遞迴神經網路與變形器

作業四

學號：313831025

姓名：俞博云

一、 題目敘述

本作業旨在實作並比較 Vision Transformer (ViT) 與 SWIN Transformer 兩種深度學習模型於 CIFAR-10 圖片分類任務中的表現。透過載入預訓練模型並對其進行全參數微調（fine-tuning），本作業將針對兩種模型在測試集上的分類性能進行評估與比較。除此之外，亦使用 Grad-CAM 視覺化兩個模型在進行分類判斷時所關注的圖片區域，以分析其決策過程中的注意力分佈差異。

二、 數據預處理

本作業選用 CIFAR-10 資料集作為分類任務的測試標的。該資料集包含 10 個不同的類別（如飛機、汽車、貓、狗等），共計 60,000 張彩色影像，其中 50,000 張作為訓練集，10,000 張作為測試集，且每張影像的原始尺寸為 32x32 像素。由於 ViT 以及 SWIN Transformer 模型皆為在 ImageNet 上預訓練，這兩個模型預期輸入尺寸為 224x224。因此先進行以下前處理步驟。以下分別進行介紹：

2.1 數據讀取

首先透過 PyTorch 的 torchvision.datasets 與 Dataloader 建立訓練資料及以及資料輸入前處理流程，程式碼如圖 2.1 所示，trainset 與 testset 分別對應 CIFAR-10 的訓練集與測試集，並透過 transform_train 與 transform_test 套用不同的前處理策略。

```
# Load CIFAR-10 dataset
trainset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform_train)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=128, shuffle=True, num_workers=2)

testset = torchvision.datasets.CIFAR10(root='./data', train=False, download=True, transform=transform_test)
testloader = torch.utils.data.DataLoader(testset, batch_size=128, shuffle=False, num_workers=2)
```

圖 2.1、數據讀取程式碼

此外，DataLoader 則用於將資料分成小 batch 並支援多執行緒載入，在訓練集中設定為 shuffle=True 確保資料隨機打亂，而測試集為持固定的資料順序進行評估。

2.2 圖片前處理流程

為了使 CIFAR-10 資料集能夠適配預訓練的 ViT 以及 SWIN Transformer 模型，首先定義每個通道的平均值與標準差進行正規化處理，這能使影像資料在經過轉換後，其像素值分佈更接近於模型預訓練時所使用的 ImageNet 資料，設定

如圖 2.2 所示。

```
mean = (0.4914, 0.4822, 0.4465)
std = (0.2023, 0.1994, 0.2010)
size = (224, 224)
```

圖 2.2、正規化參數設定

此外，為了符合 ViT 與 SWIN Transformer 模型預設的輸入尺寸，我們將原始 32x32 的圖片均調整為 224x224，訓練以及測試資料集轉換程式碼如圖 2.3 所示。

```
transform_train = transforms.Compose([
    transforms.RandomCrop(32, padding=4),
    transforms.Resize(size),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize(mean, std),
])

transform_test = transforms.Compose([
    transforms.Resize(size),
    transforms.ToTensor(),
    transforms.Normalize(mean, std),
])
```

圖 2.3、訓練以及測試資料集轉換程式碼

訓練資料集前處理流程，首先對原始圖片進行裁切，透過在影像邊緣加上 4 像素的填補後再裁切回 32x32 大小，這樣可以增加圖像的變化性，提升資料多樣性。接者將圖片縮放至 224x224，以符合 Vision Transformer 與 SWIN 模型預期的輸入尺寸，並施加隨機水平翻轉（RandomHorizontalFlip()），以 50% 的機率將圖像左右對調，並透過 ToTensor() 將影像轉換為 PyTorch 所需的 Tensor 格式，同時將像素值從 $[0, 255]$ 範圍正規化至 $[0, 1]$。最後，使用 CIFAR-10 的通道平均值與標準差進行標準化（Normalize(mean, std)），使圖像輸入特性與預訓練模型一致。

而在測試資料的處理較為簡化，僅保留必要步驟以維持結果的穩定性，同樣將測試圖像縮放至 224x224，轉換為 Tensor，並使用與訓練集相同的平均值與標準差進行正規化。

三、 建立模型及訓練細節

在本次作業中，使用 timm 套件來載入預訓練模型，分別選用 vit_base_patch16_224 與 swin_base_patch4_window7_224 作為主體模型架構，並透過 pretrained=True 參數載入其在 ImageNet 上的預訓練權重，以加速訓練並提升模型收斂效果，此外，我們實驗了兩種不同的初始學習率設定，並比較兩者在訓練初期的收斂速度與最終測試準確率之差異。分別介紹如下：

3.1 建立模型

以兩個小節介紹 ViT 以及 SWIN Transformer 的模型設定，分別介紹如下：

3.1.1 ViT 模型建立

首先使用 timm.create_model 函數載入預訓練的 ViT 模型，並且指定模型架構為"vit_base_patch16_224"， 這個模型在設計上將輸入影像劃分為 16x16 的 patch，再將其展平成序列輸入至 Transformer 架構中進行全局特徵學習，程式碼如圖 2.4 所示。

```
# Load pre-trained models
vit_model = timm.create_model('vit_base_patch16_224', pretrained=True)

# Modify classification heads
# For ViT
vit_model.head = nn.Linear(vit_model.head.in_features, 10)
```

圖 2.4、ViT 模型建立程式碼

由於原始 ViT 模型是針對 ImageNet 的 1000 類別分類任務所設計，其分類頭（head）為一個輸出維度為 1000 的線性層。為了適應 CIFAR-10 資料集，我們將模型的 head 層替換為新的線性層，將其輸出維度改為 10，使其能對應本任務中的 10 個類別。這樣的修改確保了預訓練模型的主體架構與特徵抽取能力得以保留，同時也使最終分類任務符合新的資料分佈需求。

```
(head_drop): Dropout(p=0.0, inplace=False)
(head): Linear(in_features=768, out_features=10, bias=True)
```

