利用 CNN 進行視網膜疾病判斷 (DRUSEN vs NORMAL)

1. 故事介紹

(1) 視網膜影像是眼科醫生常用來檢測疾病的重要依據。

(2) DRUSEN（玻璃膜疣）是黃斑部病變 (AMD, Age-related Macular Degeneration) 的早期徵兆，若能及早辨識，可幫助醫師進行追蹤與治療。

(3) 本專案的目標是建立一個 影像分類模型，將視網膜圖片自動分類為「DRUSEN」或「NORMAL」。

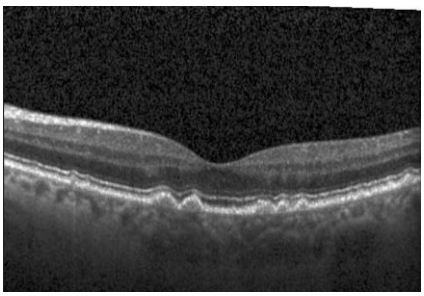
2. 資料集介紹

來源：https://www.kaggle.com/datasets/paultimothymooney/kermany2018/data

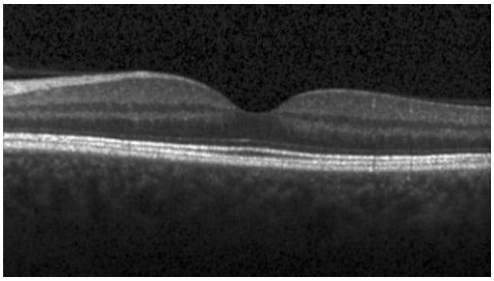
分類：DRUSEN/NORMAL

圖片大小：統一處理成 224x224。

DRUSEN: 視網膜色素上皮層呈鋸齒狀



NORMAL: 視網膜色素上皮層平滑



資料分配：

訓練集：DRUSEN 245 筆，NORMAL 755 筆。

測試集：DRUSEN 242 筆，NORMAL 242 筆。

3. 資料讀取與整理展示

從完整資料集中只挑出 DRUSEN 與 NORMAL 兩類影像。

使用 ImageFolder + Subset 篩選資料

設定 limit=1000，避免資料太大

使用 DataLoader 做 batch 載入

```
keep_classes = ['DRUSEN', 'NORMAL']
cls2idx = train_full.class_to_idx
keep_idx = set(cls2idx[c] for c in keep_classes if c in cls2idx)


def filter_subset(imgfolder, limit=None):  3 用法  Nothing Chang
    kept = [i for i, (_, y) in enumerate(imgfolder.samples) if y in keep_idx]
    if limit is not None and len(kept) > limit:
        kept = random.sample(kept, limit)  # 隨機抽 limit 筆
    return Subset(imgfolder, kept)


train_ds = filter_subset(train_full, limit=data_limit)
val_ds = filter_subset(val_full, limit=data_limit)
test_ds = filter_subset(test_full, limit=data_limit)
```

```
batch_size = 64
num_workers = 4  # Windows/本機可先用 0~2，如果有問題就設 0
prefetch_factor = 4 # 提前N個epoch準備
persistent_workers = True  # workers 過勞死，不休息
pin_memory = torch.cuda.is_available()

train_loader = DataLoader(train_ds, batch_size=batch_size, shuffle=True,
                          num_workers=num_workers, pin_memory=pin_memory,
                          prefetch_factor=prefetch_factor, persistent_workers=persistent_workers)
val_loader   = DataLoader(val_ds,   batch_size=batch_size, shuffle=False,
                          num_workers=num_workers, pin_memory=pin_memory,
                          prefetch_factor=prefetch_factor, persistent_workers=persistent_workers)
test_loader  = DataLoader(test_ds,  batch_size=batch_size, shuffle=False,
                          num_workers=num_workers, pin_memory=pin_memory,
                          prefetch_factor=prefetch_factor, persistent_workers=persistent_workers)
```

