פרויקט גמר תכנון ותכנות מערכות

שם בית הספר: מקיף י"א, ראשונים

שם העבודה: פרויקט מיון פסולת למחזור

שם התלמיד: שקד ברססט

ת.ז התלמיד: 214997900

שם המנחה: דינה קראוס

שם החלופה: למידת מכונה

תאריך ההגשה: 16.6.2022

תוכן

מבוא

הפרויקט שעבדתי עליו בשנה האחרונה, הוא פרויקט בתחום למידת מכונה.

תחום זה עוסק בניתוח נתונים אשר מלמד את המחשבים לעשות מה שבא באופן טבעי לבני אדם – ללמוד מניסיון. אלגוריתמים של Machine Learning לומדים ישירות מהנתונים ללא הסתמכות על משוואה קבועה מראש כמודל.

האלגוריתמים משתפרים באופן מיטבי בביצועיהם כאשר מספר הדגימות העומדות ללמידה גדל. השימוש בשיטות של Machine Learning בא לידי ביטוי במיוחד כאשר יש צורך לחזות תרחישים מסוימים, או צורך לזהות דפוס התנהגותי מסוים, וכל זה מאוסף נתונים שקיים.

בדומה לפרויקטים רבים, הפרויקט שלי עוסק במיון של תמונות- למחזור ולא למחזור. הכנתי קובץ של כ- 60,000 תמונות. לאחר מכן סיווגתי אותן לקבוצות, לדוגמה, במחזור סיווגתי לקרטונים, בקבוקים, ספרים ועיתונים. מטרת הפרויקט היתה ללמוד את התמונות אשר הכנתי, ובעזרתן לדעת לסווג תמונות אחרות למחזור או לא למחזור. בנוסף, בחרתי בפרויקט זה מכיוון שאיכות הסביבה הוא תחום שמאוד קרוב לליבי, אני חושבת שזהו תחום חשוב מאוד, ושכל אדם צריך להשקיע בו בחיי היום יום שלו למען העתיד ולמען המקום שבו אנו חיים כיום- כדור הארץ.

הרעיון לנושא הפרויקט הגיע בעקבות מאמר שקראתי על משבר הקורונה והקשר שלו לאיכות הסביבה. "זיהום אוויר מגביר את התחלואה מקורונה, אז למה ישראל מתעלמת".

קישור למאמר בביבליוגרפיה.

המאמר מציין כי יש קשר בין איכות הסביבה לתחלואה בקורונה- יותר אנשים נדבקים בקורונה בעקבות מצב איכות הסביבה בעולם, ההולך ומחריף מיום ליום.

מטרת הפרויקט היא לעודד מחזור, ושמירה על איכות הסביבה. ככל שנגיש את אופן פעולת המחזור, אנשים יימשכו יותר וירצו להתחיל לשמור על איכות הסיבבה אף הם. קהל היעד של הפרויקט הוא בעיקר חברות הממיינות פסולת, כמו לדוגמה חברת Greennet, הממיינת פסולת. החברות השונות יוכלו לקבל פסולת ולממיין אותה באמצעות הפרויקט שפיתחתי.

לפי המצב בשוק, חברת VERIDIS למשל משתמשת במכשירים מתקדמים, כמו מכשור אופטי להפרדה של בקבוקים ואריזות. הנושא אינו חדש בשוק, אך לרוב אינו משתמש במערכת מחשב אשר יכולה ללמוד ולעשות זאת בעצמה.

מקורות המידע העיקריים עליהם התבססתי היו מקורות אשר ציינו את אופן תהליך המיון. ראיתי כי בכל מפעל הממיין פסולת, ישנם הרבה עובדים. הפרויקט שלי יוכל לצמצם את כמות העובדים בשוק, ויוכל לייעל את מיון הפסולת.

צפויים לי במהלך הפרויקט אתגרים רבים, וביניהם, הפרויקט שלי ממיין רק ארבעה סוגים של פסולת, ובשוק ישנם יותר מארבעה סוגים. בנוסף, יהיה לי קשה לטפל בתמונות, שכן יש לי מעט.

תיאור גרפי של המודל שעליו בוצע האימון(הרחבה בהמשך)

```
model = Sequential()
model.add(Conv2D(16, (3, 3),
                 activation='relu',
                 kernel_initializer='he_uniform',
                 input_shape=(100, 100, 3)))

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(32, (3, 3),
                 activation='relu',
                 kernel_initializer='he_uniform'))

model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3),
                 activation='relu',
                 kernel_initializer='he_uniform'))

model.add(Conv2D(32, (3, 3),
                 activation='relu',
                 kernel_initializer='he_uniform'))

model.add(MaxPooling2D((2, 2)))
model.add(Flatten())

model.add(Dense(1,
                activation='sigmoid'))

Adam = SGD(lr=0.01, momentum=0.9)
model.compile(optimizer=Adam,
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

תוצאות האימון

```
155/155 [==============================] - 352s 2s/step - loss: 0.5131 - accuracy: 0.7723 - val_loss: 0.3948 -
val_accuracy: 0.8125
Epoch 2/20
155/155 [==============================] - 345s 2s/step - loss: 0.3316 - accuracy: 0.8616 - val_loss: 0.3266 -
val_accuracy: 0.8557
Epoch 3/20
155/155 [==============================] - 336s 2s/step - loss: 0.2951 - accuracy: 0.8795 - val_loss: 0.3149 -
val_accuracy: 0.8628
Epoch 4/20
155/155 [==============================] - 329s 2s/step - loss: 0.2739 - accuracy: 0.8882 - val_loss: 0.3079 -
val_accuracy: 0.8722
Epoch 5/20
155/155 [==============================] - 61974s 402s/step - loss: 0.2570 - accuracy: 0.8970 - val_loss: 0.2530 -
val_accuracy: 0.9008
Epoch 6/20
155/155 [==============================] - 331s 2s/step - loss: 0.2430 - accuracy: 0.9049 - val_loss: 0.2848 -
val_accuracy: 0.8877
Epoch 7/20
155/155 [==============================] - 319s 2s/step - loss: 0.2332 - accuracy: 0.9092 - val_loss: 0.2426 -
val_accuracy: 0.9023
Epoch 8/20
155/155 [==============================] - 320s 2s/step - loss: 0.2279 - accuracy: 0.9110 - val_loss: 0.2394 -
val_accuracy: 0.9080
Epoch 9/20
155/155 [==============================] - 328s 2s/step - loss: 0.2157 - accuracy: 0.9162 - val_loss: 0.2251 -
val_accuracy: 0.9103
Epoch 10/20
155/155 [==============================] - 317s 2s/step - loss: 0.2116 - accuracy: 0.9182 - val_loss: 0.2207 -
val_accuracy: 0.9084
Epoch 11/20
155/155 [==============================] - 318s 2s/step - loss: 0.2038 - accuracy: 0.9209 - val_loss: 0.2096 -
val_accuracy: 0.9152
Epoch 12/20
155/155 [==============================] - 319s 2s/step - loss: 0.1953 - accuracy: 0.9250 - val_loss: 0.2176 -
val_accuracy: 0.9084
Epoch 13/20
155/155 [==============================] - 317s 2s/step - loss: 0.1972 - accuracy: 0.9224 - val_loss: 0.2065 -
val_accuracy: 0.9140
Epoch 14/20
155/155 [==============================] - 318s 2s/step - loss: 0.1886 - accuracy: 0.9275 - val_loss: 0.2321 -
val_accuracy: 0.9085
Epoch 15/20
155/155 [==============================] - 317s 2s/step - loss: 0.1833 - accuracy: 0.9292 - val_loss: 0.1971 -
val_accuracy: 0.9202
Epoch 16/20
155/155 [==============================] - 330s 2s/step - loss: 0.1810 - accuracy: 0.9285 - val_loss: 0.2212 -
val_accuracy: 0.9186
Epoch 17/20
155/155 [==============================] - 328s 2s/step - loss: 0.1707 - accuracy: 0.9347 - val_loss: 0.2064 -
```

