การพัฒนาระบบตรวจสอบดวงตาด้วยโครงข่ายประสาทเทียมแบบคอลโวลูชัน

Glaucoma Classification by using Convolutional Neural Network

นางสาวรักษมินฆ์	ธนสิทธิ์	60102010343
นางสาวอรัญญาภรณ์	ลาภหลาย	60102010351
นางสาวพิชชาพร	สุริยันชัยเจริญ	60102010573

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

มหาวิทยาลัยศรีนครินทรวิโรฒ ปีการศึกษา พ.ศ. 2563

คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ

ชื่อหัวข้อโครงงาน **การพัฒนาระบบตรวจสอบดวงตาด้วยโครงข่ายประสาทเทียมแบบคอลโวลูชัน**

Glaucoma Classification by using Convolutional Neural Network

นิสิต	นางสาวรักษมินฑ์	ธนสิทธิ์	60102010343
	นางสาวอรัญญาภรณ์	ลาภหลาย	60102010351
	นางสาวพิชชาพร	สุริยันชัยเจริญ	60102010573

ปริญญา วิทยาศาสตร์บัณฑิต (วท.บ.)

ภาควิชา วิทยาการคอมพิวเตอร์

อาจารย์ที่ปรึกษาโครงงาน อ.ดร.ศิริสรรพ เหล่าหะเกียรติ

ลงชื่อ.....

(อ.ดร.ศิริสรรพ เหล่าหะเกียรติ)

อาจารย์ที่ปรึกษาโครงงาน

บทคัดย่อ

เทคโนโลยีการมองเห็นด้วยคอมพิวเตอร์(Computer Vision)ใช้เพื่อตรวจจับพื้นที่ที่น่าสนใจ หรือวัตถุที่น่าสนใจ อย่างไรก็ตามกระบวนการปฏิบัติงานต้องใช้เวลาในการประมวลผลมากในการทำนายวัตถุที่สนใจ นอกจากนี้กระบวนการบางอย่างมีความซับซ้อนมากในการคำนวณและดำเนินการ ยิ่งไปกว่านั้นภาพการวินิจฉัยโรคของต้อหินยังต้องใช้แพทย์เฉพาะทางในการวิเคราะห์ ดังนั้นงานวิจัยนี้จะเสนอสถาปัตยกรรมโครงข่ายประสาทเทียม(Convolutional Neural Network)เพื่อทำนายภาพที่เป็นปกติหรือโรคต้อหินจากการสแกนจอประสาทตา จากผลการทดลองสถาปัตยกรรมที่นำเสนอแสดงถึงความแม่นยำในการทำนายภาพต้อหินประมาณ 80 เปอร์เซ็นต์ ดังนั้นสถาปัตยกรรมที่นำเสนอจึงเป็นผู้ช่วยที่มีประโยชน์ในการวินิจฉัยภาพจอประสาทตา

Abstract

Computer vision technology is used to detect an interesting area or interesting objects. However, an operational process requires a lot of processing time to predict the interested object. In addition, some processes are very complex to compute and implement. Moreover, the diagnostic images of the glaucoma stage have to use special medical doctors to analyze. So, this research will propose a convolutional neural network architecture to predict the normal or glaucoma images from optical coherence tomography retinal scan. From the experimental results, the proposed architecture represents the accuracy to predict the glaucoma images about 80 percentage. Therefore, the proposed architecture is a useful assistant to diagnose the retinal images.

กิตติกรรมประกาศ

ขอขอบคุณภาควิชาวิทยาการคอมพิวเตอร์ สำหรับการเอื้อเฟื้ออุปกรณ์ที่จำเป็นต่อการดำเนินงานต่างๆ และขอขอบพระคุณอาจารย์ประจำภาควิชาวิทยาการคอมพิวเตอร์ทุกท่านที่ได้คอยให้ความรู้ คำแนะนำและคำปรึกษา ซึ่งเป็นส่วนสำคัญที่ทำให้การดำเนินงานเป็นไปได้อย่างราบรื่น

ขอขอบคุณมหาวิทยาลัยศรีนครินทรวิโรฒ ที่อำนวยความสะดวกเรื่องสถานที่และสิ่งที่จำเป็นต่อการดำเนินโครงงาน และเป็นแหล่งให้ความรู้ในการนำไปพัฒนาการทำงานต่อในอนาคต

ขอขอบพระคุณอาจารย์ศิริสรรพ เหล่าหะเกียรติ และ อาจารย์วีระ สอิ้ง ที่คอยให้คำปรึกษาและช่วยเหลือตลอดการดำเนินโครงงานในทุกๆด้าน จนสามารถดำเนินโครงงานสำเร็จลุล่วงไปได้ด้วยดี

สุดท้ายนี้ ขอขอบพระคุณทุกท่านที่มีส่วนเกี่ยวข้องในการดำเนินโครงงานในครั้งนี้ ที่คอยให้ความช่วยเหลือ จนโครงงานสามารถสำเร็จลุล่วงไปได้ด้วยดี คณะผู้จัดทำหวังว่าโคงงานการพัฒนาระบบตรวจสอบดวงตาด้วยโครงข่ายประสาทเทียมแบบคอลโวลูชัน(Glaucoma Classification by using Convolutional Neural Network) จะเป็นประโยชน์ในการใช้งานกับระบบที่เหมาะกับการใช้ตรวจสอบดวงตาและสามารถนำไปพัฒนาประสิทธิภาพในด้านต่างๆต่อไปในอนาคต

นางสาวรักษมินฑ์ ธนสิทธิ์

นางสาวอรัญญาภรณ์ ลาภหลาย

นางสาวพิชชาพร สุริยันชัยเจริญ

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

เนื่องจากในปัจจุบันมีการใช้เทคโนโลยีการมองเห็นของคอมพิวเตอร์ (Computer Vision) เพื่อใช้ในการตรวจจับพื้นที่ สิ่งของ หรือวัตถุต่างๆที่สนใจกันอย่างแพร่หลาย แต่กระบวนการบางวิธีจำเป็นต้องใช้เวลามากทั้งในกระบวนการศึกษาและการปฏิบัติ อีกทั้งบางกระบวนการยังมีความซับซ้อนของการคำนวณเป็นอย่างมากจึงทำให้งานวิจัยชิ้นนี้ค้นคว้าและทำการเลือกใช้การทำ CNN (Convolutional Neural Networks) ซึ่งเป็นเครื่องมือที่เป็นการพัฒนาชุดคำสั่งที่สามารถเรียกใช้ระบบได้อย่างง่าย และมีกระบวนการทำงานที่รวดเร็ว เพื่อวิเคราะห์พื้นที่และรูปแบบของวัตถุที่อยู่ในรูปภาพ ซึ่งงานวิจัยนี้จะทำการพัฒนาด้วยหลักการเขียนโปรแกรมที่มีชื่อว่า Python ที่มีความสะดวกต่อการพัฒนาและมีประสิทธิภาพสูงในการคำนวณ โดยจะทำหลักการที่นำเสนอนี้มาพัฒนาเพื่อใช้ในการตรวจจับพื้นที่ที่สนใจ (Interesting Area) ในข้อมูลภาพดวงตาที่ได้จากเครื่องถ่ายภาพของการตรวจจอประสาทตา Optical coherence tomography (OCT) ทางการแพทย์ เพื่อจะนำข้อมูลภาพที่ได้มาใช้แยกพื้นที่ภายในดวงตาตามข้อมูลที่ได้จัดเตรียมมาก่อนหน้า

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษาหลักการของการสร้างฐานข้อมูลเพื่อใช้กับหลักการ CNN ได้

1.2.2 เพื่อศึกษาหลักการทำงานของ CNN ในการตรวจจับวัตถุ

1.2.3 ศึกษาหลักการตรวจสอบความถูกต้องของ CNN ได้

1.2.4 เพื่อพัฒนาหลักการ CNN ที่สามารถแยกภาพดวงตาที่เป็นโรคต้อหินได้

1.3 ขอบเขตของโครงงาน

1.3.1 สร้างรูปแบบของการทำนายจากหลักการ CNN

1.3.2 พัฒนางานวิจัยด้วยโปรแกรม Google Colaboratory ในการพัฒนาระบบ CNN

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 เพื่อเป็นแนวทางของการพัฒนาระบบตรวจสอบดวงตา

1.4.2 เพื่อช่วยให้แพทย์สามารถแยกภาพดวงตาที่เป็นโรคต้อหินได้

บทที่ 2

องค์ความรู้ที่เกี่ยวข้อง

2.1 ทฤษฎีและแนวคิดที่ใช้ในการทำงานวิจัย

2.2.1) การพัฒนาหลักการ Google Colaboratory

รูปภาพ 1 Google Colaboratory

ที่มา : [ออนไลน์] https://leaherb.com/save-google-colab-notebook-to-html/

ใช้ในการเขียนเทรนดาต้าด้วยภาษา Python โดย Google Colab มีไลบราลี่ที่ช่วยในการเขียนโค้ด และการแสดงผลลัพธ์ได้ อีกทั้งยังสามารถเขียน Text Editer กำกับเพื่ออธิบายแต่ละโค้ดที่ถูกเขียนขึ้นมา

2.2.2) การจำแนกประเภทข้อมูลแบบรูปภาพด้วย CNN (Convolutional Neural Network)

โครงข่ายประสาทเทียบแบบคอนโวลูชัน (Convolutional Neural Network, CNN)

Convolutional Neural Network เป็น Deep Learning รูปแบบหนึ่ง ที่นำหลักการทางคณิตศาสตร์ Convolution Operation มาทำจัดการกับข้อมูลด้วยการใช้ Convolution Filtering ซึ่งเป็นการกำหนด Window ขึ้นมา และทำการคำนวณ Pixel รอบ ๆ Window นั้น เพื่อลดขนาดของข้อมูลลง โดยจะกำหนด Image Size ที่ป้อน Input เข้ามาเป็นขนาด n×n และกำหนด Window หรือ Filters ให้มีขนาด f×f ผลลัพธ์จะได้ Output ที่เป็น Feature Maps ขนาด $n - f + 1 \times n - f + 1$ ในขั้นตอนของ

การทำ Convolution จะใช้ Activation Function มาช่วยในการทำให้ผลลัพธ์ของข้อมูลอยู่ในรูปแบบที่เราต้องการ

หลังจากนั้นจะส่งภาพไปทำการ Pooling ที่เป็นการลดขนาดข้อมูลลงเช่นกัน แต่ช่วยคงคุณลักษณะสำคัญในภาพไว้ได้ และทำให้รูปภาพนั้นซับซ้อนน้อยลง Window หรือ Filter ขนาดที่นิยมใช้คือ 2×2 หรือ 3×3โดย Pooling จะมี 4 ประเภท คือ

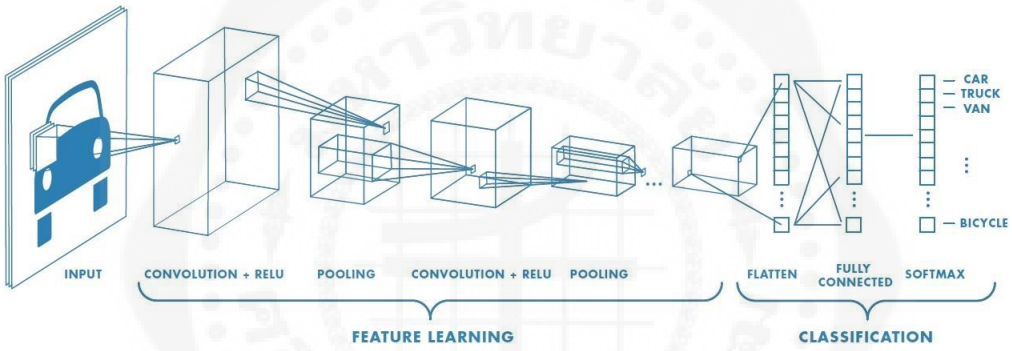
- Max pooling ที่จะเลือกค่าสูงสุด (Maximum Value) ภายใน Filters
- Average pooling ที่จะคำนวนหาค่าเฉลี่ยภายใน Filter
- Global Max Pooling ใช้ในการลดขนาดของ feature maps output แทนการ Flattening หรือ Dense layer ภายใน Filters
- Global Average Pooling ใช้แทน fully-connected หรือ densely-connected layers ช่วยลดการเกิด overfitting เนื่องจากไม่มีค่าให้ฟังก์ชันเรียนรู้มาก่อนหน้า

โดยขั้นตอนในการ Convolution, Activation และ Pooling สามารถทำวนซ้ำกันไปจนกว่าจะได้ข้อมูลที่ต้องการ ก่อนที่จะส่งข้อมูลไปทำการ Flatten คือการทำข้อมูลที่มีอยู่ในรูปแบบ Metrix ที่มีหลาย Dimension ให้กลายเป็นรูปแบบ Vector ซึ่งมี Dimension อยู่ 1 Column เท่านั้น เพื่อนำไปใช้ในการ Fully Connected ต่อไป

เมื่อทำการจัดการกับข้อมูลเสร็จแล้ว จะนำ Output ที่ได้มาใช้ร่วมกับหลักการ Neural Network Concept ซึ่งเป็นการสร้างโครงข่ายประสาทเทียม จากการรับ Input Node เข้ามา แล้วนำ Node แต่ละ Node ในหนึ่งชั้นเชื่อมต่อกับทุก Node ในชั้นถัดไป จนถึง Output Node โดยเรียกวิธีการนี้ว่า Fully Connected และใช้ Weight ในการบอกว่าแต่ละ Node เชื่อมต่อกันอย่างไร โดยในช่วง Learning Phase จะเรียนรู้การปรับ Weight จนได้ Weight ที่เหมาะสมกับการ Predict Input ที่ถูกป้อนเข้ามาใหม่

ซึ่งในส่วนของการทำ Convolutional, Activation และ Pooling เป็นขั้นตอนของการเตรียมข้อมูลเพื่อดึงคุณลักษณะที่จะใช้ในการจดจำวัตถุนั้นว่าเป็นของสิ่งใด ก่อนนำไปทำการ Classification ในส่วนของ Flatten และ Fully Connected เมื่อทำการสร้าง Model จาก Convolutional Neural Network ด้วยการใช้ Convolutional, Activation, Pooling, Flatten และ Fully Connected ตามลำดับแล้ว จะได้ Model ที่สามารถบันทึกเพื่อนำไปใช้ในการ Predict ข้อมูลอื่นๆ ต่อไปได้

- CNN จะทำการเทรนด้วยการนำทั้งภาพ หรือส่วนที่ทำการตัดออกมาไปเทรน แล้วทดสอบด้วยภาพที่มีขนาดเท่ากับข้อมูลที่นำไปเทรน



รูปภาพ 2 แสดงตัวอย่างการทำ CNN

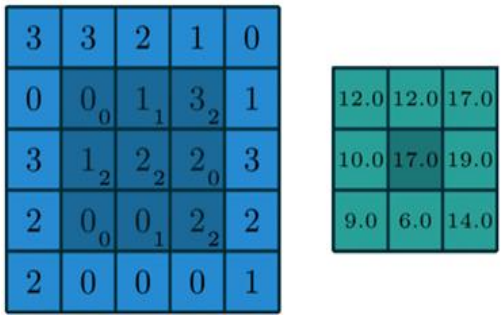
ที่มา : [ออนไลน์] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

เนื่องจาก Dataset ที่ได้รับมามีขนาดเล็ก เมื่อทำการตัดรูปภาพออกมาจะได้ขนาดรูปที่เล็กเกินไป เมื่อนำมาเทรนทำให้มีโอกาสที่จะเกิดการเทรนผิดพลาดสูง จึงต้องจำเป็นใช้รูปที่มีขนาดใหญ่ขึ้น และนำหลักการ CNN มาใช้

ขั้นตอนการทำ CNN ประกอบด้วย 5 Layer ดังนี้

- **Layer 1 : Convolution Layer** ทำหน้าที่ลดขนาดข้อมูลด้วยการแยกส่วนประกอบภายในภาพ แบ่งออกเป็น 9 ประเภท

1.) Convolution: the single channel version (1D)

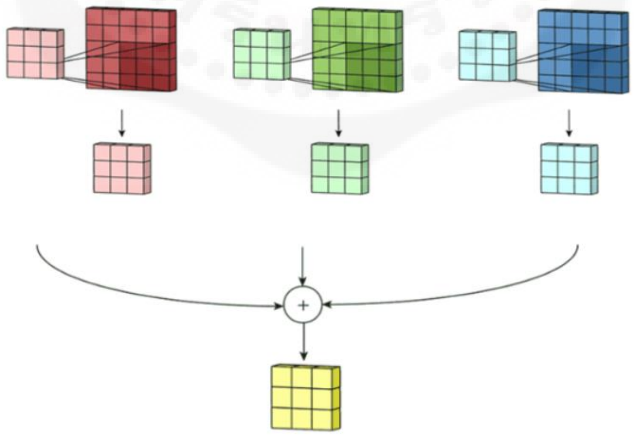


รูปภาพ 3 แสดงตัวอย่างการทำ Convolution: the single channel version (1D)

ที่มา : [ออนไลน์] https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

จากรูปจะมีขนาด filter 3×3 และมี elements เป็น [0, 1, 2], [2, 2, 0], [0, 1, 2] ตามลำดับ ทำการ stride ทีละ 1 และมีขนาด padding เท่ากับ 0 เทียบกับขนาดของ input layer เพื่อสร้าง feature map ในการช่วยลดขนาดของข้อมูล

2.) Convolution: the multi-channel version (2D)

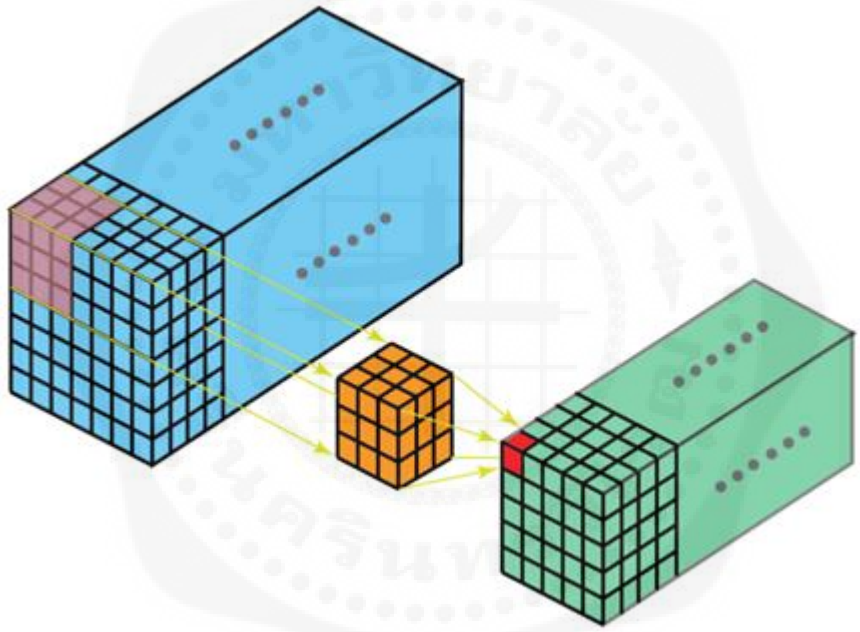


รูปภาพ 4 แสดงตัวอย่างการทำ Convolution: the multi-channel version (2D)

