22928 מבוא לראייה ממוחשבת | דו"ח פרויקט סופי

ראם קישנבסקי - 213057094

14 פברואר 2023

תקציר

כפרויקט סופי בקורס ניתנה לנו המשימה להבחין בין 5 גופנים שונים של תווים בתוך תמונות טבעיות הכוללות מלל (ראה איור 1).
במסמך זה אפרט על תהליך העבודה שלי, הרעיונות שאימצתי וגם אלו שזנחתי, המסקנות שהגעתי אליהן ומבנה הפרויקט הסופי שלי וביצועיו.
המסמך מחולק לחמישה חלקים:

הבחנות ראשוניות ועיבוד מקדים **מימוש מסווג באמצעות רשת נוירונים** **שיפור ביצועים** **מבנה הפרויקט**

איור 1: דוגמה לתמונה בה עלינו לזהות את הגופן של כל תו.

חלק I

הבחנות ראשוניות ועיבוד מקדים

הסתכלות ראשונית על תמונות המאגר

מהסתכלות ראשונית על התמונות שניתנו לנו כבר הבנתי כמה דברים שיוכיחו עצמם כהכרחיים בהמשך:

1. כל האותיות באותה מילה הן בעלי אותו גופן - משמעות הדבר היא שאפשר להיעזר במידע על תווים אחרים במילה כדי להכריע לגבי גופן של תו מסוים.

2. בעת הכרעת הגופן, אין חשיבות לתמונה המלאה ממנה מגיע התו. כלומר, אין סיבה לשמור עבור כל תו מידע לגבי התמונה בה הוא מופיע - כל המידע שנחוץ לנו זוהי הסביבה המקומית הכוללת את התו עצמו ואת שארית המילה אליה הוא שייך.

3. המלל בתמונה עובר תהליך כלשהו של עיוות, מהדוגמאות שראיתי הבחנתי כי האותיות באות בגדלים ובצבעים שונים ובנוסף לפעמים עוברות תהליך של סיבוב, עיוות (warping) ולעיתים אפילו היפוך (mirroring).
המשמעות של אלו היא שהפתרון שלי צריך להיות חסין לטרנספורמציות מהסוג הזה, וכבר עלו לי רעיונות להתמודדות עם בעיה זו אליה נחזור בהמשך.

המרת המאגר ועיבוד ראשוני

כעת, כבר תוך יישום של ההבחנה השניה - ניגשתי לכתוב את פיסת הקוד הראשונה בפרויקט. הקוד שכתבתי היה אחראי על קריאת הקובץ שניתן בפורמט HDF5 וכתיבתו לתוך קובץ חדש - גם הוא בפורמט HDF5. התכלית של המרה זו היא להגיע לפורמט מידע אשר הינו יותר תמציתי ויותר מותאם לאופן בו נרצה לעבוד איתו בהמשך.

ספציפית, הפורמט החדש כלל Dataset עבור כל תו - בו נשמרו אך ורק החלקים הקטנים בכל תמונה הכללו מופעים שלו. לשמחתינו - המידע לגבי אותם Bounding Boxes נתון לנו בעבור כל תמונה (ראה איור 2) וכל שעלינו לעשות הוא לחלץ אותם מתוך התמונה - משימה שהתבררה כלא טריוויאלית כלל, ראה "חילוץ Bounding Boxes מתוך התמונה". בנוסף שמרנו בכל Dataset גם רשימה של גופנים אשר התאימה בין כל תמונה של תו מסוים לגופן שלו.

פורמט זה אינו הפורמט הסופי בו השתמשנו לשמירת המידע, אלא שלקחתי בחשבון את החסרונות הבאים ותיקנתי בהתאם:

1. רשימת הפונטים נשמרה בתור מאפיין (attribute) של ה־Dataset, מאוחר יותר למדתי שמאפניים נועדו לשמור metadata, והינם מוגבלים בגודלם - לכן השיטה לא הייתה מותאמת למצב בו המאגר גדול מאוד, בעיה זו נפגוש בהמשך כאשר נרחיב את המאגר באמצעות אוגמנטציה.

2. לא נשמר בשום מקום מידע לגבי השייכות של תו למילה מסוימת - ובכך אבד לנו המידע החשוב לגבי אילו תווים שייכים לאותה מילה ולכן בעלי אותו גופן.

איור 2: ה־Bounding Boxes של תווים שונים בתמונה.

חילוץ Bounding Boxes מתוך התמונה

לכאורה המשימה פשוטה, הרי נתונות לנו ארבעת הקואורדינטות המגדירות את המלבן בו נמצא כל תו בתמונה. הקושי נובע בכך שהמלבנים אינם מקבילים לצירי התמונה, ולמעשה כל מלבן הוא בעל אוריאנטציה מסוימת לכן לא נוכל פשוט "לגזור" אותו מהתמונה ולקבל תמונה חדשה. בחרתי לבצע הומוגרפיה, כלומר להשתמש בזוית המחושבת של סיבוב המלבן על מנת לסובב אותו חזרה כך שיהיה מקביל לצירים - ולשמור תמונה זו.

פתרון שנזנח: בהתחלה לקחתי את ערכי המקסימום והמינימום של כל ארבעת נק' המלבן והשתמשתי בהם כדי לקבל מלבן מינימלי שמכיל את המלבן המסובב ומקביל לצירים - אך בשיטה זו נוסף מידע עודף לתמונה והאותיות עצמן לא היו באוריאנטציה אחידה.

עיבוד מקדים

כעת, לאחר חילוץ ה־Bounding Boxes של כל התווים מכל התמונות ושמירתם במאגר לפי סוג התו, הגיע שלב העיבוד המקדים (Preprocessing).

