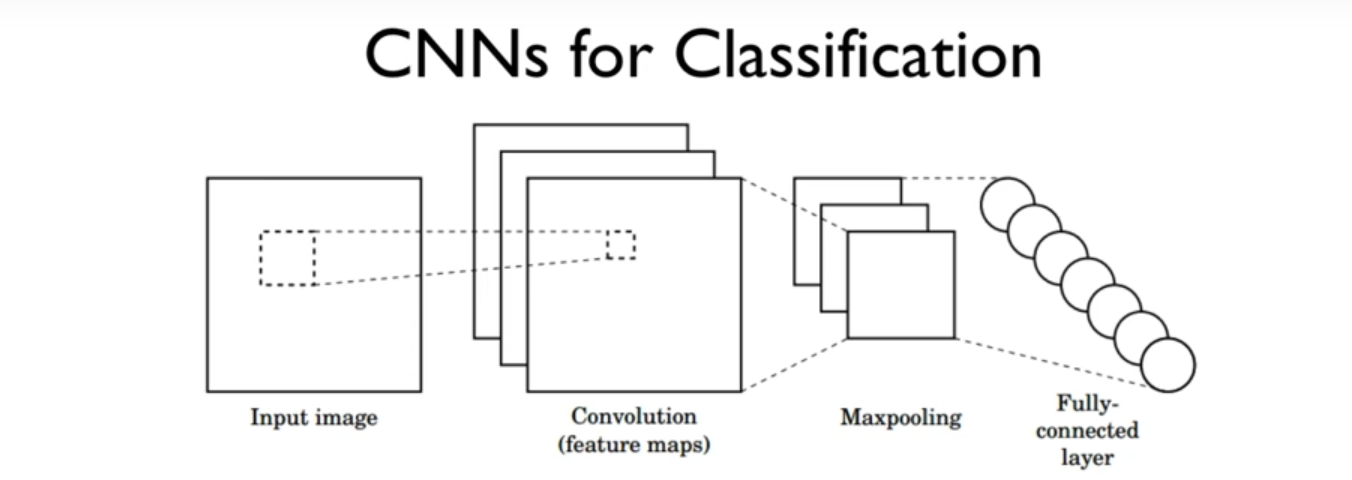
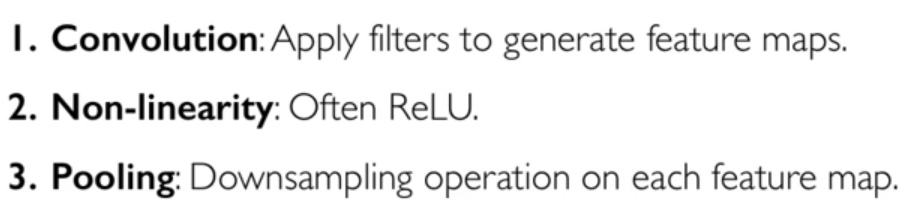
תכנית עבודה - software

******

******

*רקע-*

*כחלק מנסיון לבנות את מאיץ החומרה, החלטנו לנהל את יחידות העיבוד בעזרת קטע תוכנה. הסיבות לכך הן:*

1. *יעילות – ניהול יחידות העיבוד בעזרת בקר חומרה יעלה את רמת הסיבוכיות של המאיץ.*
2. *גמישות – יכולת לשנות את אופן זרימת המידע והפעלת היחידות.*

*החלטנו על מבנה התוכנה הבא:*

#define ADDRX /\*BASE POINTER OF THE ADDRESS WHERE THE IMAGE WILL BE LOADED \*/

#define ADDRY /\*BASE POINTER OF THE ADDRESS WHERE THE WEIGHTS WILL BE STORED\*/

#define ADDR\_CONV\_RES /\*BASE POINTER OF THE ADDRESS WHERE THE CONVOLUTION LAYER RESULT WILL BE STORED\*/

#define ADDR\_ACTIVATION\_RES /\*BASE POINTER OF THE ADDRESS WHERE THE ACTIVATION LAYER RESULT WILL BE STORED\*/

#define ADDR\_PULL\_RES /\*BASE POINTER OF THE ADDRESS WHERE THE MAX PULL LAYER RESULT WILL BE STORED\*/

#define RESULT /\*\*/

define COUNTER /\*SAVE THE NUMBER OF IMAGE \*/

void MANNIX\_NN(image\_ptr\*, weights\*, bias\*) {

    load(image\_ptr, ADDRX); load(weights, ADDRY); load(bias, ADDRY + B);

