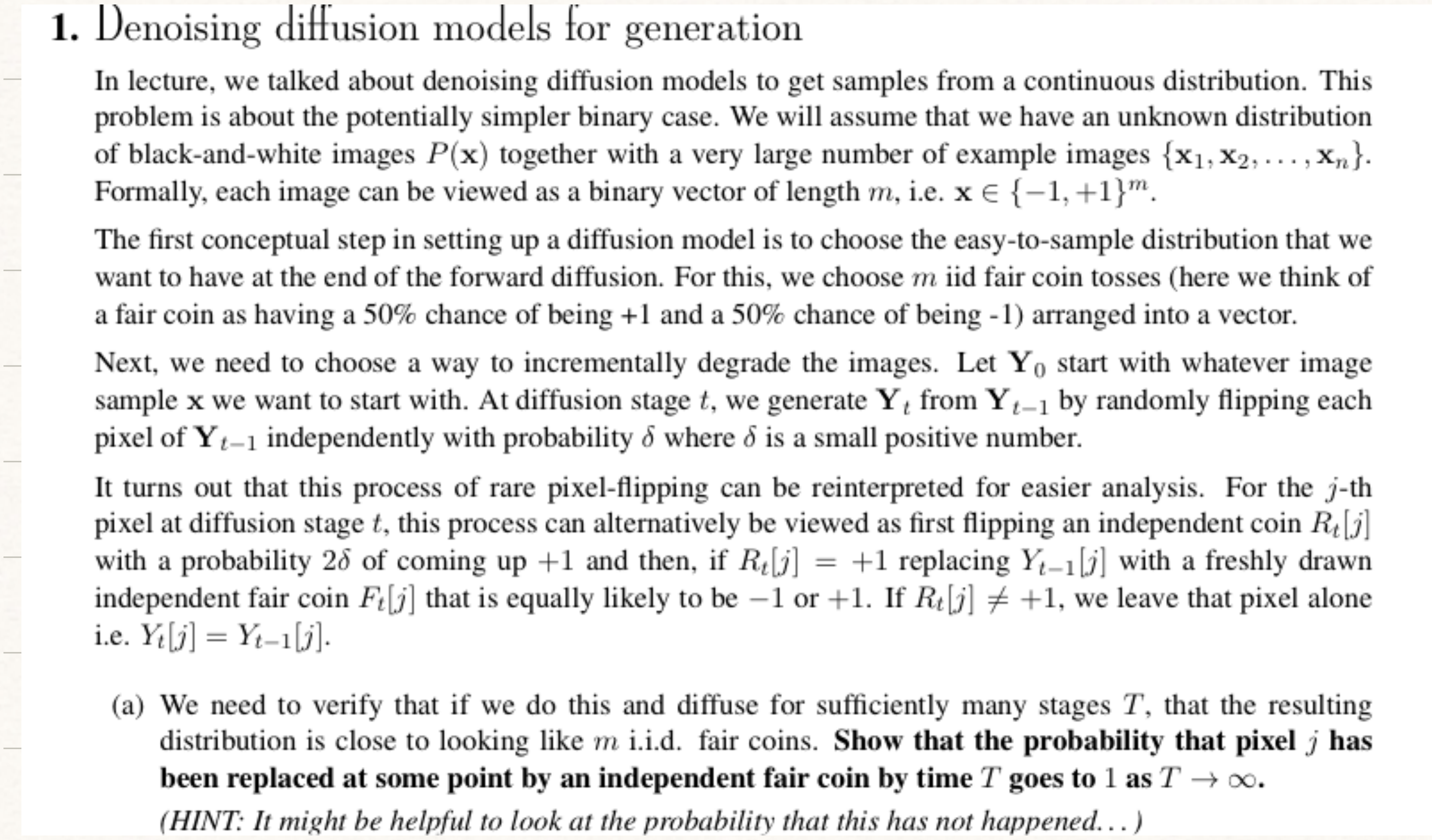
HW 12

Mengying Lin SID: 3038737132.



1. Denoising diffusion models for generation

In lecture, we talked about denoising diffusion models to get samples from a continuous distribution. This problem is about the potentially simpler binary case. We will assume that we have an unknown distribution of black-and-white images $P(\mathbf{x})$ together with a very large number of example images $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$. Formally, each image can be viewed as a binary vector of length m, i.e. $\mathbf{x} \in \{-1, +1\}^m$.

The first conceptual step in setting up a diffusion model is to choose the easy-to-sample distribution that we want to have at the end of the forward diffusion. For this, we choose m iid fair coin tosses (here we think of a fair coin as having a 50% chance of being +1 and a 50% chance of being -1) arranged into a vector.

Next, we need to choose a way to incrementally degrade the images. Let $\mathbf{Y}_0$ start with whatever image sample $\mathbf{x}$ we want to start with. At diffusion stage t, we generate $\mathbf{Y}_t$ from $\mathbf{Y}_{t-1}$ by randomly flipping each pixel of $\mathbf{Y}_{t-1}$ independently with probability δ where δ is a small positive number.

It turns out that this process of rare pixel-flipping can be reinterpreted for easier analysis. For the j-th pixel at diffusion stage t, this process can alternatively be viewed as first flipping an independent coin $R_t[j]$ with a probability 2δ of coming up $+1$ and then, if $R_t[j] = +1$ replacing $Y_{t-1}[j]$ with a freshly drawn independent fair coin $F_t[j]$ that is equally likely to be -1 or $+1$. If $R_t[j] \neq +1$, we leave that pixel alone i.e. $Y_t[j] = Y_{t-1}[j]$.

