HOMEWORK 3

16824 VISUAL LEARNING AND RECOGNITION (FALL 2025)
`https://piazza.com/cmu/fall2025/16824/home`

RELEASED: Fri, 24th Oct 2025
DUE: 11:59 PM ET, Thu, 13th Nov 2025
Instructor: Jun-Yan Zhu
TAs: Ananya Bal, Zhixuan Liu, Eungyeup Kim, Jay Karhade

START HERE: Instructions

- **Collaboration policy:** All are encouraged to work together BUT you must do your own work (code and write up). If you work with someone, please include their name in your write-up and cite any code that has been discussed. If we find highly identical write-ups or code or lack of proper accreditation of collaborators, we will take action according to strict university policies. See the Academic Integrity Section detailed in the initial lecture for more information.
- **Late Submission Policy:** There are a **total of 5** late days across all homework submissions. Submissions that use additional late days will incur a 10% penalty per late day.
- **Submitting your work:**
  - We will be using Gradescope (`https://gradescope.com/`) to submit the Problem Sets. Please use the provided template only. You do **not** need any additional packages and using them is **strongly discouraged**. Submissions must be written in LaTeX. All submissions not adhering to the template will not be graded and receive a zero.
  - **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.

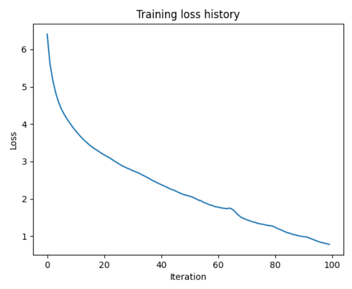
NOTE: Partial points will be given for implementing parts of the homework even if you don't get the mentioned accuracy as long as you include partial results in this pdf.

1 Image Captioning with Transformers (75 points)

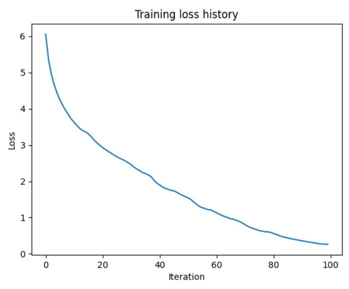
We will be implementing the different pieces of a Transformer decoder (Transformers), and train it for image captioning on a subset of the COCO dataset.

- **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

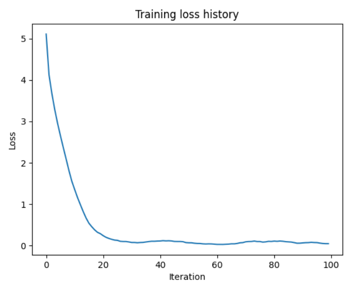
1. Paste training loss plots for each of the three hyper-param configs
   2-heads-2-layers-lr-1e-4: **loss: 0.778242**
   4-heads-6-layers-lr-1e-4: **loss: 0.258864**
   4-heads-6-layers-lr-1e-3: **loss: 0.049734**



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

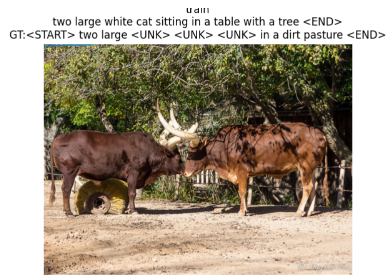


(b) 4-heads-6-layers-lre-4

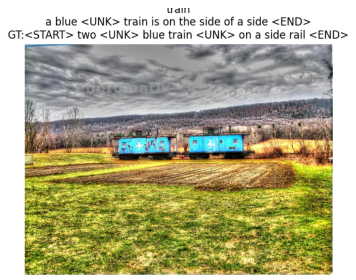


(c) 4-heads-6-layers-lre-3

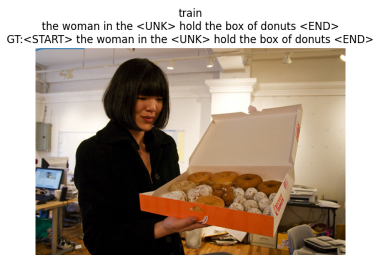
2. Paste any three generated captioning samples from the training set with the three different settings. The provided code creates these plots at the end of training.


two large white cat sitting in a table with a tree <END>
GT:<START> two large <UNK> <UNK> <UNK> in a dirt pasture <END>

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


a blue <UNK> train is on the side of a side <END>
GT:<START> two <UNK> blue train <UNK> on a side rail <END>

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

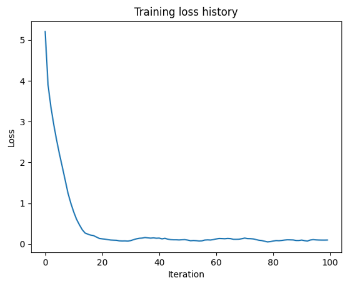

train
the woman in the <UNK> hold the box of donuts <END>
GT:<START> the woman in the <UNK> hold the box of donuts <END>

