Image Caption Generator Using Convolutional Neural Network Algorithm

Shaik Parvez

Department of Computer Science Engineering, Gayatri Vidya Parishad College of Engineering, Jawaharlal Nehru Technological University Kakinada (JNTUK), Visakhapatnam, India
shaikparvez977[at]gmail.com

Abstract: ***It is a very difficult challenge to automatically describe an image using a sentence from any natural language, such as English. It necessitates knowledge of both natural language processing and picture processing. The fusion of computer vision and natural language processing has received a lot of interest recently thanks to the advent of deep learning. This field is exemplified by image captioning, which teaches a computer to understand an image's visual information using one or more phrases. In addition to the ability to recognize the item and the scene, high-level image semantics also needs the ability to analyze the state, the properties, and the relationship between these things. Despite the fact that image captioning is a challenging and intricate endeavor, numerous academics have made substantial advancements. In artificial intelligence (AI), computer vision and natural language processing are used to automatically create an image's contents (Natural Language Processing). The regenerative neuronal model is developed. It is dependent on machine translation and computer vision. Using this technique, natural phrases are produced that finally explain the image. Convolutional neural networks (CNN)and recurrent neural networks (RNN) are also components of this architecture. RNN is utilized for phrase creation, while CNN is used to extract features from images. The model has been taught to produce captions that, when given an input image, almost exactly describe the image. On various datasets, the model's precision and the fluency or command of the language it learns from visual descriptions are examined. These tests demonstrate that the model frequently provides precise descriptions for an input image.***

Keywords: Convolutional Neural Network, Long Short-Term Memory (LSTM), Recurrent Neural Network, TensorFlow, Keras, NumPy

1. Introduction

The inclination to characterize an image with a wealth of information about it by simply taking a quick glance is a fundamental human talent. Artificial intelligence and machine learning researchers have long sought to build computer systems that can mimic human talents. Research has advanced in a number of areas, including the identification of objects in an image, attribute classification, picture classification, and the categorization of human actions. Making an image caption generator system—a computer program that can identify a picture and generate a description using natural language processing—is a difficult task. Creating a caption for a picture entails a variety of problems, such as comprehending the deeper levels of semantics and expressing those semantics in human-understandable sentences.

The computer system must learn the connections between the items in an image in order to comprehend higher levels of semantics. Natural language is typically used in human communication; therefore creating a system that can generate descriptions that people can understand is a difficult task. The process of creating captions involves numerous processes; including comprehending how items are represented visually, figuring out how the objects relate to one another and creating captions that are both linguistically and semantically accurate. The objectives of this paper include deep learning based detection, recognition, and caption generation.

We must first comprehend how critical this issue is to the scenarios that occur in the actual world. Let's look at a couple of scenarios when a solution to this issue would be quite beneficial.

Aid to the Blind: We are able to develop a product that will enable blind people to navigate the highways independently. To accomplish this, first, translate the scene into text, and then speak the text. Both are now well-known Deep Learning applications. Autonomous driving is one of the major obstacles, and if the environment around the automobile can be captioned correctly, the self-driving system will benefit. As every image could be transformed into a caption before being searched on, automatic captioning might help make Google Image Search as excellent as Google Search.

For any image that shows on the website, it's best practice in web development to include a description so that an image can be heard or read as well as seen. This facilitates access to the web material. CCTV cameras are widely used today, but in addition to seeing the outside world, if we can also produce pertinent captions, we can alert authorities as soon as some nefarious behavior takes place. This may contribute to a decrease in some accidents or crimes.

Real-time video can also be described using this concept.

2. Literature Review

Kojima et al [1] used case structure, action hierarchy, and verb patterns to generate captions of human activities in a fixed environment.

Patrick Hède et al [2] proposed a method for image caption generation, which involves a series of object names stored in a database called Dictionary. Such method can generate caption for only fixed content, but it fails to generate captions for real world.

Farhadi, A., et al [3] proposed an information retrieval-based image captioning system, where a score is generated for every object in an image and the score is compared with other images to generate captions.

Hodosh, M., et al [4] proposed a ranking based image captioning system, where the captions are generated with the help of sentence-based image captioning ranking system.

Yang, Y., et al [5] proposed a sentence making strategy, which builds a semantic phrase using verbs, nouns, and prepositions, uses trained detectors to find the image, then estimates the image using data from the English corpus.

Socher, R., et al [6] proposed a decision tree based recursive neural networks to represent the visual meaning of the image.

