Automatic Image Captioning with Model Benchmarking and Robustness Analysis

Rishabh Sukumaran (22CS10058)[a], B Pranesh Vijay (22CS10013)[a] and Saishree Pradhan (22CS10089)[a]

[a]*Indian Institute of Technology, Kharagpur*

Abstract—This assignment implements a framework for image captioning using a custom **transformer-based** encoder-decoder model, benchmarked against the zero-shot performance of **SmolVLM**. We evaluate model robustness under varying **occlusion levels** and develop a **BERT-based classifier** to distinguish between captions generated by the two models. Our custom model integrates **Vision Transformers (ViT)** for image encoding and **GPT-2** for text decoding, achieving competitive performance on standard metrics.

Keywords—*Image Captioning, Transformer-based Models, Vision Transformers (ViT), GPT-2, SmolVLM, Occlusion Levels, BERT-based Classifier, BLEU, ROUGE-L, METEOR*

1. Introduction

Deep learning has revolutionized computer vision and natural language processing, enabling models to generate descriptive captions for images. Image captioning bridges the gap between vision and language, requiring models to understand visual content and produce coherent text.

In recent years, transformer-based models like Vision Transformer (ViT) and GPT-2 have excelled in vision and language tasks. ViT effectively extracts visual features, while GPT-2 generates high-quality text. In this assignment, we build a custom image captioning model combining ViT for image encoding and GPT-2 for text generation, evaluating it against the zero-shot SmolVLM baseline. Additionally, we analyze the model's robustness to image occlusion and use a BERT-based classifier to compare captions generated by both models.

2. Zero-shot Evaluation with SmolVLM

We used SmolVLM's pretrained model as a baseline for captioning images without task-specific training. To see how different prompts affect the model's performance, we tested two prompts: *Generate a caption for this image* and *Describe this image*. The first prompt led to more detailed and contextually accurate captions, aligning better with the reference captions compared to the shorter outputs from the second prompt. We evaluated the model using BLEU, ROUGE-L, and METEOR scores, which provided a benchmark for comparing our custom encoder-decoder model.

3. Custom Encoder-Decoder Model Implementation

The custom model, implemented in the `ImageCaptionModel` class, combines a pretrained **Vision Transformer (ViT, *vit_base_patch16_224*)** for image feature extraction with a **GPT-2** decoder for caption generation. To address GPT-2's text-only limitation, we introduce a `GPT2CrossAttention` layer to allow the decoder to incorporate image features. Both ViT and GPT-2 have a hidden size of 768 and 12 layers, making them compatible and minimizing the need for additional alignment.

The model is trained and evaluated using BLEU, ROUGE-L, and METEOR scores, with performance compared to SmolVLM's zero-shot baseline.

3.1. Encoder: Vision Transformer (ViT)

The ViT encoder extracts features from input images using the *vit_base_patch16_224* architecture from the `timm` library. Key steps include:

- **Patch Embedding**: The image is split into 16×16 patches, producing 196 patches ($224/16 = 14$ per dimension). Each patch is embedded into a 768-dimensional vector.
- **CLS Token**: A class token is added to the sequence to represent the image globally.
- **Positional Embedding**: Positional information is added to the patches and class token.
- **Transformer Blocks**: The sequence of patches passes through 12 transformer blocks.
- **Output**: The output is a sequence of 197 embeddings (196 patches + 1 CLS token), each of size 768.

3.2. Decoder: GPT-2 with Cross-Attention

The GPT-2 decoder is extended with a `GPT2CrossAttention` layer to attend to both text and image features. Key components include:

- **Token Embedding**: Converts token IDs into 768-dimensional embeddings.
- **Positional Embedding**: Adds learned position embeddings for sequences up to 1024 tokens.
- **Transformer Blocks**: The decoder consists of 12 blocks with self-attention, cross-attention, and feed-forward layers.
- **Language Modeling Head**: The final layer projects the output to vocabulary logits for next-token prediction.

3.3. Cross-Attention Mechanism

The cross-attention mechanism allows the GPT-2 decoder to attend to image features from the ViT encoder. Key steps include:

- **Query**: Generated from the self-attention output.
- **Key/Value**: Derived from the image features.
- **Computation**: Multi-head attention is applied, followed by normalization and linear projection.

