ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

----- □ 🕮 □ -----

BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN: NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Giảng viên hướng dẫn: **T.S Trần Thế Hùng**

Mã lớp: 147728

Đề tài: Image Captioning

Số thứ tự nhóm: 10

Danh sách sinh viên thực hiện:

STT	Họ và tên	MSSV
1	Mạc Văn An	20210007
2	Lê Quang Chiến	20214999
3	Phan Văn Đức	20215100
4	Bùi Thị Hương Trà	20215150
5	Nguyễn Hữu Duy	20215015

Hà Nội, tháng 06 năm 2024

Bảng phân công và đánh giá chi tiết thực hiện nhiệm vụ

Họ và tên	Vai trò	Nhiệm vụ	Mức độ hoàn thành nhiệm vụ	Mức độ đóng góp (Tổng 100%)
Mạc Văn An	Nhóm trưởng	- Triển khai và theo dõi tiến độ thực hiện project - Tìm hiểu kiến trúc mô hình, huấn luyện và đánh giá mô hình.	100%	20%
Lê Quang Chiến	Thành viên	- Tìm hiểu kiến trúc mô hình, phân tích kết quả.	100%	20%
Phan Văn Đức	Thành viên	- Tìm hiểu kiến trúc mô hình, trích xuất đặc trưng ảnh.	100%	20%
Bùi Thị Hương Trà	Thành viên	- Xử lý dữ liệu (captions), slide thuyết trình.	100%	20%
Nguyễn Hữu Duy	Thành viên	- Thu thập và phân tích dataset, slide thuyết trình.	100%	20%

MỤC LỤC

LỜI NÓI ĐẦU

Ngày nay, trí tuệ nhân tạo (AI) đang ngày càng trở thành một phần quan trọng và không thể thiếu trong nhiều lĩnh vực khác nhau, từ y tế, tài chính, đến giáo dục và giải trí. Các ứng dụng của AI, như nhận diện hình ảnh, xử lý ngôn ngữ tự nhiên, và học máy, đang thay đổi cách chúng ta sống và làm việc. Việc hiểu và ứng dụng AI không chỉ giúp chúng ta giải quyết các vấn đề phức tạp mà còn mở ra những cơ hội mới trong nghiên cứu và phát triển.

Trong học kỳ này, chúng em đã có cơ hội học tập và tìm hiểu về trí tuệ nhân tạo thông qua môn học "Nhập môn trí tuệ nhân tạo". Đây là một môn học rất thú vị và quan trọng, giúp chúng em nắm bắt được những khái niệm cơ bản và các phương pháp chính trong lĩnh vực AI. Để củng cố và áp dụng những kiến thức đã học, nhóm chúng em đã thực hiện đề tài về chú thích hình ảnh tự động – Image Captioning, một trong những ứng dụng điển hình và hấp dẫn của trí tuệ nhân tạo.

Trong suốt quá trình tìm hiểu và phân tích đề tài, do còn hạn chế về mặt kiến thức nghiệp vụ, kỹ năng cũng như về mặt thời gian tìm hiểu, khảo sát, nghiên cứu đề tài, nên bài báo cáo của chúng em sau đây không tránh khỏi nhiều sai sót. Chính vì vậy, chúng em rất mong được thầy chỉ bảo.

Chúng em xin chân thành cảm ơn!

I. TỔNG QUAN ĐỀ TÀI

1. Bối cảnh

Trong thời đại số hóa ngày nay, hình ảnh và video trở thành một phần không thể thiếu của cuộc sống hàng ngày. Hàng tỷ hình ảnh được tải lên và chia sẻ trên internet mỗi ngày, tạo ra một nhu cầu cấp thiết về các công cụ tự động phân tích và hiểu nội dung hình ảnh. Một trong những nhiệm vụ quan trọng trong lĩnh vực này là chú thích hình ảnh (image captioning), tức là gán các chú thích bằng văn bản cho hình ảnh.

Chú thích hình ảnh không chỉ giúp cải thiện khả năng tìm kiếm hình ảnh mà còn có ứng dụng trong nhiều lĩnh vực như:

- Hỗ trợ người khiếm thị: Cung cấp mô tả bằng văn bản giúp người khiếm thị hiểu nội dung hình ảnh.
- Quản lý nội dung số: Giúp phân loại và tìm kiếm hình ảnh trong các thư viện số lớn.
- Mạng xã hội: Tự động tạo chú thích cho hình ảnh đăng tải trên các nền tảng mạng xã hội.

