Bastien SAUVAT
Bastien FAISANT
ESIR3 INFO SI Group 1

Université de Rennes

AI - Project

Image to text description

ECOLE SUPERIEURE D'INGENIEURS DE RENNES

4th February 2024

1. Project presentation

The aim of the project is to realize Image Captioning. Image captioning is a computer vision task that involves **generating descriptive textual captions for images**. The goal is to enable machines to understand and describe the content of an image by producing human-like captions that capture key elements and details present in the visual content.

To complete this project, we have created a GitHub repository available at the following address : https://github.com/Unstery/AI_Sauvat_Faisant. The project is organized within the **TP3-8** folder. This project contains 2 Notebooks for each created model, as well as the 2 datasets we used to train our model. The first Notebook "**exercice3_image_to_text.ipynb**" contains the model we created, while the second Notebook "**exercice3_fine_tuning.ipynb**" corresponds to the fine-tuned model. The 2 datasets, "**flickr8k**" and "**FoodImages**", used are stored in the "kaggle/input/" folder.

2. Methodology

To begin, we were looking for a tutorial for Image Captioning. We found a Youtube tutorial very well detailed : https://www.youtube.com/watch?v=fUSTbGrL1tc. This tutorial also had a web version which is available here : https://www.hackersrealm.net/post/image-caption-generator-using-python
Thanks to the 2 versions of this tutorial, we arrive at the first Image Captioning Notebook quite functional.

We then have to fine tune our model on a specific thematic. So we search datasets on a specific subject where we can improve our model. We finally chose a dataset concerning "Food ingredients and recipes" available on Kaggle website at the following address :
https://www.kaggle.com/datasets/pes12017000148/food-ingredients-and-recipe-dataset-with-images
For this part, the goal is to use pre-trained model to generate meaningful and contextually relevant captions that describe the content of food images

3. Details of the techniques tested and used

a. Image Caption Generator using Flickr 8k Dataset

The main techniques we used can be resumed in this schema :

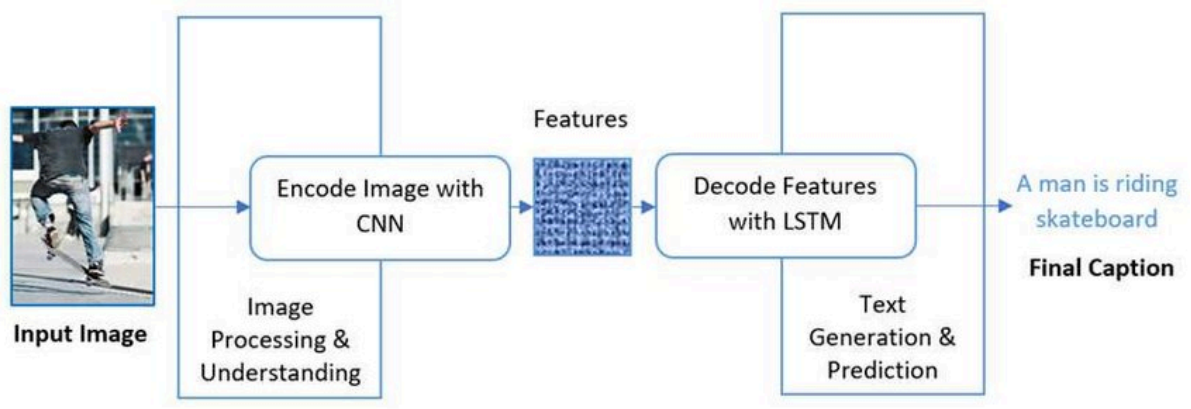


Figure 1 - Schema representing Image Captioning

Concerning the detailed steps we used, in image captioning model "**exercice3_image_to_text.ipynb**":

1. **Import modules** : We mainly used tensorflow and transformers libraries for the first model, and also pandas, numpy for basic operations

2. **Extract images features** : We employ the pre-trained VGG16 model to extract image features. It modifies the VGG16 to exclude the final classification layer and then processes a directory of images, extracting features using the modified model. The extracted features are stored in a dictionary and saved as a pickle file. The code also demonstrates loading the features from the pickle file.

3. **Load the captions data** : We read a file named 'captions.txt,' ignoring the first line, and then process the document line by line. It splits each line into tokens using commas and extracts the image identifier and its corresponding captions. The image identifier is further cleaned by removing the file extension. The script then constructs a dictionary called 'mapping,' associating each image identifier with a list of its captions. The final result is a structured mapping of image identifiers to their respective captions, facilitating further data processing or analysis.

