אליאור נחמיה	204208037
עדן לוי	203618327

דו"ח משימה מס' 1 – סדנה מעשית בלמידה עמוקה

חלק א: Data analysis

מאגר הנתונים שבחרנו לפרוייקט זה הינו CIFAR-10 הידוע.

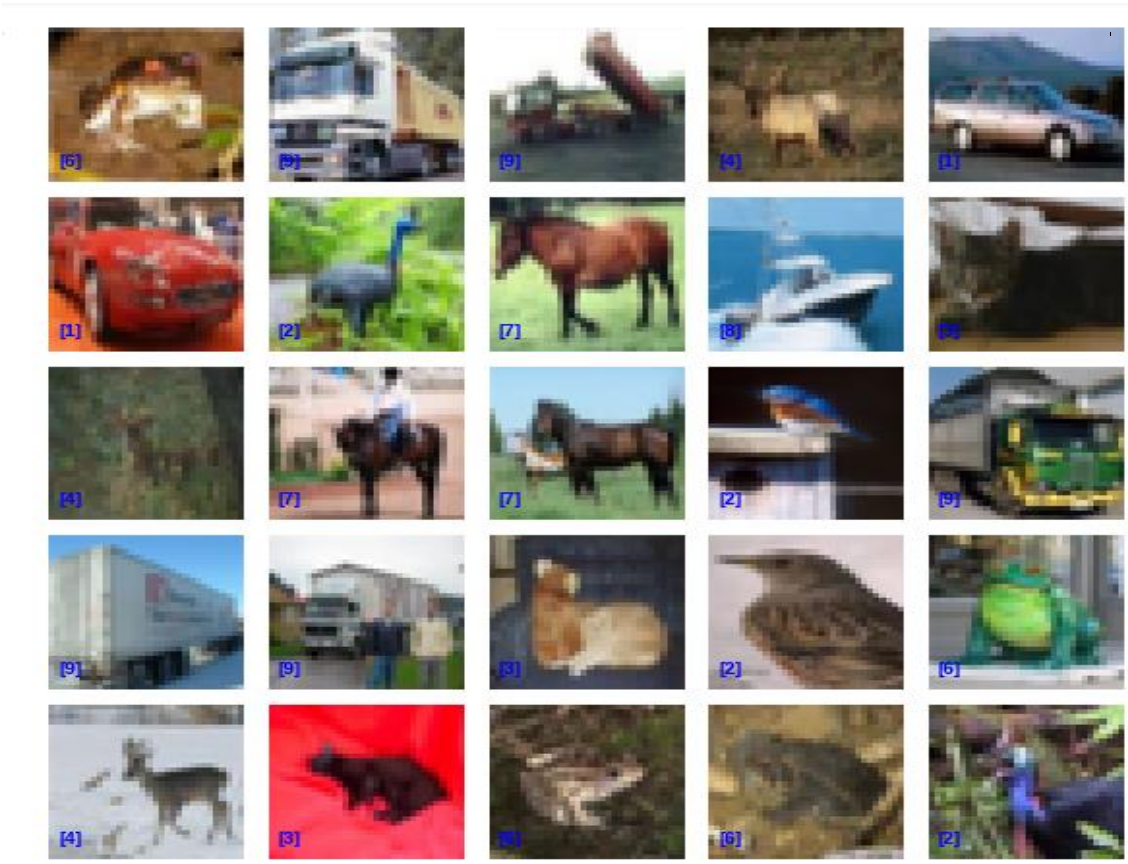
a. המאגר מכיל K60 תמונות מתוייגות.

