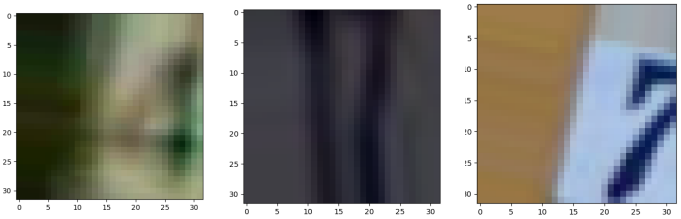
זיהוי ספרות של בתים ברחוב
למידת מכונה

שם בית ספר: מקיף י"א ראשונים
שם העבודה: זיהוי ספרות של בתים ברחוב
שם התלמיד: גל אברהם
ת.ז. התלמיד: 326246683
שם המנחה: דינה קראוס
תאריך ההגשה: 19/6/2022



מבוא

רקע

זיהוי ספרות בלמידת מכונה הוא בין המשימות היותר עתיקות בתחום. למשימה זו ולמודלים היוצאים ממנה יש שימוש נרחב בתעשייה. מודל זה אומן על תמונות של ספרות של מספרי רחוב, ולכן מדויק יותר כשמדובר בספרות של מספרי רחוב (למרות שהוא גם מדויק כאשר נותנים לו ספרות בכתב יד, מה שמראה שעשה למידה איכותית של ההבדל בין הספרות השונות). מטרת מודל זה היא שישולב בעתיד עם מודל שיכול להשתמש בו על מנת למצוא את ערכם של מספרי רחוב לפי שילוב של ספרות הרחוב אותם זיהה באמצעות מודל זה. כך, יהיה ניתן לזהות מספרי רחוב לפי תמונה של הרחוב עצמו ובכך למצוא מיקום של תמונה אך ורק לפי התמונה. זו למעשה הסיבה שבחרתי בנושא זה.

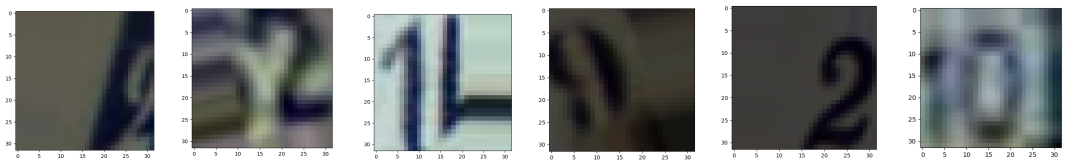
מחקר

כפי שציינתי, זיהוי ספרות באמצעות למידת מכונה היא אחת מהמשימות היותר עתיקות בתחום, ולכן השוק מוצף בהם. ניתן למצוא מודלים של זיהוי בהרבה סמארטפונים (שמזהים גם כתב) או אפליקציות. אני בחרתי בזיהוי ספרות של מספרי בתים ברחוב, תחום שפחות מיושם בתעשיה (למרות שהמודל יכול לזהות ספרות שנכתבו בכתב יד ללא הרבה בעיות). מטרת מודל זה היא שלבסוף ישתמשו בו כחלק ממודל גדול יותר לזהות מספרי רחוב מלאים ומכך להסיק את המיקום המדויק של המשתמש, ללא צורך ב GPS.
בניגוד ל machine learning קלאסי, בו משתמשים בשכבות מחוברות לחלוטין של נוירונים, השתמשתי ב Convolutional layers ועוד שכבות עליהן ארחיב בהמשך.

מבנה הפרוייקט

איסוף הכנה וניתוח הנתונים

השתמשתי במבנה נתונים SVHN שהוא מערך תמונות בעולם האמיתי לפיתוח אלגוריתמים של למידת מכונה וזיהוי אובייקטים עם דרישה מינימלית לעיבוד מוקדם ועיצוב נתונים. הוא דומה מעט ל-MNIST (למשל, התמונות הן של ספרות חתוכות קטנות), אך משלב נתונים מסומנים יותר בסדר גודל (מעל 600,000 תמונות ספרות) ומגיע מבעיה קשה משמעותית, בלתי פתורה, בעולם האמיתי (זיהוי ספרות ומספרים בתמונות סצנה טבעיות). מבנה נתונים זה נאסף ממספרי רחוב בתמונות של Google Street View.
הנתונים הגולמיים מגיעים בתוך קבצי mat (שתי קבצים train, test) המאחסנים מערכים גדולים של התמונות (שהם טנזור של 32x32x3 (רוחב 32 גובה 32 ערוצי צבע 3)) והתוויות המתאימות אליהם (לדוגמה תמונה של 1 תקבל תווית של 1). הנה דוגמה של כמה תמונות של 32 על 32 פיקסלים מתוך מבנה הנתונים.



