Malnad College of Engineering, Hassan

(An Autonomous Institution affiliated to VTU, Belgavi)

A Main Project Report

On

"Visual-Attention based Image Captioning"

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

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by

Omana Prabhakar	4MC19CS095
Rahul Jain H V	4MC19CS117
Rohan K	4MC19CS125
Sthuthi B Iyer	4MC19CS159

Under the guidance of
Mrs. Kavyasri M N
Assistant Professor

Department of Computer Science and Engineering

2022-2023

Malnad College of Engineering

Department of Computer Science and Engineering
Hassan - 573201, Karnataka, India

Certificate

This is to certify that main project work entitled **“Visual-Attention based Image Captioning”** is a bonafide work carried out by **Sthuthi B Iyer (4MC19CS159), Omana Prabhakar (4MC19CS095),Rahul Jain H V(4MC19CS117), Rohan K (4MC19CS125)** in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belgavi during the year 2022-2023. The project report has been approved as it satisfies the academic requirements in respect of main project work prescribed for the Bachelor of Engineering Degree.

Signature of the Guide	Signature of the HOD	Signature of the Principal
Mrs.Kavyasri M N	Dr. Geetha Kiran A	Dr. S. Pradeep
Assistant Professor	Prof. & HOD	Principal
Dept. of CSE, MCE	Dept. of CSE, MCE	MCE

Examiners

Name of the Examiner	Signature of the Examiner
1.	
2.	

ABSTRACT

For visually impaired individuals, understanding the content and context of images can be challenging or impossible without proper descriptions. Image captioning bridges this gap by providing textual descriptions of images, enabling visually impaired individuals to gain a better understanding of the visual content being depicted. With the help of image captioning, visually impaired individuals can access and enjoy various forms of visual media, including social media posts, news articles, educational materials, and entertainment content. Image captions provide them with a detailed account of the visual elements present in the images, allowing them to participate more fully in the digital world.

The core idea behind visual attention is inspired by human visual perception. When humans observe an image, they naturally attend to specific regions or objects that are most informative or salient for understanding the scene. Visual attention mechanisms in image captioning aim to replicate this process by enabling the model to dynamically attend to different image regions during the caption generation process. Considering these aspects, we propose a novel approach that combines visual attention-based image captioning with speech output generation, enabling the creation of image descriptions in both written and spoken forms. Our method utilizes convolutional neural networks (CNNs) to extract visual features from the input image, and employs a long short-term memory (LSTM) network with attention mechanisms to generate descriptive captions. Additionally, we incorporate a text-to-speech synthesis system to convert the generated captions into speech.

The visual attention mechanism plays a crucial role in our proposed system, allowing the model to focus on relevant regions of the image while generating captions. By attending to different parts of the image, the model can better align the visual content with the corresponding textual descriptions. This attention-based approach enhances the quality and relevance of the generated captions, resulting in more accurate and coherent image descriptions.

This work presents a novel approach for visual attention-based image captioning with speech output. By integrating visual attention mechanisms and text-to-speech synthesis, our method addresses the limitations of traditional image captioning systems, making them more inclusive and accessible to a wider range of users. The proposed approach has the potential to enhance the user experience in various applications, such as assistive technologies, image understanding, and multimedia content accessibility.

ACKNOWLEDGEMENTS

We present with immense pleasure this work titled **"Visual-Attention based Image Captioning"**. An endeavour over a long period can be successful with the advice and support of many well wishers.

We take this opportunity to express our gratitude and appreciation to all of them. The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without mentioning the people who made it possible. So, with gratitude we acknowledge all those whose guidance and encouragement made to successfully complete this project.

We would like to express sincere thanks to our Principal **Dr. S Pradeep** , Malnad College of Engineering for his encouragement made to successfully completion of the project work. We wish to express our gratitude to **Dr.Geetha Kiran A**, Professor and Head, Department of Computer Science & Engineering for providing a good working environment and for her constant support and encouragement.

It gives great pleasure in placing on record a deep sense of gratitude to our guide **Kavyasri M N**,Assistant Professor, Department of Computer Science & Engineering for her daily evaluation of the work and for providing us constant encouragement with her unflinching support and valuable guidance throughout this project.

We would also like to thank all the staff of Computer Science and Engineering department who have directly or indirectly helped us in the completion of the project work. At last we would hereby acknowledge and thank our parents who have been a source of inspiration and also instrumental in the successful completion of the project work.

Sthuthi B Iyer

Omana Prabhakar

Rahul Jain H V

Rohan K

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

Visual attention-based image captioning is a fascinating research area that combines computer vision and natural language processing to automatically generate descriptive captions for images.

1.1 Introduction to Visual attention-based image captioning

The goal of image captioning is to develop algorithms that can comprehend the visual content of an image and generate human-like textual descriptions that accurately convey the image's essence. This technology has numerous applications, including content retrieval, image understanding, and enhancing the accessibility of visual information.