b. כל תמונה במאגר בעלת הצורה הבאה - (32,32,3) כלומר גובה ורוחב של 32 פיקסלים כאשר מדובר בפורמט RGB ולכן כל פיקסל מיוצג ע"י 3 מימדים, בכל תמונה מופיע קלאס אחד בלבד כאשר בסך הכל ישנם 10 קלאסים (מטוס, מכונית, ציפור, חתול, אייל, כלב, צפרדע, סוס, ספינה, משאית). אין צורך לבצע preprocessing מיוחד ל-data, כיוון שהוא נטען בתור מטריצות של מספרים שמייצגים פיקסלים (0-255), בעזרת הפונקציה load_data. עם זאת, ניתן לנרמל את המספרים כך שיתכנסו מהר יותר (למשל לחלק ב-255 או להתפלגות נורמלית). יש אפשרות לעשות augmentation לתמונות במאגר, כמו למשל mirroring, flipping וכו', כדי לפתור בעיות של overfiting.

c. לא היה צורך להתעסק עם פילוג של ה-samples מכיוון שמאגר המידע מאוזן, כך שבכל אחד מ-10 הקלאסים קיימות 6000 תמונות. בסט האימון קיימות בדיוק 5000 תמונות מכל קלאס, מעורבבות רנדומלית, מה שמשאיר לסט הבדיקה 1000 תמונות מכל קלאס, שנבחרו רנדומלית.

d. לפי הבלוג הזה ב-GitHub.io, ב-3 המקומות הראשונים התקבלו תוצאות של 96.53%, 95.59% ו-94.16%. התוצאות נכונות לשנת 2016. ניתן למצוא קישורים למאמרים הרלוונטיים בבלוג עצמו ולחקור את העבודות שנעשו.

e. מכיוון שגודל התמונות הוא קטן (32x32), התמונות מטושטשות:



אליאור נחמיה 204208037
עדן לוי 203618327

חלק ב: Initial Neural Network

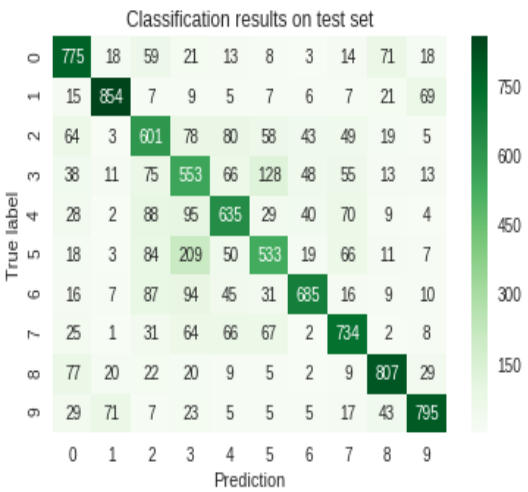
כפי שציינו קודם המאגר CIFAR10 מאוזן בצורה מוחלטת ולכן בחרנו להשתמש בשיטת בדיקה פשוטה יחסית באופן הבא: 10000-train-40000, validation-10000, test.

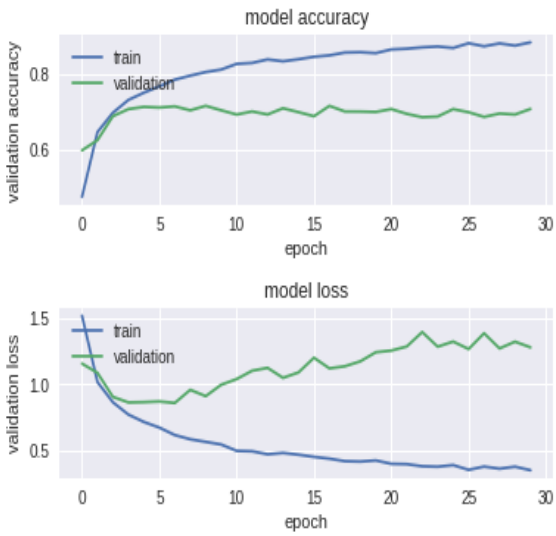
המודל הראשון נלקח מההרצאה ושונה מעט, הרעיון הנרכזי היה להעמיק את הרשת מספיק כדי ששהמודל יוכל ללמוד את המשימה ויחד עם זאת ניסינו גם להגיע לoverfiting כדי שנוכל משם רק לשפר. להלן המודל:

```
def build_CNN_1():
  model = Sequential()

  model.add(Conv2D(32,(3,3),activation='relu',padding='same',input_shape=(32,32,3)))
  model.add(Conv2D(32,(3,3),activation='relu',padding='same'))
  model.add(MaxPool2D())

  model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
  model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
  model.add(MaxPool2D())

  model.add(Conv2D(128,(3,3),activation='relu',padding='same'))
  model.add(Conv2D(128,(3,3),activation='relu',padding='same'))
  model.add(MaxPool2D())

  model.add(Flatten())
  model.add(Dense(10,activation='softmax'))

  model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

  return model
```

