**ระบบเลือกการตั้งค่าเกมอัตโนมัติ**

**ด้วย Fuzzy Logic (Mamdani)**

**นายภูผา แก้วประดิษฐ์  
รหัสประจำตัว 660610842**

**รายงานประกอบการเรียนการสอน วิชา 261456 Introduction to Computational Intelligence for Computer Engineering**

**บทคัดย่อ**

รายงานนี้นำเสนอระบบช่วยเลือกการตั้งค่าเกมอัตโนมัติ โดยใช้ ตรรกะฟัซซีแบบ Mamdani จากคะแนนสมรรถนะ CPU, GPU, RAM (0–100) เพื่อแนะนำเอาต์พุตหรือการตั้งค่า 3 รายการ:

Graphics Quality (0–100), Resolution Scale (50–100%), และ Texture Quality (0–100) ระบบนิยามชุดสังกัด (membership functions) ด้วยรูปสามเหลี่ยม/สี่เหลี่ยมคางหมู สร้างกฎ 10 ข้อ ทำอนุมานแบบ AND=min, OR=max, implication=min, aggregation=max, defuzzification=centroid และเพิ่มการแมปตัวเลขไปเป็นการตั้งค่า Low/Medium/High/Ultra พร้อม ฮิสเทอรีซิส เพื่อลดการกระพริบของการตั้งค่าในการใช้งานจริง ผลการจำลองแสดงแนวโน้มสมเหตุสมผล: GPU มีอิทธิพลสูงต่อ Resolution, RAM ต่ำเป็นคอขวดสำหรับ Texture, และ CPU+GPU ร่วมกันดัน Quality

**บทที่ 1**

**วัตถุประสงค์ของการทดลอง**

1. พัฒนาแบบจำลองฟัซซี ที่รับอินพุต CPU/GPU/RAM (0–100) แล้วแนะนำ Graphics Quality, Resolution Scale, Texture Quality
2. ออกแบบนิยามเชิงภาษา **(linguistic variables)** และสมาชิกภาพ (MFs) ของอินพุต/เอาต์พุตให้เหมาะสม (LOW/MID/HIGH/ULTRA ด้วย **trimf/trapmf**) พร้อมเหตุผลรองรับ
3. สร้างฐานกฎ **(Rule Base)** ที่สะท้อนตรรกะการตั้งค่าเกมจริง เช่น GPU หนักกับ Resolution, RAM หนักกับ Texture, CPU+GPU ส่งผลร่วมต่อ Quality รวมถึงทดลอง “น้ำหนัก” ของกฎ
4. ยืนยันกระบวนการอนุมานแบบ Mamdani ครบขั้น (fuzzification → AND/OR → implication=min → aggregation=max → centroid) ว่าทำงานถูกต้อง
5. จำลองพฤติกรรมระบบในหลายสภาวะ (low/low/low), (high/high/high), เคสผสม

ตรวจว่าผลลัพธ์สอดคล้องกับความคาดหวังของกฎ

1. วิเคราะห์ความไว **(Sensitivity)** ของผลลัพธ์ต่ออินพุตแต่ละตัว โดยตรึงอีกสองตัวไว้ เพื่อดูทิศทาง/ความชันที่สมเหตุสมผล
2. สำรวจปฏิสัมพันธ์ระหว่างอินพุต (Interactions) ด้วย Heatmap/Surface (เช่น Quality บนระนาบ CPU×GPU) เพื่อดูแนวโน้ม, ความเป็นโมโนโทน, และจุดคอขวด
3. ประเมินความเสถียรของการตั้งค่าหรือ เอาต์พุต (Labels Stability)
4. ปรับจูนพารามิเตอร์ (ตำแหน่ง MF และน้ำหนักกฎ) ให้ผลลัพธ์ลื่นไหล ไม่กระโดด และสอดคล้องกับสมมติฐานโดเมน

**บทที่ 2**

**วิธีการและแบบจำลอง**

**2.1 ขอบเขตปัญหา**

**อินพุต**: คะแนนสมรรถนะ **CPU, GPU, RAM** ∈ [0,100]

**เอาต์พุต**:

* + **Graphics Quality** ∈ [0,100]
  + **Resolution Scale** ∈ [50,100] (%)
  + **Texture Quality** ∈ [0,100]

**2.2 กรอบอนุมานแบบ Mamdani**

(0) เตรียมโดเมน → (1) Fuzzification → (2) ประเมินเงื่อนไข (AND/OR)

→ (3) Implication (clip) → (4) Aggregation (รวมหลายกฎ) → (5) Defuzzification

**2.3 การกำหนดชุดสังกัด (Membership) แบบอธิบายเป็นคำพูด**

ในงานนี้เราใช้คำภาษาคนอย่าง LOW / MID / HIGH / ULTRA เพื่ออธิบายระดับของค่าต่าง ๆ โดยกำหนดเป็น “รูปฟังก์ชันสังกัด” 2 แบบ:

* **trapmf (Trapezoidal)** = รูป “สี่เหลี่ยมคางหมู” เป็น 4 จุดบนแกนคะแนน: เริ่มนับว่าใช่, ใช่เต็มที่, ยังใช่อยู่เต็มที่, ค่อย ๆเลิกนับว่าใช่
* **trimf (Triangular)** = รูป “สามเหลี่ยม” มี 3 จุด: เริ่มนับว่าใช่, ใช่ที่สุด (ยอดสามเหลี่ยม), เลิกนับว่าใช่

เหตุผลการเลือกใช้: เราให้ LOW/HIGH ใช้ **trapmf** เพื่อเปิดกว้างขึ้นในโซนที่ “ใช่แน่นอน” (มีช่วงราบ) และให้ MID ใช้ trimf เพราะต้องการช่วงกลางที่ พอดี แคบกว่าและชัดเจน

ต่อไปนี้คือ **ความหมายเชิงช่วง** (ตัวเลขในวงเล็บคือจุดคร่าว ๆ **ที่เริ่มนับว่าใช่/ใช่เต็มที่/เลิกนับว่าใช่**):

**Input : CPU, GPU, RAM (ช่วงคะแนน 0–100)**

**LOW = trapmf(0, 0, 25, 45)**

(คะแนนแถว ๆ 0–25 ถือว่า **ต่ำแน่** ตั้งแต่ 25–45 **ยังพอต่ำ** แต่ลดน้ำหนักลงเรื่อย ๆ เกิน 45 ไปถือว่า **ไม่ต่ำ**)

**MID = trimf(35, 55, 75)**

(คะแนนใกล้ 55 = **กลางพอดี** ช่วง 35–55 ค่อยๆ **ยังไม่ใช่กลาง** ไป **กลาง** ช่วง 55–75 ค่อยๆ **กลาง** กลับเป็น **ไม่กลาง**)

**HIGH = trapmf(65, 80, 100, 100)**

(ตั้งแต่ 80–100 = **สูงชัดเจน** ช่วง 65–80 ค่อยๆ นับว่า **สูงขึ้นเรื่อย** ต่ำกว่า 65 = **ไม่สูง**)

**Output ค่าที่ 1 : Graphics Quality (0–100)**

**LOW = trapmf(0, 0, 25, 40)** (0–25 **ต่ำชัดเจน** 25–40 **ยังต่ำแต่ลดลง** > 40 **ไม่ต่ำ)**

**MED = trimf(35, 55, 70)** (55 **กลางพอดี** 35–55 **ไต่ขึ้นเป็นกลาง** 55–70 **ไต่ลงจากกลาง**)

**HIGH = trimf(65, 80, 90)** (80 **สูงพอดี** 65–80 **ไต่ขึ้นเป็นสูง** 80–90 **ไต่ลงจากสูง**)

**ULTRA = trapmf(85, 92, 100, 100)** (92–100 **โคตรสูงชัดเจน** 85–92 **กำลังจะเป็น ultra**)

**Output ค่าที่ 2 : Resolution Scale (50–100%)**

**LOW = trapmf(50, 50, 60, 75)** (50–60 **ต่ำชัดเจน** 60–75 ค่อยๆ **เลิกต่ำ** >75 **ไม่ต่ำ**)

**MED = trimf(70, 80, 90)** (80 **กลางพอดี** 70–80 **ไต่ขึ้น** 80–90 **ไต่ลง**)

**HIGH = trapmf(88, 94, 100, 100)** (94–100 **สูงชัดเจน** 88–94 **กำลังจะสูง**)

**Output ค่าที่ 3 : Texture Quality (0–100)**

**LOW = trapmf(0, 0, 30, 50)** (0–30 **ต่ำชัดเจน** 30–50 ค่อย ๆ **เลิกต่ำ**)

**MED = trimf(45, 60, 75)** (60 **กลางพอดี** 45–60 **ไต่ขึ้น** 60–75 **ไต่ลง**)

**HIGH = trapmf(70, 85, 100, 100)** (85–100 **สูงชัดเจน** 70–85 **กำลังจะสูง**)

ถ้า GPU = 72: อยู่ในช่วง 65–80 ของ “HIGH” (กำลังไต่ขึ้นไปสูง) และยังอยู่ในช่วงปลายของ “MID” (กำลังไต่ลงจากกลาง) แปลว่า “GPU ตอนนี้เริ่มเป็น HIGH มากขึ้น แต่ยังมีความเป็น MID อยู่บ้าง” ระบบจะเอาระดับความเป็นสมาชิกทั้งสองนี้ไปใช้ในกฎต่อไป