2. Mục tiêu

Mục tiêu của dự án này là phát triển một mô hình học sâu có khả năng tự động tạo ra các chú thích mô tả nội dung của hình ảnh.

Cụ thể, dự án hướng tới:

- Xây dựng mô hình trích xuất đặc trưng hình ảnh sử dụng VGG16.
- Thiết kế và huấn luyện một mô hình tạo chú thích sử dụng kiến trúc LSTM để tạo ra các câu mô tả từ các đặc trưng hình ảnh.
- Đánh giá hiệu suất của mô hình bằng cách sử dụng các chỉ số đánh giá như BLEU.

II. BỘ DỮ LIỆU

1. Mô tả bộ dữ liệu

Bộ dữ liệu được sử dụng trong dự án này là Flickr8k, một bộ dữ liệu nổi tiếng trong nghiên cứu chú thích hình ảnh. Bộ dữ liệu này bao gồm:

- Hình ảnh: 8,000 hình ảnh đa dạng về chủ đề và bối cảnh.

- Chú thích: Mỗi hình ảnh được gán từ 5 chú thích khác nhau, mỗi chú thích là một câu mô tả nội dung của hình ảnh đó.

2. Chuẩn bị dữ liệu

Tải dataset flickr8k:

```
BASE_DIR = '/kaggle/input/flickr8k'
WORKING_DIR = '/kaggle/working'
```

2.1. Xử lý hình ảnh

- Tải và chuẩn bị mô hình VGG16:
  - Tải mô hình VGG16 đã được huấn luyện trước với trọng số từ bộ dữ liệu ImageNet. Trong phần này nhóm đã tải trọng số model về và đưa lên working dir trên Kaggle.
  - Cấu trúc lại mô hình để loại bỏ các lớp phân loại cuối cùng và giữ lại lớp đặc trưng.

```
# load vgg16 model
local_weights_path = '/kaggle/input/vgg16-weights/vgg16_weights_tf_dim_ordering_tf_kernels.h5'
model = VGG16(weights=local_weights_path)
# restructure the model
model = Model(inputs=model.inputs, outputs=model.layers[-2].output)
```

- Tiền xử lý hình ảnh:
  - Thay đổi kích thước ảnh về 224x224 pixels
  - Chuyển đổi hình ảnh sang mảng numpy
  - Chuẩn hóa giá trị pixel theo cách mà mô hình VGG16 yêu cầu.

```
features = {}
directory = os.path.join(BASE_DIR, 'Images')

for img_name in tqdm(os.listdir(directory)):
    img_path = directory + '/' + img_name
    image = load_img(img_path, target_size=(224, 224))
    image = img_to_array(image)
    image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
    image = preprocess_input(image)
    feature = model.predict(image, verbose=0)
    image_id = img_name.split('.')[0]
    features[image_id] = feature
```

- Trích xuất đặc trưng:
  - Sử dụng mô hình VGG16 đã cấu trúc lại để trích xuất đặc trưng của mỗi hình ảnh.

- Lưu trữ các đặc trưng này vào một file pickle để sử dụng sau này trong quá trình huấn luyện.

```
pickle.dump(features, open(os.path.join(WORKING_DIR, 'features.pkl'), 'wb'))
```

```
with open(os.path.join(WORKING_DIR, 'features.pkl'), 'rb') as f:
    features = pickle.load(f)
```

```
with open(os.path.join(BASE_DIR, 'captions.txt'), 'r') as f:
    next(f)
    captions_doc = f.read()
```

2.2. Xử lý chú thích

- Làm sạch chú thích:
  - Chuyển các từ trong chú thích thành chữ thường.
  - Loại bỏ các ký tự đặc biệt, số và dấu câu không cần thiết.
  - Thêm các token bắt đầu và kết thúc câu để đánh dấu vị trí bắt đầu và kết thúc của mỗi chú thích.

```
mapping = {}
for line in tqdm(captions_doc.split('\n')):
    tokens = line.split(',')
    if len(line) < 2:
        continue
    image_id, caption = tokens[0], tokens[1:]
    image_id = image_id.split('.')[0]
    caption = " ".join(caption)
    if image_id not in mapping:
        mapping[image_id] = []
    mapping[image_id].append(caption)
```

- Token hóa và đệm chuỗi
  - Sử dụng Tokenizer từ thư viện Keras để biến đổi các chú thích thành chuỗi các số nguyên (token).
  - Đệm các chuỗi này đến độ dài tối đa để đảm bảo mọi chuỗi có cùng độ dài.

