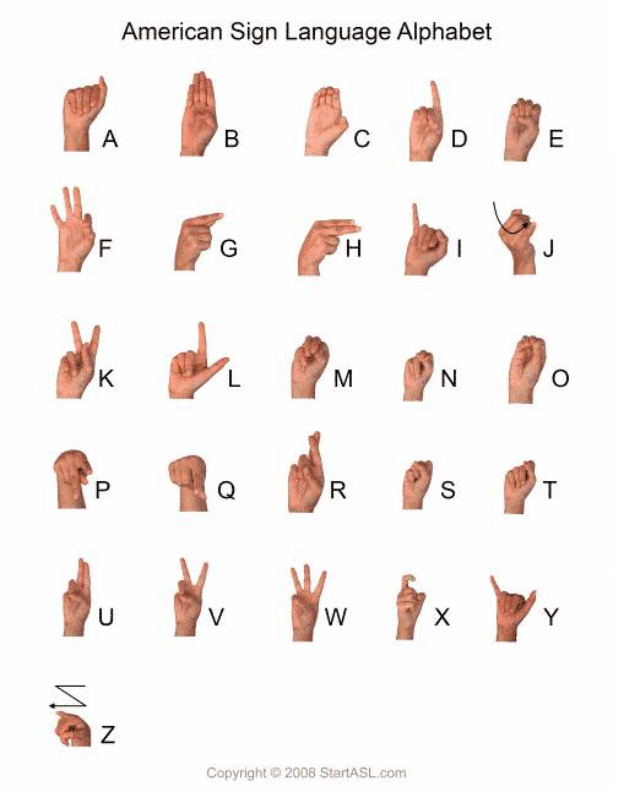


שים בית הספר: מקיף י"א ראשונים

שם העבודה: ASL letters classifier

שם התלמיד: איתמר ברנרד כץ

ת.ז. התלמיד: 214631186

שם המנחה: דינה קראוס

שם החלופה: למידת מכונה

תאריך ההגשה: 16.6.2022

תוכן עניינים

מבוא

הפרויקט גמר שבחרתי לסיום מגמת מדעי המחשב הוא זיהוי אותיות בשפת הסימנים האמריקאית. בתחילת השנה, חיפשתי נושאים שיכולים להתאים לפרויקט הגמר, הרצון התמקד בעיקר בחיפוש נושא שארגיש אליו חיבור ועניין. תחילה, מצאתי dataset של כל האותיות בשפת הסימנים, וכך מצאתי את הנושא מאוד מסקרן.

חקרתי על שפת הסימנים האמריקאית, ומצאתי כי משתמשים באותיות בשפת הסימנים לצורך הבעת שמות של אנשים, מקומות, או מצבים אחרים שאין להם מילה בעצמה. מטרת הפרויקט היא לתת לאנשים שמדברים בשפת הסימנים את האופציה לתקשר בקלות ואת היכולת להעביר מידע בסיסי, כמו שם, בצורה פשוטה. ניתן ללמד ילדים אותיות בשפת הסימנים ולבדוק אותם עם המודל, ובכך לאפשר להם לתקשר עם אנשים חירשים ולהעלות את המודעות לקשיים איתם מתמודדים אנשים חירשים במהלך היום יום.

האותיות J ו-Z הן אותיות שדרושה להן תנועה על מנת לסמן אותן, ולא ניתן לזהות אותיות אלה מתמונה, אלא אך ורק מסרטון לא ניתן להשתמש באותיות אלה במסגרת הפרויקט.

קיים חשש מכמות נמוכה של מידע, כיוון שלא נמצא dataset מספיק רחב, ולא כל dataset מתאים לאחד אחר. כלומר, אין אפשרות לשלב בין datasets שונים, מהסיבה שהתמונות אינן מצולמות באופן קבוע מאותה הזווית, ואין מספר תמונות מספק על מנת שיהיה ניתן לזהות בין מצבים שונים של אותה האות.

לא רק זאת ועוד, קיימות אותיות דומות זו לזו, כמו A ו-E, או M ו-N, כמות נתונים נמוכה, מקשה על האפשרות לזהות בצורה מיטבית אותיות אלה.

סיבות אלה הובילו אותי למצב שבו אשקול לעשות מספר אותיות מצומצם, ולא את כל 24 האותיות.

חשש נוסף בפרויקט הוא שסוגי ידיים שונים לא יזוהו בצורה מיטבית, כיוון שהרבה מה-datasets הם אותן ידיים של איש אחד, ולכן קיימת אפשרות שלמודל יהיה קשה לזהות יד בצורה קצת שונה או עם גודל אחר.

הדרך שעלתה בידיי להתמודדות עם קושי זה, היא שילוב של כמה datasets שונים, על אף הקשיים והסיכון שהתמונות לא יתאימו ויקשו על המודל ללמוד. השילוב יהווה דרך לניסיון פתרון של קשיים אלה.

הוראות התקנה

ההתקנה לספריות שהשתמשתי בהן בפרוייקט תעשה באופן הבא:

התקנה באמצעות	שם הספרייה וגרסא
Pip install tensorflow	Tensorflow2.8.0))
Pip install keras	Keras (2.8.0)
Pip install numpy	Numpy (1.20.3)
pip install Pillow	PIL (8.2.0)
Pip install matplotlib	Matplotlib (3.4.3)
Pip install opencv—contrib-python	opencv-contrib-python)
Pip install tk	Tkinter (8.6.10)

מבנה

איסוף והכנת נתונים

המידע והתמונות נלקחו מאתר Kaggle. באתר משתמשים שונים מעלים קוד, נתונים, ומשתפים פעולה במטרה לפתור בעיות.

חיפשתי באתר ומצאתי מספר אנשים שונים שהעלו נתונים בנושא אותיות בשפת הסימנים, אך ברוב המקורות הופיעו תמונות שונות מאוד זו מזו.

בדוגמה להלן, בתמונה השמאלית היד נמצאת על כל המסך, ולעומתה בתמונה הימנית האגרוף מהווה חלק מהתמונה הכוללת, והפנים של האיש מהווה חלק משמעותי יותר בתמונה. מצב זה לא קיים בתמונה השמאלית.

ההבדלים הללו מקשים על הזיהוי, במיוחד מהסיבה שמספר התמונות אינו גדול, ולכן מצב זה אילץ אותי לעבור על כל המקורות שמצאתי ולבדוק איזה מהתמונות מתאימות זו לזו ועל איזה יהיה עליי לוותר.

תמונות רבות, בחלק מהמקורות, היו דומות מאוד אחת לשנייה, ולכן במקום לקחת את כל התמונות לקחתי רק חלק מהתמונות בכל dataset, ומחקתי באקראיות חלק מהתמונות בכדי להקטין את כמות התמונות שהן כמעט זהות זו לזו ולנסות למנוע overfitting.

כיוון ששילבתי בין כמה מקורות, הגודל של התמונות היה שונה. שיניתי את גודל התמונות ל-128X128 בעזרת cv2.

מתוך שני ה-datasets, מחקתי תמונות שהיו דומות מאוד אחת לשנייה והגעתי לכמות סופית של 2840 תמונות לכל אות.

מקור התמונות נמצא בקישורים הבאים:

https://www.kaggle.com/datasets/grassknoted/asl-alphabet

https://www.kaggle.com/datasets/mrgeislinger/asl-rgb-depth-fingerspelling-spelling-it-out

כחלק מתהליך הכנת הנתונים לאימון, השתמשתי ב- PIL.Image

כדי להוסיף לכמות התמונות וליצור גיוון בסוג התמונות, יצרתי Module שתפקידו לשנות דברים שונים בתמונות לשימושים שונים.

בחרתי באקראיות תמונות לעשות עליהן אוגמנטצייה-

150% בהירות

20 מעלות ימינה

20 מעלות שמאלה

כך העלתי את הכמות תמונות שלי מ-2820 ל-6650 לכל אות.

לפני הרצת המודל, אנו מבצעים נרמול לנתונים.

