

## **Masters Programmes: Group Assignment Cover Sheet**

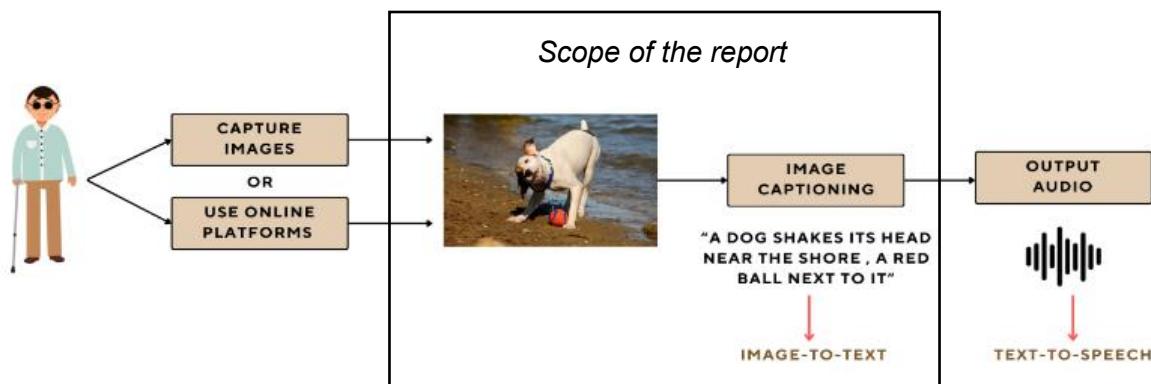
<b>Student Numbers:</b> Please list numbers of all group members	2294150, 5583308, 5584080, 5588187, 2182698, 5530977, 5567131, 5539729
<b>Module Code:</b>	IB9CW0
<b>Module Title:</b>	Text Analytics
<b>Submission Deadline:</b>	13 <sup>th</sup> May 2024
<b>Date Submitted:</b>	12 <sup>th</sup> May 2024
<b>Word Count:</b>	1,200
<b>Number of Pages:</b>	14 (excluding Appendices)
<b>Question Attempted:</b> <i>(question number/title, or description of assignment)</i>	Comprehensive study of image-to-text generation techniques and applications
<b>Have you used Artificial Intelligence (AI) in any part of this assignment?</b>	Yes
<p><b>Academic Integrity Declaration</b>            We're part of an academic community at Warwick. Whether studying, teaching, or researching, we're all taking part in an expert conversation which must meet standards of academic integrity. When we all meet these standards, we can take pride in our own academic achievements, as individuals and as an academic community.</p> <p>Academic integrity means committing to honesty in academic work, giving credit where we've used others' ideas and being proud of our own achievements.</p> <p>In submitting my work, I confirm that:</p> <ul style="list-style-type: none"> <li>▪ I have read the guidance on academic integrity provided in the Student Handbook and understand the University regulations in relation to Academic Integrity. I am aware of the potential consequences of Academic Misconduct.</li> <li>▪ I declare that this work is being submitted on behalf of my group and is all our own, , except where I have stated otherwise.</li> <li>▪ No substantial part(s) of the work submitted here has also been submitted by me in other credit bearing assessments courses of study (other than in certain cases of a resubmission of a piece of work), and I acknowledge that if this has been done this may lead to an appropriate sanction.</li> <li>▪ Where a generative Artificial Intelligence such as ChatGPT has been used I confirm I have abided by both the University guidance and specific requirements as set out in the Student Handbook and the Assessment brief. I have clearly acknowledged the use of any generative Artificial Intelligence in my submission, my reasoning for using it and which generative AI (or Ais) I have used. Except where indicated the work is otherwise entirely my own.</li> <li>▪ I understand that should this piece of work raise concerns requiring investigation in relation to any of points above, it is possible that other work I have submitted for assessment will be checked, even if marks (provisional or confirmed) have been published.</li> <li>▪ Where a proof-reader, paid or unpaid, was used, I confirm that the proof-reader was made aware of and has complied with the University's proofreading policy.</li> </ul> <p><b>Upon electronic submission of your assessment, you will be required to agree to the statements above</b></p>	

## Contents

1. Motivation .....	2
2. Methodology .....	2
2.1 Convolutional Neural Networks (CNN) – Long Short-Term Memory Networks (LSTM)2	
2.2 Transformers .....	3
2.3 Evaluation.....	6
2.3.1 Stress-test scenarios.....	6
2.3.2 Performance metrics .....	6
2.4 Limitations .....	6
3. Findings .....	6
4. Conclusion .....	11
5. References .....	12
6. Appendices .....	15
Appendix 1: Other Applications' Frameworks and Architecture.....	15
Appendix 2: Transformer-based Model Overview.....	17
Appendix 3: BLIP Architecture and Overview .....	19
Appendix 4: GIT Overview .....	21
Appendix 5: Moondream2 Overview .....	22
Appendix 6: ChatGPT Prompt.....	23
Appendix 7: Model Captions & Qualitative Judgment.....	36
Appendix 8: Model Quantitative Evaluation Scores (BLEU & ROUGE) .....	38

## 1. Motivation

Artificial Intelligence (AI) exponential advancements have emerged as a catalyst for substantial improvements in life quality (Alowais et al., 2023). This report delves into various image captioning techniques underpinning a wide range of applications for the visually impaired. These image-to-text innovations, integrated with text-to-speech, will enhance the cognitive and perceptual experiences of visually impaired users in both social (e.g. route navigation) and business settings (e.g., product description on an ecommerce platform), and other applications set out in Appendix 1.

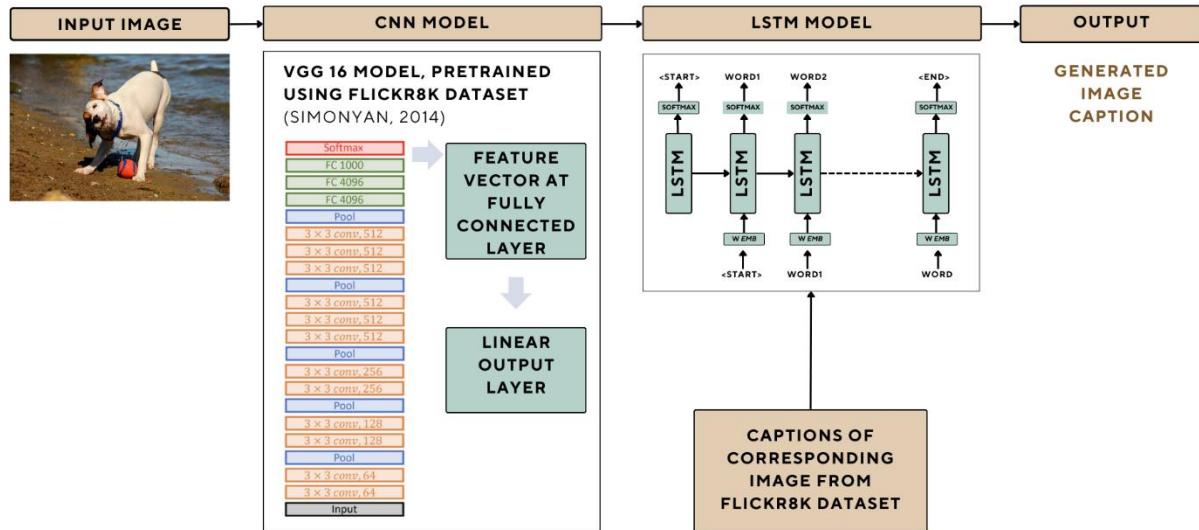


**Figure 1.** Image to Speech in Aiding Blind People

## 2. Methodology

### 2.1 Convolutional Neural Networks (CNN) – Long Short-Term Memory Networks (LSTM)

Image-to-text sits at the intersection of image-recognition and natural-language-processing, both of which have been revolutionized by the Transformer architecture. Prior to them, most image-to-text models mixed CNN and LSTMs. Examples include Google Neural Image Caption (NIC) (Vinyals et al., 2015) and its improved version Show and Tell or Show, Attend and Tell (Xu et al., 2016).

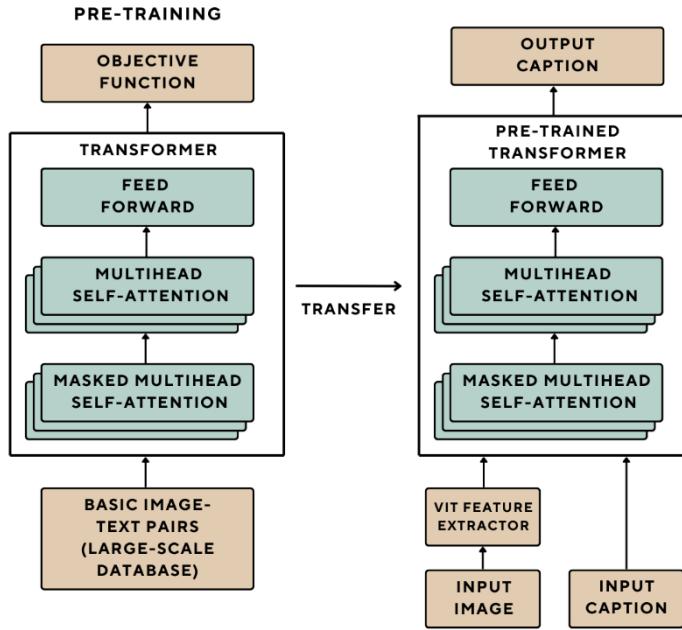


**Figure 2.** Flow Diagram of CNN and LSTM Model

This report replicates the CNN-LSTM structure by deploying a pre-trained VGG16 model (CNN with 16 layers deep) for feature extraction, which are tokenized and passed through to a pretrained LSTM decoder for text generation (Figure 2). Additionally, a transfer learning approach for a small sample is also included in the attached code for learning and demonstration purposes.