You, Q., et al [7] proposed a model of semantic attention, which deals with the semantics, stored in a hidden layer of neural networks and fusion them to gain more semantically sentence.

Vinyals, O., et al [8] proposed a generative model that combines computer vision and machine learning to generate captions for a given image.

3. Methodology

To understand the context of an image and explain it in a natural language like English, an image caption generator uses computer vision and natural language processing techniques. The goal of our project is to become familiar with the ideas of a CNN and LSTM model and to implement CNN with LSTM to create a workable model of an image caption generator.

Fig-1 illustrates the suggested process for creating captions using deep learning for item detection and recognition. Convolution Neural Network (CNN) for feature extraction and scene categorization, Recurrent Neural Network (RNN) for human and object attributes, RNN encoder, and a fixed length RNN decoder system make up this system.

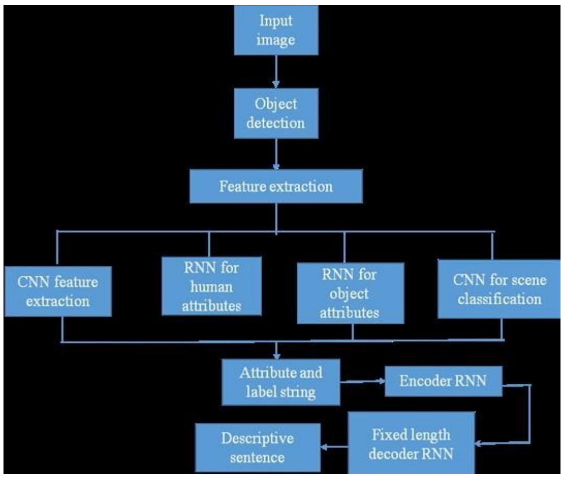


Figure 1

The following are the steps for utilizing neural networks to recognize objects and extract features to create captions.

Step 1: Object detection
In this step, the R-CNN region proposal approach is used to detect the objects in the input image.

Step 2: Feature Extraction
In this stage, principal component analysis with NumPy is used to extract the image's features. RNN is used to recognize objects and human traits whereas CNN is used to classify scenes.

Step 3: Creating attributes
In this stage, the characteristics with their label strings were defined using the features that the neural networks had retrieved.

Step 4: Encoder and Decoder
The label strings were put through an encoder RNN in this phase to put them into the correct format, and the resulting variable-length string was put through a fixed-length decoder to turn it into a fixed-length descriptive phrase.

Fig-2 represents the model diagram of the project

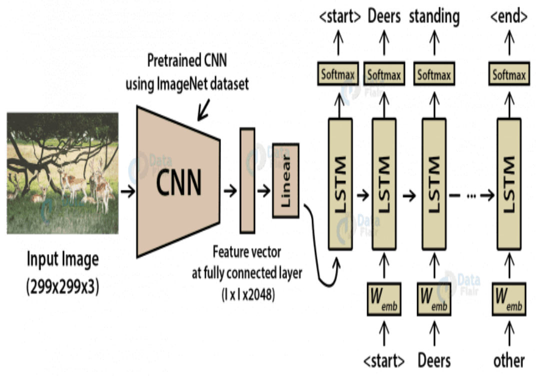

<start> Deers standing <end>
Softmax
Pretrained CNN using ImageNet dataset
CNN
Linear
LSTM
Input Image (299x299x3)
Feature vector at fully connected layer (1 x 1 x 2048)
W emb
<start> Deers other

Figure 2: Model diagram

4. Experimental Setup

4.1 System Requirements

- CPU and GPU with at least 8GB of RAM
- i5 processor
- NVDIA graphic card

4.2 System Requirements

- TensorFlow
- Keras
- NumPy
- Pillow
- Jupyterlab
- Tqdm

4.3 Data Collection

We will make use of the Flickr 8K dataset to create the image caption generator. We will use the smaller Flickr8k dataset instead of other larger ones like the Flickr 30K and MSCOCO datasets because training the network on those can take weeks. We can create better models thanks to a large dataset.

- Flicker8k Dataset
- Flickr 8k text

The main file of our dataset, Flickr8k.token, which contains the names of the images and their corresponding captions, separated by newlines ("n"), can be found in the Flickr 8k text folder.

