

Automatic Image Captioning

Group – 15 / Cohort – 20

Guide: Prof. Vineet Gandhi

Mentor: Brahmani Nutakki

Team: Hanumanth Sangewar,

Raheem Baig,

Ritesh K Singh

**­**

Table of Contents

[Acknowledgements 3](#_Toc143469026)

[Abstract 3](#_Toc143469027)

[Goal 3](#_Toc143469028)

[Overview of Image Captioning 3](#_Toc143469029)

[Proposed Approach 5](#_Toc143469030)

[Pre-Trained Models used 6](#_Toc143469031)

[Implementation details 10](#_Toc143469032)

[Deployment 13](#_Toc143469033)

[Learnings 14](#_Toc143469034)

[Challenges 15](#_Toc143469035)

[Real-world application 16](#_Toc143469036)

[Conclusion 16](#_Toc143469037)

[Next Steps/Opportunities 17](#_Toc143469038)

[References 17](#_Toc143469039)

Acknowledgements

We would like to extend our gratitude to our Faculties from IIIT-Hyderabad, TalentSprint and mentor for guiding us. Starting from routing us towards correct research papers and datasets, in solving all the setup issues, suggesting tools that help us in the development and deployment, along with her monitoring skills that kept us on track, in every stage of the project.

Abstract

Automatically describing the content of an image is a fundamental problem in artificial intelligence that connects computer vision and natural language processing.

Goal

Build an image captioning model to generate captions of an image using CNN.

In this project, we need to create a multimodal neural network that involves the concept of Computer Vision and Natural Language Process in recognizing the context of images and describing them in natural languages (English). Deploy the model and evaluate the model on different real-time images.

Overview of Image Captioning

Image captioning is the task of transforming objects into words. An Image captioning model's job is to provide a natural language description of the content in an image. Not only that, but it also needs to capture how the objects are related to each other, their attributes and the activities they are involved.

Image captioning spans the fields of computer vision and natural language processing. The image captioning task generalizes object detection where the descriptions are a single word. Recently, most research on image captioning has focused on deep learning techniques, especially Encoder-Decoder models with Convolutional Neural Network (CNN) feature extraction.

Approaches to image captioning can be divided into the following categories [2]:

* Systems that rely on computer vision techniques to extract object detections and features from the source image, using these as input to a Natural Language Generation stage.
* Systems that frame the task as a retrieval problem, where a caption, or parts thereof, is identified by computing the proximity/relevance of strings in the training data to a given image. This is done by exploiting either a unimodal or multimodal space. Many retrieval-based approaches rely on neural models to handle both image features and linguistic information.
* Systems that also rely on neural models, but rather than performing partial or wholesale caption retrieval, generate novel captions using a recurrent neural network (RNN), usually a long short-term memory (LSTM). Typically, such models use image features extracted from a pre-trained convolutional neural network (CNN) such as the VGG CNN to bias the RNN towards sampling terms from the vocabulary in such a way that a sequence of such terms produces a caption that is relevant to the image.
* Most conventional systems also utilize an encoder-decoder framework, in which an input image is encoded into an intermediate representation of the information contained within the image, and subsequently decoded into a descriptive text sequence. This encoding can consist of a single feature vector output of a CNN, or multiple visual features obtained from different regions within the image. In the latter case, the regions can be uniformly sampled, or guided by an object detector which has been shown to yield improved performance.

Image Captioning can be regarded as an end-to-end Sequence to Sequence problem, as it converts images, which are regarded as a sequence of pixels to a sequence of words. For this purpose, we need to process both the language or statements and the images.



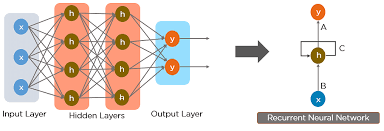
If someone is asked to describe this image one would describe it as: “A dog wearing pink glasses in front of a blue wall sitting on a white towel”. So, how are we doing this? While forming the description, we are seeing the image but at the same time, we are looking to create a meaningful sequence of words. The first part is handled by CNNs.

CNN sample architecture



and the second is handled by RNNs.

