פרויקט סופי-
יישומים בלמידה חישובית
קלסיפיקציה לפי זני ציפורים

1. רקע:

זיהוי מיני ציפורים זו משימה שמאתגרת את היכולות החזותיות גם של מומחי מביני עניין, וגם של מחשבים. הבעיה שבחרנו היא סיווג של מיני ציפורים על פי תמונות שונות. בעיה זו בעלת חשיבות רבה עבור מומחים המתעסקים עם ציפורים, ויש משמעות לסיווגם נכונה ולאיסוף הנתונים בצורה מהימנה. זה רלוונטי גם עבורם אך גם עבור חברות שעסוקות בפיתוח חדשני שיכול לסכן ציפורים. למשל - חברות שמייצרות אנרגיה ע"י תחנות רוח, ישנה סכנה להתנגשות בציפורים, דבר שיכול לערער את המאזן הביולוגי בטבע ע"י איום הכחדה של מיני ציפורים רבות .עבור פתרון הבעיה, נבנה מסווג בארכיטקטורת)cnn עם שיפורים שיתוארו בהמשך(בסיסי לסיווג תמונות של ציפורים ונשפר את ביצועיו, לאחריו נשווה עם מסווג נוסף כחלק מהניסויים על מנת לשפר ביצועים.

2. מאגר נתונים

מאגר הנתונים שלנו מכיל כ- 43622 תמונות עבור train, 1500 עבור הtest, ועוד 1500 עבור ה- validation.
יש כ-3 עמודות, אשר מכילות את הנתיבים, אנוטציות- כל ציפור מסווגת לזן מסוים, ולאן התמונה מסווגת.
ניתן לראות בתמונה דוגמא לשתי תמונות מהמאגר.

train/AFRICAN CROWNED CRANE/001.jpg	AFRICAN CROWNED CRANE	train
train/AFRICAN CROWNED CRANE/002.jpg	AFRICAN CROWNED CRANE	train

3. אלגוריתמים

הערה- על מנת לפתוח את המחברות יש ללחוץ על הקישור, ולאחר מכן על הכפתור copy my Edit בצד ימין למעלה.

<u>מודל ראשון</u>

הקישור- https://www.kaggle.com/noaelishmereni/bird-classification-first-notebook

- מודל שלישי שננסה עבור בעיה זו, גם הוא הורץ על גבי GPU.
במודל זה נעשה שימוש בkeras, tensorflow. Keras זו ספרייה חזקה בפייתון, שמספקת ממשק נקי ליצור מודלים של למידה עמודה יחד עם הטכנולוגיה של tensorflow.

1. יבוא ספריות: numpy, tensorfloe, matplotlib
2. במודל זה ניעזר באפשרות לבצע שמירת היסטוריה לכל אפוק, ולהוציא עוד מידע ונתונים. לכן בשלב השני אכן נבצע plot שכזה. (אפשר בקלות גם להדפיס היסטוריה, ומהם ניתן ליצור plots)
3. טעינת מידע וחלוקה לסט אימון,טסט וואלידציה.
4. בשלב הבא נבקש להראות מספר תמונות ממאגר הנתונים.
5. בשלב הבא, ניצור את המודל-

נעשה כאן שימוש ב- efficientnetB3 בתהליך ה-transfer learning, שזו משפחה של מודלים cnn שיש לה תוצאות גבוהות של יעילות ודיוק. כמו כן נבצע batchnormlization עם מומנטום (0.99 הוא הערך הנבחר מבין הטווח שהרצנו), כלומר כעוד שכבה נוספת כדי ליצור את המודל. ניעזר גם ברגולזיציה , relu, softmax ובהמשך נבצע dropout:

```
tf.keras.applications.efficientnet.EfficientNetB3(
    include_top=True, weights='imagenet', input_tensor=None,
    input_shape=None, pooling=None, classes=1000,
    classifier_activation='softmax', **kwargs
)
```

