Mechanical & Industrial Engineering
UNIVERSITY OF TORONTO

AUTOMATIC IMAGE CAPTIONING AND SPEECH GENERATION

Hanwen Liang	1003868291
Bo Li	1004032352
Abhijith S Nair	1004217144
Yifan Zhang	1004245952
Jiancheng Sun	1004289003
Ge Zhang	1004460610

REPORT SUBMITTED IN PARTIAL FULFILLMENT BY GROUP 15 FOR MIE 1624 INTRODUCTION TO DATA SCIENCE AND ANALYTICS IN MASTER OF ENGINEERING AT UNIVERSITY OF TORONTO

November 2018

1. OBJECTIVE

Visual impairment, also known as vision impairment is a decreased ability to see to a degree that causes problems not fixable by usual means, such as glasses. According to the World Health Organization, 285 million people are visually impaired worldwide, including over 39 million blind people [1]. Technology has the potential to significantly improve the lives of visually impaired people. Applications such as Braille displays, screen readers etc. enable the blind to use mainstream computer applications providing them information which was previously inaccessible. Implementation of automatic image captioning into applications such as web interfaces has successfully attracted visually impaired surfers.

Vision-to-Language problems have always been challenging, since it requires two or more different forms of information. In 2014, researchers at Google released a paper, *Show and Tell: A Neural Image Captioning Generator* [3], which widely addressed this issue. However, using only the classical image model captioning technique wouldn't be beneficial for the visually challenged since it may lead to **asthenopia** or eye strain, an eye condition that manifests through nonspecific symptoms such as fatigue, pain in or around the eyes, blurred vision, headache, and occasional double vision.

In this project, we have addressed this issue and implemented a model by fusing both image captioning prototype and a text-to-speech synthesizer. The project aims at generating captions for images using neural network models, a variant of Recurrent neural network (RNN) coupled with a Convolutional neural network (CNN). The captions are then transformed into natural-sounding speech using the IBM Text to Speech service [4].

2. METHODOLOGY

2.1 Architecture

Our model, *The Visual Speaker,* is based end-to-end on a neural network consisting of a CNN type model at image end, an RNN type model at language end, followed by a speech synthesizer. It can generate complete sentences in natural language from an input image, as shown by the example in Figure 1. The neural network model comprises of two parts: the encoder and the decoder. In general, the encoder takes in a raw image and extracts feature representations from it, while the decoder utilizes the feature information to generate meaningful sentences to describe the image.

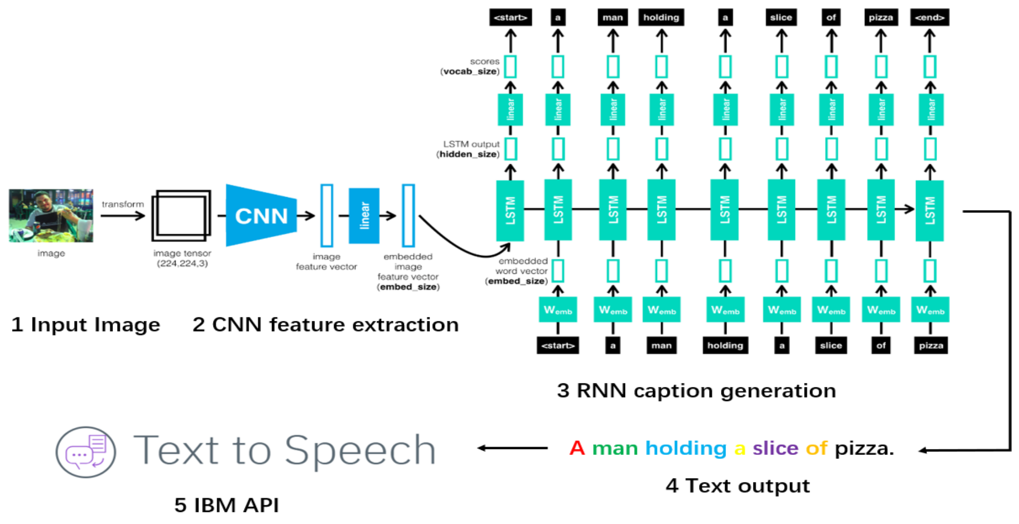


Figure 1: Architecture of Visual Speaker

2.2 Mathematical Interpretation

In this system, the goal is to generate a description of the objects, actions or events of an image:

$$S = f(I)$$

where I is an RGB image and S is the target description, f is the model we try to build.

