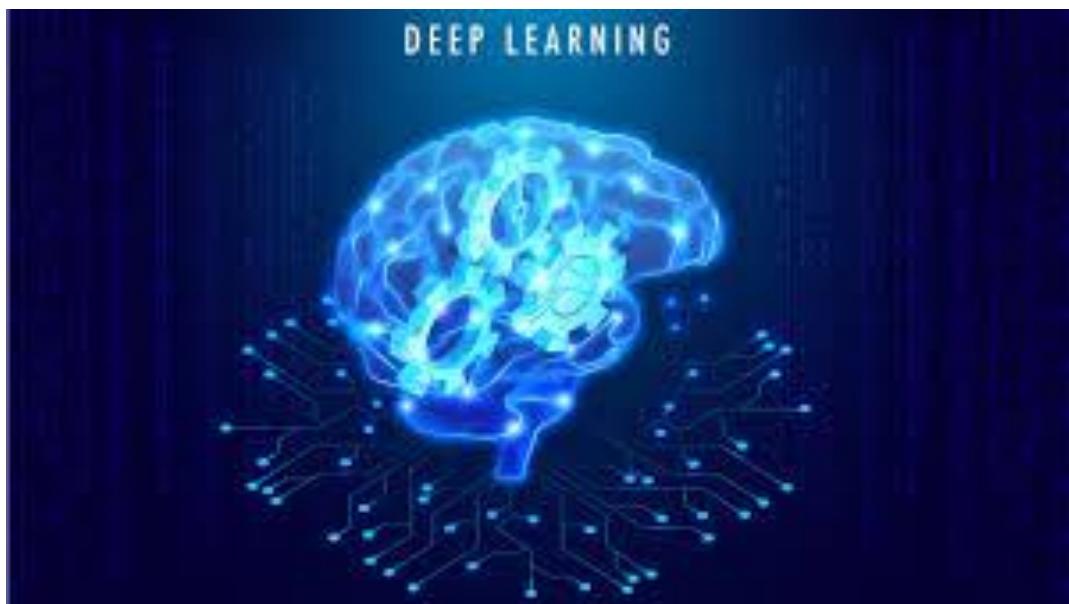


**דו"ח פרויקט סוף קורס מערכות לומדות ולמידה عمוקה  
גילוי דלקת ריאות (פנאומוניה) מצילומי רנטגן**



מוגש ע"י: יובל המר, 209158518  
חי כספי, 213038060  
רועי מרקוביץ', 209000355

בתהנחיית: פרופ' אמיר אדלר

תאריך: 12/05/24

## תוכן עניינים

3.....	מבוא : .....
4.....	דרישות הפרויקט : .....
4.....	משימות : .....
4.....	משימה I : .....
4.....	משימה II : .....
5.....	משימה III : .....
5.....	משימה IV : .....
6.....	פתרונות : .....
6.....	רשות נוירונים קובולוציונית (CNN) : .....
7.....	פתרון משימה I : .....
10.....	פתרון משימה II : .....
13.....	פתרון משימה III : .....
31.....	פתרון משימה IV : .....
42.....	רשת Transfer Learning .....
42.....	פתרון משימה I : .....
44.....	פתרון משימה II : .....
46.....	פתרון משימה III : .....
52.....	נספחים : .....

## מבוא:

דלקת ריאות (באנגלית: Pneumonia, פנאומוניה) היא זיהום של רקמת הריאה, ככלומר של שקייקי האויר החזירים שנמצאים בריאות. בשקייקים נקלט החמצן מהאויר במסגרת תהליכי הנשימה, ובhem מועבר החמצן לזרם הדם. זיהום בשקייקי האויר אלה פוגע ביכולת הגוף להעביר חמצן לאיברים השונים.

דלקת ריאות גורמים חידקיים או וירוסיים (נגיפים). לעיתים נדירות - בעיקר בחולים שסובלים מליקויים במערכת החיסון - עלולות גם פטריות לגרום לדלקת ריאות. כמו כן לעיתים רחוקות עלולה מחלת נגיפית כמו שפעת לגרום להמשך לדלקת ריאות חידקית.

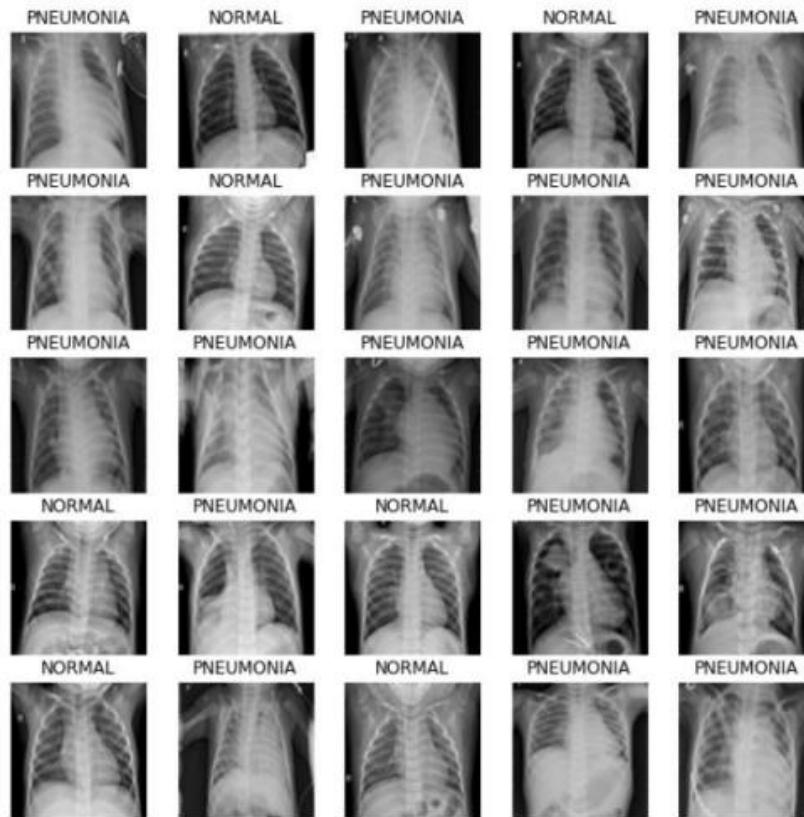
התסמינים של דלקת ריאות כוללות בדר' כחום, צמרמורות, שיעול, קוצר נשימה ואף כאבים בחזה. אבחון הדלקת מבוסס על ידי ביצוע צילום רנטגן (X-ray) של בית החזה ובדיקות דם.

בחודשי החורף (ובמיוחד בתקופת המחלת הקורונה) ישנה עלייה גבוהה בחולים ולכן קיים הצורך בפתרון מהיר ומדויק לפיענוח של תצלומי הרנטגן.

פרויקט זה מציג פתרון לפיענוח התצלומים בעזרת טכנולוגיית למידה عمוקה (Deep Learning). פתרון זה מביא לתוצאות כמעט אמת בזמן קצר של מעל 90% ובכך יסייע להורדת העומס של הרדיולוגים המפענחים המוני תצלומים של מטופלים.

המערכת "تلמד ותנתה" אלףים רבים של צילומי רנטגן ממאגר צילומי חזות של אנשים בריאים וחולים הקיימים מראש (קבוצת האימון, Training Set).

בעזרת אלגוריתם CNN (Convolutional Neural Network) נדע לסוג צילומי רנטגן חדשים שהאלגוריתם נתקל בהם (קבוצת הבדיקה, Test Set) ולאחר מכן ישנה של דלקת ריאות בצילום.



## דרישות הפרויקט:

- בפרויקט זה علينا למשתמש בתמונות מבוססות CNN לגילוי דלקת ריאות מלאפי צילומים רנטגן אמיטיים הכוללים אבחון של רפואיים מומחיהם בנושא. את התמונות ניקח מהאתר הבא : [תמונה](#)
- בנוסף, קיימת הפרדה בין סוג דלקת ריאות: חיידקית (Bacterial) ונגיפית (Viral).
- ביצוע הרשת ימודדו על סט תמונות בדיקה (Test Set) אשר יהיה מופרד באופן מוחלט מסט האימון (Train Set), ככלור קבוצות האימון והבדיקה יהיו זרות.
- ביצוע רשת טובים יחשבו לטכני שగיאה הנמוק מ-7% (ככלור אחוז דיוק גבוה מ-93%).  
סט תמונות הבדיקה יכול לפחות 200 תמונות עם דלקת ריאות (100 תמונות מכל סוג) ו-200 תמונות ללא דלקת ריאות.

## משימות:

### משימה I :

תכנון שתי רשתות עמוקות לפתרון הבעיה שתוארה :

- רשת CNN מבוססת .Transfer Learning

- רשת CNN ללא שימוש ב-Transfer Learning .

כאשר כניסה כל רשת היא תמונה ומוצאה הינה הסטבות שהתמונה מייצגת מקרה חיובי של דלקת ריאות (לא הבדל בין סוג הדלקת). נציין כי טווח הערכים שמוצא כל רשת (הסתבות) קיבל הוא ערך רציף בטווח [0,1] כאשר 1 מייצג מקרה ודאי של דלקת ריאות.

### משימה II :

הAIMON הרשות ממשימה הראשונה, תוקן ציון אלגוריתם האימון, מספר ה-EPOCHS, Learning Rate וגודל ה-Mini-Batch.

עבור הרשת המבוססת Transfer Learning יש להציג שני אופני אימון :

1. פרמטרי רשת הבסיס מוקפאים והשכבות שהוספנו מתעדכנות בזמן האימון.

2. Fine-Tuning בו מעדכנים חלק משכבות רשת הבסיס.

3. על סמך הדיוק שהתקבל, איזה מן השיטות הניל'ם הביאו לביצועים טובים יותר?

כמו כן, עבור שתי הרשות נשרטט את גרך הביצועים Precision-Recall כאשר כל נקודה בגרף תחוسب עבור רמת סף שונה (ביחס להסתבות שmpsika הרשת). להחלטה על דוגמה חיובית (עם דלקת ריאות) הסף יהיה בטווח של 0.1 עד 0.9 בקפיצות של 0.05.

בנוסף, על הגרך נסמן את נקודות ה-F-Score-Precision-Recall שיחושבו מכל זוג ערכיהם של

יתר על כן, علينا למצוא את ערך הסף עבורו התקבל ערך ה-F-Score הגבוה ביותר.

### משימה III:

נבדוק את ביצועי הרשותות שבנוינו עם אלגוריתמי האימון הבאים תוך שימוש Epochs ו-  
Learning Rate לכל אלגוריתם :

- .1. אלגוריתם SGD.
- .2. אלגוריתם SGD עם Momentum.
- .3. אלגוריתם ADAM.
- .4. אלגוריתם RMSprop.

עבור אלגוריתם האימון שסיפק את תוצאות הדיקט הטובות ביותר, נפעיל את מגנון ה- Early Stopping  
ושרטט גרף התכנסות של תהליכי האימון לכל אחד מאלגוריתמי המימון  
(Train Loss, Validation Loss, Train Accuracy, Validation Accuracy).  
sett הווילדייה יכלול לפחות 50 תמונות עם דלקת ריאות (25 מכל סוג) ו- 50 תמונות ללא דלקת ריאות.  
sett זה יילקח מותוךsett האימון (ובכך יהיו קבוצות זרות של תמונות).

### משימה IV:

نمדל את הרשות מהמשימה הראשונה (רשות לא מבוססת Transfer Learning) כך שתוכל לסוג ל-3 קטגוריות :

- .1. אין דלקת ריאות.
- .2. דלקת ריאות חיידנית.
- .3. דלקת ריאות נגיפית.

עבור רשות זו, נבצע את משימה 3 בשנית ונציג את מטריצת הבלבול (Confusion Matrix) עלsett  
הבדיקה עבור האלגוריתם שהביא את התוצאות הטובות ביותר.

## פתרונות:

### רשת נוירוניים קונבולוציונית (CNN):

יבאנו את כל הנתונים בהם השתמשנו מהאתר וחלקנו את הנתונים לשולש קבוצות – קבוצה אימון, בדיקה ו-וילדציה.

דרישות המשימה, התוצאות הנמצאות קבוצות אלה הם 0 ו-1 בלבד, כך ש- 0 מסמן צילום של אדם בריא ו-1 מסמן צילום של אדם חולה בדלקת ריאות (לא הבדל אם נגיפית או חידקית).

כמו כן, למען ביצועים טובים יותר של המודל, נורמל כל תמונה לצבעי שחור לבן (חלוקת ב-255) ונשנה את גודל התמונה ל-180x180.

לאחר מכן, חילקנו את קבוצות האימון שלנו לפי חלוקה הבאה:

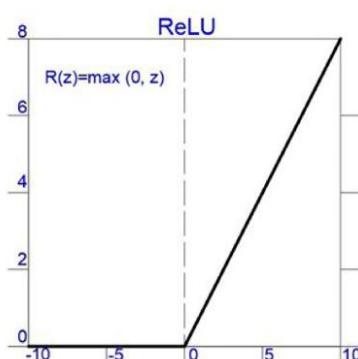
כמות התמונות	אחוז הקבוצה מכל התמונות	
1172	20%	sett validation (Validation)
3519	60%	sett training (Train)
1172	20%	sett test (Test)

את המודל ממשנו בצורה טורית (Sequential).

פירוט השכבות בהם נעשה שימוש במהלך מימוש רשת CNN :

שם השכבה	הסבר פעולה
Conv2D	מבצעת קונבולציה על הקלט בכניסה, בהתאם לפרמטרים שקיבלה (גודל גרעין הקונבולוציה, מספר הצעדים וגודל התמונה במקור)
MaxPooling2D	הקטנת ממדיהם feature-maps (הפחיתה מספר הפרמטרים הנלמדים)
Dropout	מצמצם את הפרמטרים הנלמדים (לפי אחוז הארגומנט שקיבל) באופן רנדומלי, ב כדי למנוע Over-Fitting
Flatten	הופך ווקטור בו ממדי לווקטור חד ממדי לטובת המסוג
Dense	מחבר את כל מוצאי הנוירונים לשכבה הבא (FC)

ברשת זו נעשה שימוש בפונקציית האקטיבציה ReLU שכן היא מאמנת את הרשות ללא כניסה משמעותית



בדיקת הכללי של המערכת. כמו כן, הפונקציה הניל מפסיקת את הערכאים השליליים ומשאירת את הערכאים חיוביים כפי שהם. פונקציה זו יעילה מאוד בזמן הריצה ועלות המשאבים שלה ולכון נפוצה ביותר.

**פתרון משימה I :**

**המודל:**

שם השכבה	פרמטרים <sup>1</sup>	אקטיבציה	פילטרים גרעין	גודל כניסה	גודל יציאה (פרמטרים)	מיינד היציאה
						מספר פילטרים גודל אפס
Conv2D	32 (3,3)	Relu	(3,3)	180X180X1	$\left( \begin{array}{c} (3 \cdot 3) \cdot 1 \\ Kernel \quad RGB \end{array} \right) + 1 \cdot 32 = 320$	180X180X1
MaxPooling2D	(2,2)			180X180X1	0	90X90X1
Conv2D	64 (3,3)	Relu	(3,3)	90X90X1	$\left( \begin{array}{c} (3 \cdot 3) \cdot 32 \\ Kernel \quad Pre-Layer \end{array} \right) + 1 \cdot 64 = 18496$	90X90X1
MaxPooling2D	(2,2)			90X90X1	0	45X45X1
Conv2D	64 (3,3)	Relu	(3,3)	45X45X1	$\left( \begin{array}{c} (3 \cdot 3) \cdot 64 \\ Kernel \quad Pre-Layer \end{array} \right) + 1 \cdot 64 = 36928$	45X45X1
Dropout	0.1			45X45X1	0	45X45X1
MaxPooling2D	(2,2)			45X45X1	0	23X23X1
Conv2D	64 (3,3)	Relu	(3,3)	23X23X1	$\left( \begin{array}{c} (3 \cdot 3) \cdot 64 \\ Kernel \quad Pre-Layer \end{array} \right) + 1 \cdot 64 = 36928$	23X23X1
MaxPooling2D	(2,2)			23X23X1	0	12X12X1
Conv2D	128 (3,3)	Relu	(3,3)	12X12X1	$\left( \begin{array}{c} (3 \cdot 3) \cdot 64 \\ Kernel \quad Pre-Layer \end{array} \right) + 1 \cdot 128 = 73856$	12X12X1
Dropout	0.2			12X12X1	0	12X12X1
MaxPooling2D	(2,2)			12X12X1	0	6X6X1
Conv2D	256 (3,3)	Relu	(3,3)	6X6X1	$\left( \begin{array}{c} (3 \cdot 3) \cdot 128 \\ Kernel \quad Pre-Layer \end{array} \right) + 1 \cdot 256 = 295168$	6X6X1
Dropout	0.2			6X6X1	0	6X6X1
MaxPooling2D	(2,2)			6X6X1	0	3X3X1
Flatten				3X3X1	$\left( \begin{array}{c} 3 \cdot 3 \cdot 1 \cdot 256 \\ Size \quad Output-Layers \end{array} \right) = 2304$	$\left( \begin{array}{c} col-vec \end{array} \right)$
Dropout	0.2			2304	0	2304
Dense	Sigmoid			2305	1	464001
<b>סה"כ פרמטרים לאימון</b>						

נבחר את פונקציית Sigmoid מכיוון שבמוצאה נוכל לקבל ערכים רציפים בין 0 ל-1, כנדרש.

רשת זו נבחרה לאחר חקירה באינטרנט ובמעבדות קודמות שנעשו במהלך הקורס, כמו גם בדיקת רשותות רבות אחרות. לאחר מכן הגענו למסקנה שמבנה זה נותן את התוצאות הטובות ביותר בעברנו.

<sup>1</sup> בכלל השכבות Conv2D ו MaxPooling2D נעשה שימוש ב-padding = same ו strides = 1,2.

להלן דוח הסיכום כפי שמופיע בהרצת המודל:

Layer (type)	Output Shape	Param #
<hr/>		
conv2d_18 (Conv2D)	(None, 180, 180, 32)	320
max_pooling2d_18 (MaxPooling2D)	(None, 90, 90, 32)	0
conv2d_19 (Conv2D)	(None, 90, 90, 64)	18496
max_pooling2d_19 (MaxPooling2D)	(None, 45, 45, 64)	0
conv2d_20 (Conv2D)	(None, 45, 45, 64)	36928
dropout_12 (Dropout)	(None, 45, 45, 64)	0
max_pooling2d_20 (MaxPooling2D)	(None, 23, 23, 64)	0
conv2d_21 (Conv2D)	(None, 23, 23, 64)	36928
max_pooling2d_21 (MaxPooling2D)	(None, 12, 12, 64)	0
conv2d_22 (Conv2D)	(None, 12, 12, 128)	73856
dropout_13 (Dropout)	(None, 12, 12, 128)	0
max_pooling2d_22 (MaxPooling2D)	(None, 6, 6, 128)	0
conv2d_23 (Conv2D)	(None, 6, 6, 256)	295168
dropout_14 (Dropout)	(None, 6, 6, 256)	0
max_pooling2d_23 (MaxPooling2D)	(None, 3, 3, 256)	0
flatten_3 (Flatten)	(None, 2304)	0
dropout_15 (Dropout)	(None, 2304)	0
dense_3 (Dense)	(None, 1)	2305
<hr/>		
Total params:	464001 (1.77 MB)	
Trainable params:	464001 (1.77 MB)	
Non-trainable params:	0 (0.00 Byte)	

להלן קוד המודל, כפי שנכתב בפלטפורמת Google Colab

```

CNN = models.Sequential()

CNN.add(layers.Conv2D(32, (3, 3), padding = 'same' ,activation='relu',
input_shape=(180, 180, 1)))
CNN.add(layers.MaxPooling2D((2,2) , strides = 2 , padding = 'same'))
CNN.add(layers.Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation
= 'relu'))

CNN.add(layers.MaxPooling2D((2,2) , strides = 2 , padding = 'same'))
CNN.add(layers.Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation
= 'relu'))
CNN.add(Dropout(0.1))

CNN.add(layers.MaxPooling2D((2,2) , strides = 2 , padding = 'same'))
CNN.add(layers.Conv2D(128 , (3,3) , strides = 1 , padding = 'same' , activation
= 'relu'))
CNN.add(Dropout(0.2))

CNN.add(layers.MaxPooling2D((2,2) , strides = 2 , padding = 'same'))
CNN.add(layers.Conv2D(256 , (3,3) , strides = 1 , padding = 'same' , activation
= 'relu'))
CNN.add(Dropout(0.2))

CNN.add(layers.MaxPooling2D((2,2) , strides = 2 , padding = 'same'))

CNN.add(layers.Flatten())
CNN.add(Dropout(0.2))
CNN.add(layers.Dense(1, activation = 'sigmoid'))

# Compile the model
CNN.compile(optimizer = Adagrad(learning_rate=0.01),
            loss='binary_crossentropy',
            metrics=['accuracy'])

CNN.summary()

history = CNN.fit(train_norm, train_label, epochs = 10, batch_size = 20,
                   validation_data=(val_norm, val_label), verbose=1)

```

### **פתרון משימה II :**

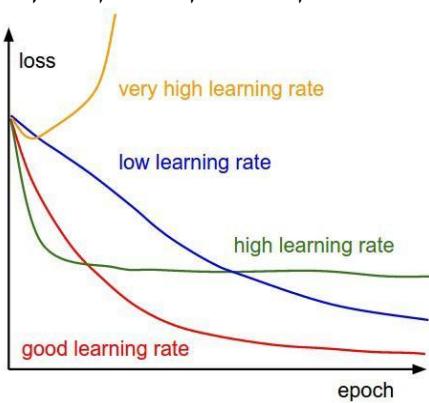
כפי שתיארנו קודם, בחרנו את אלגוריתם האימון להיות Adagrad שמיועד לאימון רשותת נוירוניים. הייחודיות של אלגוריתם זה הינה שינוי קצב הלמידה שלו (Learning Rate) לכל פרמטר בזמן הריצה.

**מספר Epochs:** 10, מספר הפעמים ששורקים את סט האימון בזמן אימון המודל.

**קצב שינוי הפרמטרים** ביחס לגרדיינט של פונקציית הנקס במהלך האימון.

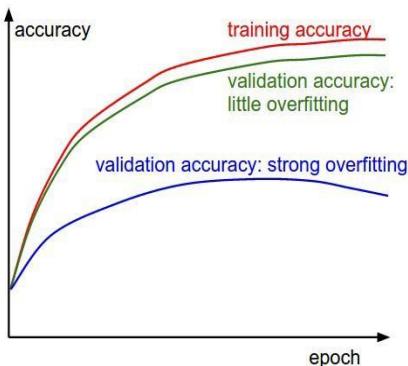
**מספר תתי קבוצות** של סט האימון, בין היתר מונע OF וגורם להתקנסות טובה יותר.

**פונקציית הפסד (Loss-Function)**: פונקציה היא ממד לכמה תוצאות המודל שלנו מתאימות לתוצאות האמתיות של הכניסה. מטרתה העיקרית של פונקציה זו הינה לבדוק חוסר עקבות בין הערך החזוי לערך האמתי של וקטור הכניסה לרשת.

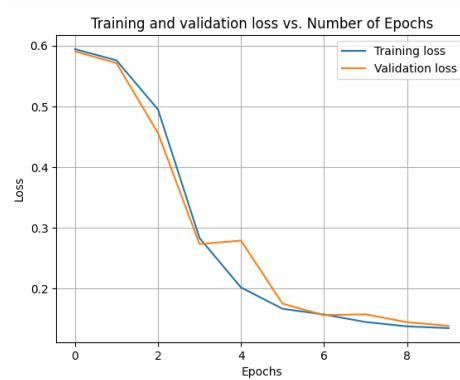


**בדיקה של מערכת לומדת קבוע על פי מידת הקربה של כמות של המדידות,**

לערך המשמעותי של אותה כמות.



### להלן תוצאות הרצת המודל:

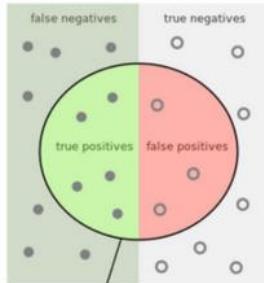


Test accuracy: 0.9496586918830872

Test loss: 0.14723195135593414

ניתן לראות כי ישנה התאמה טובה מאוד בין גרפ' הווידציה והאימון בדיאגרמת הדיווק והקנס. כמו כן, קיבלנו דיווק של 94.6% בהרצת המודל על סט הבדיקה, דבר המאשר כי למודל שבנו יישנים ביצועים טובים (אחוז דיווק של מעל 93% על טסט הבדיקה, לנדרש).

נרצה לשרטט את מטריצת הבלבול (Confusion Matrix) במטרה לבדוק את Recall ו-Precision.



$$\begin{aligned} precision &= \frac{TP}{TP + FP} \\ recall &= \frac{TP}{TP + FN} \\ F1 &= \frac{2 \times precision \times recall}{precision + recall} \\ accuracy &= \frac{TP + TN}{TP + FN + TN + FP} \end{aligned}$$

יכולת של המודל לזהות נכון את הדוגמאות הרלוונטיות מסט הנתונים.

