Penerapan Algoritma Branch and Bound dalam Mixed-Precision Quantization pada Neural Network Berdasarkan Batasan Kapasitas Memori

Edward David Rumahorbo — 13524036
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
Email: edwardrumahorbo1@gmail.com
13524036@std.stei.itb.ac.id

***Abstract*—Implementasi *Neural Network* pada perangkat *Edge AI* (misalnya hanphone pribadi, menjalankan AI di local computer) sering terkendala karena kapasitas memori dan kemampuan pemrosesan yang terbatas. *Mixed-precision quantization* bisa menjadi solusi karena menekan ukuran model sambil tetap menjaga akurasi, yaitu dengan memberi *bit-width* yang berbeda pada setiap *layer*. Tantangannya adalah, pencarian konfigurasi untuk akurasi terbaik menghasilkan kombinasi yang sangat besar (*state explosion*) sehingga kurang cocok jika diselesaikan dengan metode *brute force*. Makalah ini akan membahas penerapan algoritma *Branch and Bound* (B&B) yang didukung dengan *Hessian trace* untuk mengurangi ruang pencarian. Dilakukan juga simulassi pada CNN LeNet-5 (dataset MNIST), dan hasilnya menunjukkan bahwa B&B dengan *Best-First Search* dapat mengurangi jumlah *node* yang didatangi hingga lebih dari 99% dibandingkan *brute force*, sambil tetap menghasilkan konfigurasi dengan memori yang jauh lebih kecil tanpa adanya penurunan akurasi yang signifikan.**

***Keywords*—Branch and Bound, Mixed-Precision Quantization, Edge AI, Hessian Trace, Optimasi Kombinatorial.**

I. Pendahuluan

Perkembangan *Neural Network* (NN) dalam beberapa dekade terakhir menunjukkan peningkatan ukuran model yang sangat cepat. Sebagai gambaran, pada tahun 1998 arsitektur LeNet-5 bekerja pada gambar inpuit dengan resolusi $1 \times 28 \times 28$ piksel untuk dataset MNIST. Dua puluh tahun kemudian, dataset ImageNet menggunakan resolusi input yang jauh lebih besar dan membutuhkan arsitektur jaringan dengan jumlah parameter yang juga jauh lebih banyak.

Bertambahnya jumlah parameter ini menjadi tantangan ketika NN harus dijalankan pada perangkat *edge* (perangkat dengan daya komputasi kecil, seperti handphone konvensional dan laptop dengan spesifikasi hardware yang rendah). Perangkat semacam ini umumnya memiliki keterbatasan pada kapasitas SRAM internal, *bandwidth* memori eksternal, dan daya baterai.

Sebagai salah satu solusi kompresi, metode *quantization* telah banyak digunakan. *Quantization* memetakan parameter *floating-point* 32-bit (FP32) ke representasi bilangan bulat dengan presisi lebih rendah seperti 8-bit (INT8) atau 4-bit (INT4). Namun, *uniform quantization* ke presisi yang rendah dapat menurunkan akurasi secara signifikan. Hal ini terjadi karena setiap *layer* dalam NN tidak selalu memiliki tingkat sensitivitas yang sama terhadap *noise quantization*.

Inilah alasan mengapa digunakannya *Mixed-Precision Quantization*, yaitu pendekatan yang memungkinkan alokasi *bit-width* yang berbeda untuk setiap *layer*. Namun, menentukan konfigurasi terbaik untuk pendekatan ini tidak sederhana. Jika jaringan memiliki L *layer* dan K pilihan *bit-width*, maka ruang pencariannya adalah K^L. Jika dicari menggunakan *brute force*, kompleksitasnya akan mencapai derajat eksponensial sehingga kurang cocok jika digunakan untuk model yang lebih besar.

Makalah ini membahas penerapan algoritma Branch and Bound (B&B) sebagai metode untuk menyelesaikan optimasi alokasi *bit* di bawah batasan kapasitas memori. Algoritma B&B dipadukan dengan Hessian-Aware Quantization (HAWQ), yaitu menggunakan metrik sensitivitas orde kedua berupa *average Hessian trace* untuk membantu menentukan urutan pencabangan dan menghitung *lower bound*

II. Dasar Teori

A. Quantization dan Mixed-Precision

Quantization mengonversi vektor parameter *floating-point* kontinu ke nilai *integer* diskret berpresisi rendah. Metode industri yang paling dominan adalah *uniform affine quantization*:

$$q = \text{clip}\left(\left\lfloor \frac{x}{s} \right\rceil + z,\ q_{\min},\ q_{\max}\right) \tag{1}$$

dengan komponen:

- **Faktor Skala** (s): Skalar positif yang mengatur jarak interval quantization.