```
Epoch 17/20
155/155 [==============================] - 328s 2s/step - loss: 0.1707 - accuracy: 0.9347 - val_loss: 0.2064 -
val_accuracy: 0.9138
Epoch 18/20
155/155 [==============================] - 339s 2s/step - loss: 0.1741 - accuracy: 0.9325 - val_loss: 0.1942 -
val_accuracy: 0.9207
Epoch 19/20
155/155 [==============================] - 341s 2s/step - loss: 0.1644 - accuracy: 0.9383 - val_loss: 0.1929 -
val_accuracy: 0.9179
Epoch 20/20
155/155 [==============================] - 336s 2s/step - loss: 0.1587 - accuracy: 0.9383 - val_loss: 0.1861 -
val_accuracy: 0.9227
```

מבנה / ארכיטקטורה של הפרויקט

שלב איסוף הכנה וניתוח הנתונים

מבנה הנתונים נלקח מאתר Kaggle, קישור במסמכים. תחילה מצאתי dataset אשר יש בו תמונות של פחיות, נייר, זכוכית ופלסטיק. לא בחרתי בו כיוון שאין בו מספיק תמונות, ורוב התמונות הן כבר אחרי שינוי augmentation, ולכן לא בחרתי בכך. לאחר מכן המשכתי לחפש עד שמצאתי את ה dataset המתאים. ה dataset מחולק ל train, test ובכל אחד מהם יש 2 סוגים של תמונות- אורגני ומחזור. אני בחרתי להשתמש בחלק המחזור וחילקתי אותו לקטגוריות- ספרים, קרטונים, בקבוקי זכוכית ועיתונים. ובתמונות של האורגני השתמשתי בחלק שלא הולך למחזור ואף אותו חילקתי לקטגוריות- פירות, ירקות, בשר, טבע.

לאחר שקיבלתי את מבנה הנתונים המקורי, ערכתי בו שינויים רבים. ראשית, עברתי עליו ומחקתי ידנית תמונות שהיו נראות לי כפחות רלוונטיות, כמו לדוגמה בקטגוריית הקרטונים הייתה תמונה של מגזין אופנה, אשר אינו נחשב לקרטון. שמתי לב שלכל תמונה יש גודל אחר ושונה, ולכן הדבר הראשון שעשיתי היה לקבוע את הגודל שלהם ל 100x100. image.resize((100, 100)) .

לאחר מכן הבנתי שמספר התמונות שיש לי הוא קטן מאוד ואינו מספיק לאימון המודל. לכן עשיתי שינוי לכל תמונה. תחילה, הכפלתי כל תמונה במספר מסוים(רציתי שבכל קטגוריה יהיו 7000 תמונות, ולכן הרצתי מספר פעמים של 7000 חלקי מס התמונות המקורי) כך שכל תמונה תופיע מספר פעמים, אך תהיה שונה. שיניתי את הכהות בכל תמונה, חלק יותר בהירות וחלק יותר כהות.

בהמשך, שיניתי כל תמונה(אחרי שיש מספר גרסאות שלה ברמת כהות שונה) בכך שעשיתי לכל תמונה סיבוב בכמה מעלות. כך יצא, שהגעתי למבנה נתונים מספיק גדול, המכיל תמונות שונות זו מזו באמצעות ה augmentation שעשיתי.

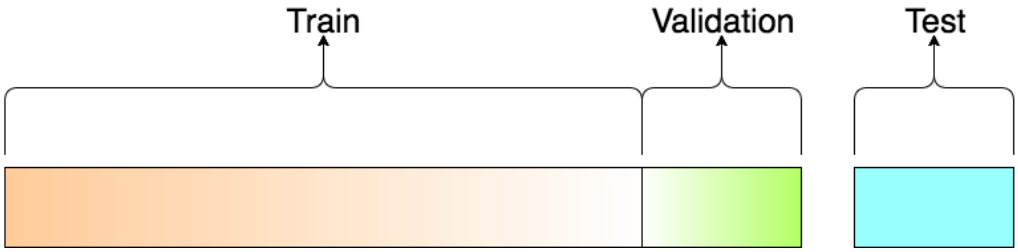
אחרי שה dataset היה מוכן, חילקתי אותו ל train, test, validation.

למעשה, אימון המודל מתבסס על דוגמאות. ועל מנת לקבל תוצאות מיטביות, יש לאמן אותו על הרבה דוגמאות שונות.

אני חילקתי את המודל (ב default) כך:
70% train- החלק ממבנה הנתונים עליו מאמנים את המודל, המודל לומד ממנו לזהות ולהכיר את הנתונים. בחלק זה נקבעים הערכים של המשקלים.

10% validation- נועד להעריך את התוצאות של המודל הנבנות ב train. הוא למעשה פועל כמו test, אך עושה זאת בחלק של בחירת המשקלים. אין חובה להשתמש ב validation.

20% test- החלק ממבנה הנתונים הנועד להעריך את המודל, כלומר עד כמה המודל טוב וכמה המשקלים שנבחרו בחלק האימון (train, validation) היו מדויקים. חלק זה יכול לקרות רק לאחר שהמודל כבר אומן ונבחרו משקלים.



כפי שניתן לראות בתמונה, train ו validation כתובים יחדיו, שכן שניהם בחלק אימון המודל. ה test מופרד מהם, מכיוון שהוא עוסק בהערכת המודל. בנוסף, ניתן לראות כי חלק האימון הוא החלק הגדול יותר(מכיל יותר ערכים) מכיוון שהוא לומד וצריך המון דוגמאות שונות כדי שיוכל להגיע לתוצאה המיטבית.

הסבר על שיטת הנרמול

נראה, כי המודל מתמודד בקלות יותר עם משתנים וערכים במקנה מידה קטן מאשר גדול.

נרמול, משמע להפוך מספרים גדולים מאוד לטווח של בין 0 ל 1. הנרמול עוזר למודל להתמודד עם מספרים כה גדולים. מטרתו היא לעזור למודל להתמודד עם בניית המשקלים העצומים. בעזרת כך, המודל יהיה מדויק יותר, וכך גם תוצאות המודל ב training.

נוסחת הנרמול נראית כך,

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Normalization

למעשה, הנוסחה מורכבת מחילוק של הערך אותו מנרמלים פחות הערך המינימלי בכל המאגר, בטווח קבוע המורכב מהערך המקסימלי פחות הערך המינימלי.

שלב בנייה ואימון המודל

תיאור גרפי של המודל שעליו בוצע האימון

```
model = Sequential()
model.add(Conv2D(16, (3, 3),
                 activation='relu',
                 kernel_initializer='he_uniform',
                 input_shape=(100, 100, 3)))

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(32, (3, 3),
                 activation='relu',
                 kernel_initializer='he_uniform'))

model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3),
                 activation='relu',
                 kernel_initializer='he_uniform'))

model.add(Conv2D(32, (3, 3),
                 activation='relu',
                 kernel_initializer='he_uniform'))

model.add(MaxPooling2D((2, 2)))
model.add(Flatten())

model.add(Dense(1,
                activation='sigmoid'))

Adam = SGD(lr=0.01, momentum=0.9)
model.compile(optimizer=Adam,
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

השכבות המרכיבות את רשת הנוירונים

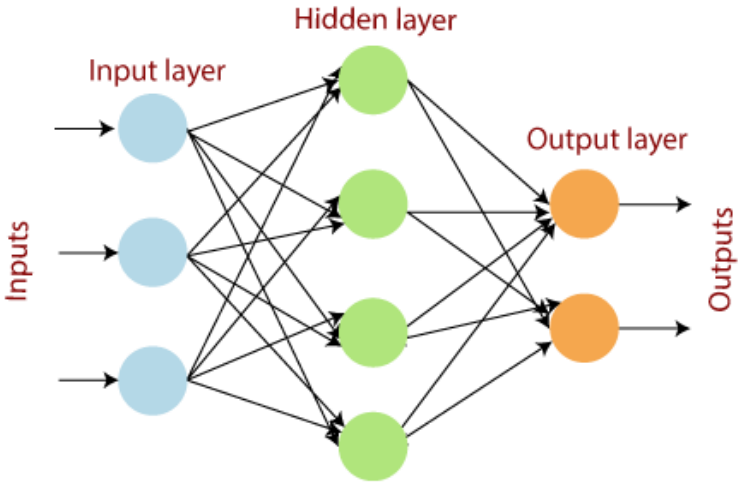
מקור השם "neural network" הוא ממוח האדם. רשת הנוירונים מחקה את התנהגות המוח האנושי, כפי שהנוירונים במוח מתפקדים.

