**פרויקט מחקר – ארכיטקטורות מחשבים מתקדמות**

**מגישים:** בר דעבול – 324079243, רועי מיכאל –

**הקדמה ומוטיבציה**בפרויקט שלנו נתכנן אלגוריתם וארכיטקטורה למסווג פתוגנים בעזרת שימוש בלמידה עמוקה.  
בפרויקט זה נקדם את הידע שלנו בתחום של ארכיטקטורות מחשבים מתקדמות. נלמד לעומק על המבנה של הגנים, וכן על כלים בלמידה עמוקה ונשתמש בכלים האלה כדי לתכנן אלגוריתם וארכיטקטורה טובים לסיווג גנים.אנחנו מבצעים את פרויקט המחקר במטרה לאפשר סיווג ואבחון מצילי חיים בזמן אמת של פתוגנים, תוך ניסיון להסיר מחסומים חישוביים.לנתונים הגנומיים יש השפעה על תחומים רבים וחשובים כמו רפואה, גנטיקה וביוטכנולוגיה. בפרט, בתחום הרפואה, זיהוי פתוגנים חשוב מאוד לקבלת החלטות מצילות חיים.  
לדוגמה, בתקופות שבהן העולם נפגע ממגיפה ויראלית (כמו הקורונה), יכולים להיווצר וריאנטים שונים של המגיפה. זיהוי האנשים שנושאים וריאנטים חדשים ומסוכנים חשוב כדי למנוע את ההתפשטות שלהם. כמו כן, לאחר ניתוחים שונים, מטופלים יכולים לפתח זיהום בדם שעלול לגרום למוות. הגילוי המוקדם של הפתוגנים שגורמים לזיהום הוא חיוני מאוד ויכול להציל את חייהם.  
קיימים מספר אתגרים בניתוח וסיווג נתונים גנומיים: מערך הנתונים גדול מאוד, הדפוסים מורכבים ויש צורך בחיזויים מדויקים. בנוסף, זיהוי פתוגנים היא בעיה קריטית בזמן ובקנה מידה גדול, ושיטות הסיווג הקיימות איטיות מאוד.בפרויקט זה החלטנו לפתח רשת עצבית (Neural network) מסווגת פתוגנים בזמן אמת באמצעות CNN, ונבדוק את טיב הרשת תוך התייחסות למספר מדדי הערכה.

**השיטה והעיצוב של המודל**בתחילה, ניסינו לעבוד עם מודל פשוט שיש בו רק מספר שכבות של fully connected layer. בהמשך, שיפרנו את המודל ע"י הוספת שכבת קונבולוציה, אחריה פונקציית הפעלה (ReLU) ולאחר מכן שכבת pooling. בסופו של דבר, לאחר בחינה של מספר מודלים עם מספר שכבות הגענו למודל הסופי. המודל הסופי (מופיע בקובץ ComplexCNN.py) מורכב ממספר שכבות שונות:  
ראשית, יש חמש שכבות שבכל אחת מהן אנחנו מבצעים – קונבולוציה חד מימדית, אחריה נורמליזציה (batch normalization), פונקציית הפעלה של ReLU (משמשת להבנת אי-לינאריות) ולבסוף max pooling (משמש לצמצום המימדים המרחביים של מפות התכונות). מטרת השכבות הללו היא ללכוד רמות שונות של תבניות ותכונות שקיימות בקלט.לאחר שכבות הקונבולוציה, הפלט משוטח לווקטור חד מימדי ומכניסים אותו לשתי שכבות של fully connected (fc1 ו-fc2). השכבות האלה מעבדות את התכונות שחולצו ובסופו של דבר מתקבל מהם פלט של ההסתברויות של הסיווג.  
כמו כן, בין השכבה הראשונה לשכבה השנייה של ה-fully connected, יש שכבה של dropout. השכבה הזאת משמשת למניעת התאמת יתר (overfitting) ומשפרת את ההכללה של המודל.

בפונקציית ה-forward של המודל מפעילים את השכבות לפי הסדר, כך שהפלט של שכבה מסוימת נכנס כקלט לשכבה הבאה (בתחילה עוברים על חמשת שכבות הקונבולוציה, בהמשך עוברים לשכבת ה-fully connected הראשונה, לאחר מכן עוברים ל-dropout, ובסופו של דבר עוברים לשכבת ה-fully connected השנייה ומקבלים במוצא את הפלט שמוחזר מהמודל ומהווה את התוצאה – טנזור עם מימד של מספר המחלקות (הסוגים של הפתוגנים) שמכיל את הסתברויות הסיווג – סוג הפתוגן שמקבל את ההסתברות הכי גבוהה הוא החיזוי של המודל). בנוסף, בהפעלה של חמשת השכבות הראשונות יש שימוש ב-checkpoints, שמאפשרים לבצע את הפעולות עם הפחתת השימוש בזיכרון.  
כאשר מפעילים את המודל נותנים לו כקלט את גודל האינפוט, את מספר המחלקות וערך שנקרא hidden\_size (הוא מהווה היפרפרמטר ששולט במספר הערוצים בשכבות הקונבולוציה).

