Deep Learning Ex3

Submitters:
Avidan Menashe
207812421
Shoham Galili
208010785

Table of Contents

Background:

בתרגיל זה מימשנו multi-class classification ע"י חמישה מודלים שונים בעזרת ספריית pytorch.
השתמשנו בDATASET- STL 10 אשר מכיל 5000 samples, כך שלכל sample ישנם 3 features 3X96X96 (כל sample הינו תמונה בעלת 96X96 פיקסלים לכל פיקסל ישנם 3 צבעים R G B), בנוסף לכל sample יש את הlabel המתאים לו.
על מנת לממש את המודל בצורה מיטבית חילקנו את הDATASET לשלושה חלקים: Train, Test , Validation.
היות והDATASET לא מכיל מספר רב של samples על מנת לאמן את המודל בצורה הטובה ביותר השתמשנו בשתי שיטות שונות לdata augmentation :
א. horizontal flip(בחרנו שבסבירות של חצי התמונה תבצע סיבוב אופקי)
ב.rotation (בחרנו שהתמונה יכולה לעשות שינוי של עד 8 מעלות)

כאשר ביצענו את הdata augmentation על החלק של הTrain בלבד.

The steps of running the code:

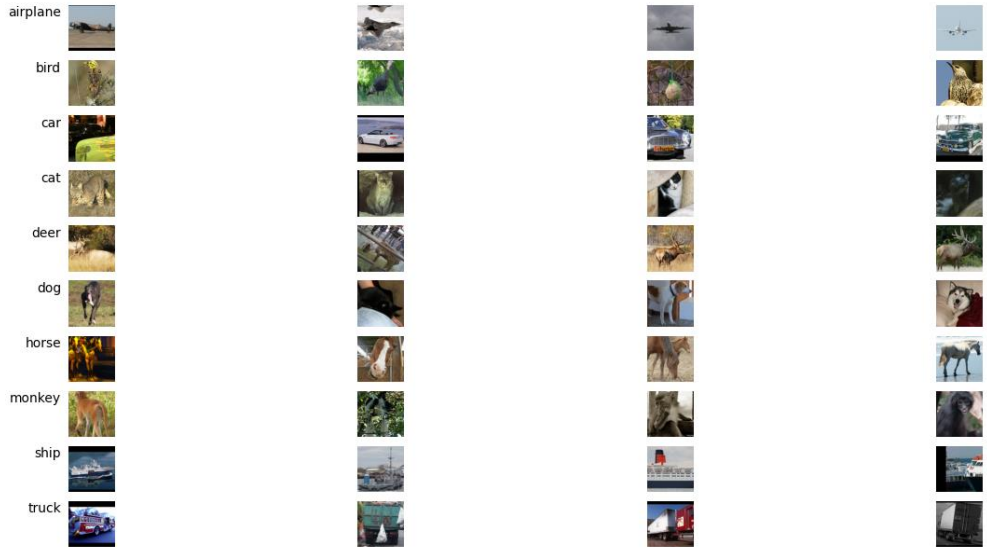
ראשית, על מנת להריץ את הקוד שלנו הנקרא -
"DEEPLEARNING_HW3_Avidan_Shoham.ipymb" יש להעלות אותו אל תיקיית הגוגל דרייב ולאחר מכן לפתוח אותו בפלטפורמת הGoogle Colab .

לאחר מכן ישנם שתי אפשרויות להרצת הקוד או להריץ בלוק באופן פרטני ע"י לחיצת המקשים Ctrl & Enter או לחילופין להריץ את כל קטעי הקוד בבת אחת – נוכל לעשות זאת על ידי לחיצה על הלשונית Runtime (הנמצאת בפינה השמאלית למעלה בGoogle Colab) ולבחור באפשרות של run all.

הערה: במידה ובחרנו באפשרות של להריץ את הבלוקים באופן יחידני יש להריץ את הבלוקים לפי הסדר, כמו כן באפשרות זאת נוכל לבחור אילו מודלים להריץ מתוך כל החמישה ואילו לא רק נשים לב שלפני שבחרנו להריץ את הקטעי קוד של המודלים המסוימים , על מנת שקודי המודלים יעבדו יש בהכרח להריץ את הקודים של הבלוקים הבאים : Imports, global variables , loaders , visualize data , show augmentation, split train and validation, train function, show plots, test function.

Part 1- Visualize the Data

בסעיף זה בחרנו בצורה אקראית 4 תמונות מכל class והצגנו זאת כך שכל שורה מייצגת class אחר, וכל עמודה מייצגת תמונה נוספת מאותו class. בתחילת כל שורה הצגנו את הlabel המייצג כל class כפי שניתן לראות בתמונה הבאה:

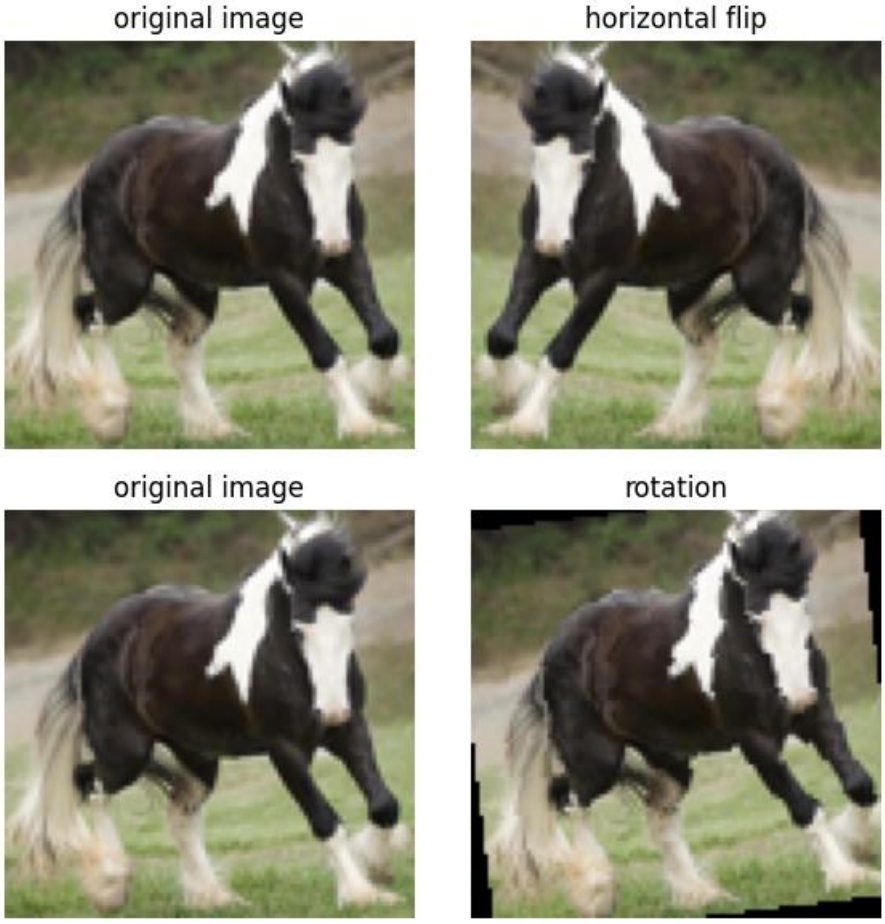


:Part 2- Show the Data Augmentation

בחלק זה בחרנו בצורה אקראית תמונה אחת מתוך ה TRAINSET והראנו כיצד שתי שיטות הaugmentation data שמימשנו על הTRAINSET משנות את התמונה ויוצרות DATA נוסף שבו נשתמש במהלך תהליך הלימוד על מנת לקבל ביצועים טובים ביותר.

ישנן שתי שורות (כמספר ה data augmentation) בשורה הראשונה אני מראים את השפעת ה horizontal flip(בחרנו שבסבירות של חצי התמונה תבצע סיבוב אופקי) ואילו בשורה השנייה אנו מראים את השפעת ה rotation (בחרנו שהתמונה יכולה לעשות שינוי של עד 8 מעלות).

בכל שורה התמונה הראשונה מייצגת את ה data ללא שינוי ואילו התמונה השנייה מייצגת את הdata לאחר השפעת הdata augmentation כפי שניתן לראות בתמונה הבאה :



:Part 3- Classification with Various Networks

בחלק זה יצרנו חמישה מודלים שונים הממשיים multi-class classification לכל אחד מהמודלים אנו נציג טבלה שמראה את תהליך האימון שבו קבענו את ההיפר פרמטרים השונים לכל מודל על מנת למקסם את ערך ה accuracy הנמדד על הvalidation set.
לבסוף, עבור סט הפרמטרים הטוב ביותר נבדוק את הaccuracy המתקבל מה test set.

בנוסף, לכל אחד מן המודלים נציג את הגרפים של פונקציית הloss וערך הaccuracy כפונקציה של מספר הepochs (גם עבור הtrain וגם עבור הvalidation).

:Logistic Regression Model 3.1

רגרסיה לוגיסטית היא סוג של ניתוח רגרסיה המשמש לניבוי ההסתברות לתוצאה בינארית בהתבסס על משתנה מנבא אחד או יותר. סוג הרגרסיה הזה נמצא בשימוש נרחב לבעיות סיווג בינארי. על מנת לממש את המודל עבור Logistic Regression כמבוקש בתרגיל בשימוש ע"י PyTorch ושיטחנו את התמונה מתלת מימד לווקטור באמצעות nn.Flatten כך דאגנו שהקלט לשכבה הליניארית יהיה בצורה הנכונה. השכבה הליניארית בעלת תפקיד מכריע במיפוי תכונות הקלט למחלקות פלט. כל נוירון בשכבה זו מחובר לכל תכונת קלט מהשכבה הקודמת.
בעצם על ידי שכבה ליניארית אחת מימשנו את המודל לרגרסיה לוגיסטית בPyTorch.

בהמשך, על מנת להגיע לסט הפרמטרים אשר מניב את התוצאות המיטביות השתמשנו בשיטת "ניסוי וטעיה". בתחילה, אימנו את המודל עם סט פרמטרים התחלתי, קיבלנו ערך התחלתי ולא מספיק טוב, על כן המשכנו לאמן את המודל כך שבכל הרצה בדקנו את השפעת שינוי אחד הפרמטרים על תוצאות ה **Accuracy** עבור ה**Validation**. המשכנו בשלבים עד אשר הגענו לרמת דיוק טובה מספיק באופן יחסי עבור מודל זה.
לבסוף הרצנו את הפרמטרים הטובים ביותר על מנת למצוא את ה**accuracy** עבור סט ה**test.**

להלן פירוט התהליך שעשינו ושינוי הפרמטרים בהתאמה:

accuracy	Loss function	optimizer	LR	Num epochs	Batch size	Weight decay
19.733	Cross Entropy	ADAM	0.001	20	64	0.0001
16.533	Cross Entropy	ADAM	0.001	20	128	0.0001
16.667	Cross Entropy	ADAM	0.001	20	32	0.0001
17.200	Cross Entropy	ADAM	0.001	40	64	0.0001
15.067	Cross Entropy	ADAM	0.001	60	64	0.0001
14.667	Cross Entropy	ADAM	0.01	20	64	0.0001

13.333	Cross Entropy	ADAM	0.03	20	64	0.0001
18.533	Cross Entropy	ADAM	0.0001	20	64	0.0001
18.533	Cross Entropy	ADAM	0.00001	20	64	0.0001
16.267	Cross Entropy	SGD	0.001	20	64	0.0001
16.000	Cross Entropy	RMSProp	0.001	20	64	.0.0001
13.867	NLL	ADAM	0.001	20	64	0.0001
20.000	Cross Entropy	ADAM	0.001	20	64	0.00005
18.800	Cross Entropy	ADAM	0.001	20	64	0.000005
26.133	Cross Entropy	ADAM	0.001	20	512	0.005

פירוט השפעת hyperparameters על רמת דיוק המודל:

תחילה בחרנו שרירותית את הנתונים ההתחלתיים על מנת לקבל איזושהי תמונה כללית על המודל ונקודה התחלתית. ניתן לראות שהגענו לערך: Accuracy =20.533

- השפעת ערך ה Learning Rate על דיוק המודל.

כידוע קצב הלמידה משפיע ישירות על המהירות וההתכנסות של אלגוריתם האופטימיזציה המשמש לאימון, בחירת קצב למידה מתאים יכולה להשפיע משמעותית על הדיוק וביצועי המודל. Learning Rate נמוך מדי עלול להביא לפתרון לא אופטימלי נוסף על זמן התכנסות ארוך- Underfitting. מאידך, Learning rate גבוה מדי עלול לגרום למודל להתנודד סביב הפתרון האופטימלי ולהוביל לOverfitting. שמנו לב כי בהעלאת ערך ה Learning Rate רמת הדיוק עלתה עד לערך lr = 0.03 ומעבר לכך קיבלנו ירידה משמעותית בדיוק המודל – זאת בהתאם למצופה שהlr לא צריך להיות גבוה מדי. ובlr= 0.001 קיבלנו את ערך הדיוק הגבוה ביותר.

- בשלב הבא, בדקנו את השפעת גודל Num of Epochs על דיוק המודל.

פרמטר זה מייצג את מספר הפעמים שה training dataset מועבר קדימה ואחורה דרך המודל בתהליך האימון. כמות נמוכה של epochs עלולה לגרום לUnderfitting כיוון שהמודל לא מספיק ללמוד מהdataset וגורם לדיוק נמוך יותר. מאידך, כמות גבוהה של epochs עלולה לגרום לOverfitting כיוון שהמודל לומד רעש מהdataset ולכן ישנה חוסר הצלחה בהכללת הלמידה לdata חדש אשר לא נראה קודם. על כן, התחלנו בepochs 20 והעלינו את הערך בהתאם לרמת הדיוק ונוכחנו לגלות כי העלאת כמות הepochs במקרה גרמה לירידה בדיוק לכן נשארנו עם epochs 20.

