**ניתוח ועיבוד מידע ספרתי – פרויקט סיום**

**מבוא**

* פרויקט זה מבוסס על אתגר עיבוד תמונה של גוגל – Google Landmark Retrieval Challenge.
* הפרויקט עוסק בבעיה יסודית וחשובה בתום הראיה הממוחשבת – בהינתן שאילתא כלשהי, האם נוכל למצוא את כל הפריטים הדומים לשאילתא בבסיס נתונים כלשהו?
* במקרה הפרטני שלנו השאילתא הינה תמונת נוף (landmark), ועלינו למצוא בבסיס הנתונים את כל התמונות המציגות את אותו נוף.
* בהינתן תמונת שאילתא – אחד מ-100 סימני נוף, עלינו לתת פלט שהוא 10 התמונות המתאימות ביותר, מסודרות לפי סדר הוודאות.

**חלק מעשי**

במסגרת העבודה על הפרויקט ביצענו מספר ניסויים, מרביתם מבוססים על שימוש ברשתות קיימות, אחד הניסויים הינו רשת שאנו בנינו בעצמנו.

נצרף כעת דו"ח מפורט על כל אחד מהניסויים. כל דו"ח כולל בתוכו:

* תוצאות הניסוי
* קונפיגורציות שונות שבחרנו ומבנה הרשת שהשתמשנו בה
* ביצועי מערכת

ניסוי 1:

בניסוי זה הסתמכנו על הרשת הקיימת VGG16. ייבאנו אותה מספריית Keras.

הוספנו לרשת זו את השכבות הבאות:

* שכבת GlobalAvaragePooling2D - השכבה המתנהגת כמו flatten ומיוחדת בכך שמקטינה את מספר הפרמטרים ע"י ממוצע של כל הערכים, ובכך מונעת over fitting.
* שכבת Dense – 1024 פיצ'רים, עם פונקציות אקטיבציה של relu - שכבה זו היא שכבת fully connected והיא האחרונה במקרה הזה שמגיעה לפני הסיווג הסופי.
* שכבת Dense בגודל 20 (ככמות המחלקות) עם פונקציית אקטיבציה softmax - שכבה זו היא השכבה האחרונה שלאחריה מתקבל סיווג של האובייקט לאחת מתוך 20 מחלקות. פונקציית האקטיבציה softmax מבצעת נורמליזציה של כל הערכים כך שיהיו בין 0 ל-1. פונקציית אקטיבציה זו מתאימה במיוחד לשכבה האחרונה, כדי שטווח הערכים יהיה אחיד בין כל המחלקות.

הרצנו את הרשת עם פונקציית loss של categorical\_crossentropy (מיועדת לסיווג בין מספר מחלקות), עם learning rate = 0.01, momentum=0.7.

הרצנו את הרשת בשלושה epochs.

batch size = 10.

כמות התמונות הכוללת בסט האימון היא 1607. התמונות מחולקות ל20 מחלוקות- בסך הכל כ80 תמונות בכל מחלקה.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

input\_1 (InputLayer) (None, 128, 128, 3) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block1\_conv1 (Conv2D) (None, 128, 128, 64) 1792

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block1\_conv2 (Conv2D) (None, 128, 128, 64) 36928

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block1\_pool (MaxPooling2D) (None, 64, 64, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block2\_conv1 (Conv2D) (None, 64, 64, 128) 73856

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block2\_conv2 (Conv2D) (None, 64, 64, 128) 147584

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block2\_pool (MaxPooling2D) (None, 32, 32, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block3\_conv1 (Conv2D) (None, 32, 32, 256) 295168

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block3\_conv2 (Conv2D) (None, 32, 32, 256) 590080

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block3\_conv3 (Conv2D) (None, 32, 32, 256) 590080

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block3\_pool (MaxPooling2D) (None, 16, 16, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block4\_conv1 (Conv2D) (None, 16, 16, 512) 1180160

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block4\_conv2 (Conv2D) (None, 16, 16, 512) 2359808

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block4\_conv3 (Conv2D) (None, 16, 16, 512) 2359808

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block4\_pool (MaxPooling2D) (None, 8, 8, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block5\_conv1 (Conv2D) (None, 8, 8, 512) 2359808

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block5\_conv2 (Conv2D) (None, 8, 8, 512) 2359808

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block5\_conv3 (Conv2D) (None, 8, 8, 512) 2359808

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

block5\_pool (MaxPooling2D) (None, 4, 4, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

global\_average\_pooling2d\_1 ( (None, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**dense\_2 (Dense) (None, 1024) 525312**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**dense\_3 (Dense) (None, 20) 20500**

=================================================================

Total params: 15,260,500

Trainable params: 2,905,620

Non-trainable params: 12,354,880

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

תוצאות האפוקים במהלך האימון:

Epoch 1/3

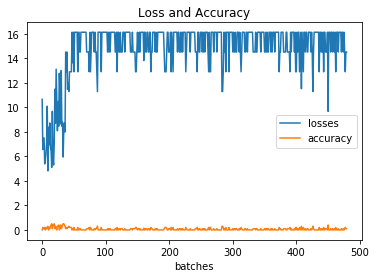
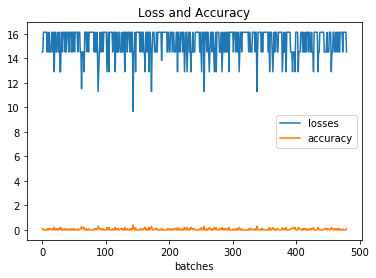
160/160 [==============================] - 201s 1s/step - loss: 13.5460 - acc: 0.1000 - val\_loss: 15.4236 - val\_acc: 0.0431

