1st Beshoy Malak Fathy
CIE
Zewail city
Giza, Egypt
201800613, s-beshoy_malak@zewailcity.edu.eg

2nd Belal Rashwan
CIE
Zewail city
Giza, Egypt
201801052, s-belalm.rashwan@zewailcity.edu.eg

***Abstract*— Image captioning has evolved into one of the most widely used tools in the modern era. Furthermore, there are built-in applications that generate and provide a caption for a specific image, all with the help of deep neural network models. Image captioning refers to the process of creating a description for an image. It necessitates recognizing the important objects, their attributes, and the relationships between the objects in an image. It generates syntactically and semantically correct sentences. In this paper, we present a deep learning model for describing images and generating captions using computer vision and machine translation. The goal of this paper is to detect different objects in an image, recognize the relationships between those objects, and generate captions. To demonstrate the proposed experiment, the dataset used is Flickr8k, the programming language used is Python3, and an ML technique called RNN and merged architectures will be implemented. The functions and structure of the various Neural networks involved will also be discussed in this paper. Image caption generation is a critical component of computer vision and natural language processing. Image caption generators can find use in image segmentation, which is used by Facebook and Google Photos, and its use can even be extended to video frames. They will easily automate a person's job of interpreting images. Not to mention its enormous potential for assisting visually impaired people. (*Abstract*)**

Keywords—Deep learning, Computer vision, sentiment analysis, image captioning, Keras (*key words****)***

```
startseq dog is running through the water endseq
```

I. INTRODUCTION

Making a computer system detect and describe objects using natural language processing (NLP) is an age-old Artificial Intelligence problem. Until recently, computer vision researchers thought this was an impossible task. With the advancements in Deep Learning techniques, the availability of large datasets, and computational power, models that generate captions for images are frequently built. Image caption generation is a task that combines image processing and natural language processing concepts in order to recognise the context of an image and describe it in a natural language such as English or another language.

captions or descriptive phrases In recent years, there has been a lot of interest in the task of automatically generating captions for images. While new datasets frequently spark significant innovation, benchmark datasets also necessitate fast, accurate, and competitive evaluation metrics to encourage rapid progress. Being able to automatically describe the content of a picture using properly formed English sentences may be a difficult task, but it could have a significant impact, such as assisting visually impaired people in better understanding the content of images on the internet. This task is significantly more difficult than well-studied image classification or visual perception tasks, which are central to the computer vision community. Deep learning methods have produced impressive results on caption generation problems. What's most impressive about these methods is that instead of requiring sophisticated data preparation or a pipeline of specially designed models, one end-to-end model is often defined to predict a caption given a photograph. Deep learning has gotten a lot of attention because it excels at a type of learning that has the potential to be extremely useful in real-world applications. The ability to learn from unlabeled or unstructured data is a huge advantage for those interested in real-world applications.

II. PROPLEM STATEMENT

The main issue in the development of image description began with object detection in the image using static object class libraries and was modelled using statistical language models.

A. Using CNN: CNN is a Deep Learning algorithm that will take a 2D matrix input image, assign importance (learnable weights and biases) to different aspects/objects in the image, and be intelligent enough to distinguish one from the other.

B. While this model was useful for naming the objects in an image, it could not tell us the relationship between them (this is standard image classification).

C. In this paper, we present a generative model based on a deep recurrent architecture that effectively generates meaningful sentences by combining recent advances in computer vision and machine translation.

D. Using an RNN: RNNs are networks with loops that allow information to persist. LSTMs are a subtype of RNN that can learn long-term dependencies.

III. METHODOLOGY

A. Task

The task is to create a system that will take an image input in the form of a dimensional array and generate an output that is syntactically and grammatically correct sentence that describes the image.

B. Corpuscular

As the corpus, we used the Flickr 8K dataset. The dataset contains 8000 images, with 5 captions for each image. The five captions for a single image aid in understanding all of the possible scenarios. The dataset includes three predefined datasets: Flickr 8k.trainImages.txt (6,000 images), Flickr 8k.devImages.txt (1,000 images), and Flickr 8k.testImages.txt (6,000 images) (1,000 images).

The images were chosen from six different Flickr groups and do not feature any well-known people or places. They are, however, hand-picked to show a variety of scenes.

C. Preparation

The descriptions are cleaned using the tokenizer class in Keras, which vectorizes the text corpus and stores it in a separate dictionary. Then, for each word in the vocabulary, a unique index value is assigned.

D. Model

Deep learning performs machine learning using an artificial neural network composed of several levels arranged in a hierarchy. The model is based on deep networks, and the flow of information begins at the first level, where the model learns something simple and then passes the output to layer two of the network, where the input is combined into something slightly more complex and passed on to the third level. This process is repeated as each level of the network generates something more complex from the input received from the ascendant level.

