Prof. Ryan Cotterell

Assignment 1

05/04/2025 - 22:42h

Disclaimer: In full transparency, some of the questions on this assignment are modified versions of recent NLP publications that treat language models formally. The relevant publications are not difficult to find. Our choice to base the assignment questions on recent results should assure you that the questions introduce you to recent research in the field and that the questions are interesting—indeed, they are *so* interesting active researchers succeeded in publishing their answers in well-respected venues! In principle, you *can* simply copy the solutions to the questions below from the original text in some cases. Moreover, unless such copying is done sloppily,[1] i.e., if you do not take care to cover your tracks, you will likely get away with it. However, we strongly advise against such a tactic. First, while it may sound hackneyed, you are simply cheating yourself. A classroom is a controlled environment where you can learn under the guidance of other humans who understand the subject matter. You ought to take advantage of this opportunity. Indeed, that is presumably why you came to ETHZ in the first place. More practically, the nature of the questions on the assignments is meant to, to a large extent, resemble the possible questions in the final exam and are meant to prepare you for it. Any content covered here is fair game for the exam. Therefore, we strongly recommend that you *think* about the problems *yourself* and try to solve them on your own. People learn mathematics best by doing it.

Collaboration policy: This assignment is *individual* work, and you are required to submit your own solutions. You can *collaborate* with peers in the form of brainstorming, working together, and cross-checking of solutions. However, *you must explicitly state who you worked with* and, crucially, *you must write up your solutions yourself*—they cannot be a copy-paste from someone else's hand-in.

The **deadline** for submitting the first assignment is **Wednesday, April 30th 2025, at 23:59**.

Submission instructions: Submit the pdf with your solutions on Moodle. The solution to each question should be on a new page of the pdf. While not strictly necessary, we highly advise you to use the official Assignment 1 Submission template when preparing your submission. It also includes a large number of LaTeX macros which can make your writing faster and easier to read. **Important**: Even if you do not use the template, you should copy the Declaration of Originality from the front page into your own submission!

You can obtain at most 100 points in this assignment. The grading scheme is the following:

Final assignment grade	Total number of points
6.0	$[96, 100]$
5.75	$[92, 96)$
5.5	$[86, 92)$
5.25	$[82, 86)$
5.0	$[72, 82)$
4.75	$[64, 72)$
4.5	$[58, 64)$
4.25	$[50, 58)$
4.0	$[40, 50)$

Some questions include *bonus* sub-questions. Solving these can help recover points lost elsewhere in the assignment, and they are *not* required to reach the full 100 points—you can reach 100 points by just (correctly) solving non-bonus questions. Bonus points do not carry over to Assignments 2 and 3.

[1]If we can prove you copied, we will report you to the rectorate. Beware!

Question 1: Representation Surgery (31 + 5 pts)

In this question, we will explore **concept erasure**—the systematic removal of specific information from a machine learning model's internal representations. This problem arises in various contexts. In interpretability research, for instance, researchers may want to determine whether a model relies on a particular concept to solve a given task. To test this, they might remove part-of-speech information, for example, from the model's representations and evaluate its performance on syntactic tasks. Another important application is fairness. For example, in language models, gender information embedded in word representations can lead to biased completions (e.g., associating "nurse" with female pronouns and "engineer" with male pronouns). Removing gender-related information from the language model's representations can help mitigate such biases while preserving the overall utility of the model.

We now formalize the concept erasure problem by introducing the relevant notation. Let $\mathbf{X}$ be a representation-valued random variable, where each representation is a real-valued vector in $\mathbb{R}^d$. These representations might, for example, come from a representations-based language model, i.e., $\mathbf{X} = \boldsymbol{h}(\boldsymbol{y})$ for a string $\boldsymbol{y} \in \Sigma^*$ and a representation function $\boldsymbol{h}\colon \Sigma^* \to \mathbb{R}^d$. Such a language model can then define $p_{\text{SM}}(y \mid \boldsymbol{y}) \overset{\text{def}}{=} \operatorname{softmax}(\mathbf{E}\mathbf{X})_y$ for some output matrix $\mathbf{E} \in \mathbb{R}^{|\Sigma| \times d}$.

Let Z be a categorical random variable, denoting the target concept, which can take k values. For example, in the case of binary gender, Z can take two values, i.e., $\mathcal{Z} \overset{\text{def}}{=} \{0, 1\}$.

In this assignment, we will *guard* our representations from a linear predictor that attempts to predict Z using the representations. Let $f : \mathbb{R}^d \to \mathbb{R}^k$ be a linear predictor, chosen from the set of all linear predictors $\mathcal{V} = \{(\mathbf{A}, \mathbf{b}) : \mathbf{A} \in \mathbb{R}^{k \times d}, \mathbf{b} \in \mathbb{R}^k\}$, where $f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$. Furthermore, let $\mathcal{L}$ be the cross-entropy loss between observed representation–concept pairs. For an observed representation $\mathbf{x}$, let $\mathbf{y}$ be the predicted value from the linear classifier, i.e., $\mathbf{y} \overset{\text{def}}{=} \mathbf{A}\mathbf{x} + \mathbf{b}$. The cross-entropy loss for this datapoint is defined as follows:

$$\mathcal{L}(\mathbf{y}, Z = i) = -\log \frac{\exp(y_i)}{\sum_{j=1}^{k} \exp(y_j)}. \tag{1}$$

Before formally defining the linear guardedness, we need to first define the trivially attainable loss.

Definition 1 (Trivially Attainable Loss). *The trivially attainable loss for predicting concepts Z, using the cross-entropy loss $\mathcal{L}$, denoted with L_τ, is the lowest possible expected loss achieved by a constant predictor $f(\mathbf{x}) = \mathbf{b}$, i.e.,*

$$L_\tau = \inf_{\mathbf{b} \in \mathbb{R}^k} \mathbb{E}[\mathcal{L}(\mathbf{b}, Z)]. \tag{2}$$

We call the predictor that achieves the trivially attainable loss the trivial predictor.

Definition 2. *We say $\mathbf{X}$ linearly guards Z with respect to loss $\mathcal{L}$ if the trivially attainable loss is optimal over all affine transformations, i.e., there is no affine classifier that can achieve a better loss than L_τ. In other words,*

$$\mathbb{E}\left[\mathcal{L}\left(f\left(\mathbf{X}\right), Z\right)\right] \geq L_\tau \tag{3}$$

for any affine predictor f.

Definition 3 (Concept전 Erasure). *A representation $\mathbf{X}$ naturally contains information about the concept Z, and one can train a classifier to predict Z from $\mathbf{X}$.*

***Concept erasure** consists of removing that information from $\mathbf{X}$ by creating a map*

$$r : \mathbb{R}^d \longrightarrow \mathbb{R}^d \tag{4}$$

that minimally alters $\mathbf{X}$ such that no classifier $c\colon r(\mathbf{x}) \mapsto c(r(\mathbf{x})) \in \mathcal{Z}$ can achieve a loss lower than the trivially attainable loss, and we say that $r(\mathbf{X})$ guards Z.[2]

*We say that concept erasure is **linear** if r is an affine function.*

[2]Trivially, the constant map $r(\mathbf{x}) = \mathbf{0}$ satisfies this requirement, but the goal is to find an r that alters $\mathbf{X}$ as little as possible.

a) **(1 pt)** Suppose that $\mathbf{X}$ linearly guards Z. Does that mean that no classifier $c\colon \mathbf{x} \mapsto c(\mathbf{x}) \in \mathcal{Z}$ can achieve a loss lower than the trivially attainable loss?

The goal of the next 3 sub-questions is to prove that $\mathbf{X}$ linearly guards Z if and only if the class-conditional mean for any concept $Z = i$ is equal to the global mean, i.e, $\mathbb{E}[\mathbf{X} \mid Z = i] = \mathbb{E}[\mathbf{X}]$.

b) **(3 pt)** Prove that if each class-conditional mean $\mathbb{E}[\mathbf{X} \mid Z = i] = \mathbb{E}[\mathbf{X}]$, then $\mathbf{X}$ linearly guards Z with respect to the cross-entropy loss $\mathcal{L}$.

c) **(3 pt)** Let $\mathbf{x}$ be an observed representation, and $\mathbf{y} \overset{\text{def}}{=} \mathbf{A}\mathbf{x} + \mathbf{b}$.

