6510503522 ปุณยภา เจริญสุข

การจำแนกพันธุ์นกด้วยข้อมูลเสียง

Bird Species Classification by Sounds

Github : [6510503522_DeepLearning_project.ipynb]

1. วัตถุประสงค์

โครงงานนี้มีวัตถุประสงค์เพื่อศึกษาและพัฒนาแบบจำลอง Convolutional Neural Network (CNN) สำหรับการจำแนกพันธุ์นกจากข้อมูลเสียงร้อง โดยการแปลงสัญญาณเสียงดิบให้เป็นภาพ Mel Spectrogram ซึ่งใช้เป็น Input Feature หลัก

การวิจัยจะมุ่งเน้นการปรับแต่งสถาปัตยกรรม CNN เพื่อให้ได้ประสิทธิภาพในการจำแนกพันธุ์นกอย่างอัตโนมัติที่สามารถช่วยให้นักวิจัยสามารถติดตามและสำรวจความหลากหลายทางชีวภาพของนกในพื้นที่ได้อย่างรวดเร็วและไม่รบกวนธรรมชาติ การสร้างแบบจำลองที่มีประสิทธิภาพจะช่วยเติมเต็มช่องว่างในการอนุรักษ์ธรรมชาติ และเป็นก้าวสำคัญในการใช้เทคโนโลยีเพื่อการศึกษาด้านนิเวศวิทยาอย่างยั่งยืน

2. การใช้ Deep Learning

2.1. ข้อดีในการใช้ Deep Learning

เนื่องจากข้อมูลเสียงมีความซับซ้อนสูงซึ่งมีปัญหาเรื่อง Noise จึงมีความเหมาะสมมากที่จะใช้ Deep Learning ในการจำแนกสายพันธุ์ เพราะโมเดลสามารถเรียนรู้ได้เองระหว่างการฝึกฝน

โดยในงานนี้ Deep Learning จะฝึกฝนด้วยข้อมูลจากภาพ Mel Spectrogram โดยการทำงานนี้ไม่ต้องพึ่งพาการกำหนดคุณลักษณะด้วยตัวผู้สร้างเอง (Manual Feature Engineering) เหมือนวิธีดั้งเดิม ทำให้สามารถจับรูปแบบเสียงที่เป็นเอกลักษณ์เฉพาะของนกแต่ละชนิดได้อย่างมีประสิทธิภาพมากกว่า

2.2. เปรียบเทียบกับการแก้ปัญหาอื่นๆ

Features	CNN	Machine Learning แบบดั้งเดิม	Rule based model
การทำงาน	เรียนรู้คุณลักษณะและจำแนกอัตโนมัติจากข้อมูลดิบ **[1]**	สร้างแบบจำลองจากคุณลักษณะที่สกัดแล้วโดยมนุษย์ **[2]**	ตัดสินใจตาม กฎ IF-THEN ที่ถูกเขียนขึ้นโดยมนุษย์
ความแม่นยำ	แม่นยำมาก เพราะจัดการความซับซ้อนและความแปรปรวนของเสียงในธรรมชาติได้ดี **[1]**	แม่นยำสำหรับปัญหาที่ไม่ซับซ้อน	ค่อนข้างต่ำ ไม่สามารถปรับตัวเข้ากับความหลากหลายของเสียงธรรมชาติได้
ความต้องการข้อมูล	มาก	ปานกลาง	ต่ำ

3. สถาปัตยกรรม Convolutional Neural Network

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 128, 128]	320
ReLU-2	[-1, 32, 128, 128]	0
BatchNorm2d-3	[-1, 32, 128, 128]	64
MaxPool2d-4	[-1, 32, 64, 64]	0
Conv2d-5	[-1, 64, 64, 64]	18,496
ReLU-6	[-1, 64, 64, 64]	0
BatchNorm2d-7	[-1, 64, 64, 64]	128
MaxPool2d-8	[-1, 64, 32, 32]	0
Conv2d-9	[-1, 128, 32, 32]	73,856
ReLU-10	[-1, 128, 32, 32]	0
BatchNorm2d-11	[-1, 128, 32, 32]	256
MaxPool2d-12	[-1, 128, 16, 16]	0
Conv2d-13	[-1, 256, 16, 16]	295,168
ReLU-14	[-1, 256, 16, 16]	0
BatchNorm2d-15	[-1, 256, 16, 16]	512
MaxPool2d-16	[-1, 256, 8, 8]	0
Dropout-17	[-1, 16384]	0
Linear-18	[-1, 512]	8,389,120
ReLU-19	[-1, 512]	0
BatchNorm1d-20	[-1, 512]	1,024
Dropout-21	[-1, 512]	0
Linear-22	[-1, 114]	58,482

