รายงานวิจัยฉบับสมบูรณ์

โครงการวิจัย

เรื่อง

การรู้จำภาพอาหารไทยด้วยโครงข่ายประสาทเทียมแบบลึกบนสมาร์ทโฟน

Thai Food Image Recognition using Deep neural networks on Smartphone

โดย

ผศ.ดร.สุรเชษฐ์ กานต์ประชา

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ธันวาคม พ.ศ. 2560

สัญญาเลขที่ R2560C142

รายงานวิจัยฉบับสมบูรณ์

โครงการวิจัย

เรื่อง

การรู้จำภาพอาหารไทยด้วยโครงข่ายประสาทเทียมแบบลึกบนสมาร์ทโฟน

Thai Food Image Recognition using Deep neural networks on Smartphone

โดย

ผศ.ดร.สุรเชษฐ์ กานต์ประชา

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ธันวาคม พ.ศ. 2560

สนับสนุนโดยกองทุนวิจัยมหาวิทยาลัยนเรศวร

บทคัดย่อ

งานวิจัยชิ้นนี้นำเสนอการออกแบบและพัฒนาแบบจำลองของโครงข่ายประสาทเทียมแบบลึก โดยมีจุดมุ่งหมายเพื่ออำนวยความสะดวกให้แก่นักท่องเที่ยวชาวต่างชาติที่ต้องการใช้งานแอพพลิเคชั่นสำหรับรู้จำภาพถ่ายอาหารไทยบนอุปกรณ์สมาร์ทโฟน โดยไม่จำเป็นต้องเชื่อมต่อเครือข่ายอินเตอร์เน็ต สำหรับการออกแบบของแบบจำลอง NU-InNet (Naresuan University Inception Network) ได้นำแนวคิด Inception module ของแบบจำลอง GoogLeNet ที่มีประสิทธิภาพด้านความถูกต้องสูง มาปรับเปลี่ยนจากลำดับชั้น Convolutional ที่มีขนาดมากกว่า 3x3 ให้เป็น double 3x3 แทน 5x5 และ triple 3x3 แทน 7x7 โดยการออกแบบแบบจำลองมุ่งเน้นไปที่การลดเวลาในการประมวลผลและขนาดของแอพพลิเคชั่นให้เหมาะสมสำหรับการนำไปใช้งานบนสมาร์ทโฟน สำหรับชุดข้อมูลที่ใช้ในการฝึกฝนได้รวบรวมและจัดเก็บภาพถ่ายอาหารไทยที่ได้รับความนิยม 50 ประเภท (THFOOD-50) จำนวน 15,770 ภาพ มาฝึกฝนให้กับโครงข่ายประสาทเทียมแบบลึก จากการทดลองพบว่าแบบจำลอง NU-InNet มีประสิทธิภาพด้านความถูกต้องเพิ่มขึ้น 4.64% เวลาในการประมวลผลบนคอมพิวเตอร์ลดลง 1.9 เท่า และขนาดของแอพพลิเคชั่นลดลง 27 เท่าจากแบบจำลอง GoogLeNet และหากเพิ่มจำนวนความลึกของแบบจำลองให้มากขึ้น พบว่าแบบจำลอง NU-InNet ความลึกที่ 4 มีประสิทธิภาพด้านความถูกต้องเพิ่มขึ้น 8.07% เวลาในการประมวลผลบนคอมพิวเตอร์ลดลง 1.5 เท่า และขนาดของแอพพลิเคชั่นลดลง 10.9 เท่า เมื่อเปรียบเทียบกับแบบจำลอง BN-Inception (แบบจำลอง GoogLeNet ที่ใช้ลำดับชั้น Batch Normalization) นอกจากนี้แบบจำลอง NU-InNet ความลึกที่ 1 และ 4 ใช้เวลาในการเปิดแอพพลิเคชั่นและเวลารู้จำภาพถ่ายบนอุปกรณ์สมาร์ทโฟน เพียง 0.68 และ 0.81 วินาทีต่อการประมวลผล 1 ภาพ

Abstract

In this research, the design and development of a model of deep neural networks is proposed. The purpose of this model is to help foreign travelers in Thailand to use Thai food image recognition applications in a smartphone without connecting the internet. For the design phase, a new network called NU-InNet (Naresuan University Inception Network) adopted the concept of the Inception module from GoogLeNet which is utilized and further improved in our model. NU-Inception replaces any 5×5 convolutional layer by two 3×3 convolutional layers and any 7×7 convolutional layer by three 3×3 convolutional layers to develop the model. The processing time and model size are reduced to be able to be used as a smartphone application. NU-InNet has been applied to and tested with the Thai food database called THFOOD-50, which contains 15,770 images of 50 famous Thai dishes. Our test showed that the processing time of NU-InNet is 1.9 times lower than GoogLeNet, and the size of the parameters of NU-InNet is 27 times less. Importantly, the accuracy of NU-InNet is higher than that of GoogLeNet by 4.64%. When adding more modules to the network, the accuracy of NU-InNet with a depth of 4 is 8.07% superior to that of the BN-Inception network. These networks improved the Batch Normalization layer of GoogLeNet and also the processing time and the parameter size from NU-InNet, which, with a depth of 4, are approximately 1.5 and 10.9 times lower than those of the BN-Inception network. Useability and convenience are shown by the NU-InNet being ready for use with 0.68 and 0.81 second start-up times for both depth sizes 1 and 4.

กิตติกรรมประกาศ

โครงการวิจัยชิ้นนี้ สำเร็จลุล่วงได้ด้วยการสนับสนุนจากทุนอุดหนุนการวิจัย มหาวิทยาลัยนเรศวร ประจำปี 2560 ซึ่งทางผู้วิจัยขอขอบคุณมา ณ ที่นี้

นอกจากนี้ทางผู้วิจัยขอขอบคุณ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร ที่ได้ให้การสนับสนุนทางด้านสถานที่และอุปกรณ์บางส่วนในการทำงานวิจัยชิ้นนี้ให้สำเร็จลุล่วงด้วยดี

ผู้วิจัย

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

บทที่ 1
บทนำ

ความเป็นมาของปัญหา

สมาร์ทโฟนจัดได้ว่าเป็นอุปกรณ์พกพาที่ได้รับความนิยมอย่างมาก ในยุคที่ผู้คนติดต่อสื่อสารผ่านเครือข่ายอินเทอร์เน็ตบนแอพพลิเคชั่นประเภท Social network ที่สามารถเขียนข้อความหรืออัพโหลดรูปภาพได้อย่างเสรี ทำให้ผู้คนสามารถติดต่อสื่อสารหรือติดตามข่าวสารผ่านอินเทอร์เน็ตได้ทุกช่วงเวลา และสำหรับกรณีนักท่องเที่ยวที่เดินทางไปยังต่างประเทศ การใช้งานอินเทอร์เน็ตขณะอยู่ต่างประเทศ อาจจะทำให้ไม่ได้รับความสะดวก เนื่องจากอาจมีค่าบริการอินเทอร์เน็ตโรมมิ่งที่สูง และพื้นที่ให้บริการอินเทอร์เน็ต Wi-Fi อาจยังไม่ครอบคลุมในบางสถานที่ทำให้นักท่องเที่ยวไม่สามารถใช้งานอินเทอร์เน็ตเพื่อค้นหาข้อมูลที่ต้องการและพลาดการสื่อสารได้

แอพพลิเคชั่นที่มีความจำเป็นสำหรับนักท่องเที่ยวชาวต่างชาติ ได้แก่ แอพพลิเคชั่นที่ใช้ค้นหาสถานที่หรือร้านอาหาร แอพพลิเคชั่นที่ใช้ค้นหาข้อมูลการเดินทาง แอพพลิเคชั่นที่ใช้รู้จำสถานที่สำคัญ (Landmark recognition) และแอพพลิเคชั่นที่ใช้รู้จำอาหาร (Food recognition) จากภาพถ่ายเพื่อระบุชื่อของอาหาร ซึ่งชื่ออาหารถือว่าจำเป็นอย่างมากสำหรับนักท่องเที่ยวนอกจากใช้ในการสั่งอาหารแล้ว ยังสามารถช่วยให้ค้นหาข้อมูลของส่วนประกอบ ปริมาณแคลอรี่ และร้านค้าที่ใกล้เคียง

แอพพลิเคชั่นบนสมาร์ทโฟนที่ใช้ในการรู้จำภาพ (Image Recognition) ส่วนใหญ่ยังต้องอาศัยคอมพิวเตอร์เป็นหลัก เนื่องจากกระบวนการในการประมวลผลต้องใช้ทรัพยากรที่เพียงพอกับปริมาณของฐานข้อมูล หากฐานข้อมูลมีจำนวนมากสมาร์ทโฟนที่มีทรัพยากรจำกัดไม่สามารถทำงานได้ จึงต้องส่งข้อมูลไปที่คอมพิวเตอร์เพื่อประมวลผลแทน (Third Party Processing) ส่งผลให้ประสิทธิภาพ (Efficiency) ในการประมวลผลขึ้นอยู่กับคอมพิวเตอร์และความเร็วของอินเทอร์เน็ต

ปัจจัยสำคัญที่ส่งผลกับความถูกต้องในการระบุชื่อของอาหาร แบ่งออกเป็นสองส่วนคือ 1) จำนวนของภาพในฐานข้อมูลที่ครอบคลุมทั้งรูปร่าง ขนาด สี และมุมมองของภาพอาหารในแต่ละประเภท 2) การแทนข้อมูลเพื่อสร้างแบบจำลอง (Model) สำหรับการเรียนรู้จากข้อมูลจำนวนมาก โดยปัจจัยทั้งสองสามารถแก้ไขโดยใช้โครงข่ายประสาทเทียมแบบลึก ที่สามารถวิเคราะห์องค์ประกอบของภาพได้อย่างแม่นยำ ด้วยวิธีการแบ่งภาพเป็นส่วนเล็กๆ แล้วแทนด้วยตัวเลขเพื่อจัดเก็บและตีความหมาย ในลักษณะเดียวกับกระบวนการจำของมนุษย์ ซึ่งทำให้แบบจำลองที่ถูกสร้างขึ้นมีขนาดเล็กพอที่จะสามารถถูกจัดเก็บบนสมาร์ทโฟนเพื่อใช้ในการระบุชื่อของอาหารได้

ดังนั้นหากนำความรู้ด้านโครงข่ายประสาทเทียมมาใช้ร่วมกับการรู้จำชื่อของอาหารไทยจะช่วยเพิ่มความสะดวกให้กับนักท่องเที่ยว เนื่องจากสามารถระบุชื่อของอาหารไทยได้โดยไม่จำเป็นต้องมีการเชื่อมต่ออินเทอร์เน็ต ด้วยเหตุนี้ จึงได้มีการพัฒนาแอพพลิเคชั่นบนสมาร์ทโฟนสำหรับการรู้จำภาพอาหารไทยที่มีฐานข้อมูลจำนวนมากด้วยโครงข่ายประสาทเทียมแบบลึก เพื่อช่วยนักท่องเที่ยวในการจดจำชื่อของอาหารไทยจากภาพถ่ายได้มากยิ่งขึ้น

จุดมุ่งหมายของการศึกษา

1. เพื่อศึกษาหลักการและทฤษฎีของโครงข่ายประสาทเทียมแบบลึก
2. เพื่อประยุกต์โครงข่ายประสาทเทียมแบบลึกมาใช้กับรู้จำภาพอาหารไทย
3. เพื่อเพิ่มความถูกต้องในการระบุชื่อของอาหารจากแบบจำลองเดิม

ขอบเขตของงานวิจัย

1. ภาพถ่ายอาหารไทยที่ได้รับความนิยม 50 ประเภท ประเภทละประมาณ 200 - 700 ภาพ
2. สมาร์ทโฟนที่นำมาพัฒนาต้องทำงานบนระบบปฏิบัติการแอนดรอยด์รุ่น 5.0 ขึ้นไป

นิยามศัพท์เฉพาะ

1. สมาร์ทโฟน (Smartphone) หมายถึง โทรศัพท์เคลื่อนที่แบบพกพาที่มีความสามารถเช่นเดียวกับคอมพิวเตอร์
2. โซเชียลเน็ตเวิร์ก (Social network) หมายถึง บริการที่ใช้เชื่อมโยงระหว่างผู้คนให้สามารถติดต่อสื่อสารกันผ่านเครือข่ายอินเทอร์เน็ต
3. การรู้จำภาพ (Image recognition) หมายถึง วิธีการเรียนรู้ของเครื่องที่สามารถจดจำและระบุชื่อของภาพถ่ายได้ จากข้อมูลที่ถูกฝึกสอนโดยมนุษย์
4. การรู้จำอาหาร (Food recognition) หมายถึง วิธีการเรียนรู้ของเครื่องที่สามารถจดจำและระบุชื่อของภาพถ่ายได้ โดยใช้ข้อมูลภาพอาหารในการฝึกสอน
5. แบบจำลอง (Model) หมายถึง การแทนข้อมูลสำหรับการเรียนรู้จากข้อมูลจำนวนมาก
6. โครงข่ายประสาทเทียมแบบลึก (Deep neural network) หมายถึง อัลกอริทึมที่จำลองการทำงานเช่นเดียวกับสมองของมนุษย์ โดยสามารถเรียนรู้ด้วยการวิเคราะห์ข้อมูลในเชิงลึก

สมมติฐานของการวิจัย

ภาพถ่ายที่นำมาใช้ต้องมีอาหารปรากฏในภาพ 1 ประเภท โดยสามารถมองเห็นอาหารได้มากกว่า 70 % ของภาพทั้งหมด และภาพถ่ายสามารถมีองค์ประกอบที่ปรากฏในภาพได้ เช่น ช้อน ส้อม ภาชนะ และพื้นหลัง เป็นต้น โดยไม่ต้องตัดหรือนำบางส่วนออกจากภาพ

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

งานวิจัยชิ้นนี้ศึกษาเกี่ยวกับการรู้จำภาพอาหารไทยจำนวนมากด้วยโครงข่ายประสาทเทียมแบบลึกบนสมาร์ทโฟน ซึ่งในบทนี้กล่าวถึงหลักการและทฤษฎีต่าง ๆ ที่เกี่ยวข้องประกอบไปด้วยภาพดิจิทัล แบบจำลองสี RGB แบบจำลองสี HSV การเรียนรู้ของเครื่อง (Machine learning) การจำแนกภาพ (Image Classification) และโครงข่ายประสาทเทียมแบบลึก (Deep Neural Networks) เพื่อนำมาวิเคราะห์และตัดสินใจระบุชื่อของอาหารไทยจากภาพถ่าย

ภาพดิจิทัล (Digital Image)

ภาพดิจิทัล [1, 2] คือการแสดงข้อมูลของภาพในรูปแบบ Spatial domain ด้วย $f(x,y)$ เมื่อ x และ y คือค่าพิกัดในระนาบ และแอมพลิจูดของ f สำหรับ (x,y) ใด ๆ คือค่าความเข้มแสงของภาพ (Intensity) โดยกำหนดให้ภาพของ $f(x,y)$ มีขนาด M แถว และ N คอลัมน์ และพิกัด $(x,y)=(0,0)$ คือตำแหน่งเริ่มต้นของภาพ ดังสมการ 2.1 ดังนั้นขนาดของ M และ N ขึ้นอยู่กับความละเอียด (Resolution) ของภาพ ซึ่งภาพดิจิทัลแบ่งออกเป็น 2 ประเภท ได้แก่

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \tag{2.1}$$

1. ภาพเวกเตอร์ (Vector) หมายถึงภาพเสมือนจริงที่สร้างจากคอมพิวเตอร์โดยใช้หลักของเวกเตอร์ในคณิตศาสตร์ เพื่อกำหนดทิศทางและสเกลของตำแหน่งในระนาบ x, y สำหรับสร้างเส้นตรง เส้นโค้ง วงกลม หรือสี่เหลี่ยม มาประกอบให้เป็นรูปทรงต่าง ๆ ทำให้ภาพที่ได้มีคุณภาพที่ไม่มีการเปลี่ยนแปลงหากย่อหรือขยายภาพ

2. ภาพแรสเตอร์ (Raster) หมายถึงภาพที่ประกอบไปด้วยจุดภาพ (Pixel) รวมกันเป็นภาพ โดยจุดภาพถูกกำหนดตำแหน่งไว้ ไม่สามารถเปลี่ยนแปลงได้ หากขยายภาพประเภทนี้จะพบภาพเป็นกล่องสี่เหลี่ยมต่อกัน นั่นคือจุดภาพ โดยแต่ละจุดภาพมีการเก็บค่าสีที่เจาะจงไว้ในแต่ละตำแหน่ง ตามความละเอียดของภาพ

เมื่อพิจารณาความแตกต่างระหว่างภาพเวกเตอร์กับภาพแรสเตอร์ พบว่าภาพเวกเตอร์มีความสามารถในการย่อหรือขยายภาพโดยที่ตำแหน่งของภาพไม่มีการเปลี่ยนแปลง จึงทำให้ได้รับ

ความนิยมในงานพิมพ์และถูกนำไปใช้ในการสร้างชุดแบบอักษร (Font) บนคอมพิวเตอร์ และภาพแรสเตอร์มีข้อดีคือ สามารถสร้างสี ระบายสี หรือตกแต่งได้อย่างสวยงาม แต่มีข้อจำกัดในเรื่องของจุดภาพมีจำนวนคงที่ เมื่อนำภาพมาขยายจึงทำให้ความละเอียดของภาพลดลง จึงเหมาะสำหรับใช้ในโปรแกรมตกแต่งภาพที่แสดงผลบนหน้าจอคอมพิวเตอร์

แบบจำลองสี (Color Model)

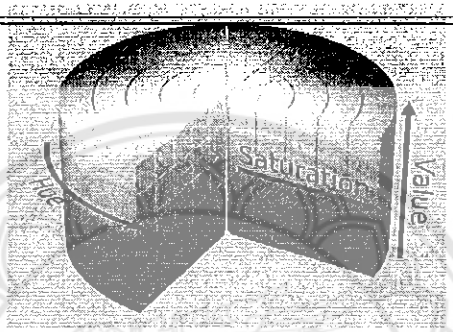
แบบจำลองสี [3-5] หมายถึงการกำหนดค่าพิกัดของสี ตามในรูปแบบที่แตกต่างกัน ซึ่งโดยทั่วไปการกำหนดพิกัดของสีจะอิงตามสีในธรรมชาติและสีที่ถูกสร้างขึ้น เช่น แบบจำลองสี RGB เป็นแบบจำลองที่สร้างตามหลักการแสดงสีของจอภาพ แบบจำลองสี HSV เป็นแบบจำลองที่สร้างตามหลักการมองเห็นสีของมนุษย์ และแบบจำลองสี CMYK เป็นแบบจำลองที่สร้างขึ้นสำหรับใช้ในเครื่องพิมพ์ เป็นต้น

แบบจำลองสี RGB เป็นแบบจำลองที่ได้รับความนิยมในการนำมาใช้สำหรับแสดงสีบนจอภาพของคอมพิวเตอร์ สมาร์ทโฟน และจอภาพ LED เนื่องจากแบบจำลอง RGB เกิดจากการนำแสงที่มนุษย์สามารถมองเห็นได้ในช่วงความยาวคลื่นประมาณ 400 - 800 นาโนเมตร โดยนำแม่สีของแสงได้แก่ แสงสีแดง (R) แสงสีเขียว (G) และแสงสีน้ำเงิน (B) ที่มีค่าระหว่าง 0 ถึง 255 มารวมกันเพื่อสร้างสีต่างๆ ได้ ดังภาพ 1 (ซ้าย) และแบบจำลองสี RGB แสดงได้ดังภาพ 1 (ขวา) ซึ่งสามารถแสดงสีได้สูงสุดถึง 16.7 ล้านสี ซึ่งใกล้เคียงกับสีที่มนุษย์มองเห็นได้

ภาพ 1 การรวมแสงของแบบจำลองสี RGB (ซ้าย) แบบจำลองสี RGB (ขวา)

ที่มา: https://en.wikipedia.org/wiki/RGB_color_model

แบบจำลองสี HSV [6] เป็นแบบจำลองที่สร้างตามหลักการมองเห็นสีของมนุษย์ ที่ประกอบไปด้วย ค่าสีที่สะท้อนออกมาจากวัตถุเข้าตามนุษย์ (Hue, H) เช่น แดง เขียว น้ำเงิน เหลือง ฟ้า หรือม่วง เป็นต้น สำหรับค่าความอิ่มตัวของค่าสี (Saturation, S) เช่น สีเลือดหมู (Crimson) มีค่าความอิ่มตัวมากว่าสีแดง (Red) และสุดท้ายระดับความสว่างของสี (Value, V) เมื่อมีค่าต่ำสุดหมายถึงสีดำและค่าสูงสุดคือสีขาว ดังภาพ 2



ภาพ 2 แบบจำลองสี HSV

ที่มา: https://en.wikipedia.org/wiki/HSL_and_HSV

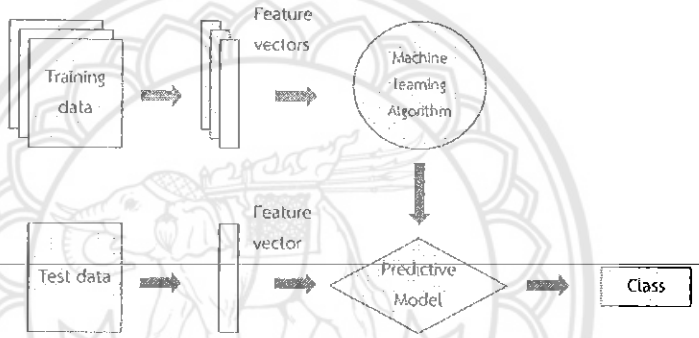
แบบจำลองสี HSV สามารถให้ความหมายของสีตามวัตถุหรือสิ่งของที่มองเห็นได้ เช่น หากพิจารณาเฉพาะค่าสีเขียว พบว่าสีเขียวสามารถมีได้ทั้ง สีเขียวอ่อน (Lime) สีเขียวสว่าง (Lightgreen) สีเขียวน้ำทะเล (Seagreen) สีเขียวป่า (Forestgreen) สีเขียวมะกอก (Olive) และสีเขียวเข้ม (Darkgreen) เป็นต้น ซึ่งทุกสีที่กล่าวมาล้วนมีค่าสี (Hue) เดียวกันทั้งหมด โดยที่มีระดับความสว่างและค่าความอิ่มตัวแตกต่างกัน ดังนั้นแบบจำลองสี HSV จึงให้สีใกล้เคียงกับสีที่มนุษย์มองเห็นได้มากกว่า แบบจำลองสี RGB

การเรียนรู้ของเครื่อง (Machine learning)

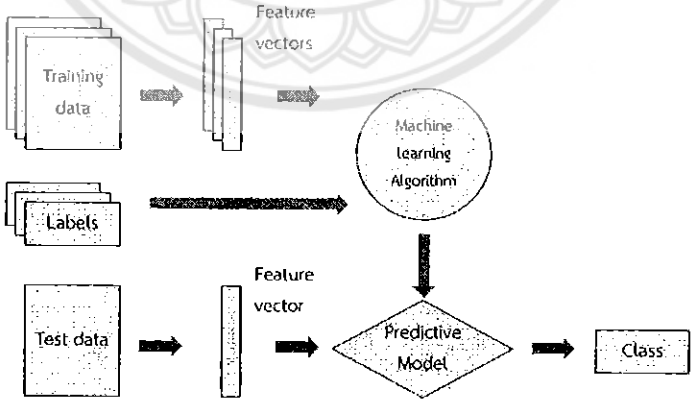
การเรียนรู้ของเครื่อง [7] คือ ศาสตร์ที่เกี่ยวข้องกับการศึกษาและสร้างขั้นตอนวิธี (Algorithm) ให้คอมพิวเตอร์สามารถเรียนรู้ตามแบบจำลองที่สร้างขึ้นจากข้อมูลตัวอย่างบางส่วนหรือเรียกว่าชุดข้อมูลฝึก (Training data) เพื่อให้คอมพิวเตอร์ศึกษา วิเคราะห์จนสามารถนำไปใช้สำหรับพยากรณ์ข้อมูลได้ด้วยตนเอง ซึ่งเรียกลักษณะการเรียนรู้นี้ว่า การเรียนรู้แบบอุปนัย (Inductive learning) [8] อีกทั้งการเรียนรู้ของเครื่องยังได้ถูกพัฒนาให้มีการเรียนรู้ในรูปแบบเดียวกับมนุษย์ที่สามารถเรียนรู้ได้จากผู้สอน ประสบการณ์ การจดจำ และการฝึกฝน ซึ่งสามารถประเมินคุณภาพของ

การเรียนรู้ได้จากการนำแบบจำลองที่สร้างจากชุดข้อมูลฝึกมาตรวจสอบด้วยชุดข้อมูลทดสอบ (Test data) โดยชุดข้อมูลฝึกกับทดสอบเป็นคนละชุด แต่ละชุดอยู่กลุ่มเดียวกัน และการประเมินประสิทธิภาพของความถูกต้อง (Accuracy) จำเป็นต้องมีฉลากหรือเฉลย (Label) กำกับไว้ในชุดข้อมูลทดสอบ เพื่อนำมาใช้ในการตรวจสอบความถูกต้องของแบบจำลองที่สร้างขึ้น

การเรียนรู้ของเครื่องสามารถแบ่งประเภทตามการติดฉลากให้กับชุดข้อมูลฝึก ซึ่งแบ่งได้ 2 ประเภท คือ การเรียนรู้แบบไม่มีผู้สอน (Unsupervised learning) หมายถึง การไม่ติดฉลากให้กับชุดข้อมูลฝึก เพื่อให้คอมพิวเตอร์เรียนรู้ด้วยตนเอง ดังภาพ 3 และการเรียนรู้แบบมีผู้สอน (Supervised learning) หมายถึง การติดฉลากหรือเฉลยให้กับชุดข้อมูลฝึก เพื่อให้คอมพิวเตอร์สามารถวิเคราะห์และเชื่อมโยงชุดข้อมูลได้ ดังภาพ 4



ภาพ 3 ภาพรวมของการเรียนรู้แบบไม่มีผู้สอน (Unsupervised learning) [9]

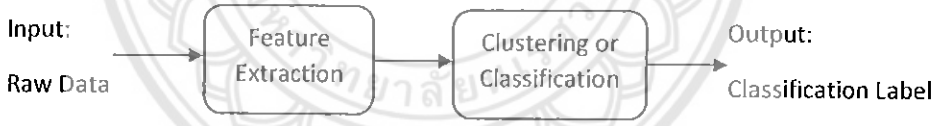


ภาพ 4 ภาพรวมของการเรียนรู้แบบมีผู้สอน (Supervised learning) [9]

การเรียนรู้แบบไม่มีผู้สอน (Unsupervised learning) จำเป็นต้องใช้ชุดข้อมูลฝึก (Training data) จำนวนมากที่อยู่ในรูปของเวกเตอร์ (Vector) ด้วยการสกัดคุณลักษณะ (Feature extraction) เพื่อให้คอมพิวเตอร์เรียนรู้ข้อมูลได้ด้วยตนเองจากข้อมูลที่ป้อนให้ เมื่อเรียนรู้เสร็จจะสร้างแบบจำลองสำหรับพยากรณ์ (Predictive Model) ขึ้นมาเพื่อใช้สำหรับทดสอบโดยนำชุดข้อมูลทดสอบ (Test data) ที่อยู่ในรูปของเวกเตอร์ มาเปรียบเทียบกับแบบจำลองที่สร้างขึ้น โดยผลลัพธ์ที่ได้จะแสดงเป็นฉลากของกลุ่มข้อมูล และการเรียนรู้แบบมีผู้สอน (Supervised learning) มีขั้นตอนเช่นเดียวกับการเรียนรู้แบบไม่มีผู้สอน แต่ต่างกันตรงที่การเรียนรู้แบบมีผู้สอนจำเป็นต้องป้อนฉลากหรือเฉลย (Label) ของชุดข้อมูลฝึกให้กับคอมพิวเตอร์ เพื่อให้คอมพิวเตอร์เรียนรู้ได้อย่างถูกต้องตามที่ผู้สอนหรือผู้ฝึกกำหนด

การสกัดคุณลักษณะ (Feature extraction)

การสกัดคุณลักษณะ [10] คือการหาลักษณะจำเพาะของแต่ละภาพที่ป้อนเข้า ให้อยู่ในรูปแบบเวกเตอร์ (Vector) เพื่อให้สะดวกในการนำไปใช้สำหรับวิเคราะห์ข้อมูลที่อาศัยหลักการสถิติมาช่วยในการพยากรณ์ ซึ่งการสกัดคุณลักษณะสามารถสกัดได้หลายมุมมอง เช่น สี รูปร่าง ลวดลาย เป็นต้น โดยข้อมูลเวกเตอร์จะถูกนำไปเรียนรู้และเข้าสู่กระบวนการจัดกลุ่ม (Clustering) (สำหรับการเรียนรู้แบบไม่มีผู้สอน) หรือการจำแนกประเภท (Classification) (สำหรับการเรียนรู้แบบมีผู้สอน) และผลลัพธ์ที่ได้คือฉลากของกลุ่มหรือประเภทของข้อมูลที่ป้อนเข้า ดังภาพ 5

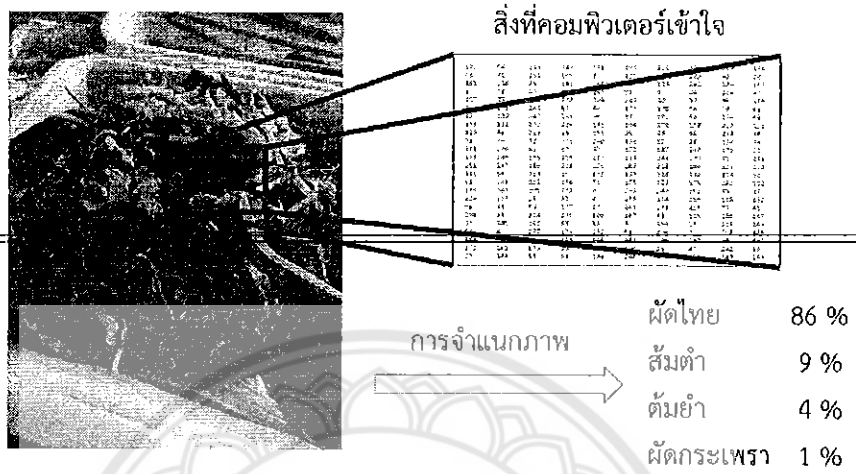


ภาพ 5 ระบบการเรียนรู้ของเครื่องจักร [7]

การจำแนกภาพ (Image Classification)

การจำแนกหรือแยกประเภทภาพ คือวิธีในการระบุประเภท (Category) ของภาพจากชุดข้อมูลฝึก (Training set) ให้สามารถเรียนรู้ในการแยกแยะประเภทของภาพได้ โดยอาศัยการสกัดคุณลักษณะ (Feature extraction) เปลี่ยนข้อมูลภาพให้อยู่ในรูปแบบตัวเลขที่คอมพิวเตอร์เข้าใจได้ เพื่อนำข้อมูลมาใช้สำหรับการเรียนรู้เพื่อสร้างตัวแทนของแต่ละประเภท ซึ่งการจำแนกภาพจัดได้ว่าเป็นแขนงหนึ่งของคอมพิวเตอร์วิทัศน์ (Computer Vision) และวิธีการที่ช่วยลดให้การจำแนกภาพมี

ความผิดพลาดน้อยลง คือการนำวิธีการตรวจหาวัตถุ (Object detection) และการแบ่งส่วน (Segmentation) มาประมวลผลก่อนการจำแนกภาพ



