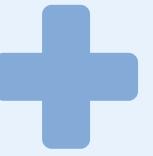
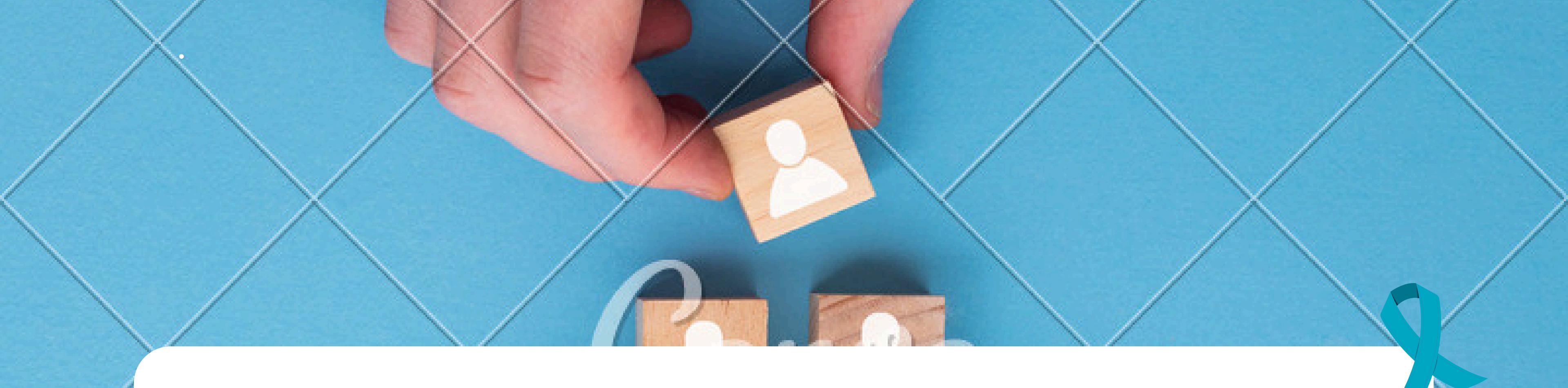


# SKIN CANCER CLASSIFICATION

Nhóm: 3





## Thành viên nhóm

Tô Minh Đức  
Nguyễn Minh Đăng  
Nguyễn Trường Giang  
Nguyễn Văn Thi  
Nguyễn Thị Thu Trang



# Mục lục

- 1. Giới thiệu**
- 2. Phân tích tập dữ liệu**
- 3. Tiền xử lý dữ liệu**
- 4. Các thuật toán phân loại**
- 5. Tuning model tốt nhất**
- 6. Kết luận**

# Tổng quan về dữ liệu

**Nguồn:** Kaggle

**Gồm:** 7 nhãn phân loại tổn thương da cùng thông tin tuổi, giới tính, vị trí cơ thể, mã ảnh.

**Quy mô:** 10 015 ảnh dermatoscopic, mỗi ảnh là một mẫu tổn thương da phục vụ nghiên cứu và huấn luyện mô hình phân loại ung thư da.





# Tổng quan về dữ liệu



Tên cột	Ý nghĩa	Đơn vị	Kiểu dữ liệu	Dạng dữ liệu
lesion_id	Định danh duy nhất cho mỗi tổn thương da (cùng bệnh nhân có thể có nhiều ảnh)	-	Chuỗi ký tự	Danh mục (ID)
image_id	Định danh duy nhất cho mỗi ảnh da	-	Chuỗi ký tự	Danh mục (ID)
dx	Loại bệnh được chẩn đoán (7 lớp): <code>akiec</code> , <code>bcc</code> , <code>bkl</code> , <code>df</code> , <code>nv</code> , <code>vasc</code> , <code>mel</code>	-	Chuỗi ký tự	Danh mục (categorical)
dx_type	Phương pháp chẩn đoán: <code>histopathology (histo)</code> , <code>follow-up (follow_up)</code> , <code>consensus</code> , <code>confocal</code>	-	Chuỗi ký tự	Danh mục (categorical)
age	Tuổi của bệnh nhân	Năm	Số nguyên	Số (numerical)
sex	Giới tính của bệnh nhân: <code>male</code> , <code>female</code>	-	Chuỗi ký tự	Danh mục (categorical)
localization	Vị trí của tổn thương trên cơ thể: ví dụ <code>back</code> , <code>chest</code> , <code>lower extremity</code> , <code>abdomen</code> , ...	-	Chuỗi ký tự	Danh mục (categorical)





