Lab 6.2 กลุ่มบาบูนตึงตึง

1. โหลด MobileNet ที่ถูกเรียนรู้มาจากชุดข้อมูล ImageNet ไม่รวมชั้นสุดท้าย (include_top=False) และกำหนดรูปร่างของข้อมูลขนาด (224, 224, 3).

```
# Load base model
base_model = MobileNet(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

2. กำหนดตัวแปร `x` เพื่อเก็บผลลัพธ์จากชั้นเอาต์พุตของ `base_model`.

```
# Add new layers
x = base_model.output
```

3. เพิ่มชั้น `GlobalAveragePooling2D` เพื่อทำการประมาณค่าเฉลี่ยกับข้อมูลที่ออกจาก `base_model` ซึ่งจะลดขนาดข้อมูลให้เหลือเพียงค่าเดียวต่อหนึ่งข้อมูล.

```
x = GlobalAveragePooling2D()(x)  # Add GlobalAveragePooling2D layer
```

4. เพิ่มชั้น `Dense` ทั้งหมด 3 ชั้น โดยมีจำนวนโหนด 1024, 1024, และ 512 ตามลำดับ และใช้ activation function เป็น ReLU.

```
# Add Dense layers
x = Dense(1024, activation='relu')(x)  # Layer 1 with 1024 nodes and ReLU activation
x = Dense(1024, activation='relu')(x)  # Layer 2 with 1024 nodes and ReLU activation
x = Dense(512, activation='relu')(x)   # Layer 3 with 512 nodes and ReLU activation
```

5. เพิ่มชั้น `Dense` สุดท้ายที่มี 3 โหนดและใช้ activation function เป็น softmax เพื่อให้ผลลัพธ์เป็นความน่าจะเป็นสำหรับ 3 คลาส.

```
# Add final prediction layer
preds = Dense(3, activation='softmax')(x)  # Output layer with 3 nodes and softmax activation
```

6. สร้างแบบจำลอง `model` โดยกำหนดข้อมูลนำเข้าจาก `base_model` และผลลัพธ์จากชั้น `preds`.

```
# Create the model
model = Model(inputs=base_model.input, outputs=preds)
```

7. ส่วนถัดไปใช้สร้างการกำหนดค่าให้ชั้นแบบจำลอง โดยมีการแบ่งชั้นที่จะสามารถเรียนรู้ได้ (trainable) และไม่สามารถเรียนรู้ได้ (frozen) โดยใช้การกำหนดค่า `trainable` เป็น `True` หรือ `False` ขึ้นอยู่กับลำดับของชั้น.

```
# Assign Trainable layers and freeze layer -> ลองเปลี่ยน
for layer in model.layers[:86]:
    layer.trainable=False #Freeze base model
for layer in model.layers[86:]:
    layer.trainable=True #Unfreeze new added denses
```

8. Summary and compile the model: แสดงสรุปของโครงสร้างแบบจำลองและคอมไพล์โมเดลโดยกำหนด optimizer ใช้ Adam, loss function เป็น categorical cross-entropy สำหรับงานจำแนกหลายคลาส และ metric เป็นความแม่นยำ (accuracy).

```
#Summary and compile the model
model.summary()
model.compile(optimizer=Adam(lr=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
```

9. จัดการข้อมูลรูปภาพ และสร้าง ImageDataGenerator สำหรับข้อมูลที่จะใช้ในการฝึกและทดสอบ.

```
#Data preprocessing and image data gen
seed_value = 42
seed_val = 123
batch_size = 32

datagen = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.2,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    horizontal_flip=True,
    preprocessing_function=preprocess_input,
    fill_mode="nearest",
)
```

10. Create Train Image generator: สร้างตัว generator สำหรับข้อมูลฝึก (train) โดยระบุโฟลเดอร์เก็บรูปภาพ, ขนาดเป้าหมาย, รูปแบบสี, และข้อมูลเป้าหมายเป็นรูปแบบ one-hot encoding.

```
#Create Train Image generator
train_generator = datagen.flow_from_directory(
    './Train/',
    target_size=(224, 224),
    color_mode='rgb',
    batch_size=batch_size,
    class_mode='categorical',
    seed=seed_value,
    shuffle=True
)
```

11. สร้างตัว generator สำหรับข้อมูลทดสอบ (validation) โดยระบุโฟลเดอร์เก็บรูปภาพ, ขนาดเป้าหมาย, รูปแบบสี, และข้อมูลเป้าหมายเป็นรูปแบบ one-hot encoding.

```
#Create Validation Image generator
val_generator = datagen.flow_from_directory(
    './Validate/',
    target_size=(224, 224),
    color_mode='rgb',
    batch_size=batch_size,
    class_mode='categorical',
    seed=seed_val,
    shuffle=True
)
```

12. Display images: แสดงภาพที่ถูกสุ่มเลือกจากข้อมูลฝึกและข้อมูลทดสอบ เพื่อตรวจสอบว่าข้อมูลถูกนำเข้าอย่างถูกต้อง.