ภาพ 6 ตัวอย่างภาพอาหารที่ใช้ในการจำแนก

จากภาพ 6 ตัวอย่างภาพอาหารที่ใช้ในการจำแนก โดยสมมติว่าในฐานข้อมูลมี 4 กลุ่มได้แก่ (ผัดไทย ส้มตำ ผัดกระเพรา ต้มยำ) และขนาดความสูง 300 จุดภาพ (Pixel) ความกว้าง 400 จุดภาพ ซึ่งข้อมูลที่คอมพิวเตอร์เข้าใจได้ อยู่ในรูปแบบแถวลำดับตัวเลข 3 มิติ (3-Dimensional array) ประกอบไปด้วยค่าจุดภาพในโหมดสี 3 ช่องสีคือ สีแดง สีเขียว สีฟ้า (RGB) ดังนั้นข้อมูลของภาพดังกล่าวมีจำนวน 300 x 400 x 3 จำนวน หรือทั้งหมด 360,000 จำนวน โดยแต่ละจำนวนสามารถมีค่าได้ระหว่าง 0 (สีดำ) ถึง 255 (สีขาว) และเมื่อทำการจำแนกภาพดังกล่าวผลที่ได้คือ "ผัดไทย"

มุมมอง (Perspective) [11, 12] ของภาพถือว่าเป็นปัญหาสำหรับการจำแนกภาพ เนื่องจากภาพสามารถถ่ายได้หลายมุม รวมทั้งในแต่ละสถานที่มีความสว่าง (Brightness) ที่แตกต่างกัน ซึ่งสามารถแบ่งปัญหาออกเป็น 7 กลุ่ม ดังภาพ 7 ดังนี้

1. การเปลี่ยนแปลงมุมมอง (Viewpoint variation) คือภาพที่ได้จากการถ่ายหลากหลายมุมมองจึงทำให้เกิดความแตกต่างของภาพ แต่ภาพที่ได้ยังคงสื่อถึงประเภทเดียวกัน

2. การเปลี่ยนแปลงสเกล (Scale variation) คือภาพที่ได้จากการถ่ายในระยะที่ใกล้ไกลแตกต่างกันออกไป ทำให้ขนาดของสิ่งที่ต้องการมีความหลากหลาย

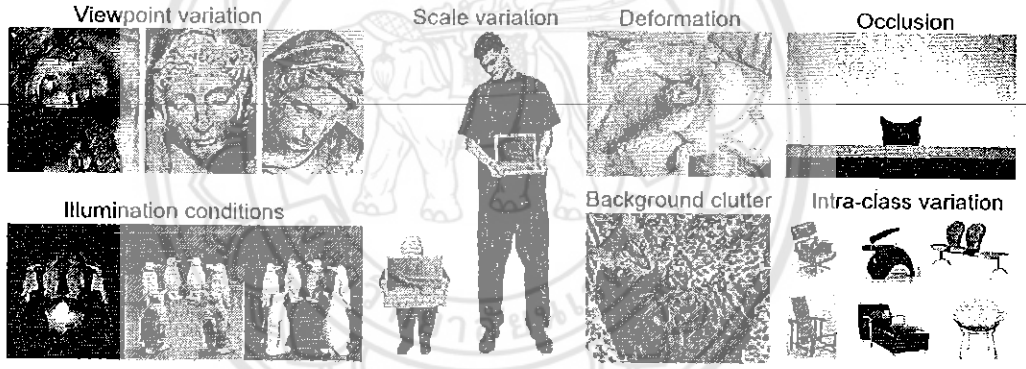
3. การเปลี่ยนรูป (Deformation) คือภาพที่ได้จากการถ่ายของสิ่งที่มีการเปลี่ยนลักษณะรูปร่างได้ ซึ่งส่วนมากส่วนเป็นสิ่งมีชีวิต เช่น ลักษณะการนอนของแมวที่เปลี่ยนรูปตามสภาพแวดล้อม

4. การซ่อนเร้น (Occlusion) คือ ภาพที่ต้องการมองเห็นสิ่งที่ต้องการ แต่สิ่งเหล่านั้นซ่อนอยู่กับสภาพแวดล้อม จึงทำให้มองเห็นได้ยากหรือมองเห็นเฉพาะบางส่วน

5. เงื่อนไขความสว่าง (Illumination conditions) คือ ความสว่างที่เกิดขึ้นระหว่างการถ่ายภาพหรือการตกแต่งภาพ โดยความสว่างเป็นปัจจัยสำคัญ หากมีความสว่างมากหรือน้อยเกินไปอาจทำให้สีของวัตถุหรือสิ่งของเปลี่ยน ทำให้ความหมายของภาพเปลี่ยนไปได้

6. การรวมเป็นกลุ่มเดียวกัน (Background clutter) คือภาพที่ต้องการมีความกลมกลืนกับสภาพแวดล้อม เช่น ภาพแมวที่นอนบนพรมสีเดียวกับลายของแมว

7. การเปลี่ยนแปลงภายในกลุ่ม (Intra-class variation) คือภาพที่สามารถมีได้หลากหลายรูปร่างและขนาดที่แตกต่างกัน แต่จัดอยู่ในกลุ่มหรือประเภทเดียวกัน



ภาพ 7 ตัวอย่างความหลากหลายจากมุมมองของภาพ

ที่มา: Fei-Fei Li, Andrej Karpathy และ Justin Johnson. Stanford university

จากปัญหาของภาพที่กล่าวมา ทำให้ภาพในฐานข้อมูลในกลุ่มเดียวกันมีความหลากหลายเกิดขึ้น ซึ่งการจำแนกภาพที่มีประสิทธิภาพต้องสามารถเรียนรู้และเข้าใจความหลากหลายนี้ได้ ซึ่งขั้นตอนในการจำแนกสามารถแบ่งออกได้ 3 ส่วนดังนี้

1. ส่วนเก็บและเตรียมข้อมูล (Input) คือการเก็บและจัดเตรียมข้อมูล โดยแต่ละกลุ่มมีชื่อที่แตกต่างกัน ซึ่งเรียกข้อมูลนี้ว่า "Training set"

2. ส่วนการเรียนรู้ (Learning) คือการนำ Training set ในแต่ละกลุ่มมาเรียนรู้ เพื่อหาข้อมูลที่ใช้แทนกลุ่ม ซึ่งเรียกขั้นตอนนี้ว่า "Training a classifier" หรือ "Learning a model"

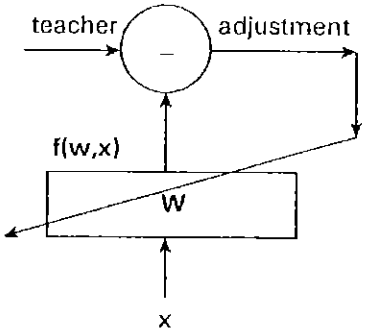
3. ส่วนการประเมินค่า (Evaluation) คือ การนำแบบจำลองที่เรียนรู้มาใช้วัดคุณภาพของความถูกต้อง โดยนำข้อมูลที่ใช้สำหรับทดสอบ (Testing set) มาเปรียบเทียบกับแบบจำลองเพื่อระบุว่าข้อมูลดังกล่าวอยู่ในกลุ่มใด หากระบุได้ถูกต้องจะเรียกแบบจำลองนี้ว่า "Ground truth"

การจำแนกเชิงเส้น (Linear Classification)

การจำแนกเชิงเส้น [13] ถือว่าเป็นส่วนหนึ่งในกระบวนการเรียนรู้ของเครื่องที่ใช้หลักในการจำแนกทางสถิติ (Statistical classification) ช่วยให้แบ่งกลุ่มหรือประเภทของชุดข้อมูลได้จากการตัดสินใจที่อาศัยหลักการของผลรวมเชิงเส้น (Linear combination) โดยใช้ชุดข้อมูลฝึก X ที่สกัดคุณลักษณะอยู่ในรูปของเมทริกซ์ที่มีขนาดมิติ $1 \times N$ เช่น $X = [x_1, x_2, ..., x_N]$ และชุดข้อมูลฝึกแต่ละชุดต้องมีฉลากหรือเฉลย Y กำกับไว้ทุกชุดตามจำนวนของชุดข้อมูลฝึก $Y = [y_1, y_2, ..., y_N]$ ดังนั้นเมื่อชุดข้อมูลฝึกประกอบไปด้วยฉลาก จึงเขียนในรูปของสมการได้ดังสมการ 2.2 [7] ซึ่งการจำแนกเชิงเส้นแบ่งขั้นตอนได้ 3 ขั้นตอน ได้แก่ ขั้นตอนการเรียนรู้ (Learning phase) ขั้นตอนการประเมินค่า (Evaluation phase) และขั้นตอนการทำนาย (Prediction phase)

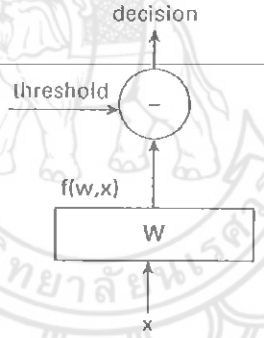
$$(X, Y) = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N) \tag{2.2}$$

ขั้นตอนการเรียนรู้ (Learning phase) คือขั้นตอนการหาค่าน้ำหนัก (W, weight) จากชุดข้อมูลฝึก (X) โดยใช้ $f(W, x)$ โดยผลลัพธ์ที่ได้จะถูกตรวจสอบด้วยฉลากหรือเฉลย (Y) ที่ผู้สอนป้อนให้ ซึ่งระบบจะปรับเปลี่ยนค่าน้ำหนักที่เหมาะสมกับชุดข้อมูลตามที่มีการกำหนดไว้ก่อน ดังภาพ 8



ภาพ 8 ขั้นตอนการเรียนรู้ (Learning phase) ของการจำแนกเชิงเส้น [7]

ขั้นตอนการประเมินค่า (Evaluation phase) คือขั้นตอนการตรวจสอบความถูกต้องจากค่าน้ำหนักที่ได้จากขั้นตอนการเรียนรู้ว่ามีความถูกต้องเพียงใด ด้วยการนำชุดข้อมูลสำหรับทดสอบ (ข้อมูลที่ไม่อยู่ในชุดข้อมูลฝึก) แต่ละชุดมาตรวจสอบกับค่าน้ำหนัก จากนั้นพิจารณาผลลัพธ์ตาม ขีดแบ่ง (Threshold) ที่ถูกกำหนดไว้ก่อนล่วงหน้า เพื่อตัดสินระบุว่าชุดข้อมูลที่นำมาทดสอบอยู่ในกลุ่มใด ดังภาพ 9



ภาพ 9 ขั้นตอนการทำนาย (Prediction phase) ของการจำแนกเชิงเส้น [7]

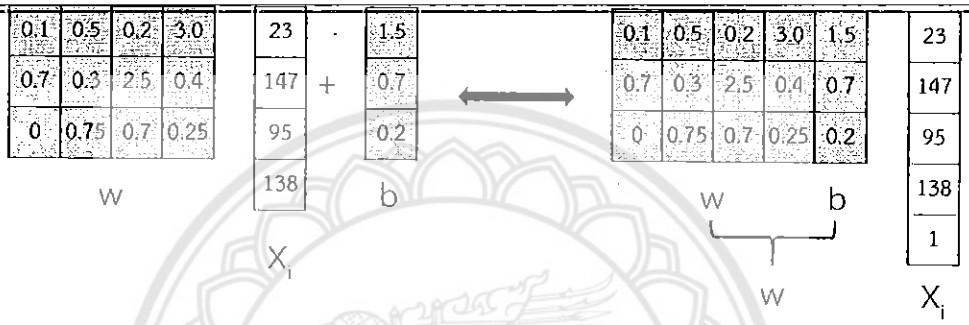
ขั้นตอนการทำนาย (Prediction phase) คือขั้นตอนการนำค่าน้ำหนักที่ผ่านขั้นตอนการประเมินค่าที่ผลลัพธ์มีความถูกต้องมากที่สุดมาใช้กับแอพพลิเคชั่นที่เกี่ยวข้องกับชุดข้อมูล

ดังนั้นจึงสามารถสรุปการทำงานของการจำแนกเชิงเส้นแบบมีผู้สอนทั้งสามขั้นตอนได้ด้วยสมการ 2.3 [13] โดยที่ x_i คือชุดข้อมูลฝึกที่อยู่ในรูปของเวกเตอร์ W คือค่าน้ำหนักอยู่ในรูปของเวกเตอร์ (w, weight vector) และ b คือค่าความโน้มเอียง (Bias vector)

$$f(x_i, W, b) = Wx_i + b \tag{2.3}$$

จากสมการ 2.3 พบว่าข้อมูลป้อนเข้าให้กับระบบมีสามค่าคือ x_i W และ b ซึ่งสามารถลดขั้นตอนการบวกกับ b ได้ โดยเพิ่มจำนวนของเมทริกซ์ W มาอีกมิติและนำค่า bias vector ไปต่อกับค่าน้ำหนักแล้วเพิ่มจำนวนของเมทริกซ์ x_i อีกมิติ โดยแทน x_i ที่เพิ่มมาด้วยค่า 1 ดังภาพ 10 ทำให้สมการ 2.3 เหลือข้อมูล x_i และ W ที่ป้อนเข้าระบบเท่านั้น ดังสมการ 2.4 [7]

$$f(x_i, W) = Wx_i \tag{2.4}$$



ภาพ 10 การลดรูปของสมการ 2.4 เพื่อลดขั้นตอนการบวกค่าความโน้มเอียง

จากภาพ 10 การลดรูปของ $f(x_i, W, b) = Wx_i + b$ ให้เหลือ $f(x_i, W) = Wx_i$ โดยที่ลดขั้นตอนการบวกค่า bias vector ทำให้ขั้นตอนการบวกไม่ได้ถูกนำมาใช้ ส่งผลให้เกิดประโยชน์ในขั้นตอนการพัฒนาแอพพลิเคชั่นบนสมาร์ทโฟนที่มีทรัพยากรจำกัด เนื่องจากสามารถช่วยลดการประมวลผลของหน่วยประมวลผลกลาง (CPU) ลงได้ ซึ่งช่วยให้สามารถประมวลผลได้เร็วขึ้น

เมื่อนำข้อมูลทดสอบเข้าขั้นตอนการทำนาย x ผลลัพธ์ที่ได้คือคะแนน (Score) ของแต่กลุ่มจากการพยากรณ์ด้วยฟังก์ชัน $f(x_i, W)$ ดังภาพ 11 คือตัวอย่างการจำแนกประเภทของอาหารที่มีอาหารสำหรับทดสอบ 3 ประเภทคือ หมูสะเต๊ะ ก๋วยเตี๋ยว และข้าวมันไก่ ซึ่งมีฐานข้อมูลของประเภทอาหารทั้งหมด 10 ประเภท ทำให้แต่ละภาพที่ทดสอบจะแสดงคะแนนทั้ง 10 ประเภท โดยสามารถหาคำตอบของการพยากรณ์ (P) จากสมการ 2.5 โดยคะแนนที่มากที่สุดจาก 10 ประเภทคือคำตอบที่พยากรณ์

$$P = \underset{i=1,\ldots,k}{\arg\max}\, f(x, W_i) \tag{2.5}$$

	หมูสะเต๊ะ	ก๋วยเตี๋ยว	ข้าวมันไก่
			
ส้มตำ	0.2	-6.41	2.94
ผัดไทย	-5.1	3.39	0.41
ผัดกระเพรา	2.03	-1.05	1.36
ก๋วยเตี๋ยว	1.17	4.04	-1.09
ข้าวหมูแดง	-1.02	1.31	2.05
ข้าวมันไก่	3.79	-2.7	-3.02
เย็นตาโฟ	-4.17	7.01	-4.1
คอหมูย่าง	0.15	-2.5	1.63
ราดหน้า	2.76	0.44	2.98
หมูสะเต๊ะ	4.82	5.07	2.5

ภาพ 11 ตัวอย่างคะแนน (Score) จากการพยากรณ์ด้วย $f(x)$

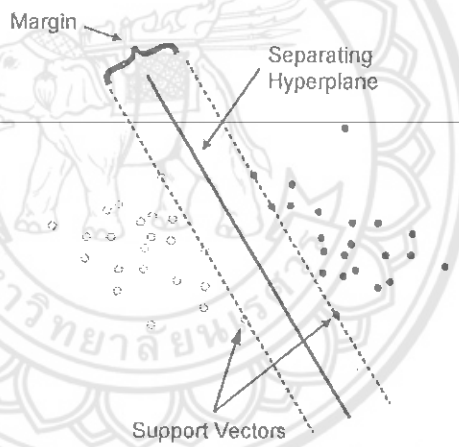
จากการพยากรณ์ของภาพ 11 พบว่าคะแนนของภาพแรก (หมูสะเต๊ะ) ที่มากที่สุดคือ 4.82 (หมูสะเต๊ะ) ซึ่งพยากรณ์ได้ถูกต้องเมื่อเทียบกับผลเฉลย และพิจารณาภาพที่สอง (ก๋วยเตี๋ยว) มีคะแนนที่มากที่สุดคือ 7.01 (เย็นตาโฟ) พบว่าพยากรณ์ผิดพลาดเมื่อเทียบกับผลเฉลย และสุดท้ายพิจารณาภาพที่สาม (ข้าวมันไก่) ได้คะแนนมากที่สุดคือ 2.98 (ราดหน้า) พบว่าพยากรณ์ผิดพลาด ดังนั้นหากพิจารณาคะแนนจากภาพที่สอง (ก๋วยเตี๋ยว) ในกรณีที่พยากรณ์ผิดพลาด พบว่าคะแนนที่พยายกรณ์ว่าเป็นก๋วยเตี๋ยว คือ 4.04 (ก๋วยเตี๋ยว) เมื่อนำคะแนนมาเรียงลำดับตามสมการ 2.5 และแสดงเพียง 3 ลำดับ คือ เย็นตาโฟ หมูสะเต๊ะ และก๋วยเตี๋ยว ซึ่งพบว่าการพยากรณ์ที่ถูกต้องของรูปที่สองอยู่ลำดับที่ 3 หากพิจารณาเฉพาะลำดับที่หนึ่งจะแสดงเพียงความถูกต้องของระบบเท่านั้น จึงต้องมีการพิจารณาคำตอบที่พยากรณ์ถูกต้องในลำดับอื่น ๆ ด้วยการใช้ฟังก์ชันความสูญเสียเพื่อระบุว่ามีการพยากรณ์มีความแม่นยำเพียงใด

ฟังก์ชันความสูญเสียสำหรับการจำแนก (Loss function for classification)

การตรวจสอบ W หรือค่าน้ำหนักว่าเหมาะสมกับ x_i, y_i สามารถวัดได้จากค่าความถูกต้องในการพยากรณ์ด้วยฟังก์ชันความสูญเสีย (Loss function) [14, 15] หมายถึงการตรวจสอบผลลัพธ์ของกลุ่ม (Class) ที่ได้จากการพยากรณ์ด้วยคะแนน (Score) ของ $f(x_i, W)$ ที่มากที่สุดของกลุ่ม

ทั้งหมดว่าเป็นกลุ่มเดียวกับผลเฉลย y เพื่อหาค่าที่แตกต่างระหว่างความถูกต้องของการพยากรณ์กับผลเฉลย ถ้าการจำแนกมีความแม่นยำในการพยากรณ์ค่าความสูญเสียจะมีค่าต่ำ แต่ถ้าการจำแนกมีความผิดพลาดในการพยากรณ์ค่าความสูญเสียจะมีค่าสูง ซึ่งฟังก์ชันความสูญเสียที่ใช้ในซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machine) คือ Hinge loss

Hinge loss [15, 16] คือฟังก์ชันความสูญเสียที่ถูกนำไปใช้กับตัวแยกประเภทข้อมูลฝึก (training classifier) เพื่อหา Maximum margin สำหรับแยกระนาบเกิน (Separating hyperplane) ที่สามารถแยกข้อมูลฝึกแต่ละกลุ่มออกจากกันในขั้นตอนการจำแนกของซัพพอร์ตเวกเตอร์แมชชีน ดังภาพ 12 ตัวอย่างการจำแนกข้อมูลสองกลุ่ม (Binary classification) ด้วยการหาเส้น Separating hyperplane จากซัพพอร์ตเวกเตอร์ (Support Vector) ของข้อมูลทั้งสองกลุ่ม เพื่อใช้สำหรับเป็นเส้นแบ่งของทั้งสองกลุ่มออกจากกัน



ภาพ 12 การแยกระนาบเกินของซัพพอร์ตเวกเตอร์แมชชีน [17]

Hinge loss จึงถูกนำมาใช้เพื่อตรวจสอบการแยกข้อมูลด้วยเส้นแบ่ง จากการคำนวณ $f(x_i, W)$ โดยผลลัพธ์คือคะแนนของแต่ละกลุ่ม ตัวอย่างเช่น คะแนนมีจำนวน j กลุ่ม จะได้ $s_j = f(x_i, W)_j$ ดังนั้นเมื่อนำ Hinge loss มาใช้กับซัพพอร์ตเวกเตอร์แมชชีนแบบหลายกลุ่ม จึงสามารถเขียนสมการการหาค่าความสูญเสียได้ดังสมการ 2.6

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \tag{2.6}$$

	หมูสะเต๊ะ	ก๋วยเตี๋ยว	ข้าวมันไก่
			
หมูสะเต๊ะ	4.82	5.07	2.5
ก๋วยเตี๋ยว	1.17	4.04	-1.09
ข้าวมันไก่	3.79	-2.7	-3.02

Loss ของหมูสะเต๊ะ = max(0, 1.17 – 4.82 + 1) + max(0, 3.79 - 4.82 + 1)
= max(0, -2.65) + max(0, -0.03)
= 0

Loss ของก๋วยเตี๋ยว = max(0, 5.07 – 4.04 + 1) + max(0, -2.7 - 4.04 + 1)
= max(0, 2.03) + max(0, -5.74)
= 2.03

Loss ของข้าวมันไก่ = max(0, 2.5 – (-3.02) + 1) + max(0, (-1.09) – (-3.02) + 1)
= max(0, 6.52) + max(0, 2.93)
= 6.52 + 2.93
= 9.45

Loss ทั้งหมด = (0 + 2.03 + 9.45) / 3 => 3.8267

ภาพ 13 ตัวอย่างคะแนน (Score) จากการพยากรณ์ 3 กลุ่มด้วย $f(x_i, W)$

เมื่อนำคะแนน (Score) ที่ได้จากการพยากรณ์ 3 กลุ่มด้วย $f(x_i, W)$ ดังภาพ 13 ที่ยกตัวอย่างเพียง 3 ประเภทของภาพที่นำมาทดสอบ มาหาค่า Hinge loss ด้วยสมการ 2.6 ซึ่งทำการคำนวณหาทีละภาพ จะได้ค่า Loss ของหมูสะเต๊ะคือ 0 ค่า Loss ของก๋วยเตี๋ยวคือ 2.03 และค่า Loss ของข้าวมันไก่คือ 9.45 ซึ่งมีความหมายว่าค่า Loss ที่มีค่าเท่ากับ 0 หมายถึงพยากรณ์ถูกต้องในอันดับหนึ่งและหากพยากรณ์ผิดพลาดค่า Loss จะเพิ่มขึ้น เช่น ภาพก๋วยเตี๋ยวมีการพยากรณ์ถูกต้องในอันดับสอง ค่า Loss ที่ได้คือ 2.03 และภาพข้าวมันไก่มีการพยากรณ์ถูกต้องในอันดับสาม ค่า Loss ที่ได้คือ 9.45 จะเห็นว่าค่า Loss สามารถมีค่าจากคะแนนที่ได้ในการพยากรณ์ ซึ่งไม่ได้มีการกำหนดขอบเขตของค่า Loss ไว้ชัดเจน จึงทำให้เมื่อนำค่า Loss แต่ละภาพมาหาค่าเฉลี่ยหรือที่เรียกว่าค่า Loss ทั้งหมด ที่มีค่าเท่ากับ 3.8267 ซึ่งหมายความว่าการพยากรณ์ทั้งหมดยังมีความผิดพลาดเกิดขึ้น

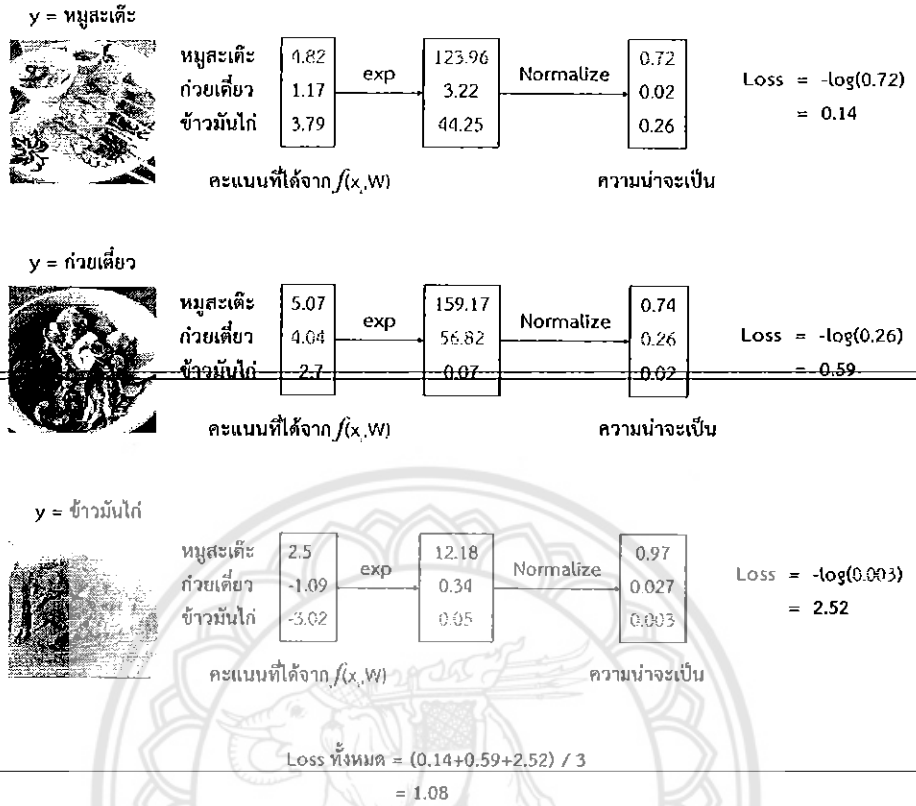
ฟังก์ชันซอฟต์แม็กซ์ (Softmax function)

ฟังก์ชันซอฟต์แม็กซ์ [18, 19] คือฟังก์ชันที่อาศัยความน่าจะเป็นของคะแนนที่ได้จากการพยากรณ์มาใช้ในการจำแนกข้อมูลหลายกลุ่ม (multiclass classification) ซึ่งคะแนนที่ได้จาก $f(x_i,W)$ ถูกทำให้เป็นบรรทัดฐานจากเลขชี้กำลัง (Normalize exponential) หรือค่าความน่าจะเป็น (probability) โดยปรับคะแนนที่ได้จาก $f(x_i,W)$ ให้เป็นเลขชี้กำลังเพื่อเปลี่ยนคะแนนที่มีค่าลบเป็นค่าบวกทั้งหมด จากนั้นทำให้เป็นบรรทัดฐานเดียวกัน จะได้ค่าของความน่าจะเป็นสำหรับพยากรณ์ โดยมีค่าระหว่าง 0 ถึง 1 และเมื่อนำฟังก์ชันลอการิทึม (logarithm) มาคำนวณจากค่าความน่าจะเป็น จะสามารถหาค่า Loss ได้ดังสมการ 2.7 ซึ่งค่า Loss ที่ได้สามารถตีความหมายได้ง่ายกว่า Hinge loss เนื่องจาก Hinge loss เหมาะสำหรับการจำแนกข้อมูลสองกลุ่ม (Binary classification) และเมื่อนำมาใช้กับการจำแนกข้อมูลหลายกลุ่มทำให้พิสัยของค่า Hinge Loss มีมาก จึงทำให้ยากในการตีความเมื่อเทียบกับแต่ค่า Loss ที่เกิดจากฟังก์ชันซอฟต์แม็กซ์

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \tag{2.7}$$

โดยที่ i คือดัชนีของคะแนนที่ต้องการหาค่า Loss และ f_j คือคะแนนที่ได้จาก $f(x_i,W)$ และ f_{y_i} คือคะแนนที่ได้จาก $f(x_i,W)$ ที่เลือกตามดัชนี i ของผลเฉลย y

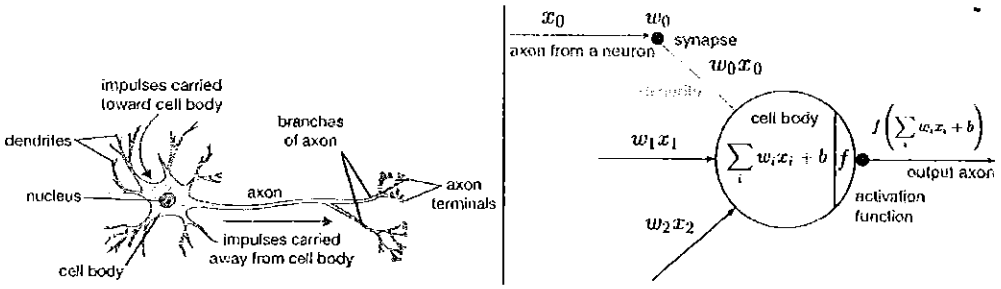
การหาค่า Loss ของแต่ละภาพ จะนำค่าความน่าจะเป็นที่ได้จาก $f(x_i,W)$ เข้าฟังก์ชันลอการิทึม หากค่าความน่าจะเป็นมีค่าสูง ค่า Loss ที่เกิดขึ้นจะมีค่าที่ต่ำและถ้าค่าความน่าจะเป็นมีค่าต่ำ ค่า Loss ที่เกิดขึ้นจะมีค่าที่สูง ตัวอย่างการหาค่า Loss ด้วยฟังก์ชันซอฟต์แม็กซ์ ดังภาพ 14 และเมื่อนำค่า Loss ที่ได้ในแต่ละภาพมาหาค่าเฉลี่ยจะได้ค่า Loss ทั้งหมดคือ 1.08 โดยค่า Loss ที่ใช้ฟังก์ชันซอฟต์แม็กซ์มีค่าน้อยกว่า Hinge loss จากภาพ 17 ที่มีค่า Loss ทั้งหมดคือ 3.8267 ซึ่งค่า Loss ที่ใช้ฟังก์ชันซอฟต์แม็กซ์มีค่าที่ต่ำกว่า เนื่องจากมีการปรับคะแนนที่ได้จาก $f(x_i,W)$ เป็นบรรทัดฐานจากเลขชี้กำลัง ทำให้ค่าที่ได้คือค่าความน่าจะเป็นที่มีพิสัยระหว่าง 0 ถึง 1 ดังนั้นค่า Loss ที่ได้จึงมีพิสัยที่ต่ำกว่า Hinge loss เนื่องจากฟังก์ชันซอฟต์แม็กซ์ได้มีการหาค่าความน่าจะเป็นของทุกกลุ่มก่อนนำมาหาค่า Loss จึงทำให้ค่า Loss ที่ได้มีค่าน้อยกว่า Hinge loss ที่ไม่ได้มีการทำให้เป็นบรรทัดฐานหรือหาค่าความน่าจะเป็นก่อนนำมาหาค่า Loss



ภาพ 14 ตัวอย่างการหาค่า Loss ด้วยฟังก์ชันซอฟต์แม็กซ์

โครงข่ายประสาทเทียมแบบลึก (Deep neural networks)

โครงข่ายประสาทเทียม (Neural networks) คืออัลกอริทึมที่จำลองการทำงานของคอมพิวเตอร์ให้สามารถทำงานเช่นเดียวกับสมองของมนุษย์ ประกอบไปด้วยเซลล์ประสาท (neurons) คือหน่วยประมวลผล (processing elements) ที่มีใจกลาง (nucleus) เป็นหน่วยประมวลผลกลาง โดยรับข้อมูลเข้าด้วยใยประสาทนำเข้า (dendrites) แล้วข้อมูลถูกส่งออกด้วยแกนประสาทนำออก (axon) และเชื่อมต่อระหว่างเซลล์ประสาทด้วยจุดประสานประสาท (synapses) ที่มีค่าน้ำหนัก (weight) ระหว่างหน่วยประมวลผลในโครงข่ายประสาท



