



**נִרְסֶד תְּרֻעָ"ד 1913 FOUNDED**  
**בֵּית הַסְּפִירָה הַרְּיָאֵלִי הַעֲבָרִי בַּחַיָּפָה**

"בן או קוף" זיהוי וקיטלוג תמנונות של קופים ובן אדם על ידי רשות נוירונים  
עמוקה

מגישה: דasha Ychilial

ת.ז: 327900817

מנחה: אסנת אנגלמן

שם חלופה: מערכות מומחה בתחום למידת מכונה

תאריך הגשה: מרץ 2023

<b>3</b>	<b>מבוא</b>
3	הרקע לפרויקט
4	תהליכי המחבר
4	אותגרים מרכזים
<b>5</b>	<b>מבנה / ארכיטקטורה</b>
5	שלב איסוף, הכנה וניתוח הנתונים - data
6	שלב בנייה וימון המודל - Build and train deep learning model
13	תיאור גרפי של המודל הסופי
15	היפר-פרמטרים
16	שלב היישום - Software development
16	כיצד היישום משתמש במודל
16	הטכנולוגיה שעלה פיה מומש המודל
16	תרשים UML של תוכנת היישום - graph3.yu
<b>19</b>	<b>מדריך למפתח</b>
19	טעינת הנתונים, בניית הרשות ואיומה
19	תוכנת היישום - graph3.yu
<b>20</b>	<b>מדריך למשתמש</b>
23	התוצר הסופי
37	תרשים UML של שלושת הרשותות:
39	רפלקציה / סיכום אישי
40	ביבליוגרפיה

## מבוא

### הרקע לפרויקט

בפרויקט זה בניתי רשת נוירונים عمוקה כדי להבדיל בין תכונות של בן, תלמיד מכיתה, ותכונות של הקוף טמרין לבן קודקוד (cotton top tamarin). סוג הקוף נבחר על ידי חידון "buzzfeed" אשר בוצע על ידי בן רופא אורן ובמרכזו השאלה "איזה קופף דומה לך בצורה הטובה ביותר". בחידון נשאלות רבות רדודות שאמורות לייצג את אישיותך ולהתאים להתנהגותו של הקוף הנבחר. מטרת הפרויקט הולכת מעבר לרובד פשוט של זיהוי בין קופף לבן אדם, הפרויקט מעלה את שאלת האבולוציה, בהנחה כי אכן נמצאים בשושינו הגנטיים עדויות של קופים, האם ישן לכך עדויות פיזיולוגיות הנראות לעין? האם מכונה בנייה אDEM מסוגלת להבחין בכך ואם כן, באיזו הצלחה? יתרה מזאת, ניתנת האפשרות לשאול את שאלת נוכנות של תוכנות חידושים באינטרנט ומהימנותן.

לבחירה נושא הפרויקט היו שלל סיבות.

ראשית, שאלת נוכנות הקלטיפיקציה היא שאלת שמעסיקה אותי בחיי היום יום, אנו משתמשים במבנה מלאכותית ובלמידת מכונה כמעט בכל דבר, אפילו בדעת התעופה בדקות אותן מכונות. השאלה האם באמצעות המכונה יכולה לזהות אותי ומה הוא טווח השגיאה של הזיהוי, עלולה לעלות בביטחוןינו. קשה נורא לחת את קפיצת האמונה הזאת, וכך לבדוק בעצמנו את הכוח שטמון בלמידת המכונה הוא צעד לעבר עתיד טכנולוגי.

בנוסף על כך, רציתי להתנסות באופן פיזי בעיבוד הנתונים, בראש שאיפות עמד הרצון להכין "מאפס" את הרשות, להתעסק בתמונות ולכתוב קוד יפה ונכון אשר ישרת אותו לאורך כל הפרויקט. לכן בחרתי אדם מהכיתה שלי, ולא מאגר תמונות מוקן מראש הייתה המעשה הנכון עבורי. רציתי להתמודד עם הקשיים הקיימים טכניים ולחוש באמצעות תחום הנדסת התוכנה, תחום שאני רואה בו את עתידי המקזע.

להלן תמונה של בן והקוף הנבחר:



## תהליכי המבחן

לאחר חיפוש عميق באינטרנט, מצאתי כי אין פרויקט מוקן בלמידה המכונה הבודק ובדיל בין בן אדם ובין קופ. למרות שאנו ממשמשים בקורסיפיקציה בחיי היום-יום - גם בטלפון שלנו, בగדרה, אנו יכולים לחפש תמונה שבה מופיעת מילת המפתח שלנו.

מכיוון שאין פרויקטים כאלה ברשות החלטתי לנסוט לשיטות מגבלות נוספות על הרשות כדי לבדוק את חוזקה של למידת מכונה. נתתי לרשות ללמידה רק מסווג אחד של קופים ורק מתנות של אדם אחד. שמחתי לגלוות שנית להבדיל בין בני אדם לקופים בעזרת דאטא מוגבל מאוד.

עקב חוסר מחקרים מוקדמים, לא נעצרתי במספר רב של מקורות מידע. מקורות המידע העיקריים שלי היו על מנת לפתור בעיות טכניות בקוד, למידה על הרשות ולראות דוגמאות של אימפלמנטציה של הלמידה.

## אתגרים מרכזיים

במהלך כתיבת הפרויקט נתקלתי בכמה קשיים מרכזיים:

- בשלב השגת התמונות והכנתן לשימוש, הייתה צריכה לאוסף מספר רב של תמונות, גם של הקופ הספציפי שבחרתי וגם של בן. אומנם את התמונות של הקופ יכולתי להוריד מהאינטרנט ישר לתוכה תקיות הנתונים אך לא יכולתי לעשות את אותו הדבר עם תמונות של התלמיד. הדרך הטריאויאלית להשגת הנתונים הייתה לצלם את התלמיד מספר רב מאוד של פעמים אך זה הציב מספר בעיות: הזיכרון המוגבל בטלפון הנייד שלי לא מאפשר לצילום רב מדי של תמונות, צילום התמונות עלול לקחת זמן רב ואם אצלם את כל התמונות בבת אחת, כשלונן על אותו רקו ויכול דומות זו לו אני מסתכו בכך שהרשת תבצע את הזיהוי לפי הרקע הדומה או לפי התאורה ופרמטרים דומים נוספים.
- בשלב בניית הרשות וכתיבת הקוד עצמו, נתקלתי בקשיי טכני גדול, ספריית keras, שבה הייתה צריכה להשתמש לא הייתה זינה לשימוש במחשבי Mac, אשר מצויים בביתי. קופעל יצא, לא היה ביכולתי לעבוד על הפרויקט מהבית במשך זמן רב עד שמצאתי תחליף לספרייה הרגילה. העברת הנתונים מחשב למחשב, כולל את התמונות, לקחה זמן רב ככל פעם שעבדתי על הפרויקט, דבר זה הוריד את המוטיבציה לעבוד על הפרויקט וכמוון את קצב העבודה.

## מבנה / ארכיטקטורה

### שלב איסוף, הכנה וניתוח הנתונים - Collect, prepare and analyze data

תחילה, נתתי לתלמיד הכתיבה שלו לענות על שאלון באזפיד שיגדר עבורו את סוג הקוף שדומה לו. לאחר מכן התחלתי בהכנות הנתונים. כדי לאמן רשות טובה צריך מספר רב של נתונים, אך ידעת שצללים אלף תמונות של בן זה עניין מסוים שיצור הרבה זמן והרבה מאוד. לכן החלטתי לאסוף כ-70 תמונות של בן, תוך כדי הקפדה על גיון ברקעם וב素质ותם ושל הקוף הנבחר ולבצע אוגמנטציה על התמונות. כאמור, שכפול של התמונות הנבחרות ולאחר מכן עברו 50% מן התמונות ביצעת סיבוב של מספרAKERAI מוגרך של מעלות ועבור 50% נוסף מן התמונות (חלק עברו גם סיבוב וגם שינוי צבע). שיניתי את בהירות הצבעים גם כן באחוז מוגרך. ביצעת את תהליך זה על שני מאגרי התמונות 100 פעמים כך שלבסוף מאגר התמונה הסופי ישיל כל 7000 תמונות של בן 7000 תמונות של קוף. התמונות צבעוניות, בפורמט jpeg ועבור חיתוך לגודל 512X512 לפני הכניסה לרשת.

