

פרויקט גמר בהשראת המאמר Character-level Convolutional Networks for Text Classification

כותב: אדם יערי, אוניברסיטת בר-אילן

מנחה: ד"ר יואב גולדברג, אוניברסיטת בר-אילן

מבוא:

קיטלוג (קלסיפיקציה) של טקסט היא משימה קלאסית בתחום עיבוד שפה טבעית (NLP) במדעי המחשב, בה נעשה קיטלוג של טקסט, הניתן כקלט למערכת, לאחת מקבוצת הקטגוריות המוגדרות מראש. במשימה זו המחקר העכשיו מתמקד בשאלות מגוונות וברמות מורכבות שונות, משינויים יחסית קטנים של מאפיינים (פיטצ'רים) במערכת בכדי למצוא את הכיול האופטימלי ללמידת המערכת, וכלה בבחירת ועיצוב מקטלגים שונים למשימה, המתבססות על תיאוריות שונות להבנת הכתוב. נכון להיום, כמעט כל הטכניקות לביצוע קיטלוג מתבססים על שיטות להבנת הטקסט המתבססות על מילים מוטבעות (*Embedded words*) ע"י טכניקות כגון *Word2Vec* ו-*Lookup Table*. בעבור הארכיטקטורות האלה ניתן לדמין את הבנת הכתוב באופן דומה לדרך בה אנחנו כבני אדם קוראים משפט ומסוגלים להגיד אם הטקסט מדבר על נושא כזה או אחר. בשיטות אלו סטטיסטיקות פשוטות, של רצף מילים מסודר כלשהו, (כדוגמת *n-gram*) לרוב מתפקדות הכי טוב [2].

הבעיה בשיטות אלה היא שהן דורשות ידע מקדים על השפה שאיננו זול – מילון מוכן מראש, דרך התמודדות וקיבוץ של מילים עם משמעות דומה, הפרדה בין מילים זהות בעלות משמעות שונה וכדומה. דרישות אלה הופכות את המשימה של הבנת הכתוב למוכונת לשפה ספציפית, כך שכאשר נרצה להשתמש בכלי ששימש להבנת טקסט בשפה א' עבור הבנה של שפה ב', נאלץ לעשות שינויים רבים ומשמעותיים בארכיטקטורת הכלי.

מצד שני, מחקרים רבים בתחום רשתות הניורונים הראו של- *CNN Convolutional Neural Networks* יכולות גבוהות בחילוץ מידע מאותות פשוטים (*raw signals*) [3,4] בתחומים רבים ומגוונים כגון: ראייה ממוחשבת (*Computer vision*), הבנת דיבור (*Speech recognition*) ועוד. ברשתות קונבולוציה המידע נאסף בעזרת מעבר בין מספר שכבות קונבולוציה, כאשר בכל שכבה יש מספר רב של פילטרים הסורקים את האות הניתן להם (האות הראשוני או הפלט של אחת משכבות הקונבולוציה הקודמות). הפילטרים הללו מורכבים ממשקולות המאומנות עם הזמן למציאת מאפיינים שונים, מורכבים יותר ואינפורמטיביים יותר מבשכבה הקודמת להם, כך שבסופו של דבר הם מסוגלים לרמוז על שיוך לקטגוריה כזו או אחרת.

בפרויקט זה התבססתי על המאמר "*Character-level Convolutional Networks for Text Classification*" והוא הראשון המתייחס למשפט כתוב כאות פשוט של אותיות (*Character level*), ומימש רשת קונבולוציה חד-מימדית הסורקת את המשפט הנתון ומקטלגת אותו לאחת מהקטגוריות הנתונות לה מראש. כותבי המאמר ציינו כי למרות, שבמאמרם הם הדגישו את הצלחת רשת הקונבולוציה על משימות קיטלוג בלבד, הם מאמינים כי היכולות של רשתות *CNN* לטובת הבנת הטקסט כאות פשוט הן רבות ומגוונות יותר מהמוצג במחקר הנ"ל, בהינתן קלט אימון גדול מספיק.

בנוסף לכך, שימוש ב- *CNN* לטובת הבנת טקסט הינה אפשרות שנחקרה בעבר והוכחה כיעילה עבור שימוש ישיר על מילים מוטבעות [5,6,7], גם ללא ידע מקדים על מבנה סינטקטי או סמנטי בטקסט, ונמצא כמתחרה עבור שיטות הקלסיפיקציה המסורתיות.

