עבודת סוף קורס:מבוא לבינה מלאכותית

מגישה:קארין בנימיניאן

ת.ז.:207375098

et

**הדאטה-סט שבחרתי:**

ה-dataset שבחרתי מkaggle- זה דאטה מאתגר של לוגיים של רכבים.



קישור לdataset: <https://www.kaggle.com/datasets/volkandl/car-brand-logos>

**למה מאתגר?**

הדאטה-סט מורכב מ-8 סוגי לוגיים שונים, כאשר לכל לוגו יש כ-300~350 תמונות.

כלומר אין המון תמונות לכל קלאס ולכן משימת הלמידה תהיה כאן מאתגרת.

**למה בחרתי דאטה-סט מאתגר:**

1. דאטה-סט שבו אין דאטה עשיר לכל קלאס, יגרור overfit מהיר יותר.

זה ימחיש בדיוק איך השיטות השונות שלמדנו (Regularization, Dropout וכו') יצליחו לשפר את הדיוק על הtest**,** ויענו בפועל על סעיף 6 בעבודה.

1. מגבלות חישוב.

השתמשתי ב-Google Colab ובמחשב עם RTX3080 עבור אימון רשתות הנוירונים. אך, מכיוון שהרצתי המון מודלים שונים (מודל בסיס, מודל הבסיס עם Batch Normalization וכו') וגם ביצעתי חיפוש להיפר-פרמטרים**,** זה לא היה פרקטי להריץ את המודלים על דאטה-סט מאוד גדול.

**שיטות לשיפור הלמידה ע"י דאטהסט קטן יחסית:**

* כדי למנוע Overfitting וכדי ליצור גיוון בדאטה, השתמשתי בData Augmentation (סיבוב, הזזה, שינוי צבעים וכו').

טרנספורמציות לדוגמה: (בנוסף לשינוי צבעים)

תמונה שמכילה שרטוט, מלבן, צילום מסך, תרשים

התיאור נוצר באופן אוטומטי

דגימות מתוך הדאטה החדש:

תמונה שמכילה צילום מסך

התיאור נוצר באופן אוטומטי

כלומר במקום למזער את השגיאה האמפירית על ההתפלגות שלנו(Empirical Risk Minimization), אנחנו ממזערים את ה- Vicinal Risk Minimization, כלומר התפלגות מאוד דומה להתפלגות המקורית, שתעזור לנו ללמוד את ההתפלגות שלנו בצורה מוכללת יותר, כלומר המודל שנקבל יהיה רובסטי לדאטה חדש שאינו למד עליו, אך זהה לדאטה שלנו.

* חילקתי את הדאטה (שאינו הtest data) לtrain data וvalidation data באחוזים של 0.8 ו0.2 בהתאמה.

**המודל שבחרתי:**

בחרתי במודל CNN (רשת נוירונים מבוססת קונבולציות)

תמונה שמכילה תרשים, תוכנית, קו, שרטוט

התיאור נוצר באופן אוטומטי

עבורו ניסיתי וארציות שונות:

* מודל CNN בסיסי (קונבולציות ושכבות Fully Connected בראש הרשת ניורונים)
* מודל CNN עם Batch Normalization.
* מודל CNN עם Dropout.
* מודל CNN עם Dropout וBatch Normalization משולב.
* המודל המשולב יחד עם L1 Regularization.
* המודל המשולב יחד עם L2 Regularization.

**למה בחרתי במודל CNN?**

כפי שלמדנו בהרצאות ובתרגול על מודל CNN יחד עם הדוגמה מהתרגול על dataset MNIST, מודל ה-CNN מצליח להכליל פיצ'רים לדאטה, ולהגיע לאחוזי דיוק גבוהים יותר על תמונות ממודלי Fully Connected.

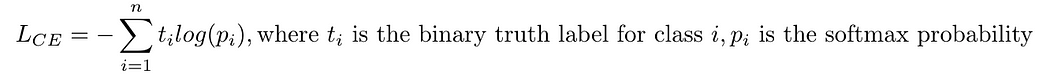
**למה השתמשתי ב-Batch Normalization, Dropout, Regularizations?**

מכיוון שהדאטה מאוד קטן ומאתגר, overfitting זו תופעה שככל הנראה תופיע במקרה שלנו, ולמדנו שיטות שונות כדי לנסות לפתור את הבעיה. עם השיטות השונות והשילובים שלהם, אני מראה את האסטרטגיה לבחירת המודל שלי, וממחישה את מה שלמדנו בקורס.

