אלגוריתמים בראייה ממוחשבת - 046746

רטוב 2

דניאל טייטלמן – 207734088 – Daniel.tei@campus.technion.ac.il

יאיר נחום – 034462796 – nahum.yair@campus.technion.ac.il

פרק 1:

1. הורדנו את ה – Database הנקרא SVHN.

```
Downloading http://ufldl.stanford.edu/housenumbers/train_32x32.mat to ./datasets/train_32x32.mat
182041600/? [00:10<00:00, 17926585.62it/s]

Downloading http://ufldl.stanford.edu/housenumbers/test_32x32.mat to ./datasets/test_32x32.mat
64275456/? [00:05<00:00, 12627021.10it/s]
```

לאחר מכן נציג 5 תמונות הראשונות ב – Dataset של סט האימון ואת התיוג המתאים להן:

2. כעת אנו נדרשים להשתמש בתהליך דומה לתרגולים 3 ו 4 לטובת אימון רשת ראשונית לסיווג הדאטא הנ"ל, על כן בשלב זה ביצענו תהליכים דומים לתרגול אותם נתאר כעת.

א. ראשית כל הגדרנו אוגמנטציות על סט האימון שהן: חיתוך אקראי לגודל 32 עם ריפוד של 4. הפיכה אקראית לאורך הציר האופקי ונרמול. בעבור סט המבחן השתמשנו אך ורק בנרמול כי כך נתבקשנו. (הערת צד ניתן לשפר רובסטיות מודלים באמצעות דבר שנקרא *TEST TIME AUGMENTATION* דבר זה מאפשר להשתמש באוגמנטציות על סט המבחן בפרדיקציה ואז לבחור את תוצאת ה – *MAX CLASS* בסיווג על פני האוגמנטציות וכך לשפר את הרובסטיות של המודל).

ב. לאחר מכן הגדרנו *Batch_size=128*, קצב לימוד של $lr = 10^{-4}$ ומספר *Epochs* של 20 בדומה לתרגול, באמצעות ה – *Batch_size* הגדרנו את ה – *DATALODAER* כאשר אנו מבצעים רנדומיזציה במיקום לסט האימון ולא מערבבים את סט המבחן.

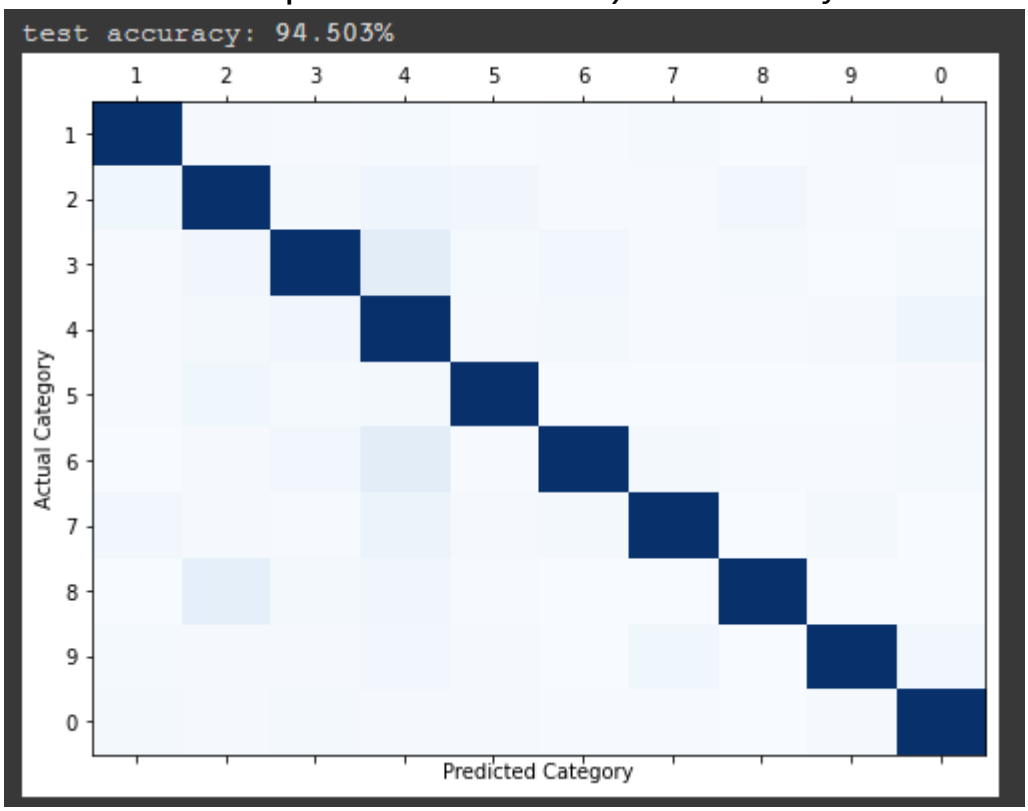
ג. הגדרת רשת *CifarCNN* בדומה לתרגול מצורפת תמונה של הגדרת הרשת. (התמונה יוצרה באמצעות *Torchsummary* שהותקנה בתחילת המחברת ופקודת *summary*).

```
Model summary:
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 32, 32]             896
       BatchNorm2d-2           [-1, 32, 32, 32]              64
              ReLU-3           [-1, 32, 32, 32]               0
            Conv2d-4           [-1, 64, 32, 32]          18,496
              ReLU-5           [-1, 64, 32, 32]               0
         MaxPool2d-6           [-1, 64, 16, 16]               0
            Conv2d-7          [-1, 128, 16, 16]          73,856
       BatchNorm2d-8          [-1, 128, 16, 16]             256
              ReLU-9          [-1, 128, 16, 16]               0
           Conv2d-10          [-1, 128, 16, 16]         147,584
             ReLU-11          [-1, 128, 16, 16]               0
        MaxPool2d-12            [-1, 128, 8, 8]               0
        Dropout2d-13            [-1, 128, 8, 8]               0
           Conv2d-14            [-1, 256, 8, 8]         295,168
      BatchNorm2d-15            [-1, 256, 8, 8]             512
             ReLU-16            [-1, 256, 8, 8]               0
           Conv2d-17            [-1, 256, 8, 8]         590,080
             ReLU-18            [-1, 256, 8, 8]               0
        MaxPool2d-19            [-1, 256, 4, 4]               0
          Dropout-20                 [-1, 4096]               0
           Linear-21                 [-1, 1024]       4,195,328
             ReLU-22                 [-1, 1024]               0
           Linear-23                  [-1, 512]         524,800
             ReLU-24                  [-1, 512]               0
          Dropout-25                  [-1, 512]               0
           Linear-26                   [-1, 10]           5,130
================================================================
Total params: 5,852,170
Trainable params: 5,852,170
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 3.96
Params size (MB): 22.32
Estimated Total Size (MB): 26.30
----------------------------------------------------------------
```

