The 8th IEEE International Conference on E-Health and Bioengineering - EHB 2020
Grigore T. Popa University of Medicine and Pharmacy, Web Conference, Romania, October 29-30, 2020

Assisted environment comprehension using a mobile terminal application

George Nitroi[1], Corneliu Florea[1], Mihai Badea[1]
[1]Image Processing and Analysis Laboratory (LAPI), University "Politehnica" of Bucharest, Romania
gnitroi@imag.pub.ro; corneliu.florea@upb.ro; mihai_sorin.badea@upb.ro

***Abstract*—Based on computer vision maturity assistive technologies have been developed for visually impaired people. This paper assumes an embedded device with mounted camera that provides efficient description of the environment. Technically, our solution is built upon the deep learning paradigm using a combination of convolutional layers for image description and recurrent ones for phrase consistency. For implementation we have successfully tested our solution on an embedded platform. (*Abstract*)**

Keywords—visually impaired; scene captioning; deep neural networks;embedding processing;.

I. INTRODUCTION

Visually impaired people face special difficulties in daily life, sometime having issues understanding the environment they are currently in. Computer vision in conjunction with vision sensors can in nowadays make their life easier [1]. Being able to automatically describe the content of an image using properly formed English sentences is a challenging task, but it could have great impact by helping visually impaired people better understand their surroundings. A wearable camera can feed those images depicting the surrounding environment. These images are then be used to generate captions that can be read out loud to the visually impaired so that they can get a better sense of what is happening around them to obtain a so-called "environment comprehension"

We propose a distributed system that implies a web server that supports a REST API interface running a machine learning algorithm capable of producing a text output if an image is given. Keeping all the intensive processing on the server side, will enable each mobile terminal that has an Android operating system and camera to use this server in order to generate captions with a relative low latency. A written caption will be generated as a response for each picture sent to the server.

II. PRIOR WORK

Image captioning is an active topic of research, and its use in helping people with reduced visual capacity quickly became one of the main applications [2 - -7]. Some of this solutions rely on accesories such as helmets [2,5,8], glasses[7], a pair of headphones [6] or a camera [4] devices that we do not tipically carry on a daily basis. Older solutions used to run those classification/machine learning algorithms on premise, which resulted in an architecture that is not scalable and easy to maintain and update [3]. Environment understanding, object detection and recognition are crucial skills that we use on a daily basis and convolutional neural networks made great progress in order to replicate those skills [1]. Due to advances in internet speeds, cloud computing quickly became the preffered solution in order to run complex algorithms that require intensive processing power such as machine learning models.

All previous implementations acknowledge that wearable technologies are enabling plenty of new applications for computer vision and use small factor cameras in their systems, embedded in a pair of glasses or wore around the neck, in order to make those systems reliable and comfortable for their users. State of the art implementations use convolutional neural networks in order to detect and classify the objects of interest and recurrent neural networks in order to summarize the image-contents in words.

III. DATABASE AND MODEL

In this section we describe the database used and respectively the deep learning model used to predict the captioning of a given query image.

A. Database

The database used for this study was COCO dataset from 2015 from cocodataset.org portal, used in Large-scale Scene Understanding (LSUN) workshop [9]. The facilities offered by this important database are the large number of annotated pictures and the increased number of captions for each image. Each pictured had at least 5 captions that allowed increased flexibility in training and validation. Examples of image from the database and the captioning are showed in figure 1.

Fig. 1. Examples of image from the database and the associated captioning

For *training*, one image was picked at random from the training lot along with one random caption. We considered a training epoch passing forward a number of random selected images equal to the number of training images.

B. Model

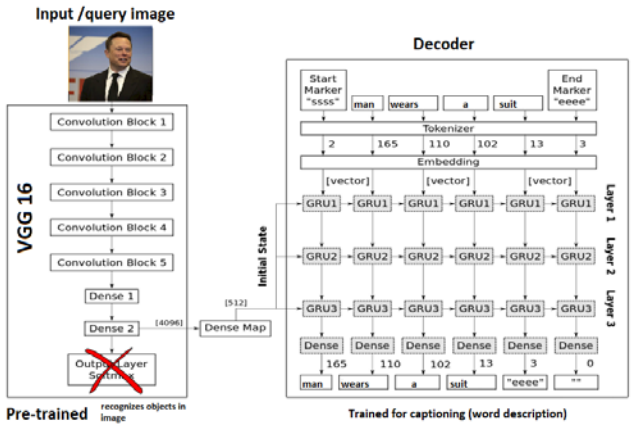

Fig 2. Schematic of the proposed solution

The model was inspired by the work of [Magnus Erik Hvass Pedersen](http://www.hvass-labs.org/)[10] who developed this network for image captioning. This solution showed good results in environment understanding and object recognition in a wide variety of scenes.

