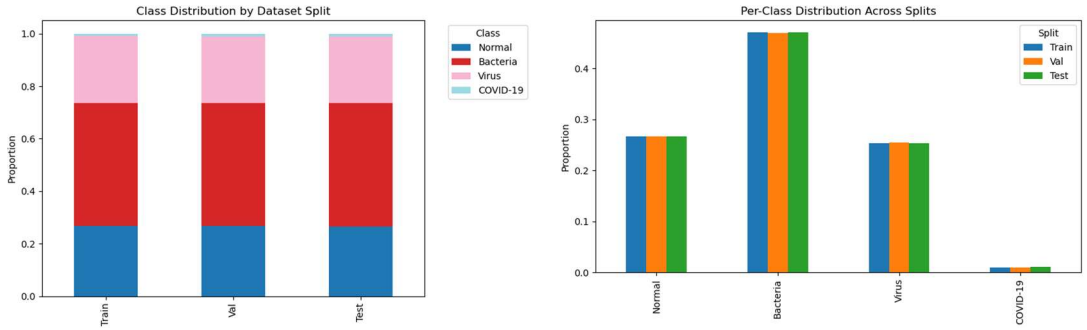
Artificial Intelligence on Medical Imaging Lab 3

314553020 許良亦

1. Introduction

本次實作並比較了四種影像分類模型，分別為 **ResNet-18**、**DenseNet-121**、**EfficientNet-B0**、**ConvNeXt-Tiny** 對於胸腔 X 光影像(CXR)多類別分類任務，資料集包含 **4017** 張訓練影像、**709** 張驗證影像與 **1182** 張測試影像，共分為四個類別：**Normal**、**Bacteria**、**Virus**、**COVID-19**。各資料分佈如下：

Split	Normal	Bacteria	Virus	COVID-19
Train	0.2669	0.4700	0.2534	0.0097
Val	0.2666	0.4697	0.2539	0.0099
Test	0.2665	0.4704	0.2530	0.0102



圖一、(train/val/test) 之資料分布

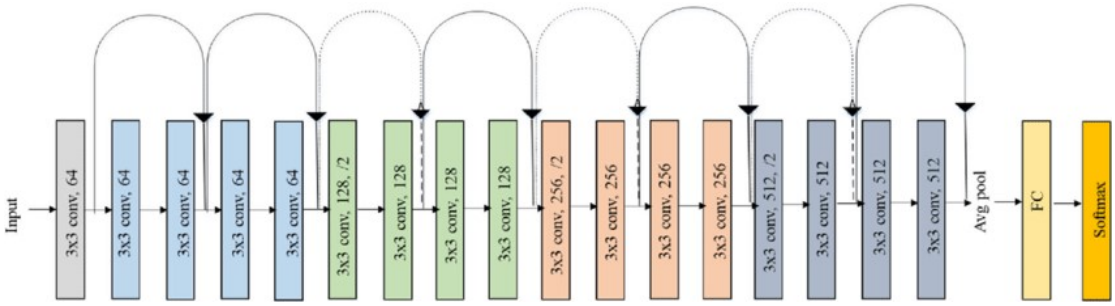
2. Implementation details

A. The details of your model

```python
def make_model(num_classes=4, dropout=0.1, backbone="resnet18"):
    if backbone == "resnet18":
        m = models.resnet18(weights=models.ResNet18_Weights.IMAGENET1K_V1)
        in_f = m.fc.in_features
        m.fc = nn.Sequential(nn.Dropout(dropout), nn.Linear(in_f, num_classes))
        return m

    if backbone == "densenet121":
        m = models.densenet121(weights=models.DenseNet121_Weights.IMAGENET1K_V1)
        in_f = m.classifier.in_features
        m.classifier = nn.Sequential(nn.Dropout(dropout), nn.Linear(in_f, num_classes))
        return m

    if backbone == "efficientnet_b0":
        m = models.efficientnet_b0(weights=models.EfficientNet_B0_Weights.IMAGENET1K_V1)
        in_f = m.classifier[-1].in_features
        m.classifier[-1] = nn.Identity()
        m.classifier = nn.Sequential(
            nn.Dropout(p=dropout, inplace=True),
            nn.Linear(in_f, num_classes)
        )
        return m

    if backbone == "convnext_tiny":
        m = models.convnext_tiny(weights=models.ConvNeXt_Tiny_Weights.IMAGENET1K_V1)
        in_f = m.classifier[2].in_features
        m.classifier = nn.Sequential(
            m.classifier[0],
            m.classifier[1],
            nn.Dropout(p=dropout, inplace=True),
            nn.Linear(in_f, num_classes)
        )
        return m

    raise ValueError("unknown backbone")
```

確保在此分類任務上的一致可比性採用 ImageNet 預訓練權重初始化，並只重設 classifier。讀出每個模型最末層的輸入維度 in_features，再以 Dropout → Linear(num_classes)重建輸出層；其餘卷積部分維持預訓練權重並進行 fine-tuning。

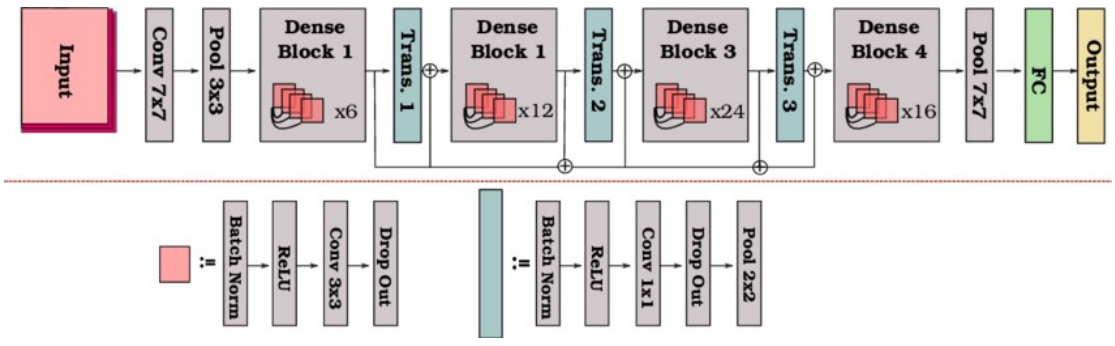
ResNet-18



圖二、ResNet-18 之整體架構

ResNet-18 屬於 Residual Network，其核心理念是透過 Residual Block 來解決深層網路在訓練過程中容易出現的梯度消失與退化問題。每個殘差模組內含有 Skip Connection，可直接將輸入訊號加回輸出，使網路能學習輸入與輸出之間的差異（Residual）而非完整映射，從而加速收斂並提升穩定性。ResNet-18 採用 18 層深度的結構，由多層 3×3 卷積、Batch Normalization 與 ReLU 組成。由於參數量相對較少，訓練效率高，因此常作為影像分類任務中的基準模型。