Traditional image captioning methods often rely solely on deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to process images and generate captions. These models attempt to learn the underlying relationships between image features and corresponding textual descriptions. However, they often fail to capture fine-grained details or accurately align the visual content with the generated captions.

To address these limitations, visual attention mechanisms have been introduced in image captioning. Inspired by the human visual system, attention mechanisms enable the model to focus on specific regions of the image while generating captions. This selective attention allows the model to align the relevant image regions with the corresponding words in the generated captions, resulting in more accurate and contextually relevant descriptions.

While visual attention has greatly improved the quality of image captioning, there is still a crucial aspect missing in many existing systems–speech output. Most image captioning models generate captions in written form, which poses accessibility challenges for individuals with visual impairments. To make image captioning more inclusive and accessible, it is essential to integrate speech output generation into the captioning process.

1.2 About Project

1.2.1 Problem Statement

The existing image captioning methods predominantly focus on generating written captions, neglecting the needs of visually impaired individuals who rely on auditory information for understanding visual content. This creates a significant barrier for inclusivity and restricts the potential applications of image captioning in assistive technologies and multimedia accessibility.

1.2.2 Objectives

The objectives of the work are:

- Develop a deep learning architecture that can effectively extract features from images using the CNN and generate natural language captions using LSTM RNNs.
- Transform the text captions into spoken words.
- Evaluate the performance of the model to ensure that the generated captions are semantically accurate and stylistically appropriate.

Chapter 2

Literature Survey

Survey of existing systems are taken into consideration which are similar to the current work:

In the method put forward by Liu, Shuang Bai, Liang Hu, Yanli Wang, Haoran et al. [1], two models of deep learning namely, Convolutional Neural Network-Recurrent Neural Network(CNN-RNN) Based Image Captioning, Convolutional Neural Network-Convolutional Neural (CNNCNN) Based Image Captioning. In CNN-RNN Based frame work, Convolutional Neural Networks for encoding and Recurrent Neural Networks for the decoding process. Using CNN the images here are converted to vectors and these vectors are called image features these are passed into Recurrent neural networks as input. In RNN, NLTK libraries are used to get the actual captions for the project. In the CNN-CNN based frame work only CNN is used for both encoding and decoding of the images. Here vocab dictionary is used and it is mapped with Image features to get the exact word for the given image using NLTK library. Thus generating the error free caption. Consisting of many models that are given at the same time of convolution techniques simultaneously is certainly quicker compared to the train the continuous flowing recurrently repetition of this techniques. CNN-CNN Model has less training time as compared to the CNN-RNN Model. The CNN-RNN Model has more training time as it is sequential but it has less loss compared to the CNN-CNN Model.

In the method presented by Ansari Hani et al [2], Here they have used encoding decoding model for image captioning. Here they have mentioned two more models for image captioning they are: Retrieval based captioning and template based captioning. Retrieval based captioning is the process where training images are placed in one space and their corresponding captions which are generated are placed in another scope now in the new scope the correlations are calculated for the test image and captions the highest valued correlation caption is retrieved as caption for the given image from the given set of captions dictionary. Prototype based descripting is the technique is done by them in this paper .Here they have used Inception V3 model as their encoder and they have used attention mechanism.

In the method outlined by Subrata Das, Lalit Jain et al [3] This model is mainly based on how the deep learning models are used for Military Image captioning. It mainly uses CNN-RNN based frame work.They have used Inception model for encoding the images and to decrease the gradient descent problem they have used Long Short Term Memory (LSTM'S) Networks.

Aishwarya Maroju, Sneha Sri Doma, Lahari Chandarlapati, 2022. [4], Here they have used encoding decoding model for image captioning. Here they have mentioned two more models for image captioning they are: Retrieval based captioning and template based captioning. In Resnet-LSTM model for the image captioning process., Resnet Architecture is used for encoding and LSTM's are used for decoding. Advantages: Resnet-LSTM model Reduces the vanishing gradient problem in CNN. ResNet decreases the loss of input features compared to CNN. Using the concept of transfer learning we are reducing the computation cost and training time and increasing the accuracy of the model. .

In the method introduced by Ankush Yadav,Aman Singh,Aniket Sharma,Ankur Sindhu,Umang Rastogi, [5] This Text-to-Speech (TTS) allows users to convert written text into spoken words, which is useful in a wide range of applications such as accessibility for visually impaired users, and language learning. Pyttsx3 in Python is a wrapper for the Microsoft Speech API (SAPI) text-to-speech engines and eSpeak, which provide high-quality speech synthesis capabilities. Pyttsx3 is easy to use and provides a simple interface for controlling speech output, including pitch, volume, and rate.

Chapter 3

Project Design

The project design includes CNN and LSTM models for image captioning process. CNN model is used for encoding and LSTM model is used for decoding. The design of the work is classified into several steps.