ที่มา : [ออนไลน์] https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

จากรูปจะมี input layer ขนาด 5×5×3 โดยมี filter ขนาด 3×3×3 โดยที่ 2D filter จะเลื่อนไปได้ 2 ทิศทาง คือความสูงและความกว้างของ input layer ขั้นแรกเราจะนำ filter ไปทาบกับ input layer แล้วทำการ stride ทีละ 1 เพื่อให้ได้ feature map ขนาด 3×3 จากนั้นนำมารวมกันเพื่อสร้างเป็น feature เดียวที่มีขนาด 3×3×1 ซึ่งช่วยลดขนาดของข้อมูล ด้วยวิธีนี้เราจะต้องมีขนาดความลึกของ input layer และ filter ที่เท่ากัน จากตัวอย่างจะเท่ากับ 3

3.) 3D Convolution

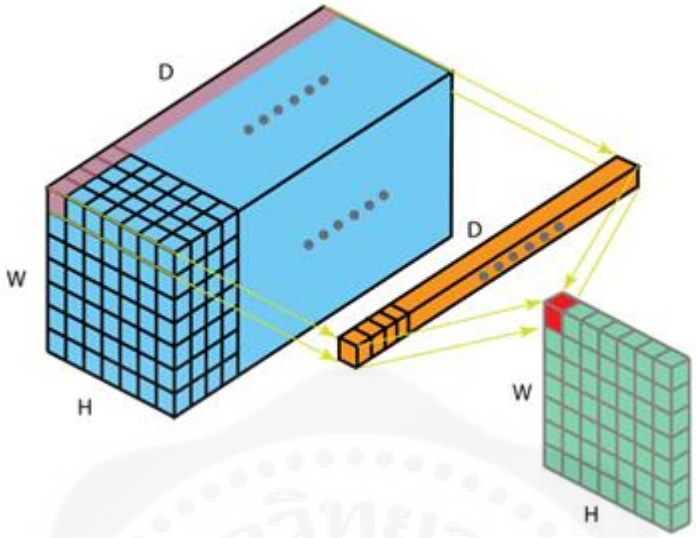


รูปภาพ 5 แสดงตัวอย่างการทำ 3D Convolution

ที่มา : [ออนไลน์] https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

หลักการนี้จะมีขนาดความลึกของ filter น้อยกว่า input layer ด้วยเหตุนี้ทำให้ 3D filter สามารถเลื่อนไปได้ 3 ทิศทาง คือ ความกว้าง , ความสูง , ช่องของ input layer ทำให้ผลลัพธ์ที่ได้จะเป็น feature map ขนาด 3 มิติที่เหมาะกับการใช้ทางการแพทย์ เช่นการสร้างภาพ MRI ในการวินิจฉัยการไหลเวียนของเส้นเลือด เป็นต้น

4.) 1 x 1 Convolution



รูปภาพ 6 แสดงตัวอย่างการทำ 1 x 1 Convolution

ที่มา : [ออนไลน์] https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

จากรูปมี input layer ขนาด H×W×D และมี filter ขนาด 1×1×D จะได้ feature map ขนาด H×W×1 หากเรานำ N ที่มีขนาด 1×1 เข้ามา จะได้ feature map ขนาด H×W×N

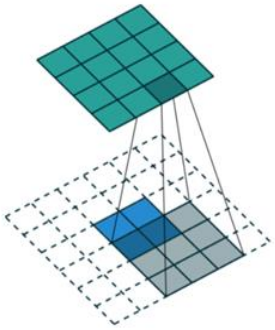
ข้อดีของหลักการนี้ คือ

1. ช่วยลดขนาดเพื่อการคำนวณที่มีประสิทธิภาพ

2. ช่วยลดขนาดในการทำ embedding หรือ pooling

3.ใช้หลักการ nonlinearity หลังการทำ convolution นี้เพื่อเพิ่มการเรียนรู้ของฟังก์ชันให้มากขึ้น

5.) Transposed Convolution (Deconvolution)

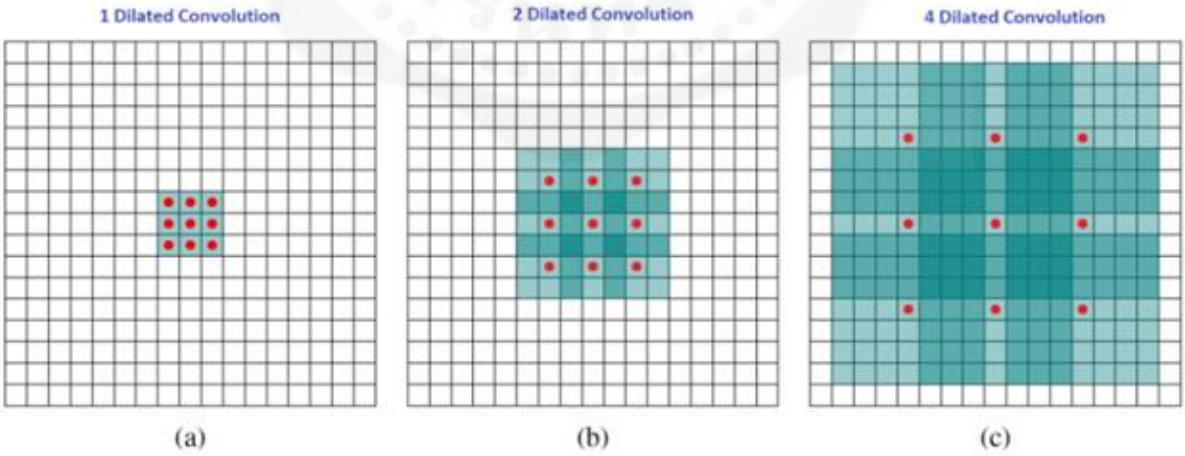


รูปภาพ 7 แสดงตัวอย่างการทำ Transposed Convolution (Deconvolution)

ที่มา : [ออนไลน์] https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

จากรูปจะทำการ Transposed Convolution ขนาด 3×3 อยู่เหนือ input layer ขนาด 2×2 ที่มี padded ขนาด 2×2 โดยจะทำการ stride ไปทีละ 1 เพื่อให้ได้ feature map ที่มีขนาด 4×4 ซึ่งหลักการนี้จะผกผันกับ Convolution โดยจะเป็นขนาดของ input layer เล็กกว่า feature map

6.) Dilated Convolution (Atrous Convolution)



รูปภาพ 8 แสดงตัวอย่างการทำ Dilated Convolution (Atrous Convolution)

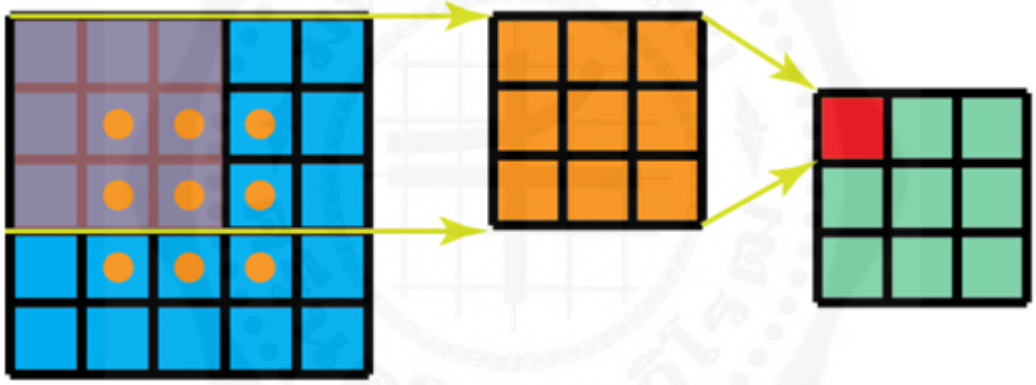
ที่มา : [ออนไลน์] https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

จากรูปจะแสดงจุดสีแดงขนาด 3×3 หลังจากการทำ convolution ถึงแม้จะมีขนาดมิติที่เหมือนกัน แต่ขนาดของ receptive filed(พื้นที่สีเขียว) จะแตกต่างกันออกไป

(a) 3 x 3 for l =1 (b) 7 x 7 for l =2 (c) 15 x 15 for l = 3

แม้ว่าเราจะมีการขยายขนาดของ receptive filed ออกไป ก็ไม่ส่งผลกระทบต่อการคำนวณ ทำให้หลักการนี้เหมาะกับการ convolution ที่มีการซ้อนทับหลายตัวแล้วเราต้องการทำการขยายพื้นที่นั้นโดยไม่ต้องเสียเวลาทำการเพิ่ม kernel

7.) Spatially Separable Convolutions

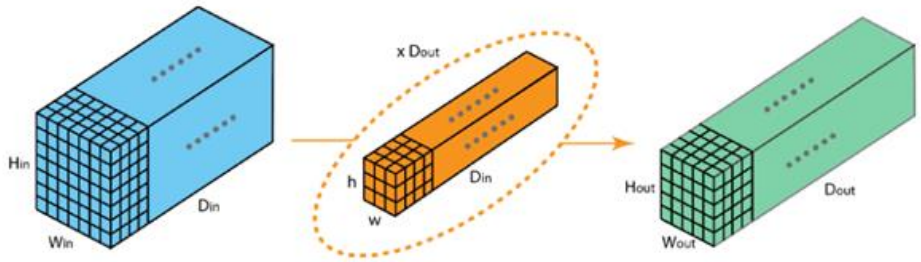


รูปภาพ 9 แสดงตัวอย่างการทำ Spatially Separable Convolutions

ที่มา : [ออนไลน์] https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

จากรูป input layer มีขนาด 5×5 ที่มี kernel ขนาด 3×3 โดยเราจะทำการ stride ทีละ 1 และมี padding 0 เราจะต้องทำการเคลื่อนไปหาจุด 3 จุดทั้งในแนวตั้งและแนวนอน รวมทั้งหมดเป็น 9 จุด (จุดสีเหลือง) โดยในแต่ละตำแหน่งจะใช้การคูณ 9 เข้าไป รวมเป็น 81 ซึ่งหลักการนี้จะเป็นการแยก 2D filter ซึ่งคือความสูงและความกว้างออกจากกันเพื่อช่วยลดขนาดของข้อมูล แต่ส่วนมากจะไม่เป็นที่นิยม เนื่องจากไม่ใช่ทุก kernel ที่จะสามารถแบ่งออกเป็นส่วนได้

8.) Grouped Convolution

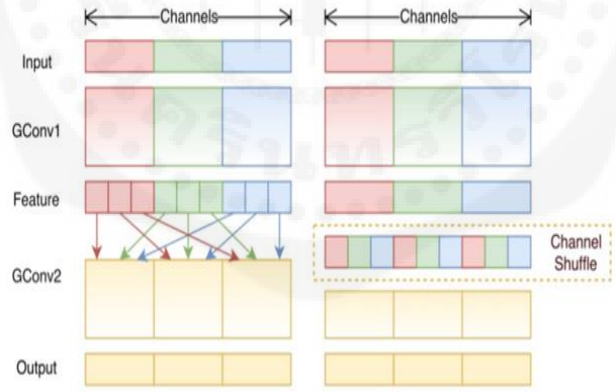


รูปภาพ 10 แสดงตัวอย่างการทำ Grouped Convolution

ที่มา : [ออนไลน์] https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

จากรูป input layer มีขนาด Hin×Win×Din จะถูกแปลงเป็น output ที่มีขนาด Hout×Wout×Dout โดย filter ขนาด h×w×Din

9.) Shuffled Grouped Convolution



รูปภาพ 11 แสดงตัวอย่างการทำ Shuffled Grouped Convolution

ที่มา : [ออนไลน์] https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

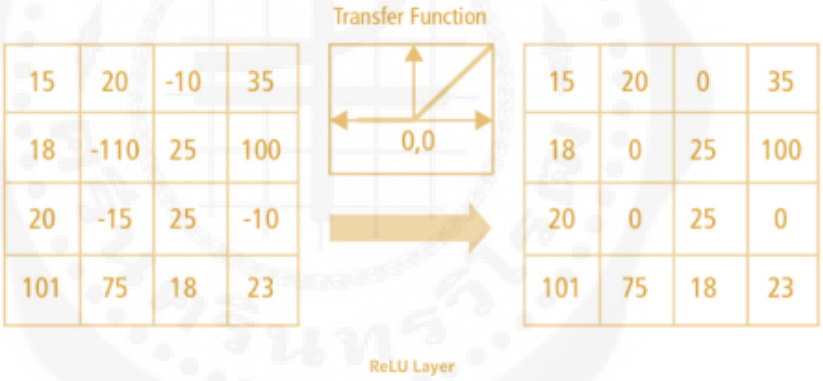
จากรูป filter สีแดงจะทำการประมวลผลข้อมูลที่มาจาก input เท่านั้น ส่วนfilter สีน้ำเงิน ก็จะทำการประมวลผลข้อมูลที่มาจาก layer ก่อนสุดท้ายเท่านั้น ทำให้ filter เหล่านี้ถูกจำกัดการเรียนรู้

ในการแก้ปัญหานี้จึงมีการผสมผสาน filter ต่างๆด้วยการสลับ Channels เพื่อให้ข้อมูลมีการหมุนเวียนกันมากขึ้น

- **Layer 2 : Detector stage** มีหน้าที่รับ Output มาจาก Convolution state และแปลงเป็น nonlinear โดยการระบุ Activation Function ที่จะใช้ทำให้ Output เหมาะสมกับการนำไปใช้ต่อใน Model ของเรา

  <u>Activation ที่ใช้</u>

  - ReLU ย่อมาจาก Rectified Linear Unit ที่เป็น non-linear operation ที่จะได้ผลลัพธ์ของ Output เป็น 0 ถ้าค่าที่ได้เป็น 0 และจะมี Output เป็นค่าอื่นที่มากกว่า 0 ถ้าค่าที่ได้เท่ากับหรือมากกว่า 0 (0 ถึง Infinity) โดยไม่เก็บค่าที่เป็นจำนวนติดลบ

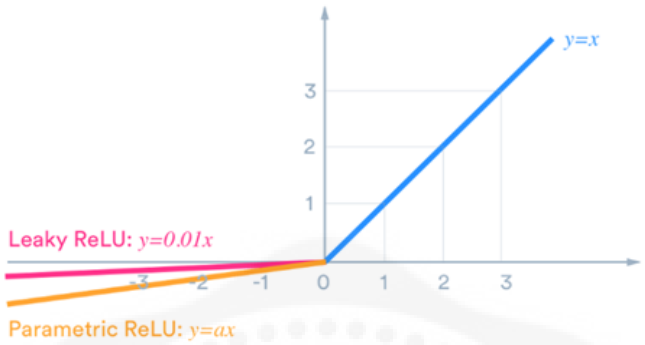


รูปภาพ 12 แสดงการทำงานของ Activation ReLU

ที่มา : [ออนไลน์] https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

โดยจะมี Negative ReLU ที่ช่วยลดปัญหาการที่ค่าติดลบเป็น 0 โดยจะมีค่าความชันของกราฟเพิ่มขึ้นมาเล็กน้อยสำหรับค่าที่ติดลบช่วยให้โมเดลเกิดการเรียนรู้มากขึ้น มี 5 ประเภท

1.) Leaky ReLU ถูกออกแบบมาเพื่อแก้ไขปัญหาของReLU ซึ่งจะเป็นการช่วยเพิ่มrangeของReLU โดย Leaky ReLU จะมีค่าความชันล็กน้อยสำหรับค่าตัวเลขที่ติดลบ แทนที่จะเป็น 0 ทั้งหมด เช่น y = 0.01x when x < 0

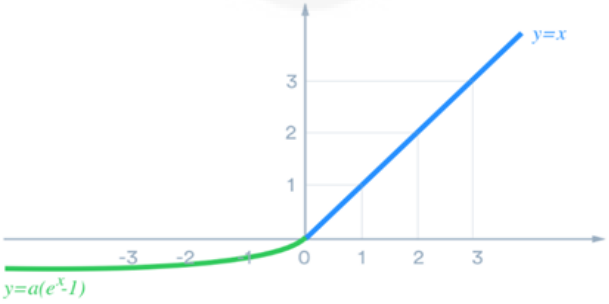


รูปภาพ 13 แสดงการทำ Leaky ReLU

ที่มา : [ออนไลน์] https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7

2.) Parametric ReLU (PReLU) คล้ายกับ Leaky ReLU ที่สร้างพารามิเตอร์สำหรับ neural network ด้วยตัวของมันเอง เช่น y = ax when x < 0 ซึ่ง PReLU จะแก้ปัญหาได้ดีกว่า Sigmoid

3.) Exponential Linear (ELU, SELU) คล้ายกับ Leaky ReLU ที่จะมีค่าความชันล็กน้อยสำหรับค่าตัวเลขที่ติดลบ โดยเส้นของกราฟจะมีความโค้งขึ้นเล็กน้อย

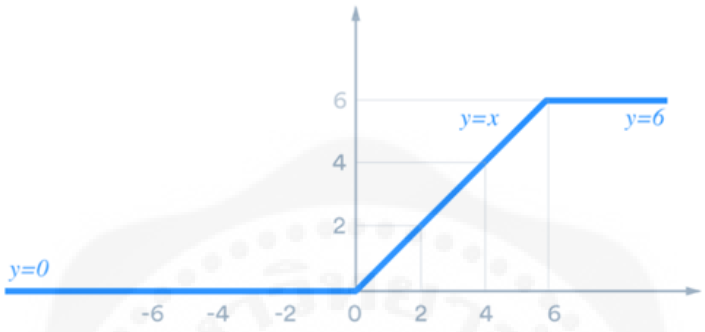


รูปภาพ 14 แสดงการทำ Exponential Linear (ELU, SELU)

