So sánh CNN, ResNet và Vision Transformers cho Phân loi Đa nhãn Bnh t nh X-quang Ngc

Đ án cui kỳ môn Deep Learning - Nhóm 1
Trng Kinh doanh - Thc sĩ Kinh t

Thành viên Nhóm 1

February 5, 2026

Abstract

Báo cáo này trình bày nghiên cu so sánh ba kin trúc deep learning chính cho bài toán phân loi đa nhãn bnh phi t nh X-quang ngc: Convolutional Neural Networks (CNN), Residual Networks (ResNet-34), và Vision Transformers (ViT). Chúng tôi thc hin các mô hình t đu bng PyTorch và so sánh vi mô hình ViT tin hun luyn t th vin `timm`. Kt qu cho thy: (1) ViT pretrained đt AUC cao nht (0.67), (2) các mô hình hun luyn t đu vi d liu nh cho hiu sut thp hn, (3) transfer learning là cn thit cho ng dng y khoa, (4) AUC là metric quan trng hn accuracy do class imbalance nghiêm trng. B d liu s dng là NIH Chest X-ray 14 vi 112,120 nh và 15 lp bnh.

T khóa: Vision Transformer, X-quang ngc, Phân loi đa nhãn, Deep Learning, Hình nh y khoa, Transfer Learning

Contents

Part I
Các Mô Hình Deep Learning

1 Mô hình CNN c s

1.1 Kin trúc mô hình

Mô hình CNN c s (`CNNClassifier`) đc thit k nh mt baseline đn gin đ so sánh vi các kin trúc phc tp hn. Kin trúc này bao gm hai khi tích chp (convolutional block) và mt b phân loi fully-connected.

Chi tit kin trúc:

- **Đu vào**: nh RGB kích thc $3 \times 224 \times 224$
- **Khi tích chp 1**:
  - Conv2d: 3 channels $\rightarrow$ 32 channels, kernel 3×3, không padding
  - Activation: ReLU
  - MaxPool2d: kernel 2×2
  - Kích thc sau khi: $32 \times 111 \times 111$
- **Khi tích chp 2**:
  - Conv2d: 32 channels $\rightarrow$ 64 channels, kernel 3×3, không padding
  - Activation: ReLU
  - MaxPool2d: kernel 2×2
  - Kích thc sau khi: $64 \times 54 \times 54$
- **B phân loi (Classifier)**:
  - Flatten: $64 \times 54 \times 54 = 186,624$ features
  - Linear: $186,624 \rightarrow 512$
  - ReLU
  - Linear: $512 \rightarrow 15$ (s lp bnh)

Tng s tham s: $\sim$95 triu tham s

Vn đ kin trúc: Vic s dng Flatten trc tip sau các lp tích chp dn đn lp fully-connected đu tiên có s tham s rt ln ($186,624 \times 512 \approx 95.5$M). Đây là mt thit k không hiu qu, nên đc thay th bng Global Average Pooling đ gim đáng k s tham s.

1.2 Cu hình hun luyn

Table 1: Cu hình hun luyn mô hình CNN

Tham s	Giá tr
Kích thc nh (Image Size)	224×224
Batch Size	32
Learning Rate	1×10^{-4}
Weight Decay	1×10^{-6}
S Epoch	10
Optimizer	AdamW
Loss Function	BCEWithLogitsLoss

Data Augmentation áp dng trong quá trình hun luyn:

- Horizontal Flip vi xác sut $p = 0.5$
- Random Rotation trong phm vi ± 5
- Color Jitter: brightness=0.1, contrast=0.1
- Normalization s dng mean và std ca ImageNet: $\mu = [0.485, 0.456, 0.406]$, $\sigma = [0.229, 0.224, 0.225]$

1.3 Kt qu thc nghim

Table 2: Kt qu hun luyn mô hình CNN (10 epoch)

Ch s	Train	Validation	Test
Loss (epoch cui)	0.2121	0.4208	–
AUC (epoch cui)	0.8961	0.5847	∼0.58
Best Val AUC	–	0.5998 (epoch 1)	–

Nhn xét:

- **Overfitting nghiêm trng**: Train AUC tăng t 0.53 lên 0.90 trong khi Val AUC gn nh không ci thin (dao đng quanh 0.56–0.60).
- **Best epoch là epoch 1**: Mô hình đt Best Val AUC = 0.5998 ngay t epoch đu tiên, sau đó validation performance gim dn.
- **Khong cách Train-Val gap**: Rt ln (∼0.32), cho thy mô hình hc thuc d liu hun luyn thay vì tng quát hóa.

