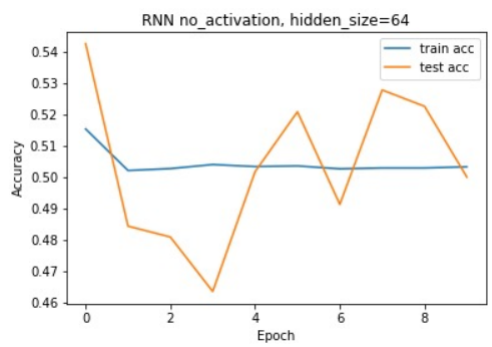
(67822) מבוא ללמידה עמוקה | תרגיל 2

שם: עמית חן (308162502) , נדב אללי (313549206)| בהצלחה :)

14 בדצמבר 2021

חלק פרקטי - דו"ח:

בתרגיל בנינו 4 רשתות - RNN , GRU, MLP, $L-RESET-SELF-ATTENTION$.
תחילה נציג את הניסויים עבור 2 הרשתות הראשונות.
הערה חשובה - כפי שנאמר בתרגול, מדד ה-$Accuracy = \frac{TN+TP}{TOTAL}$ אינו בהכרח המדד שמתאים לתיאור או למדידת טיב הפתרון לכל בעיה, ויש להתחשב במדדים נוספים כדי להרחיב את ההסבר על יכולות המודל שלנו. לכן הוספנו וריאציות נוספות לחישוב ה- $Accuracy$, שהם $F1$ ו-ROC. התוצאות שלהן מופיעות בתרגיל. מפאת העומס שנוצר בדו"ח, ובגלל שסדר הגודל של התוצאות היה זהה , הוספנו גרפים רק ל-$Accuracy = \frac{TN+TP}{TOTAL}$ "הקלאסי".
שאלה 1:
תוצאות עבור רשת ה- RNN:
הערה 1 - כשהרצנו RNN עם $lr = 0.0004$ קיבלנו תוצאות לא טובות. כלומר ה- RNN עובד יותר טוב עם $LR = 0.0001$ וגם ש- GRU עובד יותר טוב עם $LR = 0.0004$ (אנו מציינים זאת מכיוון שכך אכן נעשה עם ה-GRU).
הצדקה לכך שהתוצאות של $lr = 0.0004$ לא טובות:



כעת, עבור כל הריצות הגדרנו את ההיפר פרמטרים הבאים.

$$batch-size = 32$$

$$learning-rate = 0.0001$$

$$num - of - epoches = 10$$

$$test - interval = 50$$

$$optimizer = ADAM$$

$$Loss - criterion = Cross - Entropy$$

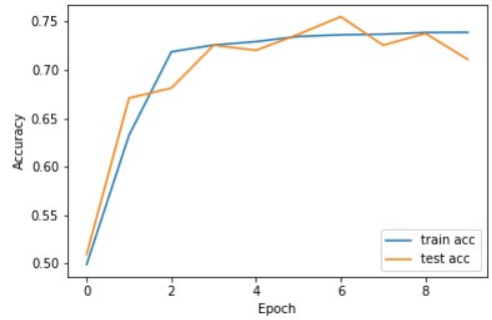
ביצענו ניסויים שונים בשאר ההיפר פרמטרים להלן:
הנוסחא של המבנה הבסיסי (הלמן) היא:

$$\begin{cases} v = h_t = \sigma_h(W_h x_t + U_n h_{t-1} + b_h) \\ o_t = \sigma_o(W_o h_t + b_y) \end{cases}$$

בניסוי הראשון ננסה למצוא את האקטיבציה האופטימלית (σ_h בנוסחה). נתחיל **ללא אקטיבצה כלל**, ונציג את התוצאות של הפעלת אקטיבציה.
נציין שלפי הוראות התרגיל ה־σ_o תמיד יוגדר כ־$sigmoid$.
נציין שחישוב ה־$Accuracy$ מתבצע באופן הבא: $Accuracy = \frac{TP+TN}{TOTAL}$

$$hidden - size = 64$$

$$no\ activation$$



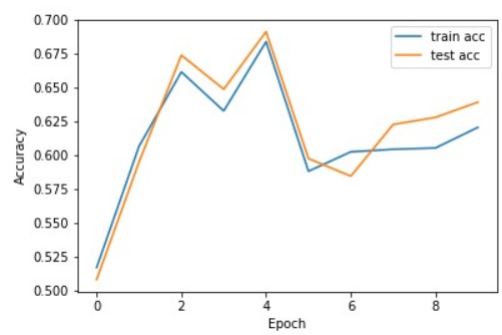
התוצאות:

	$train$	$test$
$Accuracy$	0.73	0.71
$F1$	0.735	0.714
ROC	0.781	0.775

ניתן לראות כי התוצאות הינן **גבוהות** ביחס למודלים הכוללים אקטיבציה, שנציג מיד:

$$hidden - size = 64$$

$$activation = tanh$$

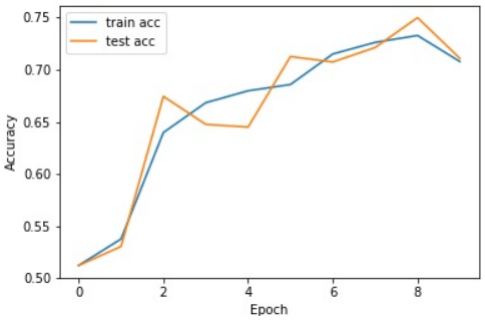


	$train$	$test$
$Accuracy$	0.64	0.62
$F1$	0.642	0.623
ROC	0.69	0.682

ניתן לראות כי התוצאות הינן פחות טובות מהניסוי ללא אקטיבציה, עבור כמות ה־$epoches$ המועטה הנתונה.

$$hidden - size = 64$$

$$activation = ReLU$$

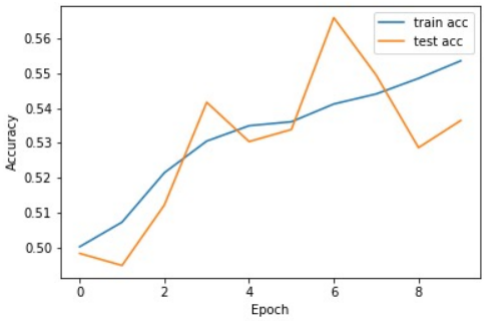


	$train$	$test$
$Accuracy$	0.71	0.709
$F1$	0.718	0.71
ROC	0.731	0.73

ניתן לראות כי התוצאות הינן פחות טובות במעט מהניסוי ללא אקטיבציה, אך טובות יותר מאשר השימוש ב־$tanh$ עבור כמות ה־$epoches$ המועטה הנתונה.

$$hidden - size = 64$$

$$activation = Sigmoid$$



	$train$	$test$
$Accuracy$	0.55	0.53
$F1$	0.552	0.533
ROC	0.591	0.588

ניתן לראות כי התוצאות הינן הכי פחות טובות.
כעת ננסה להריץ את הרשת עם שינוי ה־$hidden-size$. לעת עתה נשאיר את ההיפר פרמטר $no\ activation$ כיוון שראינו שהוא מביא לנו תוצאות טובות. מטעמי נוחות קריאה נמנע מלצרף עוד גרפים, אלא נציג את סיכום התוצאות בטבלה.
בגלל גודל הטבלה נשתמש בקיצורים הבאים:

$$hidden-size = hs$$

$$Train\ Acc, Test\ Acc = Acc$$

$$Train\ F1, Test\ F1 = F1$$

$$Train\ ROC, Test\ ROC = ROC$$

כמו כן נוותר על הסוגריים, אך יש להתייחס לתוצאות כך שהערך הראשון הוא עבור האימון והערך השני הוא עבור הטסט:

$\boldsymbol{hs}$	**64**	**72**	**80**	**88**	**96**	**104**	**112**	**120**	**128**
Acc	$0.73, 0.71$	$0.73, 0.72$	$0.73, 0.73$	$0.73, 0.71$	$0.76, 0.73$	$0.75, 0.73$	$0.75, 0.73$	$0.74, 0.73$	$0.57, 0.55$
$F1$	$0.73, 0.71$	$0.73, 0.72$	$0.73, 0.73$	$0.73, 0.71$	$0.76, 0.73$	$0.75, 0.73$	$0.75, 0.73$	$0.74, 0.73$	$0.57, 0.55$
ROC	$0.74, 0.72$	$0.73, 0.73$	$0.74, 0.73$	$0.74, 0.72$	$0.76, 0.73$	$0.76, 0.73$	$0.76, 0.74$	$0.74, 0.73$	$0.58, 0.56$

לסיכום טיב האקטיבציות לאור הניסויים שביצענו:

1. ללא אקטיבציה - לכן מומלץ לבחור באופציה זו
2. $ReLU$
3. $tanh$
4. $Sigmoid$

הערה - בניתוח היפר פרמטר של $hiddn\ size$ ניתן לראות שיפור בתוצאות עד ל־$hidden - size = 112$ ולאחר מכן ישנה ירידה באיכות התוצאות.
מטעמי מגבלות זמן חישוב ביצענו הרצה אחתת וייתכן כי הרצות נוספות תובילנה לתוצאות שונות.

