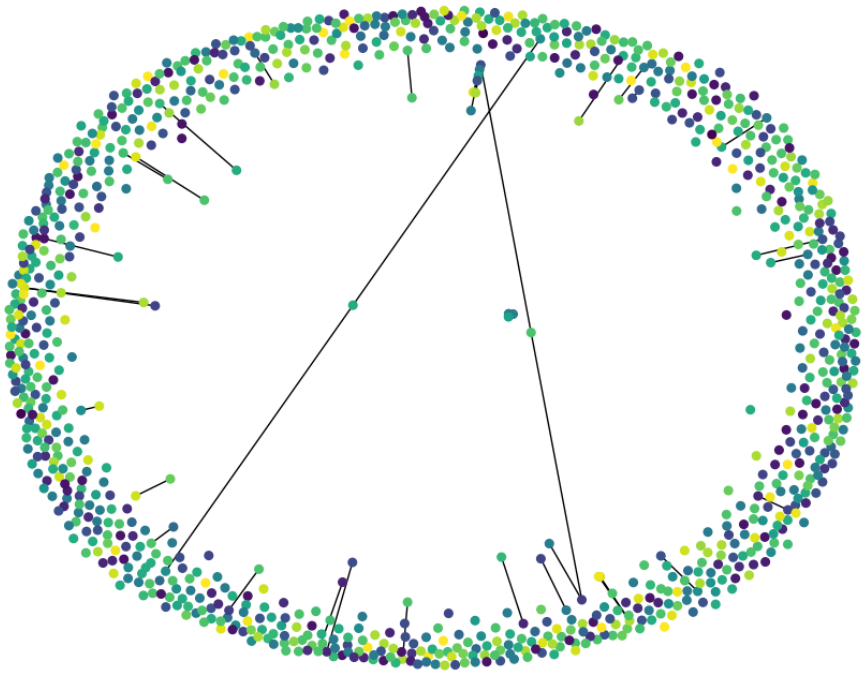
דו״ח מעבדה 3

גיא חדד 316508126
נעם שמיר 316299098

תיאור המשימה: קלסיפיקציה של צמתים בגרף המייצגים מאמרים אקדמיים ל40 קטגוריות.
מבנה הגרף: גרף מכוון בו כל קשת מייצגת ציטוט של מאמר אחר.
ייצוג הקודקודים: מיצוע של ייצוג המילים במאמר שנוצר על ידי skip gram.

תהליך:
ראשית, הורדנו את הנתונים ולמדנו להתחבר לממשק של pytorch.geometric.
הבנו את המאפיינים הכלליים של הגרף ושל הצמתים
ניסינו למשל להציג את הגרף והקשרים בעזרת networkx ע״י דגימה של 1000 קודקודים וקיבלנו את התמונה הבאה:



ניכר שדגימה אקראית של 1% מהקודקודים בגרף באופן טבעי לא תופסת קשתות רבות.
הדרגה הממוצעת היא 4.4, וקיימים גם קודקודים בדרגה 0, כלומר מבודדים (מאמר שלא צוטט ולא ציטט מתוך המאמרים שבדאטהסט).
הגרף מכוון וללא קשתות עצמיות (אין מאמר שמצטט את עצמו).

בתור התחלה יצרנו רשת נוירונים פשוטה שמבצעת משימת קלסיפיקציה עבור כל צומת בהתבסס על הייצוג שלו, כלומר וקטור בגודל 128 שמייצג את הטקסט של המאמר.
זה היה מודל ראשוני כדי ללמוד לעבוד עם המבנה של pyroch.peometric.

לאחר מכן עדכנו את המודל כך שבמקום שכבות לינאריות, הוא יפעיל GCNconv.

הרעיון של קונבלוציות גרף הוא עדכון הייצוג הפנימי של כל קודקוד בהסתמך על הקודוקדים הסמוכים אליו. כאשר מבצעים זאת באופן איטרטיבי למשך מספר חזרות - הייצוג של קודקודים מרוחקים יכול "לזרום" לעבר כל קודקוד. כמובן שככל שקודקודים קרובים יותר הם משפיעים יותר זה על זה לעומת קודקודים מרוחקים.

