DiacriticS: Fine-Tuning a Language Model for Arabic Diacritisation on SadeedDiac-25 Benchmark

Youssef S. Mohamed
KAUST
youssef.mohamed@kaust.edu.sa

Mahdi Alkhamis
IAU
mahdihusein219@gmail.com

Mohammad Alali
KFUPM
mo.alali006@gmail.com

Saad Alnafjan
KAU
saad707213@gmail.com

Abstract ***Arabic Text Diacritisation (ATD) feeds downstream Arabic NLP systems such as Text-to-Speech and Machine Translation, yet existing benchmarks suffer from data contamination through structural train/test overlap and from domain bias toward either Classical or Modern Standard Arabic. We present a contamination-controlled benchmarking and fine-tuning study of open-weights language models, evaluated on the domain-balanced SadeedDiac-25 benchmark and trained on a normalised Sadeed Tashkeela corpus with 0.4% overlap. Four findings emerge. First, the evaluation protocol is itself a confound: the benchmark's reference evaluator discards any paragraph in which the model altered the word count 15.4% of the benchmark for one of our fine-tuned models and is sensitive to the Unicode ordering of combining marks, so canonically identical text can be scored as an error; we therefore adopt a corrected scorer that aligns predictions to references, charges insertions and deletions, and normalises mark order. Second, a weak base model gains enormously from parameter-efficient adaptation while a strong one has little room: LoRA moves Qwen3.5-4B from 58.29% to 4.38% Diacritic Error Rate (DER), whereas Gemma 4 E4B-it enters at 4.18% already below where Qwen finishes and reaches 2.81%, narrowing the gap between them from 54.1 points to 1.6. Third, with base model and corpus held constant, full fine-tuning of Qwen3.5-0.8B outperforms LoRA on the same checkpoint (3.06% against 3.60% DER), and both beat a five-times-larger LoRA-adapted model, indicating that trainable fraction rather than parameter count is the operative variable for this task. Fourth, across every system evaluated, sentence-final case endings (I'rab) and Classical Arabic remain the dominant error sources.***

Keywords ***Arabic Diacritisation, Tashkeel, Language Models, Benchmark Contamination, Tashkeela, SadeedDiac-25, Supervised Fine-Tuning, LoRA, Evaluation Methodology***

I. INTRODUCTION

Arabic Text Diacritisation (ATD) is the automated restoration of short vowels and diacritical marks (tashkeel). Downstream Natural Language Processing (NLP) applications depend on it, among them Text-to-Speech (TTS) synthesis, Machine Translation (MT), and Information Extraction. Written Arabic routinely omits diacritics, so a single surface form can map to several pronunciations and meanings. That leaves lexical and syntactic ambiguity for every later stage to resolve.

Traditional ATD systems relied on specialized sequence-labeling or character-level encoder architectures. Recent work has moved toward general-purpose Small Language Models (SLMs) and open Large Language Models (LLMs). The benchmarks used to measure that work, though, carry data contamination, structural train/test overlaps, and a domain bias toward either Classical Arabic (CA) or Modern Standard Arabic (MSA). This project fine-tunes open-weights language models for Arabic diacritisation under strict contamination control.

II. LITERATURE REVIEW

Recent work on Arabic text processing and diacritisation varies along four axes: model architecture, multi-task objectives, training scale, and evaluation methodology.

A. Character-Based Transformers and Self-Training

The Character-based Arabic Tashkeel Transformer (CATT) [1] handles Arabic's rich morphology by skipping word-level tokenisation entirely. It is pretrained as a character-level BERT, fine-tuned on custom sequence models, and uses Noisy-Student self-training to exploit unlabelled text. Working at the character level sidesteps the Out-Of-Vocabulary (OOV) bottleneck that afflicts morphologically rich languages. The cost is computational, since pretraining, fine-tuning, and self-training iterations all have to be run, and the reported gains track data scale more closely than architecture.

B. Token Classification and Data Contamination Risks

The PTCAD ("Token Classification Is All You Need") approach [2] treats diacritisation as token classification over pretrained language representations rather than sequence generation. It runs in two phases: "pre-finetuning" on auxiliary tasks (Classical Arabic MLM, POS tagging, segmentation), then task-specific fine-tuning as a token classifier. PTCAD reported a 20% Word Error Rate (WER) reduction over prior baselines and beat general LLMs such as GPT-4. Later evaluation

(Sadeed, 2025 [3]) found substantial data overlap between the Abbad and Fadel evaluation splits, so those reported figures are inflated by train/test contamination to an unknown degree.

