למידת מכונה תרגיל 4 –report

רונלי ויגננסקי 211545892

ירין דאדו 316383298

בתרגיל מימשנו רשת נוירונים ע"י pytorch.

בנינו שישה מודלים בהתאם להוראות התרגיל.

נפרט להלן את הדרך בה אימנו את המודלים. לפי התוצאות שקיבלנו כיילנו את הפרמטרים.
הפרמטרים שנציג הם מה שמצאנו אחרי הרצות של פרמטרים שונים ולקיחת הערכים שהובילו ל loss מינימלי.

ראשית, טענו את ה data שקיבלנו ונרמלנו אותו כדי לשפר את ביצועי הרשת.

במהלך האימון חילקנו את הdata ל train 80% ו validation 20% כפי שהתבקשנו. שלחנו לפונקציית האימון את החלק של הtrain, שם ביצענו forward וחישוב loss. לאחר מכן שלחנו לפונקציית test שם "תיקנו" את המשקולות ע"י ביצוע ה backward.

כדי להגיע לתוצאות עבור ששת המודלים, הרצנו את מה שתארנו לעיל מספר רב של פעמים על פרמטרים שונים של learning rate, droputs, batch size, optimizer, activation function. כמובן ששאר הפרמטרים כמו epochs, מספר השכבות החבויות וגודלן עשינו כפי שהתבקשנו בהוראות.

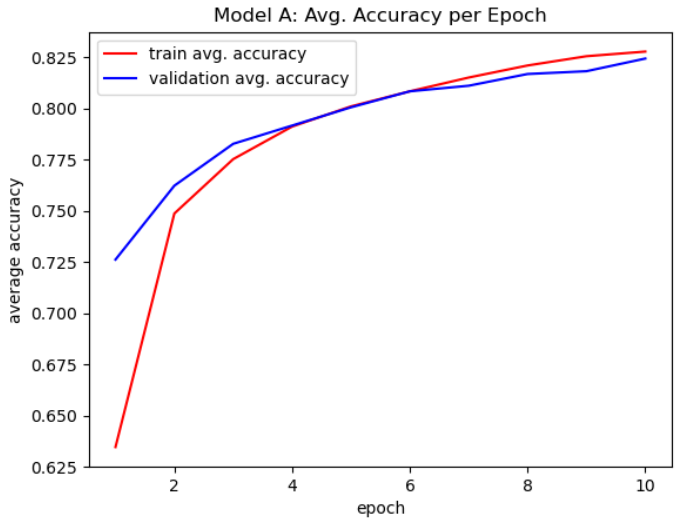
<u>מודל A:</u>

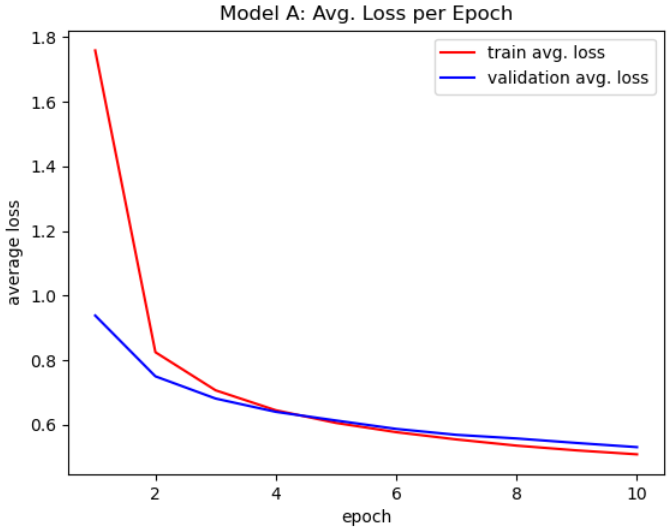
שתי שכבות חבויות, הראשונה בגודל 100 והשנייה 50.

optimizer: SGD, activation function: ReLU .

<u>ההיפר פרמטרים שבחרנו הם:</u> batch size =128, learning rate= 0.000001

הגענו לכ 82% דיוק.