ד. לאחר מכן אימנו את המודל באמצעות הפונקציה שניתנה בתרגול לאימון המודל, בנוסף השתמשנו ב – *ADAM OPTIMIZER* ופונקציית המחיר שלנו הייתה *CATEGORICAL CROSS ENTROPY*. מצורפת תמונה של ה – *LOSS* במהלך האימון.

```
Epoch: 1 | Loss: 1.3428 | Training accuracy: 78.724% | Test accuracy: 77.774% | Epoch Time: 80.67 secs
Epoch: 2 | Loss: 0.5329 | Training accuracy: 86.673% | Test accuracy: 86.524% | Epoch Time: 80.61 secs
Epoch: 3 | Loss: 0.4086 | Training accuracy: 89.130% | Test accuracy: 89.448% | Epoch Time: 80.89 secs
Epoch: 4 | Loss: 0.3534 | Training accuracy: 89.817% | Test accuracy: 90.239% | Epoch Time: 82.10 secs
Epoch: 5 | Loss: 0.3163 | Training accuracy: 91.175% | Test accuracy: 90.527% | Epoch Time: 82.28 secs
Epoch: 6 | Loss: 0.2914 | Training accuracy: 92.412% | Test accuracy: 92.187% | Epoch Time: 81.70 secs
Epoch: 7 | Loss: 0.2723 | Training accuracy: 92.848% | Test accuracy: 93.132% | Epoch Time: 81.17 secs
Epoch: 8 | Loss: 0.2585 | Training accuracy: 93.324% | Test accuracy: 92.744% | Epoch Time: 80.13 secs
Epoch: 9 | Loss: 0.2458 | Training accuracy: 93.313% | Test accuracy: 93.020% | Epoch Time: 81.24 secs
Epoch: 10 | Loss: 0.2338 | Training accuracy: 93.745% | Test accuracy: 93.562% | Epoch Time: 81.92 secs
Epoch: 11 | Loss: 0.2241 | Training accuracy: 94.338% | Test accuracy: 93.700% | Epoch Time: 82.27 secs
Epoch: 12 | Loss: 0.2173 | Training accuracy: 94.547% | Test accuracy: 94.207% | Epoch Time: 81.10 secs
Epoch: 13 | Loss: 0.2072 | Training accuracy: 94.649% | Test accuracy: 94.119% | Epoch Time: 81.57 secs
Epoch: 14 | Loss: 0.2009 | Training accuracy: 95.123% | Test accuracy: 94.272% | Epoch Time: 81.83 secs
Epoch: 15 | Loss: 0.1935 | Training accuracy: 94.932% | Test accuracy: 93.992% | Epoch Time: 81.67 secs
Epoch: 16 | Loss: 0.1875 | Training accuracy: 95.278% | Test accuracy: 94.050% | Epoch Time: 81.57 secs
Epoch: 17 | Loss: 0.1846 | Training accuracy: 95.643% | Test accuracy: 94.407% | Epoch Time: 82.20 secs
Epoch: 18 | Loss: 0.1742 | Training accuracy: 95.766% | Test accuracy: 94.242% | Epoch Time: 81.94 secs
Epoch: 19 | Loss: 0.1686 | Training accuracy: 95.669% | Test accuracy: 94.707% | Epoch Time: 81.82 secs
Epoch: 20 | Loss: 0.1649 | Training accuracy: 96.108% | Test accuracy: 94.503% | Epoch Time: 82.30 secs
==> Saving model ...
==> Finished Training ...
```

ה. לאחר מכן השתמשנו בקוד מן התרגול על מנת לחשב את ה – *Confusion matrix* ואת ה – *Accuracy* על סט המבחן.



כפי שניתן לראות אנו מצליחים ברוב התגיות אך ישנו בלבול משמעותי (יחסית) בין 8 ל – 2, בין 4 ו 6 ובין 4 ל – 0.

3. על בסיס הרשת מהסעיפים הקודמים בנינו רשת אחרת, בעלת ביצועים טובים יותר על סט המבחן. כאשר שאר ה – *Hyper parameters* והאוגמנטציות זהים לסעיפים הקודמים. מצורפת תמונה של מבנה הרשת ותוצאות האימון.