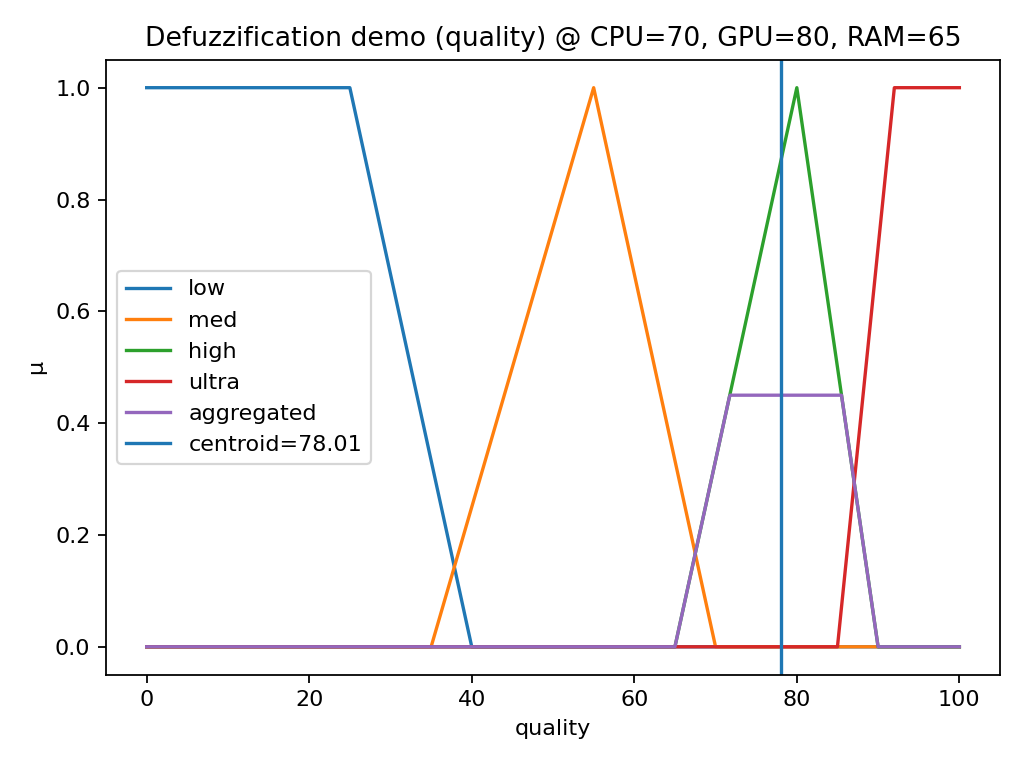
**2.4 ฐานกฎ (Rule Base)**

ใช้รูปแบบ AND=min, OR=max; บางผลมีน้ำหนัก เพื่อปรับความแรงของกฎ

|  |  |  |
| --- | --- | --- |
| **Rule** | **เงื่อนไข IF** | **ผลลัพธ์ (THEN)** |
| R1 | CPU=HIGH ∧ GPU=HIGH ∧ RAM=HIGH | Quality=ULTRA, Res=HIGH, Texture=HIGH |
| R2 | |  | | --- | |  |  |  | | --- | | GPU=HIGH ∧ CPU=MID ∧ RAM=MID | | Quality=HIGH, Res=HIGH, Texture=MED |
| R3 | CPU=MID ∧ GPU=MID ∧ RAM=MID | Quality=MED, Res=MED, Texture=MED |
| R4 | |  | | --- | |  |  |  | | --- | | GPU=LOW ∨ CPU=LOW | | Quality=LOW, Res=LOW |
| R5 | RAM=LOW | Texture=LOW |
| R6 | |  | | --- | |  |  |  | | --- | | GPU=HIGH ∧ RAM=MID | | Quality=HIGH(×0.9), Res=HIGH(×0.9), Texture=MED |
| R7 | CPU=MID ∧ GPU=HIGH ∧ RAM=HIGH | Quality=HIGH, Res=HIGH, Texture=HIGH |
| R8 | CPU=HIGH ∧ GPU=MID ∧ RAM=HIGH | Quality=HIGH(×0.8), Res=MED, Texture=HIGH |
| R9 | GPU=MID ∧ RAM=LOW | Texture=LOW, Res=LOW(×0.8) |
| R10 | CPU=LOW ∧ GPU=HIGH | Quality=MED, Res=MED |

**2.5 Defuzzification**

ใช้ **Centroid**:



**2.6 การแมปการตั้งค่าและฮิสเทอรีซิส**

* **Quality → Label:** Low (<37.5), Medium (<67.5), High (<87.5), Ultra (อื่น ๆ)
* **Res → Label:** Low (<72.5), Medium (<89.0), High (อื่น ๆ)
* **Texture → Label:** Low (<47.5), Medium (<72.5), High (อื่น ๆ)
* **Hysteresis Mapper** ลดการสลับฉลากไปมาเมื่อค่าก้ำกึ่ง โดยใช้เกณฑ์ขึ้น/ลงต่างกัน

(ทำการMapการตั้งค่า จากผลลัพท์ที่ได้จากการ Defuzzification และทำ Hysteresis เพื่อนำการตั้งค่าไปใช้งานจริง)

**บทที่ 3**

**การทดลอง**

**3.1 Environment & Reproducibility**

* **ไลบรารี**: Python, NumPy, Matplotlib, Pandas
* **การทำซ้ำ**: กำหนด random seed ภายใน random\_stats() เป็นค่าแน่นอน (np.random.default\_rng()) ทำให้ผลสุ่มชุดเดียวกันได้ซ้ำ
* **โดเมนตัวแปร (universes)**:
  + อินพุต/เอาต์พุตทั่วไป: u = linspace(0,100,1001)
  + Resolution Scale: uR = linspace(50,100,1001)

**3.2 อินพุตทดสอบและวิธีสร้างข้อมูล**

* **เคสเดี่ยว (single-case)**: ใช้ค่าคงที่เพื่ออธิบายการอนุมาน เช่น (CPU=70, GPU=80, RAM=65) สำหรับเดโม defuzzification
* **สแกนกริด (grid scan)**:
  + ใช้วิธีเติมตารางค่าบนกริด 2 มิติ (80×80 จุด) เพื่อสร้าง **Heatmap** ของเอาต์พุต โดยตรึงตัวแปรที่ 3 ไว้
  + ตัวอย่าง: Quality บนระนาบ CPU×GPU โดยตรึง RAM=70
* **สุ่มตัวอย่าง (random sampling)**:
  + สุ่มสม่ำเสมอภายในช่วงใช้งานจริง: CPU,GPU,RAM ~ U(5,95) จำนวน n=300
  + ใช้เพื่อวิเคราะห์การกระจายของเอาต์พุต (ฮิสโตแกรม) และสัดส่วนฉลาก (แท่ง)

**3.3 สถานการณ์การทดลอง (Experiment Scenarios)**

1. **ตรวจสอบ MF (Membership Visualization)**
   * วาดกราฟ MF ของทุกตัวแปร (CPU/GPU/RAM/Quality/Res/Texture)
   * จุดประสงค์: ยืนยันว่าความหมายของ **LOW/MID/HIGH/ULTRA** ถูกนิยามถูกต้อง ครอบคลุมโดเมน
2. **สาธิตกระบวนการอนุมาน (Defuzzification Demo)**
   * เลือกอินพุตหนึ่งชุด → คำนวณระดับการลั่นกฎ (antecedent) → clip ชุดสังกัดของเอาต์พุต → รวม (aggregate) → centroid
   * แสดงเส้น centroid บนกราฟ เพื่อให้เห็นผลลัพธ์เชิงตัวเลขมาจากพื้นที่ใต้กราฟอย่างไร
3. **Heatmaps (ปฏิสัมพันธ์ 2 ตัวแปร)**
   * **Quality: CPU×GPU | RAM=70** — ตรวจ “synergy” ระหว่าง CPU กับ GPU
   * **Resolution: GPU×RAM | CPU=70** — ตรวจชัดว่า Resolution ขึ้นกับ GPU; RAM มีผลรอง
   * **Texture: RAM×GPU | CPU=70** — ตรวจว่าคอขวด RAM ส่งผลกับ Texture มากกว่าปัจจัยอื่น
4. **Sensitivity (ความไวของเอาต์พุต)**
   * ตรึง 2 อินพุตไว้ (เช่น CPU=70, RAM=70) แล้วไล่ค่าที่เหลือ (GPU: 0→100)
   * วาดเส้นผลลัพธ์ quality/res/tex ตามค่าที่ไล่ เพื่อดูแนวโน้ม/ความชัน และความต่อเนื่อง
5. **Random Simulation (การกระจายโดยรวม)**
   * สุ่ม 300 เคส บันทึกผลลัพธ์เป็น .csv + วาดฮิสโตแกรมค่า Quality + แท่งสัดส่วนฉลาก
   * ใช้ดูว่าโดยรวมระบบมีแนวโน้มให้คำแนะนำระดับใดบ่อย และมี outlier หรือไม่

