours)	Ep-grp	.461	.299	.368	.279	.505	**617K**	**Yes**
RN18 (ours)	Ep-grp	.453	.291	.385	.324	.564	11.2M	**Yes**

6.2. Positioning on the Accuracy–Efficiency Pareto Frontier

Table 5 places our results alongside prior work. Evaluation protocols differ and absolute numbers are therefore not directly comparable; we report them together to make the trade-offs explicit, and visualize the resulting positioning in Figure 2.

The contribution of this work is intentionally positional rather than absolute-accuracy-focused. We are not claiming to advance disfluency detection performance under speaker-independent evaluation; the speaker-independent SVM [6] already sets a harder protocol than ours, and the evaluation gap is acknowledged. The claim is, instead, that when inference must fit within a mobile model-size and latency budget—and audio must not leave the device—the recoverable detection performance lies in the 0.38–0.40 macro-F1 range. Figure 2 makes this explicit by locating both models in the parameter-count vs. F1 space alongside server-class baselines; the on-device region is otherwise unoccupied in the published literature on SEP-28K.

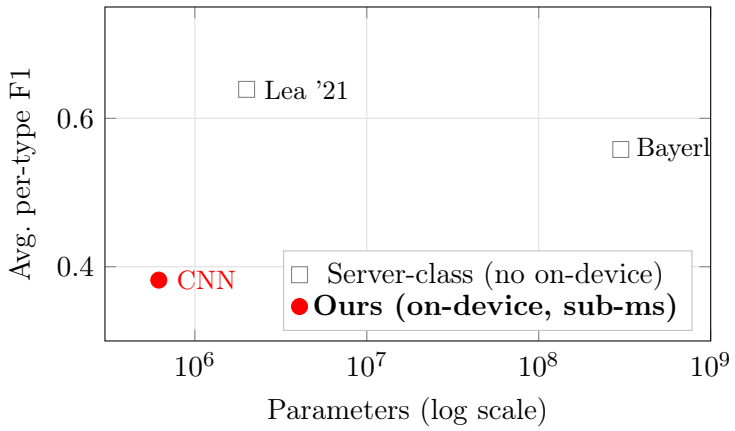

Figure 2: Accuracy–efficiency positioning on SEP-28K. F1 values are per-type averages under differing evaluation protocols and are not directly comparable as absolute numbers; the point of the figure is to locate each system in the design space. Our on-device models (red) occupy an otherwise unoccupied region: the smallest published multi-type disfluency classifiers *with* sub-millisecond CoreML inference. Server-class systems (grey) report higher F1 but have no published on-device path.

Two observations define the picture.

(i) Against the closest speaker-independent baseline, the efficiency cost is moderate. The speaker-independent SVM [6] reaches roughly macro-F1 0.50. Our CNN reaches 0.382 at 617K parameters with no server dependency; our ResNet-18 reaches 0.404 at 11.2M parameters. Per-type, our ResNet-18 matches the SVM baseline on Prolongation (0.453 vs. 0.460) and comes within 0.15 F1 on Interjection. Per-type ordering agrees with prior work: Interjection is easiest; Word Repetition and Block are hardest.

(ii) Against 300M+ parameter server-class models, the efficiency gap is four orders of magnitude. Bayerl et al.'s wav2vec 2.0 system [2] reaches higher F1 but requires a 300M+ parameter backbone. Our CNN is 486× smaller in parameter count and roughly $2.5\times10^5\times$ smaller on disk. For deployments where the model must live inside a mobile app bundle, run on an older device and never upload audio, this efficiency gap is decisive.

6.3. On-Device Latency and Compute-Unit Sweep

Rather than report a single latency number, we measure each trained model across all four CoreML compute-unit configurations (`CPU_ONLY`, `CPU_AND_GPU`, `CPU_AND_NE`, `ALL`) on an Apple M1 Max (10 cores: 8 performance + 2 efficiency; 64 GB; macOS 26; `coremltools` 9.0) and on three consumer iPhones

Table 6: CoreML inference latency on Apple M1 Max, 500 trials per compute path. "RT factor" is 3 s / mean for 3-second SEP-28K clips. Best path per model in **bold**. *NE* = Apple Neural Engine.