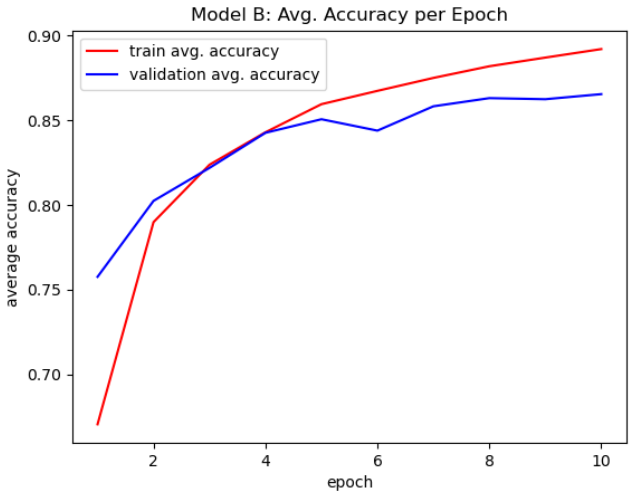
מודל B:

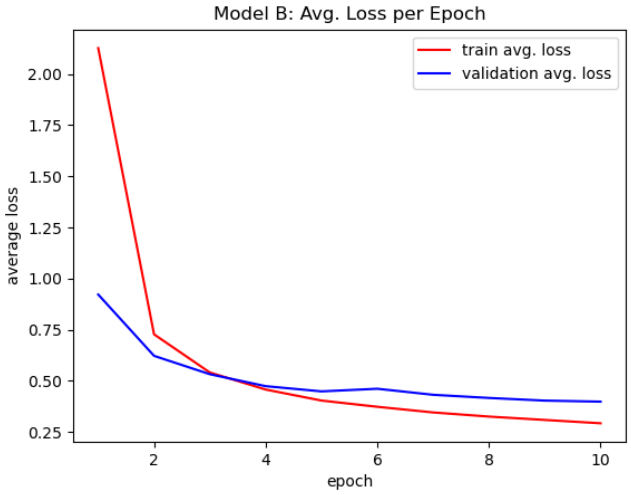
שתי שכבות חבויות, הראשונה בגודל 100 והשנייה 50.

optimizer: ADAM, activation function: ReLU .

<u>ההיפר פרמטרים שבחרנו הם:</u> batch size =128, learning rate= 0.0001

הגענו לכ 86% דיוק.





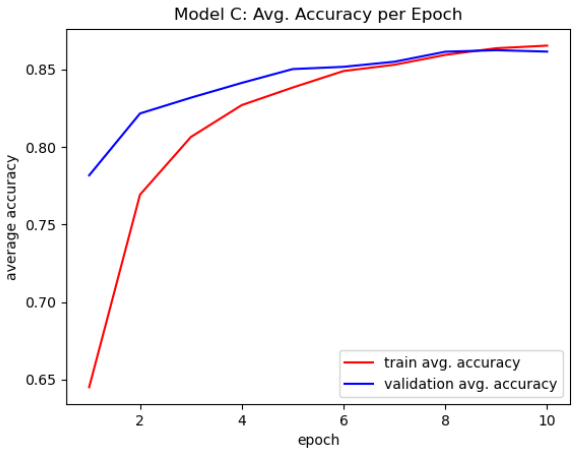
מודל C:

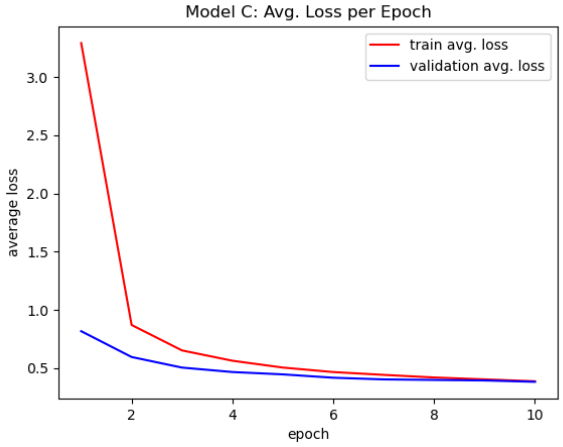
שתי שכבות חבויות, הראשונה בגודל 100 והשנייה 50.

optimizer: ADAM, activation function: ReLU. הפעלנו dropout על הפלט של השכבה שיצאה מפונקציית האקטיבציה.