**3.4 เกณฑ์ตรวจสอบ/ตัวชี้วัด (Checks & Metrics)**

* **ความต่อเนื่อง (Continuity)**: กราฟไม่มีจุดหักจากการเปลี่ยนอินพุตเล็กน้อย
* **โมโนโทน (Monotonicity) เฉพาะโดเมน**:
  + ตรึง CPU/RAM → เพิ่ม GPU แล้ว **Resolution** ไม่ควรลดลง
  + RAM ต่ำ → **Texture** ควรมีแนวโน้มต่ำ
* **สอดคล้องกฎ (Rule Consistency)**:
  + (HIGH,HIGH,HIGH) → Quality ~ สูงมาก/ultra
  + (LOW,*,*) หรือ (*,LOW,*) → Quality/Res ต่ำ
* **ความสมเหตุสมผลของสัดส่วนฉลาก**: จาก 300 เคส สัดส่วนควรสะท้อนฐานกฎ ไม่สุดโต่งจนผิดสังเกต

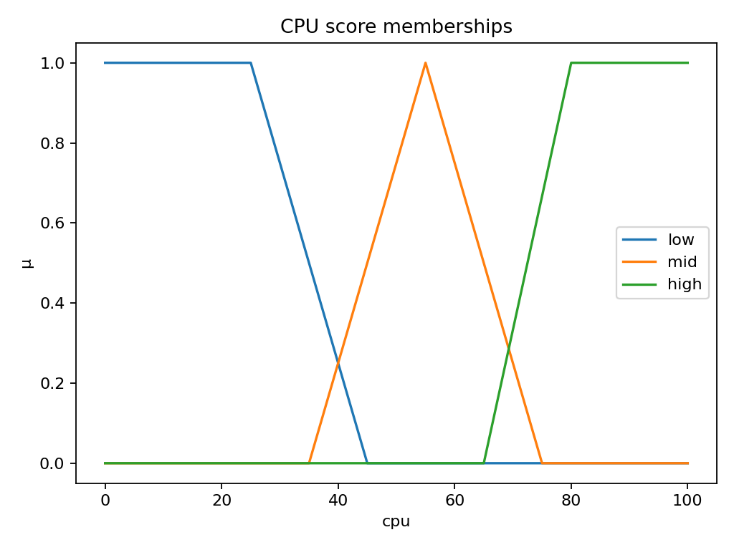
**3.5 สิ่งที่คาดหวังจากผลลัพธ์ (Expected Behaviors)**

* **Quality** สูงเมื่อ CPU+GPU สูงพร้อมกัน (ผลรวม synergistic)
* **Resolution** แปรตาม GPU เด่นชัด; RAM มีผลรอง (เมื่อ RAM ต่ำมาก Res อาจไม่ถึง High แม้ GPU กลาง)
* **Texture** ติดเพดานเมื่อ RAM ต่ำ (ลด stutter/โหลด)

**บทที่ 4**

**ผลการทดลอง**

**4.1 กราฟ Membership Functions (MF)**

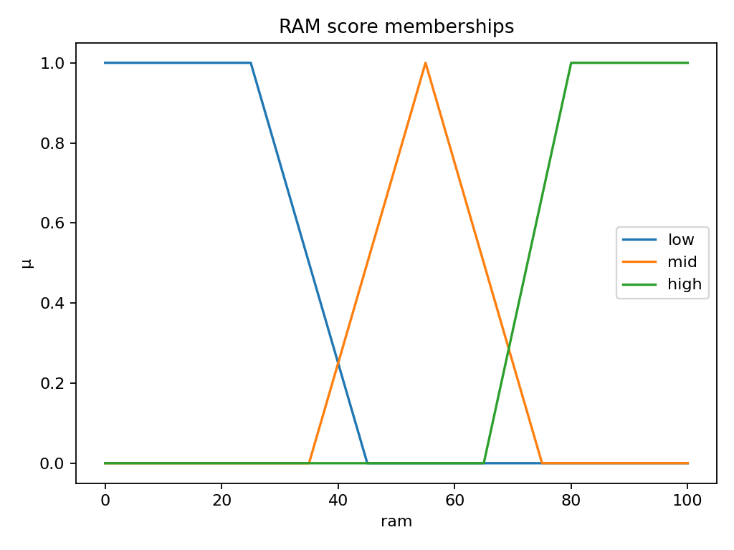
แสดงนิยามเชิงภาษา LOW / MID / HIGH ของ CPU บนโดเมน 0–100 โดยใช้ trapmf สำหรับ LOW/HIGH และ trimf สำหรับ MID

**รูปที่ 4.1**: *CPU score memberships*

*A graph of a group of members

AI-generated content may be incorrect.*

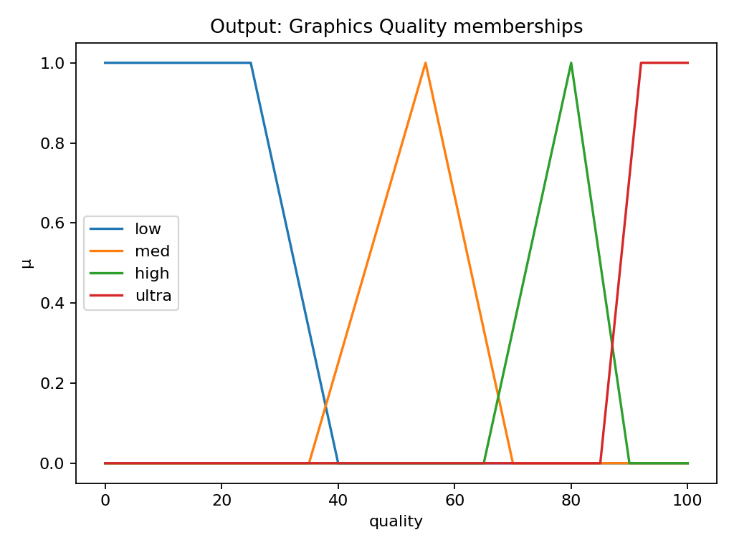
**รูปที่ 4.2**: *GPU score memberships*



**รูปที่ 4.3**: RAM *score memberships*

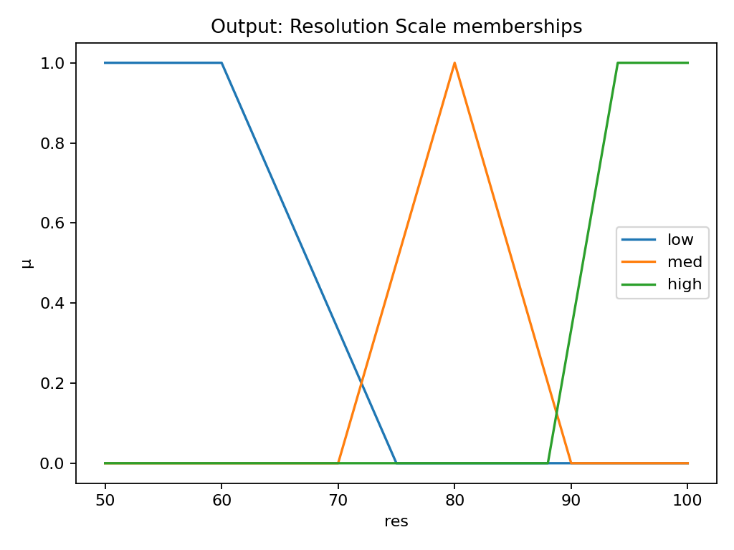
**รูปที่ 4.4**: Output: Graphics Quality memberships

* นิยามเอาต์พุตเป็น **LOW / MED / HIGH / ULTRA** โดย ULTRA ใช้ trapmf ช่วงบนสุด (≈ 92–100)
* ทำให้ผลลัพธ์ “ULTRA” เกิดขึ้นเฉพาะเมื่อฐานกฎสนับสนุนแรงจริง ๆ



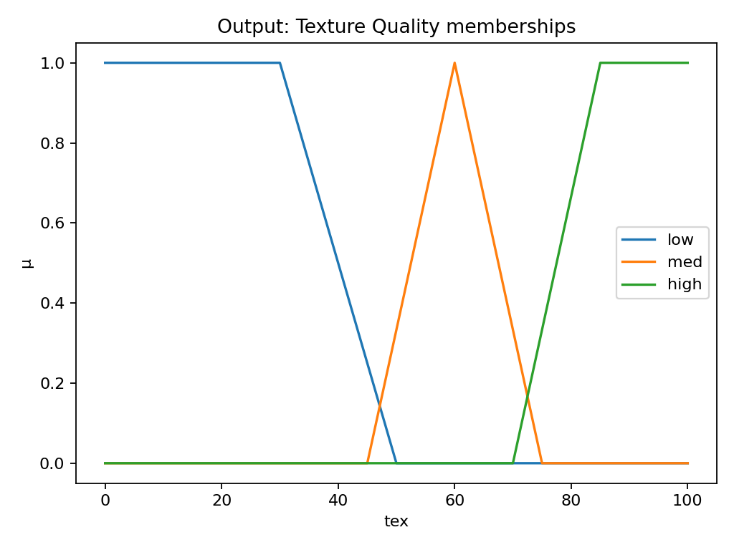
**รูปที่ 4.4**: : *Output: Graphics Quality memberships*

**รูปที่ 4.5**: *Output: Resolution Scale memberships*

* โดเมน 50–100% เพื่อกันไม่ให้ต่ำเกินใช้งานจริง
* ช่วง HIGH เริ่มตั้งแต่ ≈88% ขึ้นไป เหมาะกับเครื่องที่ GPU แข็งแรง