1.4 Đon mã minh ha

```
class CNNClassifier(nn.Module):
    def __init__(self, num_classes=15):
        super(CNNClassifier, self).__init__()
        # Input: 3 x 224 x 224
        self.features = nn.Sequential(
            # Conv Block 1
            nn.Conv2d(3, 32, kernel_size=3, padding=0),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2),
            # Conv Block 2
            nn.Conv2d(32, 64, kernel_size=3, padding=0),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2)
        )
        # 224 -> 222 -> 111 -> 109 -> 54
        self.flatten_size = 64 * 54 * 54  # = 186,624

        self.classifier = nn.Sequential(
            nn.Flatten(),
            nn.Linear(self.flatten_size, 512),
            nn.ReLU(),
            nn.Linear(512, num_classes)
        )

    def forward(self, x):
        x = self.features(x)
        x = self.classifier(x)
        return x
```

Listing 1: Đnh nghĩa lp CNNClassifier

2 Mô hình ResNet-34

2.1 Kin trúc mô hình

Mô hình ResNet-34 đc xây dng t đu (from scratch) theo kin trúc gc ca He et al. (2016). Đc đim quan trng nht ca ResNet là **skip connections** (residual connections), cho phép gradient lan truyn trc tip qua nhiu lp, gii quyt vn đ vanishing gradient trong các mng sâu.

Cu trúc BasicBlock:

- Conv2d $(3 \times 3) \rightarrow$ BatchNorm2d $\rightarrow$ ReLU
- Conv2d $(3 \times 3) \rightarrow$ BatchNorm2d
- Skip connection: $output = F(x) + x$
- ReLU activation cui cùng

Cu hình ResNet-34:

Table 3: Cu hình các layer trong ResNet-34

Layer	Channels	Blocks	Stride	Output Size
Conv1 (7×7)	64	–	2	112×112
MaxPool	64	–	2	56×56
Layer1	64	3	1	56×56
Layer2	128	4	2	28×28
Layer3	256	6	2	14×14
Layer4	512	3	2	7×7
AvgPool	512	–	–	1×1
FC	15	–	–	15 classes

Tng s tham s: $\sim$21 triu tham s
u đim so vi CNN baseline:

- S dng **Global Average Pooling** thay vì Flatten $\Rightarrow$ gim đáng k s tham s
- **Batch Normalization** n đnh quá trình hun luyn
- **Skip connections** giúp hun luyn mng sâu hiu qu hn
- **Kaiming initialization** cho các lp tích chp

2.2 Cu hình hun luyn

Table 4: Cu hình hun luyn mô hình ResNet-34

Tham s	Giá tr
Kích thc nh	224×224
Batch Size	32
Learning Rate	1×10^{-4}
Weight Decay	1×10^{-6}
S Epoch	10
Optimizer	AdamW
Loss Function	BCEWithLogitsLoss
Weight Initialization	Kaiming Normal

2.3 Kt qu thc nghim

Table 5: Kt qu hun luyn mô hình ResNet-34 (10 epoch)

Ch s	Train	Validation	Test
Loss (epoch cui)	0.2786	0.3742	–
AUC (epoch cui)	0.7768	0.5235	∼0.53
Best Val AUC	–	0.5293 (epoch 6)	–

Nhn xét:

- **Hiu sut thp hn kỳ vng**: Val AUC ch đt 0.53, thp hn CNN baseline (0.60).
- **Vn có overfitting**: Train AUC (0.78) cao hn nhiu so vi Val AUC (0.52).
- **Nguyên nhân có th**:
  - Mô hình hun luyn t đu cn nhiu d liu hn
  - Không s dng pretrained weights t ImageNet
  - S mu hun luyn nh (100 nh sample)

So sánh vi CNN:

- ResNet-34 có ít tham s hn nhiu (21M vs 95M)
- Tuy nhiên, vi lng d liu nh, mô hình đn gin nh CNN li cho kt qu tt hn
- Điu này gi ý rng cn s dng transfer learning cho các mô hình sâu

2.4 Đon mã minh ha

```
class BasicBlock(nn.Module):
    expansion = 1

    def __init__(self, in_channels, out_channels, stride=1, downsample=None):
        super(BasicBlock, self).__init__()
        self.conv1 = nn.Conv2d(in_channels, out_channels,
                               kernel_size=3, stride=stride, padding=1, bias=
    False)
        self.bn1 = nn.BatchNorm2d(out_channels)
        self.relu = nn.ReLU(inplace=True)
        self.conv2 = nn.Conv2d(out_channels, out_channels,
                               kernel_size=3, stride=1, padding=1, bias=False)
        self.bn2 = nn.BatchNorm2d(out_channels)
        self.downsample = downsample

    def forward(self, x):
        identity = x

        out = self.conv1(x)
        out = self.bn1(out)
        out = self.relu(out)
        out = self.conv2(out)
        out = self.bn2(out)

        if self.downsample is not None:
            identity = self.downsample(x)

        out += identity  # Skip connection
        out = self.relu(out)
        return out
```

```
30
31 def create_resnet34(num_classes=15):
32     """Tao mo hinh ResNet-34 voi cau hinh [3,4,6,3] blocks"""
33     return ResNet(BasicBlock, [3, 4, 6, 3], num_classes=num_classes)
```

Listing 2: Đnh nghĩa BasicBlock và hàm _make_layer

3 Mô hình Vision Transformer (ViT-v1 và ViT-v2)

Vision Transformer (ViT) là kin trúc áp dng c ch self-attention ca Transformer (Vaswani et al., 2017) vào bài toán th giác máy tính. Thay vì s dng các lp tích chp, ViT chia nh thành các patch và x lý chúng nh mt chui (sequence) các token.

3.1 Kin trúc ViT-v1

Cu hình mô hình:

Table 6: Cu hình kin trúc ViT-v1 và ViT-v2

Tham s	**Giá tr**
Image Size	224×224
Patch Size	32×32
S Patches	$(224/32)^2 = 49$
Embedding Dimension	64
Number of Heads	4
Transformer Layers (Depth)	8
MLP Ratio	2
Dropout	0.1
MLP Head Units	[2048, 1024]

Tng s tham s: ∼9.0 triu tham s
Các thành phn chính:

1. **Patch Embedding**: S dng Conv2d vi kernel = patch_size đ chiu các patch thành embedding vectors.
2. **Positional Embedding**: Learnable positional embedding đc cng vào patch embeddings.
3. **Transformer Encoder**: 8 layers, mi layer gm:
   - Layer Normalization (Pre-LN)
   - Multi-Head Self-Attention (4 heads)
   - Residual Connection
   - Layer Normalization
   - MLP (GELU activation)
   - Residual Connection
4. **Classification Head**: Flatten tt c patches → MLP [2048, 1024, 15]

Lu ý quan trng: Mô hình này **không s dng CLS token** nh trong kin trúc ViT gc. Thay vào đó, tt c patch embeddings sau Transformer đc flatten và đa vào MLP classifier.

