ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN TỬ - VIỄN THÔNG

BÁO CÁO CUỐI KÌ

HỌC PHẦN:

DEEP LEARNING

Sinh viên thực hiện : Nguyễn Bá Thành
Lớp : 21DTCLC4
Nhóm : 21.41
Giảng viên : TS.Trần Thị Minh Hạnh

Tháng 12 năm 2025

LỜI NÓI ĐẦU VÀ CẢM ƠN

Trong dòng chảy thời đại, Trí tuệ nhân tạo (Artificial Intelligence - AI) và đặc biệt là các kỹ thuật **Học sâu (Deep Learning)** đang trở thành một trong những hướng phát triển mũi nhọn, mở ra tiềm năng ứng dụng to lớn và mang tính đột phá. Không chỉ dừng lại ở các bài toán lý thuyết, Học sâu đang được triển khai mạnh mẽ trong thị giác máy tính, xử lý ngôn ngữ tự nhiên, y tế thông minh, xe tự hành và các hệ thống nhúng AIoT (AI of Things).

Tại Việt Nam, nhiều tập đoàn công nghệ lớn như Viettel, FPT, VNPT và VinAI đang đầu tư nguồn lực khổng lồ vào nghiên cứu và ứng dụng các mô hình Học sâu để giải quyết các bài toán thực tiễn. Đồng hành cùng định hướng quốc gia đó, Trường Đại học Bách khoa – Đại học Đà Nẵng, với bề dày truyền thống đào tạo và nghiên cứu kỹ thuật, đang đóng vai trò nòng cốt trong việc đào tạo nguồn nhân lực chất lượng cao, làm chủ công nghệ lõi trong kỷ nguyên số.

Trước hết, em xin bày tỏ lòng biết ơn sâu sắc đến cô T.S Trần Thị Minh Hạnh, người đã tận tình hướng dẫn và đồng hành cùng em trong suốt quá trình nghiên cứu và hoàn thiện đề tài. Những kiến thức chuyên môn, sự hỗ trợ quý báu và tinh thần khích lệ từ cô là nguồn động lực lớn để em vượt qua những khó khăn trong quá trình thực hiện.

Em xin gửi lời cảm ơn chân thành đến khoa Điện tử - Viễn thông và ban lãnh đạo Trường Đại học Bách khoa – Đại học Đà Nẵng đã tạo điều kiện thuận lợi cả về cơ sở vật chất lẫn môi trường học thuật để em có cơ hội tiếp cận và phát triển chuyên môn trong lĩnh vực công nghệ tiên tiến.

LỜI CAM ĐOAN LIÊM CHÍNH HỌC THUẬT

Em xin cam đoan rằng toàn bộ nội dung trong báo cáo/luận văn/đề tài này là kết quả của quá trình học tập, nghiên cứu và thực hiện của riêng em dưới sự hướng dẫn của giảng viên hướng dẫn.

Em hoàn toàn chịu trách nhiệm về tính trung thực và liêm chính học thuật của báo cáo này. Các số liệu, kết quả, hình ảnh và trích dẫn trong báo cáo đều được trình bày một cách trung thực, có nguồn gốc rõ ràng và tuân thủ đúng các quy định về đạo đức nghiên cứu và trích dẫn tài liệu.

Đà Nẵng, ngày 7 tháng 12 năm 2025

Người thực hiện

Nguyễn Bá Thành

MỤC LỤC

DANH MỤC BẢNG

DANH MỤC HÌNH

1 Bài toán phân vùng vết nứt

Đề bài: Thư mục data_01 gồm các thư mục "rgb", "BW", "rgb_test", "BW_test" chứa hình ảnh vết nứt và mặt nạ vết nứt (mask) (với các điểm ảnh trắng biểu diễn cho vết nứt). Sinh viên thực hiện và báo cáo các nội dung sau:

1.1 Xây dựng mô hình U-Net

Xây dựng một mô hình học sâu (không dùng mô hình YOLO) để phân vùng vết nứt. Sử dụng một bảng để liệt kê và mô tả thông tin của tất cả các lớp trong mô hình, đồng thời giải thích lý do chọn kiến trúc mô hình đó.

Bảng 1: Chi tiết kiến trúc mô hình U-Net

Tên khối	**Thành phần chi tiết**	**Output Shape**
Input	Image Input ($C_{in} = 3$)	$3 \times H \times W$
Encoder 1	DoubleConv (64 filters)	$64 \times H \times W$
Pool 1	MaxPool2d (2×2)	$64 \times H/2 \times W/2$
Encoder 2	DoubleConv (128 filters)	$128 \times H/2 \times W/2$
Pool 2	MaxPool2d (2×2)	$128 \times H/4 \times W/4$
Encoder 3	DoubleConv (256 filters)	$256 \times H/4 \times W/4$
Pool 3	MaxPool2d (2×2)	$256 \times H/8 \times W/8$
Encoder 4	DoubleConv (512 filters)	$512 \times H/8 \times W/8$
Pool 4	MaxPool2d (2×2)	$512 \times H/16 \times W/16$
Bottleneck	DoubleConv (1024 filters)	$1024 \times H/16 \times W/16$
UpConv 4	ConvTranspose2d ($1024 \to 512$)	$512 \times H/8 \times W/8$
Concat 4	Concatenation (w/ Encoder 4)	$1024 \times H/8 \times W/8$
Decoder 4	DoubleConv ($1024 \to 512$)	$512 \times H/8 \times W/8$
UpConv 3	ConvTranspose2d ($512 \to 256$)	$256 \times H/4 \times W/4$
Concat 3	Concatenation (w/ Encoder 3)	$512 \times H/4 \times W/4$
Decoder 3	DoubleConv ($512 \to 256$)	$256 \times H/4 \times W/4$
UpConv 2	ConvTranspose2d ($256 \to 128$)	$128 \times H/2 \times W/2$
Concat 2	Concatenation (w/ Encoder 2)	$256 \times H/2 \times W/2$
Decoder 2	DoubleConv ($256 \to 128$)	$128 \times H/2 \times W/2$
UpConv 1	ConvTranspose2d ($128 \to 64$)	$64 \times H \times W$
Concat 1	Concatenation (w/ Encoder 1)	$128 \times H \times W$
Decoder 1	DoubleConv ($128 \to 64$)	$64 \times H \times W$
Output	Conv2d (1×1)	$1 \times H \times W$