**פונקציית הסיכון (ענישה) שבחרתי:**

הבעיה שבחרתי לפתור זו בעיית Classification לכן בחרתי בפונקציית הסיכון הכי נפוצה לבעיה.

כפי שלמדנו, בבעיות קלסיפיקציה עבור כמה classים שונים, פונקציית הסיכון הכי נפוצה זו ה-Cross Entropy Loss.



פונקציית ה-loss הנ"ל, מחשבת וקטור הסתברויות (באמצעות Softmax) עבור הclassים השונים.

תמונה שמכילה טקסט, צילום מסך, גופן, מלבן

התיאור נוצר באופן אוטומטי

בתמונת דוגמה הנ"ל, בזמן הערכת הביצועים של המודל, ה-class שמקבל את הscore הכי גבוה, זה הclass שנבחר עבור אותה דגימה (תמונה קלט). לדוגמה בתמונה, נבחר את הclass עבורו התקבל 0.775 מכיוון שזה הclass שקיבל את ההסתברות הכי גבוהה מהמודל שלנו.

**זמן האימון:**

על RTX 3080, האימון עבור כל המודלים יחד אחד אחר השני לקח סה"כ כ-20 שעות.

**זמן inference:**

זמן הריצה במודל evaluation לוקח בממוצע כ0.009 לכל המודלים. כלומר המודלים אינם כבדים (בעלי הרבה פרמטרים), ועובדים בזמן אמת.

**Underfitting וOverfitting במקרה שלנו:**

**Underfitting** זה מצב בו המודל אינו מצליח ללמוד פיצ'רים מהדאטה הנלמד (Training data) ולכן נקבל שגיאת loss גבוהה על הדאטה הנלמד.(שגיאת אימון)

**Overfitting** כפי שלמדנו, זה מצב בו המודל לומד פיצ'רים ייחודיים רק לדאטה הנלמד (משנן את הדאטה הנלמד) ולכן שגיאת האימון תהיה נמוכה, אך הוא לא לומד פיצ'רים מוכללים גם עבור דאטה מאותה ההתפלגות שאינו למד עליו, ולכן נקבל שגיאת ולידציה ושגיאת טסט מאוד גדולה.

**תמונה שמכילה טקסט, קו, תרשים, עלילה

התיאור נוצר באופן אוטומטי**

**נוכל לקבל underfit מסיבות שונות:**

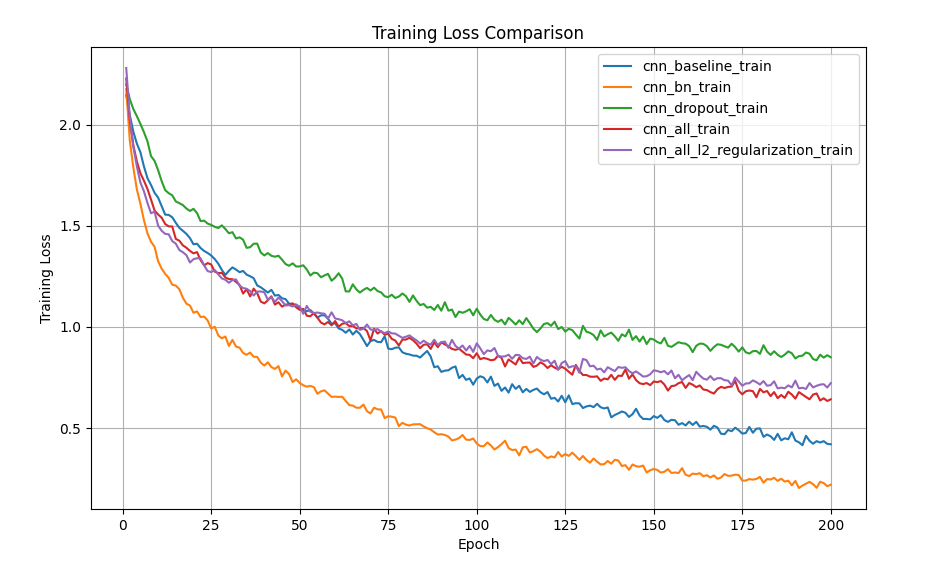
1. המודל שבחרנו יכול להיות אינו עשיר מספיק (כמות פרמטרים קטנה, כמות שכבות קטנה, ארכיטקטורה לא טובה) ולכן אינו יצליח ללמוד פיצ'רים טובים מהדאטה הנלמד.
2. כמות הדאטה אינה גדולה (כמו במקרה שלנו) ואינו מגוון, הדבר יכול לגרום למודל להתקשות ללמוד פיצ'רים שמתאימים לכל דוגמאות האימון מהדאטה ולכן אינו מצליח ללמוד.

