https://github.com/lurkingryuu/dl_term_project

Deep Learning Term Project CS60010 Spring Semester 2024

TEAM MEMBERS :

1. Anuguru Parthiv Reddy (21CS10006)
2. Vonteri Varshith Reddy (21CS10081)
3. Thota Kesava Chandra (21CS30056)
4. Yelisetty Karthikeya SM (21CS30060)

TASK: Build encoder decoder models for Automatic Image Captioning

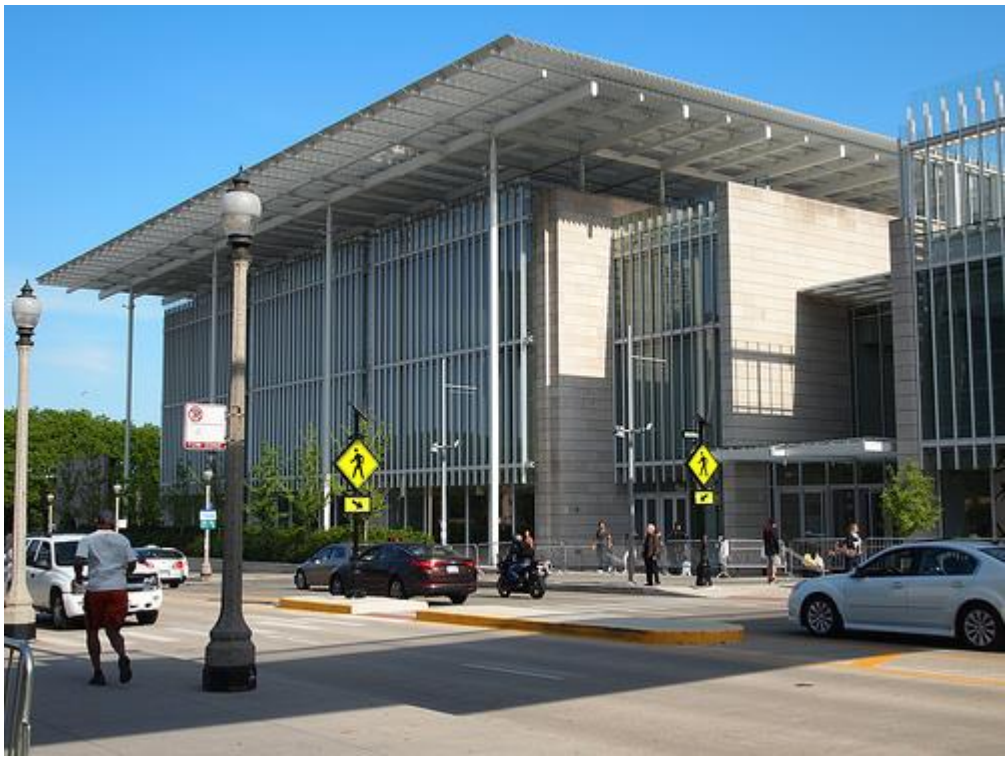
DATASET:

{https://drive.google.com/file/d/1FMVcFM78XZE1KE1rIkGBpCdcdI58S1LB/view?usp=sharing}

DESCRIPTION:

Given an input image, generate a caption for the image.

INPUT:



OUTPUT: A large building with bars on the windows in front of it. There are people walking in front of the building. There is a street in front of the building with many cars on it.

Evaluation metrics:

- ROUGE-L

PART – A : Simple CNN-RNN Encoder-Decoder Model

I have implemented the following classes and functions to build the encoder decoder model

1. Vocabulary

Description:
This class is responsible for handling the vocabulary of the captions. It provides methods to build a vocabulary from a list of sentences, convert captions to numbers, and vice versa. The vocabulary contains mappings between words and their corresponding indices.

Methods:

__init__(self): Initializes the vocabulary with special tokens.

build_vocabulary(self, sentence_list): Builds the vocabulary from a list of sentences.

caption_to_number(self, caption_text): Converts a caption text to a list of numbers using the vocabulary.