# Tổng quan về dữ liệu

metadata.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10015 entries, 0 to 10014
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   lesion_id   10015 non-null   object  
 1   image_id    10015 non-null   object  
 2   dx          10015 non-null   object  
 3   dx_type     10015 non-null   object  
 4   age         9958 non-null   float64 
 5   sex         10015 non-null   object  
 6   localization 10015 non-null   object  
dtypes: float64(1), object(6)
memory usage: 547.8+ KB
```

metadata.unique()

lesion_id	7470
image_id	10015
dx	7
dx_type	4
age	18
sex	3
localization	15

metadata.shape

(10015, 7)

metadata.duplicated().sum()

0





# Tổng quan về dữ liệu



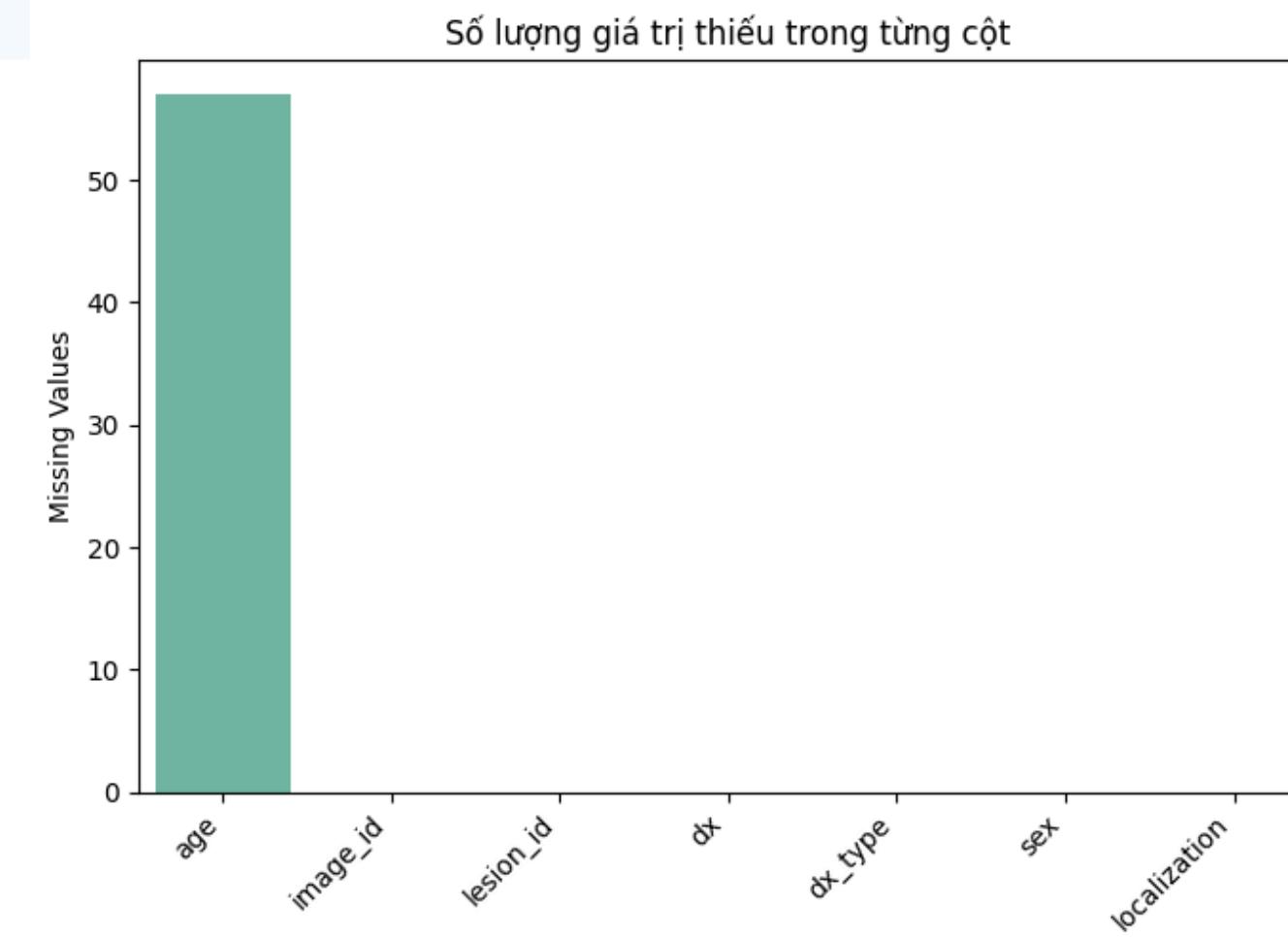
	lesion_id	image_id	dx	dx_type	age	sex	localization
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear





# Tổng quan về dữ liệu

	Missing Values	Percentage (%)
age	57	0.569146
lesion_id	0	0.000000
image_id	0	0.000000
dx	0	0.000000
dx_type	0	0.000000
sex	0	0.000000
localization	0	0.000000

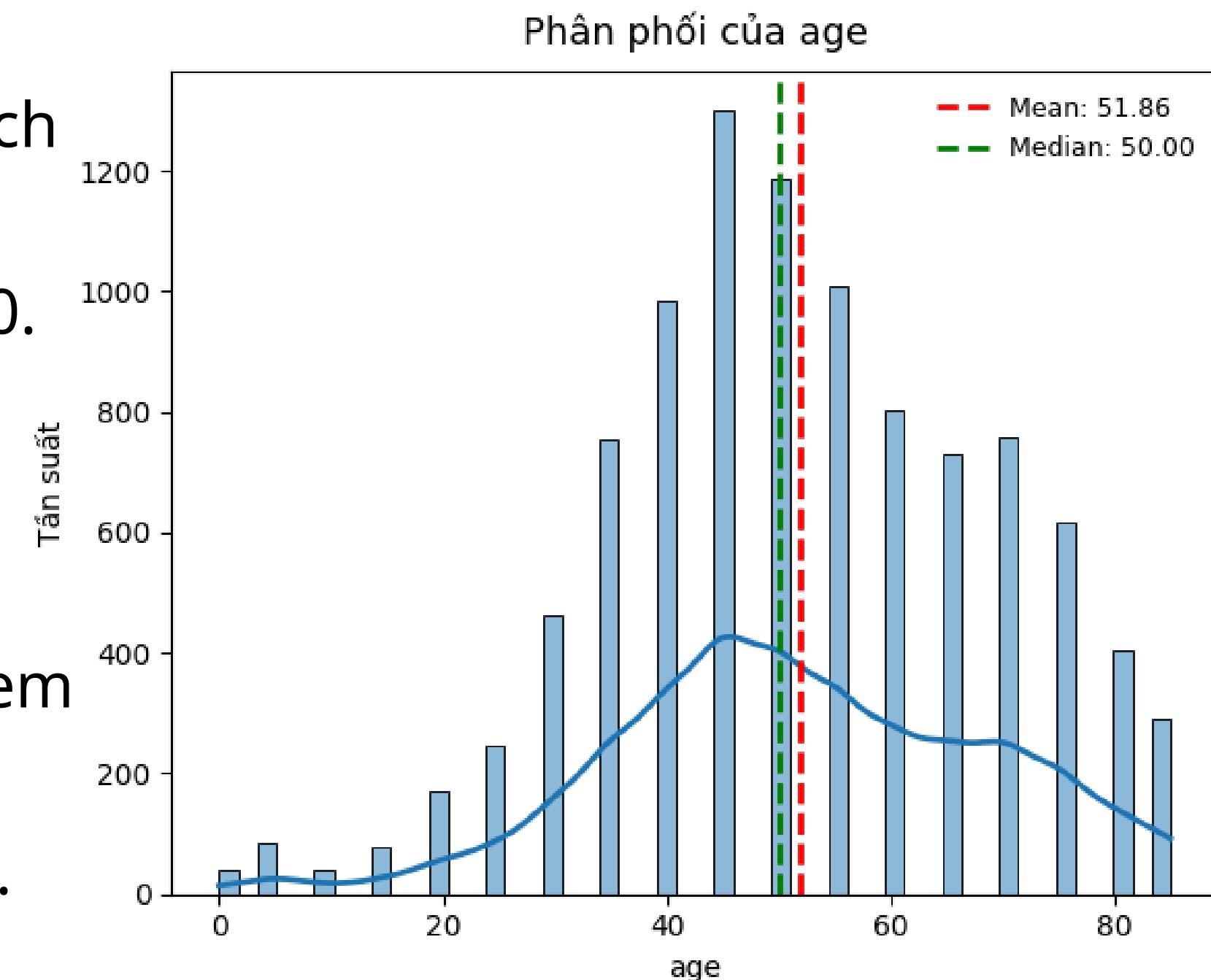


- Chỉ có cột age có giá trị thiếu, với số lượng 57 giá trị, chiếm khoảng 0.57%.
- Các cột còn lại như lesion\_id, image\_id, dx, dx\_type, sex, localization không có giá trị thiếu.



# Thống kê mô tả và trực quan hóa cột số

- **Phân phối tuổi:** Gần đối xứng, hơi lệch phải; tập trung nhiều ở 45–55 tuổi.
- **Thống kê:** Mean  $\approx 51.86$ , median = 50.
- **Khoảng tuổi:** 0–85+, rất ít mẫu  $<20$  hoặc  $>80$ .
- **Ý nghĩa:** Dữ liệu nghiêng về nhóm trung niên/cao tuổi; thiếu mẫu ở trẻ em và người rất già có thể ảnh hưởng độ chính xác dự đoán cho các nhóm này.

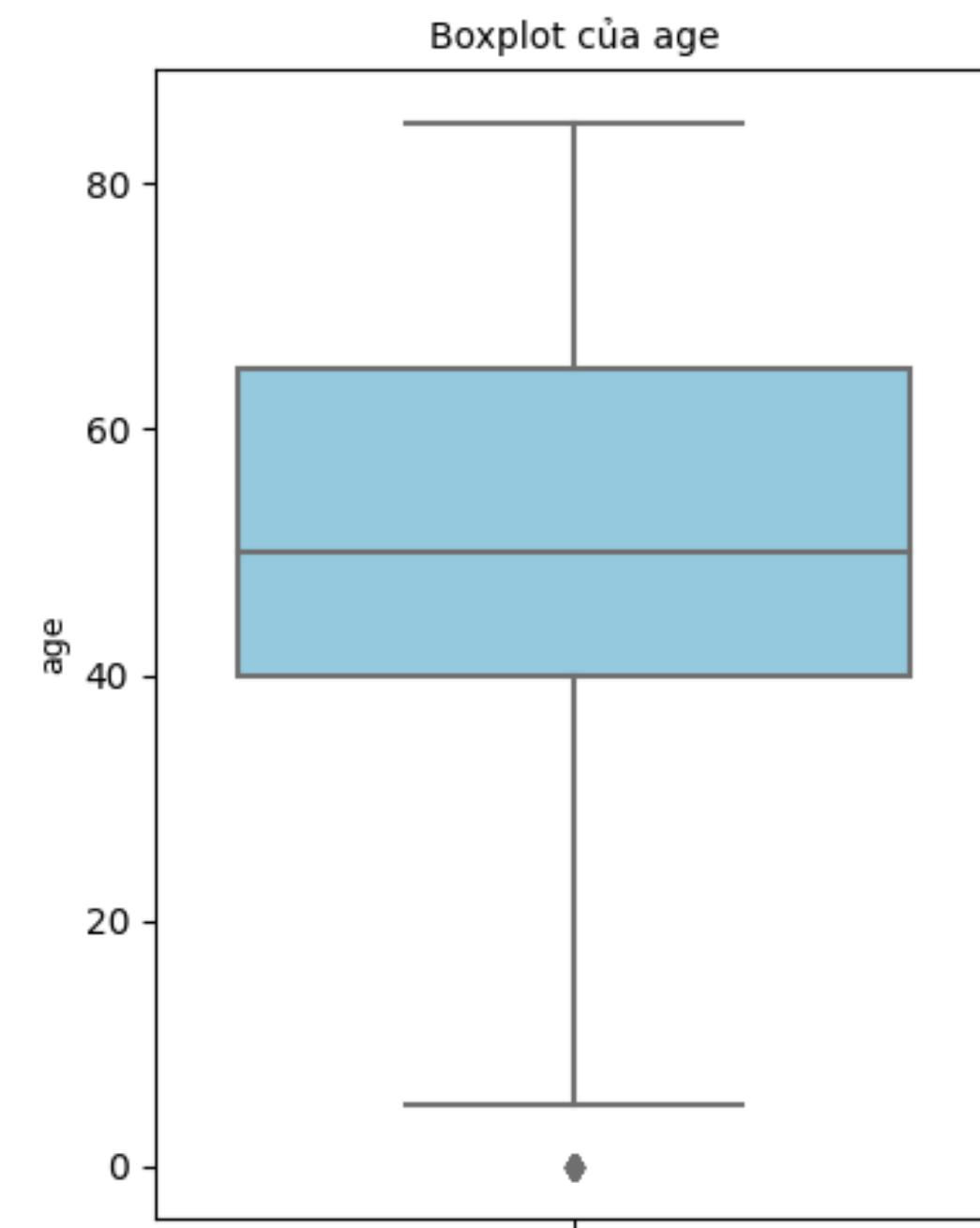




# Thống kê mô tả và trực quan hóa cột số

Ở cột age, có 39 giá trị được xác định là outlier, tuy nhiên do tất cả đều tập trung tại cùng một mức tuổi rất nhỏ nên trên boxplot chỉ thể hiện dưới dạng một điểm ngoại lai duy nhất.

Thuộc tính	Số lượng outliers
age	39

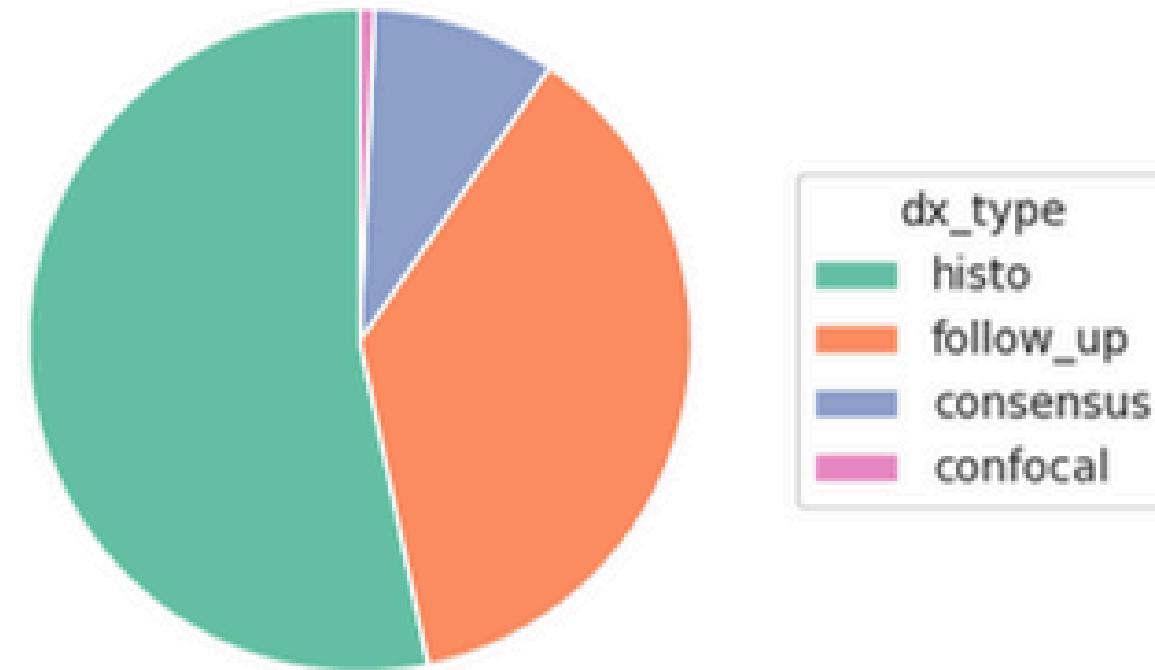




# Thống kê mô tả và trực quan hóa cột phân loại

Biểu đồ tròn: dx\_type

dx_type	Số lượng	Tỷ lệ (%)
histo	5340	53.32
follow_up	3704	36.98
consensus	902	9.01
confocal	69	0.69



**Xác nhận nhãn:** Histo 53.32% (tiêu chuẩn vàng), Follow\_up 36.98%, Consensus 9.01%, Confocal 0.69%.

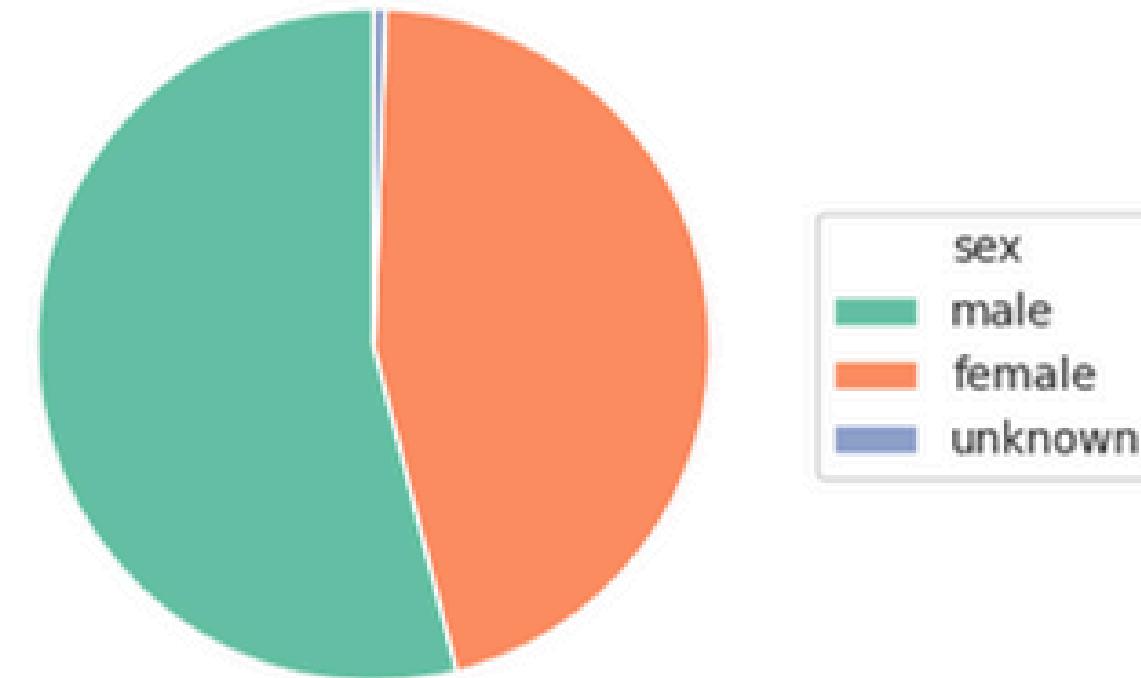
**Nhận xét:** Histo chiếm ưu thế, đảm bảo tin cậy; dữ liệu đa dạng phương pháp. Consensus & Confocal rất ít, nên gộp để giảm mất cân bằng.



# Thống kê mô tả và trực quan hóa cột phân loại

Biểu đồ tròn: sex

sex	Số lượng	Tỷ lệ (%)
male	5406	53.98
female	4552	45.45
unknown	57	0.57



**Giới tính:** Nam 53.98%, Nữ 45.45%, Unknown 0.57%.

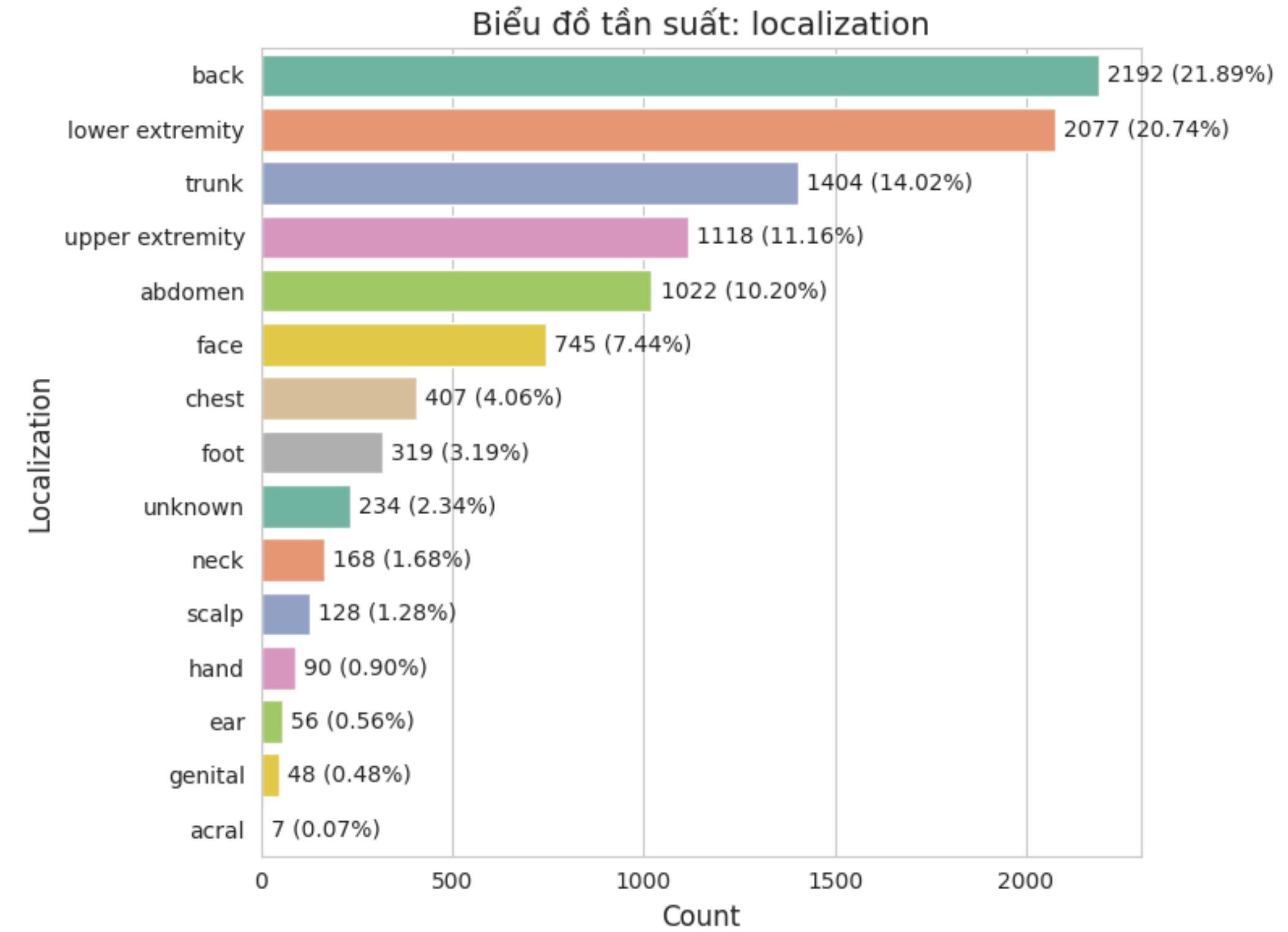
**Nhận xét:** Phân bố giới tính khá cân bằng → giảm nguy cơ thiên lèch;  
Unknown rất ít, có thể giữ hoặc loại bỏ.





# Thống kê mô tả và trực quan hóa cột phân loại

	Số lượng	Tỷ lệ (%)
Localization		
back	2192	21.89
lower extremity	2077	20.74
trunk	1404	14.02
upper extremity	1118	11.16
abdomen	1022	10.20
face	745	7.44
chest	407	4.06
foot	319	3.19
unknown	234	2.34
neck	168	1.68
scalp	128	1.28
hand	90	0.90
ear	56	0.56
genital	48	0.48
acral	7	0.07



# Thống kê mô tả và trực quan hóa cột phân loại

**Áp đảo nhất:** Lưng: 21.89%, chi dưới: 20.74% → Hai vị trí này chiếm gần 43% toàn bộ dữ liệu.

**Nhóm tỉ lệ trung bình** (10-15%): Thân : 14.02%, chi trên: 11.16%, bụng: 10.20%

**Nhóm ít hơn** (< 8%): Mặt, ngực, bàn chân, vùng không xác định, cổ, da đầu...

**Hiếm gặp** (< 1%): Bàn tay, tai, bộ phận sinh dục, acral.

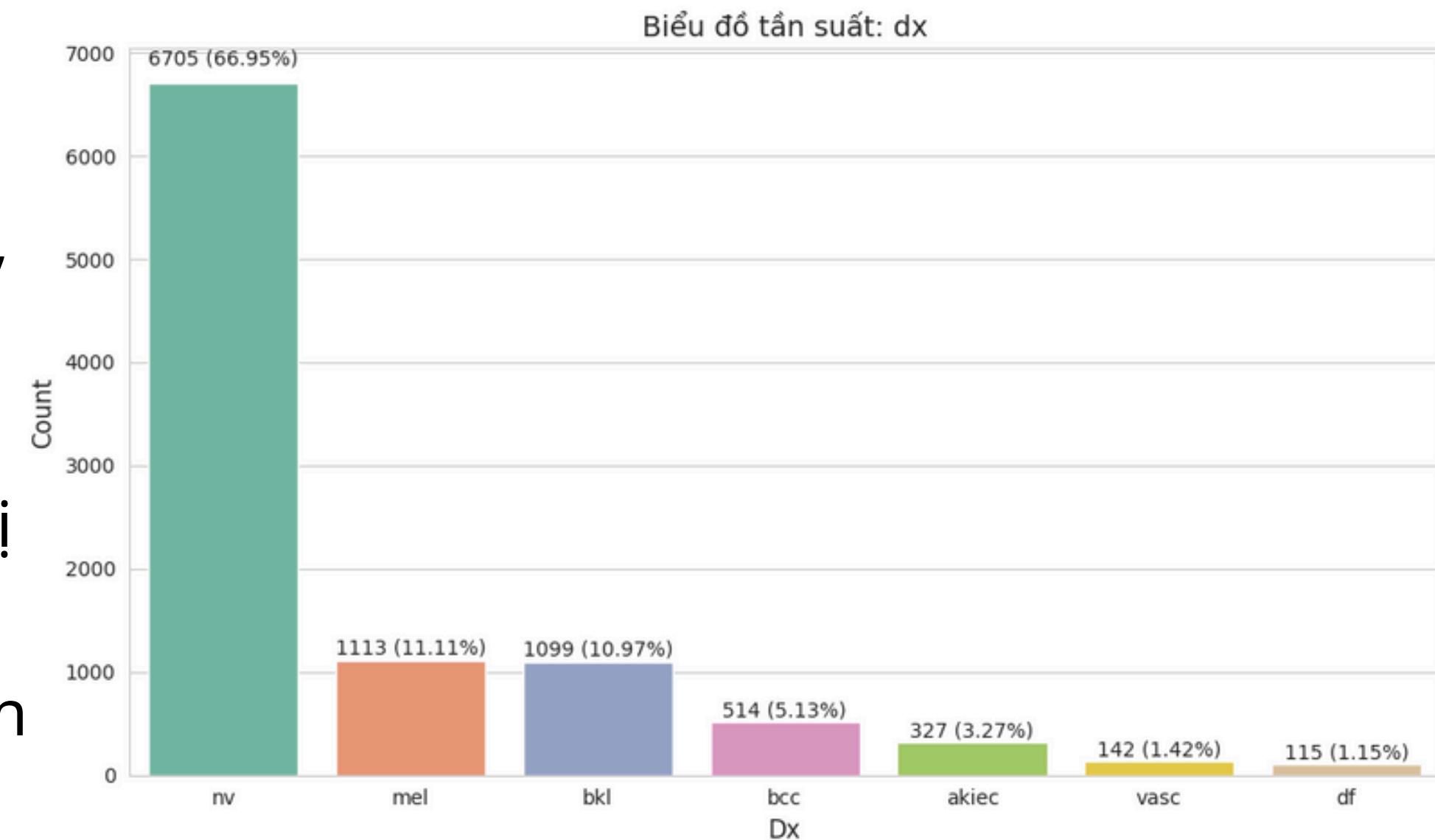
**Phân bố không đồng đều:** Rất nhiều mẫu tập trung ở vùng lưng và chi dưới, trong khi các vị trí như acral, genital, ear cực kỳ ít → nguy cơ mất cân bằng dữ liệu khi huấn luyện mô hình phân loại hoặc nhận dạng vị trí.

**Khả năng thiên lệch mô hình:** Mô hình có thể học tốt cho nhóm phổ biến, nhưng dự đoán kém cho nhóm hiếm.



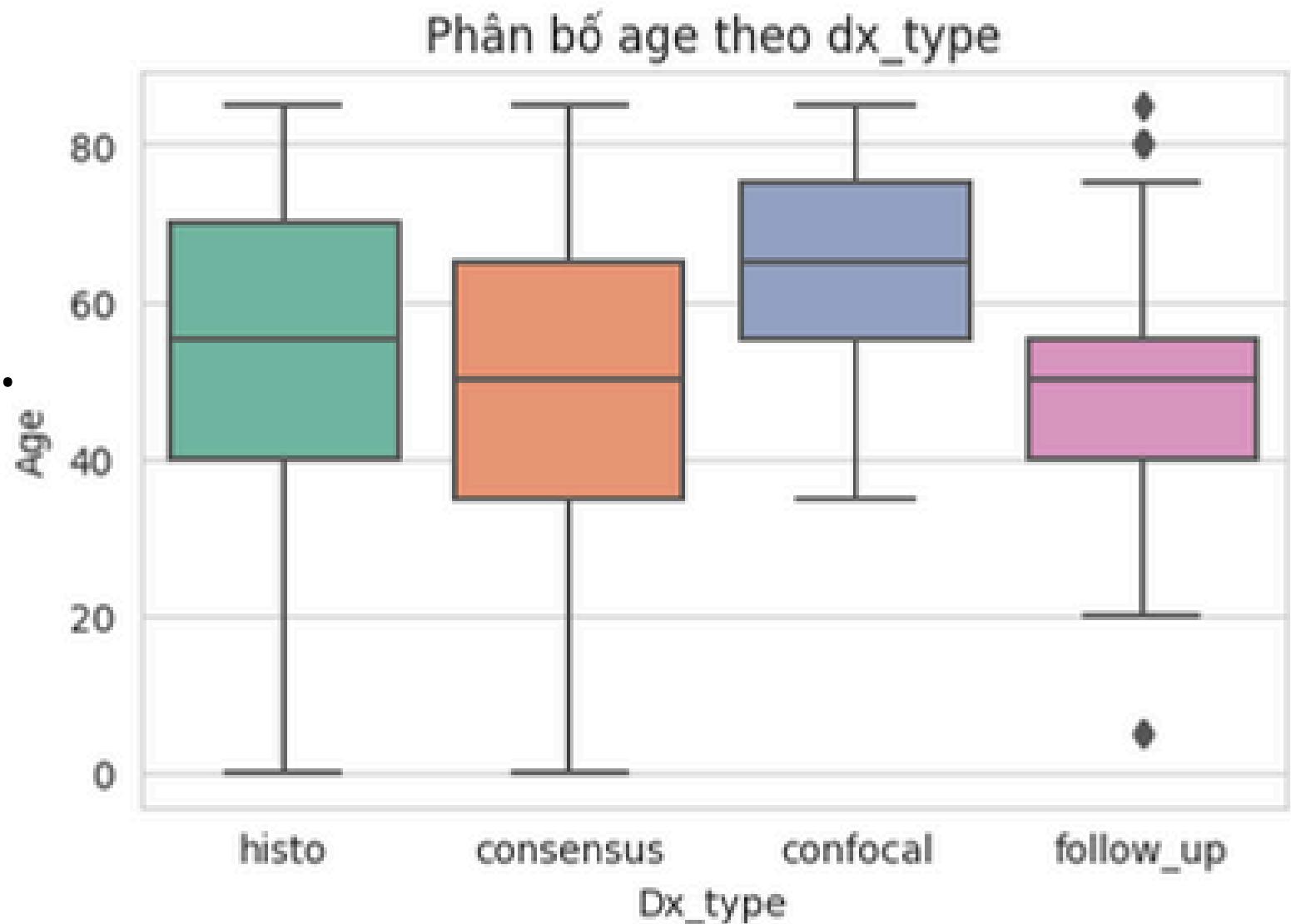
