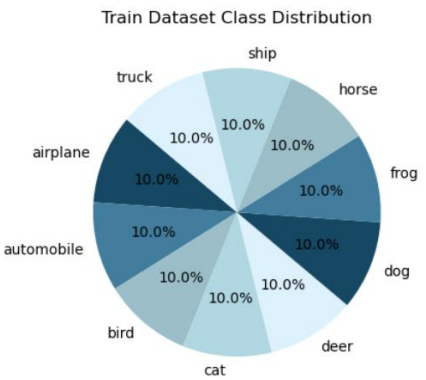
בס"ד

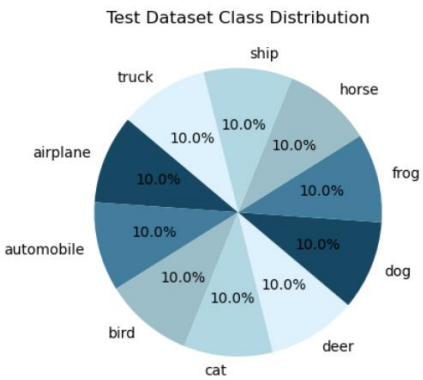
Project 2 - CNN

מגישה: גל חלילי

Data exploration:

```
Dataset labels - classes: ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
```





* ניתן לראות שכל המחלקות מתפלגות באופן אחיד גם בtrain וגם בtest – בשני הסטים בכל מחלקה יש 10% מדוגמאות הסט.

בס"ד

בניית הרשת:

מודל 1 –

התחלתי בניסוי הראשון עם שתי שכבות קונבולוציה ואחריהן 4 שכבות ליניאריות FC.

שכבת הקונבולוציה הראשונה לוקחת את התמונות ומפעילה עליהן 6 פילטרים בגודל 5X5.
השכבה השנייה לוקחת את הפלט של השכבה הקודמת (שזו בעצם activation map עם 6 ערוצים) ועליו מפעילה 12 פילטרים בגודל 5X5.

השכבה השלישית היא בעצם שכבת ה-FC הראשונה – יש בה 120 נוירונים, אחריה שכבה עם 60 נוירונים, לאחר מכן שכבה עם 40 נוירונים ולבסוף שכבת הפלט עם 10 נוירונים בהתאמה לכך שיש 10 מחלקות בבעיה זו.

בין כל שכבה ושכבה מופעלת פונקציית אקטיבציה RelU (בחרתי בה כי בהרצאות אמרנו שלרוב היא נותנת תוצאות טובות). בנוסף, פונקציית הלוס היא cross-entropy ושיטת האופטימיזציה היא SGD עם lr=0.001.

אחרי כל שכבת קונבולוציה הפעלתי שכבת Max-Pooling:

זו שכבה לא למידה שמטרתה להקטין את המימד של כל activation map בנפרד. זה סוג של פילטר שמחושב כמו פילטר רגיל אבל לוקח את הערך המקסימלי מכל חלון ולא סכום (כלומר עושה את המכפלות בין כל ערך בפילטר לבין הערך המתאים בחלון התמונה ובמרום לכלום את המכפלות, לוקח את המכפלה המקסימלית) – כך העצם מקטינים את מימד התמונה כי במקום לשמור את כל המשקולות, שומרים רק את המשקולות של הפיצאר הכי משמעותי מכל חלון.

בשכבות אלו לקחתי פילטר בגודל 2X2 עם stride 2.

*בשביל להכניס את הפלט של שכבת הmax pooling האחרונה לתוך שכבת ה FC הראשונה צריך "לשטח" את הטנזור לוקטור (חד מימדי).

*אחרי שכבת הפלט לא מופעלת RelU כי שם מופעלת softmax בשביל לקבל את ההסתברויות של להיות שייך לכל מחלקה.

גרף שמציג את הloss של סט האימון וסט הוולידציה:

הגעתי עם מודל זה ל accuracy = 0.6373 בוולידציה שזה לא דיוק מספק. בנוסף ניתן לראות פה overfitting.

בס"ד

מודל 2 –

אחרי שראיתי שנוצר במודל הראשון overfit ניסיתי להשתמש בdropout ו-early stopping בכדי למנוע זאת.

