1. a) $$J_{MLE}(\theta) = E_{p(x)} E_{\pi_{teacher}} [\log \pi_\theta(y|x)]$$
$$= \int_x p(x) \int_y \pi_t(y|x) \log \pi_\theta(y|x)\, dy dx$$

$$\boxed{J'_{MLE}(\theta) = \int_x p(x) \int_y \pi_t(y|x) \frac{\partial}{\partial \theta} [\log \pi_\theta(y|x)] dy dx}$$

Stochastic gradient algorithm

1. Sample N datapoints (x_i, y_i)
$\{x_i\}^N \sim p_{data}(x)$ and $\{y_i\}^N \sim \pi_t(y_i|x_i)$
2. Approximate $J'_{MLE}(\theta)$ from finite sample
$J'_{MLE}(\theta) \approx \frac{1}{N} \sum_{i=1}^N \frac{\partial}{\partial \theta} \log \pi_\theta(y_i|x_i)$
3. Update parameters
$\theta^{(t+1)} = \theta^{(t)} + \eta J'(\theta)$

b) $$J_{RL} = E_{p(x)} E_{\pi_\theta(y|x)} [r(x,y)]$$
$$J'_{RL} = \int_x p(x) \int_y \frac{\partial}{\partial \theta} [\pi_\theta(y|x)]\, r(x,y)\, dy dx$$
$$= \int_x p(x) \int_y \pi_\theta(y|x) \frac{\partial}{\partial \theta} [\log \pi_\theta(y|x)]\, r(x,y)\, dy dx$$
$$= \boxed{E_{p(x)} E_{\pi_\theta(y|x)} [r(x,y) \frac{\partial}{\partial \theta} \log \pi_\theta(y|x)]}$$

$\frac{\partial}{\partial \theta} \log \pi_\theta = \frac{1}{\pi_\theta} \frac{\partial}{\partial \theta} \pi_\theta$

$\pi_\theta \frac{\partial}{\partial \theta} \log \pi_\theta = \frac{\partial}{\partial \theta} \pi_\theta$

Stochastic gradient algorithm

1. Sample N datapoints (x_i, y_i)
$\{x_i\}^N \sim p_{data}(x)$ and $\{y_i\}^N \sim \pi_\theta(y_i|x_i)$ ← current probability distribution
2. Approximate $J'_{MLE}(\theta)$ from finite sample
$J'_{MLE}(\theta) \approx \frac{1}{N} \sum_{i=1}^N [r(x_i, y_i) \cdot \frac{\partial}{\partial \theta} \log \pi_\theta(y_i|x_i)]$
3. Update parameters
$\theta^{(t+1)} = \theta^{(t)} + \eta J'(\theta)$

What if we substitute $r(x,y) \rightarrow r(x,y) - b(x)$

$$J'_{RL,new} = \int_x p(x) \int_y \frac{\partial}{\partial\theta}[\pi_\theta(y|x)][r(x,y) - b(x)]\, dy\, dx$$

$$= J'_{RL,old} - \int_x p(x) \int_y \frac{\partial}{\partial\theta}[\pi_\theta(y|x)]\, b(x)\, dy\, dx$$

$$= J'_{RL,old} - \int_x p(x)\, b(x) \int_y \frac{\partial}{\partial\theta}[\pi_\theta(y|x)]\, dy\, dx$$

$$= J'_{RL,old} - \int_x p(x)\, b(x)\, \frac{\partial}{\partial\theta} \underbrace{\int_y [\pi_\theta(y|x)]\, dy}_{=1}$$

$$= J'_{RL,old} - \int_x p(x)\, b(x)\, \frac{\partial}{\partial\theta}[1]\, dx \quad (\frac{\partial}{\partial\theta}[1] \rightarrow 0)$$

$$= J'_{RL,old}$$

$\therefore J'_{RL}$ does not change

If we set $r(x,y) = 1$ for J_{RL} and J'_{RL}

$$E_{\pi_\theta}\left[\frac{\partial}{\partial\theta} \log \pi_\theta\right] = \int_y \pi_\theta(y|x) \cdot \frac{1}{\pi_\theta} \frac{\partial}{\partial\theta} \pi_\theta\, dy$$

$$= \int_y \frac{\partial}{\partial\theta} \pi_\theta\, dy$$

$$= \frac{\partial}{\partial\theta} \int_y \pi_\theta\, dy$$

$$= \frac{\partial}{\partial\theta}[1]$$

$$= \boxed{0}$$

c) Stochastic Gradient Algorithms for MLE and RL

- Similarities
  - draw (x_i, y_i) from a joint distribution to estimate $J'(\theta)$
- Differences
  - RL includes a reward factor when calculating $J'(\theta)$
  - RL model learns from itself, so the distribution $\pi_\theta(y|x)$ from which we sample y_i changes each update

