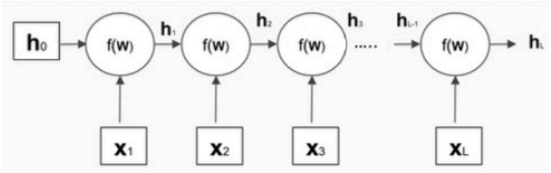
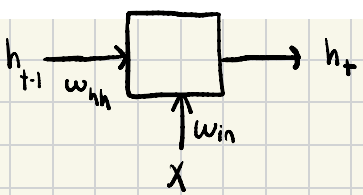
Question 1 (20 Points) Consider the following architecture of a simple Recurrent Neural Network (RNN):



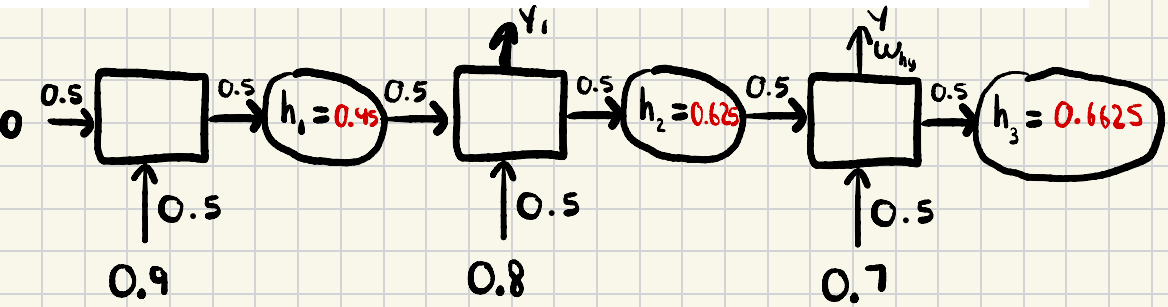
In this network, all the inputs, hidden states, and weights are assumed to be 1D. This network is to be used for a many-to-one regression task. The network produces an output y at the last time step L, which is given as $y = f(w_{hy}h_L)$. For the following questions, assume that no nonlinearities are used (i.e. no activation functions).

(a) **(3 Points)** Write a general expression for h_t in terms of w_{in}, w_{hh}, x_t, and h_{t-1}, where t is the timestep.



$$h_t = w_{in} X_t + w_{hh} h_{t-1}$$

(b) **(3 Points)** Given the sequential input $\mathbf{x} = [0.9, 0.8, 0.7]$, the initial hidden state $h_0 = 0$, and all weights initialized to 0.5, compute h_1, h_2, h_3, and y.



$h_1 = 0.5(0.9) + 0.5(0)$

$= 0.45$

$h_2 = 0.5(0.8) + 0.5(0.45)$

$= 0.625$

$h_3 = 0.5(0.7) + 0.5(0.625)$

$= 0.6625$

$y = 0.5(0.6625)$

$= 0.33125$

(c) **(10 Points)** You want to perform *Backpropagation Through Time (BPTT)*, which is used to update the weights of RNN networks. Given a target output y_r, Assume that you are using the function $l = \frac{1}{2}(y - y_r)^2$ to calculate the loss. The goal is to compute $\frac{\delta l}{\delta w_{in}}$, $\frac{\delta l}{\delta w_{hh}}$, and $\frac{\delta l}{\delta w_{hy}}$, and use them in the typical update rule given as $w_i \leftarrow w_i - \eta \frac{\delta l}{\delta w_i}$. Find the expressions for $\frac{\delta l}{\delta w_{in}}$, $\frac{\delta l}{\delta w_{hh}}$, and $\frac{\delta l}{\delta w_{hy}}$. Your expressions can include only the following: h_0, w_{in}, w_{hh}, w_{hy}, and x_i for $i \in [1, 2, 3]$. Show your work. (hint: for $\frac{\delta l}{\delta w_{in}}$ and $\frac{\delta l}{\delta w_{hh}}$, you need to consider the sum across all timesteps)

$$h_3 = w_{in} x_3 + w_{hh} h_2 \qquad l = \frac{1}{2}(w_{hy} h_3 - y_r)^2$$

$$h_3 = w_{in} x_3 + w_{hh}(w_{in} x_2 + w_{hh} h_1)$$

$$h_3 = w_{in} x_3 + w_{hh}(w_{in} x_2 + w_{hh}(w_{in} x_1 + w_{hh} h_0))$$

$$w_{in} x_3 + w_{hh}(w_{in} x_2 + w_{hh} w_{in} x_1 + w_{hh}^2 h_0)$$

$$h_3 = w_{in} x_3 + w_{hh} w_{in} x_2 + w_{hh}^2 w_{in} x_1 + w_{hh}^3 h_0$$

$$y = w_{hy} h_3 = w_{hy} w_{in} x_3 + w_{hy} w_{hh} w_{in} x_2 + w_{hy} w_{hh}^2 w_{in} x_1 + w_{hy} w_{hh}^3 h_0$$