DenseNet-121



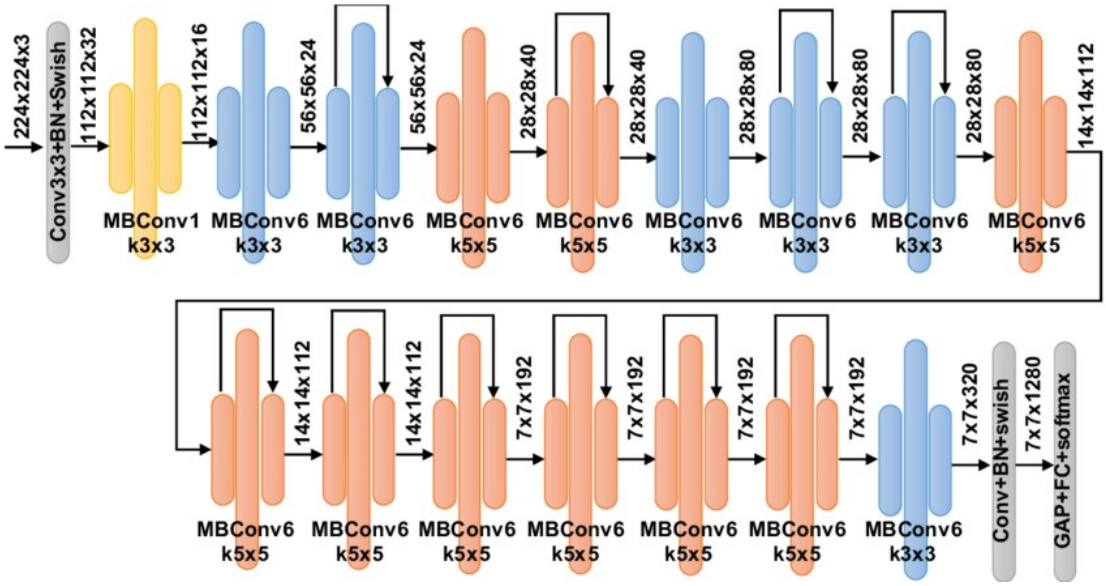
圖三、DenseNet-121 之整體架構

DenseNet-121（Densely Connected Convolutional Network）透過 Dense Connectivity 的設計，使同一區塊內的每一層都能接收前面所有層的特徵輸出。具體而言，第 l層的輸入為所有前面層的特徵拼接結果：

$$x_l = H_l([x_0, x_1, \ldots, x_{l-1}])$$

這種結構使得 Feature Reuse，同時改善梯度傳遞問題，讓模型在相同參數量下能學習更豐富的表示。DenseNet-121 由多個 **Dense Block** 與 **Transition Layer** 組成，每個 Dense Block 內包含多層 BatchNorm、ReLU 與 3×3 卷積層，而 Transition Layer 則透過 1×1 卷積與平均池化進行降維，控制特徵圖大小與通道數。

EfficientNet-B0

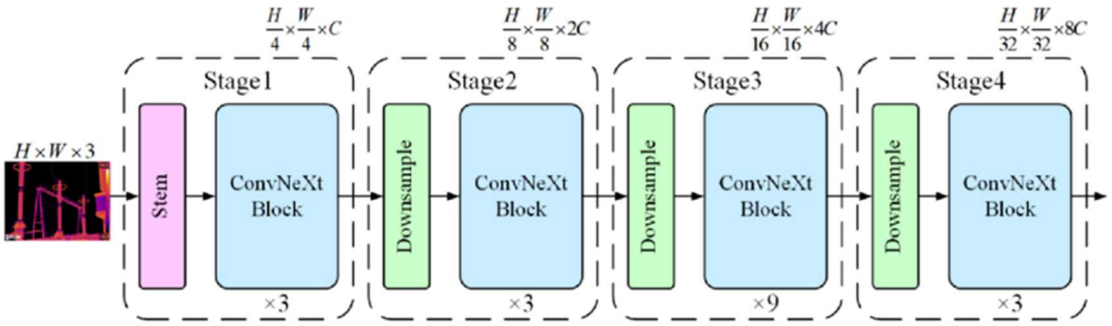


圖四、EfficientNet-B0 之整體架構

EfficientNet-B0 是一種兼顧效能與效率的深度卷積網路，提出了 Compound Scaling Method，同時考量網路的 Depth、Width 與 Resolution。傳統模型通常僅單一方向（例如增加深度）進行擴展，而 EfficientNet 透過一組固定係數 ϕ，根據經驗公式同時擴展三者，使模型在資源限制下達到最佳的準確率與推論效率平衡。此外，EfficientNet 採用了 **MBConv（Mobile Inverted Bottleneck Convolution）**結構與 **Squeeze-and-Excitation（SE）注意力模組**，能在保留精度的同時大幅減

