









עבודה 2- Labeled Face In The Wild:

כדי לפתח מודל לסיווג תמונה בצילום אחד, אנחנו שואפים ללמוד רשת נוירונים שבה נוכל להבחין בין צמד תמונות אם הן שונות או דומות.

לכן על מנת שהמודל ידע להבחין בין דומה לשונה. לכן נחלק את המודל שלנו לזוגות של תמונות. זוגות שהם דומים וזוגות שהם שונים.

		same	"cow" (speaker #1)	"cow" (speaker #2)	same
		different	"cow" (speaker #1)	"cat" (speaker #2)	different
		same	"can" (speaker #1)	"can" (speaker #2)	same
		different	"can" (speaker #1)	"cab" (speaker #2)	different

Verification tasks (training)

הבחנה בין דומה לשונה. בתמונה: ידע להבחין שהתמונות עם הכדורים הן זהות למרות שיש רקע שחור לתמונה

ביצענו ניתוח על כמות הדגימות שיש בקבצים.

העלינו את התיקיה לדרייב וקראנו את התמונות וקבצי הטקסט משם.

המספר הראשון בקובץ הינו כמות התמונות שזהות. (אותו בן אדם על פני כמה תמונות).

```
total samples amount is: 3200
The amount of Samples in Train Set : 2200
The amount of Samples in Test Set : 1000
The amount of same people in Train Set : 1100
The amount of same people in Test Set : 500
The amount of different people in the Training Set : 1100
The amount of different people in the Test Set : 500
```

מחצית מן התמונות בשני הקבצים (בסט האימון וסט המבחן) הן שייכות לסט של זוגות של תמונות שהן דומות והמחצית האחרת של התמונות, שייכות לסט של זוגות של תמונות שהן שונות זו מזו.

הקצאת לייבלים + חלוקה לסטים:

לאחר שביצענו את הניתוח, הקצינו לייבלים. 1- לתמונות זהות. 0- לתמונות שונות.

חילקנו את סט האימון וסט המבחן לשני חלקים:

1. זוגות של תמונות זהות

2. זוגות של תמונות שונות

סך הכל 4 סטים (2 לכל סט)

יצירת נתיבים לכל תמונה:

כל תמונה יצרנו את הנתיב המלא עבורה. לדוגמה:

/content/drive/MyDrive/DL/lfw2/Wendy_Kennedy/Wendy_Kennedy_0001.jpg

עבור סט של זוגות התמונות הזהות:

מערך שבו 3 ערכים:

יש את שם הדמות שבתמונה (כמו בדוגמה למעלה: Wendy Kennedy).
ושני מספרים המהווים את מספר התמונה של אותו בן אדם.

לדוגמה: ['Wendy_Kennedy', '1', '2']

עבור סט של זוגות התמונות השונות:

מערך שבו יש את שם דמות שבתמונה (כמו בדוגמה למעלה: Wendy Kennedy) ומספר
התמונה של אותו בן אדם. בנוסף, להבדיל מסט התמונות הזהות, יש תמונה של דמות
אחרת ומספר התמונה של אותה הדמות.

לדוגמה: ['Wendy_Kennedy', '1', 'Zara_Akhmadova', '1']

חלוקה ל train, validation:

בחרנו לחלק את הנתונים ל 85 אחוזים לאימון ו 15 אחוזים ל validation.

כתוצאה מכך יש לנו בסט האימון 1870 תמונות ובסט הוואלידציה 330 תמונות

כמות הנתונים שיש בידינו בסט האימון היא לא גדולה במיוחד (ברמת האלפים ולא ברמת
עשרות ומאות אלפים, יש 2200 תמונות) לכן ראינו חשיבות רבה לנצל בכמות המירבית
האפשרית (שגם לא תגרום ל overfeating) את הכמות התמונות הלא גדול שיש בידינו.

:Pictures Pre-processing

לאחר ששמרנו עבור סוג של זוגות של תמונות את הנתבים שלהם, איחדנו את הזוגות ללייבלים התואמים להם (זוגות שונים, יותאם לייבל 0. זוגות דומים- לייבל 1).