$$l = \frac{1}{2}(w_{hy} w_{in} x_3 + w_{hy} w_{hh} w_{in} x_2 + w_{hy} w_{hh}^2 w_{in} x_1 + w_{hy} w_{hh}^3 h_0 - y_r)^2$$

$$\frac{\partial L}{\partial h_{hy}} = (w_{hy} h_3 - y_r) h_3$$

$$\frac{\partial L}{\partial w_{in}} = (w_{hy} h_3 - y_r)(w_{hy} x_3 + w_{hy} w_{hh} x_2 + w_{hy} w_{hh}^2 x_1)$$

$$\frac{\partial L}{\partial h_{hh}} = (w_{hy} h_3 - y_r)(w_{hy} w_{in} x_2 + w_{hy} 2 w_{hh} w_{in} x_1 + w_{hy} 3 w_{hh}^2 h_0)$$

(d) **(4 Points)** Assume that the target for the previously given data sequence is $y_r = 0.8$, with a learning rate of $\eta = 0.1$. Calculate the updated value of each weight.

$w_{in} = w_{in} - \eta \frac{\partial L}{\partial w_{in}}$

$= 0.5 - 0.1(-0.3105)$

$w_{in} = 0.53105$

$w_{hh} = 0.5 - 0.1(-0.1992)$

$= 0.51992$

$w_{hy} = 0.5 - 0.1(-0.3105)$

$= 0.53105$

$w_{hy} = 0.5$

$h_3 = 0.6625$

$X_1 = 0.9$

X_2 0.8

X_3 0.7

$\frac{\partial L}{\partial w_{in}} = -0.3105$

$\frac{\partial L}{\partial w_{hh}} = -0.1992$

$\frac{\partial L}{\partial w_{hy}} = -0.3105$

Question 2 **(10 Points)** Consider a Support Vector Machine (SVM) problem in 1D settings (with one weight and one bias). For the following questions, assume we have the dataset $D = \{(x_1, t_1), (x_2, t_2), (x_3, t_3)\} = \{(3, 1), (5, -1), (2, 1)\}$.

(a) **(5 Points)** Define the optimization problem (objective function and constraints) of SVM based on margin maximization.

$$w^*, b^* = \text{argmin } \frac{1}{2} w^2 \quad \text{S.T.} \begin{cases} 3w + b \geq 1 \\ -5w - b \geq 1 \\ 2w + b \geq 1 \end{cases}$$

(b) **(5 Points)** Solve the optimization problem using the graphical method. Explain your work.

$3w + b \geq 1$

$-5w - b \geq 1$

$2w + b \geq 1$

$b \geq 1 - 3w$

$-5w - 1 \geq b$

$b \geq 1 - 2w$

Looking at the graph we can clearly see that the minimum point where all three regions are shaded (where all 3 constraints are restricted) is (-1,4).

W = -1

B = 4

It is also possible to get to this solution with a system of linear equitation.

Question 3 (15 Points) Suppose you are training a RNN to predict the next word in a sentence based on the previous words. Given the sentence: "The cat sat on the," you want the model to predict the next word, which should be "mat". The vocabulary consists of the following 8 words: {'The', 'cat', 'sat', 'on', 'the', 'mat', 'is', 'dog'}. Each word should be represented by a one-hot encoded vector based on the size of the vocabulary. The RNN has a hidden state size of 4 and processes the sequence one word at a time. The RNN cell computes the hidden state at each time step t as follows:

$$h_t = \tanh(W_{hx}x_t + W_{hh}h_{t-1} + b_h)$$

The output is computed as:

$$y_t = \text{softmax}(W_{hy}h_t + b_y)$$

The initial hidden state h_0 is a zero vector.

