האוניברסיטה העברית בירושלים

בית הספר להנדסה ולמדעי המחשב ע"ש רחל וסלים בנין

סדנאות תכנות בשפת C ו-C++ קורס67315
C++ – תרגיל 1

תאריך ההגשה של התרגיל: יום רביעי, ה- 17ביוני, 2020– עד השעה 23:25

נושאי־התרגיל: היכרות עם השפה, מחלקות,.operators overloading references, const & const return types, אנא הקפידו לקרוא את כל התרגיל מתחילתו ועד סופו לפני שתגשו לממשו.

1 רקע

בתרגיל זה נכתוב תוכנה לזיהוי תוכן כתב יד, נקבל כקלט תמונה של ספרה $[0:9]$ ונחזיר כפלט איזו ספרה הופיעה. נעשה זאת ע״י שימוש במודל הנקרא רשת נוירונים. הרשת אותה נריץ מגיעה ל-96% זיהוי של ספרות.

CPP Program → 6

2 תיאור כללי

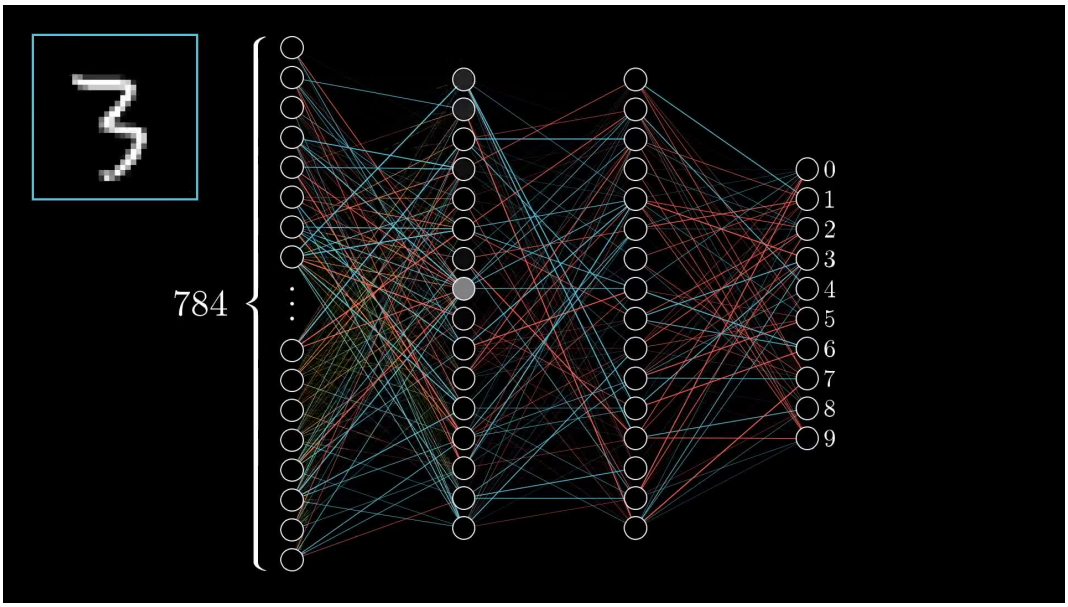
2.1 הקדמה

רשת נוירונים זהו מודל בלמידת מכונה אשר פותח בהשראת המח, הנוירון הביולוגי מקבל קלט כגירוי חשמלי, ואם הגירוי גדול מסף מסויים (סף האקטיבציה) הנוירון ״יורה״ כלומר פולט ומגרה נוירון אחר. המח מורכב ממספר רב של נוירונים המקושרים זה לזה ברשת קישורים מורכבת ויחד מסוגלים לבצע את הפעולות הנדרשות מהמח. רשת נוירונים מלאכותית (Artificial neural network) פועלת באופן דומה. אופי הבעיות שנפוץ לפתור באמצעות רשתות כאלו הינן בעיות של זיהוי ומיקום עצמים בתמונה, ״הבנת״ שפה, יצירת טקסט ועוד. מוצרים רבים בחיינו משתמשים ברשתות שכאלו החל מהעוזרים הקוליים(Amazon Alexa, Apple Siri) דרך השלמה אוטומטית לתוכן המייל ב Gmail, זיהוי מחלות בתמונות סריקה רפואית ועוד.