C. Task-Specific Small Language Models

The Sadeed study [3] argues for compact, task-specific decoder-only models instead of general-purpose LLMs. Working across 9.7M Modern Standard Arabic (MSA) tokens and 2.7M Classical Arabic (CA) tokens, it finds that domain match and data cleanliness have a large effect on the result. It is cheap to run, and it is the source of the data-leakage critique of earlier benchmarks. Its own scale is small enough that the finding still needs cross-validation on more domains.

D. Downstream Impact and Data Scale Dynamics

Two further studies measure what diacritisation costs and buys downstream:
AraXLM [4]: Treats ATD as a preprocessing step for cross-lingual plagiarism detection and semantic alignment. Comparing Fine-Tashkeel, Shakkelha, and CAMeL Tools, it finds that accurate diacritic restoration improves both semantic embedding alignment and translation fidelity.
Scaling Arabic TTS [5]: Examines how explicit diacritisation trades off against acoustic model scaling in speech synthesis. Diacritised text produces better synthesis, but scaling the data to around 4,000 hours partly compensates for its absence. Where that much audio is unavailable, and in zero-shot voice cloning, precise diacritisation still matters.

E. Deep Neural Networks and Transformers

Moving from rule-based systems to deep neural networks (DNNs) improved Arabic diacritic restoration considerably. Cherradi and El Mahajer [6] compare four character-level architectures on the Tashkeela corpus: BiLSTM, Stacked BiGRU, CBHG, and a Transformer encoder-decoder. CBHG gives fast inference and strong local sequence modeling, while the Transformer encoder-decoder scales better with more data and more capacity. That is the argument for the Transformer and LLM architectures we take up here.

III. DATASET PIPELINE & CONTAMINATION CONTROL

We kept training and evaluation data strictly separate.

Training Dataset (Sadeed Tashkeela): We trained on Sadeed Tashkeela, a filtered and normalized variant of the public Tashkeela corpus. It holds roughly 1 million diacritized examples (~53 million words) after a multi-stage cleaning pipeline that removes duplicate entries, corrects missing initial diacritics, standardizes Alif-Lam (shadda/sukun) markers, and drops non-standard character anomalies. Its overlap with the test set measures 0.4%.

Test Benchmark (SadeedDiac-25): We evaluated on SadeedDiac-25, a recent contamination-controlled benchmark built in response to leakage in earlier datasets. It contains 1,200 curated paragraphs split evenly between Modern Standard Arabic (50% MSA) and Classical Arabic (50% CA), drawn from news, sports, politics, religion, and literature. Human experts reviewed the reference annotations, and the set has zero overlap with the standard training corpora.

IV. BASELINE EVALUATION ON SADEEDDIAC-25

Before fine-tuning our target models, we ran zero-shot baselines on SadeedDiac-25 across a range of candidates. Table I gives the results.

Two patterns appear in these baselines:

1) Sensitivity to Case Endings (I'rab): Diacritic Error Rates (DER) run higher on sentence-final characters (case endings) than on internal core-word diacritics. For some models, resolving syntactic context is the main bottleneck.

2) Domain Disparity: Error rates rise sharply from MSA passages to Classical Arabic (CA) texts, which is why we train on a domain-balanced corpus.

TABLE I
BASELINE ZERO-SHOT PERFORMANCE ON SADEEDDIAC-25 BENCHMARK

Model	DER (CE)	DER (no CE)	WER (CE)	WER (no CE)
Flan-T5-Tashkeel-Small	**4.10**	**3.61**	11.86	7.88
Gemma 4 E4B-it	4.18	3.80	**9.96**	**6.78**
Tashkeel 350M-v2	6.33	5.64	15.28	10.40
Fine-Tashkeel	17.91	17.73	22.61	20.35
Fanar-1-9B-Instruct	26.14	26.35	31.48	28.81
Glonor-ByT5-Arabic	33.62	33.22	50.68	44.77
Qwen3.5-9B	39.78	41.92	59.44	55.69

gemma-3-1b-pt-10k-diacritization	50.25	50.66	55.81	52.95
Qwen3.5-4B	58.29	59.43	79.00	75.22
aya-expanse-8b	66.55	67.08	85.78	84.04
Moonlight-16B-A3B-Instruct	86.60	87.10	99.01	98.84
Qwen3.5-0.8B	92.18	92.41	99.67	99.62

V. Fine-Tuning Methodology

A. Fine-Tuning Gemma 4 E4B-it and Qwen3.5-4B via LoRA

Building on the zero-shot results in Table I, we selected two open-weights target architectures for supervised fine-tuning. Gemma 4 E4B-it had the near lowest zero-shot Diacritic Error Rate (DER) of the models we took forward, at 4.18% with case endings and 3.80% without. Qwen3.5-4B was a mid-tier baseline at 58.29% DER with case endings, built on a hybrid Gated-DeltaNet/attention backbone. Pairing a dense, per-layer-embedding architecture with a linear-attention hybrid lets us test whether the adaptation regime or the backbone itself drives diacritisation quality. We ran Supervised Fine-Tuning (SFT) with LoRA on the normalized Sadeed Tashkeela training corpus (~53M words), applying each adaptation regime to both target models under matched training conditions and the same contamination control.

