



## Đồ án môn Phân tích và thiết kế phần mềm

# Image Deep Hash

Sinh viên thực hiện:  
18120046 – Dương Anh Kiệt  
18120051 – Nguyễn Hoàng Lân

## Bảng ghi nhận thay đổi tài liệu

Ngày	Phiên bản	Mô tả	Người thay đổi
06/06/2021	1.0	Đăng ký ý kiến đề án	Dương Anh Kiệt
10/08/2021	1.1	Phát biểu bài toán	Dương Anh Kiệt
09/11/2021	1.2	Sơ đồ usecase	Dương Anh Kiệt
10/11/2021	1.3	Phát biểu bài toán, cách thiết kế phần mềm	Dương Anh Kiệt + Nguyễn Hoàng Lân
11/11/2021	1.4	Sơ đồ lớp mức phân tích	Dương Anh Kiệt
12/11/2021	1.5	Thiết kế giao diện	Nguyễn Hoàng Lân
12/11/2021	1.6	Kết quả thực hiện	Nguyễn Hoàng Lân
13/11/2021	1.7	Kiểm tra và hoàn thiện báo cáo	Dương Anh Kiệt + Nguyễn Hoàng Lân

# Mục lục

<b>Thông tin chung</b>	3
Tên đề tài	3
Môi trường phát triển ứng dụng	3
Thông tin thành viên	3
Phân chia công việc	3
<b>Phát biểu bài toán</b>	3
Bài toán	3
So sánh với các giải pháp đang có	4
Cách thiết kế phần mềm	4
<b>Mô hình use-case</b>	6
Danh sách Actor	6
Danh sách các Use-case	6
<b>Sơ đồ lớp mức phân tích</b>	7
<b>Thiết kế giao diện</b>	7
<b>Kết quả thực hiện</b>	8

## 1. Thông tin chung

### 1.1 Tên đề tài

Image Deep Hash (mã hoá sâu hình ảnh).

### 1.2 Môi trường phát triển ứng dụng

- Hệ điều hành: Windows 10 Pro
- Công cụ dùng để xây dựng ứng dụng: PyCharm, VSCode
- Các thư viện đã dùng: Flask, Numpy, OpenCV, Matplotlib, Scikit-learn, Tensorflow

### 1.3 Thông tin thành viên

STT	MSSV	Họ và tên	Điện thoại	Email
1	18120046	Dương Anh Kiệt	0393979123	18120046@student.hcmus.edu.vn
2	18120051	Nguyễn Hoàng Lân	0789416557	18120051@student.hcmus.edu.vn

### 1.4 Phân chia công việc

STT	MSSV	Họ và tên	% đóng góp	Mô tả công việc
1	18120046	Dương Anh Kiệt	100%	Machine learning + Báo cáo
2	18120051	Nguyễn Hoàng Lân	100%	Front-end + Back-end + Báo cáo

## 2. Phát biểu bài toán

### 2.1 Bài toán

- Kiểm tra tính toàn vẹn thông tin là một trong những công dụng quan trọng của hàm hash. Tuy nhiên các hàm hash hiện nay như SHA, MD,... chỉ cho ta biết được tính toàn vẹn của file mà không cho chúng ta biết được mức độ thay đổi của thông tin. Ví dụ: Đối với 1 bức ảnh chỉ cần khác 1 pixels thì kết quả hash sẽ hoàn toàn khác nhau. Trong một số trường hợp như tải một video 60 khung hình/giây và chỉ 1 khung hình sai 1 pixels thì việc tải lại là quá tốn kém. Để khắc phục vấn đề đó chúng em dùng đặc trưng rút trích từ mạng Neural của bức ảnh để đánh giá mức độ khác nhau của bức ảnh tải được và ảnh gốc từ đó đưa ra quyết định có cần thiết tải lại hay không.

- Ngoài ra ta có thể kết hợp cả hàm hash cơ bản như SHA/MD và DeepHash để khó bị tấn công hơn. Vì khi đó muốn tấn công được thì nội dung của ảnh tấn công phải trùng với nội dung của ảnh ban đầu.

