Báo Cáo TakeHomeTest: Phát Hiện Giả Mạo Khuôn Mặt (Liveness Classifier)

Phần 1: Xây dựng mô hình phân loại Liveness

Mục tiêu:

Xây dựng một mô hình AI có khả năng phân loại ảnh đầu vào là khuôn mặt thật ("normal") hay giả mạo ("spoof") trong hệ thống eKYC hoặc kiểm soát truy cập. Kết quả trả về của mô hình là liveness score nằm trong khoảng từ [0,1] với 0 đại diện cho ảnh đó thuộc về khuôn mặt thật (hay class “normal”) và 1 đại diện cho ảnh đó thuộc về khuôn mặt giả (hay class “spoof”)

Phân tích bài toán và dữ liệu:

- Tập dữ liệu được cấu trúc với 2 thư mục train, test riêng biệt. Bên trong mỗi thư mực lại chứa các subfolder là “normal” và “spoof” tương ứng với từng class. Do trong dataset chỉ có ảnh thuộc về 2 class với tỉ lệ phân bố cân bằng ở cả tập train và test nên em sẽ coi như đây là bài toán phân loại nhị phân (Binary Classification).
- Em để ý thấy trong dataset, mỗi ảnh sẽ có tên dưới dạng Oi_Ij.jpg trong đó:
    - Oi là đối tượng thứ I và Ij là ảnh thứ j thuộc về đối tượng Oi.
    - Ta thấy j<=4 (Với hầu hết là 4 ở mọi trường hợp và 1 ở một vài ảnh nhất định)
- Vì vậy em sẽ tiếp cận theo hai hướng nhỏ là:
    - Sử dụng đầu vào là một ảnh và coi từng ảnh là một đối tượng riêng biệt. (Kể cả khi các ảnh đó thuộc về cùng một đối tượng)
    - Sử dụng 4 ảnh của một đối tượng làm input. (Nếu như đối tượng có nhiều hơn 4 ảnh thì sẽ lấy 4 ảnh đầu tiên, ít hơn 4 ảnh thì sẽ lấy lặp lại ảnh đầu tiên 4 lần)
- Ngoài ra bên trong dataset có một số ảnh bị lật ngược so với số còn lại, nhưng vì số lượng ít nên em giữ nguyên và coi những ảnh này như Outlier.
- Các ảnh trong dataset hầu hết chụp người đeo khẩu trang ở góc mặt chính diện (nhưng vẫn có một số ảnh chụp ở các góc mặt khác), độ sáng cũng như background khác nhau.

Tiền xử lý dữ liệu:

- Đối với cách tiếp cận thứ nhất:
    - Sử dụng thư viện torchvision.transforms để chuẩn hóa hình ảnh:
        - Resize ảnh về kích thước cố định (224x224)
        - Augmentation với HorizontalFlip, Rotation để mô hình có thể học được các góc xoay khác của đối tượng trong ảnh.
        - Chuyển ảnh về tensor và chuẩn hóa theo ImageNet (mean, std)
- Đối với cách tiếp cận thứ 2:
    - Tạo **Custom Dataset (MultiImageLivenessDataset**) để nhóm 4 ảnh liên tiếp của cùng một đối tượng làm một sample.
    - Vẫn giữ nguyên hàm transforms cơ bản như ở cách tiếp cận thứ nhất.

Phương pháp sử dụng:

- Mô hình 1: **Resnet50 (Pretrained) - Single-Image**
    - Sử dụng **Resnet50 (Pretrained)** loại bỏ lớp Fc cuối cùng để làm backbone trích xuất đặc trưng.
    - Thay thế lớp Fc cuối cùng bằng một MLP để phân loại từ đầu ra của backbone **Resnet50.**
    - Quá trình huấn luyện chỉ cập nhật trọng số MLP dùng để thay thế lớp Fc cuối cùng của Resnet50 chứ không huấn luyện toàn bộ mô hình. Bởi vì Resnet50 đã được huấn luyện trên các bộ dữ liệu lớn và có khả năng trích xuất tốt được các đặc trưng từ ảnh nên ta không cần huấn luyện lại, và việc chỉ huấn luyện MLP sau cùng kết hợp với dropout sẽ đẩy nhanh quá trình huấn luyện, giảm bớt chi phí.
    - LossFunction: **CrossEntropyLoss**
    - Optimizer: **Adam**
    - Learning rate: 1e-4
    - Num_epochs: 10
    - Batch_size: 8
    - Đánh giá: Sử dụng classification_reports từ thư scikit-learn để đánh giá mô hình trên 3 tiêu chí: **precision**, **recall** và **f1-score.**
    - Kết quả:

	precision	recall	f1-score	support
normal	0.88	0.78	0.83	602
spoof	0.80	0.90	0.85	602
accuracy			0.84	1204
macro avg	0.84	0.84	0.84	1204
weighted avg	0.84	0.84	0.84	1204