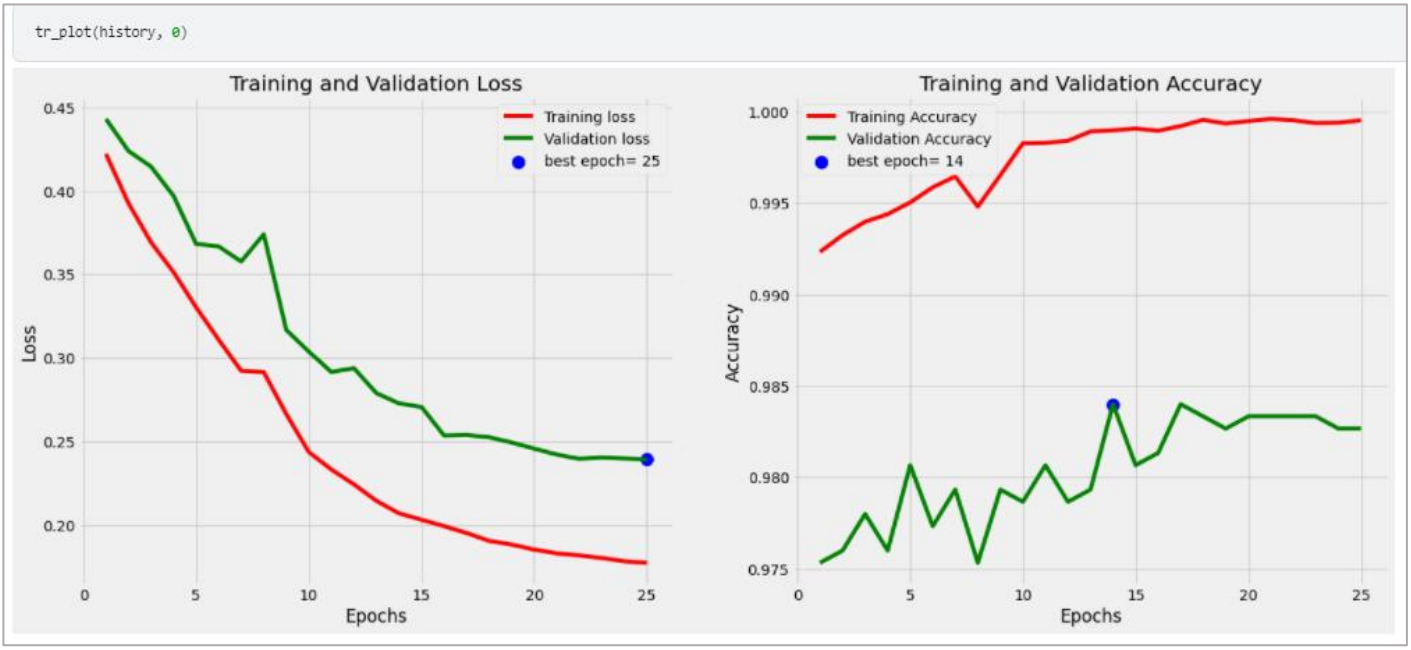
```
input=Input(shape=img_shape)
x=tf.keras.applications.EfficientNetB3(include_top=False, weights="imagenet",
x=tf.keras.layers.BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001 )(x)
x = Dense(256, kernel_regularizer = regularizers.l2(l = 0.016),activity_regularizer=regularizers.l1(0.006),
                bias_regularizer=regularizers.l1(0.006) ,activation='relu')(x)
x=Dropout(rate=.4, seed=123)(x)
output=Dense(class_count, activation='softmax')(x)
model=Model(inputs=input, outputs=output)
model.compile(Adamax(lr=.001), loss='sparse_categorical_crossentropy', metrics=['accuracy']) # note labels are i
```

```
Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb3_notop.h5
43941888/43941136 [==============================] - 0s 0us/step
```