# Thống kê mô tả và trực quan hóa cột nhãn

- **Tỉ lệ lớp:** nv 66.95% (áp đảo), mel 11.11%, bkl 10.97%, bcc 5.13%, akiec 3.27%, vasc 1.42%, df 1.15%.
- **Nhận xét:** Mất cân bằng mạnh (nv gấp ~60× df); lớp hiếm dễ bị bỏ qua → cần chiến lược sampling để cải thiện nhận diện bệnh hiếm.



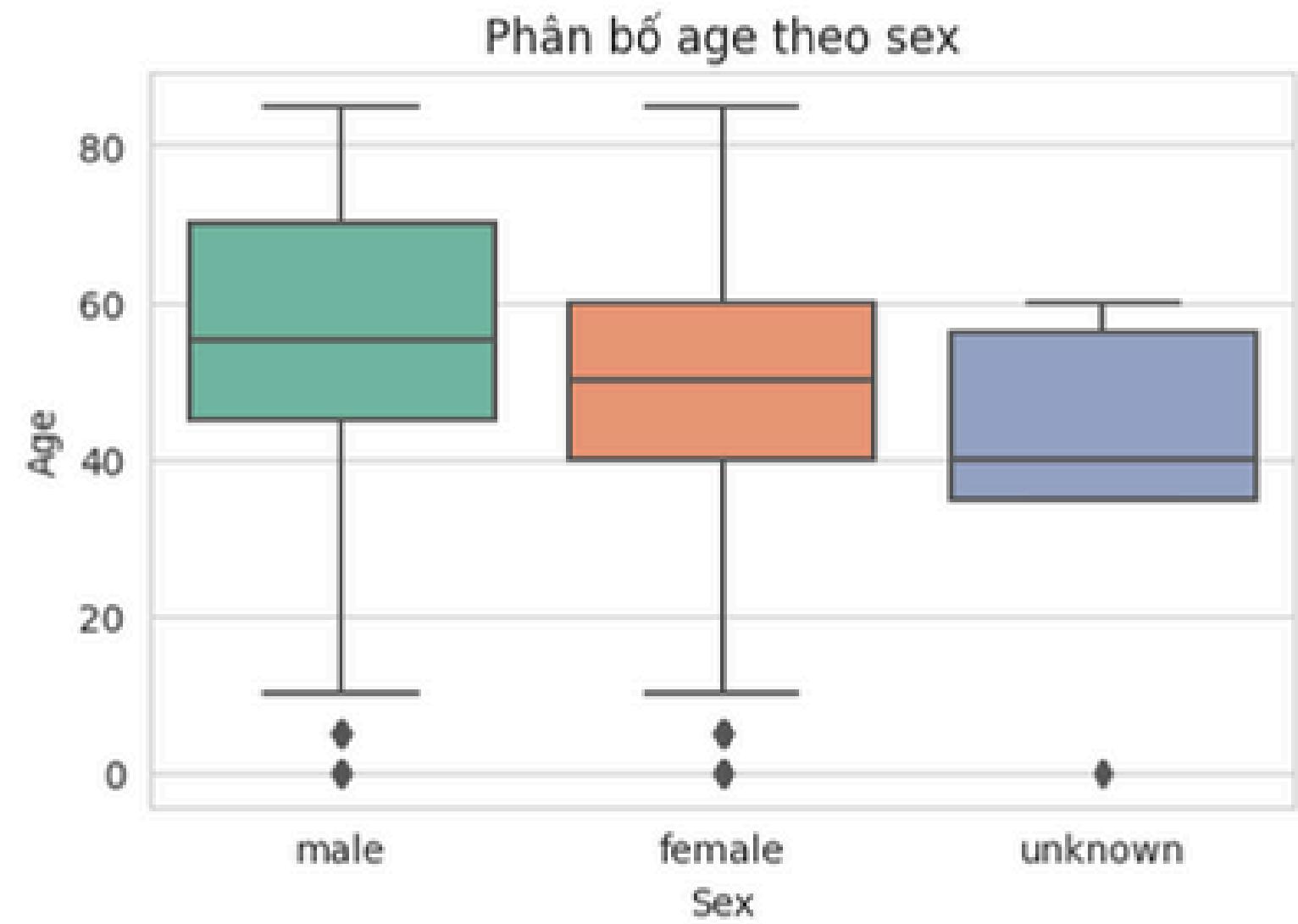
# Mối quan hệ giữa cột số và cột phân loại

- **Histo:** Phân bố 0–85 tuổi, median ~55, áp dụng mọi nhóm tuổi.
- **Consensus:** Phân bố rộng, median ~50.
- **Confocal:** Chủ yếu 35+, median ~65; dùng cho ca phức tạp, tránh xâm lấn.
- **Follow\_up:** Median ~50, có vài ca <10 tuổi.



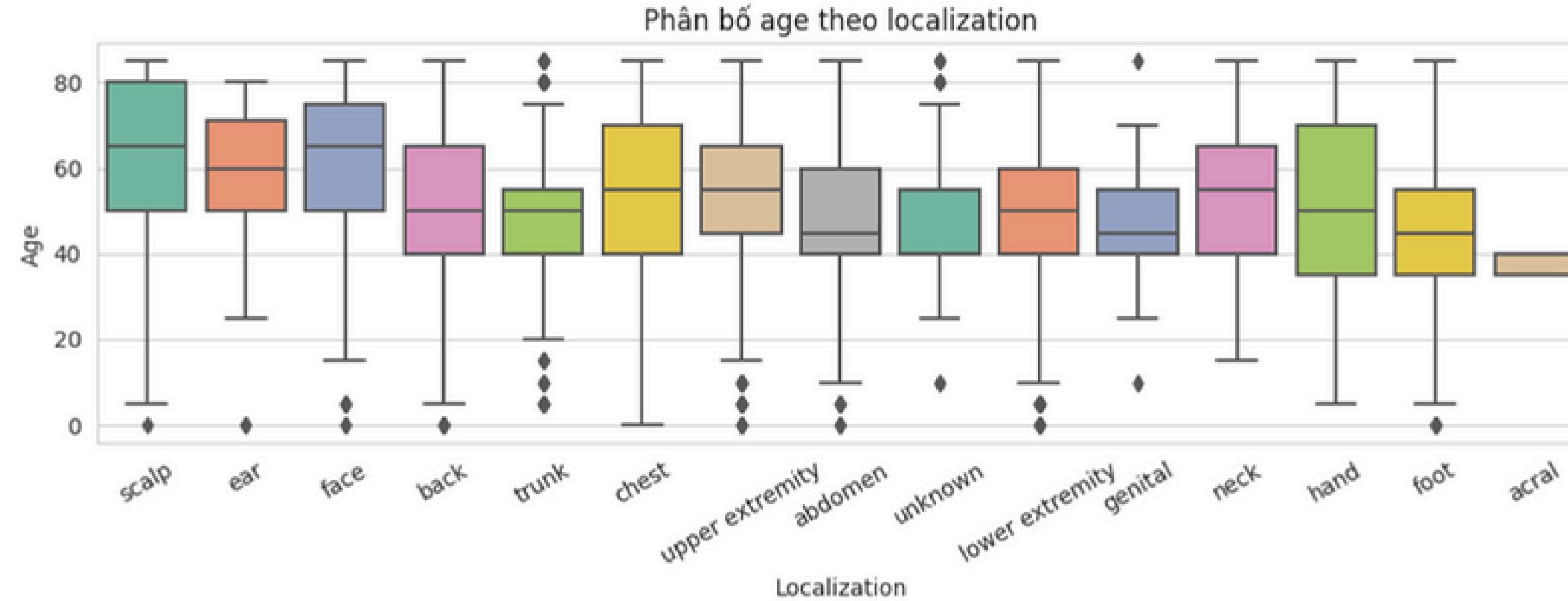
# Mối quan hệ giữa cột số và cột phân loại

- **Nam:** Median ~55, phân bố rộng, có cả <5 tuổi.
- **Nữ:** Median ~50, phân tán hẹp, ít >80.
- **Unknown:** Chủ yếu 35–60 tuổi, ít biến động.





# Mối quan hệ giữa cột số và cột phân loại



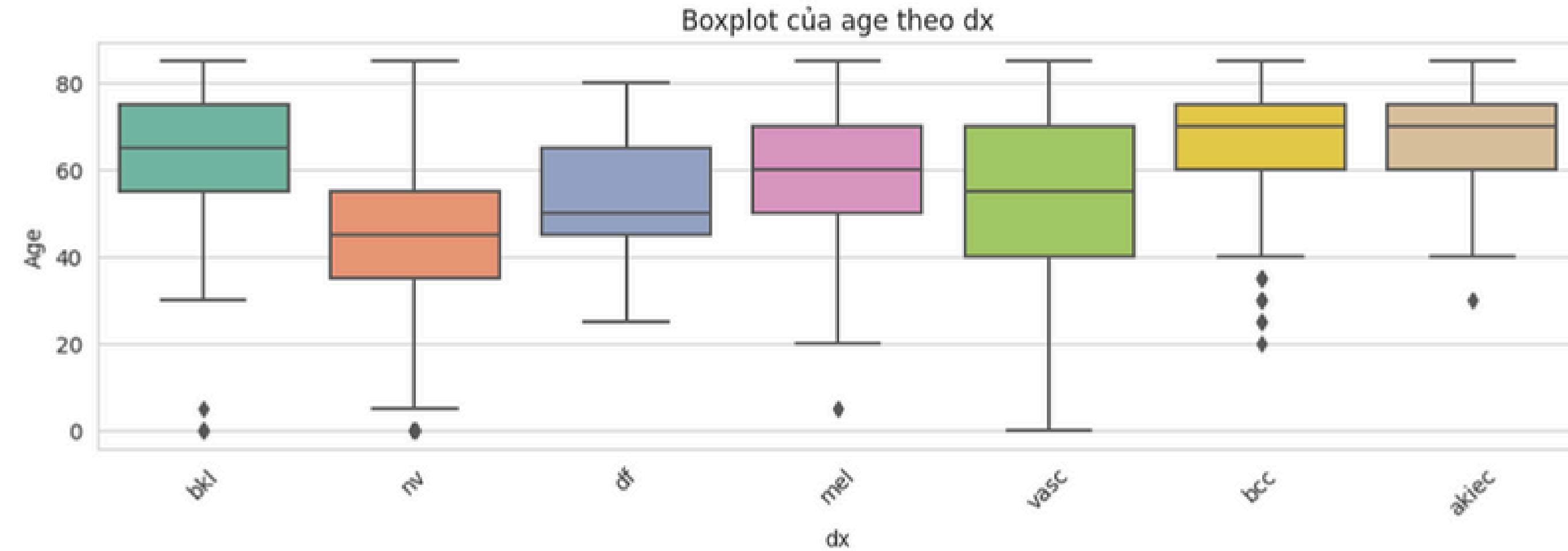
**Ear, face, scalp, trunk:** Median ~60–65, lệch về tuổi cao (tổn thương do ánh nắng tích lũy).

**Back, abdomen, lower/upper extremity, chest:** Median ~50–55.

**Genital, acral:** Median ~40–45, phân bố hẹp, dữ liệu ít và không đa dạng.



# Mối quan hệ giữa cột số và nhãn

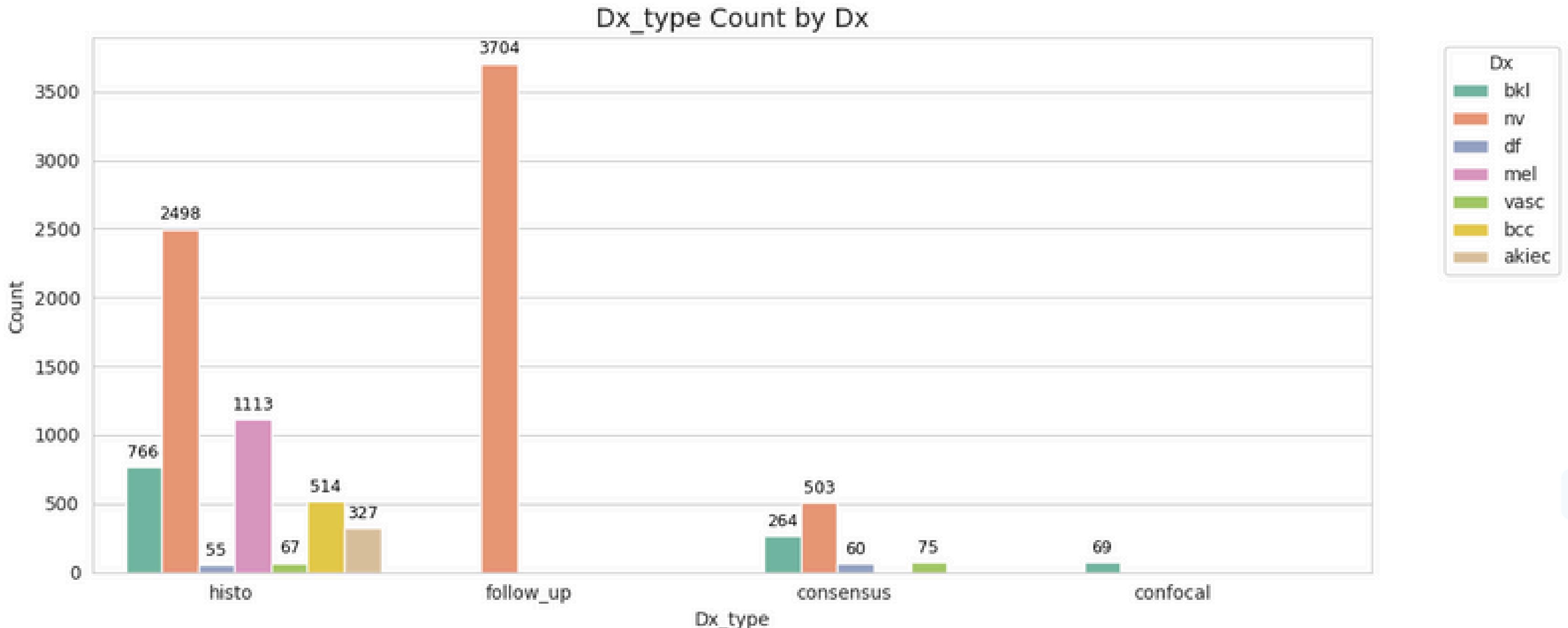


**Lành tính:** nv 45–50 (cả trẻ), df ~50, vasc ~50, bkl 65–70.

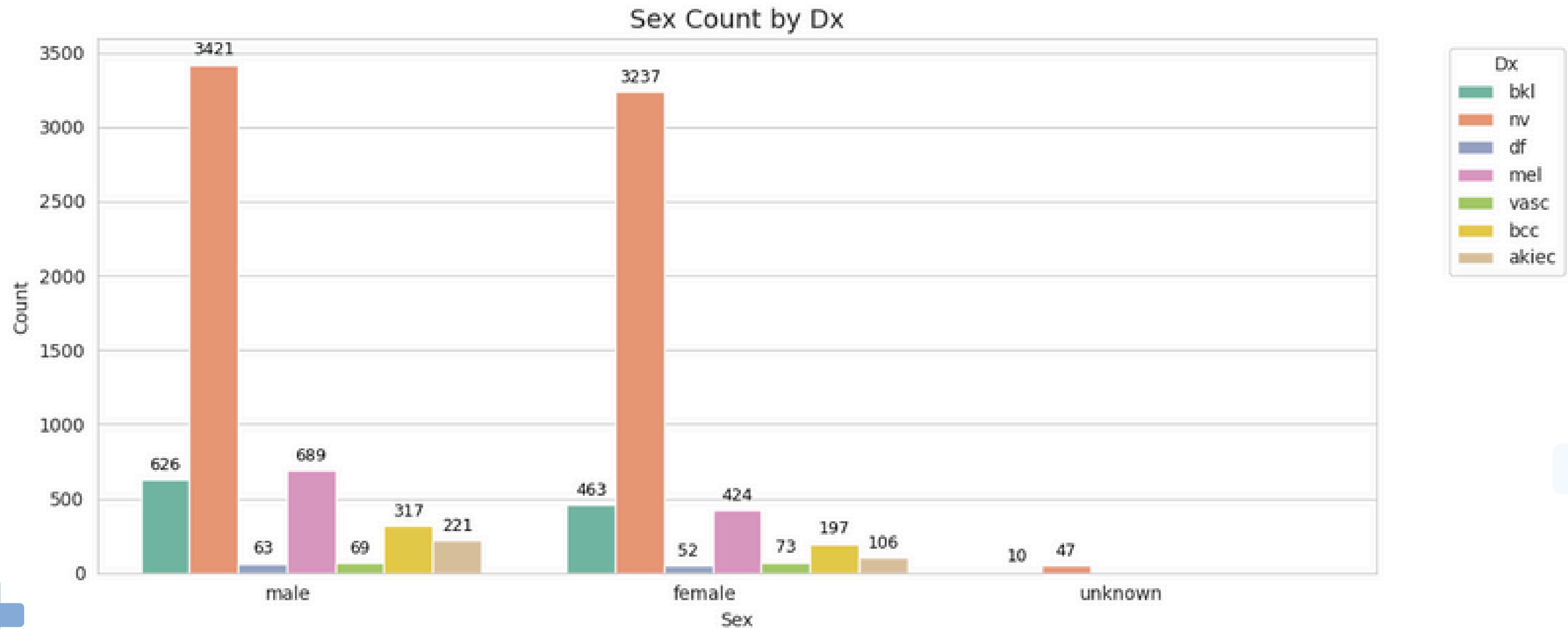
**Ác tính/tiền ung thư:** mel ~60, bcc & akiec ~70.

**Xu hướng:** Lành tính phân bố rộng, ác tính tập trung tuổi cao.

# Mối quan hệ giữa cột phân loại và nhãn



# Mối quan hệ giữa cột phân loại và nhãn



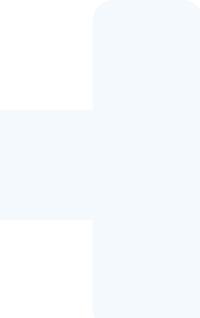


# Mối quan hệ giữa cột phân loại và nhãn

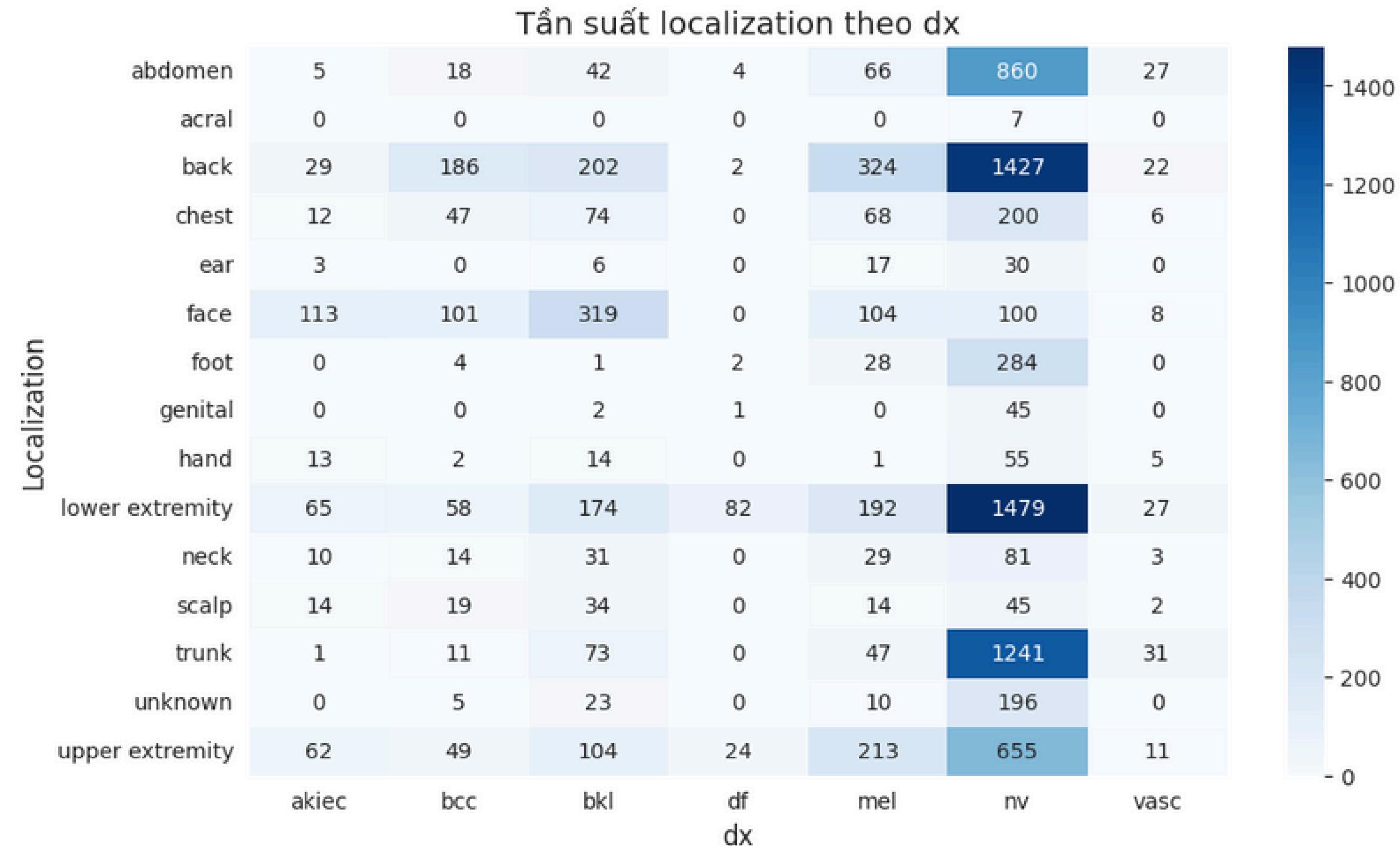
---

**Dx\_type vs Dx:** nv áp đảo hầu hết Dx\_type (follow\_up 3,704, histo 2,498); df, vasc, bcc, akiec ít và phân bố không đều. Mel nhiều ở histo (1,113), confocal chỉ có bkl (69) → dữ liệu mất cân bằng, cần cân nhắc khi huấn luyện.

**Sex vs Dx:** nv áp đảo cả nam (3,421) & nữ (3,237); mel nhiều hơn ở nam (689 vs 424). Một số bệnh ít & phân bố không đều giữa giới → cần lưu ý phân tích theo giới để tránh bias.



# Mối quan hệ giữa cột phân loại và nhãn



# Mối quan hệ giữa cột phân loại và nhãns

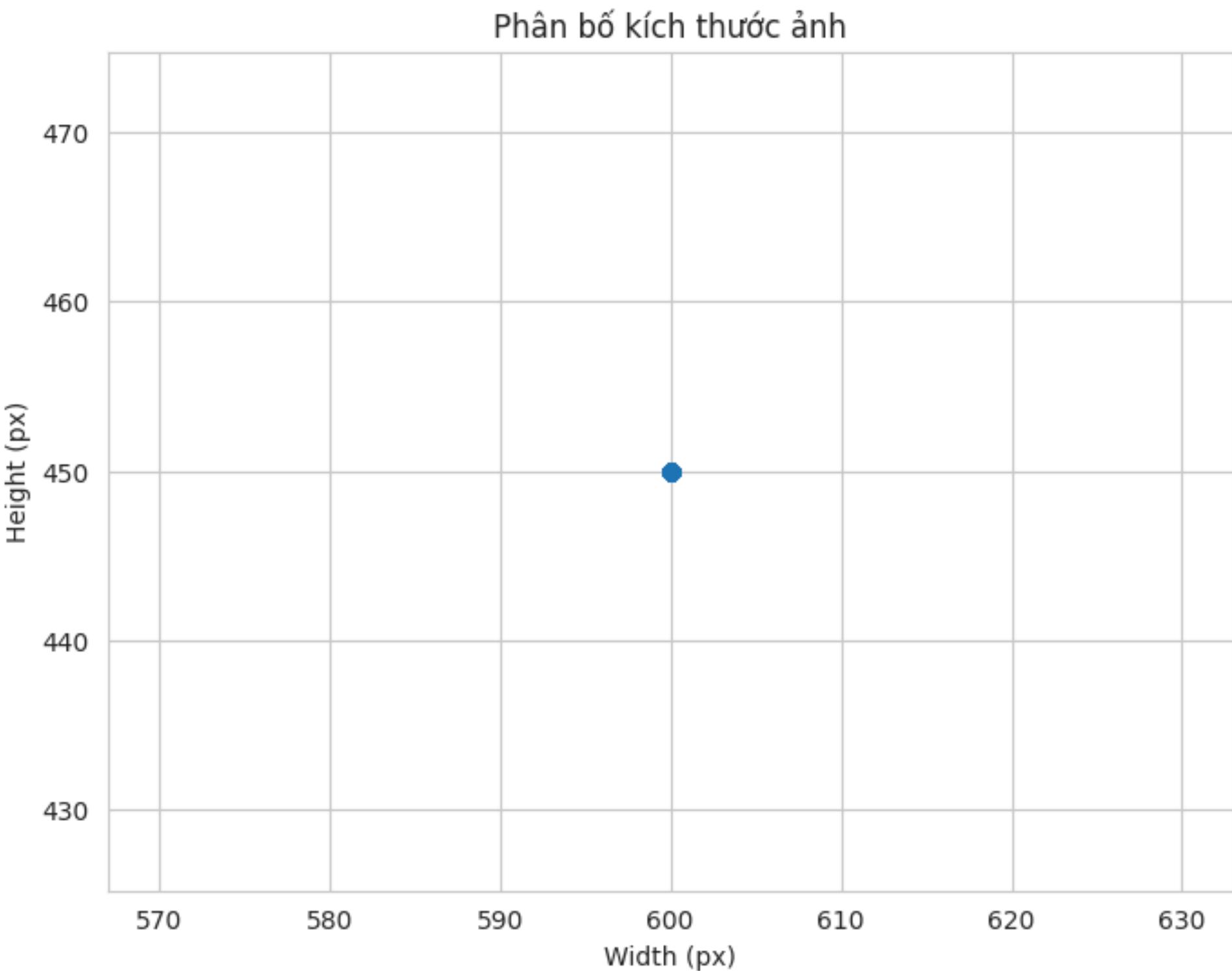
---

- **Số lượng cao:** Back & lower extremity nhiều ca nv (1,427 & 1,479), trunk cũng cao (1,241).
- **Bệnh nổi bật theo vị trí:** bcc nhiều ở back (186) & face (101), bkl ở back (202) & face (319), mel phân bố rộng, nhiều nhất back (324) & lower extremity (192).
- **Ít xuất hiện:** Acral, ear, genital, unknown rất hiếm.
- **Tổng quan:** nv chiếm ưu thế ở nhiều vị trí; các bệnh khác tập trung vùng tiếp xúc ánh sáng.

# Kiểm tra kích thước ảnh

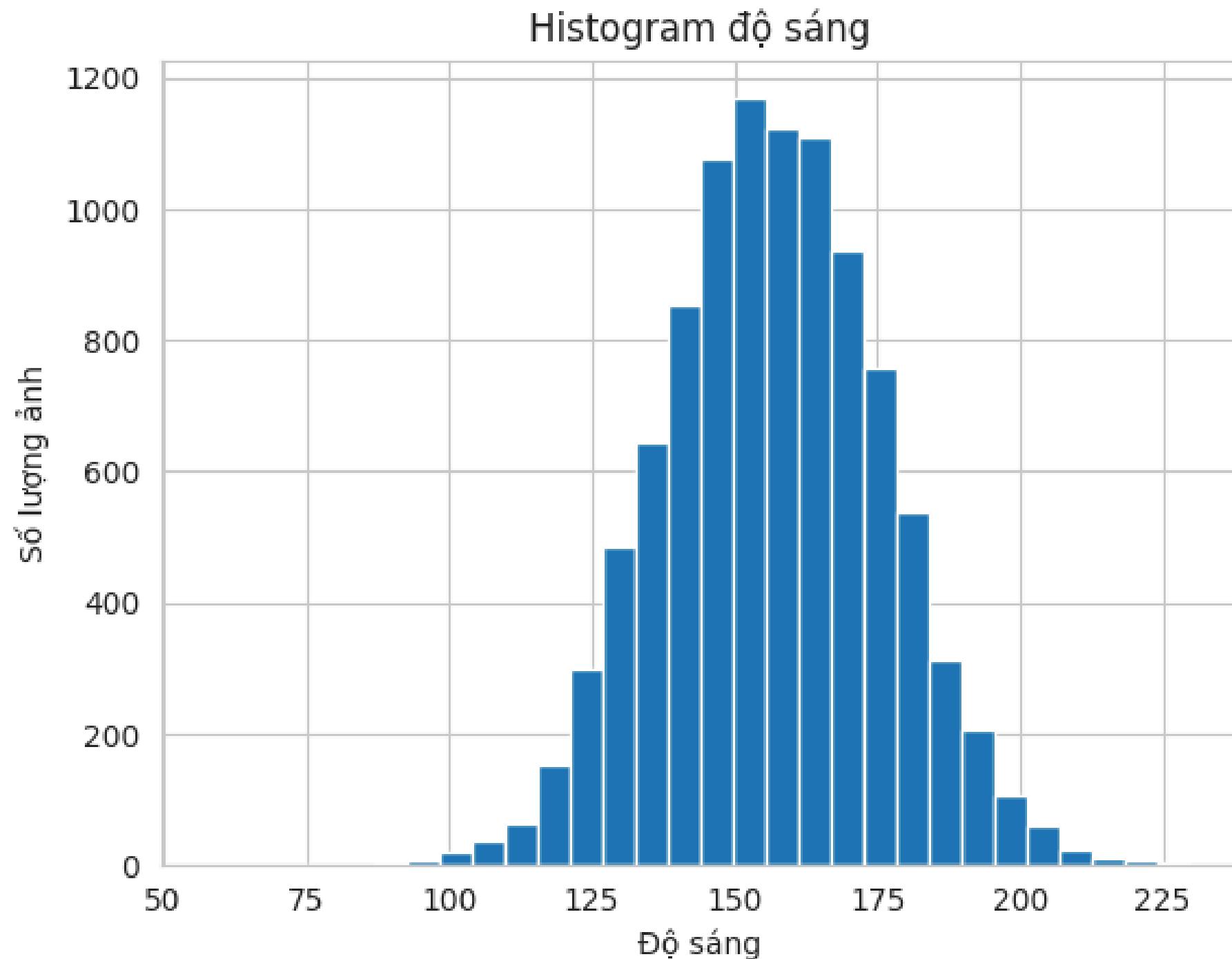
	width	height	ratio
<b>count</b>	10015.0	10015.0	1.001500e+04
<b>mean</b>	600.0	450.0	1.333333e+00
<b>std</b>	0.0	0.0	1.598801e-13
<b>min</b>	600.0	450.0	1.333333e+00
<b>25%</b>	600.0	450.0	1.333333e+00
<b>50%</b>	600.0	450.0	1.333333e+00
<b>75%</b>	600.0	450.0	1.333333e+00
<b>max</b>	600.0	450.0	1.333333e+00

Tất cả các ảnh đều có  
kích thước 600x450



# Kiểm tra chất lượng ảnh

- **Độ sáng ảnh:** Phân bố gần chuẩn, đối xứng, đỉnh ở 150–155; phần lớn ảnh 130–170.
- **Ý nghĩa:** Độ sáng đồng đều, hầu hết ảnh sáng tốt, ít bị tối hoặc cháy sáng. Một số outlier với  $>10\%$  pixel tối hoặc  $>20\%$  pixel sáng có thể gây nhiễu khi huấn luyện.



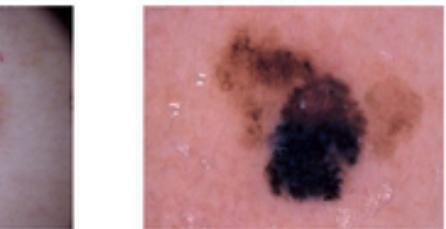
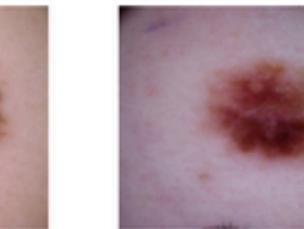


# Hiển thị ảnh mẫu mỗi lớp

Actinic keratoses



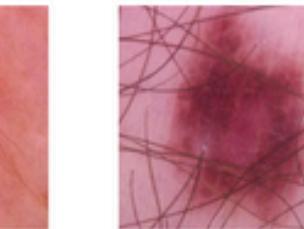
Melanoma



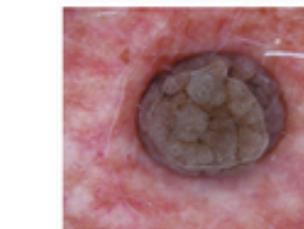
Basal cell carcinoma



Melanocytic nevi



Benign keratosis-like lesions



Vascular lesions



Dermatofibroma



Melanoma



# Histogram pixel (phân phối giá trị màu)

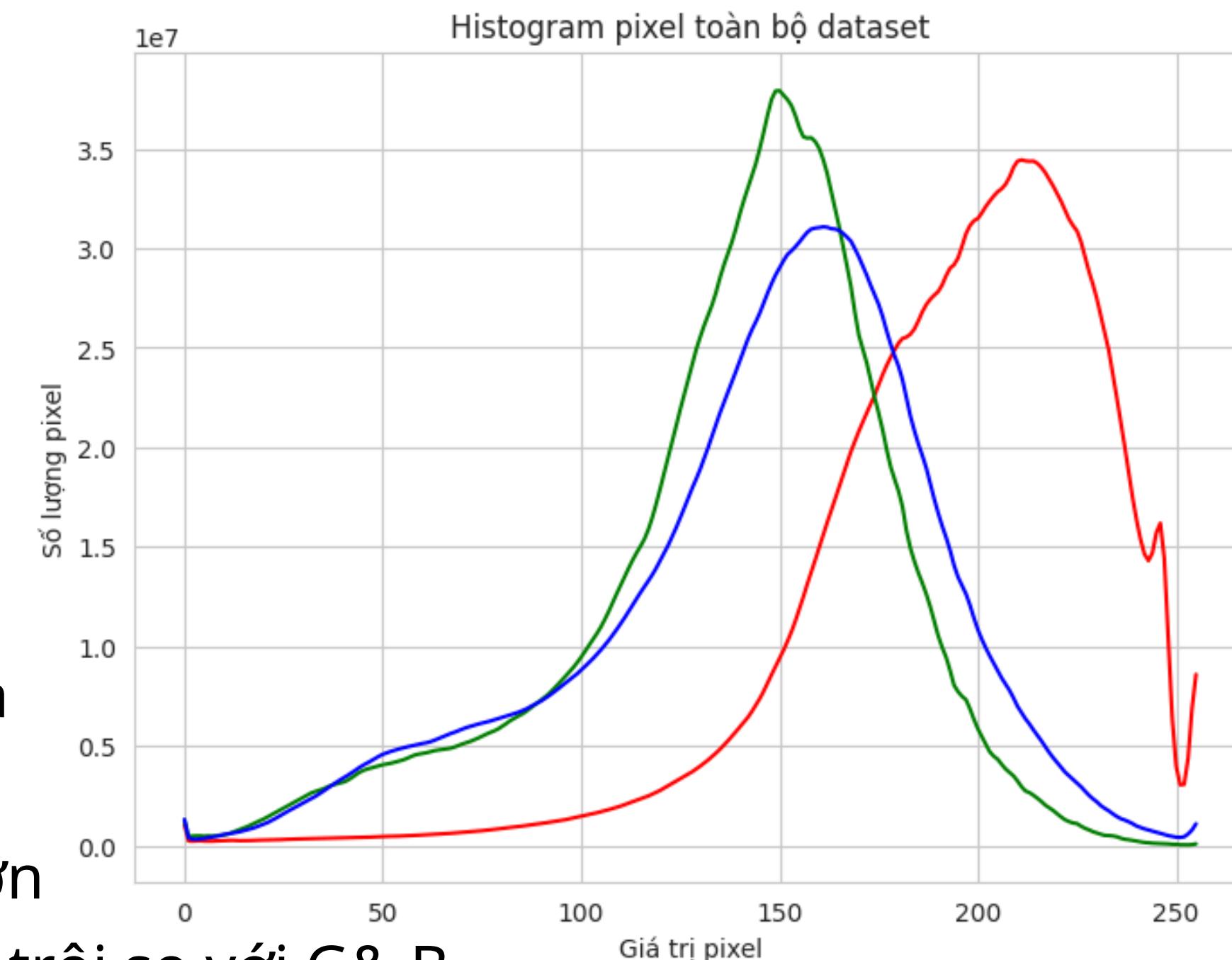
```
modes = metadata['path'].apply(  
    lambda p: Image.open(p).mode  
)  
display(modes.value_counts())
```