**รูปที่ 4.5**: *Output: Resolution Scale memberships*

**รูปที่ 4.6**: *Output: Texture Quality memberships*

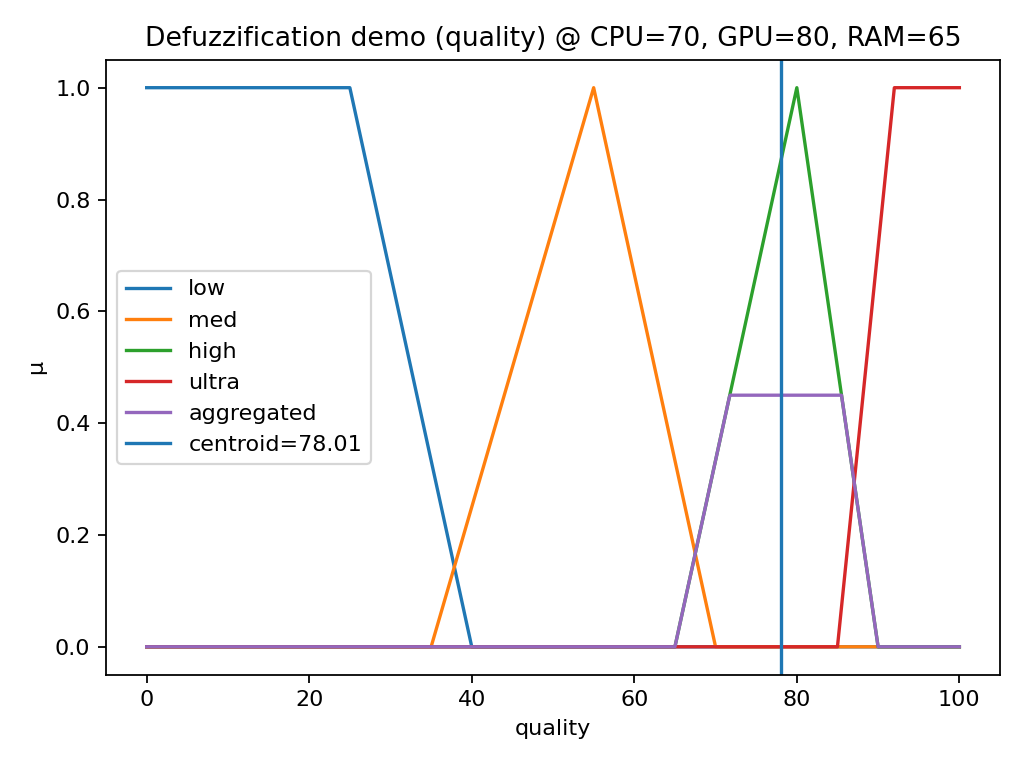
* เน้นช่วง “LOW” กว้างกว่าตัวอื่นเล็กน้อย เพื่อสะท้อนคอขวดด้านหน่วยความจำ
* ออกแบบให้ไต่ขึ้นสู่ “HIGH” เมื่อ RAM ดีพอ (≈70–85 ขึ้นไป)

**รูปที่ 4.6**: *Output: Texture Quality memberships*

**4.2 สาธิตกระบวนการ Defuzzification**

กราฟแสดงชุดสังกัดพื้นฐานของ **Quality** (LOW/MED/HIGH/ULTRA), กราฟ **aggregated** หลังถูก clip ตามระดับการลั่นกฎ และเส้น **centroid** ซึ่งเป็นค่าผลลัพธ์แบบตัวเลข

* แปลงค่าจริงของ CPU/GPU/RAM เป็นระดับความเป็นสมาชิกของคำภาษา (fuzzification)
* ประเมินกฎด้วย **AND=min / OR=max** ได้ค่าการลั่นกฎ w ของแต่ละกฎ
* **Implication=min**: นำ w ไปตัดยอดชุดสังกัดของเอาต์พุต
* **Aggregation=max**: รวมผลจากหลายกฎเป็นซองโค้งเส้นเดียว
* **Centroid**: หาจุดศูนย์ถ่วง → ค่าคำแนะนำ **Quality**

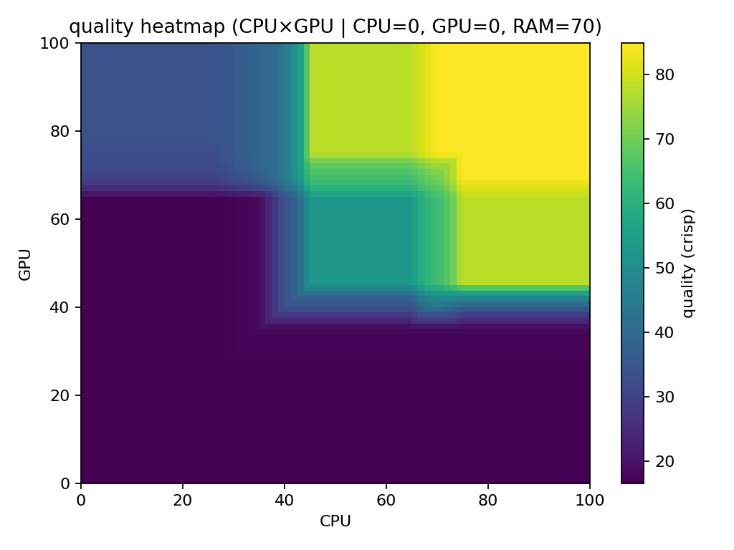


**รูปที่ 4.7**: *Defuzzification demo (Quality) @ CPU=70, GPU=80, RAM=65*

**4.3 Heatmaps: ปฏิสัมพันธ์ของอินพุตสองตัว**

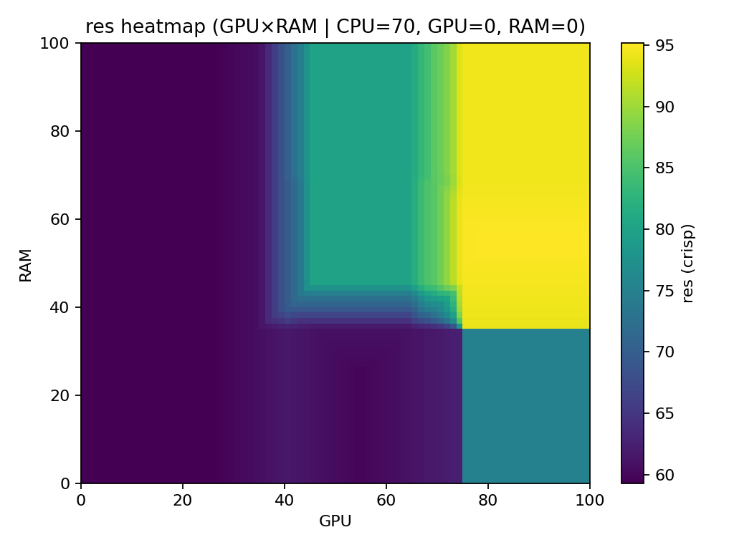
**รูปที่ 4.8:** *Quality heatmap (CPU×GPU | RAM=70)*

* สีเข้มขึ้นตามค่า Quality ที่สูงขึ้น เห็น “synergy” ระหว่าง CPU และ GPU: หากตัวใดตัวหนึ่งต่ำ คุณภาพจะถูกจำกัด
* เส้นชั้นสีลื่นไหล สอดคล้องกับการออกแบบ MF และ aggregation



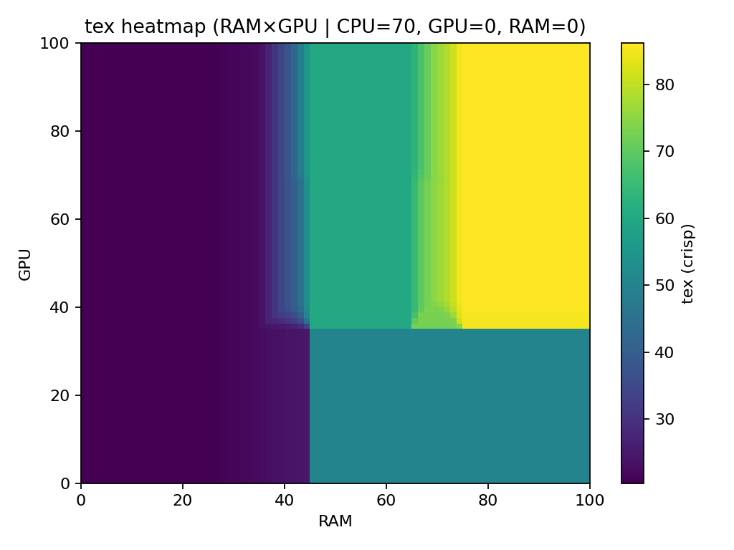
**รูปที่ 4.8:** *Quality heatmap (CPU×GPU | RAM=70)*

**รูปที่ 4.9**: *Resolution heatmap (GPU×RAM | CPU=70)*

* เห็นชัดว่า Resolution **พึ่งพา GPU เด่น** (แกน GPU เพิ่ม → Resolution เพิ่ม)
* RAM มีผลรอง: หาก RAM ต่ำมาก Resolution สูงสุดที่เข้าถึงได้อาจไม่ถึง “HIGH” แม้ GPU ปานกลาง–สูง

