PROJECT REPORT

IMAGE CAPTION GENERATION USING Vit-GPT2 IMAGE CAPTIONING PRETRAINED MODEL

SUBMITTED BY:

SANA AKBAR-026

UBAIDA WAHEED-034

TAYYBA SALAMAT-032

CODE Explanation:

1. Importing Libraries:

- The code starts by importing the necessary libraries and modules:
- `VisionEncoderDecoderModel`, `ViTFeatureExtractor`, and `AutoTokenizer` are imported from the `transformers` library.
- `torch` is imported from the PyTorch library.
- `Image` is imported from the PIL (Pillow) library.

2. Loading Pre-trained Models and Tokenizer:

- The code initializes the `model` object using `VisionEncoderDecoderModel.from_pretrained ()` and loads a pre-trained vision encoder-decoder model called "vit-gpt2-image-captioning".
- Similarly, the `feature_extractor` object is initialized using `ViTFeatureExtractor.from_pretrained ()` and loads the pre-trained feature extractor associated with the same model.
- The `tokenizer` object is initialized using `AutoTokenizer.from_pretrained()` and loads the pre-trained tokenizer associated with the model.

3. Setting the Device:

- The code checks if a CUDA-enabled GPU is available using `torch.cuda.is_available()`.
- If a GPU is available, the device is set to "cuda"; otherwise, it is set to "cpu".
- The model is then moved to the chosen device using `model.to(device)`.

4. Setting Caption Generation Parameters:

- The maximum length of the generated captions is set to 16 tokens (`max_length`).
- The number of beams used during generation is set to 4 (`num_beams`).
- These parameters are stored in the `gen_kwargs` dictionary for later use.

5. Defining the `predict_step()` Function:

- The `predict_step()` function takes a list of image paths as input (`image_paths`).
- It initializes an empty list called `images` to store the processed images.
- For each image path in the input list:
- The image is opened using `Image.open()` and stored in the variable `i_image`.
- If the image mode is not RGB, it is converted to RGB mode using `i_image.convert(mode="RGB")`.
- The processed image is appended to the `images` list.
- The `feature_extractor` extracts pixel values from the images and converts them into tensors using `feature_extractor (images=images, return_tensors="pt").pixel_values`.
- The pixel values tensor is moved to the specified device using `pixel_values.to(device)`.

- The `model.generate ()` method is called with the pixel values tensor and the caption generation parameters from `gen_kwargs`.
- The generated caption IDs are decoded into text using `tokenizer. batch_decode ()` and stored in the `preds` list.
- Any special tokens or extra whitespaces are removed from the generated captions using list comprehension.
- The final list of generated captions, `preds`, is returned.

6. Calling `predict_step ()`:

- The `predict_step ()` function is called with a list containing a single image path `['/content/WhatsApp Image 2023-04-24 at 12.28.40 AM.jpeg']`.
- This triggers the image caption generation process.

The code loads pre-trained models (vision encoder-decoder model, feature extractor, and tokenizer) and sets up the necessary components for image captioning. It defines a function, `predict_step ()`, to generate captions for a given list of image paths. Finally, it calls the `predict_step () ` function with a single image path to generate captions for that specific image.

Important Libraries used:

1. Transformers Library:

- The Transformers library is a popular open-source library developed by Hugging Face. It provides a wide range of pre-trained models and tools for natural language processing (NLP) tasks.
- In the code, the Transformers library is used to import three important classes:
- VisionEncoderDecoderModel`: This class represents a vision encoder-decoder model, which combines computer vision and natural language processing. It can be used for tasks like image captioning, where the model generates captions based on input images.
- ViTFeatureExtractor`: This class is a feature extractor specifically designed for Vision Transformers (ViT). It can extract visual features from images and convert them into tensors that can be used as input to the model.
- AutoTokenizer`: This class is used to automatically select an appropriate tokenizer based on the provided model name or type. It simplifies the process of choosing the correct tokenizer for a specific model.
- By utilizing the classes from the Transformers library, you can easily load pre-trained models, perform inference, and process input data for various NLP and computer vision tasks.

2. Torch Library:

- The Torch library is a popular open-source machine learning framework primarily focused on tensor computations. It provides a wide range of functionalities for building and training neural networks.
- In the code, the Torch library is imported to leverage its tensor operations and device management capabilities.

- Specifically, the following Torch components are used:
- `torch.device`: This class is used to specify the device where tensor computations will be performed. In the code, it checks if a CUDA-enabled GPU is available and assigns the device to "cuda" if true, otherwise "cpu".
- `model.to(device)`: This method is used to move the vision encoder-decoder model to the specified device. It ensures that the model is compatible with the chosen device for efficient computation.

3. PIL (Pillow) Library:

- The PIL (Python Imaging Library) or Pillow library is a widely-used library for image processing and manipulation in Python.
- In the code, the `Image` class from the PIL library is imported to handle image-related operations.
- Specifically, it is used to open and preprocess images before passing them to the vision encoder-decoder model. It ensures that the images are in RGB format and converts them if necessary.

By leveraging these libraries, the code combines the power of the Transformers library for pre-trained models and tokenization, the Torch library for tensor computations and device management, and the PIL library for image processing to create an image captioning pipeline.

Explanation of ViT-gpt2 Image Captioning Model:

The "vit-gpt2-image-captioning" model is an architecture that combines the Vision Transformer (ViT) and GPT-2 models for image captioning tasks. Let's dive into the details of how this model works and its architecture:

1. Vision Transformer (ViT):

- The Vision Transformer (ViT) model applies the Transformer architecture to visual data, treating images as sequences of patches.
- The input image is divided into fixed-size patches, and each patch is flattened and linearly projected to obtain a patch embedding.
- The patch embeddings are then processed by a stack of Transformer encoder layers. Each encoder layer consists of self-attention mechanisms and feed-forward neural networks.
- The self-attention mechanism allows the model to capture dependencies between different patches, capturing both local and global information.
- The outputs from the Transformer encoder layers are passed through a classification head to predict the class labels or regression targets for the image.

2. GPT-2 (Generative Pre-trained Transformer 2):

- GPT-2 is a generative language model based on the Transformer architecture, designed for natural language processing tasks.
- It consists of a stack of Transformer decoder layers, which capture the dependencies between tokens in a sequence to generate coherent and contextually relevant outputs.
- During pre-training, GPT-2 is trained to predict the next word in a sequence given the previous context. This allows it to learn rich semantic and syntactic representations of language.

- During fine-tuning or downstream tasks, GPT-2 can be used for tasks like text completion, summarization, and generation.

3. Architecture of "vit-gpt2-image-captioning":

- The "vit-gpt2-image-captioning" architecture combines the ViT model as the encoder and the GPT-2 model as the decoder within an encoder-decoder framework.
- The ViT encoder processes the input image by extracting visual features using self-attention mechanisms and feed-forward neural networks.
- The output of the ViT encoder is a representation of the image, capturing both local and global visual information.
- This image representation is then fed into the GPT-2 decoder, which generates a sequence of tokens based on the learned language patterns and the visual context from the image.
- The GPT-2 decoder is responsible for producing the textual captions for the given image.
- By combining the ViT and GPT-2 models, the "vit-gpt2-image-captioning" model is able to generate captions that are semantically relevant to the content of the input image.

In summary, the "vit-gpt2-image-captioning" model uses the Vision Transformer (ViT) as the encoder to process visual input and extract visual features, and the GPT-2 model as the decoder to generate textual captions based on the visual context. This architecture allows the model to perform image captioning tasks effectively.