בשלב הבא נעשה שימוש ב- keras.callbacks.ReduceLROnPlateau, כלומר, נשפר ע"י הקטנת קצב הלמידה כאשר המטריצה מפסיקה להשתפר. מודלים לעיתים מרוויחים כשאר מורידים את קצב הלמידה בפקטור של 2-10 כאשר המודל מתייצב, ולכן נוכל להגיד שכאשר אין עוד שיפורים לבצע, אכן נקטין את קצב הלמידה. בנוסף, נאמן את המודל. כאשר נריץ 25 אפוקים.

```
Epoch 21/25
1455/1455 [==============================] - 255s 175ms/step - loss: 0.1828 - accuracy: 0.9996 - val_loss: 0.2419 - val_accuracy: 0.983
3
Epoch 22/25
1455/1455 [==============================] - 257s 176ms/step - loss: 0.1815 - accuracy: 0.9995 - val_loss: 0.2394 - val_accuracy: 0.983
3
Epoch 23/25
1455/1455 [==============================] - 258s 177ms/step - loss: 0.1800 - accuracy: 0.9994 - val_loss: 0.2401 - val_accuracy: 0.983
3

Epoch 00023: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
Epoch 24/25
1455/1455 [==============================] - 258s 177ms/step - loss: 0.1780 - accuracy: 0.9994 - val_loss: 0.2396 - val_accuracy: 0.982
7

Epoch 00024: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
Epoch 25/25
1455/1455 [==============================] - 258s 177ms/step - loss: 0.1771 - accuracy: 0.9995 - val_loss: 0.2390 - val_accuracy: 0.982
7
```

ניתן לראות כאן את האחוזים של האפוקים האחרונים בהרצה.



ניתן לראות בגרפים, שבכל שכמות האפוקים עולה כך ה loss יורד. בנוסף ה accuracy עולה .ניתן לראות שלשימוש ברגוליזציה יש חשיבות רבה ב מודל שלנו, כיוון שבעזרת זה נוכל למנוע)overfitting כלומר מודלים שעובדים טוב מאוד על ה, train אך פחות טוב על ה(. test רגולריזציה זו שיטה ששולטת במודל . כיוון שכל תכונה מהווה משקולת מסוימת, ככל שיהיו יותר משקולות לתמונה יש סיכוי ל. overfitting לכן רגולריזציה מפחיתה את ה"עומס" שיש למשקולות האלה - כלומר יש להן פחות השפעה על פונקציית ה, loss- כלומר על המרחק בין הלייבל האמיתי למה שהצמדנו לתמונה.

6. בשלב זה נבדוק את ההצלחה על הtest, נראה שאלו התוצאות שנקבל:

```
50/50 [==============================] - 3s 62ms/step
there were 1474 correct predictions in 1500 tests for an accuracy of  98.27 %

Classification Report:
----------------------
                          precision    recall  f1-score   support

      AFRICAN CROWNED CRANE       1.00      1.00      1.00         5
          AFRICAN FIREFINCH       1.00      1.00      1.00         5
                  ALBATROSS       1.00      1.00      1.00         5
       ALEXANDRINE PARAKEET       1.00      1.00      1.00         5
            AMERICAN AVOCET       1.00      1.00      1.00         5
          AMERICAN BITTERN        1.00      1.00      1.00         5
             AMERICAN COOT        1.00      1.00      1.00         5
         AMERICAN GOLDFINCH       1.00      1.00      1.00         5
           AMERICAN KESTREL       1.00      1.00      1.00         5
```

כלומר, קיבלנו כ1474 תמונות נכונות מתוך 1500, כאשר הaccuracy עומד על 98.27.

ראינו שכאן ביצענו 25 אפוקים, עם אחוז הצלחה גבוה למדי של 98 אחוז.

4. ניסוי ובדיקת משתנים שונים

מודל שני

הקישור- https://www.kaggle.com/dorindomin/bird-classification-second-notebook

- מודל נוסף שננסה עושה אף הוא שימוש ב- transfer learning.
יש לציין שהרצנו את האלגוריתם על GPU.

