Fall 2025 | DATA 266 | Homework -4
Deadline – 11.59 PM – Dec 6th, 2025
20 Points

Complete RLHF Pipeline — Reward Modeling, PPO Fine-Tuning, and LoRA Comparison

In this assignment, you will build a complete Reinforcement Learning from Human Feedback (RLHF) training pipeline for language models. The process consists of two major phases:

(1) Train a Reward Model from human preference data.

(2) Fine-Tune a Policy Model using Proximal Policy Optimization (PPO) with the trained reward model.

You will implement both full model fine-tuning and parameter-efficient fine-tuning using Low-Rank Adaptation (LoRA).

🛠️ Datasets: You will use two datasets, each designated for a specific phase:

Reward Model Training Dataset:

➔ Stanford Human Preferences Dataset (SHP). You can find the dataset at Huggingface as well.

```
from datasets import load_dataset

dataset = load_dataset("stanfordnlp/SHP")
```

Before training the reward model, you must display a table showing at least 5 rows of the final preprocessed inputs that go directly into the model.

Use columns: history, human_ref_A, human_ref_B, preferred.

Example table:

input_chosen	input_rejected
"Prompt: … Preferred response: …"	"Prompt: … Rejected response: …"

➔ Use a subset of minimum 5k examples for manageable training.

PPO Fine-Tuning Dataset:

➔ OpenAssistant Conversations Dataset (OpenAssistant/oasst1) from Hugging Face.
➔ Extract only the prompter texts (user questions).
➔ Drop all assister (assistant response) texts.
➔ Randomly sample 2000 prompts for PPO training.

🧠 Part 1: Reward Model Training

You will train a reward model to predict human preferences between two responses to a given prompt.

- Use a recently released small causal language model and attach a scalar reward head
- Training Details:
    - Loss: Pairwise Margin Ranking Loss
    - Optimizer: AdamW
    - Learning Rate: 5e-5
    - Batch Size: 8
    - Epochs: 5
    - Tokenize input carefully (prompt + response concatenation).
- Save the best reward model based on validation loss for use in PPO fine-tuning.
- This model will be frozen during PPO training.

🧠 Part 2: PPO Fine-Tuning of Policy Model

You will fine-tune a new copy of the small causal model as the policy model using PPO, guided by the reward model you just trained.

For each prompt:

- Generate a response using the **policy model**
- Score the response using the **frozen reward model**
- Update the policy model using **PPO** to maximize reward while maintaining KL stability

Training Details**:**

- Optimizer: PPOTrainer (trl library recommended)
- Learning Rate: 1e-5
- Batch Size: 64
- Forward Batch Size: 16
- Epochs: **5**
- Max Prompt Length: 256 tokens
- Max Response Length: 128 tokens
- Target moderate KL divergence (~0.02–0.1).

Two PPO Variants

You must train **two policy models**:

1. **Full Fine-Tuning**: Update all parameters of the policy model.
2. **LoRA Fine-Tuning**: Insert LoRA adapters into attention layers (q_proj and v_proj) with:
    - Rank (r): 4-16
    - Alpha: 32
    - Dropout: 0.2
    - Only LoRA parameters are updated.

Save both fine-tuned model checkpoints separately.

📊 Part 3 — Evaluation & Reporting

You must evaluate your models along three dimensions:

1. Reward Score Evaluation (Quantitative)

Using at least **50 unseen prompts**:
Generate responses from:

(a) Pretrained model
(b) Full PPO fine-tuned model
(c) LoRA-PPO fine-tuned model

Score all responses using the frozen **reward model**.
Report:

a. Average reward for each model
b. Average Reward gain:

Reward (PPO) – Reward (Pretrained)
Reward (LoRA-PPO) – Reward (Pretrained)

2. KL Divergence Monitoring (Quantitative)

During PPO training, record: KL divergence vs. PPO steps and provide a proper interpretation

- KL too high → model diverged from base LM
- KL too low → model did not learn enough

Your report must include the KL plot and a short interpretation.

3. Manual Human Evaluation (Qualitative):

Write a comparative analysis on the manual rating.
Select any 3 unseen prompts and for each prompt, generate responses from:
(a) Original pre-trained model
(b) Full fine-tuned PPO model
(c) LoRA fine-tuned PPO model.

You must Rate each generated response on a scale of 1–5 based on: Coherence, Relevance, Helpfulness, Completeness, Hallucination/toxicity (if any)

- Create three tables like below for each one of the model options:

Table 1: Manual Ratings (1-5) of Generated Responses Using Original Pre-trained model

	Generated Response	Ratings (1 -5)
Prompt 1		Coherence: Relevance: Helpfulness: Completeness: Hallucination/toxicity:
Prompt 2		Coherence: Relevance: Helpfulness: Completeness: Hallucination/toxicity:
Prompt 3		Coherence: Relevance: Helpfulness: Completeness: Hallucination/toxicity:

BONUS (3 points): ***Evaluating Deep Convolutional Generative Adversarial Networks (DCGANs) for Medical Image Synthesis***

Implement and critically evaluate a DCGAN for generating medical images using one of the following datasets (based on your student ID):

DermaMNIST – if the final digit of your student ID is *even*

ChestMNIST – if the final digit of your student ID is *odd*

Example setup:

```
!pip install medmnist

from medmnist import PathMNIST
train_dataset = PathMNIST(split="train")
```

Implementation Requirements

- Train for at least 1000 epochs using the Adam optimizer ($\beta_1 = 0.5$, $\beta_2 = 0.999$).

- Generator: follow the standard DCGAN architecture; you may modify only the final convolutional layer to match the image dimensions of the selected dataset.
- Discriminator: must contain a minimum of four convolutional layers, each followed by suitable normalization and activation functions (e.g., Leaky ReLU + BatchNorm).

Evaluation and Analysis: Display minimum 32 synthetic images (8×4 grid) and visually evaluate sample diversity and discuss presence of mode collapse.

You are required to follow:

1. Submit one MS/PDF documents in terms of report. Failing to submit a report containing the following will be penalized from 2-5 points:

Include all the steps of your calculations (if there is any).

Include all discussion in this report.

Include the summary of the model.

Attach screenshots – showing first few epochs of model training.

Attach screenshots of the important code outputs such as confusion matrices, learning curves, and classification reports.

2. Source code:

a. Python (Jupyter Notebook)

b. Ensure it is well-organized with comments and proper indentation.

Failure to submit the source code will result in a deduction of full/partial points.

Format your filenames as follows: "your_last_name_HW2.pdf" for the document and "your_last_name_HW2_source_code.ipynb" for the source code.

Before submitting the source code, please double-check that it runs without any errors.

Must submit the files separately.

Do not compress into a zip file.

HW submitted more than 24 hours late will not be accepted for credit.