3.2 Kin trúc ViT-v2 và khác bit so vi v1

ViT-v2 có **cùng kin trúc** vi ViT-v1, nhng khác bit **cu hình hun luyn**:

Table 7: So sánh cu hình hun luyn ViT-v1 và ViT-v2

Tham s	**ViT-v1**	**ViT-v2**
Optimizer	AdamW	SGD (Nesterov)
Learning Rate	1×10^{-4}	0.01
Weight Decay	1×10^{-6}	1×10^{-5}
LR Scheduler	Không	ReduceLROnPlateau
Early Stopping	Không	Có (patience=3)
Momentum	–	0.9

Các ci tin trong ViT-v2:

- **Optimizer SGD vi Nesterov momentum**: Learning rate cao hn (0.01) kt hp vi momentum=0.9 giúp hi t nhanh hn.
- **Weight decay mnh hn** (10^{-5} thay vì 10^{-6}): Tăng regularization đ gim overfitting.
- **Learning rate scheduler**: ReduceLROnPlateau gim LR khi validation loss không ci thin.
- **Early stopping**: Dng hun luyn sm nu không ci thin sau 3 epoch liên tip.

3.3 Kt qu thc nghim

Table 8: So sánh kt qu ViT-v1 và ViT-v2

Mô hình	**Best Val AUC**	**Best Epoch**	**Test AUC**	**Test Acc**
ViT-v1	0.6431	4	0.5854	91.33%
ViT-v2	0.5947	9	0.6303	89.67%

Kt qu AUC theo tng bnh (Per-class AUC):

Table 9: Per-class AUC ca ViT-v1 và ViT-v2 trên tp Test

Bnh	**ViT-v1 AUC**	**ViT-v2 AUC**
Cardiomegaly	0.79	**1.00**
Emphysema	–	**0.84**
Effusion	0.51	**0.73**
Nodule	**0.68**	–
Pneumothorax	**0.84**	0.69
Atelectasis	**0.47**	0.32
Pleural_Thickening	0.42	**0.68**
Mass	0.11	–
Edema	–	0.42
Consolidation	**0.61**	0.33
Infiltration	**0.89**	0.52
No Finding	0.53	**0.76**

Nhn xét quan trng:

- **ViT-v1 và ViT-v2 có đim mnh b sung cho nhau**: ViT-v1 tt hn Infiltration (0.89), Pneumothorax (0.84), trong khi ViT-v2 tt hn Cardiomegaly (1.00), Emphysema (0.84), và No Finding (0.76).
- **Tim năng ensemble**: Kt hp hai mô hình có th ci thin hiu sut tng th.
- **Các bnh khó phát hin**: Mass (AUC=0.11 ViT-v1), Consolidation (0.33 ViT-v2), Atelectasis (0.32 ViT-v2) có AUC rt thp, cho thy cn ci thin đáng k.

3.4 Đon mã minh ha

```
class PatchEmbedding(nn.Module):
    def __init__(self, img_size=224, patch_size=32, in_channels=3, embed_dim=64)
    :
        super().__init__()
        self.num_patches = (img_size // patch_size) ** 2
        # Su dung Conv2d de chieu patch thanh embedding
        self.proj = nn.Conv2d(in_channels, embed_dim,
                              kernel_size=patch_size, stride=patch_size)

    def forward(self, x):
        x = self.proj(x)          # (B, embed_dim, H/P, W/P)
        x = x.flatten(2)          # (B, embed_dim, num_patches)
        x = x.transpose(1, 2)     # (B, num_patches, embed_dim)
        return x

class TransformerEncoderBlock(nn.Module):
    def __init__(self, embed_dim, num_heads, mlp_ratio=4, dropout=0.1):
        super().__init__()
        self.ln1 = nn.LayerNorm(embed_dim, eps=1e-6)
        self.attn = nn.MultiheadAttention(embed_dim, num_heads,
                                          dropout=dropout, batch_first=True)
        self.ln2 = nn.LayerNorm(embed_dim, eps=1e-6)
        self.mlp = MLP(embed_dim, int(embed_dim * mlp_ratio), embed_dim, dropout
    )

    def forward(self, x):
        # Pre-LN + Self-Attention + Residual
        x_norm = self.ln1(x)
        attn_out, _ = self.attn(x_norm, x_norm, x_norm)
        x = x + attn_out
        # Pre-LN + MLP + Residual
        x = x + self.mlp(self.ln2(x))
        return x
```