We define the training of our model as a process to maximize the probability of the correct description when given an image:

$$\theta^* = \arg max_{\theta} \sum_{(I,S)} \log p(S|I;\theta)$$

where θ are the parameters of our model, I is an image and S is its correct description.

Suppose we have a target output sentence with N words, then we can rewrite S as $(S_0, S_1, \cdots, S_{N-1})$, where S_i represents each word. Applying the chain rule, we can get:

$$\log p(S|I) = \sum_{t=0}^{N} \log p(S_t | I, S_0, \cdots, S_{t-1})$$

where the θ is dropped for convenience, S_t is the word at step t.

As we mentioned in part I, there are two main components in our model: CNN and RNN. In the component of CNN, the visual feature of RGB image is embedded to as the input to the RNN.

$$\mu = W_{\mu}\,(CNN(I))$$

where W_{μ} represents the embedding of visual feature.

In the component of RNN, we have a vocabulary list of words. For example, we can choose top M words based on the appearing frequency in the captions data set and add S additional words for start, end and padding of the caption sentence. So, we have a vocabulary list of $M + S$ words. Particularly, each word is represented by a one-hot vector S_t:

$$x_t = W_t S_t, t \in \{0 \cdots N-1\}$$

where W_t is word embedding parameter.

Now we have mapped the image and words into the same space and the next step is to decode the features u, x_t and internal hidden parameter h_t into a probability to predict the word at current time:

$$p_{t+1} = LSTM(\ \mu, x_t, h_t), t \epsilon \{0 \cdots N-1\}$$

where LSTM means Long Short-Term memory [2].

We will use Beam Search is our model to pick top-K sentences out of the sentence with the highest probability since there are always complex intermediate connections of words in a sentence. And it has been proved that beam search can get a better performance with respect to just simply choosing the result with the highest probability [3].

2.3 Model Components

Our method comprises of the following components:

2.3.1 Encoder: Convolutional Neural Network (CNN)

For our model, the first step is to pre-process a raw image to produce the input that can be used for the model training. It has been convincingly shown that CNNs can produce a rich representation of the input image by embedding it to a fixed-length vector, such that this representation can be used for a variety of vision tasks like image classification and image face recognition. Thus, for the first part, we make use of CNN to pre-process the input image and extract a set of feature vectors which we refer to as annotation vectors. The extractor will produce N vectors and each of them is a D-dimensional representation corresponding to part of the input image.

$$\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_N\}\ , \alpha_i \in\ R^D$$

In order to allow the decoder to selectively focus on certain parts of the input image we make the model select a subset of all the feature vectors. And this could make it possible to obtain a correspondence between the vectors and portions of the 2-D image, which is also the reason why we extract features from a lower convolutional layer instead of fully connected layer.

2.3.2 Decoder: Recurrent Neural Network (RNN)

LSTM-based sentence generator is of great importance to translation and sequence generation problems and it is applied to deal with vanishing and exploding gradients in RNN model. The LSTM model is trained to predict each word of the target sentence after it has seen the image as well as all preceding words. [3]

The architecture of LSTM is shown in figure 2. The core of the LSTM model is a memory cell c encoding knowledge at every time step of what inputs have been observed up to this step. There are three gates in the LSTM memory block which control whether to forget the current cell value (forget gate), if it should read its input (input gate) and whether to output the new cell value (output gate).

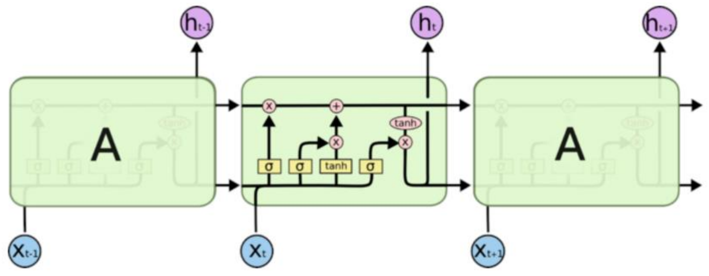


Figure 2: LSTM Architecture

This architecture can well deal with exploding and vanishing gradients as we mentioned before. For this part, we just explain the methods to implement our model and will not focus too much on the technical theories and formulas.