נרמול נתונים הוא התהליך של להפוך את כל הפיקסלים בתמונה לערכים בין 1-0, כש-0 הוא הערך הקטן ביותר ו-1 הוא הגדול ביותר. זה משפר את יכולת האימון של המודל, ונותן לו לעבוד עם טווח קטן יותר של מספרים.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

יש כמה שיטות לנרמול, אך השיטה שאנו משתמשים בה משתמשת בנוסחה הבאה-

במקרה שלנו, כיוון שהפיקסל המינימלי הוא 0 והגבוה ביותר ההוא 255, אנו יודעים שזה הטווח של המספרים כך שבפועל אנו מחלקים את כל ה dataset-ב- 255 כיוון ש -

$$\frac{x - 0}{255 - 0} = \frac{x}{255}$$

כך נקבל את התמונות בטווח של 1-0 כשפיקסל של 255 הוא 1 ופיקסל של 0 הוא 0, וכל השאר נמצאים ביניהם.

שלב בנייה ואימון המודל

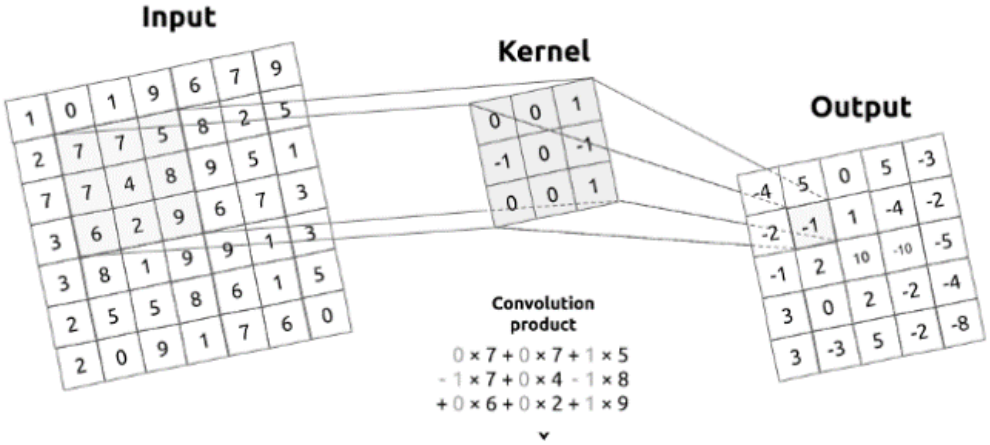
תיאור גרפי של המודל-

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 128, 32)      320

 max_pooling2d (MaxPooling2D  (None, 64, 64, 32)       0
 )

 dropout (Dropout)           (None, 64, 64, 32)        0

 conv2d_1 (Conv2D)           (None, 64, 64, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 32, 32, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 32, 32, 64)        0

 conv2d_2 (Conv2D)           (None, 32, 32, 64)        36928

 dropout_2 (Dropout)         (None, 32, 32, 64)        0

 conv2d_3 (Conv2D)           (None, 32, 32, 128)       73856

 flatten (Flatten)           (None, 131072)            0

 dense (Dense)               (None, 128)               16777344

 activation (Activation)     (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 activation_1 (Activation)   (None, 64)                0

