AJNA - Voice Assisted Captioning Tool for the Blind

Ajay Bhat, K. Kaladharshini, Karthika Menon, Chitra Babu

AJNA - Voice Assisted Captioning Tool for the Blind

Ajay Bhat, K. Kaladharshini(✉), Karthika Menon, and Chitra Babu

CSE Department, SSN College of Engineering, Chennai 603110, India
{ajay18012,kaladharshini18067,karthika18070}@cse.ssn.edu.in,
chitra@ssn.edu.in

Abstract. In India, there are currently over 18 million blind or visually impaired persons which makes up about 20% of the world's blind population. The inability to read text or identify objects with ease has impaired their quality of living greatly. Automatic caption generation that describes visual content of images captured in real time has garnered significant attention over the years. Several approaches have been proposed to achieve this task. However, they were not well-suited for mobile devices due to the fact that they were resource-intensive. This paper proposes to employ an attention based LSTM model that performs the task of captioning images that are part of the VizWiz dataset. The generated caption describes the scene captured by the image. The model generates text captions in English and these are converted to voice modality to facilitate the usage by blind people.

Keywords: Image Captioning · Encoder-Decoder · Android Application

1 Introduction

Image captioning [1] is the process of generating text from input images - based on the objects and actions present in it. It is a research problem that needs both Computer Vision (CV) and Natural Language Processing (NLP) to generate the captions. The dataset for image captioning consists of both the input images and their corresponding captions. Image captioning in Deep Learning has three components - image feature encoder, sequence decoder and sentence generator. The Image Feature Encoder takes an image as input and extracts the features with the help of a Convolutional Neural Network (CNN) model. Subsequently, the Sequence Decoder takes the extracted features and outputs the sequence of tokens. The Sentence Generator in turn takes the sequence of tokens and outputs a sentence that describes the image. The tasks of token and sentence generation are handled by a Recurrent Neural Network (RNN) model. Image captioning finds several applications in day-to-day life. It can be applied in natural language processing tools, recommendation systems, assistive technologies to aid the visually impaired, social media and publishing houses.

Published by Springer Nature Switzerland AG 2023
S. Chandran K R et al. (Eds.): ICCIDS 2023, IFIP AICT 673, pp. 243–253, 2023.
https://doi.org/10.1007/978-3-031-38296-3_19

Attention based mechanisms direct special focus on underlying salient features to generate captions for images. In a typical attention based model, the image information obtained by a CNN is passed to a language model for the generation of words and phrases. At each step of the language generation phase, salient features of the image are focused upon based on the words generated. Captions are updated dynamically until the end state is reached. The Show, Attend and Tell [8] model applies two different attention mechanisms to achieve this-stochastic hard attention & deterministic soft attention. In this model, attention is implemented by including an additional attention gate into the Long Short Term Memory (LSTM) that helps in concentrating selective attention.

The primary contribution of this paper is of creating a light-weight Android application that performs the task of captioning for images that are captured in real-time and the option of hearing the caption out aloud.

The remainder of the paper is organized as follows: Sect. 2 discusses the work related to proposed system. Section 3 talks about the proposed model while Sect. 4 explains the implementation of this model. Section 5 talks about the results obtained and finally Sect. 6 consists of conclusion and potential directions for future work.

2 Related Work

To generate accurate captions, models need to focus their attention on prominent objects, their characteristics and how they interact with the surrounding elements in the image. This will help in generating captions that are both semantically and syntactically sound. Image Captioning [2] models aim to build accurate and meaningful descriptions of an image. Mullachery et al. [1] had proposed an LSTM model in conjunction with a CNN (ResNet or VGG) model. Mapping of images is via Vision CNN and words by using word embeddings. It was trained on all three datasets - Flickr8k, Flickr30k and MS COCO. It provided a BLEU [10] score of 0.077–0.275.

One of the most intriguing sides of the visualization system present in human beings is the attention to detail. Using attention, salient features which go otherwise unnoticed are better projected instead of compressing images into one dimensional representations. Xu et al. [8] offer 2 variants: “a hard stochastic attention mechanism” and “a soft deterministic attention mechanism”. The inference from this paper is that how attention can be incorporated and how it enhances the detail to visualization or what the model “sees”, which adds an overall advantage. This work reported advanced functionality on the Flickr8k dataset, Flickr30k dataset and MS COCO dataset. On the basis of the METEOR metric, major advancements in MS COCO’s state-of-the-art performance were also made.