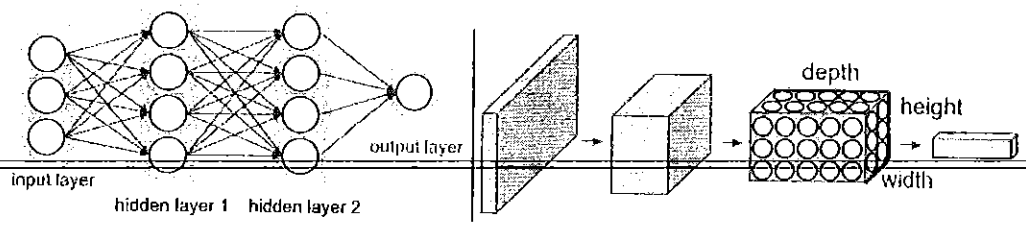
ภาพ 15 รูปแบบของเซลล์ประสาททางชีววิทยา (ซ้าย)
และแบบจำลองของเซลล์ประสาททางคณิตศาสตร์ (ขวา)

ที่มา: Fei-Fei Li, Andrej Karpathy และ Justin Johnson. Stanford university

จากภาพ 15 รูปแบบการทำงานของเซลล์ประสาททางชีววิทยา (ซ้าย) โดยแต่ละเซลล์ประสาทจะได้รับสัญญาณจากใยประสาทนำเข้าและผลิตสัญญาณส่งออกไปยังแกนประสาทนำออกเพื่อส่งต่อไปยังใยประสาทนำเข้าของเซลล์ประสาทอื่น ๆ ที่เชื่อมต่อกันด้วยจุดประสานประสาท และแบบจำลองของเซลล์ประสาททางคณิตศาสตร์ (ขวา) สัญญาณที่ได้รับจากแกนประสาทนำออก (w_0) ที่เชื่อมต่อกับจุดประสานประสาท (w_0) ที่มีค่าน้ำหนัก เมื่อสัญญาณผ่านจุดประสานประสาทมาถึงใยประสาทนำเข้า สัญญาณกับค่าน้ำหนักจะถูกนำเข้าฟังก์ชันเชิงการคูณ (multiplicative function) ทำให้ได้สัญญาณ (w_0x_0) ซึ่งในภาพ 15 (ขวา) มีใยประสาทนำเข้าอีกสองสัญญาณ (w_1x_1, w_2x_2) เมื่อทั้งสองสัญญาณมาถึงตัวเซลล์ (Cell body) จะถูกนำมารวมกันโดยนำเข้าฟังก์ชันกระตุ้น (activation function) แล้วส่งสัญญาณไปยังแกนประสาทนำออก

โครงข่ายประสาทเทียมแบบลึกหรือเชิงลึกมีลักษณะโครงสร้างเช่นเดียวกับโครงข่ายประสาทเทียมแบบปรกติ โดยโครงข่ายประสาทเทียมแบบลึกจัดอยู่ในประเภทโครงข่ายการป้อนแบบไม่ย้อนกลับ (feed-forward) ที่กำหนดให้มีการส่งข้อมูลจาก Input Layer ที่ถูกกำหนดให้เป็นลำดับชั้นแรกสุด ไปยัง Hidden Layer ที่ถูกซ่อนอยู่และส่งต่อไปยัง Output Layer ที่อยู่ลำดับชั้นสุดท้าย ดังนั้นการส่งข้อมูลจะถูกส่งในทิศทางเดียวกันจนถึง Output Layer โดยไม่มีการย้อนกลับ ซึ่งโครงข่ายประสาทเทียมแบบลึกถูกพัฒนาให้เรียนรู้ข้อมูลเชิงลึกได้ด้วยการเพิ่มจำนวนของ Hidden Layer และสามารถเรียนรู้ได้ครอบคลุมทั้งสามมิติ ได้แก่ ความกว้าง (width) ความสูง (height) และความลึก (depth) ของภาพ โดยที่ความลึกคือช่องสีของแบบจำลองสี RGB ซึ่งโครงข่ายประสาทเทียมแบบ

สังวัตนาการ (Convolutional Neural Networks หรือ ConvNets) เป็นหนึ่งในวิธีการของโครงข่ายประสาทเทียมแบบลึกที่ได้รับความนิยมและถูกนำมาใช้ในการเรียนรู้เกี่ยวกับภาพโดยเฉพาะ



ภาพ 16 โครงข่ายประสาทเทียม 3 ลำดับชั้น (ซ้าย) โครงข่ายประสาทเทียมแบบลึก (ขวา)

ที่มา: Fei-Fei Li, Andrej Karpathy และ Justin Johnson. Stanford university

จากภาพ 16 โครงข่ายประสาทเทียมแบบ 3 ลำดับชั้น (ซ้าย) ประกอบไปด้วยลำดับชั้นนำเข้า (Input Layer) ที่มี 3 ข้อมูล ลำดับชั้นที่ถูกซ่อน (Hidden Layer) 2 ลำดับชั้นแต่ละชั้นมี 4 เซลล์ประสาท และลำดับชั้นนำออก (Output Layer) โดยแต่ละลำดับชั้นจะใช้ฟังก์ชันกระตุ้นในการประมวลผล ยกเว้นลำดับชั้นนำออก เนื่องจากลำดับชั้นนำออกถูกนำไปใช้ในการแทนคะแนนของกลุ่ม (สำหรับการจำแนก) และสำหรับโครงข่ายประสาทเทียมแบบลึก (ขวา) ได้ปรับโครงสร้างของโครงข่ายให้สามารถประมวลผลในหนึ่งลำดับชั้นได้ทั้งสามมิติ (ความกว้าง ความสูง และความลึก) โดยแต่ละลำดับชั้นของโครงข่ายจะแปลงข้อมูลนำเข้าแบบสามมิติให้เป็นข้อมูลนำออกแบบสามมิติของด้วยฟังก์ชันกระตุ้นในเซลล์ประสาท ในตัวอย่างภาพ 16 ลำดับชั้นนำเข้า (สีแดง) คือภาพที่ถูกนำเข้าโครงข่าย ส่งภาพไปแบ่งเป็นส่วนเล็ก ๆ ด้วยลำดับชั้นที่ถูกซ่อน (สีน้ำเงิน) แล้วส่งต่อไปลำดับชั้นที่ถูกซ่อนถัดไปเพื่อประมวลผลความลึกของภาพ (RGB) จากนั้นส่งข้อมูลไปยังลำดับชั้นถัดไปเพื่อประมวลจนกระทั่งได้ข้อมูลที่สามารถนำไปใช้สำหรับขั้นตอนการจำแนก

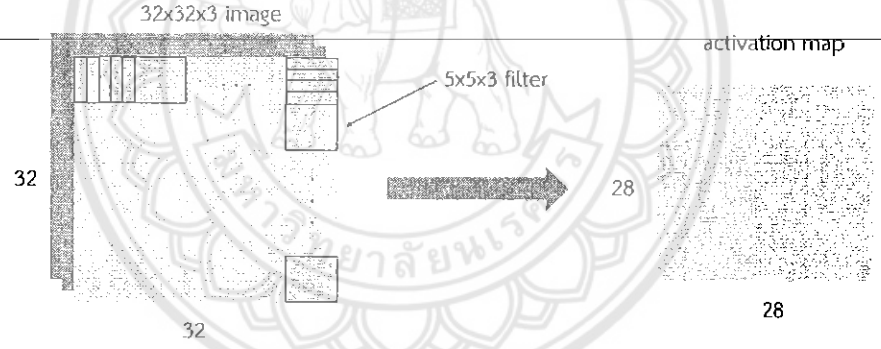
ลำดับชั้นของโครงข่ายประสาทเทียมแบบสังวัตนาการ (ConvNets layers)

โครงข่ายประสาทเทียมแบบสังวัตนาการประกอบไปด้วยลำดับชั้น Convolution Layer Pooling Layer และ Fully-Connected Layer โดยอย่างน้อยต้องมีสามลำดับชั้นดังกล่าวมาเรียงทับ

ซ้อนกันตามโครงสร้างของโครงข่ายประสาทเทียม เพื่อเรียนรู้และการจำแนกประเภทของฐานข้อมูลที่ถูกนำมาใช้ ซึ่งประกอบไปด้วยลำดับชั้นดังต่อไปนี้

Input Layer คือลำดับชั้นที่เก็บข้อมูลภาพของชุดข้อมูลฝึก (Training data) และชุดข้อมูลทดสอบ (Test data) โดยภาพอยู่ในแบบจำลองสี RGB และขนาดของภาพขึ้นอยู่กับแบบจำลองของโครงข่ายที่นำมาใช้ ตัวอย่างเช่นแบบจำลองของโครงข่ายที่ใช้ภาพความกว้าง 32 ความสูง 32 ดังนั้นข้อมูล 1 ภาพ เท่ากับ [32x32x3] โดย 3 คือจำนวนช่องของสีตามแบบจำลองสี RGB

Convolution Layer คือลำดับชั้นที่ทำการคำนวณพื้นที่ส่วนเล็ก ๆ ของข้อมูลที่ได้จากลำดับชั้นก่อนหน้า ที่หาผลคูณจุดด้วยสมการ 2.3 โดยใช้ค่าน้ำหนักหรือตัวกรอง (W) กับพื้นที่ส่วนเล็ก ๆ ของข้อมูล (x) ที่เริ่มจากตำแหน่ง $(0,0)$ ของข้อมูลเข้าเลื่อนไปทีละจุดภาพจากซ้ายไปขวาและบนลงล่าง จนครอบคลุมข้อมูลเข้าทั้งหมด ตัวอย่างเช่น ข้อมูลเข้า [32x32x3] และตัวกรอง [5x5x3] เมื่อหาผลคูณจุดจะได้ข้อมูลภาพใหม่ [28x28x1] ที่มีความลึกเท่ากับ 1 ดังภาพ 17



ภาพ 17 ตัวอย่างของ Convolution Layer ที่ข้อมูลเข้า [32x32x3] และตัวกรอง [5x5x3]

การเลื่อนตำแหน่งของตัวกรองไปทีละจุดภาพ (stride 1) ของข้อมูลเข้า [32x32x3] ดังภาพ 17 เพื่อหาผลคูณจุด เมื่อนำตัวกรอง [5x5x3] เลื่อนจากจุดเริ่มต้น $(0,0)$ จากซ้ายไปขวาไปทีละจุดภาพ โดยไม่ซ้ำกัน จะได้ 28 ตำแหน่ง และจากบนลงล่างจะได้ 28 ตำแหน่ง เมื่อหาผลคูณจุดของแต่ละตำแหน่งของข้อมูลเข้ากับตัวกรอง จะได้ข้อมูลภาพใหม่ที่เรียกว่า activation map ที่มีขนาด [28x28x1] ดังนั้นหากมีการปรับค่า stride จะส่งผลให้ขนาดของ activation map เปลี่ยนไป เช่น ข้อมูลเข้า [32x32x3] และตัวกรอง [5x5x3] โดยเลื่อนตำแหน่งของตัวกรองไปทีละ 2 จุดภาพ

(stride 2) เมื่อทำการเลื่อนไปจนถึงตำแหน่งที่ 14 พบว่าไม่สามารถเลื่อนต่อให้ครอบคลุมข้อมูลเข้าได้ทั้งหมด หากต้องการตรวจสอบว่าขนาดของข้อมูลออกนั้นมีขนาดเท่าใด สามารถคำนวณได้จากสมการ 2.8 โดยที่ขนาดของข้อมูลเข้า (N) ขนาดของตัวกรอง (F) และจำนวนที่เลื่อนตัวกรอง (stride)

$$Output = ((N-F)/stride)+1 \quad (2.8)$$

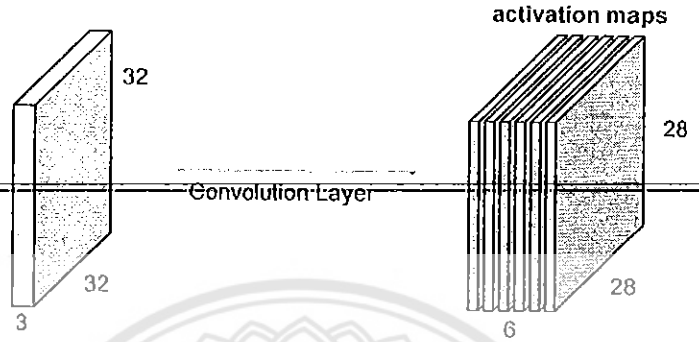
ในการตรวจสอบว่าหากใช้ข้อมูลเข้า [32x32x3] และตัวกรอง [5x5x3] โดย stride 1 ขนาดของข้อมูลออกเท่ากับ ((32-5)/1) + 1 = 28 และเมื่อเปลี่ยนเป็น stride 2 ขนาดที่ได้เท่ากับ ((32-5)/2) + 1 = 14.5 ดังนั้นในกรณีที่ข้อมูลเข้า [32x32x3] และตัวกรอง [5x5x3] จะไม่สามารถใช้ stride 2 ได้ เนื่องจากขนาดที่ได้ต้องเป็นจำนวนเต็มบวกเท่านั้น จึงจะครอบคลุมข้อมูลทั้งหมด

0	0	0	0	0	0	0	0	0
0	1	1	1	1	2	1	1	0
0	1	2	0	1	0	2	1	0
0	1	1	0	1	0	1	2	0
0	1	1	1	0	1	2	0	0
0	0	2	1	1	2	2	0	0
0	1	1	0	1	1	0	2	0
0	2	1	1	0	1	1	1	0
0	0	0	0	0	0	0	0	0

ภาพ 18 ตัวอย่างการเพิ่มข้อมูล 0 รอบ ๆ ข้อมูลเข้า [7x7] และตัวกรอง [3x3]

หากต้องการกำหนดขนาดของข้อมูลออก สามารถทำได้ด้วยการเพิ่มข้อมูล 0 รอบ ๆ ข้อมูลเข้าหรือเรียกว่า zero-padding ซึ่งช่วยให้สามารถควบคุมขนาดของข้อมูลออกได้ เช่น ในกรณีที่ต้องการกำหนดขนาดของข้อมูลออกให้มีขนาดเท่ากับขนาดของข้อมูลเข้า นอกจากช่วยให้สามารถควบคุมขนาดของข้อมูลออกได้แล้วนั้น การใช้ zero-padding ยังช่วยให้สามารถเพิ่มการวิเคราะห์ข้อมูลบริเวณขอบของภาพได้ละเอียดมากขึ้น ตัวอย่างการใช้ zero-padding กำหนดให้ ข้อมูลเข้ามีขนาด [7x7] จากนั้นเพิ่มข้อมูล 0 รอบ ๆ ข้อมูลด้วยขนาดเท่ากับ 1 ทำให้ข้อมูลเข้ามีขนาดเพิ่มขึ้นเป็น [9x9] ดังภาพ 22 เมื่อนำตัวกรอง [3x3] มา stride 1 ทำให้ขนาดของข้อมูลออกเท่ากับ ((9-3)/1) + 1 = 7 หรือ [7x7] ซึ่งเท่ากับข้อมูลเข้าก่อนเพิ่ม 0 เข้ามาเป็นกรอบ

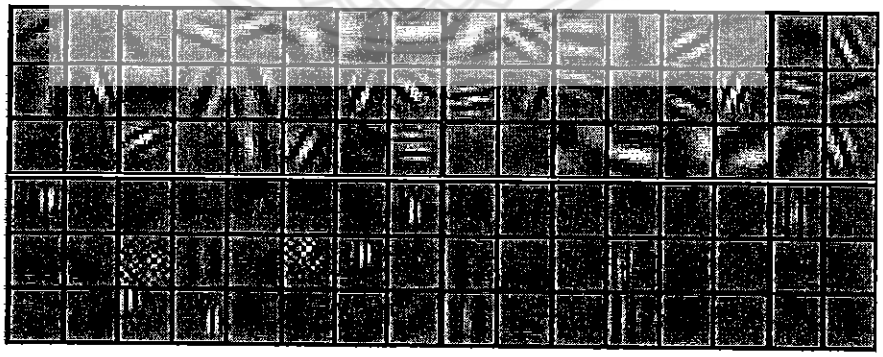
หากต้องการให้ความลึกของข้อมูลออก (ภาพใหม่ที่เกิดขึ้น) มีมากขึ้น สามารถทำได้ด้วยการเพิ่มจำนวนของตัวกรอง เช่น เมื่อนำ 6 ตัวกรอง [5x5x3] มา Convolution กับข้อมูลนำเข้า [32x32x3] ทำให้ได้ข้อมูลภาพใหม่ [28x28x6] ที่มีความลึกเท่ากับตัวกรอง ดังภาพ 19



ภาพ 19 ตัวอย่างของ Convolution Layer ที่มีข้อมูลเข้า [32x32x3] และ 6 ตัวกรอง [5x5x3]

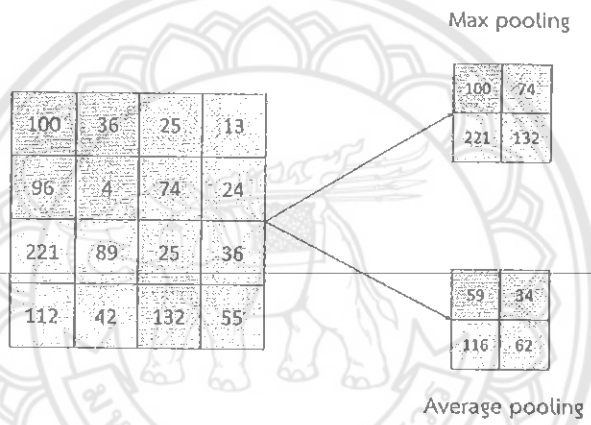
ที่มา: Fei-Fei Li, Andrej Karpathy และ Justin Johnson. Stanford university

Convolution Layer ต้องใช้ตัวกรองในการเปลี่ยนความลึกของภาพจาก 3 มิติ (ข้อมูลเข้า) ให้มีความลึกเท่ากับจำนวนของตัวกรองที่ใช้ จากภาพ 20 ตัวอย่างของ 96 ตัวกรองสำหรับการเรียนรู้ในแบบจำลองโครงข่ายของ AlexNet [20] โดยแต่ละตัวกรองมีขนาด [11x11x3] โดยใช้ stride 4 และนำไปใช้ในลำดับชั้นแรกของโครงข่าย เพื่อแปลงภาพนำเข้าที่มีขนาด [227x227x3] ให้มีขนาดที่นำออก ((227-11)/4) +1 = 55 ดังนั้นข้อมูลออกของลำดับชั้นนี้มีขนาด [55x55x96]



ภาพ 20 ตัวอย่าง 96 ตัวกรองของ AlexNet แต่ละตัวกรองมีขนาด 11x11x3 [20]

Pooling Layer คือลำดับชั้นที่ถูกนำมาใช้ถัดจาก Convolution Layer โดยทำหน้าที่ลดขนาดของข้อมูลตัวแทน (representation) เพื่อลดปริมาณของพารามิเตอร์และการคำนวณในลำดับชั้นถัดไปโดยสามารถควบคุมการคำนวณให้ครอบคลุมข้อมูลได้ ฟังก์ชันที่ถูกนำมาใช้สำหรับลดขนาดของข้อมูลให้มีขนาดที่เล็กลงคือ Max pooling ที่ใช้ตัวกรอง [2x2] และ stride = 2 เมื่อนำมาใช้กับข้อมูลเข้าทำให้ขนาดของข้อมูลทั้งความกว้างและความสูงลดลงครึ่งหนึ่ง โดยที่ข้อมูลเข้าจะยังคงข้อมูลเดิมอยู่ 25 % เมื่อใช้คำสั่ง Max ในการหาข้อมูลมากที่สุดของข้อมูลย่อยที่ถูกแบ่งให้มีขนาด 2x2 จำนวน 4 ส่วน ดังภาพ 21 ที่มีการใช้ทั้งฟังก์ชัน Max และ Average ซึ่งเมื่อใช้คำสั่ง Average จะทำการหาค่าเฉลี่ยของข้อมูลย่อยที่ถูกแบ่งขนาด 2x2 จำนวน 4 ส่วน ดังนั้นข้อมูลใหม่ที่ได้จะเป็นค่าเฉลี่ยของข้อมูลนำเข้า



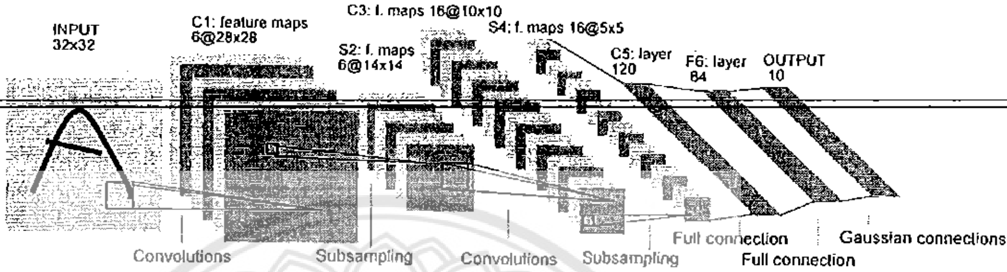
ภาพ 21 Max pooling กับ Average pooling โดยใช้ตัวกรอง [2x2] และ stride = 2

Fully-Connected Layer คือลำดับชั้นที่เชื่อมต่อกับเซลล์ประสาททั้งหมดของฟังก์ชันกระตุ้นในลำดับชั้นก่อนหน้า เพื่อปรับขนาดของข้อมูลให้มีเพียงหนึ่งมิติ (one-dimensional) ดังนั้นจึงทำให้ไม่สามารถใช้ Convolution Layer ถัดจาก Fully-Connected Layer และแบบจำลองของโครงข่ายประสาทเทียมแบบสังวัตนาการใช้ Fully-Connected Layer เป็นลำดับชั้นท้ายสุดของโครงข่าย

แบบจำลองของโครงข่ายประสาทเทียมแบบสังวัตนาการ (ConvNets models)

แบบจำลองของโครงข่ายประสาทเทียมแบบสังวัตนาการ คือการนำแต่ละลำดับชั้นมาเรียงต่อกันตามรูปแบบที่ถูกออกแบบ โดยแบบจำลองที่ได้รับความนิยมส่วนมากมาจากการแข่งขัน The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) ที่ชนะในการแข่งขันในแต่ละปี ดังนี้

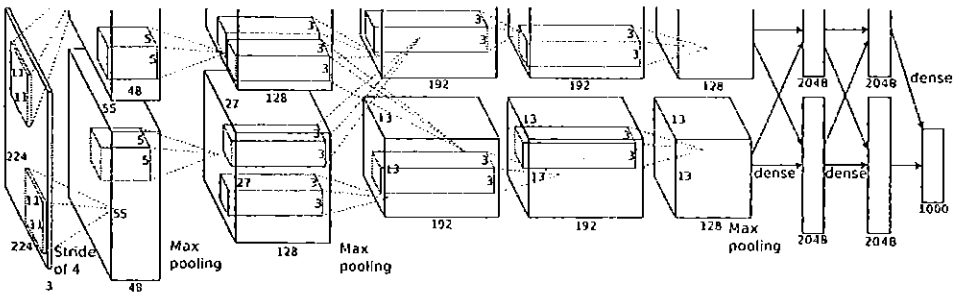
LeNet [21] ได้ถูกพัฒนาขึ้นในปี 1998 โดย Yann LeCun ซึ่งถือว่าเป็นแบบจำลองแรกของโครงข่ายประสาทเทียมแบบสังวัตนาการที่ถูกพัฒนาเพื่อใช้สำหรับแอพพลิเคชั่นการรู้จำตัวอักษรและตัวเลข โดยมีโครงสร้างดังนี้ INPUT -> [CONV -> POOL]*2 -> FC -> FC



ภาพ 22 แบบจำลอง LeNet [21]

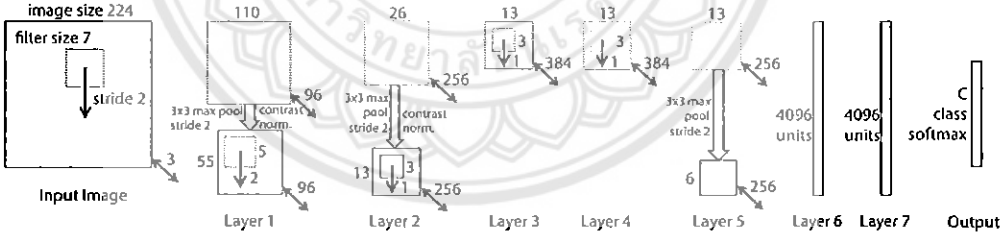
จากภาพ 22 แบบจำลอง LeNet ใช้สำหรับแอพพลิเคชั่นการรู้จำตัวอักษรและตัวเลขทำให้ลำดับชั้น Input มีขนาด [32x32x1] ซึ่งเมื่อเข้าสู่ลำดับชั้น Convolution (C1) โดยใช้ 6 ตัวกรอง [5x5x6] ทำให้ได้ข้อมูล Feature maps [28x28x6] จากนั้นทำการลดขนาดของข้อมูลลงด้วยลำดับชั้น Pooling (S2) ทำให้ได้ขนาด [14x14x6] แล้วส่งข้อมูลเข้าสู่ลำดับชั้น Convolution (C2) โดยใช้ 16 ตัวกรอง [5x5x16] ทำให้ได้ข้อมูล Feature maps [10x10x16] จากนั้นทำการลดขนาดของข้อมูลลงอีกครั้งด้วยลำดับชั้น Pooling (S4) ทำให้ได้ขนาด [5x5x16] จากนั้นเข้าสู่ลำดับชั้น Fully-Connected แล้วจะได้ Output ที่มีขนาดเท่ากับ 10

AlexNet [20] ถูกพัฒนาขึ้นโดย Alex Krizhevsky และคณะ ซึ่ง AlexNet เป็นแบบจำลองที่ได้รับความนิยมอย่างมากสำหรับโครงข่ายประสาทเทียมแบบสังวัตนาการ เนื่องจากแบบจำลอง AlexNet ชนะการแข่งขัน ILSVRC ในปี 2012 โดยเป็นแบบจำลองแรกที่สามารถนำภาพในแบบจำลองสี RGB มาใช้โดยอาศัยหน่วยประมวลผลกราฟิกส์ (GPU) ในการประมวลผล เพื่อลดเวลาในการประมวลผลในหนึ่งรอบของการเรียนรู้ ทำให้จำนวนรอบที่เรียนรู้สามารถประมวลผลได้มากขึ้น โครงข่ายของ AlexNet มีลำดับชั้นดังภาพ 23 ที่สามารถเรียนรู้ได้ทั้งครอบคลุมทั้งเชิงลึก ขนาด และสีของภาพที่นำมาเรียนรู้



ภาพ 23 แบบจำลอง AlexNet [20]

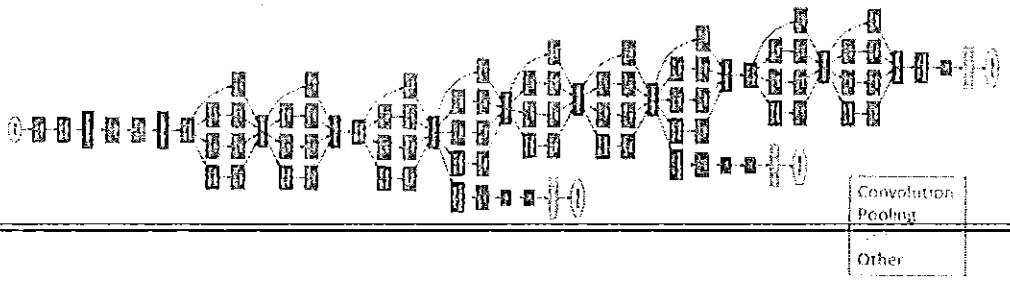
ZFNet [22] ถูกพัฒนาขึ้นโดย Matthew Zeiler และ Rob Fergus หรือที่รู้จักในชื่อ ZFNet (Zeiler & Fergus Net) ที่ชนะการแข่งขัน ILSVRC ในปี 2013 โดย ZFNet เป็นแบบจำลองที่นำแบบจำลอง AlexNet มาปรับปรุงใหม่ ซึ่งได้มีการปรับเปลี่ยนตัวกรองและ stride ให้เล็กลงจาก AlexNet (CONV1) ตัวกรอง [11x11] และ stride = 4 ให้เป็น ZFNet (CONV1) [7x7] และ stride = 2 และยังได้ปรับจำนวนของตัวกรองให้มีขนาดใหญ่ขึ้นใน (CONV3, CONV4, CONV5) โดยเปลี่ยนจาก 384, 384, 256 ตัวกรองให้เป็น 512, 1024, 512 ตัวกรอง ตามลำดับ ดังภาพ 24



ภาพ 24 แบบจำลอง ZFNet [22]

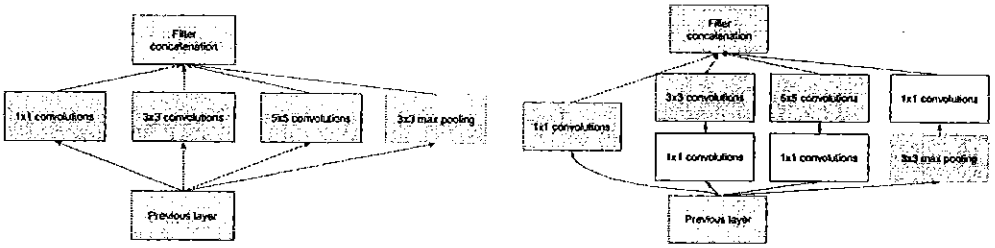
GoogLeNet [23] ถูกพัฒนาขึ้นจาก Christian Szegedy และคณะ โดยทำงานที่บริษัท Google และชนะการแข่งขัน ILSVRC ในปี 2014 แบบจำลอง GoogLeNet พัฒนามาจากแนวคิด Inception Module ที่ให้โครงข่ายมีความลึกที่ซ้อนความลึกอีกทีดังภาพ 25 เพื่อลดจำนวนพารามิเตอร์ในโครงข่ายทำให้ขนาดของแบบจำลองมีขนาดน้อยกว่า AlexNet ด้วยการเปลี่ยนมาใช้

- Average pooling ในลำดับชั้น Fully Connected เพื่อลดพารามิเตอร์ที่ไม่มีความสำคัญออก จึงทำให้แบบจำลอง GoogLeNet มีขนาดที่เล็กลง



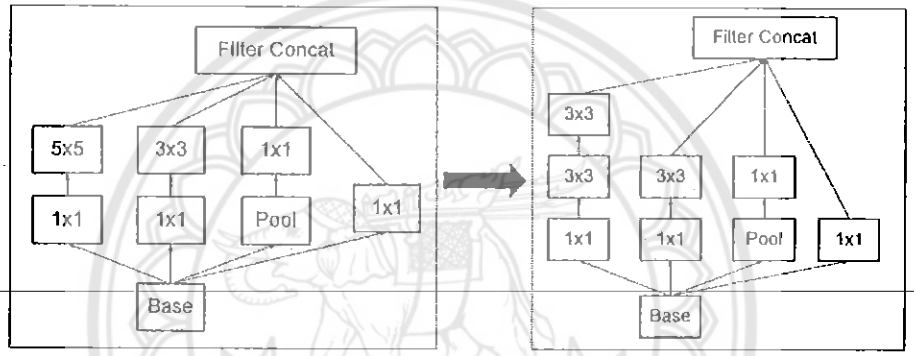
ภาพ 25 แบบจำลอง GoogLeNet [23]

