

ממן 14 – פרויקט גמר – תחרות זיהוי לווייתנים מסוג humpback ב-Kaggle

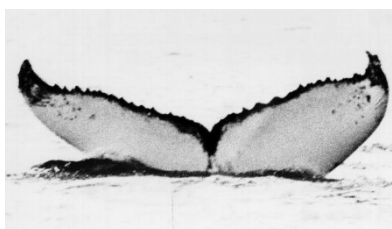
1. מבוא

התחרות הנוכחית מאתגרת את המשתתפים לזהות "זהות" של לווייתן לפי תמונות הזנב שלו. נתון מאגר של כ- 25,000 תמונות שבהן מתויגים 5005 לווייתנים. להלן סקירה של הנתונים:

#		Train	Train %	Test	Comments
1	Total # of images	25,361	100	7,960	
2	Total # of individual whales	5005	-	7,960	
3	Whales appearing only once	2073	8.2	-	
4	Whales appearing only twice	1285	5.1	-	
5	Non-identified whale images	9,664	38	-	Labeled as "new whale"
6	Identified whale images	15,697	62	-	Unequally divided between 5004 unique whales
7	Second common whale images	73	0.3	-	Next common whales image count: 65, 63, 61, 57 etc.
8	Third common whale images	65	0.2	-	

הטבלה ממחישה שסט הנתונים אינו אחיד. התווית הנפוצה ביותר היא "new_whale" ומשמעותה שהלווייתן בתמונה אינו מזוהה. יתרה מכך, יותר מ-2K לווייתנים מופיעים פעם אחת בלבד בסט האימון, יותר מ-1K פעמיים בלבד וכך הלאה. הלווייתן שתמונתו שניה בשכיחותה בסט האימון (אחרי new_whale) מופיע 73 פעמים בלבד.

בהקשר של זיהוי תמונה, התמונות אינן אחידות בגודל או באיכות. ישנן תמונות שחור לבן וישנן תמונות RGB במגוון גדלים. יתרה מכך חלק מהתמונות אינן ממורכזות, מסובבות, זווית הזנב משונה או אפילו עם כיתוב כפי שניתן לראות למטה:



Y2399



#1998

Dalhousie University

קל לראות שעיבוד מקדים הכרחי במקרה זה. נזכור שמטרתנו אינה למצוא את התמונה/תמונות מסט האימון שהכי דומה/דומות (מבחינת פיקסלים) לתמונה נתונה בסט הבדיקה, אלא למצוא את **זהות הלווייתן שבתמונה**. רשתות CNN שילמדו על התמונות הנתונות (כמו שהן) עלולות להסיק מסקנות על סמך פרטים לא רלוונטיים כגון מבנה הגלים, צבע המים או אף ציפור או כיתוב ברקע הזנב של הלווייתן. לפיכך דרוש עיבוד מקדים אינטנסיבי לביצוע זיהוי הלווייתן לפי זנבו.

2. שיטות עבודה

פעולות בסיסיות כמו חיתוך תמונות מסט האימון וסט הבדיקה בוצעו על מחשב ביתי עם מעבד intel i7 CPU. תמונות חתוכות שעברו עיבוד מקומי הועלו לענן של Kaggle בפורמט zip ונעשה בהן שימוש כדטאסט פרטי (ה) זמינות להורדה מכאן: (https://drive.google.com/open?id=1V0_W6hr8k_OHD0JwQDbWDbz2PYBH02hi).

בשל מגבלות מחשוב ביתי, עבדתי בעיקר בסביבה המוצעת ע"י Kaggle (Kernel). ניצלתי את ה-GPU המוצע ע"י Kaggle: Tesla K80 עם RAM של 14GB. הספרייה העיקרית שנעשה בה שימוש למימוש CNN היא keras.

בשל מגבלת RAM והצורך באימון על סט של עשרות אלפי תמונות, גודלן נורמל ל-100*100 עד 128*128 פיקסלים בפורמט RGB. עשיתי שימוש גם באופציית "ההזרמה" (flow_from_dataframe) ש-keras מאפשר לצורך טעינת תמונות גדולות יותר ואף ביצוע אוגמנטציות שונות על כל אצווה (ע"י ImageDataGenerator).

כפי שיוסבר בהמשך, התוצאות המיטביות אליהן הצלחתי להגיע (**Kaggle LB Score 0.545**) התקבלו תוך אימון על תמונות שגודלן הותאם לריבוע באורך/רוחב של 100-128 פיקסלים.

במאמר מוסגר אציין שבשל מחסור בזמן למחקר מעמיק, למימוש ואף מגבלות המשאבים לביצוע, **בחרתי לא לבנות רשת custom-made אלא להשתמש ב-transfer learning עם ארכיטקטורות ידועות וחזקות המאומנות מראש על ImageNet** לצורך הוצאת פיצ'רים וקלסיפיקציה של זנבות הלווייתנים. הארכיטקטורות שבחרתי:

- a. Resnet50
- b. InceptionResnetV2
- c. Xception
- d. InceptionV3
- e. VGG16

ה-transfer-learning בוצע במתודולוגיה דומה לממן 13 שאלה 3:

1. טעינת המודל הרצוי מתוך keras.applications תוך הקפאת השכבה האחרונה (include_top=False)
 2. הוספת שכבת GlobalAveragePooling2D
 3. הוספת שכבת Dropout שמאפסת באופן רנדומאלי 40% מהמשקלים שנלמדו על מנת להימנע מ-overfitting.
 4. הוספת שכבת Dense על מנת לחלץ את 5004 סוגי הלווייתנים בשלב הניבוי (ניבוי עבור תמונות סט הבדיקה).
- מסיבות שיובהרו בפרק העיבוד המקדים, בחרתי להתעלם מתמונות הלווייתנים הלא מזוהים לצרכי אימון הרשתות.** יצרתי יותר מ-10 קרנלים ב-Kaggle (עבור כל אחת מהארכיטקטורות לעיל ועוד) עם הרבה גרסאות (ניסיונות לשיפור) לכל קרנל. בסיכום זה מובאים **רק** הניסיונות המוצלחים ביותר. מודלים ומשקלים בפורמט h5, מחברות ipynb וקבצי פלט בפורמט pdf זמינים בתיקיית "kaggle_kernels":

https://drive.google.com/open?id=1JO8NXw1BTyIEzk2GImSWxl3R_9HlhFn

שלבי העבודה במרבית הקרנלים זהים ומתוארים בקובץ `kaggle_kernels_ReadMe.txt`.