少運算量。B0 為 EfficientNet 系列的基礎版本，在醫學影像分類任務中表現優異，能有效捕捉整體結構與局部細節。

ConvNeXt-Tiny



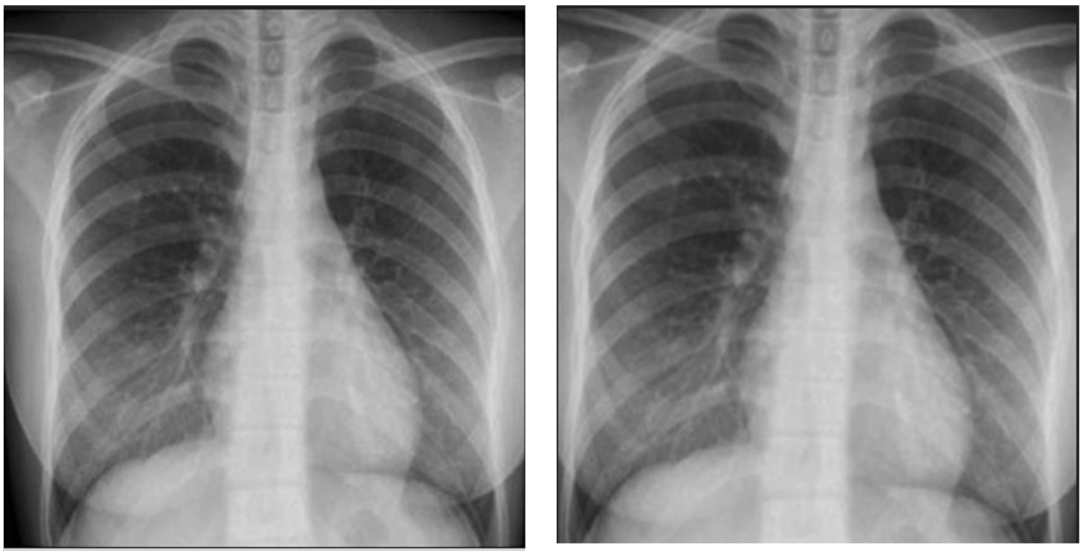
圖五、ConvNeXt-Tiny 之整體架構

ConvNeXt-Tiny 整體網路可分為四個階段（Stage），每個 Stage 由多層 ConvNeXt Block 組成，並透過 stride convolution 進行特徵圖降採樣，形成類似 Transformer 的階層式結構。模型的卷積核心採用深度可分離卷積（**Depthwise＋Pointwise**）以降低參數量與運算成本，同時維持充分的特徵表達能力。此外，ConvNeXt 使用大尺寸卷積核（如 7×7）來模擬 Transformer 的全域感受野，使模型能同時捕捉局部與整體的影像資訊。為了提升穩定性與非線性表現，網路以 Layer Normalization 取代傳統的 Batch Normalization，並採用 GELU 作為啟動函數，讓特徵分佈更平滑自然。最後，在輸入端設計上，模型使用簡化的 stem 結構（單層 4×4 卷積取代多層堆疊），有效減少早期特徵損失並提升整體運算效率。

B. The details of your Dataloader

```
if train:
    aug = [
        A.ShiftScaleRotate(shift_limit=0.02, scale_limit=0.1, rotate_limit=7,
                           border_mode=cv2.BORDER_REFLECT, p=0.7),
        A.RandomBrightnessContrast(brightness_limit=0.1, contrast_limit=0.15, p=0.7),
        A.GaussNoise(var_limit=(5.0, 15.0), p=0.15),
        A.HorizontalFlip(p=0.5),
    ]
    tfms = A.Compose(base_preproc + aug + [
        A.Normalize(),
        ToTensorV2(),
    ])
else:
```

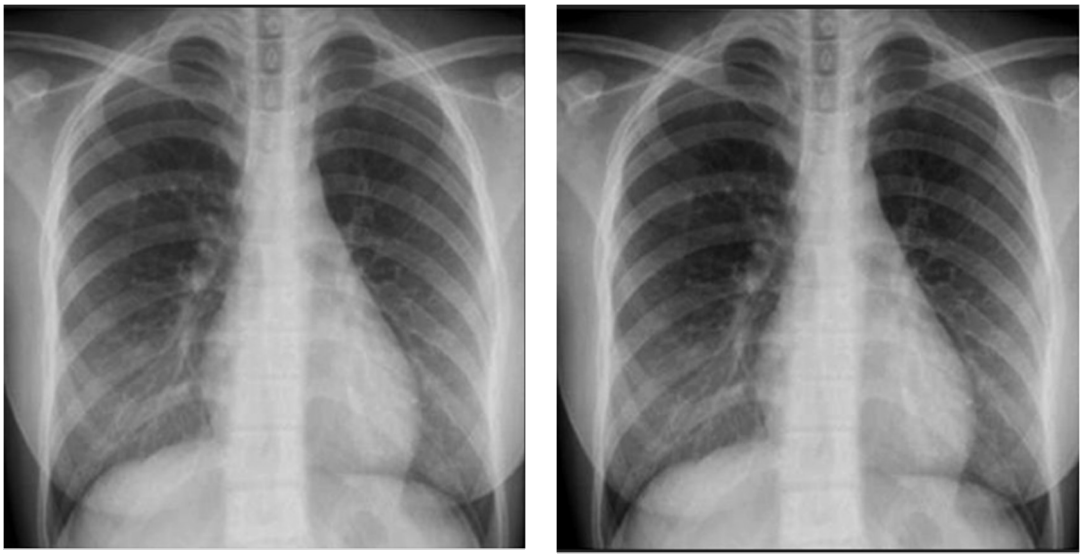
ShiftScaleRotate



圖六、原始影像(左)及增強後之影像(右)

在本次實驗中，模型訓練階段使用 Affine Transformation 作為資料增強的一部分。此方法會同時對影像進行小幅的平移、縮放與旋轉 Rotate 變化，用以模擬病人在拍攝胸腔 X 光時因體位、角度或設備參數所造成的自然差異。這樣的設計能有效降低模型對幾何細節的過擬合，提升其在不同臨床條件下的泛化能力。旋轉角度控制在 ±5°～±7° 之間，縮放比例約為 ±5%～10%，而平移幅度約 1%～3%，確保影像變形在合理範圍內，不會破壞主要解剖結構。