כפי שניתן לראות המודל מורכב משלושה בלוקים של קונבולוציה כאשר בתוך כל בלוק יש שתי שכבות של קונבולוציה של (3X3) עם ריפוד same ולסיום כל בלוק יש שכבת maxPool. כמות הפילטרים בכל בלוק עולה כאשר בראשון 32 בשני 64 ובשלישי 128, מתוך אינטואיציה שכך אנו מייצרים עוד מידע רלוונטי על סמך השכבה הקודמת אך לא נרצה להקטין את המידע אלה רק להגדיל אותו. לאחר שלושת הבלוקים אנו מבצעים flaten ולאחרי softmax. פונקציית ההפסד הוגדרה לcategory loss יחד עם adam optimizer, המודל הורץ על 30 epoch וואלידציה פשוטה כפי שהזכרנו מקודם.

התוצאות היו:






```
Model accuracy on test set is 69.72%
Model accuracy on validation set is 71.6%
Model accuracy on train set is 88.518%
```

אליאור נחמיה 204208037
עדן לוי 203618327

Train - acc 88% , loss 0.35. Validation - acc 70%, loss 1.27

Test - acc 69.7%

התוצאות מצביעות על מצב של overfiting קלאסי כאשר יש לנו פער של 18% דיוק בין האימון לאימות, אך יש לציין שקיבלנו תוצאות יפות ביחס למודל ראשוני שאותו נמשיך לשפר. כמו כן מניתוח התוצאות נראה שהמודל מתקשה בהבחנה בין תמונות של חתול לכלב כאשר הוא מסווג חתולים ככלבים בחלק מהמקרים אך לא הפוך.

נשים לב שיש לנו שתי בעיות להתייחסות, הראשונה היא שאנו נמצאים תחת overfiting מאוד חזק, והשניה היא שנרצה להעלות את רמת הדיוק של המודל על סט הבדיקה.1) תחילה נציע להשתמש בשיטת האוגמנטציה, במטרה להקטין את בעיית הovefiting תוך הנחה שיצירת השינויים בתמונה יוכלול לסייע למודל ללמוד את הפיצרים שאכן מהותיים לסיווג. 2) נציע הוספה של שכבות batch normalization כדי לסייע לרשת להתכנס מהר יותר. 3) שינוי הoptimizer או מטריקת הערכת המודל. 4) העמקת הרשת 5) הוספת שכבות dopout.

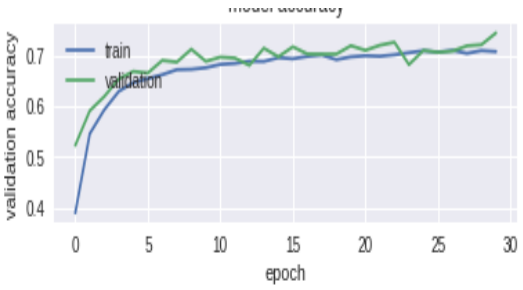
שיפור מס 1 - הוספת **data augmentation:** האוגמנטציה שבחרנו לבצע היא