-Dropout

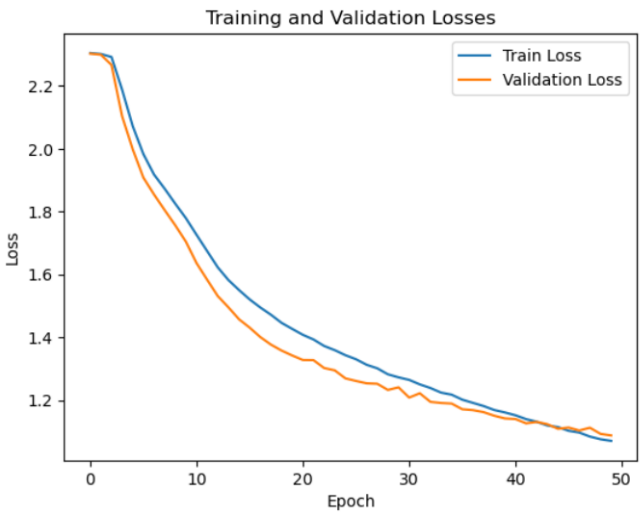
זו שכבה שמטרתה לאפס חלק מהנוירונים משמע לא להעביר דרכה את הערכים. מטרת השכבה ליצור הכללה ולמנוע overfitting. בכל איטרציה אנחנו משמיטים חלק מהנוירונים, נאפס כל נוירון בסיכוי p (p גדול יותר גורר יותר נוירונים מאופסים) כך שבעצם חסר לו מידע אחר כל פעם והוא עדיין מנסה להצליח (למזער את הלוס), מה שגורם לו "לא להיות תלוי" בקומבינציה הספציפית של הפיצ'רים שראה בסט האימון – כלומר לומד להכליל ולא לשנן את הקשרים הספציפיים שראה.

-Early stopping

גם מספר הepochים יכול להשפיע על overfit. נסתכל על ה-validation set loss (בקיצור vsl) ונחליט לעצור את האימון כך שהוא לא יגדל. בעצם השיטה אומרת שכל פעם שמזהים עלייה בvsl נשמור את הepoch בו זה קרה ונתחיל לספור n (לבחירתנו) epochs שבהם נבדוק אם זו אכן מגמת עלייה אמיתית או שזו הייתה עלייה לצורך ירידה. אם זו אכן מכמת עלייה ולא זיהינו מאז ירידה בvsl אז נחליט לעצור את הריצה ב epoch ששמרנו. אבל אם זיהינו מאז ירידה אז נמשיך בתהליך.

שיטה זו עוזרת במניעת overfitting בגלל שעליה ב loss של ה-validation מסמנת על overfit לכן לפי עלייה זו נחליט מתי לעצור בשביל לא להיכנס לoverfit. בגלל ששיטה זו עוזרת למנוע overfitting היא עוזרת למודל להכליל טוב יותר במקום לשנן כי עוצרת אותו לפני שהוא מתחיל ללמוד קשרים ותלויות שספציפיים לדוגמאות שהוא רואה בסט האימון ואם ילמד וישנן אותן הוא יהיה טוב מאוד על הtrain ופחות טוב על הvalidation ולכן הvsl יעלה.

השארתי את כל המודל כפי שהיה ורק הוספתי אחרי כל שכבת FC (חוץ משכבת הפלט כמובן) שכבת dropout שבה ה- dropout rate p=0.2. בנוסף, קבעתי שמספר האפוקים המקסימלי יהיה 100 וש- n (מספר האפוקים שבהם מחכים להשתפרות בלוס של הוולידציה) יהיה 10.



במודל זה הצלחתי להגיע ל – validation acc = 0.618

הצלחתי למנוע את ה overfitting בצורה משמעותית

אבל עדיין המודל לא לומד מספיק והביצועים נמוכים.

בס"ד

יכול להיות שהרשת שהגדרתי לא מספיק מורכבת בשביל הבעיה הזו וצריך יותר שכבות ויותר פילטרים בשביל לשפר את הלמידה.

מודל 3-

במודל זה שמתי 3 שכבות קונבולוציה ו3 שכבות FC.

שכבות הקונבולוציה:

בכל אחת גודל הפילטר הוא 3X3 עם padding 1.
בשכבה הראשונה מופעלים 32 פילטרים, בשכבה השנייה 64 ובשלישית 128.

אחרי כל שכבת קונבולוציה שמתי שכבת BatchNorm2d:

Batch Norm is a normalization technique done between the layers of a Neural Network instead of in the raw data. It is done along mini-batches instead of the full data set. It serves to speed up training and use higher learning rates, making learning easier.

לא למדנו על שיטה זו בהרצאות אך ממה שהבנתי מההסברים שקראתי (וצירפתי אחד מהם למעלה) זה ששכבה זו אמורה לסייע בייצוב והאצת תהליך האימון על ידי נרמול הקלט לכל שכבה.

