Deep Learning Assignment 2 CS60010 Spring Semester 2025

Task: Automatic Image Captioning with Model Benchmarking and Robustness Analysis

In this assignment, you will develop a **custom encoder-decoder model** for **Automatic Image Captioning** and evaluate its robustness under image perturbations. Your goal is to:

1. **Implement and train** a transformer-based encoder-decoder model for caption generation.
2. **Benchmark** its performance against an off-the-shelf **Small Vision-Language Model (SmolVLM)** in **zero-shot settings**.
3. **Analyze robustness** by evaluating the model's performance on images with various levels of occlusion.
4. **Build a classifier** to distinguish between outputs from SmolVLM and your custom model based on captions and perturbation levels.

Deadline: 14th April, 2025 EOD

Dataset:
https://drive.google.com/file/d/1FMVcFM78XZE1KE1rlkGBpCdcdI58S1LB/view?usp=sharing

Image Captioning Task Description:
Given an input image, generate a caption for the image.

Input:

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

Evaluation metrics: ROUGE-L, METEOR, ~~BertScore~~, BLEU

Task Details:

Part A: Implementing and Benchmarking a Custom Encoder-Decoder Model

1. **Zero-shot Evaluation using SmolVLM**
    - Use the SmolVLM model (without any training) to generate captions for the test set.
    - Report BLEU, ROUGE-L, and METEOR scores as a baseline.
    - Reference: https://huggingface.co/blog/smolvlm
    - Hint: Use _attn_implementation="eager" if "flash_attention_2" does not work.
2. **Develop and Train a Transformer-based Encoder-Decoder Model**
    - Use a **Vision Transformer (ViT)** as the image encoder. You may use **ViT-Small-Patch16-224** (vit-small-patch16-224).
    - Select a suitable text decoder (such as GPT-2, or some small LLM-based transformer decoder).
    - Train your model using the provided dataset, ensuring it fits within the **15GB GPU memory limit** (compatible with Google Colab Free Tier GPUs, e.g., T4 GPU).
    - Optimize hyperparameters to achieve captioning performance **comparable to or better than** the zero-shot performance of SmolVLM.
    - **Bonus Marks** will be awarded if your model outperforms SmolVLM.
3. **Inference & Performance Reporting**
    - Evaluate your trained model on the test set and compare results with SmolVLM.
    - Provide a detailed analysis of your model's performance against SmolVLM.

Part B: Studying Performance Change Under Image Occlusion

1. **Image Perturbation via Occlusion**
    - Divide each input image into **16x16 patches**.
    - Randomly select a percentage (10%, 50%, 80%) of patches and **mask** them by setting pixel values to black in the selected patches. You have to do this for all 3 settings.
2. **Evaluate Model Robustness**
    - Run inference on both **SmolVLM (zero-shot)** and **your custom trained model** using the occluded images.
    - Record performance changes in BLEU, ROUGE-L, and METEOR scores.
        - **BLEU (change) = BLEU (after masking) - BLEU (before masking)**
        - Similarly compute changes for ROUGE-L and METEOR across different perturbation levels (10%, 50%, 80%).

    - Save the **original captions, generated captions, and perturbation levels** for further analysis in Part C.

Part C: Building a BERT-based Classifier for Model Identification

1. **Create a Training Dataset**
    - Using data from Part B, construct a dataset with the format:
        - **Input Text**: `<original_caption> <SEP> <generated_caption> <SEP> <perturbation_percentage>`
        - **Output Labels**: Model A (SmolVLM) or Model B (Custom)

Example:

```
The image shows a boy falling from a banyan tree in a park <SEP> The
image shows a banyan tree <SEP> 50
```

2. **Train a BERT-base based Classification Model**
    - Use a **pretrained BERT base model** (https://huggingface.co/google-bert/bert-base-uncased) as an encoder.
    - Add **custom linear layers** for **binary classification** (predicting whether the output came from SmolVLM or the custom model).
    - Train the model using a custom train-validation-test split (70:10:20) from the data from Part B. Make sure the split is with respect to the images (there should be no overlap of images between the splits).
    - Fine-tune hyperparameters for best classification accuracy.
3. **Evaluate Classifier Performance**
    - Report **macro precision, recall, and F1 scores** on the test set.

Compulsory classes and methods for each part:

Your notebooks must necessarily contain the following classes and functions for each part. You may add some extra classes/functions to support this structure, if needed.

Class/Function Name	**Description**	**Arguments**	**Return**
Part A			
zero_shot_captioning (function)	Generate captions using the pre-trained SmolVLM model without training.	- image_path (str): Path to the input image. - model_name (str):	Returns the generated caption

		Name of the pre-trained model (default: SmolVLM).	
ImageCaptionModel (model class)	Custom Encoder-Decoder Model for Image Captioning using ViT as an encoder. It must contain a constructor and a forward() function.	-	forward() method should return a single tensor containing the model predictions.
train_model (function)	Train the encoder-decoder model.	- model (nn.Module): Custom image captioning model. - dataloader (DataLoader): Training data loader. - optimizer: Optimizer (e.g., Adam). - criterion (Loss): Loss function. - device (str): Device to use ('cuda' or 'cpu'). - epochs (int): Number of epochs.	None
evaluate_model (function)	Evaluate model performance using BLEU, ROUGE-L, METEOR.	- model (nn.Module): Trained model. - dataloader (DataLoader): Test data loader. - device (str): 'cuda' or 'cpu'.	dict: BLEU, ROUGE-L, METEOR scores for the test set.
Part B			
occlude_image (function)	Apply patch-wise occlusion to an image.	- image (np.array): Input image. - mask_percentage (int): Percentage of image to be masked.	np.array: Occluded image.
evaluate_on_occluded_images (function)	Evaluate performance after occluding images.	- model (nn.Module): Image captioning model. - dataloader	dict: BLEU, ROUGE-L, METEOR scores for the test set.