2.2 הסבר תיאורטי

למידת מכונה בכלל ורשתות נוירונים בפרט הינם נושאים רחבים ומורכבים ולכן הינם מחוץ לתוכן תרגיל זה וקורס זה. לצורך מימוש התרגיל אין צורך להבין מהו נוירון או כיצד מבצעת הרשת את פעולתה, למעט המושגים בסעיף 2.2.1 והמחלקות אשר עליכם לממש בסעיף 4. (לא נאמן רשת בתרגיל).

הסרטון הבא מסביר מהו נוירון, מהי רשת נוירונים, וכיצד ניתן לממשה ע״י שימוש באלגברה ליניארית:
https://youtu.be/aircAruvnKk



2.2.1 המושגים הדרושים לתרגיל:

נתאר רשת Fully connected:

- רשת בנויה משכבות
- הקלט של כל שכבה הוא וקטור
- הפלט של כל שכבה הוא וקטור (אחר)
- הפלט של כל שכבה הוא הקלט של השכבה הבאה
- קלט הרשת הינו וקטור המייצג את התמונה (במימוש שלנו באורך 784 - ראו סעיף 3.3.2)
- פלט הרשת הינו וקטור באורך 10 כך שהאינדקס בו הערך המקסימלי בוקטור מייצג את הספרה שהרשת זיהתה. לדוגמה
  -

$Value$	0	0.003	0.08	0	0	0	0	0.9	0.007	0.01
$Index$	0	1	2	3	4	5	6	7	8	9

  - הרשת מזהה כי הספרה בתמונה היא 7 בהסתברות 90%
  - במקרה של שיוויון בערכים יש להחזיר את האינדקס הנמוך מבין השניים

כל שכבה מורכבת מסט הפעולות הבא (בסדר בו הן מוצגות):

- **העתקה ליניארית** $T : \mathbb{R}^m \to \mathbb{R}^n$
  - אם $x \in \mathbb{R}^m$ הינו וקטור הקלט לשכבה אזי $T(x) = W \cdot x = y \in \mathbb{R}^n$
  - $W \in \mathbb{M}_{n,m}(\mathbb{R})$ הינה המטריצה המייצגת של T
  - איברי המטריצה נקראים משקולות (weights) השכבה
- **היסט** $b \in \mathbb{R}^n$
  - נחבר את ההיסט (bias) לווקטור שהתקבל מהפעלת ההעתקה הליניארית: y
  - $y + b = W \cdot x + b = z$
  - $z \in \mathbb{R}^n$
- **פונקציית אקטיבציה** $f : \mathbb{R}^n \to \mathbb{R}^n$
  - פונקציה הפועלת על תוצאת ההעתקה וחיבור ההיסט z ומחזירה את התוצאה הסופית של השכבה (וקטור).
  - $f(z) = f(W \cdot x + b) = r$ כאשר r המוצא הסופי של השכבה.
  - $r \in \mathbb{R}^n$
  - לכל שכבה פונקציית אקטיבציה אחת.
  - פונקציה זו איננה ליניארית.
  - **בתרגיל זה נממש 2 פונקציות אקטיבציה שונות:**
  - פונקציית אקטיבציה Relu:

$$\forall x \in \mathbb{R}\ Relu(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

    * כאשר הפונקציה פועלת על וקטור היא פועלת על כל קואורדינטה בנפרד.
  - פונקציית אקטיבציה Softmax:

$$\forall z \in \mathbb{R}^m \quad \sigma(z) = \frac{1}{\sum_{k=1}^{m} e^{z_k}} \begin{bmatrix} e^{z_1} \\ e^{z_2} \\ e^{z_3} \\ \vdots \\ e^{z_m} \end{bmatrix} \in \mathbb{R}^m$$

  - z_k - הקואורדינטה ה k בווקטור z
  - e^t - הפונקציה האקספוננציאלית exp(t)
    * ע"י שימוש ב-std::exp המיובאת מ-<cmath>
  - **פונקציה זו מנרמלת את התוצאה להסתברויות**

3 מימוש הרשת

שימו לב! אלא אם צויין מפורשות אחרת, טיפוס המידע איתו נעבוד לאורך התרגיל הינו float (32 ביט).