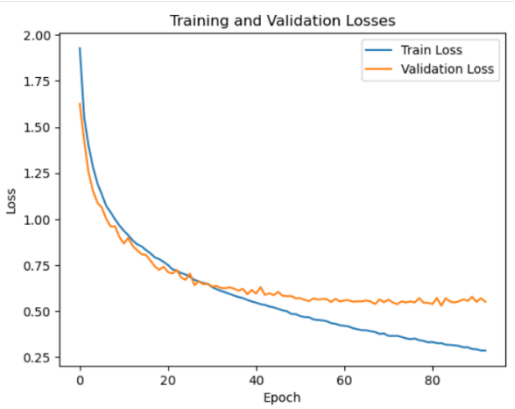
שכבות הFC:

בשכבה הראשונה יש 256 נוירונים, בשנייה 64 ובשכבה האחרונה-שכבת הפלט 10 נוירונים בהתאם לכך שיש 10 קלאסים.

שכבת ה-FC הראשונה מקבלת כקלט את הפלט משכבת הקונבולוציה אחרי שנהפך לוקטור (חד מימדי).

גם פה השתמשתי ב max pooling עם פילטר בגודל 2X2 ו-stride 2 אחרי כל שכבת קונבולוציה (אחרי הBatchNorm2). ושוב פונקציית האקטיבציה הייתה RelU עם lr=0.001.

*השארתי את שיטות מניעת ה-overfitting שהשתמשתי בהן במודל הקודם (p=0.2, n=10).



במודל זה validation accuracy = 0.8179

אכן יש שיפור משמעותי בביצועים (כמעט 20%)

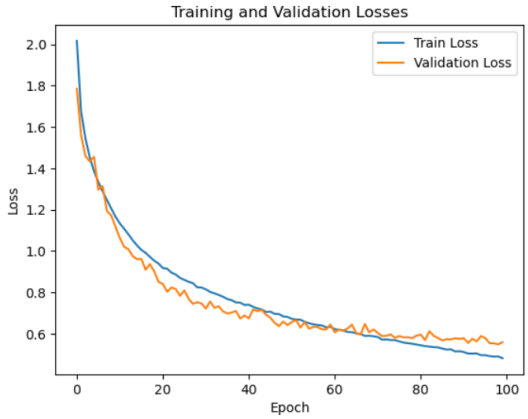
אבל שוב נוצר overfitting.

בס"ד

יכול להיות שעכשיו כשהרשת עמוקה יותר הלמידה נהייתה גם כן עמוקה וספציפית יותר וp וח שעזרו במודל 2 למנוע את ה-overfitting כעת לא ממש עוזרים. אנסה להגדיל את p ל-0.3.

מודל 4 -

מודל זה זהה למודל הקודם – רק הגדלתי את p ל-0.3.



במודל זה validation accuracy = 0.8123

הגדלת הp אכן עזרה במניעת overfitting אך עדיין יש שוני בין הלוס של האימון לוולידציה.

מה שניכר מגרף זה לדעתי זה שהגרף של הוולידציה מאוד רועש – אולי כדאי לשנות את שיטת האופטימיזציה.

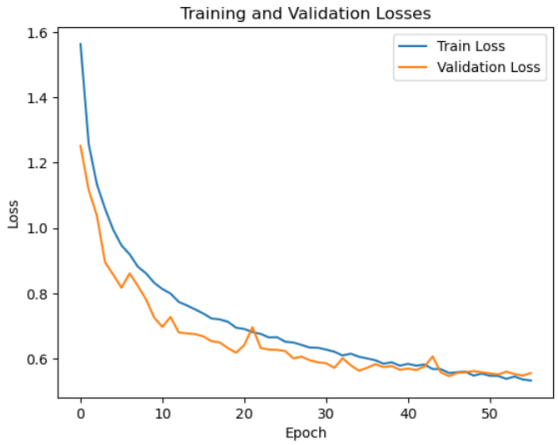
בס"ד

מודל 5 -

שוב השתמשתי באותה רשת רק שהפעם האופטימיזר הוא Adam.

זו שיטת אופטימיזיה שלא למדנו עליה בקורס אך ממה שקראתי והבנתי:
Adaptive Moment Estimation, הוא אלגוריתם קצב למידה אדפטיבי שנועד לשפר את מהירויות האימון ברשתות עצביות עמוקות ולהגיע להתכנסות במהירות. הוא מתאים אישית את קצב הלמידה של כל פרמטר על סמך היסטוריית השיפועים שלו, והתאמה זו עוזרת לרשת העצבית ללמוד ביעילות.