## 2.2 Transformers

Transformer models, with self-attention and cross-attention mechanisms and pre-training on large and diverse image-text pairs, effectively weight the importance of different parts of image input, enhance contextual understanding and facilitate more complex and nuanced language generation (Figure 3).



**Figure 3.** Visual-language Pre-training Transformer-based Captioning

Since limited computational capacities place constraints to fine-tuning, this study evaluates performance of the following recent open-sourced models: Salesforce BLIP (Radford et al. 2021) and BLIP-2 (Li et al., 2023), Microsoft GIT-based COCO (Wang et al., 2022), and Moondream2 (Vikhvatk, 2024). Architecture and differentiators of each are summarised in Table 1 with further details in Appendices 2-5.

**Table 1.** Transformer-based Model Comparison

Model	BLIP	BLIP-2	GIT based COCO	Moondream2
Release/Owner	2022/Salesforce	2023/Salesforce	2022/Microsoft	2024/Moondream.ai
Differentiator	Uses CapFit (bootstrap the captions, a <b>captioner</b> generates synthetic captions and a <b>filter</b> removes noisy image-text pairs). Allows training on larger bootstrapped dataset.	Uses <b>frozen pre-trained image encoders</b> and <b>frozen large language</b> models, reducing the number of trainable parameters.	Uses <b>one image encoder</b> and <b>one text decoder</b> under a single language modelling task while scaling up the pre-training data.	Trained on 1.8 billion parameters and still a <b>lightweight</b> model.
Model Architecture	Multimodal mixture of <b>ViT</b> image encoder, <b>BERT</b> text encoder and BERT text decoder  (Figure A.3)	Multimodal mixture of a <b>CLIP-like</b> image encoder, a <b>LLM</b> , and a <b>Querying Transformer</b> (Qformer) acting as an information bottleneck between encoder and LLM  (Figure A.5)	Multimodal mixture of <b>Swin-like vision</b> transformer image encoder and <b>BERT</b> text decoder  (Figure A.6)	Uses <b>SigLIP</b> processing one pair of image/text at a time, simplifying the encoding process (Zhai et al., 2023) and <b>Phi-1.5</b> which was trained using quality data (textbooks) (Li et al., 2023).
Beyond Image captioning	Enhances image comprehension and responsiveness to image-related queries	Zero-shot image-to-text generation including visual conversation, visual knowledge reasoning, common-sense reasoning, and storytelling.	Can extend its capability on video tasks, offering a cost sensitive alternative for video captioning	As a multimodal model it can work with text-to-text applications.
Advantages	Flexible, fast, easy to implement	Enables visual knowledge reasoning and conversation; achieves state-of-the-art performance on tasks	Simple, low run time	Small, lightweight, high precision
Disadvantages	Limited capability	High complexity, has not been tested in the real-world applications	Unclear on how to control the generated caption and how to perform in-context learning without parameter update	Higher run time

## 2.3 Evaluation

### 2.3.1 Stress-test scenarios

Inspired by Park et al. (2023), stress-tests are undertaken to evaluate model performance on the following scenarios, in the context of assistive technologies for the vision impaired:

1. Image in the dark
2. Image in the rain
3. Multi-object image
4. Motion snapshot
5. Facial emotion image
6. Fooling image (trick the viewer into perceiving non-existent elements)
7. Rotating object
8. Adversarial image (intentionally created to mislead AI system)

### 2.3.2 Performance metrics

The model performance is assessed by (i) qualitative human judgement and (ii) scalable quantitative metrics including Bilingual Evaluation Understudy (BLEU) and Recall Oriented Understudy for Gisting Evaluation (ROUGE). BLEU measures translation quality disregarding grammar whereas ROUGE evaluates summarization, including coherence and grammar (Cui et al., 2018). ROUGE is favoured for its recall focus (i.e., how much the words in the labelled caption appeared in the model generated captions) and lack of conciseness penalty, aligning well with caption evaluation.

## 2.4 Limitations

Due to computational constraints, analysis was conducted on a small sample for learning. Findings may not be representative of model performance, which should be evaluated on appropriate sample size and domain-specific contents.

## 3. Findings

Moondream2 outperformed other models in most stress-testing scenarios, indicating superior caption quality (Table 2 with further details in Appendix 7-8). Additional considerations include the ingeniousness of text, crucial for assisting visually impaired individuals.

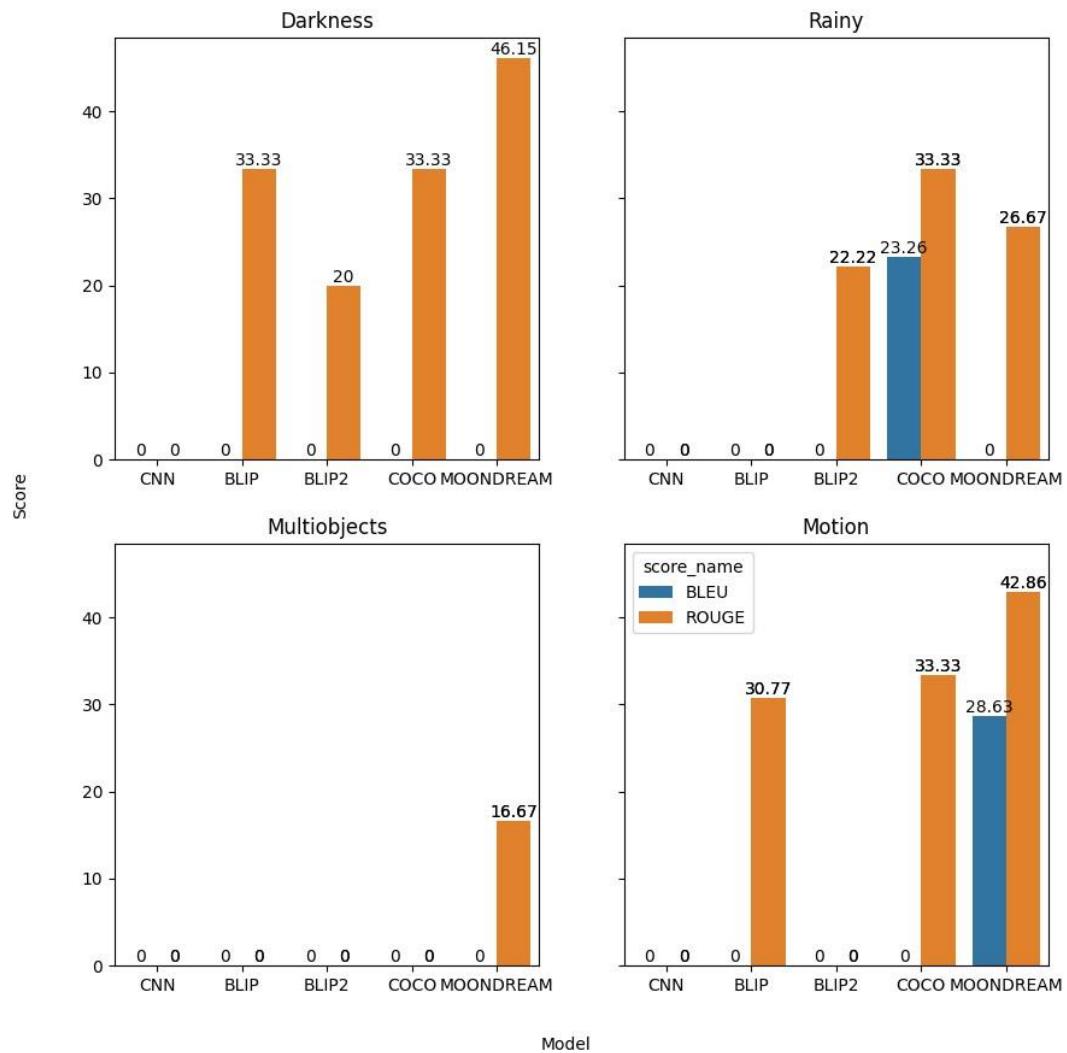
**Table 2.** Image Captioning Result

	<b>1. Darkness</b>	<b>2. Rainy</b>	<b>3. Multi objects</b>	<b>4. Motion</b>
Picture				
Actual Caption	A person eats takeout while watching a small television	A cart containing two men be pull by horse in the rain	A few dogs swim in a lake	A football player in a full stadium jumping and receiving a football
Best generated caption	A person is sitting on the ground with a television in front of them	A couple of horses pulling a cart in the rain	A group of dogs playing in a pool with a yellow stick in their mouths	A football player in a jersey with the number 36 is jumping to catch a football
Model	Moondream2	Git Coco	Moondream2	Moondream2
Justification	Highest Rouge	Highest Rouge & BLEU, most detailed	Highest Rouge, most detailed	Highest Rouge & BLEU, most detailed

	<b>5. Facial Emotion</b>	<b>6. Fooling image</b>	<b>7. Rotation</b>	<b>8. Adversarial</b>
Picture				 <span style="float: right;">Original image</span>  <span style="float: right;">Adversarial image</span>
Actual Caption	A laugh woman in a scarf	A horse mascot give high five to some football fan	A boy in a red top be hang upside down from a tree	Two panda on a tree
Best generated caption	Woman holding a black dog in her arms and laughing	Mascot greets fans at a game	A person hanging upside down from a tree in a park	Two panda bears are playing with a tree branch
Model	BLIP	Git Coco	Moondream2	Moondream2
Justification	Despite a lower Rouge than Moondream2 (20 vs. 30) BLIP is the only model captioning the laughing emotion	Despite a lower Rouge than Moondream2 (20 vs. 27), Git Coco recognizes the mascot	Highest Rouge & BLEU	Highest Rouge & resulting in the same caption for both images