Inception Module คือแนวคิดที่เกิดจากการนำลำดับชั้น Convolution ทั้งขนาด 1x1, 3x3, 5x5 และลำดับชั้น Pooling 3x3 มาคำนวณจากข้อมูลที่ได้จากลำดับชั้นก่อนหน้า (Previous layer) จากนั้นนำข้อมูลที่ได้ในแต่ละลำดับชั้นมาเรียงต่อกัน (Concatenation) ดังภาพ 26 (ซ้าย) ทำให้ได้ข้อมูลจากการคำนวณที่มีความลึกมากขึ้น แต่จะทำให้เกิดพารามิเตอร์ที่เพิ่มสูงขึ้นในลำดับชั้น Convolution ขนาด 3x3 กับ 5x5 โดยพารามิเตอร์ของลำดับชั้นคือ (ความลึกของลำดับชั้นก่อนหน้า) x (จำนวนของตัวกรอง) x (ขนาดของตัวกรอง) หากต้องการลดจำนวนของพารามิเตอร์ลงสามารถทำได้ด้วยการเพิ่มลำดับชั้น Convolution ขนาด 1x1 เข้ามาคำนวณก่อนที่จะใช้ลำดับชั้น Convolution ขนาด 3x3 กับ 5x5 และเพิ่มลำดับชั้น Convolution ขนาด 1x1 เข้ามาหลังจากลำดับชั้น Pooling 3x3 เพื่อลดจำนวนของข้อมูลก่อนที่จะส่งต่อไปยังลำดับชั้นถัดไป ดังภาพ 26 (ขวา)



ภาพ 26 Inception Module แบบเดิม (ซ้าย) Inception Module แบบลดขนาด (ขวา) [23]

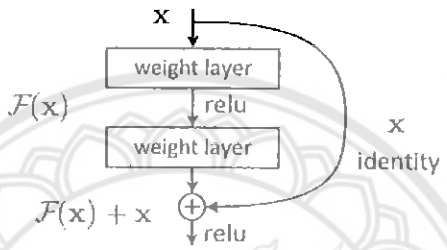
Inception Module ได้ถูกพัฒนามาอย่างต่อเนื่อง โดย Inception-v2 [24] ได้มีการเพิ่ม Batch Normalization ให้กับทุกลำดับชั้นของ Inception-v1 เพื่อช่วยให้การฝึกฝนข้อมูลมีความถูกต้องเพิ่มมากขึ้น จนถึง Inception-v3 [25] ได้มีการปรับ Inception Module ใหม่ โดยเปลี่ยนลำดับชั้น Convolution ขนาด 5x5 ให้เป็นสองลำดับชั้น Convolution ขนาด 3x3 แทนดังภาพ 27 เพื่อลดการประมวลผลให้น้อยลง เนื่องจากตัวกรองที่มีขนาด 5x5 จะใช้เวลาในการประมวลผลสูงกว่าตัวกรองที่มีขนาด 3x3 ในกรณีที่มีจำนวนของตัวกรองเท่ากัน



ภาพ 27 Inception Module (รุ่นที่ 1, ซ้าย) Inception Module (รุ่นที่ 3, ขวา) [25]

Batch Normalization [24] หนึ่งในเทคนิคที่พัฒนาจากบริษัท Google ที่ช่วยเร่งความเร็วในการฝึกฝนของโครงข่ายประสาทเทียม ด้วยการเปลี่ยนแปลงค่าเฉลี่ย (mean) ให้เท่ากับ 0 และความแปรปรวน (variance) เท่ากับ 1 สำหรับ non-linearity input (เช่น Sigmoid หรือ ReLU) เพื่อลดปัญหา Internal covariate shift ที่เกิดขึ้นจากการกระจายข้อมูล (input distribution) ระหว่างชุดข้อมูลฝึกกับข้อมูลทดสอบ สำหรับการออกแบบโครงข่ายประสาทเทียมจะทำการแทรก ลำดับชั้น Batch Normalization ระหว่างลำดับชั้น Convolutions และลำดับชั้น Activations เพื่อเปลี่ยนแปลงข้อมูลที่ได้จากลำดับชั้น Convolutions ให้เป็นบรรทัดฐานเดียวกัน ก่อนที่จะส่งไปยังลำดับชั้น Activations การเพิ่มลำดับชั้น Batch Normalization ช่วยให้ประสิทธิภาพด้านความถูกต้องและความเร็วในการลู่เข้า (Convergence) สำหรับกระบวนการฝึกฝนของโครงข่ายเพิ่มขึ้น

Residual Network [26, 27] ถูกพัฒนาขึ้นโดย Kaiming He และคณะ ที่ชนะการแข่งขัน ILSVRC ในปี 2015 โดยได้นำเสนอเทคนิค Identity mapping ที่เพิ่ม shortcut connections จากลำดับชั้นก่อนหน้า (input) มารวมกับข้อมูลที่ได้จากลำดับชั้นถัดไป (output) เพื่อปรับให้ข้อมูลในลำดับชั้นถัดไปมีค่าเท่ากับ (output + input) และปรับขนาดให้มีมิติ (dimension) เท่ากัน จึงทำให้ยังคงรักษาข้อมูล (Residual Representations) จากลำดับชั้นก่อนหน้าได้ ดังภาพ 28 ซึ่งวิธีการนี้ช่วยแก้ปัญหาที่มีผลต่อการเพิ่มประสิทธิภาพความถูกต้องของกระบวนการเรียนรู้สำหรับโครงข่ายประสาทเทียมที่มีความลึกจำนวนมาก เช่น Deep Convolutional Neural Network (DCNN)



ภาพ 28 Residual learning : 1 กลุ่ม [26]

จากเอกสารและงานวิจัยที่เกี่ยวข้องที่ได้อธิบายไว้ข้างต้น ได้แก่ ภาพดิจิทัล แบบจำลองสี การเรียนรู้ของเครื่อง การสกัดคุณลักษณะ การจำแนกภาพ การจำแนกเชิงเส้น ฟังก์ชันความสูญเสียสำหรับการจำแนก ฟังก์ชันซอฟต์แม็กซ์ โครงข่ายประสาทเทียมแบบลึก ลำดับชั้นของโครงข่ายประสาทเทียมแบบสังวัตนาการ และจำลองของโครงข่ายประสาทเทียมแบบสังวัตนาการ เป็นหนึ่งในจุดมุ่งหมายของการทำงานวิจัยชิ้นนี้ ในหัวข้อศึกษาข้อมูล เพื่อให้เข้าใจทั้งหลักการและทฤษฎีต่าง ๆ สำหรับการศึกษาข้อมูลดังกล่าวคือการนำเสนอหลักการที่สำคัญ โดยสามารถศึกษาเพิ่มเติมได้จากแหล่งอ้างอิง ในส่วนของวิธีดำเนินงานวิจัยได้อธิบายไว้ในบทถัดไป

บทที่ 3
วิธีดำเนินงานวิจัย

ในบทนี้เป็นการนำเอกสารและงานวิจัยที่เกี่ยวข้องที่ได้ศึกษามาใช้ในขั้นตอนวิธีดำเนินงานวิจัย โดยแบ่งการดำเนินงานออกเป็น 3 ส่วน ดังนี้ การเก็บและเตรียมข้อมูลภาพอาหารไทย การเรียนรู้หรือฝึกฝนโครงข่ายประสาทเทียมแบบสังวัตนาการ และการประเมินค่าของโครงข่ายประสาทเทียมแบบสังวัตนาการ

การเก็บและเตรียมข้อมูลภาพอาหารไทย (THFOOD-50)

เก็บรวบรวมภาพถ่ายอาหารไทยที่ได้รับความนิยม 50 ประเภท ด้วยโปรแกรมค้นหา (search engine) จาก Google, Bing, และ Flickr ซึ่งแต่ละประเภทมีจำนวนประมาณ 200 ถึง 700 ภาพ และได้นำข้อมูลภาพถ่ายทั้งหมดแบ่งออกเป็นสองส่วนคือชุดข้อมูลฝึก (Training data) กับชุดข้อมูลทดสอบ (Test data) ซึ่งแบ่งข้อมูลตามสัดส่วน 90% : 10% และแบ่งข้อมูลตามสัดส่วน 50% : 50% โดยปรับขนาดของภาพถ่ายให้มีความกว้าง 256 จุดภาพ (Pixel) และความสูง 256 จุดภาพ (Pixel) เพื่อให้มีขนาดเท่ากับลำดับชั้นข้อมูลของแบบจำลองของโครงข่ายประสาทเทียมแบบสังวัตนาการ

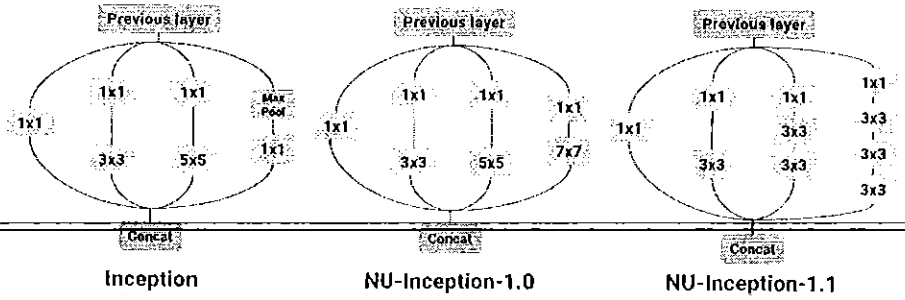
การเรียนรู้หรือฝึกฝนโครงข่ายประสาทเทียมแบบสังวัตนาการ (Learning)

การเรียนรู้หรือฝึกฝนโครงข่ายประสาทเทียมแบบสังวัตนาการ สามารถนำแบบจำลองที่ได้รับความนิยมในการแข่งขัน The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) เช่น แบบจำลอง AlexNet, GoogLeNet และ ResNet มาฝึกฝนให้กับโครงข่ายประสาทเทียมเพื่อเรียนรู้ภาพถ่ายอาหารไทย ด้วยการนำชุดข้อมูลฝึกของภาพถ่ายอาหารแต่ละประเภทมาเรียนรู้เพื่อหาข้อมูลที่ใช้แทนกลุ่มหรือประเภท

การออกแบบแบบจำลองของโครงข่ายประสาทเทียมแบบสังวัตนาการ สำหรับการรู้จำภาพถ่ายอาหารไทยได้ออกแบบและพัฒนาแบบจำลองจำนวน 2 รุ่น โดยได้นำแนวคิด Inception Module ของ GoogLeNet ดังภาพ 29 (ซ้าย) มาปรับเปลี่ยนดังนี้

NU-Inception 1.0 ปรับเปลี่ยนจาก 3x3 Max pooling กับ 1x1 Convolutions ให้เป็น 1x1 Convolutions กับ 7x7 Convolutions ตามลำดับ เพื่อนำตัวกรองที่มีขนาด 7x7 มาใช้ในแนวคิด Inception Module ดังภาพ 29 (กลาง)

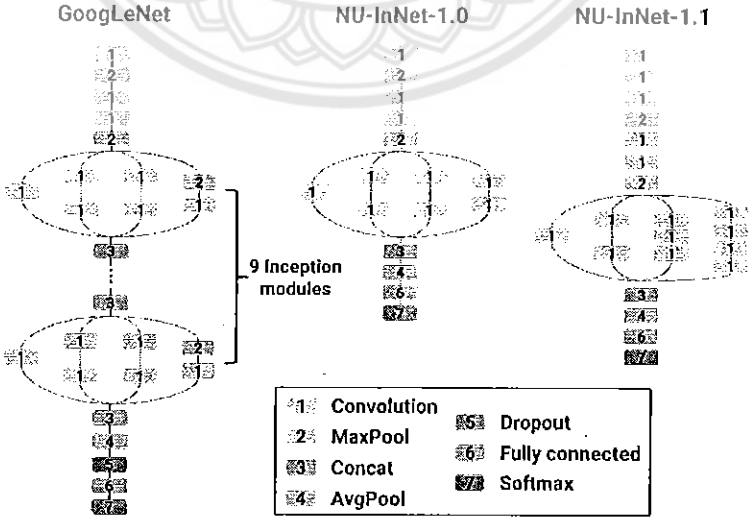
NU-Inception 1.1 ได้ทำการปรับเปลี่ยนจาก NU-Inception 1.0 โดยเปลี่ยนลำดับชั้น Convolutions ที่มีขนาดมากกว่า 3x3 ให้เป็น double 3x3 แทน 5x5 และ triple 3x3 แทน 7x7 ดังภาพ 33 (ขวา) เพื่อใช้ตัวกรองของ Convolutions เฉพาะขนาด 1x1 และ 3x3 เท่านั้น



ภาพ 29 Inception Module (ซ้าย) NU-Inception 1.0 Module (กลาง) และ NU-Inception 1.1 Module (ขวา)

1. NU-InNet

แบบจำลองที่พัฒนาขึ้นถูกนำมาใช้ในการรู้จำภาพถ่ายอาหารไทยที่ใช้ฐานข้อมูล THFOOD-50 โดยเฉพาะ และได้ตั้งชื่อแบบจำลองดังกล่าว "NU-InNet" ซึ่งย่อมาจาก Naresuan University Inception Network โดยการออกแบบแบบจำลองได้คำนึงถึงหน่วยความจำของสมาร์ทโฟนเป็นหลัก จึงทำให้แบบจำลองดังกล่าวมีความลึกเพียงไม่กี่ลำดับขั้นเมื่อเปรียบเทียบกับแบบจำลองอื่น ๆ เพื่อลดปริมาณการใช้หน่วยความจำ และเพื่อให้ได้ผลความถูกต้องในการรู้จำที่ดี จึงได้นำ NU-Inception ที่ปรับเปลี่ยนมาจากแนวคิด Inception Module มาใช้ใน NU-InNet



ภาพ 30 GoogLeNet (ซ้าย) NU-InNet 1.0 (กลาง) และ NU-InNet 1.1 (ขวา)

แบบจำลอง NU-InNet 1.0 ได้นำ NU-Inception 1.0 ที่ได้เพิ่ม 1x1 และ 7x7 Convolutions เข้ามา โดยได้กำหนดค่าน้ำหนักเริ่มต้น (weights initialization) ด้วย "Xavier" ที่มีค่าความโน้มเอียง (bias) เท่ากับ 0.2 ในรูปแบบค่าคงที่ (constant) และกำหนดให้การลดขนาด (down sampling) ของข้อมูลใช้ Max pooling ที่มีตัวกรอง 3x3 และ stride เท่ากับ 2 เพื่อลดขนาดของข้อมูลให้เหลือครึ่งหนึ่ง และส่วนท้ายของแบบจำลองใช้ Average pooling โดยรายละเอียดของแบบจำลอง NU-InNet 1.0 อยู่ในตาราง 1 และเมื่อนำไปแสดงในรูปแบบกราฟจะได้ดังภาพ 30 (กลาง)

ตาราง 1 แบบจำลอง NU-InNet 1.0

Type	Patch size / Stride	Output size
Input image	-	224x224x3
Convolution	7x7/2	109x109x96
Max pool	3x3/2	54x54x96
Convolution	1x1/1	54x54x96
Convolution	5x5/2	25x25x96
Max pool	3x3/2	12x12x96
1 x NU-Inception-1.0	จากภาพ 30 (กลาง)	12x12x256
Average pool	-	1x1x256
Fully Connected	-	1x1x50

แบบจำลอง NU-InNet 1.1 ได้นำ NU-Inception 1.1 ที่พัฒนามาจาก NU-Inception 1.0 ด้วยการใช้ตัวกรองเฉพาะขนาด 1x1 และ 3x3 Convolutions เท่านั้นในแบบจำลอง ซึ่งได้เปลี่ยนจาก 5x5 Convolutions เป็น double 3x3 Convolutions และ 7x7 Convolutions เป็น triple 3x3 Convolutions และกำหนดค่าน้ำหนักเริ่มต้น (weights initialization) ด้วย "Xavier" ที่มีค่าความโน้มเอียง (bias) เท่ากับ 0.2 ในรูปแบบค่าคงที่ (constant) และกำหนดให้การลดขนาด (down sampling) ของข้อมูลใช้ Max pooling ที่มีตัวกรอง 3x3 และ stride เท่ากับ 2 เพื่อลดขนาดของข้อมูลให้เหลือครึ่งหนึ่ง และส่วนท้ายของแบบจำลองใช้ Average pooling โดยรายละเอียดของแบบจำลอง NU-InNet 1.1 อยู่ในตาราง 2 และเมื่อนำไปแสดงในรูปแบบกราฟจะได้ดังภาพ 30 (ขวา)

ตาราง 2 แบบจำลอง NU-InNet 1.1

Type	Patch size / Stride	Zero padding	Output size
Input image	-	-	224x224x3
Convolution	3x3/2	-	111x111x32
Convolution	3x3/1	-	109x109x32
Convolution	3x3/1	1	109x109x64
Max pool	3x3/2	-	54x54x64
Convolution	1x1/2	-	27x27x96
Convolution	3x3/1	-	25x25x96
Convolution	3x3/1	1	25x25x96
Max pool	3x3/2	-	12x12x96
1x NU-Inception-1.1	จากภาพ 30 (ขวา)	-	12x12x256
Average pool	-	-	1x1x256
Fully Connected	-	-	1x1x50

ในการออกแบบโครงข่ายประสาทเทียมทั้งหมดต้องพิจารณาถึงความเร็วในการประมวลผลและขนาดของแบบจำลองให้เหมาะสมกับอุปกรณ์สมาร์ทโฟน จึงต้องออกแบบโครงข่ายที่มีประสิทธิภาพในการรู้จำทัดเทียมกับ GoogLeNet แต่เวลาในการประมวลผลและขนาดของแบบจำลองต้องน้อยกว่า GoogLeNet จึงต้องทำการตัดความลึกของแบบจำลอง NU-InNet ลง ให้เหลือเพียง 1 Module (GoogLeNet ใช้ 9 Module) ดังภาพ 30 ซึ่งจะช่วยลดเวลาในการประมวลผลและขนาดของแบบจำลองลง แต่ประสิทธิภาพในการรู้จำจะลดลงด้วยเช่นกัน ดังนั้นหากต้องการเพิ่มประสิทธิภาพในการรู้จำสามารถใช้ Batch Normalization (BN) [24] หลัง Convolutions ทุกลำดับชั้นเช่นเดียวกับ ResNet [26] ซึ่งสามารถช่วยเพิ่มค่าความถูกต้องของการเรียนรู้ได้

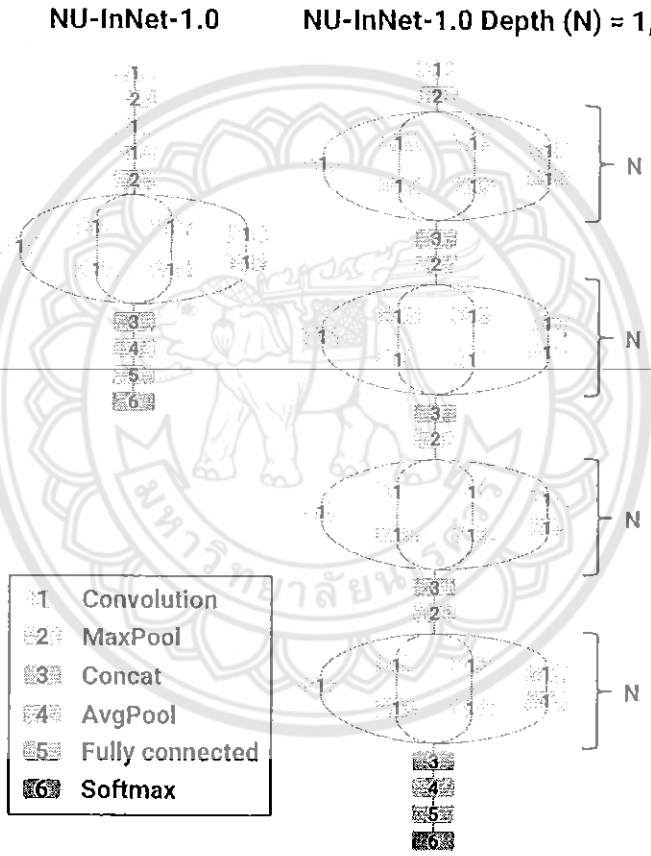
2. Deep NU-InNet

ในการออกแบบสถาปัตยกรรมของโครงข่ายประสาทเทียมให้มีประสิทธิภาพในด้านความถูกต้องของ NU-InNet 1.0/1.1 แม่นยำมากขึ้น โดยที่ขนาดของแบบจำลองยังคงเหมาะสมและสามารถนำไปใช้งานบนอุปกรณ์สมาร์ทโฟนได้ โดยได้นำ NU-InNet 1.0/1.1 มาเพิ่มจำนวนของ NU-Inception Module ในแต่ละระดับ (scale) ของภาพ [56×56, 28×28, 14×14, 7×7] เพื่อวิเคราะห์ข้อมูลภาพถ่ายอย่างละเอียดมากขึ้น ด้วยการใช้ NU-Inception 1.0/1.1 ในแต่ละระดับของภาพ โดยกำหนดให้ความลึกของแบบจำลองมีระดับความลึก (Depth) เท่ากับ 4, 8, 12 (N=1, 2, 3) ตัวอย่างเช่น ความลึกที่ 8 (N=2) ประกอบไปด้วย NU-Inception 1.0/1.1 จำนวน 4 Block (แต่ละ Block มี NU-Inception 1.0/1.1 จำนวน 2 Module ต่อกันแบบอนุกรม) และนำแต่ละ Block มาเรียงทับซ้อน (stack) กันโดยมี Max Pooling layer คั่นกลางระหว่างแต่ละ Block โดยที่กำหนดให้ตัวกรอง 3×3 และ stride เท่ากับ 2 ทำหน้าที่ลดขนาดของข้อมูลให้เล็กลงครึ่งหนึ่งและส่งข้อมูลต่อไปยัง Block ต่อไป

ตาราง 3 แบบจำลอง Deep NU-InNet ความลึกเท่ากับ 4, 8, 12 (N=1, 2, 3)

Type	Patch Size / Stride	Output size
Input Image	-	224×224×3
Convolution	5×5/2	113×113×64
Max Pool	3×3/2	56×56×64
NU-Inception × N	จากภาพ 29	56×56×64
Max Pool	3×3/2	28×28×64
NU-Inception × N	จากภาพ 29	28×28×128
Max Pool	3×3/2	14×14×128
NU-Inception × N	จากภาพ 29	14×14×256
Max Pool	3×3/2	7×7×256
NU-Inception × N	จากภาพ 29	7×7×512
Average Pool	-	1×1×512
Fully Connected	-	1×1×50

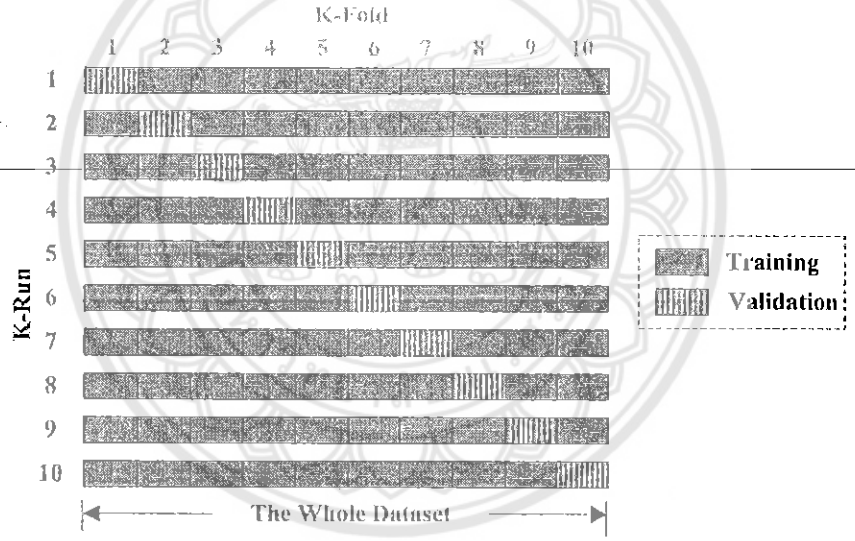
นอกจากการเพิ่มจำนวนของ NU-Inception 1.0/1.1 เพื่อให้โครงข่ายสามารถวิเคราะห์ข้อมูลภาพได้มากขึ้นแล้วยังได้ทำการเปลี่ยนมาใช้ MSRA Initialization [28] แทน Xavier Initialization และเพิ่มลำดับชั้น Batch Normalization หลังลำดับชั้น Convolutions ทุกลำดับชั้นเช่นเดียวกับ BN-Inception [24] เพื่อช่วยให้ประสิทธิภาพการฝึกฝนสู่เข้าความถูกต้องได้เร็วขึ้นและให้ผลความถูกต้องแม่นยำเพิ่มมากขึ้น โดยรายละเอียดของสถาปัตยกรรมในงานวิจัยชิ้นนี้อยู่ในตาราง 3 และเมื่อนำไปแสดงในรูปแบบกราฟจะได้ดังภาพ 31



ภาพ 31 NU-InNet 1.0 (ซ้าย) และ Deep NU-InNet 1.0 Depth (N) = 1,2,3 (ขวา)

การประเมินค่าของโครงข่ายประสาทเทียม (Evaluation)

การนำแบบจำลองที่ฝึกฝนมาใช้ประเมินค่าของความถูกต้อง โดยนำข้อมูลที่ใช้สำหรับทดสอบ (Testing set) มาทดสอบกับแบบจำลองเพื่อระบุว่าข้อมูลดังกล่าวอยู่ในกลุ่มใด ถ้าหากมีการระบุกลุ่มที่ถูกต้องจะทำให้ค่าความถูกต้องของแบบจำลองมีค่ามากขึ้น ซึ่งในการประเมินค่าดังกล่าวได้ใช้ข้อมูลที่ถูกแบ่งออกเป็นข้อมูลสำหรับทดสอบ โดยข้อมูลดังกล่าวไม่ได้ถูกนำไปเรียนรู้ แต่ถ้าหากต้องการความถูกต้องของแบบจำลองที่ฝึกฝนด้วยฐานข้อมูลภาพถ่ายอาหารไทย ซึ่งสามารถใช้ K-fold cross-validation ในการตรวจสอบโดยจะทำการแบ่งข้อมูลภาพออกเป็น 10 ส่วนเท่าๆ กัน จากนั้นนำข้อมูล 9 ส่วน ไปฝึกฝนให้กับโครงข่ายประสาทเทียม แล้วนำ 1 ส่วนที่เหลือมาใช้สำหรับทดสอบแบบจำลองเพื่อหาค่าความถูกต้อง โดยจะทำการเปลี่ยนข้อมูลสำหรับทดสอบไปทีละส่วน จนกระทั่งครบทั้ง 10 รอบ ดังภาพ 32 จากนั้นนำค่าความถูกต้องของแต่ละรอบในการฝึกฝนมาหาค่าเฉลี่ย เพื่อได้ค่าความถูกต้องของแบบจำลองที่ฝึกฝนด้วยฐานข้อมูลภาพถ่ายอาหารไทย



ภาพ 32 K-fold cross-validation

ที่มา: Zhang, Yudong, and Shuihua Wang. "Detection of Alzheimer's disease by displacement field and machine learning." PeerJ 3 (2015): e1251.

จากขั้นตอนการดำเนินงานวิจัยทั้ง 3 ส่วนข้างต้น ได้อธิบายรายละเอียดของการเก็บและเตรียมข้อมูลภาพอาหารไทยเพื่อนำมาสร้างชุดข้อมูล รวมไปถึงการออกแบบแบบจำลองเพื่อใช้การเรียนรู้หรือฝึกฝนของโครงข่ายประสาทเทียมแบบสังวัตนาการ และการประเมินค่าความถูกต้องของโครงข่ายประสาทเทียมแบบสังวัตนาการได้ผ่านกระบวนการเรียนรู้มาแล้ว

บทที่ 4
ผลการทดลอง

ในบทนี้จะกล่าวถึงการทดลองต่าง ๆ หลังจากที่ออกแบบและพัฒนาแบบจำลอง เพื่อทดลองและเก็บรวบรวมข้อมูลไว้โดยเริ่มตั้งแต่ การทดลองระหว่างแบบจำลอง NU-InNet กับแบบจำลองที่ชนะการแข่งขัน ILSVRC การทดลองระหว่างแบบจำลอง Deep NU-InNet กับแบบจำลอง GoogLeNet การทดลองแบบจำลอง NU-InNet กับ Deep NU-InNet บนสมาร์ทโฟน การทดลองของแบบจำลอง NU-InNet ที่ใช้ชุดข้อมูลฝึก 50% กับชุดข้อมูลทดสอบ 50% การทดลองของแบบจำลอง NU-InNet กับ SqueezeNet และการวิเคราะห์ข้อผิดพลาดของแบบจำลอง NU-InNet

การทดลองระหว่างแบบจำลอง NU-InNet กับแบบจำลองที่ชนะการแข่งขัน ILSVRC

ในกระบวนการฝึกฝนได้นำแบบจำลองที่ได้รับความนิยมจากการแข่งขัน ILSVRC ซึ่งประกอบไปด้วยแบบจำลอง AlexNet (ILSVRC 2012) และ GoogLeNet (ILSVRC 2014) มาเปรียบเทียบกับแบบจำลอง NU-InNet 1.0 และ NU-InNet 1.1 ที่พัฒนาขึ้น ด้วยวิธีการเรียนรู้แบบ From scratch โดยแบ่งสัดส่วนข้อมูลของ THFOOD-50 dataset เป็นชุดข้อมูลฝึกฝน (training data) จำนวน 90 % และชุดข้อมูลทดสอบ (test data) จำนวน 10% โดยใช้ 10 fold cross-validation เพื่อตรวจสอบค่าความถูกต้องของแต่ละแบบจำลอง ทำการประมวลผลบนเครื่องคอมพิวเตอร์ประสิทธิภาพสูง (High-performance computer) โดยมีองค์ประกอบด้านฮาร์ดแวร์ดังนี้ หน่วยประมวลผลหลัก (CPU) Intel(R) Xeon(R) E5-2683 v3 @ 2.00GHz จำนวน 56 แกนหลัก (Core) หน่วยความจำหลัก (RAM) จำนวน 64 จิกะไบต์ (GB) และหน่วยประมวลผลกราฟิกส์ (GPU) NVIDIA Tesla K80 ทำงานบนระบบปฏิบัติการ Ubuntu Server 14.04.5 ที่ติดตั้ง Caffe [29] ซึ่งเป็น Framework ที่ออกแบบมาสำหรับงานด้านโครงข่ายประสาทเทียมแบบสังวัตนาการ (CNN) โดยเฉพาะ

ในการทดลองนี้สามารถแบ่งตัวชี้วัดที่ใช้ฝึกฝนเป็นสองด้านหลักคือด้านประสิทธิภาพ (Performance) และด้านความสามารถในการเคลื่อนย้าย (Portability) โดยในด้านประสิทธิภาพประกอบไปด้วยการวัดประสิทธิภาพความถูกต้องที่นำผลความถูกต้องของทั้ง 10 fold cross-validation มาหาค่าเฉลี่ย เพื่อยืนยันความถูกต้องของแต่ละแบบจำลอง โดยแสดงรูปแบบความถูกต้องในการพยากรณ์ของลำดับที่ 1 (Top-1 Accuracy) กับลำดับที่ 1 ถึง 5 (Top-5 Accuracy) และ ยังสามารถพิจารณาเวลาสำหรับการประมวลผลของแต่ละแบบจำลอง โดยการกำหนดพารามิเตอร์ batch size เท่ากับ 1 ในลำดับชั้นข้อมูล (Data layer) เพื่อให้โครงข่ายประสาทเทียมประมวลผลเพียง 1 รอบ (iteration) ต่อ 1 ภาพเท่านั้น ด้วยการสุ่มภาพมาประมวลผลทั้งสิ้น 500

รอบ ซึ่งแสดงอยู่ในรูปแบบของ Average Forward-Backward ส่วนด้านความสามารถในการเคลื่อนย้ายนั้นตัวชี้วัดคือการใช้หน่วยความจำและขนาดพื้นที่สำหรับการจัดเก็บแบบจำลอง

ตาราง 4 ประสิทธิภาพด้านความถูกต้องระหว่างแบบจำลอง NU-InNet กับแบบจำลองที่ชนะการแข่งขัน ILSVRC

Model	Accuracy (%)	
	Top-1	Top-5
AlexNet (ILSVRC 2012)	58.10	86.38
GoogLeNet (ILSVRC 2014)	68.40	91.71
NU-InNet-1.0	70.25	93.25
NU-InNet-1.1	73.04	94.42