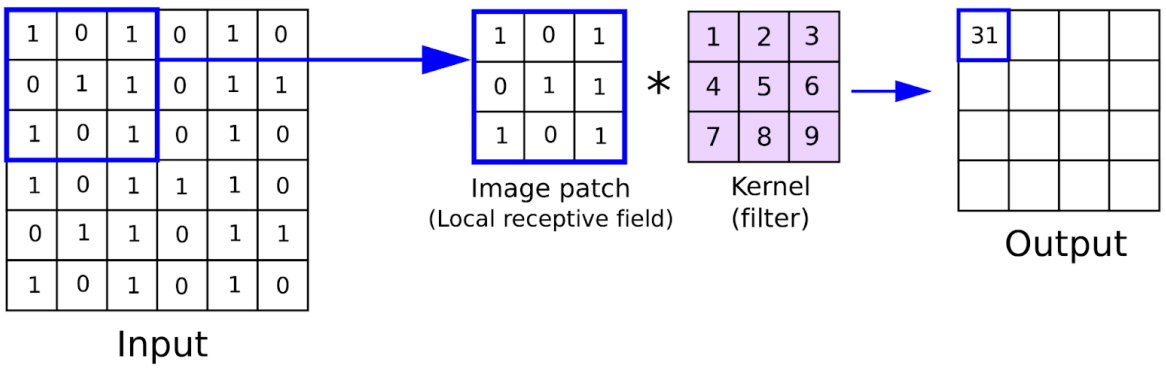
באמצעות loadmat של scipy.io אנו קוראים את קבצי ה mat ומקבלים python dictionaries. מקצים ל X את ערך ה X של מילון אחד ומסדרים אותו מחדש כדי שייצג רשימה של ndarrays שמייצגים את התמונות (רשימה רב מימדית של פיקסלים המחזיקים ערך בין 0 ל 255). על מנת לנרמל את המידע אנו מחלקים את הפיקסלים ב 255 (כדי שערכם יהיה בין 0 ל 1). מקצים את y כרשימה של ndarrays שמייצגים וקטורים של הסתברויות (כאשר האינדקס של כל איבר בוקטור מייצג את התווית של התמונה (משמע אינדקס 1 ייצג תוית של תמונה שיש בה 1).
בנוסף, אנו יוצרים גנרטורים שיעזרו לנו לטעון את המידע ב batches לתהליך האימון באמצעות ImageDataGenerator של keras.

ארכיטקטורת הפרויקט

מבנה המודל נמצא בנספחים.
<u>הסבר על שכבות הרשת:</u>

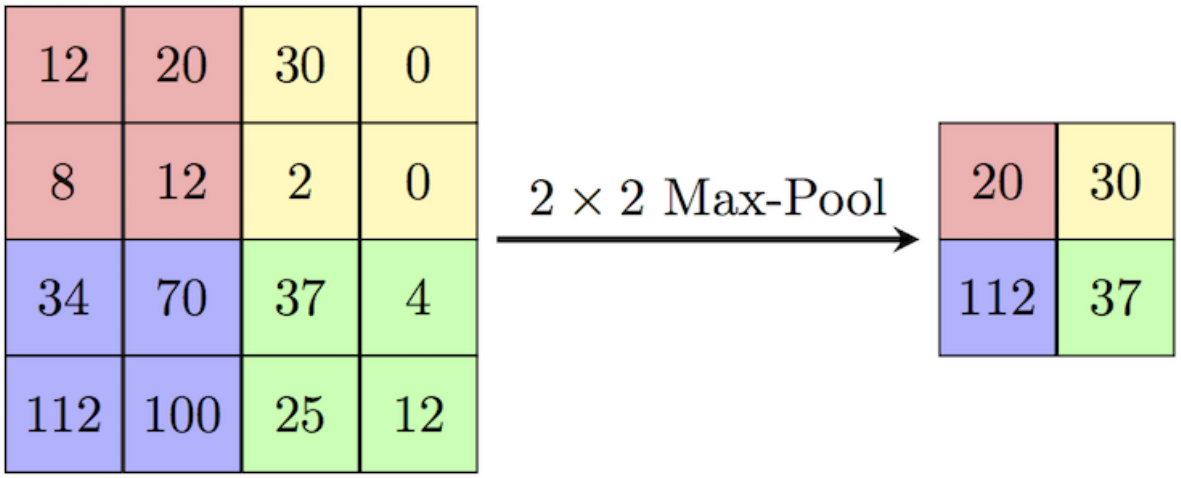
1. שכבת convolution:

שכבת convolution היא שכבה אשר מבצעת את פעולת הקונבולוציה על הקלט אותו היא מקבלת. שכבה זו היא המרכזית ביותר ב cnn. פעולת הקונבולוציה היא בעצם פעולת השמת פילטר על גבי הנתונים. במהלך הקונבולוציה, הפילטר עובר על גבי הנתונים (כמו חלון שזז, מחליק על גבי הנתונים) ומבצע מכפלה בין הפילטר אל הנתונים עליהם הוא נמצא באותו הרגע. מכפלה כזו מחזירה ערך יחיד (מכפלה בין שני מערכים, שניתן להתייחס אליהם כווקטורים). לכן, פעולת הקונבולוציה מקטינה את מערך הנתונים (אלא אם כן משתמשים בשיטות שונות, כגון padding על מנת להשאיר אותם אותו הגודל). ערכי הפילטר משתנים במהלך האימון (הם חלק מהמשתנים אותם מאמנים).
תהליך זה של השמת פילטר חושף פעמים רבות חלקים מסוימים ורלוונטים בנתונים אותם השכבה מקבלת, לדוגמא שכבת קונבולוציה אחת יכולה להדגיש פינות בתמונה, בעוד אחרת תדגיש קווי מיתר.



