TinyML — Attempt 1

MCQ

Q1.

In Post-Training Quantization (PTQ) for a TinyML model, a neural network with 32-bit floating-point weights is quantized to 8-bit integers. Which of the following is a key challenge that may significantly impact the model's performance on an edge device?

A) PTQ requires extensive retraining to map weights to discrete quantization levels, increasing deployment time.
B) PTQ may introduce significant quantization errors in layers with high weight variance, leading to accuracy degradation.
C) PTQ increases the model's memory footprint due to the need for additional lookup tables for quantized weights.
D) PTQ is incompatible with hardware accelerators that support integer operations, limiting its use in TinyML.

Answer: B

Q2.

When combining structured pruning and Post-Training Quantization (PTQ) for a TinyML model with 4 million parameters, 50% of which are pruned and then quantized from 32-bit floats to 4-bit integers, which of the following best describes the combined effect on model deployment?

A) The combined approach reduces the model size by a factor of 16 and ensures compatibility with all TinyML hardware, but it eliminates the need for calibration data.
B) The combined approach reduces the model size by a factor of 16 and lowers computational requirements, but it may require fine-tuning to mitigate accuracy loss.
C) The combined approach increases inference latency due to the complexity of handling 4-bit integers, making it unsuitable for microcontrollers.

D) The combined approach requires retraining the entire model from scratch, negating the benefits of PTQ's simplicity.

Answer: B

Q3.

What is a key advantage of Quantization-Aware Training (QAT) over Post-Training Quantization (PTQ) for a TinyML model deployed on a microcontroller?

A) QAT requires no calibration dataset, unlike PTQ.
B) QAT increases model size to improve inference speed.
C) QAT avoids integer operations, enhancing hardware flexibility.
D) QAT minimizes accuracy loss by simulating quantization during training.

Answer: D

Q4.

Edge Impulse's end-to-end pipeline explicitly includes:

A) Data collection → Feature extraction → Model training → Deployment
B) Data collection → Hyperparameter search → MLOps CI/CD → Deployment
C) Data collection → Auto-annotation only → Deployment
D) Data labelling → Explainable AI dashboard → Deployment

Answer: A

Q5.

In Edge Impulse, the **"Data Acquisition"** block is responsible for:

A) Model training
B) Collecting and labeling sensor data
C) Generating C++ inference code
D) Deploying to the cloud

Answer: B

MSQ

Q1.

Post-training quantization (PTQ):

A) Converts weights from higher bit precision to lower precision.
B) Requires no retraining.
C) Requires post-quantization training.
D) Needs no calibration dataset.

Answer: A, B

Q2.

Which techniques improve inference speed on edge devices?

A) Quantization-aware training
B) Knowledge distillation
C) Operator fusion
D) Batch Normalization folding

Answer: A, C, D

Q3.

Choose all challenges typical in deploying compressed models to embedded devices:

A) Limited compute resources
B) Low memory availability
C) Lack of floating-point units
D) Abundant GPU memory

Answer: A, B, C

Q4.

Which compression techniques directly reduce FLOPs (floating-point operations)?

A) Structured pruning
B) Weight quantization
C) Low-rank factorization
D) Knowledge distillation

Answer: A, C, D

Q5.

Based on the case study in building lightweight architectures, which of the following are essential for deploying a model on **Arduino Nano 33 BLE Sense**?

A) Use of pre-trained MobileNet or lightweight variants
B) Reducing number of pooling layers
C) Applying pruning or quantization
D) Using floating-point weights of size 64 bits

Answer: A, C

NAT

Q1.

An edge device can handle 25 million FLOPs per second. A CNN model requires 100 million FLOPs per inference. Calculate the latency (in milliseconds) to run a single inference on this device.

Answer: 4000 ms

Solution:

$$\text{Time (seconds)} = \frac{100\text{M FLOPs}}{25\text{M FLOPs/s}} = 4 \text{ seconds}$$

$$\text{Latency} = 4 \times 1000 = 4000 \text{ ms}$$

Q2.

A neural network model initially has 2.5 million parameters, each stored as a 32-bit float (4 bytes). After applying quantization, the parameters are reduced to 8-bit integers (1 byte). Additionally, pruning removes 30% of the parameters before quantization.

Calculate the compression ratio, defined as the ratio of the original model size to the compressed model size (in bytes). Round your final answer to two decimal places.

Answer: $\text{CR} = 5.71$

Q3.

Original model has 2,000,000 parameters (32-bit). You prune 25%, quantize the remaining to 16-bit (2 bytes), then apply Huffman coding that gives an additional 20% reduction in the already-quantized size. Compute the compression ratio (CR).

Answer: $\text{CR} = 3.33$

Solution:

Original size:
$2{,}000{,}000 \times 4 = 8{,}000{,}000 \text{ bytes}$

After 25% pruning (75% remaining):
$2{,}000{,}000 \times 0.75 = 1{,}500{,}000$ parameters

After quantization to 16-bit (2 bytes):
$1{,}500{,}000 \times 2 = 3{,}000{,}000 \text{ bytes}$

After Huffman coding (20% reduction):
$3{,}000{,}000 \times 0.80 = 2{,}400{,}000 \text{ bytes}$

Compression ratio:
$$\text{CR} = \frac{8{,}000{,}000}{2{,}400{,}000} = 3.3$$

Q4.

Given:

- Output of Flatten layer is 25,088 neurons
- First Dense layer → 128 units
- Second Dense layer → 10 output classes

How many trainable parameters are in the **first Dense layer** (Flatten → 128)?

Answer: 3,211,392

Solution:

Trainable parameters in first Dense layer:

$$25{,}088 \times 128 + 128 = 3{,}211{,}392$$

Q5.

In the transfer learning workflow:

You froze the CNN base and added a head:

- Dense(64) → Dense(10)

If the input to Dense(64) has 128 features, calculate the total number of trainable parameters.

Answer: 8,906

Solution:

Parameters in Dense(64):
$$128 \times 64 + 64 = 8{,}256$$

Parameters in Dense(10):

$64 \times 10 + 10 = 650$

Total trainable parameters:

$8{,}256 + 650 = 8{,}906$