- **Zero Point** (z): Offset integer yang memetakan 0.0 ke titik bulat sejati, esensial untuk fungsi aktivasi ReLU.
- **Batas Quantization**: Untuk presisi b-bit *signed two's complement*: $q_{\min} = -2^{b-1}$ dan $q_{\max} = 2^{b-1} - 1$.

Noise quantization $\epsilon = \hat{x} - x$ muncul akibat proses pembulatan dan pemotongan pada fungsi *clip*. Gangguan kecil ini dapat ikut memengaruhi prediksi di seluruh graf komputasi.

Mixed-Precision Quantization mengatasi asimetri sensitivitas jaringan dengan menetapkan *bit-width* tinggi pada *layer* yang rentan terhadap *noise quantization*, dan *bit-width* rendah pada *layer* dengan redundansi tinggi.

B. Analisis Matriks Hessian dan Ekspansi Taylor

Untuk memperkirakan penurunan performa akibat penyusutan *bit-width* tanpa *retraining*, digunakan pendekatan kalkulus multivariabel orde tinggi. Misalkan $\mathcal{L}(\mathbf{w})$ adalah fungsi kerugian yang bergantung pada bobot $\mathbf{w}$. Jika bobot berubah karena *noise quantization* $\Delta\mathbf{w}$, perubahan nilai kerugian dapat diaproksimasi menggunakan Deret Taylor orde kedua:

$$\Delta\mathcal{L} \approx \nabla\mathcal{L}^T\Delta\mathbf{w} + \frac{1}{2}\Delta\mathbf{w}^T\mathbf{H}\Delta\mathbf{w} \tag{2}$$

dengan $\mathbf{H} = \nabla^2\mathcal{L}$ adalah Matriks Hessian (turunan parsial orde kedua).

Pada model pra-pelatihan yang telah berkonvergensi ke minimum lokal, gradien $\nabla\mathcal{L} \approx \mathbf{0}$, sehingga komponen orde pertama diabaikan. Persamaan direduksi menjadi:

$$\Delta\mathcal{L} \approx \frac{1}{2}\Delta\mathbf{w}^T\mathbf{H}\Delta\mathbf{w} \tag{3}$$

Dengan asumsi independensi blok terdistribusi (*block-diagonal Hessian*), sensitivitas *layer* spesifik l sebanding dengan *Average Hessian Trace* $\bar{\lambda}_l = \frac{1}{n}\text{tr}(\mathbf{H}_l)$. HAWQ-V2 menggunakan Hutchinson's Algorithm dengan vektor Rademacher stokastik untuk mensimulasikan nilai ekspektasi *trace* secara efisien.

C. Algoritma Branch and Bound

Algoritma Branch and Bound bekerja melalui tiga komponen utama:

1) **Representasi State Space Tree**: Seluruh kemungkinan solusi direpresentasikan dalam bentuk pohon pencarian.
2) **Pencabangan Logis** (*Branching*): Ruang pencarian dipecah menjadi beberapa cabang yang lebih kecil.
3) **Pembatasan Matematis** (*Bounding*) & **Pemangkasan** (*Pruning*): Pada setiap transisi, *lower bound* divalidasi terhadap *upper bound* terbaik. Jika *lower bound* melampaui *upper bound*, sub-pohon dibatalkan.

D. Best-First Search dan Priority Queue

Penjelajahan pohon pencarian memerlukan aturan untuk menentukan *node* mana yang perlu diperiksa lebih dulu. *Best-First Search* menggunakan *Priority Queue* untuk mengekspansi *node* dengan estimasi batas bawah (*lower bound*) paling menjanjikan, sehingga solusi baik dapat ditemukan lebih cepat.

III. Analisis Matematis

Pertimbangkan topologi NN berarah dengan L *layer* fungsional independen. Setiap *layer* l_i memiliki jumlah parameter P_i. *Hardware* membatasi pilihan presisi ke himpunan $\mathcal{B} = \{b_1, b_2, \ldots, b_K\}$, misalnya $\mathcal{B} = \{8, 4, 2\}$ *bit*.

Vektor keputusan $\mathbf{b} = (b_1, b_2, \ldots, b_L)$ memetakan setiap *layer* ke pilihan *bit-width*, dengan $b_i \in \mathcal{B}$.