<u>ההיפר פרמטרים שבחרנו הם:</u> batch size =32, learning rate= 0.0001. הערך של ה dropout שבחרתי הוא 0.1. ניתן לראות שבגלל שביצענו dropout הביצועים בtrain ירדו ולכן ה train נמוך מה validation בaccuracy.

הגענו לכ 86% דיוק.





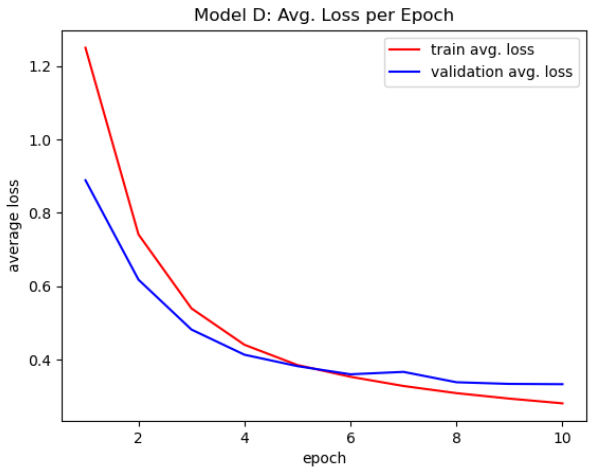
מודל D:

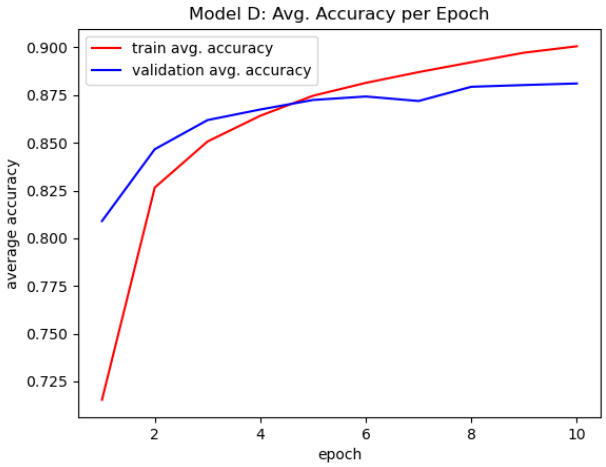
שתי שכבות חבויות, הראשונה בגודל 100 והשנייה 50.

optimizer: ADAM, activation function: ReLU .

<u>ההיפר פרמטרים שבחרנו הם:</u> batch size =128, learning rate= 0.0001

הגענו לכ 88% דיוק.





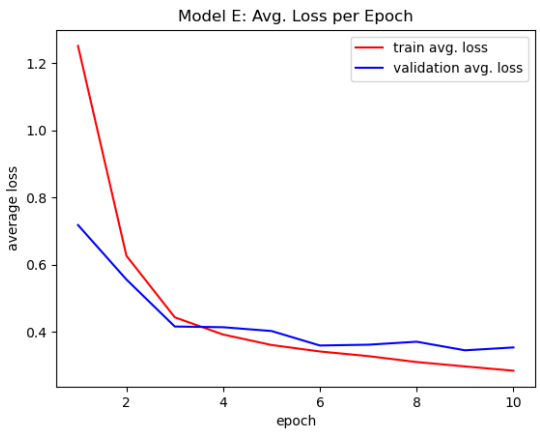
מודל E:

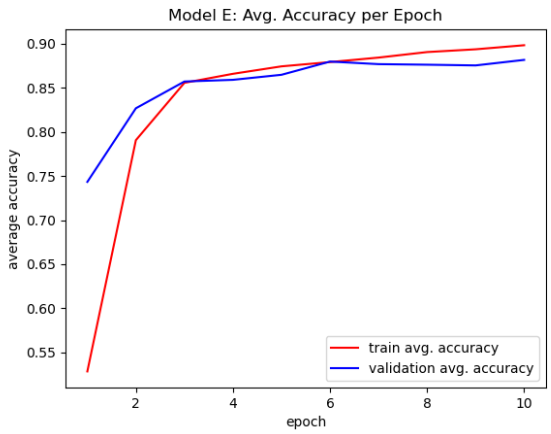
חמש שכבות חבויות, בגדלים הבאים: 128,64,10,10,10.