מבנה הרשת:

קונבולוציה (מגודל מימד הכניסה לגודל hidden_channel)
Relu
Drop out(0.2)
קונבוצולוציה (ללא שינוי המימד)
Relu
Drop out(0.2)
קונבולוציה (מגודל hidden_channel לגודל 40 כניסות)

השתמשנו באופטימייזר Adam עם:
learning_rate=0.001, weight_decay=5e-4

ערכנו מספר נסיונות עם ערכי dropuot שונים וראינו שהגדלה של הdropout מעבר ל 0.2 רק פוגעת בביצועים.
ערכנו נסיונות של החלפת פונקציית אקטיבציה ל tanh.
תוצאות הaccuracy שקיבלנו נעו סביב 57-58%

בשלב שני - רצינו להוסיף לייצוג של כל קודקוד מידע נוסף.
אמנם אין לנו מידע נוסף על התוכן של המאמר, אך כן ניתן להעשיר את הייצוג על ידי תכונות של הקודקוד מתורת הגרפים.
הוספנו את המדדים הבאים:

Degrees:
מספר השכנים של הקודקוד
In_degrees:
מספר הקשתות הנכנסות
Out_degrees:
מספר הקשתות היוצאות
Clustering:
מודד עד כמה השכנים של קודקוד מחוברים זה לזה.
Eigenvector_centrality:
מדד לחשיבות (השפעה) של קודקוד. לוקח בחשבון את הדרגה של הקודקוד ואת הדרגות של שכניו.
Pagerank:
מדד לחשיבות של קודקוד כפי שנראה עבור הילוך אקראי על גרף. האלגוריתם פותח ע"י גוגל לדירוג דפי אינטרנט.
Degree_centrality:
מדד למרכזיותו של קודקוד שמתבסס על הדרגה שלו ביחס ליתר הדרגות הגרף.
Katz_centrality:
מדד מרכזיות שלוקח בחשבון את הקשר לכל הקודקודים. קרבה לקודקודים קרובים מעניקה ציון גבוה, והמשקל של הציון דועך אקספוננציאלית ככל שמתרחקים.

בנוסף לכל אלו - הוספנו גם את **שנת כתיבת המאמר**.
את כל הערכים החדשים נירמלנו ע״י min-max scaling כדי לא לייצר פי״צרים עם ערכים מאוד גבוהים.

לאחר מכן - ניסינו לשפר את הביצועים של הרשת ע״י הוספת residual block. זהו רעיון שלא ראינו שביצעו אותו במשימה מהסוג הזה, ואת ההשראה לקחנו מ ResNet, שם הוצג הרעיון לראשונה.
העקרון של resBlock הוא לייצר מעקף שפותר את הבעיה האפשרית ששכבה של הרשת לא משפרת את הביצועים, ועדיף להמשיך עם המידע שהיה לפניה. מה שעשינו היה לשמור את התוצאות לפני הפעלת שכבת GNNconv ואז לסכום עם הפלט של השכבה.
בחרנו להוסיף שכבה אחת כזו ולבדוק את הביצועים שלה.
מיד לאחריה הוספנו batch normalization כדי לשמור על יציבות.

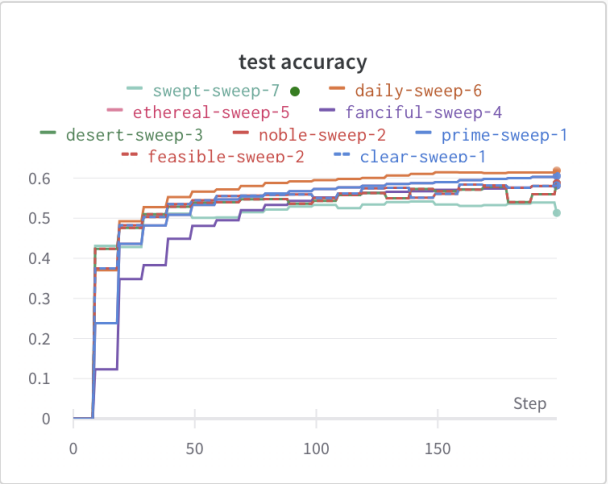
מבנה הרשת אם כן היה:

קונבולוציה (מגודל מימד הכניסה לגודל hidden_channel)
Relu
resBlock
קונבוצולוציה (ללא שינוי המימד)
Batch normalization
Relu
קונבולוציה (מגודל hidden_channel לגודל 40 כניסות)
Relu

