YOLOv8 Quantization-Aware Training Configuration Guide

This guide explains how to configure and utilize the enhanced quantization-aware training (QAT) features in your YOLOv8 model using the `qat_config.yaml` file. The configuration provides extensive control over all aspects of the quantization process, allowing you to fine-tune for optimal accuracy and performance.

Table of Contents

1. Basic Configuration Structure

The `qat_config.yaml` file is organized into several main sections, each controlling different aspects of the quantization process:

```yaml
# Model settings
model:
  ...

# Quantization parameters
quantization:
  ...

# Calibration parameters
calibration:
  ...

# QAT parameters
qat_params:
  ...

# Knowledge distillation settings
distillation:
  ...

# Training parameters
train_params:
  ...

# Analysis and measurement settings
analysis:
  ...

# Export settings
export:
  ...
```

2. Model Settings

Configure the base model architecture and variant:

```yaml
model:
  architecture: "yolov8"  # Base architecture
  variant: "n"            # Model size (n, s, m, l, x)
  pretrained_path: "models/pretrained/yolov8n.pt"  # Path to pretrained model
```

Parameter	Description	Valid Options
`architecture`	Base architecture	"yolov8"
`variant`	Size of the model	"n" (nano), "s" (small), "m" (medium), "l" (large), "x" (xlarge)
`pretrained_path`	Path to pretrained model	Any valid file path

3. Quantization Parameters

3.1 Weight and Activation Settings

Configure how weights and activations are quantized:

```yaml
quantization:
  default_qconfig: "default"  # Default quantization configuration

  weight:
    dtype: "qint8"            # Data type for weights
    scheme: "per_channel"     # Quantization scheme
    observer: "minmax"        # Observer type for collecting statistics
    symmetric: true           # Whether to use symmetric quantization
    bit_width: 8              # Bit width for quantization
    channel_axis: 0           # Channel axis for per-channel quantization
    reduce_range: false       # Whether to reduce range for compatibility

  activation:
    dtype: "quint8"           # Data type for activations
    scheme: "per_tensor"      # Quantization scheme
    observer: "moving_average_minmax"  # Observer type
    symmetric: false          # Whether to use symmetric quantization
    bit_width: 8              # Bit width for quantization
    channel_axis: 0           # Channel axis (if using per-channel)
    reduce_range: false       # Whether to reduce range for compatibility
```

Weight Configuration Options:

Parameter	Description	Valid Options
`dtype`	Data type for quantized weights	"qint8" (signed), "quint8" (unsigned)
`scheme`	Quantization scheme	"per_channel" (recommended for weights), "per_tensor"
`observer`	Observer for collecting statistics	"minmax", "moving_average_minmax", "histogram", "custom_minmax", "per_channel_minmax"
`symmetric`	Symmetric quantization	true (recommended for weights), false
`bit_width`	Bit width for quantization	4, 8 (default), 16
`channel_axis`	Channel axis for per-channel	0 (default)
`reduce_range`	Reduce range for compatibility	true, false (default)

Activation Configuration Options:

Parameter	Description	Valid Options
`dtype`	Data type for quantized activations	"quint8" (unsigned, recommended), "qint8" (signed)
`scheme`	Quantization scheme	"per_tensor" (recommended for activations), "per_channel"
`observer`	Observer for collecting statistics	"moving_average_minmax" (recommended), "histogram", "custom_minmax"
`symmetric`	Symmetric quantization	false (recommended for activations), true
`bit_width`	Bit width for quantization	4, 8 (default), 16
`channel_axis`	Channel axis (if using per-channel)	0 (default)
`reduce_range`	Reduce range for compatibility	true, false (default)

3.2 Fake Quantization Settings

Configure the fake quantization modules that simulate quantization during training:

```yaml
quantization:
  ...
  fake_quantize:
    type: "custom"          # Type of fake quantization
    grad_factor: 1.0        # Factor for gradient scaling in STE
    momentum: 0.1           # Momentum for moving averages
    eps: 1e-5               # Small value for numerical stability
```

