יונתן מנחם – 203772611
אלון מצרי – 311503841

למידה עמוקה ליישומי ראיה ממוחשבת – זיהוי דלקת ריאות

בחלק השני של עבודה זו נדרשנו להשתמש במודל רשת נוירונים מאומן לבחירתנו אשר ידע להבחין בין שני סוגים של צילומי דלקת ריאות, תקין (=בריא) ושאינו תקין (=חולה).

למעבר לגיט לחצ/י כאן.

חשוב לציין כי נעשתה עבודה רבה באופטימיזציה של המודל והמענה על הסעיפים השונים לא בהכרח יוצג בסדר המתאים. בנוסף, חלק מסעיפי העבודה כבר בוצעו ונענו בעבודה הראשונה אך אציין אותם בהתאם גם כאן.

בחרנו לנסות מספר ארכיטקטורות על מנת להגיע לזו שנתנה את הביצועים הטובים ביותר לטעמנו.

ResNet152

```
from tensorflow.keras.applications import ResNet152

num_classes = 2
input_shape = (256, 256, 3)

pretrained_resnet = ResNet152(include_top=False, input_tensor=None, input_shape=input_shape, pooling=None)

#freazing the trained layers
for layers in pretrained_resnet.layers:
    layers.trainable = False

pretrained_resnet.summary()
```

איור 1: מודל ResNet152 המאומן מראש.

בתחילה, טענו את הרשת למשתנה מקומי [איור מס. 1] וביצענו נעילה לכל השכבות על מנת שלא ישתתפו באימון.

לרשת הנ"ל קיימים 58,370,944 פרמטרים.

לאחר מכן נדרשנו לאמן את השכבות האחרונות של הרשת החל משכבה מסוימת לבחירתנו [איור מס. 2].

```
last_layer = pretrained_resnet.get_layer('conv5_block3_1_relu')
last_output = last_layer.output


x=tf.keras.layers.Flatten()(last_output)
x=tf.keras.layers.Dense(128, activation='relu')(x)
x=tf.keras.layers.Dropout(0.1)(x)
x=tf.keras.layers.Dense(num_classes, activation='sigmoid')(x)

resnet_model = tf.keras.Model(pretrained_resnet.input, x)

METRICS = ['accuracy']

resnet_model.compile(loss='binary_crossentropy', optimizer="adam", metrics=METRICS)

resnet_model.summary()
```

איור 2: הוספת שכבות הניתנות לאימון במוצא מודל ResNet152

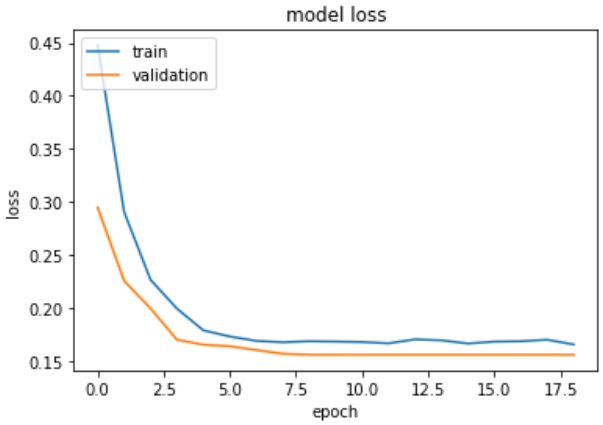
למעשה, החל משכבת הקונבולוציה 'conv5_block3_1_relu' הוספנו 4 שכבות כאשר החשובה ביניהן הייתה האחרונה על מנת שבמוצא המודל תהיה שכבת אקטיבציה שתסווג את הדאטה לשתי הקטגוריות שלו.

גם עבור הרשת הזו ובדומה לעבודה מספר 1, השתמשנו במספר טכניקות אותן רכשנו קודם לכן אך הפעם מבלי לבחון את ביצועי הרשת טרם השינוי. למשל: Learning Rate Scheduler, Early Stopping ועוד.

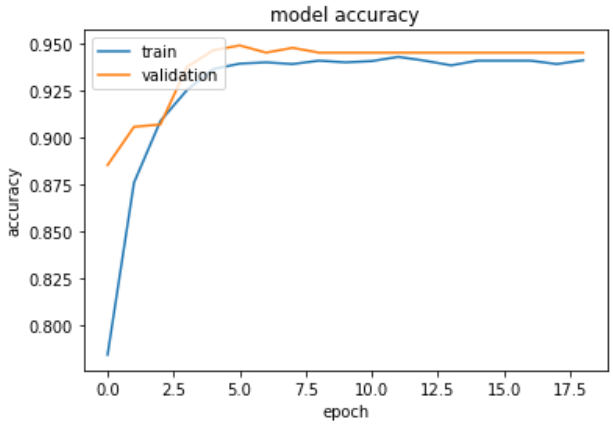
כמות הפרמטרים אותם אימנו ברשת זו היא 4,194,690.

יונתן מנחם – 203772611
אלון מצרי – 311503841

לאחר אימון הרשת ניתן לראות כי הרשת התכנסה [גרפים 1,2] אך דיוק הרשת ("Accuracy") עומד על כ- 73% [איור 3].



