הדר רוזנוולד 302905187
שירה הראל 302762075
עומר עמית 038217105

תרגיל בית 2 – נושאים מתקדמים במערכות מידע

הקבצים הרלוונטים נמצאים ב: https://github.com/shadaryr/hw2 /

1. **מספר פרמטרים**

מספר הפרטרים הראשוני היה יותר מ-170000. ניגשנו קודם לבעיית הפרמטרים, התחלנו בשינוי בשכבה הלינארית (FC) – את שכבת ה-output הורדנו מ-256 ל-64, כאשר נצפתה הרעה של פחות מ-2% ב-global correct – פגיעה שאנו מוכנים לספוג (בהינתן שיפורים נוספים בהמשך). ירדנו לכ-73000 פרמטרים. החלטנו בנוסף לא לפגוע יותר בשכבה הליארית ולצמצם במימדי הקונבולציות ובמספר שכבות הקונבולוציות. בחנו שינוי את השכבה השניה לקרנל של 5*5 במקום 3*3.
בעקבות השראה מאחד המימושים לדוגמה שהועלו, נצפה כי הורדת השכבה הלינארית אינה פוגמת מאוד בדיוק אך מורידה את כמות הפרמטרים באופן משמעותי. הורדנו את השכבה הלינארית hidden layer על מנת לנסות להפחית את מספר הפרמטרים.

2. **מבנה הרשת**

לאחר בחינת מספר קומבינציות של מספר שכבות קונבולוציה ועומקן התקבל השילוב הבאה המניב accuracy ~78% ועומד בדרישות הפרמטרים (~44K):

```
(nn.BatchFlip():float())
(cudnn.SpatialConvolution(3, 32, 5, 5)) --output- 28*28*32
model:add(cudnn.SpatialMaxPooling(2,2,2,2))      -- 14*14*32
model:add(cudnn.ReLU(true))                      -- ReLU activation function
model:add(nn.SpatialBatchNormalization(32))
model:add(cudnn.SpatialConvolution(32, 64, 3, 3)) -- 12*12*64
model:add(cudnn.SpatialMaxPooling(2,2,2,2)) -- 6*6*64
model:add(cudnn.ReLU(true))
model:add(nn.SpatialBatchNormalization(64))
model:add(cudnn.SpatialConvolution(64, 32, 3, 3)) -- =4*4*32
model:add(nn.View(32*4*4):setNumInputDims(3))  into 1D tensor of 32*4*4
model:add(cudnn.ReLU(true))
model:add(nn.Dropout(0.5))                       --Dropout layer with p=0.5
model:add(nn.Linear(512, #classes))      -- (512+1)*10
model:add(nn.LogSoftMax())
```

ניסינו לשנות עוד את הארכיטקטורה של הרשת בתקווה לקבל תוצאות טובות יותר.
תחילה ע"י הורדת שכבת הfully connected האחרונה. לצורך זה, הoutput של view צריך להיות 1*1*10, ולכן: שכבת הקונבולוציה האחרונה תחזיר מטריצת response בעומק 10, ולאחריה נוסיף שכבת avgpooling להוריד את שטח הפנים ל1*1.
ניתן לראות שכמות הפרמטרים ירדו באופן משמעותי (כ-27 אלף). לכן ניתן כעת לעבות את שכבות הקונבולוציה. שינינו את שכבת הקונבולוציה הראשונה לעומק 64 (כמו השניה). מספר הפרמטרים – 47818.
שינוי זה הוביל לירידה בaccuracy – ~75%.
ננסה שוב עם dropout(0.2) – לא חל שיפור משמעותי.

```
model:add(nn.BatchFlip():float())
model:add(cudnn.SpatialConvolution(3, 64, 5, 5)) -- output- 28*28*64
model:add(cudnn.SpatialMaxPooling(2,2,2,2))      -- 14*14*64
model:add(cudnn.ReLU(true))                      -- ReLU activation function
```

