INT3412E 20 - Computer Vision: Midterm

ImageNet Classification with Deep Convolutional Neural Networks

Nhóm 4

Lưu Văn Đức Thiệu, Hoàng Thái Quang
Phạm Thu Trang, Hồ Thị Thanh Bình

1 Lời mở đầu

Sau khi tìm hiểu chi tiết về bài báo "ImageNet Classification with Deep Convolutional Neural Networks" của Alex Krizhevsky và cộng sự, nhóm 4 quyết định tiếp cận bài toán theo hướng so sánh mô hình được đề xuất trong bài (AlexNet) với các mô hình hiện đại hơn: VGG, Resnet, ConvNeXt lần lượt được đề xuất trong bài báo "Very deep convolutional networks for large-scale image recognition", "Deep residual learning for image recognition", "A ConvNet for the 2020s", các mô hình này đều giải quyết bài toán phân loại, định vị vật thể và đều được huấn luyện và đánh giá trên bộ dữ liệu của ImageNet 1K và ImageNet 22K. Để đơn giản hóa bài toán, nhóm chỉ so sánh kết quả của các mô hình trong bài toán phân loại; nhóm đã implement lại toàn bộ mô hình AlexNet và các mô hình so sánh cho bộ dữ liệu "Intel Image Classification" trên nền tảng Kaggle.

2 Tổng quan mô hình AlexNet

Vào thời điểm năm 2012, rất hiếm có kiến trúc mô hình nào có khả năng học tập tốt một bộ dữ liệu lớn có hàng nghìn đối tượng từ hàng triệu hình ảnh. Để giải quyết vấn đề này, các tác giả của bài báo đã đề xuất một mạng lưới thần kinh tích chập sâu lớn - AlexNet. Bên cạnh đó họ cũng đề xuất thêm các kỹ thuật để giảm overfit và giúp quá trình huấn luyện mô hình được nhanh hơn.

2.1 Dữ liệu

Bộ dữ liệu huấn luyện được các tác giả của AlexNet sử dụng là bộ dữ liệu lấy từ các cuộc thi ILSVRC-2010 và ILSVRC-2012. Bộ dữ liệu này gồm các bức ảnh được phân thành 1000 loại khác nhau. Trước khi được đưa vào huấn luyện các tác giả đã tiền xử lí dữ liệu bằng cách:

- Scale và cắt trung tâm để lấy ảnh 256x256.
- Trừ các giá trị pixel hình ảnh trung bình (Chuẩn hóa)

2.2 Kiến trúc mô hình AlexNet

Kiến trúc mô hình AlexNet [5] là một mạng neural có 60 triệu tham số và 650.000 nơ-ron, bao gồm tám lớp có trọng số, năm lớp đầu tiên là các lớp tích chập, ba lớp còn lại là fully-connected. Đầu ra của lớp fully-connected cuối cùng được đưa vào một softmax 1000 chiều tạo ra phân phối trên 1000 label. Sau hai lớp tích chập đầu tiên và thứ hai, có các lớp response-normalization. Các lớp max-pooling được đặt sau cả hai lớp response-normalization và lớp tích chập thứ năm. Hàm phi tuyến ReLU được áp dụng cho đầu ra của mỗi lớp tích chập và lớp fully-connected.