התמונות הוקטנו לפני הכניסה לרשת לגודל 256x256 בغالל מספר רב מדי של פרמטרים מכדי לאמן את הרשות (הופיעה הودעת שגיאה והרשות לא רצhaft). התמונות מופוצלות לערכי `rgb` כך שנוצרת מטריצה של תלת מימדית,  $256 \times 256 \times 3$  כך שבכל איבר נמצוא ערך בין 0-255 (מתבצע במבנה המודול). התמונות חולקו ל-80% אימון ו-20% בבחינה. השתמשתי בפונקציה `image_dataset_from_directory` והשניה של הקוף ומצוותת לתקיות ערך 0 ו-1 בהתאם. הנרמול מתבצע בשלב בניה המודול שכן השכבה הראשונה שיש בראש היא שכבת Rescaling אשר מחלקת ב-255 את ערכי הפיקסלים כך שנקלע ערכים בין 0 ל-1.

```

train_ds = image_dataset_from_directory(
    "dataset2",
    validation_split=0.2,
    subset="training",
    seed=123,
    color_mode="rgb",
    image_size=image_size,
    batch_size=batch_size
)

val_ds = image_dataset_from_directory(
    "dataset2",
    validation_split=0.2,
    subset="validation",
    seed=123,
    color_mode="rgb",
    image_size=image_size,
    batch_size=batch_size
)

```

## שלב בניה וימון המודל - Build and train deep learning model

במודל הסופי, אני משתמש בשלושת הרשנות אשר עליון עבור הפרויקט: רשת נירונים عمוקה עם קובולוציה, רשת נירונים عمוקה עם fine tuning Xception ורשת transfer learning Xception. הרשנות הניבו תוצאות דומות כאשר הרשת השניה,fine tuning, מראה את התוצאות הטובות ביותר.

כעתأتאר את הארכיטקטורות של כל סוג 3 הרשנות האחרונות אשר נבדקו לקרהת הבחירה הסופית:

### 1. רשת נירונים عمוקה עם קובולוציה:

```

model = tf.keras.models.Sequential([
    tf.keras.layers.Rescaling(1. / 255, input_shape=(256, 256, 3)),
    tf.keras.layers.BatchNormalization(),

    tf.keras.layers.Conv2D(6, kernel_size=(3, 3), padding="same", activation="relu"),
    tf.keras.layers.Conv2D(8, kernel_size=(3, 3), padding="same", activation="relu"),
    tf.keras.layers.Conv2D(10, kernel_size=(3, 3), padding="same", activation="relu"),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.summary()
model.compile(optimizer='Adam',
              loss=tf.keras.losses.BinaryCrossentropy(),
              metrics=['accuracy'])

epochs = 5
tic = Time.time()

history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs,
    batch_size=batch_size,
    verbose=1,
)

toc = Time.time()

```

בנוסף על כך מצורפים גרפים ומדדים אשר מתארים את מידת המודל וסבירו של הפרמטרים שלו:

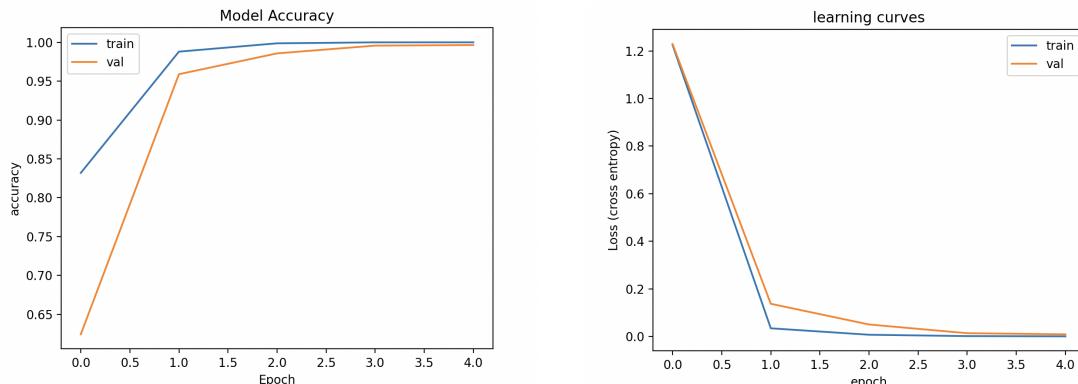
## דארה יחליאל - "קוף או ב"

Layer (type)	Output Shape	Param #
rescaling (Rescaling)	(None, 256, 256, 3)	0
batch_normalization (BatchN ormalization)	(None, 256, 256, 3)	12
conv2d (Conv2D)	(None, 256, 256, 6)	168
conv2d_1 (Conv2D)	(None, 256, 256, 8)	440
conv2d_2 (Conv2D)	(None, 256, 256, 10)	730
max_pooling2d (MaxPooling2D )	(None, 128, 128, 10)	0
flatten (Flatten)	(None, 163840)	0
dense (Dense)	(None, 64)	10485824
dense_1 (Dense)	(None, 32)	2080
dense_2 (Dense)	(None, 1)	33
<hr/>		
Total params: 10,489,287		
Trainable params: 10,489,281		
Non-trainable params: 6		
<hr/>		

```

Epoch 1/5
2023-04-17 12:04:48.360031: W tensorflow/tsl/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU frequency: 0 Hz
88/88 [=====] - 722s 8s/step - loss: 1.2244 - accuracy: 0.8320 - val_loss: 1.2287 - val_accuracy: 0.6243
Epoch 2/5
88/88 [=====] - 1850s 21s/step - loss: 0.0345 - accuracy: 0.9879 - val_loss: 0.1375 - val_accuracy: 0.9589
Epoch 3/5
88/88 [=====] - 824s 9s/step - loss: 0.0074 - accuracy: 0.9987 - val_loss: 0.0505 - val_accuracy: 0.9857
Epoch 4/5
88/88 [=====] - 715s 8s/step - loss: 0.0017 - accuracy: 0.9999 - val_loss: 0.0139 - val_accuracy: 0.9957
Epoch 5/5
88/88 [=====] - 742s 8s/step - loss: 7.8335e-04 - accuracy: 0.9999 - val_loss: 0.0092 - val_accuracy: 0.9964
Total training time (min:sec): 80:89
Train accuracy: 0.9999107122421265
val_accuracy: 0.9964285492897034
22/22 [=====] - 41s 2s/step - loss: 0.0092 - accuracy: 0.9964
Final loss: 0.01
Final accuracy: 99.64%

```



ניתן לראות כי הגענו ברשות הزاد לאחוזי ניבוי גבוהים, והרשות טואה באחוזים נמוכים. בנוסף על כך, ניתן לראות כי לקראת האפקט השלישי השימוש באחוזי ההצלחה הם זניחים ולכן ניתן לעצור כבר שם את האימון. בנוסף ניתן לראות כי אחוז ההצלחה של הטסט "final accuracy" ואחוז ההצלחה של הרשות כמעט זהים. לפיכך, ניתן לראות כי הרשות לא הגיעה למצב של התאמת יתר .overfitting

## 2. רשת נירוניים عمוקה עם Transfer Learning

הרשת בה השתמשתי ב-Transfer Learning היא Xception אשר זמינה באמצעות ספריית keras.

רשת Xception בעלת 71 שכבות והינה שיפור של רשת שנזירה כשלוש שכבות פנימית, inception, inception\_v3 וImagenet. רשת קובולוציה בעלת 48 שכבות. שתי הרשתות אומנו על אותו מאגר נתונים נתונים נתוני זיהוי ארכיטקטורת הרשת מבוססת על טכנולוגיית רשתות נירוניים עם קובולוציה שנפוצה בעיות זיהוי ויזואג. הרשת מורכבת בלבתה משכבות קובולוציה המכילות משקלים- מסננים על מנת להציג תכונות מן תמונות הקלט. בדומה ל-inception, ברשת Xception משתמשים במסלולים קובולוציוניים מקבילים על מנת לקלוט תכונות ברוחניות ובמשקלים שונים ובנוסף שתי השכבות משתמשות בשכבות "צוואר בקבוק" אשר מצמצמות את מספר ערוצי הקלט לשכבה. פעולה יוצאת, מספר הפרמטרים משכבה לשכבה יורדת והדיק של הרשת עולה.