1. Prove that all partial derivatives of the cross-entropy loss with respect to $\mathbf{y}$ are bounded.
2. Show that for any $j_1, j_2 \neq i$, we have

$$\frac{\partial}{\partial y_i}\mathcal{L}(\mathbf{y}, Z = j_1) = \frac{\partial}{\partial y_i}\mathcal{L}(\mathbf{y}, Z = j_2) \neq 0. \tag{5}$$

d) **(5 pt, *bonus*)** Assume now that $\mathbf{X}$ linearly guards Z and that furthermore the infimum in Eq. (2) is reached by some constant predictor $f(\mathbf{X}) = \mathbf{b}^* \in \mathbb{R}^K$. Assuming that all class probabilities are non-zero, i.e., $\mathbb{P}(Z = i) \neq 0$, prove that each class-conditional mean $\mathbb{E}[\mathbf{X} \mid Z = i]$ is equal to $\mathbb{E}[\mathbf{X}]$.

Hint: Write the first-order optimality condition of the linear predictor to minimize the cross-entropy loss. Then use the boundedness from the previous question together with the dominated convergence theorem to interchange the differentiation and expectation.

e) **(2 pt)** Let $\widetilde{\mathbf{Z}}$ represent the one-hot encoding of concepts. In other words, $\widetilde{\mathbf{Z}}$ is a random vector of concepts taking values in $\{0,1\}^k$, where the j^{th} entry of $\widetilde{\mathbf{Z}}$ is 1 if and only if $Z = j$. Assume that all concept probabilities are nonzero: $\mathbb{P}(Z = i) \neq 0$. Prove that the class conditional means $\mathbb{E}[\mathbf{X} \mid Z = i]$ are equal to the global mean $\mathbb{E}[\mathbf{X}]$ if and only if the cross-covariance between $\mathbf{X}$ and $\widetilde{\mathbf{Z}}$ is zero. Note that the cross-covariance is a matrix denoted with $\mathbf{\Sigma}_{\mathbf{X},\widetilde{\mathbf{Z}}}$ whose $(i,j)^{\text{th}}$ entry is $\mathrm{Cov}(X_i, \widetilde{\mathbf{Z}}_j)$.

Linear Concept Erasure. We've just established that $\mathbf{X}$ linearly guards the concept Z if and only if $\mathrm{Cov}(\mathbf{X}, \widetilde{\mathbf{Z}}) = \mathbf{0}$. Our goal now is to construct an affine map that removes information about Z from $\mathbf{X}$ with minimal disruption to $\mathbf{X}$. This is motivated by the desideratum of removing the information about Z while keeping all the remaining information from $\mathbf{X}$ intact. In the following sub-questions, we will prove that an affine map is sufficient, characterize valid affine maps, and identify the optimal map that minimally changes $\mathbf{X}$.

We consider the affine map

$$r(\mathbf{X}) = \mathbf{P}\,\mathbf{X} + \mathbf{b}, \quad \text{where } \mathbf{P} \in \mathbb{R}^{d\times d} \text{ and } \mathbf{b} \in \mathbb{R}^d, \tag{6}$$

f) **(2 pt)** Using the linearity of covariance, prove that $r(\mathbf{X})$ is linearly guarded if and only if

$$\mathbf{P}\,\mathbf{\Sigma}_{\mathbf{X},\widetilde{\mathbf{Z}}} = \mathbf{0}. \tag{7}$$

g) **(4 pt)** Define the distortion induced by an affine map as the expected Euclidean distance

$$D(\mathbf{P}, \mathbf{b}) = \mathbb{E}\left[\|\mathbf{P}\mathbf{X} + \mathbf{b} - \mathbf{X}\|^2\right] \tag{8}$$

(i) Prove that if $\mathbb{E}[\mathbf{X}] = \mathbf{0}$, then $D(\mathbf{P}, \mathbf{b}) = \mathbb{E}\left[\|\mathbf{P}\mathbf{X} - \mathbf{X}\|^2\right] + \|\mathbf{b}\|^2$.

(ii) In the next exercise you will minimize $D(\mathbf{P}, \mathbf{b})$ subject to a constraint on $\mathbf{P}$. Use (i) to show that if $\mathbb{E}[\mathbf{X}] = \mathbf{0}$, it is equivalent to minimize $D(\mathbf{P}) = \mathbb{E}\left[\|\mathbf{P}\mathbf{X} - \mathbf{X}\|^2\right]$.

Hint: $\|\mathbf{a} + \mathbf{b}\|^2 = \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 + 2\langle \mathbf{a}, \mathbf{b}\rangle$.

Thus, we can simplify Eq. (8) by taking $\mathbf{X} - \mathbb{E}[\mathbf{X}]$. Now, we only need to find the optimal $\mathbf{P}$ denoted with $\mathbf{P}^*$ that linearly guards Z by satisfying Eq. (7), which also leads to minimal disruption, i.e., minimizing *Eq.* (8). We first recall from linear algebra:

- The null space of $\mathbf{P}$ (denoted as $\ker(\mathbf{P})$) is the subspace of all vectors $\mathbf{v}$ such that $\mathbf{P}\mathbf{v} = \mathbf{0}$. Note that $\mathbf{P} \in \mathbb{R}^{d\times d}$ satisfies $\mathbf{P}\,\boldsymbol{\Sigma}_{\mathbf{X},\widetilde{\mathbf{Z}}} = \mathbf{0}$ if and only if every column of $\boldsymbol{\Sigma}_{\mathbf{X},\widetilde{\mathbf{Z}}}$ lies in $\ker(\mathbf{P})$:

$$\mathbf{P}\,\boldsymbol{\Sigma}_{\mathbf{X},\widetilde{\mathbf{Z}}} = \mathbf{0} \quad\Longleftrightarrow\quad \operatorname{colsp}(\boldsymbol{\Sigma}_{\mathbf{X},\widetilde{\mathbf{Z}}}) \subseteq \ker(\mathbf{P}). \tag{9}$$

  Linear maps are completely characterized by how they act on orthogonal complements. That is, since $\mathbb{R}^d = \operatorname{colsp}(\boldsymbol{\Sigma}_{\mathbf{X},\widetilde{\mathbf{Z}}}) \oplus \operatorname{colsp}(\boldsymbol{\Sigma}_{\mathbf{X},\widetilde{\mathbf{Z}}})^{\perp}$, we only need to chracterize how $\mathbf{P}$ acts on $\operatorname{colsp}(\boldsymbol{\Sigma}_{\mathbf{X},\widetilde{\mathbf{Z}}})^{\perp}$.

h) **(4 pt)** Assume that the representation $\mathbf{X}$ is centered (i.e., $\mathbb{E}[\mathbf{X}] = \mathbf{0}$) and that its covariance is the identity, $\boldsymbol{\Sigma}_{\mathbf{X},\mathbf{X}} = \mathbf{I}$. Prove that

$$\mathbf{P}^* = \mathbf{I} - \boldsymbol{\Sigma}_{\mathbf{X},\widetilde{\mathbf{Z}}}\,\boldsymbol{\Sigma}_{\mathbf{X},\widetilde{\mathbf{Z}}}{}^{+} \tag{10}$$

both satisfies Eq. (7) and minimizes Eq. (8). $\boldsymbol{\Sigma}_{\mathbf{X},\widetilde{\mathbf{Z}}}{}^{+}$ is the Moore–Penrose pseudoinverse of $\boldsymbol{\Sigma}_{\mathbf{X},\widetilde{\mathbf{Z}}}$.

Hint 1: Start from Eq. (8). Note that $\mathbb{E}\big[\|\mathbf{A}\mathbf{X}\|^2\big] = \operatorname{tr}(\mathbf{A}\boldsymbol{\Sigma}_{\mathbf{X},\mathbf{X}}\mathbf{A}^{\top})$.

