ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

Xử lý ngôn ngữ tự nhiên

Visual Question Answering VQA

Học kỳ 232

GVHD: Võ Thanh Hùng
SV: Nguyễn Thái Tân 2112256
Nguyễn Phúc Minh Quân 2110479
Hồ Trọng Nhân 2111899

TP. HỒ CHÍ MINH, THÁNG 3/2024

Mục lục

1 Giới thiệu bài toán VQA

Bài toán trả lời câu hỏi trực quan (Visual Question Answering) kết hợp hai lĩnh vực quan trọng của học máy (Machine Learning) là thị giác máy tính (Computer Vision) và xử lý ngôn ngữ tự nhiên (Natural Language Processing). Dựa vào một hình ảnh và một câu hỏi ngôn ngữ tự nhiên về hình ảnh đó, mô hình phải đưa ra một câu trả lời tương ứng bằng ngôn ngữ tự nhiên.

Do câu hỏi có thể tập trung vào các vùng khác nhau của hình ảnh (tiền cảnh – foreground, hậu cảnh – background, ngữ cảnh – context hoặc các chi tiết khác) nên đòi hỏi mô hình vừa phải nhận biết được các bộ phận của ảnh, vừa phải kết hợp các bộ phận đó với câu hỏi và suy luận ra câu trả lời.

Hình 1: *Ví dụ về VQA*

Để huấn luyện một mô hình trả lời được những câu hỏi phức tạp như trên, cần những bộ dữ liệu vô cùng lớn về cả hình ảnh lẫn câu hỏi - câu trả lời.

Bộ dữ liệu của `https://visualqa.org/` có thể nói là bộ dữ liệu phổ biến nhất cho mô hình VQA, với 265,016 hình ảnh, mỗi hình ảnh có ít nhất 3 câu hỏi (trung bình là 5.4 câu hỏi), mỗi câu hỏi có 10 câu trả lời đúng, 3 câu trả lời gần đúng.

Tuy nhiên, việc xử lý bộ dữ liệu này rất phức tạp, cần rất nhiều thời gian và công sức. Vì vậy, trong phạm vi môn học này, nhóm sẽ chỉ xử lý trên một bộ dữ liệu khác đơn giản hơn (sẽ được giới thiệu sau).

2 Cách tiếp cận

Với một bài toán VQA cơ bản, chúng ta có những bước sau

1. Xử lý hình ảnh
   Ở bước xử lý hình ảnh, kết quả mong muốn của chúng ta là trích xuất được những thông tin cần thiết trong bức ảnh. Mô hình mạng tích chập (Convolutional Neural Network) là mô hình phổ biến nhất để trích xuất thông tin từ hình ảnh.

2. Xử lý ngôn ngữ
   Ở bước xử lý hình ảnh, kết quả mong muốn của chúng ta là trích xuất được những thông tin cần thiết về từ câu hỏi. Mô hình mạng hồi quy (Recurrent Neural Network) là mô hình phổ biến nhất để trích xuất thông tin từ ngôn ngữ tự nhiên. Với những câu không quá phức tạp, chúng ta có thể sử dụng một mô hình khác, chính là mạng Neural Network (đơn giản hơn Recurrent Neural Network). Với bộ dữ liệu sẽ được mô tả bên dưới, nhóm sẽ chọn mô hình Neural Network.

3. Kết hợp 2 mô hình trên
   Sau khi đã có những thông tin cần thiết từ hình ảnh và câu hỏi, đây là bước kết hợp 2 lượng thông tin ở trên lại. Vì kết quả của những mô hình trên đều ở dưới dạng Vector, nên ta dễ dàng kết hợp bằng việc lấy tích element-wise multiplication.

4. Đưa ra dự đoán cuối cùng
   Sau khi có được thông tin cuối cùng, việc cần làm chỉ là bài toán classification dựa trên bộ dữ liệu về câu trả lời. Với bài toán phân loại như trên thì mô hình softmax là phổ biến nhất.

Bộ dữ liệu: mhư đã đề cập, nhóm sẽ sinh ra một bộ dữ liệu đơn giản cho việc huấn luyện.

- Về hình ảnh: tự động sinh ra các hình đơn giản (ví dụ bên dưới) bằng thư viện Pillow trong Python.
- Về câu hỏi: tiếng Anh, tự động sinh ra bằng các lệnh if else đơn giản trong Python (ví dụ như 'What shape is in the image?', 'What color is the triangle?')
- Về câu trả lời: chỉ có 13 câu trả lời cố định:
  - Về câu hỏi Yes/No: Yes, No
  - Về shape: Circle, Rectangle, Triangle
  - Về color: Red, Green, Blue, Black, Gray, Teal, Brown, Yellow

3 Xử lý ngôn ngữ - Mô hình Neural Network

3.1 Bag of words - BOW

Bag of Words (BOW) là một phương pháp để trích xuất các đặc điểm từ các dữ liệu văn bản. Mô hình Bag of Words là một cách biểu diễn từ một văn bản tùy ý thành một vector có độ dài cố định bằng cách đếm số lần mỗi từ xuất hiện (thường được gọi là quá trình vector hóa). Sử dụng kĩ thuật này, ta có thể chuyển đổi từ sang biểu diễn số trong xử lý ngôn ngữ tự nhiên, nhúng toàn bộ văn bản và đưa chúng vào nhiều thuật toán học máy khác nhau.

