การทำนายระยะการบินโดยใช้โครงข่ายประสาทเชิงลึกร่วมกับการเพิ่มประสิทธิภาพกลุ่มอนุภาคในกระบวนการผลิตฮาร์ดดิสก์ไดรฟ์

PREDICTION OF FLYING HEIGHT USING DEEP NEURAL NETWORK BASED ON PARTICLE SWARM OPTIMIZATION IN HARD DISK DRIVE MANUFACTURING PROCESS

วรวิทย์ กาญจนพฤฒิพงศ์

WORAWIT KANJANAPRUTHIPONG

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมหุ่นยนต์และระบบอัจฉริยะเชิงคำนวณ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2567

KMITL-2024-EN-M-407-247

PREDICTION OF FLYING HEIGHT USING DEEP NEURAL NETWORK BASED ON PARTICLE SWARM OPTIMIZATION IN HARD DISK DRIVE MANUFACTURING PROCESS

WORAWIT KANJANAPRUTHIPONG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
MASTER OF ENGINEERING IN ROBOTICS AND COMPUTATIONAL INTELLIGENCE SYSTEMS
SCHOOL OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2024

KMITL-2024-EN-M-407-247

หัวข้อวิทยานิพนธ์	การทำนายระยะการบินโดยใช้โครงข่ายประสาทเชิงลึกร่วมกับการเพิ่มประสิทธิภาพกลุ่มอนุภาคในกระบวนการผลิตฮาร์ดดิสก์ไดรฟ์
นักศึกษา	นายวรวิทย์ กาญจนพฤฒิพงศ์
รหัสประจำตัว	64601124
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมหุ่นยนต์และระบบอัจฉริยะเชิงคำนวณ
พ.ศ.	2567
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ผศ. ดร. ภูมิ คงห้วยรอบ

บทคัดย่อ

ในโรงงานการผลิตอุตสาหกรรมฮาร์ดดิสก์ไดรฟ์ หลังจากที่ฮาร์ดดิสก์ไดรฟ์ถูกประกอบจากสายการผลิต มีความจำเป็นที่จะต้องทำการปรับเข้าสู่ภาวะมาตรฐานด้วยวิธีการคำนวณต่างๆ อาทิ เช่น คำนวณหาพื้นความจุซึ่งเป็นพื้นที่จัดเก็บข้อมูลซึ่งมีหน่วยเป็นเทระไบต์ (Terabyte : TB) ในปัจจุบัน มีการคำนวณความสูงของการบินของหัวอ่าน/เขียนซึ่งถูกวัดระยะห่างระหว่างหัวอ่าน/เขียนและแผ่นดิสก์โดยการใช้การจ่ายกระแสไฟไปยังขวดลวดตัวทำความร้อนเพื่อให้หัวอ่าน/เขียนเกิดการยื่นออกมาจนกระทั่งถึงเป้าหมายความสูงของการบินที่ซึ่งจะให้ประสิทธิภาพการเขียนและการอ่านที่ดีที่สุดและเหมาะสมกับฮาร์ดดิสก์ไดรฟ์แต่ละตัว และทำการระบุกระแสไฟโดยมีหน่วยเป็น DAC (Digital to Analog Converter unit) แล้วทำการบันทึกค่า DAC สำหรับไว้ใช้งานของหัวอ่าน/เขียน ซึ่งจะมีอุปกรณ์ที่เรียกว่าพรีแอมป์ (Preamp) ทำงานร่วมกับเฟิร์มแวร์ (Firmware) โดยการแปลงหน่วย DAC เป็นมิลลิวัตต์ ในปัจจุบันมีการคำนวณความสูงของการบินของหัวอ่าน/เขียนหลายครั้ง ซึ่งเรียกว่า Flying Height 1 (FH1) และ Flying Height 2 (FH2) การคำนวณความสูงของการบินแต่ละ FH จะใช้เวลาประมาณ 5 ชั่วโมง เนื่องจากมีการแบ่งโซนออกเป็น 240 โซน และครอบคลุมทั่วทั้งแผ่นดิสก์ และปัจจุบันมีหัวอ่าน/เขียนถึง 20 หัวอ่าน แนวคิดหลักของงานวิจัยนี้คือการลดเวลาการคำนวณความสูงของการบินของหัวอ่าน/เขียน โดยการใช้โครงข่ายประสาทเชิงลึก (Deep Neural Network : DNN) ร่วมกับการเพิ่มประสิทธิภาพกลุ่มอนุภาค (Particle Swarm Optimization : PSO) เพื่อทำนาย DAC โปรไฟล์ของ FH2 ได้อย่างรวดเร็ว ซึ่งอยู่ใกล้กับ FH1 โดยที่ FH1 เป็นการคำนวณความสูงของการบินครั้งแรก ซึ่งจะถูกนำมาป้อนข้อมูลให้กับแบบจำลองโครงข่ายประสาทเชิงลึก DNN

Thesis Title:	Prediction of Flying Height Using Deep Neural Network Based on Particle Swarm Optimization in Hard Disk Drive Manufacturing Process
Student:	Worawit Kanjanapruthipong
Student ID:	64601124
Degree:	Master of Engineering
Program:	Robotics and Computational Intelligence Systems
Year:	2024
Thesis Advisor:	Asst. Prof. Dr. Poom Konghuayrob

ABSTRACT

In contemporary hard disk drive (HDD) manufacturing processes, after the assembly of the HDD from the production line, a series of diverse calibration procedures are necessary to ensure standardization. These include capacity calibration, which determines the storage space in terabytes (TB) presently available, and flying height (FH) calibration, which evaluates the distance between the head and the disk by applying electric current to the heater coil element to achieve the desired FH, thus optimizing the writing and reading performance and tailoring it to each HDD. Additionally, electric current is saved in a digital-to-analog converter (DAC) unit for the utilization of a read/write head, while a preamp collaborates with the drive firmware to convert the electric current in the DAC unit to milliwatts. In the present scenario, multiple calibrations of flying heights (FHs), specifically flying height 1 (FH1) and flying height 2 (FH2), are performed. Each FH calibration requires a testing time of approximately 5 h owing to the separation of measurement points into 240 locations across the disk surface, referred to as test zones, with a total of 20 heads. The primary objective of this study is to reduce the testing time by using a combination of deep neural network (DNN) and particle swarm optimization techniques to predict the DAC profiles of FH2 as it approaches FH1, where FH1 is the input for the DNN model.

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สำเร็จได้ด้วยความกรุณาจากอาจารย์ที่ปรึกษา ผศ. ดร. ภูมิ คงห้วยรอบ ที่ให้ความช่วยเหลือ ให้คำชี้แนะช่วยแก้ปัญหาตลอดจนให้ความรู้และประสบการณ์ที่ดีแก่ข้าพเจ้า

ขอขอบคุณสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ให้การสนับสนุนทุนการศึกษาและขอขอบคุณบริษัท ซีเกท เทคโนโลยี (ประเทศไทย) จำกัด ที่ให้การสนับสนุนการวิจัยนี้

ขอขอบคุณภรรยาของข้าพเจ้า คุณกัลยารักษ์ ทับทิมศรี ที่เป็นเสมือนคู่คิดและกำเป็นลังใจที่ดีตลอดมา

สำหรับคุณงามความดีอันใดที่เกิดจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้กับบิดามารดา ซึ่งเป็นที่รักและเคารพยิ่ง ตลอดจนอาจารย์ภาควิชาวิศวกรรมหุ่นยนต์และปัญญาประดิษฐ์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่เคารพทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้และถ่ายทอดประสบการณ์ที่ดีให้แก่ข้าพเจ้า

วรวิทย์ กาญจนพฤฒิพงศ์

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญตาราง

ตารางที่ **หน้า**

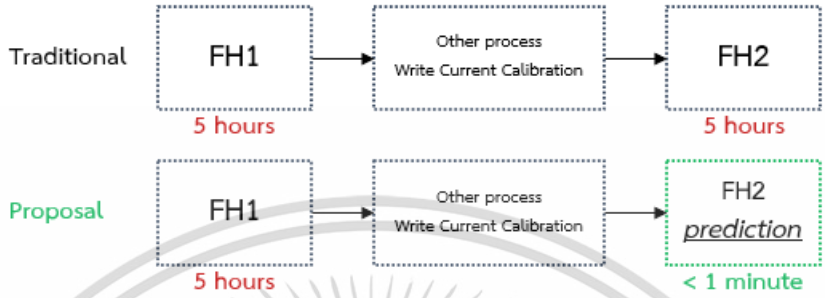
บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

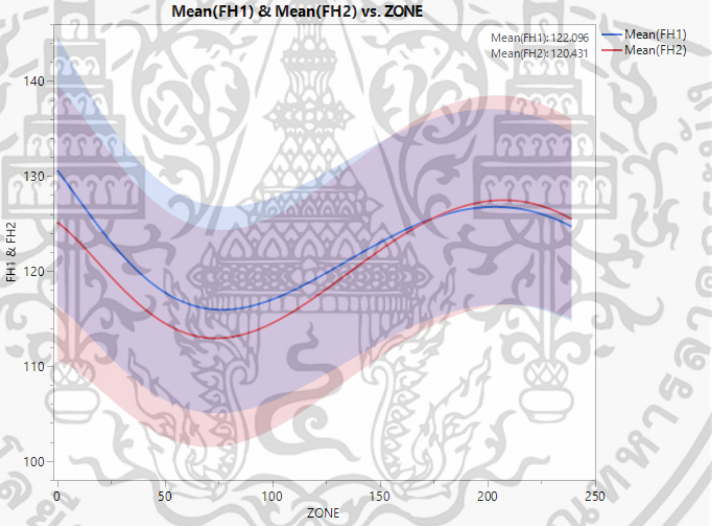
ในกระบวนการผลิตฮาร์ดดิสก์ไดรฟ์ (Hard Disk Drive : HDD) การคำนวณความสูงของการบิน ซึ่งเรียกว่าการวัดการบินของหัวอ่าน/เขียน (Flying Height : FH) เป็นเรื่องที่สำคัญและจำเป็นเพื่อให้เกิดการอ่าน/เขียนได้อย่างมีประสิทธิภาพและเหมาะสมกับฮาร์ดดิสก์ไดรฟ์แต่ละตัว Dakroub et al [1] ได้ทำงานวิจัยเกี่ยวกับการวัดการบินสำหรับฮาร์ดดิสก์ไดรฟ์โดยดูแอมพลิจูดของสัญญาณ B.C. Schardt et al [2] เป็นงานวิจัยที่มีการพัฒนาโดยใช้เทคนิคการวัดการบินโดยการขยับหัวอ่าน/เขียนจากเส้นผ่านศูนย์กลางภายในไปยังเส้นผ่านศูนย์กลางภายนอกของแผ่นดิสก์ V. J. Novotny [3] มีการทำวิจัยโดยใช้วิธีการกำหนดลักษณะแม่เหล็กแบบใหม่ได้รับการพัฒนาขึ้นเพื่อตรวจสอบระยะห่างของหัวอ่าน/เขียนและแผ่นดิสก์ที่แบนด์วิธสูงระหว่างการค้นหาแทร็ค (Track) การจอดของหัวอ่าน/เขียน และระหว่างการสั่นสะเทือนและการกระแทกในขณะที่ฮาร์ดดิสก์ไดรฟ์กำลังทำงานซึ่งจะดูจากการวัดแอมพลิจูดแม่เหล็ก J. Y. Juang et al [4] ได้วิจัยแบบจำลองเชิงตัวเลขไม่เชิงเส้นของหัวอ่าน/เขียน ซึ่งเป็นการจำลองส่วนที่ยื่นออกมาจากหัวอ่าน/เขียนได้อย่างแม่นยำที่มีผลต่อการเปลี่ยนแปลงของโปรไฟล์การบินของหัวอ่าน/เขียนในสภาพแวดล้อมต่างๆ Boettcher et al [5] ได้ศึกษาการวัดการบินแบบไดนามิกพร้อมการควบคุมล่วงหน้าเพื่อปรับการแปรผันให้เหมาะสมกับหัวอ่าน/เขียนของแต่ละหัว

งานวิจัยที่กล่าวไปข้างต้นล้วนมีจุดมุ่งหมายที่จะหาวิธีการให้หัวอ่าน/เขียนมีโปรไฟล์การบินที่เกิดจากการวัดการบินที่ดีและมีประสิทธิภาพสูงสุด ผู้จัดทำวิจัยนี้จึงมีแนวความคิดใหม่ที่จะใช้ปัญญาประดิษฐ์โดยเน้นไปที่โครงข่ายประสาทเชิงลึก (Deep Neural Network : DNN) ร่วมกับการเพิ่มประสิทธิภาพกลุ่มอนุภาค (Particle Swarm Optimization : PSO) [6-15] เพื่อหาจำนวน node ที่เหมาะสมสำหรับแต่ละ layer ของโครงข่ายประสาทเชิงลึก ซึ่งวิธีการนี้นำเสนอวิธีการนวัตกรรมโดยใช้เทคโนโลยีปัญญาประดิษฐ์โดยเฉพาะการใช้โครงข่ายประสาทเชิงลึกร่วมกับเทคนิคการเพิ่มประสิทธิภาพกลุ่มอนุภาคเพื่อปรับปรุงการทำนายให้ดีขึ้นและเพื่อทำนายโปรไฟล์ DAC (Digital-to-Analog-Converter) ซึ่งเป็นหน่วยของการวัดการบินของหัวอ่าน/เขียน เพื่อให้เกิดประสิทธิภาพสูงสุด ด้วยวิธีการนี้สามารถลดจำนวนการวัดการบินและการทำงานของหัวอ่าน/เขียนได้เป็นอย่างมาก ซึ่งจะทำให้กระบวนการวัดการบินเกิดความรวดเร็วเป็นอย่างมาก แต่ยังคงมีประสิทธิภาพที่ดีเทียบเท่าการวัดแบบดั้งเดิม โดยผู้วิจัยมุ่งหวังที่จะลดเวลาการผลิตฮาร์ดดิสก์ไดรฟ์ต่อหน่วยในสายการผลิตให้รวดเร็วขึ้นดังแสดงในรูปที่ 1.1 และโปรไฟล์ DAC สำหรับการวัดการบิน 1 (Flying Height 1 : FH1) และการวัดการบิน 2 (Flying Height 2 : FH2)

ซึ่งใช้เวลาประมาณ 5 ชั่วโมง ดังแสดงในรูปที่ 1.2 ด้วยวิธีการใหม่ที่นำเสนอนี้ เราสามารถลดเวลาการวัดการบิน 2 ด้วยวิธีการปัญญาประดิษฐ์



รูปที่ 1.1 กระบวนการวัด FHs และแนวคิดที่ต้องการลดเวลาการคำนวณความสูงของการบินของ FH2



รูปที่ 1.2 โปรไฟล์ DAC การวัดการบินของ FH1 และ FH2

1.2 วัตถุประสงค์

วัตถุประสงค์ของการวิจัยนี้เพื่อศึกษาการประยุกต์ใช้แบบจำลองปัญญาประดิษฐ์ด้วยวิธีการโครงข่ายประสาทเชิงลึก (Deep Neural Network : DNN) ร่วมกับการเพิ่มประสิทธิภาพกลุ่มอนุภาค (Particle Swarm Optimization : PSO) เพื่อทำนายโปรไฟล์ DAC ของการวัดการบินของหัวอ่าน/เขียน และเพื่อลดเวลาการวัดการบินของหัวอ่าน/เขียนในกระบวนการผลิตฮาร์ดดิสก์ไดรฟ์

1.3 ขอบเขตการวิจัย

1.3.1 ใช้ข้อมูลการวัดการบินของฮาร์ดดิสก์ไดรฟ์เพื่อทำการศึกษาและวิจัย

1.3.2 ศึกษาโครงข่ายประสาทเชิงลึก (Deep Neural Network : DNN) ร่วมกับการเพิ่มประสิทธิภาพกลุ่มอนุภาค (Particle Swarm Optimization : PSO) ให้ทำงานร่วมกัน

1.3.3. ศึกษาการเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาไพธอน (Python)

1.3.4 ใช้ Google Colab เพื่อทำการศึกษาและทำการเทรนนิ่ง (Training) แบบจำลองโครงข่ายประสาทเชิงลึกร่วมกับการเพิ่มประสิทธิภาพกลุ่มอนุภาค

1.4 ประโยชน์ที่คาดหวัง

1.4.1 เพื่อลดเวลาการผลิตฮาร์ดดิสก์ไดรฟ์ต่อหน่วยในสายการผลิต

1.4.2 ลดค่าใช้จ่ายในกระบวนการผลิต

1.4.3 เพื่อประยุกต์ใช้เทคโนโลยีปัญญาประดิษฐ์กับสายการผลิตฮาร์ดดิสก์ไดรฟ์

1.5 การจัดโครงสร้างของเนื้อหาภายในวิทยานิพนธ์

วิทยานิพนธ์นี้ประกอบด้วยเนื้อหาที่เกี่ยวข้องกับเทคนิคการใช้โครงข่ายประสาทเชิงลึก (Deep Neural Network : DNN) ร่วมกับการเพิ่มประสิทธิภาพกลุ่มอนุภาค (Particle Swarm Optimization : PSO) ให้ทำงานร่วมกัน โดยนำแบบจำลองปัญญาประดิษฐ์ไปประยุกต์ใช้ในการคำนวณความสูงของการบินในกระบวนการผลิตฮาร์ดดิสก์ไดรฟ์ (Hard Disk Drive : HDD) โดยสามารถแบ่งเนื้อหาออกเป็น 5 บท ดังนี้

1.5.1 บทที่ 1 บทนำ กล่าวถึงความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์ ขอบเขตการวิจัย ประโยชน์ที่คาดหวัง

1.5.2 บทที่ 2 ทฤษฎี กล่าวถึงภาพรวมกระบวนการทดสอบฮาร์ดดิสก์ไดรฟ์ การคำนวณความสูงของการบินในการผลิตฮาร์ดดิสก์ไดรฟ์ โครงข่ายประสาทเชิงลึก (Deep Neural Network : DNN) การเพิ่มประสิทธิภาพกลุ่มอนุภาค (Particle Swarm Optimization : PSO) งานวิจัยที่เกี่ยวข้อง

1.5.3 บทที่ 3 วิธีดำเนินงานวิจัย กล่าวถึงจุดสนใจของการวัดการบินของหัวอ่าน/เขียน การนำเสนอวิธีการ DNNpso โครงสร้าง Pseudocode ของ DNNpso การจัดเตรียมข้อมูลและการวิเคราะห์ข้อมูล

1.5.4 บทที่ 4 ผลการทดลอง กล่าวถึงผลการและการประเมนผลต่างๆ ประกอบไปด้วย การทดลองหาจำนวน iterations ของ DNNpso การทดลองหาฟังก์ชันวัตถุประสงค์ (R^2) จาก Particle Swarm ของ DNNpso การทดลองการประเมินผลแบบจำลองที่ได้จาก DNNpso กับฮาร์ดดิสก์ไดรฟ์กลุ่มใหม่

1.5.5 บทที่ 5 สรุปผล กล่าวถึงข้อสรุปที่สามารถประยุกต์ใช้กับปัญญาประดิษฐ์กระบวนการผลิตฮาร์ดดิสก์ไดรฟ์ได้อย่างลงตัว และการทำนายทำให้สามารถลดเวลาการการวัดการบินได้เป็นอย่างมาก ทำให้ช่วยลดกระบวนการผลิตของฮาร์ดดิสก์ไดรฟ์ได้อย่างมีนัยสำคัญ

บทที่ 2
ทฤษฎี

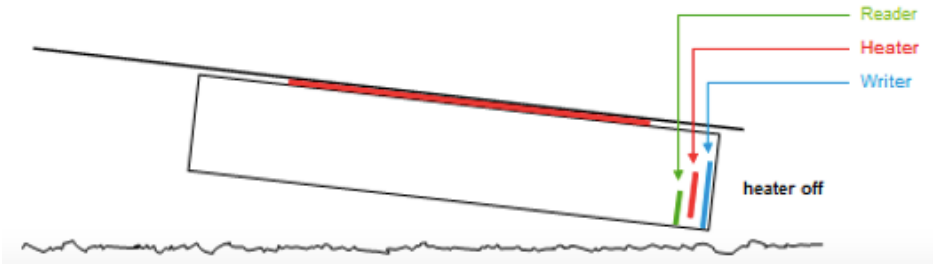
2.1 ภาพรวมกระบวนการทดสอบฮาร์ดดิสก์ไดรฟ์

ในอุตสาหกรรมฮาร์ดดิสก์ไดรฟ์ (Hard Disk Drive) ทุกตัวจะถูกทดสอบตามข้อมูลการระบุของลูกค้า กระบวนการทดสอบเป็นกระบวนการที่ใช้สำหรับแยกแยะฮาร์ดดิสก์ไดรฟ์ที่ไม่ประสบความสำเร็จตามข้อมูลการระบุ เริ่มต้นด้วยการดำเนินการกระบวนการติดตั้ง Printed Circuit Board Assembly (PCBA) บนฐานของฮาร์ดดิสก์ไดรฟ์ซึ่งเรียกว่า Head Disk Assembly (HAD) และจากนั้นฮาร์ดดิสก์ไดรฟ์ที่มีองค์ประกอบที่เสร็จสมบูรณ์ถูกนำมาคำนวณค่าต่างๆ ให้เหมาะสมกับฮาร์ดดิสก์ไดรฟ์แต่ละตัวและทำการทดสอบในเครื่องจักรทดสอบ เช่นตัวอย่างประกอบไปด้วย Flying Height (FH) เป็นการคำนวณความสูงของการบินของหัวอ่าน/เขียนและมีการคำนวณหาพื้นความจุซึ่งเป็นพื้นที่จัดเก็บข้อมูล Storage Capacity (SC) สำหรับฮาร์ดดิสก์ไดรฟ์แต่ละตัวตามลำดับ ฮาร์ดดิสก์ไดรฟ์ที่ผ่านการทดสอบทั้งหมดจะถูกบรรจุและพร้อมจัดส่งไปยังลูกค้าตามที่แสดงในรูปที่ 2.1

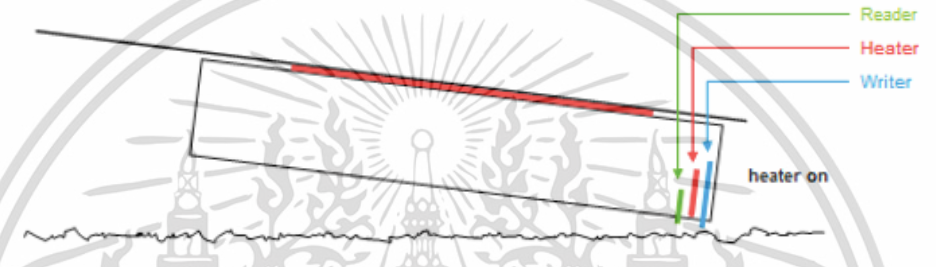
รูปที่ 2.1 ภาพรวมกระบวนการผลิตฮาร์ดดิสก์ไดรฟ์

