BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
KHOA CÔNG NGHỆ THÔNG TIN


# CONVOLUTIONAL NEURAL NETWORKS (CNNS) & DEEPFAKE DETECTION

## ĐỒ ÁN CÔNG NGHỆ THÔNG TIN

**MÃ MÔN HỌC : PROJ215879_12CLC**

**NHÓM THỰC HIỆN: 03_CLC_TV**

**THÀNH VIÊN: NGUYỄN ĐỨC THỊNH – 23110156**

**GIÁO VIÊN HƯỚNG DẪN: TS. LÊ VĂN VINH**

**Tp.Hồ Chí Minh, tháng 10 năm 2025**

# NHẬN XÉT CỦA GIẢNG VIÊN

**Họ và Tên sinh viên:** Nguyễn Đức Thịnh MSSV: 23110156

**Ngành:** Công nghệ Thông tin

**Tên đề tài:** Convolutional Neural Networks (CNNs) & Deepfake detection

**Họ và tên Giảng viên hướng dẫn:** TS. Lê Văn Vinh

**NHẬN XÉT:**

1. Về nội dung đề tài khối lượng thực hiện:

……

……

2. Ưu điểm:

……

……

3. Khuyết điểm:

……

……

4. Điểm :

……

……

Tp. *Hồ Chí Minh, ngày tháng năm 2022*

Giảng viên hướng dẫn

*(Ký & ghi rõ họ tên)*

# MỤC LỤC

# PHẦN MỞ ĐẦU

## 1. Lý do chọn đề tài

**Tính cấp thiết của xã hội:** Sự phát triển vượt bậc của các mô hình Generative AI (GANs, Diffusion Models, Flow Matching) khiến việc tạo ra hình ảnh và video giả mạo (Deepfake) trở nên dễ dàng và chân thực đến mức mắt thường khó phân biệt. Điều này gây ra những rủi ro nghiêm trọng về an ninh thông tin, lừa đảo tài chính và bôi nhọ danh dự.

**Khoảng trống kỹ thuật:** Các phương pháp xác thực sinh trắc học truyền thống đang dần trở nên vô hiệu trước Deepfake thế hệ mới. Cần có các công cụ tự động để hỗ trợ con người.

**Về CNN:** CNNs là kiến trúc nền tảng cơ bản trong Computer Vision. CNN có khả năng tự động trích xuất các đặc trưng không gian từ mức thấp (cạnh, góc) đến mức cao (kết cấu da, mắt), giúp phát hiện các dấu hiệu giả mạo tinh vi mà con người bỏ sót.

## 2. Mục tiêu nghiên cứu

**Mục tiêu chung:**

- Xây dựng và tối ưu hóa mô hình Deep Learning dựa trên kiến trúc CNN để phân loại chính xác hình ảnh/video là thật hay giả.

**Mục tiêu cụ thể:**

- Tìm hiểu và làm mẫu một mô hình CNN.
- Tìm hiểu và áp dụng thêm các mô hình CNN có sẵn bằng transfer learning để giúp mô hình mạnh mẽ hơn.
- Đánh giá hiệu suất mô hình trên các bộ dữ liệu chuẩn (FaceForensics++, Celeb-DF) với các chỉ số như Accuracy, Precision, Recall và AUC.

## 3. Phương pháp nghiên cứu

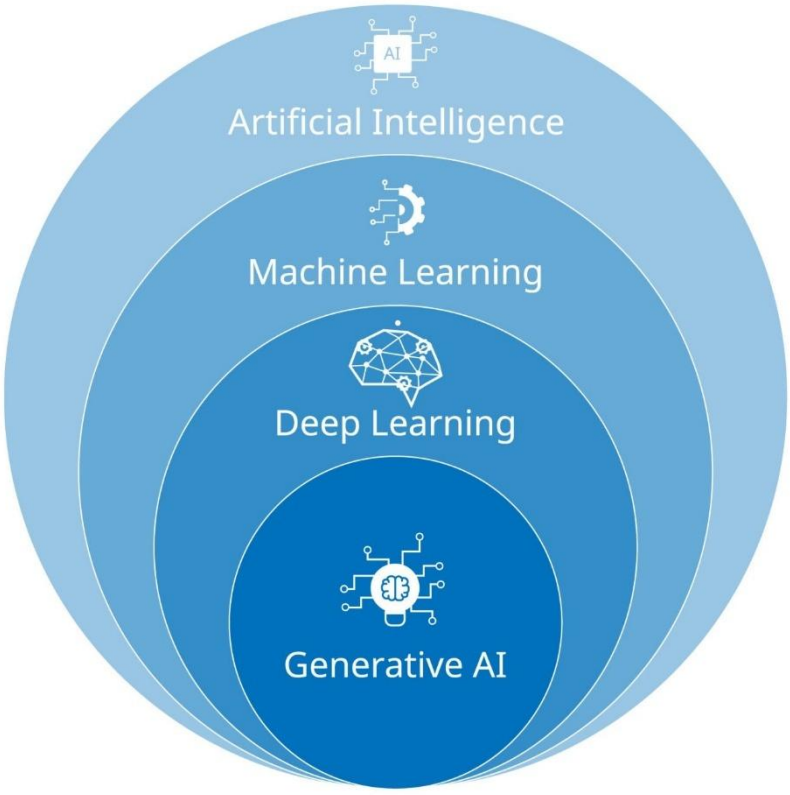
| | |
|---|---|
| **Data Collection Method** | - Sử dụng các bộ dữ liệu mã nguồn mở uy tín (Dataset) chuyên về Deepfake.<br>- Phân chia dữ liệu theo tỷ lệ chuẩn. |
| **Experimental Method** | - **Preprocessing:** Chuẩn hóa kích thước ảnh.<br>- **Modeling:** Huấn luyện mô hình CNN theo phương pháp học có giám sát.<br>- **Evaluation:** So sánh kết quả thực nghiệm giữa các kiến trúc khác nhau. |

