

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM KHOA CÔNG NGHỆ THÔNG TIN**

\*\*\*\*\*\*\*\*

**BÁO CÁO CUỐI KỲ**

**MÔN HỌC: TRÍ TUỆ NHÂN TẠO CHO IOT**

**BÁO CÁO ỨNG DỤNG AI TẠO SINH TRONG TÁI TẠO TÍN HIỆU PPG: ĐÁNH GIÁ MÔ HÌNH CVAE, VAE, CGAN VÀ GAN TRÊN DỮ LIỆU Y TẾ**

**GVHD: ThS. Hồ Nhựt Minh**

**Sinh viên thực hiện:**

**Hồng Anh Khoa - 22110351**

**Tp. Hồ Chí Minh, ngày 18 tháng 5 năm 2025**

**DANH MỤC HÌNH ẢNH VÀ BẢNG THỐNG KÊ**

**A. Danh mục hình ảnh:**

* A4.1. Model CVAE: Kiến trúc mô hình CVAE
* A4.2. Model VAE: Kiến trúc mô hình VAE
* A4.3. Model CGAN: Kiến trúc mô hình CGAN.
* A4.4. Model GAN: Kiến trúc mô hình GAN.
* A5.1. Process CVAE and VAE: Quy trình huấn luyện cho mô hình CVAE và VAE.
* A5.2. Process CGAN and GAN: Quy trình huấn luyện cho mô hình CGAN và GAN.
* A6.1. Compare CVAE – BIDMC: Biểu đồ so sánh tín hiệu thật và tín hiệu tạo sinh trên tập dữ liệu BIDMC với mô hình CVAE.
* A6.2. Compare CVAE – CAPNO: Biểu đồ so sánh tín hiệu thật và tín hiệu tạo sinh trên tập dữ liệu Capno với mô hình CVAE.
* A6.3. Compare CVAE - COMBINED: Biểu đồ so sánh tín hiệu thật và tín hiệu tạo sinh trên tập dữ liệu Kết hợp với mô hình CVAE.
* A6.4. Compare VAE - BIDMC: Biểu đồ so sánh tín hiệu thật và tín hiệu tạo sinh trên tập dữ liệu BIDMC với mô hình VAE.
* A6.5. Compare VAE - CAPNO: Biểu đồ so sánh tín hiệu thật và tín hiệu tạo sinh trên tập dữ liệu Capno với mô hình VAE.
* A6.6. Compare VAE - COMBINED: Biểu đồ so sánh tín hiệu thật và tín hiệu tạo sinh trên tập dữ liệu Kết hợp với mô hình VAE.
* A6.7. Compare CGAN - BIDMC: Biểu đồ so sánh tín hiệu thật và tín hiệu tạo sinh trên tập dữ liệu BIDMC với mô hình CGAN.
* A6.8. Compare CGAN - CAPNO: Biểu đồ so sánh tín hiệu thật và tín hiệu tạo sinh trên tập dữ liệu Capno với mô hình CGAN.
* A6.9. Compare CGAN - COMBINED: Biểu đồ so sánh tín hiệu thật và tín hiệu tạo sinh trên tập dữ liệu Kết hợp với mô hình CGAN.
* A6.10. Compare GAN - BIDMC: Biểu đồ so sánh tín hiệu thật và tín hiệu tạo sinh trên tập dữ liệu BIDMC với mô hình GAN.
* A6.11. Compare GAN - CAPNO: Biểu đồ so sánh tín hiệu thật và tín hiệu tạo sinh trên tập dữ liệu Capno với mô hình GAN.
* A6.12. Compare GAN - COMBINED: Biểu đồ so sánh tín hiệu thật và tín hiệu tạo sinh trên tập dữ liệu Kết hợp với mô hình GAN.

**B. Bảng thống kê**

* B5. Result comparison table: Bảng tổng hợp kết quả huấn luyện.
* B6. Performance evaluation table: Bảng đánh giá hiệu suất mô hình.

**MỤC LỤC**

[**1. Giới thiệu** 1](#_Toc198484610)

[**1.1. Tính cấp thiết của đề tài** 1](#_Toc198484611)

[**1.2. Mục đích nghiên cứu** 1](#_Toc198484612)

[**1.3. Phương pháp nghiên cứu** 2](#_Toc198484613)

[**2. Mô tả dữ liệu và quy trình:** 2](#_Toc198484614)

[**2.1. Mô tả dữ liệu:** 2](#_Toc198484615)

[**2.2. Quy trình chung** 3](#_Toc198484616)

[**3. Đọc và tiền xử lý dữ liệu** 4](#_Toc198484617)

[**3.1. Bộ dữ liệu BIDMC:** 4](#_Toc198484618)

[**3.2. Bộ dữ liệu Capnobase:** 6](#_Toc198484619)

[**3.3. Bộ dữ liệu kết hợp:** 10](#_Toc198484620)

[**4. Xây dựng mô hình** 14](#_Toc198484621)

[**4.1. Kiến trúc mô hình CVAE và VAE** 14](#_Toc198484622)

[**4.2. Kiến trúc mô hình CGAN và GAN** 16](#_Toc198484623)

[**5. Quy trình huấn luyện:** 18](#_Toc198484624)

[**5.1. Mô hình CVAE** 18](#_Toc198484625)

[**5.2. Mô hình VAE** 20](#_Toc198484626)

[**5.3. Mô hình cGAN** 22](#_Toc198484627)

[**5.4. Mô hình GAN** 25](#_Toc198484628)

[**5.5. So sánh tổng quan** 27](#_Toc198484629)

[**6. Trực quan hóa và đánh giá** 30](#_Toc198484630)

[**6.1. Tổng quan về tín hiệu PPG** 30](#_Toc198484631)

[**6.2. Đánh giá từng mô hình** 31](#_Toc198484632)

[**6.3. Bảng đánh giá và so sánh hiệu suất trực quan** 33](#_Toc198484633)

[**7. Tính ứng dụng và mở rộng của dự án** 34](#_Toc198484634)

[**Tài liệu tham khảo** 38](#_Toc198484635)

# **1. Giới thiệu**

## **1.1. Tính cấp thiết của đề tài**

Tín hiệu sinh học, như tín hiệu nhịp tim (PPG) và hô hấp, là nền tảng quan trọng trong giám sát sức khỏe và chẩn đoán y tế, đặc biệt trong bối cảnh các thiết bị đeo thông minh ngày càng phổ biến. Tuy nhiên, việc thu thập dữ liệu thực tế thường gặp hạn chế do chi phí cao, quyền riêng tư của bệnh nhân, và sự thiếu hụt trong các tình huống hiếm gặp. Các mô hình AI tạo sinh, chẳng hạn như VAE, GAN, và các biến thể có điều kiện (cVAE, cGAN), đã nổi lên như một giải pháp tiềm năng để tái tạo và mở rộng dữ liệu này, hỗ trợ nghiên cứu y sinh và phát triển hệ thống giám sát thời gian thực. Đề tài này đặc biệt cấp thiết khi cần so sánh hiệu suất của các mô hình trên các tập dữ liệu đa dạng (BIDMC, Capnobase, và kết hợp), nhằm tìm ra phương pháp tối ưu cho việc tạo sinh tín hiệu PPG chất lượng cao, góp phần thúc đẩy ứng dụng AI trong y tế.

## **1.2. Mục đích nghiên cứu**

Mục đích của dự án là phát triển và đánh giá bốn mô hình tạo sinh (cVAE, VAE, cGAN, GAN) trên ba tập dữ liệu tín hiệu sinh học: BIDMC (chứa tín hiệu PPG và nhịp tim), Capnobase (chứa tín hiệu hô hấp), và một tập dữ liệu kết hợp từ cả hai. Nghiên cứu hướng đến:

* Xây dựng và huấn luyện tổng cộng 12 mô hình (4 loại mô hình × 3 bộ dữ liệu).
* Đánh giá chất lượng tín hiệu tạo sinh thông qua phân tích trực quan và so sánh với tín hiệu thật.
* Phân tích ưu, nhược điểm của từng mô hình để xác định mô hình hiệu quả nhất trên từng tập dữ liệu.
* Đánh giá tiềm năng của bộ dữ liệu kết hợp trong việc cải thiện hiệu suất tổng quát của các mô hình.

## **1.3. Phương pháp nghiên cứu**

Phương pháp nghiên cứu bao gồm các bước sau:

* Thu thập và tiền xử lý dữ liệu từ ba tập dữ liệu BIDMC, Capnobase, và bộ dữ liệu kết hợp, đảm bảo tính đồng nhất và chất lượng đầu vào.
* Xây dựng và huấn luyện 12 mô hình tạo sinh (cVAE, VAE, cGAN, GAN) trên từng tập dữ liệu, sử dụng các kỹ thuật học sâu hiện đại.
* Đánh giá hiệu suất thông qua phân tích trực quan bằng cách so sánh tín hiệu PPG thật và tín hiệu tạo sinh, tập trung vào nhịp điệu, biên độ dao động, và chi tiết tín hiệu.
* Tổng hợp kết quả dưới dạng bảng so sánh và phân tích định tính để rút ra nhận định về hiệu quả của từng mô hình trên các tập dữ liệu khác nhau.