Quá trình vector hóa này gồm hai giai đoạn chính:

1. Xác định từ vựng:
   Với xác định từ vựng, ta tiến hành loại bỏ các dấu câu, tách từ, tìm tập hợp tất cả các từ được xuất hiện trong văn bản. Ví dụ: Với các câu "Is there a red shape?" và "What shape is present?", những từ được xác định là "is", "there", "a", "red", "shape", "what", "present".

2. Sinh ra vector:
   Để vector hóa văn bản, ta đếm số lần xuất hiện của các từ trong tập hơn từ vựng trong văn bản.

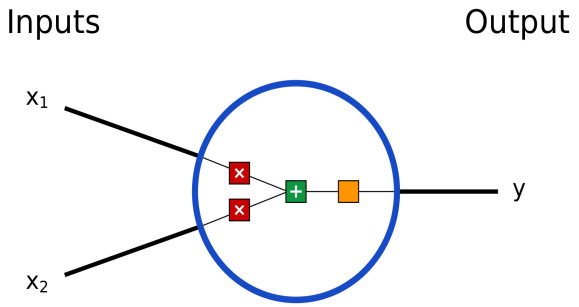
Document	is	there	a	red	shape	what	present
Is there a red shape?	1	1	1	1	1	0	0
What shape is present?	1	0	0	0	1	1	1

Cuối cùng ta có được các vector cho mỗi câu:

- Is there a red shape?: $[1, 1, 1, 1, 1, 0, 0]$
- What shape is present?: $[1, 0, 0, 0, 1, 1, 1]$

3.2 Neural Network

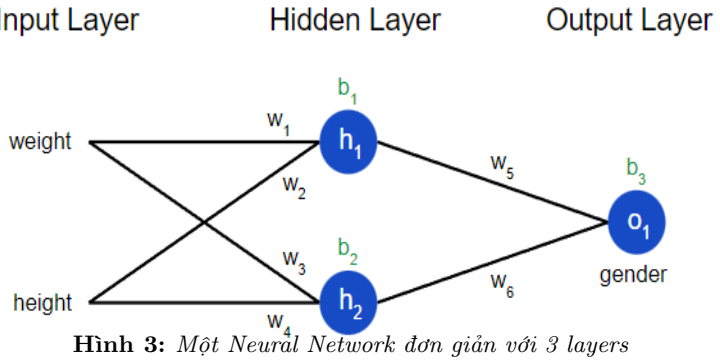
Trong mô hình Neural Network gồm nhiều layer Layer đầu tiên là input layer, các layer ở giữa được gọi là hidden layer, layer cuối cùng được gọi là output layer. Mỗi layer bao gồm các neuron (thành phần cơ bản nhất của neural network). Mỗi neuron lấy các input và đưa ra một output cùng với một hàm kích hoạt được xác định để sử dụng trong quá trình tính output.



Hình 2: *Ví dụ về neuron 2 input*

Giả sử với neuron 2 input ở trên, với input $x = [2, 3]$ cùng với weight $w = [0, 1]$, $b = 4$, hàm kích hoạt sigmoid là f. Giá trị output $y = f(w.x + b) = f(7) = 0.999$

Quay trở lại Neural Network, một Neural Network có thể có nhiều layer như đã đề cập ở trên với ý tưởng đưa các inputs chuyển tiếp qua các neuron trong mạng để nhận lấy output. Một layer với mọi nút được kết nối với mọi đầu ra từ lớp trước được gọi là fully-connected layer (FC Layer).



Hình 3: *Một Neural Network đơn giản với 3 layers*

Một cách để định lượng, đánh giá mức độ hoạt động tốt của một neural network là sử dụng hàm mất mát (loss function). Có nhiều hàm mất mát như: perceptron loss, logistic loss, mean squared error,... Thông qua hàm mất mát này các weight (trọng số) được điều chỉnh với mục tiêu tối thiểu giá trị của hàm mất mát - backpropagation. Hàm mất mát được tính toán dựa trên giá trị đầu ra (giá trị dự đoán y_{pred}) và giá trị thực sự (y_{true}) . Do đó, ta có thể viết hàm mất mát dưới dạng hàm của nhiều biến (bao gồm các weight và bias) như sau: $L(w1, w2, w3, w4, w5, w6, b1, b2, b3)$

Gỉa sử ta cần điều chỉnh $w1$, ta cần tính đạo hàm riêng phần $\frac{\partial y_{pred}}{\partial w_1}$.

Ta có thể viết lại đạo hàm riêng trên: $\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{\text{pred}}} * \frac{\partial y_{\text{pred}}}{\partial w_1}$

Ta có thể tính $\frac{\partial L}{\partial y_{\text{pred}}}$ thông qua:

$$\frac{\partial L}{\partial y_{\text{pred}}} = \frac{\partial \left(1 - y_{\text{pred}}\right)^2}{\partial y_{\text{pred}}} = -2\left(1 - y_{\text{pred}}\right)$$

Theo công thức để tính đầu ra ta có: $\frac{\partial y_{\text{pred}}}{\partial w_1}$. Với h_1, h_2, o_1 là output của các neuron như biểu diễn ở hình 3, f là hàm kích hoạt. Sau đó:

$$y_{\text{pred}} = o_1 = f\left(w_5 h_1 + w_6 h_2 + b_3\right)$$

$$\frac{\partial y_{\text{pred}}}{\partial w_1} = \frac{\partial y_{\text{pred}}}{\partial h_1} * \frac{\partial h_1}{\partial w_1}$$

$$\frac{\partial y_{\text{pred}}}{\partial h_1} = w_5 * f'\left(w_5 h_1 + w_6 h_2 + b_3\right)$$