โมเดลนี้มี 4 Convolutional ตามด้วย Fully Connected Layer

โดยใน Convolutional layer 1-4 จะประกอบด้วยการสร้างบล็อกของ Convolution → ReLU → Batch Normalization → Max Pooling โดยมีหน้าที่หลักคือการเปลี่ยน Mel Spectrogram ขนาด (1, 128, 128) ให้เป็นเวกเตอร์คุณลักษณะขนาดใหญ่ (Flattened size: 16,384)

ในส่วนของ Fully Connected Layer จะนำเวกเตอร์คุณลักษณะขนาด 16,384 เข้าสู่ชั้น Linear Layer ขนาด 512 ก่อนจะส่งต่อไปยังชั้น Linear Layer สุดท้ายเพื่อจำแนกเป็น NUM_CLASSES (นก 114 พันธุ์)

Activation Functions: ใช้ ReLU ทั้งหมดเพื่อเพิ่ม Non-linearity

BatchNorm: ใช้ BatchNorm หลัง Convolution และ Linear Layer เพื่อช่วยเรื่องความเสถียรของการ train และช่วยให้ train ได้เร็วขึ้น

Dropout: ใช้ใน Fully Connected Layer เพื่อลด Overfitting

4. ชุดข้อมูล

ชุดข้อมูลที่ใช้คือ *Sound Of 114 Species Of Birds Till 2022* dataset จาก Kaggle (**LINK** : Sound Of 114 Species Of Birds Till 2022 : DATASET **[3]**) ที่เก็บไฟล์เสียงทั้งหมด 2161 ไฟล์ของนกนานาพันธุ์ทั้งหมด 114 พันธุ์

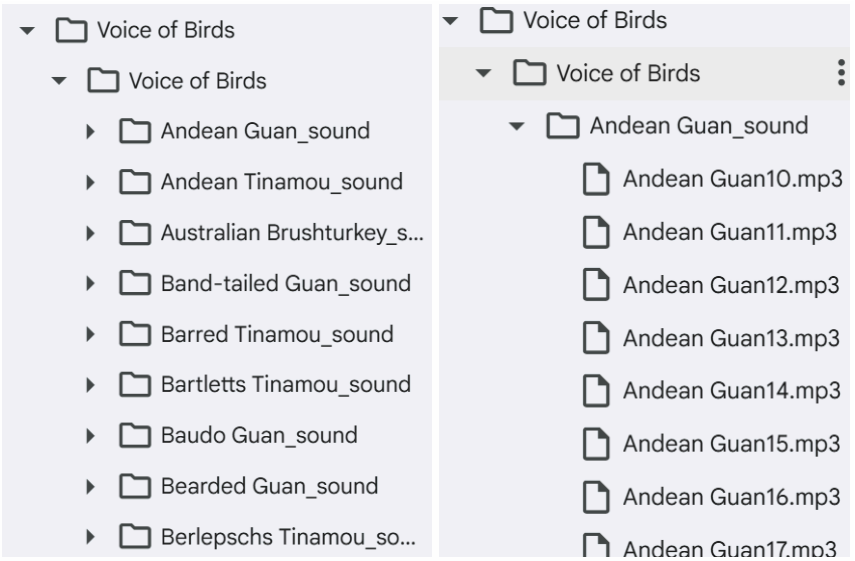
ในชุดข้อมูลมี feature ทั้งหมดดังนี้

- **common_name -** ชื่อโดยทั่วไป
- **scientific_name -** ชื่อวิทยาศาสตร์
- **recordist_name -** ชื่อผู้บันทึก
- **recording_length -** ความยาวของไฟล์เสียง
- **Date -** วันที่บันทึก
- **TYPE -** ประเภทของพฤติกรรมนก
- **xc_id -** ID พิเศษ
- **Time -** เวลาที่บันทึก
- **Country -** ประเทศของเป็นที่อยู่อาศัยของนก
- **Download_link -** ส่วนเชื่อมไปยังไฟล์เสียงนั้นๆ