2.2 การคำนวณความสูงของการบินในการผลิตฮาร์ดดิสก์ไดรฟ์

การคำนวณความสูงของการบิน (Flying Height) ในกระบวนการผลิตฮาร์ดดิสก์ไดรฟ์ (Hard Disk Drive) ใช้เทคโนโลยีการควบคุมความสูงในการบินด้วยความร้อนเพื่อวัดระยะห่างระหว่างหัวอ่าน/เขียนและแผ่นดิสก์โดยการจ่ายพลังงานให้กับองค์ประกอบที่ทำหน้าที่เป็นตัวทำความร้อน (Heater Element) ของหัวอ่าน/เขียนจนกว่าจะถึงเป้าหมายความสูงของการบิน (Target Flying Height) **รูปที่** 2.2 (ก) แสดงให้เห็นว่าหากไม่มีการจ่ายพลังงานให้กับองค์ประกอบตัวทำความร้อน หัวอ่าน/เขียนจะห่างจากแผ่นดิสก์ รูปที่ 2.2 (ข) แสดงองค์ประกอบของหัวอ่าน/เขียนที่เข้าใกล้กับแผ่นดิสก์ขณะที่จ่ายพลังงานกับองค์ประกอบที่ทำหน้าที่เป็นตัวทำความร้อน และรูปที่ 2.3 เป็นภาพตัดขวางของหัวอ่าน/เขียนและแผ่นดิสก์แสดงให้เห็นการเปลี่ยนแปลงของส่วนที่ยื่นออกมาเนื่องจากความร้อนเมื่อมีการจ่ายพลังงานเข้าตัวทำความร้อน

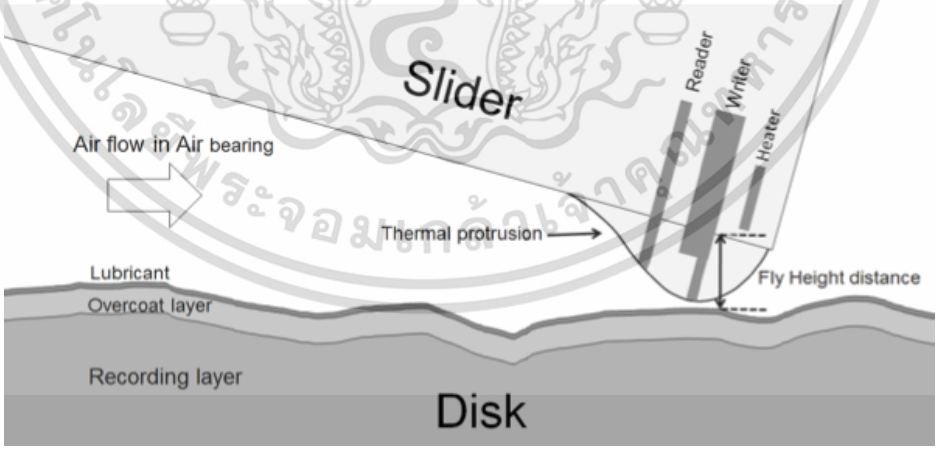


(ก)



(ข)

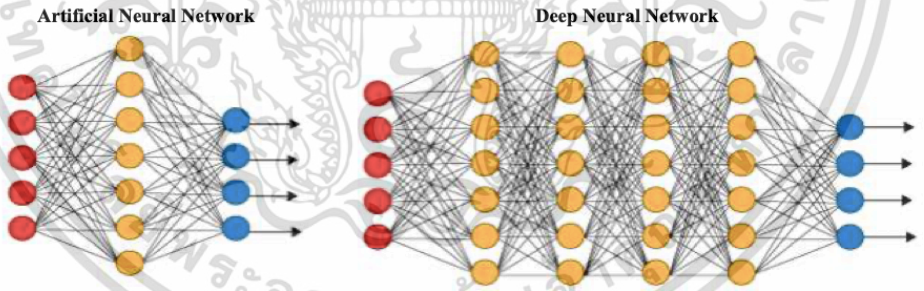
รูปที่ 2.2 (ก) องค์ประกอบของหัวอ่าน/เขียนที่ไม่มีการจ่ายพลังให้กับองค์ประกอบตัวทำความร้อน (ข) องค์ประกอบหัวอ่าน/เขียนเมื่อมีการจ่ายพลังงานให้กับองค์ประกอบตัวทำความร้อน [16]



รูปที่ 2.3 ภาพตัดขวางของหัวอ่าน/เขียนและแผ่นดิสก์แสดงให้เห็นการเปลี่ยนแปลงของส่วนที่ยื่นออกมาเนื่องจากความร้อนเมื่อมีการจ่ายพลังงานเข้าตัวทำความร้อน [17]

2.3 โครงข่ายประสาทเชิงลึก (Deep Neural Network : DNN)

โครงข่ายประสาทเทียม (Artificial Neural Network : ANN) หรือโครงข่ายประสาทเทียมแบบดั้งเดิมที่เรียบง่ายมีจุดมุ่งหมายเพื่อแก้ไขงานเล็กๆ น้อยๆ ด้วยโครงร่างเครือข่ายที่ตรงไปตรงมา โครงข่ายประสาทเทียมได้รับแรงบันดาลใจมาจากโครงข่ายประสาทเทียมทางชีววิทยา มันคือชุดของเลเยอร์ (Layers) เพื่อทำงานเฉพาะอย่าง แต่ละเลเยอร์ประกอบด้วยชุดของโหนด (Nodes) ที่จะทำงานร่วมกัน เครือข่ายเหล่านี้มักจะประกอบด้วยเลเยอร์อินพุต เลเยอร์ที่ซ่อนอยู่หนึ่งถึงสองชั้น และเลเยอร์เอาท์พุต แม้ว่าจะสามารถแก้คำถามทางคณิตศาสตร์ง่ายๆ และปัญหาคอมพิวเตอร์ได้ รวมถึงโครงสร้างเกตพื้นฐานด้วยตารางความจริงตามลำดับ แต่เป็นเรื่องยากสำหรับเครือข่ายเหล่านี้ที่จะแก้ปัญหาการประมวลผลภาพที่ซับซ้อน คอมพิวเตอร์วิชัน (Computer Vision) และงานการประมวลผลภาษาธรรมชาติ (Natural Language) สำหรับปัญหาเหล่านี้ เราใช้โครงข่ายประสาทเชิงลึก ซึ่งมักจะมีโครงสร้างเลเยอร์ที่ซ่อนอยู่ที่ซับซ้อนซึ่งมีเลเยอร์ที่แตกต่างกันมากมาย เช่น เลเยอร์คอนโวลูชัน (Convolutional Layer) เลเยอร์แมกซ์พูลลิง (Max-Pooling Layer) เลเยอร์เดนซ์ (Dense Layer) และเลเยอร์เฉพาะอื่น ๆ เลเยอร์เพิ่มเติมเหล่านี้ช่วยให้โมเดลเข้าใจปัญหาได้ดีขึ้น และมอบแนวทางแก้ไขที่เหมาะสมที่สุดให้กับโครงการที่ซับซ้อน โครงข่ายประสาทเชิงลึกมีเลเยอร์มากกว่า (ความลึกมากกว่า) ANN และแต่ละเลเยอร์จะเพิ่มความซับซ้อนให้กับโมเดล ในขณะเดียวกันก็ทำให้โมเดลสามารถประมวลผลอินพุตได้อย่างกระชับเพื่อให้ได้เอาท์พุตโซลูชันในอุดมคติ [20]

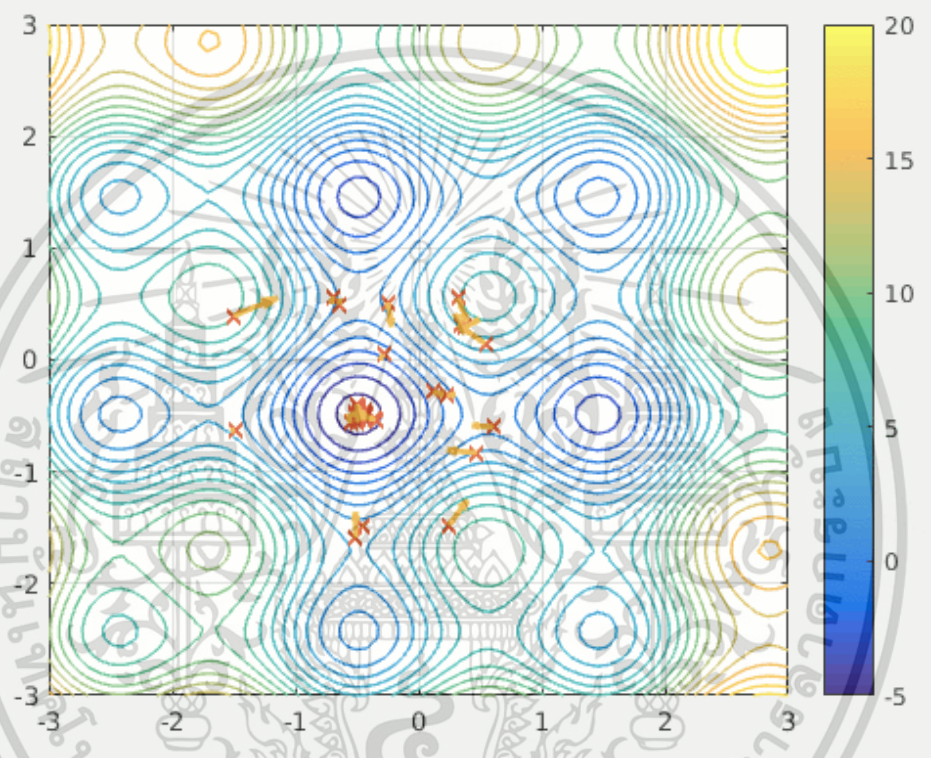


รูปที่ 2.4 โครงข่ายประสาทเทียมเปรียบเทียบกับโครงข่ายประสาทเชิงลึก [21]

2.4 การเพิ่มประสิทธิภาพกลุ่มอนุภาค (Particle Swarm Optimization : PSO)

การเพิ่มประสิทธิภาพกลุ่มอนุภาค (Particle Swarm Optimization : PSO) เป็นอัลกอริทึมเมตาฮิวริสติก (Meta-Heuristic Algorithm) ที่มักใช้ในปัญหาการหาค่าเหมาะที่สุดแบบแยกส่วน (Discrete) ต่อเนื่อง (Continuous) และแบบผสมผสาน ได้รับการพัฒนาครั้งแรกโดย Kennedy และ Eberhart ในปี 2544 [19] โดยได้แรงบันดาลใจจากลักษณะการบินของฝูงนก ในบริบทของ PSO โซลูชัน

เดี่ยวเรียกว่าอนุภาค และการรวมตัวของโซลูชันทั้งหมดเรียกว่า ฝูง แนวคิดหลักใน PSO คือแต่ละอนุภาคมีความรู้เกี่ยวกับความเร็ว (Velocity) ปัจจุบันเท่านั้น ซึ่งถือเป็นการตั้งค่าที่ดีที่สุดของมันเองที่ทำได้ในอดีต (pBest) และอนุภาคใดที่ดีที่สุดจากทั้งหมดในปัจจุบันในฝูง (gBest) ในการวนซ้ำทุกครั้ง แต่ละอนุภาคจะปรับความเร็วในลักษณะที่ตำแหน่งใหม่จะใกล้กับ gBest และ pBest มากขึ้นในเวลาเดียวกัน [7]

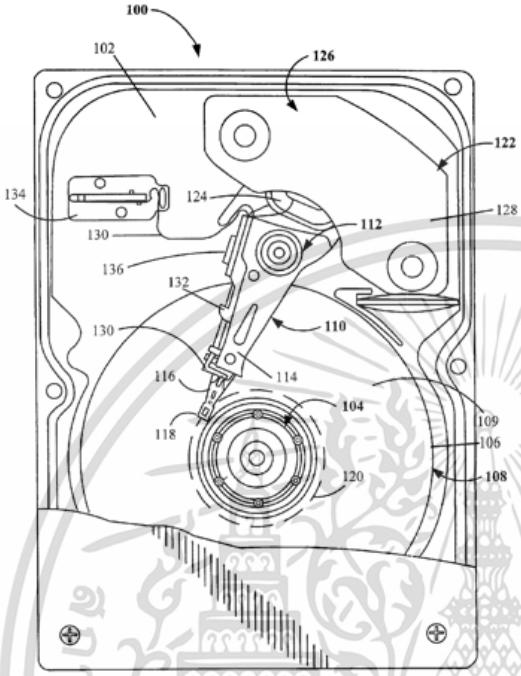


รูปที่ 2.5 ฝูงอนุภาคค้นหาค่าต่ำสุดของโกลบอลฟังก์ชัน [22]

2.5 งานวิจัยที่เกี่ยวข้อง

Dakroub et al [1] วิธีการวัดความสูงการบินของหัวอ่าน/เขียนเพื่อสร้างโปรไฟล์ความสูงความสูงการบินสำหรับใช้ในการวัดความสูงการบินของหัวอ่าน/เขียนที่สัมพันธ์กับดิสก์ของดิสก์ไดรฟ์ สัญญาณเขียนจากแหล่งคงที่ถูกนำไปใช้กับองค์ประกอบของหัวอ่าน/เขียน สร้างสนามแม่เหล็กฟลักซ์ด้วยส่วนสนามแม่เหล็กสเตรย์องค์ประกอบความต้านทานสนามแม่เหล็กของหัวอ่าน/เขียนคู่กับสนามฟลักซ์แม่เหล็กสร้างสัญญาณความแรงของสนามเริ่มต้นขณะที่แผ่นดิสก์หมุน หัวอ่าน/เขียนจะลอยไปด้านบนเพื่อลดความหนาแน่นของส่วนสนามแม่เหล็กสเตรย์ ซึ่งส่งผลให้แอมพลิจูดของสัญญาณความแรงของสนามลดลง แอมพลิจูดที่ลดลงของสัญญาณความแรงของสนามจะถูกปรับเทียบเป็นโปรไฟล์ความสูงของ

การบินที่กำหนดไว้ล่วงหน้าเพื่อเชื่อมโยงความสูงของการบินของหัวอ่าน/เขียนกับความกว้างที่ลดลงของความแรงของสนาม ซึ่งสร้างโปรไฟล์ความสูงของการบิน



(ก)

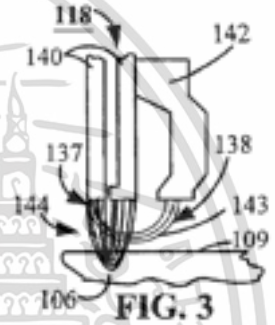
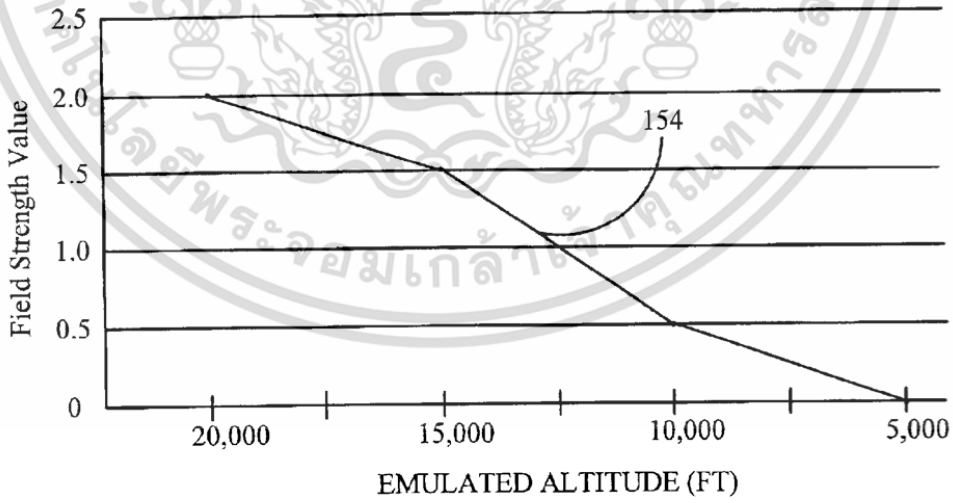


FIG. 3

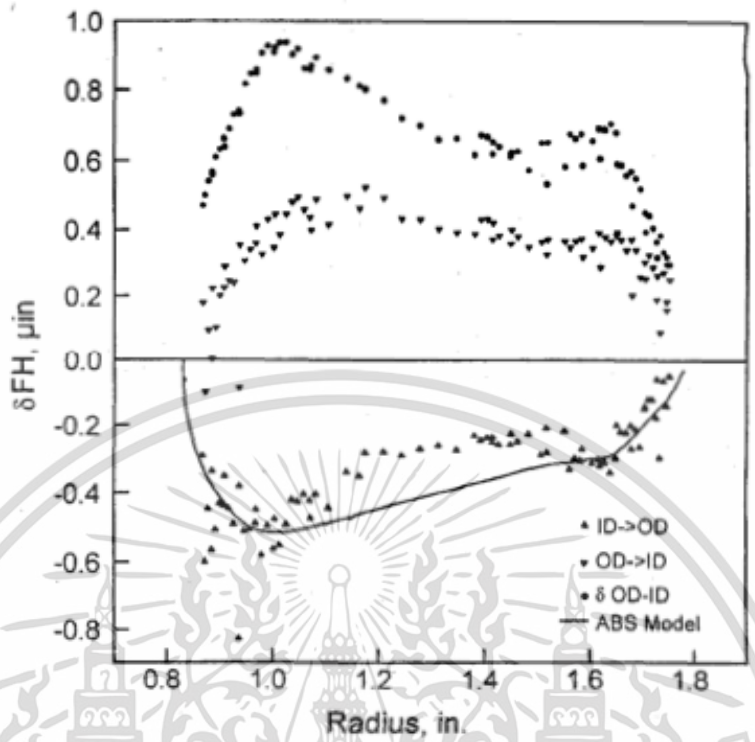
(ข)



(ค)

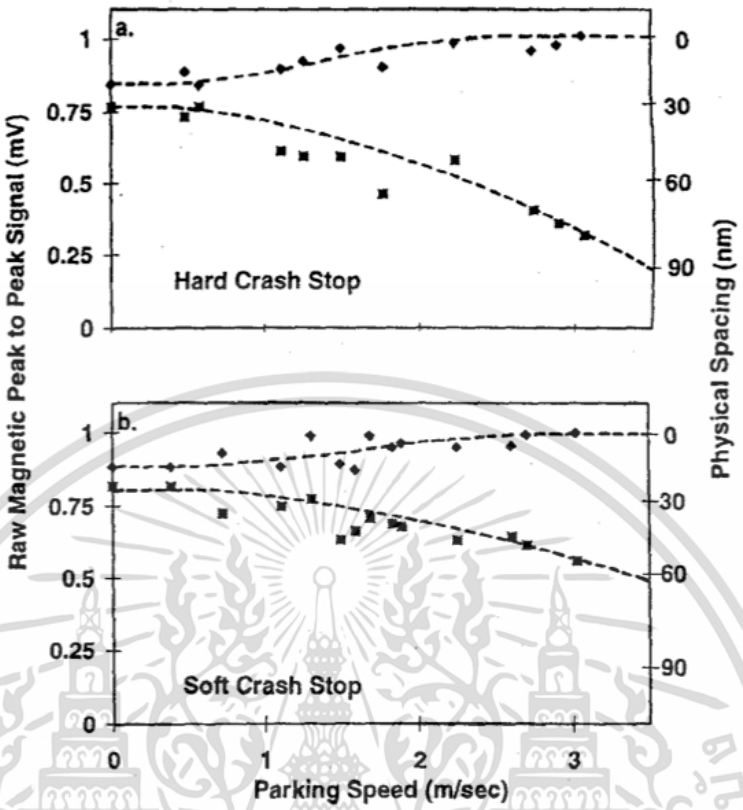
รูปที่ 2.6 (ก) มุมมองด้านบนของดิสก์ไดรฟ์พร้อมกับความสูงในการบิน (ข) มุมมองด้านข้างของหัวอ่าน/เขียนที่กำลังทำงานในการบิน (ค) ค่าความแข็งแรงของสนามแม่เหล็กไฟฟ้าในระดับความสูงในหัวอ่าน/เขียน [1]

B.C. Schardt et al [2] เทคนิคใหม่ได้รับการพัฒนาเพื่อวัดการเปลี่ยนแปลงความสูงของการบินระหว่างการหาความเร็วสูงในฮาร์ดดิสก์ไดรฟ์ที่ทำงานได้อย่างสมบูรณ์ ในระหว่างการหาการเปลี่ยนแปลงความสูงในการบินของหัวอ่าน/เขียนเป็นผลมาจากการเปลี่ยนแปลงทิศทางและความเร็วของการไหลของอากาศใต้ Air Bearing Surface (ABS) การประยุกต์ใช้สูตรการสูญเสียระยะห่างวอลเลซ (Wallace spaceing loss formula) กับแอมพลิจูดสูงสุดของสัญญาณบันทึกแม่เหล็กที่ได้มาเมื่อส่วนหัวข้ามแทร็กระหว่างการหาเป็นพื้นฐานของวิธีการนี้ วิธีนี้ยังต้องใช้วิธีวัดตำแหน่งอย่างแม่นยำตามเส้นทางที่เกิดการข้ามแต่ละแทร็ก ข้อมูลเพิ่มเติมนี้ใช้เพื่อลบความแปรผันของแอมพลิจูดของสัญญาณแม่เหล็กอ่านกลับ ซึ่งเกิดจากความขรุขระของพื้นผิวและความไม่สม่ำเสมอของสนามแม่เหล็กของพื้นผิวดิสก์ และความจริงที่ว่าการดำเนินการค้นหาแต่ละครั้งข้ามแทร็กในตำแหน่งเส้นรอบวงที่แตกต่างกัน การพล็อตค่าแอมพลิจูดสูงสุดเทียบกับตำแหน่งที่ข้ามแทร็กสำหรับการค้นหาหลายพันครั้งทำให้ได้โปรไฟล์ของการเปลี่ยนแปลงความหยาบของพื้นผิวดิสก์ตลอดระยะสั้นๆ ของแทร็ก ยกเว้นการชดเชยแอมพลิจูดคงที่ โปรไฟล์แทร็กที่ได้รับเมื่อข้ามแทร็กเฉพาะจากเส้นผ่านศูนย์กลางภายใน (ID) ไปยังเส้นผ่านศูนย์กลางภายนอก (OD) ของดิสก์จะเหมือนกับที่ได้มาระหว่างทิศทาง OD ถึง ID การชดเชยแอมพลิจูดนี้ทำให้สามารถคำนวณความแตกต่างของความสูงของการบินได้ระหว่างสามกรณีของการค้นหา ID ถึง OD, การค้นหา OD ถึง ID เทคนิคการวัดนี้สามารถแก้ไขการเปลี่ยนแปลงความสูงของการบินไดนามิกที่ 0.05 พินได้อย่างน่าเชื่อถือ การเปรียบเทียบข้อมูลที่วัดได้กับการจำลอง Air Bearing เชิงตัวเลขจึงถูกนำเสนอ