הדר רוזנוולד 302905187
שירה הראל 302762075
עומר עמית 038217105

```
model:add(nn.SpatialBatchNormalization(64))
model:add(cudnn.SpatialConvolution(64, 64, 3, 3)) --12*12*64
model:add(cudnn.SpatialMaxPooling(2,2,2,2)) -- 6*6*64
model:add(cudnn.ReLU(true))
model:add(nn.SpatialBatchNormalization(64))
model:add(cudnn.SpatialConvolution(64, 10, 3, 3)) -- 4*4*10
model:add(cudnn.SpatialAveragePooling(4,4,4,4)) --gets 4*4*10, returns 1*1*10
model:add(nn.View(10*1*1):setNumInputDims(3))  -- into 1D tensor of 32*1*1
model:add(cudnn.ReLU(true))
model:add(nn.Dropout(0.2))                --Dropout layer with p=0.2
model:add(nn.LogSoftMax())                -- converts the output to a log-probability.
```

ביצענו שינויים ברשת למבנה הבא (ללא data augmention עם השראה מ-nin).

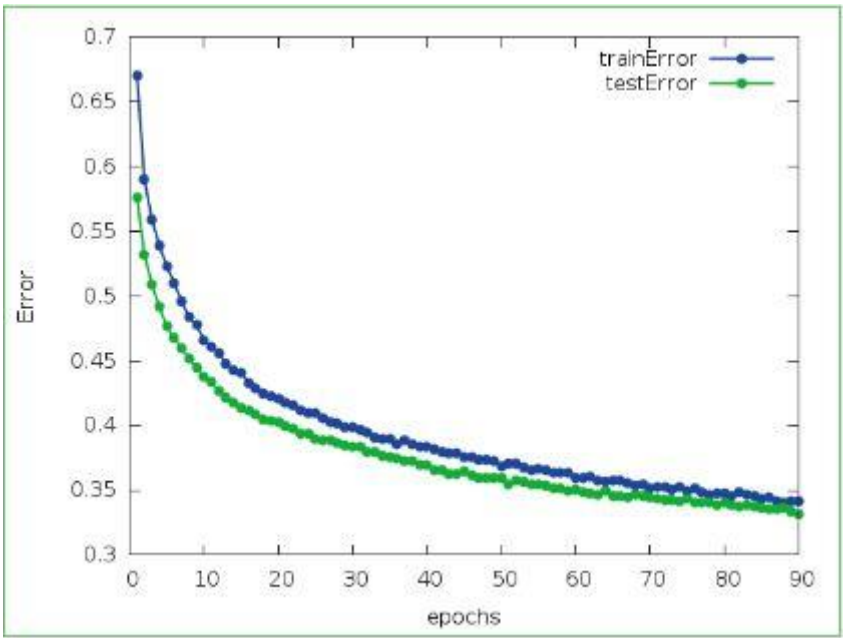
```
local model = nn.Sequential()
model:add(cudnn.SpatialConvolution(3, 64, 5, 5, 1, 1, 2, 2))
model:add(nn.SpatialBatchNormalization(64))    --Batch normalization will provide quicker
convergence
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialConvolution(64, 32, 1, 1)) --
model:add(nn.SpatialBatchNormalization(32))    --Batch normalization will provide quicker
convergence
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialConvolution(32, 32, 1, 1)) --
model:add(nn.SpatialBatchNormalization(32))    --Batch normalization will provide quicker
convergence
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialMaxPooling(3,3,2,2):ceil()) --
model:add(nn.Dropout(0.2))
model:add(cudnn.SpatialConvolution(32, 32, 5, 5, 1, 1, 2, 2)) --
model:add(nn.SpatialBatchNormalization(32))    --Batch normalization will provide quicker
convergence
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialConvolution(32, 32, 1, 1)) --
model:add(nn.SpatialBatchNormalization(32))    --Batch normalization will provide quicker
convergence
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialConvolution(32, 32, 1, 1)) --
model:add(nn.SpatialBatchNormalization(32))    --Batch normalization will provide quicker
convergence
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialAveragePooling(3,3,2,2):ceil()) --
model:add(nn.Dropout(0.2))
model:add(cudnn.SpatialConvolution(32, 32, 3, 3, 1, 1, 1, 1)) --
model:add(nn.SpatialBatchNormalization(32))    --Batch normalization will provide quicker
convergence
```