    MANNIX\_convolution\_layer       (ADDRX,  ADDRY,  ADDR\_CONV\_RES);

    MANNIX\_non\_linearity\_activation(ADDR\_CONV\_RES, ADDRY +offest, ADDR\_CONV\_RES);

    MANNIX\_pull\_layer              (ADDR\_CONV\_RES, ADDRY +offest, ADDR\_PULL\_RES);

    MANNIX\_fully\_conneted          (ADDR\_PULL\_RES, ADDRY +offest, RESULT);

}

*כאשר הפלט של כל שלב הוא הקלט של השלב הבא.*

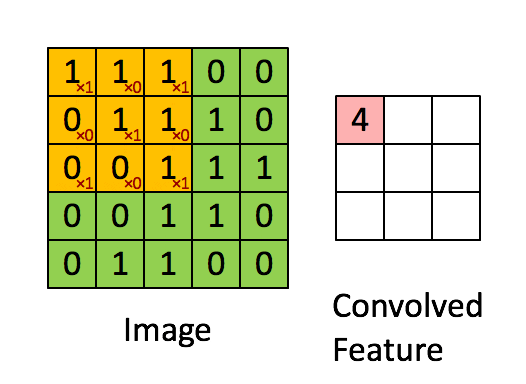
*לצורך פישוט התוכנית בשלב הראשון אנחנו מקצים איזור בזיכרון (לא מצויין בקוד למעלה) וטוענים את התמונה לזיכרון של ה MANNIX.*

*בשלב הבא חילקנו את התוכנית לארבע פונקציות אשר כל אחת מהן פונה ליחידת עיבוד אחרת.*

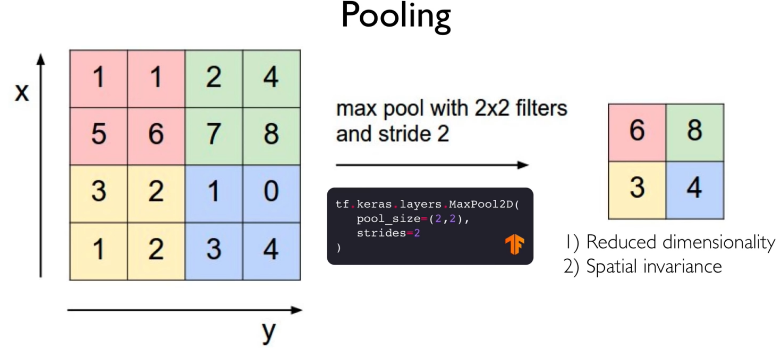
*את הקלט ניתן לסווג לשני חלקים:*

1. ***Address*** *:*

* *ליחידת הקונבולוציה : על התוכנה לשלוח אל יחידת העיבוד את כתובת ההתחלה של המידע אותו צריך לעבד (יצויין* ***בADDRX****), את כתובת המשקולות שיש לטעון (יצויין* ***בADDRY****) ואת כתובת החזרה לשמירת הפלט (יצויין* ***בADDRZ****).*
* *ליחידת הpull layer : על התוכנה לשלוח אל יחידת העיבוד את כתובת ההתחלה (יצויין* ***בADDRX****) ואת כתובת החזרה לשמירת הפלט (יצויין* ***בADDRZ****).*
* *ליחידת האקטיבציה : על היחידה לשמור את כתובת החזרה לשמירת הפלט (יצויין* ***בADDRZ****), כתובת זו היא גם כתובת הקלט.*
* *ליחידת הfcc : על התוכנה לשלוח אל יחידת העיבוד את כתובת ההתחלה (יצויין* ***בADDRX****), את כתובת המשקולות שיש לטעון (יצויין* ***בADDRY****), את כתובת הבייס (יצוין* ***בADDRB*** *) ואת כתובת החזרה לשמירת הפלט (יצויין* ***בADDRZ****).*

1. ***Sizes*** *(עבור חלק מיחידות העיבוד בלבד):*

* *ליחידת הקונבולוציה על התוכנה לשלוח את אורך התמונה )יצוין ב****Xm****) , רוחבה (יצוין ב****Xn****), אורך החלון (יצוין בYm) ורוחבו (יצוין בYn).*

