

รายงานการดำเนินการจำแนกตัวการ์ตูนโปเกมอน

ข้อมูลสำหรับรายงาน

ฐานข้อมูลตัวโปเกมอน : <https://www.kaggle.com/lantian773030/pokemonclassification>

ฐานข้อมูลไฟล์ดำเนินงาน : https://github.com/SupasanKomonlit/deep_learning_project/tree/master/classifier

ฐานข้อมูลไฟล์โมเดล : <https://drive.google.com/drive/folders/1LVzFTU-LUWzSLTyb5T8T9N8sA2F51M51?usp=sharing>

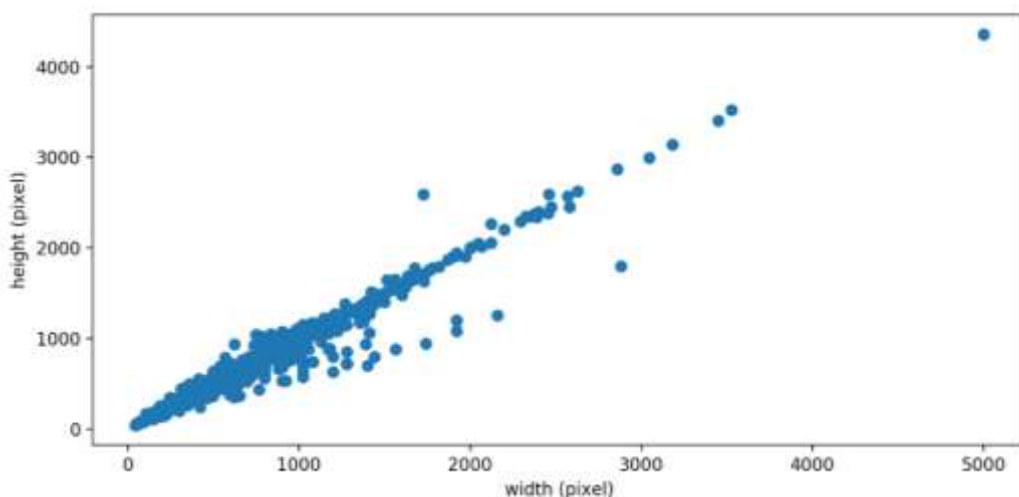
ไลบรารีในการดำเนินงาน : keras, numpy, opencv, matplotlib

การดำเนินการทดสอบผลลัพธ์ระหว่าง Convolution & Autoencoder

ตัวแปรของการทดสอบการดำเนินการ

การดำเนินการทดสอบความสามารถระหว่างการทำ Convolution กับ Autoencoder ความพยายามที่จะกำจัดโครงข่ายให้เหมือนกัน มีดังนี้

1. ข้อมูลของขาเข้าที่เป็นรูปภาพมีขนาดความกว้างยาวที่ต่างกันดังภาพที่ 1



ภาพที่ 1 ความกว้าง ความสูงของรูปภาพในฐานข้อมูล

ก่อนที่จะนำภาพดังกล่าวมาใช้ในการดำเนินการ ผู้จัดทำดำเนินการ **Crop** ให้มีขนาดจตุรัส แล้วดำเนินการ **resize** ให้มีขนาดเล็กที่สุดในฐานข้อมูล โดยถ้าเลขที่ได้เป็นเลขคี่จะบวก 1 เข้าไปให้เป็นเลขคู่

2. ขนาดของ Latent Vector

การดำเนินการทั้ง 2 ส่วนจะมีตัวแปรส่วนหนึ่งที่จะช่วยในการประเมินประสิทธิภาพคือ **latent vector** หรือ ขนาดของ **vector** ที่สอนอยู่ในโครงข่ายจะเป็นตัวเชื่อมโมเดลระหว่างส่วนต่าง ๆ

3. Layer ในการดำเนินการ

สำหรับการดำเนินการ **Layer** ในการดำเนินการจะใช้ 3 ชั้นเสมอ โดยจะเป็นในรูปแบบ 16 32 64 คุณลักษณะ ในส่วนของการดำเนินการกับรูปภาพ ในส่วนของการระบุตัวละคร จะดำเนินการในรูปแบบ นำ **latent vector** มาเข้า **layer output** โดยใช้ **activation** คือ **softmax** ดำเนินการทันที

4. Loss Value

การดำเนินการหา **loss value** ที่จะเปรียบเทียบผลลัพธ์จะใช้ตัว **categorical_crossentropy**

5. Convolution Operation

จะมีลักษณะการดำเนินงาน 3 layer มี **kernel size = (3,3)** และมี **padding = 'same'** คือมี **padding** และสุดท้ายลำดับการ **strides** หรือการขยับ **filters** จะมีเป็น 1, 2, 1 กล่าวคือการทำงานครั้งที่ 2 จะมีขนาดลด หรือเพิ่มขึ้น 2 เท่านั้นเอง

6. Activation

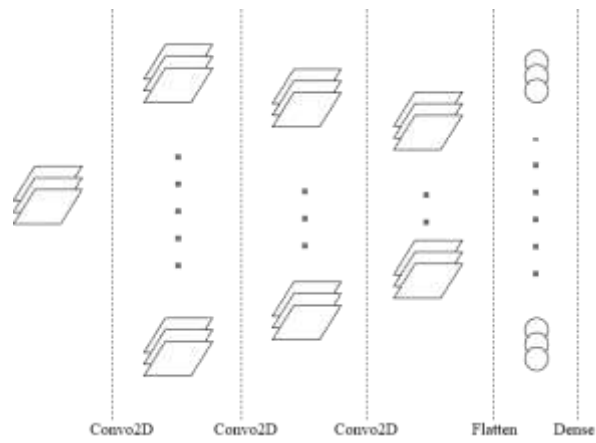
ในการดำเนินงานส่วนของการ **Activation function** จะมีการควบคุมให้เหมือนกันทั้งในส่วนของการทำ **autoencoder** และ **CNN**