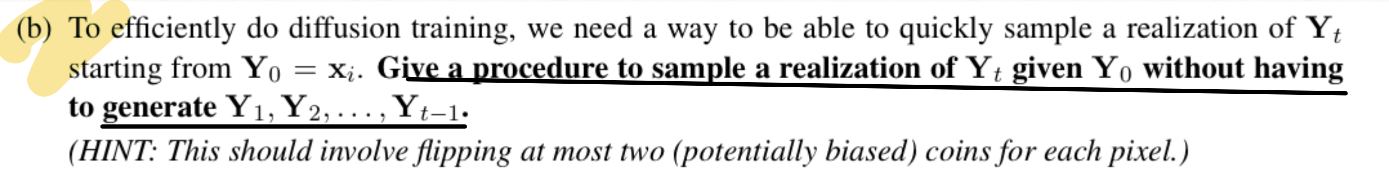
(a) We need to verify that if we do this and diffuse for sufficiently many stages T, that the resulting distribution is close to looking like m i.i.d. fair coins. **Show that the probability that pixel j has been replaced at some point by an independent fair coin by time T goes to 1 as $T \to \infty$.**
(HINT: It might be helpful to look at the probability that this has not happened...)

Denote the given event as A.

$$P(\bar{A}_T) = (1-2\delta + 2\delta \times \tfrac{1}{2})^T = (1-\delta)^T.$$

$$\lim_{T\to\infty} P(\bar{A}_T) = \lim_{T\to\infty} (1-\delta)^T = 0$$

Hence when $T \to \infty$. $P(A_T) \to 1$.



(b) To efficiently do diffusion training, we need a way to be able to quickly sample a realization of $\mathbf{Y}_t$ starting from $\mathbf{Y}_0 = \mathbf{x}_i$. **Give a procedure to sample a realization of $\mathbf{Y}_t$ given $\mathbf{Y}_0$ without having to generate $\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_{t-1}$.**
(HINT: This should involve flipping at most two (potentially biased) coins for each pixel.)

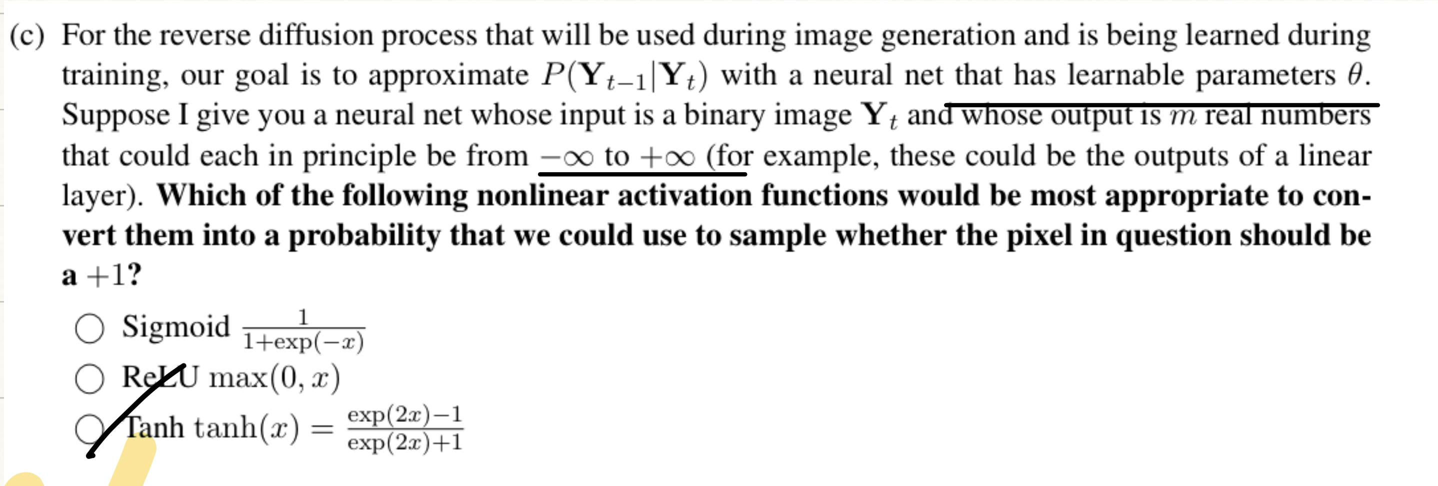
Method 1: If one pixel is flipped for a even number of times it'll be the same as Y_0, if it is flipped for an odd number of times, it'll differ from Y_0.

$$P(Y_t = Y_0) = \sum_{i=0}^{\lfloor \frac{t}{2} \rfloor} \delta^{2i}(1-\delta)^{t-2i} = \sum_{i=0}^{\lfloor \frac{t}{2} \rfloor} (\delta^2)^i [(1-\delta)^2]^{\frac{t}{2}-i} = [\delta^2 + (1-\delta)^2]^{\lfloor \frac{t}{2} \rfloor}$$

We then can flip Y_t with $1-[\delta^2+(1-\delta)^2]^{\lfloor\frac{t}{2}\rfloor}$ probability.

Method 2: Model Y_t with $Y_t=\sqrt{1-\varepsilon}\,Y_{t-1}+\sqrt{\varepsilon}N(0,1)$.