รูปที่ 2.7 การเปลี่ยนแปลงความสูงในการบินขณะการค้นหา (Seek) เป็นฟังก์ชันของรัศมี จุดแต่ละจุดคือข้อมูลที่วัดได้ เส้นทึบคือผลลัพธ์ของการจำลอง Air Bearing ระหว่างการค้นหา (Seek) ระหว่าง ID ถึง OD [2]

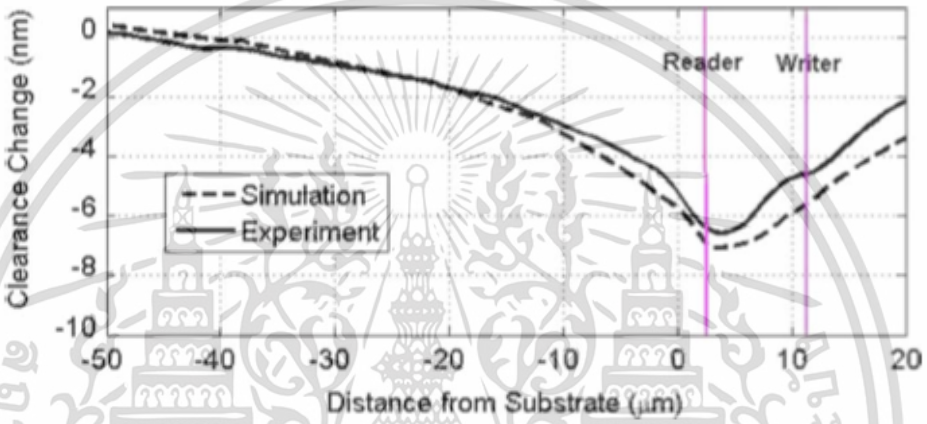
V. J. Novotny [3] เทคนิคการระบุลักษณะแม่เหล็กแบบใหม่ได้รับการพัฒนาขึ้นเพื่อตรวจสอบระยะห่างของหัวอ่าน/เขียนและดิสก์ที่แบนด์วิธสูงระหว่างการค้นหา (Seeking) การจอด (Parking) และระหว่างการสั่นสะเทือนและการกระแทกในการทำงาน เทคนิคนี้อาศัยการวัดแอมพลิจูดแม่เหล็กของรูปแบบความถี่คงที่ที่เขียนบนแทร็ก (Track) ทุกวินาทีและจากการอ่านค่าของเกรย์โค้ด (Graycode) และเซอร์โวเบิร์ส (Sevo Bursts) ระหว่างการค้นหาหรือการจอด ข้อมูลแม่เหล็กที่ได้รับระหว่างการข้ามแทร็กแม่เหล็กนับพันจะถูกวิเคราะห์ ด้วยหัวอ่าน/เขียนของไดรฟ์แบบเหนี่ยวนำและแมกนีโตรีซิสทีฟ (magnetoresistive) หัวอ่าน/เขียนของไดรฟ์ที่ออกแบบมาอย่างดีในโหมดการค้นหาปกติ ตรวจพบความผันผวนของระยะห่างระหว่างหัวอ่าน/เขียนและดิสก์ +/- 5 นาโนเมตร ความผันผวนขนาดใหญ่ของระยะห่างระหว่างหัวอ่าน/เขียนและดิสก์ถูกตรวจพบที่แครชสต็อบ (Crash Stop) และการจอดที่ความเร็วเกิน 0.5 m/sec



รูปที่ 2.8 เส้นในแต่ละกราฟแสดงถึงขอบเขตบนและล่างของแอมพลิจูดจากจุดสูงสุดของแม่เหล็กถึงจุดสูงสุด และระยะห่างระหว่างหัวอ่าน/เขียนกับดิสก์ที่สอดคล้องกัน ซึ่งเป็นฟังก์ชันของความเร็วในการเริ่มต้นการจอดหลังจากการชนกับ a. hard crash stop และ b. soft crash stop ในไดรฟ์ขนาด 95 มิลลิเมตร [3]

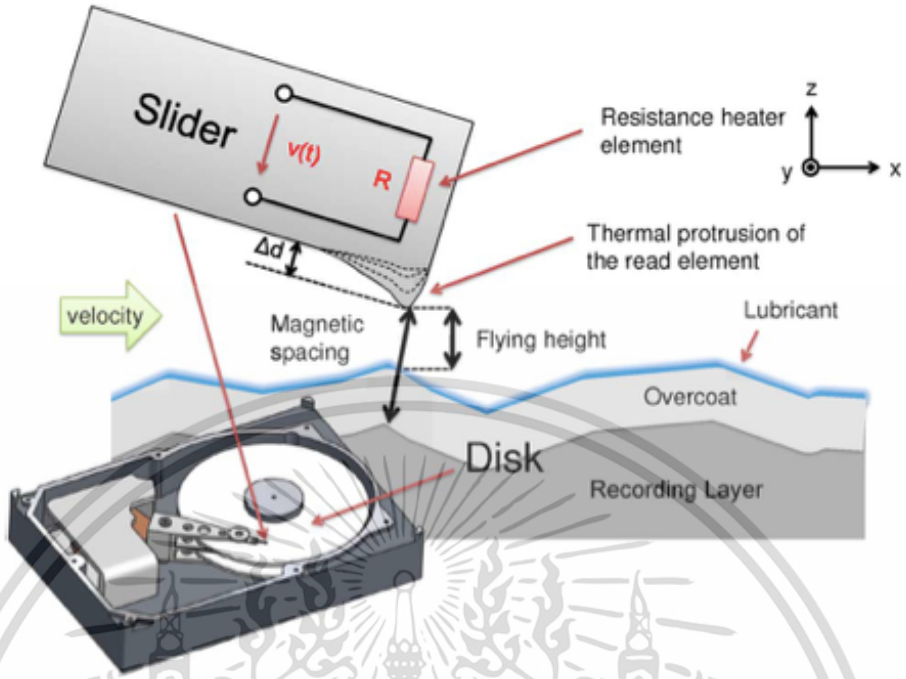
J. Y. Juang et al [4] ฮาร์ดไดรฟ์ที่มีสไลเดอร์ (Sliders) พร้อมการควบคุมความสูงการบิน (Flying Height) ที่ใช้งานอยู่โดยใช้การขยายตัวด้วยความร้อนขององค์ประกอบความร้อนเพิ่งเกิดขึ้นไม่นานมานี้นำมาใช้ในผลิตภัณฑ์ แนวทางนี้ช่วยให้สามารถชดเชยการเปลี่ยนแปลงความสูงการบินแบบคงที่และบรรลุการบินที่ต่ำกว่า 3 นาโนเมตร ระหว่างการดำเนินการอ่าน/เขียน บทความนี้อธิบายแบบจำลองเชิงตัวเลขแบบไม่เชิงเส้นของหัวบันทึกแม่เหล็กแบบตั้งฉากสำหรับการจำลองที่แม่นยำของส่วนที่ยื่นออกมาที่ปลายขั้ว (Pole-Tip) และผลกระทบต่อการเปลี่ยนแปลงความสูงการบินภายใต้สภาวะต่างๆ เช่น ที่อุณหภูมิไดรฟ์สูงขึ้น เมื่อเปิดใช้ฮีตเตอร์ (Heater) หรือระหว่างเขียน แบบจำลองนี้รวมโมเดลไฟไนต์เอลิเมนต์ (Finite-Element) แบบไฟฟ้าเชิงกลทางความร้อน (Electrical-Thermomechanical) ของสไลเดอร์และ Full Air-Bearing Solver และรวมถึงส่วน Pole-Tip ที่เหลื่อมกัน และสไลเดอร์/ดิสก์ที่เสียรูปเนื่องจากแรงดันอากาศ เราสามารถคาดการณ์พารามิเตอร์หลักที่ไม่สามารถวัดได้ง่ายๆ (เช่น ค่าขั้นต่ำ/

การอ่าน/การเขียนของความสูงการบิน โปรไฟล์ที่ยื่นออกมาที่แตกต่างกันสำหรับอุณหภูมิห้อง, การทำงานของฮีตเตอร์ และระหว่างการเขียน) นอกจากนี้ เรายังนำเสนอวิธีการทดลองแบบใหม่สำหรับการวัดส่วนที่ยื่นออกมาและโปรไฟล์เดลต้าของระยะห่างด้วยความละเอียดระดับอังสตรอม (Angstrom) ผลลัพธ์ของแบบจำลองจะถูกนำไปเปรียบเทียบกับข้อมูลการทดลองภายใต้เงื่อนไขการทดสอบต่างๆ ซึ่งแสดงให้เห็นที่เงื่อนต่างๆ อย่างยอดเยี่ยม จากวิธีนี้ เราสามารถประเมินและเพิ่มประสิทธิภาพการออกแบบฮีตเตอร์หัวอ่าน\เขียน และ ABS ที่แตกต่างกันได้อย่างรวดเร็ว

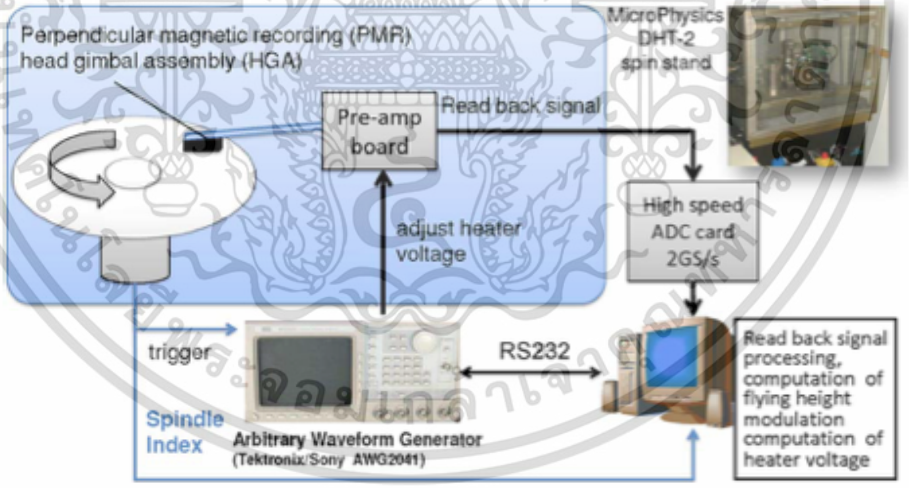


รูปที่ 2.9 การเปรียบเทียบการเปลี่ยนแปลงระยะห่างระหว่างการจำลองและการวัดความสูงการบินแบบออปติคัลภายใต้สภาวะการบิน (ทำความร้อน: 65 mW) [4]

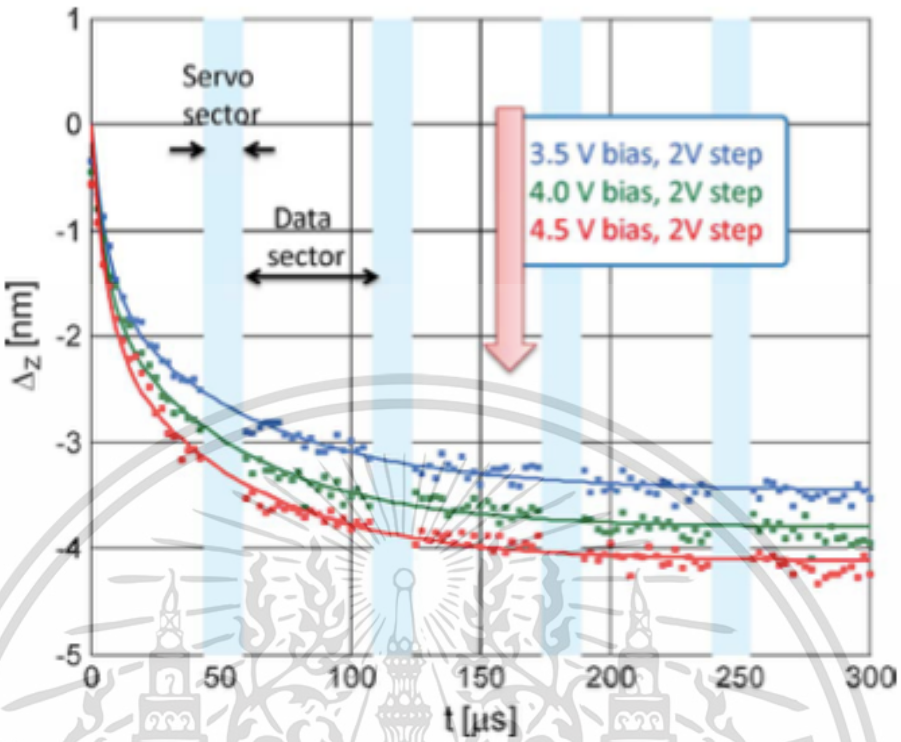
U. Boettcher et al [5] แบบจำลองไดนามิกขององค์ประกอบฮีตเตอร์ (Heater) ความต้านทานในการควบคุมความสูงของการบินด้วยความร้อน (TFC) ในสไลเดอร์ (Slider) ของฮาร์ดดิสก์ไดรฟ์ได้รับการระบุและใช้สำหรับการควบคุมความสูงของการบินแบบไดนามิก ข้อมูลการทดลองที่ได้รับจากแท่นหมุนและอัลกอริทึมการรับรู้ทั่วไปจะใช้สำหรับการระบุแบบจำลองไดนามิกแบบเวลาไม่ต่อเนื่อง (Discrete-Time) ของตัวกระตุ้นความร้อน (Thermal Actuator) การเปลี่ยนแปลงความสูงของการบินวัดได้สองวิธี โดยใช้ข้อมูลของเซอร์โวเบิร์ส (Servo Burst) ที่เขียนลงบนพื้นผิวดิสก์และใช้ข้อมูลที่เขียนในเซคเตอร์ (Sector) ข้อมูลเท่านั้น วัดการเปลี่ยนแปลงความต้านทานของตัวกระตุ้นความร้อนตามระดับพลังงานที่ป้อนเข้ามา ตามแบบจำลองเวลาไม่ต่อเนื่องที่ระบุของเครื่องทำความร้อนและการใช้เทคนิคการปรับให้เหมาะสมคอนเวกซ์ (Convex Optimization) แผนภาพการคำนวณได้รับการเสนอเพื่อให้ได้โปรไฟล์อินพุตป้อนที่เหมาะสมส่งต่อไปยังองค์ประกอบตัวทำความร้อน ซึ่งลดการเปลี่ยนแปลงความสูงของการบินซ้ำและสามารถใช้งานความสูงในการบินต่ำ



(ก)



(ข)

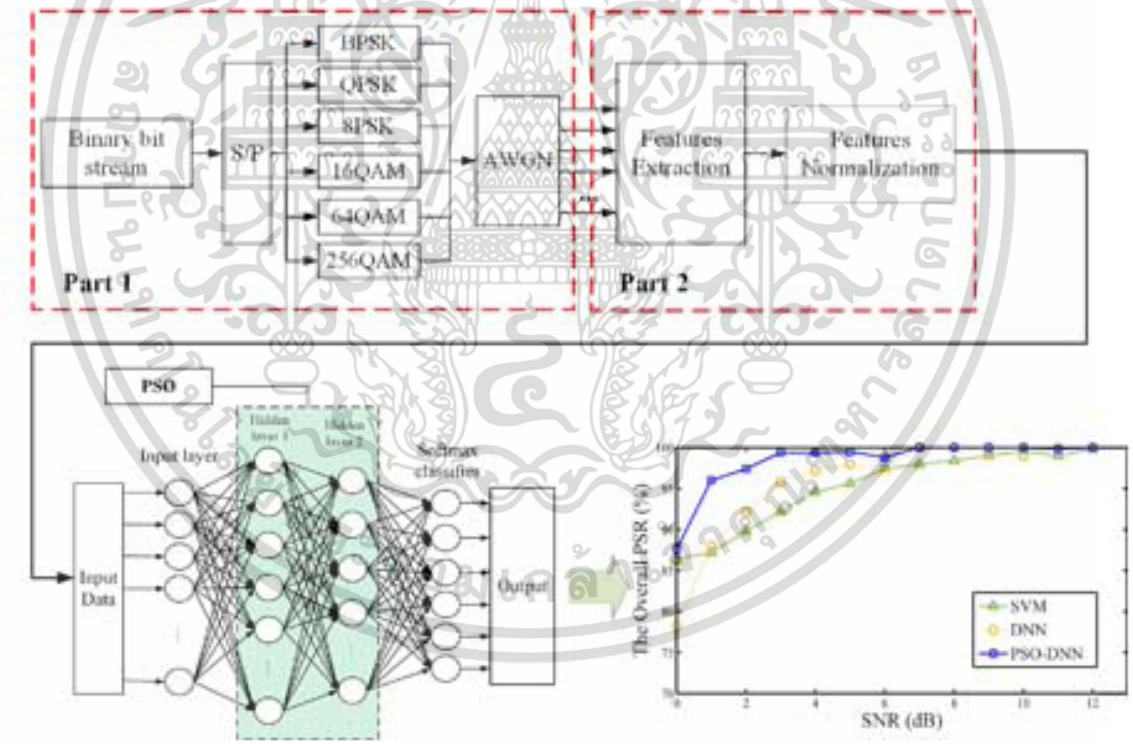


(ค)

รูปที่ 2.10 (ก) ฮาร์ดดิสก์ไดรฟ์และมุมมองด้านข้างส่วนท้ายของ Slider พร้อมองค์ประกอบตัวต้านทานของตัวทำความร้อนเพื่อการควบคุมความสูงของการบินด้วยความร้อน (ข) แผนผังการติดตั้งขาตั้งการทดลองการหมุน (ค) การตอบสนองแบบไดนามิกของความสูงในการบินสัมพันธ์กับระดับการอินพุตแรงดันไฟฟ้าที่วัดโดยใช้เซกเตอร์ข้อมูล (Data Secotrs) ที่อัตราการสุ่มตัวอย่างที่มีประสิทธิผล 380 kHz และแบบจำลอง 2nd order โดยประมาณ (เส้นทึบ) [5]

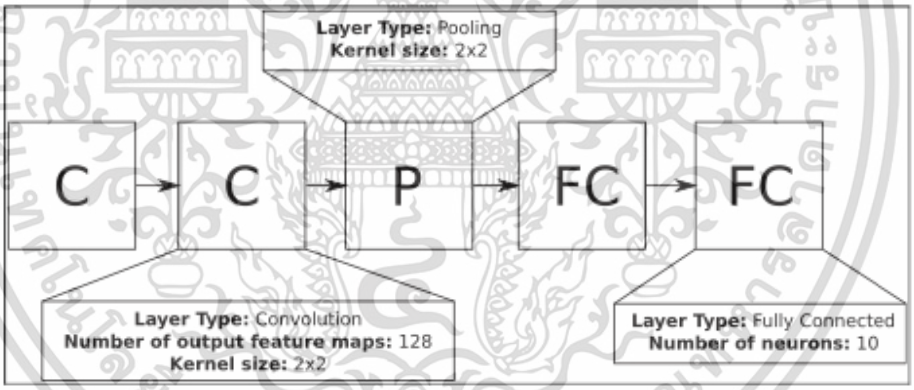
W. SHI et al [6] การจดจำการมอดูเลต (Modulation Recognition) เป็นงานหลักในระบบสื่อสารไร้สายหลายระบบ รวมถึงวิทยุสื่อสารและการตรวจตราสัญญาณ ความหลากหลายของรูปแบบการมอดูเลตและความซับซ้อนที่เพิ่มขึ้นของสภาพแวดล้อมช่องสัญญาณทำให้มีความต้องการที่สูงขึ้นในการระบุสัญญาณมอดูเลตที่ถูกต้อง การเรียนรู้เชิงลึก (Deep Learning) ถือเป็นทางออกที่มีศักยภาพในการแก้ปัญหาเหล่านี้ เนื่องจากความสามารถในการประมวลผลข้อมูลขนาดใหญ่และการจำแนกประเภทที่เหนือกว่า งานวิจัยนี้นำเสนอวิธีการจดจำการมอดูเลตแบบดิจิทัลที่มีประสิทธิภาพโดยยึดตามแบบจำลองโครงข่ายประสาทเทียมเชิงลึก (Deep Neuron Network : DNN) นอกจากนี้ เรานำเสนออัลกอริทึมการเพิ่มประสิทธิภาพการจับกลุ่มอนุภาค (Particle Swarm Optimization : PSO) เพื่อเพิ่ม

ประสิทธิภาพการหาจำนวนโหนด (Nodes) ของฮิดเดนเลเยอร์ (Hidden Layer) ของโครงข่ายประสาทเทียมเชิงลึก เพื่อแก้ปัญหาที่โครงข่ายประสาทเทียมเชิงลึกดั้งเดิมและจำนวนโหนดของของฮิดเดนเลเยอร์จำเป็นต้องกำหนดด้วยตนเอง ในบทความนี้ เราใช้วิธี PSO-DNN ที่เสนอเพื่อเรียนรู้ลักษณะสัญญาณมอดูเลตที่เพิ่มโดย additive white Gaussian noise (AWGN) และเพื่อฝึกเครือข่าย ซึ่งสามารถปรับปรุงประสิทธิภาพการจดจำภายใต้เงื่อนไขของสัญญาณต่ำของอัตราส่วนสัญญาณรบกวน (Signal-To-Noise Ratio : SNR) ผลการทดลองแสดงให้เห็นว่าอัตราการรู้จำของอัลกอริทึมนี้ดีขึ้น 9.4% และ 8.8% เมื่อเทียบกับวิธีการที่ใช้โครงข่ายประสาทเทียมเชิงลึกแบบเดิมและ Support Vector Machine (SVM) เมื่อ SNR เท่ากับ 0 และ 1 dB ตามลำดับ นอกจากนี้ การทดลองอื่นที่เปรียบเทียบกับ Genetic Algorithm (GA) ยังพิสูจน์ให้เห็นว่าอัลกอริทึมที่เราเสนอนั้นมีประสิทธิภาพมากกว่าในการปรับโครงข่ายประสาทเทียมเชิงลึกให้เหมาะสม วิธีการที่เสนอนั้นง่ายต่อการนำไปใช้เพื่อให้มีโอกาสในการพัฒนาที่กว้างขวางในการจดจำการมอดูเลต