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(all_captions)
vocab_size = len(tokenizer.word_index) + 1
```

- Tạo tập huấn luyện và kiểm tra: Chia bộ dữ liệu thành tập huấn luyện (90%) và tập kiểm tra (10%).

```
image_ids = list(mapping.keys())
split = int(len(image_ids) * 0.90)
train = image_ids[:split]
test = image_ids[split:]
```

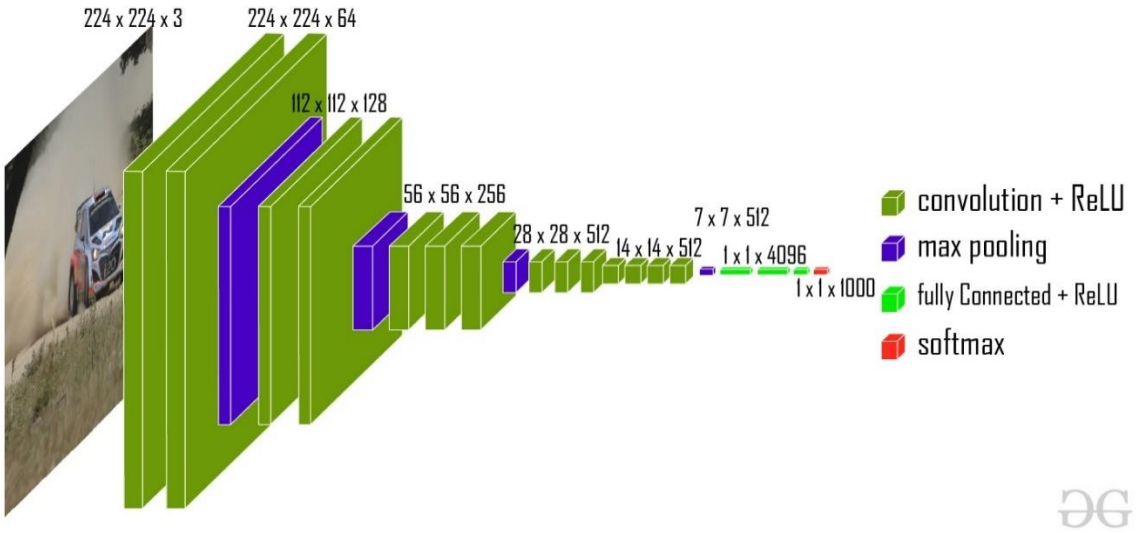
III. KIẾN TRÚC MÔ HÌNH

1. Trích xuất đặc trưng bằng VGG16

Mô hình VGG16, được phát triển bởi Visual Geometry Group (VGG) tại Đại học Oxford, là một trong những mô hình CNN (Convolutional Neural Network) nổi tiếng nhất và được huấn luyện trên bộ dữ liệu ImageNet. VGG16 có cấu trúc gồm 16 lớp (trong đó có 13 lớp chập và 3 lớp được kết nối đầy đủ) với trọng số được tối ưu hóa để nhận dạng và phân loại các đối tượng trong hình ảnh.

Cấu trúc chi tiết của VGG16

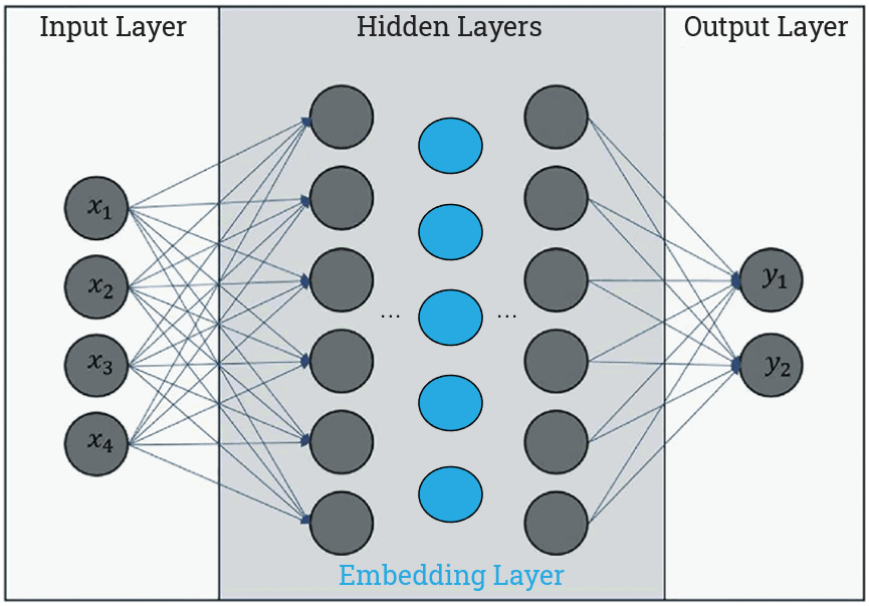
- **Các lớp tích chập (Convolutional Layers):** VGG16 gồm 13 lớp tích chập (conv layers) được sắp xếp thành các khối. Mỗi khối chứa 2 hoặc 3 lớp tích chập liên tiếp với các kích thước bộ lọc 3x3, stride 1 và padding 1.
- **Các lớp Pooling:** Sau mỗi khối tích chập là một lớp max pooling với kích thước 2x2 và stride 2, giúp giảm kích thước của feature map.
- **Các lớp Fully Connected:** Phần cuối của mô hình gốc có 3 lớp fully connected (FC), nhưng chúng ta sẽ loại bỏ 2 lớp FC cuối cùng để chỉ giữ lại lớp đặc trưng FC7.



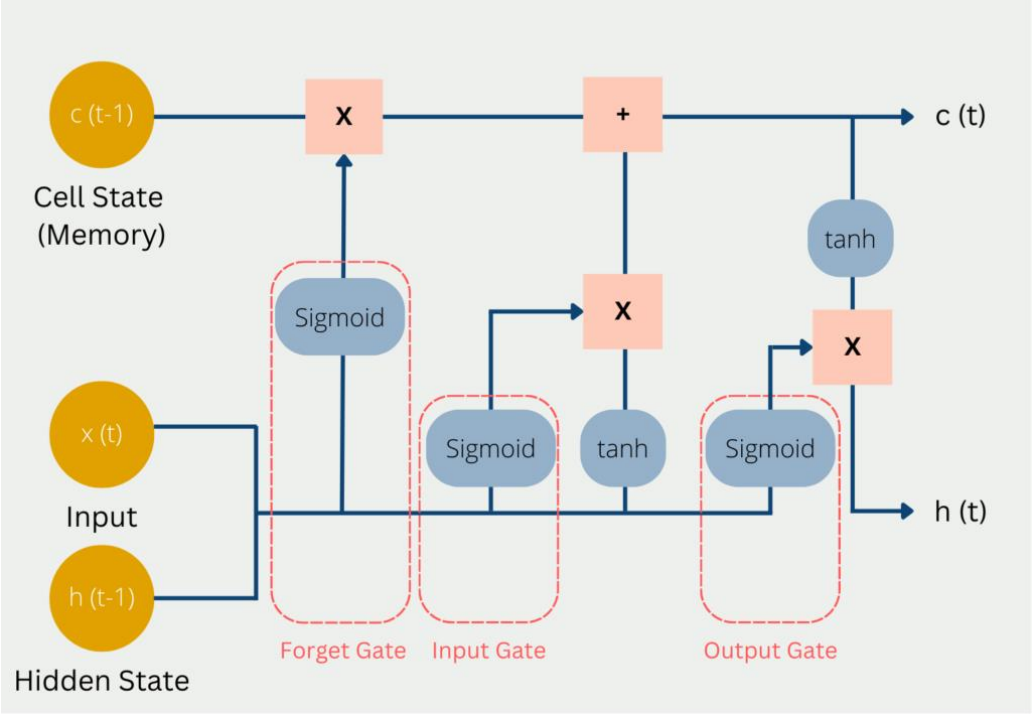
2. Mô hình tạo chú thích

Mô hình tạo chú thích bao gồm 3 phần chính: lớp Embedding, các lớp LSTM và các lớp Dense

- Lớp Embedding:
  - Chức năng: Lớp embedding chuyển đổi các từ (token) thành các vector có chiều cao hơn, giúp mô hình hiểu được mối quan hệ ngữ nghĩa giữa các từ.
  - Kích thước vector: Kích thước vector embedding thường được chọn dựa trên kinh nghiệm hoặc thử nghiệm, thường là 256 hoặc 512. Trong phần code của nhóm sử dụng kích thước vector là 256.