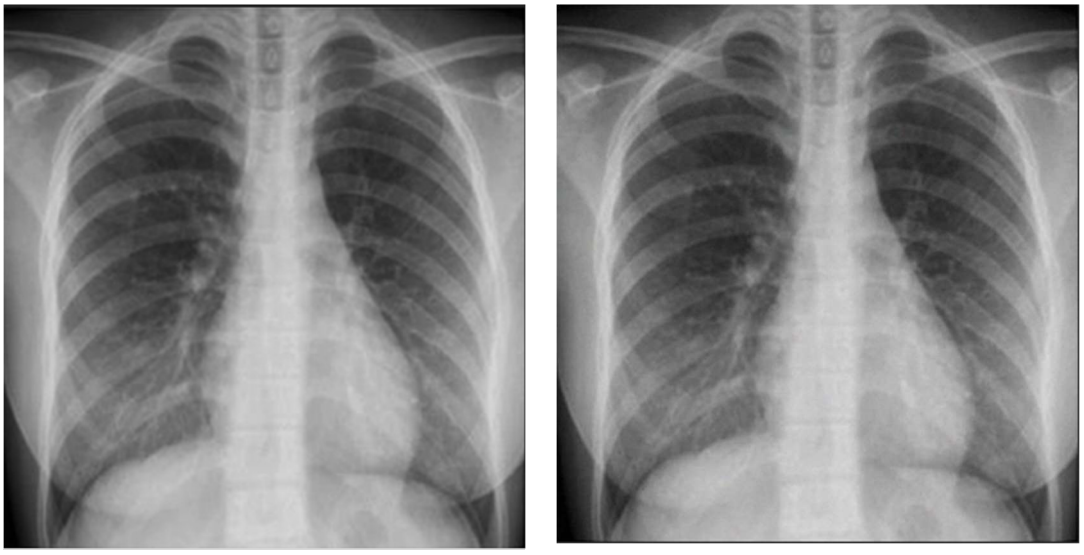
RandomBrightnessContrast



圖七、原始影像(左)及增強後之影像(右)

此外，我們於訓練階段使用 **Random Brightness & Contrast Augmentation 作為資料增強手段之一。**此方法會**對整張影像的亮度與對比度進行小幅隨機調整**，以模擬不同醫療院所、攝影設備或後處理流程所造成的曝光與灰階動態差異。透過此操作，模型能學習到更具魯棒性的特徵表示，減少對特定影像亮度分佈的依賴，進而提升跨機台與跨資料庫的泛化能力。在實作上，亮度與對比的變化幅度皆控制在 ±5%～10% 範圍內，以確保影像的診斷資訊與病灶細節不被過度強化或削弱。

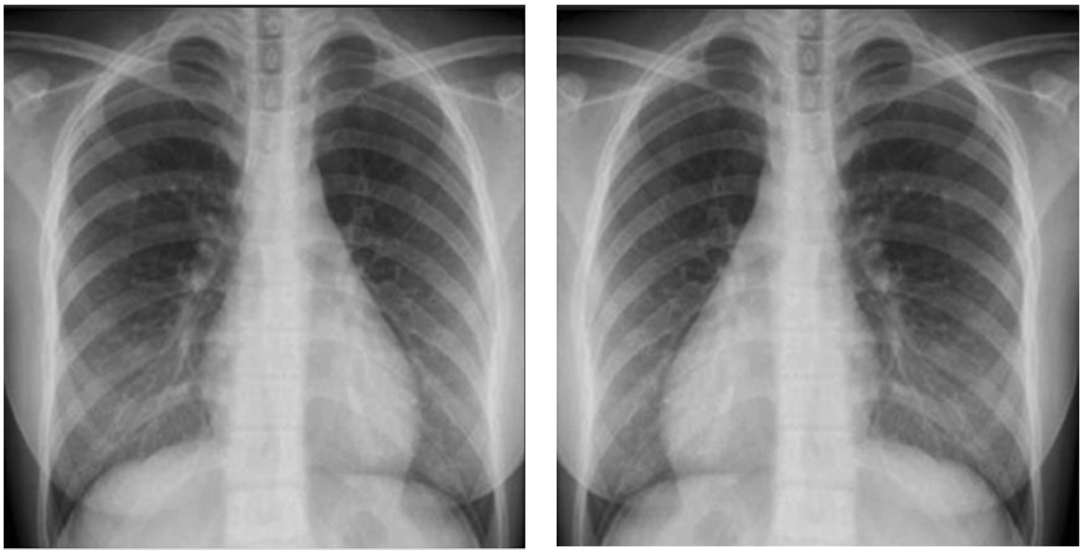
GaussNoise



圖八、原始影像(左)及增強後之影像(右)

在本次實驗中除了上面兩種操作，我還採用了**高斯雜訊**作為資料增強策略之一。此方法會**在影像上疊加零均值的隨機高斯噪音，用以模擬不同感測器或影像壓縮過程中所產生的訊號干擾。**藉由在訓練過程中加入微量噪音，可有效降低模型對過於乾淨影像的依賴，並提升其對實際臨床資料中雜訊變化的穩健性。在設定上，σ 控制於 0.01～0.05 之間，以確保影像仍維持可辨識的病灶細節與組織結構。在醫學影像應用中需特別注意，若噪音強度過大，可能導致偽影產生或掩蓋微小病灶，因此需在提升魯棒性與維持診斷可讀性之間取得平衡。

HorizontalFlip



圖九、原始影像(左)及增強後之影像(右)

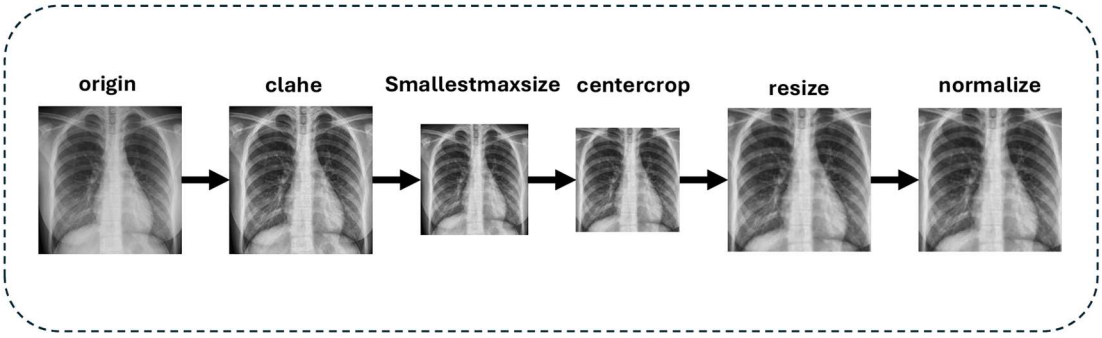
最後，我也把資料增強中加入了水平翻轉，以增加影像的視覺多樣性。此方法透過對影像進行左右鏡像轉換，使模型能學習在不同對稱方向下的一致性特徵。對於胸腔 X 光等大多數對稱性強的醫學影像而言，水平翻轉不會改變影像的語意，能有效擴增資料量並提升泛化能力。然而在特定任務中需特別注意：若研究對象與左右側或檢視角度高度相關，則不宜使用水平翻轉，或必須同步調整對應標籤，以避免產生語意錯置與分類錯誤的情況。