2.3.3 Application Programming Interface (API)

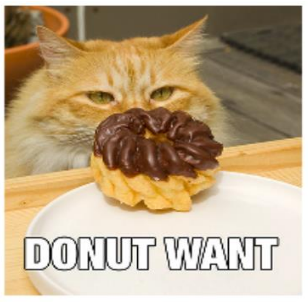
In classical image captioning models, the final output is a sentence in natural language (English) which describes the input image. However, in our model we intend to produce voice that describes the image, thus elevating the ease of information access for the visually impaired.

The IBM text to speech service provides APIs that use IBM's speech-synthesis capabilities to synthesize text into natural-sounding speech in a variety of languages, dialects and voices. Given the text, the audio is streamed back to the client with minimal delay. [4]

3. EXPERIMENTAL IMPLEMENTATION

We performed the experiment of image captioning to quantitatively and qualitatively evaluate our CNN-RNN model and application.

Dataset We conducted the experiment based on MSCOCO dataset [3]. It consists of around 80K training images, 40K validation images and 40K test images. Each image comes with five descriptive captions, as in figure 3. We used the most commonly used metrics BLEU-4 to quantitatively evaluate our performance.



```
A cat sitting in front of a delicious chocolate donut.
A cat looking at a donut on a plate.
A cat is looking over a chocolate covered donut on a plate.
A cat stares at a chocolate topped donut, with the caption reading, "DONUT WANT."
Cat looks at donut with words "donut want" along bottom
```

Figure 3: Sample Image Captioning Output

CNN image encoder We implemented the pretrained model ResNet50 from PyTorch framework to extract the global image representation that initializes the RNN state. In detail, the structures of ResNet50 end with several layers of convolutions and fully connected layers. We extracted feature presentative of the image from last convolution layer and use linear mapping to map the features to the same shape of embedding features of RNN part. Since the images available for training are of varying dimensions. Before feeding the image to CNN model, we re-scaled the image to the dimension of 224*224 in RGB channel.

Captioning vocabulary We totally used 6325 unique words from the training captions that appear at least 10 times. Words that appear less frequently are replaced by a special OUT-OF-VOCABULARY token and the start and the end of the caption is marked with a special START and STOP token.

RNN decoder We constructed our RNN decoder based on LSTM cell-model. From CNN encoder, we obtained feature presentative of image which is used as the first input to LSTM. With hidden layers and final linear mapping, we obtain the score vector of each word. Adding the soft-max layer, we can get the probability vector of each word which is used to construct the loss function. In our model we used cross-entropy loss.

Training The neural network is trained using the mini-patch stochastic gradient descent (SGD) method. The base learning rate is 0.001. The learning rate drops 50% in every 20,000 iterations. Model is selected based on total loss value on the validation set. We implemented our model with PyTorch framework and trained it on GTX TITAN X 12GB.

Inference and Web structure Each image is fed to the pre-trained model. RNN decoder will generate captions iteratively until it meets STOP token. At one time RNN generates word embedding probability vector which is continually fed to RNN to generate next word. The probability vectors are collected together and mapped to obtain the complete caption. To present our results more efficiently, we build a web application in python using framework called Flask. To use the web application, just simply choice an image file and then press the “Upload” button to send it to server as shown in Figure 4. On server, this image will be treated as input of CNN image encoder to synthesis a sentence that describe the image. Then sentence will be brought to IBM Text2Speech API to generate audio file, which will show up below the image.