ที่มา : [ออนไลน์] https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7

4.) Concatenated ReLU (CReLU) จะมี 2 output คือ ReLU และ negative ReLU โดยจะมีมิติของ output เป็น 2 เท่า

5.) ReLU-6 เรียนรู้คุณสมบัติโมเดลของตนเองจากโมเดลก่อนหน้า โดยจะปรับค่าตามการทำงานที่มีประสิทธิภาพมากที่สุด ในตัวอย่างค่าที่ดีที่สุดจะเป็น 6

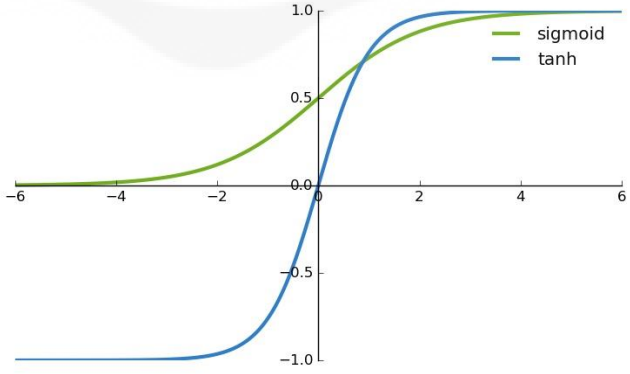


รูปภาพ 15 แสดงการทำ ReLU-6

ที่มา : [ออนไลน์] https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7

ตัวอย่าง Activation อื่นๆ

- **Sigmoid Function** เป็นฟังก์ชัน Logistic Function ที่มีรูปร่างเป็นตัว S และมีค่า Output ระหว่าง 0 - 1 ส่วนใหญ่ถูกใช้ในทางวิเคราะห์ความน่าเป็นที่มี Output เป็น 1กับ 0 ปัจจุบัน Sigmoid Function ไม่เป็นที่นิยม

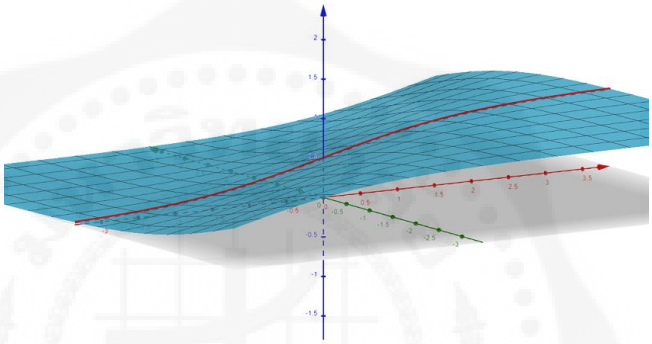


รูปภาพ 16 กราฟของ Sigmoid Function และ Tanh Function

ที่มา : [ออนไลน์] https://hvidberrrg.github.io/deep_learning/activation_functions/sigmoid_function_and_derivative.html

- **Tanh Function** หรือ Hyperbolic Tangent Activation Function เป็นฟังก์ชันที่แก้ปัญหาของ Sigmoid Function ในกรณีที่ Input เข้าใกล้ 0 แล้ว Derivative จะมี Slope ชันกว่า Sigmoid ทำให้ Gradient มากกว่า เทรนได้เร็วกว่า โดยกราฟของ Tanh Function นั้นมีรูปร่างเป็นตัว S เช่นเดียวกับ Sigmoid Function

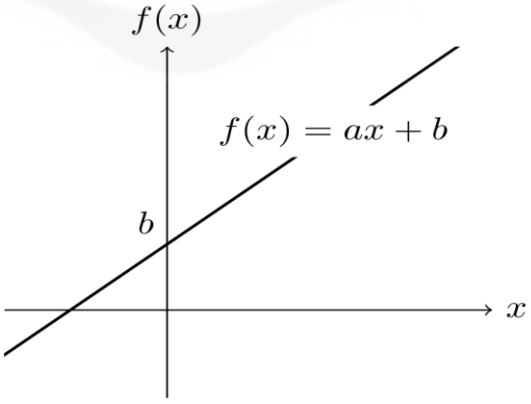
- **Softmax Function** หรือ SoftArgMax Function หรือ Normalized Exponential Function เป็นฟังก์ชันที่รับ Input เป็นvector แล้วแปลงออกมาเป็นความน่าจะเป็นที่มีผลรวมเท่ากับ 1



รูปภาพ 17 กราฟของ Softmax Function

ที่มา : [ออนไลน์] https://themaverickmeerkat.com/2019-10-23-Softmax/

- **Linear Function** เป็นฟังก์ชันที่มีเส้นตรงที่มีm เป็นความชัน หรืออัตราการเปลี่ยนแปลงเฉลี่ยของy เมื่อ เทียบกับx และมีb เป็นจุดตัดแกน y ซึ่งสามารถอยู่ในรูป function ได้

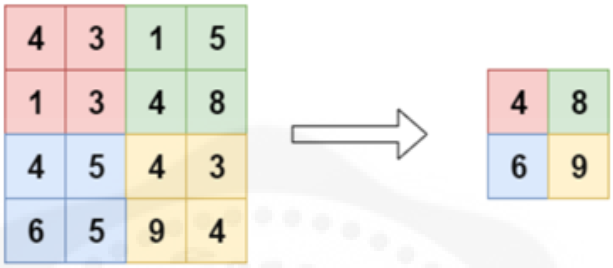


รูปภาพ 18 กราฟของ Linear Function

ที่มา : [ออนไลน์] https://internal.ncl.ac.uk/ask/numeracy-maths-statistics/core-mathematics/pure-maths/functions/linear-functions.html

- **Layer 3 : Pooling Layer** ทำให้ขนาดของ feature map ลดลง มี 4 ประเภท

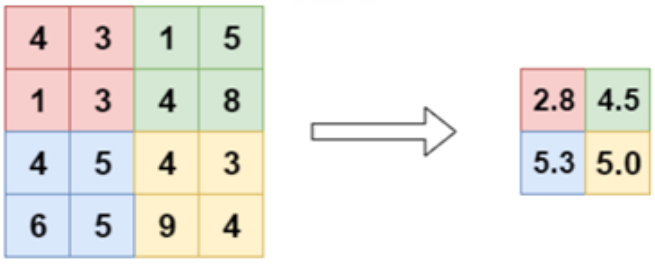
1. Max pooling ได้รับความนิยมในการใช้งาน โดยจะหาค่าสูงสุด (Maximum Value) จะใช้ Window Size ที่เรากำหนด (Window ที่นิยมใช้จะมีขนาด 2×2 หรือ 3×3) และเลื่อนไปตาม stride ที่กำหนด



รูปภาพ 19 แสดงตัวอย่างการทำ Max pooling

ที่มา : [ออนไลน์] https://www.machinecurve.com/index.php/2020/01/30/what-are-max-pooling-average-pooling-global-max-pooling-and-global-average-pooling/

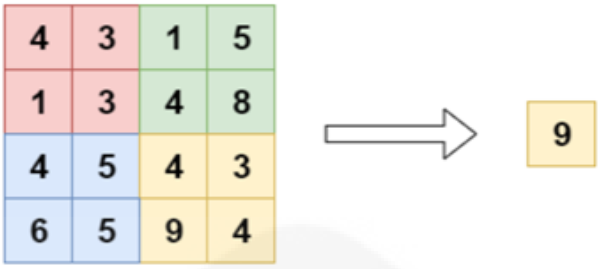
2. Average pooling ใช้ในการหาค่าเฉลี่ย (Average Value) โดยจะใช้ Window Size ที่เรากำหนด (Window ที่นิยมใช้จะมีขนาด 2×2 หรือ 3×3) และเลื่อนไปตาม stride ที่กำหนด ซึ่งจะแตกต่างจาก Max pooling ที่ Average จะสนใจในองค์ประกอบโดยรวมมากกว่าการสนใจในค่าเดียว



รูปภาพ 20 แสดงตัวอย่างการทำ Average pooling

ที่มา : [ออนไลน์] https://www.machinecurve.com/index.php/2020/01/30/what-are-max-pooling-average-pooling-global-max-pooling-and-global-average-pooling/

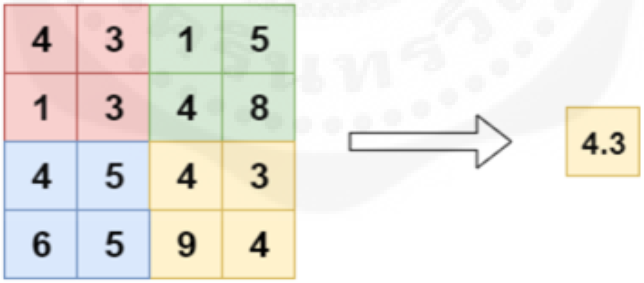
3. Global Max Pooling ใช้ในการลดขนาดของ feature maps output แทนการ Flattening หรือ Dense layer



รูปภาพ 21 แสดงตัวอย่างการทำ Global Max pooling

ที่มา : [ออนไลน์] https://www.machinecurve.com/index.php/2020/01/30/what-are-max-pooling-average-pooling-global-max-pooling-and-global-average-pooling/

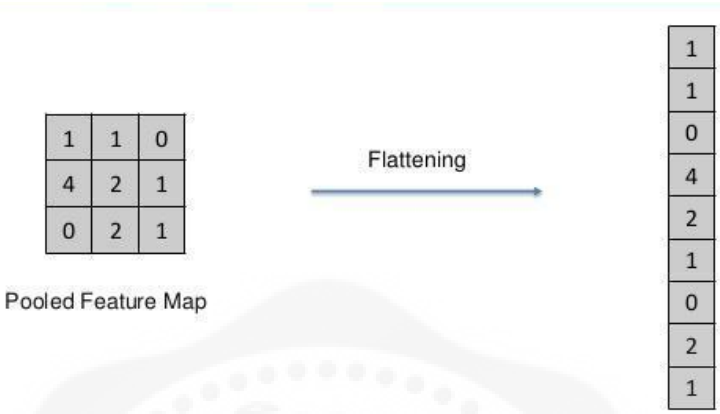
4. Global Average Pooling ส่วนมากจะใช้แทน fully-connected หรือ densely-connected layers ช่วยลดการเกิด overfitting เนื่องจากไม่มีค่าให้ฟังก์ชันเรียนรู้มาก่อนหน้า



รูปภาพ 22 แสดงตัวอย่างการทำ Global Average pooling

ที่มา : [ออนไลน์] https://www.machinecurve.com/index.php/2020/01/30/what-are-max-pooling-average-pooling-global-max-pooling-and-global-average-pooling/

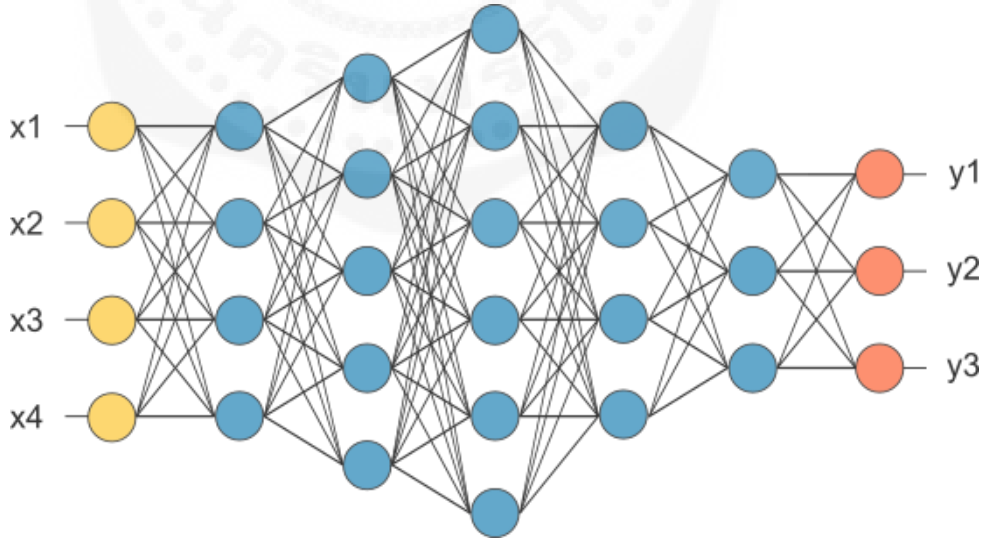
- **Layer 4 : Flatten Layer** เป็นการแปลงข้อมูลที่อยู่ในรูปแบบ Matrix ให้อยู่ในรูปแบบของ Vector (รูปแบบคอลัมน์เดียว) โดยแปลงจากซ้ายไปขวา บนไปล่างตามลำดับ



รูปภาพ 23 แสดงการทำ Flatten

ที่มา : [ออนไลน์] https://www.slideshare.net/KirillEremenko/deep-learning-az-convolutional-neural-networks-cnn-step-3-flattening

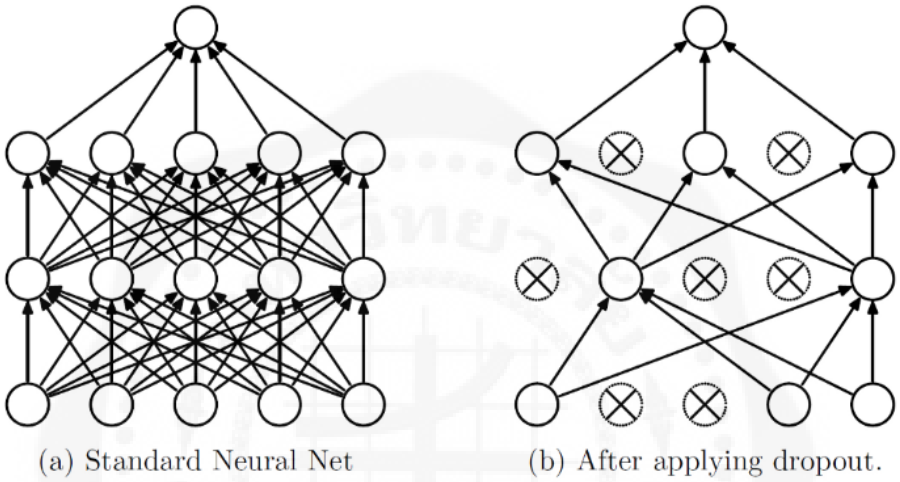
- **Layer 5 : Fully Connected Layer (Dense Layer)** คือการนำ Output จาก Flatten Layer ไปใส่ที่ Input ของ Model และจะทำการเชื่อมต่อ Input Layer กับทุก ๆ Neuron ในชั้นของ Dense Layer ถัดไป เรื่อยๆจนถึง Output Layer



รูปภาพ 24 แสดงการเชื่อมของ Input Layer กับ Dense Layer

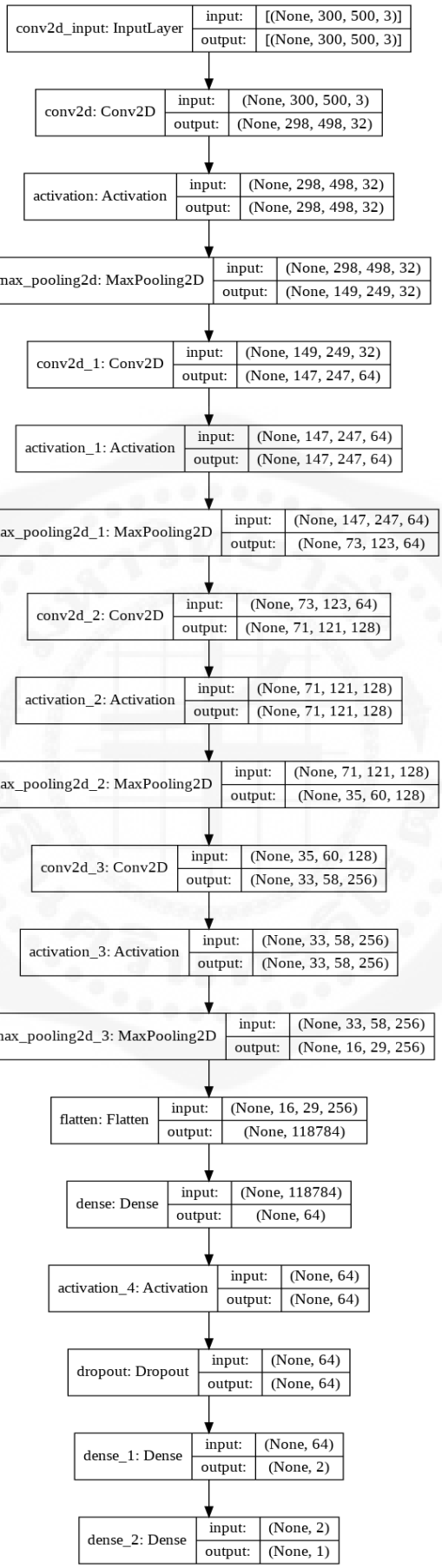
ที่มา : [ออนไลน์] https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

- **Dropout** เป็น Method ที่สุ่มปิด Node ที่ไม่น่าจะเป็น Output ซึ่งสามารถกำหนดค่าได้ 0 ถึง 1 (0-100%) เช่น หากเลือก Dropout 0.25 หมายถึงจะสุ่มปิด Node 25% ของทั้งหมด เป็นวิธีการหนึ่งที่นำมาใช้เพื่อแก้ปัญหาเวลาเทรนแล้วได้ค่า Loss ที่สูง เนื่องจากปัญหา Overfitting ที่เกิดจากการที่เทรนพารามิเตอร์มากเกินไป แต่หากเทรนพารามิตเตอร์น้อยเกินไปก็จะเกิด Underfitting ทำให้สามารถจำลองโมเดลเพิ่มขึ้นจากโมเดลเดียวได้ และทำให้ Generalization Abstraction ดีขึ้น