3. עיבוד מקדים

3.1 חיתוך הזנב מתוך תמונות האימון והבדיקה

שלב זה הוא קריטי, אך בשל אופי התמונות בסט האימון הוא מאתגר מאוד לביצוע. בעיקר כי קשה לאתר את גבולות הזנב בסט כה מגוון ולא אחיד. הדרך היעילה ביותר להשיג זאת היא אימון רשת CNN למטרה זו בלבד, תוך יצירה ידנית (!) של Bounding Boxes לתמונות שהיוו את סט האימון לרשת זו. להלן פירוט העבודה ע"י מנצח התחרות הקודמת לזיהוי לווייתנים (לפני כ-7 חודשים): <https://www.kaggle.com/martinpiotte/bounding-box-model>

על מנת לחסוך בזמן ובמשאבים, תוצאות הרצת הרשת המתוארת עבור נתוני התחרות הנוכחית, נלקחו בפורמט CSV מכאן: <https://www.kaggle.com/suicaokhoailang/generating-whale-bounding-boxes>

הקובץ 'bounding_boxes.csv' מוגש עם הפרויקט בתיקיית resources באישור מראש של המרצה.

כאמור בפסקה הקודמת, כתבתי סקריפט לחיתוך התמונות בסט האימון וסט הבדיקה והעליתי את התוצאות ל-Kaggle בפורמט zip על מנת לזרז את אימון הרשת. הסקריפט 'Whales_cut_bbox.py' זמין בתיקיית code עם קובץ ReadMe המפרט את מבנה התיקיות הדרוש להרצה.

כל התמונות הדרושות להרצה במבנה התיקיות הרצוי זמינות להורדה מכאן: https://drive.google.com/open?id=1V0_W6hr8k_OHD0JwQDbWDbz2PYBH02hi

3.2 הורדת תמונות של לווייתנים לא מזוהים מסט האימון

סט האימון כאמור אינו מאוזן סטטיסטית. רק 68% ממנו מכיל תמונות מסוגות של לווייתנים מזוהים וכ-8% ממנו הוא לווייתנים שתמונתם מופיעה פעם אחת בלבד. תמונות עם התיוג new_whale משמעותן "לווייתן לא מזוהה" והן מהוות 38% מסט האימון! כמות אדירה זו עלולה לגרום ל:

- **הכנסת "רעש" מיותר לסט האימון** – הרי הסיווג new_whale אינו תורם למציאת זהותו של בעל הזנב בתמונה.
- **הטיית פונקציית השגיאה** – סט הלווייתנים הלא מזוהים עלול להכיל תמונות של פרטים שתיוג נכון בתמונות אחרות. לפיכך, פונקציות שגיאה שנוטות "להעניש" על תוצאה לא נכונה עלולות להניב תוצאות גרועות. זאת מכיוון שזיהוי מדויק של פרט שתיוג בשמו האמיתי בתמונה אחרת אך כ-new_whale בתמונה הנוכחית, עלול לקבל פידבק שגוי (במקרה והתמונה תיוגה כ-new_whale) ולגרום לעליית השגיאה.
- **אוברפיטנינג** – הרשת עלולה להפוך מותאמת מדי לסט האימון ולהניב תוצאות גרועות בסט הבדיקה. זאת בעיקר בשל חוסר האיזון בין 9,664 התמונות של לווייתנים לא מזוהים ו-73 התמונות של הלווייתן הבא בשכיחותו וכל השאר.

על מנת להימנע מהתופעות שתוארו למעלה, אבחר לסנן החוצה את ה-new_whale מסט האימון ולאמן רק את שאר 15,697 התמונות של לווייתנים מזוהים. חשוב לציין שהסיכוי לנוכחותו של new_whale בסט הבדיקה היא גבוהה מאוד. מכיוון ש:

- ייתכן שהרשת לא תצליח לזהות בהסתברות גבוהה מספיק לווייתנים מסוימים מסט הבדיקה. בעיקר בשל:
 - חוסר האיזון בין לווייתנים שתמונתם מופיעה פעמים בודדות בלבד לאלו שתמונתם מופיעה עשרות פעמים
 - שוני בזווית הזנב בעת הצילום
 - השפעת הרקע ורעשים נוספים שאינם רלוונטיים לזהות הלווייתן על הפיצ'רים הנלמדים ע"י הרשת
 - שוני באיכות וגודל התמונה (למשל תמונות שחור-לבן מול תמונות צבעוניות בגדלים שונים)
- מכיוון שנדגמו רק 5,004 לווייתנים שונים במדגם הנתון ולא ידוע כמה פרטים סה"כ מונה אוכלוסיית הלווייתנים מסוג humpback בעולם (או מה האחוז של לווייתני המדגם הזה מתוך כלל האוכלוסייה), ייתכן וסט הבדיקה יכול תמונת לווייתנים שאינן במדגם זה.

