# การใช้โครงข่ายประสาทคอนโวลูชันในการคัดแยกภาพเอกซเรย์เพื่อตรวจหาโรคโควิด
# X-Ray image classification for detection of Corona virus disease by using convolution neural network

นายพัชรพล ลีสุขสาม
รหัส 620510514

ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์
มหาวิทยาลัยเชียงใหม่
ภาคการศึกษาที่ 2 ปีการศึกษา 2565

การใช้โครงข่ายประสาทคอนโวลูชันในการคัดแยกภาพเอกซเรย์เพื่อตรวจหาโรคโควิด
X-Ray image classification for detection of Corona virus disease by using convolution neural network

พัชรพล ลีสุขสาม

ได้รับการพิจารณาอนุมัติให้เป็นส่วนหนึ่งของการศึกษา
ตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต
สาขาคณิตศาสตร์

คณะกรรมการควบคุมการค้นคว้าอิสระ:

สุทธิดา วงศ์แก้ว ..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. สุทธิดา วงศ์แก้ว)

หทัยรัตน์ ยิ่งทวีสิทธิกุล ..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.หทัยรัตน์ ยิ่งทวีสิทธิกุล)

วันที่ 24 เดือน มีนาคม พ.ศ. 2566

# การใช้โครงข่ายประสาทคอนโวลูชันในการคัดแยกภาพเอกซเรย์เพื่อตรวจหาโรคโควิด
# X-Ray image classification for detection of Corona virus disease by using convolution neural network

นายพัชรพล ลีสุขสาม
รหัส 620510514

งานค้นคว้าอิสระนี้เป็นส่วนหนึ่งของการศึกษา
ตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต
ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่

# กิตติกรรมประกาศ

งานค้นคว้าอิสระฉบับนี้สำเร็จลุล่วงได้ด้วยความกรุณาจากผู้ช่วยศาสตราจารย์ ดร. สุทธิดา วงศ์แก้วอาจารย์ที่ปรึกษางานค้นคว้าอิสระนี้ที่ได้ให้คำแนะนำ ให้การปรึกษา และแนวทางในการแก้ไขข้อบกพร่องต่างๆในการเขียนรูปเล่ม และอาจารย์ ดร. เอกชัย ทวินันท์ ที่คอยให้คำแนะนำ ให้การปรึกษา และความรู้เกี่ยวกับเรื่องโครงข่ายประสาทเทียม ผู้เรียนจึงขอกราบขอบพระคุณเป็นอย่างสูง

ท้ายที่สุดผู้ศึกษาขอกราบขอบพระคุณผู้ที่มีส่วนเกี่ยวข้องที่คอยส่งเสริมและสนับสนุนด้านองค์ความรู้และทรัพยากรต่างๆแก่ผู้ศึกษามาโดยตลอด

พัชรพล ลีสุขสาม

| | |
|---|---|
| **หัวข้อ (ภาษาไทย)** | การใช้โครงข่ายประสาทคอนโวลูชันในการคัดแยกภาพเอกซเรย์เพื่อตรวจหาโรคโควิด |
| **(ภาษาอังกฤษ)** | X-Ray image classification for detection of Corona virus disease by using convolution neural network |
| **ชื่อผู้ทำการค้นคว้าอิสระ** | นายพัชรพล ลีสุขสาม **รหัสนักศึกษา** 620510514 |
| **ชื่ออาจารย์ที่ปรึกษา** | ผู้ช่วยศาสตราจารย์ ดร. สุทธิดา วงศ์แก้ว |
| **ชื่อกรรมการสอบ** | ผู้ช่วยศาสตราจารย์ ดร.หทัยรัตน์ ยิ่งทวีสิทธิกุล |


## บทคัดย่อ

การค้นคว้าอิสระนี้ได้สร้างโมเดลสำหรับการคัดแยกรูปภาพเอกซเรย์ของปอดว่าเป็นปอดอักเสบที่เกิดจากการติดโควิด (Covid-19 Pneumonia) หรือไม่ โดยใช้โครงข่ายประสาทเทียมซึ่งประกอบไปด้วยโครงข่ายประสาทคอนโวลูชันและโครงข่ายประสาทแบบป้อนไปหน้า โครงข่ายประสาทเทียมที่ศึกษานี้มีลักษณะโครงสร้างตาม Resnet50 และข้อมูลใช้ในการสอนโมเดลนำมาจากเว็บไซต์ kaggle ประกอบไปด้วยรูปภาพเอกซเรย์ปอดธรรมดา 3,000 รูป และเป็นรูปภาพเอกซเรย์คนไข้ที่มีเชื้อโควิดลงปอด 3,000 รูป

ในการศึกษานี้จะแบ่งข้อมูลดังกล่าวสำหรับสอนโมเดล validate โมเดล และวัดผลโมเดล ซึ่งแบ่งเป็น 60% 20% และ 20% ตามลำดับ ในการวัดผลจะใช้กราฟ ROC และ Confusion matrix ในการวัดผลบนข้อมูลชุดทดสอบ จากกราฟ ROC พบว่าพื้นที่ใต้กราฟมีค่าเท่ากับ 0.996 บ่งบอกว่าโมเดลนี้แยกข้อมูลปอดของผู้ป่วยเป็นปอดอักเสบที่เกิดจากการติดเชื้อโควิดและปอดของคนปกติได้ดีเยี่ยม สำหรับ Confusion matrix พบว่ามีความแม่นยำ 96.92% มีค่าความเที่ยง 96.87% และมีค่าการระลึกได้ได้ 96.70%

## Abstract

In this independent study, we construct an artificial neural network model for classifying X-Ray chest images, whether the image of Covid-19 patients or not. In particular, a feed-forward neural network and a convolutional neural network are the primary approaches for classifying. Our model is based on the Resnet50 architecture. The dataset is obtained from the Kaggle database. It contains 3,000 regular X-Ray chest images, and 3,000 Covid-19 Pneumonia X-Ray chest images are included.

The dataset is divided into a train set validation set and test set, which are 60% 20% and 20% of datasets, respectively. After training, the proposed model is evaluated using the ROC curve and confusion matrix. The ROC area under the curve is 0.996, suggesting that the model efficiently classifies X-ray images of normal and COVID-19 Pneumonia patients. At the confusion matrix, our model is shown 96.92 percent accuracy, 96.87 percent precision, and 96.70 percent recall.



**Keywords** : Feed forward neural network, Convolutional neural network, ROC curve, Confusion matrix

# สารบัญ

# บทที่ 1

# บทนำ (Introduction)

## 1.1 ที่มาและความสำคัญ

โรคโควิดเกิดจากเชื้อไวรัสโควิดที่เริ่มมีการแพร่ระบาดในปี 2019 ในประเทศจีน อาการของโรคนี้ผู้ป่วยจะมีอาการติดเชื้อที่ปอด (Covid Pneumonia) โดยหนึ่งในวิธีการที่สามารถตรวจหาโรคโควิดที่สามารถทำได้อย่างง่าย รวดเร็ว คือการทำเอกซเรย์ปอด และนำภาพที่ได้จากการเอกซเรย์มาวินิจฉัยในภายหลัง เนื่องจากการที่จะวินิจฉัยภาพเอกซเรย์ปอดว่าเป็น Covid pneumonia หรือไม่จำเป็นต้องใช้ผู้เชี่ยวชาญในการวินิจฉัย เป็นผลอาจทำให้เกิดความช้าในการวินิจฉัย เราจึงมีแนวคิดที่จะนำโครงข่ายประสาทแบบคอนโวลูชัน (Convolutional neural network) มาใช้ในการวินิจฉัยภาพเอกซเรย์ว่าผู้ป่วยเป็น Covid pneumonia หรือไม่ เพื่อที่จะสามารถลดเวลาในการวินิจฉัยลงได้

ในการศึกษานี้เราจะสร้างโครงข่ายประสาทคอนโวลูชันที่มีสถาปัตยกรรมแบบ ResNet50 ที่มีการสอนและวัดผลบนข้อมูลภาพเอกซเรย์ที่นำมาจากเว็บไซต์ Kaggle

## 1.2 วัตถุประสงค์

1. เพื่อศึกษาคณิตศาสตร์เบื้องหลังโครงข่ายประสาทแบบคอนโวลูชัน
2. เพื่อฝึกการใช้โปรแกรม Python ในการสร้างโครงข่ายประสาทคอนโวลูชันผ่านทาง Google Colab
3. เพื่อนำความรู้ทางโครงข่ายประสาทคอนโวลูชันมาสร้างโมเดลที่สามารถตรวจสอบภาพเอกสเรย์ปอดว่าผู้ป่วยเป็น Covid pneumonia หรือไม่

## 1.3 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้รู้ที่มาของคณิตศาสตร์เบื้องหลังโครงข่ายประสาทแบบคอนโวลูชัน
2. ได้ฝึกการใช้โปรแกรม Python ในการสร้างโครงข่ายประสาทคอนโวลูชันผ่านทาง Google Colab
3. ได้นำความรู้ทางโครงข่ายประสาทคอนโวลูชันมาสร้างโมเดลที่สามารถตรวจสอบภาพเอกสเรย์ปอดว่าผู้ป่วยเป็น Covid pneumonia หรือไม่

## 1.4 ขอบเขตและวิธีการดำเนินงาน

ใช้โปรแกรม Python ในการสร้างโครงข่ายประสาทคอนโวลูชันเพื่อแยกรูปภาพเอกซเรย์ปอดว่าเป็น Covid pneumonia หรือไม่ผ่านทาง Google Colab โดยนำข้อมูลจาก Kaggle มาใช้ในการสอนโมเดล

# บทที่ 2

# ความรู้พื้นฐาน (Preliminaries)

ในบทนี้จะกล่าวถึงความรู้พื้นฐานที่ใช้ในการค้นคว้าอิสระ โดยแบ่งเนื้อหาออกเป็น 4 ส่วน ได้แก่ ฟังก์ชันที่นิยมใช้ในโครงข่ายประสาทเทียม โครงข่ายประสาทแบบป้อนไปหน้า (Feed forward Neural Network) โครงข่ายประสาทคอนโวลูชัน (Convolutional neural network) และการวัดผลโมเดล โดยสามารถดูเนื้อหาเพิ่มเติมได้ที่ [1], [3], [4], [5], [8]

## 2.1 ฟังก์ชันที่นิยมใช้ในโครงข่ายประสาทเทียม

**บทนิยาม 2.1** จะเรียกฟังก์ชัน $f : \mathbb{R} \to \mathbb{R}$ ว่าเป็น**ฟังก์ชันเชิงเส้น (Linear function)** เมื่อ

$$f(x) = x \tag{2.1}$$

โดยเพอร์เซปตรอน (Perceptron) ในตัวอย่าง 2.8 ใช้ฟังก์ชันกระตุ้น (Activation function) เป็นฟังก์ชันเชิงเส้น

**บทนิยาม 2.2** จะเรียกฟังก์ชัน $f : \mathbb{R} \to \{0, 1\}$ ว่าเป็น**ฟังก์ชันฮาร์ดลิมิตแบบสมมาตร (Symmetric Hard-Limit)** เมื่อ

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.2}$$

โดยเพอร์เซปตรอนในตัวอย่าง 2.5 2.7 และข้อสังเกต 2.6 ใช้ฟังก์ชันกระตุ้นเป็นฟังก์ชันฮาร์ดลิมิตแบบสมมาตร

**<u>บทนิยาม 2.3</u>** จะเรียกฟังก์ชัน $f : \mathbb{R} \to [0, \infty)$ ว่าเป็น**ฟังก์ชัน ReLU (Rectified Linear Unit)** เมื่อ

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.3}$$

โดยโครงข่ายประสาทคอนโวลูชัน (Convolutional neural network) ในโมเดล Resnet50 3.1 ใช้ฟังก์ชันกระตุ้นเป็นฟังก์ชัน ReLU

**<u>บทนิยาม 2.4</u>** จะเรียกฟังก์ชัน $f : \mathbb{R} \to \mathbb{R}$ ว่าเป็น**ฟังก์ชัน Sigmoid** เมื่อ

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

โดยโครงข่ายประสาทแบบป้อนไปข้างหน้า (Feed forward neural network) ในโมเดล Resnet50 3.1 ใช้ฟังก์ชันกระตุ้นเป็นฟังก์ชัน Sigmoid

## 2.2 โครงข่ายประสาทแบบป้อนไปหน้า (Feed-forward Neural Network)

โครงข่ายประสาทแบบป้อนไปหน้าเป็นโครงข่ายประสาทเทียมอย่างง่ายที่สามารถเขียนให้อยู่ในรูปกราฟได้ โดยเรียกกราฟประเภทนี้ว่า กราฟโครงข่าย (Network Graph) ซึ่งประกอบไปด้วยปม (Node) ที่เป็นตัวแทนของค่านำเข้า (Input) หรือค่าส่งออก (Output) และเส้นเชื่อม (Edge) ซึ่งเป็นตัวแทนของน้ำหนัก(Weight) หรือไบแอส (Bias) ซึ่งกราฟดังกล่าวจะใช้ค่านำเข้า น้ำหนัก ไบแอส มาใช้ในการคำนวณหาค่าส่งออก

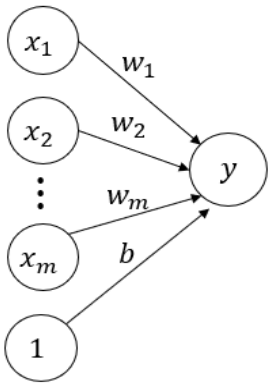
โดยในหัวข้อนี้จะมีการกล่าวถึงเพอร์เซปตรอน (Perceptron) เพอร์เซปตรอนหลายชั้น (Multi-layer perceptron) การวัดค่าความผิดพลาด ขั้นตอนวิธี Gradient descent การแพร่กลับ (Back propagation) และโครงข่าย Residual

### 2.2.1. เพอร์เซปตรอน (Perceptron)

ในกรณีนี้จะพิจารณาเพอร์เซปตรอนที่มีปมส่งออกหนึ่งปม กล่าวคือมีค่าส่งออกเพียงค่าเดียว ให้ $\mathbf{x} = (x_1, x_2, ..., x_m) \in \mathbb{R}^m$ เป็นค่านำเข้า $\mathbf{w} = (w_1, w_2, ..., w_m) \in \mathbb{R}^m$ เป็นน้ำหนัก $b \in \mathbb{R}$ เป็นไบแอส และ $f : \mathbb{R} \to \mathbb{R}$ เป็นฟังก์ชันกระตุ้น เราจะได้ว่าค่าส่งออก $y \in \mathbb{R}$ ที่เกิดจากเพอร์เซ็ปตรอนดังกล่าวสามารถคำนวณได้จาก

$$y = f(z) = f(\mathbf{w} \cdot \mathbf{x} + b) = f(\sum_{i=1}^{m} w_i x_i + b) \tag{2.5}$$

เพอร์เซปตรอนที่มีปมค่าส่งออกหนึ่งปมสามารถเขียนให้อยู่ในรูปกราฟได้ดังแสดงในรูปที่ 2.1




รูปที่ 2.1: กราฟเพอร์เซปตรอนที่มีปมค่าส่งออกหนึ่งปม

**ตัวอย่าง 2.5** สำหรับปัญหา OR ที่มีข้อมูลนำเข้า $\mathbf{x} = (x_1, x_2) \in \{0,1\}^2$ และข้อมูลเป้าหมาย $t \in \{0,1\}$ ดังตารางที่ 2.1

| $x_1$ | $x_2$ | $t$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

ตารางที่ 2.1: ปัญหา OR

เนื่องจากปัญหาดังกล่าวมีค่านำเข้าเป็น $\{(0,0),(0,1),(1,0),(1,1)\} \in \mathbb{R}^2$ และมีค่าเป้าหมายเป็น $\{0,1,1,1\} \in \mathbb{R}$ จะได้ว่าหากเราใช้เพอร์เซปตรอนหนึ่งตัวในการคัดแยกปัญหาดังกล่าว เพอร์เซปตรอนจะมีปมนำเข้า 2 ปม และปมส่งออก 1 ปม

ถ้ากำหนดค่าน้ำหนักเป็น $\mathbf{w} = (w_1, w_2) = (0.5, 0.5)$ ค่าไบแอสเป็น $b = -0.2$ และฟังก์ชันกระตุ้นเป็นฟังก์ชันฮาร์ดลิมิตแบบสมมาตร (Symmetric Hard-Limit) จะได้การคำนวณในเพอร์เซปตรอนเป็นไปตามสมการ

$$y = f(z) = f(\mathbf{w} \cdot \mathbf{x} + b) = f(0.5x_1 + 0.5x_2 - 0.2) = \begin{cases} 1, & 0.5x_1 + 0.5x_2 - 0.2 \geq 0 \\ 0, & 0.5x_1 + 0.5x_2 - 0.2 < 0 \end{cases} \tag{2.6}$$

พิจารณาค่านำเข้า $(0,0)$ จะได้

$$z = 0.5(0) + 0.5(0) - 0.2 = -0.2$$

เนื่องจาก $-0.2 < 0$ จะได้ว่า $y = 0$ นั่นคือเพอร์เซปตรอนให้ค่าส่งออก 0 สำหรับค่านำเข้า $(0,0)$

สำหรับค่านำเข้า $(0,1),(1,0),(1,1)$ จะได้ค่า $z$ ตามลำดับดังนี้

ค่านำเข้า $(0,1)$ $z = 0.5(0) + 0.5(1) - 0.2 = 0.5 - 0.2 = 0.3$

ค่านำเข้า $(1,0)$ $z = 0.5(1) + 0.5(0) - 0.2 = 0.5 - 0.2 = 0.3$

ค่านำเข้า $(1,1)$ $z = 0.5(1) + 0.5(1) - 0.2 = 1 - 0.2 = 0.8$

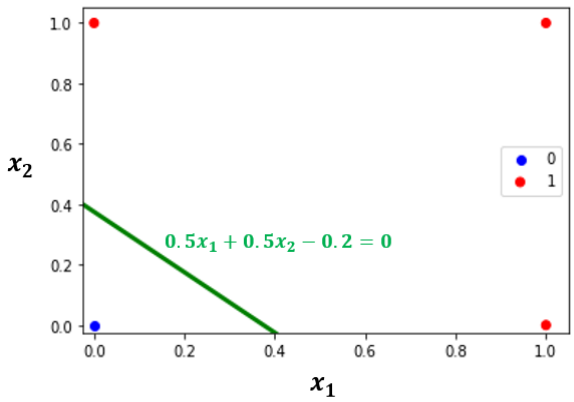
เนื่องจาก $z \geq 0$ ทั้งหมด จะได้ว่าเพอร์เซ็ปตรอนให้ค่าส่งออก 1,1,1 สำหรับค่านำเข้า $(0,1),(1,0),(1,1)$

ตาราง 2.2 สรุปการคำนวณของเพอร์เซปตรอน โดยสังเกตได้ว่าเพอร์เซปตรอนดังกล่าวสามารถคัดแยกข้อมูลได้ตรงกับค่าเป้าหมายทั้งหมด

| $x_1$ | $x_2$ | $z$ | $y$ | $t$ |
|---|---|---|---|---|
| 0 | 0 | -0.2 | 0 | 0 |
| 0 | 1 | 0.3 | 1 | 1 |
| 1 | 0 | 0.3 | 1 | 1 |
| 1 | 1 | 0.8 | 1 | 1 |

ตารางที่ 2.2: การคำนวณค่าในเพอร์เซปตรอนหนึ่งตัวของปัญหา OR

เมื่อพิจารณาสมการ 2.6 สังเกตได้ว่าเพอร์เซปตรอนใช้เส้นตรง $0.5x_1 + 0.5x_2 - 0.2 = 0$ ในการคัดแยกข้อมูลระหว่าง 0 กับ 1 โดยสามารถแสดงได้ดังแสดงในรูปที่ 2.2



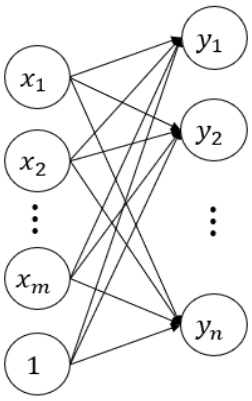

รูปที่ 2.2: เส้นตรงสำหรับคัดแยกข้อมูลในปัญหา OR

นั่นแสดงว่าสำหรับปัญหาที่มีค่าเป้าหมายคือ 0 กับ 1 หากกำหนดฟังก์ชันกระตุ้นเป็นฟังก์ชันฮาร์ดลิมิตแบบสมมาตร (Symmetric Hard-Limit) เพอร์เซปตรอนหนึ่งตัวสามารถแสดงได้ด้วยเส้นตรงหนึ่งเส้น

สำหรับกรณีที่เพอร์เซปตรอนมีปมส่งออกหลายปม เพอร์เซปตรอนดังกล่าวมีค่าส่งออกหลายค่า กำหนดให้ $\mathbf{x} = (x_1, x_2, ..., x_m) \in \mathbb{R}^m$ เป็นค่านำเข้า $\mathbf{w_j} = (w_{j,1}, w_{j,2}, ..., w_{j,m}) \in \mathbb{R}^m$ และ $b_j \in \mathbb{R}$ เป็นน้ำหนักและไบแอสของปมส่งออกที่ $j$ และ $f : \mathbb{R} \to \mathbb{R}$ เป็นฟังก์ชันกระตุ้น จะได้ว่าหากมีปมส่งออก $n$ ปม เพอร์เซปตรอนดังกล่าวจะให้ค่าส่งออก $\mathbf{y} = (y_1, y_2, y_3, ..., y_n) \in \mathbb{R}^n$ โดยค่าส่งออก $y_j$ คำนวณได้จาก

$$y_j = f(z_j) = f(\mathbf{w_j} \cdot \mathbf{x} + b_j) = f(\sum_{i=1}^{m} w_{j,i}(x_i) + b_j) \tag{2.7}$$