Lý do lựa chọn kiến trúc U-Net

- **Phù hợp hình thái vết nứt:** Vết nứt thường mảnh và uốn lượn. U-Net phân loại từng điểm ảnh (pixel-level), giúp định dạng chính xác vùng hư hỏng.
- **Bảo toàn chi tiết (Skip Connections):** Cơ chế kết nối tắt giúp chuyển giao thông tin không gian chi tiết từ lớp mã hóa (Encoder) sang lớp giải mã (Decoder), khắc phục tình trạng mất mát thông tin của các đường nứt nhỏ do quá trình giảm chiều (Pooling).

1.2 Hàm mất mát và phương pháp tối ưu hóa

Chọn và mô tả hàm mất mát (loss function) và phương pháp tối ưu hóa sinh viên sử dụng. Giải thích lý do lựa chọn.

Hàm mất mát (Loss Function)

Nghiên cứu sử dụng hàm mất mát kết hợp (**Combined Loss**) giữa *Binary Cross Entropy (BCE) With Logits* và *Dice Loss* theo công thức:

$$L_{total} = \lambda_1 \cdot L_{BCE} + \lambda_2 \cdot L_{Dice} \tag{1}$$

Trong đó $\lambda_1 = 0.3$ và $\lambda_2 = 0.7$.

Lý do lựa chọn:

- **Khắc phục mất cân bằng dữ liệu (Class Imbalance):** Đây là vấn đề lớn nhất trong phân vùng vết nứt (diện tích vết nứt thường $< 5\%$ ảnh).
  - *Weighted BCE:* Sử dụng trọng số dương $pos\_weight = 5.0$ để tăng hình phạt khi mô hình bỏ sót các pixel vết nứt.
  - *Dice Loss:* Tập trung tối ưu hóa chỉ số giao thoa (IoU) giữa dự đoán và nhãn thực tế, giúp mô hình học tốt hình dạng liền mạch của vết nứt thay vì chỉ phân loại từng điểm ảnh rời rạc.
- **Ổn định quá trình huấn luyện:** L_{BCE} giúp làm mượt gradient giúp mô hình hội tụ ổn định, trong khi L_{Dice} giúp tinh chỉnh độ chính xác của đường biên vết nứt.

Phương pháp tối ưu hóa (Optimizer)

Sử dụng thuật toán **Adam** (Adaptive Moment Estimation) với tốc độ học (learning rate) $\eta = 10^{-4}$.

Lý do lựa chọn:

- **Thích ứng tốt:** Adam tự động điều chỉnh tốc độ học cho từng tham số, giúp mô hình hội tụ nhanh hơn so với SGD truyền thống, đặc biệt hiệu quả với dữ liệu nhiễu và gradient thưa (sparse gradients) trong bài toán phân vùng ảnh.
- **Hiệu quả tính toán:** Phù hợp với kiến trúc mạng sâu như U-Net, giảm thiểu nguy cơ mắc kẹt tại các điểm tối ưu cục bộ.

1.3 Kết quả huấn luyện mô hình

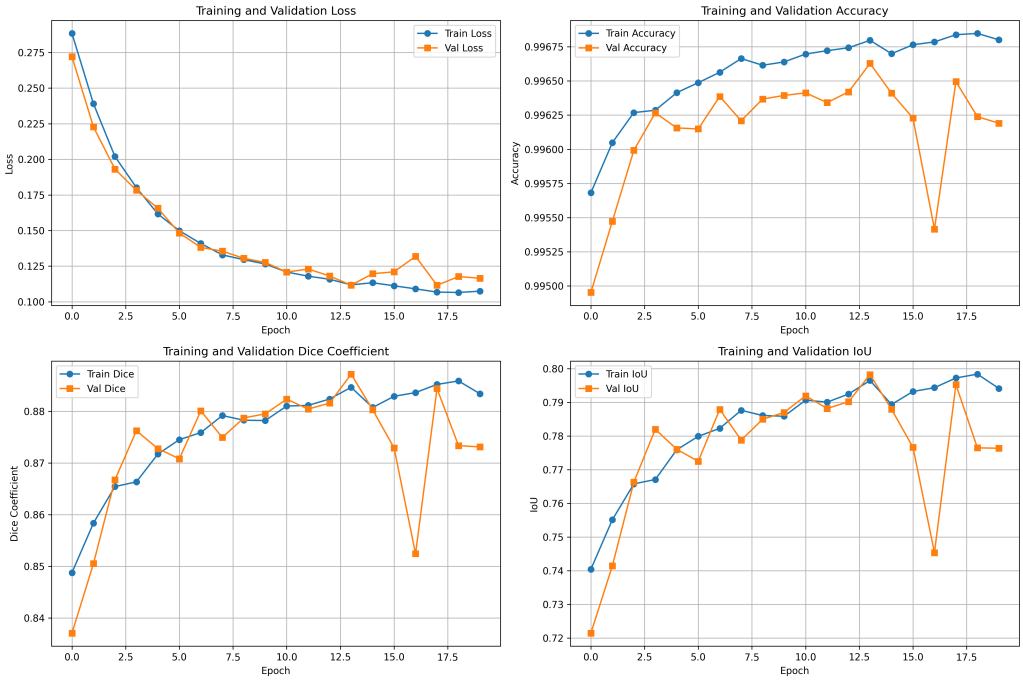
Huấn luyện mô hình sử dụng hai thư mục "rgb", "BW" là dữ liệu huấn luyện; 2 thư mục "rgb_test" và "BW_test" là thư mục kiểm tra. Báo cáo độ chính xác phân vùng và giá trị hàm mất mát (loss) của mô hình trên tập huấn luyện và kiểm tra trong ít nhất 20 epochs.