  - Embedding matrix: Ma trận embedding được học trong quá trình huấn luyện, giúp ánh xạ các từ thành các vector số.



- Các lớp LSTM (Long Short-Term Memory):
    - Chức năng: LSTM là một loại RNN (Recurrent Neural Network) có khả năng học và ghi nhớ thông tin tuần tự dài hạn, phù hợp với dữ liệu chuỗi như văn bản.
    - Cấu trúc: Mỗi LSTM cell gồm các cổng (gates) kiểm soát luồng thông tin: input gate, forget gate, và output gate. Các cổng này giúp LSTM ghi nhớ và quên thông tin theo thời gian.
    - Số lượng layers: Thông thường, mô hình sử dụng 1 hoặc 2 lớp LSTM. Mỗi lớp có thể có từ 256 đến 512 units (neurons). Trong phần code của nhóm chỉ sử dụng 1 lớp LSTM với 256 units.



- Các lớp Dense:
    - Chức năng: Lớp dense (fully connected layer) tạo ra đầu ra là các xác suất của từ vựng, giúp mô hình dự đoán từ tiếp theo trong chuỗi chú thích.
    - Activation function: Sử dụng hàm softmax để chuyển đổi output thành xác suất.
    - Output size: Kích thước của lớp dense đầu ra bằng kích thước của từ điển (vocabulary size).

IV. HUẤN LUYỆN MÔ HÌNH

1. Cài đặt huấn luyện
    - Hàm tạo dữ liệu: Hàm tạo dữ liệu có vai trò quan trọng trong việc xử lý các bộ dữ liệu lớn và ngăn chặn tình trạng tràn bộ nhớ trong quá trình huấn luyện mô hình. Thông thường, hàm này sẽ chia tập dữ liệu thành các batch nhỏ để đưa vào quá trình huấn luyện.
    - Quy trình huấn luyện:
        - Chia bộ dữ liệu thành tập huấn luyện và tập kiểm tra: Để đánh giá hiệu suất của mô hình, bộ dữ liệu thường được chia thành hai phần, tỉ lệ phổ biến là 90% cho tập huấn luyện và 10% cho tập kiểm tra.
        - Định nghĩa kích thước batch và số epoch: Kích thước batch là số lượng mẫu dữ liệu được sử dụng trong mỗi lần cập nhật trọng số mô hình, trong khi số epoch là số lần mà toàn bộ bộ dữ liệu được sử dụng để huấn luyện mô hình.