**

* *ליחידת הpull layer על התוכנה לשלוח את אורך החלון (יצוין ב****Xm),***  *רוחב החלון (יצוין ב****Xn****), צעד (יצוין בstride), אורך המטריצה המתקבלת (יצוין* ***בPm*** *) ורוחב המטריצה (****Pn****).*
* *ליחידת האקטיבציה על התוכנה לשלוח את גודל אורך החלון (יצוין ב****Xm),***  *רוחב החלון (יצוין ב****Xn****). שיטת הביצוע תהיה* ***ReLU****.*
* *ליחידת הfcc יש לשלוח את אורך וקטור הקלט (יצוין בXn), אורך וקטור המשקולות (יצוין בYm), רוחב וקטור המשקולות (יצוין בYn) ואורך הבייס (יצויין בBn).*

*הערה: ראה קובץ mannix\_software\_register\_document.doc .*

*בשלב הבא ( לא מצויין בקוד למעלה) התוכנה תקרא את התוצאה מכתובת הRESULT ותחזיר אותה למחשב.*

*בקובץ זה מתוארת תוכנית העבודה והחלוקה לשלבים, דגשים והערות לכל שלב, ושיערוך הזמן.*

***שלב ראשון – pure software***

*תיאור :*

*בניית תוכנה שתדמה את אופן הפעולה של מאיץ רשת נוירונים (MANNIX) באופן תכנותי לחלוטין. התוכנית תתאים למבנה התוכנה המוזכר בהקדמה. אופן הפעולה של התוכנה יהיה סדרתי, כלומר – שום יחידה לא תתחיל עבודה לפני שקודמתה סיימה. בנוסף, כל פונקציה תדמה את אופן הפעולה של אחת מיחידות העיבוד באופן זה שהקלט והפלט (שמירת המידע בזיכרון) יהיו זהים.*

*מטרה :*

*יצירת מעטפת תוכנה לניהול הזיכרון, בניית מודל שיעזור לבדוק האם כל יחידה חומרה מבצעת את תפקידה ויקל על האינטגרציה.*

*דגשים :*

* *המודל ייכתב בשפת C במערכת ההפעלה windows על מנת להקל על המעבר לקוד RISC-V בהמשך ולאפשר גמישות בכתיבת הקוד.*
* *התאמה למבנה החומרה – קלט התוכנה יהיה זהה לזה של יחידות העיבוד.*
* *מודולריות – לאחר בניית המודל, בכל שלב יהיה ניתן לשלב כל אחת מהיחידות באופן עצמאי.*

*קלט – פלט :*

*קלט:*

* *קלט עבור התוכנה – מקבלת סט תמונות, משקלים רלוונטיים וביאס.*
* *קלט עבור כל פונקציה (מדמה מודול) – פוינטר לכתובת התמונה, פוינטר למשקלים והבאייס ופוינטר לכתובת שבה יאחוסן המידע.*

*פלט :*

*none, התוצאה של החישוב תשמר בכתובת 'RESULT' ייעודית.*

*בדיקות :*

* *Dataset - MNIST (אולי cifar10) .*
* [*https://www.kaggle.com/oddrationale/mnist-in-csv?select=mnist\_test.csv*](https://www.kaggle.com/oddrationale/mnist-in-csv?select=mnist_test.csv) *דאטה סט בפורמט csv (אמור להקל על החישוב הן של תוכנה והן של החומרה).*

*צפי עבודה:*

* *בניית הפונקציות – שבוע/שבוע וחצי לכל פונקציה.*
* *אינטגרציה – שבועיים+-.*
* *בדיקות תקינות – שבועיים+-.*

*שאלות/ בעיות:*

* *באיזה טיפוס שומרים את המשקלים (map, מספרים לא שלמים, שלמים).*

***שלב שני – ניהול זכרון –***

*תיאור:*

*במקום לקבל מידע אחד בכל פעם ולהמתין לסיום העיבוד שלו, נשלח למאיץ בכל מחזור מידע אחר על מנת לנצל את היחידות שסיימו לעבוד. על מנת לעשות זאת אנחנו נתקלים בשתי בעיות:*