You are given the following randomly initialized weights:

$$W_{hx} = \begin{bmatrix} 0.2 & -0.4 & 0.1 & 0.3 & 0.5 & 0.2 & -0.1 & 0.4 \\ 0.5 & 0.3 & -0.2 & 0.1 & -0.3 & 0.4 & 0.2 & 0.1 \\ 0.1 & 0.4 & 0.5 & 0.2 & 0.1 & -0.5 & 0.3 & 0.6 \\ 0.6 & -0.1 & -0.3 & 0.4 & 0.2 & 0.1 & 0.5 & -0.2 \end{bmatrix}$$

$$W_{hh} = \begin{bmatrix} 0.4 & -0.2 & 0.1 & 0.3 \\ 0.2 & 0.5 & -0.4 & 0.1 \\ -0.3 & 0.3 & 0.2 & 0.2 \\ 0.1 & -0.5 & 0.6 & 0.4 \end{bmatrix}$$

$$W_{hy} = \begin{bmatrix} 0.3 & 0.1 & -0.2 & 0.4 \\ 0.5 & -0.1 & 0.3 & -0.4 \\ 0.2 & 0.6 & -0.3 & 0.1 \\ 0.4 & -0.3 & 0.1 & 0.2 \\ 0.1 & 0.4 & 0.2 & -0.5 \\ 0.2 & 0.3 & -0.1 & 0.6 \\ -0.4 & 0.2 & 0.5 & 0.1 \\ 0.3 & -0.2 & 0.1 & 0.4 \end{bmatrix}$$

$$b_h = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \\ 0.4 \end{bmatrix}, \quad b_y = \begin{bmatrix} 0.05 \\ 0.1 \\ -0.05 \\ 0.15 \\ -0.1 \\ 0.05 \\ 0.2 \\ -0.15 \end{bmatrix}$$

(a) **(3 Points)** Using one-hot encoding (as per the given order of the words in the dictionary), encode each of the words in the input and output.

(a) **(3 Points)** Using one-hot encoding (as per the given order of the words in the dictionary), encode each of the words in the input and output.

The = [1 0 0 0 0 0 0 0] X_1^T

cat = [0 1 0 0 0 0 0 0] X_2^T

sat = [0 0 1 0 0 0 0 0] X_3^T

on = [0 0 0 1 0 0 0 0] X_4^T

the = [0 0 0 0 1 0 0 0] X_5^T

mat = [0 0 0 0 0 1 0 0] $+$

is = [0 0 0 0 0 0 1 0]

dog = [0 0 0 0 0 0 0 1]

(b) **(5 Points)** Compute the hidden state value h_t after processing each word in the input sequence.

$$h_1 = \tanh(W_{hx} x_1 + W_{hh} h_0 + b_h)$$

(W_{hx}: 4x8, x_1: 8x1, W_{hh}: 4x4, h_0: 4x1)

$$\tanh\left(\begin{bmatrix}0.2\\0.5\\0.1\\0.6\end{bmatrix} + \begin{bmatrix}0\\0\\0\\0\end{bmatrix} + \begin{bmatrix}0.1\\0.2\\0.3\\0.4\end{bmatrix}\right) \Rightarrow \tanh\begin{pmatrix}0.3\\0.7\\0.4\\1\end{pmatrix} = \begin{bmatrix}0.2913\\0.6044\\0.3799\\0.7616\end{bmatrix}$$

$$h_2 = \tanh(W_{hx} x_2 + W_{hh} h_1 + b_h)$$

$$\tanh\left(\begin{bmatrix}-0.4\\0.3\\0.4\\-0.1\end{bmatrix} + \begin{bmatrix}0.2621\\0.2846\\0.3223\\0.2595\end{bmatrix} + \begin{bmatrix}0.1\\0.2\\0.3\\0.4\end{bmatrix}\right) \Rightarrow \tanh\begin{pmatrix}-0.0379\\0.7846\\1.0223\\0.5595\end{pmatrix} = \begin{bmatrix}-0.03788\\0.6553\\0.7708\\0.5076\end{bmatrix}$$

$$h_3 = \tanh(W_{hx} x_3 + W_{hh} h_2 + b_h)$$

$$\tanh\left(\begin{bmatrix}0.1\\-0.2\\0.5\\-0.3\end{bmatrix} + \begin{bmatrix}0.0831\\0.0625\\0.4636\\0.3341\end{bmatrix} + \begin{bmatrix}0.1\\0.2\\0.3\\0.4\end{bmatrix}\right) \Rightarrow \tanh\begin{pmatrix}0.2831\\0.0625\\1.2636\\0.4341\end{pmatrix} = \begin{bmatrix}0.27577\\0.0624\\0.85205\\0.4087\end{bmatrix}$$