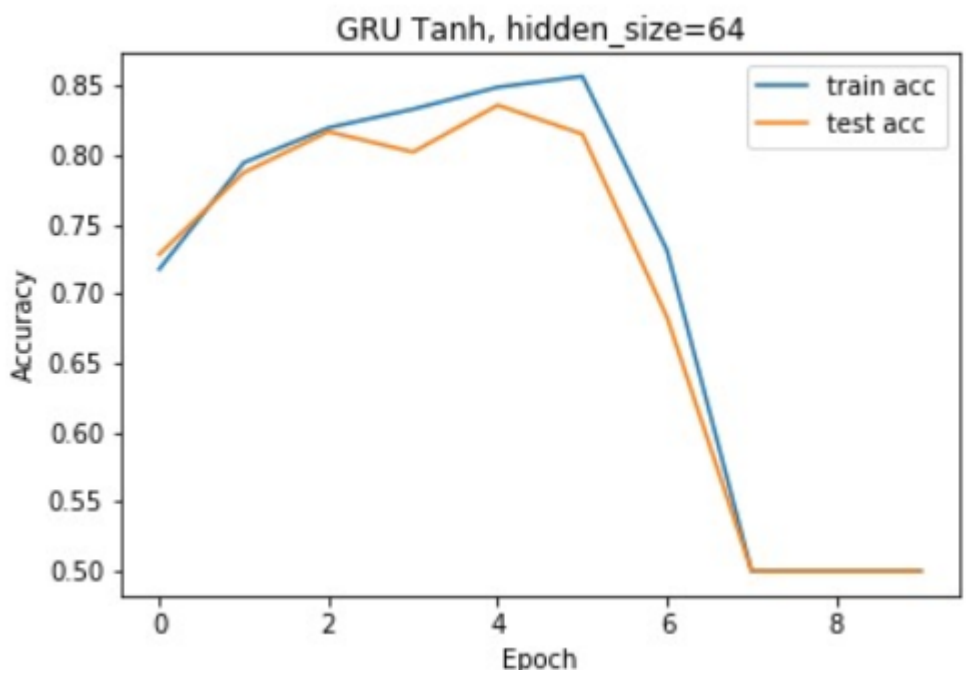
<u>תוצאות עבור רשת ה־ GRU:</u>

באופן דומה לניסוי של רשת ה־ RNN , נפעל בדרך דומה בניסוי שלנו על רשת ה־GRU. נציין שהאקטיבציה הזו רלוונטית לחישוב של $\tilde{h}$:

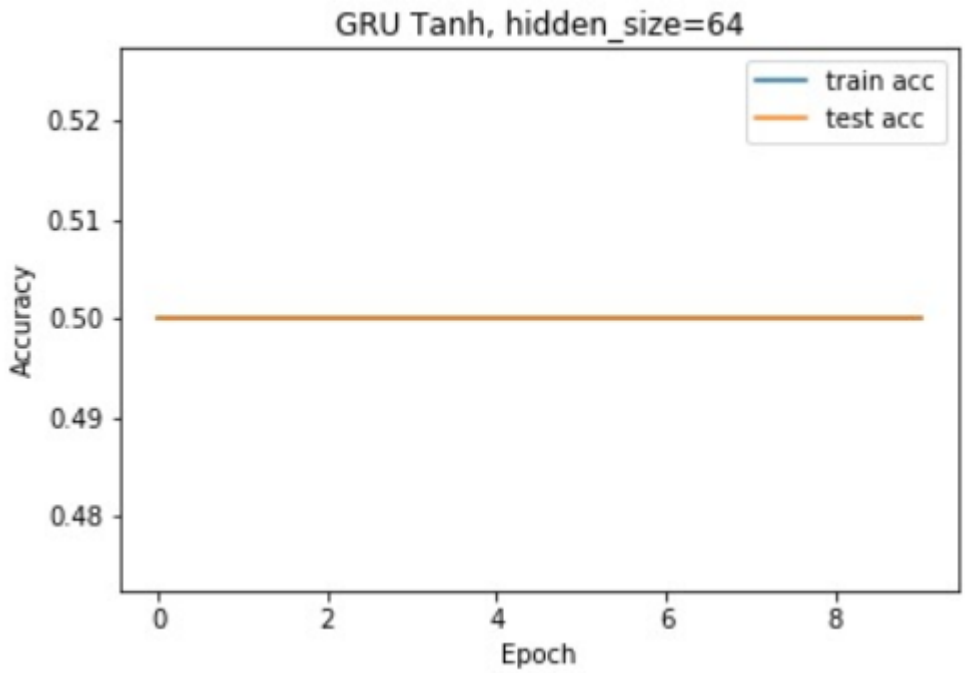
$$Update\ gate:\ z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z),\ reset\ gate:\ r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t] + b),\quad h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

הסיבה שלא שינינו את אקטיבציית ה־σ המופיעה בנוסחת החישוב (כלומר השתמשנו אך ורק באקטיבציית־$Sigmoid$) היא מכיוון שבאקטיבציות אחרות הגענו מהר מאוד ל־$vanish\ gradients$. לדוגמה, עבור $\sigma = ReLU$ נבחין שב־$epoch$ השישי קיבלנו שחיקה של הגרדיאנטים שהפכו ל־$None$, ולכן ה־$Accuracy$ ירד:



ואילו ללא אקטיביציה אפשר לראות שהירידה מופיעה משחיקת הגרדיאנטים כבר מה־$epoch$ הראשון:

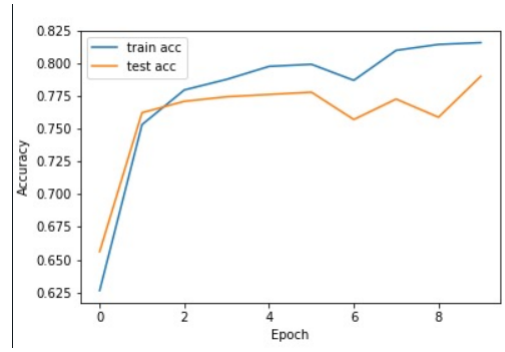


לכן החלטנו להישאר עם ה־$Sigmoid$ שלא הוביל אותנו לשחיקה של הגרדיאנטים.

לאחר המסקנה הזו, להלן הניסויים:

$$hidden - size = 64$$

$$activation = tanh$$



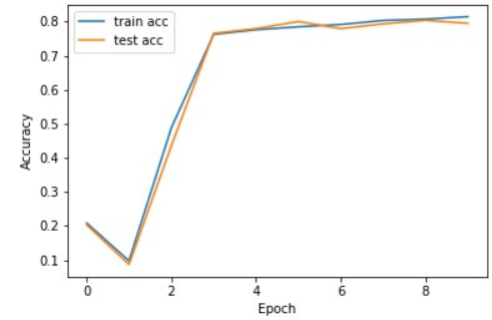
להלן התוצאות:

	$train$	$test$
$Accuracy$	0.815	0.79
$F1$	0.817	0.791
ROC	0.824	0.81

ניתן לראות שהדיוק באימון הינו מעל 0.8, ובסט המבחן כמעט 0.8, בהתאם לציפיות (התייעצנו עם המתרגל מתן בהקשר הזה כדי לראות שהתוצאות מספקות).
ניסוי נוסף:

$$hidden - size = 64$$

$$no\ activation$$



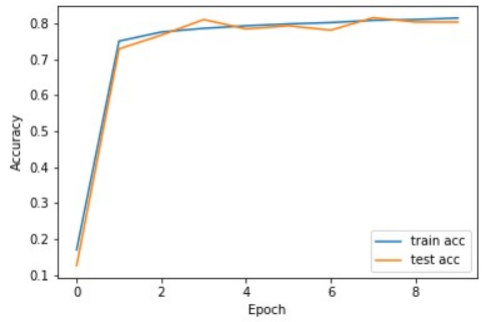
להלן התוצאות:

	$train$	$test$
$Accuracy$	0.815	0.795
$F1$	0.818	0.798
ROC	0.839	0.816

ניתן לראות שתוצאות ללא אקטיבציה יחסית זהות לתוצאות של אקטיבציית ה־$tanh$, אם כי גם ה־$train$ וגם ה־$tset$ קוהרנטיים יותר והגרפים שלהם "דומים יותר".
ניסוי נוסף:

$$hidden - size = 64$$

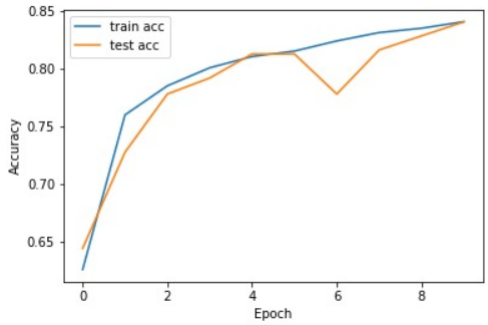
$$activation = Sigmoid$$



ניתן לראות שיפור קל בתוצאות הודות לאקטיבציה זו, לעומת הניסויים הקודמים ($acc : 0.815(train), 0.8(test)$)
ניסוי נוסף:

$$hidden - size = 64$$

$$activation = ReLU$$



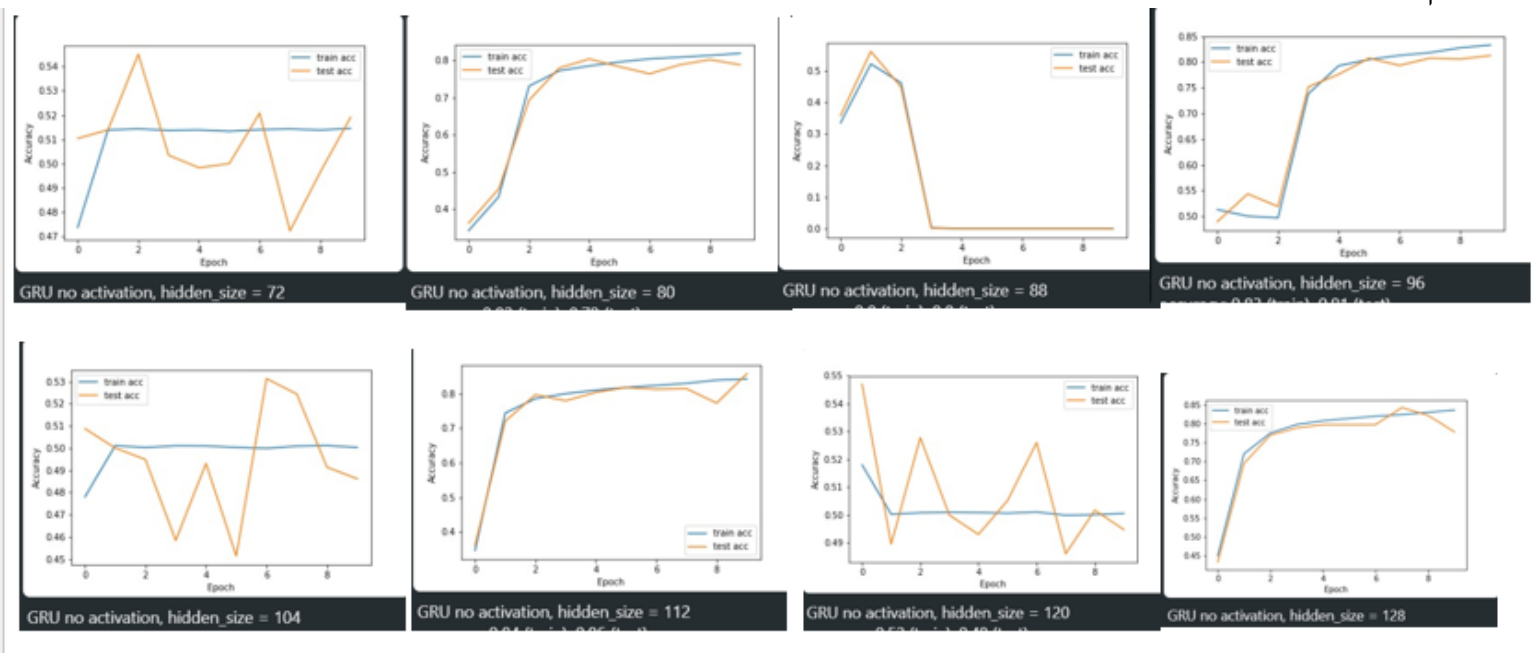
ניתן לראות שיפור בתוצאות הודות לאקטיבציה זו לעומת כל הניסויים הקודמים:

	$train$	$test$
$Accuracy$	0.84	0.84
$F1$	0.842	0.741
ROC	0.864	0.842

אם כן, ניתן לראות שנכון להרצה זו ישנו חוסר יציבות קל בסט המבחן.
כעת נרצהת כפי שעשינו עבור ה־RNN לבחון את הפשעת הפרמטר של ה־$hidden\ size$:
נציין את הנתונים בטבלה ואז נוסיף את הגרפים, לנוחות הקריאה:
עבור המודל "ללא אקטיבציה":

hs	64	72	80	88	96	104	112	120	128
Acc	$0.81, 0.79$	$0.51, 0.52$	$0.82, 0.79$	$0, 0$	$0.83, 0.81$	$0.5, 0.48$	$0.84, 0.86$	$0.52, 0.48$	$0.82, 0.83$
$F1$	$0.81, 0.79$	$0.51, 0.52$	$0.82, 0.79$	$0, 0$	$0.83, 0.81$	$0.5, 0.48$	$0.84, 0.86$	$0.52, 0.48$	$0.82, 0.83$
ROC	$0.82, 0.79$	$0.53, 0.53$	$0.75, 0.79$	$0, 0$	$0.84, 0.81$	$0.51, 0.49$	$0.85, 0.86$	$0.53, 0.49$	$0.83, 0.83$

להלן הגרפים:

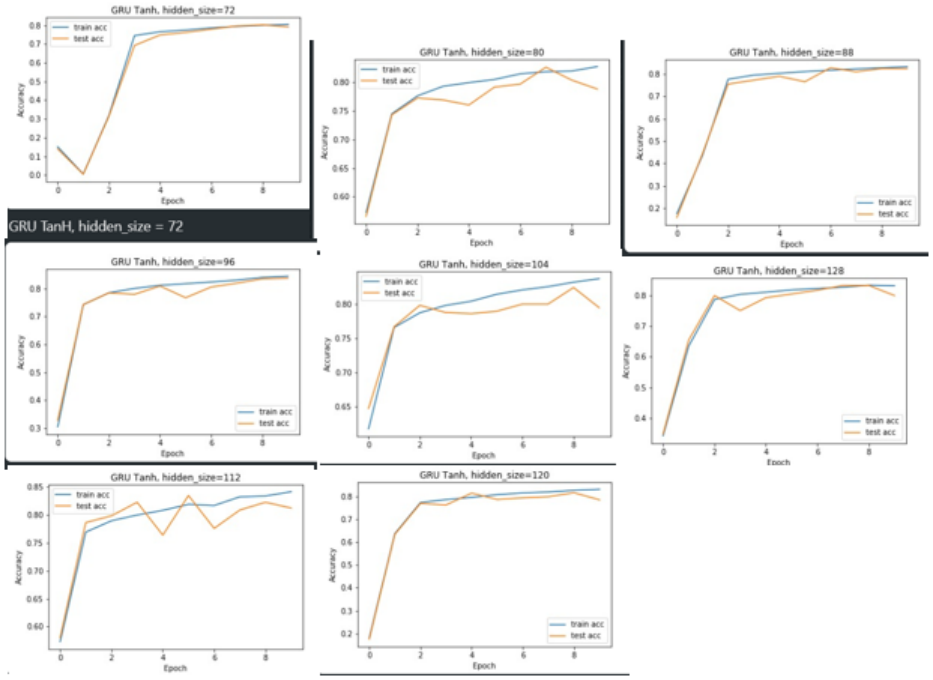


עבור המודל עם אקטיביציית $tanh$**:**

$hidden-size$	64	72	80	88	96	104	112	120	128
$Train\ Acc$	0.815	0.8	0.83	0.83	0.84	0.84	0.84	0.83	0.83
$Test\ Acc$	0.79	0.79	0.79	0.82	0.83	0.795	0.81	0.78	0.8

hs	64	72	80	88	96	104	112	120	128
Acc	$0.81, 0.79$	$0.8, 0.79$	$0.83, 0.79$	$0.83, 0.82$	$0.84, 0.83$	$0.84, 0.79$	$0.84, 0.81$	$0.83, 0.78$	$0.83, 0.8$
$F1$	$0.81, 0.79$	$0.8, 0.79$	$0.83, 0.79$	$0.83, 0.82$	$0.84, 0.83$	$0.84, 0.79$	$0.84, 0.81$	$0.83, 0.78$	$0.83, 0.8$
ROC	$0.82, 0.79$	$0.81, 0.8$	$0.84, 0.8$	$0.84, 0.83$	$0.84, 0.83$	$0.85, 0.8$	$0.85, 0.82$	$0.83, 0.79$	$0.83, 0.81$

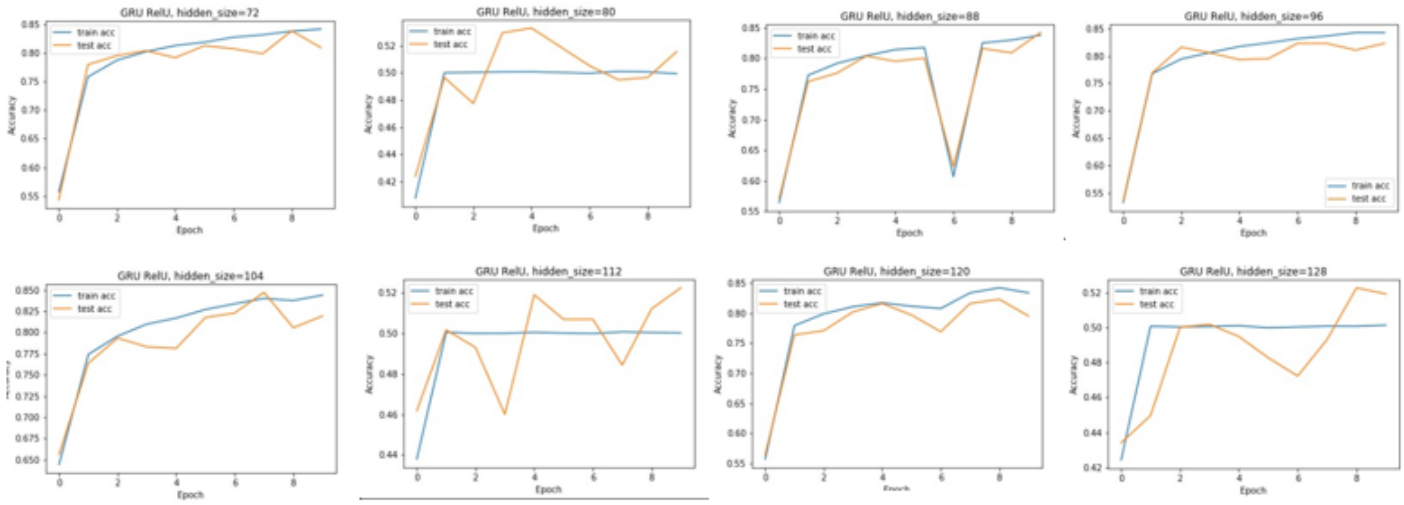
להלן הגרפים:



עבור המודל עם אקטיביציית $ReLU$**:**

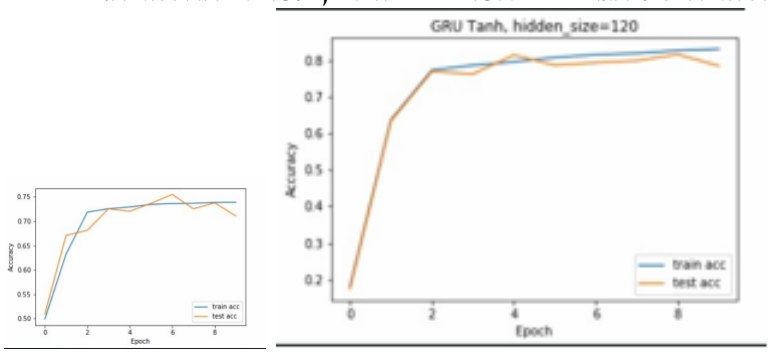
hs	64	72	80	88	96	104	112	120	128
Acc	$0.84, 0.84$	$0.84, 0.81$	$0.51, 0.49$	$0.84, 0.84$	$0.84, 0.82$	$0.84, 0.82$	$0.52, 0.5$	$0.83, 0.79$	$0.52, 0.5$
$F1$	$0.84, 0.84$	$0.84, 0.81$	$0.51, 0.49$	$0.84, 0.84$	$0.84, 0.82$	$0.84, 0.82$	$0.52, 0.5$	$0.83, 0.79$	$0.52, 0.5$
ROC	$0.85, 0.84$	$0.85, 0.82$	$0.52, 0.5$	$0.85, 0.84$	$0.85, 0.83$	$0.84, 0.82$	$0.53, 0.51$	$0.84, 0.8$	$0.53, 0.51$

להלן הגרפים:



לסיכום טיב האקטיבציות לאור הניסויים שביצענו היא עם אקטיבציית $tanh$ ועם $hidden-size = 120$, כאשר $hidden-$

$size = 112$ הגיע לדיוק גבוה יותר (כמעט 0.85) אך היה פחות יציב בתוצאותיו (ניתן לבחון זאת בגרפים שצירפנו לעיל).
כמו כן, ראינו כי בהשוואה בין RNN לבין GRU , תוצאות מודל ה־GRU היו טובות יותר:
השוואה של הגרפים "הטובים ביותר", לטובת ההשוואה:



אנו מסיקים כי קיבלנו את התוצאות הללו מכיוון ש:

1. GRU טוב יותר מ־RNN בגלל שכמות ה־FC ב־GRU גדולה יותר ולכן הפונקציות שהמודל הזה יכול ליצור מגוונות יותר.
2. למדנו שמבנה ה־GRU מגוון יותר, ובפרט האפשרות של $reset - gate$ ושל $update - gate$ מגביר את הפונקציונליות של המודל.
3. למעשה ראינו כי GRU עם $hidden - size = 64$ יכול להביא תוצאות טובות יותר מ־RNN עם $hidden - size = 128$, כלומר עם גודל כפול. הגדלה זו נעשית עבור שנוכל להשוות את כמות המשקולות (מפי שעבור GRU יש פי 2 יותר משקולות פנימיות במודל). חרץ שינוי זה, המודל של GRU עדיין נותן תוצאות טובות יותר, גם לאחר השוואת הנתונים המקדימים של הרשת.

