TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

ĐẠI HỌC QUỐC GIA HÀ NỘI

KHOA TOÁN - CƠ - TIN HỌC

BÁO CÁO PROJECT

ĐỀ TÀI: XÂY DỰNG WEBSITE XỬ LÝ ẢNH

Giảng viên hướng dẫn : Thầy Ngô Thế Quyền

Sinh viên thực hiện : Phạm Văn Tĩnh – 19000368

Nguyễn Văn Mạnh – 19000359

Trần Đức Việt – 19000379

Lớp : MAT3376 1

Hà Nội, tháng 01 năm 2022

LỜI NÓI ĐẦU

Deep Learning đã và đang là một chủ đề AI được bàn luận sôi nổi. Là một phạm trù nhỏ của Machine Learning, Deep Learning tập trung giải quyết các vấn đề liên quan đến mạng thần kinh nhân tạo nhằm nâng cấp các công nghệ như nhận diện giọng nói, thị giác máy tính và xử lý ngôn ngữ tự nhiên. Deep Learning đang trở thành một trong những lĩnh vực hot nhất trong khoa học máy tính.

Chỉ trong vài năm, Deep Learning đã thúc đẩy tiến bộ trong đa dạng các lĩnh vực như nhận thức sự vật (object perception), dịch tự động (machine translation), nhận diện giọng nói, … – những vấn đề từng rất khó khăn với các nhà nghiên cứu trí tuệ nhân tạo. Để được tiếp cận, tìm hiểu và áp dụng vào thực tế kiến thức cơ bản về Deep Learning nói chung và Computer Vision (Thị giác máy tính) nói riêng. Với mục tiêu cơ bản như trên, nhóm chúng em chọn đề tài "Xây dựng Website xử lý ảnh" sử dụng công nghệ "*Deep Learning*". Ảnh khi được đưa vào chương trình sẽ được chuyển đổi thành ảnh kỹ thuật số (Digital) hoặc nhận biết được đối tượng có trong hình ảnh đó.

Với sự hướng dẫn của thầy Ngô Thế Quyền nhóm chúng em đã hoàn thành bài báo cáo đề tài này. Trong quá trình tìm hiểu, phân tích thiết kế và cài đặt hệ thống, nhóm chúng em không thể tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự thông cảm và góp ý của thầy. Chúng em xin chân thành cảm ơn.

MỤC LỤC

Phần 1. Mở đầu

1.1 Lý do chọn đề tài

Xử lý ảnh bao gồm lý thuyết và các kỹ thuật liên quan nhằm mục đích tạo ra một hệ thống nhân tạo có thể tiếp nhận thông tin từ các hình ảnh thu được hoặc các tập dữ liệu đa chiều. Đối với mỗi người chúng ta, quá trình "học" thông qua quá trình sống của mỗi người. Tuy nhiên với các vật vô tri vô giác như máy tính, robot… thì điều đó quả thực là một bước tiến rất khó khăn. Các thiết bị ngày nay không chỉ nhận thông tin ở dạng tín hiệu đơn lẻ mà còn có thể có cái "nhìn" thật với thế giới bên ngoài. Cái "nhìn" này qua quá trình phân tích, kết hợp với các mô hình như học máy, mạng nơron… sẽ giúp cho thiết bị tiến dần tới một hệ thống nhân tạo có khả năng ra quyết định linh hoạt và đúng đắn hơn nhiều.

Nhận diện hình ảnh hoặc thị giác máy tính là một kỹ thuật tìm kiếm các cách để tự động hóa tất cả công việc mà một hệ thống thị giác của con người có thể làm. Hãy nhắc đến những cái tên như TensorFlow của Google, DeepFace của Facebook, Dự án Oxford của Microsoft. Chúng đều là những ví dụ tuyệt vời về hệ thống nhận diện hình ảnh học sâu.

Chính vì những lý do trên nhóm chúng em đã lựa chọn đề tài "Xây dựng Website xử lý ảnh".