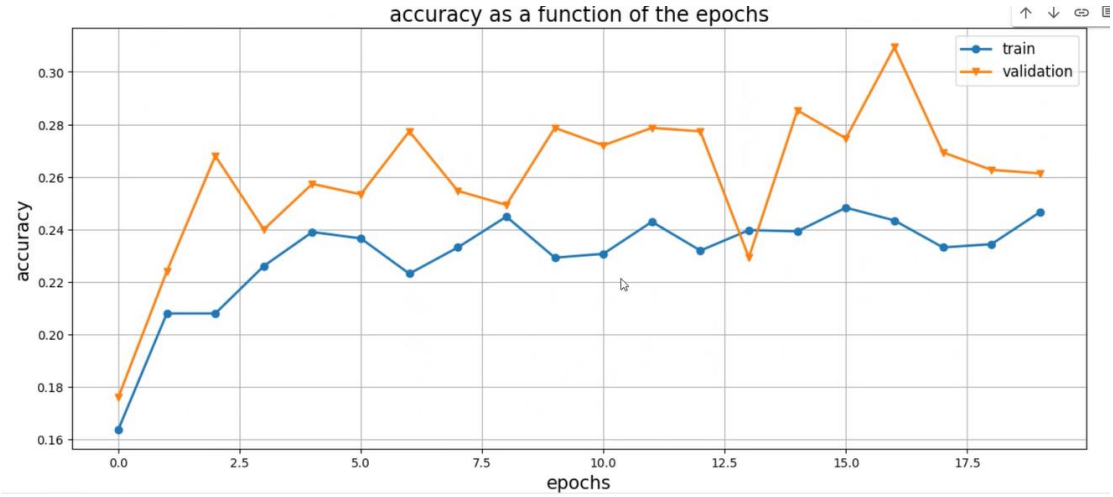
- בשלב הבא, בדקנו את השפעת גודל הBatch Size על דיוק המודל.

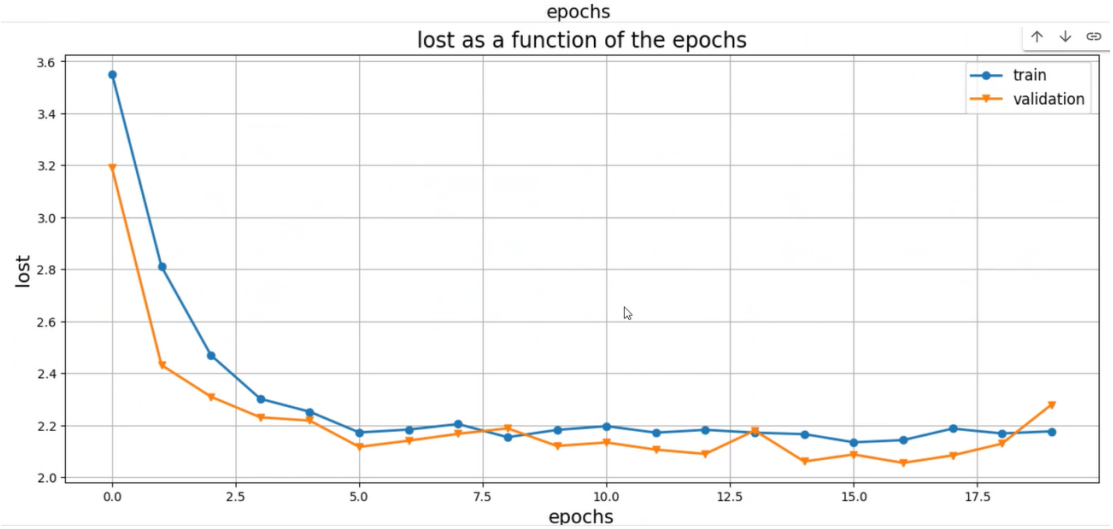
פרמטר זה קובע את מספר הדגימות המעובדות לפני עדכון הפרמטרים של המודל במהלך תהליך האימון. הוא ממלא תפקיד מכריע בקביעת היעילות והאפקטיביות של תהליך האופטימיזציה. הBatch size משפיע ביחס ישר על התכנסות המודל- זאת משום שגדלי קבוצות גדולים יותר גורמים בד"כ לעדכונים יציבים יותר והתכנסות מהירה יותר. כיוון שהן מספקות הערכות מדויקות יותר של הגראדינט של פונקציית הloss. התחלנו בערך Batch Size=64 הגדלנו אותו על מנת להגיע לביצועים טובים יותר של המודל כמצופה, אך שמנו לב כי קיים מעין חסם עליון/תחתון בערך 32/128 (בהתאמה) בהם רמת הדיוק החלה לרדת מכיוון שערך Batch Size גדול/קטן מדי עלול לגרום לאילוצי זיכרון ולהתכנסות איטית יותר.
על כן הערך האופטימלי מבחינתנו הוא Batch Size = 64 כפי שמוצג בטבלה.

- השפעת פונקציית הloss וה optimizer:

יתר על כן במהלך תהליך האימון בדקנו גם פונקציות loss שונות כגון cross entropy ו The negative log likelihood loss , מצאנו כי פונקציית loss הנותנת דיוק המירבי עבור המודל הינה **cross entropy** (כפי שניתן לראות בטבלה).
בנוסף לכך, בדקנו סוגים שונים של optimazer כמו ADAM, SGD, RMSprob
וראינו כי הoptimazer שנותן דיוק רב ביותר הינו **Adam** (כמו שמופיע בטבלה).

כעת נציג את הגרפים המתארים את ערכי ה**loss** וה **accuracy** כפונקציה של מספר ה **epochs** עבור סט ה**Train** וה**Validation**:





מהגרפים לעיל ומהתוצאות שקיבלנו עבור אימון הdata ניתן להסיק כי המודל הנ"ל הינו חלש בהשוואה לשאר המודלים. אכן ניתן להבחין כי ככל שמספר הepochים עולה ← loss יורד בצורה חדה בהתחלה ואז מתקבע על ערך מסוים, accuracy עולה בצורה איטית מאוד.
בהתאם לתיאוריה שהסברנו לעיל וכפי שציפינו שיקרה- האימון של מודל ה logistic regression אינו משיג תוצאות טובות על הaccuracy. שכן הגענו לרמת דיוק של כ 20% שאינה גבוהה במיוחד. זאת משום מספר סיבות:

1. מורכבות הנתונים: מערך הנתונים STL-10 הוא מערך נתונים מאתגר עם תמונות מורכבות. הוא מורכב ממספר רב של תמונות צבעוניות ברזולוציה גבוהה אשר שייכות ל-10 מחלקות. רגרסיה לוגיסטית עשויה ללמוד את הדפוסים והמאפיינים המורכבים הקיימים בתמונות אלה כך שיכולת הלמידה שלה על data חדש הולכת ונחלשת וכן רמת הדיוק שלה קטנה, זאת בגלל גבול ההחלטה הליניארי שלה.
2. אי-לינאריות: רגרסיה לוגיסטית מניחה קשר ליניארי בין תכונות הקלט למשתנה הפלט. עם זאת, הקשרים במערך הנתונים של STL-10 עשויים להיות לא ליניאריים. יכולות להיות אינטראקציות ודפוסים מורכבים בנתונים שרגרסיה לוגיסטית לא יכולה לתפוס מה שיקשה עליה ללמוד את הdata כפי שמצופה וגורם לה להישאר עם רמת דיוק נמוכה יחסית.
3. ממדים גבוהים: לכל תמונה במערך הנתונים של STL-10 יש ממדיות גבוהה בשל הרזולוציה שלה. רגרסיה לוגיסטית עשויה לא להצליח ללמוד את הdata בצורה מיטבית במרחבים בעלי ממדים גבוהים, במיוחד כאשר מספר הfeatures גדול בהרבה ממספר הדגימות. זה יכול להוביל להתאמת יתר או תת-התאמה של הדגם.

לסיכום, בעוד שרגרסיה לוגיסטית יכולה להיות מודל שימושי וניתן לפרשנות למשימות סיווג פשוטות, ייתכן שהיא לא מתאימה למשימות סיווג של תמונות מורכבות כמו אלו המכילות את מערך הנתונים STL-10. מודלים מתוחכמים יותר כמו CNN מועדפים בדרך כלל למשימות כאלה בשל יכולתם ללמוד תכונות מורכבות יותר. ואכן בהמשך העבודה והמחקר נוכחנו לגלות כי שאר המודלים משיגים רמת דיוק גבוהה יותר כמצופה.

בסה"כ הגענו לרמת הדיוק Accuracy validation = 20.533% על סט הTrain.

לבסוף אחרי שסיימנו את תהליך אימון המודל ומצאנו את ערכי ההיפר פרמטרים הטובים ביותר
batch size=64, num of epochs=20, lr=0.001, weight decay=0.005,optimizer=Adam, loss function=cross entropy

בדקנו את ערכי ה **loss, accuracy** שהמודל בזמן ה**test** כלומר, על הtestset ומצאנו כי עבור מודל זה:

```
Test Loss: 2.244 | Test Accuracy: 24.788%
```

3.2 Model ANN (Fully-connected NN):

על מנת ליצור בעזרת מודל זה multi-class classification, אנו ניצור ANN כלומר מודל המורכב רק משכבות לינאריות שהן fully connected וביניהן שכבת input, שכבות ביניים (hidden layers) ושכבת output.

הערה:
במהלך אימון המודל נסינו מספר שונה של שכבות ביניים לינאריות (hidden layers) וראינו כי המספר הגורם לדיוק המודל המיטבי הינו 4 שכבות ביניים כלומר סך הכל המודל שלנו יהיה מורכב מ6 שכבות לינאריות- שכבת input, ארבע שכבות ביניים (hidden layers) ושכבת output.

הערה:
לכל שכבה לינארית יש פרמטר של features in ופרמטר של features out אך הin features של השכה ה i+1 שווה בגודלה לfeatures out של השכבה הi לכן למרות שיש לנו 6 שכבות לינאריות המודל צריך ללמוד פחות מ12 היפר פרמטרים הקשורים לגדלי הfeatures .
בנוסף לכך, ישנן היפר פרמטר שלא נוכל לשחק עם גודלם כמו features out של שכבת ה classification היות ומספר הclasses של הDATASET שלנו הינו 10 הגדול של פרמטר זה חייב להיות 10.
כמו כן, גם ההיפר פרמטר של features in של השכבה הלינארית הראשונה(שכבת הinput) נקבע מגודל הsamples שלנו שהינו $3 \cdot 64 \cdot 64 = 12{,}288$ ולכן הערך שלו קבוע ולא בר שינוי

לכן, בהקשר להיפר פרמטרים הקשורים לגדלי השכבות הלינאריות אנו צריכים למצוא רק את הפרמטרים הבאים : fc1 features in= fc input features out
fc2 features in= fc1 features out
fc3 features in= fc2 features out
fc4 features in= fc3 features out
fc output features in= fc4 features out

בשל כך רק את הגדלים שלהם נשנה ונציג בטבלה המראה את שינוי הפרמטרים בתהליך האימון.

על מנת להגיע לסט הפרמטרים אשר מניב את התוצאות המיטביות השתמשנו בשיטת "ניסוי וטעיה". בתחילה, אימנו את המודל עם סט פרמטרים התחלתי, קיבלנו ערך התחלתי ולא מספיק טוב, על כן המשכנו לאמן את המודל כך שבכל הרצה בדקנו את השפעת שינוי אחד הפרמטרים על תוצאות ה **Accuracy** עבור ה**Validation**. המשכנו בשלבים עד אשר הגענו לרמת דיוק טובה מספיק.
לבסוף הרצנו את הפרמטרים הטובים ביותר על מנת למצוא את ה**accuracy** עבור סט ה**test.**

להלן פירוט התהליך שעשינו ושינוי הפרמטרים בהתאמה:

accuracy	Loss function	optimizer	FC1 input	FC2 input	FC3 input	FC4 input	FC4 output	NUM OF LINEARY LAYERS	Weight decay	LR	Num epochs	Batch size	Drop out FC1	Drop out FC2	Drop out 3FC	D	op out FC
35.467	Cross Entropy	ADAM	1024	512	216	-	-	3	0.001	0.001	30	512	0.1	0.08	0.03		-
38.400	Cross Entropy	ADAM	1024	512	216	-		3	0.001	0.001	20	512	0.1	0.08	0.03		-
38.267	Cross Entropy	ADAM	800	512	216	-		3	0.001	0.001	20	512	0.1	0.08	0.03		-
34.533	Cross Entropy	ADAM	1024	512	216	128	64	4	0.001	0.001	20	512	0.09	0.08	0.03	0	01
33.733	Cross Entropy	ADAM	1024	512	216	128	64	4	0.001	0.001	20	512	0.2	0.15	0.08	0	05
33.067%	Cross Entropy	ADAM	2048	1024	512	216	128	4	0.001	0.001	20	512	0.2	0.15	0.08	0	03
39.200	Cross Entropy	ADAM	2048	1024	512	216	128	4	0.001	0.001	20	512	0.1	0.08	0.05	0	03
32.133	Cross Entropy	ADAM	2048	1024	512	216	128	4	0.005	0.001	20	512	0.1	0.08	0.05	0	03
39.067	Cross Entropy	ADAM	2048	1024	512	216	128	4	0.0001	0.001	20	512	0.1	0.08	0.05	0	03
31.733	Cross Entropy	ADAM	2048	1024	512	216	128	4	0.001	0.003	20	512	0.1	0.08	0.05	0	03
38.800	Cross Entropy	ADAM	2048	1024	512	216	128	4	0.001	0.0005	20	512	0.1	0.08	0.05	0	03
34.933%	Cross Entropy	ADAM	4096	2048	1024	512	216	4	0.001	0.001	20	512	0.1	0.08	0.05	0	03
38.00%	Cross Entropy	ADAM	1024	512	256	128	64	4	0.001	0.001	20	512	0.1	0.08	0.05	0	03
38.267	Cross Entropy	ADAM	2048	1024	512	216	128	4	0.001	0.001	20	256	0.1	0.08	0.05	0	03
32.800	Cross Entropy	ADAM	2048	1024	512	216	128	4	0.001	0.001	20	1024	0.1	0.08	0.05	0	03
26.400	Cross Entropy	RMSProp	2048	1024	512	216	128	4	0.001	0.001	20	512	0.1	0.08	0.05	0	03
13.733%	Cross Entropy	SGD	2048	1024	512	216	128	4	0.001	0.001	20	512	0.1	0.08	0.05	0	03

36.800	The negative log likelihood loss	ADAM	2048	1024	512	216	128	4	0.001	0.001	20	512	0.1	0.08	0.05	0.03
38.00	Cross Entropy	ADAM	2048	1024	512	216	128	4	0.001	0.001	50	512	0.1	0.08	0.05	0.03
39.867	Cross Entropy	ADAM	2048	1024	512	216	128	4	0.001	0.001	34	512	0.1	0.08	0.05	0.03

פירוט השפעת **hyperparameters** על רמת דיוק המודל:

תחילה בחרנו שרירותית את הנתונים ההתחלתיים על מנת לקבל איזושהי תמונה כללית על המודל ונקודה התחלתית. ניתן לראות שהגענו לערך: **Accuracy = 35.467%**

- השפעת **מספר שכבות האמצעיות (hidden layers)** על דיוק המודל.

