International Journal of All Research Education and Scientific Methods (IJARESM), ISSN: 2455-6211
Volume 9, Issue 12, December-2021, Impact Factor: 7.429, Available online at: www.ijaresm.com

# Image Captioning: An Assistive Technology

**Yashika Chugh[1], Er Narinder Kaur[2]**

[1,2]Information Technology, Maharaja Agrasen Institute of Technology Rohini, Delhi- India

----------**********----------

## ABSTRACT

**The internet has a range of content in the form of text, images, videos, etc. While some are highly informational, others might not be suitable for given age groups and people. In this paper, we propose an application of sensitive image classification based on the captions generated for an image. We start by comparing state-of-the-art image extraction networks (VGG-16, ResNet50, InceptionV3, and Xception) via training the generated image vectors on a benchmark model. We compare the models based on their attributes, training loss, and, most importantly, BLEU scores for the quality of generated captions. Further, we used the efficient Xception model (2.59 training loss and 0.525 BLEU-1 score) for generating captions whose sentiment was captured using VADER Sentiment Analyzer. In the end, we were able to classify images representing violence, negativity, and dangerous situations as negative.**

**Keywords: image captioning, feature extraction, sentiment analysis, BLEU score.**

## I. INTRODUCTION

The Internet is full of images and videos, which are nothing but multiple frames of images. These multimedia tools add a sense of understanding and bring the text to life. A picture is quite descriptive but are all of them suitable for everyone on the Internet? The internet users range from 8-year-old kids to 80-year-old senior citizens, and some images might convey what is not suitable for the gullible minds and the weak-hearted. *Image captioning* is a technology powered by deep convolutional networks and natural language processing which finds many uses on the net. In this research, we try to establish one such application of image captioning and thus refer to it as assistive technology.

Image captioning is readily used by social media applications like Facebook to describe the contents of the image as audio, used in security systems in combination with object detection, doctors can use it to identify the type of tumors (based on previously identified ones), and it can act as a great educational tool where children can learn to form sentences from all that they see around them. The primary purpose of this research is to develop an application capable of classifying images as harmful or sensitive based on the image's content using image captioning. Consider Instagram, for example; there is a sensitive post filter in the app which blurs those images which have something violent or sensitive. Now, Instagram can do so because some users reported that image, and the team verified that. To automate this process, we suggest using image captioning and judging the sensitivity of the image based on the sentiment of the caption generated.


**Fig. 1.Proposed Application**

We put forward this approach because training a classification model on some sensitive images would put all such visuals in a box, and the system would collapse if not re-trained frequently. However, as we suggest using captions that are defined words with defined sentiments, their meaning will not alter. We aimed to select the model capable of generating quality captions and thus, experimented with various state-of-the-art techniques for image feature extraction. As a part of experimentation, we train a benchmark model using different image extraction techniques and word embeddings. In the following sections, we discuss the literature of these techniques, the dataset used, the various steps were undertaken for image and text processing, and the model used. We detail different techniques and compare and contrast all the models using the BLEU score. Finally, we show how image captioning can be extended to our application of classifying images as sensitive.

## II. LITERATURE SURVEY

Image captioning has existed for a long time now, and over the years, its applications have grown manifold. The researchers have contributed to the domain in the form of algorithms, models, and significant networks for image feature extractions which see use in many areas of Computer Vision and image processing. Deep convolutional networks for image processing came to light in 2014, and this paper is highly inspired by the work done on deep CNNs and LSTMs.

In 2014, Karen Simonyan and Andrew Zisserman investigated the impact of depth of convolutional networks on the accuracy of large-scale image recognition in their paper 'Very Deep Convolutional Networks for Large-Scale Image Recognition'[1]. The research showed how increasing depth using 3x3 convolutional filters could significantly improve previous configurations using 16-19 weight layers. This network eventually secured the first position in the ImageNet Challenge 2014. Their two best ConvNet models are available free for use and research in computer vision.