3.1 System architecture

The high level design architecture of the Visual-Attention based Image Captioning can be summarised as shown below:

The project aims to develop a Visual-Attention based Image Captioning system using the Resnet,CNN and LSTM models. The system will take an input image and generate a descriptive caption that accurately represents the contents of the image. The CNN model will be used to extract relevant visual features from the image. These features will then be fed into an attention mechanism, which will focus on the most salient regions of the image. A language model, such as a Long Short Term Memory(LSTM), will generate the caption based on the attended visual features. The model will be trained on a large dataset of images and the mapping is done between visual features and the index of an image.

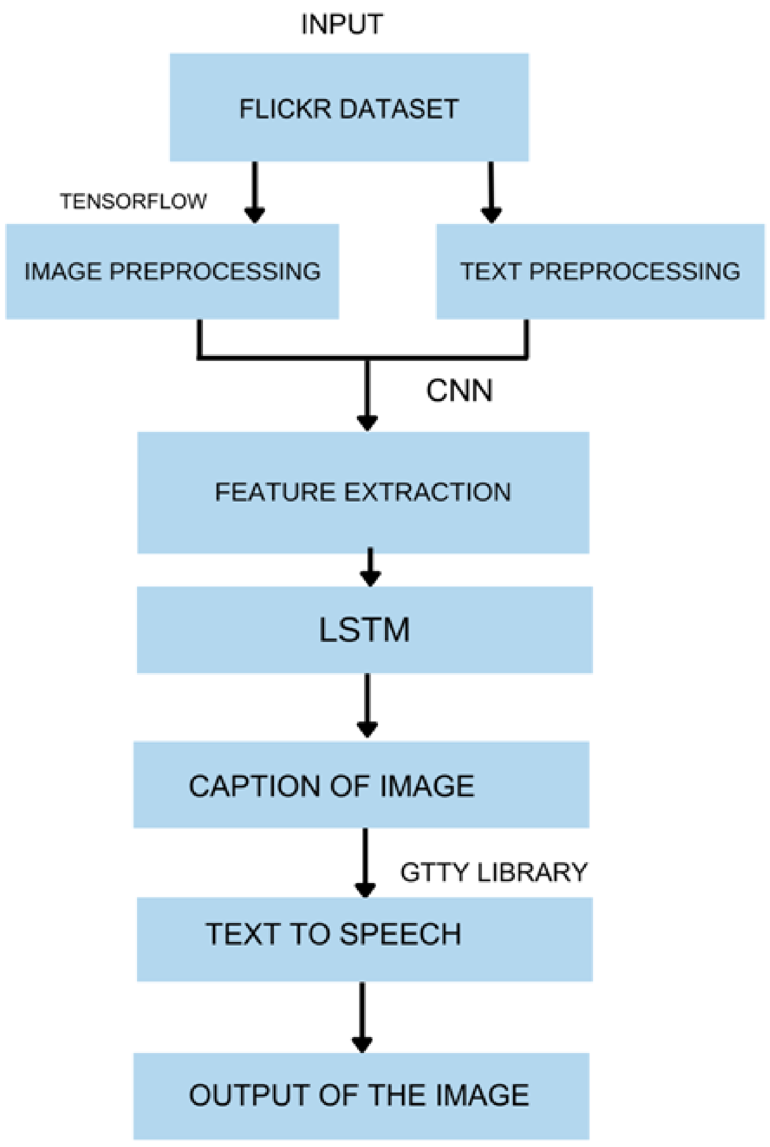


Figure 3.1: Block Diagram

Chapter 4

Implementation

To implement Visual-Attention based Image Captioning,the work is classified into four modules:

Image preprocessing:
Image preprocessing is a crucial step in image captioning that involves preparing the input image for further analysis and feature extraction. The goal is to transform the raw image data into a format that can be effectively processed by the image captioning model. The image preprocessing techniques used are:

- Resizing: The input image is often resized to a fixed resolution to ensure consistency and to match the requirements of the image captioning model. Resizing helps in reducing computational complexity and standardizing the input image size.

- Normalization: Image normalization is performed to standardize the pixel values of the image. It involves scaling the pixel values to a specific range, typically between 0 and 1 or -1 and 1. Normalization helps in reducing the impact of lighting conditions, color variations, and improves the convergence of the neural network during training.

- Cropping and Padding: Depending on the requirements of the image captioning model, images may need to be cropped or padded to a specific aspect ratio or size. Cropping involves removing parts of the image to focus on the most relevant content, while padding adds extra pixels to ensure the image has the desired dimensions.

- Color Space Conversion: In some cases, converting the image to a different color space can be beneficial. For example, converting RGB images to grayscale can reduce the computational complexity and eliminate color-related variations that might not be essential for image captioning tasks.