1.2 Chức năng

- Nhận diện đối tượng trong ảnh, ví dụ như: người, chó, mèo, ô tô, …
- Chuyển đổi sang ảnh digital

1.3 Hướng giải quyết

- Đối với chức năng nhận diện đối tượng: sử dụng thuật toán Faster-RCNN
- Đối với chức năng chuyển đổi sang ảnh digital: sử dụng thuật toán GANs

1.4 Ý nghĩa

Hướng đến mục tiêu hỗ trợ người khiếm thị và mù màu trong vấn đề về nhận diện người thân, vật thể, chuyển đổi một số vùng màu mà người mù màu khó nhận biết sang vùng màu dễ nhận biết, … (Chỉ dừng lại ở giải pháp hỗ trợ phần mềm xác định đối tượng trên một bức ảnh và chuyển ảnh sang ảnh kỹ thuật số từ một bức ảnh cố định).

Phần 2. Nội dung

2.1 Phân tích thuật toán

2.1.1 Faster-RCNN

Luồng xử lý của Faster-RCNN có thể được tóm gọn lại như sau:

- Input image được đưa qua 1 backbone CNN, thu được feature map.
- Một mạng con RPN (gồm các conv layer) dùng để trích ra các vùng gọi là RoI (Region of Interest), hay các vùng có khả năng chứa đối tượng từ feature map của ảnh. Input của RPN là feature map, output của RPN bao gồm 2 phần: binary object classification (để phân biệt đối tượng với background) và bounding box regression (để xác định vùng ảnh có khả năng chứa đối tượng), vậy nên RPN bao gồm 2 hàm loss.
- Faster-RCNN còn định nghĩa thêm khái niệm anchor, tức các predefined boxes đã được định nghĩa trước lúc huấn luyện mô hình với mục đích như sau: thay vì mô hình phải dự đoán trực tiếp các tọa độ offset của bounding box sẽ rất khó khăn thì sẽ dự đoán gián tiếp qua các anchors.
- Sau khi đi qua RPN, ta sẽ thu được vùng RoI với kích thước khác nhau (W & H) nên sử dụng RoI Pooling để thu về cùng 1 kích thước cố định.
- Thực hiện tách thành 2 nhánh ở phần cuối của model, 1 cho object classification với N + 1 (N là tổng số class, +1 là background) với bounding box regression. Vậy nên sẽ định nghĩa thêm 2 thành phần loss nữa (khá tương tự như RPN) và loss function của Faster-RCNN sẽ gồm 4 thành phần: 2 loss của RPN, 2 loss của Fast-RCNN

Hàm mất mát

Hàm mất mát của model Faster-RCNN gồm 4 thành phần:

- RPN classification (binary classification, object or background)
- RPN regression (anchor -> region proposal)
- Fast-RCNN classification (N+1 classes)
- Fast-RCNN bounding box regression (region proposal -> ground truth bounding box)

Hàm mất mát sẽ được định nghĩa như công thức bên dưới:

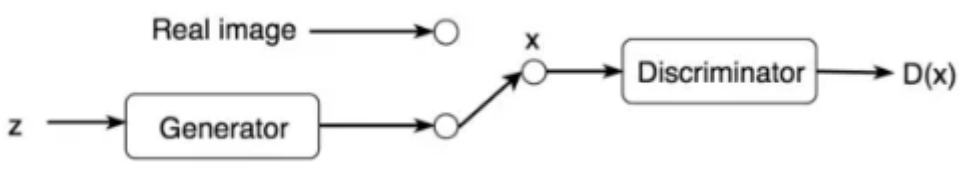
$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*)$$

Chú thích:

- i là index của anchor trong 1 mini-bath.
- p_i là xác suất sự đoán cho anchor tại index thứ i là 1 đối tượng.
- p_i^* là giá trị nhãn ground-truth bằng 1 nếu anchor là positive, và bằng 0 khi anchor là negative.
- t_i là một vector biểu diễn toạ độ của bounding box đã được dự đoán.
- t_i^* là biểu diễn tọa độ của ground truth bounding box tương ứng với positive anchor (xác định positive hay negative bằng IoU là độ đo để biểu diễn độ tương đồng giữa ground truth bounding box với bounding box).
- L_{cls} để xem anchor có chứa đối tượng hay không.
- L_{reg} là loss tính bounding box regression.