לדוג׳ במימוש מטריצה יש להניח כי איברי המטריצה הינם float.

כמו כן על מנת להמנע משגיאות נומריות[1] אנא ממשו את סדר הפעולות לפי ההגדרה המתמטית הרגילה לביצוע כפל מטריצות - ישנם מס׳ אלגוריתמים לביצוע כפל מטריצות - אנא השתמשו באלג׳ הנאיבי כפי שנלמד בליניארית 1. לדוג׳:

$$C = A \cdot B \iff c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

עבור A, B מטריצות בעלות כפל מוגדר ביניהן.

3.1 תיאור הרשת

הרשת מורכבת מ-4 שכבות:

שכבה	משקולות - Weights		Bias - היסט	אקטיבציה
1 (כניסה)	$(W_1 \in M_{128\times784})$	$T_1 : \mathbb{R}^{784} \to \mathbb{R}^{128}$	$b_1 \in \mathbb{R}^{128}$	$Relu$
2	$(W_2 \in M_{64\times128})$	$T_2 : \mathbb{R}^{128} \to \mathbb{R}^{64}$	$b_2 \in \mathbb{R}^{64}$	$Relu$
3	$(W_3 \in M_{20\times64})$	$T_3 : \mathbb{R}^{64} \to \mathbb{R}^{20}$	$b_3 \in \mathbb{R}^{20}$	$Relu$
4 (מוצא)	$(W_4 \in M_{10\times20})$	$T_4 : \mathbb{R}^{20} \to \mathbb{R}^{10}$	$b_4 \in \mathbb{R}^{10}$	$Softmax$

כלומר על מנת לממש את הרשת עלינו לשרשר את רצף הפעולות הבא:

$$\begin{aligned} r_1 &= Relu(W_1 \cdot x + b_1) \\ r_2 &= Relu(W_2 \cdot r_1 + b_2) \\ r_3 &= Relu(W_3 \cdot r_2 + b_3) \\ r_4 &= Softmax(W_4 \cdot r_3 + b_4) \end{aligned}$$

- x - וקטור הכניסה לרשת (סעיף 3.3.2)
- r_i - תוצאת השכבה ה-i, הינה הקלט לשכבה ה- $i+1$
- r_4 - וקטור תוצאת הרשת (כמוסבר בסעיף 2.2.1)

3.2 אינדקס יחיד לזכרון דו מימדי

מטריצה הינה אובייקט דו ממדי (אנו זקוקים ל-2 אינדקסים על מנת לזהות איבר באופן חח״ע) אך לשימושים שונים נוח יותר לגשת לכל איבר במטריצה באמצעות אינדקס יחיד. נבצע את המיפוי מזוג אינדקסים לאינדקס יחיד באופן הבא:

$$A(i,j) == A[i \cdot \text{rowSize} + j]$$

- A - מטריצה
- i - אינדקס השורה
- rowSize - אורך השורה (כלומר, מס׳ העמודות)
- $0 \le j <$ rowSize - אינדקס העמודה

לדוגמה עבור המטריצה $A \in M_{3\times4}$ (מט׳ בעלת 3 שורות ו 4 עמודות) מתקיים: $A(2,1) == A[2 \cdot 4 + 1] == A[9]$
וודאו כי הינכם מבינים מדוע מיפוי זה הינו חח״ע ועל.

[1] מאופן המימוש של float במעבד, פעולות האריתמטיקה אינן אסוציאטיביות, כלומר **לא בהכרח מתקיים** כי $(a+b)+c == a+(b+c)$

3.3 תיאור הקלט

3.3.1 קלט הפרמטרים (משקולות והיסטים)

התוכנה תקבל בשורת הפקודה את הנתיב לקבצי המשקולות וההיסטים כקבצים בינארים.