1. ייבוא ספריות- pandas,tensorflow,numpy.
2. השלב השני- טעינת המידע (אותו דאטא כפי שתיארנו) וחלוקה לסט אימון,טסט וואלידציה.
3. בניית המודל- ניעזר בספריית tensorflow.

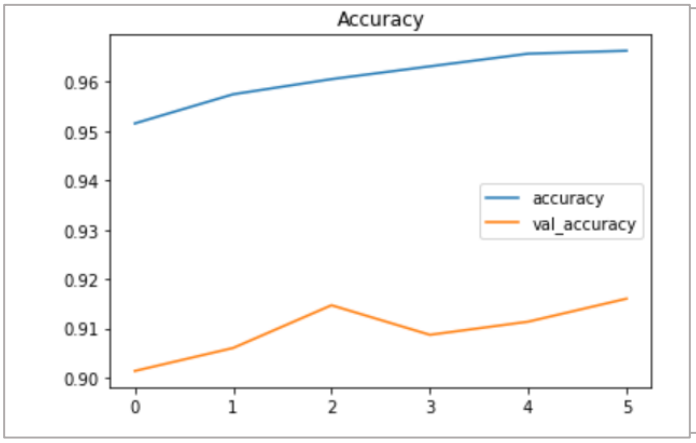
גם כאן מתבצע transfer learning, הפעם באמצעות MobileNetV2 שלוקח משקלים שאומנו מראש ב-Imagenet ומבצע pooling של ממוצע. כמו גם שימוש בפונק' אקטיבציה relu, עם אופטימייזר adam, categorical-crossentropy, פונ' softmax ונאמן במשך כ-6 אפוקים בהתחלה.

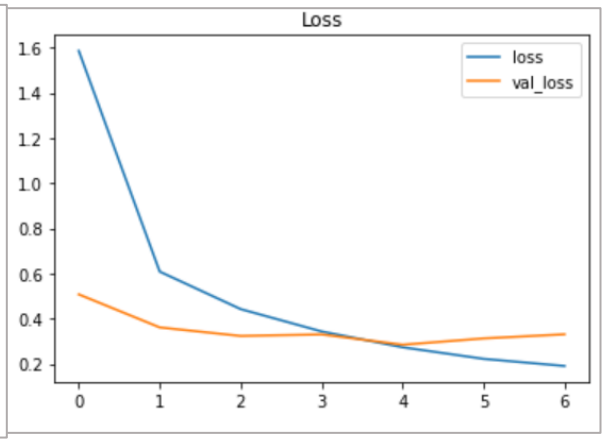
MobileNetV2 היא רשת cnn שמשתמשת ב-"inverted residual blocks" והמימוש שלה ב-keras (בו השתמשנו) דומה למקורי. המודל הזה מהיר יותר, דורש פחות פעולות תוך כדי אחוזי דיוק גבוהים יותר. כמו כן, רשתות cnn הן הרשתות הנפוצות ביותר שהכרנו לשימוש בבעיית Image classification ולכן השימוש ברשת כזו בפתרון הבעיה שלנו מתבקש.

(תיאור summary מופיע במחברת):

```
# transfer learning

x = tf.keras.layers.Dense(128, activation='relu')(pretrained_model.output)
x = tf.keras.layers.Dense(128, activation='relu')(x)

outputs = tf.keras.layers.Dense(300, activation='softmax')(x) # 300

model = tf.keras.Model(inputs=inputs, outputs=outputs)
```

אחוזי הצלחה:





```
Accuracy on the test set: 93.27%
47/47 [==============================] - 5s 89ms/step - loss: 0.3003 - accuracy: 0.9327
[0.3003265857696533, 0.9326666593551636]
```