Bảng 2: Tóm tắt quá trình huấn luyện mô hình U-Net

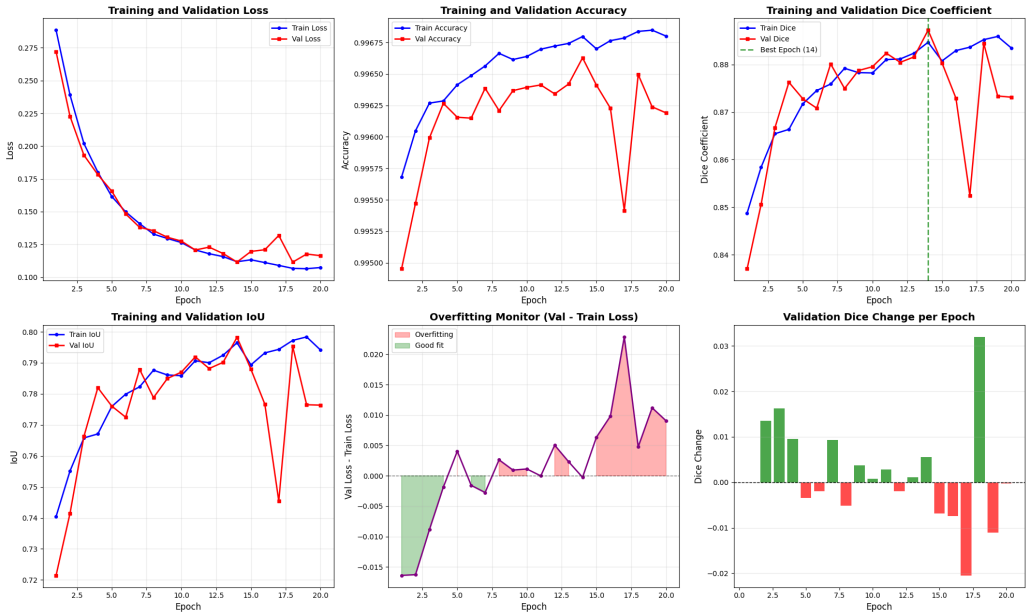
Epoch	Train Loss	Val Loss	Train Dice	Val Dice	Ghi chú
1	0.2884	0.2720	0.8488	0.8371	Khởi tạo
5	0.1616	0.1656	0.8717	0.8728	Hội tụ nhanh
10	0.1265	0.1276	0.8782	0.8795	Ổn định
14	**0.1118**	**0.1116**	**0.8847**	**0.8872**	**Mô hình tốt nhất**
17	0.1090	0.1319	0.8836	0.8524	Dao động nhẹ
20	0.1074	0.1164	0.8834	0.8731	Kết thúc

Nhận xét quá trình huấn luyện:

- **Sự hội tụ:** Hàm mất mát giảm đều từ 0.2884 xuống 0.1074, cho thấy mô hình học tốt các đặc trưng của vết nứt.
- **Hiện tượng Overfitting:** Khoảng cách giữa *Train Dice* (0.8834) và *Val Dice* (0.8731) ở epoch cuối rất nhỏ (~ 0.01), chứng tỏ mô hình có khả năng tổng quát hóa tốt, không bị quá khớp.



Hình 1: Biểu đồ biến thiên của hàm Loss, chỉ số Dice trên tập Train và Validation



Hình 2: Biểu đồ lịch sử huấn luyện mô hình U-Net

Bảng 3: Bảng số liệu chi tiết quá trình huấn luyện qua 20 epochs

Epoch	**Loss**		**Accuracy**		**Dice Score**		**IoU**	
	Train	*Val*	*Train*	*Val*	*Train*	*Val*	*Train*	*Val*
1	0.2884	0.2720	0.9957	0.9950	0.8488	0.8371	0.7404	0.7214
2	0.2391	0.2228	0.9960	0.9955	0.8584	0.8505	0.7551	0.7414
3	0.2020	0.1931	0.9963	0.9960	0.8654	0.8667	0.7658	0.7663
4	0.1801	0.1783	0.9963	0.9963	0.8664	0.8763	0.7671	0.7820
5	0.1616	0.1656	0.9964	0.9962	0.8717	0.8728	0.7759	0.7760
6	0.1499	0.1483	0.9965	0.9961	0.8745	0.8708	0.7799	0.7725
7	0.1409	0.1381	0.9966	0.9964	0.8759	0.8801	0.7823	0.7879
8	0.1329	0.1355	0.9967	0.9962	0.8792	0.8750	0.7876	0.7788
9	0.1295	0.1305	0.9966	0.9964	0.8783	0.8787	0.7861	0.7850
10	0.1265	0.1276	0.9966	0.9964	0.8782	0.8795	0.7859	0.7870
11	0.1208	0.1208	0.9967	0.9964	0.8810	0.8824	0.7906	0.7919
12	0.1179	0.1230	0.9967	0.9963	0.8812	0.8805	0.7900	0.7882
13	0.1158	0.1181	0.9967	0.9964	0.8824	0.8816	0.7924	0.7902
14	**0.1118**	**0.1116**	**0.9968**	**0.9966**	**0.8847**	**0.8872**	**0.7965**	**0.7982**
15	0.1133	0.1197	0.9967	0.9964	0.8808	0.8803	0.7893	0.7879
16	0.1112	0.1210	0.9968	0.9962	0.8829	0.8729	0.7932	0.7766
17	0.1090	0.1319	0.9968	0.9954	0.8836	0.8524	0.7943	0.7454
18	0.1068	0.1115	0.9968	0.9965	0.8852	0.8844	0.7972	0.7953
19	0.1065	0.1177	0.9968	0.9962	0.8859	0.8734	0.7984	0.7765
20	0.1074	0.1164	0.9968	0.9962	0.8834	0.8731	0.7941	0.7764