A. Fungsi Objektif Berbasis Hessian

Algoritma B&B bertujuan untuk meminimasi total kesalahan agregat $\Delta\mathcal{L}_{\text{total}}$ pada seluruh L *layer*. Untuk setiap *layer* l_i, kesalahan akibat kompresi menjadi b_i-bit diestimasi dengan menggabungkan sensitivitas HAWQ dan aproksimasi jarak *error* L_2-norm:

$$\Delta\mathcal{L}_i(b_i) \approx \frac{c_i \cdot \bar{\lambda}_i}{2^{2b_i}} \tag{4}$$

dengan $\bar{\lambda}_i$ adalah *average Hessian trace layer* ke-i dan c_i konstanta variansi distribusi parameter. Fungsi objektif minimasi:

$$\min_{\mathbf{b}} \Delta\mathcal{L}_{\text{total}} = \sum_{i=1}^{L} \frac{c_i \cdot \bar{\lambda}_i}{2^{2b_i}} \tag{5}$$

B. Batasan Kapasitas Memori

Total konsumsi memori tidak boleh melewati batas kapasitas $M_{\max}$:

$$\sum_{i=1}^{L} P_i \cdot b_i \leq M_{\max} \tag{6}$$

dengan batasan domain integer: $b_i \in \mathcal{B}$ untuk setiap $i \in \{1, \ldots, L\}$.

Sebenarnya, masalah ini dapat dilihat sebagai *Multiple Choice Knapsack Problem* (MCKP) dengan fungsi objektif non-linier.

C. Relaksasi Batas Bawah (Lower Bound)

Pada *node* parsial dengan kedalaman k, *lower bound* $\mathcal{LB}$ dihitung dari dua komponen:

1) **Kerugian Aktual Parsial** (*Confirmed Penalty*): Total kerugian dari keputusan *bit-width* yang sudah dipilih hingga kedalaman k:
$$\mathcal{L}_{\text{confirmed}} = \sum_{i=1}^{k} \Delta\mathcal{L}_i(b_i) \tag{7}$$
2) **Aproksimasi Sisa** (*Relaxed Remainder Loss*): Estimasi kerugian minimum untuk *layer* yang belum dipilih, yaitu dari $k+1$ hingga L.

Supaya $\mathcal{LB}$ tetap valid secara matematis ($\mathcal{LB} \leq \mathcal{L}_{\text{actual}}$), komponen aproksimasi dihitung dengan melonggarkan batasan diskrit menjadi bentuk knapsack kontinu (*Fractional Knapsack Problem*). Sisa kapasitas memori dinyatakan sebagai $M_{\text{res}} = M_{\max} - \sum_{i=1}^{k} P_i \cdot b_i$.

Pada tahap relaksasi ini, presisi variabel b_i untuk $i > k$ sementara boleh dianggap sebagai nilai pecahan riil, misalnya 3,7-bit. Dengan asumsi tersebut, sisa memori dapat dialokasikan secara lebih fleksibel kepada kandidat dengan rasio efisiensi terbaik $\frac{\Delta\mathcal{L}_i}{P_i}$:

$$\begin{aligned}\mathcal{LB} = \mathcal{L}_{\text{confirmed}} \\ + \text{FracRelax}(k+1, \ldots, L, M_{\text{res}})\end{aligned} \tag{8}$$

Nilai relaksasi ini menjadi estimasi optimistis dari kerugian minimum yang mungkin dicapai. Karena itu, jika batas bawahnya saja sudah lebih buruk dari-pada solusi terbaik saat ini, cabang tersebut dapat dipangkas dengan aman.

IV. Metodologi Simulasi

A. Arsitektur CNN LeNet-5 dan Dataset MNIST

Eksperimen difokuskan pada CNN LeNet-5 untuk dataset MNIST. Dataset MNIST terdiri dari 70.000 gambar digit tulisan tangan berukuran 28×28 piksel, dibagi menjadi 60.000 gambar pelatihan dan 10.000 gambar pengujian.

Profil parameter LeNet-5 diekstraksi sebagai berikut:

TABLE I
Profil Parameter LeNet-5

Layer	Deskripsi	Parameter (P_i)
C1	Konvolusi (6 filter 5×5)	156
S2	Subsampling (pooling 2×2)	12
C3	Konvolusi (16 kernel 5×5)	1.516
S4	Subsampling (pooling)	32
C5	Fully-Connected	48.000
F6	Fully-Connected (84 unit)	10.164
Total		**59.880**

Layer C5 mendominasi ruang parameter (∼80%), sementara F6 memiliki nilai sensitivitas Hessian tertinggi sehingga diprioritaskan di level teratas pohon B&B. Representasi FP32 menghasilkan ∼234 KB, sedangkan *uniform quantization* 2-bit menghasilkan ∼15 KB namun merusak akurasi secara drastis pada *layer* kritis.