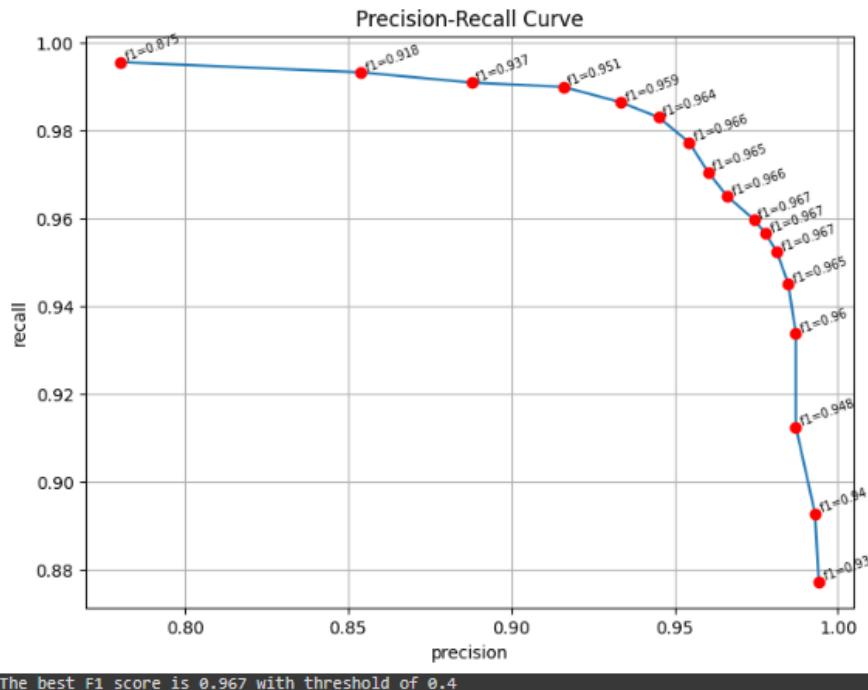
היחס בין החיזוי הנכון (TP) לסה"כ כל הדוגמאות שהמודל סיוג נכון כנכון (TP+FP).

		Confusion Matrix	
		TN	FP
True Labels	Healthy	286	30
	Sick	29	827
		FN	TP
Predicted Labels	Healthy	29	827
	Sick	30	286

$$Recall = \frac{TP}{TP + FN} = \frac{827}{827 + 29} \cdot 100\% = 96.61\%$$

$$Precision = \frac{TP}{TP + FP} = \frac{827}{827 + 30} \cdot 100\% \approx 96.5\%$$

נشرط כעת את גרפ' ה-Recall כתלות ב-Precision :  
 נרצה למצוא את ה- F1 הטוב ביותר, כלומר את מzd האיזון בין ה-Recall וה-Precision הגובה ביותר.



ניתן לראות כי עבור הסטברות סף של 0.4 (כלומר אם במצוא הרשות התקבלה הסטברות גבואה יותר אזי הצלום יסוג כחולה) עם יחס של 0.967.

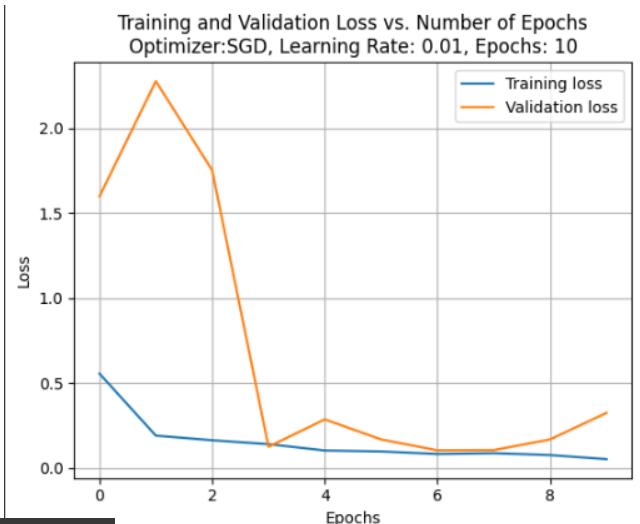
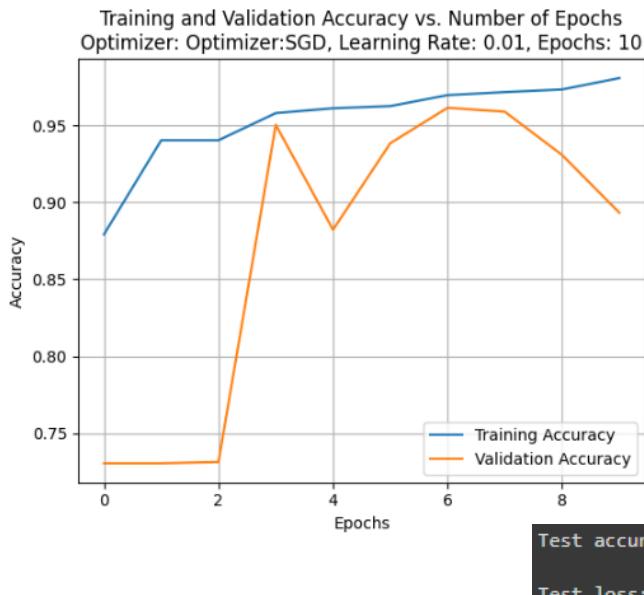
### פתרון משימה III:

את כל האלגוריתמים הרצנו על Batch-size של 20 חלקות.

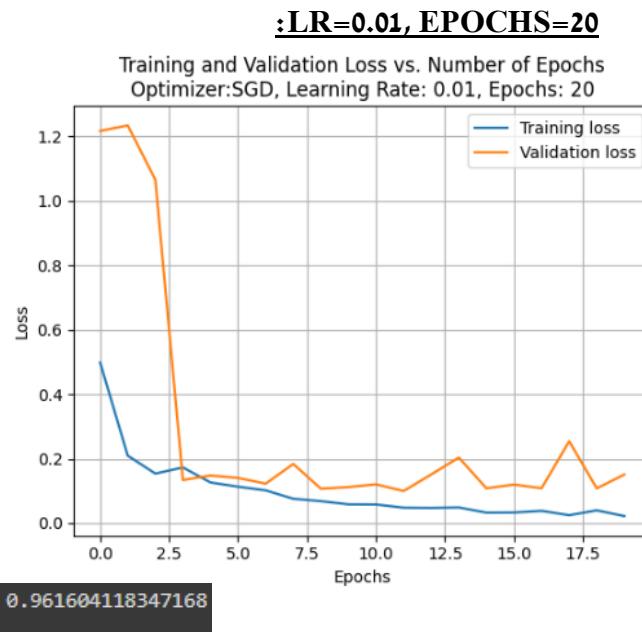
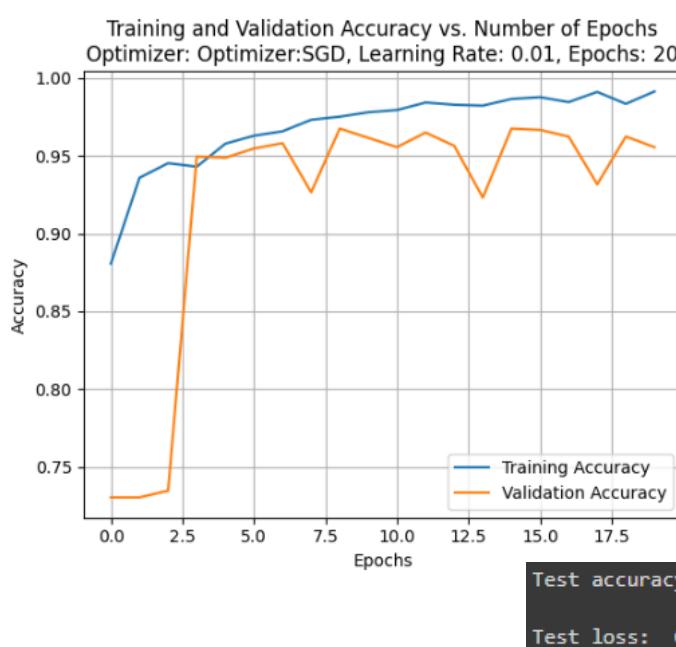
#### :SGD

אלגוריתם איטרטיבי שנועד לモען את פונקציית הכנס של המודל שלנו. האלגוריתם מחשב את הגרדיאנט ביחס ל- $\text{Mini-Batch}$ . לאחר חישוב הגרדיאנט, האלגוריתם מעדכן את הפרמטרים בכיוון המנוגד לכיוון הגרדיאנט.

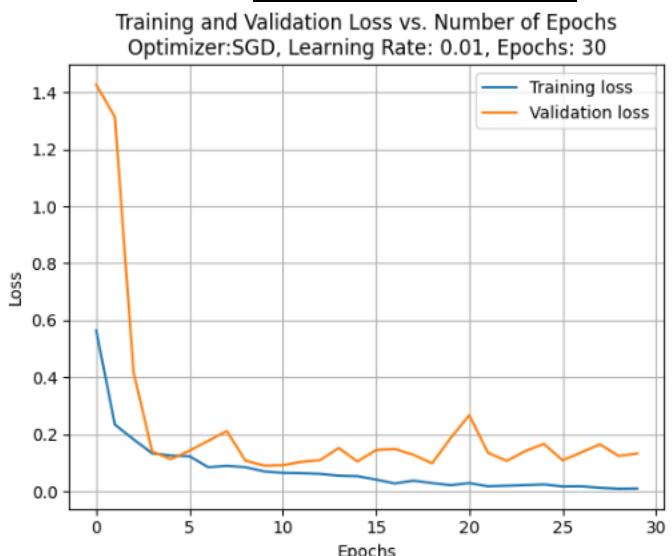
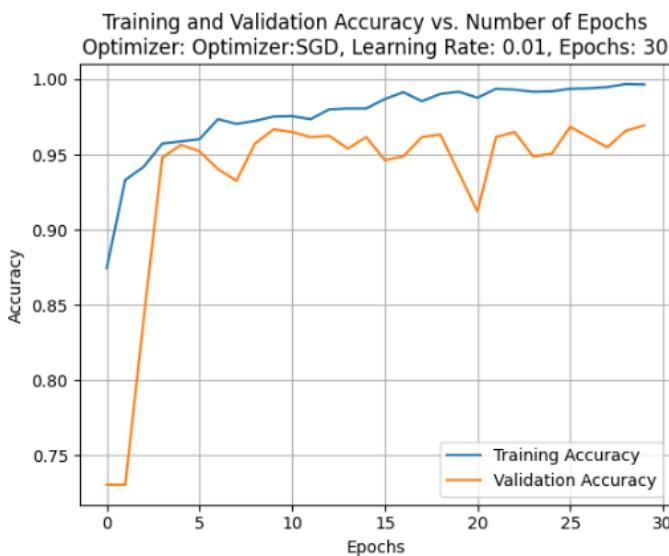
#### :LR=0.01, EPOCHS=10



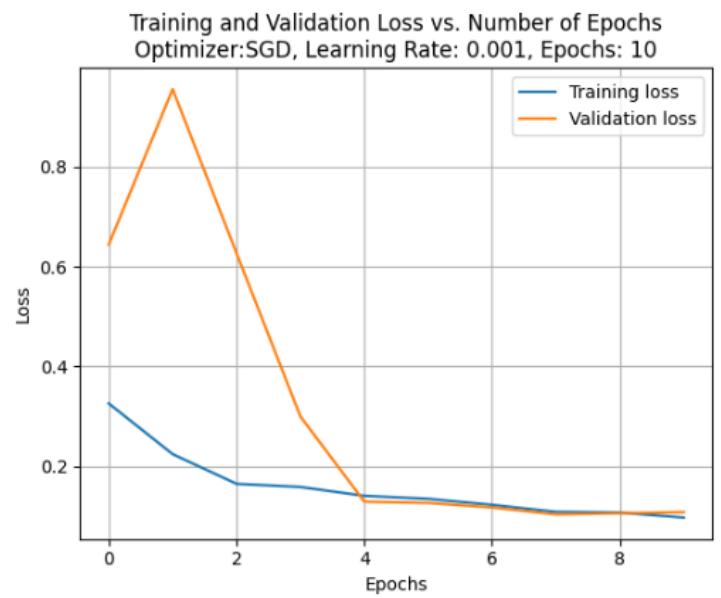
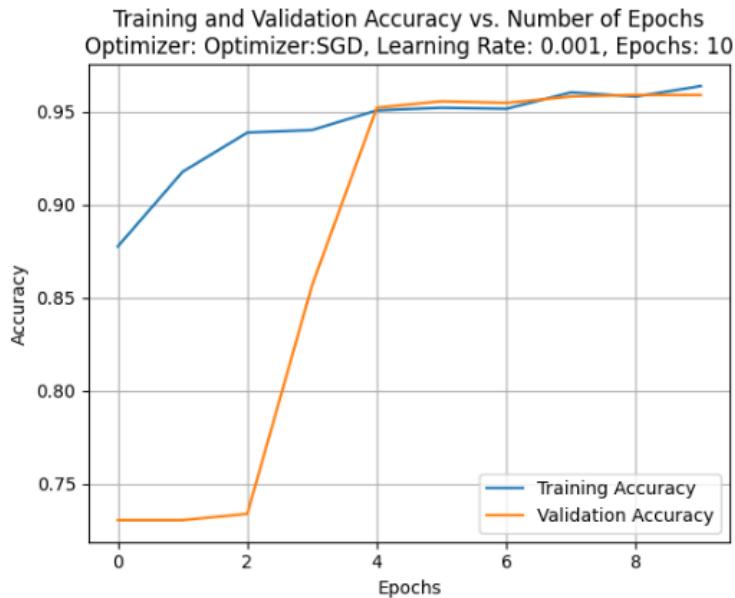
Test accuracy: 0.9027303457260132  
Test loss: 0.3178313970565796



Test accuracy: 0.961604118347168  
Test loss: 0.14710426330566406

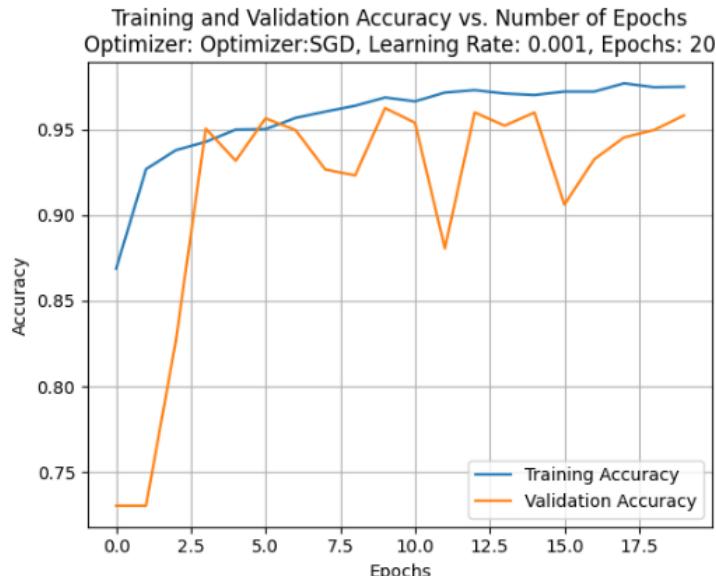


Test accuracy: 0.9633105993270874  
Test loss: 0.15190963447093964

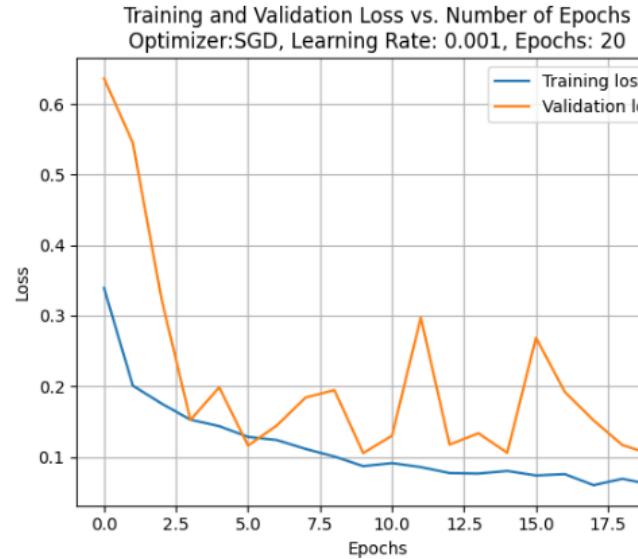


Test accuracy: 0.9590443968772888  
Test loss: 0.10816389322280884

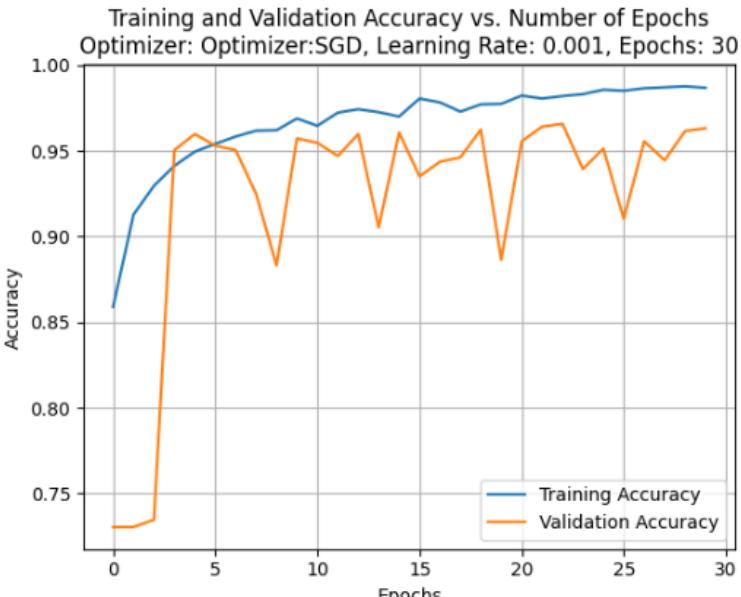
:LR=0.001, EPOCHS=20



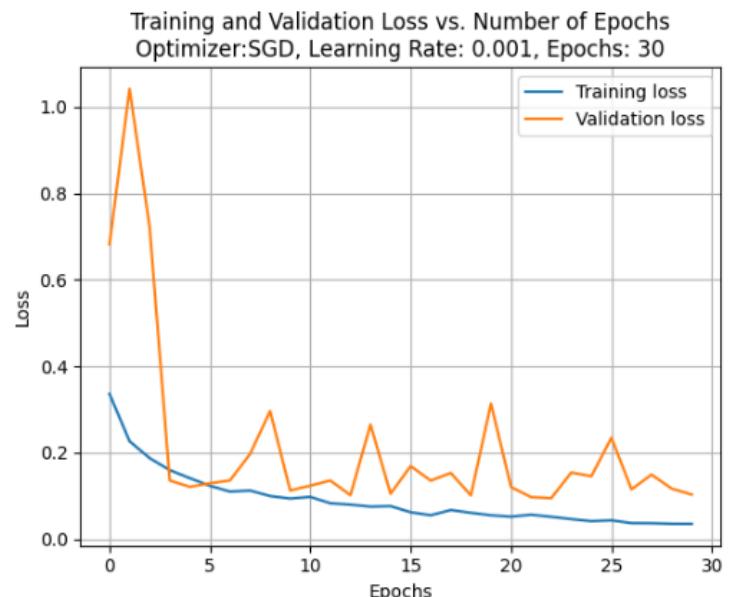
Test accuracy: 0.9650170803070068  
Test loss: 0.08840573579072952



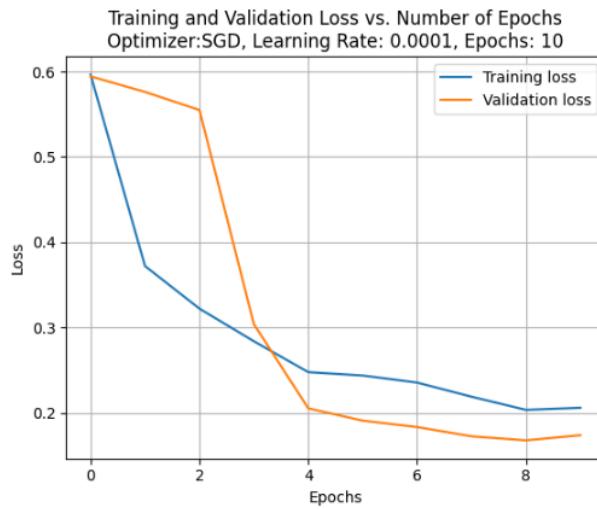
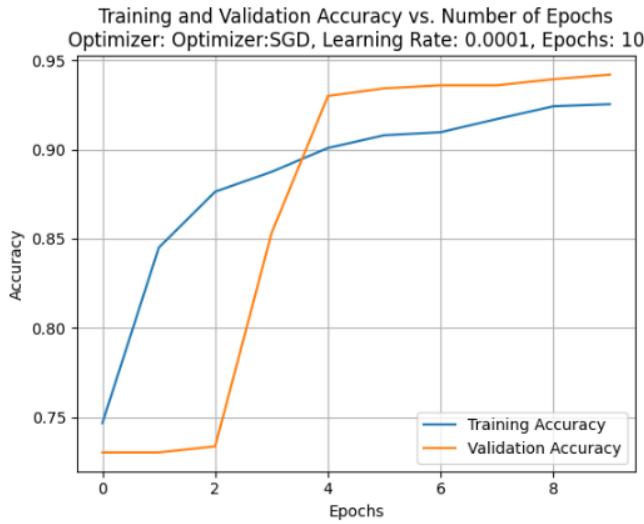
:LR=0.001, EPOCHS=30



Test accuracy: 0.9624573588371277  
Test loss: 0.10242335498332977



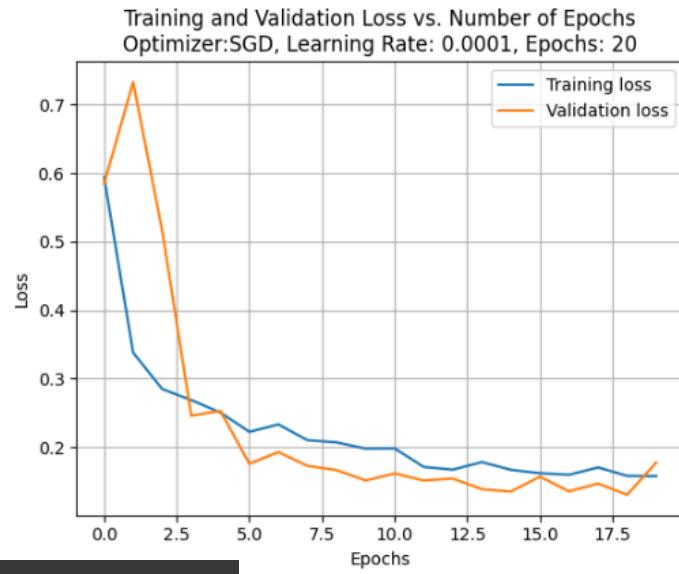
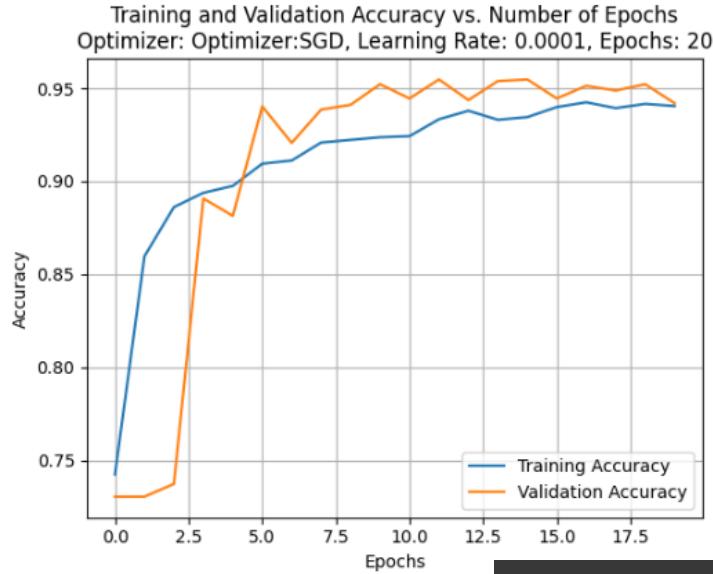
**:LR=0.0001, EPOCHS=10**



Test accuracy: 0.9308874011039734

Test loss: 0.17165958881378174

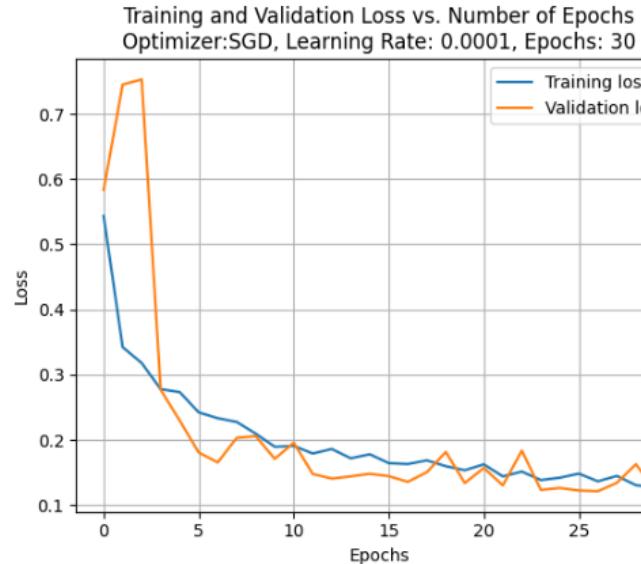
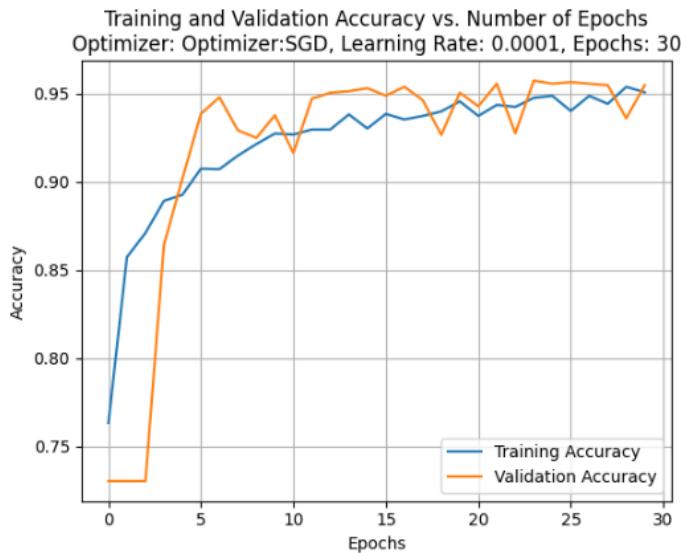
**:LR=0.0001, EPOCHS=20**



Test accuracy: 0.9394198060035706

Test loss: 0.171303853392601

**:LR=0.0001, EPOCHS=20**



Test accuracy: 0.9607508778572083

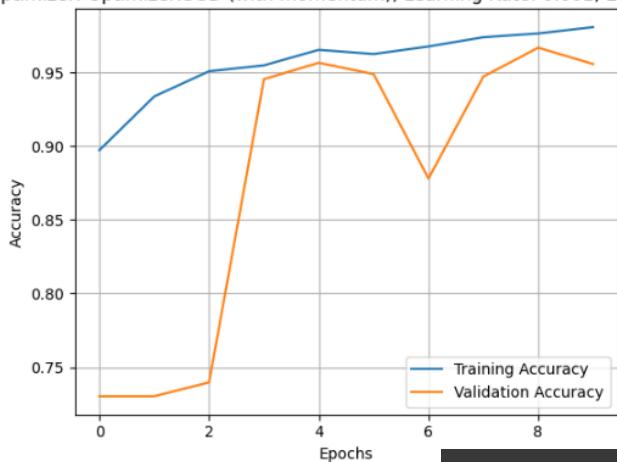
Test loss: 0.11468913406133652

**:SGD, With Momentum=0.9**

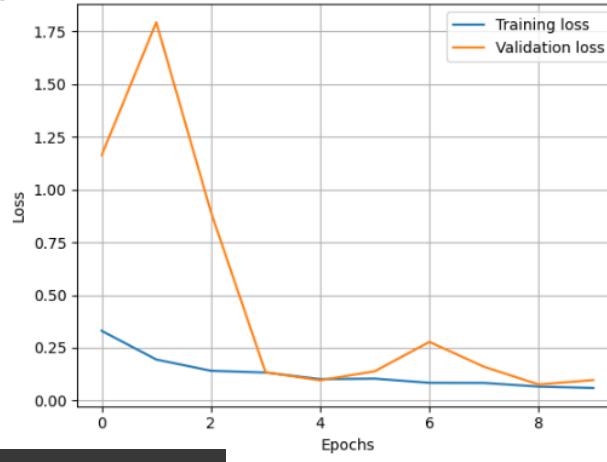
בתחילת זה, ישנה האצה של תהליכי SGD בכיוון הרלוונטי של הגרדיינט, כך שההאצה גבוהה יותר במקרים מסוימים ולאחר מכן תנודות בשינוי הגרדיינט.