Epoch 2/3

160/160 [==============================] - 200s 1s/step - loss: 15.3079 - acc: 0.0503 - val\_loss: 15.4236 - val\_acc: 0.0431

Epoch 3/3

160/160 [==============================] - 202s 1s/step - loss: 15.2935 - acc: 0.0512 - val\_loss: 15.4236 - val\_acc: 0.0431



Validation Run

Training Run

מסקנות-

ניתן לראות כי הרשת שהורצה לא הצליחה בסיווג התמונות.

קיבלנו accuracy השואף ל-0, ו- loss מאוד גבוה. הרשת לא הראתה שיפור בין הepochים, ואף גרוע מכך.

אנחנו לא בטוחים מדוע הרשת הניבה תוצאות כה גרועות, אך אנחנו משערים שהרשת התנהגה בצורה הזו מאחר שה- learning rate שהגדרנו היה גבוה מידי. בנוסף כמות התמונות הייתה גבוהה מידי, ומספר הepochים היה נמוך מידי. אבל הכי חשוב- היו לנו רק 2 שכבות fully connected, לכן החישובים שיכולים היו להתבצע הם לינאריים בלבד ולא מספיק מורכבים.

ניסוי 2

בניסוי הזה רשת הבסיס שלנו היא 50resnet. ייבאנו אותה מספריית Keras.

הוספנו לרשת זו את השכבות הבאות:

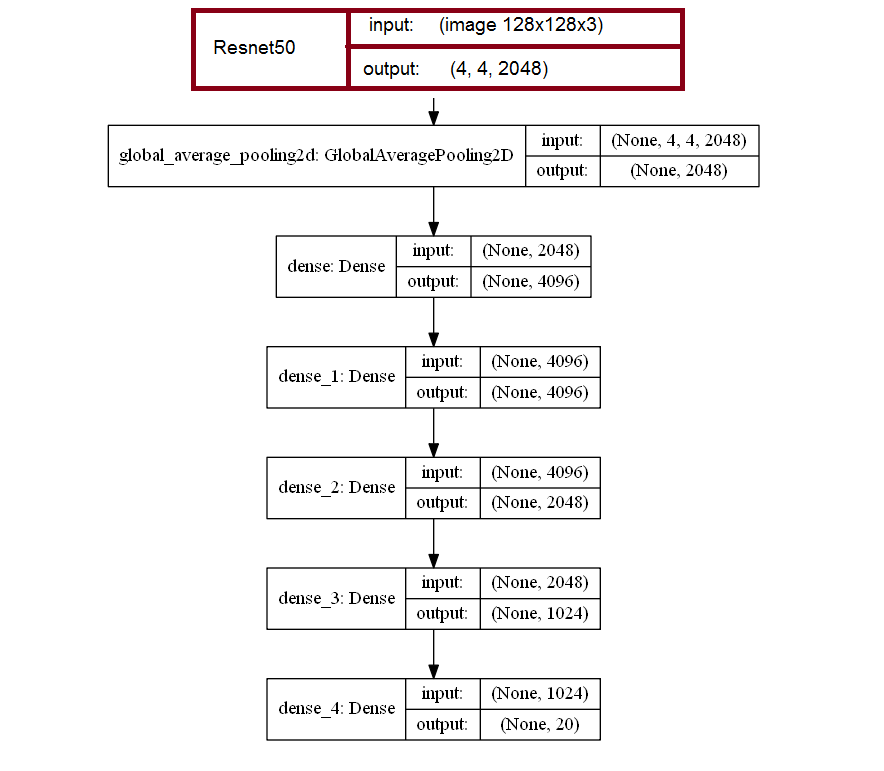
* שכבת GlobalAvaragePooling2D - השכבה המתנהגת כמו flatten ומיוחדת בכך שמקטינה את מספר הפרמטרים ע"י ממוצע של כל הערכים, ובכך מונעת over fitting.
* שכבת Dense – בגודל 4096, עם פונקציות אקטיבציה של relu
* שכבת Dense – בגודל 4096, עם פונקציות אקטיבציה של relu
* שכבת Dense – בגודל 2048, עם פונקציות אקטיבציה של relu
* שכבת Dense – בגודל 1024, עם פונקציות אקטיבציה של relu
* שכבת Dense בגודל 20 (ככמות המחלקות) עם פונקציית אקטיבציה softmax - שכבה זו היא השכבה האחרונה שלאחריה מתקבל סיווג של האובייקט לאחת מתוך 20 מחלקות. פונקציית האקטיבציה

הרצנו את הרשת עם פונקציית loss של categorical\_crossentropy (מיועדת לסיווג בין מספר מחלקות), עם learning rate = 0.03, momentum=0.99.

הרצנו את הרשת ב-6 epochs.

batch size = 10.

כמות התמונות הכוללת בסט האימון היא 1607. התמונות מחולקות ל20 מחלוקות- בסך הכל כ80 תמונות בכל מחלקה.