3. Strategy design

A. How did you pre-process your data?

```python
def build_tfms(train=True, img_size=256, center_crop_ratio=0.90):
    crop_sz = int(img_size * center_crop_ratio)

    base_preproc = [
        A.CLAHE(clip_limit=1.5, tile_grid_size=(8,8), p=1.0),
        A.SmallestMaxSize(max_size=max(img_size, crop_sz)),
        A.CenterCrop(height=crop_sz, width=crop_sz),
        A.Resize(img_size, img_size),
    ]

    if train:
        aug = [
            A.ShiftScaleRotate(shift_limit=0.02, scale_limit=0.08, rotate_limit=5,
                               border_mode=cv2.BORDER_REFLECT, p=0.7),
            A.RandomBrightnessContrast(brightness_limit=0.1, contrast_limit=0.15, p=0.7),
            A.GaussNoise(var_limit=(5.0, 15.0), p=0.15),
            A.HorizontalFlip(p=0.5),
        ]
        tfms = A.Compose(base_preproc + aug + [
            A.Normalize(),
            ToTensorV2(),
        ])
    else:
        tfms = A.Compose(base_preproc + [
            A.Normalize(),
            ToTensorV2(),
        ])
    return tfms
```

為統一不同來源影像的對比與空間尺度，我們在三個資料分割（train/val/test）上皆套用相同的前處理流程。首先以 CLAHE 進行對比度受限的自適應直方圖等化，提升胸腔結構的局部對比而不過度放大噪訊；接著採用 SmallestMaxSize 將影像等比縮放至較長邊不小於目標尺寸，再以 CenterCrop 擷取中心 90% 視野，確保感興趣區域被保留；其後以 Resize 將影像重採樣至固定輸入大小，並套用 Normalize 與 ToTensor 轉為模型張量。



圖十、pre-processing pipline

B. What makes your training strategy special?

本次實驗的訓練策略結合了資料平衡抽樣、動態學習率調整、自動混合精度訓練（AMP）與早停機制（Early Stopping），以提升模型的穩定性與泛化能力，具體特色如下：

- **類別不平衡處理（WeightedRandomSampler + Class-weighted Loss）**

```python
# Sampler
def make_sampler(targets):
    cnt = Counter(targets)
    n = len(targets)
    weights = [n / cnt[t] for t in targets]
    return WeightedRandomSampler(weights, num_samples=n, replacement=True)
```

```python
# class weights
cls_cnt = np.bincount(targets, minlength=len(CLASSES))
w = (cls_cnt.sum() / (cls_cnt + 1e-6))
crit = nn.CrossEntropyLoss(weight=torch.tensor(w, dtype=torch.float32, device=device))
```

由於各類別樣本數分布不均，訓練階段採用了 **WeightedRandomSampler** 進行平衡抽樣，使每個 mini-batch 的各類樣本機率更接近均勻；同時在損失函數中根據每類樣本數計算權重（CrossEntropyLoss(weight=w)），減少主類別對模型的偏倚，提升少數類別的辨識率。

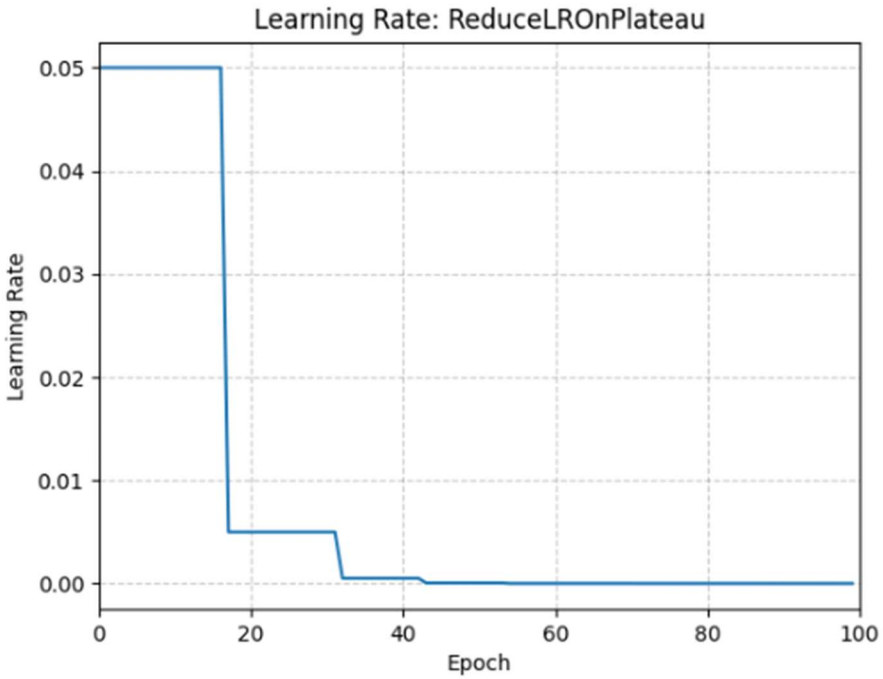
- **多階段資料處理與增強（Data Pre-processing + Augmentation）**

前處理部分使用 CLAHE 強化局部對比度，並統一影像尺度與中心視野（CenterCrop + Resize）；訓練階段額外加入隨機幾何與光度增強（ShiftScaleRotate, RandomBrightnessContrast, GaussNoise, HorizontalFlip），以模擬實際拍攝變異並提升模型的泛化能力。(2-B、3-A)