רשת הנוירונים מורכבת משלוש שכבות מרכזיות.

שכבה ראשונה נקראת "input layer". שכבה זו היא השכבה הראשונה. היא מקבלת את מאגר הנתונים ומעבירה אותו להמשך הרשת.

שכבה שניה נקראת "hidden layer". שכבה זו יכולה להופיע פעם אחת או יותר. בחלק מהמקרים, בשכבות אלו המשקלים נקבעים רנדומלית, ובמקרים אחרים הם משתנים באמצעות תהליך שנקרא "backpropagation". הנוירונים מתפקדים כמו נוירון הקיים במוח האדם- מקבל input, משנה אותם, ועובד עליהם באמצעות המשקלים ובאמצעות פונקציות ולאחר מכן ממיר אותם ל output.

שכבה שלישית ואחרונה נקראת "output layer". שכבה זו מחזיקה בתוצאות הסופיות לאחר העבודה שנעשתה בשכבה השנייה.



המודל שיצרתי:

```
model = Sequential()
model.add(Conv2D(16, (3, 3),
                 activation='relu',
                 kernel_initializer='he_uniform',
                 input_shape=(100, 100, 3)))

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(32, (3, 3),
                 activation='relu',
                 kernel_initializer='he_uniform'))

model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3),
                 activation='relu',
                 kernel_initializer='he_uniform'))

model.add(Conv2D(32, (3, 3),
                 activation='relu',
                 kernel_initializer='he_uniform'))

model.add(MaxPooling2D((2, 2)))
model.add(Flatten())

model.add(Dense(1,
                activation='sigmoid'))

Adam = SGD(lr=0.01, momentum=0.9)
model.compile(optimizer=Adam,
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

מושגים כלליים:

Conv2D:

יוצרת שכבה עם convolution kernel, אשר יוצרת קשר בין השכבה שלפניה לבין השכבה שאחריה.

Filter:

החלק מהתמונה(הפיקסלים) שהמודל יעבור עליו. כלומר, פילטר של (3,3) אומר שהמודל יעבוד על קובייה בגודל 3 שורות על 3 עמודות. הוא יעבוד כל פעם על קובייה אחרת עד שיעבוד על כל הפיקסלים שבתמונה.

Neurons:

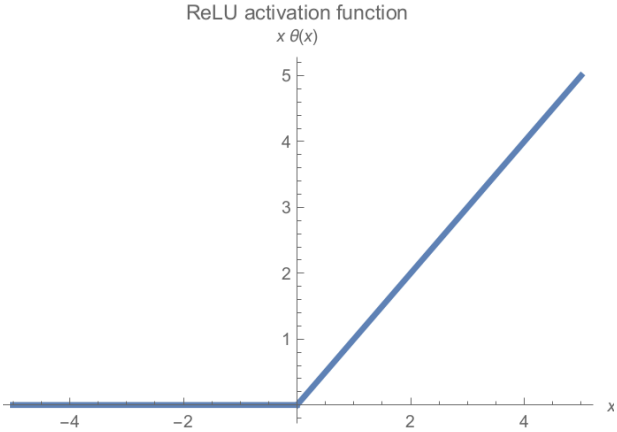
נוירונים הם הבסיס של למידת מכונה. הם מקבלים קלט, לעיתים מנוירון אחר ולעיתים מ ה data. אחרי כן, הם מבצעים חישובים ולבסוף שולחים פלט, תוצאות.

Activation function:

Activation function משנה את המשקלים שהוגדרו ברנדומליות, למשקלים עם התוצאות המיטביות ביותר. Activation function נמצאת בכל שכבה, בשכבת hidden layer, הפונקציות הן בדרך כלל זהות, ובשכבה אחרונה הפונקציה היא בדרך כלל שונה.

Relu:

Activation function הנפוצה ביותר. זאת מכיוון שזוהי פונקציה קלה לתפעול ולהבנה. הפונקציה פועלת כך שכאשר הערך המתקבל קטן מאפס הערך של הפונקציה הוא 0, וכאשר הערך המתקבל גדול מאפס, הערך של הפונקציה הוא כמו הערך שמתקבל. היתרון הוא שכאשר הערך גדול מאפס השארית היא 1.



Kernel_initializer:

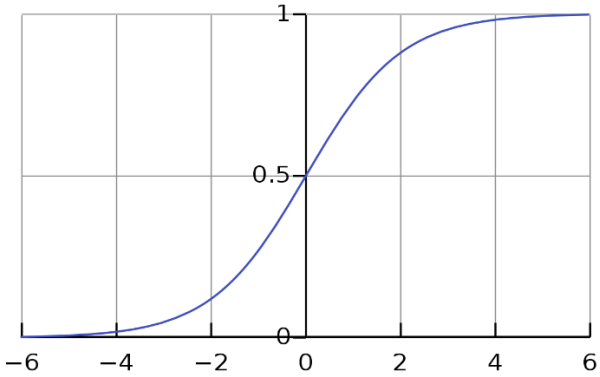
מאתחל את המשקלים לערך קבוע.

Dense:

שכבת dense היא שכבה שבה הנוירון מתחבר לכל נוירון מהשכבה הקודמת. הנוירון בשכבה מקבל פלט מהנוירונים הקדומים, ומחשב את הפלט באמצעות ערכים אלה.

Sigmoid:

Activation function. ערכי הפונקציה נעים בין 0 ל 1, וכאשר הערך שהיא מקבלת שווה ל 0.5, ערכי הפונקציה הם 0. הפונקציה שימושית בעיקר לשכבה האחרונה, ומותאמת בעיקר למקרים של עבודה עם סיווג בינארי- (0 או 1).



<u>שכבות המודל</u>:

במודל שכתבתי ישנן 5 שכבות.

שכבה 1-

Number of neurons	16
Filter	3,3