7. ประเภทของข้อมูลขาเข้า

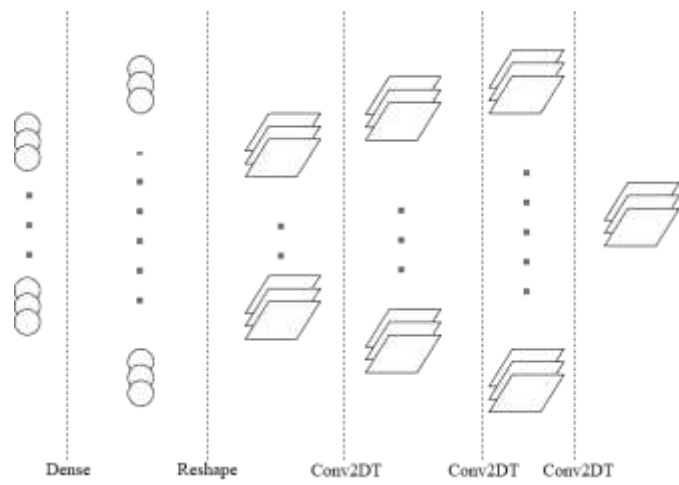
ในการดำเนินการปกติแล้วรูปภาพจะแทนค่าในแต่ละ แชนแนล แต่ละพิกเซลด้วยค่า 0 – 255 แต่การดำเนินการ **activation** การกล่าวถึง **sigmoid** การดำเนินการด้วยค่า 0 – 1 จะเหมาะสมต่อการดำเนินงาน สะดวกในเรื่องของการใช้ **function** กรณี **random** ตัวเลขด้วย **normal-distribution** เป็นต้น

โครงสร้างของระบบย่อย

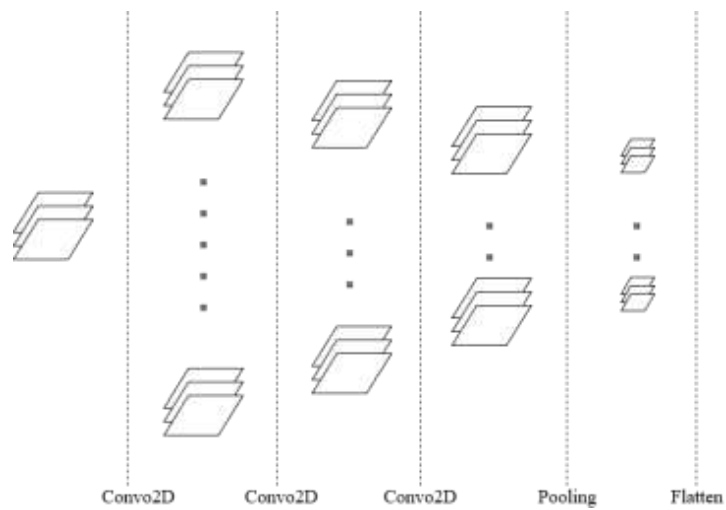
ในส่วนของโครงสร้างสามารถแบ่งเป็นระบบย่อยได้ทั้งหมด 4 ส่วนดังภาพที่ 2 – 5