Hence we have
$$Y_t=(\sqrt{1-\varepsilon})^t Y_0+\sqrt{\varepsilon}\Big(\sum_{k=0}^{t-1}(\sqrt{1-\varepsilon})^k\Big]N(0,1).$$
$$=(\sqrt{1\varepsilon})^t Y_0+\frac{(\sqrt{1-\varepsilon})^t-1}{\sqrt{1-\varepsilon}-1}N(0,1).$$

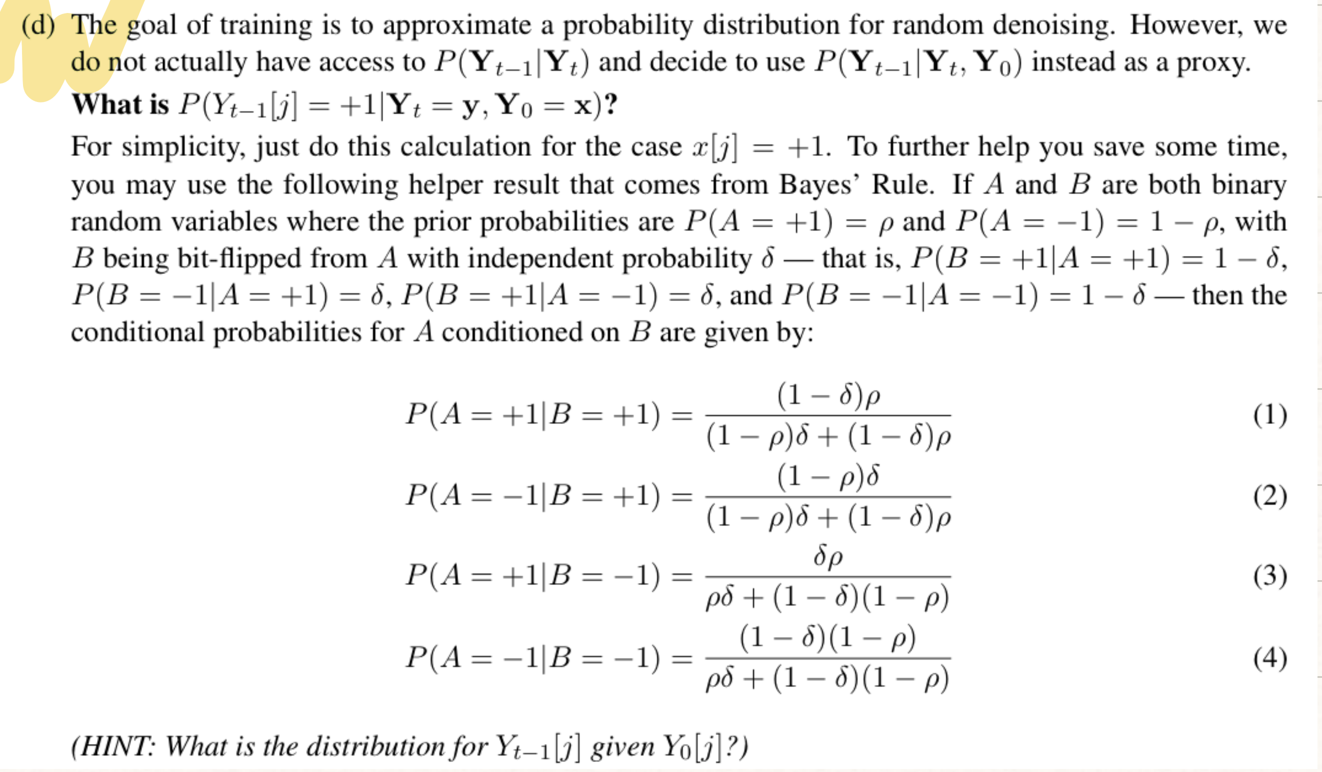


(c) For the reverse diffusion process that will be used during image generation and is being learned during training, our goal is to approximate $P(\mathbf{Y}_{t-1}|\mathbf{Y}_t)$ with a neural net that has learnable parameters θ. Suppose I give you a neural net whose input is a binary image $\mathbf{Y}_t$ and whose output is m real numbers that could each in principle be from $-\infty$ to $+\infty$ (for example, these could be the outputs of a linear layer). **Which of the following nonlinear activation functions would be most appropriate to convert them into a probability that we could use to sample whether the pixel in question should be a $+1$?**

○ Sigmoid $\frac{1}{1+\exp(-x)}$

○ ReLU $\max(0,x)$

[✓ ○ Tanh $\tanh(x)=\frac{\exp(2x)-1}{\exp(2x)+1}$]



(d) The goal of training is to approximate a probability distribution for random denoising. However, we do not actually have access to $P(\mathbf{Y}_{t-1}|\mathbf{Y}_t)$ and decide to use $P(\mathbf{Y}_{t-1}|\mathbf{Y}_t,\mathbf{Y}_0)$ instead as a proxy. **What is $P(Y_{t-1}[j]=+1|\mathbf{Y}_t=\mathbf{y},\mathbf{Y}_0=\mathbf{x})$?**