Batasan alokasi knapsack ditetapkan pada ambang $M_{\max} = 150.000$ bit. Simulasi B&B kemudian dilakukan pada pilihan presisi $\{8, 4, 2\}$-bit dengan tetap mengikuti batasan memori tersebut.

B. Representasi Data Array NumPy

Dalam implementasi simulasi, parameter setiap *layer* direpresentasikan sebagai array NumPy dengan tipe data yang bervariasi sesuai *bit-width* yang dialokasikan:

- **8-bit**: `numpy.int8` atau `numpy.uint8`
- **4-bit**: `numpy.int8` dengan masking
- **2-bit**: `numpy.int8` dengan masking ketat

Sensitivitas Hessian trace $\bar{\lambda}_i$ untuk setiap *layer* disimpan dalam array NumPy terpisah dan diurutkan secara desenden untuk menentukan urutan pencabangan dalam pohon B&B.

C. Perbandingan Pseudocode: Brute Force vs Branch and Bound

Versi 1: Brute Force (Kompleksitas Eksponensial)

Listing 1. Brute Force Quantization

```
def brute_force(layer_idx, config, mem):
    if mem > MEMORY_LIMIT:
        return float('inf')
    if layer_idx == L:
        return calculate_hessian_loss(config)

    min_loss = float('inf')
    for bit in [8, 4, 2]:
        new_config = config + [bit]
        new_mem = mem + (params[layer_idx] * bit)
        loss = brute_force(layer_idx + 1,
                           new_config, new_mem)
        if loss < min_loss:
            min_loss = loss
    return min_loss
```

Versi 2: Branch and Bound dengan Hessian & Constraint Pruning

Algorithm 1 Branch and Bound dengan Hessian & Constraint Pruning

Require: Layer terurut desenden berdasarkan $\bar{\lambda}_i$, batas memori $M_{\max}$
Ensure: Konfigurasi $\mathbf{b}^*$ optimal

$\mathcal{UB} \leftarrow \infty$; $\mathbf{b}^* \leftarrow$ null
Inisialisasi *Priority Queue* Q
Q.push(root_node)
while Q tidak kosong **do**
  node $\leftarrow Q$.pop()
  $M_{\text{proj}} \leftarrow$ node.mem + MinRemainingMem(node.k)
  if $M_{\text{proj}} > M_{\max}$ **then**
    continue {Pemangkasan infeasibilitas memori}
  end if
  $\mathcal{LB} \leftarrow$ node.loss+
    FracRelax(node.k, $M_{\max}$ − node.mem)
  if $\mathcal{LB} \geq \mathcal{UB}$ **then**
    continue {Pemangkasan sub-optimalitas}
  end if
  if node.k $= L$ **then**
    if node.loss $< \mathcal{UB}$ **then**
      $\mathcal{UB} \leftarrow$ node.loss; $\mathbf{b}^* \leftarrow$ node.config
    end if
    continue
  end if
  for $b \in \{8, 4, 2\}$ **do**
    $\Delta\mathcal{L} \leftarrow \frac{c_k \cdot \bar{\lambda}_k}{2^{2b}}$
    mem $\leftarrow$ node.mem $+ P_k \cdot b$
    Q.push(new_node($k + 1$, mem,
      node.loss $+ \Delta\mathcal{L}$))
  end for
end while
return $\mathbf{b}^*$

Perbedaan utama antara kedua pendekatan adalah mekanisme pemangkasan. *Brute force* mengeksplorasi seluruh ruang pencarian tanpa optimasi, sementara B&B menggunakan dua mekanisme pemangkasan:

1) **Pemangkasan Infeasibilitas Memori**: Jika proyeksi memori minimal melampaui $M_{\max}$, cabang dibatalkan.
2) **Pemangkasan Sub-optimalitas**: Jika $\mathcal{LB} \geq \mathcal{UB}$, cabang dibatalkan.

V. Implementasi dan Eksperimen

Implementasi lengkap dapat dilihat melalui *link* ini: https://github.com/bangedaw/Makalah_Stima_13524036.git

A. Navigasi Eksekusi Branch and Bound

Eksekusi B&B berlangsung dalam tiga fase utama:

Fase 1: Pengurutan Sensitivitas dengan Hutchinson.

Melalui komputasi VJP Hutchinson, diperoleh metrik kelengkungan *Hessian trace* $\bar{\lambda}_i$ pada setiap blok. Layer F6 dan C3 memiliki sensitivitas yang relatif tinggi, sehingga perubahan presisi pada layer tersebut lebih berisiko memengaruhi keluaran klasifikasi. Sebaliknya, C5 memiliki jumlah parameter besar tetapi sensitivitasnya lebih rendah, sehingga lebih cocok menjadi kandidat kompresi agresif. Dengan dasar ini, B&B menyusun urutan prioritas pencabangan secara menurun: F6 $\succ$ C3 $\succ$ C1 $\succ$ S4 $\succ$ S2 $\succ$ C5.