圖 2.5、ViT 模型 Head 架構示意圖

3.1.2 SWIN Transformer 模型建立

首先使用 timm.create_model 函數載入預訓練的 SWIN Transformer 模型，並且指定模型架構為"swin_base_patch4_window7_224"，這個模型表示輸入影像會先被劃分為 4×4 大小的 patch，而每個 attention block 中使用的局部注意力視窗大小為 7×7。與 ViT 採用全局自注意力不同，SWIN Transformer 採取區域性 self-attention（Window-based Multi-head Self-Attention, W-MSA），每次僅在 7×7 區域內進行注意力運算，大幅降低計算成本，程式碼如圖 2.6 所示。

```
# Load pre-trained models
swin_model = timm.create_model('swin_base_patch4_window7_224', pretrained=True, num_classes=10)
```

圖 2.6、SWIN Transformer 模型建立程式碼

不同於 ViT 模型需要手動替換分類頭，我們在建立 SWIN Transformer 模型時，直接透過 num_classes=10 參數指定輸出類別數為 10，使模型初始化時即自動將原始的分類層調整為適用於 CIFAR-10 的結構，如圖 2.7 所示。

```
(head): ClassifierHead(
  (global_pool): SelectAdaptivePool2d(pool_type=avg, flatten=Identity())
  (drop): Dropout(p=0.0, inplace=False)
  (fc): Linear(in_features=1024, out_features=10, bias=True)
  (flatten): Identity()
)
```

圖 2.7、SWIN Transformer 模型 Head 架構示意圖

3.2 訓練主程式

本作業中所建立的模型透過"train_model"函式進行訓練與驗證，訓練過程分為若干個 epoch，給個 epoch 均包含一個完整的訓練與驗證流程。在每一輪訓練中，模型會進入訓練模式，並透過 mini-batch 資料不斷更新權重參數以最小化損失函數。除此之外，還引入了自動混合精度訓練（Automatic Mixed Precision, AMP），藉由此方法加速計算並節省 GPU Memory，程式碼如所示。

```
scaler = torch.amp.GradScaler(device='cuda', enabled=True)
```

```
with autocast(device_type='cuda', enabled=True):
    outputs = model(inputs)
    loss = criterion(outputs, labels)

scaler.scale(loss).backward()
scaler.step(optimizer)
scaler.update()
```

圖 2.8、自動混合精度訓練程式碼

Loss 搭配 Optimizer 進行參數更新，同時加入 CosineAnnealingLR 學習率排程器，讓學習率在每個 epoch 後逐步下降，提升模型穩定性與最終準確率，並且每次訓練迴圈中均會記錄當前的訓練損失與準確率。程式碼如圖 2.9 所示。

```
scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(optimizer, num_epochs)
```

```
    running_loss += loss.item()
    _, predicted = torch.max(outputs, 1)
    correct += (predicted == labels).sum().item()
    total += labels.size(0)

scheduler.step()

avg_train_loss = running_loss / len(trainloader)
train_acc = correct / total * 100
train_losses.append(avg_train_loss)
train_accuracies.append(train_acc)

print(f"Epoch {epoch+1}, LR: {current_lr:.6e}, Train Loss: {avg_train_loss:.4f}, Train Acc: {train_acc:.2f}")
```

圖 2.9、訓練過程程式碼

當跑完後會進行驗證流程，透過 torch.no_grad()停用梯度計算，避免在測試階段更新模型權重，若模型於測試集上的表現優於歷史最佳準確率，則會自動儲存目前模型參數，確保最終可取得效能最佳的模型版本。此外，整個訓練流程中也會記錄每個 epoch 的學習率、損失與準確率，以便分析結果，程式碼如圖 2.10 所示。

```
# Evaluate
eval_loss, eval_acc = evaluate_model(model, testloader, criterion)
eval_losses.append(eval_loss)
eval_accuracies.append(eval_acc)

if eval_acc > best_acc:
    best_acc = eval_acc
    torch.save(model.state_dict(), "./swin/best_swin_model.pth")
    print("🍓 Best model updated.")
```

```
def evaluate_model(model, testloader, criterion):
    model.eval()
    total = 0
    correct = 0
    running_loss = 0.0

    with torch.no_grad():
        for inputs, labels in tqdm(testloader, desc="Evaluating"):
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            running_loss += loss.item()

            _, predicted = torch.max(outputs, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    avg_loss = running_loss / len(testloader)
    accuracy = 100 * correct / total
    print(f"Eval Loss: {avg_loss:.4f}, Accuracy: {accuracy:.2f}%")
    return avg_loss, accuracy
```

圖 2.10、驗證過程程式碼

3.3 模型參數設定

本實驗中，設計了兩組不同的訓練參數組合，以比較學習率（Learning Rate）與訓練輪數（Epoch）對模型訓練結果的影響。如表 3.1 所示，第一組設定採用學習率為 5e-5，訓練 10 個 epoch；第二組則採用較高的學習率 7e-4，訓練 200

個 epoch。其他超參數如 batch size 均保持一致。透過此比較，觀察模型在不同訓練策略下的收斂速度與最終性能差異，作為後續分析的重要依據。

表 3.1、超參數設定表

超參數名稱	實驗一	實驗二
Learning rate	5e-5	7e-4
Train Batch size	128	128
Val Batch size	128	128
Epoch	10	200

```
# Define loss function and optimizers
criterion = nn.CrossEntropyLoss()
swin_optimizer = optim.AdamW(swin_model.parameters(), lr=5e-5) # 7e-4, 5e-5
```

