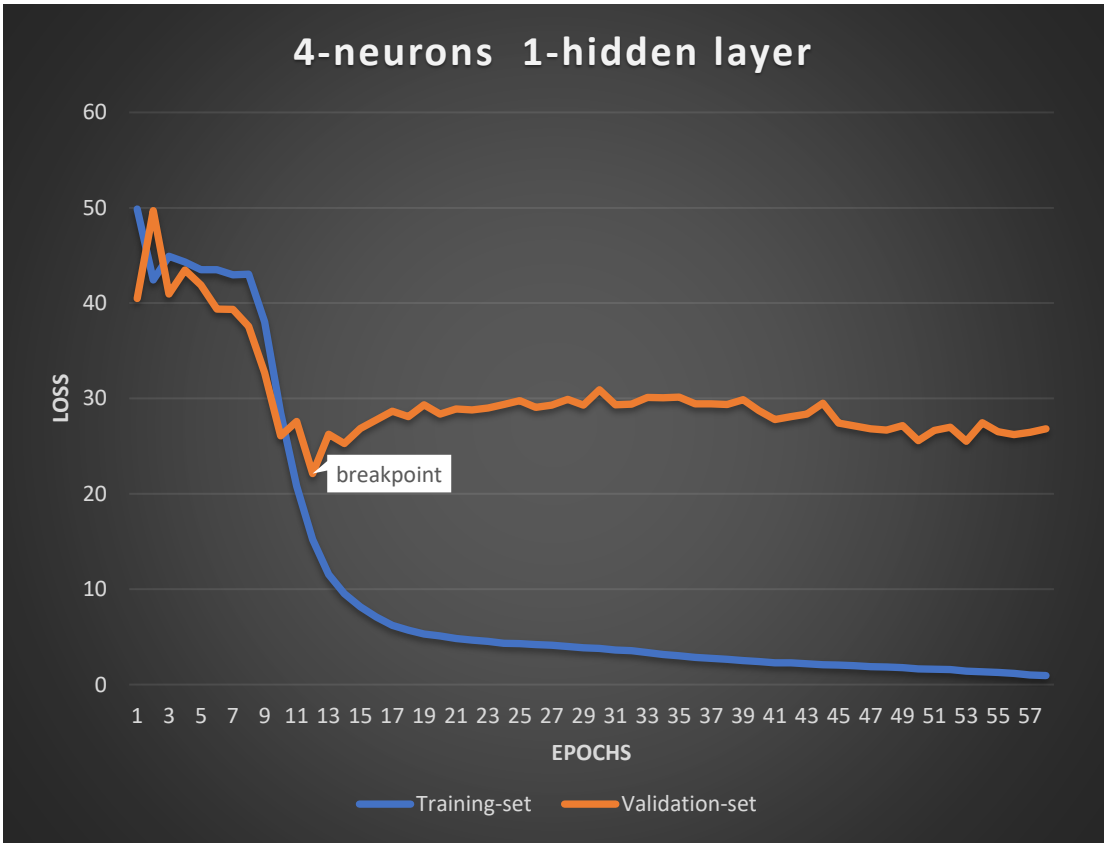
## סעיפי שאלה 1:

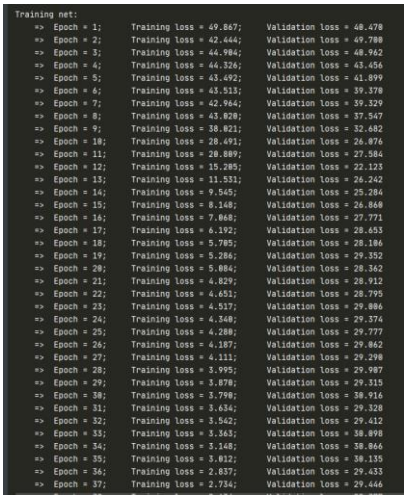
א.

נשווה בין ניתוח השגיאות לאורך הepochs של הtraining-set(אותו הרשת למדה) לניתוח הepochs של הvalidation-set (אותו הרשת לא מכירה).
בסעיף הבא נסיק מסקנות ממנו לגבי overfitting ברשת.