# **2. Mô tả dữ liệu và quy trình:**

## **2.1. Mô tả dữ liệu:**

Dự án sử dụng ba bộ dữ liệu tín hiệu sinh học:

* **Tập dữ liệu BIDMC** (Beth Israel Deaconess Medical Center): là một tập hợp tín hiệu sinh lý công khai, được trích xuất từ nguồn ***MIMIC-II*** của Bệnh viện Beth Israel Deaconess, Boston, Hoa Kỳ. Tập dữ liệu này bao gồm các bản ghi 8 phút của ***điện tâm đồ (ECG)****,* ***tín hiệu quang thể tích (PPG)****, và* ***tín hiệu trở kháng hô hấp*** từ 53 bệnh nhân người lớn (tuổi từ 19 đến trên 90, gồm 32 nữ) tại các đơn vị chăm sóc đặc biệt (ICU), bao gồm ICU y khoa, ICU phẫu thuật, và ICU tim mạch. Các tín hiệu được thu thập với tần số lấy mẫu 125 Hz và được hai nhà nghiên cứu độc lập chú thích thủ công cho từng chu kỳ hô hấp dựa trên tín hiệu trở kháng, đảm bảo độ chính xác của nhịp hô hấp tham chiếu (RR). BIDMC được sử dụng rộng rãi trong nghiên cứu xử lý tín hiệu sinh lý, đặc biệt để phát triển và đánh giá các thuật toán ước lượng nhịp hô hấp từ PPG và ECG, nhờ chất lượng tín hiệu cao và tính đại diện cho bệnh nhân ICU.
* **Tập dữ liệu CapnoBase:** là một cơ sở dữ liệu trực tuyến chứa các tín hiệu hô hấp và sinh lý được thu thập trong các ca phẫu thuật tự chọn và gây mê. Tập dữ liệu bao gồm 42 bản ghi 8 phút từ 29 bệnh nhân nhi và 13 bệnh nhân người lớn, với các tín hiệu như ***capnography (CO2 hô hấp)****,* ***ECG****,* ***PPG****, và* ***tín hiệu lưu lượng hô hấp***, được lấy mẫu ở tần số 300 Hz. Mỗi chu kỳ hô hấp trong tín hiệu capnography được chú thích thủ công bởi một nhà nghiên cứu, cung cấp nhịp hô hấp tham chiếu chính xác. CapnoBase bao gồm hai bộ dữ liệu chuẩn: một để phát hiện sự kiện hô hấp và một để ước lượng nhịp hô hấp từ PPG (IEEE TBME Respiratory Rate Benchmark). Với tính đa dạng về độ tuổi và điều kiện hô hấp (thở tự nhiên và kiểm soát), CapnoBase là nguồn tài nguyên quan trọng để kiểm tra tính mạnh mẽ của các thuật toán xử lý tín hiệu trong các ứng dụng theo dõi sức khỏe không xâm lấn.
* **Kết hợp (Combined)**: Gộp dữ liệu từ BIDMC và Capnobase, tổng cộng 14,946 mẫu (11,956 mẫu huấn luyện, 2,990 mẫu kiểm tra), được chuẩn hóa và lưu trong file combined\_data.mat.

## **2.2. Quy trình chung**

Tất cả các mô hình được xây dựng theo quy trình sau:

1. **Đọc dữ liệu**: Tải dữ liệu từ file .mat bằng thư viện scipy.io.
2. **Tiền xử lý**: Chuẩn hóa tín hiệu và nhãn (nếu có). Chuyển đổi dữ liệu thành định dạng phù hợp (ví dụ: tensor cho TensorFlow).
3. **Xây dựng mô hình**: Sử dụng các thư viện như TensorFlow/Keras để thiết kế kiến trúc mô hình.
4. **Huấn luyện**: Huấn luyện với số epoch, batch size và hàm mất mát được tối ưu hóa.
5. **Trực quan hóa**: Vẽ biểu đồ tín hiệu gốc và tín hiệu tạo sinh, hàm mất mát qua các epoch.

# **3. Đọc và tiền xử lý dữ liệu**

## **3.1. Bộ dữ liệu BIDMC:**

Tiền xử lý dữ liệu BIDMC được thiết kế để chuẩn bị tín hiệu PPG (photoplethysmogram) và nhịp tim (HR) cho các mô hình cVAE, VAE, cGAN và GAN, đảm bảo dữ liệu phù hợp với yêu cầu của từng mô hình. Quy trình bao gồm các bước sau:

1. **Tải và kiểm tra dữ liệu**:
   * Dữ liệu được tải từ file bidmc\_data.mat bằng thư viện scipy.io, chứa 53 bản ghi tín hiệu PPG và thông số nhịp tim tương ứng.
   * Mỗi bản ghi được kiểm tra để đảm bảo trường dữ liệu (v cho PPG và hr cho nhịp tim) tồn tại và không chứa giá trị nan. Hàm extract\_flat\_array được sử dụng để trích xuất và làm phẳng mảng dữ liệu, xử lý các trường hợp mảng lồng nhau hoặc định dạng không nhất quán.
2. **Thống kê và chuẩn hóa nhịp tim (HR)**:
   * Toàn bộ giá trị HR từ 53 bản ghi được thu thập để tính giá trị trung bình (mean), độ lệch chuẩn (std), giá trị tối thiểu (HR\_MIN) và tối đa (HR\_MAX). Các thống kê này được sử dụng để hiểu phân phối HR và hỗ trợ các mô hình có điều kiện (cVAE, cGAN).
   * Ví dụ, HR có phạm vi từ [HR\_MIN] đến [HR\_MAX] BPM, với trung bình [hr\_mean] và độ lệch chuẩn [hr\_std].
3. **Lọc tín hiệu PPG**:
   * Tín hiệu PPG được lọc bằng bộ lọc Butterworth bậc 4 trong dải tần [0.5-8] Hz để loại bỏ nhiễu tần số thấp (như chuyển động) và nhiễu tần số cao (như nhiễu điện). Tần số lấy mẫu là 125 Hz, đảm bảo tín hiệu được xử lý phù hợp với đặc tính sinh lý.
   * Kết quả lọc được trực quan hóa bằng cách vẽ tín hiệu PPG gốc và tín hiệu đã lọc cho mỗi bản ghi, lưu vào thư mục figures (ví dụ: ppg\_filter\_comparison\_record\_{i}.png).
4. **Phân đoạn tín hiệu**:
   * Tín hiệu PPG được chia thành các đoạn dài 750 mẫu (tương đương 6 giây tại tần số 125 Hz) với mức chồng lấp 50% (overlap = 0.5). Điều này đảm bảo đủ mẫu dữ liệu mà không bỏ sót thông tin quan trọng.
   * Mỗi đoạn PPG được chuẩn hóa riêng lẻ về khoảng [-1, 1] bằng MinMaxScaler để đảm bảo biên độ phù hợp với các mô hình học sâu, đặc biệt là GAN và cGAN, vốn nhạy cảm với phạm vi đầu vào.
   * Nhịp tim tương ứng cho mỗi đoạn được tính bằng trung bình các giá trị HR trong khoảng thời gian của đoạn, dựa trên tần số lấy mẫu của HR (hr\_fs). Các đoạn chứa giá trị nan hoặc không đủ dữ liệu HR bị loại bỏ.
5. **Chia tập huấn luyện và kiểm tra**:
   * Tổng cộng [len(ppg\_segments)] đoạn tín hiệu được tạo ra từ [valid\_records] bản ghi hợp lệ. Dữ liệu được chia ngẫu nhiên thành tập huấn luyện (80%) và tập kiểm tra (20%) bằng cách sử dụng hoán vị ngẫu nhiên (np.random.permutation).
   * Kết quả: tập huấn luyện có [X\_train.shape[0]] mẫu, tập kiểm tra có [X\_test.shape[0]] mẫu, mỗi mẫu có kích thước (750,).
6. **Đặc biệt cho từng mô hình**:
   * **cVAE**: Tín hiệu PPG và nhịp tim được giữ nguyên dạng chuẩn hóa [-1, 1] cho PPG và giá trị thô cho HR. HR được sử dụng làm nhãn điều kiện, nhập vào cả bộ mã hóa và giải mã. Dữ liệu được lưu dưới dạng X\_train\_norm, X\_test\_norm, hr\_raw\_train, hr\_raw\_test để dễ dàng tích hợp vào mô hình.
   * **VAE**: Chỉ sử dụng tín hiệu PPG (bỏ qua HR), giữ nguyên chuẩn hóa [-1, 1]. Dữ liệu được xử lý tương tự nhưng không yêu cầu nhãn điều kiện, giảm độ phức tạp trong pipeline.
   * **cGAN**: Tương tự cVAE, sử dụng cả PPG và HR. Chuẩn hóa [-1, 1] đặc biệt quan trọng để đảm bảo Generator và Discriminator hoạt động ổn định. HR được mã hóa thành vector điều kiện và nối vào đầu vào của Generator và Discriminator.
   * **GAN**: Chỉ sử dụng PPG, tương tự VAE, với chuẩn hóa [-1, 1]. Không yêu cầu xử lý nhãn HR, giúp đơn giản hóa quy trình nhưng có thể làm giảm khả năng kiểm soát đặc trưng tín hiệu.
7. **Lưu trữ và trực quan hóa**:
   * Dữ liệu đã xử lý được lưu vào file processed\_bidmc\_data.mat với các trường: X\_train\_norm, X\_test\_norm, hr\_raw\_train, hr\_raw\_test, X\_total\_norm, hr\_raw\_total.
   * Thông tin tiền xử lý được ghi vào file preprocessing\_info.txt, bao gồm số bản ghi hợp lệ, số đoạn tín hiệu, tham số (tần số lấy mẫu, độ dài đoạn, mức chồng lấp), kích thước tập dữ liệu, và thống kê HR.
   * Phân phối HR được vẽ dưới dạng histogram (hr\_distribution.png), cho thấy sự phân bố của nhịp tim trong khoảng [HR\_MIN, HR\_MAX]. Năm đoạn PPG mẫu được vẽ (preprocessed\_ppg\_segments.png) để kiểm tra chất lượng tín hiệu sau xử lý.
8. **Kiểm tra và xử lý lỗi**:
   * Các bản ghi không hợp lệ (chứa nan hoặc không đủ dữ liệu HR) bị bỏ qua, đảm bảo chỉ dữ liệu chất lượng cao được sử dụng.
   * Trong trường hợp không có dữ liệu hợp lệ, code tạo dữ liệu giả lập (100 mẫu ngẫu nhiên) để minh họa, nhưng điều này không xảy ra với BIDMC do dữ liệu thực tế đủ lớn.