In 2015, Kaiming He and others presented a residual learning framework in the paper 'Deep Residual Learning for Image Recognition'[2]. In this network, layers are explicitly reformulated as learning residual functions rather than learning unreferenced functions. This resulted in a deeper network, and it is easier to optimize because of its increased depth. It has 8x more complexity than previously existing VGG nets, leading to a 3.57% error on the ImageNet test set resulting in first place in the ILSVRC 2015 classification task. This further led to a significant improvement of 28% on the COCO object detection dataset, winning many other competitions.

In 2016, Christian Szegedy and others presented a state-of-the-art deep convolutional network in 'Rethinking the Inception Architecture for Computer Vision'[3]. In this research, they have described the factors improving the quality gains in tasks related to big-data and mobile vision scenarios, and they have explored ways of efficiently scaling networks. The methods are benchmarked on the ILSVRC 2012 classification challenge and resulted in a 21.2% increase in top-1 accuracy over the state-of-the-art. InceptionV3 is an ensemble network of 4 models, and multi-crop evaluation is carried out.

In 2017, François Chollet presented an interpretation of Inception modules in 'Xception: Deep Learning with Depthwise Separable Convolutions.'[4] The Xception network is an Inception module with many towers where Depth-wise separable convolutions replace inception. This architecture outperformed Inception V3 on both the ImageNet dataset and the larger image classification dataset. It has a similar number of parameters as Inception V3. Thus, the performance is improved due to the efficient use of model parameters and not increased capacity.

Marc Tanti et al. presented their 'Merge model' and emphasized the role of RNNs in Image captioning in 'Where to put the Image in an Image Caption Generator'[5] and 'What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator?'[6] in 2017. The former highlights the advantages of merge architecture, such as reducing the size of the RNN's hidden state vector and suggests multimodal integration rather than the joint encoding of images and captions. The latter suggests injecting the image features into RNN and only using it for encoding the linguistic features. Thus, it established RNNs as encoders and not generators.

## III. DATASET

For this research, we have used the Flickr8k dataset. Flickr is an American image and video hosting service founded in Canada in 2004. As the name suggests, this particular dataset contains around 8000 images, 8091 to be precise. There is another benchmark dataset by Flickr known as the Flickr30k, which included 31,783 images. However, choosing the Flickr8k dataset above benchmark datasets like MS-COCO and Flickr30k is the size of the dataset, i.e., 1.12 GB, which makes it easy to use it on devices with all computational capacities. Also, it is freely available to be downloaded from Kaggle and GitHub.

For every image in the dataset, there are five captions available in a text file. The dataset is divided into three parts: train, validation, and test, which includes 6000, 1000, and 1000 images each. The list of images to be classified as training and text are specified in different text files. All the captions are present in a single text file which can be processed accordingly.

## IV. TEXT PRE-PROCESSING

As there are five captions for each image file, we create a dictionary such that the image's filename is the key and the list of captions is the value. This process is done for all the images before splitting the dataset. Whenever we deal with data, it is essential to perform basic text cleaning, i.e., remove special tokens, eliminate numbers, omit stop words and change the case of all words to lower case. Advanced cleaning processes like stemming and tokenization are not required because we want to train the model on the actual captions. After data cleaning, we got a corpus with a vocabulary size of 8918, i.e., the number of unique words in 40,455 captions (8091*5) is 8918.

However, we would not be training the model for the complete vocabulary. We will only take words with more than one character, i.e., dropping words like 'a'. To make our model understand where the sentence starts and ends, adding a starting and ending token such as 'startseq' and 'endseq' is essential. Also, we need to include one extra token, which would be used for padding. A padding sequence is needed because we would be inputting captions as a sequence of word vectors. The length of this sequence shall be the maximum length of a caption in the training data. So, for all the captions with a length less than the maximum length, a padding token needs to be added to ensure all captions are of identical length. These tokens when added to the vocabulary makes the vocabulary size original+3.