Activation function	Relu
Kernel_initializer	He_uniform
Input_shape	100,100,3

שכבה 2-

Number of neurons	32
Filter	3,3
Activation function	Relu
Kernel_initializer	He_uniform

שכבה 3-

Number of neurons	64
Filter	3,3
Activation function	Relu
Kernel_initializer	He_uniform

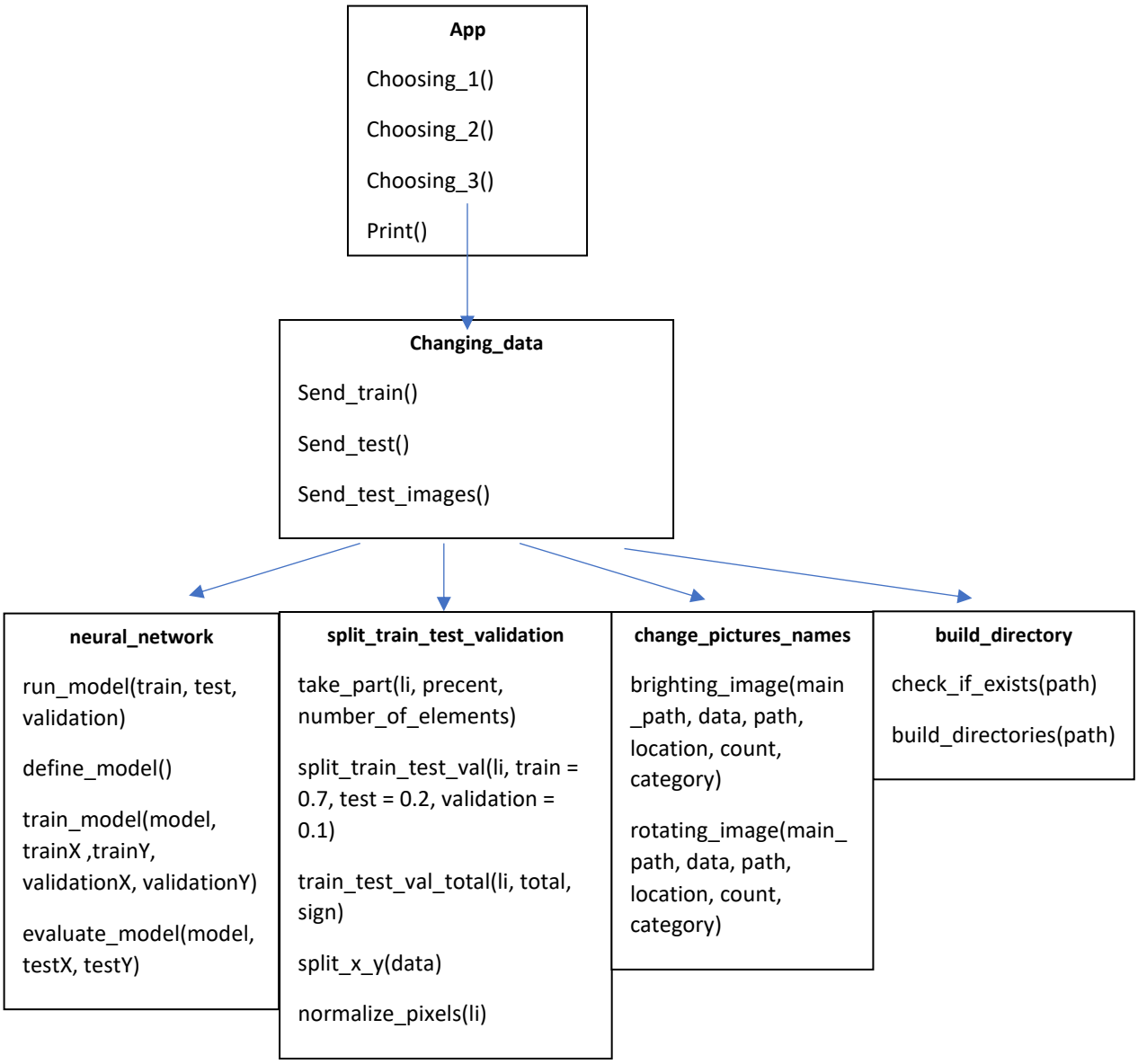
שכבה 4-

Number of neurons	32
Filter	3,3
Activation function	Relu
Kernel_initializer	He_uniform

שכבה 5-

Dense	1
Activation function	Sigmoid

תרשים UML של המודלים:



דוח הכולל ריכוז כל ה- Hyper Parameters:

Hyperparameters Related To Neural Network Structure	Hyperparameters Related To The Training Algorithm
Number of hidden layers- 5	Learning rate- 0.01
Dropout- 0.2	Epoch-20 Iterations- 155 batch size- 256
Activation function- relu and sigmoid	Optimizer algorithm- SGD
Weights initialization- he uniform	Momentum- 0.9

Sequential:

מודל Sequential הוא מודל אשר מתאים למספר מסוים של שכבות(יותר מאחד), כאשר כל שכבה יש בדיוק input ו output אחד.

MaxPooling2D:

כאשר מוסיפים אותו למודל הוא מפחית מהממדיות של התמונות. כלומר, הוא מפחית מהפיקסלים של התמונה. יש לו יתרונות רבים, ובניהם, הוא משתמש בפחות משתנים, מפחית את הסיכוי ל overfitting.

Dropout:

מודל אשר קובע ערכים מסוימים ל 0 (רנדומלית)במהלך ה training. מספר הערכים הוא קבוע ונתון לידי המשתמש. נועד על מנת למנוע overfitting- מצב אשר הערכים והתוצאות של המודל מדויקים מדי.

Flatten:

משטיח את הערכים, מבלי לשנות את ה batch size. לדוגמה אם מקבלים (batch size,2,2), אז זה יהפוך ל (batch size, 4).

Learning rate:

פרמטר אשר אחראי על כמה אנו רוצים לשנות את המודל כתגובה לשגיאה(loss) בכל פעם שהמשקלים משתנים. ערך קטן מדי עלול לגרום להרצה מאוד איטית של המודל, ואילו ערך גדול מדי יכול לשנות א המשקלים בצורה מהירה מדי ולא יעילה.

Epoch:

Epoch אחד הוא כאשר הdataset עובר במודל- (forward, backward).

Batch size:

המספר הכולל של הדוגמאות ב train, ב batch אחד.

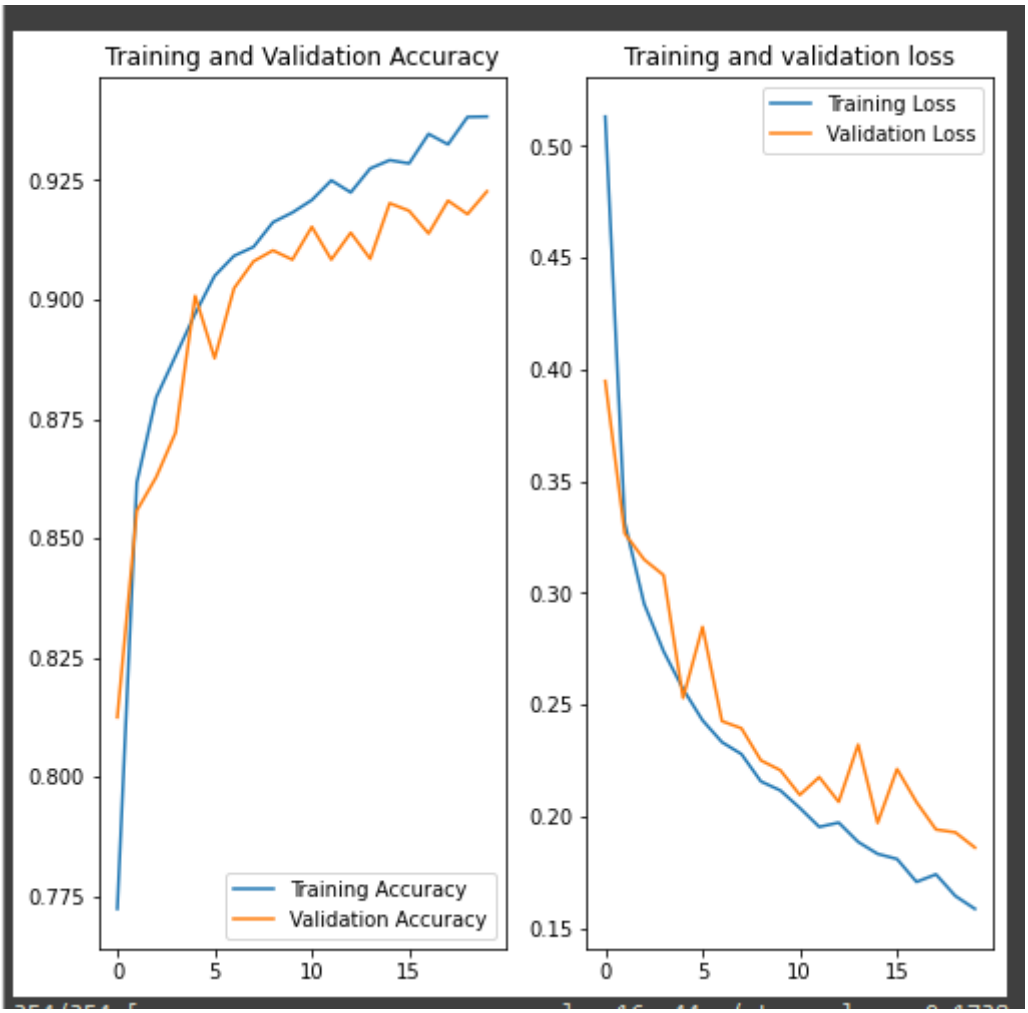
Iteration:

מספר ה batches שצריך כדי להשלים epoch אחד.

Accuracy:

מודל המעריך כמה המודל מדויק, כלומר כמה המשקלים מתאימים למודל. ככל ש ה accuracy יותר גבוה, כך המודל מדויק יותר.

<u>דוחות וגרפים המתארים את תוצאות האימון</u>



מימין ניתן לראות גרף של ה loss ב train וב validation. ניתן לראות כי תוצאות הגרף מראות כי ה loss הולך ויורד, כלומר המודל הופך ונהיה טוב יותר.

משמאל ניתן לראות גרף של ה accuracy ב train וב validation. ניתן לראות כי הערכים הולכים וגדלים, דבר המעיד על העובדה שהמודל הולך ומשתפר.

תיעוד השינויים שנעשו במודל וב Hyper Parameters

כאשר ניסיתי להריץ את המודל, שיניתי מספר גורמים רלוונטיים שיכולים להשפיע על תוצאות: מספר הנוירונים, סוג פונקציית ה loss, וכו.

המודל הראשון שבניתי:

שכבה 1-

Number of neurons	16
Filter	3,3
Activation function	Relu
Kernel_initializer	He_uniform
Input_shape	100,100,3

שכבה 2-

Number of neurons	32
Filter	3,3
Activation function	Relu
Kernel_initializer	He_uniform

שכבה 3-

Number of neurons	64
Filter	3,3
Activation function	Relu
Kernel_initializer	He_uniform

שכבה 4-

Number of neurons	64
Filter	3,3
Activation function	Relu
Kernel_initializer	He_uniform

שכבה 5-

Dense	1
Activation function	Sigmoid

