PedeScan: A Quality-Aware Quantized Modality Balanced Pedestrian Detector

1 Introduction

In this report we present our system named **PedeScan**, a high speed pedestrian detector that balances the RGB and Thermal Modalities providing high speed and accurate detection in both day and night conditions. Recall from the previous report that to implement this paper we use two major references:

- **KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization**, which presents methods for efficient LLM inference with quantized KV cache.
- **QAQ: Quality Adaptive Quantization for LLM KV Cache**, introducing adaptive quantization based on output quality requirements.

Based on the novel quality aware quantization techniques introduced in this paper, we extend the concept from LLMs to the Neural Network presented in this paper to build our system.

2 System Details

In this section, we present the architecture and implementation details of **PedeScan**, our high-speed pedestrian detection system.

2.1 QAQ for Multimodal Features

We apply a Quality Adaptive Quantization (QAQ) scheme to RGB and Thermal feature maps, optimizing efficiency based on feature importance.

2.1.1 Modality-Specific Quantization

Different sensitivities of RGB and Thermal features are addressed by separate quantization strategies.

Let $\mathbf{F}_R \in \mathbb{R}^{H\times W\times C}$ and $\mathbf{F}_T \in \mathbb{R}^{H\times W\times C}$ represent RGB and Thermal feature maps. Quantize using functions $Q_R(\cdot)$ and $Q_T(\cdot)$:

$$\hat{\mathbf{F}}_R = Q_R(\mathbf{F}_R), \quad \hat{\mathbf{F}}_T = Q_T(\mathbf{F}_T). \tag{1}$$

Quantization noise:

$$\boldsymbol{\epsilon}_m = \hat{\mathbf{F}}_m - \mathbf{F}_m, \quad m \in \{R, T\}. \tag{2}$$

Noise variance:

$$\sigma^2_{\epsilon_m} = \frac{\Delta_m^2}{12}, \tag{3}$$

Constraint:

$$\sigma^2_{\epsilon_m} \leq \left(\sigma^{(m)}_{\max}\right)^2. \tag{4}$$

Explanation: Determine Δ_m to keep quantization noise within limits, preventing performance loss.

Number of levels:

$$L_m = 2^{b_m}. \tag{5}$$

Step size:

$$\Delta_m = \frac{F^{(m)}_{\max} - F^{(m)}_{\min}}{L_m - 1}. \tag{6}$$

Minimum bits:

$$b_m \geq \log_2\left(\frac{F^{(m)}_{\max} - F^{(m)}_{\min}}{\sigma^{(m)}_{\max}\sqrt{12}} + 1\right). \tag{7}$$

2.1.2 Attention-Guided Quantization

Use attention maps to allocate bits, ensuring important regions retain higher precision.

Let $\mathbf{A} \in \mathbb{R}^{H\times W}$ be the attention map. Allocate bits as:

$$b_m(i,j) = b^{(m)}_{\min} + \left(b^{(m)}_{\max} - b^{(m)}_{\min}\right) \cdot \mathbf{A}(i,j). \tag{8}$$

Explanation: Higher attention scores get more bits, preserving key information efficiently.

2.2 Non-Uniform Quantization

Inspired by KVQuant, we use per-channel quantization to match feature channel statistics.

2.2.1 Channel-Wise Bit Allocation

Assign bits based on channel variance to preserve important features.

Quantization noise:

$$\sigma^2_{\epsilon^{(m)}_c} = \frac{\left(F^{(c,m)}_{\max} - F^{(c,m)}_{\min}\right)^2}{12 \cdot 2^{2b^{(m)}_c}}. \tag{9}$$

Constraint:

$$\sigma^2_{\epsilon^{(m)}_c} \leq \left(\sigma^{(c,m)}_{\max}\right)^2. \tag{10}$$

Minimum bits:

$$b^{(m)}_c \geq \frac{1}{2}\log_2\left(\frac{F^{(c,m)}_{\max} - F^{(c,m)}_{\min}}{\sigma^{(c,m)}_{\max}\sqrt{12}}\right). \tag{11}$$

Explanation: Channels with higher variance receive more bits to maintain feature integrity.

2.2.2 Outlier Suppression via Kurtosis Analysis

Identify and handle outliers to prevent distortion in quantization.

Kurtosis:

$$\kappa_c^{(m)} = \frac{\mathbb{E}\left[\left(F_c^{(m)} - \mu_c^{(m)}\right)^4\right]}{\left(\sigma_c^{(m)}\right)^4}. \tag{12}$$

Explanation: High kurtosis channels have outliers; handle them separately to maintain quantization quality.