- רצף של בתים, כל 4 בתים רצופים הינם float 32 bit
- תטען אותם לפי סעיף 4.1 למחלקה Matrix באמצעות האופרטור $<<$
- לכל שכבה, קובץ המשקולות/ההיסטים מכיל מערך של float שניתן לקרוא אותו כפי שהוא ממערכת הקבצים.
- לדוג׳ עבור שכבה 1:
  - לשכבה $784 \times 128 = 100352$ משקולות (מטריצה מגודל 784×128)
  - אזי עלינו לקרוא מערך של float 100352
  - ובסה״כ גודל הקובץ הינו $100352 \times 4 = 401408[Bytes]$ (sizeof(float)=4)
  - המיפוי לכל אינדקס במערך לאינדקסים עבור מטריצה יעשה לפי סעיף 3.2

3.3.2 קלט התמונה

לאחר עליית התוכנה, היא תקבל כקלט נתיב לתמונה.

- התמונות ינתנו כקבצים בינארים (רצף של בתים המייצג מערך float - ראו סעיף קודם - 3.3.1)
- התמונות מקודדות כמטריצה של פיקסלים בגווני אפור (Grayscale) מסדר 28x28 וכל כניסה במטריצה מקבלת ערכים בין 0 ל-1.
- "image matrix" $\in M_{28,28}([0,1])$
- לנוחיותכם, בקבצי העזר הנתונים נמצא הסקריפט plot_img.py אשר מקבל כקלט נתיב לתמונה ומציג אותה בחלון חדש.
- התמונה תוכנס כקלט לרשת כמטריצה בעלת עמודה אחת ו-784 שורות $(28^2 = 784)$ (מיפוי האיברים בין המטריצות $M_{28,28}$ ל-$M_{784,1}$ יתבצע לפי סעיף 3.2)
- התמונות יטענו למטריצה באמצעות האופרטור <<

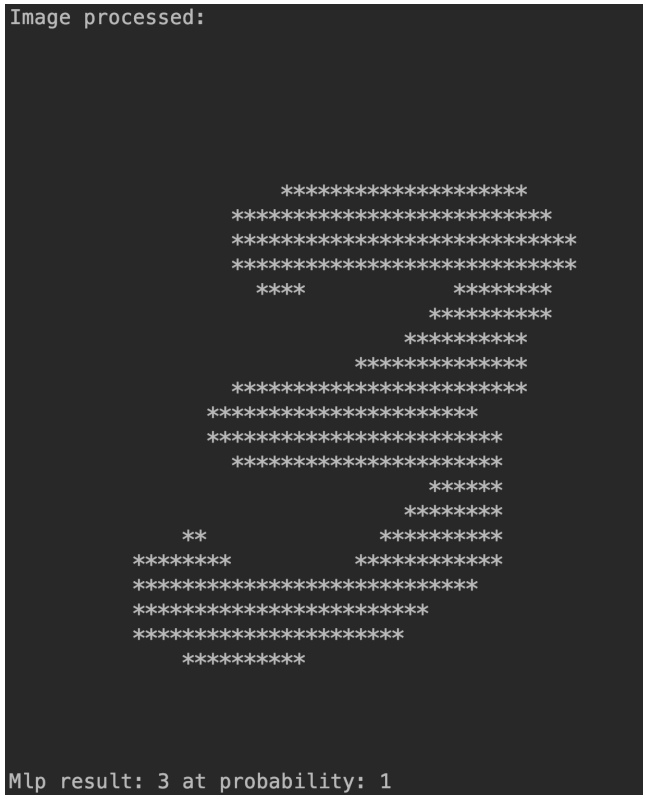
3.4 תיאור הפלט

התוכנה תפלוט (stdout)-את התמונה כ $mask$ ואת הספרה שהרשת טוענת שהיא זיהתה ובאיזו הסתברות. את הפלט יש לייצר לפי הפסאודו קוד הבא

* יש לשים לב שרק החלק של הלולאה זה הפורמט של האופרטור >> .
כל מה שמחוץ ללולאה מומש בקובץ ה main שקיבלתם.

```
print newline
    for i = 0 to (A.rows - 1):
        for j = 0 to (A.cols - 1):
            if a(i,j) <= 0.1:
                print "  " (double space)
            else:
                print "**"  (double asterisk)
        print newline
  print newline
  print "Mlp result: <res> at probability: <prob>"
  print newline
```