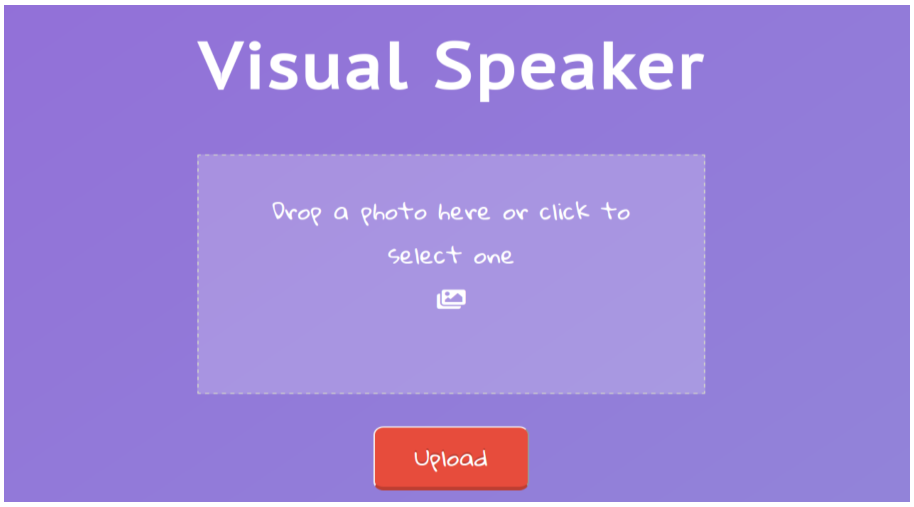


Figure 4: Homepage for the Model

4. RESULTS AND DISCUSSION

During the training, we also do validation on the validation set to choose the best model according to validation loss. The training loss and the validation loss is show in figure below:

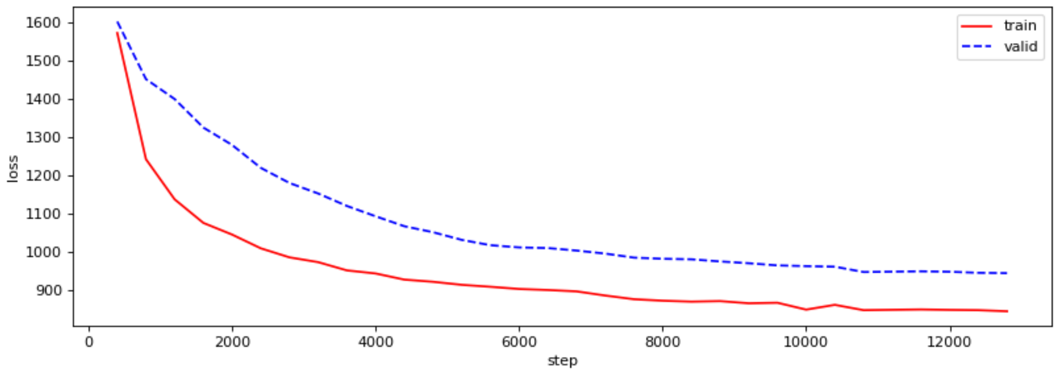
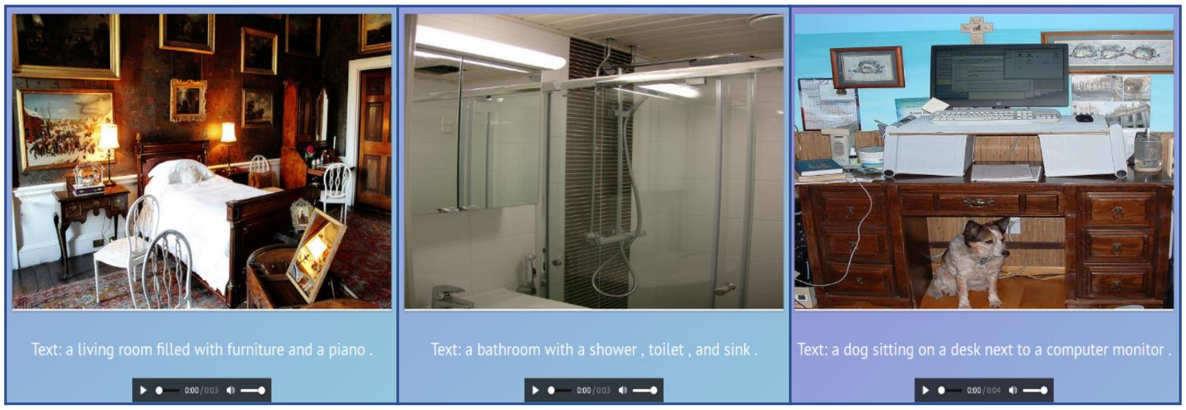


Figure 5: Training and Validation Losses

Using test dataset, we use the most popular metrics BLEU4 to quantitatively evaluate our performance. The higher this value is, the closer the generated caption to the target caption. In total, we get the value of 30, which is better than the result of 27 from the state-of-art algorithm presented in [5]. Since our application serves for the blind, following are our results based on different environment.