בהתבסס על התוצאות בקרנל הבא: <https://www.kaggle.com/pestipeti/only-new-whale-benchmark>

כ-27.6% מסט הבדיקה מכיל תמונות של לווייתנים לא מזוהים (new_whale). לפיכך כדאי מאוד להחזיר את תוויית ה-new_whale לסט התוצאות (לרשימת ההסתברויות המתקבלת בעת "חיזוי" זהויות הלווייתנים ע"י הרשת). ההחזרה מתבצעת עם סף מסוים שאת ערכו ניתן לבחור בצורה שרירותית או אמפירית (ניסוי וטעייה). בחרתי את הסף באופן שרירותי להיות שווה לממוצע של ההסתברויות הגבוהות ביותר שהתקבלו לכל הלווייתנים בסט הבדיקה. להלן הקוד המבצע את ההוספה. הסף המדובר הוא avg_of_max_predictions :

```
def add_new_whale_to_predictions(preds):
    sorted_preds = np.sort(preds)
    avg_of_max_predictions = np.average(sorted_preds[:, -1:])
    print("Average of max probabilities column:" + str(avg_of_max_predictions))
    best_threshold = avg_of_max_predictions
    shape_to_add = (np.shape(preds)[0], 1)
    # Add a column with the best threshold probability to the predictions
```

```
column_to_add = np.zeros(shape_to_add) + best_threshold
predictions_w_new_whale = np.concatenate([column_to_add, preds], axis=1)
return predictions_w_new_whale
```

3.3 דגימה עודפת (oversampling) של לווייתנים המופיעים בפחות מ-15 תמונות

עשיתי ניסיון לאזן את כמות המופעים של לווייתנים שונים בסט האימון ע"י דגימה עודפת של לווייתנים שמופיעים בפחות מ-15 תמונות. שיניתי את ה-train.csv באופן כזה, שכל שורה של לווייתן "נדיר" (מופיע בפחות מ-15 תמונות) שוכפלה עד שהמופע שלה הגיע ל-15. כל השורות עורבבו (shuffling) על מנת למנוע הטיה באימון הרשת ונשמרו לקובץ csv חדש. סקריפט המימוש whales_oversampling.py מוגש בתיקיית code עם הוראות הרצה המפורטות בקובץ whales_oversampling_ReadMe.txt.

4. תוצאות

4.1 התוצאות המוצלחות ביותר

התוצאה הטובה ביותר שהצלחתי לקבל היא **LB 0.545** (כלומר 54.5% מהלווייתנים בסט הבדיקה זוהו נכון).

NinaV 0.545

התוצאה LB 0.545 הושגה ע"י מיצוע ניבויים (predictions) של 3 מודלים שונים כפי שיתואר בטבלה מס' 1 להלן:

#	Architecture	Resnet50	InceptionResnetV2	Xception	Xception	InceptionV3	VGG16	Ensemble Score
1	Optimizer	Adam	Adam	Adam	Adagrad	Adam	Adam	
1	Loss	Categorical Crossentropy	Same	Same	Same	Same	Same	
2	Trainable Layers	All except top	Same	Same	Same	Same	Same	
3	Learning rate	default+	default+	default+	default++	default+	default+	
4	Total Epochs	60	30	30	20	100	100	
5	Image Size	100 ²	128 ²	128 ²	128 ²	128 ²	128 ²	
6	Image Augmentation	yes +++ (last 30 epochs)	no	no	no	no	no	
7	Folder Name	resnet_50_kaggle_lb_0_426*	inceptionresnetv2_kaggle_lb_0_515*	xception_kaggle_lb_0_449*	**	**	**	
8	Kaggle LB Score	0.426	0.515	0.449	0.423	0.316	0.284	
9	Ensemble #1	✓	✓					0.523 ***
10	Ensemble #2	✓	✓	✓				0.545 ***
11	Ensemble #3	✓	✓	✓	✓			0.539 ***
11	Ensemble #4		✓	✓				0.529 ***

* תוצאות הרצת הקרנלים שמורות בתיקיית 'kaggle_kernels' בתת תיקיית לפי האמור בטבלה. כל תת תיקייה מכילה:

- מודל בפורמט 'hd5'
- משקלים בפורמט 'hd5'
- קובץ תוצאות בפורמט csv
- קוד בפורמט ipynb
- פלט הקרנל בפורמט pdf

** בחרתי לא להגיש את תוצאות שלושת הקרנלים בעלי הציונים הנמוכים בטבלה.

*** תוצאות ה-ensembling והסקריפט 'whales_ensembling.py' מוגשים בתיקיית 'code' עם קובץ ReadMe המפרט את הדרישות להרצה והסבר על המימוש.

+ קצב הלמידה המהווה ברירת מחדל לאופטימיזר Adam הוא 0.001.

++ קצב הלמידה המהווה ברירת מחדל לאופטימיזר Adagrad הוא 0.01.

+++ סוגי האוגמנטציות שבוצעו על תמונות ששימשו לאימון Resnet50 ב-30 epochs האחרונים:

```
train_datagen = ImageDataGenerator(
    fill_mode='nearest',
    validation_split = 0.1,
    rotation_range=ROTATE,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=False)
```

- רוטציה בטווח של עד 20 מעלות
- מתיחה/קיצור רוחב התמונה בטווח של עד 20%
- מתיחה/קיצור גובה התמונה בטווח של עד 20%
- מתיחה אלכסונית (shear) בטווח של עד 20%
- זום בטווח של עד 0.2
- ללא סיבוב אופקי (horizontal flip)

האוגמנטציות שתוארו נבחרו במטרה לשנות רק במעט את תמונות הזנב כדי לאפשר לרשת resnet50 שאומנה על תמונות הזנב המקוריות (ללא אוגמנטציה) ללמוד מאפיינים נוספים ולשפר דיוק ללא overfitting. אכן התקבל דיוק טוב יותר מהאימון הראשוני (LB 0.426 לעומת LB 0.406). להלן דוגמאות לתוצאות האוגמנטציה (נלקח מאחד הקרנלים שיצרתי):



4.2 בחירת המטא פרמטרים לאימון הרשתות

האופטימיזצור Adam נבחר עבור רוב הארכיטקטורות. Adam מהווה הרחבה ל-SGD ומציע שילוב של יתרונות ה-Adagrad שמתמודד היטב עם מידע דליל וגם מיתרונות ה-RMSProp הכוללים התאמה ולימוד של ה-LR. בנוסף, כדי להתמודד עם הייצוג הדליל (בתמונה או שתיים) של יותר מ-13% מאוכלוסיית האימון, בחרתי לנסות גם את Adagrad בשילוב עם ארכיטקטורת Xception. אלגוריתם זה הניב תוצאה המדורגת רביעית מבין 6 התוצאות המוצגות בטבלה מס' 1 (LB 0.423).

