Final project report

question: Why should we add the momentum to SGD? What are the main two improvements Adam has compared to the simple SGD? What was the best optimization algorithm and best learning rate you've found?

answer: there are several problems, first batches come with inherent noise so the gradient direction after each batch might not be optimal in our true minimum direction. gaining momentum of several batches of low but common steady gradient direction might suggest that the minimum we are looking for is trending in a specific direction, making the momentum impact across batches and overcome this noise.
second sgd alone might not get over local minima points, adding a momentum could reveal another minima hiding down the road.

regarding Adam optimization advantages. It includes a mechanism called bias correction. It simply means that the first few steps will reduce the momentum effect. logically, since it has no true average collected momentum representation backed with enough data and examples, it might have the wrong effect we want to get out of adding the momentum feature.
second another feature favors Adam optimization is the RMSProp
Basically what it is useful for, is to normalize the descent direction making the zigzag motion caused by inconsistent strong gradient of specific examples during the training process to have a more moderate effect on the distance we make towards this gradient.

the best optimization and lr we got is Adam using lr of 0.0005

question: What is the purpose of batch normalization (why do we use it)? Did the Batch normalization improve the network preformance? Summeraize this section results in your report.

answer: Batch normalization basically means that we normalize the activation map output between the current layer to the next one.
this helps in several aspects.
first, it functions as additional regulation preventing overfit.
second it allows us to use bigger learning rates. Using batch norm we can prevent our activation function from putting our input values into the max/minimum values of our activation function. since we change weights based on the activations derivatives working with max/min values of activations would cause tiny/zero derivatives
which are tiny steps towards convergence.

i used batch norm from the beginning of the project while building the first net architecture. but i believe it did

question: Why should we use regularization? How does the regularization affect the train accuracy and loss? How does it affect the val and test accuracy and loss? What was the best regulrization method? Summarize this section results in your report.

answer: we should use regularization techniques to avoid overfitting our training data set.. though regularization might decrease the training accuracy and increase its loss values, it helps our model at solving a more generic problem (less specific examples oriented results) which causes the test and validation accuracy to increase and decrease their loss.
in our reports we can see that the dropouts got better results in test batch
even though the training accuracy for L2 and Dropouts were almost identical.

question: What is your best architecture? what its best accuracy and loss on the test set? Compare your result to assignment 2. How did you mange to improve the model? Summarize this section results in your report.

answer: the best architecture was using 3 blocks of
conv layer -> reLu activation -> batch Norm
Max Pooling and Dropouts for each block
into a 2 layer of FC net using Adam optimization with 0.001 learning rate.
The best accuracy we got was 85% and loss of 0.56 on the test dataset.

project summary

We tried to fine tune several models using regularization tools like dropout, L2, and batch normalization and testing several optimization methods
We even tried to use a resnet50 pretrained model. once with all layers frozen and once with the last layers unfrozen.. for some reason we could not achieve as high results as of our own net.
The best architecture for us was using 3 blocks of
conv layer -> reLu activation -> batch Norm
Max Pooling and Dropouts for each block
into a 2 layer of FC net using Adam optimization with 0.001 learning rate.
The best accuracy we got was 85% and loss of 0.56 on the test dataset.

blogs summary on following pages

טרנספורמרים חזותיים

בשנים האחרונות תחום ה-AI התפתח בצורה משמעותית: עד כה היו מודלים בתחום הוויז'ן שהיו טובים אך התמקדו בדברים ספציפים, והיום מודלים עם מטרות כלליות החלו להתפתח. סוג אחד של מודלים כאלו הוא טרנספורמרים. הטרנספורמרים הגיעו לביצועים מעולים והפכו לסטנדרט בתחום ה-NLP בעקבות יכולת ההכלה והפישוט שלהם (על ידי הוספת ראש MLP מתאים).

השימוש בטרנספורמרים החל בתחום הטקסט שבו הם עיבדו והבינו את הטקסט, לאחר מכן השימוש התרחב גם לתחום הוויז'ן (שם הם נקראים בקיצור ViT) שבו העיבוד הורחב גם לתמונות על ידי שימוש בפונקציות שונות לטוקנזיישן ולהטמעה, אך הארכיטקטורה הבסיסית והקידוד נשארו אותו הדבר.

המקודד של טרנספורמרי הוויז'ן מורכב מבלוקים כאשר כל אחד מהם מורכב משלושה רכיבי עיבוד עיקריים:

1. שכבה לינארית – עוזרת למודל להתאים את עצמו לכל מיני קלטי תמונות מגוונים
2. MSP (Multi-head Attention Network) – אחראית ליצירת מפות אטנשיין המסייעות לרשת להתמקד באיזורים החשובים ביותר בתמונה.
3. MLP – שתי שכבות קלאסיפיקציה עם GELU בסוף. הבלוק האחרון של MLP (הנקרא MLP head) משמש כפלט של הטרנספורמר. הפעלת softmax על פלט זה, יוצרת לייבלי סיווג.

