Deep Learning Report

Image Captioning with Encoder-Decoder Models

By Happiness and Souro:
Sourodeep Datta, 21CS10064
Ashwin Prasanth, 21CS30009
Anwesha Das, 21CS30007
Sreejita Saha, 21CS30052
Link to Repository: GitHub

Foreword:

Both of the provided notebooks have been executed in Kaggle using the P100 GPU. Their code, particularly the directory paths have been set with respect to the kaggle working directory '/kaggle/working/' and may require modification if run on Google Colab. Links to trained model weights have also been provided for both of the parts.

Part A

Data Preprocessing:

The zip file is downloaded and extracted. The captions from each of the .csv files are extracted and written to .txt files. Following this:

- Byte Pair Encodings are generated from the captions using the library 'SentencePiece', creating a vocabulary of 8000 tokens.
- A Dataset class is created called 'ImageCaptioningDataset'. It loads the necessary image transformations for the input images. This includes reshaping the image and normalizing it according to the parameters used during the training of the CNN. It then tokenizes the captions, appending an EOS token to them, followed by right-padding them to the length of the longest caption in the dataset.
- This dataset is loaded into a dataloader for all 3 partitions of the dataset (train, val and test). The train and validation dataloaders have a batch size of 64.

Model Architecture

The model uses the pretrained CNN encoder 'Inception ResNet v2' and an autoregressive LSTM decoder.

- The CNN encoder has it's output dimension size changed to $encoder\_dim = 1024$. Thus, the output encodings are of shape $(batch\_size, 1024)$. The layers below the output fully connected layer are frozen.
- The decoder has an embedding size of 1024, hidden state size of 1024 and 3 LSTM layers. The first input is the CNN encodings. It's output at each timestep is it's hidden states of shape $(batch\_size, seq\_length, 1024)$. These are passed through a fully connected layer to give an output of shape $(batch\_size, seq\_length, vocab\_size)$. The weights are initialized based on the layer, including normal initialization and He initialization.
- Model Weights

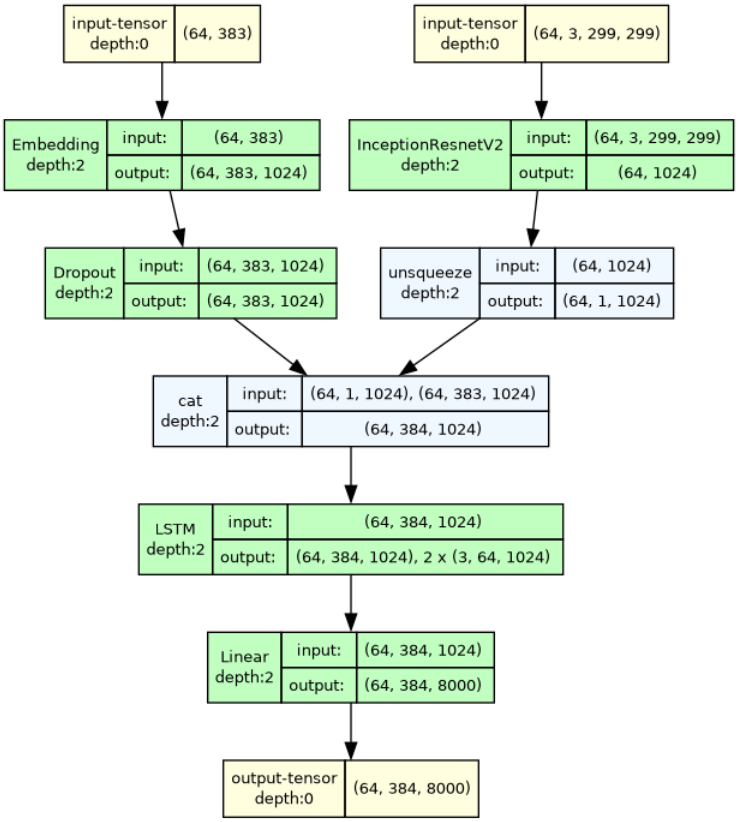

Figure 1: Model Architecture (Full Architecture)

Training and Inference:

The model is trained for 300 epochs, with checkpoints being saved every 100 epochs.

- The optimizer is AdamW with a learning rate of 0.001. The loss function is Cross Entropy, with the pad tokens being ignored during calculation.
- The decoder is trained with teacher forcing. The input is of the form [<caption>]. Within the model, it gets prepended with the encoder embeddings, giving [<encoder_embed> <caption>]. The labels are of form [<caption> <eos_token>].
- The validation set is iterated through after every epoch. **Note: The validation loss will be observed to increase after most epochs.** However this is not indicative of overfitting as the loss is with respect to the exact validation captions. It is unlikely an NLP model will generate the exact same captions as the validation captions, but may still be learning the overall context of the image. A better validation metric would have been BLEU/ROUGE-L/CIDEr but checking them during training is computationally expensive and was not feasible with currently available resources.

The inference is done via greedy decoding by inputting the CNN encodings and then autoregressively predicting the rest of the caption until is reaches an EOS token.

Results

Metric	Value
BLEU	0.420
ROUGE-L	0.326
CIDEr	0.090

Table 1: Evaluation Metrics

Part B

Data Preprocessing:

- Created a Dataset class called 'ImageCaptioningDataset'. It loads the Image Processor for ViT and Tokenizer for GPT2. It then tokenizes the captions, along with prepending "<|endoftext|>Image Caption: " and appending "<|endoftext|>" to each caption. The "Image Caption: " is meant to give GPT2 additional context as to what it's task is. Note that GPT2 has the same token for both BOS and EOS. It also left-pads the captions, and creates an attention mask for them. It creates labels of the form [-100 -100 ... <caption>], where -100 is used wherever there should be padding. Finally, it transforms the input image to get pixel_values, which are to be passed to the encoder.
- This dataset is loaded into a dataloader for all 3 partitions of the dataset (train, val and test). The train and validation dataloaders have a batch size of 8.