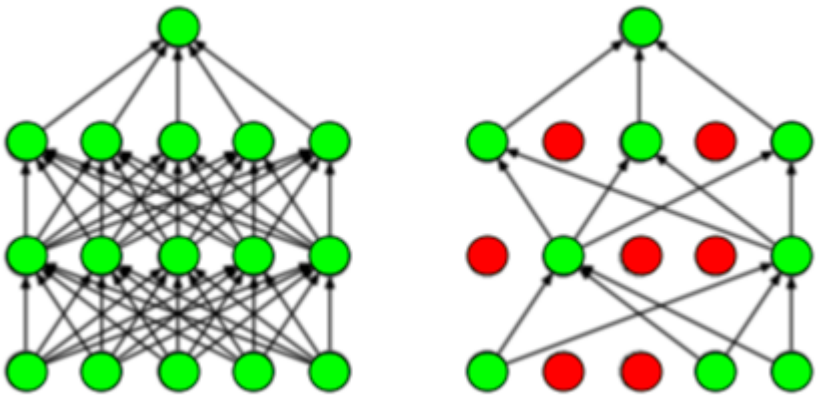
2. שכבת Batch Normalization:
על מנת להבין מהי batch normalization, עלינו להבין מהי פעולת הנרמול בכלל.
פעולת הנרמול היינה פעולה אשר מוודאת כי כל הנתונים אשר המודל מקבל היינם באותה הסקאלה. פעמים רבות הנתונים יכולים להיות ממקורות שונים ובסקאלות שונות. הבאת כל הנתונים לסקאלה סטאנדארטית אחת (לרוב בין 1-0), תורמת רבות למהירות בה המודל יכול ללמוד ולאיכות הלמידה בכלל.
פעמים רבות אנו מבצעים נרמול במהלך הכנת הנתונים, דבר שמוודא כי כאשר הנתונים נכנסים אל השכבה הראשונה במודל, הם מנורמלים. אך כל שכבה פעולת בצורה שונה ומבצעת פעולת מתמטיות שונות על הנתונים, לאחר השכבה הראשונה (וכל שכבה אחריה) הנתונים כבר לא מנורמלים ועלולים להיות בסקאלה שונה (הערך הגבוה ביותר לא יהיה 1 והנמוך ביותר לא יהיה 0). מהסיבות שהסברתי קודם, ישנו יתרון גדול לקבלת נתונים מנורמלים בכל המודל, ולכן ישנה סיבה טובה לנרמל את הנתונים מחדש בין שכבות מוסתרות (hidden) במודל. זוהי הפעולה אותה שכבת ה batch normalization מבצעת על כל mini-batch שעוברת דרך המודל במהלך האימון. היא מבצעת את פעולה זו בעזרת חישוב הממוצע והשונות (variance) של הנתונים, ביצוע נירמול שלהם, ולאחר מכן ביצוע scale and shift (הסתה - משנה את ממוצע הנתונים ושינוי הגודל - משנה את שונות (variance) הנתונים) על הנתונים. היא עושה זאת על ידי הוספת מספר מסויים אל כל ערך בנתונים (באטא) והכפלת כל ערך בנתונים במספר אחר (גאמא). שלב זה (הסטה ושינוי הגודל) היינו השלב החדשני והמהפכני בשכבה זו, שכן באטא וגאמא היינם ערכים המשתנים במהלך הלמידה (trainable parameters), ולכן שכבה זו מאפשרת להגדיל ולהסיט את הנתונים בצורה הטובה ביותר על מנת לקבל את החיזוי המדויק ביותר.
בנוסף, השכבה הזו גם מחשבת את ה Exponential Moving Average של הנתונים. נתון זה משומש במהלך ביצוע חיזוי, בו לא ניתן לחשב ממוצע ושונות של נתונים (שכן אין mini-batch אלא רק תמונה אחת), ולכן אנו השכבה משתמשת ב Exponential Moving Average שחושב על מנת לנרמל את הנתונים של התמונה אותה מנסים לחזות.

3. שכבת max pooling:
שכבה זו משומשת על מנת להקטין את גודל מערך הנתונים, תוך כדי הקטנה של "רעש" והתמקדות בחלקים המרכזיים של הנתונים (דבר שמונע overfitting - מצב בו המודל ניהיה מאוד טוב בזיהוי נתוני הלמידה, אך הופך להיות מותאם ספציפית אליהם ולא מסוגל לזהות מקרים אחרים, דבר שנוגד את המטרה של אימון מודל). שכבה זו מבצעת זאת בכך שהיא עוברת על הנתונים בעזרת "חלון" בגודל מסויים (בדומה לשכבת קונבולוציה), ולוקחת רק את הערך הגדול ביותר הנמצא בתחומי החלון אל המערך אותו היא מחזירה כפלט.



4. שכבת dropout:

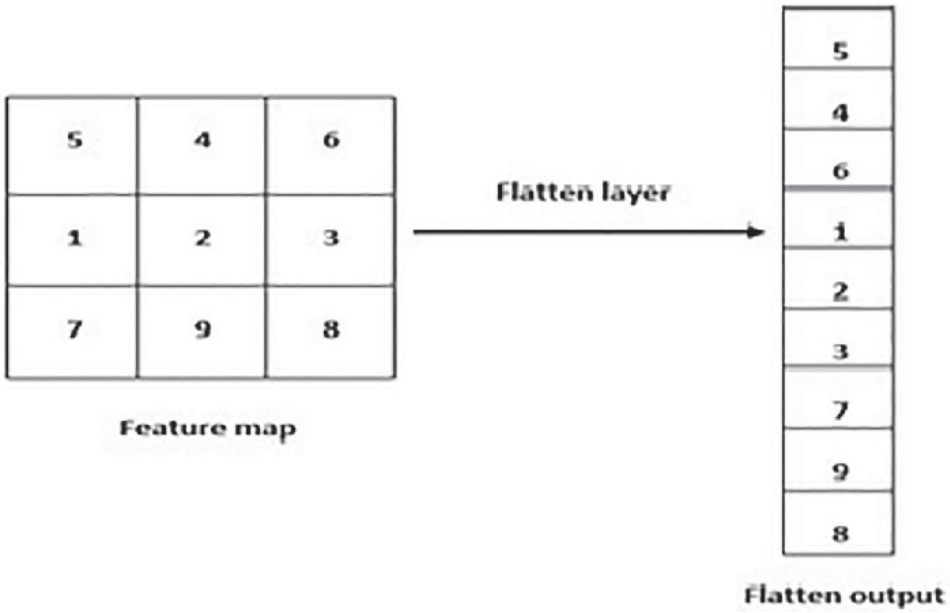
שכבה זו היא שכבה נוספת שנועדה למנוע overfitting, בכך שהיא עוצרת מערכים קבועים וחזקים במיוחד בנתונים להשפיע מאוד על חיזוי המודל. היא עושה זאת בכך שהיא dropping (הופכת את הערך ל 0) כמות מסוימת (וקבועה מראש) של ערכים רנדומלים בתמונה. דבר זה מכניס אלמנט רנדומלי אל האימון, מה שמונע overfitting.