גרף(דיווח הנתונים מופק ישירות לקובץ הנקרא stats, שנמצא בתיקייה גם כן):




(ניתן לעקוב אחרי נתונים גם מהקונסול, אבל ממליץ על קובץ האקסל – השתדלתי להציג שם מגוון רחב של ביצועים בארכיטקטורות ופרמטרים שונים והייצוא של קובץ מעקב חדש ניתן ליישום בהדלקת דגל קטן בלבד!)



```
Training net:
    => Epoch = 1;     Training loss = 49.867;     Validation loss = 48.478
    => Epoch = 2;     Training loss = 42.444;     Validation loss = 49.788
    => Epoch = 3;     Training loss = 44.984;     Validation loss = 48.962
    => Epoch = 4;     Training loss = 44.326;     Validation loss = 43.456
    => Epoch = 5;     Training loss = 43.492;     Validation loss = 41.899
    => Epoch = 6;     Training loss = 43.513;     Validation loss = 39.378
    => Epoch = 7;     Training loss = 42.964;     Validation loss = 39.329
    => Epoch = 8;     Training loss = 43.828;     Validation loss = 37.547
    => Epoch = 9;     Training loss = 38.821;     Validation loss = 32.682
    => Epoch = 10;    Training loss = 28.491;     Validation loss = 26.076
    => Epoch = 11;    Training loss = 20.889;     Validation loss = 27.584
    => Epoch = 12;    Training loss = 15.205;     Validation loss = 22.123
    => Epoch = 13;    Training loss = 11.553;     Validation loss = 26.242
    => Epoch = 14;    Training loss = 9.545;      Validation loss = 25.284
    => Epoch = 15;    Training loss = 8.148;      Validation loss = 26.860
    => Epoch = 16;    Training loss = 7.068;      Validation loss = 27.771
    => Epoch = 17;    Training loss = 6.192;      Validation loss = 28.653
    => Epoch = 18;    Training loss = 5.785;      Validation loss = 28.186
    => Epoch = 19;    Training loss = 5.286;      Validation loss = 29.352
    => Epoch = 20;    Training loss = 5.084;      Validation loss = 28.362
    => Epoch = 21;    Training loss = 4.829;      Validation loss = 28.912
    => Epoch = 22;    Training loss = 4.651;      Validation loss = 28.795
    => Epoch = 23;    Training loss = 4.517;      Validation loss = 29.086
    => Epoch = 24;    Training loss = 4.348;      Validation loss = 29.374
    => Epoch = 25;    Training loss = 4.280;      Validation loss = 29.777
    => Epoch = 26;    Training loss = 4.187;      Validation loss = 29.862
    => Epoch = 27;    Training loss = 4.111;      Validation loss = 29.298
    => Epoch = 28;    Training loss = 3.995;      Validation loss = 29.987
    => Epoch = 29;    Training loss = 3.878;      Validation loss = 29.315
    => Epoch = 30;    Training loss = 3.798;      Validation loss = 30.916
    => Epoch = 31;    Training loss = 3.634;      Validation loss = 29.328
    => Epoch = 32;    Training loss = 3.542;      Validation loss = 29.412
    => Epoch = 33;    Training loss = 3.363;      Validation loss = 30.898
    => Epoch = 34;    Training loss = 3.148;      Validation loss = 30.866
    => Epoch = 35;    Training loss = 3.012;      Validation loss = 30.135
    => Epoch = 36;    Training loss = 2.837;      Validation loss = 29.433
    => Epoch = 37;    Training loss = 2.734;      Validation loss = 29.446
```

ב.

**בעיית התאמת יתר (Overfitting)**

רשת הנוירונים עלולה **לשנן** במקום **ללמוד** את הכלל שמפריד בין קבוצות הסיווג,
במידה והיא עושה זאת יהיה לה קשה למיין קלטים חדשים בעתיד.

*זיהוי:*

מחלקים את הדוגמאות שברשותנו לקבוצת אימון וקבוצת בדיקה.

- נשים לב ל**פער גדול** בין השגיאה על סט האימון לשגיאה על סט הבדיקה.
- נשים לב למצב בו השגיאה על סט האימון יורדת בעוד שעל סט הבדיקה היא עולה.

