A project report on

IMAGE CAPTIONING: AN ASSISTIVE TECHNOLOGY

submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology (IT)

Under the supervision of

Er. Narinder Kaur
Assistant Professor
IT Department

Submitted by

Yashika Chugh

40214803118

DEPARTMENT OF INFORMATION TECHNOLOGY MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY SECTOR-22, ROHINI, DELHI –110086

December 2021

ACKNOWLEDGEMENT

I would like to express my gratitude and appreciation to all those who gave me the possibility to complete this report. A special thanks to my project guides, whose help, stimulating suggestions and encouragement helped me to coordinate my project especially in writing this report.

However, it would not have been possible without the kind support and help of many individuals and organization. I would like to extend my sincere thanks to all of them.

I am highly indebted to Ms. Narinder Kaur for her guidance and constant supervision as well as for providing necessary information regarding the project and also for her support in completing the project.

Yashika Chugh

40214803118

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this project report titled, **IMAGE CAPTIONING: AN ASSISTIVE TECHNOLOGY** submitted by me in the partial fulfillment of the requirement of the award of the degree of **Bachelor of Technology (B.Tech.)** Submitted in the Department of **Information Technology,** Maharaja Agrasen Institute of Technology is an authentic record of my project work carried out under the guidance of Er. Narinder Kaur, Assistant Professor (IT Department). The matter presented in this project report has not been submitted either in part or full to any university or Institute for award of any degree.

Date: **Yashika Chugh**

Place: Delhi 40214803118

SUPERVISOR'S CERTIFICATE

It is to certify that the Project entitled **IMAGE CAPTIONING: AN ASSISTIVE TECHNOLOGY** which is being submitted by **Ms. YASHIKA CHUGH** to the Maharaja Agrasen Institute of Technology, Rohini in the fulfillment of the requirement for the award of the degree of **Bachelor of Technology (B.Tech.)**, is a record of bonafide project work carried out by him/her under my/ our guidance and supervision.

Er. Narinder Kaur
Assistant Prof.
Department of Information Technology

Prof. (Dr.) M.L.Sharma
H.O.D.
Department of Information Technology

List of Figures

List of Tables

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.

TABLE OF CONTENTS

TITLE **PAGE NO.**

Chapter 1 Introduction

Automatically generating captions to an image shows the understanding of the image by computers, which is a fundamental task of intelligence. For a caption model it not only need to find which objects are contained in the image and also need to be able to expressing their relationships in a natural language such as English. In our project, we do image-to-sentence generation.

This application bridges vision and natural language. If we can do well in this task, we can then utilize natural language processing technologies understand the world in images. This project shows some great promises such as building assistive technologies for visually impaired people and help automating caption tasks on the internet. The problem statement is shown in Fig. 1.1.

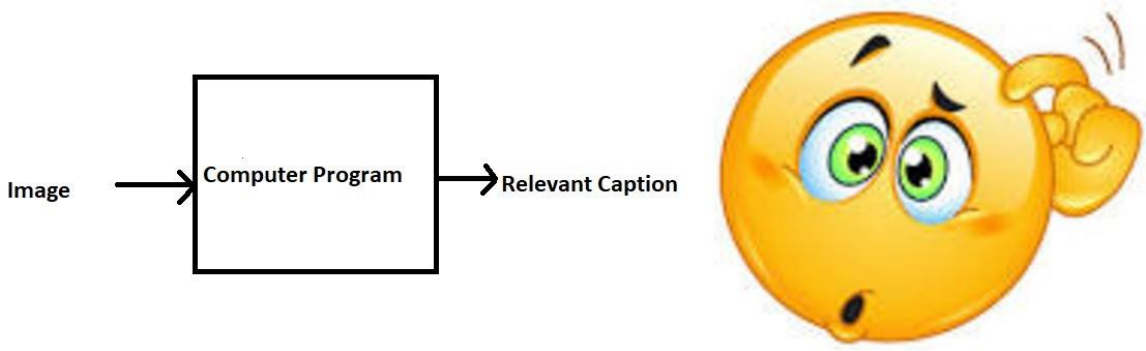

Fig. 1.1. Problem Statement

Automatic Captioning can help, make Google Image Search as good as Google Search, as then every image could be first converted into a caption and then search can be performed based on the caption.