$$h_4 = \tanh(W_{hx} x_4 + W_{hh} h_3 + b_h)$$

$$\tanh\left(\begin{bmatrix}0.3\\0.1\\0.2\\0.4\end{bmatrix} + \begin{bmatrix}0.3056\\-0.2136\\0.1881\\0.6711\end{bmatrix} + \begin{bmatrix}0.1\\0.2\\0.3\\0.4\end{bmatrix}\right) \Rightarrow \tanh\begin{pmatrix}0.7056\\0.0864\\0.6881\\1.4711\end{pmatrix} = \begin{bmatrix}0.6079\\0.0862\\0.5968\\0.8998\end{bmatrix}$$

$$h_s = \tanh(W_{hx} X_s + W_{hh} h_4 + b_h)$$

$$\tanh\left(\begin{bmatrix}0.5\\-0.3\\0.1\\0.2\end{bmatrix} + \begin{bmatrix}0.5555\\0.0159\\0.1428\\0.7357\end{bmatrix} + \begin{bmatrix}0.1\\0.2\\0.3\\0.4\end{bmatrix}\right) \Rightarrow \tanh\begin{pmatrix}1.1555\\-0.0841\\0.5428\\1.3357\end{pmatrix} = \begin{bmatrix}0.8196\\-0.0839\\0.4951\\0.8706\end{bmatrix}$$

(c) **5 Points** Compute the final output after processing all the inputs.

$$y_t = \underset{8\times1}{\text{Softmax}}\left(\underset{8\times4}{W_{hy}}\ \underset{4\times1}{h_s} + \underset{8\times1}{b_y}\right)$$

$$= \text{Softmax}\left(\begin{bmatrix}0.48671\\0.21848\\0.05211\\0.57664\\-0.28788\\0.6116\\-0.01001\\0.66041\end{bmatrix} + \begin{bmatrix}0.05\\0.1\\-0.05\\0.15\\-0.1\\0.05\\0.2\\-0.15\end{bmatrix}\right) \Rightarrow \text{Softmax}\begin{pmatrix}0.53671\\0.31848\\0.00211\\0.72664\\-0.38788\\0.6616\\0.18999\\0.51041\end{pmatrix} = \begin{bmatrix}0.147\\0.118\\0.086\\0.178\\0.058\\0.166\\0.104\\0.143\end{bmatrix}$$

(d) The predicted word is 'ON'.

Based on the target, it is not the output we wanted, therefore it suggest that the model wasn't good enough. Couple solutions are available to fix this issue, either train the model more or Ds use ddddddddsds use a different kind of input representation, word embedding instead of one hot encoding.

Question 4 (15 Points) Consider a binary classification problem where the input data is linearly separable. Your task is to find the optimal hyperplane that separates the two classes with a maximum margin. This means the data can be perfectly divided into two classes without any misclassifications or margin violations.

The primal optimization problem for a hard-margin SVM is:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

subject to:

$$t_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, N$$

where:

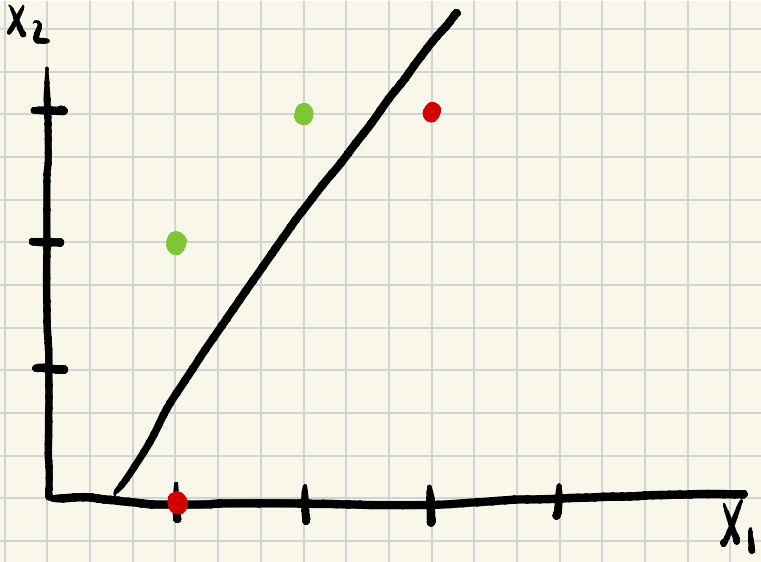
- $\mathbf{x}_i$ is the input vector for the i-th training sample.
- $t_i \in \{-1, 1\}$ is the label for the i-th training sample.
- $\mathbf{w}$ is the weight vector that defines the separating hyperplane.
- b is the bias term.