```
# Display 16 images in a 4x4 grid from train_generator
batch = train_generator.next()
train_images = batch[0]
train_images = (train_images + 1.0) / 2.0

batch = val_generator.next()
val_images = batch[0]
val_images = (val_images + 1.0) / 2.0

# Create a 4x4 grid for image display
plt.figure(figsize=(10, 8))
for i in range(16):
    plt.subplot(4, 4, i + 1)
    plt.imshow(train_images[i])
    # plt.axis('off')  # Turn off axis labels
plt.show()

plt.figure(figsize=(10, 8))
for i in range(16):
    plt.subplot(4, 4, i + 1)
    plt.imshow(val_images[i])
    # plt.axis('off')  # Turn off axis labels
plt.show()
```

13. Create optimizer: สร้าง optimizer สำหรับการฝึกโมเดลโดยใช้ Adam optimizer และกำหนดอัตราการเรียนรู้ (learning rate).

```
#Create Optimizer
opts = Adam(learning_rate=0.0001)
model.compile(loss='categorical_crossentropy', optimizer=opts, metrics=['accuracy'])
eps = 2
step_size_train = train_generator.n // train_generator.batch_size
step_size_val = val_generator.n // val_generator.batch_size
```

14. Training: ฝึกโมเดลโดยใช้ generator สำหรับข้อมูลฝึกและข้อมูลทดสอบ โดยกำหนดจำนวนรอบการฝึก (epochs) และบันทึกประวัติการฝึก.

```
#Check step size train = step size val
if step_size_train != step_size_val:
    print("Warning: step_size_train is not equal to step_size_val.")
    new_batch_size = val_generator.n // step_size_train
    val_generator = datagen.flow_from_directory(
        './Validate/',
        target_size=(224, 224),
        color_mode='rgb',
        batch_size=new_batch_size,
        class_mode='categorical',
        seed=seed_val,
        shuffle=True
    )
    step_size_val = val_generator.n // val_generator.batch_size
    print(f"Adjusted batch size to {new_batch_size} to make step_size_train equal to step_size_val.")

eps = 100
history = model.fit_generator(
    generator=train_generator,
    steps_per_epoch=step_size_train,
    validation_data=val_generator,
    validation_steps=step_size_val,
    epochs=eps,
    verbose=1)