הדר רוזנוולד 302905187
שירה הראל 302762075
עומר עמית 038217105
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialConvolution(32, 32, 1, 1)) --
model:add(nn.SpatialBatchNormalization(32)) --Batch normalization will provide quicker convergence
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialConvolution(32, 10, 1, 1)) --
model:add(nn.SpatialBatchNormalization(10)) --Batch normalization will provide quicker convergence
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialAveragePooling(8,8,1,1):ceil()) --
model:add(nn.View(#classes))

מדובר ברשת של רשתות המורכבת מ-blocks של קונבולציות, פונקיות טרנספורמציה ו-pooling.
ניתן לראות ברשת זו שימוש הן ב-average pooling והן ב-max pooling. בנוסף, נעשה שימוש בגורם רגולריזציה בדמות dropout עם פרמטר 0.2.
ארכיטקטורה זו הניבה את הביצועים הטובים ביותר ואותה בחרנו כמודל הרשת הסופי.

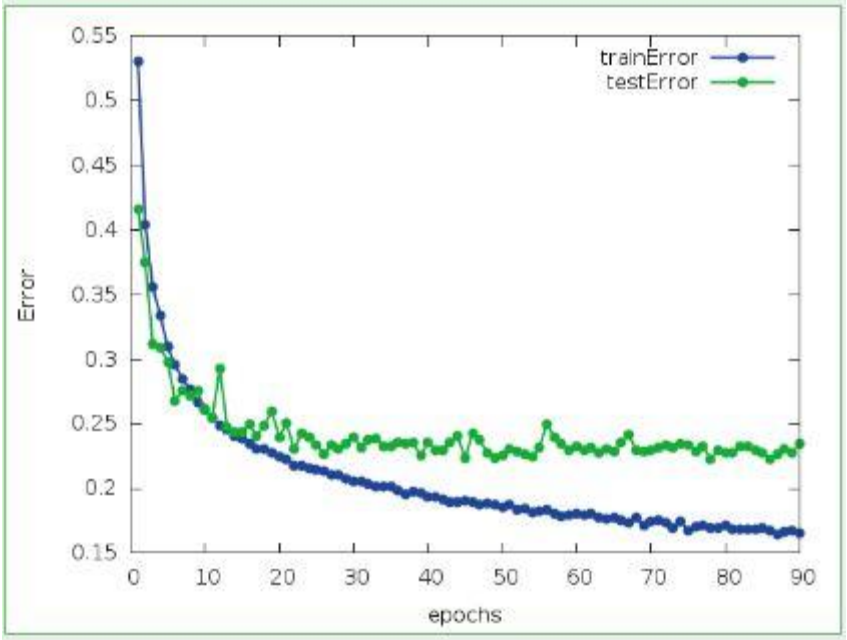
3. **פונקציית Loss ואופטימיזציה:**
ראשית נעשה שימוש בקריטריון NLL, אך לאחר בדיקות התגלה כי קריטריון Cross Entropy מניב את התוצאות הטובות ביותר, לכל הפחות accuracy=80%.
פונקציות אופטימיזציה – ראשית נבדקה פונק' adam ובבדיקה נראה כי פונ' SGD נותנת תוצאות מוצלחות יותר (כ-82% דיוק).

בנוסף, נבדקו האופטימיזציות הבאות:
AdaGrad – רמת הדיוק הגבוהה (בניסוי שלנו) נצפתה ב-epoch ה-90 כאשר הדיוק הוא 66%, אך ניתן לצפות כי רמת הדיוק תלך ותתכנס לערך גבוה יותר בהמשך.