Neural Networks with Convolutions (CNN)

Convolutional neural networks are specialised deep neural networks that can process data with a 2D matrix input shape. Images can be easily represented as a two-dimensional matrix.

Recurrent Neural Networks (RNNs) (RNN)

The human brain has evolved in such a way that it can make sense of previous words and generate the next words while keeping these in mind, resulting in a perfect sentence.

Basic neural networks are incapable of doing so.

Recurrent neural network advancements, on the other hand, address this issue. They are networks with loops that allow information to persist for a period of time by utilising their internal states, thereby creating a feedback loop.

Long Short Term Memory networks (LSTMs) are a type of RNN that can learn long-term dependencies. Remembering information for long periods of time is practically their default behaviour, which is controlled by "gates."

While RNNs process single data points, LSTMs can process entire sequences. Not only that, but they can learn which data points are important and which can be ignored. As a result, only relevant information is passed on to the next layer.

The three main gates are: input gate, output gate, and forget gate. These gates determine whether to forget the current cell value, read a value into the cell, or output the cell value. Because the previous hidden states are passed to the next step of the sequence, the hidden states play an important role. The hidden state serves as the neural network's

memory, storing data that the neural network has previously seen. As a result, the neural network can function similarly to a human brain attempting to form sentences.

Architecture

A CNN + LSTM is used to take an image as input and output a caption. An "encoder" RNN maps the variable-length source sentence into a fixed-length vector representation, which is then used as the initial hidden state of a "decoder" RNN, which generates the final meaningful sentence as a prediction. However, we will replace this RNN with a deep CNN, which can produce a rich representation of the input image by embedding it to a fixed-length vector, by first pre-training it for an image classification task and then using the final hidden layer as an input to the RNN decoder that generates sentences.

IV. Evaluation

The entire programme is carried out in five major steps. The five major modules are implemented as follows:

A. Cleaning and Preprocessing of Data:

1. For a comfortable and quick work experience, we use Google Collaboratory: a tool that provides free GPU/TPU processing power over our local machines, which can take several hours to complete tasks that a GPU can complete in a matter of minutes.

2. Our programme begins by loading the text and image files into separate variables; the test file is stored as a string.

3. This string is used and manipulated to create a dictionary that associates each image with a list of five descriptions.

The main task of data cleaning is to remove punctuation marks, convert the entire text to lowercase, remove stop words, and remove words containing numbers.

5. Additionally, a vocabulary of all unique words from all descriptions is created, which will be used in the future to generate captions for test images.

6. Another feature of Tokenizing our vocabulary with a unique index value is part of preprocessing the data.

Because a computer cannot understand regular English words, they must be represented by numbers. After that, the tokens are saved in a pickle file. i.e. as a character stream, but with all of the information required to reconstruct it into the original object type

7. For ease of writing code, the above two preprocessing tasks can be completed manually or by using the Keras. preprocessing module.

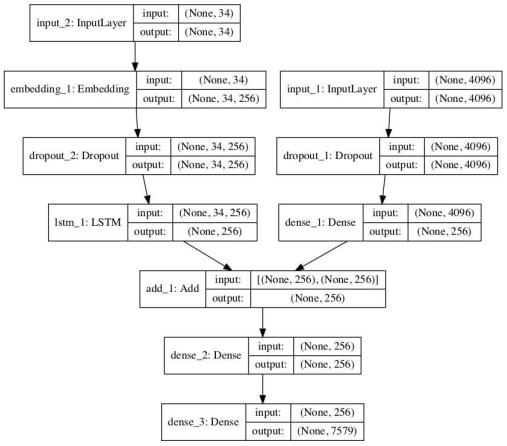
8. We then append the start> and end> identifiers to each caption, which will serve as indicators for our LSTM to understand where a caption begins and ends.

9. We will then calculate the number of words in our vocabulary and determine the maximum length.

B. Feature vector extraction:
1. A feature vector (or simply feature) is a numerical value in matrix form that contains information about an object's important characteristics, such as the intensity value of each pixel in our example. We'll eventually save these vectors in a pickle file.
2. In our model, we'll use Transfer Learning, which simply means extracting features from a pretrained model (in our case, the Xception model).
3. The Xception model is a 71-layer Convolutional Neural Network. It is trained on the well-known Imagenet dataset, which contains millions of images and over 1000 different classes from which to classify.
4. With keras.applications, Python makes it extremely simple to use this model in our code. xceptionmodule. To use it in our code, we will remove the classification layer and thus obtain the 2048 feature vector.
5. As a result, weights will be downloaded for each image, and image names will be mapped to their corresponding feature arrays.
6. Depending on your processor, this process may take several hours.

