רשתות נוירונים תרגיל 1

אופיר בירקה - 316389410
נריה אוחנה - 304933575

החלק המעשי:

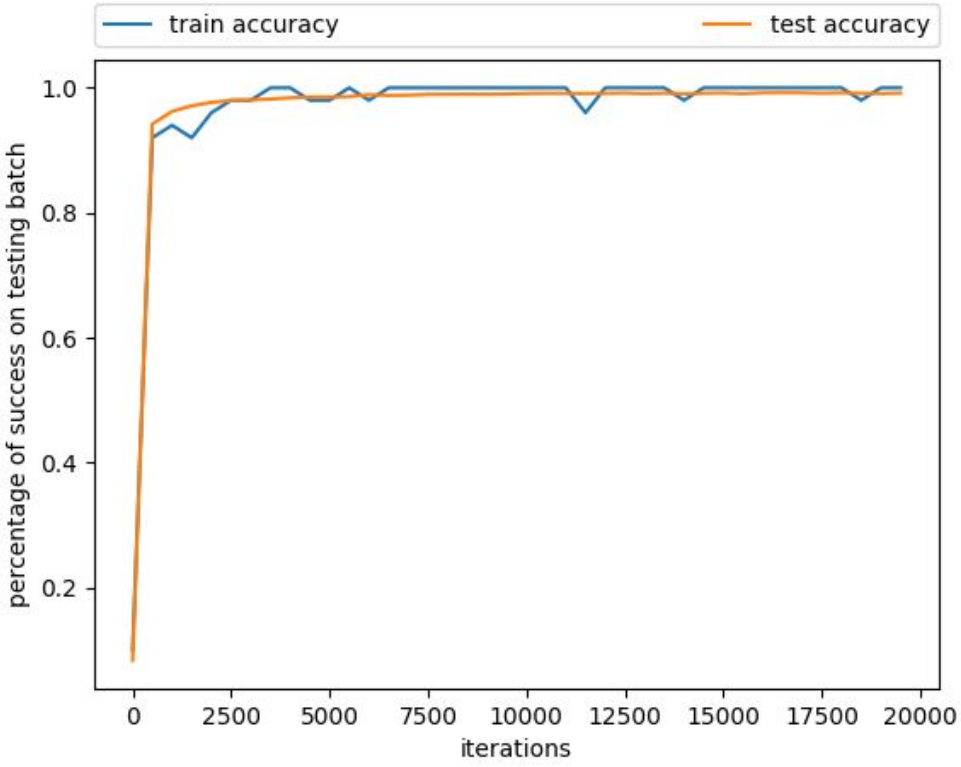
שאלה 1

חישוב מספר המשתנים החופשיים ברשת:

- בשכבת הקונבולוציה הראשונה יש 800= 5*5*1*32 משקלים ועוד 32 של bias . לכן סה"כ 832
- בשכבת הקונבולוציה השניה יש 51200 = 5*5*32*64 משקלים ועוד 64 של bias, סה"כ 51264
- בשכבת FC הראשונה יש 3211264= 7*7*64*1024 ועוד 1024 של bias. סה"כ 3212288
- בשכבת FC השניה יש 10240=10*1024 ועוד 10 של bias. סה"כ 10250

בסך הכל קיבלנו 832+51264+3212288+10250 =3274634

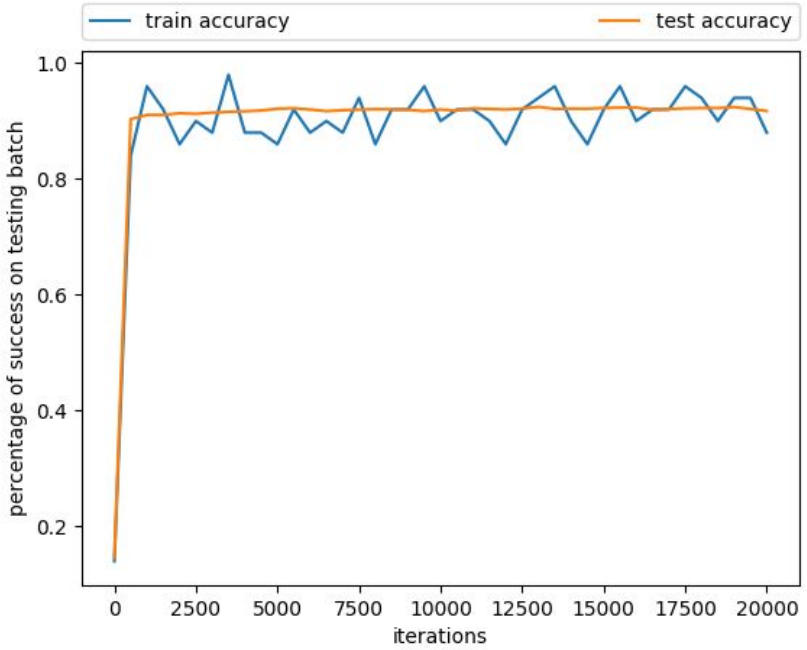
גרף של הדיוק ב train וב test:



שאלה 2

לאחר השינויים שעשינו ברשת כדי להפוך אותה ללינארית יותר- הורדת שכבות ה RELU ומעבר ל average pooling, הדיוק ירד משמעותית והגיע ל 0.917399997413 לאחר 20,000 איטרציות.

אנחנו רואים שהגרף של האימון פחות חלק, אלא מתנהל בקפיצות גדולות. כנראה שהשכבות הלא לינאריות מאפשרות עוד טווח ערכים "חלק" יותר שמאפשר להגיע לערכי ביניים, וכשמורידים אותם הגרף פחות חלק ויש בו תיקונים חדים יותר לשני הצדדים.



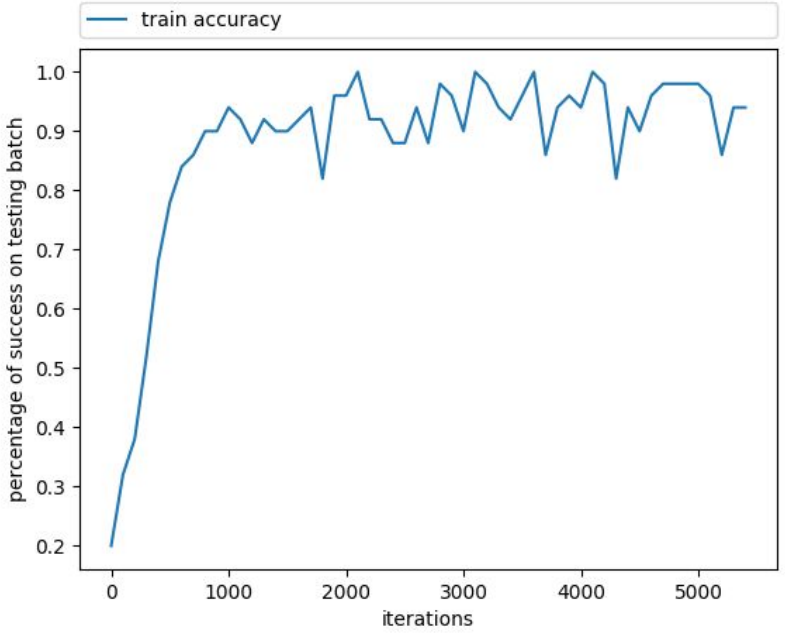
<u>שאלה 3</u>

רשת עם שכבת קונבולוציה אחת ושכבת FC אחת:

- השתמשנו לצורך הקונבולוציה ב 512= 8*8*8 משקלים + 8 בשביל ה bias. לכן סך הכל 520 משתנים.
- בשכבת ה FC השתמשנו ב 62720 משקלים ועוד 10 של bias, וסה"כ 62730.

לכן סה"כ היו ברשת 63250 משתנים חופשיים.