Low-Rank Adaptation (LoRA) applies rank-decomposition matrices to the linear projections in the attention and MLP blocks while the base weights stay frozen in bf16. Both target models use the same seven leaf-level projections (q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj), matched by full module path and filtered to true nn.Linear layers so adapters land only on real projections. On Qwen3.5-4B this resolves to 128 adapted matrices: q/k/v/o on its 8 full-attention blocks and gate/up/down on all 32 MLP blocks (its Gated-DeltaNet blocks use different projection names and are not targeted). On Gemma 4 E4B-it it resolves to 258 adapted matrices within the text decoder; the vision tower and audio encoder reuse the same seven leaf names but wrap them in a non-Linear module, so the filter skips them automatically and those pathways stay untouched. The full hyperparameter set is in Table II. Effective batch size is fixed at 96 for both models, with per-device batch and gradient accumulation adjusted for each model's memory footprint (8 per-device x 4 accumulation x 3 GPUs for Qwen3.5-4B; 4 x 8 x 3 for Gemma 4 E4B-it) so the gradient each optimizer step is computed from stays the same size for both runs.

TABLE II
LORA HYPERPARAMETER CONFIGURATION FOR TARGET MODELS

Hyperparameter	Gemma 4 E4B-it	Qwen3.5-4B
Rank (r)	16	16
Alpha (α)	32	32
Dropout	0.05	0.05
Target modules	q, k, v, o, gate, up, down	q, k, v, o, gate, up, down
Adapted matrices	258	128
Batch (per-dev. x accum x GPUs)	4 x 8 x 3 = 96	8 x 4 x 3 = 96
Learning rate	2e-4	2e-4
LR schedule, warmup	cosine, 5% warmup ratio	cosine, 5% warmup ratio
Weight decay	0.01	0.01
Epochs	1	1
Max sequence length	1024	1024
Optimizer	adamw_torch	adamw_torch
Base precision	bf16 (frozen)	bf16 (frozen)

B. Evaluation Metrics Computation

Standardized evaluation metrics on SadeedDiac-25 are defined as follows:

$$DER = (\text{Incorrect Diacritics} / \text{Total Diacritised Chars}) \times 100 \quad (1)$$

$$WER = (\text{Incorrect Words} / \text{Total Words}) \times 100 \quad (2)$$

We report both metrics with and without Case Endings (CE). For each fine-tuned model, both are computed on three splits: the SadeedDiac-25 benchmark, the full Sadeed Tashkeela test split (2,485 paragraphs), and a fixed 500-paragraph sample of the training split. A train/test gap in these otherwise-identical metrics is a direct overfitting signal.

VI. Evaluation Protocol as a Confound

Reported diacritisation scores are only comparable when they are produced by the same scorer. While reproducing published baselines we found that the reference implementation distributed with the SadeedDiac-25 benchmark and the corrected scorer used throughout this report disagree substantially on identical model output. The disagreement is not a matter of tuning; the two implementations answer different questions, and the difference is large enough to reorder a leaderboard.

A. Sentence-Level Discarding

The reference evaluator aligns a prediction to its reference positionally. It first splits both strings into Arabic tokens and raises an error if the two token lists differ in length, and raises again if the consonantal skeletons of any aligned pair differ. The enclosing loop catches that error, emits a warning, and continues to the next paragraph. The practical consequence is that any paragraph in which the model inserted, dropped, or reordered a word contributes nothing to either the numerator or the denominator: it is removed from the benchmark rather than penalised. A model that hallucinates is therefore scored only on the paragraphs it did not hallucinate on.

The corrected scorer instead aligns the two token sequences on their consonantal skeletons and keeps every reference word. Insertions surface as (none, prediction) pairs and are charged in full, deletions as (reference, none) pairs, and the DER denominator is fixed by the reference so it cannot move with the prediction. Table III makes the divergence concrete on a single four-word sentence.