```
path  
RGB      10015  
Name: count, dtype: int64
```

**R**: đỉnh ~200–230, thiên đỏ/ấm.

**G & B**: tập trung ~140–160, sáng vừa phải, ít bão hòa.

**Tổng quan**: Ít pixel tối (<50), phần lớn 100–230; cả ba kênh lệch phải, R nổi trội so với G&B.



# Kiểm tra ảnh bị lỗi

```
bad_images = []
for p in metadata['path']:
    try:
        with Image.open(p) as img:
            img.verify()
    except:
        bad_images.append(p)

print(f"Tổng ảnh lỗi: {len(bad_images)}")
if bad_images:
    print(bad_images)
```

Tổng ảnh lỗi: 0

# Kiểm tra ảnh trùng/near-duplicate

```
import imagehash
from collections import defaultdict

hash_to_paths = defaultdict(list)
for p in metadata['path']:
    with Image.open(p) as im:
        h = imagehash.phash(im.convert('RGB'))
    hash_to_paths[str(h)].append(p)

dups = {h:ps for h,ps in hash_to_paths.items() if len(ps) > 1}
print("Cụm ảnh trùng/na ná nhau:", len(dups))

# Kiểm tra trùng nhưng khác nhãn
path_to_dx = dict(zip(metadata['path'], metadata['dx']))
suspicious = {h:[(pp, path_to_dx[pp]) for pp in ps] for h,ps in dups.items()}
```

Cụm ảnh trùng/na ná nhau: 19

# Tính class weights

```
from sklearn.utils.class_weight import compute_class_weight

classes = np.unique(metadata['dx'])
weights = compute_class_weight('balanced', classes=classes, y=metadata['dx'])
class_weights = dict(zip(classes, weights))
print("Class weights: ", class_weights)

Class weights: { 'akiec': 4.375273044997815, 'bcc': 2.78349082823791, 'bkl': 1.30183283504484
6, 'df': 12.440993788819876, 'mel': 1.2854575792581184, 'nv': 0.21338020666879728, 'vasc': 1
0.075452716297788}
```

# Data Preprocessing

```
# Age scaling
scaler = MinMaxScaler()
metadata['age_scaled'] = scaler.fit_transform(metadata[['age']].fillna(metadata['age'].median()))

# Encoding
sex_ohe = pd.get_dummies(metadata['sex'].fillna('unknown'), prefix='sex')
loc_ohe = pd.get_dummies(metadata['localization'].fillna('unknown'), prefix='localization')
le_dx = LabelEncoder()
metadata['dx_type_encoded'] = le_dx.fit_transform(metadata['dx_type'])
le_dx = LabelEncoder()
metadata['dx_encoded'] = le_dx.fit_transform(metadata['dx'])
```