4. 模型建立步驟展示

```python
class SimpleCNN(nn.Module):  Nothing Chang
    def __init__(self, dropout_rate=0.5):  Nothing Chang
        super().__init__()
        self.features = nn.Sequential(
            nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, stride=1, padding=1),  # 卷積層，3通道→32通道
            nn.BatchNorm2d(32),                                    # 批次正規化
            nn.ReLU(),                                             # 激活
            nn.MaxPool2d(2, 2),                                    # 最大池化

            nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),

            nn.Conv2d(64, 128, kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),

            nn.Conv2d(128, 256, kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(256),
            nn.ReLU(),
            nn.MaxPool2d(2, 2)
        )
        self.classifier = nn.Sequential(
            nn.Dropout(dropout_rate),                              # Dropout
            nn.Linear(256 * 14 * 14, 256),                         # 全連接層，展平後→256維
            nn.ReLU(),                                             # 激活
            nn.Dropout(dropout_rate),                              # Dropout
            nn.Linear(256, 2)                                      # 全連接層，256→2類
        )
```

```python
class SimpleCNN(nn.Module):  Nothing Chang
    def forward(self, x):  Nothing Chang
        x = self.features(x)
        # x.view(x.size(0), -1) 的用法與意義：
        # x.size(0)：取得 batch_size，保持每個 batch 的樣本數不變
        # -1：自動計算剩餘維度，將所有 channel 與空間維度展平成一維
        # 例如 x shape (batch_size, 256, 14, 14) 變成 (batch_size, 256*14*14)
        x = x.view(x.size(0), -1)  # 展平多維特徵圖為二維。或是用 x = x.flatten(1)  從第1維開始展平，效果相同
        x = self.classifier(x)
        return x
```

```python
def run_one_epoch(model, loader, train=True):  2 用法  Nothing Chang
    if train:
        model.train()
    else:
        model.eval()
    total, correct, running_loss = 0, 0, 0.0
    for xb, yb in loader:
        xb, yb = xb.to(device), yb.to(device)
        if train:
            optimizer.zero_grad()
        with torch.set_grad_enabled(train):
            logits = model(xb)
            loss = criterion(logits, yb)
            if train:
                loss.backward()
                optimizer.step()
        running_loss += loss.item() * xb.size(0)
        preds = logits.argmax(dim=1)
        correct += (preds == yb).sum().item()
        total += xb.size(0)
    avg_loss = running_loss / total
    acc = correct / total
    return avg_loss, acc
```

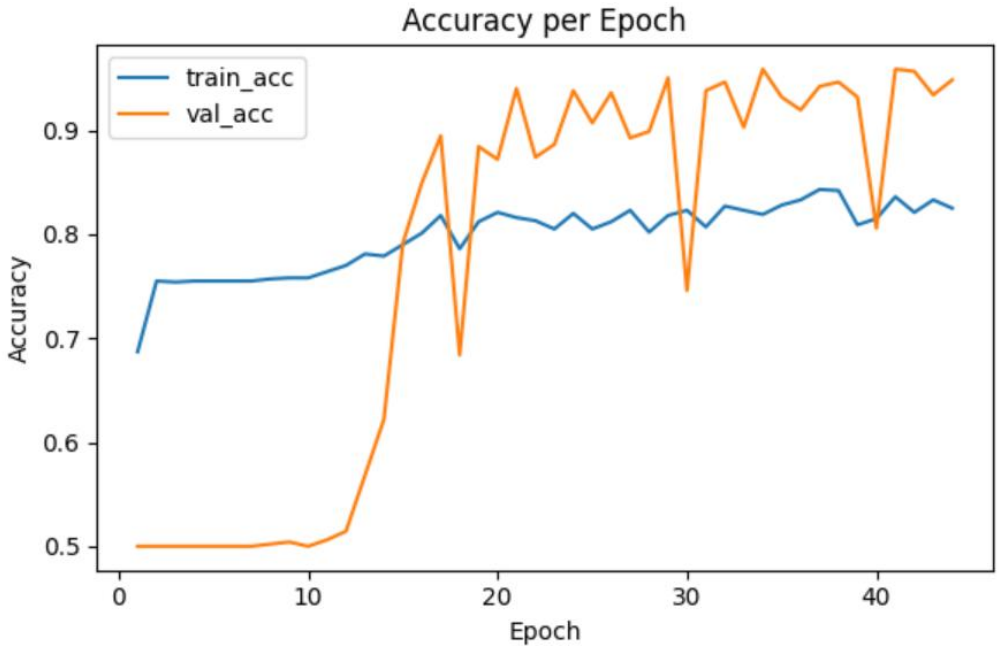
```python
print("開始epoch訓練...")
early_stopping_time = 10
early_stopping_count = early_stopping_time
# --- 訓練主迴圈 ---
for epoch in range(1, num_epochs + 1):
    t0 = time()
    train_loss, train_acc = run_one_epoch(model, train_loader, train=True)
    # val_loss, val_acc = run_one_epoch(model, val_loader, train=False)
    test_loss, test_acc = run_one_epoch(model, test_loader, train=False)
    dt = time() - t0
    log_epoch(epoch, num_epochs, train_loss, train_acc, test_loss, test_acc, dt)

    # 儲存最佳模型（以 val_acc 為準）
    # if val_acc > best_val_acc:
    if test_acc > best_val_acc:
        best_val_acc = test_acc
        torch.save(model.state_dict(), save_path)
        print(f"  -> Saved new best to {save_path} (val_acc={best_val_acc:.4f})")
        early_stopping_count = early_stopping_time
    else:
        early_stopping_count -= 1
        if early_stopping_count == 0:
            print("Early stopping...")
            break
```

