FEIC 2026 · UNIVERSITY OF LAGOS

Deployment Analysis of a Context-Aware DeBERTa Sarcasm Detector

ONNX export, INT8 quantisation and CPU latency

Olamma U. Iheanetu and Deborah I. Adaramola
Department of Computer and Information Sciences, College of Science and Technology, Covenant University, Ota, Ogun State, Nigeria

THE PROBLEM

Context improves sarcasm detection. Its running cost is rarely reported.

1

Context helps

Detectors that read earlier turns beat utterance-only models on dialogue [3]–[5].

2

Context costs

Every added turn adds computation, and accuracy studies seldom report size or latency [6].

3

A standard fix?

Dynamic INT8 quantisation gave BERT-base about 4× smaller models with under 1% loss [7].

RESEARCH QUESTIONS

Three questions, one latency target

Q1

Size and speed

Do ONNX export and dynamic INT8 quantisation reduce model size and CPU latency, and by how much?

Q2

The target

Does the model meet a design target of 50 ms 99th-percentile (P99) latency on a CPU?

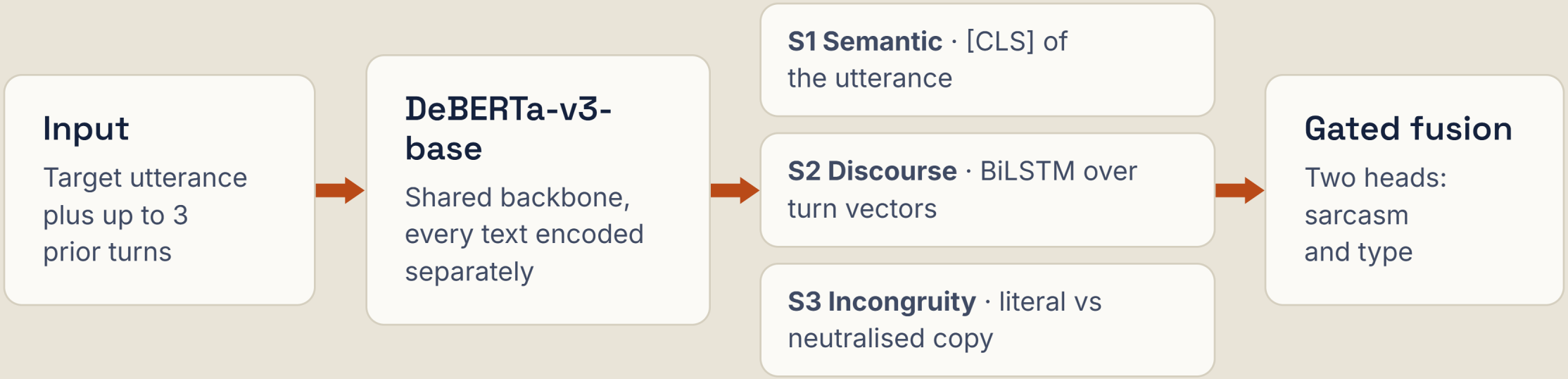
Q3

The explanation

What explains the observed size and latency?

THE MODEL UNDER STUDY

C-STAR: three streams on one shared backbone



188.9 M parameters

Macro-F1 0.7739 · AUROC 0.8556 (re-evaluated, threshold 0.40)

METHOD

A four-step post-training pipeline

1

Cast to FP32

Trained with mixed precision; a half/float mismatch appeared on CPU.

2

Export to ONNX

Opset 14, FP32 graph: 758.9 MB.

3

Quantise to INT8

ONNX Runtime dynamic. A: MatMul, Gemm, LSTM. B: MatMul, Gemm.

4

Benchmark

CPU, batch size 1, 20 warm-up and 200 timed requests.

Target: P99 ≤ 50 ms

Speed-up = P50 of PyTorch ÷ P50 of variant

RESULT 1 · MODEL SIZE

INT8 quantisation made the model larger

+10.8%

instead of the expected four-fold reduction

ONNX FP32 (baseline)
758.9 MB

INT8 variant A · MatMul, Gemm, LSTM
840.8 MB · +10.8%

INT8 variant B · MatMul, Gemm only
840.8 MB · +10.8%

Identical sizes for A and B: the LSTM operators are not the cause.

WHY THE SIZE GREW

The graph holds what the quantiser leaves alone

480

Gather operations

Sub-word embedding table (about 128,000 × 768) and relative-position embeddings stay in FP32.

394

MatMul operations

Disentangled attention computes content and position terms as separate products.