נשים לב, להבדיל מהמודל הקודם, נדרשו למודל הנ"ל פחות אפוקים כדי להגיע לאחוזי הצלחה דומים לקודם. כאן תוך 6 אפוקים הצלחנו ללמוד מסט האמון ולקבל אחוזי הצלחה > 96, ועל סט הוואלידציה באזור ה-92 אחוזי הצלחה. לגבי הלוס, קיבלנו התנהגות מתונה יותר, כאשר הלוס 'מקפיד לרדת' תוך הלמידה.
הוספת drop out לא סייעה בשיפור המודל. גם שימוש ב- sgd כאופטימייזר עם מומנטום (לא הושג שיפור משמעותי).
בנוסף, כיוון שבדקנו כבר במודל הקודם כי אין תמונות פגומות (קיבלנו בחישוב 0 תמונות פגומות), לא ביצענו בדיקה זו גם פה שכן מדובר באותו דאטא.
עבור 20 אפוקים, הצלחנו להגיע אפילו ל-96 אחוזי הצלחה בהערכת המודל (model evaluate).

מודל שלישי

הקישור- https://www.kaggle.com/dorindomin/bird-classification-third-notebook

אלגוריתם זה מתמקד ברשתות נוירונים עם transfer learning.

- יש לציין שהרצנו את האלגוריתם על GPU.

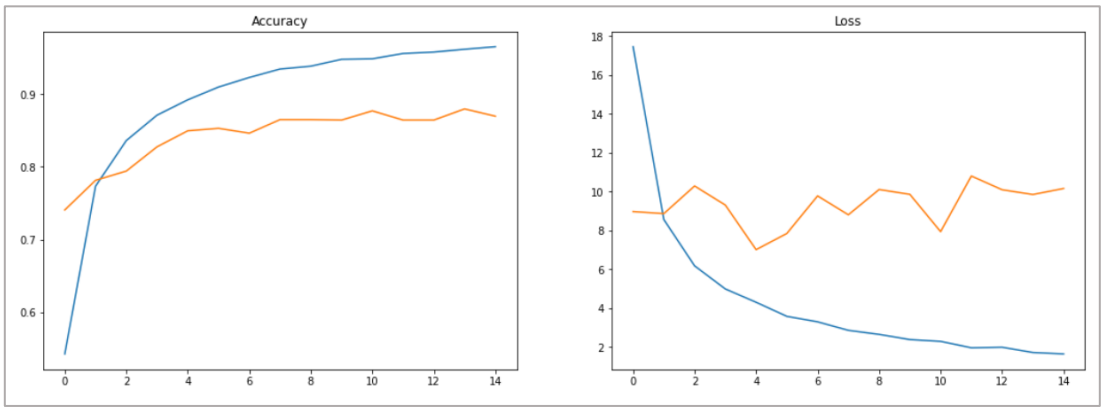
1. ייבוא ספריות- Pandas, numpy, matplotlib.
2. השלב השני- טעינת המידע וחלוקה לסט אימון,טסט וואלידציה.
מדובר בדאטא המכיל 300 זני ציפורים.
התמונות בגודל 224*224 ב-3 צבעים, בפורמט jpg.
42262 תמונות לסט האימון, 1500 לסט הטסט ו- 1500 לסט הוואלידציה. אין כפילויות.
בהמשך לשלב זה נרצה לראות שאכן יש לייבלים מתאימים לכל תמונה, נציג כ-20 תמונות.
לפני שנבנה את המודל, נרצה לבצע "ניקוי נתונים" כדי לוודא שאין תמונות פגומות. דבר זה מתבצע ע"י חישוב סטיית תקן בין כל הפיקסלים.

```
Corrupted images #: 0
```