## **3.2. Bộ dữ liệu Capnobase:**

Quy trình tiền xử lý dữ liệu Capnobase được thiết kế để chuẩn bị tín hiệu PPG (photoplethysmogram) và nhịp tim (HR) từ bộ dữ liệu Capnobase cho các mô hình cVAE, VAE, cGAN và GAN. Tiền xử lý đảm bảo tín hiệu được đồng bộ về tần số lấy mẫu, loại bỏ nhiễu, chuẩn hóa phù hợp và sẵn sàng cho huấn luyện. Dưới đây là mô tả chi tiết các bước và cách áp dụng cho từng mô hình:

1. **Tải và kiểm tra dữ liệu**:
   * Dữ liệu được tải từ các file .mat trong thư mục Capnobase bằng thư viện h5py, với mỗi file chứa tín hiệu PPG (signal/pleth/y), nhịp tim từ ECG (reference/hr/ecg/y), và tần số lấy mẫu (param/samplingrate/pleth). Hàm load\_mat\_file kiểm tra tính toàn vẹn của dữ liệu, đảm bảo các trường cần thiết tồn tại và xử lý lỗi nếu dữ liệu không đầy đủ.
   * Mỗi file được in thông tin (khóa dữ liệu, độ dài tín hiệu PPG, HR, tần số lấy mẫu) để debug. Tổng cộng [len(all\_data)] file được đọc thành công, mỗi file đại diện cho một bản ghi bệnh nhân.
2. **Downsample tín hiệu**:
   * Tín hiệu PPG có tần số lấy mẫu gốc là 300 Hz, được downsample xuống 125 Hz bằng phương pháp Fourier (scipy.signal.resample) để đồng bộ với tần số của bộ dữ liệu BIDMC và giảm khối lượng tính toán. Hàm downsample\_signal đảm bảo tín hiệu giữ được các đặc trưng quan trọng sau khi giảm tần số.
   * Tần số lấy mẫu sau downsample được cập nhật thành 125 Hz, đảm bảo tính nhất quán cho tất cả các mẫu dữ liệu.
3. **Lọc tín hiệu PPG**:
   * Tín hiệu PPG được lọc bằng bộ lọc Butterworth bậc 4 trong dải tần [0.5-8] Hz để loại bỏ nhiễu tần số thấp (như chuyển động cơ thể) và nhiễu tần số cao (như nhiễu điện). Hàm filter\_signal áp dụng lọc băng thông (filtfilt) để đảm bảo tín hiệu sạch mà không làm méo dạng sóng.
   * Tín hiệu sau khi lọc giữ được các đặc trưng sinh lý của PPG, phù hợp cho các mô hình học sâu.
4. **Phân đoạn tín hiệu**:
   * Tín hiệu PPG được chia thành các đoạn dài 750 mẫu (tương đương 6 giây tại 125 Hz) với mức chồng lấp 50% (overlap=0.5). Hàm segment\_all\_signals tính số đoạn dựa trên độ dài tín hiệu và bước di chuyển (step = segment\_length \* (1 - overlap)).
   * Mỗi đoạn PPG được chuẩn hóa riêng lẻ về khoảng [-1, 1] bằng MinMaxScaler để đảm bảo biên độ phù hợp, đặc biệt quan trọng cho GAN và cGAN do tính nhạy cảm với phạm vi đầu vào.
   * Nhịp tim (HR) tương ứng cho mỗi đoạn được tính bằng trung bình các giá trị HR trong khoảng thời gian của đoạn, dựa trên tỷ lệ độ dài giữa tín hiệu PPG và HR. Nếu không có giá trị HR trong khoảng, giá trị HR gần nhất được sử dụng để tránh dữ liệu trống.
   * Kết quả: [len(all\_ppg\_segments)] đoạn tín hiệu được tạo ra, với PPG chuẩn hóa (all\_ppg\_normalized) và HR thô (all\_hr\_raw).
5. **Thống kê và trực quan hóa HR**:
   * Phân phối nhịp tim được tính toán, với giá trị trung bình [hr\_mean], độ lệch chuẩn [hr\_std], tối thiểu [hr\_min], và tối đa [hr\_max]. Biểu đồ histogram được vẽ để hiển thị phân bố HR, giúp đánh giá tính đa dạng của nhịp tim trong dữ liệu.
   * Năm đoạn PPG mẫu được vẽ cùng với giá trị HR tương ứng, lưu vào file để kiểm tra chất lượng tín hiệu sau xử lý.
6. **Chia tập huấn luyện và kiểm tra**:
   * Dữ liệu được chia ngẫu nhiên thành tập huấn luyện (80%) và tập kiểm tra (20%) bằng hoán vị ngẫu nhiên (np.random.permutation). Hàm prepare\_data\_loaders tạo các DataLoader cho huấn luyện và kiểm tra, với batch size 64 và dữ liệu được chuyển thành tensor (torch.FloatTensor).
   * Kết quả: tập huấn luyện có [X\_train.shape[0]] mẫu, tập kiểm tra có [X\_test.shape[0]] mẫu, mỗi mẫu có kích thước (750,). HR được định dạng thành vector cột để phù hợp với các mô hình có điều kiện.
7. **Đặc biệt cho từng mô hình**:
   * **cVAE**: Sử dụng cả PPG chuẩn hóa [-1, 1] và HR thô làm nhãn điều kiện. Dữ liệu được tổ chức thành X\_train\_norm, X\_test\_norm, hr\_raw\_train, hr\_raw\_test, với HR được nhập vào bộ mã hóa và giải mã. DataLoader đảm bảo dữ liệu được cung cấp theo cặp (PPG, HR) cho huấn luyện.
   * **VAE**: Chỉ sử dụng PPG chuẩn hóa [-1, 1], bỏ qua HR. Quy trình đơn giản hơn do không cần xử lý nhãn điều kiện, nhưng vẫn giữ nguyên chuẩn hóa để đảm bảo tính nhất quán với cVAE.
   * **cGAN**: Sử dụng PPG chuẩn hóa [-1, 1] và HR thô. HR được mã hóa thành vector điều kiện, nối vào đầu vào của Generator và Discriminator. Chuẩn hóa [-1, 1] giúp ổn định quá trình huấn luyện đối kháng, đặc biệt khi Discriminator đánh giá tín hiệu giả.
   * **GAN**: Chỉ sử dụng PPG chuẩn hóa [-1, 1], không cần HR. Quy trình tương tự VAE, tập trung vào tín hiệu PPG mà không cần xử lý nhãn, giúp giảm độ phức tạp nhưng có thể hạn chế khả năng tạo tín hiệu có đặc trưng cụ thể.
8. **Lưu trữ dữ liệu**:
   * Dữ liệu đã xử lý được lưu vào file processed\_capno\_data.mat với các trường: X\_train\_norm, X\_test\_norm, hr\_raw\_train, hr\_raw\_test, X\_total\_norm, hr\_raw\_total. Điều này cho phép tái sử dụng dữ liệu mà không cần lặp lại tiền xử lý.
   * Thông tin về kích thước dữ liệu được in ra, bao gồm: X\_train ([X\_train.shape]), X\_test ([X\_test.shape]), và dữ liệu chuẩn hóa tương ứng.
9. **Kiểm tra và xử lý lỗi**:
   * Các file không hợp lệ (thiếu PPG hoặc HR) bị bỏ qua, đảm bảo chỉ dữ liệu chất lượng cao được sử dụng. Hàm load\_mat\_file xử lý các lỗi như dữ liệu không đầy đủ hoặc định dạng không đúng.
   * Dữ liệu được kiểm tra kỹ lưỡng qua các bước in thông tin (độ dài tín hiệu, tần số lấy mẫu) và trực quan hóa để xác nhận chất lượng.

## **3.3. Bộ dữ liệu kết hợp:**

Quy trình tiền xử lý bộ dữ liệu Combined nhằm kết hợp dữ liệu đã được xử lý từ hai bộ BIDMC và Capnobase, tạo ra một tập dữ liệu thống nhất để huấn luyện và đánh giá các mô hình cVAE, VAE, cGAN và GAN. Mục tiêu là tận dụng sự đa dạng của cả hai bộ dữ liệu để cải thiện khả năng tổng quát hóa của các mô hình. Dưới đây là mô tả chi tiết các bước và cách áp dụng cho từng mô hình:

1. **Tải dữ liệu đã xử lý**:
   * Dữ liệu được tải từ hai file processed\_capno\_data.mat (Capnobase) và processed\_bidmc\_data.mat (BIDMC) bằng thư viện scipy.io. Mỗi file chứa các trường dữ liệu đã được tiền xử lý trước đó, bao gồm:
     + X\_train\_norm, X\_test\_norm: Tín hiệu PPG chuẩn hóa về khoảng [-1, 1].
     + hr\_raw\_train, hr\_raw\_test: Giá trị nhịp tim (HR) thô.
     + X\_total\_norm, hr\_raw\_total: Toàn bộ dữ liệu PPG và HR (gộp train và test).
   * Các trường dữ liệu được trích xuất và làm phẳng (.flatten()) để đảm bảo định dạng nhất quán, chuyển từ mảng 2D (1, n) thành mảng 1D (n,).
2. **Kết hợp dữ liệu**:
   * Tín hiệu PPG và nhịp tim từ hai bộ dữ liệu được gộp lại bằng cách sử dụng np.vstack (cho PPG) và np.hstack (cho HR):
     + **Tập huấn luyện**: Gộp capno\_X\_train\_norm và bidmc\_X\_train\_norm thành combined\_X\_train\_norm; gộp capno\_hr\_raw\_train và bidmc\_hr\_raw\_train thành combined\_hr\_raw\_train.
     + **Tập kiểm tra**: Tương tự, gộp capno\_X\_test\_norm và bidmc\_X\_test\_norm thành combined\_X\_test\_norm; gộp capno\_hr\_raw\_test và bidmc\_hr\_raw\_test thành combined\_hr\_raw\_test.
     + **Tập tổng**: Gộp capno\_X\_total\_norm và bidmc\_X\_total\_norm thành combined\_X\_total\_norm; gộp capno\_hr\_raw\_total và bidmc\_hr\_raw\_total thành combined\_hr\_raw\_total.
   * Kết quả: Bộ dữ liệu Combined có tổng cộng [combined\_X\_total\_norm.shape[0]] mẫu, với [combined\_X\_train\_norm.shape[0]] mẫu huấn luyện và [combined\_X\_test\_norm.shape[0]] mẫu kiểm tra, mỗi mẫu PPG có độ dài 750 điểm và nhịp tim tương ứng.
3. **Kiểm tra và đảm bảo tính nhất quán**:
   * Kích thước của các tập dữ liệu được in ra để xác nhận rằng quá trình gộp không làm mất dữ liệu hoặc gây lỗi định dạng. Ví dụ:
     + combined\_X\_train\_norm: ([combined\_X\_train\_norm.shape[0]], 750)
     + combined\_X\_test\_norm: ([combined\_X\_test\_norm.shape[0]], 750)
     + combined\_hr\_raw\_total: ([combined\_hr\_raw\_total.shape[0]],)
   * Dữ liệu PPG đã được chuẩn hóa về [-1, 1] từ các bước tiền xử lý trước đó (BIDMC và Capnobase), đảm bảo tính nhất quán về biên độ. Nhịp tim giữ nguyên giá trị thô, phù hợp cho các mô hình có điều kiện.
4. **Lưu trữ dữ liệu**:
   * Dữ liệu Combined được lưu vào file combined\_data.mat với các trường: X\_train\_norm, X\_test\_norm, hr\_raw\_train, hr\_raw\_test, X\_total\_norm, hr\_raw\_total. File này cho phép tái sử dụng dữ liệu mà không cần lặp lại quá trình gộp.
   * Việc lưu trữ đảm bảo tất cả dữ liệu được tổ chức gọn gàng, dễ dàng truy cập cho các bước huấn luyện và đánh giá.
5. **Trực quan hóa dữ liệu**:
   * Để kiểm tra chất lượng dữ liệu Combined, năm mẫu ngẫu nhiên từ tập huấn luyện và bốn mẫu từ tập kiểm tra được vẽ bằng matplotlib. Mỗi biểu đồ hiển thị tín hiệu PPG (chuẩn hóa [-1, 1]) cùng giá trị nhịp tim tương ứng (ví dụ: “Mẫu train [idx], HR=[hr\_raw\_train\_loaded[idx]]”).
   * Các biểu đồ này giúp xác nhận rằng tín hiệu PPG từ cả BIDMC và Capnobase được gộp đúng cách, giữ được đặc trưng sinh lý và nhịp tim hợp lý.
6. **Đặc biệt cho từng mô hình**:
   * **CVAE**: Sử dụng cả PPG chuẩn hóa [-1, 1] và HR thô làm nhãn điều kiện. Dữ liệu Combined cung cấp sự đa dạng lớn hơn về tín hiệu PPG và nhịp tim, giúp mô hình học được các đặc trưng tổng quát hơn. HR được nhập vào bộ mã hóa và giải mã, tận dụng phân phối nhịp tim rộng từ cả hai bộ dữ liệu.
   * **VAE**: Chỉ sử dụng PPG chuẩn hóa [-1, 1], bỏ qua HR. Dữ liệu Combined tăng số lượng mẫu huấn luyện ([combined\_X\_train\_norm.shape[0]]), giúp cải thiện khả năng học các đặc trưng tiềm ẩn mà không cần nhãn điều kiện.
   * **CGAN**: Sử dụng PPG chuẩn hóa [-1, 1] và HR thô. HR được mã hóa thành vector điều kiện, nối vào đầu vào của Generator và Discriminator. Sự đa dạng của dữ liệu Combined giúp Generator tạo ra tín hiệu PPG phong phú hơn, trong khi Discriminator được huấn luyện trên tập dữ liệu lớn hơn để phân biệt tín hiệu thật/giả chính xác hơn.
   * **GAN**: Chỉ sử dụng PPG chuẩn hóa [-1, 1], không cần HR. Dữ liệu Combined cung cấp nhiều mẫu hơn ([combined\_X\_total\_norm.shape[0]]), hỗ trợ huấn luyện ổn định hơn nhưng có thể kém kiểm soát đặc trưng so với cGAN do thiếu nhãn điều kiện.
7. **Xử lý lỗi và kiểm tra**:
   * Quá trình gộp được thực hiện trên dữ liệu đã tiền xử lý, giảm thiểu nguy cơ lỗi như giá trị nan hoặc định dạng không nhất quán. Các bước kiểm tra kích thước (print shape) đảm bảo dữ liệu được gộp chính xác.
   * Nếu có lỗi (ví dụ: kích thước không khớp), chúng đã được xử lý ở giai đoạn tiền xử lý BIDMC và Capnobase, đảm bảo dữ liệu đầu vào cho Combined là sạch.

# **4. Xây dựng mô hình**

## **4.1. Kiến trúc mô hình CVAE và VAE**

A4.1.Model CVAE

A4.2.Model VAE

|  |
| --- |
| **=== Cấu trúc CVAE ===**  CVAEFixed(  (encoder): EncoderFixed(  (input\_layer): Linear(in\_features=751, out\_features=256, bias=True)  (hidden\_layers): ModuleList(  (0): Linear(in\_features=256, out\_features=128, bias=True)  (1): Linear(in\_features=128, out\_features=64, bias=True)  )  (z\_mean): Linear(in\_features=64, out\_features=64, bias=True)  (z\_log\_var): Linear(in\_features=64, out\_features=64, bias=True)  (sampling): Sampling()  )  (decoder): DecoderFixed(  (hidden\_layers): ModuleList(  (0): Linear(in\_features=64, out\_features=128, bias=True)  (1): Linear(in\_features=128, out\_features=256, bias=True)  )  (input\_layer): Linear(in\_features=65, out\_features=64, bias=True)  (output\_layer): Linear(in\_features=256, out\_features=750, bias=True)  )  ) |
| **=== Cấu trúc VAE ===**  VAE(  (encoder): Encoder(  (input\_layer): Linear(in\_features=750, out\_features=256, bias=True)  (hidden\_layers): ModuleList(  (0): Linear(in\_features=256, out\_features=128, bias=True)  (1): Linear(in\_features=128, out\_features=64, bias=True)  )  (z\_mean): Linear(in\_features=64, out\_features=64, bias=True)  (z\_log\_var): Linear(in\_features=64, out\_features=64, bias=True)  (sampling): Sampling()  )  (decoder): Decoder(  (input\_layer): Linear(in\_features=64, out\_features=64, bias=True)  (hidden\_layers): ModuleList(  (0): Linear(in\_features=64, out\_features=128, bias=True)  (1): Linear(in\_features=128, out\_features=256, bias=True)  )  (output\_layer): Linear(in\_features=256, out\_features=750, bias=True)  )  ) |

## **4.2. Kiến trúc mô hình CGAN và GAN**

A4.4.Model GAN

A4.3.Model CGAN

|  |
| --- |
| **=== Cấu trúc CGAN ===**  CGAN(  (generator): Generator(  (input\_layer): Linear(in\_features=129, out\_features=512, bias=True)  (conv\_layers): Sequential(  (0): ConvTranspose1d(128, 64, kernel\_size=(4,), stride=(2,), padding=(1,))  (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)  (2): LeakyReLU(negative\_slope=0.2)  (3): ConvTranspose1d(64, 32, kernel\_size=(4,), stride=(2,), padding=(1,))  (4): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)  (5): LeakyReLU(negative\_slope=0.2)  (6): ConvTranspose1d(32, 16, kernel\_size=(4,), stride=(2,), padding=(1,))  (7): BatchNorm1d(16, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)  (8): LeakyReLU(negative\_slope=0.2)  (9): ConvTranspose1d(16, 1, kernel\_size=(39,), stride=(23,), padding=(1,))  )  (output\_layer): Tanh()  )  (discriminator): Discriminator(  (condition\_embed): Linear(in\_features=1, out\_features=16, bias=True)  (conv\_layers): Sequential(  (0): Conv1d(17, 32, kernel\_size=(4,), stride=(2,), padding=(1,))  (1): LeakyReLU(negative\_slope=0.2)  (2): Dropout(p=0.3, inplace=False)  (3): Conv1d(32, 64, kernel\_size=(4,), stride=(2,), padding=(1,))  (4): LeakyReLU(negative\_slope=0.2)  (5): Dropout(p=0.3, inplace=False)  (6): Conv1d(64, 128, kernel\_size=(4,), stride=(2,), padding=(1,))  (7): LeakyReLU(negative\_slope=0.2)  (8): AdaptiveAvgPool1d(output\_size=93)  )  (output\_layer): Linear(in\_features=11904, out\_features=1, bias=True)  )  ) |
| **=== GAN Architecture ===**  GAN(  (generator): Generator(  (input\_layer): Linear(in\_features=128, out\_features=512, bias=True)  (conv\_layers): Sequential(  (0): ConvTranspose1d(128, 64, kernel\_size=(4,), stride=(2,), padding=(1,))  (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)  (2): LeakyReLU(negative\_slope=0.2)  (3): ConvTranspose1d(64, 32, kernel\_size=(4,), stride=(2,), padding=(1,))  (4): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)  (5): LeakyReLU(negative\_slope=0.2)  (6): ConvTranspose1d(32, 16, kernel\_size=(4,), stride=(2,), padding=(1,))  (7): BatchNorm1d(16, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)  (8): LeakyReLU(negative\_slope=0.2)  (9): ConvTranspose1d(16, 1, kernel\_size=(39,), stride=(23,), padding=(1,))  )  (output\_layer): Tanh()  )  (discriminator): Discriminator(  (conv\_layers): Sequential(  (0): Conv1d(1, 32, kernel\_size=(4,), stride=(2,), padding=(1,))  (1): LeakyReLU(negative\_slope=0.2)  (2): Dropout(p=0.3, inplace=False)  (3): Conv1d(32, 64, kernel\_size=(4,), stride=(2,), padding=(1,))  (4): LeakyReLU(negative\_slope=0.2)  (5): Dropout(p=0.3, inplace=False)  (6): Conv1d(64, 128, kernel\_size=(4,), stride=(2,), padding=(1,))  (7): LeakyReLU(negative\_slope=0.2)  (8): AdaptiveAvgPool1d(output\_size=93)  )  (output\_layer): Linear(in\_features=11904, out\_features=1, bias=True)  )  ) |

# **5. Quy trình huấn luyện:**

## **5.1. Tổng quát quy trình:**

|  |  |
| --- | --- |
| **QUY TRÌNH HUẤN LUYỆN CVAE và VAE** | **QUY TRÌNH HUẤN LUYỆN CGAN VÀ GAN** |
|  |  |
| A5.1.Process CVAE and VAE | A5.2.Process CGAN and GAN |