שאלה 2:

<u>תוצאות עבור רשת ה־ MLP:</u>

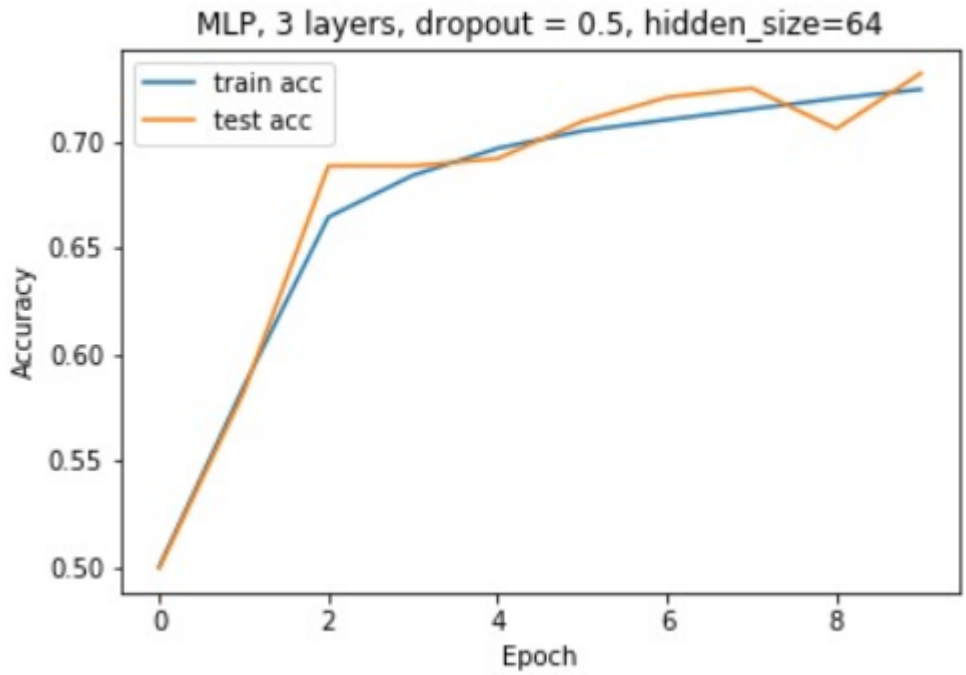
היפר הפרמטרים דומים לאלו שהצגנו במודלים קודמים, במידה ויש שינוי של היפרפרמטר נציין אותו בהצגת הגרף:
הפרמטר k מציין את מספר השכנים, כמתואר בתרגיל:

$$without\ attention$$

$$hidden - size = 64$$

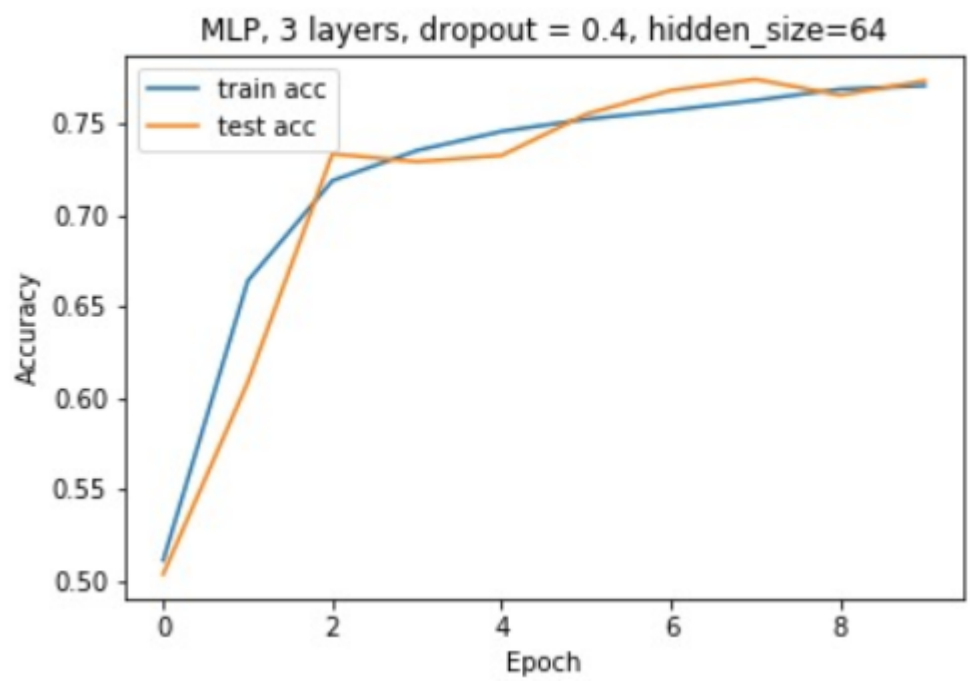
את הפרמטר האחרון ($hidden - size$) לא שינינו בניסויים שלנו מכיוון שהעדפנו לבחון את המודל שלנו תוך שינוי של מימדים בשכבות הביניים שמגיעות לאחר מכן ברשת.
המודל הבא הוא מודל בעל 3 שכבות שמבנהו הוא:

$$Linear(hiddensize, 48), Relu, Linear(48, 24), Dropout = 0.5, MatMul(24, 12)$$



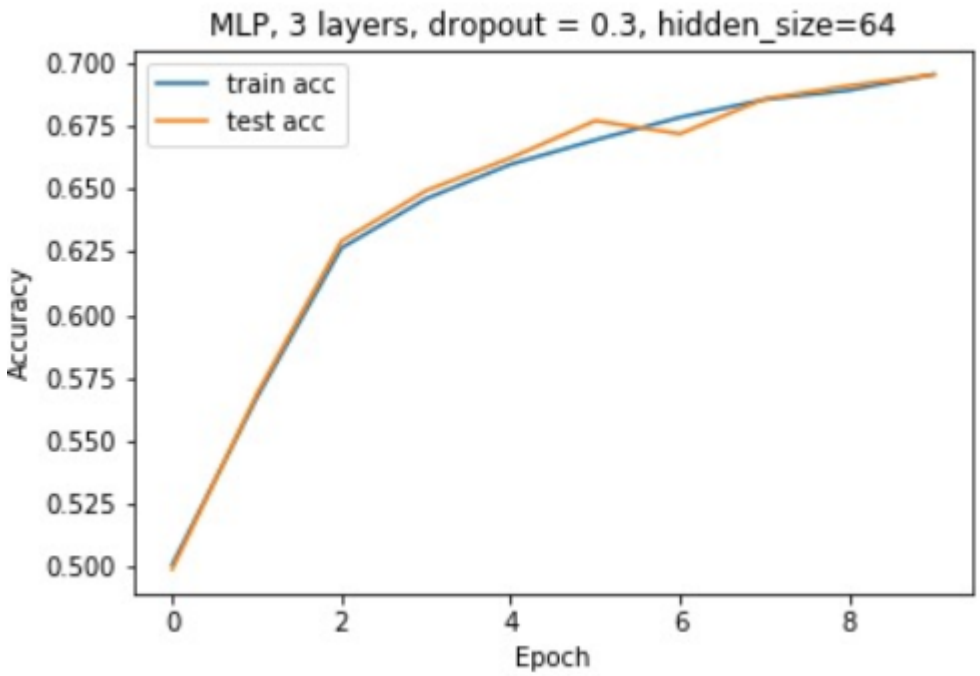
התוצאות:

	$train$	$test$
$Accuracy$	0.73	0.72
$F1$	0.721	0.714
ROC	0.738	0.724



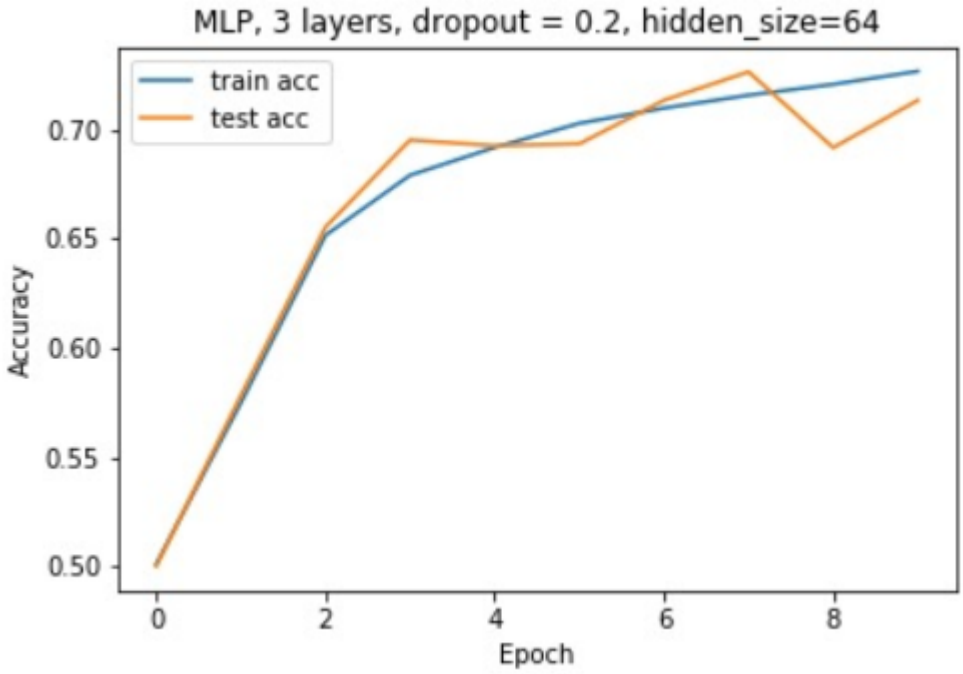
התוצאות:

	$train$	$test$
$Accuracy$	0.77	0.77
$F1$	0.751	0.753
ROC	0.779	0.775



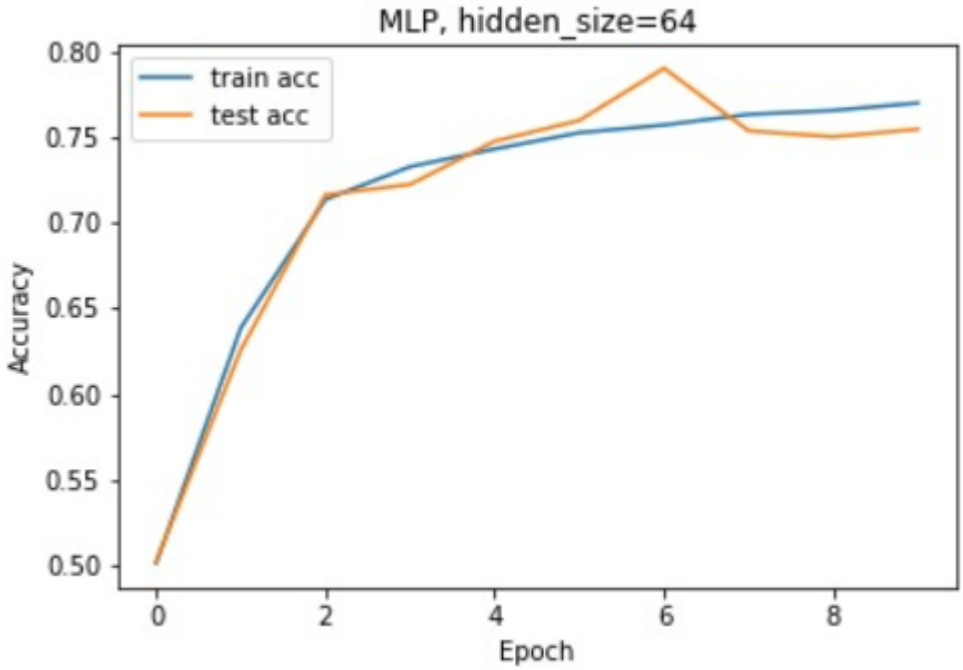
התוצאות:

	$train$	$test$
$Accuracy$	0.69	0.69
$F1$	0.669	0.65
ROC	0.73	0.71



התוצאות:

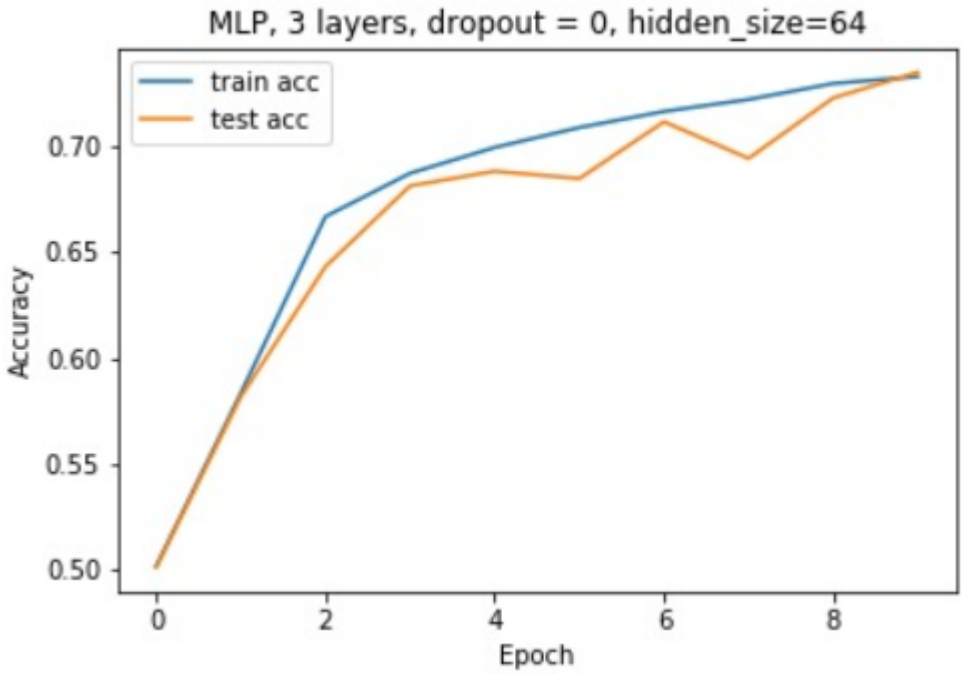
	$train$	$test$
$Accuracy$	0.72	0.71
$F1$	0.7	0.69
ROC	0.728	0.716



נציין שהפעלנו פה $Dropout = 0.1$

התוצאות:

	$train$	$test$
$Accuracy$	0.77	0.75
$F1$	0.761	0.74
ROC	0.778	0.753



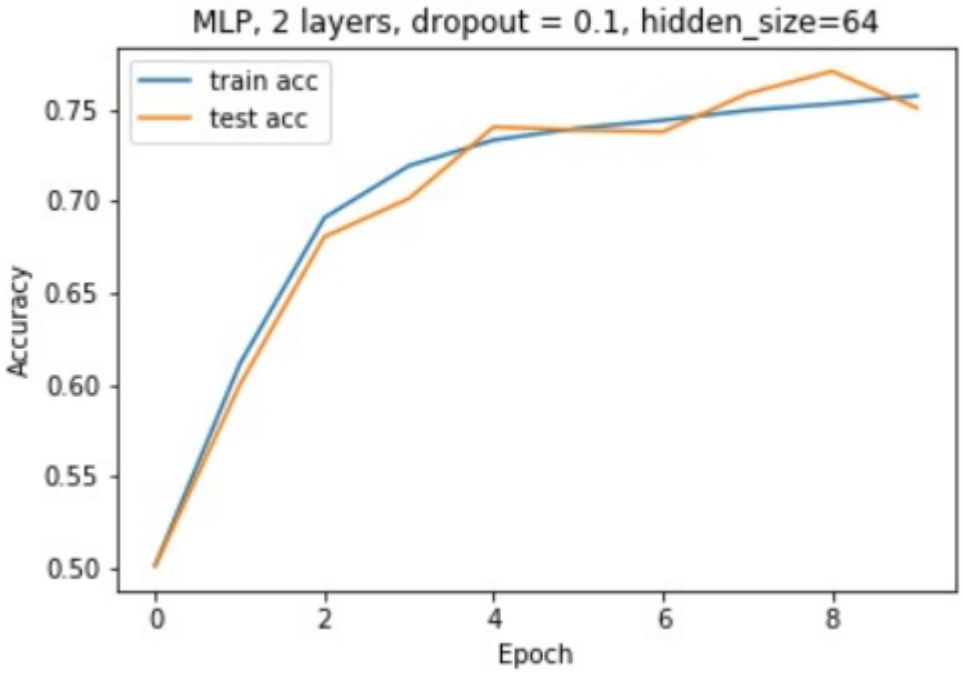
התוצאות:

	$train$	$test$
$Accuracy$	0.73	0.73
$F1$	0.715	0.713
ROC	0.739	0.738

הגענו למסקנה כי אם ה־$Dropout = 0.1$ הגענו לתוצאות הטובות (**מכיוון שהן גם היציבות**) ביותר מאשר שאר הניסיונות.
לכן ננסה כעת מספר שכבות נוספות עם אותו $Dropout$

כעת נבחן מספר שכבות נוספות:
2 שכבות:

$$Linear(hiddensize, 48), Relu, Linear(48, 24), Dropout$$

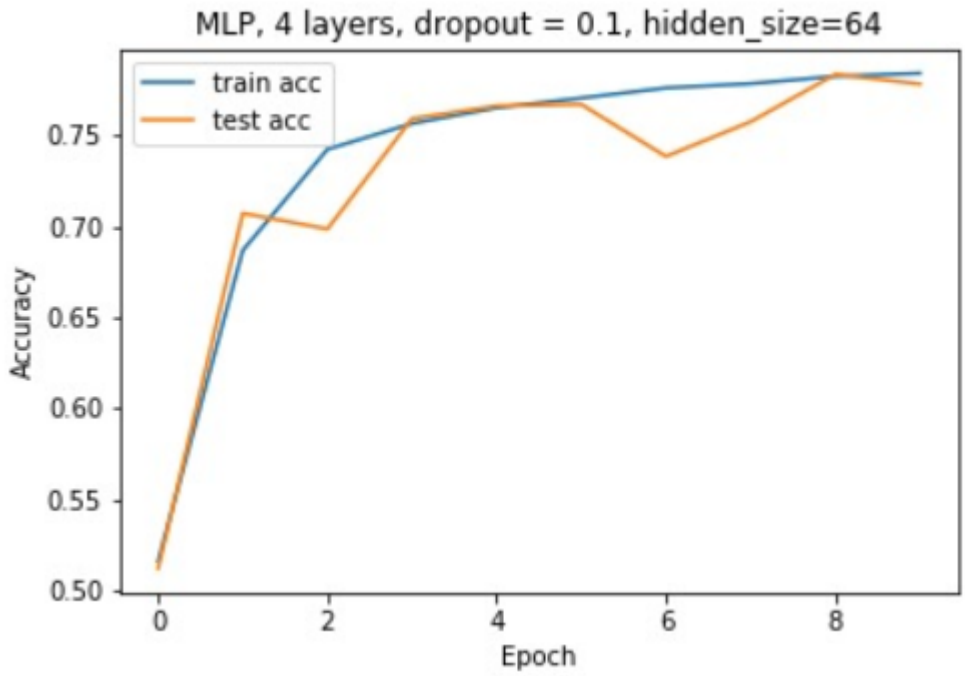


התוצאות:

	$train$	$test$
$Accuracy$	0.755	0.755
$F1$	0.748	0.745
ROC	0.761	0.759

כעת נבחן 4 שכבות:

$$Linear(hiddensize, 48), Relu, Linear(48, 24), Dropout, MatMul(24, 12), Relu, MatMul(12, 8)$$

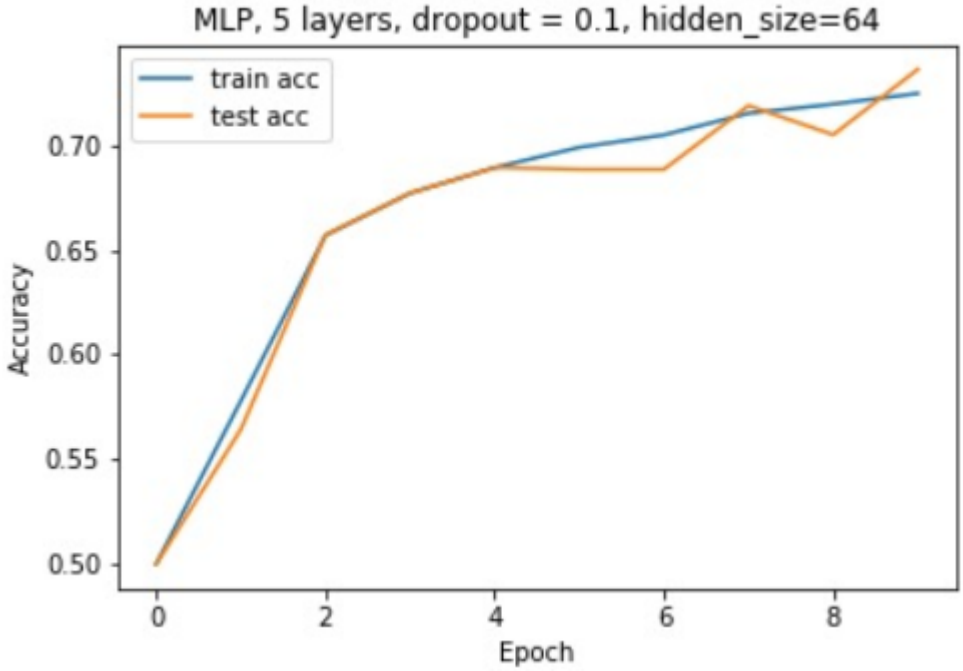


התוצאות:

	$train$	$test$
$Accuracy$	0.755	0.75
$F1$	0.746	0.743
ROC	0.79	0.782

כעת נבחן 5 שכבות:

$$Linear(hiddensize, 48), Relu, Linear(48, 24), Dropout, MatMul(24, 12), Relu, MatMul(12, 10), MatMul(10, 8)$$



התוצאות:

	$train$	$test$
$Accuracy$	0.72	0.73
$F1$	0.71	0.718
ROC	0.77	0.769

ניתן לראות שהתוצאה הטובה ביותר היא של הרשת $MLP, 4\ layers, dropout = 0.1$ עם: $Accuracy : 0.775(train), 0.78(test)$.
מסקנה - מבין רשתות ה-MLP נבחר בזו עם התוצאות הטובות ביותרת כלומר הרשת עם 4 השכבות.
הערה 2 - נשים לב שהתוצאות של $F1$ היו במעט נמוכות יותר מאשר של ה-$Accuracy$ "הקלאסי". זאת לעומת מודלים אחרים בהם ראינו כי ה- $F1$ היו במעט גבוהות יותר מאשר של ה-$Accuracy$ "הקלאסי".

עבור משימת ההדפסות

ניתן לראות בטבלה את הערכים החיוביים והשליליים עבור כל מילה במשפט. לבסוף ניתן לראות את הממוצע עבור החלק החיובי והשלילי (בנפרד) במשפט, כך שהמקסימלי מבינהם יעניק לנו את הפרדיקציה שלנו.
כפי שניתן לראות, ממוצע ה-$negative$ גדול מאשר ממוצע ה-$positive$, ולכן ההדפסות הנדרשות בוצעו כמצופה:
"משפט שלילי":

```
-------------------------------------------------------------
Sentence: this movie is very very bad the worst movie
Scores:
  Word   Positive   Negative
  this   0.087675  -0.942039
 movie  -3.609716   3.008058
    is   1.960706  -2.364913
  very   7.240577  -8.877016
  very   6.978969  -8.773827
   bad -38.106655  38.857059
   the   0.211410  -0.562696
 worst -43.445297  44.206684
 movie  -2.263288   2.054891

Positive Score: -0.5093562602996826
Negative Score: 0.10716469585895538

Sentence Sentiment: Negative
Prediction Sentiment: Negative

Predicted right!!!
-------------------------------------------------------------
```

ניתן לראות שעדיין ישנן מילים שההקשר שלהן היה שלילי, אך מכיוון שהסתכלות עליהן באופן נפרד, הן לא מקבלות קונוטצייה שלילית. לדוגמא המילים $very$ שבהקשר הן מקושרות למילה bad , אמנם הן מיוצגות עם ציון חיובי לכשעצמן.

לעומת זאת, בטסט השני ("משפט עם שלילה") ההדפסה אינה תואמת לציפיות שלנו:

```
-----------------------------------------------------
Sentence: calling this movie very very good and great
          is lie and it actually the worst
          movie ever
Scores:
     Word   Positive   Negative
  calling  -0.107223   0.156273
     this   0.126125   0.009624
    movie  -0.917017   0.825654
     very   6.173283  -3.913103
     very   6.553571  -4.520661
     good   8.235043  -5.337615
      and   3.795273  -2.426528
    great  24.630003 -17.998999
       is   3.053270  -1.922994
      lie   0.367408  -0.025914
      and   2.778945  -1.637273
       it   6.300950  -4.485884
 actually   0.349848  -0.172744
      the  -0.106051   0.151251
    worst -92.218201  64.672966
    movie  -1.605782   1.336014
     ever   9.980879  -7.409322

Positive Score: 0.17793887853622437
Negative Score: -0.041215624660253525

Sentence Sentiment: Negative
Prediction Sentiment: Positive

Predicted wrong
-----------------------------------------------------
```

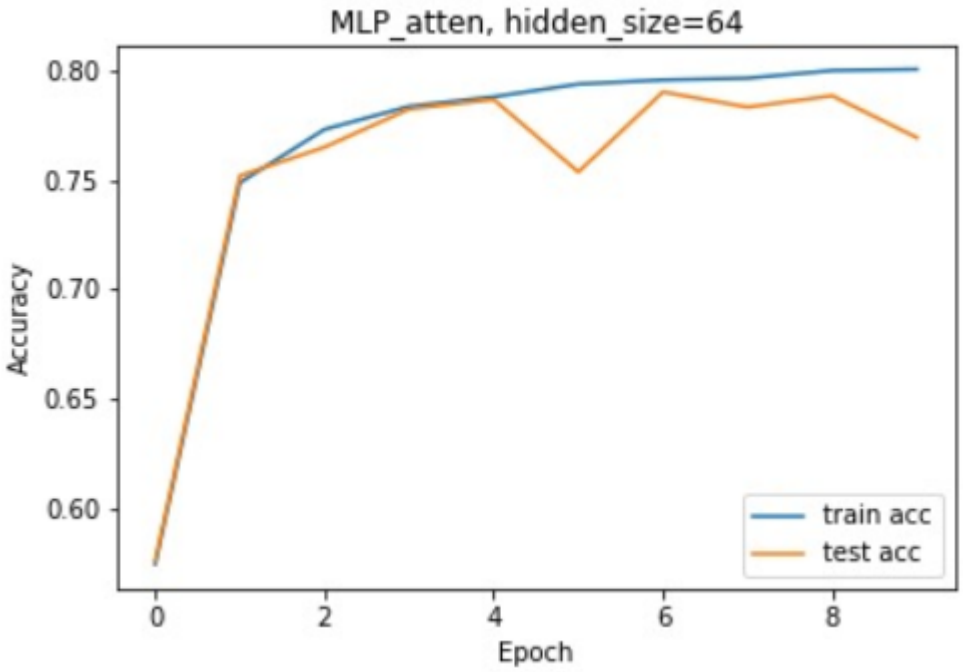
להבנתנו במשפט הראשון היו מילים שליליות ברובו של המשפט.

אולם המשפט השני מכיל מילים חיוביות אך עם תוספת **מילה שלילית מאוד**, שהופכת את המשמעות של המשפט, ולכן ללא הקשר לא נוכל לנבא בצורה טובה את התיוג של המשפט.

לדוגמאת המילה $ever$ מיוצגת בייחוס חיובי, אולם ההקשר שלה במשפט נקשר למילה $worst$. לכן פער זה משפיע על החישוב שלנו ובסופו של דבר פוגע בפרדיקציה שלנו.

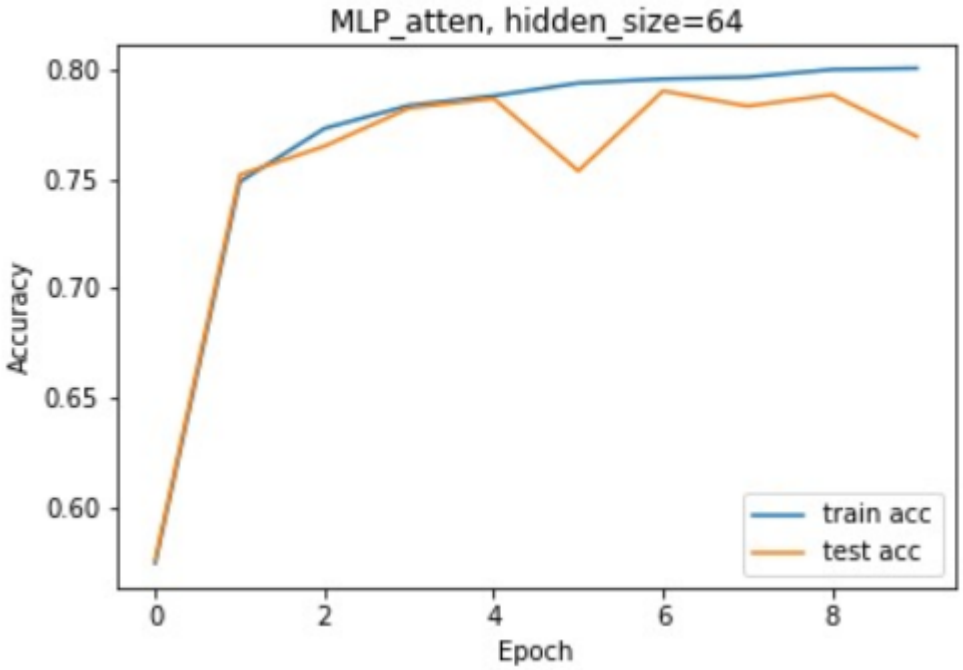
שאלה 3 - מימשנו את הנדרש , כעת עם $Attention$, להלן התוצאות:

ניסוי עבור 4 שכבות של MLP:



	$train$	$test$
$Accuracy$	0.8	0.77
$F1$	0.802	0.771
ROC	0.821	0.795

ניסוי עבור 2 שכבות של MLP:



	$train$	$test$
$Accuracy$	0.8	0.76
$F1$	0.806	0.764
ROC	0.844	0.821

נשים לב שאכן מקבלים תוצאות טובות יותר עבור $MLP + ATTENTION$. כלומר ניתן להסיק כי ש־ $MLP + ATTENTION > MLP$, כמצופה.

הערה - נציין למען הסר ספק ששאלה 3 המופיעה בקובץ השאלות אינה דורשת תשובה כתובה, אלא מימוש שביצענו והצגנו את תוצאותיו.