The machine learning model consists of 2 neural networks, a VGG16 [10] and a recurrent neural network. The detailed model is presented in figure 2 The VGG16 comes in the pre-trained form as it has its weights already trained on ImageNet. In contrast the recurrent network was trained from scratch. Thus, in order to speed up the training process some pre-processing was needed on the training and validation set. We have saved the summarized information that VGG16 returns under a 4096 elements vector on disk and loaded those values in the recurrent model for adjusting its weights.

For the vocabulary there have been chosen a number of 10000 most used word in the English vocabulary. A two-step process is used to convert text into numbers that can be used in a neural network. The first step is to convert text-words into so-called integer-tokens. The second step is to convert integer-tokens into vectors of floating-point numbers using a so-called embedding-layer.

C. Model

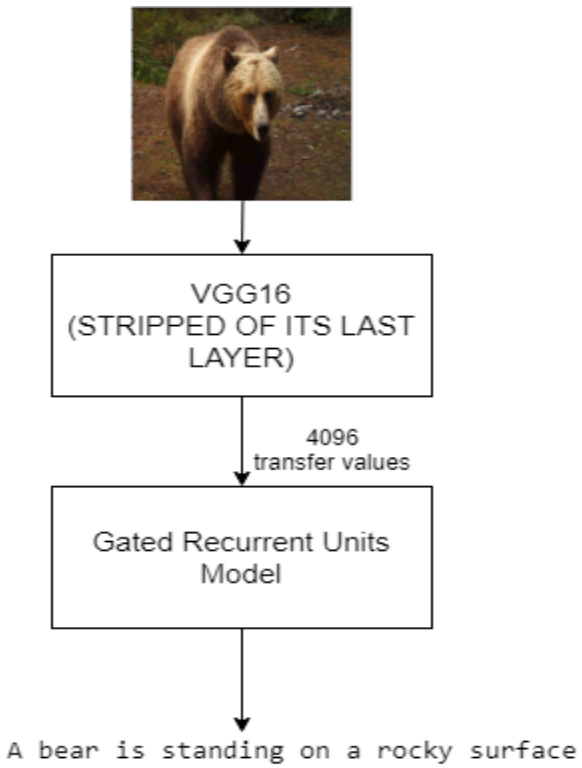

Fig 3. Caption Model

As seen in the figures 2 and 3, we have connected the penultimate layer of VGG16 to a recurrent neural network which consists of 3 layers of Gated Recurrent Units that have 512 internal states. The transfer-values are vectors of length 4096 but the size of the internal states of the GRU units are only 512, so we use a fully-connected layer to map the vectors from 4096 to 512 elements.

Using the before mentioned data pre-processing allowed us to train the recurrent model with a batch size of 3000 pictures using a nVidia GTX 1060. During our training process we have noticed that a bigger batch size led to better results.

The learning is with a RMSprop optimizer with a variable learning rate that had run for 20 epochs. By this time the loss function was no longer decreasing significantly.

IV. System Implementation

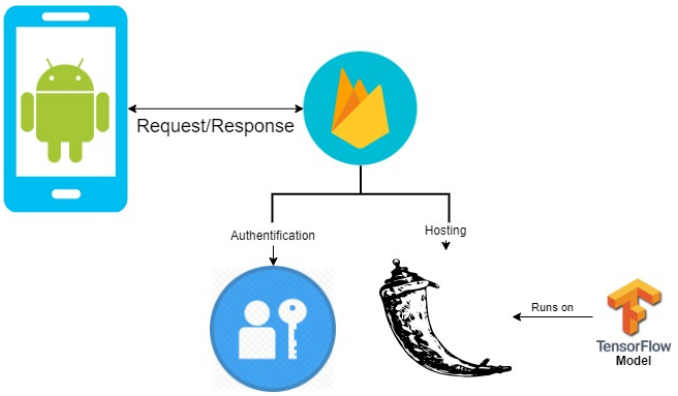

Fig 4. System architecture

The system we propose is an android applications that acts as an interface with the web service on which we will run our machine learning model. We have developed an intuitive and easy to use interface that makes it easy to authenticate into this service and send visual content to the service and get in return a textual description. This is done using a REST API built in into the server that receive a GET request from our terminal with the visual content and returns the image caption as response. For our backend we have used Firebase Authentification and Hosting. The hosting component allowed us to upscale and downscale the service running the machine learning model in order to cover the user demand.

The server has both a web application and an API service. Built using Flask, the server has been encapsulated together with the machine learning model inside a docker image and hosted on Google Cloud. The model was written using the Python version of tensorflow which made managing the dependecies of the applications easy. This type of architecture makes the service available behind a single domain name system that handles the load-balancing

V. Results

The service was tested using 3 different scenarios. We deployed it on a local host that has an Nvidia 1060 GTX and tensorflow GPU support enabled, on the same local host but with GPU support disabled and on a E2 General purpose instance type in Google Cloud. We have measured the response times for each type of deployment.