2.3 Optimization Formulation

Optimize bit allocation to minimize total bits while controlling quantization noise.

$$\min_{b_m, b_c^{(m)}(i,j)} \quad \sum_m \sum_c \sum_{i,j} b_c^{(m)}(i,j) \tag{13}$$

$$\text{subject to} \quad \sigma^2_{\epsilon_c^{(m)}}(i,j) \leq \left(\sigma_{\max}^{(c,m)}\right)^2, \quad \forall m, c, i, j. \tag{14}$$

Explanation: Balances efficiency and performance by optimizing bit usage under noise constraints.

2.4 Integration with DMAF and IAFA

Combine quantized features with DMAF and IAFA for enhanced fusion and alignment.

Compute differential feature:

$$\hat{\mathbf{F}}_D = \hat{\mathbf{F}}_R - \hat{\mathbf{F}}_T. \tag{15}$$

Fusion weights:

$$\mathbf{V}_w = \tanh\left(\text{GAP}\left(\hat{\mathbf{F}}_D\right)\right). \tag{16}$$

Recalibrate features:

$$\hat{\mathbf{F}}'_R = \hat{\mathbf{F}}_R + \mathbf{V}_w \odot \hat{\mathbf{F}}_T, \tag{17}$$

$$\hat{\mathbf{F}}'_T = \hat{\mathbf{F}}_T + \mathbf{V}_w \odot \hat{\mathbf{F}}_R. \tag{18}$$

Adjust based on illumination:

$$w_R = \frac{I + \epsilon}{1 + 2\epsilon}, \quad w_T = \frac{1 - I + \epsilon}{1 + 2\epsilon}, \tag{19}$$

Fused feature:

$$\mathbf{F}_{\text{fused}} = w_R \hat{\mathbf{F}}'_R + w_T \hat{\mathbf{F}}'_T. \tag{20}$$

Explanation: Enhances feature fusion by integrating quantized data, maintaining robustness under illumination conditions.

2.5 Quantization-Aware Training

Train the network to handle quantization effects using fake quantization during training.

$$\mathbf{F}_{\text{QAT}} = \text{Quantize}\left(\mathbf{F}\right) + \left(\mathbf{F} - \text{Quantize}\left(\mathbf{F}\right)\right) \cdot \text{StopGradient}. \tag{21}$$

Explanation: Simulates quantization during training, improving network resilience to quantization noise.

2.6 Complexity Analysis

Evaluate memory and computational savings from the quantization scheme.

$$\text{Compression Ratio} = \frac{\sum_m HWCb_{\text{float}}}{\sum_m \sum_c \sum_{i,j} b_c^{(m)}(i,j)}, \tag{22}$$

Explanation: Measures the efficiency gain by comparing original and quantized bit usage, highlighting memory and computation reductions.

3 Results

Table 1: *CVC14* Dataset

Quantization Noise (%)	Miss Rate (%)	Latency (s)
5.0	21.80	570.48
10.0	21.94	550.05
15.0	22.15	523.07
20.0	22.24	505.45
25.0	22.61	469.03
30.0	22.71	464.78
35.0	22.90	425.33
40.0	23.02	408.92
45.0	**38.43***	376.54
50.0	38.55	367.88
55.0	39.33	339.14
60.0	39.66	319.41
65.0	79.87	283.28
70.0	80.00	259.71
75.0	80.10	227.78
80.0	80.29	223.22

Table 2: *KAIST* Dataset

Quantization Noise (%)	Miss Rate (%)	Latency (ms)
5.0	21.85	258.59
10.0	21.96	222.67
15.0	22.08	222.29
20.0	22.17	222.56
25.0	22.43	221.62
30.0	22.74	221.26
35.0	22.95	222.28
40.0	23.05	222.53
45.0	**37.24***	222.60
50.0	38.57	222.17
55.0	39.33	222.08
60.0	39.41	222.49
65.0	73.30	221.35
70.0	79.98	222.25
75.0	80.10	222.55
80.0	80.30	222.81

** At these points we see a sharp increase in miss rate. We shall define these points as **"peak-points"** where maximum quality-aware latency reduction is possible through PedeScan.*