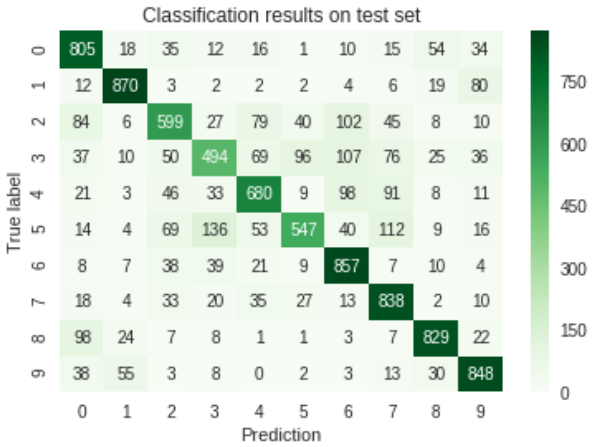
```
dataAug = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True
    )
```

שינוי זווית של 15, הזזה רוחבית 0.1, הזזה אורכית 0.1 ובנוסף הפיכה רוחבית. בחרנו בשינויים אלו כדי לשנות את התמונות אך יחד עם זאת לא בצורה חזקה מידי אשר לא תתאם את התפלגות התמונות המקוריות.

הרשת נשארה כפי שהיתה ורק הוספנו לה את האוגמנטציה להלן התוצאות:

אליאור נחמיה 204208037
עדן לוי 203618327






```
Model accuracy on test set is 73.67%
Model accuracy on validation set is 74.51%
Model accuracy on train set is 71.055%
```

Train - acc 71%, loss 0.839. Validation - 74.5%, loss 0.786.

Test - 73.67%

כפי שניתן לראות צמצמנו את הoverfiting בצורה מאוד משמעותית כאשר כעת ההפרש הוא רק של 3%. כמו כן ניכר שיפור של 4% בדיוק הכללי של המודל. התוצאות עדין אינן מספקות אותנו ונראה שעדין יש מספר סיווגים בעיתיים. פתרון אפשרי לשיפור המצב יכול ליהיות שימוש בנרמול פנימי בין השכבות כדי לסייע לרשת להתכנס וללמוד יותר מהר.

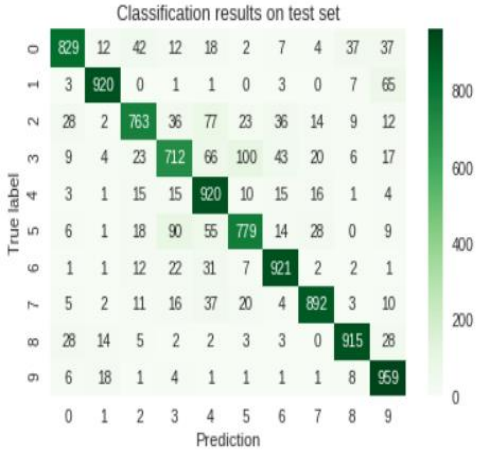
שיפור מס 2 - הוספת שכבות **batch normalization:** שילבנו ברשת הקיימת מספר שכבות של נרמול במטרה לגרום להתכנסות יותר טובה של הרשת יחד עם החלקת גורמי הרעש.

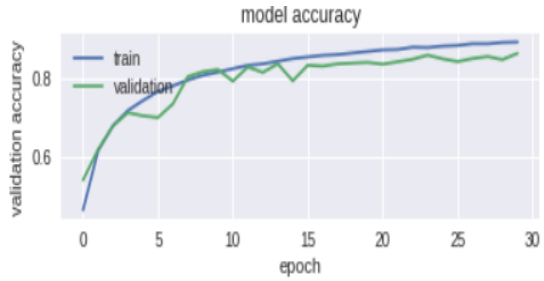
המודל החדש:

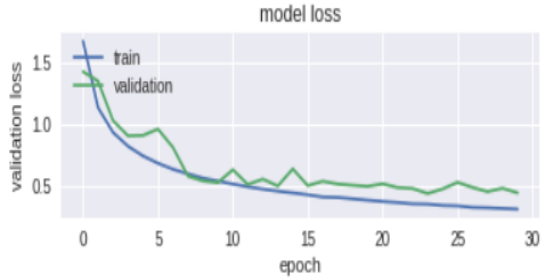
```
def build_model3():
    model = Sequential()
    model.add(Conv2D(32,(3,3),activation='relu',padding='same',input_shape=(32,32,3)))
    model.add(BatchNormalization())
    model.add(Conv2D(32,(3,3),activation='relu',padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPool2D())

    model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
    model.add(BatchNormalization())
    model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPool2D())

    model.add(Conv2D(128,(3,3),activation='relu',padding='same'))
    model.add(BatchNormalization())
    model.add(Conv2D(128,(3,3),activation='relu',padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPool2D())

    model.add(Flatten())
    model.add(Dense(10,activation='softmax'))
    model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
    model.summary()
    return model
```

אליאור נחמיה 204208037
עדן לוי 203618327

ולהלן התוצאות:








```
Model accuracy on test set is 86.1%
Model accuracy on validation set is 86.28%
Model accuracy on train set is 89.172%
```

Train - acc 89%, loss 0.3. Validation - acc 86.28%, loss 0.44.

Test - acc 86.1%.

כפי שניתן לראות שיפור זה היה קפיצת מדרגה מרשימה עבור המודל, כאשר הוא עלה ב13%. אנו מניחים שאכן ביצוע הנירמול איפשר למודל ללמוד את הפיצרים המשמעותים בצורה מהירה יותר ולהתאלם מרעשי רקע של דגימות בעייתיות.כמו כן חשוב לציין שהדיוק שהגענו אליו 86% על סט הבדיקה זה נתון מרשים גם ביחס למודלים הנוכחים בתחום עבור משימה זו.

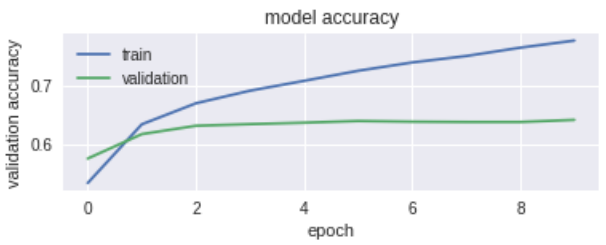
חלק ג: Transfer Learning and Feature Extraction

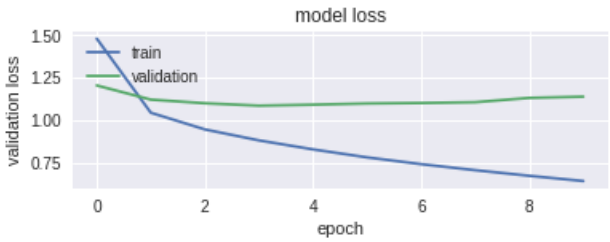
בחלק זה בחרנו להשתמש ברשת הידועה 16VGG, מהסיבה הפשוטה שהיא יחסית לא עמוקה והיא כן מעט מזכירה את הארכיטקטורה בה השתמשנו בחלק הקודם. בעזרת ה-API של keras הארכיטקטורה הייתה מוכנה מראש, ואף הורדה של המשקלים של הרשת שאומנו על מאגר הנתונים של ImageNet הייתה מאוד פשוטה. בכל הניסויים הבאים התאמנו את שכבת הקלט של הרשת לקלט של המאגר 10-CIFAR שהוא (32,32,3), הסרנו את השכבה האחרונה ב-16VGG של הפרדיקציות בעזרת include_top=False ובמקומה שמנו שכבת פרדיקציה מותאמת לבעיה שלנו. נמנענו מלהשתמש ברשתות מורכבות ועמוקות יותר כדוגמאת ResNet50, מכיוון שזמני החישוב היו ארוכים מדי לעבודה הזו.