ตัวอย่าง dataset

	common_name	scientific_name	recordist_name	recording_length	Date	TYPE	xc_id	Time	Country	Download_link
0	Common Ostrich	Struthio camelus australis	Frank Lambert	0:53	2019-10-30	call	XC516153	08:05	South Africa	https://xeno-canto.org/516153/download
1	Common Ostrich	Struthio camelus	Jeremy Hegge	0:26	2014-11-20	call	XC208209	04:00	South Africa	https://xeno-canto.org/208209/download
2	Common Ostrich	Struthio camelus	Jeremy Hegge	0:04	2014-11-21	call	XC208128	06:00	South Africa	https://xeno-canto.org/208128/download
3	Common Ostrich	Struthio camelus	Derek Solomon	0:11	2010-02-09	call	XC46725	07:00	South Africa	https://xeno-canto.org/46725/download
4	Common Ostrich	Struthio camelus	Morioka Zoological Park ZOOMO	1:47	2021-09-06	voice during egg laying, zoo collection	XC675445	17:00	Japan	https://xeno-canto.org/675445/download

ตัวอย่าง file path ในการเข้าถึงไฟล์เสียงสกุล .mp3 ที่เชื่อมกับ feature common_name และ Download_link



5. ขั้นตอนการทำ

5.1. จัดการชุดข้อมูล

งานหลักของขั้นตอนนี้คือดูรายละเอียดที่ชุดข้อมูลให้มา ทำให้พบว่าชุดข้อมูลมีไฟล์เสียงตามจำนวนตามที่ระบุมาไว้จริง และตรวจเช็คว่าค่า feature ที่ต้องการมี missing value หรือไม่

```
Percentage of Null Values:
common_name         0.000000
scientific_name     0.000000
recordist_name      0.000000
recording_length    0.000000
Date                0.000000
TYPE                1.434521
xc_id               0.000000
Time                0.000000
Country             0.000000
Download_link       0.000000
```

จะเห็นว่า feature ที่สำคัญคือ “common_name” (ชื่อนก) และ “Download_link” (ลิ้งค์ที่เชื่อมไปไฟล์เสียง) ไม่มีข้อมูลที่หายไป

จากนั้นจึงตัด feature ที่ไม่จำเป็นออกทั้งหมด และสร้าง feature เสริมชื่อ “label” เป็นการ encoding มาจาก “common_name”

ตัวอย่างการ encoding

```
species_encode = {species: i for i, species in
enumerate(df['common_name'].unique())}
int_to_species = {i: species for species, i in
species_encode.items()}
df['label'] = df['common_name'].map(species_encode)

df.head()
```

ตัวอย่างที่ได้

	common_name	Download_link	label
987	White-bellied Nothura	https://xeno-canto.org/34726/download	50
1189	Collared Brushturkey	https://xeno-canto.org/140728/download	63
129	Grey Tinamou	https://xeno-canto.org/557622/download	13
348	Cinereous Tinamou	https://xeno-canto.org/755297/download	22
355	Cinereous Tinamou	https://xeno-canto.org/683512/download	22

ทำการแบ่งชุดข้อมูลออกเป็น ดังนี้

- train dataset (70%)
- validation dataset (15%)
- test dataset (15%)

โดยก่อนแบ่งข้อมูลทุกครั้งจะทำการตรวจสอบว่ามีชนิดนกไหนที่มีจำนวนตัวอย่างน้อยกว่า 2 ตัวอย่างบ้าง เพื่อป้องกันปัญหาในการแบ่งข้อมูลแบบ stratified split โดยข้อมูลของชนิดนกที่มีตัวอย่างน้อยจะถูกกรองออกไปก่อน

ตัวอย่างโค้ด**บางส่วน**ในการกรองข้อมูลส่วนน้อยออกก่อน split

```
from sklearn.model_selection import train_test_split

# Identify classes with fewer than 2 samples to avoid errors during
stratified split
species_counts_full = df['label'].value_counts()
low_sample_species_full = species_counts_full[species_counts_full <
2].index.tolist()

# Remove rows belonging to low-sample species from the full dataset
df_filtered = df[~df['label'].isin(low_sample_species_full)].copy()
```