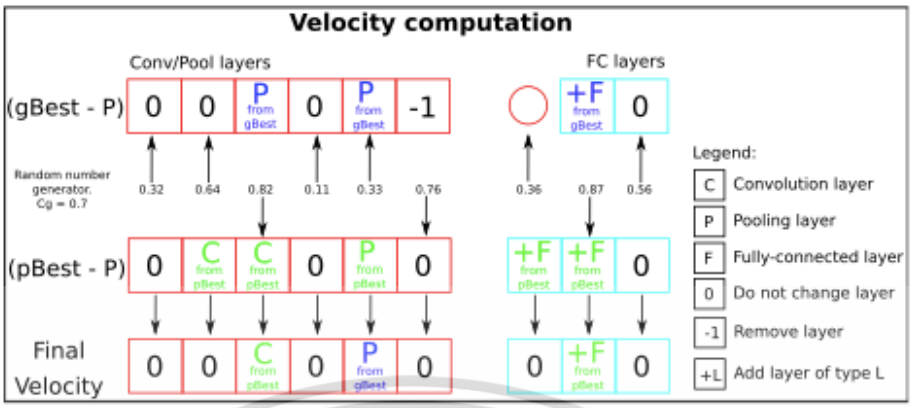


รูปที่ 2.11 แนวคิดโดยรวมการเพิ่มประสิทธิภาพการจับกลุ่มอนุภาคร่วมกับโครงข่ายประสาทเทียมเชิงลึกสำหรับการจดจำการมอดูเลตแบบดิจิทัล [6]

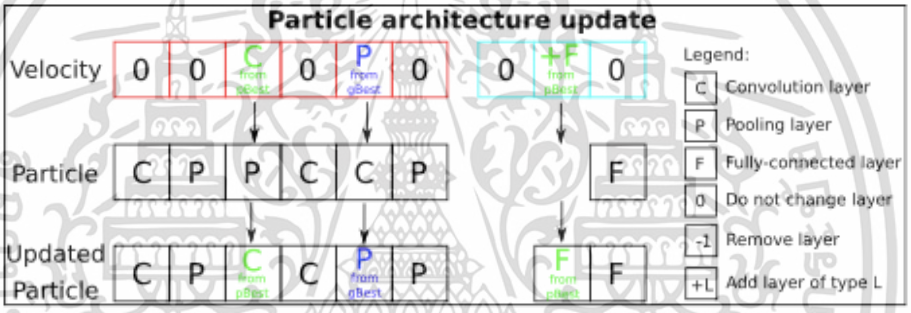
F. E. Fernandes Junior et al [7] โครงข่ายประสาทเทียมเชิงลึก (Deep Neural Networks) ได้รับการพิสูจน์แล้วว่ามีประสิทธิภาพเหนือกว่าอัลกอริทึมการเรียนรู้ของเครื่อง (Machine Learning Algorithms) แบบดั้งเดิมในการแก้ปัญหาในโลกแห่งความเป็นจริง อย่างไรก็ตาม โครงข่ายประสาทเทียมเชิงลึกที่ประสบความสำเร็จมากที่สุดนั้นถูกสร้างขึ้นด้วยมือตั้งแต่เริ่มต้น โดยคำนึงถึงโดเมนความรู้ของปัญหา วิธีการนี้มักใช้เวลาและทรัพยากรในการคำนวณอย่างมาก ในงานวิจัยนี้ เราเสนออัลกอริทึมแบบใหม่ที่อิงจากการเพิ่มประสิทธิภาพการจับกลุ่มของอนุภาค (Particle Swarm Optimization) ซึ่งสามารถลู่เข้าหากันได้อย่างรวดเร็วเมื่อเปรียบเทียบกับแนวทางวิวัฒนาการอื่นๆ เพื่อค้นหาสถาปัตยกรรมโครงข่ายประสาทเทียม (Convolutional Neural Networks) โดยอัตโนมัติสำหรับงานจำแนกภาพ (Image Classification) ชื่อ psoCNN ด้วยกลยุทธ์การเข้ารหัสแบบใหม่โดยตรงและตัวดำเนินการความเร็วได้รับการคิดค้นเพื่อให้สามารถใช้การเพิ่มประสิทธิภาพการจับกลุ่มของอนุภาคร่วมกับค้นหาสถาปัตยกรรมโครงข่ายประสาทเทียมเชิงลึกได้อย่างเหมาะสม ผลการทดลองของเราแสดงให้เห็นว่า psoCNN สามารถค้นหาสถาปัตยกรรมโครงข่ายประสาทเทียมที่ดีได้อย่างรวดเร็ว ซึ่งให้ประสิทธิภาพที่มีคุณภาพเทียบได้กับการออกแบบที่ล้ำสมัย



(ก)



(ข)



(ค)

รูปที่ 2.12 (ก) ตัวแทนของอนุภาคเดี่ยวที่นำเสนอ psoCNN (ข) ตัวอย่างการคำนวณความเร็วของอนุภาคเดี่ยว (ค) ตัวอย่างการปรับปรุงสถาปัตยกรรมของอนุภาค [7]

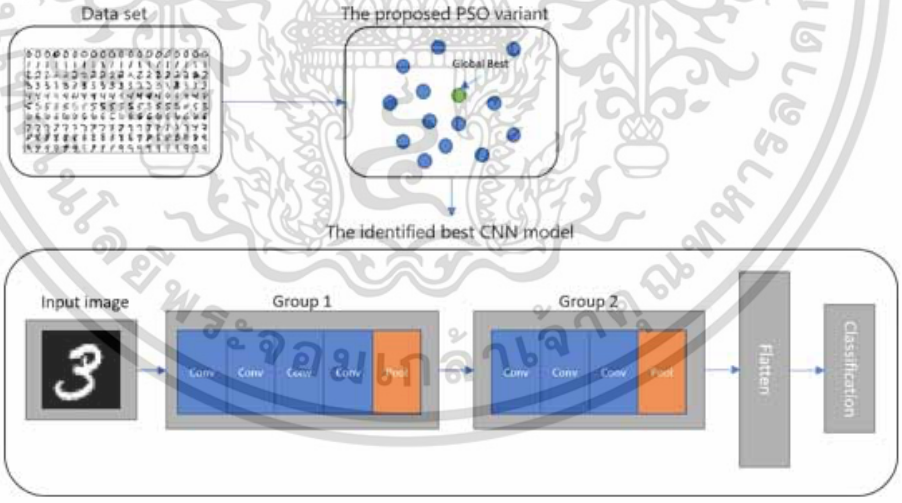
B. A. Garro and R. A. Vazquez [8] การออกแบบโครงข่ายประสาทเทียม (Artificial Neural Network) เป็นงานที่ซับซ้อน เนื่องจากประสิทธิภาพขึ้นอยู่กับสถาปัตยกรรม ฟังก์ชันการถ่ายโอนที่เลือก และอัลกอริทึมการเรียนรู้ที่ใช้ในการฝึกชุดน้ำหนักซินแนปติก (Synaptic Weights) ในงานวิจัยนี้ เรานำเสนอวิธีการที่ออกแบบโครงข่ายประสาทเทียมโดยอัตโนมัติโดยใช้อัลกอริธึมการหาค่าเหมาะที่สุดสำหรับกลุ่มอนุภาค เช่น Basic Particle Swarm Optimization (PSO), Second Generation of Particle Swarm Optimization (SGPSO) และ PSO แบบจำลองใหม่ของการหาค่าเหมาะที่สุดสำหรับกลุ่มอนุภาคเรียกว่า NMPSO จุดมุ่งหมายของอัลกอริธึมเหล่านี้คือการพัฒนาองค์ประกอบหลักสามประการของโครงข่ายประสาทเทียม ชุดของน้ำหนักซินแนปติก การเชื่อมต่อหรือสถาปัตยกรรม และ

ฟังก์ชันการถ่ายโอนสำหรับเซลล์ประสาทแต่ละอัน มีการเสนอฟังก์ชันฟิตเนสที่แตกต่างกันแปดฟังก์ชันเพื่อประเมินความเหมาะสมของแต่ละโซลูชันและค้นหาการออกแบบที่ดีที่สุด ฟังก์ชันเหล่านี้อิงตามค่า Mean Square Error และ Classification Error และใช้กลยุทธ์เพื่อหลีกเลี่ยงการฝึกมากเกินไป (Overtraining) และเพื่อลดจำนวนการเชื่อมต่อในโครงข่ายประสาทเทียมที่ออกแบบด้วยวิธีการที่เสนอนั้นจะถูกนำไปเปรียบเทียบกับที่ออกแบบด้วยตนเองโดยใช้อัลกอริทึมการเรียนรู้ Back-Propagation และ Levenberg-Marquardt ที่รู้จักกันดี สุดท้ายนี้ ความแม่นยำของวิธีการจะถูกทดสอบด้วยปัญหาการจำแนกรูปแบบแบบไม่เชิงเส้นที่แตกต่างกัน

P. Konghuayrob and S. Kaitwanidvilai [9] ความต้องการความจุในการจัดเก็บข้อมูลเพิ่มขึ้นอย่างมาก ตามเทคโนโลยี Heat-assisted Magnetic Recording (HAMR) แนวโน้มความต้องการของฮาร์ดดิสก์ไดรฟ์ (Hard Disk Drive) คาดการณ์ว่าความหนาแน่นของพื้นที่จะบรรลุถึง 10 Tbit/in^2 ก่อนปี 2020 ความหนาแน่นของพื้นที่ที่สูงส่งผลให้แทร็ก (Track) แคบลง ซึ่งค่อนข้างไวต่อการรบกวนจากภายนอกรวมถึงเสียงรบกวน จุดนี้เป็นปัญหาเกณฑ์มาตรฐานของการออกแบบคอนโทรลเลอร์ (Controller) ที่มีความแม่นยำสูงสำหรับควบคุมกลไกเซอร์โว (Servo) ของฮาร์ดดิสก์ไดรฟ์ นอกจากนี้ ยังต้องคำนึงถึงความไม่แน่นอนอย่างเป็นระบบในขั้นตอนการออกแบบคอนโทรลเลอร์ด้วย งานวิจัยนี้เสนอเมตริก v-gap ทางเลือกที่มีประสิทธิภาพซึ่งเกี่ยวข้องกับ H∞ loop shaping เพื่อทำให้มอเตอร์วอยซ์คอยล์ (Voice Coil Motor) เสถียรในฮาร์ดดิสก์ไดรฟ์ภายใต้สภาพความไม่แน่นอน การเพิ่มประสิทธิภาพการจับกลุ่มอนุภาค (Particle Swarm Optimization) ถูกนำมาใช้เพื่อลดช่องว่างระหว่างระบบและคอนโทรลเลอร์ H∞ และระบบที่มีคำสั่งควบคุม 3 คำสั่ง แทนที่จะใช้คอนโทรลเลอร์ H∞ แบบธรรมดาที่มีลำดับสูงซึ่งมีโครงสร้างซับซ้อน งานวิจัยนี้ใช้ลำดับคอนโทรลเลอร์ที่ต่ำกว่าที่เสนอตาม v-gap ซึ่งเหมาะสมกว่าในการนำไปใช้งานจริง ประสิทธิภาพและความทนทานของคอนโทรลเลอร์ทั้งสองจะถูกเปรียบเทียบในการศึกษาการจำลอง ผลลัพธ์ยืนยันลักษณะที่คล้ายคลึงกันของคอนโทรลเลอร์ทั้งสองในแง่ของการติดตามประสิทธิภาพและการตัดการรบกวน นอกจากนี้ ดัชนีความเสถียรของระบบที่เรียกว่า Stability Margin ที่ 0.472 และเงื่อนไขการทดสอบการกวนของระบบยังเน้นย้ำถึงความแม่นยำและประสิทธิผลของคอนโทรลเลอร์ที่นำเสนอ

T. Lawrence et al [10] การออกแบบการค้นหาสถาปัตยกรรมโครงข่ายประสาทเทียม (Convolutional Neural Networks) ตั้งแต่เริ่มต้นเป็นกระบวนการที่ใช้เวลานานซึ่งต้องใช้ผู้เชี่ยวชาญเฉพาะด้าน ในขณะที่มีการเสนออัลกอริทึมการสร้างสถาปัตยกรรมอัตโนมัติ กลยุทธ์การค้นหาโดยทั่วไปมักใช้การคำนวณขั้นสูงและเปลืองทรัพยากร วิธีการที่มีอยู่ยังไม่ไม่มีประสิทธิภาพเพียงพอ และมักจะนำไปสู่โซลูชันที่ไม่เหมาะสม ในงานวิจัยนี้ขอเสนอแบบจำลองใหม่ที่ใช้การเพิ่มประสิทธิภาพการจับกลุ่มอนุภาค (Particle Swarm Optimization) สำหรับการสร้างสถาปัตยกรรมเชิงลึกเพื่อจัดการกับความท้า

ทายข้างต้น โซลูชันที่เรานำเสนอประกอบด้วยองค์ประกอบใหม่สามส่วน ประการแรก มีการวางแผนกลยุทธ์การเข้ารหัสตามกลุ่ม ซึ่งบังคับให้เครือข่ายปฏิบัติตามแนวทางปฏิบัติที่ดีที่สุดเสมอ โดยเฉพาะอย่างยิ่งช่วยให้มั่นใจได้ว่าจำนวนกลุ่มสามารถปรับได้ตามขนาดภาพที่ป้อนเข้ามา ด้วยการจำกัดจำนวนกลุ่ม เราสามารถปรับความถี่ของการดำเนินการรวมกลุ่มเข้ากับขนาดภาพที่ป้อนเข้ามาได้ ด้วยเหตุนี้ จึงยืนยันตำแหน่งและความถี่สูงสุดของการดำเนินการพูลลิ่ง (Pooling Operations) ซึ่งส่งผลให้สถาปัตยกรรมเครือข่ายถูกต้องเสมอ โดยไม่จำเป็นต้องมีกฎการควบคุมที่ซับซ้อนเพิ่มเติม ประการที่สอง มีการสร้างกลไกการอัปเดตความเร็วใหม่ ซึ่งสร้างสถาปัตยกรรมเครือข่ายใหม่โดยการระบุความแตกต่างของการกำหนดค่าเครือข่ายหลัก ประการที่สาม กลไกการอัปเดตตำแหน่งใหม่โดยใช้ความเร็วถ่วงน้ำหนัก (Weighted Velocity) ที่ถูกคิดค้นขึ้น ทั้งกลไกการปรับปรุงความเร็วและตำแหน่งช่วยอำนวยความสะดวกให้กับแบบจำลองที่ใช้การเพิ่มประสิทธิภาพการจับกลุ่มอนุภาคที่เสนอในการค้นหาตำแหน่งตรงกลางของวิถีโคจรของอนุภาค ทำให้สามารถแลกเปลี่ยนระหว่างการกระจายความเสี่ยงและการเพิ่มความเข้มได้ดีขึ้น เราใช้ชุดข้อมูลที่รู้จักกันดี 8 ชุด ได้แก่ Convex, Rectangles, MNIST และตัวแปรต่างๆ สำหรับการประเมินแบบจำลอง แบบจำลองที่ใช้การเพิ่มประสิทธิภาพการจับกลุ่มอนุภาคที่นำเสนอมีความแม่นยำเพิ่มขึ้นถึง 7.58% และลดต้นทุนการคำนวณได้ถึง 63% เมื่อเปรียบเทียบกับวิธีการที่ทันสมัยในปัจจุบัน

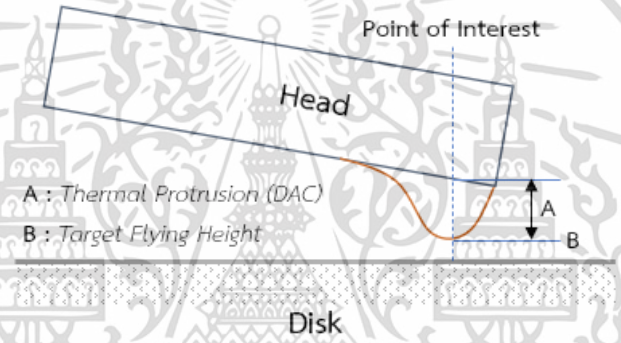


รูปที่ 2.13 สถาปัตยกรรมระบบที่นำเสนอ เมื่อพิจารณาจากชุดข้อมูล อนุภาคจำนวนหนึ่งจะระบุตำแหน่งโมเดลที่ถูกเลือกที่มีประสิทธิภาพสูงสุดโดยอัตโนมัติ โดยได้รับความช่วยเหลือจากกลยุทธ์การเข้ารหัสแบบใหม่ รูปแบบการคำนวณระยะห่างของอนุภาค ตลอดจนขั้นตอนการอัปเดตตำแหน่งแบบถ่วงน้ำหนัก [10]

บทที่ 3
วิธีดำเนินการวิจัย

3.1 จุดสนใจของการวัดการบินของหัวอ่าน/เขียน

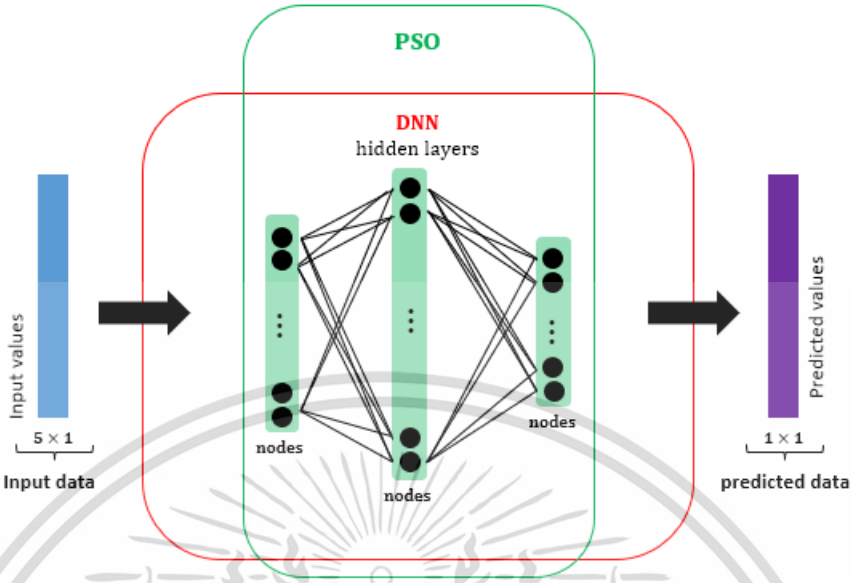
กระบวนการวัดความสูงของการบินของหัวอ่าน/เขียน คือการที่เราต้องการวัดค่าส่วนยื่นซึ่งมีหน่วยเป็น DAC จากส่วนปลายของของหัวอ่าน/เขียนมายังตำแหน่งที่เรียกว่า Target Flying Height ในจุด B ดังนั้น A คือ Thermal Protrusion ที่เราสนใจในงานวิจัยนี้ ซึ่งมีหน่วยเป็น DAC ดังแสดงในรูปที่ 3.1



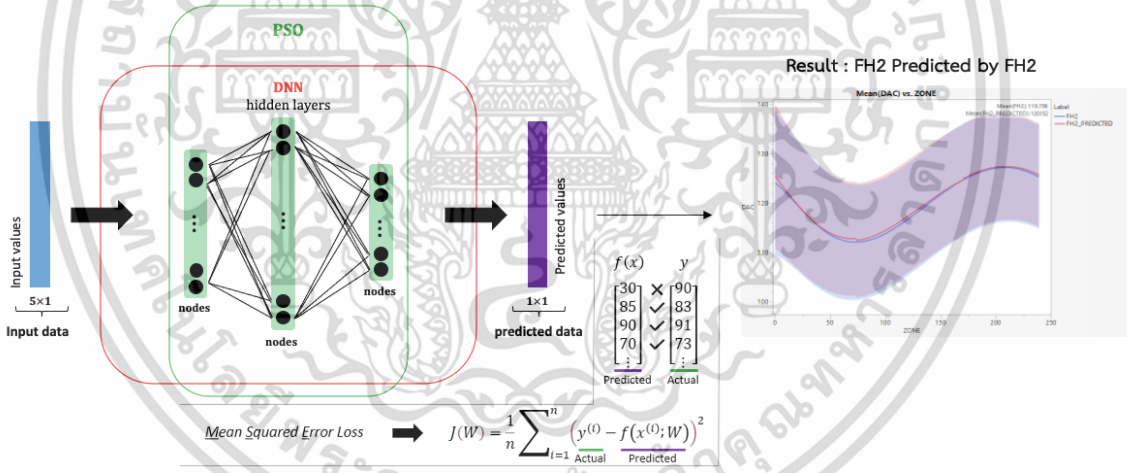
รูปที่ 3.1 จุดสนใจของการวัดการบินของหัวอ่าน/เขียน

3.2 การนำเสนอวิธีการ DNNpso

ซึ่งงานวิจัยนี้นำเสนอวิธีการใหม่ของการทำนายโดยใช้โครงข่ายประสาทเชิงลึก DNN ร่วมกับการเพิ่มประสิทธิภาพกลุ่มอนุภาค PSO เรียกว่า DNNpso เพื่อหาจำนวน nodes ของ hidden layers ที่เหมาะสมและมีประสิทธิภาพดังแสดงในรูปที่ 3.2 และองค์ประกอบโดยรวมของ DNNpso เพื่อการทำนายโปรไฟล์การบินของ FH2 ซึ่งมีหน่วยเป็น DAC ดังแสดงในรูปที่ 3.3



รูปที่ 3.2 โครงข่ายประสาทเชิงลึก DNN ร่วมกับการเพิ่มประสิทธิภาพกลุ่มอนุภาค PSO



รูปที่ 3.3 องค์ประกอบโดยรวมของ DNNpso เพื่อการทำนายโปรไฟล์การบินของ FH2 ซึ่งมีหน่วยเป็น DAC

3.3 โครงสร้าง Pseudocode ของ DNNpso

การออกแบบ Pseudocode ของงานวิจัยนี้โดยใช้โครงข่ายประสาทเชิงลึก DNN ร่วมกับการเพิ่มประสิทธิภาพกลุ่มอนุภาค PSO จะประกอบไปฟังก์ชัน dnn ดังรูปที่ 3.4 ฟังก์ชัน pso ดังรูปที่ 3.5 และการเรียกใช้ฟังก์ชัน pso และ dnn ดังรูปที่ 3.6

```
def dnn( particle ) ## particles are node of each hidden layers, ex. particles = [ly1, ly2, ly3]
    ly1, ly2, ly3 = particle

    Normalize input/output for training set (X_train, y_train) and test set (X_test, y_test)

    Dense( number_of_input )
    Activation( 'relu' )
    Loop hidden layers:
        Dense( lyX ) ## ly1, ly2, ly3
        Activation( 'relu' )
    Dense( number_of_output )
    Activation( 'linear' )

    compile model ## set loss, optimizers, metrics
    fit model ## set epochs and batch_size
    y_predicted = predict X_test set
    inverse normalization for y_predicted
    find r2, correlation between y_predicted and y_test set

    if r2 > r2_max:
        save model for later use (.json & .h5)

    return r2 ## objective function
```