**:LR=0.001, EPOCHS=10**

Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:SGD (with momentum), Learning Rate: 0.001, Epochs: 10



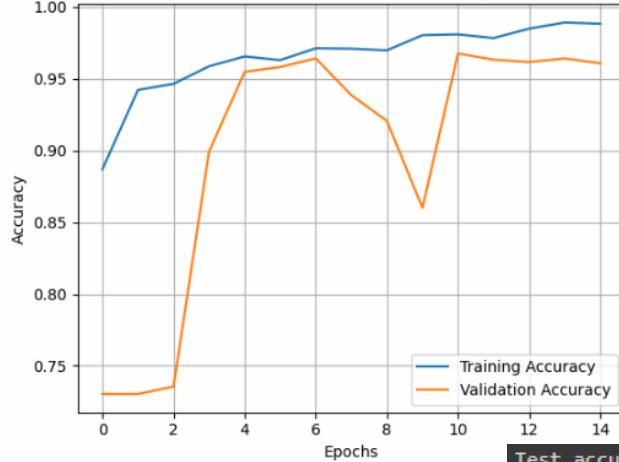
Training and Validation Loss vs. Number of Epochs  
Optimizer:SGD (with momentum), Learning Rate: 0.001, Epochs: 10



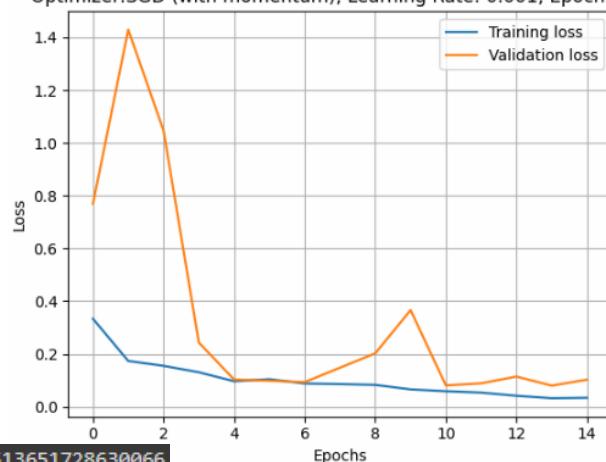
Test accuracy: 0.9564846158027649  
Test loss: 0.1181432455778122

**:LR=0.001, EPOCHS=15**

Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:SGD (with momentum), Learning Rate: 0.001, Epochs: 15



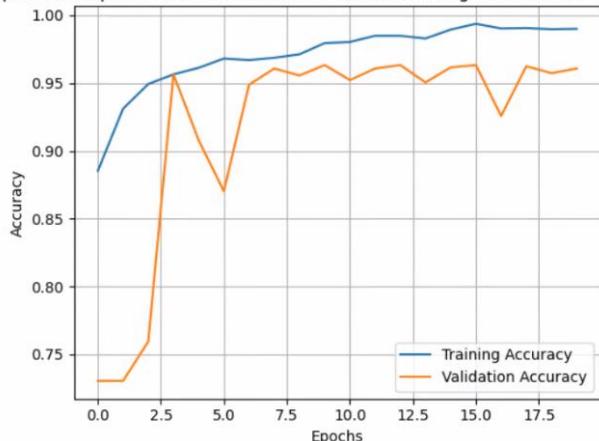
Training and Validation Loss vs. Number of Epochs  
Optimizer:SGD (with momentum), Learning Rate: 0.001, Epochs: 15



Test accuracy: 0.9513651728630066  
Test loss: 0.1442490518093109

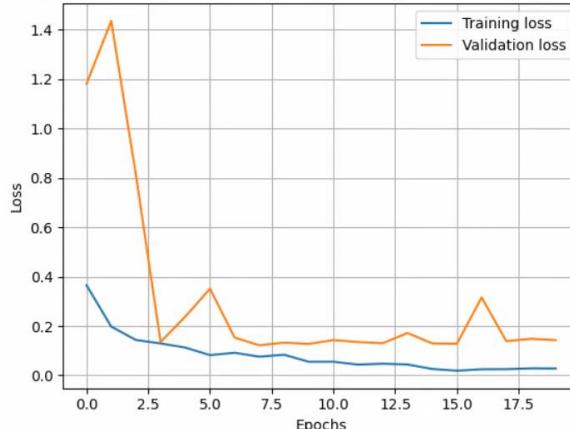
**:LR=0.001, EPOCHS=20**

Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:SGD (with momentum), Learning Rate: 0.001, Epochs: 20



Test accuracy: 0.9658703207969666

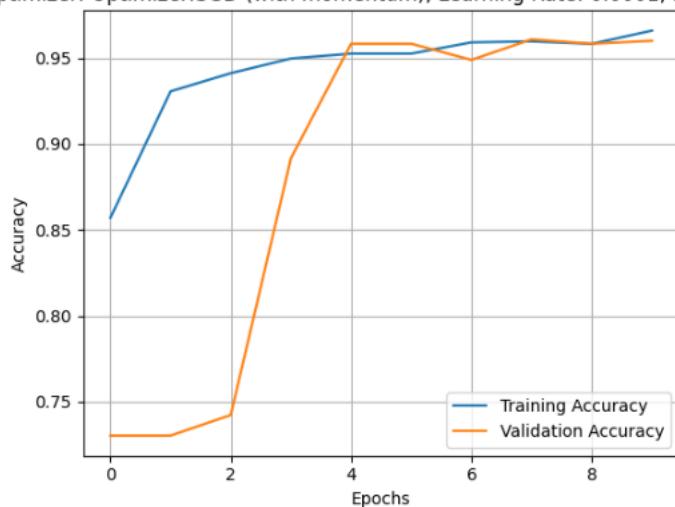
Training and Validation Loss vs. Number of Epochs  
Optimizer:SGD (with momentum), Learning Rate: 0.001, Epochs: 20



Test loss: 0.12045203149318695

**:LR=0.0001, EPOCHS=10**

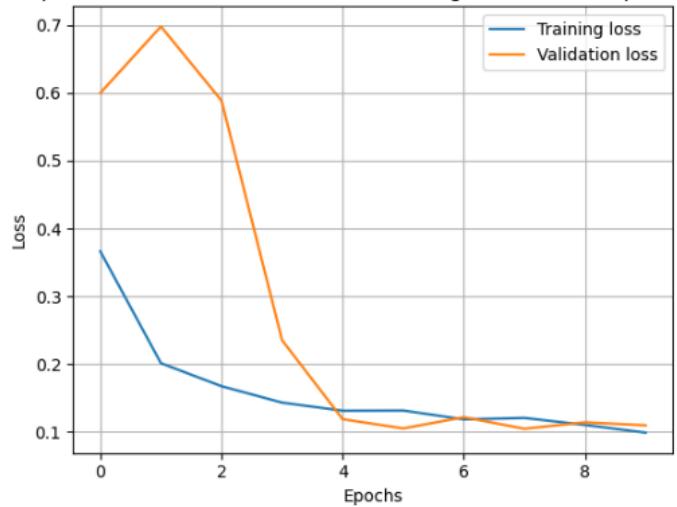
Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:SGD (with momentum), Learning Rate: 0.0001, Epochs: 10



Test accuracy: 0.9368600845336914

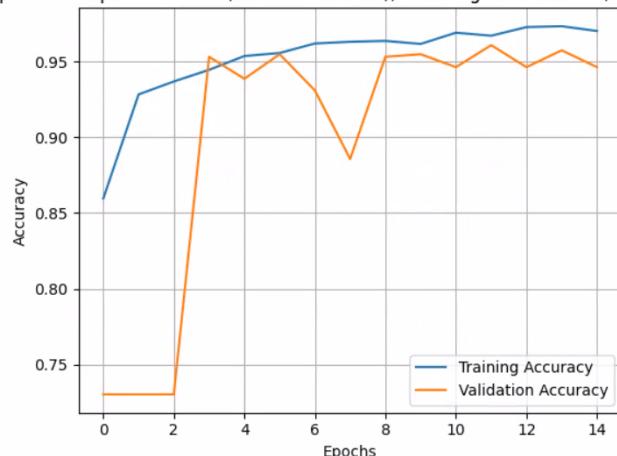
Test loss: 0.14904485642910004

Training and Validation Loss vs. Number of Epochs  
Optimizer:SGD (with momentum), Learning Rate: 0.0001, Epochs: 10

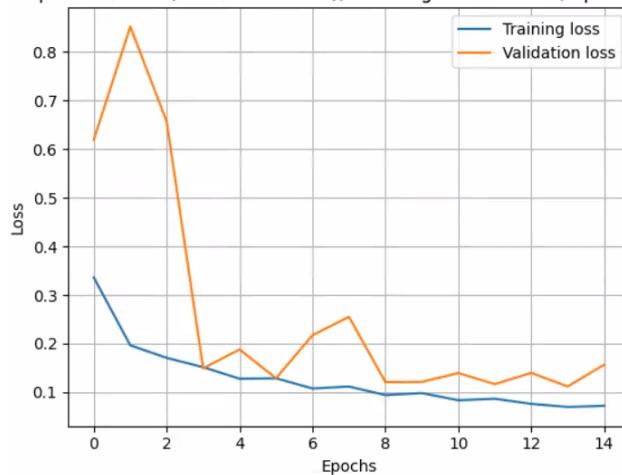


**:LR=0.0001, EPOCHS=15**

Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:SGD (with momentum), Learning Rate: 0.0001, Epochs: 15



Training and Validation Loss vs. Number of Epochs  
Optimizer:SGD (with momentum), Learning Rate: 0.0001, Epochs: 15

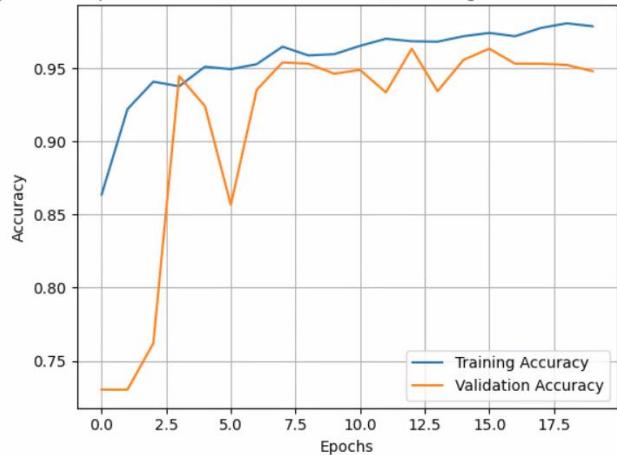


Test accuracy: 0.9488054513931274

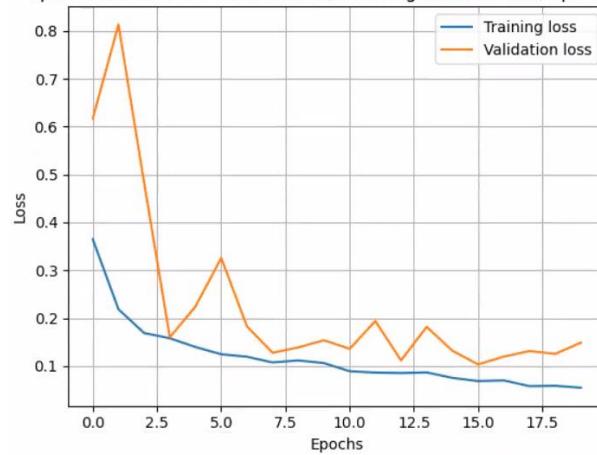
Test loss: 0.13174082338809967

**:LR=0.0001, EPOCHS=20**

Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:SGD (with momentum), Learning Rate: 0.0001, Epochs: 20



Training and Validation Loss vs. Number of Epochs  
Optimizer:SGD (with momentum), Learning Rate: 0.0001, Epochs: 20

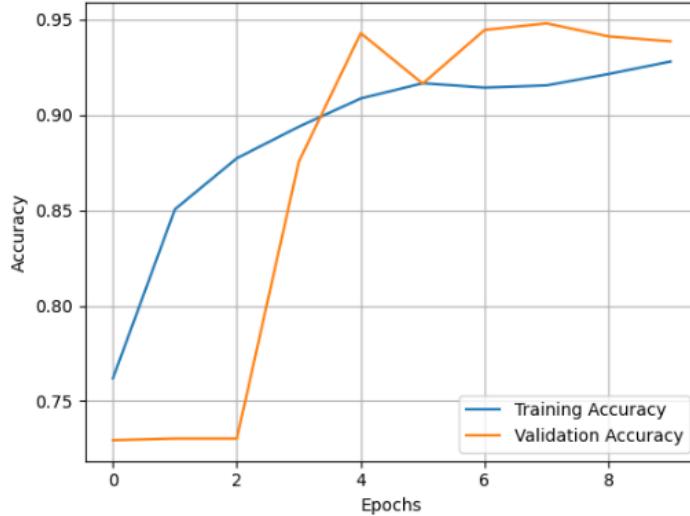


Test accuracy: 0.947098970413208

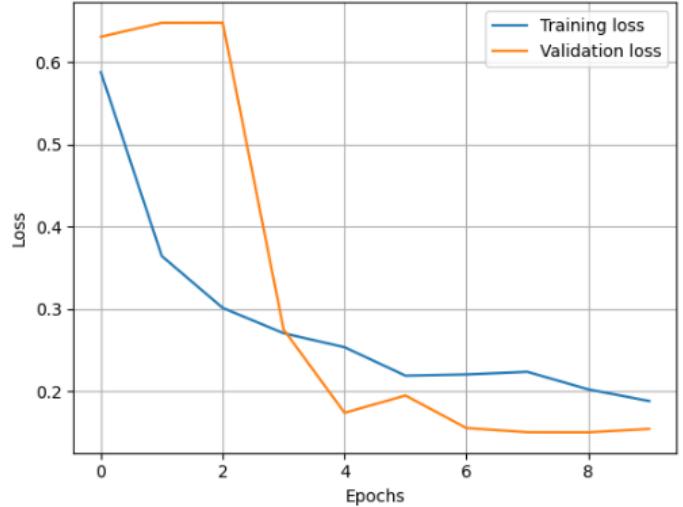
Test loss: 0.13737009465694427

**:LR=0.00001, EPOCHS=10**

Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:SGD (with momentum), Learning Rate: 1e-05, Epochs: 10



Training and Validation Loss vs. Number of Epochs  
Optimizer:SGD (with momentum), Learning Rate: 1e-05, Epochs: 10

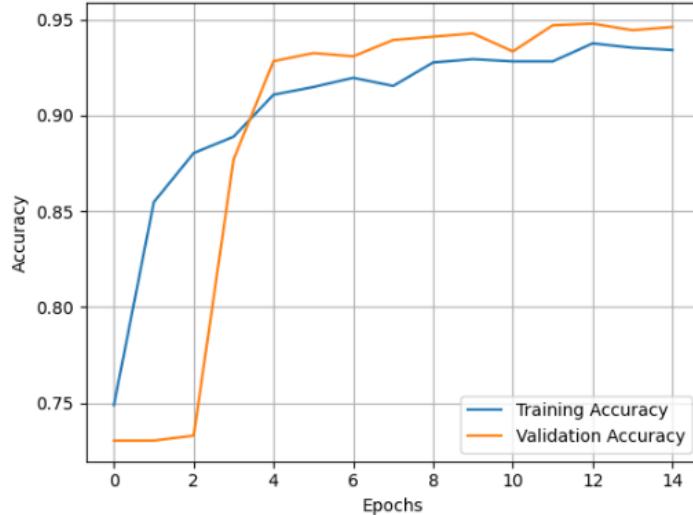


Test accuracy: 0.9274743795394897

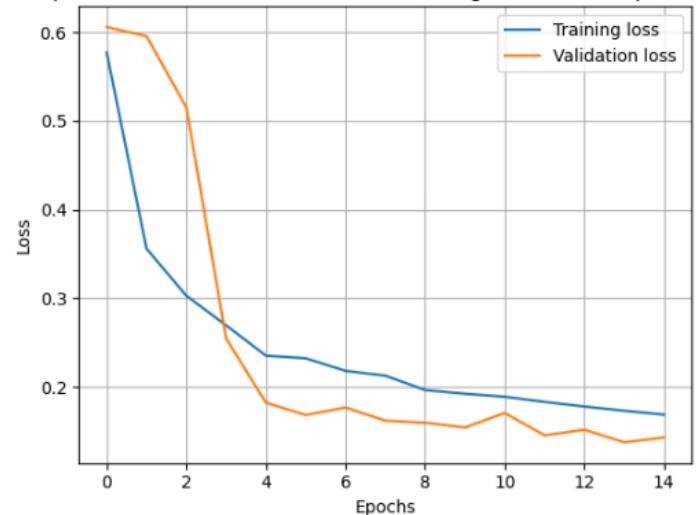
Test loss: 0.182485893368721

**:LR=0.00001, EPOCHS=15**

Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:SGD (with momentum), Learning Rate: 1e-05, Epochs: 15



Training and Validation Loss vs. Number of Epochs  
Optimizer:SGD (with momentum), Learning Rate: 1e-05, Epochs: 15

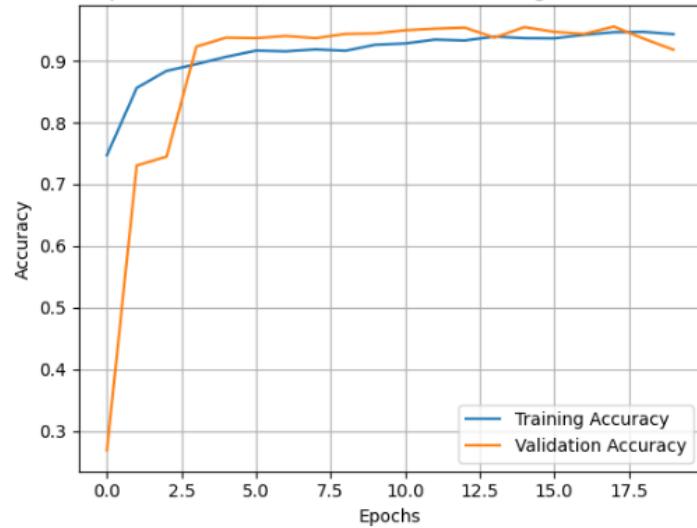


Test accuracy: 0.9402730464935303

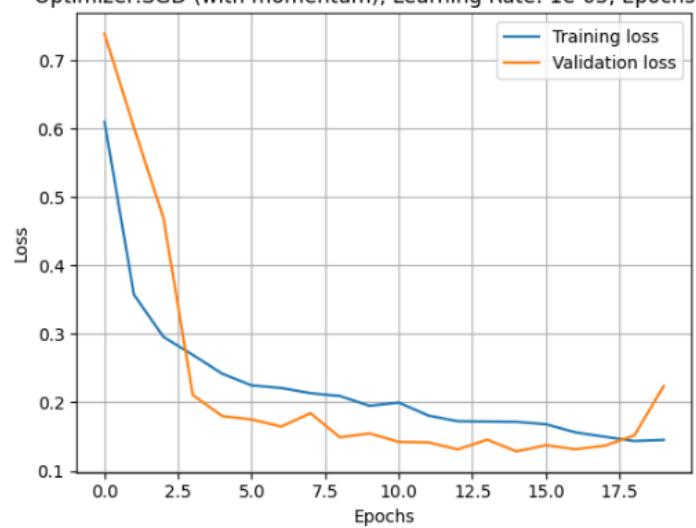
Test loss: 0.16816368699073792

:LR=0.00001, EPOCHS=20

Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:SGD (with momentum), Learning Rate: 1e-05, Epochs: 20



Training and Validation Loss vs. Number of Epochs  
Optimizer:SGD (with momentum), Learning Rate: 1e-05, Epochs: 20



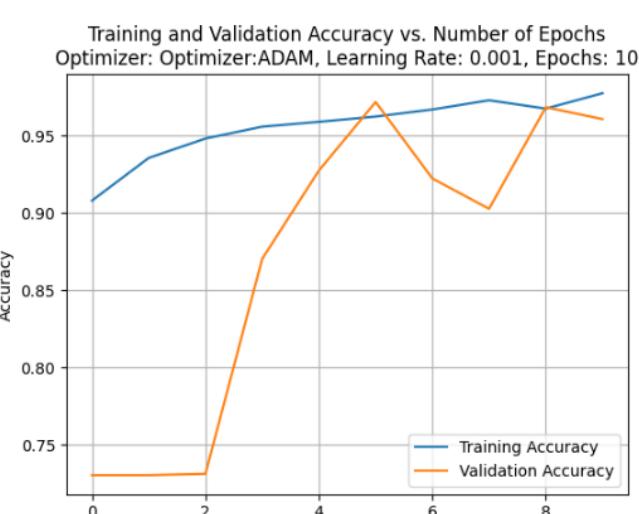
Test accuracy: 0.908703088760376

Test loss: 0.24808497726917267

### :Adam

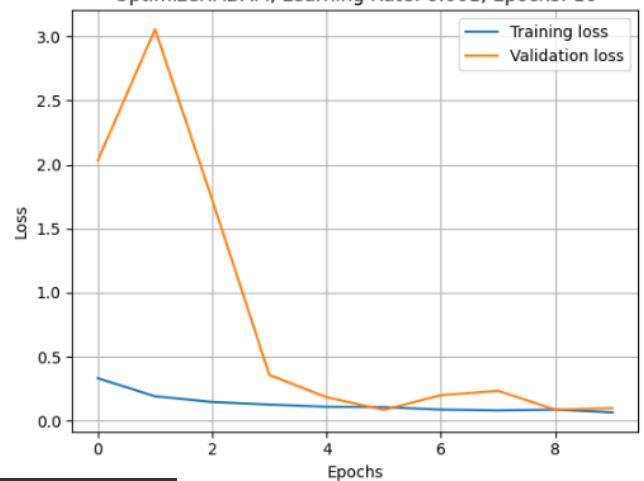
אלגוריתם זה הינו שילוב של אלגוריתם Momentum ו-RMSprop אשר גם מתאים לשיעור למידה שונה לכל משתנה וمعدכן אותו לפי הקצב שלו.

Adam גם מצליח להתגבר על בעיות ההתקנסות ומהירות וגם אפשר לטפל נכון בתנודותיות Momentum.