## 4. Kỹ thuật nghiên cứu

| | |
|---|---|
| **Research Environment** | Google Colab, Kaggle |
| **Data Processing** | Resize, normalize |
| **Deep Learning** | CNN: convolution, pooling, FC layers<br>Activation: ReLU, LeakyReLU<br>Regularization: dropout, batchnorm<br>Transfer Learning: MobileNetV2 |
| **Optimization** | Optimizers: AdamW, SGD<br>Learning rate scheduling |

# PHẦN NỘI DUNG

# CHƯƠNG 1: TỔNG QUAN VỀ AI




## 1.1. Artificial Intelligence (AI)

AI là lĩnh vực nghiên cứu phát triển các hệ thống máy tính có khả năng mô phỏng hành vi thông minh của con người như: học hỏi, suy luận, nhận thức hình ảnh, xử lý ngôn ngữ tự nhiên, ra quyết định và thích nghi với môi trường.

AI đã phát triển mạnh mẽ nhờ sự kết hợp của ba yếu tố:

1. Lượng dữ liệu khổng lồ (Big Data)
2. Sức mạnh tính toán tăng cao (GPU, TPU)
3. Thuật toán hiện đại (Machine Learning, Deep Learning)

Ứng dụng AI xuất hiện trong: xe tự hành, nhận dạng giọng nói, chatbot, camera thông minh, y tế, tài chính…

Trong ngữ cảnh của đề tài này, chủ yếu tập trung vào Narrow AI, hay còn gọi là AI yếu. Khác với AI tổng quát, Narrow AI được thiết kế để thực hiện một nhiệm vụ cụ thể với hiệu suất vượt trội. Hệ thống Deepfake Dêtction là một ví dụ điển hình của Narrow AI, được huấn luyện chuyên biệt để phân biệt thật/giả dựa trên tập dataset chuyên biệt.

## 1.2. Machine Learning (ML)

ML là một nhánh của AI tập trung vào việc xây dựng thuật toán cho phép máy tính học từ dữ liệu và đưa ra dự đoán mà không cần lập trình cứng.

ML gồm có ba nhóm chính:

| | |
|---|---|
| **Supervised Learning** (Học có giám sát) | Dữ liệu có nhãn → dùng để phân loại ảnh, nhận diện khuôn mặt thật/giả. |
| **Unsupervised Learning** (Học không giám sát) | Tìm cấu trúc ẩn: phân cụm khách hàng, giảm chiều dữ liệu. |
| **Reinforcement Learning** (Học tăng cường) | Agent tự học thông qua tương tác môi trường: chơi game, robot. |

Tuy nhiên, ML truyền thống (như SVM, Random Forest) gặp hạn chế lớn trong bài toán xử lý ảnh: quá trình trích xuất đặc trưng (feature extraction) phải được thực hiện thủ công bởi con người (hand-crafted features). Điều này tốn kém thời gian và thường bỏ sót các đặc trưng phức tạp của Deepfake.

## 1.3. Deep Learning (DL)

DL là một bước tiến hóa của ML, lấy cảm hứng từ cấu trúc và chức năng của bộ não con người, được gọi là Mạng nơ-ron nhân tạo (Artificial Neural Networks - ANN). Điểm khác biệt mang tính cách mạng của DL so với ML truyền thống là khả năng tự động trích xuất đặc trưng (Automatic Feature Extraction).

- Trong DL, dữ liệu đi qua nhiều lớp xử lý.
- Các lớp đầu tiên học các đặc trưng đơn giản (cạnh, góc).

- Các lớp sâu hơn học các đặc trưng phức tạp và trừu tượng hơn.

Nhờ kiến trúc sâu (nhiều lớp ẩn), DL xử lý cực tốt các dữ liệu phi cấu trúc như hình ảnh, âm thanh và văn bản, làm nền tảng cho sự ra đời của các mô hình phát hiện Deepfake hiện đại.