Consider the following linearly separable training data:

$\mathbf{x}_i$	t_i
$(1,2)$	1
$(2,3)$	1
$(3,3)$	-1
$(1,0)$	-1

(a) **(7 Points)** Write down the explicit form of the primal SVM optimization problem in terms of the weight vector $\mathbf{w}$ and bias term b, for the given dataset. Use scalar variables in your answer (do not use matrix notation) or combine the constraints into matrix form. Instead, write out each constraint explicitly.

(b) **(8 Points)** Derive the dual form of the optimization problem by introducing Lagrange multipliers with KKT constraints. Clearly list the objective function and the new constraints. Show your work.



(a) **(7 Points)** Write down the explicit form of the primal SVM optimization problem in terms of the weight vector $\mathbf{w}$ and bias term b, for the given dataset. Use scalar variables in your answer (do not use matrix notation) or combine the constraints into matrix form. Instead, write out each constraint explicitly.

$\mathbf{x}_i$	t_i
$(1,2)$	1
$(2,3)$	1
$(3,3)$	-1
$(1,0)$	-1

$t_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$

1. $1(w_1 + 2w_2 + b) \geq 1$

   $w_1 + 2w_2 + b \geq 1$

2. $1(2w_1 + 3w_2 + b) \geq 1$

   $2w_1 + 3w_2 + b \geq 1$

3. $-1(3w_1 + 3w_2 + b) \geq 1$

   $-3w_1 - 3w_2 - b \geq 1$

4. $-1(w_1 + b) \geq 1$

   $-w_1 - b \geq 1$

(b) **(8 Points)** Derive the dual form of the optimization problem by introducing Lagrange multipliers with KKT constraints. Clearly list the objective function and the new constraints. Show your work.

Lagrange: $\partial(w, b, a) = \frac{1}{2}\|w\|^2 - \sum a_i [t_i (w^T x_i + b) - 1]$

$\frac{\partial L}{\partial w} = w - a_i t_i x_i$

$\frac{\partial L}{\partial w} = w$

$\frac{\partial L}{\partial w} = a_i t_i x_i$

$\frac{\partial L}{\partial b} = a_i t_i$

$$L(w, b, a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} a_i t_i w^T x_i - \sum_{i=1}^{N} a_i t_i b + \sum_{i=1}^{N} a_i$$

$$L(w, a) = \sum_{i=1}^{N} a_i - \frac{1}{2} w^T w$$

$$L(a) = \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j t_i t_j \phi(x_i)^T \phi(x_j)$$

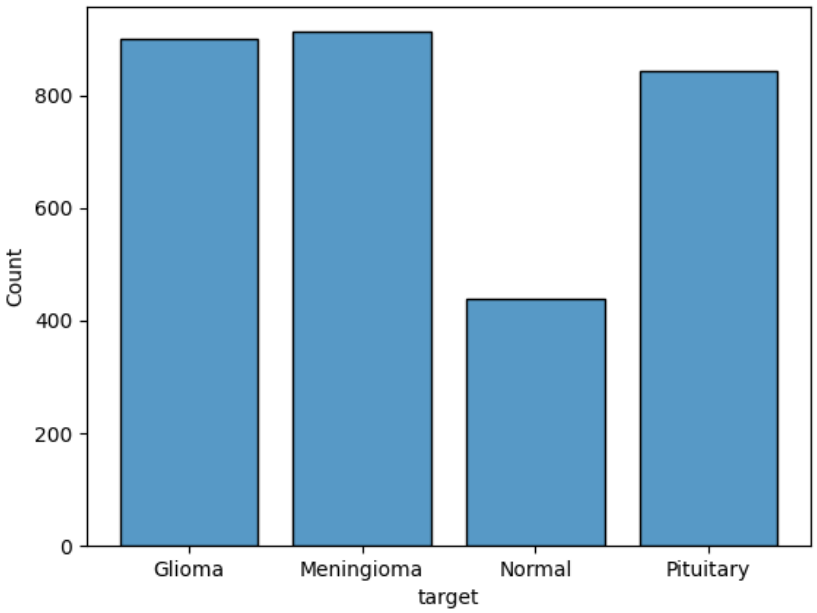
From Slides

$$\max \min \underbrace{L(w, b, a)}_{L(a)} \rightarrow \max_{a \geq 0} L(a)$$

$$a^* = \text{argmax} \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j t_i t_j (x_i)^T (x_j)$$

where $\sum a_i t_i = 0$, $a_i \geq 0$, $i = 1, \cdots, N$

IMPLEMENTATION 1



Meningioma	913
Glioma	901
Pituitary	844
Normal	438

We can see from the bar plot that the cases for Meningioma, Glioma and Pituitary cancers are evenly distributed in the dataset with a max difference of only 69 cases (Between Meningioma and Pituitary). On the other hand, there was only 438 cases where no cancer was detected.