Model	Path	Mean ± Std (ms)	Median (ms)	P95 (ms)	RT factor
CNN	CPU	1.42 ± 0.06	1.41	1.52	2,120×
	CPU+GPU	0.78 ± 0.18	0.72	1.21	3,836×
	CPU+NE	$\mathbf{0.50 \pm 0.20}$	**0.38**	**0.79**	**6,012**×
	ALL	0.50 ± 0.20	0.38	0.79	5,952×
RN18	CPU	1.67 ± 0.23	1.64	1.89	1,793×
	CPU+GPU	3.37 ± 1.32	3.36	4.74	890×
	CPU+NE	0.98 ± 0.40	0.78	1.78	3,067×
	ALL	$\mathbf{0.89 \pm 0.35}$	**0.71**	**1.66**	**3,378**×

spanning three Neural Engine generations: iPhone 17 Pro Max (iPhone18,2; A19 Pro; iOS 26.4), iPhone 16e (iPhone17,5; A18; iOS 26.4), and iPhone SE 3rd Gen (iPhone14,6; A15; iOS 18.6.1). Each configuration uses 20 warmup inferences followed by 500 timed trials on a fixed mel-spectrogram input of shape $(1, 1, 128, 94)$. Before each latency sweep, a deterministic sanity tensor is verified against embedded CPU reference logits; all 24 on-device checks (3 iPhones × 2 models × 4 paths) pass with max $|\Delta| = 0.008 \ll 0.050$ tolerance. M1 Max results appear in Table 6; iPhone results in Table 7 and Figure 3.

Table 7 extends the sweep to three iPhone generations. All CNN paths reach sub-millisecond best-path latency on every device; ResNet-18 on A18 and A19 Pro also achieves sub-millisecond latency, with only A15 ResNet-18 exceeding 1 ms (1.253 ms, still a 2,393× real-time factor). Figure 3 plots Neural Engine latency (`CPU_AND_NE`) versus hardware generation.

Four observations follow from the combined sweep.

(i) The Neural Engine consistently accelerates inference across all tested hardware generations. Relative to `CPU_ONLY`, `CPU_AND_NE` cuts CNN mean latency by 2.3× on A15, 3.7× on A18, and 2.5× on A19 Pro. For ResNet-18 the NE gain is 1.0× on A15 (indicating the early NE generation offers limited advantage over performance CPU cores at ∼1 G FLOPs at this working-set size) but reaches 1.5× on A18/A19 Pro. On M1 Max, NE provides 2.8× gain for CNN and 1.7× for ResNet-18.

(ii) CoreML compute-unit scheduling diverges between desktop and mo-

Table 7: Mean CoreML inference latency (ms) across three iPhone generations, 500 trials per path. Bold = per-device best. On all three iPhones, `CPU_AND_GPU` produces bitwise-identical logits to `CPU_ONLY` (mobile scheduler veto; see Section 6.3).

Device (SoC)	Compute path	CNN (ms)	RN18 (ms)
17 Pro Max (A19 Pro)	CPU only	0.559	0.869
	CPU+GPU	0.569	0.840
	CPU+NE	**0.225**	0.524
	ALL	0.234	**0.525**
16e (A18)	CPU only	0.961	1.004
	CPU+GPU	0.971	1.043
	CPU+NE	**0.258**	**0.679**
	ALL	0.281	0.681
SE 3rd Gen (A15)	CPU only	1.497	1.308
	CPU+GPU	1.488	1.308
	CPU+NE	0.649	1.256
	ALL	**0.635**	**1.253**

bile Silicon. On M1 Max, routing ResNet-18 through `CPU_AND_GPU` adds 2.0× latency versus `CPU_ONLY` (3.37 vs. 1.67 ms): Metal command-encoding overhead is never amortized at this working-set size. On all three iPhones, `CPU_AND_GPU` and `CPU_ONLY` produce bitwise-identical logits across all 500 trials—the mobile CoreML scheduler silently routes GPU requests to CPU. The practical consequence: `CPU_AND_GPU` should be avoided on desktop Apple Silicon for sub-10M-parameter workloads; on mobile it has no effect.

(iii) The `ALL` policy is a safe default on all tested hardware. It matches or beats any hand-picked compute-unit path for both models on all four devices. Desktop/mobile divergence in GPU handling is absorbed transparently by the scheduler.