1. ประสิทธิภาพด้านความถูกต้อง (Accuracy)

หากพิจารณาประสิทธิภาพด้านความถูกต้องดังตาราง 4 ของแบบจำลองที่ชนะการแข่งขัน ILSVRC พบว่าประสิทธิภาพด้านความถูกต้องในการพยากรณ์ลำดับที่ 1 ของแบบจำลอง AlexNet กับแบบจำลอง GoogLeNet นั้นมีค่าเท่ากับ 58.10% และ 68.40% ตามลำดับ หากพิจารณาประสิทธิภาพด้านความถูกต้องในการพยากรณ์ลำดับที่ 1 ถึง 5 ของแบบจำลอง AlexNet กับแบบจำลอง GoogLeNet ที่ให้ค่าเท่ากับ 86.38% และ 91.71% ตามลำดับ ข้อมูลประสิทธิภาพด้านความถูกต้องของแบบจำลองที่ชนะการแข่งขัน ILSVRC แสดงให้เห็นว่าแบบจำลอง GoogLeNet มีประสิทธิภาพด้านความถูกต้องสูงกว่าแบบจำลอง AlexNet ถึง 10.3% ในการพยากรณ์ลำดับที่ 1 และ 5.33% สำหรับการพยากรณ์ลำดับที่ 1 ถึง 5

แบบจำลอง NU-InNet ที่ถูกพัฒนาขึ้นในงานวิจัยชิ้นนี้ มีทั้งสิ้นสองรุ่นได้แก่ NU-InNet 1.0 และ NU-InNet 1.1 ซึ่งถูกพัฒนาด้วยแนวคิด Inception module ของ GoogLeNet ที่มีประสิทธิภาพด้านความถูกต้องในการฝึกฝนและพยากรณ์ได้แม่นยำมากขึ้นจากแบบจำลอง AlexNet จากผลการทดลองของแบบจำลอง NU-InNet ในด้านประสิทธิภาพความถูกต้องในการพยากรณ์พบว่าค่าความถูกต้องในการพยากรณ์ลำดับที่ 1 ของ NU-InNet 1.0/1.1 มีค่า 70.25% และ 73.04% ตามลำดับสำหรับค่าความถูกต้องในการพยากรณ์ลำดับที่ 1 ถึง 5 ของ NU-InNet 1.0/1.1 มีค่า 93.25% และ94.42% ตามลำดับ จากข้อมูลดังกล่าวแสดงให้เห็นว่าประสิทธิภาพด้านความถูกต้องของแบบจำลอง NU-InNet 1.1 มีความถูกต้องมากกว่าแบบจำลอง NU-InNet 1.0 ถึง 2.79% สำหรับการพยากรณ์ลำดับที่ 1 และ 1.17% ในการพยากรณ์ลำดับที่ 1 ถึง 5

จากข้อมูลประสิทธิภาพด้านความถูกต้องของแบบจำลอง NU-InNet 1.0/1.1 ดังที่กล่าวมาข้างต้น พบว่าแบบจำลอง NU-InNet ทั้งสองรุ่นมีประสิทธิภาพด้านความถูกต้องสำหรับการพยากรณ์ลำดับที่ 1 สูงกว่าแบบจำลอง GoogLeNet ถึง 1.85% และ 4.64% ตามลำดับ ส่วนการพยากรณ์ลำดับที่ 1 ถึง 5 สูงกว่าแบบจำลอง GoogLeNet ถึง 1.54% และ 2.71% ตามลำดับ เนื่องจากแบบจำลอง NU-InNet ถูกพัฒนาขึ้นมาด้วยการปรับแต่งสถาปัตยกรรมและโครงสร้างของโครงข่ายประสาทเทียมใหม่ เพื่อให้เหมาะสมกับชุดข้อมูลภาพถ่ายอาหารไทย ซึ่งแบบจำลอง NU-InNet 1.0 ได้วิเคราะห์ข้อมูลภาพถ่ายในแต่ละระดับ [224x224, 109x109, 54x54, 25x25, 12x12] แต่มีเพียงระดับ 12x12 ของภาพเท่านั้นที่เลือกใช้ NU-Inception-1.0 มาวิเคราะห์ข้อมูล จากนั้นเมื่อนำข้อมูลที่ได้จากการวิเคราะห์ ซึ่งมีข้อมูลจำนวนเท่ากับ [12x12x256] มาปรับขนาดของภาพด้วยลำดับชั้น Average pooling ทำให้ข้อมูลที่ได้เท่ากับ [1x1x256] ซึ่งหมายความว่าจำนวนภาพถ่าย 1 ภาพ ที่ผ่านกระบวนการวิเคราะห์มาแล้วจะได้คุณลักษณะ (Feature vector) จำนวน 256 ค่า เมื่อนำไปเปรียบเทียบกับแบบจำลอง AlexNet กับ GoogLeNet ที่ถูกพัฒนาขึ้นมาเพื่อใช้กับชุดข้อมูล ILSVRC ที่มีจำนวน 1,000 กลุ่ม ทำให้คุณลักษณะของแบบจำลองทั้งสองมีจำนวน 1,024 ค่า เมื่อนำแบบจำลองทั้งสองมาใช้กับชุดข้อมูล THFOOD-50 จึงทำให้มีคุณลักษณะเด่นที่มากเกินไป ซึ่งอาจส่งผลให้ประสิทธิภาพด้านความถูกต้องลดลง เมื่อเปรียบเทียบกับแบบจำลอง NU-InNet ที่มีคุณลักษณะเพียง 256 ค่า และประกอบกับแบบจำลอง NU-InNet ที่เลือกใช้ Batch Normalization [24] หลังลำดับชั้น Convolutions จึงทำให้แบบจำลอง NU-InNet มีประสิทธิภาพด้านความถูกต้องทั้งในการพยากรณ์ลำดับที่ 1 และลำดับที่ 1 ถึง 5 สูงกว่าแบบจำลอง AlexNet กับ GoogLeNet

ตาราง 5 ประสิทธิภาพด้านเวลาระหว่างแบบจำลอง NU-InNet กับ
แบบจำลองที่ชนะการแข่งขัน ILSVRC

Model	Average Forward-Backward (ms/image)	Total training time (hours)
AlexNet (ILSVRC 2012)	13.9026	60.90
GoogLeNet (ILSVRC 2014)	40.1308	175.80
NU-InNet-1.0	12.5398	54.93
NU-InNet-1.1	20.4518	89.59

2. ประสิทธิภาพด้านเวลา

ประสิทธิภาพด้านเวลาหมายถึงการตรวจวัดเวลาของแบบจำลองโครงข่ายประสาทเทียมที่ใช้ในการประมวลผลต่อ 1 ภาพ มีหน่วยเป็นมิลลิวินาที (ms/image) โดยแสดงอยู่ในรูปแบบ Average

Forward-Backward ที่ทำการคำนวณและประมวลผลทั้งหมดในแต่ละลำดับชั้นของแบบจำลอง หากพิจารณาประสิทธิภาพด้านเวลา จากตาราง 5 แบบจำลอง AlexNet ที่ชนะการแข่งขัน ILSVRC ในปีคริสต์ศักราช 2012 ใช้เวลาในการประมวลผล 1 ภาพ ต่อ 13.9026 มิลลิวินาที และแบบจำลอง GoogLeNet ชนะการแข่งขัน ILSVRC ในปีคริสต์ศักราช 2014 ใช้เวลาในการประมวลผล 1 ภาพ ต่อ 40.1308 มิลลิวินาที สาเหตุที่แบบจำลอง GoogLeNet ใช้เวลาในการประมวลผลมากกว่าแบบจำลอง AlexNet ถึง 26.2282 มิลลิวินาที เนื่องจากแบบจำลอง GoogLeNet ถูกสร้างขึ้น เพื่อเพิ่มประสิทธิภาพด้านความถูกต้องสำหรับการรู้จำภาพถ่าย ทำให้การออกแบบโครงสร้างของแบบจำลอง GoogLeNet มีความซับซ้อน และเพิ่ม Inception module ที่นำมาเรียงต่อกัน เพื่อวิเคราะห์ข้อมูลให้ละเอียดมากขึ้นจำนวน 9 module ทำให้เวลาที่ใช้ในการประมวลผลจึงเพิ่มขึ้นหากนำไปเปรียบเทียบกับแบบจำลอง AlexNet

เมื่อพิจารณาประสิทธิภาพด้านเวลาของแบบจำลอง NU-InNet ทั้งสองรุ่นพบว่าแบบจำลอง NU-InNet 1.0 ใช้เวลาในการประมวลผล 12.5398 มิลลิวินาทีต่อ 1 ภาพ และแบบจำลอง NU-InNet 1.1 ใช้เวลาในการประมวลผล 20.4518 มิลลิวินาทีต่อ 1 ภาพ จึงพบว่าแบบจำลอง NU-InNet 1.1 ใช้เวลาในการประมวลเพิ่มมากขึ้น 7.912 มิลลิวินาทีต่อ 1 ภาพ สาเหตุมาจากแบบจำลอง NU-InNet 1.1 ได้พัฒนามาจาก NU-InNet 1.0 ที่ปรับเปลี่ยนลำดับชั้น Convolutions ที่มีขนาดตัวกรอง 5x5 และ 7x7 แทนด้วยลำดับชั้น Convolutions ที่มีตัวกรอง 3x3 จำนวน 2 ลำดับชั้นและลำดับชั้น Convolutions ที่มีตัวกรอง 3x3 จำนวน 3 ลำดับชั้น ตามลำดับ เมื่อทำการเปลี่ยนลำดับชั้นดังกล่าวในแบบจำลอง NU-InNet 1.1 ทำให้ลำดับชั้นของแบบจำลองมีจำนวนเพิ่มมากขึ้น ส่งผลให้ NU-InNet 1.1 ใช้เวลาในการประมวลเพิ่มมากตามไปด้วย

จากข้อมูลประสิทธิภาพด้านเวลาของแบบจำลอง AlexNet, GoogLeNet, NU-InNet 1.0 และ NU-InNet 1.1 ข้างต้น พบว่าแบบจำลอง NU-InNet 1.0 ใช้เวลาในการประมวลผลใกล้เคียงกับแบบจำลอง AlexNet โดยใช้เวลาน้อยกว่าเพียง 1.3628 มิลลิวินาทีต่อ 1 ภาพ แต่เมื่อนำเวลาที่ใช้การประมวลผลของแบบจำลอง NU-InNet 1.1 มาเปรียบเทียบกับแบบจำลอง GoogLeNet พบว่าแบบจำลอง NU-InNet 1.1 ใช้เวลาน้อยกว่าถึง 19.679 มิลลิวินาทีต่อ 1 ภาพ

นอกจากนี้เมื่อพิจารณาเวลาที่ใช้ฝึกฝนทั้งหมด (กำหนดให้การฝึกฝนสิ้นสุดลงรอบ ที่ 100 และใช้ 10 fold cross-validation) ในการทดลองนี้จึงได้บันทึกระยะเวลาในการฝึกฝน (Total training time) ของแต่ละแบบจำลอง ดังตาราง 5 แสดงให้เห็นว่าแบบจำลองที่ใช้เวลาในการฝึกฝนน้อยที่สุดคือแบบจำลอง NU-InNet 1.0 ใช้เวลาประมาณ 54.93 ชั่วโมง และแบบจำลองที่ใช้เวลามากที่สุดคือแบบจำลอง GoogLeNet ใช้เวลาประมาณ 175.80 ชั่วโมง ซึ่งข้อมูลดังกล่าวสอดคล้องกับข้อมูลประสิทธิภาพด้านเวลา (Average Forward-Backward) ที่ได้จากตาราง 5 ดังนั้นจึงแสดงให้เห็นว่าแบบจำลอง NU-InNet 1.0 ใช้เวลาฝึกฝนชุดข้อมูลได้เร็วกว่าแบบจำลอง AlexNet และทั้ง

แบบจำลอง NU-InNet 1.0 และ NU-InNet 1.1 ใช้เวลาฝึกฝนชุดข้อมูลได้เร็วกว่าแบบจำลอง GoogLeNet

ตาราง 6 การใช้หน่วยความจำหลักและขนาดสำหรับจัดเก็บแบบจำลองระหว่างแบบจำลอง NU-InNet กับ แบบจำลองที่ชนะการแข่งขัน ILSVRC

Model	Total memory [only forward] (MB/image)	Model size (MB)
AlexNet (ILSVRC 2012)	8.96	217
GoogLeNet (ILSVRC 2014)	40.96	39.9
NU-InNet-1.0	27.12	1.67
NU-InNet-1.1	30.84	1.47

3. หน่วยความจำและขนาดสำหรับจัดเก็บแบบจำลอง

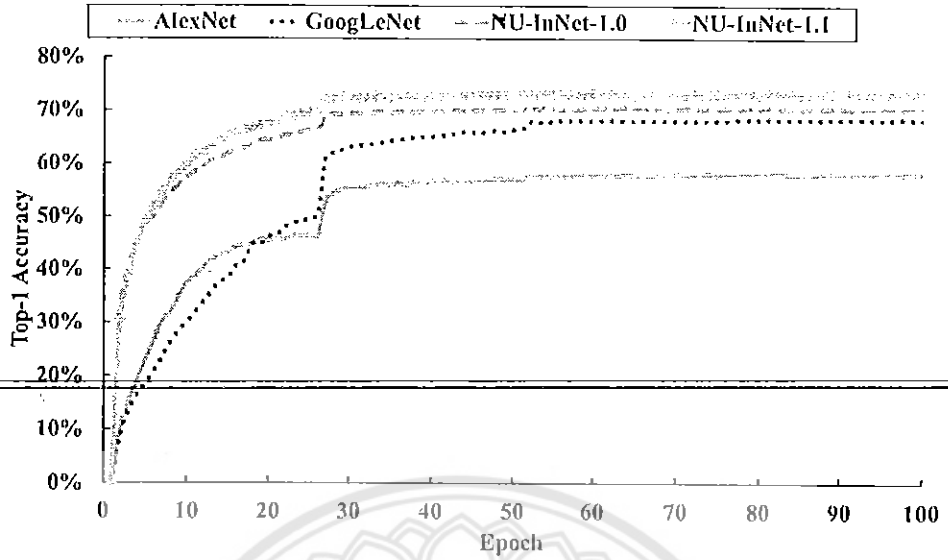
ปัจจัยของการออกแบบโครงข่ายประสาทเทียมแบบสังวัตนาการที่จำเป็นต้องคำนึงถึงคือการใช้งานหน่วยความจำหลัก (RAM) และหน่วยความจำสำรอง (Storage) เนื่องจากโครงข่ายประสาทเทียมแบบสังวัตนาการมีการวิเคราะห์ข้อมูลภาพถ่ายด้วยการขยายความลึกของภาพให้เพิ่มมากขึ้น ส่งผลให้ใช้หน่วยความจำหลักจำนวนมาก ทั้งยังส่งผลต่อกระบวนการฝึกฝนซึ่งจำเป็นต้องใช้หน่วยประมวลผลกราฟิกส์ที่มีความจุของหน่วยความจำหลักจำนวนมากเช่นเดียวกัน นอกจากนี้การออกแบบโครงข่ายมีผลต่อขนาดสำหรับจัดเก็บแบบจำลองบนหน่วยความจำสำรอง เนื่องจากปริมาณข้อมูลในแต่ละลำดับชั้นถูกจัดเก็บลงในรูปแบบค่าน้ำหนัก (weight) หรือเรียกอีกชื่อว่า "พารามิเตอร์" เมื่อนำพารามิเตอร์มารวมกันแล้วจะได้ขนาดสำหรับจัดเก็บแบบจำลอง (Model size)

เมื่อพิจารณาการใช้หน่วยความจำหลักของการประมวลผลของโครงข่ายแบบไม่ย้อนกลับ (forward) จากตาราง 6 ของแบบจำลองที่ชนะการแข่งขัน ILSVRC พบว่าแบบจำลอง AlexNet ใช้หน่วยความจำหลัก 8.96 เมกะไบต์ ต่อ 1 ภาพ หากนำมาเปรียบเทียบกับแบบจำลอง GoogLeNet ใช้หน่วยความจำหลักถึง 40.96 เมกะไบต์ ต่อ 1 ภาพ สาเหตุที่มีการใช้หน่วยความจำหลักเพิ่มมากขึ้นถึง 32 MB นั้นเนื่องจากการออกแบบของแบบจำลอง GoogLeNet ได้เพิ่มจำนวนของลำดับชั้นมากขึ้นเพื่อวิเคราะห์ข้อมูลได้ละเอียดมากขึ้นกว่าแบบจำลอง AlexNet จึงทำให้แบบจำลอง GoogLeNet ใช้หน่วยความจำหลักและเวลาในการประมวลผลเพิ่มมากขึ้นกว่าแบบจำลอง AlexNet

หากพิจารณาแบบจำลอง NU-InNet ทั้งสองรุ่น พบว่าแบบจำลอง NU-InNet-1.0 และ NU-InNet-1.1 ใช้หน่วยความจำหลักต่อ 1 ภาพ จำนวน 27.12 เมกะไบต์ และ 30.84 เมกะไบต์

ตามลำดับ โดยที่แบบจำลอง NU-InNet 1.1 สาเหตุที่ใช้หน่วยความจำหลักมากขึ้นกว่าแบบจำลอง NU-InNet 1.0 เนื่องจากสถาปัตยกรรมและโครงสร้างของแบบจำลอง NU-InNet 1.1 มีลักษณะเช่นเดียวกับแบบจำลอง NU-InNet 1.0 แต่แตกต่างกันตรงที่แบบจำลอง NU-InNet 1.1 ได้เปลี่ยนลำดับชั้น Convolutions ที่มีตัวกรองมากกว่า 3x3 ให้เหลือเพียง 3x3 เท่านั้น และเพิ่มจำนวนของลำดับชั้น Convolutions ขึ้นมาทดแทน ทำให้แบบจำลอง NU-InNet-1.1 มีลำดับชั้นเพิ่มมากขึ้น จึงส่งผลให้ใช้หน่วยความจำเพิ่มขึ้นจำนวน 3.7 เมกะไบต์ หากนำข้อมูลการใช้งานหน่วยความจำหลักของแบบจำลองที่ขนะการแข่งขัน ILSVRC กับแบบจำลอง NU-InNet ทั้งสองรุ่นมาเปรียบเทียบกัน พบว่าแบบจำลอง AlexNet ใช้หน่วยความจำหลักน้อยที่สุดเพียง 8.96 เมกะไบต์ ต่อ 1 ภาพ และเมื่อนำแบบจำลอง GoogLeNet มาเปรียบเทียบกับแบบจำลอง NU-InNet ที่นำแนวคิด Inception module มาปรับเปลี่ยน พบว่าแบบจำลอง NU-InNet ใช้หน่วยความจำหลักลดลงถึง 10 เมกะไบต์

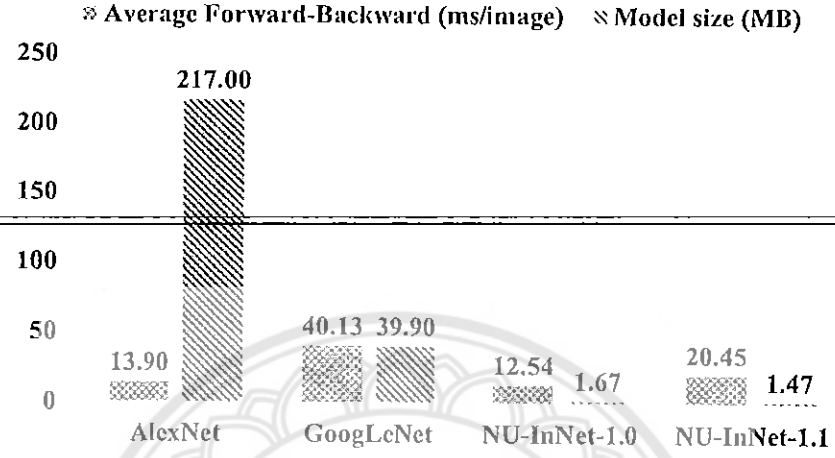
นอกจากการใช้งานหน่วยความจำหลักแล้วขนาดสำหรับจัดเก็บแบบจำลองลงในหน่วยความจำสำรองมีความสำคัญเช่นเดียวกัน ปัจจัยที่ส่งผลต่อขนาดสำหรับจัดเก็บแบบจำลอง ได้แก่ สถาปัตยกรรมและโครงสร้างของแบบจำลอง โดยแบบจำลอง GoogLeNet ได้เลือกใช้ลำดับชั้น Convolutions ที่มีตัวกรอง 1x1 มาวิเคราะห์และลดขนาดของความลึกลงก่อนส่งต่อไปยังลำดับชั้น Convolutions ที่มีตัวกรอง 3x3 หรือ 5x5 จึงทำให้ลำดับชั้นหลังจากที่ใช้ตัวกรอง 1x1 มีพารามิเตอร์ที่ลดลง ดังนั้นขนาดสำหรับจัดเก็บแบบจำลองของแบบจำลอง GoogLeNet จึงลดลงจากแบบจำลอง AlexNet ถึง 177 เมกะไบต์ เหลือเพียง 39.9 เมกะไบต์ และสำหรับแบบจำลอง NU-InNet ทั้งสองรุ่นนั้นได้นำ Inception module มาปรับเปลี่ยนใหม่ โดยที่ยังคงใช้ลำดับชั้น Convolutions ที่มีตัวกรอง 1x1 เช่นเดียวกับแบบจำลอง GoogLeNet เพื่อลดจำนวนพารามิเตอร์ลง เพียงแต่แบบจำลอง NU-InNet ทั้งสองรุ่นนั้นใช้ NU-Inception module เพียง 1 module เท่านั้น แตกต่างกับแบบจำลอง GoogLeNet ที่ใช้ Inception module ถึง 9 module จากนั้นเมื่อคำนวณพารามิเตอร์ทั้งหมดแสดงให้เห็นว่าพื้นที่สำหรับจัดเก็บแบบจำลอง NU-InNet-1.0 และ NU-InNet-1.1 ลดลงเพียง 1.67 เมกะไบต์ และ 1.47 เมกะไบต์ ตามลำดับ



ภาพ 33 Top-1 Accuracy ของ AlexNet, GoogLeNet, NU-InNet 1.0 และ NU-InNet 1.1

ในกระบวนการฝึกฝนได้กำหนดค่า Learning rate (LR) เริ่มต้นที่ 0.01 และได้ลดจำนวนลง 10 เท่าในแต่ละ 25 รอบ (Epoch) เมื่อนำ Top-1 Accuracy ของแบบจำลอง AlexNet, GoogLeNet, NU-InNet 1.0 และ NU-InNet 1.1 ที่ฝึกฝนเสร็จสิ้นมาแสดงในแต่ละรอบของการฝึกฝน จะได้ดังภาพ 33 เมื่อพิจารณาข้อมูลประสิทธิภาพด้านความถูกต้องของแบบจำลองที่ชนะการแข่งขัน ILSVRC ทำให้ทราบว่าแบบจำลอง AlexNet มีประสิทธิภาพด้านความถูกต้องสูงกว่าแบบจำลอง GoogLeNet ในการฝึกฝนรอบที่ 1 ถึง 20 เท่านั้น จากนั้นตั้งแต่รอบที่ 26 เป็นต้นไป (LR เท่ากับ 0.001) แบบจำลอง GoogLeNet กลับมีประสิทธิภาพที่ดีขึ้นและสูงกว่าแบบจำลอง AlexNet จนกระทั่งสิ้นสุดการฝึกฝนในรอบที่ 100 ประสิทธิภาพด้านความถูกต้องของแบบจำลอง AlexNet มีค่า Top-1 Accuracy ที่ต่ำกว่าแบบจำลอง GoogLeNet ถึง 10.3% และหากพิจารณาประสิทธิภาพด้านความถูกต้องของแบบจำลอง NU-InNet พบว่าแบบจำลอง NU-InNet ทั้งสองรุ่นมีค่า Top-1 Accuracy ที่ใกล้เคียงกัน โดยแบบจำลอง NU-InNet 1.1 มีประสิทธิภาพด้านความถูกต้องสูงกว่าแบบจำลอง NU-InNet 1.0 ทั้งนี้เมื่อนำไปเปรียบเทียบกับแบบจำลอง GoogLeNet พบว่าแบบจำลอง NU-InNet ทั้งสองรุ่นนั้นมีประสิทธิภาพด้านความถูกต้องลู่เข้าได้ดีและสูงกว่าแบบจำลอง GoogLeNet ตั้งแต่การฝึกฝนรอบที่ 1 ถึง 100 ซึ่งแบบจำลอง GoogLeNet เริ่มลู่เข้าในรอบที่ 26 ถึง 50 (LR เท่ากับ 0.001) จากนั้นตั้งแต่รอบที่ 51 ถึง 100 ค่า Top 1 Accuracy ของแบบจำลองทั้งหมดให้ค่าคงที่ แสดงให้เห็นว่าระหว่างการฝึกฝนในรอบที่ 1 ถึง 30 นั้น NU-InNet ทั้งสองรุ่นให้ค่า Top 1 Accuracy ที่มีประสิทธิภาพดีกว่า GoogLeNet ทั้งในด้านเวลาในการลู่เข้าและประสิทธิภาพความ

ถูกต้อง และหลังจากการฝึกฝนสิ้นสุดลงในรอบที่ 100 แบบจำลอง NU-InNet ทั้งสองรุ่นให้ผลความถูกต้องมากกว่า GoogLeNet ถึง 1.85% และ 4.64% ตามลำดับ



ภาพ 34 ประสิทธิภาพด้านเวลาและพื้นที่จัดเก็บของ AlexNet, GoogLeNet และ NU-InNet

เมื่อนำโครงข่ายประสาทเทียมแบบสังวัตนาการไปใช้กับอุปกรณ์สมาร์ทโฟน จำเป็นต้องคำนึงถึงประสิทธิภาพด้านเวลาและขนาดพื้นที่จัดเก็บแบบจำลองให้เหมาะสมสำหรับอุปกรณ์สมาร์ทโฟนที่มีทรัพยากรอย่างจำกัด ดังนั้นการนำแบบจำลอง AlexNet ที่มีขนาดพื้นที่จัดเก็บแบบจำลองมากถึง 217 เมกะไบต์ จึงไม่เหมาะสำหรับนำไปพัฒนาบนอุปกรณ์สมาร์ทโฟน หากพิจารณาขนาดพื้นที่จัดเก็บของแบบจำลอง GoogLeNet ที่มีขนาด 39.9 เมกะไบต์ ที่ลดลงจากแบบจำลอง AlexNet มากถึง 5.4 เท่า แต่ขนาดของแบบจำลอง GoogLeNet ยังคงมีขนาดที่ใหญ่เกินไป เพราะสมาร์ทโฟนบางรุ่นเหลือพื้นที่ให้เก็บข้อมูลไม่เพียงพอต่อความต้องการของผู้ใช้งาน ทำให้ผู้ใช้ต้องจัดการถอนการติดตั้งแอพพลิเคชั่นที่ใช้พื้นที่จำนวนมากออกไป ดังนั้นหากพิจารณาขนาดของแบบจำลองต่าง ๆ จากภาพ 38 สามารถสรุปได้ว่าแบบจำลอง NU-InNet ทั้งสองรุ่น เหมาะที่จะนำไปใช้งานบนสมาร์ทโฟนมากที่สุด เพราะใช้พื้นที่สำหรับจัดเก็บเพียง 1.67 เมกะไบต์ และ 1.47 เมกะไบต์ ตามลำดับ เมื่อนำไปเปรียบเทียบกับแบบจำลอง GoogLeNet ที่มีขนาดถึง 39.9 เมกะไบต์ จึงทำให้พื้นที่สำหรับจัดเก็บแบบจำลองของ NU-InNet 1.1 มีขนาดที่เล็กกว่าแบบจำลอง GoogLeNet ถึง 27 เท่า นอกจากนี้เมื่อพิจารณาประสิทธิภาพด้านเวลา (Average Forward-Backward) ในภาพ 34 พบว่าแบบจำลอง NU-InNet 1.0 ใช้เวลาสำหรับประมวลผลต่อ 1 ภาพ เพียง 12.54 มิลลิวินาที ซึ่งเวลาที่ใช้น้อยกว่าแบบจำลอง AlexNet และ GoogLeNet ประมาณ 1.36 และ 27.59 มิลลิวินาทีต่อ 1 ภาพ ตามลำดับ

การทดลองระหว่างแบบจำลอง Deep NU-InNet กับแบบจำลอง GoogLeNet

หลังจากทดลองของแบบจำลอง NU-InNet จะได้แบบจำลองที่เหมาะสมสำหรับอุปกรณ์สมาร์ทโฟนคือแบบจำลอง NU-InNet ด้วยประสิทธิภาพด้านความถูกต้องสูงที่สุด ใช้เวลาในการประมวลผลและขนาดของพื้นที่จัดเก็บน้อยที่สุด สำหรับแบบจำลอง NU-InNet 1.1 มีประสิทธิภาพด้านความถูกต้องที่มีค่า Top-1 Accuracy สูงที่สุดเท่ากับ 73.04 % และมีขนาดของพื้นที่จัดเก็บแบบจำลองเพียง 1.47 MB ในการทดลองนี้จึงได้นำแบบจำลอง NU-InNet ทั้งสองรุ่นมาปรับปรุงเพื่อเพิ่มประสิทธิภาพด้านความถูกต้องให้สูงขึ้น โดยยังคงขนาดของพื้นที่จัดเก็บแบบจำลองให้เหมาะสมกับการนำไปใช้งานบนสมาร์ทโฟน ด้วยการเพิ่มจำนวนของ NU-Inception module ในแต่ละระดับของภาพ [56×56, 28×28, 14×14, 7×7] เพื่อวิเคราะห์ข้อมูลภาพถ่ายอย่างละเอียดมากขึ้น โดยมีความลึกของแบบจำลองอยู่ 3 ระดับได้แก่ 4, 8 และ 12 (N=1, 2, 3) สำหรับกระบวนการฝึกฝนจึงได้นำแบบจำลอง GoogLeNet, BN-Inception และ NU-InNet 1.0/1.1 ทั้ง 4 ความลึก (Depth=1, 4, 8, 12) มาเรียนรู้แบบ from scratch โดยแบ่งชุดข้อมูลของ THFOOD-50 ด้วยสัดส่วนชุดข้อมูลฝึกฝนจำนวน 90% และชุดข้อมูลทดสอบจำนวน 10% โดยใช้ 10 fold cross-validation ซึ่งได้ประมวลผลบนเครื่องคอมพิวเตอร์ประสิทธิภาพสูง

ตาราง 7 ประสิทธิภาพด้านความถูกต้องระหว่างแบบจำลอง Deep NU-InNet กับแบบจำลอง GoogLeNet

Model	Depth	Accuracy (%)	
		Top-1	Top-5
GoogLeNet	9	68.40	91.71
BN-Inception	9	72.27	93.29
NU-InNet 1.0	1	70.25	93.25
NU-InNet 1.0	4	79.68	96.00
NU-InNet 1.0	8	78.62	95.68
NU-InNet 1.0	12	75.44	94.36
NU-InNet 1.1	1	73.04	94.42
NU-InNet 1.1	4	80.34	96.27
NU-InNet 1.1	8	79.75	95.93
NU-InNet 1.1	12	75.69	94.34

1. ประสิทธิภาพด้านความถูกต้อง (Accuracy)

หากกล่าวถึงประสิทธิภาพด้านความถูกต้องจากตาราง 7 ของแบบจำลอง GoogLeNet และ BN-Inception พบว่าประสิทธิภาพด้านความถูกต้องในการพยากรณ์ลำดับที่ 1 ของแบบจำลอง GoogLeNet กับ BN-Inception มีค่าเท่ากับ 68.40% กับ 72.27% ตามลำดับ และประสิทธิภาพด้านความถูกต้องในการพยากรณ์ลำดับที่ 1-5 ของแบบจำลอง GoogLeNet กับ BN-Inception มีค่าเท่ากับ 91.71% กับ 93.29% ตามลำดับ จากข้อมูลประสิทธิภาพด้านความถูกต้องของทั้งสองแบบจำลองแสดงให้เห็นว่าแบบจำลอง BN-Inception ที่พัฒนามาจาก GoogLeNet ด้วยการเพิ่มลำดับชั้น Batch Normalization (BN) เข้ามาใส่หลังลำดับชั้น Convolutions และก่อนลำดับชั้น Activations ทำให้สามารถช่วยเพิ่มประสิทธิภาพด้านความถูกต้องได้ดีขึ้น