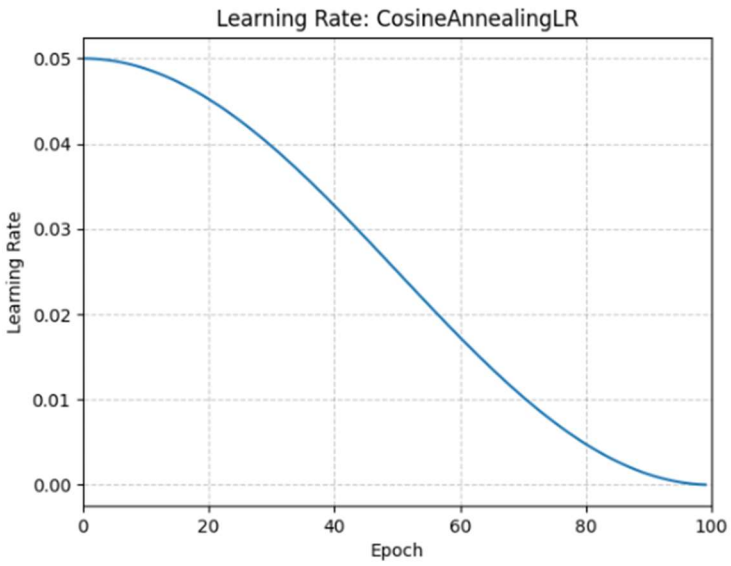
- **動態學習率調整（Adaptive LR Scheduling）**

本次實作實驗可以使用兩種學習率排程策略，讓模型更好的進行學習。

```python
def make_scheduler(optimizer, args):
    if args.sched == "plateau":
        mode = "max" if args.monitor == "val_f1" else "min"
        return ReduceLROnPlateau(
            optimizer, mode=mode,
            factor=args.plateau_factor, patience=args.plateau_patience,
            min_lr=args.min_lr
        )
    else:  # cosine
        return CosineAnnealingLR(
            optimizer, T_max=args.epochs, eta_min=args.min_lr
        )

def get_lr(optimizer):
    return optimizer.param_groups[0]["lr"]
```



ReduceLROnPlateau 會根據驗證集表現(本實驗以 macro-F1 或 validation loss 作為監控指標),在多個 epoch 未獲得顯著改善時自動降低學習率,以防止模型陷入局部最小值或收斂停滯。當模型表現連續 p 個 epoch 沒有提升時,學習率會乘上一個衰減係數 γ(如 0.5 或 0.1),讓模型以較小步伐進行細部微調。



CosineAnnealingLR 採用餘弦函數週期性地平滑調整學習率，從最大值 $\eta_{\max}$ 逐步下降到最小值 $\eta_{\min}$，模擬由快到慢的收斂過程。這樣的策略能在每個週期內讓模型經歷高學習率的探索與低學習率的穩定收斂階段，避免訓練早期過早收斂，並提高模型在不同局部最小值間的探索能力。

- **自動混合精度訓練（Automatic Mixed Precision, AMP）**

```
autocast = torch.amp.autocast

for ep in range(1, args.epochs + 1):
    # ===== Train =====
    model.train()
    running_loss = 0.0
    progress = tqdm(tr, desc=f"Epoch {ep}/{args.epochs}", ncols=200, leave=True)

    for x, y in progress:
        x, y = x.to(device, non_blocking=True), y.to(device, non_blocking=True)
        opt.zero_grad(set_to_none=True)

        with autocast('cuda', enabled=torch.cuda.is_available()):
            out = model(x)
            loss = crit(out, y)

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

在實驗中，訓練過程採用了自動混合精度訓練（Automatic Mixed Precision, AMP）機制，以提升訓練效率與資源利用率。當 GPU 支援半精度運算（FP16）時，模型會自動在單精度（FP32）與半精度（FP16） 間動態切換，達成更快的前向與反向傳遞，同時大幅減少顯存占用。

- **早停與最佳權重保存（Early Stopping & Checkpointing）**

