Gemini Chat

ยินดีครับ เพื่อให้คุณเข้าใจเนื้อหาเรื่อง **Image Segmentation (การแบ่งส่วนภาพ)** จากเอกสารนี้อย่างลึกซึ้งที่สุด ผมได้ออกแบบโครงสร้างการเรียนรู้แบ่งเป็น 6 ส่วนหลัก เพื่อค่อยๆ ปูพื้นฐานจากทฤษฎียุคเก่าไปจนถึง Deep Learning สมัยใหม่ ดังนี้ครับ:

1. **Part 1: รากฐานและจิตวิทยาการมองเห็น (Foundations & Human Perception)**
2. **Part 2: การแบ่งส่วนภาพด้วยวิธี Clustering (Classical Clustering: K-Means & Mean Shift)**
3. **Part 3: การแบ่งส่วนภาพด้วยทฤษฎีกราฟ (Graph-Based Segmentation)**
4. **Part 4: ก้าวสู่ Deep Learning - Semantic Segmentation (FCN)**
5. **Part 5: เทคนิคการขยายภาพและสถาปัตยกรรม U-Net (Upsampling & U-Net)**
6. **Part 6: Instance & Panoptic Segmentation (Mask R-CNN & Beyond)**

เรามาเริ่มกันที่ส่วนแรกเลยครับ

Part 1: รากฐานและจิตวิทยาการมองเห็น (Foundations & Human Perception)

ก่อนที่เราจะเขียนอัลกอริทึม เราต้องเข้าใจก่อนว่า "การแบ่งส่วนภาพ" คืออะไร และทำไมมันถึงเป็นเรื่องยากสำหรับคอมพิวเตอร์ แต่เป็นเรื่องง่ายสำหรับมนุษย์

1. Image Segmentation คืออะไร?

Image Segmentation คือกระบวนการแบ่งภาพดิจิทัลออกเป็นหลายๆ ส่วน (segments) หรือกลุ่มของพิกเซล (sets of pixels) เป้าหมายคือ

การเปลี่ยนการแทนภาพให้อยู่ในรูปแบบที่มีความหมายและง่ายต่อการวิเคราะห์มากขึ้น

- **เป้าหมายสูงสุด:** แยกวัตถุออกจากพื้นหลัง หรือแยกวัตถุแต่ละชิ้นออกจากกัน

2. มนุษย์มองเห็นโลกอย่างไร? (Gestalt Psychology)

เอกสารนี้เน้นย้ำเรื่อง **Gestalt Psychology (จิตวิทยากลุ่ม)** ซึ่งเป็นทฤษฎีที่อธิบายว่าสมองของมนุษย์ไม่ได้มองเห็นโลกเป็นจุดพิกเซลแยกจากกัน แต่มองเห็นเป็น "กลุ่มก้อน" หรือ "องค์รวม" โดยอัตโนมัติ

หลักการสำคัญที่คอมพิวเตอร์พยายามเลียนแบบมีดังนี้:

- **Proximity (ความใกล้ชิด):** สิ่งที่อยู่ใกล้กัน มักจะถูกมองว่าเป็นกลุ่มเดียวกัน
- **Similarity (ความเหมือน):** สิ่งที่มีลักษณะเหมือนกัน (สี, รูปร่าง, texture) จะถูกจัดเป็นกลุ่มเดียวกัน
- **Common Fate (ชะตากรรมร่วม/การเคลื่อนที่):** จุดที่เคลื่อนที่ไปในทิศทางเดียวกันด้วยความเร็วเท่ากัน จะถูกมองว่าเป็นวัตถุเดียวกัน (สำคัญมากในวิดีโอ)
- **Common Region/Connectivity:** สิ่งที่อยู่ในกรอบเดียวกันหรือเชื่อมต่อกัน จะเป็นกลุ่มเดียวกัน
- **Continuity (ความต่อเนื่อง):** สมองเราชอบลากเส้นต่อจุดให้เป็นเส้นโค้งที่ต่อเนื่อง มากกว่าเส้นหักมุม
- **Symmetry (ความสมมาตร):** วัตถุที่มีความสมมาตรมักถูกมองว่าเป็นวัตถุชิ้นเดียว
- **Illusory Contours (เส้นขอบลวงตา):** สมองเราสามารถ "สร้าง" เส้นขอบขึ้นมาเองได้แม้ไม่มีเส้นจริง หากองค์ประกอบรอบข้างบ่งชี้ว่าเป็นรูปทรงนั้น (เช่น ภาพสามเหลี่ยมที่เกิดจากวงกลมแหว่ง)

ความยาก: การแบ่งส่วนภาพเป็นเรื่อง "นามธรรม" (Subjective) มาก คนสองคนอาจแบ่งส่วนภาพเดียวกันไม่เหมือนกัน ขึ้นอยู่กับว่าเขาสนใจรายละเอียดระดับไหน (ดูภาพคน หรือดูเสื้อผ้าที่คนใส่)

3. กลยุทธ์การแบ่งส่วนภาพ (Segmentation Strategies)

ในการเขียนโปรแกรม เรามีแนวคิดหลัก 2 แบบ :