1.1 Need of the Study

- Some detailed use cases would be like an visually impaired person taking a picture from his phone and then the caption generator will turn the caption to speech for him to understand.
- Advertising industry trying the generate captions automatically without the need to make them separately during production and sales.

- Doctors can use this technology to find tumours or some defects in the images or used by people for understanding geospatial images where they can find out more details about the terrain.

1.2 Scope of the Study

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.

1.3 Objective of the Study

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.

Chapter 2 Literature Review

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.

Chapter 3 Implementation of Proposed Method

3.1 Dataset used

The Flickr 8k dataset provided by the University of Illinois at Urbana-Champaign is used as training a model with large number of images would not be feasible on a system which is not a very high-end PC/Laptop. This dataset contains 8000 images each with 5 captions. These images are bifurcated as follows:

- Training Set — 6000 images
- Dev Set — 1000 images
- Test Set — 1000 images

3.2 Data Understanding

Every image in the dataset has 5 captions stored in a text file "Flickr8k.token.txt". Thus every line contains the <image name>#i <caption>, where 0≤i≤4, i.e. the name of the image, caption number (0 to 4) and the actual caption as shown in Fig. 3.1.

```
1   101654506_8eb26cfb60.jpg#0      A brown and white dog is running through the snow .
2   101654506_8eb26cfb60.jpg#1      A dog is running in the snow
3   101654506_8eb26cfb60.jpg#2      A dog running through snow .
4   101654506_8eb26cfb60.jpg#3      a white and brown dog is running through a snow covered
5   101654506_8eb26cfb60.jpg#4      The white and brown dog is running over the surface of t
6
7   1000268201_693b08cb0e.jpg#0     A child in a pink dress is climbing up a set of stairs i
8   1000268201_693b08cb0e.jpg#1     A girl going into a wooden building .
9   1000268201_693b08cb0e.jpg#2     A little girl climbing into a wooden playhouse .
10  1000268201_693b08cb0e.jpg#3     A little girl climbing the stairs to her playhouse .
11  1000268201_693b08cb0e.jpg#4     A little girl in a pink dress going into a wooden cabin
```

Fig. 3.1. Sample Text File

We create a dictionary named “descriptions” which contains the name of the image (without the .jpg extension) as keys and a list of the 5 captions for the corresponding image as values. For example with reference to the above screenshot the dictionary will look as follows:

```
descriptions['101654506_8eb26cfb60'] = ['A brown and white dog is running through the snow .', 'A dog is running in the snow', 'A dog running through snow .', 'a white and brown dog is running through a snow covered field .', 'The white and brown dog is running over the surface of the snow .']
```

3.3 Data Cleaning

Some basic cleaning needs to be performed like lower-casing all the words (otherwise “hello” and “Hello” will be regarded as two separate words), removing special tokens (like ‘%’, ‘$’, ‘#’, etc.), eliminating words which contain numbers (like ‘hey199’, etc.).

A vocabulary of all the unique words present across all the 8000*5 (i.e. 40000) image captions (**corpus**) in the data set is created. The vocabulary size is 8425. This means we have 8425 unique words across all the 40000 image captions. All these captions along with their image names are written in a new file namely, “*descriptions.txt”* and saved on the disk.

Since we are creating a predictive model, we would not like to have all the words present in our vocabulary but the words which are more likely to occur or which are common. This helps the model become more **robust to outliers** and make less mistakes. Hence we consider only those words which **occur at least 10 times** in the entire corpus

So now we have only 1845 unique words in our vocabulary. However, we will append 0’s (for padding) and start and end sequences, thus total words = 1845+3 = **1848** (one index for the 0).

3.4 Loading the training set

The text file “Flickr_8k.trainImages.txt” which contains the names of the images that belong to the training set is loaded. Thus, we have separated the 6000 training images in the list named “train”. and the descriptions of these images from “descriptions.txt” (saved on the hard disk) in the Python

dictionary "train_descriptions". However, on loading, two tokens are added in every caption as follows:

'**s**' -> This is a start sequence token which will be added at the start of every caption
'**e**' -> This is an end sequence token which will be added at the end of every caption.