The dependent variable in this use case is the caption. A caption is made of words existing in the vocabulary, and the model will predict a word depending on the previous word one by one to form a caption. As we know that we would be feeding the captions in the form of word vectors, we need to encode every word in the corpus with an index lying between 1 and the size of the corpus, both inclusive. Thus, we created two dictionaries, word_to_index and index_to_word, which we have also dumped using Pickle for external use. The mapping is as follows:

word_to_index['abc']= 1

index_to_words[1]='abc'

Thus, any given caption would look like: [ i, j, 0,0… n]

Where i and j are word indexes in the range [1, size of corpus] and 'n' is the maximum length of the captions in the training data. Here, 0 is the token used for padding the word vectors. For making the model understand the words and their relation, it is essential to input word embeddings instead of just the word vectors. For this purpose, we have tokenized the captions.

## V. IMAGE FEATURE EXTRACTION

Like we have processed the captions in our dataset, it is essential to process the images. In our case, images are the input, and to feed it to the neural network, we have to transform all the images into a vector of fixed size. For this purpose, we performed automatic feature engineering, i.e., used transfer learning. The crux of this research is to compare the efficiency of the pre-trained models for image feature extraction. This paper has used four state-of-the-art models: ResNet50, VGG-16, InceptionV3, and Xception. The architecture and reason for choosing this model are given below, whereas the configurations and the results for each are explained in the later sections. The purpose of comparing these models is to select the one capable of predicting the most quality captions and then use it for our application of sensitive image detection.

### *A. VGG-16*

While using VGG16 for image feature extraction, the images must be inputted as a (224x224x3) array. After processing the image through a deep convolutional neural network, the final output is a one-dimensional array of size 4096. The architecture shown below does not include the network's last layer because it doubles the size to 8194.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_1 (InputLayer) | (None, 224, 224, 3) | 0 |
| block1_conv1 (Conv2D) | (None, 224, 224, 64) | 1792 |
| block1_conv2 (Conv2D) | (None, 224, 224, 64) | 36928 |
| block1_pool (MaxPooling2D) | (None, 112, 112, 64) | 0 |
| block2_conv1 (Conv2D) | (None, 112, 112, 128) | 73856 |
| block2_conv2 (Conv2D) | (None, 112, 112, 128) | 147584 |
| block2_pool (MaxPooling2D) | (None, 56, 56, 128) | 0 |
| block3_conv1 (Conv2D) | (None, 56, 56, 256) | 295168 |
| block3_conv2 (Conv2D) | (None, 56, 56, 256) | 590080 |
| block3_conv3 (Conv2D) | (None, 56, 56, 256) | 590080 |
| block3_pool (MaxPooling2D) | (None, 28, 28, 256) | 0 |
| block4_conv1 (Conv2D) | (None, 28, 28, 512) | 1180160 |
| block4_conv2 (Conv2D) | (None, 28, 28, 512) | 2359808 |
| block4_conv3 (Conv2D) | (None, 28, 28, 512) | 2359808 |
| block4_pool (MaxPooling2D) | (None, 14, 14, 512) | 0 |
| block5_conv1 (Conv2D) | (None, 14, 14, 512) | 2359808 |
| block5_conv2 (Conv2D) | (None, 14, 14, 512) | 2359808 |
| block5_conv3 (Conv2D) | (None, 14, 14, 512) | 2359808 |
| block5_pool (MaxPooling2D) | (None, 7, 7, 512) | 0 |
| flatten (Flatten) | (None, 25088) | 0 |
| fc1 (Dense) | (None, 4096) | 102764544 |
| fc2 (Dense) | (None, 4096) | 16781312 |

Total params: 134,260,544
Trainable params: 134,260,544
Non-trainable params: 0

**Fig. 2.VGG-16 Model Architecture**

### *B. ResNet50*

The ResNet50 network includes 50 layers, including padding, activation, Max pooling, normalization, and concatenation. Similar to VGG16, the images are inputted as a (224x224x3) array. The global average pooling layer's second last layer results in a one-dimensional array of size 2048. The number of trainable parameters has significantly reduced in comparison to VGG16.