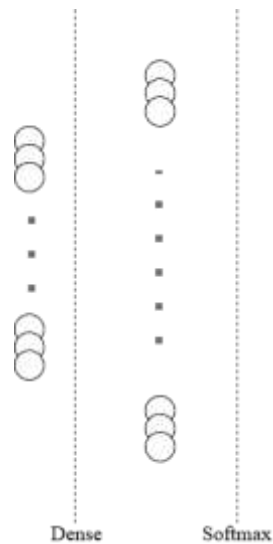
ภาพที่ 2 โครงสร้างโมเดลส่วน encoder



ภาพที่ 3 โครงสร้างโมเดลส่วน decoder

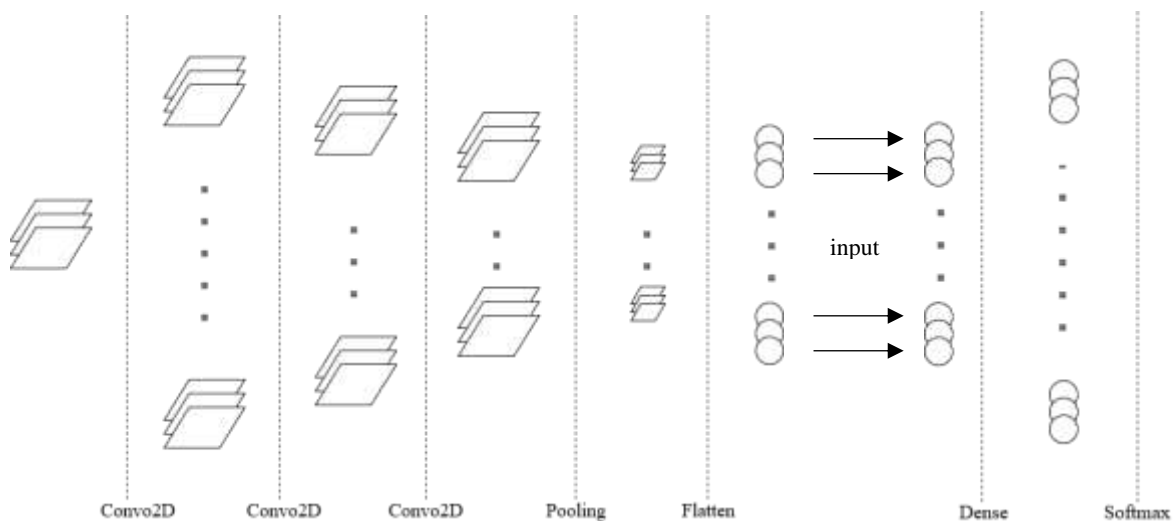


ภาพที่ 4 โครงสร้างโมเดลส่วน Convolution



ภาพที่ 5 โครงสร้างโมเดลส่วนจำแนกตัวการตูน

โครงสร้างระบบ CNN Classifier

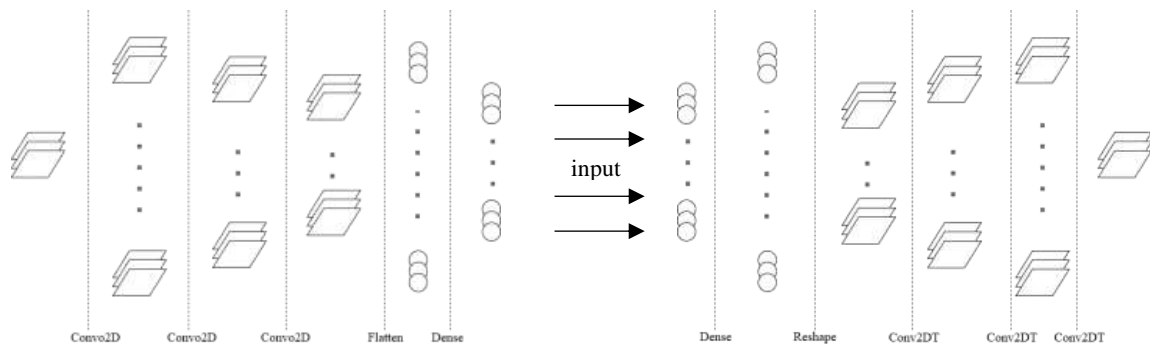


ภาพที่ 6 ภาพระบบ CNN Classifier

จากภาพที่ 6 เป็นการเชื่อมต่อการทำงานของโมเดลย่อย 2 ส่วนด้วยกัน จากภาพที่ 4 และ 5 โดยการทำงานของในส่วนฝั่งซ้ายจะรับภาพ **Input** แล้วดำเนินการจนด้วย **latent vector** ตามขนาดที่กำหนด แล้วจึงนำมาวิเคราะห์ในส่วนของฝั่งขวา

โครงสร้างระบบ Autoencoder Classifier

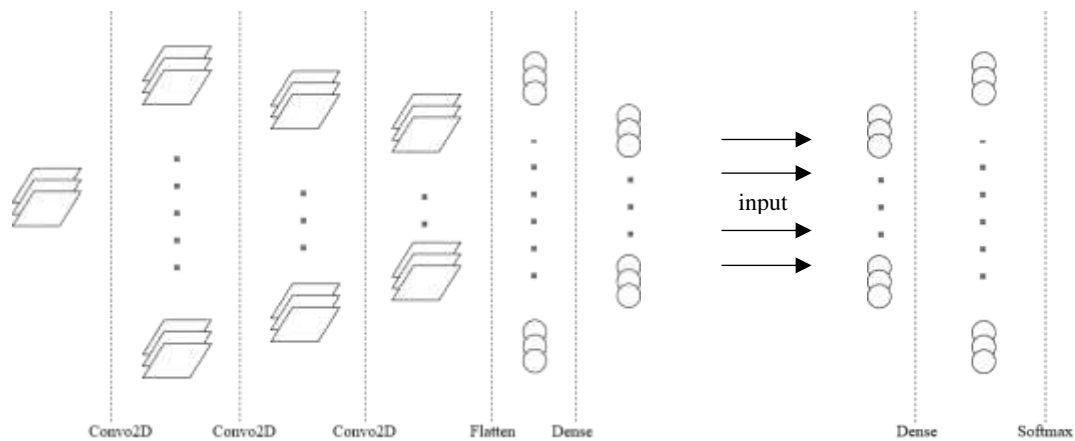
ตัวระบบจะสามารถแยกออกได้เป็น 2 ส่วนในส่วนแรกของการทำงาน **autoencoder** ที่จะทำให้การเทรนโมเดล 2 ส่วน จากภาพที่ 2 และ 3 ได้ภาพที่ 7



ภาพที่ 7 ภาพระบบ autoencoder

การดำเนินงานทั้ง 2 ส่วนนั้นจะมีลักษณะที่ทำงานตรงกันข้ามกัน แต่ใช้ตัวดำเนินการเดียวกัน activation function เหมือนกัน