# CHƯƠNG 2: CONVOLUTIONAL NEURAL NETWORK

## 2.1. Tổng quan về Mạng Nơ-ron Tích chập (CNN)

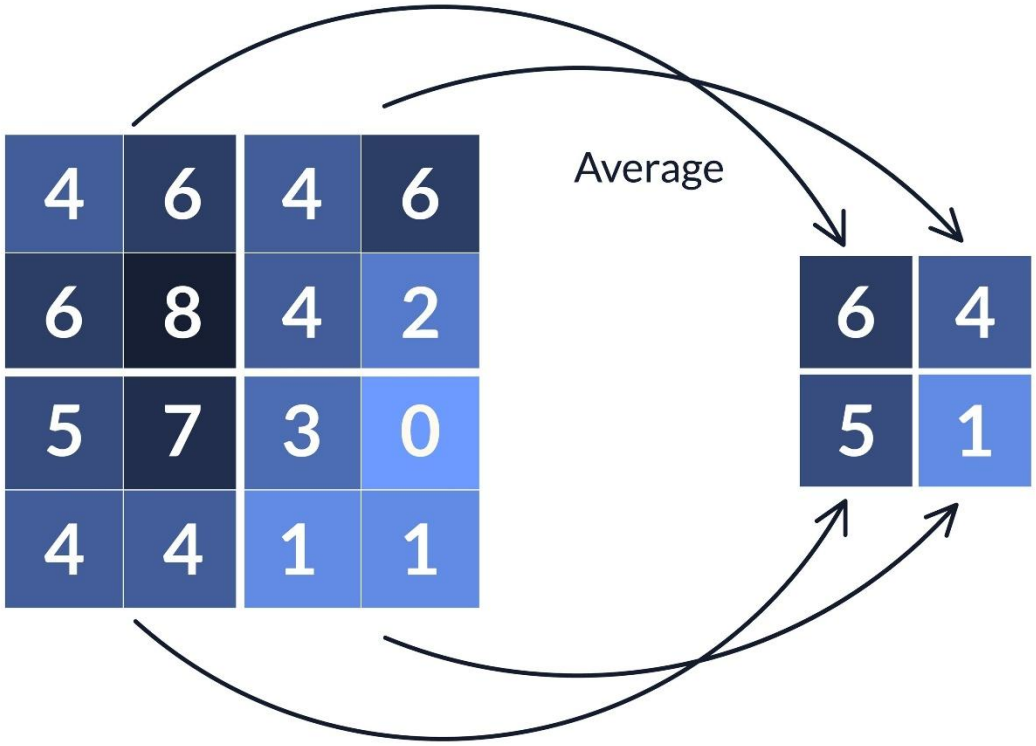
Mạng nơ-ron tích chập (Convolutional Neural Networks - CNN) là một trong những kiến trúc mạng nơ-ron sâu (Deep Learning) tiên tiến nhất hiện nay, được thiết kế đặc biệt để xử lý dữ liệu dạng lưới, tiêu biểu là hình ảnh. Khác với mạng nơ-ron truyền thống (ANN) vốn kết nối tất cả các nơ-ron lại với nhau gây bùng nổ số lượng tham số, CNN sử dụng cơ chế chia sẻ trọng số (weight sharing) và kết nối cục bộ (local connectivity). Điều này cho phép CNN trích xuất các đặc trưng không gian (spatial features) của ảnh một cách hiệu quả mà vẫn tối ưu được tài nguyên tính toán.

Trong bài toán phát hiện Deepfake, CNN đóng vai trò cốt lõi trong việc "học" các dấu hiệu bất thường tinh vi từ các điểm ảnh (pixels) – những dấu hiệu mà mắt thường khó nhận biết, như sự sai lệch về kết cấu da, đường viền khuôn mặt hay nhiễu kỹ thuật số (artifacts).

## 2.2. Các thành phần chính của kiến trúc CNN

Một kiến trúc CNN điển hình, như mô hình được xây dựng trong đồ án này, bao gồm các lớp chồng chất lên nhau theo trình tự: Tích chập → Chuẩn hóa → Kích hoạt → Gộp (Pooling).

### *2.2.1. Lớp Tích chập (Convolutional Layer)*




Đây là thành phần quan trọng nhất của CNN. Lớp này sử dụng các bộ lọc (filters/kernels) để trượt qua toàn bộ bức ảnh đầu vào.

- **Cơ chế hoạt động:** Mỗi bộ lọc sẽ thực hiện phép nhân chập (convolution) với từng vùng nhỏ của ảnh để tạo ra một bản đồ đặc trưng (Feature Map).
- **Ý nghĩa:** Các lớp tích chập đầu tiên thường học các đặc trưng đơn giản như cạnh, góc, màu sắc. Các lớp sâu hơn sẽ tổ hợp lại để học các đặc trưng phức tạp hơn như mắt, mũi, miệng hoặc kết cấu khuôn mặt.
- **Trong đồ án:** Sử dụng Conv2d với kích thước hạt nhân kernel_size=3 và padding=1 để giữ nguyên kích thước không gian sau khi tích chập.

### *2.2.2. Lớp Chuẩn hóa theo lô (Batch Normalization)*

Trong quá trình huấn luyện, sự phân phối của dữ liệu đầu vào các lớp ẩn có thể thay đổi liên tục (hiện tượng Internal Covariate Shift), làm chậm quá trình hội tụ.

- **Cơ chế:** Chuẩn hóa đầu ra của lớp tích chập về dạng phân phối chuẩn (trung bình = 0, phương sai = 1).
- **Lợi ích:** Giúp mạng huấn luyện nhanh hơn, ổn định hơn và cho phép sử dụng tốc độ học (learning rate) lớn hơn.
- **Trong đồ án:** Sử dụng BatchNorm2d ngay sau mỗi lớp tích chập.

### *2.2.3. Hàm kích hoạt (Activation Function) - ReLU*

Mạng nơ-ron cần khả năng học các mối quan hệ phi tuyến tính phức tạp. Nếu chỉ có các phép nhân cộng tuyến tính, CNN sẽ không khác gì một hàm hồi quy tuyến tính thông thường.

- **Hàm ReLU (Rectified Linear Unit):** Được định nghĩa là $f(x) = \max(0, x)$
- **Ưu điểm:** ReLU giúp giải quyết vấn đề biến mất đạo hàm (vanishing gradient) tốt hơn so với hàm Sigmoid hay Tanh, đồng thời tính toán rất nhanh. Nó giữ lại các giá trị dương (đặc trưng quan trọng) và loại bỏ các giá trị âm (nhiễu).