1) House Infrastructure

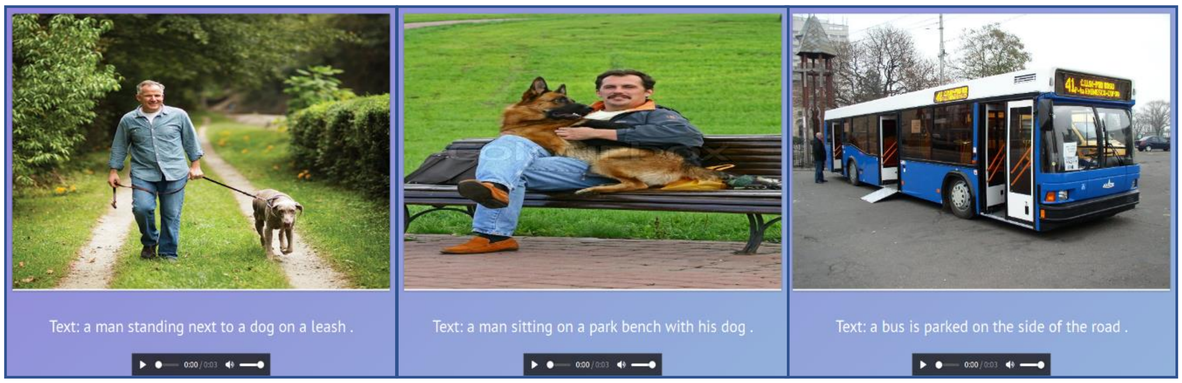


Text: a living room filled with furniture and a piano .

Text: a bathroom with a shower , toilet , and sink .

Text: a dog sitting on a desk next to a computer monitor .

2) Park

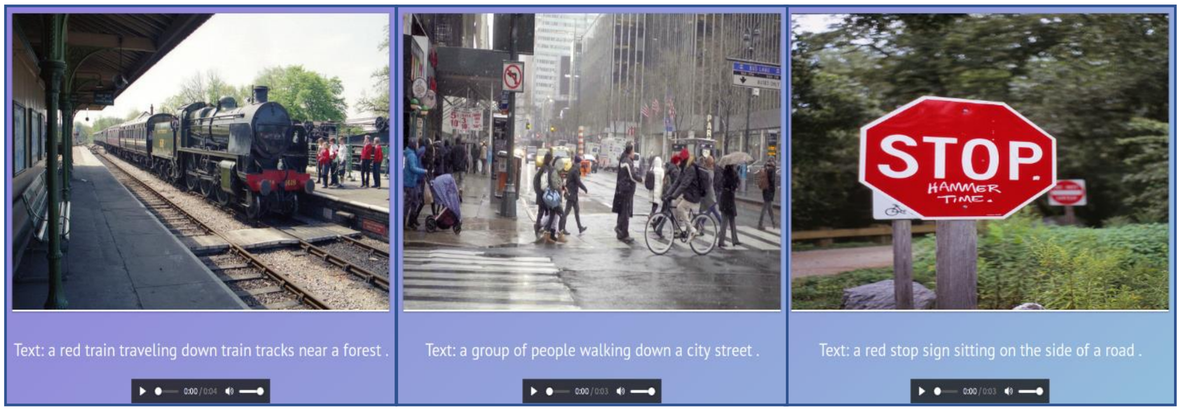


Text: a man standing next to a dog on a leash .

Text: a man sitting on a park bench with his dog .

Text: a bus is parked on the side of the road .