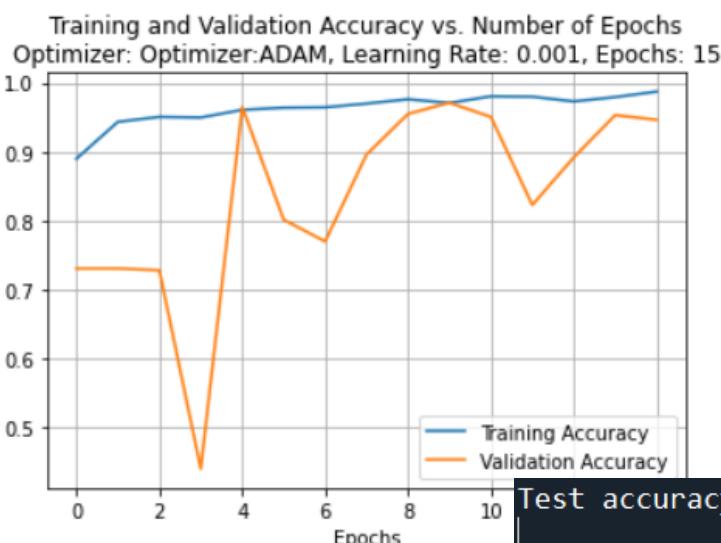


### :LR=0.001, EPOCHS=10

Training and Validation Loss vs. Number of Epochs  
Optimizer: ADAM, Learning Rate: 0.001, Epochs: 10

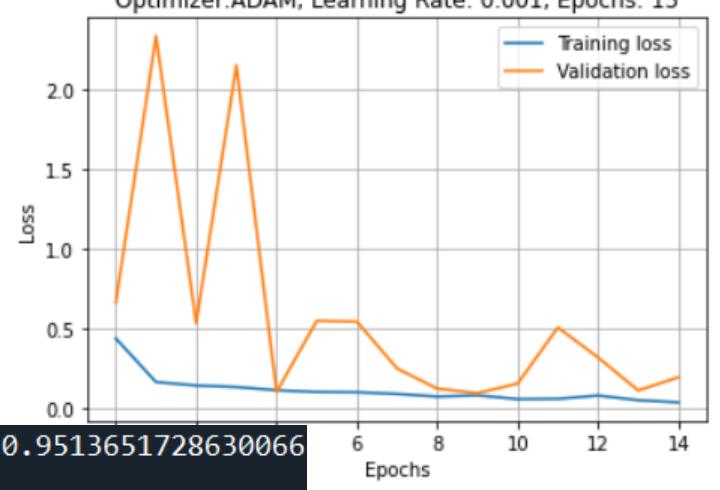


Test accuracy: 0.9402730464935303  
 Test loss: 0.15009984374046326



### :LR=0.001, EPOCHS=15

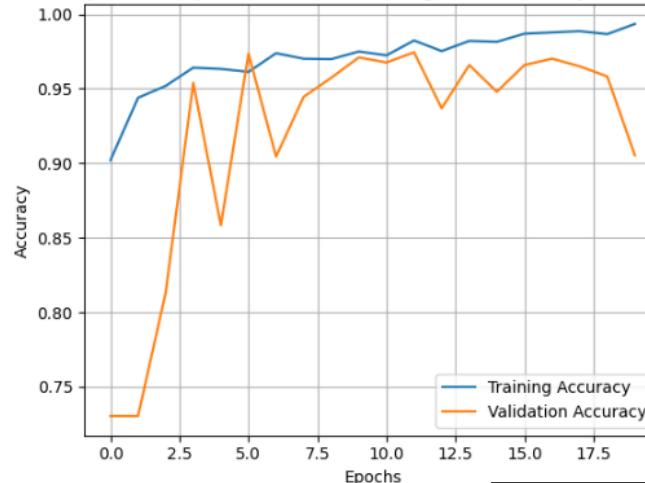
Training and Validation Loss vs. Number of Epochs  
Optimizer: ADAM, Learning Rate: 0.001, Epochs: 15



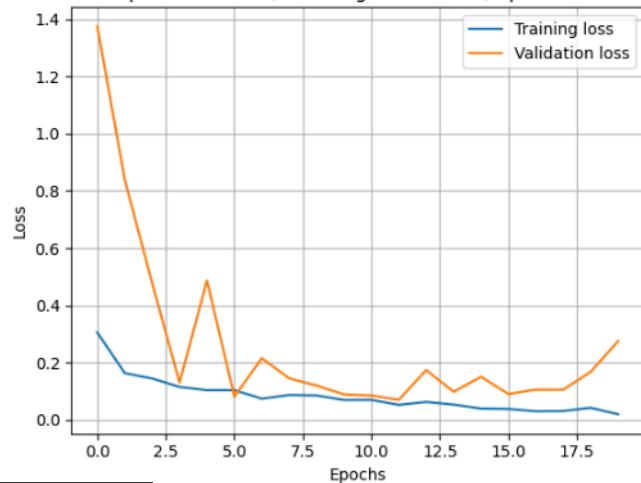
Test accuracy: 0.9513651728630066  
 Test loss: 0.17794013023376465

**:LR=0.001, EPOCHS=20**

Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:ADAM, Learning Rate: 0.001, Epochs: 20



Training and Validation Loss vs. Number of Epochs  
Optimizer:ADAM, Learning Rate: 0.001, Epochs: 20

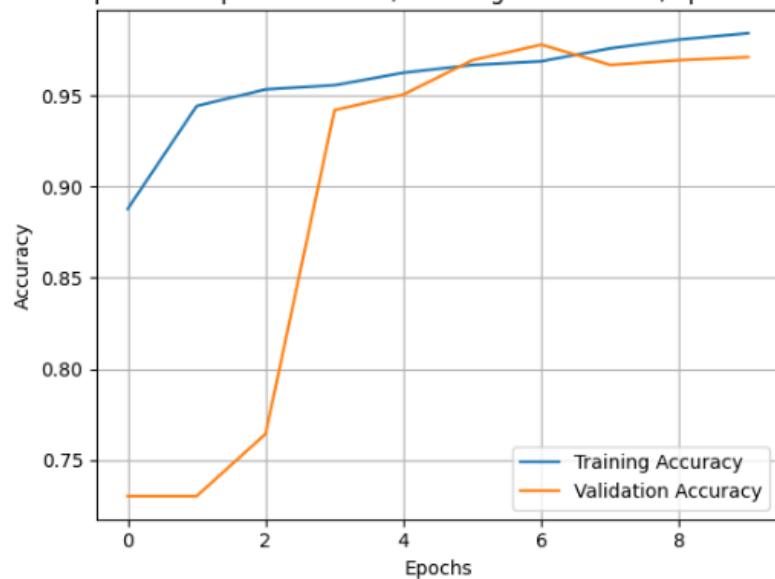


Test accuracy: 0.8873720169067383

Test loss: 0.31926748156547546

**:LR=0.0001, EPOCHS=10**

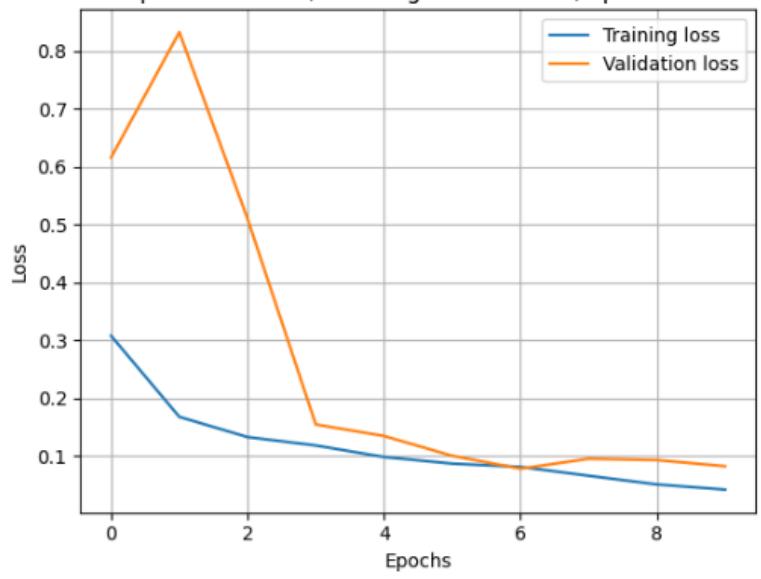
Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:ADAM, Learning Rate: 0.0001, Epochs: 10



Test accuracy: 0.953071653842926

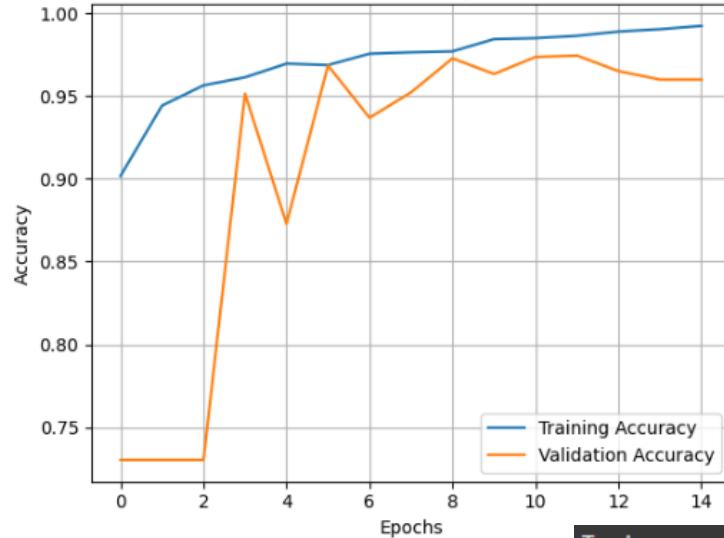
Test loss: 0.13824090361595154

Training and Validation Loss vs. Number of Epochs  
Optimizer:ADAM, Learning Rate: 0.0001, Epochs: 10



**:LR=0.0001, EPOCHS=15**

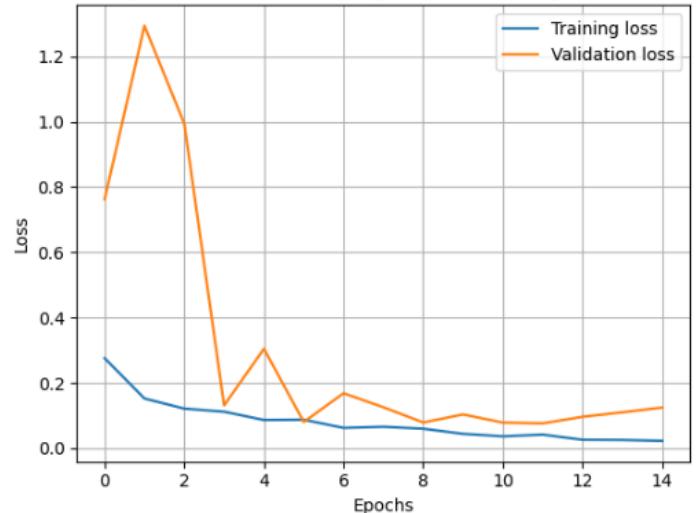
Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:ADAM, Learning Rate: 0.0001, Epochs: 15



Test accuracy: 0.9453924894332886

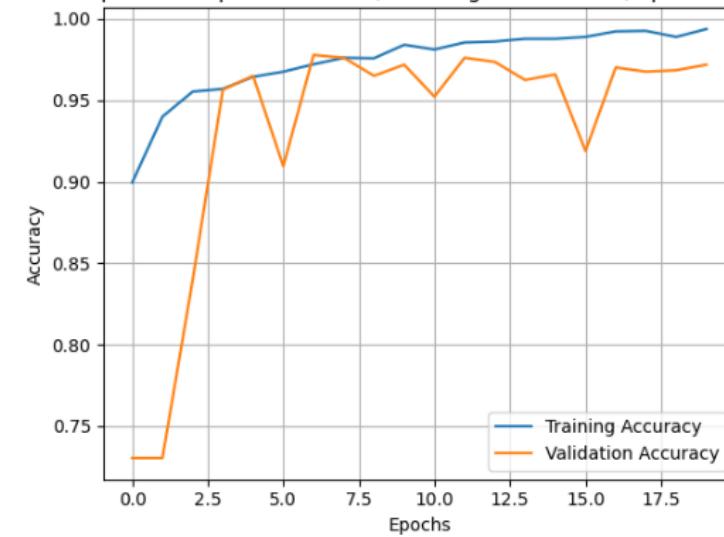
Test loss: 0.15042747557163239

Training and Validation Loss vs. Number of Epochs  
Optimizer:ADAM, Learning Rate: 0.0001, Epochs: 15



**:LR=0.0001, EPOCHS=20**

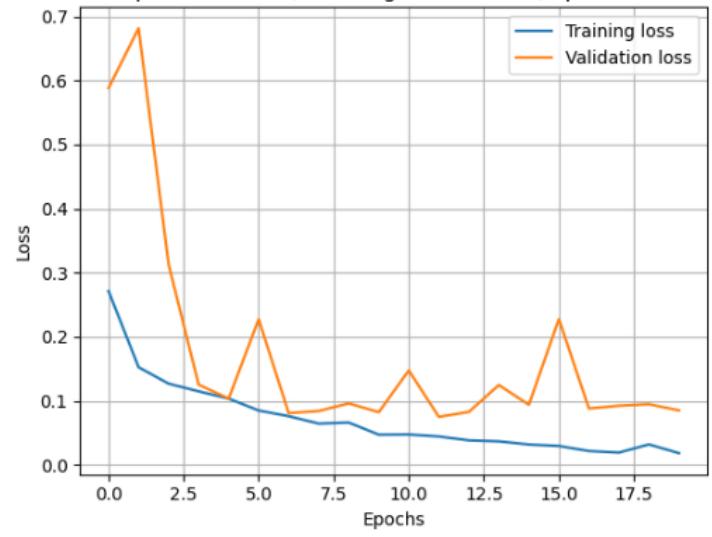
Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:ADAM, Learning Rate: 0.0001, Epochs: 20



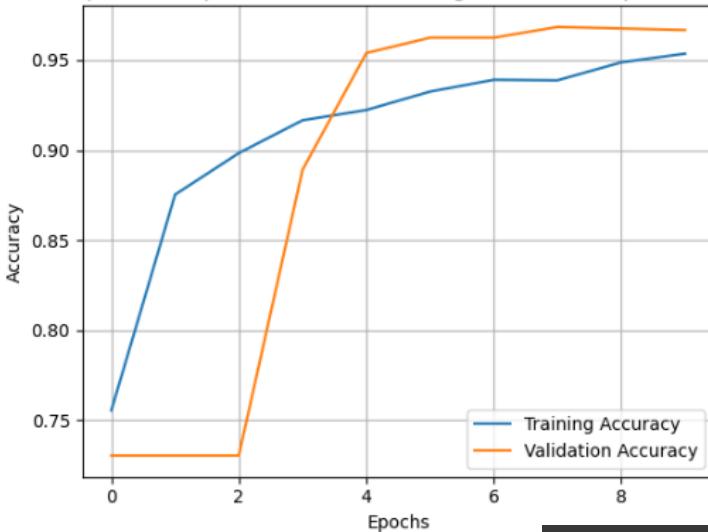
Test accuracy: 0.9564846158027649

Test loss: 0.13174504041671753

Training and Validation Loss vs. Number of Epochs  
Optimizer:ADAM, Learning Rate: 0.0001, Epochs: 20

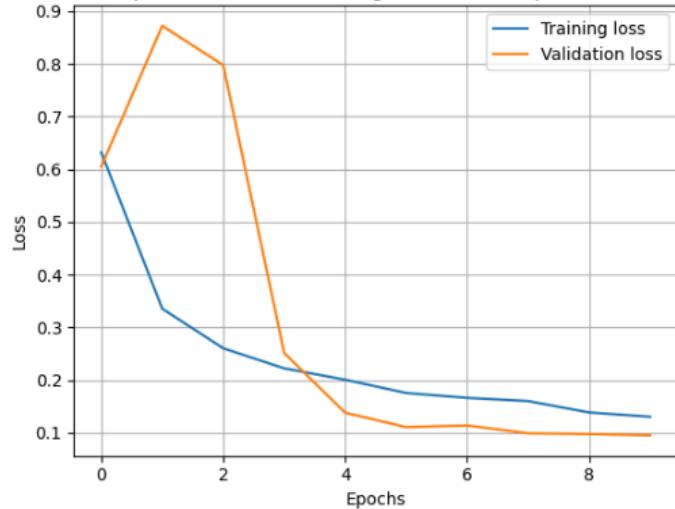


Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:ADAM, Learning Rate: 1e-05, Epochs: 10



**:LR=0.00001, EPOCHS=10**

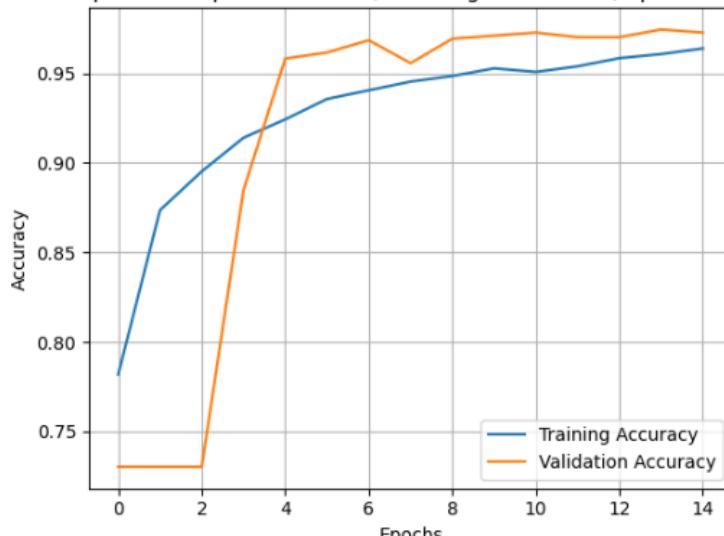
Training and Validation Loss vs. Number of Epochs  
Optimizer:ADAM, Learning Rate: 1e-05, Epochs: 10



Test accuracy: 0.9479522109031677

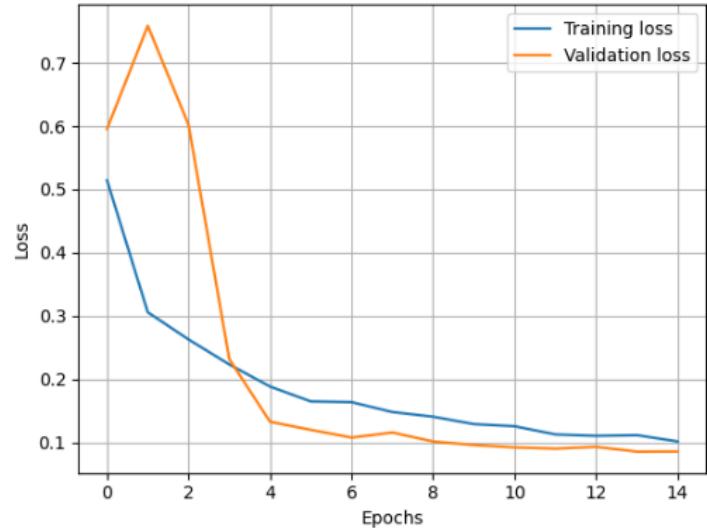
Test loss: 0.13377323746681213

Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:ADAM, Learning Rate: 1e-05, Epochs: 15



**:LR=0.00001, EPOCHS=15**

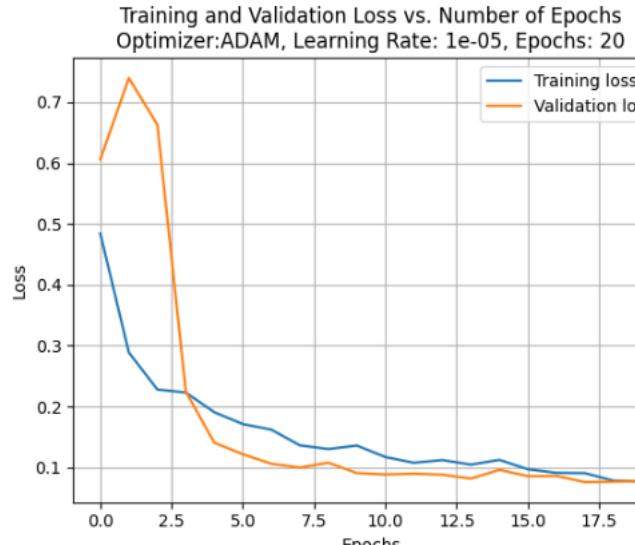
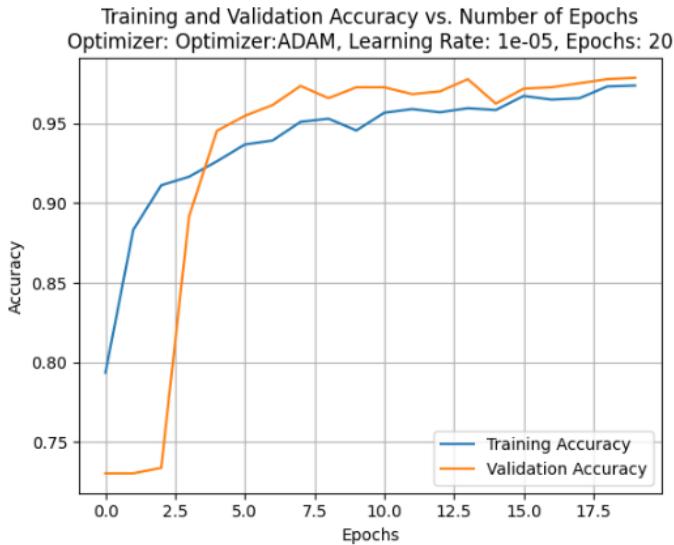
Training and Validation Loss vs. Number of Epochs  
Optimizer:ADAM, Learning Rate: 1e-05, Epochs: 15



Test accuracy: 0.9445392489433289

Test loss: 0.13239525258541107

:LR=0.00001, EPOCHS=20

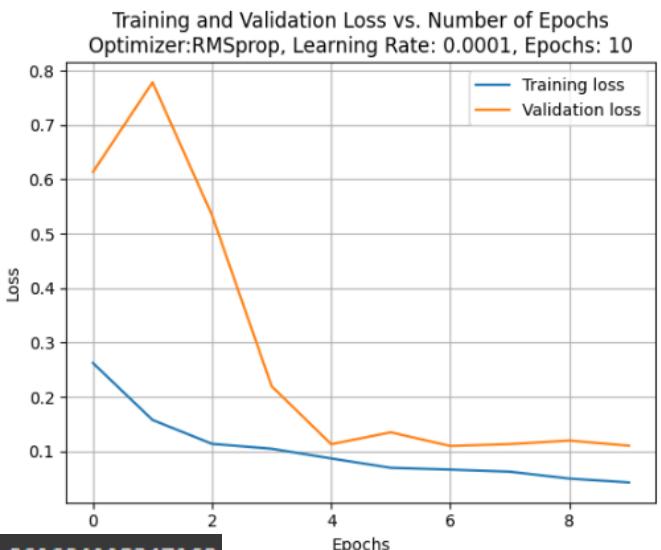
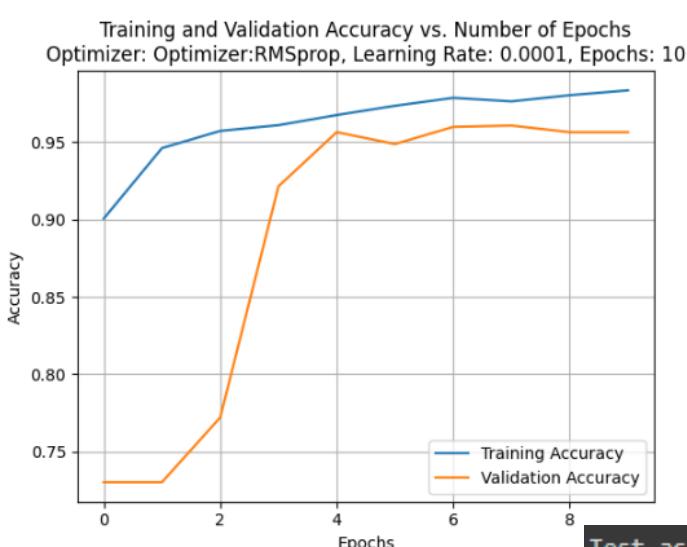


Test accuracy: 0.9564846158027649  
 Test loss: 0.12029828876256943

:RMSprop

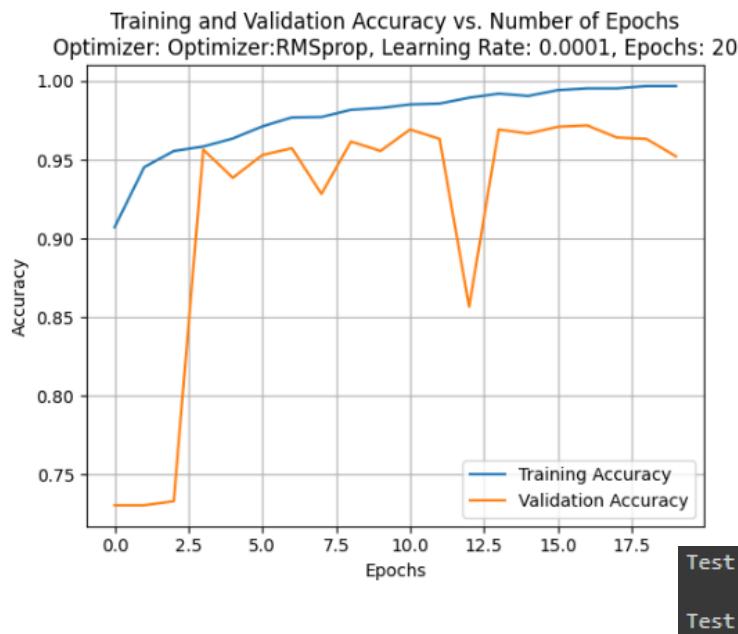
אלגוריתם זה פותח במטרה לפתור את בעיית ההתקנסות המהירה של שיעור הלמידה כלפי מטה.  
 שיטה זו מתגברת על בעיית ההתקנסות המהירה באמצעות חישוב ממוצע משוקל של ה-*Gradient*.  
 ביריבוע, והענקת משקל גדול יותר ל-*Gradient* האחרון חשוב.