C. The CNN-RNN model is layered:
The Keras Model from Functional API will be used to stack the model. The structure will be divided into three sections:
1. Feature Extractor: This tool will be used to shrink the dimensions from 2048 to 256. We'll employ a Dropout Layer. Each of these will be included in the CNN and the LSTM. We used the

Xception model to pre-process the photos (without the output layer) and will use the extracted features predicted by this model as input.
2. Sequence Processor: The textual input will be handled by this Embedding layer, which will be followed by the LSTM layer.
3. Decoder: To make the final predictions, we will combine the output from the previous two layers and use a Dense layer. A fixed-length vector is produced by both the feature extractor and the sequence processor. These are combined and processed by a Dense layer. The final layer will have the same number of nodes as our vocabulary.



D. Model training:
1. We will train our model on 6000 images with 2048 long feature vectors each.
2. Because it is not possible to hold all of this data in memory at the same time, we will use a Data Generator. This will help us create batches of data and increase speed.
3. We will also define the number of epochs (training dataset iterations) that the model must complete during training. This number must be chosen so that our model is neither underfitted nor overfitted.
4. The method model.fit generator() will be used. And, depending on the processor, the entire process will take some time.
5. The maximum description length calculated earlier will be used as the parameter value here. It will also accept clean and tokenized data as input.
6. We'll also create a sequence creator, which will predict the next word based on the previous word and image feature vectors.
7. During training, we can use the development dataset (provided with the rest of the files) to monitor the model's performance and decide when to save the model version to the fiel.
8. We will save several models, with the final one being used for testing in the future.

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:155: UserWarning
 6000/6000 [==============================] - 667s 111ms/step - loss: 4.6765
 6000/6000 [==============================] - 655s 109ms/step - loss: 3.8881
 6000/6000 [==============================] - 658s 110ms/step - loss: 3.6261
 6000/6000 [==============================] - 657s 109ms/step - loss: 3.4797
 6000/6000 [==============================] - 662s 110ms/step - loss: 3.3769

Dataset: 6000
Descriptions: train=6000
Photos: train=6000
Vocabulary Size: 7579
Description Length: 34
Model: "model_4"
```

Layer (type)	Output Shape	Param #	Connected to
input_7 (InputLayer)	[(None, 34)]	0	[]
input_6 (InputLayer)	[(None, 4096)]	0	[]
embedding_1 (Embedding)	(None, 34, 256)	1940224	['input_7[0][0]']
dropout_2 (Dropout)	(None, 4096)	0	['input_6[0][0]']
dropout_3 (Dropout)	(None, 34, 256)	0	['embedding_1[0][0]']
dense_3 (Dense)	(None, 256)	1048832	['dropout_2[0][0]']
lstm_1 (LSTM)	(None, 256)	525312	['dropout_3[0][0]']
add_1 (Add)	(None, 256)	0	['dense_3[0][0]', 'lstm_1[0][0]']
dense_4 (Dense)	(None, 256)	65792	['add_1[0][0]']
dense_5 (Dense)	(None, 7579)	1947803	['dense_4[0][0]']

```
Total params: 5,527,963
Trainable params: 5,527,963
Non-trainable params: 0
```

E. Model validation:
1. A separate Python notebook can be created, or the same notebook can be used to test. In any case, we'll load the trained model from the previous step and generate predictions.
2. In addition to the tokenizer file, the sequence generator will be used at this point.
3. The primary step of feature extraction for the specific image under examination will be carried out.
4. The function is manually passed the path of one of the remaining 2000 test images. You can also loop through the test data set, storing the prediction for each image in a dictionary or a list.
5. The actual operation of image generation involves using the start and end sequences, as well as calling the model recursively to generate meaningful sentences.
Output:

V. RESULTS

```
startseq man in red shirt is sitting on the street endseq
```

Another example:

```
startseq two men play soccer in the air endseq
```

VI. Conclusion

We can see from the results that the deep learning methodology used here yielded successful results. When the CNN and the LSTM were properly synchronized, they were able to find the relationship between objects in images. We can use the BLEU (Bilingual Evaluation Understudy) score to compare the accuracy of the predicted captions to target captions in our Flickr8k test dataset [5, 8]. In text translation, BLEU scores are used to compare translated text to one or more reference translations. Several other neural network technologies have been used to create hybrid image caption generators similar to the one proposed here over the years.

For example, instead of the Xception model, use the VGG16 model, or the GRU model instead of the STM model. Furthermore, BLEU Score can be used to compare these models to see which one provides the best accuracy. This paper introduced us to a variety of new developments in the field of machine learning and AI, as well as the breadth of this field. In fact, several topics within this paper are open to additional research and development, while the paper itself attempts to cover the fundamentals required to create an image caption generator.

References

[1] Panicker, M. et al Image Caption Generator, 2021

Final.