

המחלקה להנדסת תוכנה

בחירת דאטה משמעותי לאימון

חיבור זה מהווה חלק מהדרישות לקבלת
תואר ראשון בהנדסה

מאת:

בצלאל כהן

חתימה: 

מנחה אקדמי: דר' יהודה חסין	אישור:	תאריך:
רכז הפרויקטים: דר' אלי אנגלברג	אישור:	תאריך:



מערכות ניהול הפרויקט:

#	מערכת	מיקום
1	מאגר קוד	https://github.com/bezalelc/Data-pruning
2	קישור ליומן	https://docs.google.com/document/d/1flicN25UgrOSoRnzft5hCvXEyHCTuxpd2Q8Dh7Sxfi8/edit?usp=sharing
3	קישור לסרטון דוח סופי	https://drive.google.com/file/d/1IWV7570pZGKcW3rIGCeeOHUZ-S7GPCWI/view?usp=sharing

מידע נוסף:

סוג הפרויקט	מחקרי ממרצה במכללה
פרויקט מח"ר	לא
פרויקט ממשיך	זה פרויקט חדש
פרויקט זוגי	לא

תקציר

הפרויקט הוא פרויקט מחקרי עם דר' יהודה חסין בנושא של למידה עמוקה, מטרת הפרויקט היא לבדוק האם ואיך ניתן לקצץ חלק מדאטה קיים ולקבל תוצאות ברות השוואה על מודלים ללמידה עמוקה בשביל לחסוך במשאבים (זמן, כוח חישוב, איסוף דאטה) ו/או להגיע לביצועים יותר טובים של שגיאת המודל.

יש מחקרים שנעשו כבר על הנושא עם הצעות ומדדים שונים איך להעריך את איכות וקושי הדאטה, למצוא את הדאטה המשמעותי יותר לאימון ולקצץ את השאר כבר בתחילת תהליך הלמידה של המודל, במסגרת הפרויקט מימשנו אותם והצענו מדדים נוספים להערכת האיכות והחשיבות של כל דוגמת אימון בדאטה לפי מידע שיש לנו על הדאטה כבר בתחילת הלמידה וחילקנו את הדאטה לפי רמת הקושי שנתן לנו כל מדד.

בנוסף מימשנו מדד self-supervised שיועד לסווג את הדאטה לפי רמת קושי בלי לקבל labels ומידע נוסף על הדאטה וכך המדד יכול להיות שימושי גם בבעיות שבהם מנסים ללמוד דאטה בלי labels בעזרת self-supervised models.

כדי לבדוק איך כל חלק בדאטה משפיע על תהליך הלמידה של המודל ביצענו סידרת ניסויים שבדקו איך כל חלק בדאטה משפיע על שגיאת המודל ומצאנו שאם יש דאטה מספיק גדול אז המודל ילמד יותר טוב מהדוגמאות הקשות יותר ואפשר לוותר על חלק מהדאטה ולעומת זאת אם הדאטה קטן אז המודל צריך גם את הדוגמאות הקלות.

בעיה נוספת שנפוצה בדאטה היא שכדי להכין את הדאטה לאימון נותנים לאנשים שיסתכלו על הדאטה וייתנו אותה ובגלל שמדובר בכמויות גדולות של דאטה אז יש לפעמים טעויות בתיוג שיכולות לבלבל את המודל ולפגוע בנכונות שלו, בפרויקט מצאנו שבעזרת המדדים EL2N ו pred sum אפשר למצוא את ה"רעש" ולהוציא אותו מהדאטה.



הצהרה:

העבודה נעשתה בהנחיית דר' יהודה חסין,
עזריאלי המכללה האקדמית להנדסה ירושלים -
המחלקה להנדסת תוכנה,
החיבור מציג את עבודתנו האישית ומהווה חלק
מהדרישות לקבל תואר ראשון בהנדסה

תוכן עניינים

1.....	מבוא
2.....	תיאור הבעיה
3.....	סקירת עבודות דומות בספרות והשוואה
4.....	תיאור הפתרון
4.....	חלק 1: תיאור הדאטה
5.....	חלק 2: בחירת מודל לכל דאטה
5.....	Cifar10
5.....	Cifar100
6.....	Vegetable dataset
6.....	חלק 3: הגדרת מדדים להערכת קושי הלמידה של כל דוגמא בדאטה
6.....	פירוט על המדדים
9.....	מדד Unsupervised KM
13.....	תוצאות
13.....	ניסוי 1: הורדת הדוגמאות הקלות מה dataset
14.....	ניסוי 2: חיתוך של בין 0% ל 70% מהדוגמאות הקלות
15.....	ניסוי 3: אימון של sliding window בגודל 40% על הדאטה
16.....	ניסוי 4: השוואה בין חיתוך דוגמאות קלות לחיתוך קשות
17.....	ניסוי 5: מציאת רעשים בדאטה
18.....	ניסוי 6: מציאת ה k המתאים עבור מדד Unsupervised KM
19.....	מסקנות
19.....	סיווג חשיבות הדאטה בעזרת המדדים
19.....	מדד unsupervised
20.....	מציאת רעשים בדאטה
21.....	נספחים
21.....	קורלציה בין המדדים השונים
22.....	התפלגות הדאטה על המדדים
23.....	אימון מודל על חלק מהדאטה והוספת דאטה תוך כדי אימון
23.....	תיאור מאגר הקוד
24.....	ביבליוגרפיה

מילון מונחים, סימנים וקיצורים

EL2N - Ensemble L2 Norm

STD - Standard deviation

מבוא

למידת מכונה היא תחום במדעי המחשב שמאפשר לפתור בעיות שלא ניתן לפתור בעזרת תכנות "רגיל" כלומר בעזרת לולאות for ותנאים לוגיים של if-else, לדוגמא חיזוי מזג אוויר לפי נתוני שנים קודמות או סיווג אובייקטים בתמונה, בלמידת מכונה מלמדים את המחשב לעשות את מה שבא לבני אדם באופן טבעי - ללמוד מניסיון.

כדי "ללמד" את המחשב בונים מודל חישובי ומאמנים אותו על מידע קיים כדי שילמד תבניות מסוימות ואז המודל יכול לתת לנו הערכה על המידע האמיתי, לדוגמא אפשר לבנות מודל ולאמן אותו על דאטה עם תמונות שמכילות אובייקטים שונים ואז המודל יוכל לדעת איזה אובייקט יש בתמונה חדשה שהוא לא "ראה" קודם.

למידה עמוקה היא תחום בתוך למידת מכונה שבו משתמשים במודלים יותר גדולים עם יותר שכבות ופרמטרים, בלמידה עמוקה בדרך כלל במקום לתת למודל את התכונות של ה data מביאים לו את ה raw data והמודל מנסה לבד להבין אילו תכונות בדאטה יותר חשובות ולפי זה לעשות את ה prediction.

אימון של מודל גדול עם הרבה שכבות ופרמטרים צורך הרבה משאבים כמו חומרה, חשמל וזמן, בנוסף כדי לאמן מודל גדול צריך לאסוף הרבה דאטה – כלומר צריך גם אנשים שיאספו ויסווגו את הדאטה.

בעיה נוספת ב deep learning זה איכות הדאטה: אם לדוגמא יש לנו דאטה של תמונות אז צריך שמישהו יעבור על הדאטה ויסווג איזה אובייקט יש בכל תמונה ויכול להיות שבחלק מהתמונות יש טעויות בסיווג או שהאובייקט בתמונה לא ברור והתמונה יותר תבלבל את המודל ותפריע לתהליך האימון שלו.

בפריקט הזה ננסה לפתור את הבעיות האלו בעזרת חיתוך הדאטה:

- נממש מדדים שונים שיסווגו את הדאטה לפי רמת קושי
 - נבדוק איזה חלק מהדאטה יותר משמעותי ללמידה: הדאטה הקל או הדאטה הקשה?
 - נמצא את הדאטה הפחות איכותי (=רעש) ונוציא אותו מהדאטה
- כך נוכל לאמן מודל בצורה יותר יעילה על data יותר איכותי, נחסוך במשאבים ואולי אפילו נשפר את האיכות של המודל.

תיאור הבעיה

בלמידת מכונה המודלים השתפרו בשנים האחרונות בעקבות בניית מודלים עם יותר שכבות ופרמטרים שאומנו על מערכי נתונים גדולים. בהרבה סוגים של רשתות נוירונים כמו זיהוי תמונה, שפה או דיבור השגיאה של המודל יורדת ככל שהמודל יותר גדול או כל שיש יותר דאטה בסדר גודל שלם, ולכן כדי להוריד את השגיאה מ 3% ל 2% לדוגמא, צריך לאסוף פי 10 יותר נתונים או לבנות מודל הרבה יותר גדול עם יותר שכבות ופרמטרים שהאימון שלו ידרוש הרבה יותר משאבים וזמן מה שמוביל לשאלה:

- האם צריך את כל הנתונים כדי להגיע לביצועים האלו?
- האם כל הדאטה נחוץ בשביל לאמן את המודל או שאפשר להגיע לאותם תוצאות רק עם חלק מהדאטה בלי לפגוע בביצועים?
- איך אפשר למצוא את הדאטה שיותר חשוב שהמודל יתאמן עליהם ולאמן את המודל דווקא עליהם וכך לחסוך כוח חישוב וזמן ואפילו לשפר את הביצועים של המודל בלי לאסוף כמויות אדירות של דאטה?
- אולי אפשר אפילו להגיע למודל עם accuracy יותר גבוה אם ניקח רק חלק מהדוגמאות

הפתרון ה"נאיבי": אפשר עבור כל דוגמת אימון לאמן 2 מודלים עם ובלי הדוגמא ואז לראות את ההבדל ב test loss וב accuracy.
בעיות בפתרון הנאיבי:

- לא מעשי מבחינת זמן ריצה – עבור 1000 דוגמאות נצטרך להריץ 1000 מודלים
- כל מודל מאותחל רנדומלית עם פרמטרים שונים ולכן כל מודל יסתכל שונה על כל דוגמת אימון ודוגמא שהייתה חשובה למודל אחד לא בטוח שתהיה חשובה למודל אחר

ולכן צריך למצוא אלגוריתם שייתן מידע על הדאטה בתנאים הבאים:

1. האלגוריתם ייתן מידע על כל דוגמת אימון בדאטה כך שנוכל לדרג את כל הדוגמאות בדאטה לפי החשיבות וקושי הלמידה.
2. המידע יינתן בשלב מוקדם של הלמידה
3. היעילות של האימון (כולל ה preprocessing בשביל לקבל החלטה איזה דאטה לקצץ) מבחינת משאבים (זמן, כוח חישוב וכו') תשתפר בעקבות השימוש באלגוריתם
4. ה accuracy של המודל שיקבל רק את הדאטה המשמעותי יהיה טוב לפחות כמו של מודל שיקבל את כל הדאטה

סקירת עבודות דומות בספרות והשוואה

- *An empirical study of example forgetting during deep neural network learning*

[1]

המחקר מגדיר מדד לסיווג הדוגמאות אימון: אם מודל למד דוגמא מסוימת ונתן לה את המחלקה הנכונה ואז ב epoch הבא המודל נתן לה מחלקה לא נכונה אז המודל "שכח" אותה, במחקר מצאו שבעוד שיש דוגמאות שברגע שהמודל למד אותם הוא לא ישכח אותם יש דוגמאות שהמודל כל הזמן שוכח אותם, ולכן אפשר להגדיר מדד שככל שדוגמת אימון מסוימת תישכח יותר פעמים ככה היא יותר קשה, במחקר מצאו שדווקא התמונות שלא כל כך ברורות ויכולות להיחשב כיותר קשות נשכחות יותר פעמים.