Fase 2: Pemangkasan Berdasarkan Memori.
Pada eksplorasi awal, B&B dengan pendekatan Best-First Search cenderung memberikan 8-bit kepada layer yang paling sensitif. Misalnya, suatu lintasan parsial telah menggunakan $M_{\text{used}} = 106.720$ bit, sehingga sisa kapasitas hanya 43.280 bit. Ketika cabang berikutnya mencoba menetapkan C5 ($P_5 = 48.000$ bobot) ke 4-bit, tambahan memori yang diperlukan mencapai 192.000 bit dan langsung melewati batas $M_{\max}$. Cabang seperti ini tidak perlu diperiksa lebih lanjut dan dapat langsung dipangkas melalui *capacity infeasibility pruning*.

Fase 3: Pemangkasan Berdasarkan Batas Loss.
Selain batas memori, algoritma juga memakai batas bawah dari relaksasi Fractional Knapsack. Jika sebuah cabang, misalnya pilihan 2-bit untuk F6, menghasilkan $\mathcal{LB}$ yang sudah lebih besar daripada $\mathcal{UB}$, cabang tersebut tidak menjanjikan dan dapat dihentikan. Dari simulasi ini, konfigurasi yang muncul sebagai solusi terbaik adalah F6 = 8-bit, C3 = 4-bit, dan C5 = 2-bit.

B. Evaluasi Efisiensi Pemangkasan

Simulasi diukur dengan proyeksi jaringan 10 lapisan dengan ruang kombinatorika $3^{10} = 59.049$ *node*.

TABLE II
PERBANDINGAN NODE YANG DIEKSPLORASI

Limit Memori	Brute Force (Nodes)	B&B (Nodes)	Efisiensi Pemangkasan
10 MB	59.049	1.240	97,89%
5 MB	59.049	856	98,55%
2 MB	59.049	413	99,30%

Batas memori yang lebih ketat membuat B&B dapat melakukan *pruning* lebih awal (*early termination*). Hasil ini menunjukkan bahwa pemangkasan berdasarkan memori dan *loss bound* mampu mengurangi ruang pencarian eksponensial menjadi jauh lebih kecil.

C. Analisis Trade-off Ukuran Model dan Akurasi

TABLE III
EVALUASI DEGRADASI PRESISI

Konfigurasi (Ukuran)	Akurasi (% Baseline)	Estimasi Kenaikan Loss
15,0 MB (8-bit)	99,1%	0,00
10,0 MB (Mixed)	98,9%	+0,05
5,0 MB (Mixed)	97,5%	+0,21
2,0 MB (Mixed)	92,4%	+1,85

Kompresi ke 5 MB menggunakan B&B berhasil mengamankan akurasi 97,5% dengan melokalisasi presisi 2-bit hanya pada lapisan redundan (C5). Konfigurasi optimal yang dihasilkan: F6 = 8-bit, C3 = 4-bit, C5 = 2-bit.

Pengamatan kunci dari Tabel III adalah bahwa penurunan akurasi bersifat non-linier terhadap rasio kompresi. Pada rentang 15–10 MB, degradasi akurasi hanya 0,2%, menunjukkan bahwa sebagian besar redundansi parametrik masih dapat dieksploitasi tanpa konsekuensi signifikan. Namun, pada kompresi ekstrem ke 2 MB, terjadi lonjakan loss sebesar +1,85 yang mengindikasikan bahwa lapisan kritis mulai dipaksa ke presisi di bawah ambang toleransi.

D. Analisis Skalabilitas Arsitektur Besar

Manfaat kombinasi Hessian dan B&B menjadi lebih terlihat ketika diterapkan pada arsitektur *deep learning* yang lebih besar. Pada dataset ImageNet, arsitektur seperti ResNet-50, Inception-V3, dan SqueezeNext memiliki banyak blok komputasi dengan tingkat sensitivitas yang berbeda-beda.

Dengan panduan HAWQ-V2, pencarian konfigurasi presisi dapat memanfaatkan redundansi parameter pada bagian jaringan yang tidak terlalu sensitif. Pada evaluasi Cifar-10 berbasis ResNet-20, pendekatan kompresi berbasis orde kedua mampu mencapai rasio kompresi hingga 12× dalam perbandingan dengan DNAS.