אליאור נחמיה 204208037
עדן לוי 203618327

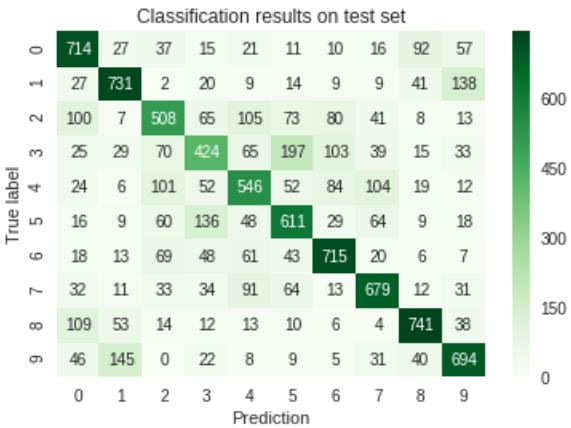
בנוסף, צריך להתייחס לעובדה שבגלל שאנחנו משתמשים ברשת שאומנה על input במימדים גדולים מאוד יחסית לבעיה שלנו (224x224 לעומת 32x32), אנחנו בעצם מפסידים הרבה מבחינת המידע שקיים במשקולות. לא הצלחנו פתרונות טובים כדי להתמודד עם הבעיה הזו, אך כמובן שכשהגדרנו את המימדים, הרשת התאימה את עצמה מבחינת הפילטרים, כך יצא שבסוף לאחר כל שכבות הקונבולוציה, קיבלנו 512 נוירונים, שלעומת הרשת המקורית (2048) זה נראה לנו בסדר.

בניסיון הראשון, מכיוון שהארגומנט include_top=False מסיר גם את 2 השכבות ה-FC מלבד ה-softmax, הוספנו שתי שכבות כאלה ידנית, ואימנו את המודל כדי לקבל תוצאה ראשונית. בנוסף השתמשנו בפונקציה המוכנה שבאה עם ה-API של VGG16 שעושה לנתונים preprocess, מתוך הנחה שהפעולות שנעשות שם אכן מתאימות את הנתונים למבנה הרשת. שימוש בפונקציה הזו הביא לתוצאות יותר טובות מאשר חילוק ב-255 או נרמול Z.






```
Model accuracy on test set is 63.63%
Model accuracy on validation set is 64.23%
Model accuracy on train set is 77.64%
```



Layer (type)	Output Shape	Param #
input_4 (InputLayer)	(None, 32, 32, 3)	0
block1_conv1 (Conv2D)	(None, 32, 32, 64)	1792
block1_conv2 (Conv2D)	(None, 32, 32, 64)	36928
block1_pool (MaxPooling2D)	(None, 16, 16, 64)	0
block2_conv1 (Conv2D)	(None, 16, 16, 128)	73856
block2_conv2 (Conv2D)	(None, 16, 16, 128)	147584
block2_pool (MaxPooling2D)	(None, 8, 8, 128)	0
block3_conv1 (Conv2D)	(None, 8, 8, 256)	295168
block3_conv2 (Conv2D)	(None, 8, 8, 256)	590080
block3_conv3 (Conv2D)	(None, 8, 8, 256)	590080
block3_pool (MaxPooling2D)	(None, 4, 4, 256)	0
block4_conv1 (Conv2D)	(None, 4, 4, 512)	1180160
block4_conv2 (Conv2D)	(None, 4, 4, 512)	2359808
block4_conv3 (Conv2D)	(None, 4, 4, 512)	2359808
block4_pool (MaxPooling2D)	(None, 2, 2, 512)	0
block5_conv1 (Conv2D)	(None, 2, 2, 512)	2359808
block5_conv2 (Conv2D)	(None, 2, 2, 512)	2359808
block5_conv3 (Conv2D)	(None, 2, 2, 512)	2359808
block5_pool (MaxPooling2D)	(None, 1, 1, 512)	0
flatten_4 (Flatten)	(None, 512)	0
dense_5 (Dense)	(None, 128)	65664
dense_6 (Dense)	(None, 128)	16512
dense_7 (Dense)	(None, 10)	1290