Parameter	Description	Valid Options
`type`	Type of fake quantization	"custom", "per_channel", "lsq" (Learned Step Size)
`grad_factor`	Gradient scaling factor	Float value (default: 1.0)
`momentum`	Momentum for moving averages	Float value (default: 0.1)
`eps`	Epsilon for numerical stability	Float value (default: 1e-5)

3.3 Layer-Specific Quantization

Configure different quantization settings for different layers:

```yaml
quantization:
  ...
  layer_configs:
    # First convolution layer (typically more sensitive)
    - pattern: "model.0.conv"
      config:
        activation:
          observer: "histogram"
          bit_width: 8
        weight:
          observer: "per_channel_minmax"
          bit_width: 8
      fake_quantize:
        type: "custom"
        grad_factor: 1.0

    # Detection head
    - pattern: "model.[0-9]+.detect"
      config:
        activation:
          observer: "histogram"
          bit_width: 8
        weight:
          observer: "per_channel_minmax"
          bit_width: 8
      fake_quantize:
        type: "lsq"  # Use LSQ for detection head
      calibration:
        method: "entropy"  # Use entropy-based calibration
        percentile: 99.99  # For percentile method if used
```

Each layer configuration can include:

Parameter	Description	Valid Options
`pattern`	Regex pattern to match layer names	Any valid regex pattern
`config`	Quantization configuration	Contains `activation` and `weight` sections
`skip`	Whether to skip quantization for this layer	true, false
`fake_quantize`	Fake quantization settings	Same as global fake_quantize
`calibration`	Layer-specific calibration	Contains calibration settings

4. Calibration Settings

Configure the calibration process that determines optimal quantization parameters:

```yaml
calibration:
  method: "histogram"       # Calibration method
  num_batches: 100          # Number of batches for calibration
  percentile: 99.99         # Percentile for percentile-based calibration
  use_coreset: true         # Whether to use coreset selection
  coreset_method: "k-center" # Coreset selection method
  coreset_size: 1000        # Coreset size
```

Parameter	Description	Valid Options
`method`	Calibration method	"histogram", "percentile", "entropy", "minmax"
`num_batches`	Number of batches	Positive integer
`percentile`	Percentile value	Float value (default: 99.99)
`use_coreset`	Whether to use coreset	true, false
`coreset_method`	Coreset selection method	"k-center", "random", "herding"
`coreset_size`	Number of samples in coreset	Positive integer

5. QAT Parameters

Configure the quantization-aware training process:

```yaml
qat_params:
  skip_detection_head: false  # Whether to skip detection head quantization
  fuse_modules: true          # Whether to fuse Conv+BN+ReLU modules
  fusion_patterns:            # Custom fusion patterns
    - {pattern: "model\\.\\d+\\.conv", modules: ["conv", "bn"], fuser_method: "fuse_conv_bn"}
    - {pattern: "model\\.\\d+\\.cv\\d+\\.conv", modules: ["conv", "bn", "silu"], fuser_method:
  use_lsq: true               # Whether to use LSQ
  use_mixed_precision: true   # Whether to apply mixed precision quantization
  use_distillation: true      # Whether to use knowledge distillation
  post_qat_fine_tuning: true  # Post-training fine-tuning
  fine_tuning_epochs: 1       # Fine-tuning epochs
  analyze_quantization_effects: true  # Whether to analyze quantization effects
  measure_layer_error: true   # Whether to measure layer-wise quantization error
  auto_adapt_bit_width: true  # Whether to automatically adapt bit widths based on sensitivity
```