Hyperparameters Related To Neural Network Structure	Hyperparameters Related To The Training Algorithm

Number of hidden layers- 5	Learning rate- 0.01
Dropout- 0.2X2	Epoch-20 Iterations- 155 batch size- 256
Activation function- relu and sigmoid	Optimizer algorithm- opt
Weights initialization- he uniform	Momentum- 0.9

תוצאות המודל הראשון:

Test_loss	0.3257
Test_accuracy	0.85

המודל השני שבניתי:

שכבה 1-

Number of neurons	16
Filter	3,3
Activation function	Relu
Kernel_initializer	He_uniform
Input_shape	100,100,3

שכבה 2-

Number of neurons	32
Filter	3,3
Activation function	Relu
Kernel_initializer	He_uniform

שכבה 3-

Number of neurons	64
Filter	3,3
Activation function	Relu
Kernel_initializer	He_uniform

שכבה 4-

Number of neurons	128
Filter	3,3
Activation function	Relu
Kernel_initializer	He_uniform

שכבה 5-

Dense	1
Activation function	Sigmoid

Hyperparameters Related To Neural Network Structure	Hyperparameters Related To The Training Algorithm	
Number of hidden layers- 5	Learning rate- 0.01	
Dropout- 0.2X2	Epoch-20 Iterations- 155 batch size- 256	
Activation function- relu and sigmoid	Optimizer algorithm- opt	
Weights initialization- he uniform	Momentum- 0.9	

תוצאות המודל השני:

Test_loss	0.466
Test_accuracy	0.798

הסבר על פונקציית השגיאה

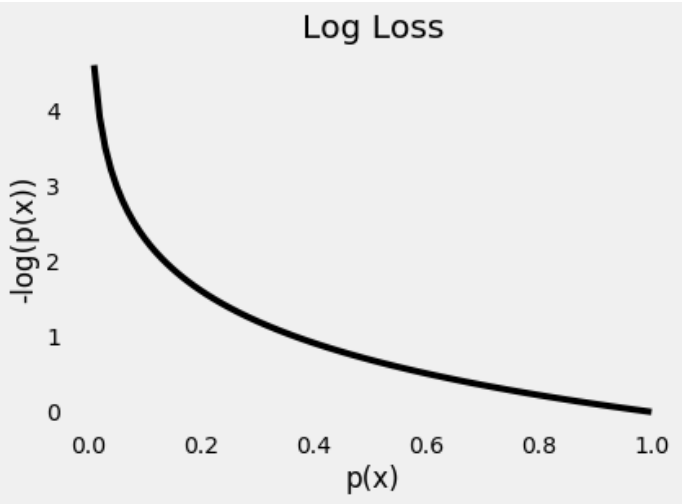
פונקציית השגיאה או loss function, היא פונקציה היודעת להעריך עד כמה המודל מדויק. הפונקציה מחשבת את ההפרש בין הפלט לבין הערך שהיה אמור להיות (שאותו הגדרנו), ובאמצעותו מחשבת את דיוק המודל. ככל שהערך של פונקציית השגיאה קטן יותר, כך המודל יותר מדויק, וככל שהערך של הפונקציה גדול יותר, כך המודל פחות מדויק. כל שינוי שנעשה במודל, ישנה לנו את ערך פונקציית השגיאה, ובאמצעותו נוכל לדעת אם השינוי הועיל או לא.

ישנן הרבה פונקציות ממשפחת פונקציית השגיאה. בפרויקט השתמשתי בפונקציית "binary cross-entropy". פונקציה זאת נועדה בעיקר עבור מקרים אשר הפלט שלהם הוא כן או לא, 0 או 1. במקרה שלי, מחזור או לא מחזור. הפונקציה, ככל פונקציות השגיאה מחשבת עד כמה המודל שלי היה מדויק.

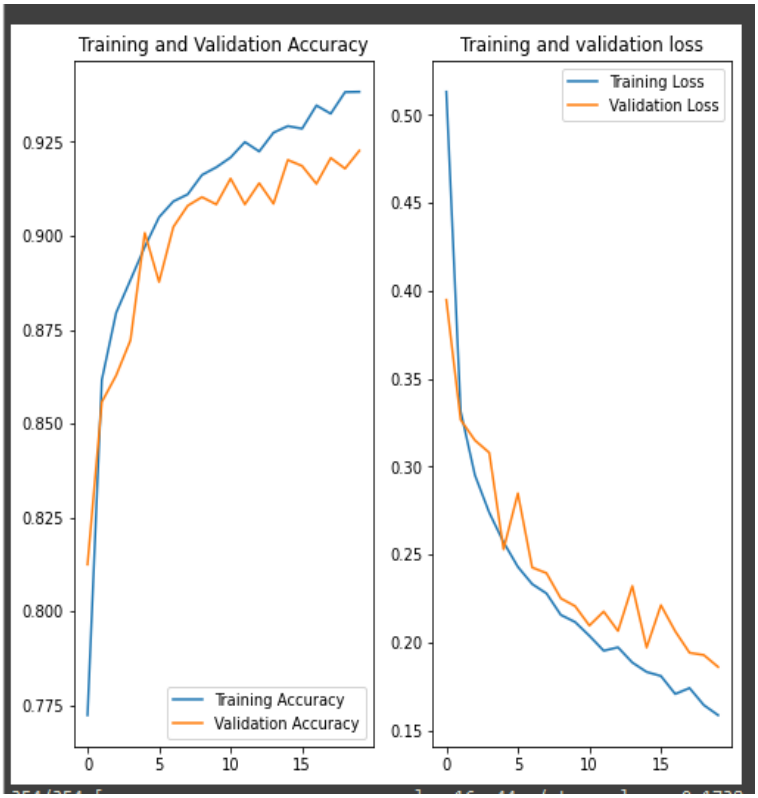
$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

זוהי הנוסחה לחישוב פונקציה זו. Y מסמל את ה label (0 או 1), והערך p(y), מסמל את ההסתברות שהנקודה y תהיה מחזור. N מסמל את מספר הנקודות, הערכים.

הנוסחה אומרת שעבור (y=1), לפונקציה נוסף הערך log(p(y)) ההסתברות שיהיה למחזור, ונוסף הערך log(1-p(y)) להסתברות שלא יהיה למחזור.



כך לבסוף, נראה הגרף של פונקציה זו. בהתאם לנוסחה ניתן לראות כי כאשר p(0), הערך של הפונקציה הוא גדול מאוד (בעקבות ה log), וכאשר p(1), הערך של הפונקציה הוא 0 (בעקבות ה log).



הסבר ייעול ההתכנסות (optimization)

תהליך ייעול ההתכנסות הוא תהליך שבו מאמנים את המודל באופן החוזר על עצמו עד ליצירת פונקציות אומדן- פונקציית מינימום ופונקציית מקסימום. אנו משתמשים בתהליך זה מכיוון שבעזרתו יוצרים מודל מדויק יותר. עבור כל שלב של אימון המודל(כל epoch), האופטימייזר משווה את תוצאות המינימום ומקסימום ומשנה אותם. ישנם אופטימייזרים רבים.

בפרויקט שלי השתמשתי ב SGD, או Stochastic gradient descent. פונקציה זו היא יעילה, שכן היא עובדת מהר עם מודלים גדולים.

תוצאות המודל הסופיות (test)