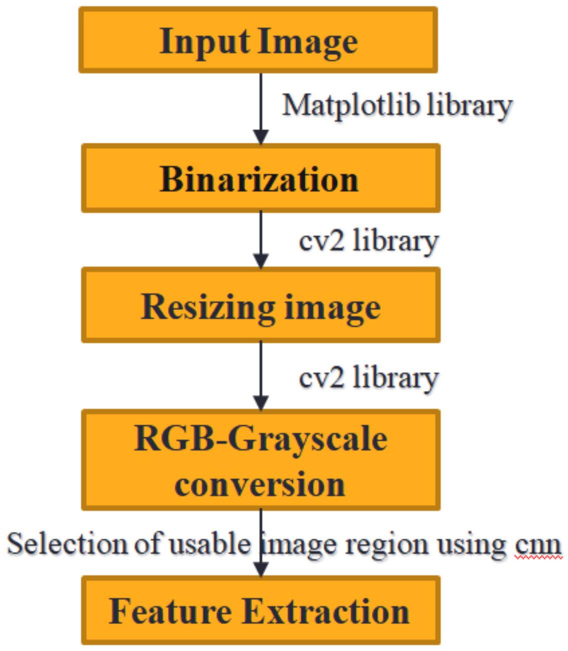


Figure 4.1: block diagram of image pre-processing

Text preprocessing:
After loading the captions for the images using FLICKR text data set we need to preprocess those captions so that there is no ambiguity or difficulty while generating vocabulary from the captions and also while training the deep learning model.We need to check whether the captions contain any numbers if found they must be removed and after that we need to remove white spaces and also missing captions in the given data set.We need to change all the upper case letters in the captions to the lower case in order to eliminate ambiguity during vocabulary building and training of the model. As this model will generate captions one word at a time and previously generated words are used as inputs along with the image features as input ,¡start seq¿ and ¡end seq¿ are attached at the starting and end of each of the caption to signal the neural

network about the starting of the caption and ending of the captions during the training and testing of the model.

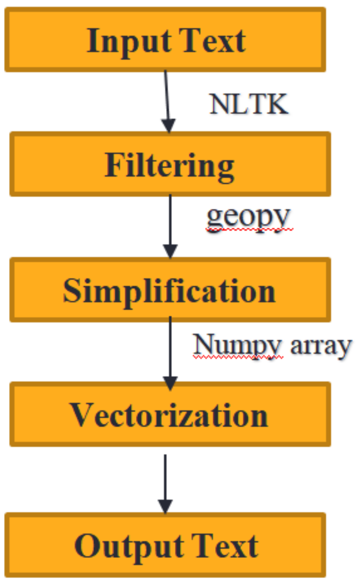


Figure 4.2: block diagram of text pre-processing

Feature extraction using CNN

CNNs are specifically designed to extract hierarchical and abstract representations from images, enabling the model to capture and understand the visual features of an image. The layers used are:

- Convolutional Layers: CNNs typically consist of multiple convolutional layers. Each layer performs convolution operations on the input image using a set of learnable filters or kernels. Convolution involves sliding the filters over the image and computing dot products between the filter weights and the corresponding image patches. This operation captures local patterns and spatial relationships between pixels, producing feature maps that highlight important visual features.

- Pooling Layers: Pooling layers are often inserted between convolutional layers to downsample the feature maps and reduce their spatial dimensions. Max pooling is a commonly used pooling technique where the maximum value within a pooling window is selected to represent a region. Pooling helps in achieving translation invariance and reducing the computational complexity of subsequent layers.

- Fully Connected Layers: Fully connected layers, also known as dense layers, connect every neuron in one layer to every neuron in the next layer. These layers aggregate the local features learned by the previous layers into higher-level representations. Fully connected layers are typically used towards the end of the CNN architecture to perform classification or regression tasks.