Hint 2 (Projection Theorem in matrix form): Suppose $\mathbf{A} \in \mathbb{R}^{d\times r}$ spans a subspace $V = \operatorname{colsp}(\mathbf{A}) \subseteq \mathbb{R}^d$. The orthogonal projections onto V and $V^{\perp}$ are given by

$$\Pi_V \;=\; \mathbf{A}\,\mathbf{A}^{+}, \quad \Pi_{V^\perp} = \mathbf{I} - \mathbf{A}\,\mathbf{A}^{+}, \tag{11}$$

In particular, for any vector $\mathbf{y} \in \mathbb{R}^d$, $\Pi_V\,\mathbf{y} = \mathbf{A}\,\mathbf{A}^{+}\,\mathbf{y}$ is the closest point to $\mathbf{y}$ in the subspace $\operatorname{colsp}(\mathbf{A})$. Likewise, projecting into $V^{\perp}$ yields the closest point to $\mathbf{y}$ in the subspace $\operatorname{colsp}(\mathbf{A})^{\perp}$.

i) **(5 pt)** Extend the result to the setting where $\mathbf{X}$ may have nonzero mean $\mathbb{E}[\mathbf{X}] \neq \mathbf{0}$ and arbitrary covariance $\boldsymbol{\Sigma}_{\mathbf{X},\mathbf{X}} \neq \mathbf{I}$. In particular, define the whitening matrix

$$\mathbf{W} \;=\; (\boldsymbol{\Sigma}_{\mathbf{X},\mathbf{X}}{}^{1/2})^{+}, \tag{12}$$

so that $\mathbf{W}\mathbf{X}$ has unit covariance again. Prove that the full transformation is

$$r(\mathbf{x}) \;=\; \mathbf{x} \;-\; \mathbf{W}^{+}\,\Pi_{\mathbf{W}\boldsymbol{\Sigma}_{\mathbf{X},\widetilde{\mathbf{Z}}}}\,\mathbf{W}\,\Big(\mathbf{x} - \mathbb{E}[\mathbf{X}]\Big), \tag{13}$$

where

$$\Pi_{\mathbf{W}\boldsymbol{\Sigma}_{\mathbf{X},\widetilde{\mathbf{Z}}}} \;=\; (\mathbf{W}\,\boldsymbol{\Sigma}_{\mathbf{X},\widetilde{\mathbf{Z}}})\,(\mathbf{W}\,\boldsymbol{\Sigma}_{\mathbf{X},\widetilde{\mathbf{Z}}})^{+} \tag{14}$$

is the orthogonal projection (in the whitened space) onto the subspace encoding concept information.

j) **(7 pt)** In this sub-question, you will use representational surgery procedures developed in the previous sub-questions to implement linear concept erasure on a real-world dataset of string representations. See the following Google Colab notebook and follow the instructions there:

https://colab.research.google.com/drive/1FRn_1yS6AajTEpEB8ftyEduQVbWr8eoK

For submission, simply include the pdf-exported notebook as part of your submission. For easier grading, please concatenate the notebook pdf to the end of your solution of Question 1.

Question 2: Properties of Finite-state Language Models (18 pts)

The elegance of simple formalisms in language modeling becomes particularly evident when we examine finite-state automata. One of their key advantages lies in the ability to compute many important quantities with polynomial-time complexity, allowing for exact rather than approximate solutions.

Let $\mathcal{A} = (\Sigma, Q, \delta, \lambda, \rho)$ be a probabilistic finite-state automaton (PFSA), with no ε-transitions, encoding a finite-state language model p_{LM}.

Definition 4. *Transition Matrix For any symbol $y \in \Sigma$, we define the **symbol-specific transition matrix** $\mathbf{T}^{(y)} \in \mathbb{R}^{|Q|\times|Q|}$ as*

$$T^{(y)}_{q,q'} \overset{\text{def}}{=} \sum_{q \xrightarrow{y/w} q'} w, \tag{15}$$

*where we index the matrix elements directly with the states for convenience. The **transition matrix** $\mathbf{T}$ is then defined as $\mathbf{T} \overset{\text{def}}{=} \sum_{y\in\Sigma} \mathbf{T}^{(y)}$.*

We assume throughout that all eigenvalues of $\mathbf{T}$ have magnitude less than 1.

Further, we define $\boldsymbol{\lambda} \in \mathbb{R}^{|Q|}$ as $\lambda_q \overset{\text{def}}{=} \lambda(q)$ and $\boldsymbol{\rho} \in \mathbb{R}^{|Q|}$ as as $\rho_q \overset{\text{def}}{=} \rho(q)$

This exploration focuses on deriving efficient algorithms for three fundamental quantities that characterize our language model p_{LM}:

(1) The expected string length, μ:

$$\mu \overset{\text{def}}{=} \sum_{\boldsymbol{y}\in\Sigma^*} p_{\text{LM}}(\boldsymbol{y})\, |\boldsymbol{y}| \tag{16}$$

(2) The expected square deviation from the mean length (the variance), σ^2:

$$\sigma^2 \overset{\text{def}}{=} \sum_{\boldsymbol{y}\in\Sigma^*} p_{\text{LM}}(\boldsymbol{y})\, (|\boldsymbol{y}| - \mu)^2 \tag{17}$$

(3) The entropy of the language model, $\mathrm{H}(p_{\text{LM}})$:

$$\mathrm{H}(p_{\text{LM}}) \overset{\text{def}}{=} - \sum_{\boldsymbol{y}\in\Sigma^*} p_{\text{LM}}(\boldsymbol{y}) \log p_{\text{LM}}(\boldsymbol{y}) \tag{18}$$

These quantities provide valuable insights into the model's behavior. The entropy, in particular, serves as a widely used statistic that characterizes the diversity of the language. A low entropy indicates a more predictable model, while a high entropy suggests a more varied distribution of possible strings. This property makes entropy particularly useful for:

- Regularizing the model to control generation diversity
- Measuring the difference between two probability distributions through divergence metrics such as the Kullback-Leibler (KL) divergence. In other words, models with more similar distributions will have a lower KL divergence, while greater differences result in a higher divergence.

In the following sections, we will derive efficient algorithms to compute these quantities exactly, demonstrating the practical advantages of finite-state language models.

a) **(8 pts)** Give an algorithm for computing the mean string length under p_{LM}. The algorithm should run in time $\mathcal{O}\left(|Q|^3\right)$. Show that your algorithm indeed runs in time $\mathcal{O}\left(|Q|^3\right)$.
Hint: This problem requires a simple modification to the allsum algorithm from the notes; indeed, if you set up the problem correctly, you can call the allsum algorithm as a black box.

b) **(4 pts)** Give an algorithm for computing the variance of the string length under p_{LM}. The algorithm should run in time $\mathcal{O}|Q|^3$. Show that your algorithm indeed runs in time $\mathcal{O}|Q|^3$.
Hint: You may wish to introduce a form of mean-centering by introducing negative lengths for certain strings.

c) **(6 pts)** In this sub-question, we additionally assume that $\mathcal{A}$ is *deterministic*. Prove that, in that case, the entropy p_{LM} is:

$$\mathrm{H}\left(p_{\text{LM}}\right) = \boldsymbol{\lambda}^\top (\mathbf{I} - \mathbf{T})^{-1} \boldsymbol{\xi}, \tag{19}$$

where $\boldsymbol{\xi} \in \mathbb{R}^{|Q|}$ is a vector defined as:

$$\xi_q = - \sum_{q \xrightarrow{y/w} q'} w \log w - \rho_q \log \rho_q. \tag{20}$$

Hint: You may want to define two random variables, Π_q and τ_q. Π_q ranges over paths in $\mathcal{A}$ starting in q and τ_q over outgoing transitions of q or termination. More concretely,

$$\mathbb{P}\left(\Pi_q = q \xrightarrow{y_1/w_1} \cdots \xrightarrow{y_\ell/w_\ell} q_\ell\right) = \left[\prod_{j=1}^{\ell} w_j\right] \rho_{q_\ell} \tag{21}$$

and

$$\mathbb{P}\left(\tau_q = \boldsymbol{t}\right) = \begin{cases} w & \textbf{if } \boldsymbol{t} = q \xrightarrow{y/w} q' \\ \rho_q & \textbf{otherwise} \ \ \text{(termination)} \end{cases}. \tag{22}$$

Then, express the entropy $\mathrm{H}(\tau_q, \Pi_{q'})$ using the chain rule of entropy:

$$\mathrm{H}(\mathrm{x}, \mathrm{y}) = \mathrm{H}(\mathrm{x}) + \mathrm{H}(\mathrm{y} \mid \mathrm{x}) = \mathrm{H}(\mathrm{x}) + \sum_x p(x) \mathrm{H}(\mathrm{y} \mid \mathrm{x} = x) \tag{23}$$

and use this to compute the entropy over *strings* (cf. Eq. (18)).
Hint: What is the relationship between $\mathrm{H}(\Pi_q)$ and $\mathrm{H}(\tau_q, \Pi_{q'})$?