לדוגמה (מתוך הרצת פתרון בית ספר):



- כמצוין ברקע לתרגיל, הרשת אותה אנו ממשים מגיעה לדיוק של כ-96% בזיהוי, לכן יתכן בהחלט כי נטעה בזיהוי אך זהו מצב תקין מבחינת המימוש שלנו, במקרים שכאלו מומלץ להשוות לפתרון בית הספר ולוודא קבלת תוצאה דומה.
- במימוש באמצעות float יתכנו שגיאות נומריות, לכן גם אם ערך ההסתברות שקיבלתם איננו זהה בדיוק לזה של פתרון בית הספר, יתכן שהמצב תקין ואינכם נדרשים לפעולה נוספת. עם זאת, אם ערך ההסתברות באופן עקבי רחוק מזה של פתרון בית הספר - זוהי אינדיקציה לבעיה.

3.5 תיאור מהלך הריצה

נריץ את התוכנה עם המשקולות וההיסטים:

```
$ ./mlpnetwork w1 w2 w3 w4 b1 b2 b3 b4
```

- w_i - קבצי המשקולות לשכבות המתאימות (לפי 3.1)
- b_i - קבצי ההיסטים לשכבות המתאימות

כאשר התוכנה רצה היא ממתינה לקלט מהמשתמש. הקלט יהא נתיב לתמונה המכילה ספרה (לפי $3.3.2$). על התוכנה:

1. לקרוא את התמונה
2. להכניסה לרשת
3. כאשר התקבלה תוצאה, לפלוט את התמונה והערך שהרשת זיהתה ובאיזו הסתברות.
4. להמתין לקלט חדש
5. כאשר המשתמש הקיש q - התוכנה תצא עם קוד 0.

שימו לב: חלק זה ממומש עבורכם בקובץ main.cpp הניתן לכם עם קבצי התרגיל.

4 המחלקות למימוש

הינכם נדרשים לממש את המחלקות הבאות בלבד. אין להגיש מחלקות נוספות.

- בטבלאות הנ״ל רשומות כלל המתודות והאופרטורים אשר עליכם לממש.
- **חישבו היטב** מהם ערכי החזרה לכל מתודה/אופרטור, ואילו מתודות/אופרטורים בהכרח משנים את האובייקט הנוכחי ומהן המשמעויות הנגזרות מכך.
- אין להרחיב את ה API המפורט, כלומר אין להוסיף מתודות (public) למחלקות.

4.1 המחלקה Matrix:

נממש באמצעותה את המטריצות והווקטורים הדרושים לריצת התוכנית. [ווקטור הינו מטריצה בעלת עמודה אחת, כלומר Matrix$(n, 1)$].
נזכיר כי טיפוס הנתונים (עבור כניסות המטריצה) הינו float.

	Description	**Comments**
Constructor	Matrix(int rows, int cols)	Constructs Matrix rows $\times$ cols while rows & cols are positive numbers Inits all elements to 0
Default c'tor	Matrix()	Constructs 1$\times$1 Matrix Inits the single element to 0
Copy c'tor	returns a "copied" matrix	Constructs matrix from another Matrix m
Destructor	~Matrix()	
	Methods	
Getter	get rows	returns the amount of rows as int
Getter	get cols	returns the amount of cols as int
	vectorize - Transforms a matrix into a coloumn vector Supports function calling concatenation.	i.e.(1) Matrix m(5,4);... m.vectorize() m.getCols() == 1 m.getRows() == 20 i.e.(2) Matrix m(5,4), b(20, 1); then m.vectorize() + b should be a valid expression
PlaintPrint	Prints matrix elements, no return value.	Prints space after each element (incl. last element in the row) prints newline after each row (incl. last row)
	Operators	
=	Assignment	Matrix a, b; ... a = b;
*	Matrix multiplication	Matrix a, b; $\rightarrow$ a * b
*	Scalar multiplication on the right	Matrix m; float c; $\rightarrow$ m * c
*	Scalar multiplication on the left	Matrix m; float c; $\rightarrow$ c * m
+	Matrix addition	Matrix a, b; $\rightarrow$ a + b
+=	Matrix addition accumulation	Matrix a, b; $\rightarrow$ a += b
()	Parenthesis indexing	For i,j indices, Matrix m: m(i,j) will return the i,j element in the matrix
[]	Brackets indexing	For i index, Matrix m: m[i] will return the i'th element (section 3.2)
>>	Input stream	Fills matrix elements has to read input stream fully, otherwise, that's an error (dont trust the user to validate it). ifstream is; Matrix m(rows, cols); ... is >> m;
<<	Output stream	Pretty export of matrix as per section 3.4