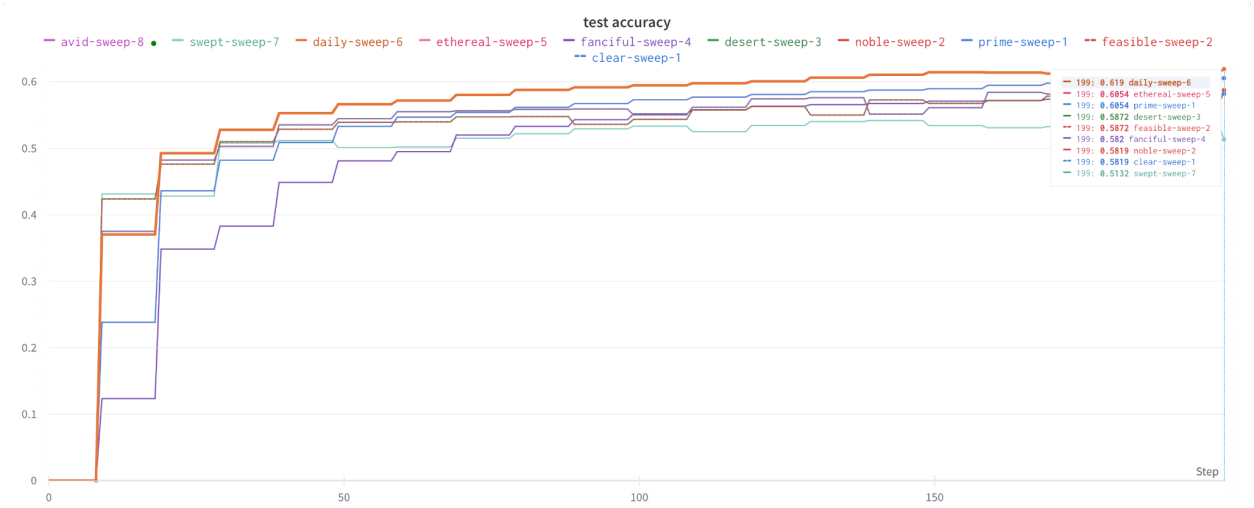
בשלב באחרון - כדי לבחור hyperparameters - השתמשנו ב weight & Bias (wandb).
נתנו לcontroller קונפיגורציה של ערכי learning rate שונים (0.001, 0.05, 0.01) וכן גדלים שונים של hidden_channels מהטווח (128, 256, 512, 1024).

ב wandb ניתן להריץ רצף של קונפיגורציות שונות על פי מתודות שונות (למשל grid serach או דגימת פרמטר מתוך התפלגות מוגדרת). בחרנו בשיטת random, כלומר הקצאת קומבינציות שונות באופן אקראי של

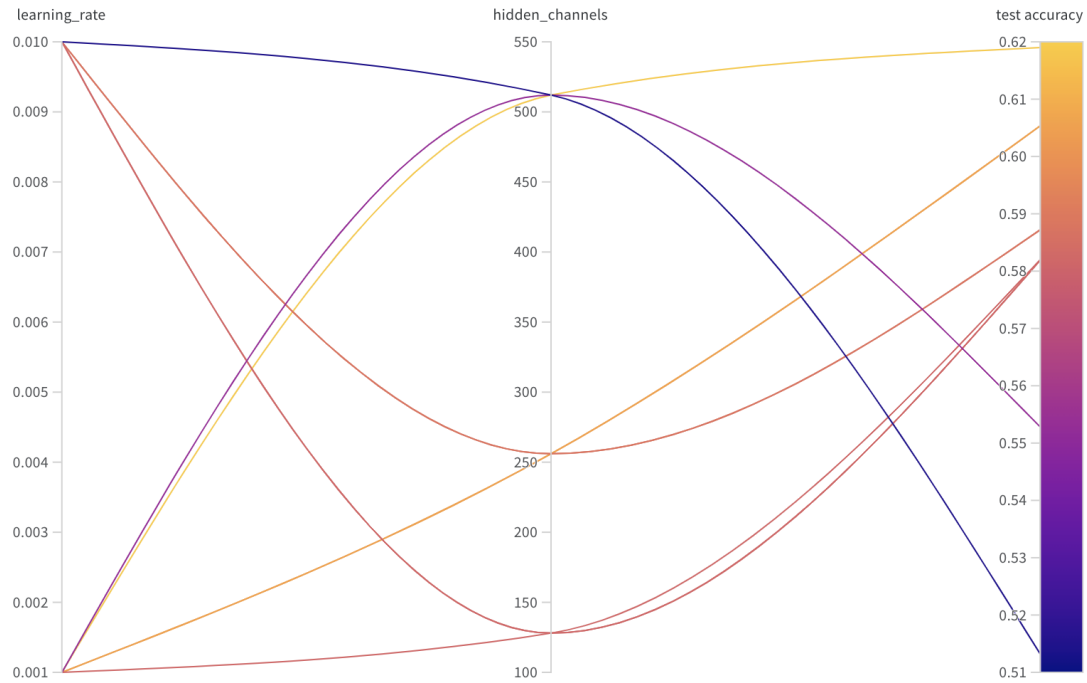
הרצנו למשך 200 אפוקים את תהליך hyper parameter tuning וקיבלנו את התוצאות הבאות עבור train_loss, test_accuracy.