4. **Preprocessing text data** : We define a clean function that takes a dictionary mapping containing image identifiers associated with lists of captions. It then performs a series of transformations on each caption, including converting to lowercase, removing non-alphabetic characters, adding special tags ("startseq" and "endseq"), and reducing consecutive spaces.
   Next, the code applies the clean function to a specific entry in the mapping dictionary. Finally, it aggregates all the transformed captions into a list called all_captions, tokenizes these captions using a Tokenizer, determines the vocabulary size, and calculates the maximum length of a caption in the dataset.

5. **Train test split :** We prepare data for a neural network model. It divides the image identifiers into training and testing sets, and then defines a data generator function. The generator produces batches of input-output pairs for the model during training. For each caption associated with an image, it tokenizes the text, creates input-output pairs by considering partial sequences, and converts them into numerical representations suitable for feeding into the neural network. The generator yields batches continuously, preventing potential session crashes by processing data incrementally.

6. **Model creation** : We define and train a neural network model for image captioning. The model consists of an encoder and a decoder. The encoder processes image features, while the decoder handles sequential text data. The encoder and decoder are connected, and the model is compiled using categorical crossentropy loss and the Adam optimizer. The training is performed over multiple epochs using a data generator that produces batches of training samples. After training, the model is saved as 'image_to_text_model.h5' in the specified working directory.

7. **Generate Caption for the Image** : We define two functions related to the evaluation of a trained image captioning model. The idx_to_word function converts an integer back to its corresponding word using the provided tokenizer. The predict_caption function generates a caption for an image using the trained model. It starts with the 'startseq' token and iteratively predicts the next word until the 'endseq' token is encountered or the maximum caption length is reached.
   The code then uses these functions to generate and compare captions for the test dataset. It calculates BLEU scores, a metric for evaluating the quality of generated text compared to reference text. The results are printed, showing BLEU-1 and BLEU-2 scores for unigram and bigram matches, respectively. These scores provide insights into the performance of the model in generating captions that align with the reference captions in the test set.

8. **Visualize the Results** : We use the trained image captioning model to generate and display captions for three different images. For each image, it prints the actual captions associated with the image in the dataset and the predicted caption generated by the model. Additionally, it visualizes the image using Matplotlib.

In the first example, the model attempts to generate captions for an image featuring dogs playing. The actual captions describe various interactions between the dogs, while the predicted caption suggests that two dogs are playing in the grass.

In the second example, the model generates captions for an image of a little girl with pigtails painting in front of a rainbow. The actual captions capture different aspects of the scene, while the predicted caption repeats a phrase about the little girl sitting in front of the rainbow.

In the third example, the model predicts captions for an image showing a person on skis looking at framed pictures set up in the snow. The actual captions describe the scene involving a skier and framed artwork, while the predicted caption repeats a phrase about a person wearing skis and displaying framed pictures.

The code provides insights into how well the trained model aligns with the actual captions for these specific images.

b. Fine tuning pre-trained model on food dataset

For the second part "**exercice3_fine_tuning.ipynb**", the goal is to generate captions for food images using pre-trained transformer models from the Hugging Face Transformers library. The dataset consists of food images along with associated titles, ingredients, instructions and image name in a CSV file. The dataset includes 13,582 images.
First of all, the dataset is loaded by using the "datasets" library. Then, a JSONL file was created to handle any missing or problematic data. Indeed, this file contains information about the image "file_name" and their corresponding titles "text":

```
{'file_name':
'miso-butter-roast-chicken-acorn-squash-panzanella.jpg',
 'text': 'Miso-Butter Roast Chicken With Acorn Squash Panzanella'}
```

To efficiently handle image and text data, a **CustomDataset** class was implemented using **PyTorch** machine learning library. PyTorch provides flexible and dynamic computational graphs, making it well-suited for tasks such as image captioning.
The pre-trained model chosen for this task is "**Salesforce/blip-image-captioning-base**". This model is based on the BLIP framework. It uses bootstrapping to effectively pre-train on noisy web data, showing very good performance in different vision-language tasks.
So, we can load the model and processor directly with the "from_pretrained" method. The processor facilitates tasks like resizing, normalization and other transformations needed before feeding the images into the model. And, in the case of text, it manages tokenization, padding and other necessary steps to prepare the text input for the model.
Due to the substantial computational resources required for training on our dataset, which we currently lack, we made the decision to work with a **subsample of 1,000 instances** for experimentation. Then, we instantiate our CustomDataset, which is

then passed as a parameter to the PyTorch's **DataLoader** for batch data loading. In our case, we choose a **batch size of 4**, the maximum feasible size limited by the available memory. We have initialized an Adam optimizer. Additionally, the model was transferred to the GPU if available. This decision was made because GPUs are particularly well-suited for training neural networks due to their parallel processing capabilities.

```
device = "cuda" if torch.cuda.is_available() else "cpu"
model.to(device)
```