## 2.2 So sánh với các giải pháp đang có

	SHA/MD	Deep Hash	SHA/MD + DeepHash
Tốc độ	Nhanh	Chậm	Chậm
Rút trích nội dung	Không	Có	Có
So sánh mức độ thay đổi	Không	Có	Có
Đụng độ (collision)	Khó	Dễ hơn	Rất khó

## 2.3 Cách thiết kế phần mềm

- Dùng mẫu **Strategy** để cho phép người dùng thay đổi backbone model khác ứng với mỗi tên lấy từ input. Ví dụ ở hàm **load\_backbone** hỗ trợ 25 backbones:

```
def load_backbone(name="ResNet50", input_shape=(224, 224, 3), classes=128):
    if name == "Xception":
        return tf.keras.applications.Xception(include_top=True, weights=None, classes=classes,
                                                input_shape=input_shape, classifier_activation='sigmoid')
    if name == "VGG16":
        return tf.keras.applications.VGG16(include_top=True, weights=None, classes=classes,
                                             input_shape=input_shape, classifier_activation='sigmoid')
    if name == "VGG19":
        return tf.keras.applications.VGG19(include_top=True, weights=None, classes=classes,
                                             input_shape=input_shape, classifier_activation='sigmoid')
    if name == "ResNet50":
        return tf.keras.applications.ResNet50(include_top=True, weights=None, classes=classes,
                                                input_shape=input_shape, classifier_activation='sigmoid')
    if name == "ResNet101":
        return tf.keras.applications.ResNet101(include_top=True, weights=None, classes=classes,
                                                 input_shape=input_shape, classifier_activation='sigmoid')
```

- Tuy nhiên bên trong nhóm kế thừa lại ý tưởng của **Flexible Attribute** là có một dictionary các model có sẵn có thể lựa chọn, đồng thời có thể cho người dùng thêm mới vào.

- Đối với model người dùng thêm mới là một file (\*.h5) theo chuẩn **Serialization** và **Deserialization** của Tensorflow.

- Cho phép người dùng thay đổi **metric** (cosine, euclidean distance,...) để tính khoảng cách giữa global feature của 2 bức ảnh.

```
def compare(self, path1, path2, metric="euclidean"):
    self.image1 = self.imageLoader.load(path1)
    self.image2 = self.imageLoader.load(path2)

    predict1 = self.model.predict(np.array([self.image1]))
    predict2 = self.model.predict(np.array([self.image2]))

    if metric == 'hamming':
        return hamming(predict1, predict2)

    return pairwise_distances(predict1, predict2, metric=metric)[0][0]
```

- Về phía lập trình viên, dễ dàng thay đổi cách thức từ global feature thành hash digest. Bằng cách dùng **Message Broker** cách biến đổi từ feature thành hash digest

có thể được linh động thay thế bằng file rule có dạng (.py). (Do không có nhiều thời gian để định nghĩa một cấu trúc message nên nhóm đã dùng luôn cấu trúc của python để dễ dàng gọi lên và thực thi). Việc này giúp cho việc cập nhật trong tương lai dễ dàng hơn.

```
self.flexible_model = {}

def order(self, name="ResNet50", input_shape=(224, 224, 3), classes=128):
    if name in self.flexible_model:
        if self.flexible_model[name].endswith(".py"):
            return self.load_message_broker_model(self.flexible_model[name])

        return self.load_model_from_path(self.flexible_model[name])

    elif name in self.factory:
        return load_backbone(name=name, input_shape=input_shape, classes=classes)

    elif os.path.isfile(name):
        self.flexible_model["custom_model" + str(len(self.flexible_model))] = name
        if name.endswith(".py"):
            return self.load_message_broker_model(name)

        return self.load_model_from_path(name)

    return False

@staticmethod
def load_model_from_path(path):
    return tf.keras.models.load_model(path, compile=False)

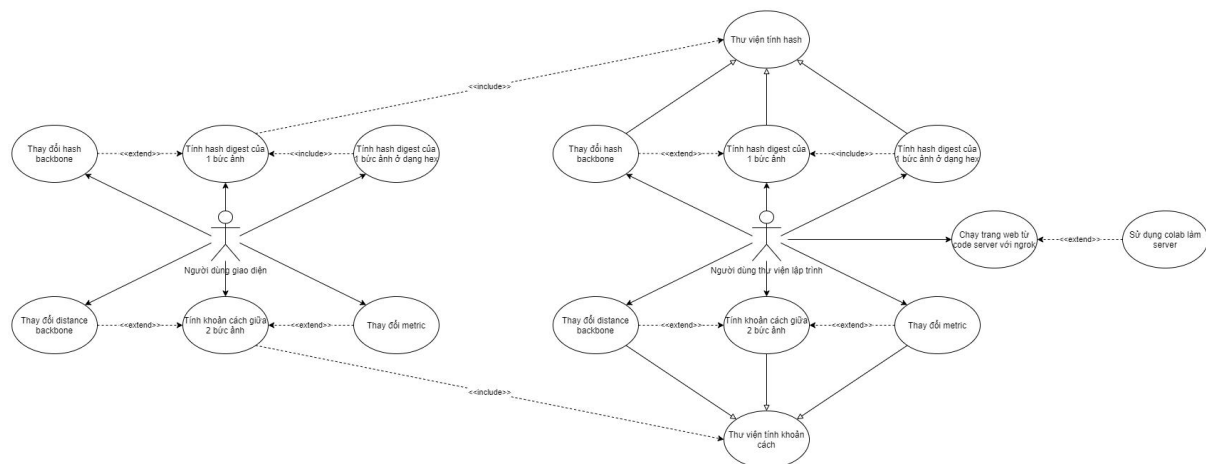
@staticmethod
def load_message_broker_model(path):
    result = {}
    exec(open(path).read(), result)
    return result["model"]
```