| ResNet50 |
|---|
| Input layer |
| Convolutional layer (7x7) |
| 3x3 MaxPooling |
| [1x1, 64; 3x3, 64; 1x1, 256] x3 |
| [1x1, 128; 3x3, 128; 1x1, 512] x4 |
| [1x1, 256; 3x3, 256; 1x1, 1024] x6 |
| [1x1, 512; 3x3, 512; 1x1, 2048] x3 |
| AveragePooling |
| Output (sigmoid neuron) |

**Fig. 3.ResNet50 Model Architecture[7]**

### *C. InceptionV3*

The Inception V3 network requires images to be inputted as a (299x299x3) array, and after processing through multiple inception modules, the output is extracted from the second last 'linear' layer. The result is a (1x2048) shape array just like ResNet50.

| Type | Kernel size/stride | Input size |
|---|---|---|
| Convolution | 3 × 3/2 | 299 × 299 × 3 |
| Convolution | 3 × 3/1 | 149 × 149 × 32 |
| Convolution | 3 × 3/1 | 147 × 147 × 32 |
| Pooling | 3 × 3/2 | 147 × 147 × 64 |
| Convolution | 3 × 3/1 | 73 × 73 × 64 |
| Convolution | 3 × 3/2 | 71 × 71 × 80 |
| Convolution | 3 × 3/1 | 35 × 35 × 192 |
| Inception module | Three modules | 35 × 35 × 288 |
| Inception module | Five modules | 17 × 17 × 768 |
| Inception module | Two modules | 8 × 8 × 1,280 |
| Pooling | 8×8 | 8 × 8 × 2,048 |
| Linear | Logits | 1 × 1 × 2,048 |
| Softmax | Output | 1 × 1 × 1,000 |

**Fig. 4.InceptionV3 Model Architecture[8]**

### *D. Xception*

While using the Xception network, the images are inputted as a (224x224x3) array similar to Inception V3. However, the Xception model is very deep compared to Inception, but the output is identical: a 2048-dimensional vector. We omit the last two layers to use this network- the fully connected layer and the classifier.

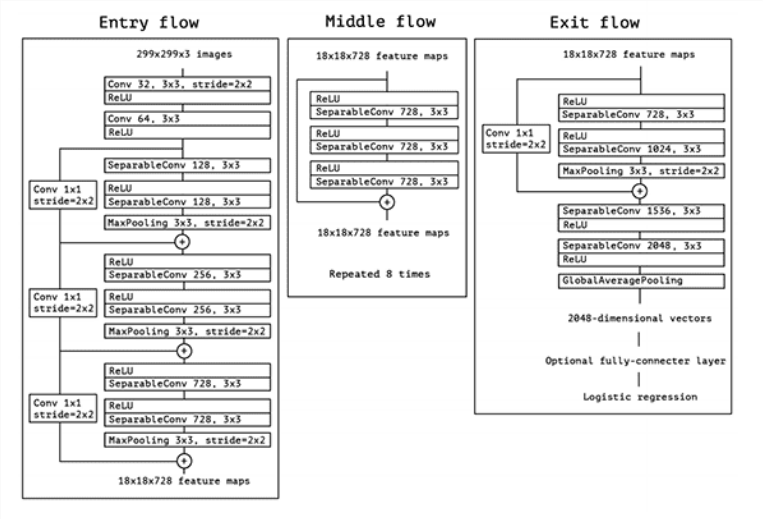


**Fig. 5.Xception Model Architecture**[4]

## VI. MODEL USED

There is not one but two inputs in our use case- image vector and partial captions, whereas we require full caption as the final output. This means that the image and caption in our supervised learning problem are not single but multiple data points. Thus, for every image, the data matrix shall look like follows:

**Fig. 6.Caption -> man is skiing on snowy hill**

TABLE I. WORD BY WORD GENERATION

| i | Image feature vector | Partial Caption | Target word |
|---|---|---|---|
| 1 | (1 x 2048 or 4096) | startseq | man |
| 2 | (1 x 2048 or 4096) | starseq man | is |
| 3 | (1 x 2048 or 4096) | starseq man is | skiing |
| 4 | (1 x 2048 or 4096) | starseq man is skiing | on |
| 5 | (1 x 2048 or 4096) | starseq man is skiing on | snowy |
| 6 | (1 x 2048 or 4096) | starseq man is skiing on snowy | hill |

The number of data points depends on the length of the caption. For processing such sequences, we have used be a Recurrent Neural Network. Due to multiple data points for every image in training, a vast data requirement is created. We have employed ImageDataGenerator class by Keras for this purpose as it does not require storing the complete data at once but just the current batch of points.

We are dealing with two different input types; we have used Keras' Sequential API to create merge models. The high-level architecture of the used model looks like this:

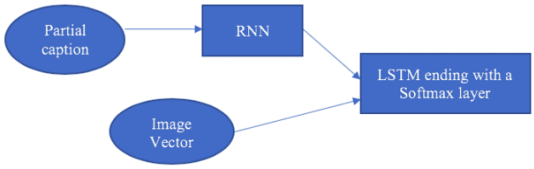


**Fig. 7.High Level Model Architecture**

*A. LSTM Model*

The model used for this research was defined based on the 'merge model' presented by Marc Tanti, Alber Gatt, and Kenneth P.Camilleri in their work. The model consists of an image feature extractor, a sequence processor, and a decoder. The image feature extractor inputs an image vector of size 2048 or 4096, and it is processed by a Dense layer resulting in a 256-element representation. The sequence processor takes sequences with a length of maximum caption in the training set and feeds it to an Embedding layer followed by a 256 memory units LSTM layer. These outputs are 256 element vectors regularized using a Dropout layer to avoid overfitting, and they are merged using addition operation. The decoder model merges the two and feeds it to a 256 units Dense layer and next to the final layer with activation as softmax used for predicting the following word in the sequence.

| Layer (type) | Output Shape | Connected to |
|---|---|---|
| input_2 (InputLayer) | (None, ? ) | |
| input_1 (InputLayer) | (None, ? ) | |
| embedding_1 (Embedding) | (None, ? , 256) | input_2[0][0] |
| dropout_1 (Dropout) | (None, x) | input_1[0][0] |
| dropout_2 (Dropout) | (None, ? ,256) | embedding_1[0][0] |
| dense_1 (Dense) | (None, 256) | dropout_1[0][0] |
| lstm_1 (LSTM) | (None, 256) | dropout_2[0][0] |
| add_1 (Add) | (None, 256) | dense_1[0][0]<br>lstm_1[0][0] |
| dense_2 (Dense) | (None, 256) | add_1[0][0] |
| dense_3 (Dense) | (None, n) | dense_2[0][0] |

**Fig. 8.Model Summary**

Here, '?' is the maximum length of the sequence, 'x' is the size of the image vector, and 'n' is the size of the training corpus. Though the size of the vocabulary is 8763, after cleaning, it drops to 7579. The maximum length of a caption in the training set comes out to be 34. For training our model, we have used a dropout rate of 0.5 and set the number of epochs as 20. A comprehensive comparison of the following is given in Table II.