**รูปที่ 4.10**: *Texture heatmap (RAM×GPU | CPU=70)*

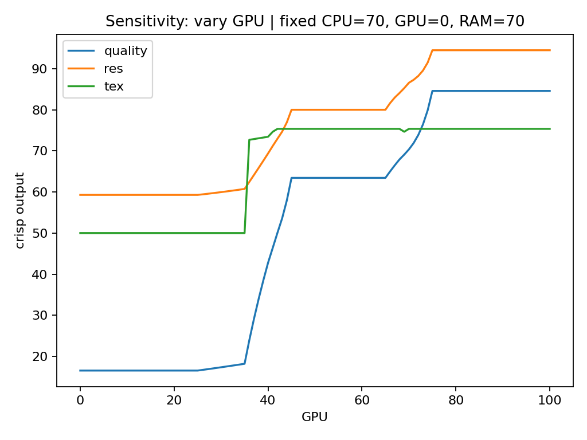
* รูปนี้ชี้ให้เห็นว่า **RAM เป็นคอขวด** ของ Texture ชัดเจน (แกน RAM ติดล่าง → Texture ต่ำ)
* เมื่อ RAM เพียงพอ, GPU ที่สูงช่วยเสริม Texture ในหลายกฎ (ผ่านการตั้งค่าที่พึ่งหน่วยความจำกราฟิกโดยอ้อม)



**รูปที่ 4.10**: *Texture heatmap (RAM×GPU | CPU=70)*

**4.4 กราฟความไว (Sensitivity)**

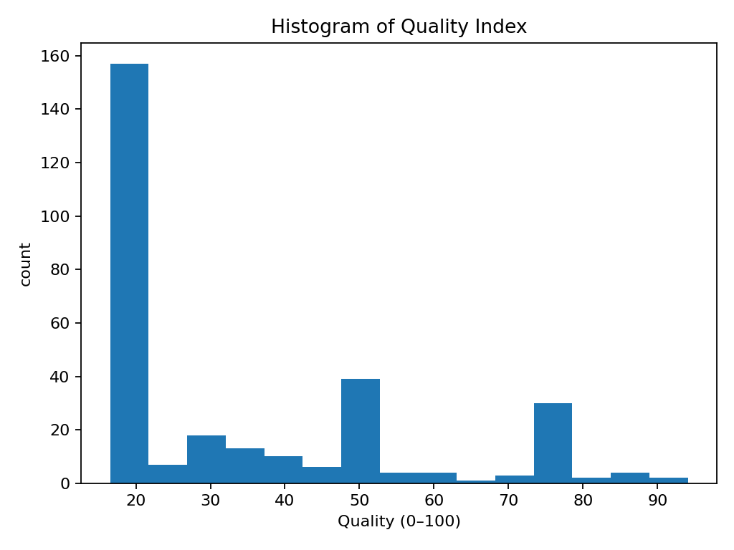
รูปที่ 4.11: *Sensitivity (vary GPU | CPU=70, RAM=70)* fig\_sensitivity\_vary\_gpu\_fix\_cpu70\_ram70.png

* แสดงเส้นผลลัพธ์ Quality / Resolution / Texture เมื่อไล่ GPU จาก 0→100 โดยตรึง CPU/RAM
* คาดหวังให้ Resolution เพิ่มตาม GPU อย่างเด่นชัด (เกือบโมโนโทน); Quality เพิ่มแต่โค้งจะค่อยกว่า; Texture มีผลบวกจาก GPU แต่ไม่แรงเท่า RAM
* จุดงอ/ไหล่ของเส้นตรงกับช่วงเปลี่ยน MF (เช่น MID→HIGH) ยืนยันว่าการออกแบบ MF มีผลต่อความชันของผลลัพธ์

**4.5 การสุ่มทดลองและสถิติภาพรวม**

**รูปที่ 4.12**: *Histogram of Quality Index*

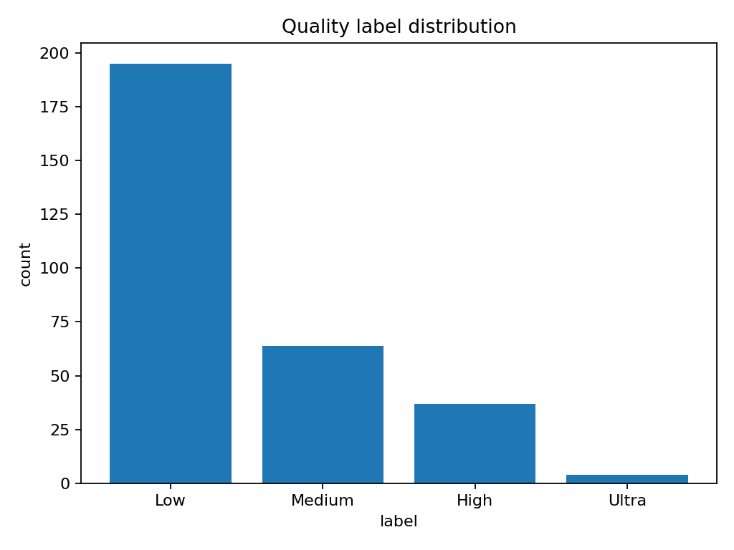
* การกระจายของค่า Quality จาก 300 เคสสุ่มในโดเมนใช้งานจริง
* ตรวจว่าการกระจายไม่ไปทางค่าต่ำหรือสูงจนเกินไป และไม่มีหางยาวที่ผิดสังเกต (ขึ้นกับฐานกฎและช่วงที่สุ่มอินพุต)

****

**รูปที่ 4.12**: *Histogram of Quality Index*

**รูปที่ 4.13**: *Quality label distribution*

* สัดส่วนฉลาก **Low / Medium / High / Ultra** ที่ระบบจัดให้จากการสุ่ม
* ตีความร่วมกับฐานกฎ: ถ้า **Medium/High** เด่น แปลว่าฐานกฎเอื้อเครื่องทั่ว ๆ ไปให้ค่าแนะนำระดับกลางขึ้นไป (สมเหตุสมผล) ส่วน **Ultra** มักมีสัดส่วนน้อย (เงื่อนไขเข้ม)



**บทที่ 5**

**สรุปผลการทดลอง**

**5.1 ภาพรวมผลลัพธ์**

โครงงานนี้พัฒนา **ระบบช่วยเลือกการตั้งค่าเกมอัตโนมัติ** ด้วยตรรกะฟัซซีแบบ **Mamdani** โดยใช้อินพุต 3 ตัว (CPU, GPU, RAM: 0–100) เพื่อแนะนำเอาต์พุต 3 ตัว (Graphics Quality 0–100, Resolution Scale 50–100%, Texture Quality 0–100) พร้อมฉลาก (Low/Medium/High/Ultra) และกลไก **Hysteresis** ลดการกระพริบของฉลาก ผลการทดลองจากกราฟ MF, เดโม Defuzzification, Heatmaps, Sensitivity และการสุ่ม 300 เคส แสดงว่าแบบจำลองให้ผลลัพธ์ **ต่อเนื่อง, ตีความง่าย, และสอดคล้องกับตรรกะของโดเมนเกม** ตามที่ตั้งเป้าไว้

**5.2 ข้อค้นพบเชิงเทคนิค (Key Findings)**

1. **อิทธิพลของ GPU ต่อ Resolution เด่นชัดที่สุด**
   * จาก Heatmap (GPU×RAM | CPU=70) และกราฟ Sensitivity (vary GPU) ค่า Resolution เพิ่มขึ้นเรียบและเกือบโมโนโทนตาม GPU
2. **RAM เป็นคอขวดของ Texture อย่างมีนัยสำคัญ**
   * Heatmap (RAM×GPU | CPU=70) แสดงว่าเมื่อ RAM ต่ำ Texture ถูกกดลงแม้ GPU ดี ซึ่งสะท้อนสถานการณ์จริง (โหลด/สตรีมมิงเท็กซ์เจอร์)
3. **Quality ต้องการพลัง CPU+GPU ร่วมกัน (synergy)**
   * Heatmap (CPU×GPU | RAM=70) ชี้ว่าคุณภาพโดยรวมสูงจะเกิดเมื่อทั้ง CPU และ GPU อยู่ระดับกลาง–สูงขึ้นไป พร้อมกัน
4. **กฎแบบน้ำหนัก (0.8–0.9)** ช่วยผ่อนแรงความขัดแย้งของกฎ
   * ตัวอย่าง R6/R8 ทำให้ผลรวมลื่นไหลขึ้นเมื่อองค์ประกอบยังไม่ครบ “สูงทุกตัว”
5. **ผลลัพธ์ค่อยเป็นค่อยไป (smooth)**
   * กราฟ Sensitivity, Heatmap และผลจาก Centroid แสดงว่า **เปลี่ยนอินพุตเล็กน้อย → เอาต์พุตไม่กระโดด** ตรงวัตถุประสงค์

**5.3 ความถูกต้องและความสอดคล้องกับโดเมน (Validation)**

* **โดเมนตรรกะผ่าน**:
  + GPU สูง → Resolution สูง (คง CPU/RAM)
  + RAM ต่ำ → Texture ต่ำ
  + CPU+GPU สูงพร้อมกัน → Quality สูง
* **ความต่อเนื่องผ่าน**: ทุกกราฟไม่มีจุดหัก หรือโครงลายเสื่อที่ผิดธรรมชาติ
* **ฉลากเสถียรขึ้นด้วย Hysteresis**: ลดการสลับก่อน–หลังเส้นแบ่งในโซนก้ำกึ่ง