1. **Top-down:** ใช้ความรู้เกี่ยวกับวัตถุ (Object knowledge) ว่า "นี่คือคน" ดังนั้นพิกเซลเหล่านี้ต้องอยู่ด้วยกัน (ยากมาก เพราะต้องรู้ก่อนว่าวัตถุคืออะไร)
2. **Bottom-up:** ใช้ข้อมูลพื้นฐานของพิกเซล (Low-level cues) เช่น สีเหมือนกัน อยู่ใกล้กัน แล้วจับกลุ่มรวมกัน (เป็นวิธีที่เราจะเรียนในยุค Classical)

4. ประเภทของงาน Segmentation ในยุคใหม่

ก่อนจะเจาะลึกเทคนิค เราต้องแยกแยะ "โจทย์" ให้ออกก่อน :

1. **Semantic Segmentation:** แยกประเภทของพิกเซล (เช่น นี่คือพิกเซล "แมว", นี่คือพิกเซล "หญ้า") แต่ถ้ามีแมว 2 ตัว มันจะระบายสีเดียวกันหมด แยกไม่ออกว่าตัวไหนเป็นตัวไหน
2. **Instance Segmentation:** แยกวัตถุเป็นชิ้นๆ (Object Detection + Segmentation) รู้ว่าเป็น "แมวตัวที่ 1" และ "แมวตัวที่ 2" แต่จะไม่สนใจพื้นหลัง (เช่น ท้องฟ้า)
3. **Panoptic Segmentation:** รวมทั้งสองอย่างเข้าด้วยกัน คือแยกทั้งวัตถุเป็นชิ้นๆ และระบายสีพื้นหลังให้ครบทุกพิกเซล

จบ Part 1 ครับ ในส่วนต่อไป เราจะลงลึกเรื่อง **Classical Methods** โดยเริ่มจากเทคนิคการจัดกลุ่มพิกเซลด้วยคณิตศาสตร์ (Clustering) ซึ่งเป็นพื้นฐานสำคัญก่อนจะมี AI

พิมพ์ **"next"** หรือ **"ถัดไป"** เพื่อเรียนต่อ Part 2 ครับ

เยี่ยมครับ เรามาต่อกันที่ **Part 2** ซึ่งเป็นจุดเริ่มต้นของการใช้ "คณิตศาสตร์" มาแก้ปัญหานี้ โดยมองว่าภาพคือข้อมูลกลุ่มใหญ่ที่ต้องจัดหมวดหมู่ครับ

Part 2: การแบ่งส่วนภาพด้วยวิธี Clustering (Classical Clustering)

ในยุค Classical Computer Vision เราไม่ได้มองภาพเป็นภาพ แต่มองพิกเซลเป็น **"จุดข้อมูลในอวกาศ (Data Points in Space)"**

1. เปลี่ยนมุมมอง: จาก Pixel สู่ Feature Vector

ปกติเราเห็นภาพเป็นตาราง (Grid) แต่ในทางคณิตศาสตร์ เราแปลงพิกเซลแต่ละจุดให้เป็น **Feature Vector**

- **แบบง่ายที่สุด:** `[R, G, B]` (เวกเตอร์ 3 มิติ) — สีเหมือนกันคือพวกเดียวกัน
- **แบบใส่ตำแหน่ง:** `[R, G, B, x, y]` (เวกเตอร์ 5 มิติ) — สีเหมือนกัน **และ** อยู่ใกล้กันคือพวกเดียวกัน

หลักการคือ **Distance = Similarity** ถ้าคำนวณระยะห่างระหว่างสองเวกเตอร์ (Euclidean Distance) แล้วได้ค่าน้อย แปลว่าพิกเซลสองจุดนั้น "เหมือนกัน" และควรอยู่กลุ่มเดียวกัน

---

2. K-Means Clustering (แบ่งกลุ่มด้วยค่าเฉลี่ย K กลุ่ม)

นี่คืออัลกอริทึมที่นิยมที่สุดและเข้าใจง่ายที่สุด

หลักการทำงาน: เรากำหนดก่อนว่าอยากได้กี่กลุ่ม (ค่า k) สมมติ $k = 3$ (อยากแบ่งภาพเป็น 3 สี)

1. **Initialization:** สุ่มจุดศูนย์กลาง (Centroids) ขึ้นมา 3 จุดใน Feature Space
2. **Assignment:** พิกเซลไหนอยู่ใกล้ Centroid สีไหนที่สุด ให้ย้ายไปอยู่แก๊งนั้น
3. **Update:** เมื่อได้สมาชิกแก๊งแล้ว ให้หา "ค่าเฉลี่ย" (Mean) ใหม่ของตำแหน่งสมาชิกทั้งหมด แล้วย้าย Centroid ไปที่จุดศูนย์กลางใหม่นั้น
4. **Loop:** ทำซ้ำข้อ 2-3 ไปเรื่อยๆ จนกว่า Centroid จะไม่ขยับ (Converge)

ผลลัพธ์: ภาพจะถูกลดทอนสีลงเหลือแค่ k สี และแบ่งพื้นที่ตามสีเหล่านั้น

ข้อดี/ข้อเสีย :

- ✅ **ดี:** ง่าย เร็ว พื้นฐานแน่น
- ❌ **แย่: ต้องเดาค่า k เอง** (เราไม่รู้หรอกว่าภาพนี้ควรมีกี่วัตถุ), อ่อนไหวต่อจุดเริ่มต้น (ถ้าสุ่มไม่ดี ผลออกมาเละ), และไม่สนใจรูปร่างที่ซับซ้อน (ชอบจัดกลุ่มเป็นก้อนกลมๆ)