Total params: 14,798,154
Trainable params: 83,466
Non-trainable params: 14,714,688

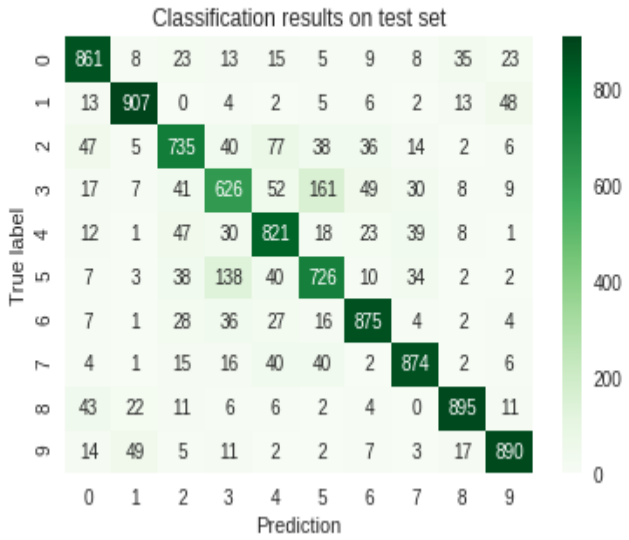
בפועל בנסיון זה הגענו רק ל-77% דיוק train ו-63% validation, מה שמראה על overfitting לנתוני ה-train. עם זאת, עקב מגבלת הזמן ונפח העבודה, הנתונים מתייחסים לאימון של רק 10 epochs. למרות זאת, ניתן לראות בגרפים שעקומות ה-validation עם שיפוע די אפסי ולכן אנחנו לא בטוחים שעוד epochs היה עוזר במקרה שכזה.

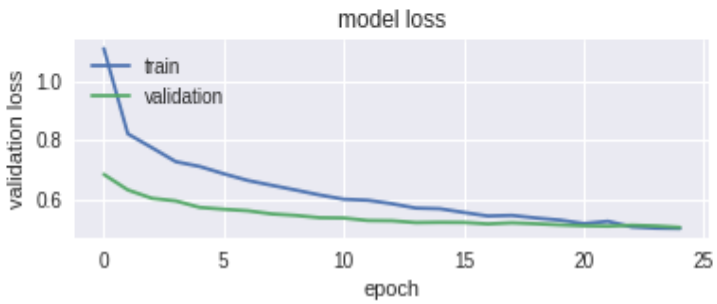
אליאור נחמיה 204208037
עדן לוי 203618327

בנסיונות הבאים ניסינו להפחית את ה-overfitting וגם להגיע לתוצאות טובות יותר ב-validation. הגענו לארכיטקטורה סופית אותה נציג. לאחר נסיונות רבים (מאוד) אחרים, בחרנו "לחתוך" את רשת ה-VGG16 בשכבה מעט פנימית ולא רק בשכבה האחרונה. הקפאנו את המשקולות של שכבות הרשת שנשארו, והוספנו שכבה אחת של FC עם 256 נוירונים. בנוסף, השתמשנו ב-dropout יחסית גדול, ששימש כעזרה לבעיה גם הוא. הגדלנו את שכבת ה-FC לפני הפרדיקציות מהפעם הקודמת (128 לעומת 256) ואף הוספנו שכבות BatchNormalization שיעזרו בהתכנסות של ה-gradients. בחרנו לא להשתמש ב-data augmentation מהסיבה הפשוטה שבדקנו את התרומה של זה והיא כלל לא תרמה לדיוק המודל.