מימשנו פונקציה הנקראת `image_preprocess`:

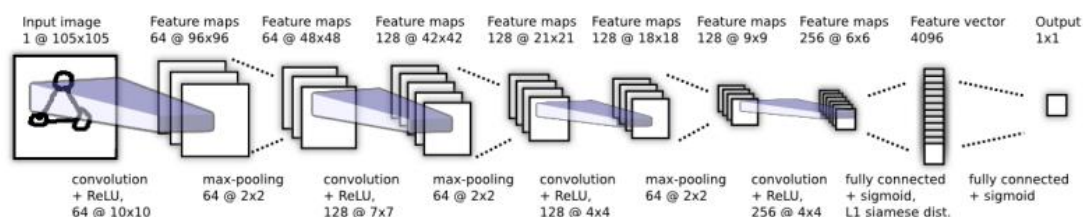
- הקוראת את התמונות מן הנתבים שלהם ומבצעת decoding (המרת קידוד התמונה ל bitmap).
- משנה את גודל התמונה לגודל התמונה שבה המאמר משתמש (105,105).
- מנרמלת את הפיקסלים (חילוק ב 255). הסיבה לנרמול ה id של הפיקסלים היא מהחשש ל גרדיאנטים מפוצצים (exploding gradient) [תהיה התעסקות עם מספרים מאוד גדולים שבעקבות זאת עדכוני הרשת יהיו קטנים מידי כדי להוריד את ערכי המשקולות ובעקבות זאת הן יתקעו מעל 1]. בנוסף, נרמול עוזר להתמודד עם רעשים ועוזר להתכנסות יותר מהירה.

מימשנו פונקציה נוספת שמזמינה את הפונקציה `image_preprocess` עבור זוג תמונות ומחזירה את תמונה מנורמלת לאחר decoding. השתמשנו בפונקציית `map` כדי לקיים זאת את עבור כל זוג תמונות. הודות לכך, הצלחנו לבצע את תהליך עיבוד התמונות בזמן מהיר מאוד (השואף ל 0 שניות) בעקבות האפשרות לבצע זאת בצורה מקבילית [tensorflow מאפשר לעבד מידע בכמויות גדולות, מסוגים שונים באופן מקבילי ומהיר מאוד]. לאחר הביצוע איחדנו את הלייבלים התואמים לזוגות התמונות (שכעת הם מסוג של tensorflow).

בנוסף לאלה, החלטנו לבצע מספר פעולות על הדאטה סטים השונים (train, test, validation):

1. חלוקה ל batch- מספר הדגימות שיש לעבור עליהם לפני עדכון פרמטרי המודל. כפי שראינו במאמר, גודל ה batch המתאים הוא 128.
2. Shuffle- ממלאים באפר בכמות תמונות (במקרה שלנו 1000). בכל פעם דוגמים באקראי מן הבאפר ומחליפים את התמונות הנבחרות בתמונות שבמאגר הנתונים. דבר זה עוזר במניעת overfitting and variance.

מימוש המודל:



```
input=Input(shape=(105,105,1),name='input_image')
#first block
first_convolution = Conv2D(64,(10,10), activation='relu', kernel_regularizer=l2(0.001),name='first_convolution')(input)
batch_normal1=BatchNormalization(name='first_batch')(first_convolution)
first_maxpool = MaxPooling2D(name='first_maxpool')(batch_normal1)
dropout1=Dropout(0.1,name='first_dropout')(first_maxpool)

#second block
second_convolution = Conv2D(128,(7,7), activation='relu', kernel_regularizer=l2(0.001),name='second_convolution')(first_maxpool)
batch_normal2=BatchNormalization(name='second_batch')(second_convolution)
second_maxpool = MaxPooling2D(name='second_maxpool')(batch_normal2)
dropout2=Dropout(0.8,name='second_dropout')(second_maxpool)

#third block
third_convolution = Conv2D(128,(4,4), activation='relu', kernel_regularizer=l2(0.001),name='third_convolution')(second_maxpool)
batch_normal3=BatchNormalization(name='third_batch')(third_convolution)
third_maxpool = MaxPooling2D(name='third_maxpool')(batch_normal3)
dropout3=Dropout(0.8,name='third_dropout')(third_maxpool)

#final block (sigmoid)/ The embedding
fourth_convolution = Conv2D(256,(4,4), activation='relu', kernel_regularizer=l2(0.001),name='fourth_convolution')(third_maxpool)
batch_normal4=BatchNormalization(name='fourth_batch')(fourth_convolution)
flatten = Flatten(name='flatten_vector')(batch_normal4)
output = Dense(4096, activation='sigmoid', kernel_regularizer=l2(0.001),name='output_embedding')(flatten)

model=Model(inputs=[input],outputs=[output])
return model
```

ביצענו מימוש כפי שמתואר במאמר. 4 בלוקים כאשר הבלוק הראשון, השני והשלישי מורכבים משכבת קונבולוציה ו maxpoolfilter. בבלוק הרביעי בנוסף לשניים אלו יש density layer שלאחריה יוצא פלט שהוא וקטור פיצרים בגודל 4096.