เพอร์เซปตรอนที่มีปมส่งออกหลายปมสามารถเขียนให้อยู่ในรูปกราฟได้ดังแสดงในรูปที่ 2.3



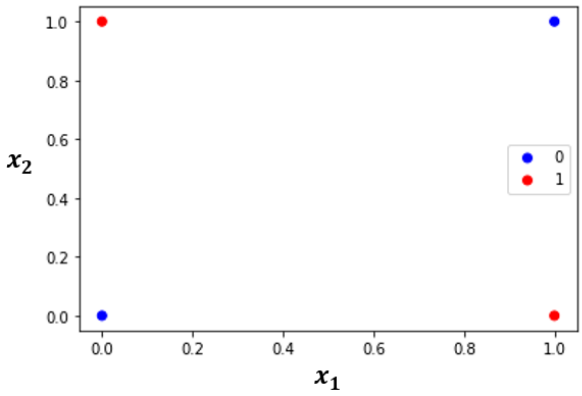

รูปที่ 2.3: กราฟเพอร์เซปตรอนที่มีปมส่งออกหลายปม

**ข้อสังเกต 2.6** พิจารณาปัญหา XOR ดังตาราง

| $x_1$ | $x_2$ | $t$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

ตารางที่ 2.3: ปัญหา XOR

เมื่อพลอตข้อมูลเป็นจุด จะได้รูปที่ 2.4




รูปที่ 2.4: ข้อมูลจุดในปัญหา XOR

เมื่อพิจารณาปัญหานี้ ค่าเป้าหมายของปัญหาคือค่า 0 กับ 1 หากกำหนดให้เพอร์เซปตรอนใช้ฟังก์ชันกระตุ้นฮาร์ดลิมิตแบบสมมาตร (Symmetric Hard-Limit) เพอร์เซปตรอนหนึ่งตัวจะแสดงได้ด้วยเส้นตรงหนึ่งเส้น และเมื่อสังเกตที่รูปภาพข้อมูลจุดจะพบว่าไม่สามารถใช้เส้นตรงเพียงหนึ่งเส้นในการแบ่งข้อมูลออกได้ แสดงว่าปัญหานี้ไม่สามารถใช้เพอร์เซปตรอนหนึ่งตัวในการคัดแยกข้อมูลได้

### 2.2.2. **เพอร์เซปตรอนหลายชั้น (Multi-layer perceptron)**

เพอร์เซปตรอนสามารถนำมาต่อกันได้ โดยเพอร์เซปตรอนตัวถัดไปจะรับค่าส่งออกของเพอร์เซปตรอนตัวก่อนมาเป็นค่านำเข้า เราจะเรียกเพอร์เซปตรอนลักษณะนี้ว่า**เพอร์เซปตรอนหลายชั้น**

กำหนดให้ $\mathbf{y}^{(l)} = (y_1^{(l)}, y_2^{(l)}, ..., y_{n_l}^{(l)}) \in \mathbb{R}^{n_l}$ เป็นค่าส่งออกจากเพอร์เซปตรอนชั้นที่ $l$ ที่มีจำนวนปมส่งออก $n_l$ ปม

$\mathbf{w_j}^{(l+1)} = (w_{j,1}^{(l+1)}, w_{j,2}^{(l+1)}, ..., w_{j,n_l}^{(l+1)}) \in \mathbb{R}^{n_l}$ เป็นน้ำหนักของปมส่งออกที่ $j$ ชั้นที่ $l+1$

$b_j^{(l+1)} \in \mathbb{R}$ เป็นไบแอสของปมส่งออกที่ $j$ ชั้นที่ $l+1$

เมื่อ $l = 0, 1, 2, ..., l^*$ โดยที่ $l^*$ คือจำนวนชั้นของเพอร์เซปตรอน และ $\mathbf{y}^{(0)} = \mathbf{x} = (x_1, x_2, ..., x_m) \in \mathbb{R}^m$

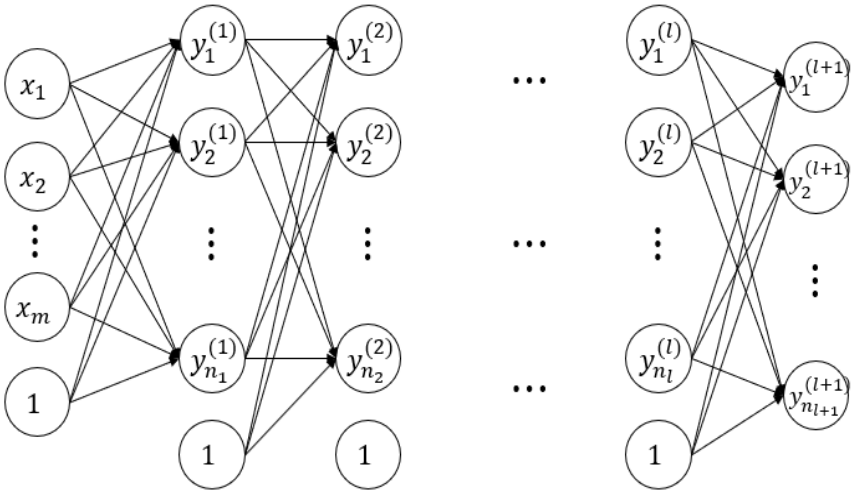
ถ้าเพอร์เซปตรอนในชั้น $l+1$ มีฟังก์ชันกระตุ้น คือ $f : \mathbb{R} \to \mathbb{R}$ และมีปมส่งออก $n_{l+1}$ ปม แล้วค่าส่งออกในชั้นที่ $l+1$ คือ

$$\mathbf{y}^{(l+1)} = (y_1^{(l+1)}, y_2^{(l+1)}, ..., y_{n_{l+1}}^{(l+1)}) \in \mathbb{R}^{n_{l+1}}$$

โดย $y_j^{(l+1)}$ คือค่าส่งออกที่ $j$ ของชั้นที่ $l+1$ ซึ่งคำนวณได้จาก

$$y_j^{(l+1)} = f(z_j^{(l+1)}) = f(\mathbf{w_j}^{(l+1)} \cdot \mathbf{y}^{(l)} + b_j^{(l+1)}) = f(\sum_{i=1}^{m} w_{j,i}^{(l+1)}(y_i^l) + b_j^{(l+1)}) \tag{2.8}$$

หากกำหนดให้ $l^* = l+1$ นั่นคือชั้นสุดท้ายของเพอร์เซปตรอนหลายชั้นคือชั้นที่ $l+1$ จะได้ว่ากราฟของเพอร์เซปตรอนหลายชั้นดังกล่าวเป็นไปตามรูปที่ 2.5



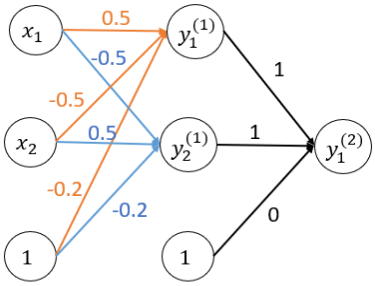

รูปที่ 2.5: กราฟเพอร์เซ็ปตรอนหลายชั้น

**ตัวอย่าง 2.7** พิจารณาปัญหา XOR (Exculsive OR) ที่มีข้อมูลนำเข้า $\mathbf{x} = (x_1, x_2) \in \{0,1\}^2$ และข้อมูลเป้าหมาย $t \in \{0,1\}$ ดังตาราง

| $x_1$ | $x_2$ | $t$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

ตารางที่ 2.4: ปัญหา XOR

จากข้อสังเกต 2.6 ปัญหาดังกล่าวไม่สามารถแบ่งข้อมูลด้วยเพอร์เซปตรอนเพียงหนึ่งตัวได้ ซึ่งในกรณีนี้เพอร์เซปตรอนใช้ฟังก์ชันกระตุ้นเป็นฟังก์ชันฮาร์ดลิมิตแบบสมมาตร (Symmetric Hard-Limit) แต่ถ้าสร้างเพอร์เซปตรอนหลายชั้นที่ใช้ฟังก์ชันกระตุ้นเป็นฟังก์ชันฮาร์ดลิมิตแบบสมมาตรในทุกๆชั้น ซึ่งเพอร์เซปตรอนมีลักษณะดังแสดงในรูปที่ 2.6




รูปที่ 2.6: เพอร์เซปตรอนสำหรับคัดแยกปัญหา XOR

ปัญหา XOR จะสามารถแบ่งได้ ซึ่งจะอธิบายดังข้างล่างนี้

พิจารณาข้อมูล (0,0) จะได้ว่าสำหรับชั้นแรก

$$z_1^{(1)} = 0.5(0) - 0.5(0) - 0.2 = -0.2, \qquad z_2^{(1)} = -0.5(0) + 0.5(0) - 0.2 = -0.2$$

เนื่องจาก $f$ เป็นฟังก์ชันฮาร์ดลิมิตแบบสมมาตร และ $z_1^{(1)}, z_2^{(1)} < 0$ จะได้ว่า $y_1^{(1)} = 0$ และ $y_2^{(1)} = 0$

สำหรับชั้นที่สองจะได้ว่า

$$z_1^{(2)} = 1(0) + 1(0) + 0 = 0$$

เนื่องจาก $f$ เป็นฟังก์ชันฮาร์ดลิมิตแบบสมมาตร และ $z_1^{(2)} < 0$ จะได้ว่า $y_1^{(2)} = 0$ นั่นคือ

เพอร์เซปตรอนคัดแยกข้อมูล (0,0) อยู่ในกลุ่ม 0

พิจารณาข้อมูล (0,1), (1,0), (1,1) จะได้ค่า $z_1^{(1)}$ และ $z_2^{(1)}$ ตามลำดับดังนี้

ค่านำเข้า $(0,1)$ $z_1^{(1)} = 0.5(0) - 0.5(1) - 0.2 = -0.7, \qquad z_2^{(1)} = -0.5(0) + 0.5(1) - 0.2 = 0.3$

ค่านำเข้า $(1,0)$ $z_1^{(1)} = 0.5(1) - 0.5(0) - 0.2 = 0.3, \qquad z_2^{(1)} = -0.5(1) + 0.5(0) - 0.2 = -0.7$

ค่านำเข้า $(1,1)$ $z_1^{(1)} = 0.5(1) - 0.5(1) - 0.2 = -0.2, \qquad z_2^{(1)} = -0.5(1) + 0.5(1) - 0.2 = -0.2$

จะได้ว่า $y_1^{(1)} = 0, 1, 0$ และ $y_2^{(1)} = 1, 0, 0$ ตามลำดับ

สำหรับชั้นที่สองจะได้ค่า $z_1^{(2)}$ ตามลำดับดังนี้

ค่านำเข้า $(0,1)$ $z_1^{(2)} = 1(0) + 1(1) + 0 = 1$

ค่านำเข้า $(1,0)$ $z_1^{(2)} = 1(1) + 1(0) + 0 = 1$

ค่านำเข้า $(1,1)$ $z_1^{(2)} = 1(0) + 1(0) + 0 = 0$

จะได้ว่า $y_1^{(2)} = 1, 1, 0$ ตามลำดับ นั่นคือเพอร์เซปตรอนคัดแยกข้อมูล (0,1),(1,0),(1,1) อยู่ในกลุ่ม 1,1,0 ตามลำดับ

ตารางที่ 2.5 สรุปการคำนวณของเพอร์เซปตรอนหลายชั้น จะเห็นได้ว่าเพอร์เซปตรอนหลายชั้นสามารถคัดแยกข้อมูลได้ตรงกับค่าเป้าหมายทั้งหมด

| $x_1$ | $x_2$ | $z_1^{(1)}$ | $z_2^{(1)}$ | $y_1^{(1)}$ | $y_2^{(1)}$ | $z_1^{(2)}$ | $y_1^{(2)}$ | $t$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | -0.2 | -0.2 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | -0.7 | 0.3 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0.3 | -0.7 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | -0.2 | -0.2 | 0 | 0 | 0 | 0 | 0 |

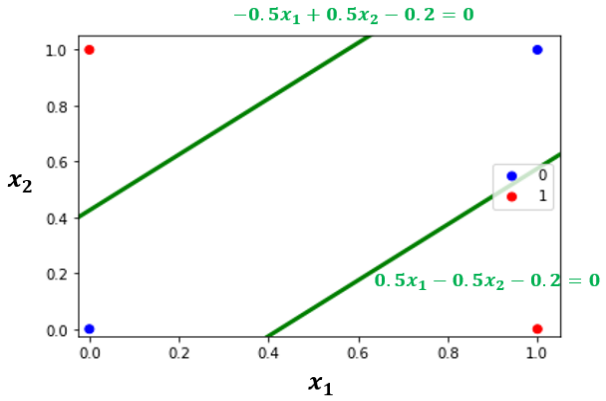
ตารางที่ 2.5: การคำนวณค่าในเพอร์เซปตรอนหลายชั้นของปัญหา XOR

พิจารณารูปที่ 2.6 จากน้ำหนัก ไบแอส และฟังก์ชันกระตุ้นฮาร์ดลิมิตแบบสมมาตร จะได้ว่าในชั้นที่หนึ่งมีการคำนวณดังนี้

$$y_1^{(1)} = f(z_1^{(1)}) = f(\mathbf{w_1}^{(1)} \cdot \mathbf{x} + b_1^{(1)}) = f(0.5x_1 - 0.5x_2 - 0.2) = \begin{cases} 1, & 0.5x_1 - 0.5x_2 - 0.2 \geq 0 \\ 0, & 0.5x_1 - 0.5x_2 - 0.2 < 0 \end{cases} \tag{2.9}$$

$$y_2^{(1)} = f(z_2^{(1)}) = f(\mathbf{w_2}^{(1)} \cdot \mathbf{x} + b_2^{(1)}) = f(-0.5x_1 + 0.5x_2 - 0.2) = \begin{cases} 1, & -0.5x_1 + 0.5x_2 - 0.2 \geq 0 \\ 0, & -0.5x_1 + 0.5x_2 - 0.2 < 0 \end{cases} \tag{2.10}$$

จากสมการ 2.9 และ 2.10 เมื่อพลอตกราฟเส้นตรง $0.5x_1 - 0.5x_2 - 0.2 = 0$ และ $-0.5x_1 + 0.5x_2 - 0.2 = 0$ จะได้รูปที่ 2.7




รูปที่ 2.7: กราฟเส้นตรงสองเส้นของเพอร์เซปตรอนชั้นที่หนึ่ง

เมื่อพิจารณากราฟและสมการ จะได้ว่าเพอร์เซปตรอนชั้นที่หนึ่งสามารถระบุตำแหน่งของข้อมูลได้ โดยเส้นตรง $0.5x_1 - 0.5x_2 - 0.2 = 0$ ถ้าข้อมูลอยู่เหนือเส้นตรงดังกล่าว จะถูกคัดแยกเป็นข้อมูลกลุ่ม 0 ในขณะที่ถ้าข้อมูลอยู่ต่ำกว่าเส้นตรงดังกล่าว จะถูกคัดแยกเป็นข้อมูลกลุ่ม 1 ในฝั่งตรงข้าม สำหรับเส้นตรง $-0.5x_1 + 0.5x_2 - 0.2 = 0$ ถ้าข้อมูลอยู่เหนือเส้นตรงดังกล่าว จะถูกคัดแยกเป็นข้อมูลกลุ่ม 1 ในขณะที่ถ้าข้อมูลอยู่ต่ำกว่าเส้นตรงดังกล่าว จะถูกคัดแยกเป็นข้อมูลกลุ่ม 0

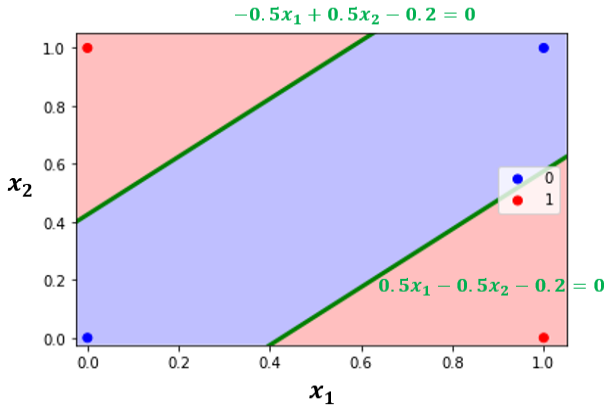
สังเกตได้ว่าข้อมูลที่มีค่าจริงเป็น 1 จะถูกคัดแยกเป็นข้อมูลกลุ่ม 1 จากเส้นตรงหนึ่งเส้นในสองเส้น แต่ข้อมูลที่มีค่าจริงเป็น 0 ถูกคัดแยกเป็นข้อมูลกลุ่ม 0 สำหรับเส้นตรงทั้งสองเส้น

พิจารณาเพอร์เซปตรอนชั้นที่สอง จะได้ว่า

$$y_1^{(2)} = f(z_1^{(2)}) = f(\mathbf{w_1}^{(2)} \cdot \mathbf{y}^{(1)} + b_1^{(2)}) = f(y_1 + y_2) = \begin{cases} 1, & y_1 + y_2 \geq 0 \\ 0, & y_1 + y_2 < 0 \end{cases} \tag{2.11}$$

จากสมการ 2.11 แสดงให้เห็นว่าเพอร์เซปตรอนชั้นที่สองสามารถคัดแยกประเภทข้อมูลจากตำแหน่งของข้อมูลในเพอร์เซปตรอนชั้นที่หนึ่งได้ โดยหากข้อมูลอยู่ในกลุ่ม 0 ทั้งสองเส้นตรง เพอร์เซปตรอนชั้นที่สองจะคัดแยกข้อมูลอยู่ในกลุ่ม 0 ในขณะที่หากข้อมูลอยู่ในกลุ่ม 1 ในเส้นตรงหนึ่งเส้น เพอร์เซปตรอนจะคัดแยกข้อมูลอยู่ในกลุ่ม 1

ดังนั้นถ้าข้อมูลอยู่ระหว่างเส้นตรง $0.5x_1 - 0.5x_2 - 0.2 = 0$ และ $-0.5x_1 + 0.5x_2 - 0.2 = 0$ ข้อมูลจะถูกคัดแยกอยู่ในกลุ่ม 0 แต่ถ้าข้อมูลไม่อยู่ระหว่างเส้นตรงจะถูกคัดแยกให้อยู่ในกลุ่ม 1 ดังแสดงในรูปที่ 2.8 ซึ่งจะเห็นได้ว่าเมื่อข้อมูลอยู่ในบริเวณสีน้ำเงินจะถูกคัดแยกเป็นกลุ่ม 0 และข้อมูลในกลุ่มสีแดงจะถูกคัดแยกเป็นกลุ่ม 1




รูปที่ 2.8: การคัดแยกข้อมูลของเพอร์เซปตรอนหลายชั้นแบ่งตามตำแหน่งข้อมูล

### 2.2.3. การวัดค่าความผิดพลาด

สำหรับข้อมูล $\{(\mathbf{x}^{\{1\}}, \mathbf{t}^{\{1\}}), (\mathbf{x}^{\{2\}}, \mathbf{t}^{\{2\}}), ..., (\mathbf{x}^{\{k\}}, \mathbf{t}^{\{k\}})\}$

กำหนดให้ $\mathbf{x}^{\{i\}} = (x_1^{\{i\}}, x_2^{\{i\}}, ..., x_m^{\{i\}}) \in \mathbb{R}^m$ เป็นข้อมูลนำเข้าชุดที่ $i$ สำหรับป้อนให้กับโครงข่ายประสาทเทียม

$\mathbf{t}^{\{i\}} = (t_1^{\{i\}}, t_2^{\{i\}}, ..., t_n^{\{i\}}) \in \mathbb{R}^n$ เป็นข้อมูลเป้าหมายชุดที่ $i$ ที่ต้องการให้โมเดลทำนาย

$\mathbf{o}^{\{i\}} = (o_1^{\{i\}}, o_2^{\{i\}}, ..., o_n^{\{i\}}) \in \mathbb{R}^n$ เป็นข้อมูลส่งออกในชั้นสุดท้ายเมื่อป้อนข้อมูลนำเข้าชุดที่ $i$

เมื่อ $i = 1, 2, ..., k$

เราสามารถวัดค่าความผิดพลาดระหว่างค่าส่งออกหรือค่าทำนายของโมเดล $\mathbf{o}^{\{i\}}$ กับค่าจริง $\mathbf{t}^{\{i\}}$ ด้วยฟังก์ชันสูญเสีย (Loss function) ซึ่งเป็นฟังก์ชันที่ใช้ในการหาค่าความแตกต่างระหว่างค่าทำนายที่เกิดจากน้ำหนักและไบแอสของโครงข่ายประสาทเทียมดังกล่าวกับค่าเป้าหมาย ในกรณีที่มีข้อมูลสำหรับการสอนโครงข่ายประสาทเทียมหลายค่า ค่าความผิดพลาดที่วัดจากฟังก์ชันสูญเสียจะถูกนำมาเฉลี่ยตามจำนวนข้อมูลดังสมการ

$$E = \frac{1}{k}\sum_{i=1}^{k} e(\mathbf{o}^{\{i\}}, \mathbf{t}^{\{i\}}) = \frac{1}{k}\sum_{i=1}^{k} e(\mathbf{x}^{\{i\}}, \mathbf{t}^{\{i\}}, \mathbf{w_1}^{(1)}, b_1^{(1)}, \mathbf{w_2}^{(1)}, b_2^{(1)}, ...) \tag{2.12}$$

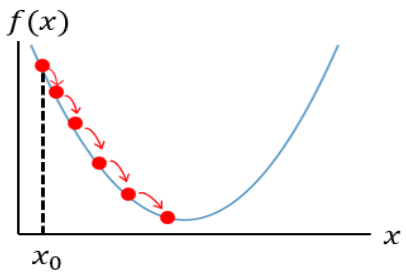
เมื่อ $e$ เป็นฟังก์ชันสูญเสีย

### 2.2.4. ขั้นตอนวิธี Gradient descent

Gradient descent เป็นกระบวนการที่ใช้ในการหาจุดที่ทำให้ฟังก์ชันมีค่าต่ำที่สุด $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$ นั่นคือหาจุด $\mathbf{x} \in \mathbb{R}^n$ ที่ทำให้ฟังก์ชัน $f : \mathbb{R}^n \to \mathbb{R}$ มีค่าต่ำที่สุด ด้วยการปรับจุดไปในทิศทางตรงกันข้ามกับความชันของจุดดังกล่าว โดยขั้นตอน Gradient descent แสดงได้ดังนี้