2

LSTM operators

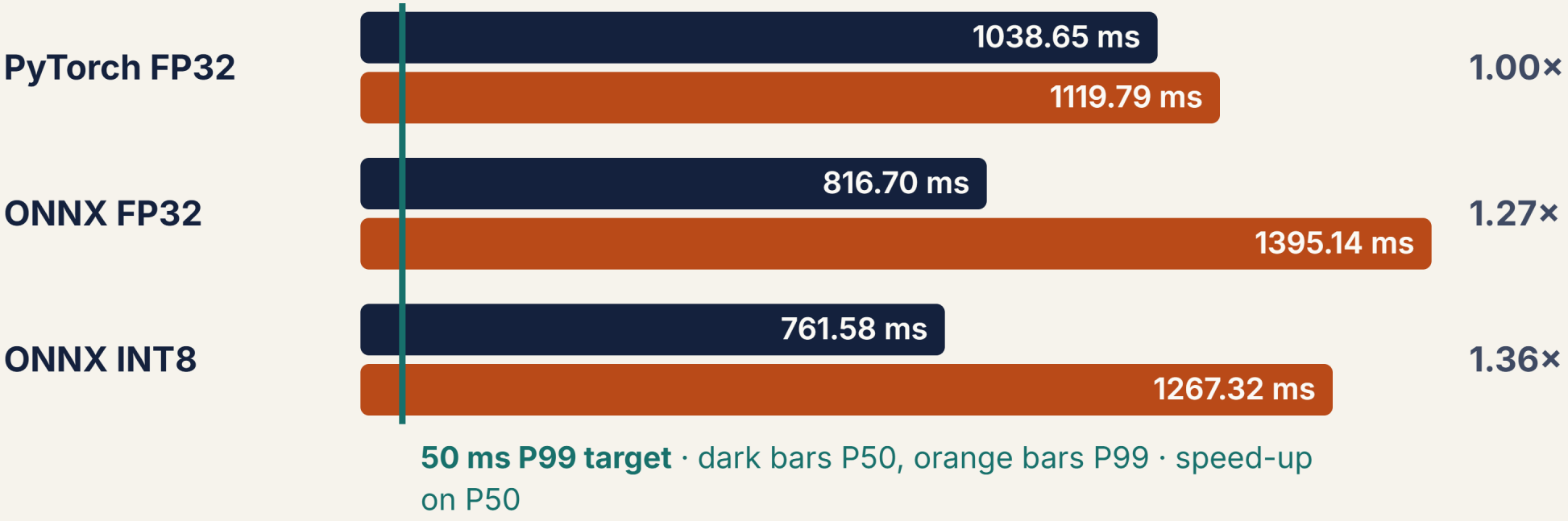
Too few to matter for size, consistent with variants A and B being identical.

Many small quantised matrices add scale and zero-point overhead that can exceed the bytes saved, while the largest weight block is not compressed.

Inferred from the operator inventory; not isolated by a controlled experiment.

RESULT 2 · CPU LATENCY

Modest speed-ups, far from 50 ms



CPU, batch size 1, 200 timed requests per variant, shared Kaggle CPU. P99 values are indicative.

RESULT 3 · SOURCE OF THE LATENCY

One prediction needs three or four backbone passes

PASS 1	PASS 2	PASS 3	PASS 4
Target utterance	Prior turn 1	Prior turn 2	Prior turn 3

BiLSTM, incongruity and fusion layers then run **once**.

Export and quantisation make each pass faster. They do not reduce the number of passes, and nothing reuses earlier turn encodings.

Request with three prior turns, as configured in the code; the target is usually longer than a context turn.

WOULD CACHING FIX IT?

Even with ideal caching, the target is still missed

Variant	P50 per pass	P99 per pass	P99 / target
PyTorch FP32	260–346 ms	280–373 ms	5.6–7.5×
ONNX FP32	204–272 ms	349–465 ms	7.0–9.3×
ONNX INT8	190–254 ms	317–422 ms	6.3–8.4×

Estimate: measured latency ÷ 4 or 3 passes. Assumes linear scaling and equal cost per pass, so it is illustrative.

6.3–8.4×

still above the 50 ms P99 target for INT8 with one fresh pass per request

A smaller or distilled encoder would also be required.

CAN SHORTER CONTEXT HELP?

Padded turns add cost without benefit

Context turns	Macro-F1
1	0.468
5	0.437
10	0.439

300-sample stratified Reddit subset with one genuine parent turn. Earlier checkpoint; the code trains with 3 turns.

Synthetic scenarios: accuracy stayed at 6/10 from one to ten turns.

Truncating to the turns that exist costs no accuracy and saves passes.

Both checks are small and illustrative.

PRINCIPAL FINDINGS

What the three questions came back with

Q1 **Size and speed**

Size grew by 10.8%. Median latency improved by at most 1.36×.

Q2 **The 50 ms target**

Missed by more than 20× in every configuration.

Q3 **The explanation**

An unquantised embedding table and hundreds of small MatMuls explain size; three or four backbone passes explain latency.

BERT-base quantisation results cannot be assumed to transfer to this hybrid architecture.

WHAT TO TRY NEXT

An ordered list of remedies to test

1

Cache turn encodings

Encode only the newest utterance afresh in a running conversation.

2

Limit the turns

Process only the turns that exist; padded turns add cost without benefit.

3

Quantise the rest

Static quantisation with calibration data, quantised embeddings, vocabulary pruning.

4

Distil the backbone

A smaller student cuts the cost of every remaining pass [13], [14].

All four are hypotheses: none was tested in this study

Each remedy should be evaluated for accuracy as well as latency.

LIMITATIONS

How far these findings reach

Shared CPU. A multi-tenant machine adds variance, so P99 is indicative. No dedicated, edge or GPU hardware.

Accuracy not re-measured. Post-quantisation accuracy was not evaluated, so a gain could not be weighed against a loss.

Narrow benchmark. Batch size 1 only, 200 timed requests per variant, one model, dynamic INT8 only.

Inferred, not isolated. The size explanation comes from graph inspection, and the per-pass estimate assumes linear scaling.

Context-length checks use synthetic scenarios and a 300-sample subset.

CONCLUSION

Real-time CPU deployment is still an open problem for this architecture

Quantisation did not compress the model: **+10.8%**, 840.8 MB.

Median latency improved by at most **1.36×**, to 761.58 ms. P99 stayed above 20× the target.

Repeated re-encoding of context dominates. Cache, truncate, quantise further, distil, then measure accuracy.

Thank you for listening.

Olamma U. Iheanetu and Deborah I. Adaramola · Covenant University

Funded by Covenant University through CUCRID.