For simplicity, just do this calculation for the case $x[j]=+1$. To further help you save some time, you may use the following helper result that comes from Bayes' Rule. If A and B are both binary random variables where the prior probabilities are $P(A=+1)=\rho$ and $P(A=-1)=1-\rho$, with B being bit-flipped from A with independent probability δ — that is, $P(B=+1|A=+1)=1-\delta$, $P(B=-1|A=+1)=\delta$, $P(B=+1|A=-1)=\delta$, and $P(B=-1|A=-1)=1-\delta$ — then the conditional probabilities for A conditioned on B are given by:

$$P(A=+1|B=+1)=\frac{(1-\delta)\rho}{(1-\rho)\delta+(1-\delta)\rho} \quad (1)$$

$$P(A=-1|B=+1)=\frac{(1-\rho)\delta}{(1-\rho)\delta+(1-\delta)\rho} \quad (2)$$

$$P(A=+1|B=-1)=\frac{\delta\rho}{\rho\delta+(1-\delta)(1-\rho)} \quad (3)$$

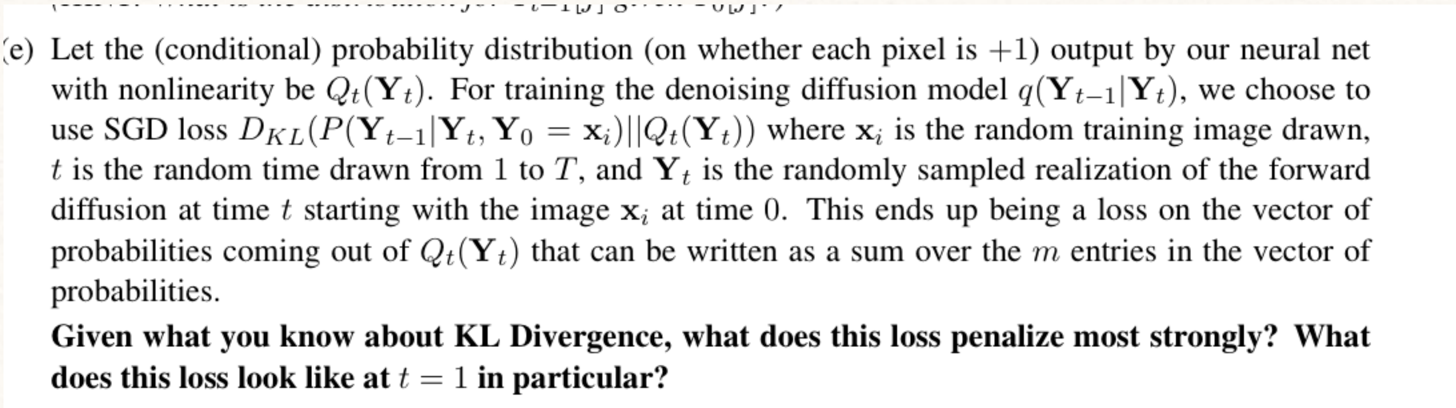
$$P(A=-1|B=-1)=\frac{(1-\delta)(1-\rho)}{\rho\delta+(1-\delta)(1-\rho)} \quad (4)$$

(HINT: What is the distribution for $Y_{t-1}[j]$ given $Y_0[j]$?)

let "$Y_{t-1}[j]=+1$" be event A, and "$Y_t=y$" be event B.

"$Y_0[j]=x$" be event C.

$$P(A|B,C)=\frac{P(A,B|C)}{P(B|C)}=\frac{P(B|A,C)P(A|C)}{P(B|C)}$$
$$=\frac{P(B|A)P(A|C)}{P(B|C)}$$