שיטת ה- *Text Understanding From Scratch* שמציע המאמר הנ"ל מאפשרת שיפור הן של גנריות והשימושיות של השיטה, והן הפחתה של מגוונים מורכבים בדרך להבנת הכתוב באופן הבא:

1. יש שימוש באותיות ולא במילים – חוסך את השימוש המיותר בשיטות הטבעת המילים לווקטור, שהינן מסובכות להפקה בפני עצמן ומייצרות מרחב גדול מאוד של ווקטורים אפשריים (*High dimensionality*).
2. אין צורך בידע סינטקטי או סמנטי מקדים על השפה – הסקה לרמות גבוהות יותר של עיבוד מספיקה. מה שמפריך את ההנחה כי יש צורך שהנחות מובנות ומודלים של שפה נדרשים לצורך הבנת הטקסט, ומאפשר שימוש באותה שיטה עבור שפות רבות.

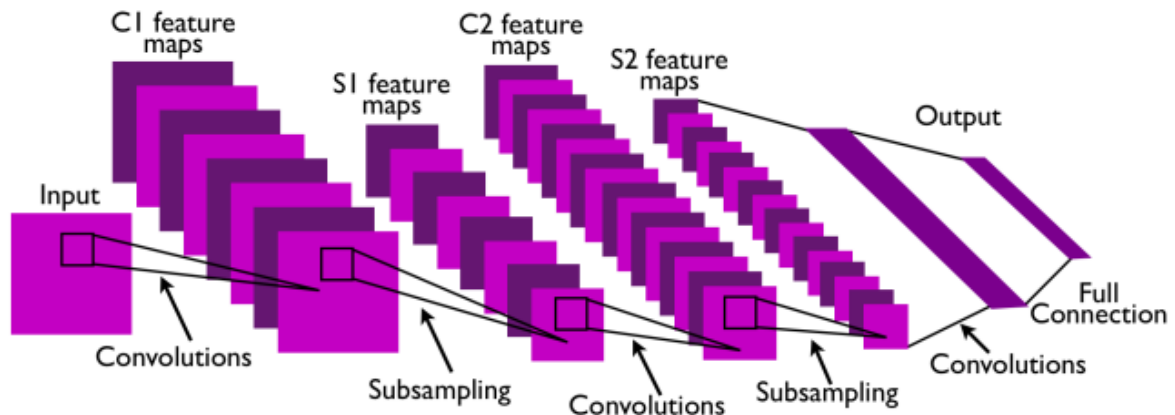
עיצוב רשת ה-CNN

בחלק זה אתאר בקצרה מבנה של רשת CNN סטנדרטית ולאחר מכן ארחיב יותר על פירוט רשת ה-CNN שתוארה במאמר המקור, ושאותה מימשתי בפרויקט שלי.

רשת CNN סטנדרטית:

באופן כללי *Convolutional Neuron Networks* מתבססות על 3 סוגי שכבות עיקריות:

- **Convolutional Layer:** זו השכבה המקנה לרשת את שמה ומהווה את השכבה העיקרית ברשת CNN סטנדרטית. בכל שכבה מסוג זה ישנם מספר פילטרים/חלונות בגודל זהה אך בעלי משקולות שונות הנלמדות במהלך ה- *Back Propagation*. פלט השכבה מיוצר ע"י קונבולוציה של כל אחד מהפילטרים הנ"ל עם מטריצת הקלט של השכבה, ובכך מייצר פלט תלת-מימדי שמימדיו תלויים בגדלי הפילטר, הקלט ומספר הפילטרים¹. פלט זה תוצר מועבר דרך פונקציית אקטיבציה, שהינה לרוב לא לינארית (כגון: *sigmoid, tanh, ReLU* וכו'), טרם העברת התוצאה כקלט לשכבה הבאה. במהלך האימון ניתן לראות כי כל אחד מהפילטרים לומד לזהות מאפיין תבניתי כלשהו, שרמת מורכבותו עולה עם עומק השכבה בה הפילטר נמצא.
- **Pooling/Subsampling Layer:** שכבה זו מהווה צמצום של פלט הקונבולוציה ע"מ למזער את כמות המידע² שיש צורך להעביר בין השכבות ולהוריד את זמן החישוב. מטרת שכבה זו היא לבחור את האיברים החשובים בכל חלון בו היא מסתכלת, למשל ברוב המצבים נחשיב את האיבר הגדול ביותר בחלון מסויים להיות המשמעותי ביותר בחלון, ולכן נשתמש ב- *Max Pooling*.
- **Dense/Fully-Connected Layer:** שכבה זו היא למעשה שכבת רשת נוירונים רגילה והפשוטה ביותר, בה כל נוירון בשכבה ה- i מחובר לכל נוירון בשכבה ה- $i+1$. שכבה אחת או מספר שכבות כאלה תמיד יהיו האחרונות ברשת CNN ובהן יבוצע מעין "סיכום" של תוצר הרשת, שבסופו של דבר גם ייתן את התוצאה הסופית, למשל בעזרת פונקציית *Softmax* עבור קלסיפיקציה.