מהלך הלמידה:

Epoch 1/6

160/160 [==============================] - 160s 1s/step - loss: 1.7710 - acc: 0.5362 - val\_loss: 1.6481 - val\_acc: 0.5260

Epoch 2/6

160/160 [==============================] - 158s 986ms/step - loss: 0.9490 - acc: 0.7329 - val\_loss: 2.0292 - val\_acc: 0.4722

Epoch 3/6

160/160 [==============================] - 157s 982ms/step - loss: 0.7181 - acc: 0.7897 - val\_loss: 2.1007 - val\_acc: 0.5117

Epoch 4/6

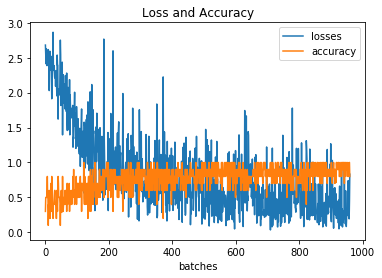
160/160 [==============================] - 157s 983ms/step - loss: 0.5774 - acc: 0.8335 - val\_loss: 1.5426 - val\_acc: 0.6212

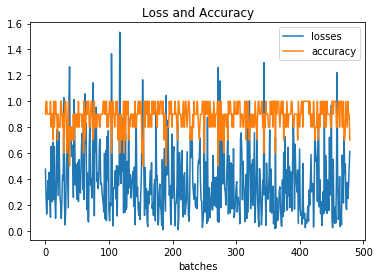
Epoch 5/6

160/160 [==============================] - 158s 990ms/step - loss: 0.5292 - acc: 0.8347 - val\_loss: 2.2132 - val\_acc: 0.5009

Epoch 6/6

160/160 [==============================] - 158s 985ms/step - loss: 0.4450 - acc: 0.8682 - val\_loss: 1.4594 - val\_acc: 0.6571





מסקנות-

בניגוד לניסוי הראשון, הפעם השתמשנו ברשת הרבה יותר עמוקה והרבה יותר חזקה- כך לפי ניסויי ביצועים שעשו בעולם. בנוסף הפעם הוספנו כמות גדולה יותר של שכבות fully connected (במקרה כזה הסיווג אינו לינארי והרבה יותר מורכב) והורדנו את הlearning rate.

ניתן לראות שיש תנודות גבוהות בגרף גם בaccuracy וגם ב-loss. אנחנו משערים שזה נובע מה-momentum הגבוה שבחרנו והlearning rate גבוה מידי גם הוא.

ניסוי 3

הרשת הזו לא מתבססת על רשת קיימת- בנינו אותה מההתחלה ועד הסוף ואימנו אותה לכל אורכה.

הרשת מקבלת תמונה בגודל 128x128. הרעיון הוא שככל שהרשת מתקדמת, שכבות הקונבולוציה מכילות מספר פילטרים גבוה יותר, ובגודל קטן יותר. למשל בשכבה הקונבולוציה הראשונה יש 64 פילטרים בגודל 5 על 5, ובשכבת הקונבולוציה האחרונה יש 1024 פילטרים בגודל 3 על 3.

הרשת מכילה את השכבות מהסוגים הבאים:

1. שכבת קונבולוציה, המקבלת את התמונה בפורמט RGB. השכבה מכילה 64 פילטרים בגודל 5x5 ומשתמשת בפונקציית אקטיבציה relu.
2. שכבת אקטיבציה עם פונקציית relu להסרת ערכים שליליים.
3. שכבת max pooling- מתנהג כמו Pooling והערך שנלקח לשכבה הבאה הוא הערך המקסימלי מבין הערכים בכל פילטר.
4. שכבת קונבולוציה עם 128 פילטרים בגודל 5x5, עם padding
5. שכבת אקטיבציה עם פונקציית relu להסרת ערכים שליליים.
6. שכבת max pooling
7. שכבת zero padding – מטרתה להוסיף שוליים לשכבה הקודמת המלאים בערכי 0 (כמו padding רק עם אפסים). היתרון המרכזי הוא שהשכבה מגדילה את גודל התמונה, מה שמאפשר מאוחר יותר להשתמש ביותר שכבות שגורמות את הקטנת התמונה כמו pooling. בנוסף השכבה גורמת לתהליך העיבוד להיות מהיר יותר (אפסים בFFT חוסך ביצועים).
8. שכבת קונבולוציה עם 128 פילטרים בגודל 3x3, עם padding
9. שכבת אקטיבציה.
10. שכבת batch normalization- מקטין את הערכים בכל המשקולות באופן פרופורציונלי, במטרה להפוך את החישובים למהירים יותר.
11. שכבת max pooling
12. שכבת zero padding
13. שכבת קונבולוציה עם 512 פילטרים בגודל 3x3, עם padding
14. שכבת אקטיבציה.
15. שכבת max pooling
16. שכבת קונבולוציה עם 1024 פילטרים בגודל 3x3, עם padding
17. שכבת batch normalization
18. שכבת אקטיבציה.
19. שכבת max pooling
20. שכבת קונבולוציה עם 2048 פילטרים בגודל 3x3, עם padding
21. שכבת max pooling
22. שכבת GlobalAvaragePooling2D
23. שכבת Dense בגודל 4096
24. שכבת Dropout (P=0.2)– היא עושה נורמליזציה ומפחיתה over fitting ע"י הסרת נוירונים וקשתות באופן אקראי מרשת הנוירונים בגודל P שנקבע. ההסרה מתבצעת באופן אקראי וכך בכל הרצה מתקבלת רשת אחרת, מה שמונע אימון יתר של אותה רשת בדיוק.
25. שכבת Dense בגודל 1024
26. שכבת Dropout (P=0.2)
27. שכבת Dense בגודל 100 (ככמות המחלקות) עם פונקציית אקטיבציה softmax - שכבה זו היא השכבה האחרונה שלאחריה מתקבל סיווג של האובייקט לאחת מתוך 100 מחלקות.