optimizer: ADAM, activation function: ReLU . הפעלנו batch normalization. בדקנו היכן הביצועים טובים יותר, אם מפעילים את ה batch normalization לפני או אחרי הפעלת פונקציית האקטיבציה, וראינו שההפעלה לפני פונקציית האקטיבציה מובילה לאחוזי accuracy גבוהים יותר.

<u>ההיפר פרמטרים שבחרנו הם:</u> batch size =32, learning rate= 0.0001.

הגענו לכ 88% דיוק.





מודל F:

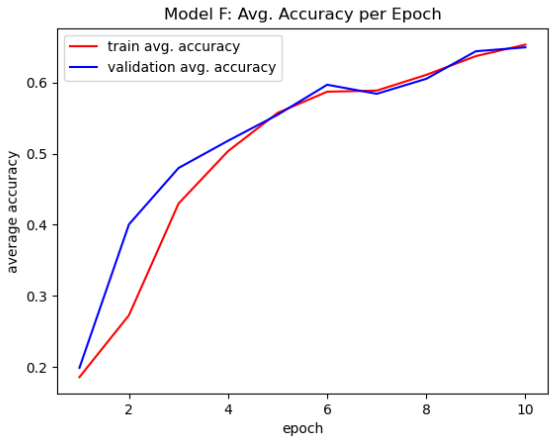
חמש שכבות חבויות, בגדלים הבאים: 128,64,10,10,10.

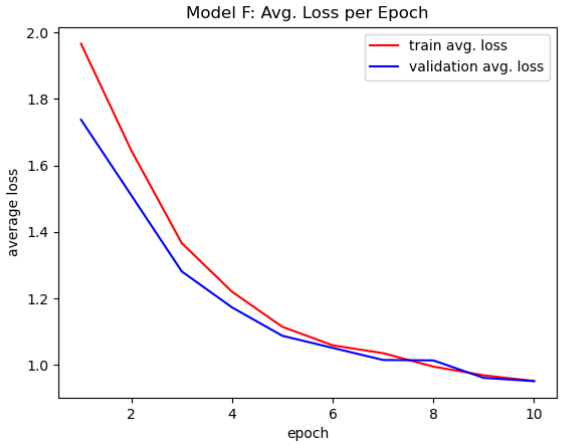
. activation function: sigmoid ,optimizer: ADAM

<u>ההיפר פרמטרים שבחרנו הם:</u> batch size =32, learning rate= 0.001.

כיוון שהשתמשנו בסיגמואיד כפונקציית האקטיבציה בכל השכבות החבויות ניתן לראות כי קיבלנו ביצועים נמוכים.

הגענו לכ 65% דיוק.





אלו ששת המודלים שנדרשנו לממש ולהציג, בעמוד הבא נציג את המודל שאנחנו בנינו והגענו לתוצאות מיטביות על ה data שקיבלנו.

מודל BestModel:

בנינו מודל שיוביל אותנו לתוצאות הטובות ביותר. בנינו אותו באופן הבא:

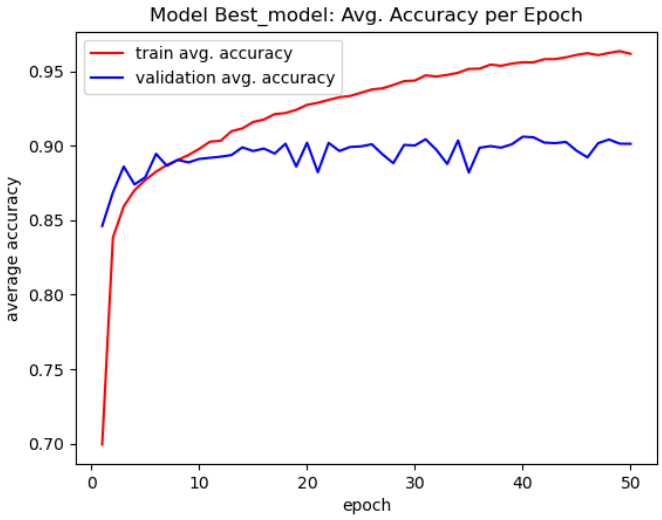
שש שכבות חבויות, בגדלים הבאים: 512, 256, 128, 64, 32, 16.