Question 3: Efficiently Simulating Bounded Stacks with RNNs (22 + 6 pts)

Introduction

In this question, we examine the capacity of RNN language models to recognize a variant of the Dyck languages of nested parentheses of k types, $\mathrm{D}(k)$. To recognize $\mathrm{D}(k)$, a model must remember any currently unclosed opening brackets and ensure they close in the correct order; once pairs are closed, they can be "forgotten", or removed from memory. Therefore, the memory needed to recognize any string in $\mathrm{D}(k)$, is proportional to the number of unclosed brackets at any given time. We formalize it by counting how many more open brackets than closed brackets there are at each time step in the string:

$$d(\boldsymbol{y}_{\leq t}) \stackrel{\text{def}}{=} \mathtt{count}(\boldsymbol{y}_{\leq t}, \langle) - \mathtt{count}(\boldsymbol{y}_{\leq t}, \rangle) \tag{24}$$

where $\mathtt{count}(\boldsymbol{y}_{\leq t}, \langle)$ refers to the number of times *any* opening bracket occurs in $\boldsymbol{y}_{\leq t}$ and $\mathtt{count}(\boldsymbol{y}_{\leq t}, \rangle)$ the number of times *any* closing bracket occurs in $\boldsymbol{y}_{\leq t}$.

While $\mathrm{D}(k)$ describes arbitrarily deep hierarchical structures, natural languages exhibit bounded nesting in practice. In this question, we investigate how to represent $\mathrm{D}(k)$ languages which can only nest up to some *bounded* depth m. We denote such languages as $\mathrm{D}(k, m)$.

Definition 5 ($\mathrm{D}(k, m)$ languages)**.** *Let $k, m \in \mathbb{N}_{\geq 1}$. We define the* ***bounded Dyck language*** $\mathrm{D}(k, m)$ *by combining* $\mathrm{D}(k)$ *with a bound on the nesting depth:*

$$\mathrm{D}(k, m) \stackrel{\text{def}}{=} \{\boldsymbol{y} \in \mathrm{D}(k) \mid d(\boldsymbol{y}_{\leq t}) \leq m, t = 1, \ldots, T\}, \tag{25}$$

where $T = |\boldsymbol{y}|$ corresponds to the length of the string.

This question is divided into three parts: In the first part, you will prove that $\mathrm{D}(k, m)$ languages are finite-state. In the second part, you will construct an Elman RNN simulating an FSA recognizing $\mathrm{D}(k, m)$ languages. In the third part, you are asked to show that the RNN indeed recognizes the language.

Part I. $\mathrm{D}(k, m)$ is Finite-State

Due to their bounded nesting depth, $\mathrm{D}(k, m)$ languages can be recognized by stacks of bounded depth. We begin with a warm-up question.

a) **(1 pt)** Suppose you are given a stack that can be used to recognize $\mathcal{L} = \mathrm{D}(2, 3)$. The current stack configuration $\boldsymbol{\gamma}$ is given as:

$$\begin{aligned} \boldsymbol{\gamma}_1 &= \langle_2 \\ \boldsymbol{\gamma}_2 &= \langle_1\langle_1\langle_1 \\ \boldsymbol{\gamma}_3 &= \langle_2\langle_1\langle_2 \end{aligned}$$

Determine the new stack configurations $\boldsymbol{\gamma}_1'$, $\boldsymbol{\gamma}_2'$, $\boldsymbol{\gamma}_3'$ after reading in each of the following symbols (each one starting from a stack $\boldsymbol{\gamma}_1$, $\boldsymbol{\gamma}_2$, $\boldsymbol{\gamma}_3$, *not* one after another)?

1. $\rangle_2$
2. $\rangle_1$
3. $\langle_2$

Specify the *nine* stack configurations in the following table. Use $\emptyset$ to denote an empty stack and "reject" if the automaton would reject a string.

	$\gamma_1 = \langle_2$	$\gamma_2 = \langle_1\langle_1\langle_1$	$\gamma_3 = \langle_2\langle_1\langle_2$
$\rangle_2$			
$\rangle_1$			
$\langle_2$			

Table 1: The first row of the table represents different stack configurations, while the first column lists the newly read symbols.

$\mathrm{D}(k, m)$ languages can be shown to be finite-state since they are recognized by stacks with bounded depth, and thus with bounded memory. One way to prove this property is by constructing an FSA recognizing $\mathrm{D}(k, m)$.

b) **(4 pts)** Let $k, m \in \mathbb{N}_{\geq 1}$. Construct a finite-state automaton

$$\mathcal{A} = (\Sigma, Q, \delta, I, F) \tag{26}$$

by defining the elements of the tuple such that $\mathcal{A}$ accepts the $\mathrm{D}(k, m)$, i.e. $\mathcal{L}(\mathcal{A}) = \mathrm{D}(k, m)$.

Prove that your constructed automaton really recognizes exactly the required languages and give a big-$\mathcal{O}$ bound on the number of states and transitions in $\mathcal{A}$ in terms of k and m.

Note: We are only interested in binary recognition of the language. All the elements of the tuple have to be defined formally, but you can reason about correctness more informally. Use the notation $[\boldsymbol{y}]$ to denote the state corresponding to the stack sequence $\boldsymbol{y}$. Avoid introducing an EOS symbol, as the automaton's final states inherently represent termination.

Hint: The automaton should encode each possible configuration of the bounded stack as a separate state and define the outgoing transitions based on what continuations of the string encoded by the stack are possible. For example, in $\mathrm{D}(2, 2)$, the only "valid" accessible state from $[\langle_1\langle_2]$ would be $[\langle_1]$ after reading the input symbol $\rangle_2$. You can use a designated "accept" state (the only final state) A and a "rejecting state" R.

Part II. Constructing RNN Dynamics

We now construct an Elman RNN that simulates a bounded stack and recognizes $\mathrm{D}(k, m)$ languages. A stack bounded to m symbols can encode up to k^m different configurations (this many different *states*). Recall that, to represent such a language with an Elman RNN through Minsky's construction directly, we would require a hidden state of size $\mathcal{O}(k^m)$—this is untractable even for small k and m. However, $\mathrm{D}(k, m)$ languages have a very clear and useful structure which makes them efficiently representable using an RNN.

We will describe how the RNN hidden states represent the states of the FSA $\mathcal{A}$ recognizing $\mathrm{D}(k, m)$ and how the recurrence relation of the RNN encodes the FSA's transition function. This will lead to an RNN with the hidden state size of $\mathcal{O}(mk)$—a vast improvement on the one based on Minsky's construction.