*מניעה:*

- ניתן לרשת הרבה מאוד דוגמאות לאימון.
- נבחר ארכיטקטורה מתאימה לבעיה!
- **נעצור את האימון בזמן הנכון**, ספציפית בנקודה בה השגיאה בסט הבדיקות עולה.
- רגולריזציה, הרשת תשלם קנס מורכבות ובכך נגרום לה לחפש פתרונות פשוטים (dropout, augmentation..)

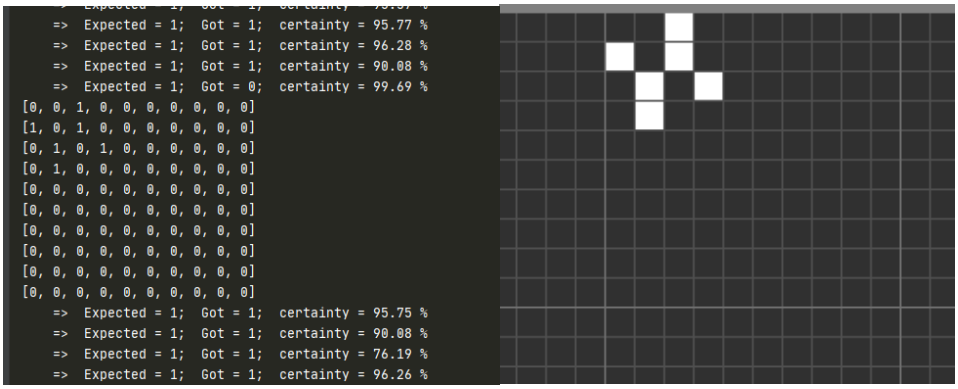
בארכיטקטורה שנבדקה בסעיף א אכן קרתה **בעיית התאמת יתר!**
הנקודה בה השינון מתחיל להשפיע סומנה ב"breakpoint" בגרף שהוצג בסעיף א.
ניתן לראות כי השגיאה על סט האימון ממשיכה לרדת אבל השגיאה על סט הוולידציה מתייצבת ואפילו עולה.
בכל הנוגע למניעת המצב,
פרמטרים רבים נוסו אך זה לא פתר את הבעיה כפי שניתן לראות בתוצאות הריצה בקובץ המצורף.
בחירת רשת אחרת (CNN בעייני) תתאים יותר לאופי הבעיה שמזכיר למידת "תמונות"(דהיינו הקונפיגיורציות).

מומלץ לדעתי לעצור את האימון לאחר מספר מסוים של epochים כפי שעשינו באלגוריתמים גנטיים לאחר דורות רבים ללא שיפור.

ג.

סביר מאוד שטעויות התיוג קורות עקב דאטה בייס קטן יחסית אשר לא מאפשר זיהוי רחב של פרמטרים מאפיינים בעבור צורות החיים, נבחן מספר מקרים כאלו בהרצת בדיקה על מבנה הרשת שנבחר בסעיף א:

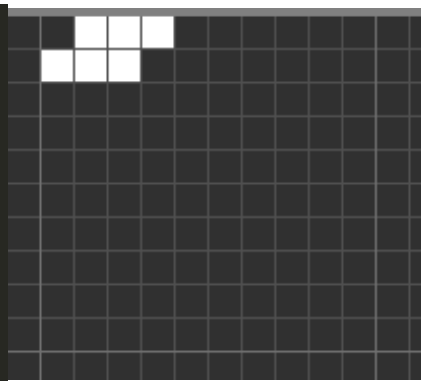
1. מחזורי, תוייג כלא מחזורי.




סיבה משוערת - ניכר כי רוב התיוגים השגויים הם של לא מחזוריים שמתויגים כמחזוריים, זו שגיאה מאזנת מעט שטוב לראות(עד כמה שטוב לראות שגיאות, כאן בכל אופן שמרתי על איזון והצגתי גם וגם).
הצורה ככל הנראה מציגה מאפיינים שמזכירים צורות חיים לא מחזוריות ונראה כי הרשת דיי בטוחה(במעט 100%) שהנוירון הנכון ירה, ייתכן שרוב צורות החיים המחזוריות ששמתי בדאטה סט היו עם 3 ריבועים חיים ברצף למשל וזה מאוד התפספס בעבור הדוגמא הזו. יכולים להיות סיבות רבות נוספות, הרשת לא יכולה להעיד על כללי הלמידה שלנו אז נסיק כמיטב יכולתנו את הסיבות.