מצ"ב תמונה של הקוד אשר יצר את המודל וVICOM הפרמטרים של המודל:

```
base_model = Xception(
    weights='imagenet',
    input_shape=(256, 256, 3),
    include_top=False)

base_model.trainable = False
model = tf.keras.models.Sequential([
    tf.keras.layers.Rescaling(1. / 255, input_shape=(256, 256, 3)),
    base_model,
    tf.keras.layers.BatchNormalization(),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.summary()
model.compile(optimizer='Adam',
              loss=tf.keras.losses.BinaryCrossentropy(),
              metrics=['accuracy'])

epochs = 1
tic = Time.time()
history = model.fit(
    train_ds,
    epochs=epochs,
    batch_size=batch_size,
    verbose=1,
)

toc = Time.time()
```

בנוסף על כך מצורפים ממדדים אשר מתארים את מידת המודל וסיכון של הפרמטרים שלו:

```

-----  

Layer (type)          Output Shape         Param #  

=====  

rescaling (Rescaling)    (None, 256, 256, 3)      0  

xception (Functional)   (None, 8, 8, 2048)     20861480  

batch_normalization_4 (Batch Normalization) (None, 8, 8, 2048) 8192  

flatten (Flatten)        (None, 131072)        0  

dense (Dense)            (None, 64)           8388672  

dense_1 (Dense)          (None, 32)           2080  

dense_2 (Dense)          (None, 1)            33  

-----  

Total params: 29,260,457  

Trainable params: 8,394,881  

Non-trainable params: 20,865,576  

-----  

2023-04-19 18:48:55.816846: W tensorflow/tsl/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU frequency: 0 Hz  

88/88 [=====] - 2838s 32s/step - loss: 0.0194 - accuracy: 0.9929 - val_loss: 0.0113 - val_accuracy: 0.9989  

Total training time (min:sec): 47:30  

Train accuracy: 0.9928571581840515  

val_accuracy: 0.9989285469055176  

22/22 [=====] - 343s 16s/step - loss: 0.0113 - accuracy: 0.9989  

Final loss: 0.01  

Final accuracy: 99.89%
-----
```

לסיכומו של דבר, ניתן לראות ממציאות הלמידה כי לא הגיעו למידת יתר מכיוון שההבדל בין האקזירוטי לטטט הוא זניח.

## 3. רשת נירוניים عمוקה עם Transfer Learning ו-Fine Tuning:

הרשת בה השתמשתי ב-Transfer Learning היא Xception אשר זמינה בعزيزת ספריית keras. בנוסף על כך ביצעת Fine Tuning כך שאימנטה את 19 השכבות האחרונות של Xception. בשימוש ב-Xception נהוג לאמן בין 10-20 שכבות אחרונות, בעוד שמספר התמונות במאגר המידע שלו קטן יחסית, היה עלי לאמן יותר שכבות על מנת לאפשר לרשת להסתגל בצורה מלאה יותר למאפיינים של מערכת הנתונים החדש ולהביא לביצועים טובים יותר במשימת הסיווג.

מצ"ב תמונה של הקוד אשר יצר את המודל:

```
base_model = Xception(
    weights='imagenet',
    input_shape=(256, 256, 3),
    include_top=False)
trainable = 19
for layer in base_model.layers[:len(base_model.layers) - trainable]:
    layer.trainable = False
for layer in base_model.layers[len(base_model.layers) - trainable:]:
    layer.trainable = True

model = tf.keras.models.Sequential([
    tf.keras.layers.Rescaling(1. / 255, input_shape=(256, 256, 3)),
    base_model,
    tf.keras.layers.BatchNormalization(),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.summary()
model.compile(
    loss=tf.keras.losses.BinaryCrossentropy(),
    optimizer=optimizers.SGD(learning_rate=1e-4, momentum=0.9),
    metrics=['accuracy']
)
epochs = 1
tic = Time.time()
history = model.fit(
    train_ds,
    epochs=epochs,
    batch_size=batch_size,
    verbose=1,
)
toc = Time.time()
```

בנוסף על כך מצורפים מגדדים אשר מתארים את מידת המודל וסיכון של הפרמטרים שלו:

```

-----  

Layer (type)          Output Shape         Param #  

=====  

rescaling (Rescaling)    (None, 256, 256, 3)      0  

xception (Functional)   (None, 8, 8, 2048)     20861480  

batch_normalization_4 (Batch Normalization) (None, 8, 8, 2048) 8192  

flatten (Flatten)        (None, 131072)        0  

dense (Dense)            (None, 64)           8388672  

dense_1 (Dense)          (None, 32)           2080  

dense_2 (Dense)          (None, 1)            33  

-----  

Total params: 29,260,457  

Trainable params: 15,721,257  

Non-trainable params: 13,539,200  

-----  

2023-04-19 18:12:58.728069: W tensorflow/tsl/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU frequency: 0 Hz  

88/88 [=====] - 3512s 40s/step - loss: 0.1458 - accuracy: 0.9418 - val_loss: 0.1349 - val_accuracy: 0.9975  

Total training time (min:sec): 58:54  

Train accuracy: 0.9417856931686401  

val_accuracy: 0.9975000023841858  

22/22 [=====] - 896s 41s/step - loss: 0.1349 - accuracy: 0.9975  

Final loss: 0.13  

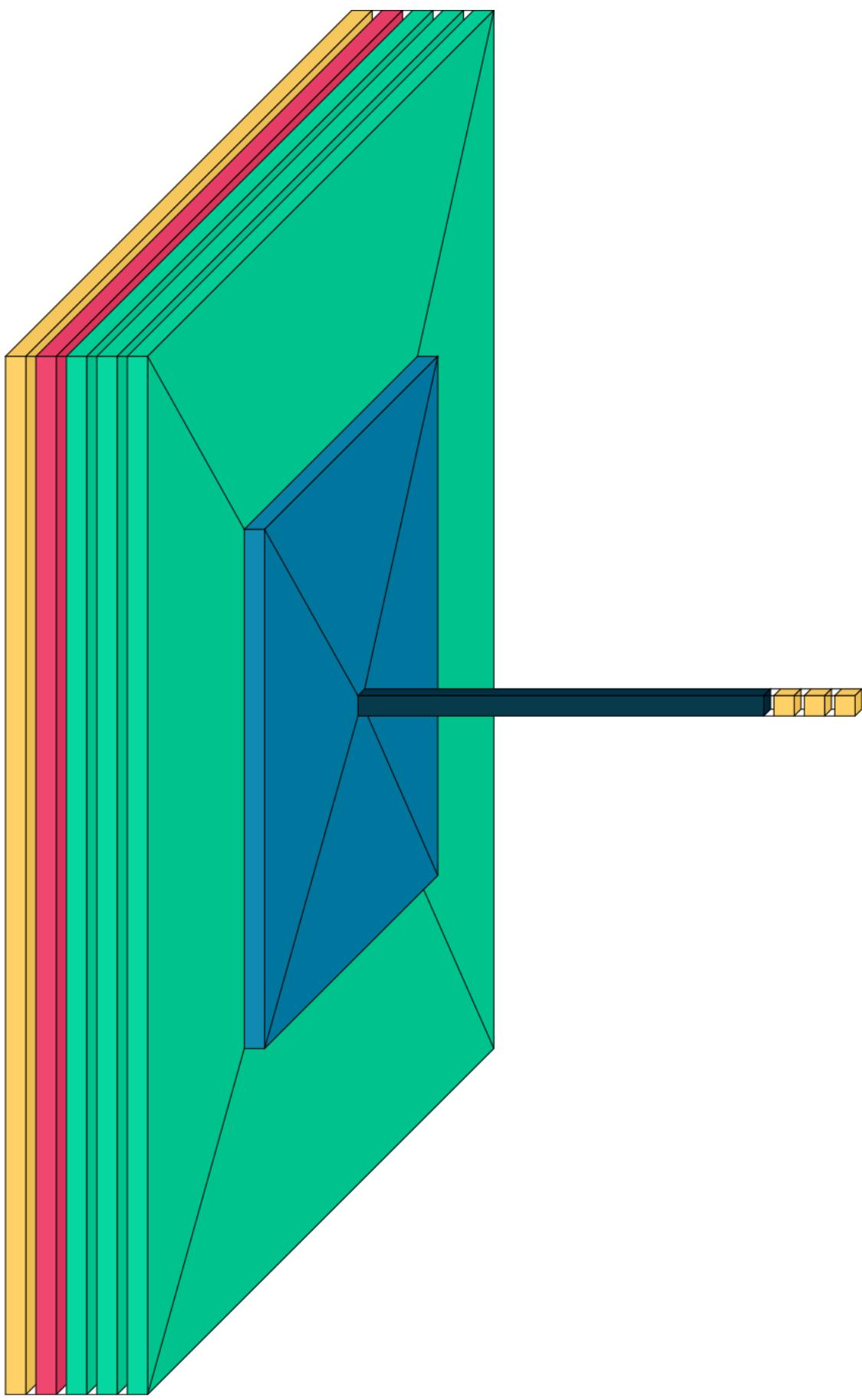
Final accuracy: 99.75%