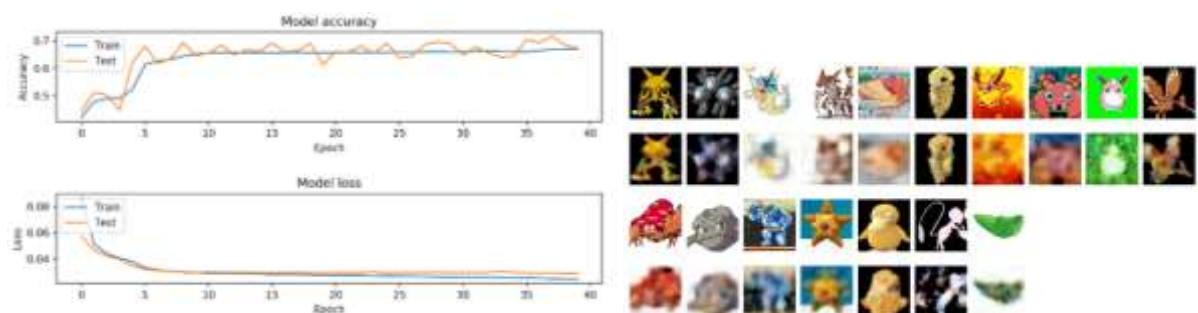
เมื่อเทรนระบบ **autoencoder** เสร็จแล้วจะดำเนินการเทรนในส่วนของ **classifier** โดยถอดโมเดลส่วน **encoder** มาดำเนินการต่อด้วยส่วนจำแนกตัวการตุ๋นได้ดังภาพที่ 8



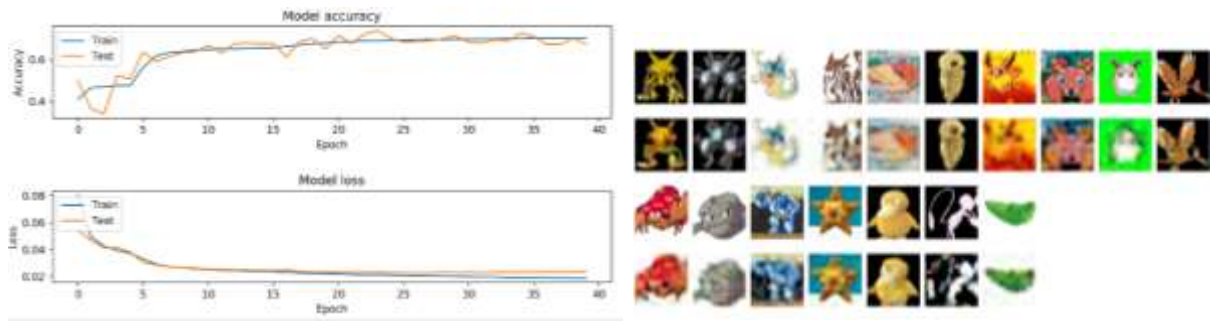
ภาพที่ 8 ภาพระบบ autoencoder classifier

ผลลัพธ์การเทรนโมเดล **Autoencoder**

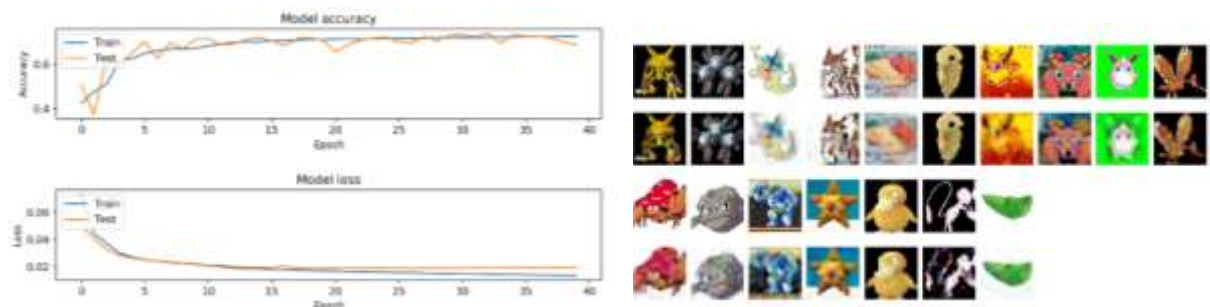
ในการทดสอบเริ่มต้นดำเนินการทดสอบโดยให้ **activation** คือ **relu** โดยมี **latent vector** ขนาด 64 128 256 512 และ 1024 ได้ผลลัพธ์ดังภาพที่ 9 – 13



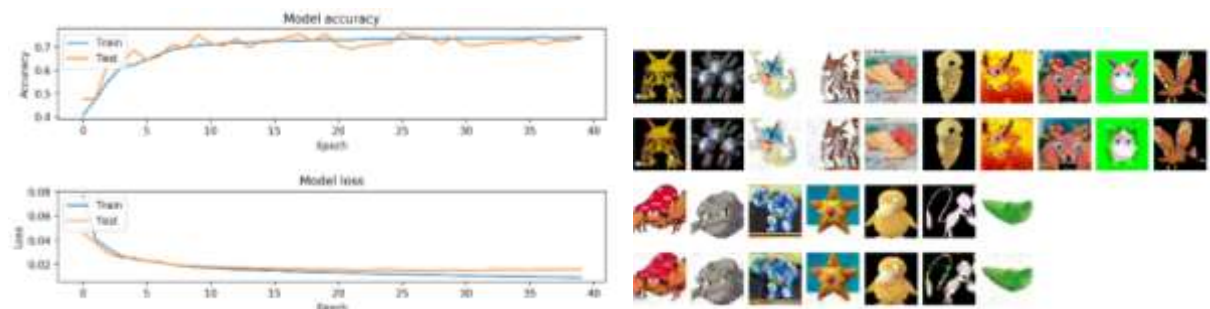
ภาพที่ 9 ผลลัพธ์กรณี **latent vector** มีขนาด 64