สำหรับแบบจำลอง Deep NU-InNet ประกอบไปด้วยแบบจำลอง NU-InNet รุ่น 1.0 กับ 1.1 ในแต่ละรุ่นมีความลึกทั้งหมด 4 ระดับได้แก่ แบบจำลอง NU-InNet (ความลึกที่ 1) แบบจำลอง NU-InNet (ความลึกที่ 4) มีโครงสร้างของแบบจำลองที่นำ NU-Inception module มาใช้จำนวน 4 module ในแต่ละระดับของภาพ [56×56, 28×28, 14×14, 7×7] แบบจำลอง NU-InNet (ความลึกที่ 8) มีโครงสร้างเช่นเดียวกับ แบบจำลอง NU-InNet (ความลึกที่ 4) โดยได้เพิ่มจำนวนของ NU-Inception module ในแต่ละระดับของภาพ ทำให้จำนวนของ NU-Inception module เท่ากับ 8 module และแบบจำลอง NU-InNet (ความลึกที่ 12) ได้เพิ่มจำนวนของ NU-Inception module เข้ามาในแต่ละระดับของภาพ ทำให้จำนวนของ NU-Inception module เท่ากับ 12 module เมื่อนำข้อมูลประสิทธิภาพด้านความถูกต้องของแบบจำลอง NU-InNet รุ่น 1.0 ทั้ง 4 ความลึก พบว่าหากเพิ่มจำนวนของ NU-Inception module 1.0 เข้าไปในแบบจำลองจำนวน 4 module ส่งผลให้ประสิทธิภาพด้านความถูกต้องในการพยากรณ์ลำดับที่ 1 เพิ่มขึ้น 9.43% และเมื่อเพิ่มจำนวนของ NU-Inception module 1.0 เข้าไปในแบบจำลอง 8 และ 12 module จึงทำให้ประสิทธิภาพด้านความถูกต้องในการพยากรณ์ลำดับที่ 1 เพิ่มขึ้น 8.37% และ 5.19% ตามลำดับ และเมื่อพิจารณาแบบจำลอง NU-InNet 1.1 ทั้ง 4 ความลึก พบว่าประสิทธิภาพด้านความถูกต้องหลังจากการเพิ่มจำนวน NU-Inception module 1.1 เข้าไปในแบบจำลอง 4 module มีค่าที่เพิ่มขึ้น 7.3% และเมื่อเพิ่มจำนวน NU-Inception module 1.1 เข้าไปในแบบจำลอง 8 และ 12 module จึงทำให้ประสิทธิภาพด้านความถูกต้องเพิ่มขึ้น 6.71% และ 75.69% และหากพิจารณาความถูกต้องในการพยากรณ์ลำดับที่ 1-5 ของแบบจำลอง NU-InNet ทั้งรุ่น 1.0 และ 1.1 พบว่าแบบจำลองที่มีความลึก 4 module มีประสิทธิภาพด้านความถูกต้องสูงสุดเช่นเดียวกับการพยากรณ์ลำดับที่ 1 ดังนั้นเมื่อนำประสิทธิภาพด้านความถูกต้องของแบบจำลอง BN-Inception ที่ให้ผลที่สูงกว่าแบบจำลอง GoogLeNet มาเปรียบเทียบแบบจำลอง NU-InNet ที่มีความลึก 4 module ทั้งรุ่น 1.0 และ 1.1

พบว่า NU-InNet 1.0 และ 1.1 (ความลึกที่ 4) มีประสิทธิภาพด้านความถูกต้องในการพยากรณ์ลำดับที่ 1 สูงกว่าแบบจำลอง BN-Inception (ความลึกที่ 9) ถึง 7.41% และ 8.07% ตามลำดับ

ตาราง 8 ประสิทธิภาพด้านเวลาระหว่างแบบจำลอง Deep NU-InNet กับแบบจำลอง GoogLeNet

Model	Depth	Average Forward-Backward (ms/image)	Total training time (hours)
GoogLeNet	9	40.1308	175.80
BN-Inception	9	63.2429	277.04
NU-InNet 1.0	1	12.5398	54.93
NU-InNet 1.0	4	37.6629	164.98
NU-InNet 1.0	8	66.0797	289.46
NU-InNet 1.0	12	95.4761	418.23
NU-InNet 1.1	1	20.4518	89.59
NU-InNet 1.1	4	40.8042	178.74
NU-InNet 1.1	8	71.7156	314.15
NU-InNet 1.1	12	100.504	440.26

2. ประสิทธิภาพด้านเวลา

หากพิจารณาประสิทธิภาพด้านเวลาของแบบจำลอง Deep NU-InNet กับแบบจำลอง GoogLeNet รวมไปถึงแบบจำลอง BN-Inception ที่ถูกพัฒนามาจากแบบจำลอง GoogLeNet โดยการตรวจวัดการประมวลผลต่อ 1 ภาพ ที่มีหน่วยเป็นมิลลิวินาที และแสดงในรูปแบบ Average Forward-Backward หลังจากตรวจวัดเรียบร้อยแล้ว จะได้ดังตาราง 8 ซึ่งเมื่อนำแบบจำลอง GoogLeNet กับแบบจำลอง BN-Inception มาเปรียบเทียบกัน พบว่าเมื่อเพิ่มลำดับชั้น Batch Normalization เข้าไปในแบบจำลอง GoogLeNet ทำให้ประสิทธิภาพด้านเวลาของแบบจำลอง BN-Inception เพิ่มขึ้น 23.1121 มิลลิวินาทีต่อ 1 ภาพ และหลังจากกระบวนการฝึกฝนเสร็จสิ้น ทั้งสองแบบจำลองใช้เวลาในการฝึกฝน 175.80 ชั่วโมง และ 277.04 ชั่วโมง ตามลำดับ

เมื่อนำข้อมูลประสิทธิภาพด้านเวลาของแบบจำลอง NU-InNet 1.0 ทั้ง 4 ความลึก มาพิจารณา พบว่าเมื่อเพิ่มจำนวน NU-Inception module 1.0 จาก 1 module เป็น 4 module

ส่งผลให้เวลาในการประมวลผลของแบบจำลอง NU-InNet 1.0 เพิ่มขึ้นเป็น 37.6629 มิลลิวินาทีต่อ 1 ภาพ หากเพิ่มจำนวน NU-Inception module 1.0 เป็น 8 module เวลาที่ใช้ในการประมวลผลเพิ่มขึ้นเป็น 66.0797 มิลลิวินาทีต่อ 1 ภาพ และเมื่อเพิ่มจำนวน NU-Inception module 1.0 เป็น 12 module ทำให้เวลาที่ใช้ในการประมวลผลเพิ่มขึ้นเป็น 95.4761 มิลลิวินาทีต่อ 1 ภาพ เมื่อพิจารณาในส่วนของรุ่น 1.1 ทั้ง 4 ความลึก พบว่าหากเพิ่มจำนวน NU-Inception module 1.1 เป็น 4 module เวลาที่ใช้ในการประมวลผลเพิ่มขึ้นเป็น 178.74 มิลลิวินาทีต่อ 1 ภาพ หากเพิ่มจำนวน NU-Inception Module 1.1 เป็น 8 Module เวลาที่ใช้ในการประมวลผลเพิ่มขึ้นเป็น 71.7156 มิลลิวินาทีต่อ 1 ภาพ และเมื่อเพิ่มจำนวน NU-Inception Module 1.1 เป็น 8 Module เวลาที่ใช้ในการประมวลผลเพิ่มขึ้นเป็น 100.504 มิลลิวินาทีต่อ 1 ภาพ

จากการเพิ่มจำนวนของ NU-Inception module นั้นมีผลให้ความลึกของแบบจำลองเพิ่มขึ้น ทำให้เวลาในการประมวลผลย่อมสูงขึ้นตาม หากพิจารณาเวลาในการฝึกฝนหลังจากกระบวนการฝึกฝนเสร็จสิ้น พบว่าเวลาในการฝึกฝนสูงขึ้นเช่นเดียวกับเวลาในการประมวลผลต่อ 1 ภาพ โดยแบบจำลอง NU-InNet 1.0 ความลึกที่ 4, 8 และ 12 ใช้เวลาในการฝึกฝนทั้งหมด 164.98, 289.46 และ 418.23 ชั่วโมง ตามลำดับ และแบบจำลอง NU-InNet 1.1 ความลึกที่ 4, 8 และ 12 ใช้เวลาในการฝึกฝนทั้งหมด 178.74, 314.15 และ 440.26 ชั่วโมง ตามลำดับ จากข้อมูลดังกล่าวแสดงให้เห็นว่าเมื่อเพิ่มจำนวนของ NU-Inception module มากขึ้น ส่งผลให้เวลาในการประมวลผลในแต่ละภาพเพิ่มขึ้น จึงทำให้เวลาที่ใช้ในการฝึกฝนทั้งหมดเพิ่มขึ้นด้วยเช่นกัน ดังนั้นหากเปรียบเทียบเวลาที่ใช้ในการประมวลผลต่อ 1 ภาพ กับเวลาที่ใช้ในการฝึกฝนทั้งหมดของแบบจำลอง Deep NU-InNet กับแบบจำลอง GoogLeNet พบว่าแบบจำลอง Deep NU-InNet ที่มีความลึก 4 module ใช้เวลาในการประมวลผลต่อ 1 ภาพ เทียบเท่าแบบจำลองแบบจำลอง GoogLeNet ที่มี Inception module จำนวน 9 module เนื่องจากการออกแบบโครงสร้างของแบบจำลอง Deep NU-InNet (ความลึกที่ 4) ใช้จำนวน module น้อยกว่าแบบจำลอง GoogLeNet แต่แบบจำลอง Deep NU-InNet ได้นำลำดับชั้น Batch Normalization มาใช้หลังจากลำดับชั้น Convolution ทุกลำดับชั้น จึงทำให้เวลาที่ใช้ในการประมวลผลเพิ่มขึ้น

ตาราง 9 การใช้หน่วยความจำหลักและขนาดสำหรับจัดเก็บแบบจำลองระหว่างแบบจำลอง Deep NU-InNet กับแบบจำลอง GoogLeNet

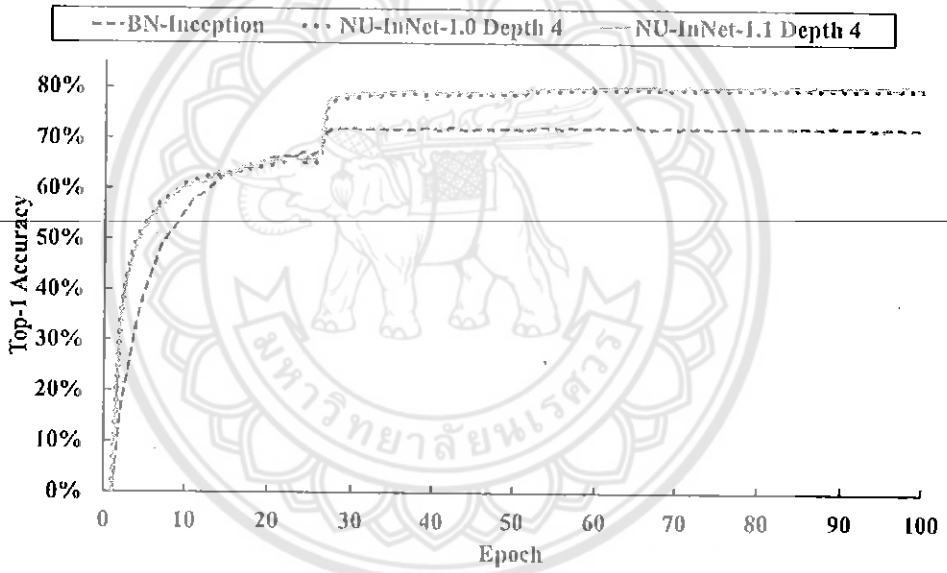
Model	Depth	Total memory [only forward] (MB/image)	Model size (MB)
GoogLeNet	9	40.96	39.9
BN-Inception	9	63.6	40.0
NU-InNet 1.0	1	27.12	1.67
NU-InNet 1.0	4	25.96	3.66
NU-InNet 1.0	8	36.52	7.70
NU-InNet 1.0	12	47.04	11.7
NU-InNet 1.1	1	30.84	1.47
NU-InNet 1.1	4	28.24	3.65
NU-InNet 1.1	8	41.04	7.69
NU-InNet 1.1	12	53.84	11.7

3. หน่วยความจำและขนาดสำหรับจัดเก็บแบบจำลอง

ข้อมูลการใช้หน่วยความจำและขนาดสำหรับจัดเก็บแบบจำลอง Deep NU-InNet กับแบบจำลอง GoogLeNet จากตาราง 9 แสดงชัดว่าข้อมูลการใช้หน่วยความจำและขนาดสำหรับจัดเก็บแบบจำลอง Deep NU-InNet ทั้งหมดนั้นใช้หน่วยความจำน้อยกว่าแบบจำลอง BN-Inception ในทุกรุ่นและทุกความลึกของแบบจำลอง Deep NU-InNet รวมไปถึงขนาดสำหรับจัดเก็บแบบจำลอง Deep NU-InNet มีขนาดที่เล็กกว่าแบบจำลอง BN-Inception โดยแบบจำลอง Deep NU-InNet ความลึกที่ 12 คือแบบจำลองที่ใช้ module มากที่สุด แต่มีขนาดสำหรับจัดเก็บแบบจำลองเพียง 11.7 เมกะไบต์ ซึ่งมีขนาดที่แตกต่างกับแบบจำลอง BN-Inception ถึง 40 เมกะไบต์

จากข้อมูลในตาราง 9 แสดงให้เห็นว่าแบบจำลอง Deep NU-InNet ที่มีความลึกมากกว่าแบบจำลอง BN-Inception ใช้หน่วยความจำและขนาดสำหรับจัดเก็บแบบจำลองน้อยกว่า เนื่องจากจำนวนของข้อมูลที่ได้จากกระบวนการฝึกฝนหรือที่เรียกว่า คุณลักษณะ (Feature) มีจำนวน 512 ค่า ซึ่งน้อยกว่าแบบจำลอง BN-Inception ถึง 488 ค่า การออกแบบของแบบจำลอง Deep NU-InNet นั้นมีจำนวนของข้อมูลนำออก (ที่ได้จากแต่ละ NU-Inception module) มีขนาดที่เล็กกว่า Inception module เช่น แบบจำลอง Deep NU-InNet ที่มีความลึก 12 module แต่ละ module

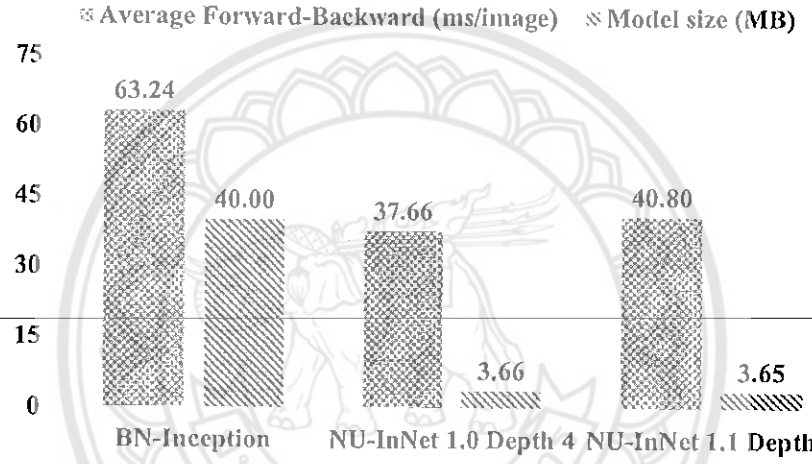
มีข้อมูลออกดังนี้ 64, 64, 64, 128, 128, 128, 256, 256, 256, 512, 512 และ 512 ค่า และแบบจำลอง BN-Inception ที่มีความลึก 9 module แต่ละ module มีข้อมูลนำออกดังนี้ 256, 480, 512, 512, 512, 528, 832, 832 และ 1024 ค่า เมื่อพิจารณาข้อมูลที่ได้จากแต่ละ module ทำให้พบว่าแบบจำลอง BN-Inception ต้องการวิเคราะห์ภาพให้มีความลึกของภาพจำนวนมาก โดยขยายจำนวนของข้อมูลนำออกเพิ่มขึ้นเรื่อย ๆ จึงทำให้มีจำนวนพารามิเตอร์เพิ่มขึ้นเป็นจำนวนมาก แต่เมื่อพิจารณาข้อมูลนำออกของแบบจำลอง Deep NU-InNet พบว่าการวิเคราะห์ภาพจะขึ้นอยู่กับความลึกของ module ดังตัวอย่างที่กล่าวมาคือ ข้อมูลนำออกที่ได้จาก NU-Inception module ที่ 2 และ 3 จะมีจำนวนเท่ากับ module ที่ 1 ซึ่งเท่ากับ 64 ค่า จากโครงสร้างของแบบจำลอง Deep NU-InNet จึงทำให้การใช้หน่วยความจำและขนาดสำหรับจัดเก็บแบบจำลองมีขนาดที่เล็กกว่าแบบจำลอง BN-Inception



ภาพ 35 Top-1 Accuracy ของแบบจำลอง BN-Inception, NU-InNet 1.0/1.1 (ความลึกที่ 4)

ในกระบวนการฝึกฝนได้กำหนดค่า Learning rate (LR) เริ่มต้นที่ 0.1 และได้ลดจำนวนลง 10 เท่าในแต่ละ 25 รอบ เมื่อนำ Top-1 Accuracy ของแบบจำลอง BN-Inception, NU-InNet 1.0 และ 1.1 (ความลึกที่ 4) ที่ฝึกฝนเสร็จสิ้นมาแสดงในแต่ละรอบของการฝึกฝน จะได้ดังภาพ 35 เมื่อพิจารณาประสิทธิภาพด้านความถูกต้องของแบบจำลอง NU-InNet 1.0 และ 1.1 (ความลึกที่ 4) พบว่ามีค่า Top-1 Accuracy ที่ใกล้เคียงกัน ตั้งแต่รอบที่ 1 ถึง 100 แต่เมื่อพิจารณาประสิทธิภาพด้านความถูกต้องของแบบจำลอง BN-Inception ระหว่างรอบที่ 1 ถึง 25 (LR เท่ากับ 0.1) ทำให้ทราบว่าแบบจำลอง BN-Inception พยามยามลู่เข้าได้ใกล้เคียงกับแบบจำลอง NU-InNet 1.0 และ 1.1

(ความลึกที่ 4) โดยที่สามารถลู่เข้าได้ใกล้ที่สุดในรอบที่ 25 ซึ่งได้ค่า Top 1 Accuracy เท่ากับ 66.50% หลังจากรอบที่ 26 (LR เท่ากับ 0.01) เป็นต้นไป ค่า Top-1 Accuracy ของแบบจำลอง NU-InNet 1.0 และ 1.1 (ความลึกที่ 4) เพิ่มขึ้นถึง 10% ในขณะที่แบบจำลอง BN-Inception เพิ่มขึ้นเพียง 5% โดยประสิทธิภาพด้านความถูกต้องของทั้งสามแบบจำลองให้ค่าคงที่จนกระทั่งถึงรอบที่ 100 จากข้อมูลดังกล่าวแสดงให้เห็นว่าระหว่างการฝึกฝนในรอบที่ 1 ถึง 25 นั้น ทั้งแบบจำลอง NU-InNet 1.0 และ 1.1 (ความลึกที่ 4) มีความเร็วในการลู่เข้าได้ดีและสูงกว่าแบบจำลอง BN-Inception และหลังจากการฝึกฝนสิ้นสุดลงในรอบที่ 100 แบบจำลอง NU-InNet 1.0 และ 1.1 (ความลึกที่ 4) ให้ผล Top 1 Accuracy สูงกว่า BN-Inception ถึง 7.41% และ 8.07% ตามลำดับ



ภาพ 36 ประสิทธิภาพด้านเวลาและขนาดพื้นที่จัดเก็บของแบบจำลอง BN-Inception NU-InNet 1.0 และ 1.1 (ความลึกที่ 4)

ในการนำแบบจำลองไปใช้กับอุปกรณ์สมาร์ทโฟน จำเป็นต้องพิจารณาประสิทธิภาพด้านเวลาและขนาดพื้นที่จัดเก็บของแบบจำลองให้เหมาะสมสำหรับสมาร์ทโฟน สำหรับแบบจำลอง Deep NU-InNet 1.0 และ 1.1 ได้เลือกความลึกที่ 4 มาใช้ เนื่องจากมีประสิทธิภาพด้านความถูกต้องสูงที่สุด จึงได้นำแบบจำลอง NU-InNet 1.0 และ 1.1 (ความลึกที่ 4) มาเปรียบเทียบประสิทธิภาพด้านเวลาและขนาดพื้นที่จัดเก็บ กับแบบจำลอง BN-Inception ดังภาพ 36 เมื่อพิจารณานำแบบจำลอง BN-Inception ที่มีขนาดพื้นที่จัดเก็บจำนวน 40 เมกะไบต์ ไปใช้กับสมาร์ทโฟน อาจทำให้ไม่เหมาะสม เนื่องจากขนาดของพื้นที่จัดเก็บใช้ไปถึง 40 เมกะไบต์ ซึ่งยังไม่ได้รวมข้อมูลที่จะเกิดขึ้นจากการพัฒนาแอพพลิเคชั่นอีก และเมื่อนำมารวมกันแล้วจะทำให้ขนาดของแอพพลิเคชั่นมีขนาดที่มากกว่า 40 เมกะไบต์ ดังนั้นหากพิจารณาขนาดของแบบจำลอง NU-InNet 1.0 และ 1.1 (ความลึกที่ 4) ที่มีขนาดของเพียง 3.66 เมกะไบต์ เมื่อนำไปเปรียบเทียบกับแบบจำลอง BN-Inception แล้วนั้น พบว่าแบบจำลอง

NU-InNet 1.0 และ 1.1 (ความลึกที่ 4) สามารถลดขนาดพื้นที่จัดเก็บลงได้มากถึง 10.9 เท่า โดยที่ประสิทธิภาพด้านความถูกต้องสูงกว่าแบบจำลอง BN-Inception นอกจากนี้เมื่อพิจารณาประสิทธิภาพด้านเวลา (Average Forward-Backward) ในภาพ 40 พบว่า แบบจำลอง NU-InNet 1.0 และ 1.1 (ความลึกที่ 4) ใช้เวลาสำหรับประมวลผลต่อ 1 ภาพ เพียง 37.66 มิลลิวินาที และ 40.80 มิลลิวินาที ตามลำดับ ซึ่งใช้เวลาน้อยกว่าแบบจำลอง BN-Inception ประมาณ 25.58 และ 22.44 มิลลิวินาทีต่อ 1 ภาพ ตามลำดับ

การทดลองแบบจำลอง NU-InNet กับ Deep NU-InNet บนสมาร์ทโฟน

หลังจาการเปรียบเทียบประสิทธิภาพด้านเวลาของแบบจำลองต่าง ๆ นั้นได้ใช้คอมพิวเตอร์ในการตรวจวัด ในหัวข้อนี้ได้นำแบบจำลองต่าง ๆ มาตรวจวัดบนอุปกรณ์สมาร์ทโฟนที่มีองค์ประกอบด้านฮาร์ดแวร์ดังนี้ หน่วยประมวลผล (CPU) Intel(R) Atom(TM) Z3580 ที่มีความเร็วนาฬิกา 2.33 จิกะเฮิรตซ์ (GHz) จำนวน 4 แกนหลัก (Core) และหน่วยความจำหลัก (RAM) จำนวน 4 จิกะไบต์ (GB) ทำงานบนระบบปฏิบัติการ Android 6.0.1

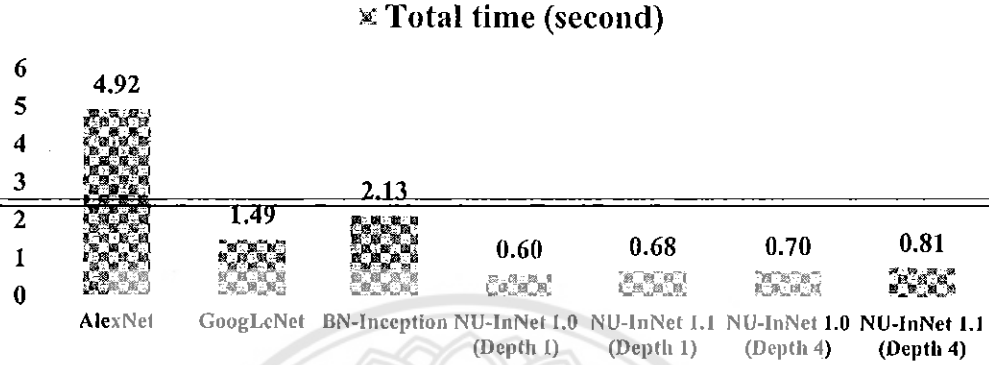
ตาราง 10 การตรวจวัดประสิทธิภาพด้านเวลาในการประมวลผลบนสมาร์ทโฟน

Model	Input: Image size	Loading time (ms)	Execution time (ms/image)
AlexNet	1080x1080	4333.22	589
GoogLeNet	1080x1080	553.16	937
BN-Inception	1080x1080	684.70	1448
NU-InNet 1.0 (Depth 1)	1080x1080	45.25	554
NU-InNet 1.1 (Depth 1)	1080x1080	50.96	633
NU-InNet 1.0 (Depth 4)	1080x1080	98.40	603
NU-InNet 1.1 (Depth 4)	1080x1080	110.64	699

สำหรับการตรวจวัดเวลาของแบบจำลองต่าง ๆ ประกอบไปด้วยสองส่วนคือ เวลาที่ใช้ในการเปิดแอพพลิเคชั่น (Loading time) และเวลาที่ใช้ในการรู้จำภาพ (Execution time) ทั้งสองส่วนจะแสดงหน่วยเป็นมิลลิวินาที ดังตาราง 10 สำหรับการตรวจวัดเวลาที่ใช้ในการเปิดแอพพลิเคชั่น จะเริ่มนับตั้งแต่มีการเปิดแอพพลิเคชั่นขึ้นมาจนกระทั่งแอพพลิเคชั่นพร้อมที่จะทำงานได้ การตรวจวัดส่วนนี้ยังรวมไปถึงเวลาที่แอพพลิเคชั่นได้บรรจุแบบจำลองที่ฝึกฝนเสร็จสิ้นเข้าไปในการทำงาน และสำหรับเวลาที่ใช้ในการเปิดแอพพลิเคชั่นมีความสำคัญอย่างยิ่งต่อผู้ใช้งาน เนื่องจากผู้ใช้งานส่วนใหญ่ต้องการความเร็วในการใช้งานแอพพลิเคชั่น ดังนั้นการเลือกแบบจำลองที่มีความเร็วในการเปิดแอพพลิเคชั่นมากที่สุดจึงเป็นเรื่องที่ควรให้ความสำคัญเป็นอย่างยิ่ง จากตาราง 10 เมื่อพิจารณาแบบจำลองที่ชนะการแข่งขัน ILSVRC พบว่าแบบจำลอง AlexNet ใช้เวลาในการเปิดแอพพลิเคชั่น 4333.22 มิลลิวินาที เมื่อนำไปเปรียบกับแบบจำลอง GoogLeNet ที่ใช้เวลาในการเปิดแอพพลิเคชั่น 553.16 มิลลิวินาที จึงทำให้ทราบว่าแบบจำลอง GoogLeNet ที่มีขนาดของพื้นที่จัดเก็บเพียง 39.9 เมกะไบต์ ใช้เวลาในการเปิดแอพพลิเคชั่นได้เร็วกว่าแบบจำลอง AlexNet ที่มีขนาดของพื้นที่จัดเก็บ 217 เมกะไบต์ จากข้อมูลดังกล่าวแสดงให้เห็นว่าขนาดของพื้นที่สำหรับจัดเก็บแบบจำลองมีผลต่อเวลาที่ใช้ในการเปิดแอพพลิเคชั่น ซึ่งเมื่อพิจารณาแบบจำลอง NU-InNet 1.0 และ 1.1 (ความลึกที่ 1) พบว่าทั้งสองแบบจำลองใช้เวลาในการเปิดแอพพลิเคชั่นเพียง 45.25 มิลลิวินาที และ 50.96 มิลลิวินาที ตามลำดับ เนื่องจากขนาดของพื้นที่สำหรับจัดเก็บแบบจำลอง NU-InNet 1.0 และ 1.1 (ความลึกที่ 1) มีขนาดเพียง 1.67 MB และ 1.47 MB ตามลำดับ จากนั้นเมื่อนำแบบจำลอง NU-InNet 1.0 และ 1.1 (ความลึกที่ 4) มาตรวจวัดพบว่าใช้เวลา 98.40 มิลลิวินาที และ 110.64 มิลลิวินาที ซึ่งเมื่อนำเวลาที่ใช้ในการเปิดแอพพลิเคชั่นแบบจำลอง NU-InNet 1.0 และ 1.1 (ความลึกที่ 4) ไปเปรียบกับแบบจำลอง BN-Inception พบว่าแบบจำลอง NU-InNet 1.0 และ 1.1 (ความลึกที่ 4) ใช้เวลาน้อยกว่าถึง 6.9 และ 6.1 เท่า ตามลำดับ

หลังจากแอพพลิเคชั่นพร้อมใช้งานเรียบร้อยแล้ว ถัดมาในส่วนของเวลาที่ใช้ในการรู้จำภาพได้ทำการทดลองนำภาพที่มีขนาด 1080x1080 จุดภาพ มารู้จำภาพและตรวจวัดเวลาที่ใช้ จากตาราง 10 เมื่อพิจารณาเวลาที่ใช้ในการรู้จำภาพของแบบจำลองที่ชนะการแข่งขัน ILSVRC พบว่าแบบจำลอง AlexNet ใช้เวลาในการรู้จำภาพ 589 มิลลิวินาที ต่อ 1 ภาพ ซึ่งน้อยกว่าแบบจำลอง GoogLeNet ที่ใช้เวลา 937 มิลลิวินาที ต่อ 1 ภาพ โดยผลการทดลองของส่วนนี้มีลักษณะเช่นเดียวกับการวัดประสิทธิภาพด้านเวลาบนคอมพิวเตอร์ ซึ่งมีผลมาจากการออกแบบโครงสร้างของแต่ละแบบจำลอง หากแบบจำลองมีความซับซ้อนมากจะส่งผลให้ใช้เวลาในการประมวลผลเพิ่มมากขึ้น และเมื่อพิจารณาแบบจำลอง NU-InNet 1.0 และ 1.1 (ความลึกที่ 1) พบว่าทั้งสองแบบจำลองใช้เวลาในการรู้จำภาพ 554 และ 633 มิลลิวินาที ต่อ 1 ภาพ ตามลำดับ และหากพิจารณาแบบจำลอง NU-InNet 1.0 และ 1.1 (ความลึกที่ 4) พบว่าทั้งสองแบบจำลองใช้เวลาในการรู้จำภาพ 603 และ 699 มิลลิวินาที ต่อ 1

ภาพ ตามลำดับ จากข้อมูลดังกล่าวแสดงให้เห็นว่าแบบจำลอง NU-InNet ทั้งสี่ ใช้เวลาในการรู้จำภาพน้อยกว่าแบบจำลอง GoogLeNet และ BN-Inception