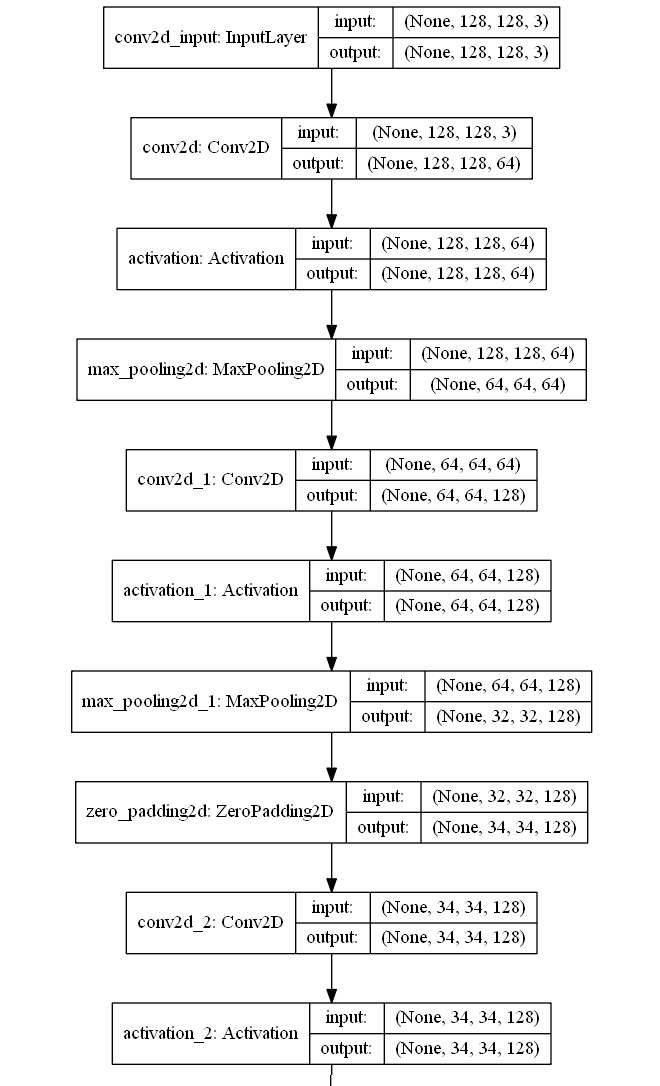
הרצנו את הרשת עם פונקציית loss של categorical\_crossentropy (מיועדת לסיווג בין מספר מחלקות), עם learning rate = 0.001, momentum=0.9.

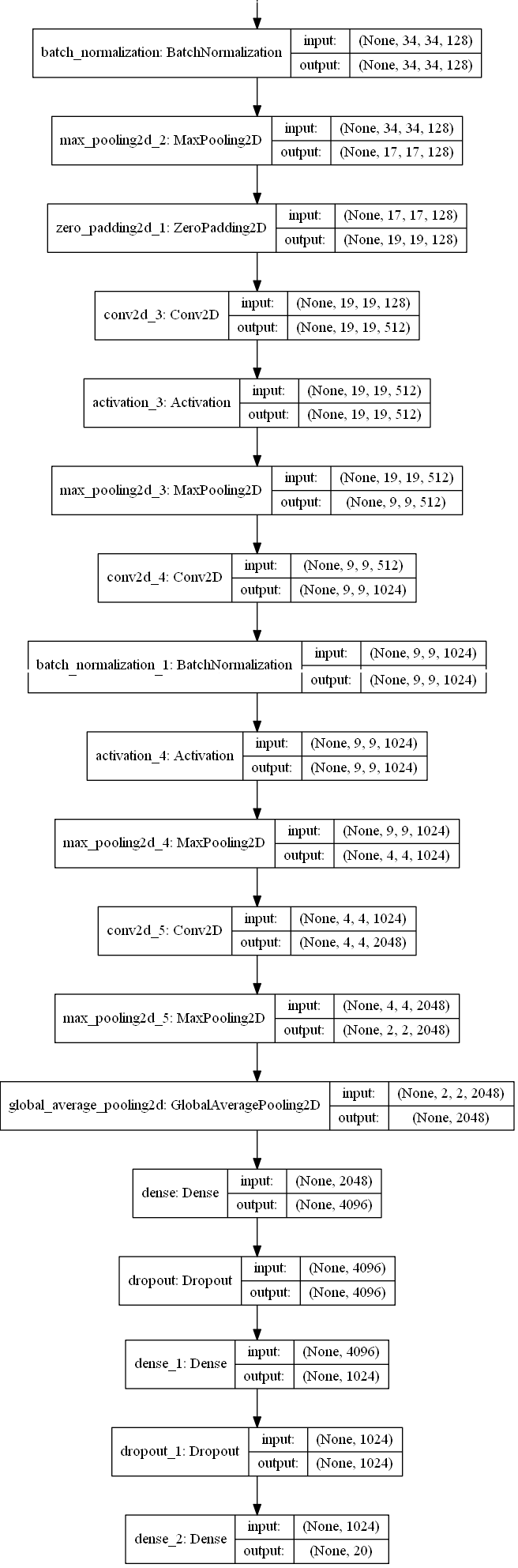
הרצנו את הרשת ב-20 epochs + 3 epochs בשלב הvalidation.

batch size = 10.