圖 3.1、Loss 及 Optimizer 選擇

此外在訓練設定中，如圖 3.1 所示，在 Optimizer 優化器使用 AdamW Optimizer，其自適應學習率調整機制有助於加快收斂並提高訓練效率，這是目前在 Transformer 架構中最常見的選擇之一。AdamW 在 Adam 的基礎上加入了對權重衰減（weight decay）的正歸化處理，有助於減少過擬合現象，提升模型的泛化能力。損失函數方面，採用交叉熵損失（CrossEntropyLoss），這是多類別分類問題中最常用且穩定的損失函數。它透過最大化預測機率與實際標籤分佈的一致性，驅動模型輸出正確分類結果，是圖像分類任務中的標準選擇。

四、 實驗介紹與結果

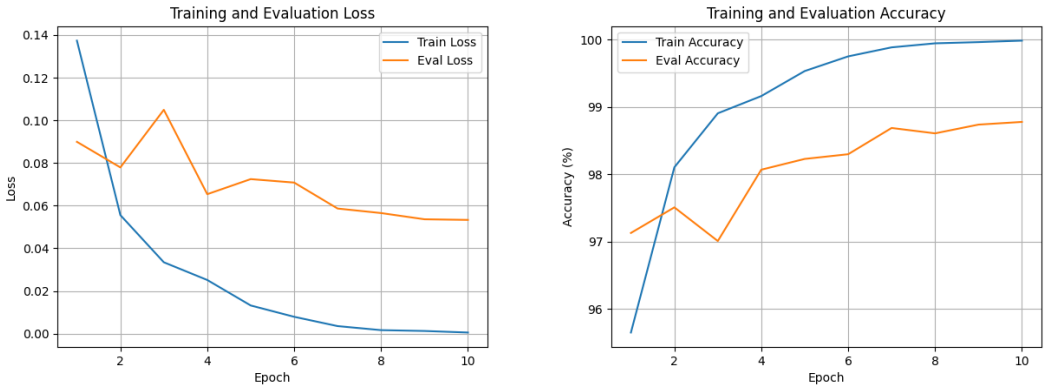
4.1 實驗介紹

本章將呈現 ViT 與 SWIN Transformer 模型在 CIFAR-10 分類任務中的訓練與測試結果，並針對不同學習率設定進行性能比較。實驗結果將依據準確率、損失值、混淆矩陣與 Grad-CAM 可視化進行全面分析，以評估兩種模型在分類效能與特徵關注上的差異。。

4.2 實驗結果

4.2.1 ViT 模型

首先在學習率為 5e-5 的設定下，僅對 ViT 模型進行 10 個 epoch 的 training，其訓練及驗證表現如圖 4.1 所示。從圖 4.1(a)可以觀察到，隨著訓練進行，訓練損失持續下降，顯示模型成功學習資料特徵；而驗證損失則在前幾個 epoch 呈現波動，之後趨於穩定，代表模型逐步收斂。圖 4.1 (b) 顯示訓練準確率快速上升至接近 100%，而驗證準確率則逐漸提高，最終穩定落在約 99% 左右。



(a) Loss (b) Accuracy

圖 4.1、ViT 模型（learning rate = 5e-5）的 Loss 與 Accuracy 示意圖

然而，從訓練準確率與驗證準確率之間的差距來看，模型在後期可能已開始出現輕微的過擬合現象。儘管整體驗證表現仍然穩定，但訓練準確率持續上升而驗證準確率未同步提升，說明模型在訓練集上表現極佳，但在看不見的新資料上則略有泛化能力下降的跡象。但綜合整體表現，在此設定下，ViT 模型於 CIFAR-10 測試集達到 **98.78%**的分類準確率，表示對於未知資料集具有一定的辨識能力。

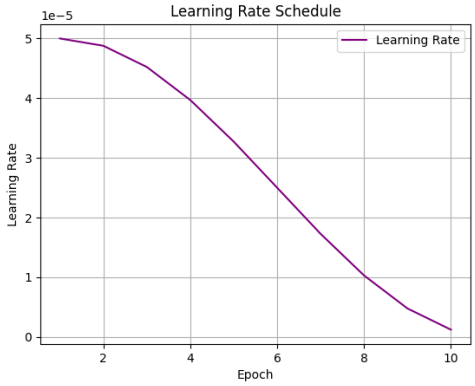


圖 4.2、學習率調整曲線(ViT, lr=5e-5)

學習率方面，圖 4.2 展示了訓練過程中採用 Cosine Annealing 調度策略的學習率變化情形。學習率從初始值 5e-5 開始，隨著 epoch 增加而逐漸下降，幫助模型在初期快速探索參數空間，後期則穩定收斂至較佳的性能表現。這種策略有助於減少過度震盪的風險，同時提升訓練效率與最終準確率。

表 4.1、ViT 模型(lr=5e-5)於 CIFAR-10 測試集之整體效能指標

指標名稱	**數值**
Accuracy	98.78 %
Top-1 錯誤率	1.22 %
模型參數總量	85.81 M
FLOPs	16.86 GFLOPs
平均推論時間	8.32 ms/image
GPU 記憶體使用量	1699.35 MB

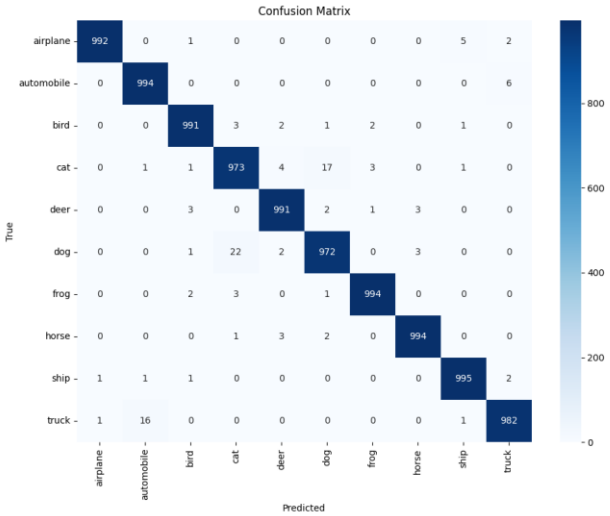


圖 4.3、ViT 模型 (lr=5e-5) 於 CIFAR-10 測試集之 Confusion Matrix

在表 4.1 可看出 ViT 模型在 CIFAR-10 測試集上的分類準確率達到 **98.78%**，展現出極佳的辨識能力與學習效果。模型總參數量為 **85.81M**，在保持高度準確率的同時，仍具備良好的運算效率。其計算複雜度為 **16.86 GFLOPs**，推論時間平均僅需 **8.32 毫秒/張圖像**，最大 GPU 記憶體使用量為 **1699.35 MB**，顯示此模型可在常見的 GPU 環境中穩定運作。