```

לסיכון של דבר, ניתן לראות ממציאות הלמידה כי לא הגיעו למידת יתר מכיוון שההבדל בין האקזורטי לטוטט הוא זניח.

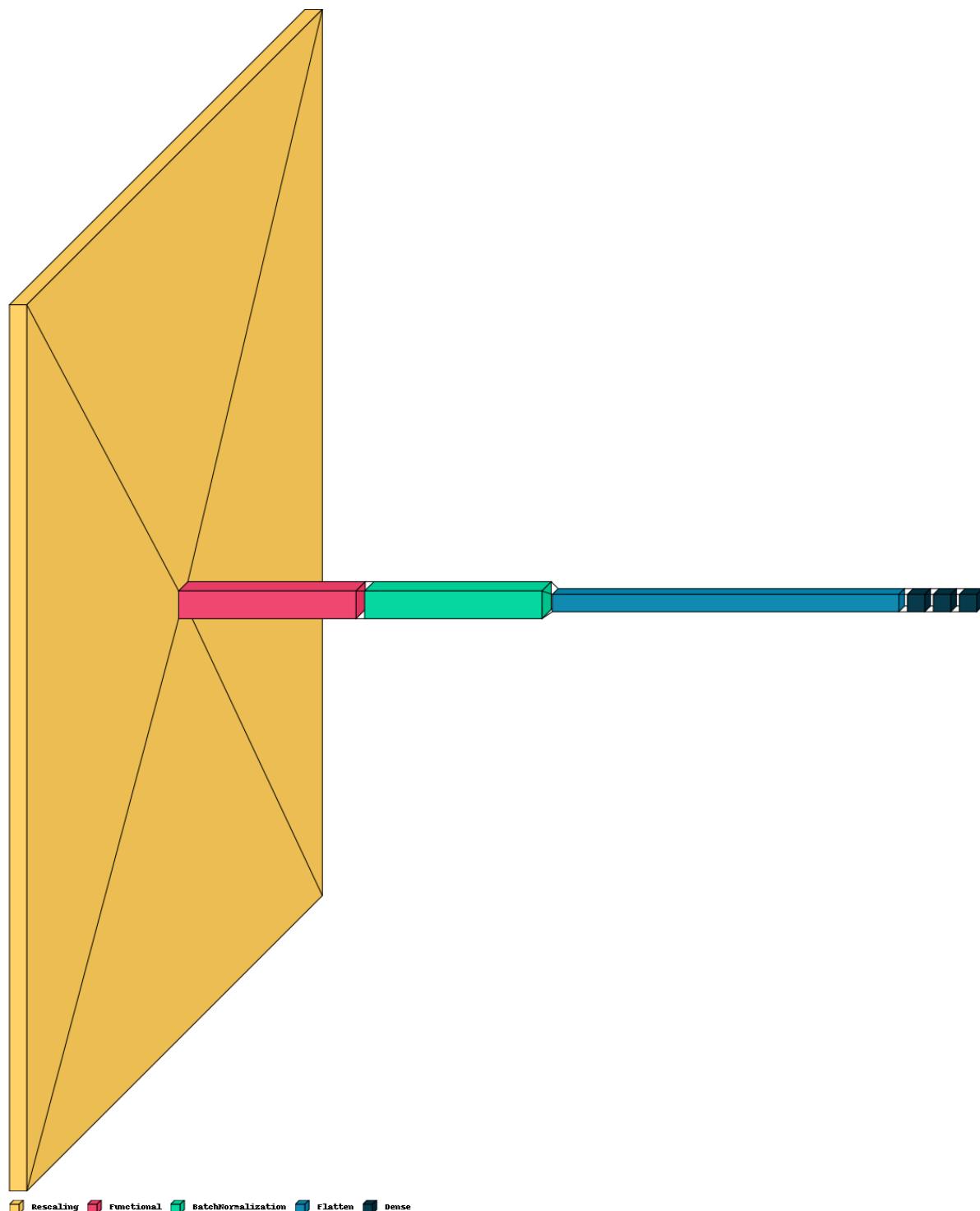
תיאור גרפי של המודל הסופי

1. רשת נירונים عمוקה עם קונבולוציה:



■ Rescaling ■ BatchNormalization ■ Conv2D ■ MaxPooling2D ■ Flatten ■ Dense

2. רשת נירוניים عمוקה עם transfer learning Xception ועבור fine tuning



## היפר-פרמטרים

לאורך הפרויקט, למרבה המזל, לא התנסתי עם מספר רב של פרמטרים, בשלושת הרשותות רבו הפרמטרים הראשוניים והמקוריים שהזנתי למודל הניבו תוצאות טובות מאוד.

בבנייה הרשת השתמשתי בשתי אקטיבציות: `relu` ו- `sigmoid`. סיגמוד הינה אקטיבציה שמשתמשים בה רבות לבעיות קולסיפיקציה ביןארית, מכיוון שהיא מחזירה ערכים בין 0 ל-1. רלו הינה אקטיבציה אשר עוזרת למנוע צמיחה אקספוננציאלית בחישוב הנדרש עבור הפעלת רשת הנוירונים. פוקנציית רלו הינה מהצורה  $\text{val} = \max(0, x)$  כלומר, עבור ערכים שליליים היא מוציאה 0 ועבור ערכים חיוביים היא משאייה כרגיל.

פונקציית השגיאה בה השתמשתי היא `binary crossentropy`. פונקציית השגיאה זו משמשת בדרך כלל במשימות סוג בינה-ארית, כאשר המטרה היא לחוץ בין שתי מחלקות אפשריות. פונקציית `binary crossentropy` מודדת את ההבדל בין התפלגות ההסתברות החזויה לבין התפלגות ההסתברות בפועל של המחלקות. במקרה אחר, הוא מחשב את המרחק בין ההסתברות החזויה של המחלקה הנכונה לבין ההסתברות האמיתית של המחלקה הנכונה.

פונקציית האופטימיזציה שבחרתי היא `"Adam"`. אלגוריתם פופולרי באימון רשותות נוירונים שהפתחה מהאלגוריתם גראינט דיסנט שיחודה בהתאמת קצב הלמידה לכל פרמטר במודל על סמך הנתונים הסטטיסטיים של השיפוע. אופטימיזציית Adam עוזרת למודל להתכנס מהר יותר ואינה יותר על ידי התאמת אוטומטית של קצב הלמידה. השתמשתי בפונקציה כדי לשפר את הדיק של המודל ולהפחית את זמן האימון. מכיוון שלמודלים של למידה عمוקה יש מספר רב של פרמטרים שצריך לכונן, האדפטיביות של הפונקציה יכולה לעזור למודל להתכנס מהר יותר ולהציג תוצאות טובות יותר.

