**สรุปเนื้อหาทั้งหมด**

**📝 example1.py - Dataset และ DataLoader พื้นฐาน**

* **TensorDataset** สำหรับข้อมูลง่าย
* **Custom Dataset** พร้อม class distribution analysis
* **DataLoader configurations** ต่างๆ (batch\_size, shuffle, drop\_last)
* **การแบ่งข้อมูล** train/val/test
* **WeightedRandomSampler** สำหรับ imbalanced data
* **SubsetRandomSampler** สำหรับ custom splitting
* **Performance optimization** และ benchmarking

**🖼️ example2.py - Image Preprocessing และ Augmentation**

* **การสร้างภาพตัวอย่าง** สำหรับทดสอบ
* **Basic transforms** (resize, crop, flip, rotation)
* **Advanced augmentation** (noise, blur, color jitter)
* **Custom Image Dataset** implementation
* **Transform compositions** สำหรับ train/validation
* **Visualization tools** สำหรับ transforms และ batches
* **Performance benchmarking** ของ augmentation techniques

**⚡ example3.py - Batch Processing และ Shuffling**

* **Custom Dataset** พร้อม metadata และ sample weights
* **Custom collate functions** สำหรับ batch processing
* **Balanced Batch Sampler** รับประกันการกระจายของ classes
* **Dynamic Batch Sampler** ปรับขนาด batch ตามความซับซ้อน
* **Shuffling analysis** และ strategies ต่างๆ
* **Memory-efficient DataLoader** configuration
* **Batch statistics analysis** และ monitoring

**🎯 example4.py - Custom Dataset Creation**

* **Base Custom Dataset** template พร้อม caching
* **Text Dataset** สำหรับ NLP tasks (tokenization, vocabulary)
* **Time Series Dataset** สำหรับ sequence prediction
* **Multi-Modal Dataset** สำหรับข้อมูลหลายประเภท
* **Dataset Factory** pattern สำหรับสร้าง datasets
* **Dataset Utilities** สำหรับการวิเคราะห์และ benchmarking
* **Custom collate functions** สำหรับแต่ละประเภท

**🎯 จุดเด่นของ Lab นี้:**

**✨ ความครบถ้วนและปฏิบัติได้จริง**

* **ครอบคลุมทุกประเภทข้อมูล**: text, image, time series, multi-modal
* **เทคนิคขั้นสูง**: caching, memory optimization, custom samplers
* **การวิเคราะห์และ monitoring**: performance, statistics, consistency

**🔧 เทคนิคที่สำคัญ**

* **Data Augmentation** สำหรับเพิ่มความหลากหลายของข้อมูล
* **Balanced Sampling** สำหรับจัดการ imbalanced datasets
* **Memory Management** สำหรับข้อมูลขนาดใหญ่
* **Custom Preprocessing** สำหรับข้อมูลเฉพาะ

**💡 การประยุกต์ใช้จริง**

* **Computer Vision**: image preprocessing และ augmentation
* **NLP**: text tokenization และ sequence processing
* **Time Series**: financial data, sensor data analysis
* **Multi-Modal**: e-commerce, social media content analysis

**📊 Performance Optimization**

* **Caching strategies** สำหรับข้อมูลที่ใช้ซ้ำ
* **Parallel loading** ด้วย num\_workers
* **Memory-efficient** batch processing
* **Custom samplers** สำหรับการกระจายข้อมูลที่เหมาะสม

**Lab นี้เป็นพื้นฐานสำคัญสำหรับการจัดการข้อมูลใน deep learning projects จริงครับ!** 🚀

**ทักษะที่ได้รับ:**

* สร้าง custom datasets สำหรับโครงการจริง
* จัดการข้อมูลประเ

Retry

T

ช่วยอธิบาย code ทั้งหมดอย่างละเอียดให้หน่อยครับทุกตัวอย่างทำเพื่ออะไรได้เรียนรู้อะไรมีแบบฝึกหัดใหม

Edit

ผมจะอธิบายโค้ดทั้ง 4 ไฟล์แบบละเอียดให้เข้าใจง่ายครับ พร้อมตัวอย่างการใช้งานจริง! 🎯

**📝 Example 1: Dataset และ DataLoader พื้นฐาน**

**🎯 ทำเพื่ออะไร:**

เรียนรู้วิธีการ **จัดเก็บและส่งข้อมูล** ให้กับ AI - เหมือนการจัดระเบียบหนังสือในห้องสมุดให้ AI อ่านได้