שאלה 4 -

<u>התוצאות עבור ה־ $Attention$:</u>

"משפט עם שלילה":

```
-------------------------------------------------------
Sentence: calling this movie very very good and great
          is lie and it actually the worst
          movie ever
Scores:
     Word   Positive   Negative
  calling  -4.266939   2.715056
     this   9.993629  -5.997334
    movie  15.856782  -9.767759
     very  22.356541 -13.574579
     very  20.453474 -13.136784
     good  10.289974  -6.148139
      and   2.997212  -2.166039
    great  11.281944  -6.236628
       is  15.813581 -10.516915
      lie  -7.614808   5.035130
      and  -0.246497   0.119621
       it -22.474375  15.816313
 actually  -7.876347   5.137033
      the -14.483573  10.434888
    worst -20.035961  14.763971
    movie -28.805351  19.440060
     ever -23.932436  16.221304

Positive Score: -0.10124897956848145
Negative Score: 0.05544460192322731

Sentence Sentiment: Negative
Prediction Sentiment: Negative

Predicted right!!!
-------------------------------------------------------
```

ניתן לראות שהמודל של ה־$Attention$ הצליח לזהות שההקשר של המשפט הוא שלילי. לצורך ההשוואה, המודל ללא ה־$Attention$ לא הצליח לזהות שמדובר במשפט שלילי. ניתן לראות שהחלק הראשון של המשפט יוצג כחיובי, ואילו החלק השני הוצג כשלילי. כלומר המודל הצליח להבין שאנו שוללים את המשפט הראשון ולכן נתן ציון שלילי בסך הכל במשפט.

דוגמא נוספת:

```
-----------------------------------------
Sentence: this movie is great said no one
Scores:
  Word   Positive  Negative
  this   6.833419 -4.411019
 movie  -3.198535  1.848296
    is  12.130595 -7.494356
 great   8.499954 -5.767892
  said  -0.250540  0.019032
    no  -2.502690  1.545835
   one   3.943444 -2.749191

Positive Score: 0.376842200756073
Negative Score: -0.35723990201950073

Sentence Sentiment: Negative
Prediction Sentiment: Positive

Predicted wrong
-----------------------------------------
```

ניתן לראות שכעת המודל לא הצליח לזהות את תההקשר השלילי של המשפט, ככל הנראה מכיוון שהתקשה לזהות את ההקשר השלילי של החלק השני לעומת ההקשר החיובי של החלק הראשון של המשפט (כלומר "$said\ no\ one$" לא תמיד מקושר באופן שלילי, לכן ייתכן שבשימוש במודל מורכב יותר היה ניתן לפרש את ההקשר שלו, ספציפית לדגומא הנ"ל).

חלק תאורטי:

שאלה 1:

1. Explain what type of a network architecture you will use to handle each of the following problems (e.g., many-to-many RNN, or a convolution NN). Explain your reasoning.
    a. Speech recognition (audio to text)
    b. Answer questions
    c. Sentiment analysis
    d. Image classification
    e. Single word translation

נסביר איזו ארכיטקטורת רשת תשמש אותנו עבור הבעיות הבאות.

א. עבור בעיית **תרגום אודיו לטקסט** נשתמש ברשת מסוג $many\ to\ many\ RNN$ שלמדנו. תחילה נבדוק את תקינות הקלט שלנו (ראינו בקורס בשנה שעברה, "עיבוד תמונה", שניתן לחלק את ה־$Speech$ למקטעי זמן (חלונות באורכים קבועים), כך שרצף מקטעים זה יהיה הקלט לבעיה שלנו. בחרנו ברשת זו מכיוון שניתן להציג את הקלט בתור $time\ series$, וגם את הפלט ניתן להציג כ־$time\ series$. ואנו מתבקשים לתרגם מ־$time\ series$ אחד ל־$time\ series$ אחר תוך שימוש במידע ממילים קודמות (בקלט). נציין כי השימוש ב־$many\ to\ many$ נובע מכך שאורך הקלט והפלט אינם ידועים, כיוון שאיננו יודעים על קלטים או פלטים קבועים בבעיות מסוג זה. בנוסף, ראינו בהרצאה של רענן השבוע שימוש ב־$self\ attention$ כדי לאפשר לרשת שלנו להתייחס ל־$context$ בין המילים בבעיה שלנו, ולכן ניתן להוסיף שכבה שכזו לרשת שלנו כדי לשפר את הביצועים שלה.

ב. עבור בעיית **מענה על שאלות** נשתמש ברשת $many\ to\ many\ RNN$, זאת מכיוון ששאלה יכולה להיות באורך שונה מאשר התשובה. שאר התשובה לסעיף (כלומר הפירוט) דומה לסעיף קודם

ג. עבור בעיית **ניתוח משפטים** נשתמש ברשת מסוג $many\ to\ one\ RNN$ שלמדנו. נשים לב שהקלט הוא מסוג $time\ series$, לכן הקלט בגודל לא ידוע (אורך המשפט שנקבל). לעומת זאת, גודל הפלט הוא קבוע מכיוון שהפלט , שלהבנתו בא לידי ביטוי על ידי קלאסיפיקציה (סיווג) ניתן לבחירתנו (לדוגמא טקסט עם קונוטציה חיובית\שלילית ואז הפלט הוא בינארי, או קביעת סקאלה של ערכים (כמו דירוג של מוצרים או שירותים).

בנוסף נדגיש שמומלץ להוסיף שימוש ב־$self\ attention$ כדי לאפשר לרשת שלנו להתייחס ל־$context$ בין המילים בקלט שלנו (על מנת להבין את הקשרים בין המילים במשפט).

ד. עבור בעיית **סיווג תמונה** נשתמש ברשת מסוג CNN. רשתות מסוג זה מקבלות כקלט תמונות ומבצעות קונבולוציות עם פילטרים שונים, כך שהרשת "לומדת" פיצ'רים של התמונות. בעזרת פיצ'רים אלו הרשת מנסה לסווג את הקלט. ראינו בכיתה דוגמאות לרשתות כאלו כמו $AlexNet$ (או $ResNet$ שעושה שימוש ב־$residuals$ כדי למנוע את בעיית ה־$vanish\ Gradient$ או $exploding\ Gradient$ ברשתות עמוקות).

ה. עבור בעיית **תרגום מילה אחת** נשתמש ברשת מסוג $one\ to\ many\ RNN$:

תחילה נשים לב שיש לנו $embedding$ (שיכון), כלומר ייצוג וקטורי של המילים משפת המקור, לשפה שנרצה לתרגם אליה. ראינו בכיתה שיש חשיבות לתהליך זה ושיש שיטות לביצוע של משימה זו (כמו $Glove$ או $word2vec$). כעת, מכיוון שמילה יכולה לקבל פירוש שונה בהתאם להקשר, נציע שהפלט יהיה $many$ כדי שנוכל לתת למשתמש תרגום אחד או מספר אפשרויות לתרגום (במקרה זה אין לנו יכולת להבין את ההקשר כי קיבלנו בקלט מילה בודדת, ולא משפט). דוגמא לכך היא תרגום המילה $date$, שראינו בכיתה שיכולה לקבל את הפירוש "תמר","צום","מפגש". לכן נקבל מספר אפשרויות של תרגום למילה.

שאלה 2:

נניח שהמימד של הוקטור x הוא $n \in \mathbb{N}$

2. Compute the analytical Jacobian of the following layers:
   a. Linear layers Ax
   b. Bias layers x+b
   c. Convolutional layers x * f

א. נשים לב שהמכפלה בין המטריצה A לבין הוקטור x מחזיר לנו וקטור.

נסמן את המטריצה A ואת הוקטור x :

$$A = \begin{bmatrix} A_{1,1} & . & . & A_{1,n} \\ . & . & . & . \\ . & . & . & . \\ A_{n,1} & . & . & A_{n,n} \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ . \\ . \\ x_n \end{bmatrix}$$

לכן נגדיר מכפלה זו עם הפונקציה f באופן הבא:

$$f(x) = \begin{bmatrix} f_1(x) \\ . \\ . \\ f_n(x) \end{bmatrix} = Ax$$

כך שהפעלת הפונקציה בכל קורדינטה שקולה להכפלת המטריצה בוקטור כך"

$$f(x) = \begin{bmatrix} \sum_{i=1}^{n} A_{1,i} \cdot x_i \\ . \\ . \\ \sum_{i=1}^{n} A_{n,i} \cdot x_i \end{bmatrix}$$

$$J_f = \frac{\partial f(x)}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & . & . & \frac{\partial f_1}{\partial x_n} \\ . & . & . & . \\ . & . & . & . \\ \frac{\partial f_n}{\partial x_1} & . & . & \frac{\partial f_n}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial\left(\sum_{i=1}^{n} A_{1,i} \cdot x_i\right)}{\partial x_1} \\ . \\ . \\ \frac{\partial\left(\sum_{i=1}^{n} A_{n,i} \cdot x_i\right)}{\partial x_n} \end{bmatrix} = \begin{bmatrix} A_{1,1} & . & . & A_{1,n} \\ . & . & . & . \\ . & . & . & . \\ A_{n,1} & . & . & A_{n,n} \end{bmatrix} = A$$

ולכן היעקוביאן של Ax הוא למעשה המטריצה המכפילה A.

ב. מכיוון שמדובר ב־"הזזה" של הוקטור x , נסמן האותו האופן כמו בסעיף קודם:

$$x = \begin{bmatrix} x_1 \\ . \\ . \\ x_n \end{bmatrix}, b = \begin{bmatrix} b_1 \\ . \\ . \\ b_n \end{bmatrix}$$

לכן:

$$f(x) = x + b = \begin{bmatrix} x_1 + b_1 \\ . \\ . \\ x_n + b_n \end{bmatrix}$$

$$J_f = \frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & . & . & \frac{\partial f_1}{\partial x_n} \\ . & . & . & . \\ . & . & . & . \\ \frac{\partial f_n}{\partial x_1} & . & . & \frac{\partial f_n}{\partial x_n} \end{bmatrix} = \frac{\partial (x+b)}{\partial x} = \begin{bmatrix} \frac{\partial x_1+b_1}{\partial x_1} & . & . & \frac{\partial x_1+b_1}{\partial x_n} \\ . & . & . & . \\ . & . & . & . \\ \frac{\partial x_n+b_n}{\partial x_1} & . & . & \frac{\partial x_n+b_n}{\partial x_n} \end{bmatrix} = \begin{bmatrix} 1 & . & . & 0 \\ . & 1 & . & . \\ . & . & 1 & . \\ 0 & . & . & 1 \end{bmatrix} = I_{nxn}$$

כלומר היעקוביאן של הזזת וקטור היא מטריצת היחידה מהמימד של הוקטור.