RNN sample architecture



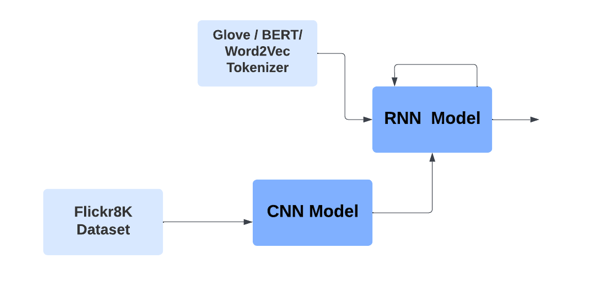
If we can obtain a suitable dataset with images and their corresponding human descriptions, we can train networks to automatically caption images. FLICKR 8K, FLICKR 30K, and MS-COCO are some most used datasets for this purpose.

Proposed Approach

For our Project, we used Flickr\_8K dataset. It is a collection of sentence-based image descriptions. It consists of 8k images in JPEG format with different shapes and sizes. All images are paired with five different captions which provide clear descriptions of the salient entities and events.

* Selected model based on the research and reference code
* CNN for training images – used VGG16 to extract the features
* Applied Various Text Clean/process techniques and Tokenized
* Caption Generation - Used LSTM to generate captions from the feature vectors and tokens fed to it.
* Once done for one, implement more pre-trained models

Refer to the image below on high level implementation of Image Captioning model design,



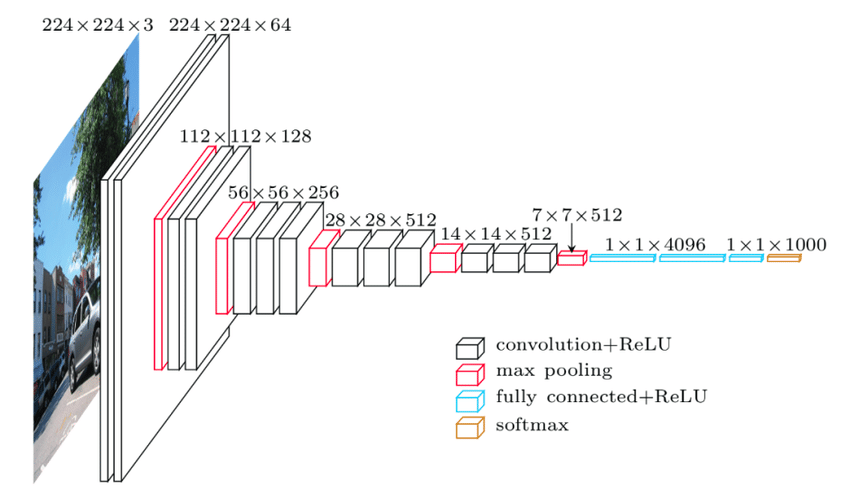
Pre-Trained Models used

VGG-16:

VGG16 refers to the VGG model, also called VGGNet. It is a convolution neural network (CNN) model supporting 16 layers. K. Simonyan and A. Zisserman from Oxford University proposed this model and published it in a paper called Very Deep Convolutional Networks for Large-Scale Image Recognition.

The VGG16 model can achieve a test accuracy of 92.7% in ImageNet, a dataset containing more than 14 million training images across 1000 object classes. It is one of the top models from the ILSVRC-2014 competition.

VGG16 improves on AlexNet and replaces the large filters with sequences of smaller 3×3 filters. In AlexNet, the kernel size is 11 for the first convolutional layer and 5 for the second layer. The researchers trained the VGG model for several weeks using NVIDIA Titan Black GPUs.



Source: [ResearchGate](https://www.researchgate.net/profile/Timea-Bezdan/publication/333242381/figure/fig2/AS:760979981860866@1558443174380/VGGNet-architecture-19.ppm)

VGG architecture:

Input—VGGNet receives a 224×224 image input. In the ImageNet competition, the model’s creators kept the image input size constant by cropping a 224×224 section from the centre of each image.

Convolutional layers—the convolutional filters of VGG use the smallest possible receptive field of 3×3. VGG also uses a 1×1 convolution filter as the input’s linear transformation.