3.5 Pre-processing of Images

Images are nothing but input (X) to our model. We need to convert every image into a fixed sized vector which can then be fed as input to the neural network. For this purpose, we opt for transfer learning. This process is called automatic feature engineering. For our purpose, we used three pre-trained models separately: ResNet50 with 2048 length vector and InceptionV3 and VGG-16 with 4096 length vectors.

We save all the bottleneck train features in a Python dictionary and save it on the disk using Pickle file, namely "**encoded_train_images.pkl**" whose keys are image names and values are corresponding 2048 or 4096 length feature vector. Similarly we encode all the test images and save them in the file "**encoded_test_images.pkl**".

3.6 Pre-processing of Text

The captions are something that we want to predict. So during the training period, captions will be the target variables (Y) that the model is learning to predict. But the prediction of the entire caption, given the image does not happen at once. We will predict the caption **word by word**. Thus, we need to encode each word into a fixed sized vector. Stating simply, we will represent every unique word in the vocabulary by an integer (index). As seen above, we have 1848 unique words in the corpus and thus each word will be represented by an integer index between 1 to 1848. These two Python dictionaries can be used as follows:

- wordtoix['abc'] -> returns index of the word 'abc'
- ixtoword[k] -> returns the word whose index is 'k'

There is one more parameter that we need to calculate, i.e., the maximum length of a caption. The maximum length can range from 30 to 35 for different corpuses.

3.7 Data Generator

Now let's try to frame it as a **supervised learning problem** where we have a set of data points D = {Xi, Yi}, where Xi is the feature vector of data point 'i' and Yi is the corresponding target variable. Let's take the first image vector **Image_1** and its corresponding caption "**startseq the black cat sat on grass endseq**". Recall that, Image vector is the input and the caption is what we need to predict. But the way we predict the caption for a sample image Fig.3.2. is as follows:

Fig. 3.2. Caption -> The black cat is walking on grass

For the first time, we provide the image vector and the first word as input and try to predict the second word, i.e.: Input = Image_1 + 'startseq'; Output = 'the'

Then we provide image vector and the first two words as input and try to predict the third word, i.e.: Input = Image_1 + 'startseq the'; Output = 'cat'. And so on…

It must be noted that, one image+caption is **not a single data point** but are multiple data points depending on the length of the caption. Similarly if we consider both the images and their captions, our data matrix will then look as follows as shown in Fig. 3.3.:

	Xi		Yi
i	Image feature vector	Partial Caption	Target word
1	Image_1	startseq	the
2	Image_1	startseq the	black
3	Image_1	startseq the black	cat
4	Image_1	startseq the black cat	sat
5	Image_1	startseq the black cat sat	on
6	Image_1	startseq the black cat sat on	grass
7	Image_1	startseq the black cat sat on grass	endseq
8	Image_2	startseq	the
9	Image_2	startseq the	white
10	Image_2	startseq the white	cat
11	Image_2	startseq the white cat	is
12	Image_2	startseq the white cat is	walking
13	Image_2	startseq the white cat is walking	on
14	Image_2	startseq the white cat is walking on	road
15	Image_2	startseq the white cat is walking on road	endseq

data points corresponding to image 1 and its caption

data points corresponding to image 2 and its caption

Fig. 3.3. Data Matrix for both the images and captions

We must now understand that in every data point, it's not just the image which goes as input to the system, but also, a partial caption which helps to **predict the next word in the sequence.**
Since we are processing **sequences**, we will employ a **Recurrent Neural Network** to read these partial captions. Since we would be doing **batch processing** (explained later), we need to make sure that each sequence is of **equal length**. Hence we need to **append 0's** (zero padding) at the end of each sequence which will lead to every sequence having a length of 34.

This would create a huge data requirement. For this reason, we use data generators a lot in Deep Learning. Data Generators are a functionality which is natively implemented in Python. The ImageDataGenerator class provided by the Keras API is nothing but an implementation of generator function in Python. This means that we do not require to store the entire dataset in the memory at once. Even if we have the current batch of points in the memory, it is sufficient for our purpose.