```
# --- Save best & Early Stopping ---
if val_f1 > best_f1:
    best_f1 = val_f1
    patience_left = args.es_patience
    torch.save(model.state_dict(), best_path)
else:
    patience_left -= 1
    if patience_left <= 0:
        print(f"Early stopping at epoch {ep} (best val_F1={best_f1:.4f})")
        break
```

當驗證集的 macro-F1 分數在多個 epoch 未提升時，自動觸發 early stopping，避免過擬合，確保最終結果對應最佳驗證表現。

C. All your training details

ResNet-18
Input size: 256 × 256
Optimizer: AdamW (lr = 5e-5, weight decay = 2e-4)
Batch size: 16
Number of epochs: 50
Loss function: Cross-Entropy Loss
Learning rate scheduler: ReduceLROnPlateau
Dropout: 0.1
Early stopping: patience = 10 epochs

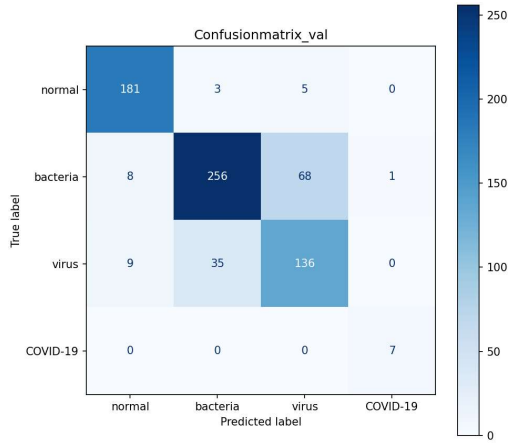
DenseNet-121
Input size: 256 × 256
Optimizer: AdamW (lr = 5e-5, weight decay = 2e-4)
Batch size: 16
Number of epochs: 50
Loss function: Cross-Entropy Loss
Learning rate scheduler: ReduceLROnPlateau
Dropout: 0.1
Early stopping: patience = 10 epochs

EfficientNet-B0
Input size: 256 × 256
Optimizer: AdamW (lr = 5e-5, weight decay = 2e-4)
Batch size: 16
Number of epochs: 50
Loss function: Cross-Entropy Loss
Learning rate scheduler: ReduceLROnPlateau
Dropout: 0.1
Early stopping: patience = 10 epochs

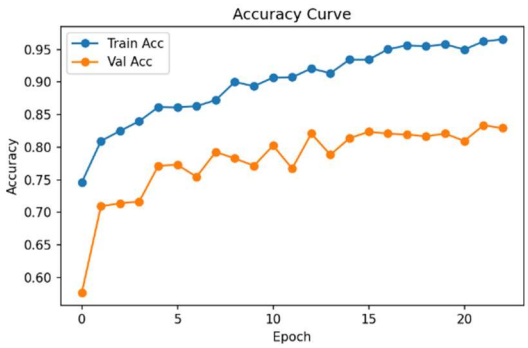
ConvNeXt-Tiny
Input size: 256 × 256
Optimizer: AdamW (lr = 5e-5, weight decay = 2e-4)
Batch size: 16
Number of epochs: 50
Loss function: Cross-Entropy Loss
Learning rate scheduler: ReduceLROnPlateau
Dropout: 0.1
Early stopping: patience = 10 epochs

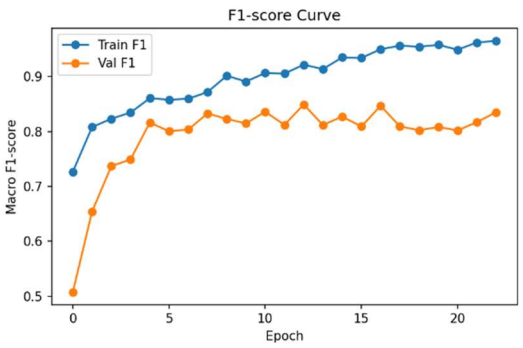
4. Discussion

ResNet-18



	precision	recall	f1-score	support
normal	0.9141	0.9577	0.9354	189
bacteria	0.8707	0.7688	0.8166	333
virus	0.6507	0.7556	0.6992	180
COVID-19	0.8750	1.0000	0.9333	7
accuracy			0.8181	709
macro avg	0.8277	0.8705	0.8461	709
weighted avg	0.8265	0.8181	0.8196	709

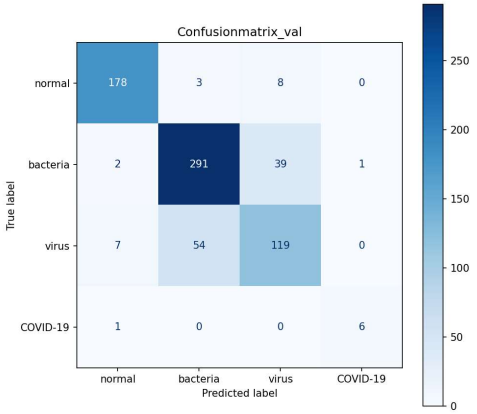




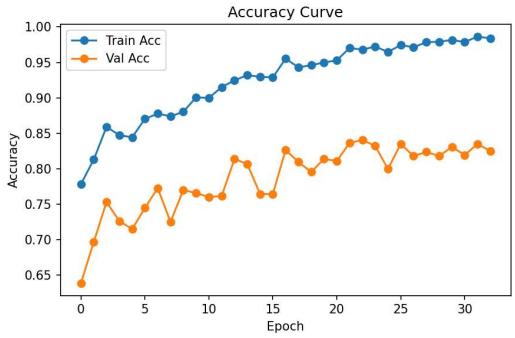
從數據可以發現，使用 ResNet-18 於胸腔 X 光多類別分類任務中取得良好表現。從混淆矩陣可觀察到模型對主要類別 bacteria 與 virus 的分類仍存在部分混淆，而 normal 與 COVID-19 的辨識表現則相對穩定。整體驗證集的 準確率 (Accuracy) 為 81.8%，macro F1-score 為 0.8461，顯示模型對各類別整體的平衡性尚可。訓練曲線顯示，隨著 epoch 增加，訓練 F1 與準確率持續上升，而驗證指標於約第 15 epoch 後趨於平穩，表示模型已接近收斂。在各類別中，normal (F1=0.935) 與 COVID-19 (F1=0.933) 的表現最佳，反映模型對明顯特徵