3.4. Training, Generation, and Evaluation Pipeline

The model is trained and evaluated using the following pipeline:

- **Training Pipeline**: The `AdamW` optimizer and `OneCycleLR` scheduler are used with mixed-precision (FP16) training. A staged unfreezing strategy fine-tunes different components progressively.
- **Generation Process**: Caption generation starts with a special `<|endoftext|>` token and proceeds autoregressively. The decoder attends to visual features and generates tokens until an `eos` token or the maximum sequence length is reached.
- **Evaluation Pipeline**: Generated captions are evaluated using BLEU, ROUGE-L, and METEOR metrics to assess quality and relevance.

4. Image Perturbation and Model Robustness Analysis

4.1. Image Perturbation via Occlusion

To test how well the image captioning models handle partial image loss, we applied a patch-wise occlusion technique. Each image was divided into 16×16 non-overlapping patches. A percentage of these patches (10%, 50%, and 80%) were randomly selected, and their pixel values were set to black (RGB = 0, 0, 0), effectively masking those regions of the image.

Let the image I have dimensions $H \times W \times C$ (height H, width W, and color channels C). Dividing the image into 16×16 patches gives a grid of $\frac{H}{16} \times \frac{W}{16}$ patches. For a given occlusion percentage

$p \in \{10, 50, 80\}$, the number of masked patches N_{mask} is calculated as:

$$N_{mask} = \text{round}\left(\frac{p}{100} \times \frac{H}{16} \times \frac{W}{16}\right)$$

The selected N_{mask} patches are masked by setting their pixels to black. This process was repeated for all images in the test set at each occlusion level.

4.2. Evaluation of Model Robustness

After occluding the images, we tested both the SmolVLM-Instruct and custom models. For each occluded image, we generated captions from both models. We then calculated the BLEU, ROUGE-L, and METEOR scores by comparing the generated captions with the ground truth captions.

To measure the impact of occlusion, we calculated the change in each metric between the original (unoccluded) images and the occluded images at each level. The change in a metric M at a perturbation level p is:

$$\Delta M(p) = M(p\%) - M(0\%)$$

This process was done for each metric (BLEU, ROUGE-L, METEOR) at occlusion levels of 10%, 50%, and 80% for both models. The captions were saved for further analysis in Part C.

5. BERT-based Classifier Implementation

5.1. Data Preparation and Preprocessing

The first step was to load the dataset from the `model_classification.csv` file into a pandas DataFrame. This DataFrame contained two columns:

- **input_text**: A combination of the original caption, generated caption, and perturbation percentage, separated by `<SEP>` tokens.
- **label**: A numerical value indicating the source of the caption, where `Model A` (SmolVLM) was mapped to 0 and `Model B` (Custom Model) to 1.

We then used the `bert-base-uncased` tokenizer to prepare the text data for BERT. The tokenizer performed the following steps:

- **Tokenization**: Splitting the input text into tokens.
- **Special Token Addition**: Adding special tokens like `[CLS]` at the beginning and `[SEP]` at the end of each sequence.
- **Token ID Conversion**: Converting tokens into corresponding IDs based on the BERT vocabulary.

To ensure uniform sequence length, we calculated the maximum sequence length across all input texts and padded shorter sequences. We then used the `encode_plus` method to tokenize the text, add special tokens, and generate attention masks. These inputs were stored as PyTorch tensors.

The dataset was split into training (70%), validation (10%), and testing (20%) sets using PyTorch's `random_split` function, and DataLoaders were created for efficient batch processing.

5.2. BERT-based Classifier Model

We built a binary classifier using the pre-trained `bert-base-uncased` model. The architecture of the `CaptionClassifier` consists of:

- **BERT Layer**: The pre-trained BERT model outputs contextualized word embeddings.
- **Classifier Layer**: A simple feed-forward neural network with:
  - (i) A linear layer mapping the `[CLS]` token output to a hidden size of 256.
  - (ii) A ReLU activation function.
  - (iii) A dropout layer with a 0.3 probability.
  - (iv) A final linear layer for binary classification (Model A or Model B).

The `forward` method of the classifier processes the input IDs and attention masks through BERT, then passes the output through the classifier to produce logits. If labels are provided, it calculates the binary cross-entropy loss.

5.3. Model Training

The training loop, implemented in the `train_classifier` function, runs for a specified number of epochs. In each epoch:

- The gradients are reset.
- A forward pass is performed, and the loss is computed.
- The loss is backpropagated, and the model's parameters are updated.
- The learning rate is adjusted by the scheduler.

After each epoch, the model is evaluated on the validation set to track its performance. We calculate precision, recall, and F1-score based on the model's predictions.