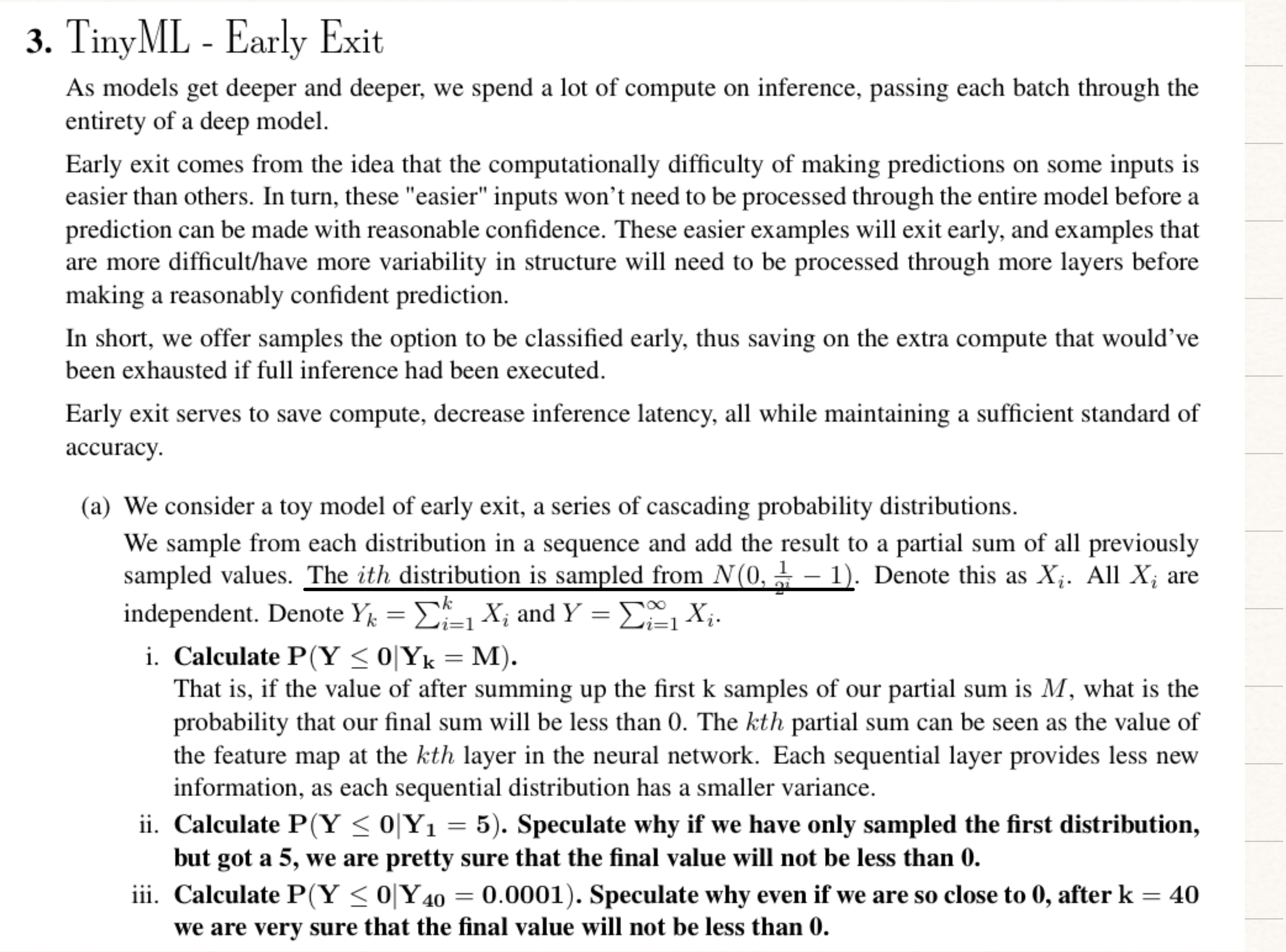


(e) Let the (conditional) probability distribution (on whether each pixel is $+1$) output by our neural net with nonlinearity be $Q_t(\mathbf{Y}_t)$. For training the denoising diffusion model $q(\mathbf{Y}_{t-1}|\mathbf{Y}_t)$, we choose to use SGD loss $D_{KL}(P(\mathbf{Y}_{t-1}|\mathbf{Y}_t, \mathbf{Y}_0 = \mathbf{x}_i)||Q_t(\mathbf{Y}_t))$ where $\mathbf{x}_i$ is the random training image drawn, t is the random time drawn from 1 to T, and $\mathbf{Y}_t$ is the randomly sampled realization of the forward diffusion at time t starting with the image $\mathbf{x}_i$ at time 0. This ends up being a loss on the vector of probabilities coming out of $Q_t(\mathbf{Y}_t)$ that can be written as a sum over the m entries in the vector of probabilities.

Given what you know about KL Divergence, what does this loss penalize most strongly? What does this loss look like at $t = 1$ in particular?

It drives $Q_t(Y_t)$. to be closer to $P(Y_{t-1}|Y_t, Y_0 = x_i)$, which means forcing the prediction to be closer to the true distribution.

when $t=1$, the term becomes $D_{KL}(P(Y_0|Y_1, Y_0 = x_i)||Q_t(Y_t))$, which measures how the model can denoise the image after one diffusion step.



3. TinyML - Early Exit

As models get deeper and deeper, we spend a lot of compute on inference, passing each batch through the entirety of a deep model.

Early exit comes from the idea that the computationally difficulty of making predictions on some inputs is easier than others. In turn, these "easier" inputs won't need to be processed through the entire model before a prediction can be made with reasonable confidence. These easier examples will exit early, and examples that are more difficult/have more variability in structure will need to be processed through more layers before making a reasonably confident prediction.

In short, we offer samples the option to be classified early, thus saving on the extra compute that would've been exhausted if full inference had been executed.

Early exit serves to save compute, decrease inference latency, all while maintaining a sufficient standard of accuracy.

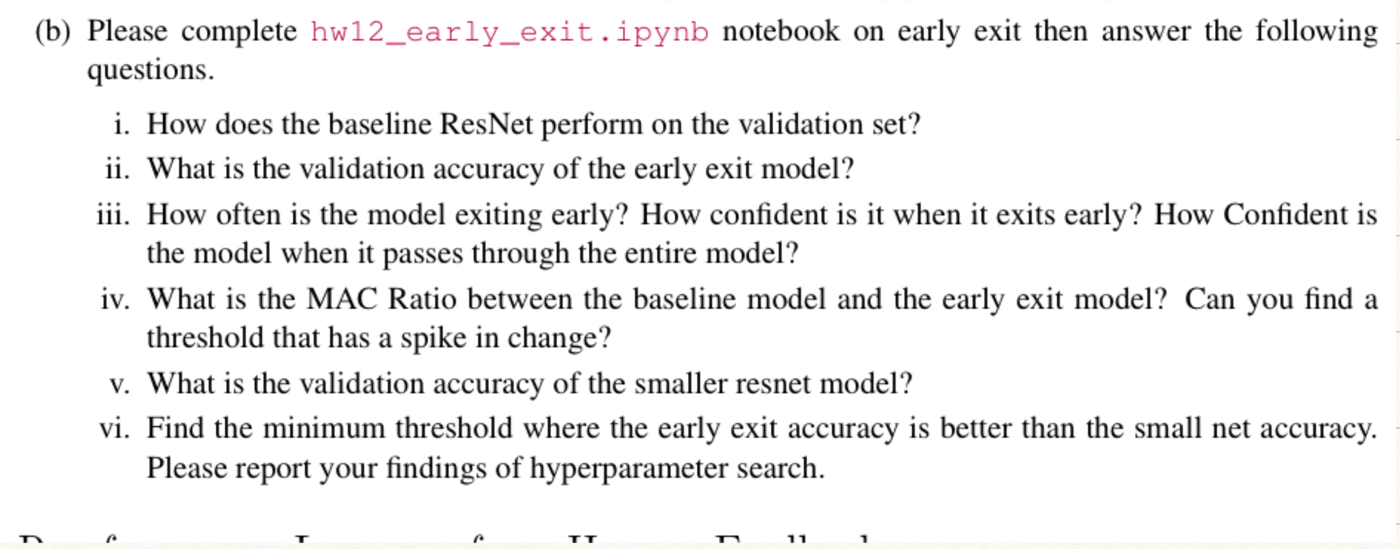
(a) We consider a toy model of early exit, a series of cascading probability distributions.
We sample from each distribution in a sequence and add the result to a partial sum of all previously sampled values. The ith distribution is sampled from $N(0, \frac{1}{2^i} - 1)$. Denote this as X_i. All X_i are independent. Denote $Y_k = \sum_{i=1}^{k} X_i$ and $Y = \sum_{i=1}^{\infty} X_i$.