รูปที่ 3.4 ฟังก์ชัน dnn

```
def pso( dnn, bounds, population_size, max_iter, w, c1, c2 )
    Initialize the population with random particles and velocity
    Initialize the best global position  ## gbest is best particle
    Loop max_iter: ## iterations
        Loop population_size: ## ex. population_size 10 = particle [3x] x velocity [3x] x 10 = 30
            Loop update velocity:
                velocity[i] = w * velocity[i] + c1 * r1 * (particle[i] - gbest[i]) + c2 * r2 * (particle[i] - gbest[i])
            Loop update particle:  ## update particle with new velocity
                particle[i] = particle[i] + velocity[i]

        r2_obj = dnn( particle )

        if r2_obj > r2_obj_max  ## gbest will be updated with current particles
            gbest = particle
```

รูปที่ 3.5 ฟังก์ชัน pso

```
## Main Function Call - start ##
bounds = [(0, 50), (0, 50), (0, 50)]  ## define number of node in each hidden layer, example 3 layers
best_particle, best_r2 = pso( dnn, bounds )
## Main Function Call - end ##
```

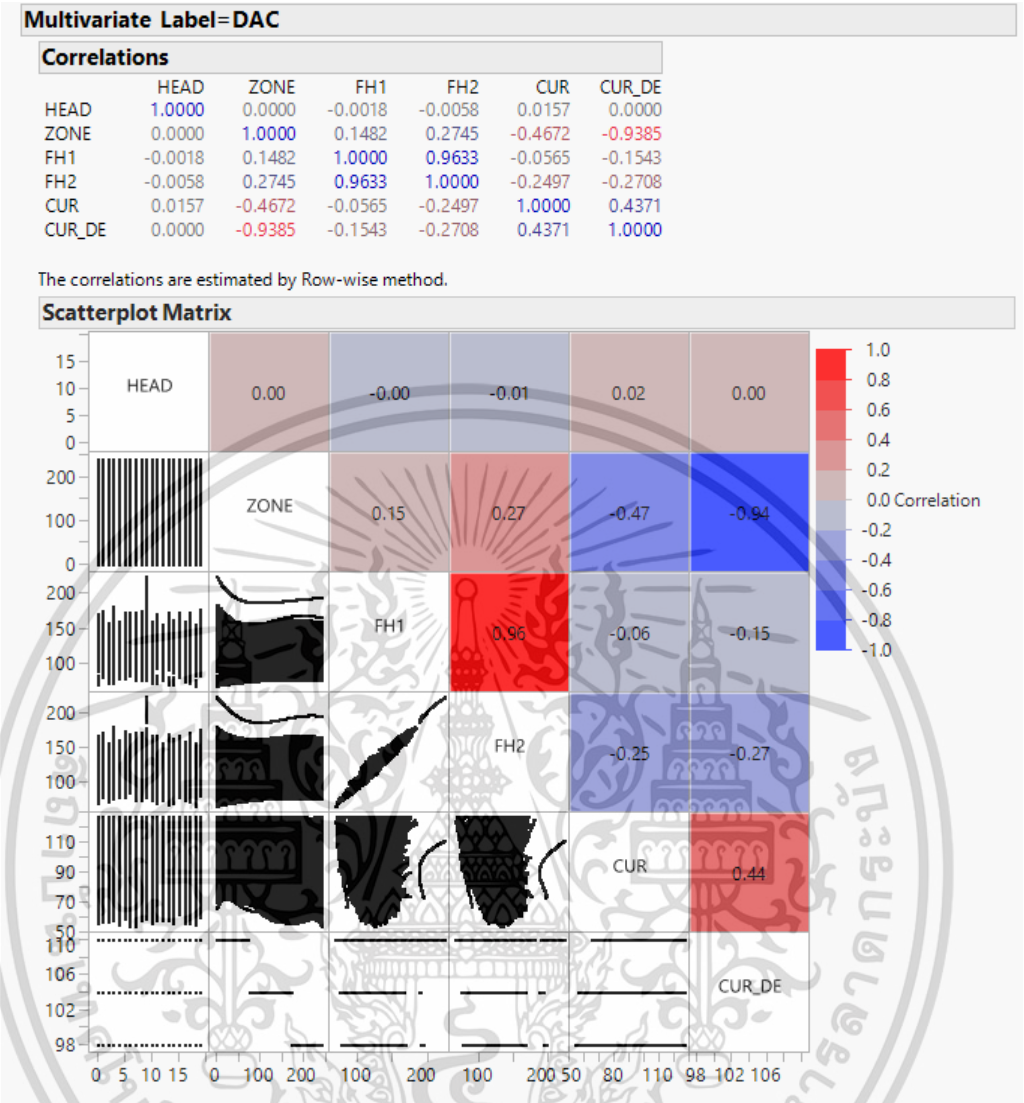
รูปที่ 3.6 การเรียกใช้ฟังก์ชัน pso และ dnn

3.4 การจัดเตรียมข้อมูล

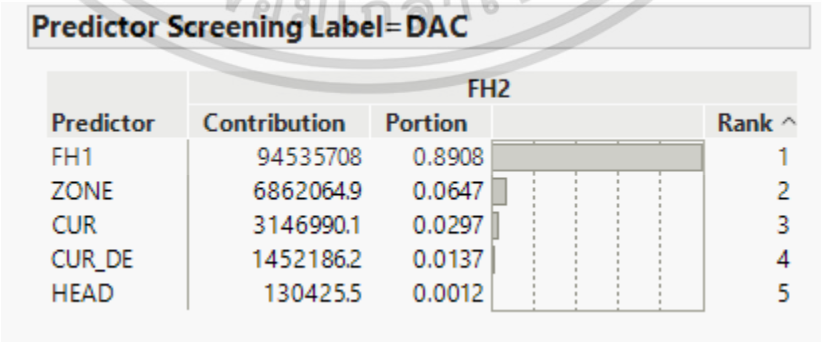
จัดเตรียมข้อมูลสำหรับการ training โดยใช้ข้อมูลจากฮาร์ดดิสก์ไดรฟ์ 50 ตัว ซึ่งแต่ละตัวจะประกอบไปด้วยหัวอ่าน/เขียน 20 หัว และมีโซน 240 โซน ดังนั้นจึงมีข้อมูล 240,000 จุด (50 x 20 x 240)

3.5 การวิเคราะห์ข้อมูล

สำหรับการ training แบบจำลอง DNN ซึ่งชุดข้อมูลถูกแบ่งออกเป็น 80% สำหรับ training และ 20% สำหรับการตรวจสอบ โดยข้อมูลป้อนเข้า input แบบจำลองจะมี DAC ของ FH1, Head, Zone, Current และ Default Current และมีผลลัพธ์ output คือ DAC ของ FH2 ดังแสดงความสัมพันธ์ในรูปที่ 3.7 และการจัดลำดับความสัมพันธ์ของข้อมูลก่อนเข้า input ด้วย Bootstrap Forest ซึ่งมีรากพื้นฐานมาจาก Decision Tree ดังแสดงในรูปที่ 3.8 ซึ่งในงานวิจัยนี้จะใช้ hidden layers 3 layers ซึ่งกลุ่มอนุภาค PSO จะทำการหาจำนวน nodes ที่เหมาะสม โดยกลุ่มอนุภาค PSO ดังสมการ (3.1) และ (3.2) มีการกำหนดฟังก์ชั่นวัตถุประสงค์โดยดูตัวชีวัดจากค่า R^2 ดังสมการที่ (3.3) จากการวัดการบินของฮาร์ดดิสก์ไดรฟ์จริงและค่าที่ทำนายจากแบบจำลอง ซึ่งการ training แบบจำลอง DNN จะใช้ตัวชี้วัด Mean Squared Error ดังสมการที่ (3.4) เป็น Loss Function โดยที่ DNN จะใช้ Backpropagation Algorithm ในการหาอนุพันธ์ของค่า error เทียบกับค่า weights รอบก่อนหน้า แล้วทำการปรับปรุงค่า weights ใหม่ ที่ทำให้ค่า error น้อยที่สุด ดังสมการที่ (3.5)



รูปที่ 3.7 ความสัมพันธ์ของ input และ output



รูปที่ 3.8 การจัดลำดับความสัมพันธ์ของ input ด้วย Bootstrap Forest (decision tree-based)

Particle Swarm Optimization (PSO):

$$V_{t+1}^{i} = wV_t^i + c_1 r_1 (pb^i - X_t^i) + c_2 r_2 (gb - X_t^i) \quad (3.1)$$

$$X_{t+1}^{i} = X_t^i + V_{t+1}^i \quad (3.2)$$

Coefficient of determination or R^2:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \bar{y_i})^2}{\sum_{i=1}^{n}(y_i - \mu)^2} \quad (3.3)$$

Mean Square Error:

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \bar{y_i})^2}{n} \quad (3.4)$$

Backpropagation Algorithm (for DNN):

$$* w_x = w_x - \alpha \left(\frac{\partial error}{\partial w_x} \right) \quad (3.5)$$

จากสมการเหล่านี้ โดยที่ y แทนค่าจริง $\bar{y}$ คือค่าที่ทำนายไว้ μ คือค่าเฉลี่ยของค่าจริง และ n แทนกลุ่มของค่าในชุดข้อมูล โดยทั่วไป คะแนนที่ต่ำของ MSE บ่งบอกถึงการทำนายที่แม่นยำ และสิ่งนี้เกิดขึ้นเมื่อค่าที่ทำนายไว้ $\bar{y}$ ใกล้เคียงกับค่าจริงมาก ค่า R^2 เป็นตัววัดความดีของความเหมาะสมสำหรับ regression และมักจะเป็นคะแนนระหว่าง 0 ถึง 1 คะแนน 1 หมายถึงการคาดทำนายที่สมบูรณ์แบบ และโดยทั่วไป ค่าที่สูงกว่าแสดงถึงประสิทธิภาพที่ดีกว่า

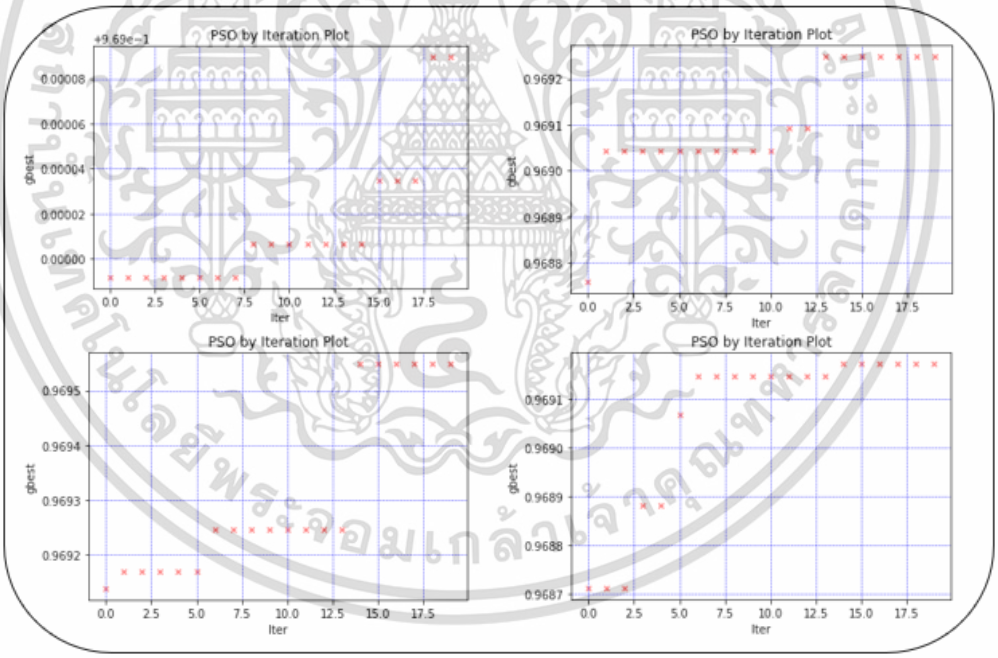
Backpropagation Algorithm ซึ่งจะทำงานในส่วน training ของ DNN จะมี w_x และ $error$ เป็นค่า weights และค่า error ของรอบก่อนหน้า และ α คือ learning rate ซึ่งจะมีการปรับปรุงค่า weights ใหม่ไปยัง $* w_x$ สำหรับการ training ในแต่ละรอบ

Particle Swarm Optimization จะทำการหาจำนวน nodes ของ hidden layers ของ DNN ได้อย่างเหมาะสมและมีประสิทธิภาพ ซึ่ง w คือสัมประสิทธิ์ของความเฉื่อย (momentum coefficient) V คือความเร็ว (velocity) ของอนุภาค X คือตำแหน่ง (position) ของอนุภาค c_1 และ c_2 คือค่าคงที่ใดๆ ที่ถูกสุ่มขึ้นมาและอยู่ในช่วง [0, 1] $pb\ (pbest)$ คืออนุภาคตัวที่มีฟังก์ชันวัตถุประสงค์สูงที่สุดที่อนุภาคนั้นเคยเจอ (local best) และ $gb\ (gbest)$ คือค่าฟังก์ชันวัตถุประสงค์ที่สูงที่สุดของรอบการดำเนินการที่ผ่านมาทั้งหมด (global best) i คือจำนวนรอบที่จะดำเนินการ t คืออนุภาคแต่ละตัว

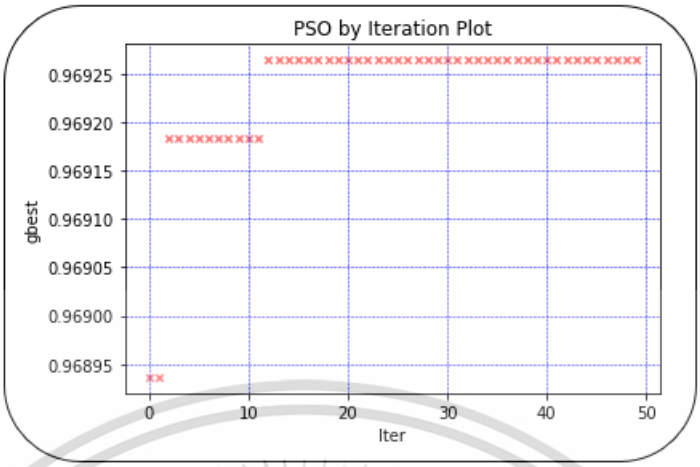
บทที่ 4
ผลการทดลอง

4.1 การทดลองหาจำนวน iterations ของ DNNpso

จากการทดลองของ DNNpso พบว่ากลุ่มอนุภาคมีความสามารถที่จะช่วยหาจำนวน nodes ของแต่ละ hidden layers ที่เหมาะสม โดยงานวิจัยนี้ได้กำหนด Particle Swarm 10 และจำนวน Iteration 20 ซึ่งจำนวนอนุภาคทั้ง 10 จะพยายามหาทิศทางเพื่อปรับค่าจำนวน nodes ให้ลู่เข้าฟังก์ชันวัตถุประสงค์ ซึ่งมี $\boldsymbol{R^2}$ เป็นตัวชี้วัด ซึ่งจะทำซ้ำทั้งหมด 20 รอบ โดยผลการทดลองแสดงให้เห็นว่าสามารถได้ $\boldsymbol{R^2 \approx 0.96}$ จากการทำ 4 ครั้งดังแสดงในรูปที่ 4.1 และตารางที่ 4.1 และมีการทดลองหาจำนวน iterations ที่เหมาะสมสำหรับงานวิจัยนี้ เราจะเห็นว่าเมื่อเริ่มเกิน 10 iterations จะไม่ค่อยเห็นการเพิ่มของ $\boldsymbol{R^2}$ ดังแสดงในรูปที่ 4.2 ดังนั้นงานวิจัยนี้จึงกำหนดไว้ที่ 20 iterations



รูปที่ 4.1 ผลการทดลอง DNNpso โดยมี R^2 เป็น Objective Function



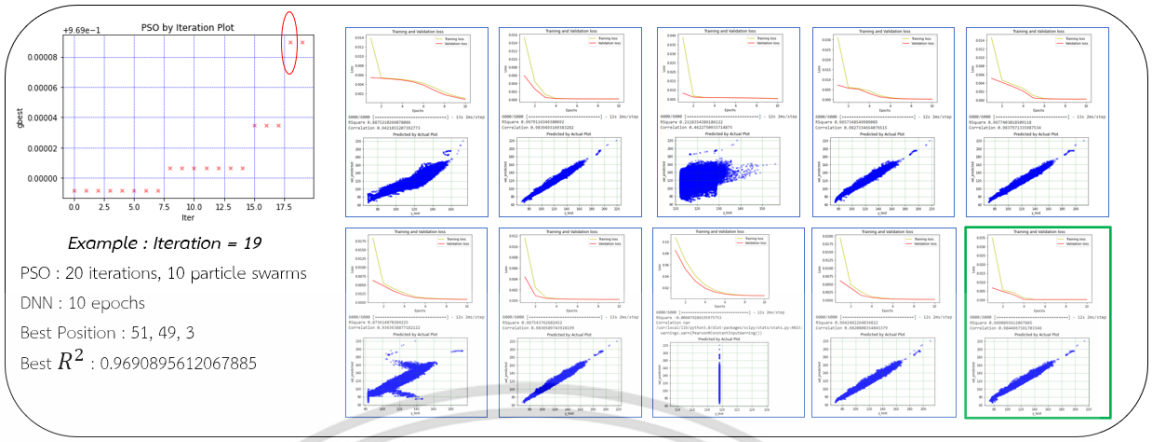
รูปที่ 4.2 ผลการทดลองการปรับ iteration ของ DNNpso

ตารางที่ 4.1 ผลการทดลองของการฝึกแบบจำลองซ้ำด้วยการกำหนดค่าที่เลือก เพื่อค้นหาฟังก์ชันวัตถุประสงค์ที่ดีที่สุด (R^2)

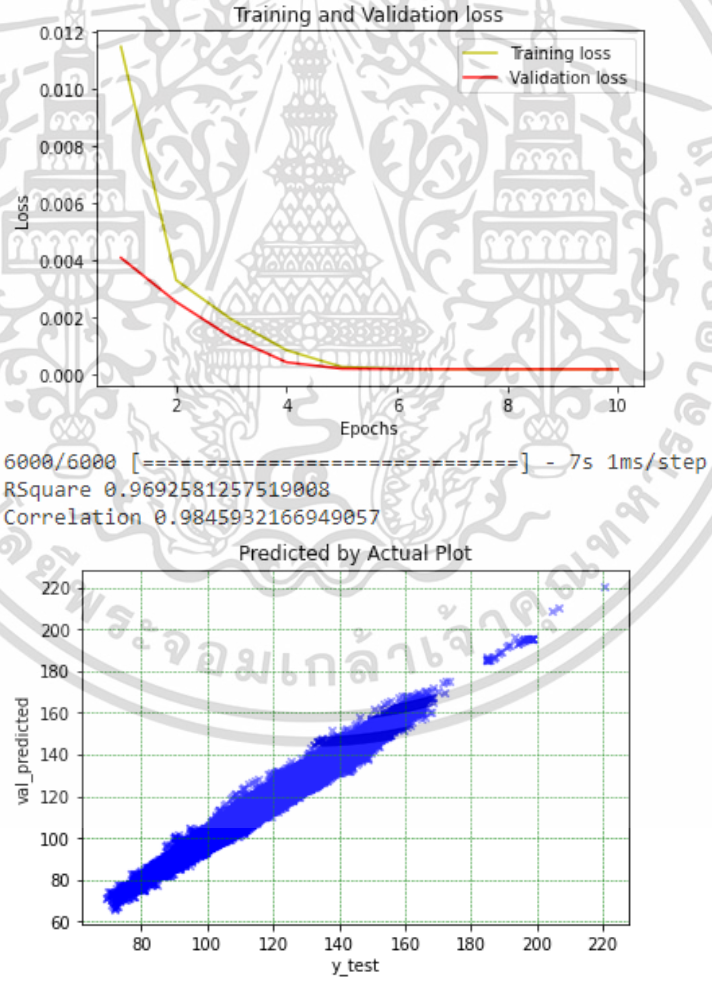
Run	Iteration	Particle Swarm	Epoch (DNN)	MSE	R^2	Training Time (sec)
1	20	10	10	0.000203	0.9690	21.0744
2	20	10	10	**0.000200**	**0.9692**	**10.0495**
3	20	10	10	0.000199	0.9695	26.4638
4	20	10	10	0.000202	0.9691	45.4285

4.2 การทดลองหาฟังก์ชันวัตถุประสงค์ (R^2) จาก Particle Swarm ของ DNNpso

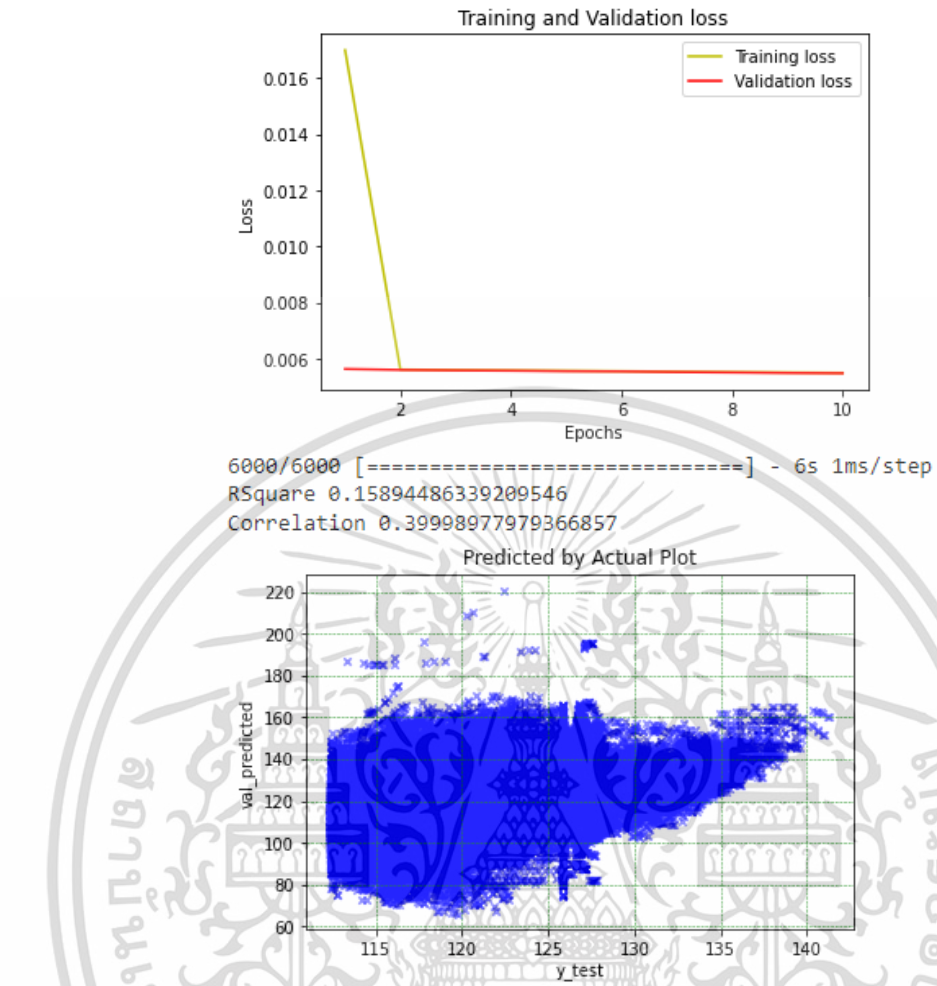
ซึ่งจากการทดลองเมื่อดูในรายละเอียดของแต่ละ Particle Swarm ที่ให้ผลลัพธ์ที่แตกต่างกัน ก็จะพบว่าถ้า R^2 ที่ดีและมีประสิทธิภาพที่ได้จาก DNNpso ก็จะส่งผลต่อความสัมพันธ์ที่ดีระหว่างค่าการวัดการบินจริงกับค่าทำนายของชุดข้อมูล validation ส่วน 20% ดังแสดงในรูปที่ 4.3 ซึ่งกรอบสีเขียวมีค่า R^2 สูงที่สุดและมีความสัมพันธ์ดีที่สุด ซึ่งเป็นจุดที่ PSO บอกว่ามีประสิทธิภาพมากที่สุดเช่นกัน รูปที่ 4.4 แสดงถึงฟังก์ชันวัตถุประสงค์ R^2 ที่ดีที่สุดในขณะที่รูปที่ 4.5 แสดงถึงฟังก์ชันวัตถุประสงค์ R^2 ที่แย่ที่สุด และการกระจายตัวของ R^2 (Distribution) ของแต่ละกลุ่มอนุภาคดังรูปที่ 4.6