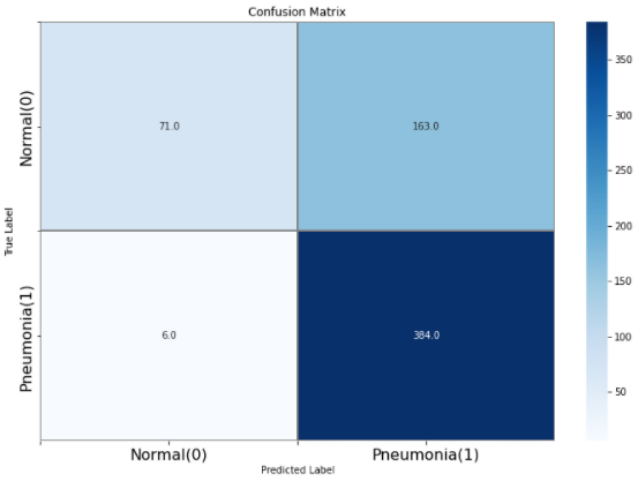
גרף 1: גרף ה-Loss כפונקציה של ה-epoch



גרף 2: דיוק המודל על סט האימון והוולידציה כפונקציה של ה-epoch

```
loss, accuracy = resnet_model.evaluate(test_ds)
print("Test loss:", loss)
print("Test accuracy:", accuracy)

20/20 [==============================] - 3s 123ms/step
Test loss: 0.6263019442558289
Test accuracy: 0.7291666865348816
```

איור 3: דיוק המודל ה-ResNet152



איור 4: מטריצת המבוכה עבור המודל הבסיסי

ניתן לראות על פי מטריצת המבוכה [איור 4] את דיוק המודל ומשם ניתן לגזור מספר מטריקות ביצועים (Accuracy, Recall, Precision).

דנו בחלק הראשון של העבודה בחשיבות של בחירת המטריקה המתאימה לבחינת ביצועי המודל וציינו כי בחירת מטריקת Recall היה העדיפה ביותר לפרויקט הנ"ל. ניתן לראות את תוצאות המטריקות השונות באיור 7.

ניתן לראות כי המודל הבסיסי קיבל ציון Recall של 98% (כאשר חיובי=דלקת ושלילי=בריא). מצד שני, תוצאת ציון ה-Precision היה 70% והמודל סיווג בשוגג מטופלים בריאים כחולים.

```
# prescision and recall calculation

from sklearn.metrics import classification_report
print(classification_report(test_labels, predictions))

              precision    recall  f1-score   support

         0.0       0.92      0.30      0.46       234
         1.0       0.70      0.98      0.82       390

    accuracy                           0.73       624
   macro avg       0.81      0.64      0.64       624
weighted avg       0.78      0.73      0.68       624
```

איור 7: תוצאות המטריקות השונות על המודל ה-ResNet152

יונתן מנחם – 203772611
אלון מצרי – 311503841

DenseNet

בדומה לארכיטקטורת ה-ResNet152 גם כאן ביצענו טעינה של הרשת למשתנה מקומי ונעלנו את כל השכבות על מנת שלא יהיו נלמדות [איור מס. 8]. לרשת קיימים 7,037,504 פרמטרים.

לאחר מכן, הוספנו ביציאת המודל את השכבות שיתאימו את המוצא לבעיה אותה אנו מנסים לפתור [איור מס. 9].

```
from tensorflow.keras.applications import DenseNet121

num_classes = 2
input_shape = (256, 256, 3)

pretrained_densenet = DenseNet121(include_top=False, input_tensor=None, input_shape=input_shape, pooling=None)

#freazing the trained layers
for layers in pretrained_densenet.layers:
    layers.trainable = False

pretrained_densenet.summary()
```

איור 8: מודל ה-DenseNet המאומן מראש.

גם כאן, בחרנו באופן שרירותי את השכבה שבה יסתיים המודל המאומן ('conv5_block15_1_relu') והחל ממנה נתחיל לאמן את הרשת.

```
last_layer = pretrained_densenet.get_layer('conv5_block15_1_relu')
last_output = last_layer.output


x=tf.keras.layers.Flatten()(last_output)
x=tf.keras.layers.Dense(128, activation='relu')(x)
x=tf.keras.layers.Dropout(0.1)(x)
x=tf.keras.layers.Dense(num_classes, activation='sigmoid')(x)

densenet_model = tf.keras.Model(pretrained_densenet.input, x)

METRICS = ['accuracy']

densenet_model.compile(loss='binary_crossentropy', optimizer="adam", metrics=METRICS)

densenet_model.summary()
```

איור 9: הוספת שכבות הניתנות לאימון במוצא מודל DenseNet

גם עבור הרשת הזו ובדומה לעבודה מספר 1, השתמשנו במספר טכניקות אותן רכשנו קודם לכן אך גם הפעם מבלי לבחון את ביצועי הרשת טרם השינוי. למשל: Learning Rate Scheduler, Early Stopping [איורים 10-11] ועוד.

```
from keras.callbacks import LearningRateScheduler

# This is a sample of a scheduler I used in the past
def lr_scheduler(epoch, lr):
    decay_rate = 0.85
    decay_step = 1
    if epoch % decay_step == 0 and epoch:
        return lr * pow(decay_rate, np.floor(epoch / decay_step))
    return lr
```