- Mô hình 2: **Resnet50 (pretrained) + LSTM – Multi-Image**
    - Đầu vào thay vì là 1 ảnh (Single-Image) như mô hình 1 thì ở đây ta sử dụng cả 4 ảnh như đã đề cập trong phần **Phân tích bài toán và dữ liệu.**
    - Sử dụng **Resnet50 (Pretrained)** loại bỏ lớp Fc cuối cùng như 1 **Feature Extractor** để trích xuất đặc trưng cho từng ảnh.
    - Sau đó **LSTM** nhận input chính là output của **Feature Extractor –** Sequence các vector đặc trưng của từng ảnh. Output của **LSTM**
    - Thay thế lớp Fc cuối cùng bằng một MLP nhỏ để phân loại từ đầu ra của **LSTM.**
    - Quá trình huấn luyện cập nhật trọng số của toàn bộ mô hình chứ không đóng băng tham số bất kì thành phần nào.
    - LossFunction: CrossEntropyLoss
    - Optimizer: Adam
    - Learning rate: 1e-4
    - Num_epochs: 5
    - Batch_size: 8

- Đánh giá: Sử dụng classification_reports từ thư scikit-learn để đánh giá mô hình trên 3 tiêu chí: **precision**, **recall** và **f1-score**.
- Kết quả:

	precision	recall	f1-score	support
normal	0.95	0.94	0.94	201
spoof	0.94	0.95	0.94	204
accuracy			0.94	405
macro avg	0.94	0.94	0.94	405
weighted avg	0.94	0.94	0.94	405

- Mô hình 3: **VIT (pretrained) + Mean Pooling – Multi-Image**
  - Đầu vào cũng sử dụng cả 4 ảnh như ở mô hình 2
  - Sử dụng **vit_base_patch16_224 (Pretrained)** loại bỏ classification head gốc (để phân loại 1000 class) như 1 **Feature Extractor** để trích xuất đặc trưng cho từng ảnh.
  - Ở đây ta không dùng **LSTM** mà thay vào đó là lấy **Mean Pooling** từ đầu ra của Feature Extractor. Lý do là vì em muốn thự nghiệm đa dạng các phương pháp, đồng thời tăng tốc độ huấn luyện, giảm tài nguyên so với việc dùng **LSTM**. Và trên hết, em không chắc chắn rằng các ảnh trong dataset có tính tuần tự hay không, cộng với việc bên trong **VIT** đã có cơ chế attention cho phép trích xuất được các đặc trưng liên quan đến mối quan hệ giữa các ảnh của một đối tượng. Do đó, trong mô hình này em quyết định chọn **Mean Pooling** thay vì **LSTM**.
  - Cuối cùng là đưa Output từ quá trình Mean Pooling vào MLP để phân loại**.**
  - Quá trình huấn luyện cập nhật trọng số của toàn bộ mô hình chứ không đóng băng tham số bất kì thành phần nào.
  - LossFunction: CrossEntropyLoss
  - Optimizer: Adam
  - Learning rate: 2e-5
  - Num_epochs: 2
  - Batch_size: 8
  - Đánh giá: Sử dụng classification_reports từ thư scikit-learn để đánh giá mô hình trên 3 tiêu chí: **precision**, **recall** và **f1-score**.
  - Kết quả:

	precision	recall	f1-score	support
normal	0.94	0.96	0.95	201
spoof	0.96	0.94	0.95	204
accuracy			0.95	405
macro avg	0.95	0.95	0.95	405
weighted avg	0.95	0.95	0.95	405

- Mô hình 4: **AutoEncoder + ResNet18 Classifier – Multi-Image**
  - Tương tư như mô hình 1 nhưng thay vì đầu vào của ResNet Backebone là ảnh gốc thì ta đưa vào Output Embbeding sinh ra bởi Encoder cảu AutoEncoder.