*איור 1: תמונה להמחשה של מבנה רשת CNN בעלת שתי שכבות מכל סוג המצויין לעיל.

חשוב לציין כי יתרונות הרשת הן חיסכון בזיכרון (ביחס לרשת *Fully Connected* באותו הגודל), היות והמשקולות חוזרות על עצמן עבור כל איבר בקלט³, ויש לה יכולות גבוהות באבחון מאפיינים מרחביים ספקטראליים כאשר קלט הרשת הוא יחידות בסיס קטנות ופשוטות, כגון פיקסלים בתמונה או פונמות בקלט אודיטורי, בדומה לאופן פעולת מערכת הראייה האנושית המזהה אובייקטים מורכבים מקלט של נקודות אור נפרדות.

1. בגרסה בה השתמשתי לא נעשה ריפוד אפסים בקצוות הקלט ולכן גודל תוצר הקונבולוציה המתקבל עבור קלט בגודל $h \times l \times k$ פילטרים בגודל $d \times r$ הוא $(l - d + 1) \times (h - r + 1) \times k$.
2. עבור חלון בגודל 3×3 ממנו נבחר רק איבר 1 נוכל לצמצם את גודל המטריצה פי 9.

3. מספר המשקולות תלוי בגודל ומספר הפילטרים שקטן משמעותית מגודל הקלט, שמהווה את מספר המשקולות בשכבה Fully Connected.

Character-level CNN עבור קיטלוג טקסט⁴:

• מרכיבי מפתח:

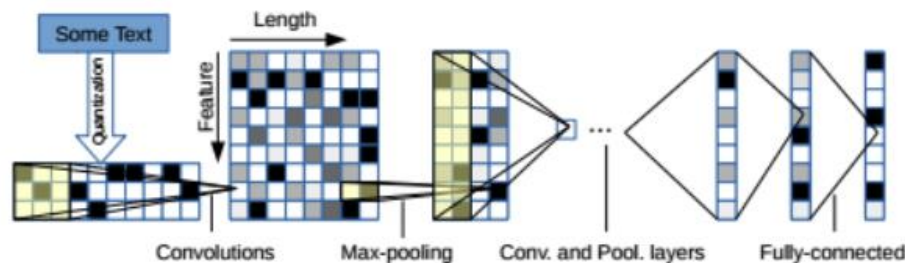
- הקונבולוציה במערכת זו הינה חד מימדית, כלומר שעבור כל פילטר ניתן רק מימד אחד (d) כגודל הפילטר ומספר פילטרים (k), ומשמעותו הוא שעבור מטריצת קלט, בגודל $l \times h$, נבצע קונבולוציה עם k פילטרים בגודל $d \times h$, שתייצר פלט דו-מימדי בגודל $(h - d + 1) \times k$.
- שיטת ה- Pooling ברשת זו היא Max Pooling, בה נעשה שימוש גם ב- CNN בראייה ממוחשבת.
- פונקציית האקטיבציה היא Threshold ReLU⁵ בכל שכבות הרשת (קונבולוציה ומלאות), כך שהסף המינימלי נקבע להיות 10^{-6} .
- שיטת למידת המשקולות היא Stochastic Gradient Decent עם גודל באטץ' של 128, מומנטום של 0.9, קצב למידה 0.01 וקבוע רגולריזציה של 10^{-6} .
- המשקולות ההתחלתיות הן רנדומליות עם התפלגות גאוסיאנית ושונות של 0.05.

• קוונטיזציה (Quantization) האותיות:

במודל זה ישנה למעשה קוואנטיזציה של אותיות ממאגר ידוע מראש לווקטורים בגודל קבוע, המחליף את מקומן של ווקטורי המילים המוטבעות. בהינתן שישנן m אותיות שונות המתקבלות כקלט לרשת, הקוואנטיזציה נעשית בשיטת 1-to- m ⁶ ומייצרת m ווקטורים שונים. עבור כל אות שנקראת במשפט ואינה חלק מאותן m אותיות הקוואנטיזציה תחזיר ווקטור אפסים בגודל m . במאמר השתמשו ב-70 אותיות ותווים מהא"ב הלטיני (26 אותיות אנגלית, 10 ספרות ו-34 תווים הכוללים גם את תו הרווח). להלן 70 התווים (ללא הרווח):
`abcdefghijklmnopqrstuvwxyz0123456789-.,!?:'''\^/_@#%&^*~'+<>()[]{}`
 בנוסף, הם הגדירו אורך $l_0 = 1014$ שהגביל את אורך הקלט הנקרא, וכל אות מעבר לאותם 1014 תווים לא נקראה, כך שאורך הטקסט לא יהיה אחד מהגורמים המרמזים לקלסיפיקציה ולא יהיה צורך להתמודד עם אורך משתנה (לא היו טקסטים קצרים מ-1014 תווים בסט האימון והבדיקה) במעבר בין הקונבולוציה לשכבות המלאות.

