CSC 245/445: Deep Learning **Fall 2025**

Project 2

Handed Out: September 27, 2025 *Due: October 17, 2025 at 11:59pm*

- Feel free to talk to other members of the class in doing the homework. I am more concerned that you learn how to solve the problem than that you demonstrate that you solved it entirely on your own. You should, however, **write down your solution yourself**. Please include at the top of your document the list of people you consulted with in the course of working on the homework. We do not have a restriction on utilizing AI for coding. If you utilized AI tools for coding or answering, please list it out.
- While we encourage discussion within and outside the class, **cheating and copying code is strictly not allowed**. Copied code will result in the entire assignment being discarded at the very least.
- Please use Blackboard if you have questions about the homework. Also, please come to the office hours.
- We STRONGLY encourage you to type and prepare your answers in LaTeX. Hand-written answers must be clear and concise or else will be graded as incorrect answers. If you don't have a lot of experience with Latex (or even if you do), we recommend using Overleaf (`https://www.overleaf.com`) to write your solutions. You will submit your solutions as a single pdf file (in addition to the package with your code; see instructions in the body of the assignment).
- The homework is due at 11:59 PM on the due date. We will be using Blackboard for collecting the homework assignments. You should have been automatically added to Blackboard. If not, please ask a TA for assistance. Please do **not** hand in a hard copy of your write-up. Post on Blackboard and contact the TAs if you are having technical difficulties in submitting the assignment.

1 Overview

In this assignment, you will experiment with transfer learning for image classification and implement a modern transformer-based language model from scratch. This project consists of two parts that provide hands-on experience with contemporary deep learning techniques in computer vision and natural language processing.

Part 1 focuses on transfer learning using the Oxford Pet Dataset. You will fine-tune pre-trained models (ResNet-50 and Swin Transformer) using two strategies: feature extraction (frozen backbone) and full fine-tuning. The implementation includes data preprocessing, training loops, and comprehensive performance analysis comparing different architectures and training approaches.

Part 2 requires implementing RocLM, a decoder-only transformer model with modern architectural components. You will build Root Mean Square Normalization (RMSNorm), SwiGLU-activated MLPs, and Multi-Head Attention with Grouped Query Attention (GQA).

2 Models

This section describes the models implemented in the assignment, beginning with computer vision architectures for Part 1 and then outlining the RocLM language model for Part 2.

2.1 Part 1: Computer Vision Models

The first part employs two established architectures: ResNet-50 and the Swin Transformer. ResNet-50 is a 50-layer convolutional network originally pre-trained on ImageNet, adapted here by replacing its final classification layer with a 37-class output. The Swin Transformer, a vision transformer that computes attention within shifted windows for improved efficiency, is similarly adapted for the same classification task.

Both models are trained under two regimes. In the feature extraction setting, the pre-trained backbone remains frozen and only the final layer is trained. In contrast, full fine-tuning updates all parameters, using carefully chosen learning rates to adapt the entire model to the dataset.

2.2 Part 2: RocLM Language Model Architecture

The second part introduces RocLM, a decoder-only transformer model that incorporates several modern components to improve both efficiency and expressiveness.

2.2.1 Overall Design

RocLM follows the general structure of a transformer decoder. Token and positional embeddings feed into stacked layers that apply normalization, multi-head attention, and feed-forward networks in the sequence

$$\text{RMSNorm} \;\rightarrow\; \text{Attention} \;\rightarrow\; \text{RMSNorm} \;\rightarrow\; \text{MLP}.$$

Residual connections link the sublayers, and a final normalization precedes the language modeling head.

2.2.2 Root Mean Square Normalization

Instead of LayerNorm, RocLM employs Root Mean Square Normalization (RMSNorm), which scales activations by their root mean square without centering. For a hidden vector $x \in \mathbb{R}^d$,

$$\text{RMSNorm}(x) = \frac{x}{\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2 + \epsilon}} \cdot \gamma,$$

where ϵ ensures stability and γ is a learnable parameter. This reduces computational cost while maintaining stable training.

2.2.3 MLP with SwiGLU

The feed-forward block uses SwiGLU, a gated variant of Swish. Given an input x,

$$\text{SwiGLU}(x) = \text{Swish}(\text{gate\_proj}(x)) \odot \text{up\_proj}(x), \quad \text{MLP}(x) = \text{down\_proj}(\text{SwiGLU}(x)).$$

This gated structure improves gradient flow and representational capacity compared to standard ReLU-based MLPs.

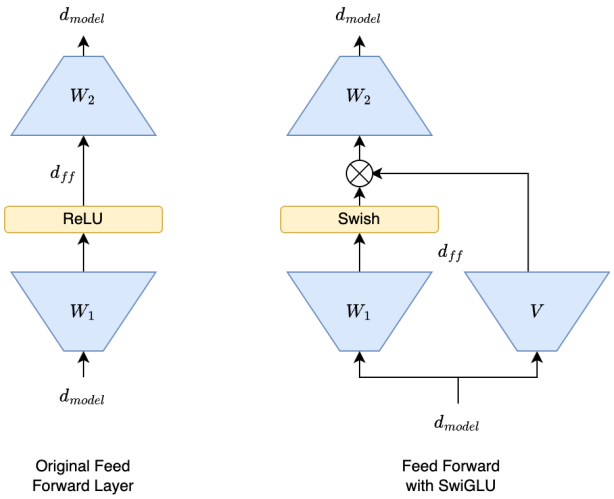

Figure 1: SwiGLU feed-forward architecture compared with original Feedforward Layer in the transformer.