As mentioned, when processing strings in $\mathrm{D}(k, m)$, we only have to keep the opening brackets on the stack as the summary of the entire string so far, $\boldsymbol{y}_{<t}$, to be able to determine which next symbols are allowed. We, therefore, choose to encode a *string* $\boldsymbol{y}_{<t}$ by encoding the *stack* it induces in the hidden state (*context encoding*) of the RNN as follows. We define the column vector

$$\boldsymbol{h}\left([\langle_{i_1}\langle_{i_2}\cdots\langle_{i_{m'}}]\right) \stackrel{\text{def}}{=} \begin{pmatrix} \mathbf{e}_{i_{m'}} \\ \mathbf{e}_{i_{m'-1}} \\ \vdots \\ \mathbf{e}_{i_2} \\ \mathbf{e}_{i_1} \\ \mathbf{0}_{\in(m-m')k} \end{pmatrix} \in \mathbb{R}^{k\cdot m}, \tag{27}$$

where $\mathbf{e}_n$ represents the n^{th} canonical basis vector. The indices $i_j \in \{1, \ldots, k\}$ refer to the identities of the brackets in Σ and m' refers to the current depth of the stack. The element encoded by the "top" $(i_{m'})$ of the hidden state, $\mathbf{e}_{i_{m'}}$, represents the symbol on the top of the stack.

c) **(1 pt)** What are the embeddings $\boldsymbol{h}(\gamma_i)$ for $i = 1, 2, 3$ from question *(a)*? What are the embeddings of the new stacks after reading in the same new inputs as in question *(a)*?

	$\gamma_1 = \langle_2$	$\gamma_2 = \langle_1\langle_1\langle_1$	$\gamma_3 = \langle_2\langle_1\langle_2$
$\rangle_2$			
$\rangle_1$			
$\langle_2$			

Table 2: The first row of the table represents different stack configurations, while the first column lists the newly read symbols.

We now start building the RNN recognizing the $\mathrm{D}(k, m)$ language.

Instruction: For the remainder of this exercise, represent matrices entry-wise or graphically; a verbal description alone is insufficient. Make sure to include the EOS symbol, and do not use $\pm\infty$ as a matrix entry. When using symbols in an ordered way, always follow this structure: First the opening brackets, then the closing brackets, and finally the EOS symbol:

$$\langle_1, \ldots, \langle_k, \rangle_1, \ldots, \rangle_k, \text{EOS}. \tag{28}$$

d) **(5 pts)** We first consider a more flexible "RNN" model *without* an activation function, in which the recurrence matrix $\mathbf{U}$ *depends* on the current input. We model the dynamics map as

$$\mathbf{h}_t = \mathbf{U}(y_t)\, \mathbf{h}_{t-1} + \mathbf{V}\mathbf{o}_{y_t}, \tag{29}$$

where $\mathbf{U} : \overline{\Sigma} \to \mathbb{R}^{k \cdot m \times k \cdot m}$ is a *function* of the current input y_t, and $\mathbf{V} \in \mathbb{R}^{k \cdot m \times (2 \cdot k + 1)}$. $\mathbf{o}_{y_t}$ refers to the one-hot-encoding of the symbol y_t. Given the encoding from Eq. (27), define the recurrence matrix function $\mathbf{U}(\cdot)$ and the input matrix $\mathbf{V}$ such that, assuming that $\mathbf{h}_{t-1}$ encodes some state $q \in Q$ (and thus the stack at time step $t - 1$), $\mathbf{h}_t$ encodes the state $q' \in Q$ such that $\mathcal{A}$ transitions into q' from q upon reading y_t (and thus $\mathbf{h}_t$ encodes the state of the stack at time t). Show that the dynamics map you constructed works as desired by showing how the transitions in the hidden state space of the RNN correspond to the transitions in the automaton $\mathcal{A}$.

Hint: You need to figure out how to push and pop from the stack represented using Eq. (27) through matrix multiplications. $\mathbf{U}(y)$ should be a simple function of the input symbol depending only on whether the stack should be *popped from* or *pushed onto*.

e) **(6 pts)** Now, suppose that we have a standard Elman RNN, in which the recurrence matrix is *fixed*:

$$\mathbf{h}_t = H\left(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{V}\mathbf{o}_{y_t} + \mathbf{b}\right), \tag{30}$$

where H refers to the Heaviside activation function as defined in the lecture. Define the parameters $\mathbf{U} \in \mathbb{R}^{2 \cdot k \cdot m \times 2 \cdot k \cdot m}$, $\mathbf{V} \in \mathbb{R}^{2 \cdot k \cdot m \times (2 \cdot k + 1)}$ and $\mathbf{b} \in \mathbb{R}^{2 \cdot k \cdot m}$ such that the update steps of the RNN again correspond to the transitions in the automaton $\mathcal{A}$ as in the previous subquestion.

Show that the dynamics map you constructed works as desired by showing how the transitions in the hidden state space of the RNN correspond to the transitions in the automaton $\mathcal{A}$.

Hint: Since the recurrence matrix $\mathbf{U}$ cannot depend on whether a symbol should be pushed or popped anymore, consider performing *both* operations at once.

Part III. Recognition with an RNN

We now have an Elman RNN whose hidden state updates correspond to the transitions of the automaton $\mathcal{A}$ recognizing $\mathrm{D}(k, m)$! The next step is to figure out a way to make it "recognize" $\mathrm{D}(k, m)$ formally.

RNNs normally define string probabilities, i.e. weighted languages. We are studying the *binary* acceptance of *unweighted* languages. For that, we must define what it means for an RNN to generate an *unweighted* language, such as $\mathrm{D}(k,m)$. The key idea is to define recognition in terms of the *local* next symbol probabilities $p_{\mathrm{SM}}(y_t \mid \boldsymbol{y}_{<t})$ instead of the full string probabilities $p(\boldsymbol{y})$—those must approach zero with sequence length.

Definition 6 (Locally η-truncated support)**.** *Let $\eta \in [0,1]$. The **locally η-truncated support** of a sequence model p_{SM} is the set*

$$\mathcal{L}_\eta\left(p_{SM}\right) \overset{\text{def}}{=} \{\boldsymbol{y} \in \Sigma^* \colon p_{SM}(y_t \mid \boldsymbol{y}_{<t}) \geq \eta, t = 1 \dots |\boldsymbol{y}|\}.$$

This is the set of strings for which the model assigns at least η probability to each symbol conditioned on the string before it. Intuitively, these are the strings for which the model does not strongly "disagree" (assign probability $< \eta$) with the next symbol at any point.

f) **(5 pts)** Given the stack encoding from Eq. (27) and one-hot encodings $\mathbf{o}_y$ of the input symbols, construct the output matrix $\mathbf{E}$ and the bias vector $\mathbf{b}'$ such that

$$\mathcal{L}_\eta\left(p_{\mathrm{SM}}\right) = \mathrm{D}(k,m), \tag{31}$$

where p_{SM} is the sequence model whose conditional probabilities are defined by the constructed RNN, i.e.,

$$p_{\mathrm{SM}}\left(y_t \mid \boldsymbol{y}_{<t}\right) = \operatorname{softmax}\left(\mathbf{E}\,\mathbf{h}_{t-1} + \mathbf{b}'\right)_{y_t}. \tag{32}$$

Introduce a real-valued parameter α into your definition of the output matrix $\mathbf{E}$ which will allow you to control the conditional distributions defined by your constructed RNN. Specify the logits you obtain for acceptable continuations and not valid ones. Then, reason how to set or bound the parameter α by computing the different possible normalization constants (there are three relevant ones). Reason shortly why your method works.[3]

Note: To satisfy the definition of η-truncated support, you should show that $p_{\mathrm{SM}}\left(y_t \mid \boldsymbol{y}_{<t}\right) \geq \eta$ if and only if the symbol y_t is an "acceptable" continuation of the input string $\boldsymbol{y}_{<t}$. For example, since $\langle_2$ is not a valid continuation of the string $\langle_2\langle_1\langle_1\rangle_1\langle_2$, it should hold that $p_{\mathrm{SM}}\left(\langle_2 \mid \langle_2\langle_1\langle_1\rangle_1\langle_2\right) < \eta$.

Note: η can be arbitrary, subject to $\eta < \frac{1}{k+1}$ (but it is fixed for a particular RNN). In other words, you cannot "set" η to some value that you find convenient. Therefore, you need to specify α as a function of η.