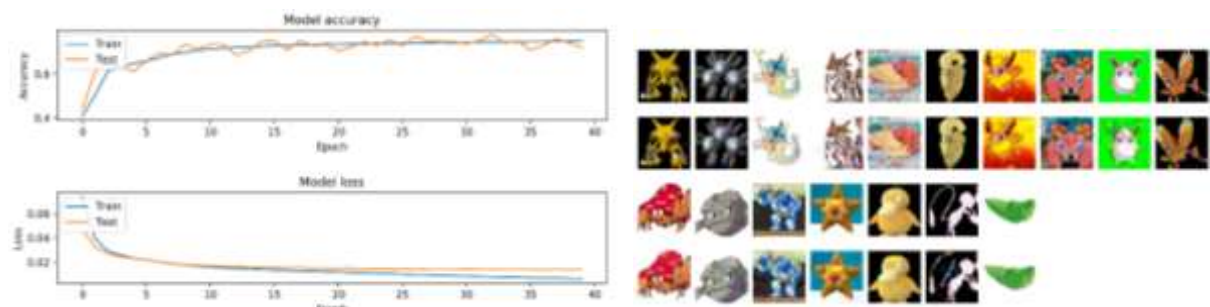
ภาพที่ 10 ผลลัพธ์กรณี latent vector มีขนาด 128



ภาพที่ 11 ผลลัพธ์กรณี latent vector มีขนาด 256



ภาพที่ 12 ผลลัพธ์กรณี latent vector มีขนาด 512

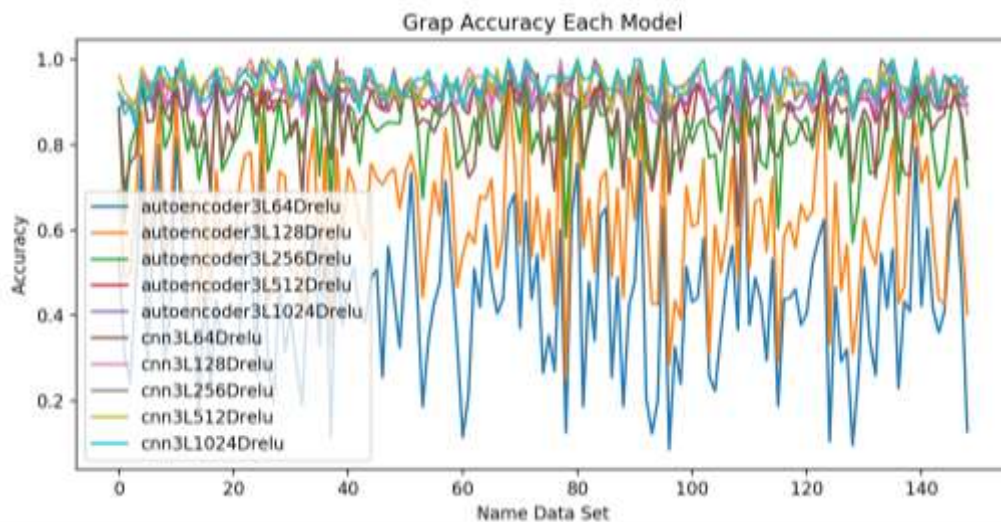


ภาพที่ 13 ผลลัพธ์กรณี latent vector มีขนาด 1024

จากภาพที่ 9 – 13 จะพบว่า การเทรนโดยมี latent vector size 512 และ 1024 ให้ผลลัพธ์ที่มีความแม่นยำสูงสุดในสองอันดับแรก ทั้งนี้สามารถสรุปเบื้องต้นได้ว่าตัว latent vector ยิ่งเยอะก็สามารถเก็บ feature ที่ใช้ในการดำเนินการส่วน decode แปลงกลับมาเป็นภาพได้ดียิ่งขึ้น

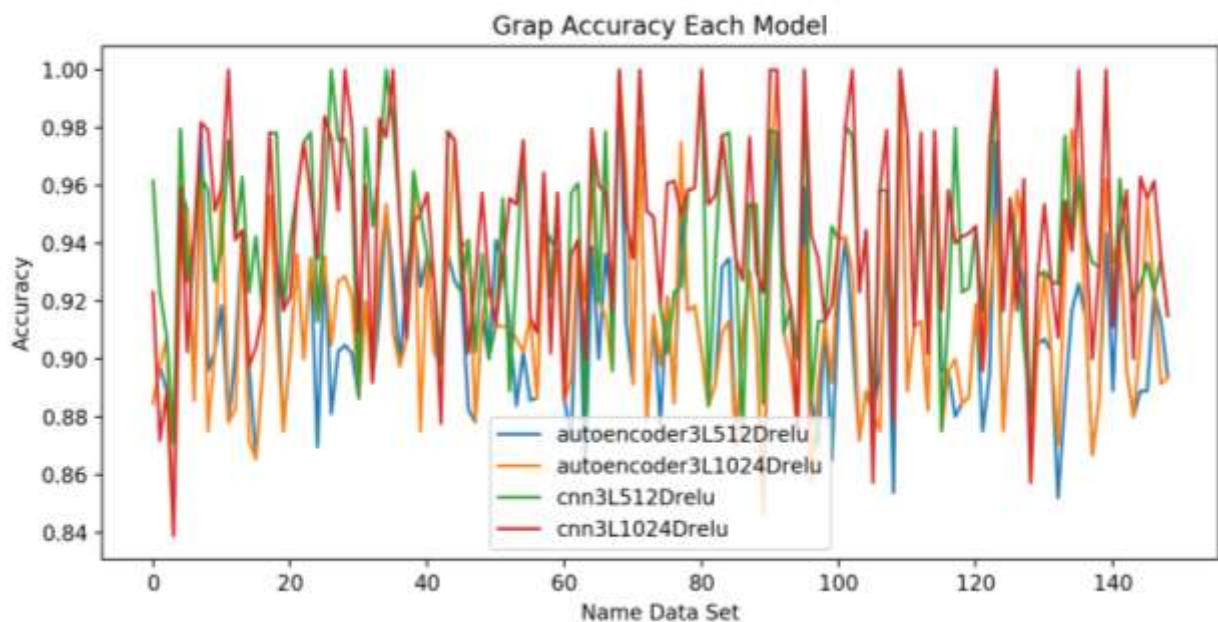
ผลลัพธ์ในการเทรนโมเดล

ภาพที่ 14 แสดงถึงผลลัพธ์รวมในกรณีการทำ Convolution Neural Network กับ Autoencoder สำหรับการทำให้ Classifier