**TABLE II. COMPARISON OF MODELS TRAINED**

| Attribute | Model 1 | Model 2 | Model 3 | Model 4 |
|---|---|---|---|---|
| Pre-trained model | VGG-16 | ResNet50 | InceptionV3 | Xception |
| Input shape (Image) | (224,224,3) | (224,224,3) | (299,299,3) | (299,299,3) |
| Output shape | (1,4096) | (1,2048) | (1,2048) | (1,2048) |
| Training Parameters | 5,527,963 | 5,003,675 | 5,003,675 | 5,003,675 |
| Training Loss | 3.0928 | 2.8485 | 2.6685 | 2.5990 |

## VII. RESULTS AND EVALUATION

After successfully training the four models, the epoch with the minimum loss for each was saved for further use. The goal is to select the model capable of generating quality captions. We evaluated the models on the test set (1000 images) using the BLEU [9] score. BLEU stands for Bilingual Evaluation Understudy, and it is a numerical value lying between 0 and 1, both included. This score helps to compare the predicted caption with the given five reference captions (corpus) and outputs a cumulative score for all the images in the test set. This research computes 4 BLEU scores to compute the precision for n-grams of size 1 to 4. A Higher BLEU score indicates better quality captions. Table III and IV show the performance of our models on the test set.

**TABLE III. EVALUATION OF ALL MODELS**

| Pre-Trained Model | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 |
|---|---|---|---|---|
| ResNet50 | 0.519163 | 0.279260 | 0.195061 | 0.093271 |
| VGG16 | 0.500829 | 0.258246 | 0.175336 | 0.079138 |
| InceptionV3 | 0.519715 | 0.280572 | 0.191283 | 0.089288 |
| Xception | 0.525859 | 0.298624 | 0.212307 | 0.102526 |

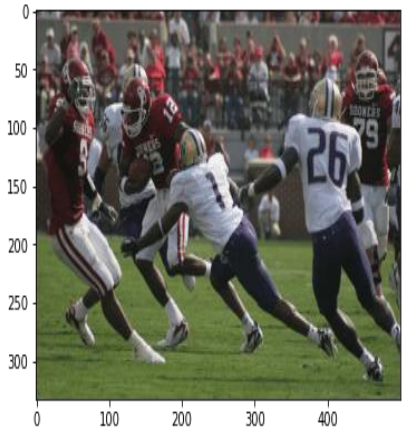


**Fig. 9.Sample Image**

**TABLE IV. CAPTIONS GENERATED BY MODELS**

| **Model** | **Generated Caption for Fig. 9.** |
|---|---|
| ResNet50 + LSTM | two men attempt to baseball teams are playing soccer on field |
| VGG16 + LSTM | two men are playing soccer in the field |
| InceptionV3 + LSTM | baseball player in red and white uniform is running |
| Xception + LSTM | two men are playing rugby |

Here, we validate that the Xception model generates the most quality captions for our dataset. For the given example, though the model only predicts the presence of two men, this is sufficient for our purpose of sentiment analysis, as we see in the final section.

## VIII. APPLICATION

The end goal of this research is to use the model generating the most quality captions to evaluate the image's sentiment. If the sentiment of a caption is negative, it might be because it talks about violence, includes derogatory terms, or radiates negativity. For sentiment analysis, we can either train the model on the given training set or use a sentiment analysis tool. As we do not have the captions labeled, it is better to use a tool that learns on a large corpus. Thus, we have used VADER [10] (Valence Aware Dictionary and sEntiment Reasoner) Sentiment Analysis. VADER is a sentiment analyzer capable of analyzing the sentiments on social media. As the primary source of our images is the Internet and image captioning finds excellent use in Social Media and other forms of media, it is the right choice. The tool is capable of categorizing the text as positive or negative and provides a positivity and negativity score.

With the help of Sentiment Intensity Analyzer, we get four scores- positive, negative, neutral, and compound. In this research, all the sentences with a compound sentiment score greater than 0.05 are classified as positive, less than -0.05 as negative, and neutral otherwise. Here is what we mean by these terms:

```
1st statement :
Two boys are playing on the street
Overall sentiment dictionary is :  {'neg': 0.0, 'neu': 0.769, 'pos': 0.231, 'compound': 0.2023}
sentence was rated as  0.0 % Negative
sentence was rated as  76.9 % Neutral
sentence was rated as  23.1 % Positive
Sentence Overall Rated As Positive

2nd Statement :
Two boys are out on the street
Overall sentiment dictionary is :  {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
sentence was rated as  0.0 % Negative
sentence was rated as  100.0 % Neutral
sentence was rated as  0.0 % Positive
Sentence Overall Rated As Neutral

3rd Statement :
Two boys are fighting on the street
Overall sentiment dictionary is :  {'neg': 0.294, 'neu': 0.706, 'pos': 0.0, 'compound': -0.3612}
sentence was rated as  29.4 % Negative
sentence was rated as  70.6 % Neutral
sentence was rated as  0.0 % Positive
Sentence Overall Rated As Negative
```