Parameter	Description	Valid Options
`skip_detection_head`	Skip quantizing detection head	true, false
`fuse_modules`	Fuse Conv+BN+ReLU modules	true, false
`fusion_patterns`	Custom fusion patterns	List of pattern dictionaries
`use_lsq`	Use Learned Step Size Quantization	true, false
`use_mixed_precision`	Apply mixed precision quantization	true, false
`use_distillation`	Use knowledge distillation	true, false
`post_qat_fine_tuning`	Apply fine-tuning after QAT	true, false
`fine_tuning_epochs`	Number of fine-tuning epochs	Positive integer
`analyze_quantization_effects`	Analyze effects of quantization	true, false
`measure_layer_error`	Measure layer-wise error	true, false
`auto_adapt_bit_width`	Auto-adapt bit widths	true, false

6. Knowledge Distillation

Configure knowledge distillation during QAT:

```yaml
distillation:
  enabled: true                # Whether to use knowledge distillation
  teacher_model: "models/pretrained/yolov8n.pt"  # Teacher model path
  temperature: 4.0             # Temperature for softening logits
  alpha: 0.5                   # Weight for distillation loss
  feature_distillation: true   # Whether to use feature distillation
  feature_layers: ["model.9", "model.14"]  # Layers for feature distillation
```

Parameter	Description	Valid Options
`enabled`	Enable knowledge distillation	true, false
`teacher_model`	Path to teacher model	Any valid file path
`temperature`	Temperature for softening	Float value (typically 1-10)
`alpha`	Weight for distillation loss	Float value between 0-1
`feature_distillation`	Use feature distillation	true, false
`feature_layers`	Layers for feature distillation	List of layer names

7. Training Parameters

Configure the training process:

```yaml
train_params:
  epochs: 2                  # Number of epochs
  batch_size: 16             # Batch size
  lr: 0.0005                 # Learning rate
  lr_scheduler: "cosine"     # Learning rate scheduler
  warmup_epochs: 1           # Warm-up epochs
  optimizer: "SGD"           # Optimizer
  momentum: 0.937            # Momentum
  weight_decay: 0.0005       # Weight decay
  patience: 100              # Patience for early stopping
  loss_weights:              # Loss weights
    box: 0.05
    cls: 0.5
    obj: 1.0
```

Parameter	Description	Valid Options
`epochs`	Number of epochs	Positive integer
`batch_size`	Batch size	Positive integer
`lr`	Learning rate	Float value (typically 0.0001-0.001)
`lr_scheduler`	Learning rate scheduler	"cosine", "step", "linear"
`warmup_epochs`	Warmup epochs	Integer (typically 0-3)
`optimizer`	Optimizer type	"SGD", "Adam", "AdamW"
`momentum`	Momentum for SGD	Float value (typically 0.9-0.99)
`weight_decay`	Weight decay	Float value (typically 0.0001-0.001)
`patience`	Early stopping patience	Positive integer
`loss_weights`	Weights for loss components	Dictionary of float values

8. Analysis and Measurement

Configure analysis of quantization effects:

```yaml
analysis:
  enabled: true              # Whether to analyze quantization effects
  measure_layer_error: true  # Whether to measure layer-wise quantization error
  test_samples: 100          # Number of test samples for error analysis
  generate_visualization: true # Whether to generate visualization
  metrics: ["abs_error", "rel_error", "mAP"]  # Metrics to track
  compare_with_fp32: true    # Whether to compare with full-precision model
```

Parameter	Description	Valid Options
`enabled`	Enable analysis	true, false
`measure_layer_error`	Measure layer-wise error	true, false
`test_samples`	Number of test samples	Positive integer
`generate_visualization`	Generate visualizations	true, false
`metrics`	Metrics to track	List of metric names
`compare_with_fp32`	Compare with FP32 model	true, false

9. Export Settings

Configure model export options:

```yaml
export:
  formats: ["onnx", "tensorrt"]  # Export formats
  include_metadata: true         # Whether to include metadata
  simplify: true                 # Whether to simplify exported models
  validate: true                 # Whether to validate exported models
  include_quant_params: true     # Whether to include quantization parameters
```

Parameter	Description	Valid Options
`formats`	Export formats	List from: "onnx", "tensorrt", "tflite", "openvino"
`include_metadata`	Include metadata	true, false
`simplify`	Simplify exported models	true, false
`validate`	Validate exports	true, false
`include_quant_params`	Include quant parameters	true, false