```
[0.17383377254009247, 0.9332627058029175]
```

מימין ניתן לראות את ה accuracy של המודל, ומשמאל ניתן לראות את ה loss של המודל.

שלב היישום

תיאור והסבר כיצד היישום משתמש במודל

האפליקציה שבניתי מכילה מספר אפשרויות שהמשתמש יכול לבחור בהן:

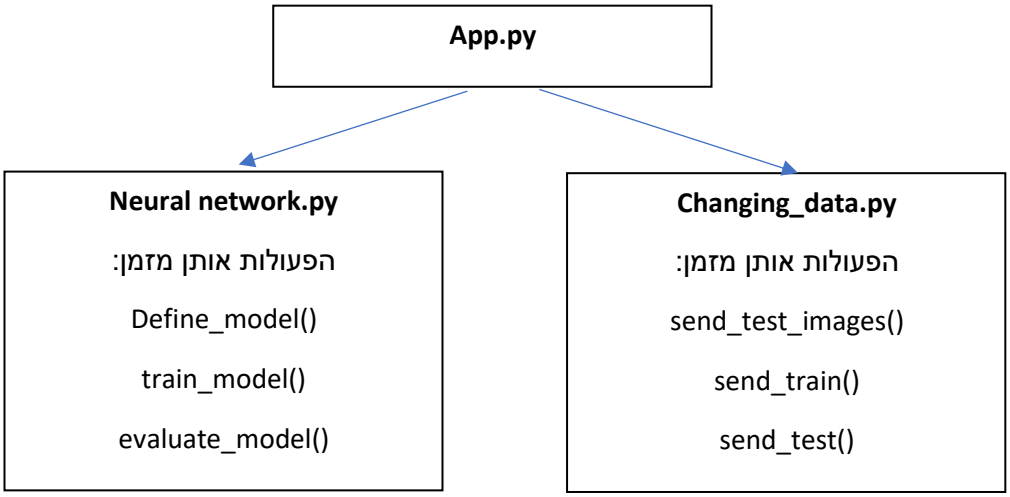
1. המשתמש יכול לבחור באופציה הנקראת: "train", כלומר להריץ את המודל עם ערכי ה train בלבד. בחלק זה הפונקציה מריצה את המודל, את הפעולה הנקראת: " train_model", אשר עליה הרחבתי בפרק במדריך למפתח. היא מדפיסה את ערכי ה accuracy ו ה loss של ה train ושל ה validation. בנוסף, מדפיסה גרפים להמחשת המודל.

2. המשתמש יכול לבחור באופציה הנקראת "test", כלומר הרצת המודל עם המודל השמור. המשתמש יראה את תוצאות המודל- accuracy ו loss.

3. אופציה נוספת היא לבחור בחיזוי תמונה ספציפית, אשר המשתמש בוחר. הוא בוחר מספר, כמובן ישנה תקינות קלט, ולאחר מכן הפונקציה שבניתי תראה את ה label של התמונה- 0 אם מחזור, 1 אם לא מחזור. בנוסף, הפונקציה מדפיסה אם החיזוי של המודל היה נכון או לא, כלומר אם היא חזתה שהתמונה היא כמו הערך האמיתי שלה.

4. אופציה אחרונה היא האופציה "exit". המשתמש יכול לבחור לצאת ולא להמשיך.

תרשים UML



מדריך למפתח

שם הקובץ	מיקומו
app.py	shakedProjectY/app.py
build_directory.py	shakedProjectY/ build_directory.py
change_pictures_names	shakedProjectY/ change_pictures_names
changing_data.py	shakedProjectY/ changing_data.py
nural_network.py	shakedProjectY/ nural_network.py
split_train_test_validation.py	shakedProjectY/ split_train_test_validation.py

<u>פעולות בקבצים + הסברי משתנים</u>

change_pictures_names- כל הפעולות מזומנות מהמודול changing_data.py.

שם הפעולה	מה מחזירה	תפקידי משתנים	מטרת הפעולה
rotating_image	לא מחזירה	**Main_path-** הכתובת שבה נמצאת הספרייה של התמונות העתידיות **Data-** ה dataset של כל קטגוריה **Path-** הכתובת המשייכת את הקטגוריה של התמונה, למשל קרטונים. **Location-** מכיל אם התמונה היא למחזור או לא. **Count-** מספר הפעמים שכל תמונה תופיע בגרסאות שונות **Category-** הקטגוריה שאליו משתייך התמונה **Last location-** שומר את המקום האחרון שהיה התמונה	הוספה ל dataset כל תמונה מס פעמים בזווית אחרת.

brighting_image	לא מחזירה	**-Main_path** הכתובת שבה נמצאת הספרייה של התמונות העתידיות **-Data** ה dataset של כל קטגוריה **-Path** הכתובת המשייכת את הקטגוריה של התמונה, למשל קרטונים. **-Location** מכיל אם התמונה היא למחזור או לא. **-Count** מספר הפעמים שכל תמונה תופיע בגרסאות שונות **-Category** הקטגוריה שאליו משתייך התמונה **-Last location** שומר את המקום האחרון שהיה התמונה	שינוי גודל התמונות והוספה ל dataset כל תמונה מס פעמים בכהות אחרת.

-build_directory.py

שם הפעולה	מה מחזירה	תפקידי משתנים	מטרת הפעולה
check_if_exists	לא מחזירה	**-isExist** מכיל כן או לא אם הכתובת הנתונה קיימת או לא **-path** הכתובת שהפעולה מקבלת	מקבל כתובת ובודק אם היא קיימת, אם לא הוא יוצר אותה
build_directories	לא מחזירה	**-path** הכתובת שהפעולה מקבלת	מזמן על כל pathשמקבל את הפעולה check_if_exists

-app.py

שם הפעולה	מה מחזירה	תפקידי משתנים	מטרת הפעולה
Choosing_1	לא מחזירה	-Model המודל הבנוי -trainX מכיל את ערכי ה train. -trainY labelsמכיל את ה של train. -validationX מכיל את ערכי ה validation -validationY labelsמכיל את ה של validation	לאמן את המודל- מה שהמשתמש בוחר.
Choosing_2	לא מחזירה	-trainX מכיל את ערכי ה test. -trainY מכיל את ה labels של test -loaded_model המודל שאומן	לתת תוצאות של המודל הקיים- מה שהמשתמש בוחר
Choosing_3	לא מחזירה	-loaded_model המודל שאומן -Img יכיל את התמונה המנורמלת -Num יכיל את מספר התמונה שהמשתמש בחר. -Result יכיל את ה predict של התמונה.	לחזות ערך של תמונה אשר המשתמש בוחר.
print	מחזירה את הבחירה של המשתמש	-Choice הבחירה של המשתמש	מדפיס על המסך את האפשרויות של המשתמש, הוא בוחר את האפשרות.

–changing_data.py

שם הפעולה	מה מחזירה	תפקידי משתנים	מטרת הפעולה
Send_train	את ערכי התמונות וה labels של ה train, validation	אין	להחזיר את ערכי התמונות וה labels של ה train, validation
Send_test	את ערכי התמונות וה labels של ה test	אין	להחזיר את ערכי התמונות וה labels של ה test
send_test_images	את התמונות הקיימות ב test(לפני נרמול) ואת ה labels שלהן.	אין	להחזיר את התמונות הקיימות ב test(לפני נרמול) ואת ה labels שלהן.

-split_train_test_validation.py

שם הפעולה	מה מחזירה	תפקידי משתנים	מטרת הפעולה
take_part	מחזיר את הרשימה החדשה	li- ה- dataset precent- מספר האחוזים מתוך li אשר הרשימה החדשה תכיל. number_of_elements- מכיל את הגודל של li. new_li- הרשימה החדשה (החלק באחוזים מ li)	מקבל את ה dataset, והופך אותו לרשימה חדשה אשר מכילה אחוז מסוים מ ה dataset.

split_train_test_val	מחזיר את 3 הרשימות, train, test, validation.	number_of_elements- אורך הרשימה li li- ה dataset train_data- מזמן את הפעולה take_part, ומכיל את ה- train 70%. test_data- זמן את הפעולה take_part, ומכיל את ה- test 20%. validation_data- זמן את הפעולה take_part, ומכיל את ה- validation 10%.	יוצר 3 רשימות- train, test, validation מתוך ה li.
train_test_val_total	מחזירה רשימה לאחר שהוסיפה את li ל total.	Li- רשימה המכילה את כל התמונות מקטגוריה מסוימת Total- ה dataset Sign- אם מכיל אמת, אז מסדר את הרשימה בסדר אקראי	הפעולה מסדרת את הרשימה מחדש, ומוסיפה את li ל total.
split_x_y	מחזירה את ה dataset ואת הרשימה עם ה labels.	x- מכיל את ה dataset y- רשימה שמכילה labels של כל תמונה, 0 אם למחזור 1 אם לא למחזור category- מכיל את הקטגוריה של התמונה, למשל בשר.	הפונקציה יוצאת labels עבור כל תמונה ב dataset.
normalize_pixels	מחזירה רשימה מנורמלת.	li- מכיל את ה dataset	מקבל רשימה ומנרמל אותה.

-nural_network.py

שם הפעולה	מה מחזירה	תפקידי משתנים	מטרת הפעולה
run_model	לא מחזירה	-Model מכיל את המודל -History מכיל את ההיסטוריה של המודל	מזמנת את הפעולות define_model, train_model, evaluate_model
define_model	מחזיר את ה model		יוצר את המודל
train_model	מחזיר את ה history	-History מכיל את ההיסטוריה של המודל -Accuracy מכיל את ה accuracy של ה train -val_acc מכיל את ה accuracy של ה validation -Loss מכיל את ה loss של ה train -val_loss מכיל את ה loss של ה validation	מריץ את המודל ב train, ויוצר גרפים של accuracy, ו loss
evaluate_model	לא מחזירה	-Score מכיל את תוצאות המודל	מדפיס את תוצאות המודל

מדריך למשתמש

התקנות וסביבת עבודה

הפרויקט שלי עובד על anaconda, בסיסת העבודה spyder.

קישור להתקנה של anaconda: https://www.anaconda.com/products/distribution

בנוסף, השתמשתי במספר ספריות, אשר אותן יש להוריד גם כן:

שם הספרייה	הקוד להורדה	לינק להורדה
opencv	pip install opencv	https://www.codegrepper.com/code-examples/python/how+to+install+opencv+in+spyder
numpy	pip install numpy	https://pypi.org/project/numpy/
keras	pip install keras	https://pypi.org/project/keras/
matplotlib	pip install matplotlib	https://pypi.org/project/matplotlib/
tensorflow	pip install tensorflow	https://pypi.org/project/tensorflow/

תקשורת עם המשתמש

סביבת התקשורת עם המשתמש היא CMD.

תחילה, מופיע מסך ראשוני למשתמש אשר פותח בפניו את כל האפשרויות שלו ומבקש ממנו לבחור מספר בין 1-4:

```
Hello!
My project is about recycling classification.

Choose an option
1. Train the model
2. Test the model
3. Predict an image
4. Exit
```

אם המשתמש לא בחר מספר בין 1-4, אלא כל דבר אחר, תודפס לו הודעת שגיאה שתבקש ממנו לבחור מחדש מספר תקין.

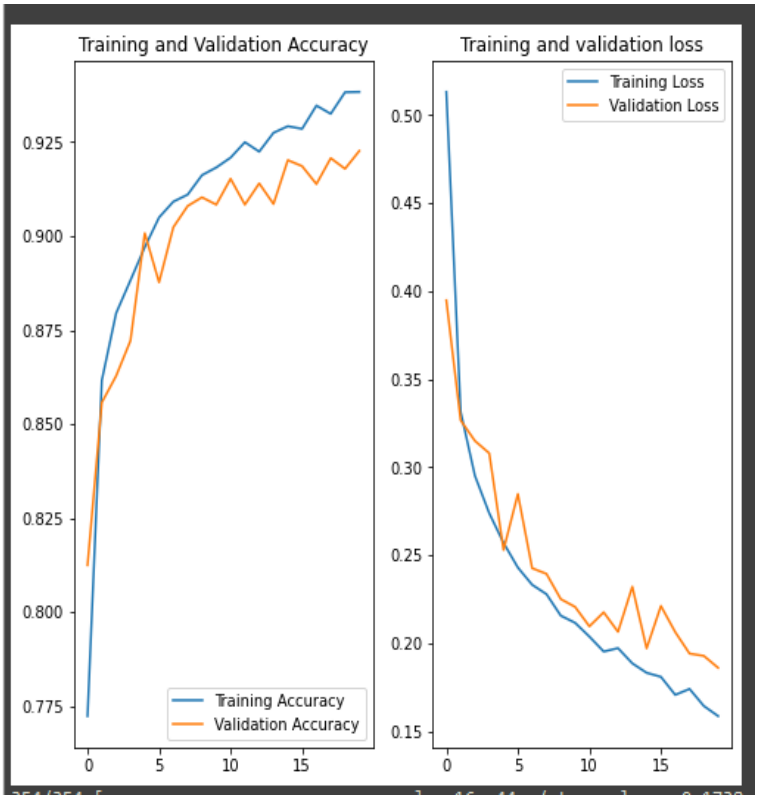
אם המשתמש בחר **באופציה 1**, הוא יראה על המסך את החלק הבא:

```
Enter your choice: 1
Epoch 1/20
WARNING:tensorflow:AutoGraph could not transform <function Model.make_train_function.<locals>.train_function at
0x00000155669EB798> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the verbosity to 10 (on Linux, `export
AUTOGRAPH_VERBOSITY=10`) and attach the full output.
Cause: 'arguments' object has no attribute 'posonlyargs'
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
WARNING: AutoGraph could not transform <function Model.make_train_function.<locals>.train_function at
0x00000155669EB798> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the verbosity to 10 (on Linux, `export
AUTOGRAPH_VERBOSITY=10`) and attach the full output.
Cause: 'arguments' object has no attribute 'posonlyargs'
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
 32/155 [=====>.......................] - ETA: 4:05 - loss: 0.7926 - accuracy: 0.6248
```

```
155/155 [==============================] - 352s 2s/step - loss: 0.5131 - accuracy: 0.7723 - val_loss: 0.3948 -
val_accuracy: 0.8125
Epoch 2/20
155/155 [==============================] - 345s 2s/step - loss: 0.3316 - accuracy: 0.8616 - val_loss: 0.3266 -
val_accuracy: 0.8557
Epoch 3/20
155/155 [==============================] - 336s 2s/step - loss: 0.2951 - accuracy: 0.8795 - val_loss: 0.3149 -
val_accuracy: 0.8628
Epoch 4/20
155/155 [==============================] - 329s 2s/step - loss: 0.2739 - accuracy: 0.8882 - val_loss: 0.3079 -
val_accuracy: 0.8722
Epoch 5/20
155/155 [==============================] - 61974s 402s/step - loss: 0.2570 - accuracy: 0.8970 - val_loss: 0.2530 -
val_accuracy: 0.9008
Epoch 6/20
155/155 [==============================] - 331s 2s/step - loss: 0.2430 - accuracy: 0.9049 - val_loss: 0.2848 -
val_accuracy: 0.8877
Epoch 7/20
155/155 [==============================] - 319s 2s/step - loss: 0.2332 - accuracy: 0.9092 - val_loss: 0.2426 -
val_accuracy: 0.9023
Epoch 8/20
155/155 [==============================] - 320s 2s/step - loss: 0.2279 - accuracy: 0.9110 - val_loss: 0.2394 -
val_accuracy: 0.9080
Epoch 9/20
155/155 [==============================] - 328s 2s/step - loss: 0.2157 - accuracy: 0.9162 - val_loss: 0.2251 -
val_accuracy: 0.9103
Epoch 10/20
155/155 [==============================] - 317s 2s/step - loss: 0.2116 - accuracy: 0.9182 - val_loss: 0.2207 -
val_accuracy: 0.9084
Epoch 11/20
155/155 [==============================] - 318s 2s/step - loss: 0.2038 - accuracy: 0.9209 - val_loss: 0.2096 -
val_accuracy: 0.9152
Epoch 12/20
155/155 [==============================] - 319s 2s/step - loss: 0.1953 - accuracy: 0.9250 - val_loss: 0.2176 -
val_accuracy: 0.9084
Epoch 13/20
155/155 [==============================] - 317s 2s/step - loss: 0.1972 - accuracy: 0.9224 - val_loss: 0.2065 -
val_accuracy: 0.9140
Epoch 14/20
155/155 [==============================] - 318s 2s/step - loss: 0.1886 - accuracy: 0.9275 - val_loss: 0.2321 -
val_accuracy: 0.9085
Epoch 15/20
155/155 [==============================] - 317s 2s/step - loss: 0.1833 - accuracy: 0.9292 - val_loss: 0.1971 -
val_accuracy: 0.9202
Epoch 16/20
155/155 [==============================] - 330s 2s/step - loss: 0.1810 - accuracy: 0.9285 - val_loss: 0.2212 -
val_accuracy: 0.9186
Epoch 17/20
155/155 [==============================] - 328s 2s/step - loss: 0.1707 - accuracy: 0.9347 - val_loss: 0.2064 -
```