:LR=0.0001, EPOCHS=10

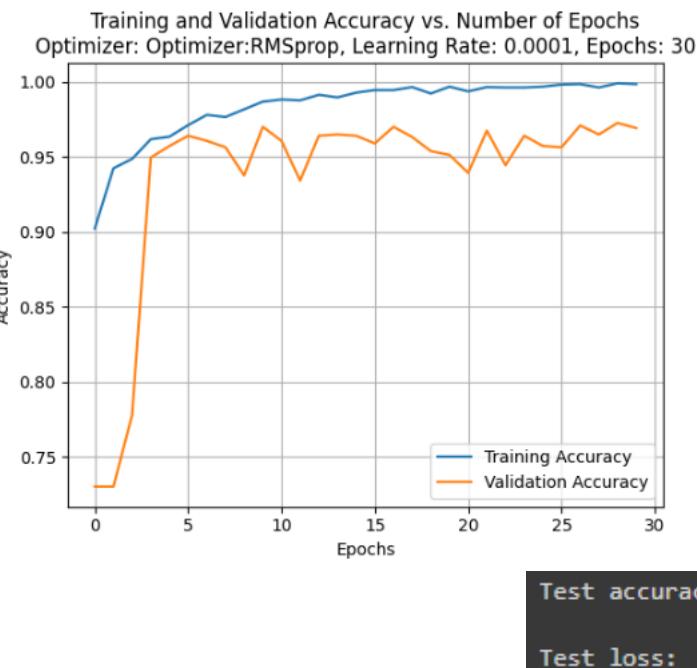
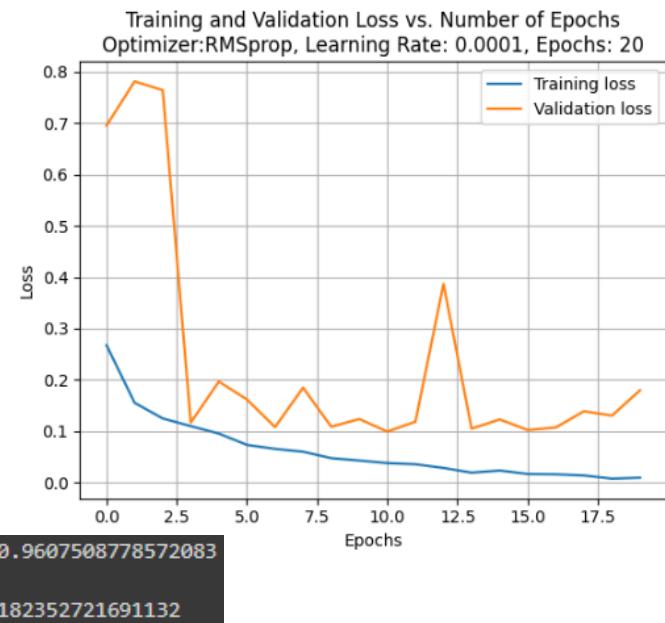


Test accuracy: 0.961604118347168

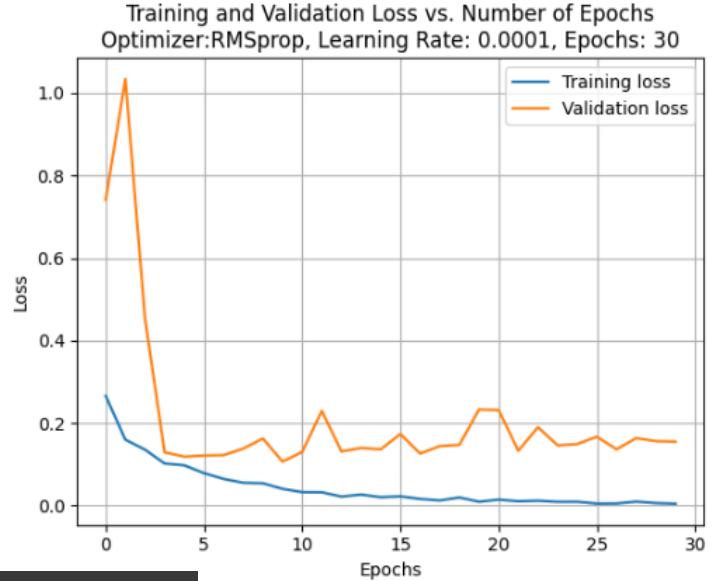
Test loss: 0.11953594535589218



:LR=0.0001, EPOCHS=20



:LR=0.0001, EPOCHS=30



**סיכום התוצאות:**

נסכם את התוצאות על סט הבדיקה בטבלאות:

SGD			
Learning Rate Epochs	0.01	0.01	0.0001
10	0.9027	0.959	0.9308
20	0.9616	0.96501	0.9394
30	0.9633	0.9624	0.9607

SGD (With Momentum = 0.9)			
Learning Rate Epochs	0.001	0.0001	0.00001
10	0.956	0.936	0.927
15	0.951	0.9488	0.94
20	0.9658	0.947	0.908

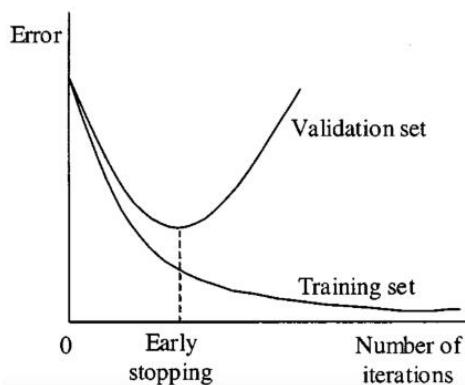
Adam			
Learning Rate Epochs	0.001	0.0001	0.00001
10	0.9402	0.953	0.947
15	0.951	0.945	0.944
20	0.887	0.956	0.956

RMSprop	
Learning Rate Epochs	0.0001
10	0.961
20	0.9607
30	0.963

התוצאה הגבוהה ביותר על סט האימון התקבלה באלגוריתם SGD עם מומנטום של 0.9 עם 20 אפוקס ו- 0.01 שיעור למידה.

נבחן את טיב התוצאות שקיבלו עבור אלגוריתם זה עם מגנון עצירה מוקדם – Early Stopping

מגנון עצירה מוקדם של אימון כאשר שנובע מכך שהפרמטרים לא מתעדכנים מספיק, בהתאם לארוגומנטים שהוכנסו למגנון העצירה.



הารוגומנטים שהוכנסו:

monitor='val\_loss', patience=5 ,mode='min', restore\_best\_weights=True

**'monitor='val\_loss'**: הפונקציה אותה נרצה לבדוק ולחקר במנגנון העצירה, במקרה שלנו פונקציית הנקס של סט הולידציה.

**5=patience**: מייצג את מספר האפוקס עبورם לא יהיה שיפור משמעותי ואחריהם האימון ייעצר. במקרה שלנו, עבור 5 אפוקס אם לא היה הנחתה בקס הולידציה - האימון ייעצר.

**'mode='min'**: ארגומנט הקובע האם המוניטור אמר לגודל או לפחות כלומר, להגיע למינימום או מקסימום.

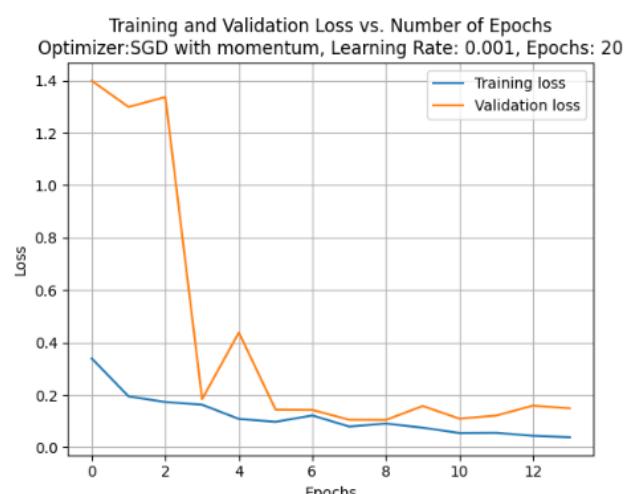
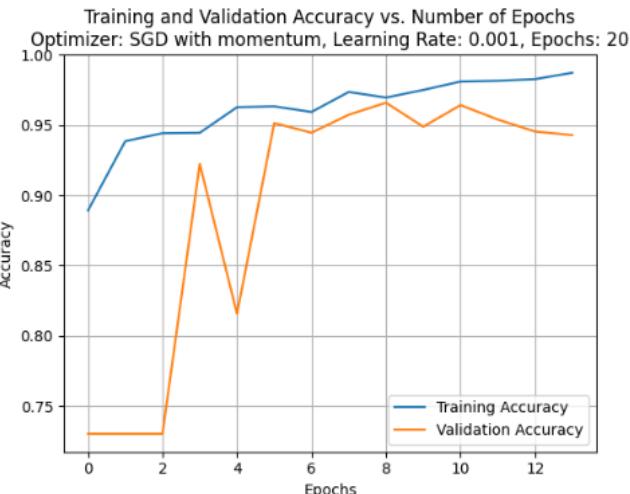
**'restore best weights=True'**: האם לשמר את ה-"משכילות" מהאפוקס עם הערכים הטובים ביותר שייצאו במהלך האימון. במקרה שלנו, ישמר הערך המינימלי של פונקציית הנקס של סט הולידציה שהתקבל במהלך האימון.

ניתן לראות שעבור אפוקס 14 התקיים מנגנון העצירה המוקדמת.

```

Epoch 1/20
176/176 [=====] - 1s 49ms/step - loss: 0.3391 - accuracy: 0.8892 - val_loss: 1.3990 - val_accuracy: 0.7304
Epoch 2/20
176/176 [=====] - 8s 46ms/step - loss: 0.1944 - accuracy: 0.9385 - val_loss: 1.2993 - val_accuracy: 0.7304
Epoch 3/20
176/176 [=====] - 8s 44ms/step - loss: 0.1730 - accuracy: 0.9442 - val_loss: 1.3371 - val_accuracy: 0.7304
Epoch 4/20
176/176 [=====] - 8s 45ms/step - loss: 0.1628 - accuracy: 0.9445 - val_loss: 0.1840 - val_accuracy: 0.9224
Epoch 5/20
176/176 [=====] - 8s 45ms/step - loss: 0.1884 - accuracy: 0.9627 - val_loss: 0.4383 - val_accuracy: 0.8157
Epoch 6/20
176/176 [=====] - 8s 43ms/step - loss: 0.0974 - accuracy: 0.9633 - val_loss: 0.1438 - val_accuracy: 0.9514
Epoch 7/20
176/176 [=====] - 9s 48ms/step - loss: 0.1218 - accuracy: 0.9593 - val_loss: 0.1428 - val_accuracy: 0.9445
Epoch 8/20
176/176 [=====] - 8s 44ms/step - loss: 0.0794 - accuracy: 0.9735 - val_loss: 0.1052 - val_accuracy: 0.9573
Epoch 9/20
176/176 [=====] - 8s 44ms/step - loss: 0.0907 - accuracy: 0.9695 - val_loss: 0.1046 - val_accuracy: 0.9659
Epoch 10/20
176/176 [=====] - 8s 46ms/step - loss: 0.0751 - accuracy: 0.9749 - val_loss: 0.1579 - val_accuracy: 0.9488
Epoch 11/20
176/176 [=====] - 8s 45ms/step - loss: 0.0547 - accuracy: 0.9809 - val_loss: 0.1095 - val_accuracy: 0.9642
Epoch 12/20
176/176 [=====] - 8s 45ms/step - loss: 0.0554 - accuracy: 0.9815 - val_loss: 0.1214 - val_accuracy: 0.9539
Epoch 13/20
176/176 [=====] - 8s 44ms/step - loss: 0.0441 - accuracy: 0.9826 - val_loss: 0.1592 - val_accuracy: 0.9454
Epoch 14/20
175/176 [=====] - ETA: 0s - loss: 0.0387 - accuracy: 0.9871Restoring model weights from the end of the best epoch: 9.
176/176 [=====] - 8s 46ms/step - loss: 0.0385 - accuracy: 0.9872 - val_loss: 0.1490 - val_accuracy: 0.9428
Epoch 14: early stopping
37/37 [=====] - 1s 14ms/step - loss: 0.1234 - accuracy: 0.9590

```



Test accuracy: 0.9590443968772888  
 Test loss: 0.12335684150457382

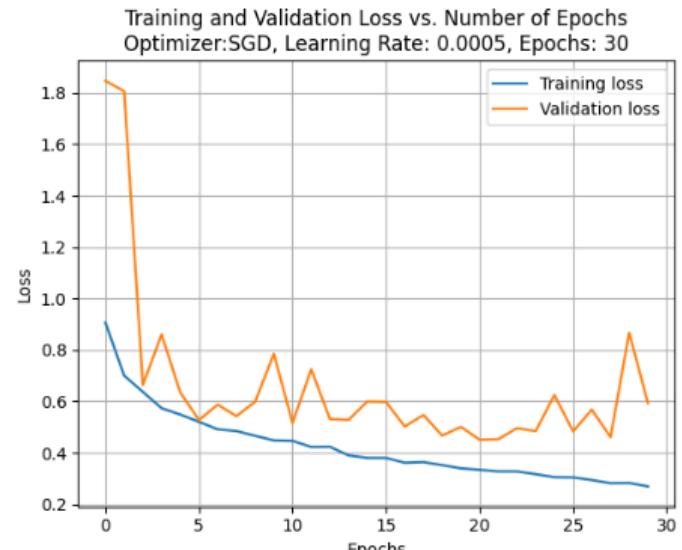
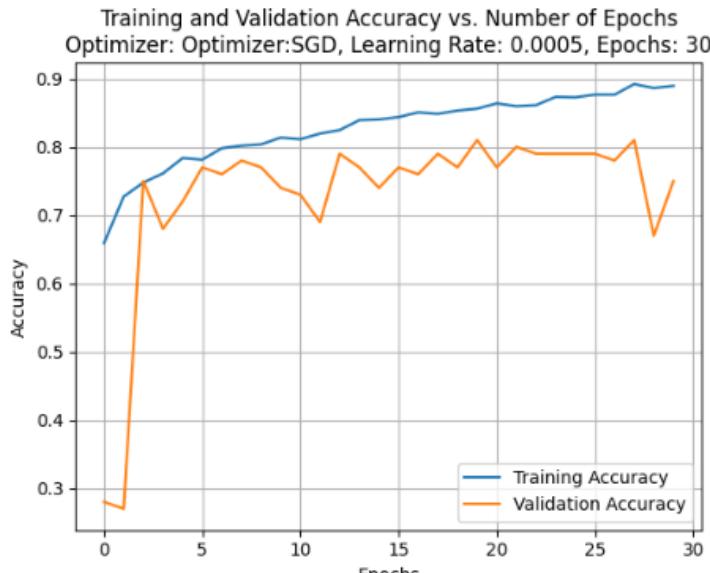
ניתן לראות כי לא הושג שיפור עבור מנגנון העצירה המוקדמת באlgorigthm SGD עם מומנטום 0.9.

### פתרון משימה IV:

בסעיף זה, השתמשנו ב-30 ו-60 אפוקס על כל שיעור למידה של 0.00001 ו-0.0005 עם של Batch-size 20.

:SGD

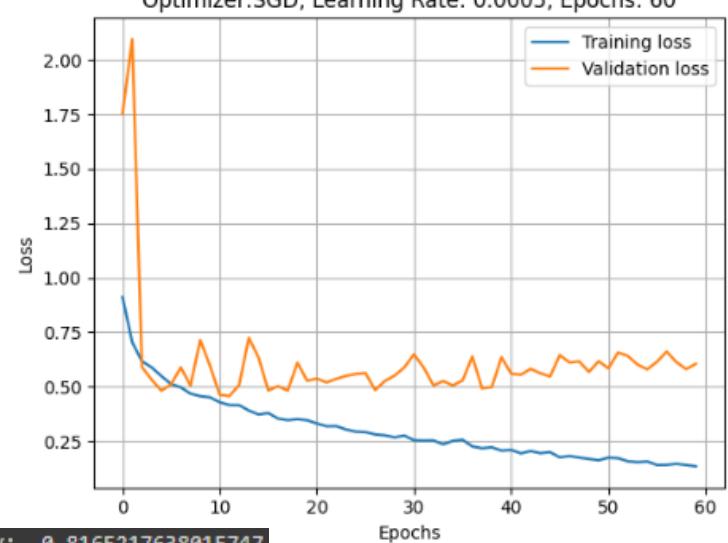
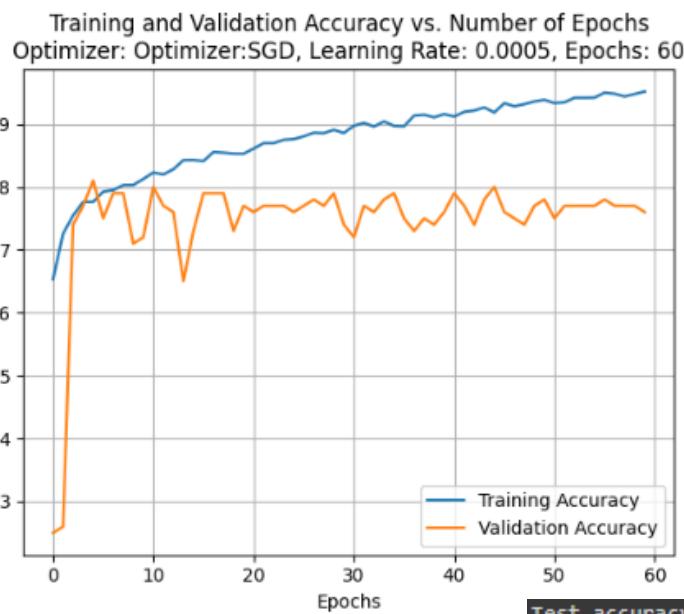
:LR=0.0005, EPOCHS=30



Test accuracy: 0.7991304397583008

Test loss: 0.4760284721851349

:LR=0.0005, EPOCHS=60

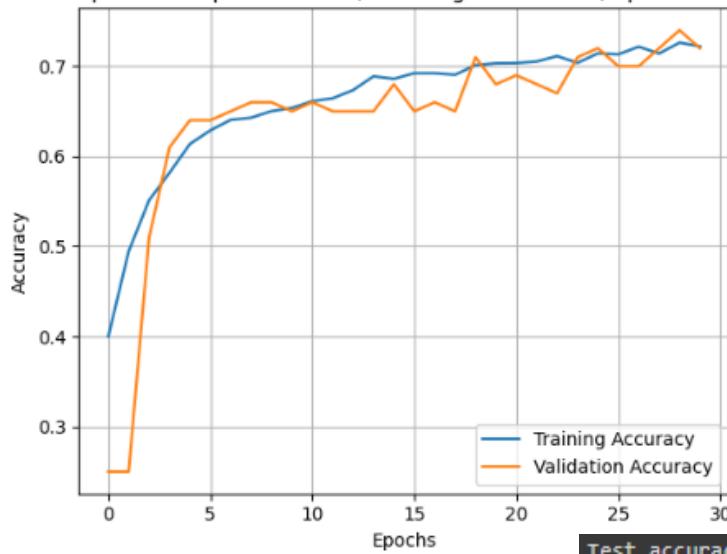


Test accuracy: 0.8165217638015747

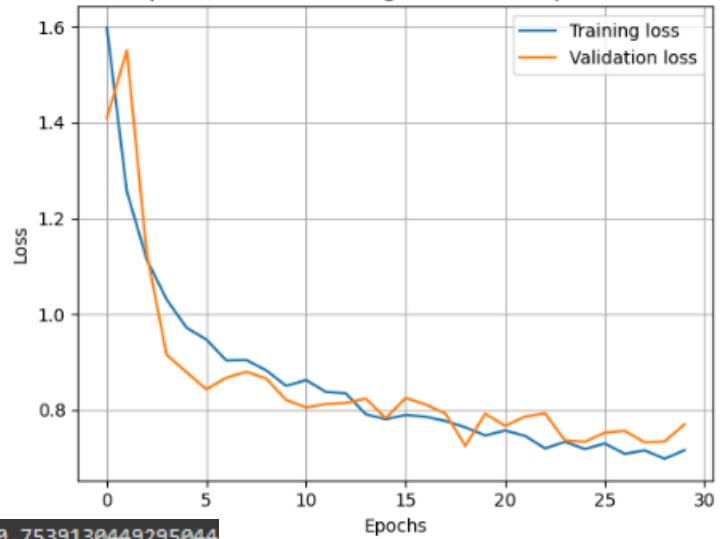
Test loss: 0.4931243062019348

:LR=0.00001, EPOCHS=30

Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:SGD, Learning Rate: 1e-05, Epochs: 30



Training and Validation Loss vs. Number of Epochs  
Optimizer:SGD, Learning Rate: 1e-05, Epochs: 30

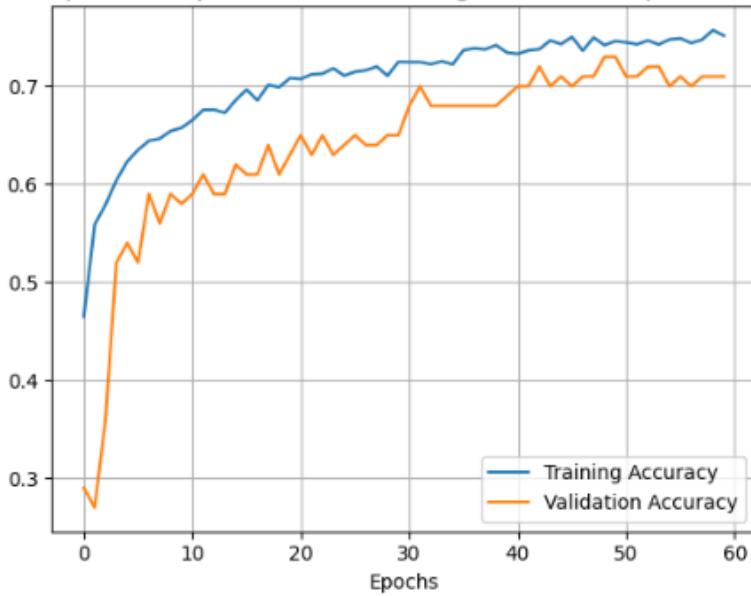


Test accuracy: 0.7539130449295044

Test loss: 0.5845428109169006

Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:SGD, Learning Rate: 1e-05, Epochs: 60

Accuracy

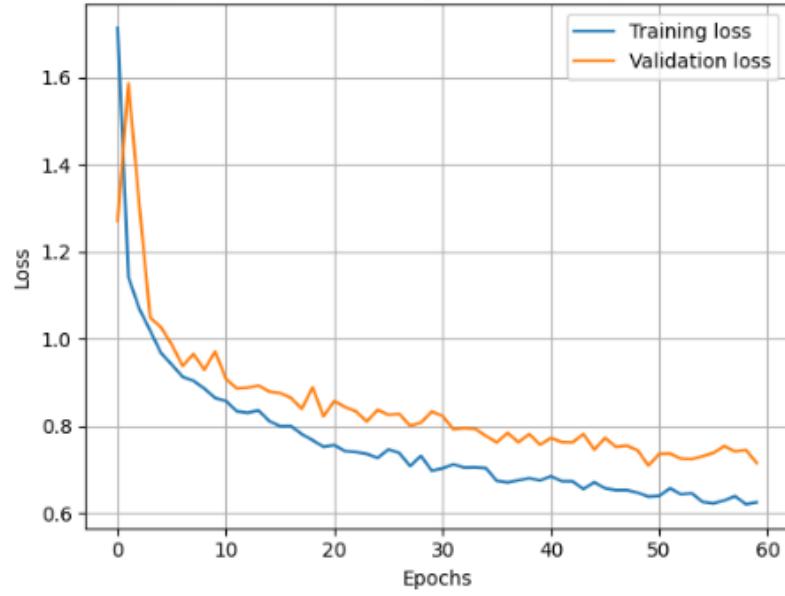


Test accuracy: 0.7626087069511414

Test loss: 0.5691484212875366

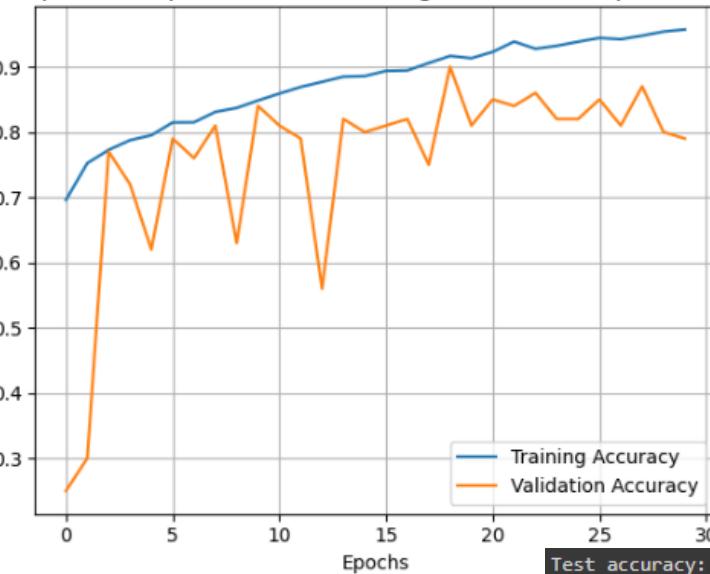
**:LR=0.00001, EPOCHS=60**

Training and Validation Loss vs. Number of Epochs  
Optimizer:SGD, Learning Rate: 1e-05, Epochs: 60



Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:SGD, Learning Rate: 0.0005, Epochs: 30

Accuracy



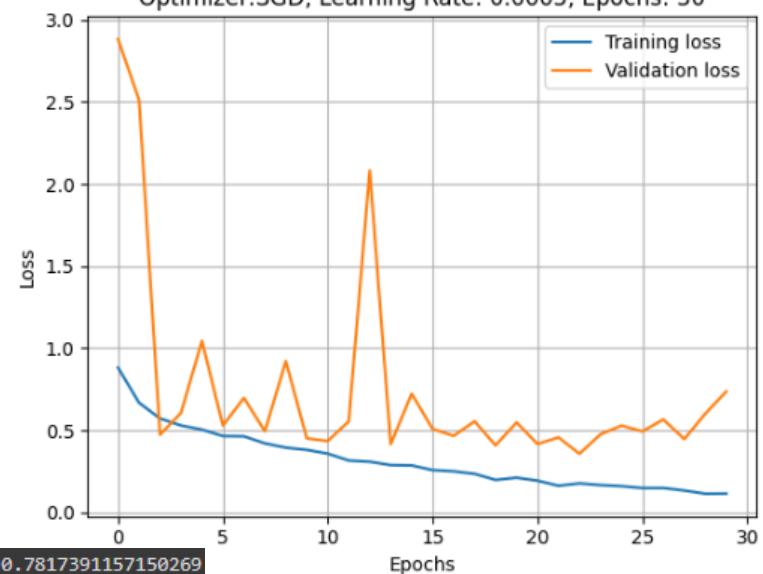
Test accuracy: 0.7817391157150269

Test loss: 0.8541609048843384

**:SGD, With Momentum=0.9**

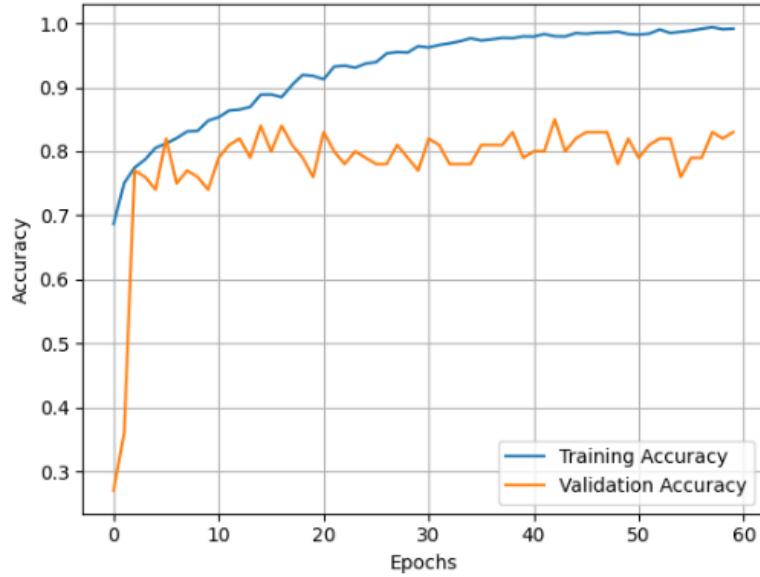
**:LR=0.0005, EPOCHS=30**

Training and Validation Loss vs. Number of Epochs  
Optimizer:SGD, Learning Rate: 0.0005, Epochs: 30