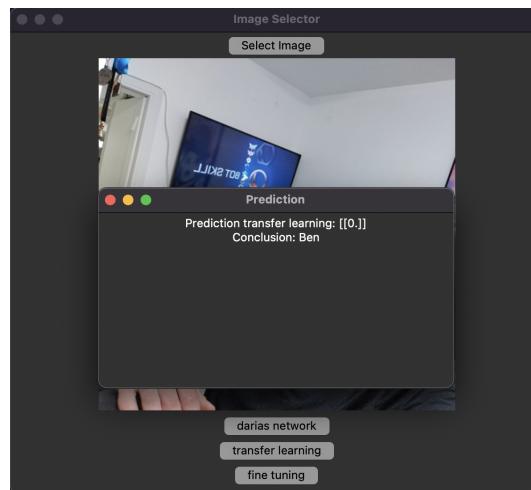
בנוסף, בשתי הרשותות הראשונות השתמשתי בקצב הלמידה הדיפולטי של אופטימיזציית `adam`, שהוא 0.001. לעומת זאת, בראשת השליישית, למידה עמוקה עם `fine tuning` ו-`transfer learning` שבה השתמשתי באופטימיזציית SGD (גרדיינט דיסנט), השתמשתי בקצב למידה איטי יותר, 0.0001 ( $1e-4$ ), זאת מכיוון שקצבו למידה מהירים עלולים לגרום למודל לשכוח מהידע שלו בהכשרה המוקדמת שלו, וגם מכיוון שקצבו למידה איטי יותר יכולם להוביל לתהילה אופטימיזציה חלקה יותר ולמנוע אוברפיטינג.

ברשת הראשונה שלי, השתמשתי בשלוש שכבות קונבולוציה. קונבולוציה היא פעולה מתמטית המשמשת בעיבוד אותות ועיבוד תמונה כדי לשלב שתי פונקציות או יותר לאחד חדש על ידי החלקה של פונקציה אחת על פני פונקציה אחרת (הקרנל) ומחשבת את החפיפה ביניהם בכל מקום. התוצאה של קונבולוציה היא פונקציה חדשה המייצגת את מידת החפיפה בין הקלט והפילטר בכל מקום.

בעיבוד תמונה, קונבולוציה משמשת לעיתים קרובות להפעלת פילטר על תמונה, כאשר הפילטר הוא מטריצה קטנה של מספרים המוחלט על כל פיקסל של התמונה. הערכים במטריצה הפילטר קבועים כיצד תמונה הקלט משתנה בכל פיקסל. לדוגמה, פילטר עשוי להיות מתוכנן כדי לשפר קצנות, ליטשטש את התמונה או לבצע סוגים אחרים של פעולות עיבוד תמונה.

## שלב היזום - Software development

היזום שבחרתי עבור הפרויקט לאפשר להעלאת תמונה שלא שומשה לאימון הרשות. כאשר נעה את התמונה, התמונה תופיע על המסך וויפיעו שלושה כפתורים אשר כל אחד מהם יציג לנו את ערך הניבוי של הרשות שאותה הוא מייצג ואת מסקנת הרשות- האם זה בן או האם זה קוף.



### כיצד היזום משתמש במודול

תחילה, שלושת המודלים (השמורים בפורמט h5), נתונים לטור שלושה משתנים שונים על ידי הפונקציה `tensorflow.keras.models.load_model()`.

לאחר שהועלתה תמונה מהמחשב של המשתמש, המשתמש מתבקש ללחוץ על אחד הכפתורים: `show_prediction`. הפונקציה מקבלת את המודל שיעשה את הניבוי ואת שם הרשות. בפונקציה `show_prediction`, נפתח חלון חדש המציג את תוצאת הפונקציה `prediction_results` שאותה כתבתי, המקבלת את כתובות התמונה שעלייה נרצה לבצע את הניבוי ואת המודל. הפונקציה מבצעת את הניבוי בעזרת פונקציית `model.predict` הנמצאת בתוך ספריית `keras`. הפונקציה תחזיר את ערך הניבוי- בין 0 ל-1, קר שאם הערך קרוב ל-0 הרשות מזאה כי מדובר בבן ומנגד, אם הערך קרוב ל-1 הרשות מזאה כי מדובר בקוף. הערך מוחזר לפונקציה `show_prediction` ושם גם מתקבלת החלטה על מסקנת הרשות, על ידי פילוג הערכים בניבוי 0.5.

**הטכנולוגיה שעל פיה מומש המודול**

כדי למש את המודול השתמשתי בספריית `tkinter`, ספרייה פופולרית בחוסט Python, לבניית ממשק משתמש גרפיים (GUI). היא מספקת דרך פשוטה וឥינטואיטיבית ליצור חלונות, לחצנים ותיבות טקסט. התchapיר של הספרייה עקבי וטריוויאלי ולכן קל מאוד להשתמש בה. בנוסף על קר, מכיוון שהיא ספרייה מובנת ב- Python, היא גם להתקינה אינה דורשת תלות נוספת.

### תרשים UML של תוכנת היזום - Graphic3.py

name	type	discription
model1	str	מחזיק את כתובות המודל הראשון- רשות נירונים عمוקה עם קונבולוציה
model2	str	מחזיק את כתובות המודל השני- רשות נירונים عمוקה עם transfer.learning
model3	str	מחזיק את כתובות המודל השלישי- transfer.nירונים عمוקה fine.tuning עם
img_ref	refrence	משתנה המחזיק הפניה לתמונה המוצגת כרגע.
Def preprocess_image	function	פונקציה מקבלת את כתובות התמונה, טענת אותה ומקטינה אותה ל-256 על 256. מmirra אותה למערך עוקע ומחזירה את המערך.
Def prediction_results	function	פונקציה מקבלת את כתובות התמונה ואת המודל ומחשבת את אחוז הnumpy.
buttons	list	רשימה השומרת על שלושת הcptורים button1, button2, button3.button 3
Def select_image	function	פונקציה select_image מאפשרת למשתמש לבחור קובץ תמונה במציאות ומציגה את התמונה שנבחרה ביז'ט, הפונקציה גם מוחקת את cptורים הישנים.
Def show_prediction	function	פונקציה מקבלת את הרשות ואת שמה, מציגה את אחוז הnumpy של הרשות הרצויה ואת מסקנתה של הרשות.
button1	tk.button	cptור המחבר לפונקציית show_prediction ומבצע את החיזוי עם הרשות הראשונה- רשות נירונים عمוקה עם קונבולוציה
button2	tk.button	cptור המחבר לפונקציית show_prediction ומבצע את החיזוי עם הרשות השנייה- רשות transfer.nירונים عمוקה עם learning
button3	tk.button	cptור המחבר לפונקציית show_prediction ומבצע את

## דארה יחליאל - "קוף או בן"

		החייזי עם הרשת השלישית- רשת מיורנים עמוקה עם fine tuning.
root	tk.Tk	חלון המרכזית שלנו- איפה שמוצגת התמונה.
button	tk.button	כפטור להעלאת התמונה הרצויה.
label	tk.Label	חלון הצגת התמונה שלנו.
root.mainloop()	method call	מפעילה את הלולאה הראשית של הfonקציה ועוקב אחרי ההקלשות של המשתמש.

## מדריך למפתח

### טעינת הנתונים, בניית הרשות וימון

- לפתח פרויקט חדש ב-**Pycharm**
- לוודא שיש גרסה מס' 3.10 של **Python**
- להוריד את הספריות: **tensorflow**, **time**, **matplotlib**
- להכניס את הקובץ **final\_model** אל תוך הפרויקט.
- ליצור **directory** עם שתי תיקיות ואת התמונות לתוךן.
- ליצור תיקיה ריקה בשם **models**.
- להריץ את **final\_model.py**.