איור 10: Learning Rate Scheduler

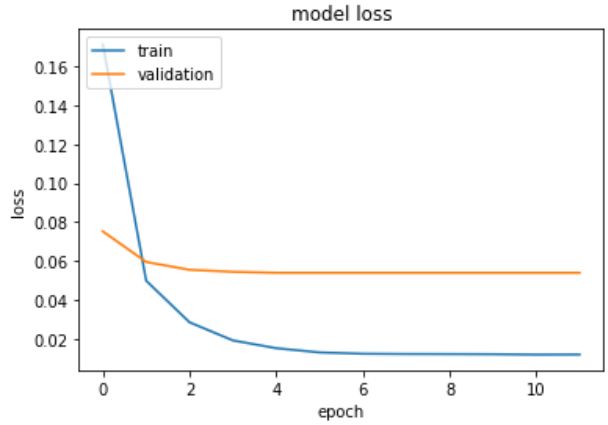
```
early_stopping = tf.keras.callbacks.EarlyStopping(patience=5, restore_best_weights=True)
```

איור 11: Early Stopping

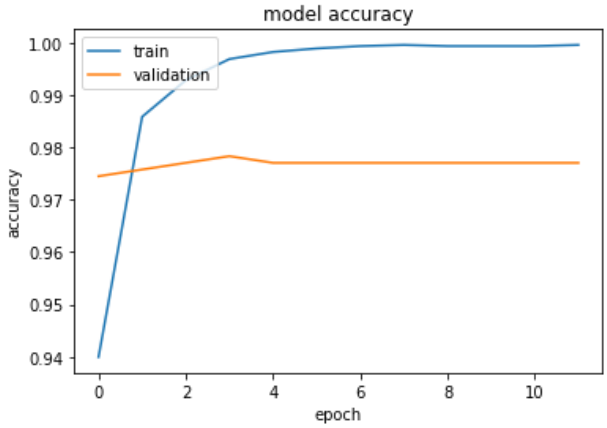
כמות הפרמטרים אותם אימנו ברשת זו היא 1,048,962.

יונתן מנחם – 203772611
אלון מצרי – 311503841

לאחר אימון הרשת ניתן לראות כי הרשת התכנסה [גרפים 3,4] אך דיוק הרשת ("Accuracy") עומד על כ- 73% [איור 12].



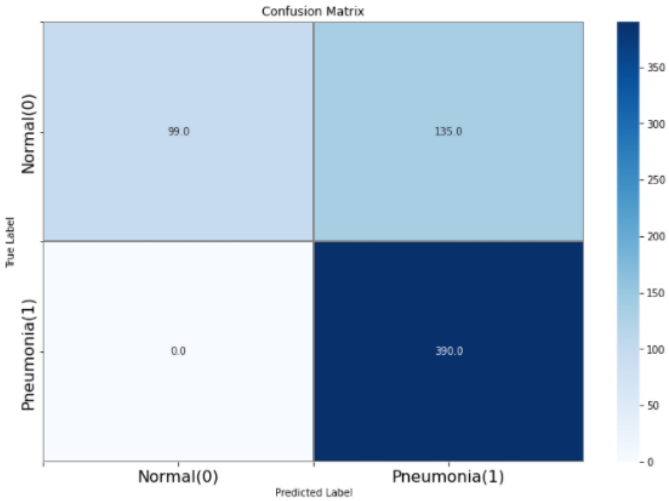
גרף 3: גרף ה-Loss כפונקציה של ה-epoch



גרף 4: דיוק המודל על סט האימון והווליידציה כפונקציה של ה-epoch

```
loss, accuracy = densenet_model.evaluate(test_ds)
print("Test loss:", loss)
print("Test accuracy:", accuracy)

20/20 [==============================] - 1s 60ms/step
Test loss: 0.7571943402290344
Test accuracy: 0.7836538553237915
```

איור 12: דיוק מודל DenseNet



איור 13: מטריצת המבוכה עבור מודל DenseNet

על בסיס מטריצת המבוכה [איור 13] ניתן לראות כי חל שיפור במודל ה-DenseNet מבחינת מטריקת ה-Recall וה-Precision.

```
# prescision and recall calculation

from sklearn.metrics import classification_report
print(classification_report(test_labels, predictions))

              precision    recall  f1-score   support

         0.0       1.00      0.42      0.59       234
         1.0       0.74      1.00      0.85       390

    accuracy                           0.78       624
   macro avg       0.87      0.71      0.72       624
weighted avg       0.84      0.78      0.76       624
```

איור 14: תוצאות המטריקות השונות עבור מודל DenseNet

ניתן לראות [איור 14] כי המודל המתקדם קיבל ציון Recall של 100% (כאשר חיובי=דלקת ושלילי=בריא). מצד שני, תוצאת ה-Precision הייתה 74% והמודל סיווג בשוגג מטופלים בריאים כחולים.

יונתן מנחם – 203772611
אלון מצרי – 311503841

InceptionV3

בדומה לארכיטקטורות ה-ResNet152 ו-DenseNet גם כאן ביצענו טעינה של הרשת למשתנה מקומי ונעלנו את כל השכבות על מנת שלא יהיו נלמדות [איור מס. 15]. לרשת קיימים 21,802784 פרמטרים.

לאחר מכן, הוספנו ביציאת המודל את השכבות שיתאימו את המוצא לבעיה אותה אנו מנסים לפתור [איור מס. 16].

```
from tensorflow.keras.applications import InceptionV3

num_classes = 2
input_shape = (256, 256, 3)

pretrained_inception = InceptionV3(include_top=False, input_tensor=None, input_shape=input_shape, pooling=None)

#freazing the trained layers
for layers in pretrained_inception.layers:
    layers.trainable = False

last_layer_name = 'conv2d_86'
last_index = 0
idx = 0
for layer in pretrained_inception.layers:
    if layer.name == last_layer_name:
        last_index = idx
        print('end layer number', last_index)
    idx += 1

pretrained_inception.summary()
```

איור 15: מודל ה-InceptionV3 המאומן מראש.

