למידת מכונה – תרגיל 4

כמה הערות לגבי הדו"ח:

לגבי סעיף ד' של ההיפר פרמטרים, מספר ה-epoches קבלנו דרישה בתרגיל לקבוע אותו ל-10 ולכן לא נפרט על זה.

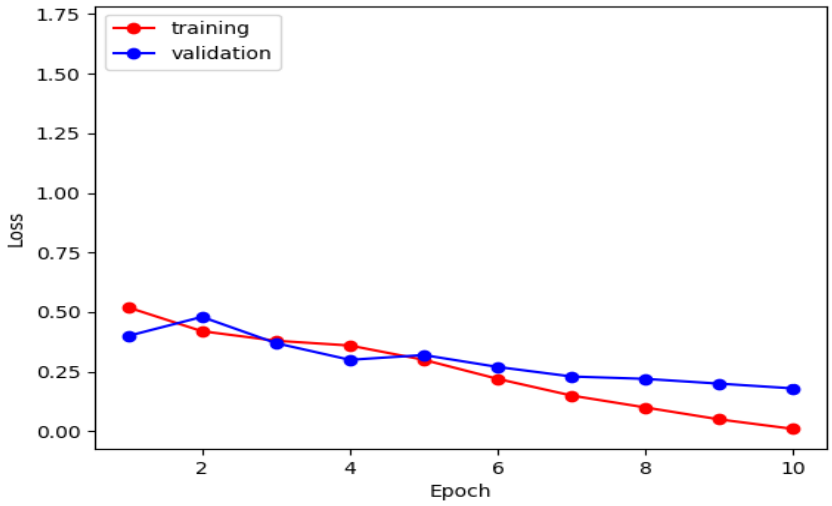
בחרנו להשאיר את מודל A בקוד ולהגיש אותו:

כי הוא הביא לנו את האחוז הגבוה ביותר (בערך 87-86%).

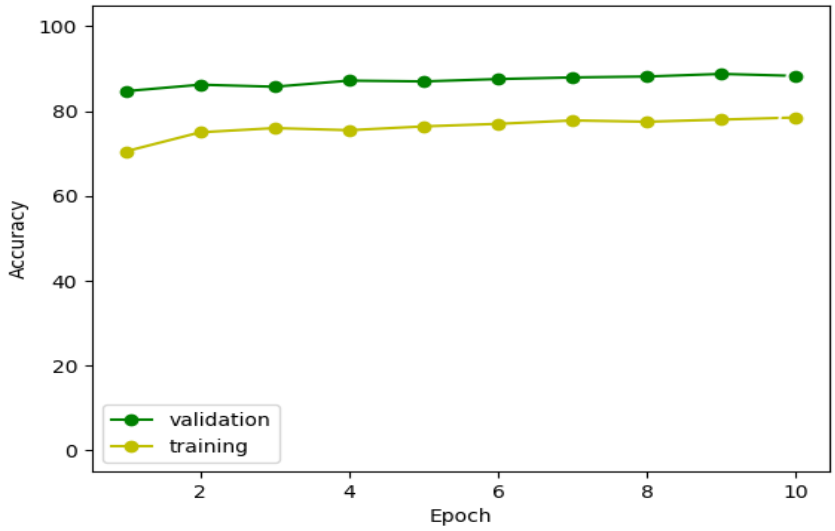
מודל A :

רשת נוירונים מורכבת מ2 שכבות פנימיות (הראשונה בגודל 100 והשניה בגודל 50) עם שימוש בפונקציית אקטיבציה –relu שממומשת בpytorch (torch.nn.functional.relu), ועל שכבת הפלט הפעלנו את log_softmax (torch.nn.functional.log_softmax) והאופטמייזר לאימון הוא SGD (torch.optim.SGD).