להלן תוצאות האימון של מודל ה-Transfer Learning לאחר השיפורים שהוסברו מעלה:

```
Model accuracy on test set is 82.1%
Model accuracy on validation set is 82.92%
Model accuracy on train set is 82.317%
```





אליאור נחמיה 204208037
עדן לוי 203618327

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 32, 32, 3)         0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 32, 32, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 32, 32, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 16, 16, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 16, 16, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 16, 16, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 8, 8, 128)         0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 8, 8, 256)         295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 8, 8, 256)         590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 8, 8, 256)         590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 4, 4, 256)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 4096)              0
_________________________________________________________________
batch_normalization_3 (Batch (None, 4096)              16384
_________________________________________________________________
dropout_3 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               1048832
_________________________________________________________________
batch_normalization_4 (Batch (None, 256)               1024
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                2570
=================================================================
Total params: 2,804,298
Trainable params: 1,060,106
Non-trainable params: 1,744,192
_________________________________________________________________
Train on 37500 samples, validate on 12500 samples
```

ניתן לראות שדיוק המודל יציב בין ה-train, validation, test ועומד על כ-82%. בגרפים ניתן לראות שאכן בעיית ה-overfitting מהמודל הקודם נפתרה – המודל הוא perfectly fit, ובנוסף הצלחנו להעלות משמעותית את אחוזי הדיוק של המודל על סט ה-validation וה-test.

כל הניסיונות שלנו נעשו באופן כזה שאנחנו מוסיפים רק שינוי אחד למודל ובודקים את השפעתו. ברור לנו שמבחינה סטטיסטית יכולים להיות קשרים בין השינויים שניסינו ואם שינוי מסויים לא עזר לבדו, הוא יכל לעזור בשילוב של שינוי אחר (תלויות בין שינויים), אך מפאת הזמן הקצר בחרנו שלא להתייחס לזה.

אליאור נחמיה 204208037
עדן לוי 203618327

Feature Extraction

לקחנו את השכבה לפני ה-softmax ברשת האחרונה שלנו מה-transfer learning model, שהיא בעצם שכבת dropout. התלבטנו איזו שכבה לקחת מבין האחרונות:

```
_________________________________________________________________
dense_3 (Dense)              (None, 256)               1048832
_________________________________________________________________
batch_normalization_4 (Batch (None, 256)               1024
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                2570
=================================================================
```

לא היינו בטוחים ש-feature extraction משכבה כמו dropout או batchnorm יצא תקין, אך עשינו ניסויים וקיבלנו תוצאות דומות על כל אחת מ-3 השכבות האחרונות, לכן בחרנו שרירותית בשכבת ה-dropout.

ביצענו את ה-Feature Extraction על כמה מודלים קלאסיים שונים של ML:

נתחיל ב-Logistic Regression. אלו הן התוצאות שקיבלנו:

```
######### Testing Logistic Regression Model #########
Model accuracy on train set is 93.33%
Model accuracy on test set is 82.01%
####################################
```

ניתן לראות שחזרנו לבעיית overfitting, אך בכל זאת אחוזי הדיוק על ה-test נשאר זהה ויחסית גבוה.

נעבור ל-Random Forest עם פרמטר של מקסימום 50 עצים, עומק מקסימלי של עץ 6, ומינימום 3 דוגמאות בעלה. קיבלנו את התוצאות הבאות:

```
######### Testing Random Forest Model #########
Model accuracy on train set is 78.71%
Model accuracy on test set is 72.72%
####################################
```

גם במקרה הזה יש overfitting מסוים, אך יותר קטן מהקודם, ואחוזי הדיוק הן של ה-train והן של ה-test ירדו משמעותית.

נסיים במודל של Decision Tree עץ החלטה בודד לעומת Random Forest עם פרמטר של מקסימום עומק של 20. תוצאות:

```
######### Testing Decision Tree Model #########
Model accuracy on train set is 95.164%
Model accuracy on test set is 63.11%
####################################
```

לסיכום פרק ה-Feature Extraction, מודל ה-Logistic Regression היה בעל הדיוק הכי גבוה על ה-validation, למרות ה-overfitting. נבחר בו בדיוק מהסיבה הזו, כי נעדיף דיוק גבוה על ה-validation בכל מקרה.