בעקבות העובדה שהViTs משמשים למטרות כלליות, יש להם שימושים רבים בתחום הוויז'ן:

1. סיווג תמונות – הביצועים של ה-ViTs נמוכים משל ה-CNN כאשר הדאטה סט הוא קטן, אך כאשר הדאטה סט גדול, הביצועים של ה-ViTs יותר טובים. זה קורה בעקבות העובדה שהCNN- מקודד בצורה יותר יעילה את המידע המקומי בתמונות.
2. התאמת כיתוב לתמונה – במקום להתאים לתמונה מילה אחת שמתארת אותה, ViTs יכולים להתאים לה משפט המתאר אותה. ViTsלומדים ייצוג כללי לאפיון תמונה במקום ייצוג של לייבלים פשוטים.
3. CLIP (Contrastive Language-Image Pre-Training) – מקשר בין תמונה לטקסט. יישום זה דורש ייצוג מופשט של טקסט וגם של תמונה ולכן הכלליות של המודל מאוד עוזרת כאן. זה יכול להתבצע על ידי אימון נפרד של שני מקודדים של טרנספורמר, האחד לטקסט והשני לתמונה והשוואת הדימיון בין התמונה המקודדת לפיצ'רים שהובאו מתוך הטקסט. סוג זה של למידה משמש גם כ-transfer learning.

לסיכום, לטרנספורמרים יש יישומים רבים בתחום ה-AI, והם מגיעים לביצועים מעולים.

ברט: המודל המוביל לעיבוד שפה טבעית

חוקרים בתחום ה-AI בגוגל, פרסמו מאמר בנוגע לברט ששינה את פני המחקר ב-ML. במאמר הוצגו תוצאות מעולות במשימות מגוונות בNLP- (כמו SQuAD, MNLI ועוד). עיקר החידוש הטכנולוגי של ברט הוא לקיחת תכונה של למידה דו כיוונית של הטרנספורמר, שהיא לקרוא את הטקסט כולו בבת אחת, וכך המודל לומד את ההקשר של המילה בהתאם לכל מה שסביבה (גם מימין וגם משמאל). את התכונה הזו הם יישמו במידול שפה. דבר זה היה בלתי אפשרי לפני כן ובניגוד למה שבוצע עד עכשיו, כאשר ניסו להסתכל על טקסט באופן חד כיווני. הטכניקה שבה השתמשו כדי לעשות זאת נקראת MLM. תוצאות המחקר הראו שמודל שפה שאימנו אותו בצורה דו כיוונית, מבין יותר את תוכן ואת תחביר השפה לעומק מאשר מודל שפה שאימנו אותו בצורה חד כיוונית.

ברט משתמש בטרנספורמרים, לאותם הטרנספורמרים יש שני מנגנונים: הראשון הוא המקודד שקורא את קלט הטקסט, והשני הוא המפענח שמפיק פרדיקציה למשימה. מכיוון שהמטרה של ברט הינה ליצור מודל שפה, אז רק המקודד נחוץ.

כאשר מאמנים מודלי שפה, לרוב משתמשים במשימה של "חיזוי המילה הבאה". משימה זו היא חד כיוונית ולא דו כיוונית, ובכך היא מגבילה את היכולת ללמוד על ההקשר של מילה. כדי להתגבר על הבעיה הזו, ברט משתמש יחד בשתי טכניקות של אימון, במטרה לקבל loss נמוך המשותף לשתיהן:

1. Masked LM (MLM) – בכל רצף של מילים כלשהו, לוקחים 15% מהמילים ושמים במקומן מסיכת טוקן (בפועל לא כל 15% מהמילים מוחלפות במסיכת טוקן, אלא רק 80% מתוך ה- 15%. 10% מתוכן מוחלפות במילה ראנדומלית, ו-10% המילים האחרות נשארות אותו הדבר). המודל צריך לחזות את ערכן של המילים המקוריות, בהקשר של המילים האחרות באותו הרצף, שלא הוחלפו במסיכת טוקן. דבר זה דורש הוספה של שכבת קלאסיפיקציה על הפלט של המקודד, הכפלתה במטריצת הembedding, המרת המטריצה לגודל של המילון והפעלת softmax. ברגע שמפעילים את פונקציית ההפסד, היא מתחשבת בפרדיקציות של המילים שהוחלפו במסיכת טוקן ומתעלמת מהפרדיקציות של שאר המילים ברצף. דבר זה גורם למודל להתכנס לאט יותר מאשר מודלים הלומדים באופן חד כיווני, אך חיסרון זה מתקזז עם העובדה שאותו מודל בסופו של דבר יבין את ההקשר של המילים בצורה טובה יותר.

1. חיזוי המשפט הבא (NSP) – המודל מקבל זוגות של משפטים, ומנסה לחזות אם המשפט השני מבין השניים בא ישר אחרי המשפט הראשון בטקסט המלא. במהלך האימון, 50 אחוזים מהקלט הינם זוגות של משפטים שאכן עוקבים בטקסט, ו-50 האחוזים הנוספים הינם זוגות ראנדומלים של משפטים מהטקסט, ההנחה היא שהזוגות הראנדומלים האלו לא יהיו קשורים אחד לשני בשום אופן. על מנת שלמודל יהיה יותר קל באימון, מעבדים את הקלט לפני כניסתו למודל. עושים זאת על ידי הוספת טוקנים לכל משפט, המכילים מידע אודות מיקום המשפט בטקסט והאם הוא המשפט הראשון או השני מבין שני המשפטים שנלקחו יחד כזוג.

לסיום, גם כאן ניתן לראות כי גודל המודל הוא חשוב. ככל שיש לברט יותר ויותר משקולות, כך הביצועים שלו טובים יותר. בנוסף, עם כמות מספקת של דאטה בסט האימון - ככל שמתבצעים יותר שלבים במהלך האימון, כך רמת הדיוק עולה. ניתן להשתמש בברט למשימות שפה מגוונות רבות על ידי הוספה לבסיסו של המודל שכבה מתאימה, ורוב ההייפר-פרמטרים נשארים אותו הדבר בין משימות שונות.