א. מבנה הרשת:

```
Our model summary:
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           1,792
       BatchNorm2d-2           [-1, 64, 32, 32]             128
              ReLU-3           [-1, 64, 32, 32]               0
            Conv2d-4          [-1, 128, 32, 32]          73,856
              ReLU-5          [-1, 128, 32, 32]               0
         MaxPool2d-6          [-1, 128, 16, 16]               0
            Conv2d-7          [-1, 256, 16, 16]         295,168
       BatchNorm2d-8          [-1, 256, 16, 16]             512
              ReLU-9          [-1, 256, 16, 16]               0
           Conv2d-10          [-1, 256, 16, 16]         590,080
             ReLU-11          [-1, 256, 16, 16]               0
        MaxPool2d-12            [-1, 256, 8, 8]               0
        Dropout2d-13            [-1, 256, 8, 8]               0
           Conv2d-14            [-1, 512, 8, 8]       1,180,160
      BatchNorm2d-15            [-1, 512, 8, 8]           1,024
             ReLU-16            [-1, 512, 8, 8]               0
           Conv2d-17            [-1, 512, 8, 8]       2,359,808
             ReLU-18            [-1, 512, 8, 8]               0
        MaxPool2d-19            [-1, 512, 4, 4]               0
          Dropout-20                 [-1, 8192]               0
           Linear-21                 [-1, 1024]       8,389,632
             ReLU-22                 [-1, 1024]               0
           Linear-23                  [-1, 512]         524,800
             ReLU-24                  [-1, 512]               0
          Dropout-25                  [-1, 512]               0
           Linear-26                   [-1, 10]           5,130
================================================================
Total params: 13,422,090
Trainable params: 13,422,090
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 7.90
Params size (MB): 51.20
Estimated Total Size (MB): 59.12
----------------------------------------------------------------
```

נתאר את השינויים שביצענו במבנה הרשת:

בעבור כל שכבת קונבולציה הכפלנו את גודלה פי 2 מהרשת המקורית, ושינינו את ה – *BatchNorm* גם כן כך שיבוצע על פי 2 רכיבים. לבסוף שינינו בשכבת ה – *Fully connected* את רכיב ה – *LINEAR* הראשון כך שבמקום כניסה של 4096 ישנה כניסה של 8192, בצורה זאת שכבה ה – *Fully connected* מקבלת כניסה בהתאם ליציאת שכבת ה – *CONV*. ה- *ACTIVATIONS* זהים לרשת הקודמת והם *RELU*. הפילטרים בשכבות הקונבולציה הם בעלי גרעין בגודל 3 ועם ריפוד של 1. בנוסף בשכבות *D2MAXPOOLING* ה – *STRIDE* הוא 2 וגודל הגרעין הוא 2. בנוסף ישנם *DROPOUTS* לשיפור יכולת ההכללה כאשר בשכבות הקונבולציה הם בהסתברות 0.05 ובשכבת *FC* הם בהסתברות 0.1.

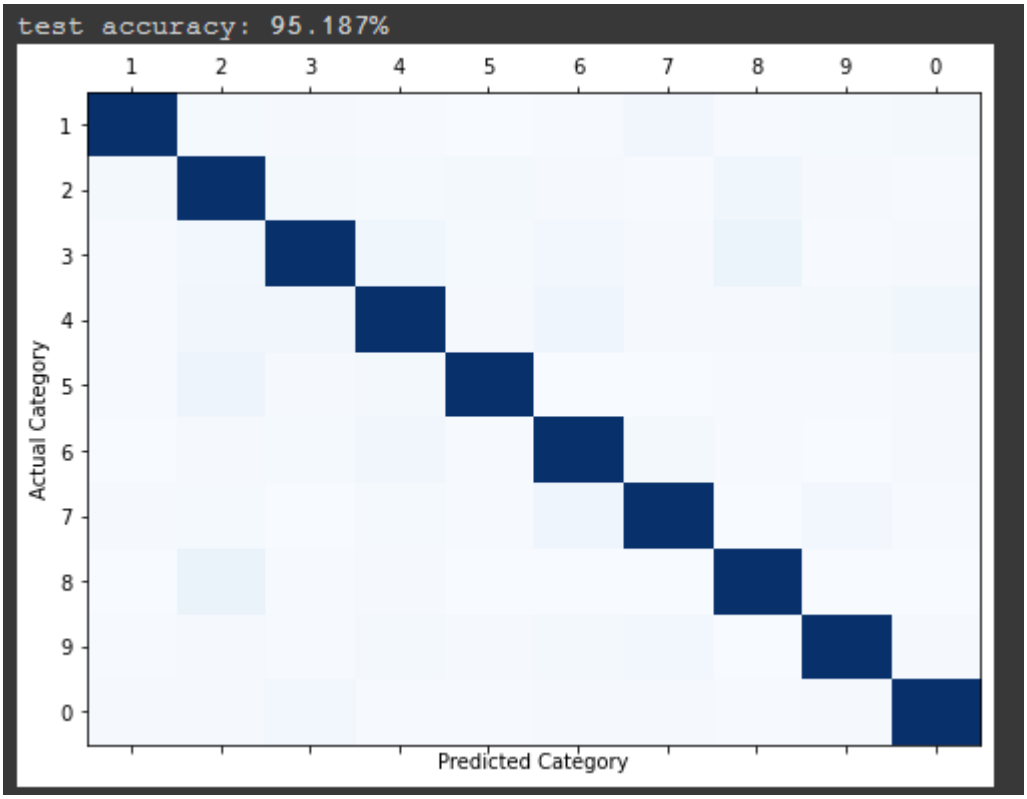
מימד הכניסה הוא טנזור 4 ממדי כאשר המימד הראשון לא רלוונטי וקשור למספר הדוגמאות הנכנסות לרשת, המימד השני הוא מספר

הערוצים (ערוצי צבע – 3 ערוצים) והמימד השלישי והרביעי הם גודל התמונה שזה 32 על 32.
יציאת הרשת היא וקטור באורך 10 של הסתברויות בעבור כל *CLASS*.
מספר הפרמטרים הניתנים לאימון ברשת הם: 13422090.
לסיכום, הסיבה שבחרנו ברשת זאת כי הרשת הראשונית כבר הניבה תוצאה מצוינת והנחנו כי אם נשחק מעט עם ערכי הרשת נוכל להגיע לתוצאה טובה, ואכן התקבל כך.