i. **Calculate $\mathbf{P(Y \leq 0|Y_k = M)}$.**
That is, if the value of after summing up the first k samples of our partial sum is M, what is the probability that our final sum will be less than 0. The kth partial sum can be seen as the value of the feature map at the kth layer in the neural network. Each sequential layer provides less new information, as each sequential distribution has a smaller variance.

ii. **Calculate $\mathbf{P(Y \leq 0|Y_1 = 5)}$. Speculate why if we have only sampled the first distribution, but got a 5, we are pretty sure that the final value will not be less than 0.**

iii. **Calculate $\mathbf{P(Y \leq 0|Y_{40} = 0.0001)}$. Speculate why even if we are so close to 0, after $\mathbf{k = 40}$ we are very sure that the final value will not be less than 0.**

a) i. $P(Y \leq 0 | Y_k = M) = P(\sum_{i=k}^{+\infty} X_i + M \leq 0) = P(\sum_{i=k}^{+\infty} X_i \leq -M)$



(b) Please complete `hw12_early_exit.ipynb` notebook on early exit then answer the following questions.

i. How does the baseline ResNet perform on the validation set?

ii. What is the validation accuracy of the early exit model?

iii. How often is the model exiting early? How confident is it when it exits early? How Confident is the model when it passes through the entire model?

iv. What is the MAC Ratio between the baseline model and the early exit model? Can you find a threshold that has a spike in change?

v. What is the validation accuracy of the smaller resnet model?

vi. Find the minimum threshold where the early exit accuracy is better than the small net accuracy. Please report your findings of hyperparameter search.

b) i. Acc: 57.24%

ii. Acc: 59.4%

iii. 97.4684% of batches exist early.

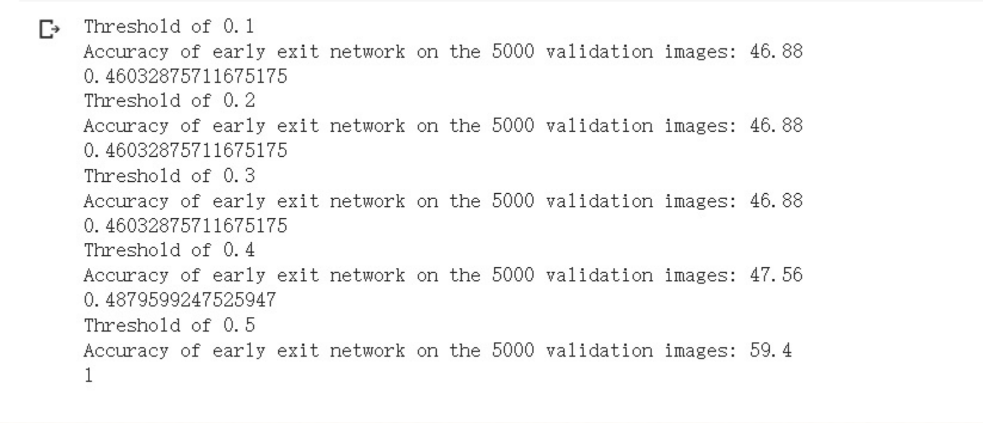
when it exists early, the confidence is 0.4240.

when it passes through the whole net the confidence is 0.4212.