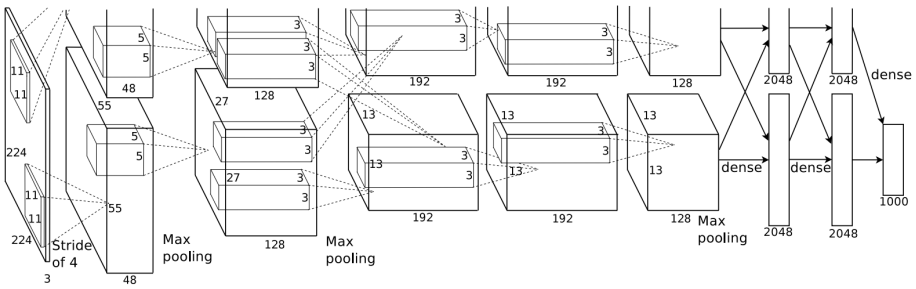

Figure 1: Minh hoạ kiến trúc Alexnet

Một số đặc trưng nổi bật trong kiến trúc mô hình bao gồm

- Sử dụng hàm phi tuyến ReLU [8] làm hàm kích hoạt thay vì hàm tanh để cải thiện thời gian huấn luyện và ngăn chặn việc overfitting
- Kích thước batch là 128
- Sử dụng 2 GPU song song
- Thêm lớp response-normalization cục bộ có thể giúp mô hình tổng quát hóa tốt hơn.
- Thay thế phương pháp pooling địa phương truyền thống thường được sử dụng trong CNN bằng pooling chồng chéo để giảm overfit.

2.3 Khắc phục vấn đề overfit

Một trong những vấn đề gặp phải trong quá trình huấn luyện mô hình AlexNet là overfit. Để giải quyết vấn đề này các tác giả đã đề xuất 2 phong pháp là tăng cường sữ liệu và sử dụng dropout.

2.3.1 Tăng cường dữ liệu

Tạo ra ảnh tham số từ tập dữ liệu gốc với tính toán ít nhất mà không cần lưu trong đĩa. Có 2 kỹ thuật:

- Tạo ra chuyển đổi ảnh và phản chiếu theo chiều ngang–>mở rộng đáng kể tập dữ liệu huấn luyện. Trong lúc test, mạng lưới trung tâm tạo dự đoán bởi chiết xuất 10 bản và trung bình dự đoán được tạo ra bởi lớp softmax.
- Điều chỉnh cường độ của các kênh RGB trong việc huấn luyện hình ảnh bằng cách sử dụng phân tích thành phần chính (PCA). Cách tiếp cận này làm giảm tỷ lệ lỗi top 1 hơn 1% và ghi lại một thuộc tính quan trọng của hình ảnh tự nhiên, tính xác mính đối tượng đó vẫn còn bất biến dưới sự thay đổi của độ chiếu sáng.

2.3.2 Dropout

- Bao gồm vô hiệu hóa ngẫu nhiên 50% số nơ-ron ẩn bằng cách thiết lập đầu ra về không. Những neuron bị vô hiệu hóa này không tham gia vào quá trình chuyển tiếp hay chuyển về trong quá trình huấn luyện. Kết quả là mạng lấy mẫu hiệu quả các kiến trúc khác nhau với mỗi đầu vào, mặc dù tất cả các kiến trúc đều có cùng trọng số.
- Kỹ thuật này làm giảm sự đồng thích ứng phức tạp của các neuron vì tế bào thần kinh không thể dựa vào sự hiện diện của các tế bào thần kinh cụ thể khác. Do đó, buộc phải tìm hiểu các tính năng mạnh mẽ hơn, hữu ích khi kết hợp với nhiều tập hợp con ngẫu nhiên khác nhau của các nơ-ron khác.

2.4 Kết quả đạt được

Bảng 1 và 2 lần lượt là các kết quả của AlexNet đạt được trong 2 cuộc thi ILSVRC-2010 và ILSVRC-2012

Model	Top-1	Top-5
Sparse coding	47.1%	28.2%
SIFT + FVs	45.7%	25.7%
CNN	37.5%	17.0%

Table 1: Comparison of results on ILSVRC-2010 test set.

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs	—	—	26.2%
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	16.4%
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	15.3%

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. Models with an asterisk* were "pre-trained" to classify the entire ImageNet 2011 Fall release. See Section 6 for details.

3 Tổng quan mô hình so sánh

3.1 VGGNet

VGGNet [9] (Viola-Jones Global Gradients Network) là một mô hình mạng nơ-ron sâu (deep neural network) tập trung vào ảnh hưởng của độ sâu mạng nơ-ron tích chập đối với độ chính xác của nó.