รูปภาพ 25 แสดงตัวอย่างการ Dropout

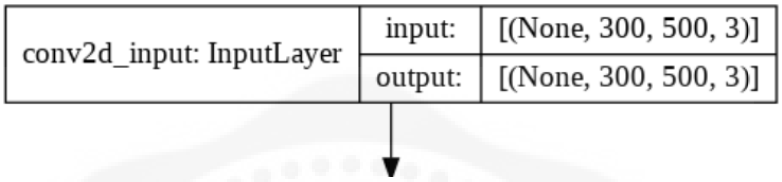
ที่มา : [ออนไลน์] https://medium.com/@pagongatchalee/overfitting-%E0%B8%9B%E0%B8%B1%E0%B8%8D%E0%B8%AB%E0%B8%B2%E0%B8%A2%E0%B8%AD%E0%B8%94%E0%B8%AE%E0%B8%B4%E0%B8%95%E0%B8%82%E0%B8%AD%E0%B8%87%E0%B8%84%E0%B8%99%E0%B8%97%E0%B8%B3-machine-learning-deep-learning-373b1235394c


conv2d_input: InputLayer
input: [(None, 300, 500, 3)]
output: [(None, 300, 500, 3)]
conv2d: Conv2D
input: (None, 300, 500, 3)
output: (None, 298, 498, 32)
activation: Activation
input: (None, 298, 498, 32)
output: (None, 298, 498, 32)
max_pooling2d: MaxPooling2D
input: (None, 298, 498, 32)
output: (None, 149, 249, 32)
conv2d_1: Conv2D
input: (None, 149, 249, 32)
output: (None, 147, 247, 64)
activation_1: Activation
input: (None, 147, 247, 64)
output: (None, 147, 247, 64)
max_pooling2d_1: MaxPooling2D
input: (None, 147, 247, 64)
output: (None, 73, 123, 64)
conv2d_2: Conv2D
input: (None, 73, 123, 64)
output: (None, 71, 121, 128)
activation_2: Activation
input: (None, 71, 121, 128)
output: (None, 71, 121, 128)
max_pooling2d_2: MaxPooling2D
input: (None, 71, 121, 128)
output: (None, 35, 60, 128)
conv2d_3: Conv2D
input: (None, 35, 60, 128)
output: (None, 33, 58, 256)
activation_3: Activation
input: (None, 33, 58, 256)
output: (None, 33, 58, 256)
max_pooling2d_3: MaxPooling2D
input: (None, 33, 58, 256)
output: (None, 16, 29, 256)
flatten: Flatten
input: (None, 16, 29, 256)
output: (None, 118784)
dense: Dense
input: (None, 118784)
output: (None, 64)
activation_4: Activation
input: (None, 64)
output: (None, 64)
dropout: Dropout
input: (None, 64)
output: (None, 64)
dense_1: Dense
input: (None, 64)
output: (None, 2)
dense_2: Dense
input: (None, 2)
output: (None, 1)

<u>โครงสร้าง CNN</u>

1. conv2d_input รับข้อมูล Input ที่เป็นภาพดวงตา ขนาด 300×500 Pixel (สูง×กว้าง) จาก Training Set และ filter 3×3 ซึ่งเป็น Channel ของสี 3 สี คือ น้ำเงิน, เขียว และแดง โดยจะเป็น Default เริ่มต้นเมื่อใส่ข้อมูล Input ที่เป็นรูปภาพ ซึ่งแต่ละรูปภาพจะมี Pixel ที่แยกส่วนกันด้วยสี เป็นค่า Value ตั้งแต่ 0-255

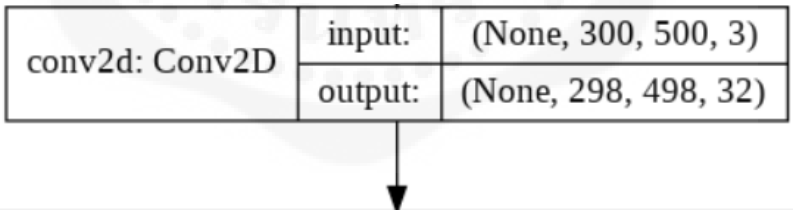
   ผลลัพธ์ที่ได้ คือ ภาพ Input ขนาด : สูง×กว้าง×ช่องสี = za = 450,000 Pixel



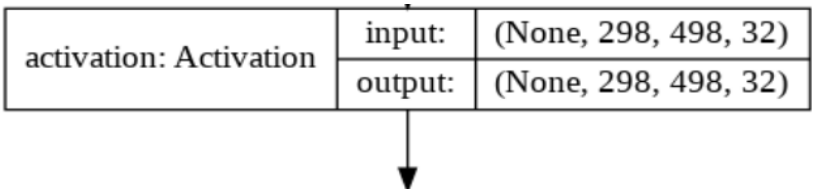
2. ส่ง Input เข้าไปใน Convolution Layer ซึ่งเป็นลดขนาดรูปภาพและการดึงคุณลักษณะของข้อมูลภายในรูปภาพ Input ด้วย Window ขนาด 32×32 จนได้ Parameter ออกมา
   การทำ Convolution ที่มีหลายขนาดนั้น จะช่วยให้ดึงคุณลักษณะของภาพได้ละเอียดมากยิ่งขึ้น และช่วยเพิ่ม Parameter ในการเทรน Model ว่าลักษณะของวัตถุที่ต้องการนั้นมีส่วนประกอบอะไรบ้าง

   ผลลัพธ์ที่ได้ คือ ภาพขนาด : สูง×กว้าง×ช่องสี = 298×498×3 = 445,212 Pixel

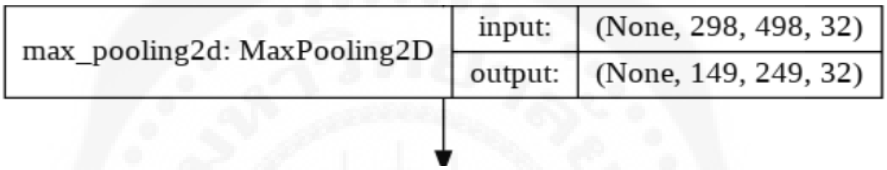
   และได้ Feature Map = (n - f +1) × (n - f + 1) = (300-3+1) × (500-3+1) = 298×498



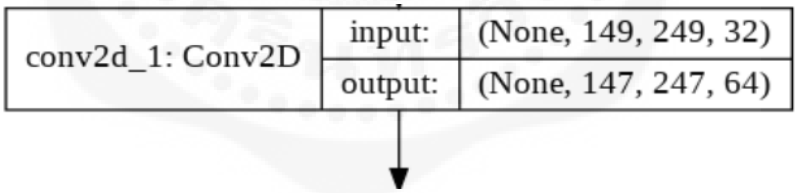
3. จากนั้นส่งภาพขนาด 298×498 ไปที่ Activation ด้วยวิธีการ ReLU เพื่อให้ได้ข้อมูลคุณลักษณะที่เป็นค่า Value 0 ถึง infinity เท่านั้น (ไม่เก็บค่าที่ติดลบ)



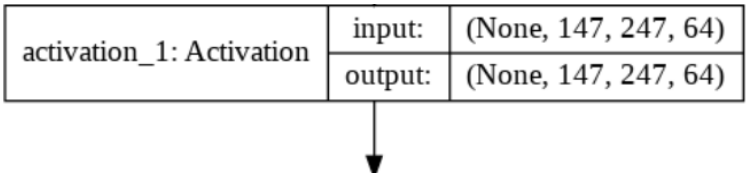
4. ส่งภาพขนาด 298×498 Pixel ที่ทำการ Activation แล้ว ไปที่ Pooling Layer โดยจะทำการ Max Pooling เพื่อลดขนาดภาพลงและเลือกคุณลักษณะสำคัญด้วยการหา Maximum Value ภายใน Filter ขนาด 2×2 และ Stirde = 1 จะได้ภาพขนาด 149×249



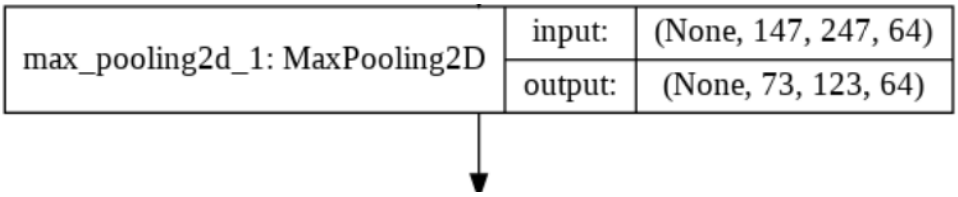
5. รับข้อมูลจาก Pooling Layer มาทำการลดขนาดรูปภาพและการดึงคุณลักษณะของข้อมูลภายในรูปภาพ ด้วย conv2d จากภาพขนาด 149×249 Pixel โดยใช้ Window ขนาด 64×64 และได้ Feature Map = (n - f +1) × (n - f + 1) = (149-3+1) × (249-3+1) = 147×247
ผลลัพธ์ที่ได้ คือ ภาพขนาด : สูง×กว้าง×ช่องสี = 147×247×3 = 108,927 Pixel



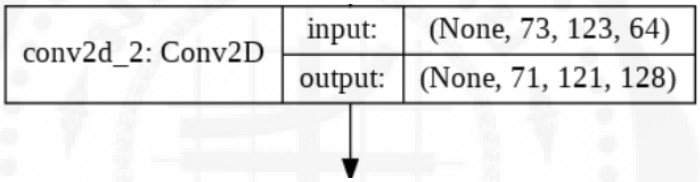
6. จากนั้นส่งภาพขนาด 147×247 Pixel ไปที่ Activation ด้วยวิธีการ ReLU เพื่อให้ได้ข้อมูลคุณลักษณะที่เป็นค่า Value 0 ถึง infinity เท่านั้น (ไม่เก็บค่าที่ติดลบ)



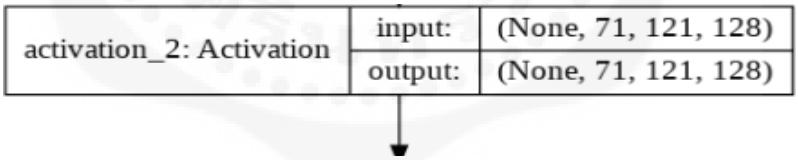
7. ส่งภาพขนาด 147×247 Pixel ที่ทำการ Activation แล้ว ไปที่ Pooling Layer โดยจะทำการ Max Pooling เพื่อลดขนาดภาพลงด้วยการหา Maximum Value ภายใน Filter ขนาด 2×2 และ Stirde = 1 จนได้ภาพขนาด 73×123 Pixel



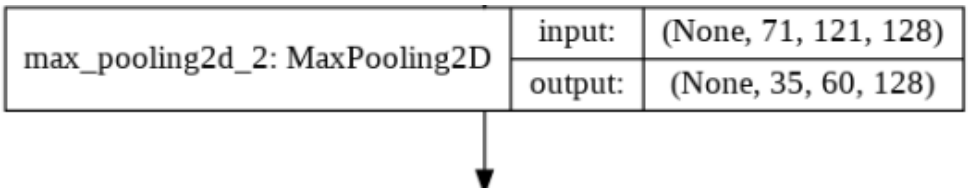
8. รับข้อมูลจาก Pooling Layer มาทำการลดขนาดรูปภาพและการดึงคุณลักษณะของข้อมูลภายในรูปภาพ ด้วย conv2d จากภาพขนาด 73×123 Pixel โดยใช้ Window ขนาด 128×128 และได้ Feature Map = (n - f +1) × (n - f + 1) = (73-3+1) × (123-3+1) = 71×121
ผลลัพธ์ที่ได้ คือ ภาพขนาด : สูง×กว้าง×ช่องสี = 71×121×3 = 25,773 Pixel



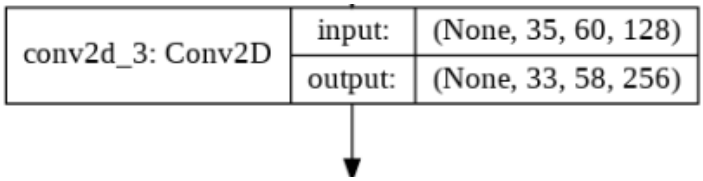
9. จากนั้นส่งภาพขนาด 71×121 Pixel ไปที่ Activation ด้วยวิธีการ ReLU เพื่อให้ได้ข้อมูลคุณลักษณะที่เป็นค่า Value 0 ถึง infinity เท่านั้น (ไม่เก็บค่าที่ติดลบ)



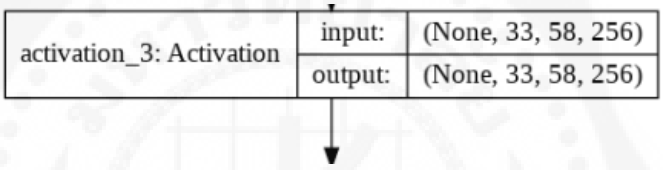
10. ส่งภาพขนาด 71×121 Pixel ที่ทำการ Activation แล้ว ไปที่ Pooling Layer โดยจะทำการ Max Pooling เพื่อลดขนาดภาพลงด้วยการหา Maximum Value ภายใน Filter ขนาด 2×2 และ Stirde = 1 จนได้ภาพขนาด 35×60 Pixel



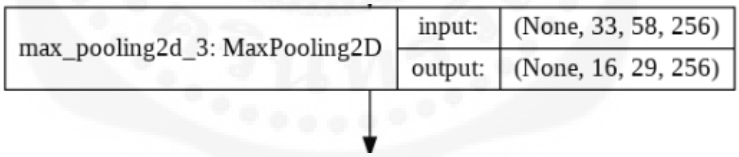
11. รับข้อมูลจาก Pooling Layer มาทำการลดขนาดรูปภาพและการดึงคุณลักษณะของข้อมูลภายในรูปภาพ ด้วย conv2d จากภาพขนาด 35×60 Pixel โดยใช้ Window ขนาด 256×256 และได้ Feature Map = (n - f +1) × (n - f + 1) = (35-3+1) × (60-3+1) = 33×58
ผลลัพธ์ที่ได้ คือ ภาพขนาด : สูง×กว้าง×ช่องสี = 33×58×3 = 5,742 Pixel



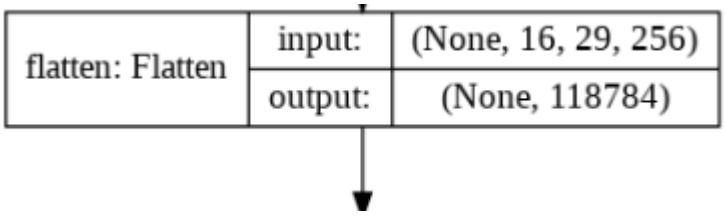
12. จากนั้นส่งภาพขนาด 33×58 Pixel ไปที่ Activation ด้วยวิธีการ ReLU เพื่อให้ได้ข้อมูลคุณลักษณะที่เป็นค่า Value 0 ถึง infinity เท่านั้น (ไม่เก็บค่าที่ติดลบ)



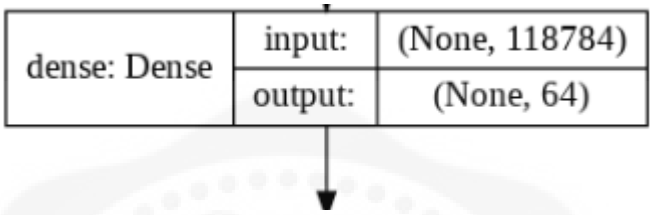
13. ส่งภาพขนาด 33×58 Pixel ที่ทำการ Activation แล้ว ไปที่ Pooling Layer โดยจะทำการ Max Pooling เพื่อลดขนาดภาพลงด้วยการหา Maximum Value ภายใน Filter ขนาด 2×2 และ Stirde = 1 จนได้ภาพขนาด 16×29 Pixel



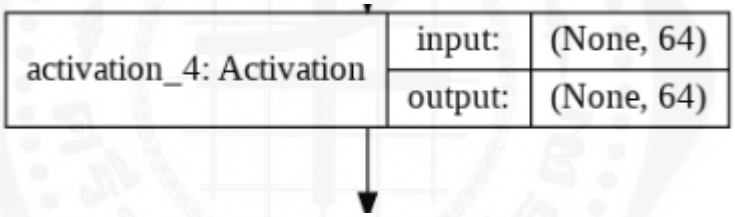
14. จากนั้นนำข้อมูลที่มีอยู่ในรูปแบบ Matrix ส่งไปที่ Flatten Layer เพื่อแปลงเป็นข้อมูลเป็น Vector ซึ่งจะได้ข้อมูล 1 Column ที่มีขนาด 118,784 Pixel



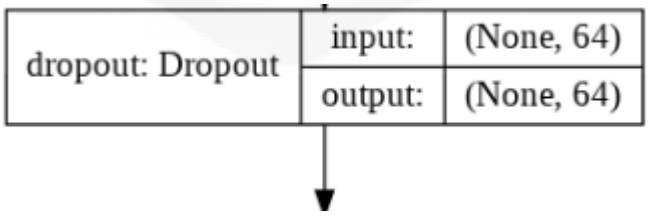
15. นำ Output จาก Flatten Layer ส่งไปที่ Fully Connected Layer ที่จะรับ Input และทำการเชื่อมต่อ Input กับทุก ๆ Node ของชั้นถัดไปที่เป็น Dense Layer โดยกำหนดให้ Dense Layer มีขนาด 64 Node คือ ทุก Node ของ Input จำนวน 118,784 จะถูกนำไปใส่ในทุก Node ของ Dense Layer ที่เรากำหนดไว้ว่าจะมี 64 Node เพื่อนำไปสร้าง Neural Network ที่เป็นการสร้างโครงข่ายประสาทเทียมที่จะใช้ในการเทรนโมเดล Classification โดยแต่ละจุดที่เชื่อมต่อกันจะมี Weight ที่ต่างกัน



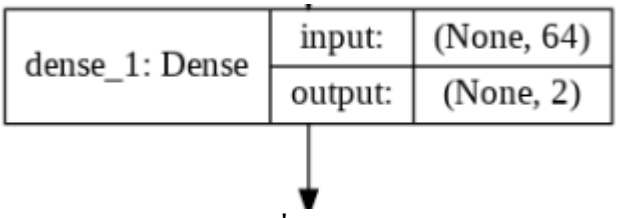
16. จากนั้นส่งไปที่ Activation ด้วยวิธีการ ReLU เพื่อให้ได้ข้อมูลคุณลักษณะที่เป็นค่า Value 0 ถึง infinity เท่านั้น (ไม่เก็บค่าที่ติดลบ)



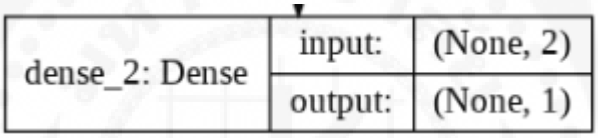
17. ทำการ Dropout เพื่อสุ่มปิด Node ที่ไม่น่าจะเป็น Output จำนวน 50% ของข้อมูลทั้งหมด เพื่อลดปัญหาในการเกิด Overfitting จากการที่เทรนข้อมูลที่มี Parameter มากเกินไป