- Ngoài ra, nhóm cũng lưu các mô hình dưới dạng biến toàn cục (global variable). Khi khởi động server, mô hình sẽ chỉ được khởi tạo một lần. Chỉ khi có sự thay đổi trong việc lựa chọn mô hình, biến này sẽ được khởi động lại, nếu không mô hình sẽ không được khởi động lại khi có các truy vấn hình ảnh mới, điều này sẽ giảm bớt phần nào thời gian xử lý.

- Ứng dụng cung cấp một nền tảng tính hash và khoảng cách của bức ảnh ở 2 mức. Mức thư viện người dùng có thể import và dùng như một thư viện. Ở mức ứng dụng người dùng có thể thực thi trên trang web.



### 3. Mô hình use-case



#### 3.1 Danh sách Actor:

STT	Tên Actor	Ý nghĩa/Ghi chú
1	Người dùng giao diện	Người dùng ở mức giao diện, giao tiếp với ứng dụng qua trang web.
2	Người dùng thư viện lập trình	Người dùng ở mức thư viện, giao tiếp với ứng dụng thông qua code. Dùng ứng dụng như một thư viện trong một dự án khác.

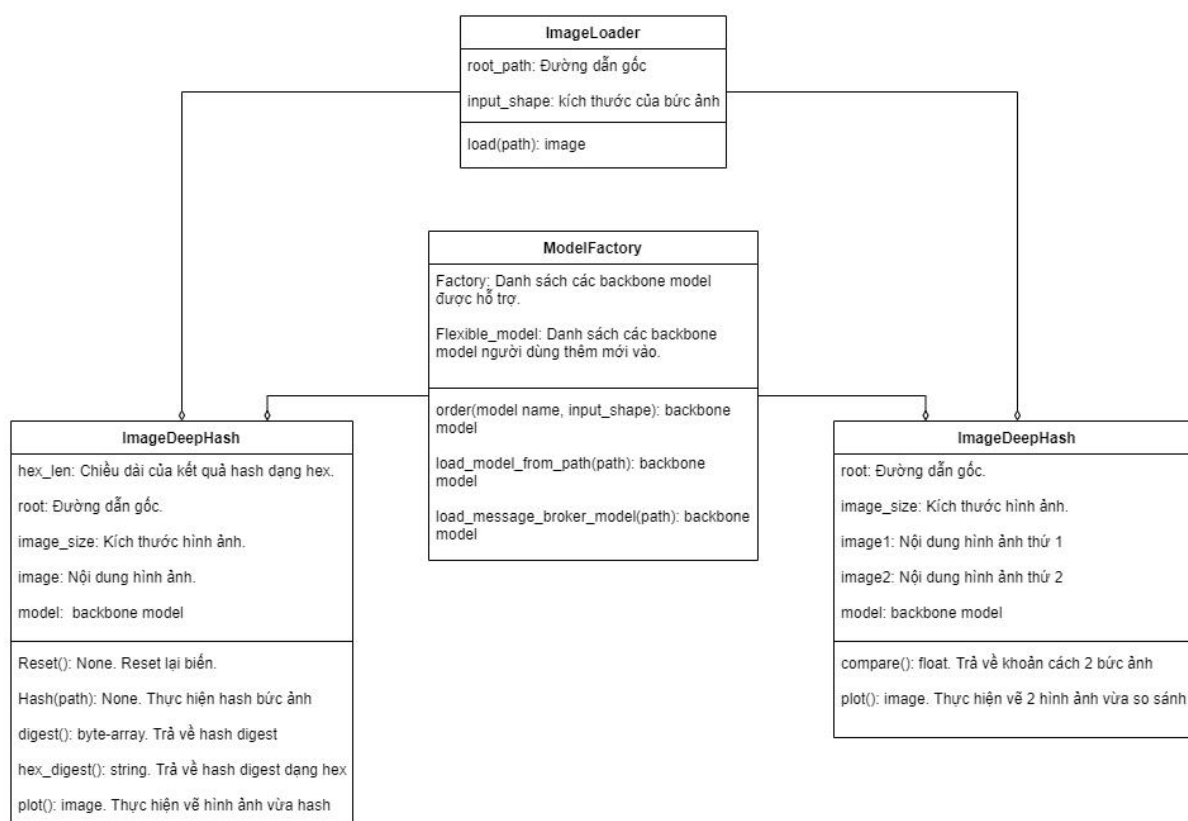
○

#### 3.2 Danh sách các Use-case

STT	Tên Use-case	Ý nghĩa/Ghi chú
1	Tính hash digest của bức ảnh (mức giao diện)	Người dùng upload ảnh lên, server trả về hash digest của bức ảnh.
2	Tính hash digest của bức ảnh (mức thư viện lập trình)	Người dùng gọi hàm của thư viện và truyền tên file ảnh vào. Thư viện tính toán và trả về hash digest.
3	Tính hash digest của bức ảnh ở định dạng hex (mức giao diện)	Người dùng upload ảnh lên, server trả về hash digest của bức ảnh ở dạng hex.