## Group 1

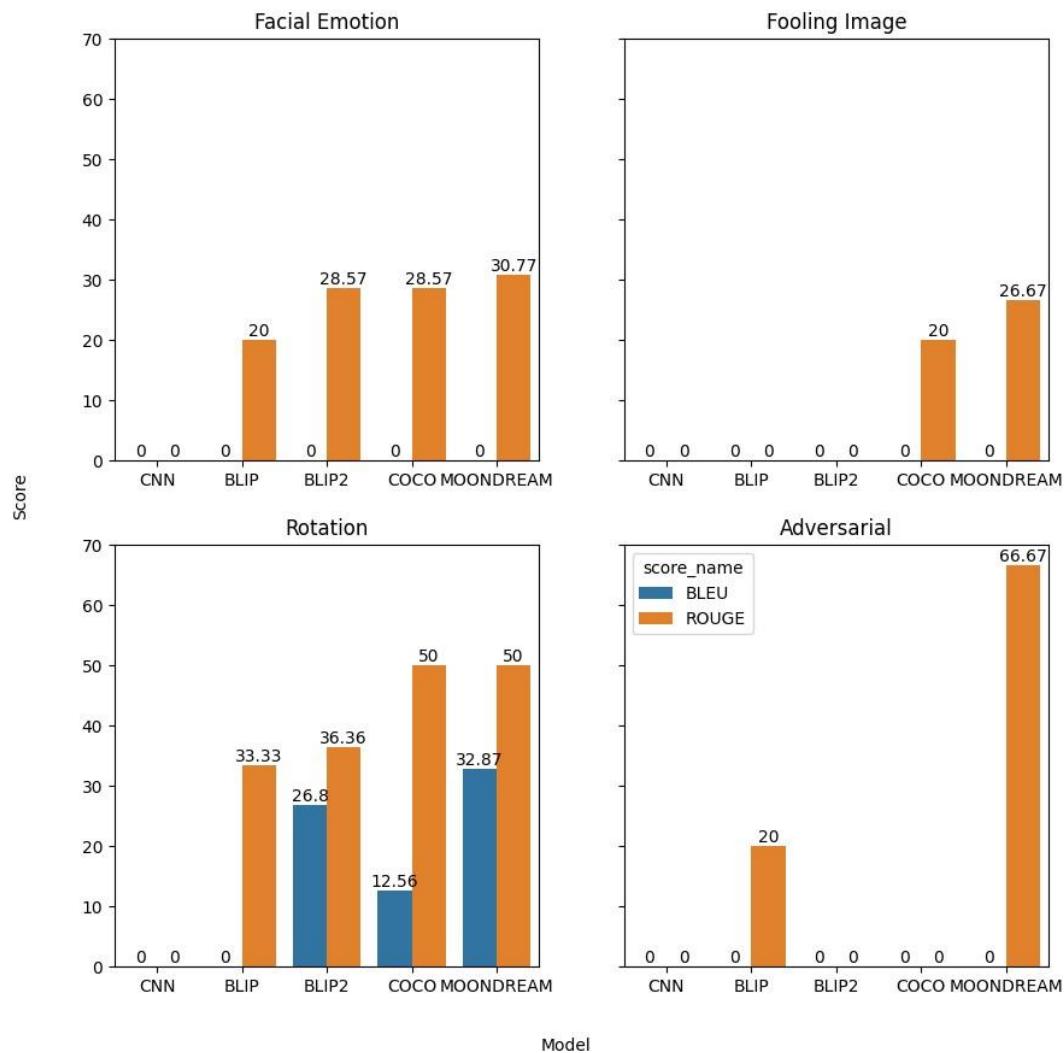
ROUGE and BLEU Results (1)



**Figure 4.** ROUGE and BLEU Result (1)

## Group 1

ROUGE and BLEU Results (2)



**Figure 5.** ROUGE and BLEU Result (2)

## 4. Conclusion

This report highlights the potential of image captioning technologies, especially for aiding the visually impaired. Among the evaluated models, Moondream2 excels in stress tests, demonstrating robust capabilities in realistic scenarios. Whilst the report provides an educational overview, advantages and disadvantages of each image technique and evaluation considerations, limitations in our approach due to computational constraints suggest that future research is required to refine these technologies, expand on evaluation techniques and datasets, and enhancing usability in practical applications.

## 5. References

- Alowais, S. A., Alghamdi, S. S., Alsuhebany, N., Alqahtani, T., Alshaya, A. I., Almhareb, S. N., Aldairem, A., Alrashed, M., Bin Saleh, K. and Badreldin, H. A. (2023). 'Revolutionizing healthcare: the role of artificial intelligence in clinical practice'. *BMC Medical Education*, 23(1), 689. doi: 10.1186/s12909-023-04698-z
- Cui, Y., Yang, G., Veit, A., Huang, X. and Belongie, S. (2018). 'Learning to evaluate image captioning'. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5804-5812. doi: 10.48550/arXiv.1806.06422
- Goyal, S., Chattopadhyay, C. and Bhatnagar, G. (2021). 'Knowledge driven Description Synthesis for Floor Plan Interpretation'. *International Journal on Document Analysis and Recognition (IJDAR)*, 24, pp.19-32. doi: 10.1007/s10032-021-00367-3
- Gunasekar, S., Zhang, Y., Aneja, J., Mendes, C.C.T., Del Giorno, A., Gopi, S., Javaheripi, M., Kauffmann, P., de Rosa, G., Saarikivi, O. and Salim, A. (2023). 'Textbooks are all you need'. doi: 10.48550/arXiv.2306.11644
- Hugging Face. (2023). Choosing a metric for your task. Available at: [https://huggingface.co/docs/evaluate/en/choosing\\_a\\_metric](https://huggingface.co/docs/evaluate/en/choosing_a_metric) (Accessed: 7 May 2024)
- Karthikeyan, B. (2022). Image Caption Generator Using CNN and LSTM. Available at: <https://www.kaggle.com/code/balajikarthikeyan/image-caption-generator-using-cnn-and-lstm> (Accessed 10 May 2024).
- Koch, D., Despotovic, M., Leiber, S., Sakeena, M., Döller, M. and Zeppelzauer, M. (2019). 'Real Estate Image Analysis: A Literature Review'. *Journal of Real Estate Literature*, 27(2), pp.269–300. doi:10.22300/0927-7544.27.2.269
- Magalhães, G. V., Santos, R. L. S., Vogado, L. H. S., Paiva, A. C. de., and Neto, P. de A. dos S. (2024). 'XRayswinGen: Automatic medical reporting for X-ray exams with multimodal model'. *Helijon*, e27516. doi: 10.1016/j.heliyon.2024.e27516
- Michelecafagna26. (2022). CIDEr score. Available at: <https://github.com/michelecafagna26/cider> (Accessed: 11 May 2024)
- Li, Y., Bubeck, S., Eldan, R., Del Giorno, A., Gunasekar, S. and Lee, Y.T. (2023). 'Textbooks Are All You Need II: phi-1.5 technical report' doi: 10.48550/arXiv.2309.05463
- Li, J., Li, D., Xiong, C. and Hoi, S. (2022). 'BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation'. International Conference on Machine Learning (ICML). doi: 10.48550/arXiv:2201.12086.

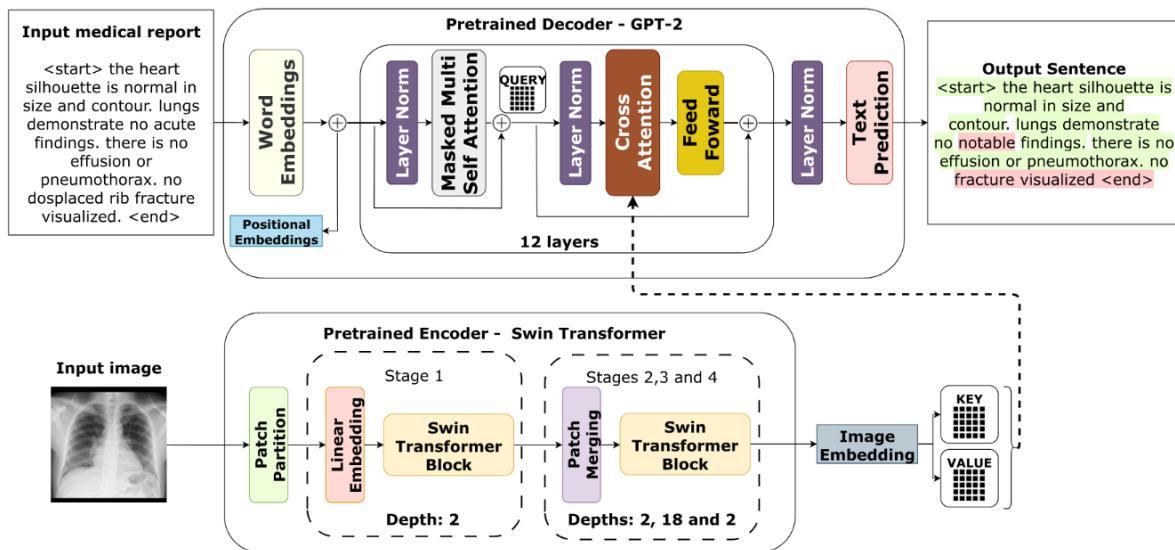
- Li, J., Li, D., Savarese, S. and Hoi, S. (2023). 'BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models'. International Conference on Machine Learning (ICML), 814, pp.19730-19742.doi: 10.48550/arXiv.2301.12597.
- OpenAI. (2023). CLIP (Contrastive Language-Image Pretraining), Predict the most relevant text snippet given an image. Available at: <https://github.com/OpenAI/CLIP> (Accessed: 11 May 2024)
- Park, S., Um, D., Yoon, H., Chun, S., Yun, S., & Choi, J. Y. (2023). RoCOCO: Robustness Benchmark of MS-COCO to Stress-test Image-Text Matching Models. ArXiv Preprint ArXiv:2304.10727.
- Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G. and Sutskever, I. (2021). 'Learning Transferable Visual Models From Natural Language Supervision', Cornell University Library, arXiv.org, Ithaca.
- Rogge, N. (2023). Transformers-Tutorials.  
Available at: <https://github.com/NielsRogge/Transformers-Tutorials/tree/master/BLIP-2> (Accessed: 9 May 2024)
- Salesforce. (2023). BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation. Available at: <https://huggingface.co/Salesforce/blip-image-captioning-large> (Accessed: 9 May 2024)
- Salesforce. (2024). BLIP-2, OPT-2.7b, pre-trained only. Available at: <https://huggingface.co/Salesforce/blip2-opt-2.7b> (Accessed: 9 May 2024)
- Simonyan, K. and Zisserman, A. (2014). 'Very Deep Convolutional Networks for Large-Scale Image Recognition'. International Conference on Learning Representations (ICLR). doi: <https://doi.org/10.48550/arXiv.1409.1556>
- Suno. (2023). Metric Evaluation. Available at: <https://huggingface.co/suno/bark-small> (Accessed: 11 May 2024)
- Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. & Bengio, Y. (2016). 'Show, Attend and Tell: Neural Image Caption Generation with Visual Attention', Cornell University Library, arXiv.org, Ithaca.
- Vikhyatk. (2024). Moondream2. Available at: <https://huggingface.co/vikhyatk/moondream2> (Accessed: 9 May 2024)