Dognin et al. [15] had proposed the Multimodal Assistive Captioner. The captioner is a multimodal transformer with 4 distinct stages. Encoding stage provides embeddings for 3 modality streams - image features, object detection and OCR results. The dataset used was VizWiz dataset. The embedded output from the encoder is passed into the decoder and minimizing loss, OCR maximization is performed before the final caption is generated. This model achieved the highest CIDEr score of 81.04.

Nivedita et al. [4] had developed the system for live image captioning from a video source. For encoding, a CNN model is used. The CNN will automatically extract the image's features, and through transfer learning, the features are supplied to LSTM for the purpose of decoding. The proposed usage of the beam search algorithm by the LSTM resulted in a significant improvement in the qualify of caption generation.

Mathur et al. [5] had proposed the Encoder-Decoder architecture. In this work, the Encoder is a pre-trained Inception V4 CNN model and the Decoder is a deep RCNN implemented with LSTM cells. It provides a 6x speedup in image pre-processing state and has good performance in real world applications.

Smartphone-based image captioning has been implemented in a few research works, however all of them use previously captured images instead of real-time images. Makav et al. [3] had proposed a "pointer-generator mechanism" which copies the text detected from the source whenever required. Here, the faster-RCNN model is pre-trained on the ImageNet dataset. The OCR token embeddings are retrieved using a "pre-trained base, uncased BERT mode" [13]. This was implemented as an app in smartphones.

3 Proposed Model

The proposed system shown in Fig. 1 details the training phase of an attention based encoder-decoder model that generates goal oriented captions.

The testing phase of the system is shown in Fig. 2 and the android component in Fig. 3.

CNN based approaches use the penultimate fully connected layer of the CNN to extract information about salient features in the image. However, this may lead to loss of useful information and will only generate high-level abstractions and not the low-level object specific details. Attention based mechanisms have been proven to address these limitations. These models change their attention to relevant parts of the image while generating each word.

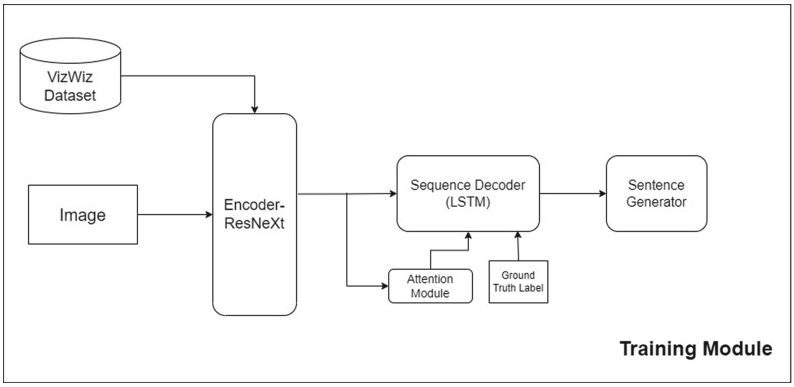

Fig. 1. Training Phase of the Proposed System.

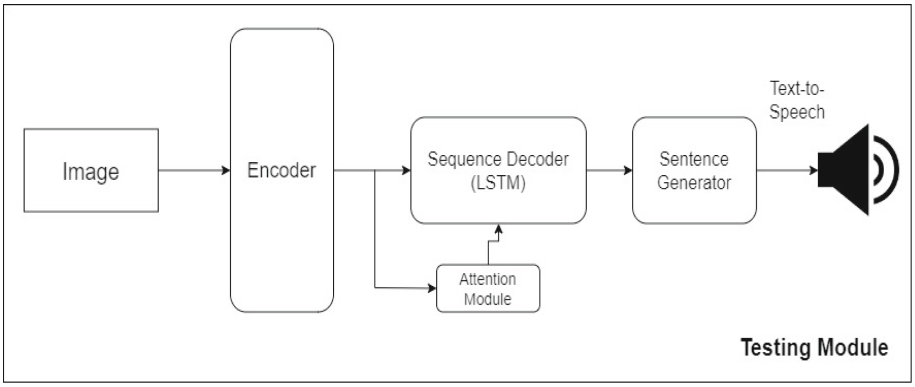

Fig. 2. Testing Phase of the Proposed System

The input image is passed to a pre-trained ResNeXt-101-32x8d CNN model to extract features from its lower convolutional layers and generate a feature vector of the low-level abstractions. ResNeXt-101-32x8d was chosen since it was trained on billions of Instagram images i.e, images captured on phones by users. This works well with the given use case since images are captured in real time. The CNN includes an input layer, a number of convolutional layers, pooling layers, batch normalization layers, dropout layers and finally an output layer. The feature abstractions are extracted in a compact encoded representation to be passed to an LSTM sequence decoder.