הדר רוזנוולד 302905187
שירה הראל 302762075
עומר עמית 038217105

AdaDelta – רמת הדיוק הגבוהה נצפתה ב-45 epochs והניבה כ-78% דיוק.



4. **Data Augmentation**

באופן כללי, נעשה שימוש בשיטות האוגמנטציה הבאות בצורות שונות.

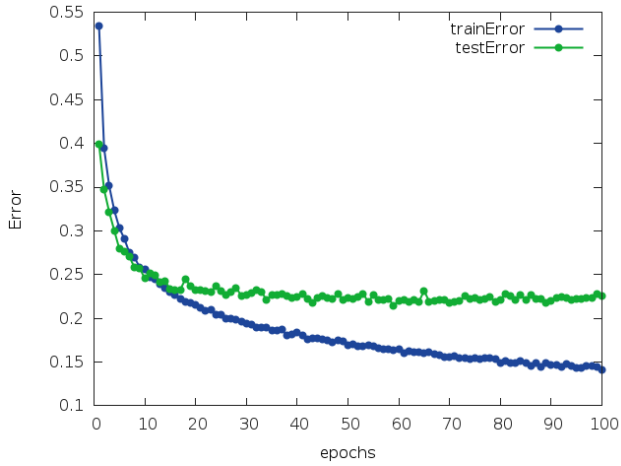
1. hflip
2. random crop – with reflection padding
3. random crop – with zero padding

החלטנו לתקוף את הבעיה ע"י padding נמוך מכפי שהוצג בתרגול. גודל התמונות הוא 32*32, כך ש- padding של 100 פיקסלים, הינו לדעתנו מיותר ולא מועיל למטרה. הורדנו אותו ל-10 פיקסלים.

<u>שיטה 1</u>

החלטנו להגדיל את מספר התמונות עליהן תתאמן הרשת פי 3 ע"י הגרלת מספר ובהתאם למודולו-3 מבצעת סוג מסוים של data augmentation. השיטה הראשונה – hflip, השיטה השניה – random crop with reflection padding. את שתי אלו מבצעת בהסתברות 1/3 ובהסתברות 1/3 לא מבצעת כלל.

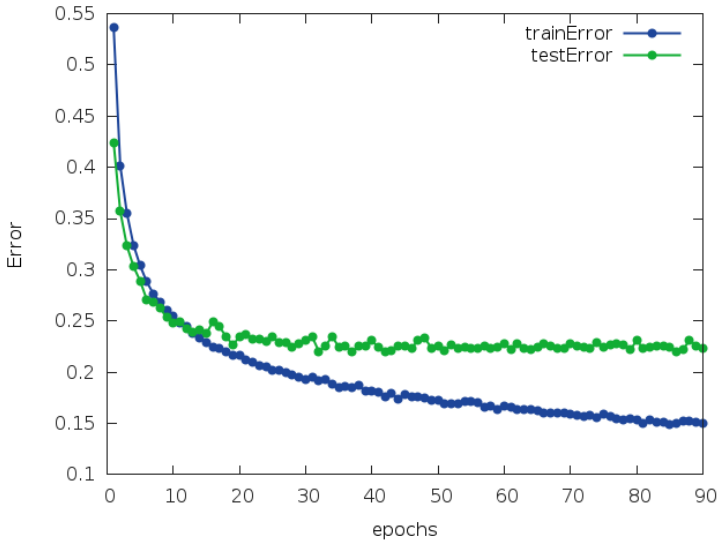
בגרף הבא ניתן לראות כי מספר ה-epochs להתכנסות (בערך 90).