5.2. Audio to Mel Spectogram

ในขั้นตอนนี้คือการ preprocess ไฟล์เสียงก่อนเข้าฝึกฝนโมเดล โดยทำการแปลงไฟล์เสียงเป็นรูปภาพ Mel Spectrogram ด้วยไลบรารี่ `Librosa` เพราะโดยทั่วไปข้อมูลเสียงดิบจะไม่ถูกป้อนเข้าสู่โมเดล deep learning โดยตรง แต่จะถูกแปลงให้อยู่ในรูปแบบของ Spectrogram ก่อน

Spectrogram เป็น "ภาพถ่าย" โดยสรุปของคลื่นเสียง ซึ่งแสดงความถี่ (แกน Y) เทียบกับเวลา (แกน X) และใช้สีเพื่อระบุความเข้มหรือ Amplitude ของแต่ละความถี่

Mel Spectrogram เป็นการปรับปรุงที่สำคัญของ Spectrogram ทั่วไป โดยมีจุดประสงค์เพื่อให้การประมวลผลข้อมูลเสียงเป็นไปในลักษณะที่สอดคล้องกับการรับรู้ของมนุษย์ **[4]**

ดังนั้น โมเดล CNN ที่ใช้ Mel Spectrogram จึงสามารถเรียนรู้คุณลักษณะ (features) ที่มีความหมายและสำคัญต่อการแยกแยะเสียงได้ดีขึ้น การใช้ Mel Scale ช่วยให้โมเดลให้ความสนใจกับความแตกต่างของความถี่ที่มนุษย์อ่อนไหวมากกว่า

6510503522 ปุณยภา เจริญสุข

ตัวอย่างโค้ด**บางส่วน**ในการแปลง audio	

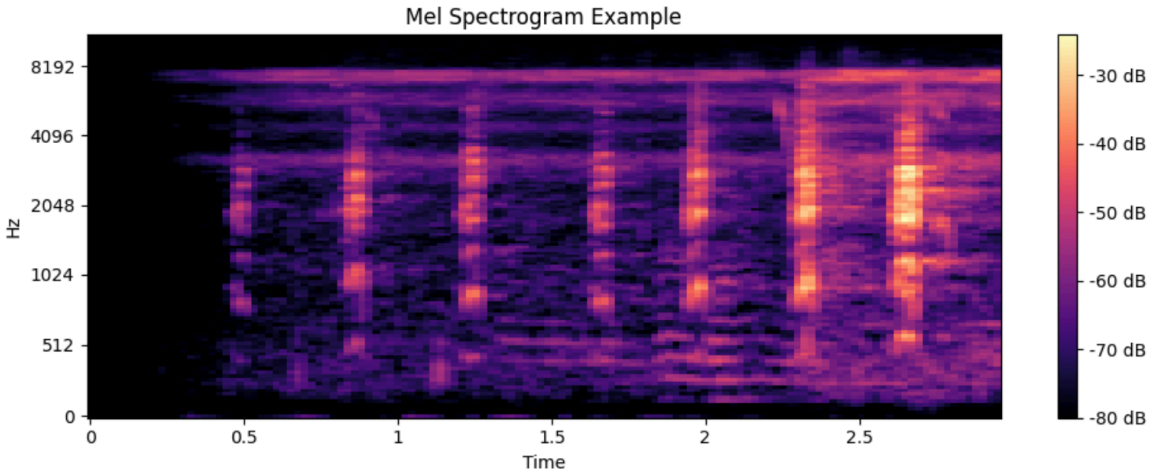
1. Audio → MelSpectrogram
 	
ส่วนที่แปลงสัญญาณเสียง (audio) เป็น Mel Spectrogram โดยใช้พารามิเตอร์ที่กำหนดไว้ (เช่น N_MELS=128, N_FFT=2048)
	`mel = librosa.feature.melspectrogram(`
	`y=audio, sr=self.sample_rate,`
	`n_fft=self.n_fft,`
	`hop_length=self.hop_length,`
	`n_mels=self.n_mels`
	`)`

2. Power → Decibel แปลงค่า Mel Spectrogram ที่อยู่ในหน่วยกำลัง (Power) ให้เป็นหน่วย เดซิเบล (dB) ซึ่งเป็นสเกลแบบลอการิทึมที่เลียนแบบการรับรู้ของหูมนุษย์และช่วยปรับปรุงประสิทธิภาพของ CNN	`mel_db = librosa.power_to_db(mel, ref=np.max)`