- *Deep learning on a data diet: Finding important examples early in training*

[2]

המחקר מציע שיטה לחיתוך הדאטה EL2N – מדד שמימשנו בפרויקט והשווינו אותו למדדים אחרים, הרעיון הוא לקחת 10 מודלים שכל אחד מאותחל עם פרמטרים שונים ולהריץ כל אחד מהם 10 epochs ואז לבדוק את הממוצע של L2 norm על וקטור השגיאה.

במחקר הדגימו איך אפשר לאמן מודל רק על הדוגמאות עם השגיאה הכי גדולה ושלאי המדד EL2N נחשבות ליותר קשות – 50% מהדאטה ל cifar10 ו 75% מהדאטה ל cifar100 בלי לפגוע ב accuracy.

- *Beyond neural scaling laws: beating power law scaling via data pruning* [3]

במחקר הזה מציעים 2 אלגוריתמים לסיווג הדאטה לפי דוגמאות קשות וקלות:

1. Student-teacher – מריצים מודל אחד epochs 10 ומודל שני epoch אחד ואז לפי ההפרש בין ה scores של שני המודלים אפשר להעריך אם הדוגמא קשה או קלה.

2. K-means – מריצים אלגוריתם להוצאת וקטור פיזורים מהדאטה ואז על הפיזורים מריצים אלגוריתם k-means על הדאטה ולכל דוגמת אימון מעריכים את הקושי שלה לפי המרחק שלה מהמרכז.
בנוסף במחקר הזה מצאו שאם יש הרבה דאטה אפשר לקצץ את הדוגמאות הקלות, אבל אם דווקא חסר דאטה עדיף לקצץ את הדוגמאות הקשות.

תיאור הפתרון

חלק 1: תיאור הדאטה

Cifar10/100:

בפרויקט בחרנו לעבוד בעיקר עם cifar10/100 עם מודל resnet18 והרצנו מספר ניסויים כדי לייצב 2 מודלים שיתנו תוצאות יחסית טובות על הדאטה ואז נוכל להריץ עליהם ניסויים בהמשך:

Cifar10 – 60000 תמונות RGB של 10 מחלקות, 6000 תמונות לכל מחלקה ברזולוציה של 32x32
Cifar100 – 60000 תמונות RGB של 100 מחלקות, 600 תמונות לכל מחלקה ברזולוציה של 32x32

Vegetables dataset:

בנוסף ל cifar10/100 שיחזרנו חלק מהניסויים על דאטה נוסף כדי לראות את סיווג הדאטה לקשים וקלים זה תכונה של הדאטה או תכונה של המודל
Vegetables dataset – 15000 תמונות RGB של 15 מחלקות, 1400 תמונות לכל מחלקה ברזולוציה של 200x200

אוגמנטציות:

כדי להגדיל את כמות הדאטה עשינו transforms:

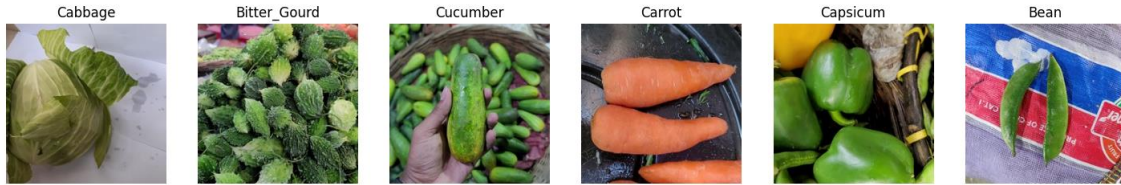
- Random crop בגודל 4
 - Random horizontal flip
 - Random rotation עם מקסימום 15 מעלות
- אפשר לראות בדוגמאות הבאות את התמונות לפני ואחרי האוגמנטציות ולראות שהוספנו דאטה בלי לפגוע בכונות של התמונות

דוגמא ל cifar100



דוגמא ל Vegetable dataset

תמונה
מקורית



אוגמטציות
אחר



חלק 2: בחירת מודל לכל דאטה

Cifar10

עבור cifar10 השתמשנו במודל (learning rate = 0.001) resnet18 + Adam.
עבור ה learning rate השתמשנו ב learning rate scheduler : תוך כדי האימון של המודל אם ה accuracy של המודל לא הישתפר על ה test-set אחרי 2 epochs ברצף חילקנו את ה learning rate בחצי.

שינויים שעשינו במודל:

1. בקונבולוציה הראשונה שינינו את ה kernel size מ 7x7 ל 3x3
2. ב fully connected בסוף המודל שינינו את גודל הפלט מ 1000 ל 10 כדי שיתאים ל cifar10 עם 10 מחלקות

Cifar100

ה accuracy של resnet18 על cifar100 הוא ~65% לכן שינינו חלק ממבנה המודל כדי להגיע ל ~75% accuracy :

השתמשנו במודל resnet18 + SGD(learning rate = 0.001) + momentum(0.9)
את ה learning rate הכפלנו ב 0.3 ב 40, 45, 50 epochs

1. בקונבולוציה הראשונה שינינו את ה kernel size ל 3x3
2. ב fully connected בסוף המודל שינינו את גודל הפלט מ 1000 ל 100
3. ביטלנו את ה residual layer לגמרי מכיוון שהוא גם לא שיפר את המודל וגם הזמן ריצה שלו לקח פי 2
4. בשכבה השנייה והשלישית שינינו את ה stride ב conv2d מ 2 ל 1

Vegetable dataset

עבור vegetable dataset השתמשנו במודל (learning rate = 0.0001) Adam + resnet18. בנוסף ב fully connected בסוף המודל שינינו את גודל הפלט מ 1000 ל 15 כדי שיתאים ל 15 מחלקות של הדאטה.

חלק 3: הגדרת מדדים להערכת קושי הלמידה של כל דוגמא בדאטה

בחלק זה נממש מדדים שונים להערכת רמת הקושי של כל דוגמא בדאטה, לכל דוגמא x_i המדד ייתן מספר d_i שייצג את רמת הקושי של הדוגמא (אפשר לראות [כאן](#) את הקורלציה בין המדדים). הרעיון הוא לתת מדדים שיתנו לנו מידע על תהליך הלמידה ויעריכו כמה קשה למודל ללמוד דוגמא מסוימת בדאטה עוד בשלב מוקדם של הלמידה וביעילות שתחסוך לנו במשאבים של כוח חישוב.

בחלק זה מימשנו את המדדים: forget – מדד שהוצע במאמר [\[1\]](#) - הרעיון של המדד זה לספור לכל דוגמא כמה פעמים המודל "שוכח" אותה תוך כדי הלמידה, EL2N – מדד שהוצע במאמר [\[2\]](#) והרעיון שלו זה לעשות ממוצע של השגיאה על ensemble של מודלים, unsupervised km – מדד שהוצע במאמר [\[3\]](#), המדד מוציא פיצ'רים מהדאטה בעזרת self-supervised model ואז בעזרת k-means מגדיר לכל דוגמא בדאטה את הקושי לפי המרחק האוקלידי מהמרכז לפי וקטור הפיצ'רים של התמונה.

בנוסף הצענו 3 מדדים נוספים: flip – לכל דוגמא נגדיר רמת קושי לפי מספר הפעמים שהמודל "היתלבט" עליה (בלי קשר לנכונות המודל), STD - בדיקת סטיית התקן של ה prediction על ensemble של מודלים, True prediction sum – הרצת ensemble של מודלים וספירת המודלים שצדקו על כל דוגמא.

פירוט על המדדים

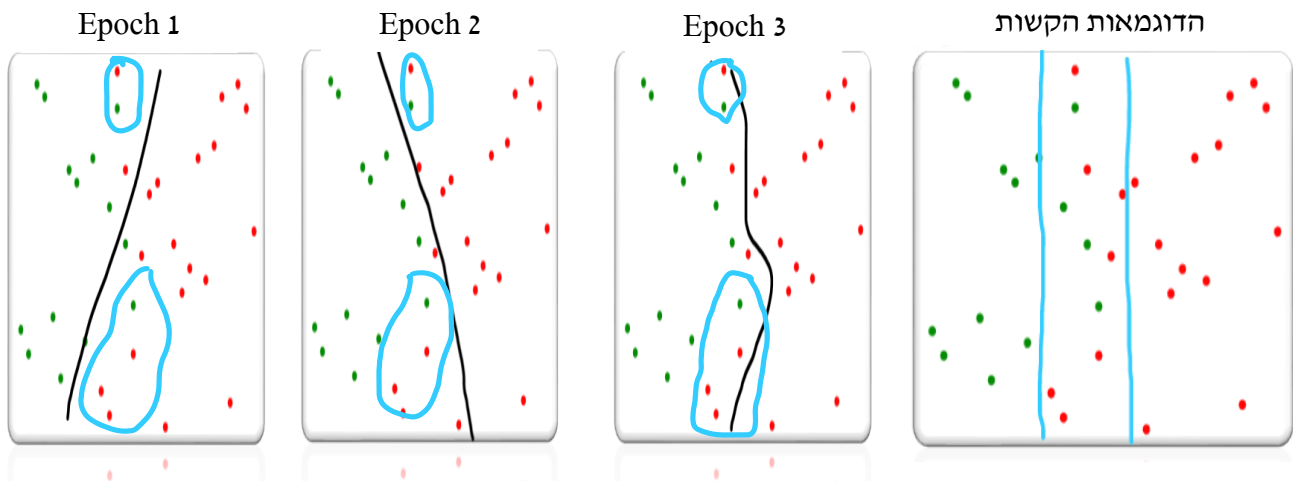
Flip

בזמן האימון של המודל יש דוגמאות של מודל ברור מה ה prediction שלהם מההתחלה לעומת דוגמאות שהמודל כל הזמן משנה את ה prediction שלהם ולכן אפשר להגדיר מדד שככל שה prediction משתנה יותר פעמים (בלי קשר לנכונות ה prediction) ככה הדוגמא יותר קשה ללימוד:

1. נאתחל מערך של counters בגודל הדאטה
2. נריץ מספר epochs ואחרי כל epoch נבדוק : אם ה prediction השתנה נעלה את ה counter של אותה דוגמה ב 1
3. הדוגמאות עם ה counter הכי גבוה הם הדוגמאות הקשות

המחשה למדד: ניקח לדוגמא מודל שאומן להבדיל בין מחלקה של ירוקים למחלקה של אדומים במשך 3 epochs – אם נסתכל על הדוגמאות שמוקפות בעיגול נראה שבכל epoch הם נמצאות בצד אחר של הקו המפריד שהמודל נותן, כלומר המודל "היתלבט" עליהם ולכן כל פעם שם אותם במחלקה שונה לעומת דוגמאות אחרות שמתחילת האימון היה ברור למודל לאיזה מחלקה הם שייכות.