(iv) Neural Engine throughput provides deployment headroom across all tested devices. CNN NE latency spans 0.225–0.635 ms from A19 Pro to A15 (2022)—real-time factors between 4,700× and 13,333×. Even the oldest device in our sweep leaves the speech processing pipeline well within its latency budget.

All CNN models and A18/A19 Pro ResNet-18 achieve sub-millisecond mean inference; A15 ResNet-18 reaches 1.253 ms, a 2,393× real-time factor—

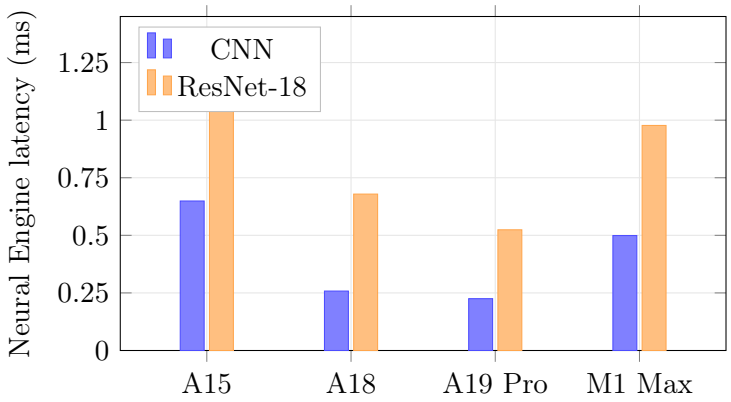

Figure 3: Neural Engine latency (`CPU_AND_NE`, mean of 500 trials) vs. hardware generation. The A19 Pro NE outperforms M1 Max NE by 2.2× on CNN (0.225 vs. 0.499 ms) and 1.9× on ResNet-18 (0.524 vs. 0.977 ms), indicating that Neural Engine throughput at this workload scale is not bounded by desktop silicon. M1 Max values from Table 6.

leaving ample headroom for mel-spectrogram extraction, voice-activity detection, stress analysis and UI rendering. This cross-generational sweep provides a concrete latency reference for devices spanning 2022 (A15) to 2025 (A19 Pro) and documents the desktop/mobile scheduler divergence under `CPU_AND_GPU` that would otherwise require empirical discovery at integration time.

6.4. Cross-Platform Export Fidelity

Latency benchmarks are only meaningful if the benchmarked CoreML model produces outputs consistent with the trained PyTorch checkpoint. We verify this numerically before reporting any latency.

6.4.1. PyTorch vs. CoreML on test-fold spectrograms

We reproduce fold 0 via `StratifiedGroupKFold` (n_splits=5, shuffle=True, seed=42) and feed $N = 500$ held-out test-fold spectrograms through both the PyTorch checkpoint and the exported `.mlpackage` (CoreML CPU path, `coremltools` 9.0). For each spectrogram we record the five raw logits from both backends and compute the element-wise absolute deviation $|\Delta\ell_i|$. Results are shown in Table 8.

Both models fall well within the CoreML float16 noise floor ($5{\times}10^{-2}$). The single binarised-prediction disagreement for CNN (one cell in 2,500) produces a macro-F1 difference of 0.003—below the inter-fold cross-validation standard

Table 8: PyTorch → CoreML export fidelity on $N = 500$ test-fold spectrograms (fold 0). "Cell agreement" is the fraction of the $500 \times 5 = 2{,}500$ threshold-binarised output cells that agree. ΔF1 is the reproduced macro-F1 difference.

Model	Max $\lvert\Delta\ell\rvert$	Mean $\lvert\Delta\ell\rvert$	Cell agree.	ΔF1
CNN	1.33×10^{-2}	1.91×10^{-3}	99.96% (2499/2500)	0.003
ResNet-18	3.75×10^{-2}	4.73×10^{-3}	100.00%	0.000

deviation of 0.011. We conclude that the CoreML export is lossless at the application's operating point.