**🔍 อธิบายแต่ละส่วน:**

**1. TensorDataset (วิธีง่ายที่สุด)**

python

X = torch.randn(1000, 10) *# ข้อมูล 1000 samples, 10 features*

y = torch.randint(0, 3, (1000,)) *# labels 3 classes*

dataset = TensorDataset(X, y)

**อุปมา:** เหมือนการใส่ข้อมูลในกล่อง - ง่าย แต่ไม่ยืดหยุ่น

**2. Custom Dataset (ยืดหยุ่นกว่า)**

python

class SimpleCustomDataset(Dataset):

def \_\_init\_\_(self, num\_samples=1000):

*# สร้างข้อมูลตามต้องการ*

def \_\_len\_\_(self):

return self.num\_samples *# บอกว่ามีข้อมูลกี่ตัว*

def \_\_getitem\_\_(self, idx):

return self.data[idx], self.labels[idx] *# ดึงข้อมูลตัวที่ idx*

**อุปมา:** เหมือนการสร้างห้องสมุดเฉพาะ - ออกแบบได้ตามต้องการ

**3. DataLoader (ระบบส่งข้อมูล)**

python

dataloader = DataLoader(

dataset,

batch\_size=32, *# ส่งข้อมูลทีละ 32 ตัว*

shuffle=True, *# สลับลำดับ*

drop\_last=False *# เก็บข้อมูลที่เหลือไว้*

)

**อุปมา:** เหมือนรถรับส่งนักเรียน

* **batch\_size=32**: รถคันละ 32 คน
* **shuffle=True**: สุ่มที่นั่ง
* **drop\_last=False**: ถ้าเหลือ 5 คน ก็ส่งไป (ไม่ทิ้ง)

**4. WeightedRandomSampler (จัดการข้อมูลไม่สมดุล)**

python

*# ถ้ามีข้อมูล Class A: 700 ตัว, Class B: 200 ตัว, Class C: 100 ตัว*

*# สร้าง weights ให้ทุก class มีโอกาสเท่ากัน*

class\_weights = {0: 1.4, 1: 5.0, 2: 10.0} *# Class น้อย weight มาก*

sampler = WeightedRandomSampler(weights=sample\_weights, ...)

**อุปมา:** เหมือนการเรียกชื่อในห้องเรียน

* **Class A** (เยอะ): เรียกน้อยกว่า
* **Class C** (น้อย): เรียกบ่อยกว่า
* **ผลลัพธ์**: ทุกคนได้โอกาสเท่ากัน

**💡 ได้เรียนรู้อะไร:**

1. **การจัดเก็บข้อมูล** แบบมีระเบียบ
2. **การส่งข้อมูล** เป็น batch
3. **การจัดการข้อมูลไม่สมดุล**
4. **Performance optimization**

**🖼️ Example 2: Image Preprocessing และ Augmentation**

**🎯 ทำเพื่ออะไร:**

**ปรับแต่งภาพ** ให้ AI เรียนรู้ได้ดีขึ้น และ **สร้างภาพใหม่** จากภาพเดิม (Data Augmentation)

**🔍 อธิบายแต่ละส่วน:**

**1. Basic Transforms**

python

transform = transforms.Compose([

transforms.Resize((64, 64)), *# ปรับขนาด*

transforms.RandomHorizontalFlip(0.5), *# กลับซ้าย-ขวา 50%*

transforms.RandomRotation(15), *# หมุน ±15 องศา*

transforms.ToTensor(), *# แปลงเป็น tensor*

transforms.Normalize(mean=[0.485, 0.456, 0.406],

std=[0.229, 0.224, 0.225]) *# Normalize*

])

**อุปมา:** เหมือนการแต่งรูปก่อนโพสต์โซเชียล

* **Resize**: ครอปให้ขนาดเท่ากัน
* **Flip**: ถ่ายในมุมต่างๆ
* **Rotation**: หมุนภาพ
* **Normalize**: ปรับแสงให้สม่ำเสมอ

**2. Advanced Augmentation**

python

*# Geometric Transforms*

transforms.RandomAffine(degrees=15, translate=(0.1, 0.1), scale=(0.8, 1.2))

*# Color Transforms*

transforms.ColorJitter(brightness=0.4, contrast=0.4, saturation=0.4)