5. 模型 Fit 過程畫圖展示

```
classes: ['DRUSEN', 'NORMAL']
class_to_idx: {'DRUSEN': 0, 'NORMAL': 1}
sizes: 34931 16 484
device: cuda
篩選後 sizes: 1000 16 484
train  -> DRUSEN: 245, NORMAL: 755, total: 1000
test   -> DRUSEN: 242,  NORMAL: 242,  total: 484
```

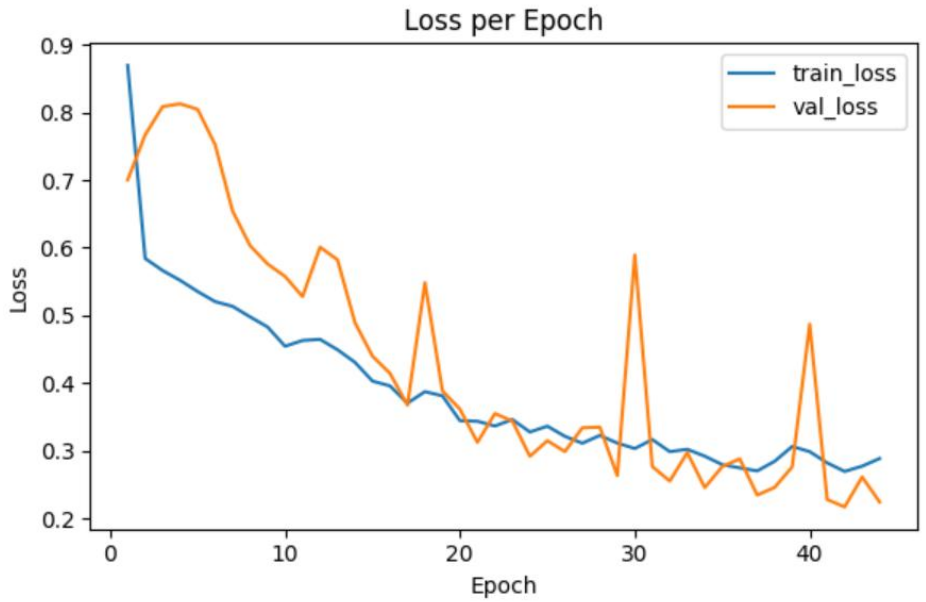
訓練集準確率：穩定上升到大約 **0.82 ~ 0.84** 左右。

驗證集準確率：前 10 個 epoch 幾乎卡在 0.5，之後快速上升，最高能到 **0.95 左右**。



藍線：持續下降，從 0.9 降到約 0.28。

橘線：下降趨勢明顯，但中間有幾次尖峰，表示某些 epoch 模型在驗證資料上表現不穩定。最後穩定在 **0.2~0.3**。



NORMAL 幾乎都能正確判斷（236/242）。

ROC-AUC 0.995：顯示模型幾乎能把 DRUSEN 與 NORMAL 完美區分。

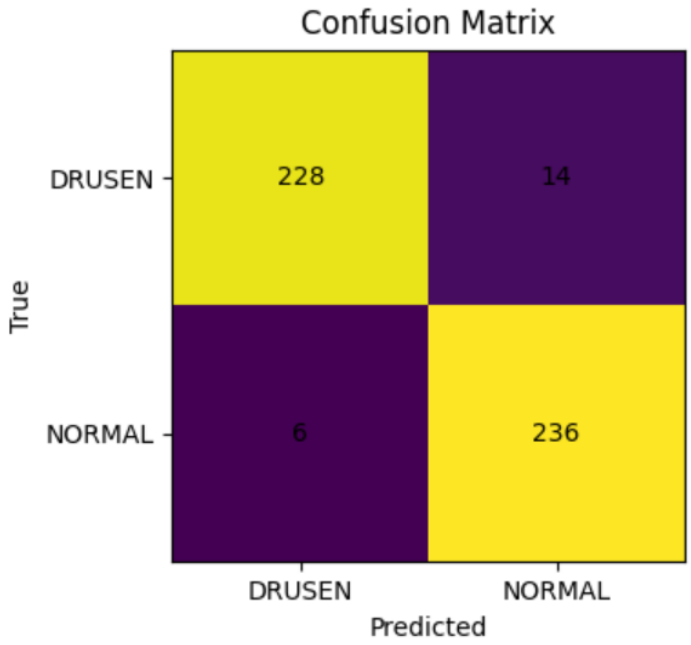
```
best_val_acc: 0.9586776859504132
[TEST] accuracy: 0.9587

[TEST] classification_report:
              precision    recall  f1-score   support

      DRUSEN     0.9744    0.9421    0.9580       242
      NORMAL     0.9440    0.9752    0.9593       242

    accuracy                         0.9587       484
   macro avg     0.9592    0.9587    0.9587       484
weighted avg     0.9592    0.9587    0.9587       484

[TEST] confusion_matrix (rows=true, cols=pred):
 [[228  14]
 [  6 236]]
[TEST] ROC-AUC (NORMAL as positive): 0.9947
```