- Vikhyatk. (2024). Moondream. Available at: <https://github.com/vikhyat/moondream> (Accessed: 9 May 2024)
- Wang, J., Yang, Z., Hu, X., Li, L., Lin, K., Gan, Z., Liu, Z., Liu, C. and Wang, L. (2022). 'GIT: A Generative Image-to-text Transformer for Vision and Language'. arXiv:2205.14100 [cs]. [online] Available at: <https://arxiv.org/abs/2205.14100>. (Accessed: 10 May 2024)
- Wang, Y., Xu, J. and Sun, Y. (2022). 'End-to-End Transformer Based Model for Image Captioning'. AAAI Conference on Artificial Intelligence, 36(3), pp. 2585–2594. doi: 10.48550/arXiv.2203.15350
- Web Content Accessibility Guidelines 2.1. (2023). W3C World Wide Web Consortium Recommendation. Available at: <https://www.w3.org/TR/WCAG21/> (Accessed: 10 May 2024)
- Zhai, X., Wang, X., Mustafa, B., Steiner, A., Keysers, D., Kolesnikov, A. and Beyer, L. (2022). 'Lit: Zero-Shot Transfer with Locked-Image Text Tuning'. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 18123-18133. doi: 10.1109/CVPR52688.2022.01759
- Zhai, X., Mustafa, B., Kolesnikov, A. and Beyer, L. (2023). 'Sigmoid Loss for Language Image Pre-Training'. IEEE/CVF International Conference on Computer Vision (ICCV), pp. 11941-11952. doi: 10.1109/ICCV51070.2023.01100

## 6. Appendices

### Appendix 1: Other Applications' Frameworks and Architecture

Image-to-text generation technology has found myriad applications in multitude of sectors. Such as, in healthcare, XRaySwinGen, a multimodal model, generates written medical reports from X-ray pictures (Magalhães et al., 2024) (see Figure A.1 for model architecture overview). X-ray textual descriptions can reveal abnormalities, fractures, and other medical issues that are often missed, helping healthcare providers quickly interpret findings and make informed patient care decisions.

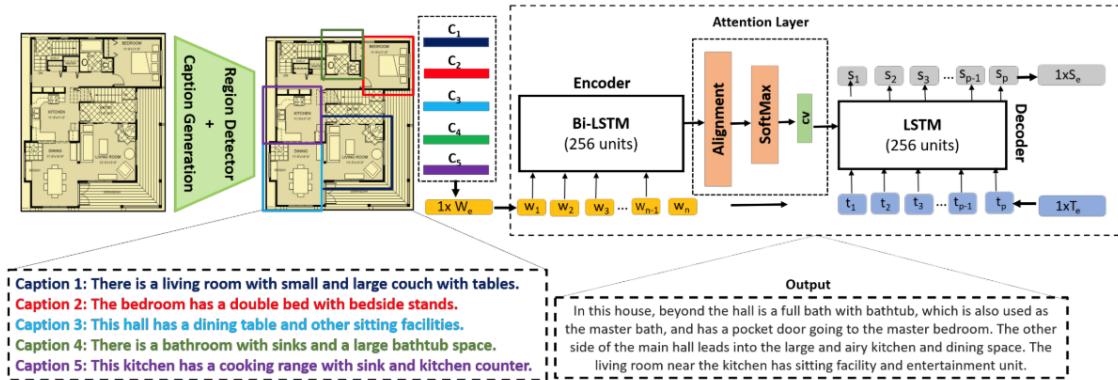


**Figure A.1.** Overview of XRaySwinGen Multimodal Model Architecture

Real estate listings use image-to-text generation to extract useful information from photographs (Koch et al., 2019). Key characteristics like room measurements and features are automatically identified in text. This gives buyers and tenants more information about home features and layouts.

Description Synthesis from Image Cue (DSIC) and Transformer Based Description Generation (TBDG) use contemporary deep neural networks for visual feature extraction to produce textual descriptions from floor plan images (Goyal et al., 2021). (See Figure A.2 for TBDG framework overview).

## Group 1



**Figure A.2.** Overview of TBDG of Generating Paragraph Description from Input Floor Plan Image.

## Appendix 2: Transformer-based Model Overview

**Table A.1.** Transformer-based Model Comparison (Complete Version of Table 1)

Model	BLIP	BLIP-2	Microsoft GIT based COCO	Moondream2
Release/Owner	2022 / Salesforce	2023 / Salesforce	2022 / Microsoft	2024 / Moondream.ai
Differentiator	BLIP improves the quality of pretrained web data by bootstrapping the captions, where a captioner generates synthetic captions and a filter removes noisy image-text pairs (CapFilt). The model then can be trained on a much larger bootstrapped dataset (vs. limited human labelled dataset in other models)	BLIP2 bootstraps vision-language pre-training from off-the-shelf frozen pre-trained image encoders and frozen large language models, significantly reducing the number of trainable parameters and associated computation costs.	GIT simplifies the architecture as one image encoder and one text decoder under a single language modeling task while scaling up the pre-training data and the model size to boost the model performance	Moondream2 implements SigLIP for the image processing evaluating one pair of image/text during training, simplifying the encoding process (Zhai et al., 2023). It also uses Phi-1.5 for decoding, which was trained using textbooks, meaning higher quality texts (Li et al. 2023).
Model Architecture	Multimodal mixture of ViT image encoder, BERT text encoder and BERT text decoder (Figure A.3)	Multimodal mixture of a CLIP-like image encoder, a large language model, and a Querying Transformer (QFormer) acting as an information bottleneck between the frozen image encoder and the frozen LLM, where it feeds the most useful visual feature for the LLM to output the desired text (Figure A.5)	Multimodal mixture of a Swin-like vision transformer image encoder and BERT text decoder (Figure A.6)	Multimodal mixture of SigLIP for encoder, and Phi-1.5 as decoder.
Beyond Image captioning	BLIP can be used to enhance models' ability to understand and respond to questions about images. Moreover, BLIP can aid in translation tasks where the input or output involves both textual and visual information.	BLIP2's instructed zero-shot image-to-text generation capability may have a wide range of application including visual conversation, visual knowledge reasoning, visual common-sense reasoning, and storytelling, adding more nuances to assistive solutions or visually impaired people.	Despite not having video-dedicated encoders, GIT can extend its capability on video tasks, offering a cost sensitive alternative for video captioning	Being a multimodal model, it can work with text-to-text applications as well. However, its primary intended usage is for image descriptions.

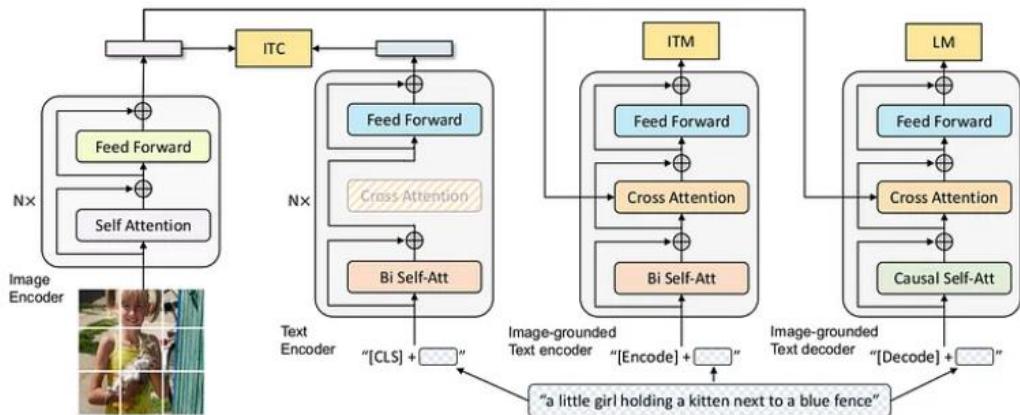
Group 1

Model	BLIP	BLIP-2	Microsoft GIT based COCO	Moondream2
Pre-training data	The bootstrapped dataset from web data and labelled COCO dataset	BLIP with 129M images (including COCO), LAION400M with 115M images (Li et.al., 2023)	GIT was trained on 0.8 billion image-text pairs GIT based was trained on 10 million image-text pairs	SigLIP was trained on LiT dataset, which uses data from YFCC100m and CC12M (Zhai et al. 2022). Phi-1.5 uses training data from Phi-1, consisting in a mix of StackOverflow code, and textbooks to emulate “textbook quality” data (Gunasekar et al. 2023).
Fine-tuning data for image captioning	N/A	COCO	COCO	N/A
Parameters	N/A	188 million	GIT 0.7 billion	1.86 billion
Running time	2 minutes	3 minutes	4 minutes	10 minutes

