

תוכן העניינים

מבוא	3
מבנה	4
שלב איסוף הכמה וניתוח הנתונים :	4
שלב בניה ואמון המודל :	9
שלב היישום :	14
מדריך למשתמש	16
17.	
18.	
18.	
19.	
ביבליוגרפיה	20
נספחים	21
נספח 1 : קוד מלא - בניית מאגר נתונים :	21
נספח 2 : קוד מלא - אימון המודל :	24

מבוא

זיהוי חיידקים הינו דבר בסיסי שנעשה מדי יום במעבדות רפואיות על מנת לזהות חיידקים אשר גורמים למחלות במטופלים ועל מנת להציג דרך טיפול.

זיהוי חיידקים הוא תהליך מייגע אשר דורש עין מiomנת להבדלה בין סוגי החיידקים הרבים שיש. חיידקים לרוב הם חסרי צבע ולכך לרובם קשה להבחין בהם על מצע הגידול. אי לכך, בעולם המדע ישנו שיטות רבות לצביעת חיידקים אשר מאפשר זיהוי קל יותר.

אך למרות השיטות הרבות הנמצאות במעבדות רפואיות להקלת על הבדיקה החידקים, בסופה של דבר נדרש זיהוי סופי ע"י עובד מעבדה מיומן.

דרישת הזיהוי הסופי מעובד מעבדה הינה עבודה קשה אשר דורשת זמן ומיומנות, עבודה עם חיידקים הינה עבודה קритית ודוחפה.

באופן כללי גדיילת חיידקים הינה גדיילה מעריצית, כלומר חיידקים מתרבים בצורה מאוד מהירה. מה שדורש מן עובדי המעבדה להביא תוצאות זיהוי בהקדם האפשרי להתחלה הטיפול במטופל.

אי לכך יש צורך במכונה שתבצע את החלק הטכני של הזיהוי הסופי ביעילות רבה יותר מאשר עובדי מעבדה. בגידול חיידקים על מצע מזון מוצק (אגר) בצלחות פטרி, ניתן להבחין בתכונות שונות של חיידקים ובכך ליהותם. תכונות הנראות לעין הינם למשל צבע, גודל, צורה, ציפויות ועוד ...

כפי שלמדנו,

בפרויקט שלי אטמוך בבנייה מערכת לומדת אשר תלמד ממאגר תמונות של חיידקים בצלחות פטרי (DIBaS), ממאגר זה בחרתי 12 סוגי חיידקים אשר הכיל שכיחים ברפואה. יש צורך בעיבוד התמונות מהמאגר המקורי לשם יצירת כמות דוגמאות גדולה יותר.

התמונות הנמצאות במאגר הנתונים המקורי הינו תמונות גדולות ולא בכמות מספקת למידת מכונה, לשם כך חילקתי את התמונות לחתתי-תמונות בגודל 224x224. לאחר חילוק התמונות התקבל מאגר נתונים בגודל מספק של כ- 12 אלף תמונות.

אי לכך שכל חיידק מתרבה בצורה אחרת, ישנו חיידקים אשר מתרבים בציפויות מה שיוצר תמונות ריקות במאגר הנתונים חדש שלנו, זהו אתגר שחוויותי במהלך הפרויקט מאחר ולא הצלחתי بكلות למצוא את התמונות הריקות באלפי התמונות שנמצאות המאגר.

מבנה

שלב איסוף הנקה וניתוח הנתונים :

מאמגר הנתונים שהשתמשתי בו בפרויקט שלי הינו עיבוד של מאגר נתונים בשם DIBaS אשר מכיל 33 סוגים חיידקים וכ- 20 תמונות לכל חיידק. כל הדגימות נקבעו בצביעת גראם (שיטת לצביעת חיידקים להבדלה ראשונית).

מאמגר התמונות הנ"ל נאסף ע"י הפולטה ל�יקרוביולוגיה של האוניברסיטה היגלונית קראקוב, פולין.

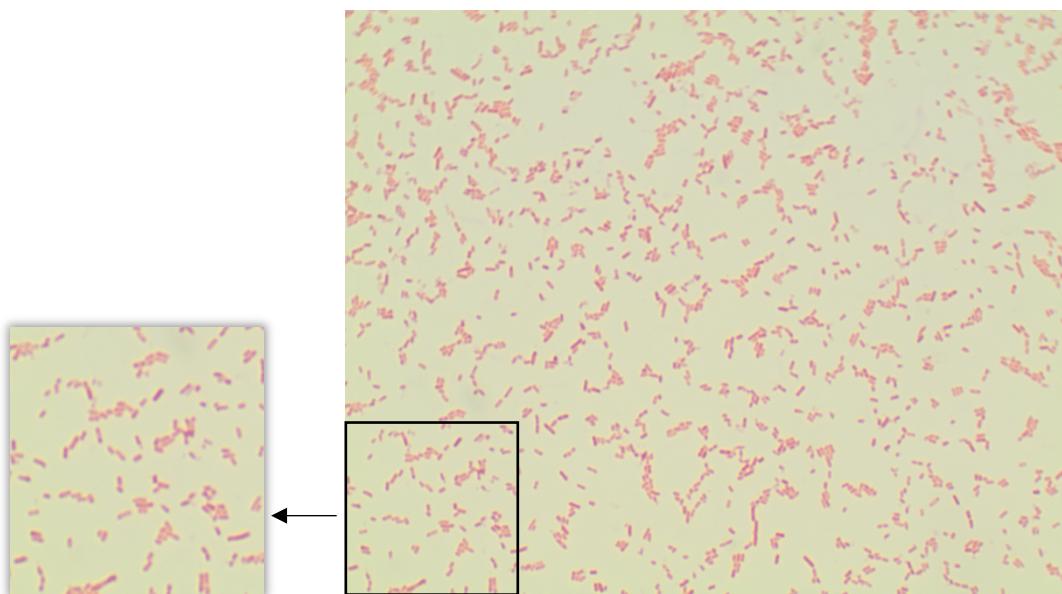
בפרויקט שלי בחרתי רק 12 סוגים חיידקים אשר יותר שכחחים בעולם הרפואה ביניהם :

אצינטובקטר, אקטינומיצס, בקטרוידס פראג'יליס, קנדידה, קלוסטרידיום, אשרכיה קולי, אנטרואוקוקוס, ניסריה, פסאודומונס, ליסטריה, סטפילוקוקוס, פרוטואס.