5. Convolutional Neural Network

Convolutional Neural Networks are customized deep neural networks that are capable of processing data with input shapes similar to a 2D matrix. CNN is particularly helpful when working with photos and can readily express images as a 2D matrix. Fig-3 illustrates the working of deep CNN. Images are scanned from top to bottom and left to right to extract key details, which are then combined to identify the images. It can handle images that have been resized, rotated, translated, and perspective-shifted.

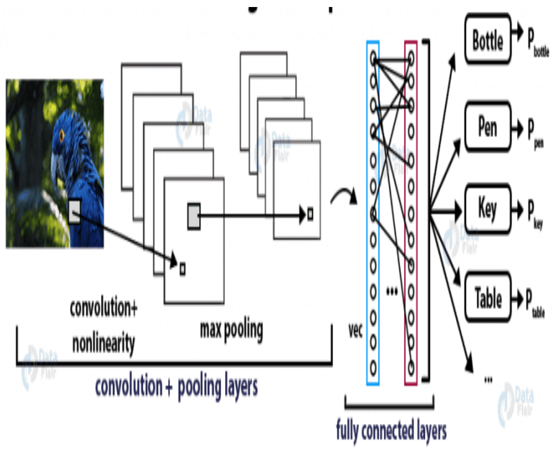


Figure 3: CNN Layers

5.1 Types of CNN Layers

There are 4 types of CNN layers (fig-4). They are
- Convolutional Layer
- ReLU Layer
- Pooling Layer
- Fully Connected Layer

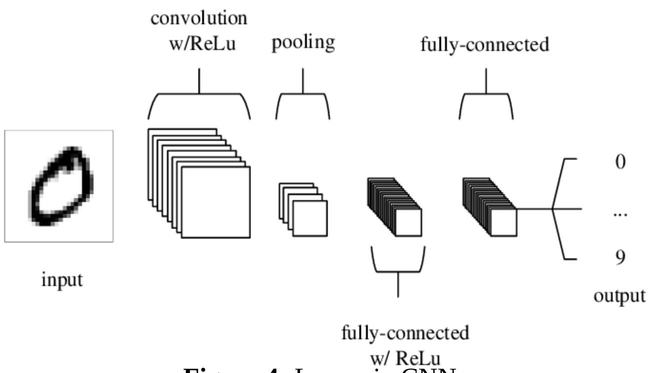


Figure 4: Layers in CNN

5.1.1 Convolutional Layer

In order to extract features from the input image, a convolution layer alters it. This transformation involves convoluting the picture with a kernel (or Filter). A kernel (also known as a filter) is a tiny matrix having dimensions that are less than those of the picture to be convolved. This filter moves across the input image's height and breadth, computing the dot product of the filter and the image at each spatial location.

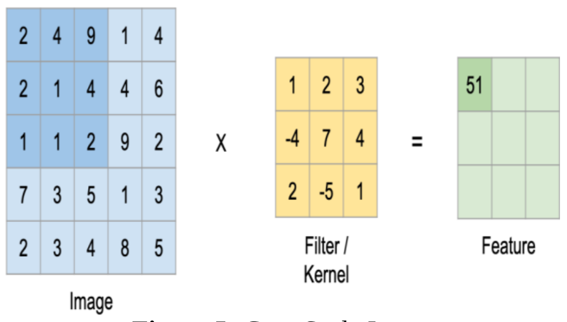


Figure 5: Gray Scale Image

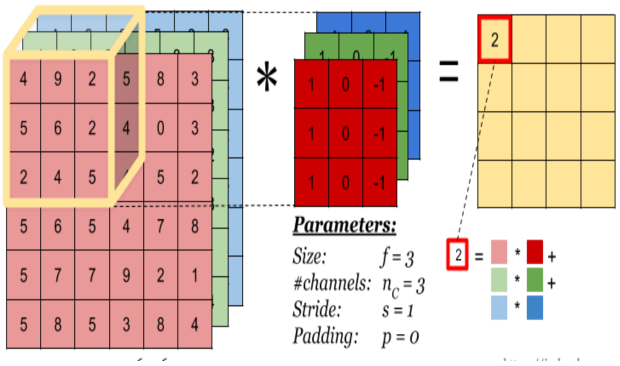


Figure 6: RGB Image

(a) Padding

In order to stop our image from shrinking, we use padding. We add layers of zero to stop shrinking.

Types of Convolutions in padding:
- Valid: No padding takes place because no need to stop shrinking of image
- Same: Pad so that the output image size is the same as the input image size.

