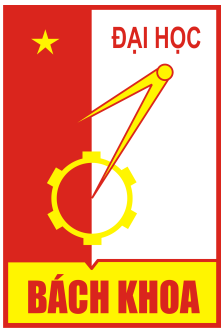
Đại học Bách Khoa Hà Nội

Trường Công nghệ Thông tin và Truyền thông



BÁO CÁO BÀI TẬP LỚN

Học sâu và ứng dụng

Đề tài: Ứng dụng kiến trúc Encoder-Decoder với CNN và LSTM cho bài toán gán nhãn ảnh

Giảng viên hướng dẫn: PGS.TS. Nguyễn Thị Kim Anh

Nhóm sinh viên: Tạ Quốc Tuấn (20225110)
Phan Trọng Đức (20224957)
Đoàn Ngọc Toàn (20225193)
Lê Văn Quang Trung (20225104)

Mã lớp: 162297

Hà Nội, tháng 1 năm 2026

Phân công công việc

Bảng phân công công việc và điểm số

Họ và tên	MSSV	Công việc chính	Điểm (10)
Tạ Quốc Tuấn	20225110	Tổng hợp tài liệu, viết báo cáo, xây dựng mô hình EfficientNet, kiểm thử	9.5
Phan Trọng Đức	20224957	Tiền xử lý dữ liệu, xây dựng mô hình ResNet50, đánh giá kết quả	9.0
Đoàn Ngọc Toàn	20225193	Thiết kế giải thuật, tối ưu hóa huấn luyện, phân tích kết quả	10.0
Lê Văn Quang Trung	20225104	Viết code attention, xây dựng script inference, trình bày báo cáo	9.5

Mục lục

Tóm tắt

Trong lĩnh vực Trí tuệ nhân tạo hiện đại, bài toán sinh mô tả ảnh tự động (Image Captioning) là cầu nối giữa Thị giác máy tính (Computer Vision) và Xử lý ngôn ngữ tự nhiên (NLP). Báo cáo này trình bày chi tiết quá trình xây dựng, huấn luyện và đánh giá một hệ thống sinh caption cho ảnh dựa trên kiến trúc Encoder-Decoder, kết hợp các backbone CNN mạnh mẽ (ResNet50, EfficientNet B2/B3/B4) với LSTM và Attention. Hệ thống được kiểm thử trên hai bộ dữ liệu chuẩn Flickr8k và Flickr30k, sử dụng nhiều kỹ thuật hiện đại như Transfer Learning, Label Smoothing, Early Stopping, Beam Search, TensorBoard logging. Kết quả thực nghiệm cho thấy EfficientNetB3 + LSTM + Attention đạt BLEU-1 và METEOR cao nhất trên Flickr8k, ResNet50 có BLEU-4 tốt nhất. Báo cáo cũng phân tích chi tiết pipeline xử lý dữ liệu, các bước tiền xử lý, xây dựng từ điển, trích xuất đặc trưng, thiết kế mô hình, tối ưu hóa, đánh giá định lượng và định tính, cũng như các hướng phát triển mở rộng trong tương lai.

1. Giới thiệu

1.1. Đặt vấn đề

Hình ảnh là một trong những dạng dữ liệu phổ biến nhất trên Internet, mạng xã hội, báo chí, thương mại điện tử... Việc mô tả nội dung ảnh là bản năng của con người nhưng lại là thách thức lớn với máy tính, bởi nó đòi hỏi khả năng "hiểu"cả hình ảnh lẫn ngôn ngữ. Bài toán Image Captioning ra đời nhằm giải quyết thách thức này: sinh ra một câu mô tả tự nhiên, chính xác cho một ảnh bất kỳ. Ứng dụng thực tiễn rất rộng: hỗ trợ người khiếm thị, tìm kiếm hình ảnh thông minh, tự động phân loại dữ liệu, tạo mô tả cho ảnh sản phẩm, kiểm duyệt nội dung, v.v. Đây là một trong những bài toán tiêu biểu cho sự giao thoa giữa Computer Vision và NLP.