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

class SimpleAutoEncoder(nn.Module):
    def __init__(self):
        super(SimpleAutoEncoder, self).__init__()
        # Encoder
        self.encoder = nn.Sequential(
            nn.Conv2d(3, 32, 3, stride=2, padding=1),  # [B, 32, 112, 112]
            nn.ReLU(),
            nn.Conv2d(32, 64, 3, stride=2, padding=1),  # [B, 64, 56, 56]
            nn.ReLU(),
            nn.Conv2d(64, 128, 3, stride=2, padding=1),  # [B, 128, 28, 28]
            nn.ReLU(),
        )
        # Decoder
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(128, 64, 3, stride=2, padding=1, output_padding=1),  # [B, 64, 56, 56]
            nn.ReLU(),
            nn.ConvTranspose2d(64, 32, 3, stride=2, padding=1, output_padding=1),  # [B, 32, 112, 112]
            nn.ReLU(),
            nn.ConvTranspose2d(32, 3, 3, stride=2, padding=1, output_padding=1),  # [B, 3, 224, 224]
            nn.Sigmoid()
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

  - Sử dụng **vit_base_patch16_224 (Pretrained)** loại bỏ classification head gốc (để phân loại 1000 class) như 1 **Feature Extractor** để trích xuất đặc trưng cho từng ảnh.
  - Ta truyền vào Input của AutoEncoder là sample 4 ảnh của đối tượng (Không sử dụng LSTM hay Mean Pooling). Sau đó tiếp tục đi qua Resnet18 Classifier để phân loại.
  - Quá trình huấn luyện cập nhật trọng số của toàn bộ mô hình chứ không đóng băng tham số bất kì thành phần nào.
  - LossFunction: CrossEntropyLoss
  - Optimizer: Adam
  - Learning rate: 5e-4
  - Num_epochs: 10
  - Batch_size: 8
  - Đánh giá: Sử dụng classification_reports từ thư scikit-learn để đánh giá mô hình trên 3 tiêu chí: **precision**, **recall** và **f1-score**.
  - Kết quả:

	precision	recall	f1-score	support
normal	0.89	0.49	0.63	201
spoof	0.65	0.94	0.77	204
accuracy			0.72	405
macro avg	0.77	0.71	0.70	405
weighted avg	0.77	0.72	0.70	405

- Ngoài 4 mô hình ở trên thì em còn thử thiết kế thêm 1 mô hình theo ý tưởng từ bài toán Anomaly Detection. Bởi vì trong thực tế thì sẽ có nhiều kiểu tấn công hơn và ta sẽ không biết hệ thống cần phân loại với những kiểu tấn công nào. Khi đó thì ta có thể dễ dàng thu thập data của class "normal" nhưng lại rất khó để thu thập data của các kiểu tấn công (hay nói chung là "spoof"). Khi đó ta cần một mô hình chỉ huấn luyện trên data "normal" nhưng vẫn phát hiện được "spoof"
- Mô hình 5: **AutoEncoder + Reconstruction Error – Multi-Image (Normal)**
    - Kiến trúc AutoEncoder tương tự với mô hình 4, nhưng thay vì dùng Resnet Classifier để phân loại thì ta sẽ phân loại dựa trên Reconstruction Error giữa Input và Output của Decoder.
    - Reconstruction Error được tính ở Pixel level và lấy trung bình cho 4 ảnh.
    - Threshold để phân loại Normal và Spoof được chọn bằng cách lấy argmax() của hiệu số (TPR-FPR).
    - Quá trình huấn luyện cập nhật trọng số của toàn bộ mô hình chứ không đóng băng tham số bất kì thành phần nào.
    - LossFunction: MSELoss
    - Optimizer: Adam
    - Learning rate: 1e-3
    - Num_epochs: 10
    - Batch_size: 16
    - Đánh giá: Sử dụng classification_reports từ thư scikit-learn để đánh giá mô hình trên 3 tiêu chí: **precision**, **recall** và **f1-score**.
    - Kết quả:

[INFO] Best threshold: 0.0024

	precision	recall	f1-score	support
normal	0.53	0.49	0.51	201
spoof	0.53	0.57	0.55	204
accuracy			0.53	405
macro avg	0.53	0.53	0.53	405
weighted avg	0.53	0.53	0.53	405