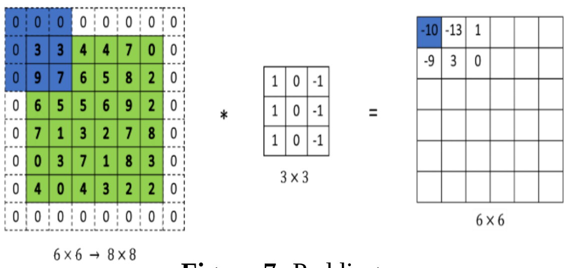


Figure 7: Padding

(b) Striding

The number of pixels shifted across the input matrix called the stride. The filters are moved to 1 pixel at a time when the stride is 1. The filters are moved to 2 pixels at a time when the stride is 2, and so on. Stride length is the length at which the filter slides.

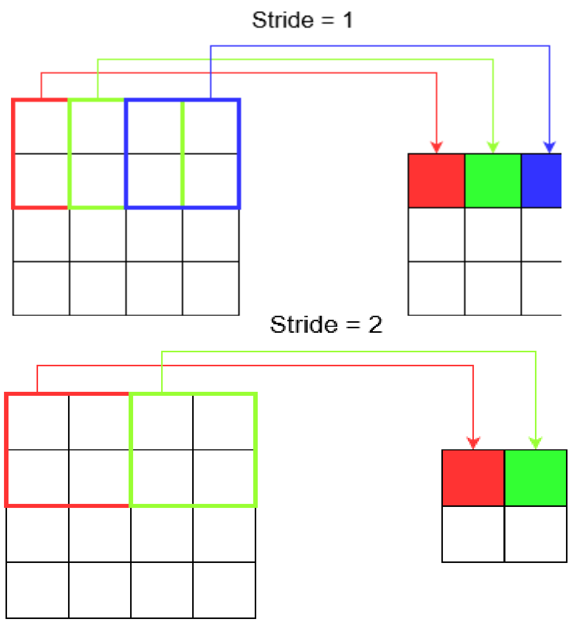


Figure 8: Striding

5.1.2 ReLU Layer

Every negative value from the filtered photos is removed and replaced with zeros in this layer. To prevent the values from adding up to 0, this is done. Rectifies only when the input exceeds a certain threshold does the Linear Unit (ReLU) transform function activate a node; otherwise, the output is zero. However, if the input climbs above the predetermined threshold, the input has a linear relationship with the dependent variable.

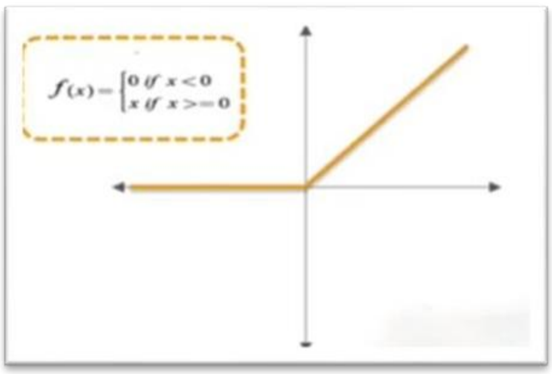


Figure 9: ReLU Function

5.1.3 Pooling Layer

The input image's size is decreased by using a pooling layer. Typically, a pooling layer is added to speed up computation and strengthen some of the identified features. Filter and stride are also utilized in pooling operations.

- Max Pooling: Here, the maximum value is chosen from each patch of the feature map to produce a reduced map.
- Average Pooling: Here, the average value is chosen from each patch of the feature map to produce a reduced map.

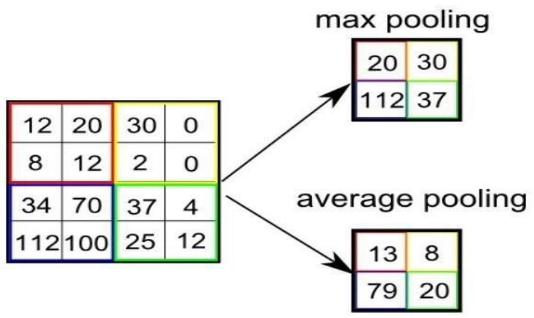


Figure 10: Pooling layer

5.1.4 Fully Connected Layer

This layer is located at the convolution neural network's conclusion. The earlier layer's features map is flattened to a vector. The complicated interactions between high level features are then captured by feeding this vector into a fully linked layer.