1.4 Kiểm tra mô hình với tập dữ liệu test

Kiểm tra mô hình với 20 ảnh trong thư mục kiểm tra "rgb_test", hiển thị kết quả phân vùng vết nứt của mô hình, hiển thị mặt nạ vết nứt tương ứng. So sánh kết quả dự đoán và mặt nạ vết nứt, và phân tích kết quả trong báo cáo.

Kết quả định lượng

Kết quả đánh giá trung bình trên 20 ảnh kiểm thử như sau:

- **Dice Score trung bình:** 0.8731
- **IoU trung bình:** 0.7764
- **Pixel Accuracy:** 99.62%

Kết quả chi tiết của một số mẫu tiêu biểu:

- **Mẫu tốt nhất (Sample 18):** Dice = 0.9167. Mô hình bắt trọn vẹn đường nứt phức tạp.
- **Mẫu trung bình (Sample 1):** Dice = 0.8429. Mô hình nhận diện đúng vị trí nhưng biên dạng chưa thực sự sắc nét.
- **Mẫu kém (Sample 13):** Dice = 0.6269. Đây là trường hợp ngoại lai (outlier), nguyên nhân do vết nứt quá mảnh hoặc độ tương phản thấp khiến mô hình bỏ sót.

Bảng 4: Kết quả định lượng chi tiết (Dice & IoU) trên 20 ảnh kiểm tra ngẫu nhiên

Sample ID	Dice	IoU	Sample ID	Dice	IoU
Sample 1	0.8429	0.7284	Sample 11	0.8917	0.8045
Sample 2	0.8679	0.7666	Sample 12	0.8837	0.7917
Sample 3	0.8698	0.7696	*Sample 13*	*0.6269*	*0.4566*
Sample 4	0.9062	0.8285	Sample 14	0.8521	0.7423
Sample 5	0.8900	0.8018	Sample 15	0.8482	0.7365
Sample 6	0.8234	0.6998	Sample 16	0.8635	0.7599
Sample 7	0.8739	0.7761	Sample 17	0.8327	0.7134
Sample 8	0.8593	0.7533	**Sample 18**	**0.9167**	**0.8463**
Sample 9	0.8889	0.8001	Sample 19	0.8707	0.7711
Sample 10	0.8085	0.6786	Sample 20	0.8573	0.7503
Trung bình (Average): Dice = 0.8577 | IoU = 0.7588					

Phân tích lỗi (Error Analysis):

1. **Hiện tượng đứt đoạn:** Ở một số vết nứt rất nhỏ (hairline cracks), mô hình dự đoán bị đứt đoạn, không liền mạch như nhãn gốc. Điều này phản ánh qua tỷ lệ pixel vết nứt dự đoán (2.15%) thấp hơn so với thực tế (2.91%).
2. **Nhiễu nền:** Mô hình xử lý tốt nhiễu bề mặt bê tông, hiếm khi nhận diện sai các vết ố thành vết nứt (False Positives thấp).

1.5 Giải pháp cải thiện kết quả

Dựa trên kết quả thực nghiệm (Dice Score đạt ≈ 0.88) và phân tích các trường hợp dự đoán sai (như hiện tượng đứt đoạn ở vết nứt mảnh hay sai số do điều kiện ánh sáng), nhóm thực hiện đề xuất các giải pháp kỹ thuật sau để nâng cao hiệu suất của hệ thống:

- **Tăng cường dữ liệu nâng cao (Advanced Data Augmentation):** Do tập dữ liệu huấn luyện còn hạn chế (416 ảnh), mô hình dễ gặp khó khăn với các biến thể lạ. Cần áp dụng thêm các kỹ thuật tăng cường dữ liệu chuyên sâu cho bài toán vết nứt như:
  - *Elastic Deformations:* Biến dạng đàn hồi để giả lập các thay đổi cấu trúc bề mặt vật liệu.
  - *Random Brightness/Contrast:* Thay đổi độ sáng và tương phản ngẫu nhiên để giúp mô hình chống lại nhiễu ánh sáng (khắc phục trường hợp ảnh quá tối hoặc thiếu sáng).
- **Tinh chỉnh hàm mất mát (Loss Function Engineering):** Kết quả cho thấy mô hình có xu hướng dự đoán vết nứt "mảnh" hơn thực tế (tỷ lệ dự đoán thấp hơn nhãn gốc). Để khắc phục, có thể thay thế Dice Loss bằng **Tversky Loss**. Bằng cách điều chỉnh tham số $\beta > 0.5$ trong công thức Tversky, ta có thể tăng trọng số phạt cho các trường hợp *False Negative* (bỏ sót vết nứt), buộc mô hình phải học các đường biên liền mạch hơn.
- **Nâng cấp kiến trúc mô hình (Architecture Upgrade):** Thay thế bộ mã hóa (Encoder) truyền thống bằng phương pháp **Transfer Learning** với các mạng Backbone mạnh mẽ hơn như *ResNet-34* hoặc *EfficientNet* đã được huấn luyện trước (Pre-trained) trên tập ImageNet. Các mạng này có khả năng trích xuất đặc trưng kết cấu (texture) tốt hơn, giúp phân biệt rõ vết nứt và nhiễu bề mặt. Ngoài ra, kiến trúc **Attention U-Net** cũng là một giải pháp tiềm năng để giúp mô hình tập trung vào các vùng quan trọng và lờ đi nhiễu nền.
- **Chiến lược huấn luyện và Hậu xử lý (Training & Post-processing):**
  - Áp dụng kỹ thuật *Test Time Augmentation (TTA)*: Thực hiện dự đoán trên nhiều phiên bản xoay/lật của ảnh đầu vào và lấy trung bình kết quả để tăng độ tin cậy.
  - Sử dụng các thuật toán hình thái học (Morphological Operations) như phép đóng (Closing) ở khâu hậu xử lý để nối liền các đoạn nứt bị đứt quãng nhỏ, đảm bảo tính liên tục của vết nứt.