הורדתי את התמונות והתחלתי בעיבודן בשביל שיתאים לצרכי הפרויקט. התמונות במאגר גדולות 224x224 ולבסוף 2048x1532 ולכן, החלטתי לחלק אותן לתמונות בגודל 224x224.

לדוגמא :

תמונה של החיידק "אי-קוליה" מהמאגר המקורי ולצדו תתמונה שתהווה חלק ממאגר הנתונים בפרויקט.



הכני תיקיות ריקות מותוגות לפי שמות החידקים אליהם העברתי את תת הtemperature שיצרת בעזרת פונקציה שמבצעת את החלוקה :

```
#input: filename - file name (str), dir_in - image path (str), dir_out - tiles saving dir (str), d - size of tile (int)1
def tile(filename, dir_in, dir_out, d):
    name, ext = os.path.splitext(filename)
    img = Image.open(os.path.join(dir_in, filename))
    w, h = img.size
    grid = product(range(0, h-h%d, d), range(0, w-w%d, d))
    for i, j in grid:
        box = (j, i, j+d, i+d)
        out = os.path.join(dir_out, f'{name}_{i}_{j}{ext}')
        img.crop(box).save(out)
```

הפונקציה tile מחלקת את התמונה לפי הגודל שהוא מקבל כפרמטר (d) ושמירת את תת הtemperature בתיקייה .(dir_out)

כתבתי פונקציית עזר file_inf אשר מקבלת תיקייה ומחזירה רשימה המכילה את שמות התמונות הנמצאות בתיקייה :

```
#input: folder (path - str) ; output: filename - images' names (list)
def file_inf(folder):
    filename = []
    for dir in os.listdir(folder):
        filename.append(dir)
    return filename
```

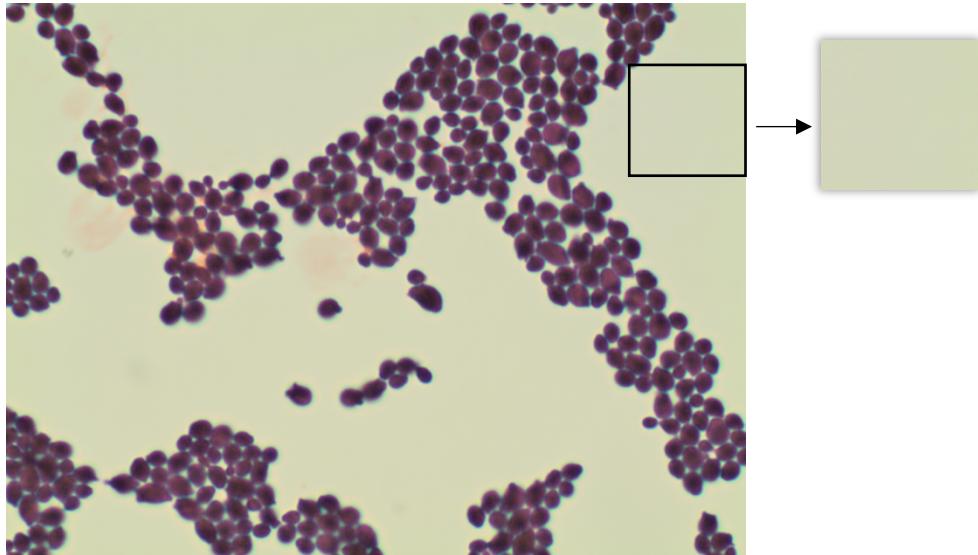
לאחר מכן, עברתי בלולאה על כל התמונות במאגר הנתונים שהורדתי וייצרת מאגר נתונים חדש.

```
#slicing images and saving as a new file using tile and file_inf functions
classes=['Acinetobacter.baumanii','Actinomyces','Bacteroides','Candida','Clostridium','Enterococcus','Escherichia','Listeria','Neisseria','Proteus','Pseudomonas','Staphylococcus']
folder = "/content/drive/MyDrive/ML_project/DIBaSdataset"
folder_out = "/content/drive/MyDrive/ML_project/DIBaS_sliced"
for c in classes:
    path = os.path.join(folder,c)
    filename = file_inf(path)
    for f in filename:
        tile(f, path,os.path.join(folder_out,c),224)
```

¹ הקוד המלא [בנספח 1](#)

מחלוקת התמונות לנתבי תמונות התקבלו נתבי תמונות ריקות, שלא עוזרות ואף מזיקות למדידת המכונה בזיהוי החידקים.

לדוגמא: תמונה מקורית מן המאגר של החידק "קנדידה" ולצדו נתת תמונה ריקה אפשרית כתוצאה מן החלוקה.



אי לכך, היה צורך במחיקה של התמונות הריקות שהתקבלו ממאגר הנתונים החדש.
מחיקה ידנית לא הייתה פרקטית מכיוון שיש במאגר כ 12 אלף תמונות, אך החלטתי למצוא את התמונות הריקות בעזרת ממוצע RGB של התמונות.

לאחר ניסיונות רבים, הדרך הטוב ביותר למציאת תמונות ריקות עם פחות טעויות הייתה לעבור על כל קטגוריה (חידק) בנפרד ולמחק תמונות בהתאם לתנאי RGB שונים. זאת מכיוון שככל חידק נקבע באופן שונה וחלקם יותר בהרים מאחרים.