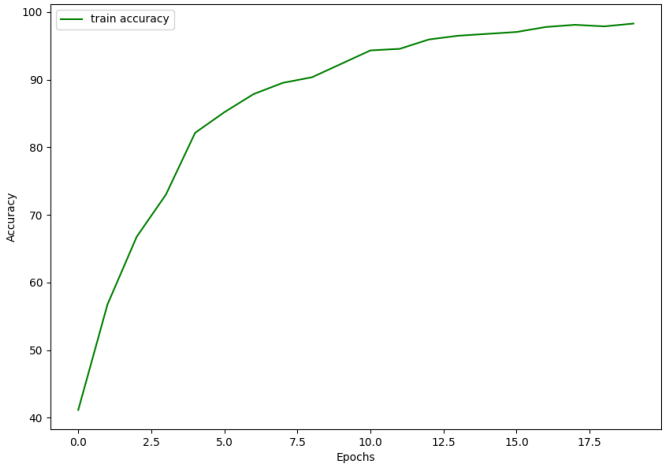
For this experience, I used Stochastic Gradient Descent.
After experimenting a little bit with the hyper parameters, I found that the best ones were the following:

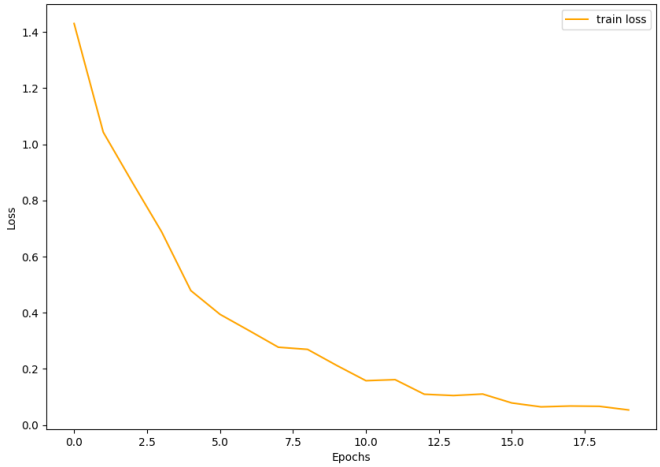
Learning rate = 0.001
Momentum = 0.9
Number of epochs = 20
Batch size = 4

I also used the Cross Entropy Loss as the objective function as it's the one used for multi-classification problem.

At first I was tasked with training a reset-18 model from scratch.
During training the loss went from 1.4301 to 0.0539 and the accuracy went from 41.16% to 98.29%.
This indicates that the training went as expected and the model is learning well.

Standard Model





Training Accuracy

Training loss

Loss over epochs

```
Epoch 1/20, Train Loss: 1.4301, Train Acc: 41.16%,
Epoch 2/20, Train Loss: 1.0430, Train Acc: 56.76%,
Epoch 3/20, Train Loss: 0.8637, Train Acc: 66.77%,
Epoch 4/20, Train Loss: 0.6883, Train Acc: 73.00%,
Epoch 5/20, Train Loss: 0.4791, Train Acc: 82.14%,
Epoch 6/20, Train Loss: 0.3945, Train Acc: 85.19%,
Epoch 7/20, Train Loss: 0.3364, Train Acc: 87.86%,
Epoch 8/20, Train Loss: 0.2773, Train Acc: 89.52%,
Epoch 9/20, Train Loss: 0.2697, Train Acc: 90.36%,
Epoch 10/20, Train Loss: 0.2125, Train Acc: 92.34%,
Epoch 11/20, Train Loss: 0.1582, Train Acc: 94.32%,
Epoch 12/20, Train Loss: 0.1619, Train Acc: 94.55%,
Epoch 13/20, Train Loss: 0.1100, Train Acc: 95.94%,
Epoch 14/20, Train Loss: 0.1053, Train Acc: 96.49%,
Epoch 15/20, Train Loss: 0.1106, Train Acc: 96.77%,
Epoch 16/20, Train Loss: 0.0789, Train Acc: 97.05%,
Epoch 17/20, Train Loss: 0.0650, Train Acc: 97.78%,
Epoch 18/20, Train Loss: 0.0680, Train Acc: 98.11%,
Epoch 19/20, Train Loss: 0.0669, Train Acc: 97.88%,
Epoch 20/20, Train Loss: 0.0539, Train Acc: 98.29%,
```