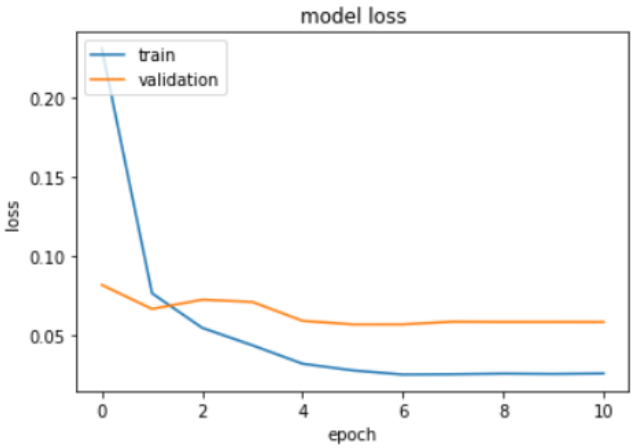
```
last_layer = pretrained_inception.get_layer(index=last_index)
last_output = last_layer.output


x=tf.keras.layers.Flatten()(last_output)
x=tf.keras.layers.Dense(128, activation='relu')(x)
x=tf.keras.layers.Dropout(0.1)(x)
x=tf.keras.layers.Dense(num_classes, activation='sigmoid')(x)

inception_model = tf.keras.Model(pretrained_inception.input, x)

METRICS = ['accuracy']

inception_model.compile(loss='binary_crossentropy', optimizer="adam", metrics=METRICS)

inception_model.summary()
```

גם כאן, בחרנו באופן שרירותי את השכבה שבה יסתיים המודל המאומן ('conv2d_86') והחל ממנה נתחיל לאמן את הרשת.

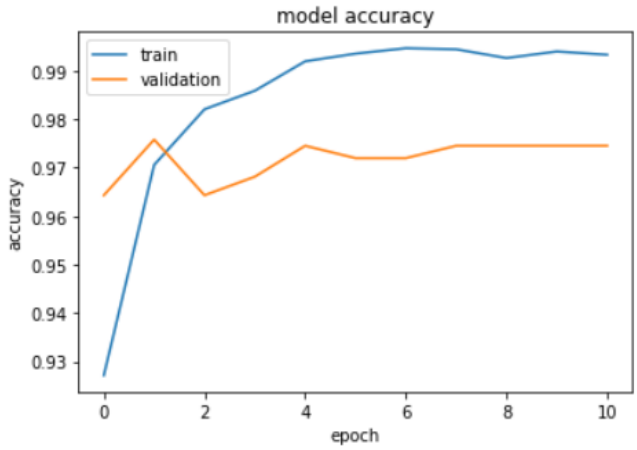
כמות הפרמטרים אותם אימנו ברשת זו היא 1,769,858.

איור 16: הוספת שכבות הניתנות לאימון במוצא מודל InceptionV3

לאחר אימון הרשת ניתן לראות כי הרשת התכנסה [גרפים 5,6] ודיוק הרשת ("Accuracy") עומד על כ- 78% [איור 17].



גרף 5: גרף ה-Loss כפונקציה של ה-epoch



גרף 6: דיוק המודל על סט האימון והווליידציה כפונקציה של ה-epoch

יונתן מנחם – 203772611
אלון מצרי – 311503841

על בסיס מטריצת המבוכה [איור 17] ניתן לראות כי חל שיפור קל במודל ה-DenseNet מבחינת מטריקת ה-Recall וה-Precision.

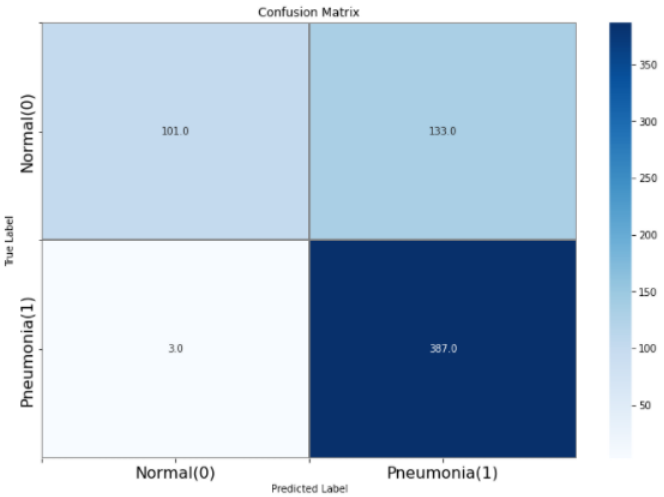
ניתן לראות [איור 18] כי המודל המתקדם קיבל ציון Recall של 99% (כאשר חיובי=דלקת ושלילי=בריא). מצד שני, תוצאת ה-Precision הייתה 74% והמודל סיווג בשוגג מטופלים בריאים כחולים.

```
loss, accuracy = inception_model.evaluate(test_ds)
print("Test loss:", loss)
print("Test accuracy:", accuracy)

20/20 [==============================] - 1s 47ms/step
Test loss: 0.7936044931411743
Test accuracy: 0.7820512652397156
```

איור 16: דיוק מודל InceptionV3

```
# prescision and recall calculation

from sklearn.metrics import classification_report
print(classification_report(test_labels, predictions))

              precision    recall  f1-score   support

         0.0       0.97      0.43      0.60       234
         1.0       0.74      0.99      0.85       390

    accuracy                           0.78       624
   macro avg       0.86      0.71      0.72       624
weighted avg       0.83      0.78      0.76       624
```

איור 18: תוצאות המטריקות השונות עבור מודל InceptionV3



איור 17: מטריצת המבוכה עבור מודל InceptionV3

בעקבות ביצועי המודל הן מבחינת אורך זמן האימון והן מבחינת ביצועים, בחרנו למדוד את השפעת אימון השכבת האמצעיות של המודל לעומת אימון השכבות האחרונות על מודל ה-InceptionV3.