לדוגמא:
חידק "בקטרואידס פראג'יליס" - תנאי למחיקת תמונה (ע"פ ממוצע RGB של תמונה אחת) :

```
if(r>220 and g>219 and b>187)...
```

לעומת זאת, לחידק "קנדידה" –

```
if(r>205 and g>206 and b>177)...
```

לשם יישום הדריך הזה כתבתני פונקציה אשר מחזירה את ממוצע ה - RGB של התמונה הנתונה כפרמטר.

```
def avg_rgb(filename):
    P = 50176 #num of pixels
    rgb = [0,0,0]
    img = Image.open(filename)
    for i in range(224):
        for j in range(224):
            r,g,b = img.getpixel((i,j))
            rgb[0] += r
            rgb[1] += g
            rgb[2] += b
        rgb[0] = int(rgb[0]/P)
        rgb[1] = int(rgb[1]/P)
        rgb[2] = int(rgb[2]/P)

    return tuple(rgb)
```

ולאחר מכן, כתבתני פונקציה אשר מקבלת תיקייה ומחזירה רשימה של תמונות ריקות בתיקייה הנתונה.
בהתאם לתנאי ה RGB כפי שציינתי לעיל.

```
def blank(folder):
    blank_files = []
    filename = file_inf(folder)
    for f in filename:
        r,g,b = avg_rgb(os.path.join(folder,f))
        if(r>210 and g>212 and b>188):
            blank_files.append(f)
    return blank_files
```

לבסוף, העברתי את התמונות הריקות מתיקייה המוצא אל תיקייה בה ימצאו כל התמונות הריקות בצד
לבדוק שלא היו טעויות בזיהוי התמונות הריקות.

```
folder = "/content/drive/MyDrive/ML_project/DIBaS_sliced/Staphylococcus"
blank_folder = '/content/drive/MyDrive/ML_project/empty'
files = blank(folder)
for f in files:
    shutil.move(os.path.join(folder,f),blank_folder)
```

לשם העברת הקבצים מתיקייה אחת לאחרת, השתמשתי בספרייה shutil אשר מכילה פונקציות לביצוע
פעולות על קבצים. מן הספרייה השתמשתי בפונקציה move.

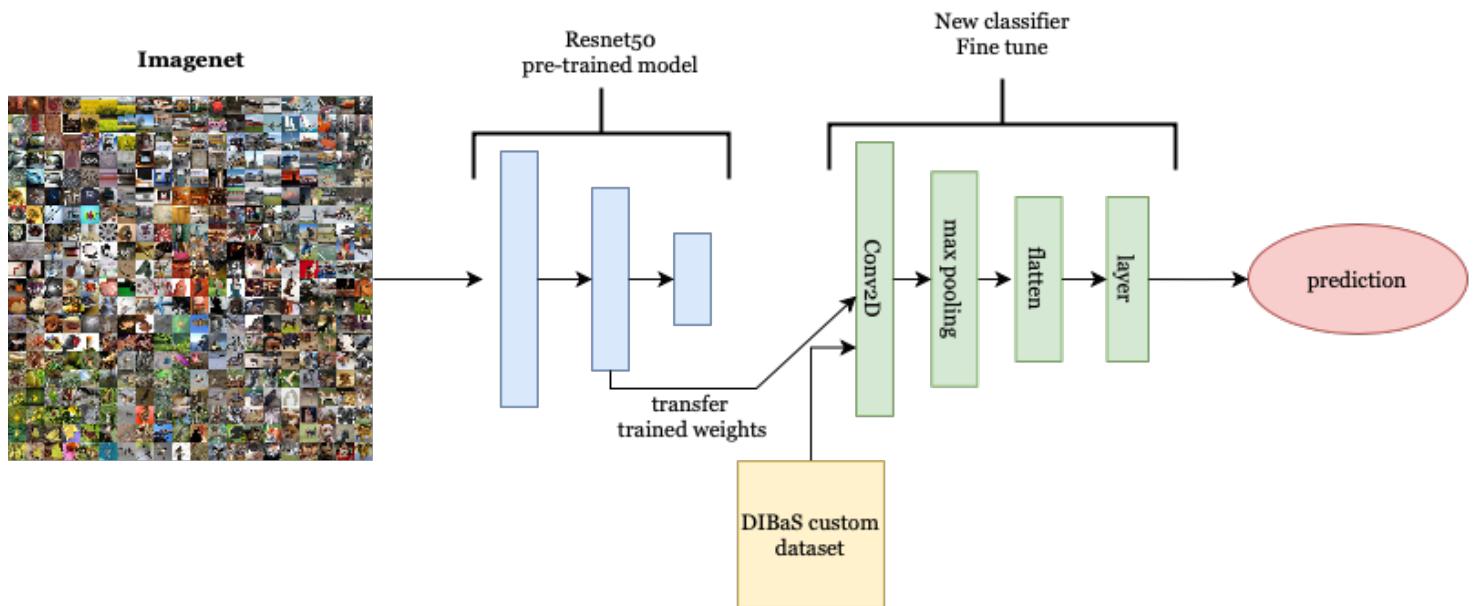
לאחר שבניתי מאגר נתונים חדש, העלייתי אותו לגול קולאָב לשם הכנסתו לשלבים הבאים.
בכדי לשחרר את הקבצים מהקובץ zip אל גול קולאָב השתמשתי בקטע קוד הבא :

```
from zipfile import ZipFile  
%cd '/content'  
with ZipFile('/content/drive/MyDrive/ML_project/DIBaS_sliced2.zip','r') as zip:  
    zip.extractall()
```

ויצרתי שני NumPy array - X ו- y כאשר X מכיל את התמונות בטור רשימות של פיקסלים בגודל 3x224x224 . התמונות הן צבעוניות מכיוון שצבע הינו מרכיב חשוב בזיהוי החידק ולכן יש לכל תמונה 3 ממדים (RGB).

y מכיל את התיאוג של כל תמונה בהתאם, במספרים מ 0-11 .
חילקתי את X ו y לדוגמאות אימון ודוגמאות מבחן. כך ש70 אחוז מן המאגר נתונים ישמש לאימון המודל
ול-30 אחוז ישמש לבדיקת המודל.
לאחר מכן, המרתתי את ערכי הרשימות y_train, y_test לערכים בינאריים זאת מכיוון שבמודל פונקציית
ההפסד תהיה .categorical cross entropy

שלב בנית ואמון המודל :



בפרויקט שלי השתמשתי בהעברת למידה (transfer learning).