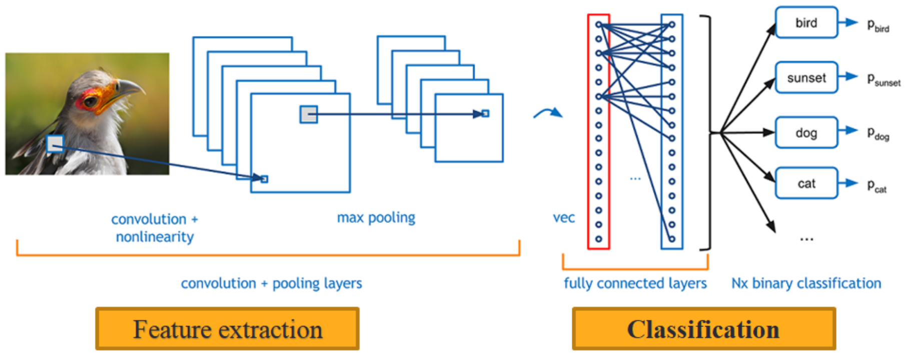


Figure 4.3: Working of CNN

Generation of caption using LSTM architecture

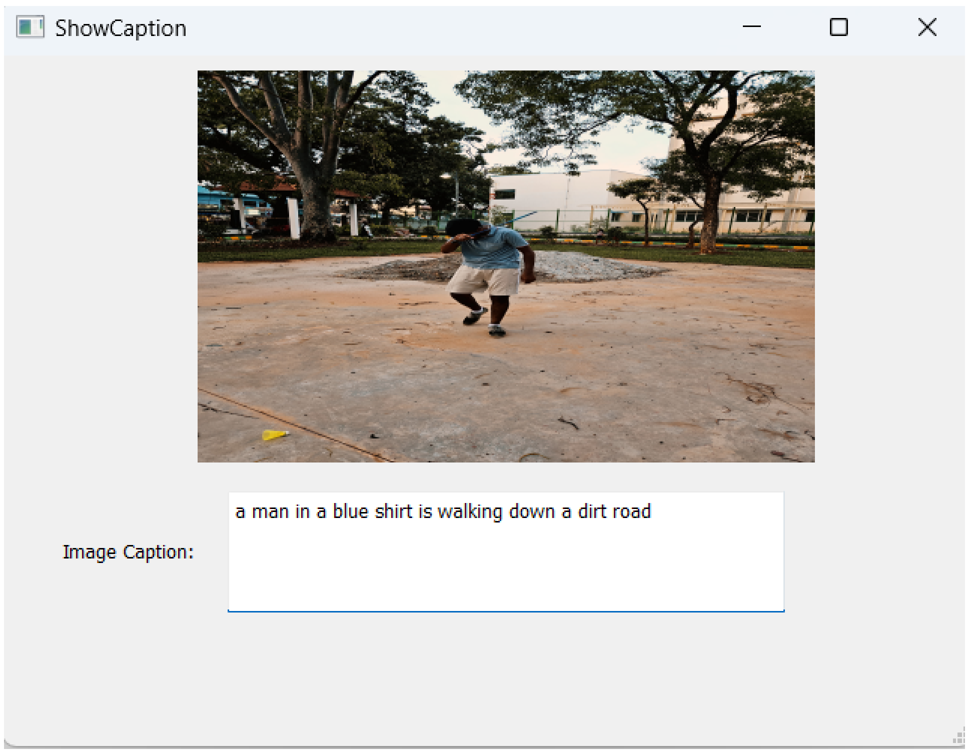
Preprocess the input image and convert it into a fixed-size representation, such as a feature vector, using a CNN. Initialize the LSTM's hidden state and cell state using the feature vector.Start with a special start-of-sequence token as the input to the LSTM.Iterate over the LSTM until an end-of-sequence token is generated or a maximum caption length is reached, generating one word at each time step based on the previous word and LSTM's hidden state.Output the generated caption.

Chapter 5

Results

Final output for visual attention-based image captioning using the ResNet LSTM is seen by giving the input image into the ResNet model for feature extraction. The visual features are then combined with textual context using an RNN with an attention mechanism. The model attends to different regions of the image while generating each word of the caption, resulting in a descriptive and contextually relevant caption.

The actual output may vary depending on the specific implementation, training data, and the complexity of the image being captioned.



a man in a blue shirt is walking down a dirt road

Figure 5.1: Output 1

It is observed that during the initial epochs of training, the accuracy is very low and the captions generated are not much related to the given test images. If we train the model for atleast 20 epochs then we have observed that the captions generated are some what related to the given test images.