פונק' השגיאה שבחרתי היא Categorical Crossentropy המתאימה לסיווג למחלקות רבות כאשר התוויות קודדו בשיטת one-hot-encoding (ייצוג כל קטגוריה ע"י מטריצת 0-ים כאשר מופיע 1 רק בעמודה המתאימה לתווית הסיווג).

4.3 ensemble של תוצאות מוצלחות

בתחרות זו נדרשים להציג 5 אפשרויות לזהותו של הלווייתן המופיע בכל תמונה. לפיכך, מיצוע ה-probabilities שקיבלו Kaggle LB Score גבוה, הוא בעל פוטנציאל להגדיל את הדיוק של חמשת ה-probabilities המקסימליים. יתרה מכך, מיצוע כזה דורש מעט כוח מחשוב והוא מהיר לביצוע. החלטתי לחשב מס' קומבינציות של התוצאות, כפי שמסבר בטבלה מס' 1. בסוף כל אימון מודל וניבוי סט הבדיקה, שמרתי את מערך ההסתברויות בפורמט npy (ממומש ע"י ספריית numpy) על מנת לאפשר מיצוע עם תוצאות מאוחרות יותר. המיצוע אכן שיפר את יכולת הניבוי של המודלים.

התוצאה המיטבית, כאמור התקבלה ממיצוע ההסתברויות של שלושת התוצאות הבאות:

- 1 Resnet50 עם דיוק של LB 0.426
- 2 InceptionResnetV2 עם דיוק של LB 0.515
- 3 Xception עם דיוק של LB 0.449

מיצוע שלושת ההסתברויות הללו הניב דיוק של LB 0.545. כאמור בתחתית טבלה מס' 1, תוצאות ה-ensembling והסקריפט 'whales_ensembling.py' מוגשים בתיקיית 'code' עם קובץ ReadMe המפרט את הדרישות להרצה והסבר על המימוש.

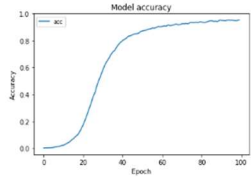
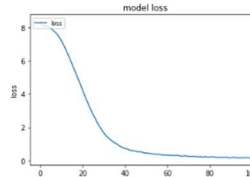
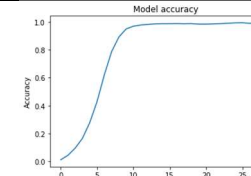
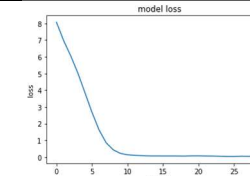
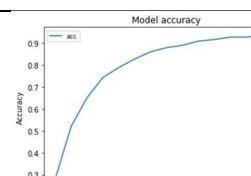
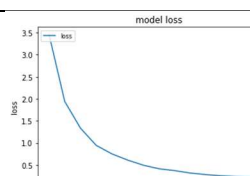
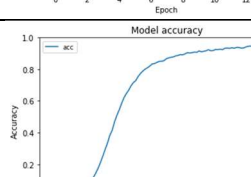
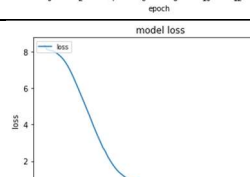
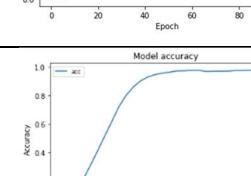
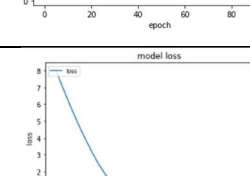
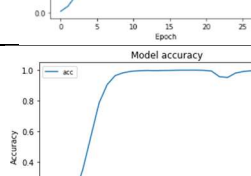
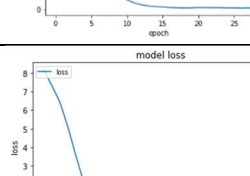
4.4 ניתוח התוצאות

בטבלה מס' 2 מוצגות 5 רשתות שנבדקו בסדר ביצועים עולה על הדטאסט ImageNet (הנתונים מאתר Keras):

#	Model	Size	Top-1 Accuracy *	Top-5 Accuracy *	Parameters	Depth
1	VGG16	528 MB	0.713	0.901	138,357,544	23
2	ResNet50	99 MB	0.749	0.921	25,636,712	168
3	InceptionV3	92 MB	0.779	0.937	23,851,784	159
4	Xception	88 MB	0.790	0.945	22,910,480	126
5	InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572

* דיוק מסוג Top-1 ואף מסוג Top-5 מתייחס לתוצאות על סט הווילדציה של ImageNet

בטבלה מס' 3 מוצגים גרפים של דיוק ושגיאת האימון עבור חמשת הארכיטקטורות שצינו לעיל על תמונות הלווייתנים. הרשתות מופיעות בסדר זהה לטבלה מס' 2 לצורך השוואה:

#	Architecture	Epochs Until Convergence	Accuracy	Error	Kaggle LB Score
1	VGG16	~80			0.284
2	Resnet50 Initial Training	~12			0.406
3	Resnet50 Secondary Training (with image augmentation)	~15			0.426
4	InceptionV3	~75			0.316
5	Xception	~20			0.449
6	InceptionResNetV2	~12			0.515

ניתן לראות שהדירוג לפי רמת הדיוק של המודלים עבור זנבות הלווייתנים דומה מאוד לדירוג עבור ImageNet:

- **InceptionResnetV2** הניבה את הדיוק (היחסי) הגבוה ביותר גם עבור ImageNet וגם עבור הזנבות.
- **Xception** דורגה שניה מבין החמישייה שנבדקה בשני הדטאסטים.
- **InceptionV3** יוצאת דופן עם ביצועים נמוכים מהצפוי - היא הניבה דיוק **שלישי** בגובהו עבור ImageNet אך רק **רביעי** בגובהו עבור זנבות הלווייתנים.
- **Resnet50** דורגה **שלישית** עבור הזנבות אך **רביעית** עבור ImageNet. ייתכן והשיפור בביצועים בסט הלווייתנים קרה הודות לאוגמנטציה שבוצעה על הזנבות בשלב השני של אימון הרשת.
- **VGG16** דורגה אחרונה בחמישייה בשני הדטאסטים.