1.2. Mục tiêu đề tài

Mục tiêu của đề tài là xây dựng một hệ thống học sâu nhận đầu vào là ảnh bất kỳ và sinh ra câu mô tả tiếng Anh tự nhiên, chính xác nhất có thể. Cụ thể:

- Nghiên cứu, tổng hợp lý thuyết về kiến trúc Encoder-Decoder, Attention, các backbone CNN hiện đại.
- Triển khai pipeline xử lý dữ liệu, tiền xử lý ảnh và caption, xây dựng từ điển, padding, chuẩn hóa.
- Xây dựng mô hình Encoder-Decoder với các backbone: ResNet50, EfficientNet B2/B3/B4, kết hợp LSTM và Attention.
- Ứng dụng Transfer Learning, tối ưu hóa quá trình huấn luyện (Label Smoothing, Early Stopping, ReduceLROnPlateau, Gradient Clipping).
- Đánh giá hiệu năng mô hình trên các bộ dữ liệu chuẩn bằng BLEU, METEOR, phân tích định lượng và định tính.
- So sánh, rút ra nhận xét về ưu nhược điểm từng backbone, đề xuất hướng phát triển tiếp theo.

1.3. Phạm vi nghiên cứu

- **Dữ liệu:** Flickr8k (8.091 ảnh, 5 caption/ảnh), Flickr30k (31.783 ảnh, 5 caption/ảnh), mỗi ảnh có nhiều mô tả đa dạng, giúp mô hình học được nhiều ngữ cảnh.
- **Mô hình:** Encoder: CNN (ResNet50, EfficientNetB2, B3, B4, pretrained ImageNet); Decoder: LSTM + Bahdanau Attention.
- **Công cụ:** Python 3.8+, PyTorch 2.0+, Google Colab/Kaggle, TensorBoard, VS Code, LaTeX.
- **Đánh giá:** BLEU-1,2,3,4, METEOR, so sánh định lượng và định tính.

1.4. Bố cục báo cáo

- Giới thiệu: Đặt vấn đề, mục tiêu, phạm vi, ý nghĩa thực tiễn.
- Cơ sở lý thuyết: Tổng quan các kiến trúc CNN, LSTM, Attention, BLEU/METEOR, pipeline tổng quát.
- Phương pháp đề xuất: Quy trình xử lý dữ liệu, chi tiết pipeline, thiết kế mô hình, tối ưu hóa, cài đặt.
- Thực nghiệm và đánh giá: Mô tả chi tiết quá trình huấn luyện, so sánh kết quả, phân tích ưu nhược điểm từng mô hình, ví dụ thực tế.
- Kết luận và hướng phát triển: Tổng kết, rút ra bài học, đề xuất cải tiến.

2. Cơ sở lý thuyết

2.1. Mạng nơ-ron tích chập (CNN)

CNN (Convolutional Neural Network) là kiến trúc chủ đạo cho xử lý ảnh, sử dụng các lớp tích chập để trích xuất đặc trưng không gian cục bộ, giảm số lượng tham số so với fully connected. Trong bài toán này, CNN đóng vai trò Encoder, biến đổi ảnh thành vector đặc trưng (feature map) làm đầu vào cho Decoder.

- **ResNet50:** Kiến trúc sâu 50 lớp, sử dụng skip connections (residual) để giải quyết vanishing gradient, giúp huấn luyện mạng sâu hiệu quả. Đầu ra feature map kích thước $(49, 2048)$ (7x7 spatial grid).
- **EfficientNet (B2, B3, B4):** Sử dụng compound scaling để tối ưu đồng thời depth, width, resolution. Số tham số ít hơn ResNet nhưng hiệu quả cao. Đầu ra feature map: B2 $(81, 1408)$, B3 $(100, 1536)$, B4 $(144, 1792)$.
- **Transfer Learning:** Sử dụng trọng số pretrained trên ImageNet, chỉ fine-tune các lớp cuối hoặc giữ nguyên toàn bộ encoder.