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

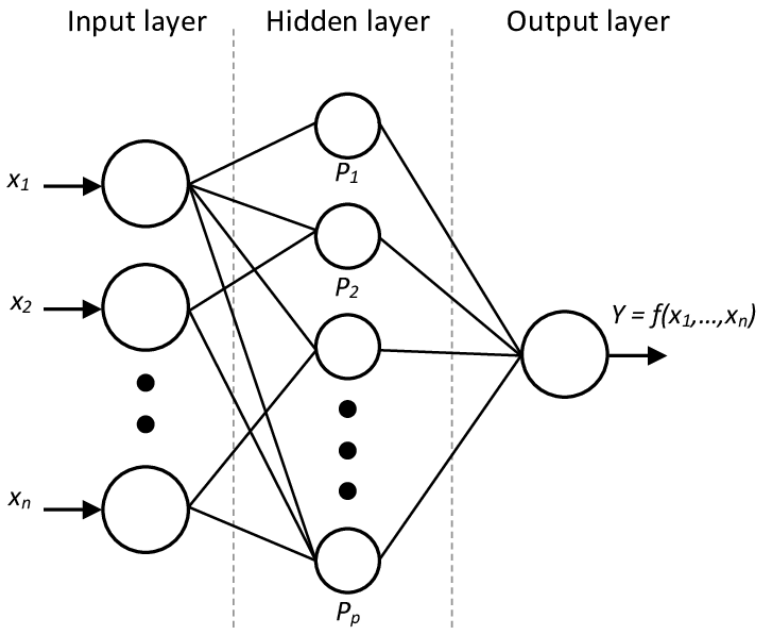
5. שכבת flatten:

שכבה זו הופכת את המערך הדו מימדי שהועבר בין השכבות עד כה (שכן זהו מערך פיקסלים של התמונה), אל מערך חד מימדי שיועבר בין השכבות הבאות.

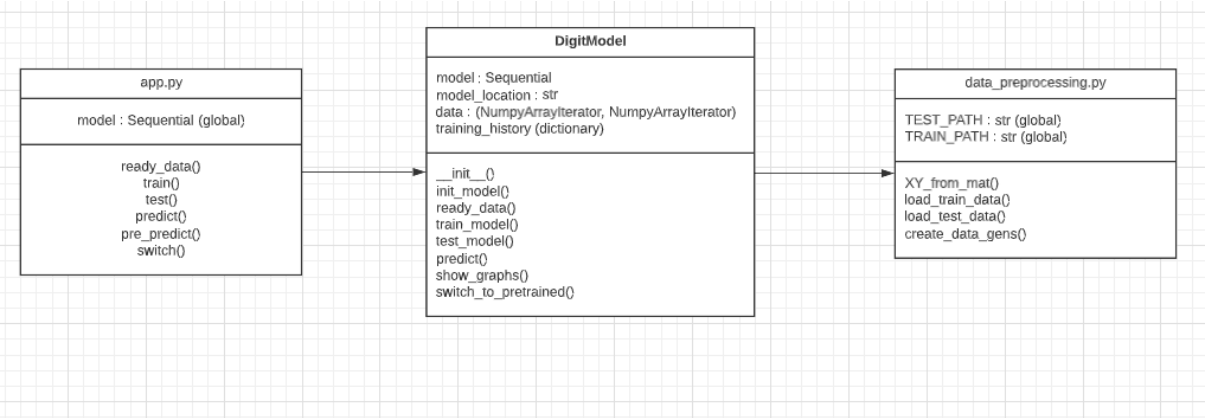


6. שכבת dense:

שכבה זו בנויה משורה של "נוירונים" המחוברים לשכבות ה dense שלפניה ואחריה (אם יש כאלו). יש לציין כי כל נוירון בשכבה אחת מחובר לכל נוירון בשכבות שלפניו ואחריו. במהלך העברת המידע בין נוירון לנוירון הערך מוכפל במשקל (המשתנה במהלך הלמידה) ולפעמים גם מוסף לו ערך הנקרא bias (שגם הוא משתנה במהלך הלמידה). לנוירונים יכולה להיות פונקציה אשר הערכים שהם מקבלים עוברים דרכה הנקראת פונקציית אקטיבציה. הערך שפונקציית האקטיבציה תקבל היינו הערך שהנוירון מקבל.