הפעם, על מנת לגשת רק לשכבות האמצעיות של המודל, בחרנו באופן שרירותי 3 שכבות [איור 18 א' ב']:

1. שכבת התחלה שהחל ממנה כל השכבות יוגדרו כניתנות לאימון
2. שכבת סוף שהחל ממנה השכבות יוגדרו כבלתי ניתנות לאימון
3. שכבה אחרונה שבה יסתיים המודל ויתווספו השכבות שרלוונטיות לבעיה שלנו.

```
#freazing the trained layers
start_layer = 'conv2d_44'
end_layer = 'average_pooling2d_4'
last_layer_name = 'conv2d_86'
start_index = 0
end_index = 0
last_index = 0
idx = 0
for layer in pretrained_inception_mid.layers:
    if layer.name == start_layer:
        start_index = idx
        print('start layer number', start_index)
    if layer.name == end_layer:
        end_index = idx
        print('end layer number', end_index)
    if layer.name == last_layer_name:
        last_index = idx
        print('end layer number', last_index)
    idx += 1

start layer number 133
end layer number 151
end layer number 283
```

איור 18 א': חלוקת השכבות עבור מודל InceptionV3

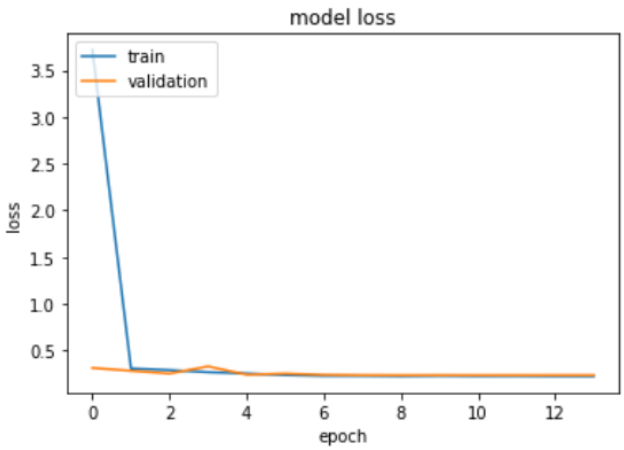
```
#freazing the trained layers

for layer in pretrained_inception_mid.layers[:start_index]:
    layer.trainable=False
for layer in pretrained_inception_mid.layers[start_index:end_index]:
    layer.trainable=True
for layer in pretrained_inception_mid.layers[end_index:]:
    layer.trainable=False
```

איור 18 ב': חלוקת השכבות עבור מודל InceptionV3

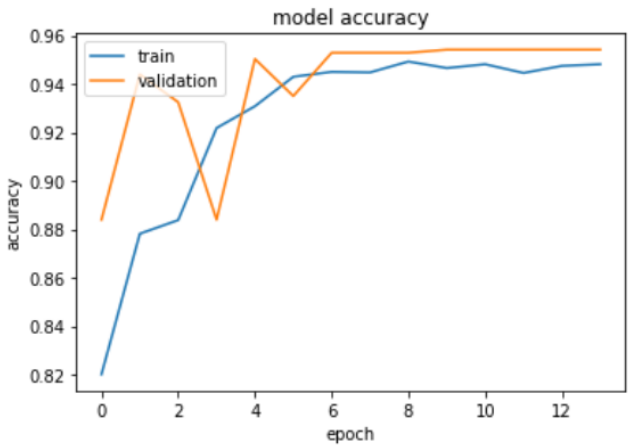
כמות הפרמטרים אותם אימנו ברשת זו היא 2,733,378.

יונתן מנחם – 203772611
אלון מצרי – 311503841

לאחר אימון הרשת ניתן לראות כי הרשת התכנסה [גרפים 7,8] ודיוק הרשת ("Accuracy") עומד על כ- 75% [איור 19].



גרף 7: גרף ה-Loss כפונקציה של ה-epoch



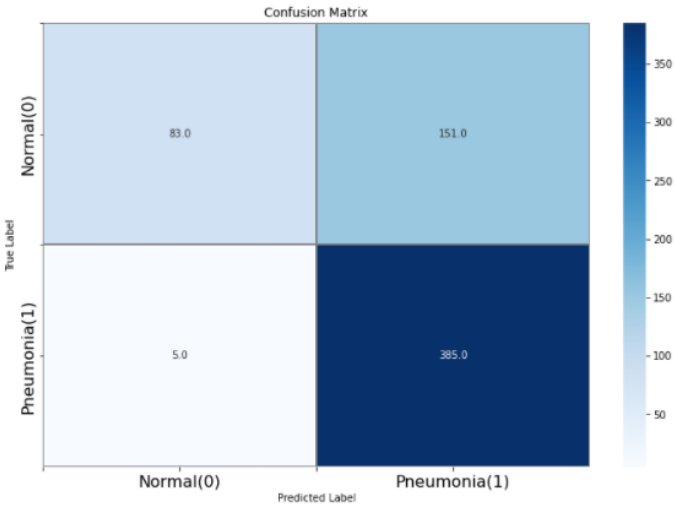
גרף 8: דיוק המודל על סט האימון והוולידציה כפונקציה של ה-epoch

על בסיס מטריצת המבוכה [איור 20] ניתן לראות כי במודל ה-InceptionV3 לאחר אימון השכבות האמצעיות, מבחינת מטריקת ה-Recall וה-Precision.