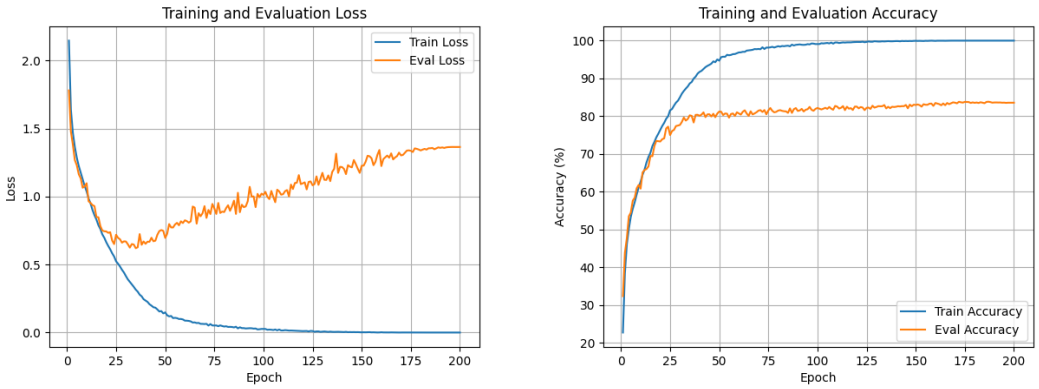
而在混淆矩陣圖 4.3 可看出，其大多數類別結果與實際標籤高度一致。最大值出現在"ship -> ship"，相對的最小值則多出現在"cat"與"dog"之間的混淆。這兩類動物圖像在 CIFAR-10 中常因姿勢、背景與色彩分布相似而造成模型誤判。例如，有部分"cat"圖像被誤分類為"dog"，顯示模型在區分這兩類語意相近的類別時仍有一定挑戰，但從整體來看，準確率極高。

	precision	recall	f1-score	support
airplane	1.00	0.99	0.99	1000
automobile	0.98	0.99	0.99	1000
bird	0.99	0.99	0.99	1000
cat	0.97	0.97	0.97	1000
deer	0.99	0.99	0.99	1000
dog	0.98	0.97	0.97	1000
frog	0.99	0.99	0.99	1000
horse	0.99	0.99	0.99	1000
ship	0.99	0.99	0.99	1000
truck	0.99	0.98	0.99	1000
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

圖 4.4、ViT 模型 (lr=5e-5) 於 CIFAR-10 測試集之分類指標

而在分類指標如圖 4.4 所示，模型在所有類別的三項指標表現穩定，平均值均為 0.99，顯示模型不僅準確，亦具備良好的類別平衡能力，在"airplane"類別的 Precision 高達 1.00，相對的"cat"與"dog"這兩個語意相近、圖像特徵接近的類別，其 Precision 與 Recall 為 **0.97** 左右，略低於其他類別，反映出模型在區分這類相似物體時仍有待加強。

接者在學習率為 7e-7 的設定下，對 ViT 模型進行 200 個 epoch 的 training，其訓練及驗證表現如圖 4.5 所示。



(a) Loss (b) Accuracy

圖 4.5、ViT 模型（learning rate = 7e-7）的 Loss 與 Accuracy 示意圖

從圖 4.5 (a)可以觀察到，隨著訓練進行，訓練損失隨著 epoch 數增加迅速下降，並在中後期趨近於 0，顯示模型對訓練資料幾乎已完全擬合。然而，驗證損失在初期雖略有下降，但隨後逐漸上升並持續波動，顯示模型在測試資料上的表現逐漸劣化，完全是過擬合的現象。同樣，從圖 4.5 (b) 中也可以觀察到相似的趨勢。訓練準確率迅速上升並接近 100%，而驗證準確率則在大約 80% 附近停滯並略有波動，最佳數值僅有 **83.84%**，與訓練準確率出現明顯落差。這表明在此設定下模型雖能充分擬合訓練資料，卻未能有效泛化至測試資料。

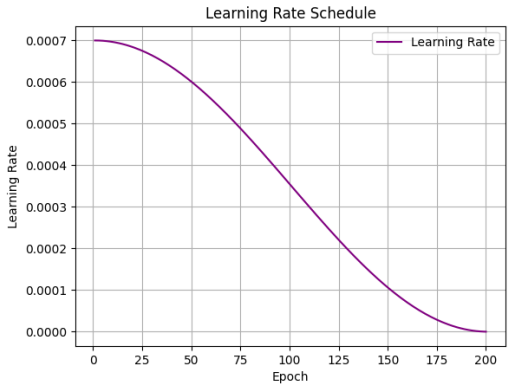


圖 4.6、學習率調整曲線(ViT, lr=7e-4)

學習率方面，圖 4.6 展示了訓練過程中採用 Cosine Annealing 調度策略的學習率變化情形。學習率從初始值 7e-4 開始，隨著 epoch 增加而逐漸下降，幫助模型在初期快速探索參數空間，後期則穩定收斂至較佳的性能表現。這種策略有助於減少過度震盪的風險，同時提升訓練效率與最終準確率。