אימון רשת נוירונים عمוקה הינו תהליך ארוך מאוד שיכול לקחת ימים ואף יותר.

בעצם העברת למידה נותנת לנו את האפשרות לחתך משלבים של מודל קיימים ומאמנים על מאגר נתונים גדול כלשהו, אשר הגיע לתוצאות אופטימליות. ככלומר העברת למידה מאפשרת לנו כביכול לדלג על שלב האימון הארוך ולספק לנו ידע רב כך שנצטרכ רק לבניית השכבות הסופיות במודול וכונונו עדין לצרכיינו. בכך תהליכי הלמידה מתקצר משמעותית.

בהעברת למידה שאני עשית השתמשתי ברשת נוירונים عمוקה בשם "resnet50" אשר התאימה על מאגר הנתונים "imagenet".

מודלי resnet לMINIIMH הם בעצם רשתות عمוקות המכילות שכבות קונבולוציה רבות ובין השאר מכילות בלוקים שיוריים. תפקיים של בלוקים אלו הוא לבצע פונקציית זהות על האקטיבציה של השכבה הרדודה (הראשונית), אשר בתורה תיציר את אותה אקטיבציה. לאחר מכן, הפלט מוגוסף לאקטיבציה של השכבה הבאה.

מכילה הרבה מודלים מאומנים שנייתן לחתוך, ממנה לחתוך את המשקלים של הרשת resnet50 תוך הקפתה השכבות של הרשת המקורית ומחיקת השכבות הסופיות (שכבות הפלט) מכיוון שברצוני לסוג דברים שונים מן המאגר נתונים שיצרתתי.

בשלב זהה מתחילה המודול החדש שאני כתבתי לשם התאמה ולמידה על מאגר הנתונים שלי. המודול מורכב מ:

- שכבת קונבולוציה עם פונקציית אקטיבציה "relu".
- שכבת MaxPooling אשר מקטינה את גודל הקלט בכך שהיא לוקחת את הערכים המקסימליים, פעולה זו מותירה מאפיינים יותר דומיננטיים שמקילים יותר אינפורמציה מהשאר.

- שכבה flatten אשר ממיר את הקלט ההתלת ממדי שיש לנו לקלט בעל ממד אחד. שכבה זו הכרחית למעבר משכבות קונבולוציה לשכבות dense.
- שכבה Dropout אשר מבצעת רגולרציה באמצעות בחירת נוירונים רנדומליים שלא ישתתפו בתהילך, בהתאם לkeepprob שהגדכנו. שכבת הדROUPאות מסייעת במניעת אובר-פייטינג.
- שכבה BatchNormalization אשר עוברת ומבצעת נרמול כל פעם על קבוצה שונה של דוגמאותaimon. להבדיל משכבת נורמליזציה אשר מבצעת נרמול לכל דוגמאות האימון בבת אחת.
- שכבת Dense שהיא שכבת נוירונים פשוטה. בהגדרת ה- Dense נבחר את מספר הנוירונים הרצויים בשכבה ואת פונקציית האקטיבציה.
- שכבת ~~Dense~~ הינה השכבה האخונה אשר פולטת את הפרדיקציה. שכבת זו הינה שכבת Dense עם נוירונים כמספר המחלקות שיש לנו והוא גם כן, משתמשת בפונקציית אקטיבציה לבחירתנו. במספר סיוגים גדול מ-2 נשתמש לרוב בפונקציית האקטיבציה softmax, אשר נותנת הסתברות סיוג לכל מחלוקת בנפרד.

Hyper Parameters:

epochs	Batch size	Optimizer	Dropout 1	Dropout 2	Dense 1	Activation function	Dense 2	Activation function
50	32	Adam	0.2	0.4	32	relu	12	softmax

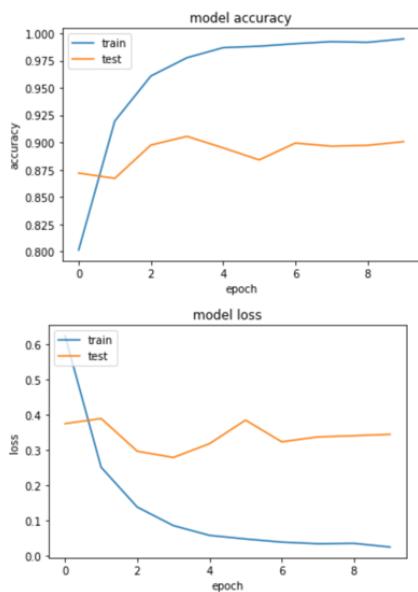
המודל הראשון:

מבנה:

```
2model = keras.models.Sequential()
model.add(resnet_model)
model.add(Conv2D(32,(3,3),activation='relu',kernel_initializer='he_uniform',input_shape=(224,224,3)))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(layers.BatchNormalization())
model.add(Dense(12, activation='softmax'))
```

batch = 32
epoch = 10

ביצועים:



דיקן המודל וההפסד מאד מוצלחים על התמונות שהוא התראמן
עליהן, אך בבדיקה על תמונות אחרות ביצועי המודל יורדים
משמעותית – זאת אי להתאמת יתר של המודל.
כפי שניתן לראות במודל הנ"ל כמעט ולא הוספה שכבות
נורמליזציה אשר מפחיתה את הסיכויים להתאמת יתר.