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

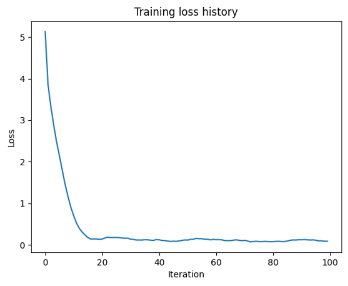
3. Based on the observations of the three different settings, What would you change in the training procedure to get better validation performance? Why tweaking these hyper-parameters will lead to better performances?

Solution:
Based on the above plots, we can see that the 4-heads-6-layers-lr-1e-3 configuration rapidly converges to a lower loss than the first two experiments, but fails to consistently converge well afterward. To fix this, focusing only on the hyperparameter part of the training, I would follow two main improvements: [1] Decrease the learning rate with an increased number of epochs and a learning rate scheduler. 1e-3 is clearly too aggressive as you can see in the oscillations. One could possibly add a warmup in this stage if the lr is low enough, but with so little data, I believe that is not required. [2] With the mentioned overfitting, I would attempt to alleviate overfitting with the following: I would include a weight decay in the optimizer as well as increased dropout and an increased batch size. These should sufficiently include regularization to improve training on this dataset though precise values would need experimentation to find the right fit.

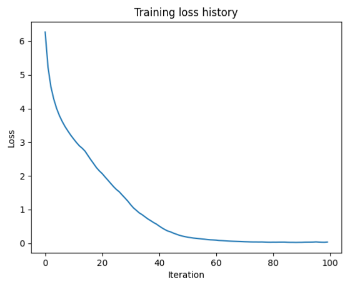
4. Experiment by replacing the activations in your transformer implementation with more modern activations such as SwiGLU. Paste the training loss plot below. Then describe the loss function used and performance changes observed.



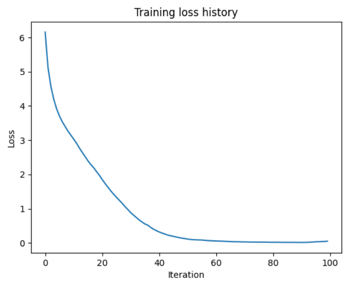
(a) Sample: SwiGLU training



(b) Sample: SinGLU training



(a) Sample: SwiGLU LR: 1.7e-4



(b) Sample: SinGLU LR: 1.7e-4

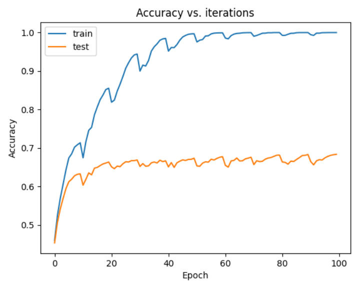
Solution:
I attempted both SwiGLU as requested and a supposed improvement over SwiGLU called SinGLU, a Sin-based GLU activation function proposed in this paper: "Improving Vision Transformers with

Novel GLU-Type Nonlinearities". I also did this across the first final learning rate above of 1e-3 and a smaller, but still slightly larger learning rate of 1.7e-4 to benefit from some slightly faster convergence than 1e-4. As you can see, the Gated Linear Unit activation functions prefer a lower learning rate to provide improvements over RELU. As for the loss, I did not make a change from Cross-Entropy loss as we did not change the expected output here, and the activation function change does not affect that.
SwiGLU training loss: **loss: 0.096604**, SinGLU training loss: **loss: 0.092521**
SwiGLU LR: 1.7e-4 training loss: **loss: 0.034074**, SinGLU LR: 1.7e-4 training loss: **loss: 0.052016**, (This one actually got down to a loss of 0.015 at epoch 90 but jumped back up at the end to lose to SwiGLU, so a scheduler and some finetuning would be interesting.)

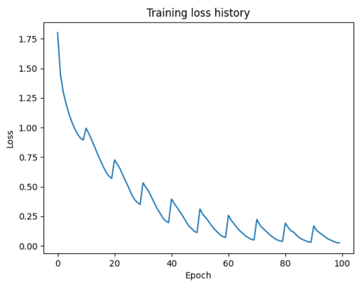
2 Classification with Vision Transformers (25 points)

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 around 65%, train accuracy should be $\approx 100\%$, and training loss ≤ 0.3.



(a) Train/test accuracy



(b) Training loss

Solution:
SwiGLU Implementation Test Accuracy: **0.683700**

Train Accuracy: **1.000000 [Achieved from epoch 97 onward]**

Training Loss: **loss: 0.025288**

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.