מכון טכנולוגי חולון
Holon Institute of Technology

למידה עמוקה ליישומי ראייה ממוחשבת

מטלה 1

אופיר גוריאל, ת.ז. 322975871
אור כהן, ת.ז. 325046969

מספר קורס:
51283
ימי שישי , 11:00-15:00

סמסטר א תשפ"ב – 2022

סעיפים א-ב

בסעיף זה ניצור רשת נוירונים עמוקה – DNN המסווגת את המאגר שלנו באופן בינארית (חולה בדלקת ריאות או בריא) בצורה המיטבית.

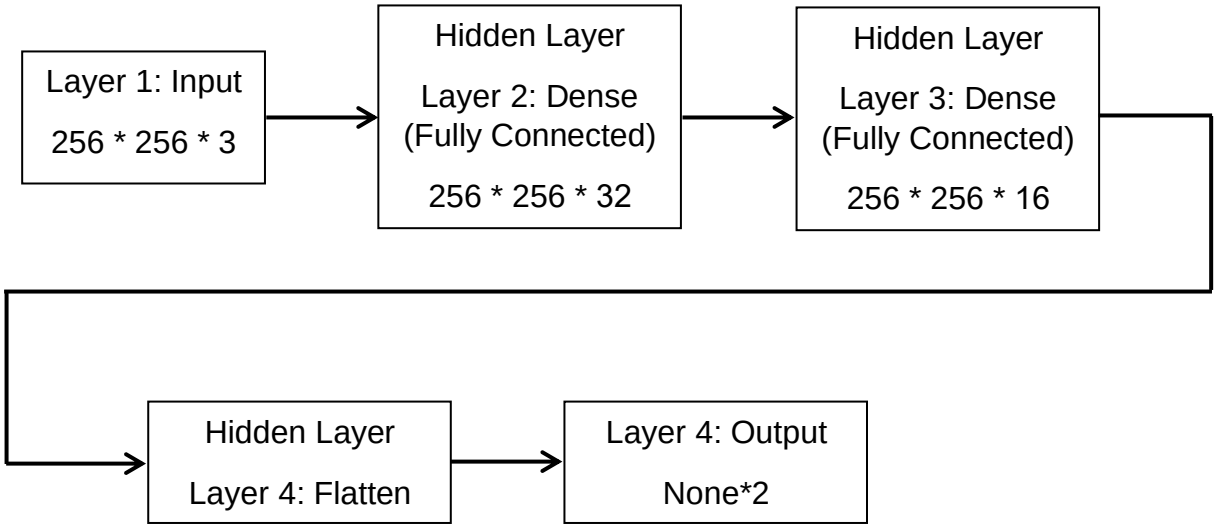
מאגר הנתונים שעליו נאמן את המודל הוא מקבץ של צילומי ריאות, שבו יש סה"כ 1583 תמונות אצל אנשים בריאים, ו 4273 תמונות של אנשים החולים בדלקת ריאות. את המאגר הזה נחלק לשלושת החלקים שאיתם נעבוד במהלך אימון הרשת:

- ב – 70% מהמאגר נשתמש בתור קבוצת ה train.
- ב – 15% מהמאגר נשתמש בתור קבוצת ה validation.
- ב – 15% מהמאגר נשתמש בתור קבוצת ה test.

בכל קבוצה תהיה החלוקה ביחס המתאים של חולים ובריאים, ואת החלוקה ביצענו בעצמנו. למעשה, סט הנתונים בא ממוין ל 3 הקבוצות הללו, אך לא ביחס שאנחנו רוצים, ולכן איחדנו את כל תמונות הריאות של האנשים הבריאים ואז פיצלנו אותם לפי היחס שאנחנו רוצים, ודבר זהה עשינו לקבוצת התמונות של האנשים החולים.

בחלק הראשון בנינו רשת מסוג fully connected, כאשר התמונות בשכבת הקלט הן בגודל של 256*256, אותן נקלוט ב RGB (למרות שבמקור אלו הן תמונות gray-scale).

להלן הארכיטקטורה בה השתמשנו:

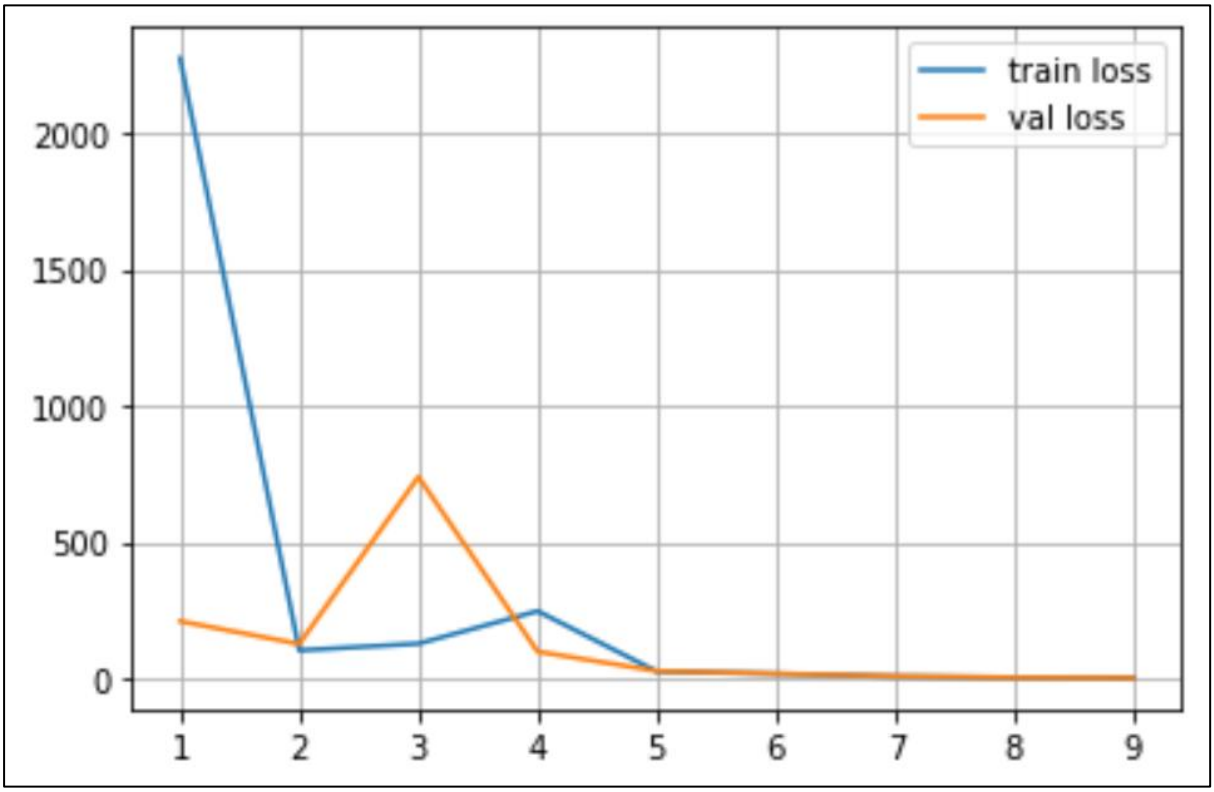


סה"כ נקבל במודל זה למעלה מ 2 מיליון פרמטרים לאופטימיזציה:

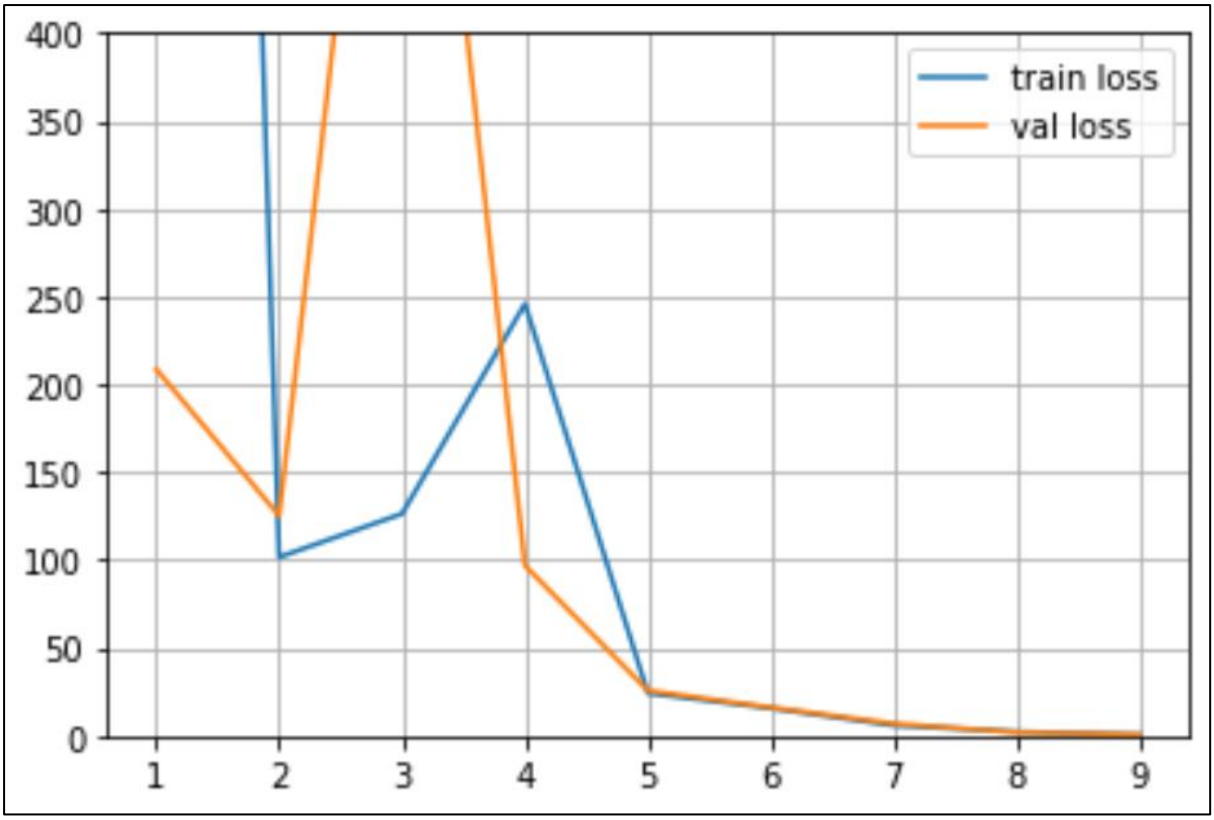
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 256, 256, 32)      128