```
epochs = 20
batch_size = 64
steps = len(train) // batch_size
```

2. Biên dịch mô hình
    - Hàm mất mát và bộ tối ưu hóa:
        - Hàm mất mát categorical_crossentropy: Đây là hàm mất mát thích hợp cho các bài toán phân loại đa lớp, được sử dụng để đo lường sự khác biệt giữa dự đoán của mô hình và nhãn thực tế.
        - Bộ tối ưu hóa Adam: Adam là một phương pháp tối ưu hóa hiệu quả và phổ biến trong deep learning. Nó kết hợp hai kỹ thuật là momentum và RMSprop để cập nhật trọng số của mạng nơ-ron.

```
inputs1 = Input(shape=(4096,), name="image")
fe1 = Dropout(0.4)(inputs1)
fe2 = Dense(256, activation='relu')(fe1)
inputs2 = Input(shape=(max_length,), name="text")
se1 = Embedding(vocab_size, 256, mask_zero=True)(inputs2)
se2 = Dropout(0.4)(se1)
se3 = LSTM(256, use_cudnn=False)(se2)

decoder1 = add([fe2, se3])
decoder2 = Dense(256, activation='relu')(decoder1)
outputs = Dense(vocab_size, activation='softmax')(decoder2)

model = Model(inputs=[inputs1, inputs2], outputs=outputs)
model.compile(loss='categorical_crossentropy', optimizer='adam')

plot_model(model, show_shapes=True)
```

3. Vòng lặp huấn luyện

Trong vòng lặp huấn luyện, quá trình diễn ra như sau:

- Sử dụng hàm tạo dữ liệu để tạo ra các batch dữ liệu: Dữ liệu được chia thành các batch nhỏ để đưa vào mô hình.
- Sử dụng phương pháp fit để huấn luyện mô hình theo từng epoch: Mỗi epoch, các batch dữ liệu được đưa vào mô hình và cập nhật trọng số để tối ưu hóa hàm mất mát. Điều này tiếp tục cho đến khi số epoch đã được xác định trước kết thúc.