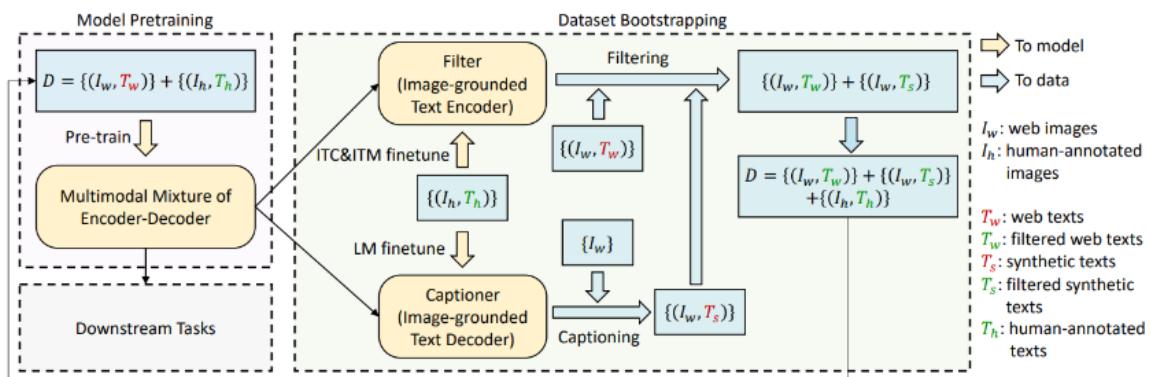
## Appendix 3: BLIP Architecture and Overview

Bootstrapping Language-Image Pre-training (BLIP) is a BERT-based pre-training method, also known as Multimodal mixture of Encoder-Decoder (MED). It facilitates unified vision-language understanding and generation tasks by jointly training on large-scale multimodal datasets, as illustrated in Figure A.3. This model card describes a pre-trained model for image captioning, trained on the COCO dataset. The model architecture is based on a ViT large backbone.

BLIP, also known as Captioning and Filtering (shown in Figure A.4), is flexible, fast (only takes 2 minutes to run the model in the context of this study), and easy to implement, which makes it adaptable to many industries and applications. Its capabilities are restricted compared to newer models.

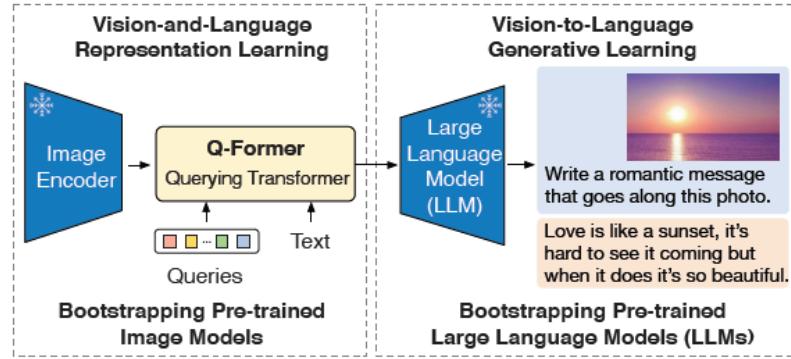


**Figure A.3.** Pre-training Model Architecture and Objectives of BLIP (Li et.al., 2022)



**Figure A.4.** Learning Framework of BLIP (Li et.al., 2022)

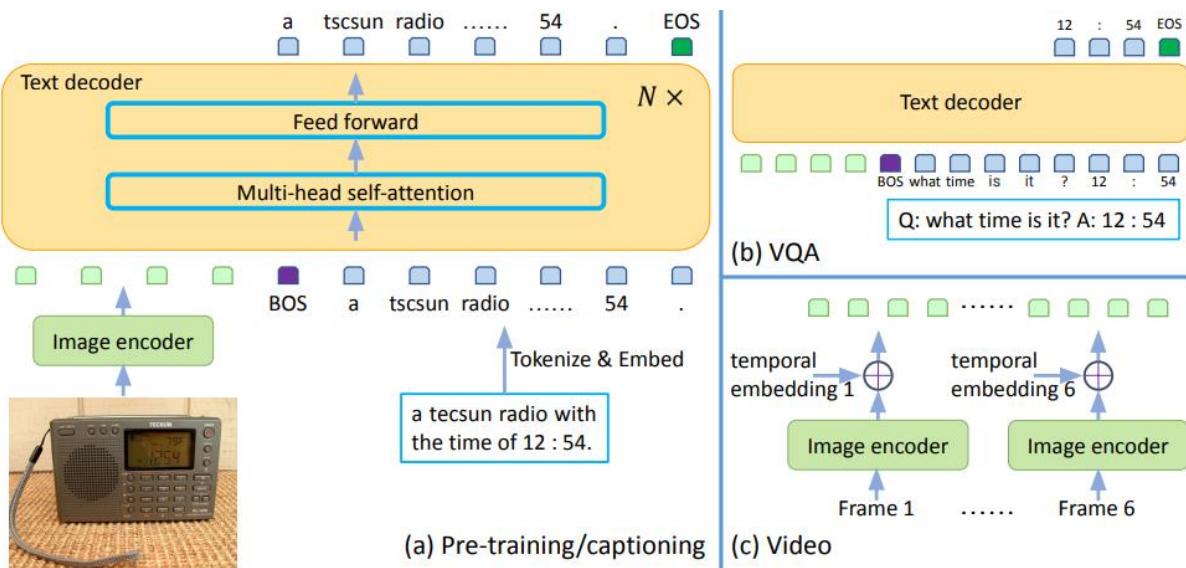
BLIP-2, an improved version of BLIP, helps LLMs process and understand images for zero-shot image-to-text generation. Figure A.5 shows BLIP-2 pre-training using the Querying Transformer in two stages: vision-language representation learning with a frozen image encoder and generative learning with a frozen LLM. This model performs well in visual knowledge reasoning and communication tasks, especially visual question answering. However, BLIP-2 has not been tested in the real-world applications.



**Figure A.5.** Overview of BLIP-2’s Framework (Li et.al., 2023)

## Appendix 4: GIT Overview

It adopts a smaller variant of GIT (Generative Image to Text Transformation) model that was pre-trained on 10 million image-text pairs then finetuned on the COCO dataset. Advantages of GIT is its simple architecture with one image encoder and one text decoder under a single language modelling task, reducing the running time, and proven to be beneficial to handle additional category data. However, as the model focuses on the pretraining-and-finetuning strategy, it is unclear on how to control the generated caption and how to perform in-context learning without parameter update.



**Figure A.6.** Overview of GIT's Framework (Wang et.al., 2022)

## Appendix 5: Moondream2 Overview

Moondream2 is a small lightweight multimodal model that has extensive visual descriptions, including poetic styles. It was released during early 2024, its open-source nature allows for customisation. However, it took 10 minutes to run the same 10 photos, compared to 2-4 minutes for other models. Its major benefit is that it produces better captions than other models despite taking longer to process. Thus, recommended for precision at lower runtimes. Since it was recently introduced, upgrades are frequent, therefore production and implementation require careful attention.

As for its architecture, it was trained on 1.86 billion parameters and uses transformers for encoding and decoding. The image encoder uses SigLIP (Sigmoid loss for Language-Image Pre-training), processing each “image-text” pair independently instead of considering all images at the same time, turning it into a binary classification problem (Zhai et al., 2023).

Decoding uses Microsoft's Phi-1.5, trained on textbooks and “textbook-like” data mixed with internet selected prompts for diversity, resulting in a smaller language model of 1.3 billion parameters, performing similarly to models five times larger (Li et al., 2023).

## Appendix 6: ChatGPT Prompt

ChatGPT 3.5 ▾

N You

```
modify this code:  
# Set Directory  
BASE_DIR = '/content/CNN/input/flickr8k'  
WORKING_DIR = '/CNN/working'
```

to set the base and working directory from this github repo:  
[https://github.com/carlotafg/text\\_analytics\\_group1/](https://github.com/carlotafg/text_analytics_group1/)

ChatGPT

To set the base and working directory from the provided GitHub repository, you need to first clone the repository to your local machine and then specify the appropriate directories in your code. Here's how you can modify the code:

```
python  
  
import os  
  
# Clone the GitHub repository  
!git clone https://github.com/carlotafg/text_analytics_group1.git  
# Change directory to the cloned repository  
os.chdir('text_analytics_group1')  
  
# Set base directory  
BASE_DIR = os.getcwd()  
  
# Set working directory  
WORKING_DIR = os.path.join(BASE_DIR, 'CNN/working')  
  
# Ensure the working directory exists  
os.makedirs(WORKING_DIR, exist_ok=True)
```

Replace '/path/to/text\_analytics\_group1' with the actual path where you cloned the repository on your local machine. This code assumes that within the cloned repository, there is a directory named 'CNN' containing a directory named 'working'. If the directory structure is different, you need to adjust the paths accordingly.