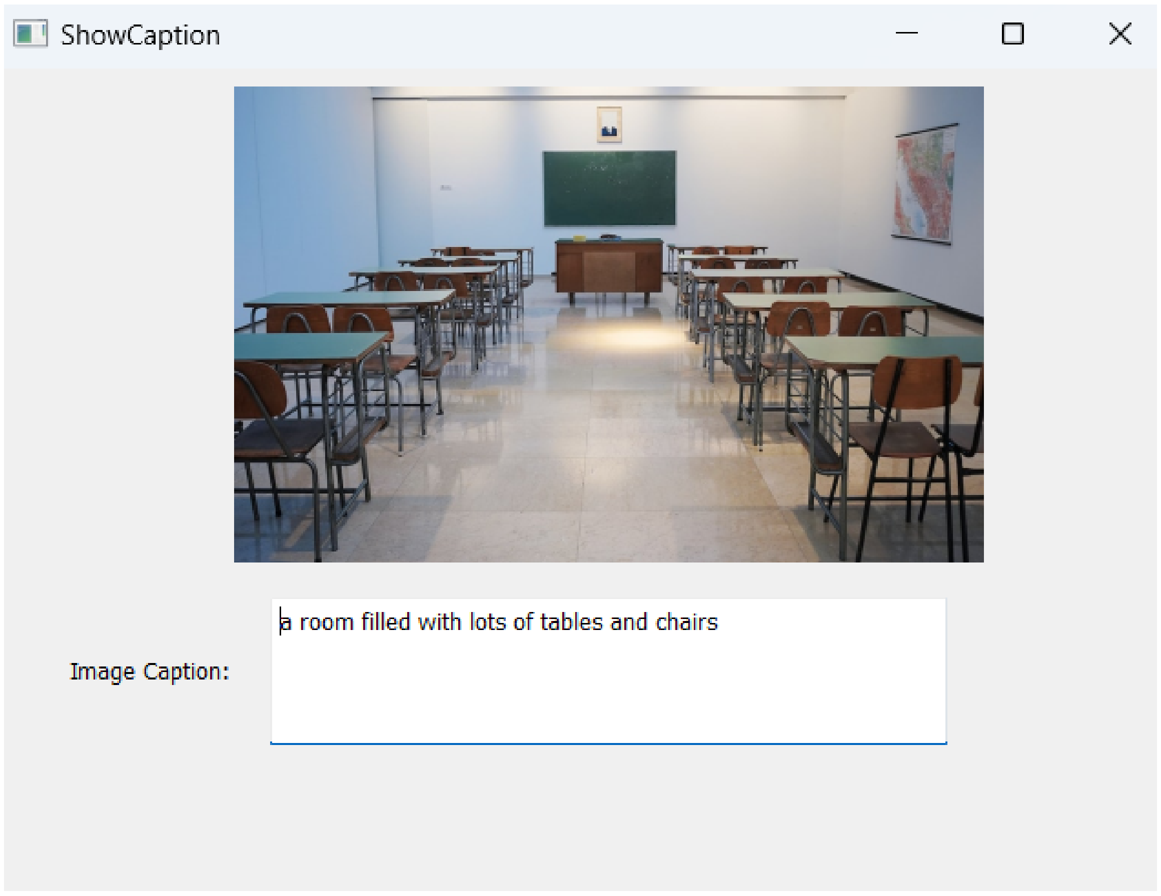


Image Caption: a room filled with lots of tables and chairs

Figure 5.2: Output 2

Evaluation :

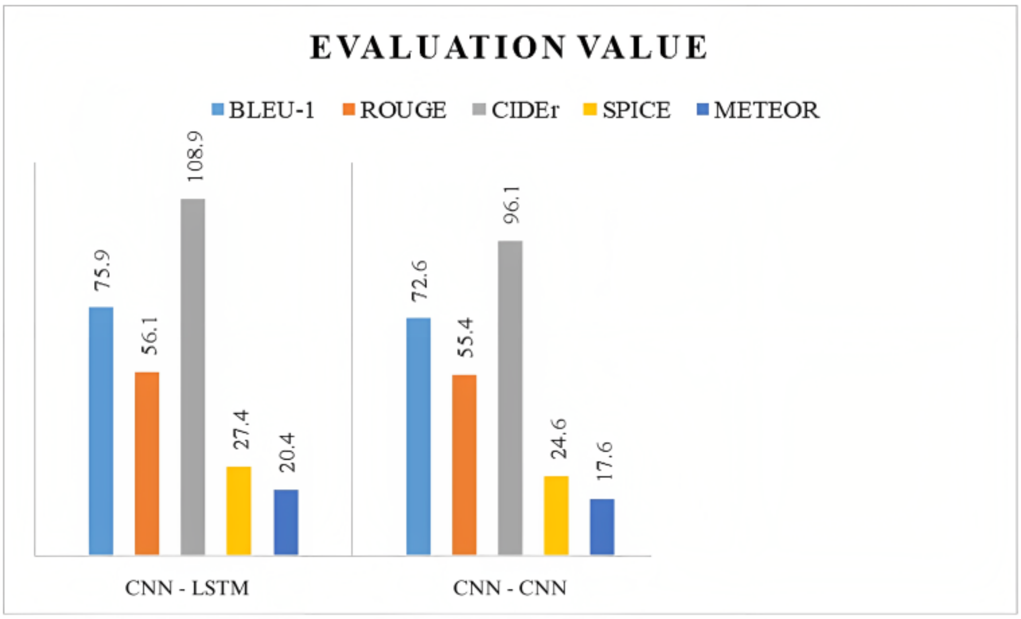


Figure 5.3: Graph

The best results of the above three methods for five evaluation metrics are shown. The work resulted that both the CNN-RNN based and the Reinforcement based methods can get the better performance than the CNN-CNN based framework, which greatly improves the training speed without seriously affecting the accuracy.

Chapter 6

Conclusion

Visual attention-based image captioning using the ResNet model is a powerful technique that combines convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to generate descriptive captions for images. The ResNet model is used to extract visual features from the input image, and these features are combined with textual context using an RNN with an attention mechanism.

By incorporating visual attention, the model can dynamically focus on different regions of the image while generating each word of the caption. This attention mechanism allows the model to capture relevant details and generate more accurate and contextually appropriate captions.

Visual attention-based image captioning using ResNet model has shown promising results in generating captions that are descriptive and meaningful, capturing both the visual content of the image and the textual context. It has numerous applications in areas such as image understanding, assistive technology, and content generation in various domains.