## **5.2. Mô hình CVAE**

1. **Thiết lập tham số**:
   * Kích thước không gian tiềm ẩn: latent\_dim = 64.
   * Kích thước các lớp ẩn: [256, 128, 64].
   * Tốc độ học: learning\_rate = 1e-3.
   * Kích thước batch: batch\_size = 64.
   * Số epoch huấn luyện: epochs = 600.
   * Thiết bị tính toán: GPU (CUDA) nếu khả dụng, nếu không sử dụng CPU.
2. **Khởi tạo mô hình và tối ưu hóa**:
   * Mô hình cVAE được chuyển sang thiết bị tính toán (GPU/CPU).
   * Sử dụng bộ tối ưu hóa **Adam** với tốc độ học ban đầu là 1e-3.
   * Áp dụng bộ lập lịch tốc độ học **ReduceLROnPlateau** với hệ số giảm factor = 0.5, ngưỡng kiên nhẫn patience = 20, dựa trên giá trị mất mát kiểm tra (test loss).
3. **Hàm mất mát**:
   * Hàm mất mát tổng hợp bao gồm:
     + **Mất mát tái tạo (Reconstruction Loss)**: Đo lường sai lệch giữa tín hiệu PPG đầu vào và tín hiệu tái tạo, sử dụng Mean Squared Error (MSE).
     + **Mất mát KL-Divergence**: Đảm bảo phân phối không gian tiềm ẩn gần với phân phối chuẩn Gaussian, thúc đẩy tính chính quy hóa.
   * Tổng mất mát được tính bằng cách kết hợp hai thành phần trên: loss = recon\_loss + kl\_loss.
4. **Quy trình huấn luyện**:
   * **Giai đoạn huấn luyện**:
     + Mô hình được đặt ở chế độ huấn luyện (cvae.train()).
     + Với mỗi batch dữ liệu (data, condition) từ train\_loader:
       - Dữ liệu PPG (batch, 750) và điều kiện HR (batch, 1) được chuyển sang thiết bị tính toán.
       - Encoder tạo ra z\_mean, z\_log\_var, và mẫu tiềm ẩn z từ dữ liệu và HR.
       - Decoder tái tạo tín hiệu PPG từ z và HR.
       - Tính mất mát, thực hiện lan truyền ngược (backward), và cập nhật tham số mô hình bằng bộ tối ưu hóa.
     + Tổng hợp mất mát huấn luyện (train loss), mất mát tái tạo, và mất mát KL cho mỗi epoch.
   * **Giai đoạn đánh giá**:
     + Mô hình chuyển sang chế độ đánh giá (cvae.eval()).
     + Với mỗi batch từ test\_loader, tính mất mát kiểm tra (test loss) mà không cập nhật tham số (vô hiệu hóa gradient).
     + Tổng hợp mất mát kiểm tra, mất mát tái tạo, và mất mát KL.
5. **Cơ chế dừng sớm và lưu trữ**:
   * **Early stopping**: Nếu mất mát kiểm tra không cải thiện sau patience = 50 epoch, quá trình huấn luyện dừng sớm để tránh quá khớp.
   * **Lưu checkpoint**: Cứ sau 50 epoch, trạng thái mô hình, bộ tối ưu hóa, và lịch sử mất mát được lưu vào thư mục checkpoints với định dạng cvae\_checkpoint\_epoch\_{epoch}.pth.
   * **Lưu mô hình tốt nhất**: Mô hình với mất mát kiểm tra thấp nhất được lưu vào best\_cvae\_combined.pth.
   * **Lưu lịch sử huấn luyện**: Các giá trị mất mát (train, test, reconstruction, KL) được lưu vào training\_history\_combined.pth để phân tích sau này.

## **5.3. Mô hình VAE**

Quá trình huấn luyện mô hình **Variational Autoencoder (VAE)** được thực hiện để tái tạo tín hiệu quang thể tích (PPG) mà không sử dụng điều kiện bổ sung, tập trung vào học biểu diễn tiềm ẩn của dữ liệu. Dưới đây là các bước chi tiết của quy trình huấn luyện:

1. **Thiết lập tham số**:
   * Kích thước đầu vào: input\_size = 750 (tín hiệu PPG).
   * Kích thước không gian tiềm ẩn: latent\_dim = 64.
   * Kích thước các lớp ẩn: [256, 128, 64].
   * Tốc độ học: learning\_rate = 1e-3.
   * Số epoch huấn luyện: epochs = 600.
   * Thiết bị tính toán: GPU (CUDA) nếu khả dụng, nếu không sử dụng CPU.
2. **Khởi tạo mô hình và tối ưu hóa**:
   * Mô hình VAE được chuyển sang thiết bị tính toán (GPU/CPU).
   * Sử dụng bộ tối ưu hóa **Adam** với tốc độ học ban đầu là 1e-3.
   * Áp dụng bộ lập lịch tốc độ học **ReduceLROnPlateau** với hệ số giảm factor = 0.5, ngưỡng kiên nhẫn patience = 20, dựa trên giá trị mất mát kiểm tra (test loss).
3. **Hàm mất mát**:
   * Hàm mất mát tổng hợp bao gồm:
     + **Mất mát tái tạo (Reconstruction Loss)**: Đo lường sai lệch giữa tín hiệu PPG đầu vào và tín hiệu tái tạo, sử dụng Mean Squared Error (MSE).
     + **Mất mát KL-Divergence**: Đảm bảo phân phối không gian tiềm ẩn gần với phân phối chuẩn Gaussian, với trọng số kl\_weight = 0.5 để cân bằng với mất mát tái tạo.
   * Tổng mất mát: loss = recon\_loss + kl\_weight \* kl\_loss.
4. **Quy trình huấn luyện**:
   * **Giai đoạn huấn luyện**:
     + Mô hình được đặt ở chế độ huấn luyện (vae.train()).
     + Với mỗi batch dữ liệu (data,) từ train\_loader:
       - Dữ liệu PPG (batch, 750) được chuyển sang thiết bị tính toán.
       - VAE tạo ra tín hiệu tái tạo x\_recon, z\_mean, và z\_log\_var.
       - Tính mất mát, thực hiện lan truyền ngược (backward), và cập nhật tham số mô hình bằng bộ tối ưu hóa.
     + Tổng hợp mất mát huấn luyện (train loss), mất mát tái tạo, và mất mát KL cho mỗi epoch.
   * **Giai đoạn đánh giá**:
     + Mô hình chuyển sang chế độ đánh giá (vae.eval()).
     + Với mỗi batch từ test\_loader, tính mất mát kiểm tra (test loss) mà không cập nhật tham số (vô hiệu hóa gradient).
     + Tổng hợp mất mát kiểm tra, mất mát tái tạo, và mất mát KL.
5. **Cơ chế dừng sớm và lưu trữ**:
   * **Early stopping**: Nếu mất mát kiểm tra không cải thiện sau patience = 50 epoch, quá trình huấn luyện dừng sớm để tránh quá khớp.
   * **Lưu checkpoint**: Cứ sau 50 epoch, trạng thái mô hình, bộ tối ưu hóa, và lịch sử mất mát được lưu vào thư mục checkpoints với định dạng vae\_checkpoint\_epoch\_{epoch}.pth.
   * **Lưu mô hình tốt nhất**: Mô hình với mất mát kiểm tra thấp nhất được lưu vào best\_vae\_combined.pth.
   * **Lưu lịch sử huấn luyện**: Các giá trị mất mát (train, test, reconstruction, KL) được lưu vào training\_history\_vae.pth để phân tích sau này.

## **5.4. Mô hình cGAN**

Quá trình huấn luyện mô hình **Conditional Generative Adversarial Network (cGAN)** được thực hiện để tạo tín hiệu quang thể tích (PPG) có điều kiện dựa trên nhịp tim (HR), với mục tiêu đối kháng giữa **Generator** và **Discriminator**. Dưới đây là các bước chi tiết của quy trình huấn luyện:

1. **Thiết lập tham số**:
   * Kích thước nhiễu: noise\_dim = 128.
   * Kích thước điều kiện HR: condition\_size = 1.
   * Tốc độ học: learning\_rate = 0.00005.
   * Kích thước batch: batch\_size = 32.
   * Số epoch huấn luyện: epochs = 600.
   * Hệ số beta1 của Adam: beta1 = 0.5.
   * Hằng số phạt gradient: lambda\_gp = 2.
   * Số lần huấn luyện Discriminator mỗi batch: n\_critic = 3.
   * Thiết bị tính toán: GPU (CUDA) nếu khả dụng, nếu không sử dụng CPU.
2. **Khởi tạo mô hình và tối ưu hóa**:
   * Mô hình cGAN (bao gồm Generator và Discriminator) được chuyển sang thiết bị tính toán (GPU/CPU).
   * Sử dụng bộ tối ưu hóa **Adam** riêng biệt cho Generator (optimizer\_G) và Discriminator (optimizer\_D), với tốc độ học 0.00005 và betas=(0.5, 0.999).
   * Áp dụng bộ lập lịch tốc độ học **ReduceLROnPlateau** cho cả Generator và Discriminator, với hệ số giảm factor = 0.5 và ngưỡng kiên nhẫn patience = 20, dựa trên mất mát tương ứng.
3. **Hàm mất mát**:
   * Sử dụng **BCEWithLogitsLoss** để tính mất mát đối kháng cho cả Generator và Discriminator.
   * **Mất mát Discriminator**: Kết hợp mất mát từ dữ liệu thật (d\_loss\_real) và dữ liệu giả (d\_loss\_fake), tính trung bình: d\_loss = (d\_loss\_real + d\_loss\_fake) / 2.
   * **Phạt gradient (Gradient Penalty)**: Được thêm vào mất mát Discriminator mỗi 10 batch với hệ số lambda\_gp = 2, để ổn định huấn luyện theo phương pháp WGAN-GP.
   * **Mất mát Generator**: Được tính dựa trên khả năng lừa Discriminator, sử dụng nhãn thật (real\_label) cho dữ liệu giả.
   * **Label smoothing**: Giá trị nhãn thật giảm tuyến tính từ 0.9 xuống 0.7 theo epoch để cải thiện tính ổn định.
4. **Quy trình huấn luyện**:
   * **Giai đoạn huấn luyện**:
     + Mô hình được đặt ở chế độ huấn luyện (cgan.train()).
     + Với mỗi batch dữ liệu (data, condition) từ train\_loader:
       - Dữ liệu PPG (batch, 750) và điều kiện HR (batch, 1) được chuyển sang thiết bị tính toán.
       - **Huấn luyện Discriminator**: Thực hiện n\_critic = 3 lần mỗi batch:
         * Tính dự đoán cho dữ liệu thật (real\_pred) và giả (fake\_pred).
         * Tính mất mát thật, giả, và phạt gradient, sau đó cập nhật tham số Discriminator.
       - **Huấn luyện Generator**: Tính mất mát dựa trên dự đoán của Discriminator cho dữ liệu giả, sau đó cập nhật tham số Generator.
       - Áp dụng **gradient clipping** (max\_norm=1.0) cho cả Generator và Discriminator để ổn định huấn luyện.
     + Tổng hợp mất mát trung bình của Generator (avg\_g\_loss) và Discriminator (avg\_d\_loss) cho mỗi epoch.
   * **Không có giai đoạn đánh giá riêng**: Do đặc tính đối kháng của GAN, không sử dụng tập kiểm tra để đánh giá trực tiếp.
5. **Cơ chế dừng sớm và lưu trữ**:
   * **Early stopping**: Nếu mất mát Generator không cải thiện sau patience = 30 epoch (sau ít nhất 300 epoch), quá trình huấn luyện dừng sớm để tránh lãng phí tài nguyên.
   * **Lưu checkpoint**: Cứ sau 50 epoch, trạng thái Generator, Discriminator, bộ tối ưu hóa, và lịch sử mất mát được lưu vào thư mục checkpoints với định dạng cgan\_checkpoint\_epoch\_{epoch}.pth.
   * **Lưu mô hình tốt nhất**: Mô hình với mất mát Generator thấp nhất được lưu vào best\_cgan\_combined.pth.
   * **Lưu lịch sử huấn luyện**: Các giá trị mất mát (g\_losses, d\_losses, best\_epoch) được lưu vào training\_history\_cgan\_combined.pth để phân tích sau này.