כאשר הגדלנו את מספר השכבות הלינאריות האמצעיות (hidden layers) מ3 ל4 ושנינו את ערכי ההיפר פרמטרים קיבלנו דיוק טוב יותר מאשר ב3 שכבות אמצעיות.
היות והגדלת מספר השכבות הנסתרות מ-3 ל-4 יכולה לספק למודל קיבולת נוספת(כל שכבה לינארית מאפשרת יותר טרנספורמציות על ה input data וכך נוכל ללמוד טוב יותר על הקשרים בין הנוירונים) לכן נוכל לקבל למידת ייצוג טובה יותר עבור המודל.
בנוסף, כל תוספת של שכבה יכולה לאפשר לגרום לרעש להשפיע פחות על הDATA האמיתי.
יתר על כן, על ידי תוספת של שכבות נוכל לקבל תוצאות gradient מדויקות יותר כי הגרדיאנט יכול עם יותר שכבות לחלחל (propagate) באופן יותר אפקטיבי דרך הרשת.
עם זאת, חשוב לציין שרשתות עמוקות יותר מגיעות גם עם אתגרים כמו מורכבות חישוב מוגברת ורגישות להתאמת יתר, ולכן יש צורך בניסויים ובכוונן זהירים כדי לייעל את הביצועים.

- השפעת ערך ה **Learning Rate** על דיוק המודל.

כידוע קצב הלמידה משפיע ישירות על המהירות וההתכנסות של אלגוריתם האופטימיזציה המשמש לאימון, בחירת קצב למידה מתאים יכולה להשפיע משמעותית על הדיוק וביצועי המודל. **Learning Rate** נמוך מדי עלול להביא לפתרון לא אופטימלי נוסף על זמן התכנסות ארוך- **Underfitting**. מאידך, **Learning rate** גבוה מדי עלול לגרום למודל להתנודד סביב הפתרון האופטימלי ולהוביל ל**Overfitting**. שמנו לב כי בהעלאת ערך ה **Learning Rate** לערך 0.03 והורדת ערך ה **Learning Rate** לערך 0.0005 קיבלנו ירידה בדיוק המודל, רמת הדיוק היתה מטיבית בערך **lr = 0.001** - זאת בהתאם למצופה שה**lr** לא צריך להיות גבוה מדי אך לא נמוך מדי.

- לאחר מכן, בדקנו את השפעת ה **regularization** בעזרת פרמטר **weight decay** על דיוק המודל שלנו.

פרמטר זה "מעניש" משקלים גדולים מדי ומנמיך את ערכם ובכך מונע **Overfitting**, מביא לשליטה מיטבית במורכבות הדגם ושיפורו. בהתחלה קבענו את הערך **weight decay** =0.001
ניסנו לשנות את ערכו של הפרמטר אך ראינו כי בעת הגדלת ערך הפרמטר לערך **0.005** או הקטנת ערך הפרמטר אל הערך **0.0001** דיוק המודל פחת.
לכן, קיבלנו כי עבור הערך **weight decay=0.001** אנו מקבלים את רמת הדיוק המיטבי וקיבענו את פרמטר זה לערך הנ"ל.

- בשלב הבא, בדקנו את השפעת גודל **Num of Epochs** על דיוק המודל.

פרמטר זה מייצג את מספר הפעמים שה **training dataset** מועבר קדימה ואחורה דרך המודל בתהליך האימון. כמות נמוכה של **epochs** עלולה לגרום ל**Underfitting** כיוון שהמודל לא מספיק ללמוד מה**dataset** וגורם לדיוק נמוך יותר. מאידך, כמות גבוהה של **epochs** עלולה לגרום ל**Overfitting** כיוון שהמודל לומד רעש **מה**dataset ולכן ישנה חוסר הצלחה בהכללת הלמידה לdata חדש אשר לא נראה קודם. על כן, התחלנו ב**epochs=20** ונסינו להעלות את הערך, שמנו לב כי ב**epochs=50** קיבלנו ירידה בדיוק של המודל, אך עבור הערך **epochs=34** קיבלנו דיוק גדול יותר ולכן קיבענו את הפרמטר לערך זה.

- בשלב הבא, בדקנו את השפעת גודל ה**Batch Size** על דיוק המודל.

פרמטר זה קובע את מספר הדגימות המעובדות לפני עדכון הפרמטרים של המודל במהלך תהליך האימון. הוא ממלא תפקיד מכריע בקביעת היעילות והאפקטיביות של תהליך האופטימיזציה. ה**Batch size** משפיע ביחס ישר על התכנסות המודל- זאת משום שגדלי קבוצות גדולים יותר גורמים בד"כ לעדכונים יציבים יותר והתכנסות מהירה יותר. כיוון שהן מספקות הערכות מדויקות יותר של הגראדינט של פונקציית ה**loss**. כלקח מאימון מודלים קודמים (אימנו את מודל 4 לפני שאימנו את מודל 2) ולכן אתחלנו את פרמטר ה **Batch Size**לערך הטוב ביותר שקיבלנו עבור מודל 4 שהוא 512=**Batch Size** .
שיננו את ערכו וראינו כי כאשר הקטנו אותו לערך 256 שמנו לב כי רמת הדיוק של המודל יורדת.
גם כאשר הגדלנו את ערכו של ה **Batch Size** לערך 1024 בניסיון להעלות את רמת הדיוק של המודל אך, גם עבור ערך זה ראינו כי רמת הדיוק של המודל פוחתת.
על כן הערך האופטימלי מבחינתנו הוא **Batch Size = 512** כפי שמוצג בטבלה.

- בשלב הבא, בדקנו את השפעת גודל השכבות הלינאריות (**features in, features out**) על דיוק המודל.

פרמטרים אלו קובעים את מספק הנוירונים **בשכבות הלינאריות** שברשת הנוירונים שבנינו עבור אימון המודל. מספר קטן של נוירונים **בשכבות הלינאריות** עלול לגרום ל**Underfitting** כיוון שהמודל לא מספיק ללמוד מה**dataset** את הדפוסים הבסיסיים מהנתונים מה שגורם לדיוק נמוך יותר. מאידך, מספר גבוה של נוירונים **בשכבות הלינאריות** עלול לגרום ל**Overfitting** כיוון שהמודל עלול ללמוד לשנן את נתוני האימון הללו ולא להכליל מהם דפוס התנהגות למקרים הכלליים יותר.
כמו כן זמן האימון מושפע מפרמטר זה. **שכבות לינאריות** גדולות מדי דורשות יותר חישוב במהלך האימון משום שיש המון פרמטרים לעדכן בזמן האימון- מה שגורם להארכת זמן האימון במיוחד ברשתות עמוקות. מאידך, **שכבות לינאריות** נמוכות מדי דורשת פחות חישוב אומנם אך עשויות להביא לרמות דיוק נמוכות כיוון שלא מספיקות ללכוד את התכונות הרלוונטיות ב**dataset**.
בתחילת האימון התחלנו עם הערכים הבאים:

FC1 input=1024
FC2 input=512
FC3 input=256
FC4 input=128
FC4 output=64

ראינו כי הגדלת ערכי הפרמטרים אל הערכים :
FC1 input=2048
FC2 input=1024
FC3 input=512
FC4 input=256
FC4 output=128
אכן משפרת את ביצועי המודל אך, כאשר ניסנו לבצע הגדלה נוספת אל הערכים :

FC1 input=4096
FC2 input=2048
FC3 input=1024
FC4 input=512
FC4 output=256

דווקא קיבלנו כי המודל פחות מדייק ולכן קיבענו את הפרמטרים לערכים :
FC4 , **FC4 input=256**, **FC3 input=512**, **FC2 input=1024**,**FC1 input=2048**
output=128.

- בשלב האחרון, בדקנו את השפעת גודל השפעת **dropout rate** לכל אחת מארבעת השכבות הלינאריות כחלק מתהליך ה**Dropout** בשלב האימון על דיוק המודל.

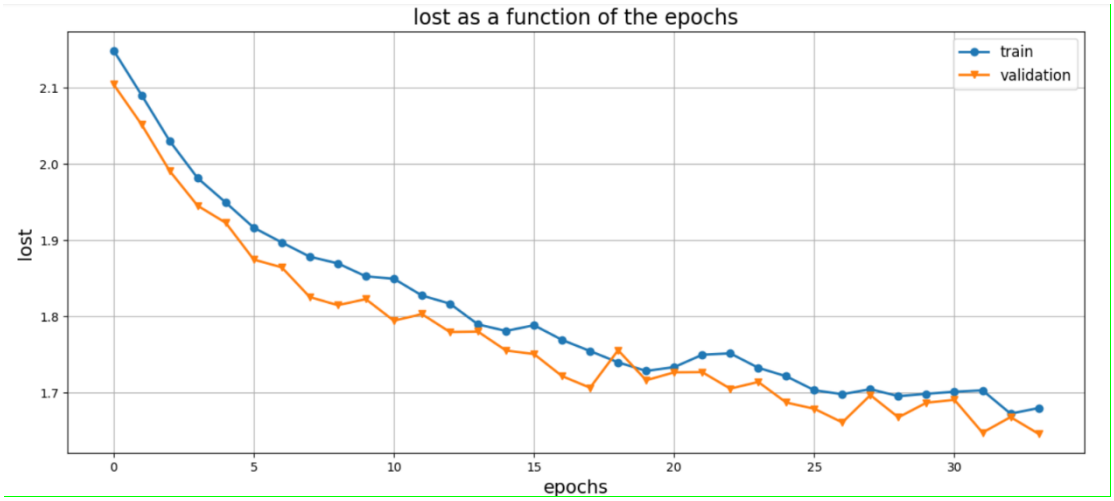
פרמטר זה קובע את ההסתברות לניתוק נוירון X מן השכבה שעליה מופעל ה**dropout** במקרה של מודל זה מודל בשכבה הלינארית ברשת הנוירונים. על מנת לאמן מספר רב של רשתות בו זמנית , בשלב האימון אנו מייצרים מספר רב של רשתות בו זמנית כך שבכל פעם מנתקים נוירונים אחרים מהרשת(לכל נוירון אנו מגרילים בכל mini batch מספר רנדומלי בין 0 ל 1 ובמידה ומספר זה קטן יותר מה **dropout rate** אזי באותו mini batch אנו מנתקים אותו מהרשת).
באופן זה, בכל mini batch נוצרת רשת חדשה אשר דומה לרשת המקורית בשינויים קלים.
דרך זו מאפשרת למנוע **Overfitting** בשלב אימון המודל כיוון שזה מונע מהמודל הסתמכות וקיבעון לסט מסוים של תכונות. כמו כן, שיעור **dropout rate** גבוה מדי עלול להוביל לחוסר התאמה- כיוון שיש יותר מדי יחידות שמנותקות בשלב האימון מה שמגביל את יכולת המודל ללמוד את ה **dataset** בצורה נכונה. מאידך, שיעור **drop out** נמוך מדי עלול שלא לספק רגולריזציה מספקת.
מהטבלה ניתן לראות שהתחלנו בערכים הבאים :
FC4 , **FC3 Dropout=0.03**,**FC2 Dropout=0.08 ,FC1 Dropout=0.09**
Dropout=0.01.
תוך כדי תהליך האימון ומציאת ההיפר פרמטרים הטובים ביותר ראינו כי הגדלת ערכי הפרמטרים אל הערכים :
FC4 Dropout=0.03, **FC3 Dropout=0.05**,**FC2 Dropout=0.08 ,FC1 Dropout=0.1** .

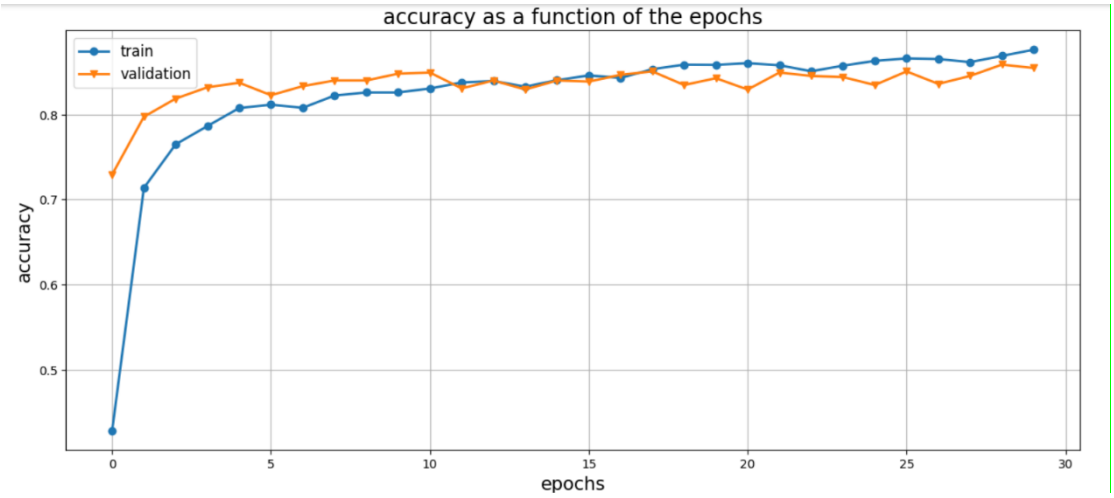
אכן משפרת את ביצועי המודל אך, כאשר ניסנו לבצע הגדלה נוספת אל הערכים :
FC4 Dropout=0.05, **FC3 Dropout=0.08**,**FC2 Dropout=0.15 ,FC1 Dropout=0.2** .

דווקא קיבלנו כי המודל פחות מדייק ולכן קיבענו את הפרמטרים לערכים :
FC4 Dropout=0.03, **FC3 Dropout=0.05**,**FC2 Dropout=0.08 ,FC1 Dropout=0.1**

- יתר על כן במהלך תהליך האימון בדקנו גם פונקציות loss שונות כגון cross entropy ו The negative log likelihood loss , מצאנו כי פונקציית loss הנותנת דיוק המירבי עבור המודל הינה **cross entropy** (כפי שניתן לראות בטבלה).
בנוסף לכך, בדקנו סוגים שונים של optimizer כמו ADAM, SGD, RMSprob וראינו כי הoptimizer שנותן דיוק רב ביותר הינו **Adam** (כמו שמופיע בטבלה).

כעת נציג את הגרפים המתארים את ערכי ה**loss** וה **accuracy** כפונקציה של מספר ה **epochs** עבור סט ה**Train** וה**Validation**:





מהגרפים לעיל ניתן לראות כי ככל שמספר ה**epoch**ים עולה ← **loss** יורד, **accuracy** עולה. בהתאם לתיאוריה שהסברנו לעיל וכפי שציפינו שיקרה. עם הגדלת מספר ה**epoch**ים המודל הופך מאומן יותר, מכיר את ה**data** בצורה טובה. ה**loss** פוחת כיוון שמכל **epoch** המודל לומד מנתוני האימון ומעדכן את המשקלים בהתאם על מנת למזער את פונקציית ה**loss**. לכן ככל שיש יותר **epoch**ים המודל מקבל יותר הזדמנויות להתאים את הפרמטרים שלו לנתוני האימון וכך להיות מדויק יותר עד לפתרון האופטימלי.