6.4.2. Cross-device sanity tensor

To enable device-side auditing without redistributing the 22 GB training corpus, we archive reference logits for a deterministic synthetic spectrogram

$$x[m, t] = \sin(0.10\, m) \cos(0.05\, t) + 0.01(m + t), \quad m \in [0, 128),\ t \in [0, 94), \quad (1)$$

per-sample standardised to zero mean and unit variance. The reference logits (CNN: $[+7.363, +0.164, +1.037, -3.787, -1.597]$; ResNet-18: $[+3.309, +0.632, -0.274, -2.285, -1.4$ are computed on CoreML CPU via `verify_coreml.py` and embedded in `BenchmarkView.swift`. Before each latency sweep begins, every compute-unit path is required to reproduce the reference within tolerance 5×10^{-2}; a failing sanity check halts the run and flags a mis-exported model. All 24 on-device sanity checks (3 iPhones × 2 models × 4 compute-unit paths) pass with a maximum observed deviation of 0.0078, confirming that the Neural Engine implementations on A15, A18 and A19 Pro are all bit-consistent with the CoreML CPU reference at the float16 noise floor.

7. Discussion

7.1. Interpreting Moderate Absolute F1 as an Operating Point

Three factors bound what any detector trained on majority-vote SEP-28K labels can achieve: (1) low inter-annotator agreement (Fleiss' κ: prolongation 0.11, block 0.25 [5]); (2) variable podcast audio quality; (3) some events span <0.5 s within a 3 s clip. Our per-type F1 ordering mirrors the inter-annotator agreement ordering, suggesting that type difficulty is intrinsic to the data, not model-specific. The speaker-independent SVM [6] reaches

macro-F1 ≈ 0.50, closer to our 0.40 than to Lea et al.'s 0.64; the remaining gap reflects the accuracy–efficiency trade-off we chose to pay. We view this result not as a failed attempt at state-of-the-art absolute accuracy, but as a concrete measurement of what is achievable under the constraints that matter for privacy-sensitive on-device deployment.

7.2. Implications for Multimodal Mobile Architectures

The absence of meaningful linear association between voice stress and disfluency (Section 5) suggests that these signals can be treated as complementary rather than redundant in multimodal systems, allowing a single mobile core to host both modules concurrently without contention.

The cross-generational latency profile (Section 6.3) has a further practical implication: any iPhone manufactured since 2022 (A15 or later) can run both models at sub-millisecond NE latency with ample real-time headroom. This paper is an Apple Silicon deployment study; the CoreML scheduler behavior and Neural Engine results are specific to this runtime and do not generalize to Android/ONNX or Qualcomm NPU deployments where inference characteristics differ. The speech-classification architectures and on-device privacy model are platform-agnostic, but the concrete latency and scheduler findings are a reference specifically for iOS/macOS practitioners.

7.3. Privacy and Deployment Ethics

All inference occurs locally on the user's device. No audio data is transmitted, which substantially reduces privacy risk and eliminates network-connectivity dependencies. On-device processing also provides sub-millisecond latency versus 100+ ms cloud round-trips. On-device inference is necessary but not sufficient for full regulatory compliance with user-consent, data-storage and telemetry requirements; these are application-layer concerns outside the scope of this paper. All experiments reported here use adult speech from SEP-28K; any deployment targeting minors would additionally require age-appropriate consent flows and IRB oversight.

7.4. Limitations

(1) We evaluate on 71.4% of SEP-28K due to missing episode downloads. (2) SEP-28K contains only adult podcast speech; pediatric validation (e.g., FluencyBank) is required before clinical use and before the

stress-independence results can be generalized to child populations. (3) Optimized per-class thresholds may need recalibration for different populations. (4) We do not yet exploit temporal context across consecutive clips. (5) The independence analysis tests only linear associations; non-linear or conditional dependencies between stress and disfluency cannot be ruled out. (6) This study is scoped to Apple Silicon; results for other SoC families (Qualcomm, MediaTek, Android NNAPI) require separate measurement. Real-time factors on tested devices range from 2,393× (A15 ResNet-18) to 13,333× (A19 Pro CNN), covering the full performance spectrum of current iOS devices. (7) End-to-end mel-spectrogram numerical parity between the Python/torchaudio training pipeline and the Swift/Accelerate on-device implementation has not yet been formally verified on a shared reference clip; a future element-wise comparison would further tighten the reproducibility guarantee for the preprocessing boundary.