```
data_transforms = {
    'train': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}
```



# Data Preprocessing

---

- **Với metadata :**
  - Giá trị thiếu: cột số (age) điền giá trị median
  - Mã hóa:
    - sex, localization: One-Hot Encoding
    - dx\_type, dx: Label Encoding
- **Với ảnh:**
  - Tập train:
    - Resize (224×224), RandomHorizontalFlip (augmentation)
    - Chuyển thành tensor, chuẩn hóa theo ImageNet
  - Tập val:
    - Resize & chuẩn hóa như train
    - Không áp dụng augmentation để đảm bảo khách quan





# Chiến lược chọn mô hình

---

## Bước 1 – Sweep 10 epoch (đánh giá nhanh):

- Mô hình thử: Resnet50, Resnet18, Efficientnet\_b0, VGG16\_bn, Densenet121, Mobilenet\_v2.
- Thiết lập giống nhau: LR=1e-4, Adam, batch=32, chia train-val-test với tỉ lệ 70/15/15
- Chọn hai metric chính: Loss và Accuracy
- So sánh trong 2 trường hợp: chỉ có ảnh và khi có cả ảnh và metadata

## Bước 2 – Fine-tune 50 epoch (mô hình thắng):

- Unfreeze toàn bộ hoặc unfreeze theo tầng (Layer-wise LR decay).
- Tối ưu: AdamW, CosineAnnealing, EarlyStopping (patience 8-10), MixUp/CutMix.
- Theo dõi: F1-macro, Recall melanoma, Loss, LR schedule.



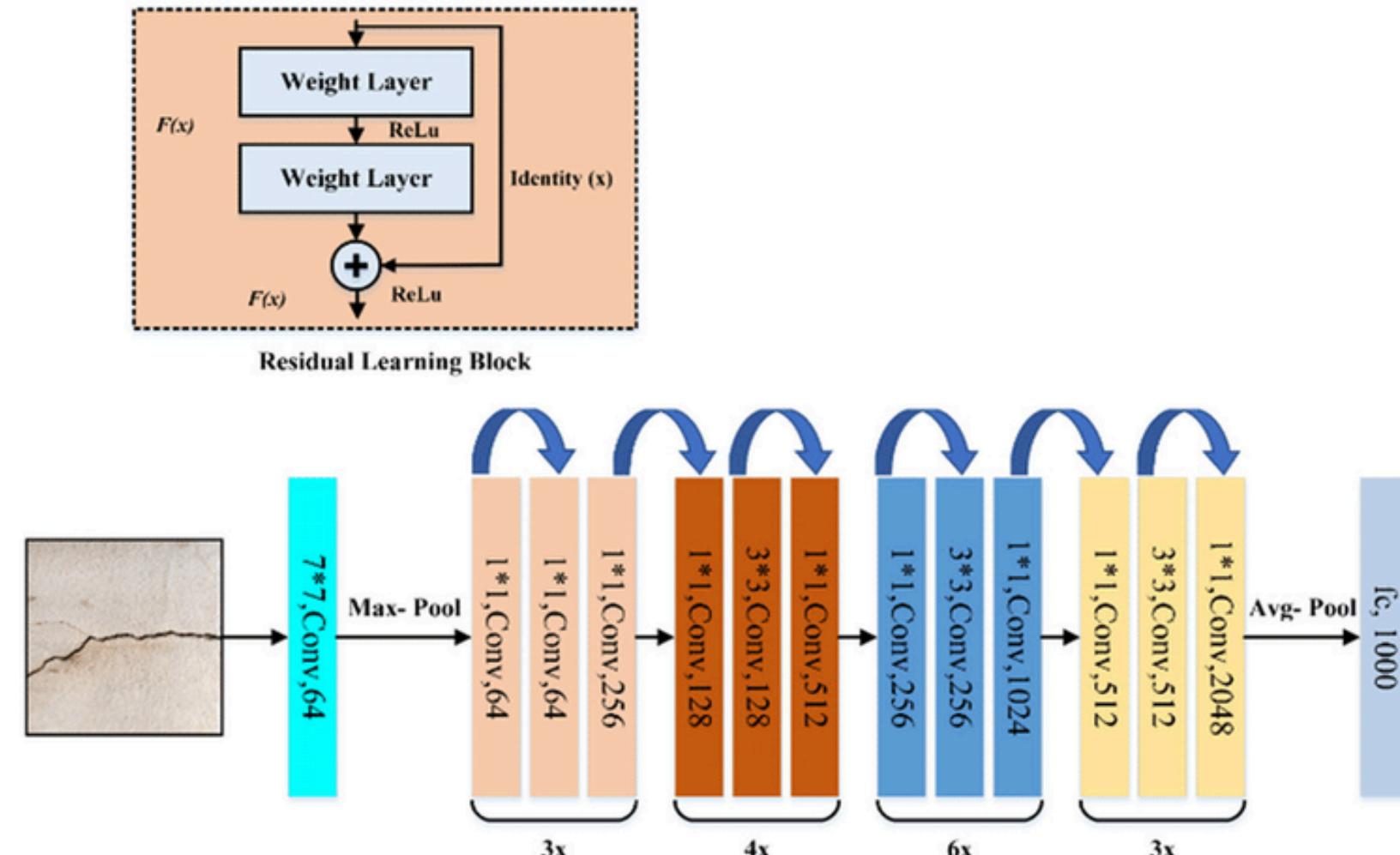
# ResNet50



# ResNet50

ResNet50 đạt độ chính xác cao và huấn luyện ổn định, nhưng mô hình mới như EfficientNet hay Vision Transformer thường vượt trội hơn về hiệu suất hoặc tối ưu tài nguyên.

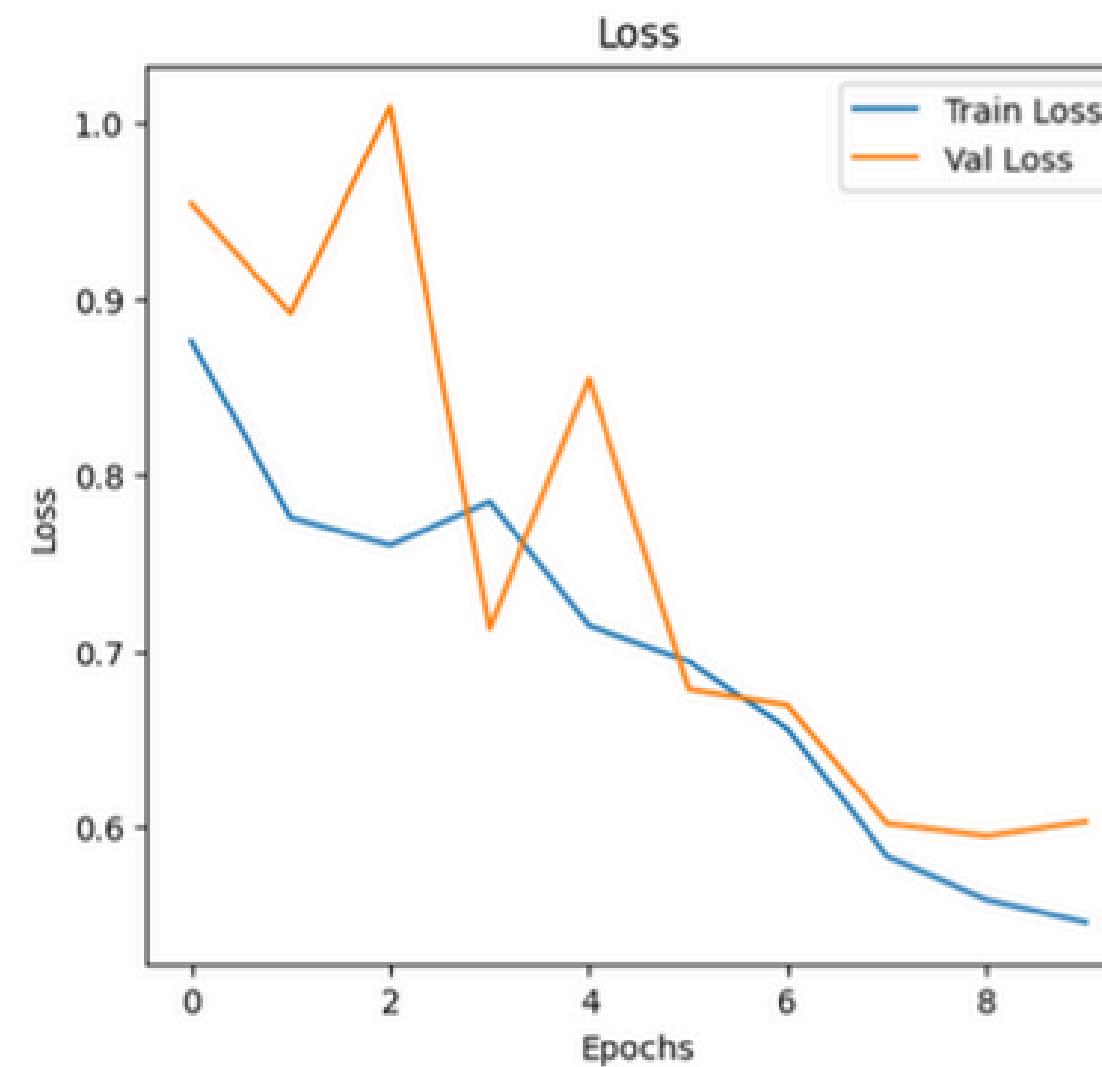
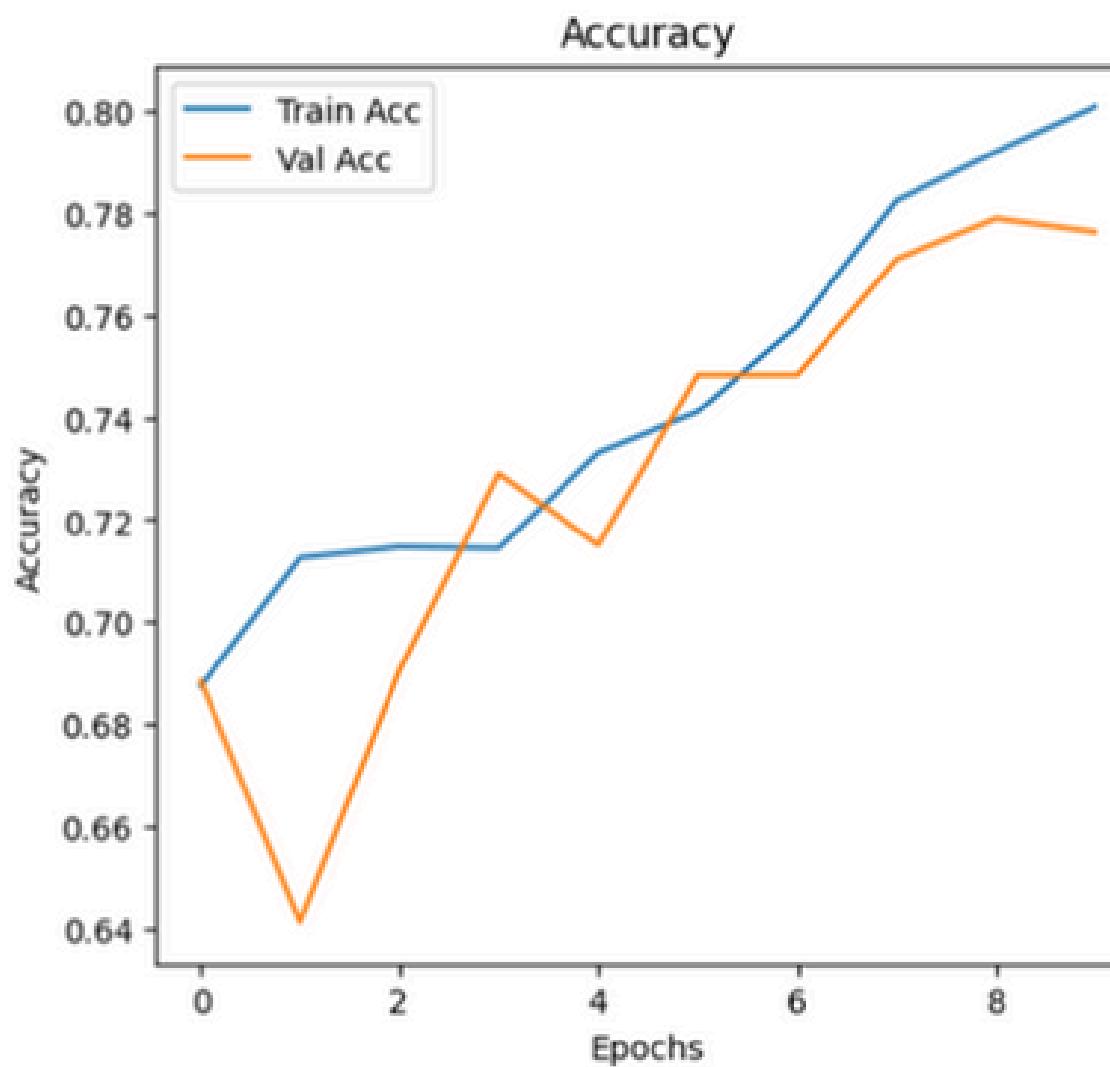
```
model = models.resnet50(weights=models.ResNet50_Weights.IMGNET1K_V2)
model.fc = nn.Linear(model.fc.in_features, len(train_dataset.classes))
```





# ResNet50 - Model Ánh