2 Bài toán phân loại chủ đề tin tức

Đề bài: Hai file "train_02.csv" và "test_02.csv" chứa dữ liệu văn bản dùng cho bài toán phân loại chủ đề tin tức (News topic classification). Mỗi dòng trong mỗi file gồm hai cột: chỉ số loại chủ đề (class index) và mô tả (description). Chỉ số loại chủ để gồm: 1-Thế giới (World), 2-Thể thao (Sports), 3-Kinh doanh (Business), 4-Khoa học & Công nghệ (Science and Technology). Sinh viên thực hiện và báo cáo các nội dung sau:

2.1 Xây dựng mô hình Bi-LSTM

Xây dựng một bộ phân loại sử dụng nhiều lớp LSTM cùng các lớp khác để xác định 4 loại chủ đề tin tức. Liệt kê và mô tả tất cả các lớp trong mô hình, bao gồm: tên lớp, hàm kích hoạt. Đối với các lớp LSTM, cần mô tả số timestep và kích thước vector đầu vào. Giải thích lý do lựa chọn cấu trúc mô hình.

Cấu hình mô hình

Mô hình được xây dựng dựa trên kiến trúc Bi-directional LSTM (BiLSTM) sử dụng bộ từ điển $V = 20,000$ từ, độ dài chuỗi tối đa $T = 60$, và kích thước vector nhúng $D = 100$.

Bảng 5: Chi tiết kiến trúc mô hình Bi-directional LSTM

Tên lớp (Layer)	**Cấu hình chi tiết**	**Output Shape**	**Hàm kích hoạt**
Input	Sequence Input (Indices) Max Length: 60	(Batch, 60)	-
Embedding	**Lớp nhúng từ** - Vocab Size: 20,000 - Embed Dim: 100 - Param: 2,000,000 (80% tổng param)	(Batch, 60, 100)	Linear
Bi-LSTM	**Lớp LSTM 2 chiều** - Hidden Dim: 192 - Bidirectional: *True* - Input: Vector 100 chiều - Tổng hợp 2 chiều: $192 \times 2 = 384$	(Batch, 384)	Tanh / Sigmoid
FC1 (Linear)	**Lớp kết nối đầy đủ** - Input Features: 384 - Units: 128	(Batch, 128)	**ReLU**
Dropout	**Lớp chống Overfitting** - Rate: 0.3	(Batch, 128)	-
FC Output	**Lớp đầu ra** - Input Features: 128 - Units: 4 (Số lớp chủ đề)	(Batch, 4)	-

Lý do lựa chọn cấu trúc mô hình

- **Phân bổ tham số tối ưu:**
  - Lớp *Embedding* chiếm gần 80% lượng tham số (2 triệu), đảm bảo khả năng biểu diễn ngữ nghĩa phong phú cho 20,000 từ vựng.
  - Lớp *FC1 (Linear)* kết hợp với hàm kích hoạt *ReLU* đóng vai trò bộ phân loại phi tuyến tính, trích xuất các đặc trưng cấp cao từ vector ngữ cảnh của LSTM trước khi đưa ra quyết định cuối cùng.
- **Kiểm soát Overfitting:** Với tổng số tham số khoảng 2.5 triệu, việc sử dụng lớp *Dropout (p=0.3)* ngay trước lớp đầu ra là cần thiết để ngắt ngẫu nhiên các liên kết nơ-ron, giúp mô hình tổng quát hóa tốt hơn trên tập kiểm thử.

2.2 Xử lý dữ liệu văn bản

Trong báo cáo, mô tả chi tiết các bước xử lý dữ liệu (input, labels). Mô tả dữ liệu đầu vào của mô hình phân loại sau khi xử lý.

1. **Xử lý nhãn (Label Encoding):** Dữ liệu gốc có nhãn từ 1 đến 4. Để phù hợp với hàm mất mát *CrossEntropyLoss* trong PyTorch, nhãn được chuyển đổi về miền giá trị $[0,3]$ theo quy tắc:

   1 (World) $\rightarrow$ 0; 2 (Sports) $\rightarrow$ 1; 3 (Business) $\rightarrow$ 2; 4 (Science & Tech) $\rightarrow$ 3.

2. **Tiền xử lý văn bản (Text Preprocessing):** Văn bản được chuẩn hóa bằng cách chuyển về dạng chữ thường (lowercase), loại bỏ các ký tự đặc biệt và dấu câu thừa (ví dụ: "Reuters - Short-sellers..." $\rightarrow$ "reuters short sellers...").