4	Tính hash digest của bức ảnh ở định dạng hex (mức thư viện lập trình)	Người dùng gọi hàm của thư viện và truyền tên file ảnh vào. Thư viện tính toán và trả về hash digest ở dạng hex.
5	Tính khoảng cách giữa 2 bức ảnh (mức giao diện)	Người dùng upload 2 bức ảnh lên, server trả về độ giống nhau của 2 bức ảnh.
6	Tính khoảng cách giữa 2 bức ảnh (mức thư viện lập trình)	Người dùng gọi hàm của thư viện và truyền vào đường dẫn 2 file ảnh. Thư viện tính toán và trả về độ giống nhau của 2 bức ảnh.
7	Thay đổi hash backbone model (mức giao diện)	Người dùng chọn backbone model trên trang web.
8	Thay đổi hash backbone model (mức thư viện lập trình)	Người dùng truyền tên backbone model hoặc đường dẫn đến file model vào hàm của thư viện.

	trình)	
9	Thay đổi distance backbone (mức giao diện)	Người dùng chọn backbone model trên trang web.
10	Thay đổi distance backbone (mức thư viện lập trình)	Người dùng truyền tên backbone model hoặc đường dẫn đến file model vào hàm của thư viện.
11	Thay đổi metric (mức giao diện)	Người dùng chọn metric trên trang web.
12	Thay đổi metric (mức thư viện lập trình)	Người dùng truyền tên metric vào hàm của thư viện.
13	Chạy trang web từ code server với ngrok.	Người dùng mức thư viện có thể thực hiện chạy code server để tạo một trang web.