2.1.2 GANs

GANs được kết hợp từ 2 model là: Generator(G) và Discriminator(D), và triển khai theo như hình dưới:



Chú thích:

- Generator: tạo ra các ảnh giả, gần giống với real image, để đánh lừa D, có đầu vào là tập hợp các vector nhiễu z.
- Discriminator: kiểm tra ảnh mà G tạo ra, phân biệt nó là thật hay giả.
- z: Nhiễu (được sinh ngẫu nhiên từ 1 phân phối xác suất).
- x: ảnh thật.

Cụ thể thuật toán GANs như sau:

Bước 1: Từ một nhiễu z bất kì, G sinh ra ảnh giả G(z) có kích thước như ảnh thật.

Bước 2: x và G(z) cùng được đưa vào D kèm nhãn đúng sai. Train D để học khả năng phân biệt ảnh thật, ảnh giả.

Bước 3: Đưa G(z) vào D, dựa vào feedback của D trả về, G sẽ cải thiện khả năng fake của mình.

Bước 4: Quá trình trên sẽ lặp đi lặp lại như vậy, D dần cải thiện khả năng phân biệt, G dần cải thiện khả năng fake. Đến khi nào D không thể phân biệt được ảnh nào là ảnh do G tạo ra, ảnh nào là x, khi đó quá trình dừng lại.

Hàm mất mát

Hàm mất mát cho G & D:

$$L(G, D) = \omega_{adv}L_{adv}(G, D) + \omega_{con}L_{con}(G, D) + \omega_{gra}L_{gra}(G, D) + \omega_{col}L_{col}(G, D)$$

Trong đó:

- $L_{adv}(G, D)$ là adversarial loss dùng để chuyển đổi sang ảnh digital trong Generator.
- $L_{con}(G, D)$ là content loss giúp khi chuyển đổi ảnh vẫn giữ lại được content của ảnh đầu vào.

- $L_{gra}(G, D)$ là gray style loss giúp cho việc chuyển đổi ảnh digital có thể rõ ràng, có đường nét.
- $L_{col}(G, D)$ là color reconstruction loss giúp cho ảnh chuyển đổi lấy được màu gốc của ảnh input.
- w_{adv}, w_{con}, w_{gra}, w_{col} là các trọng số để cân bằng các hàm mất mát.

Hàm mất mát cho G:

$$L(G) = \omega_{adv}E_{pi\sim S_{data}(p)}[(G(p_i) - 1)^2] + \omega_{con}L_{con}(G, D) + \omega_{gra}L_{gra}(G, D) + \omega_{col}L_{col}(G, D)$$

Hàm mất mát cho D:

$$L(D) = w_{adv}[E_{ai\sim S_{data}(a)}[D(a_i) -1)^2] + E_{pi\sim S_{data}(p)}[(D(G(p_i)))^2] + E_{xi\sim S_{data}(x)}[(D(x_i))^2] + 0.2E_{yi\sim S_{data}(y)}[(D(y_i))^2]]$$