כמות התמונות הכוללת בסט האימון היא 1607. התמונות מחולקות ל20 מחלוקות- בסך הכל כ80 תמונות בכל מחלקה.

להלן תרשים המתאר את רשת הנוירונים:





מהלך הלמידה:

Epoch 1/20

160/160 [==============================] - 314s 2s/step - loss: 3.0867 - acc: 0.0603 - val\_loss: 2.9477 - val\_acc: 0.0880

Epoch 2/20

160/160 [==============================] - 312s 2s/step - loss: 2.8549 - acc: 0.1264 - val\_loss: 2.8909 - val\_acc: 0.0880

Epoch 3/20

160/160 [==============================] - 312s 2s/step - loss: 2.6427 - acc: 0.1844 - val\_loss: 3.0241 - val\_acc: 0.0934

Epoch 4/20

160/160 [==============================] - 312s 2s/step - loss: 2.4120 - acc: 0.2437 - val\_loss: 2.6120 - val\_acc: 0.1724

Epoch 5/20

160/160 [==============================] - 312s 2s/step - loss: 2.1750 - acc: 0.3144 - val\_loss: 2.4279 - val\_acc: 0.2783

Epoch 6/20

160/160 [==============================] - 312s 2s/step - loss: 2.0678 - acc: 0.3592 - val\_loss: 5.0671 - val\_acc: 0.1023

Epoch 7/20

160/160 [==============================] - 312s 2s/step - loss: 1.8784 - acc: 0.4033 - val\_loss: 1.7548 - val\_acc: 0.4399

Epoch 8/20

160/160 [==============================] - 312s 2s/step - loss: 1.7111 - acc: 0.4708 - val\_loss: 2.2489 - val\_acc: 0.3178

Epoch 9/20

160/160 [==============================] - 312s 2s/step - loss: 1.5937 - acc: 0.5029 - val\_loss: 2.0782 - val\_acc: 0.3645

Epoch 10/20

160/160 [==============================] - 312s 2s/step - loss: 1.4286 - acc: 0.5454 - val\_loss: 1.9592 - val\_acc: 0.4291

Epoch 11/20

160/160 [==============================] - 312s 2s/step - loss: 1.3927 - acc: 0.5759 - val\_loss: 1.3421 - val\_acc: 0.5727

Epoch 12/20

160/160 [==============================] - 312s 2s/step - loss: 1.2690 - acc: 0.6104 - val\_loss: 1.6668 - val\_acc: 0.5081

Epoch 13/20

160/160 [==============================] - 312s 2s/step - loss: 1.1560 - acc: 0.6372 - val\_loss: 1.5198 - val\_acc: 0.5386

Epoch 14/20

160/160 [==============================] - 312s 2s/step - loss: 1.1036 - acc: 0.6532 - val\_loss: 1.6080 - val\_acc: 0.5242

Epoch 15/20

160/160 [==============================] - 312s 2s/step - loss: 1.0565 - acc: 0.6629 - val\_loss: 1.2592 - val\_acc: 0.6445

Epoch 16/20

160/160 [==============================] - 311s 2s/step - loss: 0.9998 - acc: 0.6961 - val\_loss: 1.6051 - val\_acc: 0.5171

Epoch 17/20

160/160 [==============================] - 312s 2s/step - loss: 1.0063 - acc: 0.6907 - val\_loss: 1.1306 - val\_acc: 0.6463

Epoch 18/20

160/160 [==============================] - 312s 2s/step - loss: 0.9187 - acc: 0.7095 - val\_loss: 3.5872 - val\_acc: 0.2711

Epoch 19/20

160/160 [==============================] - 312s 2s/step - loss: 0.8523 - acc: 0.7210 - val\_loss: 1.7347 - val\_acc: 0.5099

Epoch 20/20

160/160 [==============================] - 311s 2s/step - loss: 0.8213 - acc: 0.7279 - val\_loss: 1.7616 - val\_acc: 0.4919

Epoch 1/3

159/160 [============================>.] - ETA: 1s - loss: 0.8053 - acc: 0.7588

Epoch 00001: val\_acc improved from -inf to 0.34470, saving model to C:\DeepLearning\custom\_1549479754.h5

160/160 [==============================] - 313s 2s/step - loss: 0.8019 - acc: 0.7604 - val\_loss: 2.7498 - val\_acc: 0.3447

Epoch 2/3

159/160 [============================>.] - ETA: 1s - loss: 0.7043 - acc: 0.7777

Epoch 00002: val\_acc improved from 0.34470 to 0.52244, saving model to C:\DeepLearning\custom\_1549479754.h5