__len__(self): Returns the length of the vocabulary.

2. CustomCaptionDataset

Description:
This class is a custom dataset class that inherits from PyTorch's Dataset class. It loads image paths and corresponding captions from an annotation file, builds a vocabulary using the captions, and provides methods to get items from the dataset.

Methods:

__init__(self, root_dir, annotation_file, transform=None): Initializes the dataset with root directory, annotation file, and optional transform.

__len__(self): Returns the number of items in the dataset.

__getitem__(self, index): Returns an image and its corresponding caption at a given index.

3. EncoderCNN

Description:
This class defines the CNN-based encoder model using Inception V3 pretrained on ImageNet. It takes an image as input and outputs a feature vector of a specified size.

Methods:

__init__(self, embed_size, train_CNN=False): Initializes the encoder with Inception V3 and a fully connected layer.

forward(self, images): Forward pass of the encoder to extract features from images.

4. DecoderRNN

Description:
This class defines the RNN-based decoder model using an LSTM. It takes image features and caption embeddings as input and generates a sequence of words as captions.

Methods:

__init__(self, embed_size, hidden_size, vocab_size, num_layers): Initializes the decoder with an embedding layer, LSTM, and linear layer.

forward(self, features, captions): Forward pass of the decoder to generate captions from features and embeddings.

5. CNNtoRNN

Description:
This class combines the EncoderCNN and DecoderRNN to create an end-to-end model for image captioning. It takes an image and its corresponding captions as input and generates captions using the encoder and decoder.

Methods:

__init__(self, embed_size, hidden_size, vocab_size, num_layers): Initializes the encoder and decoder models.

forward(self, images, captions): Forward pass of the combined model to generate captions.

caption_image(self, image, vocabulary, max_length=50): Generates captions for a single image using the trained model and vocabulary.

6. Utility Functions

Description:
These are utility functions used for saving and loading model checkpoints and printing examples during training and testing.

Functions:

save_checkpoint(state, filename="my_checkpoint.pth.tar"): Saves model and optimizer state.

load_checkpoint(checkpoint, model, optimizer): Loads model and optimizer state from a checkpoint.

print_examples(model, device, dataset): Prints examples of captions generated by the model during training.

print_test(model, device, dataset): Prints examples of captions generated by the model on the test set during testing.

7. Training and Evaluation Functions

Description:
These are functions responsible for training the model, calculating metrics, and evaluating the model's performance.

Functions:

train(): Main function to train the model using the custom dataset and DataLoader.

calculate_metrics(references, hypotheses): Calculates various evaluation metrics including BLEU, METEOR, ROUGE-L, CIDEr, and SPICE scores.

The submission file is part-1.ipynb

PART – B: Transformer-Based Encoder-Decoder Model with ViT

Model Architecture:
The encoder uses a Vision Transformer (ViT), specifically the vit-small-patch16-224 model, to extract image features. The decoder is a transformer-based model, which takes the flattened feature map from ViT and generates captions using self-attention mechanisms.

Results and Analysis:

Rouge_L score: Score(precision=0.3614999245327389, recall=0.2839090920931191, fmeasure=0.30195367117005245)

Submission file is part-2.ipynb

Conclusion:

In this project, we implemented two different encoder-decoder models for automatic image captioning. The transformer-based model with ViT as the encoder outperformed the CNN-RNN model in terms of evaluation metrics. Both models successfully generated coherent and relevant captions for the given test images.

- Encoder-Decoder Models: https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning
- RNN Coding Tutorial: https://www.kaggle.com/code/kanncaa1/recurrent-neural-network-with-pytorch
- Image Captioning Coding Tutorial: https://www.youtube.com/watch?v=y2BaTt1fxJU&list=PLCJHEFznK8ZybO3cpfWf4gKbyS5VZgppW&index=1

Related papers:

- https://aclanthology.org/P18-1238.pdf
- https://arxiv.org/pdf/1411.4555.pdf