The training uses the `AdamW` optimizer with a learning rate of 2e-5 and a learning rate scheduler. The loss function used is `BCEWithLogitsLoss`.

5.4. Model Evaluation

The `evaluate_classifier` function evaluates the trained model using the validation DataLoader. During evaluation:

- A forward pass is performed to obtain logits.
- The logits are passed through a sigmoid function to get probabilities.
- Probabilities are converted to binary predictions using a 0.5 threshold.

After processing all batches, we calculate the precision, recall, and F1-score using scikit-learn's `precision_recall_fscore_support` function. The evaluation metrics are returned as a dictionary and printed for the final model assessment.

6. Results and Analysis

6.1. Image Captioning Performance

This section compares the baseline performance of two image captioning models—SmolVLM-Instruct (zero-shot) and Custom Model on the unoccluded test set (0% occlusion). These results serve as a reference for evaluating the impact of image occlusion. SmolVLM-Instruct was evaluated using two distinct prompt formulations to assess its robustness under zero-shot conditions:

- **Prompt P1**: `"Generate a caption for this image"`
- **Prompt P2**: `"Describe this image"`

Table 1. Baseline Image Captioning Performance (0% Occlusion)

Model	BLEU	ROUGE-L	METEOR
SmolVLM-Instruct (Zero-Shot, P1)	0.0872	0.2691	0.2824
SmolVLM-Instruct (Zero-Shot, P2)	0.0393	0.2277	0.3021
Custom Model	0.0493	0.2216	0.2450

As seen in Table 1, the custom model outperforms one of the SmolVLM-Instruct variants across some evaluation metrics. Among the two prompt formulations, Prompt P1 leads to marginally better performance, highlighting the sensitivity of zero-shot models to prompt design. These results emphasize the advantage of task-specific fine-tuning for achieving superior captioning quality.

6.2. Robustness to Image Occlusion

This section analyzes the performance of both models—SmolVLM-Instruct (Zero-Shot) and the Custom Model under varying levels of image occlusion (10%, 50%, and 80%).

The following tables summarize the absolute and relative performance of both models as occlusion severity increases.

Table 2. Image Captioning Performance Under Occlusion — SmolVLM (P1)

Occlusion Level	BLEU	ROUGE-L	METEOR
10%	0.0847	0.2676	0.2797
50%	0.0814	0.2575	0.2602
80%	0.0695	0.2372	0.2312

Table 3. Image Captioning Performance Under Occlusion — SmolVLM (P2)

Occlusion Level	BLEU	ROUGE-L	METEOR
10%	0.0361	0.2295	0.3160
50%	0.0306	0.2195	0.2990
80%	0.0293	0.2126	0.2594

Table 4. Image Captioning Performance Under Occlusion - Custom Model

Occlusion Level	BLEU	ROUGE-L	METEOR
10%	0.0530	0.2234	0.2476
50%	0.0482	0.2223	0.2434
80%	0.0443	0.2134	0.2316

Table 5. Relative Change in Performance Due to Occlusion — SmolVLM (P1)

Occlusion Level	Δ BLEU	Δ ROUGE-L	Δ METEOR
10%	-0.0025	-0.0015	-0.0027
50%	-0.0058	-0.0116	-0.0222
80%	-0.0177	-0.0319	-0.0512

Table 6. Relative Change in Performance Due to Occlusion — SmolVLM (P2)

Occlusion Level	Δ BLEU	Δ ROUGE-L	Δ METEOR
10%	-0.0032	0.0018	0.0139
50%	-0.0087	-0.0082	-0.0031
80%	-0.0100	-0.0151	-0.0427

Table 7. Relative Change in Performance Due to Occlusion — Custom Model

Occlusion Level	Δ BLEU	Δ ROUGE-L	Δ METEOR
10%	0.0037	0.0018	0.0026
50%	-0.0011	0.0007	-0.0016
80%	-0.0050	-0.0082	-0.0134

Despite varying levels of occlusion, both the SmolVLM (Zero-Shot) model and the custom-trained model demonstrated consistent performance trends, with degradation proportional to the severity of the occlusion.

For **SmolVLM (P1)**, as shown in Table 2, performance drops steadily across all three metrics (BLEU, ROUGE-L, METEOR) as occlusion increases. The relative change, detailed in Table 5, indicates moderate robustness at lower occlusion levels (10% drop: BLEU −0.0025, ROUGE-L −0.0015) but a more noticeable decline at 80% occlusion (BLEU −0.0177, METEOR −0.0512).

