

The CPU Inference Challenge

Models designed for accelerators can run slowly on CPUs.

DLRMv3 illustrates this challenge. **Long user interaction histories** increase the computational cost of attention.

Confidence guided selection reduces history before attention, therefore targeting efficient inference for AGI CPU.

PyTorch Layer Design

Proposed nn.Module wrapper

Wrap attention with a request-conditioned selector. Configure the confidence gate, score calibration and retention bounds as hyperparameters. Select context, gather aligned data and execute attention.

Proposed PyTorch interface

```
layer = ConfidenceGuidedAttention(
    attention=attention_layer, confidence_threshold=tau,
    min_keep_ratio=r_min, max_keep_ratio=r_max,
    spread_range=spread_bounds, gap_range=gap_bounds )
```

Inputs: history, candidates, aligned features

Output: attention features after gated selection

Preserve sequence semantics

Keep original order and positional metadata to preserve temporal user interaction information. Rebuild tensors and preserve the candidate region.

Confidence-guided attention layer

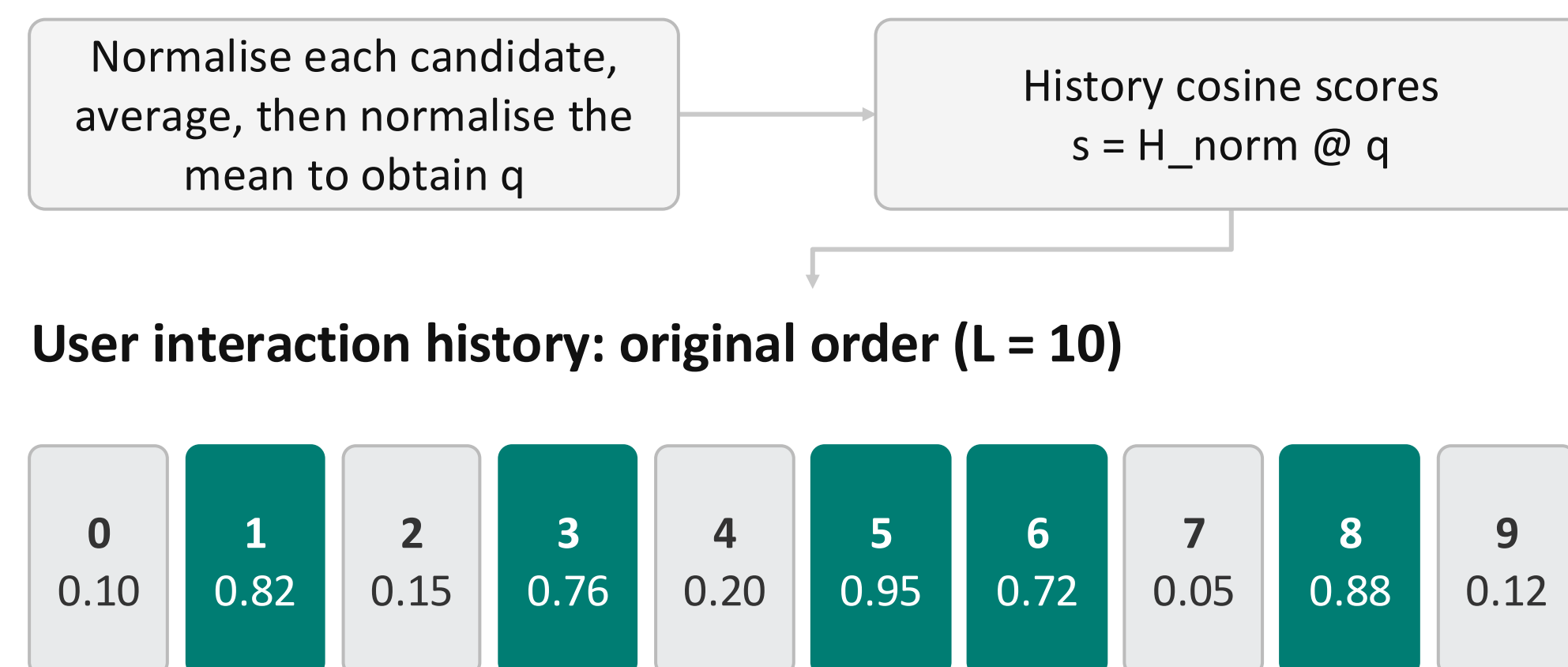
```
score_spread = Q90 - Q50
top_score_gap = Q95 - Q90

# Calibrate spread and gap using pre-determined bounds
spread_confidence = clip(
    (score_spread - spread_low)/(spread_high - spread_low), 0, 1
)
gap_confidence = clip(
    (top_score_gap - gap_low)/(gap_high - gap_low), 0, 1
)

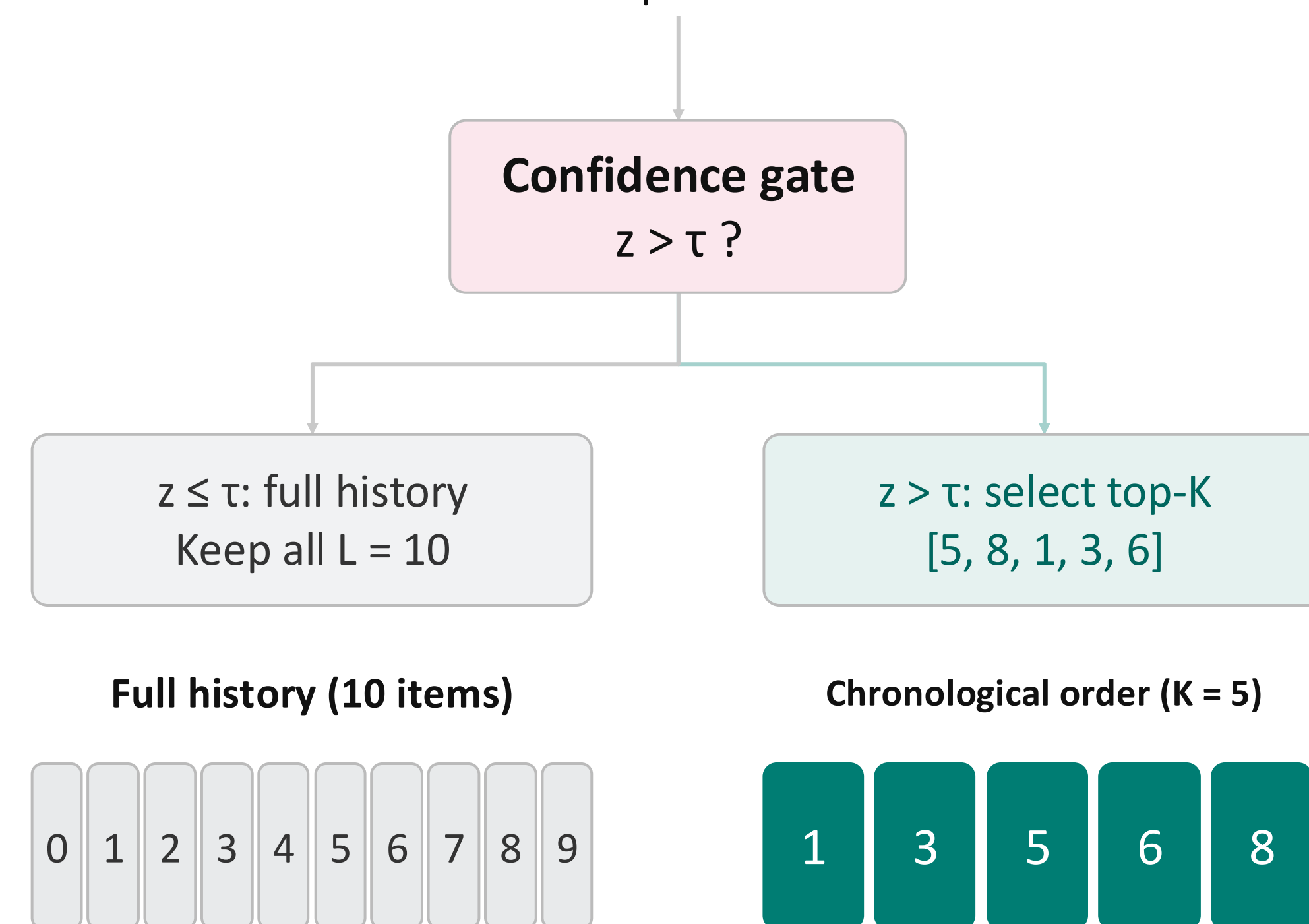
# Blend confidence using relative spread/gap strengths
spread_weight = score_spread/(score_spread + top_score_gap)
gap_weight = top_score_gap/(score_spread + top_score_gap)
pruning_confidence = (
    spread_weight * spread_confidence + gap_weight * gap_confidence
)

# Above tau, rescale confidence from [tau, 1] to [0, 1]
# Interpolate the retention ratio from r_max to r_min
if pruning_confidence > tau:
    gated_confidence = (pruning_confidence - tau) / (1 - tau)
    keep_ratio = r_max - gated_confidence * (r_max - r_min)
    keep_count = ceil(keep_ratio * history_length)
else:
    keep_count = history_length
```