*# Noise*

def add\_gaussian\_noise(img):

noise = np.random.normal(0, 25, img.shape)

return np.clip(img + noise, 0, 255)

**อุปมา:** เหมือนการถ่ายรูปในสภาพแวดล้อมต่างๆ

* **การเปลี่ยนมุม**: ถ่ายจากหลายมุม
* **การเปลี่ยนแสง**: ถ่ายเวลาต่างกัน
* **การเพิ่ม noise**: ถ่ายในสภาพแวดล้อมจริง

**3. Custom Image Dataset**

python

class CustomImageDataset(Dataset):

def \_\_init\_\_(self, image\_dir, transform=None):

self.image\_dir = image\_dir

self.transform = transform

self.samples = self.\_make\_dataset() *# หาไฟล์ภาพทั้งหมด*

def \_\_getitem\_\_(self, idx):

image\_path, target = self.samples[idx]

image = Image.open(image\_path).convert('RGB')

if self.transform:

image = self.transform(image) *# ประมวลผลภาพ*

return image, target

**อุปมา:** เหมือนแกลเลอรี่รูปถ่าย

* **อ่านรูป**: เปิดไฟล์รูป
* **แต่งรูป**: ใช้ transform
* **ส่งรูป**: ให้ AI ดู

**💡 ได้เรียนรู้อะไร:**

1. **การปรับแต่งภาพ** ให้เหมาะกับ AI
2. **Data Augmentation** เพิ่มข้อมูลจากข้อมูลเดิม
3. **การจัดการไฟล์ภาพ** แบบมีระเบียบ
4. **Performance optimization** สำหรับภาพ

**⚡ Example 3: Batch Processing และ Shuffling**

**🎯 ทำเพื่ออะไร:**

เรียนรู้การ **จัดกลุ่มข้อมูล** และ **สุ่มลำดับ** อย่างชาญฉลาด เพื่อให้ AI เรียนรู้อย่างมีประสิทธิภาพ

**🔍 อธิบายแต่ละส่วน:**

**1. Custom Collate Function**

python

def custom\_collate\_fn(batch):

*# รวมข้อมูลจาก samples หลายตัวเป็น batch เดียว*

data = torch.stack([sample['data'] for sample in batch])

labels = torch.stack([sample['label'] for sample in batch])

return {

'data': data,

'labels': labels,

'batch\_size': len(batch)

}

**อุปมา:** เหมือนการจัดของในกล่อง

* **แยกประเภท**: ข้อมูล, labels แยกกัน
* **รวมกัน**: เข้ากล่องเดียวกัน
* **ติดป้าย**: บอกว่ามีกี่ชิ้น

**2. Balanced Batch Sampler**

python

class BalancedBatchSampler(BatchSampler):

*# รับประกันว่าใน batch จะมีทุก class*

def \_\_iter\_\_(self):

for class\_id in self.class\_indices:

*# เลือกจากแต่ละ class เท่าๆ กัน*

for \_ in range(self.samples\_per\_class):

batch.append(next(class\_iterators[class\_id]))

**อุปมา:** เหมือนการจัดทีมให้สมดุล

* **ทีม A**: นักกีฬาเก่ง 2 คน
* **ทีม B**: นักกีฬาปานกลาง 2 คน
* **ทีม C**: นักกีฬาใหม่ 2 คน
* **ผลลัพธ์**: ทุกทีมแข็งแรงเท่ากัน

**3. Dynamic Batch Sampler**

python

class DynamicBatchSampler(BatchSampler):

def \_\_iter\_\_(self):

if complexity < 0.5: *# ข้อมูลง่าย*

batch\_size = self.max\_batch\_size *# ใช้ batch ใหญ่*

elif complexity < 1.0: *# ข้อมูลปานกลาง*

batch\_size = medium\_size

else: *# ข้อมูลยาก*

batch\_size = self.min\_batch\_size *# ใช้ batch เล็ก*

**อุปมา:** เหมือนการสอนนักเรียน

* **เนื้อหาง่าย**: สอนคลาสใหญ่ (50 คน)
* **เนื้อหาปานกลาง**: สอนคลาสกลาง (25 คน)
* **เนื้อหายาก**: สอนคลาสเล็ก (10 คน)

**4. Shuffling Analysis**

python

*# ไม่ shuffle: ลำดับเดิมเสมอ*

[1, 2, 3, 4, 5, ...] → [1, 2, 3, 4, 5, ...]