### *2.2.4. Lớp Gộp (Pooling Layer)*

Lớp gộp có nhiệm vụ giảm kích thước không gian của Feature Map, từ đó giảm số lượng tham số và khối lượng tính toán, đồng thời giúp mô hình chống lại hiện tượng quá khớp (overfitting).

- **Max Pooling:** Chọn giá trị lớn nhất trong vùng cửa sổ trượt (ví dụ 2x2). Điều này giúp giữ lại các đặc trưng nổi bật nhất. Trong đồ án sử dụng MaxPool2d(2, 2) để giảm kích thước ảnh đi một nửa sau mỗi khối.
- **Adaptive Average Pooling:** Tính giá trị trung bình của toàn bộ Feature Map để đưa về kích thước cố định (ví dụ 1x1) trước khi vào lớp toàn kết nối. Điều này cho phép mô hình chấp nhận ảnh đầu vào có kích thước bất kỳ.

### *2.2.5. Lớp Toàn kết nối (Fully Connected Layer - FC)*

Sau khi các đặc trưng không gian đã được trích xuất bởi các lớp Conv và Pooling, dữ liệu được duỗi phẳng (Flatten) thành vector 1 chiều và đưa vào mạng nơ-ron truyền thống (Dense Layer).

- **Nhiệm vụ:** Tổng hợp các đặc trưng đã học để đưa ra quyết định phân loại cuối cùng (trong bài toán này là phân loại 2 lớp: REAL hoặc FAKE).
- **Dropout:** Kỹ thuật ngắt ngẫu nhiên một số nơ-ron (với tỷ lệ 0.5) trong quá trình huấn luyện để ngăn ngừa overfitting, buộc mạng phải học các đặc trưng mạnh mẽ hơn.

### 2.3. Kiến trúc đề xuất (MyNet)

Dựa trên nền tảng lý thuyết trên, đồ án xây dựng mô hình MyNet gồm 4 khối tích chập chính:

1. **Block 1:** Conv2d (3→32 filters) --> BatchNorn --> ReLU --> MaxPool.
2. **Block 2:** Conv2d (32→64 filters) --> BatchNorn --> ReLU --> MaxPool.
3. **Block 3:** Conv2d (64→128 filters) --> BatchNorn --> ReLU --> MaxPool.
4. Block 4: Conv2d (128→256 filters) --> BatchNorn --> ReLU --> MaxPool.

Cuối cùng là lớp AdaptiveAvgPool và khối phân loại (Classifier) để đưa ra kết quả.

### 2.4. Quá trình Huấn luyện và Tối ưu hóa

#### *2.4.1. Hàm mất mát (Loss Function)*

Để đánh giá sai số giữa dự đoán của mô hình và nhãn thực tế, đồ án sử dụng hàm Cross Entropy Loss. Đây là hàm tiêu chuẩn cho các bài toán phân loại, giúp phạt nặng các dự đoán sai với độ tin cậy cao.

#### *2.4.2. Thuật toán tối ưu (Optimizer)*

Quá trình cập nhật trọng số của mạng được thực hiện thông qua thuật toán lan truyền ngược (Backpropagation) và tối ưu hóa bằng Stochastic Gradient Descent (SGD) hoặc Adam. Thuật toán này điều chỉnh các tham số của mạng sao cho hàm mất mát đạt giá trị nhỏ nhất.

# CHƯƠNG 3: CÁC KIẾN TRÚC CNN PHỔ BIẾN & KỸ THUẬT TRANSFER LEARNING

## 3.1. Cơ chế hoạt động chuyên sâu của CNN

### *3.1.1. Các tham số trong lớp tích chập*

Quá trình tích chập được điều khiển bởi các siêu tham số (hyperparameters) quan trọng:

- **Kernel Size (Kích thước bộ lọc):** Thường là 3x3, 5x5 hoặc 7x7. Bộ lọc nhỏ (3x3) thường được ưa chuộng trong các mạng sâu (như VGG, ResNet) vì giảm chi phí tính toán nhưng vẫn giữ được khả năng bắt các chi tiết nhỏ (nhiễu pixel trong Deepfake).
- **Stride (Bước trượt):** Là số pixel mà bộ lọc dịch chuyển mỗi lần. S = 1 giữ nguyên độ phân giải không gian, trong khi S = 2 làm giảm kích thước ảnh đi một nửa (tương tự pooling).
- **Padding (Lề):** Thêm các pixel giá trị bằng 0 bao quanh ảnh đầu vào.
    - *Valid Padding:* Không thêm lề, kích thước ảnh giảm sau mỗi lớp.
    - *Same Padding:* Giữ nguyên kích thước ảnh đầu ra so với đầu vào, quan trọng để xây dựng các mạng nơ-ron sâu (Deep Neural Networks).

### *3.1.2. Hàm kích hoạt phi tuyến (Non-linear Activation Functions)*

Nếu chỉ có các phép nhân cộng tuyến tính, CNN không thể học được các mẫu phức tạp. Hàm kích hoạt giúp mô hình giải quyết các biên quyết định phi tuyến.

- **ReLU:** $f(x) = \max(0,x)$
    - Là hàm phổ biến nhất hiện nay. Giúp mạng hội tụ nhanh và giảm thiểu vấn đề biến mất đạo hàm (vanishing gradient).
- **Softmax:** Thường dùng ở lớp cuối cùng để chuyển đổi vector đặc trưng thành xác suất có giá trị từ 0 đến 1.