ב. מהלך תהליך האימון:

```
Epoch: 1 | Loss: 1.1598 | Training accuracy: 82.761% | Test accuracy: 83.117% | Epoch Time: 121.55 secs
Epoch: 2 | Loss: 0.4390 | Training accuracy: 89.440% | Test accuracy: 89.232% | Epoch Time: 121.59 secs
Epoch: 3 | Loss: 0.3414 | Training accuracy: 91.335% | Test accuracy: 91.568% | Epoch Time: 121.53 secs
Epoch: 4 | Loss: 0.2996 | Training accuracy: 92.241% | Test accuracy: 92.390% | Epoch Time: 121.46 secs
Epoch: 5 | Loss: 0.2714 | Training accuracy: 92.928% | Test accuracy: 92.959% | Epoch Time: 121.29 secs
Epoch: 6 | Loss: 0.2522 | Training accuracy: 93.497% | Test accuracy: 93.393% | Epoch Time: 121.39 secs
Epoch: 7 | Loss: 0.2348 | Training accuracy: 93.841% | Test accuracy: 93.431% | Epoch Time: 121.26 secs
Epoch: 8 | Loss: 0.2227 | Training accuracy: 94.657% | Test accuracy: 94.150% | Epoch Time: 120.89 secs
Epoch: 9 | Loss: 0.2097 | Training accuracy: 95.020% | Test accuracy: 94.042% | Epoch Time: 121.32 secs
Epoch: 10 | Loss: 0.1972 | Training accuracy: 95.052% | Test accuracy: 93.777% | Epoch Time: 121.41 secs
Epoch: 11 | Loss: 0.1877 | Training accuracy: 95.109% | Test accuracy: 93.788% | Epoch Time: 121.29 secs
Epoch: 12 | Loss: 0.1804 | Training accuracy: 95.422% | Test accuracy: 94.207% | Epoch Time: 121.38 secs
Epoch: 13 | Loss: 0.1739 | Training accuracy: 95.461% | Test accuracy: 94.280% | Epoch Time: 121.76 secs
Epoch: 14 | Loss: 0.1651 | Training accuracy: 96.146% | Test accuracy: 94.856% | Epoch Time: 121.29 secs
Epoch: 15 | Loss: 0.1562 | Training accuracy: 96.428% | Test accuracy: 94.868% | Epoch Time: 121.25 secs
Epoch: 16 | Loss: 0.1514 | Training accuracy: 96.216% | Test accuracy: 94.660% | Epoch Time: 121.60 secs
Epoch: 17 | Loss: 0.1441 | Training accuracy: 96.485% | Test accuracy: 94.906% | Epoch Time: 121.46 secs
Epoch: 18 | Loss: 0.1387 | Training accuracy: 96.848% | Test accuracy: 94.987% | Epoch Time: 121.40 secs
Epoch: 19 | Loss: 0.1338 | Training accuracy: 96.959% | Test accuracy: 95.010% | Epoch Time: 121.31 secs
Epoch: 20 | Loss: 0.1268 | Training accuracy: 97.049% | Test accuracy: 95.187% | Epoch Time: 121.05 secs
==> Saving model ...
==> Finished Training ...
```

ג. תוצאת המודל על סט המבחן: כפי ניתן לראות התוצאה (*Accuracy*) השתפרה באזור ה - 0.6%.



4. השתמשנו ב – *Validation set* על מנת לבצע *Tuning* להיפר פרמטרים של המודל שלנו כך שפיצלנו את ה – *TRAINING SET* ל 80% סט אימון ו 20% סט ולידציה. על פי תוצאת ה – *Accuracy* על סט הוולידציה שינינו את ה – *HYPER PARAMERTS* על מנת לשפר את התוצאה שלנו. לאחר בחירת הפרמטרים הטובים ביותר אימנו מאפס שוב את הרשת עם כלל סט האימון (כלומר סט אימון וסט ולידציה).

הפרמטרים הטובים ביותר הם:

$$lr = 1.5e-4, batch\ size = 128, epochs = 30$$

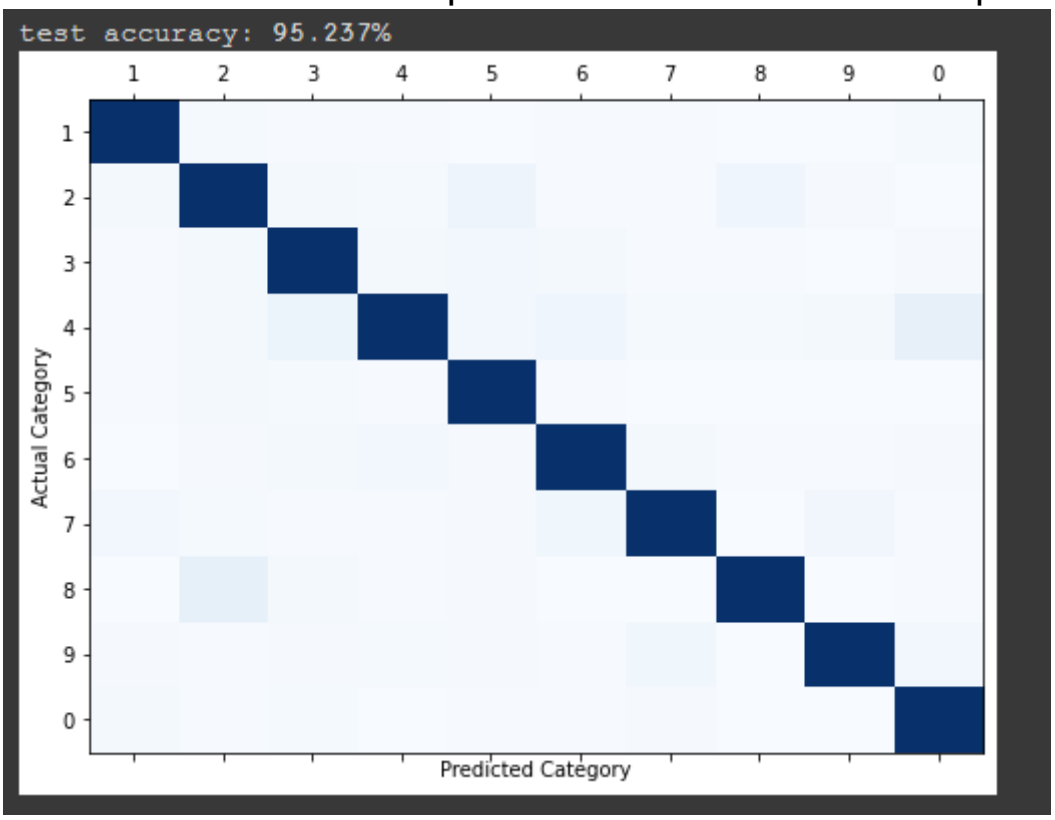
תהליך האימון בעבור הפרמטרים הנ"ל כאשר אנו מודדים ביחס לסט הוולידציה.

```
Epoch: 1 | Loss: 1.1234 | Training accuracy: 83.413% | Validation accuracy: 84.131% | Epoch Time: 99.65 secs
Epoch: 2 | Loss: 0.4383 | Training accuracy: 88.298% | Validation accuracy: 88.547% | Epoch Time: 99.31 secs
Epoch: 3 | Loss: 0.3479 | Training accuracy: 90.615% | Validation accuracy: 90.246% | Epoch Time: 99.10 secs
Epoch: 4 | Loss: 0.3023 | Training accuracy: 92.069% | Validation accuracy: 90.970% | Epoch Time: 99.46 secs
Epoch: 5 | Loss: 0.2727 | Training accuracy: 92.914% | Validation accuracy: 91.782% | Epoch Time: 99.23 secs
Epoch: 6 | Loss: 0.2564 | Training accuracy: 93.801% | Validation accuracy: 92.915% | Epoch Time: 99.66 secs
Epoch: 7 | Loss: 0.2366 | Training accuracy: 93.936% | Validation accuracy: 92.649% | Epoch Time: 99.66 secs
Epoch: 8 | Loss: 0.2215 | Training accuracy: 94.180% | Validation accuracy: 92.785% | Epoch Time: 100.05 secs
Epoch: 9 | Loss: 0.2140 | Training accuracy: 94.716% | Validation accuracy: 93.018% | Epoch Time: 99.54 secs
Epoch: 10 | Loss: 0.2007 | Training accuracy: 94.999% | Validation accuracy: 93.611% | Epoch Time: 99.37 secs
==> Saving model ...
Epoch: 11 | Loss: 0.1912 | Training accuracy: 95.233% | Validation accuracy: 93.700% | Epoch Time: 99.41 secs
Epoch: 12 | Loss: 0.1835 | Training accuracy: 95.432% | Validation accuracy: 93.762% | Epoch Time: 99.26 secs
Epoch: 13 | Loss: 0.1752 | Training accuracy: 95.569% | Validation accuracy: 93.632% | Epoch Time: 99.81 secs
Epoch: 14 | Loss: 0.1674 | Training accuracy: 95.746% | Validation accuracy: 93.741% | Epoch Time: 99.70 secs
Epoch: 15 | Loss: 0.1621 | Training accuracy: 96.004% | Validation accuracy: 93.755% | Epoch Time: 99.67 secs
Epoch: 16 | Loss: 0.1528 | Training accuracy: 96.244% | Validation accuracy: 94.130% | Epoch Time: 99.51 secs
Epoch: 17 | Loss: 0.1458 | Training accuracy: 96.378% | Validation accuracy: 93.987% | Epoch Time: 99.59 secs
Epoch: 18 | Loss: 0.1404 | Training accuracy: 96.000% | Validation accuracy: 93.686% | Epoch Time: 99.91 secs
Epoch: 19 | Loss: 0.1331 | Training accuracy: 96.942% | Validation accuracy: 94.151% | Epoch Time: 99.92 secs
Epoch: 20 | Loss: 0.1295 | Training accuracy: 97.028% | Validation accuracy: 94.273% | Epoch Time: 99.62 secs
==> Saving model ...
Epoch: 21 | Loss: 0.1185 | Training accuracy: 97.202% | Validation accuracy: 94.321% | Epoch Time: 99.61 secs
Epoch: 22 | Loss: 0.1179 | Training accuracy: 97.280% | Validation accuracy: 94.362% | Epoch Time: 97.93 secs
Epoch: 23 | Loss: 0.1118 | Training accuracy: 97.386% | Validation accuracy: 94.294% | Epoch Time: 98.62 secs
Epoch: 24 | Loss: 0.1073 | Training accuracy: 97.695% | Validation accuracy: 94.232% | Epoch Time: 98.60 secs
Epoch: 25 | Loss: 0.1015 | Training accuracy: 97.413% | Validation accuracy: 94.212% | Epoch Time: 98.44 secs
Epoch: 26 | Loss: 0.0988 | Training accuracy: 97.927% | Validation accuracy: 93.932% | Epoch Time: 99.31 secs
Epoch: 27 | Loss: 0.0920 | Training accuracy: 97.872% | Validation accuracy: 94.376% | Epoch Time: 99.78 secs
Epoch: 28 | Loss: 0.0931 | Training accuracy: 97.969% | Validation accuracy: 94.444% | Epoch Time: 99.78 secs
Epoch: 29 | Loss: 0.0851 | Training accuracy: 98.101% | Validation accuracy: 94.437% | Epoch Time: 99.36 secs
Epoch: 30 | Loss: 0.0820 | Training accuracy: 98.174% | Validation accuracy: 94.628% | Epoch Time: 98.62 secs
==> Saving model ...
==> Finished Training ...
```