2. מחזורי, תוייג כלא מחזורי.

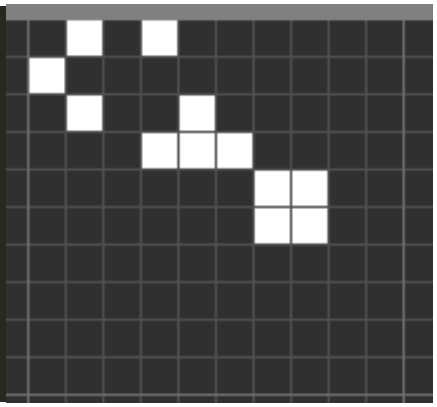
```
    =>  Expected = 1;  Got = 0;  certainty = 99.53 %
[0, 1, 1, 1, 0, 0, 0, 0, 0, 0]
[1, 1, 1, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    =>  Expected = 1;  Got = 1;  certainty = 95.40 %
    =>  Expected = 1;  Got = 1;  certainty = 91.00 %
    =>  Expected = 1;  Got = 1;  certainty = 96.31 %
    =>  Expected = 1;  Got = 1;  certainty = 96.36 %
    =>  Expected = 1;  Got = 1;  certainty = 93.65 %
    =>  Expected = 1;  Got = 1;  certainty = 93.10 %
```



סיבה משוערת – הצורה מאופיינת עם סממנים מחזוריים כמו הריבוע במרכזה אך היא קטנה יחסית ולדעתי רוב הדוגמאות שהצגתי בדאטה סט בעבור מחזוריים היו גדולות יחסית.
זה פרמטר מטעה בעבור דאטה סט קטן מדי.

3. לא מחזורי, תוייג כמחזורי.

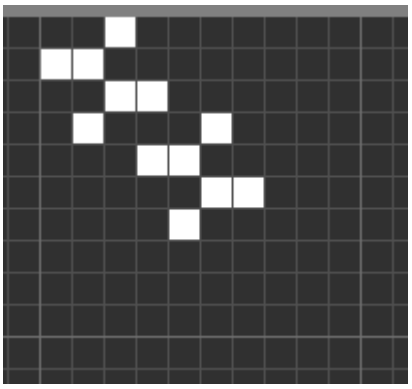
```
    =>  Expected = 1;  Got = 1;  certainty = 96.31 %
    =>  Expected = 1;  Got = 1;  certainty = 96.36 %
    =>  Expected = 1;  Got = 1;  certainty = 93.65 %
    =>  Expected = 1;  Got = 1;  certainty = 93.10 %
    =>  Expected = 1;  Got = 1;  certainty = 94.58 %
    =>  Expected = 0;  Got = 1;  certainty = 96.06 %
[0, 1, 0, 1, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 1, 0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 1, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```



סיבה משוערת – ייתכן והריבוע שמזכיר still life הציג מאפיינים של מחזוריים..

4. לא מחזורי, תוייג כמחזורי.

```
    =>  Expected = 0;  Got = 0;  certainty = 99.70 %
    =>  Expected = 0;  Got = 0;  certainty = 89.41 %
    =>  Expected = 0;  Got = 0;  certainty = 98.40 %
    =>  Expected = 0;  Got = 0;  certainty = 99.24 %
    =>  Expected = 0;  Got = 1;  certainty = 63.58 %
[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 1, 0, 0, 0, 0, 0, 0]
[0, 1, 0, 0, 0, 1, 0, 0, 0, 0]
[0, 0, 0, 1, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1, 1, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```



סיבה משוערת – לשמחתנו נראה שהנוירון לא ירה במלוא העוצמה בעבור התיוג הזה! הוא היסס 😊 מאמין שגודלה וצורתה מפוזרת של צורת החיים הזו מזכירה יותר את הדאטה של המחזוריים.
ניכר כי צריך דאטה סט רחב יותר כדי להבדיל בין פרמטרים בצורה מדויקת יותר.

ד.