Listing 3: Đnh nghĩa PatchEmbedding và TransformerEncoderBlock

4 Mô hình ViT tin hun luyn (Pretrained ViT)

4.1 Kin trúc mô hình

Mô hình ViT tin hun luyn s dng th vin `timm` (PyTorch Image Models) đ ti trng s đã đc hun luyn trên ImageNet. Đây là phng pháp **transfer learning**, tn dng kin thc đã hc t tp d liu ln đ gii quyt bài toán y khoa vi ít d liu hn.

Model đc s dng: `vit_base_patch16_224`

Table 10: Cu hình kin trúc ViT-Base/16

Tham s	**Giá tr**
Model Name	vit_base_patch16_224
Pretrained Dataset	ImageNet-21K
Image Size	224×224
Patch Size	16×16
S Patches	$(224/16)^2 = 196$
Embedding Dimension	768
Number of Heads	12
Transformer Layers	12
MLP Ratio	4

Tng s tham s: ∼86 triu tham s
Thay đi classification head:

- Head gc: Linear $768 \rightarrow 1000$ (ImageNet classes)
- Head mi: Linear $768 \rightarrow 15$ (15 bnh phi)

So sánh vi ViT t đu:

Table 11: So sánh ViT Pretrained và ViT from Scratch

Thuc tính	**ViT Scratch**	**ViT Pretrained**
Patch Size	32	16
S Patches	49	196
Embedding Dim	64	768
Transformer Layers	8	12
S tham s	9M	86M
Pretrained	Không	ImageNet-21K

4.2 Cu hình hun luyn

Table 12: Cu hình hun luyn ViT Pretrained

Tham s	**Giá tr**
Batch Size	16
Learning Rate	1×10^{-4}
Optimizer	AdamW
Loss Function	BCEWithLogitsLoss
S Epoch	10
Fine-tuning	Toàn b mô hình

Lu ý v Batch Size: Batch size gim t 32 xung 16 do mô hình ln hn đáng k (86M params vs 9M params), đòi hi nhiu GPU memory hn.

4.3 Kt qu thc nghim

Table 13: Kt qu hun luyn ViT Pretrained

Ch s	Train	Validation	Test
Loss (epoch cui)	0.2425	0.3232	0.3768
Accuracy	90.22%	86.67%	87.00%
AUC (epoch cui)	0.8820	0.6673	**0.6694**
Best Val AUC	–	0.6836 (epoch 4)	–

So sánh tng hp tt c các mô hình:

Table 14: So sánh hiu sut tt c các mô hình

Mô hình	Params	Best Val AUC	Test AUC	Test Acc	Xp hng
CNN Baseline	95M	0.5998	∼0.58	∼89%	4
ResNet-34	21M	0.5293	∼0.53	∼91%	5
ViT-v1	9M	0.6431	0.5854	91.33%	3
ViT-v2	9M	0.5947	0.6303	89.67%	2
ViT Pretrained	86M	**0.6836**	**0.6694**	87.00%	**1**

Nhn xét chính:

1. **ViT Pretrained đt AUC cao nht** (0.6694), xác nhn hiu qu ca transfer learning trong y khoa.
2. **Khong cách gia pretrained và scratch**: Test AUC tăng t 0.58–0.63 (scratch) lên 0.67 (pretrained), ci thin khong 7-15%.
3. **Trade-off gia tham s và hiu sut**: ViT-v1/v2 ch có 9M tham s nhng đt AUC gn bng pretrained (86M params).
4. **Accuracy không phn ánh đúng cht lng mô hình**: CNN đt accuracy cao (89%) nhng AUC thp (0.58), cho thy mô hình ch yu d đoán "No Finding" (chim 53.8% d liu).