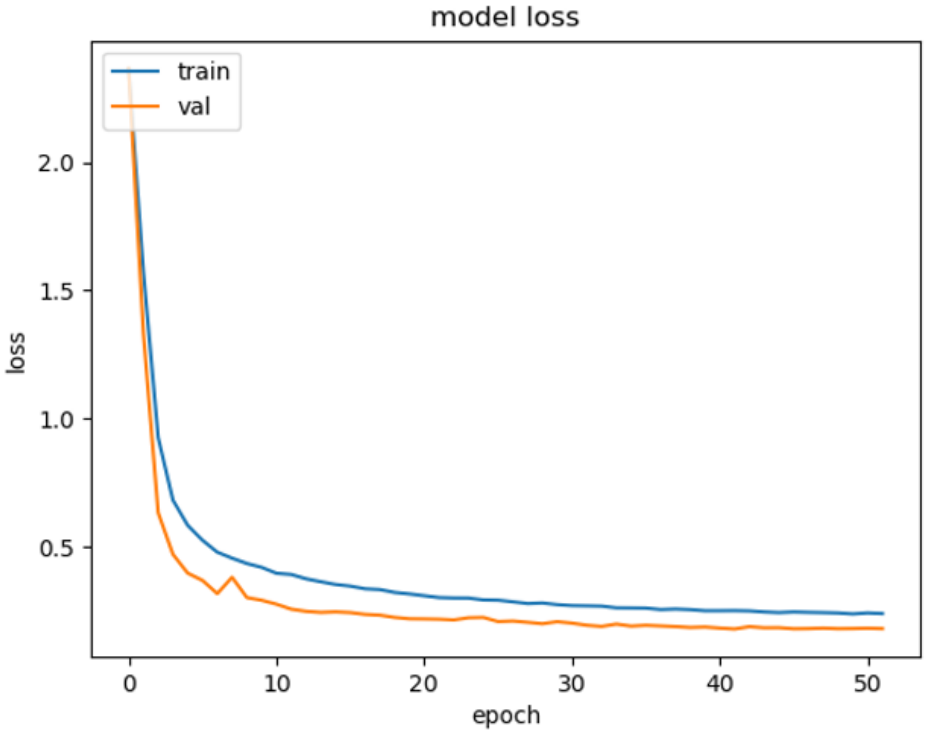


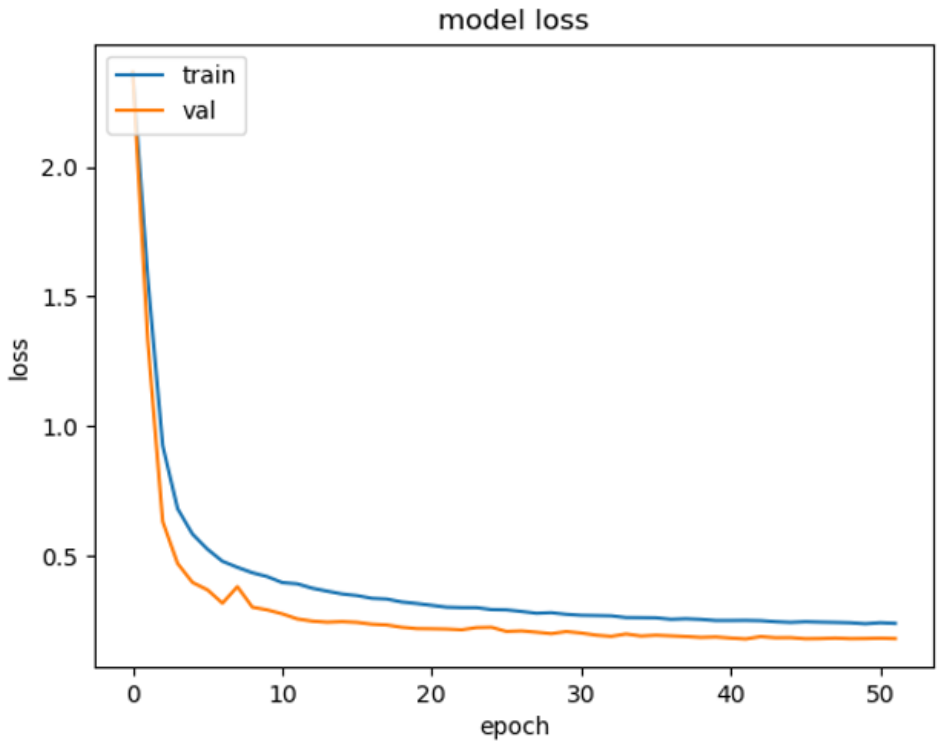
תיאור UML של המחלקות:



דוחות וגרפים המתארים את שלב האימון:

```
Epoch 40/80
287/287 [==============================] - 49s 170ms/step - loss: 0.2513 - accuracy: 0.9237 - val_loss: 0.1878 - val_accuracy: 0.9460
Epoch 41/80
287/287 [==============================] - 49s 171ms/step - loss: 0.2513 - accuracy: 0.9234 - val_loss: 0.1836 - val_accuracy: 0.9474
Epoch 42/80
287/287 [==============================] - 49s 170ms/step - loss: 0.2517 - accuracy: 0.9233 - val_loss: 0.1801 - val_accuracy: 0.9489
Epoch 43/80
287/287 [==============================] - 49s 171ms/step - loss: 0.2506 - accuracy: 0.9229 - val_loss: 0.1892 - val_accuracy: 0.9459
Epoch 44/80
287/287 [==============================] - 49s 171ms/step - loss: 0.2466 - accuracy: 0.9241 - val_loss: 0.1849 - val_accuracy: 0.9466
Epoch 45/80
287/287 [==============================] - 49s 171ms/step - loss: 0.2441 - accuracy: 0.9250 - val_loss: 0.1852 - val_accuracy: 0.9461
Epoch 46/80
287/287 [==============================] - 49s 169ms/step - loss: 0.2465 - accuracy: 0.9233 - val_loss: 0.1809 - val_accuracy: 0.9492
Epoch 47/80
287/287 [==============================] - 49s 170ms/step - loss: 0.2450 - accuracy: 0.9250 - val_loss: 0.1816 - val_accuracy: 0.9468
Epoch 48/80
287/287 [==============================] - 49s 171ms/step - loss: 0.2439 - accuracy: 0.9257 - val_loss: 0.1834 - val_accuracy: 0.9464
Epoch 49/80
287/287 [==============================] - 49s 170ms/step - loss: 0.2426 - accuracy: 0.9262 - val_loss: 0.1813 - val_accuracy: 0.9472
Epoch 50/80
287/287 [==============================] - 49s 171ms/step - loss: 0.2390 - accuracy: 0.9271 - val_loss: 0.1818 - val_accuracy: 0.9479
Epoch 51/80
287/287 [==============================] - 49s 170ms/step - loss: 0.2426 - accuracy: 0.9257 - val_loss: 0.1830 - val_accuracy: 0.9474
Epoch 52/80
287/287 [==============================] - 49s 169ms/step - loss: 0.2403 - accuracy: 0.9269 - val_loss: 0.1815 - val_accuracy: 0.9474
```