## 4. Sơ đồ lớp mức phân tích



## 5. Thiết kế giao diện

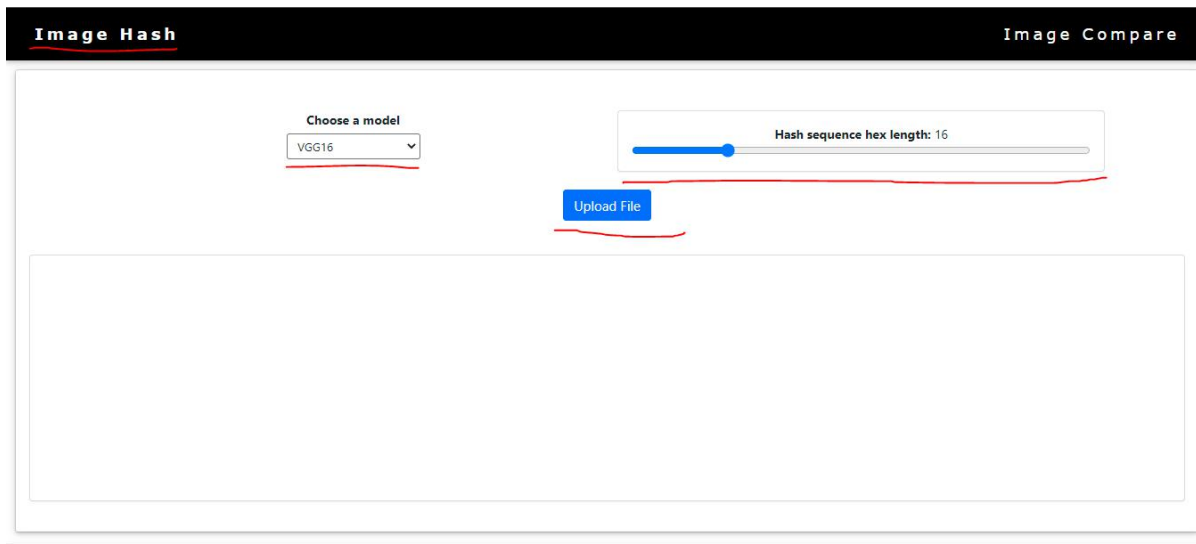
Giao diện của chương trình bao gồm 2 trang: **Image Hash** và **Image Compare**. Chúng ta có thể nhấn vào một trong hai nút trên thanh menu để điều hướng đến trang tương ứng.

1. Trang **Image Hash**: bao gồm các thành phần để tương tác với người dùng như sau:
  - **Backbone type**: Ta có thể chọn 1 trong 25 backbones có sẵn trong ô để tiến



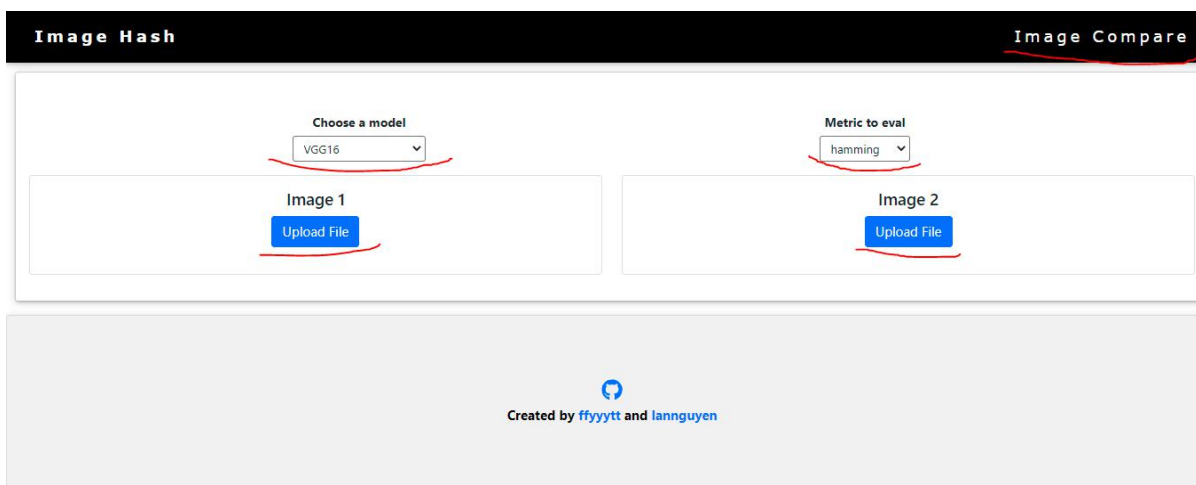
hành rút trích đặc trưng nội dung ảnh (global feature).

- **Hash sequence length:** Độ dài chuỗi hash dạng hexa được chuyển hóa từ global feature của ảnh.
- **Upload file button:** Dùng để tải file lên (chỉ hỗ trợ ảnh)



## 2. Trang Image Compare:

- **Backbone type:** Rút trích global feature của 2 ảnh.
- **Metric:** Khoảng cách giữa 2 global feature của 2 ảnh.
- **Upload file buttons:** Tải lần lượt 2 ảnh lên giao diện.