וכן ישנה עלייה ב**accuracy** כיוון שככל שהמודל לומד יותר מנתוני האימון במהלך כל **epoch** הוא משתפר בביצוע התחזיות. השיפור הנ"ל ביכולות החיזוי מוביל לדיוק גבוה יותר, כמצופה.

כמו כן, נשים לב כי אחוז ה**accuracy** הנמדד עבור ה**train** הינו גבוה יותר מאשר אחוז ה**accuracy** הנמדד עבור סט ה**validation**, הסיבה המרכזית לכך הינה שהמודל רואה את סט ה**train** הכי הרבה (כל **mini batch** ולא כל**epoch** כמו ה**validation**) ולכן המודל יותר מותאם (overfit) עבורו ומצליח לדייק באופן מירבי יותר מאשר סט ה**validation**.

כעת, אחרי שסיימנו את תהליך אימון המודל ומצאנו את ערכי ההיפר פרמטרים הטובים ביותר - **FC4 Dropout=0.03**, **FC3 Dropout=0.05**,**FC2 Dropout=0.08** ,**FC1 Dropout=0.1**

FC4 , FC4 input=256, FC3 input=512, FC2 input=1024,FC1 input=2048 output=128
batch size=512, num of epochs=34, lr=0.001, weight decay=0.001, num of linear layers=4, optimizer=Adam, loss function=cross entropy

בדקנו את ערכי ה **loss, accuracy** של המודל בזמן ה**test** כלומר, על הtestset ומצאנו כי עבור מודל זה :

```
Test Loss: 1.696 | Test Accuracy: 37.663%
```

לבסוף, נסביר את מספר הפרמטרים שניתנים לאימון במודל זה, המודל הינו ANN המורכב מ6 שכבות לינאריות fully connected וביניהן שכבת input , ארבע שכבות ביניים (hidden layers) ושכבת output.
הפרמטרים המתקבלים ממודל זה נובעים מ:
1. כל ששת השכבות הלינאריות במודל (input, output, 4hidden)הינן שכבות לינאריות שהן fully connected כלומר ,בכל שכבה ישנה קשת המחברת בין כל אחד מהנוירונים בכניסה לשכבה (features in) לבין כל אחד מהנוירונים ביציאה מהשכבה(features out). כל קשת כזאת תורמת משקל (פרמטר) הנלמד במהלך תהליך האימון. בנוסף, כל אחד מהנוירונים ביציאה של השכבה הלינארית (features out) תורם לנו פרמטר נוסף שהינו biasה (biasה מהווה את הרעש של כל נוירון).
2.batch normalization – את הbatch normalization אנו מפעילים על כל אחת משכבות הלינאריות שהן hidden בלבד (סך הכל 4 שכבות ללא שכבות הinput, output) על ידי הפעלת הbatch normalization כל נוירון ביציאה מהשכבה הלינארית(features out)תורם לנו שני פרמטרים אחד עבור scaling והשני עבור shifting .

לכן החישוב המספרי הינו :

שכבת input :

$$features\ in \cdot features\ out + bias_{fetures\ out} = 12288 \cdot 2048 + 2048 = 25{,}167{,}872$$

שכבת hidden 1 :

$$features\ in \cdot features\ out + bias_{fetures\ out} = 2048 \cdot 1024 + 1024 = 2{,}098{,}176$$

$$feature\ out \cdot 2 = 1024 \cdot 2 = 2048$$

שכבת hidden 2 :

$$features\ in \cdot features\ out + bias_{fetures\ out} = 1024 \cdot 512 + 512 = 524{,}800$$

$$feature\ out \cdot 2 = 512 \cdot 2 = 1024$$

שכבת hidden 3 :

$$features\ in \cdot features\ out + bias_{fetures\ out} = 512 \cdot 256 + 256 = 131{,}328$$

$$feature\ out \cdot 2 = 256 \cdot 2 = 512$$

שכבת hidden 4 :

$$features\ in \cdot features\ out + bias_{fetures\ out} = 256 \cdot 128 + 128 = 32{,}896$$

$$feature\ out \cdot 2 = 128 \cdot 2 = 256$$

שכבת output:

$$features\ in \cdot features\ out + bias_{fetures\ out} = 1280 \cdot 10 + 10 = 1290$$

סך הכל מספר הפרמטרים הנלמדים במודל זה הינו :

$$total\ trianable\ parmas = 25{,}167{,}872 + 2{,}098{,}176 + 2048 + 524{,}800$$
$$+1024 + 131{,}328 + 512 + 32{,}896 + 256 + 1290 = \mathbf{27{,}960{,}202}$$

כפי שניתן לראות בתמונה הבאה זה גם המספר שיצא לנו כאשר חשבנו את מספר הפרמטרים הנלמדים במודל במהלך תהליך האימון על ידי פקודת *summary* :