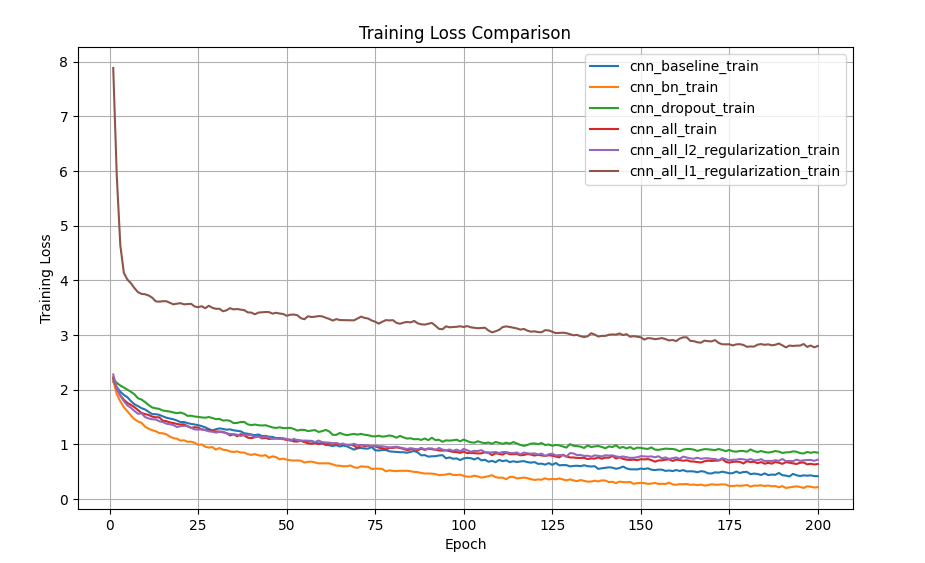
יש עוד סיבות שונות שיכולות לגרום לunderfit כמו היפר-פרמטרים שאינם טובים, מספר epochs קטן מידי וכו'.

כעת אראה את הגרפים ואילו תופעות קרו באימונים שביצעתי.

**הערה:** אימנתי פעמיים את המודלים עם סיד שונה כדי לוודא את התוצאות. הגרפים היו נראים כמעט זהים לגמרי עבור שתי הריצות, הגרפים הבאים של ריצה מספר 2.

**גרף שגיאת אימון ללא L1 regularization (בשביל scale טוב יותר):**



**גרף שגיאת אימון מלא:**

**מסקנות:**

1. נוכל לראות לפי הגרפים שעם המודלים שבחרנו, למרות הדאטה, לא קיבלנו underfit כי המודלים מצליחים להגיע לשגיאת אימון נמוכה, למעט המודל עם l1 regularization.
2. כפי שלמדנו, l1 regularization מוביל ל sparse weights. כלומר להמון ערכי משקולות השוות לאפס.