2.2 Xây dựng chương trình

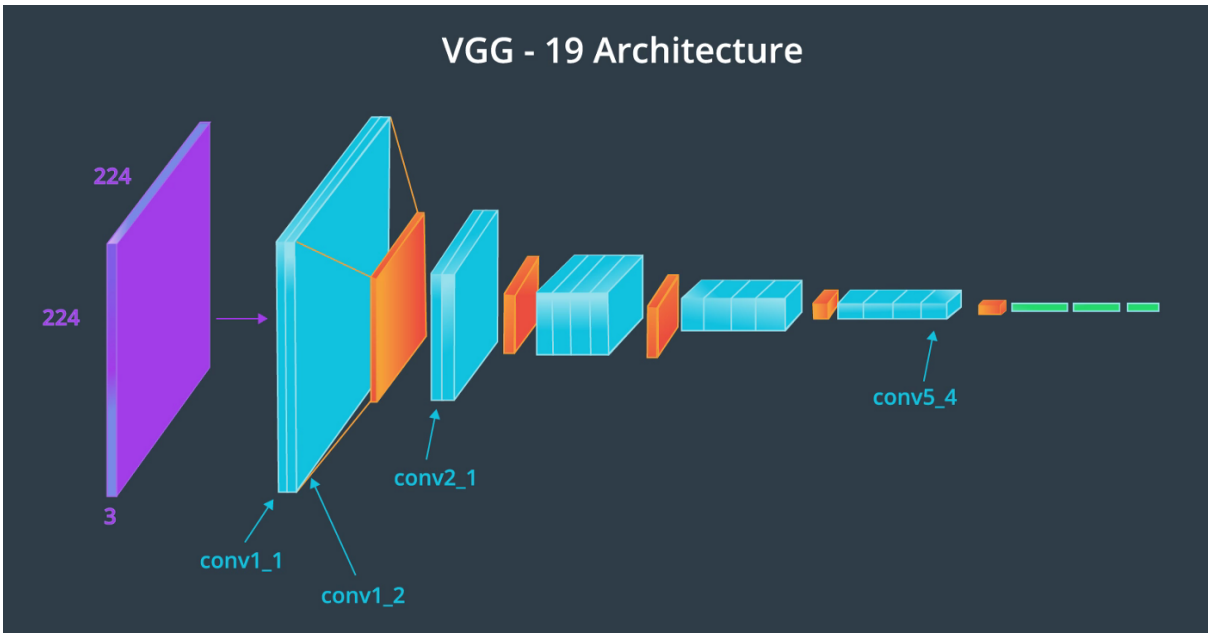
2.2.1 Các công nghệ đã sử dụng

- ❖ Các thư viện Python
  - Pytorch (Documents)
  - Numpy (Documents)
  - Tqdm (Documents)
  - Opencv (Documents)
  - Streamlit (Documents)
  - Argparse (Documents)
- ❖ Công nghệ
  - Docker (Documents)
  - Cog (Documents)
  - Detectron2 (Documents)
  - Deep Learning

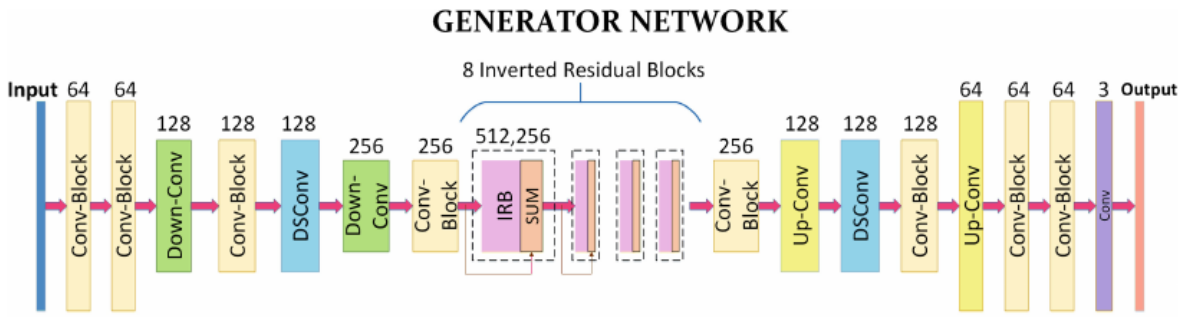
2.2.2 Quá trình xây dựng chương trình:

❖ GANs:

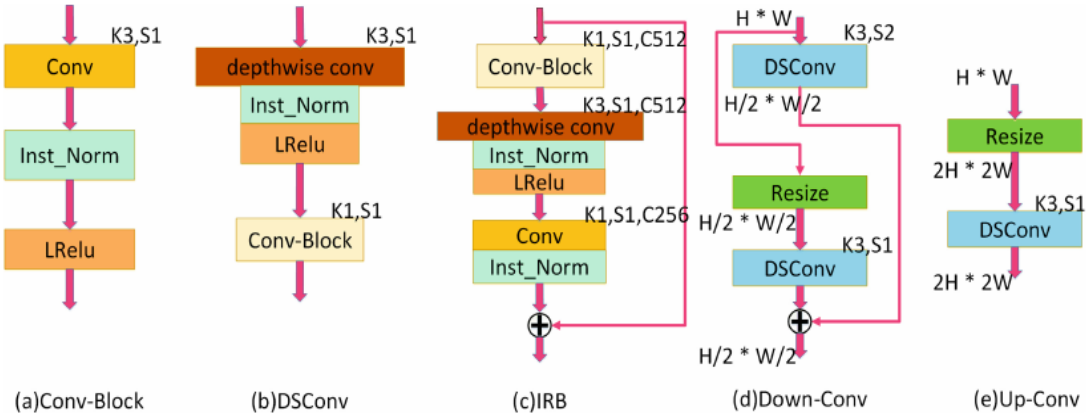
Xuất phát ban đầu là ảnh cần được xử lý qua CNN để lấy feature map. Trong ứng dụng mà nhóm em đã xây dựng, nhóm sử dụng mạng VGG 19:



Sau khi có được Feature Map, tiếp theo nhóm xây dựng mạng Generator để tạo ảnh giả. Mạng Generator có cấu trúc như sau:



Trong đó các khối block có cấu trúc như sau:



Generator có thể được coi là một mạng encoder-decoder, trong đó bao gồm chủ yếu là convolutions, separable convolutions, inverted residual blocks, unsampling và downsampling. Ở trong Generator, lớp chập với kernels 1x1 không sử dụng normalization layer và sử dụng hàm tanh làm hàm kích hoạt. Conv-Block bao gồm một tích chập tiêu chuẩn với kernels 3x3, lớp chuẩn hóa (instance normalization layer) và hàm kích hoạt LRelu. DSConv bao gồm separable convolutions với kernels 3x3, lớp chuẩn hóa, và hàm kích hoạt LRelu. Inverted Residual Block bao gồm Conv-Block, depth_wise convolutions, và lớp chuẩn hóa. Để tránh mất thông tin đối tượng bởi max-pooling, Down-Conv dùng để giảm độ phân giải của feature maps. Nó bao gồm DSConv với stride 2 và DSConv với stride 1. Ở Down-Conv, feature maps được thay đổi kích thước bằng 1 nửa feature maps đầu vào. Đầu ra của Down-Conv là tổng đầu ra của DSConv với stride 2 và DSConv với stride 1. Up-Conv dùng để tăng độ phân giải của feature maps. Ở Up-Conv, feature maps được thay đổi kích thước bằng 2 lần feature maps đầu vào. Để giảm thiểu số lượng các thông số của generator, chúng ta sử dụng 8 inverted residual block liên tiếp ở giữa của mạng.

