מגישה: אריאל ברוך

ת"ז: 205925704

APML – תרגיל 1:

1. בדאטאסט שסופק לנו (train וdev) יש מספר בעיות:

- סביב ה1% מהדאטא מתוייג לא נכון. הבחנתי בכך על ידי כך שבחנתי מדגמית את הדאטא ומצאתי דגימות כאלו. עשיתי מעבר על הדאטא כדי לתקן את התיוגים הבעייתיים, דבר שהתאפשר בשל הגודל הקטן יחסית של שני הדאטאסטים.
- היחס בין מספר הדוגמאות מכל מחלקה מאוד לא שיוויוני, למשל:
  ב trainset היחס הוא: {'car': 4493, 'truck': 449, 'cat': 683}
  בdev set היחס הוא: {'car': 503, 'truck': 51, 'cat': 71}
  כלומר, בשני הסטים שסופקו לנו יש משמעותית יותר דגימות מסוג "car" מאשר שאר הקטגוריות.

2. ביצועי המודל שקיבלנו –

ביצועי המודל שסופק לנו נבחנו על ידי כך שהרצתי אותו על ה train set ו dev set (המקורי ומתוקן לאחר שינוי הלייבלים השגויים). להלן תוצאות ההרצה:

Data set	Overall accuracy	Car accuracy	Truck accuracy	Cat accuracy
Train-original	87%	100%	0%	0%
Train-fixed	79%	100%	0%	0%
Dev-original	88%	100%	0%	0%
Dev -fixed	80%	100%	0%	0%

כלומר, המודל המאומן חוזה עבור כל דגימה כי היא מכונית, ככל הנראה משום שהוא אומן על הדאטא הלא מאוזן אך ללא אמצעים שנועדו לשפר את הביצועים למרות זאת. כמו כן, ניתן להבחין שבשני הדאטאסטים המתוקנים הביצועים פחות טובים, ככל הנראה כיוון שרוב התיקונים היו שינוי של אובייקט שתויג כ"car" ובפועל היה אחד מהאובייקטים האחרים, ויתכן כי התיקונים שינו את ההתפלגות של הדאטאסט.
כיוון שהמודל לא עוזר לנו ללמוד את הדאטא באופן מהימן, עלינו ליצור מודל משופר שינסה להתגבר על הקשיים שנובעים מחוסר האיזון של הדאטאסטים.

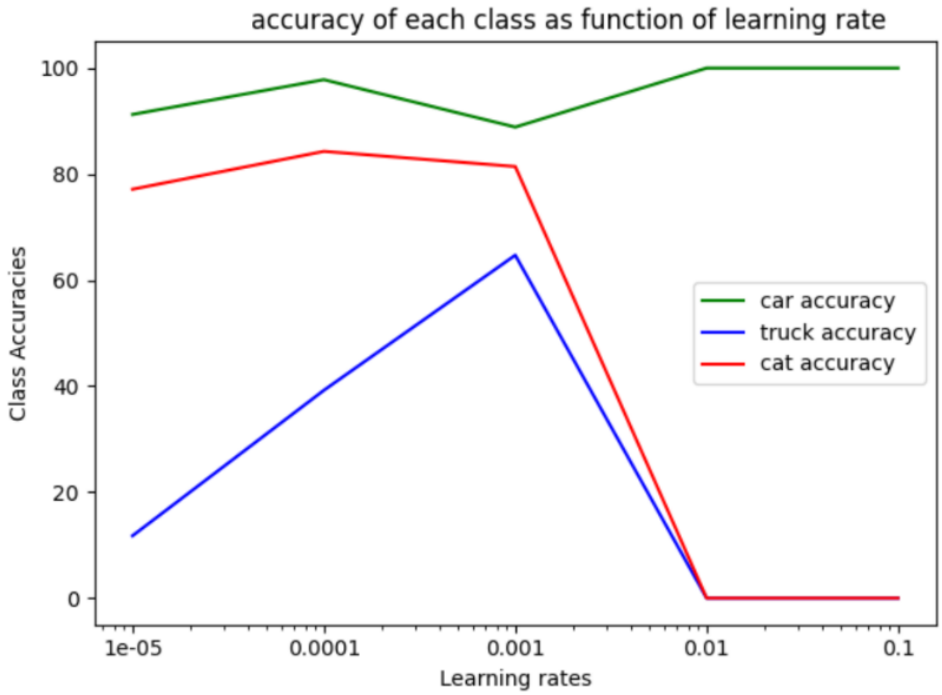
3. יצירת מודל משופר:

כדי לשפר את הביצועים ולהימנע מלחזות רק מחלקה אחת בשל חוסר האיזון בדאטא, היה צורך לאמן מודל חדש ולבחון hyperparameters שונים כדי להבין אילו נתונים עוזרים לשפר את ביצועי המודל ובאילו תנאים. נשים לב כי בשל חוסר האיזון בין שלושת המחלקות, הaccuracy הכולל מוטה לטובת המחלקה המיוצגת ביותר. אך הloss מתוקן כך שהמחלקות הפחות מיוצגות "מענישות" יותר את המודל (באמצעות weighted loss, ויוסבר על כך בהמשך). לכן נשתדל להסתכל על ה-accuracy של שלושת המחלקות בנפרד או על הממוצע שלהן ועל הloss הכולל, בשל העובדה שה-accuracy הכולל מוטה עבור הצלחה במחלקה "car".

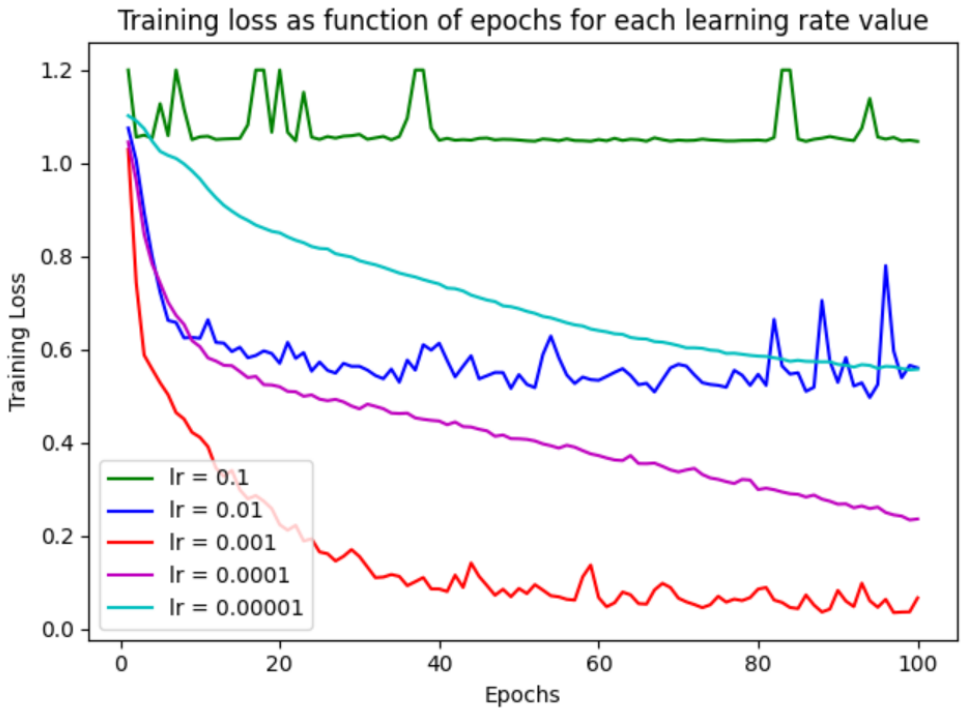
בנוסף, היה חשוב לוודא שהמודל לא עושה overfit לtrainset, וזאת ע"י השוואה של תוצאות הaccuracy בין הtrain set לtest set, ובדיקה שאין פערים גדולים מדי לטובת הtrain set.