ReLu activation—next is the Rectified Linear Unit Activation Function (ReLU) component, AlexNet’s major innovation for reducing training time. ReLU is a linear function that provides a matching output for positive inputs and outputs zero for negative inputs. VGG has a set convolution stride of 1 pixel to preserve the spatial resolution after convolution (the stride value reflects how many pixels the filter “moves” to cover the entire space of the image).

Hidden layers—all the VGG network’s hidden layers use ReLU instead of Local Response Normalization like AlexNet. The latter increases training time and memory consumption with little improvement to overall accuracy.

Pooling layers–A pooling layer follows several convolutional layers—this helps reduce the dimensionality and the number of parameters of the feature maps created by each convolution step. Pooling is crucial given the rapid growth of the number of available filters from 64 to 128, 256, and eventually 512 in the final layers.

Fully connected layers—VGGNet includes three fully connected layers. The first two layers each have 4096 channels, and the third layer has 1000 channels, one for every class.

InceptionV3:

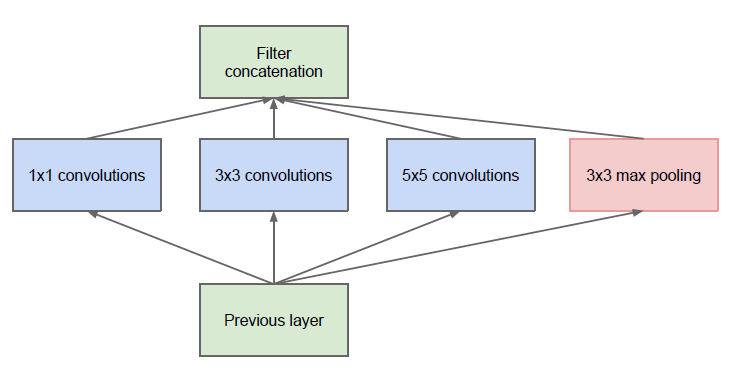
The Inception V3 is a deep learning model based on Convolutional Neural Networks, which is used for image classification. The inception V3 is a superior version of the basic model Inception V1 which was introduced as GoogLeNet in 2014. As the name suggests it was developed by a team at Google.

The basic module of the Inception V1 model is made up of four parallel layers.

* + 1×1 convolution
  + 3×3 convolution
  + 5×5 convolution
  + 3×3 max pooling

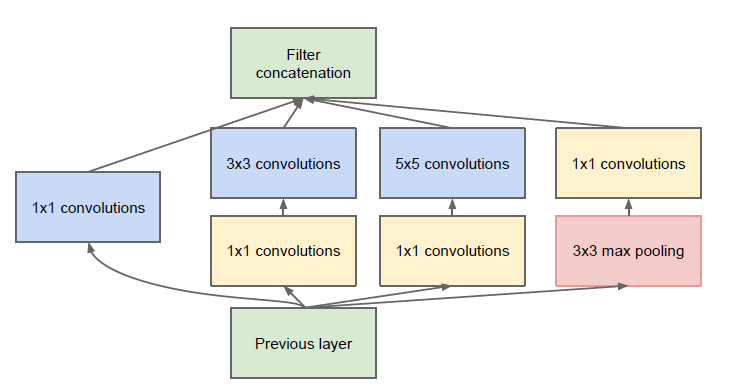
Convolution - The process of transforming an image by applying a kernel over each pixel and its local neighbours across the entire image.

Pooling - Pooling is the process used to reduce the dimensions of the feature map. There are different types of pooling, but the most common ones are max pooling and average pooling.



This module of the Inception V1 is called the Naive form. One of the drawbacks of this naive form is that even the 5×5 convolutional layer is computationally pretty expensive i.e., time-consuming and requires high computational power.

To overcome this the authors added a 1×1 convolutional layer before each convolutional layer, which results in reduced dimensions of the network and faster computations.