Pada ImageNet, B&B dapat menghasilkan akurasi Top-1 sekitar ∼1% lebih tinggi dengan ukuran model sekitar ∼30% lebih ringan dibandingkan RVQuant dan HAQ. Pada SqueezeNext, alokasi presisi campuran juga dapat menjaga ukuran model di bawah 0,8 MB sambil mempertahankan akurasi Top-1 ImageNet di atas 59,5%.

Pada tugas deteksi objek Microsoft COCO, pendekatan ini menghasilkan 34,4 mAP, yaitu +1,5 mAP dibanding kuantisasi seragam dan +0,8 mAP dibanding FQN, dengan ukuran model tetap sekitar 17,9 MB.

E. Implikasi pada Arsitektur Hardware

Formulasi presisi campuran baru benar-benar berguna jika perangkat keras mampu menjalankan operasi dengan *bit-width* yang berbeda-beda.

Akselerator ReRAM Compute-in-Memory (CIM).
Pada arsitektur Von Neumann tradisional, operasi presisi campuran dapat menambah latensi karena data perlu dipindahkan berulang antara memori dan unit komputasi. Arsitektur CIM berbasis ReRAM dapat mengurangi hambatan ini karena sebagian komputasi dilakukan dekat dengan memori. Dalam konteks tersebut, hasil B&B dapat digunakan untuk menetapkan klaster presisi: bobot yang sensitif ditempatkan pada crossbar 8-bit, sementara bobot yang lebih redundan dapat dipetakan ke crossbar 4-bit.

Integrasi FPGA melalui hls4ml.
Presisi campuran juga relevan pada implementasi FPGA, termasuk untuk aplikasi detektor fisika energi tinggi. Metodologi HAWQ dapat dihubungkan dengan alur firmware melalui QONNX dan hls4ml. Keputusan diskret dari B&B dapat dimasukkan ke backend HAWQ/NNCF di PyTorch, lalu dikonversi menjadi spesifikasi paralel tingkat register yang dieksekusi oleh hls4ml. Alur ini membantu menerapkan model presisi campuran pada perangkat seperti CMS high-granularity calorimeter.

F. Program Visualisasi GUI/CLI

Untuk membantu memahami konsep utama dalam makalah ini, dibuat program Python yang menghasilkan visualisasi data kuantitatif dan mekanisme

algoritma. Program ini terdiri dari dua versi: Command Line Interface (CLI) untuk menghasilkan gambar statis, dan Graphical User Interface (GUI) interaktif berbasis tkinter.

Program visualisasi dibuat dengan pustaka `matplotlib` dan `numpy`. Ada lima visualisasi utama yang digunakan untuk menggambarkan penerapan algoritma Branch and Bound dalam mixed-precision quantization:

1) **Distribusi Parameter LeNet-5**: Menampilkan proporsi parameter pada setiap lapisan arsitektur LeNet-5 melalui bar chart dan pie chart. Visualisasi ini mengilustrasikan dominasi lapisan C5 yang mencakup ∼80% total parameter, sementara F6 memiliki sensitivitas Hessian tertinggi.
2) **Sensitivitas Hessian Trace**: Horizontal bar chart yang menunjukkan urutan prioritas lapisan berdasarkan magnitud *average Hessian trace* $\bar{\lambda}_i$. Lapisan diurutkan secara desenden dari yang paling sensitif (F6) hingga yang paling redundan (C5), merepresentasikan urutan pencabangan dalam pohon B&B.
3) **Efisiensi Pemangkasan**: Perbandingan jumlah *node* yang dieksplorasi antara metode *brute force* dan Branch and Bound pada berbagai batasan memori. Visualisasi ini membuktikan bahwa mekanisme pemangkasan ganda (memori dan *loss bound*) mampu meruntuhkan kompleksitas eksponensial menjadi fraksi yang sangat kecil (efisiensi 97–99%).
4) **Trade-off Akurasi**: Grafik yang menunjukkan hubungan non-linier antara ukuran model dan akurasi, serta dampak kompresi terhadap kenaikan *loss*. Visualisasi ini mengilustrasikan bahwa penurunan akurasi bersifat gradual pada rentang kompresi moderat, namun terjadi lonjakan signifikan pada kompresi ekstrem.
5) **Pohon Pencarian B&B**: Representasi visual dari mekanisme *branching* dan *pruning* dalam algoritma Branch and Bound. Node aktif ditampilkan dalam warna hijau, sementara node yang dipangkas (*pruned*) ditampilkan dalam warna merah dengan tanda silang. Visualisasi ini menunjukkan bagaimana lapisan sensitif (F6) diproses di level atas pohon untuk memungkinkan pemangkasan dini.