ภาพ 37 เวลาที่ใช้ผู้ใช้เปิดแอพพลิเคชั่นและรู้จำภาพถ่ายจำนวน 1 ภาพ บนสมาร์ทโฟน

เมื่อนำเวลาที่ใช้ในการเปิดแอพพลิเคชั่นและเวลารู้จำภาพถ่ายจำนวน 1 ภาพ มารวมกัน เพื่อตรวจวัดเวลาที่ผู้ใช้ต้องการรู้จำภาพถ่ายจำนวน 1 ภาพ ต้องใช้เวลาเท่าใด จึงสามารถแสดงผลลัพธ์ให้กับผู้ใช้ได้ โดยแสดงในหน่วยของวินาที ดังภาพ 37 เมื่อพิจารณาแบบจำลองที่ชนะการแข่งขัน ILSVRC พบว่าแบบจำลอง AlexNet ที่ใช้เวลามากที่สุดถึง 4.92 วินาที สำหรับแบบจำลอง GoogLeNet ใช้เวลาเพียง 1.49 วินาที และสำหรับแบบจำลอง BN-Inception ที่พัฒนามาจากแบบจำลอง GoogLeNet ใช้เวลา 2.13 วินาที และเมื่อพิจารณาทั้งแบบจำลอง NU-InNet 1.0 กับ 1.1 (ความลึกที่ 1) และ NU-InNet 1.0 กับ 1.1 (ความลึกที่ 4) พบว่าใช้เวลาในการรู้จำภาพถ่ายจำนวน 1 ภาพ ไม่ถึง 1 วินาทีทั้งหมด ซึ่งใช้เวลาน้อยกว่าแบบจำลองที่ชนะการแข่งขัน ILSVRC ดังนั้นหากต้องการพัฒนาแอพพลิเคชั่นสำหรับรู้จำภาพถ่ายอาหารไทยบนสมาร์ทโฟน ควรเลือกแบบจำลอง NU-InNet รุ่น 1.0 กับ 1.1 (ความลึกที่ 1 หรือ 4) เนื่องจากการเปิดแอพพลิเคชั่นและรู้จำภาพถ่ายของแบบจำลอง NU-InNet ใช้เวลาเพียง 0.6 ถึง 0.81 วินาที

ผลการทดลองจาก 3 การทดลองนี้ แสดงถึงการออกแบบโครงสร้างของแบบจำลองสำหรับการรู้จำภาพอาหารไทย ด้วยโครงข่ายประสาทเทียมแบบลึกบนสมาร์ทโฟน เริ่มตั้งแต่ การเปรียบเทียบประสิทธิภาพทั้งด้านความถูกต้อง เวลาที่ใช้ในการประมวลผล และขนาดสำหรับจัดเก็บของแบบจำลองที่ออกแบบขึ้นมากับแบบจำลองที่ชนะการแข่งขัน ILSVRC จากนั้นได้ทดลองนำแบบจำลองที่ออกแบบจากการแบบจำลอง NU-InNet มาปรับปรุงเพื่อเพิ่มประสิทธิภาพด้านความถูกต้องให้สูงขึ้น หลังจากทดลองเสร็จสิ้นได้นำแบบจำลองที่ออกแบบมาตรวจวัดประสิทธิภาพด้านเวลาบนอุปกรณ์สมาร์ทโฟน

บทที่ 5
สรุปผลการทดลอง

งานวิจัยชิ้นนี้ได้นำเสนอการรู้จำภาพอาหารไทยบนสมาร์ทโฟน ซึ่งแอพพลิเคชั่นสำหรับรู้จำภาพอาหารไทย มีความสำคัญอย่างยิ่งต่อนักท่องเที่ยวชาวต่างชาติ เนื่องจากนักท่องเที่ยวสามารถทราบชื่อของอาหารได้จากถ่ายภาพ จากนั้นใช้แอพพลิเคชั่นเพื่อระบุชื่อของอาหาร ซึ่งชื่อของอาหารถือว่าจำเป็นอย่างมากสำหรับนักท่องเที่ยว นอกจากใช้การสั่งอาหารแล้วยังสามารถช่วยให้ค้นหารายละเอียด ส่วนประกอบ ปริมาณแคลอรี่ และร้านค้าที่ใกล้เคียงได้ สำหรับแอพพลิเคชั่นรู้จำภาพส่วนใหญ่ สมาร์ทโฟนเป็นอุปกรณ์สำหรับรับส่งข้อมูลเท่านั้น สมาร์ทโฟนทำหน้าที่ส่งภาพไปยังคอมพิวเตอร์ผ่านเครือข่ายอินเทอร์เน็ต เพื่อประมวลผลจากนั้นส่งข้อมูลกลับมายังสมาร์ทโฟน เพื่อแสดงให้กับผู้ใช้งาน ดังนั้นงานวิจัยชิ้นนี้จึงทำการศึกษาปัญหาและปัจจัยในด้านต่าง ๆ ของแอพพลิเคชั่นรู้จำภาพ เพื่อนำไปพัฒนาแอพพลิเคชั่นสำหรับรู้จำภาพอาหารไทยให้มีประสิทธิภาพที่ดีขึ้น โดยเริ่มจากการศึกษาปัจจัยสำคัญที่ส่งผลกับความถูกต้องในการระบุชื่อของอาหาร ปัจจัยที่ส่งผลต่อความเร็วในการประมวลผล และปัจจัยที่ส่งผลต่อขนาดของแอพพลิเคชั่น ซึ่งปัจจัยดังกล่าวสามารถแก้ไขได้โดยใช้โครงข่ายประสาทเทียมแบบลึกที่สามารถเรียนรู้และจำแนกประเภทของภาพได้อย่างมีประสิทธิภาพ โดยจะวิเคราะห์องค์ประกอบของภาพได้แม่นยำ ด้วยวิธีการแบ่งภาพออกเป็นส่วนเล็ก ๆ แล้วแทนด้วยตัวเลขเพื่อจัดเก็บและตีความหมาย เช่นเดียวกับการทำงานของสมองมนุษย์ ดังนั้นเมื่อนำโครงข่ายประสาทเทียมแบบลึกมาประยุกต์ใช้ร่วมกับการรู้จำภาพอาหารไทย จึงทำให้เพิ่มความสะดวกให้กับนักท่องเที่ยวได้ เนื่องจากการทำงานของโครงข่ายประสาทเทียมแบบลึกสามารถนำไปใช้งานกับอุปกรณ์สมาร์ทโฟนได้ โดยไม่จำเป็นต้องมีการเชื่อมต่ออินเทอร์เน็ต แต่เนื่องจากการออกแบบสถาปัตยกรรมของโครงข่ายประสาทเทียมแบบลึกนั้น ส่วนใหญ่มุ่งเน้นไปที่การเพิ่มประสิทธิภาพด้านความถูกต้องเป็นหลัก โดยไม่ได้ให้ความสำคัญกับเวลาที่ใช้ในการประมวลผลและขนาดของแบบจำลองโครงข่ายประสาทเทียมแบบลึก ส่งผลให้การนำโครงข่ายประสาทเทียมแบบลึกไปใช้บนสมาร์ทโฟนมีข้อจำกัด ในด้านเวลาในการประมวลผลและขนาดของแอพพลิเคชั่น ดังนั้นงานวิจัยชิ้นนี้จึงออกแบบและพัฒนาแบบจำลองของโครงข่ายประสาทเทียมแบบลึกใหม่ โดยมีวัตถุประสงค์เพื่อลดเวลาในการประมวลผลและขนาดของพื้นที่สำหรับจัดเก็บแบบจำลองลง เพื่อให้เหมาะสมสำหรับการนำไปใช้งานบนสมาร์ทโฟน นอกจากนี้ยังได้จัดเก็บและรวบรวมภาพถ่ายอาหารไทยที่ได้รับความนิยม 50 ประเภท (THFOOD-50) เพื่อให้โครงข่ายประสาทเทียมแบบลึกฝึกฝนและเรียนรู้ ซึ่งหลังจากกระบวนการเรียนรู้เสร็จสิ้น จึงได้นำแบบจำลองที่มีความเหมาะสมที่สุดไปใช้กับสมาร์ทโฟน เพื่อให้การทำงานของแอพพลิเคชั่นมีประสิทธิภาพสูงขึ้นเมื่อเปรียบเทียบกับแบบจำลองเดิมที่ใช้ในปัจจุบัน

ผลการทดลอง

การศึกษาและการทดลองระหว่างแบบจำลอง NU-InNet กับแบบจำลองที่ชนะการแข่งขัน ILSVRC นั้นมีปัจจัยในด้านประสิทธิภาพความถูกต้อง เวลาที่ใช้ในการประมวลผล รวมไปถึงการใช้หน่วยความจำ ทั้งสามปัจจัยคือสิ่งที่ต้องคำนึงถึงเมื่อพัฒนาแอพพลิเคชั่นบนสมาร์ทโฟน จากการทดลองพบว่าแบบจำลอง AlexNet มีประสิทธิภาพด้านความถูกต้องต่ำ ใช้เวลาในการประมวลผลต่ำ และแบบจำลองมีขนาดใหญ่ สำหรับแบบจำลอง GoogLeNet มีประสิทธิภาพด้านความถูกต้องสูง ใช้เวลาในการประมวลผลปานกลาง และแบบจำลองมีขนาดปานกลาง สำหรับแบบจำลอง NU-InNet ทั้งสองรุ่นมีประสิทธิภาพด้านความถูกต้องสูง ใช้เวลาในการประมวลผลต่ำ และแบบจำลองมีขนาดเล็ก ดังนั้นแบบจำลอง NU-InNet จึงมีประสิทธิภาพโดยรวมเหมาะสมมากกว่าแบบจำลองที่ชนะการแข่งขัน ILSVRC เมื่อนำไปใช้บนสมาร์ทโฟนด้วยฐานข้อมูลภาพถ่ายอาหารไทย เนื่องจากแบบจำลอง NU-InNet ที่ออกแบบมาแล้วนั้นมีประสิทธิภาพด้านความถูกต้องที่ยังไม่สูงมาก จึงออกแบบและพัฒนาแบบจำลองใหม่ ด้วยการเพิ่มจำนวนความลึกของแบบจำลองให้มากขึ้น โดยได้ทดลองเพิ่มความลึก 3 ระดับ คือความลึกที่ 4, 8 และ 12 ดังนั้นแบบจำลอง Deep NU-InNet จึงประกอบไปแบบจำลองรุ่น 1.0 และ 1.1 แต่ละรุ่นมีความลึกจำนวน 4 ระดับ ได้แก่ ความลึกที่ 1 ความลึกที่ 4 ความลึกที่ 8 และ ความลึกที่ 12 จากการทดลองระหว่างแบบจำลอง Deep NU-InNet กับแบบจำลอง GoogLeNet จะเห็นได้ว่าการเพิ่มจำนวนความลึกเข้าไปในแบบจำลอง NU-InNet สามารถเพิ่มประสิทธิภาพด้านความถูกต้องได้ แต่เวลาที่ใช้ในการประมวลผลและขนาดของแบบจำลองเพิ่มสูงขึ้นเช่นเดียวกัน ดังนั้นระดับความลึกที่มีประสิทธิภาพโดยรวมดีที่สุดคือ ระดับความลึกที่ 4 ซึ่งมีประสิทธิภาพด้านความถูกต้องสูง 80.34% สำหรับการพยากรณ์ลำดับที่ 1 ใช้เวลาในการประมวลผลปานกลาง และแบบจำลองมีขนาดเล็ก 3.66 เมกะไบต์

หลังจากกระบวนการฝึกฝนเสร็จสิ้น จึงได้นำแต่ละแบบจำลองที่ได้จากการทดลองข้างต้น มาตรวจวัดเวลาที่ผู้ใช้ต้องการรู้จำภาพถ่ายจำนวน 1 ภาพ ต้องใช้เวลาเท่าใด ถึงจะแสดงผลลัพธ์ให้กับผู้ใช้ได้ จากการตรวจวัดเวลาให้ผลลัพธ์ว่าแบบจำลอง AlexNet ใช้เวลา 4.92 วินาที แบบจำลอง GoogLeNet ใช้เวลา 1.49 วินาที และแบบจำลอง NU-InNet รุ่น 1.0 กับ 1.1 (ความลึกที่ 1 กับ 4) ใช้เวลา 0.6 ถึง 0.81 วินาที จากข้อมูลแสดงให้เห็นว่าแบบจำลอง NU-InNet ใช้เวลาในการรู้จำภาพไม่ถึงหนึ่งวินาที ดังนั้นเมื่อพิจารณาข้อมูลที่ได้จากการทดลองทั้งสามสามารถสรุปได้ว่า แบบจำลองของโครงข่ายประสาทเทียมแบบลึกที่เรียนรู้ด้วยชุดข้อมูลภาพถ่ายอาหารไทยจำนวน 50 ประเภท ที่มีประสิทธิภาพทั้งด้านความถูกต้อง เวลาที่ใช้ในการประมวลผล และขนาดของแบบจำลอง เหมาะสมที่สุดสำหรับนำไปใช้งานบนอุปกรณ์สมาร์ทโฟน คือแบบจำลอง NU-InNet รุ่น 1.1 ความลึกที่ 4 ที่มีแบบจำลองขนาด 3.66 เมกะไบต์ และใช้เวลารู้จำบนสมาร์ทโฟน 0.81 วินาที

ปัญหาและแนวทางการแก้ไข

ปัญหาที่พบจากการทำงานวิจัยชิ้นนี้คือ กระบวนการฝึกฝนของโครงข่ายประสาทเทียมแบบลึกจำเป็นต้องใช้คอมพิวเตอร์ที่มีประสิทธิภาพสูงที่ติดตั้งหน่วยประมวลผลกราฟิกส์และสามารถใช้งานได้ตลอด 24 ชั่วโมง เนื่องจากกระบวนการฝึกฝนของแต่ละแบบจำลองใช้เวลาตั้งแต่ 50 ถึง 440 ชั่วโมง สำหรับชุดข้อมูลภาพถ่ายอาหารไทย 50 ประเภท มีจำนวนภาพถ่ายทั้งหมด 15,770 ภาพ ทำให้การทดลองเพื่อตรวจวัดและเก็บข้อมูลในแต่ละแบบจำลองใช้เวลาจำนวนมากถึงจะทราบผลลัพธ์ หากนำโครงข่ายประสาทเทียมแบบลึกไปฝึกฝนด้วยชุดข้อมูลอื่น ๆ ที่มีปริมาณของภาพจำนวนมาก เช่น ชุดข้อมูล ILSVRC หรือ Places365 อาจส่งผลให้เวลาที่ใช้สำหรับการฝึกฝนนานถึง 1 เดือน เนื่องจากสมมติฐานของงานวิจัยชิ้นนี้คือภาพถ่ายที่นำมาใช้ต้องมีอาหารปรากฏในภาพเพียง 1 ประเภท หากนำภาพถ่ายที่มีอาหารมากกว่า 1 ประเภทมาใช้อาจจะทำให้ประสิทธิภาพด้านความถูกต้องลดลง แนวทางการแก้ไขคือ ใช้วิธีระบุตำแหน่งของอาหารในภาพ (Object detection) หรือใช้วิธีตัดหรือนำบางส่วนที่ไม่เกี่ยวข้องกับอาหารออกจากภาพ (Image segmentation) เพื่อช่วยลดความผิดพลาดที่อาจจะเกิดขึ้นในกระบวนการรู้จำภาพถ่ายอาหารด้วยโครงข่ายประสาทเทียมแบบลึก สำหรับแบบจำลองของโครงข่ายประสาทเทียมแบบลึกที่ฝึกฝนหรือเรียนรู้กับชุดข้อมูลหนึ่งแล้วมีประสิทธิภาพสูง แต่เมื่อนำมาใช้กับชุดข้อมูลอื่น ๆ อาจทำให้ไม่ได้ประสิทธิภาพเช่นเดียวกับชุดข้อมูลแรก เช่น เมื่อนำแบบจำลองที่ชนะการแข่งขัน ILSVRC มาใช้กับชุดข้อมูลภาพถ่ายอาหารไทย 50 ประเภท พบว่าแบบจำลองที่ชนะการแข่งขัน ILSVRC มีประสิทธิภาพด้อยกว่าแบบจำลอง NU-InNet ที่ออกแบบและพัฒนาขึ้นในงานวิจัยชิ้นนี้ เนื่องจากแต่ละแบบจำลองได้ถูกออกแบบและทดลองกับชุดข้อมูลนั้น ๆ เมื่อนำไปใช้งานกับชุดข้อมูลอื่นจึงให้ผลลัพธ์ที่แตกต่างกัน ดังนั้นหากต้องการนำโครงข่ายประสาทเทียมแบบลึกไปใช้กับชุดข้อมูลอื่น ควรที่จะออกแบบและทดลองเพื่อเปรียบเทียบแบบจำลองที่มีความเหมาะสมกับชุดข้อมูลที่นำมาใช้ที่สุด สำหรับชุดข้อมูลจัดได้ว่าเป็นส่วนสำคัญสำหรับโครงข่ายประสาทเทียมแบบลึก เนื่องจากกระบวนการเรียนรู้ยังคงต้องอาศัยการเรียนรู้แบบมีผู้สอนซึ่งต้องให้มนุษย์ป้อนข้อมูลให้ ทำให้การสร้างชุดข้อมูลต้องมีความหลากหลายของภาพถ่ายภายในกลุ่มหรือประเภทเดียวกัน ซึ่งหากมีปริมาณจำนวนของภาพมาก ทำให้โครงข่ายประสาทเทียมแบบลึกสามารถเรียนรู้มีประสิทธิภาพมากขึ้น ดังที่ได้กล่าวอธิบายถึงปัญหาไว้ข้างต้นนั้น การพัฒนาแอพพลิเคชั่นสำหรับรู้จำภาพอาหารไทยด้วยโครงข่ายประสาทเทียมแบบลึกบนสมาร์ทโฟนในงานวิจัยชิ้นนี้ ได้ทดลองใช้ชุดข้อมูลภาพถ่ายอาหารไทยจำนวน 50 ประเภทเท่านั้น เพื่อให้แบบจำลอง NU-InNet มีประสิทธิภาพที่น่าเชื่อถือ จึงควรนำแบบจำลอง NU-InNet ไปทดสอบกับชุดข้อมูลภาพถ่ายอาหารอื่น ๆ เช่น ชุดข้อมูลภาพถ่ายอาหารญี่ปุ่น (UEC-100 และ UEC-256 [30]) หรือ ชุดข้อมูลภาพถ่ายอาหารตะวันตก (Food-101) [31] เป็นต้น เนื่องจากแต่ละชุดข้อมูลมีประเภทของอาหารมากกว่า 100 ประเภท และหากต้องการเพิ่มประสิทธิภาพให้กับแบบจำลอง NU-InNet ควรกำหนดจำนวนความลึกและจำนวนคุณลักษณะที่ได้ให้เหมาะสมกับชุดข้อมูลที่นำมาเรียนรู้ หรือเพิ่มเทคนิค

ต่าง ๆ เข้าไป เช่น เทคนิค Identity Mapping จากแบบจำลอง ResNet โดยพิจารณาตามความเหมาะสมจากชุดข้อมูลหรือวัตถุประสงค์ของการนำไปใช้งาน

บรรณานุกรม

[1] C. M. Eastman, "Vector versus raster: a functional comparison of drawing technologies," *Computer Graphics and Applications, IEEE,* vol. 10, pp. 68-80, 1990.

[2] A. A. DESAI, *Computer Graphics*: PHI Learning, 2008.

[3] R. Hassanpour, A. Shahbahrami, and S. Wong, "Adaptive Gaussian mixture model for skin color segmentation," *World Academy of Science, Engineering and Technology,* vol. 41, pp. 1-6, 2008.

[4] M. W. Schwarz, W. B. Cowan, and J. C. Beatty, "An experimental comparison of RGB, YIQ, LAB, HSV, and opponent color models," *ACM Transactions on Graphics (TOG),* vol. 6, pp. 123-158, 1987.

[5] P. Shih and C. Liu, "Comparative assessment of content-based face image retrieval in different color spaces," *International Journal of Pattern Recognition and Artificial Intelligence,* vol. 19, pp. 873-893, 2005.

[6] S. Tangkawanit and S. Kanprachar, "Performance of HSV Color System in Vehicle Detection Application under Different Illumination Intensities," *ITC-CSCC: 2009,* pp. 525-528, 2009.

[7] S. Y. Kung, *Kernel methods and machine learning*: Cambridge University Press, 2014.

[8] S. Dumais, J. Platt, D. Heckerman, and M. Sahami, "Inductive learning algorithms and representations for text categorization," in *Proceedings of the seventh international conference on Information and knowledge management*, 1998, pp. 148-155.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, *et al.*, "Scikit-learn: Machine learning in Python," *The Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

[10] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: foundations and applications* vol. 207: Springer, 2008.

[11] H. Su, M. Sun, L. Fei-Fei, and S. Savarese, "Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object

categories," in *Computer Vision, 2009 IEEE 12th International Conference on*, 2009, pp. 213-220.

[12] A. Kuznetsova, S. J. Hwang, B. Rosenhahn, and L. Sigal, "Exploiting View-Specific Appearance Similarities Across Classes for Zero-shot Pose Prediction: A Metric Learning Approach," 2016.

[13] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, "Recent advances of large-scale linear classification," *Proceedings of the IEEE*, vol. 100, pp. 2584-2603, 2012.

[14] R. Moore and J. DeNero, "L1 and L2 regularization for multiclass hinge loss models," in *MLSLP*, 2011, pp. 1-5.

[15] L. Rosasco, E. De Vito, A. Caponnetto, M. Piana, and A. Verri, "Are loss functions all the same?," *Neural Computation*, vol. 16, pp. 1063-1076, 2004.

[16] K.-B. Duan and S. S. Keerthi, "Which is the best multiclass SVM method? An empirical study," in *Multiple classifier systems*, ed: Springer, 2005, pp. 278-285.

[17] D. Meyer and F. T. Wien, "Support vector machines," *The Interface to libsvm in package e1071*, 2015.

[18] C. M. Bishop, "Pattern Recognition," *Machine Learning*, 2006.

[19] B. Krishnapuram, L. Carin, M. A. Figueiredo, and A. J. Hartemink, "Sparse multinomial logistic regression: Fast algorithms and generalization bounds," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, pp. 957-968, 2005.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097-1105.

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278-2324, 1998.

[22] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer vision–ECCV 2014*, ed: Springer, 2014, pp. 818-833.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1-9.

[24] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of The 32nd International Conference on Machine Learning*, 2015, pp. 448-456.

[25] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *arXiv preprint arXiv:1512.00567*, 2015.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, 2016.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks," in *Computer Vision–ECCV 2016*, 2016.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026-1034.

[29] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 675-678.

[30] K. Yanai and Y. Kawano, "Food image recognition using deep convolutional network with pre-training and fine-tuning," in *Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on*, 2015, pp. 1-6.

[31] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101–mining discriminative components with random forests," in *Computer Vision–ECCV 2014*, ed: Springer, 2014, pp. 446-461.

ภาคผนวก

- ผลงานวิจัย ที่ถูกตีพิมพ์เผยแพร่ ประเภทวารสาร ระดับนานาชาติ

Termritthikun C, Muneesawang P, Kanprachar S. "NU-InNet: Thai Food Image Recognition Using Convolutional Neural Networks on Smartphone." Journal of Telecommunication, Electronic and Computer Engineering (JTEC) 9.2-6 (2017): 63-67.

- ผลงานวิจัย ที่ถูกนำเสนอในการประชุมวิชาการ ระดับนานาชาติ

Termritthikun C, Kanprachar S. "Accuracy Improvement of Thai Food Image Recognition Using Deep Convolutional Neural Networks." Electrical Engineering Congress (iEECON), 2017 International; 2017.

NU-InNet: Thai Food Image Recognition Using Convolutional Neural Networks on Smartphone

Chakkrit Termritthikun, Paisarn Muneesawang and Surachet Kanprachar
Department of Electrical and Computer Engineering, Faculty of Engineering,
Naresuan University, Phitsanulok, Thailand.
surachetka@nu.ac.th

***Abstract*—Currently, Convolutional Neural Networks (CNN) have been widely used in many applications. Image recognition is one of the applications utilizing CNN. For most of the research in this field, CNN is used mainly to increase the effectiveness of the recognition. However, the processing time and the amount of the parameters (or model size) are not taken into account as the main factors. In this paper, the image recognition for Thai food using a smartphone is studied. The processing time and the model size are reduced so that they can be properly used with smartphones. A new network called NU-InNet (Naresuan University Inception Network) that adopts the concept of Inception module used in GoogLeNet is proposed in the paper. It is applied and tested with Thai food database called THFOOD-50, which contains 50 kinds of famousThai food. It is found that NU-InNet can reduce the processing time and the model size by the factors of 2 and 10, respectively, comparing to those obtained from GoogLeNet while maintaining the recognition precision to the same level as GoogLeNet. This significant reduction in the processing time and the model size using the proposed network can certainly satisfy users for Thai-food recognition application in a smartphone.**

***Index Terms*—Deep Learning; Food Recognition; Convolutional Neural Networks; Smartphone; Thai Food; Dataset; Inception.**

I. Introduction

Food image recognition is one of the crucial applications used these days. It allows smartphone users to know the name of the food. This is quite important for travelers who travel to foreign countries. It also helps them to be able to order food properly and know the information about the food, for example the amount of calories, possible allergies, and so on. Currently, image recognition application in a smartphone mainly requires a computer [1,2] since the recognition process requires a considerably amount of resources to serve the used database. If the size of the database is large, the limited resource in a smartphone cannot keep up processing the data. Thus, the smartphone must send the data to be processed at the third-party computer. The effectiveness of the recognition in this case then depends on the performance of the computer and the speed of the internet connection.

Research on food image recognition, however, has been mainly focused on the correctness [2–6] of the food name for the given food image. Many techniques are applied, for example using image segmentation [7, 8] to separate the food from the background image. This technique will increase the effectiveness of the food identification. However, it is not appropriate to be used in a smartphone application since more image processing is needed. To overcome such problem, one possible approach called Convolutional Neural Network (CNN) can be adopted. At present, CNN has been used widely with image recognition. In the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) for the year 2012, AlexNet [9] won the competition by using the resized RGB (Red, Green, and Blue) images as the input for CNN to learn and produce the output probabilities of the categorized classes. Further, CNN has been adopted by the winning competitors of ILSVRC, that are GoogLeNet [10] and ResNet [11] in 2014 and 2015, respectively.

CNN has been continuously studied and developed so that the recognition of the effectiveness of CNN is higher than that from the conventional techniques used in computer vision. CNN can help extracting the features including colors, textures, and shapes. Moreover, image classification can also be done by CNN. In a research [12], CNN has been applied to food image recognition by adopting AlexNet [9] and tuning the output in the fully connected layer from 4,096 to 6,144. With a large-scale food database of 2,000 categories, the pre-training technique was done with the tuned AlexNet, resulting in the extracted features with 6,144 vectors per image. These extracted features were tested with food images from the databases UEC-FOOD100 and UEC-FOOD256. It was found that the correctness of the food image recognition was improved. However, with AlexNet [9,12], the storage required is quite large; that is, at least 240 MB, which is not suitable to be used with a smartphone. SqueezeNet [13] has been developed to reduce the storage size, while keeping the effectiveness of the recognition. The storage size was lessened by the factor of 50 that is, 4.8 MB.

To further improve the effectiveness of recognition, GoogLeNet [10] was proposed by Google. It was developed under the concept of Inception module that allows CNN to analyze an image with 1×1, 3×3, and 5×5 filters. Having done these, the obtained effectiveness of recognition was better than that of AlexNet. Additionally, the dimension reduction can be achieved by adding a 1×1 convolutional layer prior to the 3×3 or 5×5 convolutional layers. The required storage was reduced by the factor of 4.8, in comparison to that of AlexNet, in which only 51.1 MB was needed.

Considering the Food Image Recognition in a smartphone, it can seen that not just the correctness of the recognition, but the processing time and the required storage (or parameters) are important as well. In this paper, all these three important factors are taken into account. A new network called NU-InNet (Naresuan University Inception Network) is proposed. The concept of the inception module

adopted by GoogLeNet was utilized and further improved. The aim is to reduce the processing time and the parameters in comparison to those obtained from GoogLeNet, while maintaining the recognition accuracy to be at the same level.

The organization of this paper is done as follows. The related technology is given in Section II. The proposed network in this paper is explained in Section III. In Section IV, the obtained results are presented and discussed. Finally, the conclusion is given in Section V

II. Related Technology

Considering the current computer technology, it is seen that the data can be processed much faster utilizing GPU (Graphics Processing Unit) to function with CPU (Central Processing Unit). This then allows the deep learning technique, which is widely used in object detection and image recognition, to perform much better in terms of the resulting accuracy in detection and recognition. Convolutional Neural Network (CNN), one of the variants of deep learning technique, is adopted in this paper to be used for Thai food image recognition. In this section, the detail about CNN and related ones will be given.

A. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a type of feed-forward artificial neural network (ANN). It contains different layers, including the input layer, convolutional layer, pooling layer, and fully-connected layer. These layers are stacked on top of each other according to the CNN architecture in order to do the recognition tasks. These layers are explained briefly as follows.

Input layer is the layer that contains images of the training data and testing data. These images are in the format of RGB and the image size depends on the model used in the network. For example, an image of 256 × 256 pixels, the data contained in such image equal [256×256×3], where the number 3 refers to the 3 channels of RGB.

Convolutional layer is the important layer of CNN where the dot product between the filter and the particular volume of the input data is determined according to the filter size. The product starts from the position (0, 0) of the input data and moves one pixel (stride 1) at a time from the left to the right and from the top to the bottom of the image. The obtained output from these products is the activation map. For example, with a data of [224×224×3] and 96 3×3 filters, moving the filter 2 pixels (stride 2) at a time, the size of the achieved activation map will be [111×111×96].

Pooling layer is the layer put after a convolutional layer in order to reduce the representation parameters of the network, resulting in a less computational process for the following steps. The function to be used in this layer can be one of the non-linear functions, for example max pooling, average pooling, L2 pooling, and so on. Among these functions, max pooling is currently the most common pooling to be used since it has shown to give a better performance. For the max pooling, the maximum value of the considered elements in the filtering area will be selected. The parameters of the whole network will be reduced depending on the size of the filter and the striding step. For example, with the input data sized [111×111×96], using a 3×3 filter with striding step of 2, the resulting data size will be reduced to [55×55×96].

Fully-connected layer is the layer put at the end of the network. All activations in the preceding layers are connected to this layer. The layer reduces the size of the data to be one-dimension data.

CNN has been developed to recognize the data in the deep level by adding more hidden layers to the network. An image can be recognized in three dimensions: the width, the height, and the depth. The network will divide the image into parts and analyze each to extract the important features, for example, colors, shapes, textures, and so on. These features can certainly be used to classify the image.

B. Deep Learning Framework [14]

In order to develop CNN, the framework to be used must be specifically designed. The designed framework can be chosen depending on the computer language or the operation system that users are working with. Considering this research, it is focused on the application to be used with an Android smartphone, hence, one possible choice is the framework called Caffe, which is widely used, developed with C++. Normally, to develop an application for an Android device, Java is commonly used. Hence, to work with Caffe framework in order to develop an Android application, certainly C/C++ has to be adopted, Native Development Kit (NDK) has to be used.

III. The Proposed Network

In this section, the dataset, the proposed network architecture, and the implementation of such network in Thai food image recognition in a smartphone will be described.

A. THFOOD-50 Dataset

The dataset of 50 kinds of Thai popular food images were collected as shown in Figure 1. These images were collected from search engines, namely the Google, Bing, and Flickr. In each kind of Thai food, there are approximately 200 to 700 images. These images are divided into 2 groups; that is, 90% of the images are in the training group and 10% of the images are in the testing group. The images are resized to have the size of 256×256 pixels suiting the CNN model to be used.