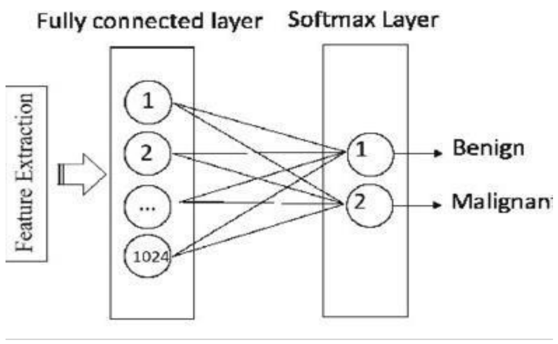


Figure 11: Fully Connected Layer

6. Long Short-Term Memory

Recurrent neural networks (RNNs) of the LSTM variety are very effective at solving sequence prediction issues. We can anticipate the following word based on the prior text. By addressing the short-term memory restrictions of RNN, it has distinguished itself as an effective alternative to regular RNN. Through the use of a forget gate, the LSTM may carry out relevant information while processing inputs and reject irrelevant information.

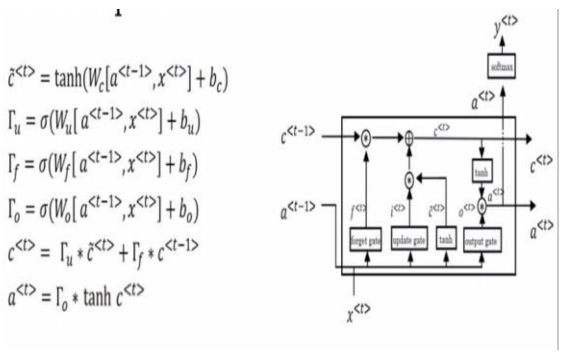

$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$
$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$
$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$
$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$
$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$
$a^{<t>} = \Gamma_o * \tanh c^{<t>}$
$y^{<t>}$
softmax
$a^{<t>}$
$c^{<t-1>}$
$c^{<t>}$
tanh
$a^{<t-1>}$
forget gate
update gate
tanh
output gate
$x^{<t>}$

Figure 12: LSTM Structure

LSTM units are horizontally connected (parallel). The input for the following time stamp is the output of the preceding one.

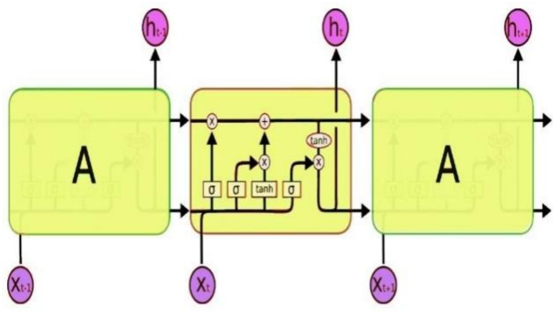


Figure 13: LSTM Units Connected in Parallel

7. Implementation

Steps performed while building the model has been discussed below.

a) **First, we import all necessary packages**.

- TensorFlow
- NumPy
- Keras
- Tqdm
- Pillow
- Jupyterlab

b) Dataset loading for model training

We have a file called Flickr 8k.trainImages.txt in our Flickr 8k test folder that has a list of 6000 picture names that we will use for training.

More functions are required to load the training dataset, including:

- load_photos(filename): returns the list of image names after loading the text file as a string.
- load_clean_descriptions(filename, image): builds a dictionary with captions for each picture in the collection of pictures.
- load_features(image): provides us with the list of image names and the feature vector associated with them, both of which we previously collected from the Xception model.

c) Tokenizing the vocabulary

Because English words are not understood by computers, we must represent them using numbers. Therefore, we will assign a distinct index value to each word in the lexicon. The tokenizer function from the Keras package is what we will use to generate tokens from our vocabulary and store them to the "tokenizer.p" pickle file. Our vocabulary is 7577 words strong. The maximum length of the descriptions is determined. This is significant for choosing the parameters for the model structure. The maximum description length is 32.

d) Create data generator

Let's first have a look at what our model's input and output will entail. We must give the model input and output for training in order to convert this task into a supervised learning task. Our model needs to be trained on 6000 photos, each of which has a 2048-length feature vector and a caption that is likewise represented as a number. Because it is impossible to store this much data in memory for 6000 photos, we will use a generator method that will produce batches.

e) The CNN-RNN model is defined.