**מתודולוגיית הערכה והפעלת המודל**ראשית, כתבנו קוד בקובץ ProduceData.py שבו הורדנו דאטה של פתוגנים של קורונה (קבצי fasta) וכן יצרנו קובץ בשם dataset.csv שבו יצרנו רשימה של נתונים של כל הגנים שהורדנו. **עבור הדאטה בחרנו באקראי 5,000 קבצי נתונים של פתוגנים השייכים ל-10 מחלקות שונות כקלט.**  
לאחר מכן, יצרנו את הקובץ preprocessing.py שבו אנחנו קוראים את קבצי ה-fasta ואת קובץ ה-csv ואנחנו מבצעים להם תהליך של preprocess (אנחנו מחלצים את הדאטה, וממירים אותו לייצוג בינארי (יוצרים לו קידוד) בהתאם לערך שהוא מקבל A/C/G/T, אם מופיעה אות אחרת אנחנו מחליטים לאיזה ערך לשייך אותה בהתאם למשמעות שלה). בסופו של דבר, לכל קובץ fasta אנחנו מייצרים קובץ טקסט עם שם תואם שיש לו ערכים בינאריים שמייצגים את הערכים המקוריים.  
האימון, ההפעלה וההערכה של המודל מתבצעים ב-modelTraining.py (הקובץ האחרון שמריצים):  
בתחילה, אנחנו מפעילים את המתודה load\_data. במתודה הזאת אנחנו קוראים את הדאטה המקודד ואת קובץ ה-csv, אנחנו דואגים שהגדלים יהיו זהים ל-target\_length (אם יש צורך מתבצע חיתוך של הדאטה או ריפוד באפסים) ואחרי כן אנחנו יוצרים קובץ בשם converted\_data.csv שהוא הקובץ עם הדאטה הסופי שאיתו נאמן את המודל.  
בהמשך, אנחנו מחלצים מהדאטה הסופי את הרצפים המקודדים ואת הסוגים שלהם, ואנחנו מחלקים אותם ל-train ול-test (80% מהדאטה לאימון ו-20% לטסט) כאשר x מייצג את הרצפים הגנומיים (הקלט) ו-y מייצג את הסוג של הגן (הפלט). אנחנו מסדרים את הנתונים כטנסורים, קובעים את הערך של hidden\_size ומגדירים את המודל בעזרת קריאה ל-ComplexCNN.  
כמו כן, מגדירים פונקציית Loss מהסוג של Cross Entropy, שמשמשת בדרך כלל למשימות סיווג מרובות מחלקות. משתמשים גם באופטימיזציה של Adam עם learning rate של 0.001 ו- weight decay של , וכן במתזמן קצב למידה (scheduler) מהסוג StepLR (מקטין את קצב הלמידה בפקטור של 0.1 כל 10 epochs). כל אלה מגדירים את המסווג עם המספר הנבחר של epochs.  
מפעילים את המודל על סט הנתונים של האימון, ואחרי כן מפעילים את המתודה evaluate על סט הנתונים של הטסט. המתודה הזאת מפעילה את המודל המאומן על הטסט, בודקת את התוצאות ומחזירה בהתאם מדדי הערכה שונים כמו top1\_accuracy ו-top5\_accuracy.  
בסופו של דבר, מפעילים את המתודה visualize\_results שמציגה תוצאות ותרשימים שונים (כמו: precision, recall, confusion matrix) שמתארים את טיב המודל (לפי השוואה בין תחזיות המודל למחלקות האמיתיות של נתוני הבדיקה).

**תוצאות**לאחר הרצת הקוד, אימנו את המודל, הרצנו אותו על סט הנתונים של הטסט וקיבלנו את התוצאות.  
\*\*\*נצטרך להציג ולהסביר פה את המדדים הבאים:   
**אקיורסי, טופ 5 אקיורסי,** כדאי גם לפחות חלק מאלה: ביצועים/יעילות אנרגטית/שטח ועלות (אולי מבחינת זמן ריצה וזיכרון), error tolerance

**מסקנות**המודל שיצרנו מתבסס על רשת מורכבת מהסוג של CNN שמורכבת משכבות של קונבולוציה, נורמליזציה, פונקציית הפעלה, pooling, fully connected ו-dropout. ניתן לראות שהמודל מספק תוצאות טובות יחסית. ה-accuracy הממוצע שהתקבל הוא ... וה-top 5 accuracy הוא... וכו...תוצאות נוספות, אולי תרשים וכו...  
בעתיד נוכל להמשיך את המחקר האקדמי ולנסות לשלב את האלגוריתם של המודל עם מאיץ חומרה ולחקור את התוצאות והביצועים שיתקבלו.