מהתוצאות ניתן להסיק שרעיון ה-Transfer Learning עובד במקרה זה כי רשתות שמצליחות לחלץ פיצ'רים כללים מתמונות ה-ImageNet (כגון פינות, נקודות עניין, קווי מתאר ועוד ועוד...) מצליחות לעשות זאת גם עבור זנבות הלווייתנים.

אומנם, אין מנוס מלהתמודד עם השאלה המטרידה – מדוע הדיוק בכל זאת כה נמוך עבור זנבות הלווייתנים?

כפי שניתן לראות בטבלה מס' 3 - כל הרשתות התכנסו על סט האימון והניבו דיוק קרוב ל-1 בסט זה. לעומת זאת, הדיוק על סט הבדיקה עובר את ה-0.5 במקרה הטוב ואף נמוך מזה במקרה הרע. חשוב לציין שבשל חוסר האיזון בסט האימון (לווייתנים המופיעים בעשרות תמונות מול לווייתנים המופיעים בתמונות בודדות) נבחר לא "לחתוך" נתח לטובת וולידציה ולהשתמש בכל סט האימון לאימון.

המסקנה היא **overfitting**. התצפיות הן דוגמא קלאסית לתופעה – דיוק גבוה בסט האימון אך נמוך בסט הבדיקה. **ניסיתי להתמודד עם התופעה תוך נקיטה באסטרטגיות מגוונות:**

1. **דגימה עודפת** - Oversampling כפי שהוסבר בסעיף 3.3 על העיבוד המקדים של המידע.
2. **אוגמנטציה של תמונות** – זום/רוטציה/מתיחה באלכסון, לאורך ו/או לרוחב לצורך הגדלה וגיוון של סט האימון.
3. **הגדלת גודל תמונות הקלט** – על מנת לאפשר לרשת לחלץ מאפיינים באופן מדויק יותר.
4. **חיפוש learning rate אופטימאלי** הממקסם את קצב ההתכנסות מבלי לגרום ל-overfitting.

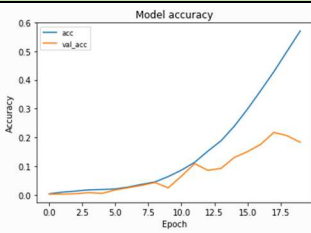
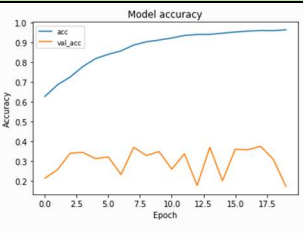
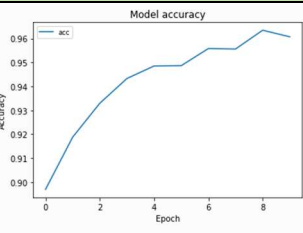
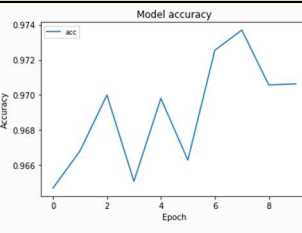
4.5 ניסיונות לשיפור הדיוק**4.5.1 דגימה עודפת, הגדלת תמונות ואוגמנטציה על הרשת שהניבה דיוק מקסימאלי בשלב הקודם:****InceptionResnetV2**

המחברות ופלט ההרצות זמינים בתיקייה "kaggle_kernels\inceptionresnetv2_oversampling". במקרה זה נקטתי בשלושת האסטרטגיות הראשונות מהרשימה למעלה: דגימה עודפת (ללא new_whale כמו בשלב הקודם), אוגמנטציה והגדלת התמונות. דוגמאות האימון נטענו מקובץ "oversampled_train_and_val_shuffled.csv" שיצירתו מתוארת בחלק 3.3 על העיבוד המקדים של המידע. כל תמונה עברה resizing לגודל: 299*299 פיקסלים ואוגמנטציה לפי הפרמטרים בדוגמת הקוד הבאה:

```
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rescale=1./255,
    fill_mode='nearest',
    validation_split = 0.1,
    rotation_range=ROTATE,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=False
)
```

גם כאן נבחר טווח פרמטרים "שמרני" לאוגמנטציה כדי לא לשנות באופן דרסטי את התמונות של הזנבות ולשמור על אחידות מסוימת. הפרמטרים זהים לאלו שפורטו בתחתית טבלה מס' 1.

חתכתי 10% מסט האימון לטובת וולידציה. מפאת אורך ההרצות (7.3 שעות ל-20 epochs!) הרצתי ב-4 שלבים כפי שמתואר בטבלה מס' 4. בטבלה 4 להלן מוצגים גרפים של ה-Accuracy בשלושת ההרצות:

	Epochs 1-20	Epochs 21-40	Epochs 41-50	Epochs 51-60
Accuracy Graph				
Architecture	InceptionResnetV2	same	same	same
Optimizer	Adam (default lr of 0.001)	same	same	same
Validation [%]	10	10	0	0
Run Duration	26,246 sec (~7.3h)	26,307 sec (~7.3h)	14,384 sec (~4h)	14,921 sec (~4h)
Kaggle LB Score		0.348	0.429	0.499

ניתן לראות שהיה שיפור בציון לאחר 2 סבבים של אימון ללא וולידציה (epochs 41-60). ייתכן שחיתוך 10% לטובת וולידציה, גרם להיעדר ייצוג של לווייתנים "נדירים" שמופיעים בתמונות בודדות בסט האימון.