1. กำหนดอัตราการเรียนรู้ (learning rate) $\beta > 0$ และจำนวนครั้งในการเรียนรู้ $k \in \mathbb{Z}^+$
2. กำหนดฟังก์ชันที่ต้องการจะหาค่าต่ำสุด $f$ กำหนดจุดที่ต้องการหาเพื่อทำให้ฟังก์ชันมีค่าต่ำที่สุด $\mathbf{x}$ และหาค่าอนุพันธ์ของฟังก์ชันเทียบกับจุดดังกล่าว $f'(\mathbf{x})$
3. กำหนดจุดเริ่มต้น $\mathbf{x}(0)$ คำนวณ $\mathbf{x}(1) = \mathbf{x}(0) - \beta(f'(\mathbf{x}(0)))$
4. ทำซ้ำสมการ $\mathbf{x}(i+1) = \mathbf{x}(i) - \beta(f'(\mathbf{x}(i)))$ จนกระทั่งคำนวณจุด $\mathbf{x}(k)$ ได้

เมื่อ $n = 1$ ขั้นตอน 4 ขั้นตอนสามารถอธิบายได้ด้วยรูปภาพดังแสดงในรูปที่ 2.9




รูปที่ 2.9: กระบวนการ Gradient descent

จากรูปที่ 2.9 ขนาดของการปรับค่า $\mathbf{x}$ จะเป็นไปตามอัตราการเรียนรู้ หากอัตราการเรียนรู้สูงอาจทำให้ Gradient descent ปรับค่า $\mathbf{x}$ ได้รวดเร็วขึ้น แต่ถ้าขนาดการปรับมากเกินไป อาจส่งผลให้ $f$ ไม่ลู่เข้าสู่จุดต่ำสุด ในขณะเดียวกันถ้าอัตราการเรียนรู้ต่ำจะทำให้การปรับค่า $\mathbf{x}$ ได้ช้าลง ซึ่งอาจเพิ่มโอกาสให้ฟังก์ชันลู่เข้าจุดต่ำสุดได้สำเร็จมากขึ้น หากฟังก์ชันที่ใช้ในการปรับค่ามีจุดต่ำสุดสัมพัทธ์หลายจุด ค่าที่ได้จาก gradient descent อาจเป็นเพียงจุดต่ำสุดสัมพัทธ์ โดยหนึ่งในวิธีสำหรับแก้ไขปัญหาข้างต้นคือการสุ่มเลือกจุดเริ่มต้น $\mathbf{x}(0)$ ใหม่

ในโครงข่ายประสาทเทียมจะมีการใช้ Gradient descent สำหรับการหาน้ำหนักและไบแอสที่ทำให้ค่าสูญเสียมีค่าน้อยที่สุด กล่าวคือทำให้โมเดลสามารถทำนายผลให้ตรงตามข้อมูลจริงมากที่สุด โดยในบางครั้งข้อมูลที่ใช้ในการสอนโมเดลอาจมีจำนวนมากเกินไป ทำให้ไม่สามารถป้อนข้อมูลทั้งหมดและทำการปรับน้ำหนักผ่านทาง Gradient descent ในครั้งเดียวได้ ดังนั้นจึงมีการแบ่งข้อมูลออกเป็นหลายๆ batch ที่มีจำนวนข้อมูลเท่ากัน แล้วปรับค่าน้ำหนักและไบแอสหลายๆ ครั้งในการเรียนรู้หนึ่งรอบ ขั้นตอนวิธีดังกล่าวเรียกว่า **Stochastic Gradient descent**

### 2.2.5. การแพร่กลับ (Back propagation)

การแพร่กลับเป็นกระบวนการที่ใช้ในการคำนวณหาค่าอนุพันธ์ของค่าสูญเสียเทียบกับน้ำหนักหรือไบแอสในโครงข่ายประสาทเทียมสำหรับใช้ใน Gradient descent โดยใช้กฎลูกโซ่ กำหนดให้
$E = \frac{1}{k}\sum_{i=1}^{k} e(\mathbf{x^{\{i\}}}, \mathbf{t^{\{i\}}}, \mathbf{w_1}^{(1)}, b_1^{(1)}, \mathbf{w_2}^{(1)}, b_2^{(1)}, ...)$ เป็นค่าเฉลี่ยของค่าสูญเสีย สำหรับโครงข่ายประสาทเทียมที่มีน้ำหนัก $\mathbf{w_j}^{(l)}$ และไบแอส $b_j^{(l)}$ หากโครงข่ายประสาทเทียมมีชั้นสุดท้ายที่ชั้น $l$ การแพร่กลับมีขั้นตอนดังนี้

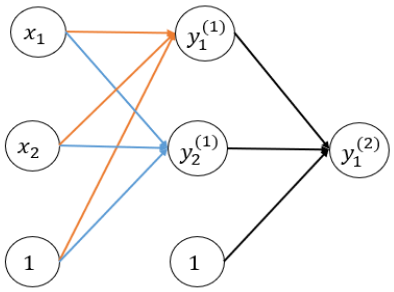
1. สำหรับชั้นสุดท้าย คำนวณหาค่าอนุพันธ์ $\frac{\partial E}{\partial w_{j,i}^{(l)}}$ และ $\frac{\partial E}{\partial b_j^{(l)}}$ สำหรับทุกๆน้ำหนักและไบแอสในชั้นที่ $l$
2. คำนวณหาค่าอนุพันธ์ $\frac{\partial E}{\partial y_i^{(l-1)}}$ สำหรับทุกๆ $y_i^{(l-1)}$ ของชั้นที่ $l-1$
3. นำค่าอนุพันธ์ในข้อที่ 2. มาใช้ในการคำนวณหาอนุพันธ์ $\frac{\partial E}{\partial w_{j,i}^{(l-1)}}$ และ $\frac{\partial E}{\partial b_j^{(l-1)}}$ สำหรับทุกๆน้ำหนักและไบแอสในชั้นที่ $l-1$
4. ทำซ้ำจนสามารถคำนวณหาค่า $\frac{\partial E}{\partial w_{j,i}^{(1)}}$ และ $\frac{\partial E}{\partial b_j^{(1)}}$ สำหรับทุกๆน้ำหนักและไบแอสในชั้นที่ 1 แล้วจึงหยุด

**ตัวอย่าง 2.8** พิจารณาปัญหา XOR ดังตาราง

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

ตารางที่ 2.6: ปัญหา XOR

พิจารณาเพอร์เซปตรอนหลายชั้นดังแสดงในรูป




รูปที่ 2.10: เพอร์เซปตรอนหลายชั้นสำหรับปัญหา XOR

สมมุติให้ฟังก์ชันกระตุ้นในทุกเพอร์เซปตรอนเป็นฟังก์ชันเชิงเส้น และฟังก์ชันสูญเสียเป็นฟังก์ชันค่าผิดพลาดกำลังสอง (Mean Square Error) $E = \frac{1}{4}\sum_{k=1}^{4}(o^{\{k\}} - t^{\{k\}})^2$ เราสามารถใช้ gradient descent ในการหาน้ำหนักและไบแอสที่เหมาะสมสำหรับคัดแยกปัญหา XOR โดยการหาค่าน้ำหนักและไบแอสที่ทำให้ค่าสูญเสียมีค่าน้อยที่สุด กล่าวคือทำให้เพอร์เซปตรอนแยกข้อมูลได้ถูกต้องที่สุด

กำหนดชุดข้อมูลนำเข้า $\mathbf{x^{\{1\}}}, \mathbf{x^{\{2\}}}, \mathbf{x^{\{3\}}}, \mathbf{x^{\{4\}}} = (0,0),(0,1),(1,0),(1,1)$ ชุดข้อมูลเป้าหมาย $t^{\{1\}}, t^{\{2\}}, t^{\{3\}}, t^{\{4\}} = 0,1,1,0$ และกำหนดให้ $y_j^{(l),\{k\}}$ แทนค่าส่งออกที่ $j$ ในชั้นที่ $l$ ของข้อมูลชุดที่ $k$

จะได้ว่า $E = \frac{1}{4}\sum_{k=1}^{4}(o^{\{k\}} - t^{\{k\}})^2 = \frac{1}{4}\sum_{k=1}^{4}(y_1^{(2),\{k\}} - t^{\{k\}})^2$

สำหรับการแพร่ย้อนกลับ พิจารณาชั้นที่ 2 จะได้ว่า

$$y_1^{(2),\{1\}} = w_{1,1}^{(2)} y_1^{(1),\{1\}} + w_{1,2}^{(2)} y_2^{(1),\{1\}} + b_1^{(2)}$$

$$y_1^{(2),\{2\}} = w_{1,1}^{(2)} y_1^{(1),\{2\}} + w_{1,2}^{(2)} y_2^{(1),\{2\}} + b_1^{(2)}$$

$$y_1^{(2),\{3\}} = w_{1,1}^{(2)} y_1^{(1),\{3\}} + w_{1,2}^{(2)} y_2^{(1),\{3\}} + b_1^{(2)}$$

$$y_1^{(2),\{4\}} = w_{1,1}^{(2)} y_1^{(1),\{4\}} + w_{1,2}^{(2)} y_2^{(1),\{4\}} + b_1^{(2)}$$

เนื่องจาก $E$ เป็นฟังก์ชันของ $y_1^{(2),\{k\}}$ และ $y_1^{(2),\{k\}}$ เป็นฟังก์ชันของ $w_{1,1}^{(2)}, w_{1,2}^{(2)}, b_1^{(2)}$ เมื่อ $k = 1,2,3,4$ จะได้ว่าจากกฎลูกโซ่

$$\frac{\partial E}{\partial w_{1,1}^{(2)}} = \sum_{k=1}^{4} \frac{\partial E}{\partial y_1^{(2),\{k\}}} \frac{\partial y_1^{(2),\{k\}}}{\partial w_{1,1}^{(2)}} = \sum_{k=1}^{4} \frac{1}{4}(2)(y_1^{(2),\{k\}} - t^{\{k\}})(y_1^{(1),\{k\}})$$

$$\frac{\partial E}{\partial w_{1,2}^{(2)}} = \sum_{k=1}^{4} \frac{\partial E}{\partial y_1^{(2),\{k\}}} \frac{\partial y_1^{(2),\{k\}}}{\partial w_{1,2}^{(2)}} = \sum_{k=1}^{4} \frac{1}{4}(2)(y_1^{(2),\{k\}} - t^{\{k\}})(y_2^{(1),\{k\}})$$

$$\frac{\partial E}{\partial b_1^{(2)}} = \sum_{k=1}^{4} \frac{\partial E}{\partial y_1^{(2),\{k\}}} \frac{\partial y_1^{(2),\{k\}}}{\partial b_1^{(2)}} = \sum_{k=1}^{4} \frac{1}{4}(2)(y_1^{(2),\{k\}} - t^{\{k\}})(1)$$

พิจารณาค่าส่งออกในชั้นที่ 1 ของข้อมูลชุดที่ 1 เนื่องจาก $E$ เป็นฟังก์ชันของ $y_1^{(2),\{1\}}$ และ $y_1^{(2),\{1\}}$ เป็นฟังก์ชันของ $y_1^{(1),\{1\}}, y_2^{(1),\{1\}}$ จะได้

$$\frac{\partial E}{\partial y_1^{(1),\{1\}}} = \frac{\partial E}{\partial y_1^{(2),\{1\}}}\frac{\partial y_1^{(2),\{1\}}}{\partial y_1^{(1),\{1\}}} = \frac{1}{4}(2)(y_1^{(2),\{1\}} - t^{\{1\}})(w_{1,1}^{(2)})$$
$$\frac{\partial E}{\partial y_2^{(1),\{1\}}} = \frac{\partial E}{\partial y_1^{(2),\{1\}}}\frac{\partial y_1^{(2),\{1\}}}{\partial y_2^{(1),\{1\}}} = \frac{1}{4}(2)(y_1^{(2),\{1\}} - t^{\{1\}})(w_{1,2}^{(2)})$$

ในทำนองเดียวกันกับข้อมูลชุด 2,3,4 จะได้

$$\frac{\partial E}{\partial y_1^{(1),\{2\}}} = \frac{\partial E}{\partial y_1^{(2),\{2\}}}\frac{\partial y_1^{(2),\{2\}}}{\partial y_1^{(1),\{2\}}} = \frac{1}{4}(2)(y_1^{(2),\{2\}} - t^{\{2\}})(w_{1,1}^{(2)})$$
$$\frac{\partial E}{\partial y_2^{(1),\{2\}}} = \frac{\partial E}{\partial y_1^{(2),\{2\}}}\frac{\partial y_1^{(2),\{2\}}}{\partial y_2^{(1),\{2\}}} = \frac{1}{4}(2)(y_1^{(2),\{2\}} - t^{\{2\}})(w_{1,2}^{(2)})$$
$$\frac{\partial E}{\partial y_1^{(1),\{3\}}} = \frac{\partial E}{\partial y_1^{(2),\{3\}}}\frac{\partial y_1^{(2),\{3\}}}{\partial y_1^{(1),\{3\}}} = \frac{1}{4}(2)(y_1^{(2),\{3\}} - t^{\{3\}})(w_{1,1}^{(2)})$$
$$\frac{\partial E}{\partial y_2^{(1),\{3\}}} = \frac{\partial E}{\partial y_1^{(2),\{3\}}}\frac{\partial y_1^{(2),\{3\}}}{\partial y_2^{(1),\{3\}}} = \frac{1}{4}(2)(y_1^{(2),\{3\}} - t^{\{3\}})(w_{1,2}^{(2)})$$
$$\frac{\partial E}{\partial y_1^{(1),\{4\}}} = \frac{\partial E}{\partial y_1^{(2),\{4\}}}\frac{\partial y_1^{(2),\{4\}}}{\partial y_1^{(1),\{4\}}} = \frac{1}{4}(2)(y_1^{(2),\{4\}} - t^{\{4\}})(w_{1,1}^{(2)})$$
$$\frac{\partial E}{\partial y_2^{(1),\{4\}}} = \frac{\partial E}{\partial y_1^{(2),\{4\}}}\frac{\partial y_1^{(2),\{4\}}}{\partial y_2^{(1),\{4\}}} = \frac{1}{4}(2)(y_1^{(2),\{4\}} - t^{\{4\}})(w_{1,2}^{(2)})$$

เนื่องจาก $E = \frac{1}{4}\sum_{k=1}^{4}(y_1^{(2),\{k\}} - t^{\{k\}})^2 = \frac{1}{4}\sum_{k=1}^{4}(w_{1,1}^{(2)}y_1^{(1),\{k\}} + w_{1,2}^{(2)}y_2^{(1),\{k\}} + b_1^{(2)} - t^{\{k\}})^2$ จะได้ $E$ เป็นฟังก์ชันของ $y_1^{(1),\{k\}}, y_2^{(1),\{k\}}$ เมื่อ $k = 1, 2, 3, 4$

พิจารณาชั้นที่ 1 จะได้ว่า

$$y_1^{(1),\{1\}} = w_{1,1}^{(1)}x_1^{\{1\}} + w_{1,2}^{(1)}x_2^{\{1\}} + b_1^{(1)}, \qquad y_2^{(1),\{1\}} = w_{2,1}^{(1)}x_1^{\{1\}} + w_{2,2}^{(1)}x_2^{\{1\}} + b_2^{(1)}$$
$$y_1^{(1),\{2\}} = w_{1,1}^{(1)}x_1^{\{2\}} + w_{1,2}^{(1)}x_2^{\{2\}} + b_1^{(1)}, \qquad y_2^{(1),\{2\}} = w_{2,1}^{(1)}x_1^{\{2\}} + w_{2,2}^{(1)}x_2^{\{2\}} + b_2^{(1)}$$
$$y_1^{(1),\{3\}} = w_{1,1}^{(1)}x_1^{\{3\}} + w_{1,2}^{(1)}x_2^{\{3\}} + b_1^{(1)}, \qquad y_2^{(1),\{3\}} = w_{2,1}^{(1)}x_1^{\{3\}} + w_{2,2}^{(1)}x_2^{\{3\}} + b_2^{(1)}$$
$$y_1^{(1),\{4\}} = w_{1,1}^{(1)}x_1^{\{4\}} + w_{1,2}^{(1)}x_2^{\{4\}} + b_1^{(1)}, \qquad y_2^{(1),\{4\}} = w_{2,1}^{(1)}x_1^{\{4\}} + w_{2,2}^{(1)}x_2^{\{4\}} + b_2^{(1)}$$

พิจารณา $w_{1,1}^{(1)}, w_{1,2}^{(1)}$ และ $b_1^{(1)}$ เนื่องจาก $E$ เป็นฟังก์ชันของ $y_1^{(1),\{k\}}$ และ $y_1^{(1),\{k\}}$ เป็นฟังก์ชันของ $w_{1,1}^{(1)}, w_{1,2}^{(1)}$ และ $b_1^{(1)}$ เมื่อ $k = 1, 2, 3, 4$ จากกฎลูกโซ่จะได้ว่า

$$\frac{\partial E}{\partial w_{1,1}^{(1)}} = \sum_{k=1}^{4} \frac{\partial E}{\partial y_1^{(1),\{k\}}} \frac{\partial y_1^{(1),\{k\}}}{\partial w_{1,1}^{(1)}} = \sum_{k=1}^{4} \frac{1}{2}(y_1^{(2),\{k\}} - t^{\{k\}})(w_{1,1}^{(2)})(x_1^{\{k\}})$$

$$\frac{\partial E}{\partial w_{1,2}^{(1)}} = \sum_{k=1}^{4} \frac{\partial E}{\partial y_1^{(1),\{k\}}} \frac{\partial y_1^{(1),\{k\}}}{\partial w_{1,2}^{(1)}} = \sum_{k=1}^{4} \frac{1}{2}(y_1^{(2),\{k\}} - t^{\{k\}})(w_{1,1}^{(2)})(x_2^{\{k\}})$$

$$\frac{\partial E}{\partial b_1^{(1)}} = \sum_{k=1}^{4} \frac{\partial E}{\partial y_1^{(1),\{k\}}} \frac{\partial y_1^{(1),\{k\}}}{\partial b_1^{(1)}} = \sum_{k=1}^{4} \frac{1}{2}(y_1^{(2),\{k\}} - t^{\{k\}})(w_{1,1}^{(2)})(1)$$

ในทำนองเดียวกัน สำหรับ $w_{2,1}^{(1)}, w_{2,2}^{(1)}$ และ $b_2^{(1)}$ จะได้ว่า

$$\frac{\partial E}{\partial w_{2,1}^{(1)}} = \sum_{k=1}^{4} \frac{\partial E}{\partial y_2^{(1),\{k\}}} \frac{\partial y_2^{(1),\{k\}}}{\partial w_{2,1}^{(1)}} = \sum_{k=1}^{4} \frac{1}{2}(y_1^{(2),\{k\}} - t^{\{k\}})(w_{1,2}^{(2)})(x_1^{\{k\}})$$

$$\frac{\partial E}{\partial w_{2,2}^{(1)}} = \sum_{k=1}^{4} \frac{\partial E}{\partial y_2^{(1),\{k\}}} \frac{\partial y_2^{(1),\{k\}}}{\partial w_{2,2}^{(1)}} = \sum_{k=1}^{4} \frac{1}{2}(y_1^{(2),\{k\}} - t^{\{k\}})(w_{1,2}^{(2)})(x_2^{\{k\}})$$

$$\frac{\partial E}{\partial b_2^{(1)}} = \sum_{k=1}^{4} \frac{\partial E}{\partial y_2^{(1),\{k\}}} \frac{\partial y_2^{(1),\{k\}}}{\partial b_2^{(1)}} = \sum_{k=1}^{4} \frac{1}{2}(y_1^{(2),\{k\}} - t^{\{k\}})(w_{1,2}^{(2)})(1)$$

สำหรับ gradient descent กำหนดอัตราการเรียนรู้ $\beta = 0.01$ จำนวนครั้งในการเรียนรู้ $k = 25$ เราต้องการหาค่าน้ำหนักและไบแอสที่ทำให้ค่าสูญเสียต่ำที่สุด จึงกำหนดค่าเริ่มต้นน้ำหนักและไบแอสดังนี้

$$w_{1,1}^{(1)}(0) = 1, \qquad w_{1,2}^{(1)}(0) = 1, \qquad b_1^{(1)}(0) = 1$$

$$w_{2,1}^{(1)}(0) = 1, \qquad w_{2,2}^{(1)}(0) = 1, \qquad b_2^{(1)}(0) = 1$$

$$w_{1,1}^{(2)}(0) = 1, \qquad w_{1,2}^{(2)}(0) = 1, \qquad b_1^{(2)}(0) = 1$$

กำหนดให้ในการสอนโมเดล ไม่มีการเปลี่ยนแปลงข้อมูลระหว่างสอน จะได้ว่าข้อมูลนำเข้าและข้อมูลส่งออกเป็นค่าคงที่ นั่นคือค่าสูญเสียเป็นฟังก์ชันของน้ำหนักและไบแอสเท่านั้น จะได้ว่าในการเรียนรู้ครั้งที่ 1

$$w_{1,1}^{(1)}(1) = w_{1,1}^{(1)}(0) - \beta\frac{\partial E}{\partial w_{1,1}^{(1)}}(w_{1,1}^{(1)}(0), w_{1,2}^{(1)}(0), ..., b_1^{(2)}(0))$$

$$w_{1,2}^{(1)}(1) = w_{1,2}^{(1)}(0) - \beta\frac{\partial E}{\partial w_{1,2}^{(1)}}(w_{1,1}^{(1)}(0), w_{1,2}^{(1)}(0), ..., b_1^{(2)}(0))$$

$$b_1^{(1)}(1) = b_1^{(1)}(0) - \beta\frac{\partial E}{\partial b_1^{(1)}}(w_{1,1}^{(1)}(0), w_{1,2}^{(1)}(0), ..., b_1^{(2)}(0))$$

$$w_{2,1}^{(1)}(1) = w_{2,1}^{(1)}(0) - \beta\frac{\partial E}{\partial w_{2,1}^{(1)}}(w_{1,1}^{(1)}(0), w_{1,2}^{(1)}(0), ..., b_1^{(2)}(0))$$

$$w_{2,2}^{(1)}(1) = w_{2,2}^{(1)}(0) - \beta\frac{\partial E}{\partial w_{2,2}^{(1)}}(w_{1,1}^{(1)}(0), w_{1,2}^{(1)}(0), ..., b_1^{(2)}(0))$$

$$b_2^{(1)}(1) = b_2^{(1)}(0) - \beta\frac{\partial E}{\partial b_2^{(1)}}(w_{1,1}^{(1)}(0), w_{1,2}^{(1)}(0), ..., b_1^{(2)}(0))$$

$$w_{1,1}^{(2)}(1) = w_{1,1}^{(2)}(0) - \beta\frac{\partial E}{\partial w_{1,1}^{(2)}}(w_{1,1}^{(1)}(0), w_{1,2}^{(1)}(0), ..., b_1^{(2)}(0))$$

$$w_{1,2}^{(2)}(1) = w_{1,2}^{(2)}(0) - \beta\frac{\partial E}{\partial w_{1,2}^{(2)}}(w_{1,1}^{(1)}(0), w_{1,2}^{(1)}(0), ..., b_1^{(2)}(0))$$

$$b_1^{(2)}(1) = b_1^{(2)}(0) - \beta\frac{\partial E}{\partial b_1^{(2)}}(w_{1,1}^{(1)}(0), w_{1,2}^{(1)}(0), ..., b_1^{(2)}(0))$$