 dense_2 (Dense)             (None, 32)                2080

 activation_2 (Activation)   (None, 32)                0

 dense_3 (Dense)             (None, 16)                528

 activation_3 (Activation)   (None, 16)                0

 dense_4 (Dense)             (None, 4)                 68
```

הסבר על סוגי השכבות-

Conv2d(size=(128,128),filter=32, kernel_size = (3,3))
Output: same size

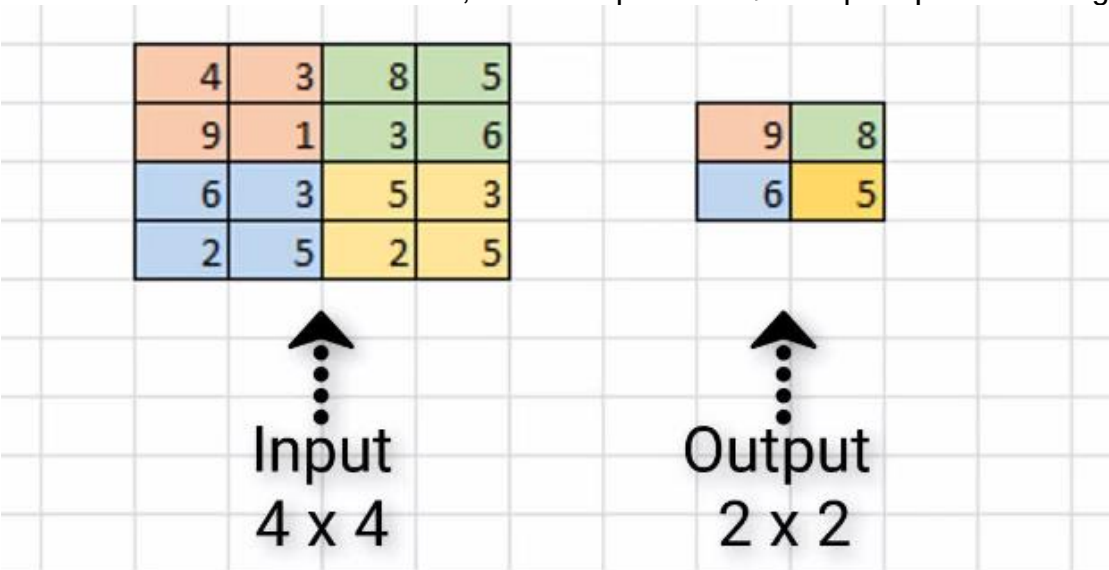
השכבה משתמשת בגודל kernel של שלוש על שלוש ועוברת על התמונה.
בפועל, השכבה לוקחת כל קופסא של פיקסלים שעוברת עליה ומכפילה אותה בערכים שונים, ובכך לוקחת מכל קופסא של (3,3) במקרה הזה ערכים מסוימים ומחדדת את התמונה.



לדוגמה, ניתן לראות כי בריבוע שנלקח זה מכפיל את כל המספרים בערך מסוים, ומגיע לבסוף לתוצאה חדשה.

MaxPooling2D(pool_size = (2,2), strides = (2,2)
Output: (64x64)

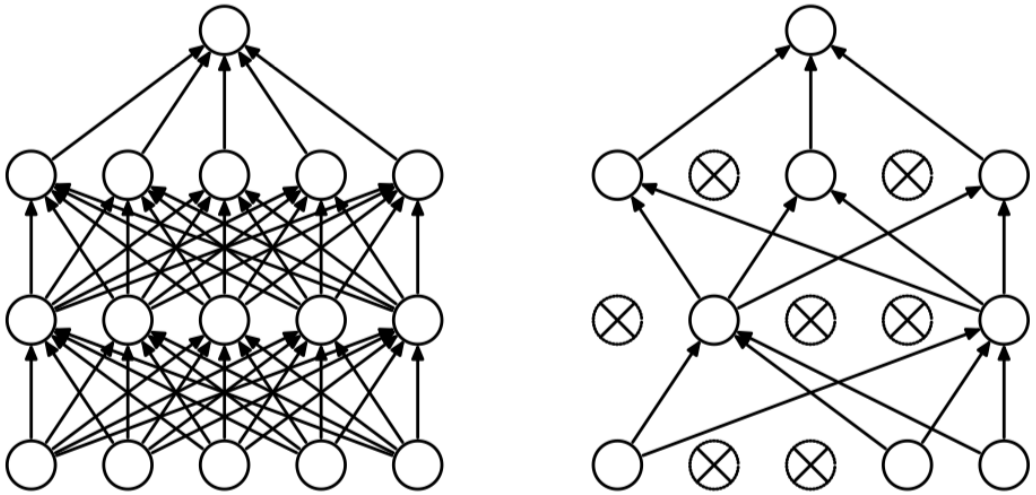
השכבה עוברת על התמונה שהיא בגודל (128X128) בשלב הזה ועוברת עליה עם קופסא בגודל (2x2) בקפיצות באותו גודל ולוקחת מכל ארבעה פיקסלים רק את הגדול ביותר.
שלב זה עוזר להקטין את גודל התמונה ולהקל על האימון, בנוסף הוא עוזר להילחם ב-overfitting כיוון שלוקח מידע יותר מדויק מכל תמונה, ושם לב ליותר פינות ושינויים.



בדוגמה הזו, כל צבע מייצג קופסא של 2x2. ניתן לראות ב-output שנותר רק המספר הגדול ביותר בכל קופסא.

self.model.add(Dropout(rate = 0.3))

השכבה הופכת באקראיות ערכים של השכבות ל-0 במהלך האימון, ובכך להסיר אותו מהרשת זמנית.
גם שלב זה עוזר להקטין אפשרות של overfitting, כיוון שהנוירונים יהיו מאומנים פחות על הנתונים הספציפיים האלה, ונותר יותר מקום ללמידה.



(a) Standard Neural Net (b) After applying dropout.

בדוגמה רואים כי שנוירונים שבמצב רגיל היו מתקשרים עם נוירונים אחרים, לא יעבדו כשה-dropout פעיל.

על שלושת השכבות האלה אנו חוזרים עוד פעם אחת :
Conv2d(size=(64,64),filter=64, kernel_size = (3,3))
Output: same size
MaxPooling2D(pool_size = (2,2), strides = (2,2)
Output: (32x32)
self.model.add(Dropout(rate = 0.3))
ולאחר מכן אנו מבצעים:
Conv2d(size=(32,32),filter=64, kernel_size = (3,3))

Output: same size
self.model.add(Dropout(rate = 0.3))

Conv2d(size=(32,32),filter=64, kernel_size = (3,3))
Output:same size
כל השכבות עד עכשיו נועדו לחדד את התמונות ולהדגיש חלקים חשובים בתמונות שיעזרו לנו לזהות את המודל ועזרו להקטין אפשרות של overfitting.

Flatten()

השכבות הבאות במודל הן dense, שכבה שה-input_size שלה הוא מספר אחד, ולכן כל האלמנטים שבשכבה הקודמת צריכים להיות במערך חד ממדי. השכבה הזו לוקחת את ה-output מהשכבות הקודמות והופכת אותו למערך חד ממדי.

לפני flatten ←

אחרי flatten ←

Dense(128)

Activation('relu')

זו השכבה השכיחה ביותר בשכבות נוירונים, ותפקידה הוא פשוט מאוד. היא מעבירה מטריצה על התמונה ומכפילה אותה בערכים האלו, שהם המשקלים, מפעילה את ה-activation function על התמונה ומעבירה את הערכים לשכבה הבאה. המספר מייצג את ה-output שיצא מהשכבה.

שכבת ה-Activation קובעת את ה-activation function שיופעל על השכבה, במקרה זה השתמשתי ב-relu עליו אפרט בהמשך.

על השכבה הזו חזרתי 3 פעמים באותו אופן, ורק ה-output השתנה.

Dense(64)

Activation('relu')

Dense(32)

Activation('relu')

Dense(16)

Activation('relu')

לבסוף, השתמשתי באותה שכבה עם 'activation ='softmax

Dense(4, activation = "softmax")

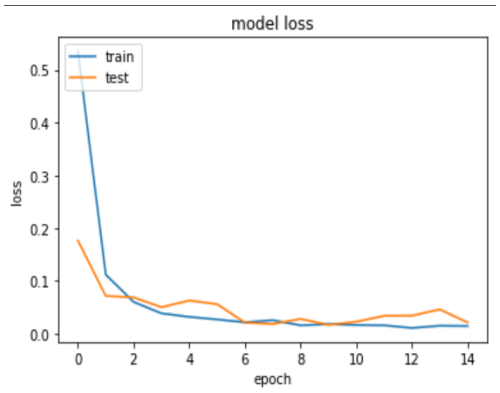