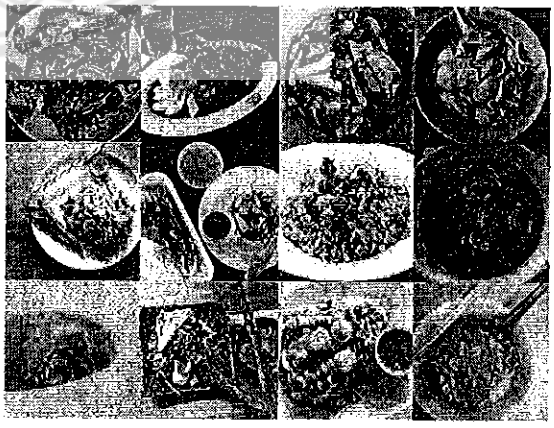


Figure 1: Examples of Thai food images in the THFOOD-50 dataset

B. Proposed Network Architecture

Inception module used in GoogLeNet10 is adopted in the proposed network. Two versions of the network are

proposed. The architectures of these inception modules are shown in Figure 2.

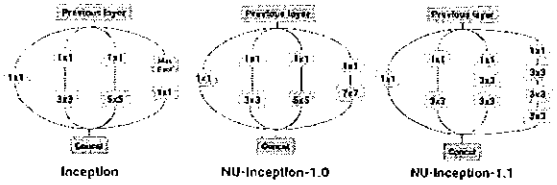


Figure 2: Module architecture of Inception (left), NU-Inception-1.0 (middle), and NU-Inception-1.1 (right)

Table 1
Detail of layers in NU-InNet 1.0

Type	Patch Size/Stride	Output Size
Input Image	-	224×224×3
Convolution	7×7/2	109×109×96
Max Pool	3×3/2	54×54×96
Convolution	1×1/1	54×54×96
Convolution	5×5/2	25×25×96
Max Pool	3×3/2	12×12×96
1× NU-Inception 1.0	As in Figure 2	12×12×256
Average Pool	-	1×1×256

Table 2
Detail of layers in NU-InNet 1.1

Type	Patch Size/Stride	Output Size
Input Image	-	224×224×3
Convolution	3×3/2	111×111×32
Convolution	3×3/1	109×109×32
Convolution	3×3/1	109×109×64
Max Pool	3×3/2	54×54×64
Convolution	1×1/1	54×54×64
Convolution	5×5/2	25×25×96
Max Pool	3×3/2	12×12×96
1× NU-Inception 1.1	As in Figure 2	12×12×256
Average Pool	-	1×1×256

a. NU-InNet 1.0

The inception module (as shown in Figure 2 (left)) is modified by changing its 3×3 max pooling layer and 1×1 convolutional layer to be 1×1 and 7×7 convolutional layers, respectively, as shown in Fig.2 (middle). The filter weights are set by "Xavier" with a constant filter bias of 0.2. And, for down sampling the size of the activation map after convolutional layers, the average pooling layer with a stride of 2 is used. The detail of this proposed NU-InNet 1.0 network is shown in Table 1 and Figure 3. There are totally 16 layers used in this network as shown in Figure 3 (middle). The details of these layers are given in Table 1.

b. NU-InNet 1.1

For this proposed network, the NU-Inception 1.0 module is modified by changing [15,16] any 5×5 convolutional layer to be 2 3×3 convolutional layers and changing any 7×7 convolutional layer to be 3 3×3 convolutional layers. These changes can be viewed in Figure 2 (right). By doing this, the processing time in the module will be lessened. Similarly, the detail of this proposed NU-InNet 1.1 network is shown in Table 2 and Figure 3. There is a total of 21 layers used in this network, as shown in Figure 3 (right). The detail of these layers is given in Table 2.

C. Implementation

The speed of data processing and the size of model are very crucial in designing a neural network, especially for its usage with a smartphone. In the proposed networks, one of the benchmark is the recognition accuracy that has to be at least not poorer than that from GoogLeNet, while the processing time and model size have to be lower. To obtain these properties, the number of modules in the proposed networks is set to be one module; while in GoogLeNet, nine modules were used as seen in Figure 3. The use of less number of modules decrease the processing time and model size; however, the recognition efficiency is also lessened. To improve the recognition efficiency, Batch Normalization [16] has to be put at the end of each convolutional layer as performed in ResNet [11]. By doing this, the training accuracy can be improved.

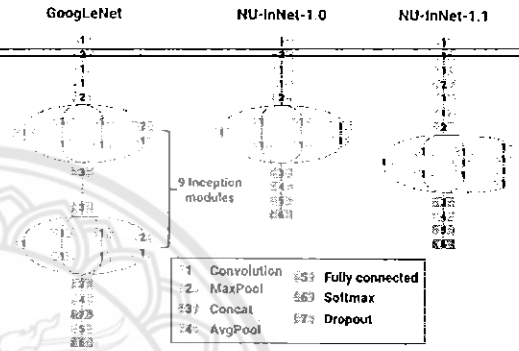


Figure 3: GoogLeNet, NU-InNet 1.0, and NU-InNet 1.1 architectures

IV. Results and Discussion

The proposed networks were used in Thai food image recognition. For the training process, a HPC (High-Performance Computer) with specifications Inter(R) Xeon(R) E5-2683 v3 @2.00-GHz 56-Core CPU, 64-GB RAM, and NVIDIA Tesla K80 GPU were used with the operating system Ubuntu Server 14.04.5 and Caffe14. For the testing process, a smartphone with specifications Intel(R) Atom(TM) Z3580 @2.33-GHz 4-Core CPU and 4-GB RAM was used with Android 6.0.1 operating system.

A. Comparisons between NU-InNets and Winning Models from ILSVRC

For the training process, AlexNet, GoogLeNet, SqueezeNet, NU-InNet 1.0, and NU-InNet 1.1 were trained from scratch. The database was divided into 2 parts; that is, 90% for training and 10% for testing. To get reliable testing results, 10-fold-cross-validation was used. The following hyper-parameters were used: adaptive gradient solver, mini-batch size of 64, learning rate of 0.01, weight decay of 0.0005, and epoch size of 100.

Three aspects of performance were studied. The first one was the accuracy obtained from the average of 10-fold-cross-validation. Top-1 and Top-5 accuracies were reported. The second one was the processing time required by each model. Batch size was set to be one in the data layer in order to allow CNN to process one iteration per image. 500 images were randomly selected and the average forward-backward time required per image was determined. The last one was the portability which contains two parameters; that

is, the total number of parameters and the storage required to store the trained model.

Table 3
Performance of NU-InNet 1.0, NU-InNet 1.1, and the winning models from ILSVRC

Model	Average Accuracy Top1 (%)	Top5 (%)	Average Forward-Backward (ms/image)	Parameters ($\times 10^6$)	Model size (MB)
AlexNet [9]	58.1	86.4	13.90	58.48	217
SqueezeNet [13]	58.2	87.4	24.53	0.75	2.86
GoogLNet [10]	68.4	91.7	40.13	10.45	39.9
NU-InNet 1.0	69.8	92.3	18.16	0.88	3.37
NU-InNet 1.1	68.7	92.3	36.52	0.89	3.42

The performance of the proposed models and the winning models from ILSVRC is shown in Table 3. Considering Top-1 and Top-5 accuracies, it is seen that the proposed NU-InNet 1.0 and 1.1 are better than the winning models from ILSVRC. For example, for Top-1 accuracy, the best accuracy is from NU-InNet 1.0, that is 69.8% accuracy. However, considering the average forward-backward time, it was found that the smallest one is from AlexNet (that is, 13.90 ms/image) since the architecture of AlexNet is less complicated. For the number of parameters and model size, it is seen that SqueezeNet requires the smallest number of these two factors, that are 0.75×106 and 2.86 MB, respectively.

Comparing NU-InNets with AlexNet in terms of the average forward-backward time, NU-InNet 1.0 requires slightly longer time to process, that is, 4.26 ms/image longer. While NU-InNet 1.1 requires the average time of approximately 2 times larger than that from Nu-InNet 1.0 since the number of layers in NU-InNet 1.1 is larger than the number of layers in NU-InNet 1.0. Additionally, considering NU-InNets with SqueezeNet in terms of the required number of parameters and the model size, it is seen that both NU-InNets require small numbers of parameters and model size similar to SqueezeNet.

From the previous discussion, it has been shown that the proposed NU-InNet 1.0 and 1.1 can deliver the accuracy with the same range as that from GoogLeNet. Moreover, as shown in Figyure 4, the obtained average of the forward-backward time, the number of parameters, and the model size of the proposed models are smaller, especially for the number of parameters and the model size. These impressive performances allow the proposed models to be appropriately utilized in a smartphone at which these factors are very limited.

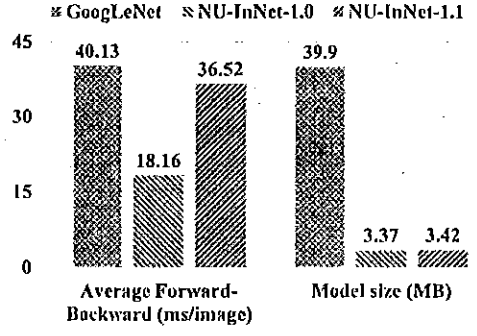


Figure 4: Comparisons between GoogLeNet, NU-InNet 1.0, and NU-InNet 1.1

B. Thai Food Image Recognition Application for Android

After the training process, the proposed models and GoogLeNet were applied to use with an Android smartphone in order to test for the required execution time. The tested image size is 1080×1080 pixels. The execution time for each model is shown in Table 4.

Table 4
Execution time required by GoogLeNet, NU-InNet 1.0, and NU-InNet 1.1

Model	Execution Time (ms)
GoogLeNet	906
NU-InNet 1.0	351
NU-InNet 1.1	691

From Table 4, it is seen that the use of NU-InNets significantly reduces the execution time in comparison to that from GoogLeNet. A reduction of 555 and 215 ms/image can be obtained from NU-InNet 1.0 and NU-InNet 1.1, respectively. The faster execution time in NU-InNets can then result in a fast responding time in the Thai Food Image Recognition application. Certainly, the use of these proposed models in the application improves the satisfaction of smartphone's users.

V. Conclusion

In this paper, NU-InNet 1.0 and 1.1 have been proposed. The inception concept in GoogLeNet was adopted and modified in the proposed models. A new Convolutional Neural Network (CNN) was designed. The main objective of the proposed models was to reduce the processing time and the model size, while keeping the accuracy not to be poorer than that of GoogLeNet so that the proposed models can be used in a smartphone. The proposed models have been tested with the dataset THFOOD-50 which contains images of 50 famous Thai menu. It is found that the accuracies obtained from NU-InNet 1.0 and 1.1 are slightly better than that of GoogLeNet. The required processing time is reduced by a factor of 2 for the case of NU-InNet 1.0 comparing to GoogLeNet. The model size from both proposed models is less than one-tenth of the model size required by GoogLeNet. It is clearly shown that the significant reductions in terms of processing time and model size from the proposed NU-InNet 1.0 and 1.1 can lead to a more suitable Thai Food Image Recognition application in a smartphone.

Acknowledgments

This work was supported by Naresuan University, Thailand.

References

[1] T. Maruyama, Y. Kawano, and K. Yanai, "Real-time mobile recipe recommendation system using food ingredient recognition," in *Proceedings of the 2nd ACM international workshop on Interactive multimedia on mobile and portable devices*, 2012, pp. 27-34.

[2] N. Tammachat and N. Pantuwong, "Calories analysis of food intake using image recognition," in *Information Technology and Electrical Engineering (ICITEE), 2014 6th International Conference on*, 2014, pp. 1-4.

[3] M. M. Anthimopoulos, L. Gianola, L. Scarnato, P. Diem, and S. G. Mougiakakou, "A food recognition system for diabetic patients based on an optimized bag-of-features model," *Biomedical and Health Informatics, IEEE Journal of*, vol. 18, pp. 1261-1271, 2014.

[4] Y. Kawano and K. Yanai, "Foodcam: A real-time food recognition system on a smartphone," *Multimedia Tools and Applications*, pp. 1-25, 2015.

[5] V. Bettadapura, E. Thomaz, A. Parnami, G. D. Abowd, and I. Essa, "Leveraging context to support automated food recognition in restaurants," in *2015 IEEE Winter Conference on Applications of Computer Vision*, 2015, pp. 580-587.

[6] Y. Matsuda, H. Hoashi, and K. Yanai, "Recognition of multiple-food images by detecting candidate regions," in *Multimedia and Expo (ICME), 2012 IEEE International Conference on*, 2012, pp. 25-30.

[7] D. Mery and F. Pedreschi, "Segmentation of colour food images using a robust algorithm," *Journal of Food engineering*, vol. 66, pp. 353-360, 2005.

[8] Y.-W. Chang and Y.-Y. Chen, "An improve scheme of segmenting colour food image by robust algorithm," *Proc. Algo2006*, 2006.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097-1105.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, et al., "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1-9.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[12] K. Yanai and Y. Kawano, "Food image recognition using deep convolutional network with pre-training and fine-tuning," in *Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on*, 2015, pp. 1-6.

[13] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 1MB model size," *arXiv preprint arXiv:1602.07360*, 2016.

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, et al., "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 675-678.

[15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *arXiv preprint arXiv:1512.00567*, 2015.

[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

Accuracy Improvement of Thai Food Image Recognition Using Deep Convolutional Neural Networks

Chakkrit Termritthikun[1] *and Surachet Kanprachar*[2]
Department of Electrical and Computer Engineering
Faculty of Engineering, Naresuan University
Phitsanulok, Thailand
[1]*chakkritte57@nu.ac.th and* [2]*surachetka@nu.ac.th*

Abstract— **To improve the performance of the convolutional neural networks, it is normally done by increase the deepness or put more layers to the network. By doing such, the number of parameters is increased. In this paper, NU-InNet, which was developed from GoogLeNet, is modified by adding more layers to the network in order to improve the accuracy of the network while keeping the number of the parameters to be suitable for being used in a smartphone. Testing the proposed model with a database containing 50 well-known kinds of Thai food, it is found that the processing time and size of the parameters of NU-InNet with a depth of 4 are less than those of BN-Inception network by 1.5 and 11 times, respectively. Importantly, the accuracy of NU-InNet with a depth of 4 is higher than that of BN-Inception network by 8.07%.**

Keywords*— *deep learning; food regcognition; convolutional neural networks; smartphone; Thai food; BN-Inception

I. Introduction

To allow a computer to learn from a huge amount of data like a human, an algorithm called deep learning has been developed. Segmenting the considered data and assigning a number to each segment, deep learning algorithm can help studying the data; e.g., photos, voices, and videos. This algorithm is widely used in photo analysis [1, 2] since the characteristics in each class of photos are different but can lead to the same meaning. Normally, the analysis is done by using image detection and image recognition. There has been a contest in this field every year called the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [3] in order to find the best algorithm in image recognition with a provided dataset of images. One crucial algorithm called Convolutional Neural Network (CNN), which is a kind of deep learning algorithm, has been adopted in this contest. This algorithm has been shown to be a good technique in improving the image recognition performance. A new model called GoogLeNet [4] proposed by Google has won the ILSVRC in the year 2014. CNN was adopted by GoogLeNet with a concept of inception module. The data from the preceding layer is calculated and used in 1×1, 3×3, 5×5 convolutional layers and 3×3 pooling layer. Then, the determined data is concatenated increasing the depth of the data. By doing this, the number of parameters in 3×3 and 5×5 convolutional layers is growing significantly. To reduce such number, a 1×1 convolutional layer can be placed prior to the 3×3 or 5×5 convolutional layers. Additionally, putting a 1×1 convolutional layer after the 3×3 pooling layer, the number of filters to be used in each layer can be lessened.

Inception module has been continuously developed; for example, in Inception-v2 [5], batch normalization has been added to every layer in Inception-v1 in order to improve the correctness of the training process. In Inception-v3 [6], each 5×5 convolutional layer has been replaced by two 3×3 convolutional layers so that the required data processing is reduced.

The inception module adopted in GoogLeNet has also been utilized in NU-InNet [7] by redesigning the structure of CNN in order to minimize the processing time and the model size for being suited for a smartphone application. NU-InNet has been tested with a Thai-food dataset (THFOOD-50) including images of 50 famous kinds of Thai food. It has been shown that the processing time and the model size were reduced by 2 and 10 times, respectively, comparing to GoogLeNet, while keeping the image recognition accuracy in the same level.

In this work, NU-InNet, which has a model size between 3.37×10^6 – 3.42×10^6, is adopted and further improved in terms of its image recognition accuracy by adding batch normalization [5, 8] at the end of each convolutional layer to increase the training efficiency. According to the wide residual networks [9], the width of the proposed network is also increased so that the features can be analyzed more in detail. Additionally, Xavier is replaced by MSRA (Microsoft Research Asia) initialization [10] to make the convergence of the training improved.

The organization of this paper is done as follow. The related technology will be given in Section II. The design of the proposed network will be provided and explained in Section III. And, in Section IV, the testing results will be shown and discussed. Finally, the research work is summarized in Section V.

II. Related Technology

A. NU-InNet [7]

NU-InNet has been developed starting from the inception module from GoogLeNet as shown in Fig.1 (left) by reducing

978-1-5090-4666-9/17/$31.00 ©2017 IEEE

the parameter size and the processing time in image recognition. There are 2 versions of NU-InNet module; that is, NU-Inception 1.0 and NU-Inception 1.1. These two are explained below.

- NU-Inception 1.0 is the module that the 3×3 Max pooling layer and 1×1 convolutional layer are changed to be 1×1 and 7×7 convolutional layers, respectively, as seen from Fig.1 (middle).
- NU-Inception 1.1 is the module that adopts NU-Inception 1.0 and replaces any 5×5 convolutional layer by two 3×3 convolutional layers and any 7×7 convolutional layer by three 3×3 convolutional layers, as shown in Fig.1 (right).

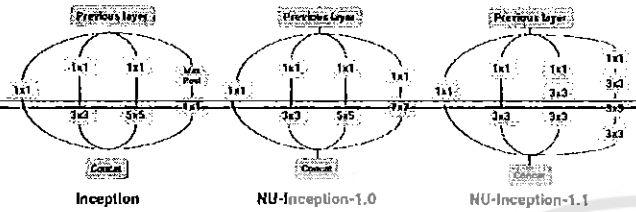


Fig.1. NU-InNet architectures [7].

As explained previously, NU-InNet has been developed to suit the smartphone application; thus, only one module was used so that the model size and processing time can be reduced significantly.

B. Batch Normalization [5]

Batch normalization or BN is the technique developed by Google to accelerate the convergence of neural network training process. It is done by changing the mean and variance of the nonlinear input (for example, Sigmoid and ReLU responses) to be 0 and 1, respectively. By doing this, the problem according to the internal covariate shift between the training and testing data sets will be lessened. In a neural network design, BN layer will be put between a convolutional layer and an activation layer in order to normalize a smaller size of data called mini-batch so that it can help increasing the accuracy and the speed of training process.

III. Proposed Network

First, as discussed, the proposed network will be tested with THFOOD-50 dataset. Example images of Thai food are shown in Fig.2.

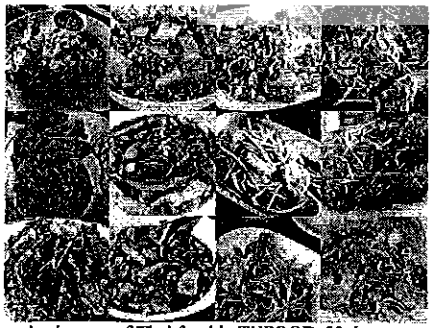


Fig.2. Examples images of Thai food in THFOOD-50 dataset.

From Fig.2, 12 images; that is, 4 kinds of Thai food, each with 3 different images, are represented. These images are example images of THFOOD-50 dataset, which contains 15,770 images of 50 famous kinds of Thai food. In each image, only one kind of food is presented. This dataset will be used in the training and testing processes later.

For the proposed network, the accuracy of the image recognition will be improved comparing to those of NU-InNet 1.0 and 1.1 while keeping the model size and processing time to be in the level that suit to be used in a smartphone. This is done by increasing the number of NU-Inception modules in each level of the image scale [e.g., 56×56, 28×28, 14×14, 7×7] so that the image can be analyzed deeply. It is assigned the depth of the model to be 4, 8, and 12 (that is, N = 1, 2, 3, respectively). For example, with a depth of 8 (that is, N = 2), the network will consist of 4 NU-Inception 1.0/1.1 blocks and in each block there will be 2 NU-Inception 1.0/1.1 modules cascaded. Between each block, a max pooling layer is placed with a patch size of 3×3 and stride of 2 so that the data size to be transferred to the next block will be reduced by one half. The detail of the proposed network is shown in Table I and Fig.3.

TABLE I. Detail of Layers in NU-InNet 1.0/1.1

Type	Patch Size/Stride	Output Size
Input Image	-	224×224×3
Convolutional	5×5/2	113×113×3
Max Pool	3×3/2	56×56×64
NU-Inception × N	As in Fig. 1	56×56×64
Max Pool	3×3/2	28×28×64
NU-Inception × N	As in Fig. 1	28×28×128
Max Pool	3×3/2	14×14×128
NU-Inception × N	As in Fig. 1	14×14×256
Max Pool	3×3/2	7×7×256
NU-Inception × N	As in Fig. 1	7×7×512
Average Pool	-	1×1×512
Fully Connected	-	1×1×50

From Fig.3 (right), it is clearly seen that the depth of the proposed network is longer than that of NU-InNet since more NU-Inception modules are added. The output from each block is shown in Table I. It is seen that starting from the input image of 224×224×3, the output from the first block is 56×56×64. The process is done repeatedly for 4 times and at the end of the network, the output size is 1×1×50.

Additionally, in the proposed network, MSRA initialization is adopted instead of Xavier initialization. And, batch normalization is placed after every convolutional layer in order to improve the convergence of the training process.

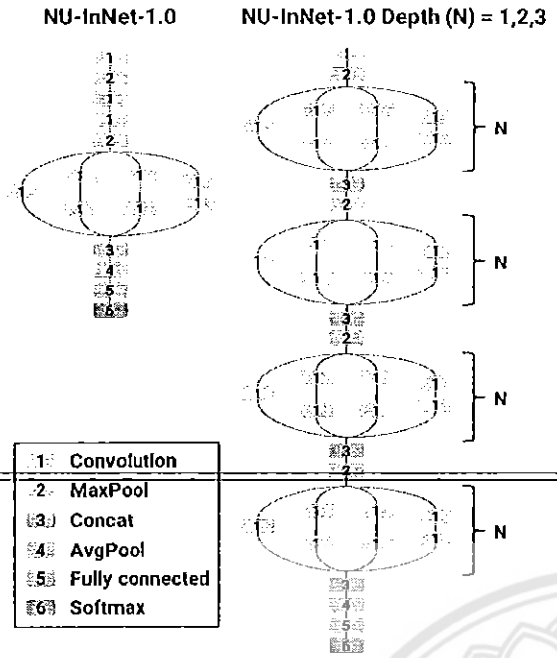


Fig.3. Proposed network architecture.

IV. Results and Discussion

For testing the proposed network, the training process is done by using a high performance computer consisting of Inter(R) Xeon(R) E5-2683 v3 @2.00GHz 56-core CPU, 64 GB RAM, and NVIDIA Tesla K80 GPU. The operating system is Ubuntu Server 14.04.5 with an installed Caffe [11], which is the framework for developing CNN with C++. And, the training process is done from scratch.

The data to be used in the training and testing processes is THFOOD-50 which is the dataset containing 15,770 images of 50 famous kinds of Thai food. The data is divided into 10 sets to be used in 10 fold cross-validation. It is assigned that in each experiment, 90% of the data is used for training process and the rest (that is, 10%) is used for testing process. All images are resized to be 256×256 pixels.

In the training process, to compare between different kinds of network, GoogLeNet, BN-Inception, and NU-InNet 1.0/1.1 with the depth of 1, 4, 8, and 12 are used. With 10 fold cross-validation, the following hyper-parameters are assigned: Stochastic Gradient Descent (SGD) solver [12], mini-batch size of 64, learning rate starting at 0.1 and reduced by 1/10 every 25 epochs, weight decay of 0.0005, and epoch size of 100.

In the experiments, 3 major factors are focused; that is, the parameter size, the processing time in terms of the average forward-backward time per image, and the accuracy both top-1 and top-5 accuracies. It is seen that these performance factors are quite important, especially, as mentioned previously, for the case of being used in a mobile or smartphone. The parameter size and the processing time should be as small as possible without affecting the recognition accuracy. The results from the experiments are shown in Table II.

TABLE II. Results from testing various models with THFOOD-50 Dataset

Model	Depth	Parameter Size ($\times 10^6$)	Avg forward-backward time (ms/image)	Avg-Top-1 Acc. (%)	Avg Top-5 Acc. (%)
GoogLeNet [4]	9	10.45	40.13	68.40	91.71
BN-Inception [5]	9	10.46	63.24	72.27	93.29
NU-InNet 1.0 [7]	1	0.43	12.54	70.14	93.22
	4	0.95	37.66	79.68	96.00
	8	2.01	66.08	78.62	95.68
	12	3.06	95.48	75.44	94.36
NU-InNet 1.1 [7]	1	0.44	20.03	74.24	94.53
	4	0.94	40.80	80.34	96.27
	8	2	71.72	79.75	95.93
	12	3.05	100.50	75.69	94.34

From Table II, considering NU-InNet 1.0 and 1.1, it is seen that, in terms of the accuracy, adding more modules to the network can increase the recognition accuracy. For NU-InNet 1.0, the top-1 accuracy is increased from 70.14% to 79.68%, 78.62%, and 75.44% for the depth of 4, 8, and 12, respectively. The highest top-1 accuracy is from NU-InNet 1.0 with the depth of 4. An increase of 9.54% in terms of top-1 accuracy is obtained. Similarly, for NU-InNet 1.1, the top-1 accuracy is increased from 74.24% to 80.34%, 79.75%, and 75.69% for the depth of 4, 8, and 12, respectively. The highest top-1 accuracy is from NU-InNet 1.1 with the depth of 4. An increase of 6.10% in terms of top-1 accuracy is obtained. Comparing all NU-InNet results, it is found that NU-InNet 1.1 with the depth of 4 provides the best top-1 accuracy. Further considering about the number of parameters and processing time, it is seen that increasing the depth of the network results in an increase of both number of parameters and the processing time. However, as seen from the table, the numbers of parameters shown for NU-InNet 1.0/1.1 with all depths are lower than 3.06×10^6 which is still small and suited for a smartphone application. For NU-InNet 1.0/1.1 with the depth of 4, which convey the best top-1 accuracy, it is seen that the parameter size is still less than 10^6. Furthermore, the processing time (i.e., the average forward-backward time) for NU-InNet 1.0/1.1 with the depth of 4 is less than 41 ms/image, which is still in the acceptable range for a smartphone application.

Since, as discussed, the NU-InNet [7] is based on the inception idea from GoogLeNet and in this work the concept of batch normalization is adopted, to suitably compare the proposed network, BN-Inception network should be also tested. BN-Inception network has been developed from GoogLeNet with an addition of batch normalization to the network. All three major factors for BN-Inception, NU-InNet 1.0 (depth = 4), and NU-InNet 1.1 (depth = 4) from Table II are shown graphically in Fig.4.

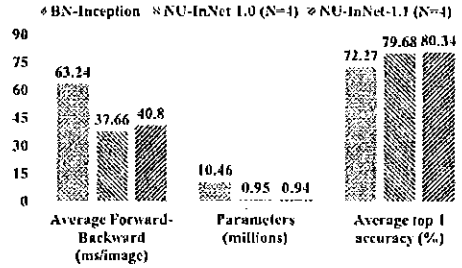


Fig.4. Comparisons between BN-Inception, NU-InNet 1.0 (depth = 4), and NU-InNet 1.1 (depth = 4).

From Fig.4, it is seen that for the average forward-backward time, BN-Inception requires the longest time; that is, 63.24 ms/image while NU-InNet 1.0 and 1.1 require 37.66 and 40.80 ms/image, respectively. This then means that both NU-InNet networks use less processing time than BN-Inception network. Considering the number of parameters, it is seen that both NU-InNet networks require less than 1×10^6 of parameter size while for BN-Inception network, the parameter size is about 10.46×10^6 or more than 10 times larger. Considering the top-1 accuracy, it is seen that the lowest accuracy is from BN-Inception network; that is, 72.27% accuracy is obtained. The top-1 accuracies from NU-InNet 1.0 and 1.1 are 79.68% and 80.34%, respectively. These two top-1 accuracies are higher than that of BN-Inception network for 7.37% and 8.07%, respectively. It is clearly seen that the better performance in terms of the processing time, the parameter size, and the top-1 accuracy can be obtained from the proposed networks with the depth of 4.

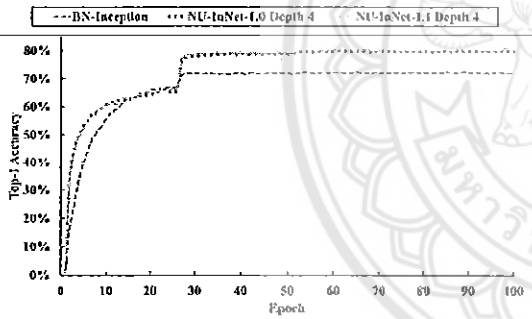


Fig.5. Top–1 accuracy vs. number of epochs: for BN-Inception, NU-InNet 1.0 (depth = 4), and NU-InNet 1.1 (depth = 4).

The convergence of the top-1 accuracy is shown in Fig.5. In this figure, three types of network are compared; that is, BN-Inception, NU-InNet 1.0 (depth = 4), and NU-InNet 1.1 (depth = 4). It is seen that initially both NU-InNet 1.0 and 1.1 networks converge to a high level of top-1 accuracy faster than BN-Inception. For example, to reach the top-1 accuracy of 50%, the number of epochs required by both NU-InNet 1.0, and NU-InNet 1.1 is 4, which is one half of that required by BN-Inception (that is, 8 epochs). However, as the number of epochs grows further to 25, the top-1 accuracies of three networks reach the same level of 66%. And, as the number of epochs increases to 28, a sudden increase of top-1 accuracy is obtained and reaches its highest value; that is, 72.27%, 79.68%, and 80.34, for BN-Inception, NU-InNet 1.0, and NU-InNet 1.1, respectively.

V. Conclusions

In this paper, the NU-InNet network is modified in order to increase the image recognition accuracy while keeping the parameter size and processing time to suit a smartphone application. More layers are added to the network so that the data analysis can be done more in detail. The proposed network is trained and tested with a dataset containing images for 50 famous kinds of Thai food. It is shown that considering three key performance factors, NU-InNet 1.0 and 1.1 with a depth of 4 are better than that of BN-Inception network. The top-1 accuracy from NU-InNet 1.1 with a depth of 4 is 80.34% which is 8.07% superior to that of BN-Inception network. And, the processing time and the parameter size from NU-InNet 1.0 and 1.1 with a depth of 4 are approximately 1.5 and 11 times lower than those of BN-Inception network. These show that better recognition accuracy with a small processing time and a small parameter size can be obtained from the proposed NU-InNet 1.0/1.1 network with a depth of 4.

Acknowledgment

This work was supported by Naresuan University, Thailand.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097-1105.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, 2015, pp. 436-444.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, et al., "Imagenet large scale visual recognition challenge," International Journal of Computer Vision, vol. 115, 2015, pp. 211-252.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, et al., "Going deeper with convolutions," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1-9.

[5] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in Proceedings of The 32nd International Conference on Machine Learning, 2015, pp. 448-456.

[6] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," arXiv preprint arXiv:1512.00567, 2015.

[7] C. Termritthikun, P. Muneesawang, and S. Kanprachar, "NU-InNet: Thai Food Image Recognition Using Convolutional Neural Networks on Smartphone," in 2016 Computer Sciences and Information Technology International Conference (COMSIT'2016), Krabi, Thailand, 2016.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," arXiv preprint arXiv:1512.03385, 2015.

[9] S. Zagoruyko and N. Komodakis, "Wide Residual Networks," arXiv preprint arXiv:1605.07146, 2016.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1026-1034.

[11] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, et al., "Caffe: Convolutional architecture for fast feature embedding," in Proceedings of the 22nd ACM international conference on Multimedia, 2014, pp. 675-678.

[12] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in Proceedings of COMPSTAT'2010, ed: Springer, 2010, pp. 177-186.