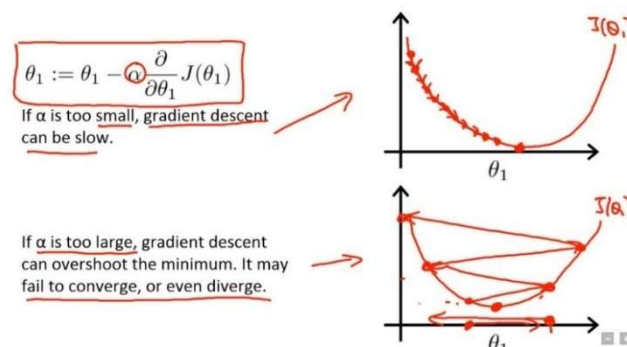
ניסיונות אלו הניבו LB 0.499 שאינו מהווה שיפור ל-LB 0.515 שהתקבל במודל InceptionResnetV2 ללא oversampling ועם תמונות בגודל 128*128. סיבות אפשריות:

- עיבוד מקדים של התמונה - שימוש בפונק' preprocess_input של המודל InceptionResnetV2 ומיצוע עוצמות הפיקסלים ע"י חלוקה ב-255 בכל הערוצים. ייתכן שגודל תמונה כמו 299*299 דורש התאמות אחרות.
- השוואת תמונות GL עם RGB – ייתכן שככל שהתמונות גדולות, יכולת ההשוואה בין תמונות אפורות לצבעוניות נמוכה יותר ודורשת עיבוד והתאמה.
- בחירת פרמטרים לאוגמנטציה – ייתכן שדרושים שינויים "אגרסיביים" יותר כדי למקסם את תרומת דגימת היתר.
- מטא-פרמטרים לא מיטביים - אופטימיזר Adam עם lr ברירת מחדל ופונק' שגיאה Categorical Crosentropy.

4.5.2 חיפוש Learning Rate אופטימאלי לשלוש ארכיטקטורות שהניבו דיוק גבוה בשלב הקודם (ראה טבלה

מס' 1: Inception-v2, Resnet50, Xception

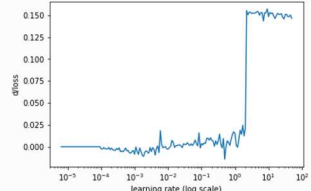
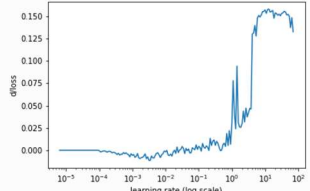
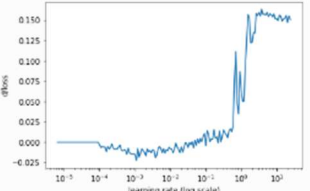
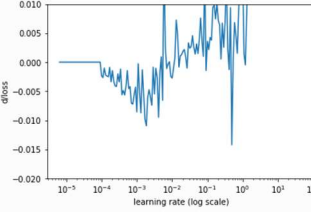
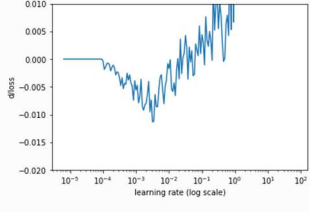
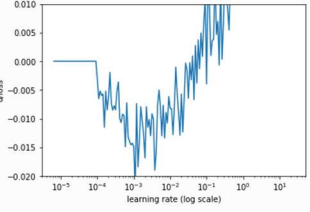
המחברות ופלט ההרצות זמינים בתת התיקיה "kaggle_kernels\learning_rate_optimization".
ה-lr הוא אחד המטא-פרמטרים החשובים ביותר. אם ערכו נמוך מדי, ייקח לרשת זמן ארוך להתכנס כי הצעדים להקטנת השגיאה יהיו מזעריים. לחילופין אם ערכו גדול מדי הרשת עלולה לפספס את המינימום של פונק' השגיאה בשל עדכון "גס" מדי, כפי שמתואר באיור הבא (מקור – קורס Machine Learning מאת Andrew Ng בקורסרה):



במאמר הבא: "Cyclical Learning Rates for Training Neural Networks", 2015 by Leslie N. Smith מומלץ להתחיל אימון מ-lr נמוך ולהגדילו אקספוננציאלית כל אצווה, עד שערך השגיאה יגדל משמעותית. לבסוף יש לבחון את גרף השגיאה כפונקציה של ה-lr, במטרה למצוא את המקטע שבו פונק' השגיאה יורדת בקצב הכי מהיר. ניתן למצוא את ה-lr עבורו השגיאה יורדת בקצב מקסימאלי ע"י חיפוש נק' המינימום בגרף הנגזרת של השגיאה כפונק' של ה-lr. ב-keras אין מימוש זמין לחיפוש lr, לכן מימשי בעצמי בהשראת הפוסטים הבאים:

- <https://www.jeremyjordan.me/nn-learning-rate/>
- <https://towardsdatascience.com/estimating-optimal-learning-rate-for-a-deep-neural-network-ce32f2556ce0>

בטבלה מס' 5 להלן מוצגות תוצאות חיפוש ה-Lr עבור 3 ארכיטקטורות שניבאו זהות לווייתנים בדיוק גבוה יחסית:

	InceptionResNetV2	Xception	ResNet50
Optimizer	Adam	Adam	Adam
Default learning rate	10^{-3}	10^{-3}	10^{-3}
Loss Change Graph			
Loss Change Graph Zoomed In			
Optimized Learning Rate	$\sim 10^{-3}$	$\sim 3 \cdot 10^{-3}$	$\sim 10^{-3}$
Kaggle LB Score with default lr	0.515	0.449	0.426
Kaggle LB Score with optimized lr		0.452	

ניתן לראות בטבלה 5 כי שניים מתוך שלושת החיפושים, עבור המודלים InceptionResNetV2 ו-ResNet50 הניבו ערך קרוב מאוד ל-Lr ברירת המחדל עבור האופטימיזר Adam. לפיכך אין טעם לאמן את הרשתות הללו מחדש עם ה-Lr המיטבי כי הוא זהה ל-Lr ברירת המחדל. רק עבור ארכיטקטורת ה-Xception התקבל ערך שונה: $lr = 0.003$. הרשת אומנה שוב עם ה-Lr המיטבי והתקבל שיפור מזערי (מ-LB 0.449 ל-LB 0.452).