Ta làm điều tương tự cho $\frac{\partial h_1}{\partial w_1}$:

$$h_1 = f\left(w_1 x_1 + w_2 x_2 + b_1\right)$$

$$\frac{\partial h_1}{\partial w_1} = x_1 * f'\left(w_1 x_1 + w_2 x_2 + b_1\right)$$

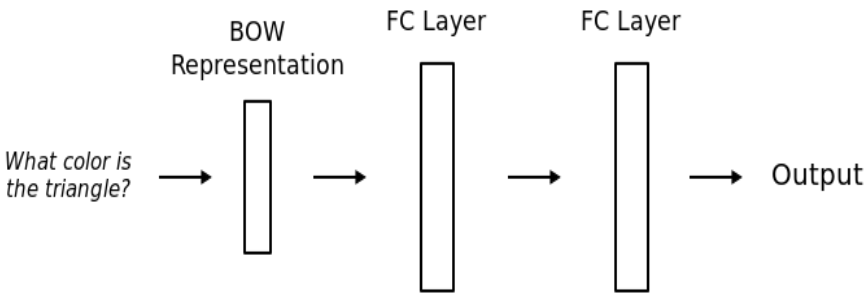
Do đó, ta đã chuyển $\frac{\partial y_{pred}}{\partial w_1}$ thành các biểu thức có thể tính toán được: $\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{\text{pred}}} * \frac{\partial y_{\text{pred}}}{\partial h_1} * \frac{\partial h_1}{\partial w_1}$

Cuối cùng, ta có thể cập nhật weight w_1 với Stochastic gradient descent theo công thức sau:

$$w_1 \leftarrow w_1 - \eta \frac{\partial L}{\partial w_1}$$

η là một hằng số được gọi là learning rate.

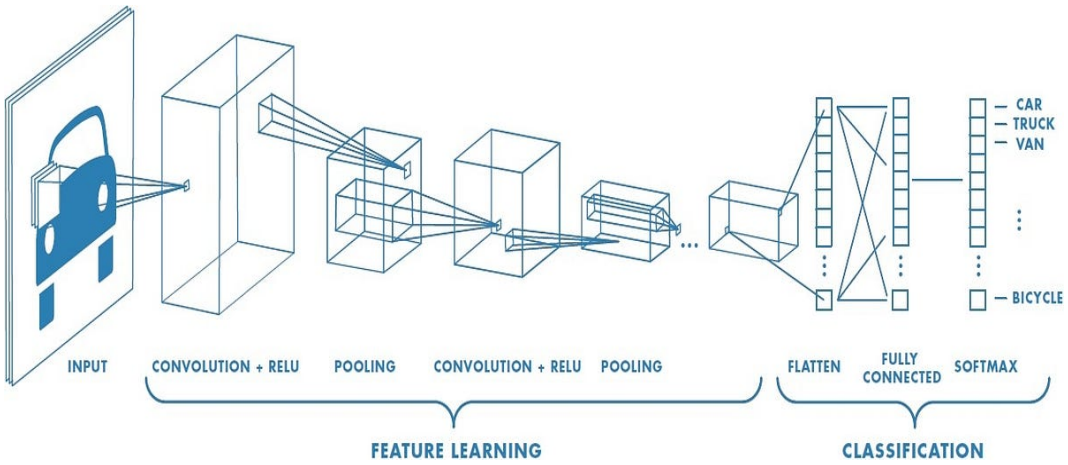
Như đã đề cập, tập hợp câu hỏi trong bộ dữ liệu đơn giản, ngắn gọi và là tập hợp cố định, mô hình Neural Network sẽ được sử dụng. Ta sẽ sử dụng vector được sinh ra từ phương pháp BOW trên làm đầu vào thông qua 2 fully-connected (FC) neural network layers, mỗi layer 32 neuron với hàm kích hoạt là hàm tanh.



Hình 4: *Mô hình sử dụng trong quá trình xử lý ngôn ngữ*

4 Xử lý hình ảnh - Mô hình CNN

Vì đây là phần liên quan nhiều hơn đến vấn đề thị giác máy tính (Computer Vision), không phải xử lý ngôn ngữ tự nhiên (Natural Language Processing), nên nhóm xin phép chỉ giới thiệu ngắn gọn về mô hình này.



Convolutional Neural Network là cơ bản là một Neural Network sử dụng các Convolutional Layers. Các Convolutional Layer dựa trên phép toán tích chập. Các Convolutional Layer bao gồm một tập hợp các bộ lọc (filter) mà có thể coi là ma trận hai chiều. Đây là một ví dụ về bộ lọc 3x3:

-1	0	1
-2	0	2
-1	0	1

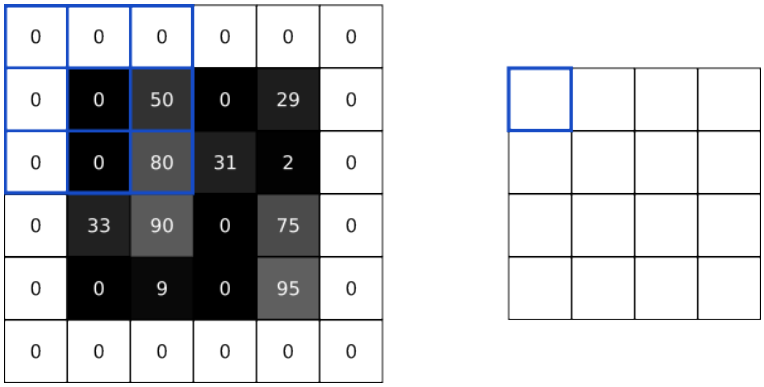
Ta có thể sử dụng hình ảnh đầu vào bộ lọc (filter) để tạo ra hình ảnh đầu ra bằng cách kết hợp bộ lọc với hình ảnh đầu vào, bao gồm các bước:

- Phủ bộ lọc lên trên hình ảnh ở một vị trí nào đó.
- Thực hiện phép nhân theo từng phần tử giữa các giá trị trong bộ lọc và các giá trị tương ứng của chúng trong hình ảnh.
- Tính tổng tất cả các kết quả của các phép nhân trên. Tổng này là giá trị đầu ra cho pixel đích trong ảnh đầu ra.
- Lặp lại cho tất cả các vị trí.