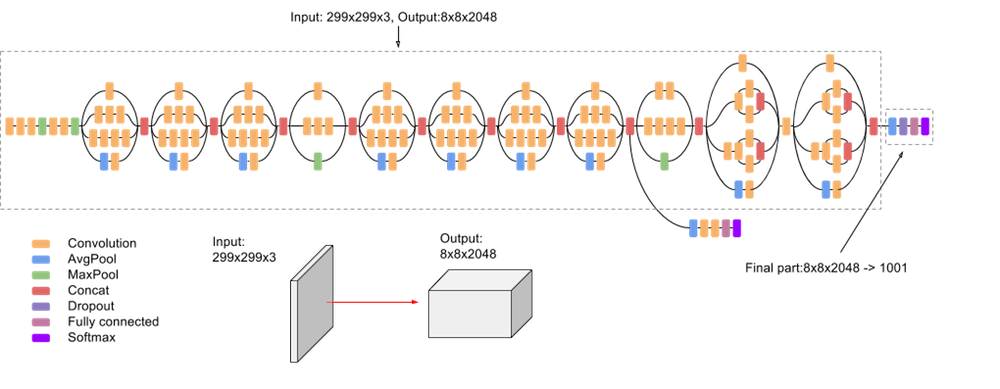


The inception V3 is just the advanced and optimized version of the inception V1 model. The Inception V3 model used several techniques for optimizing the network for better model adaptation.

* + It has higher efficiency
  + It has a deeper network compared to the Inception V1 and V2 models, but its speed isn't compromised.
  + It is computationally less expensive.
  + It uses auxiliary Classifiers as regularizes.

The major modifications done on the Inception V3 model are:

* + Factorization into Smaller Convolutions
  + Spatial Factorization into Asymmetric Convolutions
  + Utility of Auxiliary Classifiers
  + Efficient Grid Size Reduction

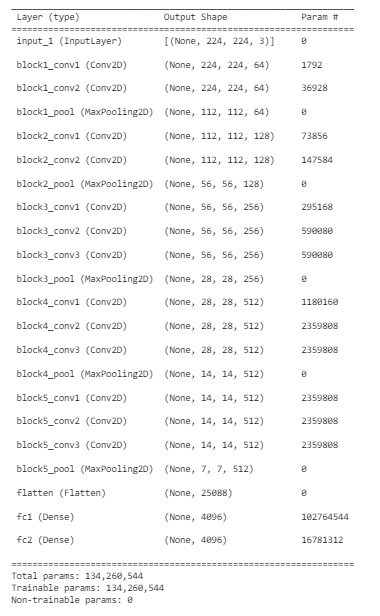


Source: <https://iq.opengenus.org/inception-v3-model-architecture/>

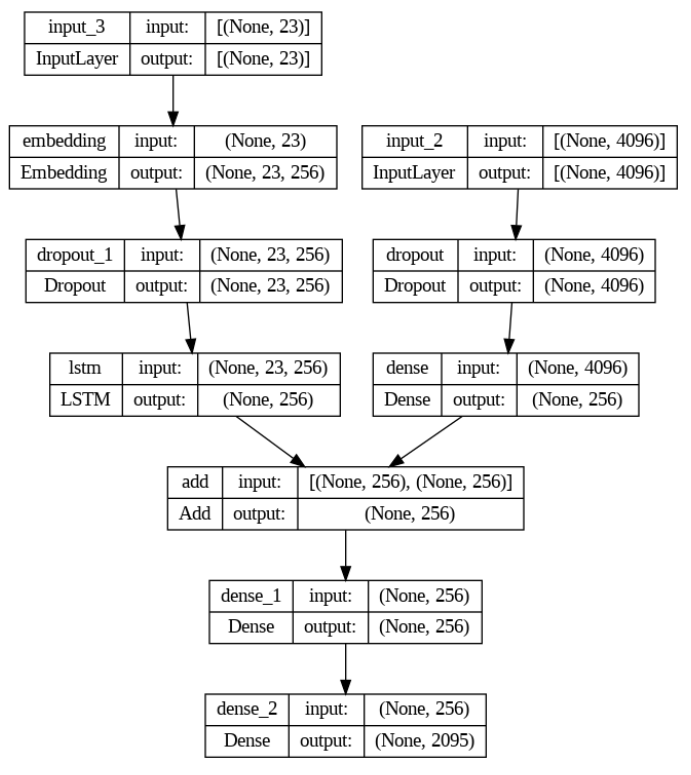
Implementation details

**Model Design**

VGG16 Pretrained model to generate features of Flicker 8K dataset:

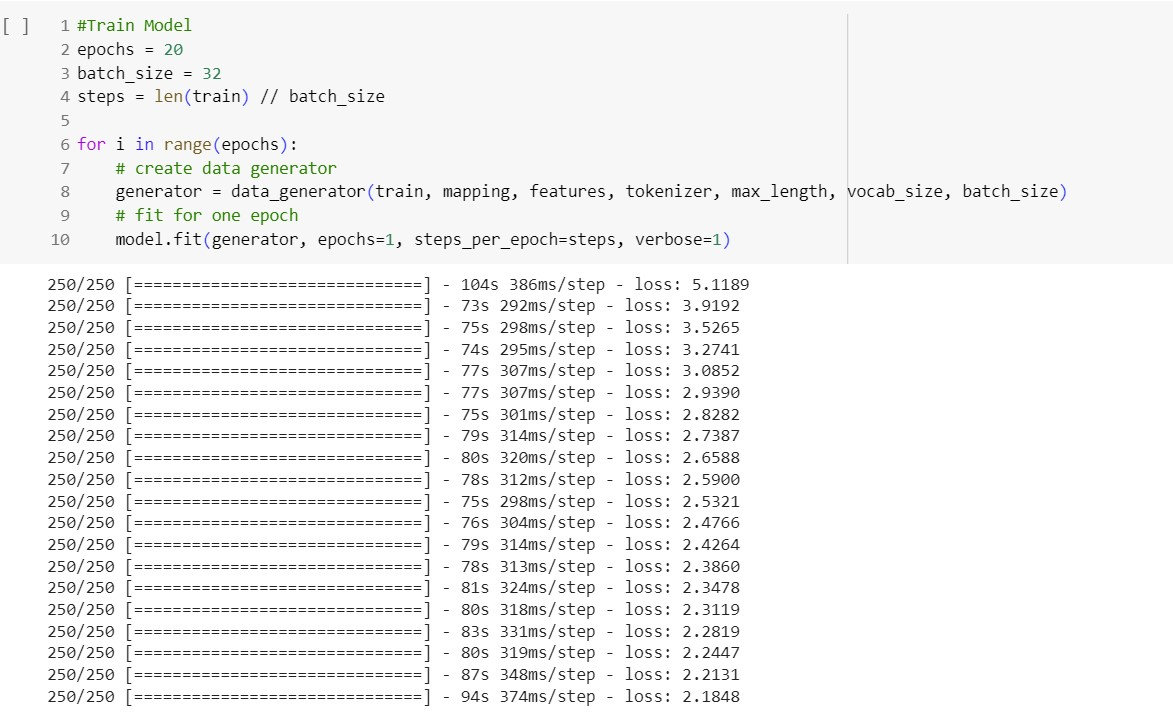


Implemented CNN + LSTM model to generate captions:



**Training**

For Training the model, we used a data generator function. The function was fed with features generated using VGG16, tokenized text, mapping of images & captions from flicker\_8k dataset. To avoid crashing of model training sessions, the mapping data was split into 32 batches and iterated over 20 epochs.



**BLUE-Scores from trained models**

|  |  |
| --- | --- |
| VGG16 + CNN + LSTM | Inception v3 + CNN + LSTM |

Deployment

Once the model is ready, we tested caption generation in google collab and then deployed the same model using FastAPI-Uvicorn in local PC server.