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
           Flatten-1                [-1, 12288]               0
            Linear-2                 [-1, 2048]      25,167,872
            Linear-3                 [-1, 1024]       2,098,176
       BatchNorm1d-4                 [-1, 1024]           2,048
           Dropout-5                 [-1, 1024]               0
            Linear-6                  [-1, 512]         524,800
       BatchNorm1d-7                  [-1, 512]           1,024
           Dropout-8                  [-1, 512]               0
            Linear-9                  [-1, 256]         131,328
      BatchNorm1d-10                  [-1, 256]             512
          Dropout-11                  [-1, 256]               0
           Linear-12                  [-1, 128]          32,896
      BatchNorm1d-13                  [-1, 128]             256
          Dropout-14                  [-1, 128]               0
           Linear-15                   [-1, 10]           1,290
================================================================
Total params: 27,960,202
Trainable params: 27,960,202
Non-trainable params: 0
```

3.3 CNN Model (Convolutional neural network):

המודל השלישי אותו נדרשנו לממש הינו CNN Model. CNN = Convolutional Neural Network. המודל מממש מעין רשת עצבית מלאכותית המתאימה במיוחד לעיבוד וניתוח רשתות מובנות של נתונים, כגון תמונות. רשתות CNN משיגות תוצאות מיטביות במשימות ראייה ממוחשבת שונות, כולל סיווג תמונות, זיהוי אובייקטים ופילוח תמונה.

לפי דרישות המטלה על המודל להכיל לפחות שתי שכבות convolution & pooling וכן שתי שכבות ליניאריות ושכבת classification. כמו כן ביצענו batch normalization לשכבות הconvolution ו dropout לשכבות הליניאריות.
על מנת לאמן את המודל בצורה המיטבית ניסינו להגדיל (החל מ2) את מספר שכבות ה convolution & pooling זאת בהתאם לדרישת המטלה. שכבות אלה דואגות להקטנת מימדי התמונות כך שהראייה תהיה יותר גלובלית

הערה: לכל שכבה לינארית יש פרמטר של features in ופרמטר של features out אך ה features in של השכבה ה i+1 שווה בגודלה לfeatures out של השכבה ה i לכן למרות שיש לנו 2 שכבות קונבולוציה או יותר המודל צריך ללמוד פחות היפר פרמטרים הקשורים לגדלי הfeatures .
על כן בהקשר להיפר פרמטרים הקשורים לגדלי שכבות הקונבולוציה אנו צריכים למצוא רק את הפרמטרים הבאים : conv input features out =conv1 features in
conv1 features out =conv2 features in
conv2 features out =conv3 features in

בנוסף לכך, ישנן היפר פרמטר שלא נוכל לשחק עם גודלם כמו features out של שכבת ה classification היות ומספר הclasses של הDATASET שלנו הינו 10 הגדול של פרמטר זה חייב להיות 10.וכן גודל ה features in של השכבה הראשונה שהוא קבוע על 3 בהתאם ל3 הממדים של התמונות הנתונות.

בשל כך רק את הגדלים שלהם נשנה ונציג בטבלה המראה את שינוי הפרמטרים בתהליך האימון.

על מנת להגיע לסט הפרמטרים אשר מניב את התוצאות המיטביות השתמשנו בשיטת "ניסוי וטעיה". בתחילה, אימנו את המודל עם סט פרמטרים התחלתי, קיבלנו ערך התחלתי ולא מספיק טוב, על כן המשכנו לאמן את המודל כך שבכל הרצה בדקנו את השפעת שינוי אחד הפרמטרים על תוצאות ה **Accuracy** עבור ה**Validation**. המשכנו בשלבים עד אשר הגענו לרמת דיוק טובה מספיק.
לבסוף הרצנו את הפרמטרים הטובים ביותר על מנת למצוא את ה**accuracy** עבור סט ה**test.**

להלן פירוט התהליך שעשינו ושינוי הפרמטרים בהתאמה:

accuracy	Loss function	optimizer	FC1 output	FC2 output	Num of conv &pool layers	Conv2 input	Conv2 output	Conv3 output	Pool1	Pool2	Pool3	Max pool /avg poo	Weight decay	LR	Num epochs	Batch size	Dropout FC1	Dropout FC2
50.667%	Cross Entropy	ADAM	200	100	2	6	12	-	2	2	-	Max pool	0.001	0.001	20	64	0.03	0.03
50.800%	Cross Entropy	ADAM	200	100	2	6	12	-	2	4	-	Max pool	0.001	0.001	20	64	0.03	0.03
43.733%	Cross Entropy	ADAM	200	100	2	6	12	-	2	8	-	Max pool	0.001	0.001	20	64	0.03	0.03

40.000%	Cross Entropy	ADAM	200	100	2	6	12	-	8	4	-	Max pool	0.001	0.001	20	64	0.03		03
50.800%	Cross Entropy	ADAM	200	100	2	6	12	-	2	4	-	Max pool	0.0001	0.01	20	64	0.03		03
50.800%	Cross Entropy	ADAM	200	100	2	6	12	-	2	4	-	Max pool	0.03	0.001	20	64	0.03		03
50.800%	Cross Entropy	ADAM	200	100	2	6	12	-	2	4	-	Max pool	0.001	0.0001	20	64	0.03		03
53.061%	Cross Entropy	ADAM	200	100	2	6	12	-	2	4	-	Max pool	0.001	0.01	40	64	0.03		03
57.200%	Cross Entropy	ADAM	200	100	2	6	12	-	2	4	-	Max pool	0.001	0.01	60	64	0.03		03
58.933%	Cross Entropy	ADAM	200	100	2	6	12	-	2	4	-	Max pool	0.001	0.01	60	32	0.03		03
58.133%	Cross Entropy	ADAM	200	100	2	6	12	-	2	4	-	Max pool	0.001	0.01	60	16	0.03		03
58.400%	Cross Entropy	ADAM	200	100	2	6	12	-	2	4	-	Max pool	0.001	0.01	60	128	0.03		03
54.133%	Cross Entropy	ADAM	200	100	2	6	12	-	2	4	-	Max pool	0.001	0.01	60	256	0.03		03
49.333%	Cross Entropy	ADAM	200	100	2	6	12	-	2	4	-	Max pool	0.001	0.01	60	512	0.03		03
57.333%	Cross Entropy	ADAM	200	100	2	6	12	-	2	4	-	Max pool	0.001	0.01	60	32	0.003		03
56.800%	Cross Entropy	ADAM	200	100	2	6	12	-	2	4	-	Max pool	0.001	0.01	60	32	0.03		006
57.333%	Cross Entropy	ADAM	200	100	2	6	12	-	2	4	-	Max pool	0.001	0.01	60	32	0.03		0.1
59.067%	Cross	ADAM	200	100	3	6	12	24	2	4	2	Max pool	0.001	0.01	60	32	0.03		0.1

	Entropy																		
56.933%	Cross Entropy	RMS Prop	200	100	3	6	12	24	2	4	2	Max pool	0.001	0.01	60	32	0.03		0.1
43.600%	Cross Entropy	SGD	200	100	3	6	12	24	2	4	2	Max pool	0.001	0.01	60	32	0.03		0.1
60.933%	Cross Entropy	RMS Prop	200	100	3	12	12	42	2	4	2	Max pool	0.001	0.01	60	32	0.03		0.1
62.800%	Cross Entropy	RMS Prop	200	100	3	32	32	52	2	4	2	Max pool	0.001	0.01	60	32	0.03		0.1
63.333%	Cross Entropy	RMS Prop	200	100	3	64	80	92	2	4	2	Max pool	0.001	0.01	60	32	0.03		0.1
63.867%	Cross Entropy	RMS Prop	200	100	3	100	120	140	2	4	2	Max pool	0.001	0.01	60	32	0.03		0.1
56.400%	Cross Entropy	ADAM	300	100	3	6	12	24	2	4	2	Max pool	0.001	0.01	60	32	0.03		0.1
57.067%	Cross Entropy	ADAM	200	200	3	6	12	24	2	4	2	Max pool	0.001	0.01	60	32	0.03		0.1
54.533%	Cross Entropy	ADAM	200	100	3	6	12	24	2	4	2	Avg pool	0.001	0.01	60	32	0.03		0.1

פירוט השפעת **hyperparameters** על רמת דיוק המודל:

תחילה בחרנו שרירותית את הנתונים ההתחלתיים על מנת לקבל איזושהי תמונה כללית על המודל ונקודה התחלתית. ניתן לראות שהגענו לערך: **Accuracy = 50.676%**

- השפעת **מספר שכבות הקונבולוציה והPooling (conv & pooling layers)** על דיוק המודל:

לפי הגדרות המטלה על המודל להכיל לפחות 2 שכבות קונבולוציה. על כן התחלנו ב2 שכבות והעלנו בהדרגה ל3. נוכחנו לגלות כי בהגדלת מספר השכבות הללו ללא שינוי נוסף בשאר ההיפרפרמטרים קיבלנו רמת דיוק גבוהה יותר של המודל.
ניתן להסביר זאת על ידי מטרת שכבות אלה: שכבות הPooling עוזרות להפחית ממדים מרחביים תוך שמירה על תכונות חשובות של הDataset. המספר והסידור של שכבות אלו יכולים להשפיע על עומק התכונות הנלמדות בפרט, ועל יכולתו של המודל להכליל וללמוד סוגים שונים של Datasets ככלל. המספר האופטימלי של שכבות אלו עשוי להשתנות בהתאם למורכבות ולגודל של מערך הנתונים. מערכי נתונים מסוימים עשויים להפיק תועלת

מארכיטקטורות עם מספר גדול יותר של שכבות כדי ללכוד דפוסים מורכבים, בעוד שאחרים עשויים להשיג ביצועים טובים יותר עם מודלים פשוטים יותר למניעת overfitting. במקרה של הDataset הנתון- תמונות בעלות 3 ממדים 64X64X3 שהיו יחסית פשוטות הספיק מספר נמוך של שכבות קונבולוציה וpooling.

- השפעת ערך ה **Learning Rate** על דיוק המודל.

כידוע קצב הלמידה משפיע ישירות על המהירות וההתכנסות של אלגוריתם האופטימיזציה המשמש לאימון, בחירת קצב למידה מתאים יכולה להשפיע משמעותית על הדיוק וביצועי המודל. **Learning Rate** נמוך מדי עלול להביא לפתרון לא אופטימלי נוסף על זמן התכנסות ארוך- **Underfitting**. מאידך, **Learning rate** גבוה מדי עלול לגרום למודל להתנודד סביב הפתרון האופטימלי ולהוביל ל**Overfitting**. שמנו לב כי בהעלאת ערך ה **Learning Rate** לערך 0.01 קיבלנו שרמת הדיוק עלתה. וכן רמת הדיוק היתה מטיבית בערך **lr = 0.01** - זאת בהתאם למצופה שה**lr** לא צריך להיות גבוה מדי אך לא נמוך מדי.

- לאחר מכן, בדקנו את השפעת ה**regularization** בעזרת פרמטר **weight decay** על דיוק המודל שלנו.

פרמטר זה "מעניש" משקלים גדולים מדי ומנמיך את ערכם ובכך מונע **Overfitting**, מביא לשליטה מיטבית במורכבות הדגם ושיפורו. בהתחלה קבענו את הערך **weight decay** =0.001
ניסנו לשנות את ערכו של הפרמטר אך ראינו כי בעת הגדלת ערך הפרמטר לערך 0.003 או הקטנת ערך הפרמטר אל הערך **0.0001** דיוק המודל נשאר בערך זהה וללא שינוי משמעותי. קיבענו על הערך **weight decay=0.01** והמשכנו באימון המודל.

- בשלב הבא, בדקנו את השפעת גודל **Num of Epochs** על דיוק המודל.

פרמטר זה מייצג את מספר הפעמים שה **training dataset** מועבר קדימה ואחורה דרך המודל בתהליך האימון. כמות נמוכה של **epochs** עלולה לגרום **לUnderfitting** כיוון שהמודל לא מספיק ללמוד מה**dataset** וגורם לדיוק נמוך יותר. מאידך, כמות גבוהה של **epochs** עלולה לגרום **לOverfitting** כיוון שהמודל לומד רעש **מה**dataset ולכן ישנה חוסר הצלחה בהכללת הלמידה לdata חדש אשר לא נראה קודם. על כן, התחלנו ב**epochs=20** והעלנו את הערך עד ל60, שמנו לב כי בהעלאת הערך קיבלנו רמת דיוק גבוהה יותר, אך בהעלאה יותר מדי רמת הדיוק קטנה, לכן קיבענו את הפרמטר לערך של 60 והמשכנו באימון המודל.

- בשלב הבא, בדקנו את השפעת גודל ה**Batch Size** על דיוק המודל.

פרמטר זה קובע את מספר הדגימות המעובדות לפני עדכון הפרמטרים של המודל במהלך תהליך האימון. הוא ממלא תפקיד מכריע בקביעת היעילות והאפקטיביות של תהליך האופטימיזציה. ה**Batch size** משפיע ביחס ישר על התכנסות המודל- זאת משום שגדלי קבוצות גדולים יותר גורמים בד"כ לעדכונים יציבים יותר והתכנסות מהירה יותר. כיוון שהן מספקות הערכות מדויקות יותר של הגראדינט של פונקציית ה**loss**.
התחלנו מערך התחלתי של batch size = 64 וניסינו להעלות/להקטין את הערך על מנת להבין כיצד הערך ישפיע על דיוק המודל. נוכחנו לגלות כי בהקטנת הערך ל batch size = 32 רמת הדיוק עלתה משמעותית, וכשהמשכנו להוריד לערך של 16 היא ירדה משמעותית. גם כאשר הגדלנו את ערכו של ה **Batch Size** לערך 128 ו 512 בניסיון להעלות את רמת הדיוק של המודל ראינו כי רמת הדיוק של המודל ירדה.
על כן הערך האופטימלי מבחינתנו הוא **Batch Size =32** כפי שמוצג בטבלה.

- בשלב הבא, בדקנו את השפעת גודל השכבות הלינאריות (**features in, features out**)על דיוק המודל.

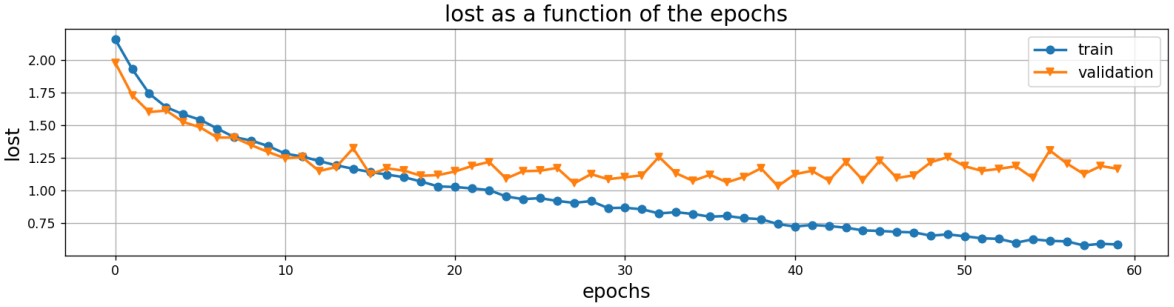
פרמטרים אלו קובעים את מספק הנוירונים **בשכבות הלינאריות** שברשת הנוירונים שבנינו עבור אימון המודל. מספר קטן של נוירונים **בשכבות הלינאריות** עלול לגרום ל**Underfitting** כיוון שהמודל לא מספיק ללמוד מה**dataset** את הדפוסים הבסיסיים מהנתונים מה שגורם לדיוק נמוך יותר. מאידך, מספר גבוה של נוירונים **בשכבות הלינאריות** עלול לגרום ל**Overfitting** כיוון שהמודל עלול ללמוד לשנן את נתוני האימון הללו ולא להכליל מהם דפוס התנהגות למקרים הכלליים יותר.
כמו כן זמן האימון מושפע מפרמטר זה. **שכבות לינאריות** גדולות מדי דורשות יותר חישוב במהלך האימון משום שיש המון פרמטרים לעדכן בזמן האימון- מה שגורם להארכת זמן האימון במיוחד ברשתות עמוקות. מאידך, **שכבות לינאריות** נמוכות מדי דורשת פחות חישוב אומנם אך עשויות להביא לרמות דיוק נמוכות כיוון שלא מספיקות ללכוד את התכונות הרלוונטיות ב**dataset**.
התחלנו מערך התחלתי רנדומלי של FC1 Output = 200, FC2 Output = 100.
שמנו לב כי בהעלאת/הקטנת הערכים הללו בהדרגה ובאופן מבוקר (על מנת להימנע ממצב של Overfitting) קיבלנו שרמת הדיוק ירדה- לכן נשארנו עם הערכים ההתחלתיים והמשכנו לאמן את המודל.

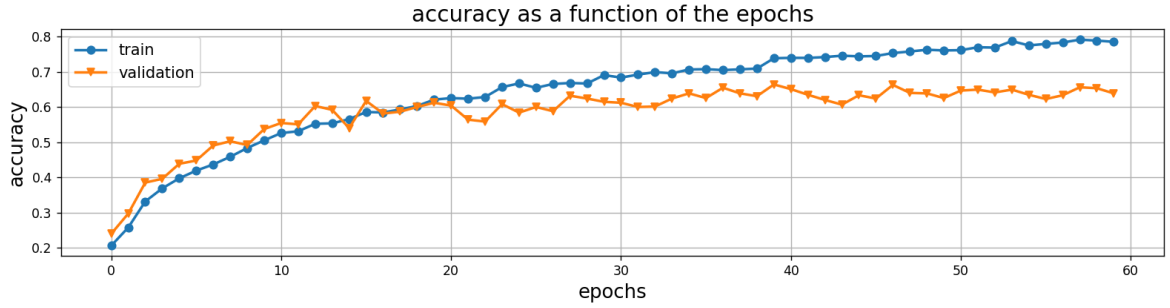
- בשלב האחרון, בדקנו את השפעת גודל השפעת **dropout rate** על השכבות הלינאריות כחלק מתהליך ה**Dropout** בשלב האימון על דיוק המודל.

פרמטר זה קובע את ההסתברות לניתוק נוירון X מן השכבה שעליה מופעל ה**dropout** במקרה של מודל זה מודל בשכבה הלינארית ברשת הנוירונים. על מנת לאמן מספר רב של רשתות בו זמנית , בשלב האימון אנו מייצרים מספר רב של רשתות בו זמנית כך שבכל פעם מנתקים נוירונים אחרים מהרשת(לכל נוירון אנו מגרילים בכל mini batch מספר רנדומלי בין 0 ל 1 ובמידה ומספר זה קטן יותר מה **dropout rate** אזי באותו mini batch אנו מנתקים אותו מהרשת).
באופן זה, בכל mini batch נוצרת רשת חדשה אשר דומה לרשת המקורית בשינויים קלים.
דרך זו מאפשרת למנוע **Overfitting** בשלב אימון המודל כיוון שזה מונע מהמודל הסתמכות וקיבעון לסט מסוים של תכונות. כמו כן, שיעור **dropout rate** גבוה מדי עלול להוביל לחוסר התאמה- כיוון שיש יותר מדי יחידות שמנותקות בשלב האימון מה שמגביל את יכולת המודל ללמוד את ה **dataset** בצורה נכונה. מאידך, שיעור **drop out** נמוך מדי עלול שלא לספק רגולריזציה מספקת.
מהטבלה ניתן לראות שהתחלנו בערכים הבאים dropout FC1 = 0.03, dropout FC2= 0.03
שמנו לב כי בהקטנת ערכים אלה לערכי: dropout FC1 = 0.003, dropout FC2= 0.03 ו dropout FC1 = 0.03, dropout FC2= 0.006 רמת דיוק המודל ירדה. לכן ניסינו להעלות את הערך ל dropout FC1 = 0.03, dropout FC2= 0.1 וגילינו כי רמת הדיוק עלתה! על כן נשארנו עם ערך זה להמשך אימון המודל.

- יתר על כן במהלך תהליך האימון בדקנו גם פונקציות loss שונות כגון cross entropy ו The negative log likelihood loss , מצאנו כי פונקציית loss הנותנת דיוק המירבי עבור המודל הינה **cross entropy** (כפי שניתן לראות בטבלה).
בנוסף לכך, בדקנו סוגים שונים של optimizer כמו ADAM, SGD, RMSprob וראינו כי הoptimizer שנותן דיוק רב ביותר הינו RMSprob (כמו שמופיע בטבלה).

כעת נציג את הגרפים המתארים את ערכי ה**loss** וה **accuracy** כפונקציה של מספר ה **epochs** עבור סט ה**Train** וה**Validation**:





מהגרפים לעיל ניתן לראות כי ככל שמספר ה**epoch**ים עולה ← **loss** יורד, **accuracy** עולה. בהתאם לתיאוריה שהסברנו לעיל וכפי שציפינו שיקרה. עם הגדלת מספר ה**epoch**ים המודל הופך מאומן יותר, מכיר את ה**data** בצורה טובה. ה**loss** פוחת כיוון שמכל **epoch** המודל לומד מנתוני האימון ומעדכן את המשקלים בהתאם על מנת למזער את פונקציית ה**loss**. לכן ככל שיש יותר **epoch**ים המודל מקבל יותר הזדמנויות להתאים את הפרמטרים שלו לנתוני האימון וכך להיות מדויק יותר עד לפתרון האופטימלי.

וכן ישנה עלייה ב**accuracy** כיוון שככל שהמודל לומד יותר מנתוני האימון במהלך כל **epoch** הוא משתפר בביצוע התחזיות. השיפור הנ"ל ביכולות החיזוי מוביל לדיוק גבוה יותר, כמצופה.

כמו כן, נשים לב כי אחוז ה**accuracy** הנמדד עבור ה**train** הינו גבוה יותר מאשר אחוז ה**accuracy** הנמדד עבור סט ה**validation**, הסיבה המרכזית לכך הינה שהמודל רואה את סט ה**train** הכי הרבה (כל **mini batch** ולא כל**epoch** כמו ה**validation**) ולכן המודל יותר מותאם (overfit) עבורו ומצליח לדייק באופן מירבי יותר מאשר סט ה**validation**.

לבסוף אחרי שסיימנו את תהליך אימון המודל ומצאנו את ערכי ההיפר פרמטרים הטובים ביותר -
FC2 output=100, FC1 output=200,FC2 Dropout=0.03 ,FC1 Dropout=0.1 num of conv layers= 3 , conv2 input= 100, conv2 output= 120, conv3 output= 140, Pool1= 2, Pool2= 4, Pool3= 2, Max Pool, ,batch size=512, num of epochs= 60, lr=0.01, weight decay=0.001, optimizer=RMSProp, loss function=cross entropy, batch size = 32

בדקנו את ערכי ה **loss, accuracy** של המודל בזמן ה**test** כלומר, על הtestset ומצאנו כי עבור מודל זה :

```
Test Loss: 1.193 | Test Accuracy: 57.575%
```

לבסוף, נסביר את מספר הפרמטרים שניתנים לאימון במודל הCNN. המודל מורכב מ2 שכבות ליניאריות fully connected ו3 שכבות convolution, ושכבת output.
הפרמטרים המתקבלים ממודל זה נובעים מ:
1. השכבות הלינאריות במודל הינן שכבות ליניאריות שהן fully connected כלומר, בכל שכבה ישנה קשת המחברת בין כל אחד מהנוירונים בכניסה לשכבה (features in) לבין כל אחד מהנוירונים ביציאה מהשכבה(features out). כל קשת כזאת תורמת משקל (פרמטר) הנלמד במהלך תהליך האימון. בנוסף, כל אחד מהנוירונים ביציאה של השכבה הלינארית (features out) תורם לנו פרמטר נוסף שהינו הbias (הbias מהווה את הרעש של כל נוירון).
2.batch normalization – את הbatch normalization אנו מפעילים על כל אחת משכבות הקונבולוציה. על ידי הפעלת הbatch normalization כל נוירון ביציאה מהשכבה הלינארית(features out)תורם לנו שני פרמטרים אחד עבור scaling והשני עבור shifting .

3.שכבות הקונבולוציה- עבור כל שכבת קונבולוציה ישנם filters הניתנים ללמידה, כך שכל filter יוצר תכונת פלט אחת. כדי לחשב את מספר הפרמטרים עבור כל filter- השתמשנו במימדי הfilter תוך התחשבות במספר הערוצי הקלט והפלט.

לכן החישוב המספרי הינו :

שכבת Conv1 :

$$(in\ channels \cdot kernel\ height * kernel\ width + 1) * out\ channels$$
$$= (3 * 3 * 3 + 1) * 12 = 336$$

שכבת Conv2 :

$$(in\ channels \cdot kernel\ height * kernel\ width + 1) * out\ channels$$
$$= (12 * 3 * 3 + 1) * 12 = 1{,}308$$

שכבת Conv3 :

$$(in\ channels \cdot kernel\ height * kernel\ width + 1) * out\ channels$$
$$= (12 * 3 * 3 + 1) * 42 = 4{,}578$$

שכבת hidden 1 :

$$features\ in \cdot features\ out + bias_{dropout} = 672 * 200 + 200 = 134{,}600$$

שכבת hidden 2 :

$$features\ in \cdot features\ out + bias_{dropout} = 200 * 100 + 100 = 20{,}100$$

שכבת output:

$$features\ in \cdot features\ out + bias_{fetures\ out} = 100 \cdot 10 + 10 = 1{,}010$$

סך הכל מספר הפרמטרים הנלמדים במודל זה הינו :

$$total\ trianable\ parmas = 336 + 1{,}308 + 4{,}578 + 24 + 24 + 84 + 134{,}600 + 20{,}100 + 1{,}010 = \mathbf{162{,}064}$$

כפי שניתן לראות בתמונה הבאה זה גם המספר שיצא לנו כאשר חשבנו את מספר הפרמטרים הנלמדים במודל במהלך תהליך האימון על ידי פקודת *summary* :