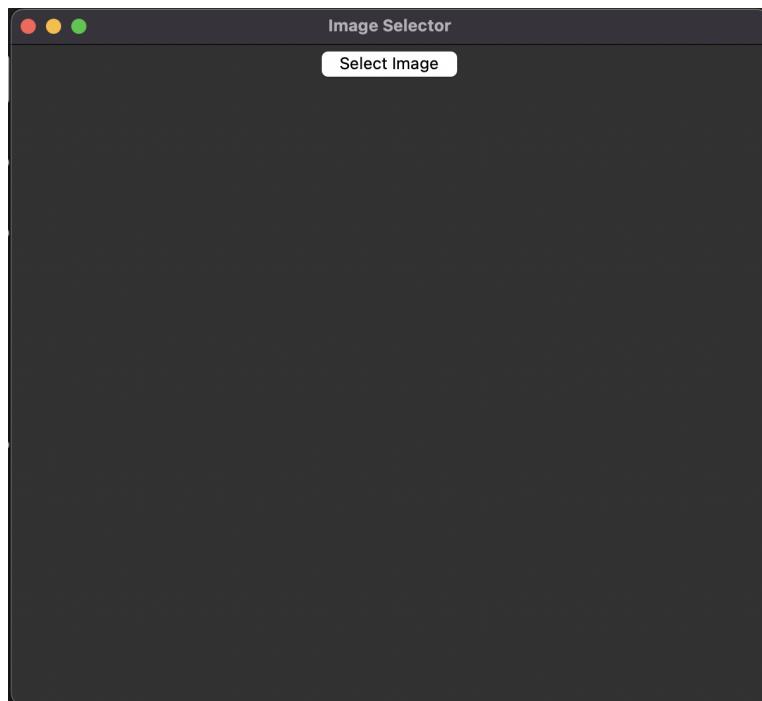
### תוכנת היישום - **graph3.py**

כדי לטעון ולהריץ את היישום של הפרויקט יש צורך ב-:

- לפתח פרויקט חדש ב-**Pycharm**
- לוודא שיש גרסה מס' 3.10 של **Python**
- להוריד את הספריות: **tensorflow**, **numpy**, **PIL**, **tkinter**
- להכניס את הקובץ **Graphics\_final** לפרויקט
- להכין תמונה רצiosa מראש בפורמט **jpeg/png**
- להריץ את הקובץ **Graphics\_final.py**.

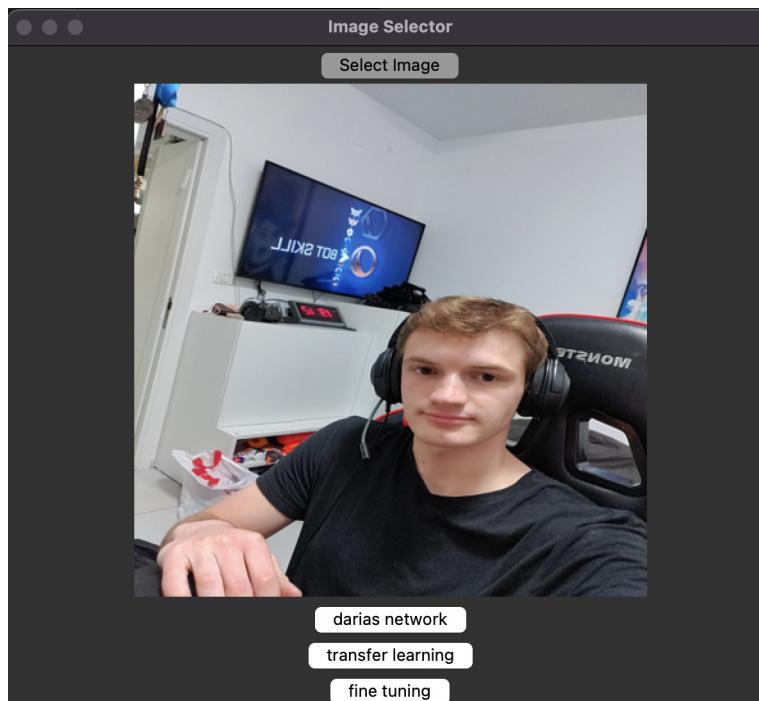
## מדריך למשתמש

בכדי להריץ את היישום של הפרויקט, יש צורך בקובץ `graph3.py` בשלושת הרשאות `works.h5`, `transfer learning.h5`, `fine tuning.h5`. כאשר נריץ את הקובץ נקבל את המסר הבא:

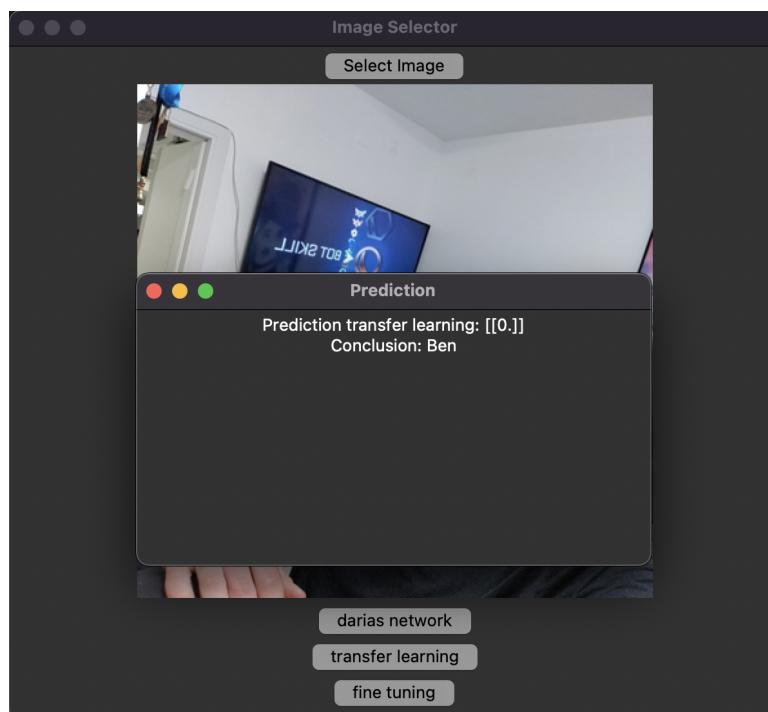


יש ללחוץ על הכפתור `Select image`, ולבחר תמונה שהורדה מראש למחשב. ברגע שנלחץ על תמונה וונעל איתה. התמונה תופיע על המסך.

## דארה יחליאל - "קוף או בן"



בנוסף לתמונה, ניתן לראות כי מופיעים על המסך שלושה כפתורים. בלחיצה על אחד מהם יפתח חלון אשר יציג את תוצאת החיזוי של הרשות ששם מופיע על ה캡טור ואת מסקנתה. לדוגמה, כאשר נלחץ על הceptor "transfer learning", יפתח מסך קסוד אשר יציג מספר בין 0 ל-1 (כאשר מספר קרוב ל-0 מייצג שהרשות חושבת שמדובר בבן ולהפך, בקוף), ובנוסף יציג את מסקנת הרשות כליה:



**דאשה יחליאל - "קופ או בן"**

ממסך זה ניתן לעשות מספר פעולות:

1. לבחור רשות אחרת ולראות את הניבוי שלו.
2. לבחור תמונה אחרת.

**התוצר הסופי****Resizing.py**

לקח תמונות מדירקטורי הנמצא בroot\_directory מקטין אותן לגודל הרצוי וubahן לoutput\_directory

---

```

import os
from PIL import Image
import math

root_directory = "BENoriginal"
output_directory = "BENresized"
# did this once for ben and the other one for monkey
# root_directory = "MONKEoriginal"
# output_directory = "MONKEresized"

size = (512, 512) # the size of pic we want
images = []

for index1, image in enumerate(os.listdir(root_directory)):
    print(image)
    print(f"appending image num {index1}")
    images.append((Image.open(root_directory + "/" + image), image))

for index, image in enumerate(images):
    print(f"processed {math.floor(index*100/len(images))}")
    image[0].resize(size).save(output_directory + "/" + image[1])

```

## Augmentation.py

לקוח את התמונות שהוקטנו, מעתיק כל תמונה 100 פעמים ועושה ל-50% שינויצבע ול-50% שינוי בסיבוב.

```

import random
import os
from PIL import ImageEnhance, Image

folder_path = 'BENresized' # using the resized pictures
destination_path = 'BENmodified'
# did this once for ben and then once for monkey
# folder_path = 'MONKEresized'
# destination_path = 'MONKEmodified'

# Loop through all files in the folder
for file in os.listdir(folder_path):
    print(file)
    for i in range(100):
        print(i)
        image = Image.open(os.path.join(folder_path, file))
        modified_image = image.copy()

        if random.random() < 0.5: # 50% chance
            # Decrease color intensity by a random percent
            modified_image = ImageEnhance.Color(modified_image).enhance(random.random())
        if random.random() < 0.5: # 50% chance
            # Rotate the copy by a random number of degrees
            modified_image = modified_image.rotate(random.random() * 360)

        # Save the modified image
        modified_image.save(os.path.join(destination_path, 'modified_' +str(i)+ file))

print('All images have been modified and saved to the destination folder.')