Similarly, **SmolVLM (P2)**, shown in Table 3, followed the same trend. Interestingly, Table 6 reveals slightly smaller absolute drops for BLEU but larger fluctuations in METEOR compared to P1, suggesting that P2 captions retained partial semantic relevance even when token-level matching (BLEU) suffered under heavy occlusion.

The **Custom Model** demonstrated a more stable profile under occlusion, as seen in Table 4. The relative performance changes, presented in Table 7, are generally less severe than those observed for SmolVLM, with METEOR showing minimal degradation even at high occlusion levels (from −0.0016 at 50% to −0.0134 at 80%).

Overall, the results show that while both models degrade under increased occlusion, the **custom** model is more resilient to partial image loss compared to **SmolVLM-Instruct**, particularly at higher occlusion percentages. Additionally, differences between P1 and P2 for SmolVLM suggest that prompt design can subtly affect the robustness of zero-shot captioning under visual corruption.

6.3. Classification Performance

This section presents the performance of a BERT-based classifier trained to distinguish between captions generated by SmolVLM-Instruct (P1) and the custom model. The input to the classifier was a concatenation of the original caption, the generated caption, and the associated perturbation level.

Table 8. Performance of the BERT-based Classifier (P1 vs. Custom Model)

Evaluation Setup	Precision	Recall	F1-Score
Classifier (P1 vs. Custom Model)	0.9972	0.9944	0.9958

The classifier achieves near-perfect performance, with a precision of 99.72%, recall of 99.44%, and an F1-score of 99.58%. These results highlight that the captions generated by SmolVLM-Instruct (P1) and the custom model exhibit clear and distinguishable patterns, which the classifier can reliably learn, even when the input captions correspond to occluded images.

7. Conclusion

This report analyzed the performance of two image captioning models: the zero-shot SmolVLM model and a custom-trained model, under varying levels of image occlusion. The results indicate that the custom-trained model outperforms the zero-shot model on unoccluded images. However, as the level of occlusion increases, both models exhibit significant degradation in caption quality. Notably, the custom-trained model, despite its stronger baseline performance, maintains more stable performance across varying levels of occlusion, suggesting it is more resilient to partial image loss.

In addition to caption generation, we evaluated the effectiveness of a BERT-based classifier designed to distinguish between captions produced by the two models. The classifier achieved near-perfect precision, recall, and F1-scores, even when the input included captions associated with occluded images. This indicates that the linguistic patterns and structures of the generated captions remain distinct, likely reflecting underlying differences in model training and architecture.

This assignment offers two key takeaways. First, it highlights the challenge of building image captioning models that are robust to real-world visual noise, such as occlusion. Future work should prioritize strategies for improving model resilience to incomplete or corrupted visual inputs. Second, the strong performance of the BERT-based classifier underscores the potential of language models in evaluating and analyzing generated captions, even in the absence of the original image or ground truth.

8. Appendix

8.1. Custom Model Architecture

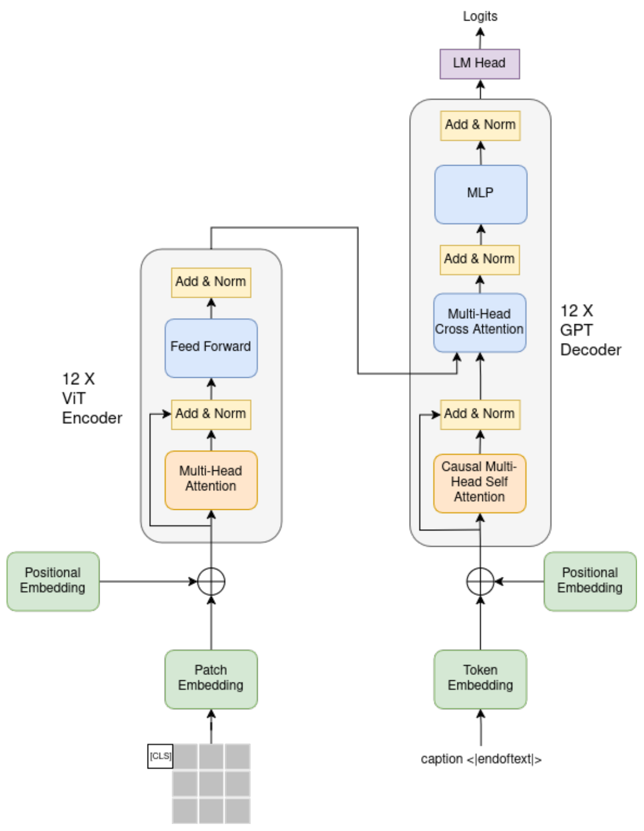

Figure 1. Model Architecture.