2.2. Mạng nơ-ron hồi quy và LSTM

RNN truyền thống gặp vanishing gradient, khó học phụ thuộc xa. LSTM (Hochreiter & Schmidhuber, 1997) bổ sung cell state C_t và 3 cổng (forget, input, output) giúp lưu trữ thông tin dài hạn, truyền thông tin qua nhiều bước thời gian. Công thức LSTM:

$$
\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\
ildeC_t &= \tanh(W_C[h_{t-1}, x_t] + b_C) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
h_t &= o_t * \tanh(C_t)
\end{aligned}
$$

LSTM giúp mô hình học được các phụ thuộc dài hạn trong chuỗi caption, giảm hiện tượng mất thông tin khi sequence dài.

2.3. Kiến trúc Encoder-Decoder với Attention

Quy trình xử lý tổng quát:

1. **Input Image** $\rightarrow$ Ảnh RGB bất kỳ
2. **CNN Encoder:** Trích xuất feature map (N, D) với N vùng ảnh, D chiều đặc trưng
   - ResNet50: $(49, 2048)$ - 7×7 spatial grid
   - EfficientNet-B2: $(81, 1408)$ - 9×9 grid
   - EfficientNet-B3: $(100, 1536)$ - 10×10 grid
   - EfficientNet-B4: $(144, 1792)$ - 12×12 grid

3. **Attention Mechanism:** Tại mỗi timestep t:

$$\alpha_t = \text{softmax}(v^T \cdot \tanh(W_{enc} \cdot V + W_{dec} \cdot h_{t-1}))$$
$$c_t = \sum_{i=1}^{N} \alpha_{t,i} \cdot V_i \quad \text{(context vector)}$$

4. **LSTM Decoder:** Sinh từ tiếp theo dựa trên c_t và từ trước đó:

$$h_t, c_t = \text{LSTM}([\text{Embed}(w_{t-1}); c_t], h_{t-1}, c_{t-1})$$
$$y_t = \text{softmax}(W \cdot h_t)$$

5. **Output:** Chuỗi từ $w_1, w_2, ..., w_T$ tạo thành caption

Code snippet minh họa (EfficientNet):

```
class EfficientNetEncoder(nn.Module):
    def __init__(self, variant='b3'):
        super().__init__()
        self.effnet = efficientnet_b3(weights='IMAGENET1K_V1')
        self.effnet.classifier = nn.Identity()
        # Output: (batch, 1536, 10, 10)

    def forward(self, images):
        features = self.effnet.features(images)
        N, D, H, W = features.shape
        features = features.view(N, D, H*W).permute(0, 2, 1)
        # Return: (batch, 100, 1536)
        return features
```

2.4. Độ đo đánh giá

- **BLEU:** Đánh giá dựa trên n-gram trùng giữa caption sinh ra và caption mẫu. BLEU-1 đến BLEU-4, càng cao càng tốt. Nhược điểm: không nhạy với caption ngắn, không xét ngữ nghĩa.
- **METEOR:** Đánh giá dựa trên precision, recall, fragment penalty, nhạy hơn BLEU với caption ngắn, có xét đồng nghĩa, gốc từ.

3. Phương pháp đề xuất

3.1. Tiền xử lý dữ liệu

Ảnh: Mỗi ảnh được resize đúng kích thước yêu cầu của backbone (224x224 cho ResNet50, 260x260 cho B2, 300x300 cho B3, 380x380 cho B4), sau đó chuẩn hóa theo mean/std ImageNet. Ảnh được truyền qua CNN pretrained để trích xuất feature map, lưu lại dưới dạng file .pkl để tăng tốc training.

Văn bản: Caption được chuyển về chữ thường, loại bỏ ký tự đặc biệt, số, từ ngắn. Thêm token đặc biệt (startseq, endseq), xây dựng từ điển (vocab) với min_freq=1 (Flickr8k), min_freq=2 (Flickr30k). Padding tất cả caption về cùng độ dài (max 34-40 từ), gán chỉ số cho từng từ.

