JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

AUTOMATED IMAGE CAPTION GENERATOR FOR VISUALLY IMPAIRED

[1]Yashwanth G, [2]Karthik R, [3]Yashaswini N L, [4]Yakshitha K C, [5]Anusha J, [6]Dr.Revathi V
[1]Student, [2]Student, [3]Student, [4]Student, [5]Student, [6]Associate Professor
[1]Computer Science Engineering,
[1]Dayananda Sagar University, Bangalore, India.

Abstract: Image processing is used in various industries and is one of the most advanced technologies used in Google, the medical field etc. google vision is one of the most used APIs in our daily lives for image labelling, face and landmark detection, optical character recognition (OCR), and tagging of explicit content, etc. The widely used google photos and iCloud use image processing to recognize the scenario of the photo and facial recognition. Image processing can help the visually impaired describe their surroundings. A voice-based image caption generation which is built using encoder-decoder architecture is used to describe the image.
Images of the surroundings are used to generate captions which will be read aloud to the visually impaired so that they'll get a stronger sense of what's happening around them. As an encoder, we have a pre-trained Resnet50 model where ResNet-50 is a convolutional neural network which is 50 layers deep. This network learns rich feature representations for a large range of images. Besides extracting high-level features from images using Resnet50, we also maintain the image colour composition using OpenCV techniques which also helps the model to extract the features from small components within the image. As a decoder, we have an LSTM network. Long remembering (LSTM) is a recurrent neural network (RNN) architecture and contains the Time Distributed Layer that's, it can process not only single image data points but also entire sequences of image data.

Index Terms – Deep learning, Resnet50, LSTM, Flutter, Encoder-Decoder, Flicker8k

I. INTRODUCTION

Most people have some sort of visual problem in their lives where some cannot see objects completely while others have problems reading small print. These kinds of conditions are often easily treated with eyeglasses or contact lenses. But when one or more parts of the human brain that are required to process images become diseased or damaged, severe or total loss of vision can occur which cannot be treated.
Visual impairment being a major detriment to the quality of individual life, it is worth exploring options available to improve the quality of the individual visually impaired person's life. Almost everything we witness today along with artificial intelligence is thanks to deep learning. The deep learning models work by deploying statistics to find patterns in data and also have proved immensely powerful in mimicking human skills such as the ability to hear and see, and recognise the interpretation of the images Deep learning-based image recognition has become "superhuman" and produces more accurate results than human contestants. This progress that has been made in the field of image recognition is the increasing application of deep learning techniques to various visual art tasks and social causes.
In this paper, we propose using encoder-decoder models where the encoder is Resnet50 which is a Convolutional Neural Network that is 50 layers deep and has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer and the decoder is LSTM (Long-short term memory) which is a Recurrent Neural Network which is capable of learning long-term dependencies in data, these models allow for a process in which a machine learning model generates a sentence describing an image. It receives the images captured by individuals as the input and outputs a sequence of words. and Natural Language Processing (NLP), which allows machines to break down and interpret human language using these we develop an application to aid the visually impaired like normal people's even visually impaired are exposed to various objects, articles, schematics in documents, and advertisements, living and non-living things. These things in front of the visually impaired must be interpreted by them to know what's happening around them.
Our proposed model with the Talkback feature, the Google screen reader included on Android devices that give eyes-free control of the Android device to the visually impaired person which helps them to capture the pictures. With the help of deep learning models and NLP technology, individuals get the description of the pictures captured by visually impaired persons.

II. RELATED WORKS

According to the research [1], the Inception architecture's complexity makes it more difficult to make network adjustments, and if the design is scaled up too quickly, huge portions of the computational advantages might be lost. Inception is compared to other models in this research, and it produces less error and has a low computational cost. The conclusion reached from this article is that Inception is a complicated network in which making modifications is challenging.

Many of the difficulties they encountered when training the models were due to overfitting, according to this research [2]. True, purely supervised algorithms need a big quantity of data, but high-quality datasets contain less than 100,000 photos. The process of providing a description is far more difficult than classifying an object. Different datasets were examined, and it was discovered that as the size of the available datasets for picture description grows larger, so does the performance, but giving descriptions to each image gets more challenging. The Flickr8k dataset is more efficient, according to the findings.