² הקוד המלא [בנספח 2](#)

מודל שני:

מבנה:

```
model = keras.models.Sequential()
model.add(resnet_model)
model.add(Conv2D(32,(3,3),activation='relu',kernel_initializer='he_uniform',input_shape=(224,224,3)))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dropout(0.2))
model.add(layers.BatchNormalization())
model.add(Dropout(0.4))
model.add(layers.BatchNormalization())
model.add(Dense(32, activation='relu'))
model.add(Dense(12, activation='softmax'))
```

```
early_stopping =
keras.callbacks.EarlyStopping(monitor='val_accuracy',patience=20,restore_best_weights=True)
```

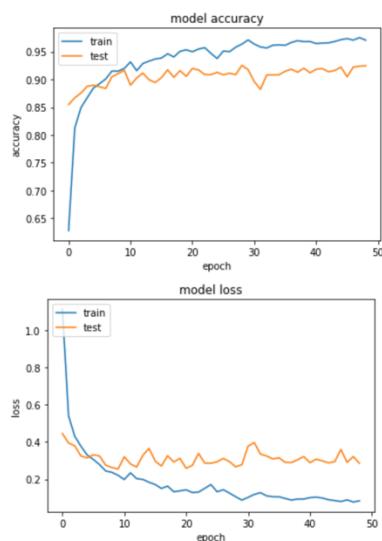
הוספתי שכבות דרופ-אут ושכבות רגולציה נוספת לשם טיפול בהתאמת יתר שראינו במודל הראשון.

בנוסף, הוספתי שכבת `dense` רגילה עם 32 נוירונים ופונקציית אקטיבציה – `relu`.
כמו כן, הוספתי שכבת `callback` `EarlyStopping`. זהה פונקציית `callback` בסוף כל epoch ובuzzרטו אנו יכולים להגדיר למערכת מתי לעצור את האימון.

למשל, במודל שלי הוספתי `earlystopping` אשר אחרי 20 epochs שבhem לא נעשה שיפור האימון מעכז. יתרה מכך הגדרתי כי בשלב העצירה נחזיר לערכי הפרמטרים הטובים ביותר שנמצאו.

בעזרת הפונקציה הזו ניתן להגדיר מספר epoch-ים יותר גדול וייתכן שלא יהיה צורך בכלם.

batch = 32



epoch = 50

ביצועים:

במודל הנ"ל נוכל לראות שיפור ממשמעותי בדיקוק ובהפסד בשלב המבחן על דוגמאות שהמודל לא ראה. ככלمر הטיפול בהתאמות יתר שראינו קודם.

המודל הגיע לביצועים טובים של 91% דיקוק.

פונקציית ההפסד:

פונקציית ההפסד הינה פונקציה אשר מוגנת לנו אינדיקטיה לגבי ההצלחה של המודל בחיזויים. באימון רשותנו נוירונים עמוקה נרצה להתמקד במצוור פונקציית ההפסד. ערך פונקציית הפסד קטן יותר מעיד על פחות שגיאות ככלמר, ביצועים יותר טובים.

במודל שלי השתמשתי בפונקציית ההפסד " categorical cross entropy " המשומשת בעיקר ברשומות רבות סיווגים.

פונקציית ההפסד הזה הינה בעצם שילוב של פונקציית אקטיבציה (לרוב "softmax" עם cross-entropy loss function:

$$Loss = - \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

פונקציית ההפסד הזה מוחזירה לנו קלסיפיקציה בצורת הסתברות בין 0 ל-1 לכל הקטגוריות (הסיווג עם ההסתברות הגבוהה ביותר הינו הפרדייקציה).

יעול התוכניות – אופטימיזציה:

תהליך למידת המכונה הינו ניסיון במצומס פונקציית העלות בעזרת אלגוריתם הנקרא gradient descent. תפקיד האלגוריתם להגעה לשגיאה המינימלית, באימון מכונה נחזר על התהליך מס' פעמים עד שמנגעים לנקודה בה לא נראה שיפור (נקודת ההתקנסות).

הכוון והקצב בהם הפונקציה נעה צריכה להיות מתואימים בכך להימנע מזמן למידה ארוך מדי או פספוס נקודת המינימום (over-shooting), כאן מגיע תפקידם של פונקציות האופטימיזציה אשר מתאימים את גודל הצעד מהתול השיפוע, כלומר הקטנת הצעדים (קצב הלמידה) כשהמודול תלול ואילו הגדלתם כשהמודול רדוד.

פונקציית האופטימיזציה במודל שלי הינה פונקציית ADAM אשר משלבת שני אלגוריתמים : **Gradient Descent With Momentum - GDwM** : עדכון הפרמטרים ע"פ המוצע הנע של הגרדיאנטים. המוצע הנע מאזנות את הקפיצות האקראיות של הגרדיינט דסנט על מנת לקבל קו התקדמות אחד יותר . **Root-Mean Square Propogation - RMSProp** : חילוק הגרדיאנטים במוצע הנע שלהם כדי לבטל קפיצות גדולות בכיוון מסוים.

שלב היישום :

בכדי לישם את המודל בחרתי לבנות אפליקציה ב – Android Studio, ועזרתי ב Tensor flow אשר מאפשרת לחת מודל מאומן, להעלות אותו ולהשתמש בו במכשירים שונים. לשם כך, לאחר אימון המודל בקורס המרתתי את המודל למודל מסוג ” שמרתתי את המודול והורדתי למחשב.

```
TF_LITE_MODEL_FILE_NAME = "tf_lite_model.tflite"
model_name = TF_LITE_MODEL_FILE_NAME
open(model_name,"wb").write(fmmodel)
```

בניתי אפליקציה בסיסית בה ניתן להעלות תמונה מסוימת קבצי התמונות במכשיר, לבצע חיזוי על התמונה שהועלתה ולקבל סיווג ע"פ המודול.