iv. 0.4761. 0.5 is a threshold that has a spike in change

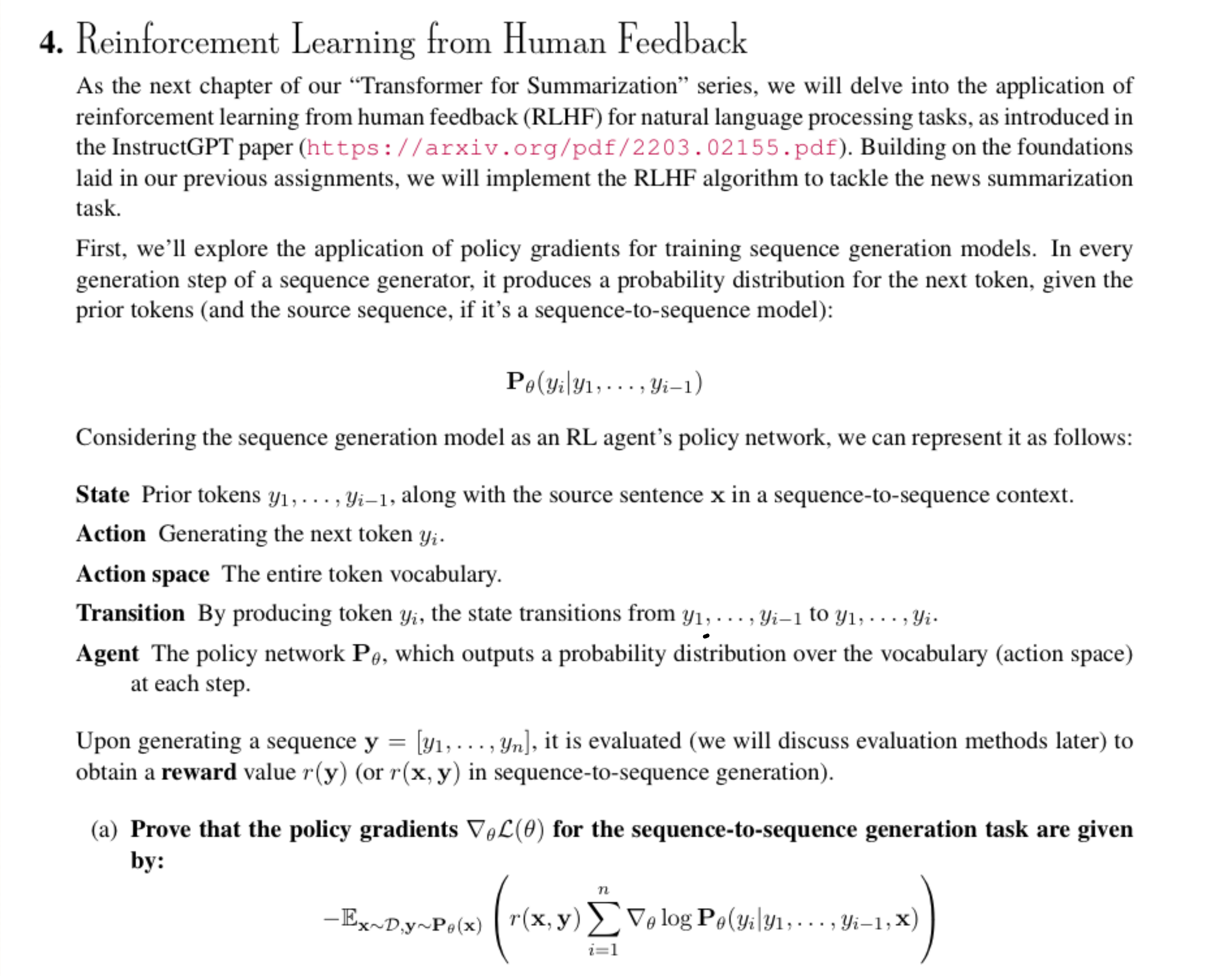
v. Acc: 45.1800%

vi



```
Threshold of 0.1
Accuracy of early exit network on the 5000 validation images: 46.88
0.46032875711675175
Threshold of 0.2
Accuracy of early exit network on the 5000 validation images: 46.88
0.46032875711675175
Threshold of 0.3
Accuracy of early exit network on the 5000 validation images: 46.88
0.46032875711675175
Threshold of 0.4
Accuracy of early exit network on the 5000 validation images: 47.56
0.4879599247525947
Threshold of 0.5
Accuracy of early exit network on the 5000 validation images: 59.4
1
```

The model seems to predict each sample point with similar confidence. when tuning the threshold from 0.1 to 0.4. we did not see a significant change in both acc & mac ratio. when it reached 0.5, the mac ratio rose to 1 and acc increased greatly



4. Reinforcement Learning from Human Feedback

As the next chapter of our "Transformer for Summarization" series, we will delve into the application of reinforcement learning from human feedback (RLHF) for natural language processing tasks, as introduced in the InstructGPT paper (https://arxiv.org/pdf/2203.02155.pdf). Building on the foundations laid in our previous assignments, we will implement the RLHF algorithm to tackle the news summarization task.

First, we'll explore the application of policy gradients for training sequence generation models. In every generation step of a sequence generator, it produces a probability distribution for the next token, given the prior tokens (and the source sequence, if it's a sequence-to-sequence model):

$$\mathbf{P}_\theta(y_i|y_1,\ldots,y_{i-1})$$

Considering the sequence generation model as an RL agent's policy network, we can represent it as follows:

State Prior tokens $y_1,\ldots,y_{i-1}$, along with the source sentence $\mathbf{x}$ in a sequence-to-sequence context.

Action Generating the next token y_i.

Action space The entire token vocabulary.

Transition By producing token y_i, the state transitions from $y_1,\ldots,y_{i-1}$ to $y_1,\ldots,y_i$.

Agent The policy network $\mathbf{P}_\theta$, which outputs a probability distribution over the vocabulary (action space) at each step.

Upon generating a sequence $\mathbf{y} = [y_1,\ldots,y_n]$, it is evaluated (we will discuss evaluation methods later) to obtain a **reward** value $r(\mathbf{y})$ (or $r(\mathbf{x},\mathbf{y})$ in sequence-to-sequence generation).