```
for i in range(epochs):
    generator = data_generator(train, mapping, features, tokenizer, max_length, vocab_size, batch_size)
    model.fit(generator, epochs=1, steps_per_epoch=steps, verbose=1)
```

V. ĐÁNH GIÁ KẾT QUẢ VÀ PHÂN TÍCH

1. Quy trình dự đoán

Hàm idx_to_word được sử dụng để chuyển đổi chỉ số (index) của từ được dự đoán trở lại thành từ (word) tương ứng trong từ điển của tokenizer.

```
def idx_to_word(integer, tokenizer):
    for word, index in tokenizer.word_index.items():
        if index == integer:
            return word
    return None
```

Hàm predict_caption được sử dụng để tạo ra các chú thích cho hình ảnh từ mô hình đã huấn luyện.

```
def predict_caption(model, image, tokenizer, max_length):
    in_text = 'startseq'
    for i in range(max_length):
        sequence = tokenizer.texts_to_sequences([in_text])[0]
        sequence = pad_sequences([sequence], max_length)
        yhat = model.predict([image, sequence], verbose=0)
        yhat = np.argmax(yhat)
        word = idx_to_word(yhat, tokenizer)
        if word is None:
            break
        in_text += " " + word
        if word == 'endseq':
            break

    return in_text
```

2. Chỉ số đánh giá BLEU
    - BLEU là viết tắt của Bilingual Evaluation Understudy, là phương pháp đánh giá một bản dịch dựa trên các bản dịch tham khảo, được giới thiệu trong paper BLEU: a Method for Automatic Evaluation of Machine Translation). BLEU được thiết kế để sử dụng trong dịch máy (Machine Translation), nhưng thực tế, phép đo này cũng được sử dụng trong các nhiệm vụ như tóm tắt văn bản, nhận dạng giọng nói, sinh nhãn ảnh v..v
3. Cách tính BLEU Score

    BLEU score được tính bằng cách kết hợp giữa precision và recall của các từ trong các chuỗi dự đoán so với các chuỗi thực tế. Cụ thể:

    - Precision: Tỉ lệ số từ dự đoán đúng trên tổng số từ dự đoán.
    - Recall: Tỉ lệ số từ dự đoán đúng trên tổng số từ thực tế.

    BLEU score tính toán bằng cách kết hợp precision và recall sử dụng một hàm kết hợp như geometric mean hoặc arithmetic mean.

    Điểm số BLEU-1 là precision của các từ đơn lẻ, trong khi BLEU-2 là precision của các cặp từ liên tiếp.

Trong mã nguồn được cung cấp, nhóm sử dụng thư viện NLTK để tính toán BLEU score. Hàm corpus_bleu trong thư viện này được sử dụng để tính toán BLEU score dựa trên các chuỗi dự đoán và chuỗi thực tế.

```
from nltk.translate.bleu_score import corpus_bleu
actual, predicted = list(), list()

for key in tqdm(test):
    captions = mapping[key]
    y_pred = predict_caption(model, features[key], tokenizer, max_length)
    actual_captions = [caption.split() for caption in captions]
    y_pred = y_pred.split()
    actual.append(actual_captions)
    predicted.append(y_pred)

print("BLEU-1: %f" % corpus_bleu(actual, predicted, weights=(1.0, 0, 0, 0)))
print("BLEU-2: %f" % corpus_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0)))
```

4. Kết quả định lượng:

```
BLEU-1: 0.552357
BLEU-2: 0.318630
```

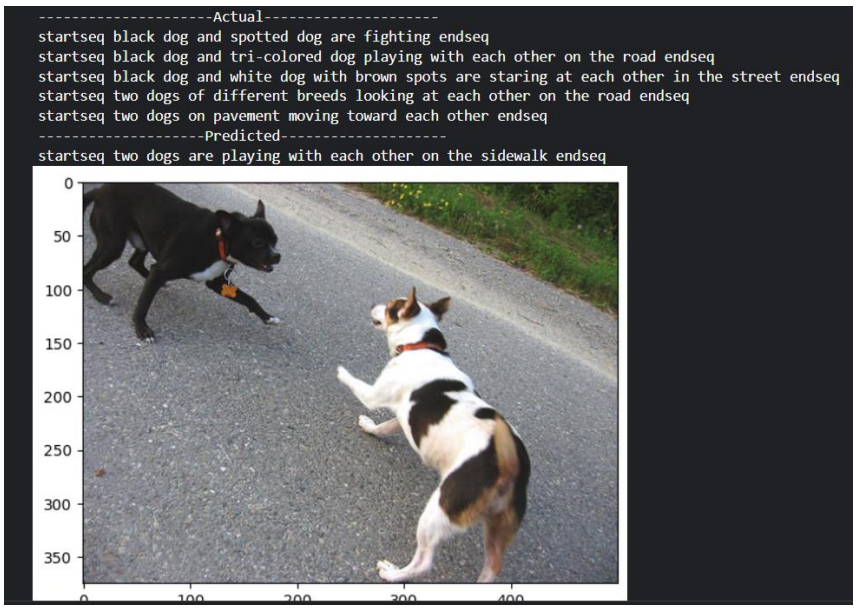
➔ Điểm số BLEU-1 và BLEU-2 của mô hình cho thấy mô hình có khả năng tạo ra các câu chú thích tương đối chính xác so với các câu tham chiếu, đặc biệt là ở cấp độ từ đơn lẻ (unigram). Tuy nhiên, điểm BLEU-2 thấp hơn cho thấy mô hình còn gặp khó khăn trong việc tạo ra các cặp từ liên tiếp chính xác.

5. Kết quả định tính

Test 1:

➔ Đã xác định đúng được little girl và stairs, nhưng sai hành động

Test 2:



➔ Xác định đúng trong ảnh bao gồm 2 chú chó, nhưng hành động chưa thực sự phù hợp

➔ Các ví dụ trên cho thấy mô hình có khả năng tạo ra các chú thích hợp lý và có nghĩa. Tuy nhiên, đôi khi mô hình dự đoán các từ hoặc cụm từ không hoàn toàn chính xác so với thực tế.

VI. KẾT LUẬN

1. Tóm tắt kết quả

   Dự án đã chưa thực sự thành công trong việc phát triển một mô hình chú thích hình ảnh tự động, với các điểm số BLEU-1 và BLEU-2 lần lượt là 0.552357 và 0.318630. Mô hình đã chứng minh khả năng tạo ra các chú thích có nghĩa tuy nhiên chỉ dừng lại ở mức tương đối chính xác so với các chú thích thực tế.

2. Thách thức và hạn chế

   Một số thách thức và hạn chế bao gồm:

   - Số lượng dữ liệu huấn luyện hạn chế.
   - Độ phức tạp của các chú thích thực tế, yêu cầu mô hình phải nắm bắt được ngữ cảnh và cấu trúc ngôn ngữ phức tạp.
   - Cần điều chỉnh và tối ưu hóa các tham số của mô hình để cải thiện kết quả.

3. Nhiệm vụ trong tương lai

   Các hướng cải tiến tiềm năng cho mô hình bao gồm:

- Sử dụng bộ dữ liệu lớn hơn và đa dạng hơn để huấn luyện mô hình.
- Thử nghiệm với các kiến trúc mô hình khác, như Transformer.
- Triển khai các kỹ thuật tiền xử lý nâng cao để làm sạch và chuẩn hóa dữ liệu tốt hơn.
- Sử dụng các chỉ số đánh giá khác để có cái nhìn toàn diện hơn về hiệu suất của mô hình.

VII. TÀI LIỆU THAM KHẢO

[1]. https://www.researchgate.net/figure/The-architecture-of-the-embedding-layer-For-a-given-word-ie-that-Every-character_fig2_335580898

[2]. https://www.geeksforgeeks.org/vgg-16-cnn-model/

[3]. https://trituenhantao.io/kien-thuc/bleu-phep-do-trong-dich-may/

[4]. https://machinelearningmastery.com/calculate-bleu-score-for-text-python/

[5]. https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/

[6]. https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/

[7]. https://www.geeksforgeeks.org/understanding-of-lstm-networks/