รายงาน

โมเดลการวิเคราะห์ภาพสัตว์ชนิดต่างๆ

ผู้จัดทำ

นายชัยพร พูลสวัสดิ์	6530200096
นายภควัต จิตรพรทรัพย์	6530200321
นายรัตนพงศ์ ม่วงกระโทก	6530200410
นายวัชรากร รัศมีดิษฐ์	6530200444

นำเสนอ

อาจารย์ ชโลธร ชูทอง

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา 01418362-65 การเรียนรู้ของเครื่องเบื้องต้น
คณะวิทยาการจัดการ มหาวิทยาลัยเกษตรศาสตร์ วิทยาเขตศรีราชา
ภาคเรียนที่ 2 ปีการศึกษา 2567

ลักษณะของข้อมูล

ใช้ข้อมูล Dataset จาก Animal Image Dataset (90 Different Animals) ใน Kaggle โดยเลือกสัตว์มาทั้งหมด 6 ชนิด โดยเลือกมาทำการเทรนโมเดลชนิดละ 60 รูป รวมทั้งหมด 360 รูป ประกอบไปด้วย

1.Antelope

โดยทั่วไปแล้วแอนทิโลปมีลักษณะคล้ายกับกวางซึ่งเป็นสัตว์กีบคู่เหมือนกันและอยู่ในวงศ์กวาง (Cervidae) และคำนิยามในพจนานุกรมก็มักจะระบุเช่นนั้นว่า แอนทิโลปเป็นสัตว์จำพวกเนื้อและกวางชนิดที่มีเขาเป็นเกลียว

2.Badger

เป็นชื่อสามัญของสัตว์เลี้ยงลูกด้วยนมหลายชนิด ในอันดับสัตว์กินเนื้อที่อยู่ในวงศ์เพียงพอน (Mustelidae) โดยทั่วไปแล้วแบดเจอร์มีตัวขนาดค่อนข้างเล็กและอ้วน มีขาที่สั้น เหมาะสำหรับการขุดเป็นอย่างดี ใบหูมีขนาดเล็ก หางมีความยาวมาก หางของแบดเจอร์สามารถยาวถึง 18 ถึง 20 นิ้ว (46-51 เซนติเมตร) หรือขึ้นอยู่กับอายุ มีลายแถบ

3.Bear

หมีจัดเป็นสัตว์เลี้ยงลูกด้วยนมที่มีขนาดใหญ่ ปัจจุบันได้มีการจำแนกหมีออกเป็นทั้งหมด 8 ชนิด ได้แก่ หมีแพนด้าหรือแพนด้ายักษ์,หมีแว่น,หมีดำ,หมีสีน้ำตาลหรือหมีกริซลีย์,หมีหมา,หมีขั้วโลก,หมีควาย,หมีสล็อธ

4.Bee

ผึ้ง จัดอยู่ในประเภทสัตว์ไม่มีกระดูกสันหลัง ไฟลัมอาร์โธรพอด จัดเป็นแมลงชนิดหนึ่งอาศัยรวมกันอยู่เป็นฝูง ลักษณะทั่วไปของผึ้ง แบ่งออกได้เป็น 3 ส่วน คือ

1. ส่วนหัว ประกอบด้วยอวัยวะรับความรู้สึกต่าง ๆ ที่สำคัญ คือ
    1. ตารวม มีอยู่ 2 ตา
    2. ตาเดี่ยว
    3. หนวด
2. ส่วนนอก ประกอบด้วยปล้อง 4 ปล้อง ส่วนด้านล่างของอกปล้องแรกมีขาคู่หน้า อกปล้องกลางมีขาคู่กลางและด้านบนปล้องมีปีกคู่หน้าซึ่งมีขนาดใหญ่หนึ่งคู่ ส่วนล่างอกปล้องที่ 3 มีขาคู่ที่สามซึ่งขาหลังของผึ้งงานนี้จะมีตระกร้อเก็บละอองเกสรดอกไม้ และด้านบนจะมีปีกคู่หลังอยู่หนึ่งคู่ที่เล็กกว่าปีกหน้า
3. ส่วนท้อง ส่วนท้องของผึ้งงานและผึ้งนางพญาเราจะเห็นภายนอกเพียง 6 ปล้อง ส่วนปล้องที่ 8-10 จะหุบเข้าไปแทรกตัวรวมกันอยู่ในปล้องที่ 7 ส่วนผึ้งตัวผู้จะเห็น 7 ปล้อง