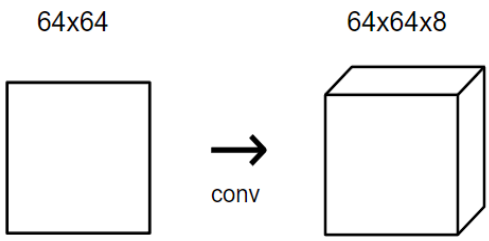
Như filter trên ví dụ là vertical Sobel filter, có khả năng phát hiện các cạnh dọc, ngược lại horizontal Sobel filter có khả năng phát hiện các cạnh ngang. Hình ảnh đầu ra được diễn giải dễ dàng: một pixel sáng (một pixel có giá trị cao) trong hình ảnh đầu ra cho biết có một cạnh mạnh xung quanh đó trong ảnh gốc. Nhìn chung, convolution giúp chúng ta tìm kiếm các đặc điểm hình ảnh cục bộ cụ thể (như các cạnh) mà chúng ta có thể sử dụng sau này trong mạng.

Original | Through vertical Sobel filter | Through horizontal Sobel filter

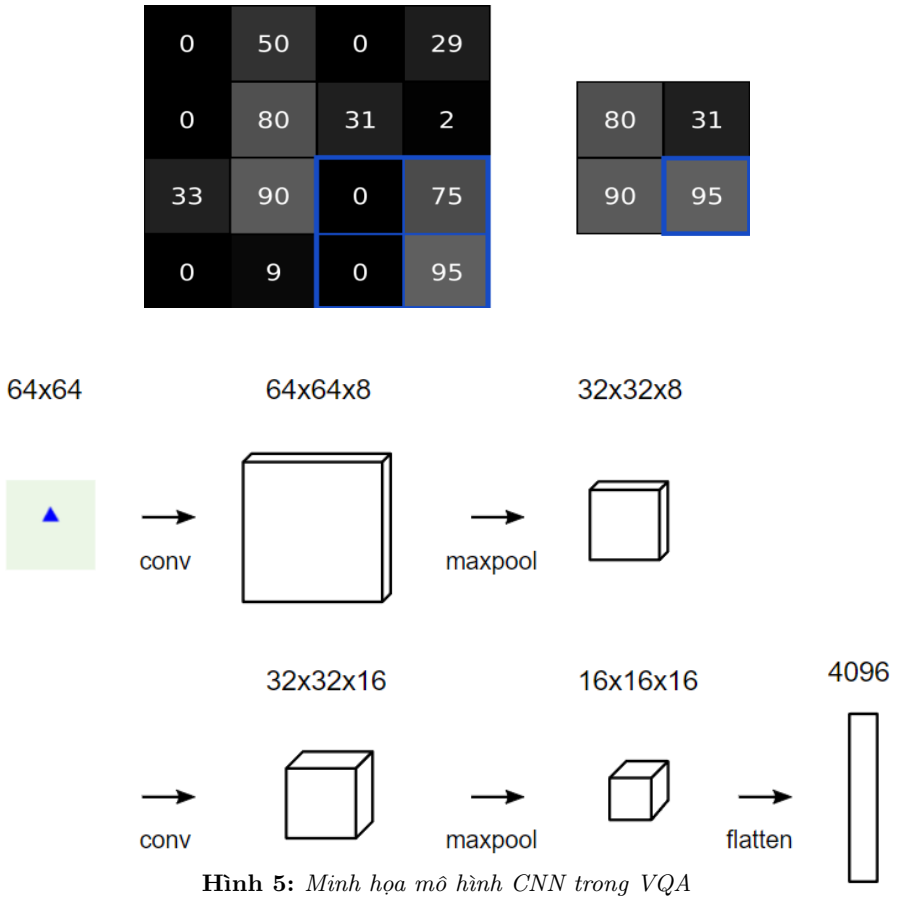
Khi ta muốn ma trận biểu diễn hình ảnh đầu ra có cùng kích thước với ma trận biểu diễn hình ảnh ban đầu, ta phải thêm các số 0 xung quanh hình ảnh để có thể phủ bộ lọc ở nhiều vị trí hơn, đó được gọi là padding.



Quay lại với Convolutional Layer, giả sử ta sử dụng một convolutional layer với 8 filter 3x3 với ma trận biểu diễn hình ảnh ban đầu có kích thước 64x64, đầu ra sẽ có dạng 64x64x8 (có sử dụng padding). Mỗi filter sẽ tạo ra một output kích thước 64x64 (do sử dụng padding).