## Group 1

ChatGPT 3.5 ▾

-  You  
the `BASE_DIR` and `WORKING_DIR` will be the same at a folder inside `text_analytics_group1` name `CNN-LSTM`
-  ChatGPT  
If both `'BASE_DIR'` and `'WORKING_DIR'` are the same and located within a folder named `'CNN-LSTM'` inside the cloned repository, you can modify the code accordingly:

```
python  Copy code

import os

# Clone the GitHub repository
!git clone https://github.com/carlotafg/text_analytics_group1.git
# Change directory to the cloned repository
os.chdir('text_analytics_group1')

# Set base directory
BASE_DIR = os.path.join(os.getcwd(), 'CNN-LSTM')

# Set working directory
WORKING_DIR = BASE_DIR

# Ensure the working directory exists
os.makedirs(WORKING_DIR, exist_ok=True)
```

This code will clone the specified GitHub repository into your Colab environment, set both `'BASE_DIR'` and `'WORKING_DIR'` to the same directory within the `'CNN-LSTM'` folder, and ensure that the working directory exists. Adjust the paths if needed based on the actual directory structure.

## Group 1

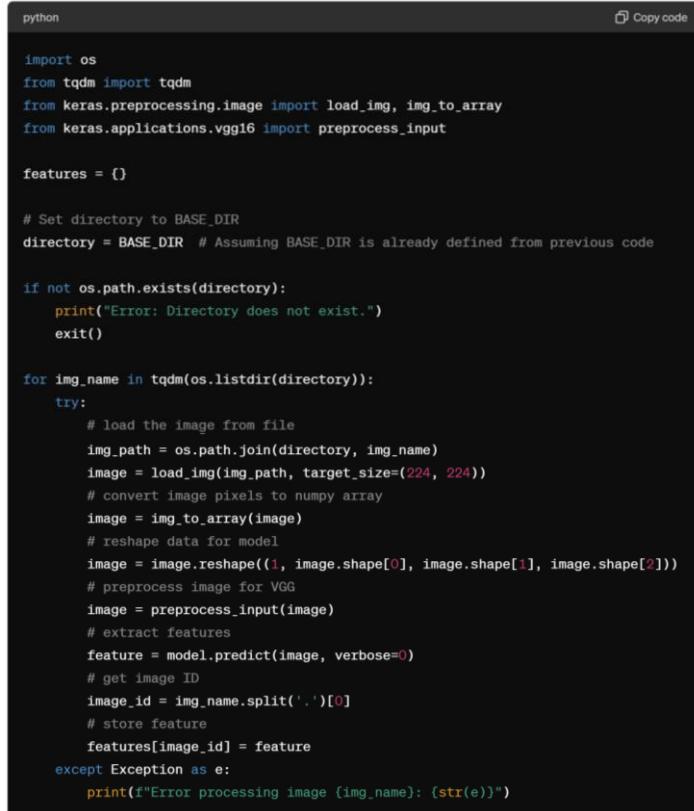
N You

```
modify this code:  
# Extract Features from Image  
import os  
from tqdm import tqdm  
from keras.preprocessing.image import load_img, img_to_array  
from keras.applications.vgg16 import preprocess_input  
  
features = {}  
#directory = '/content/CNN/input/flickr8k/Images'  
directory = '/content/sample_data/best_model.h5'  
  
if not os.path.exists(directory):  
    print("Error: Directory does not exist.")  
    exit()  
  
for img_name in tqdm(os.listdir(directory)):  
    try:  
        # load the image from file  
        img_path = os.path.join(directory, img_name)  
        image = load_img(img_path, target_size=(224, 224))  
        # convert image pixels to numpy array  
        image = img_to_array(image)  
        # reshape data for model  
        image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))  
        # preprocess image for VGG  
        image = preprocess_input(image)  
        # extract features  
        feature = model.predict(image, verbose=0)  
        # get image ID  
        image_id = img_name.split('.')[0]  
        # store feature  
        features[image_id] = feature  
    except Exception as e:  
        print(f"Error processing image {img_name}: {str(e)}")
```

to read the data from the github link before (BASE\_DIR)

ChatGPT

To modify the code to read images from a directory specified by 'BASE\_DIR', you can simply replace the 'directory' variable with 'BASE\_DIR'. Here's the modified code:



The screenshot shows a code editor window with a dark theme. The title bar says "python". The code is identical to the one above, but the line "directory = '/content/CNN/input/flickr8k/Images'" has been replaced by "directory = BASE\_DIR". The code is syntax-highlighted in green and blue.

```
python  
Copy code  
  
import os  
from tqdm import tqdm  
from keras.preprocessing.image import load_img, img_to_array  
from keras.applications.vgg16 import preprocess_input  
  
features = {}  
  
# Set directory to BASE_DIR  
directory = BASE_DIR # Assuming BASE_DIR is already defined from previous code  
  
if not os.path.exists(directory):  
    print("Error: Directory does not exist.")  
    exit()  
  
for img_name in tqdm(os.listdir(directory)):  
    try:  
        # load the image from file  
        img_path = os.path.join(directory, img_name)  
        image = load_img(img_path, target_size=(224, 224))  
        # convert image pixels to numpy array  
        image = img_to_array(image)  
        # reshape data for model  
        image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))  
        # preprocess image for VGG  
        image = preprocess_input(image)  
        # extract features  
        feature = model.predict(image, verbose=0)  
        # get image ID  
        image_id = img_name.split('.')[0]  
        # store feature  
        features[image_id] = feature  
    except Exception as e:  
        print(f"Error processing image {img_name}: {str(e)}")
```

This modification will make the code read images from the directory specified by 'BASE\_DIR', assuming that 'BASE\_DIR' is correctly defined with the desired directory path.

# Group 1

ChatGPT 3.5 ▾

N You

modify this code to only read file with .jpeg format in the BASE\_DIR

ChatGPT

To modify the code to only read files with the ` `.jpeg` format in the ` `BASE\_DIR` ` , you can use the ` `glob` ` module to filter files with the desired extension. Here's the modified code:

```
python
Copy code

import os
from tqdm import tqdm
from keras.preprocessing.image import load_img, img_to_array
from keras.applications.vgg16 import preprocess_input
import glob

features = {}

# Set directory to BASE_DIR
directory = BASE_DIR # Assuming BASE_DIR is already defined from previous code

if not os.path.exists(directory):
    print("Error: Directory does not exist.")
    exit()

# Get all .jpeg files in BASE_DIR
jpeg_files = glob.glob(os.path.join(directory, '*.jpeg'))

for img_path in tqdm(jpeg_files):
    try:
        # load the image from file
        image = load_img(img_path, target_size=(224, 224))
        # convert image pixels to numpy array
        image = img_to_array(image)
        # reshape data for model
        image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
        # preprocess image for VGG
        image = preprocess_input(image)
        # extract features
        feature = model.predict(image, verbose=0)
        # get image ID
        image_id = os.path.splitext(os.path.basename(img_path))[0]
        # store feature
        features[image_id] = feature
    except Exception as e:
        print(f"Error processing image {img_path}: {str(e)}")
```

This modification uses ` `glob.glob()` ` to find all files with the ` `.jpeg` ` extension in the ` `BASE\_DIR` ` directory. Then, it iterates over these files, loading and processing each image as before.

## Group 1

ChatGPT 3.5 ▾

N You  
this code:

```
# Import Modules
import os
import pickle
import numpy as np
from tqdm.notebook import tqdm

from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input # extract features
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Model
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.layers import Input, Dense, LSTM, Embedding, Dropout, add

loops the folder name:
/content/text_analytics_group1/text_analytics_group1/text_analytics_group1/text_analytics_group1/images
```

ChatGPT

It seems like you want to modify the code to avoid looping through the same folder multiple times.  
Here's the corrected code where the directory path is specified only once:

```
python
Copy code

# Import Modules
import os
import pickle
import numpy as np
from tqdm.notebook import tqdm

from tensorflow.keras.applications.vgg16 import VGG16 , preprocess_input # extract fe
from tensorflow.keras.preprocessing.image import load_img , img_to_array
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Model
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.layers import Input , Dense , LSTM , Embedding , Dropout , add

# Specify the folder containing images
images_folder = "/content/text_analytics_group1/text_analytics_group1/text_analytics_group1/images"

# Other code goes here
```

Now, you can use the `images\_folder` variable whenever you need to reference the folder containing the images, ensuring that the directory path is specified only once. Adjust the path as needed based on your actual directory structure.

## Group 1

ChatGPT 3.5 ▾

N You  
modify this code:  
import os  
  
# Clone the GitHub repository  
!git clone https://github.com/carlotafg/text\_analytics\_group1.git  
# Change directory to the cloned repository  
os.chdir('text\_analytics\_group1')  
  
# Set base directory  
BASE\_DIR = os.path.join(os.getcwd(), 'images')  
  
# Set working directory  
WORKING\_DIR = os.path.join(BASE\_DIR, 'cnn-lstm')

the base dir is group1/images and working\_dir should be group1/cnn-lstm not group1/images/cnn\_lstm

ChatGPT

Got it! Here's the modified code with the base directory set to `group1/images` and the working directory set to `group1/cnn-lstm`:

```
python Copy code  
  
import os  
  
# Clone the GitHub repository  
!git clone https://github.com/carlotafg/text_analytics_group1.git  
# Change directory to the cloned repository  
os.chdir('text_analytics_group1')  
  
# Set base directory  
BASE_DIR = os.path.join(os.getcwd(), 'images')  
  
# Set working directory  
WORKING_DIR = os.path.join(os.path.dirname(BASE_DIR), 'cnn-lstm') # Assuming 'cnn-ls  
  
# Ensure the working directory exists  
os.makedirs(WORKING_DIR, exist_ok=True)
```

This code sets `BASE\_DIR` to `group1/images` and `WORKING\_DIR` to `group1/cnn-lstm`. Additionally, it ensures that the working directory is created if it doesn't already exist.