18. ส่งข้อมูลที่ Dropout แล้วไปทำการเชื่อมต่อกับทุก ๆ Node ของ Dense Layer ถัดไป โดยกำหนดให้มีขนาด 2 Node เพื่อ เป็นการแยก Parameter ที่คล้ายกันทั้งหมดออกเป็น 2 กลุ่ม คือ Glaucoma (ภาพดวงตาที่มีอาการต้อหิน) และ Normal (ภาพดวงตาปกติ)

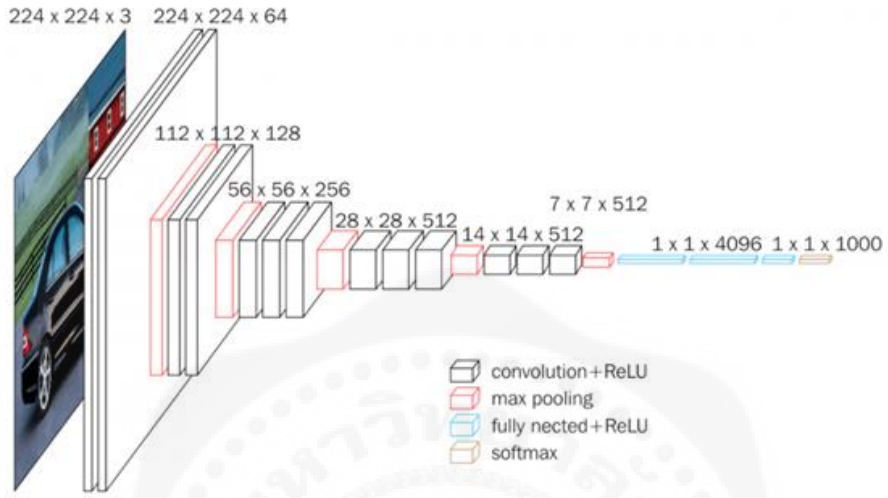


19. ส่งข้อมูลที่ประเภท Glaucoma (ภาพดวงตาที่มีอาการต้อหิน) กับ Normal (ภาพดวงตาปกติ) ไปทำการเชื่อมต่อกับทุก ๆ Node ของ Dense Layer ถัดไป โดยกำหนดให้มีขนาด 1 Node เพื่อให้เป็น Output ของการ Predict Model ว่าภาพที่เราทำการส่งเข้าไป Predict นั้นเป็น Glaucoma หรือ Normal



2.2.3) Architecture Network อื่นๆ ที่นำมาทดลอง

- VGG 16 model

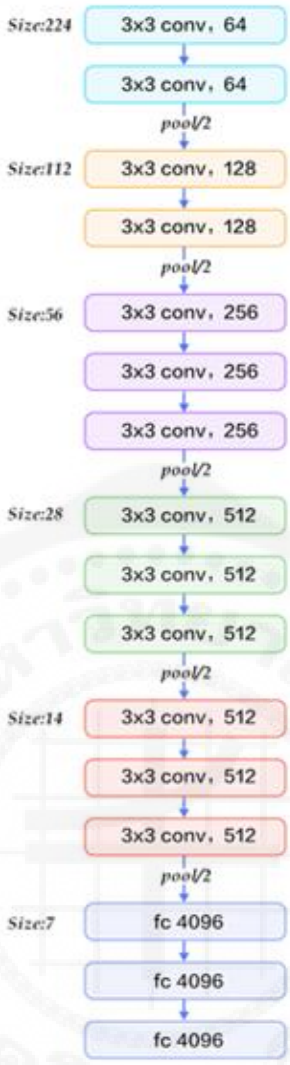


รูปภาพ 26 แสดงโครงข่ายประสาทเทียม VGG16

ที่มา : [ออนไลน์] https://neurohive.io/en/popular-networks/vgg16/

VGG16 เป็นโครงข่ายประสาทเทียม (convolutional neural network) ที่ถูกคิดค้นโดย K. Simonyan และ A. Zisserman จากมหาวิทยาลัยออกซ์ฟอร์ด โดยที่โมเดลรูปแบบนี้มีความแม่นยำในการทดสอบสูงสุดถึง 92.7% จากการทดสอบกับ ImageNet ซึ่งเป็นชุดข้อมูลที่มีรูปภาพมากกว่า 14 ล้านรูปภาพที่อยู่ในclass 1,000 classes

VGG16 ประกอบด้วยเลเยอร์ Convolutional 16 ชั้น เนื่องจากเป็นสถาปัตยกรรมที่คล้ายกับ AlexNet ที่มีเพียง 3x3 convolutions แต่หลาย filters โดยที่ VGG16 มีพารามิเตอร์ทั้งหมด 138 ล้านพารามิเตอร์ มีขนาด convolutional kernels 3x3 และ maxpool kernels ที่ 2x2 และ 2 stride



รูปภาพ 27 แสดงจำนวนชั้นเลเยอร์ใน VGG16

ที่มา : [ออนไลน์] https://iq.opengenus.org/vgg16/

<u>VGG16 ประกอบไปด้วย 16 layers ดังนี้</u>

1. Convolution using 64 filters

2. Convolution using 64 filters + Max pooling

3. Convolution using 128 filters

4. Convolution using 128 filters + Max pooling

5. Convolution using 256 filters

6. Convolution using 256 filters

7. Convolution using 256 filters + Max pooling

8. Convolution using 512 filters

9. Convolution using 512 filters

10. Convolution using 512 filters + Max pooling

11. Convolution using 512 filters

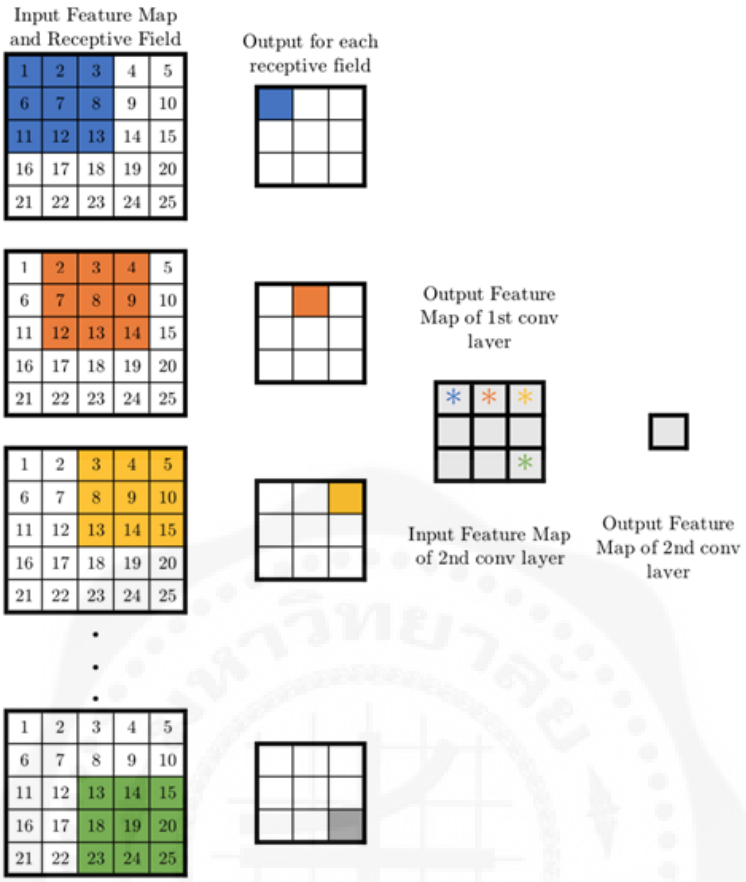
12. Convolution using 512 filters

13. Convolution using 512 filters+Max pooling

14. Fully connected with 4096 nodes

15. Fully connected with 4096 nodes

16. Output layer with Softmax activation with 1000 nodes.



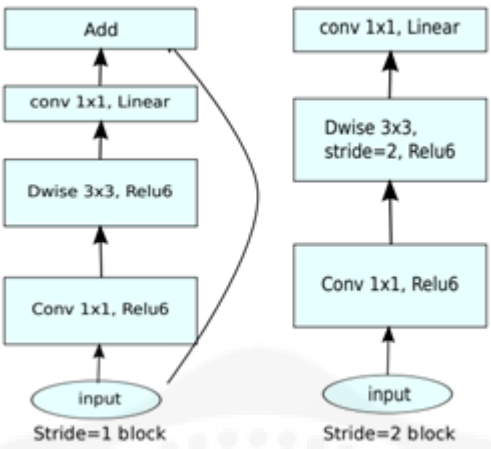
รูปภาพ 28 แสดงตัวอย่างของ input ที่ขนาด 5x5x1

ที่มา : [ออนไลน์] https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96

จากตัวอย่างข้างต้นเมื่อมีการเข้ามาที่ input layer ขนาด 5x5x1 จากนั้นนำไปที่ convolution layer ที่ kernel ขนาด 5x5 และ stride ไป 1 ช่อง จะได้ลักษณะของ output feature map ที่ขนาด 1x1 ในขณะเดียวกันลักษณะของค่า output feature map ที่ได้จะเหมือนกับการใช้ convolution layer ชั้นที่ 2 ขนาด 3x3 และ stride ไป 1 ช่องที่ได้ค่าเหมือนกัน

จำนวนตัวแปรของข้อมูลที่ใช้ในการเทรน สำหรับชั้น convolution ที่มีขนาด 5x5 = 25 และชั้นที่ 2 ของ convolution layer มีขนาด 3x3 จะมีทั้งหมด 3x3x2 = 18 (ลดลงไป 28%) จำนวนตัวแปรหลังจากการเทรนลดลง หมายถึงตัวแปรจะเกิดการเรียนรู้ที่เร็ว และมีประสิทธิภาพต่อการเกิด overfitting มากยิ่งขึ้น

- MobileNetV2



รูปภาพ 29 แสดงตัวอย่างการทำงานของ MobileNetV2

ที่มา : [ออนไลน์] https://paperswithcode.com/method/mobilenetv2

MobileNetV2 เป็นโครงข่ายประสาทเทียม (convolutional neural network) ที่ทำงานได้ดีบนอุปกรณ์มือถือ ขึ้นอยู่กับการทำงานด้วยโครงสร้างแบบคอขวด ส่วนตรงกลางจะใช้การทำงานแบบ depthwise convolutions เพื่อกรองข้อมูล non-linearity จะสามารถแสดงได้ว่าสถาปัตยกรรมแบบ MobileNetV2 จะประกอบไปด้วย fully convolution layer 32 filters และบริเวณคอขวดอีก 19 layers ซึ่ง MobileNetV2 สร้างขึ้นจากแนวคิดเดิมของ MobileNetV1

```
tf.keras.applications.MobileNetV2(
    input_shape=None,
    alpha=1.0,
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
    **kwargs
)
```

รูปภาพ 30 แสดง MobileNetV2 function

ที่มา : [ออนไลน์] https://keras.io/api/applications/mobilenet/

Arguments ที่ใช้งาน

- input_shape : ไม่จำเป็นต้องระบุขนาดในส่วนตรงนี้ก็ได้ ยกเว้นต้องการใช้โมเดลที่มีความละเอียดของรูปภาพที่ไม่ใช่ขนาด (224, 224, 3) สามารถระบุขนาดอินพุตลงไปได้ ซึ่งขนาดของอินพุตควรมี 3 channels และหากต้องการใช้ค่าจาก input_tensor ก็สามารถไม่ต้องระบุค่าได้ เว้นแต่จะใช้ค่าทั้ง input_tensor และ input_shape ระบบจะเลือกใช้ค่าจาก input_shape ถ้าค่า shape ของทั้งคู่ตรงกัน หากไม่ตรงกันจะแสดงผล error

- alpha : เป็นค่าทศนิยมระหว่าง 0 กับ 1 ที่ระบุถึงความกว้างของ network บน MobileNetV2

alpha < 1.0 : จำนวน filters ในแต่ละ layers จะลดลง

alpha > 1.0 : จำนวน filters ในแต่ละ layers จะเพิ่มขึ้น

alpha = 1.0 : จำนวน filters ในแต่ละ layers จะเท่าเดิม

- include_top : ค่า true/false ของ fully-connected layer ที่เป็นชั้นบนสุดของ network โดยค่า default จะเป็นค่า true

- weights : เป็นค่า string โดย none คือการสุ่มชื่อย่อมาใช้งาน และ imagenet คือการ pre-training จาก ImageNet มาใช้งาน หรือเป็นการเรียก path ไฟล์ที่ต้องการจะโหลด

- input_tensor : ไม่จำเป็นต้องระบุค่าในส่วนนี้ได้ ใช้สำหรับ input รูปภาพของ model

- pooling : เป็นค่า string จะถูกใช้งานเมื่อส่วน include_top มีค่าเป็น false

none : ค่า output ที่ได้ของ model จะเป็นรูปแบบ 4D tensor ของ convolutional block ชั้นสุดท้าย

avg : เป็นค่า global average pooling ที่จะนำไปใช้กับ output ของ convolutional block ชั้นสุดท้าย โดยจะเป็นรูปแบบ 2D tensor

max : ค่าของ global max pooling

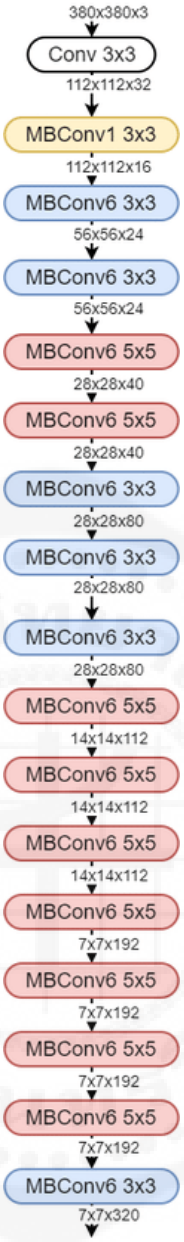
- classes : เป็นค่าจำนวนเต็ม โดยตัวเลขจะเป็นการจัดประเภท classes ของรูปภาพ สามารถระบุค่าในส่วนนี้ได้เฉพาะกรณีที่ include_top เป็น true และไม่มีการระบุ weights arguments

- classifier_activation : ค่า string หรือค่าที่เรียกใช้งาน activation function ใช้กับ top layer เช่น softmax เป็นต้น

- **kwargs : สำหรับการ backwards compatibility เท่านั้น

- EfficientNet B0

EfficientNet B0 เป็นโครงข่ายประสาทเทียมที่ได้รับการฝึกฝนกับภาพมากกว่าหนึ่งล้านภาพจากฐานข้อมูล โดยจะสามารถจำแนกรูปภาพออกเป็นหมวดหมู่วัตถุได้ 1,000 ประเภท สามารถรู้การลักษณะของภาพที่มีความหลากหลาย มีขนาดอินพุตภาพ 224 x 224



รูปภาพ 31 การทำงานของ EfficientNetB0

ที่มา : [ออนไลน์] https://www.researchgate.net/figure/The-architecture-of-EfficientNet-b0_fig1_339462624

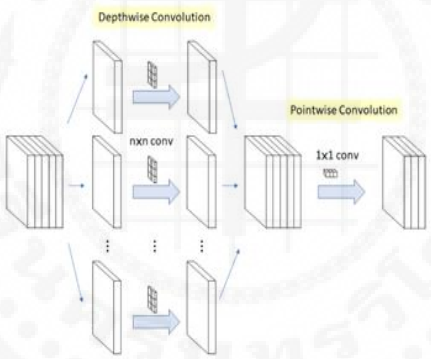
- Xception

Xception นั้นมาก่อน MobileNet และใช้ depthwise separable convolution เช่นเดียวกัน แต่ต่างจาก depthwise separable convolution แบบดั้งเดิมตรงที่จะทำ pointwise convolution ก่อน แล้วจึงค่อยทำ depthwise convolution โดยพัฒนา Inception ที่ทำ 1x1 convolution เพื่อลดจำนวน channel ก่อน

ซึ่ง Xception มีประสิทธิภาพดีกว่า Inception V3 เพราะประสิทธิภาพที่เพิ่มขึ้นจึงไม่ได้เกิดจากความจุที่เพิ่มขึ้น แต่เป็นการใช้พารามิเตอร์แบบจำลองให้มีประสิทธิภาพมากขึ้น

<u>Xception จะมีการใช้งานการทำงานรูปแบบต่างๆดังนี้</u>

1. Original Depthwise Separable Convolution



รูปภาพ 32 การทำงานแบบ Original Depthwise Separable Convolution

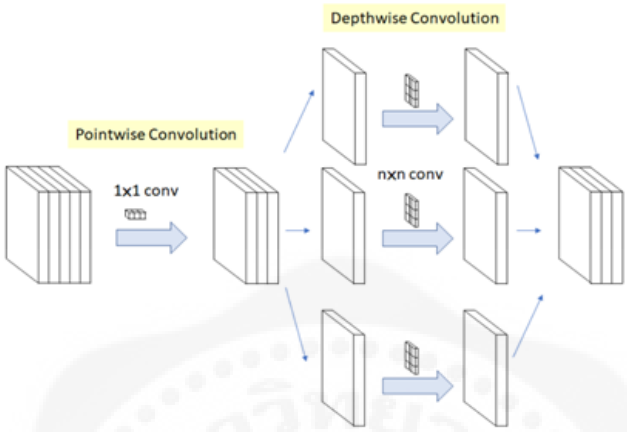
ที่มา : [ออนไลน์] https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568

โดย original depthwise separable convolution จะมี Depthwise convolution และ Pointwise convolution

1.) Depthwise convolution เป็นการแปลงพื้นที่แบบ n x n อย่างเช่น ถ้าเรามีพื้นที่ 5 ช่อง จะทำให้เกิดการแบ่งพื้นที่เป็น 5 n x n

2.) Pointwise convolution เป็นการแปลงพื้นที่แบบ 1 x 1 เพื่อเป็นการเปลี่ยนมิติ

2. Modified Depthwise Separable Convolution in Xception



รูปภาพ 33 การทำงานแบบModified Depthwise Separable Convolution in Xception

ที่มา : [ออนไลน์] https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568

เป็นการปรับพื้นที่ด้วยวิธี inception โดยการใช้ Inception-v3 กับการแปลงพื้นที่เป็น 1 x 1 ซึ่งจะทำเสร็จก่อนการแปลงพื้นที่แบบ n x n