**5.4 ประสิทธิภาพเชิงคำนวณ**

* โดเมนถูกแยกละเอียดที่ 1001 จุด (0–100 และ 50–100 สำหรับ Resolution) ทำให้ **Centroid** คำนวณเร็วและแม่นพอ โดยรวม **เวลาประมวลผล/อินพุต 1 ชุดอยู่ในระดับมิลลิวินาที** (ขึ้นกับเครื่อง) จึงเหมาะกับการนำไปเรียกใช้แบบอินเทอร์แอกทีฟ

**5.5 ข้อจำกัดของงาน**

1. **สเกลคะแนน 0–100 เป็นเชิงอนุโลม**: หากคะแนนไม่อ้างอิง benchmark จริง ผลลัพธ์อาจต่างจากการใช้งานจริง
2. **กฎคงที่**: เหมาะกับ “เกมทั่วไป” ถ้าเกมใดพึ่ง CPU/VRAM มากผิดปกติ อาจต้องปรับ MF/กฎเฉพาะ
3. **ไม่มีเป้าหมาย FPS ระบุชัด**: ระบบนี้แนะนำ “แนวโน้มการตั้งค่า” ไม่รับประกันจำนวนเฟรมต่อวินาที
4. **ไม่รวมปัจจัย VRAM/Threads/สถาปัตยกรรม**: ใส่ RAM รวมแล้ว แต่ยังไม่แยก VRAM/CPU threads/รุ่น GPU แบบละเอียด

**5.6 คำแนะนำสำหรับการใช้งานจริง**

* ใช้ระบบนี้เป็น **จุดเริ่มต้นตั้งค่า** แล้วให้ผู้ใช้ปรับเพิ่ม–ลดละเอียดตามรสนิยม/จอ/เกม
* หากพบ **stutter/โหลดช้า** ให้ลด Texture ก่อน (สะท้อนข้อจำกัด RAM/VRAM)
* ถ้า FPS ไม่ถึงเป้า ให้ **ลด Resolution Scale** ก่อน แล้วค่อยลด Quality รายพารามิเตอร์ (เงา/แสง/โพสต์โปรเซส)

**ภาคผนวก**

import numpy as np

import matplotlib.pyplot as plt

from typing import Callable, Dict, List, Tuple

import pandas as pd

import os

# ============================================

# อินพุต: CPU/GPU/RAM (0..100)

# เอาต์พุต: Graphics Quality (0..100), Resolution Scale (50..100), Texture (0..100)

# ขั้นตอน: fuzzification -> evaluate rules (AND=min, OR=max)

#          -> implication=min -> aggregation=max -> centroid defuzz

# ============================================

# -------- Membership helpers --------

def trimf(x, abc):

    """ Triangular MF: จุด (a,b,c) — ไต่ขึ้น a->b แล้วไต่ลง b->c """

    a, b, c = abc

    y = np.zeros\_like(x, dtype=float)

    # ไต่ขึ้น

    idx = (a < x) & (x <= b); y[idx] = (x[idx] - a) / (b - a) if b != a else 0.0

    # ไต่ลง

    idx = (b < x) & (x < c);  y[idx] = (c - x[idx]) / (c - b) if c != b else 0.0

    # จุดยอด (b) = 1

    y[x == b] = 1.0

    return np.clip(y, 0, 1)

def trapmf(x, abcd):

    """ Trapezoidal MF: จุด (a,b,c,d) — ไต่ขึ้น a->b, ราบ b..c, ไต่ลง c->d """

    a, b, c, d = abcd

    y = np.zeros\_like(x, dtype=float)

    # ไต่ขึ้น

    idx = (a < x) & (x <= b); y[idx] = (x[idx] - a) / (b - a) if b != a else 0.0

    # ช่วงราบ

    idx = (b < x) & (x <= c); y[idx] = 1.0

    # ไต่ลง

    idx = (c < x) & (x < d);  y[idx] = (d - x[idx]) / (d - c) if d != c else 0.0

    y[(x == b) | (x == c)] = 1.0

    return np.clip(y, 0, 1)

def defuzz\_centroid(x, mu):

    """ Centroid of area (CoA): ค่าศูนย์ถ่วงของพื้นที่ใต้โค้ง mu(x) """

    area = np.trapz(mu, x)

    if area == 0:

        # ถ้าไม่มีพื้นที่ (ไม่มีกฎลั่น) คืนค่ากลางโดเมนเพื่อกันหารศูนย์

        return 0.5 \* (x[0] + x[-1])

    return float(np.trapz(x \* mu, x) / area)

# -------- โครงสร้างพื้นฐานของระบบฟัซซี --------

class FuzzySet:

    """ 1 ชุดสังกัด (เช่น 'low', 'mid', 'high') พร้อมฟังก์ชัน membership """

    def \_\_init\_\_(self, name: str, mf: Callable[[np.ndarray], np.ndarray]):

        self.name = name

        self.mf = mf

class FuzzyVar:

    """ ตัวแปรฟัซซี 1 ตัว: มีโดเมน (universe) + เซตย่อย (FuzzySet) หลายชุด """

    def \_\_init\_\_(self, name: str, universe: np.ndarray, sets: Dict[str, FuzzySet]):

        self.name = name

        self.universe = universe

        self.sets = sets

    def mu(self, set\_name: str, xval: float) -> float:

        """ อ่านค่า membership (0..1) ของ set\_name ที่ตำแหน่ง xval """

        xs = self.universe

        mus = self.sets[set\_name].mf(xs)

        return float(np.interp(xval, xs, mus))  # interpolate ให้ค่าเดียวจากโค้ง

class Rule:

    """ กฎฟัซซี: antecedent (degree 0..1) -> consequents [(out\_var, set\_name, weight)] """

    def \_\_init\_\_(self,

                 antecedent: Callable[[Dict[str, float]], float],

                 consequents: List[Tuple[str, str, float]]):

        self.antecedent = antecedent

        self.consequents = consequents

class MamdaniSystem:

    """ แกนหลักของ Mamdani: implication=min, aggregation=max, defuzz=centroid """

    def \_\_init\_\_(self, inputs: Dict[str, FuzzyVar], outputs: Dict[str, FuzzyVar], rules: List[Rule]):

        self.inputs, self.outputs, self.rules = inputs, outputs, rules

    def infer(self, x: Dict[str, float]) -> Dict[str, float]:

        # เริ่มกราฟสะสมของแต่ละเอาต์พุตเป็นศูนย์ (สำหรับ aggregation)

        agg = {name: np.zeros\_like(var.universe, dtype=float) for name, var in self.outputs.items()}

        for rule in self.rules:

            # 1) ประเมินส่วน IF: AND=min, OR=max (นิยามใน lambda ของ RULES)

            w = float(rule.antecedent(x))

            if w <= 0:

                continue

            # 2) implication = min: clip ชุดสังกัดของเอาต์พุตด้วยระดับ w (คูณน้ำหนักได้)

            for out\_name, set\_name, weight in rule.consequents:

                var = self.outputs[out\_name]

                mu\_set = var.sets[set\_name].mf(var.universe)

                clipped = np.minimum(mu\_set, w \* weight)

                # 3) aggregation = max: รวมหลายกฎเข้าด้วยกันโดยเอาค่าสูงสุดในแต่ละจุด

                agg[out\_name] = np.maximum(agg[out\_name], clipped)

        # 4) defuzz ทุกเอาต์พุตด้วย centroid

        return {name: defuzz\_centroid(var.universe, agg[name]) for name, var in self.outputs.items()}

# -------- โดเมนของตัวแปร (แกน x) --------

u  = np.linspace(0, 100, 1001)   # ใช้กับตัวแปรที่อยู่ใน 0..100 (ละเอียด 1001 จุด)

uR = np.linspace(50, 100, 1001)  # Resolution scale จำกัด 50..100

def mk\_perf\_sets():

    """ นิยาม MF ของอินพุต CPU/GPU/RAM: low/mid/high """

    return {

        "low":  FuzzySet("low",  lambda x: trapmf(x, (0, 0, 25, 45))),   # ต่ำชัด 0..25, ค่อยๆ เลิกต่ำ 25..45

        "mid":  FuzzySet("mid",  lambda x: trimf(x, (35, 55, 75))),      # กลางพอดีที่ 55

        "high": FuzzySet("high", lambda x: trapmf(x, (65, 80, 100, 100)))# เริ่มสูง ~65, สูงชัด 80..100

    }

# -------- อินพุต 3 ตัว --------

inputs = {

    "cpu": FuzzyVar("cpu", u, mk\_perf\_sets()),

    "gpu": FuzzyVar("gpu", u, mk\_perf\_sets()),

    "ram": FuzzyVar("ram", u, mk\_perf\_sets()),

}

# -------- เอาต์พุต: MF ของ Quality / Resolution / Texture --------

quality\_sets = {

    "low":   FuzzySet("low",   lambda x: trapmf(x, (0, 0, 25, 40))),

    "med":   FuzzySet("med",   lambda x: trimf(x, (35, 55, 70))),

    "high":  FuzzySet("high",  lambda x: trimf(x, (65, 80, 90))),

    "ultra": FuzzySet("ultra", lambda x: trapmf(x, (85, 92, 100, 100))), # ultra เฉพาะกรณีแรงมาก

}

res\_sets = {

    "low":   FuzzySet("low",   lambda x: trapmf(x, (50, 50, 60, 75))),   # scale ไม่ต่ำกว่า 50%