This model [3] is provided with LRCN, a family of models that is both spatially and temporally deep, as well as versatile enough to be used for a wide range of visual problems requiring sequential inputs and outputs. LSTM outperforms STM, according to the results.

The model was presented as a family of mobile-first computer vision models for Tensor flow, according to the research [4], and was developed to successfully optimise accuracy while being conscious of the limited resources for on-device or embedded applications. Mobile Nets are low-latency, low-power models that have been parameterized to satisfy the resource restrictions of various use cases. With a percentage of 70.6 per cent, Mobile Net outperforms ImageNet when compared to other models.

They employed the ResNet50 network as an encoder to transform the pictures into a vector representation, and the LSTM network as a decoder to decode the vector and create captions in this article [5]. Soft Attention is a method that allows the model to focus on a specific region of the image to better anticipate phrases. The captions are most likely based on the greatest probability. As a consequence, appropriate captions are created automatically with an image classification accuracy of 71%.

In this [6] presented model, the effects of changing the image feature like edges, height, width while using the encoder model for feature extraction on the image classification task by analysing the BLEU score at both the sentence and corpus levels are observed.

According to the study [7], LSTM is used to increase the precision of image captioning by mapping variable sequences of words in NLP to a distributed vector using a decoder. The neural model creates a description for the picture in this method. With CNN as the image encoder, the RNN decoder uses the categorised image to create captions using the hidden layer.

CNN splits the image into numerous objects, activities, and qualities in [8] and then creates a caption for each image afterwards. The produced description comprises the items in a picture, but it also represents the relationship between individual objects, their properties, and the activities in which they are participating.

The authors of the research [9] propose a model called Headcap that uses IoT and deep learning techniques and contains hierarchical features capable of collecting varied groups of semantics while also easing attention defocus. The Raspberry Pi is linked to a camera, speaker, and microphone, and has access to a deep learning model that predicts captions for the picture.

In this paper [10], the proposed hybrid image caption generator uses R-CNN and RODe for object detection to provide descriptive output. The model is based on CNN-RNN architecture.

III. ENCODER-DECODER ARCHITECTURE

3.1 Encoder

The encoder is used to encode the data to obtain insights. An encoder is used to extract high-level properties from images. In this situation, ResNet50 is used as the encoder. ResNet50, also known as Residual Networks, is a widely used neural network in computer vision. Using ResNet, we were able to effectively train 150-layer deep neural networks. The problem of vanishing gradients made training very deep neural networks difficult before ResNet. ResNet is a computer vision backbone model that is utilised in a range of applications. ResNet uses skip connections to fix the problem of disappearing gradients.
Each step of the ResNet50 model has its convolution and identity block. There are three convolution layers in each convolution block, and three convolution layers in each identity block. There are around 23 million trainable parameters in the ResNet-50. A minor change was made for ResNet50 and higher: shortcut connections previously skipped two layers; now, they skip three levels, and 1 x 1 convolution layers have been introduced.

3.2 Decoder

Long-term memory networks are a type of Recurrent Neural Network that can learn long-term dependencies. Long-term memory networks are also called LSTMs. They're currently commonly utilized and function admirably in a variety of situations. LSTMs were created to overcome the problem of long-term reliance. They don't have to exert much effort to recall information for long periods; it comes naturally to them. Recurrent neural networks are made up of a series of modules that repeat. In classic RNNs, this repeating module will have a basic structure, such as a single tanh layer.
The cell state, which is represented by the horizontal line running along the top of the image, is the key to LSTMs. The cell is in a condition similar to that of a conveyor belt. With only a few modest linear exchanges, it flows along the whole chain. It's really

simple for data to pass through it unaltered. The LSTM's capacity to remove or add information to the cell state is closely controlled by the structure.