## Group 1

ChatGPT 3.5 ▾

N You  
modify this code:  
# Clone the GitHub repository  
!git clone https://github.com/carloatafg/text\_analytics\_group1.git  
# Change directory to the cloned repository  
os.chdir('text\_analytics\_group1')  
  
# Set base directory  
BASE\_DIR = os.path.join(os.getcwd(), 'images')  
  
# Set working directory  
WORKING\_DIR = os.path.join(os.path.dirname(BASE\_DIR), 'cnn\_lstm') # Assuming 'cnn-lstm'  
is desired, not 'cnn\_lstm'  
  
so if i run it again, it will check whether the github clone already exist in the colab folder, if so,  
don't clone it again, or just overwrite the existing

ChatGPT

To modify the code so that it checks whether the GitHub repository has already been cloned in the Colab environment and either clones it if it doesn't exist or updates it if it does, you can use the following approach:

```
python
Copy code

import os

# Check if the GitHub repository already exists in the current directory
if not os.path.exists('text_analytics_group1'):
    # If it doesn't exist, clone the GitHub repository
    !git clone https://github.com/carloatafg/text_analytics_group1.git
else:
    # If it exists, change directory to the cloned repository and update it
    %cd text_analytics_group1
    !git pull origin master # Update the repository

# Set base directory
BASE_DIR = os.path.join(os.getcwd(), 'images')

# Set working directory
WORKING_DIR = os.path.join(os.path.dirname(BASE_DIR), 'cnn_lstm') # Assuming 'cnn-lstm'

# Ensure the working directory exists
os.makedirs(WORKING_DIR, exist_ok=True)
```

This modified code first checks whether the `text\_analytics\_group1` directory exists in the current directory. If it doesn't exist, it clones the GitHub repository. If it does exist, it changes the directory to `text\_analytics\_group1` and updates the repository using `git pull`. This ensures that the repository is up to date. Then, it sets the base directory and working directory as before.

## Group 1

### N You

this code still duplicating the folder inside group1 if i run it multiple times:  
import os

```
# Check if the text_analytics_group1 directory exists in the current directory
if not os.path.exists('text_analytics_group1'):
    # If it doesn't exist, clone the GitHub repository
    !git clone https://github.com/carlotafg/text_analytics_group1.git
else:
    # If it exists, change directory to text_analytics_group1 and update it
    %cd text_analytics_group1
    !git pull origin master # Update the repository
    %cd .. # Move back to the original directory

# Set base directory
BASE_DIR = os.path.join(os.getcwd(), 'text_analytics_group1', 'images')

# Set working directory
WORKING_DIR = os.path.join(os.getcwd(), 'text_analytics_group1', 'cnn_lstm') # Assuming
'cnn_lstm' is desired, not 'cnn-lstm'

# Ensure the working directory exists
os.makedirs(WORKING_DIR, exist_ok=True) ↓
```

### ChatGPT

I see, to prevent the duplication of nesting inside the `text\_analytics\_group1` directory, we need to adjust the logic. Here's the modified code:

```
python
Copy code

import os

# Check if the text_analytics_group1 directory exists in the current directory
if not os.path.exists('text_analytics_group1'):
    # If it doesn't exist, clone the GitHub repository
    !git clone https://github.com/carlotafg/text_analytics_group1.git
else:
    # If it exists, change directory to text_analytics_group1 and update it
    %cd text_analytics_group1
    !git pull origin master # Update the repository
    %cd .. # Move back to the original directory

# Set base directory
BASE_DIR = os.path.join(os.getcwd(), 'text_analytics_group1', 'images')

# Set working directory
WORKING_DIR = os.path.join(os.getcwd(), 'text_analytics_group1', 'cnn_lstm') # Assum

# Ensure the working directory exists
os.makedirs(WORKING_DIR, exist_ok=True)
```

This modification ensures that after cloning or updating the repository, the working directory is set to the parent directory of `text\_analytics\_group1`, so no further nesting occurs within the `text\_analytics\_group1` directory. Then, the base directory and working directory are set accordingly.

### N You

this code still duplicating the folder inside group1 if i run it multiple times:  
import os

```
# Check if the text_analytics_group1 directory exists in the current directory
if not os.path.exists('text_analytics_group1'):
    # If it doesn't exist, clone the GitHub repository
    !git clone https://github.com/carlotafg/text_analytics_group1.git
else:
    # If it exists, change directory to text_analytics_group1 and update it
    %cd text_analytics_group1
    !git pull origin master # Update the repository
    %cd .. # Move back to the original directory ↓
```

## Group 1

### ChatGPT

I see the issue now. To avoid the duplication of folders inside `text\_analytics\_group1`, we need to ensure that the `git clone` command clones the repository directly into the current directory rather than creating another `text\_analytics\_group1` directory within it. Let's modify the code to achieve this:

```
python
import os

# Check if the text_analytics_group1 directory exists in the current directory
if not os.path.exists('text_analytics_group1'):
    # If it doesn't exist, clone the GitHub repository
    !git clone https://github.com/carlotafg/text_analytics_group1.git
    # Rename the cloned directory to 'text_analytics_group1'
    !mv text_analytics_group1/* .
    !mv text_analytics_group1/. .
    !rm -rf text_analytics_group1
else:
    # If it exists, change directory to text_analytics_group1 and update it
    %cd text_analytics_group1
    !git pull origin master # Update the repository
    %cd .. # Move back to the origin directory

# Ensure the working directory exists
os.makedirs(WORKING_DIR, exist_ok=True)
```

This modified code first checks whether the `text\_analytics\_group1` directory exists in the current directory. If it doesn't exist, it clones the GitHub repository. If it does exist, it changes the directory to `text\_analytics\_group1` and updates the repository using `git pull`. This ensures that the repository is up to date. Then, it sets the base directory and working directory as before.

### You

it keeps duplicating nesting inside the text\_analytic\_group1

### ChatGPT

I see, to prevent the duplication of nesting inside the `text\_analytics\_group1` directory, we need to adjust the logic. Here's the modified code:

```
python
import os

# Check if the text_analytics_group1 directory exists in the current directory
if not os.path.exists('text_analytics_group1'):
    %cd .. # Move back to the original directory

# Set base directory
BASE_DIR = os.path.join(os.getcwd(), 'images')

# Set working directory
WORKING_DIR = os.path.join(os.getcwd(), 'cnn_lstm') # Assuming 'cnn_lstm' is desired

# Ensure the working directory exists
os.makedirs(WORKING_DIR, exist_ok=True)
```

This modified code renames the cloned directory to `text\_analytics\_group1` after cloning. This ensures that there's no additional nesting when running the code multiple times. Then, the base directory and working directory are set accordingly.



## Group 1

ChatGPT 3.5 ▾

N You

fatal: destination path 'text\_analytics\_group1' already exists and is not an empty directory.

ChatGPT

I see. If the directory already exists and you're trying to clone into it, it will throw an error because Git expects an empty directory for cloning. To handle this, you can first check if the directory is empty before cloning or pull the repository. Here's how you can do it:

```
python
import os

# Check if the text_analytics_group1 directory exists in the current directory
if not os.path.exists('text_analytics_group1'):
    # If it doesn't exist, clone the GitHub repository
    !git clone https://github.com/carlotafg/text_analytics_group1.git
else:
    # If it exists, change directory to text_analytics_group1 and update it
    %cd text_analytics_group1
    # Check if the directory is empty
    if os.listdir(): # If not empty
        print("Directory is not empty. Pulling updates...")
        !git pull origin master # Update the repository
    else:
        # If it exists, change directory to text_analytics_group1 and update it
        %cd text_analytics_group1
        # Check if the directory is empty
        if os.listdir(): # If not empty
            print("Directory is not empty. Pulling updates...")
            !git pull origin master # Update the repository
        else:
            print("Directory is empty. Cloning repository...")
            %cd ..
            !rm -rf text_analytics_group1
            !git clone https://github.com/carlotafg/text_analytics_group1.git
            %cd text_analytics_group1

    %cd .. # Move back to the original directory

# Set base directory
BASE_DIR = os.path.join(os.getcwd(), 'text_analytics_group1', 'images')

# Set working directory
WORKING_DIR = os.path.join(os.getcwd(), 'text_analytics_group1', 'cnn_lstm') # Assume

# Ensure the working directory exists
os.makedirs(WORKING_DIR, exist_ok=True)
```

## Group 1

ChatGPT 3.5 ▾

This modification first checks if the directory is empty. If it's not empty, it pulls updates from the repository. If it's empty, it clones the repository. This should prevent the fatal error you encountered.

N You

```
/content/text_analytics_group1  
Directory is not empty. Pulling updates...  
fatal: couldn't find remote ref master  
[Errno 2] No such file or directory: '.. # Move back to the original directory'  
/content/text_analytics_group1
```

⌚ ChatGPT

It seems the error occurs because the 'master' branch might not exist in the repository. Let's adjust the code to handle this situation more gracefully. We'll also address the issue with moving back to the original directory. Here's the updated code:

```
python Copy code  
  
import os  
  
# Check if the text_analytics_group1 directory exists in the current directory  
if not os.path.exists('text_analytics_group1'):  
    # If it doesn't exist, clone the GitHub repository  
    !git clone https://github.com/carlotafg/text_analytics_group1.git  
else:  
    # If it exists, change directory to text_analytics_group1 and update it  
    %cd text_analytics_group1  
    # Check if the directory is empty  
    if os.listdir(): # If not empty  
        print("Directory is not empty. Pulling updates...")  
        # Attempt to pull updates  
        !git pull origin main # Assuming 'main' is the main branch name  
    else:  
        print("Directory is empty. Cloning repository...")  
        %cd ..  
        !rm -rf text_analytics_group1  
        !git clone https://github.com/carlotafg/text_analytics_group1.git  
        %cd text_analytics_group1  
  
    # Move back to the original directory  
    %cd ..  
  
    # Set base directory  
    BASE_DIR = os.path.join(os.getcwd(), 'text_analytics_group1', 'images')  
  
    # Set working directory  
    WORKING_DIR = os.path.join(os.getcwd(), 'text_analytics_group1', 'cnn_lstm') # Assume  
  
    # Ensure the working directory exists  
    os.makedirs(WORKING_DIR, exist_ok=True)
```

In this version, I've replaced 'master' with 'main' assuming 'main' is the name of the main branch in the repository. Additionally, I removed the unnecessary comment causing the error related to moving back to the original directory. This should resolve the issues you encountered.