**:LR=0.0005, EPOCHS=60**

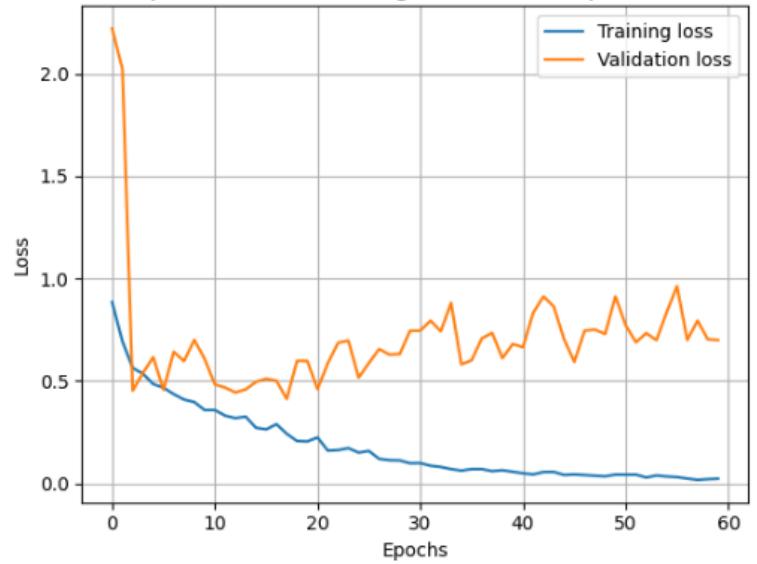
Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:SGD, Learning Rate: 0.0005, Epochs: 60



Test accuracy: 0.7895652055740356

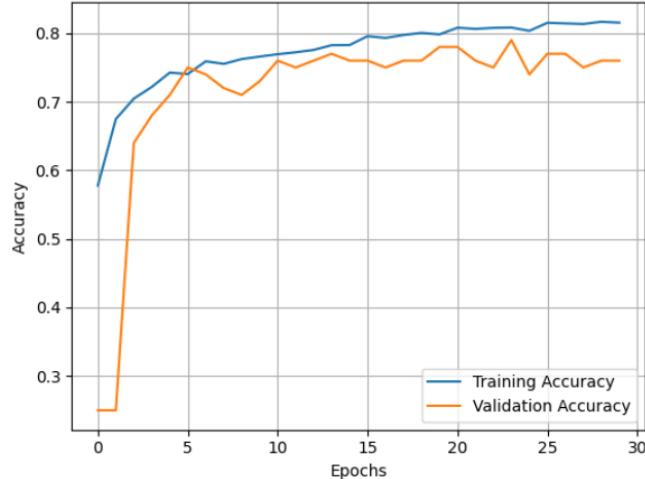
Test loss: 0.8848826289176941

Training and Validation Loss vs. Number of Epochs  
Optimizer:SGD, Learning Rate: 0.0005, Epochs: 60



**:LR=0.00001, EPOCHS=30**

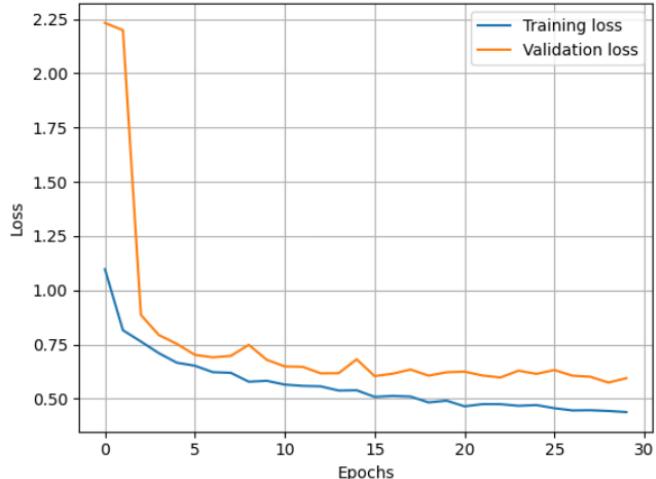
Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:SGD, Learning Rate: 1e-05, Epochs: 30



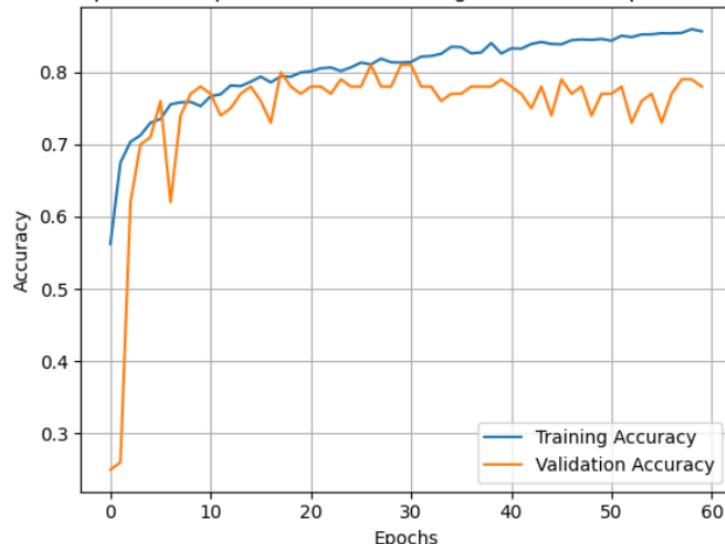
Test accuracy: 0.7947825789451599

Test loss: 0.47666120529174805

Training and Validation Loss vs. Number of Epochs  
Optimizer:SGD, Learning Rate: 1e-05, Epochs: 30

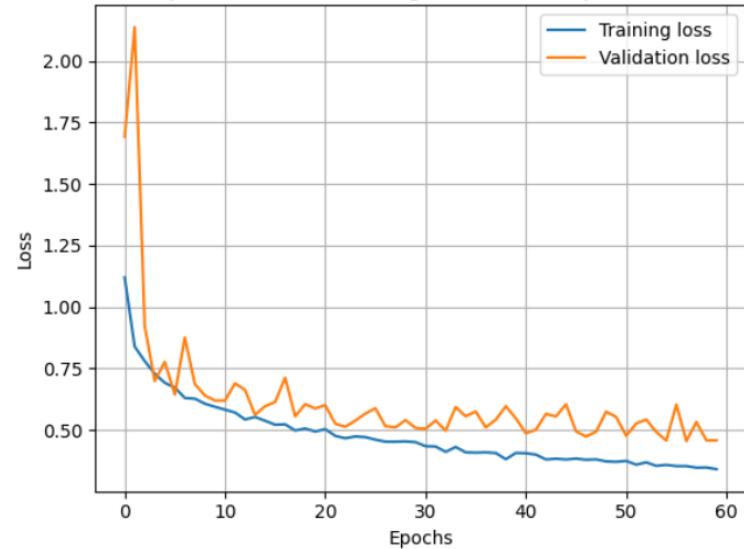


Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:SGD, Learning Rate: 1e-05, Epochs: 60



**:LR=0.00001, EPOCHS=60**

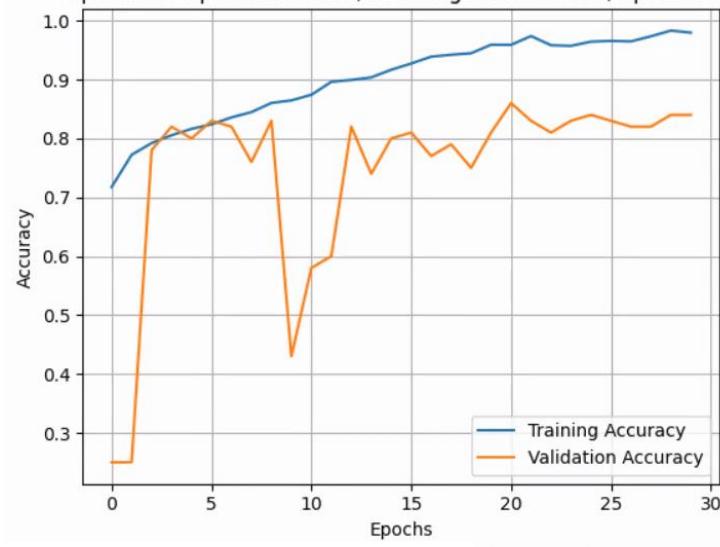
Training and Validation Loss vs. Number of Epochs  
Optimizer:SGD, Learning Rate: 1e-05, Epochs: 60



Test accuracy: 0.7808695435523987

Test loss: 0.5194790959358215

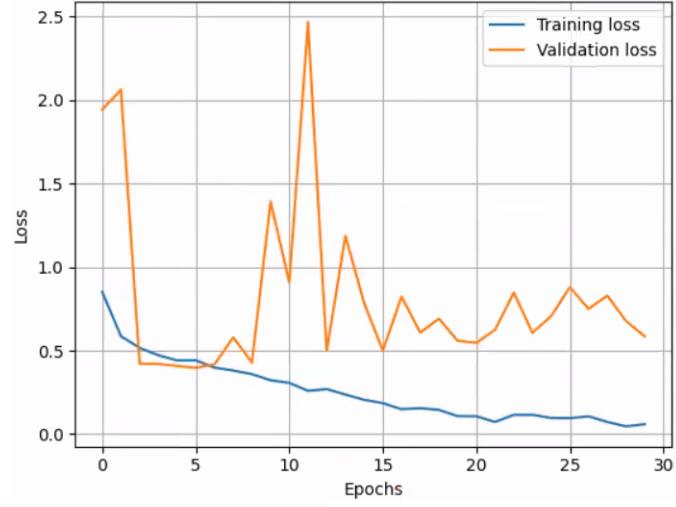
Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:Adam, Learning Rate: 0.0005, Epochs: 30



**:Adam**

**:LR=0.0005, EPOCHS=30**

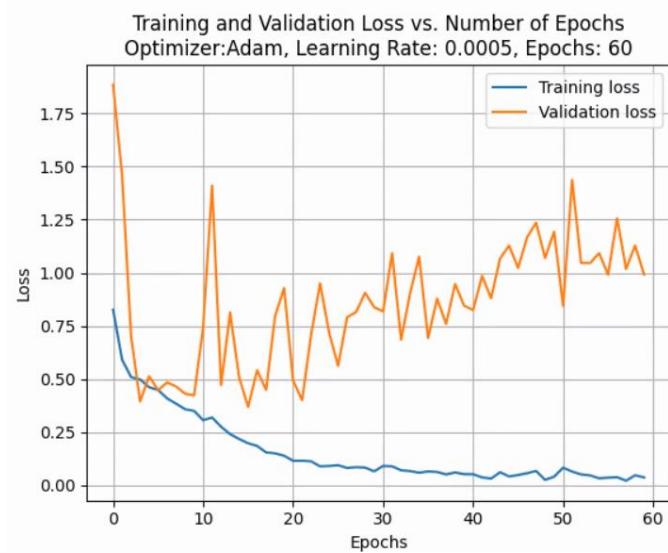
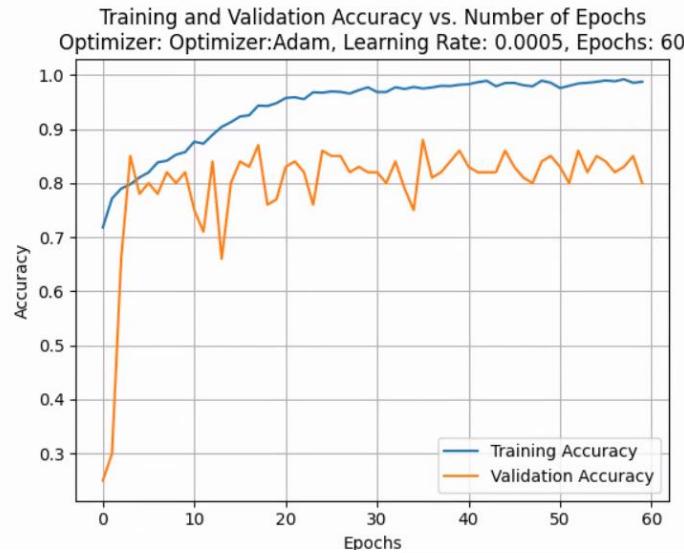
Training and Validation Loss vs. Number of Epochs  
Optimizer:Adam, Learning Rate: 0.0005, Epochs: 30



Test accuracy: 0.791304349899292

Test loss: 0.8670274019241333

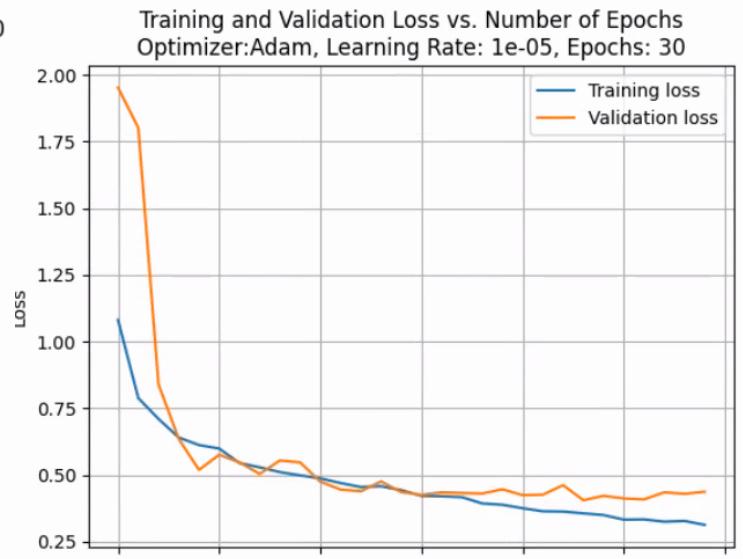
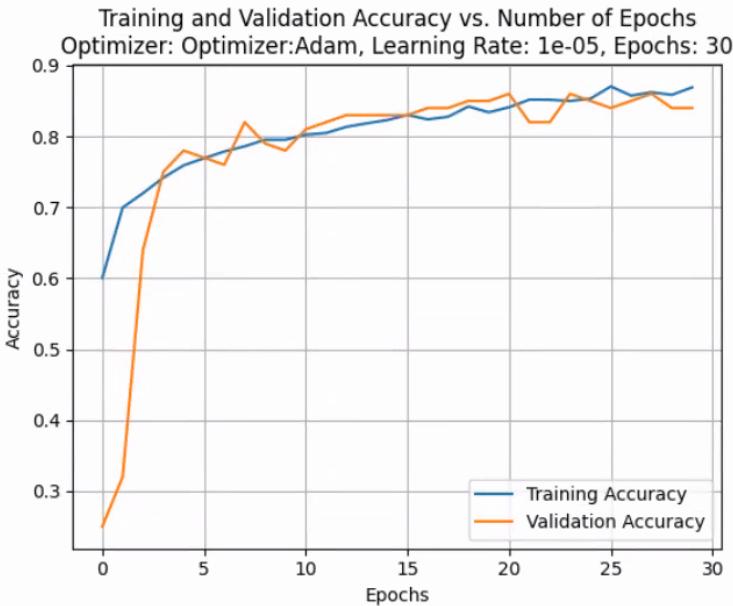
**:LR=0.0005, EPOCHS=60**



Test accuracy: 0.7965217232704163

Test loss: 0.9980506896972656

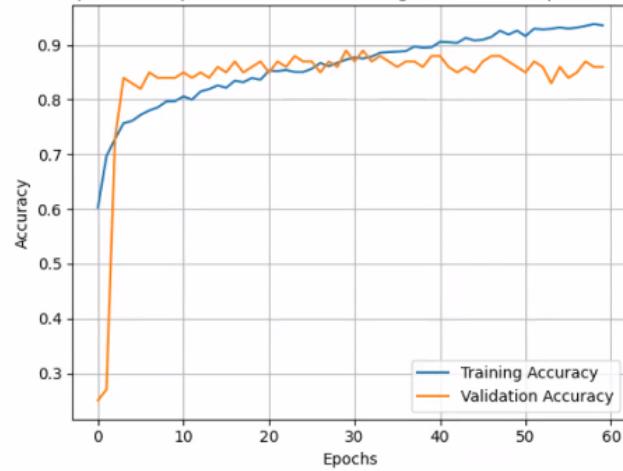
**:LR=0.00001, EPOCHS=30**



Test accuracy: 0.7904347777366638

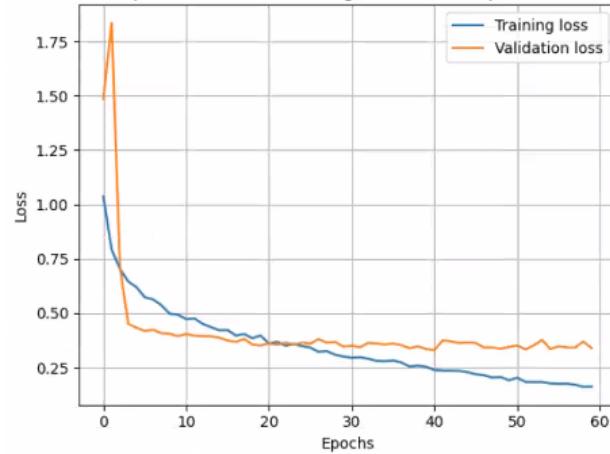
Test loss: 0.5100498795509338

Training and Validation Accuracy vs. Number of Epochs  
 Optimizer: Optimizer:Adam, Learning Rate: 1e-05, Epochs: 60



:LR=0.00001, EPOCHS=60

Training and Validation Loss vs. Number of Epochs  
 Optimizer: Adam, Learning Rate: 1e-05, Epochs: 60



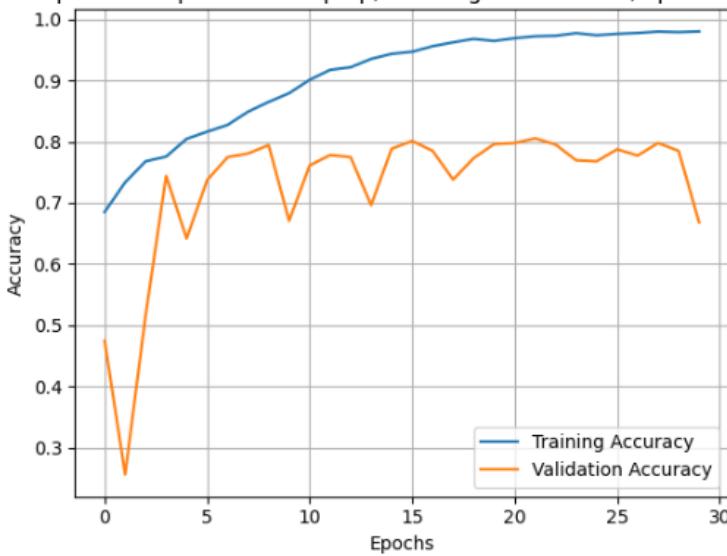
Test accuracy: 0.8086956739425659

Test loss: 0.5014658570289612

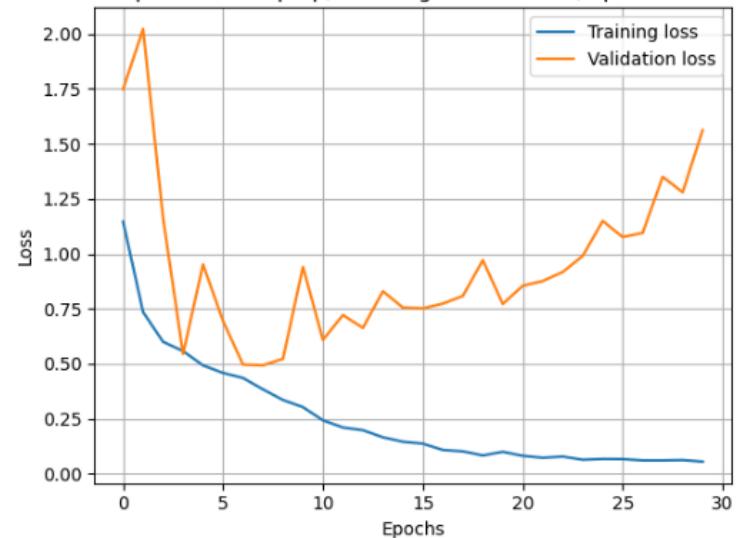
:RMSprop

:LR=0.0005, EPOCHS=30

Training and Validation Accuracy vs. Number of Epochs  
 Optimizer: Optimizer:RMSprop, Learning Rate: 0.0005, Epochs: 30



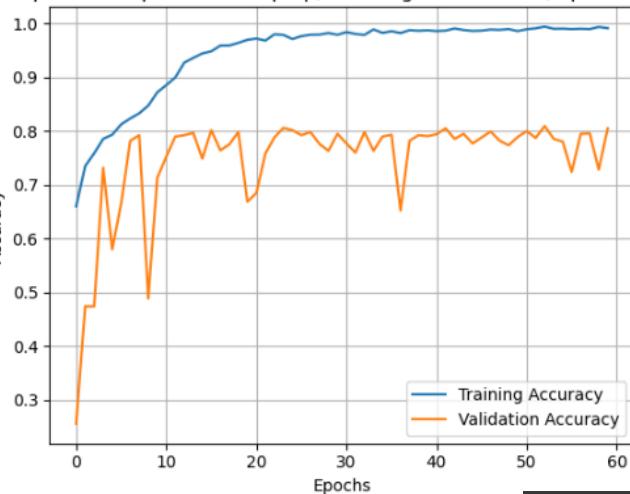
Training and Validation Loss vs. Number of Epochs  
 Optimizer: RMSprop, Learning Rate: 0.0005, Epochs: 30



Test accuracy: 0.6825938820838928

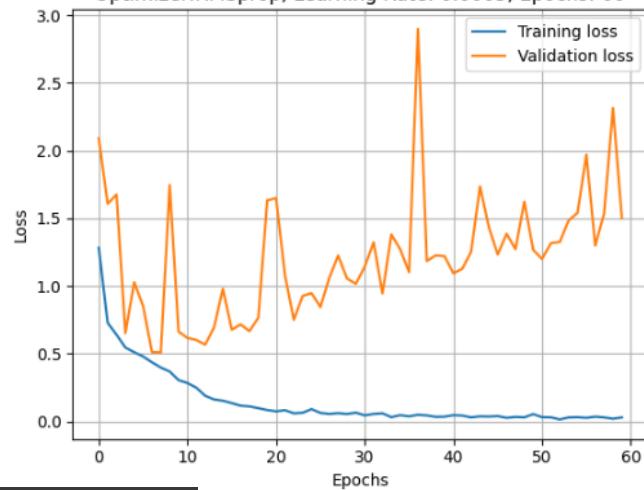
Test loss: 1.4811924695968628

Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:RMSprop, Learning Rate: 0.0005, Epochs: 60



**:LR=0.0005, EPOCHS=60**

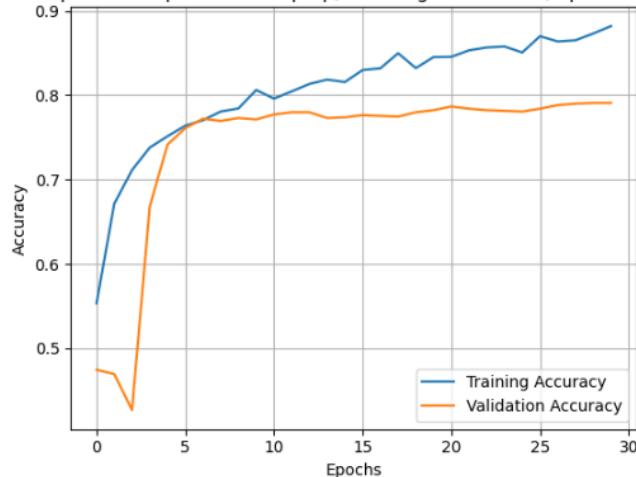
Training and Validation Loss vs. Number of Epochs  
Optimizer:RMSprop, Learning Rate: 0.0005, Epochs: 60



Test accuracy: 0.7977815866470337

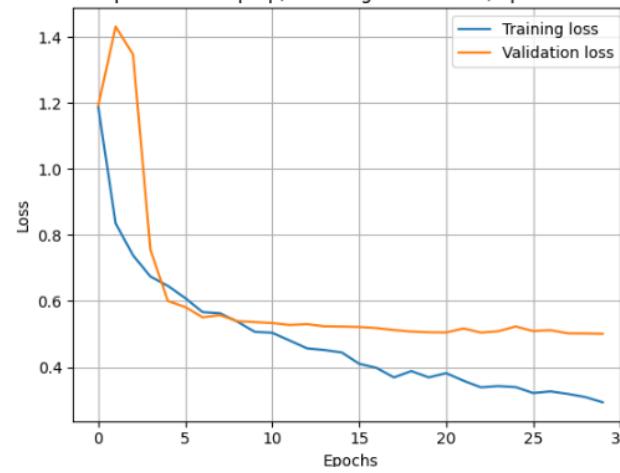
Test loss: 1.546295404434204

Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:RMSprop, Learning Rate: 1e-05, Epochs: 30