 dense_4 (Dense)             (None, 256, 256, 16)      528

 flatten_1 (Flatten)         (None, 1048576)           0

 dense_5 (Dense)             (None, 2)                 2097154

=================================================================
Total params: 2,097,810
Trainable params: 2,097,810
Non-trainable params: 0
_________________________________________________________________
```

נפעיל את אימון המודל ונקבל את הגרפים הבאים:

ראשית עבור פונקציות המחיר:

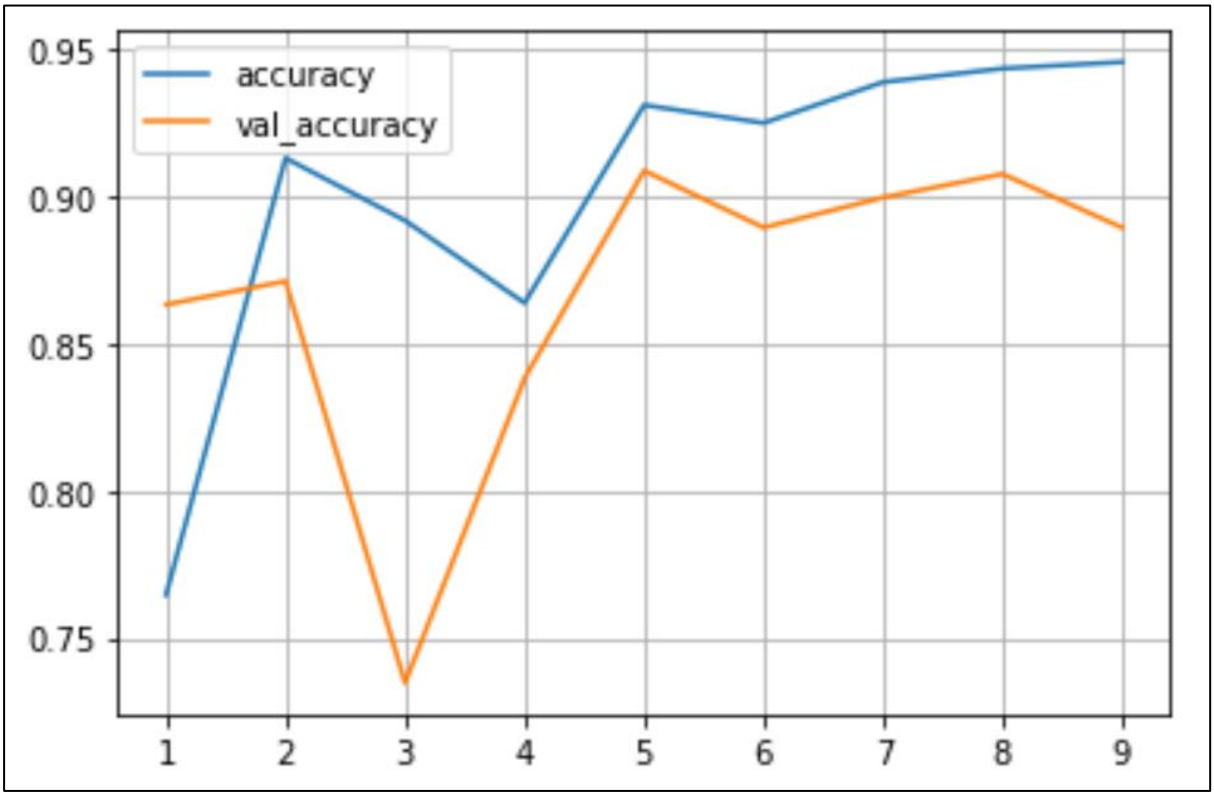


נתקרב סביב הערכים הקטנים יותר:



ניתן לראות שקיימת ירידה עקבית בפונקציית המחיר לאחר האפוק הרביעי, גם ה loss שמחושב מקבוצת האימון וגם זה שמחושב מקבוצת ה validation יורדים.

נוכל לראות תופעה דומה גם בגרפי הדיוק של הרשת:



בסופו של דבר אנחנו רואים שיפור ברמת הדיוק גם בקבוצת האימון וגם בקבוצת הוולדיציה, כאשר באפוק האחרון המודל מתכנס וכך גם רמת הדיוק שלו, מה שמונע עלייה נוספת של רמות הדיוק (ואף מעט ירידה בדיוק הנמדד ב validation).

לבסוף, נבדוק את הדיוק של המודל על קבוצת ה test, ונקבל:

```
loss, accuracy = model.evaluate(TestData)
print("Loss :", loss)
print("Accuracy :", accuracy)
```

```
1/1 [==============================] - 16s 16s/step - loss: 0.5481 - accuracy: 0.9101
Loss : 0.5481081604957581
Accuracy : 0.9101251363754272
```

כלומר קיבלנו כ 91% דיוק עבור סט התמונות שמיועד למבחן, שנחשב גבוה ביחס לארכיטקטורה פשוטה יחסית. עם זאת, נזכור שעשינו אופטימיזציה על למעלה מ 2 מיליון פרמטרים, ולכן זה הגיוני שקיבלנו תמורה לכך בדיוק גבוה במיוחד.

לבסוף, נציג את הדיוק של הרשת גם בעזרת המטריקות המקובלות:

```
Precision: 0.9636963696369637
Recall: 0.9110764430577223
confusion matrix: tf.Tensor(
[[584  57]
 [ 22 216]], shape=(2, 2), dtype=int32)
```

כלומר עבור 879 תמונות שנמצאות בקבוצת ה test, נקבל את הסיווג הבא:

	Predicted Label		
Actual Label		Positive	Negative
	Positive (Pneumonia)	584 True Positive	57 False Negative
	Negative (Normal)	22 False Positive	216 True Negative

נשים לב שקיבלנו שיעור recall גבוה במיוחד השווה בערך ל 91%. נזכיר ש recall מחושב באופן הבא:

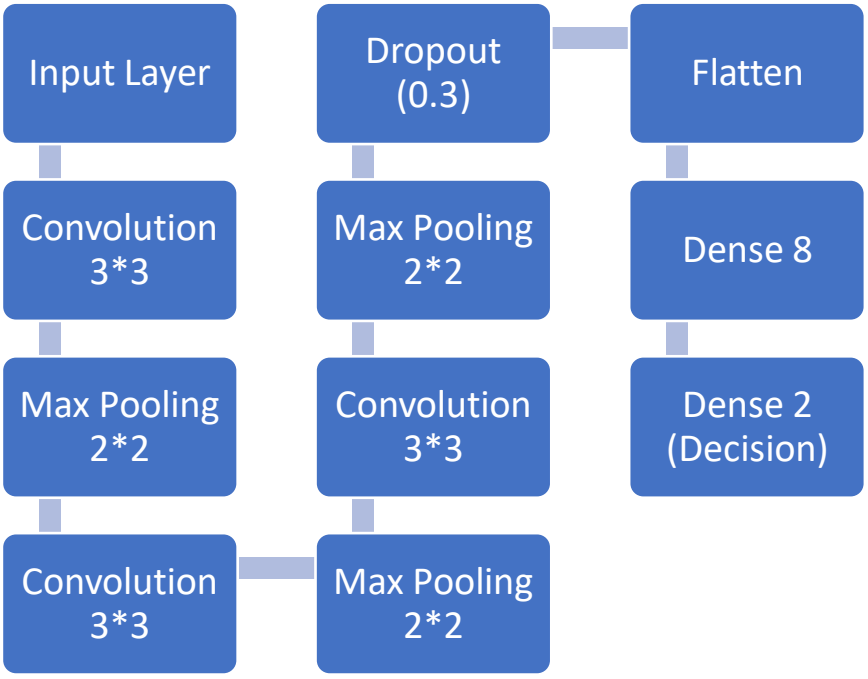
$$Recall = \frac{\#\ True\ Positive}{\#\ True\ Positive + \#\ False\ Negative}$$

כלומר עצם העובדה שקיבלנו recall גבוה אומר לנו שקיים סיכוי נמוך במיוחד להגיע למצב של false negative, וזה לא סביר שהרשת תיתן תשובה שלילית עבור בנאדם בעל דלקת ריאות. בניסוי זה אנחנו משתמשים ברשת לצרכים רפואיים, ולכן תהיה חשיבות הרבה יותר משמעותית למנוע false negative מאשר למנוע false positive, מה שאומר שקבלת recall גבוה הרבה יותר חשוב לנו מאשר קבלת precision גבוה, ובמקרה של הרשת הזו ניתן לומר שעמדנו ביעדנו.

סעיפים ג-ד

כעת, נחליף את המודל לכזה הבנוי משכבות קונבולוציה במקום שכבות fully connected, להלן תיאור של המודל:

```
model = tf.keras.Sequential([
        tf.keras.layers.Input(shape=(256,256,3)),
        tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Conv2D(128, kernel_size=(3, 3), activation="relu"),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(16),
        tf.keras.layers.Dense(2, activation="softmax"),
])

model.compile(optimizer="Adam",
              loss="categorical_crossentropy",
              metrics=['accuracy'])
```



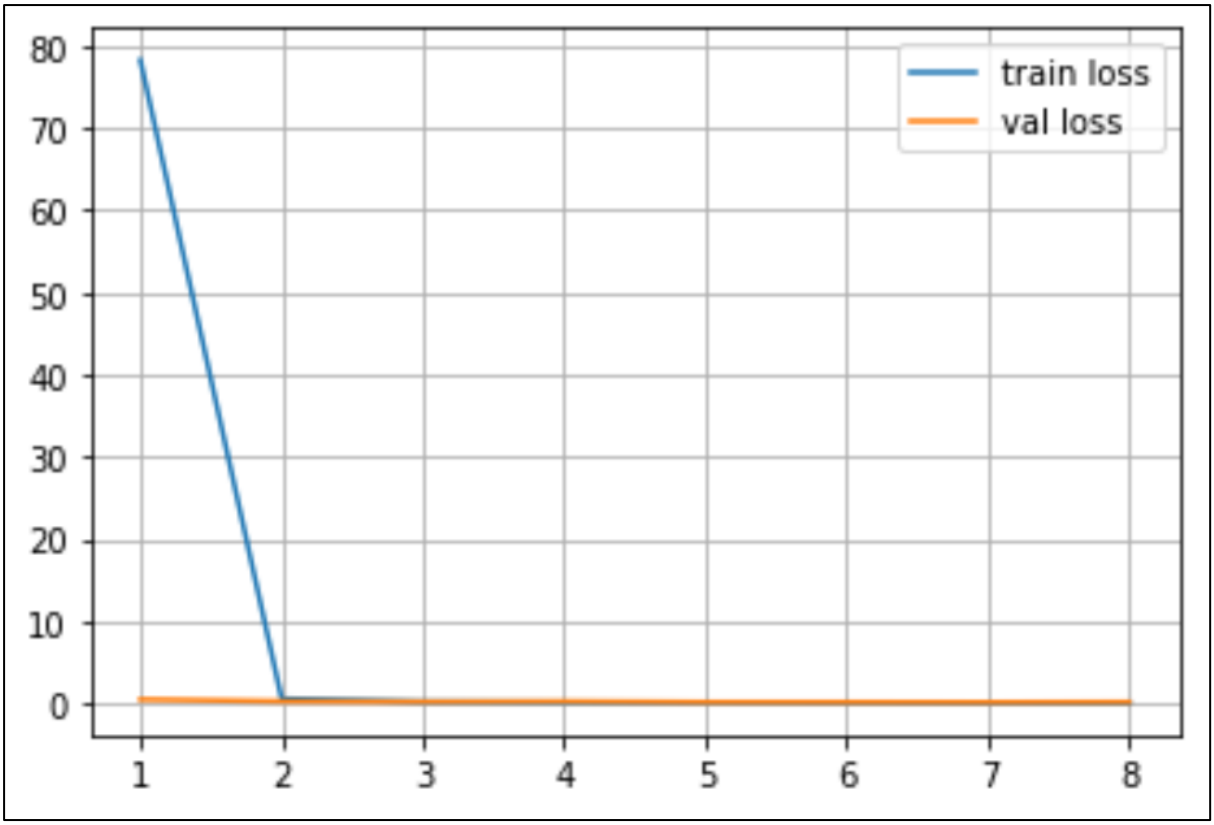
כלומר בנינו 3 זוגות של ביצוע קונבולוציה (בסדר עולה בכמות הפילטרים בכל שכבה, כלומר 32, 64 ואז 128), ולאחר מכן ביצוע max pooling (שהרי ביחד היעילות של כל שכבה גדלה). לבסוף נוסיף

שכבת Flatten, שכבת fully connected אחת של 8 נוירונים (על מנת להפריד בין ה flatten להחלטה הסופית) ושכבת החלטה בינארית. נקבל אם כן את סיכום המערכת:

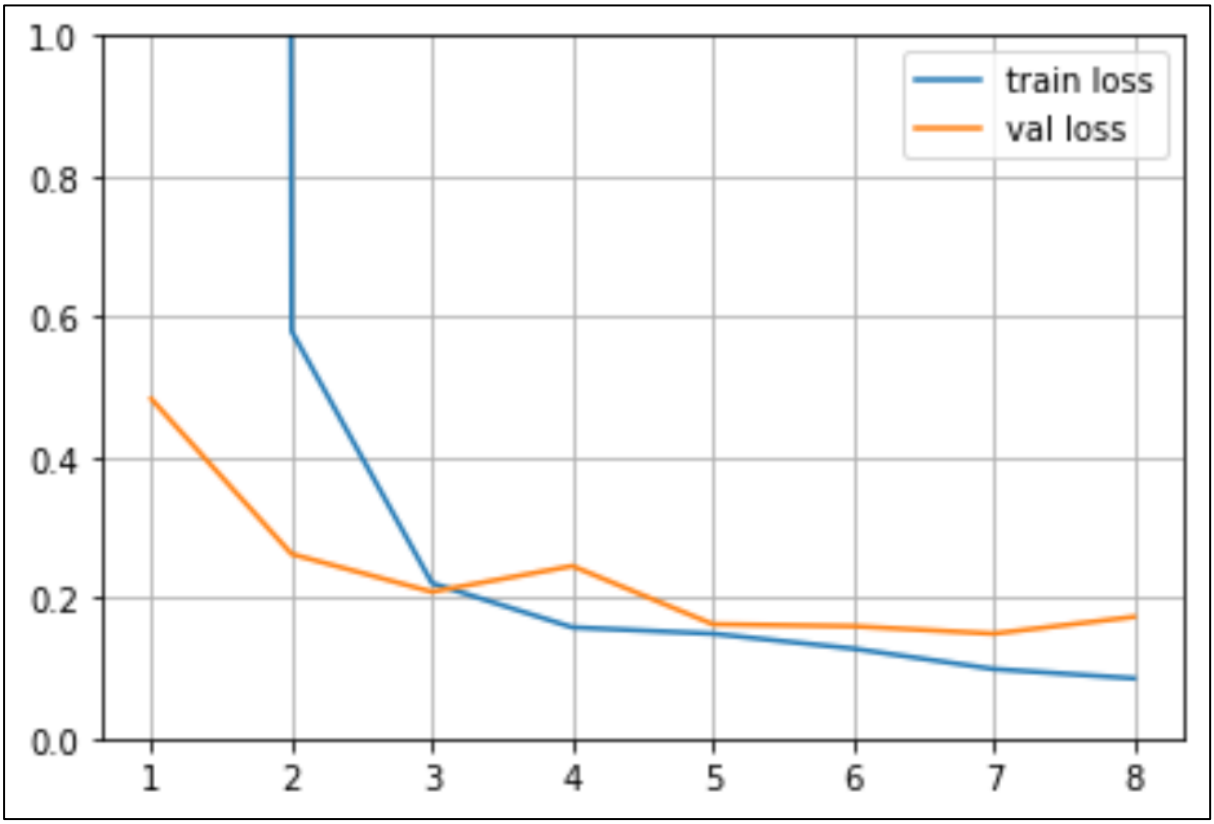
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 62, 62, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 60, 60, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 30, 30, 128)      0
 2D)

 dropout (Dropout)           (None, 30, 30, 128)       0

 flatten (Flatten)           (None, 115200)            0

 dense (Dense)               (None, 8)                 921608

 dense_1 (Dense)             (None, 2)                 18

=================================================================
Total params: 1,014,874
Trainable params: 1,014,874
Non-trainable params: 0
_________________________________________________________________
```