ב – Android Studio ניתן להעלות את קובץ ה – ”tensor flow lite model” – “predict” ולהשתמש בו . קוד הפעלת הלחצן ”predict” בעת לחיצה מן המשתמש וקבלת החיזוי (שפה תכנות – java) :
לאחר קבלת לחיצה על הלחצן, התמונה שהועלתה מומרת ל TensorImage בגודל המתאים לקליטה ע"י המודול .

```
TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int[]{1,224,224,3}, DataType.FLOAT32);
TensorImage tensorImage = new TensorImage(DataType.FLOAT32);
```

תוצאות הסיווג מן המודול נקלטות במשתנה inputFeature0

```
TfLiteModel.Outputs outputs = model.process(inputFeature0);  
TensorBuffer outputFeature0 = outputs.getOutputFeature0AsTensorBuffer();
```

וכפי שכבר הוסבר לעיל, פונקציית הפלט של המודל הינה softmax אשר מוחזירה הסתברות חיזוי לכל סיווג.
לכן, מנו הפלט נצורך לחת את האינדקס בעל ההסתברות הגבוהה ביותר בעזרת פונקציית עוזר

```
.getMax(float[] arr)
```

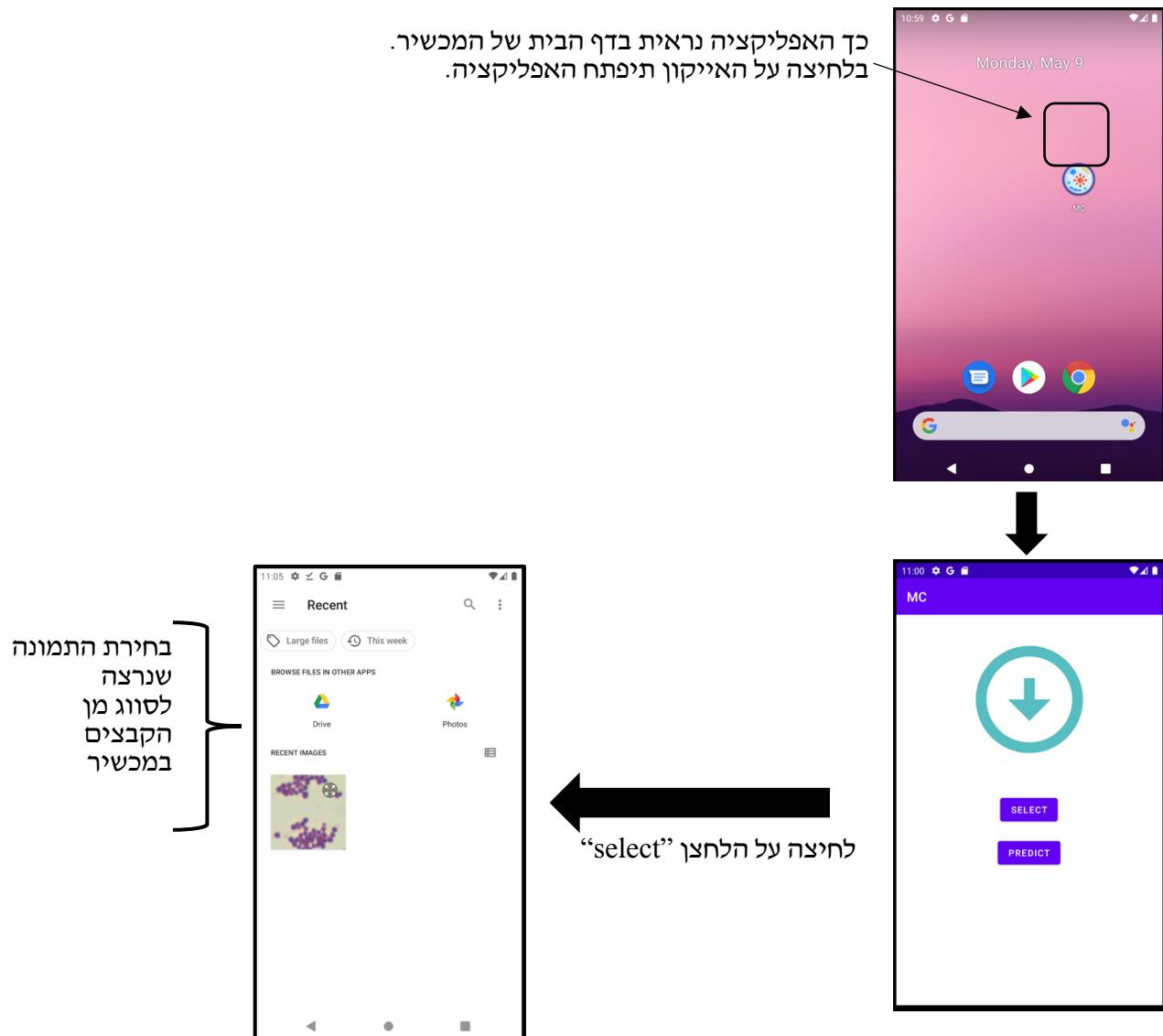
```
int max = getMax(outputFeature0.getFloatArray());
```

לאחר קבלת אינדקס ההסתברות הגבוה ביותר מן רשימת הפלט של המודל נדייס למשתמש את הסיווג.

```
String[] classes =  
{"Acinetobacter.baumanii","Actinomyces","Bacteroides","Candida","Clostridium","Enterococcus","Escherichia","Listeria","Neisseria","Proteus","Pseudomonas","Staphylococcus"};  
  
txt.setText("Prediction: "+classes[max]);
```

מדריך למשתמש

לקבלת חיזויים בעזרת האפליקציה כל מה שנctrיך הוא הלאת תמונה בלחיצה על הלחץן “select”, בחירת התמונה הרצויה ולחיצה על הלחץן “predict” “תדפיס את החיזוי עבור התמונה שתועלה מממשק המשתמש. להלן דוגמא :





*התמונה שנבחרה
mozgat ul hamsak

*נקבל חיזוי להיחדך

שבתמונה בתחתית המסך

רפלקציה

בשבילי העבודה על הפרויקט הייתה חוויה בעלת המון כישלונות וכמו כן, הצלחות. במהלך העשייה למדתי איך להתמודד עם קשיים לבד ולמצוא פתרונות לביעות שעולות בדרכי, לכל בעיה הצלחתי למצוא פתרון גם אםלקח לי הרבה זמן וזה ייאש אותי.