במדד flip הדוגמאות שהיו ברורות למודל מתחילת האימון (בלי קשר לנכונות המודל) יחשבו לדוגמאות קלות ולעומת זאת דוגמאות שהמודל "היתלבט" וכל פעם שם אותם במחלקה שונה יחשבו לדוגמאות קשות.



Forget

המדד הוצע במאמר [1], בזמן האימון של המודל יש דוגמאות שהמודל למד מה המחלקה שלהם הוא לא "ישכח" אותם לעומת דוגמאות אחרות שאפילו אם המודל ילמד אותם הוא יכול "לשכוח" אותם ב epoch הבא ולכן נגדיר מדד כך שכל שדוגמא נשכחה יותר פעמים כך היא יותר קשה:

1. נאתחל מערך של counters בגודל הדאטה
2. נריץ מספר epochs ואחרי כל epoch נבדוק : עבור כל הדוגמאות שהמודל נתן להם prediction נכון ב epoch הקודם – אם ב epoch הנוכחי ה prediction השתנה נעלה את ה counter של אותה דוגמא ב 1
3. הדוגמאות עם ה counter הכי גבוה הם הדוגמאות הקשות

EL2N

המדד הוצע במאמר [2], סיווג קושי הלמידה של דוגמאות האימון לפי ממוצע L2 Norm על וקטור השגיאה של ה prediction של ה ensemble :

1. ניקח 10 מודלים שונים (מאותחלים רנדומלית עם פרמטרים שונים) ונאמן אותם 10 epochs
2. נעשה ממוצע על השגיאה של כל המודלים לפי - (EL2N) כלומר לכל מודל נעריך את השגיאה לפי L2 Norm ואז נעשה ממוצע על השגיאות של כל המודלים
3. לכל דוגמת אימון נעריך כמה היא קשה ללמידה לפי הציון של EL2N : אם הוא גדול אז זה אומר שההפרש בין ה prediction של ה ensemble ל true labels גדול מידי גם אחרי 10 epochs ולכן נסווג את הדוגמא כקשה לאימון

חישוב EL2N :

E = size of ensemble

M = dataset size

C = number of classes

$yOneHot[M][C]$ = true labels of dataset one hot encoded

$ensembleSoftmax[E][M][C]$ = softmax on prediction of 10 models on the data

$EL2N[M]$ = empty vector

for $m \in 0 \dots M - 1$:

for $e \in 0 \dots E - 1$:

$$L2Scores = \sqrt{\sum_{c=0}^{C-1} (yOneHot[m][c] - ensembleSoftmax[e][m][c])^2}$$

$$EL2N[m] = \frac{\sum L2Scores}{E}$$

STD

בדיקת סטיית התקן של ה prediction של ensemble של מודלים שונים לכל דוגמת אימון :

1. נאמן 10 אלגוריתמים שונים על הדאטה כמו ב EL2N
2. לכל דוגמת אימון נבדוק את השונות לכל class בין המודלים השונים - אם יש שונות גבוהה אז כנראה שמדובר בדוגמא קשה ולכן כל מודל נותן ציון אחר ואם יש שונות נמוכה אז כל המודלים חשבו אותו דבר ואז כנראה שמדובר בדוגמא קלה

חישוב STD:

$E = \text{size of ensemble}$

$M = \text{dataset size}$

$C = \text{number of classes}$

$yOneHot[M][C] = \text{true labels of dataset one hot encoded}$

$ensembleSoftmax[E][M][C]$

$= \text{softmax on prediction of 10 models on the data}$

$ensembleStd[M] = \text{empty vector}$

for $m \in 0 \dots M - 1$:

$ensembleStd[m]$

$$= \sum std(ensembleSoftmax[0 \dots E - 1, m, 0 \dots C - 1], dim = 0)$$

True prediction sum

1. נאמן 10 מודלים שונים על הדאטה במשך 10 epochs
2. נבדוק לכל דוגמא מתוך הדאטה כמה מודלים מתוך ה ensemble צדקו עליה - ככל שיותר מודלים צדקו על הדוגמא ככה היא יותר קלה ללמידה וההפך

מדד Unsupervised KM

בכל השיטות הקודמות לסיווג הדאטה כדי לסווג את הדוגמאות לקשות וקלות השתמשנו ב supervised model כמו Resnet18 ואז בשלב ההתחלתי של הלמידה הוצאנו את המדדים וסיווגנו את הדאטה. הבעיה בשיטות אלו שהם דורשות דאטה עם labels ולכן במאמר [3] הציעו מדד נוסף בעזרת unsupervised model :

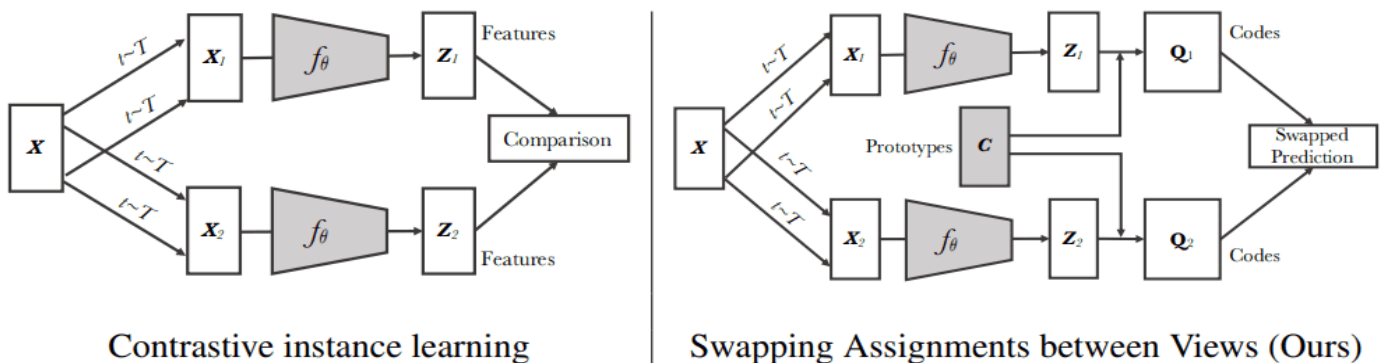
- נוציא את הפיצורים של הדאטה בעזרת unsupervised מודל
- נרץ k-means על הפיצורים ונגדיר את רמת הקושי של כל דוגמא לפי המרחק שלה למרכז הכי קרוב שלה לפי k-means

שלב 1 - הוצאת הפיצורים מהדאטה בעזרת אלגוריתם SWaV

כדי להוציא את הפיצורים מהדאטה השתמשנו במודל SWaV שמחשב וקטור של 512 פיצורים לפי אוגמנטציות שונות של אותה תמונה:

1. יצירת N מופעים שונים של אותה תמונה X_i ע"י אוגמנטציות שונות של התמונה כגון חיתוך, שינוי רזולוציה, סיבוב וכ"י
2. הוצאת הפיצורים Z_i מכל מופע של התמונה ע"י רשת נוירונים (Resnet)
3. C - מטריצה של prototypes שמחזיקה וקטורים עם התכונות העיקריות של התמונות, בעזרת אלגוריתם k-nearest-neighbors מוצאים את הווקטור המתאים לתכונות של X_i בתוך המטריצה C
4. חישוב ה"קוד" Q_i של X_i בעזרת וקטור התכונות מהשלב הקודם ובעזרת Z_i
5. swapped prediction: במקום להשוות ישירות בין שני הפיצורים של התמונות כמו ב Contrastive learning, ע"י הקוד של כל מופע X_i ננסה לעשות prediction של התכונות של מופע אחר Z_j של התמונה