**:LR=0.00001, EPOCHS=30**

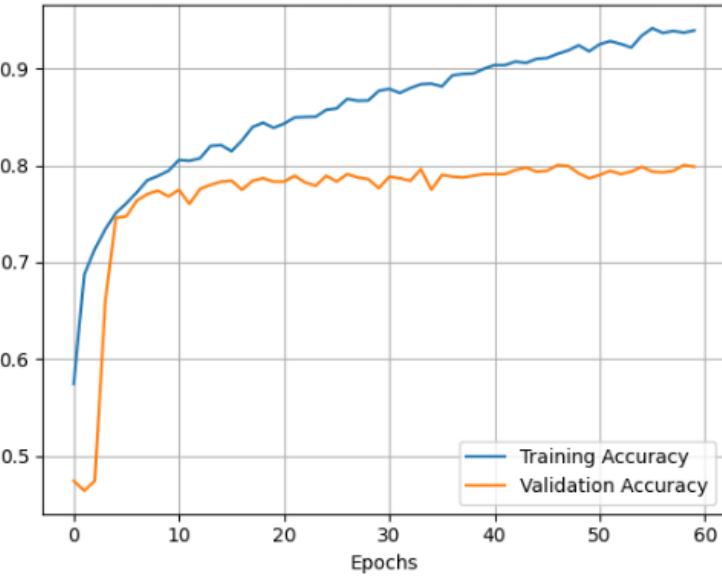
Training and Validation Loss vs. Number of Epochs  
Optimizer:RMSprop, Learning Rate: 1e-05, Epochs: 30



Test accuracy: 0.8046075105667114

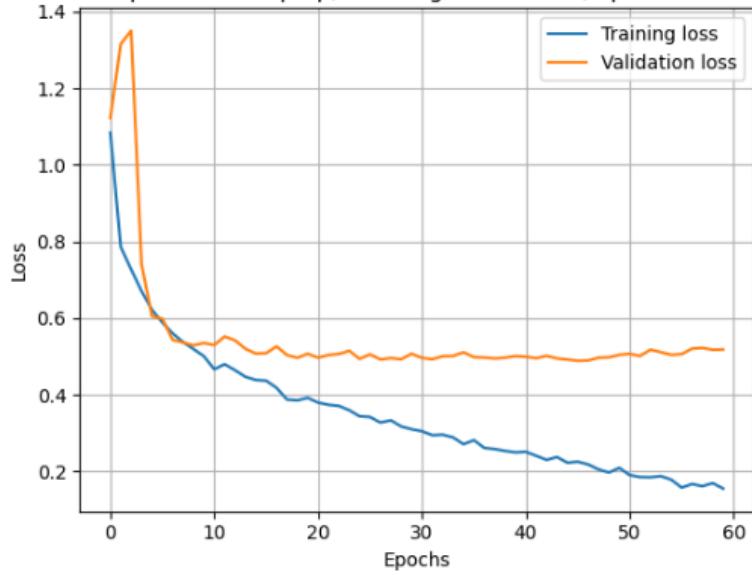
Test loss: 0.4837980270385742

Training and Validation Accuracy vs. Number of Epochs  
Optimizer: Optimizer:RMSprop, Learning Rate: 1e-05, Epochs: 60



:LR=0.00001, EPOCHS=60

Training and Validation Loss vs. Number of Epochs  
Optimizer:RMSprop, Learning Rate: 1e-05, Epochs: 60



Test accuracy: 0.8105801939964294

Test loss: 0.49024057388305664

#### סיכום התוצאות:

נסכם את התוצאות על סט הבדיקה בטבלאות:

SGD		
Learning Rate	0.0005	0.00001
Epochs		
30	0.7991	0.7539
60	0.8165	0.762

SGD (With Momentum = 0.9)		
Learning Rate	0.0005	0.00001
Epochs		
30	0.7817	0.794
60	0.789	0.78

RMSprop		
Learning Rate	0.0005	0.00001
Epochs		
30	0.682	0.804
60	0.7977	0.8105

Adam		
Learning Rate	0.0005	0.00001
Epochs		
30	0.7913	0.7904
60	0.7965	0.8086

ניתן לראות כי עבור האלגוריתם SGD עם 0.0005 שיעור למידה ו-60 אפוקס קיבלנו את התוצאה הגבוהה ביותר על סט הבדיקה.

ניתן לראות כי ישנה ירידת גודלה בBITS回首ים של האלגוריתמים ביחס למשימות הקודמות. הדבר יכול לנבוע מכך שכעת אנחנו מסוגים ל-3 קטגוריות שונות (ולא ל-2 כמו במשימות הקודמות) כך שכעת נדרש סט נתונים גדול יותר.

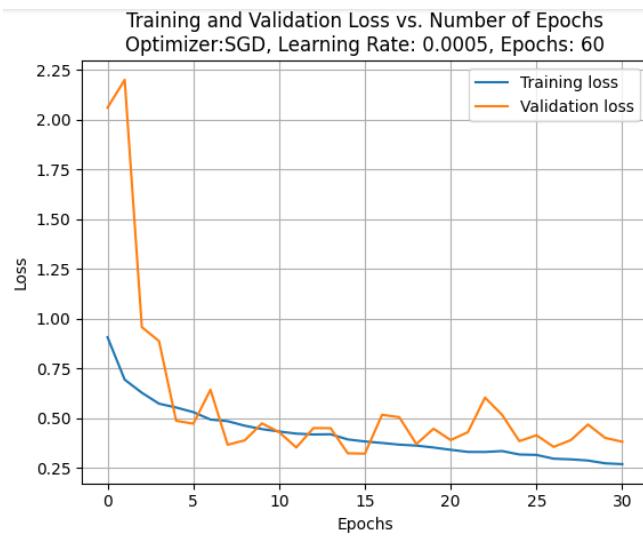
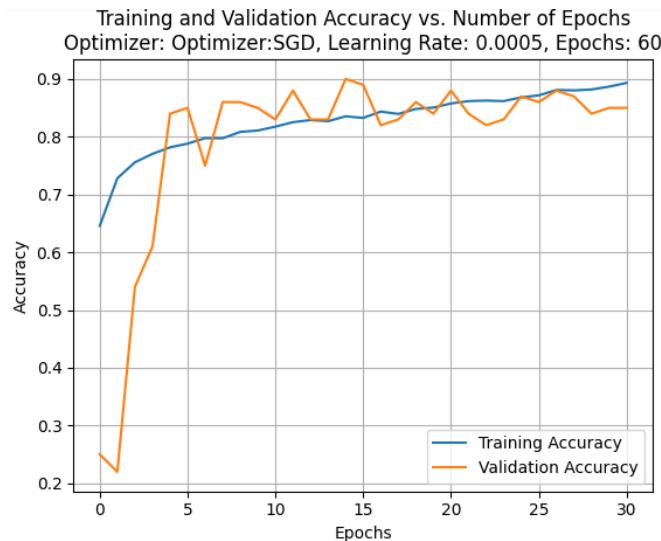
פתרון אפשרי לכך הוא שנייתו להשתמש בעשרות הנתונים שלנו (Data Augmentation) על ידי BITS回首 ערייה לתמונות (זום אין או א웃, סיבוב התמונה וכדומה) ובכך ליצור תמונה חדשה שהמודל יצטרך לעבד.

כמו כן, ניתן להוסיף פרמטרים נוספים לאלגוריתם שלנו (Decay, momentum, Nesterov) וכדומה במטרה לשפר את אחוז הדיק בסט הבדיקה.

בנוסף, ניתן להעמיק את הרשות שבנו על ידי הוספה שכבות אליה, דבר אשר יכול לעזור בזיהוי מדויק יותר של תכונות הקיימות בתמונות.

نبצע CUTOT את מנגנון העצירה המוקדם עבור האלגוריתם SGD עם 0.0005 שיעור למידה ו-60 אפוקס:

```
Epoch 28/60
232/232 [=====] - 10s 42ms/step - loss: 0.2927 - accuracy: 0.8804 - val_loss: 0.3893 - val_accuracy: 0.8700
Epoch 29/60
232/232 [=====] - 10s 43ms/step - loss: 0.2868 - accuracy: 0.8819 - val_loss: 0.4680 - val_accuracy: 0.8400
Epoch 30/60
232/232 [=====] - 10s 43ms/step - loss: 0.2727 - accuracy: 0.8868 - val_loss: 0.4004 - val_accuracy: 0.8500
Epoch 31/60
232/232 [=====] - ETA: 0s - loss: 0.2686 - accuracy: 0.8933Restoring model weights from the end of the best epoch: 16.
232/232 [=====] - 10s 43ms/step - loss: 0.2686 - accuracy: 0.8933 - val_loss: 0.3811 - val_accuracy: 0.8500
Epoch 31: early stopping
36/36 [=====] - 1s 13ms/step - loss: 0.5133 - accuracy: 0.7861
```

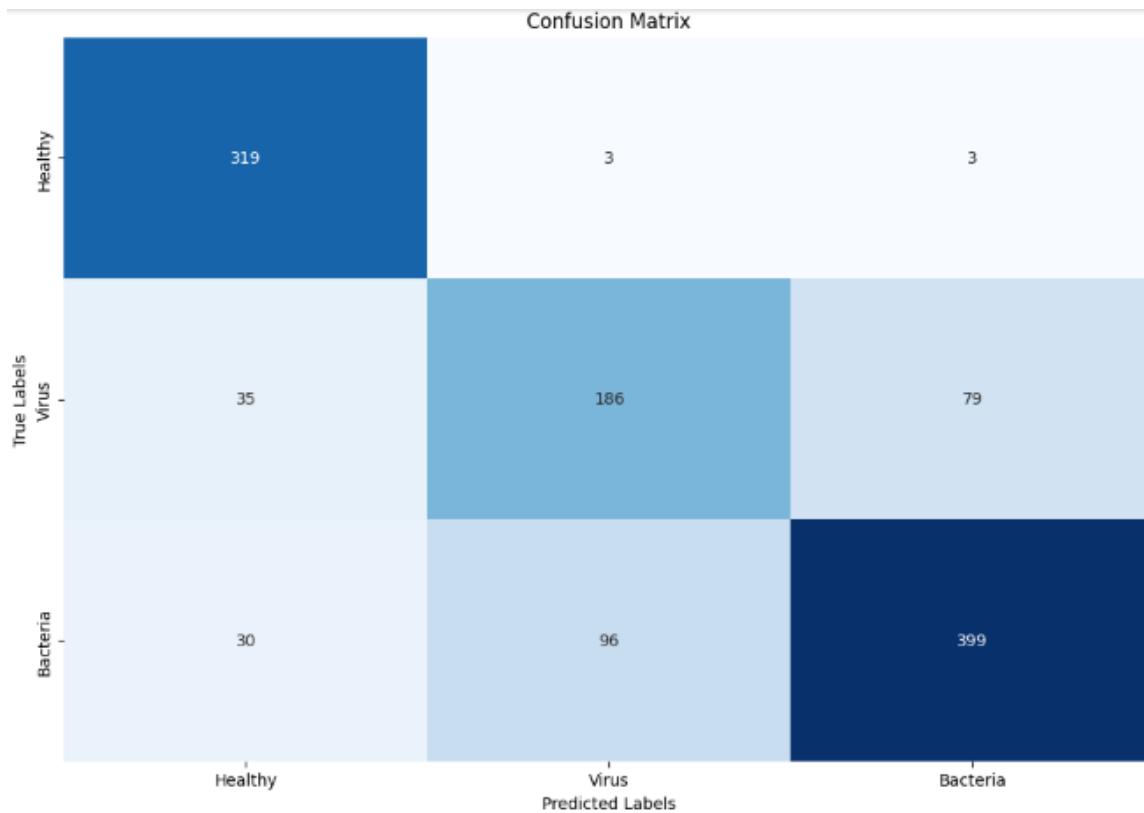


Test accuracy: 0.8260869765281677

Test loss: 0.5132984519004822

ניתן לראות כי עבור מנגנון העצירה המוקדם הביא לתוצאות טובות כמעט ביחס לאלגוריתם ללא העצירה המוקדמת.

להלן מטריצת הבלבול שיצאה עבורי הרצה זו :



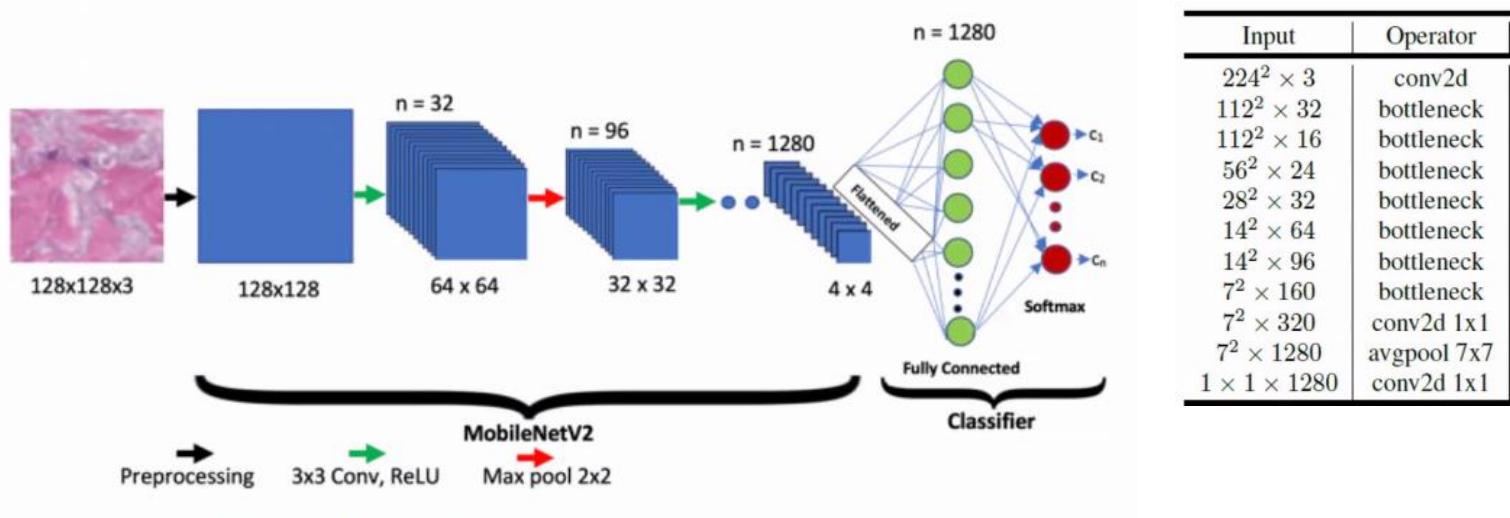
## רשות : Transfer Learning

### פתרונות שימושה I :

#### המודל:

לבנייה רשות מסווג זה השתמשנו ברשות MobileNetV2 (בה השתמשנו במעבדה).

ברשות זו קיימות השכבות הבאות (154 שכבות) :



The proposed MobileNetV2 network architecture.

נציין כי כניסה רשות יכולה להיות כל תמונה מסדר גודל כלשהו ובצבע מסוים.

את כל השכבות הכרנו בעבר חוץ משכבה-h**Bottleneck**. שכבה זו נועדה לצמצם את החישובים והזיכרון המשמשים בו ברשות. רשות זו מכילה לרוב שילוב של רשתות קוונטציוניות עם צמצום הממדים, כולל צמצום מספר ה-**Feature-maps**. שכבה זו בעצם מצמצמת את כמות הפרמטרים הנלמדים בשכבה.

לאחר מספר ניסיונות במטרה לשפר את תוצאות הרשות, הוספנו את השכבות הבאות עbor רשות בסיס :

```
Conv2D(32, (3, 3), activation='relu', padding='same')(x)
MaxPooling2D((2, 2))(x)
Conv2D(64, (3, 3), activation='relu', padding='same')(x)
MaxPooling2D((2, 2))(x)
Conv2D(128, (3, 3), activation='relu', padding='same')(x)
global_average_layer(x)
tf.keras.layers.Dropout(0.1)(x)
```

כאשר השכבה **global average layer** מחשבת את הממוצע לכל Feature-maps בכניסה כך שבמוצא שלה נקבע ערך קבוע ללא תלות בגודל הכניסה. מטרת שכבה זו הינה לצמצם את ממד ה-**Feature-maps** ואותה נשימה לשמור ולהלץ את המידע.

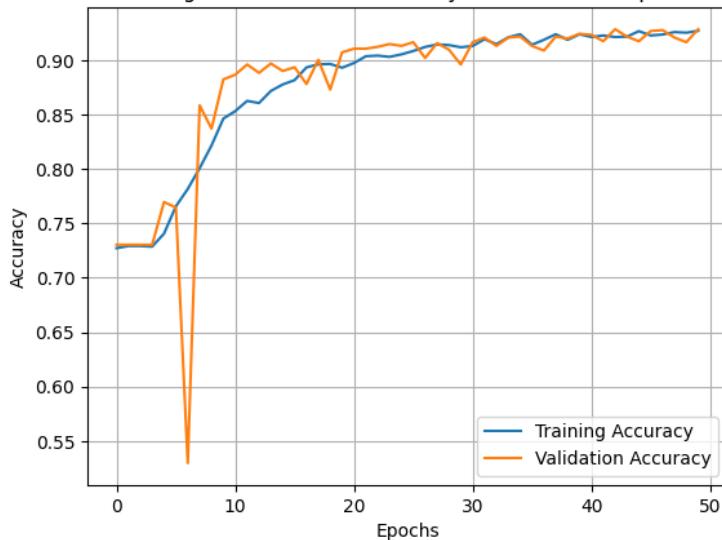
Layer (type)	Output Shape	Param #
<hr/>		
input_28 (InputLayer)	[(None, 180, 180, 3)]	0
tf.math.truediv_13 (TFOpLambda)	(None, 180, 180, 3)	0
tf.math.subtract_13 (TFOpLambda)	(None, 180, 180, 3)	0
mobilenetv2_1.00_224 (Functional)	(None, 6, 6, 1280)	2257984
conv2d_23 (Conv2D)	(None, 6, 6, 32)	368672
max_pooling2d_17 (MaxPooling2D)	(None, 3, 3, 32)	0
conv2d_24 (Conv2D)	(None, 3, 3, 64)	18496
max_pooling2d_18 (MaxPooling2D)	(None, 1, 1, 64)	0
conv2d_25 (Conv2D)	(None, 1, 1, 128)	73856
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 128)	0
dropout_10 (Dropout)	(None, 128)	0
dense_13 (Dense)	(None, 1)	129
<hr/>		
Total params: 2719137 (10.37 MB)		
Trainable params: 461153 (1.76 MB)		
Non-trainable params: 2257984 (8.61 MB)		

### פתרון משימה II :

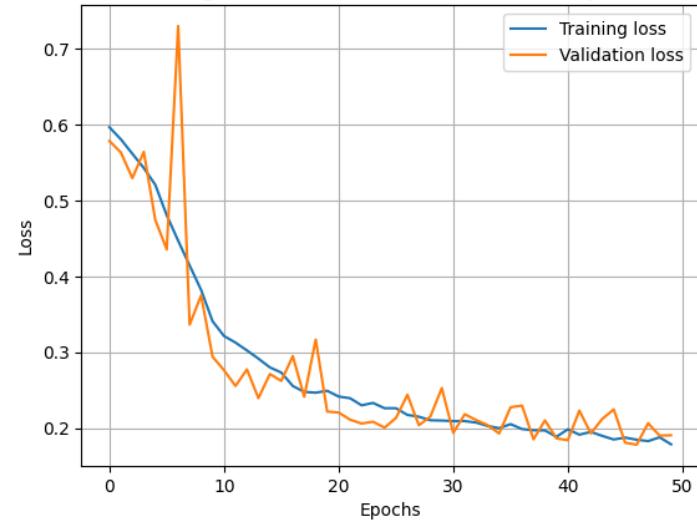
#### שכבות מוקפות:

נريץ את הרשת שמיישנו כאשר שכבות רשת הבסיס מוקפות והאלגוריתם הינו Adagrad עם שיעור למידה של  $0.01 - 1e-6$ . כמו כן, מספר האפוקס הם 50 וגודל ה- $\text{batch}$  הוא 20.

Training and Validation Accuracy vs. Number of Epochs



Training and validation loss vs. Number of Epochs

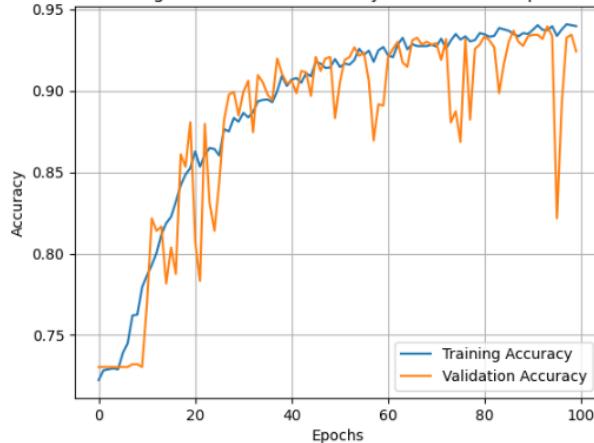


```
Test accuracy: 0.9240614175796509
Test loss: 0.188186377286911
```

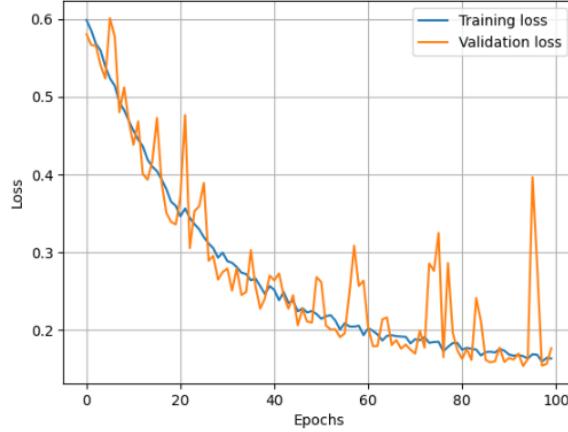
#### :Fine-Tuning

נריץ את הרשת שמיישנו כאשר 60 השכבות הראשונות של רשת הבסיס מוקפות והאלגוריתם הינו Mini-Batch Adagrad עם שיעור למידה של  $0.001 - 1e-6$ . כמו כן, מספר האפוקס הם 100 וגודל ה- $\text{batch}$  הוא 20

Training and Validation Accuracy vs. Number of Epochs



Training and validation loss vs. Number of Epochs



```
Test accuracy: 0.9180887341499329
Test loss: 0.1848318725824356
```

נשים לב כי עברו הרשות עם השכבות המוקפאות קיבלו תוצאות מעט טובות יותר.

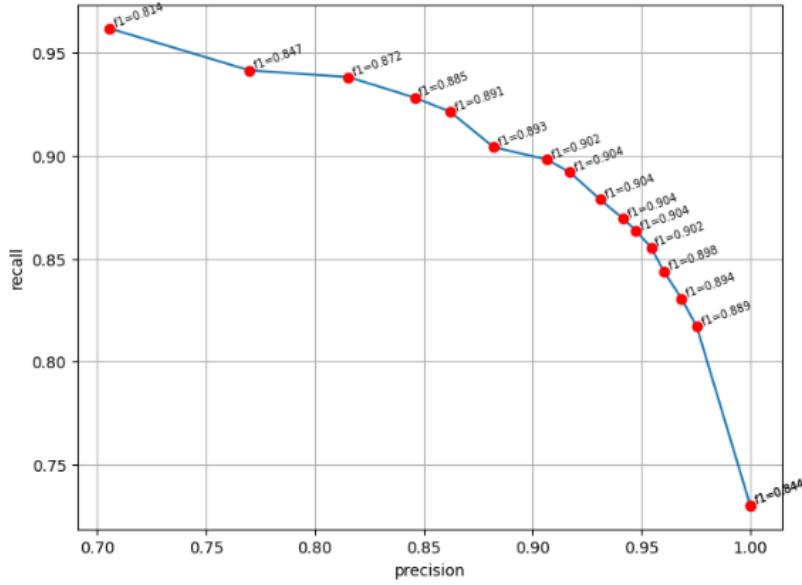
ニיצור כעת את מטሪיצת הבלבול של רשות ה-TL עם השכבות המוקפאות:

Confusion Matrix

		TN	FP	
		Healthy		
True Labels	Healthy	206	110	$Recall = \frac{TP}{TP + FN} = \frac{797}{797 + 59} \cdot 100\% = 93.1\%$
	Sick	59	797	$Precision = \frac{TP}{TP + FP} = \frac{797}{797 + 110} \cdot 100\% = 87.8\%$
		Healthy	Sick	
		Predicted Labels		

ニיצור כעת את גף Precision-Recall

Precision-Recall Curve



ניתן לראות כי עברו הסתברות סף של 0.45 (כלומר אם בМОצה הרשות התקבלה הסתברות גבוהה יותר אזី הצלום יסוווג כחולה) עם יחס של 0.904.

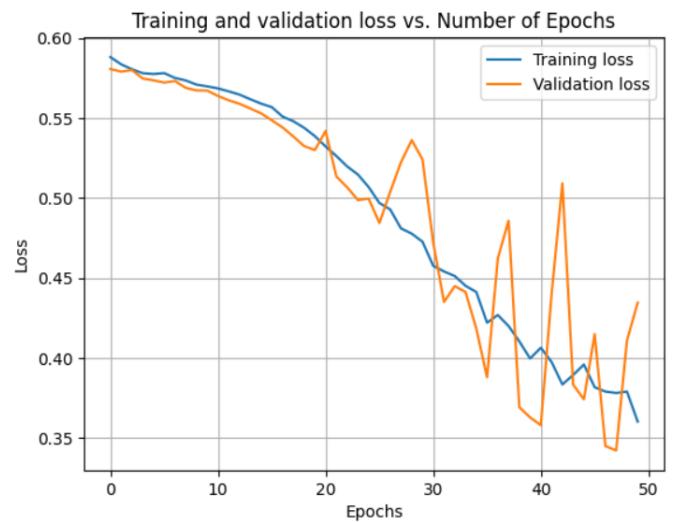
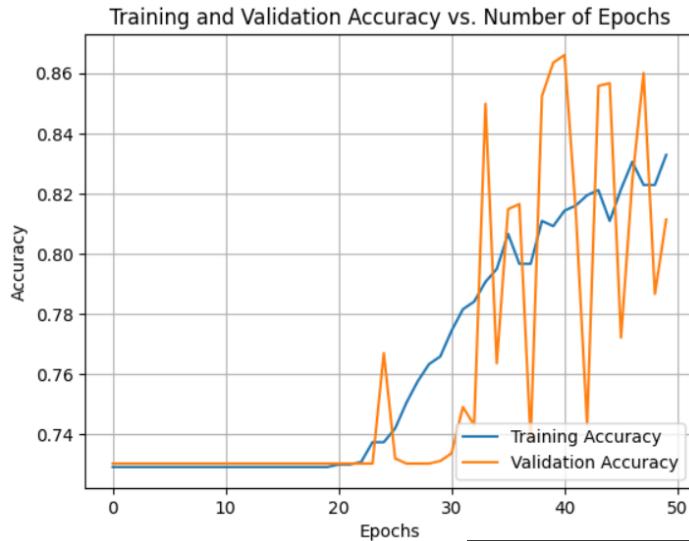
נבחר את הרשות שהניבה את התוצאות הטובות ביותר – רשות TL עם השכבות המוקפאות.