1 - גרף עבור ה-loss הממוצע בכל epoch עבור ה- training set וה- validation set :



2 - גרף עבור ה-accuracy הממוצע בכל epoch עבור ה- training set וה- validation set :



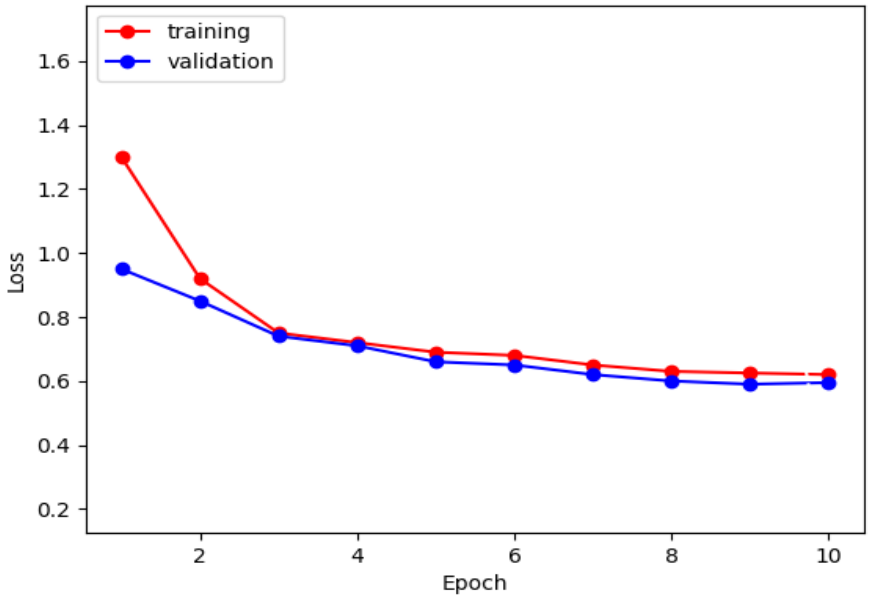
3 – hyper-parameters :

בדקנו מספר ערכים ל- learning rate כמו : 0.01,0.05,0.1,0.001,0.005,0.009 ובסוף אחרי בדיקות של accuracy ו- loss עבור כל ערך, קיבלנו ש 0.1 נותן את accuracy הכי גבוה וה- loss הכי נמוך.

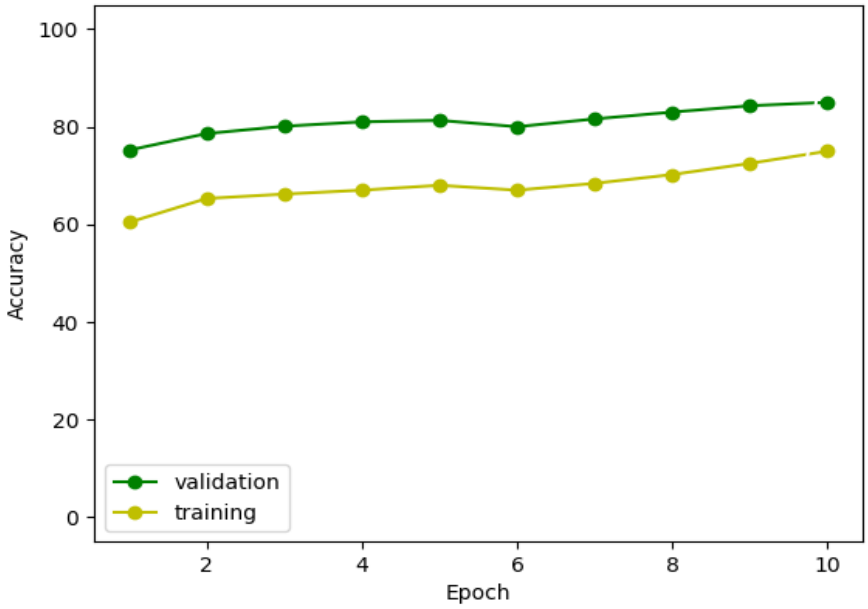
מודל B :

דומה למודל A , ההבדל היחיד זה האופטימייזר שהתבצע על אימון הרשת , במודל זה השתמשנו באופטימייזר ADAM (torch.optim.Adam).

1 - גרף עבור ה-loss הממוצע בכל epoch עבור ה- training set וה- validation set :



2 - גרף עבור ה-accuracy הממוצע בכל epoch עבור ה- training set וה- validation set :