พิจารณา $w_{1,1}^{(1)}(1)$ เมื่อแทนค่าน้ำหนักและไบแอสเริ่มต้น อัตราการเรียนรู้ ข้อมูลนำเข้าและข้อมูลเป้าหมาย จะได้

$$w_{1,1}^{(1)}(1) = 1 - 0.01\frac{\partial E}{\partial w_{1,1}^{(1)}}(1, 1, ..., 1)$$

เนื่องจาก

$$\begin{aligned}\frac{\partial E}{\partial w_{1,1}^{(1)}} &= \sum_{k=1}^{4}\frac{1}{2}(y_1^{(2),\{k\}} - t^{\{k\}})(w_{1,1}^{(2)})(x_1^{\{k\}}) \\ &= \frac{w_{1,1}^{(2)}}{2}[(y_1^{(2),\{1\}} - t^{\{1\}})(x_1^{\{1\}}) + (y_1^{(2),\{2\}} - t^{\{2\}})(x_1^{\{2\}}) + (y_1^{(2),\{3\}} - t^{\{3\}})(x_1^{\{3\}}) + (y_1^{(2),\{4\}} - t^{\{4\}})(x_1^{\{4\}})]\end{aligned}$$

เมื่อแทนค่าน้ำหนัก ข้อมูลนำเข้า $\mathbf{x^{\{1\}}}, \mathbf{x^{\{2\}}}, \mathbf{x^{\{3\}}}, \mathbf{x^{\{4\}}} = (0,0), (0,1), (1,0), (1,1)$ และข้อมูลเป้าหมาย $t^{\{1\}}, t^{\{2\}}, t^{\{3\}}, t^{\{4\}} = 0, 1, 1, 0$ จะได้

$$\begin{aligned}\frac{\partial E}{\partial w_{1,1}^{(1)}}(1, 1, ..., 1) &= \frac{1}{2}[(y_1^{(2),\{1\}} - 0)(0) + (y_1^{(2),\{2\}} - 1)(0) + (y_1^{(2),\{3\}} - 1)(1) + (y_1^{(2),\{4\}} - 0)(1)] \\ &= \frac{1}{2}[(y_1^{(2),\{3\}} - 1) + (y_1^{(2),\{4\}})]\end{aligned}$$

เนื่องจาก $y_1^{(2),\{3\}} = w_{1,1}^{(2)} y_1^{(1),\{3\}} + w_{1,2}^{(2)} y_2^{(1),\{3\}} + b_1^{(2)}, \quad y_1^{(2),\{4\}} = w_{1,1}^{(2)} y_1^{(1),\{4\}} + w_{1,2}^{(2)} y_2^{(1),\{4\}} + b_1^{(2)}$

เมื่อแทนค่าน้ำหนักและไบแอสเริ่มต้นจะได้

$$y_1^{(2),\{3\}} = (1)(y_1^{(1),\{3\}}) + (1)(y_2^{(1),\{3\}}) + 1 = y_1^{(1),\{3\}} + y_2^{(1),\{3\}} + 1$$
$$y_1^{(2),\{4\}} = (1)(y_1^{(1),\{4\}}) + (1)(y_2^{(1),\{4\}}) + 1 = y_1^{(1),\{4\}} + y_2^{(1),\{4\}} + 1$$

เนื่องจาก

$$y_1^{(1),\{3\}} = w_{1,1}^{(1)} x_1^{\{3\}} + w_{1,2}^{(1)} x_2^{\{3\}} + b_1^{(1)}, \qquad y_2^{(1),\{3\}} = w_{2,1}^{(1)} x_1^{\{3\}} + w_{2,2}^{(1)} x_2^{\{3\}} + b_2^{(1)}$$
$$y_1^{(1),\{4\}} = w_{1,1}^{(1)} x_1^{\{4\}} + w_{1,2}^{(1)} x_2^{\{4\}} + b_1^{(1)}, \qquad y_2^{(1),\{4\}} = w_{2,1}^{(1)} x_1^{\{4\}} + w_{2,2}^{(1)} x_2^{\{4\}} + b_2^{(1)}$$

เมื่อแทนค่าจะได้

$$y_1^{(1),\{3\}} = (1)(1) + (1)(0) + 1 = 2, \qquad y_2^{(1),\{3\}} = (1)(1) + (1)(0) + 1 = 2$$
$$y_1^{(1),\{4\}} = (1)(1) + (1)(1) + 1 = 3, \qquad y_2^{(1),\{4\}} = (1)(1) + (1)(1) + 1 = 3$$

นั่นทำให้

$$y_1^{(2),\{3\}} = 2 + 2 + 1 = 5, \qquad y_1^{(2),\{4\}} = 3 + 3 + 1 = 7$$

จะได้

$$\frac{\partial E}{\partial w_{1,1}^{(1)}}(1, 1, ..., 1) = \frac{1}{2}[(5 - 1) + 7] = 5.5$$

ดังนั้น

$$w_{1,1}^{(1)}(1) = 1 - 0.01(5.5) = 1 - 0.055 = 0.945$$

ในทำนองเดียวกันจะได้ว่า

$$w_{1,2}^{(1)}(1) = 1 - 0.01(5.5) = 0.945, \qquad b_1^{(1)}(1) = 1 - 0.01(9) = 0.91$$

$$w_{2,1}^{(1)}(1) = 1 - 0.01(5.5) = 0.945, \qquad w_{2,2}^{(1)}(1) = 1 - 0.01(5.5) = 0.945, \qquad b_2^{(1)}(1) = 1 - 0.01(9) = 0.91$$

$$w_{1,1}^{(2)}(1) = 1 - 0.01(20) = 0.8, \qquad w_{1,2}^{(2)}(1) = 1 - 0.01(20) = 0.8, \qquad b_1^{(2)}(1) = 1 - 0.01(9) = 0.91$$

สำหรับการเรียนรู้ครั้งที่ 25 จะได้

$$w_{1,1}^{(1)}[25] = 0.835 - 0.01(0.004) = 0.835$$

$$w_{1,2}^{(1)}[25] = 0.835 - 0.01(0.004) = 0.835$$

$$b_{1}^{(1)}[25] = 0.731 - 0.01(0.006) = 0.731$$

$$w_{2,1}^{(1)}[25] = 0.835 - 0.01(0.004) = 0.835$$

$$w_{2,2}^{(1)}[25] = 0.835 - 0.01(0.004) = 0.835$$

$$b_{2}^{(1)}[25] = 0.731 - 0.01(0.006) = 0.731$$

$$w_{1,1}^{(2)}[25] = 0.031 - 0.01(0.35) \quad = 0.027$$

$$w_{1,2}^{(2)}[25] = 0.031 - 0.01(0.35) \quad = 0.027$$

$$b_{1}^{(2)}[25] = 0.501 - 0.01(0.196) = 0.499$$

เมื่อทดสอบน้ำหนักและไบแอสในการเรียนรู้ครั้งที่ 25 กับข้อมูลชุดที่ 1 จะได้

$$y_1^{(1),\{1\}} = 0.835(0) + 0.835(0) + 0.731 = 0.731, \qquad y_2^{(1),\{1\}} = 0.835(0) + 0.835(0) + 0.731 = 0.731$$

$$y_1^{(2),\{1\}} = 0.027(0.731) + 0.027(0.731) + 0.449 = 0.485$$

ในทำนองเดียวกันผลลัพธ์ที่ได้จากการทดสอบเป็นไปดังตารางที่ 2.7

| $x_1$ | $x_2$ | $y_1^{(1)}$ | $y_2^{(1)}$ | $y_1^{(2)}$ | $t$ |
|---|---|---|---|---|---|
| 0 | 0 | 0.731 | 0.731 | 0.485 | 0 |
| 0 | 1 | 1.566 | 1.566 | 0.534 | 1 |
| 1 | 0 | 1.566 | 1.566 | 0.534 | 1 |
| 1 | 1 | 2.401 | 2.401 | 0.579 | 0 |

ตารางที่ 2.7: ผลลัพธ์ในเพอร์เซปตรอนในการเรียนรู้ครั้งที่ 25

หากคำนวณผลลัพธ์ที่ได้จากการทดสอบน้ำหนักและไบแอสก่อนการเรียนรู้จะได้ผลลัพธ์ดังตาราง 2.8

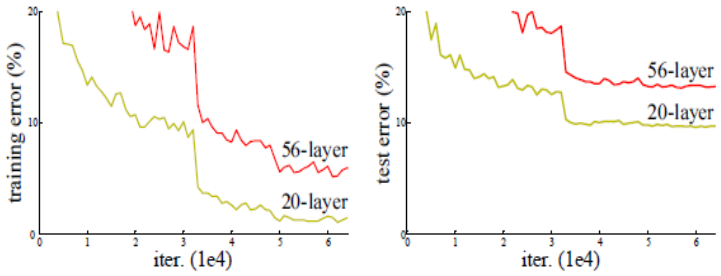
| $x_1$ | $x_2$ | $y_1^{(1)}$ | $y_2^{(1)}$ | $y_1^{(2)}$ | $t$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 3 | 0 |
| 0 | 1 | 2 | 2 | 5 | 1 |
| 1 | 0 | 2 | 2 | 5 | 1 |
| 1 | 1 | 3 | 3 | 7 | 0 |

ตารางที่ 2.8: ผลลัพธ์ในเพอร์เซปตรอนที่ไม่ผ่านการเรียนรู้

เมื่อเปรียบเทียบค่าที่โมเดลทำนายกับค่าจริง $t$ สังเกตได้ว่าโมเดลที่ผ่านการเรียนรู้ด้วย gradient descent สามารถทำนายค่าได้ใกล้เคียงกับค่าจริงมากกว่าโมเดลที่ไม่ผ่านการเรียนรู้

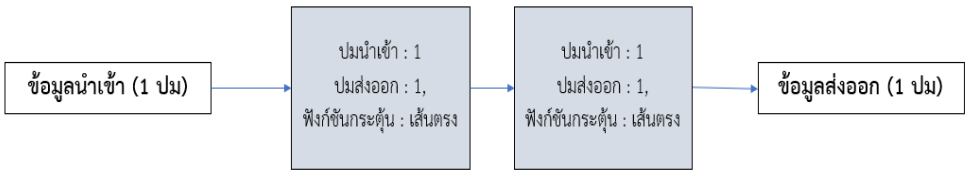
### 2.2.6. **โครงข่าย Residual**

บางครั้งเราจำเป็นต้องใช้โครงข่ายที่มีชั้นเป็นจำนวนมากในการทำงาน เช่น โครงข่ายประสาทหลายชั้นจำนวน 100 ชั้น โครงข่ายประสาทหลายชั้นจำนวน 1000 ชั้น โดย [6] กล่าวว่าโครงข่ายประสาทเทียมที่มีจำนวนชั้นมากจะเสี่ยงต่อการเกิดปัญหา vanishing gradient กล่าวคือค่า gradient ที่ได้จากการแพร่กลับในขั้นตอน gradient descent มีค่าเข้าใกล้ 0 ซึ่งขัดขวางการลู่เข้าสู่จุดต่ำสุดของกระบวนการดังกล่าว นอกจากปัญหา vanishing gradient ในบางครั้งการเพิ่มจำนวนชั้นในโครงข่ายประสาทเทียมอาจทำให้ประสิทธิภาพในการทำนายข้อมูลของโมเดลลดลงดังแสดงในรูปที่ 2.11



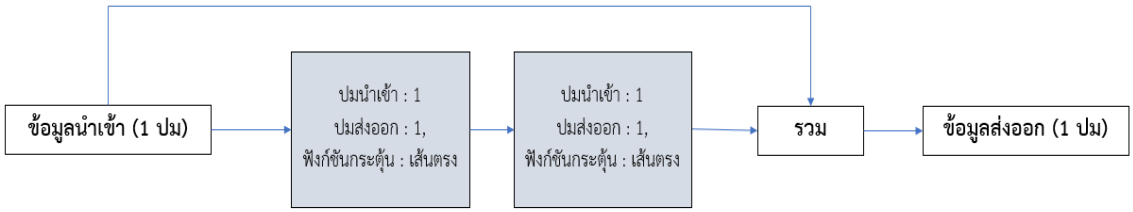

รูปที่ 2.11: การเปรียบเทียบประสิทธิภาพการทำนายข้อมูล CIFAR-10 ด้วยโครงข่ายประสาทเทียม 20 ชั้น กับ 56 ชั้น

ปัญหาที่กล่าวมาข้างต้นสามารถแก้ได้ด้วยการปรับปรุงโครงสร้างโครงข่ายประสาทเทียม โดยหากเรามีโครงข่ายประสาทเทียมซึ่งเขียนให้อยู่ในรูปของกล่องดังแสดงในรูปที่ 2.12




รูปที่ 2.12: โครงข่ายประสาทเทียมในรูปกล่อง

จากรูปข้อมูลส่งออกในชั้นที่สองมีขนาดเท่ากันกับข้อมูลนำเข้า $\mathbf{y^{(2)}} = (y_1^{(2)}), \mathbf{x} = (x_1) \in \mathbb{R}$ เราสามารถปรับปรุงโครงข่ายให้มีลักษณะดังแสดงในรูปที่ 2.13

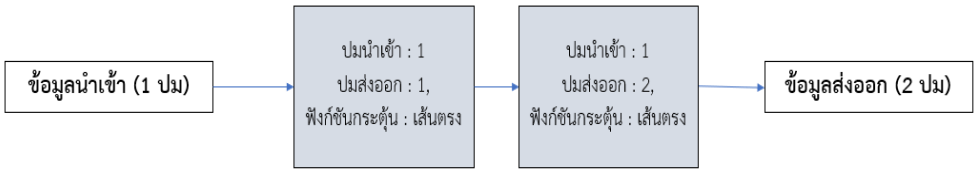



รูปที่ 2.13: โครงข่ายประสาทเทียมที่เพิ่มการรวมกันของค่าในภายหลัง

กำหนดให้ข้อมูลส่งออกคือ $\mathbf{o} = (o_1) \in \mathbb{R}$ จากรูปจะได้ว่าข้อมูลส่งออกสามารถคำนวณค่าได้จาก

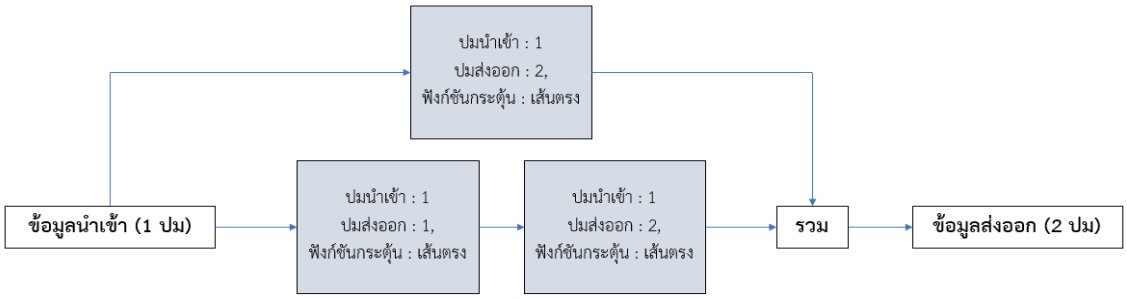
$$o_1 = y_1^{(2)} + x_1 \tag{2.13}$$

ในกรณีที่ขนาดข้อมูลส่งออกในชั้นสุดท้ายไม่เท่ากับขนาดข้อมูลนำเข้า $\mathbf{y^{(2)}} = (y_1^{(2)}, y_2^{(2)}) \in \mathbb{R}^2$ แต่ $\mathbf{x} = (x_1) \in \mathbb{R}$ ดังแสดงในรูปที่ 2.14




รูปที่ 2.14: โครงข่ายประสาทเทียมในรูปกล่องที่มีขนาดข้อมูลส่งออกไม่เท่ากับข้อมูลนำเข้า

เราจะเพิ่มโครงข่ายประสาทเทียมหนึ่งโครงข่ายที่รับข้อมูลนำเข้าและให้ข้อมูลส่งออกมีขนาดเท่ากับข้อมูลส่งออกในโครงข่ายหลักดังแสดงในรูปที่ 2.15




รูปที่ 2.15: โครงข่ายประสาทเทียมที่เพิ่มการรวมกันของค่าในภายหลัง

กำหนดให้ข้อมูลส่งออกคือ $\mathbf{o} = (o_1, o_2) \in \mathbb{R}^2$ การคำนวณข้อมูลส่งออกเป็นไปตามสมการดังนี้

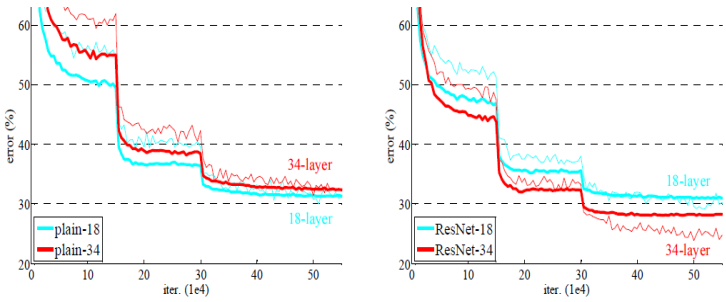
$$o_1 = y_1^{(2)} + y_1^* = y_1^{(2)} + w_{1,1}^* x_1 + b_1^* \tag{2.14}$$

$$o_2 = y_2^{(2)} + y_2^* = y_1^{(2)} + w_{2,1}^* x_1 + b_2^* \tag{2.15}$$

โดย $\mathbf{y}^* = (y_1^*, y_2^*) \in \mathbb{R}^2$ คือผลลัพธ์ที่ได้จากโครงข่ายประสาทเทียมเพิ่มเติม $w_{1,1}^*, w_{2,1}^* x_1, b_1^*, b_2^*$ คือน้ำหนักและไบแอสในโครงข่ายประสาทเพิ่มเติม

การนำค่านำเข้ารวมกับค่าส่งออกเรียกว่า residual โดยการเพิ่มโครงข่าย residual เข้าไปในโครงข่ายประสาทเทียมที่มีจำนวนชั้นเป็นจำนวนมากสามารถแก้ปัญหา vanishing gradient ได้ และสามารถเพิ่มประสิทธิภาพในการเรียนรู้ของโมเดล [5]

รูปที่ 2.16 แสดงการเปรียบเทียบประสิทธิภาพของโมเดลที่มีจำนวน 18 ชั้นกับ 34 ชั้น โดยเปรียบเทียบระหว่างโมเดลปกติกับโมเดลที่เพิ่มโครงข่าย residual




รูปที่ 2.16: เปรียบเทียบประสิทธิภาพโครงข่าย 18 ชั้นกับ 34 ชั้นของโมเดลปกติกับโมเดลเพิ่มโครงข่าย residual บนข้อมูล ImageNet

จากรูปจะเห็นได้ว่าเมื่อเพิ่มโครงข่าย residual ในโมเดล 34 ชั้น ทำให้โมเดลมีประสิทธิภาพสูงกว่าโมเดล 18 ชั้น

## 2.3 โครงข่ายประสาทคอนโวลูชัน (Convolutional neural network)

จากที่กล่าวมาในหัวข้อ 2.2 โครงข่ายประสาทแบบป้อนไปหน้าสามารถนำไปใช้งานกับข้อมูลประเภทรูปภาพ ซึ่งสามารถนำไปประยุกต์ใช้ในงานประเภทการประมวลผลภาพและวิสัยทัศน์ของคอมพิวเตอร์ได้ แต่อย่างไรก็ตามยังมีโครงข่ายประสาทคอนโวลูชันที่ออกแบบมาเพื่อใช้งานกับข้อมูลประเภทรูปภาพโดยเฉพาะ โดยจะเรียนรู้ข้อมูลพื้นที่ในแต่ละพิกเซลแล้วนำข้อมูลที่ได้มาใช้ในการทำนายผล ซึ่งการเรียนรู้ของโครงข่ายคอนโวลูชันนี้จึงเหมาะสมกับงานประเภทรูปภาพมากกว่าโครงข่ายประสาทแบบป้อนไปหน้า

ในหัวข้อนี้จะกล่าวถึงตัวดำเนินการคอนโวลูชัน (Convolution operator) ชั้นคอนโวลูชัน (Convolution layer) ชั้นกรองค่าสูงสุด (Max pooling layer) ชั้นกรองค่าเฉลี่ย (Average pooling layer) และไฮเปอร์พารามิเตอร์ (Hyperparameter)

เนื่องจากการสอน/ปรับน้ำหนักและไบแอสของโครงข่ายประสาทคอนโวลูชันใช้แนวคิดลักษณะเดียวกันกับโครงข่ายประสาทแบบป้อนไปหน้า ในหัวข้อนี้เราจึงไม่กล่าวถึงการสอนและปรับไบแอสของโมเดลนี้