```
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 12, 62, 62]             336
       BatchNorm2d-2           [-1, 12, 62, 62]              24
         MaxPool2d-3           [-1, 12, 31, 31]               0
            Conv2d-4           [-1, 12, 29, 29]           1,308
       BatchNorm2d-5           [-1, 12, 29, 29]              24
         MaxPool2d-6           [-1, 12, 14, 14]               0
            Conv2d-7           [-1, 42, 12, 12]           4,578
       BatchNorm2d-8           [-1, 42, 12, 12]              84
         MaxPool2d-9             [-1, 42, 4, 4]               0
           Flatten-10                  [-1, 672]               0
            Linear-11                  [-1, 200]         134,600
           Dropout-12                  [-1, 200]               0
            Linear-13                  [-1, 100]          20,100
           Dropout-14                  [-1, 100]               0
            Linear-15                   [-1, 10]           1,010
================================================================
Total params: 162,064
Trainable params: 162,064
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 1.07
Params size (MB): 0.62
Estimated Total Size (MB): 1.74
----------------------------------------------------------------
None
```

3.4 fixed pre-trained MobileNetV2:

על מנת ליצור בעזרת מודל זה multi-class classification, אנו נשתמש במודל קיים הנקרא MobileNetV2 לא נשנה את המשקלים שלו, אלא נשתמש במשקלים הקיימים שלו ונוסיף לו 3 שכבות לינאריות – 2 שכבות fully connected ושכבת classification אנו נלמד את המשקלים רק עבור השכבות שאותם הוספנו.

הערה:
לכל שכבה לינארית יש פרמטר של features in ופרמטר של features out אך הfeatures in של השכה ה i+1 שווה בגודלה לfeatures out של השכבה הi לכן למרות שיש לנו 3 שכבות לינאריות המודל צריך ללמוד פחות מ6 היפר פרמטרים הקשורים לגדלי הfeatures .
בנוסף לכך, ישנן היפר פרמטר שלא נוכל לשחק עם גודלם כמו features out של שכבת ה classification היות ומספר הclasses של הDATASET שלנו הינו 10 הגדול של פרמטר זה חייב להיות 10.
כמו כן, גם ההיפר פרמטר של features in של השכבה הלינארית הראשונה נקבע מהארכיטקטורה של הרשת MobileNetV2 ולכן הערך שלו קבוע ולא בר שינוי (מצאנו את הערך שהינו 1000 בהינתן המימדים של ה data שלנו בעזרת הפונקציה הסטטית שיצרנו במחלקה של המודל הנקראת clac_input_size).

לכן, בהקשר להיפר פרמטרים הקשורים לגדלי השכבות הלינאריות אנו צריכים למצוא רק את הפרמטרים הבאים : fc1 features out= fc2 features in
fc2 features out= classification features in

בשל כך רק את הגדלים שלהם נשנה ונציג בטבלה המראה את שינוי הפרמטרים בתהליך האימון.

על מנת להגיע לסט הפרמטרים אשר מניב את התוצאות המיטביות השתמשנו בשיטת "ניסוי וטעיה". בתחילה, אימנו את המודל עם סט פרמטרים התחלתי, קיבלנו ערך התחלתי ולא מספיק טוב, על כן המשכנו לאמן את המודל כך שבכל הרצה בדקנו את השפעת שינוי אחד הפרמטרים על תוצאות ה **Accuracy** עבור ה**Validation**. המשכנו בשלבים עד אשר הגענו לרמת דיוק טובה מספיק.
לבסוף הרצנו את הפרמטרים הטובים ביותר על מנת למצוא את ה**accuracy** עבור סט ה**test.**

להלן פירוט התהליך שעשינו ושינוי הפרמטרים בהתאמה:

accuarcy	Loss function	optimazer	FC2 output	FC1 output	Weight decay	LR	Num epochs	Batch size	Dropout FC1	Dropout FC2
76.933	Cross Entropy	ADAM	100	200	0.0001	0.001	20	64	0.08	0.03
74.533	Cross Entropy	ADAM	100	400	0.0001	0.001	20	64	0.08	0.03
74.933	Cross Entropy	ADAM	100	150	0.0001	0.001	20	64	0.08	0.03
74.667	Cross Entropy	ADAM	150	200	0.0001	0.001	20	64	0.08	0.03
75.867	Cross Entropy	ADAM	50	200	0.0001	0.001	20	64	0.08	0.03
74.933	Cross Entropy	ADAM	100	200	0.0001	0.001	20	64	0.1	0.03
75.867	Cross Entropy	ADAM	100	200	0.0001	0.001	20	64	0.05	0.03
70.667	Cross Entropy	ADAM	100	200	0.0001	0.001	20	64	0.08	0.05
74.800	Cross Entropy	ADAM	100	200	0.0001	0.001	20	64	0.08	0.01
76.267	Cross Entropy	ADAM	100	200	0.00005	0.001	20	64	0.08	0.03
77.733	Cross Entropy	ADAM	100	200	0.001	0.001	20	64	0.08	0.03
76.800	Cross Entropy	ADAM	100	200	0.005	0.001	20	64	0.08	0.03
77.732	Cross Entropy	ADAM	100	200	0.001	0.005	20	64	0.08	0.03
77.732	Cross Entropy	ADAM	100	200	0.001	0.0001	20	64	0.08	0.03
75.467	Cross Entropy	ADAM	100	200	0.001	0.001	20	32	0.08	0.03

78.533%	Cross Entropy	ADAM	100	200	0.001	0.001	20	128	0.08	0.03
80.133	Cross Entropy	ADAM	100	200	0.001	0.001	20	256	0.08	0.03
84.264	Cross Entropy	ADAM	100	200	0.001	0.001	20	512	0.08	0.03
16.235	The negative log likelihood loss	ADAM	100	200	0.001	0.001	20	512	0.08	0.03
49.200	Cross Entropy	SGD	100	200	0.001	0.001	20	512	0.08	0.03
83.532	Cross Entropy	RMSProp	100	200	0.001	0.001	20	512	0.08	0.03
84.000	Cross Entropy	ADAM	100	200	0.001	0.001	50	512	0.08	0.03
85.467	Cross Entropy	ADAM	100	200	0.001	0.001	30	512	0.08	0.03

פירוט השפעת **hyperparameters** על רמת דיוק המודל:

תחילה בחרנו שרירותית את הנתונים ההתחלתיים על מנת לקבל איזושהי תמונה כללית על המודל ונקודה התחלתית. ניתן לראות שהגענו לערך: **Accuracy = 76.933%**

- השפעת ערך ה **Learning Rate** על דיוק המודל.

כידוע קצב הלמידה משפיע ישירות על המהירות וההתכנסות של אלגוריתם האופטימיזציה המשמש לאימון, בחירת קצב למידה מתאים יכולה להשפיע משמעותית על הדיוק וביצועי המודל. **Learning Rate** נמוך מדי עלול להביא לפתרון לא אופטימלי נוסף על זמן התכנסות ארוך- **Underfitting**. מאידך, **Learning rate** גבוה מדי עלול לגרום למודל להתנודד סביב הפתרון האופטימלי ולהוביל ל**Overfitting**. שמנו לב כי בהעלאת ערך ה **Learning Rate** לערך 0.05 והורדת ערך ה **Learning Rate** לערך 0.0001 קיבלנו ירידה בדיוק המודל, רמת הדיוק היתה מיטבית בערך **lr = 0.001** - זאת בהתאם למצופה שה**lr** לא צריך להיות גבוה מדי אך לא נמוך מדי.

- לאחר מכן, בדקנו את השפעת ה **regularization** בעזרת פרמטר **weight decay** על דיוק המודל שלנו.

פרמטר זה "מעניש" משקלים גדולים מדי ומנמיך את ערכם ובכך מונע **Overfitting**, מביא לשליטה מיטבית במורכבות הדגם ושיפורו. תחילה קבענו 0.0001= **weight decay** ניסינו להקטין את ערכו לערך **0.00005** אך, רמת הדיוק של המודל ירדה ולכן העלנו את ערכו לערכים **0.001** ו**0.005** כאשר עבור הערך 0.001 קיבלנו שיפור בדיוק המודל ועבור הערך 0.005 קיבלנו ירידה בדיוק המודל.
לכן, קיבלנו כי עבור הערך **weight decay=0.001** אנו מקבלים את רמת הדיוק המיטבי וקיבענו את פרמטר זה לערך הנ"ל.

- בשלב הבא, בדקנו את השפעת גודל **Num of Epochs** על דיוק המודל.

פרמטר זה מייצג את מספר הפעמים שה **training dataset** מועבר קדימה ואחורה דרך המודל בתהליך האימון. כמות נמוכה של **epochs** עלולה לגרום **לUnderfitting** כיוון שהמודל לא מספיק ללמוד מה**dataset** וגורם לדיוק נמוך יותר. מאידך, כמות גבוהה של **epochs** עלולה לגרום **לOverfitting** כיוון שהמודל לומד רעש **מהdataset** ולכן ישנה חוסר הצלחה בהכללת הלמידה לdata חדש אשר לא נראה קודם. על כן, התחלנו ב**epochs=20** ונסינו להעלות את הערך, שמנו לב כי ב**epochs=50** קיבלנו ירידה בדיוק של המודל, אך עבור הערך **epochs=30** קיבלנו דיוק גדול יותר ולכן קיבענו את הפרמטר לערך זה.

- בשלב הבא, בדקנו את השפעת גודל ה**Batch Size** על דיוק המודל.

פרמטר זה קובע את מספר הדגימות המעובדות לפני עדכון הפרמטרים של המודל במהלך תהליך האימון. הוא ממלא תפקיד מכריע בקביעת היעילות והאפקטיביות של תהליך האופטימיזציה. ה**Batch size** משפיע ביחס ישר על התכנסות המודל- זאת משום שגדלי קבוצות גדולים יותר גורמים בד"כ לעדכונים יציבים יותר והתכנסות מהירה יותר. כיוון שהן מספקות הערכות מדויקות יותר של הגראדינט של פונקציית ה**loss**. התחלנו בערך **Batch Size**=64 הקטנו אותו לערך 32 אך, שמנו לב כי רמת הדיוק של המודל יורדת. לכן התחלנו להעלות את ערכו של **Batch Size** עד לערך 512 שבו קיבלנו את רמת הדיוק המקסימלית עבור המודל.
על כן הערך האופטימלי מבחינתנו הוא **Batch Size = 512** כפי שמוצג בטבלה.

- בשלב הבא, בדקנו את השפעת גודל השכבות הלינאריות (**features in, features out**)על דיוק המודל.

פרמטרים אלו קובעים את מספק הנוירונים **בשכבות הלינאריות** שברשת הנוירונים שבנינו עבור אימון המודל. מספר קטן של נוירונים **בשכבות הלינאריות** עלול לגרום ל**Underfitting** כיוון שהמודל לא מספיק ללמוד מה**dataset** את הדפוסים הבסיסיים מהנתונים מה שגורם לדיוק נמוך יותר. מאידך, מספר גבוה של נוירונים **בשכבות הלינאריות** עלול לגרום ל**Overfitting** כיוון שהמודל עלול ללמוד לשנן את נתוני האימון הללו ולא להכליל מהם דפוס התנהגות למקרים הכלליים יותר.
כמו כן זמן האימון מושפע מפרמטר זה. **שכבות לינאריות** גדולות מדי דורשות יותר חישוב במהלך האימון משום שיש המון פרמטרים לעדכן בזמן האימון- מה שגורם להארכת זמן האימון במיוחד ברשתות עמוקות. מאידך, **שכבות לינאריות** נמוכות מדי דורשת פחות חישוב אומנם אך עשויות להביא לרמות דיוק נמוכות כיוון שלא מספיקות ללכוד את התכונות הרלוונטיות ב**dataset**.
מהטבלה ניתן לראות שהתחלנו בערכים הבאים : **FC2 output=100 ,FC1 output=200**
כל שינוי שנסינו לערוך על ערך הפרמטרים הללו הקטנה או הגדלה רק גרמה למודל להיות פחות מדויק.
לכן קיבענו את הפרמטרים על הערכים שאיתם התחלנו **FC2 ,FC1 output=200 output=100**.

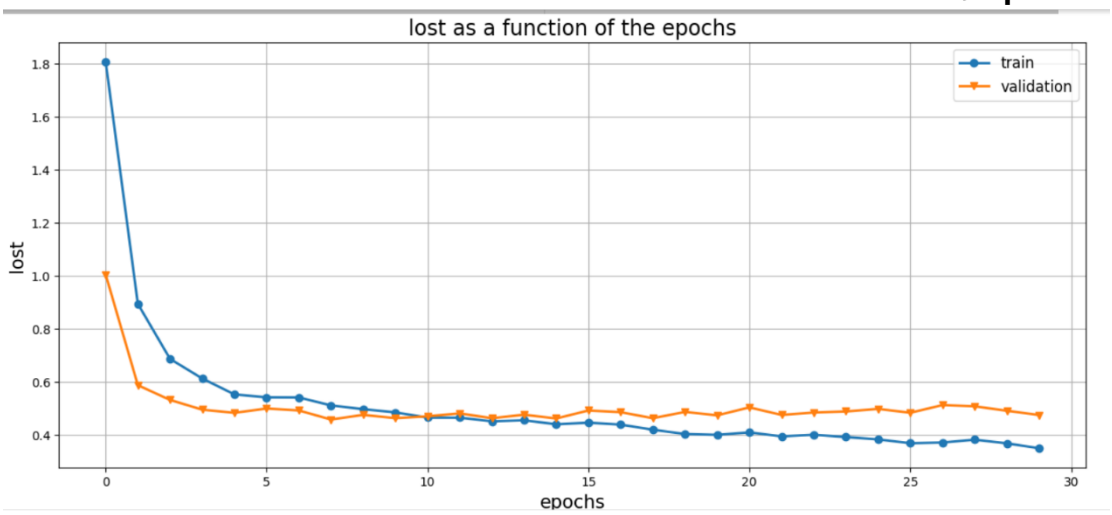
- בשלב האחרון, בדקנו את השפעת גודל השפעת **dropout rate** לכל אחת משתי שכבות הלינאריות כחלק מתהליך ה**Dropout** בשלב האימון על דיוק המודל.

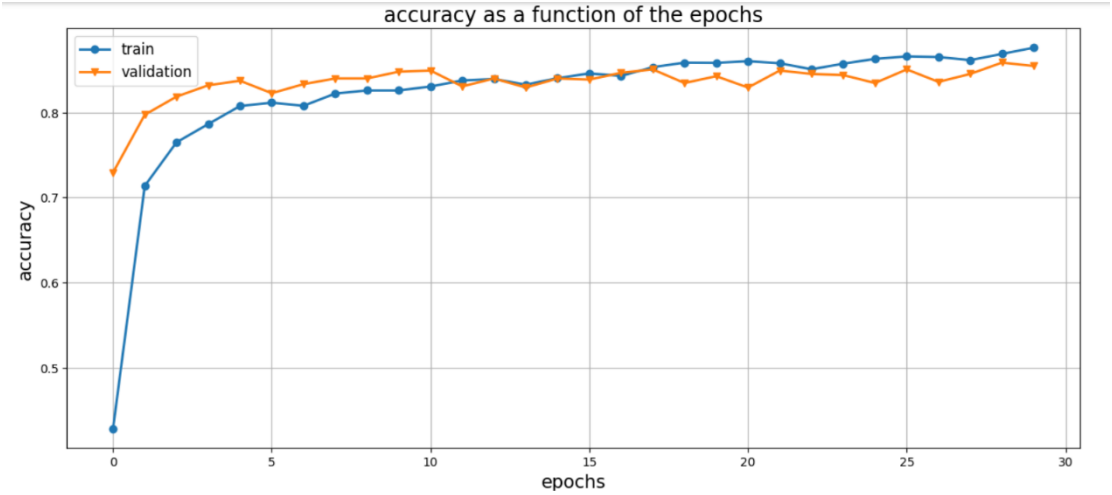
פרמטר זה קובע את ההסתברות לניתוק נוירון X מן השכבה שעליה מופעל ה**dropout** במקרה של מודל זה מודל בשכבה הלינארית ברשת הנוירונים. על מנת לאמן מספר רב של רשתות בו זמנית , בשלב האימון אנו מייצרים מספר רב של רשתות בו זמנית כך שבכל פעם מנתקים נוירונים אחרים מהרשת(לכל נוירון אנו מגרילים בכל mini batch מספר רנדומלי בין 0 ל 1 ובמידה ומספר זה קטן יותר מה **dropout rate** אזי באותו mini batch אנו מנתקים אותו מהרשת).
באופן זה, בכל mini batch נוצרת רשת חדשה אשר דומה לרשת המקורית בשינויים קלים.
דרך זו מאפשרת למנוע **Overfitting** בשלב אימון המודל כיוון שזה מונע מהמודל הסתמכות

וקיבעון לסט מסוים של תכונות. כמו כן, שיעור **dropout rate** גבוה מדי עלול להוביל לחוסר התאמה- כיוון שיש יותר מדי יחידות שמנותקות בשלב האימון מה שמגביל את יכולת המודל ללמוד את ה **dataset** בצורה נכונה. מאידך, שיעור **drop out** נמוך מדי עלול שלא לספק רגולריזציה מספקת.
מהטבלה ניתן לראות שהתחלנו בערכים הבאים : **FC2 ,FC1 Dropout=0.08**
Dropout=0.03
כל שינוי שנסינו לערוך על ערך הפרמטרים הללו הקטנה או הגדלה רק גרמה למודל להיות פחות מדוייק.
לכן קיבענו את הפרמטרים על הערכים שאיתם התחלנו **FC2 ,FC1 Dropout=0.08**
Dropout=0.03.

- יתר על כן במהלך תהליך האימון בדקנו גם פונקציות loss שונות כגון cross entropy ו The negative log likelihood loss , מצאנו כי פונקציית loss הנותנת דיוק המירבי עבור המודל הינה **cross entropy** (כפי שניתן לראות בטבלה).
בנוסף לכך, בדקנו סוגים שונים של optimizer כמו ADAM, SGD, RMSprob וראינו כי הoptimizer שנותן דיוק רב ביותר הינו **Adam** (כמו שמופיע בטבלה).

כעת נציג את הגרפים המתארים את ערכי ה**loss** וה **accuracy** כפונקציה של מספר ה **epochs** עבור סט ה**Train** וה**Validation**:





מהגרפים לעיל ניתן לראות כי ככל שמספר ה**epoch**ים עולה ← **loss** יורד, **accuracy** עולה. בהתאם לתיאוריה שהסברנו לעיל וכפי שציפינו שיקרה. עם הגדלת מספר ה**epoch**ים המודל הופך מאומן יותר, מכיר את ה**data** בצורה טובה. ה**loss** פוחת כיוון שמכל **epoch** המודל לומד מנתוני האימון ומעדכן את המשקלים בהתאם על מנת למזער את פונקציית ה**loss**. לכן ככל שיש יותר **epoch**ים המודל מקבל יותר הזדמנויות להתאים את הפרמטרים שלו לנתוני האימון וכך להיות מדויק יותר עד לפתרון האופטימלי.

וכן ישנה עלייה ב**accuracy** כיוון שככל שהמודל לומד יותר מנתוני האימון במהלך כל **epoch** הוא משתפר בביצוע התחזיות. השיפור הנ"ל ביכולות החיזוי מוביל לדיוק גבוה יותר, כמצופה.

כמו כן, נשים לב כי אחוז ה**accuracy** הנמדד עבור ה**train** הינו גבוה יותר מאשר אחוז ה**accuracy** הנמדד עבור סט ה**validation**, הסיבה המרכזית לכך הינה שהמודל רואה את סט ה**train** הכי הרבה (כל **mini batch** ולא כל**epoch** כמו ה**validation**) ולכן המודל יותר מותאם (overfit) עבורו ומצליח לדייק באופן מירבי יותר מאשר סט ה**validation**.

לבסוף אחרי שסיימנו את תהליך אימון המודל ומצאנו את ערכי ההיפר פרמטרים הטובים ביותר -
FC2 output=100, FC1 output=200,FC2 Dropout=0.03 ,FC1 Dropout=0.08
batch size=512, num of epochs= 30, lr=0.001, weight decay=0.001,
optimizer=Adam, loss function=cross entropy

בדקנו את ערכי ה **loss, accuracy** של המודל בזמן ה**test** כלומר, על הtestset ומצאנו כי עבור מודל זה :