פונקציית שגיאה:

פונקציית השגיאה בה השתמשתי היא cross entropy loss. פונקציות של שגיאה בכללי נותנות מאין ציון על הביצועים של המודל שלנו בתהליך הלמידה, באמצעות אלגוריתם ייעול ההתכנסות אנו משנים את הפרמטרים של המודל על מנת שיכנסו את השגיאה של המודל לערך המינימלי שלה.
הפונקציה לוקחת את התווית המצופה למקרה אחד (או batch של מקרים) ממבנה הנתונים (שזה כמובן וקטור הסתברויות, כאשר באינדקס המתאים למקרה (נגיד ספרה 8 אינדקס שמיני בוקטור) יש ערך 1 (שכן זה התווית של המקרה הספציפי הזה) ובכל האינדקסים האחרים יש ערך 0. הפונקציה לוקחת גם את הערך שהמודל צפה בפועל (לדוגמה 0.5 באינדקס 8 בוקטור ההסתברויות שהמודל יצר). הפונקציה מכפילה את המצופה כפול מינוס לוג המצוי.

o - output, e - expected output
loss = -o*log(e)

אלגוריתם ייעול ההתכנסות:

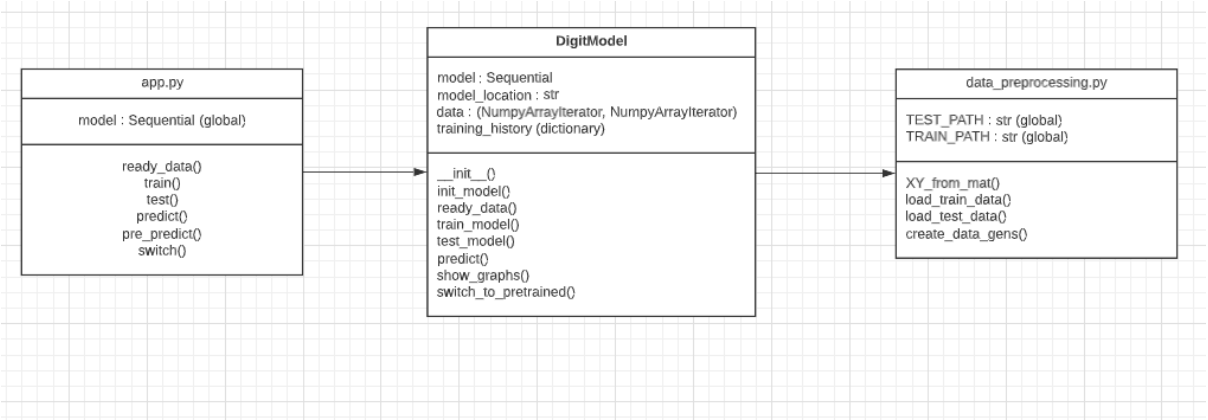
Adam - adaptive moment estimation

אלגוריתם ייעול ההתכנסות בו השתמשתי הוא Adam, והוא שילוב של שני אלגוריתמי ייעול התכנסות אחרים, Momentum, Rmsprop.
Momentum - תנופה, גורם להתכנסות של הפרמטרים לעשות צעדים גדולים יותר בצירים בהם יש ירידה עקבית בפונקציית השגיאה (n פרמטרים, n צירים). אלגוריתם זה בנוי בצורה שבה הוא נותן משקל שקטן באופן מעריכי לנגזרות היחסיות בעבר, כך האלגוריתם מתעדף מגמה של עקבית של ירידה בשגיאה מאשר מגמה לא עקבית של ירידה בשגיאה, ובאופן טבעי יגדיל את קצב הלמידה כאשר יש מגמה עקבית (כמו כדור שנופל במורד גבעה, כל עוד הוא ממשיך להתקדם במורד הגבעה, הוא כמעט ואינו זז בצירים שלא חשובים לכך (הציר האופקי)).
RMSprop - אלגוריתם ייעול התכנסות, שגורם לפרמטרים לעשות צעדים נמוכים יותר כאשר יש מגמה של נגזרות גבוהות ייתר על המידה (אם ניקח עוד פעם את הדוגמה של הכדור, הוא מתנהג כמו כדור כבד עם חיכוך ותנע, שמתעדף משטחי מינימום מאוד שטוחים מאשר משופעים).

הוא עושה את זה גם בנתינת משקלים הקטנים באופן מעריכי לנגזרות העבר, וכך אם יש נגזרות גבוהות מדי, מוריד את קצב הלמידה. לרוב Momentum מכריע יותר מ RMSprop.

שלב היישום

היישום מכין את מבנה הנתונים לאימון המודל ולבחינת המודל. מאמן את המודל, מראה את הגרפים של תהליך זה, בוחן את המודל (מדפיס את התוצאות), עושה predictions בשימוש במודל, ויכול לטעון מודל שאומן מראש מהדיסק.