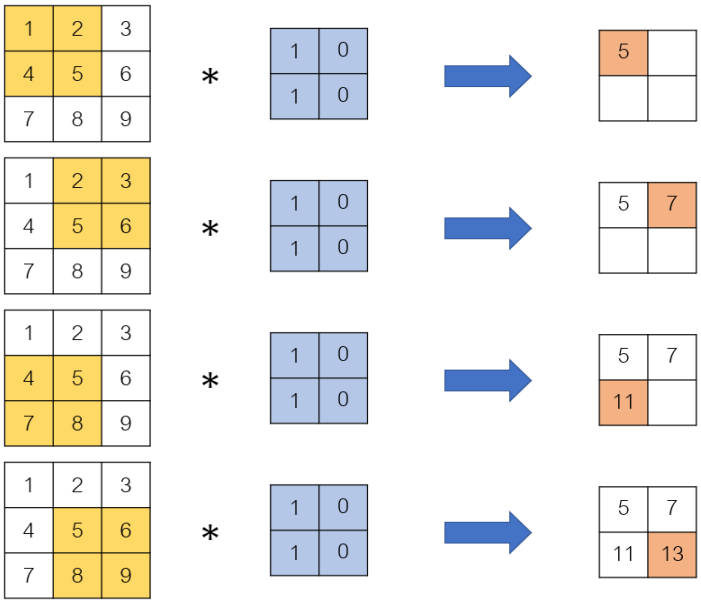
### 2.3.1. ตัวดำเนินการคอนโวลูชัน (Convolution operator)

เพื่อความง่ายในการอธิบายเราพิจารณากรณีที่ข้อมูลนำเข้าและตัวกรอง (filter) เป็นสองมิติ ให้ $X = (x_{i,j}) \in \mathbb{R}^{m_1 \times m_2}$ เป็นข้อมูลนำเข้า สำหรับตัวกรอง $K = (k_{i,j}) \in \mathbb{R}^{h_1 \times h_2}$ เมื่อ $1 \leq h_1 \leq m_1, 1 \leq h_2 \leq m_2$ จะได้ผลลัพธ์จากการดำเนินการคอนโวลูชันคือเมทริกซ์ $(X * K) \in \mathbb{R}^{m_1 - h_1 + 1 \times m_2 - h_2 + 1}$ โดยค่า ณ ตำแหน่ง $(i, j)$ ของเมทริกซ์สามารถคำนวณได้จาก

$$(X * K)_{i,j} = \sum_{u=1}^{h_1} \sum_{v=1}^{h_2} k_{u,v}(x_{i+u-1,j+v-1}) \tag{2.16}$$

**ตัวอย่าง 2.9** กำหนดให้ข้อมูลนำเข้าคือ $X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \in \mathbb{R}^{3 \times 3}$ และตัวกรองคือ $K = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \in \mathbb{R}^{2 \times 2}$

การทำคอนโวลูชันคือการนำตัวกรองไปทาบกับอินพุตแล้วคำนวณผลคูณจุด (dot product) ในตำแหน่งที่ตัวเลขตรงกัน โดยเมื่อคำนวณผลลัพธ์ได้แล้วจะขยับตัวกรองที่ทาบไปทีละหนึ่งหน่วยซึ่งสามารถแสดงได้ดังแสดงในรูปที่ 2.17




รูปที่ 2.17: ตัวอย่างการคำนวณค่าในตัวดำเนินการคอนโวลูชัน

### 2.3.2. ชั้นคอนโวลูชัน (Convolution layer)

ในชั้นคอนโวลูชันข้อมูลนำเข้ากับตัวกรองจะเป็นสามมิติ โดยข้อมูลนำเข้าสามมิติจะสามารถเขียนให้อยู่ในรูปข้อมูลนำเข้าสองมิติหลายๆข้อมูลได้ และตัวกรองสามมิติก็สามารถเขียนให้อยู่ในรูปตัวกรองสองมิติหลายๆตัวได้เช่นกัน ชั้นคอนโวลูชันจะนำตัวข้อมูลนำเข้าสองมิติกับตัวกรองสองมิติในตำแหน่งเดียวกันมาดำเนินการคอนโวลูชัน แล้วนำผลลัพธ์ที่ได้จากการดำเนินการคอนโวลูชันทุกตำแหน่งมารวมกันแล้วบวกกับเมทริกซ์ไบแอส โดยในส่วนนี้จะมีตัวอย่างการทำคอนโวลูชันที่ประกอบไปด้วยตัวกรองสามมิติหนึ่งตัว และตัวกรองสามมิติหลายตัว

**สำหรับชั้นคอนโวลูชันที่มีตัวกรองสามมิติหนึ่งตัว**

กำหนดให้ $\mathbf{X} = (X_1, X_2, ..., X_{m_3}) \in \mathbb{R}^{m_1 \times m_2 \times m_3}$ เป็นข้อมูลนำเข้า โดย $X_i$ เป็นเมทริกซ์ขนาด $m_1 \times m_2$

$\mathbf{K} = (K_1, K_2, ..., K_{m_3}) \in \mathbb{R}^{h_1 \times h_2 \times m_3}$ เป็นตัวกรอง โดย $K_i$ คือเมทริกซ์ขนาด $h_1 \times h_2$

$B \in \mathbb{R}^{m_1 - h_1 + 1 \times m_2 - h_2 + 1}$ เป็นเมทริกซ์ไบแอส

ให้ $Z \in \mathbb{R}^{m_1 - h_1 + 1 \times m_2 - h_2 + 1}$ เป็นการนำผลลัพธ์จากคอนโวลูชันมาบวกกับเมทริกซ์ไบแอสโดย

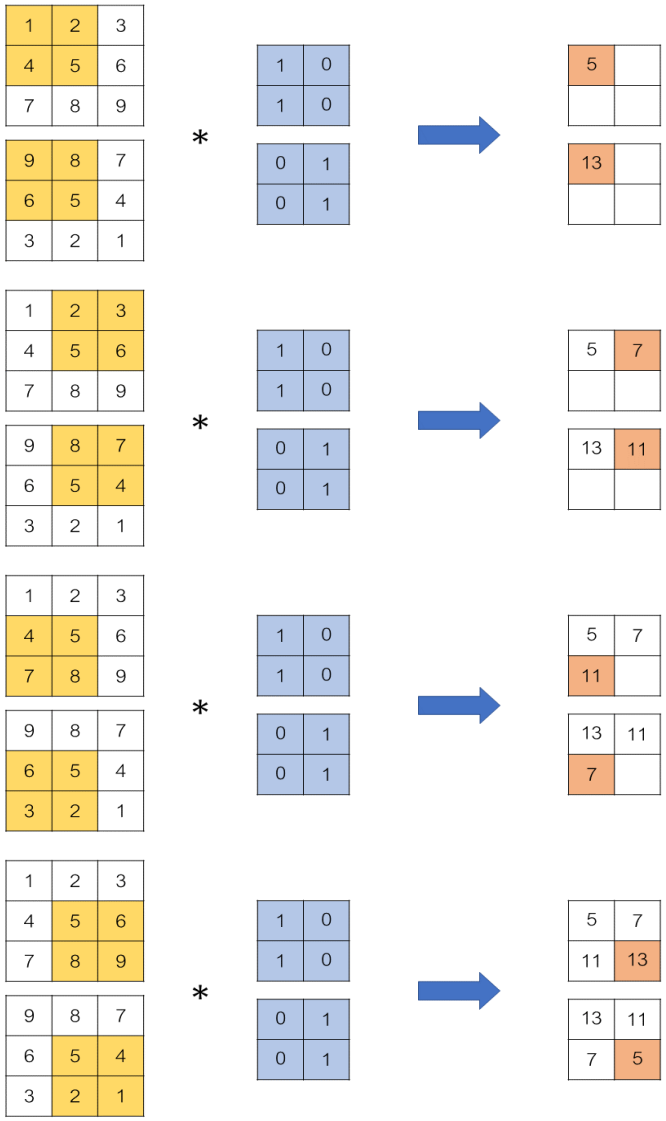
$$Z = \sum_{r=1}^{m_3} (X_r * K_r) + B \tag{2.17}$$

สำหรับฟังก์ชันกระตุ้น $f : \mathbb{R} \to \mathbb{R}$ จะได้ว่าข้อมูลส่งออกของชั้นคอนโวลูชันคือ $Y \in \mathbb{R}^{m_1 - h_1 + 1 \times m_2 - h_2 + 1}$ โดย

$$Y = \begin{bmatrix} f(z_{1,1}) & \cdots & f(z_{1,m_2-h_2+1}) \\ \vdots & \ddots & \vdots \\ f(z_{m_1-h_1+1,1}) & \cdots & f(z_{m_1-h_1+1,m_2-h_2+1}) \end{bmatrix} \tag{2.18}$$

**ตัวอย่าง 2.10** กำหนดให้ข้อมูลนำเข้าคือ $X = \left[\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}\right] \in \mathbb{R}^{3\times3\times2}$ ตัวกรองคือ $K = [\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}] \in \mathbb{R}^{2\times2\times2}$ ไบแอสคือ $\begin{bmatrix} 5 & 5 \end{bmatrix} \in \mathbb{R}^{2\times2}$ และฟังก์ชันกระตุ้นคือฟังก์ชัน ReLU การคำนวณค่าเป็นไปตามขั้นตอนดังนี้

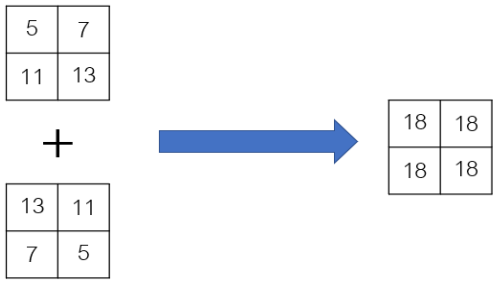
1. คำนวณผลลัพธ์จากการทำคอนโวลูชัน $(X_r * K_r)$ เมื่อ $r = 1, 2$ ซึ่งเป็นไปดังแสดงในรูปที่ 2.18




รูปที่ 2.18: การคำนวณคอนโวลูชัน

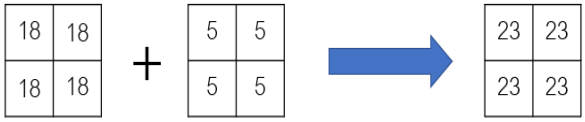
โดยเมทริกซ์ขวาบนคือเมทริกซ์ $(X_1 * K_1)$ และเมทริกซ์ขวาล่างคือเมทริกซ์ $(X_2 * K_2)$

2. คำนวณค่า $\sum_{r=1}^{2}(X_r * K_r)$ ดังแสดงในรูปที่ 2.19




รูปที่ 2.19: การรวมผลลัพธ์จากการทำคอนโวลูชัน

3. นำผลลัพธ์ที่ได้ในข้อ 2. บวกไบแอส $B$ ดังแสดงในรูปที่ 2.20 ได้ผลลัพธ์เมทริกซ์ $Z$




รูปที่ 2.20: การรวมค่าที่ได้ในข้อ 2. กับค่าไบแอส

4. นำสมาชิกทุกตัวในเมทริกซ์ $Z$ ป้อนเข้าสู่ฟังก์ชันกระตุ้น โดยเนื่องจากค่าทุกค่าใน $Z$ มีค่ามากกว่า 0 จะได้ว่าสำหรับฟังก์ชัน ReLU $f(x) = x$ ได้ผลลัพธ์ดังแสดงในรูปที่ 2.21


รูปที่ 2.21: การนำค่าทุกค่าใน $Z$ ป้อนเข้าสู่ฟังก์ชันกระตุ้น

**ชั้นคอนโวลูชันที่มีตัวกรองสามมิติหลายตัวโดยแต่ละตัวมีขนาดที่เท่ากัน**

กำหนดให้ $\mathbf{X} = (X_1, X_2, ..., X_{m_3}) \in \mathbb{R}^{m_1 \times m_2 \times m_3}$ เป็นข้อมูลนำเข้า โดย $X_i$ เป็นเมทริกซ์ขนาด $m_1 \times m_2$

$\mathbf{K_a} = (K_{a,1}, K_{a,2}, ..., K_{a,m_3}) \in \mathbb{R}^{h_1 \times h_2 \times m_3}$ เป็นตัวกรองตัวที่ $a$ โดย $K_{a,i}$ คือเมทริกซ์ขนาด $h_1 \times h_2$

$B_a \in \mathbb{R}^{m_1 - h_1 + 1 \times m_2 - h_2 + 1}$ เมทริกซ์ไบแอสตัวที่ $a$

จะได้ว่าค่า $Z_a \in \mathbb{R}^{m_1 - h_1 + 1 \times m_2 - h_2 + 1}$ สามารถคำนวณได้ดังสมการ

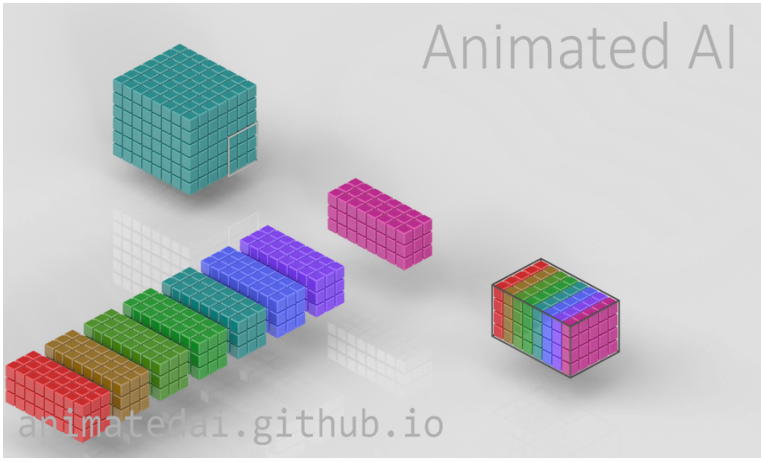
$$Z_a = \sum_{r=1}^{m_3}(X_r * K_{a,r}) + B_a \tag{2.19}$$

ให้ $f : \mathbb{R} \to \mathbb{R}$ เป็นฟังก์ชันกระตุ้น จะได้ข้อมูลส่งออกที่ $a$ คือ $Y_a \in \mathbb{R}^{m_1-h_1+1 \times m_2-h_2+1}$ โดย

$$Y_a = \begin{bmatrix} f(z_{1,1,a}) & \cdots & f(z_{1,m_2-h_2+1,a}) \\ \vdots & \ddots & \vdots \\ f(z_{m_1-h_1+1,1,a}) & \cdots & f(z_{m_1-h_1+1,m_2-h_2+1,a}) \end{bmatrix} \tag{2.20}$$

หากมีตัวกรองจำนวน $n$ ตัว จะได้ค่าส่งออกจากชั้นคอนโวลูชันคือ $\mathbf{Y} = (Y_1, Y_2, ..., Y_n) \in \mathbb{R}^{m_1-h_1+1 \times m_2-h_2+1 \times n}$ โดยค่า $Y_a$ สามารถคำนวณได้โดยใช้สมการที่ 2.19 และ 2.20 เมื่อ $a = 1, 2, ..., n$

รูปที่ 2.22 แสดงอนิเมชั่นการดำเนินการคอนโวลูชันในแต่ละตัวกรอง โดยผลลัพท์ขวาล่างที่ได้คือเมทริกซ์ $[\sum_{r=1}^{8}(X_r * K_{1,r}), \sum_{r=1}^{8}(X_r * K_{2,r}), ..., \sum_{r=1}^{8}(X_r * K_{8,r})]$ ซึ่งเป็นเมทริกซ์ 3 มิติที่เกิดจากการนำผลลัพธ์ที่ได้จากการทำคอนโวลูชันในแต่ละตัวกรองมาซ้อนกัน




รูปที่ 2.22: อนิเมชั่นแสดงการการดำเนินการคอนโวลูชัน รูปภาพนำมาจาก [2]

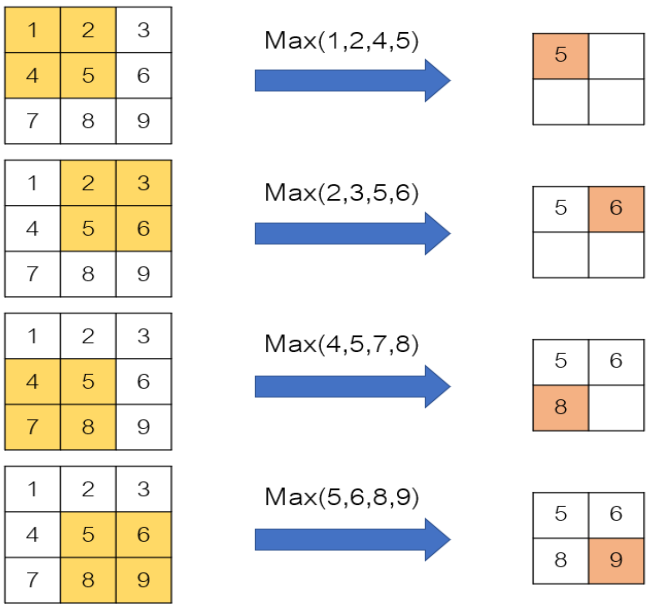
เมื่อนำเมทริกซ์ดังกล่าวมาบวกไบแอสที่ซ้อนกัน $[B_1, B_2, ..., B_8]$ และนำสมาชิกทุกตัวป้อนเข้าสู่ฟังก์ชันกระตุ้น จะได้ค่าส่งออกเช่นเดียวกันกับการใช้วิธีในสมการที่ 2.19 และ 2.20 เพื่อคำนวณหา $\mathbf{Y} = (Y_1, Y_2, ..., Y_8)$

### 2.3.3. ชั้นกรองค่าสูงสุด (Max pooling layer)

เพื่อความง่ายในการอธิบายเราพิจารณากรณีข้อมูลนำเข้าเป็นภาพสองมิติก่อน ให้ $X = (x_{i,j}) \in \mathbb{R}^{m_1 \times m_2}$ เป็นข้อมูลนำเข้า สำหรับตัวกรองค่าสูงสุดขนาด $h_1 \times h_2$ จะได้ว่าผลลัพธ์จากการนำข้อมูลนำเข้ามาผ่านตัวกรองค่าสูงสุด คือ $Y = (y_{i,j}) \in \mathbb{R}^{m_1 - h_1 + 1 \times m_2 - h_2 + 1}$ โดยค่าของ $Y$ ณ ตำแหน่ง $(i, j)$ คำนวณได้จาก

$$y_{i,j} = \max\{x_{i+u-1,j+v-1} | 1 \leq u \leq h_1, 1 \leq v \leq h_2\} \tag{2.21}$$

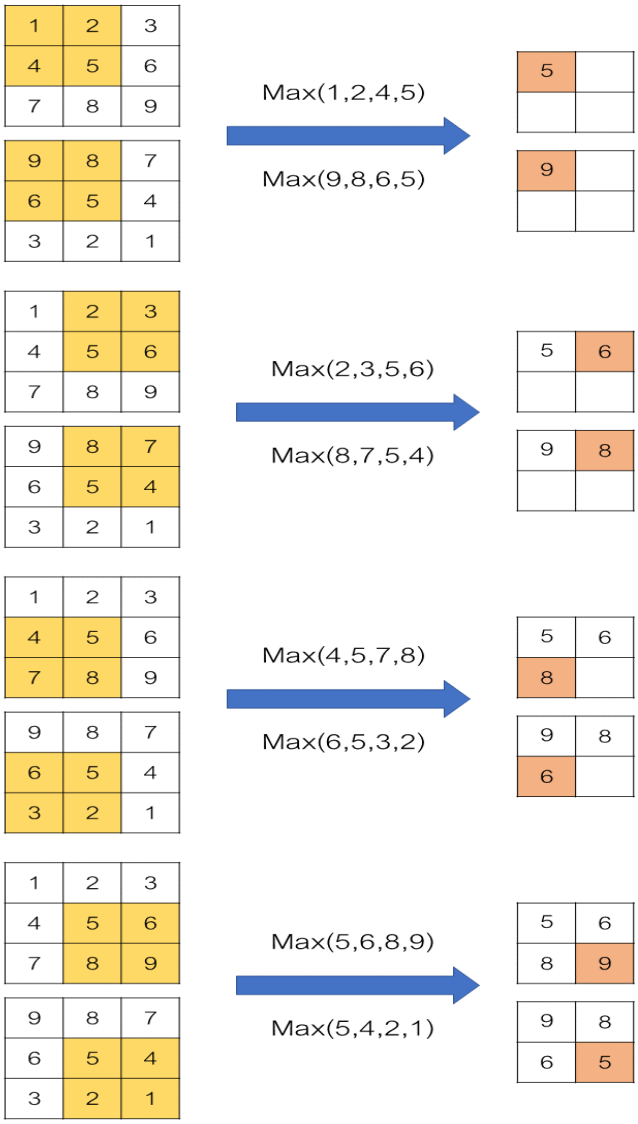
**ตัวอย่าง 2.11** กำหนดให้ข้อมูลนำเข้าคือ $X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \in \mathbb{R}^{3 \times 3}$ สำหรับตัวกรองค่าสูงสุดที่มีขนาด $2 \times 2$ ผลลัพธ์ที่ได้จากชั้นกรองค่าสูงสุดสามารถคำนวณได้ดังแสดงในรูปที่ 2.23




รูปที่ 2.23: ตัวอย่างการคำนวณค่าในชั้นกรองค่าสูงสุดสำหรับข้อมูลนำเข้าสองมิติ

เนื่องจากในโครงข่ายประสาทคอนโวลูชันข้อมูลนำเข้าเป็นสามมิติ ให้ $\mathbf{X} = (X_1, X_2, ..., X_{m_3}) \in \mathbb{R}^{m_1 \times m_2 \times m_3}$ เป็นข้อมูลนำเข้า สำหรับตัวกรองค่าสูงสุดขนาด $h_1 \times h_2$ จะได้ว่าผลลัพธ์จากการนำข้อมูลนำเข้ามาผ่านตัวกรองค่าสูงสุดคือ $\mathbf{Y} = (Y_1, Y_2, ..., Y_{m_3}) \in \mathbb{R}^{m_1 - h_1 + 1 \times m_2 - h_2 + 1 \times m_3}$ โดย $Y_i$ คือผลลัพธ์ที่ได้จากการนำ $X_i$ มาผ่านตัวกรองค่าสูงสุด เมื่อ $i = 1, 2, ..., m_3$