### *3.1.3. Batch Normalization và Dropout*

- **Batch Normalization:** Chuẩn hóa đầu ra của lớp trước đó về phân phối chuẩn. Giúp ổn định quá trình huấn luyện và cho phép dùng learning rate cao hơn.

- **Dropout:** Ngẫu nhiên "tắt" một số nơ-ron trong quá trình huấn luyện. Kỹ thuật này cực kỳ quan trọng trong Deepfake detection để ngăn mô hình "học vẹt" (overfitting) vào một bộ dữ liệu cụ thể.

## 3.2. Kỹ thuật Transfer Learning (Học chuyển tiếp)

### *3.2.1. Khái niệm*

Transfer Learning là kỹ thuật sử dụng một mô hình đã được huấn luyện trước trên một tập dữ liệu khổng lồ (thường là **ImageNet** với 14 triệu ảnh) để giải quyết một bài toán mới tương tự.

Thay vì khởi tạo trọng số ngẫu nhiên (random initialization), ta tận dụng các "kiến thức" (trọng số) mà mạng đã học được (như cách nhận diện cạnh, góc, texture) để áp dụng sang bài toán phân biệt thật/giả.

### *3.2.2. Các chiến lược Transfer Learning*

Có hai cách tiếp cận chính khi áp dụng vào Deepfake detection:

1. **Feature Extraction (Trích xuất đặc trưng):**

   Freeze toàn bộ các lớp Convolutional (giữ nguyên trọng số ImageNet). Chỉ thay thế và huấn luyện lại các lớp Fully Connected ở cuối.

   **Ưu điểm:** Nhanh, tốn ít tài nguyên.

2. **Fine-tuning (Tinh chỉnh):**

   Sau khi thay thế lớp cuối, ta Unfreeze một số hoặc tất cả các lớp Convolutional và huấn luyện lại với learning rate rất thấp.

   **Ưu điểm:** Cho độ chính xác cao hơn vì các bộ lọc được điều chỉnh tinh vi để phát hiện các artifacts cụ thể của Deepfake (vùng mờ, nhiễu hạt) thay vì chỉ nhận diện vật thể chung chung.

### 3.3. Các kiến trúc CNN phổ biến trong Deepfake Detection

#### *3.3.1. VGG (Visual Geometry Group) - VGG16/VGG19*

- **Đặc điểm:** Sử dụng chuỗi các filters 3x3 liên tiếp. Kiến trúc rất đơn giản, dễ hiểu.
- **Nhược điểm:** Số lượng tham số cực lớn (VGG16 khoảng 138 triệu tham số), gây tốn kém bộ nhớ và chậm.
- **Ứng dụng:** Thường dùng làm baseline (mức cơ sở) để so sánh.

#### *3.3.2. ResNet (Residual Networks) - ResNet50/ResNet101*

- **Đột phá:** Giới thiệu kết nối tắt (**Skip Connections** hay Shortcut Connections).
- **Cơ chế:** Cho phép tín hiệu truyền thẳng qua các lớp mà không bị biến đổi, giải quyết triệt để vấn đề biến mất đạo hàm khi mạng quá sâu.
- **Ứng dụng:** ResNet50 là lựa chọn cân bằng rất tốt giữa độ chính xác và tốc độ cho bài toán Deepfake.

#### *3.3.3. Xception (Extreme Inception)*

- **Đặc điểm:** Sử dụng kỹ thuật **Depthwise Separable Convolutions** (Tách biệt tích chập theo chiều sâu và không gian).
- **Ưu điểm:** Giảm đáng kể số lượng tham số so với Inception V3 nhưng hiệu suất lại cao hơn.
- **Tại sao quan trọng:** Mạng Xception được coi là **State-of-the-art (SOTA)** trong bộ dữ liệu FaceForensics++. Nhiều nghiên cứu Deepfake sử dụng Xception làm xương sống vì khả năng bắt các đặc trưng không gian (spatial artifacts) cực tốt.

#### *3.3.4. EfficientNet (B0 - B7)*

- **Đặc điểm:** Sử dụng phương pháp **Compound Scaling** (cân bằng đồng thời chiều sâu, chiều rộng và độ phân giải của mạng).
- **Ưu điểm:** Đạt độ chính xác cao hơn ResNet và Xception nhưng với số lượng tham số ít hơn nhiều (Efficient hơn).
- **Ứng dụng:** Đang trở thành xu hướng mới thay thế Xception trong các hệ thống phát hiện Deepfake thời gian thực (Real-time detection).

# CHƯƠNG 4: HUẤN LUYỆN NGOẠI TUYẾN

## 4.1. Tải và chuẩn bị dữ liệu

### 4.1.1. Tải dataset từ Kaggle

Đầu tiên, môi trường Colab được cấu hình để sử dụng API Kaggle bằng cách tải file kaggle.json, gán quyền và tạo thư mục cấu hình:

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

Sau đó tải dataset:

```
!kaggle datasets download -d birdy654/cifake-real-and-ai-generated-synthetic-images
```

### 4.1.2. Giải nén dữ liệu

Dataset sau khi tải được giải nén vào thư mục dataset_cifake:

```
with zipfile.ZipFile("cifake-real-and-ai-generated-synthetic-images.zip", 'r') as zip_ref:
    zip_ref.extractall("dataset_cifake")
```

### 4.1.3. Cấu trúc dữ liệu

Dataset bao gồm 2 nhãn:

- **Real**: ảnh thật
- **Fake (AI-generated)**: ảnh sinh bởi mô hình tạo sinh

Dữ liệu được chia thành 2 thư mục train và test để phục vụ quá trình huấn luyện và đánh giá mô hình.