Activation function זה ממיר את ה-input לחלוקת הסתברויות על פי מספר המחלקות שניתנו לו, במקרה הזה 4, ולכן מתאים לבעיות classification כיוון שניתן לקחת ממנו את אחוזי ההצלחה של זיהוי כל class.

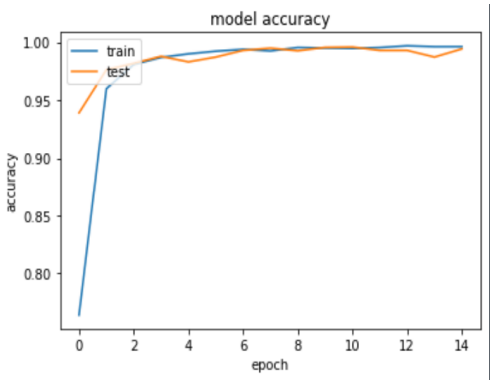
בחלק האימון יצרתי מחלקה אחת, שמסודרת להלן:

	Class MODEL
רשימת האותיות אותן ננסה לזהות	self.capitals
התאמה בין האותיות לבין מספר בין0-3	self.match_labels
המודל עליו נאמן את התמונות	self.model
התמונות בהן נשתמש לאימון	self.x_train
תווית תמונות האימון	self.y_train
התמונות בהן נשתמש לבדיקה	self.x_test
תווית תמונות הבדיקה	self.y_test
ממיר את האות אל מערך של אפסים כשבמקום של המספר המתאים לאות הנכונה יש 1	def convert_into_category(self,character)
טוען את התמונות והתוויות שלהן	def load_dataset(self,path)
בניית המודל והשכבות השונות בו	def build_model(self)
מחלק את התמונות לtrain and test, מנרמל את התמונות ומסדר את המערך בצורה הנכונה	def split(self, data, labels)
מאמן את המודל ומחזיר את תוצאותיו. בנוסף שומר את המודל	def train_model(self)
בודק על ה-test את המודל שאומן	def test_images(self)

דוחות וגרפים:

המודל הראשוני שהצגתי הניב את התוצאות הבאות-





ניתן לראות בגרפים כי ה-accuracy עולה מהר מאוד, ונשאר קרוב ל-1 כמעט כל התהליך, וכנל ל-loss שנמצא כמעט על 0. תוצאות אלה הן סימן ל-overfitting.

נכון לרגע זה, לא הצלחתי לייצר תוצאות יותר טובות מאלה עם התמונות שנמצאות בידי.

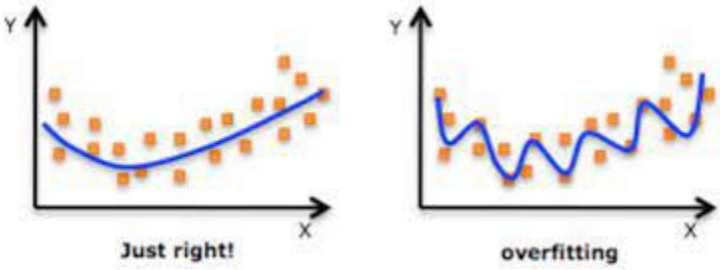
Overfitting נגרם מחוסר של תמונות שונות זו מזו.

כאשר כל התמונות דומות מאוד או לחילופין, אין מספיק תמונות לאמן עליהן, המודל לומד לזהות רק את התמונות עליהן מאמנים, במקום ללמוד לזהות את הנושא עליו מאמנים.

כאשר מצב זה נגרם, המודל שאימנו מכיר רק את התמונות שניתנו לו, אך לא מזהה טוב תמונות חיצוניות, מה שמביס את מטרת המודל.

קיימים מספר פתרונות אפשריים, רבים מהם ניסיתי ללא הצלחה.

מספר התמונות בתחום זה מצומצם מאוד, וגם הנתונים שקיימים מאוד שונים זה מזה ובכמות מאוד קטנה כך שלא מצליח לזהות בצורה מיטבית, ושילוב בין המקורות מקשה על המודל לאמן והתוצאות שמתקבלות טובות אף פחות.



לאחר מכן הוספתי שכבות dropout שאמורות להקטין את ה-overfitting, והפסקתי לאמן את המודל גם על האותיות A כיוון שהצורה שלה גנרית, ולמודל היה קשה להבדיל בינה לבין אותיות אחרות כיוון שלא היו לה מאפיינים ייחודיים.

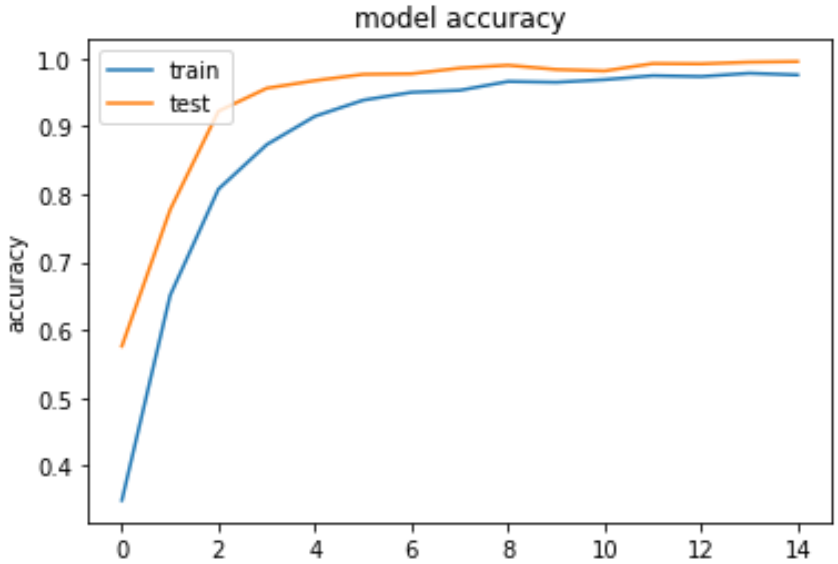
האות A

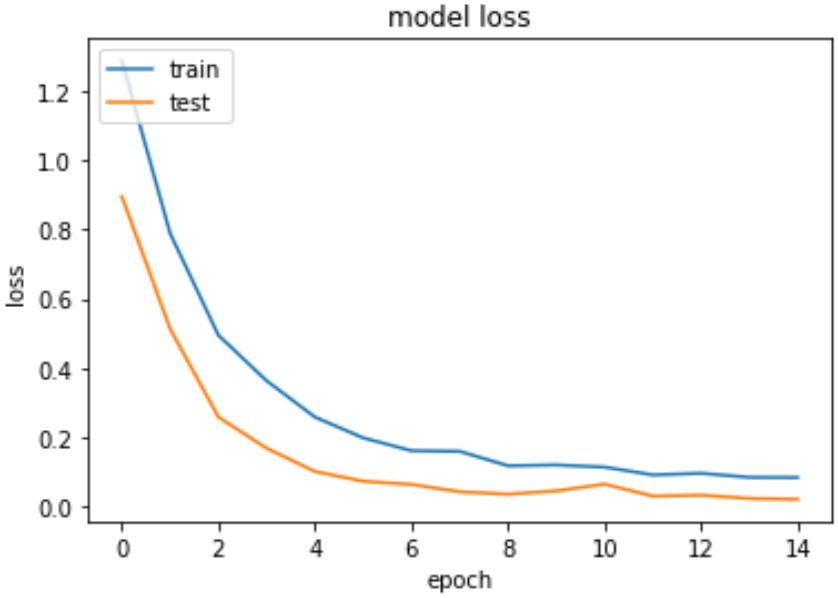
האות I

לדוגמה, ניתן לראות שהאות I בעצם זהה ל-A, מלבד הזרת שמורמת. הבדלים קטנים אלו מאפשרים לנו להבדיל ביניהן, אך למודל שמאומן על מספר תמונות כה קטן קשה לזהות את האות A.

לאחר השינויים האלה, האותיות הסופיות שהמודל אומן עליהן הן I,F,H,B.

כשהרצתי את המודל מחדש-





כבר לפי הגרפים ניתן לראות שהתוצאות יותר טובות, המודל לא מזנק כבר ב-epoch הראשון ל-accuracy מעל 0.9, אלא מגיע לשם בהדרגה וגם ה-loss יורד במגמה יותר הדרגתית מהתוצאות הקודמות.

Hyperparamters-

Hyperparameters Related To The Training Algorithm	Hyperparameters Related To Neural Network Structure
Learning rate = 0.001	Number of hidden layers = 12
Epoch = 20, iterations and batch size = 32	Dropout = 0.3
Optimizer algorithm	Activation function = relu, softmax
Momentum	Weights initialization = random

מעקב אחרי שינוי -

כשהתחלתי את העבודה על המודל, חיפשתי איזה שכבות יתאימו לפרויקט שלי.

ניסיתי בהתחלה להריץ עם כמה שכבות פשוטות –