Model Architecture

The model uses the pretrained encoder 'ViT' and the pretrained decoder 'GPT2'.

- The encoder is loaded with it's pretrained weights, except for the last pooling layer. This is because the pooling layer is not required, as we require the hidden states. The encoder outputs it's hidden states of shape *(batch_size, 197, 768)*. All of it's layers are frozen.
- The decoder uses the 'GPT2LMHeadModel' from Hugging Face, with add_cross_attention set to True. All the weights other than the cross attention layer weights are initialized from the pretrained model. All of it's layers are trainable. The decoder takes the tokenized input (input_ids), attention mask, labels and the encoder's hidden states as inputs. The decoder uses the encoder hidden states for cross attention. It's final layer is a language modelling head i.e. it's a fully connected layer having output dimension $vocab\_size = 50257$. It returns both the loss as well as the logits of the last layer.
- Model Weights

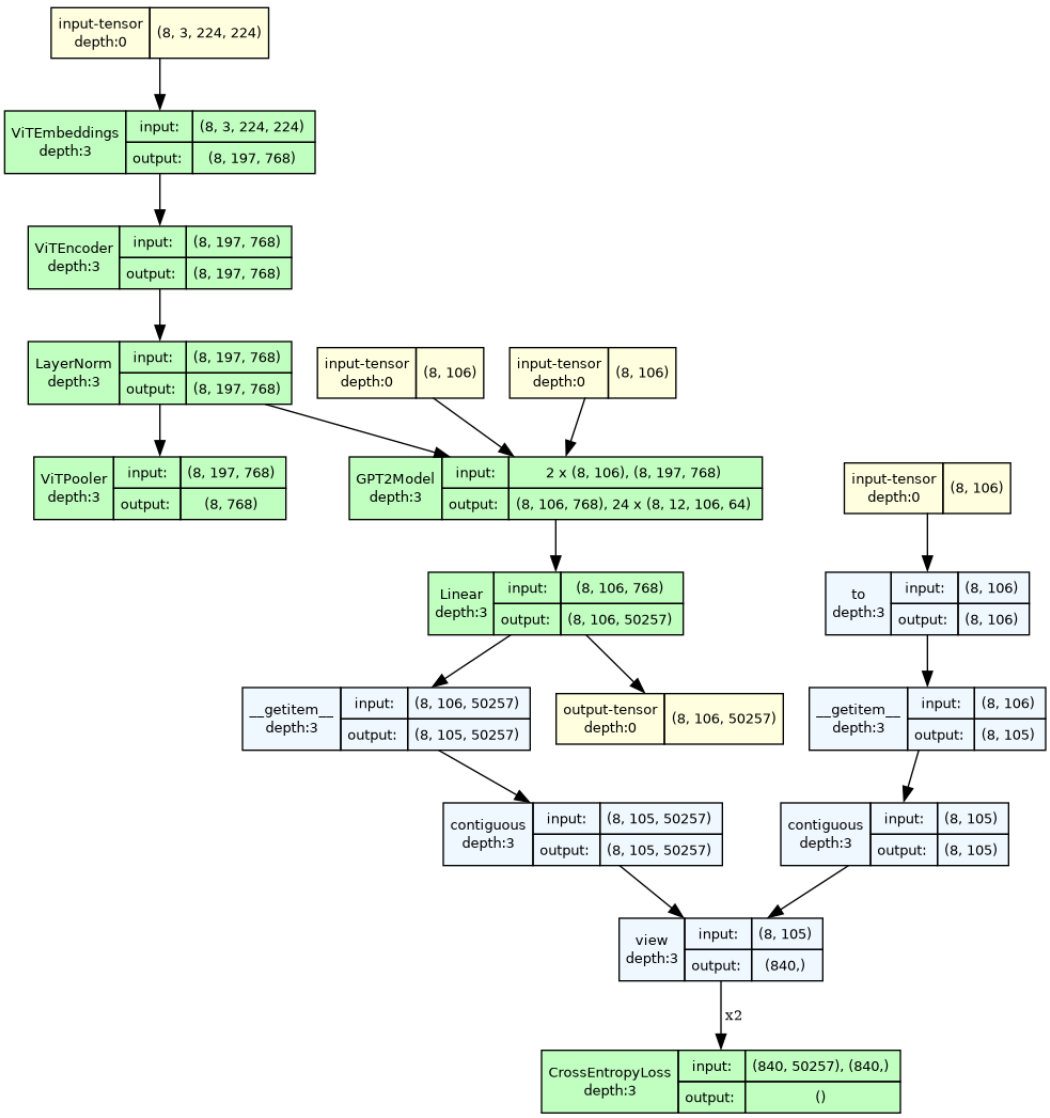

Figure 2: Model Architecture (Full Architecture)

Training and Inference:

The model is trained for 30 epochs, with checkpoints being saved every 10 epochs.

- The optimizer is AdamW with a learning rate of 3e-4.
- The decoder is trained with teacher forcing.
- The validation set is iterated through after every epoch. **Note: The validation loss will be observed to increase after most epochs (Reason explained in Part A)**

The inference is done via greedy decoding by inputting ["<|endoftext|>Image Caption:"] and then autoregressively predicting the rest of the caption until is reaches the EOS token.

Results

Metric	Value
BLEU	0.516
ROUGE-L	0.369
CIDEr	0.168

Table 2: Evaluation Metrics