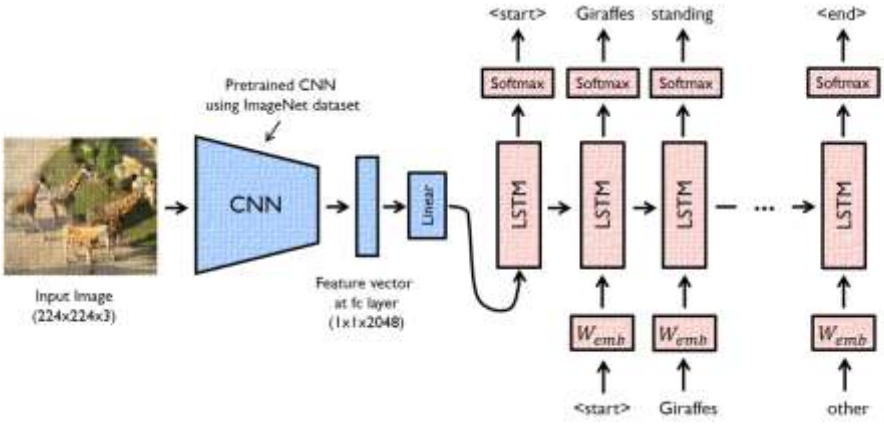


Fig. 1 Encoder-Decoder model

IV. METHODOLOGY

4.1 Image and Caption Dataset Access and Reading:

We use the flicker 8k dataset which contains 8000 images provided by the University of Illinois at Urbana-Champaign. This dataset is available on the Kaggle website. Each image is provided with 5 captions which are later combined by the model to output a single sentence that describes the image the best.

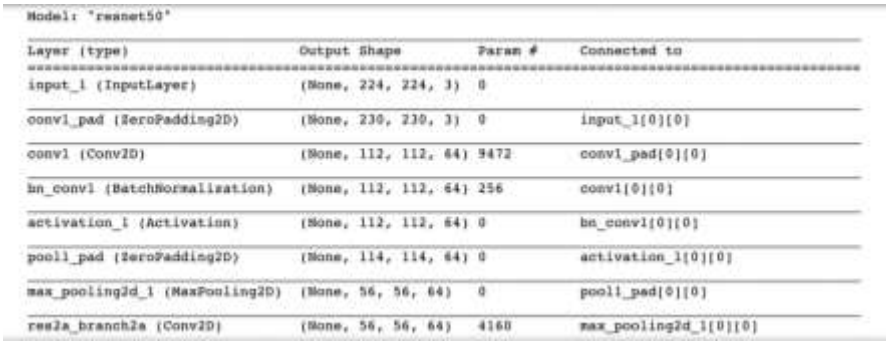
4.2 Image Data Preparation and Processing:

We need to convert the image into an encoding so that the machine can understand the patterns in it. InceptionV3, a model that is trained on large datasets is used to extract the features from images. Every image should be converted to a fixed-sized vector which can be fed as an input to the neural network. the last SoftMax layer is removed from the model and a 2048 length vector is extracted for every image.

Fig. 2 Image preparation

4.3 Creating a Resnet50 encoder model and extracting features from images:

An image from any source is given as input to the encoder-decoder model, which processes it into a pre-trained Resnet50 model and extracts the 2048-dimensional vector from the model. This contains high-level image features and then it embeds the vector into the dimensions of the sentence vector, which is initially a tag and whose two vectors are concatenated and passed over a Time Distributed Layer followed by LSTM.



Model: "resnet50"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 224, 224, 3)	0	
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_1[0][0]
conv1 (Conv2D)	(None, 112, 112, 64)	9472	conv1_pad[0][0]
bn_conv1 (BatchNormalization)	(None, 112, 112, 64)	256	conv1[0][0]
activation_1 (Activation)	(None, 112, 112, 64)	0	bn_conv1[0][0]
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	activation_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 64)	0	pool1_pad[0][0]
res2a_branch2a (Conv2D)	(None, 56, 56, 64)	4160	max_pooling2d_1[0][0]

Fig. 3 Encoder

4.4 Cleaning and conducting word embeddings on the text data:

The model must predict the captions based on the image. Hence, during the training period captions will be the target variables that the model is learning to predict. It then predicts the caption word by word. Therefore; each word should be encoded into a fixed-sized vector. Every unique word in the vocabulary is represented by an integer which is later used for pre-processing.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               262272
_________________________________________________________________
repeat_vector_1 (RepeatVecto (None, 40, 128)           0
=================================================================
Total params: 262,272
Trainable params: 262,272
Non-trainable params: 0
```

Fig. 4 Text pre-processing

4.5 LSTM Decoder Model development:

In long-short term memory, the output of each LSTM cell is transmitted as an input to the next LSTM cell. The final output is created by concatenating the outputs of all LSTM cells. The last output sentence provides a concise description of the input image. Finally, the text is converted to the audio format for the output.