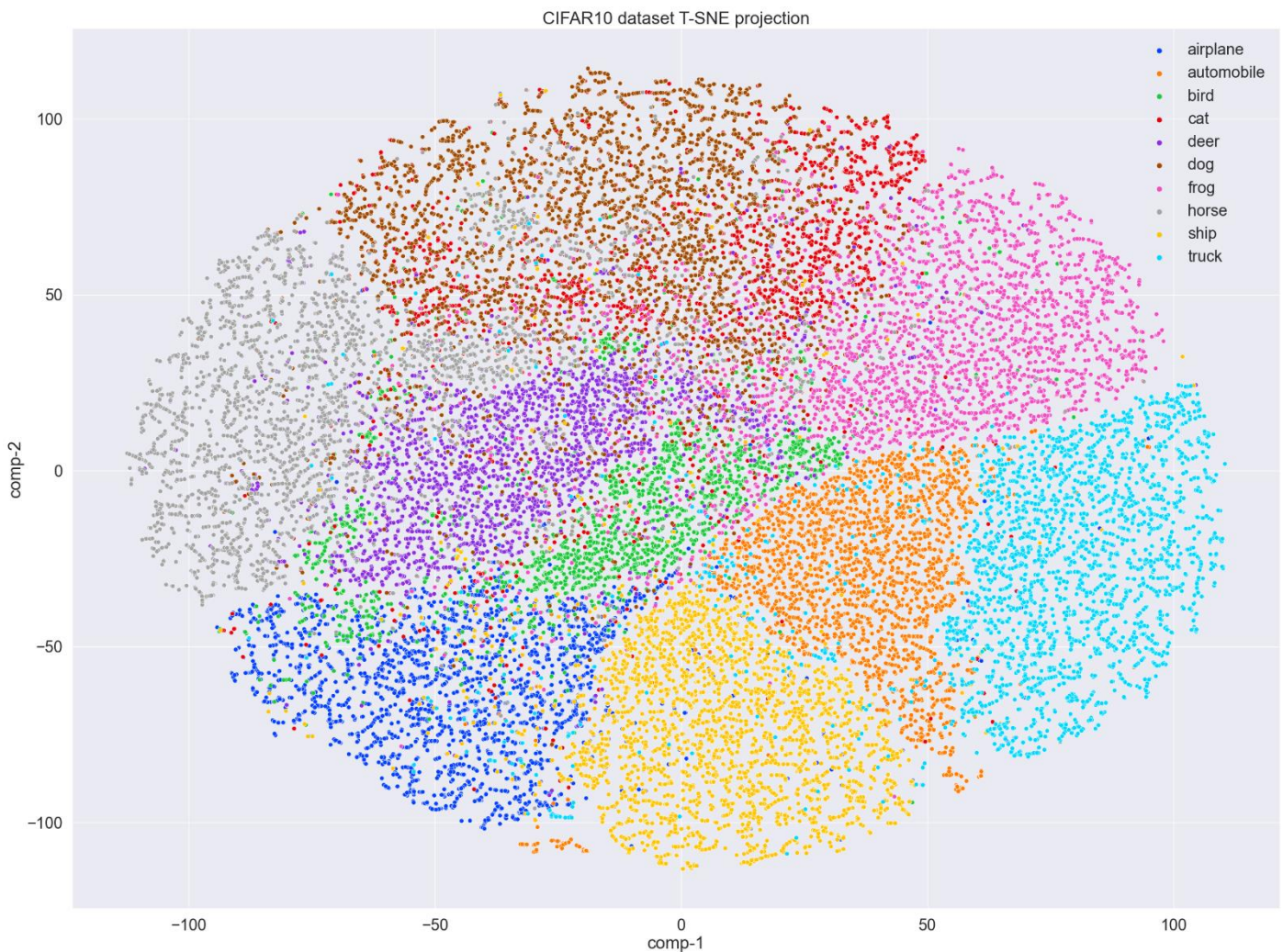
היתרון ב SWaV לעומת Contrastive learning שהשוואה בשלב האחרון היא בין prototypes ולא ישירות בין התכונות של האוגמנטציות ואז יותר קל למצוא את התכונות הכלליות של כל תמונה.



השוואה בין SWaV ל Contrastive learning מתוך [4]

בדיקת הפיצ'רים ש SWaV מוציא:

- הרצנו SWaV על cifar10 והוצאנו וקטור של 512 פיצ'רים לכל דוגמת אימון
- כדי שיהיה ניתן לראות ויזואלית את ההפרדה בין המחלקות עשינו הורדת ממד מ 512 ל 2 ממדים בעזרת אלגוריתם t-SNE (אלגוריתם להורדה מממד גבוה לממד נמוך), כדי לראות אם הפיצ'רים שהוצאנו מתארים טוב את ההפרדה בין המחלקות.
- בגרף הבא שמנו כל דוגמת אימון על הגרף לפי הערכים שקיבלנו מ t-SNE וצבענו כל מחלקה בצבע אחר:



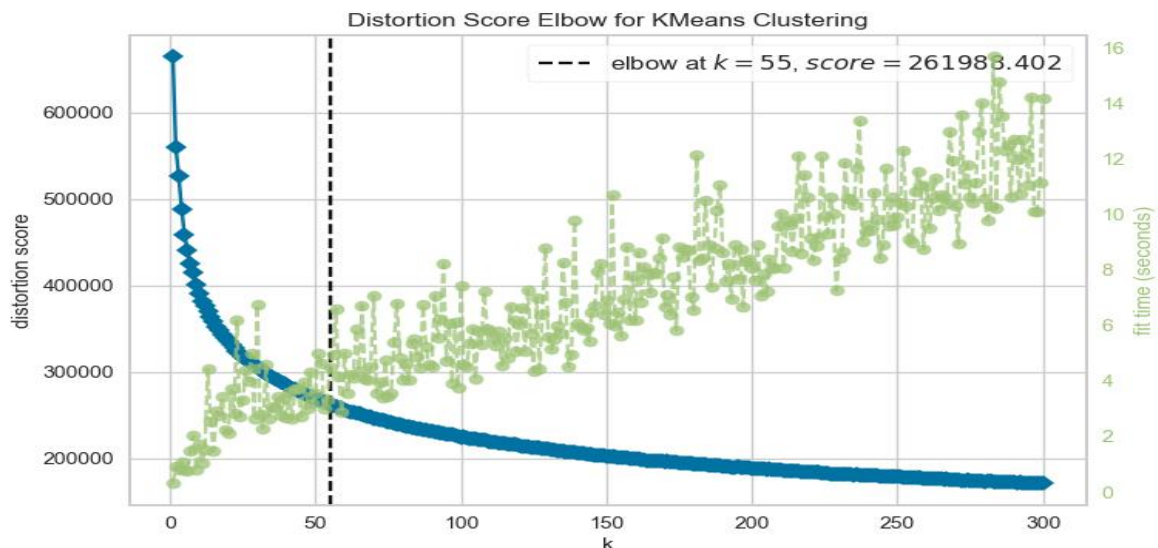
אפשר לראות בגרף שלכל מחלקה רוב הדוגמאות שלה מרוכזות באזור מסוים בגרף ולכן ניתן להסיק שהפיצ'רים ש SWaV מוציא מתארים בצורה טובה את הדאטה ונותנים הפרדה טובה בין המחלקות השונות.

שלב 2 – הרצת k-means על הפיצ'רים

אחרי שהוצאנו את הפיצ'רים מהדאטה בעזרת SWaV וראינו שהם מתארים טוב את ההפרדה בין המחלקות השונות נישאר להריץ k-means על הפיצ'רים, הבעיה ב k-means זה למצוא את ה k (מספר המרכזים) הספציפי שמתאר הכי טוב את הדאטה.

כדי למצוא את ה k המתאים לאלגוריתם k-means השתמשנו ב elbow method : מריצים אלגוריתם kmeans עם k שונים ובודקים עבור איזה k הירידה ב loss של k-means הופכת לליניארית, בהמשך נפרט על עוד אפשרויות למצוא את ה k ונשווה ל k שמצאנו בעזרת elbow method

השוואת ה loss של k-means כתלות ב k



שלב 3 – סיווג רמת הקושי של הדוגמאות

בשלב הזה לאחר שהוצאנו פיצ'רים מהדאטה והרצנו k-means נגדיר לכל דוגמת אימון את רמת הקושי שלה לפי המרחק האוקלידי שלה מהמרכז הכי קרוב אליה ב k-means – דוגמאות שיותר רחוקות מהמרכז יחשבו לדוגמאות יותר קשות וההפך.
למדד זה יש יתרון על המדדים האחרים:

- כי אם יש לנו מדד מאומן שיוגע להוציא את הפיצ'רים מהדאטה אפשר להשתמש בו לסווג את הדאטה בלי הרבה כוח חישוב
- אפשר להשתמש בו גם בבעיות אחרות שבהם יש לנו דאטה בלי labels ואנחנו מנסים ללמוד בעזרת self-supervised model

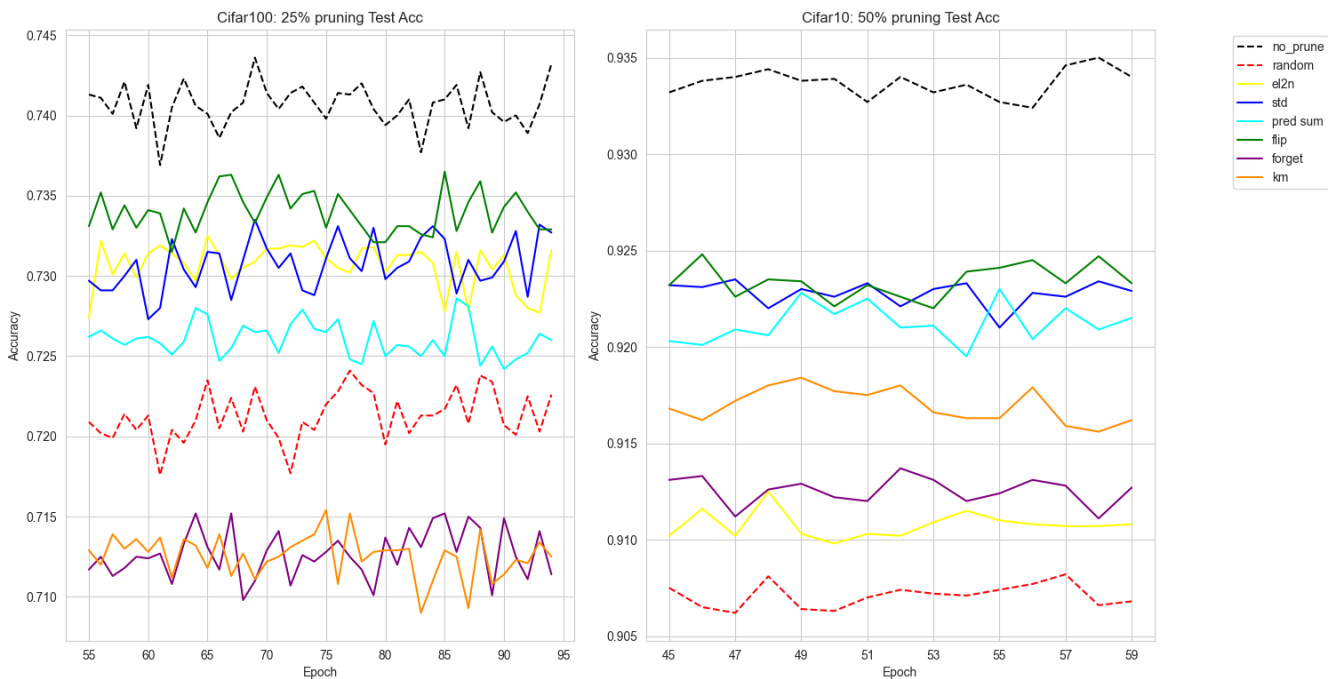
תוצאות

בחלק זה ננסה להשוות בין המדדים השונים על ידי סידרת ניסויים, בכל הניסויים עשיתי השוואה למודל שאומן על כל הדאטה (הקו המקווקו השחור) ולמודל שבו חתכנו מהדאטה רנדומלית (הקו המקווקו האדום).

ניסוי 1: הורדת הדוגמאות הקלות מה dataset

ניסוי על cifar10/100: בניסוי הזה הורדנו 25% מהדוגמאות הקלות מ cifar100 ו 50% מ cifar10 ובדקנו כמה זה משפיע על המודל, ציר ה x תואם למספר ה epoch – החל מה epoch שבו המודל התייצב וציר ה y תואם ל accuracy.