3) Traffic

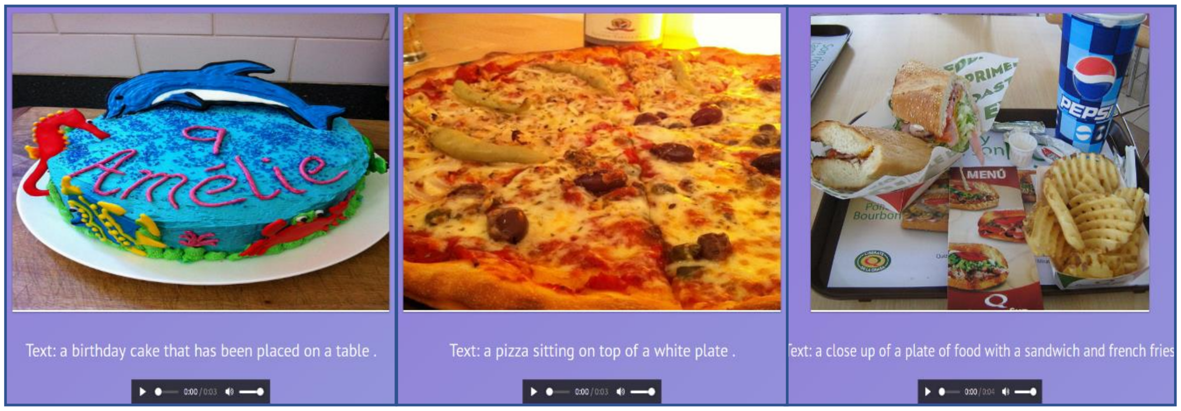


Text: a red train traveling down train tracks near a forest .

Text: a group of people walking down a city street .

Text: a red stop sign sitting on the side of a road .

4) Food

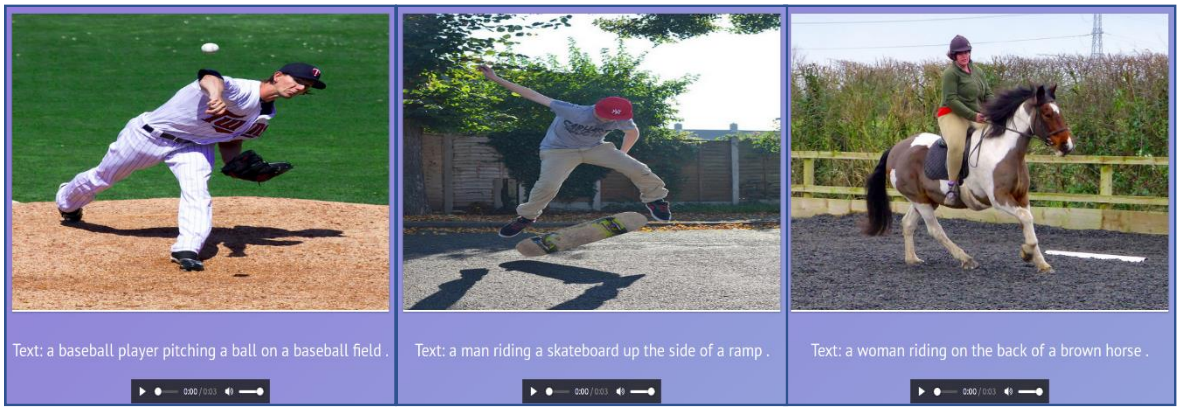


Text: a birthday cake that has been placed on a table .

Text: a pizza sitting on top of a white plate .

ext: a close up of a plate of food with a sandwich and french fries

5) Sports

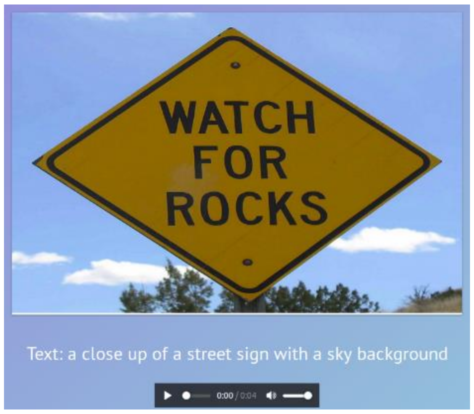


Text: a baseball player pitching a ball on a baseball field .

Text: a man riding a skateboard up the side of a ramp .

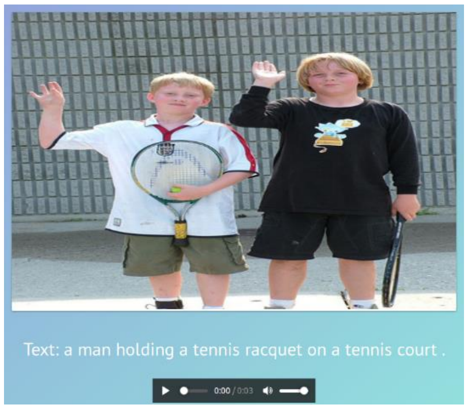
Text: a woman riding on the back of a brown horse .