การทำงานแบบ Original Depthwise Separable Convolution กับ Modified Depthwise Separable Convolution in Xception มีความแตกต่างกันอยู่ คือการทำงานแบบ Original Depthwise Separable Convolution จะใช้การแปลงพื้นที่แบบ n x n ก่อน (Original Depthwise Separable Convolution) หลังจากนั้นจึงจะทำการแปลงพื้นที่แบบ 1 x 1 เพื่อเปลี่ยนมิติ (Pointwise convolution) ซึ่งจะแตกต่างจากการทำงานแบบ Modified Depthwise Separable Convolution in Xception โดยการทำงานแบบ Modified Depthwise Separable Convolution in Xception จะทำการแปลงพื้นที่แบบ 1 x 1 เพื่อเปลี่ยนมิติก่อน แล้วจึงทำการแปลงพื้นที่แบบ n x n

- Inception

Inception Net v3 เป็น convolutional neural network architecture จากกลุ่ม Inception family เพื่อปรับปรุงประสิทธิหลายๆอย่าง และใช้ Label Smoothing, Factorized 7 x 7 convolutions และรวมถึง classifer ใช้เพื่อ propagate label information ที่ต่ำกว่า network (พร้อมใช้ batch normalization สำหรับ layer ใน sidehead)

Inception v3 Architecture

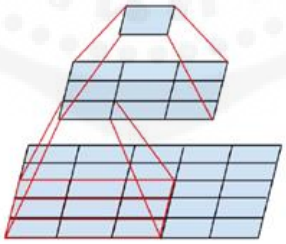
1. Factorized Convolutions เป็นการลดจำนวนพารามิเตอร์ใน network เพื่อลดการคำนวณ และยังช่วยตรวจสอบประสิทธิภาพของ network ด้วย

1.1) Smaller convolutions แทนที่ convolutions ที่ใหญ่ ด้วย convolutions ที่เล็กลง โดยการใช้ filters 3×3 จำนวน 2 ครั้งแทน filter 5×5

หากใช้ filters 5×5 จำนวน 1 layer โดยจะได้ Parameters จำนวน 5×5 = 25

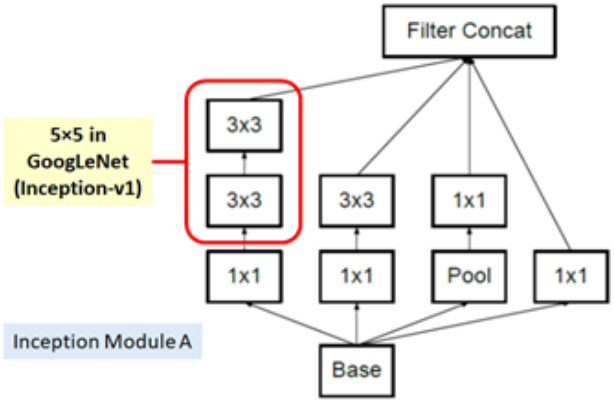
แต่ถ้าใช้ filters 3×3 จำนวน 2 layer โดยจะได้ Parameters จำนวน 3 × 3 + 3 × 3 = 18

ซึ่งหมายความว่า จำนวน Parameter ลดลงไป 28%



รูปภาพ 34 แสดงการใช้ filter 3×3 จำนวน 2 layer แทน filter 5×5 จำนวน 1 layer

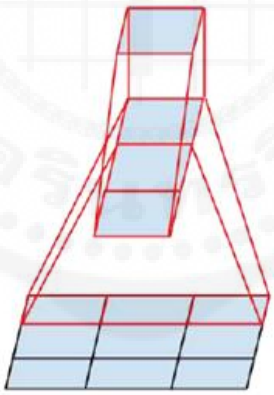
ที่มา : [ออนไลน์] https://sh-tsang.medium.com/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c



รูปภาพ 35 แสดง model ของ Inception-v1

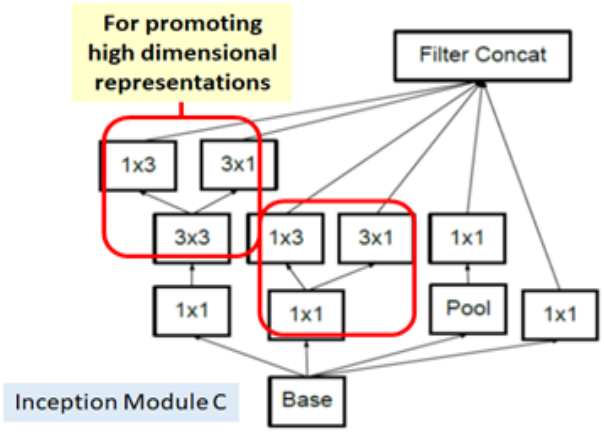
ที่มา : [ออนไลน์] https://sh-tsang.medium.com/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c

1.2) Asymmetric convolutions ตัว convolution 3×3 สามารถแทนที่ด้วย filter 3×1 ตามด้วย 1×3 โดยจะได้จำนวน parameters ทั้งหมด (3×1)+(1×3) = 6 ทำให้จำนวนพารามิเตอร์ลดลง 33%



รูปภาพ 36 แสดงการใช้ filter 1×3 กรอง filter 3×3

ที่มา : [ออนไลน์] https://sh-tsang.medium.com/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c

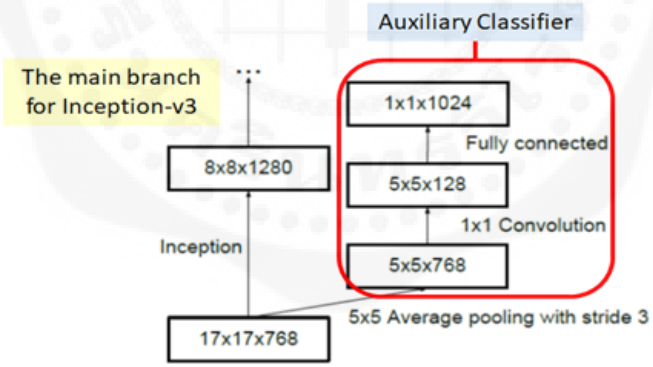


รูปภาพ 37 แสดงภาพ Inception Module C ที่ใช้ filter 3×1 และ 1×3

ที่มา : [ออนไลน์] https://sh-tsang.medium.com/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c

2. Auxiliary classifier คือ CNN ขนาดเล็ก ที่แทรกระหว่าง layer ในตอน training และ loss incurred จะถูกเพิ่มไปใน main network loss

โดยใช้ 1 auxiliary classifier ใส่ก่อน layer สุดท้ายที่เป็น 17×17

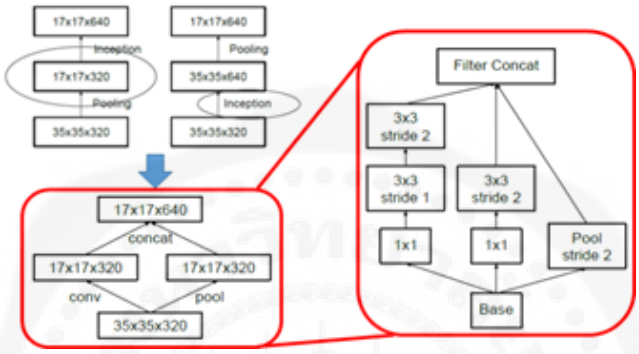


รูปภาพ 38 แสดงภาพ Auxiliary classifier

ที่มา : [ออนไลน์] https://sh-tsang.medium.com/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c

3. Grid size reduction โดยปกติการลดขนาด Grid ทำได้โดยการใช้ pooling operations แต่เพื่อแก้ปัญหา bottlenecks ของ computational cost ซึ่งเสนอวิธีที่มีประสิทธิภาพเพิ่มขึ้น

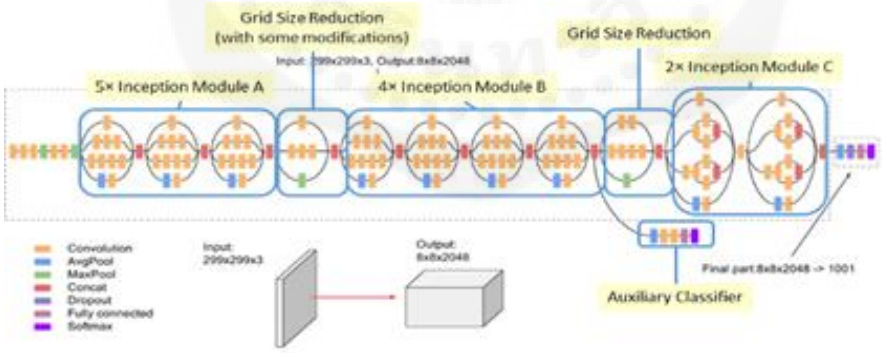
โดยการลดขนาด grid จะได้ 320 feature maps ที่ถูกสร้างจาก convolution และ stride ทีละ 2 step จากนั้นจะถูกนำไป max pooling และทั้ง 2 ชุด feature maps จะเชื่อมกันเป็น 640 feature maps ก่อนจะส่งไปที่ inception module ถัดไป



รูปภาพ 39 แสดงภาพการทำ Grid size reduction

ที่มา : [ออนไลน์] https://sh-tsang.medium.com/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c

4. Inception-v3 Architecture ได้ผลลัพธ์คือ



รูปภาพ 40 แสดงภาพ Inception-v3 Architecture

ที่มา : [ออนไลน์] https://sh-tsang.medium.com/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c

เป็นการรวมทั้ง 3 ข้อข้างต้น มาทำเป็น Architecture ของ Inception-v3

- Resnet50

ResNet (Deep Residual Network) จากงานวิจัย Deep Residual Learning for Image Recognition ที่เสนอวิธีแก้ปัญหา Vanishing gradient ซึ่งเกิดกับ convolutional neural network ที่มีความลึกมาก โดยการใส่ shortcut เข้าไปช่วยเป็นเทคนิคที่เรียกว่า "residual mapping"

Vanishing Gradient

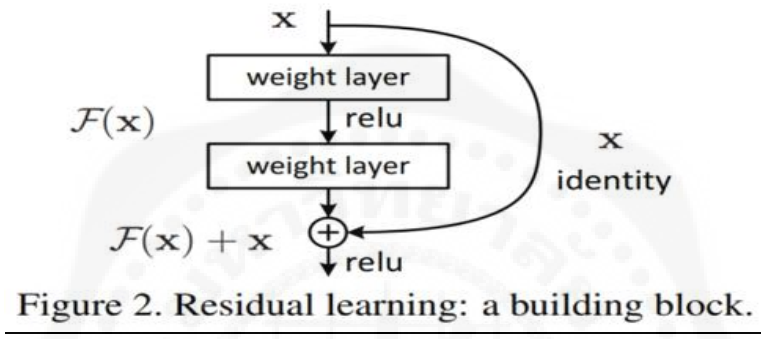


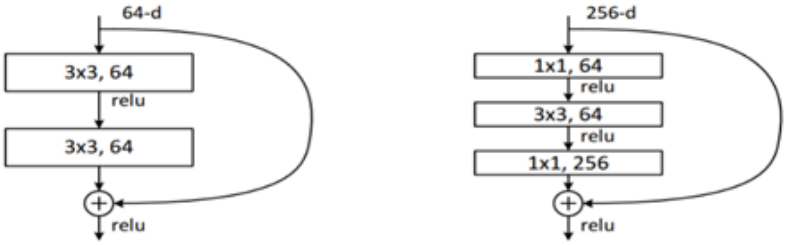
Figure 2. Residual learning: a building block.

รูปภาพ 41 แสดงหลักการของการทำ building block

ที่มา : [ออนไลน์] https://laptrinhx.com/ma-thakhwam-rucak-resnet-kan-di-kwa-334005315/

ให้ผลลัพธ์จาก convolution layer อยู่ที่ตำแหน่ง arrays x และมี convolution layer เพิ่มอีก 2 layer ซึ่งถ้าให้ผลลัพธ์ x ผ่าน layer ถัดมา อาจจะทำให้ผลลัพธ์คลาดเคลื่อนไปจากแบบเดิม ทำให้ไม่สามารถหาได้ว่า parameter ใน layer แรกให้ค่าอะไรเปลี่ยนแปลงไปบ้าง

จึงแก้ปัญหาโดยการส่งค่า x ลัดไปบวกกับผลลัพธ์ใน layer สุดท้ายแทน



รูปภาพ 42 แสดงการเปรียบเทียบระหว่าง Features map ที่มีขนาดเท่ากัน (ภาพด้านซ้าย) และ Features map ที่มีขนาดต่างกัน

ที่มา : [ออนไลน์] https://laptrinhx.com/ma-thakhwam-rucak-resnet-kan-di-kwa-334005315/

โดยหาก Layer สุดท้ายมี Features map ขนาดที่ต่างออกไป (ตามภาพด้านขวา) ที่เปลี่ยนจาก Features map ขนาด 256 ไปเป็นขนาด 64 ก่อนที่จะนำผลลัพธ์จาก layer สุดท้ายไปบวกกับตำแหน่ง x จำเป็นต้องใช้ Feature map ขนาด 1x1 โดยเรียกการปรับ Feature map แบบนี้ว่า bottleneck building block

ResNet50 Architecture มีองค์ประกอบดังต่อไปนี้

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

รูปภาพ 43 แสดง ResNet50 Architecture

ที่มา : [ออนไลน์] https://iq.opengenus.org/resnet50-architecture/#:~:text=ResNet50%20is%20a%20variant%20of,explored%20ResNet50%20architecture%20in%20depth

- จาก Output size ส่งไปที่ Convolution layer ที่มี kernel size 7×7 และ 64 kernels ที่ต่างกัน stride ไปทีละ 2 step ได้ 1 layer ออกมา max pooling 3×3 ด้วย stride 2

- Convolution layer ถัดไป 1×1 และ 64 kernels หลังจากนั้นทำ 3×3 และ 64 kernels และสุดท้ายนำมารวมกับ 1×1 และ 256 kernels ดังนั้น layer นี้จะถูกทำซ้ำรวม 3 ครั้ง จึงได้ออกมา 9 layers

- Convolution layer ถัดไป 1×1 และ 128 kernels หลังจากนั้นทำ 3×3 และ 128 kernels และสุดท้ายนำมารวมกับ 1×1 และ 512 kernels ดังนั้น layer นี้จะถูกทำซ้ำรวม 4 ครั้ง จึงได้ออกมา 12 layers

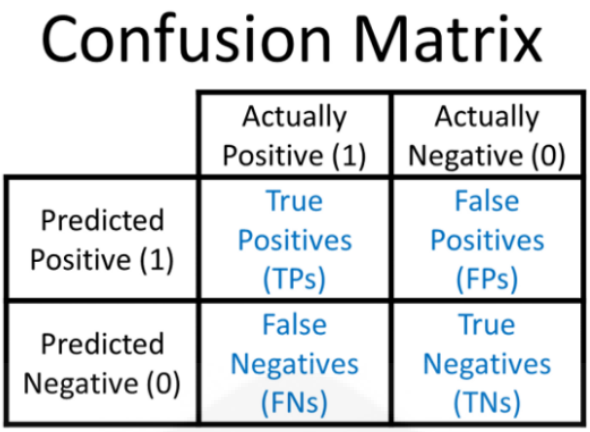
- Convolution layer ถัดไป 1×1 และ 256 kernels หลังจากนั้นทำ 3×3 และ 256 kernels และสุดท้ายนำมารวมกับ 1×1 และ 1024 kernels ดังนั้น layer นี้จะถูกทำซ้ำรวม 6 ครั้ง จึงได้ออกมา 18 layers

- Convolution layer ถัดไป 1×1 และ 512 kernels หลังจากนั้นทำ 3×3 และ 512 kernels และสุดท้ายนำมารวมกับ 1×1 และ 1048 kernels ดังนั้น layer นี้จะถูกทำซ้ำรวม 3 ครั้ง จึงได้ออกมา 9 layers

- จากนั้น เราจะทำ average pool ด้วย activation sotfmax และทำ fully connected layer ที่มีทั้งหมด 1000 nodes ด้วย activation sotfmax จะได้ 1 layers ออกมา

- โดยเราจะได้ Deep Convolutional network รวมทั้งหมด 1 + 9 + 12 + 18 + 9 + 1 คือ 50 layers

Confusion Matrix



รูปภาพ 44 แสดงConfusion Matrix

ที่มา : [ออนไลน์] https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

- True Positive (TP) หมายถึงสิ่งที่ทำนายตรงกับสิ่งที่เกิดขึ้นจริง ในกรณีทำนายว่า จริง สิ่งที่เกิดขึ้นก็คือจริง
- True Negative (TN) หมายถึงสิ่งที่ทำนายตรงกับสิ่งที่เกิดขึ้น ในกรณีทำนายว่า ไม่จริง สิ่งที่เกิดขึ้น ก็คือไม่จริง
- False Positive (FP) หมายถึงสิ่งที่ทำนายไม่ตรงกับสิ่งที่เกิดขึ้น คือทำนายว่า จริง แต่สิ่งที่เกิดขึ้น คือไม่จริง
- False Negative (FN) หมายถึงสิ่งที่ทำนายไม่ตรงกับที่เกิดขึ้นจริง คือทำนายว่า ไม่จริง แต่สิ่งที่เกิดขึ้นคือจริง

ค่าในการวัดประสิทธิภาพ

1. Precision คือ ค่าความแม่นยำของการวัดได้เป็นค่าใดค่าหนึ่งซ้ำๆเสมอ เกิดจากการนำ ค่า TP มาเทียบกับ FP
2. Recall คือ ค่าความถูกต้องที่ค้นพบข้อมูลที่ถูกต้องจากจำนวนข้อมูลที่ถูกต้องทั้งหมด เกิดจากการนำค่า TP มาเทียบกับ FN

3. f1-score คือ เป็นค่าที่ได้จากการเอาค่า precision และ recall มาคำนวณรวมกัน F1 สร้างขึ้นมาเพื่อเป็น single metric ที่วัดความสามารถของโมเดล ไม่จำเป็นต้องเลือกระหว่าง precision, recall เพราะการทำ F1-score จะมีการเฉลี่ยค่าให้
4. support คือ ค่าสนับสนุน ความถี่ของจำนวนข้อมูลทั้งหมด
5. sensitivity คือ ค่าความไว ความแม่นยำที่สนใจในส่วนของเหตุการณ์จริงที่เกิดขึ้น ซึ่งเป็นสัดส่วนของผลบวกที่เป็นจริง สำหรับภาวะนั้น ๆ เช่น สัดส่วนของการตรวจพบโรคในผู้ที่ป่วยจริง
6. specificity คือ ค่าความจำเพาะ เป็นสัดส่วนของผลลบที่เป็นจริงสำหรับภาวะนั้น ๆ เช่น สัดส่วนของการตรวจไม่พบโรคในผู้ที่ไม่ป่วย

