Consistency in Decision-Making under Quantised Constraints

CS 5891: Algorithms for Decision Making Final Project

Youngjae Moon

VANDERBILT UNIVERSITY

Contents

Problem Statement

- Neural networks are computationally expensive for real-time decision-making on resource-constrained devices (e.g., mobile phones, IoT).
- Quantisation reduces model size and inference time but may compromise accuracy.
- Key questions:
  - How can we balance accuracy and efficiency using quantisation?
  - Can quantised models retain consistency and reliability?

VANDERBILT UNIVERSITY

Objectives

- Implement Post-Training Quantisation (PTQ) and Quantisation-Aware Training (QAT) to optimize neural networks for decision-making.
- Evaluate trade-offs between:
  - Model size
  - Inference speed
  - Decision accuracy
- Check decision consistency of quantised models using Interval Neural Networks (INNs).

VANDERBILT UNIVERSITY

Basic Quantisation Concepts

- Quantisation:
  - A process of reducing the numerical precision of weights and activations in a neural network to optimise inference speed and memory usage.
  - Commonly used formats: FP32, INT8, and mixed precision.
- Quantisation Backend:
  - Hardware-specific configurations (e.g., qnnpack, fbgemm) designed to leverage low-precision arithmetic for efficient execution.

VANDERBILT UNIVERSITY

PTQ vs QAT – Concept

- PTQ is a quantisation technique applied after a model has been fully trained.
- It reduces the precision of weights and activations (e.g., from 32-bit floating point to 8-bit integers) without modifying the model's parameters.
- Quantisation is applied statically or dynamically to improve efficiency during inference.

- QAT integrates quantisation into the training process by simulating low-precision arithmetic during forward passes.
- The model is trained or fine-tuned to adapt its parameters to quantisation-induced errors, leading to higher post-quantisation accuracy.

VANDERBILT UNIVERSITY

PTQ vs QAT – Workflow

- Train a model as usual using full precision (FP32).
- Apply quantisation techniques (e.g., dynamic or static) to reduce precision for weights and activations.
- Deploy the quantised model for inference.

- Prepare a model with quantisation-specific configurations (e.g., fake quantisation).
- Train the model while simulating quantised behaviour during forward propagation.
- Convert the model to a fully quantised version for deployment.

VANDERBILT UNIVERSITY

PTQ vs QAT – Advantages

- Ease of Use
  - No additional retraining is required, making it ideal for scenarios where training data is unavailable.
- Deployment Efficiency
  - Reduces model size and increases inference speed with minimal implementation effort.

- Higher Accuracy
  - Since the model learns to compensate for quantisation errors, accuracy loss is significantly reduced compared to PTQ.
- Customisable
  - Allows fine-grained control over quantisation strategies, enabling optimisation for specific hardware or use cases.

VANDERBILT UNIVERSITY

PTQ vs QAT – Disadvantages

- Can lead to significant accuracy degradation for certain models, especially those sensitive to precision changes.
- Not suitable for models with complex or highly non-linear architectures.

- Requires access to training data and computational resources for retraining or fine-tuning.
- Introduces additional training complexity, increasing development time and cost.

VANDERBILT UNIVERSITY

PTQ vs QAT – Use Cases

- Scenarios where training data is unavailable or access to the training pipeline is restricted.
- Suitable for large, robust models where the loss of precision has minimal impact on performance.

- Critical applications where accuracy is a priority, such as autonomous systems, medical diagnostics, or financial modelling.
- Deployment on resource-constrained hardware requiring low-precision operations.

VANDERBILT UNIVERSITY

Tools and Technologies

- Python
  - Simple programming language to use and have many libraries/frameworks
- PyTorch
  - For Neural network development
  - For quantisation – fbgemm
- Gymnasium
  - Reinforcement learning environment (CartPole)

VANDERBILT UNIVERSITY

Methodology

- Baseline Model Training:
  - Train a reinforcement learning policy on the CartPole environment using PyTorch.
- Quantisation:
  - Apply PTQ to optimize model size and inference speed.
  - Fine-tune using QAT to retain accuracy.
- Evaluation:
  - Compare baseline, PTQ, and QAT models based on size, speed, and accuracy.
- Verification:
  - Use Interval Neural Networks to check decision consistency between quantised and unquantised models.

VANDERBILT UNIVERSITY

Experimental Setup

- Environment:
  - Used the CartPole-v1 environment from Gymnasium as the testbed for evaluating policies under quantisation techniques.
- Baseline Model:
  - Implemented a fully connected neural network (PolicyNetwork) to act as the policy model.
  - Trained the baseline model using a reinforcement learning approach to maximize rewards.

VANDERBILT UNIVERSITY

Experimental Setup

- Quantisation Techniques:
  - Post-Training Quantization (PTQ):
    - Dynamically quantised the baseline model to reduce model size and improve inference speed.
  - Quantisation-Aware Training (QAT):
    - Incorporated quantisation simulation during training to improve post-quantization accuracy.

Experimental Setup

- Evaluation Metrics:
  - Reward Comparison:
    - Measured average rewards over 10 episodes for both baseline and quantized models.
  - Decision Consistency:
    - Verified that outputs of quantised models were consistent with the baseline model using interval bounds.
- Hardware:
  - Experiments conducted on CPU with PyTorch’s fbgemm quantised backend.
  - Incorporated optional GPU acceleration for baseline model evaluation.

Experimental Setup

- Interval Neural Networks (INN):
  - Used for verification of consistency between quantised and unquantised models.
  - Propagated input intervals through the network to ensure decisions remain within tolerable bounds.

VANDERBILT UNIVERSITY

Results

- Evaluating baseline model…
  - Episode 1 completed with total reward: 200.0
  - Episode 2 completed with total reward: 200.0
  - Episode 3 completed with total reward: 200.0
  - Baseline Reward: 200.0
- Evaluating PTQ model…
  - Episode 1 completed with total reward: 200.0
  - Episode 2 completed with total reward: 200.0
  - Episode 3 completed with total reward: 200.0
  - PTQ Reward: 200.0
- Evaluating QAT model…
  - Episode 1 completed with total reward: 200.0
  - Episode 2 completed with total reward: 200.0
  - Episode 3 completed with total reward: 200.0
  - QAT Reward: 200.0

VANDERBILT UNIVERSITY

Results

- Verifying consistency for PTQ...
  - Test Input 0:
  - Baseline Output: tensor([0.0090, 0.9910], grad_fn=<SoftmaxBackward0>)
  - Quantized Output Interval: tensor([-2.4092, 4.5747], grad_fn=<AddBackward0>), tensor([-2.4559, 4.5559], grad_fn=<AddBackward0>)
  - Decision Consistency (Baseline vs PTQ): False
- Verifying consistency for QAT...
  - Test Input 0:
  - Baseline Output: tensor([0.0090, 0.9910], grad_fn=<SoftmaxBackward0>)
  - Quantized Output Interval: tensor([-2.4191, 4.5817], grad_fn=<AddBackward0>), tensor([-2.4675, 4.5625], grad_fn=<AddBackward0>)
  - Decision Consistency (Baseline vs QAT): False

VANDERBILT UNIVERSITY

Future Work

- Representing real-world decision scenarios in a synthetic environment.
- Integrate advanced neural network verification tools like NNV or Marabou.
- Explore mixed-precision quantisation.
- Extend to larger, more complex neural networks.
- Verification of original neural network does not guarantee verification of quantised neural network.
- Thus, verification of quantised neural network does not guarantee verification of original neural network and interval neural network (representation of unquantised neural network).

VANDERBILT UNIVERSITY

Thank you!

Any questions?