**ตัวอย่าง 2.12** กำหนดให้ข้อมูลนำเข้าคือ $X = \left[\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}\right] \in \mathbb{R}^{3 \times 3 \times 2}$ สำหรับตัวกรองค่าสูงสุดที่มีขนาด $2 \times 2$ ผลลัพธ์ที่ได้จากชั้นกรองค่าสูงสุดสามารถคำนวณได้ดังแสดงในรูปที่ 2.24



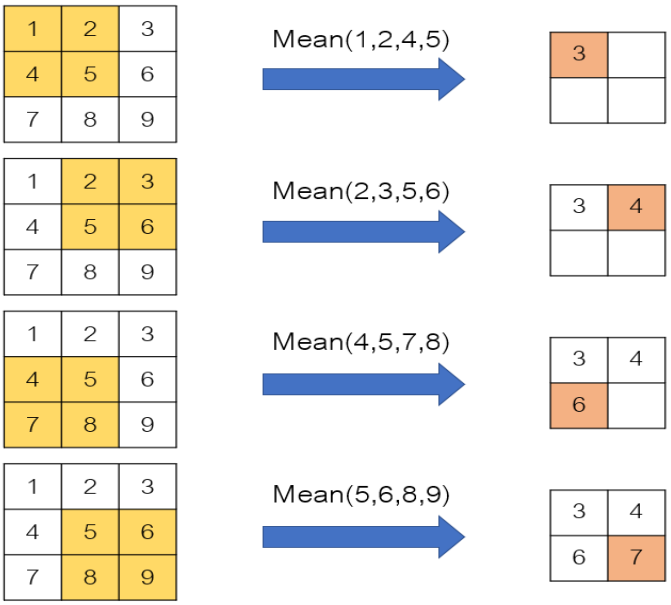

รูปที่ 2.24: ตัวอย่างการคำนวณค่าในชั้นกรองค่าสูงสุดสำหรับข้อมูลนำเข้าสามมิติ

### 2.3.4. ชั้นกรองค่าเฉลี่ย (Average pooling layer)

ชั้นกรองค่าเฉลี่ยใช้หลักการเดียวกันกับชั้นกรองค่าสูงสุดแต่ใช้การเฉลี่ยแทนการหาค่าสูงสุด พิจารณากรณีข้อมูลนำเข้าเป็นสองมิติ ให้ $X = (x_{i,j}) \in \mathbb{R}^{m_1 \times m_2}$ เป็นข้อมูลนำเข้า สำหรับตัวกรองค่าเฉลี่ยขนาด $h_1 \times h_2$ จะได้ว่าผลลัพธ์จากการนำข้อมูลนำเข้ามาผ่านตัวกรองค่าเฉลี่ยคือ $Y = (y_{i,j}) \in \mathbb{R}^{m_1 - h_1 + 1 \times m_2 - h_2 + 1}$ โดยค่าของ $Y$ ณ ตำแหน่ง $(i, j)$ คำนวณได้จาก

$$y_{i,j} = \text{mean}\{x_{i+u-1, j+v-1} | 1 \leq u \leq h_1, 1 \leq v \leq h_2\} \tag{2.22}$$

**ตัวอย่าง 2.13** กำหนดให้ข้อมูลนำเข้าคือ $X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \in \mathbb{R}^{3 \times 3}$ สำหรับตัวกรองค่าเฉลี่ยที่มีขนาด $2 \times 2$ ผลลัพธ์ที่ได้จากชั้นกรองค่าเฉลี่ยสามารถคำนวณได้ดังแสดงในรูปที่ 2.25



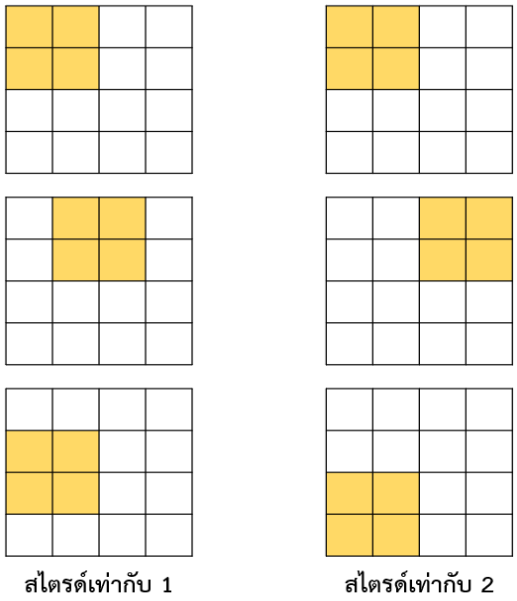

รูปที่ 2.25: ตัวอย่างการคำนวณค่าในชั้นกรองค่าเฉลี่ยสำหรับข้อมูลนำเข้าสองมิติ

สำหรับข้อมูลนำเข้าที่เป็นสามมิติ เช่นเดียวกันกับตัวกรองค่าสูงสุด ให้ $\mathbf{X} = (X_1, X_2, ..., X_{m_3}) \in \mathbb{R}^{m_1 \times m_2 \times m_3}$ และตัวกรองค่าเฉลี่ยขนาด $h_1 \times h_2$ จะได้ว่าผลลัพธ์จากการนำข้อมูลนำเข้ามาผ่านตัวกรองค่าเฉลี่ยคือ $\mathbf{Y} = (Y_1, Y_2, ..., Y_{m_3}) \in \mathbb{R}^{m_1 - h_1 + 1 \times m_2 - h_2 + 1 \times m_3}$ โดย $Y_i$ คือผลลัพธ์จากการนำ $X_i$ มาผ่านตัวกรองค่าเฉลี่ย เมื่อ $i = 1, 2, ..., m_3$

### 2.3.5. ไฮเปอร์พารามิเตอร์ (Hyperparameters)

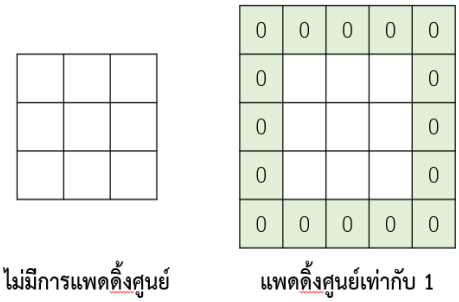
ในการสร้างโครงข่ายประสาทเทียมไฮเปอร์พารามิเตอร์เป็นปัจจัยหลักในการกำหนดโครงสร้างของโครงข่ายประสาทเทียม โดยในโครงข่ายประสาทแบบป้อนไปหน้าไฮเปอร์พารามิเตอร์คือจำนวนข้อมูลนำเข้า จำนวนข้อมูลส่งออก และฟังก์ชันกระตุ้นในแต่ละชั้น สำหรับโครงข่ายประสาทคอนโวลูชันไฮเปอร์พารามิเตอร์คือ ขนาดตัวกรอง จำนวนตัวกรอง สไตรด์ (Stride) และการแพดดิ้งศูนย์ (Zero padding) ในแต่ละชั้น โดยในส่วนนี้จะกล่าวถึงแนวคิดพื้นฐานและตัวอย่างของสไตรด์และแพดดิ้ง

สไตรด์เป็นไฮเปอร์พารามิเตอร์สำหรับควบคุมการเลื่อนของตัวกรองในโครงข่ายประสาทคอนโวลูชัน โดยในเนื้อหาก่อนหน้าจะเป็นการพูดถึงกรณีเริ่มต้นที่มีค่าสไตรด์เท่ากับ 1 คือขยับตัวกรองทีละหนึ่งหน่วย รูปที่ 2.26 แสดงให้เห็นถึงความแตกต่างในการเลื่อนตัวกรองระหว่างสไตรด์เท่ากับ 1 กับสไตรด์เท่ากับ 2 ในกรณีที่ข้อมูลนำเข้ามีขนาด $4 \times 4$ และตัวกรองมีขนาด $2 \times 2$




รูปที่ 2.26: การขยับของตัวกรองในกรณีสไตรด์เท่ากับ 1 และสไตรด์เท่ากับ 2

แพดดิ้งศูนย์เป็นไฮเปอร์พารามิเตอร์ที่ใช้ในการช่วยควบคุมขนาดข้อมูลส่งออกในชั้นต่างๆของโครงข่ายประสาทคอนโวลูชัน โดยแพดดิ้งจะเป็นการเติมเลข 0 ล้อมรอบเมทริกซ์ข้อมูลนำเข้าตามจำนวนที่กำหนด สำหรับเนื้อหาก่อนหน้าจะใช้กรณีเริ่มต้นที่ไม่มีการเติมแพดดิ้ง รูปที่ 2.27 แสดงให้เห็นถึงความแตกต่างของข้อมูลนำเข้าเมื่อไม่มีการเติมแพดดิ้งศูนย์ กับการเติมแพดดิ้งศูนย์ 1 ชั้น โดยอินพุตมีขนาด $3 \times 3$




รูปที่ 2.27: อินพุตเมื่อไม่มีการแพดดิ้งศูนย์และเมื่อมีการแพดดิ้งศูนย์ 1 ชั้น

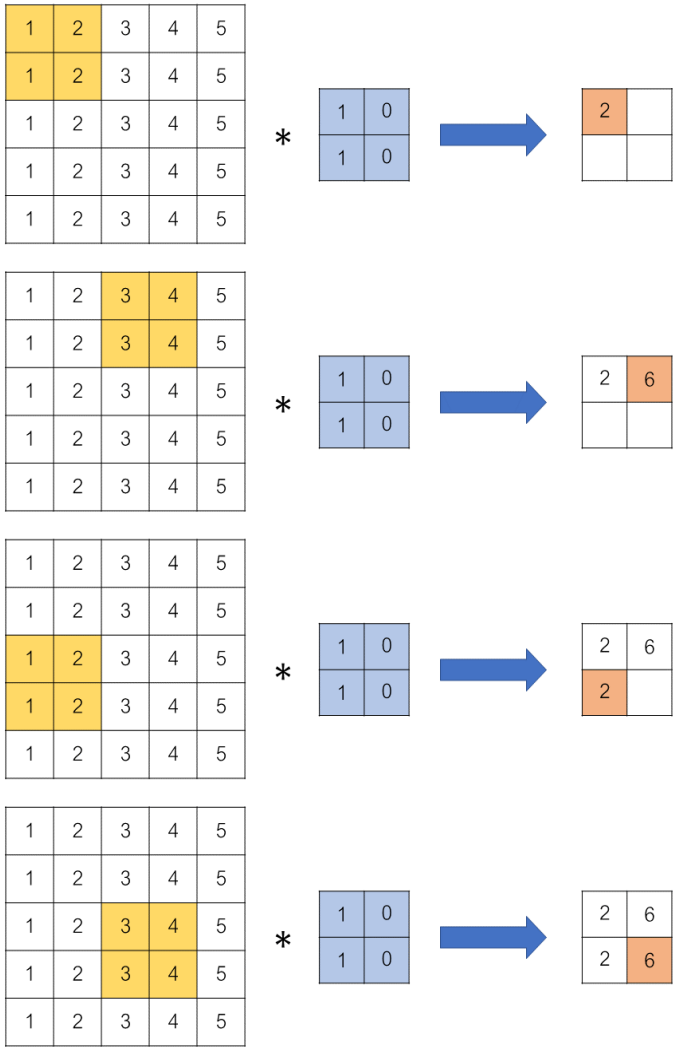
เมื่อมีการปรับไฮเปอร์พารามิเตอร์สไตรด์และแพดดิ้งจากค่าเริ่มต้น จะส่งผลให้ความกว้างและความยาวในเอาท์พุตเปลี่ยนแปลง หากอินพุตมีความกว้าง $m_1$ มีความยาว $m_2$ และตัวกรองมีความกว้าง $h_1$ มีความยาว $h_2$ ความกว้างและความยาวของเอาท์พุตสามารถคำนวณได้ดังนี้

$$output\_width = [(m_1 - h_1 + 2Padding)/Stride] + 1 \tag{2.23}$$

$$output\_height = [(m_2 - h_2 + 2Padding)/Stride] + 1 \tag{2.24}$$

เมื่อ $Padding$ แทนจำนวนชั้นของแพดดิ้งศูนย์ และ $Stride$ แทนขนาดสไตรด์

**ข้อสังเกต 2.14** บางครั้งการกำหนดค่าสไตรด์ที่มากกว่า 1 สามารถทำให้เกิดการสูญเสียข้อมูลในโครงข่ายประสาทคอนโวลูชันได้ เช่น หากเรามีข้อมูลนำเข้า $X = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} \in \mathbb{R}^{5\times5}$ มีตัวกรองคือ $K = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \in \mathbb{R}^{2\times2}$ หากกำหนดสไตรด์เท่ากับ 2 การคำนวณคอนโวลูชันเป็นไปดังแสดงในรูปที่ 2.28



รูปที่ 2.28: การคำนวณคอนโวลูชันที่สไตรด์เท่ากับ 2 โดยมีการสูญเสียข้อมูล

โดยจากรูปสังเกตได้ว่าข้อมูลในแถวสุดท้าย และข้อมูลในหลักสุดท้ายของข้อมูลนำเข้าไม่ถูกนำมาใช้ในการคำนวณคอนโวลูชัน จึงเกิดการสูญเสียข้อมูลจากการกำหนดค่าสไตรด์เท่ากับ 2

## 2.4 การวัดผลโมเดล

หนึ่งในการใช้งานโครงข่ายประสาทเทียมคือการนำโมเดลไปใช้งานในการทำนายข้อมูลไบนารีที่ประกอบไปด้วยกลุ่มบวกและกลุ่มลบ โดยหากโมเดลได้รับข้อมูลนำเข้าแล้วโมเดลจะทำนายว่าข้อมูลดังกล่าวเป็นบวกหรือเป็นลบ โมเดลดังกล่าวจะมีชั้นสุดท้ายเป็นเพอร์เซ็ปตรอนที่มีปมส่งออกหนึ่งปมและให้ค่าสูงออกอยู่ระหว่าง 0 ถึง 1 ซึ่งคือค่าความน่าจะเป็นในการเป็นบวก โดยทั่วไปจะกำหนดว่าหากค่าดังกล่าวมากกว่า 0.5 จะกำหนดให้โมเดลทำนายเป็นบวก และหากค่าดังกล่าวน้อยกว่า 0.5 จะกำหนดให้โมเดลทำนายเป็นลบ เรียกค่า 0.5 ดังกล่าวว่าค่าขีดขั้น (Threshold)

เมื่อกำหนดโมเดล และรวบรวมข้อมูลครบตามที่ต้องการ จะมีการแบ่งข้อมูลออกเป็นข้อมูลสำหรับสอนโมเดลและข้อมูลสำหรับทดสอบโมเดล โดยข้อมูลสำหรับสอนโมเดลใช้ในการปรับน้ำหนักและไบแอสในโมเดลเพื่อให้ทำนายว่าเป็นบวกหรือเป็นลบได้ตรงตามข้อมูลจริง และข้อมูลสำหรับทดสอบโมเดลเป็นข้อมูลที่ไว้สำหรับทดสอบว่าหากนำโมเดลดังกล่าวไปใช้งานจริง เมื่อเจอข้อมูลใหม่จะทำนายได้ดีมากน้อยเพียงใด

ในหัวข้อนี้จะกล่าวถึงการใช้ Confusion matrix ช่วยในการวัดผลโมเดล และการวัดผลด้วยกราฟ ROC (Receiver Operating Characteristic curve)

### 2.4.1. การวัดผลด้วย Confusion matrix

Confusion matrix มีลักษณะดังตารางที่ 2.9

| | | **ค่าทำนาย** | |
|---|---|---|---|
| | | **บวก** | **ลบ** |
| **ค่าจริง** | **บวก** | บวกจริง | ลบเท็จ |
| | **ลบ** | บวกเท็จ | ลบจริง |

ตารางที่ 2.9: Confusion matrix

โดย ค่าบวกจริง (True positive) คือจำนวนข้อมูลที่โมเดลทำนายว่าเป็นบวก และค่าจริงเป็นบวก

ค่าบวกเท็จ (False positive) คือจำนวนข้อมูลที่โมเดลทำนายว่าเป็นบวก แต่ค่าจริงเป็นลบ

ค่าลบจริง (True negative) คือจำนวนข้อมูลที่โมเดลทำนายว่าเป็นลบ และค่าจริงเป็นลบ

ค่าลบเท็จ (False negative) คือจำนวนข้อมูลที่โมเดลทำนายว่าเป็นลบ แต่ค่าจริงเป็นบวก

จาก Confusion matrix กำหนดให้ $TP, FP, TN, FN$ คือตัวย่อของค่าบวกจริง ค่าบวกเท็จ ค่าลบจริง ค่าลบเท็จ ตามลำดับ

**ค่าความแม่นยำ (Accuracy)** สามารถคำนวณได้จากสมการ

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.25}$$

โดยค่าความแม่นยำจะบ่งบอกถึงภาพรวมของโมเดลที่สามารถทำนายข้อมูลได้ตรงกับค่าจริงกี่เปอร์เซ็นต์

**ค่าความเที่ยง (Precision)** สามารถคำนวณได้จาก

$$Precision = \frac{TP}{TP + FP} \tag{2.26}$$

โดยค่าความเที่ยงจะใช้ในการพิจารณาว่าสิ่งที่โมเดลทำนายมาว่าเป็นบวกสามารถเชื่อถือได้มากน้อยเพียงใด

**ค่าการระลึกได้ (Recall) หรืออัตราค่าบวกจริง (True positive rate)** สามารถคำนวณได้จาก

$$Recall = \frac{TP}{TP + FN} \tag{2.27}$$

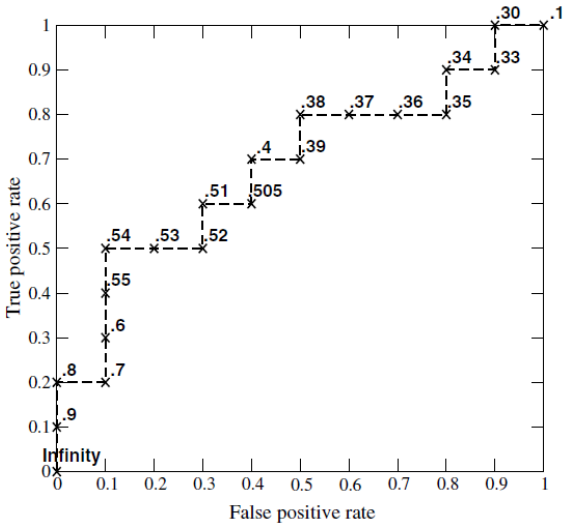
โดยค่าการระลึกได้จะใช้ในการพิจารณาว่าเมื่อโมเดลเจอข้อมูลที่เป็นบวก โมเดลจะทำนายว่าเป็นบวกได้กี่เปอร์เซ็นต์

### 2.4.2. ROC (Receiver Operating Characteristic curve)

พิจารณาค่าขีดขั้นสำหรับการแบ่งข้อมูลออกเป็นบวกและลบของโมเดล หากกำหนดค่าขีดขั้นหลายค่าตั้งแต่ 0 ถึง 1 และคำนวณ Confusion matrix สำหรับค่าแต่ละค่า ข้อมูลในคอนฟิวชันเมทริกซ์จะสามารถนำมาใช้ในการสร้างกราฟ ROC ได้ โดยหากคำนวณอัตราบวกจริง (True positive rate) ซึ่งคำนวณได้จากสมการ 2.27 และอัตราบวกเท็จ (False positive rate) ซึ่งเป็นไปตามสมการที่ 2.28

$$False\ Positive\ Rate = \frac{FP}{FP + TN} \tag{2.28}$$

เมื่อนำอัตราบวกจริงและอัตราบวกเท็จของแต่ละค่าระดับกั้นมาใช้ในการสร้างกราฟโดยแกน x เป็นอัตราบวกเท็จ และแกน y เป็นอัตราบวกจริง จะได้กราฟ ROC ซึ่งกราฟจะมีลักษณะดังแสดงในรูป 2.29 โดยค่า ณ แต่ละจุดของกราฟคือค่าขีดขั้นแต่ละระดับ




รูปที่ 2.29: กราฟ ROC

**ตัวอย่าง 2.15** กำหนดให้ข้อมูลส่งออกจากโครงข่ายประสาทเทียมคือ $o^{\{1\}}, o^{\{2\}}, o^{\{3\}}, o^{\{4\}} = 0.25, 0.4, 0.6, 0.9$ และกำหนดให้ค่าจริงคือ $t^{\{1\}}, t^{\{2\}}, t^{\{3\}}, t^{\{4\}} = 0, 0, 1, 1$ การสร้างกราฟ ROC ที่ระดับขีดขั้นเท่ากับ 0.1,0.3,0.7,1.0 ตามลำดับ สามารถทำได้ดังนี้

หากกำหนดค่าขีดขั้นเท่ากับ 0.1 จะได้ว่าข้อมูลทำนายที่มีค่ามากกว่า 0.1 เป็นกลุ่มบวก และข้อมูลที่มีค่าทำนายน้อยกว่า 0.1 เป็นกลุ่มลบ จะได้ข้อมูลทำนายเป็น 1,1,1,1 เนื่องจาก 0.25,0.4,0.6,0.9 มีค่ามากกว่า 0.1 ทั้งหมด และ confusion matrix เป็นไปตามตารางที่ 2.10

| | | ค่าทำนาย: บวก | ค่าทำนาย: ลบ |
|---|---|---|---|
| ค่าจริง: บวก | | 2 | 0 |
| ค่าจริง: ลบ | | 2 | 0 |

ตารางที่ 2.10: Confusion matrix เมื่อค่าขีดขั้นเท่ากับ 0.1

จากตาราง จะได้อัตราบวกจริงเท่ากับ $\frac{2}{2+0} = 1$ อัตราบวกเท็จเท่ากับ $\frac{2}{2+0} = 1$ ที่ค่าขีดขั้นเท่ากับ 0.1