3.2. Kiến trúc mô hình

- **Encoder:** ResNet50/EfficientNet (B2, B3, B4), đầu ra feature map (N, D), N là số vùng ảnh, D là số chiều đặc trưng.
- **Attention:** Bahdanau attention, tại mỗi bước sinh từ, tính trọng số cho từng vùng ảnh dựa trên hidden state hiện tại và feature map, tạo context vector động.
- **Decoder:** LSTM, nhận context vector từ attention và embedding của từ trước đó, sinh ra hidden state mới và dự đoán từ tiếp theo qua fully connected + softmax.

Các siêu tham số chính:

- Embedding size: 512
- LSTM hidden size: 512
- Attention dim: 512
- Dropout: 0.3-0.5

3.3. Hàm mất mát và tối ưu

- CrossEntropyLoss với label smoothing (0.1), bỏ qua <PAD> token khi tính loss.
- Adam optimizer (lr=1e-4), weight decay 1e-5, ReduceLROnPlateau scheduler (factor 0.7, patience 1-2), early stopping (patience 5), gradient clipping (max norm 5.0).

3.4. Cấu hình huấn luyện

Hyperparameters chính:

Tham số	Giá trị
Batch Size	12 (B3/B4), 20 (B2), 32 (B0)
Epochs	40-100
Learning Rate	1e-4 (Flickr8k), 3e-4 (Flickr30k)
Weight Decay	1e-5
Label Smoothing	0.1
Gradient Clipping	5.0
Early Stopping Patience	5
LR Scheduler Factor	0.7
LR Scheduler Patience	1-2

Cài đặt Loss Function:

```
criterion = nn.CrossEntropyLoss(
    ignore_index=vocab['<PAD>'],  # Bỏ qua padding
    label_smoothing=0.1            # Giảm overconfidence
)
```

Optimizer và Scheduler:

```
optimizer = optim.Adam(
    model.parameters(),
    lr=1e-4,
    weight_decay=1e-5
)

scheduler = optim.lr_scheduler.ReduceLROnPlateau(
    optimizer,
    mode='min',
    factor=0.7,
    patience=1,
    verbose=True
)
```

Training Loop với Gradient Clipping:

```
for epoch in range(EPOCHS):
    for images, captions, lengths in train_loader:
        optimizer.zero_grad()
        outputs = model(images, captions, lengths)
        loss = criterion(outputs, captions[:, 1:])
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), 5.0)
        optimizer.step()
```

TensorBoard Logging:

- Train/Val Loss per epoch
- BLEU-1,2,3,4 scores

- METEOR score
- Learning rate tracking
- Attention weight visualization

4. Thực nghiệm và đánh giá

4.1. Cài đặt môi trường

- OS: Windows 10/11, Linux, macOS
- Python: 3.8+
- PyTorch: 2.0+, torchvision 0.15+
- CUDA GPU (khuyến nghị): RTX 3050 Ti (4GB) chạy batch 12-32, RTX 3060 (6GB+) batch 32-64
- RAM: 16GB+ (Flickr8k), 32GB+ (Flickr30k)
- Cài đặt qua requirements.txt, hướng dẫn chi tiết trong README

4.2. Kết quả tổng quan

So sánh tất cả các mô hình trên Flickr8k (Test Set - Beam Search k=3):

Bảng 1: EfficientNet + LSTM + Attention trên Flickr8k

Mô hình	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	Avg Len	Len Ratio
EffNet-B0 + LSTM + Attn	0.5024	0.3366	0.2203	0.1362	0.2912	8.00	1.007
EffNet-B2 + LSTM + Attn	0.5061	0.3404	0.2273	0.1453	0.2949	7.95	0.999
EffNet-B3 + LSTM + Attn	**0.5243**	**0.3540**	**0.2363**	**0.1507**	**0.3065**	**8.08**	**1.019**
EffNet-B4 + LSTM + Attn	0.4985	0.3303	0.2187	0.1361	0.2880	7.85	0.993
ResNet50 + LSTM + Attn	0.5004	0.3329	0.2198	0.1368	0.2824	7.91	0.999