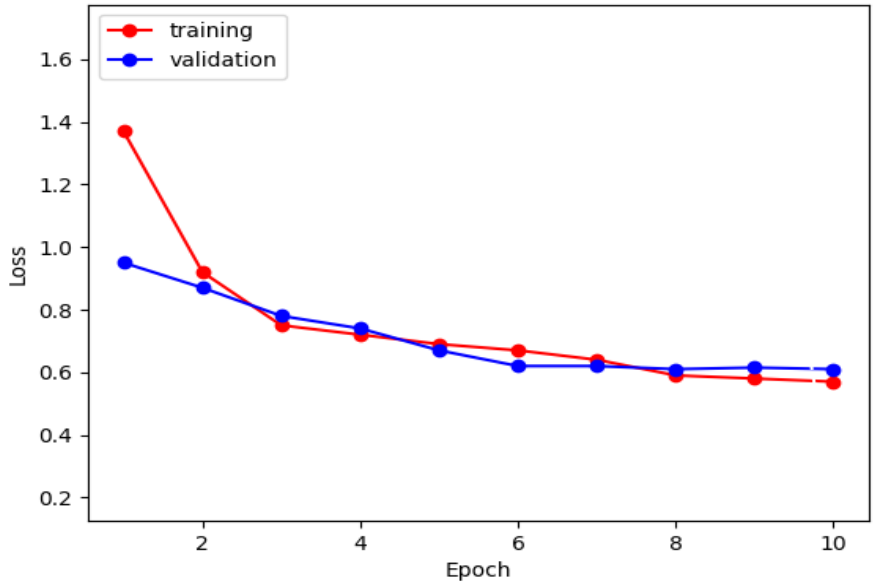
3 – hyper-parameters :

בדקנו מספר ערכים ל- learning rate כמו : 0.01,0.015,0.1,0.001,0.005,0.0015 ובסוף אחרי בדיקות של accuracy ו- loss עבור כל ערך, קיבלנו ש 0.001 נותן את accuracy הכי גבוה וה- loss הכי נמוך.

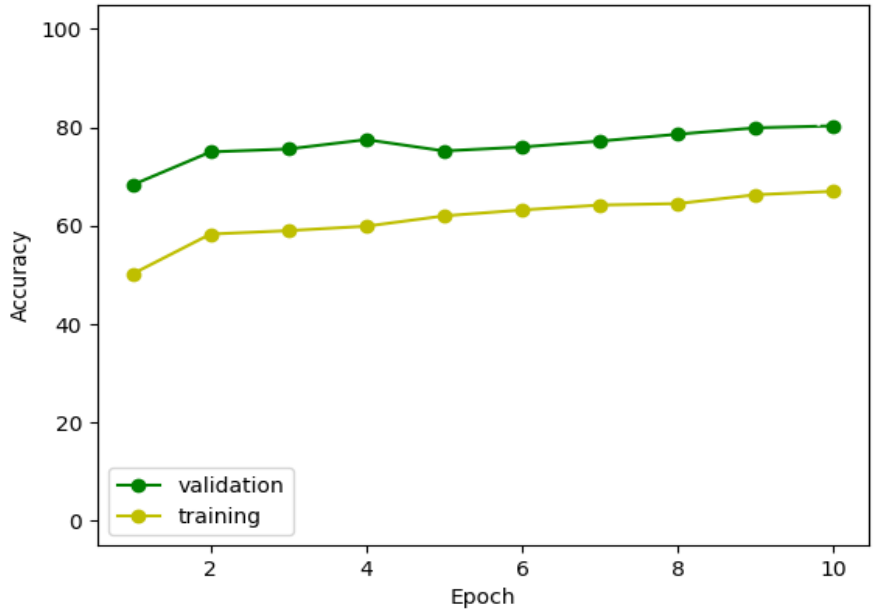
מודל C :

דומה למודל B אבל השינוי היחיד הוא שביצענו ה- Dropout (torch.nn.Dropout) על השכבות הפנימיות שהמיקום שלו (של ה – dropout) היה ב-output של השכבות הפנימיות כמו שנדרש בתרגיל.

1 - גרף עבור ה-loss הממוצע בכל epoch עבור ה- training set וה- validation set :



2 - גרף עבור ה-accuracy הממוצע בכל epoch עבור ה- training set וה- validation set :



3 – hyper-parameters :

עבור ה-learning rate בחרנו כמו במודל B (0.001) שנתן את accuracy הכי גבוה וה- loss הכי נמוך.