    "med":   FuzzySet("med",   lambda x: trimf(x, (70, 80, 90))),

    "high":  FuzzySet("high",  lambda x: trapmf(x, (88, 94, 100, 100))),

}

tex\_sets = {

    "low":   FuzzySet("low",   lambda x: trapmf(x, (0, 0, 30, 50))),     # RAM ต่ำ = เท็กซ์เจอร์ต่ำ

    "med":   FuzzySet("med",   lambda x: trimf(x, (45, 60, 75))),

    "high":  FuzzySet("high",  lambda x: trapmf(x, (70, 85, 100, 100))),

}

# -------- ประกาศตัวแปรเอาต์พุต --------

outputs = {

    "quality": FuzzyVar("quality", u,  quality\_sets),

    "res":     FuzzyVar("res",     uR, res\_sets),

    "tex":     FuzzyVar("tex",     u,  tex\_sets),

}

def MU(v, s, x):

    """ ช็อตคัต: μ(ตัวแปร v เป็นเซต s) ที่ค่าอินพุต x[v] """

    return inputs[v].mu(s, x[v])

# -------- กฎแบบภาษาคน (ใช้ใส่รายงาน) --------

RULES\_HUMAN = [

    "R1: IF CPU is HIGH AND GPU is HIGH AND RAM is HIGH THEN Quality is ULTRA, Resolution is HIGH, Texture is HIGH",

    "R2: IF GPU is HIGH AND CPU is MID AND RAM is MID THEN Quality is HIGH, Resolution is HIGH, Texture is MED",

    "R3: IF CPU is MID AND GPU is MID AND RAM is MID THEN Quality is MED, Resolution is MED, Texture is MED",

    "R4: IF GPU is LOW OR CPU is LOW THEN Quality is LOW, Resolution is LOW",

    "R5: IF RAM is LOW THEN Texture is LOW",

    "R6: IF GPU is HIGH AND RAM is MID THEN Quality is HIGH, Resolution is HIGH, Texture is MED (weighted)",

    "R7: IF CPU is MID AND GPU is HIGH AND RAM is HIGH THEN Quality is HIGH, Resolution is HIGH, Texture is HIGH",

    "R8: IF CPU is HIGH AND GPU is MID AND RAM is HIGH THEN Quality is HIGH (slightly), Resolution is MED, Texture is HIGH",

    "R9: IF GPU is MID AND RAM is LOW THEN Texture is LOW; Resolution tends to LOW",

    "R10: IF CPU is LOW AND GPU is HIGH THEN Quality is MED, Resolution is MED",

]

# -------- กฎใช้งานจริง (AND=min, OR=max, implication=min, aggregation=max) --------

RULES = [

    # R1: ทั้งสามสูง -> คุณภาพรวม/สเกล/เท็กซ์เจอร์ สูงสุด

    Rule(lambda x: min(MU('cpu','high',x), MU('gpu','high',x), MU('ram','high',x)),

         [('quality','ultra',1.0), ('res','high',1.0), ('tex','high',1.0)]),

    # R2: GPU สูง, CPU/RAM กลาง -> คุณภาพ/สเกล สูง, เท็กซ์เจอร์ กลาง

    Rule(lambda x: min(MU('gpu','high',x), MU('cpu','mid',x), MU('ram','mid',x)),

         [('quality','high',1.0), ('res','high',1.0), ('tex','med',1.0)]),

    # R3: ทั้งหมดกลาง -> เอาต์พุตกลาง

    Rule(lambda x: min(MU('cpu','mid',x), MU('gpu','mid',x), MU('ram','mid',x)),

         [('quality','med',1.0), ('res','med',1.0), ('tex','med',1.0)]),

    # R4: GPU ต่ำ หรือ CPU ต่ำ -> กดคุณภาพรวม/สเกล ลง

    Rule(lambda x: max(MU('gpu','low',x), MU('cpu','low',x)),

         [('quality','low',1.0), ('res','low',1.0)]),

    # R5: RAM ต่ำ -> เท็กซ์เจอร์ต่ำ (คอขวดหน่วยความจำ)

    Rule(lambda x: MU('ram','low',x),

         [('tex','low',1.0)]),

    # R6: GPU สูง + RAM กลาง -> ดันคุณภาพ/สเกลขึ้น แต่ใส่ weight 0.9 ให้ไม่ชนกับ R1 แรงเกิน

    Rule(lambda x: min(MU('gpu','high',x), MU('ram','mid',x)),

         [('quality','high',0.9), ('res','high',0.9), ('tex','med',1.0)]),

    # R7: CPU กลาง + GPU สูง + RAM สูง -> เอาต์พุตสูง

    Rule(lambda x: min(MU('cpu','mid',x), MU('gpu','high',x), MU('ram','high',x)),

         [('quality','high',1.0), ('res','high',1.0), ('tex','high',1.0)]),

    # R8: CPU สูง + GPU กลาง + RAM สูง -> Quality สูง (0.8) เพื่อละมุน, Res กลาง, Tex สูง

    Rule(lambda x: min(MU('cpu','high',x), MU('gpu','mid',x), MU('ram','high',x)),

         [('quality','high',0.8), ('res','med',1.0), ('tex','high',1.0)]),

    # R9: GPU กลาง + RAM ต่ำ -> เท็กซ์เจอร์ต่ำ; สเกลมีแนวโน้มต่ำ (0.8)

    Rule(lambda x: min(MU('gpu','mid',x), MU('ram','low',x)),

         [('tex','low',1.0), ('res','low',0.8)]),

    # R10: CPU ต่ำ + GPU สูง -> คุณภาพ/สเกล ระดับกลาง (คอขวด CPU)

    Rule(lambda x: min(MU('cpu','low',x), MU('gpu','high',x)),

         [('quality','med',1.0), ('res','med',1.0)]),

]

# ประกอบระบบ

system = MamdaniSystem(inputs, outputs, RULES)

# -------- Numeric -> Label mapping (ไว้โชว์/รายงาน/ UI) --------

def quality\_label(q: float) -> str:

    if q < 37.5: return "Low"

    if q < 67.5: return "Medium"

    if q < 87.5: return "High"

    return "Ultra"

def resolution\_label(res: float) -> str:

    if res < 72.5: return "Low"

    if res < 89.0: return "Medium"

    return "High"

def texture\_label(tex: float) -> str:

    if tex < 47.5: return "Low"

    if tex < 72.5: return "Medium"

    return "High"

# -------- Hysteresis: กันฉลาก "กระพริบ" เมื่อค่าใกล้เส้นแบ่ง --------

class HysteresisMapper:

    """

    levels: รายการฉลาก (เรียงลำดับจากต่ำไปสูง)

    rise: เกณฑ์ "ขยับขึ้นระดับ"

    fall: เกณฑ์ "ขยับลงระดับ"

    """

    def \_\_init\_\_(self, levels, rise, fall, initial=None):

        self.levels, self.rise, self.fall = levels, rise, fall

        self.idx = 0 if initial is None else levels.index(initial)

    def update(self, x: float) -> str:

        # ขึ้นระดับเมื่อถึงเกณฑ์ rise

        while self.idx < len(self.rise) and x >= self.rise[self.idx]:

            self.idx += 1

        # ลงระดับเมื่อค่าต่ำกว่าเกณฑ์ fall

        while self.idx > 0 and x < self.fall[self.idx - 1]:

            self.idx -= 1

        return self.levels[self.idx]

quality\_mapper = HysteresisMapper(["Low","Medium","High","Ultra"], [40.0, 70.0, 90.0], [35.0, 65.0, 85.0])

res\_mapper     = HysteresisMapper(["Low","Medium","High"],          [75.0, 90.0],       [70.0, 88.0])

tex\_mapper     = HysteresisMapper(["Low","Medium","High"],          [50.0, 75.0],       [45.0, 70.0])

# -------- APIs หลักสำหรับเรียกใช้งาน --------

def infer(cpu: float, gpu: float, ram: float) -> Dict[str, float]:

    """ คืนค่าเอาต์พุตแบบตัวเลข (crisp) ทั้ง 3 ตัว จากอินพุต cpu/gpu/ram """

    return system.infer({"cpu": cpu, "gpu": gpu, "ram": ram})

def infer\_with\_labels(cpu: float, gpu: float, ram: float, use\_hysteresis: bool = False) -> Dict[str, object]:

    """ คืนทั้งค่าเลข + ฉลาก (เลือกใช้ hysteresis ได้) — เหมาะกับ UI/รายงาน """

    out = infer(cpu, gpu, ram)

    if use\_hysteresis:

        qlbl = quality\_mapper.update(out["quality"])

        rlbl = res\_mapper.update(out["res"])

        tlbl = tex\_mapper.update(out["tex"])

    else:

        qlbl = quality\_label(out["quality"])

        rlbl = resolution\_label(out["res"])

        tlbl = texture\_label(out["tex"])

    return {

        "cpu": cpu, "gpu": gpu, "ram": ram,

        "quality\_idx": round(out["quality"], 2), "quality\_lbl": qlbl,

        "res\_scale\_%": round(out["res"], 1),     "res\_label": rlbl,

        "texture\_idx": round(out["tex"], 1),     "texture\_lbl": tlbl,

    }

def get\_rules\_text() -> str:

    """ รวมกฎแบบภาษาคนไว้ใช้แสดงในรายงาน/คอนโซล """

    return "\n".join(RULES\_HUMAN)

# -------- Utilities วาดกราฟ (สำหรับรายงาน) --------

def plot\_memberships(var, title, filename=None):