```
self.model.add(Flatten())

self.model.add(Dense(64, activation='sigmoid'))
self.model.add(Dense(32, activation='sigmoid'))
self.model.add(Dense(16, activation='sigmoid'))
self.model.add(Dense(8, activation='sigmoid'))
self.model.add(Dense(4, activation='softmax'))
self.model.add(Conv2D(filters = 128, kernel_size = (3,3),padding = 'Same',activation ='relu', input_shape = (128,128,1)))
self.model.add(MaxPooling2D(pool_size = (2,2), strides = (2,2)))
self.model.add(Dropout(0.3))
```

אך כמובן שהמודל לא עבד בצורה טובה, במיוחד בגלל בעיית ה-overfitting.

לאחר מכן הוספתי עוד שכבות שאמורות לעזור במקרה כזה –

```
self.model.add(Conv2D(32,(5,5), input_shape=(128,128, 1), activation='sigmoid'))
self.model.add(MaxPooling2D(pool_size=(2, 2)))

self.model.add(Conv2D(32,(5,5), activation='relu'))
self.model.add(MaxPooling2D(pool_size=(2, 2)))

self.model.add(Flatten())
self.model.add(Dense(1024, activation='sigmoid'))
self.model.add(Dense(512, activation='sigmoid'))
self.model.add(Dense(256, activation='sigmoid'))
self.model.add(Dense(128, activation='sigmoid'))
self.model.add(Dense(64, activation='sigmoid'))
self.model.add(Dense(32, activation='sigmoid'))
self.model.add(Dense(16, activation='sigmoid'))
self.model.add(Dense(8, activation='sigmoid'))
self.model.add(Dense(4, activation='softmax'))
```

גם כאן המודל לא השתפר בהרבה, ולכן הוספתי שכבות dropout שגם אמורות לעזור לבעיה.

```
self.model.add(Conv2D(32,(5,5), input_shape=(128,128, 1), activation='sigmoid'))
self.model.add(MaxPooling2D(pool_size=(2, 2)))
self.model.add(Dropout(0.3))

self.model.add(Conv2D(32,(5,5), activation='relu'))
self.model.add(MaxPooling2D(pool_size=(2, 2)))
self.model.add(Dropout(0.3))
```

גם עם ה-dropout, המודל לא זיהה חלק גדול מהתמונות החיצוניות שהבאתי לו.

```
self.model.add(Conv2D(32,(5,5), input_shape=(128,128, 1), activation='relu'))
self.model.add(MaxPooling2D(pool_size=(2, 2)))
self.model.add(Dropout(0.3))

self.model.add(Conv2D(32,(5,5), activation='relu'))
self.model.add(MaxPooling2D(pool_size=(2, 2)))
self.model.add(Dropout(0.3))

self.model.add(Flatten())

self.model.add(Dense(128, activation='relu'))
self.model.add(Dense(64, activation='relu'))
self.model.add(Dense(32, activation='relu'))
self.model.add(Dense(16, activation='relu'))
self.model.add(Dense(8, activation='relu'))
self.model.add(Dense(4, activation='softmax'))
```

בשלב האחרון, סידרתי את המודל והוספתי עוד שכבות conv2d ו-maxpooling וקיבלתי את המודל הסופי, שאמנם לא הציג תוצאות טובות, אך כן הראה שיפור מועט והמודל אף הצליח לזהות חלק מהאותיות בתמונות שיצרתי בעצמי.