The encoded output from the encoder, along with the hidden state of the previous timestep are passed to the attention module. An attention module is attached to the decoder to obtain the aggregate attention weights from the encoded output. This helps in focusing attention on relevant parts of the image. It outputs a weighted average which is concatenated with the word that was previously generated and passed as input into the decoder. During training, the

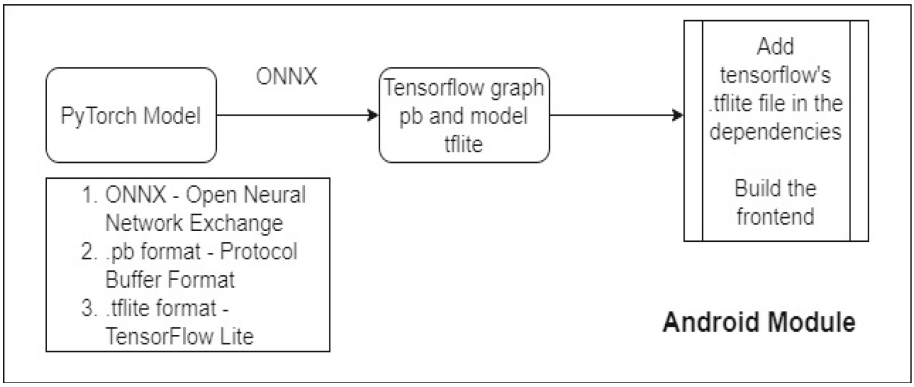

Fig. 3. Android Module of the Proposed System

decoder returns the next output sequence and the hidden state which is again passed back for the next timestep. "Teacher forcing [9] technique is used to determine the next input to the decoder, wherein the target word (ground truth label) is the next input to the decoder instead of the previous output sequence". For evaluation, the same training loop is employed without teacher forcing. The attention weights are stored for every time step. This continues until the model predicts the 'end' token and prediction stops.

The generated sentence is then tested and assessed against the ground truth label using the "BLEU score" metric, which evaluates the performance of the model. It is a quality metric score that measures the level of correlation between a machine translated output and human translation.

The trained Pytorch Model is converted to Tensorflow graph .pb format using Open Neural Network Exchange (ONNX). The .pb file is converted to .tflite with the help of tensorflowlite, and added to the Android Studio project dependencies to develop the backend. The frontend is built and integrated with the backend to make the complete android app. The corresponding caption for the input image is generated and translated to voice.

4 Implementation

4.1 Dataset

In the proposed system, the VizWiz dataset has been used. It is a dataset comprising images taken by the blind. It consists of 39,181 images that are each paired with 5 captions. Of the 39, 181 images; there are 23, 431 training images with 1,17,155 corresponding training captions, forming 60% of the dataset and the remaining form the validation and testing sets, 20% each. In particular, there are 7,750 validation images with 38,750 captions and 8,000 testing images with 40,000 captions.

4.2 ResNeXt

ResNet conventionally requires a large number of parameters which is significantly reduced by ResNext. This reduction of parameters is obtained by the use of a supplementary parameter called cardinality, which is an addition to the dimensions of ResNet. Cardinality is a term used to define the number of transformations in the set.

ResNext's architecture is described mainly by the use of 2 rules [11]: "First, if the blocks produce same-dimensional spatial maps, they share the same set of hyperparameters", and "if at all the spatial map is downsampled by a factor of 2, the width of the block is multiplied by a factor of 2".

Probability of error in ResNext-50 and ResNext-101 are inversely proportional to the cardinality, i.e. higher cardinality is preferable. The model implemented in the present paper is a 101 layer model (Fig. 4).