לאחר מכן אימנו על פני כלל ה – *DATASET* (הדיוק על הוולידציה מוצג בתמונה אך לא רלוונטי מפני שכעת הינו חלק מסט האימון).

```
Epoch: 1 | Loss: 1.1656 | Training accuracy: 81.480% | Validation accuracy: 81.646% | Epoch Time: 116.39 secs
Epoch: 2 | Loss: 0.4191 | Training accuracy: 89.852% | Validation accuracy: 90.376% | Epoch Time: 117.74 secs
Epoch: 3 | Loss: 0.3293 | Training accuracy: 90.491% | Validation accuracy: 90.936% | Epoch Time: 117.93 secs
Epoch: 4 | Loss: 0.2933 | Training accuracy: 92.451% | Validation accuracy: 93.461% | Epoch Time: 117.64 secs
Epoch: 5 | Loss: 0.2632 | Training accuracy: 92.788% | Validation accuracy: 93.338% | Epoch Time: 117.37 secs
Epoch: 6 | Loss: 0.2455 | Training accuracy: 93.689% | Validation accuracy: 93.809% | Epoch Time: 117.46 secs
Epoch: 7 | Loss: 0.2297 | Training accuracy: 93.954% | Validation accuracy: 94.608% | Epoch Time: 117.95 secs
Epoch: 8 | Loss: 0.2168 | Training accuracy: 94.040% | Validation accuracy: 93.966% | Epoch Time: 117.11 secs
Epoch: 9 | Loss: 0.2064 | Training accuracy: 94.906% | Validation accuracy: 94.990% | Epoch Time: 116.80 secs
Epoch: 10 | Loss: 0.1959 | Training accuracy: 95.276% | Validation accuracy: 95.666% | Epoch Time: 117.28 secs
==> Saving model ...
Epoch: 11 | Loss: 0.1853 | Training accuracy: 95.430% | Validation accuracy: 95.673% | Epoch Time: 117.34 secs
Epoch: 12 | Loss: 0.1783 | Training accuracy: 95.598% | Validation accuracy: 96.171% | Epoch Time: 116.79 secs
Epoch: 13 | Loss: 0.1731 | Training accuracy: 95.654% | Validation accuracy: 96.150% | Epoch Time: 116.71 secs
Epoch: 14 | Loss: 0.1655 | Training accuracy: 95.973% | Validation accuracy: 96.656% | Epoch Time: 117.07 secs
Epoch: 15 | Loss: 0.1577 | Training accuracy: 96.115% | Validation accuracy: 96.355% | Epoch Time: 116.79 secs
Epoch: 16 | Loss: 0.1530 | Training accuracy: 96.164% | Validation accuracy: 96.526% | Epoch Time: 115.80 secs
Epoch: 17 | Loss: 0.1463 | Training accuracy: 96.699% | Validation accuracy: 97.195% | Epoch Time: 115.49 secs
Epoch: 18 | Loss: 0.1387 | Training accuracy: 96.631% | Validation accuracy: 97.031% | Epoch Time: 115.52 secs
Epoch: 19 | Loss: 0.1358 | Training accuracy: 96.761% | Validation accuracy: 97.304% | Epoch Time: 115.70 secs
Epoch: 20 | Loss: 0.1275 | Training accuracy: 97.073% | Validation accuracy: 97.440% | Epoch Time: 115.46 secs
==> Saving model ...
Epoch: 21 | Loss: 0.1236 | Training accuracy: 97.251% | Validation accuracy: 97.591% | Epoch Time: 115.13 secs
Epoch: 22 | Loss: 0.1174 | Training accuracy: 97.214% | Validation accuracy: 97.270% | Epoch Time: 115.45 secs
Epoch: 23 | Loss: 0.1136 | Training accuracy: 97.008% | Validation accuracy: 97.243% | Epoch Time: 115.95 secs
Epoch: 24 | Loss: 0.1102 | Training accuracy: 97.177% | Validation accuracy: 97.516% | Epoch Time: 117.32 secs
Epoch: 25 | Loss: 0.1061 | Training accuracy: 97.699% | Validation accuracy: 97.823% | Epoch Time: 116.29 secs
Epoch: 26 | Loss: 0.1016 | Training accuracy: 97.917% | Validation accuracy: 98.062% | Epoch Time: 116.16 secs
Epoch: 27 | Loss: 0.0969 | Training accuracy: 97.806% | Validation accuracy: 98.137% | Epoch Time: 116.51 secs
Epoch: 28 | Loss: 0.0930 | Training accuracy: 97.902% | Validation accuracy: 98.212% | Epoch Time: 115.76 secs
Epoch: 29 | Loss: 0.0903 | Training accuracy: 97.802% | Validation accuracy: 98.096% | Epoch Time: 115.67 secs
Epoch: 30 | Loss: 0.0876 | Training accuracy: 98.123% | Validation accuracy: 98.444% | Epoch Time: 115.39 secs
==> Saving model ...
==> Finished Training ...
```

לבסוף התוצאה הסופית היא על סט המבחן:



התוצאה השתפרה בכ – 0.1%.