```
self.model.add(Conv2D(filters = 128, kernel_size = (3,3),padding = 'Same',activation ='relu', input_shape = (128,128,1)))
self.model.add(MaxPooling2D(pool_size = (2,2), strides = (2,2)))
self.model.add(Dropout(0.3))

self.model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',activation ='relu' ))
self.model.add(MaxPooling2D(pool_size = (2,2), strides = (2,2)))
self.model.add(Dropout(0.3))

self.model.add(Conv2D(filters = 32, kernel_size = (3,3),padding = 'Same',activation ='relu' ))
self.model.add(Dropout(0.3))
self.model.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same',activation ='relu'))

self.model.add(Flatten())
self.model.add(Dense(128))
self.model.add(Dropout(0.5))

self.model.add(Activation('relu'))
self.model.add(Dense(64))
self.model.add(Dropout(0.5))

self.model.add(Activation('relu'))
self.model.add(Dense(32))
self.model.add(Dropout(0.5))

self.model.add(Activation('relu'))
self.model.add(Dense(16))
self.model.add(Activation('relu'))
self.model.add(Dense(4, activation = "softmax"))
```

פונקציית השגיאה

ב-deep learning, אנו מנסים למצוא את המשקלים שיביאו אותנו לתוצאות הנכונות ביותר למודל. אנו עושים זאת באמצעות אימון המודל, שוב ושוב, צעד קדימה וצעד אחורה.

מטרתנו היא להקטין את השגיאה כמה שיותר. כדי להעביר את השגיאה לערך מספרי, אנו משתמשים בפונקציית שגיאה, שמחשבת את השגיאה לפי החיזוי שהתבצע בפועל לעומת התוצאה האמתית.

אני השתמשתי בפונקציית השגיאה categorical_crossentropy

פונקציה זו מתאימה לבעיות classification בהן מנסים להבדיל בין 2 או יותר classes, ולכן מתאימה לפרויקט זה ובמיוחד מומלץ להשתמש בה עם activation function שהוא softmax.

התוויות של התמונות בפונקציית שגיאה זו צריכים להיות בצורת רשימה באופן הבא-

אם התמונה שייכת למחלקה 0, התווית שלה תיראה כך: [1,0,0,0]

אם התמונה שייכת למחלקה 1, התווית שלה תיראה כך: [0,1,0,0]

וכן הלאה..

החישוב של הפונקציה נעשה כך:

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

כאשר:

y_i = תוצאה האמתית של התמונה לאותה class, כלומר 0 בכל מקרה חוץ מהאחד הנכון בו יהיה 1

Output size= כמות ה-classes, במקרה הזה 4

$\hat{y}_i$ =תוצאה שהתקבלה מהמודל באותה תמונה

כיוון שאנו יודעים שהחיזויים הלא נכונים יוכפלו ב-0, והחיזוי הנכון באחד, אנו יודעים שהתוצאה הסופית של השגיאה תהיה $\log \hat{y}_i$

ייעול התכנסות

בתהליך האימון של המודל, אנו משנים את המשקלים של המודל על מנת לנסות ולהגיע ל-accuracy יותר גבוה ו-loss יותר נמוך.

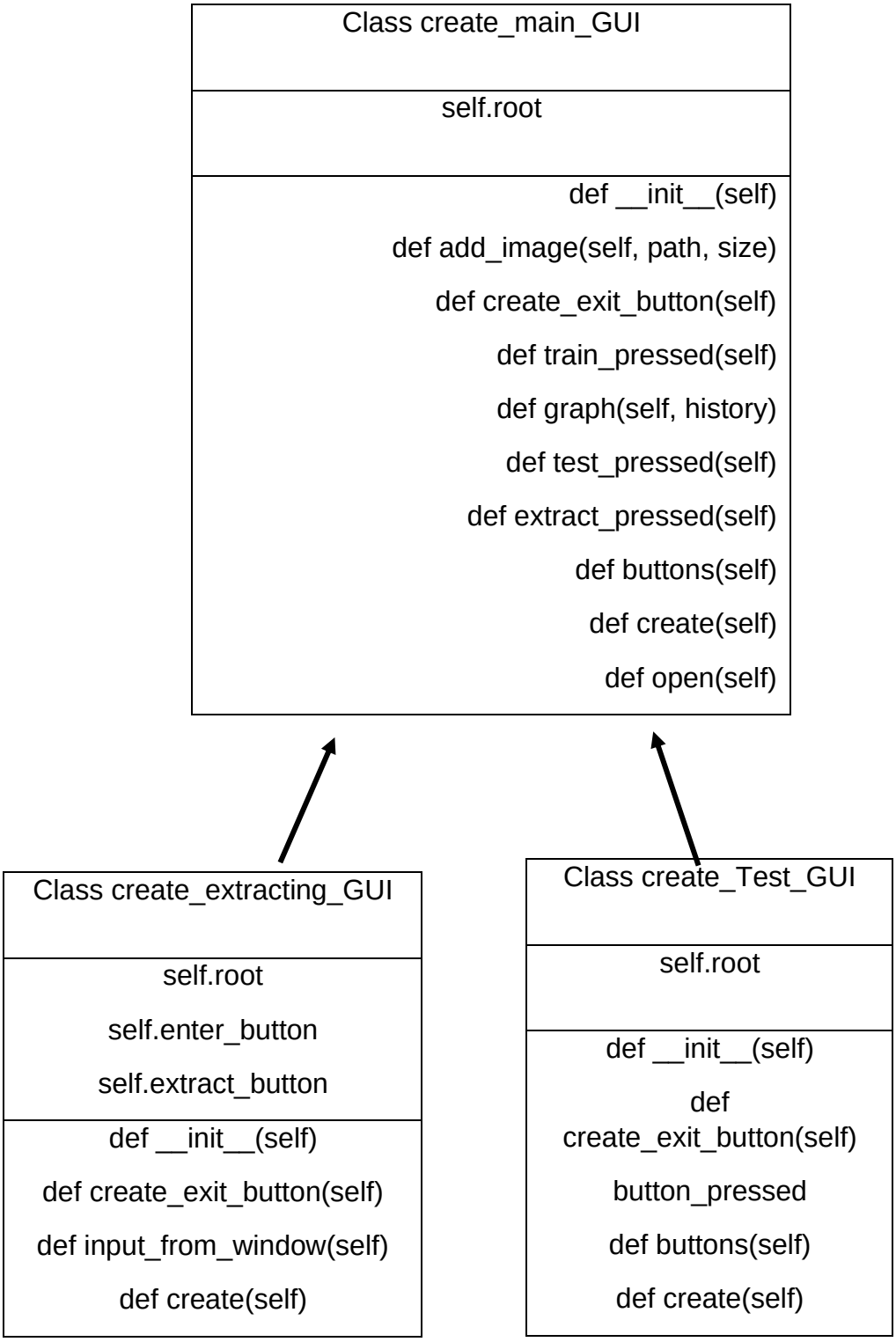
התהליך שאחראי על השיפור של המשקלים ומחליט בכמה לשנות ואיך, הוא ה-optimizer.

ישנם סוגים שונים של optimizers שמבצעים את הפעולה הזו בדרכים שונות, ולכן צריך לדעת לבחור את האחד הנכון למצב שנמצאים בו.

ה-optimizer שאני בחרתי בו הוא ADAM (adaptive moment estimation).

היתרון של ADAM הוא שהמשקלים הם לא הדבר היחיד שמשתנה בזמן האימון, הוא משנה גם את קצב הלמידה כדי להתאים למצב, ובכך משפר את מהירות האימון, אך מקטין את הדיוק אליו מגיעים. בנוסף, הוא פשוט ליישום ולא מכביד על הזיכרון.

שלב היישום


Class create_main_GUI
self.root
def __init__(self)
def add_image(self, path, size)
def create_exit_button(self)
def train_pressed(self)
def graph(self, history)
def test_pressed(self)
def extract_pressed(self)
def buttons(self)
def create(self)
def open(self)
Class create_extracting_GUI
self.root
self.enter_button
self.extract_button
def __init__(self)
def create_exit_button(self)
def input_from_window(self)
def create(self)
Class create_Test_GUI
self.root
def __init__(self)
def create_exit_button(self)
button_pressed
def buttons(self)
def create(self)

היישום מנצל את המודל בכמה דרכים –

1) אם המשתמש רוצה לאמן את המודל בעצמו, הוא ישתמש ב-Module של המודל שתואר למעלה בכדי לאמן את המודל עם התמונות שהתקבלו ב-extract.
2) אם המשתמש רוצה רק לעשות test, הוא יוכל להשתמש במודל השמור שקיים: My model בקבצים מהתיקייה.

היישום בנוי כולו ב-tkinter.

Tkinter היא הספרייה ליצירת GUI הפופולארית ביותר ב-Python. הספרייה נותנת ליצור חלונות בצורה פשוטה בשילוב עם טקסט, תמונות וכפתורים, ובנוסף מאפשרת קליטה של טקסט או תמונות.

כשהמשתמש לוחץ על הכפתור שמאמן את המודל, מתבצעת טעינה של התמונות בעזרת –

tensorflow.keras.preprocessing.image.load_img

tensorflow.keras.preprocessing.image.img_to_array

התמונות נטענות ומומרות למערך מסוג numpy array

הפעולה load_dataset עוברת על כל התמונות בתיקייה ומוסיפה אותן לרשימה אחת, ובו זמנית יוצרת רשימה של התוויות של כל תמונה.

מדריך למפתח

קובץ ראשון- מסך למשתמש:

קובץ זה אחראי על כל המסכים שהמשתמש רואה בעת ההרצה, ועל הרצת הפעולות הנכונות בעת לחיצה על הכפתורים. הקובץ מחולק לשלוש מחלקות שתוארו למעלה, ואפרט עליהן כעת.

Class create_main_GUI

מחלקה זו אחראית על יצירת מסך הפתיחה הראשי שמופיע, פתיחת מסכים נוספים לפי הצורך, ומשתמשת במחלקה Use_Model בכדי לאמן את המודל.

הקובץ מחולק לפעולות הבאות-

1) def add_image(self, path, size)

מוסיף את התמונה שמופיעה במסך הבית.

משתנים:

Im- מכיל את התמונה שנפתחת באמצעות PIL Image

Ph- מכיל את התמונה בצורה שיכולה להיות מיושמת בחלון ה-tkinter