Nhận xét và đánh giá:

- Mô hình 5 cho kết quả thấp nhất bởi vì chỉ được train trên tập normal, và chỉ train 10 epoch nên có khả năng model đang bị UnderFitting. Lúc này mô hình chưa học được toàn bộ đặc trưng để phân biệt được 2 class. Đây vẫn là một phương pháp để có thể tiếp tục phát triển trong tương lai.
- Hai mô hình 2 và 3 cho kết quả tốt nhất với Accuracy ở mức 0.94 và 0.95. Điều này cho thấy việc kết hợp các ảnh của một đối tượng thành một sample, sau đó đưa vào mô hình sẽ tốt hơn là phân loại trên từng ảnh riêng lẻ như mô hình 1.
- Thời gian huấn luyện của mô hình 4 là lâu nhất, tiếp đó lần lượt là mô hình 3, mô hình 5, mô hình 1 và cuối cùng là mô hình 2. Điều đáng chú y là mô hình 2 với độ chính xác tương đương mô hình 3 nhưng thời gian huấn luyện và thời gian thực thi trong thực tế đều nhanh hơn nhiều. (Nhanh gấp 5 lần)

  -> Với ngữ cảnh hiện tại của bài toán (hay Dataset có phân bố tương tự với thực tế) thì mô hình 2 sẽ là phương pháp tối ưu đối với bài toán Liveness Classifier.

Phần 2: Báo cáo hiệu quả mô hình (30 điểm)

Các chỉ số đánh giá:

- **Accuracy**: Tỷ lệ dự đoán đúng của mô hình, Accuracy trung bình trên toàn bộ tập dev là 0.94.
- **Recall (cho spoof)**: Đây là một chỉ số rất quan trọng vì hệ thống cần bắt được tất cả các trường hợp spoof, đặc biệt là đối với CakebyVPBank – hệ thống ngân hàng số, trực tiếp liên quan đến tài chính, thẻ tín dụng và các thông tin quan trọng của khách hàng. Recall trung bình trên toàn bộ tập dev là 0.94 và cho từng class là 0.94 và 0.95 cho normal/spoof.
- **Precision**: Precision trung bình trên toàn bộ tập dev là 0.94 và cho từng class là 0.95 và 0.94 cho normal/spoof.
- **F1-score**: Là chỉ số điều hòa cân bằng Precision và Recall, đạt 0.94 trên toàn bộ tập dev
- **ROC-AUC**: Đạt 0.8573 trên toàn bộ tập dev

Lý do lựa chọn:

- Đây là các chỉ số quen thuộc với bài toán Classification và thường được dùng để đánh giá một mô hình có phân loại chính xác các class hay không.
- Như đã nói ở trên, thì vì CakebyVPBank là nền tảng ngân hàng số nên Recall đặc biệt quan trọng, ta có thể đánh đổi (trade-off) Precision để đạt một mức Recall đủ cao và được coi là an toàn. Như vậy thì mới có thể đảm bảo sự an toàn cho thông tin khách hàng.
- F1-score giúp đánh giá tổng thể.

Hạn chế mô hình:

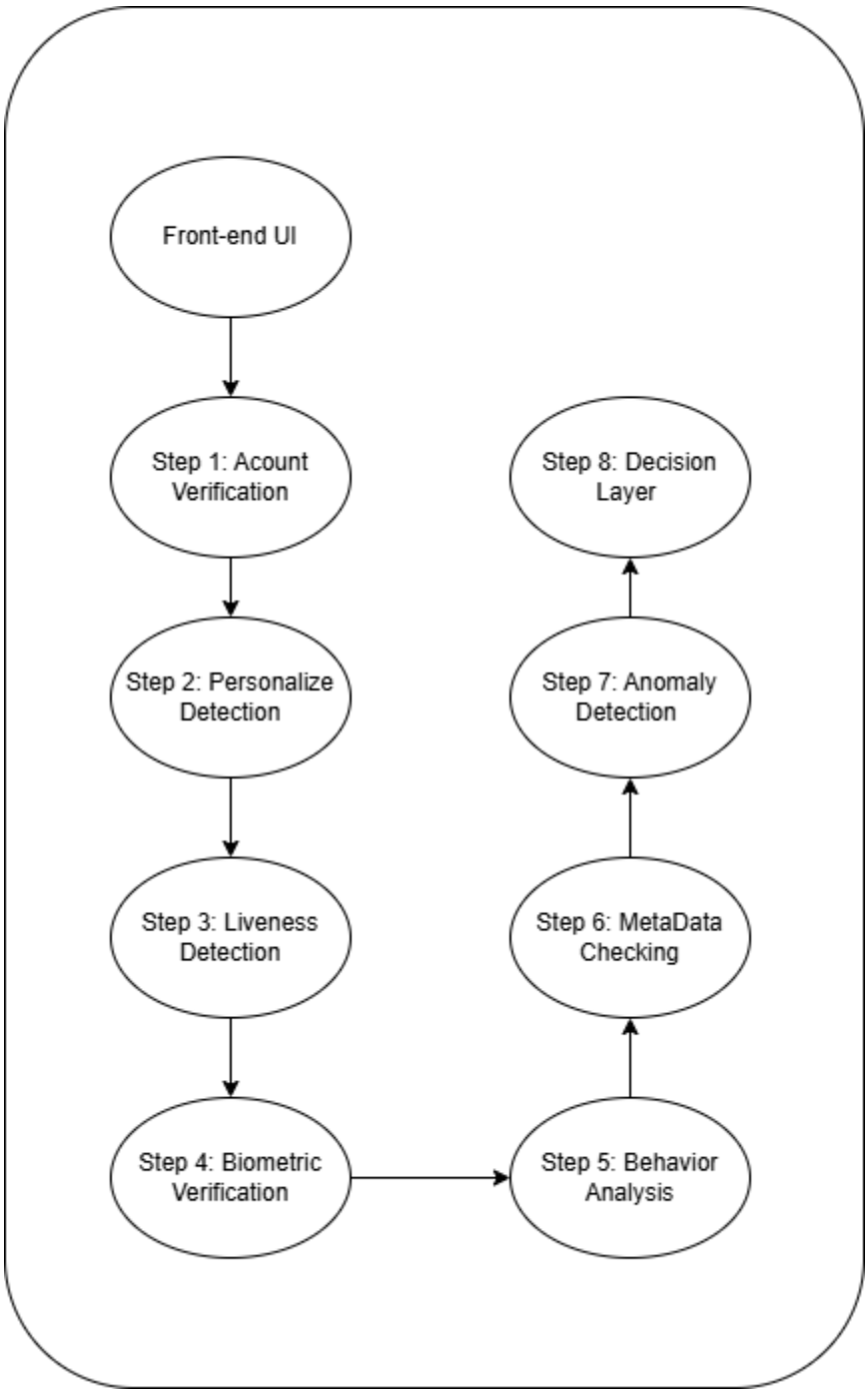
- Độ phân giải ảnh thấp, nhiều ảnh bị mờ hoặc thiếu thông tin có thể dẫn đến việc mô hình đoán sai. Một số sample thuộc class spoof nhưng rất khó để phân biệt (kể cả mắt thường) nên mô hình vẫn còn nhầm lẫn ở một số case đặc biệt.
- Dataset chỉ chứa các loại spoof cơ bản như display hay printed, chưa bao quát được các trường hợp như 3d-mask hay deepfake.
- Người trong ảnh đeo khẩu trang nên làm hạn chế các đặc trưng có thể sử dụng để phân loại như mũi hay miệng.
- Việc các ảnh là ảnh tĩnh, rời rạc cũng gây khó cho việc mô hình dự đoán.

Gợi ý cải tiến:

- Sử dụng video (chuỗi ảnh) để học chuyển động vi mô.
- Áp dụng thêm kỹ thuật contrastive learning hoặc triplet loss để tăng khả năng phân biệt.
- Sử dụng thêm các metadata như depth, IR (nếu có) từ các thiết bị phần cứng.
- Dùng ensemble nhiều model (ResNet + ViT).
- Sử dụng kiến trúc GAN và các Generative Model để Augment dữ liệu với các mẫu spoof tinh vi hơn nhằm phục vụ training.
- Có thể sử dụng thêm một số personalize information như vị trí địa lý, thời tiết, thời gian để có thể phân loại tốt hơn. Ví dụ một người thực hiện giao dịch lúc 10h sáng tại TP.HCM vào mùa hè thì thời tiết lúc sử dụng hệ thống Liveness Classifier không thể là buổi tối hay là có tuyết được.
- Ngoài ra, tích hợp một số cơ chế Explainable để phân loại nhằm tăng uy tín cho hệ thống. Ví dụ: Visualize các đặc trưng nổi bật mà mô hình dùng để phân loại như sống mũi, khuôn miệng hay ánh sáng phanr chiếu lên da.