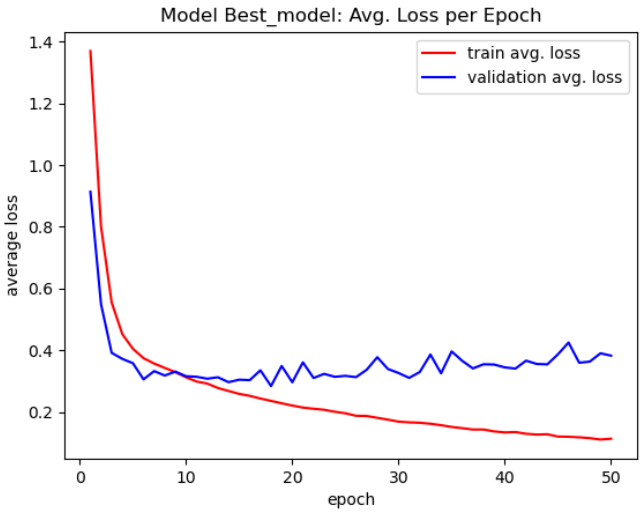
optimizer: ADAM, activation function: ReLU . הפעלנו dropout על הפלט של השכבה שיצאה מפונקציית האקטיבציה. כמו כן, הפעלנו batch normalization, וכפי שהסברתי קודם בדקנו זאת לפני ואחרי הפעלת פונקציית האקטיבציה וראינו שלפני מוביל לביצועים טובים יותר, ולכן מיקמנו לפני ביצוע הReLU.

<u>ההיפר פרמטרים שבחרנו הם:</u> batch size =128, learning rate= 0.0003, number of epochs = 50.
הערך של ה dropout שבחרתי הוא 0.1.

בחרנו היפר פרמטרים אלו כיוון שלמדנו בכיתה שפונקציית האקטיבציה ReLU מובילה לביצועים טובים.

הגענו לכ 90% דיוק.





מצורף בזה ההדפסות של האחוזים על המודל הטוב שלנו, ניתן לראות את העלייה באחוזי הדיוק של הaccuracy ככל שמספר הephoc עולה. ולראות שבסופו של דבר הגענו לאחוזי דיוק של 90.14%!