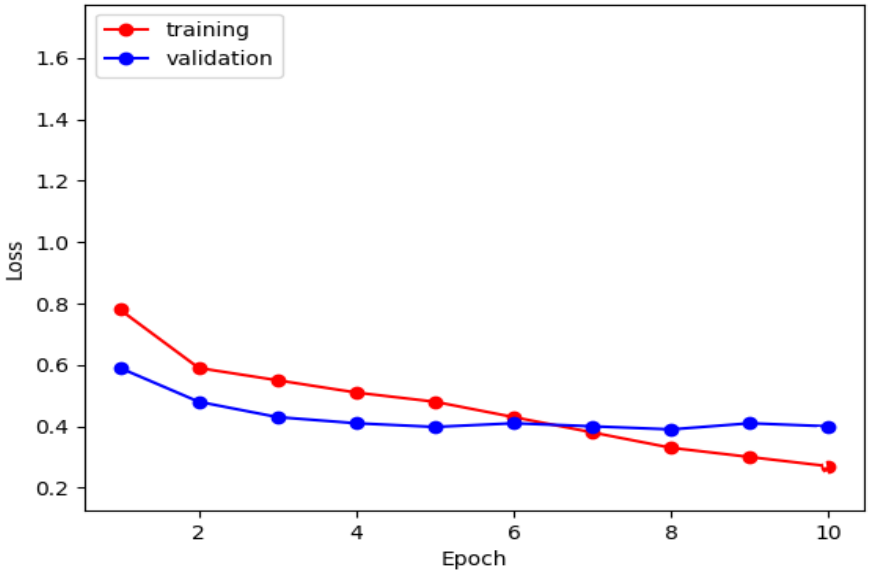
עבור ה dropout : פונקציה זו מקבלת p שהוא הערך של ההסתברות לקחת את הנוירון המסוים מתוך הרשת, ניסינו עבור כמה ערכים כמו : 0.4,0.45,0.5,0.55,0.6 (כי ידוע שערכים טובים הם בין 0.4 ל 0.6)וקיבלנו 0.45 נותן אחוז דיוק גבוה ביותר.

מודל D :

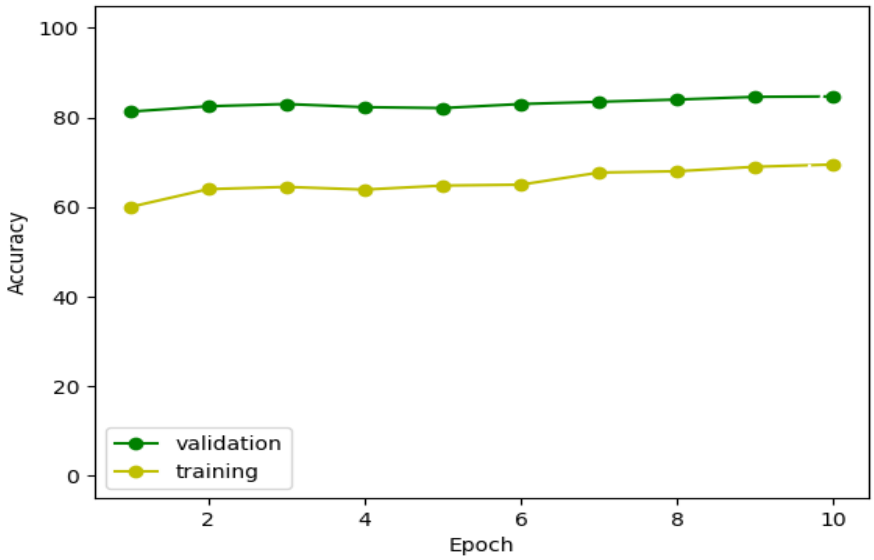
דומה למודל B אבל השינוי היחיד הוא שביצענו Batch Normalization.

נימוק למיקום ביצוע ה-Batch Normalization : ביצענו את ה-Batch Normalization לפני הפעלת פונקציית האקטיבציה: בדקנו פעם לפני ופעם אחרי וקבלנו שזה משנה את אחוזי הדיוק ובסוף קבענו לבצע לפני , כי זה הניב לאחוז דיוק יותר גבוה.