## **5.5. Mô hình GAN**

Quá trình huấn luyện mô hình **Generative Adversarial Network (GAN)** được thực hiện để tạo tín hiệu quang thể tích (PPG) mà không sử dụng điều kiện bổ sung, với mục tiêu đối kháng giữa **Generator** và **Discriminator**. Dưới đây là các bước chi tiết của quy trình huấn luyện:

1. **Thiết lập tham số**:
   * Kích thước nhiễu: noise\_dim = 128.
   * Kích thước đầu vào PPG: input\_size = 750.
   * Tốc độ học: learning\_rate = 0.00005.
   * Kích thước batch: batch\_size = 32.
   * Số epoch huấn luyện: epochs = 600.
   * Hệ số beta1 của Adam: beta1 = 0.5.
   * Hằng số phạt gradient: lambda\_gp = 2.
   * Số lần huấn luyện Discriminator mỗi batch: n\_critic = 3.
   * Thiết bị tính toán: GPU (CUDA) nếu khả dụng, nếu không sử dụng CPU.
2. **Khởi tạo mô hình và tối ưu hóa**:
   * Mô hình GAN (bao gồm Generator và Discriminator) được chuyển sang thiết bị tính toán (GPU/CPU).
   * Sử dụng bộ tối ưu hóa **Adam** riêng biệt cho Generator (optimizer\_G) và Discriminator (optimizer\_D), với tốc độ học 0.00005 và betas=(0.5, 0.999).
   * Áp dụng bộ lập lịch tốc độ học **ReduceLROnPlateau** cho cả Generator và Discriminator, với hệ số giảm factor = 0.5 và ngưỡng kiên nhẫn patience = 20, dựa trên mất mát tương ứng.
3. **Hàm mất mát**:
   * Sử dụng **BCEWithLogitsLoss** để tính mất mát đối kháng cho cả Generator và Discriminator.
   * **Mất mát Discriminator**: Kết hợp mất mát từ dữ liệu thật (d\_loss\_real) và dữ liệu giả (d\_loss\_fake), tính trung bình: d\_loss = (d\_loss\_real + d\_loss\_fake) / 2.
   * **Phạt gradient (Gradient Penalty)**: Được thêm vào mất mát Discriminator mỗi 10 batch với hệ số lambda\_gp = 2, để ổn định huấn luyện theo phương pháp WGAN-GP.
   * **Mất mát Generator**: Được tính dựa trên khả năng lừa Discriminator, sử dụng nhãn thật (real\_label) cho dữ liệu giả.
   * **Label smoothing**: Giá trị nhãn thật giảm tuyến tính từ 0.9 xuống 0.7 theo epoch để cải thiện tính ổn định.
4. **Quy trình huấn luyện**:
   * **Giai đoạn huấn luyện**:
     + Mô hình được đặt ở chế độ huấn luyện (gan.train()).
     + Với mỗi batch dữ liệu (data, \_) từ train\_loader (bỏ qua điều kiện HR):
       - Dữ liệu PPG (batch, 750) được chuyển sang thiết bị tính toán.
       - **Huấn luyện Discriminator**: Thực hiện n\_critic = 3 lần mỗi batch:
         * Tính dự đoán cho dữ liệu thật (real\_pred) và giả (fake\_pred).
         * Tính mất mát thật, giả, và phạt gradient, sau đó cập nhật tham số Discriminator.
       - **Huấn luyện Generator**: Tính mất mát dựa trên dự đoán của Discriminator cho dữ liệu giả, sau đó cập nhật tham số Generator.
       - Áp dụng **gradient clipping** (max\_norm=1.0) cho cả Generator và Discriminator để ổn định huấn luyện.
     + Tổng hợp mất mát trung bình của Generator (avg\_g\_loss) và Discriminator (avg\_d\_loss) cho mỗi epoch.
   * **Không có giai đoạn đánh giá riêng**: Do đặc tính đối kháng của GAN, không sử dụng tập kiểm tra để đánh giá trực tiếp.
5. **Cơ chế dừng sớm và lưu trữ**:
   * **Early stopping**: Nếu mất mát Generator không cải thiện sau patience = 30 epoch (sau ít nhất 300 epoch), quá trình huấn luyện dừng sớm để tránh lãng phí tài nguyên.
   * **Lưu checkpoint**: Cứ sau 50 epoch, trạng thái Generator, Discriminator, bộ tối ưu hóa, và lịch sử mất mát được lưu vào thư mục checkpoints với định dạng gan\_checkpoint\_epoch\_{epoch}.pth.
   * **Lưu mô hình tốt nhất**: Mô hình với mất mát Generator thấp nhất được lưu vào best\_gan\_combined.pth.
   * **Lưu lịch sử huấn luyện**: Các giá trị mất mát (g\_losses, d\_losses, best\_epoch) được lưu vào training\_history\_gan\_combined.pth để phân tích sau này.

## **5.6. So sánh tổng quan**

**\*Phân tích hiệu suất**

**1. Hiệu suất mất mát**

* **CVAE**:
  + **BIDMC**: Test loss thấp nhất (**0.1124**, epoch 270, recon: 0.0644, KL: 0.0959) sau 296 epoch.
  + **Capnobase**: Test loss tốt nhất (**0.1053**, epoch 540, recon: 0.0525, KL: 0.1055) sau 600 epoch.
  + **Kết hợp**: Test loss **0.1143** (epoch 360, recon: 0.0642, KL: 0.1001) sau 452 epoch.
  + **Nhận xét**: CVAE hoạt động tốt nhất trên Capnobase với reconstruction loss thấp, cho thấy điều kiện HR cải thiện tái tạo PPG.
* **VAE**:
  + **BIDMC**: Test loss **0.1223** (epoch 230, recon: 0.0665, KL: 0.1116) sau 281 epoch.
  + **Capnobase**: Test loss **0.1157** (epoch 310, recon: 0.0539, KL: 0.1236) sau 316 epoch.
  + **Kết hợp**: Test loss thấp nhất (**0.1134**, epoch 430, recon: 0.0644, KL: 0.0981) sau 448 epoch.
  + **Nhận xét**: VAE hiệu quả trên Kết hợp, nhưng reconstruction loss kém hơn CVAE trên Capnobase.
* **CGAN**:
  + **BIDMC**: Generator loss **0.8510**, discriminator loss **0.6691** (epoch 320) sau 356 epoch.
  + **Capnobase**: Generator loss tốt nhất (**0.8427**, epoch 340, discriminator: 0.6840) sau 379 epoch.
  + **Kết hợp**: Generator loss **0.8614**, discriminator loss **0.6582** (epoch 440) sau 448 epoch.
  + **Nhận xét**: CGAN trên Capnobase có generator loss thấp nhất, cho thấy khả năng tạo PPG giống thật.
* **GAN**:
  + **BIDMC**: Generator loss **0.8891**, discriminator loss **0.6313** (epoch 300) sau 300 epoch.
  + **Capnobase**: Generator loss **0.8535**, discriminator loss **0.6660** (epoch 320) sau 378 epoch.
  + **Kết hợp**: Generator loss **0.8779**, discriminator loss **0.6377** (epoch 300) sau 300 epoch.
  + **Nhận xét**: GAN kém hơn CGAN, với generator loss cao và dừng sớm.

**2. Thời gian huấn luyện**

* **CVAE/VAE**: Dưới 10 phút, hiệu quả tính toán cao.
* **CGAN/GAN**: 30-35 phút, do huấn luyện đối kháng phức tạp.

**3. Số epoch và Early Stopping**

* **CVAE/VAE**: BIDMC hội tụ sớm (281-296 epoch), Capnobase cần nhiều epoch (316-600), Kết hợp cần nhiều nhất (448-452).
* **CGAN/GAN**: CGAN cần 356-448 epoch, GAN dừng sớm (300-378 epoch), cho thấy khó khăn khi không có điều kiện HR.

**4. So sánh tổng thể**

* **Mất mát**: CVAE (Capnobase: 0.1053) và VAE (Kết hợp: 0.1134) dẫn đầu về test loss. CGAN (Capnobase: 0.8427) tốt nhất về generator loss.
* **Điều kiện HR**: CVAE và CGAN vượt trội trên Capnobase, nhờ thông tin HR.
* **Tập dữ liệu**: Capnobase dễ học hơn BIDMC và Kết hợp.
* **Thời gian**: VAE/CVAE nhanh hơn CGAN/GAN 3-4 lần.

**5. Điểm nổi bật**

* **Điều thú vị**: CVAE trên Capnobase đạt reconstruction loss thấp nhất (**0.0525**), cho thấy tái tạo PPG chính xác.
* **Bất ngờ**: GAN trên BIDMC/Kết hợp dừng sớm (300 epoch) với generator loss cao (~0.8891), do thiếu HR.