Part IV. *Bonus*: Reducing the Space Complexity

In the previous sub-questions, we used one-hot encoding to represent different types of opening brackets on the stack. Now, we aim for a more efficient encoding that requires only $\mathcal{O}\left(\log k\right)$ space per symbol. To achieve this, we can simply assign a unique binary encoding from $\{0,1\}^{\lceil \log k \rceil + 1}$ to each opening bracket. Specifically, we encode a bracket $\langle_i$ as the binary representation of the number i, using exactly $\lceil \log k \rceil + 1$ bits. The stack is encoded in the hidden state of the RNN as

$$\boldsymbol{h}\left(\left[\langle_{i_1}\langle_{i_2}\cdots\langle_{i_{m'}}\right]\right) \overset{\text{def}}{=} \begin{pmatrix} \mathbf{e}_{i_{m'}} \\ \mathbf{e}_{i_{m'-1}} \\ \vdots \\ \mathbf{e}_{i_2} \\ \mathbf{e}_{i_1} \\ \mathbf{0}_{\in(m-m')(\lceil \log k \rceil + 1)} \end{pmatrix} \in \mathbb{R}^{(\lceil \log k \rceil + 1)\cdot m}, \tag{33}$$

where $\mathbf{e}_n$ represents the binary representation of n in $\lceil \log k \rceil + 1$ bits. The indices $i_j \in \{1, \dots, k\}$ refer to the identities of the brackets in Σ and m' refers to the current depth of the stack.

[3]Computing the exact expressions for α can be tedious; feel free to reason about the choice less rigorously. Nevertheless, it should be clear from your derivation that "for sufficiently large α", the construction goes through.

g) **(1 pt, *bonus*)** For $\mathcal{L} = \mathrm{D}(4,3)$, the current stack configuration $\boldsymbol{\gamma}$ is given as:

$$\begin{aligned}\gamma_1 &= \langle_2 \\ \gamma_2 &= \langle_1\langle_3\langle_1 \\ \gamma_3 &= \langle_2\langle_1\langle_4\end{aligned}$$

What are the embeddings of the new stacks after reading in the following inputs?

1. $\rangle_1$
2. $\rangle_4$
3. $\langle_3$

	$\gamma_1 = \langle_2$	$\gamma_2 = \langle_1\langle_3\langle_1$	$\gamma_3 = \langle_2\langle_1\langle_4$
$\rangle_1$			
$\rangle_4$			
$\langle_3$			

Table 3: The first row of the table represents different stack configurations, while the first column lists the newly read symbols.

h) **(1 pts, *bonus*)** Suppose that we have a standard Elman RNN

$$\mathbf{h}_t = H\left(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{V}\mathbf{o}_{y_t} + \mathbf{b}\right), \tag{34}$$

where $\mathbf{o}_{y_t}$ still refers to the *one-hot-encoding* of the symbol y_t. How do $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{b}$ change compared to question *(e)*? You can reason about it in words. Specify the dimensions of $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{b}$.

i) **(4 pts, *bonus*)** Construct the output matrix $\mathbf{E}$ and the bias vector $\mathbf{b}'$ such that

$$\mathcal{L}_\eta\left(p_{\text{SM}}\right) = \mathrm{D}(k,m) \tag{35}$$

You will need to extend the encodings of each bracket in a specific way. Specify the logits for valid and invalid continuations. You do not need to specify bounds on α or calculate the normalization constants.
Hint: $\mathbf{E} \in \mathbb{R}^{(2k+1)\times 2\cdot(\lceil \log k\rceil+1)\cdot m}$.

Question 4: Limitations of Transformers on Simple Languages (29 + 5 pts)

In this question, we investigate an inherent limitation of transformers as *recognizers* of languages. We examine this through a deceptively simple task: Recognizing whether a binary string contains an odd number of 1s.

Definition 7 (The `Odd-1` language)**.** *The* **`Odd-1`** *language is defined as*

$$\texttt{Odd-1} \overset{\text{def}}{=} \{\boldsymbol{y} \in \{0,1\}^* \mid \boldsymbol{y} \textit{ contains an odd number of 1s}\}. \tag{36}$$

An example of a string in `Odd-1` is "`10011`" while an example of a string not in this language is "`10010`".

The property of having an odd number of 1s is particularly interesting because it's trivial for humans to check and can be recognized a simple finite-state automaton. Furthermore, recognizing `Odd-1` is a sub-problem of common tasks such as correctly determining the truth value of statements with negations (the truth value depends on whether the number of negations is odd or even) and computing the value $(-1)^n$ (the sign depends on whether n is odd or even). Yet, it reveals a limitation in the computational capabilities of transformers with soft attention.

We consider a simple transformer (classifier) model with L layers and a single head at each layer. In contrast to language models described in the notes, will work with *unmasked* transformers, allowing each token to be contextualized with both past and future tokens. The results, however, can be extended to masked transformers used in language models easily. Similarly, the results can be extended to multi-head transformers with minor modifications (and more obnoxious notation).

Let $D \in \mathbb{N}_{\leq 1}$ be the dimension of the input and output vectors. We denote with

$$\mathbf{X}^{(\ell)} \overset{\text{def}}{=} \left(\mathbf{x}_1^{(\ell)}{}^\top, \mathbf{x}_2^{(\ell)}{}^\top, \ldots, \mathbf{x}_T^{(\ell)}{}^\top\right) \in \mathbb{R}^{T\times D} \tag{37}$$

the sequence of T contextual representations (activations) at layer ℓ.

We define the initial representations to be

$$\mathbf{x}_t^{(0)} \overset{\text{def}}{=} \begin{pmatrix}\mathbf{r}(y_t)\\ \mathbf{u}(t)\end{pmatrix}. \tag{38}$$

Here, $y_t \in \{0,1\}$ for $t < T$, and we assume the final symbol of the input sequence is the end-of-string token $y_T = \textsc{eos}$.[4] Observe that $\mathbf{r}(y_t)$ is bounded since it is a fixed embedding vector of finitely many symbols. Additionally, we *assume* that the positional encoding function $\mathbf{u}(t)$ is bounded.

Definition 8 (The attention mechanism)**.** *Let* $\mathbf{X}^{(\ell)} \overset{\text{def}}{=} \left(\mathbf{x}_1^{(\ell)}{}^\top, \mathbf{x}_2^{(\ell)}{}^\top, \ldots, \mathbf{x}_T^{(\ell)}{}^\top\right)$ *be the sequence of activations at layer ℓ. Let the query, key, and value matrices be* $\mathbf{W}_Q^{(l)}, \mathbf{W}_K^{(l)}, \mathbf{W}_V^{(l)} \in \mathbb{R}^{D\times D}$*, respectively.* ***Dot-product (soft) attention mechanism*** *is defined as follows:*

$$\mathbf{Q}^{(\ell)} \overset{\text{def}}{=} \mathbf{X}^{(\ell)}\mathbf{W}_Q^{(\ell)} = \left(\mathbf{q}_1^{(\ell)}{}^\top, \mathbf{q}_2^{(\ell)}{}^\top, \ldots, \mathbf{q}_T^{(\ell)}{}^\top\right) \in \mathbb{R}^{T\times D}, \tag{39a}$$

$$\mathbf{K}^{(\ell)} \overset{\text{def}}{=} \mathbf{X}^{(\ell)}\mathbf{W}_K^{(\ell)} = \left(\mathbf{k}_1^{(\ell)}{}^\top, \mathbf{k}_2^{(\ell)}{}^\top, \ldots, \mathbf{k}_T^{(\ell)}{}^\top\right) \in \mathbb{R}^{T\times D}, \tag{39b}$$

$$\mathbf{V}^{(\ell)} \overset{\text{def}}{=} \mathbf{X}^{(\ell)}\mathbf{W}_V^{(\ell)} = \left(\mathbf{v}_1^{(\ell)}{}^\top, \mathbf{v}_2^{(\ell)}{}^\top, \ldots, \mathbf{v}_T^{(\ell)}{}^\top\right) \in \mathbb{R}^{T\times D}, \tag{39c}$$

$$\mathbf{s}_t^{(\ell)} \overset{\text{def}}{=} \text{softmax}\left(\mathbf{K}^{(\ell)}\mathbf{q}_t^{(\ell)}\right) \in \mathbb{R}^{T}, \tag{39d}$$

$$\mathbf{z}_t^{(\ell)} \overset{\text{def}}{=} \mathbf{s}_t^{(\ell)}{}^\top \mathbf{V}^{(\ell)} = \sum_{t'=1}^{T} s_{t,t'}^{(\ell)}\mathbf{v}_{t'}^{(\ell)} \in \mathbb{R}^{D}, \tag{39e}$$

$$\mathbf{x}_t^{(\ell+1)} \overset{\text{def}}{=} \mathbf{f}\left(\mathbf{z}_t^{(\ell)}, \mathbf{x}_t^{(\ell)}\right) \in \mathbb{R}^{D}. \tag{39f}$$