10. Advanced Configuration Examples

Example 1: Mixed Precision Configuration

This example configures mixed precision quantization with 4-bit weights for bottleneck layers and 8-bit for more sensitive layers:

```yaml
quantization:
  default_qconfig: "default"

  # Default configuration (8-bit)
  weight:
    dtype: "qint8"
    scheme: "per_channel"
    observer: "minmax"
    symmetric: true
    bit_width: 8

  # Layer-specific configurations
  layer_configs:
    # First convolution layer (8-bit)
    - pattern: "model.0.conv"
      config:
        activation:
          observer: "histogram"
          bit_width: 8
        weight:
          observer: "per_channel_minmax"
          bit_width: 8

    # Bottleneck layers (4-bit)
    - pattern: "model.[0-9]+.bottleneck"
      config:
        activation:
          observer: "moving_average_minmax"
          bit_width: 8  # Keep activations at 8-bit
        weight:
          observer: "minmax"
          bit_width: 4  # Use 4-bit for weights
        fake_quantize:
          type: "lsq"  # Use LSQ for low-bit quantization

qat_params:
  use_mixed_precision: true
  auto_adapt_bit_width: false  # Manually specified bit widths
```

Example 2: Distillation with Entropy Calibration

This example uses knowledge distillation with entropy-based calibration:

yaml

```yaml
calibration:
  method: "entropy"
  num_batches: 100
  use_coreset: true
  coreset_method: "k-center"
  coreset_size: 1000

distillation:
  enabled: true
  teacher_model: "models/pretrained/yolov8n.pt"
  temperature: 4.0
  alpha: 0.6
  feature_distillation: true
  feature_layers: ["model.9", "model.14"]

qat_params:
  use_distillation: true
  fuse_modules: true
  use_lsq: true
```

Example 3: Full Production Configuration

A comprehensive configuration for production deployment:

yaml

```yaml
model:
  architecture: "yolov8"
  variant: "s"  # Small variant

quantization:
  weight:
    dtype: "qint8"
    scheme: "per_channel"
    observer: "minmax"
    symmetric: true
    bit_width: 8

  activation:
    dtype: "quint8"
    scheme: "per_tensor"
    observer: "moving_average_minmax"
    symmetric: false
    bit_width: 8

  # First and last layers get special treatment
  layer_configs:
    - pattern: "model.0.conv"  # First layer
      config:
        activation:
          observer: "histogram"
          bit_width: 8
        weight:
          observer: "per_channel_minmax"
          bit_width: 8

    - pattern: "model.[0-9]+.detect"  # Detection head
      config:
        activation:
          observer: "histogram"
          bit_width: 8
        weight:
          observer: "per_channel_minmax"
          bit_width: 8
      fake_quantize:
        type: "lsq"

calibration:
  method: "histogram"
  num_batches: 200  # More batches for better calibration
  use_coreset: true
  coreset_method: "k-center"
```

```yaml
  coreset_size: 2000

qat_params:
  fuse_modules: true
  use_lsq: true
  use_mixed_precision: false  # Uniform 8-bit for production stability
  use_distillation: true
  analyze_quantization_effects: true

distillation:
  enabled: true
  teacher_model: "models/pretrained/yolov8s.pt"
  temperature: 4.0
  alpha: 0.5

train_params:
  epochs: 5
  batch_size: 32
  lr: 0.0003
  lr_scheduler: "cosine"
  warmup_epochs: 1
  optimizer: "AdamW"
  weight_decay: 0.0005

export:
  formats: ["onnx", "tensorrt", "openvino"]
  include_metadata: true
  simplify: true
  validate: true
```

This comprehensive configuration guide should help you understand and effectively utilize all the features available in the enhanced quantization-aware training system. Experiment with different settings to find the optimal balance between model size, performance, and accuracy for your specific application.