- Thay vì sử dụng một kernel size lớn, VGG chỉ sử dụng các kernel size 3x3 cho tất cả lớp Convolution, với stride 1 và padding 1.
- VGG sử dụng các lớp Convolution liên tiếp nhau để học đặc trưng, và các lớp Max Pooling để down-sampling, các lớp Max Pooling có kernel size 2, stride 2. Khi đưa qua lớp Max Pooling này, kích thước của ảnh sẽ giảm đi 1/2.
- Cuối VGGNet có ba lớp fully-connected. Hai lớp đầu mỗi lớp đều có 4096 kênh, trong khi lớp thứ ba chứa 1000 kênh, mỗi kênh tương ứng với một lớp.
- Sử dụng hàm kích hoạt ReLU cho tất cả các lớp ẩn.
- Số lượng tham số cực lớn với hơn 128 triệu tham số, không gian lưu trữ của mô hình ước tính khoảng trên 600Mb.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 2: Kiến trúc mô hình VGGNet

3.2 ResNet

Mạng ResNet[2] là một mạng CNN được thiết kế để làm việc với hàng trăm lớp. Sự ra đời của mạng ResNet đã góp phần giải quyết được vấn đề Vanishing Gradient khi xây dựng mạng CNN với nhiều lớp tích chập sẽ dẫn tới quá trình học tập không tốt.

- Mạng ResNet xây dựng dựa trên mạng VGG 19. Tuy nhiên để khắc phục vấn đề Vanishing Gradient mạng ResNet có thêm kết nối "tắt" (shortcut connections) đồng nhất để xuyên qua một hay nhiều lớp. Một khối như vậy được gọi là một Residual Block, như trong hình sau:

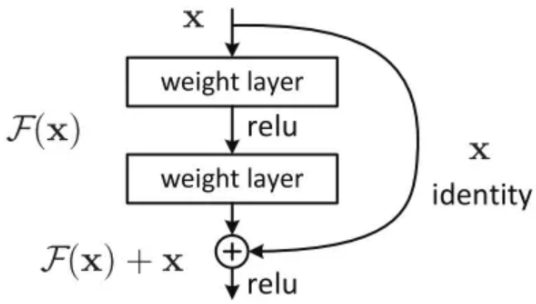

Figure 3: Kiến trúc Residual Block

– Nếu đầu vào và đầu ra có cùng kích thước thì sử dụng identity shortcuts:

$$y = \mathcal{F}(\mathrm{x}, \{W_i\}) + \mathrm{x}$$

– Nếu đầu ra kích thước tăng lên thì hoặc vẫn sử dụng identity shortcuts và thêm zero padding hoặc sử dụng projection shortcut :