השתמשתי ב tkinter שזוהי ספרית יצירת UI ב python שמבוסס על tk. ספריה זו נותנת אופציה קלה ליצירת UI למחשבים.
השתמשתי בספריה PILLOW על מנת לטעון את התמונה מהדיסק (עליה רוצים לעשות חיזוי) ולשנות את הגודל שלה לגודל שהמודל שלנו תומך בו 32x32. אנו ממירים את תמונה זו ל numpy array ואז עושים על המערך הזה חיזוי (בפורמט שהמודל שלנו יכול לקלוט).

```
filename = filedialog.askopenfilename()
if not filename:
    return
image = Image.open(filename)
image = image.resize((32,32))
image_array = np.array(image)
im = image_array.astype(int)
plt.imshow(im, interpolation='nearest')
plt.show()

image_array = image_array.astype(float)
image_array = np.reshape(image_array, (1, 32, 32, 3))
```

מדריך למפתח

תפקיד פעולה/מחלקה + תיאור אופן פעולה	שם + תכונות	מיקום
מחלקה המשמשת כמעטפה למודל שלנו ולכל	DigitModel:	model.py

	initial_lr model_location data training_history model	הפונקציות שבהם נרצה להשתמש. initial_lr = 0.001 קצב הלמידה ההתחלתי model_location = model_saves/model1 מיקום השמירה של המודל בקבצים data = None הגנרטורים של train test, עליהם נאמן/מבחן את המודל. training_history = None הסטוריית תהליך האימון של המודל (למקרה שנרצה להראות את הגרפים שלו). model = {model} המודל שלנו, מאותחל עם תחילת האפליקצייה ולכן קורא לפונקציה הבאה על מנת לאתחל את הערך של המודל.
model.py	init_model()	פעולה פנימית במחלקה הקודמת. מטרתה לבנות את המודל (שטרם אומן) על מנת שבהמשך נוכל לאמן אותו.
model.py	ready_data()	פעולה פנימית במחלקה הקודמת. מכינה את התכונה data ושמה בתוכה את הגנרטורים אותם היא יוצרת במקום.
model.py	train_model(epochs)	פעולה פנימית במחלקה הקודמת. מאמנת את המודל על ה data שהמשתמש הורה על יצירתו קודם לכן. מאמנת את המודל למספר epochs שנתנו לה. משתמשת בmodel.fit של keras. וב callbacks של earlystopping (עם סבלנות 10 שמפקח על ה validation_loss, כאשר לאחר 10 epochs זה לא משתפר, עוצר את תהליך הלמידה) ושל LearningRateScheduler שלוקח פונקציה שכתבתי שמחזירה learning rate שקטן בצורה מעריכית. בסוף תהליך האימון, שומרת את המודל באמצעות model.save של keras למיקום model_location שהוא תכונה של המחלקה.
model.py	test_model()	פעולה פנימית במחלקה הקודמת. בוחנת את המודל על test generator של data. מדפיסה לterminal את תוצאות ה test.
model.py	predict(image)	פעולה פנימית במחלקה הקודמת. מטרתה לחזות על image שהיא לוקחת איזה ספרה היא מייצגת.

		עושה את זה באמצעות model.predict של keras.
model.py	show_graphs()	פעולה פנימית במחלקה הקודמת. מטרתה להראות את הגרפים משלב האימון. עושה את זה באמצעות ה training history (שהיא תכונה פנימית) ו pyplot של matplotlib.
model.py	switch_to_pretrained()	פעולה פנימית במחלקה הקודמת. מטרתה להחליף את המודל למודל שאומן מראש (בעזרת load_model של keras).
data_preprocessing.py	XY_from_mat(path)	מטרתה קובץ mat ולהמיר אותו לקלט ופלט רצוי (תוויות) של מבנה הנתונים בתוך הקובץ). עושה זאת באמצעות loadmat של scipy.io שממיר את קובץ ה mat ל dictionary גדול ששומר בתוכו X שמתאים לקלט של המודל (כל התמונות) ו y שמתאים לפלט של המודל (כל האינדקסים של התמונות (ששארית 10 שלהם מייצגת את התווית)). מחזירה את X ו y.
data_preprocessing.py	load_train_data()	מטרתה לטעון את קובץ ה mat של תהליך האימון בשימוש בפונקציה הקודמת עם מיקום קובץ ה mat של תהליך האימון (train_32x32.mat)
data_preprocessing.py	load_test_data()	מטרתה לטעון את קובץ ה mat של תהליך הבחינה בשימוש בפונקציה הקודמת עם מיקום קובץ ה mat של תהליך הבחינה (test_32x32.mat)
data_preprocessing.py	create_data_gens()	מטרתה לטעון את מבנה הנתונים (באמצעות הפונקציות הקודמות), להשתמש במבנה הנתונים על מנת ליצור גנרטורים באמצעות ImageDataGenerator של keras שגם יגדיל את כמות הפריטים במבנה הנתונים שלנו וגם יתן לנו ממשק קל עם המידע שעליו אנחנו עוברים בתהליך האימון והבחינה. מחזירה את הגנרטורים האלו.
app.py	ready_data()	פעולה עוטפת ל ready data של DigitModel. נקראת כאשר לוחצים על כפתור Ready Data באפליקציה.
app.py	test_model()	פעולה עוטפת ל train_model של DigitModel. נקראת כאשר לוחצים על כפתור Train Model.

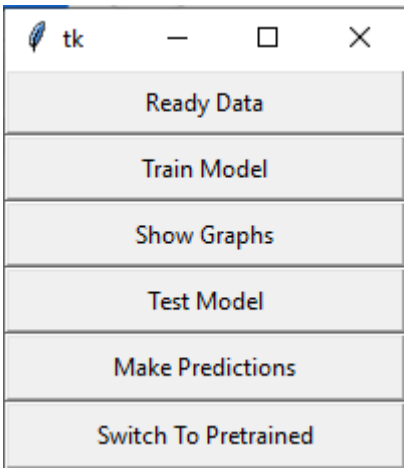
פעולה עוטפת ל predict של DigitModel. נקראת כאשר לוחצים על Predict באפליקציה. מעבדת את התמונה על מנת שתתאים להיות קלט של המודל (32x32x3), ושולחת את התמונה המתוקנת ל oredict של DigitModel.	predict()	app.py
פעולה שמכינה את התמונה הפשוטה שנתנו לה את שם הקובץ שלה על מנת שתתאים להיות קלט של המודל (predict משתמשת בה).	pre_predict()	app.py
פעולה עוטפת ל switch_ro_pretrained של DigitModel. מחליפה למודל שהוא pretrained.	switch()	app.py

