Deep Learning Term Project

Team AI4ALL

Team Members:
Gautam Kumar (21CS30020)
Deepraj Das (21CS30017)
I. Sai Hari Krishna(21CS30025)
Kovvada Dhyana Vardhan(21CS30030

Contents

1 Introduction

Image captioning, the process of generating textual descriptions for images, is a fascinating interdisciplinary research area at the intersection of computer vision and natural language processing (NLP). It plays a crucial role in various applications such as assistive technologies, content-based image retrieval, and enhancing user experiences on social media platforms.

In this project, we aim to design a transformer-based encoder-decoder model for image captioning. We will utilize the Vision Transformer (ViT) as the image encoder and a text decoder of our choice. The ViT architecture has shown remarkable performance in various computer vision tasks by leveraging self-attention mechanisms to capture global dependencies in images efficiently. By incorporating ViT as the image encoder, we expect to extract meaningful visual features that can facilitate accurate and contextually relevant caption generation.

To ensure fair evaluation, we will adhere to certain constraints: the model's size should not exceed 15GB of GPU space, which is the limit on the T4 GPU available on platforms like Google Colab or Kaggle. Additionally, we will not utilize end-to-end trained models available on external repositories; instead, we will design and train our model from scratch, adhering to the given dataset for training and evaluation.

The project will be divided into two parts: Part A involves designing and training the encoder-decoder model, while Part B focuses on fine-tuning pre-trained components and conducting thorough evaluations.
Part A: Encoder Decoder Model from Scratch In Part A, we will follow a systematic approach:

- Data preprocessing to parse and format the given dataset.
- Model creation, where we design the architecture of the encoder-decoder model.
- Model training using the provided training set.
- Evaluation of the trained model on the test set using specified evaluation metrics, including CIDEr, ROUGE-L, and SPICE.

2 PART A- CNN Encoder and RNN Decoder

2.1 Methodology

Below we define the steps and methods we followed in preparing the model

2.1.1 Data Preparation

- Obtain the custom dataset containing images and corresponding captions.
- Preprocess the dataset by resizing images, tokenizing captions, and creating vocabulary.

2.1.2 Model Design

- Implement a CNN-based encoder to extract features from images.
- Implement an RNN-based decoder to generate captions based on the extracted features.
- Define the structure and parameters of each component (encoder and decoder).

2.1.3 Training

- Initialize the encoder and decoder models.
- Train the models using the training dataset.
- Fine-tune the models using techniques like gradient descent and backpropagation.
- Monitor the training process for convergence and overfitting.

2.1.4 Validation

- Evaluate the trained models using the validation dataset.
- Calculate evaluation metrics such as BLEU, CIDEr, ROUGE, etc., to assess the quality of generated captions.

2.1.5 Testing

- Test the final trained models using the testing dataset.
- Generate captions for test images and evaluate the performance using the same evaluation metrics used for validation.

2.1.6 Code Implementation

- Implement the designed models and training/validation/testing procedures in code.
- Ensure modularity, readability, and reusability of the code.
- Include comments and documentation to explain the functionality of each component.

2.2 Model Description

Below we describe the model that we have used for our CNN based encoder and RNN decoder

2.2.1 Encoder Model:

The `EncoderCNN` class implements an image encoder using a ResNet-50 architecture. It takes images as input and extracts features from them, which are then embedded into a lower-dimensional space using a linear layer. This embedding is used as the input to the Decoder model for generating captions.

Key Components:

- **ResNet-50 Architecture**: The model utilizes a ResNet-50 architecture for feature extraction. The ResNet-50 is widely used in computer vision tasks due to its effectiveness in capturing hierarchical features.
- **Linear Embedding Layer**: After extracting features from the ResNet, a linear layer is applied to reduce the dimensionality of the feature vector to the specified embedding size.

2.2.2 Decoder Model:

The `DecoderRNN` class implements a text decoder using an LSTM-based architecture. It takes the embedded image features from the Encoder as input along with the caption tokens and generates captions word by word.

Key Components:

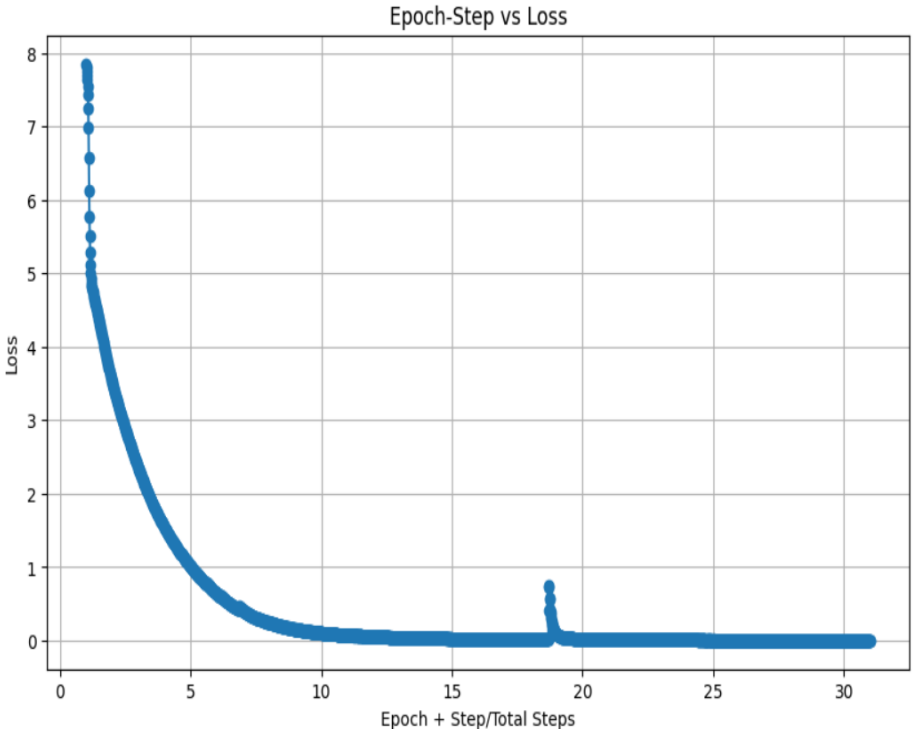
- **Embedding Layer**: The class initializes an embedding layer to convert tokenized captions into dense embedding vectors.

- **LSTM Layers**: The model consists of LSTM layers to process the embedded features and generate captions sequentially. It uses a configurable number of layers and dropout probability to control overfitting.
- **Dropout Layer**: Dropout regularization is applied to the output of the LSTM layers to prevent overfitting.
- **Fully-Connected Layer**: The final fully-connected layer maps the hidden states of the LSTM to the vocabulary size, producing the output logits for each word in the vocabulary.
- **Softmax Activation**: A softmax activation is applied to normalize the output logits, producing a probability distribution over the vocabulary at each time step.

The `sample` method in the `DecoderRNN` class is used for generating captions during inference. It takes the embedded image features as input and recursively predicts the next word in the caption until either the maximum caption length is reached or an end token is predicted.

2.3 Result and Discussion

Below is the loss function vs Epoch for our model

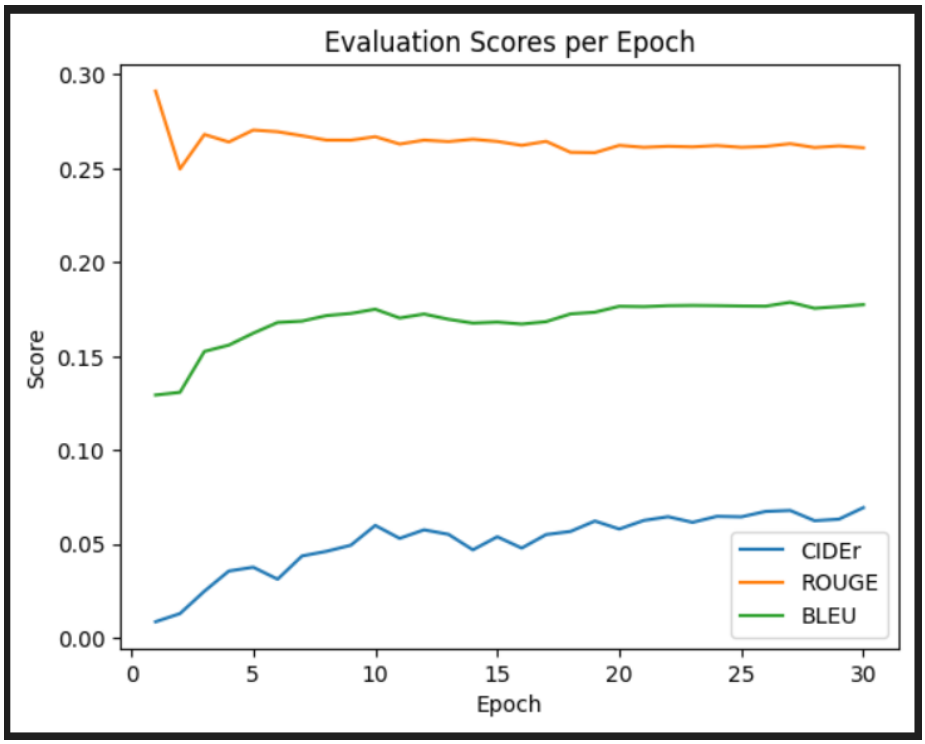


Img 1: Loss vs Epoch

We have trained the model for 30 epochs and saved the model after each of the epochs. Once done we load the model as the model to test. Running the model on all the 30 models we trained we generated the captions and then derived the scores for each of the different evaluation metrics to use. Then we select the one of the models which gave the highest scores, in out case we used model 20 (The model achieved after training 20 epochs).

Then we generate captions by using this model and print the generated captions

Below is the image for the scores of each model we have trained against the metric used



Img 2: Cider vs Bleu vs Rouge L