3. Pad/Trim Time Steps
 	
ปรับขนาดมิติของเวลา (T) ของ Mel Spectrogram ให้เป็นค่าคงที่ตามที่กำหนด (TIME_STEPS = 128) โดยการ ตัด (Trim) หรือ เติม (Pad) ด้วยค่าที่ต่ำที่สุด เพื่อให้ทุกตัวอย่างมีขนาดภาพที่เท่ากันสำหรับป้อนเข้า CNN
	`T = mel_db.shape[1]`
	`if T >= self.time_steps:`
	`mel_proc = mel_db[:,`
	`:self.time_steps]`
	`else:`
	`pad = self.time_steps - T`
	`mel_proc = np.pad(mel_db,`
	`((0,0),(0,pad)),`
	`mode='constant',`
	`constant_values=np.min(mel_db))`

4. MelSpectrogram → Tensor แปลง array ของ Mel Spectrogram ที่ผ่านการปรับขนาดแล้ว (mel_proc) ให้เป็น PyTorch Tensor ชนิด float และเพิ่มมิติที่ 0 (unsqueeze(0)) ซึ่งเป็นมิติของ ช่องสัญญาณ (Channel) (จาก (128, 128) เป็น (1, 128, 128)) เพื่อให้ตรงตาม Input Format ของ CNN	`mel_tensor =` `torch.from_numpy(mel_proc).float().unsqueeze(0)`

ตัวอย่าง Heatmap จาก Mel Spectrogram



ตัวอย่าง Tensor ที่จากการแปลง Mel Spectrogram

```
Shape of Mel Spectrogram batch: torch.Size([32, 1, 128, 128])
Shape of Labels batch: torch.Size([32])

Sample of Mel Spectrogram data (first sample, first channel, first 5x5 pixels):
[[-80. -80. -80. -80. -80.]
 [-80. -80. -80. -80. -80.]
 [-80. -80. -80. -80. -80.]
 [-80. -80. -80. -80. -80.]
 [-80. -80. -80. -80. -80.]]
```

5.3. สร้างโมเดล CNN

ลักษณะของสถาปัตยกรรมได้ระบุไปแล้วในข้อที่ 3

ตัวอย่างโค้ด<u>บางส่วน</u>ในการสร้างโมเดล CNN	

1. ชั้น Convolutional
 	
ทำหน้าที่สกัด Features ที่สำคัญจากภาพ Mel Spectrogram โดยการใช้ตัวกรองหลายชั้น เพื่อลดขนาดภาพและเพิ่มความลึกของช่องสัญญาณ (Channels)
	`# 1. Input (1, 128, 128) -> Output (32, 64, 64)`
	`self.layer1 = nn.Sequential(`
	`nn.Conv2d(1, 32, kernel_size=3, stride=1, padding=1),`
	`nn.ReLU(),`
	`nn.BatchNorm2d(32),`
	`nn.MaxPool2d(kernel_size=2, stride=2)`
	`)`
	
	`# 2. Input (32, 64, 64) -> Output (64, 32, 32)`
	`self.layer2 = nn.Sequential(`
	`nn.Conv2d(32, 64, kernel_size=3,`

	stride=1, padding=1), nn.ReLU(), nn.BatchNorm2d(64), nn.MaxPool2d(kernel_size=2, stride=2)) # 3. Input (64, 32, 32) -> Output (128, 16, 16) self.layer3 = nn.Sequential(. . . . .) # 4. Input (128, 16, 16) -> Output (256, 8, 8) self.layer4 = nn.Sequential(. . . . .)
2. ชั้น Fully Connected ทำหน้าที่จำแนกประเภท (Classifier) โดยรับคุณลักษณะที่สกัดออกมาแล้ว เพื่อคำนวณความน่าจะเป็นของแต่ละพันธุ์นก และให้ผลลัพธ์สุดท้ายเป็น num_classes	self.fc = nn.Sequential(nn.Dropout(0.5), nn.Linear(self.fc_input_size, 512), nn.ReLU(), nn.BatchNorm1d(512), # ใช้ BatchNorm1d ก่อน Dropout ใน FC Layer nn.Dropout(0.5), nn.Linear(512, num_classes))
3. โครงสร้างการไหลของข้อมูล (Forward Pass) กำหนดทิศทางการไหลของข้อมูล (Input x) ผ่านชั้นต่างๆของโมเดล ตั้งแต่ Layer 1 ไปจนถึงชั้น Fully Connected	def forward(self, x): out = self.layer1(x) out = self.layer2(out) out = self.layer3(out) out = self.layer4(out) out = out.reshape(out.size(0), -1) out = self.fc(out) return out