APPENDIX A

Python Program

The following is the code used to extract the author names and write the result into a file.

```
import torch
from torch import nn
import torchvision

device = torch.device("cuda" if torch.cuda.is_available()
else "cpu")
class Encoder(nn.Module):
    """
    Encoder.
    """

    def __init__(self, encoded_image_size=14):
        super(Encoder, self).__init__()
        self.enc_image_size = encoded_image_size

        resnet = torchvision.models.resnet101(pretrained=True)
        # pretrained ImageNet ResNet-101

        # Remove linear and pool layers (since we’re not
        doing classification)
        modules = list(resnet.children())[:-2]
        self.resnet = nn.Sequential(*modules)
        # Resize image to fixed size to allow input images of
        variable size
        self.adaptive_pool = nn.AdaptiveAvgPool2d(
        (encoded_image_size, encoded_image_size))

        self.fine_tune()

    def forward(self, images):
        """
        Forward propagation.

        :param images: images, a tensor of dimensions (batch_size,
        3, image_size, image_size)
        :return: encoded images
        """
        out = self.resnet(images)  # (batch_size, 2048,
        image_size/32, image_size/32)
        out = self.adaptive_pool(out)
        # (batch_size, 2048, encoded_image_size,
        encoded_image_size)
        out = out.permute(0, 2, 3, 1)
        # (batch_size, encoded_image_size,
        encoded_image_size, 2048)
        return out

    def fine_tune(self, fine_tune=True):
        """
        Allow or prevent the computation of gradients for
        convolutional blocks 2 through 4 of the encoder.

        :param fine_tune: Allow?
        """
        for p in self.resnet.parameters():
            p.requires_grad = False
        # If fine-tuning, only fine-tune convolutional blocks
        2 through 4
        for c in list(self.resnet.children())[5:]:
            for p in c.parameters():
                p.requires_grad = fine_tune


class Attention(nn.Module):
    """
    Attention Network.
    """

    def __init__(self, encoder_dim, decoder_dim, attention_dim):
        """
        :param encoder_dim: feature size of encoded images
        :param decoder_dim: size of decoder's RNN
        :param attention_dim: size of the attention network
        """
        super(Attention, self).__init__()
        self.encoder_att = nn.Linear(encoder_dim, attention_dim)
        # linear layer to transform encoded image
        self.decoder_att = nn.Linear(decoder_dim, attention_dim)
        # linear layer to transform decoder's output
        self.full_att = nn.Linear(attention_dim, 1)
        # linear layer to calculate values to be softmax-ed
        self.relu = nn.ReLU()
        self.softmax = nn.Softmax(dim=1)
        # softmax layer to calculate weights

    def forward(self, encoder_out, decoder_hidden):
        """
        Forward propagation.

        :param encoder_out: encoded images, a tensor of
        dimension (batch_size, num_pixels, encoder_dim)
        :param decoder_hidden: previous decoder output,
        a tensor of dimension (batch_size, decoder_dim)
        :return: attention weighted encoding, weights
        """
        att1 = self.encoder_att(encoder_out)
        # (batch_size, num_pixels, attention_dim)
        att2 = self.decoder_att(decoder_hidden)
        # (batch_size, attention_dim)
        att = self.full_att(self.relu(att1 +
        att2.unsqueeze(1))).squeeze(2)
        # (batch_size, num_pixels)
        alpha = self.softmax(att)
        # (batch_size, num_pixels)
        attention_weighted_encoding = (encoder_out *
        alpha.unsqueeze(2)).sum(dim=1)
        # (batch_size, encoder_dim)

        return attention_weighted_encoding, alpha


class DecoderWithAttention(nn.Module):
    """
    Decoder.
    """

    def __init__(self, attention_dim, embed_dim, decoder_dim,
    vocab_size, encoder_dim=2048, dropout=0.5):
        """
        :param attention_dim: size of attention network
```

```
:param embed_dim: embedding size
:param decoder_dim: size of decoder's RNN
:param vocab_size: size of vocabulary
:param encoder_dim: feature size of encoded images
:param dropout: dropout
"""
super(DecoderWithAttention, self).__init__()

self.encoder_dim = encoder_dim
self.attention_dim = attention_dim
self.embed_dim = embed_dim
self.decoder_dim = decoder_dim
self.vocab_size = vocab_size
self.dropout = dropout

self.attention = Attention(encoder_dim, decoder_dim,
attention_dim)  # attention network

self.embedding = nn.Embedding(vocab_size, embed_dim)
# embedding layer
self.dropout = nn.Dropout(p=self.dropout)
self.decode_step = nn.LSTMCell(embed_dim + encoder_dim,
decoder_dim, bias=True)  # decoding LSTMCell
self.init_h = nn.Linear(encoder_dim, decoder_dim)
# linear layer to find initial hidden state of LSTMCell
self.init_c = nn.Linear(encoder_dim, decoder_dim)
# linear layer to find initial cell state of LSTMCell
self.f_beta = nn.Linear(decoder_dim, encoder_dim)
# linear layer to create a sigmoid-activated gate
self.sigmoid = nn.Sigmoid()
self.fc = nn.Linear(decoder_dim, vocab_size)
# linear layer to find scores over vocabulary
self.init_weights()