TABLE III
SCORER BEHAVIOUR ON A CONTROLLED FOUR-WORD EXAMPLE

Prediction differs from reference by	Sadeed DER	Sadeed WER	Corrected DER	Corrected WER
A. nothing (exact match)	0.00	0.00	0.00	0.00
B. one wrong case ending	5.26	25.00	5.26	25.00
C. one word dropped	discarded	discarded	15.79	25.00
D. two words hallucinated	discarded	discarded	29.63	33.33
E. explanatory commentary appended	discarded	discarded	32.14	42.86

Rows A and B, where the word count is preserved, agree to the second decimal: the corrected scorer is the same metric, not a different one. Rows C to E are exactly the failure modes that distinguish a fluent instruction-tuned model from a faithful diacritiser, and they are the rows the reference implementation removes.

The effect is not hypothetical. Scoring the 1,200 SadeedDiac-25 predictions of our Qwen3.5-4B LoRA model at the 50% data fraction, the reference evaluator discards 185 paragraphs, 15.4% of the benchmark, and reports its metrics on the surviving 1,015 (Table IV).

TABLE IV
THE SAME 1,200 PREDICTIONS UNDER BOTH SCORERS (QWEN3.5-4B LORA, 50%)

Scorer / coverage	Scored	DER (CE)	DER (no CE)	WER (CE)	WER (no CE)
Sadeed evaluator	1,015 of 1,200	7.04	5.21	26.32	19.82
Corrected scorer, same subset	1,015	1.88	1.50	5.85	3.30
Corrected scorer, full benchmark	1,200	4.38	4.07	8.18	5.57

Row 1 versus row 2 isolates the scoring rules, since both cover the identical 1,015 paragraphs. Row 2 versus row 3 isolates the cost of the 185 discarded paragraphs, which are the hardest in the set: including them more than doubles DER.

B. Sensitivity to Combining-Mark Order

The second difference is subtler and affects even paragraphs the reference evaluator accepts. Arabic gemination is written as a shadda plus a short vowel on the same consonant, and the two combining marks may appear in either order in the underlying byte stream. Unicode treats the orderings as equivalent, since both normalise to the same NFC form, and no reader can tell them apart. The reference implementation walks the string left to right and keeps only the first mark it meets per letter unless that mark was the shadda, so the sequence shadda-then-fatha is read as gemination with a vowel, while fatha-then-shadda is read as a bare fatha and the shadda is dropped. A model that happens to emit the second ordering is charged an error on every geminated letter it gets right. The corrected scorer collects all marks attached to a letter and sorts them, so canonically equivalent spellings compare equal by construction. This, together with punctuation and numeral normalisation, accounts for the residual gap between rows 1 and 2 of Table IV.

C. Consequence for Cross-Paper Comparison

Because the two protocols differ this much, figures quoted from the Sadeed study and from commercial systems evaluated under its harness cannot be placed in the same table as ours without qualification, and we have deliberately excluded them from Table I. All numbers we report are produced by a single implementation, applied identically to every model and every split, so that comparisons within this report are internally valid even where they cannot be aligned with external literature.

VII. Adaptation Regime: LoRA versus Full Fine-Tuning

Sections IV and V established that Gemma 4 E4B-it is already a competent zero-shot diacritiser while Qwen3.5-4B is not. That asymmetry makes it difficult to separate the contribution of the adaptation regime from the contribution of the base model. We therefore added a third system in which the regime is the only variable of interest: Qwen3.5-0.8B, a sub-billion-parameter model small enough to fine-tune in full on a single node.

Its zero-shot behaviour establishes the floor. Prompted with the same instruction used for every other model in Table I, and decoded greedily, the stock checkpoint reaches 92.18% DER and 99.67% WER, which is effectively no diacritisation at all. Two implementation details were necessary to obtain even that number honestly. The checkpoint ships no generation configuration and its declared end-of-sequence token is not the token that terminates a chat turn, so without supplying both, generation runs to the token cap and every truncated tail is scored as a deletion. Padding and truncation must also be applied on the left, because right truncation removes the trailing turn marker that instructs the model to answer, causing it to continue the prompt instead. Both faults produce plausible-looking but meaningless scores.