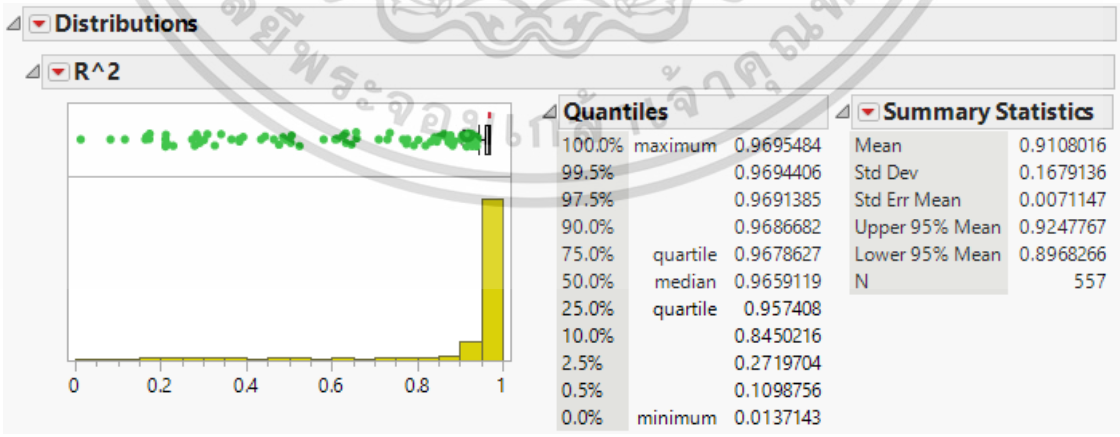
รูปที่ 4.3 เปรียบเทียบความสัมพันธ์กับ R^2 ของ DNNpso ในแต่ละ Particle Swarm



รูปที่ 4.4 ฟังก์ชันวัตถุประสงค์ R^2 ที่ดีที่สุด



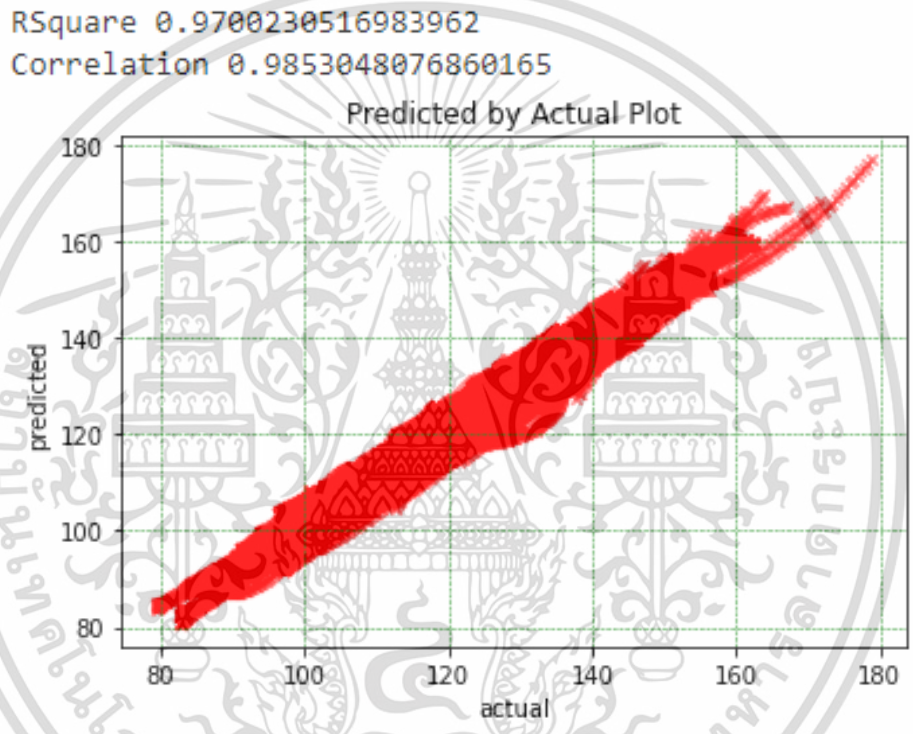
รูปที่ 4.5 ฟังก์ชันวัตถุประสงค์ R^2 ที่แย่ที่สุด



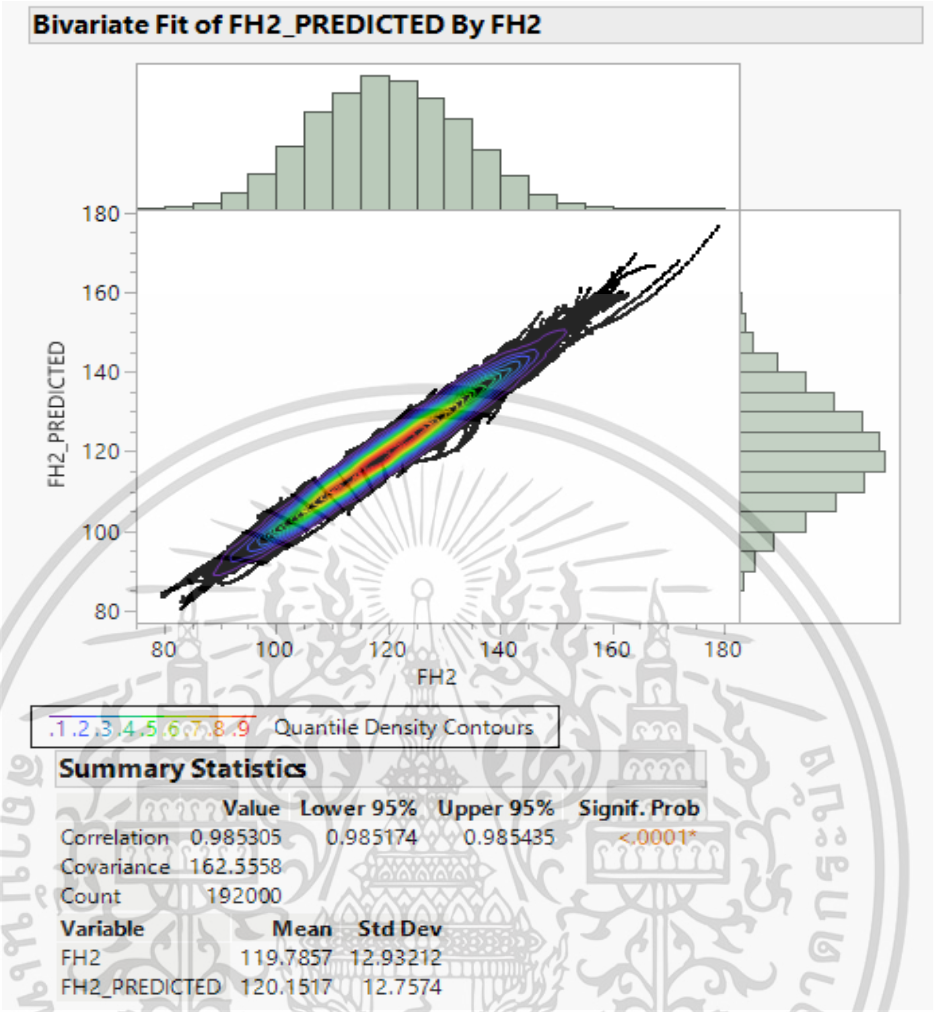
รูปที่ 4.6 การกระจายตัว (distribution) R^2 ของ DNNpso ในแต่ละ Particle Swarm

4.3 การทดลองการประเมินผลแบบจำลองที่ได้จาก DNNpso กับฮาร์ดดิสก์ไดรฟ์กลุ่มใหม่

หลังจากได้แบบจำลองที่เป็นที่น่าพอใจ ก็ทำการบันทึกแบบจำลองเพื่อเก็บไว้ใช้ จึงได้มีการนำข้อมูลของฮาร์ดดิสก์ไดรฟ์จำนวน 40 ตัว ซึ่งไม่ได้อยู่ในกลุ่ม training 80% และ validation 20% นำมาทดลองด้วยแบบจำลอง ให้ผลการทดลองเป็นที่น่าพอใจ และมีค่า $R^2 \approx 0.97$ ดังรูปที่ 4.7 และ $Correlation \approx 0.98$ ดังรูปที่ 4.8 นั่นหมายความว่าแบบจำลองมีความแม่นยำและมีความสัมพันธ์กันเป็นอย่างมากระหว่างการวัดการบินจริงและค่าจากการทำนาย



รูปที่ 4.7 ค่า R^2 จากการวัดจริงและการทำนายจากฮาร์ดดิสก์ไดรฟ์กลุ่มใหม่

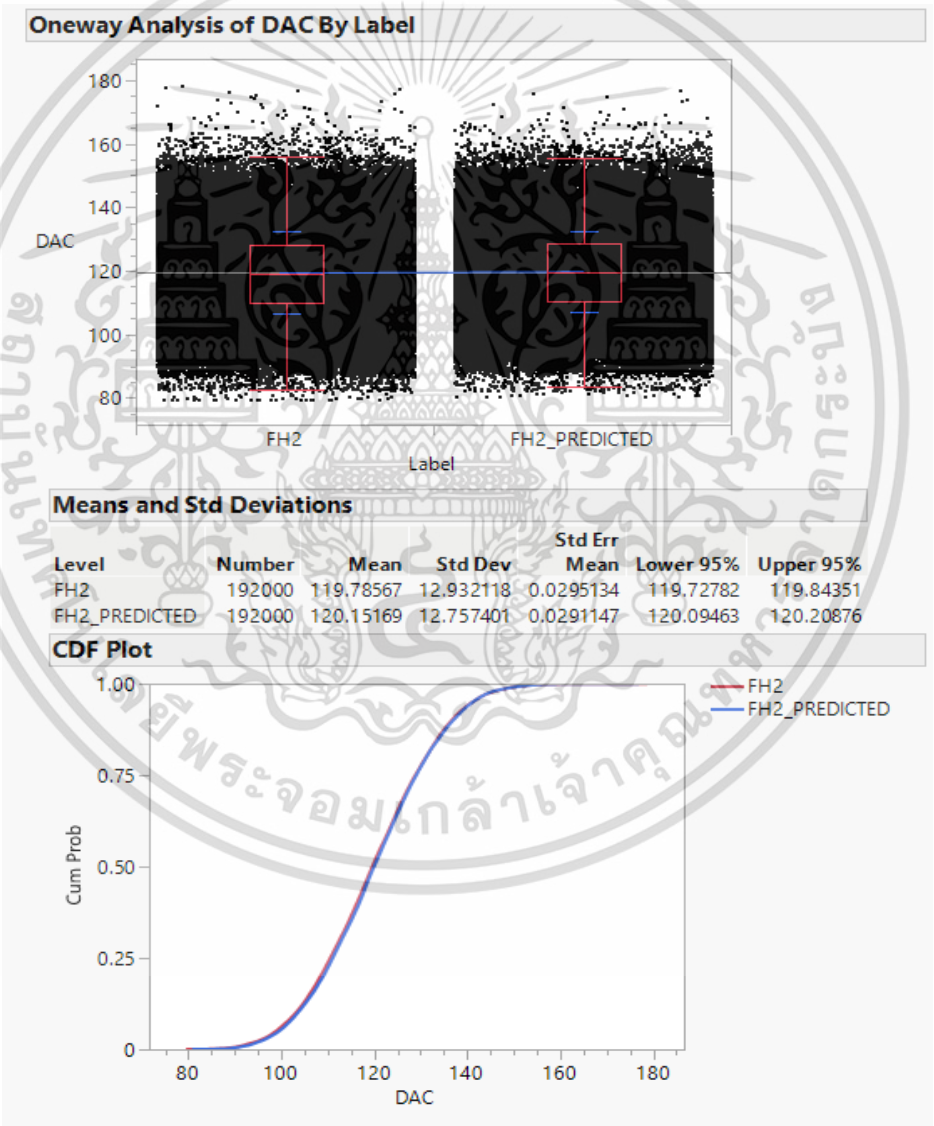


รูปที่ 4.8 ค่า Correlation จากการวัดจริงและการทำนายจากฮาร์ดดิสก์ไดรฟ์กลุ่มใหม่

Boxplot แสดงให้เห็นถึงความเท่ากัน โดยค่า $mean \approx 120\ DAC$ และ $Std\ Dev \approx 12.8$ และ CDF Probability ดังสมการ (4.1) ของการวัดการบินจริงและการทำนายให้ผลที่เท่ากัน ดังแสดงในรูปที่ 4.9

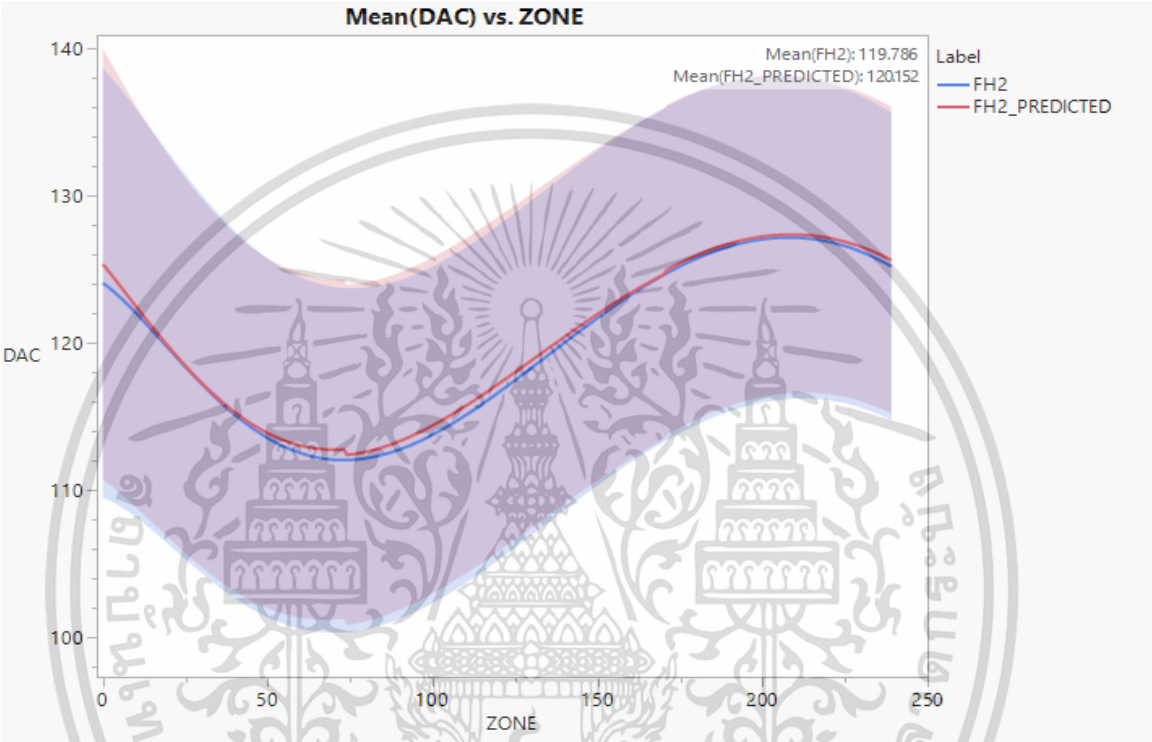
$$F_x(x) = P(X \leq x) = \sum_{x_i<x} P(X = x_i), \tag{4.1}$$

$x \in [0, 255]$, x : DAC

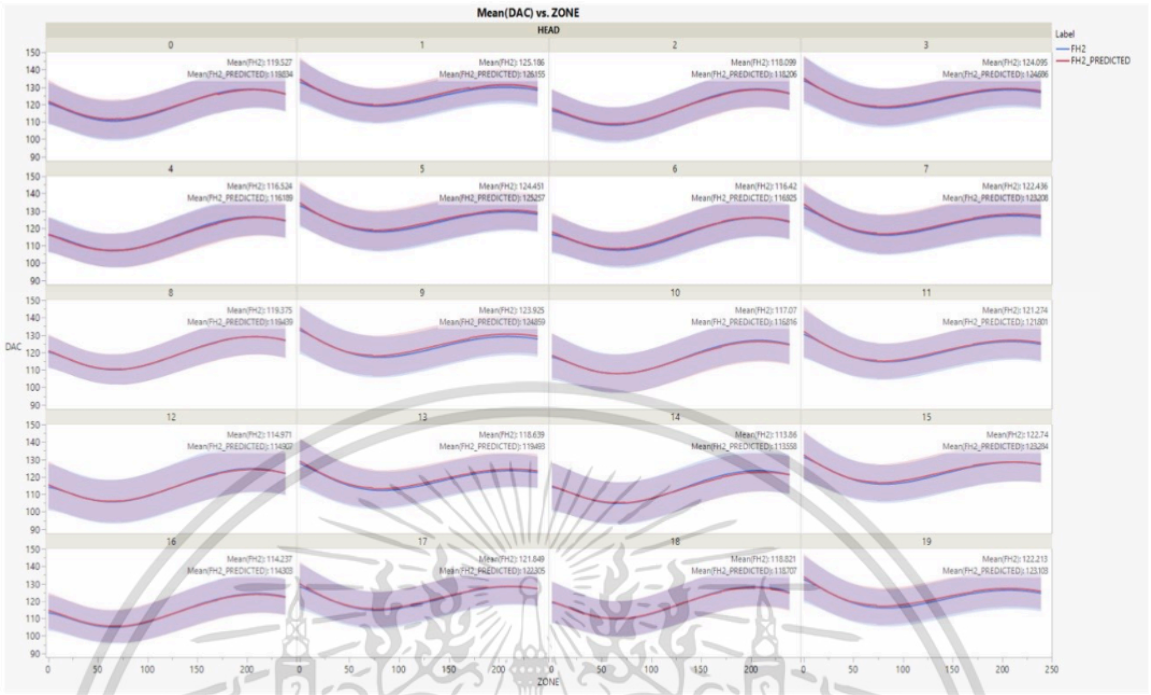


รูปที่ 4.9 Boxplot และ CDF จากการวัดจริงและการทำนายจากฮาร์ดดิสก์ไดรฟ์กลุ่มใหม่

และภาพโปรไฟล์ DAC ซึ่งเป็นภาพรวมที่เกิดจากการวัดจริงและการทำนายให้ผลเป็นที่น่าพอใจ ซึ่งมีความใกล้เคียงเป็นอย่างมาก ดังรูปที่ 4.10 และได้แยกโปรไฟล์ DAC ออกเป็นแต่ละหัวอ่าน/เขียน ซึ่งผลการทดลองก็ยังมีประสิทธิภาพที่ดีและเป็นที่น่าพอใจดังแสดงในรูปที่ 4.11 ซึ่งสามารถดูค่า mean, Std Dev ได้จากรูปที่ 4.9



รูปที่ 4.10 โปรไฟล์ DAC จากการวัดจริงและการทำนายของ FH2 จากฮาร์ดดิสก์ไดรฟ์กลุ่มใหม่

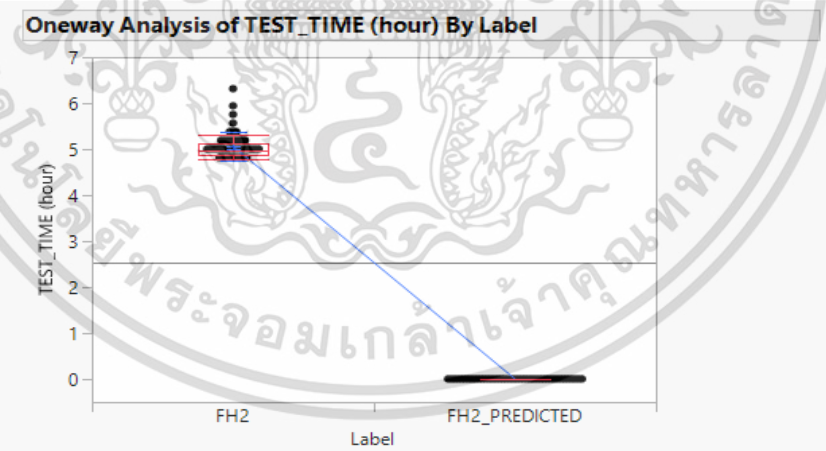


รูปที่ 4.11 โปรไฟล์ DAC ของแต่ละหัวอ่าน/เขียน จากการวัดจริงและการทำนายของ FH2 จากฮาร์ดดิสก์ไดรฟ์กลุ่มใหม่

บทที่ 5
บทสรุป

งานวิจัยนี้ได้นำเสนอวิธีการวัดการบินโดยใช้การทำนายของ DNN ร่วมกับวิธีการ PSO ตามที่เรียกว่า DNNpso ซึ่งวิธีการนี้ช่วยให้หาจำนวน nodes ของ hidden layers ที่เหมาะสมและเกิดประสิทธิภาพและเป็นการนำเสนอวิธีการใหม่ในการวัดการบินซึ่งสามารถประยุกต์ใช้กับปัญญาประดิษฐ์และกระบวนการผลิตฮาร์ดดิสก์ไดรฟ์ได้อย่างลงตัว และการทำนายทำให้สามารถลดเวลาการการวัดการบินได้เป็นอย่างมาก ทำให้ช่วยลดกระบวนการผลิตของฮาร์ดดิสก์ไดรฟ์ได้อย่างมีนัยสำคัญซึ่งตรงกับวัตถุประสงค์ที่ตั้งไว้ของงานวิจัยนี้ดังแสดงในรูปที่ 5.1 ซึ่งการใช้ข้อมูลจากต้นทางอย่าง FH1 ก็แสดงให้เห็นถึงความสามารถที่จะทำนาย FH2 ได้อย่างมีประสิทธิภาพ

วิธีการ DNNpso สามารถที่จะนำแนวคิดนี้ไปประยุกต์ใช้กับงานทางด้านปัญญาประดิษฐ์อื่นๆ ได้อย่างหลากหลายเพื่อการค้นหาการเรียนรู้แบบจำลองที่ดีที่สุดและอัตโนมัติ แต่อย่างไรก็ตาม DNNpso ของขอบเขตงานวิจัยนี้มีการกำหนด hidden layer ของ DNN ไว้เพียง 3 layers เท่านั้น ถ้ามีการกำหนดขอบเขตให้กว้างขึ้นเพื่อให้ DNNpso สามารถค้นหาจำนวน hidden layer ได้อย่างเหมาะสม จะเป็นการดึงความสามารถของ DNN ออกมาได้อย่างมีประสิทธิภาพมากขึ้น



Means and Std Deviations

Level	Number	Mean	Std Dev	Std Err Mean	Lower 95%	Upper 95%
FH2	40	5.063642	0.3108686	0.0491526	4.9642214	5.1630626
FH2_PREDICTED	40	0.0017615	0.0004312	6.8185e-5	0.0016236	0.0018994

รูปที่ 5.1 เปรียบเทียบเวลาจากการวัดด้วยวิธีดั้งเดิมและเวลาการวัดด้วยการทำนายของ FH2 จากฮาร์ดดิสก์ไดรฟ์กลุ่มใหม่

เอกสารอ้างอิง

[1] H. Dakroub, M. C. Rao and A. Gay Sam, “Fly Height Measurement for A Disc Drive,” US Patent: US6898034, Assignee: Seagate Technology LLC, Publication date on May 24, 2005.