❖ *Implement code Python như sau:*

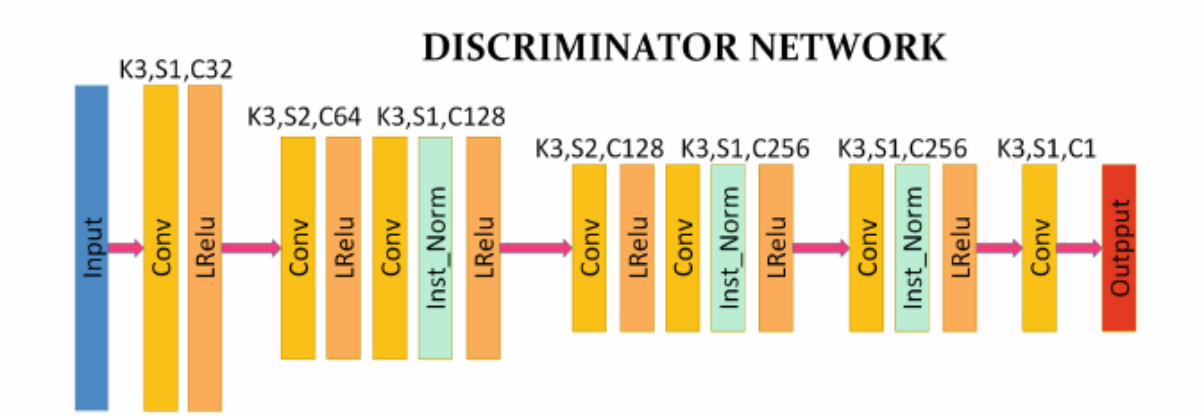
```
class Generator(nn.Module):

    def __init__(self, dataset=''):

        super(Generator, self).__init__()
```

```
        self.name = f'generator_{dataset}'

        bias = False


        self.encode_blocks = nn.Sequential(
            ConvBlock(3, 64, bias=bias),
            ConvBlock(64, 128, bias=bias),
            DownConv(128, bias=bias),
            ConvBlock(128, 128, bias=bias),
            SeparableConv2D(128, 256, bias=bias),
            DownConv(256, bias=bias),
            ConvBlock(256, 256, bias=bias),
        )


        self.res_blocks = nn.Sequential(
            InvertedResBlock(256, 256, bias=bias),
            InvertedResBlock(256, 256, bias=bias),
            InvertedResBlock(256, 256, bias=bias),
            InvertedResBlock(256, 256, bias=bias),
            InvertedResBlock(256, 256, bias=bias),
            InvertedResBlock(256, 256, bias=bias),
            InvertedResBlock(256, 256, bias=bias),
            InvertedResBlock(256, 256, bias=bias),
        )
```

```
            self.decode_blocks = nn.Sequential(
                ConvBlock(256, 128, bias=bias),
                UpConv(128, bias=bias),
                SeparableConv2D(128, 128, bias=bias),
                ConvBlock(128, 128, bias=bias),
                UpConv(128, bias=bias),
                ConvBlock(128, 64, bias=bias),
                ConvBlock(64, 64, bias=bias),
                nn.Conv2d(64,3,  kernel_size=1,  stride=1,padding=0,
bias=bias),
               nn.Tanh(),
            )

            initialize_weights(self)

        def forward(self, x):
            out = self.encode_blocks(x)
            out = self.res_blocks(out)
            img = self.decode_blocks(out)

            return img
```

Generator có đầu ra là ảnh fake, ảnh fake này sẽ được kiểm tra bởi Discriminator. Discriminator sẽ cố gắng phân biệt ảnh fake được sinh ra bởi Generator và ảnh gốc.

Discriminator có cấu trúc như sau:



Discriminator sử dụng lớp chập tiêu chuẩn. Với mỗi weight của mỗi lớp chập, sử dụng spectral normalization để giúp việc training ổn định hơn.

```
class Discriminator(nn.Module):

    def __init__(self,  args):

        super(Discriminator, self).__init__()

        self.name = f'discriminator_{args.dataset}'

        self.bias = False

        channels = 32



        layers = [

           nn.Conv2d(3,channels,kernel_size=3,stride=1,padding=1,
bias=self.bias),

           nn.LeakyReLU(0.2, True)
```

```
        ]

        for i in range(args.d_layers):

            layers += [

                nn.Conv2d(channels,  channels  *  2,  kernel_size=3,
stride=2, padding=1, bias=self.bias),

                nn.LeakyReLU(0.2, True),

                nn.Conv2d(channels*2,channels*4,kernel_size=3,
stride=1, padding=1, bias=self.bias),

                nn.InstanceNorm2d(channels * 4),

                nn.LeakyReLU(0.2, True),

            ]

            channels *= 4



        layers += [

            nn.Conv2d(channels,  channels,  kernel_size=3,  stride=1,
padding=1, bias=self.bias),

            nn.InstanceNorm2d(channels),

            nn.LeakyReLU(0.2, True),

            nn.Conv2d(channels,    1,    kernel_size=3,    stride=1,
padding=1, bias=self.bias),

        ]



        if args.use_sn:

            for i in range(len(layers)):
```

```
            if isinstance(layers[i], nn.Conv2d):
                layers[i] = spectral_norm(layers[i])

        self.discriminate = nn.Sequential(*layers)

        initialize_weights(self)

    def forward(self, img):
        return self.discriminate(img)
```