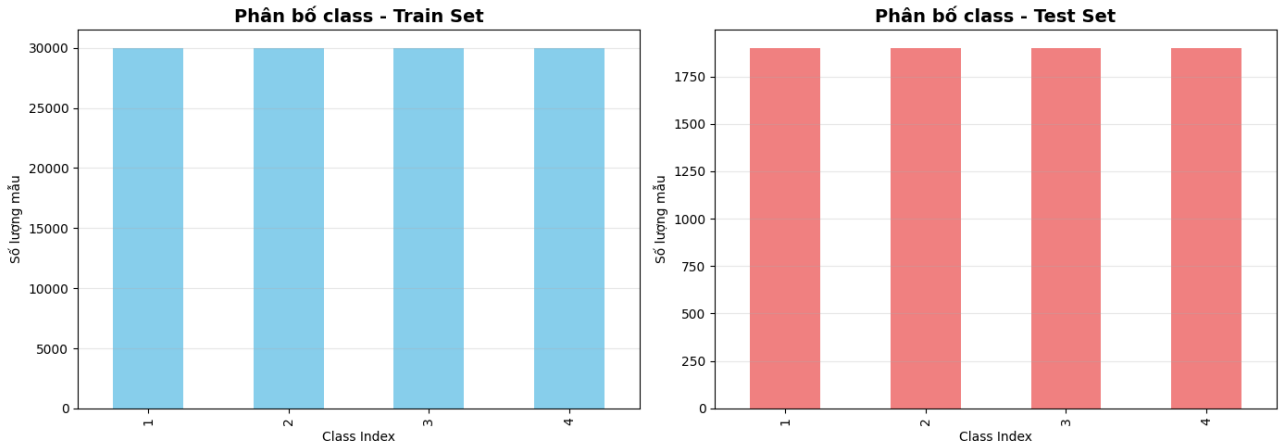
3. **Xây dựng từ điển (Vocabulary):** Từ tập dữ liệu huấn luyện (3.75 triệu token), thiết lập bộ từ điển gồm **20,000** từ phổ biến nhất để giảm thiểu tham số mô hình. Hai token đặc biệt được thêm vào:

4. **Mã hóa và Padding (Tokenization & Padding):** Mỗi câu văn bản được chuyển đổi thành chuỗi các số nguyên (index) dựa trên từ điển. Độ dài chuỗi được cố định là **60 tokens** (cắt bớt nếu dài hơn, thêm padding 0 nếu ngắn hơn).

5. **Phân chia tập dữ liệu (Data Splitting):** Tập huấn luyện gốc (120,000 mẫu) được chia tách theo tỷ lệ 80:20:
   - **Train Set:** 96,000 mẫu.
   - **Validation Set:** 24,000 mẫu.
   - **Test Set:** 7,600 mẫu (giữ nguyên từ file gốc).

Mô tả dữ liệu đầu vào của mô hình (Final Model Input):

- **Input IDs:** Tensor kích thước $(128, 60)$.
- **Labels:** Tensor kích thước $(128,)$. Chứa nhãn phân loại tương ứng $[0,1,2,3]$.



Hình 3: Biểu đồ phân bố class train/test

2.3 Thống kê tham số mô hình

Trong code, hiển thị số lượng tham số của từng lớp trong mô hình và báo cáo kết quả.

Tổng số lượng tham số của mô hình là **2,501,380**. Chi tiết phân bố tham số cho từng lớp được trình bày trong Bảng 6.

Bảng 6: Thống kê tham số chi tiết của mô hình Bi-LSTM

Lớp (Layer)	**Chi tiết (Configuration)**	**Tham số**	**Tỷ lệ (%)**
Embedding	Vocab: 20,000 × Dim: 100	2,000,000	79.96%
Bi-LSTM	Input: 100, Hidden: 192 (2 chiều)	451,584	18.05%
FC1 (Linear)	Input: 384, Output: 128	49,280	1.97%
Output (Linear)	Input: 128, Output: 4	516	0.02%
TỔNG CỘNG		**2,501,380**	**100%**

Nhận xét: Lớp Embedding chiếm phần lớn dung lượng bộ nhớ (gần 80%) để lưu trữ biểu diễn vector của 20,000 từ vựng.

2.4 Kết quả huấn luyện và đánh giá

Trình bày hàm mất mát, thuật toán tối ưu hóa, và learning rate mà sinh viên đã chọn. Giải thích lý do lựa chọn. Huấn luyện mô hình ít nhất 10 vòng lặp (iterations). Sau khi huấn luyện, trình bày và nhận xét kết quả train và test mà sinh viên đạt được.

Thiết lập huấn luyện (Training Setup)

- **Hàm mất mát (Loss Function):** *CrossEntropyLoss.*
  Lý do: Đây là hàm mất mát tiêu chuẩn cho bài toán phân loại đa lớp (Multi-class Classification), giúp tối ưu hóa xác suất dự đoán của 4 lớp chủ đề.

- **Thuật toán tối ưu (Optimizer):** *Adam.*
  Lý do: Adam tự động điều chỉnh tốc độ học (adaptive learning rate) cho từng tham số, giúp mô hình hội tụ nhanh và ổn định hơn so với SGD truyền thống đối với dữ liệu văn bản thưa.

- **Tốc độ học (Learning Rate):** Khởi tạo $LR = 0.0005$. Kết hợp với *ReduceLROnPlateau* để giảm LR nếu Loss không cải thiện, và *Early Stopping* để chống Overfitting.