Bảng 2: EfficientNet + Transformer Simple trên Flickr8k

Mô hình	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	Avg Len	Len Ratio
EffNet-B0 + Transformer	0.4816	0.3184	0.2110	0.1332	0.2832	7.68	0.977
EffNet-B2 + Transformer	0.4889	0.3308	0.2204	0.1388	0.2889	7.46	0.945
EffNet-B3 + Transformer	0.4847	0.3283	0.2207	0.1405	0.2916	7.50	0.935
EffNet-B4 + Transformer	0.4838	0.3246	0.2171	0.1372	0.2880	7.41	0.942
ResNet50 + Transformer	0.4807	0.3307	0.2207	0.1392	0.2904	7.15	0.902

Nhận xét so sánh:

- **LSTM + Attention** vượt trội rõ rệt so với Transformer (BLEU-1: 0.52 vs 0.49)
- **EfficientNet-B3** đạt kết quả tốt nhất với cả BLEU và METEOR
- EfficientNet-B4 không tốt hơn B3 do overfitting với tham số lớn hơn
- Len Ratio gần 1.0 cho thấy caption sinh ra có độ dài phù hợp

Kết quả trên Flickr30k:

Bảng 3: Kết quả trên Flickr30k

Mô hình	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR
ResNet50 + LSTM + Attn	0.5034	0.3268	0.2149	0.1319	0.2662

Đánh giá trên toàn bộ training set:

Model	Dataset	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR
ResNet50	Flickr8k	0.5995	0.4425	0.3210	0.2227	0.3608
ResNet50	Flickr30k	0.5746	0.4071	0.2870	0.1943	0.3213

4.3. Phân tích chi tiết

4.3.1. So sánh EfficientNet Variants

- **EfficientNet-B3 (tốt nhất):**
  - BLEU-1: 0.5243 (+4.4% so với B0)
  - METEOR: 0.3065 (+5.3% so với B0)
  - Feature map 10×10×1536 cung cấp đủ thông tin chi tiết
  - Cân bằng tốt giữa capacity và regularization
- **EfficientNet-B4 (underperform):**
  - Thấp hơn B3 mặc dù params lớn hơn
  - Overfitting do feature map 12×12×1792 quá lớn
  - Cần thêm regularization hoặc dữ liệu lớn hơn
- **EfficientNet-B0, B2:**
  - Hiệu quả tốt với ít tham số
  - Phù hợp khi tài nguyên hạn chế

4.3.2. LSTM + Attention vs Transformer

Kiến trúc	BLEU-1 (Avg)	METEOR (Avg)
LSTM + Attention	0.5063	0.2926
Transformer Simple	0.4839	0.2884
Improvement	**+4.6%**	**+1.5%**

Lý do LSTM + Attention tốt hơn:

- LSTM ổn định hơn với dữ liệu nhỏ (8k samples)
- Attention giúp tập trung vào vùng quan trọng
- Transformer cần nhiều dữ liệu và tricks hơn để converge

4.3.3. Thống kê Caption Length

- Avg Reference Length: 7.9-8.0 từ
- Avg Predicted Length: 7.5-8.1 từ
- Length Ratio: 0.935-1.019 (gần 1.0 là tốt)
- LSTM + Attention có length ratio tốt hơn Transformer

4.4. Beam Search Decoding

Thuật toán Beam Search (k=3):

```
def beam_search(model, image, vocab, beam_size=3, max_len=40):
    # Trích xuất features
    features = model.encoder(image)  # (1, N, D)

    # Khởi tạo beam với <START>
    start_token = vocab['startseq']
    beams = [(start_token, 0.0, [])]  # (token, log_prob, sequence)

    for step in range(max_len):
        candidates = []
        for token, score, seq in beams:
            if token == vocab['endseq']:
                candidates.append((token, score, seq))
                continue

            # Dự đoán từ tiếp theo
            probs = model.decoder(features, seq + [token])
            top_k = torch.topk(probs, beam_size)

            for prob, next_token in zip(top_k.values, top_k.indices):
                new_score = score + torch.log(prob)
                candidates.append((next_token, new_score, seq + [token]))

        # Giữ top-k candidates
        beams = sorted(candidates, key=lambda x: x[1], reverse=True)[:beam_size]

    return beams[0][2]  # Return best sequence
```