Image_label- המשתנה שמאפשר לנו להציג את התמונה על המסך, ובעזרתו אנו קובעים את מיקומה.

2) def create_exit_button(self)

יוצרת את כפתור היציאה שסוגר את המסך ויופיע בתחתית

משתנים:

exit_button - יכיל את המופע של הכפתור.

3) def create(self)

יצור את החלון ובו נסדר את כל ההגדרות של החלון- גודל, מספר שורות ועמודות וכותרת.

4) def buttons(self)

יצור את שלושת הכפתורים שמופיעים במסך הפתיחה - test, extract, train

משתנים:

Test_button- יכיל את כפתור הטסט

Extract_button- יכיל את כפתור החילוץ

Train_button- משתנה של המחלקה, שיהיה כבוי עד שכפתור ה-extract ילחץ וקבצי ה-train יהיו על המחשב.

5) def extract_pressed(self)

יקרא כאשר כפתור החילוץ נלחץ. יפתח את כפתורי האימון והבדיקה לשימוש, ויצור מופע של המחלקה שמייצרת את חלון החילוץ.

6) def test_pressed(self)

יקרא כאשר כפתור ה-test נלחץ. יצור מופע של המחלקה שמייצרת את חלון ה-test ויצור אותו.

7) def train_pressed(self)

יקרא כאשר כפתור ה-train נלחץ. סוגר את החלון הראשי עד סוף תהליך האימון, יציג 2 הודעות בעת האימון- אחת כשמתחיל לטעון את התמונות, והשנייה כשמתחיל לאמן את המודל. לאחר מכן יקרא לפונקציה הבאה שתייצר גרפים מהתוצאות.

משתנים:

History- יכיל את המידע אודות תהליך האימון.

8) def graph(self, history)

יקבל את המידע על האימון מהפונקצייה הקודמת. יציג 2 גרפים- אחד של השגיאה ואחד של הדיוק לאורך ה-epoch במודל שאומן.

9) def open(self)

לבסוף, הפונקציה תפתח את החלון לאחר שכל הכפתורים והתמונה סודרו והחלון מוכן.

<u>Class create_Test_GUI</u>

המחלקה אחראית על ייצור החלון ממנו נעשה בדיקה לתמונות שהמשתמש יעלה.

הוא מחולק לכמה פעולות –

1) def create_exit_button(self)
יוצרת את כפתור היציאה שסוגר את המסך ויופיע בתחתית
משתנים:

exit_button - יכיל את המופע של הכפתור.

2) def create(self)

הפוקנצייה תיצור את החלון ואת כל ההגדרות שלו, בנוסף תקרא לפעולה הקודמת ותיצור את כפתור ה-test.

משתנים-

Testing- יכיל את המופע של הכפתור שממנו נעלה תמונות, איתו נמקם את הכפתור במסך.

3) def button_pressed(self)

הפונקצייה תיקרא כאשר כפתור ה-test ילחץ.

היא תטען את המודל המאומן, ותבקש מהמשתמש לבחור תמונה עליה נבדוק את המודל.

היא תטען את התמונה ותריץ עליה את המודל, לבסוף תציג הודעה של איזה תווית זיהתה ובכמה אחוזים.

משתנים-

Model- יכיל את המודל שנטען.

Filename- יכיל את הכתובת של התמונה שבחר המשתמש.

Capitals- רשימה של האותיות שהמודל מזהה.

Img- התמונה שהמשתמש בחר.

Predictions – תוצאות המודל על התמונה.

Predicted – האות אותה המודל זיהה בתמונה.

Confidence- אחוז הביטחון של המודל שהאות היא התמונה.

Class create_extracting_GUI

המחלקה שאחראית על יצירת ויישום חלון החילוץ שיוציא את הקבצים של הפרויקט מקובץ zip.

1) def create_exit_button(self)

יוצרת את כפתור היציאה שסוגר את המסך ויופיע בתחתית

משתנים:

exit_button - יכיל את המופע של הכפתור.

2) def create(self)

ייצר את החלון ויגדיר את הגודל והכפתורים שבו, וגם את הכותרת וכמות העמודות והשורות ולבסוף יפתח את החלון.

3) def input_from_window(self)

יתן למשתמש לבחור קובץ zip ממנו לחלץ קבצים, ויחלץ אותם אלא אם כן כבר קיימת התיקייה עם השם הזה, כלומר הקבצים כבר חולצו.

משתנים:

Path- יכיל את הכתובת של קובץ ה-zip לאחר שהמשתמש בוחר קובץ.

קובץ שני – המודל

המחלקה הזו מכילה את כל המודל, טעינה של התמונות מהתיקייה וה-train, test.

הוא מכיל class אחד- Use_Model שבאמצעותה ניתן מקבצים אחרים להשתמש במודל ולאמן אותו.

Use_Model

המחלקה מכילה כמה משתנים שצריך לאורך המחלקה בפעולות שונות ששימושם תואר בחלק המבנה ולכן לא אפרט עליהם שוב, רק אפרט יותר על הפעולות שבמחלקה.

1) def convert_into_category(self,character)

הפעולה תקבל אות מבין האותיות שהמודל יכול לזהות ותמיר אותו למערך שמתאים לבעיות classification. כלומר, המערך יראה בצורה הבאה-

אם התווית המתאימה לתמונה היא A, המערך יהיה [1,0,0,0]

אם התווית המתאימה היא B, המערך שיוחזר הוא [0,1,0,0] וכן הלאה.

2) def load_dataset(self,path)

המחלקה אחראית על הטעינה של כל התמונות והעברת כל המערכים ל-numpy array.

היא עוברת על כל תיקייה בקובץ ה-dataset, טוענת כל תמונה ומכניסה אותו לרשימה ומכניסה את התווית שלו לרשימה אחרת(בעזרת הפעולה הקודמת) כך שכל תווית ברשימת התוויות מתאימה במיקומה לתווית ברשימת התמונות. בסוף הפעולה המערכים מומרים ל-numpy array ומוחזרים.

משתנים:

Data – רשימת התמונות

Labels- רשימת התוויות

Image- התמונה הנוכחית שמוספת בלולאה.

3) def build_model(self)

פעולה זו בונה את המודל- השכבות שלו, פונקציית השגיאה וייעול ההתכנסות. בנוסף הפעולה מדפיסה את ה-summary של המודל, שהוצג למעלה בחלק המבנה ומציג את ה-output של כל שכבה והפרמטרים בה.

4) def split(self, data, labels)

הפעולה מקבלת את ה-data וה-labels שהתקבלו בפעולה load_dataset ומחלקת אותם ל-train ו-test, כש-90% הולך ל-train ו-10% ל-test.

לאחר החלוקה, משתנה הצורה של המערכים וסוגם הופך ל-numpy array גם, ומתבצע הנרמול (חילוק כל המספרים ב-255).

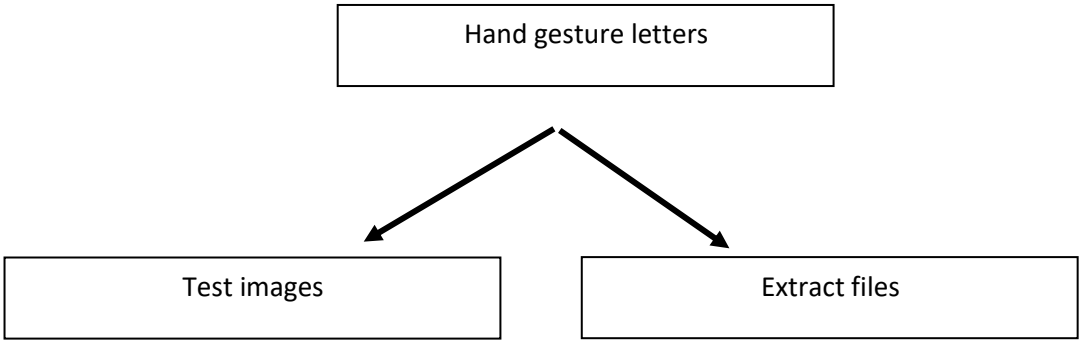
5) def train_model(self)

הפעולה מאמנת את המודל ושומרת את התוצאות במשתנה history. בנוסף היא שומרת את המודל בתיקייה בה נמצא הקוד, למקרה ונרצה להשתמש בו אחר כך. בסוף היא מחזירה את history.

6) def test_images(self)

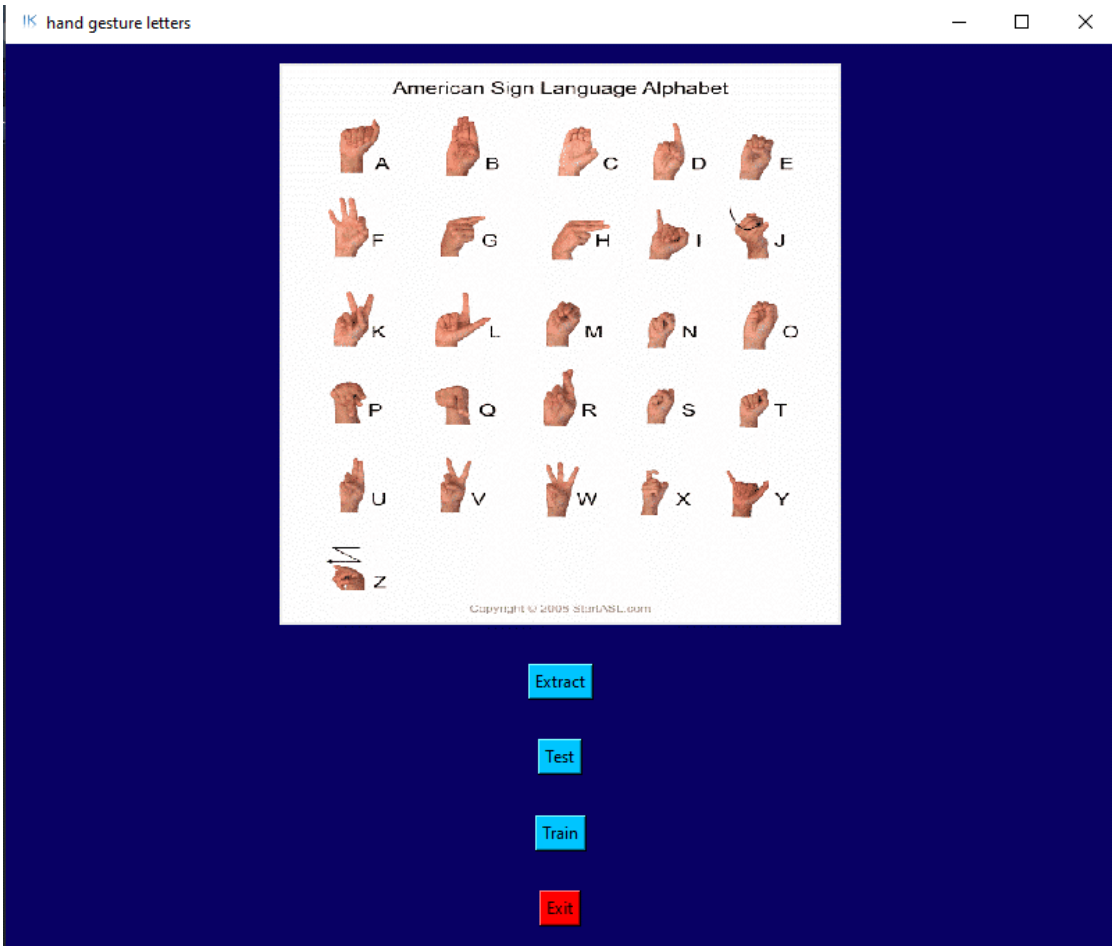