## Group 1

ChatGPT 3.5 ▾

N You

```
i have best_model.h5 but i can't open it:  
from keras.models import load_model  
  
# Path to the saved model  
model_path = WORKING_DIR + '/best_model.h5'  
  
# Load the model  
loaded_model = load_model(model_path)
```

```
-----  
OSErr... Traceback (most recent call last)  
<ipython-input-55-04b27fc3ee4b> in <cell line: 7>()  
    5  
    6 # Load the model  
----> 7 loaded_model = load_model(model_path)  
  
3 frames  
/usr/local/lib/python3.10/dist-packages/h5py/_hl/files.py in make_fid(name, mode,  
userblock_size, fapl, fcpl, swmr)  
    229     if swmr and swmr_support:  
    230         flags |= h5f.ACC_SWMR_READ  
--> 231     fid = h5f.open(name, flags, fapl=fapl)  
    232     elif mode == 'r+':  
    233         fid = h5f.open(name, h5f.ACC_RDWR, ↓, pl=fapl)  
h5py/_objects.pyx in h5py._objects.with_phil.wrapper()  
  
h5py/_objects.pyx in h5py._objects.with_phil.wrapper()  
  
h5py/h5f.pyx in h5py.h5f.open()  
  
OSErr: Unable to open file (truncated file: eof = 3145728, sblock->base_addr = 0, stored_eof  
= 71972196)
```

ChatGPT

The error you're encountering suggests that the HDF5 file you're trying to open might be corrupted or incomplete. The message "truncated file" indicates that the file ends abruptly, which could be due to incomplete saving or other issues.

Here are a few things you can try to troubleshoot and resolve the issue:

1. **Check File Integrity:** Verify that the file `best\_model.h5` is intact and not corrupted. You can try opening it with other tools or viewing its content to ensure it's not corrupted.
2. **Re-save the Model:** If you have access to the original source code that generated `best\_model.h5`, try re-saving the model file. Ensure that the saving process completes successfully without any errors. ↓
3. **Check File Size:** Verify the file size of `best\_model.h5`. It should be consistent with the size you expect based on the model's architecture and parameters. If the file size is significantly smaller than expected, it might indicate corruption.
4. **Try Different Versions:** If you have multiple versions of `best\_model.h5` saved, try loading a different version to see if the issue persists. This can help determine if the problem is specific to one file or a broader issue.
5. **Use File Recovery Tools:** If you suspect that the file might be corrupted, you can try using file recovery tools to attempt to recover the data. However, success with this approach can vary depending on the extent of corruption.
6. **Re-train the Model:** If none of the above solutions work and you're unable to recover the model file, you may need to re-train the model from scratch using your original dataset and code.

By following these steps, you should be able to diagnose and resolve the issue with loading `best\_model.h5`.

## Group 1

ChatGPT 3.5 ▾

N You

```
from PIL import Image
import matplotlib.pyplot as plt
def generate_caption(image_name):
    # load the image
    # image_name = "image1.jpg"
    image_id = image_name.split('.')[0]
    img_path = os.path.join(BASE_DIR, image_name)
    image = Image.open(img_path)
    captions = mapping[image_id]
    print('-----Actual-----')
    for caption in captions:
        print(caption)
    # predict the caption
    y_pred = predict_caption(model, features[image_id], tokenizer, max_length)
    print('-----Predicted-----')
    print(y_pred)
    plt.imshow(image)

generate_caption("test.jpg")
-----
KeyError          Traceback (most recent call last)
<ipython-input-64-df9a41fc9f07> in <cell line: 1>()
----> 1 generate_caption("test.jpg")

<ipython-input-51-2c135fab74ef> in generate_caption(image_name)
      7     img_path = os.path.join(BASE_DIR, image_name)
      8     image = Image.open(img_path)      ↓
----> 9     captions = mapping[image_id]
     10    print('-----Actual-----')
     11    for caption in captions:

KeyError: 'test'
```

ChatGPT

The error you're encountering (`'KeyError: 'test'`) indicates that the `'image_id'` extracted from the image name `"test.jpg"` is not present in the `'mapping'` dictionary. This KeyError occurs because there is no corresponding key `'test'` in the dictionary.

To fix this issue, you can add error handling to check if the `'image_id'` exists in the `'mapping'` dictionary before trying to access it. Here's how you can modify the `'generate_caption'` function to handle this:

```
python
Copy code

def generate_caption(image_name):
    # load the image
    image_id = image_name.split('.')[0]
    img_path = os.path.join(BASE_DIR, image_name)
    image = Image.open(img_path)

    # Check if image_id exists in the mapping dictionary
    if image_id in mapping:
        captions = mapping[image_id]
        print('-----Actual-----')
        for caption in captions:
            print(caption)
        # Predict the caption
        y_pred = predict_caption(model, features[image_id], tokenizer, max_length)
        print('-----Predicted-----')
        print(y_pred)
        plt.imshow(image)
    else:
        print(f"No captions found for image: {image_name}")
```

With this modification, if the `'image_id'` is not found in the `'mapping'` dictionary, it will print a message indicating that no captions were found for that image. This should prevent the KeyError from occurring and provide a more graceful handling of the situation when an image is not found in the mapping.

## Appendix 7: Model Captions &amp; Qualitative Judgment

**Table A.2.** Image Captioning Result (Complete Version of Table 2)

	<b>1. Darkness</b>	<b>2. Rainy</b>	<b>3. Multi objects</b>	<b>4. Motion</b>
Picture				
Actual Caption	A person eats takeout while watching a small television	A cart containing two men be pull by horse in the rain	A few dogs swim in a lake	A football player in a full stadium jumping and receiving a football
CNN-LSTM based	startseq of throw the boy costume in in in...	startseq climb white air as hats in in in...	startseq purple boy running woman while the on red in in in in...	startseq dresses runs on tikes' dresses runs blue bench in in in...
BLIP	someone is sitting on the ground watching a television with a cat on it	horses pulling a carriage with people in it on a rainy day	dogs playing in a pool of water with a ball in it	a rafed football player catching a ball while another player tries to catch it
BLIP-2	person sitting on the ground	a man and his dog	a group of dogs in a pool	a man in a yellow and white uniform
GIT COCO	a person sitting on the ground watching tv	a couple of horses pulling a cart in the rain.	a group of dogs swimming in a pond.	american football player catches a touchdown pass over american football player.
Moondream2	A person is sitting on the ground with a television in front of them.	A horse pulling a carriage with two horses and a man in a red hat.	A group of dogs playing in a pool with a yellow stick in their mouths.	A football player in a jersey with the number 36 is jumping to catch a football.

Group 1

	<b>5. Facial Emotion</b>	<b>6. Fooling image</b>	<b>7. Rotation</b>	<b>8. Adversarial</b>
Picture				
Actual Caption	A laugh woman in a scarf	A horse mascot give high five to some football fan	A boy in a red top be hang upside down from a tree	Two panda on a tree
CNN-LSTM based	startseq tries tries tries of on boy girl man boy with has the on child in in in...	startseq tattoos wrestling in in in in car out in in in...	startseq grass to the camera down and while man children jump in the four in in in...	startseq boy with team standing in dog boy with team standing in in in.../startseq boy with team is dogs below in large dogs dog walking with in in in
BLIP	woman holding a black dog in her arms and laughing	araffe dressed in red and white holding a tennis racket	araffed girl upside down on a tree branch in a park	panda bear eating bamboo in zoo enclosure with tree branch in foreground / pandas eating bamboo in a zoo enclosure with a tree branch
BLIP-2	a woman holding a dog	a man in a red shirt	a young woman hanging upside down from a tree	two pandas eating bamboo / two pandas eating bamboo
GIT COCO	a woman with a dog in her mouth.	mascot greets fans at a game.	a boy in a tree	pandas playing with a branch / two pandas are playing with a branch.
Moondream2	A woman with a dog in her arms and a shirt that says Stop Bitching Stars.	A man in a red shirt with the number 1 on it is shaking hands with a mascot.	A person hanging upside down from a tree in a park.	Two panda bears are playing with a tree branch / Two panda bears are playing with a tree branch.

Group 1

Appendix 8: Model Quantitative Evaluation Scores (BLEU & ROUGE)

	<b>1. Darkness</b>	<b>2. Rainy</b>	<b>3. Multi objects</b>	<b>4. Motion</b>	<b>5. Facial Emotion</b>	<b>6. Fooling image</b>	<b>7. Rotation</b>	<b>8A. Original</b>	<b>8B. Adversarial</b>
<b>BLEU score</b>									
CNN-LSTM based	0	0	0	0	0	0	0	0	0
BLIP	0	0	0	0	0	0	0	0	0
BLIP-2	0	0	0	0	0	0	26.8	0	0
GIT COCO	0	23.26	0	0	0	0	12.56	0	0
Moondream2	0	0	0	28.63	0	0	32.87	0	0
<b>ROUGE score</b>									
CNN-LSTM based	0	0	0	0	0	0	0	0	0
BLIP	33.33	0	0	30.77	20	0	33.33	33.33	20
BLIP-2	20	22.22	0	0	28.57	0	36.36	0	0
GIT COCO	33.33	33.33	0	33.33	28.57	20	50	0	0
Moondream2	46.15	26.67	16.67	42.86	30.77	26.67	50	66.67	66.67