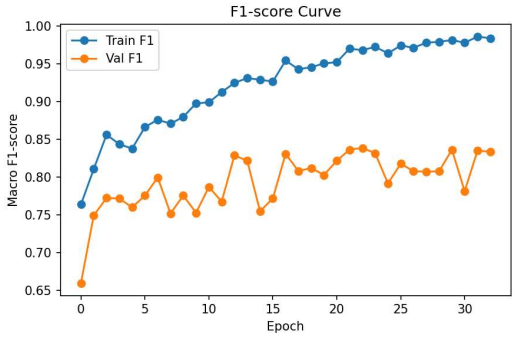
的樣本具有良好辨識能力；virus (F1=0.699) 的結果相對偏低，推測與影像特徵與 bacteria 類別較為相似有關。

DenseNet-121



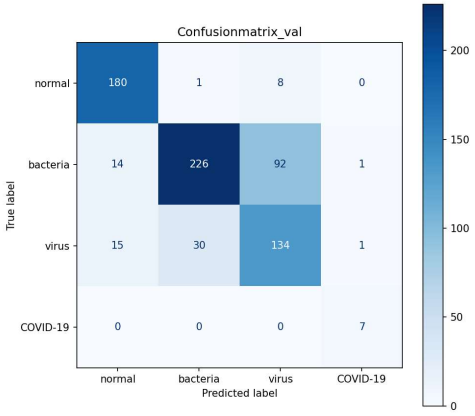
	precision	recall	f1-score	support
normal	0.9468	0.9418	0.9443	189
bacteria	0.8362	0.8739	0.8546	333
virus	0.7169	0.6611	0.6879	180
COVID-19	0.8571	0.8571	0.8571	7
accuracy			0.8378	709
macro avg	0.8393	0.8335	0.8360	709
weighted avg	0.8356	0.8378	0.8362	709



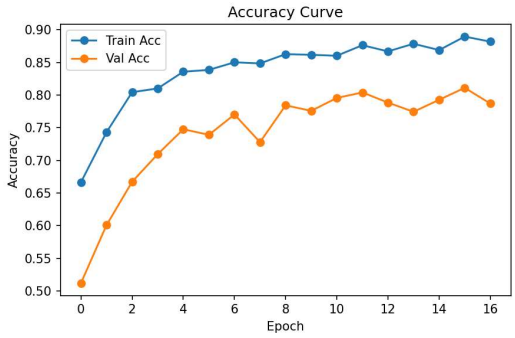


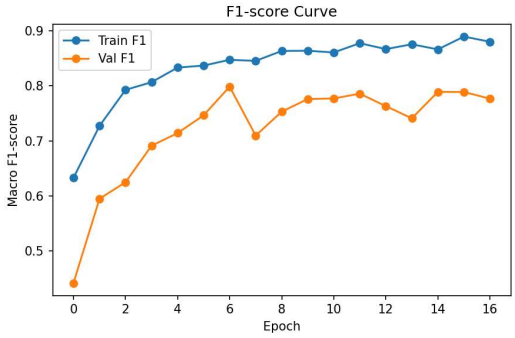
在相同的資料與訓練條件下，DenseNet-121 較 ResNet-18 展現出更強的特徵重用能力與穩定的訓練收斂表現。整體驗證集準確率為 83.8%、macro F1-score 為 0.8360，與 ResNet-18 相近，顯示 DenseNet 的密集連接在 CXR 任務中能有效保留細微影像特徵，但對於中等難度類別（如 virus）仍存在混淆。混淆矩陣顯示，bacteria 與 virus 間的誤判比例仍高，反映兩者在胸腔影像中影像表現相似，模型難以明確分界。各類別的表現中，normal (F1=0.944) 仍為最佳，其次是 bacteria (F1=0.855)，而 virus (F1=0.688) 最低。COVID-19 類別樣本數較少，因此其表現（F1=0.857）略受限制，但模型仍能穩定辨識。

EfficientNet-B0



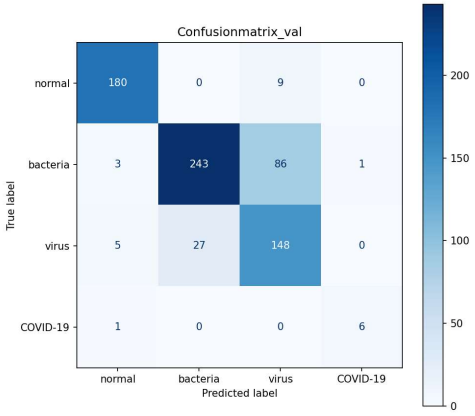
	precision	recall	f1-score	support
normal	0.8612	0.9524	0.9045	189
bacteria	0.8794	0.6787	0.7661	333
virus	0.5726	0.7444	0.6473	180
COVID-19	0.7778	1.0000	0.8750	7
accuracy			0.7715	709
macro avg	0.7728	0.8439	0.7982	709
weighted avg	0.7957	0.7715	0.7739	709



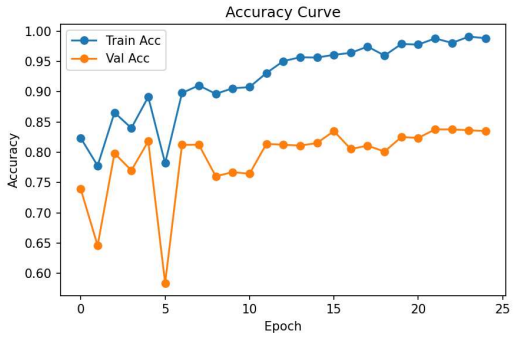


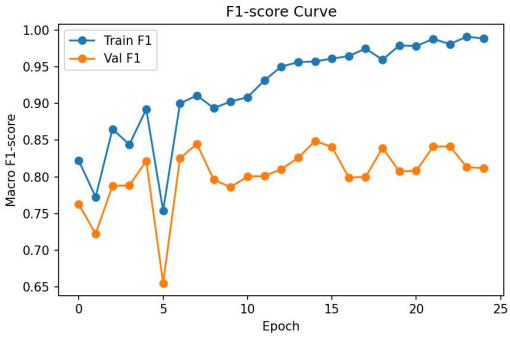
在本次實驗中，EfficientNet-B0 整體驗證集表現為 Accuracy 77.15%、Macro F1-score 0.798，略低於 ResNet-18 與 DenseNet-121，顯示其基礎版本的容量較小，在處理多類別胸腔 X 光影像時對細微差異的學習仍有限。從混淆矩陣可見，模型對 bacteria 與 virus 類別仍有顯著混淆，特別是 virus 的 F1 僅 0.647，說明模型在區分相似病灶紋理時能力不足。然而，normal (F1 = 0.905) 與 COVID-19 (F1 = 0.875) 仍維持不錯的準確度，代表模型在辨識明顯差異影像時表現穩定。

ConvNeXt-Tiny



	precision	recall	f1-score	support
normal	0.9524	0.9524	0.9524	189
bacteria	0.9000	0.7297	0.8060	333
virus	0.6091	0.8222	0.6998	180
COVID-19	0.8751	0.8571	0.8571	7
accuracy			0.8138	709
macro avg	0.8296	0.8404	0.8288	709
weighted avg	0.8397	0.8138	0.8185	709





ConvNeXt-Tiny 作為一種基於 Vision Transformer 改進的卷積架構，結合了深層卷積與 Layer Normalization，具備更高的表徵能力與穩定性。本次實驗中，其驗證集整體表現為 **Accuracy 83.9%**、**Macro F1-score 0.837**，在四種模型中屬於前段水準，顯示該架構在胸腔 X 光影像分類任務中能有效捕捉高階語意特徵。混淆矩陣顯示，virus 與 bacteria 間的混淆仍最為明顯，代表兩類肺炎影像在病灶紋理上仍存在高度相似性。各類別表現中，normal (F1 = 0.94) 與 COVID-19 (F1 = 0.87) 準確率最佳，而 virus (F1 ≈ 0.80) 相對較低，顯示模型在少樣本與中度病變影像上的辨識仍可提升。

Github

https://github.com/Ianuyu/AIMI/tree/main