הפעולה משתמשת במודל שאומן ובודקת את השגיאה והדיוק על תמונות ה-test, ומחזירה את התוצאות.

מדריך למשתמש



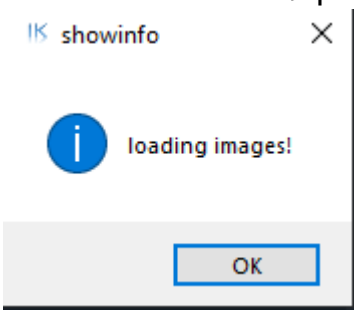
מסך הפתיחה:

מסך הפתיחה מכיל ארבעה כפתורים-

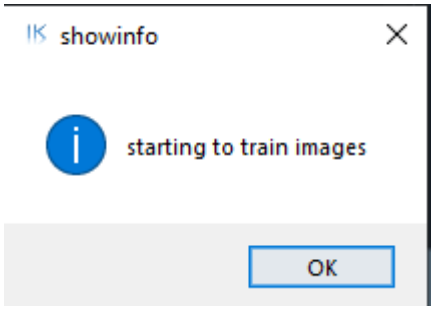


1) Extract - פותח את המסך ממנו נעשה extract לקבצים, פירוט בהמשך.

2) Test - פותח את המסך ממנו נעשה בדיקה על תמונות שיכניס המשתמש, פירוט בהמשך.
3) Train – הכפתור לא זמין עד אחרי שמשתמשים בכפתור ה-extract. לאחר שכל



הקבצים חולצו מה-zip, למשתמש יש את האופציה לאמן את המודל בעצמו. לאחר לחיצה על הכפתור, המסך הראשי יעלם עד השלמת האימון. בזמן זה יופיעו שתי הודעות.
הראשונה תיידע את המשתמש שהתמונות נמצאו ונטענות-

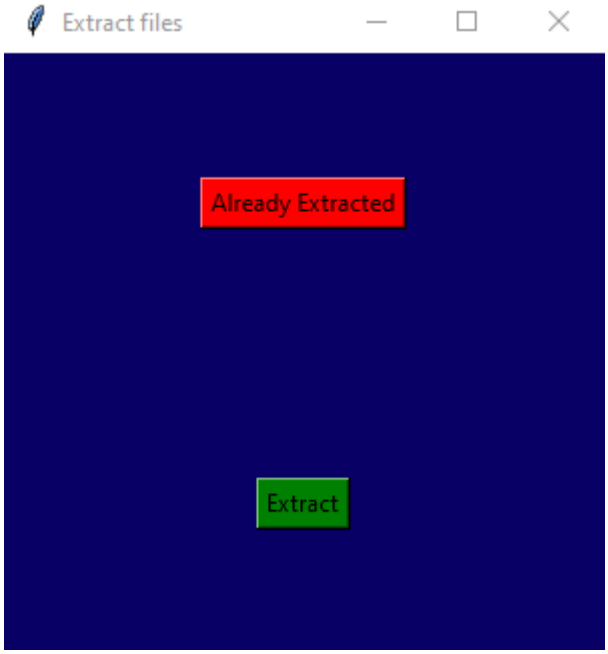


השנייה תיידע את המשתמש שהאימון של המודל מתחיל-

כשהאימון יסתיים, הגרפים של ה-accuracy וה-loss יוצגו ב-plots.

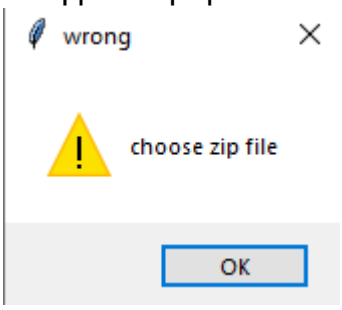
4) כפתור יציאה שסוגר את המסך.

מסך ה-Extract:

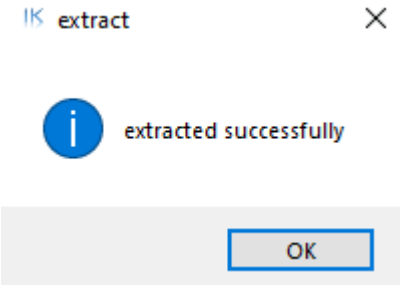


המסך מכיל 2 כפתורים.

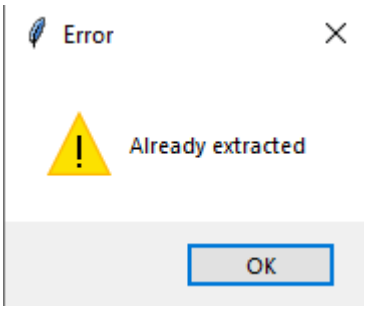
1) הכפתור הראשון הוא ה-Extract, שיתן למשתמש לבחור קובץ zip ויחלץ את הקבצים שבו לתיקייה בה נמצא הקוד.
אם יבחר קובץ לא תקין תופיע השגיאה-



לאחר שהקבצים מחולצים תופיע ההודעה –



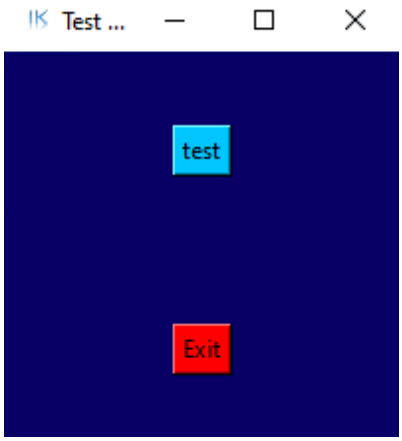
אם המשתמש ינסה לחלץ קבצים לאחר שהם כבר חולצו, תופיע האזהרה-



2) כפתור ה-Already Extracted שיסגור את המסך, למקרה שהמשתמש כבר חילץ את הקבצים ולחץ על הכפתור רק בשביל להפעיל את הכפתורים במסך הראשי.

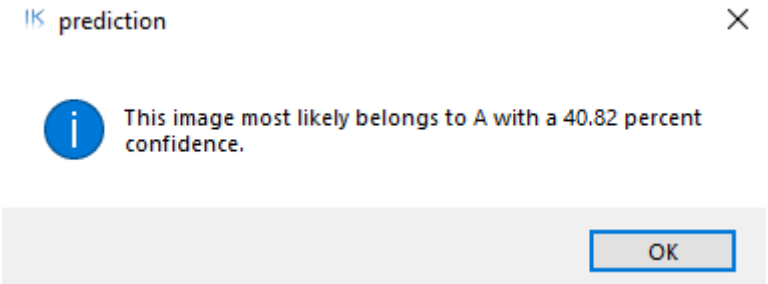
התוכנית מניחה שהמשתמש לא ינסה ללחוץ על הכפתורים האחרים בלי שיחלץ את הקבצים, וגם שהקבצים בתיקייה לא ימחקו והתיקייה תישאר אם התוכנית רצה שוב.

מסך ה- **Test**:



במסך זה מופיעים שני כפתורים-

1) כפתור ה-test ייתן למשתמש לבחור תמונה מהמחשב ולהריץ אותה על המודל הקיים שנטען כשנפתח המסך. תקפוץ הודעה שתרשום את התווית של האות ואחוז הביטחון של המודל.



לדוגמה-

התמונה זוהתה בתור האות A עם ביטחון של 40.82%.

2) כפתור ה-exit יסגור את המסך.

רפלקציה

הבחירה במגמת מדעי המחשב הייתה הבחירה הטבעית עבורי. לאורך השנים הבחנתי כי פעולות שנתפשות עבורי כפשוטות, הן למעשה מורכבות מאוד עבור אחרים. יחד עם זאת, בתחילת השנה, כשהתחלנו ללמוד את הנושא עליו נבסס את הפרויקט, נתקלתי בתחושה של בהלה. הבנתי את חשיבות הפרויקט ויחד עם העומס, הלמידה הרגישה לי מסובכת למדי, ומצב זה הוביל אותי לחשש שלא אצליח להתמודד עם המשימה. ככל שהמשכנו ללמוד, הפעולות המצופות ממני לאורך השנה התחילו להתחבר, והחשש הוסר, אולם לא במלואו, מפני שמעולם לא ניגשתי לפרויקט בסדר גודל כזה.

התהליך שעברתי במהלך השנה היה כל כך משמעותי מבחינתי, הן מבחינה רגשית והן מבחינת הלמידה. הליווי של המורה לאורך הדרך הפחית גם הוא את מידת החששות, ובמבט לאחור, אני מרגיש שהתמודדתי עם המשימה בצורה טובה. מבחינה לימודית – השקעתי מאמץ רב על מנת שהפרויקט יהיה באיכות מקסימלית, אני מודע לעובדה שהתוצאה הסופית לא מושלמת, אך יחד עם זאת, אני מרגיש שהשקעתי את מיטב יכולותיי כדי להביא למצב שבו הפרויקט יעבוד.

תכנון נכון יותר של זמני הלמידה היה מפחית את הלחץ שנוצר במהלך השנה, אך זו גם חלק מהלמידה עבורי – דחייה של חלקים מהפרויקט לסוף השנה הביא למצב של עומס מיותר בחודשיים האחרונים.

הנושא של למידת מכונה מאז ומתמיד היה מסקרן בעבורי, אך הרגיש מרוחק ועתידי. כיום, אני מרגיש שאני מבין קצת יותר על עולם תוכן שמעניין אותי מאוד והכל מרגיש יותר אמיתי.

האתגר הראשי שעמד בפניי הוא הנושא של התמונות. נכון לרגע זה, לא הצלחתי למצוא כמות תמונות מספקת בכדי לאמן את המודל בצורה שלא תגרום ל-overfitting. כל המקורות שמצאתי שיצרו תמונות על נושא זה היו מאוד שונים זה מזה, כמעט כולם במרחקים שונים מהמצלמה, בגדלים שונים, ומכילים יותר או פחות אלמנטים אחרים שעלולים להשפיע על הלמידה. בנוסף, כמעט כולם הכילו כמות גדולה מאוד של תמונות דומות מאוד זו לזו, שילבתי בין כמה מקורות שהיו הדומים ביותר, והסרתי מתוכם אלפי תמונות שהיו דומות לאחרות.

במהלך הניסיונות לאמן את המודל הוספתי תמונות, הורדתי תמונות, שיניתי את מספר האותיות שאני מאמן את המודל עליו, שילבתי בין datasets שונים, עשיתי אוגמנטצייה לתמונות ונקטתי בשלבים שמטרתם להקטין את הבעיה כמו שכבות ה-dropout או conv2d, אך לבסוף עדיין לא הצלחתי לשפר הרבה.

למרות שהמודל כן מצליח לזהות חלק מהאותיות, הוא עדיין נכשל במספר רב של מקרים.

לסיום, אני מרגיש ששנה זו הייתה המשמעותית ביותר בלימודי המגמה. לימוד דרך הגשת פרויקט מביא אותי למצב שבו אני חוקר, מתנסה, טועה, מצליח וכל אלה מביאים לרגשות משתנים ובעיקר ללמידה איך מתמודדים עם רגשות אלה. במהלך השנה רכשתי כלים וכישורים שאני בטוח אשתמש בהם בהמשך, גם מחוץ לתחום של למידת מכונה: בנושא של עבודה עם תמונות וקבצים, יצירת חלונות בפייתון, התמודדויות עם כישלונות, הצלחות, כתיבת פרויקטים ועוד.

ביבליוגרפיה

Ampadu, H.(2021). Dropout in Deep Learning, AI Pool https://ai-pool.com/a/s/dropout-in-deep-learning

Gupta, A.(2021). A Comprehensive Guide on Deep Learning Optimizers, Analytics Vidhya https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/

Thevenot, A. (2020). Conv2d: Finally Understand What Happens in the Forward Pass, Medium https://towardsdatascience.com/conv2d-to-finally-understand-what-happens-in-the-forward-pass-1bbaafb0b148