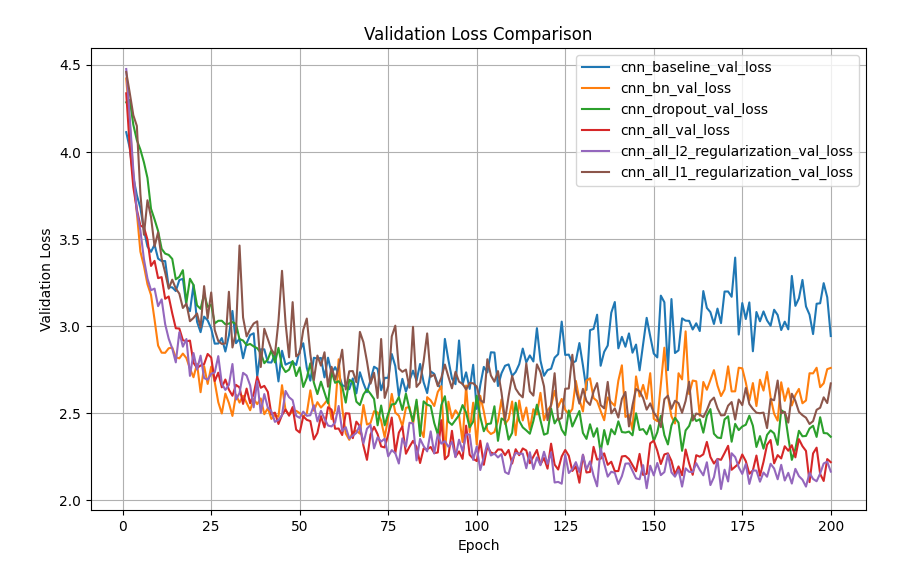
בשיטה זו, כנראה הגבלת המודל למשקולות sparse-יות מנע מהמודל לייצר משקולות שמצליחות לזהות את הפיצ'רים החיוניים מספיק ללמידה דאטה האימון בצורה מושלמת.

(הסבר נוסף על Lasso regression והרחבה על הsparse-יות בהמשך העבודה)

(הקוד מצורף בקבצים הנוספים)

**נוכל לקבל overfit מסיבות שונות:**

1. כאשר יש כמות דאטה שאינה מספיק גדולה, המודל עלול "לשנן" את דוגמאות האימון.
2. אימון ארוך מידי (מספר epochs גדול מידי של אימון)

כפי שלמדנו ישנן סיבות נוספות, אך נשים לב שבמקרה שלנו (בגרפים הבאים) כנראה אלו שני הסיבות שהובילו לOverfitting.

**מסקנות:**

1. ניתן לראות בבירור שהמודל הבסיסי, ללא שום שיטה למניעת Overfitting (גרף כחול), מאוד מהר ביצע overfit.
2. ניתן לראות שכל השיטות השונות למניעת overfitting הצליחו לעכב ואף למנוע את התופעה.
3. המודלים שהצליחו ללמוד פיצ'רים טובים יותר ומוכללים של הדאטה, זה המודל המשולב (עם Batch Normalization ו(Dropout ואותו המודל המשולב בתוספת נוספת של L2-regularization.

**תמונה שמכילה טקסט, עלילה, קו, תרשים

התיאור נוצר באופן אוטומטי**נשים לב בגרף מטה, גרף הדיוק על הvalidation dataset, כי כמה מודלים הצליחו להגיע לדיוק מאוד גבוה.

אך, בניגוד לסיווג נכון של התווית בתמונה, הגרף הקודם על הValidation Loss, מראה לנו שהcross-entorpy loss של המודל עם הl2 regularization, הרבה יותר טוב.

כלומר המודל הרבה יותר בטוח בעצמו כשהוא מסווג את התוויות הנכונות של התמונות (כלומר בוקטור ההסתברות שהמודל פולט, התווית הנכונה קיבל הסתברות גבוהה ממש ביחס לאחרות)

**המסקנה הסופית מכל האימון:** ככל הנראה, המודל המשולב על l2 regularization הוא המודל שיקבל את הדיוק הגבוה ביותר על הtest dataset. זאת מכיוון שהוא גם הצליח ללמוד בצורה טובה את הtrain data אך גם למד פיצ'רים מוכללים שמצליחים לסווג נכון גם את הvalidation data.

**בחירת היפר-פרמטרים:**

השתמשתי בשיטת Grid Search לחיפוש הפרמטרים הטובים ביותר עבור המודל שחזיתי שישיג את התוצאות הטובות ביותר. (מגבלות חישוב).