אפשר לראות שאפשר להגיע כמעט לאותו תוצאה אפילו עם חיתוך של 25% מהדוגמאות הקלות ביותר ב cifar100 או 50% ב cifar10, ודווקא אם חותכים בצורה רנדומלית מגיעים לתוצאות פחות טובות.

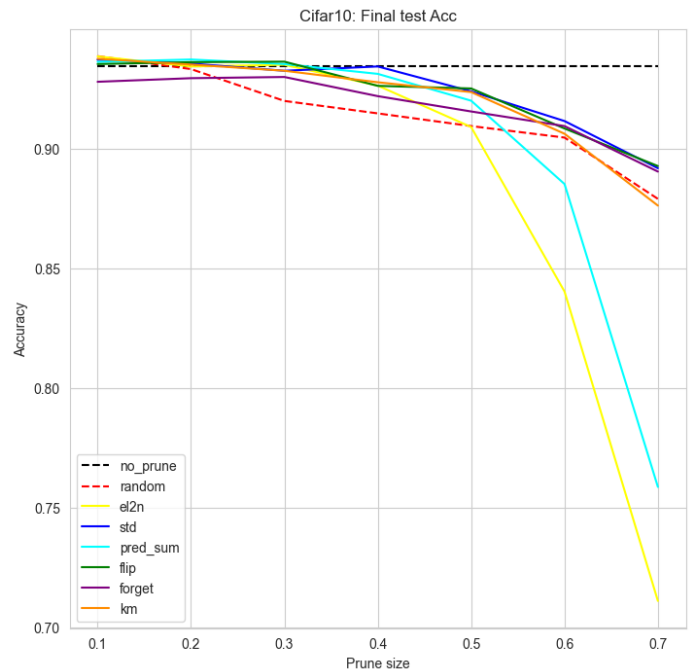
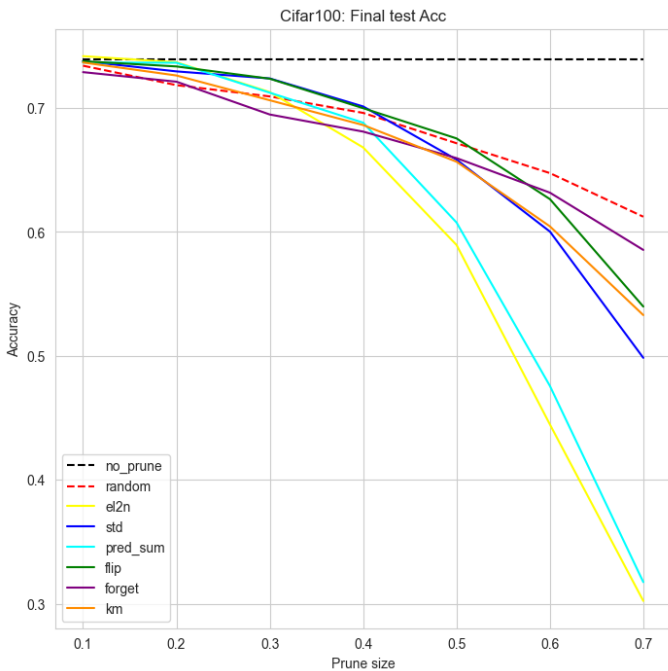


ניסוי 2: חיתוך של בין 0% ל 70% מהדוגמאות הקלות

ניסוי על cifar10/100: בניסוי הזה בדקנו איך חיתוך הדאטה לפי המדדים השונים משפיע על המודל ככל שמורידים יותר דוגמאות קלות.
ציר ה x תואם לגודל החיתוך של הדוגמאות הקלות לדוגמא אם $x = 0.1$ אז המודל אומן על 90% מהדוגמאות הקשות אחרי שקיצצנו את ה 10% הכי קלות וציר ה y תואם לתוצאה הסופית של המודל על ה test-set.

תוצאות מהניסוי:

- אפשר לראות שאפילו שהשתמשנו ב cifar100 שבו יש 50000 דוגמאות אימון ו 100 מחלקות כלומר 500 תמונות למחלקה ואין בו מספיק דאטה - בכל זאת אפשר לחתוך 20% מהדאטה בלי לפגוע ב accuracy ואפילו אם נוריד אחוז יותר גבוה מהקלים (עד 35%) עדיין נקבל תוצאות יותר טובות מאשר חיתוך רנדומלי.
- בנוסף אפשר לראות שמתי שחתכנו יותר מידי מהדוגמאות הקלות המודל למד יותר גרוע מאשר חיתוך רנדומלי.

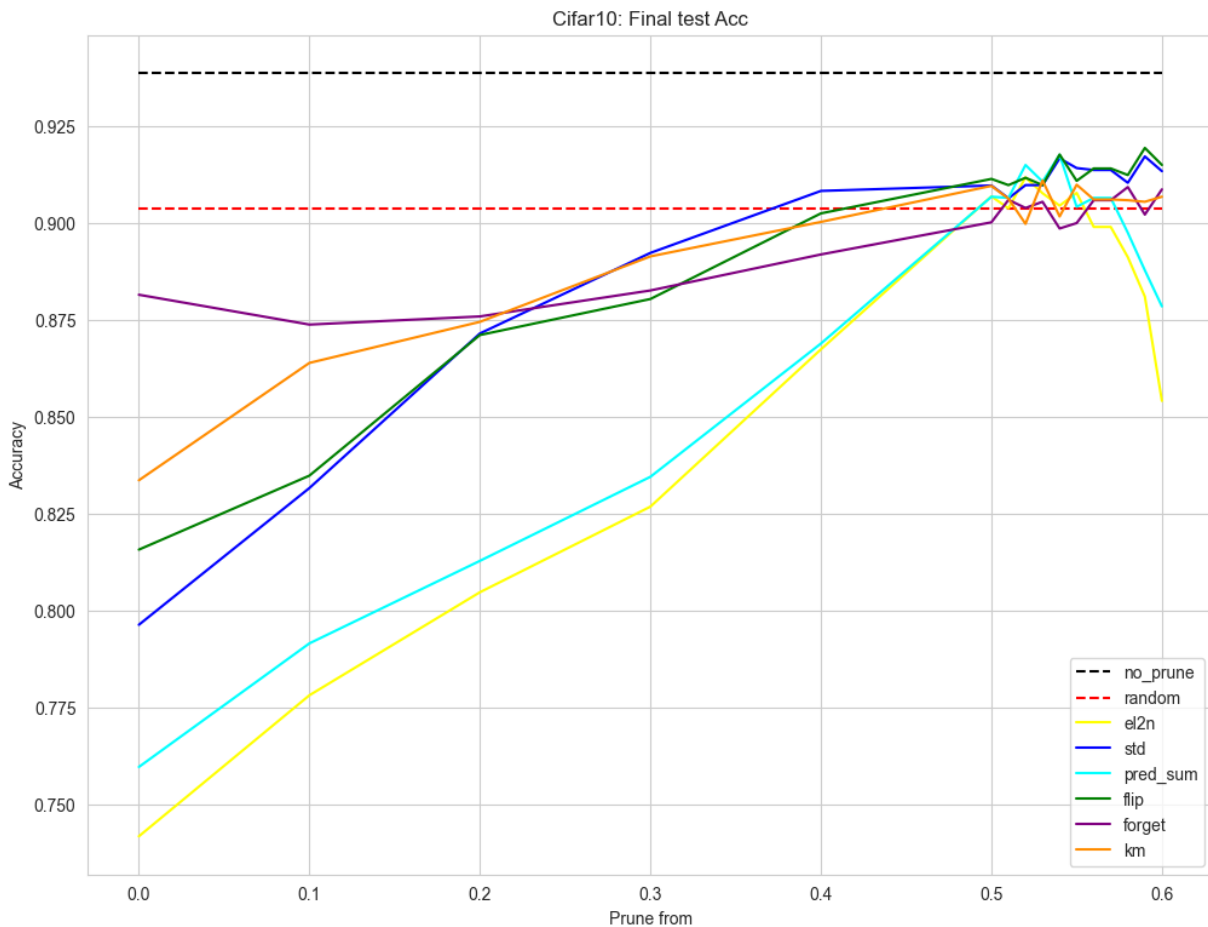


האינטגרל על השטח מתחת לגרף מנורמל לפי no pruning:

	random	EL2N	STD	Pred sum	flip	forget	km
Cifar10	0.98	0.96	0.991	0.976	0.989	0.984	0.987
Cifar100	0.928	0.83	0.911	0.849	0.925	0.913	0.906

ניסוי 3: אימון של sliding window בגודל 40% על הדאטה

ניסוי על cifar10: תוצאה סופית על ה test-set אחרי אימון של sliding window בגודל 40% מהדאטה, ציר ה x תואם לנקודה ממנה התחלנו לקחת את ה 40% : מה 40% הכי קלים בצד שמאל עד ל 40% הכי קשים בצד ימין לפי המדדים השונים וציר ה y תואם לתוצאה הסופית של המודל על ה test-set.
לדוגמא אם בציר ה x הערך הוא 0.3 אז לקחנו את הדאטה בטווח של 30%-70% מהדוגמאות לפי רמת הקושי של כל מדד.



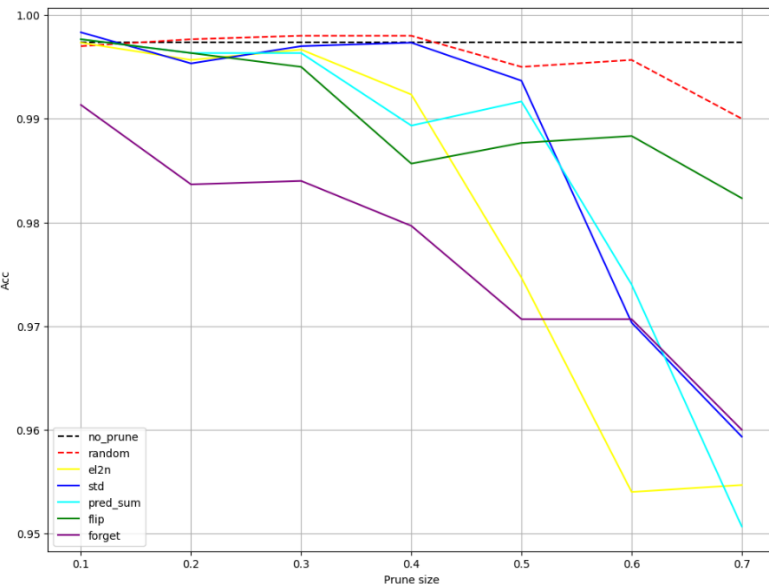
תוצאות מהניסוי:

1. בניסוי הזה אפשר לראות שככל שה sliding window מתקרב לדוגמאות הקשות ככה המודל לומד יותר טוב
2. אם לוקחים דווקא את הדוגמאות הקשות (אבל לא הכי קשות) אז המודל לומד הרבה יותר טוב מאשר random pruning
3. בנוסף אפשר לראות במדדים EL2N ו pred sum שמתי שלוקחים דווקא את הדוגמאות הכי קשות אז המודל לומד קצת פחות טוב.