ภาพที่ 14 ภาพผลลัพธ์การทำ Classifier โดยแกน x คือ label ข้อมูลแต่ละชุด

ผู้จัดทำได้หยิบยกในกรณี Latent Vector 512 และ 1024 มาเนื่องจากให้ผลลัพธ์โดยเฉลี่ยดีที่สุด
ดังภาพที่ 15 – 16



ภาพที่ 15 กราฟแสดงความแม่นยำระหว่าง CNN กับ Autoencoder


```
====> Result of Model from 149 label and 6781 data
Model autoencoder3L512Drelu      have average accuracy 0.90882/1
Model autoencoder3L1024Drelu     have average accuracy 0.91368/1
Model cnn3L512Drelu              have average accuracy 0.94069/1
Model cnn3L1024Drelu             have average accuracy 0.94351/1
```

ภาพที่ 16 แสดงผลลัพธ์ความแม่นยำโดยเฉลี่ย

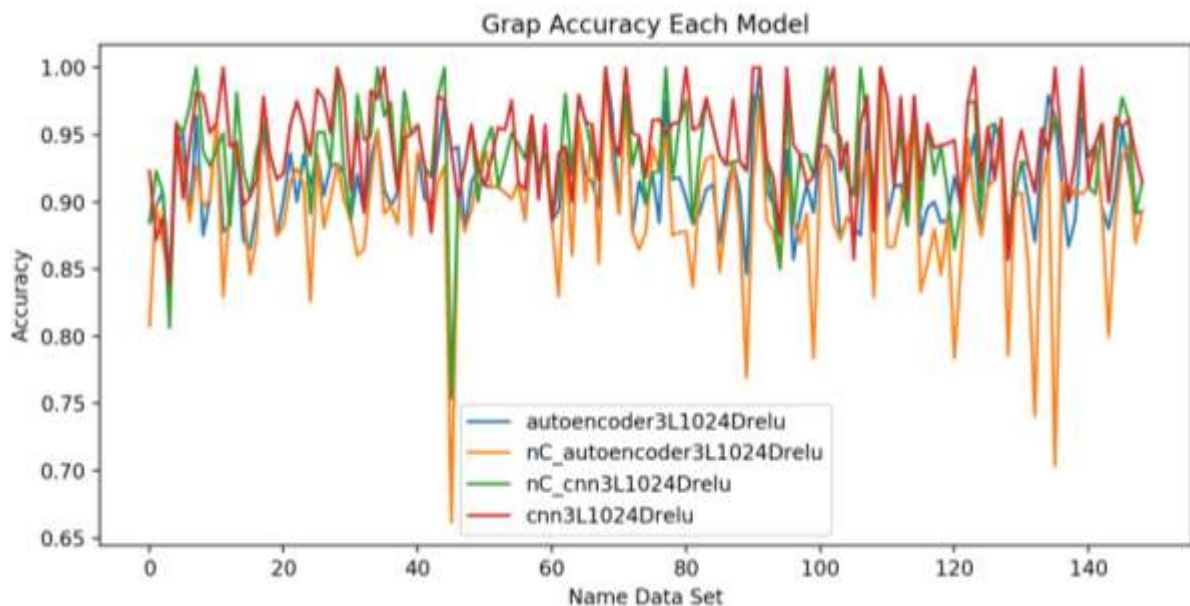
จากภาพที่ 16 แสดงให้เห็นถึงการว่าโมเดลในรูปแบบการเทรน CNN สามารถจำแนกได้ดีกว่า

การทดสอบการปรับรูปภาพเข้าสำหรับการดำเนินการ

ในการดำเนินการปรับรูปภาพจากข้อมูลข้างต้นผู้จัดทำได้ทำการ **Crop** ให้มีขนาดเป็นจัตุรัส แล้วทำการ **resize** รูปภาพ แล้วถ้าในกรณีกลับกัน ผู้จัดทำดำเนินการปรับรูปภาพเลย โดยไม่ **Crop** ก่อน ผลลัพธ์จะได้ดังภาพที่ 17 – 18 ในกรณีที่รูปผ่านการ **crop** ก่อนดำเนินการได้ผลลัพธ์ดังภาพที่ 19 - 20

```
====> Result of Model from 149 label and 6781 data
Model autoencoder3L1024Drelu     have average accuracy 0.91368/1
Model nC_autoencoder3L1024Drelu have average accuracy 0.89463/1
Model nC_cnn3L1024Drelu         have average accuracy 0.93672/1
Model cnn3L1024Drelu            have average accuracy 0.94351/1
```