## 6. Kết quả thực hiện

### ● Thiết lập môi trường:

- Phiên bản **Python** được sử dụng là **3.9** và **Tensorflow** là **2.7.0**. Ngoài ra cần cài đặt thêm các thư viện được liệt kê trong tệp **requirements.txt**. Để có thể chạy server trên máy cục bộ, cấu hình đề nghị là GPU với bộ nhớ ít nhất 3GB, hoặc có thể sử dụng

dùng ngrok để chạy server trên Google Colab. Để sử dụng ngrok cần phải tạo một tài khoản miễn phí trên trang chủ và sử dụng token xác thực được cung cấp. Ở đây nhóm có chuẩn bị sẵn notebook để có thể chạy thử trên Google Colab một cách nhanh chóng và tiện lợi:

[https://colab.research.google.com/drive/1N2AyKf\\_G8lmdRdpqgzLbyQf0PgH7jwlt?usp=sharing](https://colab.research.google.com/drive/1N2AyKf_G8lmdRdpqgzLbyQf0PgH7jwlt?usp=sharing)


- Kết quả chạy thử:

- Image hash

Choose a model  
VGG16

Hash sequence hex length: 16

Upload File



Ta chạy thử ảnh đầu tiên với các input là mặc định



Hash Result

0xf62dbea3fd5d7879049e710150c6e2b7


Ta thấy kết quả trả ra là 1 hash sequence có độ dài là 16 như mặc định. Ta đã hoàn thành xong việc deep hash một tấm ảnh

Choose a model  
MobileNetV2

Hash sequence hex length: 24

Upload File

4E0744CD-793A-4EF8-B550B54F7F2C4406.jpg



Ta thử thay backbone là MobileNetV2 và đổi độ dài chuỗi hash thành 24



Hash Result

0x762db6a2d55d787d06de2061d2d682f7

Kết quả trả ra là chuỗi hash có độ dài 24

## • Image Compare

Choose a model

ResNet50

Metric to eval

hamming

Image 1

Upload File

trump.jpg





Image 2

Upload File

5c9a115d8e436a63e42c2883.jpg



Ta thử chạy với 2 bức ảnh đầu tiên và input là ResNet50 cùng với metric để đánh giá là Hamming Distance



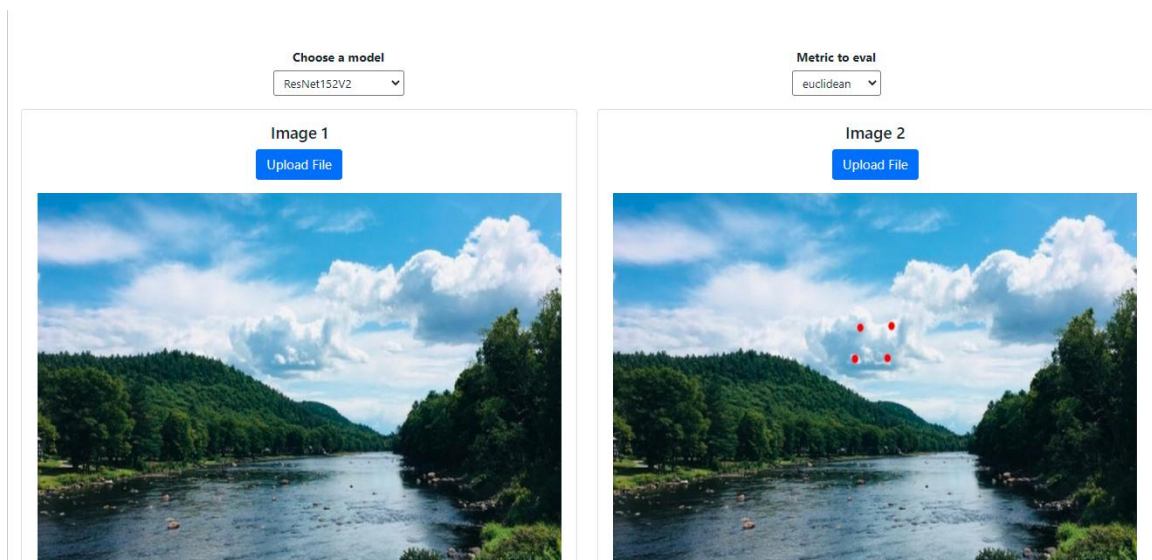


Compare result with hamming metric

**0.99951171875**

The closer the value to 0, the more similar the images

Kết quả so sánh là gần với 1. Điều đó cho thấy 2 bức ảnh là gần hoàn toàn khác nhau.



Ta thử nghiệm với 2 ảnh gần giống nhau. Ở đây sự khác biệt chỉ là 4 chấm đỏ



Compare result with euclidean metric

**0.0007470433**

The closer the value to 0, the more similar the images

Với metric được sử dụng là Euclidean Distance, ta thấy được giá trị so sánh xấp xỉ với 0. Điều đó cho thấy 2 bức ảnh là gần hoàn toàn giống nhau.