```
Test Loss: 0.518 | Test Accuracy: 82.938%
```

3.5 MobileNetV2 pre-trained learned:

על מנת ליצור בעזרת מודל זה multi-class classification, אנו נשתמש במודל קיים הנקרא MobileNetV2 הפעם ניתן אפשרות לשנות את המשקלים הקיימים של המודל MobileNetV2 במהלך תהליך האימון ובנוסף, נוסיף לו 3 שכבות לינאריות – 2 שכבות fully connected ושכבת classification ונלמד את המשקלים גם עבור השכבות שאותם הוספנו.

הערה:
לכל שכבה לינארית יש פרמטר של features in ופרמטר של features out אך הfeatures in של השכה ה i+1 שווה בגודלה לfeatures out של השכבה הi לכן למרות שיש לנו 3 שכבות לינאריות המודל צריך ללמוד פחות מ6 היפר פרמטרים הקשורים לגדלי הfeatures .
בנוסף לכך, ישנן היפר פרמטר שלא נוכל לשחק עם גודלם כמו features out של שכבת ה classification היות ומספר הclasses של הDATASET שלנו הינו 10 הגדול של פרמטר זה חייב להיות 10.
כמו כן, גם ההיפר פרמטר של features in של השכבה הלינארית הראשונה נקבע מהארכיטקטורה של הרשת MobileNetV2 ולכן הערך שלו קבוע ולא בר שינוי (מצאנו את הערך שהינו 1000 בהינתן המימדים של ה data שלנו בעזרת הפונקציה הסטטית שיצרנו במחלקה של המודל הנקראת clac_input_size).

לכן, בהקשר להיפר פרמטרים הקשורים לגדלי השכבות הלינאריות אנו צריכים למצוא רק את הפרמטרים הבאים : fc1 features out= fc2 features in
fc2 features out= classification features in

בשל כך רק את הגדלים שלהם נשנה ונציג בטבלה המראה את שינוי הפרמטרים בתהליך האימון.

על מנת להגיע לסט הפרמטרים אשר מניב את התוצאות המיטביות השתמשנו בשיטת "ניסוי וטעיה". בתחילה, אימנו את המודל עם סט פרמטרים התחלתי, קיבלנו ערך התחלתי ולא מספיק טוב, על כן המשכנו לאמן את המודל כך שבכל הרצה בדקנו את השפעת שינוי אחד הפרמטרים על תוצאות ה **Accuracy** עבור ה**Validation**. המשכנו בשלבים עד אשר הגענו לרמת דיוק טובה מספיק.
לבסוף הרצנו את הפרמטרים הטובים ביותר על מנת למצוא את ה**accuracy** עבור סט ה**test.**

להלן פירוט התהליך שעשינו ושינוי הפרמטרים בהתאמה:

accuarcy	Loss function	optimizer	FC2 output	FC1 output	Weight decay	LR	Num epochs	Batch size	Dropout FC1	Dropout FC2
9.733	Cross Entropy	ADAM	100	200	0.0001	0.001	20	64	0.08	0.03
73.467	Cross Entropy	ADAM	100	200	0.0001	0.001	20	128	0.08	0.03
81.333	Cross Entropy	ADAM	100	200	0.0001	0.001	20	256	0.08	0.03
84.400	Cross Entropy	ADAM	100	200	0.0001	0.001	20	512	0.08	0.03
78.800	Cross Entropy	ADAM	100	200	0.0001	0.001	20	1024	0.08	0.03
81.733	Cross Entropy	ADAM	100	400	0.0001	0.001	20	512	0.08	0.03
81.467	Cross Entropy	ADAM	100	150	0.0001	0.001	20	512	0.08	0.03
80.933	Cross Entropy	ADAM	150	200	0.0001	0.001	20	512	0.08	0.03
85.333	Cross Entropy	ADAM	50	200	0.0001	0.001	20	512	0.08	0.03
55.200%	Cross Entropy	ADAM	30	200	0.0001	0.001	20	512	0.08	0.03
10.800	Cross Entropy	ADAM	50	200	0.0001	0.001	20	512	0.1	0.03
82.933	Cross Entropy	ADAM	50	200	0.0001	0.001	20	512	0.05	0.03
83.600	Cross Entropy	ADAM	50	200	0.0001	0.001	20	512	0.08	0.06
85.600	Cross Entropy	ADAM	50	200	0.0001	0.001	20	512	0.08	0.01
9.600	Cross Entropy	ADAM	50	200	0.0001	0.005	20	512	0.08	0.01

86.267	Cross Entropy	ADAM	50	200	0.0001	0.0005	20	512	0.08	0.01
87.067	Cross Entropy	ADAM	50	200	0.0005	0.0005	20	512	0.08	0.01
86.133	Cross Entropy	ADAM	50	200	0.001	0.0005	20	512	0.08	0.01
10.933	The negative log likelihood loss	ADAM	100	200	0.001	0.001	20	512	0.08	0.01
63.867	Cross Entropy	SGD	50	200	0.0005	0.0005	20	512	0.08	0.01
8.133	Cross Entropy	RMSProp	50	200	0.0005	0.0005	20	512	0.08	0.01
85.876	Cross Entropy	ADAM	50	200	0.0005	0.0005	50	512	0.08	0.01

פירוט השפעת **hyperparameters** על רמת דיוק המודל:

תחילה בחרנו שרירותית את הנתונים ההתחלתיים על מנת לקבל איזושהי תמונה כללית על המודל ונקודה התחלתית. ניתן לראות שהגענו לערך התחלתי נמוך יחסית לשאר המודלים:
Accuracy = 9.733%

- תחילה, בדקנו את השפעת גודל ה**Batch Size** על דיוק המודל.

פרמטר זה קובע את מספר הדגימות המעובדות לפני עדכון הפרמטרים של המודל במהלך תהליך האימון. הוא ממלא תפקיד מכריע בקביעת היעילות והאפקטיביות של תהליך האופטימיזציה. ה**Batch size** משפיע ביחס ישר על התכנסות המודל- זאת משום שגדלי קבוצות גדולים יותר גורמים בד"כ לעדכונים יציבים יותר והתכנסות מהירה יותר. כיוון שהן מספקות הערכות מדויקות יותר של הגראדינט של פונקציית ה**loss**. התחלנו בערך **Batch Size**=64 וראינו כי קיבלנו תוצאת accuracy נמוכה מאוד לכן הסקנו כי אין מספיק דגימות בכל batch ולכן אנו לא מצליחים להתכנס לערך יציב וגבוה יותר בתהליך הלמידה.
הגדלנו את ערכו של **Batch Size** ובכל פעם קיבלנו שיפור בדיוק המודל, עד לערך 512 שבו קיבלנו את רמת הדיוק המקסימלית עבור המודל.
כאשר ניסנו להגדיל את ערכו של **Batch Size** ל1024 דווקא קיבלנו ירידה בדיוק המודל.
על כן הערך האופטימלי מבחינתנו הוא **Batch Size = 512** כפי שמוצג בטבלה

- לאחר מכן, בדקנו את השפעת ה **regularization** בעזרת פרמטר **weight decay** על דיוק המודל שלנו.

פרמטר זה "מעניש" משקלים גדולים מדי ומנמיך את ערכם ובכך מונע **Overfitting**, מביא לשליטה מיטבית במורכבות הדגם ושיפורו. תחילה קבענו =0.0001 **weight decay** ניסינו להגדיל את ערכו לערך **0.0005** וקיבלנו כי רמת הדיוק של המודל עלתה לכן, המשכנו להגדיל את הפרמטר אל הערך **0.001** אך הפעם ראינו שרמת הדיוק של המודל דווקא ירדה. לכן הסקנו כי עבור הערך **weight decay=0.0005** אנו מקבלים את רמת הדיוק המיטבי וקיבענו את פרמטר זה לערך הנ"ל.

- בשלב הבא, בדקנו את השפעת גודל **Num of Epochs** על דיוק המודל.

פרמטר זה מייצג את מספר הפעמים שה **training dataset** מועבר קדימה ואחורה דרך המודל בתהליך האימון. כמות נמוכה של **epochs** עלולה לגרום **לUnderfitting** כיוון שהמודל לא מספיק ללמוד מה**dataset** וגורם לדיוק נמוך יותר. מאידך, כמות גבוהה של **epochs** עלולה לגרום **לOverfitting** כיוון שהמודל לומד רעש **מהdataset** ולכן ישנה חוסר הצלחה בהכללת הלמידה לdata חדש אשר לא נראה קודם. על כן, התחלנו ב**epochs=20** ונסינו להעלות את הערך, שמנו לב כי ב**epochs=50** קיבלנו ירידה בדיוק של המודל ובנוסף ראינו לפי הגרף של ערך הaccuracy של הvalidation כתלות במספר הepoch שערך הגדול ביותר מתקבל לאחר epochs20 ולאחר מכן אנו מקבלים ירידות ועליות אך לא מצליחים לעקוף את ערך הvalidation accuracy המתקבל לאחר 20 epochs.
כתוצאה מכך הסקנו כי עבור **epochs=20** המודל הינו הכי מדויק ולכן קיבענו את הפרמטר לערך זה.

- בשלב הבא, בדקנו את השפעת ערך ה **Learning Rate** על דיוק המודל.

כידוע קצב הלמידה משפיע ישירות על המהירות וההתכנסות של אלגוריתם האופטימיזציה המשמש לאימון, בחירת קצב למידה מתאים יכולה להשפיע משמעותית על הדיוק וביצועי המודל. **Learning Rate** נמוך מדי עלול להביא לפתרון לא אופטימלי נוסף על זמן התכנסות ארוך- **Underfitting**. מאידך, **Learning rate** גבוה מדי עלול לגרום למודל להתנודד סביב הפתרון האופטימלי ולהוביל ל**Overfitting**.
שמנו לב כי בהעלאת ערך ה **Learning Rate** לערך 0.005 קיבלנו ירידה בדיוק המודל אך, בהורדת ערך ה **Learning Rate** לערך 0.0005 דווקא קיבלנו עליה בדיוק המודל.
לכן כחלק מתהליך האימון הסקנו כי רמת הדיוק הינה מיטבית כאשר **lr = 0.0005** - זאת בהתאם למצופה שה**lr** לא צריך להיות גבוה מדי אך לא נמוך מדי.

- בשלב הבא, בדקנו את השפעת גודל השכבות הלינאריות (**features in, features out**) על דיוק המודל.

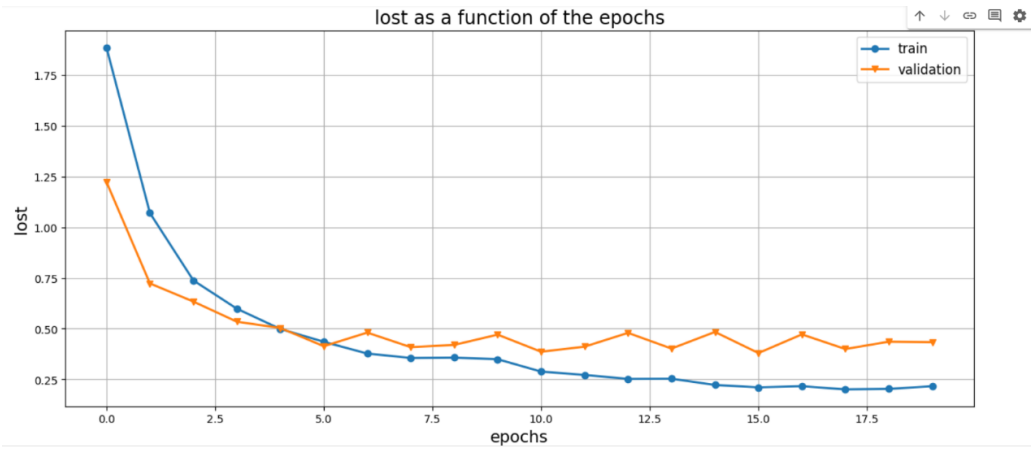
פרמטרים אלו קובעים את מספק הנוירונים **בשכבות הלינאריות** שברשת הנוירונים שבנינו עבור אימון המודל. מספר קטן של נוירונים **בשכבות הלינאריות** עלול לגרום ל**Underfitting** כיוון שהמודל לא מספיק ללמוד מה**dataset** את הדפוסים הבסיסיים מהנתונים מה שגורם לדיוק נמוך יותר. מאידך, מספר גבוה של נוירונים **בשכבות הלינאריות** עלול לגרום ל**Overfitting** כיוון שהמודל עלול ללמוד לשנן את נתוני האימון הללו ולא להכליל מהם דפוס התנהגות למקרים הכלליים יותר.
כמו כן זמן האימון מושפע מפרמטר זה. **שכבות לינאריות** גדולות מדי דורשות יותר חישוב במהלך האימון משום שיש המון פרמטרים לעדכן בזמן האימון- מה שגורם להארכת זמן האימון במיוחד ברשתות עמוקות. מאידך, **שכבות לינאריות** נמוכות מדי דורשת פחות חישוב אומנם אך עשויות להביא לרמות דיוק נמוכות כיוון שלא מספיקות ללכוד את התכונות הרלוונטיות ב**dataset**.
מהטבלה ניתן לראות שהתחלנו בערכים הבאים : **FC2 output=100 ,FC1 output=200**
כל שינוי שנסינו לערוך על ערך הפרמטר **FC1 output** ,הקטנה או הגדלה רק גרמה למודל להיות פחות מדויק.
בניגוד לכך, עבור ערך הפרמטר **FC2 output** אומנם הגדלת ערך הפרמטר לערך **150** אכן גרמה למודל להיות פחות מדויק אך, כאשר הקטנו את ערך הפרמטר לערך **50** קיבלנו מודל מדויק יותר.
כתוצאה מכך, נסינו להמשיך להקטין את ערך הפרמטר לערך **30** אך הפעם קיבלנו ירידה בדיוק המודל. בשל כך קיבענו את הפרמטרים על הערכים שעבורם קיבלנו את התוצאות המדויקות ביותר במהלך תהליך האימון- **FC2 output=50 ,FC1 output=200**.

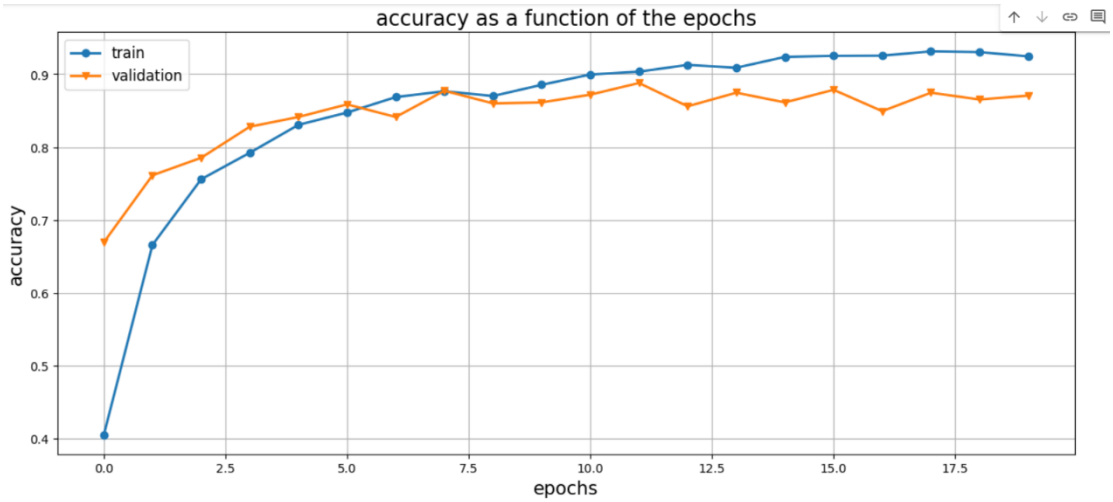
- בשלב האחרון, בדקנו את השפעת גודל השפעת **dropout rate** לכל אחת משתי שכבות הלינאריות כחלק מתהליך ה**Dropout** בשלב האימון על דיוק המודל.

פרמטר זה קובע את ההסתברות לניתוק נוירון X מן השכבה שעליה מופעל ה**dropout** במקרה של מודל זה מודל בשכבה הלינארית ברשת הנוירונים. על מנת לאמן מספר רב של רשתות בו זמנית , בשלב האימון אנו מייצרים מספר רב של רשתות בו זמנית כך שבכל פעם מנתקים נוירונים אחרים מהרשת(לכל נוירון אנו מגרילים בכל mini batch מספר רנדומלי בין 0 ל 1 ובמידה ומספר זה קטן יותר מה **dropout rate** אזי באותו mini batch אנו מנתקים אותו מהרשת).
באופן זה, בכל mini batch נוצרת רשת חדשה אשר דומה לרשת המקורית בשינויים קלים.
דרך זו מאפשרת למנוע **Overfitting** בשלב אימון המודל כיוון שזה מונע מהמודל הסתמכות וקיבעון לסט מסוים של תכונות. כמו כן, שיעור **dropout rate** גבוה מדי עלול להוביל לחוסר התאמה- כיוון שיש יותר מדי יחידות שמנותקות בשלב האימון מה שמגביל את יכולת המודל ללמוד את ה **dataset** בצורה נכונה. מאידך, שיעור **drop out** נמוך מדי עלול שלא לספק רגולריזציה מספקת.
בתחילת תהליך האימון אתחלנו את הפרמטרים בערכים הבאים : **FC1 Dropout=0.08,** **FC2 Dropout=0.03**
עבור הפרמטר **FC1 Dropout** כל שינוי שנסינו לערוך על ערך הפרמטר - הקטנה או הגדלה רק גרמה למודל להיות פחות מדויק.
לעומת זאת, עבור הפרמטר **FC1 Dropout** כאשר הגדלנו את ערכו לערך **0.06** קיבלנו מודל פחות מדויק אך, כאשר הקטנו את ערכו לערך **0.01** קיבלנו דווקא מודל מדויק יותר.
לכן קיבענו את הפרמטרים על הערכים שעבורם במהלך תהליך האימון קיבלנו את המודל בעל הדיוק הרב ביותר - **FC1 Dropout=0.08, FC2 Dropout=0.01**.

- יתר על כן במהלך תהליך האימון בדקנו גם פונקציות loss שונות כגון cross entropy ו The negative log likelihood loss , מצאנו כי פונקציית loss הנותנת דיוק המירבי עבור המודל הינה **cross entropy** (כפי שניתן לראות בטבלה).
בנוסף לכך, בדקנו סוגים שונים של optimizer כמו ADAM, SGD, RMSprob וראינו כי הoptimizer שנותן דיוק רב ביותר הינו **Adam** (כמו שמופיע בטבלה).

כעת נציג את הגרפים המתארים את ערכי ה**loss** וה **accuracy** כפונקציה של מספר ה **epochs** עבור סט ה**Train** וה**Validation**:





מהגרפים לעיל ניתן לראות כי ככל שמספר ה**epoch**ים עולה ← **loss** יורד, **accuracy** עולה. בהתאם לתיאוריה שהסברנו לעיל וכפי שציפינו שיקרה. עם הגדלת מספר ה**epoch**ים המודל הופך מאומן יותר, מכיר את ה**data** בצורה טובה. ה**loss** פוחת כיוון שמכל **epoch** המודל לומד מנתוני האימון ומעדכן את המשקלים בהתאם על מנת למזער את פונקציית ה**loss**. לכן ככל שיש יותר **epoch**ים המודל מקבל יותר הזדמנויות להתאים את הפרמטרים שלו לנתוני האימון וכך להיות מדויק יותר עד לפתרון האופטימלי.

וכן ישנה עלייה ב**accuracy** כיוון שככל שהמודל לומד יותר מנתוני האימון במהלך כל **epoch** הוא משתפר בביצוע התחזיות. השיפור הנ"ל ביכולות החיזוי מוביל לדיוק גבוה יותר, כמצופה.

כמו כן, נשים לב כי אחוז ה**accuracy** הנמדד עבור ה**train** הינו גבוה יותר מאשר אחוז ה**accuracy** הנמדד עבור סט ה**validation**, הסיבה המרכזית לכך הינה שהמודל רואה את סט ה**train** הכי הרבה (כל **mini batch** ולא כל**epoch** כמו ה**validation**) ולכן המודל יותר מותאם (overfit) עבורו ומצליח לדייק באופן מירבי יותר מאשר סט ה**validation**.

לבסוף אחרי שסיימנו את תהליך אימון המודל ומצאנו את ערכי ההיפר פרמטרים הטובים ביותר -
FC2 output=50, FC1 output=200,FC2 Dropout=0.01 ,FC1 Dropout=0.08
batch size=512, num of epochs= 20, lr=0.0005, weight decay=0.0005,
optimizer=Adam, loss function=cross entropy

בדקנו את ערכי ה **loss, accuracy** של המודל בזמן ה**test** כלומר, על הtestset ומצאנו כי עבור מודל זה :

