TAMU CSCE 625 (Spring 2025)
Assignment #4

1 Starting Point

Scripts: From the GitHub repository, you will find:

- Latex template, which you will be amending by adding your results and answers.
- Scripts for the image captioning task using Transformers.
- Scripts for the image classification task using Transformers.

Data: Please see the corresponding questions and links for data download.

Submission: Please strictly follow the instruction for the submission.

- Please use the provided template only. Submission must be written in LaTeX. All submissions not adhering to the template will not be graded and receive a zero.
- Please submit all the `.py` files. Add all relevant plots and text answers in the boxes provided in this file. To include plots you can simply modify the already provided latex code. Submit the complied `.pdf` report as well.

  Note: Partial points will be given for implementing parts of the homework even if you don't get the mentioned accuracy as long as you include partial results in this pdf.
- a `collaboration.txt` that lists with whom you have discussed the homework.

  Collaboration: You may discuss your homework with your classmates. However, you need to write your solutions and submit them separately. In your submission, you need to list with whom you have discussed the homework in a `.txt` file `collaboration.txt`. Please list each classmate's name and NetID as a row in the .txt file (e.g., Cheng Zhang, chzhang). That is, if you discussed the homework with two classmates, your `.txt` file will have two rows. If you did not discuss your homework with your classmates, just write "no discussion" in `collaboration.txt`. Please consult the syllabus for what is and is not an acceptable collaboration.
- Please zip all files (`.py`, `.pdf`, `.txt`) into one package and upload via Canvas.

2 Image Captioning with Transformers (70 points)

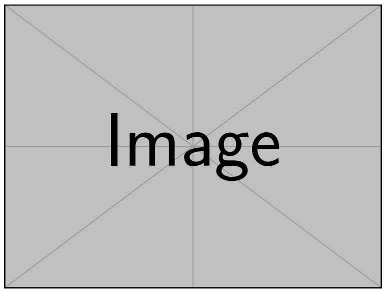
We will be implementing the different pieces of a Transformer decoder (Transformers), and train it for image captioning on a subset of the COCO dataset.

- **Setup:** Run the following command to extract COCO data, in the `transformer_captioning/datasets` folder : `./get_coco_captioning.sh`
- **Question:** Follow the instructions in the `README.md` file in the `transformer_captioning` folder to complete the implementation of the transformer decoder.
- **Deliverables:** After implementing all parts, use run.py for training the full model. The code will log plots to `plots`. Extract plots and paste them into the appropriate section below.

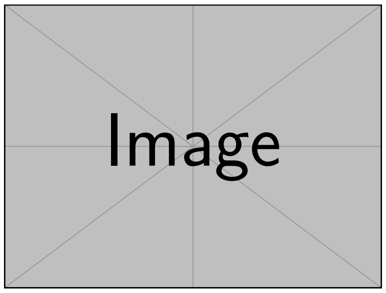
- **Expected results:** These are expected training losses after 100 epochs. Do not change the seed in run.py.
  - 2-heads, 2-layers, lr 1e-4: Final loss ≤ 1
  - 4-heads, 6-layers, lr 1e-4: Final loss ≤ 0.3
  - 4-heads, 6-layers, lr 1e-3: Final loss ≤ 0.05

1. Paste training loss plots for each of the three hyper-param configs

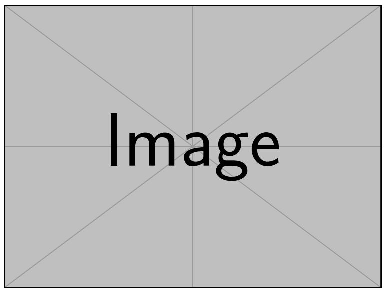
   2-heads-2-layers-lr-1e-4: **TODO: fill in final train loss here.**
   4-heads-6-layers-lr-1e-4: **TODO: fill in final train loss here.**
   4-heads-6-layers-lr-1e-3: **TODO: fill in final train loss here.**



(a) 2-heads-2-layers-lre-4

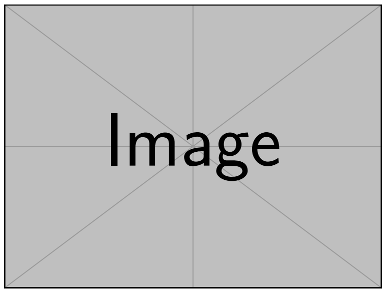


(b) 4-heads-6-layers-lre-4

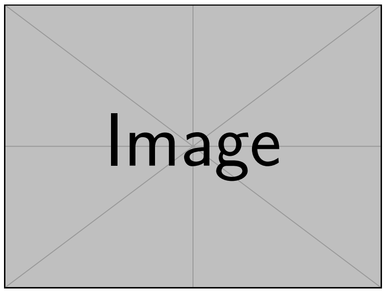


(c) 4-heads-6-layers-lre-3

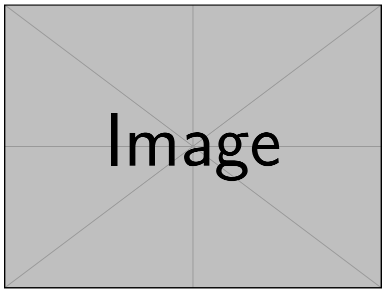
2. Paste any three generated captioning samples from the training set with the three different settings. The provided code creates these plots at the end of training.



(a) Sample1: 2-heads-2-layers-lre-4



(b) Sample2:4-heads-6-layers-lre-4



(c) Sample3:4-heads-6-layers-lre-3

3. Based on the observations of the three different settings, What would you change in the training procedure to get better validation performance? Why tweaking these hyper-parameters will lead to better performances?

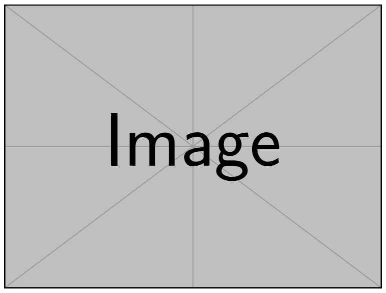
   Solution: TODO: fill in your answers here.

3 Classification with Vision Transformers (30 points)

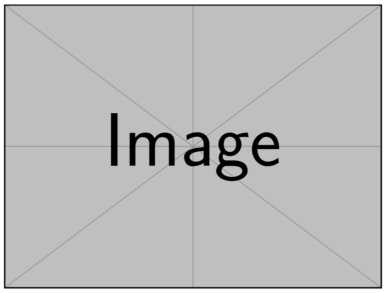
We will use the transformer you implemented in the previous part to implement a Vision Transformer (ViT), for classification on CIFAR10.

- **Question:** Follow the instructions in the `README.md` file in the `vit_classification` folder. You are encouraged to resuse code from the previous question.
- **Deliverables:** Run training using `run.py` for training the full model. The code will log plots `acc_out.png` (train and test accuracy) and `loss_out.png` (train loss).
- **Expected Results:** After 100 epochs, test accuracy should be $\geq 65\%$, train accuracy should be $\approx 100\%$, and training loss ≤ 0.3.

1. Test Accuracy: **TODO: Fill in the accuracy here.**
2. Train Accuracy: **TODO: Fill in the accuracy here.**
3. Training Loss: **TODO: Fill in the loss here.**



(a) Train/test accuracy



(b) Training loss