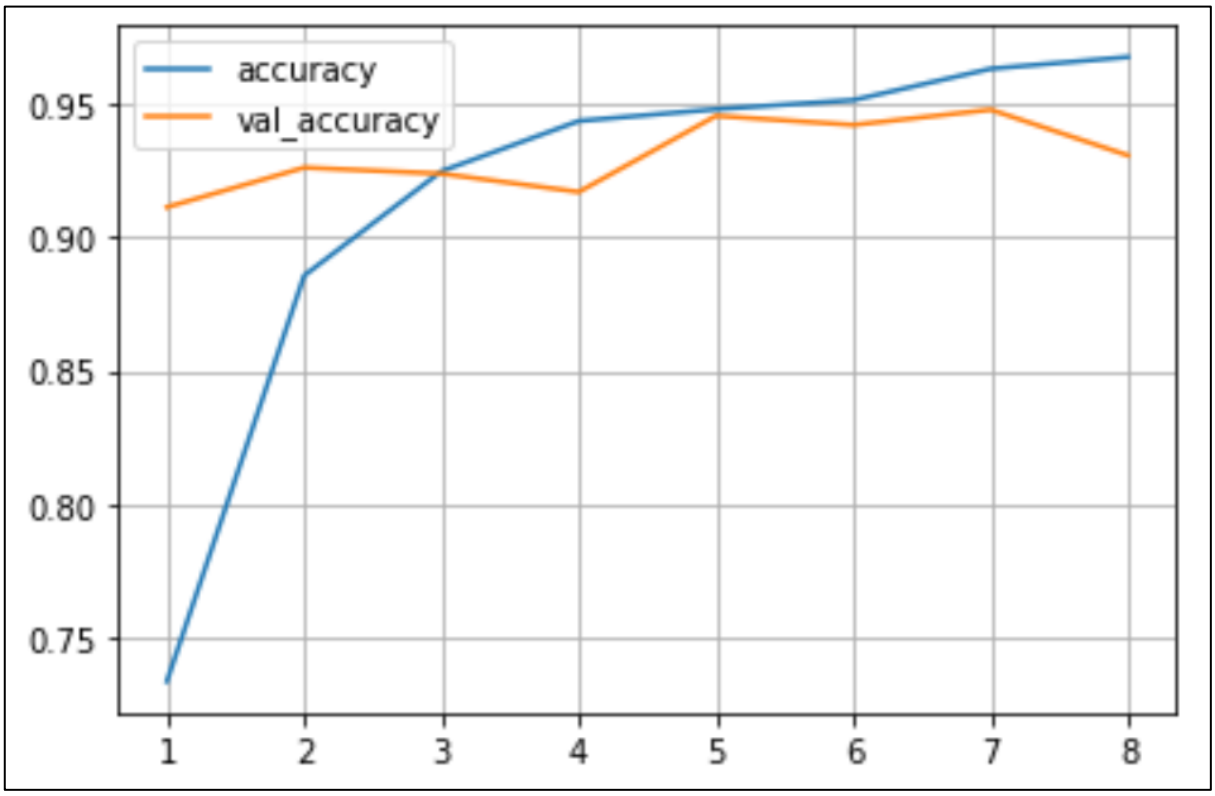
נפעיל אותה ונקבל את גרפי הלמידה הבאים:



נסתכל בקנה מידה קטן יותר על מנת לראות את השינויים ב validation loss:



בנוסף, נסתכל על גרפי הדיוק של הרשת:



ניתן לראות למידה עקבית וטובה שמתחילה מהאפוק השני , ואחריו גם גרפי הדיוק וגם פונקציות המחיר מתייצבות סביב ערך קבוע. מהעובדה שגם פה עשינו שימוש במספר גדול במיוחד של פרמטרים לאופטימיזציה, ניתן להסיק שיכול להיות שנכנסנו למצב של over fitting לאחר האפוק השמיני, אך דבר זה נוכל לראות רק לאחר בדיקת המודל על סט המבחן.

```
loss, accuracy = model.evaluate(TestData)
print("Loss :", loss)
print("Accuracy :", accuracy)
```

```
1/1 [==============================] - 46s 46s/step - loss: 0.1558 - accuracy: 0.9579
Loss : 0.15583935379981995
Accuracy : 0.9579067230224609
```

ניתן לראות שערך הדיוק שקיבלנו בקבוצת המבחן הוא 95.8%, שזה קצת יותר מהגובה סביבו התנדנד הדיוק של סט ה validation לקראת האפוק האחרון. לכן, נוכל להשיג שתופעת ה over fitting אולי קרתה, אך זה גרם לגודל ה accuracy להישאר קבוע יחסית ולא לרדת (כלומר הגענו ליציבות סביב נקודת מינימום כלשהי אך לא עלינו לערך גבוה בהרבה מהמינימום).