5.Beetle

ด้วง หรือ แมลงปีกแข็งนั้น นับเป็นแมลงที่มีจำนวนมากที่สุดในโลก กล่าวคือ มีประมาณร้อยละ 40 ของแมลงที่มีอยู่ทั้งหมด (ประมาณ 400,000 ชนิด) มีลักษณะเด่นโดยรวม คือ ในวัยเต็มตัวจะมีปีก 2 คู่ โดยปีกคู่หน้าเป็นปีกที่มีความแข็งเท่ากันหรือเกือบเท่ากันตลอดทั้งแผ่น

6.Bison

วัวกระทิง เป็นวัวขนาดใหญ่ในสกุลวัวกระทิงในเผ่าโบวินี มีการรับรู้ถึงสองสายพันธุ์ที่ยังหลงเหลืออยู่และสูญพันธุ์ไปแล้วมากมาย ในบรรดาสองสายพันธุ์ที่ยังมีชีวิตอยู่

ตัวอย่าง Dataset

อธิบายการทำงานของโมเดล

```python
import zipfile
import os

zip_path = "/content/drive/MyDrive/model/animal_detect.v2i.yolov5pytorch.zip"
extract_path = "/content/dataset"

with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

print("✅ Dataset Extracted Successfully!")
```

```
✅ Dataset Extracted Successfully!
```

นำเข้า dataset และทำการแตกไฟล์

```python
import glob
import shutil

output_dir = "/content/classification_dataset"
os.makedirs(output_dir, exist_ok=True)

for txt_file in glob.glob(os.path.join(extract_path, "train", "labels", "*.txt")):
    with open(txt_file, "r") as f:
        lines = f.readlines()

    if len(lines) > 0:
        class_id = lines[0].split()[0]  # ดึง Class ID
        class_dir = os.path.join(output_dir, class_id)
        os.makedirs(class_dir, exist_ok=True)

        image_file = txt_file.replace("labels", "images").replace(".txt", ".jpg")
        if os.path.exists(image_file):
            shutil.copy(image_file, class_dir)  # คัดลอกรูปไปที่โฟลเดอร์ Class

print("✅ แปลง YOLO เป็น Classification Format สำเร็จ!")
```

```
✅ แปลง YOLO เป็น Classification Format สำเร็จ!
```

ทำการดึง ID ของแต่ละรูปและคัดลอกรูปไปที่โฟลเดอร์ Class

```python
import torch
import torchvision.transforms as transforms
import torchvision.datasets as datasets
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
import torchvision.models as models
from PIL import Image
import glob
import shutil
import os
import cv2
from efficientnet_pytorch import EfficientNet

transform = transforms.Compose([
    transforms.Resize((128, 128)),
    transforms.ToTensor(),
])

train_dataset = datasets.ImageFolder(root=output_dir, transform=transform)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)

# สร้าง CNN Model
class CNN(nn.Module):
    def __init__(self, num_classes):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, kernel_size=3, stride=1, padding=1)
        self.relu = nn.ReLU()
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1)
        self.fc1 = nn.Linear(64 * 32 * 32, 128)
        self.fc2 = nn.Linear(128, num_classes)

    def forward(self, x):
        x = self.pool(self.relu(self.conv1(x)))
        x = self.pool(self.relu(self.conv2(x)))
        x = torch.flatten(x, start_dim=1)
        x = self.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

```python
# สร้าง ResNet Model
def get_efficientnet_b0_model(num_classes):
    model = EfficientNet.from_pretrained('efficientnet-b0')  # โหลดโมเดล pretrained
    model._fc = nn.Linear(model._fc.in_features, num_classes)  # เปลี่ยน layer สุดท้ายให้เป็นจำนวน class ของคุณ
    return model

# สร้าง mobilenet
def get_mobilenet_model(num_classes):
    model = models.mobilenet_v2(pretrained=True)  # โหลดโมเดล MobileNetV2
    model.classifier[1] = nn.Linear(model.classifier[1].in_features, num_classes)  # ปรับ Fully Connected Layer
    return model

#  ✅ Train Function
def train_model(model, optimizer, train_loader, num_epochs=10, device="gpu"):
    model.train()
    for epoch in range(num_epochs):
        running_loss = 0.0
        for images, labels in train_loader:
            images, labels = images.to(device), labels.to(device)

            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            running_loss += loss.item()

        print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {running_loss/len(train_loader):.4f}")
```

```python
# ✅ ตั้งค่า Training
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

num_classes = len(train_dataset.classes)
cnn_model = CNN(num_classes).to(device)
ef_model = get_efficientnet_b0_model(num_classes).to(device)
mobilenet_model = get_mobilenet_model(num_classes).to(device)

criterion = nn.CrossEntropyLoss()
optimizer_cnn = optim.Adam(cnn_model.parameters(), lr=0.001)
optimizer_ef = torch.optim.Adam(ef_model.parameters(), lr=0.001)
optimizer_mobilenet = torch.optim.Adam(mobilenet_model.parameters(), lr=0.001)