---

3. Mean Shift Segmentation (ไต่เขาหาจุดหนาแน่น)

เพื่อแก้ปัญหาการที่ต้องเดาค่า k นักวิจัยจึงคิดค้นวิธี **Mean Shift**

หลักการทำงาน (Hill Climbing): จินตนาการว่า Feature Space เป็นพื้นที่ที่มีภูเขา (ความหนาแน่นของข้อมูลสูง) และหุบเหว

1. **Window:** กำหนดหน้าต่างวงกลม (Window/Kernel) ครอบลงไปบนจุดข้อมูล
2. **Shift:** คำนวณหา "จุดศูนย์ถ่วง" (Center of Mass) ของข้อมูลในวงกลมนั้น แล้วขยับวงกลมไปหาจุดศูนย์ถ่วงใหม่ (เหมือนไต่ขึ้นที่สูง)
3. **Converge:** ทำซ้ำจนกว่าจะถึง "ยอดเขา" (Mode/Peak)
4. **Grouping:** พิกเซลไหนที่ไต่ขึ้นไปจบที่ "ยอดเขาเดียวกัน" ถือว่าเป็นพวกเดียวกัน

ผลลัพธ์: คอมพิวเตอร์จะบอกเองว่ามีกี่กลุ่ม (k) ตามจำนวนยอดเขาที่เจอ

ข้อดี/ข้อเสีย :

- ✅ **ดี:** ไม่ต้องเดา k (Automatic k), ทนทานต่อสัญญาณรบกวน (Outliers), และรองรับรูปร่างข้อมูลที่ซับซ้อนได้ดีกว่า K-Means
- ❌ **แย่:** ช้ามาก (Computationally Expensive) และยังมีพารามิเตอร์ที่ต้องเลือกคือ "ขนาดหน้าต่าง" (Window Size / Bandwidth W) ถ้าเล็กไปก็แบ่งยิบย่อย ถ้าใหญ่ไปก็รวบหมด

สรุป Part 2: วิธีการแบบ Clustering ใช้สีและตำแหน่งเป็นหลัก แต่มันยัง "ตาบอด" ต่อโครงสร้าง (Structure) และขอบเขต (Boundary) ของวัตถุจริงๆ ทำให้บางครั้งวัตถุชิ้นเดียวกันแต่สีต่างกันนิดหน่อยก็ถูกแยกขาดจากกัน

ใน **Part 3** เราจะไปดูวิธีการที่ฉลาดกว่านั้น คือการมองภาพเป็น **"กราฟ" (Graph Theory)** เพื่อตัดแบ่งภาพให้เนียนกริบตามขอบวัตถุครับ

พิมพ์ **"next"** หรือ **"ถัดไป"** เพื่อลุยต่อ Part 3 ครับ

มาต่อกันที่ **Part 3** ครับ ส่วนนี้จะเป็นการยกระดับจากการมองพิกเซลเดี่ยวๆ มาเป็นการมองภาพทั้งภาพเป็น "เครือข่าย" ที่เชื่อมโยงกัน ซึ่งเป็นแนวคิดที่ทรงพลังและคลาสสิกมากครับ

Part 3: การแบ่งส่วนภาพด้วยทฤษฎีกราฟ (Graph-Based Segmentation)

ใน Part 2 (Clustering) เราดูแค่ว่า "สีใครเหมือนใคร" แต่ใน Part 3 นี้ เราจะเพิ่มมิติของ "ความสัมพันธ์" เข้าไป โดยใช้ทฤษฎีกราฟ (Graph Theory) ครับ

1. เปลี่ยนภาพให้เป็น "กราฟ" (Images as Graphs)

แทนที่เราจะมองภาพเป็นตารางสี่เหลี่ยม เราจะมองมันเป็น **กราฟ** $G = (V, E)$

- **Vertices (V - จุดยอด):** คือ **Pixel** แต่ละจุดในภาพ
- **Edges (E - เส้นเชื่อม):** คือ เส้นที่ลากเชื่อมระหว่างพิกเซล (เช่น เชื่อมกับเพื่อนบ้าน 4 ทิศ หรือ 8 ทิศ)
- **Weight (น้ำหนัก):** ความหนาของเส้นเชื่อม จะขึ้นอยู่กับ **Affinity (ความเหมือน)**
  - ถ้าพิกเซลสีเหมือนกันมาก = เส้นหนา (Weight มาก)
  - ถ้าพิกเซลสีต่างกันมาก = เส้นบาง (Weight น้อย)

สูตรคำนวณความสัมพันธ์ (Affinity) มักใช้ฟังก์ชัน Exponential:

$$w(i,j) = e^{\frac{-1}{2\sigma^2} S(f_i, f_j)}$$

แปลง่ายๆ ว่า: ยิ่งพิกเซลต่างกันมากเท่าไหร่ (S มาก) ค่า w จะยิ่งเข้าใกล้ 0 (เส้นบางจนแทบขาด)

---

2. การตัดกราฟ (Graph Cut / Min-Cut)

เป้าหมายของการแบ่งส่วนภาพในวิธีนี้คือ **"การตัดเส้นเชือก (Edges)"** เพื่อแยกกราฟก้อนใหญ่ออกเป็นกราฟย่อยๆ (Subgraphs) ที่แยกจากกันโดยสิ้นเชิง (V_A และ V_B)