Here, $\mathbf{f}$ *is an MLP with pre- and post-residual connections and a non-linear activation function.*

[4] This is in slight contrast to the lecture notes, where we assume that $y_1, \ldots, y_T \in \Sigma$, but we adopt the convention that $y_T = \textsc{eos}$ for easier notation.

Define $\boldsymbol{h}(\boldsymbol{y}) \overset{\text{def}}{=} \mathbf{x}_T^{(L)}$ as the final activation of the EOS token. A **transformer acceptor** takes $\boldsymbol{h}(\boldsymbol{y})$ and maps it to a scalar output using a sigmoid activation function:

$$\hat{y} = \sigma\left(\boldsymbol{h}(\boldsymbol{y})^\top \mathbf{w}\right) \in (0, 1). \tag{40}$$

$\hat{y}$ is the predicted probability that the input string $\boldsymbol{y}$ is in `Odd-1`. Using the predicted probability $\hat{y}$, we can classify whether the string belongs to `Odd-1` language or not,

$$\boldsymbol{y} \in \texttt{Odd-1}, \quad \text{if } \hat{y} \geq 0.5, \tag{41a}$$

$$\boldsymbol{y} \notin \texttt{Odd-1}, \quad \text{otherwise} \tag{41b}$$

We can now proceed with showing of the limitations of the transformer architecture in recognizing `Odd-1`. By solving the following subquestions, you will formalize and prove the following lemma.

Lemma 1. *Let a soft attention transformer be given, and let T be the input length. If we flip one input symbol y_t $(t < T)$, then the change in the resulting activation $\boldsymbol{h}(\boldsymbol{y})$ at the final layer is bounded as $\mathcal{O}\left(\frac{1}{T}\right)$ with constants depending on the parameter matrices.*

We will make use of the Lipschitz continuousness of the individual components of the transformer architecture at multiple points in the proof.

Definition 9. *A function $f\colon \mathbb{R}^d \to \mathbb{R}^m$ is said to be **Lipschitz continuous** with Lipschitz constant L_f if there exists a constant $L_f > 0$ such that for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,*

$$\|f(\mathbf{x}) - f(\mathbf{y})\| \leq L_f \|\mathbf{x} - \mathbf{y}\| \tag{42}$$

You will also find the following definition useful.

Definition 10. *We define $\|\mathbf{X}\|_{2,\infty}$ for $\mathbf{X} \in \mathbb{R}^{D_1 \times D_2}$, defined as the larger row norm of $\mathbf{X}$:*

$$\|\mathbf{X}\|_{2,\infty} = \max_{d=1}^{D_1} \|\mathbf{X}_{i,:}\| \tag{43}$$

where the notation $\|\mathbf{X}\|_{2,\infty}$ suggests that the "inner" norm over rows is the L2-norm (implicit in $\|\cdot\|$).

Now let $\boldsymbol{y} = y_1 \dots y_T$ and $\boldsymbol{y}' = y_1 \dots y_{t^*-1}, y'_{t^*}, y_{t^*+1} \dots y_T$ be two strings that differ only in input at t^*, with $t^* < T$. Let $\delta = \left\|\mathbf{x}_{t^*}^{(0)} - \mathbf{x}_{t^*}^{\prime(0)}\right\|$ be the norm of the difference of the input embeddings at this position. For the rest of this question, we assume $\|\cdot\|$ to be L2-norm unless otherwise stated.

a) **(1 pt) Simplifying the attention mechanism.**

To simplify notation, we will absorb $\mathbf{W}_V^{(\ell)}$ into $\mathbf{f}$. To do so, show that, given our definition of attention mechanism, there exists a function $\mathbf{f}'$ such that

$$\mathbf{x}_t^{(\ell+1)} = \mathbf{f}'\left(\sum_{t'=1}^{T} s_{t,t'}^{(\ell)} \mathbf{x}_{t'}^{(\ell)}, \mathbf{x}_t^{(\ell)}\right) = \mathbf{f}\left(\sum_{t'=1}^{T} s_{t,t'}^{(\ell)} \mathbf{v}_{t'}^{(\ell)}, \mathbf{x}_t^{(\ell)}\right). \tag{44}$$

Note: Your function $\mathbf{f}'$ should be a simple modification of $\mathbf{f}$.

b) **(3 pt) Bounding $\left\|\mathbf{x}_t^{(\ell)}\right\|$.**

Let $L_\mathbf{f}$ be the Lipschitz constant of the MLP $\mathbf{f}\colon \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^D$ with a pre- and post-residual connections and a non-linear activation function. Assuming for simplicity that $\mathbf{f}(\mathbf{0}, \mathbf{0}) = \mathbf{0}$,[5] show that

$$\left\|\mathbf{x}_t^{(\ell)}\right\| \leq (2L_\mathbf{f})^\ell \left\|\mathbf{X}^{(0)}\right\|_{2,\infty} \tag{45}$$

[5]An analogous result can easily be shown with the assumption removed.

c) **(2 pt) Bounding Attention Logits.**
Show that the attention logits $\mathbf{l} = \mathbf{Kq}$ (i.e, the infinity norm of the pre-softmax attention scores, $\|\mathbf{l}\|_\infty$) are bounded by $A = F^2 L_{\mathtt{att}}$, where $F \geq \max\left(1, (2L_{\mathbf{f}})^L\right)\left\|\mathbf{X}^{(0)}\right\|_{2,\infty}$ and $L_{\mathtt{att}}$ is a constant that depends on the model parameters (matrices).

d) **(2 pt) Bounding Softmax Values**
Let $\mathbf{l} \in \mathbb{R}^T$ be the logits of the attention mechanism such that $\|\mathbf{l}\| \leq A$. Show that the softmax values $\mathbf{s} = \mathrm{softmax}(\mathbf{l})$ can be bounded as

$$\frac{\exp(-2A)}{T} \leq s_t \leq \frac{\exp(2A)}{T}. \tag{46}$$

The proof of Lemma 1 hinges on bounding the difference in the intermediary layer ℓ activations,

$$\mathbf{z}_t^{(\ell)} = \sum_{t'=1}^{T} s_{t,t'}^{(\ell)} \mathbf{x}_{t'}^{(\ell)} \tag{47}$$

where $s_{t,t'}^{(\ell)}$ is the attention value at layer ℓ between input positions t and t'.

In the following, any quantity with a prime (e.g., $\mathbf{z}_t'^{(\ell)}$, $\mathbf{x}_{t'}'^{(\ell)}$) denotes the (intermediary) representation computed for the one-bit flipped string. For instance, $\boldsymbol{y}$ might be "`1001011`" while a possible $\boldsymbol{y}'$ might be "`1001010`" differs only in last bit. Then, $\mathbf{z}_t^{(\ell)}$ refer to the activations computed for $\boldsymbol{y}$ and $\mathbf{z}_t'^{(\ell)}$ for activations computed for $\boldsymbol{y}'$.

$$\Delta_t^{(\ell)} \overset{\text{def}}{=} \left\|\mathbf{z}_t^{(\ell)} - \mathbf{z}_t'^{(\ell)}\right\|. \tag{48}$$

where $s_{t,t'}^{(\ell)}$ is the attention value at layer ℓ between input position t and t'.