### פתרון משימה III:

משימה זו, הרצנו על שיעור למידה של 1.000 עם אפוקס 50 ו- 80 ו- Mini-Batches

SGD

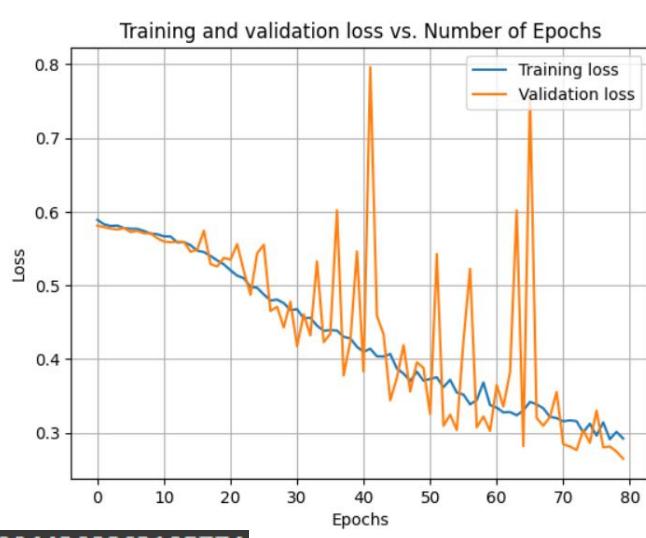
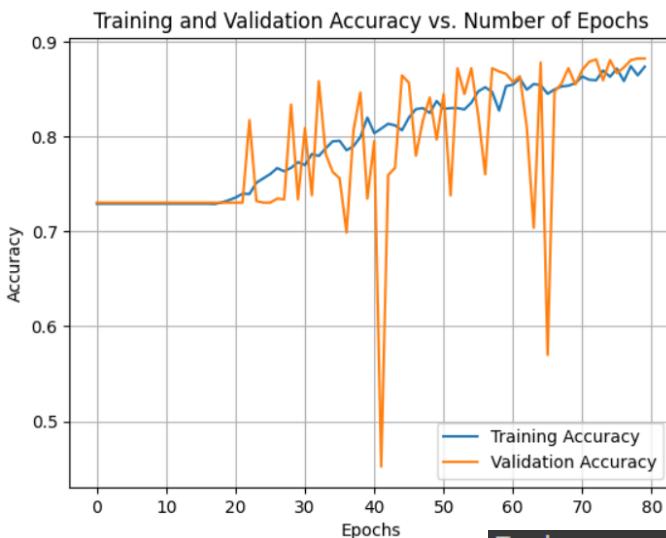
LR=0.001, EPOCHS=50



Test accuracy: 0.8370307087898254

Test loss: 0.40495315194129944

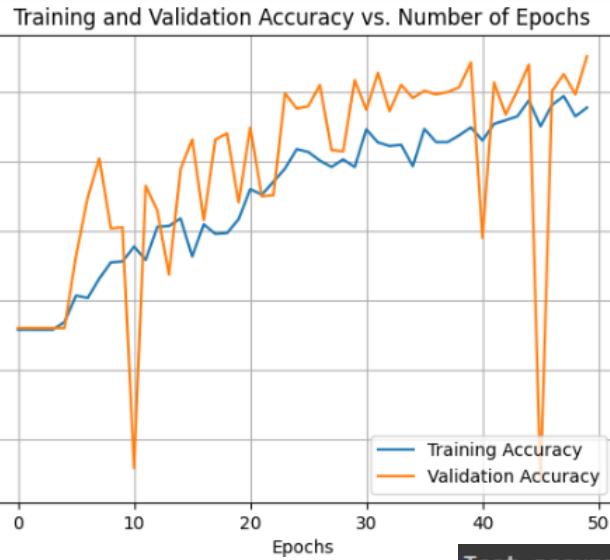
LR=0.001, EPOCHS=80



Test accuracy: 0.9044368863105774

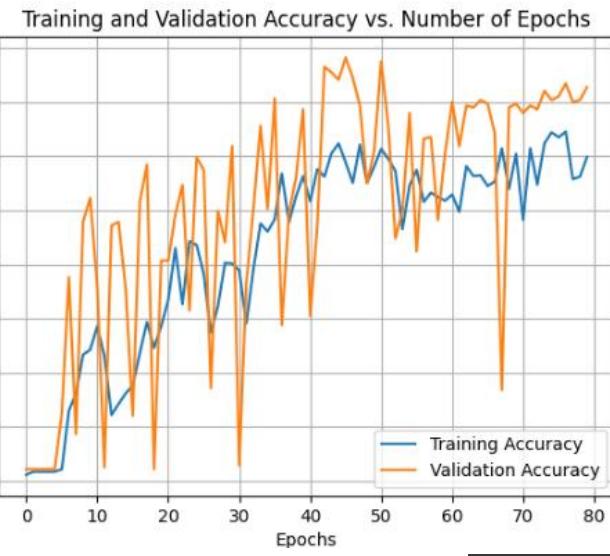
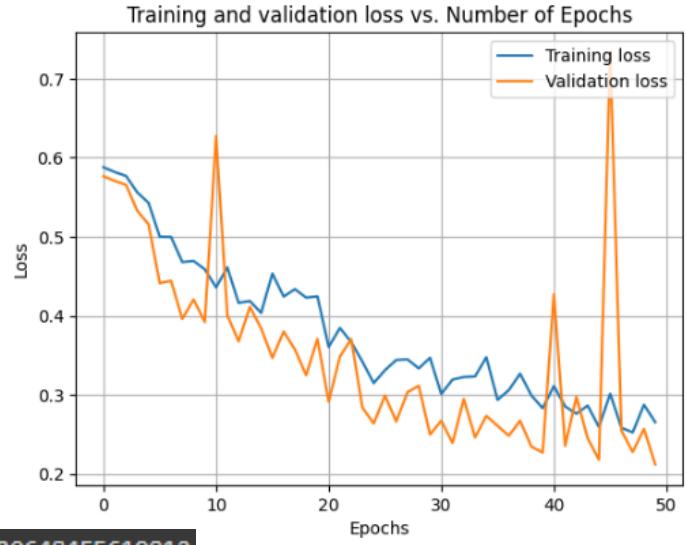
Test loss: 0.24288317561149597

**:SGD, With Momentum=0.9**  
**LR=0.001, EPOCHS=50**



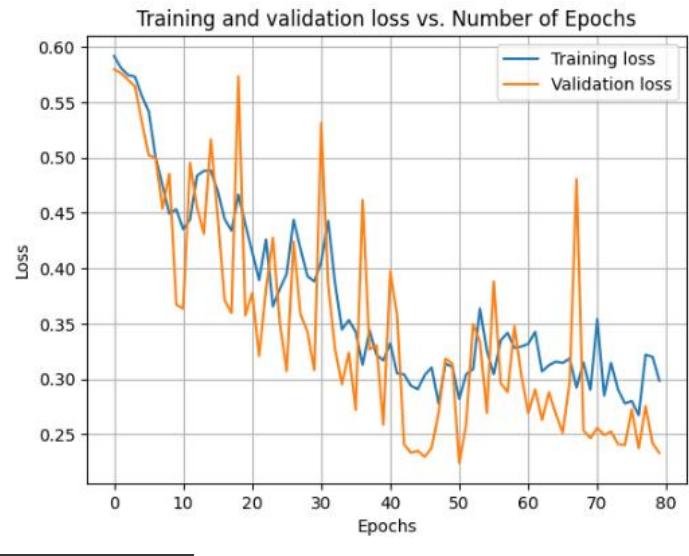
Test accuracy: 0.920648455619812

Test loss: 0.20544059574604034



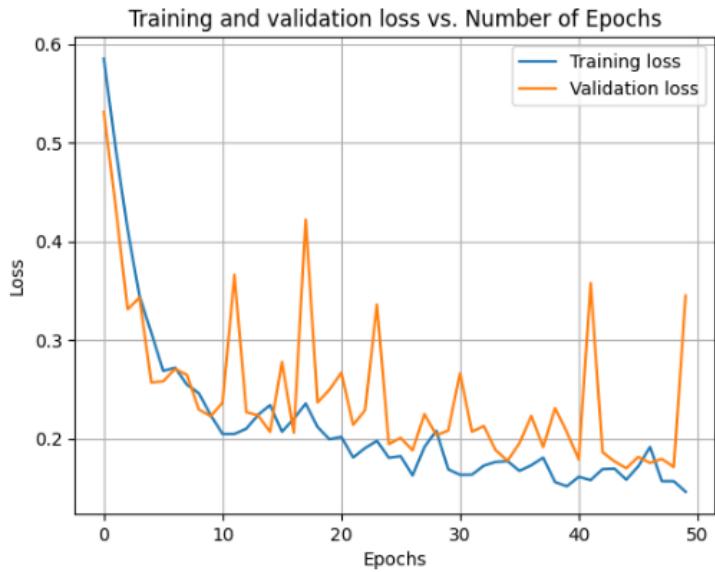
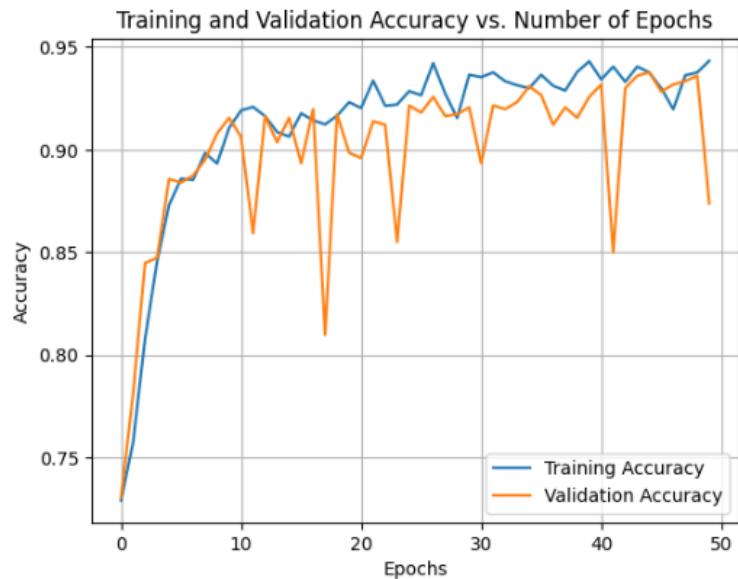
Test accuracy: 0.9138225317001343

Test loss: 0.23079127073287964



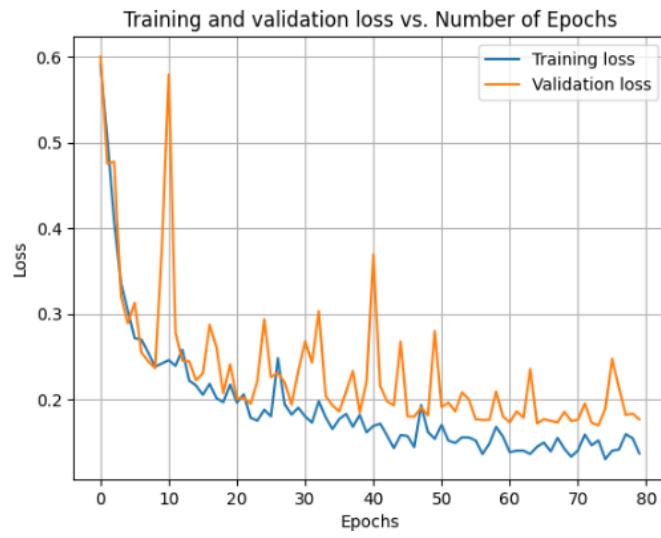
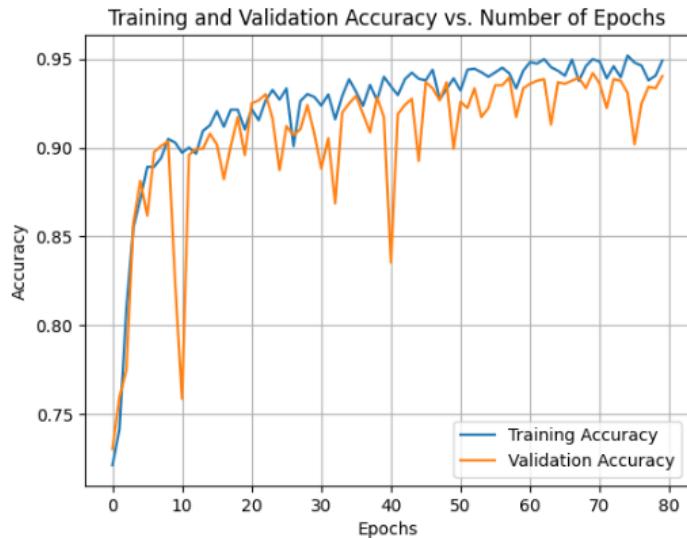
:Adam

LR=0.001, EPOCHS=50



Test accuracy: 0.8788396120071411  
Test loss: 0.3416346609592438

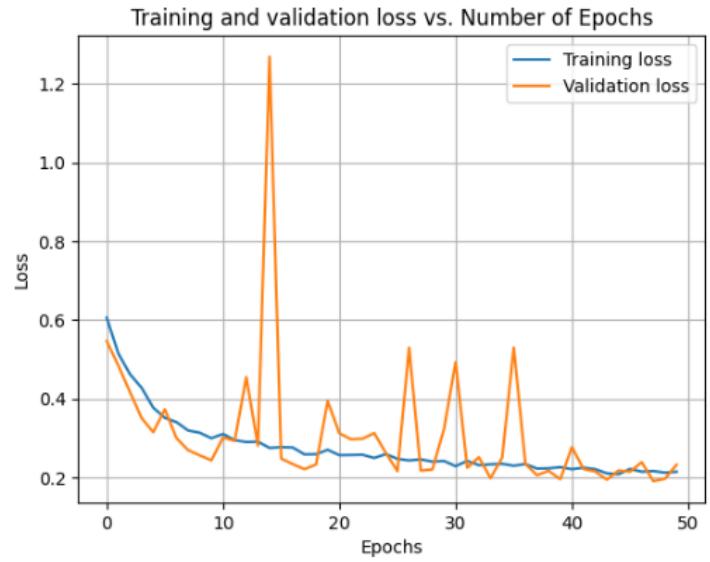
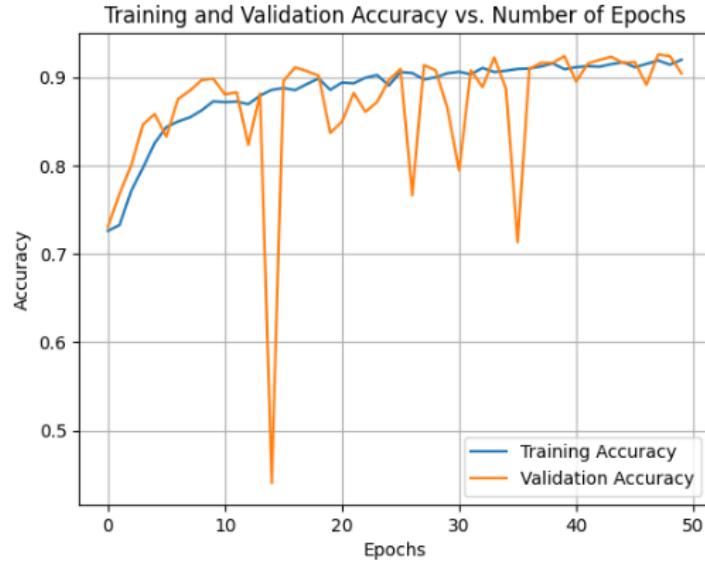
LR=0.001, EPOCHS=80



Test accuracy: 0.9283276200294495  
Test loss: 0.17620277404785156

:RMSprop

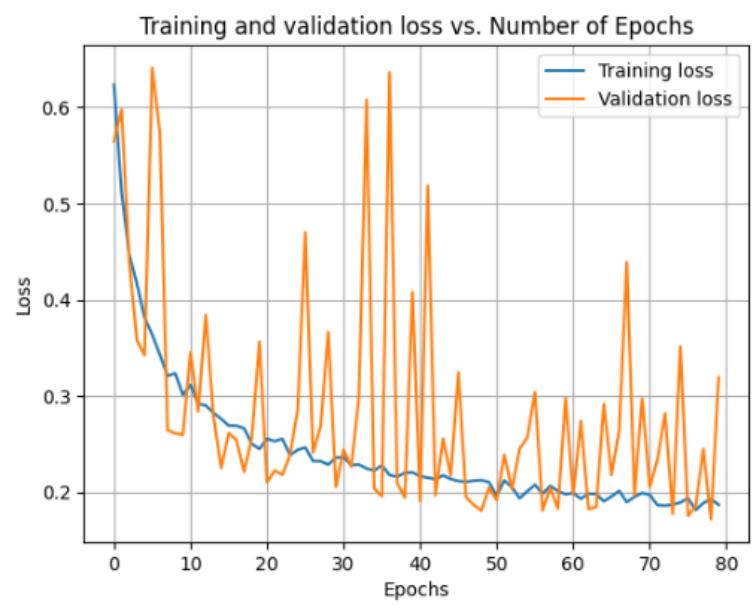
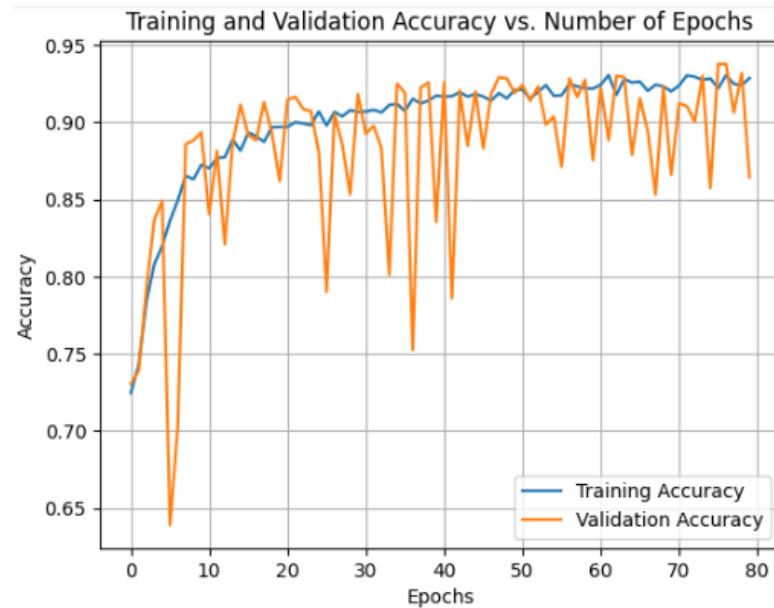
LR=0.001, EPOCHS=50



Test accuracy: 0.9010238647460938

Test loss: 0.24718108773231506

LR=0.001, EPOCHS=80



Test accuracy: 0.8532423377037048

Test loss: 0.33216291666030884

**סיכום התוצאות:**

נסכם את התוצאות על סט הבדיקה בטבלאות:

SGD	
Epochs	Learning Rate
50	0.837
80	0.904

SGD (With Momentum = 0.9)	
Epochs	Learning Rate
50	0.9206
80	0.9138

RMSprop	
Epochs	Learning Rate
50	0.901
80	0.853

Adam	
Epochs	Learning Rate
50	0.878
80	0.928

ניתן לראות כי עבור האלגוריתם Adam עם שיעור למידה של 0.001 ו- 80 אפוקס קיבלנו את התוצאה הטובה ביותר ביותר עם 92.8% דיוק על סט הבדיקה.

נבחן את טיב התוצאות שקיבלו עבור אלגוריתם זה עם מנגנון עצירה מוקדם – Early Stopping –

הפרמטרים למנגנון העצירה המוקדם:

monitor='val\_loss'

patience=10

mode='min'

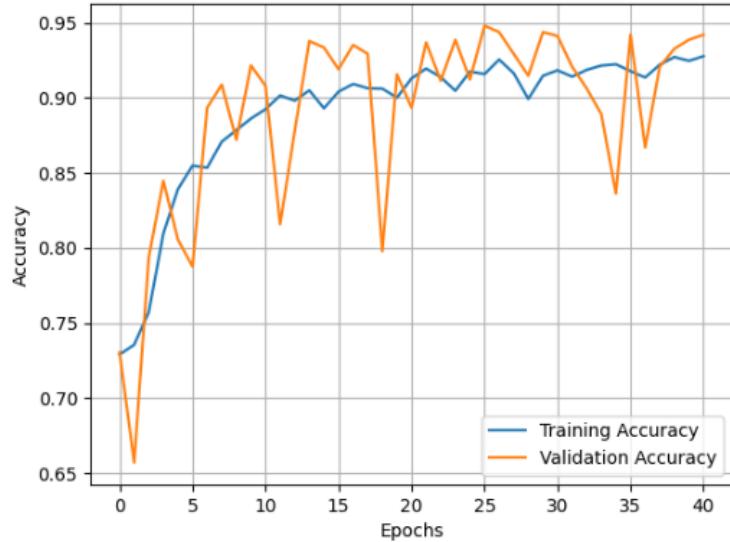
restore\_best\_weights=True

```

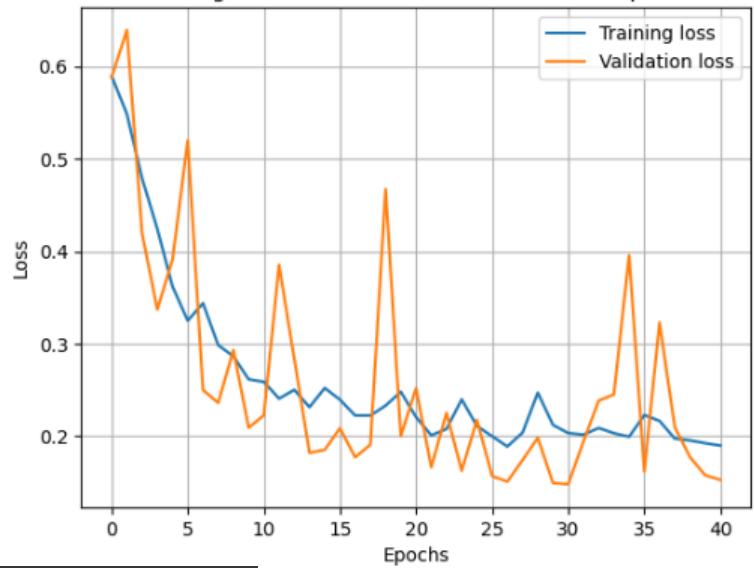
Epoch 32/80
176/176 [=====] - 2s 13ms/step - loss: 0.2019 - accuracy: 0.9140 - val_loss: 0.1945 - val_accuracy: 0.9206
Epoch 33/80
176/176 [=====] - 2s 13ms/step - loss: 0.2090 - accuracy: 0.9186 - val_loss: 0.2386 - val_accuracy: 0.9061
Epoch 34/80
176/176 [=====] - 2s 13ms/step - loss: 0.2034 - accuracy: 0.9214 - val_loss: 0.2453 - val_accuracy: 0.8891
Epoch 35/80
176/176 [=====] - 2s 13ms/step - loss: 0.1996 - accuracy: 0.9223 - val_loss: 0.3955 - val_accuracy: 0.8362
Epoch 36/80
176/176 [=====] - 2s 14ms/step - loss: 0.2230 - accuracy: 0.9177 - val_loss: 0.1621 - val_accuracy: 0.9420
Epoch 37/80
176/176 [=====] - 2s 13ms/step - loss: 0.2167 - accuracy: 0.9134 - val_loss: 0.3230 - val_accuracy: 0.8669
Epoch 38/80
176/176 [=====] - 2s 13ms/step - loss: 0.1976 - accuracy: 0.9220 - val_loss: 0.2100 - val_accuracy: 0.9206
Epoch 39/80
176/176 [=====] - 2s 13ms/step - loss: 0.1956 - accuracy: 0.9271 - val_loss: 0.1776 - val_accuracy: 0.9326
Epoch 40/80
176/176 [=====] - 2s 13ms/step - loss: 0.1926 - accuracy: 0.9245 - val_loss: 0.1579 - val_accuracy: 0.9386
Epoch 41/80
174/176 [=====] - ETA: 0s - loss: 0.1909 - accuracy: 0.9270Restoring model weights from the end of the best epoch: 31.
176/176 [=====] - 3s 14ms/step - loss: 0.1900 - accuracy: 0.9277 - val_loss: 0.1529 - val_accuracy: 0.9420
Epoch 41: early stopping

```

Training and Validation Accuracy vs. Number of Epochs



Training and validation loss vs. Number of Epochs



Test accuracy: 0.9428327679634094

Test loss: 0.15784472227096558

ניתן לראות כי מנגנון העצירה המוקדמת שיפור במעט את הדיווק של הרשות וכעת הדיווק על סט הבדיקה עומד על כ- 94.28% הנחשב לביצועים טובים לפי הגדרת המשימה.

מערכות לומדות  
ולמידה عمוקה,  
31245

**נספחים:**

- .1 [צילומי רנטגן – קבוצת הנתונים \(Data Set\)](#)
- .2 [הצעות לפתרון \(Transfer Learning\)](#)