5.4. การฝึกฝน

การฝึกฝนจะใช้เพียง 20 epochs เพื่อความเหมาะสมของเวลาในการประมวลผล โดยโค้ดตัวอย่างต่อไปนี้แบ่งออกเป็น 3 ส่วนหลักตามลำดับการทำงานในแต่ละ epoch ดังนี้

<table>
<tr><td colspan="2">ตัวอย่างโค้ด<u>บางส่วน</u>ในฝึกฝนโมเดล</td></tr>
<tr><td>1. Training

แบ่งเป็น 4 ส่วน
- Forward Pass
ป้อน data เข้าสู่โมเดลเพื่อคำนวณ output และค่า Loss

- Zero Grad
ใช้ OPTIMIZER.zero_grad() เพื่อล้างค่า Gradients เก่าก่อนเริ่ม Batch ใหม่

- Backward Pass
ใช้ loss.backward() เพื่อคำนวณการไล่ระดับสี (Gradients)

- Optimization
ใช้ OPTIMIZER.step() เพื่อปรับน้ำหนัก (Weights) ของโมเดลให้ Loss ลดลง</td><td><pre>
# Lists สำหรับเก็บค่า Loss และ Accuracy เพื่อวาดกราฟ
train_losses = []
val_losses = []

NUM_EPOCHS = 20

# Training Loop
for epoch in range(NUM_EPOCHS):
    # --- Training Phase ---
    model.train()
    running_loss = 0.0
    running_accuracy = 0.0

    # ตัวแปรสำหรับแสดงตัวอย่าง
    example_printed = False

    for batch_idx, (data, labels) in
enumerate(tqdm(train_loader, desc=f"Epoch
{epoch+1} Training")):
        data = data.to(DEVICE)
        labels = labels.to(DEVICE)

        # Forward pass
        outputs = model(data)
        loss = CRITERION(outputs, labels)

        # Backward and optimize
        OPTIMIZER.zero_grad()
        loss.backward()
        OPTIMIZER.step()
        ...
</pre></td></tr>
<tr><td>2. Validation

- ช่วง Setup</td><td><pre>
model.eval()
    val_loss = 0.0
    val_accuracy = 0.0
</pre></td></tr>
</table>

`model.eval()` มีหน้าที่ตั้งค่าโมเดลเป็นโหมดประเมินผล (หยุดการทำงานของ `Dropout` และ `BatchNorm` เพื่อให้ผลลัพธ์คงที่) `torch.no_grad()` ช่วยปิดการคำนวณ Gradient เพื่อประหยัดหน่วยความจำและเวลา เนื่องจากเราไม่ต้องการปรับน้ำหนักโมเดลในขั้นตอนนี้
 	
- **ช่วง Execution**
การวัดประสิทธิภาพ: ทำการป้อนข้อมูล Validation (`val_loader`) เข้าสู่โมเดลเพื่อคำนวณค่า Val Loss และ Val Accuracy ค่าเหล่านี้จะใช้วัดว่าโมเดลที่ถูกฝึกไปแล้ว
	`with torch.no_grad():`
	`for data, labels in val_loader:`
	`data = data.to(DEVICE)`
	`labels = labels.to(DEVICE)`
	
	`outputs = model(data)`
	`loss = CRITERION(outputs, labels)`
	
	`val_loss += loss.item() * data.size(0)`
	`val_accuracy += calculate_accuracy(outputs, labels) * data.size(0)`
	`...`