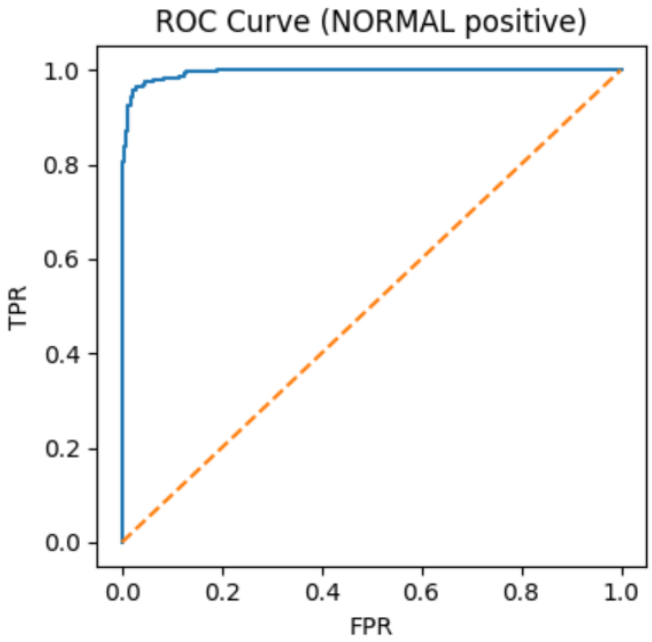
6. 準確度展示

準確率 (Accuracy) = 464 / 484 ≈ 95.9%

模型在區分 DRUSEN 和 NORMAL 時非常準確，誤判比例很低

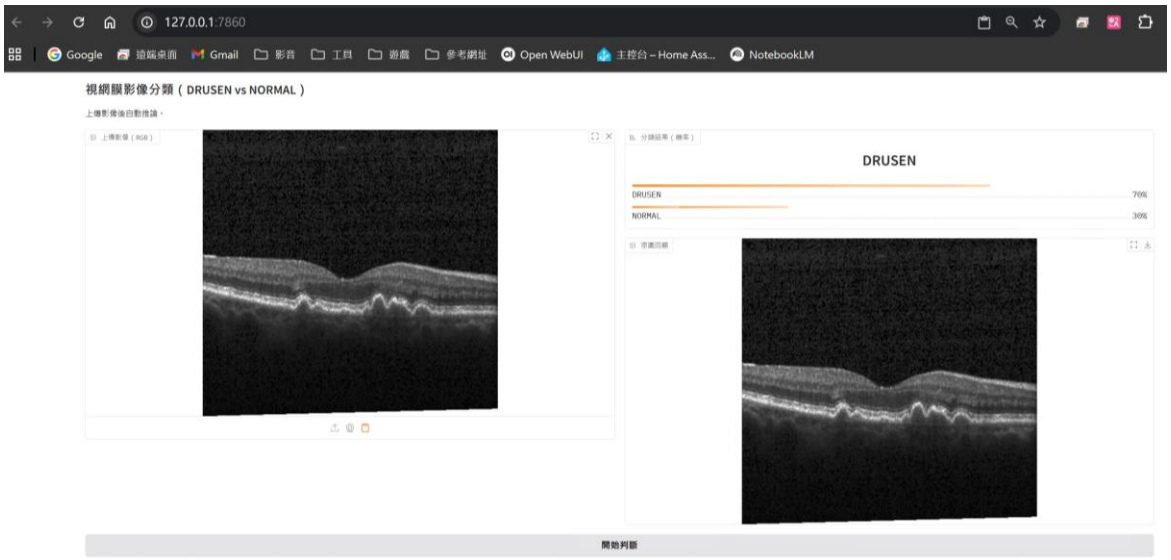
對正常眼底 的識別稍微比 DRUSEN 更準確。



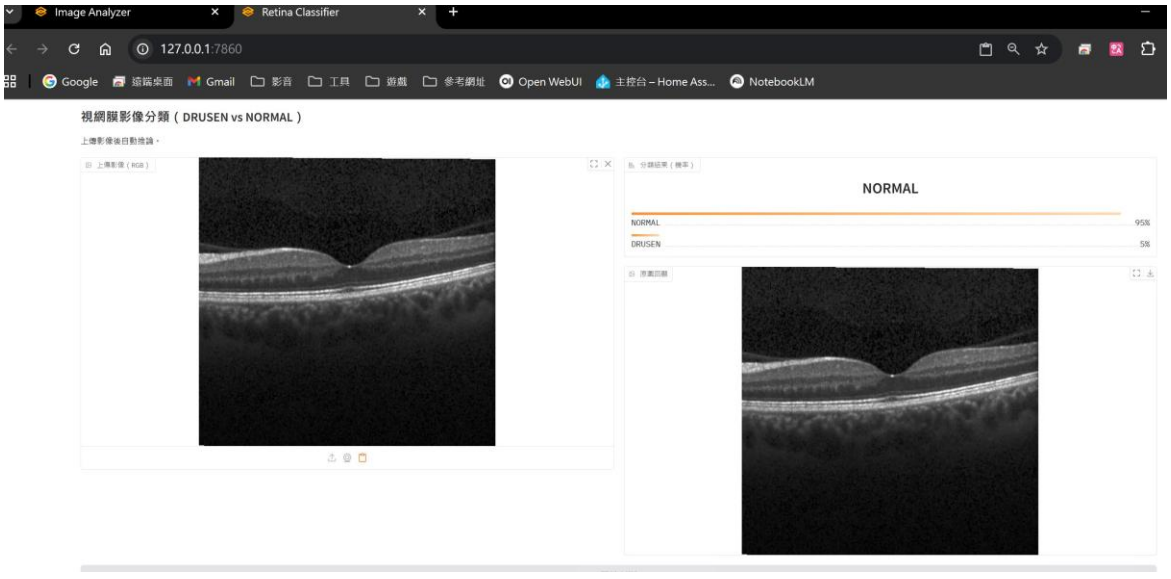


7. 新的一筆 sample 如何得到辨識的答案

用有疾病的照片測試得到的結果有 70%準確率



用正常視網膜的照片測試得到的結果有 95%準確率



8. 參考資料與參考網頁的連結

Kaggle: https://www.kaggle.com/datasets/paultimothymooney/kermany2018/data

Pytorch: https://pytorch.org/get-started/locally/

gradio: https://www.gradio.app/docs

Sklearn: https://scikit-learn.org/0.21/documentation.html