הרשת הגיעה לדיוק של **0.9605**



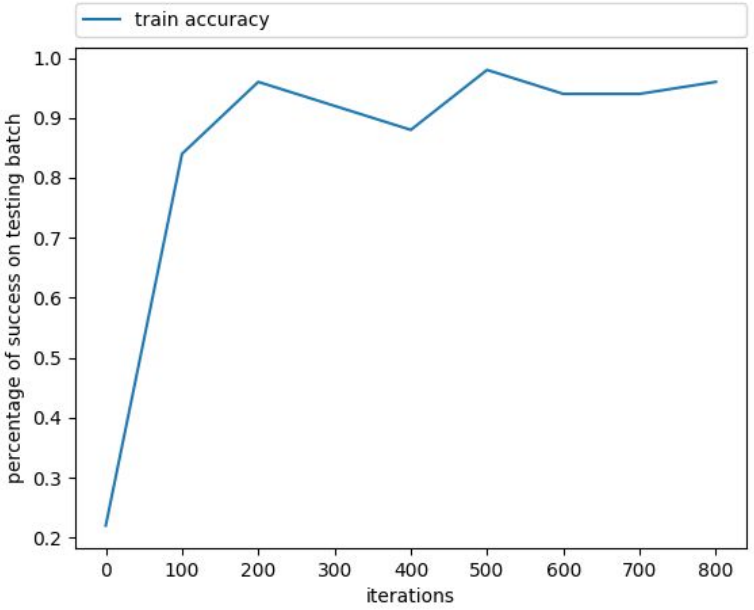
רשת עם מספר מינימלי של פרמטרים:

- שכבת קונבולוציה ראשונה 294= 7*7*16 ועוד 16 של bias נקבל 310
- שכבת קונבולוציה שניה 10400=5*5*16*26 ועוד 26 של bias נקבל 10426
- שכבת FC ראשונה: 1304576=7*7*26*1024 ועוד 1024 של bias נקבל 1305600
- שכבת FC שניה 1024*10 ועוד 10 של bias נקבל 10250

סה"כ קיבלנו ברשת 1326586 פרמטרים חופשיים

הרשת הגיע לדיוק של **0.9586**

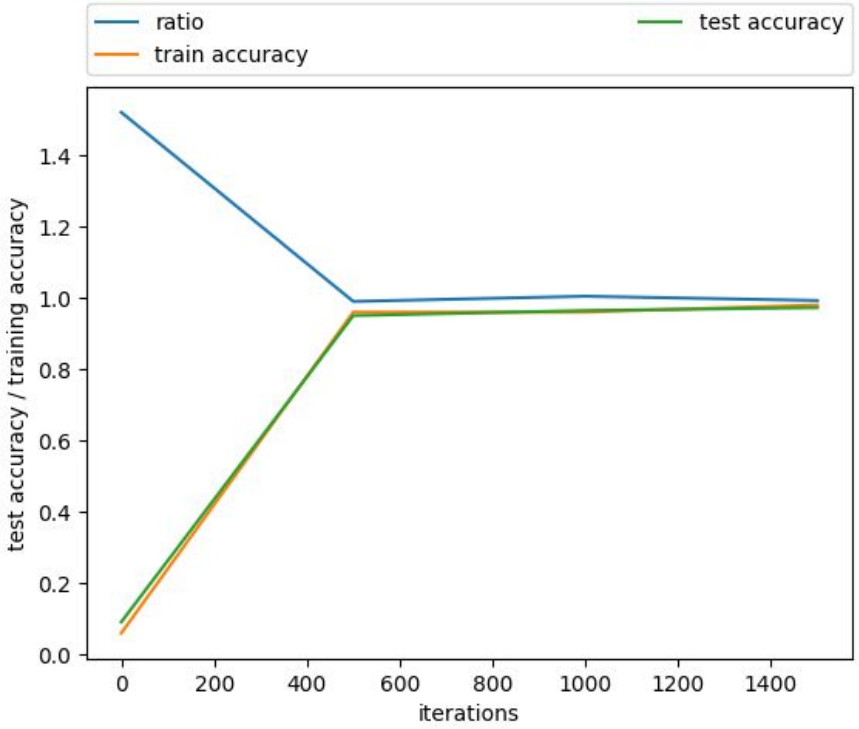
הגרף (בדף הבא):