*# มี shuffle: ลำดับสุ่มทุกครั้ง*

[1, 2, 3, 4, 5, ...] → [3, 1, 5, 2, 4, ...]

**อุปมา:** เหมือนการสอบ

* **ไม่ shuffle**: ข้อสอบเรียงลำดับ A→B→C (ทายได้)
* **มี shuffle**: ข้อสอบสุ่ม C→A→B (ทายไม่ได้)

**💡 ได้เรียนรู้อะไร:**

1. **การจัดกลุ่มข้อมูล** อย่างชาญฉลาด
2. **Balanced sampling** สำหรับข้อมูลไม่สมดุล
3. **Dynamic batching** ปรับตามความยาก
4. **Memory management** สำหรับข้อมูลใหญ่

**🎯 Example 4: Custom Dataset Creation**

**🎯 ทำเพื่ออะไร:**

เรียนรู้การสร้าง **ระบบจัดการข้อมูลเฉพาะ** สำหรับโครงการจริง - เหมือนการสร้างห้องสมุดเฉพาะทาง

**🔍 อธิบายแต่ละส่วน:**

**1. Text Dataset (สำหรับ NLP)**

python

class TextDataset(BaseCustomDataset):

def \_text\_to\_sequence(self, text):

words = text.lower().split()

sequence = [self.word\_to\_idx.get(word, self.word\_to\_idx['<UNK>'])

for word in words]

*# เพิ่ม START และ END*

sequence = [START] + sequence + [END]

*# Padding ให้ความยาวเท่ากัน*

while len(sequence) < self.max\_length:

sequence.append(PAD)

return torch.tensor(sequence)

**อุปมา:** เหมือนการแปลภาษา

* **"Hello World"** → **[15, 892]** (เลขแทนคำ)
* **เพิ่ม START/END**: **[1, 15, 892, 2]**
* **Padding**: **[1, 15, 892, 2, 0, 0, 0, ...]**

**2. Time Series Dataset (สำหรับข้อมูลเวลา)**

python

def \_create\_sequences(self):

for i in range(len(self.data) - self.sequence\_length - self.prediction\_horizon + 1):

*# ใช้ข้อมูล 50 วันที่แล้ว*

input\_seq = self.data.iloc[i:i + 50].values

*# ทำนาย 5 วันข้างหน้า*

target\_seq = self.data.iloc[i + 50:i + 55].values

**อุปมา:** เหมือนการทำนายราคาหุ้น

* **Input**: ราคาหุ้น 50 วันที่แล้ว
* **Target**: ราคาหุ้น 5 วันข้างหน้า
* **AI เรียนรู้**: ดูแล้วจากอดีต → ทำนายอนาคต

**3. Multi-Modal Dataset (หลายประเภทข้อมูล)**

python

def \_\_getitem\_\_(self, idx):

sample = self.samples[idx]

*# ประมวลผลภาพ*

image = self.\_process\_image(sample['image\_array'])

*# ประมวลผลข้อความ*

text = self.\_process\_text(sample['text'])

*# ข้อมูลตัวเลข*

numeric = torch.tensor(sample['numeric\_features'])

return {

'image': image, *# ภาพสินค้า*

'text': text, *# รีวิวสินค้า*

'numeric': numeric, *# ราคา, rating, ฯลฯ*

'label': label *# หมวดหมู่สินค้า*

}

**อุปมา:** เหมือนการขายสินค้าออนไลน์

* **ภาพสินค้า**: รูปถ่าย
* **คำอธิบาย**: รีวิว, รายละเอียด
* **ข้อมูลตัวเลข**: ราคา, คะแนน, จำนวนขาย
* **AI เรียนรู้**: วิเคราะห์ทุกอย่างร่วมกัน

**4. Dataset Factory (โรงงานสร้าง Dataset)**

python

class DatasetFactory:

@staticmethod

def create\_dataset(dataset\_type, \*\*kwargs):

if dataset\_type == 'text':

return TextDataset(\*\*kwargs)

elif dataset\_type == 'timeseries':

return TimeSeriesDataset(\*\*kwargs)

elif dataset\_type == 'multimodal':

return MultiModalDataset(\*\*kwargs)