Kết quả thực nghiệm

Mô hình được huấn luyện trong **20 epochs** (dừng sớm do không cải thiện sau 5 epochs liên tiếp). Kết quả tốt nhất đạt được tại **Epoch 15**.

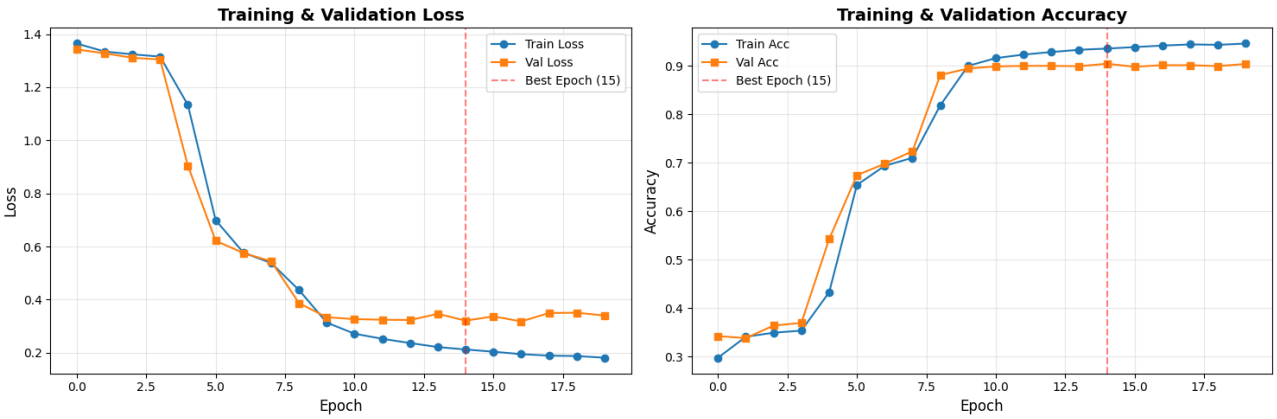
Bảng 7: Kết quả đánh giá trên tập Huấn luyện và Kiểm thử

Tập dữ liệu	Accuracy	Loss	F1-Score (Macro)
Train (Best Epoch)	93.60%	0.2121	-
Validation (Best Epoch)	90.47%	0.3206	-
Test Set	**89.39%**	-	**0.8936**

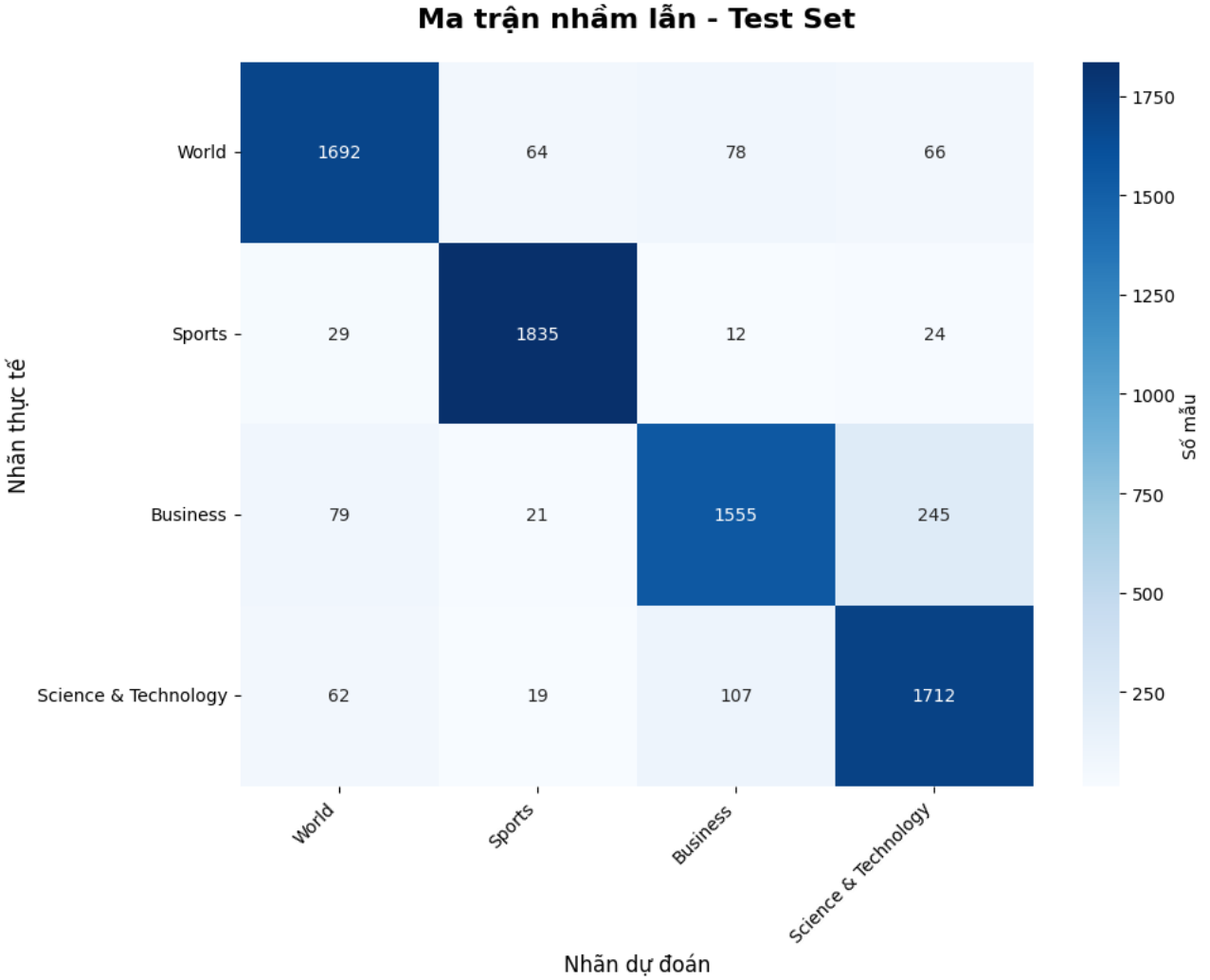
Nhận xét và Phân tích

- **Hiệu năng tổng quát:** Mô hình đạt độ chính xác kiểm thử **89.39%**, xấp xỉ với kết quả trên tập Validation (90.47%), cho thấy mô hình không bị Overfitting và có khả năng tổng quát hóa tốt.