3.8 Model Architecture

Since the input consists of two parts, an image vector and a partial caption, we cannot use the Sequential API provided by the Keras library. For this reason, we use the Functional API which allows us to create Merge Models. The training has been done on 6 different models explained below in Fig. 3.4.:

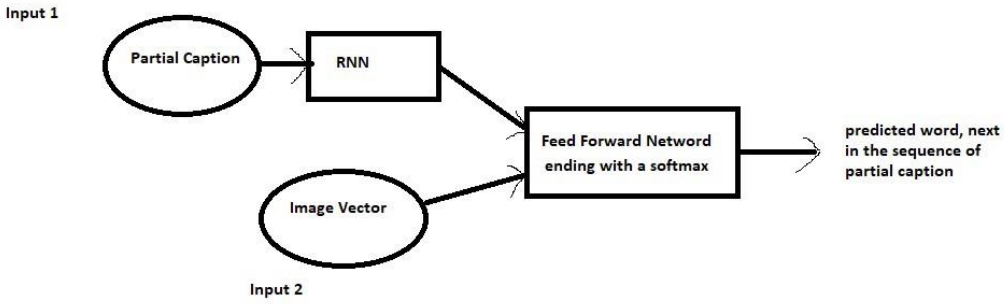

Fig. 3.4. High Level Architecture for all models

3.9 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. Fig. 3.5. shows 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. 3.5. Example of VADER Sentiment Analyzer

Chapter 4 Comparison and Results

4.1 Comparison

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.

4.1.1.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. Fig. 4.1. shows the model architecture of VGG-16.

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. 4.1. VGG-16 Model Architecture

4.1.2 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. Fig. 4.2. shows the model architecture of ResNet50.

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. 4.2. ResNet50 Model Architecture[7]

4.1.3 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. The model architecture is shown in the Fig. 4.3.

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. 4.3. InceptionV3 Model Architecture [8]

4.1.4 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. The architecture is shown in Fig. 4.4.

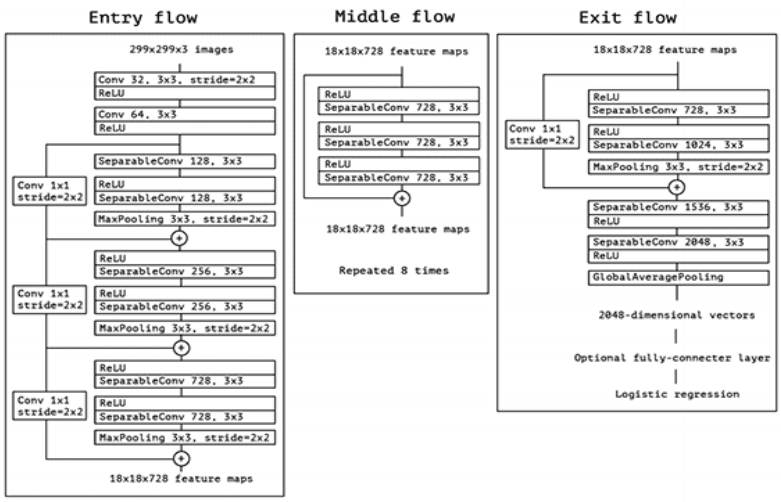

Fig. 4.4. Xception Model Architecture

4.2 Results of Image Captioning

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 shown in table 4.1.:

Table 4.1. 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

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. The comparison is shown in Table 4.2.

Table 4.2. Evaluation of all models

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

Here, we validate that the Xception model generates the most quality captions for our dataset.

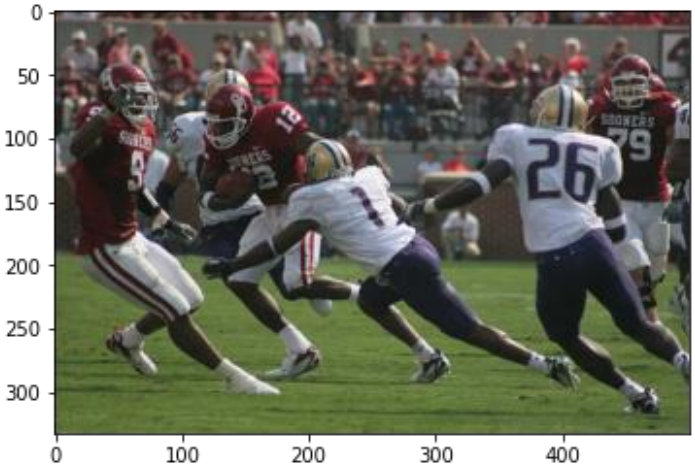

Fig. 4.5. Sample Image

The captions generated for our sample image Fig. 4.5. are shown in table 4.3.