หลักการคือเราต้องหาจุดตัดที่มี **"ราคาถูกที่สุด" (Minimum Cost)**

- **Cost of Cut:** ผลรวมของน้ำหนัก (Weight) ของเส้นเชือกทั้งหมดที่เราตัดขาด
- **Min-Cut:** พยายามตัดเส้นที่บางที่สุด (สีต่างกันที่สุด) เพื่อให้แยกวัตถุออกจากกันได้ง่ายที่สุด

ปัญหาใหญ่ของ Min-Cut : อัลกอริทึม Min-Cut มีนิสัย "ขี้เกียจ" ครับ สมมติมีภาพคนยืนอยู่ แทนที่มันจะตัดรอบตัวคน (ซึ่งต้องตัดเส้นเชือกจำนวนมาก) มันมักจะเลือกตัด "พิกเซลเดียว" ที่อยู่โดดๆ ตรงมุมภาพทิ้งไป เพราะการตัดพิกเซลเดียวนั้นใช้ "ต้นทุนต่ำกว่า" การลากเส้นยาวๆ ผ่ากลางภาพ ผลลัพธ์ที่ได้จึงมักจะเป็นพิกเซลเล็กๆ หลุดออกมา ไม่ใช่วัตถุที่เราต้องการ

---

3. ทางออก: Normalized Cut (NCut)

เพื่อแก้ความขี้เกียจของ Min-Cut นักวิจัย (Shi & Malik, 2000) จึงเสนอวิธี **Normalized Cut (NCut)**

แนวคิดคือ: **"อย่าดูแค่ราคาตัด แต่ต้องดูขนาดของชิ้นส่วนที่ได้ด้วย"** สูตรของ NCut จะบังคับว่า การตัดที่ดีนั้น ชิ้นส่วนทั้งสองฝั่ง (V_A และ V_B) จะต้องมี **"ความสัมพันธ์ภายในกลุ่ม (Association)"** ที่แน่นแฟ้น และมีขนาดใหญ่พอสมควร ไม่ใช่แค่ตัดเศษเล็กเศษน้อยทิ้งไป

$$NCut(V_A, V_B) = \frac{cut(V_A, V_B)}{assoc(V_A, V)} + \frac{cut(V_A, V_B)}{assoc(V_B, V)}$$

- ตัวหาร ($assoc$) คือตัวที่บังคับให้กราฟย่อยมีขนาดใหญ่
- วิธีนี้ทำให้ได้ผลลัพธ์การแบ่งส่วนภาพที่สมดุลและถูกต้องตามการรับรู้ของมนุษย์มากที่สุดในยุค Classical

ความท้าทาย: การคำนวณหา NCut ที่ดีที่สุดเป๊ะๆ เป็นปัญหาทางคณิตศาสตร์ที่ยากระดับ **NP-Complete** (ต้องใช้เวลาคำนวณนานจน

จักรวาลแตกดับถ้าข้อมูลเยอะ) แต่เรามีวิธีลัดโดยใช้ **Eigenvectors** (ค่าเจาะจง) มาช่วยประมาณคำตอบได้เร็วขึ้น

สรุป Part 3: Graph-based Segmentation ถือเป็นเทคนิคที่สวยงามทางคณิตศาสตร์มาก มันแก้ปัญหาเรื่องขอบเขตของวัตถุได้ดีกว่า Clustering แต่ข้อเสียคือคำนวณหนักมากสำหรับภาพความละเอียดสูง

ใน **Part 4** เราจะก้าวข้ามยุค Classical เข้าสู่ยุคปฏิวัติวงการ นั่นคือ **Deep Learning** ซึ่งเปลี่ยนวิธีคิดจากการคำนวณมือ เป็นการให้ AI เรียนรู้เองว่า "แมว" หน้าตาเป็นยังไง

พร้อมแล้วพิมพ์ **"next"** หรือ **"ถัดไป"** เพื่อเข้าสู่โลกของ Neural Networks ครับ

มาถึง **Part 4** ครับ นี่คือจุดเปลี่ยนที่สำคัญที่สุดในประวัติศาสตร์ Computer Vision เมื่อเราก้าวข้ามจากการคำนวณมือ (Classical) เข้าสู่ยุค **Deep Learning** ครับ

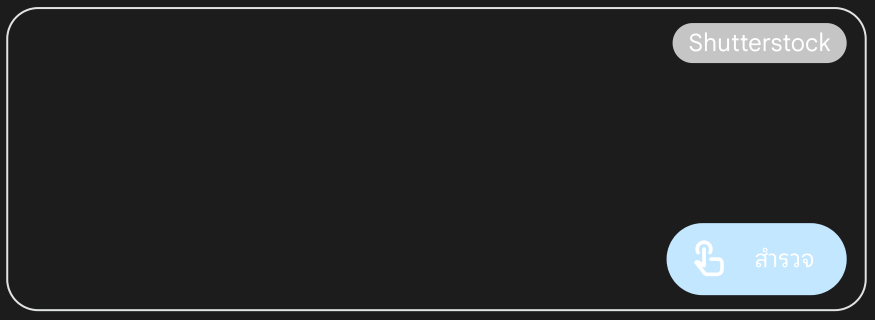
Part 4: ก้าวสู่ Deep Learning - Semantic Segmentation (FCN)