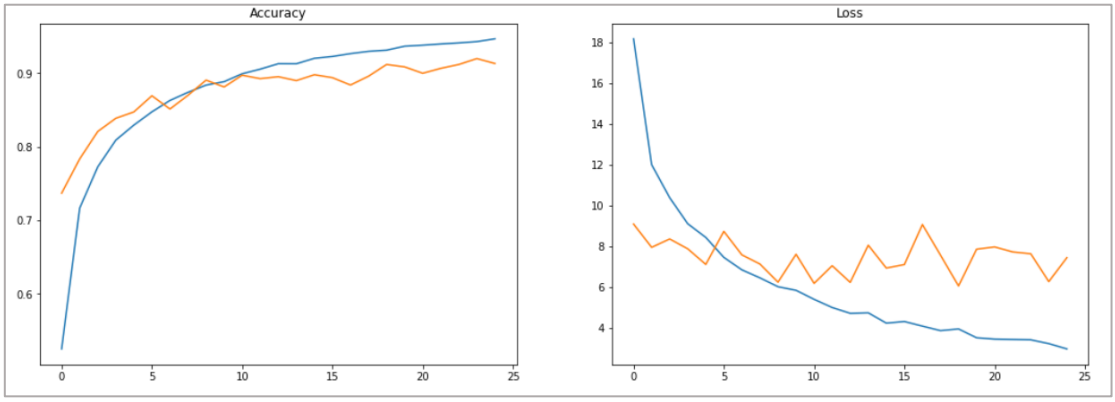
3. בניית המודל- ניעזר בספריית tensorflow.
נבנה resnet עם אופטימייזר adam, categorical-crossentropy, פונ' softmax, ומשקלים שאומנו מראש ב-Imagenet, נאמן במשך 25 אפוקים:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_resnet_v2 (Functio (None, 5, 5, 1536)        54336736
_________________________________________________________________
flatten (Flatten)            (None, 38400)             0
_________________________________________________________________
dense (Dense)                (None, 300)               11520300
=================================================================
Total params: 65,857,036
Trainable params: 11,520,300
Non-trainable params: 54,336,736
```

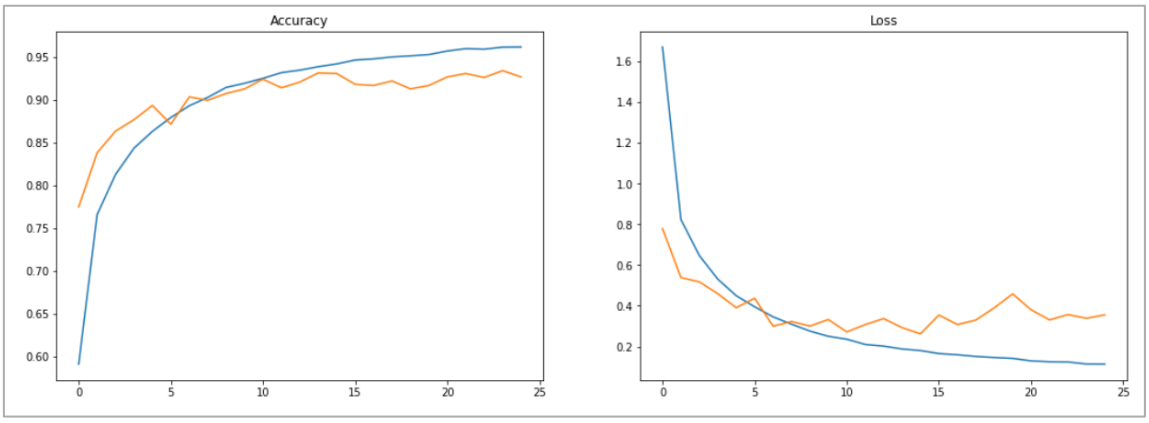
הוכח כי אימון על Imagenet נותן למודלים 'boost' גדול ודורש 'רק' ביוונון עדין של משימות זיהוי אחרות. לכן בחרנו לנסות להשתמש בו כחלק מתהליך ה-transfer learning.
בהמשך ישנם גרפי loss,accuracy כאשר בכחול תוצאות על סט האימון ובכתום על וואלידציה.



בשביל לשפר את יכולת המודל ללמוד את הייצוג של ציפורים (ולהתמודד עם decreasing overfitting), נשתמש ב-data augmentation. זה יעזור לנו בנוסף בתמונות בהן הציפור נמצאת במרכז התמונה, מה שיכול להוביל את המודל להתמקד באזור זה של התמונה ולפספס אלמנטים רלוונטיים מסביב.

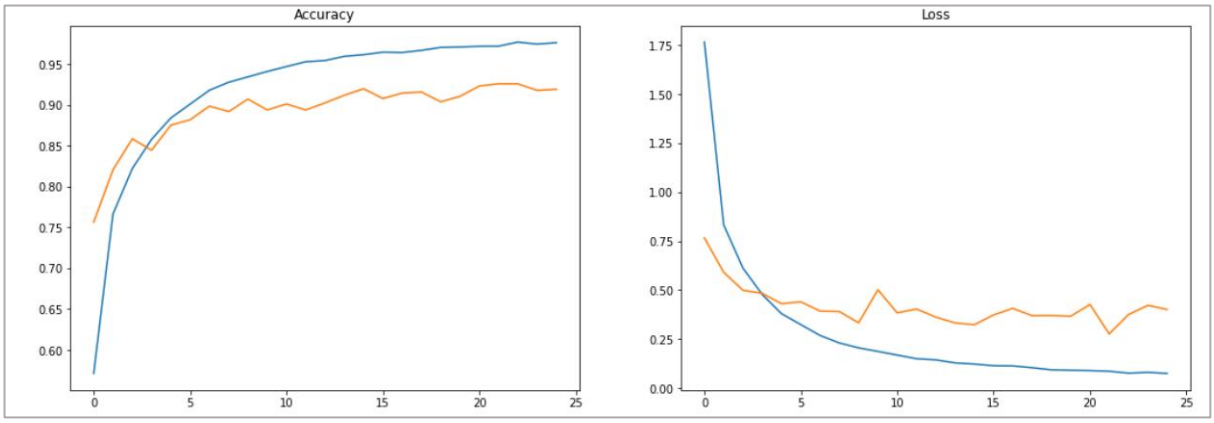


ניתן לראות שיפור משמעותי בנכונות המודל על סט האלידציה וכמו גם ירידה בערכי הלוס.
נבחן שיפור המודל ע"י הוספת שכבות, בהן נעבוד עם relu,batchnormalization:



```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_resnet_v2 (Functio (None, 5, 5, 1536)        54336736
_________________________________________________________________
flatten_2 (Flatten)          (None, 38400)             0
_________________________________________________________________
dense_2 (Dense)              (None, 1950)              74881950
_________________________________________________________________
batch_normalization_609 (Bat (None, 1950)              7800
_________________________________________________________________
dense_3 (Dense)              (None, 300)               585300
=================================================================
Total params: 129,811,786
Trainable params: 75,471,150
Non-trainable params: 54,340,636
_________________________________________________________________
```

אם נשווה בין הגרפים, ניתן לראות שיפור גם בנכונות המודל וגם בחישוב הלוס. כמובן שלא תמיד הרחבת הרשת בעוד שכבות יעבוד (כמו שלמדנו, ניתן להגיע בין היתר לאובר פיטינג).
שיפור נוסף שננסה הוא :dropout. מבין טווח הערכים 0.1-0.9 הערך שנתן שיפור משמעותי הוא 0.2.



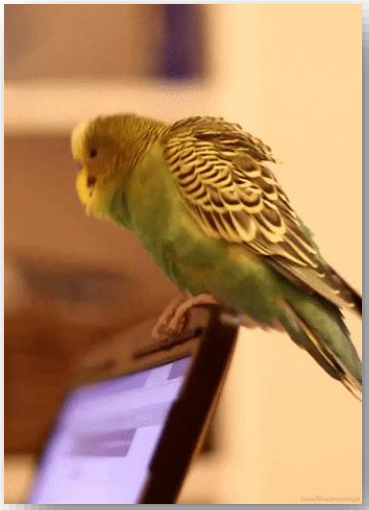
לאחר כל השיפורים:

```
layers_model.evaluate(test_generator,use_multiprocessing=True,workers=10)
```