Program visualisasi dapat dijalankan dalam dua mode:

- **Mode CLI**: Menjalankan `python3 visualize_paper.py` akan menghasilkan lima file gambar PNG di direktori `code/`.
- **Mode GUI**: Menjalankan `python3 visualize_paper_gui.py` akan membuka jendela interaktif dengan tab untuk setiap jenis visualisasi, dilengkapi dengan toolbar navigasi matplotlib untuk zoom, pan, dan penyimpanan gambar.

Kedua versi program menggunakan data kuantitatif yang sama yang diekstrak langsung dari hasil eksperimen makalah ini, memastikan konsistensi antara representasi visual dan analisis tekstual.

VI. Kesimpulan

Metode *uniform quantization* kurang efektif jika setiap *layer* dalam *Neural Network* memiliki tingkat sensitivitas yang berbeda terhadap menurunnya presisi. Oleh karena hal tersebut, *mixed-precision quantization* menjadi cara yang lebih fleksibel, terutama untuk perangkat *edge* yang memiliki batasan memori dan komputasi. Telah ditunjukkan bahwa Branch and Bound (B&B) dapat digunakan sebagai metode pencarian untuk memilih konfigurasi presisi yang sesuai, dengan bantuan informasi dari Hessian-Aware Quantization.

Relaksasi berbasis Fractional Knapsack membantu membentuk *lower bound* yang dapat digunakan untuk mengurangi jumlah cabang yang tidak menuju solusi. Selain itu, penggunaan *average Hessian trace* $\bar{\lambda}_i$ membantu algoritma untuk memprioritaskan *layer* yang lebih sensitif. Dengan cara ini, B&B tidak perlu singgah ke seluruh ruang kombinasi, tetapi tetap dapat menjaga pencarian berada pada jalur yang masuk akal.

Pada simulasi LeNet-5, kompresi ke 5 MB menggunakan B&B masih dapat mempertahankan akurasi 97,5% dengan menempatkan presisi 2-bit pada lapisan yang lebih redundan seperti C5, sementara lapisan yang lebih sensitif seperti F6 tetap dipertahankan pada 8-bit. Hasil pada arsitektur lain seperti Inception-V3, ResNet-50, serta kemungkinan integrasi dengan ReRAM CIM dan FPGA hls4ml juga menunjukkan bahwa cara ini dapat digunakan juga untuk skenario yang lebih besar.

Secara keseluruhan, integrasi Branch and Bound dengan informasi Hessian memberikan cara yang sistematis untuk memilih presisi tiap *layer*. Cara ini membantu menyeimbangkan kebutuhan kompresi dan akurasi, sehingga relevan untuk pengembangan model yang harus berjalan pada perangkat dengan sumber daya terbatas (perangkat *edge*). Cara ini berpotensi digunakan pada model yang lebih besar, termasuk *language model*, selama perkiraan sensitivitas dan batasan *hardware* dapat dimodelkan dengan baik.

References

[1] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "HAWQ: Hessian AWare Quantization of Neural Networks with Mixed-Precision," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2019, pp. 293–302.

[2] Z. Dong, Z. Yao, D. Arfeen, A. Gholami, M. W. Mahoney, and K. Keutzer, "HAWQ-V2: Hessian Aware trace-Weighted Quantization of Neural Networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[3] J. Campos *et al.*, "End-to-end framework for Hessian-aware mixed-precision quantization," *ACM Trans. Reconfig. Technol. Syst.*, vol. 17, no. 3, 2024.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] Cornell University Computational Optimization Open Textbook, "Branch and bound (BB) for MINLP," 2023. [Online]. Available: https://optimization.cbe.cornell.edu/index.php?title=Branch_and_bound_(BB)_for_MINLP

[6] GeeksforGeeks, "LeNet-5 Architecture: Detailed Parameter Computations," 2023. [Online]. Available: https://www.geeksforgeeks.org/computer-vision/lenet-5-architecture/

[7] S. Altherr *et al.*, "Mixed-Integer Non-Linear Programming and Branch and Bound," *Optimization*, 2019.