ในยุคก่อน เราพยายามเขียนสมการคณิตศาสตร์เพื่อแยกสีแยกขอบ แต่ในยุคนี้เราจะสร้างสมองกล (Neural Network) ขึ้นมา แล้วโยนรูปภาพพร้อมเฉลย (Ground Truth) ให้มันดูเป็นพันๆ รูป เพื่อให้มันเรียนรู้เองว่า "รูปร่างแบบไหนคือแมว" และ "สีแบบไหนคือหญ้า"

1. โจทย์ใหม่: Semantic Segmentation

เราไม่ได้ต้องการแค่รู้ว่า "ในภาพมีแมวไหม" (Image Classification) แต่เราต้องการรู้ว่า **"พิกเซลไหนบ้างที่เป็นส่วนหนึ่งของแมว"**

- **Input:** รูปภาพสี (RGB) ขนาด $H \times W \times 3$
- **Output:** แผนที่ประเภท (Class Map) ขนาด $H \times W$ โดยแต่ละจุดจะระบุว่าพิกเซลนั้นคืออะไร (เช่น 0=พื้นหลัง, 1=แมว, 2=ต้นไม้)



2. วิวัฒนาการจากตัวจำแนก (Classifier) สู่ตัวแบ่งส่วน (Segmenter)

- **วิธีดึกดำบรรพ์: Sliding Window (หน้าต่างเลื่อน)**
  - **แนวคิด:** ตัดภาพเป็นชิ้นเล็กๆ (Patch) แล้วส่งเข้า AI ทีละชิ้น เพื่อถามว่า "ชิ้นนี้คืออะไร" แล้วเอาคำตอบมาแปะรวมกัน
  - **ข้อเสีย:** ไม่มีประสิทธิภาพอย่างรุนแรง! เพราะพิกเซลที่อยู่ติดกันถูกคำนวณซ้ำซ้อนมหาศาล สิ้นเปลืองพลังงานคอมพิวเตอร์โดยใช่เหตุ
- **จุดเปลี่ยนสำคัญ: Fully Convolutional Networks (FCN)**
  - ในปี 2015 ทีมนักวิจัย (Long et al.) เสนอไอเดียว่า "ทำไมเราต้องตัดภาพ? ส่งภาพทั้งใบเข้าไปเลยสิ!"
  - **Fully Convolutional:** พวกเขาเปลี่ยน Layer ท้ายสุดของ AI จากที่เคยเป็น Fully Connected (ที่ต้องบีบข้อมูลเหลือบรรทัดเดียวเพื่อตอบว่ารูปนี้คืออะไร) ให้กลายเป็น **Convolutional Layer** ทั้งหมด
  - **ผลลัพธ์:** AI สามารถรับภาพขนาดเท่าไหร่ก็ได้ ($H \times W$) และพ่นคำตอบออกมาเป็นแผนที่ความน่าจะเป็น ($ScoresMap$) ขนาด $C \times H \times W$ (โดย C คือจำนวนประเภทวัตถุ) ได้ในการรันครั้งเดียว

3. ปัญหาของการทำ Convolution แบบตรงๆ

ถ้าเราใช้ Convolutional Layer ต่อกันยาวๆ โดยไม่ย่อขนาดภาพเลย จะเกิดปัญหา 2 อย่าง:

1. **กินเมมโมรี่มหาศาล:** การรักษาความละเอียดภาพเดิม ($H \times W$) ไว้ตลอดทุก Layer ต้องใช้ RAM มหาศาล

2. **Receptive Field แคบ:** AI จะมองเห็นแค่จุดเล็กๆ ไม่เห็นภาพรวม (เหมือนมองขนแมวแต่ไม่รู้ว่าเป็นแมว เพราะไม่เห็นหูหรือหาง)

4.

ทางออก: โครงสร้างแบบ Encoder-Decoder

เพื่อแก้ปัญหานี้ สถาปัตยกรรมยุคใหม่จึงใช้วิธี **"ย่อแล้วขยาย"**:

1. **Downsampling (Encoder):** ใช้ Pooling หรือ Strided Convolution เพื่อย่อภาพลงเรื่อยๆ ทำให้ AI เข้าใจ "บริบท" (Context) และลดภาระการคำนวณ
2. **Upsampling (Decoder):** นี่คือส่วนที่ยากครับ เพราะเมื่อย่อภาพไปแล้ว รายละเอียดพิกเซลจะหายไป เราต้องหาวิธี **"ขยายภาพ"** กลับมาให้เท่าเดิมเพื่อระบายสีพิกเซลให้แม่นยำ

สรุป Part 4: เราเปลี่ยนจากการตัดภาพมาใช้ **FCN** ที่ประมวลผลทั้งภาพรวดเดียว แต่ต้องแลกมาด้วยความจำเป็นที่ต้องมีการ "ย่อภาพ" เพื่อทำความเข้าใจ และ "ขยายภาพ" เพื่อสร้างผลลัพธ์

คำถามสำคัญคือ **"เราจะขยายภาพ (Upsampling) อย่างไรให้ภาพไม่แตกและข้อมูลยังอยู่ครบ?"** นี่คือหัวใจสำคัญของ Part ต่อไปครับ