Khi đã có mô hình của hai mạng Generator và Discriminator, tiếp theo nhóm xây dựng chương trình tính toán Loss Function với công thức đã được trình bài ở mục *2.1.2.*

Sau đó, nhóm xây dựng thêm những hàm chức năng phục vụ cho quá trình training cũng như test model: *common.py* và *image_processing.py*

Trong chương trình common, nhóm xây dựng một số hàm liên quan đến quá trình train:

- *read_image():* đọc ảnh từ đường dẫn.
- *save_checkpoint():* lưu lại file checkpoint khi quá trình training kết thúc.
- *load_checkpoint(), load_weight():* để load weight đã thu được sau quá trình training.
- *initialize_weight(): khởi tạo trọng số.*
- *set_lr:* thiết lập hệ số learning rate *(hệ số học máy).*

Trong chương trình image_processing.py, nhóm xây dựng các hàm giúp hỗ trợ tiền xử lý hình ảnh:

- *gram():* tính toán ma trận Gram.
- *resize_image():* sửa đổi kích thước ảnh.
- *normalize_image(), denormalize_image().*

- *compute_data_mean():* để tính giá trị trung bình và convert ảnh từ YUV sang RGB chuẩn bị cho quá trình training.

Trước khi xây dựng chương trình cho quá trình training, nhóm xây dựng thêm chương trình edge_smooth để tạo ảnh noise phục vụ quá trình tính toán compute mean.

Tập training set cũng cần được xây dựng một cách kỹ lưỡng trước khi bước vào quá trình training. Tập training của nhóm bao gồm 6649 có kích thước 256*256 được thu thập từ internet và một số ảnh của thành viên nhóm. Bên cạnh đó, nhóm còn xây dựng tập Hayao từ nguồn internet để giúp tạo ảnh noise và tính toán compute mean.

Cuối cùng là chương trình *train.py* và *inference_image.py* để training model và thực thiện transform ảnh sau khi đã có được weight từ quá trình trainning.

Nhóm cũng triển khai code lên google colab giúp thuận tiện hơn trong quá trình training đối với laptop hay pc có cấu hình yếu.

❖ Detection:

Trong quá trình tìm hiểu thuật toán Faster-RCNN cho ứng dụng detection, nhóm có tìm hiểu được một platform hỗ trợ khá đầy đủ các công cụ phục vụ quá trình training cũng như get output cho phần Faster-RCNN: "*Detectron 2*" - Facebook. Được xây dựng và hỗ trợ bởi Facebook (hiện tại là Meta) một trong những công ty công nghệ hàng đầu thế giới nên nhóm quyết định sử dụng nền tảng này. Nhóm xây dựng hai chương trình detection.py và get_detection.py giúp thực hiện quá trình training cũng như get output.

❖ Tích hợp Docker và Cog:

Cuối cùng nhóm đóng gói lại toàn bộ những chương trình trên và thực hiện xây dựng image bằng Docker và Cog về phía server.

Cog (Container for machine learning) là một công cụ cung cấp một môi trường nhất quán để chạy mô hình, training hoặc chia sẻ cho nhiều người cùng làm việc với model. Sau đó khi model đã được training, Cog hỗ trợ deploy ứng dụng lên Docker và các HTTP API tiêu chuẩn.

Cog cũng hộ trợ xây dựng chương trình dự doán bằng python, giúp việc lấy kết quả đầu ra từ model trở lên dễ dàng hơn.