The architecture of the model used for training consists of a base pre-trained GPT model with additional layers for fine-tuning. The early epochs involve freezing the GPT layers, while later epochs progressively unfreeze them to allow for further adaptation to the task at hand. The final step involves unfreezing all layers for full training.

8.2. Training and Validation Perplexity

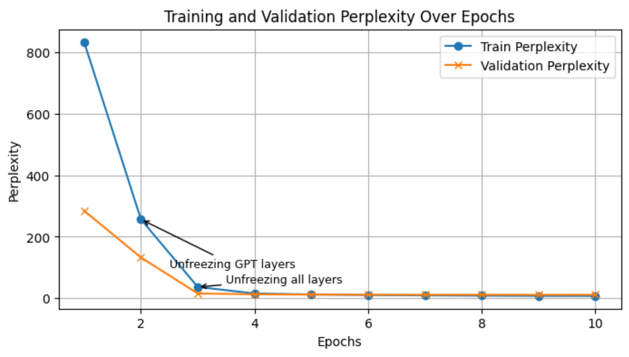

Figure 2. Training and Validation Perplexity over Epochs

Figure 2 shows the perplexity scores for both the training and validation sets over the 10 epochs. As the model progresses, the perplexity decreases, with the validation perplexity reaching its lowest value at epoch 8, indicating the best generalization performance.

8.3. Training and Validation Loss

Figure 3 illustrates the training and validation loss across the 10 epochs. The arrows in the graph indicate the points where the model's layers were unfrozen:

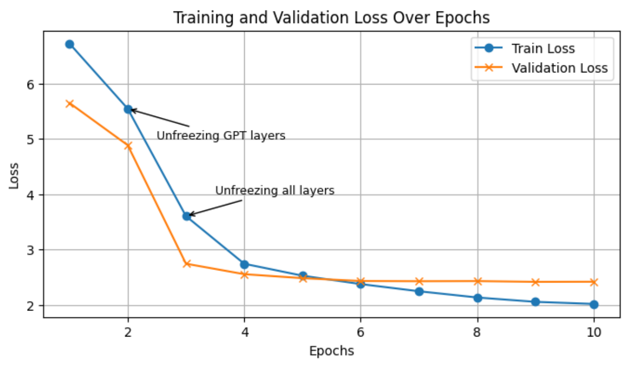

Figure 3. Training and Validation Loss over Epochs

- Epoch 2: Unfreezing the GPT layers for further fine-tuning.
- Epoch 4: Unfreezing all layers for full training.

These unfreezing steps allowed the model to progressively adapt and improve performance, as seen in the loss and perplexity trends.

8.4. BERT Classifier Training Results

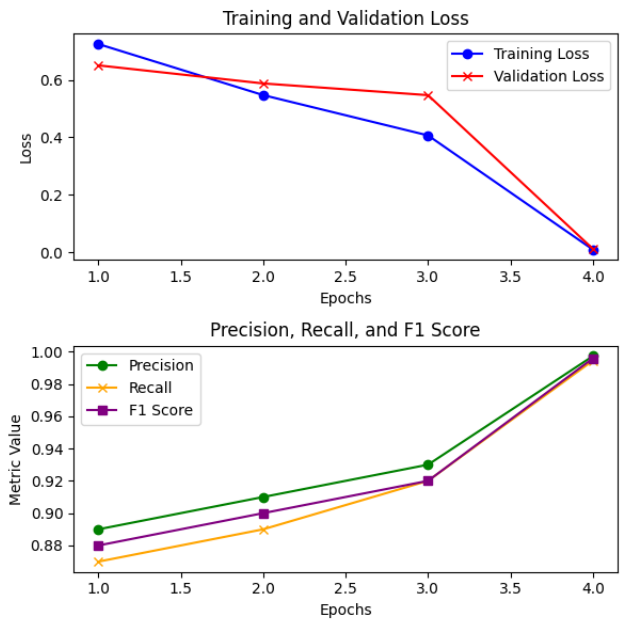

Figure 4. BERT Classifier Training: Loss and Performance Metrics

Figure 4 shows the training and validation loss, as well as the precision, recall, and F1 scores for the BERT classifier over 4 epochs. The first graph shows the loss trends, and the second graph illustrates the performance metrics across these epochs, helping to assess the classifier's effectiveness in training.