2.2 งานวิจัยที่เกี่ยวข้อง

2.3.1) โรคต้อหินชนิดความดันตาปกติ (Normal Tension Glaucoma) โดย ฉัตรมงคล พรวนเจริญ

บทความนี้เกี่ยวกับอาการของโรคต้อหิน ซึ่งเป็นเกี่ยวกับระบบประสาทภายในดวงตา โดยจะระบุถึงสาเหตุการเกิดโรค ลักษณะของอาการ และการรักษาเป็นหลัก ไม่ได้ลงไปถึงหลักการเขียนโปรแกรม แต่ทำให้เราสามารถรู้จักลักษณะของภาพดวงตาเมื่อตรวจจับแล้วพบก้อนต้อหินได้

ข้อดี : ทำให้ได้เรียนรู้เกี่ยวกับอาการของการเกิดโรคนี้ ว่าเกิดในลักษณะรูปแบบใดภายในดวงตา ทำให้สามารถศึกษามาต่อยอดในโครงงานได้

ข้อเสีย : มีรูปภาพประกอบน้อย ส่วนมากเป็นทฤษฎีและหลักการทางการแพทย์ ทำให้ศัพท์ยากต่อการเข้าใจได้ทันที ต้องใช้เวลาในการทำความเข้าใจบทความ

2.3.2) Simple Image Classification with CNN โดย Shraddha Anala

เป็นบทความการสร้าง Convolutional Neural Network (CNN) เพื่อทำ Image Classification

ข้อดี : เป็นตัวอย่าง Step ในการทำ Image Classification ด้วยการใช้ CNN

2.3.3) Develop a Deep Convolutional Neural Network Step-by-Step to Classify Photographs of Dogs and Cats โดย Jason Brownlee

เป็นการสร้าง model ของ Deep Convolutional Neural Network ในการ Classify รูปสุนัขและแมว โดยมี accuracy สูงถึง 97%

ข้อดี : โมเดลมีความแม่นยำสูง

ข้อเสีย : รูปที่ใช้จะต้องเป็นภาพสี และมีความชัดเจนในการแยก Classify ในระดับนึง

2.3.4) The 4 Convolutional Neural Network Models That Can Classify Your Fashion Images โดย James Le

เป็นการสร้าง machine learning model สำหรับการจดจำภาพของ fashion objects โดยใช้ชุดข้อมูล Fashion-MNIST โดยแบ่งหมวดหมู่ของเสื้อผ้า ด้วยลักษณะที่มองเห็นของแต่ละคลาส

ข้อดี : มีการแบ่งคลาสที่ชัดเจน มีการทดสอบเปรียบเทียบกับ VGG-19

บทที่3

วิธีการดำเนินโครงงาน

3.1 วิธีการดำเนินงาน

3.1.1 เริ่มต้นและวางแผนโครงการ

3.1.1.1 เลือกหัวข้อโครงงาน

3.1.1.2 หาข้อมูลศึกษาแนวคิดและทฤษฎี

3.1.1.3 ศึกษาเทคโนโลยีที่จะนำมาใช้

3.1.2 การออกแบบระบบ

3.1.2.1 กำหนดวัตถุประสงค์และขอบเขตของระบบ

3.1.3 การเตรียมข้อมูลรูปภาพ

3.1.3.1 นำรูปภาพดวงตาที่ได้มาจากโรงพยาบาล มาทำการศึกษา และประมวลผล

3.1.3.2 ทำ Object Detection และปรับขนาดรูปภาพ กับฐานข้อมูลรูปภาพที่มี

3.1.3.3 ตรวจสอบความพร้อมของฐานข้อมูล

3.1.4 การพัฒนาโมเดลสำหรับการวิเคราะห์รูปภาพดวงตา

3.1.4.1 ทำการเทรนและพัฒนาโมเดล

3.1.4.2 ทำการทดสอบและปรับปรุงโมเดล

3.1.5 สรุปและเผยแพร่งานวิจัย

3.1.5.1 ทำการปรับปรุงระบบต่าง ๆ

3.1.5.2 สรุปผลงาน เพื่อนำเสนอ

3.2 แผนการดำเนินงานตลอดโครงงาน

แผนการดำเนินงานโครงงานวิจัย						
ขั้นตอนการดำเนินงาน	**ก.ค. 63**	**ส.ค 63**	**ก.ย. 63**	**ต.ค. 63**	**พ.ย. 63**	**ธ.ค. 63**
1.เริ่มต้นและวางแผนโครงการ						
เลือกหัวข้อโครงการ						
หาข้อมูลศึกษาแนวคิดและ ทฤษฎี						
ศึกษาเทคโนโลยีที่จำเป็นมาใช้						
2. รวบรวมข้อมูล						
รวบรวมบทความนำมาศึกษา						
3.สร้าง CNN ตรวจจับวัตถุ						
ทดสอบ Demo ในการตรวจจับวัตถุ						

4. รวบรวม Data Set						
รวบรวม Data Set ที่เป็นการตรวจจับสิ่งของ						
รวบรวม Data Set ที่เป็นรูปดวงตาจากทางการแพทย์						
5.สร้าง Model ด้วย CNN ในการตรวจจับภาพดวงตา						
ทดสอบ Model ด้วย CNN ในการตรวจจับภาพดวงตา						
6. ทำการทดสอบและปรับปรุง						
ปรับปรุงให้มีความแม่นยำมากยิ่งขึ้น						
7. สรุปและเผยแพร่งานวิจัย						
สรุปผลและนำเสนอ						

ตาราง 1 แผนการดำเนินโครงการวิจัย

3.3 อุปกรณ์และเครื่องมือที่ใช้

3.3.1 อุปกรณ์

3.3.1.1Computer

3.3.2ซอฟแวร์

3.3.2.1 Google Colaboratory

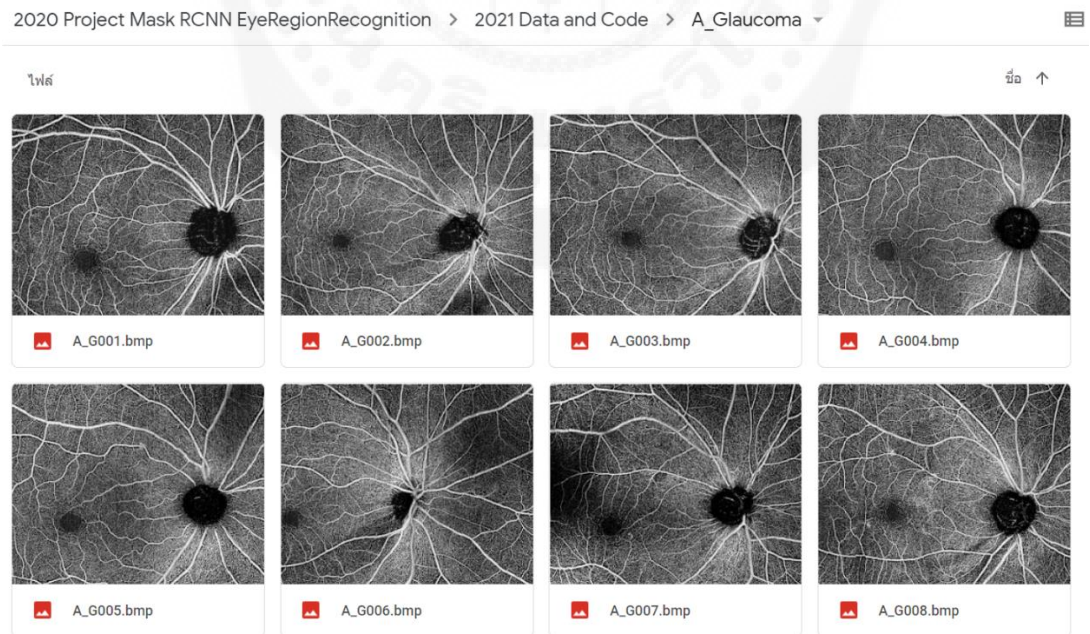
3.3.3ภาษาที่ใช้

3.3.3.1Python

3.4 ชุดข้อมูล (Dataset Set)

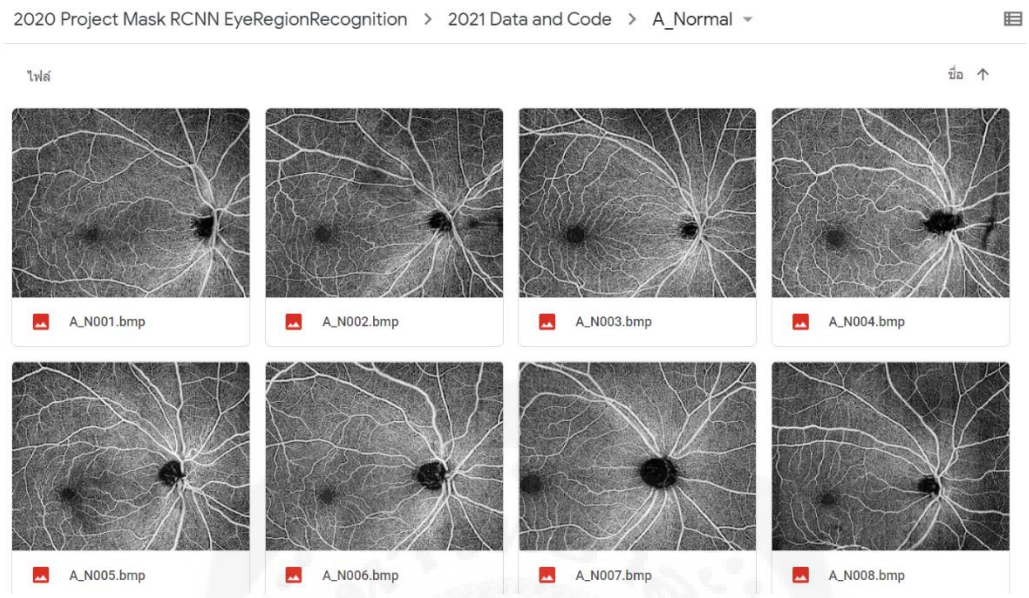
ชุดข้อมูลที่ใช้ทดสอบในการเทรนข้อมูล CNN แบ่งเป็น 2 โฟลเดอร์ คือ

1.) Glaucoma เป็นชุดข้อมูลที่แสดงรูปภาพของดวงตาที่มีอาการต้อหิน จำนวน 281 รูป



รูปภาพ 45 แสดงภาพดวงตาที่มีอาการต้อหิน (Glaucoma)

2.) Normal เป็นชุดข้อมูลที่แสดงรูปภาพดวงตาปกติ จำนวน 132 รูป



รูปภาพ 46 แสดงภาพดวงตาที่มีอาการปกติ (Normal)

3.5 ปัญหาและอุปสรรค

3.5.1 เนื่องจากต้องมาศึกษาหลักการในการทำ CNN ทำให้ต้องใช้ระยะเวลาในการศึกษาข้อมูลในการทำโครงงานนี้

3.5.2 จากชุดข้อมูลรูปภาพที่พบปัญหาในตอนแรก ทำให้ต้องเปลี่ยนวิธีมาใช้ CNN Model

บทที่ 4

ผลการดำเนินโครงงาน

4.1 ผลลัพธ์ของ CNN

โครงงานนี้ได้นำเทคนิคการวิเคราะห์รูปภาพมาใช้ในการวิเคราะห์หารูปภาพของดวงตาที่มีความผิดปกติด้วยการแบ่งรูปภาพเป็นขนาดย่อย ๆ โดยใช้หลักการ Convolutional Neural Network (CNN) มาใช้ในการวิเคราะห์

ชุดข้อมูลที่ใช้ในการดำเนินงานมีทั้งหมด 413 รูป โดยที่แต่ละรูปมีขนาด 834 x 500 pixel แบ่งเป็น 2 Class เพื่อใช้ในการทดสอบ คือ

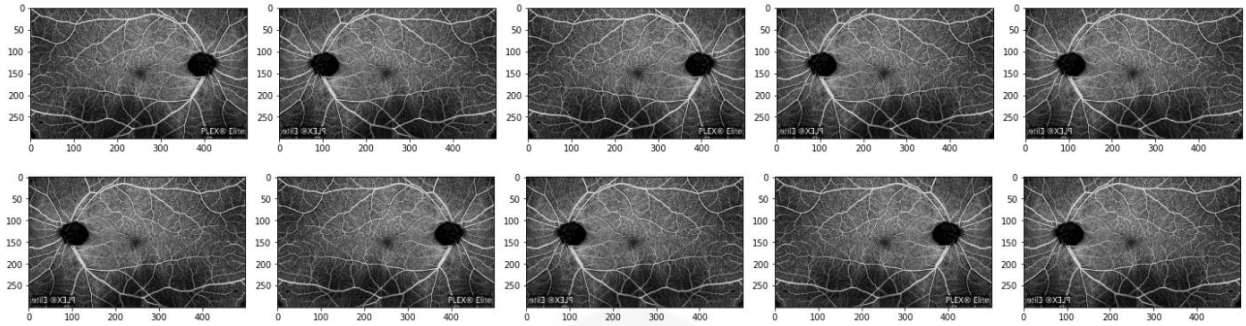
```
Train Data
Found 331 images belonging to 2 classes.
Test Data
Found 82 images belonging to 2 classes.
```

รูปภาพ 47 แสดงจำนวนรูปภาพ และ Class ทั้งหมดที่ใช้ในการเทรนข้อมูล

1.) Train Data จำนวน 331 รูป

2.) Test Data จำนวน 82 รูป

เนื่องจากชุดข้อมูลที่มีจำนวนน้อยต่อการเทรนข้อมูล จึงได้มีการทำ Data Augmentation เพื่อลดการเกิด Overfitting ในการเทรนข้อมูล



รูปภาพ 48 แสดงการทำ Argumentation

```
{'A_Glaucoma': 0, 'A_Normal': 1}
[[0.]]
GLACOMA
{'A_Glaucoma': 0, 'A_Normal': 1}
[[1.]]
NORMAL
```

รูปภาพ 49 แสดงผลลัพธ์ด้วยการทดสอบจาก path file

จากการทดสอบ path file ในการเก็บข้อมูล โดย path แรกเก็บข้อมูลประเภท Glaucoma และ path ต่อมาเก็บข้อมูล Normal ผลลัพธ์ที่ได้จากการทดสอบมีคำตอบที่ถูกต้อง

ผลลัพธ์ที่ได้จากการเทรนข้อมูลมีค่า accuracy อยู่ที่ 0.81 คิดเป็น 81%

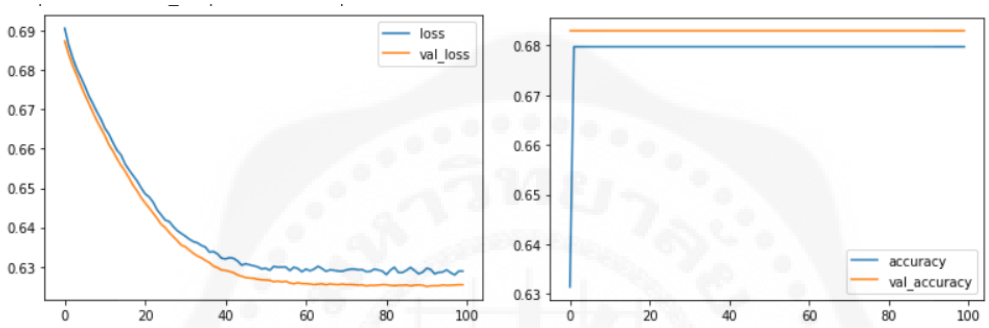
```
1/1 [==============================] - 1s 1s/step - loss: 0.6015 - accuracy: 0.8171

Model's Evaluation Metrics:
---------------------------
Accuracy: 0.8170731663703918
Loss: 0.601502537727356
```

รูปภาพ 50 แสดงผลลัพธ์ของค่า accuray and loss

4.2 ผลลัพธ์ของ Network model ประเภทอื่นๆ

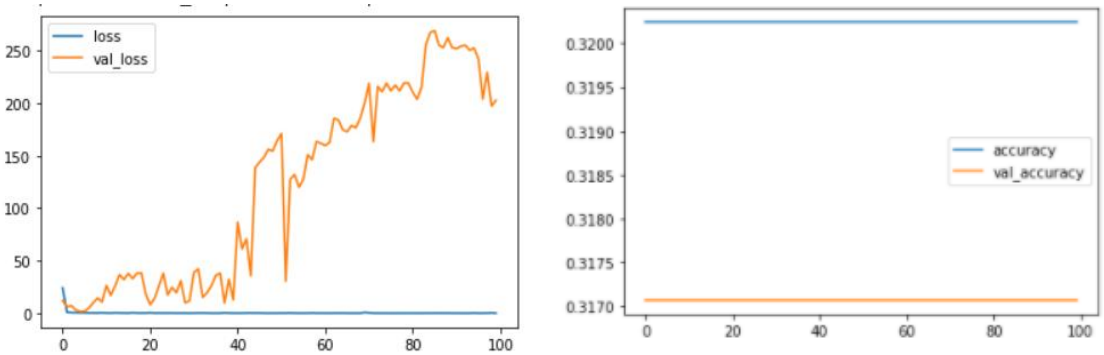
4.2.1) VGG 16 model : accuracy = 0.33 และ loss = 0.34

```
4/4 [==============================] - 7s 2s/step - loss: 0.2730 - accuracy: 0.2880 - val_loss: 0.7376 - val_accuracy: 0.3171
Epoch 98/100
4/4 [==============================] - 8s 2s/step - loss: 0.2876 - accuracy: 0.3005 - val_loss: 0.8663 - val_accuracy: 0.3171
Epoch 99/100
4/4 [==============================] - 8s 2s/step - loss: 0.3803 - accuracy: 0.3515 - val_loss: 0.8101 - val_accuracy: 0.3171
Epoch 100/100
4/4 [==============================] - 7s 2s/step - loss: 0.3488 - accuracy: 0.3361 - val_loss: 0.6177 - val_accuracy: 0.3171
```