		(DataLoader): Test dataloader. - device (str): 'cuda' or 'cpu'. - occlusion_levels (list): List of occlusion percentages.	
Part C			
CaptionClassifier (model class)	BERT-based classifier for identifying which model generated a caption. It must contain a constructor and a forward() function.	-	forward() method must return the model output logits.
train_classifier (function)	Train the BERT-based caption classifier.	- model (nn.Module): Custom image captioning model. - dataloader (DataLoader): Training data loader. - optimizer: Optimizer (e.g., Adam). - criterion (Loss): Loss function. - device (str): Device to use ('cuda' or 'cpu'). - epochs (int): Number of epochs.	None
evaluate_classifier (function)	Evaluate the classification model.	- model (nn.Module): Trained model. - dataloader (DataLoader): Test data loader. - device (str): 'cuda' or 'cpu'.	dict: Precision, Recall and F1 scores for the test set.

Deliverables:

- **Python notebooks for Part A, Part B and Part C:** Make sure the outputs are saved and the evaluation results are printed in the output cell after implementation. You may choose to make separate notebooks for each part, in which case, include all the files that need to be uploaded in each notebook (except the main dataset). For example, if Part C is implemented in a separate notebook, include the file containing the generated captions from Part B (which is needed in Part C). *At most, there should be 3 python notebook files submitted (1 for each part). You are free to combine all of the code in 1 or 2 files as well.*

- **README file:** containing running instructions (or any other additional info). Also include names, email ids and roll numbers of ALL team members.
- **Project Report:** containing the following sections (max pages: 4):
    - Methodology for Part A and Part C (with explanations and diagrams for each custom model)
    - Results for Parts A, B and C (with table containing eval metrics for test set) and Analysis

General Guidelines:

- You can use Colab or Kaggle for GPU.
- You cannot use end-to-end trained models available on huggingface or github as the custom model for Part A. Your notebooks must clearly contain the model class, showing the design of the model.
- Pre trained encoders or decoders may be used for Part A, but they must be fine tuned on the train set.
- You may not use any train set other than the one given for finetuning/training your models.
- Your notebooks must contain the compulsory methods for each Part for the work to be considered for grading. Deviation from the given code structure may result in penalties or the entire section being discarded from grading.
- You are allowed to use either pytorch or tensorflow.
- Before each section of the code in the notebook, write a brief description in a text cell.
- You may use libraries/boilerplate code for calculating the evaluation metrics.
- You are compulsorily required to show the output of your model on the test set, as output in the ipython notebook.
- You may use https://app.diagrams.net/ to make the diagrams for your report.
- Remember to fine tune your hyperparameters!

Submission Guidelines:

- All the deliverables are to be submitted in a single zip file, with the naming format team_id_<num>*_project.zip. (Eg: team*_id_1_project.zip for Team 1)
- The notebooks are to be named team_id_<num>_a.ipynb, team_id_1_b.ipynb and team_id_<num>_c.ipynb respectively (Eg: team_id_1_a.ipynb, team_id_1_b.ipynb, team_id_1_c.ipynb for Team 1).
- Report should be named team_id_<num>_report.pdf.
- There should be one submission per team.

IMPORTANT: PLEASE FOLLOW THE NAMING CONVENTIONS STRICTLY. FAILURE TO DO SO WILL RESULT IN DEDUCTION OF MARKS.

Plagiarism: We will be employing strict plagiarism checking. If your code matches with another team's code, all those students will be awarded zero marks for the assignment. Therefore, please ensure there is no sharing of code. Model design may be coincidentally the same. **Unfair usage of LLMs (like ChatGPT, Gemini, Claude, etc) to complete the assignment will also be heavily penalized.**

Code Error: If the code doesn't run for a particular experiment, partial marks may be awarded based on structure and logic of code. If required, you may be contacted by the TAs to explain your code.

Additional Resources:

Part A (Image captioning model)

1) Github repo on an example model
https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning
2) Image Captioning Coding Tutorial Video
https://www.youtube.com/watch?v=y2BaTt1fxJU&list=PLCJHEFznK8ZybO3cpfWf4gKbyS5VZgppW&index=2
3) Kaggle notebook
https://www.kaggle.com/code/saeedghamshadzai/image-caption-generator-transformers-nlp

Part C (Building BERT Classifier)

1) Blog on building a simple classifier using BERT
https://medium.com/@khang.pham.exxact/text-classification-with-bert-7afaacc5e49b#:~:text=The%20classifier%20is%20built%20on,the%20extra%20layers%20we%20added.