ניתן לראות [איור 21] כי המודל קיבל ציון Recall של 99% (כאשר חיובי=דלקת ושלילי=בריא). מצד שני, תוצאת ה-Precision הייתה 72% והמודל סיווג בשוגג מטופלים בריאים כחולים.

```
loss, accuracy = inception_mid_model.evaluate(test_ds)
print("Test loss:", loss)
print("Test accuracy:", accuracy)

20/20 [==============================] - 0s 5ms/step - los
Test loss: 0.7444496750831604
Test accuracy: 0.75
```

איור 19: דיוק מודל InceptionV3 עבור השכבות האמצעיות



איור 20: מטריצת המבוכה עבור מודל InceptionV3

```
# prescision and recall calculation

from sklearn.metrics import classification_report
print(classification_report(test_labels, predictions))

              precision    recall  f1-score   support

         0.0       0.94      0.35      0.52       234
         1.0       0.72      0.99      0.83       390

    accuracy                           0.75       624
   macro avg       0.83      0.67      0.67       624
weighted avg       0.80      0.75      0.71       624
```

איור 21: תוצאות המטריקות השונות עבור מודל InceptionV3

ניתן לראות כי בהשוואה לתוצאות המודל שבו לא אימנו את השכבות האמצעיות דיוק המודל ירד מבחינת כל המטריקות. הירידה בביצועים יכולה לנבוע מכמה סיבות אך בעיקר מבחירת השכבות הנלמדות. בסופו של דבר בחירת השכבות ללימוד היא היפר פרמטר בפני עצמו שניתן ללמוד רק לאחר ניסוי וטעיה.

יונתן מנחם – 203772611
אלון מצרי – 311503841

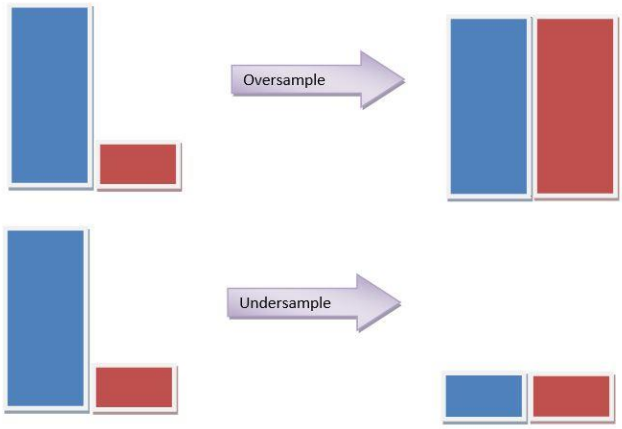
בעיית חוסר האיזון

בתחום הלמידה העמוקה ובעיקר כאשר מדובר בבעיית קלסיפיקציה רפואית, לעיתים קיים חוסר איזון בדאטה.

חוסר האיזון בא לידי ביטוי בכמות מידע שונה עבור כל קטגוריה ובמקרה שלנו כמות הדגימות התקינות (=בריא) קטנה בערך פי 3 מכמות הדגימות שאינן תקינות (=דלקת ריאות).

קיימות מספר שיטות שבאמצעותן ניתן להתמודד עם בעיית חוסר איזון. אנחנו הולכים לסקור כמה מהן אך לא את כולן. חשוב לציין שאת כל השיטות יש לבצע אך ורק על סט האימון ולא על סט הוולידציה או הטסט.

1. Resampling – Oversampling / Undersampling
2. Ensemble
3. Class Weight



Resampling – Oversampling / Undersampling

באמצעות Undersampling אנו למעשה מצמצמים את כמות דגימות הרוב כך שתתאים לכמות דגימות המיעוט ובעצם כך תייצר איזון בין הקטגוריות השונות.

לעומת זאת, באמצעות Oversampling מרחיבים את כמות הדגימות המיעוט באמצעות שימוש באוגמנטציות מסוגים שונים. בעצם יוצרים שכפולים מעט שונים של הדאטה הקיים כך שיתאים לכמות דגימות הרוב.

Ensemble

באמצעות שיטה זו אנו מחלקים את דגימות הרוב למספר מקטעים שתהיה תואמות את כמות דגימות המיעוט ויוצרים מודל נפרד לכל מקטע. מבצעים אימון לשלושת המודלים כך שדגימות המיעוט זהות עבור שלושת המודלים אך דגימות הרוב המחולקות הן שונות. בסוף האימון, ההחלטה המתקבלת היא שקלול של כל המודלים שאומנו בשיטת הרוב קובע.

Class Weight

בשיטה זו דנו כבר בעבודה הראשונה, ובאמצעותה מגדירים משקל גבוהה לדגימות המיעוט ומשקל נמוך לדגימות הרוב בצורה שתשקף את היחס ביניהן.