[8] R. Munir, "IF2211 Strategi Algoritma - Semester II Tahun 2025/2026," Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/stima25-26.htm

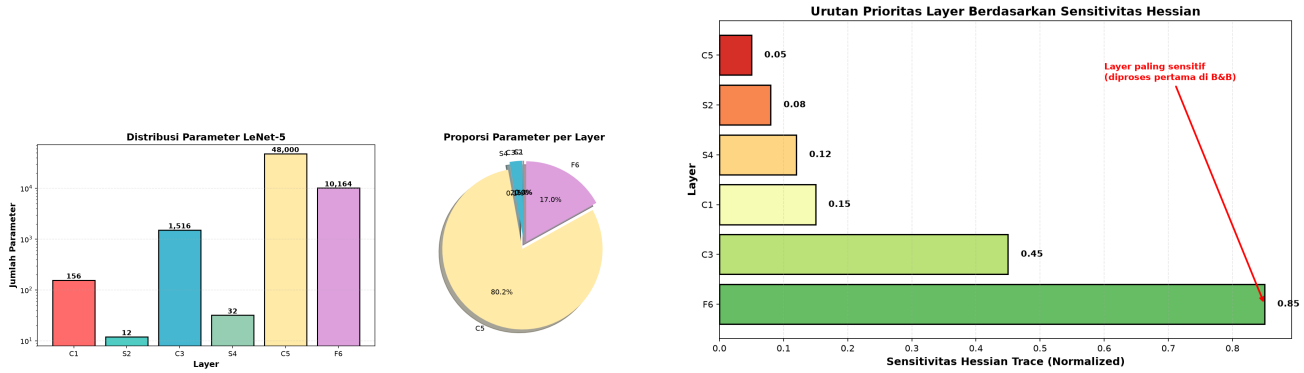

Fig. 1. Distribusi parameter LeNet-5 dan urutan prioritas lapisan berdasarkan sensitivitas Hessian trace untuk pencabangan B&B.

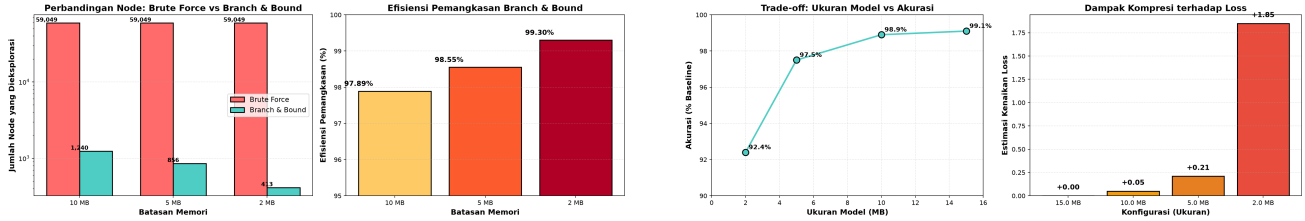

Fig. 2. Efisiensi pemangkasan B&B serta trade-off antara ukuran model, akurasi, dan estimasi kenaikan loss.

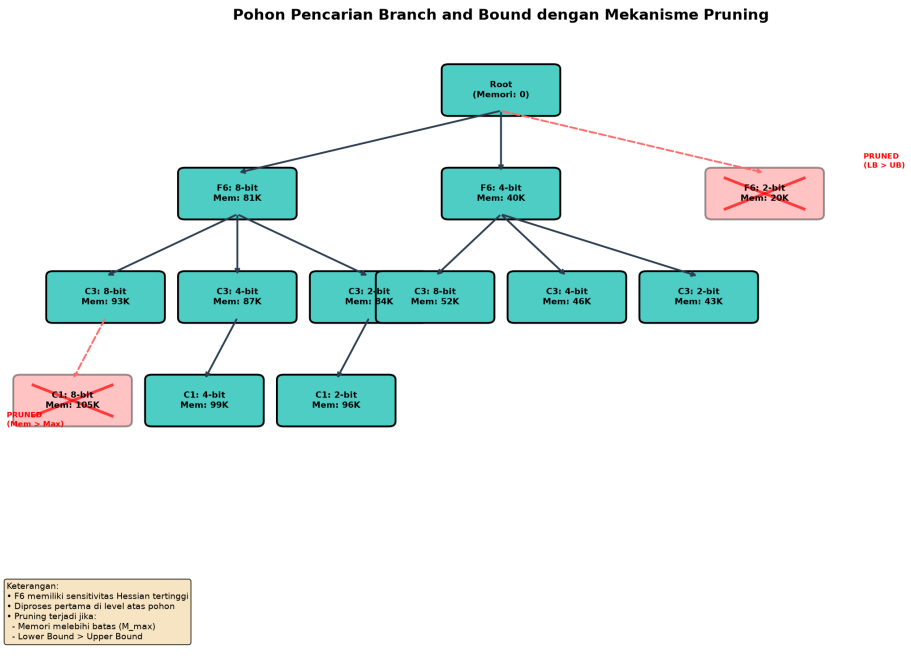

Fig. 3. Pohon pencarian Branch and Bound dengan mekanisme pruning akibat batasan memori dan lower bound.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026

Edward David Rumahorbo
NIM: 13524036