```

## El networke.py

```
import tensorflow as tf
import time as Time
from matplotlib import pyplot as plt

from tensorflow.keras.preprocessing import image_dataset_from_directory

image_size = (256, 256)
batch_size = 128

train_ds = image_dataset_from_directory(
    "dataset2",
    validation_split=0.2,
    subset="training",
    seed=123,
    color_mode="rgb",
    image_size=image_size,
    batch_size=batch_size
)

val_ds = image_dataset_from_directory(
    "dataset2",
    validation_split=0.2,
    subset="validation",
    seed=123,
    color_mode="rgb",
    image_size=image_size,
    batch_size=batch_size
)

test_ds = image_dataset_from_directory(
    "dataset2",
    validation_split=0.2,
    subset="validation",
    seed=123,
```

```

color_mode="rgb",
image_size=image_size,
batch_size=batch_size
)

model = tf.keras.models.Sequential([
    tf.keras.layers.Rescaling(1. / 255, input_shape=(256, 256, 3)),
    tf.keras.layers.BatchNormalization(),

    tf.keras.layers.Conv2D(6, kernel_size=(3, 3), padding="same", activation="relu"),
    tf.keras.layers.Conv2D(8, kernel_size=(3, 3), padding="same", activation="relu"),
    tf.keras.layers.Conv2D(10, kernel_size=(3, 3), padding="same", activation="relu"),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.summary()
model.compile(optimizer='Adam',
              loss=tf.keras.losses.BinaryCrossentropy(),
              metrics=['accuracy'])

epochs = 5
tic = Time.time()

history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs,
    batch_size=batch_size,
    verbose=1,
)

```

---

```

toc = Time.time()

# Calculate training time and format as min:sec
minutes = format((toc - tic) // 60, '.0f')
sec = format(100*((toc-tic) % 60)/60, '.0f')
print(f"Total training time (min:sec): {minutes}:{sec}")

# evaluating the model

print("Train accuracy: " + str(max(history.history['accuracy'])))
print("val_accuracy: " + str(max(history.history['val_accuracy'])))

# Print final loss and accuracy
loss, accuracy = model.evaluate(test_ds)
print("Final loss: {:.2f}".format(loss))
print("Final accuracy: {:.2f}%".format(accuracy * 100))

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('accuracy')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

plt.title('learning curves')
plt.xlabel('epoch')
plt.ylabel('Loss (cross entropy)')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='val')
plt.legend()
plt.show()

```

## TransfereLearning.py

```

import ssl
ssl._create_default_https_context = ssl._create_unverified_context
import tensorflow as tf
import time as Time
from keras.applications import Xception
from tensorflow.keras.preprocessing import image_dataset_from_directory

image_size = (256, 256)
batch_size = 128

train_ds = image_dataset_from_directory(
    "dataset2",
    validation_split=0.2,
    subset="training",
    seed=123,
    color_mode="rgb",
    image_size=image_size,
    batch_size=batch_size
)

test_ds = image_dataset_from_directory(
    "dataset2",
    validation_split=0.2,
    subset="validation",
    seed=123,
    color_mode="rgb",
    image_size=image_size,
    batch_size=batch_size
)

base_model = Xception(
    weights='imagenet',
    input_shape=(256, 256, 3),
    include_top=False)

```

```

base_model.trainable = False
model = tf.keras.models.Sequential([
    tf.keras.layers.Rescaling(1. / 255, input_shape=(256, 256, 3)),
    base_model,
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.summary()
model.compile(optimizer='Adam',
              loss=tf.keras.losses.BinaryCrossentropy(),
              metrics=['accuracy'])

epochs = 1
tic = Time.time()
history = model.fit(
    train_ds,
    epochs=epochs,
    batch_size=batch_size,
    verbose=1,
)

toc = Time.time()

# Calculate training time and format as min:sec
minutes = format((toc-tic)//60, '.0f')
sec = format(100*((toc-tic) % 60)/60, '.0f')
print(f"Total training time (min:sec): {minutes}:{sec}")

```

```
# evaluating the model

print("Train accuracy: " + str(max(history.history['accuracy'])))

# Print final loss and accuracy
loss, accuracy = model.evaluate(test_ds)
print("Final loss: {:.2f}".format(loss))
print("Final accuracy: {:.2f}%".format(accuracy * 100))

# saving the model
name = input("please enter the name of the model\n")
model.save("models/" + name + ".h5")
print("Saved model to disk")
```

## Fine tuning.py

```

import certifi
import ssl
from tensorflow.keras import optimizers
ssl._create_default_https_context = ssl._create_unverified_context
import tensorflow as tf
import time as Time
from keras.applications import Xception
from tensorflow.keras.preprocessing import image_dataset_from_directory

image_size = (256, 256)
batch_size = 128

train_ds = image_dataset_from_directory(
    "dataset2",
    validation_split=0.2,
    subset="training",
    seed=123,
    color_mode="rgb",
    image_size=image_size,
    batch_size=batch_size
)

test_ds = image_dataset_from_directory(
    "dataset2",
    validation_split=0.2,
    subset="validation",
    seed=123,
    color_mode="rgb",
    image_size=image_size,
    batch_size=batch_size
)

```

```

base_model = Xception(
    weights='imagenet',
    input_shape=(256, 256, 3),
    include_top=False)
trainable = 19
for layer in base_model.layers[:len(base_model.layers) - trainable]:
    layer.trainable = False
for layer in base_model.layers[(len(base_model.layers) - trainable):]:
    layer.trainable = True

model = tf.keras.models.Sequential([
    tf.keras.layers.Rescaling(1. / 255, input_shape=(256, 256, 3)),
    base_model,
    tf.keras.layers.BatchNormalization(),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.summary()
model.compile(
    loss=tf.keras.losses.BinaryCrossentropy(),
    optimizer=optimizers.SGD(learning_rate=1e-4, momentum=0.9),
    metrics=['accuracy']
)
epochs = 1
tic = Time.time()
history = model.fit(
    train_ds,
    epochs=epochs,
    batch_size=batch_size,
    verbose=1,
)
toc = Time.time()

```

```
# Calculate training time and format as min:sec
minutes = format((toc-tic)//60, '.0f')
sec = format(100*((toc-tic) % 60)/60, '.0f')
print(f"Total training time (min:sec): {minutes}:{sec}")

# -----
# evaluating the model

print("Train accuracy: " + str(max(history.history['accuracy'])))

loss, accuracy = model.evaluate(test_ds)
print("Final loss: {:.2f}".format(loss))
print("Final accuracy: {:.2f}%".format(accuracy * 100))

# saving the model
name = input("please enter the name of the model\n")
model.save("models/" + name + ".h5")
print("Saved model to disk")
```

## Graph3.py

```

import tkinter as tk
import tkinter.filedialog
import tensorflow as tf
from PIL import Image, ImageTk
from tensorflow.keras.preprocessing import image
import numpy as np

# Load the models
model1 = tf.keras.models.load_model('models/works.h5')
model2 = tf.keras.models.load_model('models/transferred.h5')
model3 = tf.keras.models.load_model('models/finetuning.h5')

img_ref = None
# a list for the buttons
buttons = []

def preprocess_image(img_path):
    # loads the image and sizes it
    img = image.load_img(img_path, target_size=(256, 256)) #resize
    img_array = image.img_to_array(img) #convert to array
    img_array /= 255. #all values between 0 and 1
    return img_array

def prediction_results(img_path, model):
    # calculates the prediction value
    image = tf.io.read_file(img_path)
    tensor = tf.io.decode_image(image, channels=3, dtype=tf.dtypes.uint8)
    tensor = tf.image.resize(tensor, [256, 256])
    input_tensor = tf.expand_dims(tensor, axis=0)
    results = model.predict(input_tensor)
    return results