התהליך מציג היעדר אופטימיזציה כלשהי בלמידת הרשת(והיעדר למידה למעשה).
כל הבדיקות(על סט האימון, הווליד‌ציה ושאר הדאטה בייס) מציגות הצלחה של 50% בדיוק.
ההצלחה של 50% זיהוי בתיוג נובעת מחלוקה התחלתית של הדאטה בייס לחצי מחזוריים, וחצי לא מחזוריים בדיוק.
כעת, רק נוירון אחד יורה מבין שני נוירוני הפלט תמיד – ומחצית מהדאטה ששייכת לתיוג הזה נתפסת כ"הצלחה" טריוויאלית.

מכיוון שהנגזרת תמיד מתאפסת, השגיאה בכל איטרציה של אימון נשארת באותו ערך, המשקולות לא משתנות (אין צעדים כלל לעבר מינימום מקומי).

מצורף קובץ אקסל מיוצא בשם stats_binarActivation להתרשמות מנתונים, כמו כן ניתן לראות את הדפסת הריצה בשורת הקונסול כמצורף בצילום מסך למטה.

```
Feed forward training_set on the net:
    =>  success: 40/80
    =>  success rate:  50.00 %
    =>  avrage confidence rate:  100.00 %

Feed forward validation_set on the net:
    =>  success: 20/40
    =>  success rate:  50.00 %
    =>  avrage confidence rate:  100.00 %

Feed forward test_set on the net:
    =>  success: 12/24
    =>  success rate:  50.00 %
    =>  avrage confidence rate:  100.00 %

Training net:
    =>  Epoch = 1;      Training loss = 80.000;     Validation loss = 80.000
    =>  Epoch = 2;      Training loss = 80.000;     Validation loss = 80.000
    =>  Epoch = 3;      Training loss = 80.000;     Validation loss = 80.000
    =>  Epoch = 4;      Training loss = 80.000;     Validation loss = 80.000
    =>  Epoch = 5;      Training loss = 80.000;     Validation loss = 80.000
    =>  Epoch = 6;      Training loss = 80.000;     Validation loss = 80.000
    =>  Epoch = 7;      Training loss = 80.000;     Validation loss = 80.000
    =>  Epoch = 8;      Training loss = 80.000;     Validation loss = 80.000
    =>  Epoch = 9;      Training loss = 80.000;     Validation loss = 80.000
    =>  Epoch = 10;     Training loss = 80.000;     Validation loss = 80.000
    =>  Epoch = 11;     Training loss = 80.000;     Validation loss = 80.000
    =>  Epoch = 12;     Training loss = 80.000;     Validation loss = 80.000
    =>  Epoch = 13;     Training loss = 80.000;     Validation loss = 80.000
    =>  Epoch = 14;     Training loss = 80.000;     Validation loss = 80.000
    =>  Epoch = 15;     Training loss = 80.000;     Validation loss = 80.000
    =>  Epoch = 16;     Training loss = 80.000;     Validation loss = 80.000
    =>  Epoch = 17;     Training loss = 80.000;     Validation loss = 80.000
```

ה.