**Results**

Results of Captions Generated in **training & testing** phase:



Results of Captions Generated in **Deployment** phase:

|  |
| --- |
|  |

**“Caption Generated” using different models**

|  |  |  |
| --- | --- | --- |
|  |  |  |

Learnings

* Overall, Experienced as Core Developer during the capstone project phase, after a long time.
* How to generate feature vectors from the images using CNN.
* How to feed both feature vectors and tokens to an LSTM network.
* GIT, VSCode, JupyterLab, Python Logging, and setting up the development environment in local PC.
* All learning earlier during the course was in Google Collab & TalentSprint-LMS.
* Literature Research, refer to internet content and understand the available/implemented code to design our project model.
* In the training, used “import logging” library to write debug log messages to a file
* Setup development Environment in local PC – reduce dependency on Google Collab (using only to refer course material)
* Used requirements.txt to overcome setup issues between team members.
* Various challenges in the installation of packages and imports
* long paths, AttributeError: module 'collections' has no attribute 'Callable’ (modify ‘py3k\_compat.py ’)
* FastAPI deployment and Basics of HTML & templates.
* Use the trained model (“.h5” and “.pkl”) in Python code on the local PC
* How to output images in the deployment
* User Interface to select an image to generate captions.
* Used GitHub initially to select images and then updated to select any of the 8k images saved on the local PC.
* Collaborate and Archive progressive development in GitHub.
* Started training team members @ workplace – KEY is The Confidence Gained in Capstone Project
* Tested different models (ResNet18 & InceptionV3) and encountered challenges and lessons learnt.

Challenges

Here are the challenges encountered in all phases of capstone project duration,

* Encountered issues in training the base model on the entire flickr8K dataset.
* Used requirements.txt to overcome setup issues between team.
* Various challenges in the installation of packages and imports
* Google Collab session was crashing frequently.
* Shifted development environment to iPython/ VS4 to resolve the above.
* Caption Generation was the most complex part.
* Some captions were empty after initial text cleaning.
* Used Python Logging libs to debug and fine tune Caption.
* There were some missing images corresponding to captions present in the tokens.
* After a keen look at the logging and images vs captions, we found missing images and corrected/removed the related captions.

Real-world application

Automatic Image Captioning is playing an essential role in today's AI-Centric world. Some examples listed below:

* **Accessibility:** It can help people with low or no eyesight in identify images by providing them with textual descriptions of images. This enables them to understand and engage with visual content on the internet or in digital media.
* **Virtual Assistants:** Image Captioning can be integrated into virtual assistants such as smart glasses or camera-based systems, to provide real-time descriptions of the user's surroundings. It can also help individuals with visual impairments in navigating unfamiliar environments or recognizing objects.
* **Image Indexing:** It can be used to index and retrieve images based on their content. By generating captions for images, it becomes easier to search for specific images using text-based queries, improving the efficiency of image retrieval systems.
* **User Experience on Web:** Image captioning can enhance user experiences on social media platforms and e-commerce websites. Captions can be automatically generated for user-uploaded images, making them more accessible and understandable. This allows for improved search functionality and helps users find relevant content more effectively.
* **Medical Analysis:** It can be valuable in medical imaging analysis. By generating captions for medical images, it becomes easier for healthcare professionals to review and interpret images, improving diagnosis, treatment planning, and medical research.
* **Surveillance and Security:** In security systems, image captioning can be used to automatically analyse and interpret images or video footage, enabling the detection of specific objects, events, or anomalies. This enhances surveillance capabilities and facilitates the monitoring of large amounts of visual data.

Conclusion

We successfully trained our model using Flickr8k dataset to identify and generate caption for images based on the various objects depicted in it by mapping the regions as well.

Next Steps/Opportunities

1. Train model using **30K Flicker dataset** and generate better captions.
2. **Web deployment** using back4app.com & render.com
3. Learn **from different implementations** (Cascaded LSTM/Transformers)

References

[1] Marc Tanti and AlbertGatt, where to put the Image in an Image CaptionGenerator, arXiv:1703.09137, 2018

[2] Marc Tanti Albert Gatt Kenneth P. Camilleri What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator? arXiv:1708.02043, 2017

[3] Andrej Karpathy and Li Fei-Fei, Deep Visual-Semantic Alignments for Generating Image Descriptions, <https://cs.stanford.edu/people/karpathy/cvpr2015.pdf>

[4] Abhijit Roy, A Guide to Image Captioning, <https://towardsdatascience.com/a-guide-to-image-captioning-e9fd5517f350>

[5] Image Captioning <https://paperswithcode.com/task/image-captioning>

[6] <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00571-w>