• עיצוב המודל:

המודל הכיל 6 שכבות קונבולוציה ו-3 שכבות מלאות.



*איור 2: אילוסטרציה של המודל.

כאמור לעיל, גודל הקלט היה 1014×70 , עקב מספר שיטת הקוואנטיזציה והגבלת אורך הקלט. כותבי המאמר ציינו כי מבדיקה אמפירית נראה כי 1014 תווים מסוגלים לתפוס את רוב עניין הטקסט הנתון. בנוסף לכך, בין 3 השכבות המלאות הוסיפו שתי שכבות השמטה (Dropout)⁷ עם הסתברות 0.5 בכדי ליצור רגולריזציה.

4. במאמר המקורי פורטו שתי רשתות שונות, זהות מבחינת הארכיטקטורה אך שונות בגודלן, בפרויקט זה מימשתי ועבדתי רק עם הרשת הקטנה מבין השתיים עקב אילוצי חומרה של מחסור במעבד גרפי (GPU) שייצרו זמני ריצה ארוכים במיוחד שלא התכנסו לזמני ההגשה.

5. $\text{Threshold ReLU}(x) = \max(\text{threshold}, x)$.

6. מקרה של שיטת "One Hot Vector" המייצרת m ווקטורים בגודל m כך שעבור האות ה- i האיבר ה- i בווקטור יהיה 1 וכל השאר אפסים.

7. שכבת השמטה משמיטה את הפלט שלה (מאפסת אותו) בהסתברות הנתונה לה (0.5 במקרה שלנו).

טבלה 1 מפרטת את קונפיגורציית חלק הקונבולוציה של הרשת:

מספר שכבה	מספר פילטרים	גודל פילטר	גודל חלון Pooling
1	256	7	3
2	256	7	3
3	256	3	-
4	256	3	-
5	256	3	-
6	256	3	3

בפילטר הקונבולוציה גודל הצעד היה 1, וה- Pooling היה ללא חפיפה. טבלה 2 מפרטת את קונפיגורציית חלק השכבות המלאות של הרשת:

מספר שכבה	גודל פלט
7	1024
8	1024
9	מספר הקטגוריות

במודל זה ע"מ לחשב את גודל הקלט לשכבה 7 (השכבה המלאה הראשונה) ניתן לעשות חישוב בנוסחה הבאה:

$$l_6 = (l_0 - 96)/27$$
כך ש- l_i הוא אורך הפלט של השכבה ה- i .

קישור לקוד

מאגר המידע לעבודה:

בפרויקט זה השתמשתי באחד ממאגרי המידע בו השתמשו במאמר המקורי ע"מ שאוכל להשוות את תוצאות הריצות שלי לתוצאות שהושגו ע"י צמד החוקרים המקורי. המאגר בו בחרתי להשתמש הוא - *AG's news corpus*⁸ המכיל 496,835 מאמרי חדשות מקוטלגים לפי נושאים מיותר מ-2000 מקורות עיתונות. במאגר המידע שנלקח במאמר, ובו גם אני השתמשתי, נלקחו 4 הקטגוריות הגדולות ביותר, ומכל אחת נדגמו 30,000 מאמרים לסט האימון ו-1,900 לסט הבדיקה.

הטקסט שהועבר כקלט לרשת ה- *CNN* הוא שירשור של כותרת המאמר ותיאור הכתוב במאמר, שנקטע לאחר 1014 תווים מהסיבות המפורטות לעיל.

⁸ http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

תהליך העבודה והמודל הסופי

בחלק זה אתאר תחילה את תהליך הלמידה לפני קבלת המשימה הספציפית אותה אני מגיש, ולאחר מכן אתמקד יותר במשימה שעשיתי, במכשולים ובמימוש הסופי שביצעתי.