פרק 2:

1. הורדנו את מודל ה vgg16 המאומן ונכנסנו למצב evaluation.

```
VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace)
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU(inplace)
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU(inplace)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU(inplace)
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18): ReLU(inplace)
    (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): ReLU(inplace)
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): ReLU(inplace)
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25): ReLU(inplace)
    (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (27): ReLU(inplace)
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29): ReLU(inplace)
    (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
  (classifier): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace)
    (2): Dropout(p=0.5)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU(inplace)
    (5): Dropout(p=0.5)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
```

2. קראנו את שתי התמונות של הציפורים בעזרת cv2 (אפשר היה גם להשתמש ב PIL.Image):

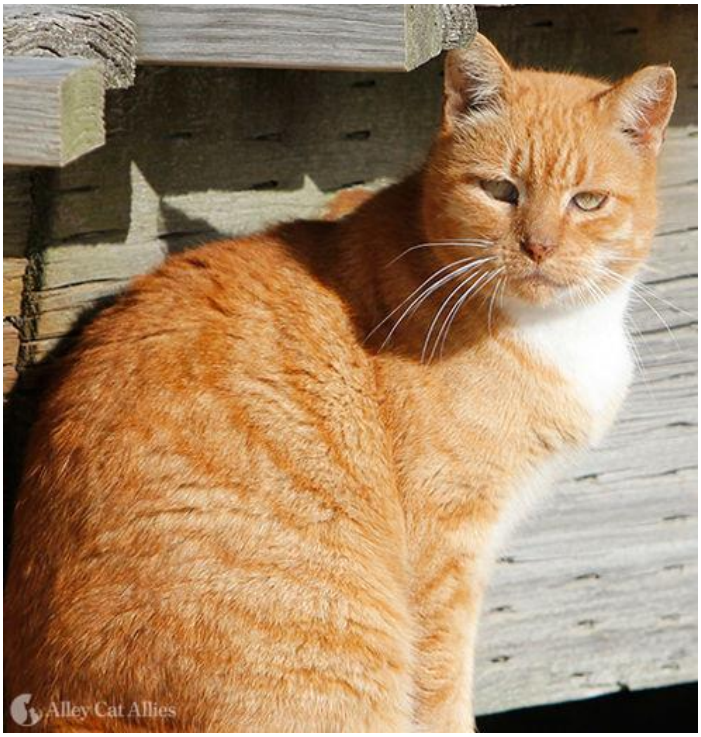
3. על התמונות האלה, הפעלנו טרנספורמציות של resize לגודל שאותו מצפה המודל ונירמול (לפי הממוצע והשונות של סט האימון של המודל).
4. נראה שהמודל עשה קלסיפיקציה נכונה (לפי תמונות אחרות של סוגי הציפורים שראינו באינטרנט). המודל חזה class 94 על יונק הדבש, ו 90

על הסוג של התוכי. VGG16 אומן על ILSVRC שהוא הקטנה של ImageNet ל 1000 מחלקות (https://neurohive.io/en/popular-networks/vgg16) .

לפי המיפוי שמצאנו באינטרנט מאינדקס לשם המלא של המחלקה (https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a), רואים את שמות הציפורים:

90: 'lorikeet',
91: 'coucal',
92: 'bee eater',
93: 'hornbill',
94: 'hummingbird',

5. לקחנו תמונה רנדומלית של חתול מהאינטרנט, הפעלנו את אותן טרנספורמציות של עיבוד מקדים, והכנסנו למודל לקבל פרדיקציה. קיבלנו זיהוי נכון של המחלקה (אינדקס 282 ב 1000 המחלקות ב ILSVRC):



281: 'tabby, tabby cat',
282: 'tiger cat',
283: 'Persian cat',

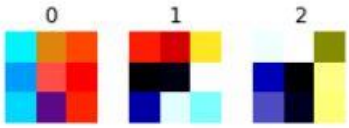
6. הפעלנו טרנספורמציות שונות:
   a. טרנספורמציה פרספקטיבית – בוצעה ע"י ספריית cv2. כמו שלמדנו בהרצאה, ע"י הגדרת מיפוי של 4 נקודות מתקבלת טרנספורציה פרספקטיבית. התמונה המקורית בגודל 494x518:

(150,150)->(0,0), (494-10,10)=(484,10)->(494,0), (10,518-10)=(10,508)->(0,518), (494-150,518-150)=(344,368)->(494,518)