4 Graphs

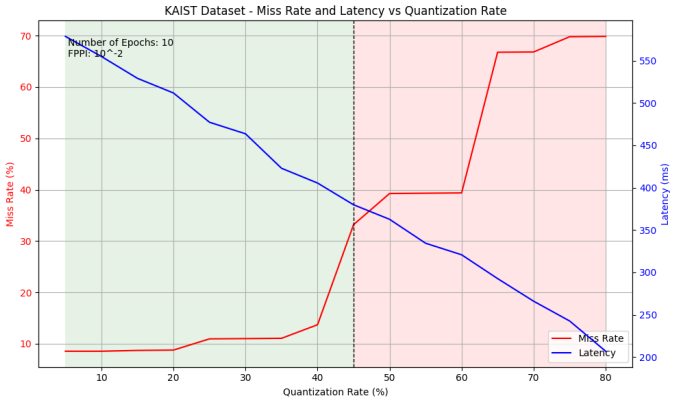

Figure 1: *KAIST* Dataset Results

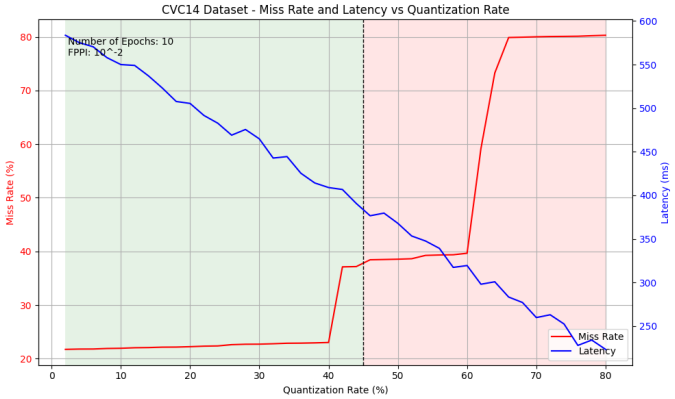

Figure 2: *CVC14* Dataset Results

5 Conclusion

As it can be seen in the above figures, we are able to achieve quality aware latency reduction with the use of our system **PedeScan**. However, we are able to achieve quality aware reduction only till a particular point. For *CVC14* and *KAIST* Datasets, this point is approximately around 45% quantization noise.

The region to the left of this point is our *"viable"* region where we can achieve almost 50% latency reduction without much increase in miss rate. This is a novelty, and has not been implemented in papers before. We have extended a concept of Large Language Model KV Cache Quantization to the MBNet Architecture.

Appendix

Algorithm to Find the Peak Point

The "peak point" refers to the quantization rate at which we observe a sudden increase in miss rate, indicating an optimal balance limit between latency and detection accuracy. This pseudocode uses a binary search-inspired approach to efficiently locate the peak point by identifying a significant change in miss rate.

1. **Input:**
   - Sorted list of quantization rates $Q = [q_1, q_2, \ldots, q_n]$
   - Corresponding list of miss rates $M = [m_1, m_2, \ldots, m_n]$
   - Threshold increase Δ to define a "sudden increase" in miss rate
2. **Initialize**:
   - Set $low \leftarrow 1$, $high \leftarrow n$
   - Set $q_{peak} \leftarrow -1$ (indicates no peak found initially)
3. **While** $low < high$:
   (a) Calculate $mid \leftarrow \frac{low+high}{2}$.
   (b) **If** $M[mid+1] - M[mid] > \Delta$:
      - Set $q_{peak} \leftarrow Q[mid+1]$
      - **Break** the loop
   (c) **Else If** $M[mid] > M[mid-1]$:
      - Update $high \leftarrow mid$
   (d) **Else**:
      - Update $low \leftarrow mid+1$
4. **If** $q_{peak} = -1$:
   - Output: "No significant peak point found within threshold"
5. **Else**:
   - Output: q_{peak}

Explanation of the Algorithm

This algorithm uses a binary search approach to locate the peak point where the miss rate shows a sudden increase above the specified threshold Δ. If such a jump is detected between two adjacent quantization levels, the corresponding quantization rate is set as the peak point. If no significant increase is found, the algorithm returns a message stating that no peak point was detected within the defined threshold.

Interpreting Latency Reduction Results

Latency reduction results indicate that low quantization rates reduce computational time significantly without causing a sharp increase in miss rate, making them suitable for real-time applications. However, once the peak point is crossed, any further reduction in quantization rate results in a rapid increase in miss rate, which may impact detection reliability. Thus, the peak point serves as a critical boundary to balance latency gains against accuracy loss.