160/160 [==============================] - 312s 2s/step - loss: 0.7050 - acc: 0.7772 - val\_loss: 1.6303 - val\_acc: 0.5224

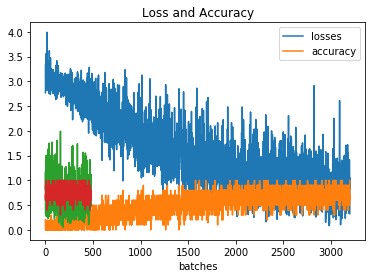
Epoch 3/3

159/160 [============================>.] - ETA: 1s - loss: 0.6615 - acc: 0.7913

Epoch 00003: val\_acc improved from 0.52244 to 0.64991, saving model to C:\DeepLearning\custom\_1549479754.h5

160/160 [==============================] - 312s 2s/step - loss: 0.6608 - acc: 0.7913 - val\_loss: 1.1294 - val\_acc: 0.6499

להלן גרף loss vs accuracy שיצא בעקבות האימון:



מסקנות-

העיקרון המנחה ברשת זו היא העלאת כמות הפילטרים והקטנת גודל הפילטרים משכבה לשכבה, ובכך להתחיל עם פילטרים ברמת "על" וככל שמתקדמים בשכבות הולכים ומתמקדים בפרטים יותר קטנים בתמונה ועם כמות פילטרים רבה יותר. בנוסף, משתמשים בשכבת האקטיבציה "relu" על מנת להפחית את הסבירות להיעלמות הgradient ושכבת max Pooling על מנת לייצא את הפיצ'רים החשובים. לבסוף יש שימוש בGlobal Avarage Pooling 2D על מנת להפוך את הפיצ'רים למימד אחד ואז משתמשים בdropout וdense כשכבות fully connected.

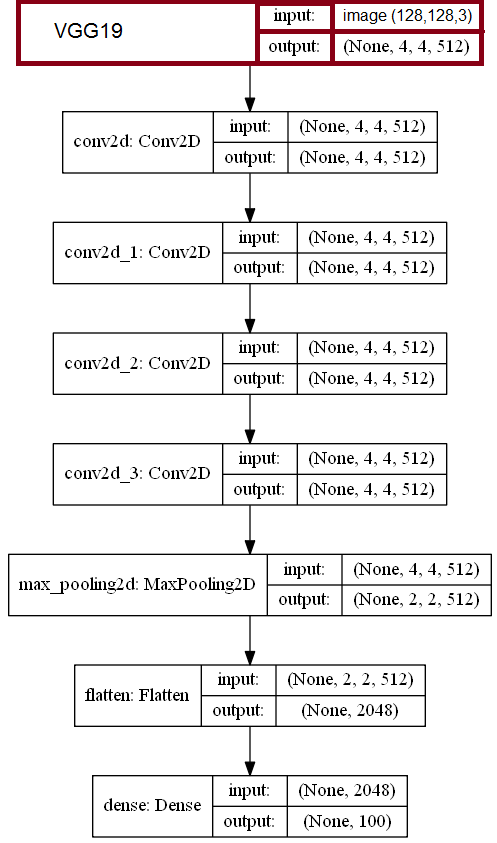
ניתן לראות בבירור כי יש שיפור תמידי בaccuracy והloss יורד כל הזמן, אם כי הקצב יחסית רגוע הן בגלל שהlearning rate נמוך והן בגלל שהרשת נבנתה בצורה מדורגת.

ניסוי 4

הרשת מבוססת על VGG19 המאומנת עם Imagenet.

לרשת הוספנו את השכבות הבאות:

1. שכבת קונבולוציה עם 512 פילטרים בגודל 4x4. פונקציית אקטיבציה relu, עם padding.
2. שכבת קונבולוציה עם 512 פילטרים בגודל 4x4. פונקציית אקטיבציה relu, עם padding.
3. שכבת קונבולוציה עם 512 פילטרים בגודל 4x4. פונקציית אקטיבציה relu, עם padding.
4. שכבת קונבולוציה עם 512 פילטרים בגודל 4x4. פונקציית אקטיבציה relu, עם padding.
5. שכבת max pooling עם poolsize בגודל 2x2 וstrides של 2x2, פונקציית אקטיבציה relu, עם padding.
6. שכבת flatten
7. שכבת Dense בגודל 100 (ככמות המחלקות) עם פונקציית אקטיבציה softmax - שכבה זו היא השכבה האחרונה שלאחריה מתקבל סיווג של האובייקט לאחת מתוך 100 מחלקות.



הרצנו את הרשת עם learning rate 0.001, momentum 0.7.

Batch size 30. הרצנו 3 epochים עד שעצרנו את האימון.