- **Phân tích từng lớp chủ đề:**
  - *Tốt nhất:* Lớp **Sports** (F1-Score: 0.9560). Các từ vựng thể thao thường đặc thù và ít trùng lặp ngữ nghĩa.
  - *Khó nhất:* Lớp **Business** (F1-Score: 0.8516). Phân tích lỗi cho thấy các bản tin kinh tế thường bị nhầm lẫn với *World* (chính trị) hoặc *Sci/Tech* (công nghệ), do sự giao thoa về nội dung.



Hình 4: Biểu đồ phân bố class train/val



Hình 5: Ma trận nhầm lẫn

2.5 Đánh giá mô hình và đề xuất cải thiện

1. Ưu điểm và Nhược điểm

Ưu điểm:

- **Ngữ cảnh hai chiều (Bidirectional):** Khắc phục điểm yếu của LSTM thường khi tận dụng được thông tin từ cả hai phía (trước và sau), giúp hiểu rõ các từ đa nghĩa trong câu.
- **Ghi nhớ dài hạn:** Giải quyết tốt vấn đề triệt tiêu đạo hàm (Vanishing Gradient) của RNN truyền thống, nắm bắt được các phụ thuộc xa trong câu văn dài.
- **Tối ưu tài nguyên:** Mô hình nhẹ (2.5 triệu tham số), thời gian huấn luyện nhanh nhưng vẫn đạt độ chính xác cao ($\approx 89.4\%$).

Nhược điểm:

- **Lỗi ngữ nghĩa chồng chéo:** Dễ nhầm lẫn các lớp có từ vựng giao thoa (ví dụ: *Business* dễ nhầm với *World* hoặc *Sci/Tech*), dẫn đến F1-score của lớp *Business* thấp nhất (0.85).
- **Tốc độ huấn luyện tuần tự:** Do bản chất của mạng hồi quy (RNN), các bước tính toán phải thực hiện tuần tự theo thời gian, không tận dụng được tối đa khả năng tính toán song song của GPU như các mô hình CNN hay Transformer.

2. Đề xuất giải pháp cải thiện

1. **Dùng Pre-trained Embeddings (GloVe/Word2Vec):** Thay thế việc học Embedding từ đầu bằng các vector đã huấn luyện sẵn trên tập dữ liệu lớn để cải thiện khả năng biểu diễn ngữ nghĩa.
2. **Thêm cơ chế Attention:** Tích hợp lớp Attention sau Bi-LSTM để giúp mô hình tập trung trọng số vào các từ khóa quan trọng (keywords) thay vì xử lý bình đẳng mọi từ.
3. **Nâng cấp lên Transformer (BERT):** Sử dụng mô hình BERT (hoặc RoBERTa) để tận dụng cơ chế Self-Attention, giúp hiểu ngữ cảnh sâu hơn và có thể đạt độ chính xác >95%.
4. **Tăng cường dữ liệu (Data Augmentation):** Áp dụng kỹ thuật thay thế từ đồng nghĩa hoặc dịch ngược (Back-translation) tập trung vào lớp yếu nhất (*Business*) để cân bằng hiệu suất.

KẾT LUẬN CHUNG

Báo cáo đã hoàn thành việc nghiên cứu và thực nghiệm hai bài toán nền tảng trong lĩnh vực Học sâu:

- **Đối với bài toán Phân vùng vết nứt (Computer Vision):** Mô hình **U-Net** đã chứng minh hiệu quả vượt trội nhờ cơ chế *Skip Connections*, cho phép khôi phục chính xác các đường biên nứt mảnh mà không bị mất mát thông tin qua các lớp tích chập. Kết quả Dice Score đạt ≈ 0.89 khẳng định đây là kiến trúc chuẩn mực cho các bài toán phân đoạn y tế và kiểm định công trình.
- **Đối với bài toán Phân loại tin tức (NLP):** Kiến trúc **Bi-LSTM** đã khai thác thành công ngữ cảnh hai chiều của dữ liệu văn bản, đạt độ chính xác kiểm thử **89.39%**. Mặc dù còn hạn chế nhỏ ở các chủ đề có ngữ nghĩa giao thoa, mô hình vẫn cho thấy sự ưu việt của mạng hồi quy trong việc xử lý dữ liệu chuỗi so với các phương pháp truyền thống.

Tổng kết: Thông qua hai bài toán, nhóm thực hiện đã nắm vững quy trình xây dựng hệ thống Deep Learning từ khâu xử lý dữ liệu đặc thù (ảnh và văn bản), thiết kế kiến trúc mô hình phù hợp, đến các kỹ thuật tối ưu hóa nhằm đạt hiệu suất cao nhất trên tập dữ liệu thực tế.

PHỤ LỤC

Bài 1: $\rightarrow$ "$Source - Colab$" $\rightarrow$ "$Source - Github$"

Bài 2: $\rightarrow$ "$Source - Colab$" $\rightarrow$ "$Source - Github$"