```
train Loss: 0.5457 Acc: 0.8009  
val Loss: 0.6031 Acc: 0.7763
```



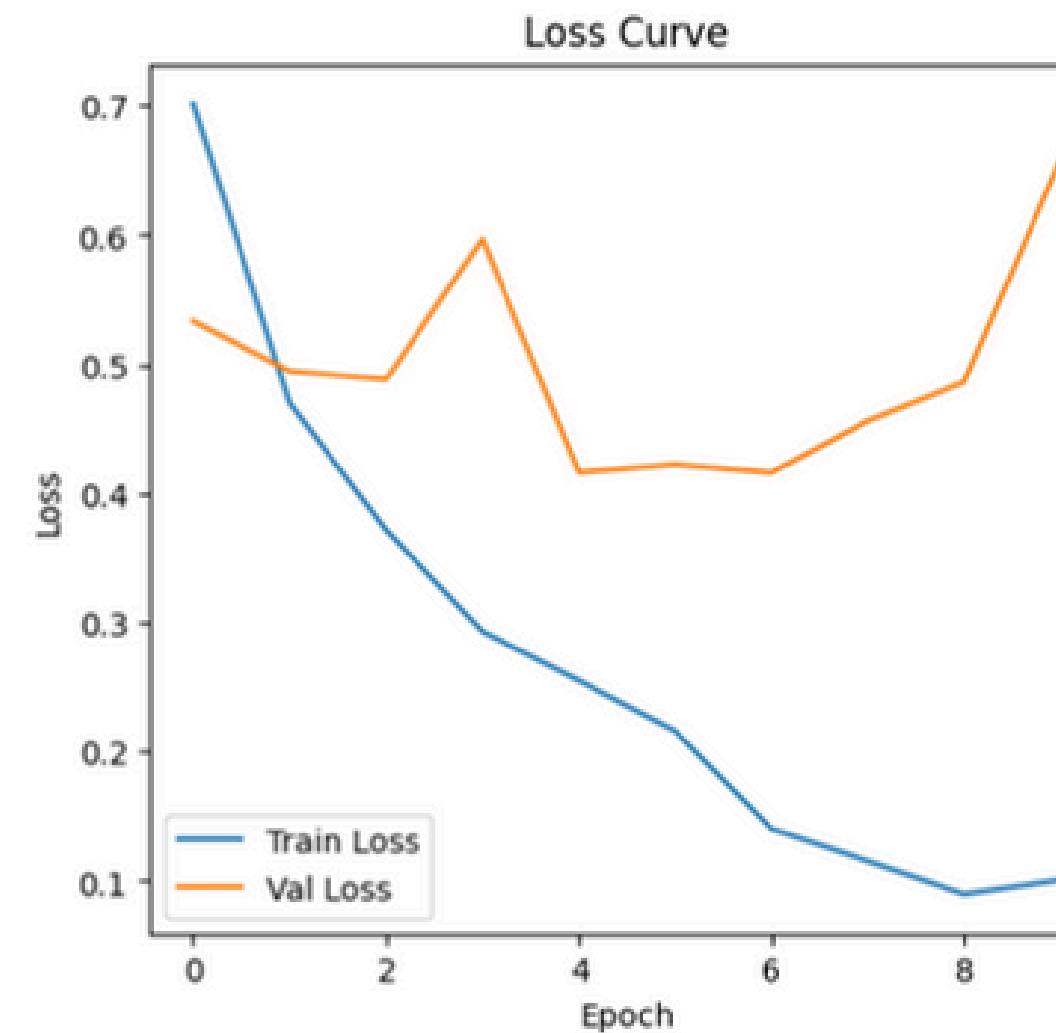
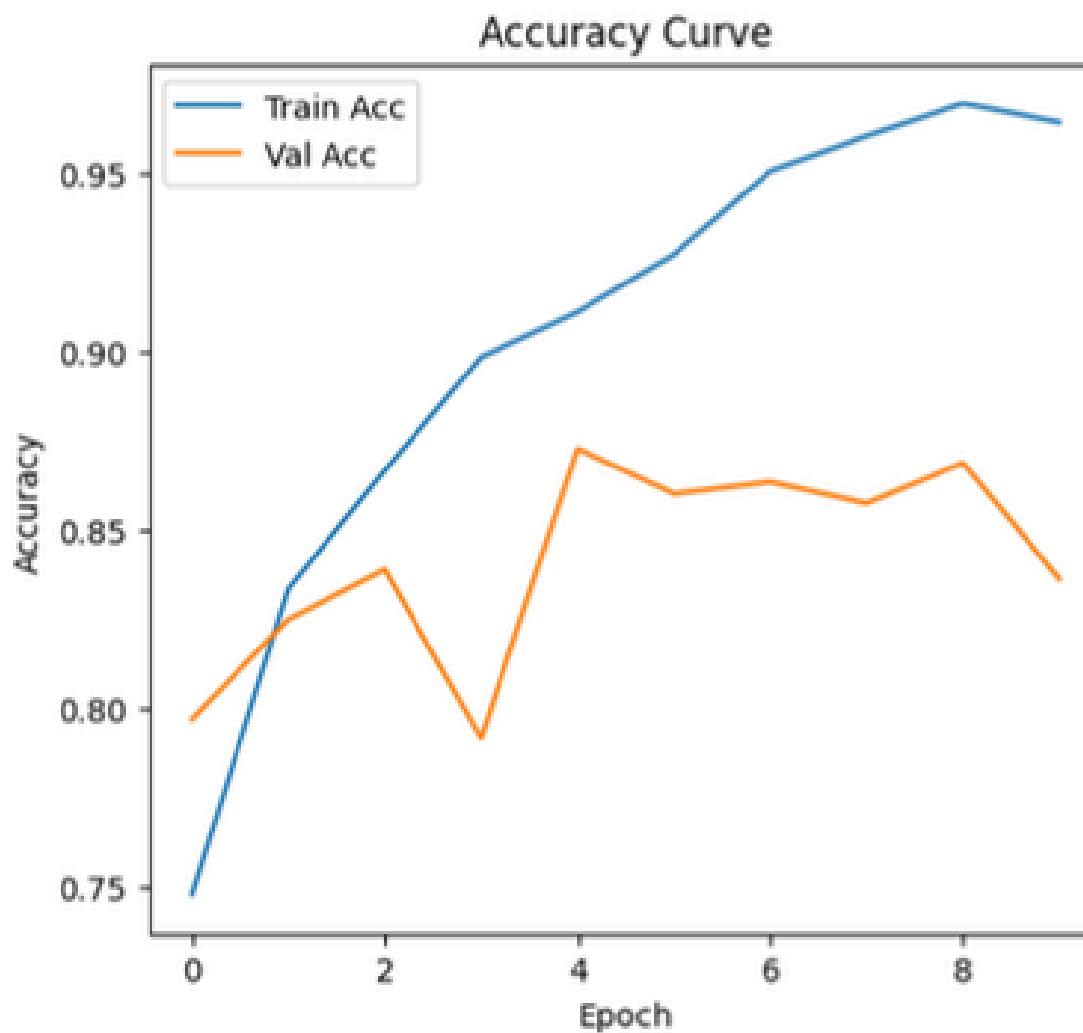
Classification Report:

	precision	recall	f1-score	support
akiec	0.51	0.47	0.49	49
bcc	0.63	0.58	0.61	77
bkl	0.54	0.49	0.51	165
df	0.00	0.00	0.00	17
mel	0.56	0.43	0.48	167
nv	0.86	0.94	0.90	1006
vasc	0.89	0.73	0.80	22
accuracy			0.79	1503
macro avg	0.57	0.52	0.54	1503
weighted avg	0.76	0.79	0.77	1503



# ResNet50 - Model kết hợp Ảnh và Metadata

Train Loss: 0.1008 Train Acc: 0.9645 Val Loss: 0.6581 Val Acc: 0.8362



Classification Report:

	precision	recall	f1-score	support
akiec	0.35	0.84	0.50	49
bcc	0.67	0.86	0.75	77
bkl	0.84	0.52	0.64	165
df	0.82	0.53	0.64	17
mel	0.76	0.45	0.56	167
nv	0.92	0.97	0.94	1006
vasc	0.93	0.59	0.72	22
accuracy			0.84	1503
macro avg	0.76	0.68	0.68	1503
weighted avg	0.86	0.84	0.84	1503

# ResNet50 - Nhận xét biểu đồ

---

- **Chỉ ảnh:** Accuracy train và val tăng ổn định, chênh lệch nhỏ (~0.02), loss giảm đều → mô hình học tốt, ít overfitting nhưng độ chính xác tối đa (~0.78) chưa cao.
- **Ảnh + metadata:** Train acc tăng nhanh tới ~0.97, nhưng val acc dao động quanh ~0.85, chênh lệch lớn. Val loss giảm nhẹ rồi tăng cuối quá trình → overfitting rõ rệt. Metadata giúp mô hình học nhanh hơn nhưng giảm khả năng tổng quát hóa.

# ResNet50 - Nhận xét bảng báo cáo phân loại

---

- **Chỉ ảnh:** Accuracy ~0.79, macro F1 thấp (0.54) → mô hình dự đoán tốt cho lớp lớn (nv) nhưng yếu với lớp ít mẫu (df, akiec). Một số lớp có recall rất thấp (df = 0).
- **Ảnh + metadata:** Accuracy tăng lên 0.84, macro F1 ~0.68 → cải thiện rõ rệt ở các lớp hiếm (akiec recall 0.84, df recall 0.53). Tuy nhiên, một số lớp (mel, vasc) vẫn bị giảm recall so với precision, cho thấy mất cân bằng nhạy-đặc hiệu giữa các lớp.



# ResNet50 - Kết luận

---

Vậy với model ResNet50, chúng ta nên dùng **cả ảnh và metadata** vì:

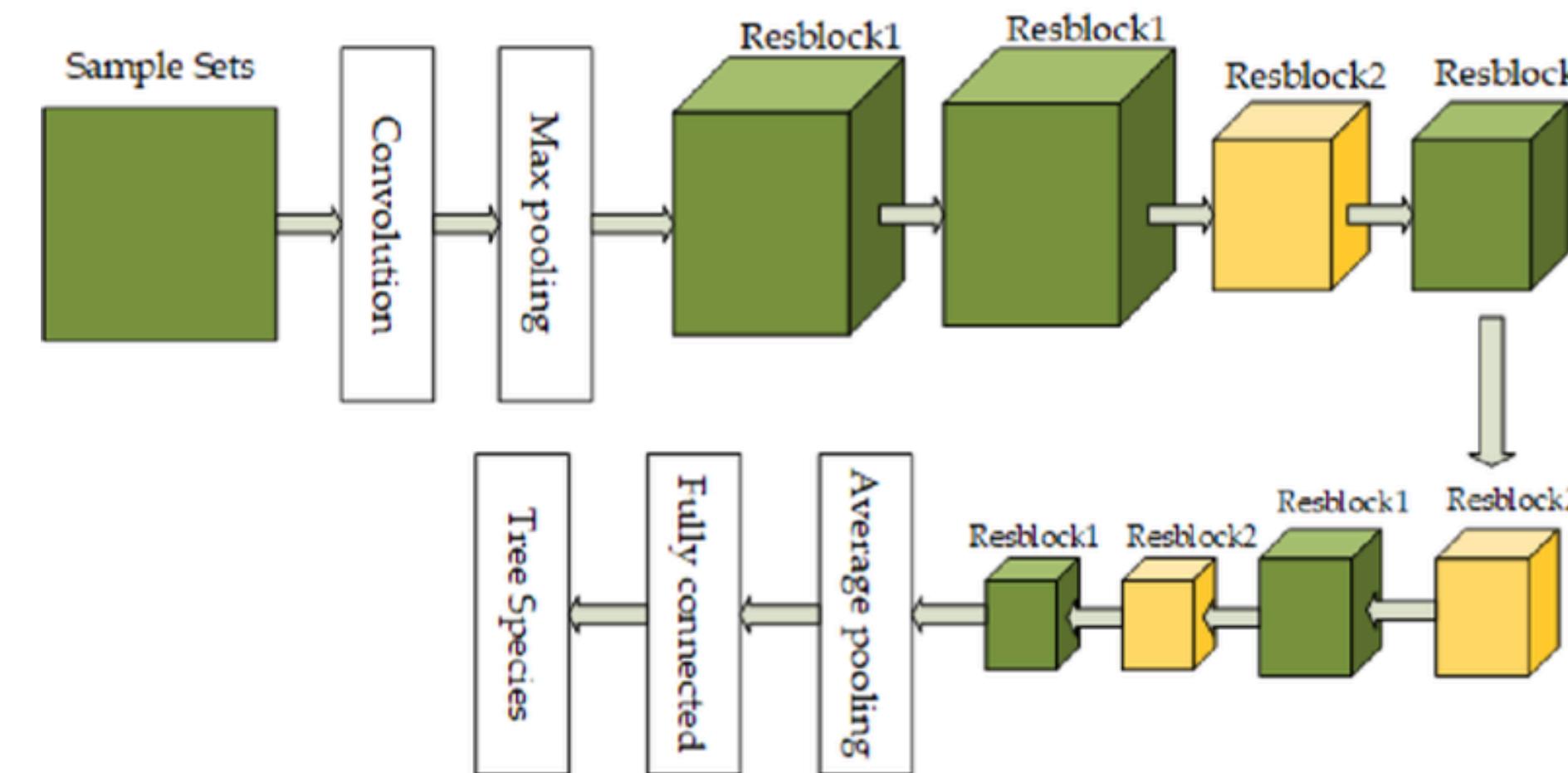
- Ảnh chứa đặc trưng trực quan, giúp mô hình học được hình dạng, màu sắc, hoa văn của tổn thương da.
- Metadata cung cấp thông tin bổ sung như tuổi, giới tính, vị trí trên cơ thể... → hỗ trợ phân loại chính xác hơn, đặc biệt khi nhiều bệnh nhìn rất giống nhau.
- Kết hợp cả hai giúp mô hình có nguồn thông tin đa chiều, tăng khả năng dự đoán so với chỉ dùng ảnh.



# ResNet18

# ResNet18

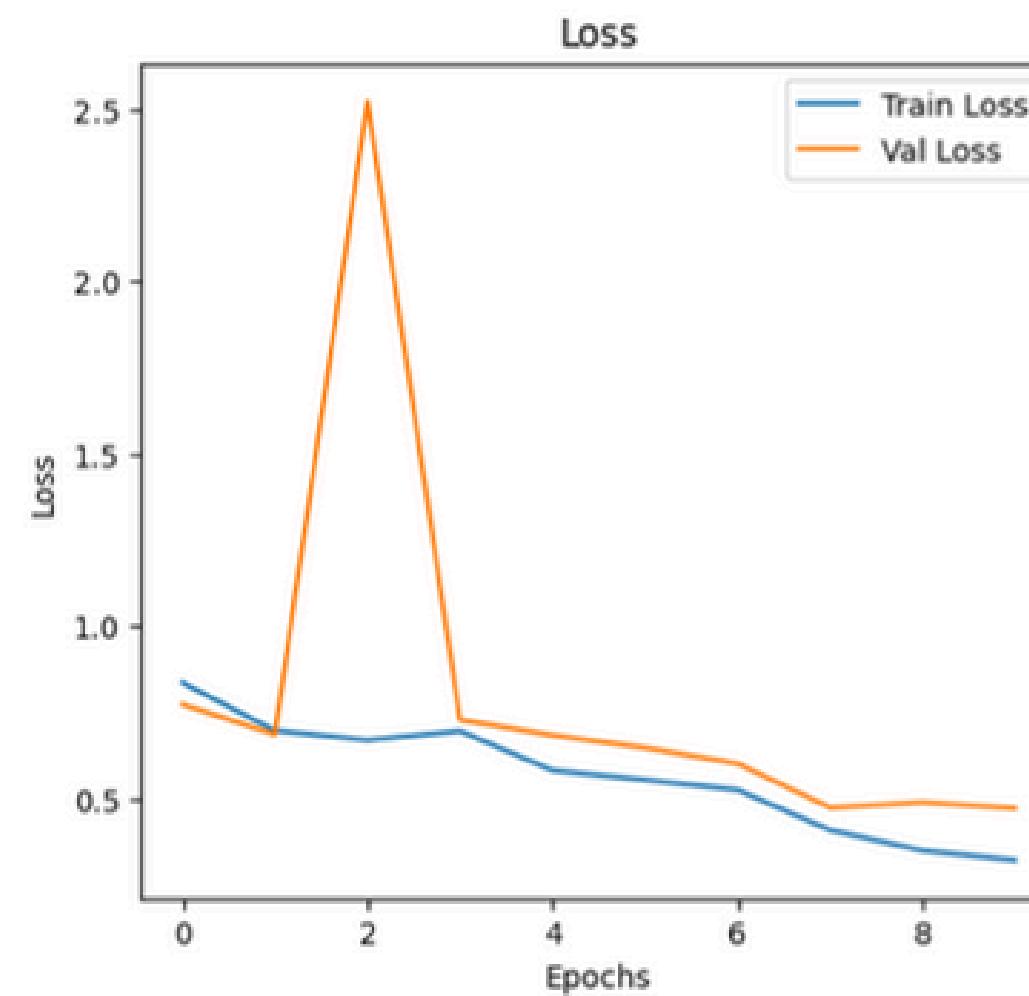
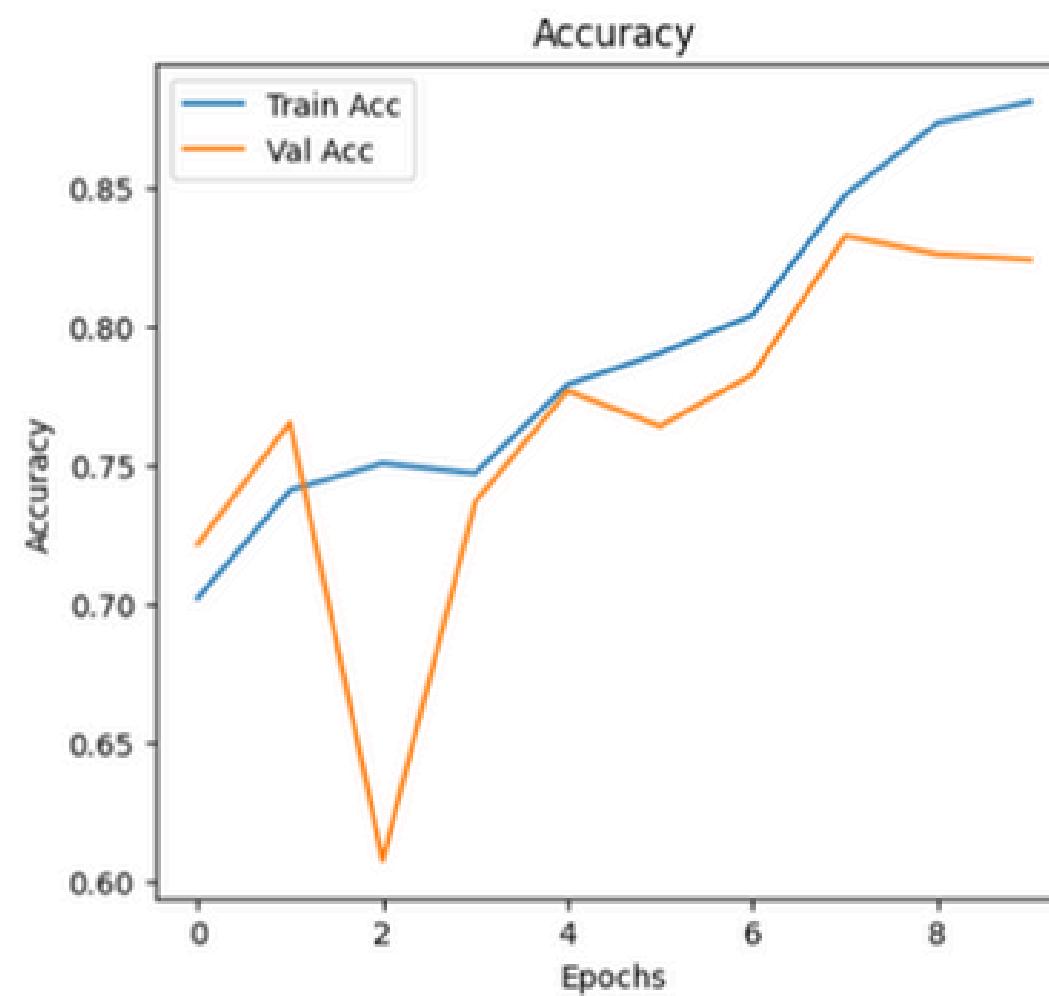
ResNet18 nhỏ gọn hơn ResNet50 (11.7M tham số so với ~25M), huấn luyện nhanh hơn và ít tốn tài nguyên, phù hợp khi dữ liệu hạn chế hoặc cần suy luận thời gian thực. Tuy nhiên, độ chính xác thường thấp hơn ResNet50 và các kiến trúc hiện đại như EfficientNet khi làm việc với tập dữ liệu phức tạp.





# ResNet18 - Model Ánh

train Loss: 0.3207 Acc: 0.8813  
val Loss: 0.4720 Acc: 0.8242



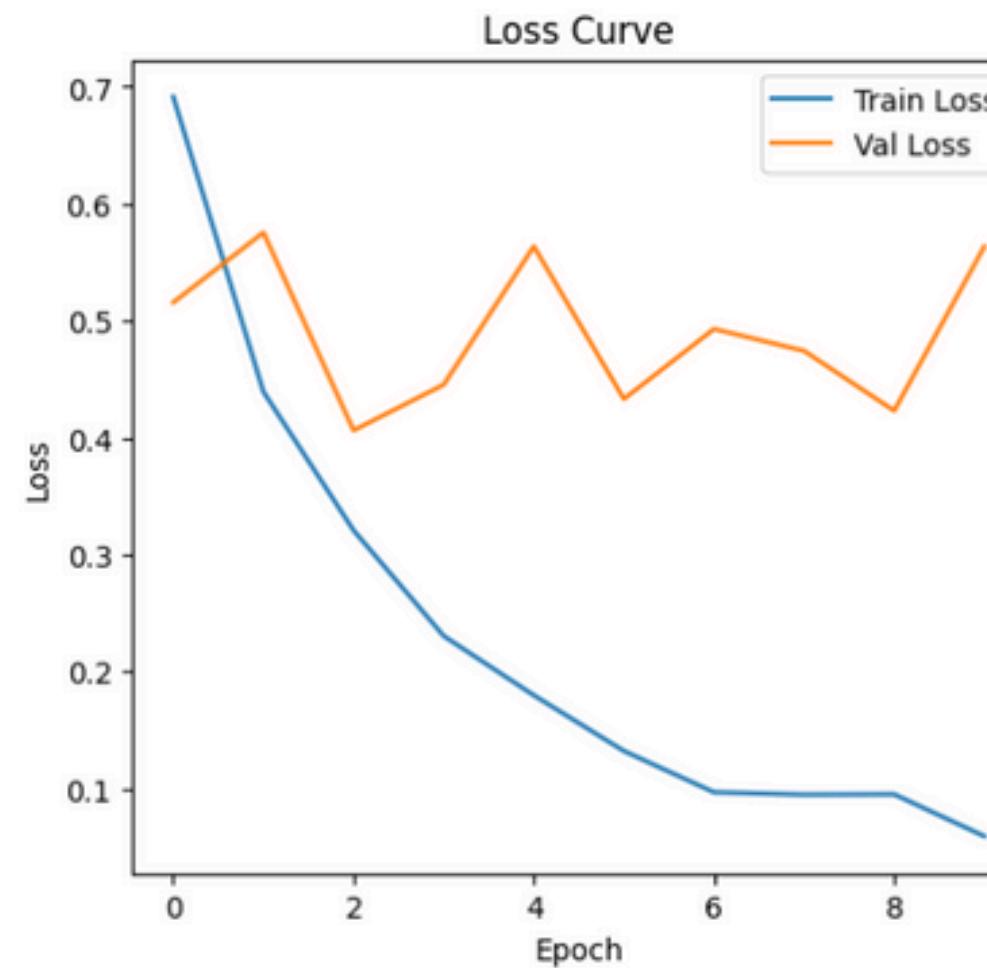
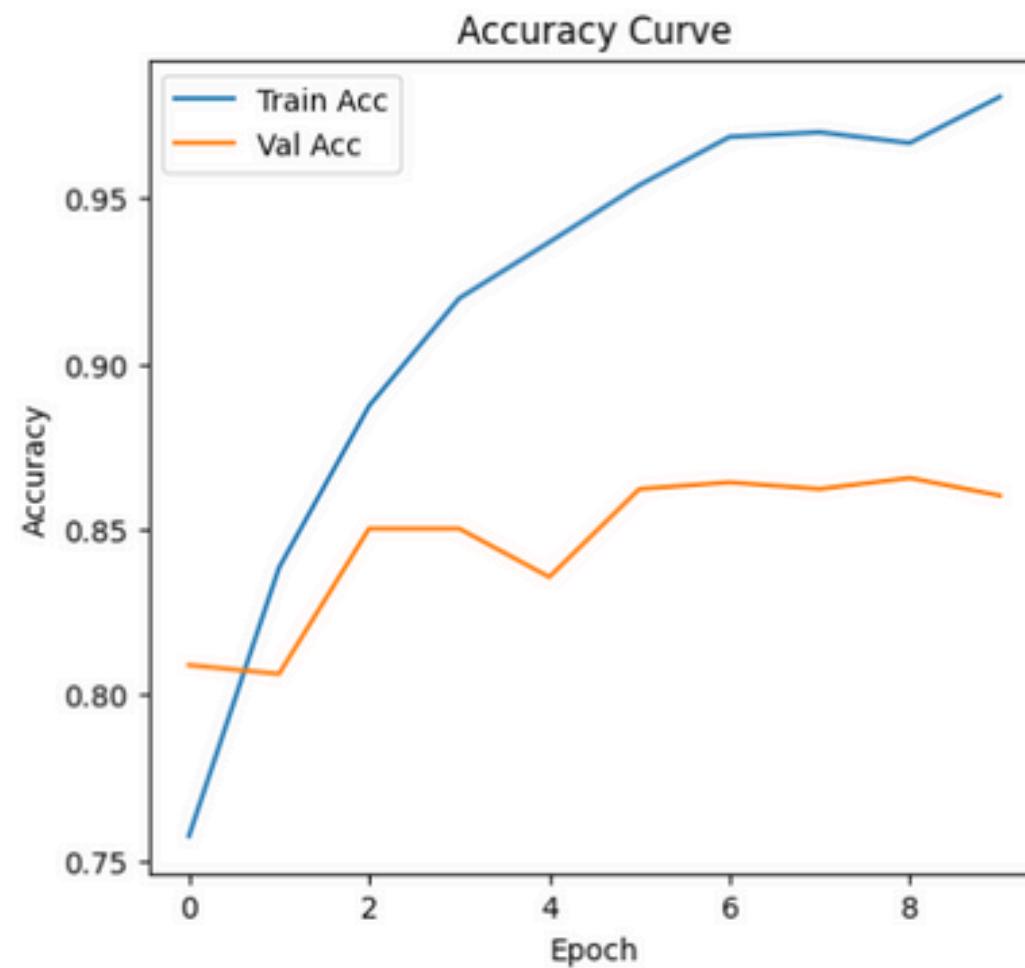
Classification Report:

	precision	recall	f1-score	support
akiec	0.56	0.61	0.58	49
bcc	0.65	0.86	0.74	77
bkl	0.67	0.64	0.65	165
df	0.86	0.35	0.50	17
mel	0.65	0.47	0.55	167
nv	0.90	0.93	0.92	1006
vasc	1.00	0.73	0.84	22
accuracy			0.83	1503
macro avg	0.75	0.66	0.68	1503
weighted avg	0.82	0.83	0.82	1503



# ResNet18 - Model kết hợp Ảnh và Metadata

Train Loss: 0.0591 Train Acc: 0.9807 Val Loss: 0.5634 Val Acc: 0.8602



Classification Report:

	precision	recall	f1-score	support
akiec	0.63	0.49	0.55	49
bcc	0.78	0.77	0.77	77
bkl	0.68	0.75	0.71	165
df	1.00	0.47	0.64	17
mel	0.75	0.43	0.54	167
nv	0.88	0.96	0.92	1006
vasc	0.93	0.64	0.76	22
accuracy			0.84	1503
macro avg	0.81	0.64	0.70	1503
weighted avg	0.84	0.84	0.83	1503

# ResNet18 - Nhận xét biểu đồ

---

- **Chỉ ảnh:** Accuracy tăng đều, đạt ~0.88 train và ~0.82 val; loss giảm ổn định. Tuy nhiên, val loss ban đầu có đỉnh bất thường (epoch 2) → mô hình có thể gặp khó khăn tạm thời trong giai đoạn đầu học.
- **Ảnh + metadata:** Accuracy train tăng nhanh, gần tiệm cận 0.98; val accuracy ổn định quanh 0.86. Train loss giảm mạnh, val loss dao động nhưng vẫn giữ mức thấp hơn so với chỉ ảnh.

# ResNet18 - Nhận xét bảng báo cáo phân loại

---

- **Chỉ ảnh:** macro avg F1 = 0.68, weighted avg F1 = 0.82. Một số lớp hiếm (df) recall thấp.
- **Ảnh + metadata:** macro avg F1 = 0.70, weighted avg F1 = 0.83. Một số lớp khó (bkl, akiec) cải thiện rõ về precision hoặc recall.

# ResNet18 - Kết luận

---

- **Kết hợp ảnh + metadata** giúp ResNet18 đạt Val Acc cao hơn, macro avg F1 tốt hơn và cải thiện độ cân bằng giữa các lớp, đặc biệt ở các lớp ít mẫu.
- Tuy nhiên, mô hình cũng có dấu hiệu overfitting nhẹ do train acc rất cao so với val acc.

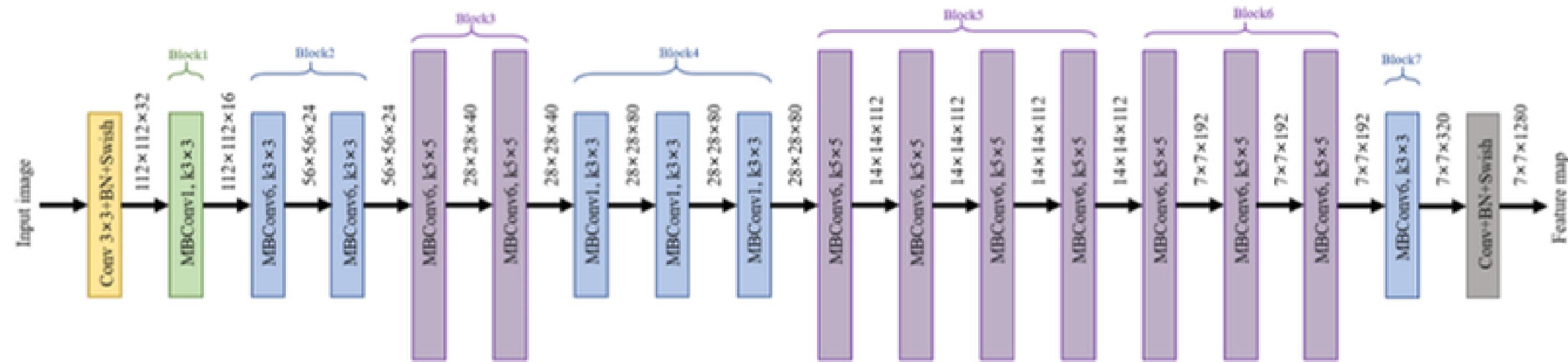


# EfficientB0



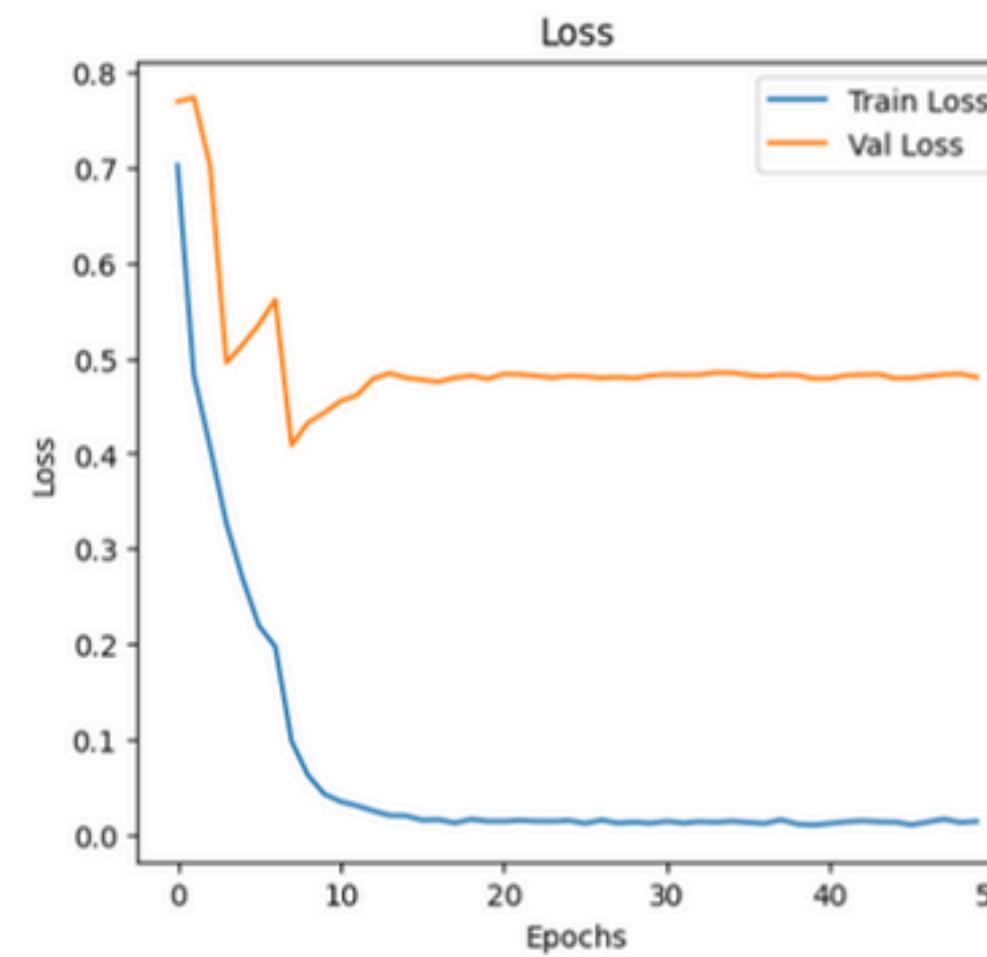
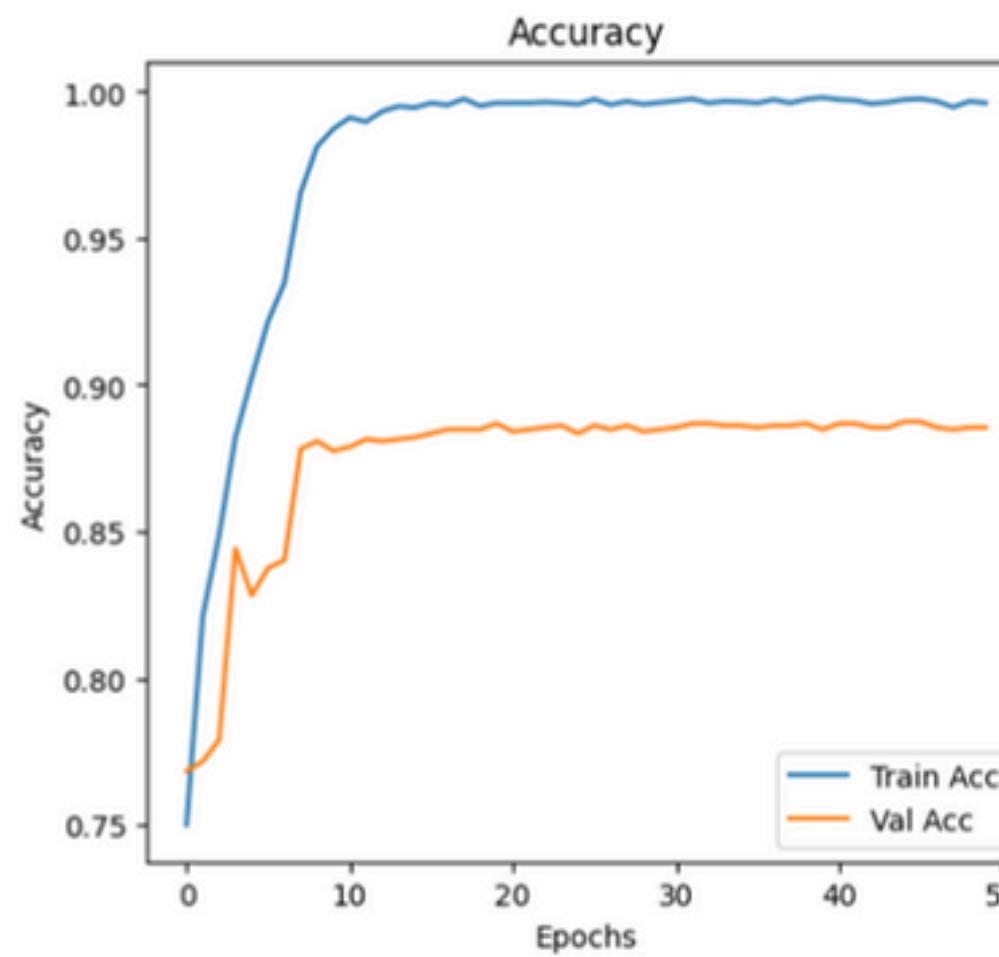
# EfficientB0

EfficientNet-B0 tối ưu hóa giữa độ chính xác và số tham số (~5.3M), nhẹ hơn ResNet50 nhưng vẫn đạt hiệu suất cao nhờ kiến trúc compound scaling.



# EfficientB0 - Model Ánh

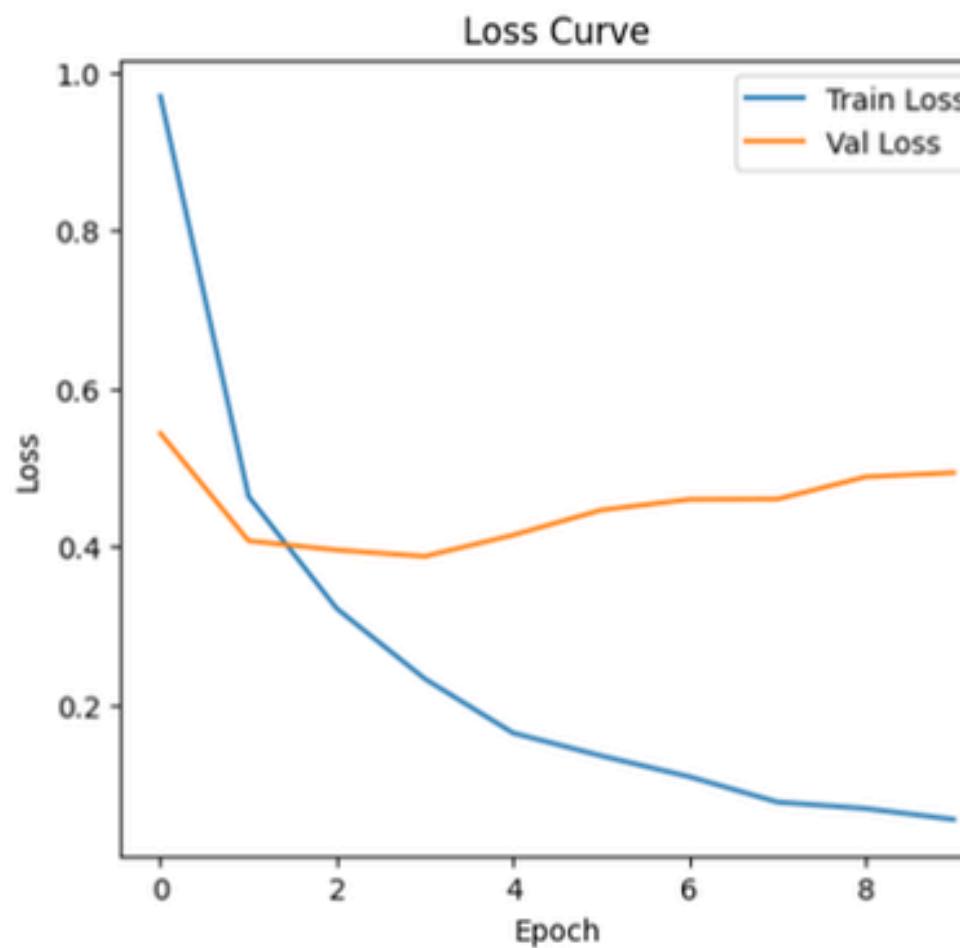
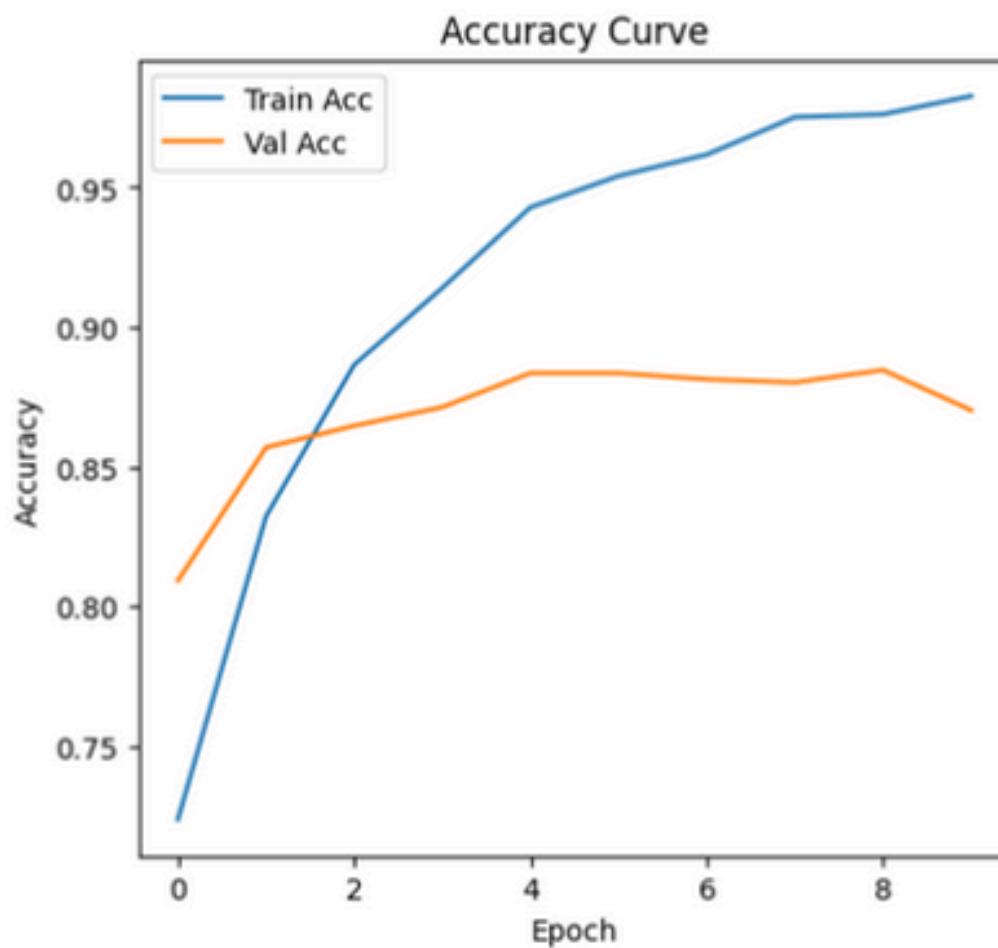
train Loss: 0.0141 Acc: 0.9961  
val Loss: 0.4799 Acc: 0.8855



	Classification Report:			
	precision	recall	f1-score	support
akiec	0.74	0.71	0.73	49
bcc	0.75	0.82	0.78	77
bkl	0.82	0.78	0.80	165
df	0.91	0.59	0.71	17
mel	0.80	0.63	0.71	167
nv	0.92	0.97	0.95	1006
vasc	1.00	0.86	0.93	22
accuracy			0.89	1503
macro avg	0.85	0.77	0.80	1503
weighted avg	0.89	0.89	0.88	1503

# EfficientB0 - Model kết hợp Ảnh và Metadata

Train Loss: 0.0550 Train Acc: 0.9827 Val Loss: 0.4936 Val Acc: 0.8703



Classification Report:

	precision	recall	f1-score	support
akiec	0.69	0.76	0.72	33
bcc	0.93	0.75	0.83	51
bkl	0.83	0.78	0.80	110
df	0.83	0.83	0.83	12
mel	0.69	0.73	0.71	111
nv	0.94	0.95	0.95	671
vasc	0.86	0.86	0.86	14
accuracy			0.89	1002
macro avg	0.82	0.81	0.81	1002
weighted avg	0.89	0.89	0.89	1002

# EfficientB0 - Nhận xét biểu đồ

---

## Chỉ ảnh:

- Accuracy tăng nhanh, train gần đạt 1.0, val ổn định ~0.886.
- Loss train giảm rất mạnh, val loss ổn định ~0.48 → không overfit nặng.

## Ảnh + metadata:

- Train accuracy ~0.983, val accuracy ~0.87 → **hơi thấp hơn.**