$$y = \mathcal{F}(\mathrm{x}, \{W_i\}) + W_s\mathrm{x}$$

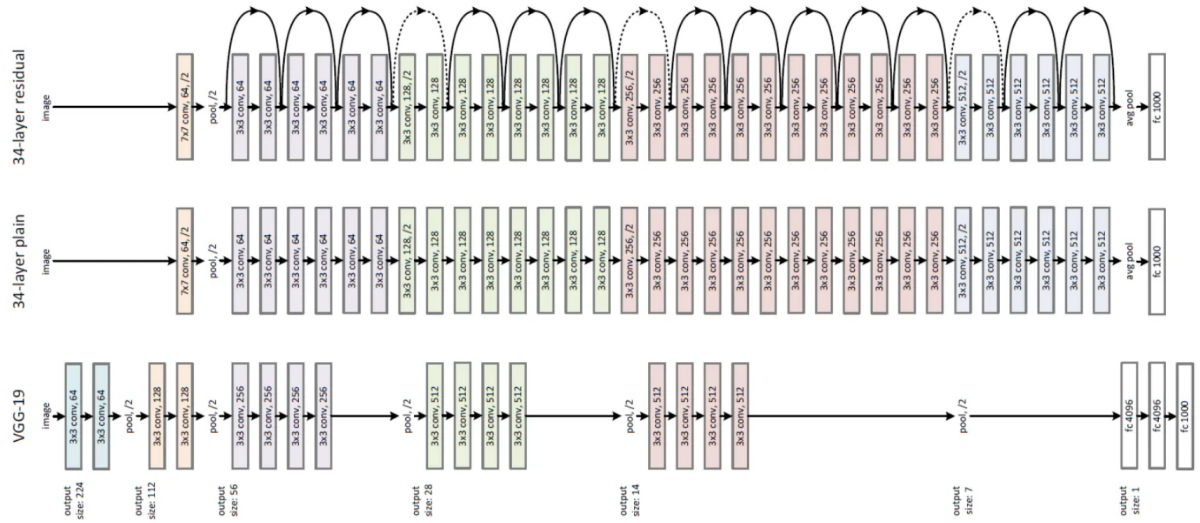

Figure 4: Kiến trúc mô hình ResNet

3.3 ConvNeXt

ConvNeXt [7] là một mô hình mạng tích chập thuần túy được tạo ra để cạnh tranh với các mô hình Transformer [10] cho thị giác máy tính. ConvNeXt được xây dựng hoàn toàn từ các mô-đun mạng tích chập tiêu chuẩn, nhưng có một số thiết kế khác biệt so với ResNet [2] truyền thống, bao gồm:

- **Thay đổi tỷ lệ tính toán:** Điều chỉnh số khối trong mỗi giai đoạn từ ResNet-50 thành một phân phối mới tương tự Swin-T [6] để cải thiện độ chính xác của mô hình.

	output size	• ResNet-50	• ConvNeXt-T
stem	56×56	7×7, 64, stride 2 3×3 max pool, stride 2	4×4, 96, stride 4
res2	56×56	[1×1, 64 3×3, 64 1×1, 256] × 3	[d7×7, 96 1×1, 384 1×1, 96] × 3
res3	28×28	[1×1, 128 3×3, 128 1×1, 512] × 4	[d7×7, 192 1×1, 768 1×1, 192] × 3
res4	14×14	[1×1, 256 3×3, 256 1×1, 1024] × 6	[d7×7, 384 1×1, 1536 1×1, 384] × 9
res5	7×7	[1×1, 512 3×3, 512 1×1, 2048] × 3	[d7×7, 768 1×1, 3072 1×1, 768] × 3
FLOPs		4.1×10^9	4.5×10^9
# params.		25.6×10^6	28.6×10^6

Figure 5: Tỷ lệ tính toán của ResNet50 và ConvNeXt

- **Thay đổi từ Stem sang "Patchify":** chuyển đổi phần xử lý ảnh đầu vào từ một chuỗi tích chập 7x7 với bước nhảy 2 kết hợp với max pooling 3x3 bước nhảy 2, sang một quá trình tích chập 4x4 với bước nhảy 4 (từ chồng lấn sang không chồng lấn).
- **Sử dụng tích chập nhóm:** áp dụng ý tưởng của ResNeXt [11] sử dụng convolution nhóm, cụ thể ConvNeXt sử dụng tích chập theo chiều sâu là một trường hợp đặc biệt của tích chập nhóm, trong đó các bộ lọc convolution được chia thành các nhóm khác nhau giúp giảm *FLOPs* [1] một cách đáng kể và mở rộng chiều rộng của mạng để bù đắp sự mất mát về dung lượng.
- **Sử dụng cổ chai đảo ngược (inverted bottleneck):** nhằm giảm số lượng kênh trước khi thực hiện tích chập không gian. Thay vì 384 -> 96 -> 384 (a) , số lượng channels trong mạng được thay đổi thành 96 -> 384 -> 96 (b).

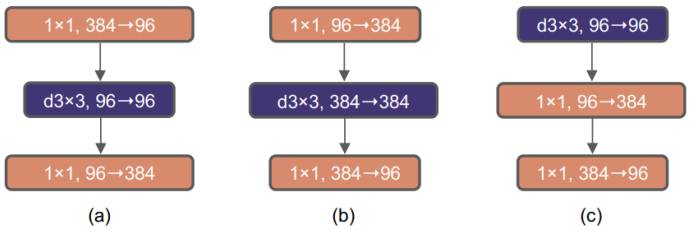

Figure 6: (a): Khối ResNeXt, (b) Khối cổ chai ngược, (c) Di chuyển lớp tích chập sâu lên trên

- **Sử dụng kích thước kernel lớn hơn:** di chuyển vị trí lớp tích chập sâu lên trên (Figure 6 (b)->(c)) và sử dụng tích chập 7x7 trong mỗi khối.

[1]Thước đo hiệu suất máy tính, hữu ích trong các lĩnh vực tính toán khoa học yêu cầu tính toán dấu phẩy động.