הדר רוזנוולד 302905187
שירה הראל 302762075
עומר עמית 038217105

שיטה 2

נעשה שימוש ב-vflip יחד עם randomcrop במצב reflection. מספר ה-epochs המיטבי היה 75, כאשר רמת ה-accuracy הייתה 77.6%. באופן כללי ניתן לראות שקומבינציה זו נותן תוצאות פחות מוצלחות מאשר הזו הקודמת.



שיטה 3

השתמשנו בכל 3 השיטות הנ"ל:

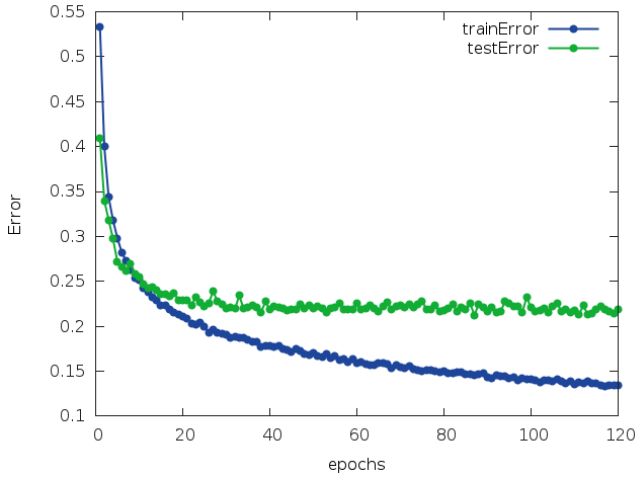
1. hflip
2. reflection random crop
3. zero random crop

כל אחד בהסתברות 1/3.

ה-accuracy הנצפה הוא כ-80%

מספר ה-epochs האופטימלי – 90/110. נבחר ב-110 מטעמי יציבות.

שיטה זו הניבה את אחוז הדיוק הגבוה ביותר ולכן מומשה במודל שלנו.



הדר רוזנוולד 302905187
שירה הראל 302762075
עומר עמית 038217105

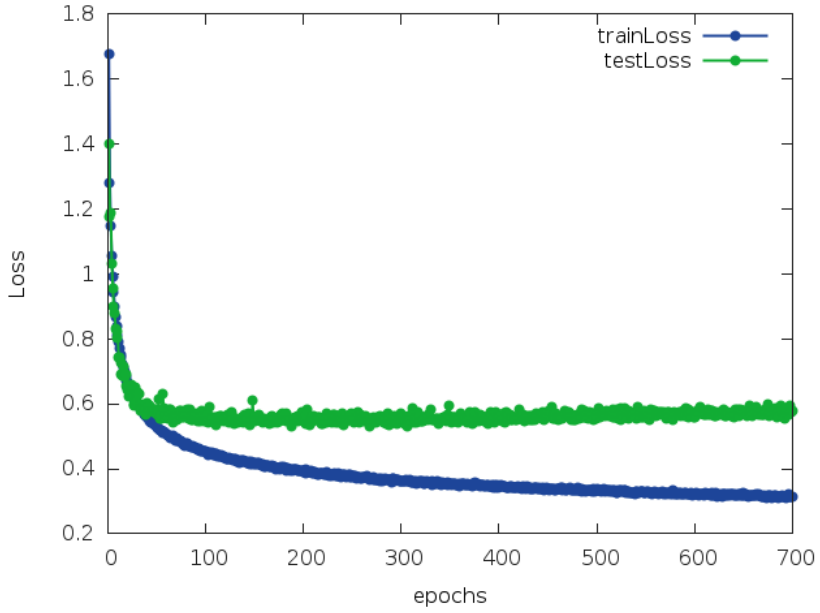
סיכום

המודל הנבחר הינו הרשת הנ"ל, המאופיינת ב:

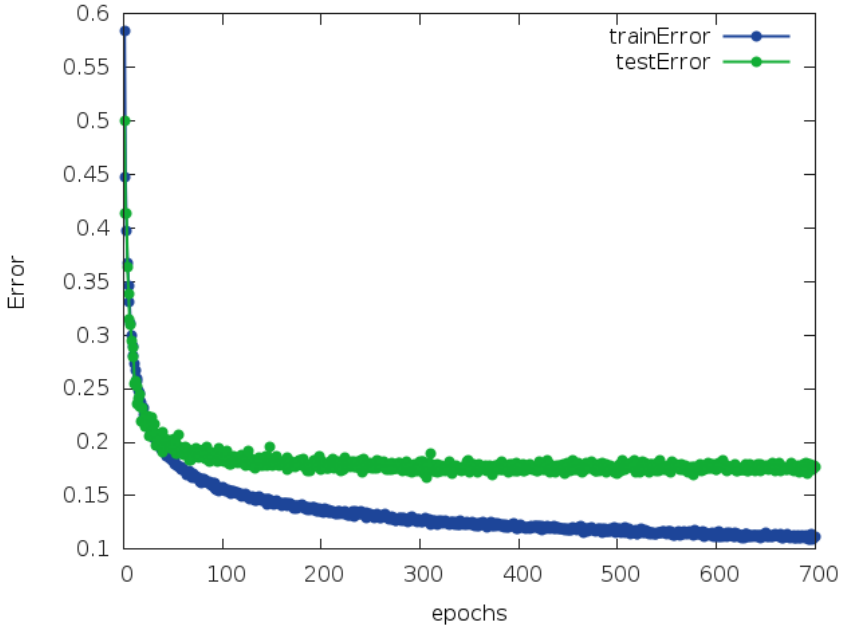
1. פונקצית loss מסוג CrossEntropy
2. אופטימיזציה – SGD
3. batch size = 32
4. ללא **Data Augmentation**
5. Dropout עם פרמטר 0.2
6. התכנסות לרמת דיוק אופטימלית ב-epoch ה-600.

גרפי התכנסות –

Loss:



Error:



הדר רוזנוולד 302905187
שירה הראל 302762075
עומר עמית 038217105

הרשת:

```
local model = nn.Sequential()
model:add(cudnn.SpatialConvolution(3, 64, 5, 5, 1, 1, 2, 2))
model:add(nn.SpatialBatchNormalization(64))    --Batch normalization will provide quicker convergence
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialConvolution(64, 32, 1, 1)) --
model:add(nn.SpatialBatchNormalization(32))    --Batch normalization will provide quicker convergence
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialConvolution(32, 32, 1, 1)) --
model:add(nn.SpatialBatchNormalization(32))    --Batch normalization will provide quicker convergence
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialMaxPooling(3,3,2,2):ceil()) --
model:add(nn.Dropout(0.2))
model:add(cudnn.SpatialConvolution(32, 32, 5, 5, 1, 1, 2, 2)) --
model:add(nn.SpatialBatchNormalization(32))    --Batch normalization will provide quicker convergence
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialConvolution(32, 32, 1, 1)) --
model:add(nn.SpatialBatchNormalization(32))    --Batch normalization will provide quicker convergence
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialConvolution(32, 32, 1, 1)) --
model:add(nn.SpatialBatchNormalization(32))    --Batch normalization will provide quicker convergence
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialAveragePooling(3,3,2,2):ceil()) --
model:add(nn.Dropout(0.2))
model:add(cudnn.SpatialConvolution(32, 32, 3, 3, 1, 1, 1, 1)) --
model:add(nn.SpatialBatchNormalization(32))    --Batch normalization will provide quicker convergence
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialConvolution(32, 32, 1, 1)) --
model:add(nn.SpatialBatchNormalization(32))    --Batch normalization will provide quicker convergence
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialConvolution(32, 10, 1, 1)) --
model:add(nn.SpatialBatchNormalization(10))    --Batch normalization will provide quicker convergence
model:add(cudnn.ReLU(true))
model:add(cudnn.SpatialAveragePooling(8,8,1,1):ceil()) --
model:add(nn.View(#classes))
```