למעשה,Grid Search במודל למידה, מבצע אימון עבור כל קומבינציה של היפר-פרמטרים ובוחר את הערכים הטובים ביותר.

לאחר שביצעתי מספר ניסויים עם המודלים שונים, ולאחר שבניתי את המודל הטוב ביותר כפי שלמדנו בכיתה, ביצעתי חיפוש פרמטרים עבור אותו המודל (מופיע בתוצאות ההמשך).

התוצאות הטובות ביותר שקיבלתי היו עבור הפרמטרים שניסיתי ממורקרים בצהוב.

|  |  |  |
| --- | --- | --- |
| Grid Search | | |
| 256 | 128 | Batch size: |
| 0.01 | 0.001 | Learning rate: |
| 0.5 | 0.25 | Dropout rate: |
| 0.001 | 0.0001 | Regularization strength: |

בשל מגבלות חישוב ביצעתי אימון של 50 epochs עבור כל שילוב פרמטרים בטווחים שהסקתי שיכולים להיות חיוניים לאימון (אחרי המון ניסויים קטנים). אם לא היו מגבלות חישוב, היה ניתן לבדוק טווח ערכים רחב יותר עבור כל היפר-פרמטר.

**הערה**: בנוסף לקוד הGrid Search עבור המודל הטוב ביותר, צירפתי גם בקוד אפשרות להריץ חיפוש פרמטרים הטובים לכל מודל ומודל, אך בשל מגבלות חישוב הדבר אינו פרקטי ואינו יעיל ולכן הקוד נמצא בהערות ורק כהדגמה.

**תוצאות סופיות:**

ריצה 1:

|  |  |
| --- | --- |
| מודל | דיוק על הTest dataset- |
| CNN Base Model | 64.50% |
| CNN – Dropout | 64.25% |
| CNN - Batch Normalization (BN) | 63.25% |
| CNN - Dropout & BN | 67.75% |
| CNN - Dropout & BN & L2- regularization | 68.25% |
| CNN - Dropout & BN & L1- regularization | 63.75% |

בריצה הראשונה, ניתן לראות כי אמנם מודל הבסיס הגיע לתוצאות קצת טובות יותר ממודל הBN, אך קצב ההתכנסות של מודל הBN משמעותית מהיר יותר. (ניתן לראות זאת בגרפים)

בנוסף, מודל הבסיס הגיע מהר יותר לOverfitting ביחס לשאר המודלים כצפוי.

ריצה 2:

|  |  |
| --- | --- |
| מודל | דיוק על הTest dataset- |
| CNN Base Model | 61.25% |
| CNN – Dropout | 65.75% |
| CNN - Batch Normalization (BN) | 65.25% |
| CNN - Dropout & BN | 68.00 |
| CNN - Dropout & BN & L2- regularization | 69.00% |
| CNN - Dropout & BN & L1- regularization | 62.00% |

בריצה הזאת, הפעם מודל הבסיס נתן את הדיוק הנמוך ביותר לפתירת המשימה ושוב היה בOverfitting- ביחס לדאטה האימון.

בנוסף, המודל הטוב ביותר כפי שחזינו, היה המודל שמשלב את כל השיטות יחד עם L2 regularization. הבדל של כמעט 8% דיוק מהמודל המקורי!

שני הריצות מאשרות כי השיטות השונות שלמדנו, אכן מונעות או מעכבות Overfitting ושילוב כל השיטות מביא לתוצאות הכי טובות.

**הערה נוספת על התוצאות:**

במבט ראשוני זה נראה שהתוצאות אינן מרשימות כי הן לא באחוזון ה90%+. אך התוצאות שלנו ברות השוואה וכן טובות ביחס לדאטה משתי סיבות.

1. יש לנו מעט דאטה עבור כל class ודאטה שמורכב מתמונות מאתגרות ולכן קשה מאוד ללמוד בצורה מוכללת את הפיצ'רים.
2. ב-Kaggle יש מחברת אחת ששיתפו על הדאטה, בה השתמשו ברשת מאומנת של VGG19, רשת גדולה יותר משלי עם המון פרמרטרים והיא אומנה לפני על הדאטה סט Imagenet, דאטהסט עם 14 מיליון תמונות.