[2] B.C. Schardt, E. Schreck, R. Sonnenfeld, Q. Haddock and J.R. Haggis “Flying Height Measurement while Seeking in Hard Disk Drives,” IEEE Transactions on Magnetics, vol.34, Issue 4, p.1765-1767, July 1998.

[3] V. J. Novotny, “Magnetic Recording Drive Dynamics during Seeking and Parking”, IEEE Transactions on Magnetics, vol.33, p.3115-3317, September 1997.

[4] J. Y. Juang, T. Nakamura, B. Knigge, Y. Luo, W. C. Hsiao, K. Kuroki, F. Y. Huang and P. Baumgart, “Numerical and Experiment Analyses of Nanometer-Scale Flying Height Control of Magnetic Head with Heating Element,” IEEE Transactions on Magnetics, vol.44, issue 11, p.3679-3682, November 2008.

[5] U. Boettcher, H. Li, R. A. de Callafon and F. E. Talke, “Dynamic Flying Height Adjustment in Hard Disk Drives Through Feedforward Control,” IEEE Transactions on Magnetics, vol.47, issue 7, p.1823-1829, June 2011.

[6] W. SHI, D. LIU, X. CHENG, Y. LI, AND Y. ZHAO, “Particle Swarm Optimization-Based Deep Neural Network for Digital Modulation Recognition,” IEEE Access, vol.7, p.104591-104600, July 2019.

[7] F. E. Fernandes Junior and G. G. Yen, “Particle swarm optimization of deep neural networks architectures for image classification,” Swarm and Evolutionary Computation, vol.49, p.62-74, September 2019.

[8] B. A. Garro and R. A. Vazquez, “Designing Artificial Neural Networks Using Particle Swarm Optimization Algorithms,” 2009 International Joint Conference on Neural Networks, July 2009.

[9] P. Konghuayrob and S. Kaitwanidvilai, “LOW ORDER ROBUST v-GAP METRIC H∞ LOOP SHAPING CONTROLLER SYNTHESIS BASED ON PARTICLE SWARM OPTIMIZATION,” International Journal of Innovative Computing, Information and Control, vol.13, No.6, p.1777-1789, December 2017.

[10] T. Lawrence, L. Zhang, C. P. Lim and E. J. Philips, “Particle Swarm Optimization for Automatically Evolving Convolutional Neural Networks for Image Classification,” IEEE Access, vol.9, p.14369-14386, January 2021.

[11] J. Chen, W. Ding, X. M. Li, X. Xi, K. P. Ye, H. B. Wu and R. X. Wu, “Absorption and Diffusion Enabled Ultrathin Broadband Metamaterial Absorber Designed by Deep Neural Network and PSO,” IEEE Antennas and Wireless Propagation Letters, vol.20, issue 10, p.1993-1997, August 2021.

[12] Y. Gao, H. Liu, F. Niu and Y. Tian, “A Capability Fitting and Data Reconstruction Model Based on Particle Swarm Optimization-Bidirectional Deep Neural Network for Search and Rescue System of Systems,” IEEE Access, vol.11, p.10366-10383, January 2023.

[13] B. Lin, Y. Huang, J. Zhang, J. Hu, X. Chen and J. Li, “Cost-Driven Off-Loading for DNN-Based Applications Over Cloud, Edge, and End Devices,” IEEE Transactions on Industrial Informatics, vol.16, issue 8, p.5456-5466, December 2019.

[14] X. Chen, J. Zhang, B. Lin, Z. Chen, K. Wolter and G. Min, “Energy-Efficient Offloading for DNN-Based Smart IoT Systems in Cloud-Edge Environments,” IEEE Transactions on Parallel and Distributed Systems, vol.33, issue 3, p.683-697, July 2021.

[15] A. Rajagopal, G. P. Joshi, A. Ramachandran, R. T. Subhalakshmi, M. Khari, S. Jha, K. Shankar and J. You, “A Deep Learning Model Based on Multi-Objective Particle Swarm Optimization for Scene Classification in Unmanned Aerial Vehicles,” IEEE Access, vol.8, p.135383-135393, July 2020.

[16] N. Rungtalay, “An Alternative Solution for Hard Disk Drive Classification Process Improvement,” College of data storage innovation King Mongkut’s Institute of Technology Ladkrabang, July 2014.

[17] T. Thanasarn, “Study of Neural Network for Fly Height Failure Pattern Classification in Hard Disk Drive,” College of data storage innovation King Mongkut’s Institute of Technology Ladkrabang, July 2014.

[18] B. D. Strom, S. C. Lee, G. W. Tyndall and A. Khurshudov, “Hard Disk Drive Reliability Modeling and Failure Prediction,” IEEE transactions on Magnetics, vol.43, issue 9, p.3676-3684, September 2007.

[19] J. Kennedy, R.C. Eberhart, Swarm Intelligence, Academic Press, San Diego, CA, 2001.

[20] Introduction to Deep Neural Networks. [Online]. เข้าถึงได้จาก : https://www.datacamp.com/tutorial/introduction-to-deep-neural-networks

[21] B. M. Mostafa, N. El-Attar, S. Abd-Elhafeez and W. Awad, “Machine and Deep Learning Approaches in Genome,” Alfarama Journal of Basic & Applied Sciences, vol.2, issue 1, p.105-113, January 2021.

[22] Particle swarm optimization. [Online]. เข้าถึงได้จาก : https://en.wikipedia.org/wiki/Particle_swarm_optimization

ภาคผนวก ก
บทความที่ได้รับการตีพิมพ์และเผยแพร่

Sensors and Materials, Volume 36, Number 4(2) (2024)

pp. 1377-1387
S&M3605 Research Paper of Special Issue
https://doi.org/10.18494/SAM4825
Published: April 19 , 2024

Prediction of Flying Height Using Deep Neural Network Based on Particle Swarm Optimization in Hard Disk Drive Manufacturing Process [PDF]

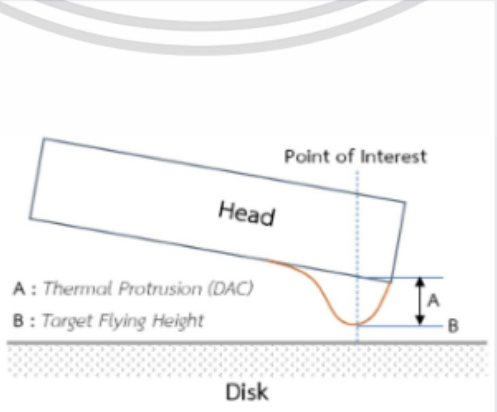
Worawit Kanjanapruthipong, Pitcha Prasitmeeboon, and Poom Konghuayrob

(Received December 26, 2023; Accepted March 13, 2024)

Keywords: flying height, deep neural network, particle swarm optimization, hard disk drive, DNNpso, AI

In contemporary hard disk drive (HDD) manufacturing processes, after the assembly of the HDD from the production line, a series of diverse calibration procedures are necessary to ensure standardization. These include capacity calibration, which determines the storage space in terabytes (TB) presently available, and flying height (FH) calibration, which evaluates the distance between the head and the disk by applying electric current to the heater coil element to achieve the desired FH, thus optimizing the writing and reading performance and tailoring it to each HDD. Additionally, electric current is saved in a digital-to-analog converter (DAC) unit for the utilization of a read/write head, while a preamp collaborates with the drive firmware to convert the electric current in the DAC unit to milliwatts. In the present scenario, multiple calibrations of flying heights (FHs), specifically flying height 1 (FH1) and flying height 2 (FH2), are performed. Each FH calibration requires a testing time of approximately 5 h owing to the separation of measurement points into 240 locations across the disk surface, referred to as test zones, with a total of 20 heads. The primary objective of this study is to reduce the testing time by using a combination of deep neural network (DNN) and particle swarm optimization techniques to predict the DAC profiles of FH2 as it approaches FH1, where FH1 is the input for the DNN model.

Corresponding author: Poom Konghuayrob



Sensors and Materials, Vol. 36, No. 4 (2024) 1377–1387
MYU Tokyo

S & M 3605

Prediction of Flying Height Using Deep Neural Network Based on Particle Swarm Optimization in Hard Disk Drive Manufacturing Process

Worawit Kanjanapruthipong,[1] Pitcha Prasitmeeboon,[2] and Poom Konghuayrob[3*]

[1]Department of Robotics and AI Engineering, School of Engineering, King Mongkut's Institute of Technology Ladkrabang, 1, Soi Chalong Krung 1, Ladkrabang, Bangkok 10520, Thailand
[2]Department of Control Engineering, School of Engineering, King Mongkut's Institute of Technology Ladkrabang, 1, Soi Chalong Krung 1, Ladkrabang, Bangkok 10520, Thailand
[3]Department of Electrical Engineering, School of Engineering, King Mongkut's Institute of Technology Ladkrabang, 1, Soi Chalong Krung 1, Ladkrabang, Bangkok 10520, Thailand

(Received December 26, 2023; accepted March 13, 2024)

Keywords: flying height, deep neural network, particle swarm optimization, hard disk drive, DNNpso, AI

In contemporary hard disk drive (HDD) manufacturing processes, after the assembly of the HDD from the production line, a series of diverse calibration procedures are necessary to ensure standardization. These include capacity calibration, which determines the storage space in terabytes (TB) presently available, and flying height (FH) calibration, which evaluates the distance between the head and the disk by applying electric current to the heater coil element to achieve the desired FH, thus optimizing the writing and reading performance and tailoring it to each HDD. Additionally, electric current is saved in a digital-to-analog converter (DAC) unit for the utilization of a read/write head, while a preamp collaborates with the drive firmware to convert the electric current in the DAC unit to milliwatts. In the present scenario, multiple calibrations of flying heights (FHs), specifically flying height 1 (FH1) and flying height 2 (FH2), are performed. Each FH calibration requires a testing time of approximately 5 h owing to the separation of measurement points into 240 locations across the disk surface, referred to as test zones, with a total of 20 heads. The primary objective of this study is to reduce the testing time by using a combination of deep neural network (DNN) and particle swarm optimization techniques to predict the DAC profiles of FH2 as it approaches FH1, where FH1 is the input for the DNN model.

1. Introduction

In hard disk drive (HDD) manufacturing, the calculation of the flying height (FH) for read/write heads is crucial to ensure efficient and effective read/write operations that are optimized for each HDD. Dakroub *et al.*[1] studied the measurement of FH for an HDD by examining the amplitude of the signals used in this process. Schardt *et al.*[2] developed a technique for measuring FH, which involves the movement of read/write heads from the inner track to the

*Corresponding author: e-mail: poom.ko@kmitl.ac.th
https://doi.org/10.18494/SAM4825

ISSN 0914-4935 © MYU K.K.
https://myukk.org/

outer track of the disk. Novotny[3] developed a novel technique for characterizing the magnetic amplitude to investigate the spacing between the read/write heads and the high-bandwidth disk during track seeking, head parking, vibrations, and shocks while the HDD is operational. This investigation involved the measurement of magnetic flux densities to assess the magnetic profiles. Juang *et al.*[4] performed a nonlinear modeling of the read/write head, accurately simulating its protrusion, which affects the variation in FH under different environmental conditions. Boettcher *et al.*[5] performed a dynamic FH measurement with feedforward control to optimize variations.

These studies attempted to find methods for obtaining read/write heads with optimal and efficient FH profiles. In this study, we propose a novel approach by combining a deep neural network (DNN) with an enhanced particle swarm optimization (PSO)[6–15] algorithm to find a suitable number of nodes for each hidden layer of the DNN. This innovative approach uses AI techniques, specifically a DNN, in conjunction with improved PSO techniques to predict the digital-to-analog converter (DAC) profiles of flying height 2 (FH2) with optimal performance. By employing this methodology, it is anticipated that the amount of FH measurement and the operation of the read/write head will be significantly reduced. This would result in a highly accelerated FH measurement process while maintaining comparable efficiency to conventional measurement methods. We aim to reduce the production time per unit in the manufacturing line of an HDD, as shown in Fig. 1, and for measuring the DAC profiles for flying height 1 (FH1) and FH2, which currently takes approximately 5 h (Fig. 2). In this way, we can reduce the measurement time of FH2 using an AI-based approach.

2. Experimental Procedure

2.1 Point of interest in FH measurement

The process of measuring the FH of the read/write head involves measuring the protrusion value in the DAC unit, which is from the tip of the read/write head to the position called the target FH at point B in Fig. 3. Distance A refers to the thermal protrusion in the DAC unit, the focus of this study.

Fig. 1. (Color online) FH measurement process and test time reduction concept for FH2 measurement.

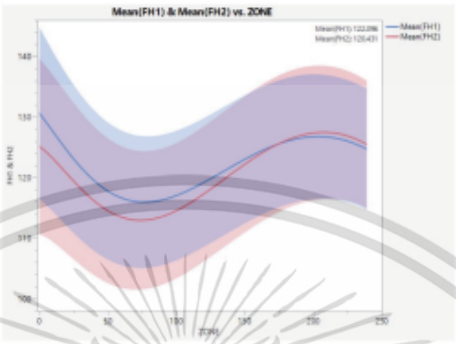


Fig. 2. (Color online) Profile of DAC for FH1 and FH2 measurements.

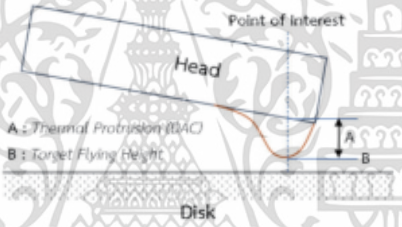


Fig. 3. (Color online) Points of interest in head reading/writing for FH measurement.

2.2 Deep neural network based on particle swarm optimization

In this study, we present a novel method of FH prediction using a DNN combined with an efficiency-enhanced PSO algorithm, called DNNpso. The objective is to find an appropriate and efficient number of nodes in the hidden layers for predicting the FH profile, which is measured in the DAC unit, as shown in Fig. 4.

2.3 Preparation of data and evaluation criterion

Preparing data for training involves utilizing information from 50 HDDs, each comprising 20 read/write heads across 240 zones. Consequently, there is a dataset of 240,000 points (50 × 20 × 240) for training the DNN model. This dataset is partitioned into 80% for training and 20% for validation. The input features for the model include the DAC value for FH1, the head, the zone, the current, and the default current. The output result is the DAC value for FH2, as shown in the

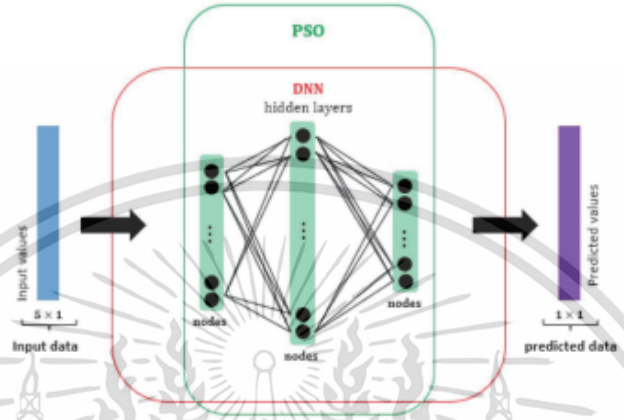


Fig. 4. (Color online) DNNpso architecture.

relationship depicted in Fig. 5. Data input relationships are sequenced using the Bootstrap Forest method, which is based on the Decision Trees method, as shown in Fig. 6. In this study, three hidden layers are utilized in the DNN. PSO is utilized to determine the appropriate number of nodes using Eqs. (1) and (2). Objective functions are defined on the basis of the R^2 metric, as outlined in Eq. (3), derived from the FH measurements of the actual HDD and the values predicted with the model. The DNN model is trained using the mean square error (MSE) as the loss function, represented by Eq. (4). The DNN utilizes the backpropagation algorithm to calculate the derivatives of the error with respect to the weights from the previous iteration, then the weights are adjusted to minimize the error, as depicted in Eq. (5).

Particle swarm optimization (PSO):

$$V_{i+1}^{i} = wV_i^i + c_1 r_1 \left(pb^i - X_i^i \right) + c_2 r_2 \left(gb - X_i^i \right) \tag{1}$$

$$X_{i+1}^{i} = X_i^i + V_{i+1}^{i} \tag{2}$$

Coefficient of determination or R^2:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(y_i - \overline{y}_i \right)^2}{\sum_{i=1}^{n} \left(y_i - \mu \right)^2} \tag{3}$$

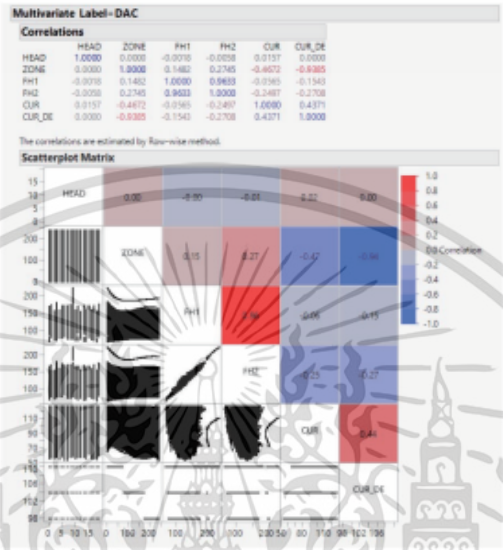


Fig. 5. (Color online) Relationship between input and output.

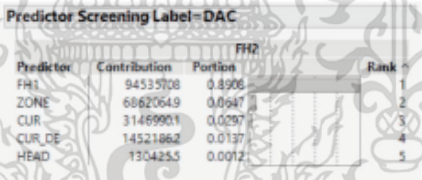


Fig. 6. (Color online) Arrangement of input relationships using Bootstrap Forest method.

Mean square error:

$$MSE = \frac{\sum_{i=1}^{n} (y_i - \overline{y}_i)^2}{n} \quad (4)$$

Backpropagation algorithm (for DNN):

$$*w_x = w_x - \alpha\left(\frac{\partial error}{\partial w_x}\right) \tag{5}$$

In these equations, y represents true values, $\bar{y}$ denotes predicted values, μ is the mean of the true values, and n is the number of data points in the dataset. Generally, a lower MSE signifies more accurate predictions, namely, the predicted values $\bar{y}$ closely align with the true values. R^2 is a metric for the goodness of fit in regression and ranges between 0 and 1, with a score of 1 indicating perfect prediction and higher values indicating better predictive performance.

The backpropagation algorithm, employed in the training phase of the DNN, involves parameters including w_x and *error*. w_x represents the weights and *error* denotes the error from the previous iteration. The parameter α signifies the learning rate. The algorithm iteratively updates the weights, forwarding them to $*w_x$, during each training cycle.

PSO aims to determine a suitable and efficient number of nodes for the hidden layers in a DNN. In this context, w represents the momentum coefficient, V is the velocity of the particle, X denotes the position of the particle, and $c1$ and $c2$ are random constants within the range [0, 1]. *pb* (*pbest*) refers to the particle's personal best, which is the highest objective function value the particle has encountered locally, and *gb* (*gbest*) is the highest objective function value encountered globally over all previous iterations. i is the number of iterations and t is the current particle.

3. Results and Discussion

From the experiments conducted with DNNpso, it was observed that the particle swarm group effectively aids in determining a suitable number of nodes for each hidden layer. In this study, a Particle Swarm of 10 was employed and 20 iterations were performed, where the 10 particles attempted to find directions to adjust the number of nodes to align with the objective function, using R^2 as the performance indicator. This process was repeated for 20 iterations, giving a consistent R^2 of ≈ 0.96, as shown in Fig. 7 and Table 1. Furthermore, we attempted to identify the optimal number of iterations for this study. It was observed that after 10 iterations, there was only a marginal improvement in R^2, as shown in Fig. 8. Consequently, we set the number of iterations at 20.

A detailed examination of the experimental results for each particle swarm reveals variations among the swarms. It is observed that a favorable R^2 obtained from DNNpso indicates a strong relationship between the actual FH measurements and the predictions from the 20% validation dataset, as shown in Fig. 9 in which the green-bordered frame indicates the highest R^2 and the best correlation, and therefore the optimal performance of PSO. The distribution of R^2 for each particle swarm is shown in Fig. 10.