พิมพ์ **"next"** หรือ **"ถัดไป"** เพื่อเจาะลึกเทคนิคการขยายภาพและโมเดลในตำนานอย่าง **U-Net** ครับ

มาต่อกันที่ **Part 5** ครับ นี่คือส่วนสำคัญทางเทคนิค เพราะการทำให้ภาพเล็กลงนั้นง่าย (แค่ตัดทิ้ง) แต่การ **"ขยายภาพ (Upsampling)"** ให้กลับมาชัดเหมือนเดิมนั้นยากกว่ามากครับ

Part 5: เทคนิคการขยายภาพและสถาปัตยกรรม U-Net (Upsampling & U-Net)

เมื่อโครงสร้าง Network มีการย่อภาพ (Downsampling) เพื่อหา Feature หลัก เราจำเป็นต้องขยายภาพกลับมา (Upsampling) เพื่อให้ได้ Output ขนาดเท่าภาพต้นฉบับ ซึ่งมีเทคนิคหลายระดับตั้งแต่แบบ "กำปั้นทุบดิน" ไปจนถึงแบบ "ให้ AI เรียนรู้เอง"

1. วิธีขยายภาพแบบพื้นฐาน (Fixed Upsampling)

กลุ่มนี้เป็นวิธีทางคณิตศาสตร์ที่ไม่ต้องมีการเรียนรู้ (No training parameters)

- **Nearest Neighbor (เพื่อนบ้านใกล้สุด):**
  - **วิธี:** ก๊อปปี้ค่าของพิกเซลเดิมไปแปะในช่องว่างใหม่ข้างๆ เลย
  - **ผลลัพธ์:** ภาพจะเป็นบล็อกสี่เหลี่ยม (Blocky) เหมือนภาพ 8-bit ยุคเก่า
- **Bed of Nails (ตะปูตอก):**
  - **วิธี:** เอาค่าพิกเซลเดิมไปวางไว้ที่มุมซ้ายบนของบล็อกใหม่ ส่วนช่องอื่นที่งอกขึ้นมาให้เติม 0 ทั้งหมด
  - **ผลลัพธ์:** ข้อมูลกระจายตัวห่างกัน
- **Max Unpooling (คืนค่าสู่จุดเดิม):**
  - **วิธี:** ต้องใช้คู่กับ Max Pooling ตอนขาลง (Downsampling) โดยระบบจะ "จำตำแหน่ง" (Indices) ว่าค่า Max มาจากพิกเซลไหน ตอนขยายกลับก็นำค่านั้นไปวางคืนที่ตำแหน่งเดิมเป๊ะๆ
  - **ข้อดี:** รักษาตำแหน่งของขอบวัตถุได้แม่นยำกว่าวิธีอื่น

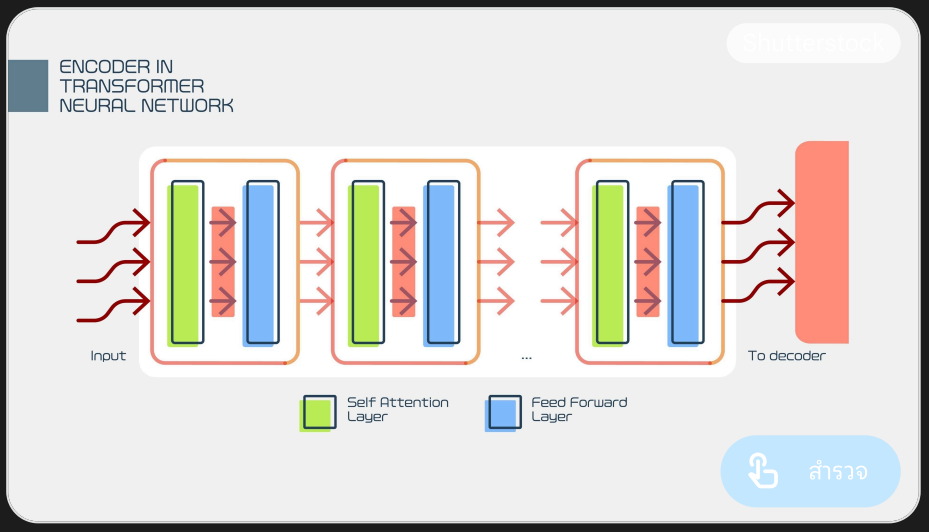
2. วิธีขยายภาพแบบเรียนรู้ได้ (Learnable Upsampling)

วิธีนี้คือหัวใจของ Deep Learning คือให้ AI เรียนรู้เองว่าควรจะขยายภาพอย่างไรให้เนียนที่สุด

- **Transposed Convolution (หรือเรียกว่า Deconvolution / Upconvolution)**
  - **แนวคิด:** ปกติ Convolution จะยุบหลายพิกเซลรวมเป็นหนึ่ง (Many-to-One) แต่ Transposed Convolution จะกระจายหนึ่งพิกเซลออกไปเป็นหลายพิกเซล (One-to-Many)
  - **การทำงาน:** AI จะมี Filter (ตัวคูณ) ที่เรียนรู้ได้ มันจะเอาค่า Input ไปคูณกับ Filter แล้ว "แปะ" ผลลัพธ์ลงไปใน Output ที่มีขนาดใหญ่ขึ้น บริเวณที่ Filter ทับซ้อนกัน (Overlap) ก็จะเอาค่ามาบวกกัน
  - **ผลลัพธ์:** ได้ภาพขยายที่รายละเอียดดีกว่าการใช้สูตรคณิตศาสตร์ตายตัว