We will prove Lemma 1 by induction on the layer ℓ. Before we do that, however, we will set up a few intermediate results that we will use in the induction proof.

e) **(4 pt) Bounding $\Delta_t^{(\ell)}$.** Show that $\Delta_t^{(\ell)}$ can be bounded as

$$\Delta_t^{(\ell)} \leq \underbrace{\sum_{t'=1}^{T} \left|s_{t,t'}^{(\ell)} - s_{t,t'}'^{(\ell)}\right| \left\|\mathbf{x}_{t'}^{(\ell)}\right\|}_{\text{Difficult Part}} + \underbrace{\sum_{t'=1}^{T} s_{t,t'}'^{(\ell)} \left\|\mathbf{x}_{t'}^{(\ell)} - \mathbf{x}_{t'}'^{(\ell)}\right\|}_{\text{Easy Part}}. \tag{49}$$

f) **(1 pt) Bounding The Difficult Part, Initial Step.** Let us first analyze the Difficult Part:

$$\sum_{t'=1}^{T} \left|s_{t,t'}^{(\ell)} - s_{t,t'}'^{(\ell)}\right| \left\|\mathbf{x}_{t'}^{(\ell)}\right\| \tag{50}$$

We will do this across multiple subquestions. First, show that the Difficult Part can be bounded as

$$\sum_{t'=1}^{T} \left|s_{t,t'}^{(\ell)} - s_{t,t'}'^{(\ell)}\right| \left\|\mathbf{x}_{t'}^{(\ell)}\right\| \leq \left(\sum_{t'=1}^{T} \left|s_{t,t'}^{(\ell)} - s_{t,t'}'^{(\ell)}\right|\right) \left\|\mathbf{X}^{(\ell)}\right\|_{2,\infty} \tag{51}$$

g) **(3 pt) Convergence.** Now, consider a twice differentiable function $f\colon \mathbb{R} \to \mathbb{R}$. Show that for bounded sequences $(a_T)_{T\in\mathbb{N}}$ and $(b_T)_{T\in\mathbb{N}}$, if $|a_T - b_T| = \mathcal{O}\left(\frac{1}{T}\right)$, then $|f(a_T) - f(b_T)| = \mathcal{O}\left(\frac{1}{T}\right)$.

Hint: Use the mean value theorem.

Use the above to show that for any bounded sequences $(a_T)_{t\in\mathbb{N}}$ and $(b_T)_{T\in\mathbb{N}}$ such that $|a_T - b_T| = \mathcal{O}\left(\frac{1}{T}\right)$, then $|\exp(a_T) - \exp(b_T)| = \mathcal{O}\left(\frac{1}{T}\right)$.

h) **(5 pt, *bonus*) Softmax.**
Consider two bounded sequences $(\mathbf{a}_T)_{T\in\mathbb{N}}$ and $(\mathbf{b}_T)_{T\in\mathbb{N}}$ with dimensionality that grows with T (just as the attention logits in a transformer): $\mathbf{a}_T, \mathbf{b}_T \in \mathbb{R}^T$. Suppose these sequences satisfy $|(\mathbf{a}_T)_t - (\mathbf{b}_T)_t| = \mathcal{O}\left(\frac{1}{T}\right)$ for $t \in [T]$. Show that under these conditions, it holds that

$$|\text{softmax}(\mathbf{a}_T)_t - \text{softmax}(\mathbf{b}_T)_t| = \mathcal{O}\left(\frac{1}{T^2}\right) \tag{52}$$

for $t \in [T]$.

i) **(2 pt) Bounding The Difficult Part, Final Step.**
Conclude that, assuming that $\left\|\mathbf{x}_t^{(\ell)} - \mathbf{x}_t'^{(\ell)}\right\|_\infty = \mathcal{O}\left(\frac{1}{T}\right)$, we have that

$$\sum_{t'=1}^{T} \left|s_{t,t'}^{(\ell)} - s_{t,t'}'^{(\ell)}\right| \left\|\mathbf{x}_{t'}^{(\ell)}\right\| = \mathcal{O}\left(\frac{1}{T}\right) \tag{53}$$

j) **(6 pt) Combining All Results.**
Let us return to the bound of $\Delta_t^{(\ell)}$.

$$\begin{aligned}\Delta_t^{(\ell)} &\leq \sum_{t'=1}^{T} \left|s_{t,t'}^{(\ell)} - s_{t,t'}'^{(\ell)}\right| \left\|\mathbf{x}_{t'}^{(\ell)}\right\| + \sum_{t'=1}^{T} s_{t,t'}'^{(\ell)} \left\|\mathbf{x}_{t'}^{(\ell)} - \mathbf{x}_{t'}'^{(\ell)}\right\| &\text{(54a)}\\ &= \sum_{t'=1}^{T} \left|s_{t,t'}^{(\ell)} - s_{t,t'}'^{(\ell)}\right| \left\|\mathbf{x}_{t'}^{(\ell)}\right\| + \sum_{\substack{t'=1\\ t'\neq t^*}}^{T} s_{t,t'}'^{(\ell)} \left\|\mathbf{x}_{t'}^{(\ell)} - \mathbf{x}_{t'}'^{(\ell)}\right\| + s_{t,t^*}'^{(\ell)} \left\|\mathbf{x}_{t^*}^{(\ell)} - \mathbf{x}_{t^*}'^{(\ell)}\right\| &\text{(54b)}\end{aligned}$$

Using the results from the previous subquestions and by induction on ℓ, show that

$$\Delta_t^{(\ell)} = \begin{cases} \mathcal{O}(1) & \textbf{if } t = t^*, \\ \mathcal{O}\left(\frac{1}{T}\right) & \textbf{otherwise} \end{cases} \tag{55}$$

and

$$\left\|\mathbf{x}_t^{(\ell)} - \mathbf{x}_t^{(\ell)\prime}\right\| = \begin{cases} \mathcal{O}(1) & \textbf{if } t = t^*, \\ \mathcal{O}\left(\frac{1}{T}\right) & \textbf{otherwise} \end{cases}. \tag{56}$$

Hint: Your inductive hypothesis should bound $\left\|\mathbf{x}_t^{(\ell)} - \mathbf{x}_t^{(\ell)\prime}\right\|$.
What does that imply for the norm $\|\boldsymbol{h}(\boldsymbol{y}) - \boldsymbol{h}(\boldsymbol{y}')\|$?
Hint: By assumption, we have $t^* < T$ where t^* indicates the location where $\boldsymbol{y}$ and $\boldsymbol{y}'$ differ for single bit.

k) **(3 pt) Implications for `Odd-1`.**
Let us now explore what the results of the previous sub-questions mean for the transformer's ability to recognize `Odd-1`. Concretely, reason about the following points:

(i) Let $\boldsymbol{y} \in \{0,1\}^*$ be a string in `Odd-1`: $\boldsymbol{y} \in$ `Odd-1`. What does that mean for the string $\boldsymbol{y}'$ that differs from $\boldsymbol{y}$ in a single bit?

(ii) Suppose that a transformer assigns $\boldsymbol{y}$ the probability $\hat{p} = \sigma\left(\boldsymbol{h}(\boldsymbol{y})^\top \mathbf{w}\right) > 0.5$ of being in `Odd-1` (i.e., it classfies it correctly). What do the results derived in this question imply about the probability $\hat{p'} = \sigma\left(\boldsymbol{h}(\boldsymbol{y}')^\top \mathbf{w}\right)$ of $\boldsymbol{y}'$ being in `Odd-1`?

(iii) In particular, what happens as $|\boldsymbol{y}| \to \infty$?

l) **(2 pt) A Discussion.**

(i) How would our results and derivations change if we added layer normalization?

(ii) What does the asymptotic nature of the results imply? In particular, what does it tell us about the correct classification of strings up to a specific fixed length, $|\boldsymbol{y}| \leq T^*$?