**Bảng tổng hợp kết quả huấn luyện**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mô hình** | **Tập dữ liệu** | **Mất mát cuối cùng** | **Epoch** | **Thời gian huấn luyện** |
| CVAE | BIDMC | Test Loss: 0.1124 (Recon: 0.0644, KL: 0.0959) | 296 | Dưới 10 phút |
| CVAE | Capnobase | Test Loss: 0.1053 (Recon: 0.0525, KL: 0.1055) | 600 | Dưới 10 phút |
| CVAE | Kết hợp | Test Loss: 0.1155 (Recon: 0.0653, KL: 0.1004) | 452 | Dưới 10 phút |
| VAE | BIDMC | Test Loss: 0.1234 (Recon: 0.0674, KL: 0.1118) | 281 | Dưới 10 phút |
| VAE | Capnobase | Test Loss: 0.1157 (Recon: 0.0539, KL: 0.1236) | 316 | Dưới 10 phút |
| VAE | Kết hợp | Test Loss: 0.1134 (Recon: 0.0644, KL: 0.0981) | 448 | Dưới 10 phút |
| CGAN | BIDMC | Generator Loss: 0.8510, Discriminator Loss: 0.6691 | 356 | 30-35 phút |
| CGAN | Capnobase | Generator Loss: 0.8427, Discriminator Loss: 0.6840 | 379 | 30-35 phút |
| CGAN | Kết hợp | Generator Loss: 0.8614, Discriminator Loss: 0.6582 | 448 | 30-35 phút |
| GAN | BIDMC | Generator Loss: 0.8891, Discriminator Loss: 0.6313 | 300 | 30-35 phút |
| GAN | Capnobase | Generator Loss: 0.8535, Discriminator Loss: 0.6660 | 378 | 30-35 phút |
| GAN | Kết hợp | Generator Loss: 0.8779, Discriminator Loss: 0.6377 | 300 | 30-35 phút |

B5.1.Result comparison table

# **6. Trực quan hóa và đánh giá**

Các biểu đồ được cung cấp thể hiện sự so sánh trực quan giữa tín hiệu PPG thật và tín hiệu PPG tạo sinh từ bốn mô hình học sâu (CVAE, VAE, CGAN, GAN) trên ba tập dữ liệu (BIDMC, Capnobase, và Kết hợp). Mỗi biểu đồ hiển thị một số mẫu tín hiệu với nhịp tim (HR) tương ứng, cho phép đánh giá khả năng tái tạo tín hiệu của từng mô hình. Dưới đây là phân tích chi tiết:

## **6.1. Đánh giá từng mô hình**

**\*Mô hình CVAE**

|  |
| --- |
| **Tập dữ liệu BIDMC** |
|  |
| A6.1.Compare CVAE - BIDMC  **BIDMC**: Tín hiệu tạo sinh từ CVAE trên tập BIDMC cho thấy khả năng tái tạo tốt các đặc trưng chính của tín hiệu PPG thật, bao gồm các đỉnh systolic và nhịp điệu tổng thể. Tuy nhiên, tín hiệu tạo sinh có biên độ dao động lớn hơn một chút (ví dụ, mẫu #0 với HR = 91.4 bpm), và một số điểm dicrotic notch không được tái tạo rõ ràng. |

|  |
| --- |
| **Tập dữ liệu Capnobase** |
|  |
| A6.2.Compare CVAE - CAPNO  **Capnobase**: Trên tập Capnobase, CVAE tiếp tục thể hiện khả năng tái tạo nhịp điệu và hình dạng tín hiệu khá tốt, nhưng tín hiệu tạo sinh có xu hướng bị mất đi các chi tiết nhỏ, chẳng hạn như biến động nhẹ ở đáy sóng (ví dụ, mẫu HR = 0.223). Biên độ dao động của tín hiệu tạo sinh cũng không đồng đều so với tín hiệu thật. |

|  |
| --- |
| **Tập dữ liệu Kết hợp** |
|  |
| A6.3.Compare CVAE - COMBINED  **Kết hợp**: Trên tập Kết hợp, CVAE tạo ra tín hiệu có nhịp điệu tương đối chính xác, nhưng biên độ dao động của tín hiệu tạo sinh thường nhỏ hơn so với tín hiệu thật (ví dụ, mẫu HR = 0.416). Một số mẫu (như HR = 0.509) cho thấy tín hiệu tạo sinh có hình dạng sóng bị biến dạng nhẹ, không tái hiện đầy đủ các đặc trưng của tín hiệu thật. |

**\*Mô hình VAE**

|  |
| --- |
| **Tập dữ liệu BIDMC** |
|  |
| A6.4.Compare VAE - BIDMC  **BIDMC**: Tín hiệu tạo sinh từ VAE trên tập BIDMC có nhịp điệu khá tương đồng với tín hiệu thật, nhưng biên độ dao động thường nhỏ hơn (ví dụ, mẫu #0). Một số chi tiết nhỏ như dicrotic notch bị mất đi, và tín hiệu tạo sinh có xu hướng bị "làm mịn" quá mức, làm giảm độ sắc nét của các đỉnh sóng. |

|  |
| --- |
| **Tập dữ liệu Capnobase** |
|  |
| A6.5.Compare VAE - CAPNO  **Capnobase**: Trên tập Capnobase, VAE tái tạo tín hiệu với nhịp điệu chính xác, nhưng biên độ dao động của tín hiệu tạo sinh không nhất quán, đôi khi quá lớn hoặc quá nhỏ so với tín hiệu thật (ví dụ, mẫu #3). Tín hiệu tạo sinh cũng thiếu các biến động nhỏ ở đáy sóng, làm cho tín hiệu trông kém tự nhiên hơn. |

|  |
| --- |
| **Tập dữ liệu Kết hợp** |
|  |
| A6.6.Compare VAE - COMBINED  **Kết hợp**: Trên tập Kết hợp, VAE cho kết quả tương tự, với tín hiệu tạo sinh có nhịp điệu đúng nhưng biên độ dao động thường nhỏ hơn tín hiệu thật (ví dụ, mẫu HR = 81.0 bpm). Một số mẫu (như HR = 100.0 bpm) cho thấy tín hiệu tạo sinh bị mất đi các đặc trưng quan trọng, như độ sắc nét của đỉnh sóng. |

**\*Mô hình CGAN**

|  |
| --- |
| **Tập dữ liệu BIDMC** |
|  |
| A6.7.Compare CGAN - BIDMC  **BIDMC**: CGAN trên tập BIDMC tạo ra tín hiệu có nhịp điệu và biên độ dao động khá sát với tín hiệu thật (ví dụ, mẫu HR = 91.5 bpm). Tuy nhiên, tín hiệu tạo sinh có xu hướng bị nhiễu ở một số đoạn (ví dụ, mẫu HR = 108.2 bpm), làm giảm độ tự nhiên của tín hiệu. |

|  |
| --- |
| **Tập dữ liệu Capnobase** |
|  |
| A6.8.Compare CGAN - CAPNO  **Capnobase**: Trên tập Capnobase, CGAN tái tạo tín hiệu với nhịp điệu tốt, nhưng biên độ dao động của tín hiệu tạo sinh thường lớn hơn tín hiệu thật (ví dụ, mẫu HR = 104.3 bpm). Một số mẫu (như HR = 109.5 bpm) cho thấy tín hiệu tạo sinh bị biến dạng nhẹ ở các đỉnh sóng, làm mất đi tính đồng nhất. |

|  |
| --- |
| **Tập dữ liệu Kết hợp** |
|  |
| A6.9.Compare CGAN - COMBINED  **Kết hợp**: Trên tập Kết hợp, CGAN thể hiện khả năng tái tạo nhịp điệu tốt, nhưng tín hiệu tạo sinh có biên độ dao động không ổn định (ví dụ, mẫu HR = 91.0 bpm có biên độ nhỏ hơn tín hiệu thật). Một số mẫu (như HR = 100.0 bpm) cho thấy tín hiệu tạo sinh bị mất chi tiết ở đáy sóng, làm giảm tính chân thực. |

**\*Mô hình GAN**

|  |
| --- |
| **Tập dữ liệu BIDMC** |
|  |
| A6.10.Compare GAN - BIDMC  **BIDMC**: GAN trên tập BIDMC tạo ra tín hiệu có nhịp điệu khá chính xác, nhưng biên độ dao động của tín hiệu tạo sinh thường nhỏ hơn tín hiệu thật (ví dụ, mẫu #1). Tín hiệu tạo sinh cũng thiếu các chi tiết nhỏ như dicrotic notch, và một số đoạn có hiện tượng "làm mịn" quá mức. |

|  |
| --- |
| **Tập dữ liệu Capnobase** |
|  |
| A6.11.Compare GAN - CAPNO  **Capnobase**: Trên tập Kết hợp, GAN tạo ra tín hiệu với nhịp điệu khá sát, nhưng biên độ dao động thường nhỏ hơn tín hiệu thật (ví dụ, mẫu #1). Tín hiệu tạo sinh cũng bị mất đi các chi tiết nhỏ, như biến động ở đáy sóng, làm cho tín hiệu trông kém tự nhiên. |

|  |
| --- |
| **Tập dữ liệu Kết hợp** |
|  |
| A6.12.Compare GAN - COMBINED  **Kết Hợp**: Trên tập Capnobase, GAN tái tạo tín hiệu với nhịp điệu tương đối đúng, nhưng tín hiệu tạo sinh có biên độ dao động không nhất quán (ví dụ, mẫu #1875 có biên độ nhỏ hơn tín hiệu thật). Một số mẫu (như #1064) cho thấy tín hiệu tạo sinh bị biến dạng nhẹ ở các đỉnh sóng. |

## **6.2. Bảng đánh giá và so sánh hiệu suất trực quan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Mô hình** | **Tập dữ liệu** | **Nhịp điệu (Rhythm)** | **Biên độ dao động (Amplitude)** | **Chi tiết tín hiệu (Signal Details)** | **Đánh giá tổng quát** |
| CVAE | BIDMC | Tốt (cao) | Ổn định (trung bình) | Trung bình (thiếu dicrotic notch) | Tốt |
| CVAE | Capnobase | Tốt (cao) | Không đồng đều (trung bình) | Kém (mất chi tiết đáy sóng) | Trung bình |
| CVAE | Kết hợp | Tốt (cao) | Nhỏ hơn thật (trung bình) | Trung bình (biến dạng nhẹ) | Tốt |
| VAE | BIDMC | Tốt (cao) | Nhỏ hơn thật (kém) | Kém (làm mịn quá mức) | Trung bình |
| VAE | Capnobase | Tốt (cao) | Không nhất quán (kém) | Kém (thiếu chi tiết) | Kém |
| VAE | Kết hợp | Tốt (cao) | Nhỏ hơn thật (kém) | Kém (mất chi tiết) | Trung bình |
| CGAN | BIDMC | Tốt (cao) | Sát thật (tốt) | Trung bình (nhiễu nhẹ) | Tốt |
| CGAN | Capnobase | Tốt (cao) | Lớn hơn thật (trung bình) | Trung bình (biến dạng đỉnh sóng) | Trung bình |
| CGAN | Kết hợp | Tốt (cao) | Không ổn định (trung bình) | Kém (mất chi tiết đáy sóng) | Trung bình |
| GAN | BIDMC | Tốt (cao) | Nhỏ hơn thật (kém) | Kém (làm mịn quá mức) | Trung bình |
| GAN | Capnobase | Tốt (trung bình) | Nhỏ hơn thật (kém) | Kém (mất chi tiết) | Kém |
| GAN | Kết hợp | Tốt (trung bình) | Không nhất quán (kém) | Kém (biến dạng nhẹ) | Trung bình |

B6.Performance evaluation table

**Ghi chú:**

* **Nhịp điệu (Rhythm)**: Đánh giá mức độ chính xác của chu kỳ tín hiệu tạo sinh so với tín hiệu thật (cao, trung bình, kém).
* **Biên độ dao động (Amplitude)**: Đánh giá mức độ đồng nhất và sát với tín hiệu thật (tốt, trung bình, kém).
* **Chi tiết tín hiệu (Signal Details)**: Đánh giá khả năng tái hiện các đặc trưng nhỏ như dicrotic notch hoặc biến động đáy sóng (tốt, trung bình, kém).
* **Đánh giá tổng quát**: Tổng hợp dựa trên ba tiêu chí trên, phản ánh hiệu suất tổng thể của mô hình trên từng tập dữ liệu (tốt, trung bình, kém).