In general, our model could successfully describe the image. However, there are still some images which cannot be correctly interpreted. We also give some of the examples below.



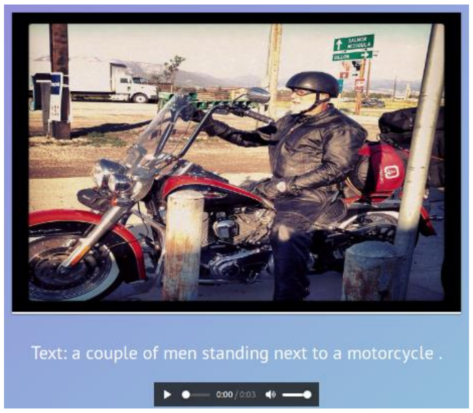
Text: a close up of a street sign with a sky background

A



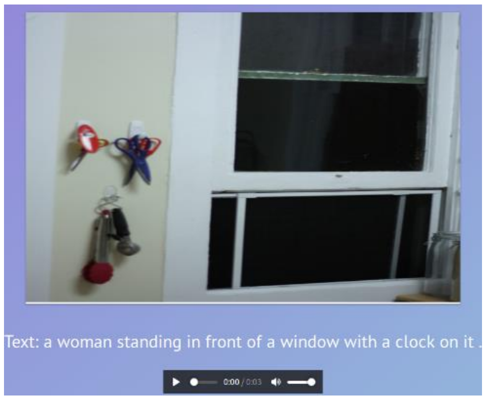
Text: a man holding a tennis racquet on a tennis court .

B



Text: a couple of men standing next to a motorcycle .

C



Text: a woman standing in front of a window with a clock on it .

D

Case A: Though the model can tell it's a traffic sign it fails to distinguish the words in the sign, which is the important information. The reason being that our model was trained on more general images rather than only focusing on a specific field. Also, the captions in the training set fail to provide enough information about the traffic signs' meanings.

Case B: Here, the model successfully describes the tennis racquet but incorrectly captures only one person.

Case C: For the third one which depicts a man driving an automobile, it incorrectly identifies as "a couple of" men. The reason is since most of the captions generated from training include terms such as "one", "several", "a couple of" etc.

Case D: For the fourth one which depicts a window and a scissor, the model identifies a woman. This shows the bias of the model.

5. CONCLUSION

We have built and modified a classical image captioning method by fusing it with the IBM Text to Speech tool. The final application is a fully-functional website that generates a seamless speech about the image, which can be further developed to help the blind to understand images. To understand the model completely, we decomposed the model to CNN, RNN, sentence and speech generation. The modified method is trained and evaluated on the COCO caption corpus. We have also developed a simple but attractive graphical user interface using Flask server, which allows the user to upload an image, transform the image into a text and then convert the text into a speech. The API used for the text-to-speech synthesizer supports voices in languages like Brazilian Portuguese, English (UK and USA), French, German, Italian, Japanese and Spanish.

In future, we plan to make efforts to create engines for localized Nigerian language to make text to speech technology more accessible to a wider range of Nigerians. Similar models already exist in some native languages e.g. Swahili, Konkani and Telugu language. Another area of further work in on the implementation of this model on other platforms, such as telephony systems, ATM machines, video games and any other platforms where image and speech analytics would be an added advantage and increase its current functionality.

REFERENCES

[1] "Visual Impairment and Blindness." *World Health Organization*. (2014). Web. 10 Apr. 2016.

[2] Hochreiter, Sepp and Jürgen Schmidhuber. "Long Short-Term Memory." Neural Computation 9 (1997): 1735-1780.

[3] Vinyals, Oriol et al. "Show and tell: A neural image caption generator." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): 3156-3164.

[4] IBM Text to Speech Service: https://www.ibm.com/watson/developercloud/text-to-speech/api/v1/curl.html?curl#introduction

[5] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick.Microsoft COCO: common objects in context. In ECCV,2014.