- Val loss ~0.49, gần bằng mô hình chỉ ảnh.

# EfficientB0 - Nhận xét bảng báo cáo phân loại

---

- **Chỉ ảnh:** Macro F1 = 0.80 (cao hơn), đặc biệt tốt ở các lớp lớn như nv, bkl.
- **Ảnh + metadata:** Macro F1 = 0.81 (gần tương đương), nhưng kết quả không vượt trội, một số lớp nhỏ (bcc, df) được cải thiện.

# EfficientB0 - Kết luận

---

- Với EfficientNetB0, **chỉ ảnh** cho Val Acc **cao hơn** (0.8855 vs 0.8703) và macro F1 tương đương.
- **Ảnh + metadata** không cải thiện rõ rệt, nhưng có thể giúp một số lớp ít mẫu dự đoán tốt hơn.
- Nếu ưu tiên độ chính xác tổng thể, dùng chỉ ảnh; nếu cần cân bằng hơn giữa các lớp, có thể cân nhắc thêm metadata.

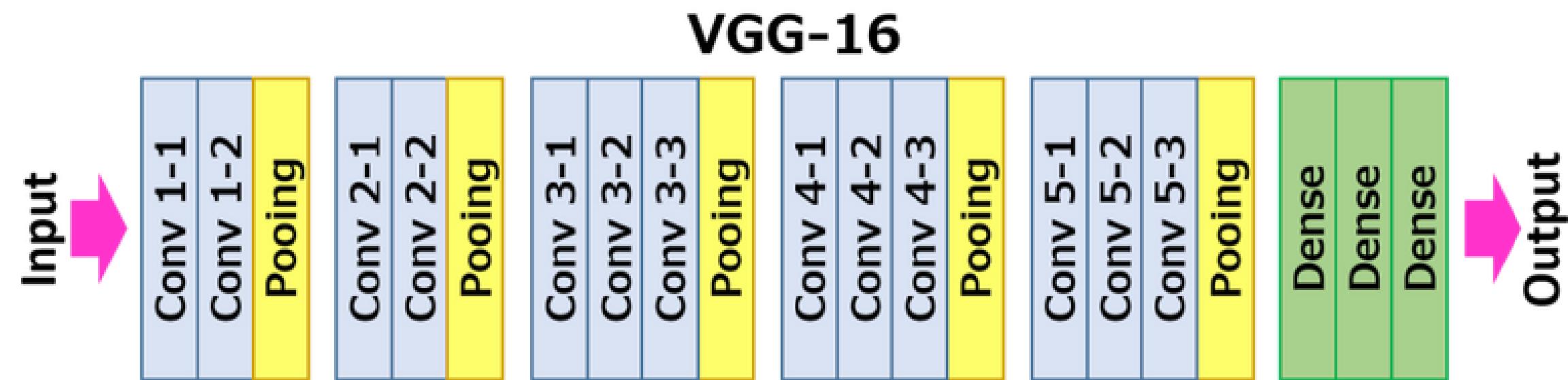


# VGG16



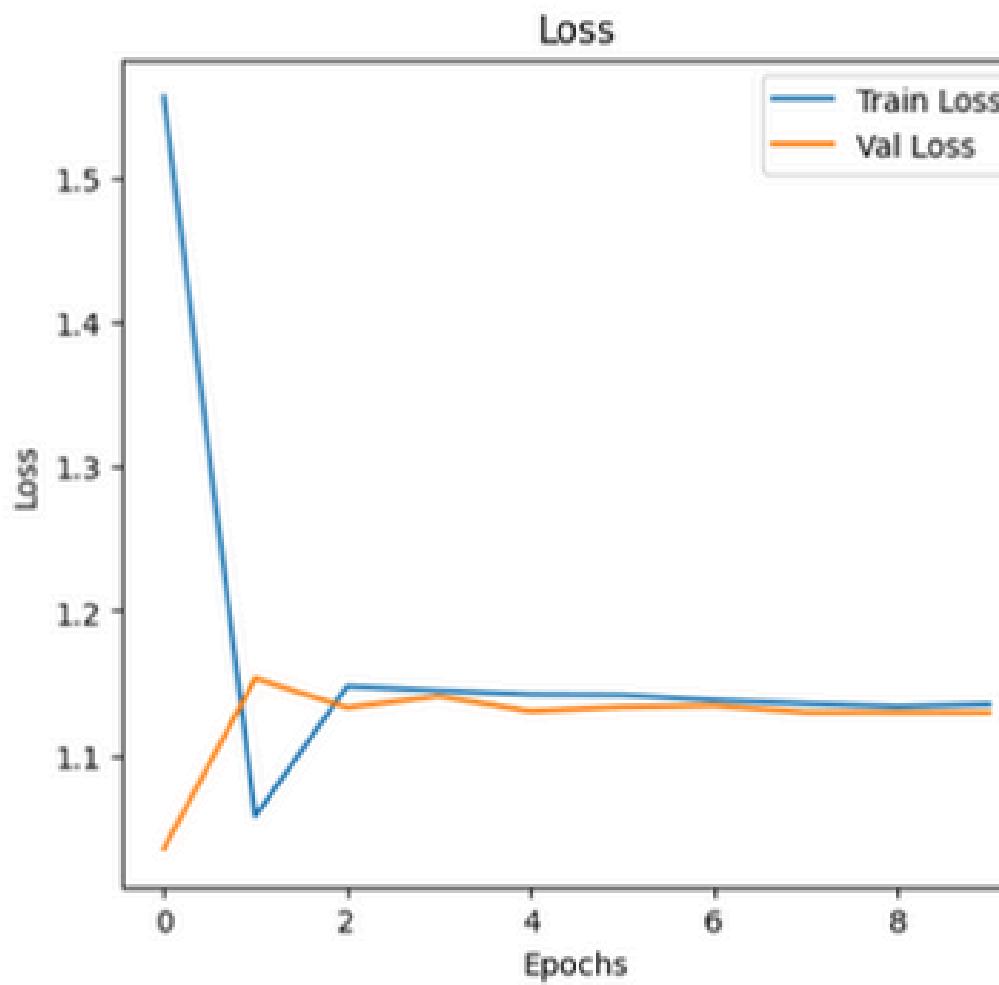
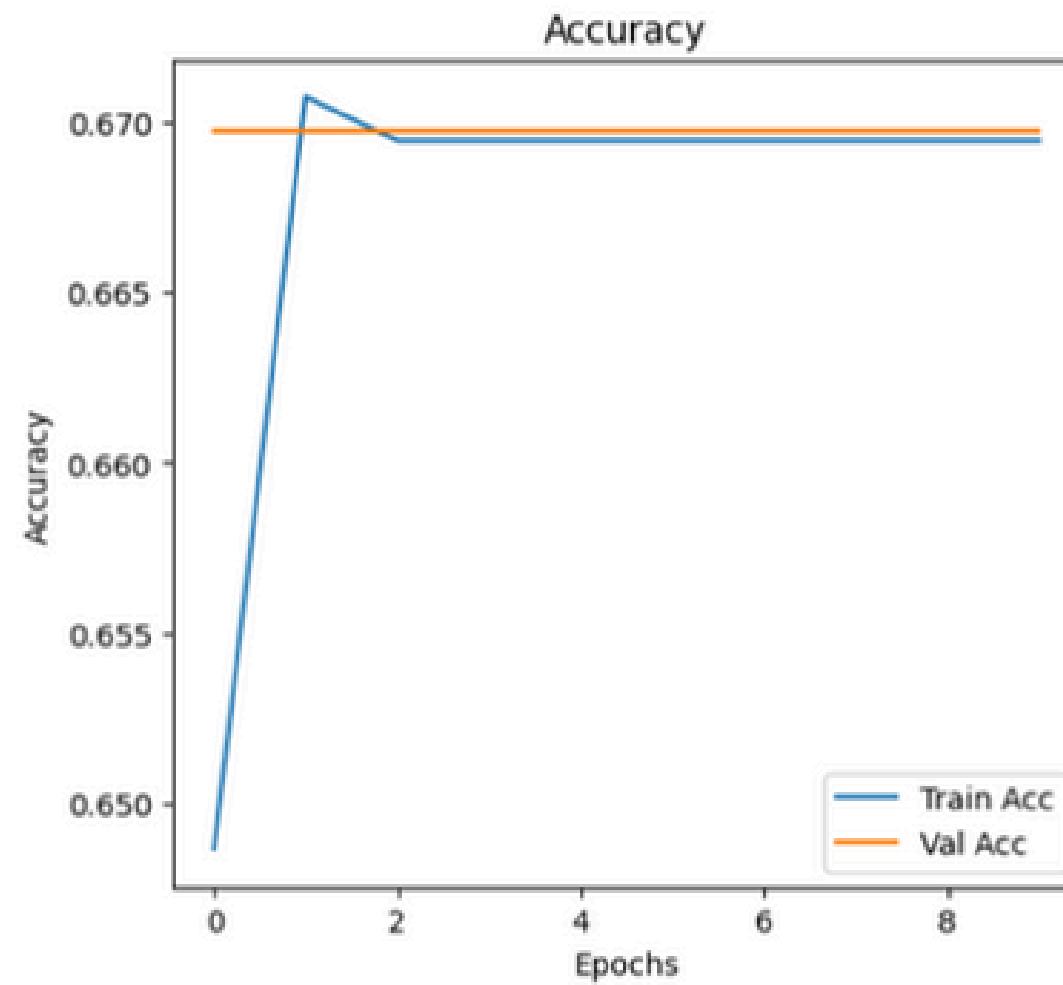
# VGG16

VGG16 có kiến trúc đơn giản, dễ hiểu, nhưng rất nặng (~138M tham số) và tốn tài nguyên tính toán. Dù từng đạt hiệu quả cao, hiện nay nó thường cho độ chính xác thấp hơn các kiến trúc mới như ResNet hay EfficientNet, đặc biệt khi huấn luyện trên dữ liệu vừa và nhỏ.



# VGG16 - Model Ánh

train Loss: 1.1353 Acc: 0.6695  
val Loss: 1.1297 Acc: 0.6698

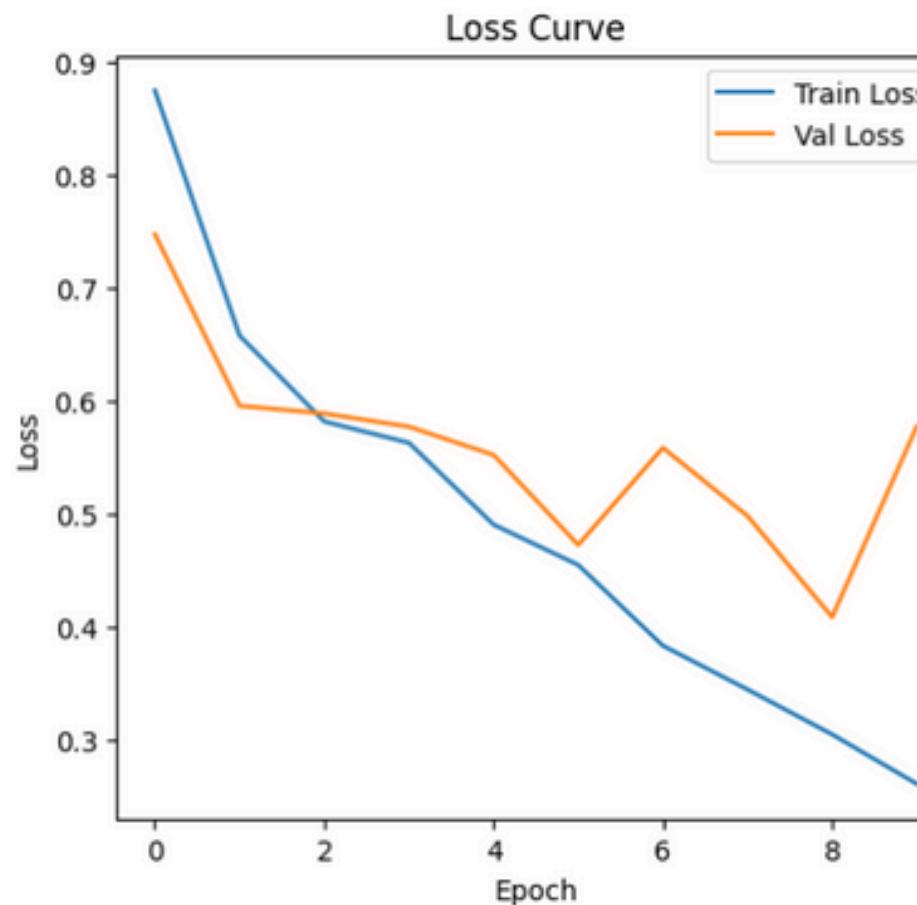
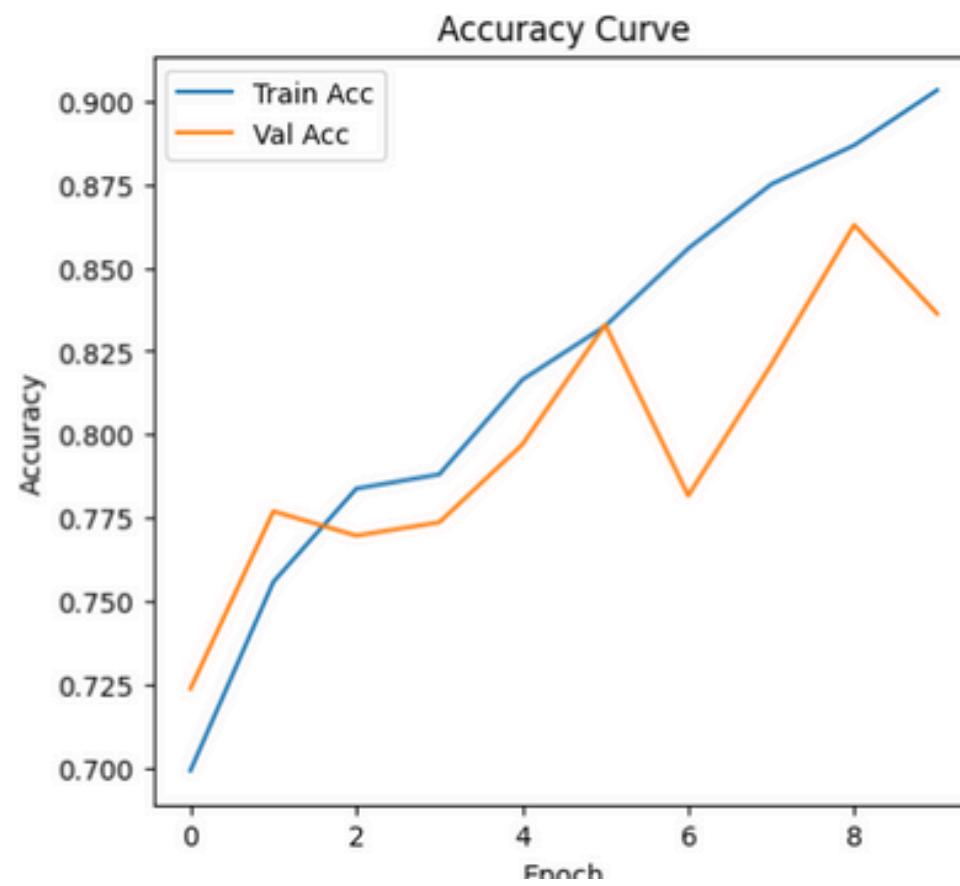


Classification Report:

	precision	recall	f1-score	support
akiec	0.00	0.00	0.00	49
bcc	0.00	0.00	0.00	77
bkl	0.00	0.00	0.00	165
df	0.00	0.00	0.00	17
mel	0.00	0.00	0.00	167
nv	0.67	1.00	0.80	1006
vasc	0.00	0.00	0.00	22
accuracy			0.67	1503
macro avg	0.10	0.14	0.11	1503
weighted avg	0.45	0.67	0.54	1503

# VGG16 - Model kết hợp Ảnh và Metadata

Train Loss: 0.2611 Train Acc: 0.9033 Val Loss: 0.5772 Val Acc: 0.8362



Classification Report:

	precision	recall	f1-score	support
akiec	0.44	0.57	0.50	49
bcc	0.83	0.56	0.67	77
bkl	0.71	0.73	0.72	165
df	0.64	0.41	0.50	17
mel	0.74	0.49	0.58	167
nv	0.90	0.97	0.93	1006
vasc	1.00	0.32	0.48	22
accuracy			0.84	1503
macro avg	0.75	0.58	0.63	1503
weighted avg	0.84	0.84	0.83	1503

# VGG16 - Nhận xét biểu đồ

---

- **Chỉ ảnh:** Train loss tiếp tục giảm, nhưng validation loss lại tăng và dao động, cho thấy mô hình không tổng quát hóa tốt trên dữ liệu mới.
- **Ảnh+metadata:** Mô hình thiếu khớp (underfitting) hoặc dừng học quá sớm. Cả măt măt và độ chính xác của tập huấn luyện và kiểm định đều ổn định sau vài epoch đầu, cho thấy mô hình chưa học đủ từ dữ liệu.



# VGG16 - Nhận xét bảng báo cáo phân loại

---

## Chỉ ảnh:

- Accuracy huấn luyện: ~66.95%
- Accuracy validation: ~66.98%
- Classification report cho thấy hầu hết các lớp (trừ “nv”) gần như không được mô hình dự đoán đúng → mô hình underfitting và phụ thuộc vào lớp chiếm đa số.

## Ảnh + metadata:

- Accuracy huấn luyện: ~90.33%
- Accuracy validation: ~83.62%
- Precision, recall, và F1-score cải thiện rõ rệt ở nhiều lớp, phân bổ dự đoán cân bằng hơn.

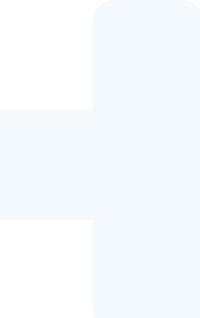




# VGG16 - Kết luận

---

Với VGG16, việc thêm metadata giúp mô hình học được nhiều thông tin hơn, tăng độ chính xác validation từ 66.98% lên 83.62%, đồng thời cải thiện khả năng phân biệt giữa các lớp hiếm.

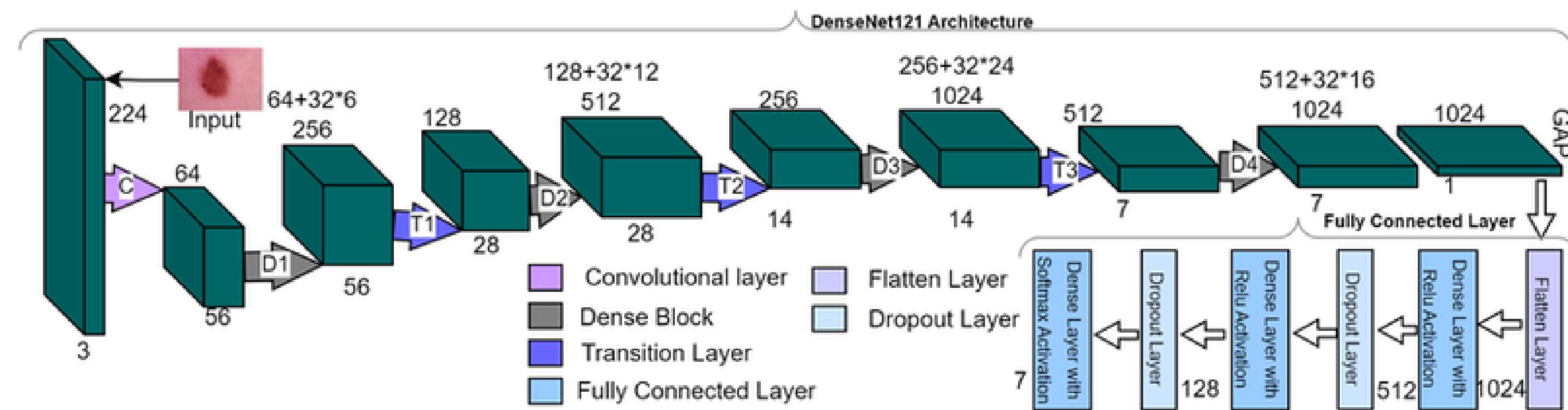


# DenseNet121



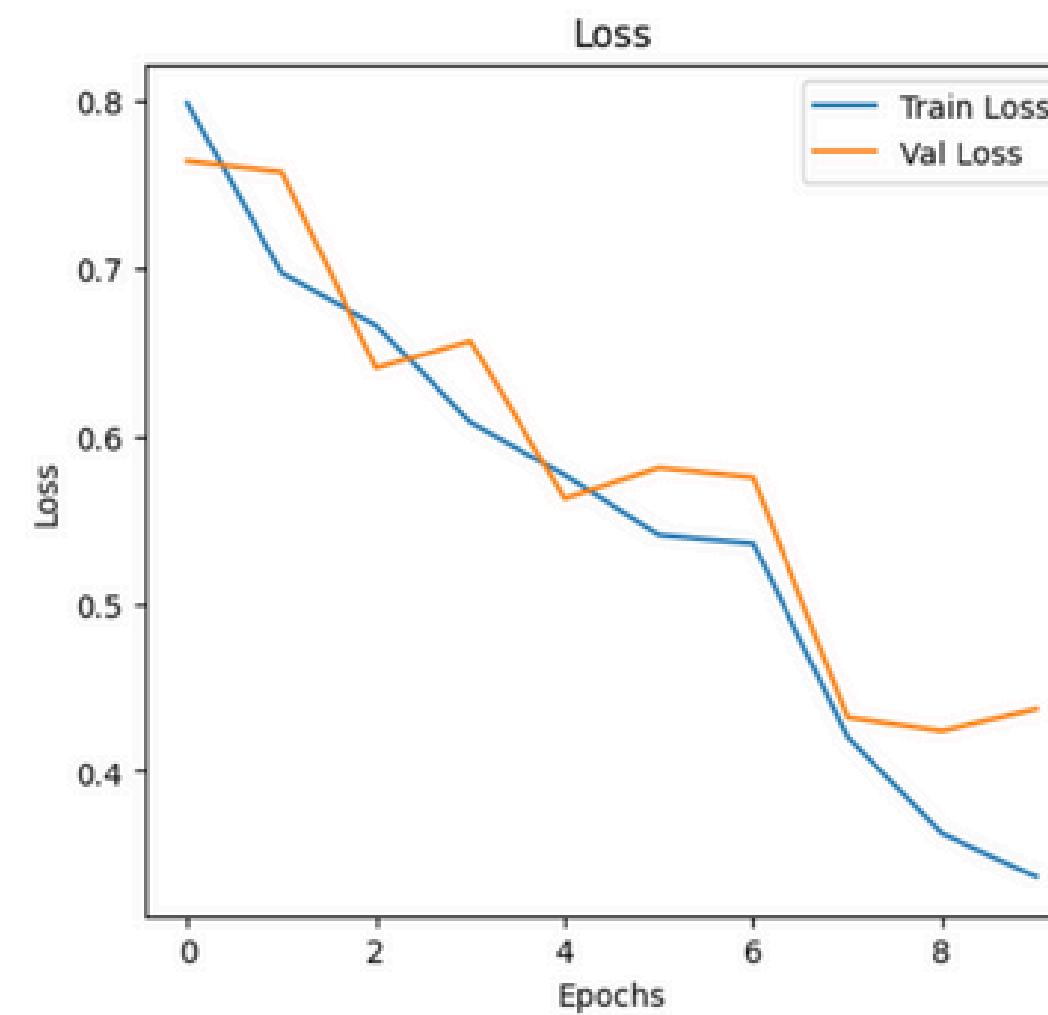
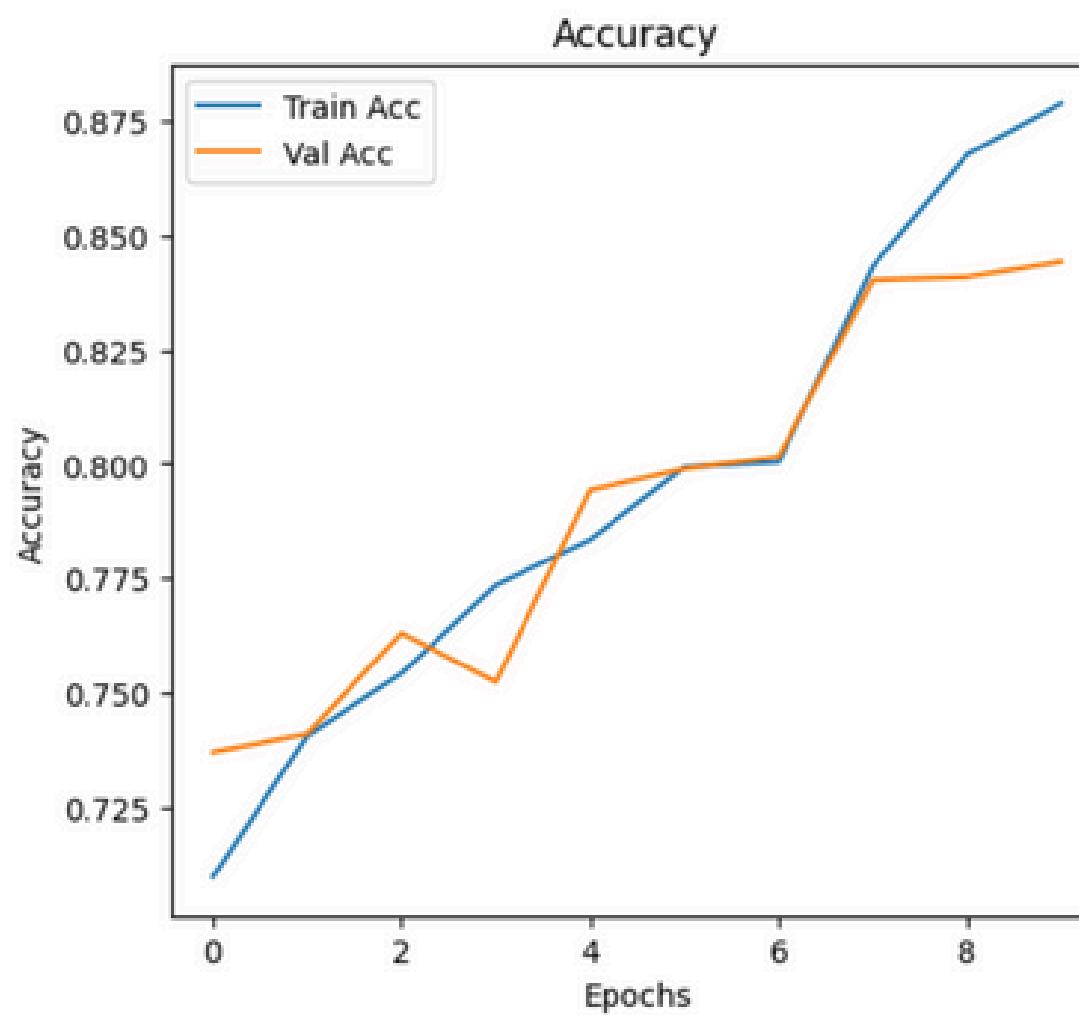
# DenseNet121

DenseNet121 có số tham số vừa phải (~8M) nhưng tận dụng kết nối dày đặc giữa các lớp, giúp tái sử dụng đặc trưng hiệu quả và thường đạt độ chính xác cao hơn VGG16 trên dữ liệu vừa và nhỏ. Tuy nhiên, tốc độ suy luận chậm hơn so với mô hình nhẹ.



# DenseNet121 - Model Ánh

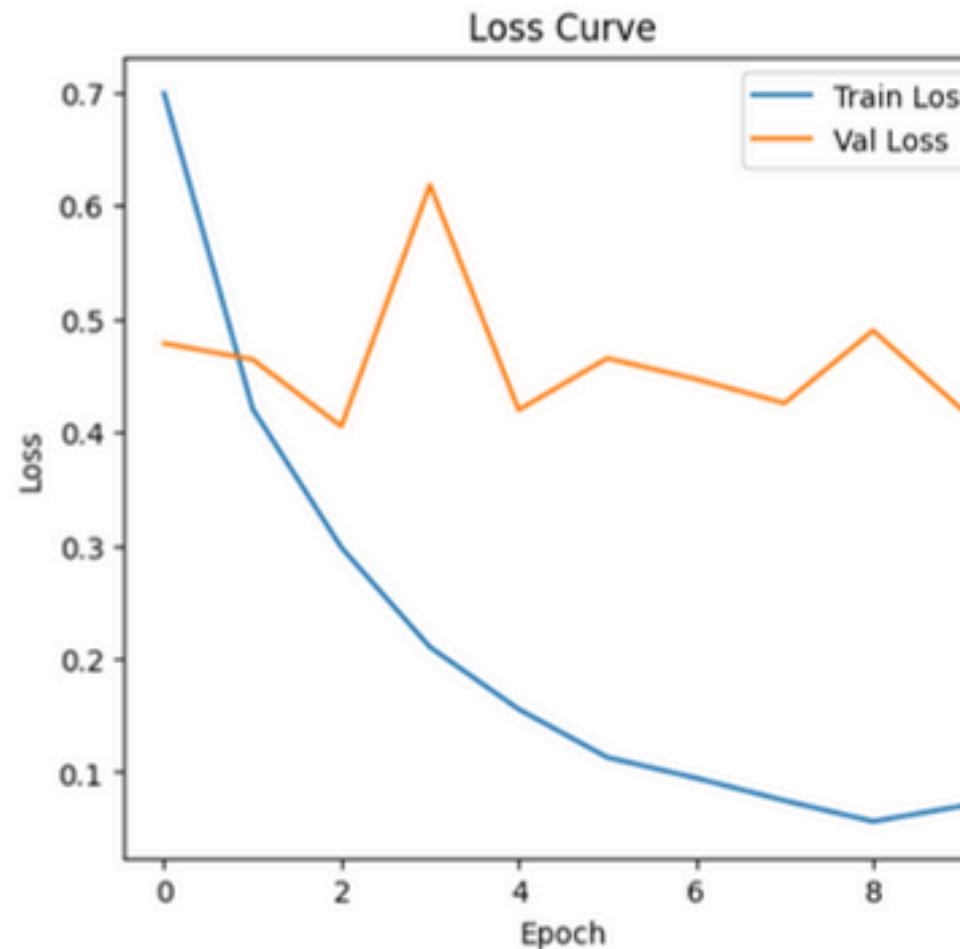
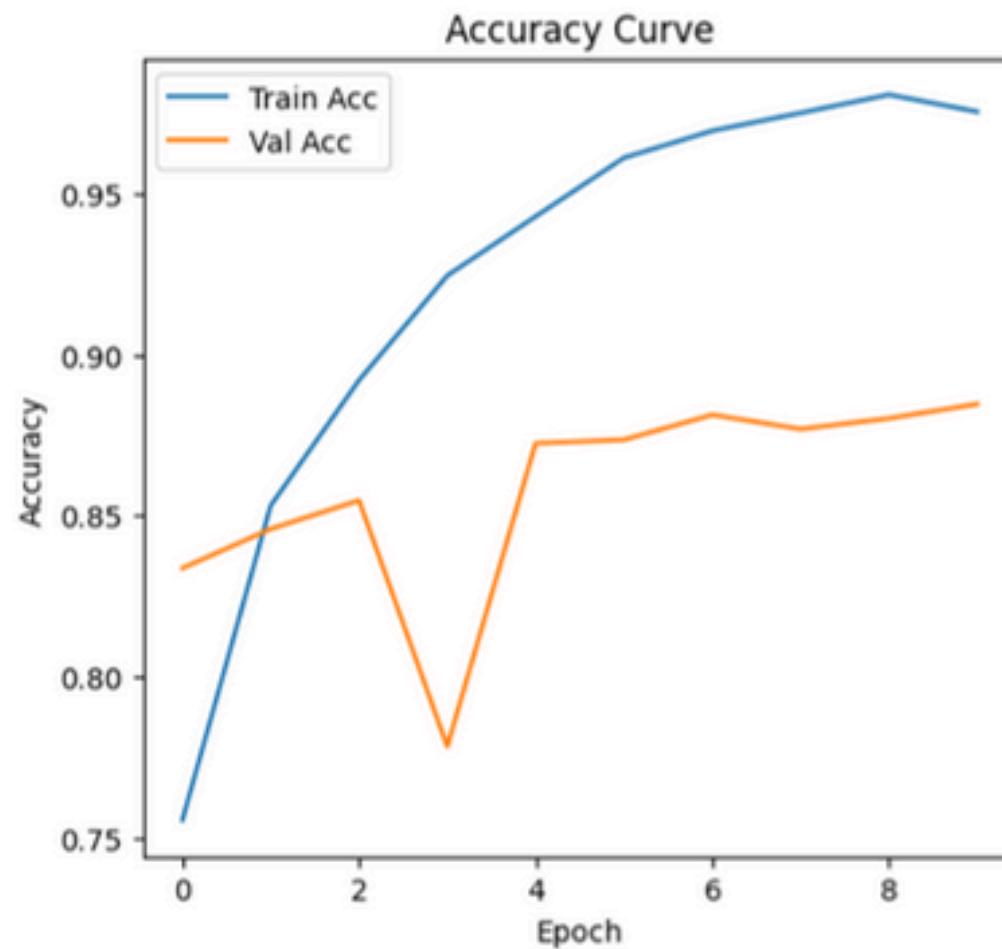
```
train Loss: 0.3369 Acc: 0.8789  
val Loss: 0.4371 Acc: 0.8442
```



Classification Report:				
	precision	recall	f1-score	support
akiec	0.65	0.61	0.63	49
bcc	0.68	0.83	0.75	77
bkl	0.69	0.67	0.68	165
df	0.64	0.41	0.50	17
mel	0.69	0.51	0.58	167
nv	0.91	0.95	0.93	1006
vasc	0.89	0.77	0.83	22
accuracy				1503
macro avg	0.74	0.68	0.70	1503
weighted avg	0.84	0.84	0.84	1503

# DenseNet121 - Model kết hợp Ảnh và Metadata

Train Loss: 0.0699 Train Acc: 0.9756 Val Loss: 0.4198 Val Acc: 0.8847



Classification Report:

	precision	recall	f1-score	support
akiec	0.68	0.79	0.73	33
bcc	0.91	0.76	0.83	51
bkl	0.68	0.85	0.75	110
df	1.00	0.83	0.91	12
mel	0.72	0.67	0.69	111
nv	0.96	0.93	0.94	671
vasc	0.87	0.93	0.90	14
accuracy			0.88	1002
macro avg	0.83	0.82	0.82	1002
weighted avg	0.89	0.88	0.88	1002

# DenseNet121 - Nhận xét biểu đồ

---

## Chỉ ảnh:

- Mô hình học tốt, cả accuracy và loss đều cải thiện qua các epoch.
- Có thể đã bắt đầu overfit nhẹ sau epoch 7–8 (Train Acc > Val Acc rõ rệt, Train Loss < Val Loss rõ rệt).

## Ảnh và metadata:

- Mô hình đã overfit nặng sau khoảng epoch 2–3.
- Nếu mục tiêu là tổng quát hóa tốt hơn, nên dùng early stopping tại epoch 2 hoặc 3.

# DenseNet121 - Nhận xét bảng báo cáo phân loại

---

- **Ảnh + metadata** cho accuracy cao hơn (0.88 vs 0.84), macro avg f1 cũng cao hơn đáng kể (0.82 vs 0.70) → cải thiện rõ rệt về độ cân bằng giữa các lớp.
- Các lớp khó như df và akiec tăng mạnh recall (df: 0.41 → 0.83, akiec: 0.61 → 0.79).
- Một số lớp phổ biến như nv giữ f1 rất cao nhưng thay đổi không nhiều.
- Tuy nhiên, việc cải thiện này có thể đi kèm rủi ro overfitting như đã thấy ở biểu đồ trước, do metadata cung cấp thông tin phân biệt dễ học nhưng có thể kém tổng quát hóa.

# DenseNet121 - Kết luận

---

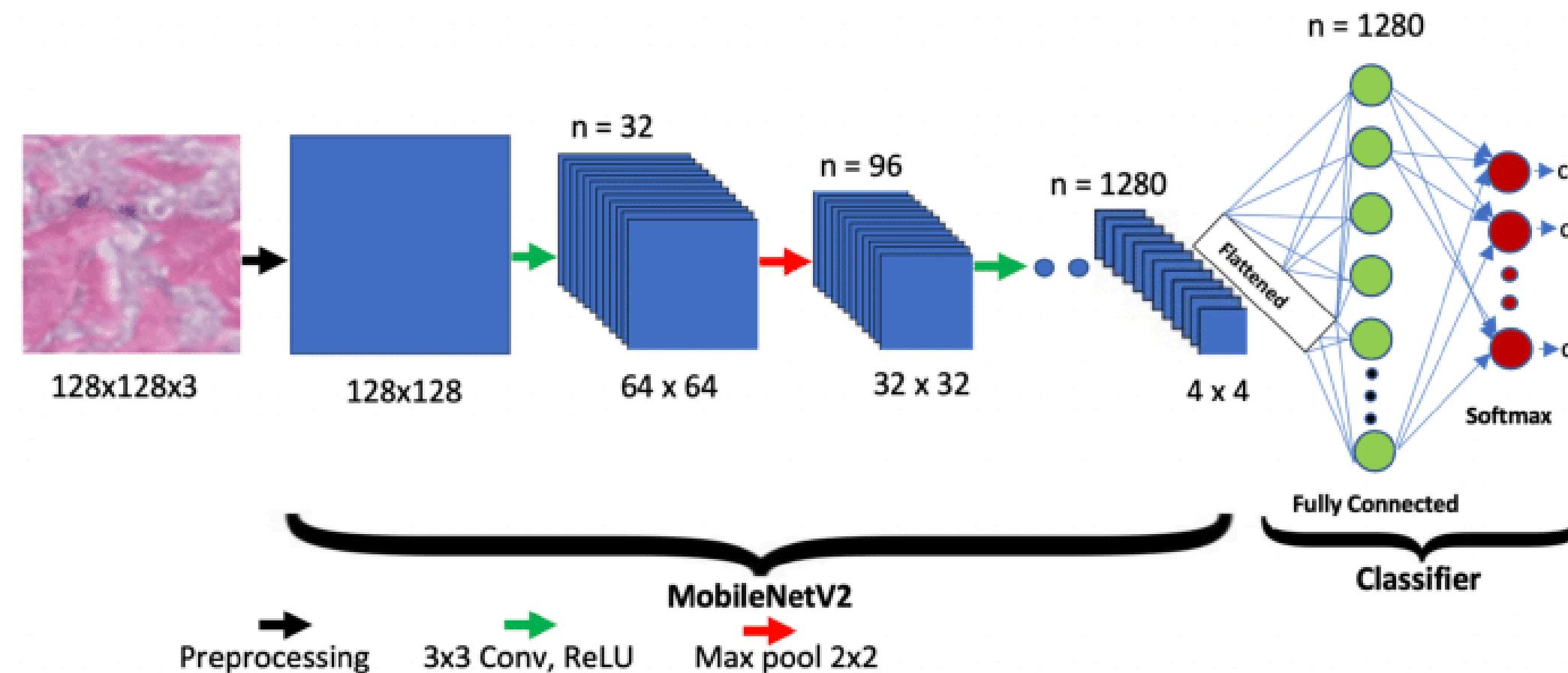
- **Chỉ dùng ảnh:** Accuracy và loss cải thiện ổn định, overfitting nhẹ.
- **Ảnh + metadata:** Train accuracy rất cao (~0.98) nhưng Val accuracy dừng ~0.88, loss validation dao động → overfitting nặng, mô hình phụ thuộc nhiều vào metadata.
- Metadata giúp mô hình học nhanh nhưng dễ làm giảm khả năng tổng quát hóa, cần regularization hoặc early stopping.

# MobileNet



# MobileNet

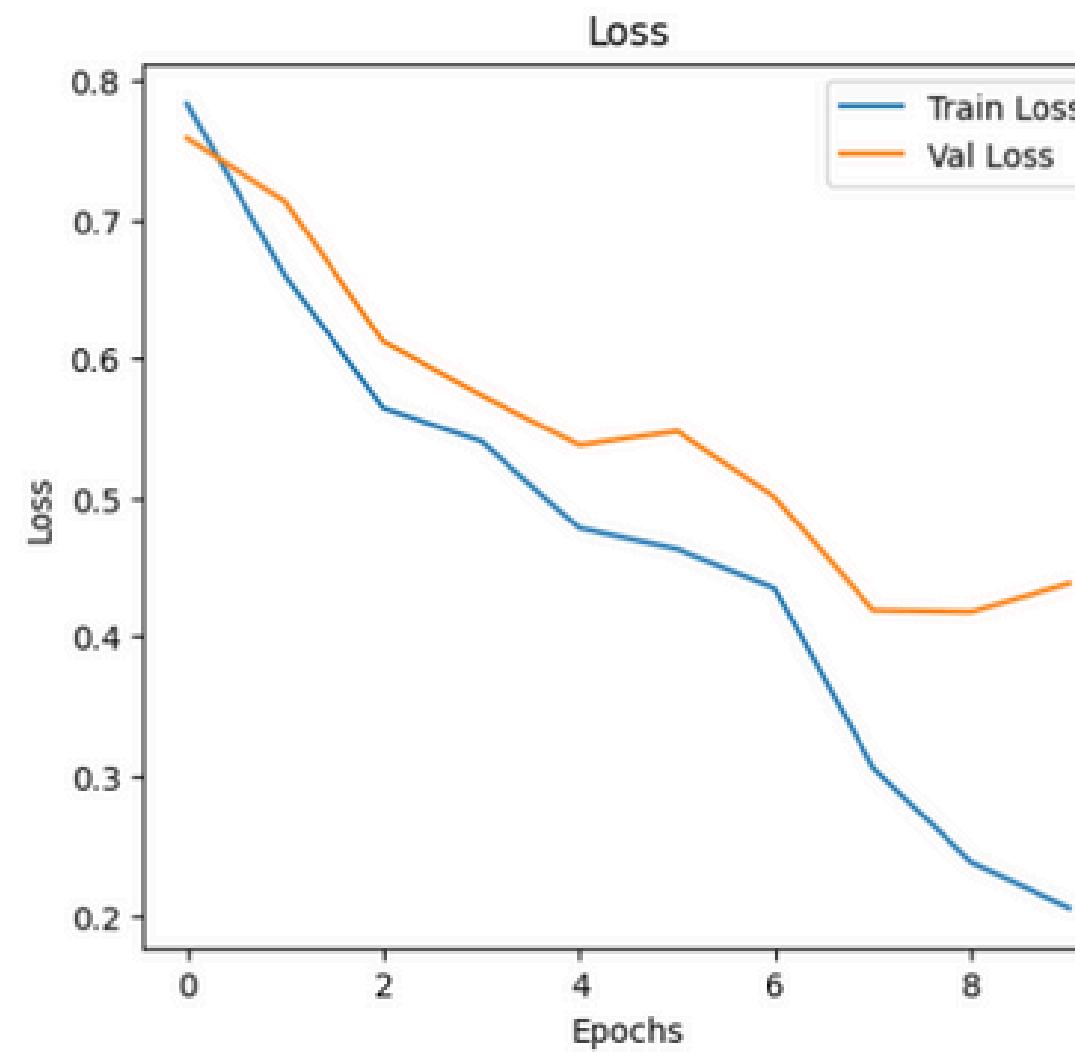
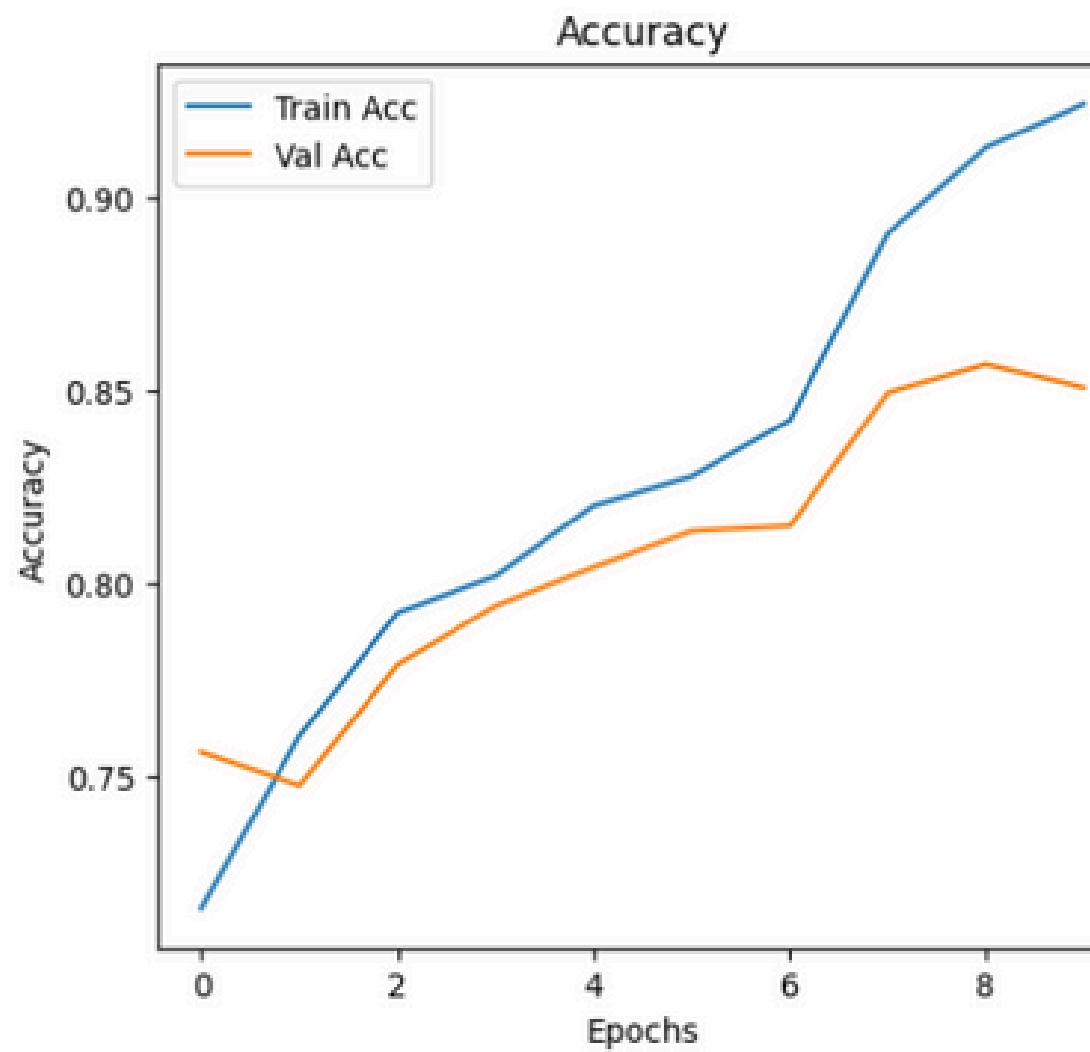
MobileNetV2 rất nhẹ (~3.4M tham số), tối ưu cho thiết bị di động và môi trường tính toán hạn chế, huấn luyện và suy luận nhanh. Tuy nhiên, độ chính xác thường thấp hơn các mô hình lớn như hay trên các tác vụ phức tạp.



# MobileNet - Model Ánh