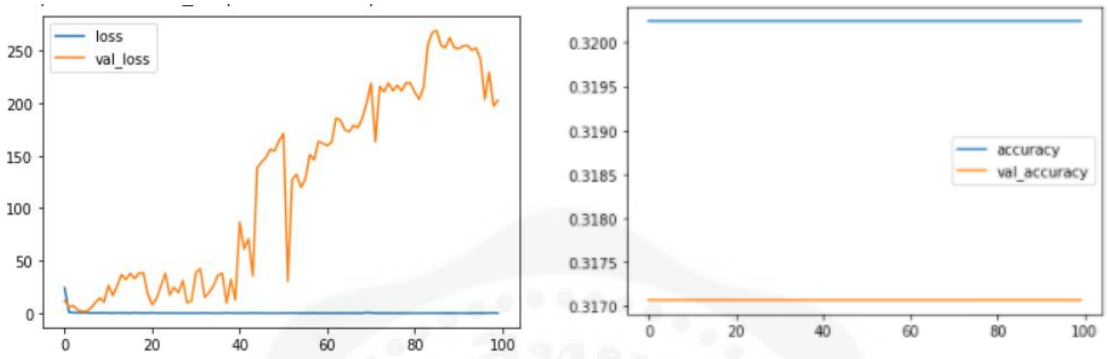
4.2.2) MobileNetV2 : accuracy = 0.32 และ loss = 0.04

	loss	accuracy	val_loss	val_accuracy
0	23.934759	0.320242	12.023609	0.317073
1	0.871383	0.320242	5.854974	0.317073
2	0.559165	0.320242	7.100817	0.317073
3	0.503382	0.320242	3.069846	0.317073
4	0.416995	0.320242	1.604718	0.317073
..	...	...	...	...
95	0.023450	0.320242	242.537109	0.317073
96	0.020584	0.320242	204.158676	0.317073
97	0.104406	0.320242	229.510452	0.317073
98	0.324777	0.320242	197.452988	0.317073
99	0.041260	0.320242	202.739334	0.317073



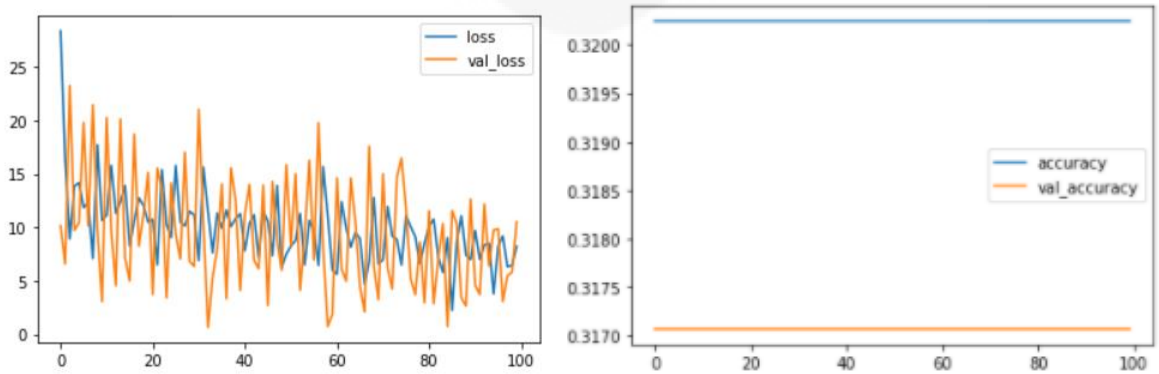
4.2.3) MobileNet Ex2 : accuracy = 0.30 และ loss = 0.27

```
4/4 [==============================] - 7s 2s/step - loss: 0.2358 - accuracy: 0.3258 - val_loss: 0.8574 - val_accuracy: 0.3171
Epoch 98/100
4/4 [==============================] - 7s 2s/step - loss: 0.2263 - accuracy: 0.3292 - val_loss: 0.7407 - val_accuracy: 0.3171
Epoch 99/100
4/4 [==============================] - 7s 2s/step - loss: 0.1672 - accuracy: 0.3117 - val_loss: 0.8180 - val_accuracy: 0.3171
Epoch 100/100
4/4 [==============================] - 7s 2s/step - loss: 0.2754 - accuracy: 0.3094 - val_loss: 0.9687 - val_accuracy: 0.3171
```



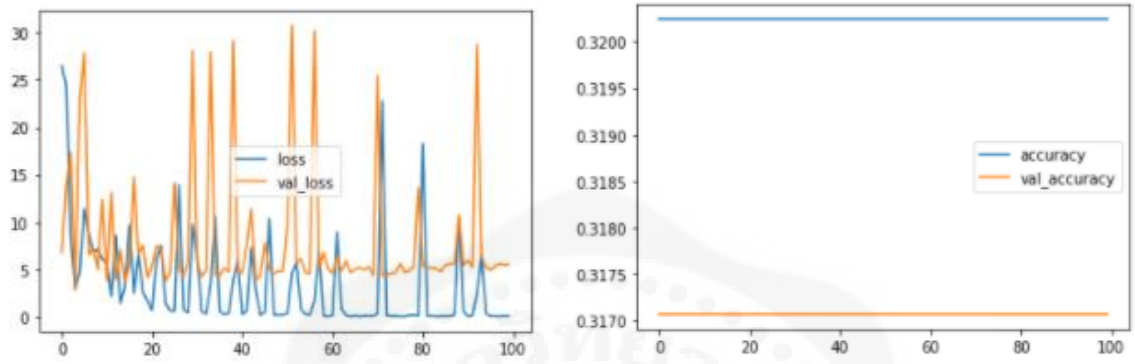
4.2.4) EfficientNet B0 : accuracy = 0.30 และ loss = 8.4

	loss	accuracy	val_loss	val_accuracy
0	28.386335	0.320242	10.148320	0.317073
1	16.323578	0.320242	6.577157	0.317073
2	8.933468	0.320242	23.257057	0.317073
3	13.845317	0.320242	9.729277	0.317073
4	14.183225	0.320242	10.431584	0.317073
..	...	...	...	...
95	8.255069	0.320242	9.864823	0.317073
96	9.186574	0.320242	3.056899	0.317073
97	6.310441	0.320242	5.458708	0.317073
98	6.448401	0.320242	5.804930	0.317073
99	8.204464	0.320242	10.521410	0.317073



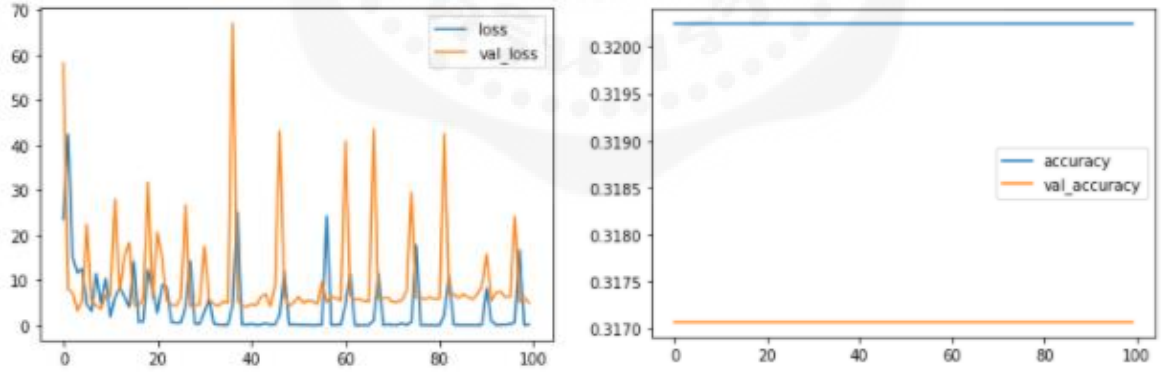
4.2.5) Xception : accuracy = 0.29 และ loss = 0.09

```
Epoch 97/100
4/4 [==============================] - 7s 2s/step - loss: 0.0114 - accuracy: 0.3308 - val_loss: 2.4863 - val_accuracy: 0.3171
Epoch 98/100
4/4 [==============================] - 7s 2s/step - loss: 0.0477 - accuracy: 0.3112 - val_loss: 2.2296 - val_accuracy: 0.3171
Epoch 99/100
4/4 [==============================] - 7s 2s/step - loss: 0.0386 - accuracy: 0.3081 - val_loss: 2.2977 - val_accuracy: 0.3171
Epoch 100/100
4/4 [==============================] - 7s 2s/step - loss: 0.0966 - accuracy: 0.2946 - val_loss: 9.7655 - val_accuracy: 0.3171
```



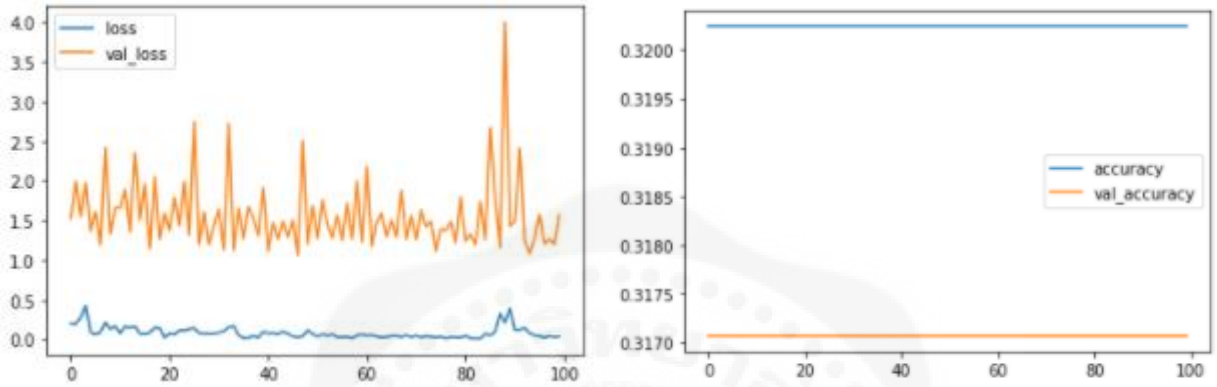
4.2.6) Inception : accuracy = 0.33 และ loss = 0.01

```
Epoch 97/100
4/4 [==============================] - 5s 1s/step - loss: 0.0748 - accuracy: 0.3282 - val_loss: 10.4214 - val_accuracy: 0.3171
Epoch 98/100
4/4 [==============================] - 4s 965ms/step - loss: 0.0390 - accuracy: 0.3301 - val_loss: 9.0770 - val_accuracy: 0.3171
Epoch 99/100
4/4 [==============================] - 5s 1s/step - loss: 0.1224 - accuracy: 0.2999 - val_loss: 7.2059 - val_accuracy: 0.3171
Epoch 100/100
4/4 [==============================] - 4s 944ms/step - loss: 0.0164 - accuracy: 0.3338 - val_loss: 13.3512 - val_accuracy: 0.3171
```



4.2.7) Restnet : accuracy = 0.32 และ loss = 0.04

```
Epoch 97/100
4/4 [==============================] - 5s 2s/step - loss: 0.0527 - accuracy: 0.3222 - val_loss: 1.4258 - val_accuracy: 0.3171
Epoch 98/100
4/4 [==============================] - 5s 2s/step - loss: 0.0449 - accuracy: 0.3279 - val_loss: 1.5955 - val_accuracy: 0.3171
Epoch 99/100
4/4 [==============================] - 6s 1s/step - loss: 0.0554 - accuracy: 0.3333 - val_loss: 1.1218 - val_accuracy: 0.3171
Epoch 100/100
4/4 [==============================] - 6s 1s/step - loss: 0.0452 - accuracy: 0.3219 - val_loss: 1.5810 - val_accuracy: 0.3171
```



บทที่ 5

สรุปผล อภิปรายผล และข้อเสนอแนะ

5.1 สรุปผลการศึกษาค้นคว้า

จากการศึกษาและพัฒนาระบบตรวจสอบดวงตาด้วยโครงข่ายประสาทเทียมแบบคอลโวลูชัน (Glaucoma Classification by using Convolutional Neural Network) สามารถสรุปผลลัพธ์ได้ดังต่อไปนี้ ผู้วิจัยได้ทำการศึกษาและพัฒนาระบบตรวจสอบดวงตาด้วยการนำหลักการโครงข่ายประสาทเทียมแบบคอลโวลูชัน (Convolutional Neural Network) ที่ใช้หลักการทางคณิตศาสตร์ มาจัดการกับข้อมูล โดยใช้ขั้นตอนหลัก 5 ขั้นตอน คือ Convolution, Activation, Pooling, Flatten และ Fully Connected จากนั้นนำมาสร้างเป็นโมเดลที่มีความแม่นยำสูงสุดถึงร้อยละ 0.81 แสดงให้เห็นว่าการออกแบบโมเดลโดยใช้หลักการCNN (Convolutional Neural Network) นั้นเป็นแนวทางที่ดีในการนำมาศึกษาต่อยอด และพัฒนาต่อไปในอนาคต

โดยทางผู้จัดทำได้นำโมเดลที่ออกแบบด้วยหลักการ CNN (Convolutional Neural Network) มาเปรียบเทียบกับ Network Architecture Model อื่นๆอีก 6 รูปแบบ เพื่อทดสอบว่าโมเดลที่ได้จัดทำมีประสิทธิภาพดีหรือแย่กว่าโมเดลอื่นๆ

5.2 อภิปรายผล

จากการดำเนินโครงงานระบบตรวจสอบดวงตาด้วยการนำหลักการโครงข่ายประสาทเทียมแบบคอลโวลูชัน (Glaucoma Classification by using Convolutional Neural Network) โดยผู้จัดทำได้ทำการพัฒนาโมเดลด้วยหลักการ Convolution Operation ที่เรียกว่า CNN (Convolutional Neural Network) พบว่าเป็นโมเดลที่มีประสิทธิภาพในการตรวจจับภาพดวงตาสูงกว่า Network Model อื่นๆที่นำมาเปรียบเทียบ ได้แก่ VGG16 , MobileNetV2 , EfficientNet B0 , Xception , Inception , Restet เนื่องจากโมเดลที่พัฒนาด้วยหลักการ CNN (Convolutional Neural Network) นั้นได้มีการปรับปรุงค่าโมเดลให้รองรับในการตรวจสอบภาพ OCT ทางการแพทย์ของดวงตาโดยเฉพาะ พร้อมทั้งลดการเกิดข้อผิดพลาด (Errors) ในการเรียนรู้ข้อมูลด้วยหลักการ Data

Augmentation ซึ่งเป็นการเพิ่มจำนวนรูปในการเทรนข้อมูลด้วยการปรับรูปภาพ เช่น แปลงค่าพิกเซล , หมุนภาพ, พลิกภาพ, บิดภาพแบบสุ่ม,ขยายภาพแบบสุ่ม, ปรับแสงของภาพ เป็นต้น และ Validation ที่นำมาใช้ทดสอบว่าโมเดลทำงานได้ดีหรือไม่

5.3ข้อเสนอแนะ

เนื่องจากโครงงานนี้ยังมีจุดบกพร่องอยู่ อาทิเช่น ชุดข้อมูลที่ได้รับมีจำนวนที่ไม่มาก ทำให้ประสิทธิภาพของการตรวจสอบภาพดวงตานั้นยังไม่สมบูรณ์มากเพียงพอต่อการนำไปใช้งานจริง ซึ่งหากได้รับชุดข้อมูลที่มากเพียงพอต่อการเรียนรู้ของโมเดล จะช่วยเพิ่มประสิทธิภาพของผลลัพธ์ในการตรวจสอบภาพดวงตาได้ดีขึ้น อีกทั้งยังสามารถนำไปต่อยอดในการพัฒนาโมเดลให้สามารถใช้งานได้จริงในทางการแพทย์ เพื่อช่วยลดทั้งเวลาและเพิ่มประสิทธิภาพในการวินิจฉัยของแพทย์ได้อีกด้วย

บรรณานุกรม

[1] Activation Function คืออะไร ใน Artificial Neural Network, Sigmoid Function คืออะไร – Activation Function ep.1 - BUA Labs. (n.d.). Retrieved November 29, 2020, from https://www.bualabs.com/archives/1261/what-is-activation-function-what-is-sigmoid-function-activation-function-ep-1/

[2] Applied Deep Learning - Part 4: Convolutional Neural Networks | by Arden Dertat | Towards Data Science. (n.d.). Retrieved November 29, 2020, from https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

[3] Convolutional Neural Network (CNN) คืออะไร | by Natthawat Phongchit | Medium. (n.d.). Retrieved November 29, 2020, from https://medium.com/@natthawatphongchit/มาลองดูวิธีการคิดของ-cnn-กัน-e3f5d73eebaa

[4] Deep Learning แบบฉบับสามัญชน EP 2 Optimization & Activation Function เรียนกันสบายๆ สไตส์ชิลๆ | by Mr.P L | mmp-li | Medium. (n.d.). Retrieved November 29, 2020, from https://medium.com/mmp-li/deep-learning-แบบฉบับสามัญชน-ep-2-optimization-activation-function-xเรียนกันสบายๆสไตส์ชิลๆ-9feb5a87e3b2

[5] Garway-Heath, D. F., Poinoosawmy, D., Fitzke, F. W., & Hitchings, R. A. (2000). Mapping the visual field to the optic disc in normal tension glaucoma eyes. Ophthalmology, 107(10), 1809–1815. https://doi.org/10.1016/S0161-6420(00)00284-0

[6] Layer activation functions. (n.d.). Retrieved November 29, 2020, from https://keras.io/api/layers/activations/

[7] Park, K., Kim, J., & Lee, J. (2019). Visual Field Prediction using Recurrent Neural Network. Scientific Reports, 9(1), 1–12. https://doi.org/10.1038/s41598-019-44852-6

[8] Project Jupyter | Home. (n.d.). Retrieved June 27, 2020, from https://jupyter.org/

[9] Region of Interest Pooling - Towards Data Science. (n.d.). Retrieved June 27,2020, from https://towardsdatascience.com/region-of-interest-pooling-f7c637f409af

[10] What is CNN? (n.d.). Retrieved November 29, 2020, from https://analazia.wordpress.com/2019/04/22/what-is-cnn/

[11] เริ่มต้น Deep Learning ด้วย Keras | by NakarinSTK | Medium. (n.d.). Retrieved November 29, 2020, from https://medium.com/@NakarinSTK/เริ่มต้น-deep-learning-ด้วย-keras-b13edc47b1b3