עבודת ההכנה ומימוש הפרוייקט

ע"מ להבין את התכנים איתם אני עובד במהלך הפרוייקט, ובכדי להיות מסוגל לממש את הנדרש ממני, הייתי צריך ללמוד שני קורסים ייעודיים העוסקים ברשתות נוירונים. הראשון הוא הקורס הבסיסי של ד"ר גדי פנקס הנקרא "רשתות נוירונים" ומתעסק ברמות יותר בסיסיות של סוגי למידת מכונה בעזרת רשתות נוירונים, שהיה חלק מתוכנית הלימודים המתוכננת בתואר שלי, והשני הוא קורס מתקדם יותר של ד"ר יואב גולדברג, מנחה פרוייקט זה, שנקרא "עיבוד שפה טבעית בעזרת רשתות נוירונים עמוקות", קורס שלא היה חלק מהסילבוס של התואר שלי, ושביצעתי כחלק מהפרוייקט כדי להעמיק את הידע שלי בתחום ולהבין את המושגים איתם אתעסק לעומק. במסגרת שני הקורסים הנ"ל, ובנוסף למימוש במפורט בפרוייקט זה, ביצעתי מספר מימושים של רשתות נוירונים רבות המבצעות משימות שונות, מהרמה הפשטנית ביותר של שני נוירונים לינארים הפותרים את בעיית XOR ועד רשת Bi-LSTM המקטלגת טקסט לקטגוריות.

במהלך פרוייקט זה ביצעתי שני מימושים נפרדים לרשת הנוירונים המפורטת לעיל, בשתי שפות *Open Source* ייעודיות לעבודה עם רשתות נוירונים עמוקות – *Keras*⁹ ו-*Pycnn*¹⁰. כאשר מטרת הפרוייקט הייתה לממש את מודל רשת ה-*Character-level CNN* ב-*Pycnn*, ולנסות לשפר את התוצאות המוצגות במאמר עצמו.

Keras הינה שפה גבוהה יותר מבחינת כמות הפונקציות ונוחיות העבודה בה, ולכן מימשי את הרשת בעזרתה, טרם מימוש ה-*Pycnn*. *Keras* איפשרה לי להגיע לתוצאות מהירות יחסית מבחינת מימוש (ומבחינת זמן ריצה), לוודא את ההבנה הבסיסית שלי לארכיטקטורת הרשת ולבצע מספר ניסויים מקדימים למציאת פיטצ'רים ושינויים מבניים יעילים יותר. לעומת זאת, כמו כל שפה גבוהה בעלת פונקציות ייעודיות מרובות ונוחיות עבודה טובה, *Keras* מאפשרת פחות נגיעה בפרטים הקטנים של הרשת ופחות גמישות בכל מה שאינו במסגרת העבודה שלה. אזי, בכדי להגיע לרמת הבנה עמוקה יותר של המימוש, ולוודא כי אכן המנגנון פועל כמו שאנו מצפים ממנו, השקעתי את מרבית זמני במימוש הרשת גם ב-*Pycnn*.

במהלך העבודה, היות ומימוש ה-*Keras* הראה למידה מהירה ויעילה, האתגר המרכזי היה לגרום למימוש ה-*Pycnn* ללמוד באופן זהה, או במילים אחרות ליצור ב-*Pycnn* רשת שקולה לזו של ה-*Keras*. מה שהוסיף לפרוייקט מטרה נוספת הנובעת משתי המטרות הראשוניות, לבצע השוואה בין המתודות הקיימות ב-*Keras*, שבהן עשיתי שימוש, לבין המתודות שכתבתי ב-*Pycnn*.

מודל ה-*Pycnn* הסופי:

כאמור לעיל, מספר הפונקציות המוכנות מראש של *Pycnn* קטן יחסית, ולא עונה על כל דרישות רשת ה-*CNN* אותה רציתי לממש. מצב שיוצר אתגרים לא פשוטים בהתחשב בעובדה שהשפה עובדת עם טיפוס מסוג *Expression* כמחלקת הבסיס שלה, טיפוס יחסית דל בפונקציונאליות וקשה להרחבה (בשונה מעבודה עם טיפוסים פרימיטיביים למשל, כגון *String*, *Integer* וכו'). בחלק זה אפרט את האתגרים העיקריים איתם נאלצתי להתמודד, ואת דרך הפיתרון שאותה אני מציג בסופו שלדבר:

1. אחד האתגרים האחרונים שאיתם התמודדתי, הוא דווקא הראשון שאציג פה, מכיוון שהייתה לו השלכה גדולה על אופן פעולת הרשת הסופית.

הבעיה: ב-*Pycnn* עבודה עם מטריצה כקלט היא בעייתית בשני מישורים, הראשון הוא שלא כמו בעבודה עם ווקטורים, במטריצה אין גישה לאיבר בודד כלל (כמו למשל באופן הבא $Matrix[i][j]$), ושנית צורת העבודה דרשה פעולות *reshape* ו-*transpose* רבות כדי לממש את פונקציות ה-*Pooling* והקונבולוציה, שהן פעולות כבדות מאוד בזיכרון ואיטיות בזמן הריצה בשפה הנ"ל.