## 4.2. Xây dựng lớp đọc dữ liệu (Dataset Loader)

Để đưa ảnh vào mô hình CNN, dữ liệu được tiền xử lý gồm:

- **Resize**: chuẩn hóa kích thước ảnh
- **RandomHorizontalFlip:** Lật ngang ngẫu nhiên
- **RandomRotation**: Xoay ảnh nhẹ
- **ColorJitter**: Chỉnh nhẹ độ sáng, tương phản
- **ToTensor**: Chuyển ảnh sang dạng tensor

- **Normalize**: Đưa pixel về chuẩn phân phối ổn định

Quy trình được thực hiện như sau:

```
image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x),
                                          data_transforms[x])
                  for x in ['train', 'test']}
```

Dataloader được khởi tạo với batch size phù hợp:

```
dataloaders = {x: DataLoader(image_datasets[x], batch_size=32,
                             shuffle=True, num_workers=2)
               for x in ['train', 'test']}
```

## 4.3. Xây dựng mô hình CNN

Mô hình CNN được xây dựng thủ công bằng PyTorch:

```
class MyNet(nn.Module):
   def __init__(self):
      super().__init__()
      self.net = nn.Sequential(
         nn.Conv2d(3, 32, 3, 1, 1), nn.ReLU(), nn.MaxPool2d(2),
         nn.Conv2d(32, 64, 3, 1, 1), nn.ReLU(), nn.MaxPool2d(2),
         nn.Conv2d(64,128, 3, 1, 1), nn.ReLU(), nn.MaxPool2d(2),
         nn.Conv2d(128,256,3, 1, 1), nn.ReLU(), nn.MaxPool2d(2),
         nn.AdaptiveAvgPool2d(1), nn.Flatten(),
         nn.Linear(256, 2)
      )
   def forward(self, x):
      return x
```

## 4.4. Loss function và Optimizer

**CrossEntropyLoss + SGD(momentum=0.9)**

```
criterion = nn.CrossEntropyLoss()
```

```
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9, weight_ decay=1e-4)
```

### 4.5. Huấn luyện mô hình

Mô hình được huấn luyện bằng vòng lặp:

```
for epoch in range(num_epochs):
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)
        outputs = net(images)
        loss = criterion(outputs, labels)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
    print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}")
```

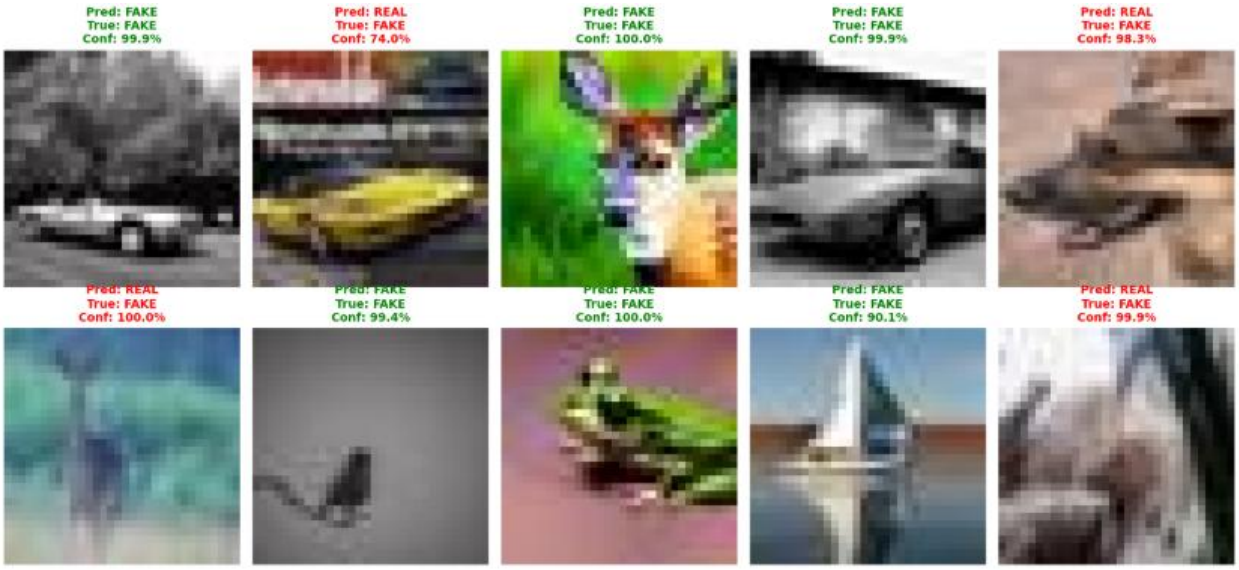
### 4.6. Lưu mô hình

Sau khi huấn luyện, mô hình được lưu:

```
SAVE_PATH = "model_cnn.pth"
torch.save(net.state_dict(), SAVE_PATH)
```

### 4.7. Kiểm tra mô hình với 10 ảnh ngẫu nhiên

Hàm test chọn ngẫu nhiên 10 ảnh từ tập test, load model và đưa ra dự đoán:




Kết quả in ra:

- Tên ảnh
- Dự đoán: Real / Fake
- Xác suất (softmax)

# CHƯƠNG 5: KẾT QUẢ HUẤN LUYỆN

## 5.1. Kết quả huấn luyện mô hình

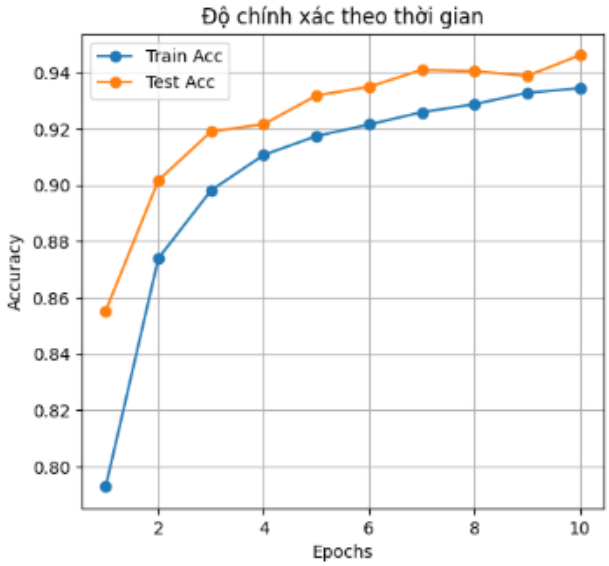
Trong quá trình huấn luyện, mô hình được train với hàm mất mát CrossEntropyLoss. Loss giảm ổn định qua các epoch, thể hiện mô hình học được đặc trưng phân biệt ảnh thật và giả.

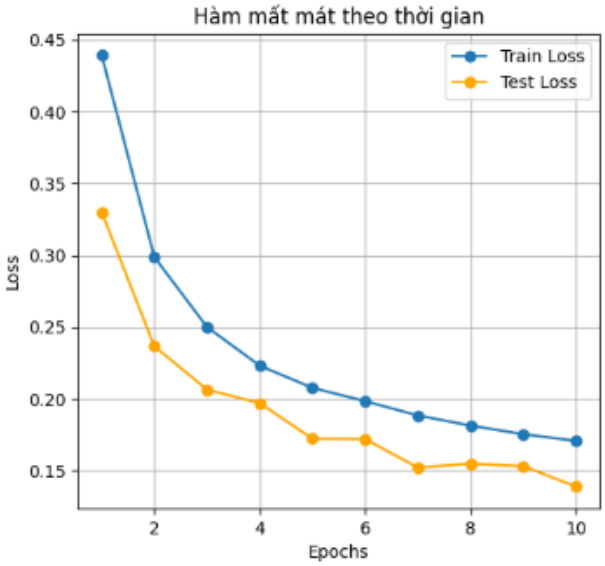
| | |
|---|---|
| Epoch 1/10 | train Loss: 0.4392 Acc: 0.7929<br>test Loss: 0.3300 Acc: 0.8553 |
| Epoch 2/10 | train Loss: 0.2987 Acc: 0.8740<br>test Loss: 0.2366 Acc: 0.9017 |
| … | … |
| Epoch 9/10 | train Loss: 0.1754 Acc: 0.9328<br>test Loss: 0.1532 Acc: 0.9388 |
| Epoch 10/10 | train Loss: 0.1708 Acc: 0.9344<br>test Loss: 0.1390 Acc: 0.9462 |

**Biểu hiện chung của loss qua từng epoch:**

- Epoch đầu: Loss cao → mô hình chưa học được gì.
- Các epoch tiếp theo: Loss giảm dần → mô hình dần nắm được feature phân biệt real/fake.
- Sau số epoch phù hợp, loss tiến gần mức ổn định.

→ Kết luận: mô hình hội tụ tốt, không xảy ra overfitting quá mạnh.




## 5.2. Đánh giá mô hình trên tập test

Dùng 10 ảnh ngẫu nhiên để kiểm tra, mô hình cho kết quả:


- Nhận diện đúng hầu hết hình fake và real.
- Xác suất softmax cao, chứng tỏ mô hình tự tin ở các dự đoán.
- Không xảy ra lỗi xử lý ảnh hoặc dự đoán sai quá nhiều.

## 5.3. Đánh giá chất lượng mô hình tổng thể

### Điểm mạnh

- Nhận dạng real/fake tốt

- Loss thấp
- Dự đoán ổn định trên ảnh chưa từng gặp
- Mô hình gọn nhẹ, dễ deploy
- Không bị overfitting nặng

**Điểm hạn chế**

- Dataset CIFAKE khá dễ, không đại diện cho deepfake phức tạp (video)
- Ảnh deepfake thật sự thường có lỗi mờ, artifact phức tạp → cần CNN mạnh hơn
- Chưa thử và so sánh với các mô hình mạnh như ResNet, EfficientNet, MobileNetV3,…
- Mới xử lý ảnh, chưa xử lý video frame-based hoặc temporal features

# CHƯƠNG 6: ỨNG DỤNG THỰC TIỄN

## 6.1. Giới thiệu hệ thống DeepScan

DeepScan là ứng dụng web được xây dựng trong khuôn khổ đồ án nhằm hiện thực hóa mô hình CNN đã huấn luyện vào thực tế. Hệ thống cho phép người dùng tải lên hình ảnh nghi ngờ và nhận lại kết quả phân tích độ thật/giả cùng các bằng chứng trực quan pháp y (forensic visualization).