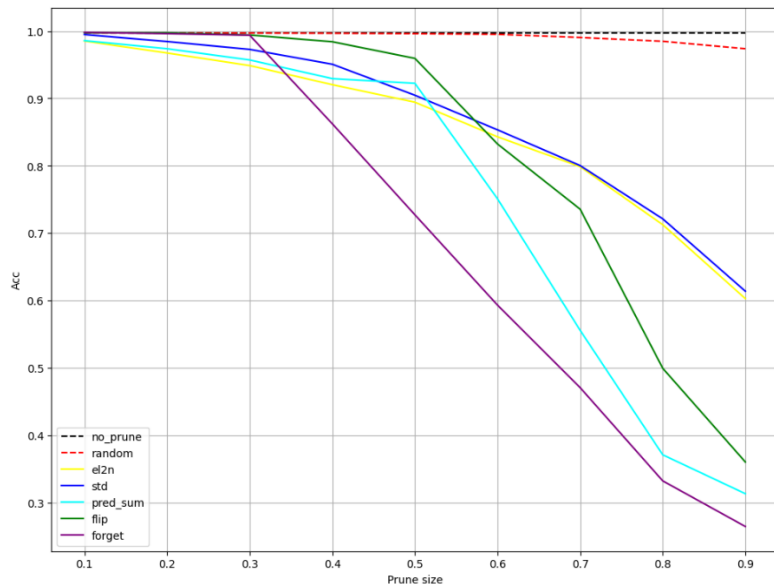
ניסוי 4: השוואה בין חיתוך דוגמאות קלות לחיתוך קשות

ניסוי על Vegetables dataset: בניסוי הזה בדקנו איזה חלק מהדאטה יותר משפיע על הלמידה של המודל - הדאטה הקשה או הקל.
בגרף הימני הורדנו מהדוגמאות הקשות בין 10% - 90%, לעומת הגרף השמאלי שבו הורדנו דווקא מהדוגמאות הקלות.
ציר ה-x תואם לגודל החיתוך של הדוגמאות הקלות/קשות, וציר ה-y תואם לתוצאה הסופית של המודל על ה-test-set, לדוגמא בגרף הימני אם $x = 0.1$ אז הורדנו מהדוגמאות הקשות. ובגרף השמאלי אם $x = 0.4$ אז הורדנו 40% מהדוגמאות הכי קלות.

Prune 10%-70% from easiest examples



Prune 10%-90% from hardest examples



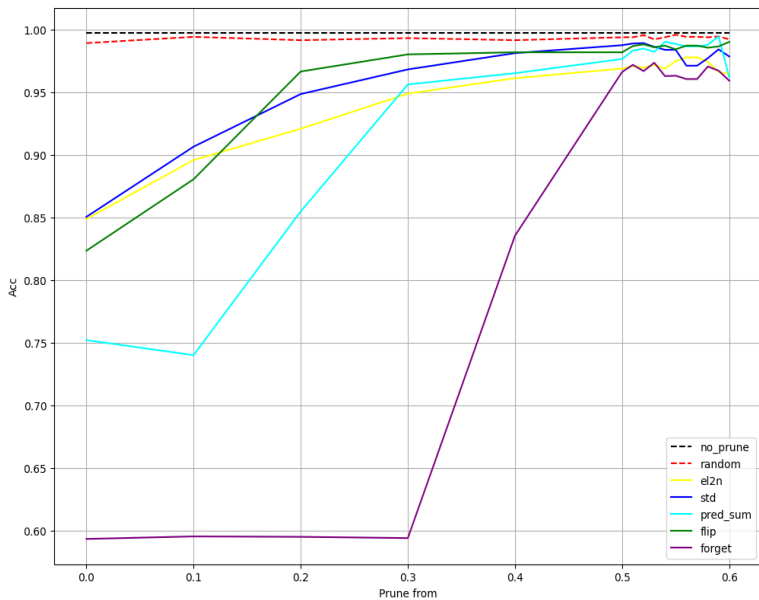
תוצאות מהניסוי:

- גם אם חתכנו חלק גדול מהדאטה הקל ה accuracy של המודל ירד רק באחוזים בודדים, לעומת זאת אם הורדנו חלק גדול מהדאטה הקשה דווקא, הביצועים של המודל ירדו ב- 30% - 50%.
- אפשר לחתוך את הדאטה הכי קל - 10% ולקבל תוצאות יותר טובות ממודל שאומן על כל הדאטה.
- בניסוי הזה חיתוך רנדומלי נותן תוצאות יותר טובות מאשר אימון רק על הקשים לעומת הניסויים הקודמים על cifar10/100.

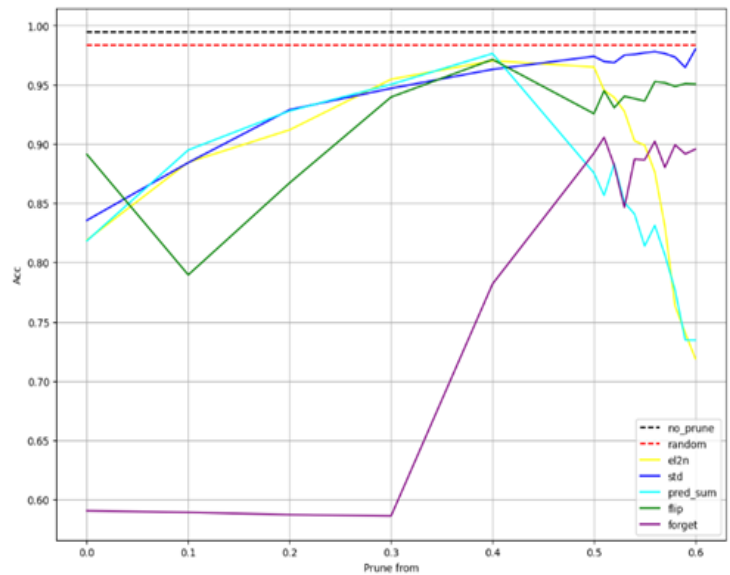
ניסוי 5: מציאת רעשים בדאטה

ניסוי על Vegetables dataset: תוצאה סופית על ה test-set אחרי אימון של sliding window בגודל 40% מהדאטה, ציר ה x תואם לנקודה ממנה התחלנו לקחת את ה 40% : מה 40% הכי קלים בצד שמאל עד ל 40% הכי קשים בצד ימין לפי המדדים השונים וציר ה y תואם לתוצאה הסופית של המודל על ה test-set. לדוגמא אם בציר ה x הערך הוא 0.3 אז לקחנו את הדאטה בטווח של 30%-70% מהדוגמאות לפי רמת הקושי של כל מדד. בצד שמאל הרצנו את הניסוי על הדאטה הרגיל ובצד ימין שינינו את המחלקות של 10% מהדאטה למחלקות רנדומליות אחרות כדי ליצור רעשים בדאטה.

0% mixed classes



10% mixed classes

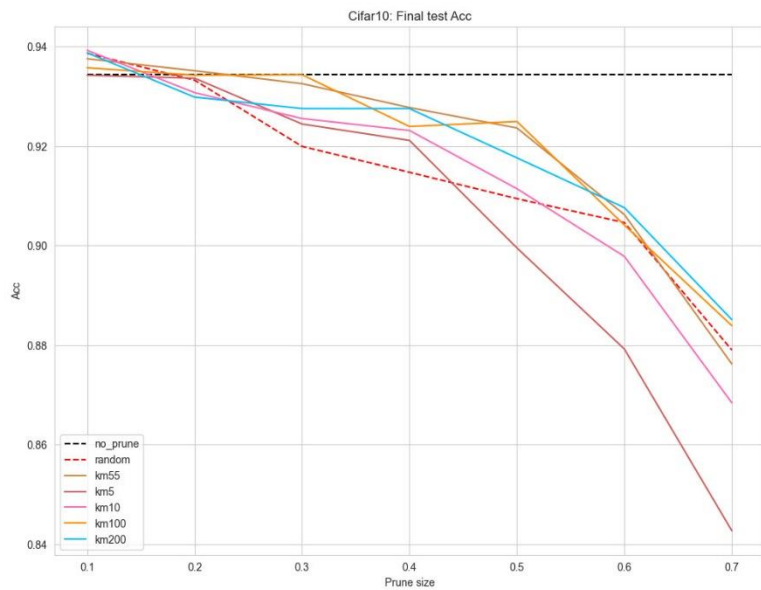
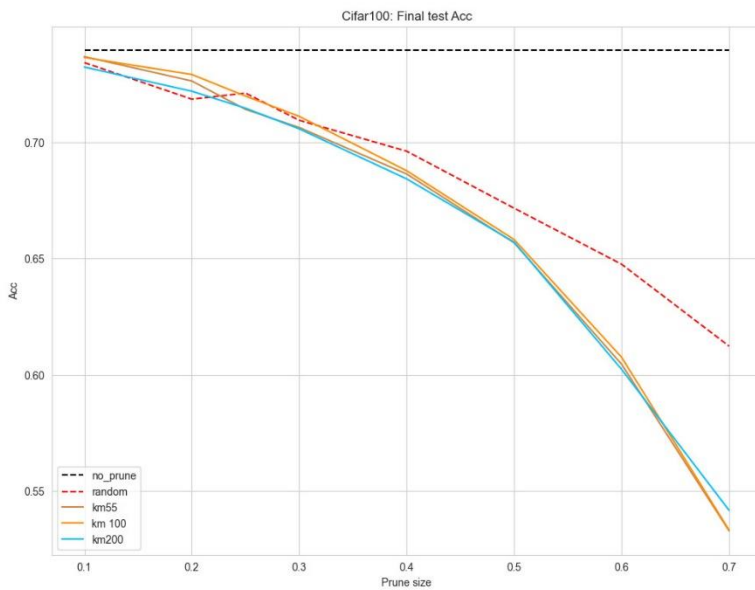


תוצאות מהניסוי:

- אפשר לראות שככל שהמודל מתאמן על דאטה יותר קשה ככה הדיוק שלו השתפר לפי כל המדדים - הוכחה חד משמעית שהדאטה הקשה יותר חשוב ללמידה של המודל
- מדד forget נותן תוצאות די דומות עבור הדוגמאות הקלות ורק שמתקרבים עם ה sliding window לדוגמאות הקשות הוא מתחיל להשתפר
- בצד ימין אפשר לראות שמתי שהרסנו 10% מהדאטה אז לפי המדדים el2n ו pred sum הדיוק של המודל ירד כשהתקרבו עם ה sliding window לדוגמאות הקשות ביותר כלומר המדדים האלה הצליחו לזהות את הרעש בדאטה
- יש הבדל בין el2n ו pred sum: ה el2n הצליח יותר טוב לזהות את הרעשים והדיוק שלו התחיל לרדת רק שהתקרבו ל 10% הכי קשים לעומת ה pred sum שהדיוק שלו התחיל לרדת עוד באזור ה 20% הקשים

ניסוי 6: מציאת ה k המתאים עבור מדד Unsupervised KM

בתיאור הפתרון הראיתי איך אפשר למצוא את ה k המתאים בעזרת elbow method, בניסוי הבא ננסה לבחון אם אפשר למצוא k יותר טוב אם נבחר k שמתאים למספר המחלקות. בניסוי הבא חזרתי על ניסוי מספר 2 שבו חתכנו בין 10% ל 70% מהדוגמאות הקלות רק שהפעם עשינו את החיתוך לפי מדד unsupervised km עם ערכי k שונים: k שמתאים למספר המחלקות, k שמצאנו בעזרת elbow method ($k=55$) ועוד כמה k נוספים בטווח שלהם.