```
Test Loss: 0.389 | Test Accuracy: 88.125%
```

Part 4- Summary:

בתרגיל זה התבקשנו לפתור משימות סיווג רב מימדיות עבור סוגים שונים של Neural Networks כגון: Ann, Cnn, MobileNetV2 . באמצעות PyTorch עבדנו על מערך הנתונים STL-10 הנתון אשר מורכב מ500 תמונות עבור סט הtrain ו800 תמונות מתויגות עבור סט הtest.

בשלב הראשון הצגנו את הData כפי שמבוקש וכן ביצענו Augmentations של היפוך וסיבוב על חלק מהData והראינו זאת בצורה ויזואלית. בשלבים הבאים בנינו את הרשתות עבור כל מודל, ביצענו אימון של הTrain על ידי כל אחד מן המודלים ולבסוף לאחר שקיבלנו את הפרמטרים שהביאו לרמת הדיוק הגבוהה ביותר- השתמשנו בהם עבור סט הTest.

מהתוצאות שפירטנו לעיל ניתן לראות כי המודל החלש ביותר הינו מודל ה Logistic Regression שהגיע לרמת דיוק של כ 24.7%. זאת משום פשטות וחוסר הליניאריות של המודל ביחס למורכבות הData שSTL-10 מייצג.

וכן המודל החזק ביותר הינו המודל learned pre-trained MobileNetV2 שהגיע לרמת דיוק של כ- 88.125%. זאת משום מורכבות המודל והיכולת שלו למדל את ה Data ולאמן אותו בצורה טובה .

כמו כן, שמנו לב כי המודלים מסודרים לפי רמת חזק- מהמודל החלש ביותר למודל החזק ביותר ביכולות האימון. כלומר בכל מודל שאימנו קיבלנו ערך accuracy גבוה יותר בהדרגתיות.