תרגיל בית 1 - סיכום ותוצאות:

בחרנו להשתמש כתבנית ראשונית בקוד הרשום בקובץ classification_mnist.lua ועליו ביצענו שינויים שונים על מנת להוסיף רגולריזציה ולבצע אופטימיזציה מבחינת זמני חישוב, מזעור מספר הפרמטרים ומיקסום רמת הדיוק של הפרדיקציה עבור דגימות מבחן.

ארכיטקטורת המודל מתבסס על SGD אשר עושה שימוש בפרמטר המומנטום שמאותחל ל0.9. היתרון בשימוש בארכיטקטורה זו טמון בכך שהוא מתכנס מהר יותר מGD רגיל לנקודה האופטימאלית. ההבדל בין SGD ל GD הוא שבראשון העדכון של הגרדיאנטים נעשה על סמך דגימת אימון אחת או כמה דגימות, ואילו בשני עדכון הגרדיאנט מתבצע עפ"י חישוב הממוצע של כל דגימות האימון.

כמו כן, ארכיטקטורת המודל עושה שימוש ב-weightDecay אשר משמשת להגבלת מספר הפרמטרים החופשיים במודל ובכך להימנעות מאובר-פיטינג. היא עושה זאת באמצעות פרמטר רגולריזציה ("רעש" , מאותחל כך: weightDecay = 1e-3) שנותן עלות גבוהה למשקולות גדולות. קבוע הרגולריזציה הזה משמש כמכריע בקביעת הטרייד-אוף בין מזעור השגיאה לבין הימנעות ממשקולות גדולות.

הקריטריון של המודל הוא ClassNLLCriterion במטרה למזער את מינוס log הנראות.

המודל עושה שימוש בConfusion Matrix אשר משווה בין תוויות אפשריות לתוויות שהתקבלו ומחשבת את מספר הפעמים שהתוצאה שהתקבלה הייתה נכונה (ביחס לכל תווית אפשרית).

בתחילת הסקריפט, הפכנו את הדגימות לfloat. כמו כן, נרמלנו את הדגימות – מרכזנו אותן סביב האפס.

אופטימיזציה נוספת שעשינו הייתה להשתמש בפונקציית Shuffle אשר מערבלת את דגימות האימון ובכך נקטין את ה overfitting ושגיאת המבחן תקטן.

על מנת לעמוד בדרישת מספר הפרמטרים ניסינו לשנות את הוקטור layerSize המציין את מספר הפרמטרים בכל שכבה. בחלק מהניסיונות ניסינו לעמוד בדרישה זו בכך ששינינו את מספר השכבות עצמן, זאת בכדי שתתבצע למידה עמוקה יותר. להלן התוצאות:

הוקטור layerSize	מספר הפרמטרים	epoch אופטימלי	Min test error	Min test loss	Max global correct
inputSize, 16,128,256	50330	18	0.03215	0.10115	96.7848%
inputSize, 64,64,64,64	**63370**	**19**	**0.02313**	**0.081142**	**97.6862%**
inputSize , 32,64,128,64,128	53418	16	0.03114	0.106964	96.8850%

הלמידה מתבצעת באמצעות הפונקציה forwardNet.

פונקציה זו עושה שימוש בפונקציות forward, backword הסטנדרטיות (לא מימשנו מחדש) שרצות על גבי שכבות הרשת (הפרמטרים) , מחשבות את הגרדיאנטים ומעדכנות את המשקולות.

learningRate = 0.1 – מקדם הלמידה של המודל. ביצענו אופטימיזציה בבחירת ערך אחר עבור הLearning Rate, להלן התוצאות:

Learning Rate	epoch אופטימלי	Min test error	Min test loss	Max global correct
0.1	19	0.02313	0.081142	97.6862%
0.2	**20**	**0.02253**	**0.0758**	**97.7463%**

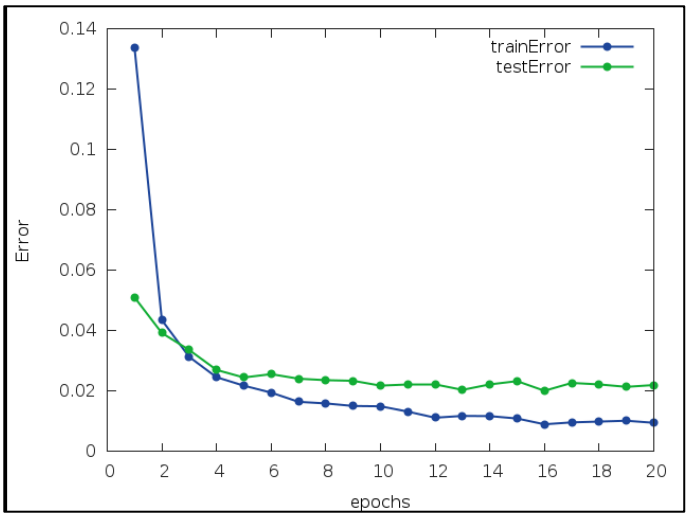
בשלב הזה ניסינו לבצע שינויים בגודל הbatch ו/או בכמות הepochs. ראינו ששינויים אלה לא משפרים את הMin test error .

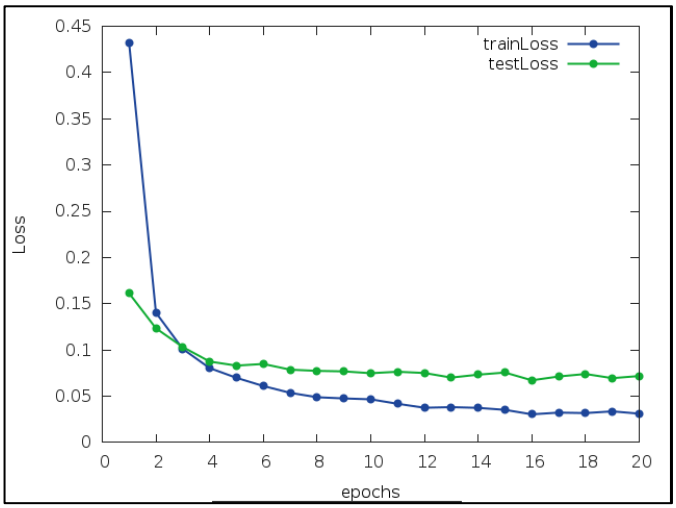
לכן, חזרנו לנסות לשנות את מספר השכבות ואת גודל כל שכבה:

הוקטור layerSize	מספר הפרמטרים	epoch אופטימלי	Min test error	Min test loss	Max global correct
`inputSize, 64,64,64,64`	63370	19	0.0231	0.0811	97.6862%
`inputSize, 64,64,128`	**64010**	**16**	**0.0199**	**0.0656**	**98.0068%**

לכן נבחר מודל עם הוקטור הזה ובעל epochs=16.

<u>גרפים:</u>





<u>מסקנות:</u>

המודל הנבחר הינו בעל 4 שכבות כך שמספר הפרמטרים הינו 64010. השכבות הן: `inputSize, 64,64,128` .

האימון נעשה עם מקדם למידה: lr=0.2. האימון נעשה עם חלוקה לepochs=16.

לבסוף, שגיאת המבחן שהתקבלה עבור המודל ששמרנו הייתה **0.0212**.

<u>קישור לGitHub:</u>

https://github.com/hanama/Deep_Learning