- בקוד ניתן לראות הוספה של batch normalization ו dropout. לאחר הרצת המודל הראשוני הבסיסי כפי שמתואר במאמר, הרצנו שיטות אלו לצורך שיפורים וניסויים על המודל. בתור בסיס, הפעלנו רק את המודל שמצויין במאמר ולאחר מכן לצורך ניסויים ושיפורים הרצנו את קטע הקוד שמופיע בתמונה.
- במאמר מוזכר על הטווח על רגולוציה. הטווח שמציין המאמר היינו [0,0.01]. אנו הענשנו בערך של 0.001.

input_image (InputLayer)	[(None, 105, 105, 1)]	0
first_convolution (Conv2D)	(None, 96, 96, 64)	6464
first_batch (BatchNormaliza tion)	(None, 96, 96, 64)	256
first_maxpool (MaxPooling2D)	(None, 48, 48, 64)	0
first_dropout (Dropout)	(None, 48, 48, 64)	0
second_convolution (Conv2D)	(None, 42, 42, 128)	401536
second_batch (BatchNormaliz ation)	(None, 42, 42, 128)	512
second_maxpool (MaxPooling2 D)	(None, 21, 21, 128)	0
second_dropout (Dropout)	(None, 21, 21, 128)	0
third_convolution (Conv2D)	(None, 18, 18, 128)	262272
third_batch (BatchNormaliza tion)	(None, 18, 18, 128)	512
third_maxpool (MaxPooling2D)	(None, 9, 9, 128)	0
third_dropout (Dropout)	(None, 9, 9, 128)	0
fourth_convolution (Conv2D)	(None, 6, 6, 256)	524544
fourth_batch (BatchNormaliz ation)	(None, 6, 6, 256)	1024
flatten_vector (Flatten)	(None, 9216)	0
output_embedding (Dense)	(None, 4096)	37752832
=====		
Total params: 38,949,952		
Trainable params: 38,948,800		
Non-trainable params: 1,152		

הרצות וניסויים:

- אתחול המשקולות זה bias- ברירת המחדל של tensorflow.

כלל עצירה:

```
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=10)
```

כלל עצירה זה אומר שהמודל יעצור כאשר הוא יפסיק לראות בסט הוואלידציה במשך 10 epochs.

ניסוי 1: ADAM/SGD

הרצנו את ADAM ו SGD ב 20 epochs, כשקצב הלמידה הוא 0.00001.

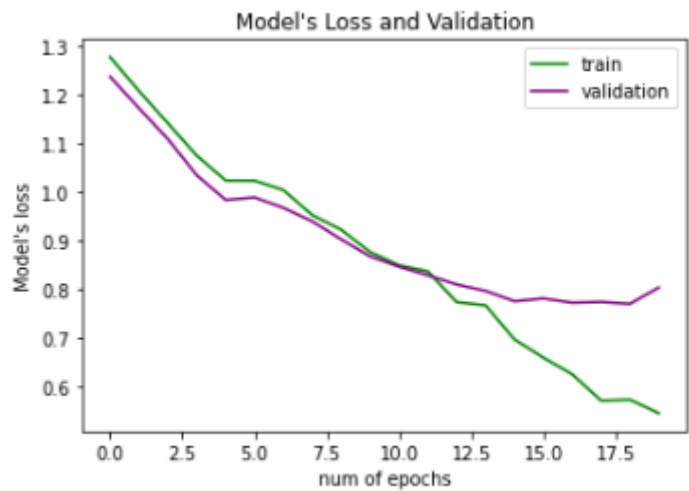
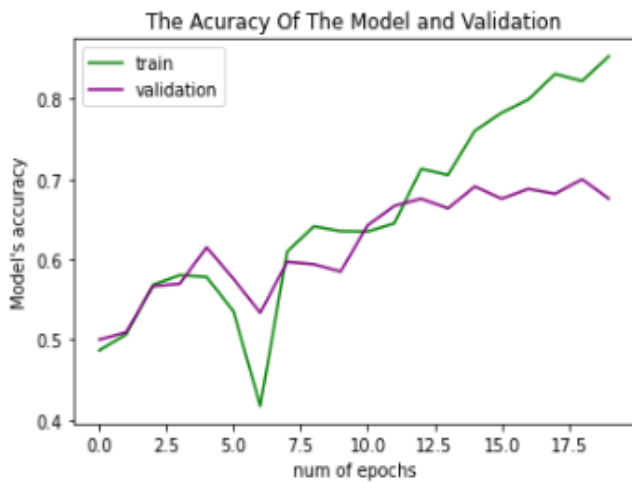
קיבלנו ש:

:Adam

כמות ה epochs-20.

