Optimizing Neural Networks for Embedded Systems

https://github.com/as-budi/Embedded_AI.git

Pruning: Menghapus Bagian Model yang Tidak Penting

- Pruning adalah teknik kompresi model dengan **menghilangkan bobot (weight) atau neuron** yang dianggap tidak terlalu berkontribusi pada hasil prediksi model.

Jenis-Jenis Pruning

1. **Weight Pruning (Unstructured)**
   Menghapus bobot individual yang nilainya mendekati nol.

2. **Neuron Pruning (Structured)**
   Menghapus seluruh neuron, filter, atau channel dari layer tertentu (lebih cocok untuk deployment karena lebih mudah dioptimalkan oleh compiler).

3. **Layer Pruning**
   Menghapus layer secara utuh—jarang dilakukan kecuali untuk arsitektur besar seperti ResNet.

Proses Umum

1. Latih model seperti biasa.
2. Evaluasi kontribusi setiap weight atau neuron (misalnya via magnitude, sensitivitas, atau importance score).
3. Hapus weight atau neuron yang kontribusinya rendah.
4. **Fine-tune** model hasil pruning agar akurasinya tetap optimal.

Contoh

- Jika sebuah layer memiliki 512 neuron, dan 40% di antaranya memberikan kontribusi kecil terhadap output, maka 200 neuron bisa dihapus.

Kelebihan

- Mengurangi jumlah parameter → **hemat memori dan kecepatan inferensi meningkat**.
- Bisa menjaga akurasi jika dilakukan dengan hati-hati.

Quantization: Mereduksi Presisi Data

- Quantisasi mengubah representasi **floating-point (FP32)** menjadi representasi dengan **presisi lebih rendah**, seperti **INT8, INT4**, atau bahkan **binary (1-bit)**.

Jenis-Jenis Quantisasi

1. **Post-Training Quantization (PTQ)**
   Dilakukan setelah model selesai dilatih.
   - Contoh: convert bobot dari float32 ke int8.
2. **Quantization-Aware Training (QAT)**
   Simulasi quantisasi saat training agar model terbiasa dengan penurunan presisi.

3. **Dynamic Quantization**
   Hanya bobot saja yang diquantisasi, sedangkan aktivasi tetap float (berguna untuk NLP dengan RNN/LSTM).

4. **Static Quantization**
   Bobot dan aktivasi diquantisasi; perlu data kalibrasi untuk menetapkan range.

Proses Umum PTQ

1. Latih model dalam FP32.
2. Tentukan range min–max bobot dan aktivasi (mis. [-1, 1]).
3. Ubah representasi:
   Misal:
   $w_{int8} = \text{round}\left(\frac{w_{fp32}}{scale}\right)$
   dengan `scale = (max - min)/255` untuk INT8.
4. Gunakan representasi integer saat inferensi.

Kelebihan

- Ukuran model dapat berkurang **hingga 4×** (FP32 → INT8).
- Inferensi lebih cepat dan hemat daya karena integer arithmetic jauh lebih ringan dari floating point.

Kombinasi Pruning + Quantization

- Biasanya digunakan bersamaan untuk mencapai efisiensi maksimal.
- Contoh: model MobileNet-V1 dapat diperkecil dari 16MB → <4MB dengan kombinasi teknik ini.

Contoh Tools yang Mendukung

- **TensorFlow Lite Converter**
- **PyTorch Quantization API**
- **ONNX Runtime**
- **TVM (Tensor Virtual Machine)**
- **STM32Cube.AI** untuk perangkat embedded STM32

Contoh Kasus di Embedded Device

Misalnya kita deploy CNN untuk klasifikasi suara pada **ESP32 (RAM < 520KB)**:

- **Pruning** digunakan untuk memangkas layer CNN hingga separuh neuron dihilangkan.
- **Quantisasi** digunakan untuk mengubah model FP32 ke INT8, hingga ukuran model turun dari 1.2MB → 300KB dan bisa disimpan di flash ESP32.

Contoh Implementasi di Python

- Berikut contoh **kode Python** menggunakan **TensorFlow** untuk **pruning dan quantization** model sederhana (misalnya `Dense Neural Network`) menggunakan dataset MNIST:

1. Install dependensi

```
pip install tensorflow tensorflow-model-optimization
```

2. Latih model dasar (baseline model)

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

# Load data
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
y_train, y_test = to_categorical(y_train), to_categorical(y_test)

# Baseline model
model = Sequential([
    Flatten(input_shape=(28, 28)),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=2, validation_split=0.1)
```

3. Pruning model menggunakan TensorFlow Model Optimization Toolkit

```
import tensorflow_model_optimization as tfmot

# Define pruning schedule
pruning_params = {
    'pruning_schedule': tfmot.sparsity.keras.PolynomialDecay(
        initial_sparsity=0.0,
        final_sparsity=0.5,
        begin_step=0,
        end_step=len(x_train) // 128 * 2  # 2 epochs
    )
}

# Apply pruning
pruned_model = tfmot.sparsity.keras.prune_low_magnitude(model, **pruning_params)

# Re-compile and train
pruned_model.compile(optimizer='adam',
                     loss='categorical_crossentropy',
                     metrics=['accuracy'])

pruned_model.fit(x_train, y_train,
                 epochs=2,
                 batch_size=128,
                 validation_split=0.1,
```

4. Post-Training Quantization (PTQ)

```
converter = tf.lite.TFLiteConverter.from_keras_model(pruned_model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]  # Aktivasi quantisasi
tflite_quant_model = converter.convert()

# Simpan model hasil quantization
with open('model_pruned_quantized.tflite', 'wb') as f:
    f.write(tflite_quant_model)
```

5. Cek Ukuran Model Sebelum dan Sesudah

```
import os

# Save original model
model.save('original_model.h5')

# Ukuran file
print("Original Model Size (MB):", os.path.getsize("original_model.h5") / 1e6)
print("Pruned + Quantized Model Size (MB):", os.path.getsize("model_pruned_quantized.tflite") / 1e6)
```

Ringkasan

- **Pruning** dilakukan selama fine-tuning dengan polynomial decay untuk sparsity.
- **Quantization** dilakukan setelah training (PTQ) untuk konversi ke INT8.
- Model dapat di-deploy ke **microcontroller** (via TensorFlow Lite for Microcontrollers) jika ukurannya < 1MB.