Phần 3: Thiết kế hệ thống phát hiện gian lận đa lớp (30 điểm)

Kiến trúc hệ thống đề xuất:



Chi tiết từng lớp:

- **Step 1: Liveness Detection**:
  - **Mục tiêu**: Xác thực danh tính ban đầu bằng tài khoản đã đăng ký trước. Việc này có thể thực hiện bằng việc nhập tài khoản-mật khẩu hoặc sử dụng vân tay.
  - **Thực hiện**: So sánh thông tin đầu vào với cơ sở dữ liệu. Kiểm tra tính hợp lệ, phát hiện các thông tin trùng lặp hoặc bất thường (Ví dụ có 2 tài khoản sử dụng chung số điện thoại hoặc căn cước công dân.)

- **Step 2: Personalize Detection**:
  - **Mục tiêu**: Sử dụng các thông tin cá nhân để xác định khách hàng, ví dụ câu hỏi bảo mật cấp 2 và câu trả lời chỉ có mình chủ sở hữu biết).
  - **Thực hiện**: So sánh với cơ sở dữ liệu.
- **Step 3: Liveness Detection**:
  - **Mục tiêu**: Kiểm tra người dùng có thật sự đang hiện diện không (Có phải người thật hay không).
  - **Thực hiện**: Sử dụng ảnh hoặc chuỗi ảnh đầu vào để phát hiện fake ảnh, video, deepfake. Áp dụng các mô hình Liveness Classifier như VIT-based, CNN hoặc AutoEncoder
- **Step 4: Biometric Verification:**
  - **Mục tiêu:** So khớp khuôn mặt người dùng với dữ liệu đã đăng ký.
  - **Thực hiện:** Trích xuất embedding khuôn mặt từ ảnh/video bằng mô hình DeepLearning. Tính cosine similarity giữa ảnh hiện tại và ảnh đã lưu. Nếu độ tương đồng đủ lớn thì chuyển qua bước tiếp theo, còn không thì dừng lại tại bước này.
- **Step 5: Behavior Analysis**
  - **Mục tiêu**: Phát hiện bot hoặc giả mạo qua hành vi không tự nhiên.
  - **Thực hiện**: So sánh với hành vi trước đó hoặc mẫu hành vi thật. Ta có thể đưa ra chuỗi hành động và nhờ khách hàng thực hiện đúng theo thứ tự như quay sang trái phải, nháy mắt,.. Những thông tin này đã được thu thập từ lần đầu khách hàng đăng kí tài khoản. Nếu khách hàng làm đúng thì ta qua bước tiếp theo.
- **Step 6: Metadata Checking**:
  - **Mục tiêu**: Phát hiện dấu hiệu giả mạo từ thiết bị & môi trường sử dụng.
  - **Thực hiện**: Thu thập IP, OS, Camera Info,... Ta sẽ ngăn lại nếu phát hiện bất thường (ví dụ Camera Samsung Galaxy S23 Ultra nhưng độ phân giải lại thấp, hoặc định vị ở khu đô thị nhưng lại xuất hiện cảnh đồng ruộng, thác nước,…)
- **Step 7: Anomaly Detection:**
  - **Mục tiêu:** Phát hiện bất thường trong quá trình xác minh (Ví dụ thực hiện 100 giao dịch trong 1 phút).
  - **Thực hiện:** Kết hợp rule-based, Machine Learning cùng với Các mô hình như Isolation Forest, AutoEncoder, hoặc One-Class SVM để phát hiện outliers.
- **Step 8: Decision Layer**
  - **Mục tiêu**: Đưa ra quyết định cuối cùng: Duyệt, Yêu cầu xác minh thêm, hoặc Từ chối.
  - **Thực hiện**: Dựa vào risk score tổng hợp từ các bước trên. Gán nhãn hành vi theo ngưỡng:
    - Low Risk: Tự động duyệt.
    - Medium Risk: Gửi yêu cầu xác minh thêm.
    - High Risk: Từ chối hoặc chuyển đến kiểm tra thủ công.