- **Sử dụng ít hàm kích hoạt hơn:** thay thế hàm ReLU[8] thông thường thành hàm GELU [3], loại bỏ một số lớp GELU từ khối residual, giữ lại chỉ một lớp GELU giữa hai lớp convolution 1x1.
- **Ít lớp chuẩn hóa hơn:** loại bỏ hai lớp BatchNorm [4] (BN), chỉ còn lại một lớp BN trước các lớp tích chập 1x1 và thay thế bằng Layer Normalization[1] (LN) nhằm tăng hiệu suất.
- **Sử dụng các lớp downsampling riêng biệt:** Lớp downsampling bao gồm một Layer Normalization và một tích chập 2x2 với bước nhảy 2. Chúng được thêm vào giữa các giai đoạn của mạng để giảm kích thước của feature map và tạo điều kiện cho việc học các đặc trưng ở các mức độ khác nhau trong mô hình.

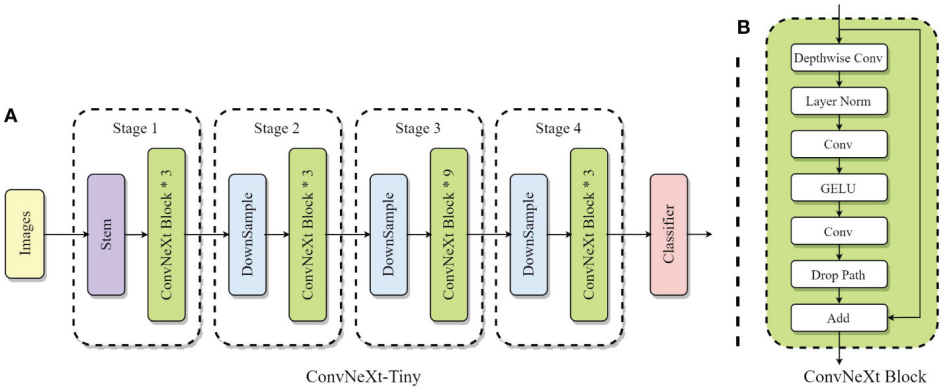

Figure 7: Kiến trúc mô hình ConvNeXt

4 Thực nghiệm

4.1 Dữ liệu

Nhóm sử dụng hai bộ dữ liệu là ImageNet và bộ dữ liệu cho Intel Image Classification (cả hai bộ dữ liệu đều được lấy trên nền tảng Kaggle).

4.1.1 ImageNet

Bộ dữ liệu gồm các bức ảnh được phân 1000 loại khác nhau:

- Train data: 1,2 triệu ảnh
- Valid data: 50000 ảnh
- Test data: 100000 ảnh

4.1.2 Intel Image Classification

Bộ dữ liệu gồm các bức ảnh phong cảnh được phân thành 6 loại {buildings, forest, glacier, mountain, sea, street}:

- Train data: 14034 ảnh
- Valid data: 3000 ảnh
- Test data: 7000 ảnh

4.2 Đánh giá

4.2.1 ImageNet

Nhóm lựa chọn sử dụng top-1 accuracy và top-5 accuracy để đánh giá và so sánh các mô hình:

Top 1 Accuracy: Dự đoán của model(label có xác suất cao nhất (kết quả hàm softmax)) phải đúng với ground truth.

Top 5 Accuracy: bất kì label nào trong 5 labels có xác suất cao nhất giống với ground truth đều được tính là dự đoán đúng.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

4.2.2 Intel Image Classification

Nhóm lựa chọn sử dụng accuracy, macro-average-precision, macro-average-recall, macro-average-f1 để đánh giá và so sánh các mô hình:

Accuracy: Đây là độ đo của bài toán phân loại mà đơn giản nhất, tính toán bằng cách lấy số dự đoán đúng chia cho toàn bộ các dự đoán.