    """ วาดกราฟ MF ของตัวแปรหนึ่งตัว (อินพุตหรือเอาต์พุต) """

    plt.figure()

    for name, fset in var.sets.items():

        plt.plot(var.universe, fset.mf(var.universe), label=name)

    plt.title(title)

    plt.xlabel(var.name)

    plt.ylabel("μ")

    plt.legend(loc="best")

    plt.tight\_layout()

    if filename:

        plt.savefig(filename, dpi=160)

        plt.close()

    else:

        plt.show()

def demo\_defuzz(cpu=70, gpu=80, ram=65, out\_name="quality", filename=None):

    """ เดโม: โชว์ขั้น clip/aggregate และ centroid ของเอาต์พุตชื่อ out\_name """

    x = {"cpu": cpu, "gpu": gpu, "ram": ram}

    # รวมผลหลายกฎ (เฉพาะเอาต์พุตที่สนใจ) ลงในกราฟ agg

    agg = np.zeros\_like(outputs[out\_name].universe, dtype=float)

    for rule in RULES:

        w = float(rule.antecedent(x))

        if w <= 0:

            continue

        for o\_name, set\_name, weight in rule.consequents:

            if o\_name != out\_name:

                continue

            var = outputs[o\_name]

            mu = var.sets[set\_name].mf(var.universe)

            clipped = np.minimum(mu, w \* weight)   # implication=min

            agg = np.maximum(agg, clipped)         # aggregation=max

    xs = outputs[out\_name].universe

    c = defuzz\_centroid(xs, agg)                  # centroid = คำตอบแบบตัวเลข

    # วาดทั้งเซตพื้นฐาน + กราฟ aggregated + เส้น centroid

    plt.figure()

    for name, fset in outputs[out\_name].sets.items():

        plt.plot(xs, fset.mf(xs), label=name)

    plt.plot(xs, agg, label="aggregated")

    plt.axvline(c, label=f"centroid={c:.2f}")

    plt.title(f"Defuzzification demo ({out\_name}) @ CPU={cpu}, GPU={gpu}, RAM={ram}")

    plt.xlabel(out\_name)

    plt.ylabel("μ")

    plt.legend(loc="best")

    plt.tight\_layout()

    if filename:

        plt.savefig(filename, dpi=160)

        plt.close()

    else:

        plt.show()

    return c

def plot\_heatmap(x\_name, y\_name, out\_name, fixed, filename=None):

    """ Heatmap เอาต์พุต out\_name บนระนาบอินพุต 2 ตัว (อีกตัวตรึงตาม fixed) """

    nx = 80; ny = 80  # ปรับความละเอียดได้

    xs = np.linspace(0, 100, nx)

    ys = np.linspace(0, 100, ny)

    Z = np.zeros((ny, nx), dtype=float)

    for j, yv in enumerate(ys):

        for i, xv in enumerate(xs):

            sample = {\*\*fixed, x\_name: float(xv), y\_name: float(yv)}

            val = infer(sample["cpu"], sample["gpu"], sample["ram"])[out\_name]

            Z[j, i] = val

    plt.figure()

    plt.imshow(Z, origin="lower", extent=[xs.min(), xs.max(), ys.min(), ys.max()], aspect="auto")

    plt.colorbar(label=f"{out\_name} (crisp)")

    plt.xlabel(x\_name.upper())

    plt.ylabel(y\_name.upper())

    fixed\_txt = ", ".join(f"{k.upper()}={v}" for k,v in fixed.items())

    plt.title(f"{out\_name} heatmap ({x\_name.upper()}×{y\_name.upper()} | {fixed\_txt})")

    plt.tight\_layout()

    if filename:

        plt.savefig(filename, dpi=160)

        plt.close()

    else:

        plt.show()

def sensitivity\_curve(vary, fixed, out\_names=("quality","res","tex"), filename=None):

    """ ไล่ค่าตัวแปรหนึ่งตัว (vary) เพื่อดูความไวของเอาต์พุตที่เลือก """

    xs = np.linspace(0,100,101)

    plt.figure()

    for out\_name in out\_names:

        ys = []

        for xval in xs:

            sample = {\*\*fixed, vary: float(xval)}

            out = infer(sample["cpu"], sample["gpu"], sample["ram"])

            ys.append(out[out\_name])

        plt.plot(xs, ys, label=out\_name)

    plt.title(f"Sensitivity: vary {vary.upper()} | fixed " + ", ".join(f"{k.upper()}={v}" for k,v in fixed.items()))

    plt.xlabel(vary.upper())

    plt.ylabel("crisp output")

    plt.legend(loc="best")

    plt.tight\_layout()

    if filename:

        plt.savefig(filename, dpi=160)

        plt.close()

    else:

        plt.show()

def random\_stats(n=300, csv\_path="fuzzy\_sim\_results.csv", fig\_hist=None, fig\_bar=None):

    """ สุ่มอินพุต n เคส -> เก็บผล/ฉลากเป็น CSV + วาดฮิสโตแกรม/แท่งสัดส่วนฉลาก """

    rng = np.random.default\_rng(42)  # seed เดิมเพื่อทำซ้ำ

    rows = []

    lblQ = {"Low":0,"Medium":0,"High":0,"Ultra":0}

    lblR = {"Low":0,"Medium":0,"High":0}

    lblT = {"Low":0,"Medium":0,"High":0}

    Q, R, T = [], [], []

    for \_ in range(n):

        cpu = float(rng.uniform(5,95))

        gpu = float(rng.uniform(5,95))

        ram = float(rng.uniform(5,95))

        out = infer\_with\_labels(cpu, gpu, ram, use\_hysteresis=False)

        rows.append(out)

        Q.append(out["quality\_idx"]); lblQ[out["quality\_lbl"]] += 1

        R.append(out["res\_scale\_%"]); lblR[out["res\_label"]] += 1

        T.append(out["texture\_idx"]); lblT[out["texture\_lbl"]] += 1

    df = pd.DataFrame(rows)

    df.to\_csv(csv\_path, index=False, encoding="utf-8")

    # Histogram of Quality

    plt.figure()

    plt.hist(Q, bins=15)

    plt.title("Histogram of Quality Index")

    plt.xlabel("Quality (0–100)")

    plt.ylabel("count")

    plt.tight\_layout()

    if fig\_hist:

        plt.savefig(fig\_hist, dpi=160)

        plt.close()

    else:

        plt.show()

    # Bar: Quality label distribution

    plt.figure()

    xs = list(lblQ.keys()); ys = [lblQ[k] for k in xs]

    plt.bar(xs, ys)

    plt.title("Quality label distribution")

    plt.xlabel("label")

    plt.ylabel("count")

    plt.tight\_layout()

    if fig\_bar:

        plt.savefig(fig\_bar, dpi=160)

        plt.close()

    else:

        plt.show()

    return df

# -------- Main: ตัวอย่างรัน + สร้างรูปสำหรับรายงาน --------

if \_\_name\_\_ == "\_\_main\_\_":

    outdir = os.getcwd()

    # ตัวอย่างอนุมาน 1 ชุด + แสดงกฎภาษาคน

    print(infer\_with\_labels(90, 40, 95))

    print("--- Rules ---")

    print(get\_rules\_text())

    # 1) Memberships

    plot\_memberships(inputs["cpu"], "CPU score memberships", os.path.join(outdir, "fig\_membership\_cpu.png"))

    plot\_memberships(inputs["gpu"], "GPU score memberships", os.path.join(outdir, "fig\_membership\_gpu.png"))

    plot\_memberships(inputs["ram"], "RAM score memberships", os.path.join(outdir, "fig\_membership\_ram.png"))

    plot\_memberships(outputs["quality"], "Output: Graphics Quality memberships", os.path.join(outdir, "fig\_membership\_quality.png"))

    plot\_memberships(outputs["res"], "Output: Resolution Scale memberships", os.path.join(outdir, "fig\_membership\_res.png"))

    plot\_memberships(outputs["tex"], "Output: Texture Quality memberships", os.path.join(outdir, "fig\_membership\_tex.png"))

    # 2) Defuzz demo

    demo\_defuzz(70, 80, 65, "quality", os.path.join(outdir, "fig\_defuzz\_quality\_demo.png"))

    # 3) Heatmaps

    plot\_heatmap("cpu","gpu","quality", fixed={"cpu":0,"gpu":0,"ram":70}, filename=os.path.join(outdir, "fig\_heatmap\_quality\_cpu\_gpu\_ram70.png"))

    plot\_heatmap("gpu","ram","res", fixed={"cpu":70,"gpu":0,"ram":0}, filename=os.path.join(outdir, "fig\_heatmap\_res\_gpu\_ram\_cpu70.png"))

    plot\_heatmap("ram","gpu","tex", fixed={"cpu":70,"gpu":0,"ram":0}, filename=os.path.join(outdir, "fig\_heatmap\_tex\_ram\_gpu\_cpu70.png"))

    # 4) Sensitivity

    sensitivity\_curve("gpu", {"cpu":70,"gpu":0,"ram":70},

                      filename=os.path.join(outdir, "fig\_sensitivity\_vary\_gpu\_fix\_cpu70\_ram70.png"))

    # 5) Random stats + CSV

    random\_stats(n=300,

                 csv\_path=os.path.join(outdir, "fuzzy\_sim\_results.csv"),

                 fig\_hist=os.path.join(outdir, "fig\_hist\_quality.png"),

                 fig\_bar=os.path.join(outdir, "fig\_bar\_quality\_labels.png"))