9. <https://github.com/fchollet/keras>

10. <https://github.com/clab/cnn>

הפתרון: במקום עבודה עם מטריצה בחרתי לעבוד רק עם ווקטורים, כך שבמקום מטריצה בגודל $m \times n$ עבדתי עם ווקטור בגודל $1 \times (m * n)$, שבו האיבר $i * m + j$ מתאים לאיבר (i, j) במטריצה המקורית. פתרון זה איפשר פונקציית קונבולוציה ופונקציית *Pooling* פשוטות ומהירות משמעותית, בנוסף לצמצום של צריכת הזיכרון לעשירית מהכמות הקודמת¹¹.

2. **הבעיה:** ל- *Pycnn* יש אפשרות לפונקציית *ReLU* סטנדרטית אך ללא אפשרות ה- *Threshold* (שהוכחה בניסויי מקדים ב- *Keras* כי היא הכרחית ללמידה), ובשל העבודה עם טיפוס ה- *Expression* אין אפשרות לבדוק את הערך המספרי של תוצאה במהלך אמצע הריצה בזמן עבודה סביר.

הפתרון: מכיוון שערך ה- *Threshold* מאוד קטן (10^{-6}) השתמשתי בפונקציית ה- *ReLU* הקיימת ולכל התוצרים הוספתי את ערך ה- *Threshold*¹².

3. **הבעיה:** ל- *Pycnn* אין פונקציית *Max Pooling* שבעזרתה נעשה צמצום המימדים ב- 3 מתוך 6 שכבות הקונבולוציה, בנוסף לבעיית הערך המספרי של ה- *Expression* הנ"ל.

הפתרון: ל- *Pycnn* כן יש פונקציית *KMax Pooling* הבוחרת את k האיברים הגבוהים ביותר מהווקטור הנתון לה, לכן השתמשתי בה באופן הבא: לאחר כל d צעדים של הקונבולוציה, כאשר d הוא גודל חלון ה- *Pooling* (במקרה שלנו 3 בכל השכבות), שרשרתי אותם לווקטור v והעברתי אותו לקלט לפונקציית *KMax Pooling* עם $k = 1$. מה שבסופו של דבר נתן *Max Pooling* לכל d אברים.

4. **הבעיה:** ל- *Pycnn* אין פונקציית קונבולוציה כלל (לא חד-מימדית ולא דו-מימדית).

הפתרון: עבור גודל פילטר fl מספר פילטרים בשכבה fn וגודל מסגרת הקלט (גובה המטריצה) h , ייצגתי כל פילטר כווקטור באורך $fl * h$ וקיבצתי אותם ביחד למטריצה מסדר $(fl * h) \times fn$, וביצעתי כפל מטריצה בווקטור עבור כל מיקום i בווקטור הקלט של השכבה, כך ש- i גדל בקפיצות של h . כל הכפלה כזו של המטריצה הנ"ל בווקטור בגודל $1 \times (fl * h)$ יצרה ווקטור חדש בגודל fn , אותו שרשרתי עם שאר תוצרי הקונבולוציה ליצירת ווקטור הפלט של השכבה.

ממשק ה- *Web* לשימוש בתוצר הלמידה:

בנוסף לאפשרות הלמידה של הרשת יצרתי ממשק של דף *Web*, המאפשר טעינה של משקולות המודל לאחר תהליך הלמידה, וביצוע קיטלוג למשפט חדש, הניתן בדף כקלט, לאחת מהקטגוריות של מאגר המידע עליו נעשה האימון (כרגע מותאם ל- *AG's news corpus* אך בשינויים קלים בלבד יכול להיות מותאם לכל מאגר מידע אחר).

קוד הדף מבוסס על שפת *Bottle* המאפשרת ממשק *Web* נוח לשפת *Python*, שהינה גם הבסיס לשפת *Pycnn*, כך שבעת קבלת הקלט מהדף האינטרנטי, בהינתן משקולות המודל המאומנות, כל שנותר הוא לעשות קוואנטיזציה פשוטה של הקלט, להזין אותו ל- *Forward Propagation* ברשת ולהחזיר את תשובתה.

NLP CNN From Scratch Website

This web page is classifying articles to 4 categories (World, Sports, Business & Sci/Tech) according to 'Text Understanding from Scratch' article, based on a 1D NLP convolutional neural network

Link to original paper from X. Zhang & Y. LeCun: [Text Understanding from Scratch paper](#)

Please insert article text to the box below

Article:

*איור 3: תצוגת הדף לפני הזנת קלט.

11. במימוש המטריצה מספר הדוגמאות המקסימלי שניתן היה להריץ יחדיו ב- *Batch* אחד היה 3 (עבור GB 2 של זיכרון לריצה), לעומת 35 דוגמאות ב- *Batch* במימוש עם ווקטור על אותה כמות זיכרון.