表 4.2、ViT 模型(lr=7e-4)於 CIFAR-10 測試集之整體效能指標

指標名稱	**數值**
Accuracy	83.84 %
Top-1 錯誤率	16.16 %
模型參數總量	85.81 M
FLOPs	16.86 GFLOPs
平均推論時間	10.68 ms/image
GPU 記憶體使用量	2698.10 MB

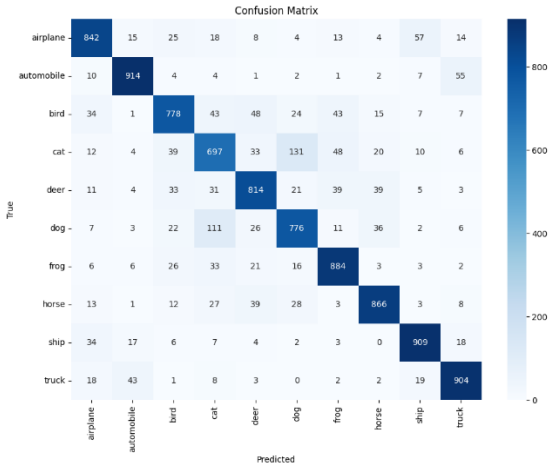


圖 4.7、ViT 模型 (lr=5e-5) 於 CIFAR-10 測試集之 Confusion Matrix

在表 4.1 可看出 ViT 模型在 CIFAR-10 測試集上的分類準確率僅達 **83.84%**，展現出極佳的辨識能力與學習效果。模型總參數量為 **85.81M**，在保持高度準確率的同時，仍具備良好的運算效率。其計算複雜度為 **16.86 GFLOPs**，推論時間平均僅需 **8.32 毫秒/張圖像**，最大 GPU 記憶體使用量為 **1699.35 MB**，顯示此模型可在常見的 GPU 環境中穩定運作。

而在混淆矩陣圖 4.7 可看出，其大多數類別結果與實際標籤高度一致。最大值出現在"automobile -> automobile"，相對的最小值則多出現在"cat"與"dog"之間的混淆。這兩類動物圖像在 CIFAR-10 中常因姿勢、背景與色彩分布相似而造成模型誤判。例如，有部分"cat"圖像被誤分類為"dog"，顯示模型在區分這兩類語意相近的類別時仍有一定挑戰，與 Lr=5e-5 的結果相似。

	precision	recall	f1-score	support
airplane	0.85	0.84	0.85	1000
automobile	0.91	0.91	0.91	1000
bird	0.82	0.78	0.80	1000
cat	0.71	0.70	0.70	1000
deer	0.82	0.81	0.82	1000
dog	0.77	0.78	0.77	1000
frog	0.84	0.88	0.86	1000
horse	0.88	0.87	0.87	1000
ship	0.89	0.91	0.90	1000
truck	0.88	0.90	0.89	1000
accuracy			0.84	10000
macro avg	0.84	0.84	0.84	10000
weighted avg	0.84	0.84	0.84	10000

圖 4.8、ViT 模型 (lr=7e-4) 於 CIFAR-10 測試集之分類指標

而在分類指標如圖 4.8 所示，ViT 模型在學習率為 7e-4 的設定下於 CIFAR-10 測試集上達到 整體準確率 84%，而 macro average 與 weighted average 的 precision、recall 與 f1-score 均為 0.84，顯示模型具備一定的分類能力，但整體表現仍有提升空間。在各類別細項表現上，automobile、ship 與 truck 等類別的 precision 與 recall 均高於 0.90，顯示模型對於結構明確、特徵穩定的目標具有良好的判別能力。相對地，模型在"cat"與"dog"這兩個語意相近、圖像特徵容易混淆的類別上表現較差，是整體分類效能中相對的弱點，符合上述內文所說。

4.2.2 SWIN Transformer 模型

首先在學習率為 5e-5 的設定下，僅對 SWIN Transformer 模型進行 10 個 epoch 的 training，其訓練及驗證表現如圖 4.9 所示。從圖 4.9 (a)可以觀察到，隨著訓練進行，訓練損失持續下降，顯示模型成功學習資料特徵；而驗證損失則在前幾個 epoch 呈現波動，之後趨於穩定，代表模型逐步收斂。圖 4.9 (b) 顯示訓練準確率快速上升至接近 100%，而驗證準確率則逐漸提高，最終穩定落在約 99% 左右。

(a) Loss　　(b) Accuracy

圖 4.9、SWIN 模型（learning rate = 5e-5）的 Loss 與 Accuracy 示意圖

然而，從訓練準確率與驗證準確率之間的差距來看，模型在後期可能已開始出現輕微的過擬合現象。儘管整體驗證表現仍然穩定，但訓練準確率持續上升而驗證準確率未同步提升，說明模型在訓練集上表現極佳，但在看不見的新資料上則略有泛化能力下降的跡象。但綜合整體表現，在此設定下，SWIN Transformer 模型於 CIFAR-10 測試集達到 **98.93%**的分類準確率，表示對於未知資料集具有一定的辨識能力。

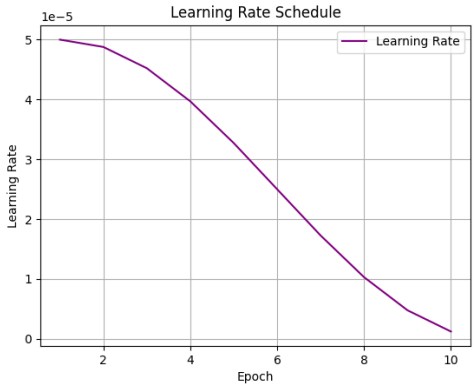


圖 4.10、學習率調整曲線(SWIN, lr=5e-5)

學習率方面，圖 4.10 展示了訓練過程中採用 Cosine Annealing 調度策略的學習率變化情形。學習率從初始值 5e-5 開始，隨著 epoch 增加而逐漸下降，幫助模型在初期快速探索參數空間，後期則穩定收斂至較佳的性能表現。這種策略有助於減少過度震盪的風險，同時提升訓練效率與最終準確率。