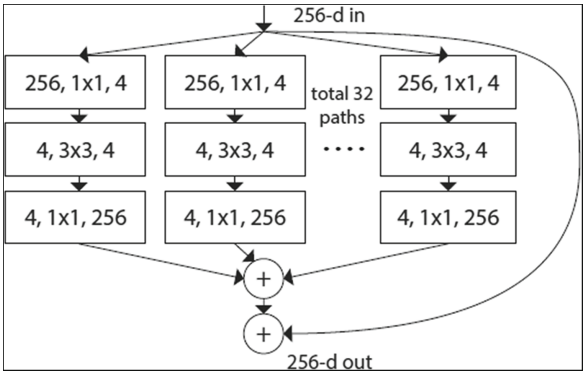

Fig. 4. ResNeXt block structure

4.3 Self-attention Based LSTM

Long Short Term Memory (LSTM) [14] is a recurrent neural network (RNN) architecture used in sequence prediction. The Sequence Decoder in the proposed system consists of several LSTM layers that decode the encoded image and predict a sequence of words. In addition, a self-attention module is implemented that helps the decoder focus on the most relevant parts of the image for generating the next word. Self-attention module allows the inputs to interact with each other and determine which input should be given greater importance. At every timestep, the Attention module takes the output from the previous state of the Decoder and encoded image as input. This generates a "weighted representation" of the image that is appended with the word that was generated before and fed into the Decoder. The next output sequence and hidden state for this timestep are subsequently generated by the Decoder. This iteration is repeated until the entire output sequence is generated.

4.4 Early Stopping with BLEU

The model performance is evaluated using Bilingual Evaluation Understudy metric. One of the most widely used automated and affordable measures, BLEU has a strong association with human judgments of quality. It has two ingredients: a numerical metric of "translation closeness" and a corpus of good quality human reference translations. The BLEU score is a numeric metric between zero and one that measures the similarity index of the machine generated text and a set of high-quality reference text [12]. This value represents the similarity between newly generated text called "candidate text" and how near in similarity it is to the "reference text". The value 1 indicates very good similarity. For each generated caption, the reference captions are the captions that are present for an image. The Show, Attend, and Tell [8] paper's authors note that "correlation between the loss and the BLEU score breaks down after a point and hence it is recommended to stop training early on when the BLEU score begins to degrade, even if the loss continues to decrease".

4.5 Creation of Android Application

ONNX (Open Neural Network Exchange Format) is a format designed to represent any type of Machine Learning and Deep Learning model. To integrate the model into the Android application, it was required to convert the model from pytorch environment to tensorflow environment. This is where ONNX was used as the pytorch environment .pth file was converted into .onnx file as intermediate representation. The .onnx file was then converted into the tensorflow environment model .pb file, which was then converted into .tflite file, which is the model to be used for integration with the Android application. The Google's tensorflow lite demo application was used as a reference to create the application for image captioning using tensorflow lite files that were created.

5 Results and Discussion

The hyperparameters that best suit the model were decided upon by continuous experimentation. They are shown in Table 1.

On training for 5 epochs with an encoder learning rate of 0.0001, decoder learning rate of 0.0004 and decay rate of 0.8, a training accuracy of 75.34% was obtained along with a BLEU score of 0.2203. For 10 epochs, the training accuracy and BLEU score incremented to 79.66% and 0.2734 respectively. The model was trained for a total of 20 epochs when it finally converged to a training accuracy of 83.98% and BLEU score of 0.3014. This is summarized in Table 2.

Though each image in the training dataset has five captions, when we have trained the model, we have created random batches of 64 in size where the batch loads an image with one of its five associated captions as a single entity. So, on testing, it generates only one caption per image by using the associated caption as its ground truth label.

Table 1. Hyperparameters used for training.

Hyperparameter	Value
Optimizer	Adam
Loss Function	Cross Entropy Loss
Encoder Learning Rate	0.0001
Decoder Learning Rate	0.0004
Batch Size	64
No. of Epochs	20
Steps per Epoch	9153

Table 2. Model Performance.

Epochs	Training Accuracy	BLEU Score
5	73.54	0.2203
10	79.66	0.2734
20	83.98	0.3014

A BLEU Score of 0.3–0.4 signifies understandable to good quality translations. The model performs well on newly introduced images captured in real time, and is able to generate coherently phrased sentences as captions.

The model was deployed into an android application, with the help of tensorflowlite, that proved to be extremely lightweight with minimal startup time. The app memory specifications are detailed in Table 3.

Table 3. App Memory Specifications.

Component	Memory
Storage	275 MB
RAM	200 MB

In comparison to the Dognin [15] model which incorporated additional components for text detection and object recognition in their encoder, the model presented in this paper is better suited for android applications being less resource intensive. However, the captions produced by the Dognin model had a better correlation to human levels of judgement, vastly due to the OCR functionality. When comparing their model's performance without the two additional modules (OCR & Object Detection), our model's BLEU scores proved to be higher. Table 4 details the 'IMG only (minus OCR & OBJ)' BLEU score comparison between the two models.

Table 4. Model Performance Comparison.

Model	BLEU Score
Our model	0.3014
Dognin model	0.2597

Here are a few sample captions generated by the model (Figs. 5, 6 and 7).