```
Epoch 17/20
155/155 [==============================] - 328s 2s/step - loss: 0.1707 - accuracy: 0.9347 - val_loss: 0.2064 -
val_accuracy: 0.9138
Epoch 18/20
155/155 [==============================] - 339s 2s/step - loss: 0.1741 - accuracy: 0.9325 - val_loss: 0.1942 -
val_accuracy: 0.9207
Epoch 19/20
155/155 [==============================] - 341s 2s/step - loss: 0.1644 - accuracy: 0.9383 - val_loss: 0.1929 -
val_accuracy: 0.9179
Epoch 20/20
155/155 [==============================] - 336s 2s/step - loss: 0.1587 - accuracy: 0.9383 - val_loss: 0.1861 -
val_accuracy: 0.9227
```

הקוד יריץ את המודל עם ערכי ה train, validation. הוא יציג על המסך את המודל הרץ ואת ערכי ה loss, accuracy, ואף, יראה גרף של הערכת ביצועים- loss, accuracy.



אם המשתמש בחר **באופציה 2**, הוא יראה על המסך את החלק הבא:

```
Enter your choice: 2
WARNING:tensorflow:AutoGraph could not transform <function Model.make_test_function.<locals>.test_function at
0x00000155668268B8> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the verbosity to 10 (on Linux, `export
AUTOGRAPH_VERBOSITY=10`) and attach the full output.
Cause: 'arguments' object has no attribute 'posonlyargs'
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
WARNING: AutoGraph could not transform <function Model.make_test_function.<locals>.test_function at
0x00000155668268B8> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the verbosity to 10 (on Linux, `export
AUTOGRAPH_VERBOSITY=10`) and attach the full output.
Cause: 'arguments' object has no attribute 'posonlyargs'
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
354/354 [==============================] - 15s 42ms/step - loss: 0.1925 - accuracy: 0.9274
[0.19246117770671844, 0.9274364113807678]
```

המשתמש יראה על המסך את הערכת הביצועים- accuracy, loss של המודל הקיים ששמרתי, את ערכי ה test.

אם המשתמש בחר **באופציה 3**, הוא יראה על המסך את החלק הבא:

```
Enter your choice: 3

Choose a number between 1 to 120000: 99
You choose to predict  0
WARNING:tensorflow:AutoGraph could not transform <function Model.make_predict_function.<locals>.predict_function at
0x000001CA43BD9798> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the verbosity to 10 (on Linux, `export
AUTOGRAPH_VERBOSITY=10`) and attach the full output.
Cause: 'arguments' object has no attribute 'posonlyargs'
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
WARNING: AutoGraph could not transform <function Model.make_predict_function.<locals>.predict_function at
0x000001CA43BD9798> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the verbosity to 10 (on Linux, `export
AUTOGRAPH_VERBOSITY=10`) and attach the full output.
Cause: 'arguments' object has no attribute 'posonlyargs'
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
Prediction is:  [0]
Prediction is true
```

המשתמש יתבקש לבחור מספר בין 1 ל 12,000, אשר ישמש כמיקום של תמונה מה test. הפונקציה הכתובה choosing_3, תחשב את ה label של התמונה הספציפית, וידפיס אותה למשתמש. זאת עוד, הוא יחפש את ה prediction של התמונה, ואם היא תהיה זהה ל label שלה, הוא ידפיס שהחיזוי הוא נכון. אם יהיה שונה, הוא ידפיס שהחיזוי שגוי.

אם המשתמש בחר **באופציה 4**, הוא יראה על המסך את החלק הבא:

```
Enter your choice: 4

In [2]:
```

המשתמש יוכל להמשיך לבחור כל מספר שירצה בין 1-4, כל עוד הוא אינו בוחר באופציה 4. אופציה זו מסיימת את ההרצה, ולמעשה את הקשר עם המשתמש.

רפלקציה

- העבודה על הפרויקט היתה קשה. בהתחלה, לא היו לי את כל הכלים על מנת להצליח אותה, אך עם הזמן למדתי לעבוד לבד וללמוד לבד. בשנים הקודמות החומר התבסס על חומרים מהעבר, וזהו תחום שהיה לי חדש מאוד וכמעט ולא ידעתי עליו כלום. בסופו של דבר הגעתי למצב שאני יודעת את החומר ולומדת דברים חדשים תוך כדי.
- קיבלתי המון כלים מהעבודה, בין אם לעבוד לבד, להיעזר במורה ובחברים לעיתים, לדעת איזה באיזה אתרים לחפש עזרה ואיזה פחות מתאימים לי. בנוסף, למדתי המון על החומר, דברים שלא ידעתי קודם שמאוד שמחתי ללמוד.
- מתוך הכלים שציינתי, אשמח לקחתי איתי להמשך ואמשיך לחזק את היכולת לעבוד לבד, את ההתמודדות עם קשיים בעצמי, ואשמח להמשיך ללמוד עוד על התחום המאוד גדול והחשוב הזה.
- במהלך העבודה נתקלתי במספר אתגרים. תחילה, הייתי צריכה ללמוד לתכנת בשפת הפייתון, שפה שהייתה חדשה בשבילי. לאחר מכן, היה מאתגר להבין את עומק למידת המכונה, את "העבודה השחורה" שהמחשב עושה- חלחול לאחור, חלחול קדימה... לאחר מכן הייתי צריכה להנגיש את הפרויקט למשתמש, וללמוד כיצד עושים זאת. האתגר הגדול היה להתמודד עם זה בעצמי.
- המסקנות שלי מהפרויקט הן שלפעמים צריך לצאת מהאזור הנוח וללמוד עוד דברים חדשים, למדתי שתחום למידת המכונה הוא תחום מאוד נרחב, ושזהו תחום חשוב מאוד כיום.
- לו הייתי מתחילה את העבודה היום מחדש, הייתי משנה מספר דברים, וביניהם את הוותרנות שבהתחלה הופיעה, את הלחצים שהכנסתי לעצמי עקב חוסר ידע בתחום. בנוסף, הייתי מתחילה ללמוד קודם חומר עיוני ורק אחר כך מתחילה את הקוד עצמו, זאת כדי שיהיה לי ידע בסיסי.
- למדתי על עצמי שזו לא בשה לפעמים לא לדעת חומר מסוים ולבקש עזרה, למדתי שאני מסוגלת ללמוד תחום חדש מאפס, ושאני לא צריכה לפחד מדברים שאני לא מכירה.

ביבליוגרפיה

קישור למאמר מהמבוא-

https://www.globes.co.il/news/article.aspx?did=1001335742

הסבר על אופטימייזרים-

https://towardsdatascience.com/understanding-optimization-algorithms-in-machine-learning-edfdb4df766b

הסבר על פונקציות loss-

https://www.datarobot.com/blog/introduction-to-loss-functions/

קישור ל dataset של Kaggle, אשר ראיתי בהתחלה, אך לא השתמשתי בו בסוף-

https://www.kaggle.com/datasets/jinfree/recycle-classification-dataset?select=glass

ה dataset שבו השתמשתי-

https://www.kaggle.com/code/gauravrajpal/waste-classification-vgg16-97-00/data