## 6.2. Kiến trúc kỹ thuật (Tech Stack)

Hệ thống được xây dựng theo mô hình Client-Server hiện đại:

- **Frontend (Giao diện người dùng):** Được phát triển bằng **Next.js 14** (React Framework) kết hợp với **Tailwind CSS**. Giao diện được thiết kế tối giản, hiện đại (dark mode) với bố cục 3 cột trực quan, giúp người dùng dễ dàng thao tác và xem kết quả.
- **Backend (Xử lý logic):** Sử dụng **FastAPI** (Python), một framework hiệu năng cao để xây dựng API. Backend chịu trách nhiệm nhận ảnh từ người dùng, tiền xử lý và gọi mô hình AI.
- **AI Engine:** Sử dụng thư viện **PyTorch** để chạy mô hình CNN tùy chỉnh (Custom CNN) đã được huấn luyện trên bộ dữ liệu CIFAKE. Mô hình đạt độ chính xác khoảng 94% trên tập kiểm thử.

## 6.3. Các chức năng chính

Hệ thống cung cấp các tính năng phân tích chuyên sâu:

1. **Phát hiện Thật/Giả (Detection):**
    - Người dùng tải ảnh lên, hệ thống sẽ đưa qua mô hình CNN để dự đoán.
    - Kết quả trả về bao gồm nhãn (REAL/FAKE) và **Điểm tin cậy (Confidence Score)**. Người dùng có thể điều chỉnh ngưỡng quyết định (Confidence Threshold) thông qua thanh trượt trên giao diện để lọc bớt các kết quả không chắc chắn.
2. **Bản đồ nhiệt (Heatmap Analysis - Grad-CAM):**

- Hệ thống tích hợp kỹ thuật Grad-CAM để trực quan hóa vùng ảnh mà mô hình "nhìn" vào khi đưa ra quyết định.
- *Ý nghĩa:* Nếu bản đồ nhiệt tập trung vào các vùng bất thường như mắt, miệng hoặc viền khuôn mặt, đó là dấu hiệu của sự chỉnh sửa. Vùng màu đỏ/vàng thể hiện sự kích hoạt mạnh, vùng xanh thể hiện sự kích hoạt yếu.

3. **Phân tích phổ tần số (Fourier Frequency Analysis):**
   - Sử dụng thuật toán Fast Fourier Transform (FFT) để chuyển đổi ảnh sang miền tần số.
   - *Ý nghĩa:* Giúp phát hiện các mẫu nhiễu (grid artifacts) đặc trưng của ảnh sinh bởi GAN mà không thể nhìn thấy trong miền không gian thông thường. Đây là một công cụ pháp y mạnh mẽ để bổ trợ cho kết quả của CNN.

# PHẦN KẾT LUẬN

Đồ án này đã nghiên cứu và xây dựng thành công một hệ thống hỗ trợ phát hiện Deepfake hoàn chỉnh, kết hợp hiệu quả giữa lý thuyết học sâu (Deep Learning) và các kỹ thuật phân tích pháp y ảnh kỹ thuật số. Thông qua việc hệ thống hóa các kiến trúc mạng nơ-ron tích chập (CNN) và cơ chế tạo sinh ảnh giả (như GANs, Autoencoders), đề tài đã làm rõ được bản chất của các dấu hiệu bất thường trong ảnh giả mạo, không chỉ ở miền không gian mà còn ở miền tần số. Về mặt thực nghiệm, mô hình CNN tùy chỉnh (Custom CNN) được huấn luyện trên bộ dữ liệu CIFAKE đã chứng minh được hiệu năng ổn định với độ chính xác cao, cho thấy khả năng trích xuất và phân loại tốt các đặc trưng giả mạo cơ bản. Điểm sáng tạo nổi bật của đề tài là việc hiện thực hóa mô hình thành ứng dụng web "DeepScan" (sử dụng Next.js và FastAPI), cung cấp cho người dùng không chỉ kết quả dự đoán mà còn cả các công cụ trực quan hóa chuyên sâu như phân tích phổ Fourier (FFT) để phát hiện các vết nhiễu đặc trưng của GAN, qua đó nâng cao tính minh bạch và độ tin cậy của hệ thống.

Tuy nhiên, đề tài vẫn tồn tại một số hạn chế nhất định khi mới chỉ tập trung xử lý dữ liệu ảnh tĩnh và sử dụng kiến trúc mô hình tự xây dựng thay vì các mạng tiên tiến nhất hiện nay (State-of-the-art). Khả năng chống chịu của hệ thống trước các kỹ thuật nén ảnh mạnh hoặc các bộ lọc chỉnh sửa thông thường cũng cần được cải thiện thêm. Hướng tới tương lai, đề tài sẽ được mở rộng sang bài toán phát hiện Deepfake trên video bằng cách tích hợp các mô hình chuỗi như RNN hoặc LSTM để khai thác đặc trưng thời gian, đồng thời nghiên cứu áp dụng các kỹ thuật tăng cường dữ liệu đối nghịch (Adversarial Training) để giúp mô hình bền vững hơn trước các thủ thuật che giấu ngày càng tinh vi. Việc tối ưu hóa mô hình để triển khai trên các thiết bị di động hoặc tiện ích trình duyệt cũng là một hướng đi tiềm năng nhằm mang lại khả năng bảo vệ người dùng theo thời gian thực.