יונתן מנחם – 203772611
אלון מצרי – 311503841

בחרנו לממש בעבודה זו את שיטת Oversampling ובנוסף שילבנו שוב את Class Weight אך לא במקביל מאחר ואיזון הדאטה על ידי Oversampling מייתר את שינוי המשקולים.

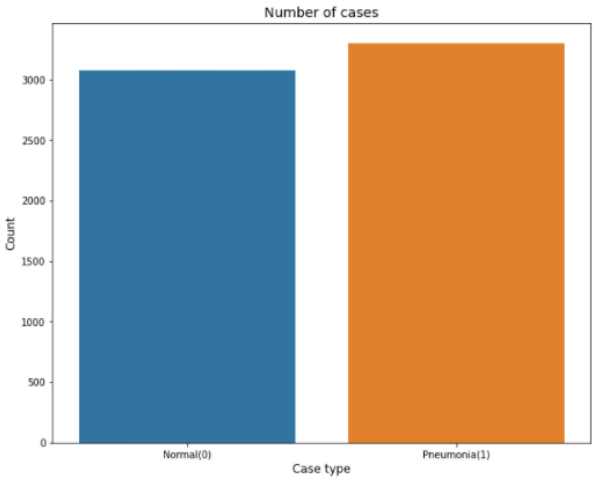
על מנת לבצע איזון לדאטה השתמשנו בחבילת albumentations ובאמצעותה ביצענו אוגמנטציות שונות לדאטה [איור מס. 22].

```
import albumentations as A
import cv2
import os
from matplotlib import pyplot as plt

# Declare an augmentation pipeline
transform = A.Compose([
    A.CropAndPad(percent=-0.15, keep_size=False, sample_independently=True, interpolation=cv2.INTER_LANCZOS4, always_apply=False, p=0.5),
    A.Rotate(limit=18, interpolation=1, border_mode=4, value=None, mask_value=None, always_apply=False, p=0.5),
    A.RandomBrightnessContrast(brightness_limit=0.2, contrast_limit=0.2, brightness_by_max=True, always_apply=False, p=0.5),
    A.RandomScale(scale_limit=0.1, interpolation=1, always_apply=False, p=0.5),
    A.ElasticTransform(alpha=1, sigma=50, alpha_affine=50, interpolation=1, border_mode=4, always_apply=False, p=0.5),
    A.GaussNoise(var_limit=(10.0, 50.0), mean=0, per_channel=True, always_apply=False, p=0.5),
])
```

איור 22: שימוש בחבילת albumentations לביצוע אוגמנטציות

לאחר מכן הפרדנו ללא ביצוע אוגמנטציות בין סט הוולידציה לסט האימון וביצענו שינוי ושכפול רק עבור דגימות ה-NORMAL של סט האימון על מנת להגיע לאיזון בין שתי הקטגוריות.

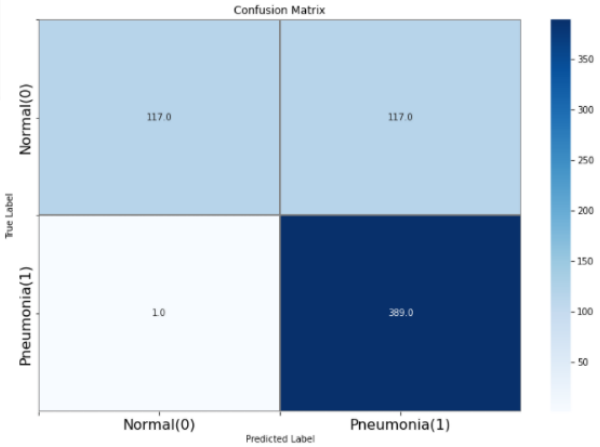
מלבד זאת, מימוש המודל היה זהה לחלוטין למימוש מודל ה-InceptionV3 שבו אימנו את השכבות האמצעיות, למעט שינוי הדאטה, שבמקרה הנ"ל היה מאוזן [איור מס. 23].



איור 23: התפלגות התמונות בסט האימון לפי קטגוריה לאחר ביצוע ה- Oversampling

בניגוד לציפיות, ביצועי המודל לאחר השימוש ב-Oversampling לא השתפרו באופן ניכר.

ניתן לראות [איור 25] כי המודל קיבל ציון Recall של 100% (כאשר חיובי=דלקת ושלילי=בריא). מצד שני, תוצאת ה-Precision הייתה 77% והמודל סיווג בשוגג מטופלים בריאים כחולים.



איור 24: מטריצת המבוכה עבור מודל InceptionV3 תוך שימוש ב-Oversampling

```
# prescision and recall calculation

from sklearn.metrics import classification_report
print(classification_report(test_labels, predictions))
```

	precision	recall	f1-score	support
0.0	0.99	0.50	0.66	234
1.0	0.77	1.00	0.87	390
accuracy			0.81	624
macro avg	0.88	0.75	0.77	624
weighted avg	0.85	0.81	0.79	624

איור 25: תוצאות המטריקות השונות עבור מודל InceptionV3 תוך שימוש ב-Oversampling

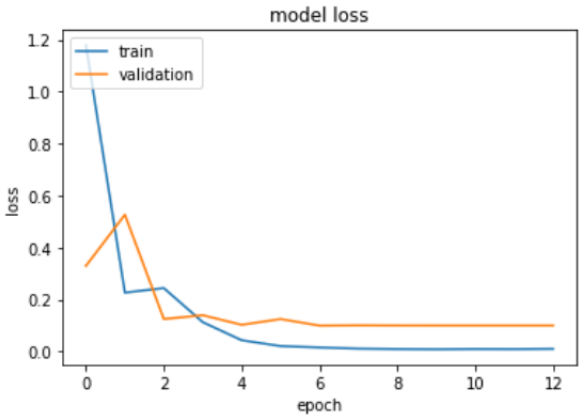
יונתן מנחם – 203772611
אלון מצרי – 311503841

כאשר שילבנו את שיטת Class Weight בחרנו הפעם לממש אותה עבור מודל ResNet50V2 ללא אימון השכבות האמצעיות משום שטרם יצא לנו להתנסות בו.