3. สถาปัตยกรรม U-Net (The Legend of Segmentation)

นี่คือโมเดลที่โด่งดังที่สุดในวงการ Segmentation (เดิมทีออกแบบมาเพื่องานการแพทย์)



ความเจ๋งของ U-Net:

1. **โครงสร้างรูปตัว U:**
    - **ขาลง (Contractive Path / Encoder):** ฝั่งซ้ายทำหน้าที่ย่อภาพเพื่อดึง "เนื้อหา" (What) ว่าคือวัตถุอะไร
    - **ขาขึ้น (Expansive Path / Decoder):** ฝั่งขวาทำหน้าที่ขยายภาพเพื่อระบุ "ตำแหน่ง" (Where) ว่าวัตถุอยู่ตรงไหน
2. **Skip Connections (ทางลัดสีเทา):** นี่คือทีเด็ดครับ!
    - ปัญหาของ Encoder คือพอย่อภาพมากๆ "รายละเอียดตำแหน่ง" จะหายไป (Spatial Loss)
    - U-Net แก้โดยการลากเส้นเชื่อม (Copy and Crop) นำ Feature Map ความละเอียดสูงจากฝั่งซ้าย (Encoder) มา **"แปะรวม" (Concatenate)** กับฝั่งขวา (Decoder) ในชั้นเดียวกัน
    - ทำให้ตอนขยายภาพ AI จะได้ทั้ง "ข้อมูลตำแหน่งที่แม่นยำ" จากฝั่งซ้าย และ "ข้อมูลประเภทวัตถุ" จากฝั่งขวามาประมวลผลร่วมกัน

สรุป Part 5: เรามีเครื่องมือครบแล้ว ทั้งการย่อภาพเพื่อเข้าใจ และการขยายภาพด้วย **Transposed Convolution** ร่วมกับ **Skip**

Connections ใน U-Net เพื่อสร้างภาพผลลัพธ์ที่คมชัด

ใน **Part 6 (ตอนจบ)** เราจะไปดูเทคนิคขั้นเทพที่สุดในปัจจุบัน คือการแยกวัตถุเป็นชิ้นๆ ไม่ใช่แค่แยกสี (Instance Segmentation) ด้วยโมเดล **Mask R-CNN** ครับ

พิมพ์ **"next"** หรือ **"ถัดไป"** เพื่อไปสู่บทสรุปครับ

มาถึง **Part 6 (บทสรุป)** ครับ นี่คือจุดสูงสุดของเทคโนโลยีในเอกสารชุดนี้ ซึ่งรวมเอาทุกอย่างที่เราเรียนมา (Classification + Detection + Segmentation) เข้าด้วยกันเป็นโมเดลเดียวที่ "มองเห็น" ได้ละเอียดเหมือนมนุษย์ครับ

Part 6: Instance & Panoptic Segmentation (Mask R-CNN & Beyond)

ใน Part 4 (Semantic Segmentation) เราแยกพิกเซล "แมว" ออกจาก "หญ้า" ได้แล้ว แต่ปัญหาคือถ้ามีแมว 2 ตัวยืนซ้อนกัน AI จะมองเห็นเป็น "ก้อนแมว" ก้อนใหญ่ก้อนเดียว ไม่สามารถนับจำนวนแมวได้

1. แยกประเภทวัตถุ: Things vs. Stuff

ก่อนจะเข้าใจโมเดล เราต้องเข้าใจศัพท์เทคนิค 2 คำนี้ก่อนครับ :

- **Things (สิ่งของ):** คือวัตถุที่ **"นับจำนวนได้"** และมีรูปทรงชัดเจน เช่น แมว, คน, รถยนต์, แก้วน้ำ (นี่คือเป้าหมายของ Instance Segmentation)
- **Stuff (สสาร/พื้นหลัง):** คือสิ่งที่ **"นับไม่ได้"** มีรูปร่างไม่แน่นอน เช่น ท้องฟ้า, หญ้า, ถนน, น้ำ (นี่คือเป้าหมายของ Semantic Segmentation)

---

2. Instance Segmentation (แยกชิ้นวัตถุ)

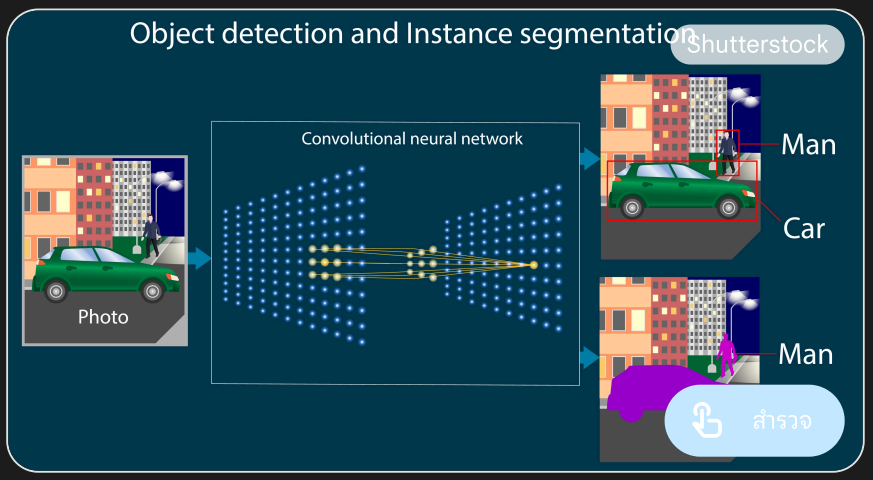
โจทย์คือ: "ตรวจจับวัตถุทุกชิ้น (Object Detection) และระบายสีพิกเซลของวัตถุชิ้นนั้น (Segmentation)"