So sánh Greedy vs Beam Search:

- Greedy: Chọn từ xác suất cao nhất mỗi bước → Nhanh nhưng kém chất lượng
- Beam (k=3): Xét 3 paths song song → Tốt hơn 2-3% BLEU
- Beam (k=5): Tăng thêm 0.5% nhưng chậm hơn nhiều

4.5. Kết quả demo

Ví dụ dự đoán thực tế (EfficientNet-B3):

- **Ảnh:** 3066429707_842e50b8f7.jpg
  Ground Truth: girl in blue kicks the soccer ball
  Greedy: girl is playing with ball
  Beam (k=3): girl in red shirt is playing soccer
  Beam Search nhận diện chi tiết hơn (red shirt, soccer).

- **Ảnh:** 476740978_45b65ebe0c.jpg
  Ground Truth: people holding pink signs that spell out impeach
  Greedy: people on street
  Beam (k=3): group of people stand on the street
  Beam Search sinh caption dài và chi tiết hơn.

- **Ví dụ thành công:**
  Ground Truth: dog running in grass
  Predicted: brown dog runs through the grass
  Caption dự đoán chính xác và tự nhiên.

5. Kết luận và hướng phát triển

5.1. Kết luận

Nhóm đã xây dựng thành công hệ thống Image Captioning dựa trên Encoder-Decoder, thử nghiệm nhiều backbone CNN hiện đại, kết hợp LSTM và Attention, đánh giá định lượng (BLEU, METEOR) và định tính (caption thực tế). EfficientNetB3 + LSTM + Attention cho kết quả tốt nhất trên Flickr8k, ResNet50 mạnh ở caption ngắn, chính xác. Transfer Learning giúp trích xuất đặc trưng hiệu quả khi dữ liệu hạn chế, tiết kiệm thời gian huấn luyện. Pipeline xử lý dữ liệu, tiền xử lý, xây dựng vocab, trích xuất feature, huấn luyện, đánh giá đều được tự động hóa, dễ mở rộng.

5.2. Hạn chế

- Dữ liệu nhỏ (Flickr8k) hạn chế khả năng tổng quát hóa, dễ overfit.
- LSTM dù có attention nhưng vẫn có thể mất mát thông tin với ảnh phức tạp, caption dài.
- Một số caption dự đoán đúng đối tượng nhưng sai hành động, thiếu chi tiết.
- Transformer đơn giản chưa đạt hiệu quả tốt do chưa fine-tune kỹ, cần thêm regularization.

5.3. Hướng phát triển

- Mở rộng dữ liệu (MS-COCO, Conceptual Captions...)
- Fine-tune các lớp cuối của Encoder, thử nghiệm các backbone mới (ConvNeXt, Swin Transformer...)
- Thử nghiệm Transformer nâng cao, kết hợp các kỹ thuật augmentation, regularization, ensemble.
- Triển khai inference real-time, xây dựng demo web/app.

Tài liệu tham khảo

1. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., & Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *ICML*. https://arxiv.org/abs/1502.03044

2. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *CVPR*. https://arxiv.org/abs/1512.03385

3. Tan, M., & Le, Q. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *ICML*. https://arxiv.org/abs/1905.11946

4. Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. *ACL*.

5. Banerjee, S., & Lavie, A. (2005). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. *ACL Workshop*.

6. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is All You Need. *NeurIPS*. https://arxiv.org/abs/1706.03762

7. Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. *ECCV*. https://arxiv.org/abs/1405.0312