ภาพที่ 17 แสดงผลลัพธ์ความแม่นยำโดยเฉลี่ยกรณีรูปภาพเข้าผ่านการ **crop**



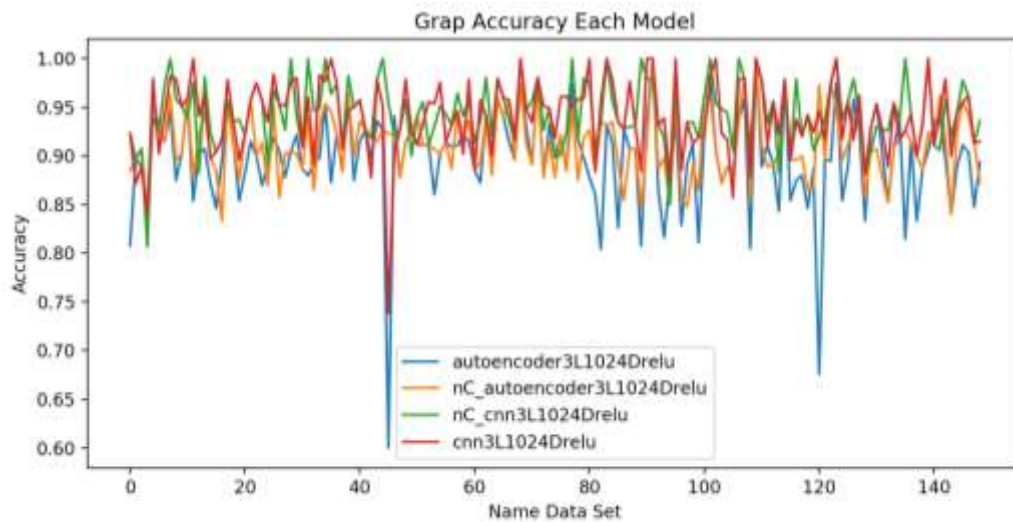
ภาพที่ 18 กราฟแสดงความแม่นยำของผลลัพธ์แยกตามประเภทข้อมูลกรณีรูปภาพเข้าผ่านการ **crop**


```

====> Result of Model from 149 label and 6781 data
Model autoencoder3L1024Drelu have average accuracy 0.89712/1
Model nC_autoencoder3L1024Drelu have average accuracy 0.91171/1
Model nC_cnn3L1024Drelu have average accuracy 0.94360/1
Model cnn3L1024Drelu have average accuracy 0.94031/1

```

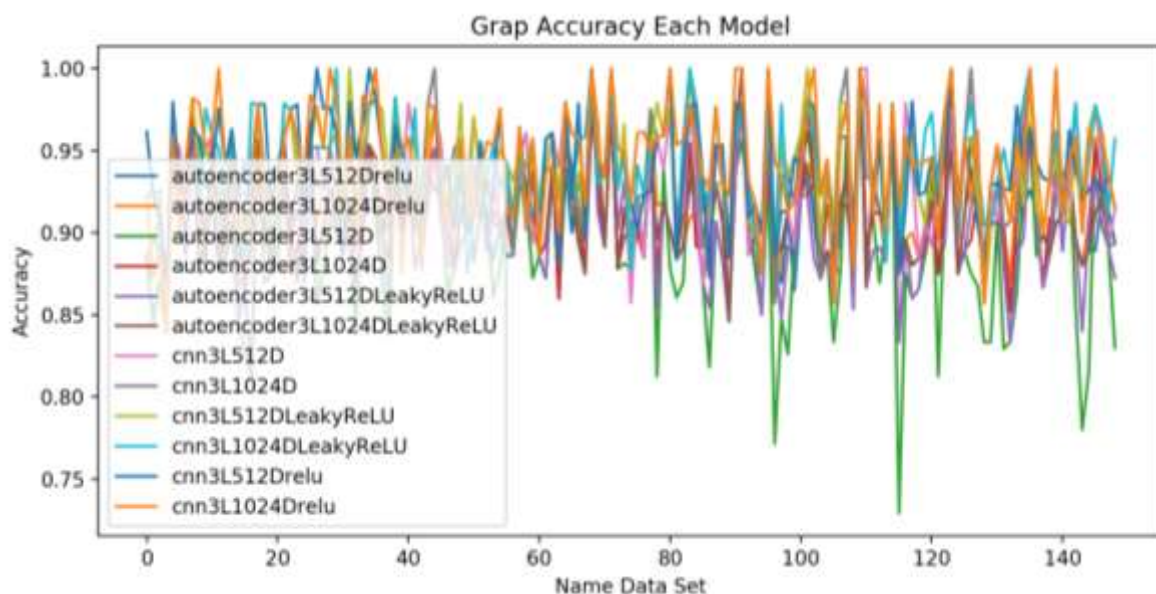
ภาพที่ 19 แสดงผลลัพธ์ความแม่นยำโดยเฉลี่ยกรณีรูปขาเข้าไม่ผ่านการ crop



ภาพที่ 20 กราฟแสดงความแม่นยำของผลลัพธ์แยกตามประเภทข้อมูลกรณารูปขาเข้าผ่านการ crop

การทดสอบการเปลี่ยนการดำเนินการส่วน Activation

จากการทดสอบที่ผ่านมาผู้จัดทำดำเนินการโดยใช้ activation คือ relu ทั้งนี้ผู้จัดทำได้ดำเนินการทดสอบในการใช้ activation เป็น linear และ LeakyReLU ได้ผลลัพธ์ดังภาพที่ 21 – 22



ภาพที่ 21 กราฟแสดงความแม่นยำของโมเดลตามประเภทข้อมูล

```
====> Result of Model from 149 label and 6781 data
```