Best_model

IN EPOCH 0

(84.62%) Test set: Average loss: 10048.0858, Accuracy: 9308/11000

IN EPOCH 1

(86.87%) Test set: Average loss: 6048.2390, Accuracy: 9556/11000

IN EPOCH 2

(88.61%) Test set: Average loss: 4315.6254, Accuracy: 9747/11000

IN EPOCH 3

(87.41%) Test set: Average loss: 4101.1104, Accuracy: 9615/11000

IN EPOCH 4

(87.89%) Test set: Average loss: 3946.0237, Accuracy: 9668/11000

IN EPOCH 5

(89.46%) Test set: Average loss: 3371.1342, Accuracy: 9841/11000

IN EPOCH 6

(88.66%) Test set: Average loss: 3665.7346, Accuracy: 9753/11000

IN EPOCH 7

(89.05%) Test set: Average loss: 3508.3347, Accuracy: 9796/11000

IN EPOCH 8

(88.89%) Test set: Average loss: 3646.7909, Accuracy: 9778/11000

IN EPOCH 9

(89.12%) Test set: Average loss: 3483.0968, Accuracy: 9803/11000

IN EPOCH 10

(89.20%) Test set: Average loss: 3462.9403, Accuracy: 9812/11000

IN EPOCH 11

(89.27%) Test set: Average loss: 3394.6026, Accuracy: 9820/11000

IN EPOCH 12

(89.38%) Test set: Average loss: 3446.1698, Accuracy: 9832/11000

IN EPOCH 13

(89.90%) Test set: Average loss: 3271.8658, Accuracy: 9889/11000

IN EPOCH 14

(89.65%) Test set: Average loss: 3358.5777, Accuracy: 9862/11000

IN EPOCH 15

(89.82%) Test set: Average loss: 3344.5752, Accuracy: 9880/11000

IN EPOCH 16

(89.48%) Test set: Average loss: 3694.1859, Accuracy: 9843/11000

IN EPOCH 17

(90.15%) Test set: Average loss: 3133.1041, Accuracy: 9916/11000

IN EPOCH 18

(88.61%) Test set: Average loss: 3848.4214, Accuracy: 9747/11000

IN EPOCH 19

(90.21%) Test set: Average loss: 3268.6794, Accuracy: 9923/11000

IN EPOCH 20

(88.23%) Test set: Average loss: 3976.4991, Accuracy: 9705/11000

IN EPOCH 21

(90.20%) Test set: Average loss: 3420.2383, Accuracy: 9922/11000

IN EPOCH 22

(89.65%) Test set: Average loss: 3566.7879, Accuracy: 9862/11000

IN EPOCH 23

(89.93%) Test set: Average loss: 3462.3063, Accuracy: 9892/11000

IN EPOCH 24

(89.96%) Test set: Average loss: 3496.6256, Accuracy: 9896/11000

IN EPOCH 25

(90.11%) Test set: Average loss: 3450.0045, Accuracy: 9912/11000

IN EPOCH 26

(89.43%) Test set: Average loss: 3713.2265, Accuracy: 9837/11000

IN EPOCH 27

(88.84%) Test set: Average loss: 4159.2247, Accuracy: 9772/11000

IN EPOCH 28

(90.06%) Test set: Average loss: 3739.9603, Accuracy: 9907/11000

IN EPOCH 29

(90.03%) Test set: Average loss: 3598.0592, Accuracy: 9903/11000

IN EPOCH 30

(90.45%) Test set: Average loss: 3423.8545, Accuracy: 9949/11000

IN EPOCH 31

(89.73%) Test set: Average loss: 3633.4337, Accuracy: 9870/11000

IN EPOCH 32

(88.78%) Test set: Average loss: 4254.9082, Accuracy: 9766/11000

IN EPOCH 33

(90.36%) Test set: Average loss: 3586.3384, Accuracy: 9940/11000

IN EPOCH 34

(88.21%) Test set: Average loss: 4366.3465, Accuracy: 9703/11000

IN EPOCH 35

(89.87%) Test set: Average loss: 4028.1238, Accuracy: 9886/11000

IN EPOCH 36

(89.98%) Test set: Average loss: 3760.9380, Accuracy: 9898/11000

IN EPOCH 37

(89.88%) Test set: Average loss: 3908.9403, Accuracy: 9887/11000

IN EPOCH 38

(90.11%) Test set: Average loss: 3898.1573, Accuracy: 9912/11000

IN EPOCH 39

(90.62%) Test set: Average loss: 3797.6670, Accuracy: 9968/11000

IN EPOCH 40

(90.57%) Test set: Average loss: 3754.5188, Accuracy: 9963/11000

IN EPOCH 41

(90.22%) Test set: Average loss: 4036.2014, Accuracy: 9924/11000

IN EPOCH 42

(90.18%) Test set: Average loss: 3920.3010, Accuracy: 9920/11000

IN EPOCH 43

(90.26%) Test set: Average loss: 3904.6459, Accuracy: 9929/11000

IN EPOCH 44

(89.67%) Test set: Average loss: 4262.7420, Accuracy: 9864/11000

IN EPOCH 45

(89.23%) Test set: Average loss: 4681.5065, Accuracy: 9815/11000

IN EPOCH 46

(90.18%) Test set: Average loss: 3963.1246, Accuracy: 9920/11000

IN EPOCH 47

(90.43%) Test set: Average loss: 4003.6369, Accuracy: 9947/11000

IN EPOCH 48

(90.15%) Test set: Average loss: 4298.4543, Accuracy: 9916/11000

IN EPOCH 49

(90.14%) Test set: Average loss: 4215.3123, Accuracy: 9915/11000