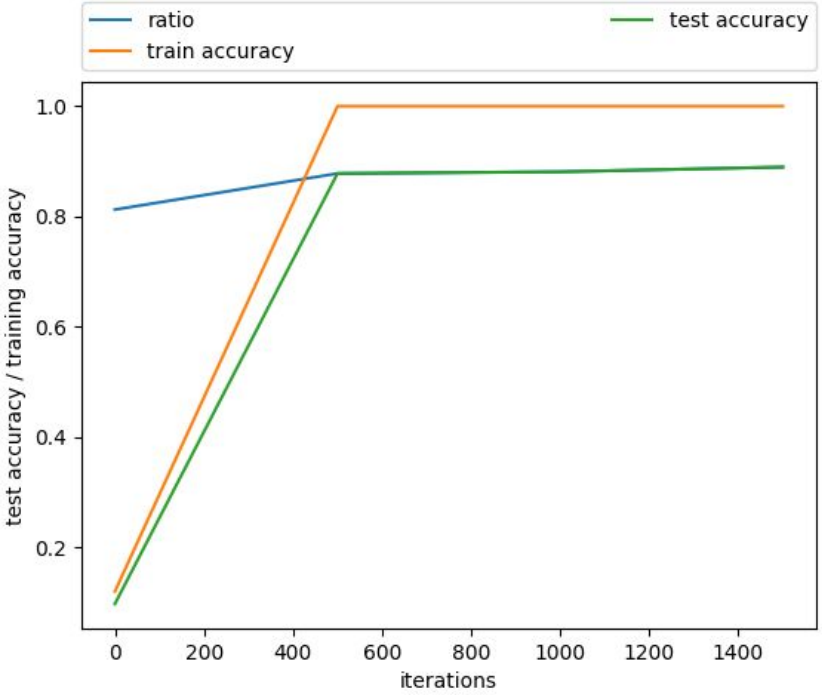
שאלה 4

חלק ראשון:

גרף של אימון על כל ה dataset בלי dropout:



גרף של אימון על 250 דוגמאות בלבד בלי dropout:

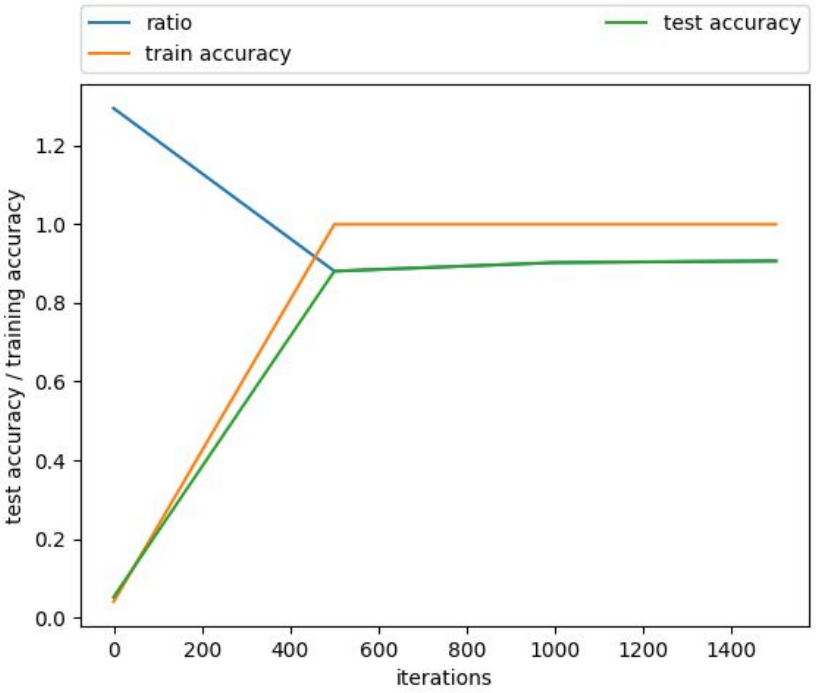


אפשר לראות בצורה ברורה שבגרף הקודם ה train וה test היו דומים לאורך כל האימון, כיוון שאימנו על dataset גדול מספיק הצלחנו ללמוד טוב ולהצליח גם על דוגמאות שלא ראינו.
בגרף השני, מוקדם מאוד ה test מקבל ערכים נמוכים יותר מה train. כלומר בגלל שהיה לו רק 250 דוגמאות אז היה overfitting על הדוגמאות שהוא ראה ועליהם הוא הצליח טוב, אך נכשל על דוגמאות חדשות.
ולכן בעוד שההצלחה ב test כאשר אימנו על כל המדגם הגיעה ל 0.972, כאשר אימנו על 250 דוגמאות הצלחנו הרבה פחות - 0.8895

<u>חלק שני</u>
נראה שניתן לשפר את הדיוק על ידי הפחתת מספר הפרמטרים. הפחתת דרגות החופש של המודל תגביל את ה overfitting, ולכן נצפה לשיפור בתוצאות. ביצענו את השינויים הבאים:

- בשכבה השניה של הקונבולוציה הפחתנו את גודל הפילטר- במקום 5x5 השתמשנו ב 3x3
- בשכבה הראשונה של FC, במקום לעבוד עם וקטור באורך 1024, עבדנו עם וקטור באורך 300

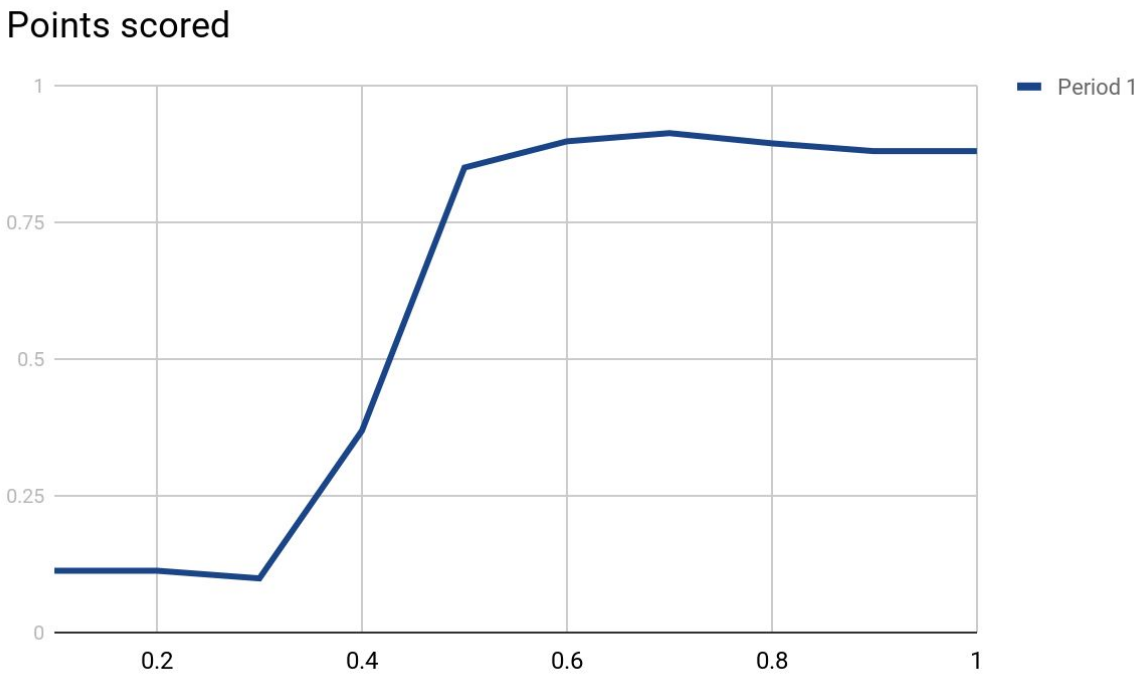
ואכן לאחר הפחתת המשתנים התוצאה השתפרה והגיע ל 0.906 על הטסט הסופי



<u>חלק שלישי</u>

הרצנו מודל עם הוספה של dropout layer בין כל שתי שכבות.

התוצאות השונות עבור אפשרויות שונות של keep_prob:



ניתן לראות שאפשר לשפר את הדיוק על ידי הגדלת ההסתברות ל dropout, וכך להתמודד עם overfitting.

שאלה 5

תיאור הארכיטקטורה:

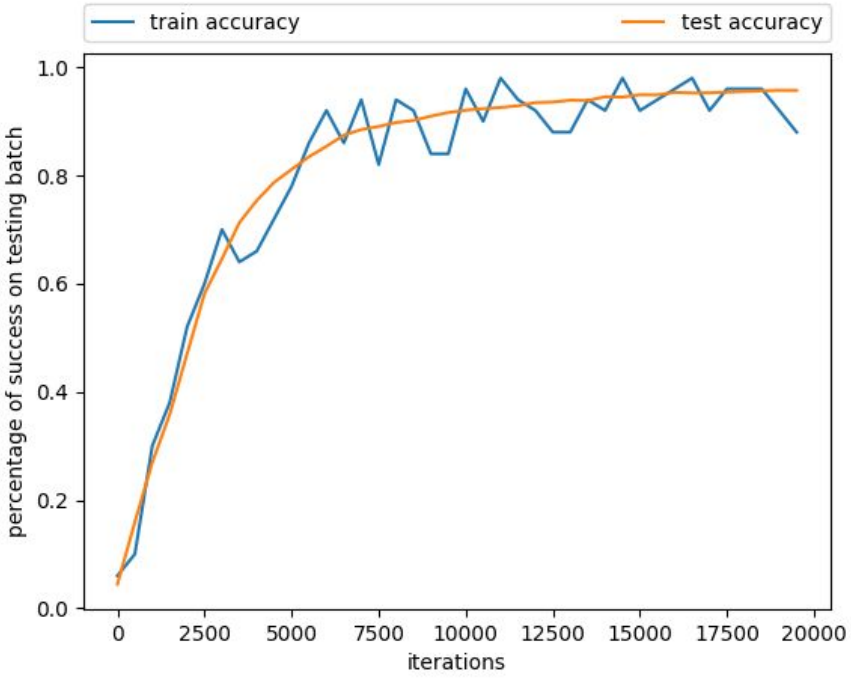
- קלט: הקלט יהיה 2 תמונות של ספרות, מגודל 56x28, כלומר 2 תמונות אחת מעל השניה
- שכבה ראשונה של קונבולוציה עם קרנל מהצורה [5, 5, 1, 32] ואקטיבציה של RELU
- שכבה של 2x2 pooling
- שכבה שניה של קונבולוציה עם קרנל [5, 5, 32, 64] ואקטיבציה של RELU
- שכבה של 2x2 pooling
- שכבה שבה מגדירים משקלים ו bias, ויוצרים וקטור באורך 2048, כך שאותה מטריצה של משקלים מוכפלת פעמיים בקלט: פעם אחת מכפילים את המשקלים בחצי העליון של מטריצת הקלט ומוסיפים את ה bias,וכך מקבלים וקטור באורך 1024 שהוא חצי מוקטור הפלט. ובפעם השניה מכפילים את אותם המשקלים בחצי התחתון ומוסיפים את ה bias כדי לקבל את החצי השני של וקטור הפלט.
- שכבה של FC שמקבלת וקטור באורך 2048 , מכפילה במשקלים ומוסיפה bias, ומוציאה וקטור באורך 19, שזה מספר האפשרויות לסכום של 2 ספרות.

את הרשת מאמנים על קלט שמורכב מחיבור של 2 תמונות של ספרות, והפלט הרצוי הוא וקטור שמכיל 1 במקום של הסכום של 2 הספרות, ואפס בשאר המקומות

השתמשנו בארכיטקטורה זו כי הקלט מורכב מ2 ספרות, כך שהגיוני בשלב הראשון להפעיל פעולות דומות על שתיהן ולהשתמש באותם משתנים כדי לבצע עיבוד של התמונה לאיזשהו יצוג שקשור לערך של הספרה. כך עושים את 2 השכבות של הקונבולוציה עם אותם המשקלים, ואחר כך מכפילים את שני החלקים שקיבלנו מהקונבולוציה באותם משקלים כי כיוון ששתיהם ספרות העיבוד צריך להיות דומה.

אחר כך מקבלים וקטור באורך 2048 שמכיל יצוג של שתי ספרות אחת אחרי השניה, ואז אנו מצפים ששכבה אחת של FC תדע ללמוד את הייצוג של שתי הספרות ולחשב מתוכו את הסכום. לכן השכבה מקבלת וקטור באורך 2048 ומחזירה וקטור באורך 19 שבאינדקס של סכום שתי הספרות אמור להיות 1.

הארכיטקטורה אכן מצליחה לעבוד ומגיעה להצלחה של 0.95 . הגרף של האימון:



Theoretical Questions:

December 3, 2017

1 Convolution

We will show that $(f([k+t]*g[k])[n] = (f[k]*g[k])[n+t]$ and $(f[k]*g[k+t])[n] = (f[k]*g[k])[n+t]$
If we move g by t units:$(f[k]*g[k+t])[n] = \sum_{k=-\infty}^{\infty} f[k]\cdot g[n+t-k] = (f[k]*g[k])[n+t]$
And if we move f by t units: $(f([k+t]*g[k])[n] = \sum_{k=-\infty}^{\infty} f[k+t]\cdot g[n-k] = \sum_{k=-\infty} f[k]\cdot g[n+t-k] = (f[k]*g[k])[n+t]$
The convolution operator as a dot product looks like:
$(f*g)[n] = \sum_{k=-\infty}^{\infty} f[k]\cdot g[n-k]$
and translation:
$(f[t]*g)[n] = \sum_{k=-\infty}^{\infty} f[t+k]\cdot g[n-k]$
When convolving a signal of length n with a filter of length $k <= n$ and discarding all the cases where the signal is not fully included in the support of the filter, the length of the output filter is $n-k+1$ because we discarding the cases the filter is not fully included.
When $k = n$ the output will be just one scalar.