4.5.3 שינוי גישה מ-Similarity Learning Classification

ראינו עד כה שארכיטקטורות שהניבו כ-100% דיוק על ImageNet מדיקות מתחת ל-50% דיוק על זנבות הלווייתנים. זה מרמז על צורך בגישה שונה. בשונה מ-ImageNet מדובר כאן במשימת זיהוי ולא קלסיפיקציה. נתונים 5004 לווייתנים שונים (לא כולל את הלא מזוהים תחת תווית ה-new_whale) וכל פרט מופיע בעשרות תמונות במקרה הטוב או בתמונות בודדות במקרה הרע. גישת Similarity Learning לומדת פונק' דמיון על כל זוג דוגמאות, שמטרתה לכמת עד כמה זהות הדוגמאות הנתונות. גישה זו מתאימה מאוד לזיהוי, מכיוון שהיא עוזרת לקבוע עבור כל דוגמא בסט הבדיקה, לאיזו תווית בסט האימון היא הכי דומה. שיטה זו היא יותר Scalable ביחס לקלסיפיקציה ומתאימה לשימוש ב-production כי ניתן להוסיף תוויות (לווייתנים חדשים) בלי אימון מחדש, הנדרש במקרה של קלסיפיקציה ל-5004 קלאסים קבועים. לפיכך Similarity Learning תתאים לזיהוי לווייתנים באתר <https://happywhale.com/home>, בניגוד לקלסיפיקציה. שיטה זו משמשת גם לזיהוי פנים ואף לזיהוי דולפינים כפי שמסבר במאמרים הבאים:

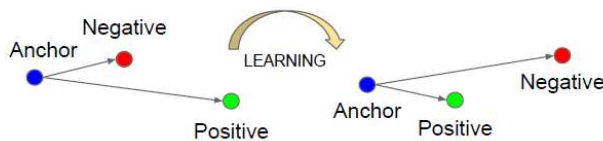
- F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. CVPR, 2015
- S. Bouma, M. Pawley, K. Hupman and A. Gilman Individual common dolphin identification via metric embedding learning. 2019

מפאת חוסר זמן, השתמשתי בקרנל מוכן שמממש Similarity Learning על דטאסט הזנבות:

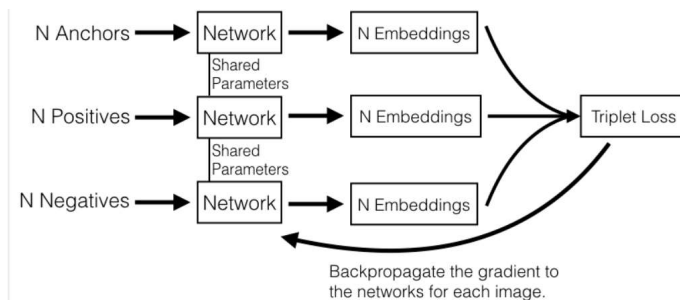
<https://www.kaggle.com/iafoss/similarity-resnext50>

הקרנל מממש רשת בהשראת Triplet Architecture. שבפילתי את הקוד (fork) והתאמתי אותו לעבוד עם דטאסט הזנבות החתוכים שלי (ראה סעיף 3.1 בפרק עיבוד מקדים). המחברת והפלט זמינים בתיקיית "Kaggle_kernels/similarity_learning". **הקרנל בגרסה שלי קיבל 0.752 LB!** לא התייחסתי לתוצאה זו עד כה, גם לא בתחילת פרק התוצאות כי נעשה שימוש בקוד קיים שלא נכתב על ידי. **תוצאה זו מובאת במטרה להראות את היתרון של similarity learning על פני קלאסיפיקציה.** במקרה זה, נלמדות אצוות המורכבות משלישיות של תמונות:

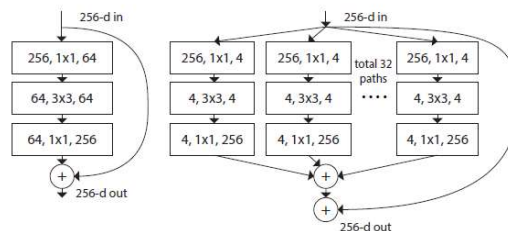
- תמונת Anchor
 - תמונה שונה עם תווית זהה ל-Anchor, הרי היא ה-Positive
 - תמונה שונה עם תווית שונה, הרי היא ה-Negative
- להלן דיאגרמה הממחישה את פעולת ה-Triplet Loss:



Triplet Loss מקטין את המרחק בין תמונת Anchor ו-Positive, כאשר שתיהן **תמונות שונות של אותו לוויתן**. בנוסף, המרחק בין תמונת ה-Anchor ו-Negative המראים זנבות של **לווייתנים שונים** מובא למקסימום. להלן מוצג איור המסביר את שיטת העבודה של Triplet Network:



בקרנל שתואר למעלה, נעשה שימוש ברשת Resnext50 מאומנת עם משקלים מ-ImageNet כ-"שלד". באיור הבא מוצג ההבדל בין בלוק בארכיטקטורת ResNet50 (סיכום מפורט על הארכיטקטורה ניתן בממן 13) לבין בלוק בארכיטקטורת ResNext50:



על השלד של ResNext50 התווספו: שכבת AdaptivePooling שתומכת במגוון גדלים של תמונות (AdaptiveConcatPool2D). בראש הרשת, הוספו 2 שכבות FC על מנת להמיר את ה-predictions של חלק ה-CNN ל-embedding space. לבסוף, הוספה מטריקה מותאמת ללימוד המרחקים הווקטוריים ב-embedding space עבור כל התמונות ב-batch. בהרצה הנוכחית התמונות עברו resizing לגודל 224×224 ואוגמנטציה. האיור הבא ממחיש את מבנה הרשת עם הסתייגות: **במקום L2 נעשה כאן שימוש במטריקה מותאמת במרחב ממדים גבוה יותר:**



בשלב הטסט, נעשה שימוש בשיטת K Nearest Neighbors על מנת לחלץ את חמשת ה-embeddings (עם תוויות שונות) הקרובים ביותר לכל תמונה בסט הבדיקה. שלישיות האימון עצמן נבחרות באופן "מתוחכם" ולא רנדומאלי. מבוצע גילוי (mining) של השלישיות הקשות ביותר כדי להרכיב כל batch אימון: **בבחרות 64 התמונות הזרות ביותר עם תוויות שונה עבור כל תמונת Anchor**. טכניקה זו היא המפתח להצלחת הניבוי של הקרנל, מכיוון שדגימה רנדומאלית מניבה תוצאות נמוכות מאוד.

5. סיכום

זיהוי לווייתנים מסוג Humpback לפי מבנה הזנב בכלל והתחרות הנתונה ב-Kaggle בפרט מהווים אתגר לא פשוט. בניגוד לדטאסטים שפגשנו בממנים קודמים (e.g. MNIST, ImageNet), כמות תמונות הזנב פר לווייתן קטנה יחסית (בקירוב בין 1 ל-73) לעומת כמות תוויות גדולה (5-5004, תלוי אם כוללים את ה-new_whale). יתרה מכך, לא ידועה כמות התוויות "האמיתית", הרי התוויות המאגדת את הלווייתנים הבלתי מזוהים, ה-new_whale ככל הנראה אוצרת בתוכה מס' תוויות (זהויות חדשות של לווייתנים) לא ידוע. עשיית שימוש ברשתות CNN שנלמדו בקורס. בחרתי בארכיטקטורות שהוכיחו עצמן על דטאסט ה-ImageNet ואף עברו את הדיוק האנושי בקלסיפיקציה של התמונות הנתונות. **להלן טבלה מס' 6 המסכמת את רוב התוצאות** (למעט תוצאות ה-ensemble המסוכמות בטבלה מס' 1 בפרק 4).

#	Model	Size	Top-1 Accuracy ImageNet	Top-5 Accuracy ImageNet	Parameters	Depth	Whales Kaggle LB Score	Whales Kaggle LB Score Oversampling	Whales Kaggle LB Score Optimal LR
1	VGG16	528 MB	0.713	0.901	138,357,544	23	0.284		
2	ResNet50	99 MB	0.749	0.921	25,636,712	168	0.426		
3	InceptionV3	92 MB	0.779	0.937	23,851,784	159	0.316		
4	Xception	88 MB	0.790	0.945	22,910,480	126	0.449		0.452
5	InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572	0.515	0.499	

מטבלה 6 ניתן לראות קורלציה בין עומק הרשת לבין ביצועיה על דטאסט הזנבות – רשת InceptionResnetV2 העמוקה ביותר הניבה Kaggle LB Score הכי גבוה. תצפית זו עומדת בקנה אחד עם המורכבות הרבה של מאפייני זנבות הלווייתן, הדורשת יכולות קלסיפיקציה גבוהות מהיכולת האנושית לסווג תמונות בכלל וזנבות לווייתנים בפרט. חשוב להזכיר את זמני הריצה הארוכים (9-3 שעות!) ומשאבי החומרה המיוחדים הנדרשים לאימון הרשתות (ראה פירוט בפרק 2 – שיטות עבודה). בנוסף לתוצאות ההתחלתיות (עמודת Whales Kaggle LB Score בטבלה 6) שבוצעו על תמונות בגודל 100*100 עד 128*128 שניתן לזכרון ה-RAM הזמין בקאגל, נעשו ניסיונות לשפר ביצועים ע"י הגדלת תמונות, איזון כמות התמונות של הלווייתנים הנצפים הכי פחות, אוגמנטציה של תמונות ואף חיפוש learning rate אופטימאלי. ניסיונות אלו לא שיפרו ביצועים באופן מובהק כפי שמוצג בשתי העמודות הימניות בטבלה 6.

שימוש ברשתות מאומנות מראש לצורך קלאסיפיקציה של זנבות הלווייתנים התגלה כבעל יעילות מוגבלת. **הושג דיוק מקסימאלי של Kaggle LB Score 0.545 ע"י מיצוע של 3 סטים של ההסתברויות מיטביות שהתקבלו: ResNet50, InceptionResNetV2 ו-Xception כמפורט בטבלה 1 בפרק התוצאות.**

בשל אופי המשימה – זיהוי לווייתנים נתונים ולא דווקא קלסיפיקציה, מומלץ להשתמש בשיטות שלא נלמדו בקורס, כגון **Similarity Learning ו-Triplet Loss** שהוסברה בחלק 4.5.3. עשיית שימוש בקוד קיים על דטאסט תמונות שיצרת (ראה חלק 3.1 בפרק העיבוד המקדים) וקיבלתי **Kaggle LB Score 0.752** המהווה שיפור משמעותי על פני הארכיטקטורות שפורטו למעלה. בהינתן הזמן והמשאבים המתאימים, הייתי בוחרת להתעמק בכיוון זה של Similarity Learning וממשת בעצמי רשת המזהה לווייתנים בשיטה זו. במגבלות הזמן הנתונות, סיפקתי פירוט על ארכיטקטורת הרשת ובחירת דוגמאות האימון, שהיוו מפתח להגדלת הדיוק. **בדיקה נעימה!!!**