# ✅ Train Model
train_model(cnn_model, optimizer_cnn, train_loader, num_epochs=10, device=device)
train_model(ef_model, optimizer_ef, train_loader, num_epochs=10, device=device)
train_model(mobilenet_model, optimizer_mobilenet, train_loader, num_epochs=10, device=device)

for epoch in range(10):
    model.train()
    running_loss = 0.0

    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()

    print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {running_loss/len(train_loader):.4f}")

print("🎉 Training Finished!")
```

ทำการสร้างโมเดลเพื่อใช้ในการTrainโดยเลือกใช้โมเดล CNN Resnet และ Mobilenet

```
[ ]  from torch.utils.data import random_split

     # กำหนดสัดส่วน Train / Validation
     train_size = int(0.8 * len(train_dataset))  # 80% Train
     val_size = len(train_dataset) - train_size  # 20% Validation

     train_data, val_data = random_split(train_dataset, [train_size, val_size])

     # สร้าง DataLoader
     train_loader = DataLoader(train_data, batch_size=32, shuffle=True)
     val_loader = DataLoader(val_data, batch_size=32, shuffle=False)

     print(f"✅ Train Data: {len(train_data)} images")
     print(f"✅ Validation Data: {len(val_data)} images")
```

```
✅ Train Data: 604 images
✅ Validation Data: 152 images
```

กำหนดสัดส่วน Train และ Valifation

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import numpy as np

def evaluate_model(model, val_loader, device):
    model.eval()  # เปลี่ยนโมเดลเป็นโหมดประเมินผล (Evaluation)

    all_preds = []
    all_labels = []

    with torch.no_grad():
        for images, labels in val_loader:
            images, labels = images.to(device), labels.to(device)

            outputs = model(images)
            _, predicted = torch.max(outputs, 1)

            all_preds.extend(predicted.cpu().numpy())  # เก็บค่าที่โมเดลพยากรณ์
            all_labels.extend(labels.cpu().numpy())  # เก็บค่าจริง

    accuracy = accuracy_score(all_labels, all_preds)
    precision = precision_score(all_labels, all_preds, average='weighted')
    recall = recall_score(all_labels, all_preds, average='weighted')
    f1 = f1_score(all_labels, all_preds, average='weighted')

    print("\n📊 Model Performance Metrics:")
    print(f"✅ Accuracy:  {accuracy:.4f}")
    print(f"✅ Precision: {precision:.4f}")
    print(f"✅ Recall:    {recall:.4f}")
    print(f"✅ F1-score:  {f1:.4f}")

# ประเมินผลโมเดล CNN
print("Evaluating CNN Model:")
evaluate_model(cnn_model, val_loader, device)

# ประเมินผลโมเดล ResNet
print("Evaluating EfficientNet Model:")
evaluate_model(ef_model, val_loader, device)

# ประเมินผลโมเดล VGG
print("Evaluating MobileNet Model:")
evaluate_model(mobilenet_model, val_loader, device)
```

ทำการประเมินผลการคำนวณของโมเดลต่างๆ โดยจะมีค่า Accuracy Precision Recall และ F1 score แล้วจะได้ผลลัพธ์ดังนี้

```
Evaluating CNN Model:

📊 Model Performance Metrics:
✅ Accuracy:  0.9934
✅ Precision: 0.9937
✅ Recall:    0.9934
✅ F1-score:  0.9934
Evaluating EfficientNet Model:

📊 Model Performance Metrics:
✅ Accuracy:  0.8421
✅ Precision: 0.8975
✅ Recall:    0.8421
✅ F1-score:  0.8469
Evaluating MobileNet Model:

📊 Model Performance Metrics:
✅ Accuracy:  0.9934
✅ Precision: 0.9937
✅ Recall:    0.9934
✅ F1-score:  0.9934
```

```python
import requests
from PIL import Image
import torchvision.transforms as transforms
import torch

# URL ของรูปที่ต้องการทดสอบ
image_url = "https://th.bing.com/th/id/OIP.1xu0K6Htrhjg-67mXKBwaAHaF3?rs=1&pid=ImgDetMain"  # 🔺 เปลี่ยน URL เป็นรูปที่ต้องการใช้

# ดาวน์โหลดไฟล์จาก URL และบันทึกเป็น "test.jpg"
image_path = "test.jpg"
response = requests.get(image_url, stream=True)

if response.status_code == 200:
    with open(image_path, "wb") as file:
        for chunk in response.iter_content(1024):
            file.write(chunk)
    print(f"✅ Downloaded image to {image_path}")
else:
    print("❌ Failed to download image")
```

```
✅ Downloaded image to test.jpg
```