נבחן כעת את המודל גם בעזרת המטריקות המקובלות:

```
Precision: 0.9467455621301775
Recall: 0.9984399375975039
confusion matrix: tf.Tensor(
[[640   1]
 [ 36 202]], shape=(2, 2), dtype=int32)
```

כלומר עבור 879 תמונות שנמצאות בקבוצת ה test, נקבל את הסיווג הבא:

	Predicted Label		
Actual Label		Positive	Negative
	Positive (Pneumonia)	640 True Positive	1 False Negative
	Negative (Normal)	36 False Positive	202 True Negative

נשים לב שקיבלנו שיעור recall מדהים השווה ל 99.85%, כלומר מתוך כל קבוצת הנסיינים שלנו (בסט המבחן) המכילה 879 אנשים, רק מטופל אחד אינו יאובחן בדלקת ריאות למרות שהוא חולה בה. יתרה מכך, השיפור בביצועי ה recall בא על חשבון ירידה ב precision.

לסיכום, קיבלנו שרשת הקונבולוציה מציגה ביצועים טובים יותר בצורה משמעותית גם ברמת הדיוק וגם ברמת ה recall ביחס לרשת ה fully connected הרגילה, וזאת תוך כדי ביצוע אופטימיזציה על מחצית מכמות הפרמטרים (ב CNN יש כמיליון פרמטרים כאשר ב fully connected יש כ-2 מיליון פרמטרים שעליהם מבצעים את האופטימיזציה).

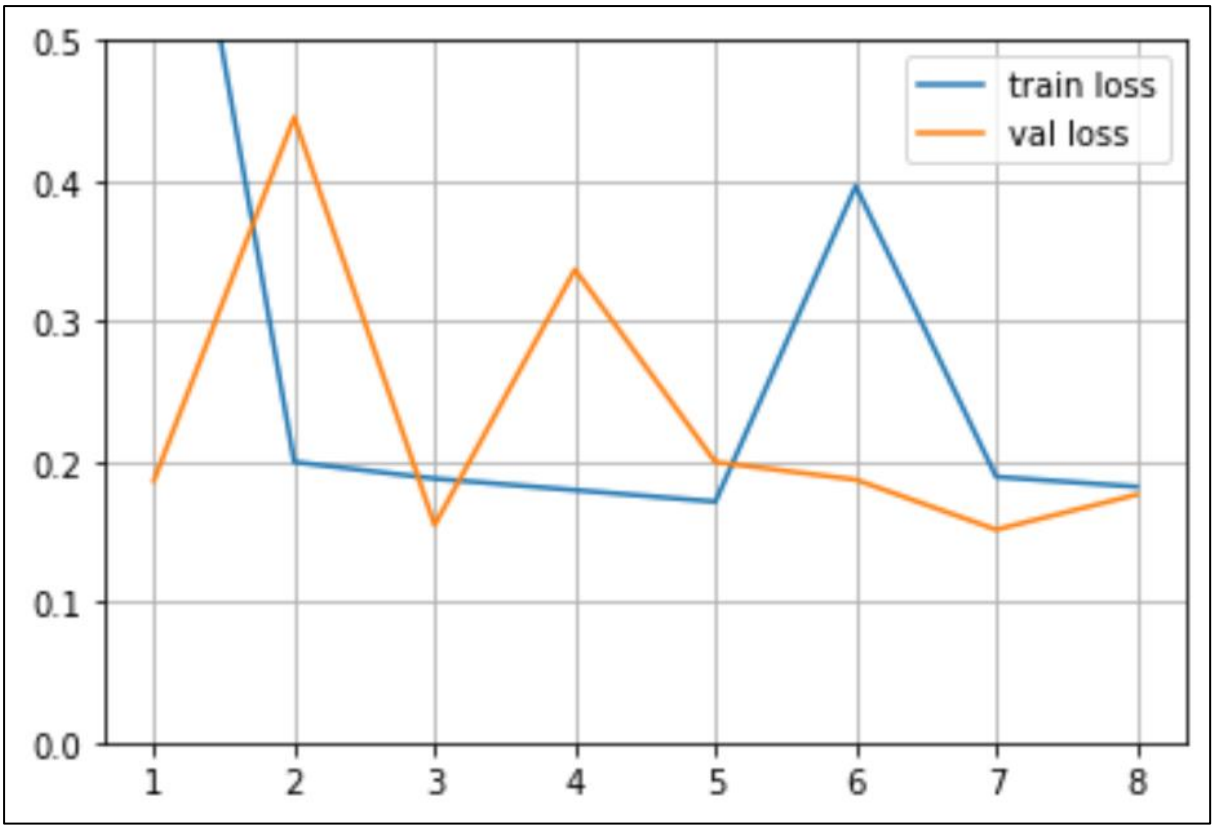
סעיף ה

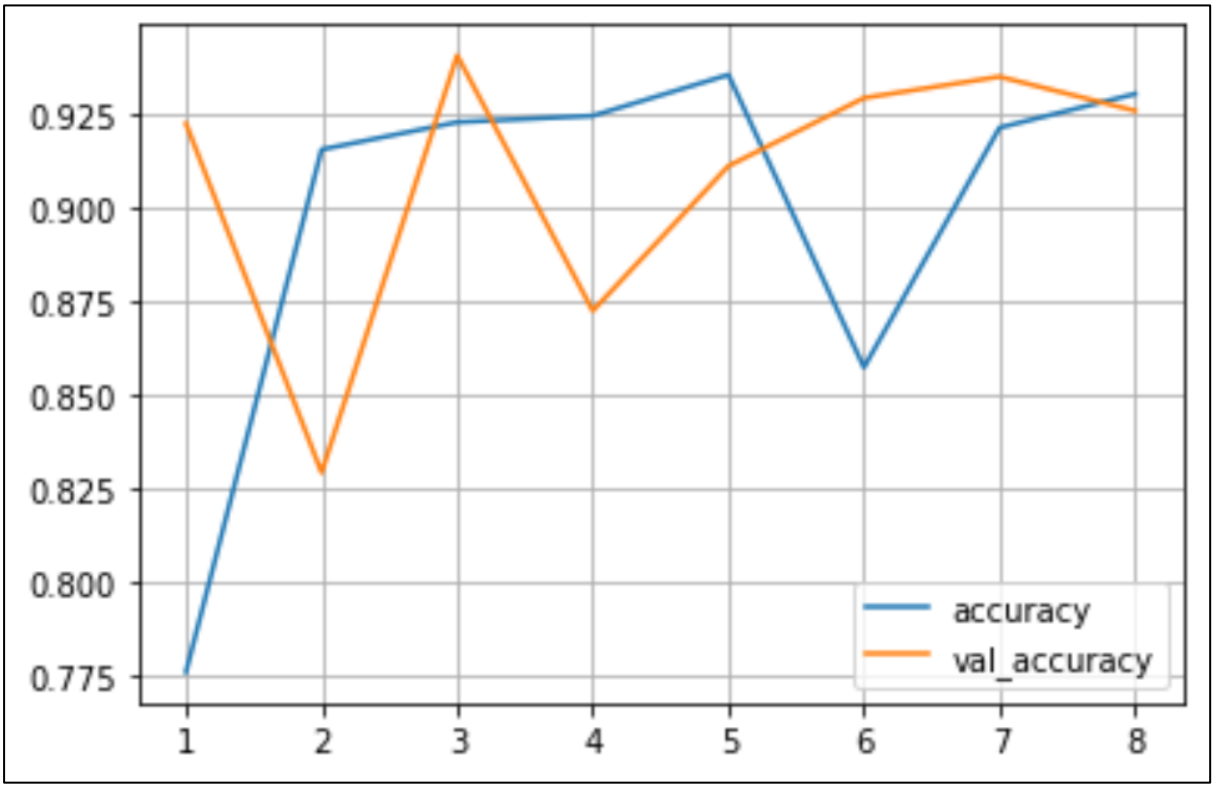
נבדוק כעת את ההשפעה של פונקציית האקטיבציה. נשנה אותה מ relu ל tanh בשתי בשכבות הראשונות, אך כדי למנוע מוות מסיבי של נוירונים (מאזורי הרוויה של פונקציית tanh) נשאיר את פונקציית האקטיבציה הקודמת בשכבה האחרונה. להלן סיכום הארכיטקטורה כפי שהיא כעת:

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 254, 254, 32)      896

 max_pooling2d_3 (MaxPooling  (None, 127, 127, 32)     0
 2D)

 dropout_1 (Dropout)         (None, 127, 127, 32)      0

 conv2d_4 (Conv2D)           (None, 125, 125, 64)      18496

 max_pooling2d_4 (MaxPooling  (None, 62, 62, 64)       0
 2D)

 dropout_2 (Dropout)         (None, 62, 62, 64)        0

 conv2d_5 (Conv2D)           (None, 60, 60, 128)       73856

 max_pooling2d_5 (MaxPooling  (None, 30, 30, 128)      0
 2D)

 dropout_3 (Dropout)         (None, 30, 30, 128)       0

 flatten_1 (Flatten)         (None, 115200)            0

 dense_2 (Dense)             (None, 8)                 921608

 dense_3 (Dense)             (None, 2)                 18

=================================================================
Total params: 1,014,874
Trainable params: 1,014,874
Non-trainable params: 0
_________________________________________________________________
```

נפעיל את אימון הרשת ונקבל את הגרפים הבאים:


0.5
0.4
0.3
0.2
0.1
0.0
1
2
3
4
5
6
7
8
train loss
val loss


0.925
0.900
0.875
0.850
0.825
0.800
0.775
1
2
3
4
5
6
7
8
accuracy
val_accuracy

ניתן לראות שהמערכת מתקשה ללמוד, והתהליך של הלמידה הרבה פחות יציב מאשר בפעם הקודמת. יתרה מכך, ניתן לראות תהליך של ירידה עקבית ב validation loss (ואיתו עלייה ב validation accuracy) רק באפוקים 4-7. כפי שלמדנו בכיתה, פונקציית האקטיבציה המקובלת היום היא relu, ובשקלול כל אלה אנחנו יכולים לצפות שסה"כ רמת הדיוק של המודל תרד.

להלן התוצאות מהפעלת המודל על סט המבחן:

```
loss, accuracy = model.evaluate(TestData)
print("Loss :", loss)
print("Accuracy :", accuracy)
```

```
1/1 [==============================] - 47s 47s/step - loss: 0.1442 - accuracy: 0.9511
Loss : 0.14418593049049377
Accuracy : 0.9510807991027832
```

ועם המטריקות המקובלות:

```
Precision: 0.9779179810725552
Recall: 0.9672386895475819
confusion matrix: tf.Tensor(
[[620  21]
 [ 14 224]], shape=(2, 2), dtype=int32)
```

כלומר עבור 879 תמונות שנמצאות בקבוצת ה test, נקבל את הסיווג הבא:

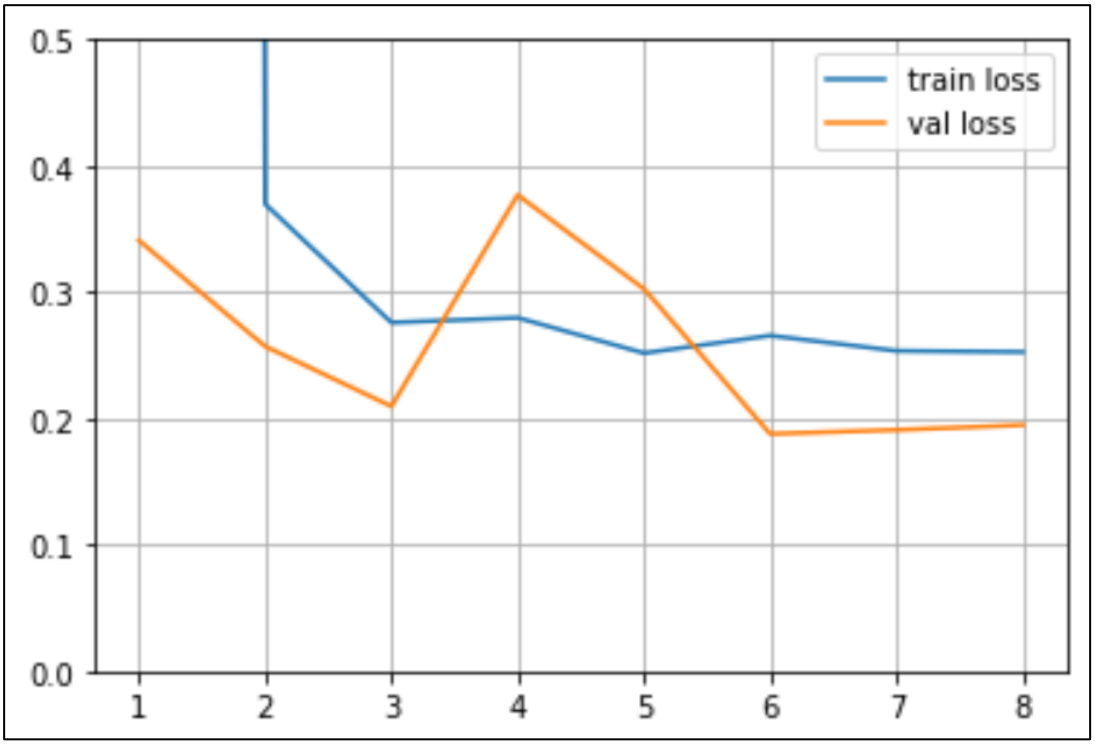
	Predicted Label		
Actual Label		Positive	Negative
	Positive (Pneumonia)	620 True Positive	21 False Negative
	Negative (Normal)	14 False Positive	224 True Negative

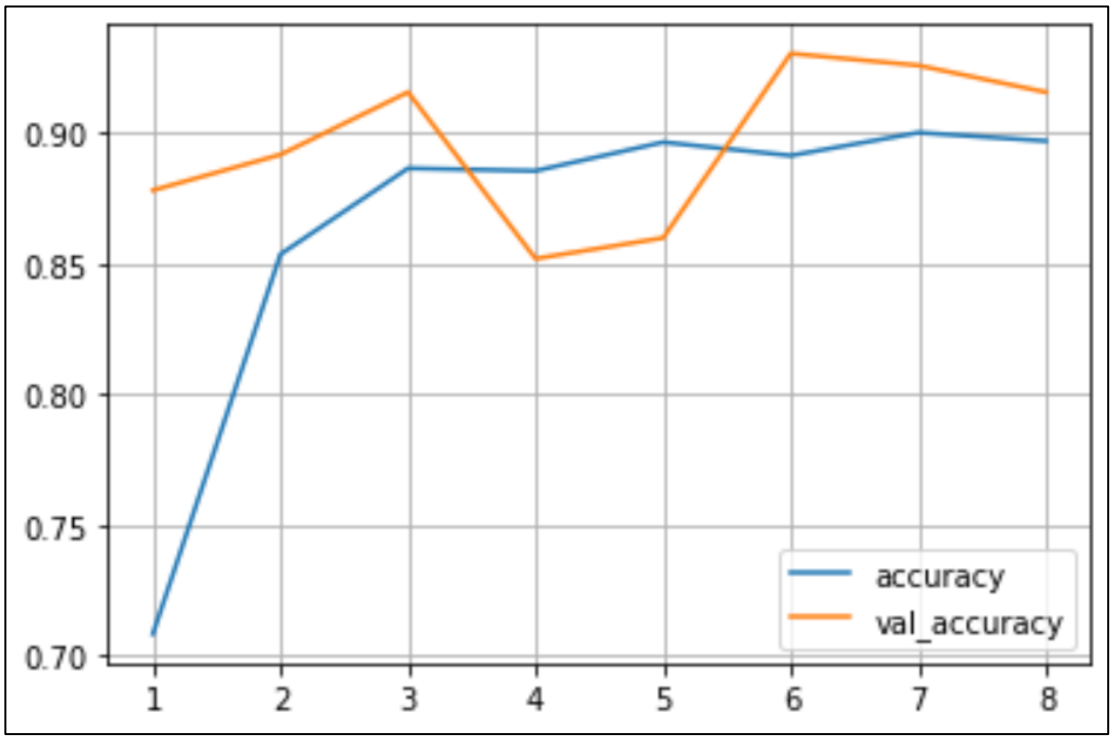
נשים לב שקיבלנו שיעור recall נמוך יותר השווה ל 96.7%, כלומר מתוך כל קבוצת הנסיינים שלנו (בסט המבחן) המכילה 879 אנשים, 21 מטופלים אינם יאובחנו בדלקת ריאות למרות שהם חולים בה. יתרה מכך, הירידה בביצועי ה recall באה על חשבון עלייה ב precision, שזה מעבר בו אנו לא מעוניינים במערכות רפואיות.

בסופו של דבר נוכל לסכם ששימוש בפונקציית אקטיבציה של tanh גרמה לירידה בביצועים גם ברמת ה accuracy הכללי שנמדד על קבוצת המבחן, וגם ברמת ה recall שירד, ואילו היינו מגדירים את כל השכבות עם tanh סביר להניח שהירידה בביצועים אפילו הייתה יותר משמעותית. תופעה זו נובעת מכיוון שלפונקציה זו טווח רחב של ערכים בהם הנגזרת שלה אפס והיא מגיעה לרוויה, אשר גורמת לגרדיאנט להתאפס ואיתו להפסקת ההשפעה של נוירונים מסוימים ברשת. השימוש בפונקציית relu לחלוטין עדיפה במקרה זה.

סעיף ו

בסעיף זה נבדוק את ההשפעה של ביצוע אוגמנטציות על התמונות לפני אימון הרשת. האוגמנטציות שאותן נפעיל הן היפוך אנכי ואופקי באופן אקראי, כמו גם סיבוב אקראי שמוגבל ע"י חסם עליון ותחתון של ±0.2 רדיאנים (שהם כ 11 מעלות). נאמן את המודל כאשר הפעלנו את הפעולות הללו על סט המבחן וסט הולידציה, כאשר את המודל עצמו נחזיר ל CNN המקורי (כלומר כאשר כל פונקציות האקטיבציה הן relu). נקבל את התוצאות הבאות:





נפעיל את המודל על סט המבחן ונקבל:

```
loss, accuracy = model.evaluate(TestData)
print("Loss :", loss)
print("Accuracy :", accuracy)
```

```
1/1 [==============================] - 74s 74s/step - loss: 0.2082 - accuracy: 0.9170
Loss : 0.20821347832679749
Accuracy : 0.916951060295105
```

```
Precision: 0.945141065830721
Recall: 0.9407176287051482
confusion matrix: tf.Tensor(
[[603  38]
 [ 35 203]], shape=(2, 2), dtype=int32)
```

לסיכום, קיבלנו ירידה בביצועי הרשת בעקבות הוספת שכבות האוגמנטציה. ירידה זו באה לידי ביטוי בלמידה פחות יציבה ורציפה (שגרמה לירידה בדיוק בין אפוקים 3 ל 6), כמו גם לקבלת דיוק סופי נמוך יותר בקבוצת המבחן. יתרה מכך, זה גם לא עזר לשפר את מטריקת ה recall, שקטנה ל 94%.

לינק לקוד ב GitHub:

https://github.com/OfirGuriel/DeepLearning1.git