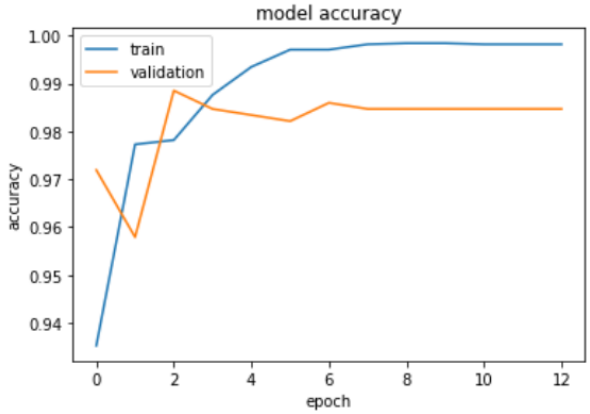
```
weight_for_0 = (1 / COUNT_NORMAL)*(TRAIN_IMG_COUNT)/2.0
weight_for_1 = (1 / COUNT_PNEUMONIA)*(TRAIN_IMG_COUNT)/2.0

class_weight = {0: weight_for_0, 1: weight_for_1}

print('Weight for class 0: {:.2f}'.format(weight_for_0))
print('Weight for class 1: {:.2f}'.format(weight_for_1))
```

איור 26: בחירת המשקולים למודל

על מנת לבחור משקולים, מצאנו ברשת מודל פשוט לחישוב משקולים על בסיס הכמות היחסית בין הקטגוריות השונות [איור מס. 26].



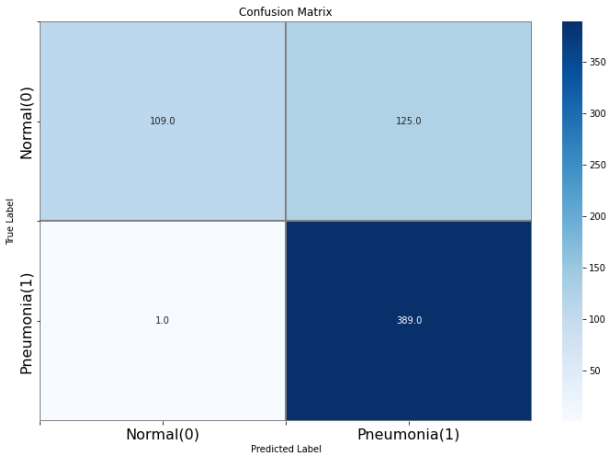
גרף 9: גרף ה-Loss כפונקציה של ה-epoch



גרף 10: דיוק המודל על סט האימון והוולידציה כפונקציה של ה-

על בסיס מטריצת המבוכה [איור 27] ניתן לראות כי במודל ה-ResNet50V2 לאחר שילוב Class weight, מבחינת מטריקת ה-Recall וה-Precision חל שיפור קל.

ניתן לראות [איור 28] כי המודל קיבל ציון Recall של 99% (כאשר חיובי=דלקת ושלילי=בריא). מצד שני, תוצאת ה-Precision הייתה 72% והמודל סיווג בשוגג מטופלים בריאים כחולים.



איור 27: מטריצת המבוכה עבור מודל ResNet50V2 בשילוב עם Class Weight

```
# prescision and recall calculation

from sklearn.metrics import classification_report
print(classification_report(test_labels, predictions))
✓ 0.4s
```

	precision	recall	f1-score	support
0.0	0.99	0.47	0.63	234
1.0	0.76	1.00	0.86	390
accuracy			0.80	624
macro avg	0.87	0.73	0.75	624
weighted avg	0.84	0.80	0.78	624

איור 28: תוצאות המטריקות השונות עבור מודל ResNet50V2 בשילוב עם Class Weight

לסיכום, לאחר שילוב המודלים בשיטת Transfer Learning, למדנו כי אמנם קל יותר מבחינה חישובית להשתמש במודל שלמד מראש על ידי שימוש בכוח חישוב ובמאגר תמונות גדול, מדובר באליה וקוץ בה, משום שבמקרה הנ"ל המודל לא בהכרח נתקל במאגר צילומי רנטגן ועלינו לבצע אופטימיציות שונות על מנת למצוא את כל הפרמטרים הרלוונטיים לצורך קלסיפיקציה נכונה.

במהלך אימוני המודל בשיטת Oversampling נתקלנו בOverfitting היות ושכפלנו את התמונות בצורה שבה הייתה קיימת חפיפה.

לאחר טבילת האש בנושא הנ"ל וניתוח תוצאות האימון, לדעתנו, חשוב לבחור את שיטת הפתרון בהתאם לבעיה ולא בהכרח שפתרון מתוחכם יותר ייתן תוצאות טובות יותר.