The organizational structure of the model will be specified using the Keras Model from the Functional API. It will include three main sections:

- Feature Extractor: With a dense layer, we can shrink the 2048 node feature that was retrieved from the image to 256 nodes.
- Sequence Processor: The textual input will be handled by an embedding layer, then by an LSTM layer.
- Decoder: We will process by the dense layer after integrating the output from the aforementioned two layers in order to arrive at the final forecast. The number of nodes in the final layer will be equal to the size of our vocabulary.

f) Training the model

By creating the input and output sequences in batches and fitting them to the model using the model.fit generator() method, we will be using the 6000 training photos to train the model. The model is also saved to our model's folder. Depending on the capabilities of your equipment, this will take some time.

g) Testing the model

We will create a second file called testing caption generator.py to load the learned model and produce predictions. We will use the same tokenizer.p pickle file to extract the words from their index values because the predictions contain the maximum length of index values.

8. Results

Following model implementation, the output is shown in fig. 14 below in terms of accuracy. RNN and CNN algorithms are both used in the model's construction.

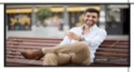
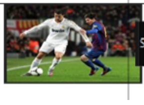
IMAGE	CAPTION GENERATED	ACCURACY
	start boy in pink bathing suit is jumping into pool end	7/9 = 78%
	start man riding bike on the road end	6/6 = 100%
	start dog runs through the grass end	5/5 = 100%
	start man in black wetsuit is surfing end	6/6 = 100%
	start young boy in swim trunks is jumping on the beach end	3/10 = 30%
	man in black shirt and jeans is sitting on bench	7/10 = 70%
	start two boys playing soccer on field end	6/6 = 100%

Figure 14: Output

AVERAGE ACCURACY
Generating > 70% accurate captions for 64 out of 100 images
Generating 50% to 70% accurate captions for 16 out of 100 images
Generating < 50% accurate captions for 20 out of 100 images

9. Conclusion

By developing an image caption generator, we developed a CNN-RNN model in this advanced model Python project. It's important to keep in minds that because our model is data-dependent, it cannot predict terms that don't exist in the English language. We worked with a tiny dataset of 8000 photos. We need to train on datasets bigger than 100,000 photos for production-level models in order to get models with higher levels of accuracy.

Using neural networks and deep learning, a method for creating image captions is provided. The suggested method was tested on the Flickr 8k dataset. Compared to the currently available image caption generation generators, the proposed deep learning methodology produced captions with more descriptive meaning. In the future, a hybrid picture caption generator model may be created for captions that are more accurate.

References

[1] Kojima, Atsuhiro & Tamura, Takeshi & Fukunaga, Kunio. (2002). Natural Language Description of Human Activities from Video Images Based on Concept Hierarchy of Actions. International Journal of Computer Vision. 50. 171-184. 10.1023/A:1020346032608.

[2] Patrick Hède, Pierre-Alain Moëllic, Joël Bourgeoys, Magali Joint, and Corinne Thomas. 2004. Automatic generation of natural language descriptions for images. In Coupling approaches, coupling media and coupling languages for information retrieval (RIAO '04). LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, Paris, FRA, 306–313.

[3] Farhadi, A. *et al.* (2010). Every Picture Tells a Story: Generating Sentences from Images. In: Daniilidis, K., Maragos, P., Paragios, N. (eds) Computer Vision – ECCV 2010. ECCV 2010. Lecture Notes in Computer Science, vol 6314. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-15561-1_2

[4] Hodosh, M., Young, P., & Hockenmaier, J. (2013). Framing image description as a ranking task: Data, models and evaluation metrics. Journal of Artificial Intelligence Research, 47, 853-899. https://doi.org/10.1613/jair.3994

[5] Yang, Y., Teo, C.L., Daumé, H., & Aloimonos, Y. (2011). Corpus-Guided Sentence Generation of Natural Images. EMNLP.

[6] Socher, R., Karpathy, A., Le, Q., Manning, C., & Ng, A. (2014). Grounded Compositional Semantics for Finding and Describing Images with Sentences. Transactions of the Association for Computational Linguistics, 2, 207-218. Retrieved from https://transacl.org/index.php/tacl/article/view/325

[7] You, Q., Jin, H., Wang, Z., Fang, C., & Luo, J. (2016). Image Captioning with Semantic Attention. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4651-4659.

[8] Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3156-3164.