Vì các pixel lân cận trong hình ảnh có xu hướng có các giá trị tương tự nhau, do đó các convolutional layer thường cũng sẽ tạo ra các giá trị tương tự cho các pixel lân cận ở đầu ra, dẫn đến kết quả, một số thông tin trong ma trận đầu ra có thể dư thừa. Các Pooling Layers sẽ giải quyết vấn đề này, giảm kích thước của đầu vào được cung cấp bằng cách gộp các giá trị lận cận lại trong đầu vào với nhau. Việc gộp nhóm thường được thực hiện bằng một thao tác đơn giản như max, min, average. Giả sử với pool size bằng 2, hình ảnh đầu vào sẽ được duyêttj theo khối 2x2 (vì pool size bằng 2) và đặt giá trị tối đa vào hình ảnh đầu ra ở pixel tương ứng.



Hình 5: *Minh họa mô hình CNN trong VQA*

5 Đưa ra dự đoán - Softmax

Sau khi kết hợp 2 mô hình, ta sẽ dùng mô hình đó để đưa ra kết quả cần tìm. Trong bước này ta sẽ sử dụng phương pháp Softmax.

Softmax là phương pháp biến giá trị thực bất kỳ thành xác suất, thường rất hữu ích trong học máy. Phép toán đằng sau nó rất đơn giản.

Giả sử cho một tập các số thực $x_1, x_2, ..., x_n, n \in \mathbb{N}$

Với một số $x_i, i \in \mathbb{N}, 1 \leq i \leq n$ bất kỳ, sẽ được biến đổi thành:

$$s(x_i) = \frac{e^{x_i}}{\Sigma_{j=1}^{n} e^{x_j}}$$

Ví dụ minh họa: Giả sử có một tập các số -2, 0, 5, 7

$$\begin{aligned} Denominator &= e^{-2} + e^{0} + e^{5} + e^{7} \\ &= 1246.1817 \end{aligned}$$

Ta có bảng:

x	$Numerator(e^x)$	$Probability(\frac{e^x}{1246.1817})$
-2	0.1353	0.0001
0	1	0.0008
5	148.4132	0.1191
7	1096.6332	0.88

6 Hiện thực

6.1 Sinh bộ dữ liệu

Ta chỉ xét một số màu và hình sau

```
class Color(Enum):
  BLACK = (0, 0, 0)
  GRAY = (128, 128, 128)
  RED = (255, 0, 0)
  GREEN = (0, 255, 0)
  BLUE = (0, 0, 255)
  YELLOW = (255, 255, 0)
  TEAL = (0, 128, 128)
  BROWN = (165, 42, 42)

class Shape(Enum):
  RECTANGLE = 1
  CIRCLE = 2
  TRIANGLE = 3
```

Ta sẽ sinh ra 2 tập dữ liệu là dữ liệu để train và dữ liệu để test. Với hai hằng số *NUM_TRAIN* và *NUM_TEST* mà ta chọn trước, dùng vòng lặp *for* để sinh dữ liệu, mỗi lần lặp, ta **create_image** và **create_questions** tương ứng:

- **def create_image(filename, shape, color)**: dùng thư viện Pillow để vẽ hình ảnh dựa trên *shape* và *color*, sau đó lưu dưới dạng *.png*. Ví dụ

```
draw = ImageDraw.Draw(im)
if shape is Shape.RECTANGLE:
    w = randint(MIN_SHAPE_SIZE, MAX_SHAPE_SIZE)
    h = randint(MIN_SHAPE_SIZE, MAX_SHAPE_SIZE)
    x = randint(0, IM_DRAW_SIZE - w)
    y = randint(0, IM_DRAW_SIZE - h)
    draw.rectangle([(x, y), (x + w, y + h)], fill=color.value)

```

- **def create_questions(shape, color, image_id)**: tạo sẵn bộ câu hỏi và tạo câu trả lời tương ứng dựa trên *shape* và *color*. Ví dụ

```
shape_name = shape.name.lower()
color_name = color.name.lower()

questions = [
    (f'what shape is in the image?', shape_name),
    (f'what color is the {shape_name}?', color_name),
  ]

yes_no_questions = []
for s in Shape:
    cur_shape_name = s.name.lower()
    pos_answer = 'yes' if s is shape else 'no'
    yes_no_questions.append((f'is there a {cur_shape_name}?', pos_answer))
    yes_no_questions.append((f'is there a {cur_shape_name} in the image?',
    pos_answer))
    yes_no_questions.append((f'does the image contain a {cur_shape_name}?',
    pos_answer))
    yes_no_questions.append((f'is a {cur_shape_name} present?', pos_answer))

```

6.2 Xử lý hình ảnh - CNN

```
im_input = Input(shape=im_shape)
x1 = Conv2D(8, 3, padding='same')(im_input)
x1 = MaxPooling2D()(x1)
x1 = Conv2D(16, 3, padding='same')(x1)
x1 = MaxPooling2D()(x1)
x1 = Conv2D(32, 3, padding='same')(x1)
```

```
x1 = MaxPooling2D()(x1)
x1 = Flatten()(x1)
x1 = Dense(32, activation='tanh')(x1)
```

6.3 Xử lý ngôn ngữ - Neural Network

```
q_input = Input(shape=(vocab_size,))
x2 = Dense(32, activation='tanh')(q_input)
x2 = Dense(32, activation='tanh')(x2)
```

6.4 Đưa ra dự đoán - Softmax

```
def build_model(im_shape, vocab_size, num_answers):
    ## The CNN Code mentioned above
    ## The Neural Network Code mentioned above

    out = Multiply()([x1, x2])
    out = Dense(32, activation='tanh')(out)
    out = Dense(num_answers, activation='softmax')(out)

    model = Model(inputs=[im_input, q_input], outputs=out)
    model.compile(optimizer=Adam(learning_rate=5e-4), loss='categorical_crossentropy',
    metrics=['accuracy'])

    return model
```