```
train Loss: 0.2055 Acc: 0.9244  
val Loss: 0.4385 Acc: 0.8509
```

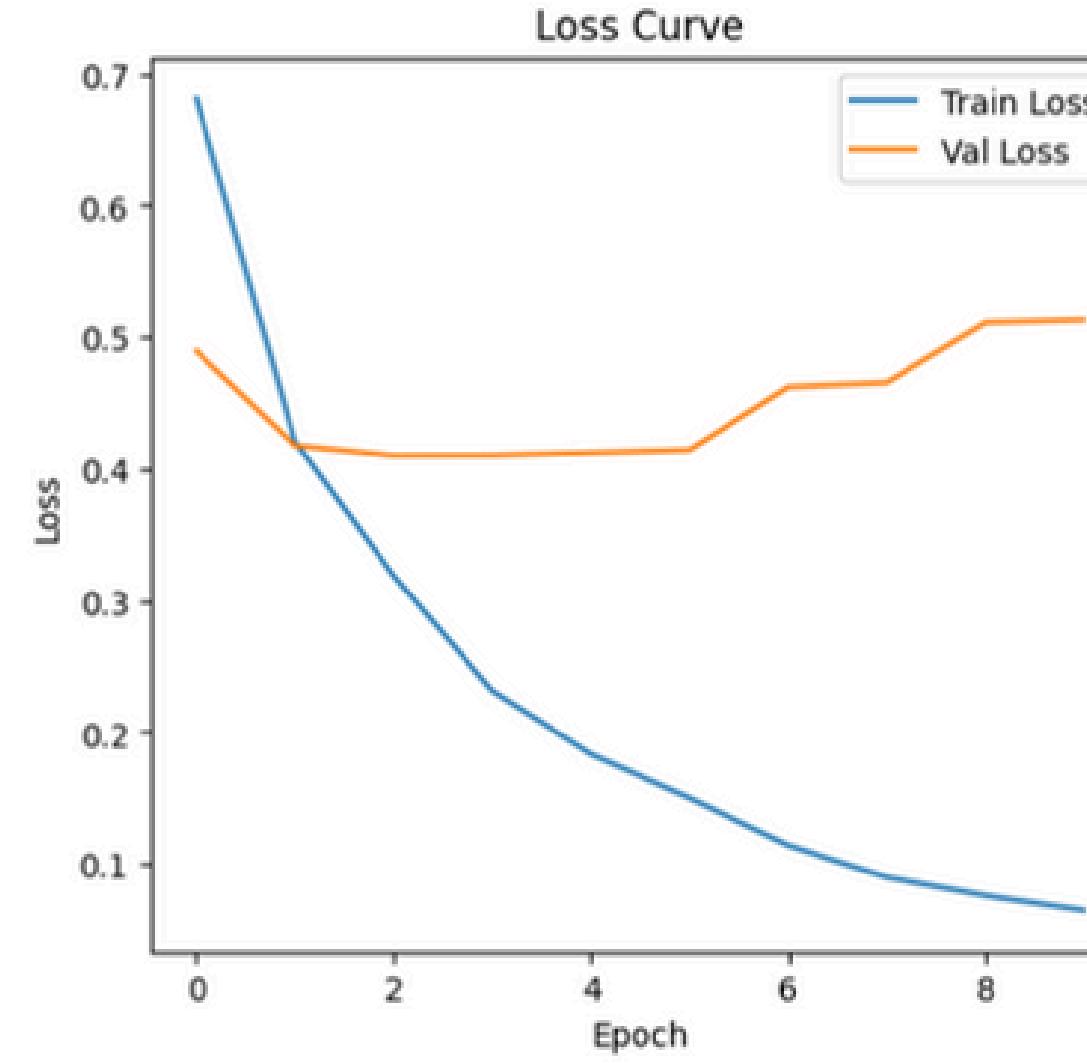
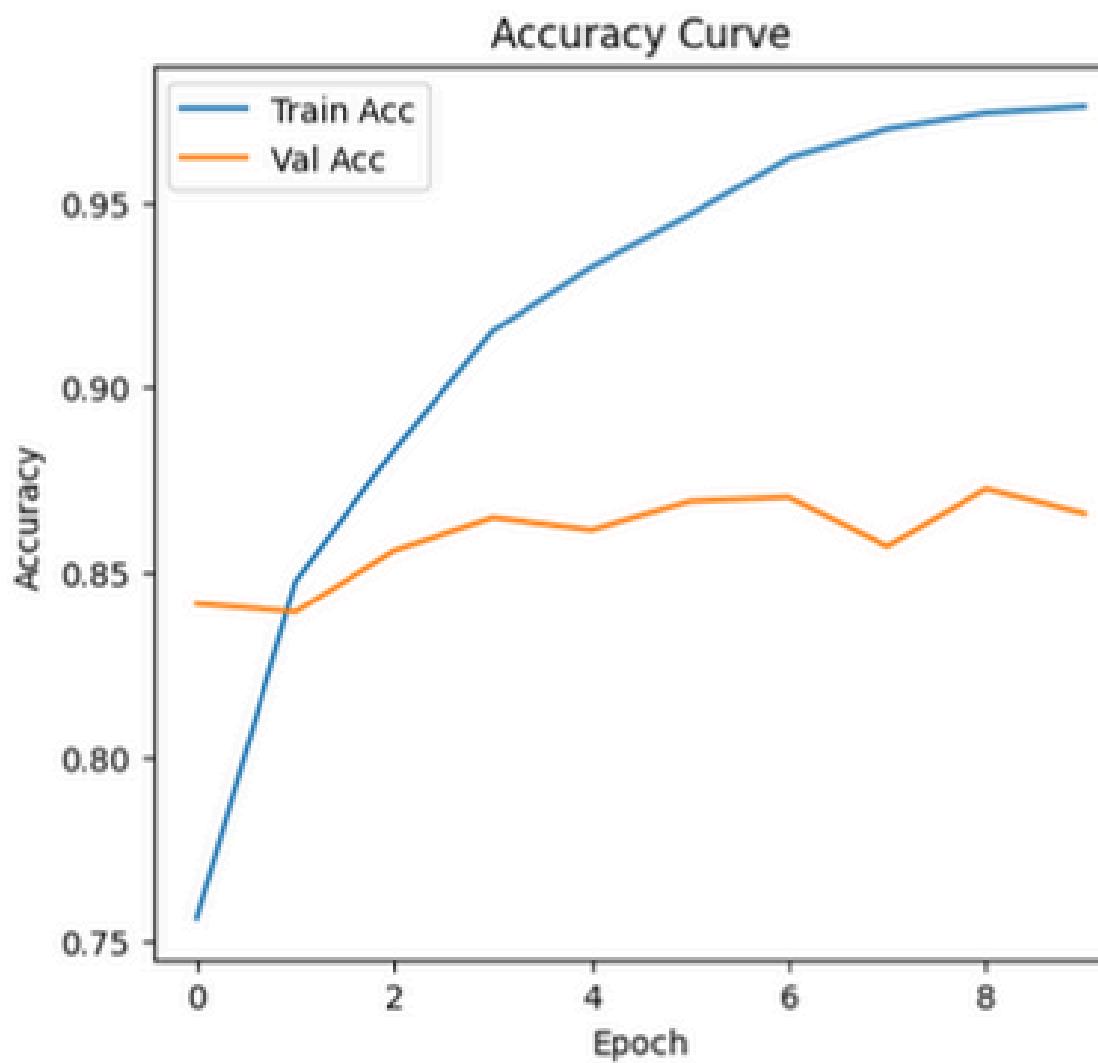


Classification Report:

	precision	recall	f1-score	support
akiec	0.59	0.61	0.60	49
bcc	0.78	0.73	0.75	77
bkl	0.66	0.73	0.69	165
df	0.78	0.41	0.54	17
mel	0.64	0.59	0.61	167
nv	0.93	0.94	0.93	1006
vasc	1.00	0.73	0.84	22
accuracy			0.85	1503
macro avg	0.77	0.68	0.71	1503
weighted avg	0.85	0.85	0.84	1503

# MobileNet - Model kết hợp Ảnh và Metadata

Train Loss: 0.0647 Train Acc: 0.9761 Val Loss: 0.5131 Val Acc: 0.8659



Classification Report:

	precision	recall	f1-score	support
akiec	0.53	0.88	0.66	33
bcc	0.80	0.71	0.75	51
bkl	0.85	0.69	0.76	110
df	0.90	0.75	0.82	12
mel	0.68	0.56	0.61	111
nv	0.93	0.97	0.95	671
vasc	0.92	0.86	0.89	14
accuracy			0.87	1002
macro avg	0.80	0.77	0.78	1002
weighted avg	0.87	0.87	0.87	1002



# MobileNet - Nhận xét biểu đồ

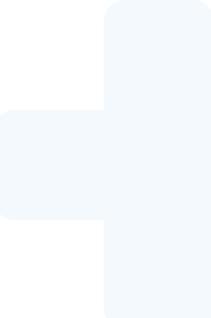
---

## ModelẢnh

- Accuracy train và val tăng ổn định, khoảng cách không quá lớn → overfitting nhẹ.
- Loss giảm đều cho cả train và val, nhưng từ epoch 6 trở đi train loss giảm nhanh hơn → dấu hiệu bắt đầu lệch.

## ModelẢnh + Metadata

- Train accuracy tăng rất nhanh, đạt ~0.98; val accuracy chỉ khoảng 0.87 và chững lại từ sớm → overfitting mạnh.
- Train loss giảm mạnh, val loss dao động và không giảm đáng kể sau vài epoch đầu.



# MobileNet - Nhận xét bảng báo cáo phân loại

---

## Chỉ ảnh:

- Train acc: 92.44%, Val acc: 85.09%
- Các lớp được dự đoán khá cân bằng, macro F1-score = 0.71
- Hiệu suất tốt nhưng vẫn còn khoảng cách giữa train và val → có chút overfitting nhẹ.

## Ảnh + metadata:

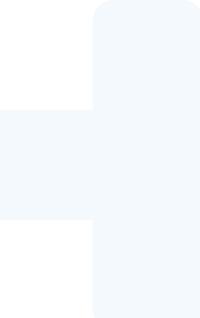
- Train acc: 97.61%, Val acc: 86.59%
- Macro F1-score tăng từ 0.71 → 0.78, recall cao hơn ở nhiều lớp, đặc biệt là các lớp hiếm như “akiec” và “df”.
- Mô hình phân loại tốt hơn cho hầu hết các lớp và vẫn giữ được cân bằng giữa precision và recall.



# MobileNet - Kết luận

---

- Thêm metadata giúp cải thiện độ chính xác tổng thể và đặc biệt nâng cao hiệu suất với các lớp hiếm (akiec, df) → macro F1 tăng mạnh.
- Tuy nhiên, biểu đồ cho thấy mô hình dễ overfit hơn khi thêm metadata (train acc gần tuyệt đối, val acc chững sớm, val loss không giảm).
- Metadata mang lại lợi ích lớn về cân bằng giữa các lớp, nhưng cần thêm regularization hoặc early stopping để giảm overfitting và cải thiện khả năng tổng quát hóa.



# Kết quả lựa chọn mô hình

Model	Input	Train Loss	Train Acc	Val Loss	Val Acc	Test Acc
ResNet50	Chỉ ảnh	0.5457	0.8009	0.6031	0.7763	0.79
	Ảnh + metadata	0.1008	0.9645	0.6581	0.8362	0.84
ResNet18	Chỉ ảnh	0.3207	0.8813	0.4720	0.8242	0.83
	Ảnh + metadata	0.0591	0.9807	0.5634	0.8602	0.84
EfficientNetB0	Chỉ ảnh	0.0141	0.9961	0.4799	0.8855	<b>0.89</b>
	Ảnh + metadata	0.0550	0.9827	0.4936	0.8703	<b>0.89</b>
VGG16	Chỉ ảnh	1.1353	0.6695	1.1297	0.6698	0.67
	Ảnh + metadata	0.2611	0.9033	0.5772	0.8362	0.84
DenseNet121	Chỉ ảnh	0.3369	0.8789	0.4371	0.8442	0.84
	Ảnh + metadata	0.0699	0.9756	0.4198	0.8847	0.88
MobileNetV2	Chỉ ảnh	0.2055	0.9244	0.4385	0.8509	0.85
	Ảnh + metadata	0.0647	0.9761	0.5131	0.8659	0.87

# Kết quả lựa chọn mô hình

---

Sau khi thử nghiệm 6 mô hình (ResNet50, ResNet18, EfficientNet-B0, VGG16, DenseNet121, MobileNetV2) với cùng điều kiện huấn luyện, với cả 2 trường hợp: chỉ có ảnh và có cả ảnh và metadata thì EfficientNet-B0 đều đạt kết quả tốt nhất mà không bị overfitting với Test Accuracy lần lượt là 0.8855 và 0.8703

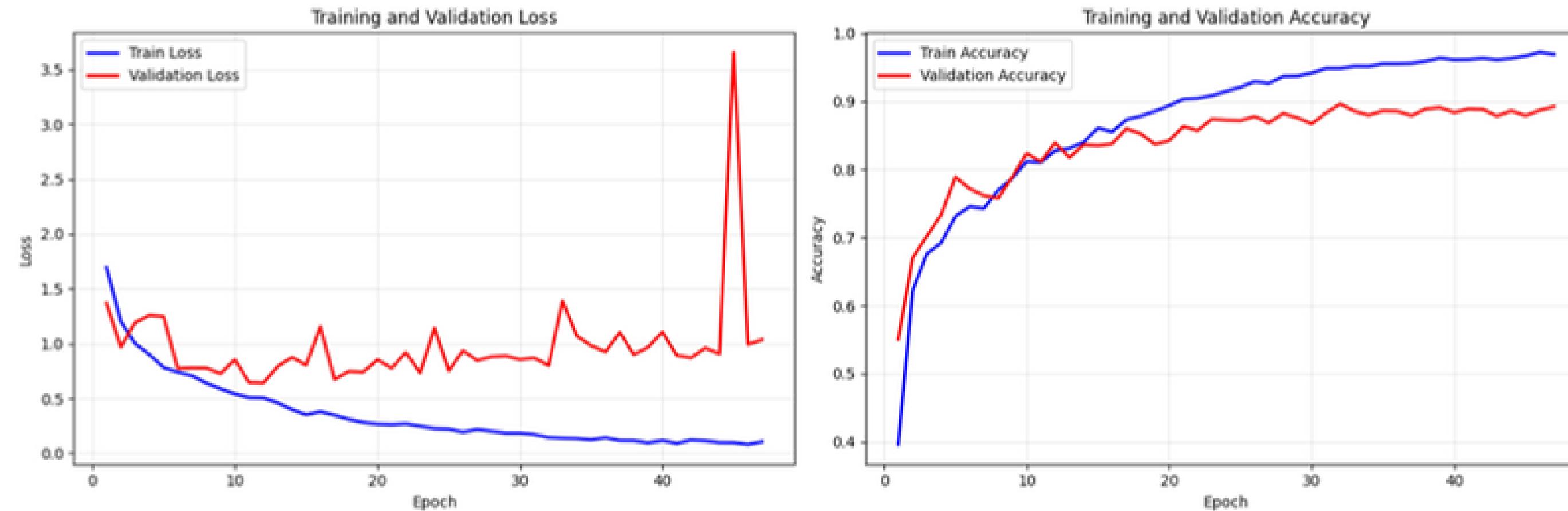
## Bước tiếp theo:

Nhóm đã fine-tune EfficientNet-B0 bằng cách:

- Sử dụng EfficientNet-B0 pretrained trên ImageNet
- Thay đổi kiến trúc classifier khi fine-tune
- Data augmentation
- Class balancing để cải thiện fine-tuning
- Sử dụng các kỹ thuật tối ưu: Optimizer, Learning Rate Scheduler, Gradient Clipping
- Early Stopping
- Metadata fusion



# Kết quả mô hình sau khi fine-tune



- Train Loss (xanh) giảm đều theo số epoch, cho thấy mô hình học tốt trên tập huấn luyện.
- Validation Loss (đỏ) ban đầu cũng giảm, nhưng dao động khá mạnh từ khoảng epoch 5 trở đi, và có một spike rất lớn gần epoch 45.
- Train Accuracy (xanh) tăng đều, đạt ~97% ở cuối quá trình.
- Validation Accuracy (đỏ) tăng nhanh ban đầu, sau đó bão hòa quanh ~89%.
- Khoảng cách giữa train và validation accuracy dần lớn hơn → dấu hiệu overfitting.





# Kết quả mô hình sau khi fine-tune

---

Test Accuracy: 0.8762

Detailed Classification Report:

	precision	recall	f1-score	support
akiec	0.7143	0.5932	0.6481	59
bcc	0.8353	0.8161	0.8256	87
bkl	0.7292	0.7500	0.7394	140
df	1.0000	0.8462	0.9167	13
mel	0.6645	0.6867	0.6754	150
nv	0.9420	0.9402	0.9411	1036
vasc	0.7200	1.0000	0.8372	18
accuracy			0.8762	1503
macro avg	0.8007	0.8046	0.7976	1503
weighted avg	0.8772	0.8762	0.8761	1503





# Kết quả mô hình sau khi fine-tune

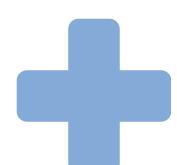
## Tổng quan kết quả

### Điểm mạnh:

- Các lớp ít dữ liệu như df và vasc cải thiện rõ rệt:
  - df: Precision tăng từ 0.83 → 1.00, F1 tăng từ 0.83 → 0.9167.
  - vasc: Recall tăng từ 0.86 → 1.00, F1 giảm từ 0.86 → 0.8372.
  - Điều này cho thấy fine-tune giúp mô hình học đặc trưng tốt hơn cho các lớp hiếm.
- Macro Recall giảm nhẹ (0.81 → 0.8046).
- Một số lớp giữ ổn định hiệu suất cao, như nv (Precision và Recall gần như không đổi).

### Điểm yếu:

- Accuracy chung giảm nhẹ (~1.38%), đặc biệt Precision/Recall của một số lớp như akiec, bkl, mel giảm rõ rệt.
- Lớp akiec: Recall giảm mạnh từ 0.76 → 0.5932 → mô hình bỏ sót nhiều mẫu hơn.
- Lớp bkl và mel: cả Precision và Recall đều giảm, cho thấy mô hình chưa tối ưu cho nhóm này.
- Lớp dữ liệu nhiều như nv không được cải thiện thêm, nghĩa là fine-tune chưa khai thác tối đa tiềm năng của backbone pretrained cho lớp trội.





# Kết quả mô hình sau khi fine-tune

---

Kết quả chưa tốt thể hiện rằng:

- **Mất cân bằng giữa các lớp vẫn tồn tại:** mặc dù lớp hiếm cải thiện, nhưng điều này đôi khi làm giảm hiệu suất lớp phổ biến do mô hình "nhường" sự ưu tiên.
- **Dấu hiệu overfitting cục bộ ở một số lớp hiếm:** Precision cao nhưng Recall thấp (hoặc ngược lại), như akiec sau fine-tune.
- **Backbone EfficientNet-B0 có thể chưa đủ mạnh** cho dataset đa dạng màu sắc và kết cấu như HAM10000, nhất là khi số epoch chưa đủ dài để khai thác triệt để.
- **Chiến lược data augmentation chưa chuyên biệt cho từng lớp**, nên những lớp khó phân biệt như bkl và mel vẫn bị nhầm lẫn.





# Định hướng tương lai

- **Cải thiện chiến lược cân bằng dữ liệu:**

- Kết hợp oversampling có điều kiện và mixup/cutmix augmentation để tăng dữ liệu ảo cho lớp hiếm.
- Thử focal loss để giảm ảnh hưởng của lớp dễ phân loại và tập trung vào lớp khó.

- **Điều chỉnh chiến lược fine-tune:**

- Giai đoạn 1: Freeze backbone, train classifier → tránh phá hỏng feature pretrained.
- Giai đoạn 2: Unfreeze từng phần backbone, giảm LR → giúp học sâu hơn mà không gây overfitting nhanh.
- Sử dụng discriminative learning rates (LR khác nhau cho backbone và classifier).

- **Nâng cấp backbone:**

- Thử EfficientNet-B3/B4, ConvNeXt, hoặc Swin Transformer để tăng khả năng nhận diện đặc trưng nhỏ.
- Kết hợp đa mô hình (ensemble) để tận dụng ưu điểm từng backbone.

- **Tối ưu data augmentation theo từng lớp:**

- Lớp hiếm: tăng mạnh augmentation để tránh overfitting.
- Lớp nhiều: augmentation nhẹ để giữ đặc trưng gốc.





# Kết luận

---

Fine-tune đã giúp cải thiện đáng kể các lớp hiếm nhưng gây suy giảm nhẹ ở một số lớp phổ biến, dẫn đến accuracy tổng thể giảm. Điều này phản ánh thách thức trong việc cân bằng giữa hiệu suất chung và hiệu suất cho lớp hiếm. Trong tương lai, cần kết hợp nhiều chiến lược cân bằng dữ liệu, cải tiến backbone và tối ưu quy trình fine-tune để đạt được mô hình vừa mạnh tổng thể, vừa công bằng giữa các lớp



# Thank You