הם עשו לה Finetune עבור הדאטה המאתגר, והתוצאה שהגיעו אליה היא 74% דיוק על הvalidation.

אני הגעתי ל69% בשיא בלי אימון על דאטה מוקדם יותר לכן התוצאות מספקות מאוד, במיוחד לרשת קונבולציה בסיסית, שהצלחנו לנצל את יכולת הלמידה שלה כמה שיותר, באמצעות שיטות למניעת Overfitting!

**תוצאות נוספות על המודל המנצח:**

**Confusion Matrix:**

תמונה שמכילה טקסט, צילום מסך, מלבן, ריבוע

התיאור נוצר באופן אוטומטי

**פירוט על המטריצה:**

תמונה שמכילה טקסט, צילום מסך, מספר, תרשים

התיאור נוצר באופן אוטומטי

**הmatrix confusion-** זו מטריצה שמסכמת את הביצועים של המודל במשימה הקלסיפיקציה שלנו.

בכל פעם שמסתכלים על הביצועים עבור class, **i - מסוים**, התא בו הפרדיקציה שלנו הייתה נכונה (התא המתאים באלכסון), זה יהיה מספר הזיהויים הנכונים של המודל True Positive, ונשאף שהמספר בו יהיה הכי גבוה.

בשאר התאים, יהיו הFalse Negative, False PositiveוTrue Negative עבור אותו הקלאס.

True Positive (TP) – מספר הדגימות של הקלאס i, שסווגו נכון.

True Negative (TN) – מספר הדגימות שאינן של אותו קלאס i, ולא סווגו כקלאס i.   
False Positive (TP) – מספר הדגימות שאינן של אותו קלאס i, אך כן סווגו כקלאס i.

False Negative (FN) – מספר הדגימות של קלאס i, שסווגו כקלאס אחר שאינו i.

**תמונה שמכילה טקסט, גופן, מספר, קבלה

התיאור נוצר באופן אוטומטי** **Metrics Table:**

**פירוט על המטריקות:**

1. Recall – מעריך את הרגישות של המודל עבור אותו הclass, כלומר בהינתן דגימות שליליות (שאינן של אותו האובייקט) וחיוביות, הוא מעריך מה היחס בין הדגימות החיוביות שהוא סיווג נכון לבין כל הסיווגים שלו לאותו הclass. (גם לא נכונים)
2. Precision – מדד למידה שמציין את אחוז הדיוק בהינתן אותו הclass, כלומר כמה המודל מסווג נכון את הדגימות החיוביות שבידיו ולא מסווג אותם כclass אחר.
3. F1 Score – שילוב של ה-Recall וה-Precision שתופס גם את הרגישות של אותו class ביחס לclass-ים אחרים וגם את הדיוק עבור אותו הclass.

**המשוואות למטריקות:**

1.

2.

3.

**תמונה שמכילה טקסט, צילום מסך, מלבן, מספר

התיאור נוצר באופן אוטומטיבונוס: השוואת הsparse-יות במודלים עם הרגולריזציה**

הערה: בpython 1e-4 = 0.001

ניתן לראות שכפי שלמדנו, המודל עם L1 regularization,הוא מודל sparse בהרבה ביחס למודל המנצח.

האינטואיציה אומרת שמכיוון ש-Lasso regression (L1 regularization) מכריח את המודל להיות sparse, במשימה שלנו זה יגביל את המודל מלעשות overfitting על הפיצ'רים של הדאטה הנלמד, אך עלול למנוע מהמודל ללמוד פיצ'רים חשובים.

במקרה שלנו, בהחלט ניתן לראות שאינו ביצע overfitting, אך המודל מתקשה עם דאטה שאינו ראה (הולידציה). למרות כל זאת, הוא הגיע לתוצאת דיוק ברת השוואה לשאר המודלים על אחוז השגיאה בvalidation ובאחוז הדיוק על הtest. (62% שאפילו ניצח את מודל הבסיס).