```
Model: "functional_17"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
embedding_8_input (InputLayer)  [(None, 40)]         0
__________________________________________________________________________________________________
dense_22_input (InputLayer)     [(None, 2048)]       0
__________________________________________________________________________________________________
embedding_8 (Embedding)         (None, 40, 128)      1256512     embedding_8_input[0][0]
__________________________________________________________________________________________________
dense_22 (Dense)                (None, 128)          262272      dense_22_input[0][0]
__________________________________________________________________________________________________
lstm_24 (LSTM)                  (None, 40, 256)      394240      embedding_8[0][0]
__________________________________________________________________________________________________
repeat_vector_6 (RepeatVector)  (None, 40, 128)      0           dense_22[0][0]
__________________________________________________________________________________________________
time_distributed_8 (TimeDistrib (None, 40, 128)      32896       lstm_24[0][0]
__________________________________________________________________________________________________
concatenate_8 (Concatenate)     (None, 40, 256)      0           repeat_vector_6[0][0]
                                                                 time_distributed_8[0][0]
__________________________________________________________________________________________________
lstm_25 (LSTM)                  (None, 40, 128)      197120      concatenate_8[0][0]
__________________________________________________________________________________________________
lstm_26 (LSTM)                  (None, 512)          1312768     lstm_25[0][0]
__________________________________________________________________________________________________
dense_24 (Dense)                (None, 8254)         4234302     lstm_26[0][0]
__________________________________________________________________________________________________
activation_8 (Activation)       (None, 8254)         0           dense_24[0][0]
==================================================================================================
Total params: 7,490,110
Trainable params: 7,490,110
Non-trainable params: 0
```

Fig. 5 Encoder

4.6 Using Data Generators and Progressive loading to train the model:

The amount of memory to store data blocks in the data matrix consumes a lot of space in the main memory. Even if it is managed to store in the RAM, it slows down the computer. The ImageDataGenerator class provided by the Keras API is nothing but an implementation of the generator function in Python. This makes sure that the entire dataset need not be stored in the main memory at a time, reducing storage consumption.

```
Epoch 1/100
117/117 [==============================] - ETA: 0s - loss: 0.4010 - accuracy: 0.8673
Epoch 00001: accuracy improved from -inf to 0.86725, saving model to best_model_acc.h5
117/117 [==============================] - 54s 460ms/step - loss: 0.4010 - accuracy: 0.8673
Epoch 2/100
117/117 [==============================] - ETA: 0s - loss: 0.3812 - accuracy: 0.8718
Epoch 00002: accuracy improved from 0.86725 to 0.87184, saving model to best_model_acc.h5
117/117 [==============================] - 54s 457ms/step - loss: 0.3812 - accuracy: 0.8718
Epoch 3/100
117/117 [==============================] - ETA: 0s - loss: 0.3784 - accuracy: 0.8730
Epoch 00003: accuracy improved from 0.87184 to 0.87297, saving model to best_model_acc.h5
117/117 [==============================] - 53s 452ms/step - loss: 0.3784 - accuracy: 0.8730
Epoch 4/100
117/117 [==============================] - ETA: 0s - loss: 0.3768 - accuracy: 0.8732
Epoch 00004: accuracy improved from 0.87297 to 0.87319, saving model to best_model_acc.h5
117/117 [==============================] - 52s 445ms/step - loss: 0.3768 - accuracy: 0.8732
Epoch 5/100
117/117 [==============================] - ETA: 0s - loss: 0.3799 - accuracy: 0.8732
Epoch 00005: accuracy did not improve from 0.87319
```

Fig. 6 Data Training

4.7 Use the test dataset to evaluate the trained model:

The Flicker 8k dataset is divided into train and test datasets to check how efficient is its prediction for unseen data i.e., test data. The model is evaluated with the test dataset to check the prediction accuracy. The dataset is split into a 75:25 ratio where 6000 images are used for training and the rest 2000 images are used for testing.