Table 4.3. Caption generated by models

Model	**Generated Caption for Fig. 11.**
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

4.3 Results of Sentiment Analysis

When we obtain an image as the independent variable, we obtain a caption which we check for sentiment for a sample image Fig.4.6. as follows:

Fig. 4.6. 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.

4.4 Evaluation

These generated captions are compared to the actual captions from the dataset and evaluated using BLEU scores as the evaluation metrics. A score closer to 1 indicates that the predicted and actual captions are very similar. As the scores are calculated for the whole test data, we get a mean value which includes good and not so good captions. Some of the examples of good and bad captions can be seen below:

true: black dog and spotted dog are fighting

pred: black and white dog is running on the grass

BLEU: 0.7598356856515925

true: black dog running in the surf

pred: dog is running through the water

BLEU: 0.8408964152537145

true: man drilling hole in the ice

pred: man in black shirt is jumping over the water

BLEU: 0.7598356856515925

Fig 4.7. Good captions

true: little girl covered in paint sits in front of painted rainbow with her hands in bowl

pred: man in red shirt is standing on the street

BLEU: 0.2652496231798079

true: man lays on bench while his dog sits by him

pred: boy in red shirt is jumping over the camera

BLEU: 0

true: collage of one person climbing cliff

pred: man in red shirt is standing on the air

BLEU: 0

Fig 4.8. Bad captions

4.5 Running Application

The Flask application of the project is shown below in Fig. 4.9.

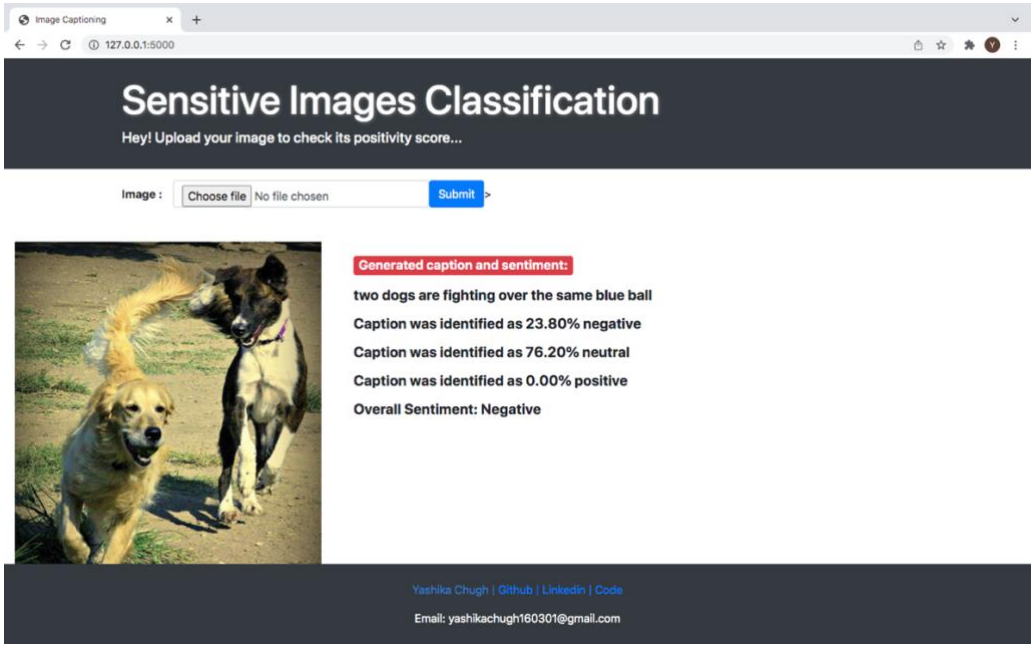

Fig.4.9. Flask Application

Chapter 5 Conclusion and Further Scope

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).