#Performance Visualization
epochs = range(1, 100 + 1)
```

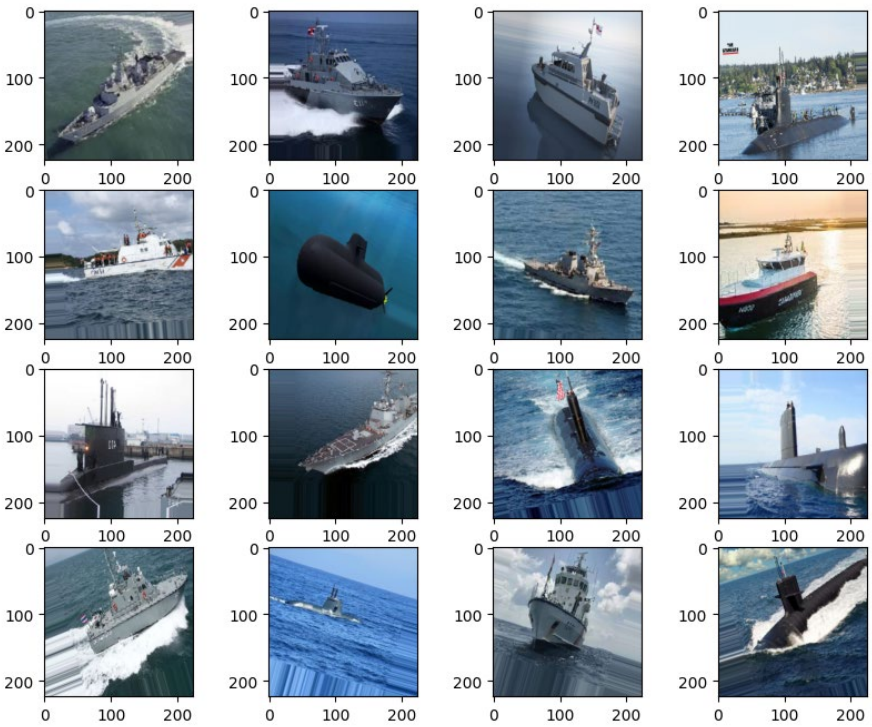
15. Performance Visualization: แสดงกราฟเพื่อติดตามประสิทธิภาพของโมเดล รวมถึงความแม่นยำของการฝึกและการทดสอบ และค่าขาดหาย (loss) ของการฝึกและการทดสอบ.

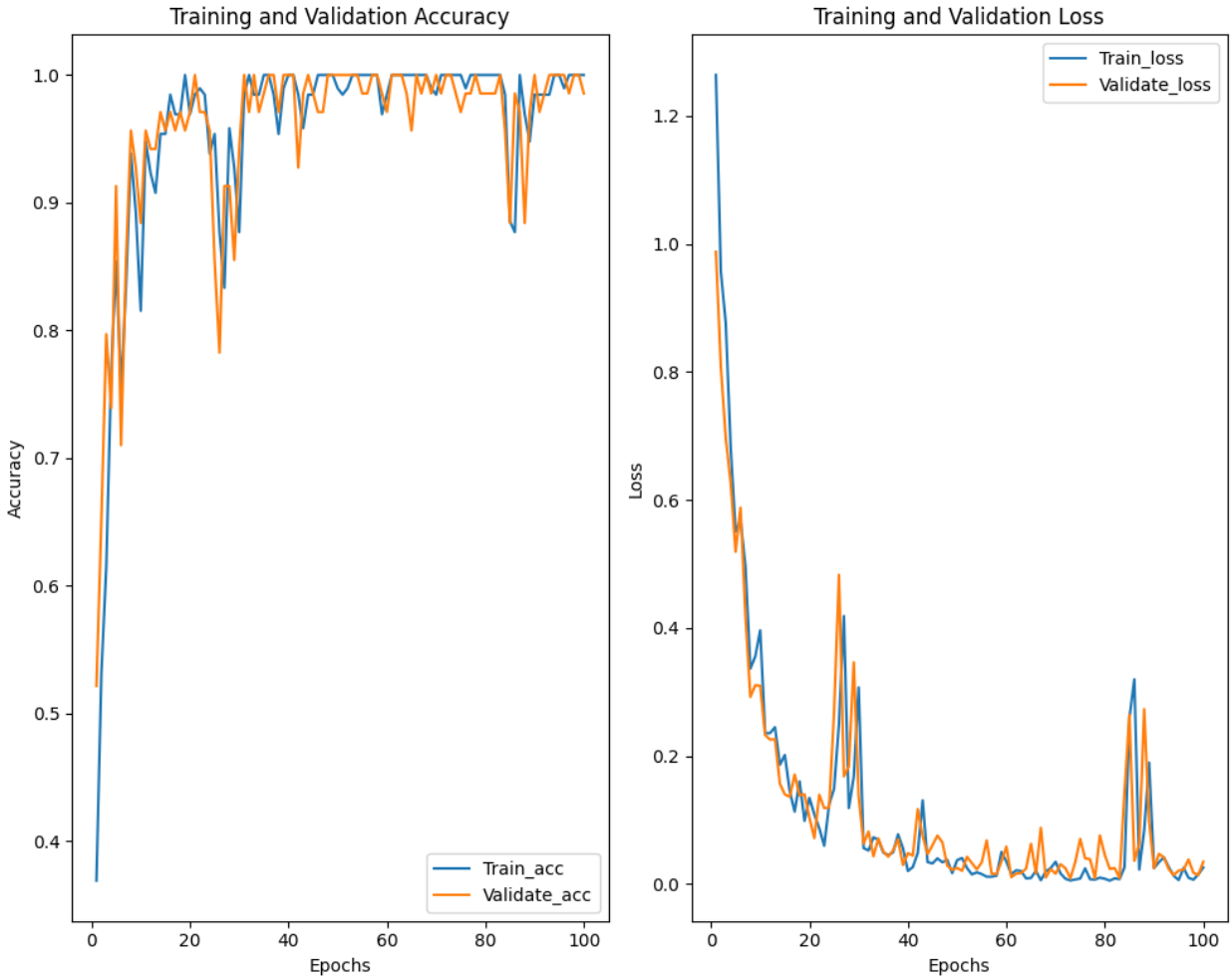
```python
#View Accuracy (Training, Validation)
plt.figure(figsize=(10, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs, history.history["accuracy"], label="Train_acc")
plt.plot(epochs, history.history["val_accuracy"], label="Validate_acc")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.title("Training and Validation Accuracy")
plt.legend()

#View Loss (Training, Validation)
plt.subplot(1, 2, 2)
plt.plot(epochs, history.history['loss'], label="Train_loss")
plt.plot(epochs, history.history['val_loss'], label="Validate_loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Training and Validation Loss")
plt.legend()

plt.tight_layout()
plt.show()
```

ผลลัพธ์


Training and Validation Accuracy
Accuracy
Epochs
Train_acc
Validate_acc
Training and Validation Loss
Loss
Epochs
Train_loss
Validate_loss