תוצאות מהניסוי:

- אפשר לראות שעל cifar100 עבור $k=100$ קיבלנו תוצאות קצת יותר טובות מאשר שאר ה k שבחרנו
- ב cifar10 דווקא מתי ש $k=10$ שזה מספר המחלקות התוצאות פחות טובות

מסקנות

סיווג חשיבות הדאטה בעזרת המדדים

בניסויים הוכחנו שאפשר עוד בתחילת האימון ובמדד $unsupervised\ km$ אפילו עוד לפני האימון עצמו ובמעט זמן וכוח חישוב לסווג את הדאטה ולדעת איזה דוגמאות יהיו יותר משמעותיות ללמידת ודיוק המודל ואיזה דוגמאות פחות ילמדו את המודל. במהלך הניסויים עלו מספר מסקנות מעניינות:

1. בניסוי 3 [c] אפשר לראות בבירור שכל שבועות לאמן את המודל על הדאטה הקשה יותר ככה המודל מגיעה לדיוק יותר גבוה, כלומר הדאטה הקשה יותר משמעותי בתהליך הלמידה של המודל מאשר הדאטה הקל
2. אפשר לוותר על חלק מהדאטה הקל בלי לפגוע כמעט בדיוק של המודל ואפילו לשפר קצת את הדיוק כמו בניסוי 2 [b] על $cifar10$ אבל אם נוותר על הדאטה הקשה הדיוק של המודל ייפגע בעשרות אחוזים כמו בניסוי 4 [d]
3. מתי שחתכנו יותר מידי מהדוגמאות הקלות המודל למד יותר גרוע מאשר חיתוך רנדומלי, במאמר [2] הסבירו שכנראה מתי שהמודל רואה רק את הדוגמאות הקשות אז הוא מתאים את עצמו דווקא לקבוצה קטנה של דוגמאות קשות ולא למגוון של דוגמאות ולכן למרות שהדוגמאות הקשות יותר חשובות למודל עדיין צריך קצת מהקלות כדי שהמודל יישאר מאוזן
4. מתי שמספר דוגמאות האימון יחסית קטן קשה לוותר על הדוגמאות הקלות כמו ב $Vegetables\ dataset$, לעומת זאת אם יש הרבה דוגמאות אימון כמו ב $cifar10/100$ אפשר לוותר על חלק גדול מהדאטה הקל כמו בניסוי 2 [b] אבל אי אפשר לוותר על הדאטה הקשה
5. יש קורלציה גבוהה בין המדדים [g] ולכן דוגמא קשה לפי מדד אחד תהיה בסבירות גבוהה קשה גם לפי מדדים אחרים
6. אפשר לזהות את הדוגמאות הקשות והקלות עוד לפני שמתחילים את תהליך הלמידה: בניסויים 4 ו 5 [d] [e] קודם אימנו מודל פשוט עם שכבת קונבולוציה יחידה, חילצנו את המדדים ורק אז הרצנו את הניסויים על המודל ה "כבד" $resnet18$

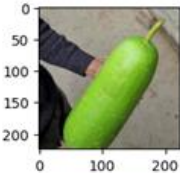

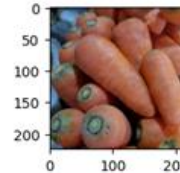
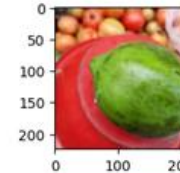
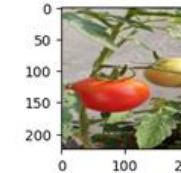
מדד $unsupervised$

1. אפשר לסווג את הדאטה גם בלי labels בעזרת $unsupervised$ מודל ולקבל תוצאות דומות בסיווג הדאטה, מדד זה יכול להיות שימושי בבעיות שהם לא בעיית סיווג ואין להם labels, בנוסף אם יש לנו מודל שכבר מאומן להוציא פיצ'רים מהדאטה או שיש לנו את הפיצ'רים עצמם אפשר בקצת כוח חישוב וזמן לסווג את הדאטה
2. במדד $unsupervised\ km$ אפשר לבחור k בעזרת $elbow\ method$ ולקבל קירוב טוב למספר המרכזים האופטימלי k בשביל k -means, אפשר להשתמש גם ב k שווה למספר המחלקות אם יודעים כמה מחלקות יש בדאטה אבל אם מספר המחלקות קטן יחסית נקבל ביצועים גרועים של המדד אם נבחר k =מספר המחלקות

מציאת רעשים בדאטה

בניסוי 5 [e] רואים שאפשר לזהות רעשים בדאטה בעזרת המדדים el2n ו pred sum כי מתי שערבנו 10% מהמחלקות וה sliding window הגיע ל 10%-15% של הדאטה הקשה הדיוק התחיל לרדת לעומת הניסוי שלא ערבנו בו את המחלקות שבו הדיוק עלה ככל שלקחנו יותר קשים.

אפשר לראות גם שבאופן כללי ה el2n הצליח יותר טוב לזהות את הרעשים והדיוק שלו התחיל לרדת רק שהתקרבו ל 10% הכי קשים לעומת ה pred sum שהדיוק שלו התחיל לרדת עוד מה 20% הכי קשים, הסיבה כנראה בגלל ששני המדדים דומים - שניהם עושים ממוצע על ה ensemble אבל ה el2n הוא מדד "רך" ויותר מדויק לעומת ה pred sum שהוא מדד "קשה" שעובד עם counter.

<p>el2n score: 1.329 True class: Pumpkin Bottle_Gourd: 92% Papaya: 6% Brinjal: 1% Cucumber: 1% Bean: 0% Capsicum: 0% Carrot: 0% Pumpkin: 0% Potato: 0% Bitter_Gourd: 0%</p> 	<p>el2n score: 1.361 True class: Radish Papaya: 94% Cucumber: 2% Brinjal: 1% Bottle_Gourd: 1% Bean: 1% Capsicum: 1% Cabbage: 0% Bitter_Gourd: 0% Broccoli: 0% Carrot: 0%</p> 	<p>el2n score: 1.317 True class: Papaya Carrot: 83% Potato: 9% Radish: 2% Tomato: 2% Pumpkin: 1% Capsicum: 1% Cauliflower: 1% Bean: 0% Papaya: 0% Bitter_Gourd: 0%</p> 	<p>el2n score: 1.354 True class: Cucumber Papaya: 86% Bitter_Gourd: 7% Capsicum: 3% Brinjal: 3% Bottle_Gourd: 1% Cucumber: 1% Bean: 0% Carrot: 0% Pumpkin: 0% Radish: 0%</p> 	<p>el2n score: 1.314 True class: Potato Tomato: 92% Cauliflower: 3% Potato: 1% Radish: 1% Capsicum: 1% Carrot: 1% Pumpkin: 1% Cabbage: 0% Bitter_Gourd: 0% Bean: 0%</p> 
--	--	--	---	---

דוגמא לדוגמאות שמדד el2n מסווג אותם כקשות בניסוי השני מתי שהרסנו 10% מהמחלקות, ניתן לראות שבדוגמאות אלו המחלקה ה "נכונה" (True class) היא לא המחלקה האמיתית

נספחים

קורלציה בין המדדים השונים

<u>Cifar100</u>	EL2N	STD	True pred sum	Flip	Forget	km
EL2N	1	0.763640	0.931154	0.631182	0.015292	0.299591
STD	0.763640	1	0.653038	0.667967	0.121611	0.384026
True pred sum	0.931154	0.653038	1	0.544786	0.114446	0.250940
Flip	0.631182	0.667967	0.544786	1	0.394457	0.306184
Forget	0.015292	0.121611	0.114446	0.394457	1	0.006370
km	0.299591	0.384026	0.250940	0.306184	0.006370	1

<u>Cifar10</u>	EL2N	STD	True pred sum	Flip	Forget	km
EL2N	1	0.780107	0.944299	0.658627	0.399751	0.529046
STD	0.780107	1	0.660269	0.586532	0.377763	0.496631
True pred sum	0.944299	0.660269	1	0.627331	0.350162	0.460980
Flip	0.658627	0.586532	0.627331	1	0.770378	0.394874
Forget	0.399751	0.377763	0.350162	0.770378	1	0.253914
km	0.529046	0.496631	0.460980	0.394874	0.253914	1