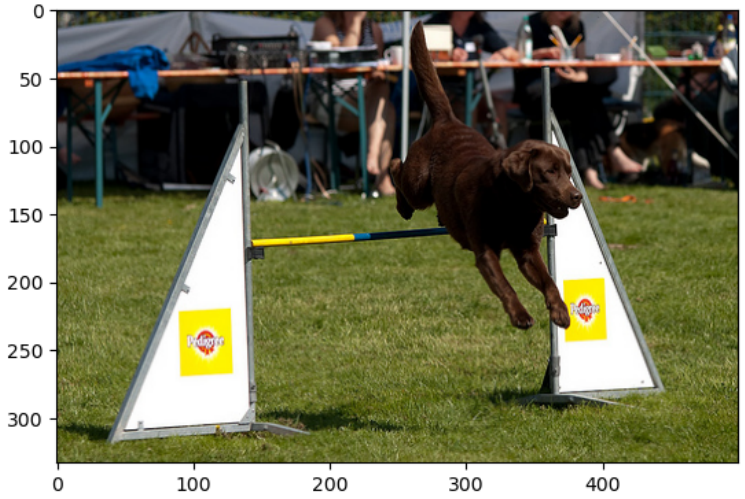
Following, a training loop was implemented with **3 epochs**. Batches of data from the DataLoader were used to train the model and to update the model parameters accordingly. After training, the fine-tuned model and its processor were saved to be reusable later.
For the test part, the fine-tuned model and its processor are loaded and the inference is demonstrated by generating captions for new images.

4. Hypotheses

We hypothesized that combining convolutional neural networks for image feature extraction and recurrent neural networks for sequential text processing would result in an effective image captioning model. Additionally, it was expected that optimizing preprocessing steps and model parameters could enhance caption quality.
Additionally, we hypothesized that fine-tuning a model on domain-specific dataset 'FoodImages' would enhance the model's ability to generate contextually relevant and accurate textual descriptions.

5. Results and conclusion

Training the model is very long, it tooks us several hours to launch the epochs for our programs. However, the image captioning model, developed using the Flickr dataset, demonstrated promising outcomes. The **BLEU-1** and **BLEU-2** scores for unigram and bigram matches on the test data were **0.258641** and **0.128733**, respectively. These scores indicate a moderate alignment between the generated captions and the actual reference captions in the test set.



'startseq a dog jumping over an obstacle course in an obstacle courseendseq endseq'

Figure 2 - Result obtained for the prediction of the image caption with the first model

Although the model showed promise, it could benefit from more fine-tuning to improve caption quality. It occasionally repeated words and faced challenges in capturing nuanced details. Future enhancements might include adjusting hyperparameters, exploring advanced architectures and considering an increased number of training epochs for potentially improved and accurate results.

For the second model, the process successfully fine-tuned a transformer model for image captioning on a food dataset. Indeed, the description of the image with the fine-tuned model is more precise than that of the basic model. However, there are still some inaccuracies compared to the real title of the image

model : 'a plate of food with a chicken and vegetables'
fine-tuned model : 'roast chicken with potatoes and greens'
legend of the image : 'Miso-Butter Roast Chicken With Acorn Squash Panzanella'

Figure 3 - Comparison of the results obtained for the prediction of the image caption with the model and the fine-tuned model

In the goal of optimizing the model, we can fine-tune hyperparameters, including learning rates and batch sizes, during the training process. In future work, an expansion of the dataset could be considered, provided the availability of ample computational resources. In our case, training the model on the complete dataset took a lot of time. This is why we opted to reduce the dataset size. Despite this adjustment, the execution time was long. To mitigate this, we found a solution by doing the training on Google Colab by utilizing the T4 GPU execution option instead of a simple CPU.
Then, a human evaluation is recommended to systematically assess the qualitative aspects of the generated captions. Additionally, incorporating several performance metrics is also advised like **BLEU**, **ROUGE** or **METEOR** metrics.

In general, results are very positive, we are very satisfied. The first model we use concerning all types of images is nearly instant to use once the model is launched.
In terms of prospects, we have the possibility to fine-tune on every image type that have a dataset (for example birds, flowers…). Another very interesting feature could be the ability to display the recipe corresponding to the input image.