**6.4. Đánh giá mô hình**

* **CVAE** và **CGAN** cho thấy hiệu suất tốt nhất trên tập BIDMC, với nhịp điệu chính xác và biên độ dao động tương đối ổn định. Tuy nhiên, cả hai mô hình đều gặp khó khăn trong việc tái hiện chi tiết tín hiệu, đặc biệt trên tập Capnobase và Kết hợp.
* **VAE** và **GAN** có xu hướng làm mịn tín hiệu quá mức, dẫn đến mất chi tiết và biên độ dao động không nhất quán, đặc biệt kém trên tập Capnobase.
* **CGAN** nổi bật với khả năng tái tạo biên độ dao động sát với tín hiệu thật trên tập BIDMC, nhưng hiệu suất giảm trên các tập khác do nhiễu hoặc biến dạng.
* Tổng thể, **CVAE** và **CGAN** là hai mô hình đáng tin cậy nhất, trong khi **VAE** và **GAN** cần cải thiện để tăng độ chính xác và tự nhiên của tín hiệu tạo sinh.

# **7. Tính ứng dụng và mở rộng của dự án**

Dự án nghiên cứu việc tạo sinh tín hiệu PPG (Photoplethysmogram) dựa trên các mô hình học sâu (CVAE, VAE, CGAN, GAN) trên các tập dữ liệu BIDMC, Capnobase và Kết hợp mang lại nhiều tiềm năng ứng dụng thực tiễn trong lĩnh vực y tế và công nghệ, đồng thời có khả năng mở rộng đáng kể trong tương lai.

**Tính ứng dụng**

1. **Hỗ trợ chẩn đoán y tế**: Tín hiệu PPG tạo sinh có thể được sử dụng để hỗ trợ các thiết bị đeo thông minh (smartwatches, fitness trackers) trong việc giám sát nhịp tim và các chỉ số sức khỏe khác (như oxy trong máu, huyết áp) khi dữ liệu thật bị thiếu hụt hoặc không đầy đủ. Các mô hình như CGAN và CVAE, với khả năng tái tạo nhịp điệu và hình dạng sóng sát với tín hiệu thật, có thể cải thiện độ chính xác của các hệ thống giám sát từ xa.
2. **Đào tạo và kiểm định thuật toán**: Tín hiệu PPG tạo sinh cung cấp một nguồn dữ liệu phong phú để huấn luyện và kiểm định các thuật toán xử lý tín hiệu hoặc mô hình học máy khác mà không cần phụ thuộc hoàn toàn vào dữ liệu y tế thực tế, vốn thường bị hạn chế bởi quyền riêng tư và chi phí thu thập.
3. **Nghiên cứu mô phỏng y sinh**: Dự án có thể được ứng dụng trong việc mô phỏng các kịch bản sức khỏe khác nhau (ví dụ: nhịp tim bất thường, rối loạn tuần hoàn) để nghiên cứu phản ứng sinh lý, hỗ trợ các nhà khoa học phát triển các phương pháp điều trị mới mà không cần thử nghiệm trực tiếp trên bệnh nhân.
4. **Ứng dụng trong huấn luyện y tế**: Tín hiệu tạo sinh có thể được tích hợp vào các chương trình đào tạo y khoa hoặc mô phỏng lâm sàng, cho phép sinh viên và bác sĩ thực hành phân tích tín hiệu mà không cần tiếp xúc với dữ liệu nhạy cảm của bệnh nhân.

**Khả năng mở rộng**

1. **Mở rộng tập dữ liệu**: Dự án có thể tích hợp thêm các tập dữ liệu PPG từ nhiều nguồn khác nhau (ví dụ: MIMIC, PhysioNet) để cải thiện độ tổng quát của mô hình, đồng thời mở rộng sang các loại tín hiệu sinh lý khác như ECG (electrocardiogram) hoặc EEG (electroencephalogram).
2. **Tối ưu hóa mô hình**: Các mô hình như CGAN và CVAE có thể được cải tiến bằng cách kết hợp các kỹ thuật mới (ví dụ: attention mechanism, transfer learning) để tăng độ chính xác trong tái tạo chi tiết tín hiệu, đặc biệt là các đặc trưng nhỏ như dicrotic notch, và giảm nhiễu trong tín hiệu tạo sinh.
3. **Ứng dụng đa nền tảng**: Kết quả của dự án có thể được triển khai trên các nền tảng khác nhau, từ các thiết bị y tế chuyên dụng đến ứng dụng di động, với khả năng tích hợp vào hệ thống AI thời gian thực để phân tích sức khỏe liên tục.
4. **Hỗ trợ nghiên cứu đa lĩnh vực**: Khả năng mở rộng của dự án cho phép áp dụng vào các lĩnh vực liên quan như trí tuệ nhân tạo y tế, xử lý tín hiệu sinh học, và phát triển phần mềm mô phỏng, mở ra cơ hội hợp tác với các tổ chức nghiên cứu hoặc doanh nghiệp công nghệ.
5. **Tích hợp với công nghệ đám mây**: Dữ liệu và mô hình có thể được lưu trữ trên các nền tảng đám mây, cho phép phân tích quy mô lớn và chia sẻ với cộng đồng khoa học, đồng thời hỗ trợ cập nhật mô hình theo thời gian thực dựa trên dữ liệu mới.

# **8. Kết luận**

Dự án nghiên cứu việc tạo sinh tín hiệu PPG sử dụng các mô hình học sâu (CVAE, VAE, CGAN, GAN) trên ba tập dữ liệu BIDMC, Capnobase và Kết hợp đã đạt được những kết quả đáng ghi nhận, đồng thời mở ra nhiều tiềm năng ứng dụng và hướng phát triển trong tương lai.

Qua quá trình đánh giá trực quan, các mô hình CVAE và CGAN thể hiện hiệu suất vượt trội trong việc tái tạo nhịp điệu và hình dạng sóng của tín hiệu PPG, đặc biệt trên tập dữ liệu BIDMC và Kết hợp, với khả năng tái hiện sát tín hiệu thật. Trong khi đó, VAE và GAN gặp hạn chế về việc làm mịn tín hiệu quá mức, dẫn đến mất chi tiết và biên độ dao động không nhất quán, đặc biệt trên tập Capnobase. Kết quả này cho thấy CVAE và CGAN là hai mô hình tiềm năng nhất cho các ứng dụng yêu cầu tín hiệu PPG chất lượng cao, mặc dù vẫn cần cải thiện để tái hiện đầy đủ các đặc trưng nhỏ như dicrotic notch.

Về tính ứng dụng, dự án mang lại giá trị lớn trong các lĩnh vực như giám sát sức khỏe từ xa, đào tạo thuật toán, nghiên cứu y sinh, và huấn luyện y tế. Khả năng mở rộng của dự án cũng được đánh giá cao, với tiềm năng tích hợp thêm dữ liệu mới, cải tiến mô hình, và triển khai trên các nền tảng đa dạng, từ thiết bị y tế đến hệ thống đám mây.

Tóm lại, dự án không chỉ cung cấp một giải pháp hiệu quả để tạo sinh tín hiệu PPG mà còn đặt nền móng cho các nghiên cứu tiếp theo trong lĩnh vực y tế và trí tuệ nhân tạo. Những cải tiến trong tương lai, tập trung vào việc nâng cao độ chính xác của tín hiệu tạo sinh và mở rộng ứng dụng thực tiễn, hứa hẹn sẽ mang lại tác động tích cực trong việc hỗ trợ chẩn đoán và chăm sóc sức khỏe.

# **Tài liệu tham khảo**

1. BIDMC PPG and Respiration Dataset: Pimentel, M. A. F., Johnson, A. E. W., Charlton, P. H., Birrenkott, D., Watkinson, P. J., Tarassenko, L., & Clifton, D. A. (2016). "Toward a Robust Estimation of Respiratory Rate from Pulse Oximeters." *IEEE Transactions on Biomedical Engineering*, 63(8), 1914-1923. Available at PhysioNet: <https://physionet.org/content/bidmc/1.0.0/>
2. Capnobase TBME Benchmark Dataset: Karlen, W., Turner, M., Dumont, G. A., & Ansermino, J. M. (2010). "CapnoBase: Signal Processing and Machine Learning Benchmark Dataset for Respiratory and Cardiovascular Analysis." *IEEE Transactions on Biomedical Engineering*. Available at CapnoBase: <http://www.capnobase.org>
3. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). "Generative Adversarial Nets." *Advances in Neural Information Processing Systems (NeurIPS)*, 2672-2680.
4. Kingma, D. P., & Welling, M. (2014). "Auto-Encoding Variational Bayes." *International Conference on Learning Representations (ICLR)*.
5. Sohn, K., Lee, H., & Yan, X. (2015). "Learning Structured Output Representation using Deep Conditional Generative Models." *Advances in Neural Information Processing Systems (NeurIPS)*, 3483-3491.
6. Allen, J. (2007). "Photoplethysmography and its application in clinical physiological measurement." *Physiological Measurement*, 28(3), R1-R39.
7. Elgendi, M. (2012). "On the Analysis of Fingertip Photoplethysmogram Signals." Current Cardiology Reviews, 8(1), 14-25.

Trang Github của dự án: <https://github.com/CterHAK/AIOT/tree/main/HongAnhKhoa_cuoiky/final>