Epoch 1/5

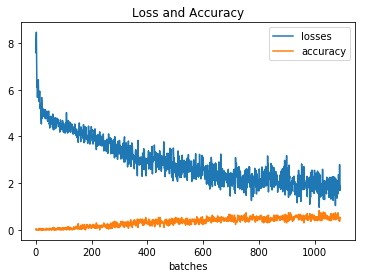
335/335 [==============================] - 1326s 4s/step - loss: 4.2019 - acc: 0.1141 - val\_loss: 5.9405 - val\_acc: 0.0061

Epoch 2/5

335/335 [==============================] - 1314s 4s/step - loss: 2.7507 - acc: 0.3660 - val\_loss: 7.4348 - val\_acc: 0.0064

Epoch 3/5

335/335 [==============================] - 1315s 4s/step - loss: 2.0954 - acc: 0.4968 - val\_loss: 7.9504 - val\_acc: 0.0086



מסקנות-

אפשר לראות שבמהלך האימון הרשת משתפרת מאוד, אך בקצב איטי מידי (יתכן שהlearning rate נמוך מידי). החלטנו לעצור את האימון לאחר 3 epochים כדי לשפר את קצב הלמידה. בנוסף אפשר לראות שהרשת סובלת מ-over fitting- למשל בepoch האחרון אפשר לראות accuracy 0.49 אך validation accuracy של 0.0086 כלומר שהרשת ידעה לסווג נכון כמחצית מהתמונות שהיא למדה, אבל אפילו לא אחוז אחד מתמונות שלא הכירה. ברשת מספר 5 נציג את השיפורים שעשינו לרשת הזו (כולל שימוש בdropout).

ניסוי 4 ב'

ניסוי זה מתבסס על הניסוי הקודם שהופסק באמצע לטובת ביצוע שיפורים.

השינוי המרכזי ברשת זו הוא הוספת שכבת dropout, לאחר רשת הvgg19 המוכנה, ולפני שאר השכבות שהוספנו בעצמנו.

שכבת Dropout (P=0.15)– היא עושה נורמליזציה ומפחיתה over fitting ע"י הסרת נוירונים וקשתות באופן אקראי מרשת הנוירונים בגודל P שנקבע. ההסרה מתבצעת באופן אקראי וכך בכל הרצה מתקבלת רשת אחרת, מה שמונע אימון יתר של אותה רשת בדיוק.

הגדלנו את הmomentum ל-0.99 (כי חשבנו שמומנטום יעזור לביצועים עקב השיפור הלינארי הנמוך), והגדלנו את הlearning rate ב-0.001 על מנת להגדיל את קצב הלמידה.

כמו כן הקטנו את הbatch size מ-30 ל-20 כדי שהרשת תבצע back propagation יותר פעמים במהלך epoch יחיד.

כעת הרצנו את הרשת למשך 5 epochים:

Epoch 1/5

80/80 [==============================] - 219s 3s/step - loss: 2.8428 - acc: 0.3293 - val\_loss: 1.5106 - **val\_acc: 0.5637**

Epoch 2/5

80/80 [==============================] - 219s 3s/step - loss: 1.2635 - acc: 0.6307 - val\_loss: 1.1721 - **val\_acc: 0.6607**

Epoch 3/5

80/80 [==============================] - 219s 3s/step - loss: 0.8822 - acc: 0.7570 - val\_loss: 0.8853 - **val\_acc: 0.7576**

Epoch 4/5

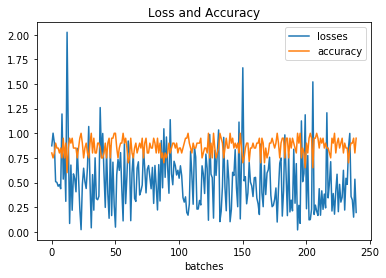
80/80 [==============================] - 219s 3s/step - loss: 0.7897 - acc: 0.7845 - val\_loss: 0.7868 - **val\_acc: 0.7666**

Epoch 5/5

80/80 [==============================] - 220s 3s/step - loss: 0.6916 - acc: 0.8076 - val\_loss: 0.7647 - **val\_acc: 0.7971**

הרצה עם validation:

Epoch 1/3 79/80 [============================>.] - ETA: 2s - loss: 0.5695 - acc: 0.8551 Epoch 00001: val\_acc improved from -inf to 0.78636, saving model to C:\depp\Output\ML5\VGG19\_WITH\_DROPOUT\_and\_extra\_conv\_layer\_1549020705.h5 80/80 [==============================] - 221s 3s/step - loss: 0.5706 - acc: 0.8550 - val\_loss: 0.7577 - val\_acc: 0.7864 Epoch 2/3 79/80 [============================>.] - ETA: 2s - loss: 0.5441 - acc: 0.8603 Epoch 00002: val\_acc improved from 0.78636 to 0.81149, saving model to C:\depp\Output\ML5\VGG19\_WITH\_DROPOUT\_and\_extra\_conv\_layer\_1549020705.h5 80/80 [==============================] - 221s 3s/step - loss: 0.5441 - acc: 0.8602 - val\_loss: 0.6569 - val\_acc: 0.8115 Epoch 3/3 79/80 [============================>.] - ETA: 2s - loss: 0.4513 - acc: 0.8759 Epoch 00003: val\_acc improved from 0.81149 to 0.82406, saving model to C:\depp\Output\ML5\VGG19\_WITH\_DROPOUT\_and\_extra\_conv\_layer\_1549020705.h5 80/80 [==============================] - 228s 3s/step - loss: 0.4481 - acc: 0.8769 - val\_loss: 0.7130 - val\_acc: 0.8241



מסקנות:

הdropout תרם בפתרון בעיית הover fitting שהיתה לנו בניסוי הקודם.  
הגדלת הmomentum ו- learning rate , והקטנת הbatch size איפשרו לרשת ללמוד מהר יותר.