Model autoencoder3L512Drelu	have average accuracy	0.90882/1
Model autoencoder3L1024Drelu	have average accuracy	0.91368/1
Model autoencoder3L512D	have average accuracy	0.89366/1
Model autoencoder3L1024D	have average accuracy	0.91039/1
Model autoencoder3L512DLeakyReLU	have average accuracy	0.90388/1
Model autoencoder3L1024DLeakyReLU	have average accuracy	0.91134/1
Model cnn3L512D	have average accuracy	0.93484/1
Model cnn3L1024D	have average accuracy	0.93625/1
Model cnn3L512DLeakyReLU	have average accuracy	0.94022/1
Model cnn3L1024DLeakyReLU	have average accuracy	0.94197/1
Model cnn3L512Drelu	have average accuracy	0.94069/1
Model cnn3L1024Drelu	have average accuracy	0.94351/1

ภาพที่ 22 ผลลัพธ์แสดงความแม่นยำของการทำนายโดยเฉลี่ย

จากภาพที่ 21 – 22 เมื่อมองภาพรวมทำให้สรุปได้ว่าการดำเนินการบน **activation function** คือ **relu** ได้ให้ประสิทธิภาพดีที่สุดในกรณีที่ **Latent Vector** มีขนาดที่เท่ากัน

สรุปผลการดำเนินงานและวิเคราะห์ผลลัพธ์

กรณีขนาดของ **Latent Vector**

สำหรับการจำแนกประเภทสามารถสรุปได้ว่ายังมี **Latent Vector** ขนาดใหญ่ยังสามารถทำให้ได้ผลลัพธ์มีความแม่นยำที่มากยิ่งขึ้น

เนื่องจาก **Latent Vector** ที่กล่าวถึงนี้ เปรียบเสมือน **Feature** ที่จะนำมาใช้ในการวิเคราะห์ข้อมูล ยังมีเยอะยังมีข้อมูลให้ช่วยในการตัดสินใจที่มากยิ่งขึ้น

แต่เมื่อมองในทางการดำเนินการ **autoencoder** จากภาพที่ 12 – 13 การมี **Latent Vector** ที่เยอะจะทำให้เกิดข้อมูลในส่วนของ **background** เกิด **noise** ที่มากยิ่งขึ้นในการดำเนินการ **decode** หรือ **reconstruct** ภาพจาก **latent vector**

กรณีของการดำเนินการ **Activation Function**

จากผลลัพธ์การดำเนินการระหว่าง **linear**, **LeakyReLU** กับ **relu** ได้ผลลัพธ์ที่ว่า **relu** ทำให้โมเดลมีประสิทธิภาพที่สูงที่สุด

เนื่องจากการดำเนินการ **relu** จะใช้ในรูปแบบของค่า $\max(0, x)$ เป็นการจำกัดรูปแบบข้อมูลให้ มีค่ามากกว่า 0 อยู่แล้ว จึงจะสอดคล้องกับข้อมูลขาเข้ามากกว่ากรณีอื่นๆ

กรณีของการดำเนินการ CNN กับ Autoencoder

จากผลลัพธ์การดำเนินการได้ข้อสรุปที่ว่า การดำเนินการแบบ CNN ได้ผลลัพธ์ที่ดีกว่า ทั้งนี้เนื่องจากโครงสร้างของตัวโมเดลในการจำแนกประเภทนั่นเอง

Autoencoder Classifier นั้นเป็นนำส่วนที่เป็น **output** ของส่วน **encoder** มาพิจารณา จำแนกตัวละคร การดำเนินการเทรนส่วนจำแนกนั้นจะไม่มีการปรับ **weights** ในส่วนของ **encoder** เนื่องจาก **encoder** จะต้องรักษาคุณลักษณะของการบีบอัดข้อมูลสำหรับการ **reconstruct** อยู่ จึงกล่าวได้ว่า ข้อมูลขาออกจาก **encoder** ที่จะมาเป็นข้อมูลขาเข้าในส่วนจำแนกนั้นเหมาะกับการ **reconstruct** เสียมากกว่า

CNN การดำเนินการหา **latent vector** ที่จะมาเป็นข้อมูลขาเข้าสำหรับการดำเนินการจำแนกนั้น จะมีการปรับ **weights** ไปพร้อม ๆ กัน จึงทำให้ได้โครงข่ายที่เหมาะสมกับการจำแนกมากกว่าการทำงานในส่วน of **autoencoder**

กล่าวสรุปคือ CNN จะได้ผลลัพธ์ที่ดีกว่าเนื่องจากส่วนเชื่อมต่อการจำแนกข้อมูลนั้น มีการปรับ **weights** ไปพร้อม ๆ กันเพื่อให้ได้จุด หรือข้อมูลที่จะดึงออกมาได้อย่างมีประสิทธิภาพมากที่สุดนั่นเอง

กรณีการ Crop รูปภาพกับไม่ Crop รูปภาพก่อนดำเนินการ train

จากการทดสอบพบว่าในส่วน of โมเดลที่ผ่านการเทรนโดยรูปภาพที่ทำการ **crop** ก่อน โดย CNN สามารถที่จะคงความแม่นยำได้มากกว่า นั้นหมายถึงความเสถียรของโมเดลในกรณีที่ไม่ทราบว่า **input** ที่ได้รับนั้นจะเป็นภาพที่ต้อง **crop** หรือไม่ต้อง **crop** อย่างไหนนั่นเอง

ทั้งนี้การทดสอบวิธีนี้เป็น การทดสอบในส่วน of ภาพที่ได้มาโดยตัดพิจารณาตัวโปเกมอนโดยเฉพาะ แล้วจึงไม่สามารถอธิบายถึงความเป็นไปได้ที่ชัดเจนเนื่องจากภาพที่ตัดมาข้อมูลที่ต้องการจะอยู่ตรงส่วนกลางที่ทำให้ภาพไม่ถูกตัดออกไปนั่นเอง