- אחוז דיוק האימון-0.8535
- אחוז דיוק על סט ה validation-0.6758
- Train Loss-0.5436
- Validation loss-0.865
- אחוז דיוק על סט המבחן-0.645
- אחוז ה loss על סט המבחן-1.06

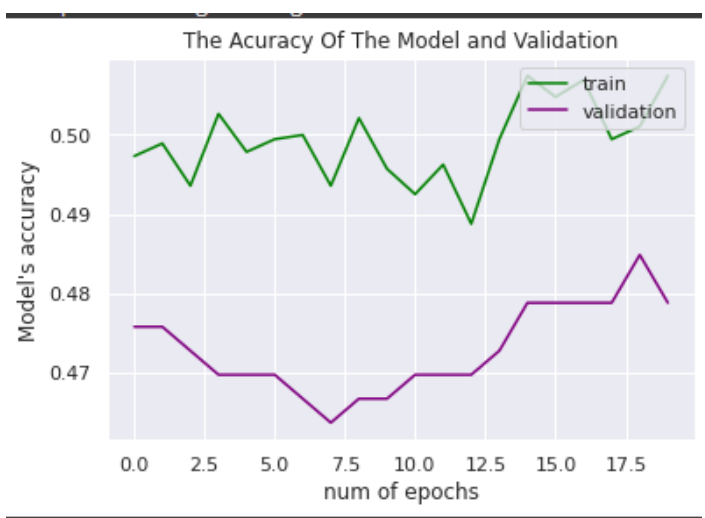
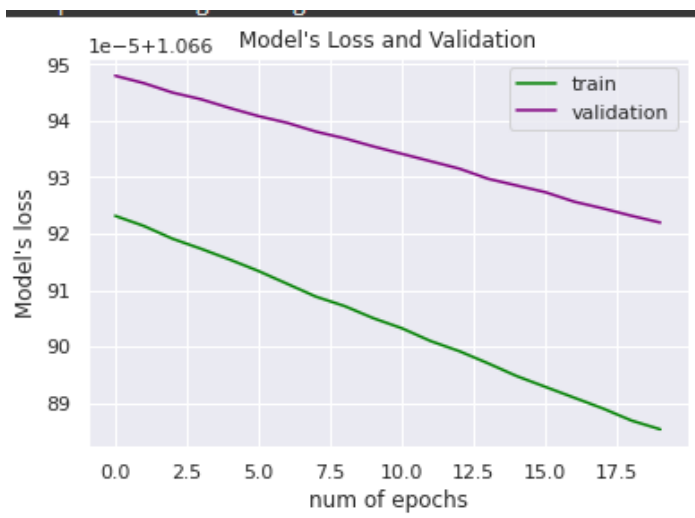
- זמן ממוצע לכל epoch- 16.8 שניות.



:SGD

כמות ה epochs-20.

- אחוז דיוק האימון- 0.512
- אחוז דיוק על סט ה validation- 0.48
- Train Loss (בדיוק של 5 ספרות ה הפסד קטן יותר מן הוואלידציה) 1.06
- Validation loss- 1.06
- אחוז דיוק על סט המבחן- 0.502
- ה loss על סט המבחן- 1.12
- זמן ממוצע לכל epoch- 13.2 שניות.