Thách thức và giải pháp:

- **Chi phí tính toán & Độ trễ cao:**
  - **Vấn đề**: Việc áp dụng nhiều tầng kiểm tra (Liveness, Face Matching, Behavior, Metadata…) có thể gây độ trễ lớn → ảnh hưởng trải nghiệm người dùng
  - **Giải pháp**:
    - Sử dụng pipeline linh hoạt, chỉ kích hoạt các tầng sâu hơn khi nghi ngờ (ví dụ: nếu Liveness pass rõ ràng → bỏ qua Behavior).
    - Tối ưu hóa mô hình: dùng model nhẹ hơn (MobileNet, EfficientNet, TinyViT).
    - Triển khai song song hoặc bất đồng bộ (async API) để giảm thời gian chờ.
- **Dữ liệu không đồng đều hoặc thiếu dữ liệu hành vi:**
  - **Vấn đề**: Behavior analysis yêu cầu dữ liệu hành vi người dùng theo thời gian → khó khăn với người dùng mới (cold-start).
  - **Giải pháp**:
    - Sử dụng các đặc trưng tổng quát không phụ thuộc người dùng cụ thể (ví dụ: thời gian hoàn tất thao tác, độ dao động chuột).
    - Áp dụng anomaly detection không giám sát như Isolation Forest hoặc AutoEncoder để phát hiện bất thường so với mẫu chung.
- **Tấn công giả mạo tinh vi (Deepfake, Ảnh chỉnh sửa, Emulator)**
  - **Vấn đề**: Các hình thức tấn công ngày càng tinh vi (deepfake video, face morphing, spoof ảnh giấy…).
  - **Giải pháp**:
    - Cập nhật mô hình Liveness thường xuyên, huấn luyện với dataset phong phú chứa nhiều loại spoof.
    - Sử dụng cross-modal verification: đối chiếu ảnh/video với metadata thiết bị, phân tích ánh sáng, phản chiếu (spoof cues).
    - Kết hợp mô hình học sâu với các chỉ số thủ công (manual cues).
- **Tính riêng tư & bảo mật dữ liệu cá nhân**
  - **Vấn đề**: Các bước như Biometric Verification và Behavior Tracking có thể vi phạm quyền riêng tư người dùng.
  - **Giải pháp**:
    - Tuân thủ quy định bảo mật (VD: GDPR, Decree 13/2023/NĐ-CP tại Việt Nam).
    - Dữ liệu nhạy cảm (khuôn mặt, hành vi) cần được mã hóa, lưu trữ cục bộ hoặc giới hạn thời gian giữ.
    - Minh bạch với người dùng về lý do thu thập dữ liệu.
- **Khó đánh giá & giám sát toàn hệ thống**
  - **Vấn đề**: Hệ thống nhiều tầng dễ gây khó khăn trong việc debug, theo dõi lỗi, hoặc hiểu vì sao người dùng bị từ chối.
  - **Giải pháp**:
    - Thiết kế hệ thống ghi log rõ ràng từng bước (audit trail).
    - Cung cấp Explainable AI (giải thích được quyết định) ở tầng cuối.
    - Tạo giao diện Admin để kiểm tra các quyết định và điểm số từ từng tầng
- **Khả năng thích nghi với tấn công mới**
  - **Vấn đề**: Kẻ gian luôn tìm cách vượt qua hệ thống bằng cách sáng tạo kỹ thuật tấn công mới.

- **Giải pháp:**
  - Hệ thống cần thu thập dữ liệu phản hồi liên tục, cập nhật model thường xuyên (online learning, continual learning).
  - Kết hợp thống kê bất thường theo thời gian, tự động phát hiện pattern mớ

Hướng dẫn chạy code:

1. Upload Dataset.zip lên drive
2. Upload notebook Liveness_Classification_Resnet50Multi.ipynb lên Google Colab, sau đó chạy từng cell code hoặc chạy all. Cài đặt thời gian chạy GPU
3. Nếu đã có weights của mô hình thì chỉ cần load model (không train lại)
4. Nếu chưa có weights thì ta cần train lại từ đầu.
5. Chạy cell evaluate để coi kết quả trên tập dev.
6. Thay đường dẫn ảnh mong muốn trong cell predict để dự đoán một hoặc nhiều ảnh (thuộc về một đối tượng) là normal hay spoof (kèm confidence score)

Link Github: ShouyiLeee/TakeHomeTest