หากกำหนดค่าขีดขั้นเท่ากับ 0.3 จะได้ว่าข้อมูลทำนายที่มีค่ามากกว่า 0.3 เป็นกลุ่มบวก และข้อมูลที่มีค่าทำนายน้อยกว่า 0.3 เป็นกลุ่มลบ จะได้ข้อมูลทำนายเป็น 0,1,1,1 เนื่องจาก 0.25 มีค่าน้อยกว่า 0.3 ในขณะที่ 0.4,0.6,0.9 มีค่ามากกว่า 0.1 และ confusion matrix เป็นไปตามตารางที่ 2.11

| | | ค่าทำนาย | |
|---|---|---|---|
| | | บวก | ลบ |
| ค่าจริง | บวก | 2 | 0 |
| | ลบ | 1 | 1 |

ตารางที่ 2.11: Confusion matrix เมื่อค่าขีดขั้นเท่ากับ 0.3

จากตาราง จะได้อัตราบวกจริงเท่ากับ $\frac{2}{2+0} = 1$ อัตราบวกเท็จเท่ากับ $\frac{1}{1+1} = 0.5$ ที่ค่าขีดขั้นเท่ากับ 0.3

หากกำหนดค่าขีดขั้นเท่ากับ 0.7 จะได้ว่าข้อมูลทำนายที่มีค่ามากกว่า 0.7 เป็นกลุ่มบวก และข้อมูลที่มีค่าทำนายน้อยกว่า 0.7 เป็นกลุ่มลบ จะได้ข้อมูลทำนายเป็น 0,0,0,1 เนื่องจาก 0.25,0.4,0.6 มีค่าน้อยกว่า 0.7 ในขณะที่ 0.9 มีค่ามากกว่า 0.7 และ confusion matrix เป็นไปตามตารางที่ 2.12

| | | ค่าทำนาย | |
|---|---|---|---|
| | | บวก | ลบ |
| ค่าจริง | บวก | 1 | 1 |
| | ลบ | 0 | 2 |

ตารางที่ 2.12: Confusion matrix เมื่อค่าขีดขั้นเท่ากับ 0.7

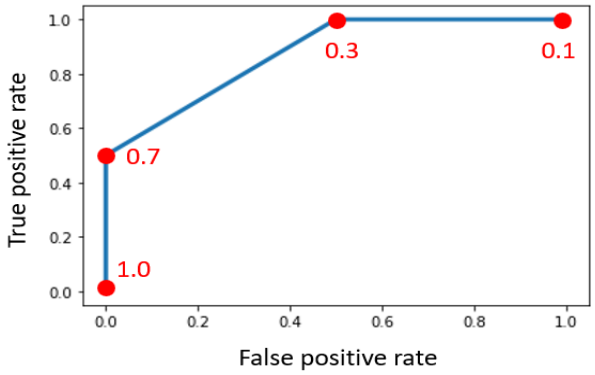
จากตาราง จะได้อัตราบวกจริงเท่ากับ $\frac{1}{1+1} = 0.5$ อัตราบวกเท็จเท่ากับ $\frac{0}{0+2} = 0$ ที่ค่าขีดขั้นเท่ากับ 0.7

หากกำหนดค่าขีดขั้นเท่ากับ 1.0 จะได้ว่าข้อมูลทำนายที่มีค่ามากกว่า 1.0 เป็นกลุ่มบวก และข้อมูลที่มีค่าทำนายน้อยกว่า 1.0 เป็นกลุ่มลบ จะได้ข้อมูลทำนายเป็น 0,0,0,0 เนื่องจาก 0.25,0.4,0.6,0.9 มีค่าน้อยกว่า 1.0 ทั้งหมด และ confusion matrix เป็นไปตามตารางที่ 2.13

| | | ค่าทำนาย | |
|---|---|---|---|
| | | บวก | ลบ |
| ค่าจริง | บวก | 0 | 2 |
| | ลบ | 0 | 2 |

ตารางที่ 2.13: Confusion matrix เมื่อค่าขีดขั้นเท่ากับ 1.0

จากตาราง จะได้อัตราบวกจริงเท่ากับ $\frac{0}{0+2} = 0$ อัตราบวกเท็จเท่ากับ $\frac{0}{0+2} = 0$ ที่ค่าขีดขั้นเท่ากับ 1.0 เมื่อพลอตอัตราบวกจริง และอัตราบวกเท็จบนค่าขีดขั้นต่างๆ จะได้กราฟดังแสดงในรูปที่ 2.30




รูปที่ 2.30: กราฟ ROC

โดยค่าสีแดงแสดงถึงค่าขีดขั้นซึ่งมีค่า 0.1,0.3,0.7,1.0 ตามลำดับ

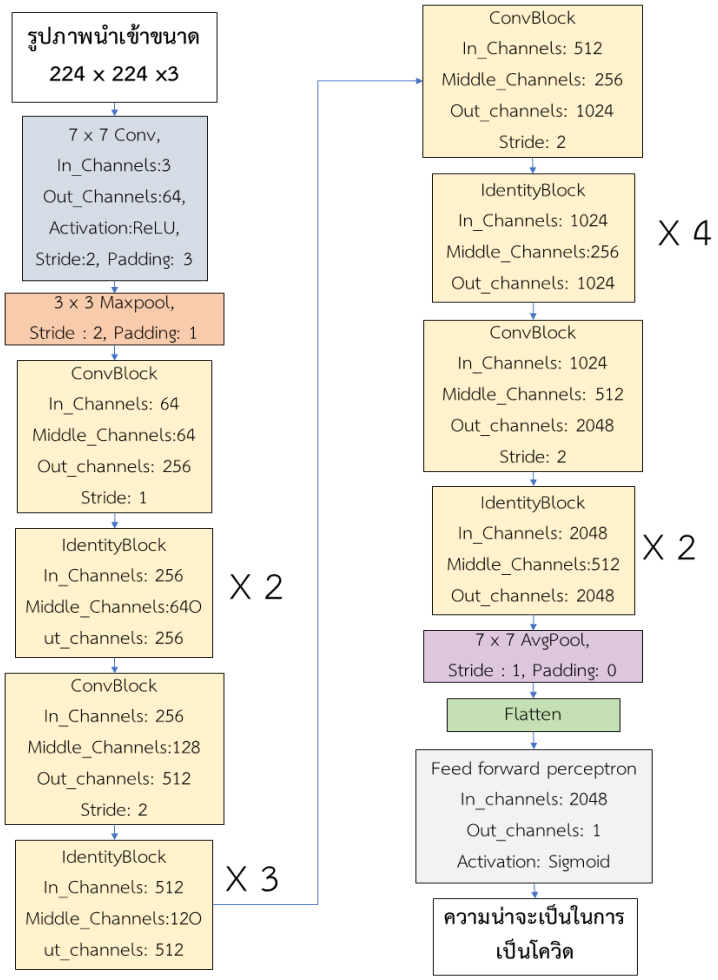
หากคำนวณหาพื้นที่ใต้กราฟจะได้ค่าระหว่าง 0 ถึง 1 โดยหากค่าดังกล่าวเท่ากับ 0.5 จะได้ว่าโมเดลทำนายข้อมูลแบบสุ่ม หากค่าดังกล่าวมีค่าเข้าใกล้ 1 จะได้ว่าโมเดลมีประสิทธิภาพในการแยกข้อมูลบวกและข้อมูลลบ ในขณะที่หากคำนวณค่าพื้นที่ใต้กราฟแล้วเข้าใกล้ 0 จะได้ว่าโมเดลสามารถแยกบวกและลบได้ดี แต่แยกสลับกันโดยทำนายข้อมูลบวกเป็นลบ และทำนายข้อมูลลบเป็นบวก นอกจากนี้ข้อมูลจากกราฟ ROC สามารถใช้ในการตัดสินใจกำหนดค่าขีดขั้นในการแบ่งกลุ่มข้อมูลเป็นบวกและเป็นลบได้ โดยมีข้อแลกเปลี่ยนคือหากกำหนดค่าขีดขั้นเพิ่มขึ้น ข้อมูลที่เป็นบวกจริงและข้อมูลที่เป็นบวกเท็จจะน้อยลง ในขณะที่ถ้ากำหนดค่าขีดขั้นลดลงข้อมูลที่เป็นบวกจริงและเป็นบวกเท็จจะเพิ่มขึ้น ดังนั้นค่าขีดขั้นที่เหมาะสมที่สุดที่ควรใช้เมื่อพิจารณาจากกราฟ ROC คือค่าขีดขั้นที่มีข้อมูลบวกจริงสูง ในขณะที่ข้อมูลบวกเท็จต่ำ นั่นคืออัตราบวกจริงสูงและอัตราบวกเท็จต่ำ

# บทที่ 3

# ผลการศึกษา (Results)

## 3.1 โมเดลที่ใช้ในการศึกษา

งานค้นคว้าอิสระฉบับนี้ใช้โมเดลโครงข่ายประสาทคอนโวลูชัน ResNet50 [6] ในการคัดแยกภาพเอกสเรย์ โดยตัวโมเดลได้มีการสร้างและสอนโดยใช้ไลบรารี pytorch โมเดล ResNet50 มีลักษณะดังแสดงในรูปที่ 3.1

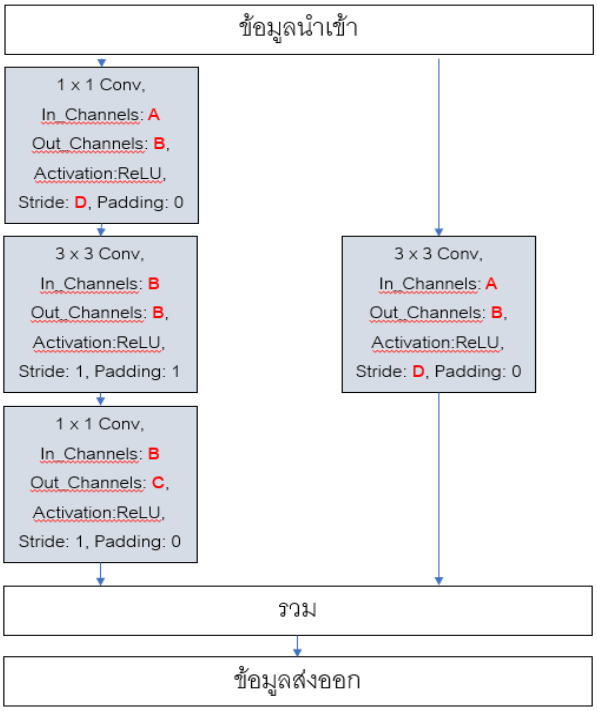



รูปที่ 3.1: โครงสร้าง ResNet50

พิจารณา ResNet50 ชั้น Conv คือชั้นคอนโวลูชัน Maxpool คือชั้นกรองค่าสูงสุด AvgPool คือชั้นกรองค่าเฉลี่ย Feed forward perceptron คือชั้นโครงข่ายประสาทป้อนไปข้างหน้า และ Flatten คือชั้นสำหรับแปลงข้อมูล 3 มิติเป็นข้อมูล 1 มิติ

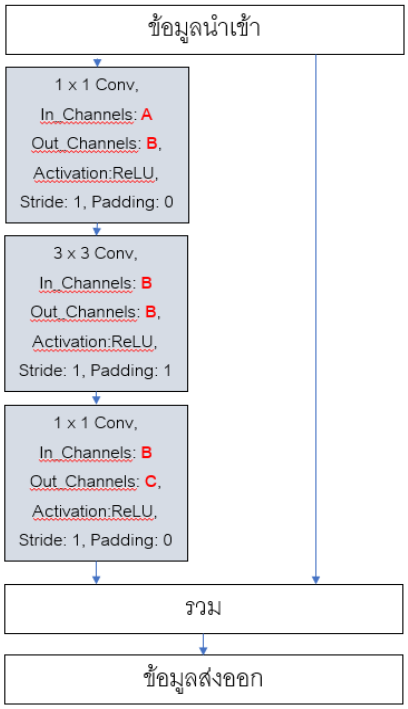
สำหรับชั้นทุกชั้นยกเว้นชั้น ConvBlock และชั้น IdentityBlock หากเป็นชั้นคอนโวลูชัน In_channels จะเป็นจำนวนข้อมูลนำเข้าสองมิติ ในขณะที่ชั้นโครงข่ายประสาทป้อนไปข้างหน้าจะเป็นจำนวนข้อมูลนำเข้าหนึ่งมิติ ในทำนองเดียวกัน หากเป็นชั้นคอนโวลูชัน Out_channels เป็นจำนวนข้อมูลส่งออกสองมิติ ในขณะที่ชั้นโครงข่ายประสาทป้อนไปข้างหน้าจะเป็นจำนวนข้อมูลส่งออกหนึ่งมิติ Activation คือฟังก์ชันกระตุ้นที่ใช้ และสำหรับชั้นคอนโวลูชัน ชั้นกรองค่าสูงสุด และชั้นกรองค่าเฉลี่ย ตัวเลขข้างหน้าชั้นคือขนาดตัวกรอง Stride คือขนาดของสไตรด์ และ Padding คือจำนวนชั้นที่ทำการแพดดิ้ง

สำหรับชั้น ConvBlock มีโครงสร้างดังแสดงในรูปที่ 3.2




รูปที่ 3.2: โครงสร้าง ConvBlock เมื่อ In_channels=A Middle_channels=B Out_channels=C และ Stride=D

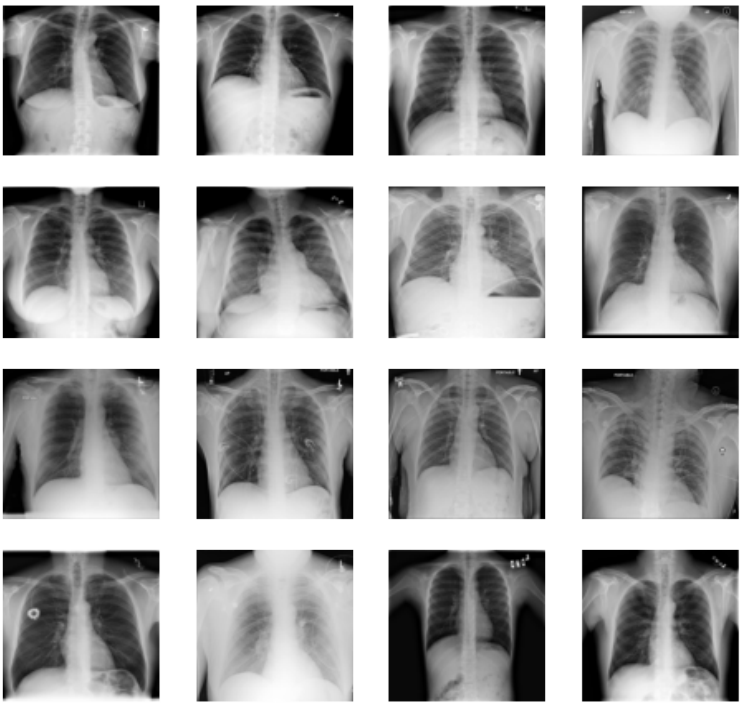
และชั้น IdentityBlock มีโครงสร้างดังแสดงในรูปที่ 3.3



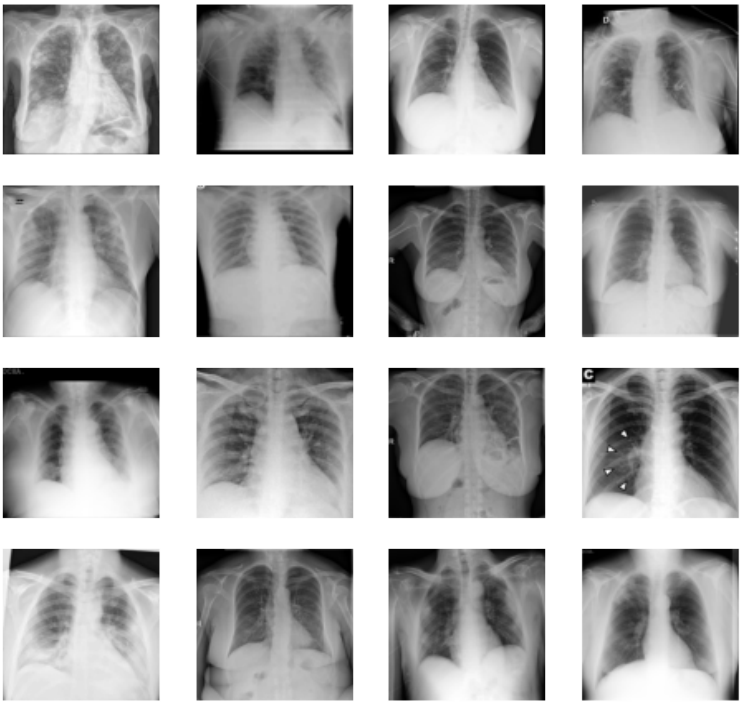

รูปที่ 3.3: โครงสร้าง Identity เมื่อ In_channels=A Middle_channels=B และ Out_channels=C

## 3.2 ข้อมูลที่ใช้ในการสอนโมเดล

ข้อมูลที่ใช้ในการสอนโมเดลนำมาจากเว็บไซต์ Kaggle จากฐานข้อมูล COVID-19 Radiography Database [7] โดยข้อมูลดังกล่าวเป็นข้อมูลรูปภาพเอกซเรย์ปอดซึ่งประกอบไปด้วยรูปภาพเอกซเรย์ปอดธรรมดา รูปภาพเอกซเรย์คนไข้ที่มีอาการปอดบวม รูปภาพเอกซเรย์คนไข้ที่เป็นมะเร็งปอด และรูปภาพเอกซเรย์ของคนไข้ที่มีเชื้อโควิดลงปอด โดยผู้ศึกษาได้นำข้อมูลรูปเอกซเรย์ปอดธรรมดาเป็นจำนวน 3,000 รูป และรูปภาพเอกซเรย์ของคนไข้ที่มีเชื้อโควิดลงปอด 3,000 รูป มาใช้ในการศึกษานี้ โดยในการแบ่งข้อมูล จะแบ่งข้อมูลออกเป็นสามกลุ่มคือข้อมูลชุดสำหรับสอนโมเดล 60% ข้อมูล validate 20% สำหรับวัดการ overfit ในโมเดล และข้อมูลชุดสำหรับทดสอบโมเดล 20% รูปที่ 3.4 แสดงให้เห็นถึงตัวอย่างของรูปเอกซเรย์ปอดธรรมดา และรูปที่ 3.5 แสดงให้เห็นตัวอย่างของรูปเอกซเรย์ปอดของคนไข้ที่มีเชื้อโควิดลงปอด



รูปที่ 3.4: ตัวอย่างรูปภาพเอกซเรย์ปอดธรรมดา



รูปที่ 3.5: ตัวอย่างรูปภาพเอกซเรย์คนไข้ที่มีเชื้อโควิดลงปอด

## 3.3 รายละเอียดในการสอนโมเดล

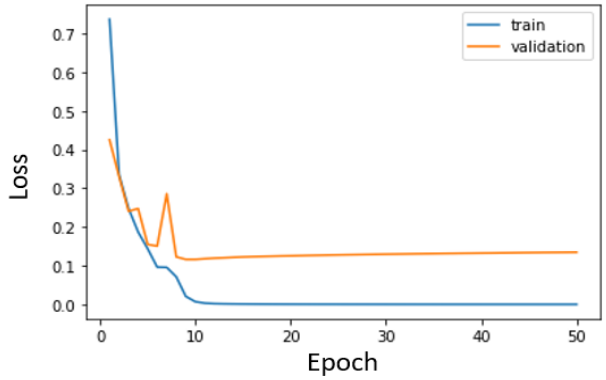
สำหรับข้อมูลที่ใช้ในการสอนโมเดล กำหนดให้รูปภาพที่เป็นโควิดมีค่าจริงเป็น 1 และรูปภาพที่ไม่เป็นโควิดมีค่าจริงเป็น 0 เนื่องจากปัญหาดังกล่าวเป็นปัญหาการคัดแยกข้อมูลไบนารี ฟังก์ชันสูญเสียที่ใช้ในการสอนโมเดลจึงเป็นฟังก์ชัน Binary cross entropy loss ซึ่งมีสมการดังนี้

$$L = \frac{1}{k}\sum_{i=1}^{k} -t^{\{i\}} \log\left(o^{\{i\}}\right) - (1 - t^{\{i\}}) \log\left(1 - o^{\{i\}}\right)$$

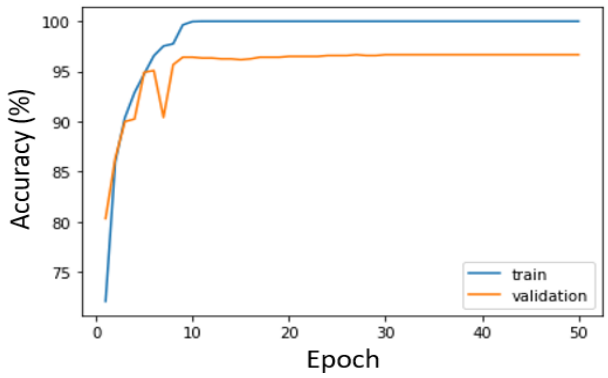
โดย $k$ เป็นจำนวนข้อมูลทั้งหมดที่ใช้ในการสอนโมเดล $t^{\{i\}}$ เป็นค่าค่าจริงซึ่งมีค่า 1 หรือ 0 ของข้อมูลชุดที่ $i$ และ $o^{\{i\}}$ เป็นค่าทำนายความน่าจะเป็นในการเป็นโควิดของข้อมูลชุดที่ $i$

การศึกษานี้ได้ใช้ Stochastic Gradient descent ในการปรับปรุงค่าน้ำหนักและไบแอสเพื่อให้ค่าที่ได้จากฟังก์ชันสูญเสียมีค่าน้อยที่สุด โดยใช้อัตราการเรียนรู้ที่ 0.01 จำนวนครั้งในการสอนโมเดลที่ 50 ครั้ง และขนาด batch เท่ากับ 64