התוצאה הטובה ביותר התקבלה עבור learning rate = 0.001 ועבור hidden_channels=512
ניתן לראות את הביצועים של כל קונפיגורציה על הטסט, והגרסה הטובה ביותר השיגה accuracy 61.9% לאחר 200 אפוקים.



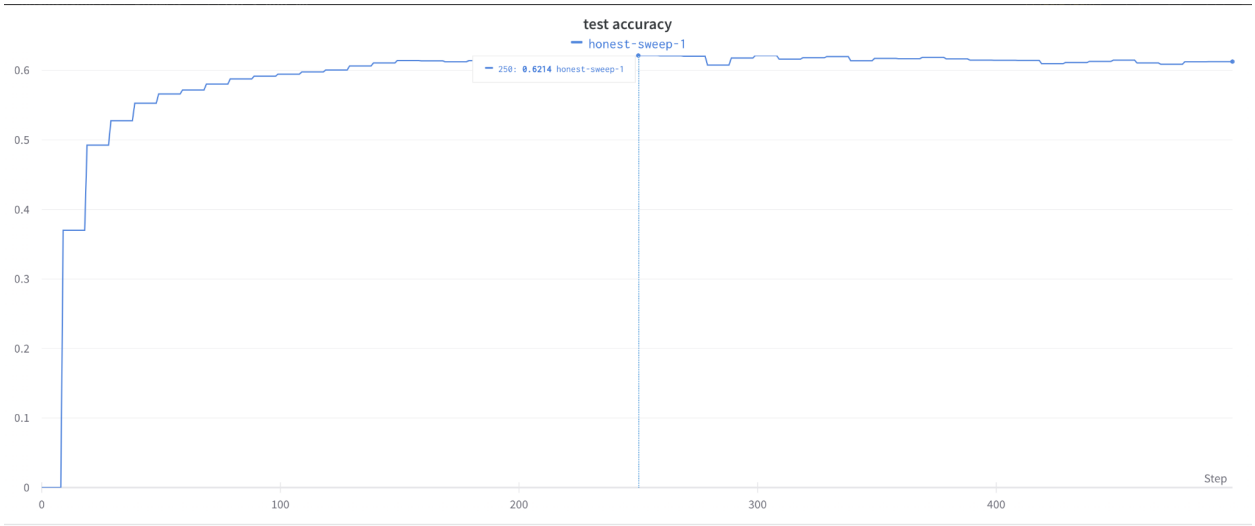
ניתן לראות את הביצועים של ההיפרפרמטרים השונים בכל אחד מהניסויים:



ניתן לפרש את הגרף המקבילי לעיל על פי הצבעים שבו. כל עמודה מייצגת פרמטר וכל קו מייצג ניסוי. הצבעים מתארים את הביצועים על הטסט (צהוב=ביצועים טובים, סגול=ביצועים נמוכים).
לא ניתן להצביע על ערך מיטבי של lr או של hidden_channels (כלומר- אין ערך שעבורו עוברים בעיקר קוים צהובים ומעט סגולים). רק הקומבינציה הנכונה של ההיפרפרמטרים משיגה את התוצאות הטובות.

על פי התוצאות הללו בחרנו את ההיפרפרמטרים:
{hidden_channels': 512, 'learning_rate': 0.001'}

לאחר שבחרנו את ההיפרפרמטרים ועל פיהם הגדרנו באופן מלא את המודל - רצינו לבחון את המספר האפוקים הרצוי.
הרצנו 500 אפוקים והתקבלה התוצאה הבאה:



הביצועים עם סט המבחן השיגו את הaccuracy המקסימלי **(62.1%)** עבור 250 אפוקים, ולאחר מכן ירידה קטנה מאוד והתייצבות. בסה״כ ניתן לראות שלאחר 100 אפוקים הערך לא יורד מתחת ל 59% accuracy, והוא דיי יציב, ולכן כל מספר אפוקים בטווח הזה יהיה כנראה בסדר.

לסיכום: הצגנו קונבולציה על גרף, עם ResBlock, ואופטימיזציה מנוהלת של הפרמטרים.

קרדיט:
המודל התבסס על המחברת של pytorch.geometric שבה הם מציגים דוגמה של קלסיפיקצית קודקודים.

קישור לגיט:
https://github.com/Noam-Shamir-1/lab_03_GNN.git