$$\text{accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Macro-average precision: Macro-average precision là trung bình cộng của các precision theo class

$$\text{Macro-average precision} = \sum_{c=1}^{C} \frac{TP_c}{TP_c + FP_c}$$

Macro-average recall: Macro-average recall là trung bình cộng của các recall theo class

$$\text{Macro-average recall} = \sum_{c=1}^{C} \frac{TP_c}{TP_c + FN_c}$$

Macro-average F1: Macro-average recall được tính dựa trên macro-average precision và macro-average recall

$$\text{F1} = \frac{2 * precision * recall}{precision + recall}$$

4.3 Cài đặt

4.3.1 ImageNet

1. Tiền xử lý dữ liệu
   Do bộ dữ liệu test của dataset "ImageNet Object Localization Challenge" không có label nên chúng tôi sử dụng bộ validation dataset để làm làm thực nghiệm.
   - Đầu tiên do sự không thống nhất giữa categories id của các label trong các model pretrained trong pytorch với encoded categories của các label trong dataset (eg: categories id của label "godonla" trong các pretrained model là 576, encoded categories trong dataset là "n03447447"), chúng tôi xây dựng bộ mapping giữa 2 giá trị này.
   - Do bộ label của validation dataset trong bộ dự liệu có cả ground truth của 2 bài toán phân loại và định vị, chúng tôi loại bỏ phần ground truth của bài toán định vị
   - Từ đó chúng tôi có bộ validation label theo định dạng categories id theo chuẩn của các pretrained model để tiện cho việc đánh giá.
2. Cài đặt model
   Trong phần này, chúng tôi train lại từ đầu các model của thư viện pytorch với training data là bộ ImageNet 1K.

4.3.2 Intel Image Classification

Bên cạnh bộ dữ liệu gốc, nhóm còn tăng cường thêm dữ liệu bằng cách cắt ảnh gốc, xoay ảnh và lật ảnh theo chiều dọc.

- **AlexNet:** Các ảnh đều được chỉnh về kích thước $300 * 300$ và được chuẩn hóa trước khi đưa vào huấn luyện. Huấn luyện mô hình AlexNet với Adam optimizer, learning rate = 1e-4, batch size = 100, epoch = 25.
- **VGG 19:** Đầu vào là ảnh với kích thước $224 * 224$ và được chuẩn hóa. Huấn luyện mô hình VGG-19 layers với Adam optimizer, learning rate = 1e-4, batch size = 64, epoch = 30.
- **ResNet 152:** Đầu vào là ảnh với kích thước $224 * 224$ và được chuẩn hóa. Huấn luyện mô hình ResNet-152 layers với Adam optimizer, learning rate = 1e-4, batch size = 64, epoch = 30.
- **ConvNeXt:** Đầu vào là ảnh với kích thước $224 * 224$ và được chuẩn hóa. Fine-tune pretrained ConvNeXt của thư viện pytorch với Adam optimizer, learning rate = 1e-4, batch size = 64, epoch = 30.

5 Kết quả

5.1 ImageNet

Nhóm sử dụng tập valid để đánh giá: Kết quả thu được ở trong bảng 3.Từ bảng kết quả ta thấy ConvNext đạt kết quả cao nhất với top-1 và top-5 lần lượt là 80.25% và 94.81% tiếp đến là ResNet 152, VGG 19 và thấp nhất là AlexNet.

Table 3: Kết quả pretrained model cho tập dữ liệu "ImageNet Object Localization Challenge"

Model	Top-1 Accuracy	Top-5 Accuracy
AlexNet	41.81%	65.54%
VGG 19	56.27%	78.52%
ResNet 152	69.56%	87.07%
ConvNeXt	80.28%	94.81%

5.2 Intel Image Classification

Nhóm sử dụng tập valid để đánh giá: Kết quả thu được ở trong bảng 4. Đạt kết quả cao nhất là VGG 19, xếp ngay sau là mô hình ResNet 152. Còn hai mô hình AlexNet và ConvNet có kết quả gấn giống nhau.