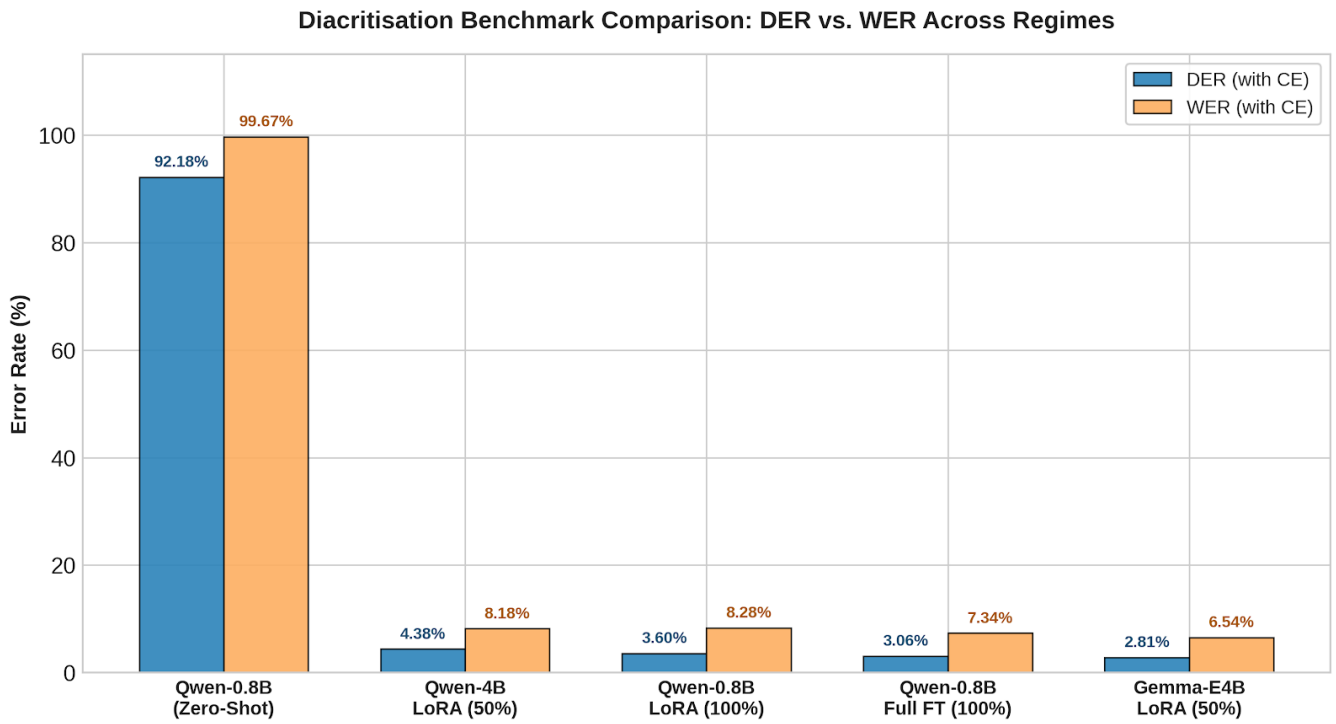

Fig. 1. Diacritic and word error rates on SadeedDiac-25 across adaptation regimes and model capacities, with case endings. The zero-shot 0.8B bar is the untrained baseline; every other bar is a completed run scored by the corrected scorer of Section VI. Note that WER falls far more slowly than DER a single missed case ending is enough to mark a whole word wrong.

TABLE V
QWEN3.5-0.8B UNDER THREE ADAPTATION REGIMES (DER / WER, WITH CASE ENDINGS)

Regime	Data	Train (500)	Test (2,485)	SadeedDiac-25 (1,200)
None (zero-shot)				92.18 / 99.67
LoRA	100%			3.60 / 8.28
Full fine-tuning	100%	0.94 / 2.28	2.52 / 5.62	3.06 / 7.34
Qwen3.5-4B, LoRA (reference)	50%	1.94 / 3.38	13.74 / 16.48	4.38 / 8.18

Train and test splits for the LoRA row have not been scored; only the benchmark has. The final row is the closest larger comparator, a five-times-larger model adapted with LoRA on half the corpus.

Full fine-tuning moves the 0.8B model from 92.18% to 3.06% DER on the benchmark, a reduction of roughly thirty-fold, and the resulting system is competitive with every adapted 4B model we trained. The train-to-benchmark gap is informative: 0.94% on a held-in training sample against 3.06% on the contamination-controlled benchmark is a modest spread, indicating the model generalised rather than memorised the outcome the 0.4% corpus overlap was designed to make interpretable.

Applying LoRA to the same 0.8B checkpoint, on the same corpus, reaches 3.60% DER. Base model, data, prompt and scorer are identical; what differs is the fraction of weights allowed to move and, necessarily, the optimiser settings that go with it (2e-5 with 3% warmup for the full fine-tune against 2e-4 with 5% for LoRA, since a rate tuned for low-rank adapters diverges when applied to every weight). Full fine-tuning wins by 0.54 DER points. The margin is modest, which is itself the result: a rank-16 adapter recovers roughly 94% of the benefit of updating every weight, at a fraction of the optimiser memory. Both 0.8B systems also beat the 4B model adapted with LoRA on half the corpus (4.38%), at a fifth of the parameter count. For a task as narrow and surface-oriented as diacritic restoration, capacity is not the binding constraint; how much of the model is free to move, and how much data it sees, matter more.