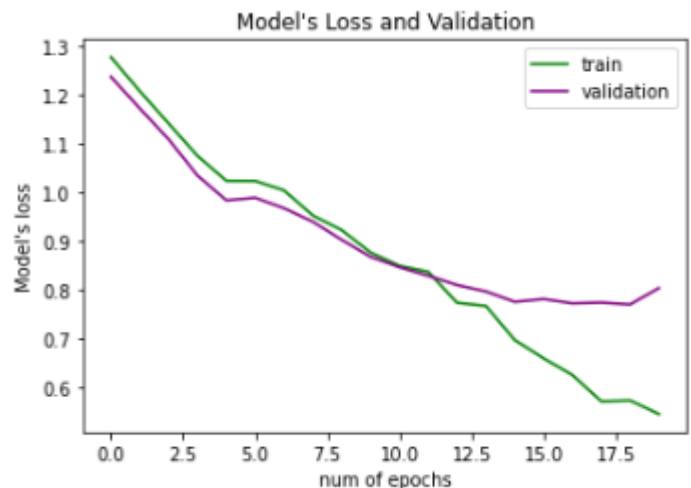
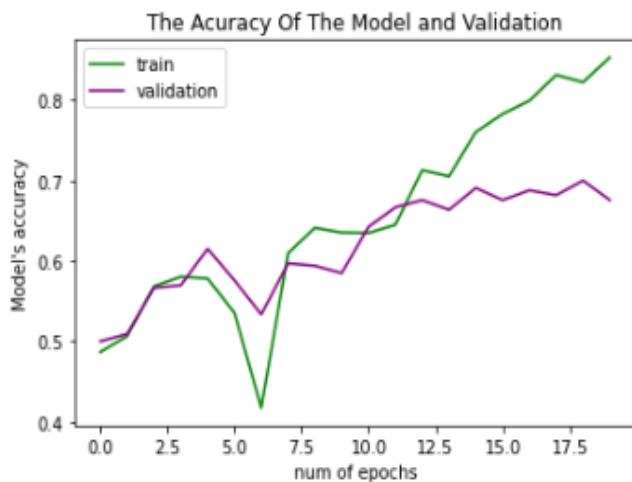
מסקנה: Adam משמעותיות יותר טוב. אחוז הדיוק בכל הסטים יותר טובים ו loss של המודל הולך וקטן ככל שכמות ה epochs הולכת ועולה (כנ"ל לדיוק). בנוסף לכך, המודל מתכנס בצורה יותר מהירה מאשר ב SGD.

פעלנו במהלך כל שאר הניסויים עם ADAM בגלל הביצועים הטובים שהניב עבור המודל שלנו.

ניסוי 2: Batch Normalization

בלי batch normalization:

- אחוז דיוק האימון-0.8535
- אחוז דיוק על סט ה validation-0.6758
- Train Loss-0.5436
- Validation loss-0.865
- אחוז דיוק על סט המבחן-0.645
- ה loss על סט המבחן-1.06
- זמן ממוצע לכל epoch-16.8 שניות

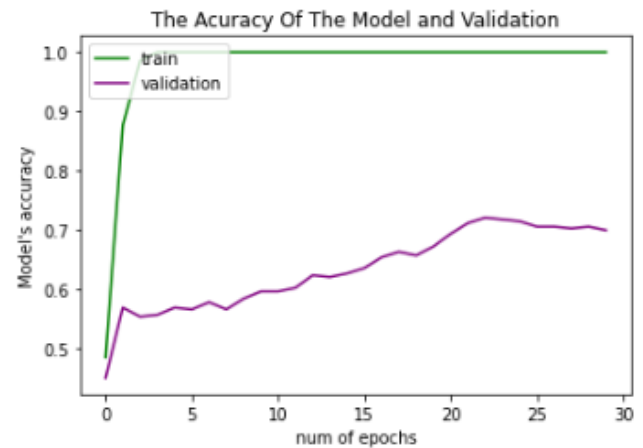
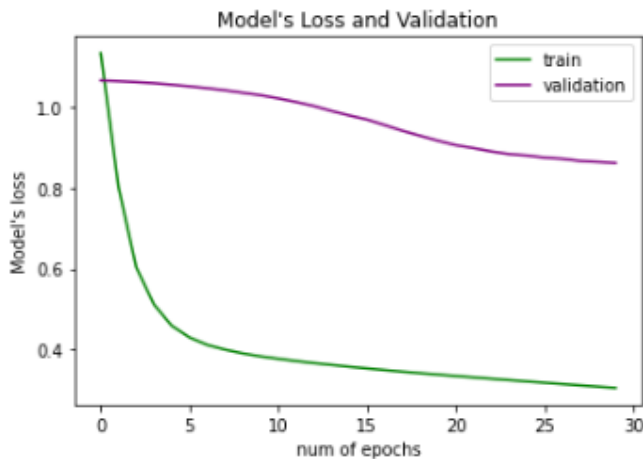


עם batch normalization:

כמות ה epochs-30

- אחוז דיוק האימון-1.00

- אחוז דיוק על סט ה-validation - 0.71
- Train Loss - 0.342
- Validation loss - 0.8514
- אחוז דיוק על סט המבחן - 0.688
- loss על סט המבחן - 0.969
- זמן ממוצע לכל epoch - 12.67 שניות.