พระเอกของงานนี้: Mask R-CNN Mask R-CNN พัฒนาต่อยอดมาจาก Faster R-CNN (ซึ่งเป็นโมเดลตรวจจับวัตถุยอดฮิต) โดยเพิ่ม "กิ่ง"

(Branch) ใหม่เข้าไปเพื่อทำหน้าที่วาดหน้ากาก (Mask) โดยเฉพาะ

โครงสร้างการทำงาน (Architecture):

1. **Backbone (CNN):** รับภาพเข้ามาแล้วสกัด Feature (ใช้ ResNet หรือ FPN)
2. **RPN (Region Proposal Network):** สแกนหาพื้นที่ที่ "น่าจะมีวัตถุอยู่" (Proposals) แล้วส่งกรอบสี่เหลี่ยมคร่าวๆ ไปต่อ
3. **RoI Align (พระเอกตัวจริง):**
   - ในรุ่นเก่า (Faster R-CNN) ใช้ *RoI Pooling* ซึ่งจะปัดเศษทศนิยมของพิกเซลให้เป็นจำนวนเต็ม ทำให้ตำแหน่งหน้ากากคลาดเคลื่อน (Misalignment)
   - **Mask R-CNN** เปลี่ยนมาใช้ **RoI Align** ซึ่งใช้คณิตศาสตร์ (Bilinear Interpolation) คำนวณค่าพิกเซลในจุดทศนิยมได้แม่นยำ ทำให้ตัดขอบวัตถุได้คมกริบ
4. **Heads (ส่วนหัว):** แยกการทำงานเป็น 3 ทางขนานกัน :
   - **Classification:** ตอบว่าคือ "แมว"
   - **BBox Regression:** ตีกรอบสี่เหลี่ยมรอบตัวแมวให้แม่นขึ้น
   - **Mask Prediction:** พ่น Binary Mask (ขาว-ดำ) ขนาด 28×28 ออกมาเพื่อระบายสีตัวแมว



---

3. Panoptic Segmentation (การมองเห็นแบบสมบูรณ์)

นี่คือเทรนด์ใหม่ล่าสุด (CVPR 2019) ที่รวมเอา Semantic (เพื่อดูพื้นหลัง) และ Instance (เพื่อดูวัตถุ) มารวมกัน

- **นิยาม:** ระบุป้ายกำกับ (Label) ให้กับ **"ทุกพิกเซล"** ในภาพ
    - ถ้าเป็น **Stuff** (ฟ้า, หญ้า): ระบายสีตามประเภทเฉยๆ ไม่ต้องแยกชิ้น
    - ถ้าเป็น **Things** (วัว): ระบายสีแยกชิ้นเป็น "วัวตัวที่ 1", "วัวตัวที่ 2"

ผลลัพธ์ที่ได้: ภาพจะถูกระบายสีจนเต็มพื้นที่ ไม่มีช่องว่าง (No gaps) และเรารู้จักทุกอณูของภาพว่าคืออะไรและเป็นของใคร (ดูภาพประกอบวัวใน Slide หน้า 95 จะเห็นชัดเจนว่าวัวแต่ละตัวสีต่างกัน และพื้นหลังก็ถูกระบุประเภทด้วย)

บทสรุปของหลักสูตร (Conclusion)

เราได้เดินทางผ่านวิวัฒนาการของการแบ่งส่วนภาพ (Image Segmentation) มาไกลมาก:

1. **ยุค Classical:** ใช้คณิตศาสตร์เรื่องสี (Clustering) และกราฟ (Graph Cut) ซึ่งเก่งแต่ต้องอาศัยฝีมือคนปรับแต่งเยอะ
2. **ยุค Deep Learning:**
    - **FCN & U-Net:** ใช้ CNN เรียนรู้ที่จะระบายสีพิกเซล (Semantic)
    - **Mask R-CNN:** เพิ่มความฉลาดในการนับจำนวนวัตถุและวาดหน้ากากแยกชิ้น (Instance)
    - **Panoptic FPN:** รวมทุกศาสตร์เข้าด้วยกันเพื่อความเข้าใจภาพระดับสูงสุด

นี่คือพื้นฐานทั้งหมดที่คุณต้องรู้เพื่อที่จะต่อยอดไปสู่โมเดลยุคปัจจุบันอย่าง **SAM (Segment Anything Model)** ที่อยู่ในหน้าสุดท้ายของแผนผัง ซึ่งสามารถตัดต่ออะไรก็ได้โดยไม่ต้องสอนเพิ่มครับ!

จบหลักสูตร Image Segmentation Deep Dive ครับ! หากคุณมีคำถามเจาะจงจุดไหน หรืออยากให้ผมอธิบายโค้ดของส่วนไหนเพิ่มเติม บอกได้เลยนะครับ ยินดีรับใช้เสมอครับ!