After obtaining a satisfactory model, it was saved for future use. The model was then tested using data from 40 HDDs, which were not part of the 80% training or 20% validation group. The

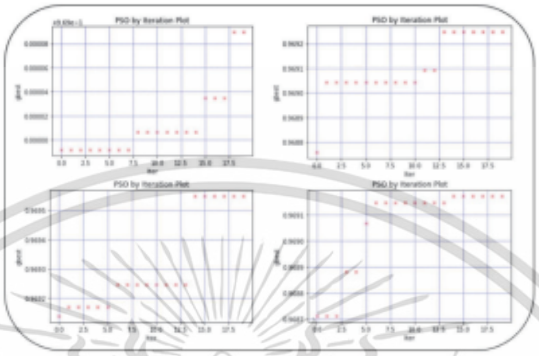


Fig. 7. (Color online) Experimental results of DNNpso with R^2 as the objective function.

Table 1
Experimental results of repeated model training with selected configuration to find the best objective function (R^2).

Run	Iterations	Particle swarm	Epochs (DNN)	*MSE*	R^2	Training time (s)
1	20	10	10	0.000203	0.9690	21.0744
2	20	10	10	0.000200	0.9692	10.0495
3	20	10	10	0.000199	0.9695	26.4638
4	20	10	10	0.000202	0.9691	45.4285

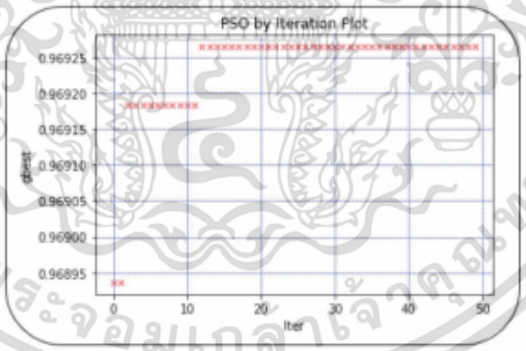


Fig. 8. (Color online) Experimental outcomes of adjusting iterations in DNNpso.

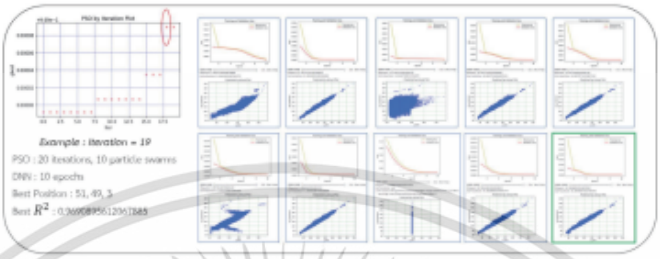


Fig. 9. (Color online) Experimental results of comparative analysis of relationship between number of iterations and R^2 in DNNpso for each particle swarm.

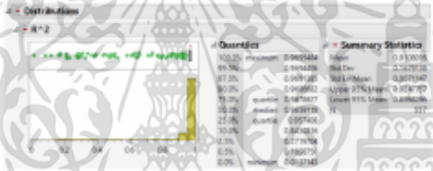


Fig. 10. (Color online) Distribution of R^2 in DNNpso for each particle swarm.

experiments yielded promising results with a correlation of ≈ 0.98 (R^2 ≈ 0.97), as shown in Fig. 11. This value indicates a high accuracy and a strong correlation between the actual FH measurements and the values predicted with the model. The boxplot exhibits uniformity, with a mean value of approximately 120 DAC and a standard deviation of around 12.8, and the cumulative distribution function (CDF) probability [Eq. (6)] was consistent between the actual FH measurements and the corresponding predictions, as shown in Fig. 12.

$$F_x(x) = P(X \le x) = \sum_{x_i \le x} P(X = x_i) \qquad (6)$$
$$x \in [0, 255],\ x : DAC$$

Furthermore, the DAC profile, which is a comprehensive representation derived from both actual measurements and predictions, demonstrates a highly satisfactory alignment between the two, as clearly illustrated in Fig. 13. Moreover, DAC profiles are dissected for each read/write head. The experimental results exhibit consistently good performance, as depicted in Fig. 14. Detailed statistical values such as mean and standard deviation can be extracted from Fig. 12.

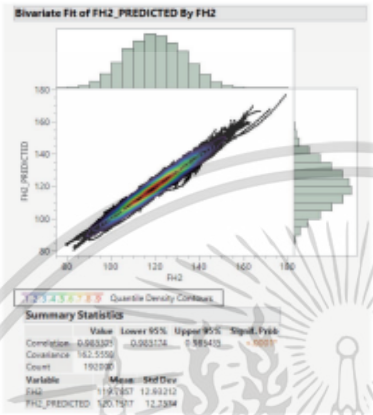


Fig. 11. (Color online) Correlation values obtained from actual measurements and predictions using the new HDD group.

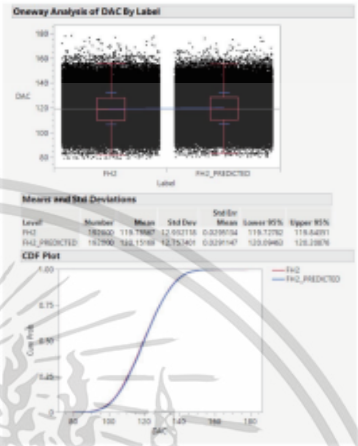


Fig. 12. (Color online) Boxplot and CDF obtained from actual measurements and predictions using the new HDD group.

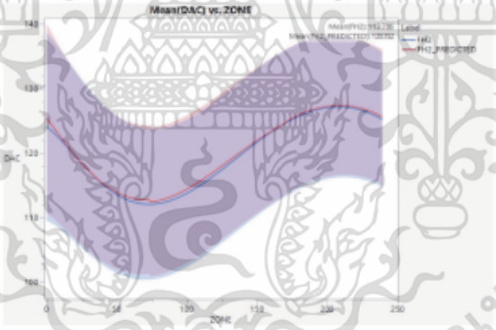


Fig. 13. (Color online) DAC profile resulting from the actual measurements and predictions of FH2 from the new HDD group.

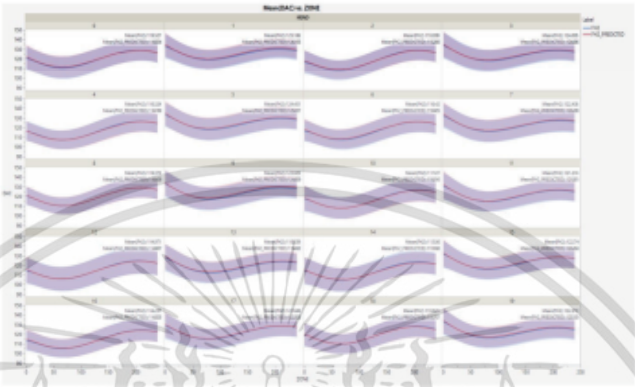


Fig. 14. (Color online) DAC profiles for each read/write head obtained from actual measurements and predictions of FH2 from the new HDD group.

4. Conclusions

We present an approach to FH measurement using the predictive power of a DNN based on PSO, termed as DNNpso. This method helps determine an appropriate and efficient number of nodes in hidden layers and is a novel approach to FH measurement that can be applied seamlessly to AI. The predictive capability significantly reduces the time required for FH measurement; therefore, it is expected to play a crucial role in streamlining the production of HDDs. The utilization of data from the source FH1 demonstrates the effectiveness of accuracy in predicting FH2 efficiently.

Acknowledgments

This research was made possible through the funding received from the School of Engineering, King Mongkut's Institute of Technology Ladkrabang for industrial purposes (grant number 2565-02-01-070).

References

1 H. Dakroub, M. C. Rao, and A. G. Sam: "Fly Height Measurement for A Disc Drive," US Patent: US6898034, Assignee: Seagate Technology LLC, Publication date on May 24, 2005 https://patents.google.com/patent/US6898034B2/en (accessed February 2023).
2 B. C. Schardt, E. Schreck, R. Sonnenfeld, Q. Haddock, and J.R. Haggis: IEEE Trans. Magn. **34** (1998) 1765. https://doi.org/10.1109/20.706699
3 V. J. Novotny: IEEE Trans. Magn. **33** (1997) 3115. https://doi.org/10.1109/20.617862
4 J. Y. Juang, T. Nakamura, B. Knigge, Y. Luo, W. C. Hsiao, K. Kuroki, F. Y. Huang, and P. Baumgart: IEEE Trans. Magn. **44** (2008) 3679. https://doi.org/10.1109/TMAG.2008.2002612

5 U. Boettcher, H. Li, R. A. de Callafon, and F. E. Talke: IEEE Trans. Magn. **47** (2011) 1823. https://doi.org/10.1109/TMAG.2011.2136328

6 W. Shi, D. Liu, X. Cheng, Y. Li, and Y. Zhao: IEEE Access **7** (2019) 104591. https://doi.org/10.1109/ACCESS.2019.2932266

7 F. E. Fernandes Junior, and G. G. Yen: Swarm Evol. Comput. **49** (2019) 62. https://doi.org/10.1016/j.swevo.2019.05.010

8 B. A. Garro and R. A. Vazquez: Comput. Intell. Neurosci. **2015** (2015) 369298. https://doi.org/10.1155/2015/369298

9 P. Konghuayrob and S. Kaitwanidvilai: Int. J. Innovative Comput., Inf. Control **13** (2017) 1777. https://doi.org/10.24507/ijicic.13.06.1777

10 T. Lawrence, L. Zhang, C. P. Lim, and E. J. Philips: IEEE Access **9** (2021) 14369. https://doi.org/10.1109/ACCESS.2021.3052489

11 J. Chen, W. Ding, X. M. Li, X. Xi, K. P. Ye, H. B. Wu, and R. X. Wu: IEEE Antennas Wirel. Propag. Lett. **20** (2021) 1993. https://doi.org/10.1109/LAWP.2021.3101703

12 Y. Gao, H. Liu, F. Niu, and Y. Tian: IEEE Access **11** (2023) 10366. https://doi.org/10.1109/ACCESS.2023.3240434

13 B. Lin, Y. Huang, J. Zhang, J. Hu, X. Chen, and J. Li: IEEE Trans. Ind. Inf. **16** (2019) 5456. https://doi.org/10.1109/TII.2019.2961237

14 X. Chen, J. Zhang, B. Lin, Z. Chen, K. Wolter, and G. Min: IEEE Trans. Parallel Distrib. Syst. **33** (2021) 683. https://doi.org/10.1109/TPDS.2021.3100298

15 A. Rajagopal, G. P. Joshi, A. Ramachandran, R. T. Subhalakshmi, M. Khari, S. Jha, K. Shankar, and J. You: IEEE Access **8** (2020) 135383. https://doi.org/10.1109/ACCESS.2020.3011502

ภาคผนวก ข

PYTHON SOURCE CODE

งานวิจัยนี้เขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาไพธอน (Python) โดยใช้วิธีการทางปัญญาประดิษฐ์ซึ่งเน้นไปทางด้านโครงข่ายประสาทเชิงลึก (Deep Neural Network : DNN) ร่วมกับการเพิ่มประสิทธิภาพกลุ่มอนุภาค (Particle Swarm Optimization : PSO) โดยแบ่งเป็น 2 ส่วน ซึ่งประกอบไปด้วย การเรียนรู้ (Training) เพื่อสร้างแบบจำลอง (Model) และบันทึกแบบจำลองและการใช้แบบจำลองเพื่อทำการทำนาย

การการเรียนรู้และสร้างแบบจำลอง

```python
import random
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import csv
import matplotlib.pyplot as plt
import time
import numpy as np
from keras.models import Model
from keras.layers import Dense, Input, Activation
from keras import backend as K
from scipy import io
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from tensorflow import keras
import scipy.stats
from sklearn.preprocessing import MinMaxScaler
import pickle
max_r2 = 0.0

def pso(cost_function, bounds, population_size=10, max_iter=50, w=0.8, c1=0.025, c2=0.025):

    # Initialize the population with random particles
    population = []
    for i in range(population_size):
        particle = [int(random.uniform(bounds[i][0], bounds[i][1])) \
                    for i in range(len(bounds))]
        velocity = [random.random() for _ in range(len(bounds))]
        population.append((particle, velocity))
        print("i", i)
        print("particle", particle, type(particle))
        print("velocity", velocity)
        print("population", population)

    # Initialize the best global position
    gbest = population[0][0]
    gbest_cost, corrTemp, trainingTimeTemp, mse, rmse = cost_function(gbest)
    for particle, _ in population[1:]:
        print("particle____", particle)
        cost, corrTemp, trainingTimeTemp, mse, rmse = cost_function(particle)
        if cost > gbest_cost: # < find min, > find max
            gbest = particle
            gbest_cost = cost
```

```
44
45      data = []
46      max_gbest = []
47      iteration = [iter for iter in range(max_iter)]
48
49      # Run the main loop
50      for _ in range(max_iter):
51          for idx, (particle, velocity) in enumerate(population):
52              # Update the velocity
53              for i in range(len(velocity)):
54                  r1 = random.random()
55                  r2 = random.random()
56                  velocity[i] = w * velocity[i] + c1 * r1 * (particle[i] - gbest[i]) \
57                                + c2 * r2 * (particle[i] - gbest[i])
58                  print("velocity[i]", velocity[i], int(velocity[i]+0.5))
59
60              # Update the position
61              for i in range(len(particle)):
62                  particle[i] = particle[i] + velocity[i]
63                  particle[i] = abs(particle[i]) # should we reject DNN and return 0.0?
64                  particle[i] = int(particle[i]) # should we reject DNN and return 0.0?
65
66              # Check if the particle's position is better than the global best
67              cost, corrTemp, trainingTimeTemp, mse, rmse = cost_function(particle)
68
69              print("iter", _, "idx", idx, "cost", cost, "gbest_cost", gbest_cost)
70              if cost > gbest_cost: # < find min, > find max
71                  print("iter_2", _, "idx", idx, "cost", cost, "gbest_cost", gbest_cost, \
72                        "gbest", gbest, "particle", particle)
73                  gbest = particle.copy() # problem can be list() and dict()
74                  gbest_cost = cost
75                  data.append([_, idx, cost, particle[0], particle[1], particle[2], gbest_cost, \
76                               gbest[0], gbest[1], gbest[2], corrTemp, trainingTimeTemp, mse, rmse])
77
78              #data.append([_, idx, cost, particle[0], particle[1], particle[2], \
79                           #gbest_cost, gbest[0], gbest[1], gbest[2], corrTemp, trainingTimeTemp, mse, rmse])
80              data.append([_, idx, cost, particle[0], particle[1], particle[2], \
81                           0, 0, 0, 0, corrTemp, trainingTimeTemp, mse, rmse])
82
83          max_gbest.append(gbest_cost)
84
85      # Plot the position
86      plt.scatter(iteration, max_gbest, s=20, c='red', marker='x', alpha=0.5)
87      plt.title('PSO by Iteration Plot')
88      plt.xlabel('Iter')
89      plt.ylabel('gbest')
90      plt.grid(color='blue', linestyle='--', linewidth=0.5)
91      plt.show()
92
93
94      # Write CSV file
95      print("data", data)
96
97      path = 'drive/MyDrive/Colab Notebooks/Prototype_PSO_AI_file.csv'
98
99      header = ['ITER', 'SWARM_IDX', 'COST', 'PAR1', 'PAR2', 'PAR3', 'GBEST_COST', \
100               'GBEST_PAR1', 'GBEST_PAR2', 'GBEST_PAR3', 'CORR', 'TRAINING_TIME', 'MSE', 'RMSE']
101     with open(path, 'w') as f:
102         writer = csv.writer(f)
103
104         # Write the header
105         writer.writerow(header)
106
107         # Write the data
108         writer.writerows(data)
109
110     return gbest, gbest_cost
```

```
111
112 def cost_function(position):
113     global max_r2
114     ly1, ly2, ly3 = position
115
116     layer1 = int(ly1)
117     layer2 = int(ly2)
118     layer3 = int(ly3)
119
120     if layer1 <= 0 or layer2 <= 0 or layer3 <= 0:
121         return 0.0, 0.0, 0.0, 0.0, 0.0
122
123     print("cost_function", layer1, layer2, layer3)
124
125     # Load data
126     dataset = io.loadmat('drive/MyDrive/Colab Notebooks/FH/FH_DAC_x200.mat')['data']
127
128     train_predictor = dataset[:,[0,1,2,4,5]]
129     train_prediction = np.array([dataset[:,3]]).T
130     #print(train_predictor)
131     #print(train_prediction)
132
133     scaler_predictor = pickle.load(open('drive/MyDrive/Colab Notebooks/FH/ \
134                                 predictor_normalization.pkl', 'rb'))
135     train_predictor = scaler_predictor.transform(train_predictor)
136     #print("train_predictor\n", train_predictor)
137
138     scaler_prediction = pickle.load(open('drive/MyDrive/Colab Notebooks/FH/ \
139                                 prediction_normalization.pkl', 'rb'))
140     train_prediction = scaler_prediction.transform(train_prediction)
141     #print("train_prediction\n", train_prediction)
142
143     X_train, X_test, y_train, y_test = train_test_split(train_predictor, train_prediction, \
144                                                         test_size=0.2, random_state=0)
145
146     # Design your network architecture
147     K.clear_session()
148     train_inputs = Input(shape = (X_train.shape[1],))
149     y = Dense(5, input_dim = X_train.shape[1], kernel_initializer = 'glorot_normal')(train_inputs)
150     y = Activation('relu')(y)
151     y = Dense(layer1, kernel_initializer = 'glorot_normal')(y)
152     y = Activation('relu')(y)
153     y = Dense(layer2, kernel_initializer = 'glorot_normal')(y)
154     y = Activation('relu')(y)
155     y = Dense(layer3, kernel_initializer = 'glorot_normal')(y)
156     y = Activation('relu')(y)
157     y = Dense(y_train.shape[1], kernel_initializer = 'glorot_normal')(y)
158     y = Activation('linear')(y)
159
160     # Train the model
161     model = Model(inputs = train_inputs, outputs = y)
162     optim = keras.optimizers.Adam(0.001, 0.9, 0.999, None, 0.0, False)
163     model.compile(loss='mse', optimizer=optim, metrics = ['accuracy'])
164     epochs = 10
165     batch_size = 10000
166
167     start = time.time()
168
169     history = model.fit(X_train,
170                         y_train,
171                         validation_data=(X_test,y_test),
172                         epochs = epochs,
173                         batch_size = batch_size,
174                         verbose = 0)
175
176     end = time.time()
177     trainingTime = end - start
178     print("Time to train the model", trainingTime)
```

```
179
180     # Get the training loss
181     loss_per_epoch = history.history['loss']
182     loss_per_epoch_val = history.history['val_loss']
183     print("mse", loss_per_epoch, loss_per_epoch[-1])
184
185     rmse = np.sqrt(loss_per_epoch)
186     print("rmse", rmse, rmse[-1])
187
188     epochs = range(1, len(loss_per_epoch) + 1)
189     plt.plot(epochs, loss_per_epoch, 'y', label='Training loss')
190     plt.plot(epochs, loss_per_epoch_val, 'r', label='Validation loss')
191     plt.title('Training and Validation loss')
192     plt.xlabel('Epochs')
193     plt.ylabel('Loss')
194     plt.legend()
195     plt.show()
196
197     # Prediction
198     val_predicted = model.predict(X_test)
199
200     scaler_prediction = pickle.load(open('drive/MyDrive/Colab Notebooks/FH/ \
201                                          prediction_normalization.pkl', 'rb'))
202     val_predicted = scaler_prediction.inverse_transform(val_predicted)
203     y_test = scaler_prediction.inverse_transform(y_test)
204
205     r2 = r2_score(y_test, val_predicted)
206     print("RSquare", r2)
207
208     corr = scipy.stats.pearsonr(y_test.T.flatten(), val_predicted.T.flatten())[0]
209     print("Correlation", corr)
210
211     plt.scatter(val_predicted, y_test, s=20, c='blue', marker='x', alpha=0.5)
212     plt.title('Predicted by Actual Plot')
213     plt.xlabel('y_test')
214     plt.ylabel('val_predicted')
215     plt.grid(color='green', linestyle='--', linewidth=0.5)
216     plt.show()
217
218     # Save the best model
219     if r2 > max_r2:
220         print("WK--r2", r2, "max_r2", max_r2)
221         max_r2 = r2
222
223         save_dir = 'drive/MyDrive/Colab Notebooks/FH/Model_PSO'
224         print(save_dir)
225
226         model_to_disk = save_dir + '.json'
227         model_json = model.to_json()
228         with open(model_to_disk, "w") as json_file:
229                 json_file.write(model_json)
230
231         weight_to_disk = save_dir + '.h5'
232         model.save_weights(weight_to_disk)
233
234     return r2, corr, trainingTime, loss_per_epoch[-1], rmse[-1]
235
236
237 # Define number of node in each hidden layer, example 3 layers
238 bounds = [(0, 50), (0, 50), (0, 50)]
239
240 best_position, best_cost = pso(cost_function, bounds)
241
242 # Print the best hyperparameters
243 print("best_position", best_position)
244 print("best_cost", best_cost)
```

การใช้แบบจำลองเพื่อทำการทำนาย

```python
import matplotlib.pyplot as plt
from keras.models import model_from_json
from keras import backend as K
from scipy import io
import scipy.stats
from sklearn.metrics import r2_score
import pickle
import csv
import time

K.clear_session()

start = time.time()

# Load data
dataset = io.loadmat('drive/MyDrive/Colab Notebooks/FH/FH_DAC_x40.mat')['data']

train_predictor = dataset[:,[0,1,2,4,5]]
train_prediction = dataset[:,3]
#print(train_predictor)
#print(train_prediction)

scaler_predictor = pickle.load(open('drive/MyDrive/Colab Notebooks/FH/predictor_normalization.pkl', 'rb'))
train_predictor = scaler_predictor.transform(train_predictor)
#print("train_predictor\n", train_predictor)

save_dir = 'drive/MyDrive/Colab Notebooks/FH/Model_PSO'
json_file = open(save_dir + '.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
model = model_from_json(loaded_model_json)
model.load_weights(save_dir + '.h5')
#print("json_file", json_file)

# Make prediction
val_predicted = model.predict(train_predictor)

scaler_prediction = pickle.load(open('drive/MyDrive/Colab Notebooks/FH/prediction_normalization.pkl', 'rb'))
val_predicted = scaler_prediction.inverse_transform(val_predicted)
#print("val_predicted\n", val_predicted)

end = time.time()
trainingTime = end - start
print("Prediction time", trainingTime)

r2 = r2_score(train_prediction.T.flatten(), val_predicted.T.flatten())
print("RSquare", r2)

corr = scipy.stats.pearsonr(train_prediction.T.flatten(), val_predicted.T.flatten())[0]
print("Correlation", corr)

plt.scatter(train_prediction.T.flatten(), val_predicted.T.flatten(), s=20, c='red', marker='x', alpha=0.5)
plt.title('Predicted by Actual Plot')
plt.xlabel('actual')
plt.ylabel('predicted')
plt.grid(color='green', linestyle='--', linewidth=0.5)
plt.show()
```