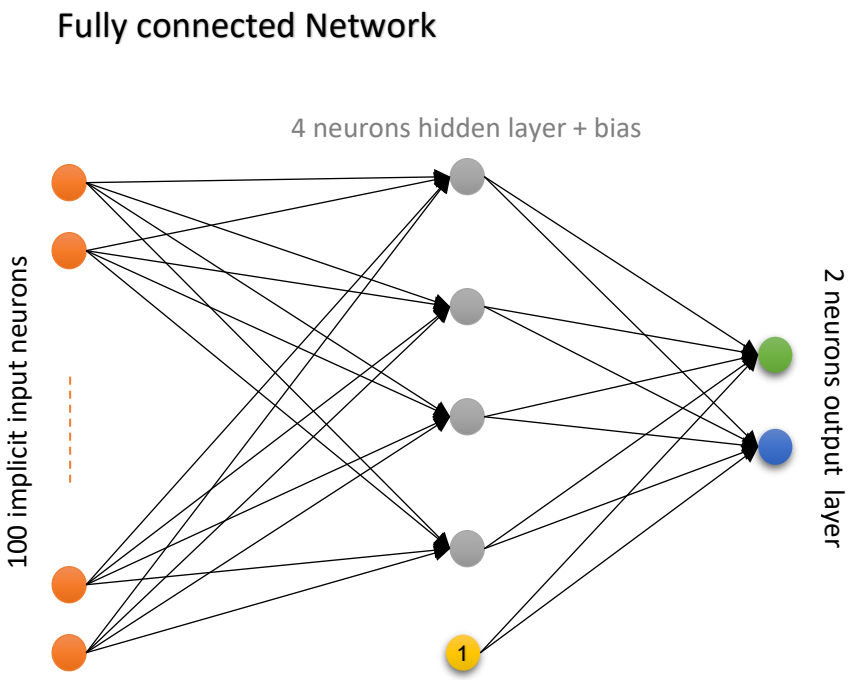
הבעיה העיקרית עם פונקציית האקטיבציה הבינארית באלגוריתם BP היא שהנגזרת מתאפסת.
בBP אנו מעוניינים לבצע דעיכה איטית(gradient descent) לעבר מינימום מסוים(עמק) על ידי עדכון המשקולות בצעדים קטנים, מכיוון שהשיפוע/מדרון/נגזרת תמיד 0 בפונקציה הבינארית לא נוכל לבצע התקדמות כלל.
BP למעשה תחייב אותנו בשימוש בפונקציית אקטיבציה דיפרנציאלית, כזו שיש לה נגזרת(שיפוע) שהוא בין 0 ל1 ולא תמיד 0, על מנת שנוכל לעשות צעד (צעד קטן, דעיכה אל עבר המינימום המקומי).

## ארכיטקטורה:

חשבתי להוסיף כאן תיאור קצר של הרשת לקריאות נעימה,
מציין כי בהמשך עורך שינויים בעומק, רוחב ופרמטרי האופטימיזציה השונים.

1. קלט – כל קלט מציג 100 נוירונים מרומזים המייצגים מטריצה 10X10 (בפועל 100 משקולות אל כל נוירון בשכבת הביניים הראשונה).
2. פלט –שכבת הפלט מכילה 2 נוירונים לתיוג מחזורי ולא מחזורי בהתאמה, היורה מביניהם מסמל את התיוג.
3. עומק – שכבה חסויה אחת( משתנה לאורך הבדיקות).
4. רוחב – 4 נוירונים + נוירון ביאס בכל שכבת ביניים (ישתנה גם כן בבדיקות).
5. קישוריות – מלאה.
6. פונקציית אקטיבציה – סיגמויד.
7. פרמטרי אופטימיזציה דיפולטיביים:
   I. גודל קבוצת האימון – 80
   II. גודל קבוצת האימות - 40
   III. פונקציית שגיאה – $\frac{1}{2}\sum(expected_n - output_n)^2$
   IV. גודל הbatch – 20
   V. קצב הלמידה – 0.5

## אילוסטרציה(לשכבות החסויות נוספים עומק וגודל לאורך הבדיקות):



## BP algorithm:

להלן פסואדו-קוד של אלגוריתם הפעפוע, גם זה מוגש כתוספת לשם הנוחות.
(יש תיעוד בקוד, אפשר להתייחס לתוספות כטיוטה! כבר כתבתי אז חשבתי להשאיר כאן בקובץ)

1. בעבור כל שכבה החל מהאחרונה:
   I. אם זו השכבה האחרונה:
      i. בעבור כל נוירון בשכבה:
         1. חשב,
            התוצאה הרצויה בנוירון (1\0) **פחות** הערך הנוכחי שלו.
            ושמור בסט השגיאות.
   II. אם זה לא השכבה האחרונה:
      i. בעבור כל נוירון בשכבה:
         1. בעבור כל נוירון בשכבה הבאה:
            a. חשב,
               משקל הקשת בין הנוירון בשכבה הבאה אל הנוירון בשכבה **כפול**
               הדלתא של הנוירון בשכבה הבאה.
               ושמור בסט השגיאות.
   III. בעבור כל נוירון:
      i. חשב,
         ערך השגיאה של הנוירון(חושב בשלבים 1.i 1.iii) **כפול** נגזרת סיגמויד של ערך הנוירון.
         ושמור כערך הדלתא של הנוירון.