**อุปมา:** เหมือนโรงงานรถยนต์

* **สั่ง "รถเก๋ง"** → ได้รถเก๋ง
* **สั่ง "รถบรรทุก"** → ได้รถบรรทุก
* **สั่ง "รถสปอร์ต"** → ได้รถสปอร์ต

**💡 ได้เรียนรู้อะไร:**

1. **การออกแบบ Dataset** สำหรับข้อมูลซับซ้อน
2. **การจัดการหลายประเภทข้อมูล** ในระบบเดียว
3. **Caching และ Performance** optimization
4. **Factory Pattern** สำหรับความยืดหยุ่น

**🏋️ แบบฝึกหัดปฏิบัติ**

**แบบฝึกหัดที่ 1: สร้าง Custom Dataset สำหรับ E-commerce**

python

*# TODO: สร้าง dataset สำหรับระบบแนะนำสินค้า*

class EcommerceDataset(Dataset):

def \_\_init\_\_(self, products\_file, reviews\_file):

*# 1. โหลดข้อมูลสินค้า (ชื่อ, ราคา, หมวดหมู่)*

*# 2. โหลดรีวิว (คะแนน, ความคิดเห็น)*

*# 3. รวมข้อมูลเข้าด้วยกัน*

pass

def \_\_getitem\_\_(self, idx):

*# TODO: ส่งคืน*

*# - ชื่อสินค้า (text)*

*# - ราคา (numeric)*

*# - รีวิว (text)*

*# - คะแนน (target)*

pass

**แบบฝึกหัดที่ 2: Image Augmentation สำหรับ Medical Images**

python

*# TODO: สร้าง augmentation ที่เหมาะสำหรับภาพทางการแพทย์*

medical\_transform = transforms.Compose([

*# 1. ห้ามใช้ flip แนวตั้ง (อวัยวะต้องอยู่ตำแหน่งถูก)*

*# 2. หมุนได้แค่เล็กน้อย (±5 องศา)*

*# 3. ปรับ contrast ได้ (ช่วยดูรายละเอียด)*

*# 4. ห้าม crop ส่วนสำคัญ*

])

**แบบฝึกหัดที่ 3: Time Series สำหรับพยากรณ์อากาศ**

python

*# TODO: สร้าง dataset สำหรับทำนายอุณหภูมิ*

class WeatherDataset(Dataset):

def \_\_init\_\_(self, weather\_data, sequence\_length=7, predict\_days=3):

*# ใช้ข้อมูลอากาศ 7 วันที่แล้ว*

*# ทำนายอุณหภูมิ 3 วันข้างหน้า*

*# รวม: อุณหภูมิ, ความชื้น, ความเร็วลม, ฝน*

pass

**แบบฝึกหัดที่ 4: Balanced Batch Sampler สำหรับ Medical Diagnosis**

python

*# TODO: สร้าง sampler สำหรับการวินิจฉัยโรค*

*# ปัญหา: โรคหายาก vs โรคธรรมดา*

*# - โรคธรรมดา: 90% ของข้อมูล*

*# - โรคหายาก: 10% ของข้อมูล*

*# เป้าหมาย: ให้ AI เรียนรู้โรคหายากได้ด้วย*

class MedicalBalancedSampler:

def \_\_init\_\_(self, dataset, rare\_disease\_weight=5.0):

*# TODO: ให้น้ำหนักโรคหายากมากกว่า*

pass

**คำแนะนำการทำแบบฝึกหัด:**

1. **เริ่มจากโค้ดตัวอย่าง** → แก้ไขเล็กน้อย
2. **ทดสอบเป็นขั้น** → ทีละส่วน
3. **ใช้ print() เยอะๆ** → ดูว่าข้อมูลถูกไหม
4. **สร้างข้อมูลตัวอย่างเล็กๆ** → ทดสอบก่อน

**🎯 สรุปความสำคัญ:**

**Lab นี้สอนทักษะพื้นฐานที่สำคัญที่สุดใน Deep Learning:**

1. **จัดการข้อมูล** = พื้นฐานของทุกโปรเจค
2. **Preprocessing** = ทำให้ข้อมูลพร้อมสำหรับ AI
3. **Augmentation** = เพิ่มข้อมูลจากที่มี
4. **Custom Dataset** = ปรับให้เข้ากับงานจริง

**หากเชี่ยวชาญส่วนนี้ = สามารถทำโปรเจค AI ได้จริง!** 🚀