Environment	Response time
Local GPU enabled	0.5-1 s
Local GPU disabled	10-12 s
E2 General	15 s

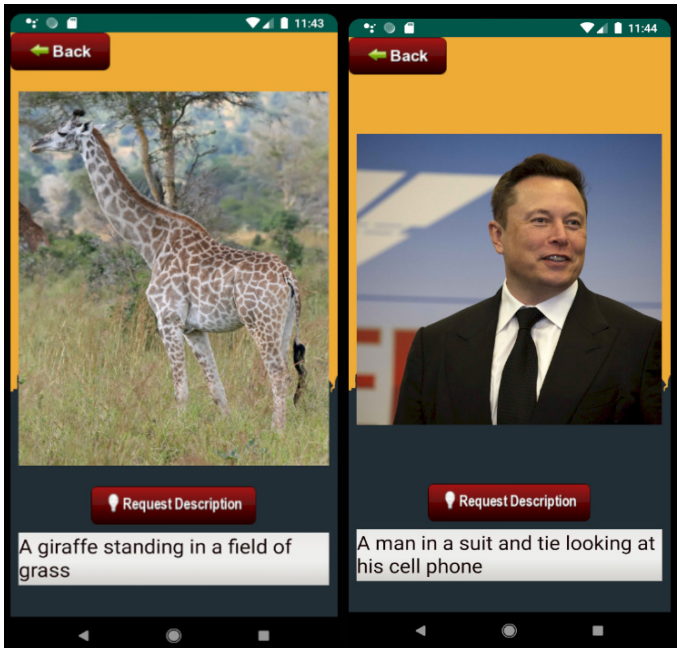

Fig 5. System architecture

In **Fig** 5. you can see 2 captions produced by our application. As seen in the first example, the model is able to correctly caption images where background information is reach and distinguish correctly between the foreground and background. In the second picture, the background information is scarce, leaving the model struggling to make a lengthy proposition. The model identifies the man and what he is wearing correctly, but also adds incorect information.

When calculating the accuracy on the validation set, the score was obtained by averagin the value for each picture got on the number of available captions. We have used BLEU(2002) and METEOR(2005) in order to calculate the accuracy.

BLEU(2002) is based on the degree of n-gram overlapping between the strings of words produced by the machine and the human translation references. METEOR(2005) addresses weaknesses in BLEU(2002) such as lack of recall and lack of explicit word matching, making it more precise.

TABLE I. Model Accuracy

Metric	**Accuracy**
BLEU(2002)	34.08%
METEOR(2005)	34.38%

VI. Conclusions

In this paper, we presented a solution of image captioning for visually impaired using 2 neural networks, the VGG16 deep learning architecture and GRU model. The recurrent network was trained and tested on the MSCOCO dataset. Both models are then deployed on a web server to make them accessible within the Android application. The user either selects image from gallery or captures new image using the smartphone camera. The selected image will be uploaded to our server which runs our proposed image captioning approach. The generated captions will be sent back to the application again and displayed to the user.

Acknowledgment

This work is partially supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS – UEFISCDI, number PN-II-RU-TE-2019-4-07331

VII. References

[1] Marco L.; Farinella (G.M) (Editors**)*:*** “Computer Vision for Assistive Healthcare”, Academic Press 2018.

[2] Jiang, B., Yang, J., Lv, Z. and Song, H, "Wearable Vision Assistance System Based on Binocular Sensors for Visually Impaired Users", *IEEE Internet of Things Journal, vol. 6, no. 2, pp. 1375-1383, April 2019,*

[3] Costa, P., Fernandes, H., Barroso, J., Paredes, H. and Hadjileontiadis, L.J., "Obstacle detection and avoidance module for the blind,"," *2016 World Automation Congress (WAC), Rio Grande,* pp. pp. 1-6, 2016.

[4] Kevin M. Irick ; Peter A. Zientara ; Jack Sampson ; Vijaykrishnan Narayanan, ""Cognitive cameras: Assistive vision systems"," *International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES), Amsterdam,* 2015.

[5] Murillo, A.C., Gutiérrez-Gómez, D., Rituerto, A., Puig, L. and Guerrero, "Wearable omnidirectional vision system for personal localization and guidance"," *IEEE Computer Vision and Pattern Recognition Workshops,* 2012 .

[6] Laubhan, K., Trent, M., Root, B., Abdelgawad, A. and Yelamarthi, K, ""A wearable portable electronic travel aid for blind"," *International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT),* 2016 .

[7] K. Zhan, F. Ramos and S. Faux, "Activity recognition from a wearable camera," 2012 12th International Conference on Control Automation Robotics & Vision (ICARCV), Guangzhou, 2012, pp. 365-370, doi: 10.1109/ICARCV.2012.6485186.

[8] Zhan, K., Ramos, F. and Faux, S., "Activity recognition from a wearable camera"," *12th International Conference on Control Automation Robotics & Vision (ICARCV), Guangzhou,* 2012 .

[9] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C.L., 2014, September. Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer, Cham. Available: https://cocodataset.org/.

[10] https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/22_Image_Captioning.ipynb