Confidence Guided Selection



Quantile spread (Q90 – Q50) and upper-tail gap (Q95 – Q90) determine request confidence z

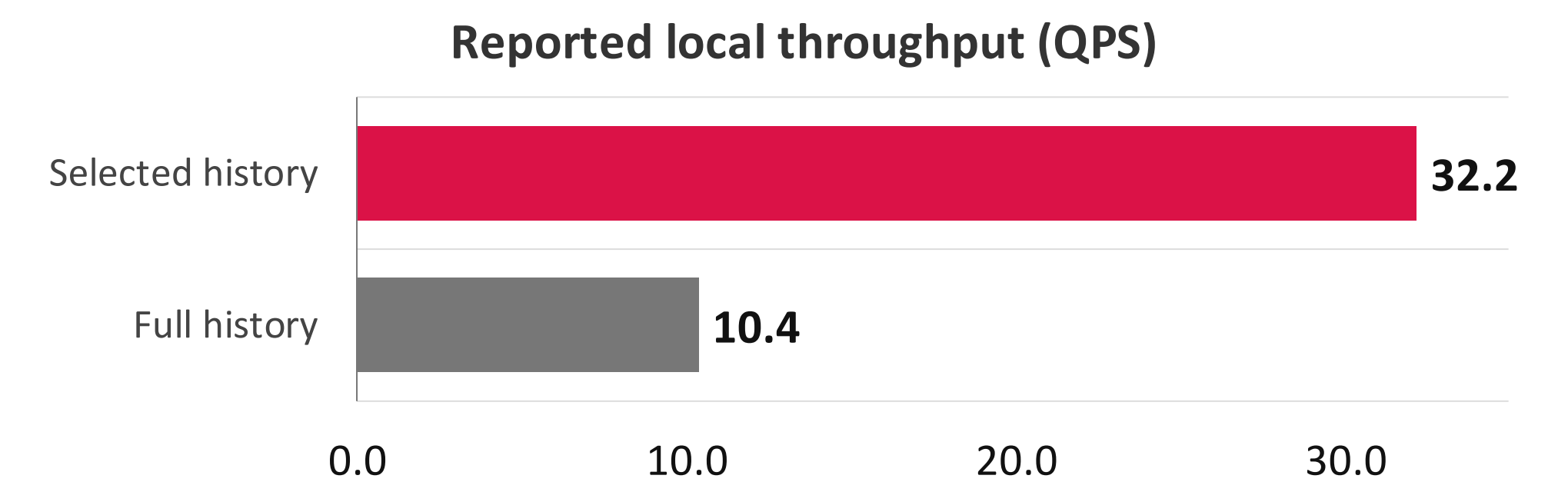


Illustrative confident request
r is the retention ratio set by confidence
 $K = \text{ceil}(rL) = 5$

Confidence controls K.
Relevance ranks the history items.

Confidence is a request-level heuristic, not a probability.

DLRMv3 Case Study



3.10× QPS (+209.73%)

Original internal average timing: –64.30%

Original run: fixed gates and weights.

Original-run local quality gates:

Metric	Observed	Local gate	Status
NE	0.8658687471	≤ 0.8677377377	PASS
Accuracy	0.6968458990	≥ 0.69581349	PASS
GAUC	0.7870746150	≥ 0.78584337	PASS

Fine-tuning the Gates

During model fine-tuning, use validation-driven search to tune τ , score-calibration ranges and retention bounds.

Evaluate each configuration on held-out data, measuring quality and CPU cost, then choose one that reduces inference cost while passing the quality gates.

Conclusions

We propose a reusable PyTorch layer, primarily for AGI CPU, with gate hyperparameters that can be tuned during model fine-tuning.