6.5 Bắt đầu huấn luyện mô hình

Trước hết, ta load dữ liệu từ những hình ảnh và câu hỏi đã sinh ở trên.

```
train_X_ims, train_X_seqs, train_Y, test_X_ims, test_X_seqs, test_Y, im_shape, vocab_size
    , num_answers, all_answers, test_qs, test_answer_indices, test_image_ids = setup()
```

Trong đó:

- **train_X_ims**: tập dữ liệu hình ảnh đầu vào, dùng để huấn luyện
- **train_X_seqs**: tập dữ liệu câu hỏi đầu vào, dùng để huấn luyện
- **train_Y**: tập dữ liệu câu trả lời, dùng để huấn luyện
- **test_X_ims**: tập dữ liệu hình ảnh đầu vào, dùng để kiểm tra
- **test_X_seqs**: tập dữ liệu câu hỏi đầu vào, dùng để kiểm tra
- **test_Y**: tập dữ liệu câu trả lời, dùng để kiểm tra
- **im_shape**: kích thước của hình ảnh
- vocab_size: kích thước của từ điển, trong trường hợp này, bộ câu hỏi đã được điều chỉnh sẵn từ trước, vì thế chỉ có *27* từ vựng trong toàn bộ bộ dữ liệu này.
- **all_answers**: list tất cả các câu trả lời có thể có.
- **num_answers**: độ dài của all_answers, trong trường hợp này là 13.
- **test_qs**: list tất cả các câu hỏi trong tập dữ liệu dùng để kiểm tra
- **test_answer_indices**: list câu trả lời của các câu hỏi tương ứng trong tập dữ liệu kiểm tra
- **test_image_ids**: list số thứ tự hình ảnh của các câu hỏi tương ứng trong tập dữ liệu kiểm tra

Sau đó, ta bắt đầu huấn luyện bằng thư viện Keras với *16 epochs*. Sau khi huấn luyện ta lưu kết quả mô hình vào file *model.keras*

```
model = build_model(im_shape, vocab_size, num_answers)
checkpoint = ModelCheckpoint('model.keras', save_best_only=True)

model.fit(
  [train_X_ims, train_X_seqs],
  train_Y,
  validation_data=([test_X_ims, test_X_seqs], test_Y),
  shuffle=True,
  epochs=16,
  callbacks=[checkpoint],
)
```

Dưới đây là kết quả huấn luyện:

```
--- Training model...
Epoch 1/16
2993/2993 ━━━━━━━━━━━━━━━━━━━━ 94s 28ms/step - accuracy: 0.6302 - loss: 0.9371 - val_accuracy: 0.6877 - val_loss: 0.7058
Epoch 2/16
2993/2993 ━━━━━━━━━━━━━━━━━━━━ 94s 31ms/step - accuracy: 0.6982 - loss: 0.6613 - val_accuracy: 0.7088 - val_loss: 0.5931
Epoch 3/16
2993/2993 ━━━━━━━━━━━━━━━━━━━━ 101s 34ms/step - accuracy: 0.7332 - loss: 0.5199 - val_accuracy: 0.7879 - val_loss: 0.4359
Epoch 4/16
2993/2993 ━━━━━━━━━━━━━━━━━━━━ 84s 28ms/step - accuracy: 0.8085 - loss: 0.3875 - val_accuracy: 0.8164 - val_loss: 0.3665
Epoch 5/16
2993/2993 ━━━━━━━━━━━━━━━━━━━━ 66s 22ms/step - accuracy: 0.8407 - loss: 0.3214 - val_accuracy: 0.8331 - val_loss: 0.3333
Epoch 6/16
2993/2993 ━━━━━━━━━━━━━━━━━━━━ 75s 25ms/step - accuracy: 0.8564 - loss: 0.2888 - val_accuracy: 0.8444 - val_loss: 0.3025
Epoch 7/16
2993/2993 ━━━━━━━━━━━━━━━━━━━━ 75s 25ms/step - accuracy: 0.8735 - loss: 0.2549 - val_accuracy: 0.8867 - val_loss: 0.2388
Epoch 8/16
2993/2993 ━━━━━━━━━━━━━━━━━━━━ 108s 36ms/step - accuracy: 0.9118 - loss: 0.1915 - val_accuracy: 0.9347 - val_loss: 0.1614
Epoch 9/16
2993/2993 ━━━━━━━━━━━━━━━━━━━━ 71s 23ms/step - accuracy: 0.9579 - loss: 0.1144 - val_accuracy: 0.9735 - val_loss: 0.0889
Epoch 10/16
2993/2993 ━━━━━━━━━━━━━━━━━━━━ 57s 19ms/step - accuracy: 0.9865 - loss: 0.0511 - val_accuracy: 0.9860 - val_loss: 0.0475
Epoch 11/16
2993/2993 ━━━━━━━━━━━━━━━━━━━━ 61s 20ms/step - accuracy: 0.9922 - loss: 0.0298 - val_accuracy: 0.9887 - val_loss: 0.0432
Epoch 12/16
2993/2993 ━━━━━━━━━━━━━━━━━━━━ 76s 25ms/step - accuracy: 0.9943 - loss: 0.0210 - val_accuracy: 0.9921 - val_loss: 0.0337
Epoch 13/16
2993/2993 ━━━━━━━━━━━━━━━━━━━━ 70s 23ms/step - accuracy: 0.9959 - loss: 0.0156 - val_accuracy: 0.9783 - val_loss: 0.0897
Epoch 14/16
2993/2993 ━━━━━━━━━━━━━━━━━━━━ 52s 17ms/step - accuracy: 0.9960 - loss: 0.0141 - val_accuracy: 0.9908 - val_loss: 0.0289
Epoch 15/16
2993/2993 ━━━━━━━━━━━━━━━━━━━━ 51s 17ms/step - accuracy: 0.9972 - loss: 0.0102 - val_accuracy: 0.9927 - val_loss: 0.0259
Epoch 16/16
2993/2993 ━━━━━━━━━━━━━━━━━━━━ 128s 33ms/step - accuracy: 0.9980 - loss: 0.0068 - val_accuracy: 0.9914 - val_loss: 0.0349
```