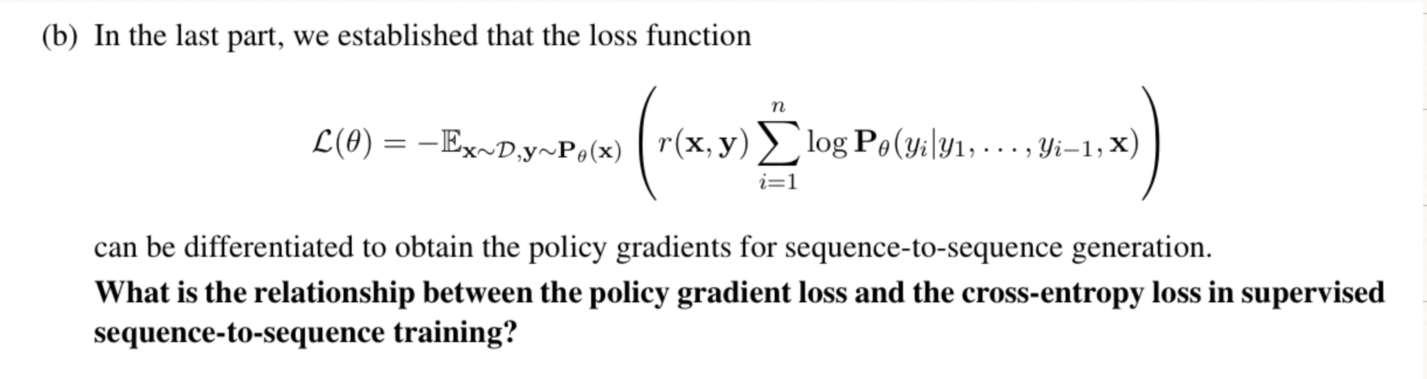
(a) **Prove that the policy gradients $\nabla_\theta\mathcal{L}(\theta)$ for the sequence-to-sequence generation task are given by:**

$$-\mathbb{E}_{\mathbf{x}\sim\mathcal{D},\mathbf{y}\sim\mathbf{P}_\theta(\mathbf{x})}\left(r(\mathbf{x},\mathbf{y})\sum_{i=1}^{n}\nabla_\theta\log\mathbf{P}_\theta(y_i|y_1,\ldots,y_{i-1},\mathbf{x})\right)$$

a) $\mathcal{L}(\theta) = -\sum_{i=1}^{n} P_\theta(y_i|y_1,\cdots y_{i-1},x)\, r(x,y_i)$

$\nabla_\theta \mathcal{L}(\theta) = -\sum_{i=1}^{n}[p_\theta(y_i|y_1\cdots y_{i-1})\, r(x,y_i)\, \nabla_\theta \log P_\theta(y_i|y_1\cdots y_{i-1},x)]$

$= -E_{x\sim D,\, y\sim P_\theta(x)}[r(x,y)\sum_{i=1}^{n}\nabla_\theta \log P_\theta(y_i|y_1\cdots y_{i-1},x)]$.

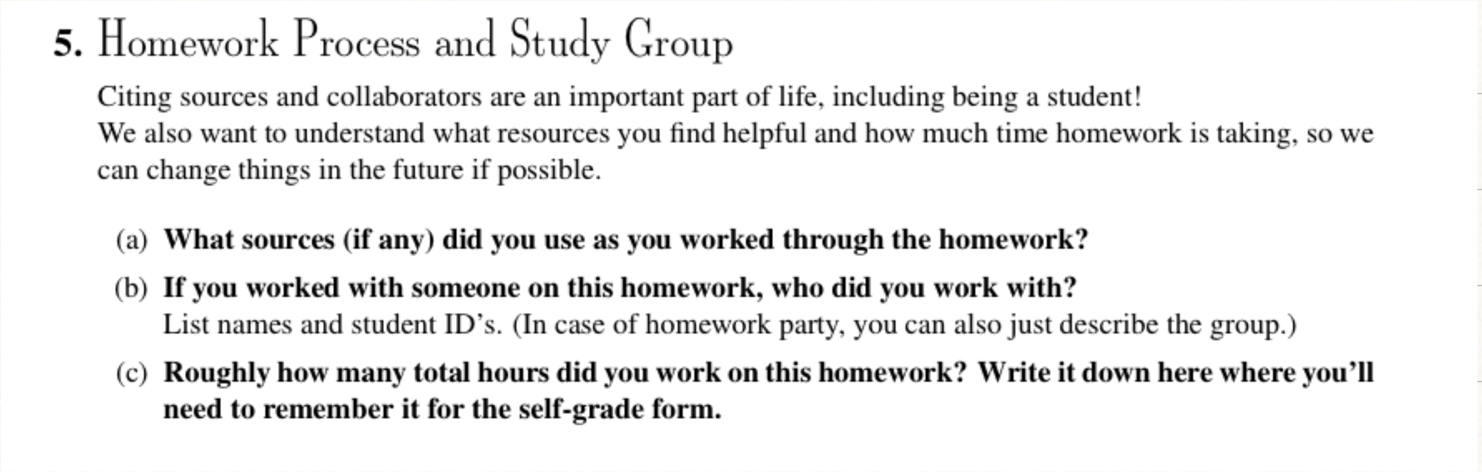


(b) In the last part, we established that the loss function

$$\mathcal{L}(\theta) = -\mathbb{E}_{\mathbf{x}\sim\mathcal{D},\mathbf{y}\sim\mathbf{P}_\theta(\mathbf{x})}\left(r(\mathbf{x},\mathbf{y})\sum_{i=1}^{n}\log\mathbf{P}_\theta(y_i|y_1,\ldots,y_{i-1},\mathbf{x})\right)$$

can be differentiated to obtain the policy gradients for sequence-to-sequence generation.
What is the relationship between the policy gradient loss and the cross-entropy loss in supervised sequence-to-sequence training?

It reward function $r(x,y)$ is designed to be the true distribution of y, policy gradient loss is the same as cross-entropy loss.



5. Homework Process and Study Group

Citing sources and collaborators are an important part of life, including being a student!
We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

(a) **What sources (if any) did you use as you worked through the homework?**

(b) **If you worked with someone on this homework, who did you work with?**
List names and student ID's. (In case of homework party, you can also just describe the group.)

(c) **Roughly how many total hours did you work on this homework? Write it down here where you'll need to remember it for the self-grade form.**

a) Torch official document

b) NA

c) 15h

Happy ending! A nice rose for all 😁!