```

```

def select_image():
    global img_ref, buttons
    # Destroy the previous buttons
    for b in buttons:
        b.destroy()

    img_path = tk.filedialog.askopenfilename()

    if img_path:
        # Display the image
        img_pil = Image.open(img_path)
        img_pil = img_pil.resize((400, 400))
        img_tk = ImageTk.PhotoImage(img_pil)
        label.config(image=img_tk)
        label.image = img_tk # keep a reference to the image

def show_prediction(network, network_name):
    # Create a new window to display the prediction
    prediction_window = tk.Toplevel()
    prediction_window.geometry('400x200')
    prediction_window.title('Prediction')
    # Get the prediction
    results = predict_results(img_path, network)

    conclusion = "Monkey"
    if results < 0.5:
        conclusion = "Ben"

    prediction_label = tk.Label(prediction_window,
                                text=f'Prediction {network_name}: {results}\nConclusion: {conclusion}')
    prediction_label.pack()

# Creates buttons for each network
button1 = tk.Button(root, text='darias network', command=lambda: show_prediction(model1, "daria network"))
button2 = tk.Button(root, text='transfer learning', command=lambda: show_prediction(model2, "transfer learning"))
button3 = tk.Button(root, text='fine tuning', command=lambda: show_prediction(model3, "fine tuning"))

```

```
button1.pack()
button2.pack()
button3.pack()

# Store the buttons
buttons.append(button1)
buttons.append(button2)
buttons.append(button3)

# Store the image object
img_ref = img_tk

root = tk.Tk()
root.geometry('600x525')
root.title('Image Selector')

# button for picking the picture
button = tk.Button(root, text='Select Image', command=select_image)
button.pack()

label = tk.Label(root)
label.pack()
root.mainloop()
```

## תרשים אום של שלושת הרשות:

File name	name	type	discription
EI network + transeferelearning + finetuning	image_size	tuple	The size of the input image, used for .resizing and scaling
EI network + transeferelearning + finetuning	batch_size	int	The number of samples per batch for training the model.
EI network + transeferelearning + finetuning	train_ds	tf.data.Dataset	The training dataset generated using image_dataset_from_directory method.
EI network	val_ds	tf.data.Dataset	The validation dataset generated using image_dataset_from_directory method
EI network + transeferelearning + finetuning	test_ds	tf.data.Dataset	The testing dataset generated using image_dataset_from_directory method.
EI network + transeferelearning + finetuning	model	tf.keras.Sequential	The sequential model containing a series of .layers
EI network + transeferelearning + finetuning	model.summary()	method	Displays a summary of the model's architecture.
EI network + transeferelearning + finetuning	model.compile()	method	Compiles the model with the specified optimizer, loss function and metrics.
EI network + transeferelearning + finetuning	epochs	int	The number of times the training dataset is passed through the model.
EI network + transeferelearning + finetuning	tic	float	The time at which the model training starts.

El network + transeferelearning + finetuning	history	tf.keras.callbacks. History	The object that contains the training metrics and loss.
El network + transeferelearning + finetuning	toc	float	The time at which the model training ends.
El network + transeferelearning + finetuning	min	str	The number of minutes it took to train the model.
El network + transeferelearning + finetuning	sec	str	The number of seconds it took to train the model.
El network + transeferelearning + finetuning	loss	float	The final loss value of the trained model.
El network + transeferelearning + finetuning	accuracy	float	The final accuracy value of the trained model.
transeferelearning + finetuning	base_model	object	Pre-trained Xception model object
transeferelearning + finetuning	base_model.trainable	boolean	Boolean indicating whether the base model is trainable or not
finetuning	trainable	int	The number of layers that will be trained.
finetuning	layer.trainable	boolean	The attribute of a layer that determines whether it will be trained or not.

## רפלקסיה / סיכום אישי

העבודה על הפרויקט הייתה מתוגרת אך מרגשת ביותר, לאחר שנה של לימודים על נושא למידת המcona סוף סוף הונקה לי האפשרות ל"הריגש בידים" את התחום. תחום שהוא רלוונטי ועכשווי.

麥iouן שרב הלימודים היו תיאורתיים הרגשתי שהחומר צף ולא שוקע, لكن כאשר התחלתי את הפרויקט הרגשתי הלם ולא ידעת מייפה להתחיל. הידע המוקדם שלי בפייתון היה מצומצם ביותר והייתי באמת צריכה להתחילה הכל "מאפס" - ולא רק את הרשות והנתונים. נתקلت במספר רב של בעיות טכניות ולוגיסטיות אך קיבלת תמייה גם ממורותי וגם מחברי לכיתה. הפרויקט לימד אותי למידה אמצעית והמן משמעת עצמית: עמידה בלוחות זמן, אחריות, ארגון ועוד.

למדתי שבתחום מדעי המחשב חשוב נראה להיות רלוונטיים וכל הזמן למדוד ולהתחדש ביחד עם התחום ולהתעסק בגיל כל כרך צעיר בפרויקט שלא יביש אף מהנדס תוכנה בלימודי לתואר ראשון-היא באמת זכות.

הכנת מאגר הנתונים הייתה החלק החביב עלי ביותר לאור הפרויקט, לראות את התוצאות, את השינויים שנעשים בתמונה ולראות את מספר התמונות גדול ונעשה למהימן יותר, גרם לモטיבציה שלי להתרומם ובאמת להצטרף לפרויקט. ללא ספק, בחירת מאגר נתונים קיים אינה הבחרה הנכונה בשבייל.

לאחרונה נחשפתי לאלגוריתם סיוו אשר אפשר לקבל "ריבוע" שמציג הין הרשות Ziיתה את אחת מהקטגוריות, אלגוריתם זה ייעיל במצבים בהם יש גם קופ וגם בן אדם באויה התמונה. כרגע, כאשר נכנס לרשף תמונה צאת-חלק מהרשפות יראו כי מדובר בן והשאר בקופ. הייתה שמחה להכין את זה לפרויקט שלי אם הייתה מתחילה אותו מחדש.

בנוסף, אם הייתה מתחילה את הפרויקט מחדש הייתה מאמנת את הרשות על יותר מבן אדם אחד ועל סוגים שונים של קופים. למרות שלפי ניסוי וטעיה שערכת עס סוגים של קופים ואנשים שונים שאינם דומים לבן בצבע עורם ובתווי פנים שונים, כל הרשות מזוהה כי מדובר בן (אדם) וכן סוגים שונים של קופים. ولكن, אחת ממסקנותי בפרויקט זה, שאפילו באימון רשות על אדם אחד וסוג קופ אחד- הרשות יודעת לזהות ולסואג.

תוצאה מעניינת נוספת שהתקבלה במהלך בדיקת הרשות היא תגבען של הרשות על בובת קופ. שלושת הרשות- בפה אחד, טענות כי בתמונה מוצג בן (אדם). ניתן לשער לפיך כי הרשות יכולה לזהות "חיות" ולא רק קופים. השערה זאת אומתה כאשר העלתה לבדיקה תמונה של כלב (צחהה, לבן), שאינו בצבע המצוי של קופים (חום, שחור, כתום) ושלושת הרשות קבועו כי מדובר בקופ. לכן, ניתן להסיק כיまいון של קופ ובן אדם אחד- ניתן לסואג בין חיות ובני אדם.

לסיכום, הפרויקט הזה היה הගולה שבכורת של 3 שנים למדעי המחשב והנדסת תוכנה ותרם רבות לרgesch האהדה שיש לי כלפי המקצוע הזה.

## ביבליוגרפיה

1. “*What kind of monkey are you?*” created by *fire\_blazing691* on 8.11.2021.  
<https://www.playbuzz.com/emeryjones10/what-kind-of-monkey-are-you>
2. “*Tf.keras.utils.image\_dataset\_from\_directory*”, Tensorflow, 23.3.23.  
[https://www.tensorflow.org/api\\_docs/python/tf/keras/utils/image\\_dataset\\_from\\_directory](https://www.tensorflow.org/api_docs/python/tf/keras/utils/image_dataset_from_directory)
3. “*Labels in Tkinter (GUI Programming)*”, Python, 2022.  
<https://pythonbasics.org/tkinter-label/>
4. “*TensorFlow 2 Tutorial: Get Started in Deep Learning with tf.keras*”, Machine Learning Mastery. 19.12.2019.  
<https://machinelearningmastery.com/tensorflow-tutorial-deep-learning-with-tf-keras/>
5. “*Deep Learning Specialization*”, Andrew Ng on Coursera (n.d.).  
<https://www.coursera.org/specializations/deep-learning>