```
Argmax_Search = predict_captions(test_img)
z = Image(filename=im)
display(z)

print(Argmax_Search)
```

Two dogs wrestle in the grass .

Fig. 7 Generated caption during testing

4.8 Text to speech conversion:

After the caption is predicted for the image, it is converted into audio using the flutter_tts plugin.

4.9 Developing a user-friendly interface:

An application using flutter is developed where the user can get a description of the image in the form of speech.

V. RESULTS AND DISCUSSION

Fig. 8 Generated caption

VI. FUTURE SCOPE

The model's accuracy can be improved even further by training on a larger dataset. It can be built into a smart glass or a portable gadget that is easy to use for visually impaired people. With facial recognition and optical character recognition, this model may be updated to provide a caption for a live video feed. Depending on the user's preferences, it can create captions in a variety of languages.

VII. CONCLUSION

The suggested model is resilient, requires little processing resources, and requires no specific training since its accuracy on training data is over 92.4 per cent with a loss of 0.32, and its validation accuracy is 90 per cent with a loss of 0.12. This architecture incorporates previous pre-trained model advances as well as a variety of deep learning and natural language methodologies. The scalability of big applications with this paradigm is significant since the whole system is streamlined. Because we employed Time distributed layers followed by LSTM cells, the sentence's sequence information will be saved, and because video frames are time-based, these time-distributed layers will perform very well on such data for creating captions for movies. The generated captions while validating the model with test data are accurate and have well-defined semantic meaning.

VIII. ACKNOWLEDGEMENT

We are grateful to our respectable teacher Dr Revathi whose insightful leadership and knowledge benefitted us to steer our project successfully.

IX. REFERENCES

[1] Christian Szegedy Google Inc & Vincent Vanhoucke & Sergey Ioffe & Jonathon Shlens & Zbigniew Wojna. (2015). Rethinking the Inception Architecture for Computer Vision.

[2] Oriol Vinyals Google & Alexander Toshev Google & Samy Bengio Google & Dumitru Erhan Google. (2015). A Neural Image Caption Generator.

[3] Jeff Donahue & Lisa Anne Hendricks & Marcus Rohrbach & Subhashini Venugopalan & Sergio Guadarrama & Kate Saenko & Trevor Darrel. (2015). Long-term recurrent convolutional networks for visual recognition and Description.

[4] Andrew G. Howard & Menglong Zhu & Bo Chen, Dmitry Kalenichenko & Weijun Wang & Tobias Weyand & Marco Andreetto & Hartwig Adam. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.

[5] Yan Chu & Xiao Yue & Lei Yu & Mikhailov Sergei & Zhengkui Wang. (2020). Automatic Image Captioning Based on ResNet50 and LSTM with Soft Attention.

[6] TSS Chandana & Harsh Maru & Dinesh Naik. (2021). Comparison of Image Encoder Architectures for Image Captioning. IEEE Xplore Part Number: CFP21K25-ART.

[7] Chetan Amritkar & Vaishali Jabade. (2018). Image Caption Generation using Deep Learning Technique. Fourth International Conference on Computing Communication Control and Automation (ICCUBEA).

[8] Mrs T.Kranthi & B.Sai Kumar & A.Srikar & A.Ephraim & D.Ganesh Teja. (2019). Identifying Object In An Image And Generating Caption For Given Image, JETIR, Volume 6, Issue 4.

[9] Long Lan & Wenjing Yang & Shiwei Wang. (2021). HeadCap: A Hierarchical Encoder-And-Decoder Image Captioning Model.

[10] N.Komal Kumar & D.Vigneswari & A.Mohan & K.Laxman & J.Yuvaraj. (2019). Detection and Recognition of Objects in Image Caption Generator System: A Deep Learning Approach.