באלגוריתמים שאנחנו למדנו, קצב הלמידה α קבוע (hyperparameter), כלומר עלינו להתחיל בקצב למידה כשלהו ולשנות ידנית את האלפא לפי שלבים או לפי לוח זמנים למידה כלשהו. קצב למידה נמוך יותר בתחילתו יוביל להתכנסות איטית מאוד, בעוד ששיעור גבוה מאוד בהתחלה עלול להחמיץ את המינימום. Adam פותר בעיה זו על ידי התאמת קצב הלמידה α לכל פרמטר θ, מה שמאפשר התכנסות מהירה יותר בהשוואה לגרדיאנט דיסנט סטנדרטי עם קצב למידה קבוע.



במודל זה validation accuracy = 0.8131

הגרף אכן פחות רועש מהמודל הקודם וגם ה-overfitting

שהיה ירד.

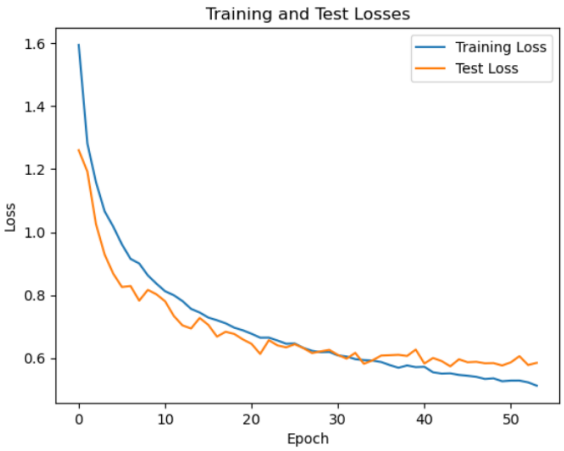
בנוסף, ניתן לראות שאכן התכנסנו הרבה יותר מהר מבמודל

הקודם- במודל 4 היו 100 אפוקים ואילו במודל הנוכחי קצת

יותר מ- 50.

כלומר המודל היה צריך קרוב לחצי מכמות האפוקים במודל 4!

בחרתי במודל האחרון וכעת אני אאמן את המודל על כל סט האימון (ללא וולידציה) ואז אשערך על ה-test set.



קיבלתי Test Accuracy = 0.8068

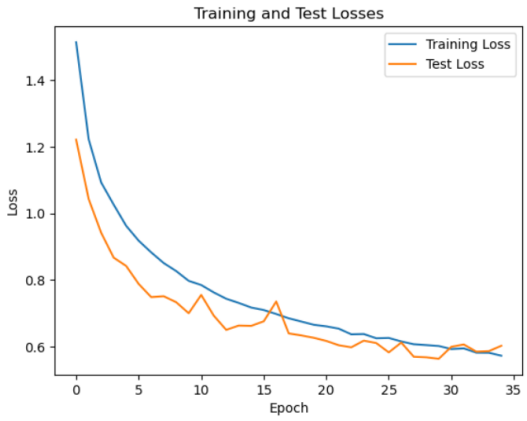
יש עדיין קצת overfitting בסוף.

אולי המודל למד קשר כלשהו שספציפי לדוגמאות מהאימון

בס"ד

והוולידציה ועכשיו בtest יש קצת חוסר התאמה ולכן יש overfitting.

אני אנסה להריץ את אותו מודל ורק להקטין את n ל-5 כי אולי אני נותנת יותר מידיי זמן השתפרות.



אכן ההקטנה של n עזרה להוריד את הoverfitting.

והצלחתי להגיע ל-Test Accuracy: 0.8065

הסבר מימדים של הרשת שנבחרה:

32X32X3 ➡ RelU(BatchNorm2d(Conv(in: 3, out 32, kernel: 3, paddind: 1) ➡ (32+2*1-3)/1 +1 ➡ 32X32X32

32X32X32 ➡ Max-Pooling(kernel: 2, stride: 2)➡(32-2)/2 +1 ➡ 16X16X32

Dropout()

16X16X32 ➡ RelU(BatchNorm2d(Conv(in: 32, out: 64, kernel: 3, paddind: 1)➡ (16+2*1-3)/1 +1 ➡ 16X16X64

16X16X64 ➡ Max-Pooling(kernel: 2, stride: 2) ➡(16-2)/2 +1➡ 8X8X64

Dropout()

8X8X64 ➡ RelU(BatchNorm2d(Conv(in: 64, out: 128, kernel: 3, paddind: 1) ➡ (8+2*1-3)/1 +1 ➡8X8X128

8X8X128 ➡ Max-Pooling(kernel: 2, stride: 2)➡(8-2)/2 +1➡ 4X4X128

Dropout()

Flatten 4X4X128 to 4*4*128

Dropout(RelU(Linear(in: 4*4*128, out: 256)

Dropout(RelU(Linear(in: 256, out: 128)

Linear(in: 128, out: 10)