8. Reproducibility

Models are trained with PyTorch and `sklearn.StratifiedGroupKFold` (seed 42). All hyperparameters are reported in Section 4. CoreML export uses `coremltools` 9.0 with TorchScript tracing and per-class threshold metadata on the exported package. Mel-spectrogram extraction uses matching parameters in Python/torchaudio (training) and Swift/Accelerate (inference); element-wise numerical parity on a shared reference clip has not yet been formally verified and is noted as a limitation in Section 7.4. Export fidelity is verified as described in Section 6.4. On-device iPhone benchmarks were collected using `BenchmarkView.swift` in DisfluoApp (tap the speedometer toolbar icon → "Run Full Benchmark"); raw JSON results for all three iPhones are archived in `Speech/Benchmarks/` and available from the author. The JSON schema matches `benchmark_coreml.py` for direct cross-device comparison. Code, trained CoreML packages and benchmark archives are available from the author upon reasonable request.

9. Conclusion

We have presented DisfluoSDK, a deployment-oriented framework for on-device multi-type disfluency classification occupying an otherwise unpopulated accuracy–efficiency region: a 617K-parameter CNN (macro-F1 0.382) and an 11.2M-parameter ResNet-18 (0.404), both reaching sub-millisecond

Neural Engine inference. A four-way compute-unit sweep across four hardware generations confirms this latency range from A15 through A19 Pro and reveals a desktop/mobile scheduler divergence under `CPU_AND_GPU`, making `ALL` the safe default on both platforms. PyTorch-to-CoreML export fidelity is numerically verified (cell-level agreement $\geq 99.96\%$, $\Delta\text{F1} \leq 0.003$). As an auxiliary result, voice-stress markers and disfluency labels show no practically meaningful linear associations in SEP-28K ($|r| < 0.05$, $N = 14{,}645$), supporting the architectural separation of stress and disfluency modules. Future work will validate on pediatric speech (FluencyBank), add lightweight temporal modeling, and conduct clinical pilot studies.

Acknowledgements

The preparation of this manuscript was assisted by Claude (Anthropic), used for drafting and editing portions of the text. All technical content, experimental design, implementation and analysis were conducted by the author.

References

[1] E. Yairi, N. Ambrose, Epidemiology of stuttering: 21st century advances, Journal of Fluency Disorders 38 (2) (2013) 66–87.

[2] S. P. Bayerl, et al., Cross-corpus stuttering detection as a multi-label problem, in: Proc. Interspeech, ISCA, 2023.

[3] Y. He, et al., Streaming end-to-end speech recognition for mobile devices, in: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 6381–6385.

[4] N. Cummins, S. Scherer, J. Krajewski, S. Schnieder, J. Epps, T. F. Quatieri, A review of depression and suicide risk assessment using speech analysis, Speech Communication 71 (2015) 10–49.

[5] C. Lea, V. Mitra, A. Joshi, S. Kajarekar, J. P. Bigham, SEP-28k: A dataset for stuttering event detection from podcasts with people who stutter, in: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021, pp. 6798–6802.

[6] S. P. Bayerl, et al., The influence of dataset partitioning on dysfluency detection systems, in: Proc. ACL Workshop on Computational Approaches to Linguistic Code-Switching, 2022.

[7] S. A. Sheikh, et al., Machine learning for stuttering identification: Review, challenges, and future directions, Neurocomputing 514 (2022) 385–402.

[8] Anonymous, Revisiting rule-based stuttering detection, arXiv:2508.16681, reports rule-based prolongation detection ≥97% on UCLASS, FluencyBank and SEP-28K. (2025).

[9] Apple Inc., Core ML framework documentation, `https://developer.apple.com/documentation/coreml` (2023).

[10] M. Abadi, et al., TensorFlow: A system for large-scale machine learning, in: Proc. OSDI, 2016, pp. 265–283.

[11] ONNX Runtime, ONNX Runtime Mobile, `https://onnxruntime.ai` (2023).

[12] K. R. Scherer, Vocal communication of emotion: A review of research paradigms, Speech Communication 40 (1-2) (2003) 227–256.

[13] K. Pisanski, et al., Voice parameters predict sex-specific body morphology in men and women, Animal Behaviour 112 (2016) 13–22.

[14] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[15] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, Q. V. Le, SpecAugment: A simple data augmentation method for automatic speech recognition, in: Proc. Interspeech, ISCA, 2019, pp. 2613–2617.

[16] A. de Cheveigné, H. Kawahara, YIN, a fundamental frequency estimator for speech and music, The Journal of the Acoustical Society of America 111 (4) (2002) 1917–1930.

[17] B. P. Welford, Note on a method for calculating corrected sums of squares and products, Technometrics 4 (3) (1962) 419–420.