**Fig. 10. Example of VADER Sentiment Analyzer**

In this example, we can see that 'fighting' is classified as harmful by VADER, and thus, the sentiment of the whole sentence is negative. When we obtain an image as the independent variable, we obtain a caption which we check for sentiment as follows:

**Fig. 11. Sample Image with negative expected sentiment**

**Input:** Image processed through OpenCV library

**Caption:** A man lose control of his watercraft

**Sentiment:** Overall sentiment dictionary is :

{'neg': 0.31, 'neu': 0.69, 'pos': 0.0, 'compound': -0.4019}

sentence was rated as 31.0 % Negative

sentence was rated as 69.0 % Neutral

sentence was rated as 0.0 % Positive

Sentence Overall Rated As **Negative**

Here, we observe that the picture suggests some danger or accident situation, which is rated as negative depending on the generated caption.

## CONCLUSION

The research started with studying the state-of-the-art feature extraction methods for images. We stuck to the four deep convolutional networks that came out between 2014 to 2017 for our goal. With multiple networks, a natural question arose as to which one was the best in terms of training and generating captions. As expected, the Xception model performed much better than the previous state-of-the-art algorithms regarding low training loss and high BLEU scores. Thus, we can establish an inversely proportional relationship between the two.

Further, we demonstrated the use of image captioning technology in image classification. We capture the overall sentiment of the generated caption to label the image as sensitive or not. The results are reasonably good given the number of epochs we have trained the model for and the computing resources.

The dataset we have used here is a memory-efficient one. However, for real-life applications, more extensive datasets and more goal-oriented datasets can be chosen. The models can be significantly improved by using GloVe word embeddings, increasing the number of epochs, and adding layers such as the RepeatVector and Bidirectional LSTM[11] to the network. Not only this, we wish to take this approach further to video captioning and make it a system capable of continuous learning, i.e., a training model based on captions entered by the user for an image. New research in the domain using PyTorch and Transformers has shown a significant hike in BLEU scores, and going ahead, we would like to implement one of those for our system.

## REFERENCES

[1] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
[2] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
[3] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).
[4] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1251-1258).

[5] Tanti, M., Gatt, A., & Camilleri, K. P. (2018). Where to put the image in an image caption generator. *Natural Language Engineering*, *24*(3), 467-489.
[6] Tanti, M., Gatt, A., & Camilleri, K. P. (2017). What is the role of recurrent neural networks (rnns) in an image caption generator?. *arXiv preprint arXiv:1708.02043*.
[7] A. Kwasigroch, A. Mikołajczyk and M. Grochowski, "Deep neural networks approach to skin lesions classification — A comparative analysis," 2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR), 2017, pp. 1069-1074, doi: 10.1109/MMAR.2017.8046978.
[8] Feng, Chuncheng & Zhang, Hua & Wang, Shuang & Li, Yonglong & Wang, Haoran & Yan, Fei. (2019). Structural Damage Detection using Deep Convolutional Neural Network and Transfer Learning. KSCE Journal of Civil Engineering. 23. 10.1007/s12205-019-0437-z.
[9] Papineni, Kishore & Roukos, Salim & Ward, Todd & Zhu, Wei Jing. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. 10.3115/1073083.1073135.
[10] Hutto, C., & Gilbert, E. (2014, May). Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the International AAAI Conference on Web and Social Media* (Vol. 8, No. 1).
[11] Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3156-3164).