❖ **Xây dựng client bằng streamlit:**

Về phía client, nhóm quyết định sử dụng thư viện streamlit để xây dựng giao diện cơ bản, do thư viện này hỗ trợ mạnh mẽ cho việc triển khai các ứng dụng machine learning lên nền tảng web. Nhóm tạo ra giao diện cơ bản để người dùng có thể chuyển đổi ảnh một cách dễ dàng, ngoài ra còn có thêm tính năng tải ảnh về máy nếu người dùng muốn lưu trữ bức ảnh sau khi đã được xử lý.

2.3 Hướng dẫn cài đặt chương trình

Ứng dụng được xây dựng tập trung trên hệ điều hành Unix (Linux + Mac OS) nên đối với hệ điều hành Window nhóm em khuyến nghị sử dụng WSL2 (Window Subsystem for Linux) để cài đặt chương trình.

Quá trình training cũng có thể chạy trên Google Colab

❖ Đầu tiên, thực hiện cài đặt thư viện python được sử dụng trong ứng dụng:

```
pip3 install -r requiments.txt
```

❖ Thiết lập Docker và Cog

- Docker: (Install)
- Cog:

```
sudo curl -o /usr/local/bin/cog -L
https://github.com/replicate/cog/releases/latest/download/cog_`uname -
s`_`uname -m`
sudo chmod +x /usr/local/bin/cog
```

❖ Xây dựng image Docker using Cog

```
cd ./server
sudo cog build -t tvm-com-vision
```

❖ Run client

```
cd ../client && streamlit run app.py
```

Phần 3. Tổng kết

3.1 Kết quả đã đạt được

Sau thời gian thực hiện nhóm chúng em đã hoàn thiện hệ thống theo đúng tiến độ và yêu cầu đặt ra.

Tìm hiểu hệ thống kỹ càng chính xác. Phân tích rõ được :

- Bài toán đặt ra.
- Hiện trạng hệ thống và yêu cầu người sử dụng.
- Các chức năng cần có của hệ thống bao gồm:
    - Chuyển đổi ảnh sang ảnh digital.
    - Nhận biết được vật thể có trong bức ảnh .

Giao diện thân thiện, dễ sử dụng.

Đưa hệ thống vào thử nghiệm thành công.

3.2 Những điểm còn hạn chế

- Tốc độ training chậm.
- Kết quả khi mà chuyển đổi ảnh một vài vùng chưa lấy lại được độ nét như cũ.
- Chưa xây dựng được nhiều chức năng trên web.

3.3 Hướng phát triển

Hi vọng rằng trong thời gian tới, nhóm chúng em sẽ có thể tiếp tục nghiên cứu, phân tích và thiết kế hệ thống để hệ thống phát triển thêm nhiều chức năng, tiếp cận đến được người dùng và phát triển hơn nữa.

TÀI LIỆU THAM KHẢO

Thuật toán GAN

Thuật toán Faster-RCNN

Source Code tham khảo

BÁO CÁO PROJECT NHÓM 15

- Nhóm gửi file đúng thời gian giảng viên đã định.
- Nhóm gửi đủ chương trình và hướng dẫn cài đặt, chương trình đã hoạt động theo hướng dẫn nhóm đưa ra.
- **Lỗi :**
    - Khi chưa select vào 1 ảnh nào đó mà bấm vào button xóa thì chương trình vẫn báo xóa thành công.
    - Chương trình chưa tự reload lại bảng hiển thị file ảnh khi thêm 1 file ảnh mới.
- **Hạn chế và cần cải thiện :**
    - Chức năng thêm sửa xóa chưa thực sự hoàn thiện.
    - Chưa có Regex để kiểm tra các input khi nhập vào (Email, số điện thoại, ID hình ảnh).
    - Ngày có thể dùng hàm để lấy thời gian thực chứ không cần người dùng nhập bằng tay.
    - Mới chỉ có 2 danh mục phân loại, người dùng không thể thêm danh mục phân loại.
    - Định dạng ảnh input cũng mới chỉ cho phép ảnh JPG và PNG.
    - Mục định dạng ảnh chương trình chưa tự lấy định dạng ảnh được, mà phải do người dùng tự chọn.
    - Quản lý ảnh nhưng chưa hiển thị ra được hình ảnh xem trước.