1 - גרף עבור ה-loss הממוצע בכל epoch עבור ה- training set וה- validation set :



2 - גרף עבור ה-accuracy הממוצע בכל epoch עבור ה- training set וה- validation set :



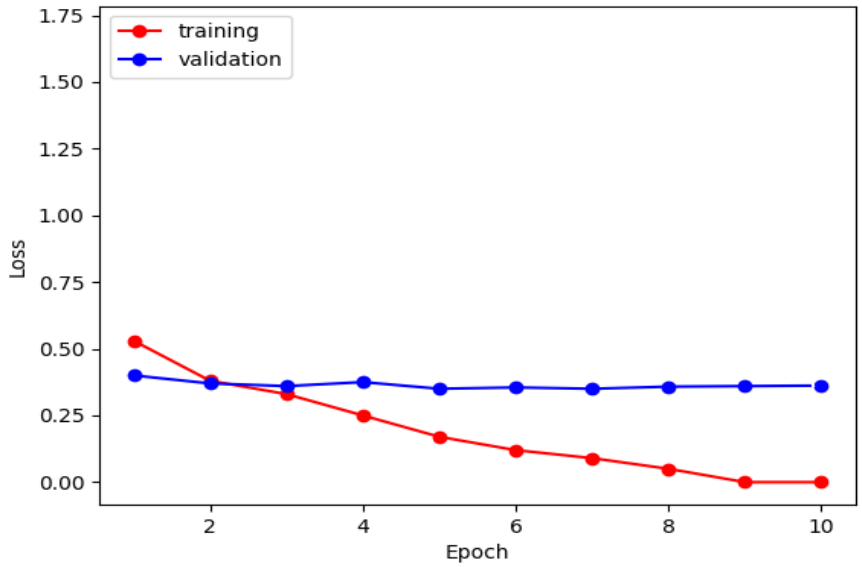
3 – hyper-parameters :

בדקנו מספר ערכים ל- learning rate כמו : 0.01,0.015,0.1,0.001,0.009,0.005 ובסוף אחרי בדיקות של accuracy ו- loss עבור כל ערך, קיבלנו ש 0.001 נותן את accuracy הכי גבוה וה- loss הכי נמוך.

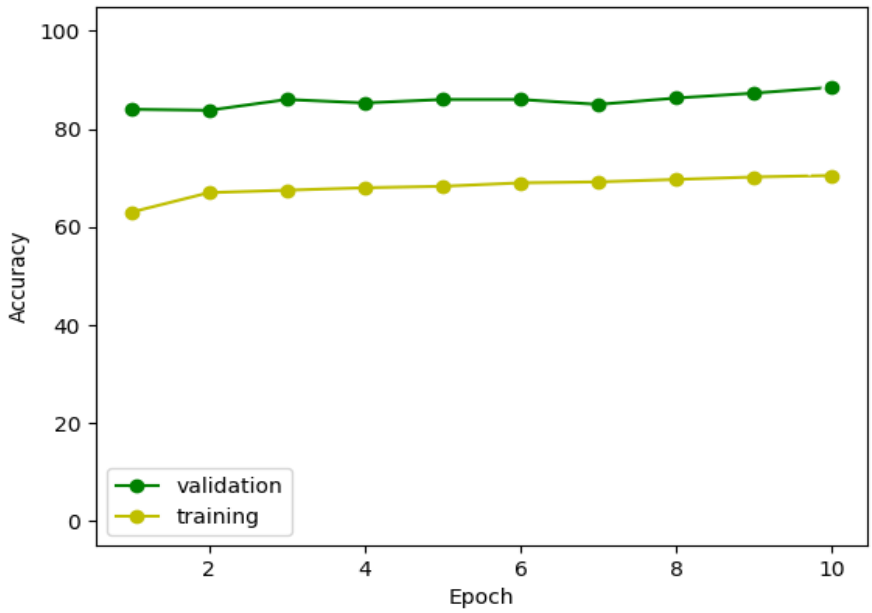
מודל E :

רשת נוירונים מורכבת מ5 שכבות פנימיות (בגדלים – 128,64,10,10,10) עם שימוש בפונקציית אקטיבציה –relu שממומשת בpytorch (torch.nn.functional.relu), ועל שכבת הפלט הפעלנו את log_softmax (torch.nn.functional.log_softmax) והאופטמייזר לאימון הוא Adam (torch.optim.Adam) וזה אחרי בדיקת כמה אופטימייזרים שונים.