ทำการนำเข้ารูปที่ต้องการทดสอบ แล้วดาวน์โหลดรูปให้เป็นไฟล์ test.jpg

```python
from PIL import Image

def predict_image(model, image_path, class_names, device="cpu"):
    model.eval()
    image = Image.open(image_path).convert("RGB")

    transform = transforms.Compose([
        transforms.Resize((128, 128)),
        transforms.ToTensor(),
    ])

    image = transform(image).unsqueeze(0).to(device)

    with torch.no_grad():
        outputs = model(image)
        probabilities = torch.softmax(outputs, dim=1)
        confidence, predicted_class = torch.max(probabilities, 1)

    # Mapping index to class name
    class_map = {
        0: "antelope",
        1: "badger",
        2: "bear",
        3: "bee",
        4: "beetle",
        5: "bison",
    }

    predicted_index = predicted_class.item()

    if predicted_index in class_map:
        predicted_label = class_map[predicted_index]
    else:
        predicted_label = "Unknown"

    print(f"🔍 Prediction: {predicted_label}")
    print(f"🎯 Confidence: {confidence.item() * 100:.2f}%")
# ทดสอบการพยากรณ์ภาพ
test_image_path = "/content/test.jpg"
print("CNN Model Predict :")
predict_image(cnn_model, test_image_path, train_dataset.classes)
print("Efficient Model Predict :")
predict_image(ef_model, test_image_path, train_dataset.classes)
print("Mobilenet Model Predict :")
predict_image(mobilenet_model, test_image_path, train_dataset.classes)
```

```
CNN Model Predict :
🔍 Prediction: bear
🎯 Confidence: 95.91%
Efficient Model Predict :
🔍 Prediction: bison
🎯 Confidence: 100.00%
Mobilenet Model Predict :
🔍 Prediction: bison
🎯 Confidence: 90.34%
```

ทำการ predict โดยใช้ภาพที่มีขนาด 128 * 128 และให้โมเดล predict ออกมาว่าเป็นสัตว์ชนิดไหน มีความมั่นใจกี่เปอร์เซ็นต์

```python
def predict_and_draw(model, image_path, class_names, output_path="output.jpg"):
    model.eval()
    image = Image.open(image_path).convert("RGB")
    transform = transforms.Compose([
        transforms.Resize((128, 128)),
        transforms.ToTensor(),
    ])
    image_tensor = transform(image).unsqueeze(0).to(device)

    with torch.no_grad():
        outputs = model(image_tensor)
        probabilities = torch.softmax(outputs, dim=1)
        confidence, predicted_class = torch.max(probabilities, 1)

    predicted_label = class_names[predicted_class.item()]
    accuracy = confidence.item() * 100

    # โหลดภาพต้นฉบับด้วย OpenCV
    image_cv = cv2.imread(image_path)
    height, width, _ = image_cv.shape

    # วาด label ลงบนภาพ
    label = f"{predicted_label}: {accuracy:.2f}%"
    font = cv2.FONT_HERSHEY_SIMPLEX
    cv2.putText(image_cv, label, (10, 30), font, 1, (0, 255, 0), 2, cv2.LINE_AA)

    # บันทึกภาพที่มี label
    cv2.imwrite(output_path, image_cv)
    print(f"✅ ผลลัพธ์ถูกบันทึกที่: {output_path}")

# ทดสอบการพยากรณ์และบันทึกภาพพร้อม label
test_image_path = "/content/test.jpg"  # แก้เป็นภาพที่ต้องการ
output_image_cnn = "/content/output_cnn.jpg"
output_image_ef = "/content/output_ef.jpg"
output_image_mobile = "/content/output_vgg.jpg"

predict_and_draw(cnn_model, test_image_path, train_dataset.classes, output_image_cnn)
predict_and_draw(ef_model, test_image_path, train_dataset.classes, output_image_ef)
predict_and_draw(mobilenet_model, test_image_path, train_dataset.classes, output_image_mobile)
```

```
✅ ผลลัพธ์ถูกบันทึกที่: /content/output_cnn.jpg
✅ ผลลัพธ์ถูกบันทึกที่: /content/output_ef.jpg
✅ ผลลัพธ์ถูกบันทึกที่: /content/output_vgg.jpg
```

ทำการ output รูปออกมา โดยมีข้อความบอกว่า เป็นสัตว์ชนิดใดมีความถูกต้องกี่เปอร์เซ็นต์และความมั่นใจกี่เปอร์เซ็นต์

ข้อเสนอแนะ

โมเดลนี้ควรมีการDetectภาพสัตว์หลายชนิดในภาพเดียวได้ โดยที่ยังมีความแม่นยำอยู่