פונקציות שימושיות למחלקה זו:

- istream.read: istream& read (char* s, streamsize n); -> Read block of data
  .Extracts n characters from the stream and stores them in the array pointed to by s

- istream.good Public member function inherited from ios
  similiar to eof that you already know, the good function:
  checks whether state of stream is good (public member function)

4.2 המחלקה Activation:[2]

במחלקה זו נגדיר את פעולת פונקציית האקטיבציה:

	Description	**Comments**
Constructor	Activation(ActivationType actType)	Accepts activation type (Relu/Softmax) and defines this instace's activation accordingly
	Methods	
Getter	get activation type	Returns this activation's type(Relu/Softmax)
	Operators	
()	Parenthesis	Applies activation function on input. (Does not change input) Matrix output = act(input)

4.3 המחלקה Dense:

המחלקה מייצגת שכבה (שורה בטבלה - 1.3) וניעזר בה ע״מ להגדיר ולהפעיל את פעולות השכבות השונות ברשת.

	Description	**Comments**
Constructor	Dense(w, bias, ActivationType)	Inits a new layer with given parameters note: c'tor accepts 2 matrices and activation type
	Methods	
Getter for weights	get weights	Returns the weights of this layer forbids modification
Getter for bias	get bias	Returns the bias of this layer forbids modification
Getter for activation	get activation object	Returns the activation function of this layer forbids modification
	Operators	
()	Parenthesis	Applies the layer on input and returns output matrix Layers operate as per section 3.1 Matrix output = layer(input)

[2] ה-enum ActivationType נמצא בקבצי התרגיל הנתונים. הפתרון ״היותר נכון״ למבנה האקטיבציות היה ירושה אך זו מחוץ ל-Scope תרגיל זה.

4.4 המחלקה MlpNetwork:

מחלקה זו תשמש אותנו לסדר את השכבות השונות למבנה רשת ותאפשר הכנסה של קלט לרשת וקבלת הפלט המתאים. מחלקה זו ממשת ספציפית את הרשת המתוארת במסמך זה (סעיף 3.1). כתרגיל עצמאי לבדיקת הבנה, חשבו מה היה נדרש לממש (מתודות ואופרטורים) במחלקה זו על מנת לתמוך ברשת עם מספר שכבות הניתן בזמן ריצה.[3]

	Description	**Comments**
Constructor	MlpNetwork(weights[], biases[])	Accepts 2 arrays, size 4 each. one for weights and one for biases. constructs the network described (sec. 3.1)
	Operators	
()	Parenthesis	Applies the entire network on input returns digit struct MlpNetwork m(...); digit output = m(img);

5 טיפול בשגיאות

טרם למדנו exceptions ולכן במקרה של שגיאה:

- נדפיס הודעת שגיאה אינפורמטיבית המתחילה בפתיח "Error:" ל-stderr
- הדפסת הודעת השגיאה תסתיים בירידת שורה ותבוצע כך:
  - std::cerr << errMsg << std::endl;
- נצא מהתוכנית עם קוד 1 (למעט אם נאמר מפורשות אחרת במסמך זה).
- במקרה שכזה, בתרגיל זה אינכם נדרשים לשחרר את הזכרון.