ג. נחשב יעקוביאן של שכבת קונבולוציה :

נסמן כרגיל

$$x = \begin{bmatrix} x_1 \\ . \\ . \\ x_n \end{bmatrix}$$

נזכיר שראינו בכיתה שגודל k רכיב הקונבולוציה (הקרנל) קטן משמעותית מגודלו של x (כלומר אם מימד הפילטר הוא k' אז $k' << n$).

ניזכר בהגדרה של קונבולוציה:

$$y_i = \sum_{l=1}^{n} x_l \cdot k_{i-l+1}$$

לכן:

$$J_{i,l} = \frac{\partial y_i}{\partial x_l} = \begin{cases} k_{i-l+1} & i > l+1 \\ 0 & otherwise \end{cases}$$

לכן מטריצת היעקוביאן מורכבת משורות של הזזות של רכיבי הקרנל, כל k_1 "תזוז" צעד ימינה בכל שורה, עד להופעת $k_{k'}$ ברכיב הראשון של השורה ה־k', ולאחר מכן $k_{k'}$ "תזוז" צעד ימינה בכל שורה עד שתגיע לרכיב האחרון בשורה האחרונה:

$$J = \begin{bmatrix} \boldsymbol{k_1} & 0 & 0 & .. & .. & 0 \\ k_2 & \boldsymbol{k_1} & 0 & .. & .. & .. \\ .. & .. & .. & .. & .. & 0 \\ k_{k'-1} & .. & .. & .. & \boldsymbol{k_1} & 0 \\ k_{k'} & k_{k'-1} & .. & .. & k_2 & \boldsymbol{k_1} \\ 0 & k_{k'} & .. & .. & .. & 0 \\ .. & 0 & .. & & & .. \\ .. & .. & & & & 0 \\ 0 & .. & & & .. & k_{k'} \end{bmatrix}$$

כנדרש.

שאלה 3:

3. Text-to-Image:
   a. Describe the architecture of a network that reads a sentence and generates an image based on the text. Do not address the question of how such a network is trained, just explain why it should have the capacity to perform this task. You can assume that the images come from a restricted class of images, e.g., faces, and can be encoded (and decoded) in a low-dimensional latent space.

   b. Assume the image is encoded using 4 latent codes that correspond to its four quadrants. Explain how an attention layer can be used to allow such a network to better support fine-grained descriptions in the input text, as well as references to different regions (top, bottom, left, right, sky, ground, etc.). What would be the queries, keys, and values in this case?

א. הרשת המקבלת משפט ומייצרת תמונה מאותו משפט - נוכל להגדיר ארכיטקטורה של רשת כזו באופן הבא:

- מפני שהקלט הוא מסוג "משפט", אורכו אינו ידוע לנו. לכן, נרצה להשתמש בארכיטקטורה מסוג $Many\ to\ One$ (נתון כי מאגר התמונות זמין וסוג התמונות ידוע לנו) ולשם כך נשתמש ברשת מסוג GRU או $LSTM$, כאשר נדאג שההקלט יעבור $Embedding$ לפני שיכנס אליה. בסוף שלב זה, לאחר שהקלט יעבור בכל ה- $Hidden\ Layers$, נקבל $Low\ Dimensonal\ Latent\ Vector$.

  - כעת, נעביר את הפלט שקיבלנו למודל מסוג $Self\ Attention$ שיפלוט וקטור המייצג יחס בין מילים שונות בטקסט.
  - אחרי שקיבלנו את הקידוד המתאים לקלט הטקסט נרצה להטיל אותו למרחב התמונות. נעשה זאת באמצעות רשת מסוג MLP שתבצע את ההטלה למרחב הלטנטי של התמונות. בנקודה זו אנו מסיימים את תהליך ה- $Encoding$.
  - לסיום הוקטור הסופי שנקבל יעבור ל- $Convolutinal\ Decoder$ שיתורגם לתמונה המתאימה. נוכל להשתמש במקום ה- $Convolutinal\ Decoder$ בשיטות כמו $average\ pooling$ בשביל למצוא את קידוד התמונה (מהמאגר הנתון) הקרוב ביותר ללפי ממוצע הוקטורים.

ב. התבקשנו להשתמש בשכבות $Attention$ כדי לספק תיאור **מקומי** על קלט מסוג טקסט בהינתן $4-Latent\ Vectors$ המתאימים לכל רביע בתמונה:

נרצה לממש $Multi-Headed\ Self-Attention$ כאשר מספר ה- $heads$ הוא 4 . משום שמדובר בשכבות $Self\ Attention$ ה- $tokens$ במודל שלנו ייצגו את המילים הנכנסות בקלט (לאחר שעברו $Embedding$ באמצעות $GLOVE$ או $Word2Vec$). כעת, בתהליך הלמידה נחשב את ה- $Loss$ של הרשת לפי כל רביע **בנפרד**. באמצעות למידה בהפרדה "נאלץ" את ה- $heads$ שברשת שלנו ללמוד את הפיטצ'רים הרלוונטים לכל רביע. נציין כי לאחר השימוש בשכבות ה- $Attention$ הפלט ישלח כקלט לשכבות $LSTM$ (בדומה לדוגמה שראינו בהרצאה - $(Lec.5)$"$Image\ Captioning$").

ה־ $queries$ יהיו ה־ $latent\ codes$ המופיעים בשאלה ואילו ה־ $keys$ וה־ $values$ הם חלקי המשפט המתארים חלקים שונים בתמונה.

שאלה 4:

סעיף א:

התמונה שאנו מקבלים בקלט היא בגודל 128 על 128. בתהליך הקונבולוציה התמונה תקטן בכל שלב, כתלות בגודל הפילטר, באופן הבא:

־ (נציין שבכל שלב של קונבולוציה נתון כי $stride = 1$)

קונבולוציה של 3 על 3 תחזיר פלט של 126 על 126 ($128 - 3 + 1$).

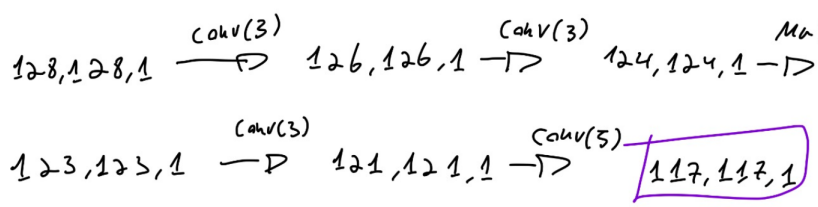
קונבולוציה של 3 על 3 תחזיר פלט של 124 על 124 ($126 - 3 + 1$).

הפעלת $Max - pooling$ כך שה־$sample - rate = 2$ יחזיר פלט של 123 על 123, זאת מכיוון שה־ $stride = 1$.

קונבולוציה של 3 על 3 תחזיר פלט של 121 על 121 ($123 - 3 + 1$).

ולבסוף קונבולוציה של 5 על 5 תחזיר פלט של 117 על 117 ($121 - 5 + 1$).

לכן הפלט של התהליך יהיה 117 על 117



סעיף ב:

חישוב 4.ב:

- ניעזר בנוסחה (לא מצאנו מידע מפורט על כך בתרגול אז חיפשנו באינטרנט: $https://www.baeldung.com/cs/cnn-receptive-field-size$):

$$r_0 = \sum_{l=1}^{L} \left((k_l - 1) \cdot \prod_{i=1}^{l-1} s_i \right) + 1$$

- כאשר k_l מציין את מימד הפילטר , ו־ s_i מציין את ה־ $strides$ שבכל פילטר

- מפני שלפי הגדרה השאלה בכל שכבה $stride = 1$, נוכל להתאים את הנוסחה הנתונה באופן הבא:

$$r_0 = \sum_{l=1}^{L} (k_l - 1) + 1$$

- ניזכר כי:

$$k_1 = k_2 = k_4 = 3$$

- קונבולוציה בהתאם לנתונים בשאלה

$$k_3 = 2$$

- כאשר מדובר בגודל ה־ $maxpooling$
-

$$k_5 = 5$$

- קונבולוציה בהתאם לנתונים בשאלה
- נחשב:

$$r_0 = (3 \cdot (3-1) + (2-1) + (5-1)) + 1$$

$$r_0 = (6 + 1 + 4) + 1$$

$$r_0 = 11 + 1$$

$$r_0 = 12$$

- כלומר, קיבלנו כי גודל ה־ $receptive\ field$ של כל נוירון בשכבה האחרונה הוא 12×12.

סעיף ג:

c. Describe a method to estimate the importance (contribution) of each region in the input image to the final prediction of the network (on that image).

נתאר שיטה לאומדן החשיבות של כל $region$ בקלט התמונה שלנו, אל מול הפרדיקציה הסופית של הרשת על התמונה. למעשה נרצה לראות באילו חלקים של התמונה מופיע מידע חשוב שעוזר לרשת לסווג את התמונה בהצלחה. ניזכר בכך שבסוף הרשת ישנה שכבת $fully\ connected$ אשר תפלוט וקטור באורך כלשהו (נסמנו x'), שכבה זו תסייע בסיווג התמונה על ידי בחירת הקורדינטה בעלת הערך הגבוהה ביותר בוקטור הנ"ל, כלומר הסיווג בעל ה־$score$ הגבוה ביותר מבין האפשרויות לסיווג (הקורדינאטות בוקטור הסופי). לדוגמא, אם המודל שלנו צריך להבחין בין כלב לחתול, הוקטור הסופי יהיה באורך 2 (2 אפשרויות) כך שהקורדינאטה עם הערך הגבוה ביותר תכריע האם מדובר בכלב או בחתול. כעת ניזכר שהוקטור שתיארנו נוצר בשכבת ה־$fully\ connected$ בעזרת כפל במטריצה כלשהי (נניח שמטריצה A כופלת את הוקטור שנכנס ל־$fully\ connected$ שנסמנו ב־x ונקבל $Ax = x'$). כעת למדנו בלינארית שהערך הגבוהה ביותר בוקטור x' (לצורך העניין, אשר שוכן בקורדינאטנה i בוקטור הנ"ל) מחושב על ידי $\sum_{j=1} A_{ij} \cdot x_j = x'_i$. כלומר נוכל לזהות אילו קורדינאטות במטריצה השפיעו על מתן ה־$score$ הגבוהה ביותר בוקטור x'. בזכות זאת נוכל לזהות את ה־$features\ vectors$ המשמעותיים ביותר ברשת, וכך בתחקור לאחור נוכל לזהות את החלקים ברשת שהובילו את הרשת לסווג את התמונה הנבחרת. כעת נשתמש ב־$receptive\ field$ כדי להבין באילו חלקים של התמונה הסיווג של הרשת "נעזר".