1. *ניהול הקצאת זיכרון- כל תמונה שמגיעה צריכה להטען לאזור פנוי בזיכרון בנוסף להקצת זיכרון לכל יחידה.*
2. *ניהול כתובות ליחידות העיבוד – כל יחידת עיבוד מטפלת במידע בזמן שלא תלוי ביחידות האחרות. דבר זה יוצר בעיה כאשר כמה תמונות ממתינות ליחידה אחת.*

*מכיוון שפעולות אלה בחומרה מעלות את הסיבוכיות, החלטנו שנכון לשלב זה הן ינוהלו על ידי התוכנה.*

*הפתרון לבעיה הראשונה יהיה לשמור אינדקס כתובות. מכיוון שלכל תמונה יש להקצות גודל אחיד של זיכרון, ניתן לשמור מספר התמונות שהתקבלו מודולו כמות התמונות שמגיעה עד המאיץ מסיים לטפל בתמונה אחת.*

*הפתרון לבעיה השנייה יהיה לנהל תור כתובות (ADDRESS QUEUE) לכל יחידה (גודל התור יכול להיות שונה מיחידה ליחידה.*

*מטרה :*

*יצירת מודל ניהול זיכרון באופן כזה שיהיה ניתן להחליף אותו במנגנון חומרתי.*

*דגשים :*

* *שלב זה גם כן ינוהל בתוכנה בלבד.*
* *תור הכתובות ינוהל מחוץ לפונקציות שכן בשלב זה עדיין לא ברור האם נחליף אותו במנגנון חומרה.*
* *הוספת רכיבי תוכנה (ת'רדים) המאפשרים מקבול על מנת לבצע הדמייה של מנגנון החומרה. הכוונה היא ליצירת מצב שבו כמה יחידות ממתינות לאותה פונקציה.*
* *ברגע שסיימנו לטפל בתמונה, המקום שלה בזיכרון יתפנה.*

*בדיקות:*

* *הרצת סימולציות במטרה לנסות לפגוע במערכת – כדי להצליח לתקן את הטיימינג.*
* *מעבר נקי בין מודולים – ברגע שחישוב מסוים (גם אם הוא חלקי) סיים – בהנחה והמודול הבא (או לפחות חלק מתאים ממנו) סיים את החישוב הקודם (בצורת פייפלין) נעביר את החישוב החלקי אל המודול הבא.  
  המטרה היא ליצור מעבר חופשי וכמה שיותר מהיר ויעיל במקום כל פעם לחכות שמודול מסוים יסיים את החישוב שלו.*
* *הרצת כמה תמונות אחת אחרי השנייה.*

*צפי:*

*סיום המודל התוכנתי כחודש אחרי פסח.*

*בעיות:*

*None.*

***שלב שלישי – שילוב יחידות עיבוד בתוך החומרה –***

*תיאור :*

*שלב זה, ייתכן שיהיה מקביל לשלב 2 כתלות בקצב ההתקדמות של בניית יחידות העיבוד. מטרתו להכניס כל אחת מהיחידות לתוך מודל התוכנה. על מנת לעשות זאת יש צורך להמיר את קוד התוכנה לקוד RISC-V.*

*מטרה :*

* *להטמיע כל מודול חומרתי שנבנה במערכת התוכנתית בנפרד.*
* *המרת קוד התוכנית לקוד RISCV.*

*דגשים :*

*הטמעת כל יחידת עיבוד תעשה בפניי עצמה לפני שילובה עם יחידות אחרות על מנת לאתר תקלות.*

*בדיקה –*

* *תוצאות זהות לסימולציה התוכנתית של המודול.*
* *שילובי מודולים ובדיקת התאמה לתוצאות התוכנה.*
* *שילוב כל היחידות.*

*צפי:*

*None*

*הערות:*

* *לבדוק בעיות טיימינג וזכרון למיניהם.*
* *בדיקת כדאיות של בניית תוכנה שמשווה את תוצאות החומרה והתוכנה.*

*שלב רביעי – יצירת מעטפת python לקוד /\**אופציונלי*\*/*

דרישות מהחומרה –

* *לשמור במקום בזכרון סיגנל done – רגיסטר בזכרון הכללי שישמור את הסיגנל לכל אינסטנס בכל מודול על מנת שנוכל לקדם את התור ולהקצות מקום בצורה יעילה.*

רשימת סיגנלים -