VIII. Architecture and Data Scale

The two 4B-class targets differ in more than weights. Gemma 4 E4B-it is a dense decoder with per-layer embeddings, and LoRA resolves to 258 adapted matrices across its text decoder. Qwen3.5-4B is a hybrid that interleaves Gated-DeltaNet blocks with full attention; only eight of its blocks expose the attention projections we target, so the same configuration resolves to 128 adapted matrices. Holding rank, alpha, dropout, learning rate, schedule, sequence length, seed, epoch count and effective batch size fixed across both runs, the adapted fraction of the network is thus roughly twice as large for Gemma. Table VI reports both models across nested training fractions drawn with a common shuffle seed, so the 10% subset is contained in the 30% subset and that in the 50% subset.

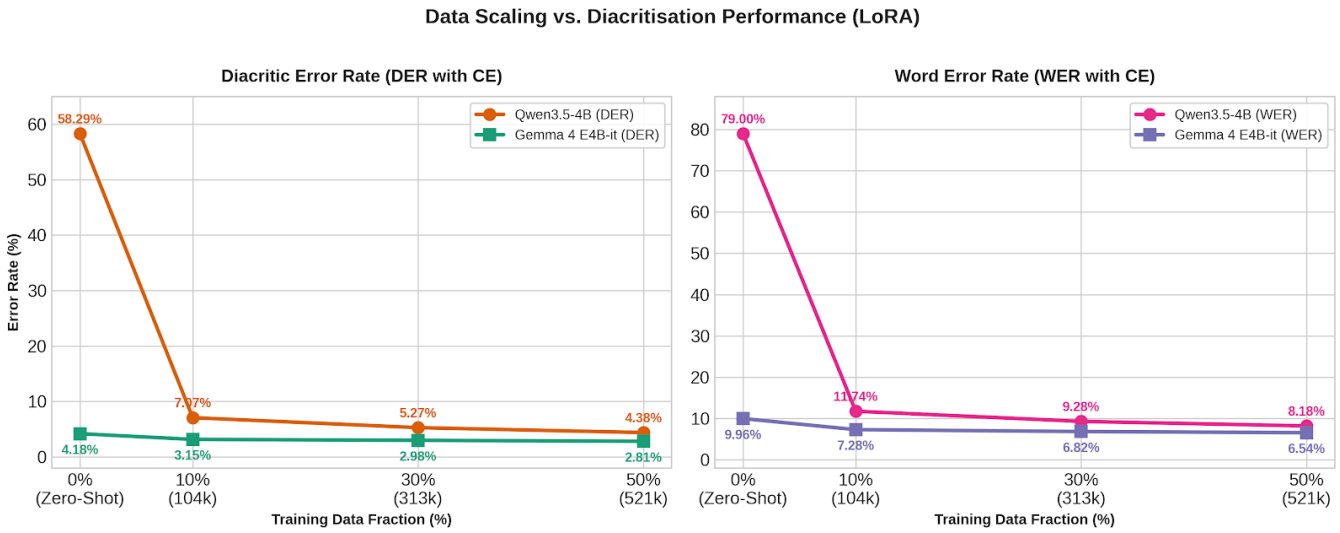

Fig. 2. Error rate against training-data fraction for both 4B targets under identical LoRA settings, with case endings. Qwen3.5-4B (upper curves) collapses from 58.29% DER at zero-shot to 7.07% after a tenth of the corpus, while Gemma 4 E4B-it (lower curves) enters below where Qwen finishes and improves far less in relative terms. Both curves are monotonic with sharply diminishing returns.

TABLE VI
DATA SCALE VERSUS ARCHITECTURE ON SADEEDDIAC-25 (DER / WER, WITH CASE ENDINGS)

Training fraction	Rows	Gemma 4 E4B-it + LoRA	Qwen3.5-4B + LoRA
0% (zero-shot)		4.18 / 9.96	58.29 / 79.00
10%	104,270	3.15 / 7.28	7.07 / 11.74
30%	312,809	2.98 / 6.82	5.27 / 9.28
50%	521,349	2.81 / 6.54	4.38 / 8.18

The dominant effect is the starting point, not the slope. Qwen3.5-4B begins at 58.29% DER and falls to 7.07% after a tenth of the corpus, 5.27% at three tenths and 4.38% at half a thirteen-fold reduction, the bulk of it purchased by the first tenth. Gemma 4 E4B-it begins at 4.18%, already far below where Qwen finishes, and improves to 3.15%, 2.98% and 2.81% across the same fractions. Both curves are monotonic with clean diminishing returns: for Qwen the first tenth buys 51.2 points, the next fifth 1.8, the final fifth 0.9. In absolute terms Gemma leads at every fraction; in relative terms it has far less to gain, and the gap between the two narrows from 54.1 points to 1.6.