表 4.3、SWIN 模型(lr=5e-5)於 CIFAR-10 測試集之整體效能指標

指標名稱	**數值**
Accuracy	98.93 %
Top-1 錯誤率	1.07 %
模型參數總量	86.75 M
FLOPs	15.17 GFLOPs
平均推論時間	30.79 ms/image
GPU 記憶體使用量	4301.61 MB

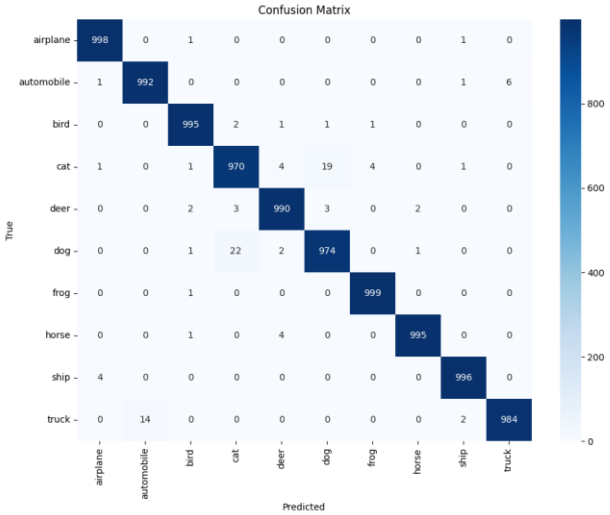


圖 4.11、SWIN 模型 (lr=5e-5) 於 CIFAR-10 測試集之 Confusion Matrix

在表 4.3 可看出 SWIN Transformer 模型在 CIFAR-10 測試集上的分類準確率達到 **98.93%**，展現出極佳的辨識能力與學習效果。模型總參數量為 **86.75M**，在保持高度準確率的同時，仍具備良好的運算效率。其計算複雜度為 **15.17 GFLOPs**，推論時間平均需 **30.79 毫秒/張圖像**，最大 GPU 記憶體使用量為 **4301.61 MB**，顯示此模型可在常見的 GPU 環境中穩定運作。

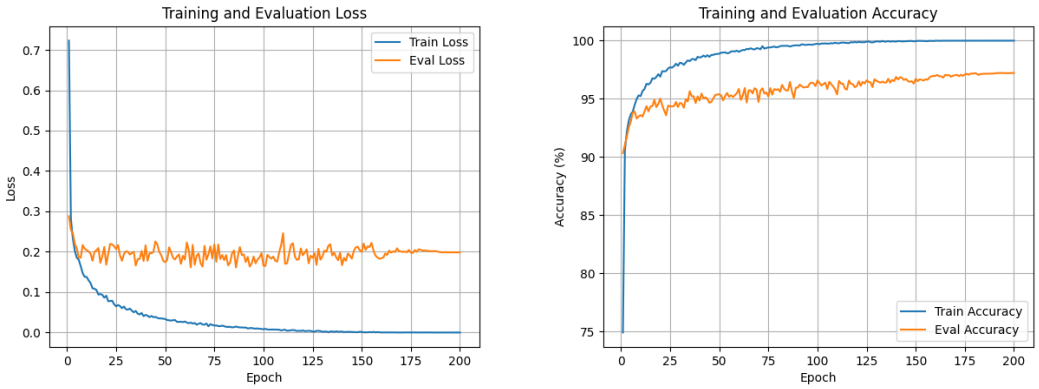
而在混淆矩陣圖 4.11 可看出，其大多數類別結果與實際標籤高度一致。最大值出現在"frog -> frog"，相對的最小值與 ViT 模型相同，多出現在"cat"與"dog"之間的混淆。顯示其模型一樣在區分這兩類語意相近的類別仍有一定的困難。

	precision	recall	f1-score	support
airplane	0.99	1.00	1.00	1000
automobile	0.99	0.99	0.99	1000
bird	0.99	0.99	0.99	1000
cat	0.97	0.97	0.97	1000
deer	0.99	0.99	0.99	1000
dog	0.98	0.97	0.98	1000
frog	1.00	1.00	1.00	1000
horse	1.00	0.99	1.00	1000
ship	1.00	1.00	1.00	1000
truck	0.99	0.98	0.99	1000
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

圖 4.12、SWIN 模型 (lr=5e-5) 於 CIFAR-10 測試集之分類指標

而在分類指標如圖 4.12 所示，模型在所有類別的三項指標表現穩定，平均值均為 0.99，顯示模型不僅準確，亦具備良好的類別平衡能力。從類別細項來看，多數類別如 airplane、automobile、frog、ship 等的 Precision 與 Recall 均達到 **0.99~1.00**，說明模型對這些圖像特徵明顯的類別具有極佳的辨識能力。即便是相對語意接近的 cat 與 dog 類別，其 Precision 與 Recall 也維持在 **0.97** 左右，雖略低於其他類別，但已展現出良好的區辨效果，顯示此模型在處理類別間特徵相似的情境中仍具一定的穩定性。

接者在學習率為 7e-7 的設定下，對 SWIN 模型進行 200 個 epoch 的 training，其訓練及驗證表現如圖 4.13 所示。



(a) Loss (b) Accuracy

圖 4.13、SWIN 模型（learning rate = 7e-7）的 Loss 與 Accuracy 示意圖

從圖 4.13 (a)可以觀察到，隨著訓練進行，訓練損失快速下降，並在中後期趨近於 0，顯示模型已幾乎完全擬合訓練資料。然而，驗證損失在初期略有下降後便維持在約 0.20 並呈現波動，無法持續下降，顯示模型在測試資料上的表現並未同步提升，出現了輕微過擬合的現象。同樣，從圖 4.13 (b) 中，訓練準確率迅速上升並最終接近 100%，而驗證準確率則在約 95% 附近震盪上升，最終穩定在 **97.22%** 左右，與訓練準確率仍存在可觀的落差。這樣的結果說明，雖然模型在訓練資料上具備極高表現，但在測試資料上的泛化能力尚未達到同等水準。

