START HERE: Instructions

Submitting your work:

- **Deliverables:** 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 compiled `.pdf` report as well.

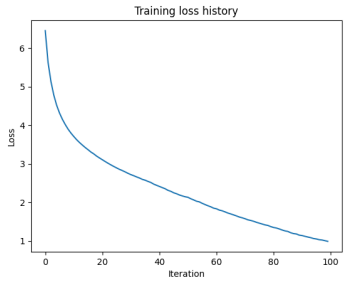
NOTE: Partial points will be given for implementing parts of the homework even if you don't get the mentioned numbers as long as you include partial results in this pdf.

1 Image Captioning with Transformers (70 points)

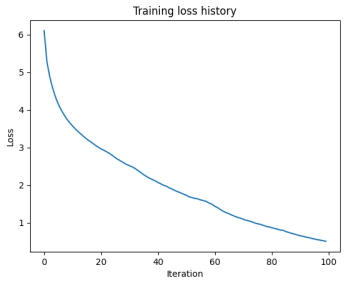
We will be implementing the different pieces of a Transformer decoder (Transformers), and train it for image captioning on a subset of the COCO dataset.

- **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

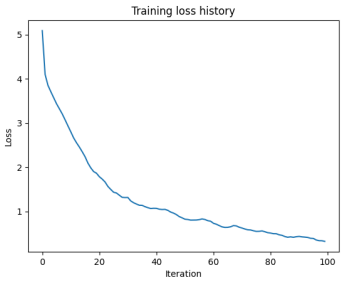
1. Paste training loss plots for each of the three hyper-param configs
   2-heads-2-layers-lr-1e-4: **TODO: fill in final train loss here.**
   4-heads-6-layers-lr-1e-4: **TODO: fill in final train loss here.**
   4-heads-6-layers-lr-1e-3: **TODO: fill in final train loss here.**



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

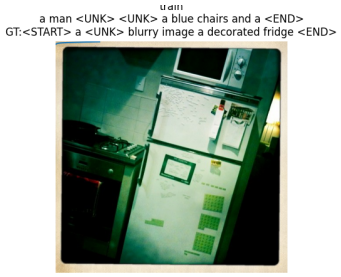


(b) 4-heads-6-layers-lre-4

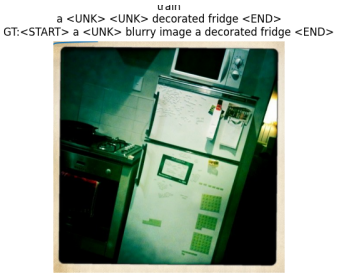


(c) 4-heads-6-layers-lre-3

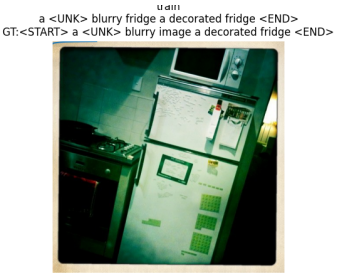
2. Paste any three generated captioning samples from the training set with the three different settings. The provided code creates these plots at the end of training.



(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:

- Learning Rate

  Learning rate describes the step size when getting the mininum of the loss function.If the learning rate is too low the training process will be slow and may get stuck in a local minimum.And if it is too high, it may cause the model to oscillate around,as the loss history picture of model with learing rate = 1e-3 shows that the loss is oscillating when converge to the mininum.

- number of layers

  A big number of layer can improve the Representational Capacity of the model, and may do more complicated works,and as a result it will also make the model more complex,which will raise the trainging time and training difficulty,and it may require more computation resource and trainging data.

- number of attention heads

  More attention heads lead to a better information acquisition from the trainging data,but will also consume more computaton resource because every attention head need to calculate its attention function and generate more temperorary results.

- Optimizer

  The use of different optimizers have various impacts on the trainging speed and performance of the model

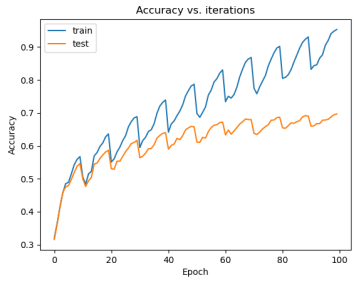
- Batch size

  Larger batch sizes typically provide more stable gradient estimates,and may improve the model performance.However, if the batch size if too large it may lead to overfitting.

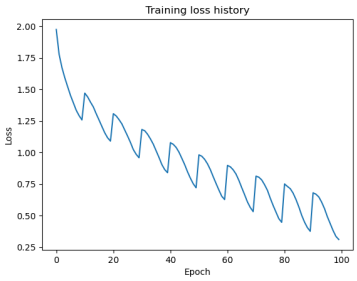
2 Classification with Vision Transformers (30 points)

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.



(a) Train/test accuracy



(b) Training loss

Collaboration Survey Please answer the following:

1. Did you receive any help whatsoever from anyone in solving this assignment?

   ◯ Yes

   √ **No**

   - If you answered 'Yes', give full details:
   - (e.g. "Jane Doe explained to me what is asked in Question 3.4")

2. Did you give any help whatsoever to anyone in solving this assignment?

   ◯ Yes

   √ **No**

   - If you answered 'Yes', give full details:
   - (e.g. "I pointed Joe Smith to section 2.3 since he didn't know how to proceed with Question 2")

3. Note that copying code or writeup even from a collaborator or anywhere on the internet violates the Academic Integrity Code of Conduct.