Two readings are consistent with this. The first is that Gemma's instruction tuning already encodes much of the orthographic and morphological competence the task requires, so fine-tuning mostly teaches output format; the second is that its larger adapted surface (258 matrices against 128) lets a fixed-rank adapter express more of the necessary correction. The present design cannot separate these, because architecture and adapted fraction covary. Distinguishing them would require holding the adapted-matrix count fixed across the two backbones, for instance by raising Qwen's rank until the trainable parameter counts match. We flag that as the natural next step rather than something we claim to have run.

One anomaly deserves recording rather than smoothing over. Both models show a benchmark DER well below their DER on the held-out Sadeed Tashkeela test split: 4.38% against 13.74% for Qwen at 50%, and 2.81% against 12.57% for Gemma at 50%. The two evaluation sets are not interchangeable: SadeedDiac-25 is expert-reviewed and domain-balanced, while the Tashkeela test split inherits the residual annotation noise of the source corpus. A model that has learned the cleaner convention will appear to do worse on the noisier reference. We report both rather than selecting the flattering one.

IX. Qualitative Behaviour

Aggregate error rates say how often a model is wrong, not what it gets wrong. This section takes the two strongest fine-tuned systems Gemma 4 E4B-it and Qwen3.5-4B, both adapted on 50% of the corpus — and shows two correct and two incorrect predictions each, drawn from the 1,200-paragraph benchmark. The failures were chosen to be diagnosable: in every case exactly one word differs from the reference, so the error can be named rather than merely counted.

TABLE VII
GEMMA 4 E4B-IT + LORA (50%) — CORRECT AND INCORRECT PREDICTIONS

#	Reference / prediction	Outcome
111	قَوْلُهُ: ﴿الْخُرْءُ نَجِسٌ إِلَخْ﴾ (33).	Correct — exact match
184	قَوْلُهُ تَعَالَى: ﴿وَإِنَّ يُونُسَ لَمِنَ الْمُرْسَلِينَ﴾.	Correct — exact match
490	ref: فَلِلْمُشْتَرِي أَخْذُهُ بِثَمَنِهِ وَقِسْطِ الزَّائِدِ. pred: فَلِلْمُشْتَرِي أَخْذُهُ بِثَمَنِهِ وَقِسْطِ الزَّائِدِ.	Wrong — وَقِسْطِ read as وَقِسْطَ. Both the stem vowel and the case ending change: the model parses the word as a genitive continuing بِثَمَنِهِ rather than a nominative coordinated with أَخْذُهُ.
161	ref: وَضْعَهَا وَجَبَ غَسْلُ مُقَدَّمِهِ فَقَطْ. pred: وَضَعَهَا وَجَبَ غَسْلُ مُقَدَّمِهِ فَقَطْ.	Wrong — the verbal noun وَضْعَهَا (“its placing”) read as the perfect verb وَضَعَهَا (“he placed it”). Identical consonants; only the wider syntax distinguishes them.

Both Gemma failures are syntactic rather than orthographic. In 490 every letter inside the word is plausible in isolation; what fixes the reading is which earlier noun the word is coordinated with. In 161 the consonantal skeleton و-ض-ع-ه-ا admits a noun and a verb, and choosing between them requires resolving the clause, not the word. This is the same failure mode the aggregate numbers report as the gap between DER with and without case endings.

TABLE VIII
QWEN3.5-4B + LORA (50%) — CORRECT AND INCORRECT PREDICTIONS

#	Reference / prediction	Outcome
41	قَوْلُهُ: ﴿وَلَا تُؤْخَذُ الْجِزْيَةُ مِنْ نَصَارَى بَنِي تَغْلِبَ (هَذَا الْمَذْهَبُ).﴾	Correct — exact match
95	قَوْلُهُ: أَرْسَلَ لَهُ مَمْسُوحًا (أَيْ: وُجُوبًا).	Correct — exact match
1069	ref: … “الْكُفْرَ بِقَانُونٍ يُجَرِّمُ ” pred: … “الْكُفْرَ بِقَانُونٍ يَجْرِمُ ”	Wrong — Form II يُجَرِّمُ (“criminalises”) read as Form I يَجْرِمُ. A modern legal sense the classical training corpus rarely carries; this is a Modern Standard Arabic news headline.
597	ref: قَوْلُهُ: ﴿وَإِنْ خَلَّفَ أَبَوَيْنِ كَافِرَيْنِ، وَابْنَيْنِ مُسْلِمِينَ﴾. pred: قَوْلُهُ: ﴿وَإِنْ خَلَّفَ أَبَوَيْنِ كَافِرَيْنِ، وَابْنَيْنِ مُسْلِمَيْنِ﴾.	Wrong — sound plural مُسْلِمِينَ read as dual مُسْلِمَيْنِ, over-agreeing with the three duals that precede it (أَبَوَيْنِ، كَافِرَيْنِ، ابْنَيْنِ). Gemma makes the identical error on this paragraph.