הפרויקט היה אחידות גדולה ופרויקט אישי מה שגרם לי לא לוותר עליו גם כשלא הצלחתי והיה קשה. מכל ההתמודדות במהלך הפרויקט קיבלתי מגוון כלים עצום שאני מאמין שיעזרו לי בהמשך דרכי למשל, איך לחפש פתרונות בצורה יעילה... כיצד ניתן לעקוף מכשולים שעולים בדרך... וכמובן שהכלי הכי חשוב הוא הבסיס לעולם הנרחב והעמוק שנקרו "בינה מלאכותית".

כפי שציינתי בתחילת הרפלקציה, במהלך עשיית הפרויקט היה מיגע ורויי בקשאים אשר עם כולם נאלצתי להתמודד בכך המשיך בדרכי. אחד הקשיים הכי גדולים אשר גזל את רוב הזמן העבודה על הפרויקט היה מאגר הנתונים.

נדרש ממני לחפש מאגר נתונים מתאים לרעיון שלי, לבנות מהתמודנות המעודת מאגר נתונים של אלפי תמונות, מציאת תמונות ריקות בתוך אלפי התמונה במאגר (לשם כך, נאלצתי לחשב וליחסם הרבה פתרונות עד שהגעתי להצלחה)...

לאחר שהגעתי לשלב זהה, השלב בו כל הפרויקט מוכן ועובד, הקשיים מאחוריו... אני מבינה כי כל עוד לא מוגדרים ויש מספיק רצון להתקדם ולעשות יש ביכולתנו לעשות כל מה שנרצה.

כמו כן, ראייתי את כל חברי עובדים על הפרויקטים השונים והבנתי שהצלחה היא לא רק ההגעה למה שרצית בשלב הначала. ככלمر אף פעם לא מאוחר לשנות, להתחרט, להביא רעיונות שונים וליצור מהם הצלחה.

אני חושבת שבמידה והיה ניתן יותר זמן ללמידה את החומר וכמו כן, יותר זמן לעשיית הפרויקט, העבודה הייתה יותר טובה, מושקעת ויעילה...
לסיכום, העבודה הייתה מלמדת, מהנה, מأتגרת ואני מרוצה מהתוצר הסופי.

ביבליוגרפיה

Loss function. (2021, April 18). Wikipedia. Retrieved from
https://en.wikipedia.org/wiki/Loss_function

tf.keras.losses.CategoricalCrossentropy / TensorFlow Core v2.7.0. (n.d.). TensorFlow. Retrieved from https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy

Steven. (2019, August 24). *Microbe classification using Deep Learning*. Medium. Retrieved from <https://towardsdatascience.com/microbe-classification-using-deep-learning-e84312046334>

יעקובסון, ר. (יולי, 2019). הסבר למתחללים על העברת למידה - Transfer Learning . בלוג בינה מלאכותית. <https://www.ai-blog.co.il/2019/07/17/transfer-learning-למתחללים-על-העברת-למידה/>

נספחים

נספח 1 : קוד מלא - בניית מאגר נתונים:

“””DIBaS_dataset

Automatically generated by Colaboratory.

Original file is located at

<https://colab.research.google.com/drive/1eFhMkaVOL0VfZiFnIDP5xb0fviZuh-oq>

```
# Imports
“””
```

```
import os           #interacting with the operating system library
from PIL import Image    #Python Imaging Library
from itertools import product  #Mathematics Cartesian Product
import shutil        #high-level operation on a file
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
“””# data preparation functions
```

file_inf(folder) is a function that returns the name of the files in the given folder
“””

```
#input: folder (path – str) ; output: filename – images' names (list)
def file_inf(folder):
    filename = []
    for dir in os.listdir(folder):
        filename.append(dir)
    return filename
```

#input: filename – file name (str), dir_in – image path (str), dir_out – tiles saving dir (str), d – size of tile (int)

```
def tile(filename, dir_in, dir_out, d):
    name, ext = os.path.splitext(filename)
    img = Image.open(os.path.join(dir_in, filename))
    w, h = img.size
```

```
grid = product(range(0, h-h%d, d), range(0, w-w%d, d))
for i, j in grid:
    box = (j, i, j+d, i+d)
    out = os.path.join(dir_out, f'{name}_{i}_{j}{ext}')
    img.crop(box).save(out)
```

```
# Commented out Ipython magic to ensure Python compatibility.
# %cd "/content/drive/MyDrive/ML_project/DIBaSdataset"
```

```
“””# DIBaS dataset slicing“””
```

#slicing images and saving as a new file using tile and file_inf functions

```

classes =
['Acinetobacter.baumanii', 'Actinomyces', 'Bacteroides', 'Candida', 'Clostridium', 'Enterococcus',
'Escherichia', 'Listeria', 'Neisseria', 'Proteus', 'Pseudomonas', 'Staphylococcus']
folder = "/content/drive/MyDrive/ML_project/DIBaSdataset"
folder_out = "/content/drive/MyDrive/ML_project/DIBaS_sliced"
for c in classes:
    path = os.path.join(folder,c)
    filename = file_inf(path)
    for f in filename:
        tile(f, path,os.path.join(folder_out,c),224)

#check images countaty
images = []
folder = "/content/drive/MyDrive/ML_project/DIBaS_sliced"
for c in classes:
    path = os.path.join(folder,c)
    print(len(os.listdir(path)))

"""# DIBaS sliced dataset empty images"""

def avg_rgb(filename):
    P = 50176 #num of pixels
    rgb = [0,0,0]
    img = Image.open(filename)
    for i in range(224):
        for j in range(224):
            r,g,b = img.getpixel((i,j))
            rgb[0]+=r
            rgb[1]+=g
            rgb[2]+=b
    rgb[0]=int(rgb[0]/P)
    rgb[1]=int(rgb[1]/P)
    rgb[2]=int(rgb[2]/P)

    return tuple(rgb)

print(avg_rgb('/content/drive/MyDrive/ML_project/DIBaS_sliced/Staphylococcus/Staphylococcus.aureus_0005_0_1120.tif'))
print(avg_rgb('/content/drive/MyDrive/ML_project/DIBaS_sliced/Staphylococcus/Staphylococcus.aureus_0005_672_896.tif'))

def blank(folder):
    blank_files = []
    filename = file_inf(folder)
    for f in filename:
        r,g,b = avg_rgb(os.path.join(folder,f))
        if(r>210 and g>212 and b>188):
            blank_files.append(f)
    return blank_files

def convert(path,classes):
    for c in classes:
        folder = os.path.join(path,c)
        filename = file_inf(folder)