Hình 6: *Kết quả huấn luyện*

7 Đánh giá kết quả

Sau khi hiện thực, ta tiến hành kiểm thử.

7.1 Kết quả dự đoán trên bộ dữ liệu đã sinh

Ta cũng thực hiện load dữ liệu test (tương tự như load dữ liệu để huấn luyện) để kiểm tra mô hình.

```
train_X_ims, train_X_seqs, train_Y, test_X_ims, test_X_seqs, test_Y, im_shape, vocab_size
    , num_answers, all_answers, test_qs, test_answer_indices, test_image_ids = setup()
```

Ta dùng hàm **load_weights** để load mô hình đã được huấn luyện, và dùng hàm **predict** để dự đoán trên tập test.

```
model = build_model(im_shape, vocab_size, num_answers)

model.load_weights('model.keras')
predictions = model.predict([test_X_ims, test_X_seqs])
```

Cách chạy: Ở *terminal* trong folder *VQA-simple*, gõ dòng lệnh

```
python prediction.py test
```

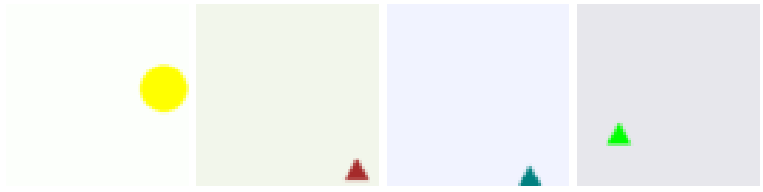
Dưới đây là kết quả dự đoán:

```
Image: 5     what is the color of the shape?              Answer: green        Prediction: teal
Image: 6     what is the yellow shape?                    Answer: circle       Prediction: rectangle
Image: 6     does the image not contain a rectangle?      Answer: yes          Prediction: no
Image: 17    what is the brown shape?                     Answer: triangle     Prediction: circle
Image: 17    what is the color of the triangle?           Answer: brown        Prediction: gray
Image: 17    what color is the shape?                     Answer: brown        Prediction: gray
Image: 17    is no gray shape present?                    Answer: yes          Prediction: no
Image: 17    does the image contain a brown shape?        Answer: yes          Prediction: no
Image: 44    what color is the triangle?                  Answer: teal         Prediction: gray
Image: 44    is a gray shape present?                     Answer: no           Prediction: yes
Image: 46    is there a triangle in the image?            Answer: no           Prediction: yes
Image: 64    what is the teal shape?                      Answer: rectangle    Prediction: circle
Image: 146   is there a green shape?                      Answer: yes          Prediction: no
Image: 146   is there not a green shape?                  Answer: no           Prediction: yes
Image: 146   does the image not contain a teal shape?     Answer: yes          Prediction: no
Image: 147   what color is the triangle?                  Answer: green        Prediction: teal
Image: 147   what is the color of the triangle?           Answer: green        Prediction: teal
Image: 147   does the image not contain a triangle?       Answer: no           Prediction: yes
Image: 147   is no green shape present?                   Answer: no           Prediction: yes
Image: 232   what is the yellow shape?                    Answer: rectangle    Prediction: circle
Image: 263   is no circle present?                        Answer: yes          Prediction: no
Image: 287   what shape is present?                       Answer: circle       Prediction: triangle
Image: 299   what is the blue shape?                      Answer: rectangle    Prediction: circle
Image: 299   is there a rectangle?                        Answer: yes          Prediction: no
Image: 299   is a rectangle present?                      Answer: yes          Prediction: no
Image: 299   does the image not contain a circle?        Answer: yes          Prediction: no
Image: 301   what shape does the image contain?          Answer: circle       Prediction: triangle
Image: 305   what shape is in the image?                 Answer: circle       Prediction: rectangle
Image: 305   what shape is present?                       Answer: circle       Prediction: rectangle
Image: 305   does the image not contain a blue shape?    Answer: no           Prediction: yes
Image: 380   what is the brown shape?                    Answer: circle       Prediction: triangle
Image: 380   is no triangle present?                      Answer: yes          Prediction: no
Image: 414   what shape is present?                       Answer: circle       Prediction: triangle
Image: 490   what color is the shape?                    Answer: gray         Prediction: black
Image: 493   is there not a triangle?                     Answer: yes          Prediction: no
----------------------
Total prediction: 4794
Wrong prediction: 35
```

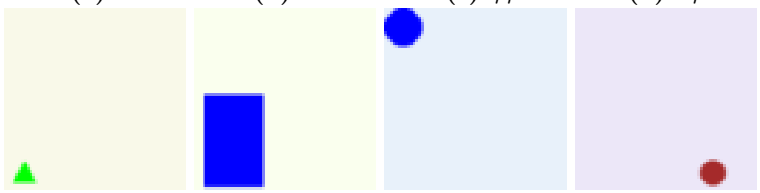
Hình 7: *Kết quả dự đoán*

Nhận xét:

- Tỷ lệ dự đoán đúng cao, là $\dfrac{4794 - 35}{4794} \cdot 100 = 0.992699207\%$
- Các dự đoán sai đa số tập trung vào một ảnh (có rất nhiều dự đoán sai ở hình ảnh thứ 6, 17, 44, 146, 147, 299, 305, 380), tức là có những hình ảnh mà mô hình dự đoán không tốt.