```

```
        # initialize some layers with the uniform distribution

    def init_weights(self):
        """
        Initializes some parameters with values from the
        uniform distribution, for easier convergence.
        """
        self.embedding.weight.data.uniform_(-0.1, 0.1)
        self.fc.bias.data.fill_(0)
        self.fc.weight.data.uniform_(-0.1, 0.1)

    def load_pretrained_embeddings(self, embeddings):
        """
        Loads embedding layer with pre-trained embeddings.

        :param embeddings: pre-trained embeddings
        """
        self.embedding.weight = nn.Parameter(embeddings)

    def fine_tune_embeddings(self, fine_tune=True):
        """
        Allow fine-tuning of embedding layer?
        (Only makes sense to not-allow if using pre-trained
        embeddings).

        :param fine_tune: Allow?
        """
        for p in self.embedding.parameters():
            p.requires_grad = fine_tune

    def init_hidden_state(self, encoder_out):
        """
        Creates the initial hidden and cell states for
        the decoder's LSTM based on the encoded images.

        :param encoder_out: encoded images, a tensor of
        dimension (batch_size, num_pixels, encoder_dim)
        :return: hidden state, cell state
        """
        mean_encoder_out = encoder_out.mean(dim=1)
        h = self.init_h(mean_encoder_out)
        # (batch_size, decoder_dim)
        c = self.init_c(mean_encoder_out)
        return h, c

    def forward(self, encoder_out, encoded_captions,
    caption_lengths):
        """
        Forward propagation.

        :param encoder_out: encoded images, a tensor of
        dimension
        (batch_size, enc_image_size, enc_image_size,
        encoder_dim)
        :param encoded_captions: encoded captions, a
        tensor of
        dimension (batch_size, max_caption_length)
        :param caption_lengths: caption lengths, a
        tensor of
        dimension (batch_size, 1)
        :return: scores for vocabulary, sorted encoded
        captions,
        decode lengths, weights, sort indices
        """

        batch_size = encoder_out.size(0)
```

```
encoder_dim = encoder_out.size(-1)
vocab_size = self.vocab_size

# Flatten image
encoder_out = encoder_out.view(batch_size, -1,
encoder_dim)  # (batch_size, num_pixels, encoder_dim)
num_pixels = encoder_out.size(1)

# Sort input data by decreasing lengths; why?
apparent below
caption_lengths, sort_ind =
caption_lengths.squeeze(1).sort(dim=0, descending=True)
encoder_out = encoder_out[sort_ind]
encoded_captions = encoded_captions[sort_ind]

# Embedding
embeddings = self.embedding(encoded_captions)
# (batch_size, max_caption_length, embed_dim)

# Initialize LSTM state
h, c = self.init_hidden_state(encoder_out)
# (batch_size, decoder_dim)

# We won’t decode at the <end> position,
since we’ve finished generating as soon as we
generate <end>
# So, decoding lengths are actual lengths - 1
decode_lengths = (caption_lengths - 1).tolist()

# Create tensors to hold word predicion scores and
alphas

predictions = torch.zeros(batch_size,
max(decode_lengths),
vocab_size).to(device)
alphas = torch.zeros(batch_size,
max(decode_lengths),
num_pixels).to(device)

# At each time-step, decode by
# attention-weighing the encoder’s output based on the
decoder’s previous hidden state output
# then generate a new word in the decoder with the
previous word and the attention weighted encoding
for t in range(max(decode_lengths)):

    batch_size_t = sum([l > t for l in decode_lengths])

    attention_weighted_encoding, alpha =
    self.attention(encoder_out[:batch_size_t],
    h[:batch_size_t])
    gate = self.sigmoid(self.f_beta(h[:batch_size_t]))
    # gating scalar, (batch_size_t, encoder_dim)

    attention_weighted_encoding = gate *
    attention_weighted_encoding
    h, c = self.decode_step(

        torch.cat([embeddings[:batch_size_t, t, :],
        attention_weighted_encoding], dim=1),

        (h[:batch_size_t], c[:batch_size_t]))
        # (batch_size_t,
        decoder_dim)
    preds = self.fc(self.dropout(h))  # (batch_size_t,
    vocab_size)
```

```
            predictions[:batch_size_t, t, :] = preds
            alphas[:batch_size_t, t, :] = alpha

        return predictions, encoded_captions, decode_lengths,
        alphas, sort_ind
```

References

[1] Liang Hu Yanli Wang Haoran Liu, Shuang Bai. *Image Captioning Based on Deep Neural Networks.* MATEC Web of Conferences. 232. 01052. 10.1051/matecconf/201823201052., 2018.

[2] N. Tagougui A. Hani and M. Kherallah. *Image Caption Generation Using A Deep Architecture.* International Arab Conference on Information Technology (ACIT), 2019.

[3] L. Jain S. Das and A. Das. *Deep Learning for Military Image Captioning.* 21st International Conference on Information Fusion (FUSION),, 2018.

[4] Lahari Chandarlapati Aishwarya Maroju, Sneha Sri Doma. *Image Caption Generating Deep Learning Model.* 21st International Conference on Information Fusion (FUSION),, 2022.

[5] Aniket Sharma Ankur Sindhu Umang Rastogi Ankush Yadav, Aman Singh. *Voice Assistance for Visually Impaired.* 21st International Conference on Information Fusion (FUSION),, 2020.