รูปที่ 3.6 แสดงถึงค่าสูญเสียของโมเดลเมื่อทำนายค่าบนข้อมูลชุดเรียนรู้ และค่าสูญเสียของโมเดลเมื่อทำนายค่าบนข้อมูลชุด validate และรูปที่ 3.7 แสดงถึงความแม่นยำในการทำนายค่าของโมเดลบนข้อมูลชุดเรียนรู้ และข้อมูลชุด validate




รูปที่ 3.6: กราฟค่าสูญเสียเมื่อโมเดลทำนายข้อมูลชุดเรียนรู้ และข้อมูลชุด validation

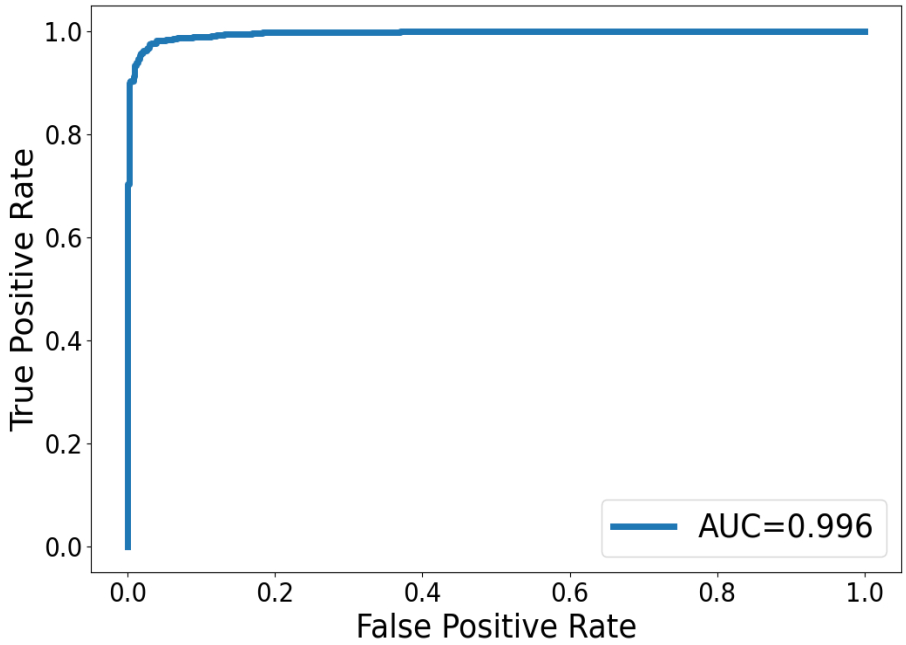



รูปที่ 3.7: กราฟค่าความแม่นยำเมื่อโมเดลทำนายข้อมูลชุดเรียนรู้ และข้อมูลชุด validation

เมื่อพิจารณากราฟค่าสูญเสีย สังเกตว่าโมเดลเริ่มทำนายข้อมูลชุด validation ได้ดีที่สุดที่ epoch 9 และหลังจาก epoch 9 เป็นต้นไปโมเดลเริ่มทำนายข้อมูลได้แย่ลง นั่นแสดงว่าโมเดลเริ่ม overfit กับข้อมูลชุดเรียนรู้ กล่าวคือโมเดลเริ่มจะทำนายข้อมูลอิงตามข้อมูลชุดเรียนรู้มากเกินไป ทำให้โมเดลสามารถทำนายข้อมูลชุดอื่นๆได้แย่ลง เราจึงนำน้ำหนักและไบแอสของโมเดลในการเรียนรู้ epoch ที่ 9 มาใช้ในการวัดผลโมเดลในครั้งนี้

## 3.4 การวัดผลโมเดล

ในการค้นคว้านี้ได้ใช้กราฟ ROC และ confusion matrix ในการประสิทธิภาพโมเดลบนข้อมูลชุดทดสอบ เมื่อทดสอบโมเดลที่ epoch 9 บนข้อมูลชุดทดสอบ ได้กราฟ ROC ดังแสดงในรูปที่ 3.8




รูปที่ 3.8: กราฟ ROC เมื่อวัดผลโมเดล epoch 9 บนข้อมูลชุดทดสอบ

โดยกราฟดังกล่าวมีพื้นที่ใต้กราฟเท่ากับ 0.996 ซึ่งใกล้ 1 นั่นคือโมเดลนี้สามารถแยกข้อมูลภาพเอกซเรย์ปอดทั่วไป และภาพเอกซเรย์ปอดของผู้ป่วยที่มีเชื้อโควิดลงปอดได้ดี เมื่อวัดผลโมเดลดังกล่าวด้วย Confusion matrix ได้ผลดังตารางที่ 3.1

| | | ค่าทำนาย | |
|---|---|---|---|
| | | **บวก** | **ลบ** |
| **ค่าจริง** | **บวก** | 557 | 19 |
| | **ลบ** | 18 | 606 |

ตารางที่ 3.1: Confusion matrix เมื่อวัดผลโมเดล epoch 9 บนข้อมูลชุดทดสอบ

เมื่อคำนวณค่าความแม่นยำ ค่าความเที่ยง และค่าการระลึกได้ จะได้ว่าโมเดลดังกล่าวมีความแม่นยำ 96.92% มีค่าความเที่ยง 96.87% และมีค่าการระลึกได้ 96.70%

# บทที่ 4

# สรุปผลการศึกษา (Conclusion)

การค้นคว้าอิสระนี้ได้ทำการสร้างโครงข่ายประสาทเทียม Resnet50 ด้วยภาษา Python เพื่อใช้ในการคัดแยกภาพเอกสเรย์ปอดว่าผู้ป่วยเป็น Covid pneumonia หรือไม่ โดยข้อมูลที่นำมาสอนให้กับโมเดลนำมาจากเว็บไซต์ Kaggle ซึ่งประกอบไปด้วยรูปภาพเอกซเรย์ปอดทั่วไปจำนวน 3000 รูป และรูปภาพเอกซเรย์ปอดของคนไข้ที่มีเชื้อโควิดลงปอดจำนวน 3000 รูป ข้อมูลดังกล่าวแบ่งออกเป็นสามกลุ่ม โดย 60% ของข้อมูลทั้งหมดนำไปใช้ในการสอนโมเดล 20% ของข้อมูลทั้งหมดนำมาใช้ในการ validate โมเดล เพื่อวัดความ overfit ของโมเดลในการเรียนรู้ และ 20% ของข้อมูลทั้งหมดใช้ในการวัดผลโมเดล โดยในการสอนโมเดลได้มีการสอนโมเดลทั้งหมด 50 รอบ ฟังก์ชันสูญเสียที่ใช้ในการสอนโมเดลคือฟังก์ชัน Binary cross entropy loss และใช้ Stochastic gradient descent ด้วยอัตราการเรียนรู้ 0.01 ในการปรับปรุงค่าน้ำหนักและไบแอสในโมเดลเพื่อลดค่าสูญเสียลง

ในการวัดผลได้มีการใช้กราฟ ROC และ Confusion matrix ในการวัดผลบนข้อมูลชุดทดสอบ โดยวัดผลบนโมเดลที่เรียนรู้ใน epoch 9 ซึ่งเป็น epoch ที่โมเดลสามารถทำนายข้อมูล validation ได้ดีที่สุด จากกราฟ ROC พบว่าพื้นที่ใต้กราฟมีค่าเท่ากับ 0.996 บ่งบอกว่าโมเดลนี้แยกข้อมูลรูปภาพเอกซเรย์ปอดคนปกติและรูปภาพเอกซเรย์ปอดของผู้ป่วยที่มีเชื้อโควิดลงปอดได้ดี เมื่อวัดผลโมเดลบน confusion matrix พบว่าโมเดลมีความแม่นยำ 96.92% มีค่าความเที่ยง 96.87% และมีค่าการระลึกได้ 96.70%

# บรรณานุกรม


[1] A. Ghosh, "Fundamental Concepts of Convolutional Neural Network," *Intelligent Systems Reference Library*, vol. 172, pp. 519-567, 2020.

[2] Animated AI., "*The Basic Algorithm*." github.io https://animatedai.github.io/ (accessed february, 2023).

[3] D. Stutz, "Understanding Convolutional Neural Networks," Rwthaachen University, 2014.

[4] F. Tom, et al., "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, pp. 861-874, 2006.

[5] G. Aurelien, *Hand-On Machine Learning with Scikit-Learn and TensorFlow.* California:O'Reilly Media, 2017.

[6] H. Kaiming, et al., "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770-778, 2016.

[7] R. Tawsifur, et al., "*COVID-19 Radiography Database*." kaggle.com https:// www.kaggle.com/ datasets/ tawsifurrahman/ covid19-radiography-database? select=COVID-19_Radiography_Dataset (accessed September, 2022).

[8] ปริญญา สงวนสัตย์, *Artificial Intelligent with Machine Learning AI สร้างได้ด้วยแมชชีนเลิร์นนิ่ง.* นนทบุรี:ไอดีซี พรีเมียร์, 2562.

**ภาคผนวก**

โปรแกรม Python สำหรับการสร้างโมเดลสำหรับการคัดแยกรูปภาพเอกสเรย์ว่าเป็น Covid pneumonia หรือไม่

```
from google.colab import drive
drive.mount('/content/drive')
```

```
!unzip /content/drive/MyDrive/Covid_X-ray/covid_x-ray.zip

!mkdir /content/covid_normal_img
!cp -a /content/COVID-19_Radiography_Dataset/COVID/images/.
    /content/covid_normal_img/COVID/
!cp -a /content/COVID-19_Radiography_Dataset/Normal/images/.
    /content/covid_normal_img/Normal/
```

```
import cv2
from datetime import datetime
import glob
import matplotlib.pyplot as plt
import numpy as np
import random
import os
import pickle
from PIL import Image
import torch
from torch import nn
import torchvision
from torchvision.io import read_image
from torch.utils.data import Dataset, DataLoader, WeightedRandomSampler
from sklearn import metrics

torch.manual_seed(0)
```

```
class ImageDataset():
   def __init__(self, path):
      self.path = path
      self.image_name_list = []
      self.image_label_list = []
   for label in os.listdir(self.path):
      all_image_path = os.path.join(self.path,label)
      for image_name in os.listdir(all_image_path):
      self.image_name_list.append(image_name)
      self.image_label_list.append(label)
      self.mapper = dict(zip(sorted(os.listdir(self.path), reverse =
          True),range(len(os.listdir(self.path)))))

def __len__(self):
   return len(self.image_name_list)

def __getitem__(self, inx):
   image = self.image_name_list[inx]
   label = self.image_label_list[inx]
   full_image_path = os.path.join(os.path.join(self.path,label),image)
   resize_transform = torchvision.transforms.Resize((224, 224))

   return (resize_transform(read_image(full_image_path))/255).to(torch.float32),
       torch.tensor(self.mapper[label], dtype = torch.float32)
```

```
def subset_label(subset):
   label_list = []
   for batch in subset:
      label = batch[1].int().item()
      label_list.append(label)

   return label_list

def subset_label_count(subset):
   label_counter = {i:0 for i in list(subset.dataset.mapper.values())}
   for batch in subset:
      label = batch[1].int().item()
      label_counter[label]+=1

   return label_counter

def subset_label_percent(subset):
   label_counter = {i:0 for i in list(subset.dataset.mapper.values())}
   for batch in subset:
      label = batch[1].int().item()
      label_counter[label]+=1

   percent_list = [count/sum(label_counter.values()) for count in
```

```
        list(label_counter.values())]
    label_percent = {label:round(percent, 3) for label, percent in
        zip(list(label_counter.keys()), percent_list)}

    return label_percent
```

```
train_set, val_set, test_set = torch.utils.data.random_split(dataset, [0.6, 0.2, 0.2])

label_count_tuple = [(k, v) for k, v in subset_label_count(train_set).items()]
class_weights = {label:11000/count for label, count in label_count_tuple}
weighted_label = list(map(class_weights.get, subset_label(train_set)))
sampler = WeightedRandomSampler(weighted_label, len(train_set))

train_dataloader = DataLoader(train_set, batch_size=64, sampler=sampler)
val_dataloader = DataLoader(val_set, batch_size=64)
test_dataloader = DataLoader(test_set, batch_size=64)

data_dict = {"train_dataloader" : train_dataloader, "val_dataloader" :
    val_dataloader, "test_dataloader" : test_dataloader}

#Save test set for later
with open(os.path.join("/content/drive/MyDrive/Covid_X-ray/all_dataloader.pkl"),
    "wb") as f:
    pickle.dump(data_dict, f)
```

```
class ConvBlock(nn.Module):
    def __init__(self, in_channels, middle_channels, out_channels, stride = 1):
        super(ConvBlock, self).__init__()
        self.conv1 = nn.Sequential(nn.Conv2d(in_channels, middle_channels, kernel_size =
            1, stride = stride), nn.BatchNorm2d(middle_channels), nn.ReLU())
        self.conv2 = nn.Sequential(nn.Conv2d(middle_channels, middle_channels,
            kernel_size = 3, stride = 1, padding = 1), nn.BatchNorm2d(middle_channels),
            nn.ReLU())
        self.conv3 = nn.Sequential(nn.Conv2d(middle_channels, out_channels, kernel_size
            = 1, stride = 1), nn.BatchNorm2d(out_channels), nn.ReLU())
        self.conv4 = nn.Sequential(nn.Conv2d(in_channels, out_channels, kernel_size = 1,
            stride = stride), nn.BatchNorm2d(out_channels))
    def forward(self, x):
        residual = x.clone()
        output = self.conv1(x)
        output = self.conv2(output)
        output = self.conv3(output)
        residual = self.conv4(residual)

        output = residual + output
        return output
```

```
class IdentityBlock(nn.Module):
  def __init__(self, in_channels, middle_channels, out_channels):
    super(IdentityBlock, self).__init__()
    self.conv1 = nn.Sequential(nn.Conv2d(in_channels, middle_channels, kernel_size =
        1, stride = 1), nn.BatchNorm2d(middle_channels), nn.ReLU())
    self.conv2 = nn.Sequential(nn.Conv2d(middle_channels, middle_channels,
        kernel_size = 3, stride = 1, padding = 1),
    nn.BatchNorm2d(middle_channels), nn.ReLU())
    self.conv3 = nn.Sequential(nn.Conv2d(middle_channels, out_channels, kernel_size
        = 1, stride = 1), nn.BatchNorm2d(out_channels), nn.ReLU())
  def forward(self, x):
    residual = x.clone()
    output = self.conv1(x)
    output = self.conv2(output)
    output = self.conv3(output)

    output = residual + output
    return output

class ResNet(nn.Module):
  def __init__(self, layers = [3, 4, 6, 3], num_classes = 2):
    super(ResNet, self).__init__()
    self.conv1 = nn.Sequential(nn.Conv2d(1, 64, kernel_size = 7, stride = 2, padding
        = 3), nn.BatchNorm2d(64), nn.ReLU())
    self.maxpool = nn.MaxPool2d(kernel_size = 3, stride = 2, padding = 1)
    self.stack1 = self.stacked_block([64, 64, 256], layers[0])
    self.stack2 = self.stacked_block([256, 128, 512], layers[1], stride = 2)
    self.stack3 = self.stacked_block([512, 256, 1024], layers[2], stride = 2)
    self.stack4 = self.stacked_block([1024, 512, 2048], layers[3], stride = 2)
    self.avgpool = nn.AvgPool2d(7, stride = 1)
    if num_classes == 2:
      self.fc = nn.Sequential(nn.Linear(2048, 1), nn.Sigmoid())
    else:
      self.fc = nn.Sequential(nn.Linear(2048, num_classes), nn.Sigmoid())

  def stacked_block(self, planes, no_blocks, stride = 1):
    layers = []

    layers.append(ConvBlock(planes[0], planes[1], planes[2], stride = stride))

    for i in range(1, no_blocks):
      layers.append(IdentityBlock(planes[2], planes[1], planes[2]))


    return nn.Sequential(*layers)

  def forward(self, x):
    x = self.conv1(x)
    x = self.maxpool(x)
```

```
        x = self.stack1(x)
        x = self.stack2(x)
        x = self.stack3(x)
        x = self.stack4(x)
        x = self.avgpool(x)
        x = x.view(x.size(0), -1)
        x = self.fc(x)

        return x
```

```
num_classes = 2
layer = [3, 4, 6, 3] #Resnet50
num_epochs = 20
learning_rate = 0.01

model = ResNet(layer, num_classes)

loss_fn = nn.BCELoss()
optimizer = torch.optim.SGD(model.parameters(), lr = learning_rate)
```

```
def fit(model, epochs, train_dataloader, val_dataloader, loss_fn, optimizer,
    save_path, load_path = None):
  train_loss_list = []
  train_acc_list = []
  val_loss_list = []
  val_acc_list = []

  if load_path:
    model.load_state_dict(torch.load(load_path))

  for epoch in range(epochs):
    print('epoch :',epoch+1)
    model = model.to('cuda')

    train_size = len(train_dataloader.dataset)
    num_batches = len(train_dataloader)
    train_loss = 0
    correct = 0

    #train
    for batch, (img, label) in enumerate(train_dataloader):
      img = img.to('cuda')
      label = label.to('cuda')

      pred = torch.squeeze(model(img))

      correct += (pred.cpu().detach().apply_(lambda x: 1 if x>=0.5 else 0) ==
          label.cpu()).sum().item()
```

```
    # correct += (torch.argmax(pred, dim = 1).cpu().detach() ==
        torch.argmax(label, dim = 1).cpu()).sum().item()

    loss = loss_fn(pred, label)
    train_loss += loss

    optimizer.zero_grad()
    loss.backward()

    optimizer.step()

    if batch % 20 == 0:
        loss, current = loss.item(), batch * len(img)
        print(f"loss: {loss:>7f} [{current:>5d}/{train_size:>5d}]")

train_loss /= num_batches
correct /= train_size
train_loss_list.append(train_loss)
train_acc_list.append(100*correct)
print(f"Average train loss: {train_loss:>7f}")

#val
val_size = len(val_dataloader.dataset)
num_batches = len(val_dataloader)
val_loss = 0
correct = 0

with torch.no_grad():
    for img, label in val_dataloader:
        img = img.to('cuda')
        label = label.to('cuda')
        pred = torch.squeeze(model(img))
        val_loss += loss_fn(pred, label)

        correct += (pred.cpu().apply_(lambda x: 1 if x>=0.5 else 0) ==
            label.cpu()).sum().item()

val_loss /= num_batches
correct /= val_size
print(f"Validation Error: \n Accuracy: {(100*correct):>0.1f}%, Avg loss:
    {val_loss:>8f} \n")

val_loss_list.append(val_loss)

val_acc = 100*correct

current_time = datetime.now().strftime("%H:%M:%S")

if len(val_acc_list) >= 1 and val_acc > max(val_acc_list):
```

```
        #Save entire model
        torch.save(model, os.path.join(save_path, 'best_model'+ current_time +'.pt'))

      val_acc_list.append(val_acc)

  loss_dict = {'train_loss':train_loss_list, 'train_acc':train_acc_list,
    'val_loss':val_loss_list, 'val_acc':val_acc_list}

  with open(os.path.join(save_path ,"log" + current_time + ".pkl"), "wb") as f:
    pickle.dump(loss_dict, f)
```

```
!mkdir ./save_model
fit(model, num_epochs, train_dataloader, val_dataloader, loss_fn, optimizer,
    './save_model')
```

```
!mv '/content/save_model/best_model11:50:26.pt'
    '/content/drive/MyDrive/Covid_X-ray/best_model11:50:26.pt'
!mv '/content/save_model/log12:07:02.pkl'
    '/content/drive/MyDrive/Covid_X-ray/log12:07:02.pkl'
```

```
pred_prob = torch.tensor([])
with torch.no_grad():
  for img, label in test_dataloader:
    img = img.to('cuda')
    pred = torch.squeeze(model(img)).cpu()

    pred_prob = torch.cat((pred_prob, pred), dim = 0)

pred_prob
```

```
fpr, tpr, threshold = metrics.roc_curve(np.array(full_label), np.array(pred_prob))
auc = metrics.roc_auc_score(np.array(full_label), np.array(pred_prob))

plt.figure(figsize=(12, 8))
plt.plot(fpr,tpr,label="AUC="+str(round(auc, 3)))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.show()
```

```
print(torch.argmax(tpr-fpr))
```

```
#test
with open('/content/drive/MyDrive/Covid_X-ray/all_dataloader.pkl', 'rb') as f:
data_dict = pickle.load(f)

test_dataloader = data_dict['test_dataloader']

model_file = glob.glob('/content/drive/MyDrive/Covid_X-ray/best_model*')[-1]
model = torch.load(model_file).to('cuda')
model.eval()

full_pred = torch.tensor([])
full_label = torch.tensor([])

with torch.no_grad():
for img, label in test_dataloader:
img = img.to('cuda')
pred = torch.squeeze(model(img)).cpu().apply_(lambda x: 1 if x>=0.5 else 0)

full_pred = torch.cat((full_pred, pred), dim = 0)
full_label = torch.cat((full_label, label), dim = 0)
```

```
pred_label_dict = {"full_pred" : full_pred, "full_label" : full_label}

#Save prediction and label and turn off GPU
with open(os.path.join("/content/drive/MyDrive/Covid_X-ray/pred_label_dict.pkl"),
    "wb") as f:
pickle.dump(pred_label_dict, f)
```

```
with open("/content/drive/MyDrive/Covid_X-ray/pred_label_dict.pkl", 'rb') as f:
pred_label_dict = pickle.load(f)

full_pred = pred_label_dict['full_pred']
full_label = pred_label_dict['full_label']

TP = 0
FP = 0
TN = 0
FN = 0

for index in range(len(full_pred)):
#True positive
if full_pred[index] == 1 and full_label[index] == 1:
TP += 1
elif full_pred[index] == 1 and full_label[index] == 0:
FP += 1
elif full_pred[index] == 0 and full_label[index] == 0:
TN += 1
```

```
else:
FN += 1

print(f'TP : {TP}, FP : {FP} , TN : {TN}, FN : {FN}')
print(f'Accuracy : {100*(TP + TN)/(TP+FP+TN+FN):>0.2f} %')
print(f'Recall : {100*TP/(TP+FN):>0.2f} %, Precision : {100*TP/(TP+FP):>0.2f} %')
```