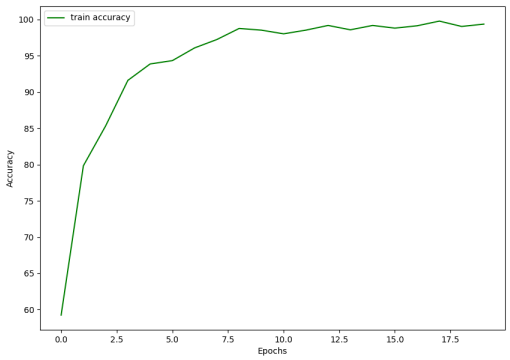
Classification report

```
Test Loss: 0.3730, Test Accuracy: 89.77%

Classification Report:

                precision    recall  f1-score   support

    glioma_tumor       0.97      0.80      0.88       271
meningioma_tumor       0.88      0.87      0.88       270
          normal       0.82      0.97      0.89       120
 pituitary_tumor       0.90      0.99      0.94       268

        accuracy                           0.90       929
       macro avg       0.89      0.91      0.90       929
    weighted avg       0.90      0.90      0.90       929
```

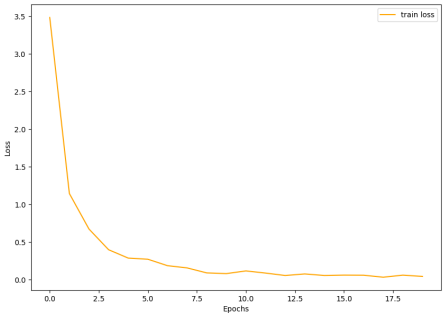
The test Loss is 0.3730 and the accuracy is 89.77% it might be a sign that the model is a little bit overfitting.

Pre-trained Model

Training Accuracy



Training Loss



Loss over epochs

```
Epoch 1/20, Train Loss: 3.4842, Train Acc: 59.25%,
Epoch 2/20, Train Loss: 1.1458, Train Acc: 79.83%,
Epoch 3/20, Train Loss: 0.6740, Train Acc: 85.33%,
Epoch 4/20, Train Loss: 0.3994, Train Acc: 91.60%,
Epoch 5/20, Train Loss: 0.2884, Train Acc: 93.86%,
Epoch 6/20, Train Loss: 0.2733, Train Acc: 94.32%,
Epoch 7/20, Train Loss: 0.1876, Train Acc: 96.08%,
Epoch 8/20, Train Loss: 0.1582, Train Acc: 97.23%,
Epoch 9/20, Train Loss: 0.0916, Train Acc: 98.75%,
Epoch 10/20, Train Loss: 0.0821, Train Acc: 98.52%,
Epoch 11/20, Train Loss: 0.1178, Train Acc: 98.02%,
Epoch 12/20, Train Loss: 0.0893, Train Acc: 98.52%,
Epoch 13/20, Train Loss: 0.0573, Train Acc: 99.17%,
Epoch 14/20, Train Loss: 0.0776, Train Acc: 98.57%,
Epoch 15/20, Train Loss: 0.0575, Train Acc: 99.17%,
Epoch 16/20, Train Loss: 0.0623, Train Acc: 98.80%,
Epoch 17/20, Train Loss: 0.0610, Train Acc: 99.12%,
Epoch 18/20, Train Loss: 0.0344, Train Acc: 99.77%,
Epoch 19/20, Train Loss: 0.0619, Train Acc: 99.03%,
Epoch 20/20, Train Loss: 0.0450, Train Acc: 99.35%,
```

Classification report

```
Test Loss: 0.2172, Test Accuracy: 94.62%

Classification Report:

                precision    recall  f1-score   support

    glioma_tumor       0.97      0.92      0.94       271
meningioma_tumor       0.90      0.96      0.93       270
          normal       0.97      0.90      0.94       120
 pituitary_tumor       0.97      0.99      0.98       268

        accuracy                           0.95       929
       macro avg       0.95      0.94      0.94       929
    weighted avg       0.95      0.95      0.95       929
```

As expected, we notice that the results a little bit better on the pre-trained model the accuracy starts off higher (we have the same seed so the data is exactly the same).