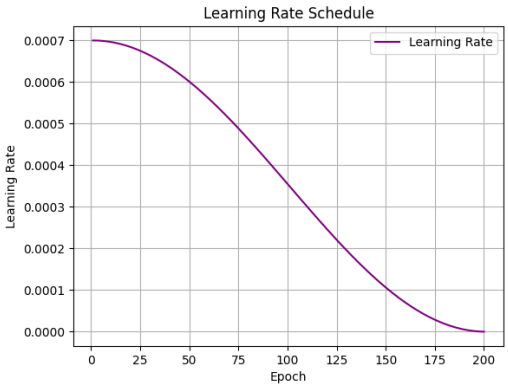


圖 4.14、學習率調整曲線(SWIN, lr=7e-4)

學習率方面，圖 4.14 展示了訓練過程中採用 Cosine Annealing 調度策略的學習率變化情形。學習率從初始值 7e-4 開始，隨著 epoch 增加而逐漸下降，幫助模型在初期快速探索參數空間，後期則穩定收斂至較佳的性能表現。這種策略有助於減少過度震盪的風險，同時提升訓練效率與最終準確率。

表 4.4、SWIN 模型(lr=7e-4)於 CIFAR-10 測試集之整體效能指標

指標名稱	**數值**
Accuracy	97.22 %
Top-1 錯誤率	2.78 %
模型參數總量	86.75 M
FLOPs	16.86 GFLOPs
平均推論時間	31.57 ms/image
GPU 記憶體使用量	4301.61 MB

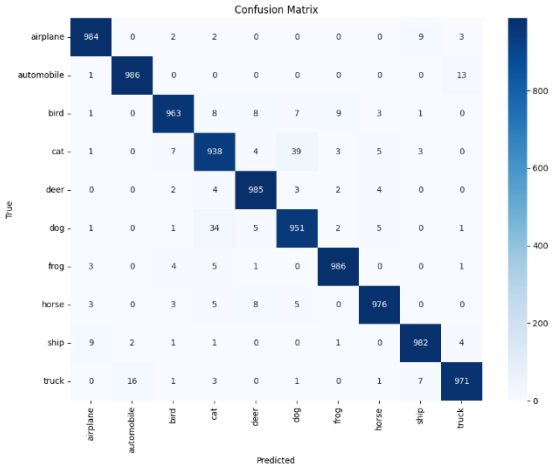


圖 4.15、SWIN 模型 (lr=5e-5) 於 CIFAR-10 測試集之 Confusion Matrix

在表 4.1 可看出 ViT 模型在 CIFAR-10 測試集上的分類準確率達 **97.22%**，展現出極佳的辨識能力與學習效果。模型總參數量為 **86.75M**，在保持高度準確

率的同時，仍具備良好的運算效率。其計算複雜度為 **15.17 GFLOPs**，推論時間平均僅需 **31.57 毫秒/張圖像**，最大 GPU 記憶體使用量為 **4301.61 MB**，顯示此模型可在常見的 GPU 環境中穩定運作。

而在混淆矩陣圖 4.15 可看出，其大多數類別結果與實際標籤高度一致。最大值出現在"automobile -> automobile"與"frog->frog"，相對的最小值則一樣多出現在"cat"與"dog"之間的混淆。這兩類動物圖像在 CIFAR-10 中常因姿勢、背景與色彩分布相似而造成模型誤判。例如，有部分"cat"圖像被誤分類為"dog"，顯示模型在區分這兩類語意相近的類別時仍有一定挑戰，前面結果均相似。

	precision	recall	f1-score	support
airplane	0.98	0.98	0.98	1000
automobile	0.98	0.99	0.98	1000
bird	0.98	0.96	0.97	1000
cat	0.94	0.94	0.94	1000
deer	0.97	0.98	0.98	1000
dog	0.95	0.95	0.95	1000
frog	0.98	0.99	0.98	1000
horse	0.98	0.98	0.98	1000
ship	0.98	0.98	0.98	1000
truck	0.98	0.97	0.97	1000
accuracy			0.97	10000
macro avg	0.97	0.97	0.97	10000
weighted avg	0.97	0.97	0.97	10000

圖 4.16、SWIN 模型 (lr=7e-4) 於 CIFAR-10 測試集之分類指標

而在分類指標如圖 4.16 所示，SWIN Transformer 模型在學習率為 7e-4 的設定下於 CIFAR-10 測試集上達到整體準確率 97%，而 macro average 與 weighted average 的 precision、recall 與 f1-score 皆為 0.97。此結果顯示模型具備高度分類準確率與良好的類別平衡能力，整體分類表現穩定。在各類別的細項表現中，多數類別如 automobile、ship、truck、airplane 以及 horse 的 precision 與 recall 均高達 0.98~0.99，顯示模型對這些結構特徵明確的圖像具備良好辨識效果。相對地，模型在"cat"與"dog"這兩個語意相近、視覺特徵容易混淆的類別上表現略為偏低，是整體分類效能中相對的弱點，但仍維持在高準確率區間。

4.3 結論

表 4.5、ViT 與 SWIN 模型於不同學習率設定下之準確率比較總結

模型	Learning Rate	Accuracy
ViT	5e-5	98.78 %
ViT	7e-4	83.84 %
SWIN Transformer	5e-5	**98.93 %**
SWIN Transformer	7e-4	97.22 %

從表 4.5 可觀察到，ViT 與 SWIN Transformer 兩種模型在 CIFAR-10 任務中於較小學習率（5e-5）下皆能達到極佳的準確率表現，分別為 98.78%（ViT）