Bên cạnh đánh giá các mô hình qua bốn độ đo nêu trên, nhóm còn sử dụng thêm confusion matrix

Table 4: Kết quả cho bộ dữ liệu "Intel Image Classification"

Model	Accuracy	Macro-Precision	Macro-Recall	Macro-F1
AlexNet	88.33%	88.57%	88.41%	88.42%
VGG 19	90.17%	90.85%	90.3%	90.35%
ResNet 152	89.23%	89.52%	89.33%	89.39%
ConvNet	88.36%	88.5%	88.54%	88.48%

để xem xét thêm kết quả đầu ra của các mô hình. Hình 8 lần lượt từ trái sang phải chính là confision matrix của 4 mô hình AlexNet, VGG 19, ResNet 152, Convnext. Dễ dàng nhận thấy rằng cả 4 mô hình đều dự đoán nhầm khá nhiều những bức tranh glacier(băng) thành mountain(núi) và ngược lại. Ngoài ra, những bức tranh về street(đường phố) và buildings(tòa nhà) cũng hay bị nhầm lẫn với nhau. Một trong những nguyên nhân cho sự dự đoán sai này là do một vài bức ảnh về đường phố có bao

gồm hình của các toà nhà hay các bức ảnh về băng lại chính là các núi tuyết. Điều này được thể hiện rõ qua một vài ví dụ ở hình 9.

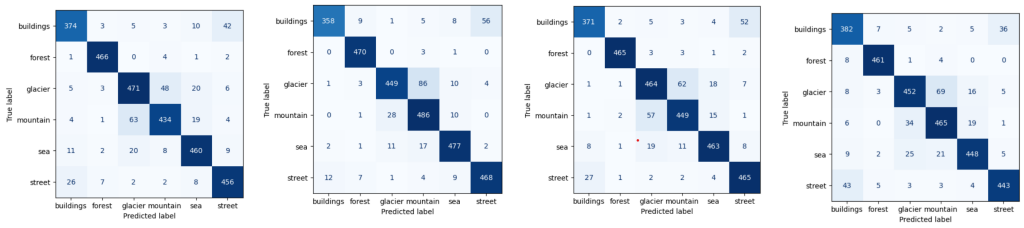

Figure 8: Kết quả Confusion matrix của AlexNet, VGG 19, ResResNet 152, Convnext

Figure 9: Một vài kết quả đầu ra của các mô hình

6 Source Code

https://github.com/lvdthieu/Computer$_V$*ision.git*

7 Tự đánh giá

- Phạm Thu Trang: Xây dựng và huấn luyện AlexNet cho bộ dữ liệu Intel Image Classification theo hướng from scratch, làm slides thuyết trình, viết báo cáo.
- Hồ Thị Thanh Bình: Xây dựng và huấn luyện VGG 19 cho bộ dữ liệu Intel Image Classification theo hướng from scratch, làm slides thuyết trình, viết báo cáo.
- Lưu Văn Đức Thiệu: Xây dựng và huấn luyện ConvNext cho bộ dữ liệu Intel Image Classification theo hướng from scratch, chạy pretrain model của AlexNet, VGG, ResNet, ConvNext cho bộ ImageNet, viết báo cáo.
- Hoàng Thái Quang: Xây dựng và huấn luyện ResNet cho bộ dữ liệu Intel Image Classification theo hướng from scratch, viết báo cáo.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer Normalization". In: *arXiv:1607.06450* (2016).

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: 2015.

[3] Dan Hendrycks and Kevin Gimpel. "Gaussian Error Linear Units (GELUs)". In: *arXiv:1606.08415* (2016).

[4] Sergey Ioffe. "Batch Renormalization: Towards Reducing Mini-Batch Dependence in Batch-Normalized Models". In: *NeurIPS*. 2017.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. "Imagenet classification with deep convolutional neural networks". In: *NeurIPS*. 2012.

[6] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows". In: *arXiv:2103.14030* (2021).

[7] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. "A ConvNet for the 2020s". In: *arXiv:2201.03545* (2020).

[8] Vinod Nair and Geoffrey E. Hinton. "Rectified linear units improve restricted Boltzmann machines". In: *Proc. 27th International Conference on Machine Learning*. 2010.

[9] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations (ICLR)*. Visual Geometry Group, Department of Engineering Science, University of Oxford. 2015.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Aidan N. Jones, Lukasz Kaiser Gomez, and I. Polosukhin. "Attention is all you need". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[11] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. "Aggregated Residual Transformations for Deep Neural Networks". In: *CVPR*. 2017.