```

```

for f in filename:
    file = os.path.join(folder,f)
    filename=file.split(".")
    img = Image.open(file)
    target_name = filename[0] + ".jpg"
    rgb_image = img.convert('RGB')
    rgb_image.save(target_name)

convert('/content/drive/MyDrive/ML_project/DIBaS_sliced',classes)

def countPix(filename):
    blank_files = []
    count = 0
    img = Image.open(filename)
    for i in range(224):
        for j in range(224):
            r,g,b = img.getpixel((i,j))

folder = "/content/drive/MyDrive/ML_project/DIBaS_sliced/Staphylococcus"
blank_folder = '/content/drive/MyDrive/ML_project/empty'
files = blank(folder)
for f in files:
    shutil.move(os.path.join(folder,f),blank_folder)

"""**rgb condition by class: **
Acinetobacter.baumanii: r>215 and g>213 and b>185
Actinomyces: r>208 and g>210 and b>185
Bacteroides: r>220 and g>219 and b>187
Candida: r>205 and g>206 and b>177
Clostridium: r>203 and g>203 and b>180
Enterococcus: r>209 and g>210 and b>183
Escherichia: r>217 and g>219 and b>187
Listeria: r>217 and g>219 and b>187
Neisseria: r>213 and g>215 and b>180
Proteus: r>200 and g>200 and b>178
Pseudomonas: r>212 and g>215 and b>185
Staphylococcus: r>210 and g>212 and b>188
"""

```

נספח 2: קוד מלא - אימון המודל:

"""/Microbe_Classification.ipynb

Automatically generated by Colaboratory.

Original file is located at

<https://colab.research.google.com/drive/1h4enLRD81bNlvvcıwsGuZbFR-DkFf4bQ>

```
# Imports
"""

# Commented out IPython magic to ensure Python compatibility.
import numpy as np
import os
import tensorflow as tf
from PIL import Image
from tensorflow import keras
from keras.models import save_model,load_model
from keras import layers
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from keras.layers import Dense, Flatten, Activation, Dropout, Conv2D, MaxPooling2D

# %matplotlib inline

"""# Drive mount"""

from google.colab import drive
drive.mount('/content/drive')

"""# Data loading

Extracting dataset (zip file) into google colab.

"""

# Commented out IPython magic to ensure Python compatibility.
from zipfile import ZipFile
# %cd '/content'
with ZipFile('/content/drive/MyDrive/ML_project/DIBaS_sliced2.zip','r') as zip:
    zip.extractall()

"""Converting the images in the dataset to numpy array.
x - images as numpy array
y - images labels
labels - python dictionary - {labels: class...}
The labels are 0-11 integers matching the classes.

"""

data_dir = "/content/DIBaS_sliced"
```

```

classes =
['Acinetobacter.baumanii','Actinomyces','Bacteroides','Candida','Clostridium','Enterococcus','Es
cherichia','Listeria','Neisseria','Proteus','Pseudomonas','Staphylococcus']
labels = { }
y = []
x = []
i=0
for c in classes:
    path = os.path.join(data_dir,c)
    for dir in os.listdir(path):
        if(dir!='.DS_Store'):
            y.append(i)
            img = Image.open(os.path.join(path,dir))
            x.append(np.asarray(img))
    labels[i] = c
    i+=1
x = np.asarray(x)
y = np.asarray(y)

""Spliting x and y into train(70%) data and test(30%) data.
Using sklearn.model_selection.train_test_split function.
"""

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3,train_size=0.7)

"""Plot a sample of 10 randomly selected images from train dataset"""

indexes = np.random.randint(0, x_train.shape[0], size=10)
images = x_train[indexes]
lables = y_train[indexes]

# plot 10 images
plt.figure(figsize=(5,5))
for i in range(len(indexes)):
    plt.subplot(5, 5, i + 1)
    image = images[i]
    plt.imshow(image, cmap='gray')
    plt.axis('off')

plt.show()
plt.close('all')

print(lables)

"""# Data Preparation """

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train)
y_test = keras.utils.to_categorical(y_test)

"""# Transfer learning - resnet50

Loading frozen resnet50 model from keras without the final layers
"""

```

```

resnet_model =
keras.applications.ResNet50(include_top=False,weights='imagenet',input_shape=(224,224,3))

resnet_model.trainable=False

""""Final layers to fit our classification"""

model = keras.models.Sequential()
model.add(resnet_model)
model.add(Conv2D(32,(3,3),activation='relu',kernel_initializer='he_uniform',input_shape=(224,224,3)))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dropout(0.2))
model.add(layers.BatchNormalization())
model.add(Dropout(0.4))
model.add(layers.BatchNormalization())
model.add(Dense(32, activation='relu'))
model.add(Dense(12, activation='softmax'))

""""Early stopping callback. stops the training if there's no improve while saving best weights."""

early_stopping =
keras.callbacks.EarlyStopping(monitor='val_accuracy',patience=20,restore_best_weights=True)

""""Model training"""

batch_size = 32
epochs = 50

model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
history = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation_data = (x_test, y_test), callbacks=early_stopping)

""""# Model Evaluation"""

score = model.evaluate(x_test, y_test, verbose=0)
print("%s: %.2f%%" % (model.metrics_names[1], score[1]*100))

#accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# "Loss"
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')

```

```
plt.legend(['train', 'test'], loc='upper left')
plt.show()

"""# Converting to tf lite model"""

model.save("MC")

converter = tf.lite.TFLiteConverter.from_keras_model(model)
tfmodel = converter.convert()

TF_LITE_MODEL_FILE_NAME = "tf_lite_model.tflite"
model_name = TF_LITE_MODEL_FILE_NAME
open(model_name,"wb").write(tfmodel)
```