Qwen's two failures separate along register. Paragraph 1069 is Modern Standard Arabic and the error is lexical: the fine-tuning corpus is overwhelmingly Classical, so a contemporary legislative verb is under-represented. Paragraph 597 is Classical and the error is agreement — the model carries a run of dual endings one word too far. That both models fail identically on 597 suggests the cue is genuinely ambiguous at the paragraph level rather than a weakness of either backbone.

One further observation comes from paragraphs the corrected scorer marks as exact but which are not byte-identical. In benchmark paragraph 493 Qwen writes the shadda and the short vowel of a geminated letter in the opposite Unicode order from the reference. The two strings normalise to the same NFC form and no reader can distinguish them, and the scorer of Section VI treats them as equal by sorting the marks on each letter. The benchmark's own evaluator would have charged an error on every geminated letter in that paragraph. The behaviour described in Section VI is therefore not hypothetical: it occurs in the output of the models measured here.

X. Conclusion

This phase set out to fine-tune open-weights language models for Arabic diacritisation under strict contamination control, and it produced one result we did not anticipate. The evaluation protocol turned out to be a larger source of variance than several of the modelling choices it was meant to arbitrate. An evaluator that discards any paragraph in which the model altered the word count removes 15.4% of the benchmark for one of our systems, and removes precisely the failures that separate a fluent generator from a faithful diacritiser; an evaluator sensitive to combining-mark order penalises text that Unicode itself declares identical. Neither behaviour is visible in a reported score. We consider the corrected scorer, applied uniformly to every model and split in this report, to be the more defensible instrument, and we regard cross-paper comparisons made without reconciling harnesses as unsafe.

On the modelling side, three conclusions are supported. Parameter-efficient adaptation is sufficient to close most of the gap for a weak starting model: LoRA moves Qwen3.5-4B from 58.29% to 4.38% DER, with the majority of that gain realised on

the first tenth of the corpus. A strong instruction-tuned starting point compresses the available headroom rather than enlarging it: Gemma 4 E4B-it enters at 4.18% and reaches 2.81%, leading at every fraction while gaining far less in relative terms. And model size is not the binding constraint: on Qwen3.5-0.8B, with corpus and prompt held constant, full fine-tuning reaches 3.06% DER against LoRA's 3.60%, and both beat a five-times-larger LoRA-adapted model so the trainable fraction, not the parameter count, is the operative variable.

Three limitations bound these conclusions. The architecture comparison in Section VIII confounds backbone with adapted surface a fixed-rank adapter resolves to 258 matrices on Gemma against 128 on Qwen so the two cannot be separated without matching trainable parameter counts across backbones, which we flag as the natural next experiment rather than one we ran. The LoRA-versus-full comparison in Section VII holds model, data and prompt constant but not learning rate or warmup, which must differ between the two regimes. And the train and test splits for the Qwen3.5-0.8B LoRA run remain unscored, so that row rests on the benchmark alone. Across every system evaluated, sentence-final case endings and Classical Arabic remain the dominant residual error sources, and are where further gains are most likely to be found.

Project Resources

Project website: https://diacritics.vercel.app/

Code and evaluation harness: https://github.com/Psycodem/DiacriticS-Language-Model-for-Arabic-Diacritisation

REFERENCES

[1] Abbad et al., "Character-based Arabic Tashkeel Transformer (CATT)," github.com/abjadai/catt, 2023.

[2] Author et al., "Arabic Text Diacritization In The Age Of Transfer Learning: Token Classification Is All You Need," 2024.

[3] Author et al., "Sadeed: Advancing Arabic Diacritization Through Small Language Model," 2025.

[4] Author et al., "AraXLM: Evaluating Arabic Diacritization Tools for Cross-Language Plagiarism Detection," 2024.

[5] Author et al., "More Data, Fewer Diacritics: Scaling Arabic TTS," 2024.

[6] M. Cherradi and H. El Mahajer, "Arabic Text Diacritization Using Deep Neural Networks and Transformer-Based Architectures," Knowledge and Decision Systems with Applications, 2025.