12. כלומר החלפתי את הפונקציה $\max(threshold, x)$ ב- $\max(threshold, x + threshold)$.

תוצאות

תחילה אציין שעמדתי במטרת הפרויקט הראשית, שהייתה חיקוי מוצלח ועובד של המודל המוצג במאמר *Character-level Convolutional Networks for Text Classification* [1] בשפה *Pycnn*, כך שנוכל לדעת האם המוצג במאמר אינו מטעה או חסר בפרטים. כאשר התוצאה הסופית המוצגת במאמר (עבור הרשת המקבילה לזו שמימשת) היא 84.35%, ואילו התוצאה הסופית שלי היא (עדיין אין תוצאה סופית, אבל התוצאה בסוף האפוק ה-6 היא מעל 80%) חשוב לציין כי התוצאה אליה הגיעו צוות החוקרים ברשת המקורית התקבלה לאחר 30 סבבי למידה (*Epochs*)¹³, ואילו התוצאה הסופית ברשת שלי התקבלה לאחר 10 סבבים בלבד, כנראה בשל מספר שינויים שעשיתי ואותם אפרט בהמשך. צמצמתי את מספר הסבבים אותם בדקתי משתי סיבות: הראשונה, היא איטיות הריצה (קצב ממוצע של כ-1 *Epoch* ליום) הנובעת מהמחזור ב-*GPU* במחשבים עליהם עבדתי ומהמחזור באופטימיזציות של מהירות הריצה ב-*Pycnn*, והשניה היא התכנסות התוצאה לערכים יחסית קבועים, כך שלא היה נראה שעוד 20 סבבים ישנו את התוצאה עוד הרבה.

הניסויים העיקריים שביצעתי וייצרו למידה היו: שינויי גודל ה-*Batch*, ביטול איתחול המשקולות הגאוסיאני, שינויי פונקציית האקטיבציה בשכבות המלאות (ה-3 האחרונות) ללינארית (במקום *ReLU*). את שינויים אלה ביצעתי במספר שילובים אחד עם השני ובחנתי בעיקר אל מול קצב ההתכנסות ופחות אל מול תוצאה סופית של סט הבדיקה בסוף הלמידה עקב אותו שיקול זמן המצויין לעיל. חלק מהשינויים יצרו מצב בו לא הייתה למידה כלל, אך בשילוב שונה דווקא יצרו התכנסות מהירה וטובה יותר, כמו למשל ביטול איתחול המשקולות הגאוסיאני עם פונקציית האקטיבציה של *ReLU* על כל שכבות המערכת לא התכנסה ללמידה כלל, אך בשילוב שינויי פונקציית האקטיבציה בשכבות המלאות ללינארית יצרה התכנסות מהירה וטובה יותר בערך הסופי.

מבין כל השינויים הנ"ל השילוב המוצלח ביותר שבדקתי היה גודל *Batch* של 32 (לעומת 128 במקור), ללא איתחול משקולות גאוסיאניות ועם אקטיבציה לינארית בשכבות המלאות, ואת תוצאות רשת זו הצגתי קודם לכן. תוצאה דומה התקבלה עבור שינויים אלה במימוש בשתי השפות *Keras* ו-*Pycnn*.

השאלה האחרונה שהתעסקתי בה הייתה – האם מתודת הקונבולוציה הנתונה ב-*Keras* וזו שכתבתי ב-*Pycnn* שקולות? שאלה שהטרידה אותי רבות בשלב בו עוד לא הצלחתי להגיע להתכנסות הפרמטרים למצב של למידת הטקסט. לשאלה זו לא אציג תוצאות גרפיות או מספריות, מכיוון שהן חסרות עניין, ורק אציין שתחליף הקונבולוציה שהצגתי אכן זהה לפונקציית ה-*Keras Convolution1D* המשתמשת מאחורי הקלעים בפונקציית *Convolution2D* עם חלון בגודל הקרנל החד מימדי וגודל מסגרת הקלט (גובה המטריצה).