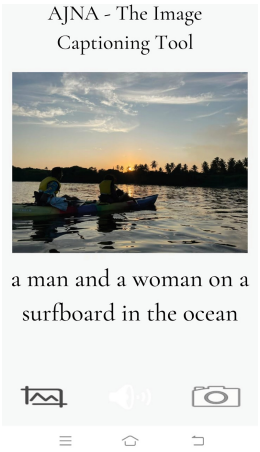

Fig. 5. Sample Image 1

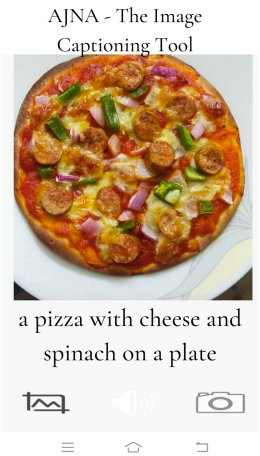

Fig. 6. Sample Image 2

Fig. 7. Sample Image 3

6 Conclusion and Future Work

An android application for image captioning was developed using an Encoder-Decoder based CNN-LSTM model. The proposed model is well suited for android applications, and could hence be integrated into a lightweight app. It produced significantly higher accuracy than existing smartphone-based systems, however it does not out-perform other state-of-the-art Encoder-Decoder models. This is largely because the model was optimized for app deployment and therefore does not utilize memory-intensive components.

Based on our research, it was inferred that goal-oriented image captioning is sparsely implemented in existing models. Scene Text Recognition [7] using CRAFT [6] proved to be more advantageous in recognising characters in real-world images than other systems like pytesseract and EAST as it recognized more objects in the input image. The GPU specifications were GeForce 2060 and 16 GB RAM, with which the complete dataset was trained by running just the CNN model as encoder but it proved insufficient to train all 23000 images in the VizWiz dataset using CRAFT. Future work can aim to incorporate Scene Text Recognition into the encoder with access to advanced resources. Incorporating regional languages in the app is an additional functionality that could be included.

References

1. Vikram, M., Motwani, V.: Image captioning. arXiv abs/1805.09137 (2018)
2. Ahsan, H., Bhalla, N., Bhatt, D., Shah, K.: Multi-modal image captioning for the visually impaired. arXiv abs/2105.08106 (2021)

3. Makav, B., Kılıç, V.: Smartphone-based image captioning for visually and hearing impaired. In: 11th International Conference on Electrical and Electronics Engineering (ELECO), pp. 950–953 (2019). https://doi.org/10.23919/ELECO47770.2019.8990395
4. Nivedita, M., Asnath Victy Phamila, Y., Harsh, P.V.: Captioning for motion detection for video surveillance applications using deep learning. Int. J. Innov. Technol. Exploring Eng. (IJITEE) **8**(8) (2019). ISSN 2278-3075
5. Mathur, P., Gill, A., Yadav, A., Mishra, A., Bansode, N.K.: Camera2Caption: a real-time image caption generator. In: International Conference on Computational Intelligence in Data Science (ICCIDS), pp. 1–6 (2017). https://doi.org/10.1109/ICCIDS.2017.8272660
6. Baek, Y., Lee, B., Han, D., Yun, S., Lee, H.: Character region awareness for text detection. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 9365–9374 (2019). arXiv:1904.01941
7. Baek, J., et al.: What is wrong with scene text recognition model comparisons? Dataset and model analysis. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 4714–4722. arXiv:1904.01906
8. Xu, K., et al.: Show, attend and tell: neural image caption generation with visual attention. In: Proceedings of the 32nd International Conference on Machine Learning, pp. 2048–2057 (2015)
9. Goyal, P., Pandey, S., Jain, K.: Deep Learning for Natural Language Processing - Creating Neural Networks with Python, pp. 155–156. Apress (2018). ISBN 978-1-4842-3685-7
10. Papineni, K., Roukos, S., Ward, T., Zhu, W.: BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, pp. 311–318 (2002)
11. Kurama, V.: A Review of Popular Deep Learning Architectures: DenseNet, ResNeXt, MnasNet, and ShuffleNet v2. https://blog.paperspace.com/popular-deep-learning-architectures-densenet-mnasnet-shufflenet/
12. "Evaluating models". Google Cloud. https://cloud.google.com/translate/automl/docs/evaluate
13. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. https://arxiv.org/abs/1810.04805
14. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
15. Dognin, P., et al.: Image captioning as an assistive technology: lessons learned from VizWiz 2020 challenge. J. Artif. Intell. Res. **73**, 437–459 (2022)