(a) *6* **(b)** *17* **(c)** *44* **(d)** *146*



(e) *147* **(f)** *299* **(g)** *305* **(h)** *380*

Hình 8: *Các hình có nhiều dự đoán sai*

Có thể thấy các hình này có điểm chung là hình bên trong nhỏ, và sát biên.

7.2 Kết quả dự đoán trên bộ dữ liệu khác

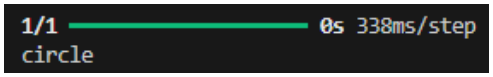
Cách chạy: Ở *terminal* trong folder *VQA-simple*, gõ dòng lệnh

```
python prediction.py custom filepath question
```

trong đó *filepath* là đường dẫn đến file *.png*, *question* là câu hỏi, cả hai đều ở dạng chuỗi.

Ta kiểm thử với dữ liệu như hình bên dưới và câu hỏi là **'what is the shape?'**

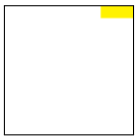
(a) *Dữ liệu mới*



(b) *Dự đoán dữ liệu mới*

Hình 9: *Dữ liệu mới*

Ta kiểm thêm một dữ liệu nữa với hình bên trong nhỏ và sát biên, câu hỏi là 'what is the color?'



(a) *Dữ liệu mới*

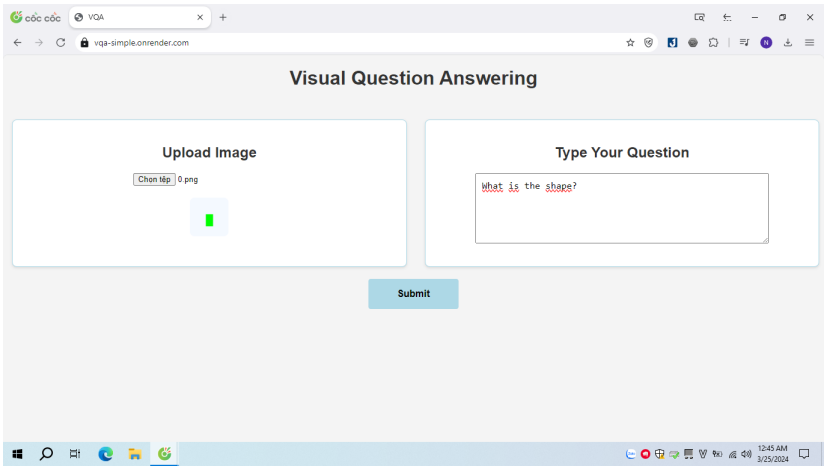
(b) *Dự đoán dữ liệu mới*

Hình 10: *Dữ liệu mới*

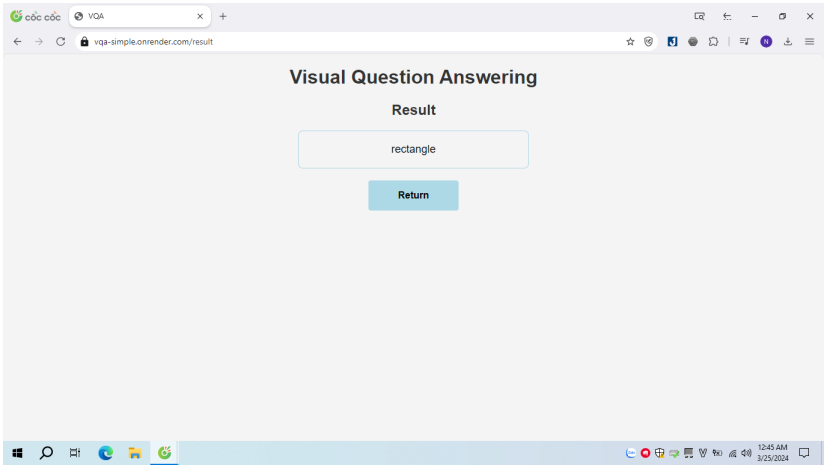
7.3 Kết quả dự đoán trên web

Ghi chú: Nhóm sử dụng *Flask* để hiện thực web và sử dụng *Render* để build web. Tuy nhiên vì đây không phải mục đích chính của bài tập nên nhóm sẽ không trình bày code hiện thực web ở đây.

Cách chạy: Truy cập đường dẫn `https://vqa-simple.onrender.com`, sau đó chọn file và nhập câu hỏi, sau đó ấn nút *Submit*.



Hình 11: *Giao diện web*



Hình 12: *Kết quả dự đoán trên web*

References

[1] Victor Zhou & Phillip Wang. *Easy Visual Question Answering*. August 16, 2020.

[2] Niralidedaniya. *Visual Question Answering — A Deep Learning Classification Case Study*. November 16, 2022.

[3] VQA. *Visual Question Answering*.