דיון

בנוסף ללמידה האישית שלי, תוצאות פרוייקט זה חיזקו את ממצאם של כותבי המאמר' הטוענים שניתן לבצע קיטלוג של טקסט בעזרת רמת ה"אות" בלבד ואין צורך במאמץ הנוסף הנדרש בכדי לקרוא את הטקסט ברמת ה"מילה". מה שנראה כמתאפשר במיוחד ברשת *CNN* המתמחה בחילוף תבניות מורכבות מאות פשוט (*Raw Signal*), ויכול להיות שאף מאפשרת אבחון דקויות שאנו מאבדים כאשר נקרא את הטקסט ברמת המילה. כוונתי היא שבשכבות הגבוהות של רשת *CNN* יכול להיות שנוצרו פילטרים ה"מחפשים" מילים ספציפיות כתבנית ספקטרלית של אותיות, באותו האופן שקו ישר בזווית כלשהי היא תבנית ספקטרלית של פיקסלים בתמונה (כשאת אותו קו ישר פילטרים לומדים לחפש בסריקה של התמונה ומאפשרים ראייה ממוחשבת ברמה גבוהה), מה שלמעשה נותן לנו פונקציונאליות דומה לזו של הזנת מילים בשיטות כמו *Word2Vec* או *n-gram* (רק ללא הצורך בעיבוד מקדים). בנוסף לאפשרויות שונות ומשונות, שלא נוכל לקבל כאשר נסתכל רק על מילה כמבנה שלם, כשלדוגמא אולי הסיומת *_tion* מספר מסויים של פעמים בתוך טקסט מרמז שהוא שייך לטקסט המדבר על עסקים, או התחילית *re_* מרמזת על טקסט המדבר על נושאים טכנולוגיים, שהן שתיהן דקויות שלא היינו רואים אם המילה הייתה מאופיינת כווקטור שמנסה לתפוס את משמעותה הסמנטית של המילה ולא כרצף אותיות שיכול להכיל גם משמעות סינטקטית כלשהי.

לעומת זאת, חלק ממסקנות הפרויקט מנוגדות לכתוב במאמר, הדוגמה המובהקת ביותר לכך היא שכותבי המאמר טוענים כי שיטה זו דורשת מספר רב מאוד של דוגמאות בכדי לייצר למידה אפקטיבית, וכי 30,000 דוגמאות מכל קטגוריה המוצגות במאגר ה"ל", הן כמות לא מספקת (במאמר הראו מצב של *Over Training* על אותו סט אימון ובדיקה בו אני השתמשתי). כאשר

13. במאמר הם פירטו כי ביצעו 30 סבבים, כך שלאחר כל 10 סבבים חילקו את קבוע הלמידה ב-2.

בתוצאות פרוייקט זה ניתן לראות איך בשינויים לא גדולים במיוחד ניתן לשפר את קצב ההתכנסות ואת איכות הלמידה ללא תוספת של דוגמאות נוספות.

מסקנה נוספת, שלא נבעה מפרוייקט זה, אך אני רואה בה ערך רב לגבי יעילות השיטה, היא שהמודל איננו מוגבל רק לשפה אחת כמו מודלים המתבססים על רמת המילה, המתבססים על הקשר בין המילים השונות ומבנה השפה. החוקרים כותבי המאמר הראו שבעזרת תהליך פשוט של לטיניזציה (המרת מילה לאותיות לטיניות ע"פ הצליל שלה כך: "מילה" <="mila"), ניתן לבצע למידה של טקסטים בכל שפה ללא שינויי כלל לרשת הקיימת.

לסיכומי של דבר, עיבוד שפה טבעית ברמת האות ובעזרת רשתות נוירונים עמוקות הינו מעניין ביותר ויכול לספק פתרונות למכשולים רבים בעולם ההבנה הממוחשבת של טקסט. יהיה מעניין לבדוק את החלפת שכבות הקונבולוציה בשכבות *Bi-LSTM*, אך להישאר ברמת עיבוד בסיסית של אותיות ולא מילים, בנוסף לאפשרויות שיפור נוספות של רשת ה-*CNN*.

ובנימה יותר אישית, אני חושב שתחום עיבוד השפה הטבעית בכלל והאפליקטיביות של תחום רשתות הנוירונים העמוקות בפרט הוא תחום מרתק עבורי, וללא ספק הפקתי רבות מללמוד אותו ומהתמקדות בו בפרוייקט הנ"ל. אין לי ספק שאמשיך להעמיק בתחום זה בהמשך לימודי לתארים מתקדמים.

ביבליוגרפיה

1. X. Zhang, Y. LeCun. *Text Understanding from Scratch*. 4 April 2016.
2. T. Joachims. *Text categorization with suport vector machines: Learning with many relevant features*. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142. Springer-Verlag, 1998.
3. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. *Gradient-based learning applied to document recognition*. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.
4. J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. *DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia*. *Semantic Web Journal*, 2014.
5. C. dos Santos and M. Gatti. *Deep convolutional neural networks for sentiment analysis of short texts*. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics.
6. R. Johnson and T. Zhang. *Effective use of word order for text categorization with convolutional neural networks*. *CoRR*, abs/1412.1058, 2014.
7. Y. Kim. *Convolutional neural networks for sentence classification*. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.