מציאת learning rate מתאים:

הlearning rate מגדיר את גודל הצעד שהאלגוריתם יקח בכל איטרציה של האימון בכיוון ההפוך לגרדיאנט (ובכך למינימום של הפונקציה). כאשר ה learning rate גבוה מדי, כל צעד יהיה גדול מדי ולא תהיה התכנסות למינימום לוקלי לא מיטבי. כאשר הוא קטן מדי, נוכל להיעצר במינימום לוקלי ולא להתקדם לפתרון טוב יותר, או לחילופין ההתכנסות תהיה מאוד איטית ותצריך מספר איטרציות גבוה. נרצה למצוא את הערך האופטימלי עבור המודל שלנו. לשם כך ייצרתי שני גרפים – האחד מודד את הaccuracy של כל מחלקה כתלות בlearning rate, והשני את הloss כתלות במספר הepochs עבור כל ערך learning rate. החישובים נעשו עם גודל batch של 16.
בגרף הנ"ל נוכל לראות כי בערכי lr גבוהים, המודל חוזה רק car (שמיוצג הכי טוב בדאטא) ובכך טועה בשאר המחלקות ב100% מהמקרים, ככל הנראה כי המודל לא מגיע לנקודת מינימום לא מיטבית. דבר דומה קורה עבור ערכי lr קטנים מדי, ככל הנראה בשל הגעה למינימום מקומי שאי אפשר לצאת ממנו, או התכנסות איטית מדי ואי הגעה למינימום תוך מספר האיטרציות שהוגדרו.
נשים לב כי עבור ערך lr=0.001 יש ערכי accuracy מיטביים עבור כל אחת מהמחלקות, למרות שיש עדיין הבדלים שנובעים מההבדלים בייצוג של כל מחלקה בדאטא:



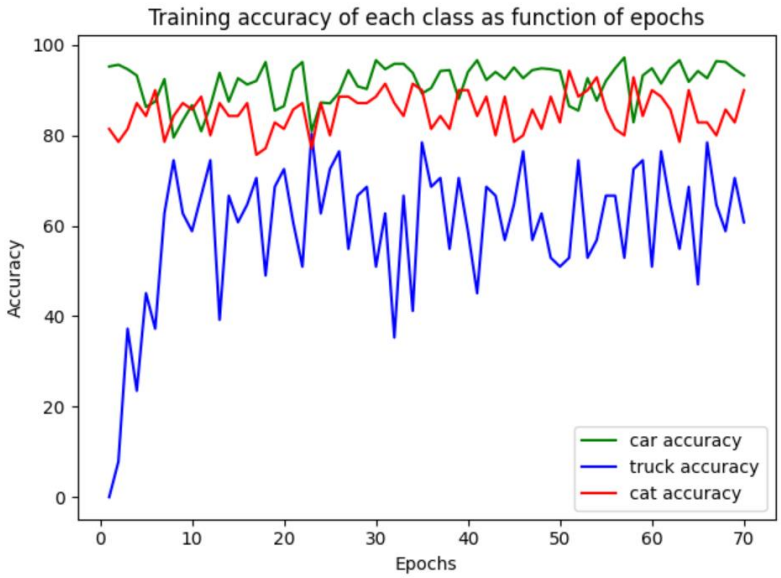
הגרף השני מציג את הtrain loss כפונקציה של מספר הepochs עבור כל ערך של learning rate. גם פה ניתן לראות כי הערך הגבוה של lr לא מצליח להתכנס, ואילו הערך 0.00001 מתכנס לאט מאוד ומצריך עוד איטרציות כדי להגיע להתכנסות. ערך ה lr המיטבי הוא 0.001, שמתכנס לערך הloss הנמוך ביותר ותוך מספר האיטרציות הקצר ביותר (כ-50). כמו כן, נבחין כי ככל שערך הlr קטן יותר כך הגרף יותר חלק ויש פחות אוסלציות, מה שנובע מכך שכל צעד הוא קטן לכן הloss לא יסטה משמעותית מהערך שבאיטרציה הקודמת:



מספר האיטרציות:

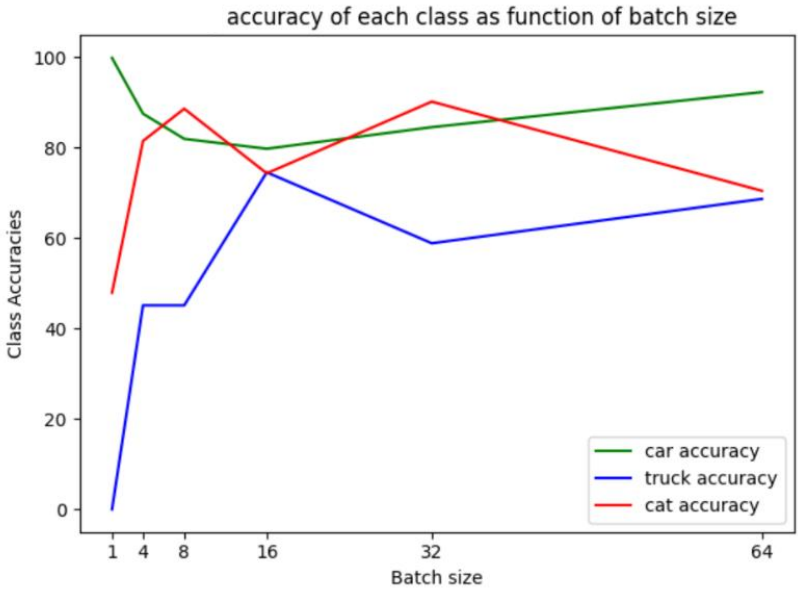
מספר האיטרציות (epochs) מגדיר כמה הפעמים בהם עוברים על סט של דאטא ומתקנים לפיו את המשקולות. כדי לבדוק מה הערך האופטימלי עבור המודל, יש לבדוק את תוצאות הloss וaccuracy כתלות בערכי epochs שונים. מהגרף הקודם ניתן לראות כי עבור ערך של lr=0.001 יש התכנסות לloss מינימלי אחרי כ-50 איטרציות. ניצור גרף נוסף שמראה את הaccuracy של כל מחלקה כתלות במספר האיטרציות. האנליזה נעשתה עם lr=0.001 ו batch size = 16.

נבחין כי עד מספר קטן של איטרציות (עד 10 בערך) המודל נוטה לחזות את המחלקות שיותר מיוצגות בדאטא (car ובמידה פחותה יותר cat) אבל החל מ10 איטרציות המודל חוזה את truck באחוזי accuracy יותר גבוהים. אולם, אין התכנסות למצב בו הaccuracy של שלושת המחלקות דומה, ובשלושתן, ובמיוחד במחלקה truck שהכי פחות מיוצגת בדאטא יש אוסילציות חזקות.



גודל batch:

ב mini batches יש השפעה על קצב התכנסות המודל ומידת הaccuracy שלו. Batch size גבוה יביא להתכנסות מהירה יותר של המודל. עם זאת, ל batches גדולים או קטנים מדי תהיה השפעה על דיוק המודל. הגרף הבא מציג את הדיוק בחישוב הaccuracy של כל מחלקה בtest set כתלות בגודל הbatch. ניתן לראות שהביצועים פחות טובים עם batches קטנים, ובערך ב16 = batch_size יש איזון מירבי של הaccuracy בכל מחלקה:



WeightedRandomSampler ו Weighted loss:

כדי שהמודל שיאומן לא יושפע מחוסר האיזון בדאטא בצורה משמעותית, ניסיתי להוסיף לו WeightedRandomSampler שיאזן את מספר הדוגמאות מכל מחלקה בכל batch. הפונקציה נותנת משקל לכל דוגמא בtrain set שמוגדר ביחס הפוך לשכיחות התיוג של הדוגמא. (למשל, "truck" מהווה 0.1 מהדוגמאות, לכן המשקל שלו יהיה יותר גבוה מ"car" שמהווה 0.7 מהדוגמאות). הבעיה היא שאותן הדוגמאות מהמחלקות שלא מיוצגות היטב בדאטא יופיעו מספר רב של פעמים בtraining set, והדבר יכול לפגוע בלמידה של המחלקות הנפוצות יותר.
בנוסף הוספתי לפונקציית הloss של הcross entropy משקולות (weighted loss) כך שהמודל יקבל loss מוגבר יותר על המחלקות שלא מיוצגות היטב בדאטאסט, ומשתפר באיטרציות הבאות באופן בו הוא חוזה אותן.
המודל הציג ביצועים טובים יותר עם השיטה של weighted loss, שהצליח יותר לאזן בaccuracy בין המחלקות השונות במרבית ההרצות (הloss גבוה יותר בweighted loss שכזכור מעניש יותר על המחלקות הפחות שכיחות בדאטא). להלן דוגמא של הרצה עם שתי השיטות: (בשתי ההרצות הנתונים האחרים הם: lr=0.001 , epochs=50, batch_size = 16, adam algorithm, weight_decay=0.01)

	Overall accuracy	Car accuracy	Truck accuracy	Cat accuracy	Avg accuracy
Weighted loss	89%	88%	64%	82%	78.61%
WeightedRandomSampler	88%	94%	43%	75%	70.67%

Augmentation:
נועד להגדיל את הדאטאסט ולאמן את המודל על ואריאציות שונות של אותו הדאטא. ניסיתי מספר סוגים של טרנספורמציות (CenterCrop, Normalize, RandomAffine, RandomGrayscale) אבל רובם פגעו בביצועי המודל, לכן נבחר השימוש בNormalize בלבד (יתכן שנובע מכך שהמרה של התמונה לפורמט אחר לטובת הטרנספורמציה גורמת לאיבוד מידע).

Regularization:
שימוש בweight decay עם פרמטר 0.01 יחד עם Adam מבצע L2 regularization על הקוד, ומשפר את ביצועי המודל (יותר מSGD עם weight decay). השיטה נועדה להקטין את סיבוכיות המודל ע"י כך שהפרמטרים של המשקולות יקטנו. זה נעשה על ידי הוספה לפונקציית הloss ביטוי שתלוי בנורמה של המשקולות.
בנוסף, כיוון שהארכיטקטורה של הרשת כוללת dropouts, כדי למנוע הורדת קודקודים בשלב הtest השתמשתי בפונקציה ()eval לפני המעבר על הtest set.
המודל הציג ביצועים טובים יותר עם השיטה של Adam, לכן השתמשתי בה. להלן דוגמא של הרצה עם שתי השיטות: (בשתי ההרצות הנתונים האחרים הם: lr=0.001 , epochs=50, batch_size = 16, weighted loss, weight_decay=0.01)

	Overall accuracy	Car accuracy	Truck accuracy	Cat accuracy	Avg accuracy
Adam	89%	88%	64%	82%	78.61%
SGD	88%	94%	43%	78%	71.67%

4. Adversarial examples
תקיפה של המודל על ידי יצירת דגימות שכאשר המודל ינסה לחזות אותן, הוא יכשל בגלל האופן שבו הן נוצרו. על המודל שלי יצרתי adversarial example מסוג Fast Gradient Sign Attack, שמתבסס על כך שהקלט ימקסם את הloss בהתאם לגרדיאנטים שחושבו בbackpropogation. יש לציין כי תקיפה זאת מצריכה שיהיו בידי התוקף הגרדיאנטים.
באופן ספציפי, כדי ליצור את הדגימה המשובשת משתמשים בדגימה המקורית, בגרדיאנט הloss ובקבוע אפסילון (ששולט בכמה רעש מוסיפים) ומשנים את התמונה בצורה הבאה:
$$perturbed\_image=image+epsilon*sign(data\_grad)=x+\epsilon*sign(\nabla_x J(\theta,x,y))$$

דוגמא:
התמונה הבאה היא תמונה שנחזתה ע"י המודל כ"car" בהצלחה. אך לאחר הוספת הרעש המתאים, התמונה נחזתה כ"cat"-