與 98.93%（SWIN）。這顯示 CIFAR-10 作為較小且結構清晰的資料集，對於大型模型而言，只需設定較小的學習率即可穩定學習並獲得良好結果，無需過度調整高變動率的訓練策略。

其中，ViT 模型對學習率的敏感性特別明顯。在學習率設定為 5e-5 時能表現出近乎最好的結果，但當學習率提高至 7e-4 時，模型的準確率大幅下降至 83.84%。這可歸因於 ViT 在訓練初期缺乏如 SWIN Transformer 那樣的區域限制結構，使其在高學習率下更容易產生梯度不穩與收斂困難的問題。這也反映在其 loss 曲線異常震盪、驗證誤差持續增加的現象中，明顯顯示出過擬合與訓練不穩的情形。

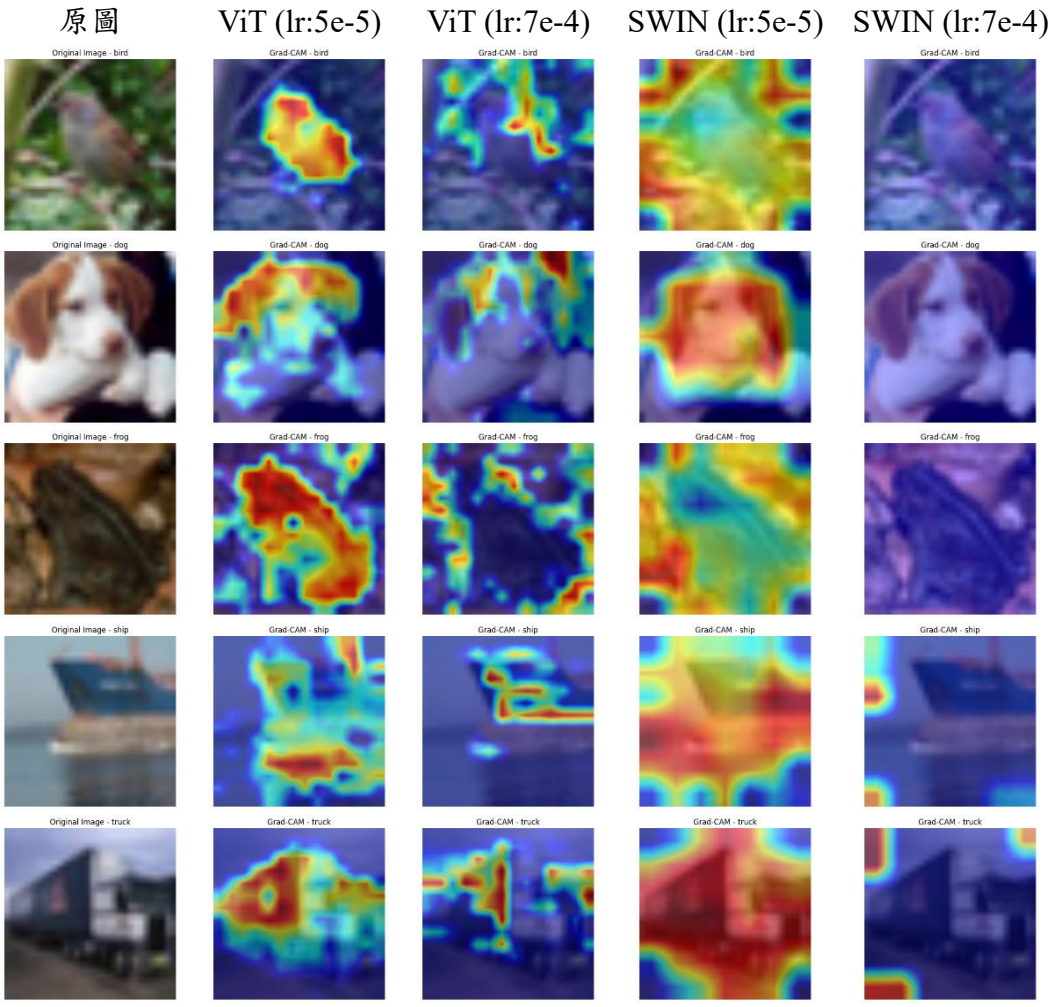


圖 4.16、Grad-Cam 示意圖

圖 4.16 展示了 ViT 與 SWIN 模型在不同學習率設定下，針對 CIFAR-10 測試集中數張圖像所產生的 Grad-CAM 可視化結果。從圖中可以清楚觀察到，各模型對於不同圖像的注意力分佈差異。

可以發現，ViT 模型在學習率為 5e-5 的設定下所產生的 Grad-CAM 圖最為集中且覆蓋目標物體本體區域，顯示模型能準確關注影像中具有判別性的區域，判斷邏輯清晰，解釋性良好。相較之下，ViT 在高學習率（7e-4）設定下的注意

力分佈則較為分散與模糊，部分樣本甚至未能對焦於目標，顯示其訓練效果不穩、模型關注區偏移，這與前述 loss 曲線異常與分類表現不佳一致。

另一方面，SWIN 模型在兩種學習率設定下所產生的 Grad-CAM 可視化結果整體呈現穩定的注意力分佈，大多能涵蓋影像中主要物體區域，具有一定的可解釋性與辨識邏輯。然而，從個別樣本中仍可觀察到注意力分散或偏移的現象，特別是在學習率為 7e-4 時，部分關注區略微偏離主要辨識目標。儘管 SWIN 模型的注意力分佈不如 ViT（lr＝5e-5）那般集中與明確，但其在學習率為 5e-5 的設定下卻達到整體測試準確率最高（98.93%）的表現。

整體而言，SWIN 與 ViT 兩者在不同條件下皆能展現優異表現，SWIN 取得最高的分數表現，而在可視圖上 ViT 展現更具可解釋性的結果。