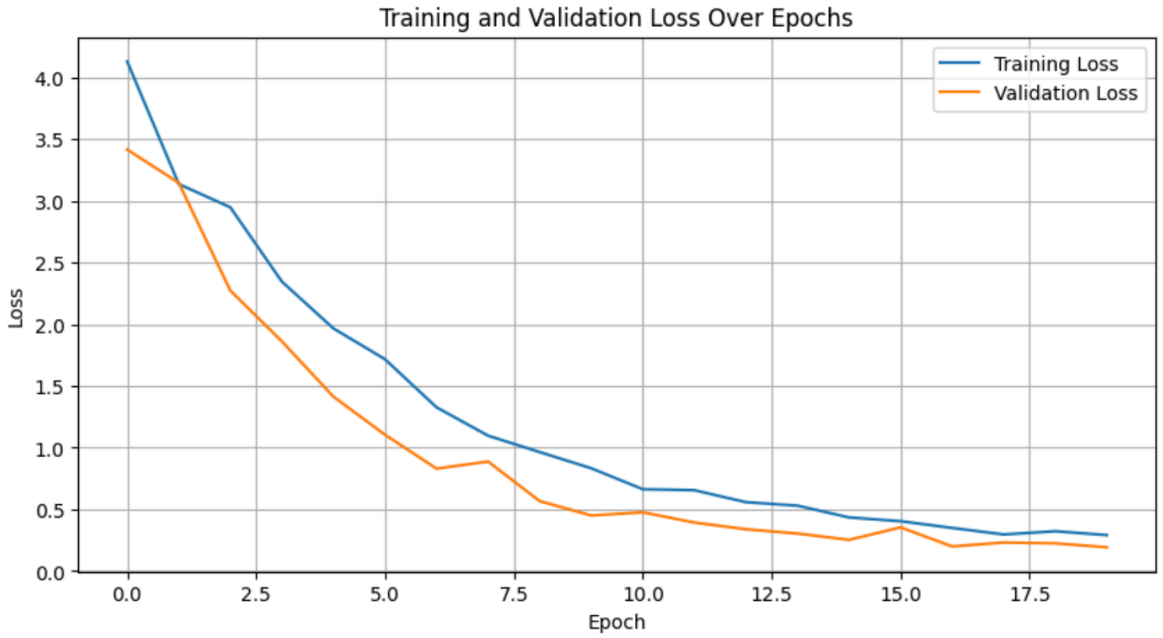
3. การบันทึกและแสดงผล
 	
คำนวณค่า Loss และ Accuracy เฉลี่ยต่อ Epoch ทั้งชุด Training และ Validation จากนั้นแสดงผลลัพธ์ และเก็บค่าไว้ใน `train_losses` และ `val_losses` เพื่อใช้วาดกราฟสรุปในภายหลัง
	`epoch_val_loss = val_loss / len(val_dataset)`
	`epoch_val_acc = val_accuracy / len(val_dataset)`
	`val_losses.append(epoch_val_loss)`
	
	`print(f"Epoch {epoch+1}/{NUM_EPOCHS}: Train Loss: {epoch_train_loss:.4f}, Train Acc: {epoch_train_acc:.4f}, Val Loss: {epoch_val_loss:.4f}, Val Acc: {epoch_val_acc:.4f}\n")`

5.5. การประเมินผล

การประเมินใช้ทั้ง **Accuracy, Precision, Recall, F1-score และทำการแสดงกราฟ train loss vs validation loss** เพราะการดูหลาย metrics ช่วยให้เราเข้าใจประสิทธิภาพของโมเดลได้รอบด้านมากขึ้นครับ แทนที่จะดูแค่ Accuracy เพียงอย่างเดียว

ซึ่งจะเห็นได้ว่าทุกค่ามีค่ามากกว่า 80 % ค่า test loss น้อย (0.2408) และค่า training loss และค่า validation loss ลู่เข้าสู่ 0 บ่งบอกว่าโมเดลมีประสิทธิภาพที่ดีและเรียนรู้ข้อมูลได้ค่อนข้างประสบความสำเร็จ

```
Final Test Loss: 0.2408
--- Evaluation Metrics ---
Accuracy: 95.30
Precision (Macro): 92.95
Recall (Macro): 92.45
F1 Score (Macro): 91.96
```



6. แหล่งอ้างอิง

[1] https://www.mmthailand.com/
[2] https://aws.amazon.com/th/compare/the-difference-between-ML-and-DL
[3] Sound Of 114 Species Of Birds Till 2022 : DATASET
[4] Audio Deep Learning Made Simple | Ketan Doshi Blog