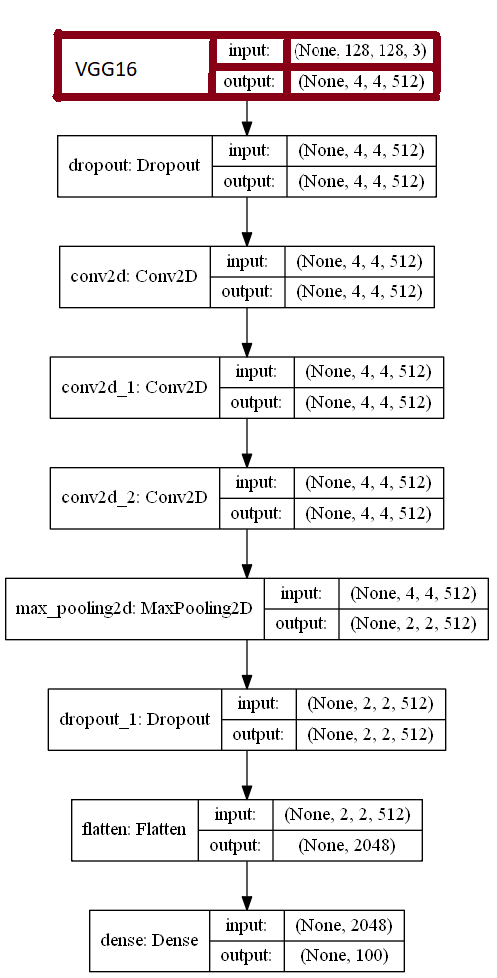
ניסוי 5

הרשת מבוססת על VGG16 מאומנת על imagenet, ומכילה שיפורים ולקחים גם מהרשתות הקודמות.

הרשת מכילה את השכבות הנוספות הבאות:

1. שכבת Dropout (P=0.4)
2. שכבת קונבולוציה עם 512 פילטרים בגודל 3x3. עם padding, ופונקציית אקטיבציה tanh. הפעם בחרנו בtanh ולא בrelu לאחר שקראנו באינטרנט כי טווח הערכים שבין מינוס 1 ל-1 נותן לנו גרדיאנטים חזקים יותר.
3. שכבת קונבולוציה עם 512 פילטרים בגודל 3x3. עם padding, ופונקציית אקטיבציה tanh.
4. שכבת קונבולוציה עם 512 פילטרים בגודל 3x3. עם padding, ופונקציית אקטיבציה tanh.
5. שכבת max pooling עם poolsize בגודל 2x2 וstrides של 2x2
6. שכבת Dropout (P=0.4)
7. שכבת flatten
8. שכבת Dense בגודל 20 (ככמות המחלקות) עם פונקציית אקטיבציה softmax - שכבה זו היא השכבה האחרונה שלאחריה מתקבל סיווג של האובייקט לאחת מתוך 20 מחלקות.

להלן תרשים המתאר את הרשת:



לתהליך הלמידה בחרנו בlearning rate 0.001 ומומנטום של 0.9. השתמשנו בbatch size של 20 תמונות וביצענו הרצה של 8 epochים כמפורט:

Epoch 1/8

80/80 [==============================] - 188s 2s/step - loss: 2.3059 - acc: 0.3306 - val\_loss: 1.2205 - val\_acc: 0.6984

Epoch 2/8

80/80 [==============================] - 192s 2s/step - loss: 1.1494 - acc: 0.6826 - val\_loss: 0.7902 - val\_acc: 0.7648

Epoch 3/8

80/80 [==============================] - 195s 2s/step - loss: 0.7994 - acc: 0.7625 - val\_loss: 0.6112 - val\_acc: 0.8133

Epoch 4/8

80/80 [==============================] - 191s 2s/step - loss: 0.6550 - acc: 0.7972 - val\_loss: 0.5490 - val\_acc: 0.8420

Epoch 5/8

80/80 [==============================] - 194s 2s/step - loss: 0.5507 - acc: 0.8300 - val\_loss: 0.4631 - val\_acc: 0.8689

Epoch 6/8

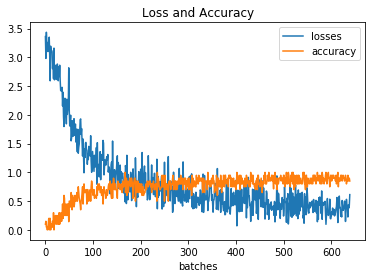
80/80 [==============================] - 192s 2s/step - loss: 0.5089 - acc: 0.8439 - val\_loss: 0.4182 - val\_acc: 0.8851

Epoch 7/8

80/80 [==============================] - 192s 2s/step - loss: 0.4395 - acc: 0.8669 - val\_loss: 0.4940 - val\_acc: 0.8366

Epoch 8/8

80/80 [==============================] - 192s 2s/step - loss: 0.3763 - acc: 0.8912 - val\_loss: 0.4556 - val\_acc: 0.8528



מסקנות:

לפי ערכי הvalidation accuracy בכל epoch- ערכים הנחשבים גבוהים יחסית, אפשר להניח שפונקציות הdropout תרמה למניעת over fitting.  
המומנטום הגבוה תרם להתאמת קצב הלמידה בצורה טובה.  
ביצועי הרשת הזו הם הטובים ביותר שהשגנו עד לנקודה זו.