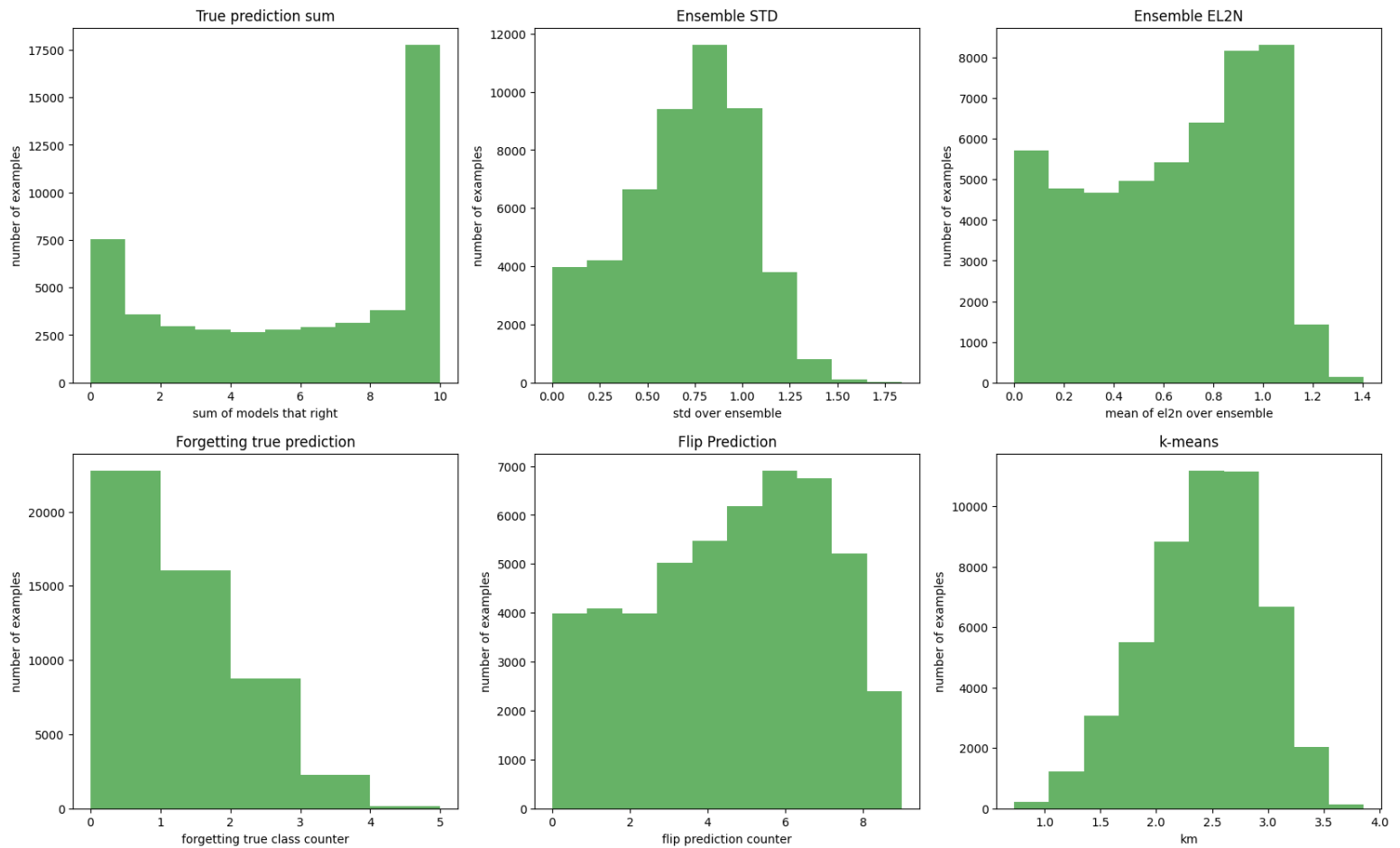
מהקורלציה אפשר לראות :

- יש קורלציה גבוהה בין המדדים חוץ מה forget ולכן דוגמת אימון שמדד אחד יסווג אותה כקשה תהיה בסבירות גבוהה קשה גם לפי המדדים האחרים, כלומר אפשר להבין שקושי של מודל ללמוד דוגמא מסוימת זה תכונה של הדאטה ולא ספציפי למודל או מדד מסוים.
- הסיבה שלמדד forget יש קורלציה נמוכה היא בגלל שברוב דוגמאות האימון אחרי שמודל למד את הדוגמא הוא לא שוכח אותה ולכן גם אם הדוגמא לא הכי קלה ה forget שלה יהיה 0
- ל EL2N ול pred sum יש קורלציה יחסית גבוהה ובעצם אפשר להגדיר את EL2N כמדד רך של pred sum בגלל ש EL2N במקום לבדוק כמה מודלים צדקו על דוגמא מסוימת (מספר שלם) הוא בודק איזה אחוז כל מודל נותן לכל דוגמא עבור כל המחלקות
- למרות שיש קורלציה גבוהה בין רוב המדדים אפשר לראות שיש ביניהם הבדלים ויכול להיות שמדד מסוים ייתן תוצאות יותר טובות מאחרים בסיווג דוגמאות או במציאת רעשים בדאטה כמו שרואים בחלק של הניסויים

התפלגות הדאטה על המדדים

אפשר לראות מההתפלגות שמדד forget מזהה דווקא את הדוגמאות הקשות יותר ולכן את רוב הדאטה הוא מזהה כקל ורק חלק קטן מהדוגמאות הוא מזהה כקשים לעומת שאר המדדים שבהם הדאטה מתחלק יותר יחסית בין הרמות השונות של המדד.

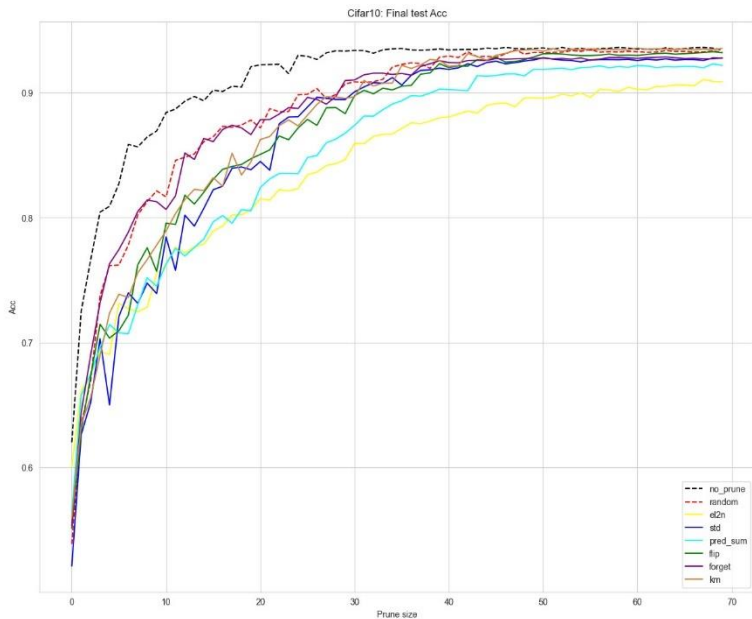
Cifar100



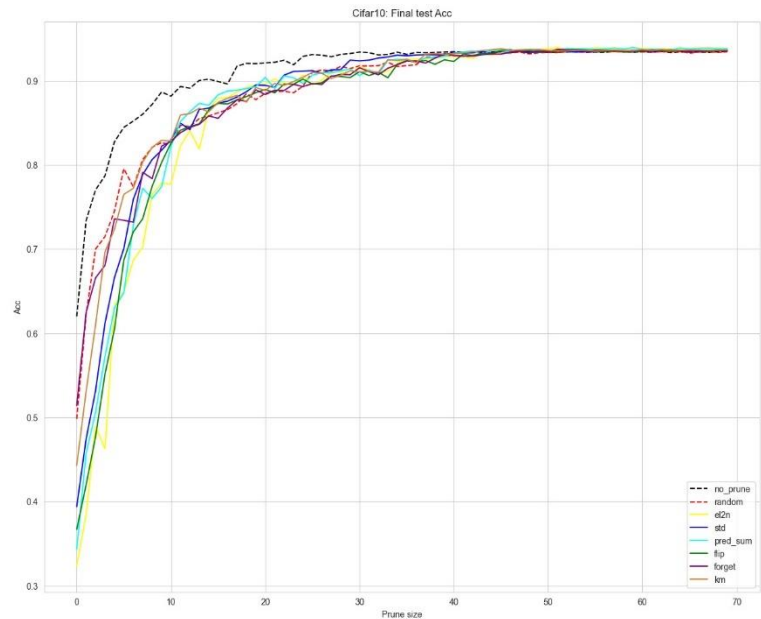
אימון מודל על חלק מהדאטה והוספת דאטה תוך כדי אימון

בניסוי נוסף שעשינו המודל למד בהתחלה רק על חצי מהדוגמאות ואז תוך כדי האימון הוספנו עוד מהדוגמאות הקלות לקשות עד שבסוף המודל אומן על כל הדאטה. מטרת הניסוי היא לראות אם אפשר להגיע בדרך כזאת למודל עם שגיאה יותר קטנה. בצד ימין המודל אומן בהתחלה על 50% מהקשות ובכל epoch הוספנו עוד 10% מהדוגמאות הפחות קשות עד שהגענו לקלות ובצד שמאל ההפך.

אימון מודל מהקלות לקשות



אימון מודל מהקשות לקלות



תיאור מאגר הקוד

כל הקוד שבעזרתו בוצע המחקר נימצא [כאן](#), הקוד כולל מחברות Jupyter שבהם נעשו חישוב המדדים והניסויים כאשר בראש כל מחברת כתוב איזה ניסויים היא כוללת או איזה מדדים חושבו בה. בנוסף יש כמה קבצי Python שכוללים את הקוד המשותף בין המחברות.

ביבליוגרפיה

- [1] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. In ICLR, 2019. [link](#)
- [2] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. Adv. Neural Inf. Process. Syst. 34, December 2021. [link](#)
- [3] Sorscher, Ben, et al. "Beyond neural scaling laws: beating power law scaling via data pruning." arXiv preprint arXiv:2206.14486 (2022). [link](#)
- [4] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. "Unsupervised learning of visual features by contrasting cluster assignments." In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 9912–9924. Curran Associates, Inc., 2020. [link](#)

Abstract

This project is a research project with Dr. Yehuda Hassin about deep learning, the purpose of the project is to test whether and how it is possible to prune part of existing dataset and obtain comparable results on deep learning models in order to save resources (time, computing power, data collection) and/or achieve better performance of the model error.

There are researches that have already been done on the subject with various suggestions and metrics on how to evaluate the quality and difficulty of the data, find the most significant data for training and prune the rest at the very beginning of the learning process of the model, within the project we implemented them and offered additional metrics to evaluate the quality and importance of each data training example according to information that we have on the dataset already at the beginning of learning and we divided the data according to the level of difficulty that each metric gave us.

In addition, we implemented a self-unsupervised metric KM that knows how to classify the dataset according to the level of difficulty without receiving labels and additional information about the dataset, so KM metric can also be useful in problems where we try to learn data without labels with the help of self-supervised models. To test how each part of the data affects the learning process of the model, we performed a series of experiments that tested how each part of the data affects the model's error and we found that if there is a large enough data then the model will learn better from the more difficult examples and we can give up on the least important part of the dataset and on the other hand if the dataset is small then the model also need the easy examples.

Another common problem with datasets is that in order to prepare the data for training, we need to hire people to look at the data and label it, and because it is a large amount of data, there are sometimes mistakes in the labeling that can confuse the model and damage its correctness. In this project we found that with the help of EL2N and pred sum metrics it is possible to find the "noisy" examples and remove them from the data.



Software Engineering Department

Selection of Significant Data for Training

by

Bezalel Cohen

Academic Supervisor: Dr. Yehuda Hassin