6 קימפול והרצה

בקבצי העזר לתרגיל הניתנים לכם, מצורף קובץ Makefile על מנת לקמפל את התוכנה. על התוכנה להתקמפל באמצעות הפקודה הבאה:

```
$ make mlpnetwork
```

נריץ את התוכנית כמפורט בסעיף 3.5

7 הקבצים להגשה

Matrix.h	Matrix.cpp
Activation.h	Activation.cpp
Dense.h	Dense.cpp
MlpNetwork.h	MlpNetwork.cpp
	Makefile

- עליכם ליצור קובץ tar הכולל את הקבצים לעיל. ניתן ליצור קובץ tar כדרוש על ידי הפקודה:

```
$ <...tar -cvf cpp_ex4.tar <files
```

[3] **digit struct מוגדר בקבצים הנתונים לכם**

8 הערות וסיכום

8.1 הנחיות כלליות

- קראו בקפידה את הוראות תרגיל זה ואת ההנחיות להגשת תרגילים שבאתר הקורס.
- זכרו שהחל מתרגיל זה עליכם לקמפל את התוכנית כנגד מהדר לשפת ++C בתקן שנקבע בקורס. כמו כן, זכרו שעליכם **לתעדף** פונקציות ותכונות של ++C על פני אלו של C. למשל, נעדיף להשתמש ב-new ו-delete על פני malloc ו-free, וכן נעדיף להשתמש ב-std::string מאשר ב-*char.
- **נזכיר:** כאמור בהנחיות הכלליות להגשת תרגילים - הקצאת זיכרון דינמית מחייבת את שחרור הזיכרון, למעט במקרים בהם ישנה שגיאה המחייבת סגירת התוכנית באופן מיידי עם קוד שגיאה (כלומר קוד יציאה 1). תוכלו להיעזר בתוכנה valgrind כדי לחפש דליפות זיכרון בתוכנית שכתבתם.
- אנא וודאו כי התרגיל שלכם עובר את ה-Pre-submission Script **ללא שגיאות או אזהרות**. קובץ ה-Pre-submission Script זמין בנתיב.

~labcc2/www/ex4/presubmit_ex4

- גרסת ++C לתרגיל: ++C17
- הקפידו לא להעתיק by value משתנים כבדים, אלא להעבירם (היכן שניתן) by reference.
- הקפידו מאוד על שימוש במילה השמורה const בהגדרות הפונקציות והפרמטרים שהן מקבלות. פונקציות שאינן משנות פרמטר מסויים – הוסיפו const לפני הגדרת הפרמטר. מתודות של מחלקה שאינן משנות את משתני המחלקה – הוסיפו const להגדרת המתודה.
- אין להשתמש ב - std::vector
- אין להשתמש בקונטיינרים דינאמים - Dynamic containers

8.2 פתרון בית הספר

פתרון בית הספר (גרסה מקומפלת הניתנת להרצה) זמינה לכם להרצה בנתיב הבא:

~labcc2/www/ex4/school_solution/mlpnetwork

- בכל מקרה של אי התאמה בין הנאמר במסמך זה לבין הפתרון, המסמך הוא זה שקובע.
- אם אתם חושבים שמצאתם אי התאמה שכזו, אנא הביאו אותה לידיעתנו ע״י שימוש בפורום ע״מ שנוכל לבדוק את הנושא.
- אנו נעשה את מירב המאמצים לפתור זאת מהר וביעילות ככל הניתן.

8.3 שאלות על התרגיל

במידה ומסמך זה איננו ברור לכם/נתקלתם בבעיות במימוש, אנו מעודדים אתכם לשאול שאלות בפורום.
הפנייה צריכה לכלול:

- כותרת מתומצתת לבעיה
- תיאור מילולי מפורט (וקריא)
- צילום מסך של הרצת פתרון בית הספר עם הבעיה/ צילום מסך של מסמך זה בחלק הרלוונטי.

ככל שהפנייה תהיה ברורה יותר, כך נוכל להשיב עליה במהירות.

מומלץ להתחיל את התרגיל בהקדם, על מנת לתכנן ולנצל את הזמן עד להגשה באופן המיטבי.
לא ינתנו הארכות, גם אם מאוד נרצה וזאת לאור הזמן המוגבל עד סוף הסמסטר.

בהצלחה!