1 - גרף עבור ה-loss הממוצע בכל epoch עבור ה- training set וה- validation set :



2 - גרף עבור ה-accuracy הממוצע בכל epoch עבור ה- training set וה- validation set :



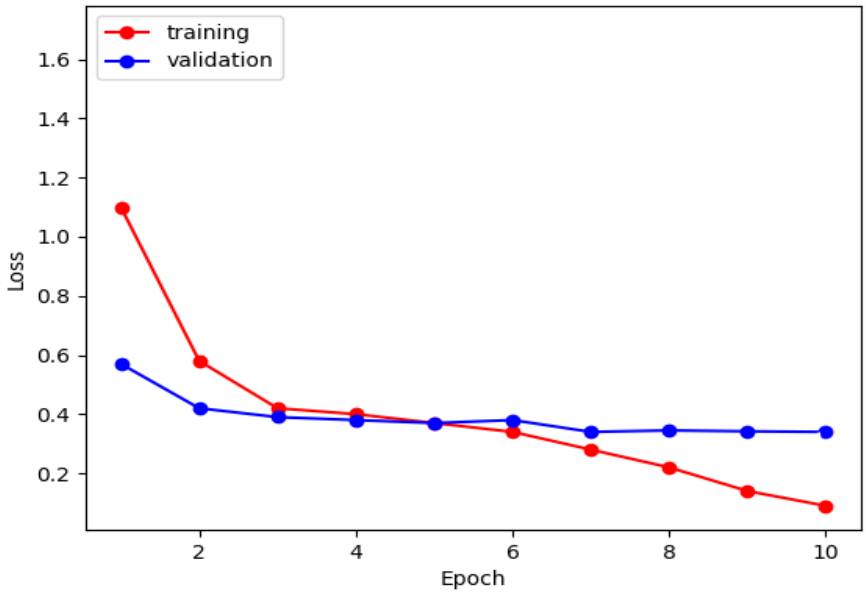
3 – hyper-parameters :

בדקנו מספר ערכים ל- learning rate כמו : 0.01,0.015,0.1,0.001,0.009,0.005 ובסוף אחרי בדיקות של accuracy ו- loss עבור כל ערך, קיבלנו ש - 0.005 נותן את accuracy הכי גבוה וה- loss הכי נמוך.

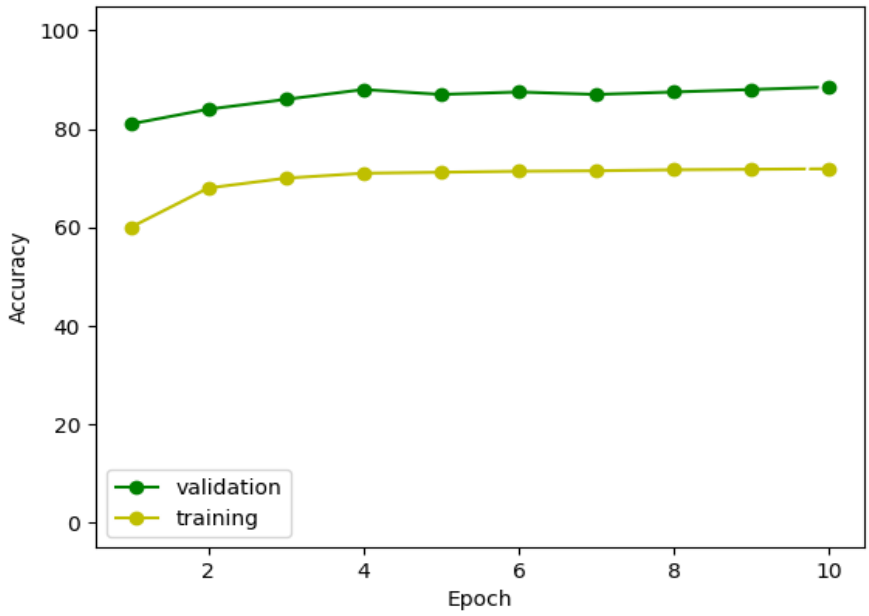
מודל F :

דומה למודל E אבל השינוי הוא שהשתמשנו בפונקציית אקטיבציה Sigmoid .

1 - גרף עבור ה-loss הממוצע בכל epoch עבור ה- training set וה- validation set :



2 - גרף עבור ה-accuracy הממוצע בכל epoch עבור ה- training set וה- validation set :



3 – hyper-parameters :

בדקנו מספר ערכים ל- learning rate כמו : 0.01,0.015,0.1,0.001,0.009,0.005 ובסוף אחרי בדיקות של accuracy ו- loss עבור כל ערך, קיבלנו ש – 0.009 נותן את accuracy הכי גבוה וה- loss הכי נמוך.