4.4 Đon mã minh ha

```
import timm
import torch.nn as nn

# Tao mo hinh ViT pretrained tu thu vien timm
model = timm.create_model('vit_base_patch16_224', pretrained=True)

# Thay the classification head cho bai toan 15 lop benh
num_classes = 15
model.head = nn.Linear(model.head.in_features, num_classes)

# Chuyen model len GPU
model = model.to(device)

# In thong tin mo hinh
total_params = sum(p.numel() for p in model.parameters())
trainable_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
print(f"Total parameters: {total_params:,}")       # ~86M
```

```
print(f"Trainable parameters: {trainable_params:,}")
```

Listing 4: To mô hình ViT Pretrained vi timm

```
def compute_auc_safe(targets, outputs, num_classes):
    """Tinh AUC an toan, xu ly truong hop chi co 1 label value"""
    # Tim cac class co ca mau positive va negative
    valid_classes = []
    for i in range(num_classes):
        unique_labels = np.unique(targets[:, i])
        if len(unique_labels) > 1:  # Can ca 0 va 1
            valid_classes.append(i)

    if len(valid_classes) == 0:
        return 0.0

    # Tinh macro AUC chi tren cac class hop le
    auc = roc_auc_score(
        targets[:, valid_classes],
        outputs[:, valid_classes],
        average='macro'
    )
    return auc
```

Listing 5: Hàm tính AUC vi x lý trng hp đc bit

Part II
Tng Kt và Khuyn Ngh

5 Tng kt kt qu

Qua quá trình thc nghim vi 5 mô hình khác nhau trên b d liu NIH Chest X-ray 14, chúng tôi rút ra các kt lun sau:

1. **Transfer learning là cn thit**: Mô hình ViT pretrained cho kt qu tt nht (AUC = 0.67), cao hn đáng k so vi các mô hình hun luyn t đu.
2. **Kin trúc đn gin có th hiu qu vi d liu nh**: CNN baseline (95M params) đt AUC tng đng ViT scratch (9M params) trên tp d liu sample nh.
3. **AUC là metric quan trng hn accuracy**: Accuracy cao (89-91%) không phn ánh chính xác kh năng phân loi bnh do class imbalance nghiêm trng.
4. **Các bnh him cn đc chú ý đc bit**: Mass, Hernia, Pneumonia có prevalence $< 2\%$ và AUC thng thp hn 0.5.

6 Khuyn ngh ci tin

6.1 V kin trúc mô hình

- S dng Global Average Pooling thay vì Flatten trong CNN
- Áp dng pretrained ResNet hoc EfficientNet cho so sánh công bng hn
- Th nghim ViT-Small hoc DeiT đ cân bng gia hiu sut và chi phí tính toán

6.2 V hun luyn

- S dng Focal Loss hoc Weighted BCE đ x lý class imbalance
- Áp dng Cosine Annealing LR scheduler
- Tăng s epoch và s dng early stopping vi patience ln hn
- Hun luyn trên toàn b dataset (112,120 nh) thay vì sample

6.3 V d liu

- S dng patient-level split đ tránh data leakage
- Gim horizontal flip probability (0.3 thay vì 0.5) do đc thù nh y khoa
- Áp dng thêm augmentation: MixUp, CutMix, AutoAugment

6.4 V đánh giá

- Báo cáo per-class AUC cho tt c 15 bnh
- S dng bootstrap đ tính confidence interval
- Thc hin cross-validation đ đánh giá robustness

Tài liu tham kho

References

[1] Jain, A., Bhardwaj, A., Murali, K., & Surani, I. (2024). *A Comparative Study of CNN, ResNet, and Vision Transformers for Multi-Classification of Chest Diseases*. arXiv preprint arXiv:2406.00237.

[2] Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., & Summers, R. M. (2017). *ChestX-ray8: Hospital-scale chest x-ray database and benchmarks*. IEEE CVPR.

[3] Dosovitskiy, A., et al. (2020). *An image is worth 16x16 words: Transformers for image recognition at scale*. ICLR 2021.

[4] He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition*. IEEE CVPR.

[5] Vaswani, A., et al. (2017). *Attention is all you need*. NeurIPS 2017.