```
47/47 [==============================] - 8s 145ms/step - loss: 0.2172 - accuracy: 0.9407
[0.21719253063201904, 0.940666675567627]
```

5. מסקנות

- קאגל יכול להיות מתיש למחשב ממוצע. כמעט והשתמשנו בכל ה-40 שעות שמוקדשות בשבוע ל-GPU, על גבי שני מחשבים.
-המודל הראשון בניסוי מתאים ביותר לחיזוי עם אחוזי הצלחה גבוהים עבור הדאטא שלנו.
- באופן לא מפתיע, רשתות cnn כמו שתיארנו נותנות אחוזי הצלחה גבוהים לבעיית סיווג תמונות שלנו.
- data augmetion הוא חלק חשוב בתהליך כמו שלנו שכן כחלק מסקירת הדאטא שעשינו הוא עוזר בהתגברות על כמה וכמה בעיות ביניהן יכולת הלמידה מהתמונות עצמן.
-יש חוסר איזון מבחינת היחס של תמונות זכר לתמונות נקבה. כ-80% הם תמונות של זכר והשאר של נקבה. עובדה זו חשובה כיוון שהזכרים אופייניים במגוון רחב יותר של צבעים לעומת הנקבות. כלומר תמונות של זכרים ונקבות יכולים להיראות שונים לחלוטין ולהשפיע על תהליך הלמידה ויכולותיו של המודל.
-במקרה של כמות מוגבלת של דגימות אימון, אפשר לעשות שימוש חוזר במשקלים של המודל שאומן מראש על דאטא אחר, במודל שלנו.
-מודל שאומן מראש (transfer learning) מהווה מרכיב חשוב בפתרון הבעיה שלנו (שכן הוא נכלל בכל מודל בו התנסינו).
- שיפור הארכיטקטורה של הרשת על ידי הוספת שכבות חדשות **במודל שהוספנו לו**, הופך את המודל למדויק יותר בסיווג מיני ציפורים.
- dropout עזר לדיוק חלק מהמודלים.
-הגדלת מספר האפוקים יכול לעזור לדיוק המודל.
- סטיית תקן בין פיקסלים יכולה לשמש כדרך לזיהוי תמונות פגומות (חידוש בשבילנו).
-אפשר להשתמש בהקטנת קצב הלמידה כאשר נתוני המודל מתייצבים, מה שעוזר לייעל את המודל.



סיווג מיני ציפורים הוא משימה לא פשוטה, אבל בעלת מספר פתרונות ע"י אלגוריתמים של למידת מכונה לבניית מודלים שיכולים ביעילות לזהות את מיני הציפורים על סמך תכונות.
לדעתנו, התקדמות נוספת ניתנת להשגה על ידי אופטימיזציה נוספת של hyper-parameters, שימוש ב-data augmentation חזק יותר, רגולזיציה וטכניקות meta-learning ועוד רבות אחרות.

לפתרון הבעיה שלנו עבור חיזוי זני ציפורים בתמונות, ניסינו לגשת בכמה שיטות. לא הופתענו מהתוצאות שכן מודל cnn הוא הנפוץ ביותר לפתרון בעיות כמו שלנו. התנסינו בכמה ערכים עבור משתנים היפר פרמטרים ואופציות לשיפור דיוק המודל כפי שתיארנו. לחלקם ציפינו לראות שיפור (למשל dropout) וחלקם שיפרו בצורה מפתיעה. הבסיסיים לדעתנו ואלו שחזרו שוב שוב הם transfer learning ו- data augmentation .

גילינו שהמודל שעובד הכי טוב ומביא את תוצאת החיזוי הכי מדויקת הינו המודל הראשון. התוצאות שקיבלנו אכן תאמו לציפיות שהיו לנו כיוון שעבדנו עם רשת, cnn עם שימוש באימון מראש. בנוסף המודל הראשון הוא זה שהושקעה בו עבודה רבה של שיפור.