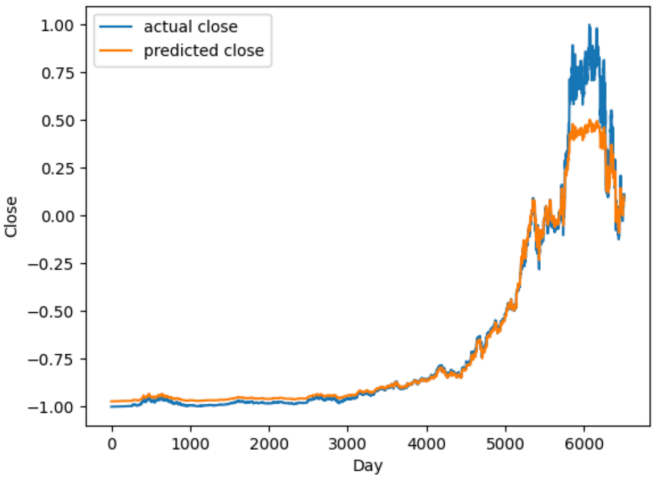
Implementation 2

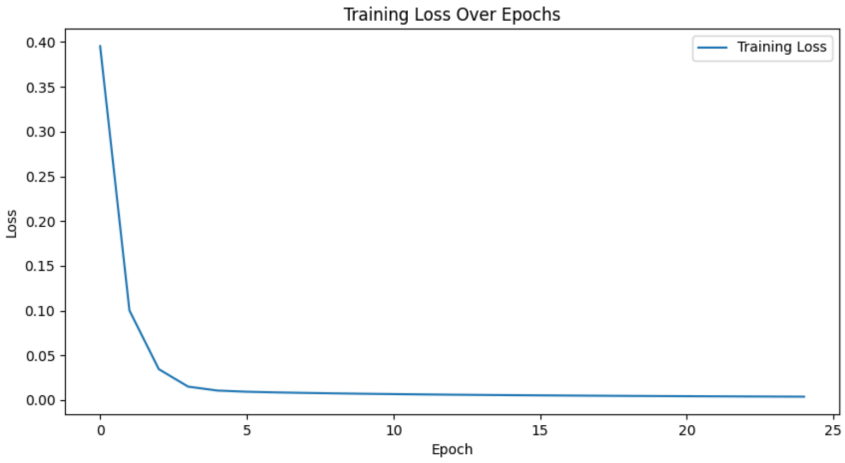
To perform this task, I created a function that takes a dataframe as an input and a number of steps. This function then creates a ‘window’ of data to be used as input.
In this task we wanted a to predict the 11th value using the previous 10 values. Therefore the new dataframe looks like this:

Date	Close	Close (t-1)	...	Close (t-9)	Close (t-10)
1997-05-30	0.075000	0.075260	...	0.086458	0.097917
1997-06-02	0.075521	0.075000	...	0.085417	0.086458
1997-06-03	0.073958	0.075521	...	0.081771	0.085417
1997-06-04	0.070833	0.073958	...	0.071354	0.081771
1997-06-05	0.077083	0.070833	...	0.069792	0.071354
...	...	...	...	...	...
2023-03-30	102.000000	100.250000	...	98.949997	100.040001
2023-03-31	103.290001	102.000000	...	97.709999	98.949997
2023-04-03	102.410004	103.290001	...	100.610001	97.709999
2023-04-04	103.949997	102.410004	...	98.699997	100.610001
2023-04-05	101.099998	103.949997	...	98.709999	98.699997

After that I scalled the values from -1 to 1 to keep them in ‘compact’ range to help the model learn better.
Each input sequence was of the shape (10,1)

I created the LSTM model with 4 hidden layers to not over complexity the training and because the dataset was pretty small.
I used Stochastic Gradient Descent as the optimizer with a learning rate of 0.001.
I used a batch size of 10 as the training dataset was small.
The model was trained over 25 epochs.





As we can see from the training curve, the model learns well over the epochs.
The graph on the left represent how our prediction (orange) fits the actual data (blue).
There is still place for improvement but the model does well enough !

IMPLEMENTATION 3

The model performs very well on the training set (Accuracy of 93%), it means that the model fit the training data well.
The model also maintained a high accuracy (92%), this indicates that the model isn't overfitting and is able to generalize well on unseen data.
The difference in precision between class 0 (0.67) and class 1 (0.94) means that the model is less effective to predict non cancerous case.

Overall the SVM model shows strong performance especially for detecting cancerous cases.

```
Classification report for TRAINING set
              precision    recall  f1-score   support

           0       0.73      0.73      0.73        30
           1       0.96      0.96      0.96       186

    accuracy                           0.93       216
   macro avg       0.85      0.85      0.85       216
weighted avg       0.93      0.93      0.93       216



Classification report for TEST set
              precision    recall  f1-score   support

           0       0.67      0.44      0.53         9
           1       0.94      0.98      0.96        84

    accuracy                           0.92        93
   macro avg       0.80      0.71      0.75        93
weighted avg       0.92      0.92      0.92        93
```