מסקנות: כפי שניתן לראות ה batch normalization משפר את המודל הן מבחינת דיוק והן מבחינת loss באופן משמעותי על הסטים השונים.

הסיבה לכך היא שיש הבדלים בין שכבות אקטיבציה שונות. לדוגמה: שכבה 5 מוציאה פלט של מידע עם סטיית תקן של 2 וממוצע 1 ושכבה 7 מוציאה סטיית תקן 7 וממוצע 10. דבר זה עלול לגרום לבעיות כי זה מקטין את הגרדיאנט בין השכבות. לכן יש צורך בנרמול.

כתוצאה מכך, batch אחד לא מושפע מ batch אחר. המודל יותר עמיד בפני שינויים של ה data.

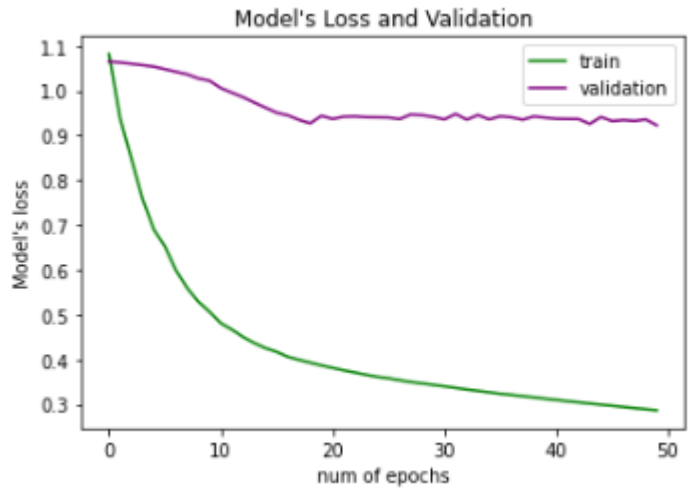
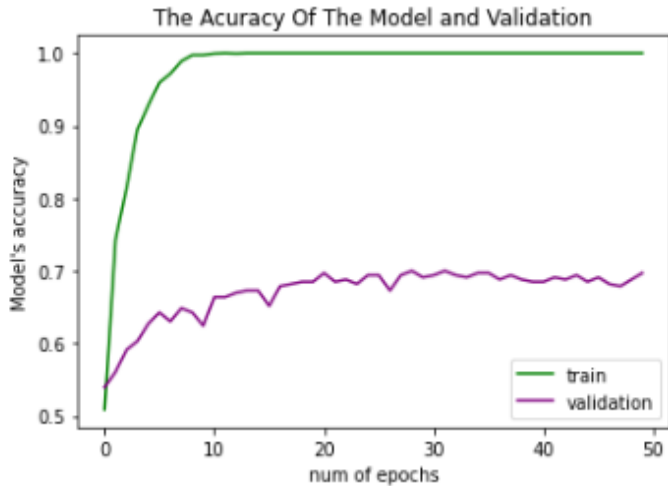
ניסוי 3:

עם batch normalization+dropout:

כמות ה epochs-50. עם dropout rate=0.1

- אחוז דיוק האימון - 1.00
- אחוז דיוק על סט ה-validation - 0.72

- 0.30 - Train Loss
- 0.9229 - Validation loss
- אחוז דיוק על סט המבחן - 0.72
- loss על סט המבחן - 0.908
- זמן ממוצע לכל epoch - 11.9



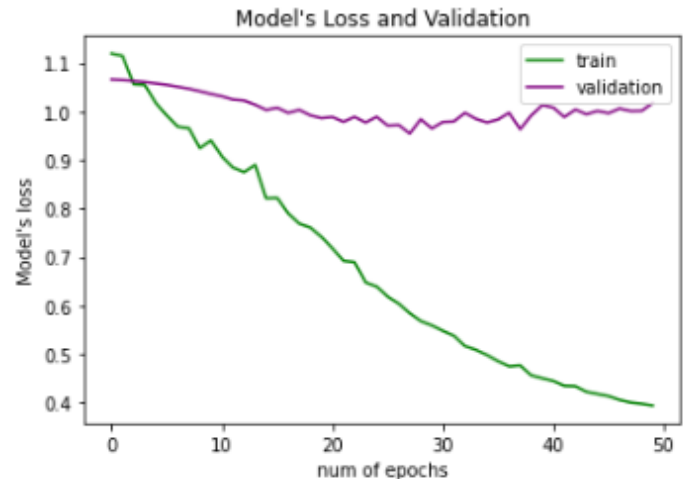
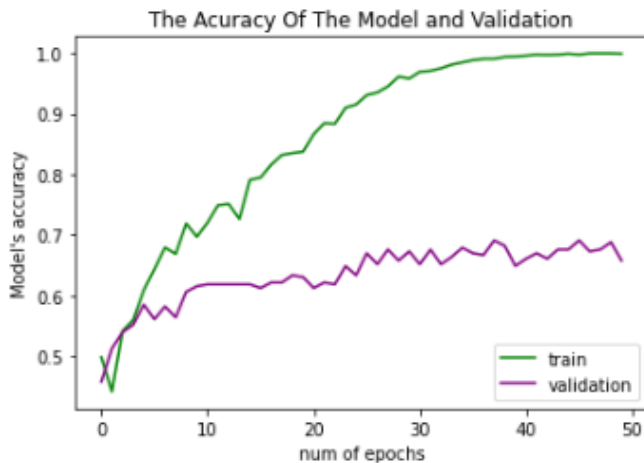
ניסוי 4:

עם batch normalization+dropout:

כמות ה epochs-50. עם dropout rate=0.5

- אחוז דיוק האימון - 0.9995
- אחוז דיוק על סט ה validation - 0.6576
- 0.3941 - Train Loss
- 1.0176 - Validation loss
- אחוז דיוק על סט המבחן - 0.774
- loss על סט המבחן - 0.8448

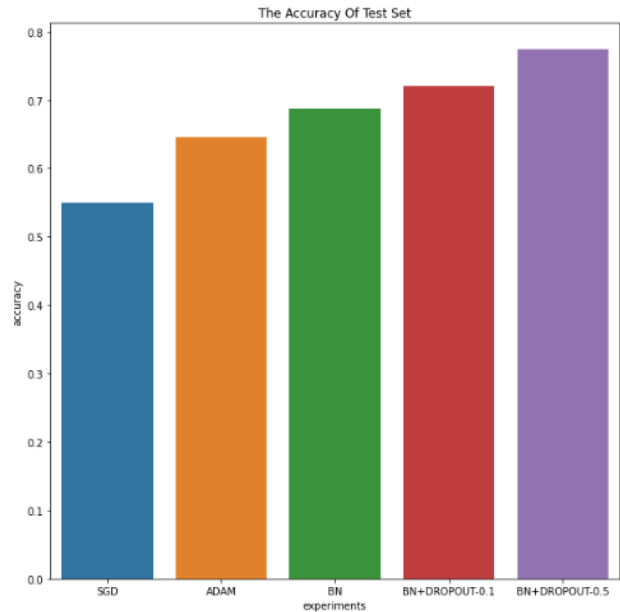
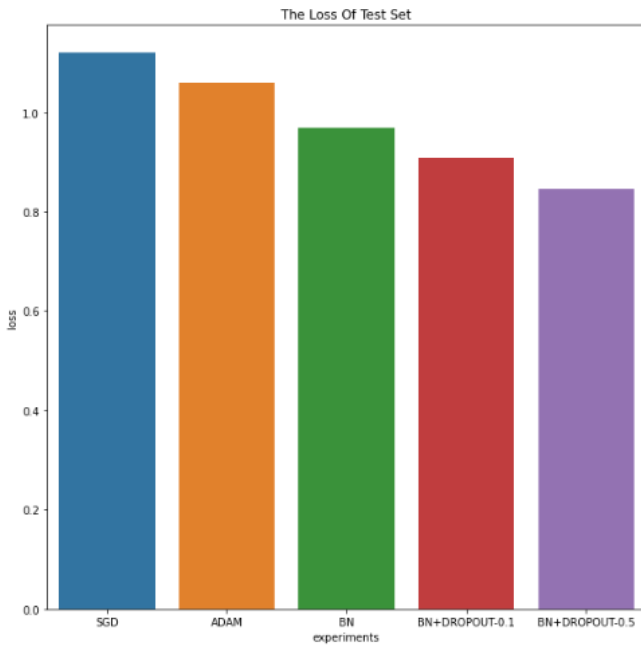
11.12 - זמן ממוצע לכל epoch



מסקנות:

- Dropout משפר את הביצועים של המודל בכל הסטים, הן מבחינת דיוק, זמן לכל epoch ו Loss. שכבת dropout גורמת למודל להיות פשוט יותר. אנו מעלימים באקראי בכל איטרציה חלק מן ה inputs (הפלטנים שלהם יהיו 0). כתוצאה מכך, אף נזירון לא בטוח שקלט מנזירון קודם שעליו הסתמך, יגיע אליו בפעם הבאה. בעקבות זאת, הנזירונים חייבים לדאוג שיהיה להם מגוון של מקורות מידע, ז"א שיסתמכו על כמה שיותר נזירונים ולא על נזירון בודד (שעליו תהיה תלות מוחלטת בקלט הבודד שנקבל, אלא אנו רוצים מספר תלויות). דבר זה עוזר להימנע מ overfit ומשפר את ביצועי האימון, הואלידציה והמבחן.
- רצינו לבדוק אם ציון ה dropout ככל שיגדל אז נראה ביצועים פחות טובים/ יותר טובים. כאשר העלינו ל 0.5 את ה dropout rate ראינו שביצועי האימון והוואלידציה ירדו. הסיבה לכך היא שאנו מעלימים בכל איטרציה 0.5 מן הקלטים לנזירון. למודל קשה ללמוד על סט האימון והוואלידציה. אך בעקבות זאת ראינו שהביצועים על סט המבחן עלו (כי אנו פחות מכירים את הנתונים שלנו).

גרפים המראים את שיפור ביצועי test set לאורך הניסויים:



שגיאות במודל:

סיווג שני אנשים כשונים אך בפועל הם זהים - false negative.



אנו חושבים שהשגיאה נובעת כתוצאה מזווית הצילום של הבן אדם. כפי שניתן לראות, הפרופיל של הבן אדם המופיע בתמונה שונה מכל פרצופו. תמונת פרופיל נותנת לנו מבט של חלק מהאלמנטים ולא את הכל (עין אחת, חצי מסידור השיער, אוזן אחת וכו...). דבר המקשה מאוד על המודל לסווג אותם כזהים.

סיווג שני אנשים כזהים למרות שהם שונים - false positive



ניתן לראות שהתמונות שונות.

לדעתנו המודל טעה בגלל לכסון העיניים. הבחורה מנשקת את הבן אדם וכתוצאה מכך העיניים מתלכסנות, דבר הדומה לתמונה של הבן אדם האחר. בנוסף, למרות שהשיער שונה מבחינת אורך, סידור השיער דומה אחד לשני. עוד דבר שכנראה הפריע למודל זה חוסר התפיסה של הפנים המלאות של הגבר בתמונה שבה אישה מנשקת אותו.

תוצאות נכונות של המודל:

סיווג תמונות זהות שבפועל הם זהים: True Positive



ניתן לראות שזה אותו בן אדם שנמצא בשתי התמונות.

למרות שהתמונה השמאלית כהה יותר ובתמונה הימנית יש כיווץ עיניים מה שאין בשמאלית, המודל זיהה את פרצופו של הבן אדם בצורה טובה (את האף, עיניים, שפם, סידור השיער הזהה) ובעקבות זאת סיווג כזהים.

סיווג תמונות שונות שבפועל הם שונות: True Negative



המודל זיהה שהדמויות שונות למרות גורמים מפריעים כגון: מבט לא ישר למצלמה, כובע על ראש אחת הדמויות (קשה לתפוס את השיער), לבוש דומה בחלק העליון (צווארון לבן). ידע לזהות את פרצופי האנשים ובעקבות זאת לקלוט שהם אכן שונים

סיכום העבודה:

לאחר הרבה ניסוי ותהייה של פרמטרים שונים, הוספה של שכבות במודל, חוונו על בשרינו שעות של הרצות כדי להביא את הביצועים הכי טובים. דבר זה הקנה לנו ניסיון בחשיבות של הבנת המושגים התיאורטיים שנלמדו בכיתה ואיך זה בא לידי ביטוי בצורה יפה בקוד ובביצועים השונים.

בנוסף, חשיבות הכנת הנתונים היא קריטית ומהותית על ביצועי המודל. הצגה נכונה של הנתונים, עיבוד מקדים (כגון: חלוקה לבאצ'ים, חלוקה ל טריין טסט ואלידציה) מהווה קשר ישיר לתוצאות המודל.

התוצאות שקיבלנו ברוב הניסויים היא שלמדנו בצורה מאוד טובה את סט האימון. הגענו אף לאחוז דיוק ששואף ואף מגיע ל 1 אך בסט הוואלידציה הגענו 0.65-0.77. דבר זה נגרם מהתאמת יתר של הנתונים. הסיבה שדבר זה קורה היא שלדעתנו הדאטה סט קטן, אין בו מספיק זוגות של תמונות כדי ללמוד על יותר מקרים ובצורה יותר טובה ומוכללת.