2.2.4 Grouped Query Attention

Attention in RocLM adopts Grouped Query Attention (GQA), which reduces the number of key–value heads relative to query heads. Multiple queries thus share the same key–value pairs, lowering memory cost while preserving accuracy. After projection, queries and keys are enriched with Rotary Position Embeddings, and attention is computed as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V.$$

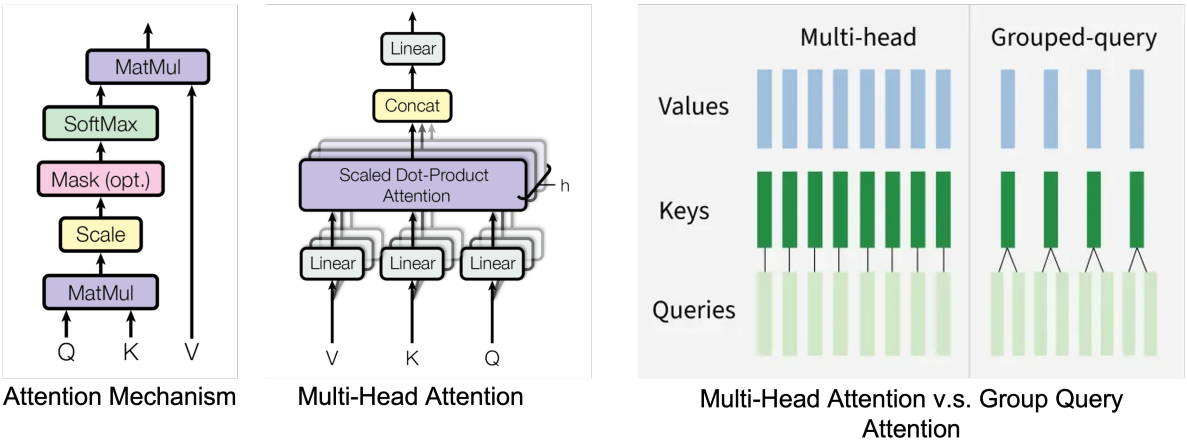

Figure 2: Grouped Query Attention compared with standard multi-head attention (Image source: "GQA: Training Generalized Multi-Query Transformer Models" paper).

2.2.5 Rotary Position Embeddings

Positional information is encoded with RoPE, which rotates query and key vectors according to their position index m:

$$\begin{pmatrix} q_m \\ q_{m+1} \end{pmatrix} = \begin{pmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{pmatrix} \begin{pmatrix} q_m \\ q_{m+1} \end{pmatrix},$$

with $\theta = 10000^{-2i/d}$. RoPE encodes relative positions and generalizes well to longer sequences.

2.2.6 Causal Masking

Autoregressive generation is enforced with causal masking, which blocks tokens from attending to future positions:

$$\text{mask}_{i,j} = \begin{cases} 0 & j \leq i, \\ -\infty & j > i. \end{cases}$$

2.3 Forward Pass

The RocLM forward pass begins with token and positional embeddings, followed by stacked layers applying RMSNorm, GQA attention with residual connections, and SwiGLU-based MLPs. After the final normalization, the language modeling head produces logits. By combining RMSNorm, SwiGLU, GQA, and RoPE, RocLM achieves improved efficiency and modeling capacity over traditional transformer designs.

3 Experiments

3.1 Evaluation Metrics and Reporting

The evaluation of Part 1 is based primarily on both quantitative and qualitative measures. Quantitatively, we report classification accuracy on the validation set, as well as per-class precision, recall, and F1-scores to capture performance across the 37 categories. In addition, we monitor training dynamics through convergence speed, measured in epochs to convergence, and computational efficiency, recorded as average training time per epoch. Complementing these metrics, qualitative analysis focuses on examining error patterns to identify systematic weaknesses, interpreting architectural differences between ResNet-50 and the Swin Transformer, and discussing the effectiveness of transfer learning strategies.

Part 2 evaluation emphasizes the correctness of the RocLM implementation. At the implementation level, correctness is assessed through the mathematical accuracy of RMSNorm, the proper design of the SwiGLU gating mechanism in the MLP, and the completeness of the grouped query attention mechanism. Successful integration of these components into the overall model architecture is required. Functionality is further demonstrated by verifying that the model loads and runs without errors, that text generation produces coherent and reasonable outputs, and that the interactive demo operates as intended. Finally, the tokenizer lab must be completed, including the required analysis linked to the student ID.

3.2 Submission Requirements

The submission for Part 1 must include a complete implementation of all modules within the `part1/` directory, along with generated plots, analytical results in the `part1/results/` folder, and experimental logs and checkpoints that demonstrate successful training.

For Part 2, the deliverables consist of a completed `roclm_model.py` with all three core tasks implemented, the finished `Tokenizer_Lab.ipynb` notebook containing the required analysis, and demonstration files that confirm the RocLM model can be executed. For the demo, you should run it multiple times and analyze the difference between using thinking mode and without using thinking mode. Also you need to write the analysis of what types of things it is good at, what types of questions it is not good at. Please write your findings and paste the conversations between you and RocLM in the submitted latex/pdf file.

3.3 Submission Instructions

We will use Blackboard to turn in the Python code and writeup pdfs. You should have been automatically added to Blackboard. If you do not have access and are not sure about the provided source files, please ask the TA staff on Blackboard.

After you finish the assignment, please upload it as a zip file to Balckabord before the deadline which contains the following:

- "latex": This is the assignment where you should upload your write-up as a PDF.
- "evaluation" This folder includes the evaluation output of the experiments; the files in this folder should be automatically generated by the code.
- "Code": This is the folder where you should upload all your code.