2 ReLU

The partial derivative:
Note that Relu(x) max(0,x) => Relu'(x)$\in \{0,1\}$
$\frac{\partial}{\partial f}F(Relu(f*I+b)) = \frac{\partial}{\partial f}F[Relu(f*I+b)]\cdot\frac{\partial}{\partial f}Relu[f*I+b)]\cdot\frac{\partial}{\partial f}(f*I+b) = \frac{\partial}{\partial f}F[0]\cdot\frac{\partial}{\partial f}Relu[<0]\cdot\frac{\partial}{\partial f}(f*I+b) = 0$
$\frac{\partial}{\partial b}F(Relu(f*I+b)) = \frac{\partial}{\partial b}F[Relu(f*I+b)]\cdot\frac{\partial}{\partial b}Relu[f*I+b]\cdot\frac{\partial}{\partial b}(f*I+b) = \vec{0}\cdot\vec{0}\cdot\vec{1} = \vec{0}$
What would happen to these filters and biases in a gradient-descent process consisting solely on examples which produce negative responses?
Nothing, because both of derivatives are zero (so the algorithm thinks it got an optimal point)

3 Backpropagation

Assume we have signal in length of n and we would like to do convolution with filter f in length m.
By definition we deal only the cases when the filter is fully included in summation.
By definition (f*I)(x) = $\sum_{a=1}^{n} f_a I_{x-a}$
We can differentiate by every variables of f, and get $\frac{\partial}{\partial f_i}(f*I)(x) = \frac{\partial}{\partial f_i}\Sigma_{a=1}^{n} f_a I_{x-a} = \Sigma_{a=1}^{n}\frac{\partial}{\partial f_i} f_a I_{x-a} = I_{x-i}$
Therefore we can represent $\frac{\partial}{\partial f}(f*I)(x)$ as a vector contains all sub-derivatives by each f_i: $[\frac{\partial}{\partial f_1}, \frac{\partial}{\partial f_2}, \frac{\partial}{\partial f_3}\cdots\frac{\partial}{\partial f_m}]$
Therefore we get $[I_{x-1}, I_{x-2}, I_{x-3,}\cdots I_{x-m}]$
(f*I)(x) is vector of derivatives for each x (that correspond .to one row in output matrix)
The first element in the matrix is when $x = m+1$ that output $[I_m, I_{m-1,} I_{m-2}\cdots I_1]$
So we get,

$$\begin{matrix} I_m & I_{m-1} & I_{m-2} & \cdots & I_1 \\ I_{m+1} & I_m & I_{m-1} & \cdots & I_2 \\ I_{m+2} & I_{m+1} & I_m & \cdot & \cdot I_3 \\ I_{m+3} & I_{m+2} & I_{m+1} & \cdots & \cdot I_4 \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{matrix}$$

We can see this is a Circulant matrix, that we can multiply efficiently with FFT, therefore we can efficiently multiply the gradient.

4 Parameters vs. Constraints

How many parameters are involved in a convolution layer (+ its biases) consisting of 96, 5 by 5 pixel filters, that operate on RGB image?

$96 \cdot 5 \cdot 5 + 96 \cdot 3 = 1728$

How many parameters are involved in a fully-connected layer (plus its biases) mapping 4-by-4 images of 256 channels (reshaped into a vector) to the same dimension?

$(4 \cdot 4 \cdot 256) \cdot (4 \cdot 4 \cdot 256) + (4 \cdot 4 \cdot 256) = 16781312$

Assuming each loss terms is fulfilled, i.e., rather than minimizing the loss, we set each loss term as an equation (or constraint). How many constraints a dataset of 10000 images with 1000 categories produce?

$10000 * 1000$ (for every image we want 1 in the correct option and 0 in the others, we have 1000 option, multiply by 10000 images)