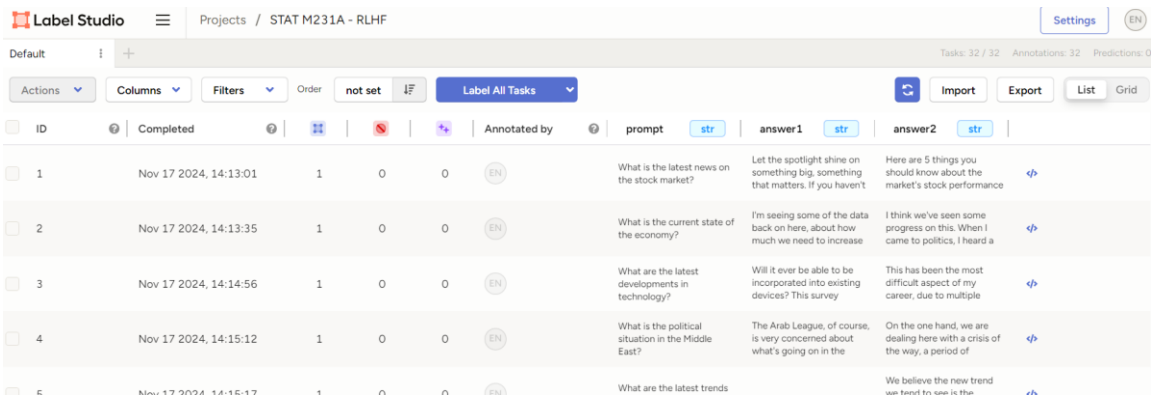
2a) Explanation of Code

We first create a custom dataset using some predefined prompts and answers generated using GPT-2. We (the user) rank the predictions generated, and this labeled data is used as the training data for training the reward model. For the preprocessing step: the token sequence inputs use padding to make them same length, and an attention map to distinguish between actual tokens and padding. GPT-2 is used as the base for the reward model, and the first 70% of hidden layers are frozen (only latter 30% are updated).

We then fine-tune a GPT-2 model using PPO and our pre-trained reward model to generate summaries from online posts (using Reddit TL;DR dataset for training). Only the last 8 layers will be refined. We define a reward function to be the difference in scores (calculated using our reward model) between the original sample data and the predicted sample data using the GPT model. This serves as the loss function for which we backpropagate and fine-tune the model.

2b) Tested Code Notebook

I was able to use Label Studio to annotate the custom dataset.



I then was able to train the reward model based on the labeled data.

```
training_args = TrainingArguments(
    output_dir="rm_checkpoint/",
    num_train_epochs=100,
    logging_steps=10,
    gradient_accumulation_steps=4,
    save_strategy="steps",
    evaluation_strategy="steps",
    per_device_train_batch_size=4,
    per_device_eval_batch_size=4,
    eval_accumulation_steps=1,
    eval_steps=10,
    save_steps=10,
    warmup_steps=100,
    logging_dir="./logs",
    fp16=True,
    bf16=False,
    learning_rate=1e-4,
    # deepspeed="ds_config_gpt_j.json"
    save_total_limit=1
)

Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    compute_metrics=compute_metrics,
    eval_dataset=val_dataset,
    data_collator=data_collator,
).train()
```

Step	Training Loss	Validation Loss	Accuracy
10	0.575200	0.778556	1.000000
20	0.545700	0.774925	0.857143
30	0.484300	0.773462	0.857143
40	0.396500	0.785950	0.857143
50	0.269000	0.835993	0.857143
60	0.149300	0.940576	0.857143
70	0.057200	1.131552	0.857143
80	0.012300	1.351765	0.571429
90	0.001700	1.518373	0.571429
100	0.000700	1.607360	0.571429

```
TrainOutput(global_step=100, training_loss=0.249198
'train_steps_per_second': 0.283, 'total_flos': 0.0,
```

Unfortunately, I was not able to train the language model because I could not get trlx to install properly. (I was using Google Colab since I do not have a GPU). Attempting to train on my local machine's CPU was extremely slow. Note: the authors of the jupyter notebook commented that trlx is not compatible with Google Colab and cannot be trained if using Google Colab.

```
# !pip install -U git+https://github.com/CarperAI/trlx.git

import os
from typing import List

import torch
from datasets import load_dataset
from reward_model import GPTRewardModel
from tqdm import tqdm
from transformers import AutoTokenizer

import trlx
from trlx.data.configs import (
    ModelConfig,
    OptimizerConfig,
    SchedulerConfig,
    TokenizerConfig,
    TrainConfig,
    TRLConfig,
)
from trlx.models.modeling_ppo import PPOConfig

--INSERT--
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-22-6754258bbefe> in <cell line: 13>()
     11
     12 import trlx
---> 13 from trlx.data.configs import (
     14     ModelConfig,
     15     OptimizerConfig,

ModuleNotFoundError: No module named 'trlx.data'
```