מדריך למשתמש

על מנת להתקין את הפרויקט יש לפתוח anaconda prompt, לנווט לפולדר של הפרויקט ולהריץ:

```
conda env create -n gal -f dependencies.yml
conda activate gal
python app.py
```

לאפליקציה מסך אחד:



יש ללחוץ על הכפתור הראשון על מנת להכין את ה data לשימוש גם באימון וגם בבחינה של המודל.
יש ללחוץ על הכפתור השני על מנת לאמן את המודל, אפשר לראות את תוצאות האימון ב Anaconda Prompt (או ב terminal של ה IDE ממנו מריצים את זה)
יש ללחוץ על הכפתור השלישי על מנת להראות את הגרפים של תהליך הלמידה של המודל (גרף פונקצית השגיאה כתלות ב epoch וגרף ה accuracy כתלות ב epoch).
יש ללחוץ על הכפתור הרביעי על מנת לבחון את המודל, ניתן לראות את תוצאות הבחינה ב Anaconda Prompt (או ב terminal של ה IDE ממנו מריצים את זה)
יש ללחוץ על הפתור החמישי על מנת לעשות חיזויים באמצעות המודל על תמונות חדשות מהקבצים של המשתמש (png או jpeg).

יש ללחוץ על הכפתור השישי (והאחרון) על מנת להחליף את המודל למודל שאומן מראש (לא יהיה ניתן להראות את הגרפים של המודל הזה).
* כאשר המשתמש לוחץ על Make Predictions, החיזוי של המודל יוצג אל המסך מתחת לכפתור האחרון.

רפלקציה

העבודה על הפרויקט הייתה קשה אך מתגמלת. רציתי שהקוד יהיה סולידי ולכן לקח לי הרבה זמן להגיע לארכיטקטורת קוד שאני אוהב.
השתפרתי רבות בכתיבת פרויקט זה ואני מרגיש שקיבלתי כלים רבים בתחום פיתוח התוכנה. אני היום מבין מושגים עמוקים הרבה יותר בלמידת מכונה מאשר שהכרתי פעם ואני מרגיש שאני יכול להבין קונספטים מופשטים יותר בתחום. הייתי רוצה להמשיך ללמוד תחום זה ואולי אפילו לשפר את המודל שיצרתי ולבנות מעליו מודל בעל שימוש נרחב יותר בתעשיה.
האתגרים המרכזיים שעמדו בפני זה שחליתי בקורונה באמצע שנת הלימודים, מה שגרם לירידה בפרודקטיביות שלי מאז. בנוסף, בהתחלה מצאתי שנושא זה דורש הבנה מאוד גבוהה של קונספטים מתמטיים, ולכן לקח לי זמן להבין אותם ברמה מספקת.
המסקנות שלי הן שאם אני אתאמץ מספיק, אוכל לקחת על עצמי פרויקטים גדולים יותר בכל תחום הקשור למחשבים, בגלל כל המידע הנמצא באינטרנט בחינם.
לו הייתי מתחיל היום את העבודה על הפרויקט, הייתי משנה את איך שהאפליקציה עובדת כי אני מרגיש שהיא לוקה קצת בחסר. אני חושב שיש מספיק פונקציונליות לאפליקציה, אבל אפשר לדעתי לעצב אותה בצורה יותר נעימה לעין.
אני מרגיש שבסופו של דבר העבודה הייתה יחסית יעילה עבורי בגלל השימוש הנרחב שלי במקורות קוד מהאינטרנט, אך ללא ספק הייתי משנה כמה דברים בפרויקט (כפי שציינתי, עיצוב האפליקציה).

ביבליוגרפיה

http://ufldl.stanford.edu/housenumbers/
https://medium.com/swlh/deep-learning-for-house-number-detection-25a45e62c8e5
https://stackoverflow.com/questions/874461/read-mat-files-in-python
https://www.holisticseo.digital/python-seo/resize-image/
https://www.tensorflow.org/guide/keras/save_and_serialize
https://machinelearningmastery.com/how-to-make-classification-and-regression-predictions-for-deep-learning-models-in-keras/
https://www.tutorialspoint.com/python/tk_button.htm
https://stackhowto.com/how-to-change-label-text-on-button-click-in-tkinter/
https://stackoverflow.com/questions/41274007/anaconda-export-environment-file
https://medium.com/zero-equals-false/early-stopping-to-avoid-overfitting-in-neural-network-keras-b68c96ed05d9

https://mlfromscratch.com/optimizers-explained/
https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c

נספחים

```
_________________________________________________________________
conv2d (Conv2D)              (None, 32, 32, 32)        2432
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 32)        128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 64)        51264
_________________________________________________________________
batch_normalization_1 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
dropout_1 (Dropout)          (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 128)       204928
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 256)         819456
_________________________________________________________________
batch_normalization_3 (Batch (None, 8, 8, 256)         1024
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 256)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 4, 4, 512)         3277312
_________________________________________________________________
batch_normalization_4 (Batch (None, 4, 4, 512)         2048
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 512)         0
_________________________________________________________________
dropout_4 (Dropout)          (None, 2, 2, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 1000)              2049000
_________________________________________________________________
batch_normalization_5 (Batch (None, 1000)              4000
_________________________________________________________________
dropout_5 (Dropout)          (None, 1000)              0
_________________________________________________________________
dense_1 (Dense)              (None, 500)               500500
_________________________________________________________________
batch_normalization_6 (Batch (None, 500)               2000
_________________________________________________________________
dropout_6 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5010
=================================================================
Total params: 6,919,870
Trainable params: 6,914,886
Non-trainable params: 4,984
```