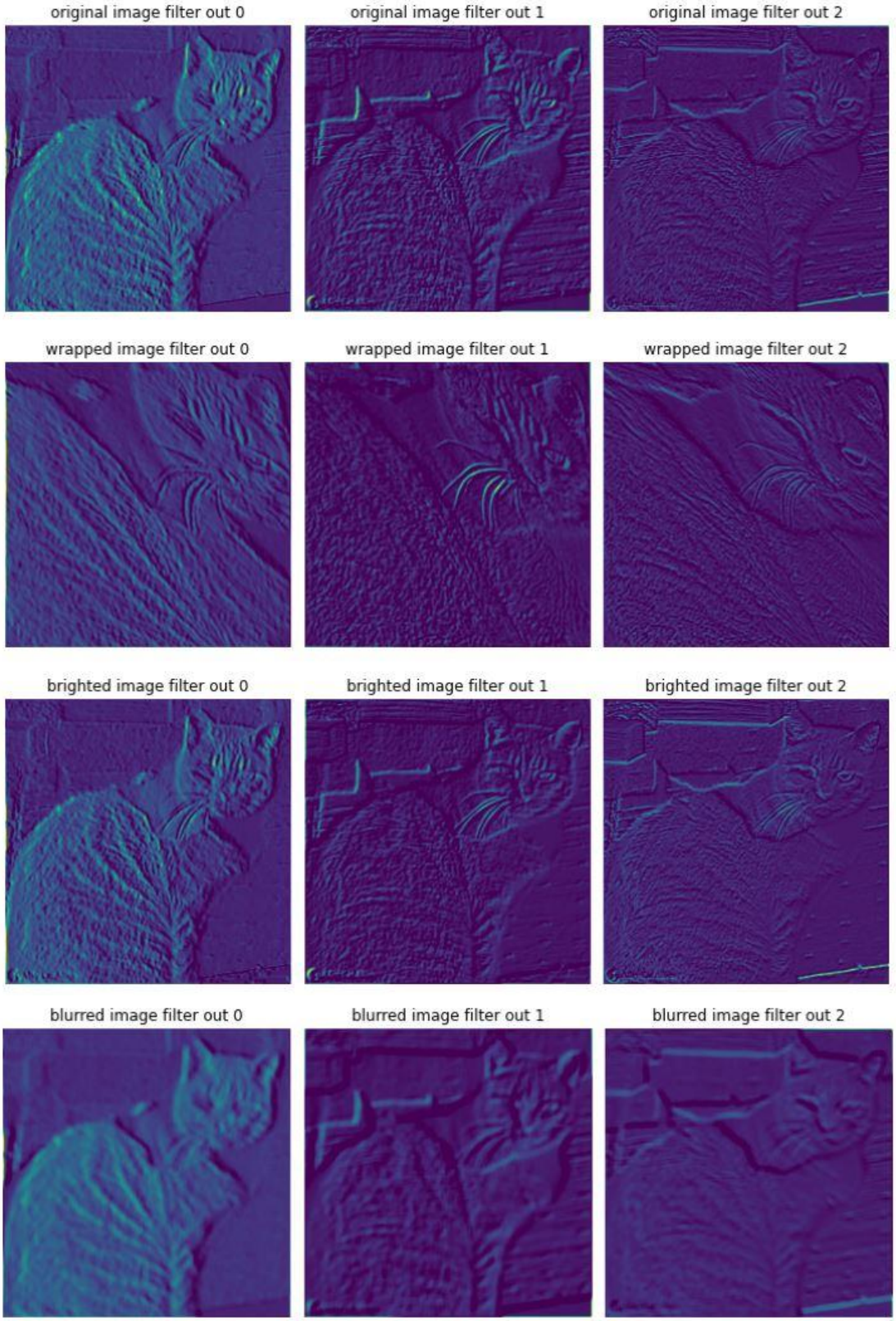
b. הגברת בהירות – המרנו את מרחב הצבע מ BGR ל HSV (קראנו ע"י cv2). והוספנו לערך V 90 (עד כדי רוויה בערך 255) והחזרנו חזרה ל BGR.

c. החלקה ע"י פילטר גאוסי – השתמשנו בפונקציית GaussianBlur (מספריית cv2) עם פילטר בגודל 11x11 וסטיית תקן 10.

7. לאחר resize לגודל שהמודל מצפה לו, המודל עדיין חזה נכון את המחלקה כמו קודם ולא נכשל למרות הטרנספורמציות החזקות על התמונות.
8. הוצאנו מהמודל את הפילטרים עצמם (הראנו תמונת RGB כמו בתרגול שכן הפילטרים הם במימד עומק כמו התמונה, 3):



הצגנו גם את התגובות של הפילטרים לתמונה המקורית שהורדנו מהאינטרנט ולכל התמונות של הטרנספורמציות:



בניסיון להסביר את התגובה של כל פילטר, ניתן לראות שמדובר על עיבוד ראשוני של התמונות בזהוי שפות בתדרים שונים.
פילטר 0 נראה שמזהה תדרים גבוהים כמו בפרווה של החתול וכך פילטרים מאוחרים יותר יכולים להסיק שאולי מדובר בפרווה (ולכן בסוף זה מוסיף לסיכוי שזהו חתול).

פילטר 1 נראה שמגיב חזק לשפם, יכול להיות שזהו פילטר שפה בכיוון מסוים בתדר מסוים שמתאים לכך. נראה גם שהוא מזהה שפות בכיוון Y. כאשר המעבר הוא מרמה נמוכה לגבוהה (צל לאור).
פילטר 2 נראה שמגיב חזק לשינויי שפה בציר Y גם כן, אולם בניגוד לפילטר הקודם התגובה חזקה כאשר עוברים מרמה גבוהה לנמוכה (אור לצל).

9. בחרנו בשכבת ה FC הלפני אחרונה בגודל 4096 נוירונים, ממש לפני השכבה הליניארית שממפה אותם ל 1000 מחלקות. העברנו את התמונות של הכלבים והחתולים עד לשכבה הזו (כולל) ושמרנו את הייצוג של התמונות לקובץ (כל תמונה בוקטור של 4096).
ביצענו FWD של התמונות דרך המודל עד לשכבות הקלסיפיקציה (שכבות ה FC ושכבת ה loss) .
תמונות של חתולים יגיעו לייצוג דומה כמו חתולים אחרים שכן נוירונים בשכבות אחרונות כבר אוספים מידע ברמה גבוהה המצביע על סיכוי גבוה לכל מחלקה. ולכן ניתן יהיה להפריד בקלות לאשכולות ומחלקות שונות בהסתמך על ייצוג שכזה.

```
(classifier): Sequential(
  (0): Linear(in_features=25088, out_features=4096, bias=True)
  (1): ReLU(inplace=True)
  (2): Dropout(p=0.5, inplace=False)
  (3): Linear(in_features=4096, out_features=4096, bias=True)
  (4): ReLU(inplace=True)
```

10. ה LinearSVC מודל שאימנו רק על 20 התמונות הצליח לסווג את תמונות ה testing שהורדנו מהאינטרנט בצורה מושלמת. זה לא מפתיע (למרות שיש לנו מעט מאוד מידע לאמן את ה SVM) שכן יש לנו רק שתי מחלקות להפריד ביניהם במסווג ליניארי כדוגמת SVM ווקטורי ה input עצמם כבר מכילים בתוכם את המאפיינים של כל מחלקה ע"פ מה שנלמד ב VGG על הרבה מאוד דוגמאות של חתולים וכלבים. בכך בעצם עשינו סוג של transfer learning ולכן ניתן להבדיל (גם ע"י unsupervised learning בשימוש clustering פשוט) בין המחלקות בצורה קלה יחסית.