בשלב זה ניסיתי לחשוב אילו פעולות עיבוד אוכל לבצע על כל תמונה על מנת "להכין" את המידע שהיא שומרת לקבלה לרשת המסווגת שלי (בשלב זה כבר ידעתי שארצה לפתור את המשימה באמצעות רשת נוירונים).

לשם כך בניתי רשת פשוטה ובחנתי את ביצועיה כאשר התמונות שהיא מקבלת עברו עיבוד מסוגים שונים. בשלב מאוחר יותר בפרויקט, כאשר מבנה הרשת פחות או יותר התקבע, חזרתי לבעיית העיבוד המקדים ובחנתי שנית את האופן בו אני מעבד את התמונות לפני הכנסתם לרשת. על בניית הרשת המסווגת ארחיב בחלק השלישי: **מימוש מסווג באמצעות רשת נוירונים**.

המרת התמונה לגווני אפור

המרת התמונה לגווני אפור הייתה נראית לי צעד הגיוני, שכן הגופן אינו תלוי בצבע הטקסט, ובעצם כל שאנו רוצים לדעת זה את צורת האות, התפקיד שהצבע משרת בהקשר זה הוא הפרדת האות מהרקע, ובכך הדגשת הצורה שלו. ואילו צורת האות נשמרת גם בתמונת גווני אפור, בה אמנם התמונה נראית דלה יותר אך ההבדל בגוון בין צבע הטקסט לצבע הרקע נותר על קינו. בנוסף, ההמרה לגווני אפור מהווה ירידה במימד התמונה ומאפשרת אימון מהיר יותר של הרשת.

במבחן התוצאה, לא ראיתי הבדל ניכר בין ביצועי הרשת כאשר מוזנות אליה תמונות צבע וכאשר מדובר בתמונות גווני אפור, לכן בחרתי לבצע המרה זו.

מציאת קצוות באמצעות אלגוריתם Canny

כאמור, מרבית מתכלית העיבוד המקדים מתנקז ליצירת הפרדה בין החזית (במקרה שלנו - האות) לבין הרקע. אלגוריתם מתבקש במקרה זה הוא Canny, מאחר והמתווה של האות מהווה "קצה" בתמונה, ניתן לזהות ולהדגיש אותו באמצעות Canny. בחנתי מספר אפשרויות לשימוש ב־Canny לשם טיוב התמונה, אך לבסוף נטשתי את כולם מפעת כך שלא הובילו לשיפור בתוצאות.

במקרה הבסיסי ביותר, כל שעשיתי הוא להפעיל את האלגוריתם עם ערכי threshold מסוימים שמצאתי באמצעות ניסוי ותהייה. שמתי לב לתופעה מעניינת: עם שיטה זו, הרשת דאז הגיעה לדיוק של 44% כבר ב־Epoch השני של האימון, בעוד בלעדיו לקח לה לפחות שישה Epochs להגיע לדיוק כזה - אך אחרי מספיק זמן אימון, הרשת שאינה מקבלת מידע שעבר Canny, הגיעה לתוצאות דיוק יותר טוב. ההסבר שאני מציע לתופעה זו הוא כזה - הרצת Canny בעצם ממירה את התמונה לתמונת שחור־לבן, בה לכל פיקסל שני ערכים אפשריים, והיא עושה זאת תוך כדי שימור המידע אשר הכי נחוץ לנו - המבנה של האות, לכן, כאשר המידע נכנס בצורה הזאת לרשת - הפונקציה המקשרת בין התמונה לוקטור המייצג את הגופן המתאים הייתה פשוטה משמעותית מהפונקציה המקבילה במקרה בו התמונות לא עברו Canny, ומכאן מדוע הצליחה הרשת להגיע לדיוק ניכר כבר בשלבי האימון המוקדמים. ואילו הסיבה שהרשת לא מגיעה לאחוזי דיוק גבוהים כמו במקרה של תמונת גווני האפור היא שפשוט מקודד פחות מידע בתמונת השחור לבן ולמעשה מאפיינים "רכים" יותר של התמונה נעלמו לגמרי, לכן הדיוק המקסימלי של הרשת הוגבל.

נורמליזציה

ביצעתי נרמול של ערכי הפיקסלים בכל תמונת גווני אפור, על מנת למרכז את כל המאגר סביב אותה ראשית ובכך להגביר את ביצועי הרשת. שיטה פשוטה זו, אשר מימשתי בשורת קוד אחת, אחראית לקפיצה ניכרת בדיוק הסיווג של הרשת.

אוגמנטציה

אחד המרכיבים החשובים ביותר בשלב העיבוד המקדים זהו האוגמנטציה: על מנת להגיע לביצועים טובים של רשתות נוירונים, ישנו צורך בכמות מידע גדולה. בנוסף, בשביל חוסן (robustness) של המודל, דרוש שהוא יתאמן על דוגמאות מגוונות. הנקודה האחרונה התבררה כקריטית במקרה שלנו, ניתן לראות שלאחר מעבר לגווני אפור, בחלק מהתמונות, התו מופיע בגוון כהה על רקע בהיר ובחלק אחר הוא מופיע בגוון בהיר על רקע כהה. בנוסף, פעמים רבות התו הפוך. ישנן שתי דרכים אפשריות להתמודד עם תופעה זו:

1. מציאת דרך כלשהי תוך שימוש בכלים של עיבוד תמונה על מנת להביא את כל התמונות לפורמט אחיד, לדוג' - כל התווים יהיו שחורים על רקע לבן ולא הפוכים. ניסיתי בדרך זו, בהצלחה מספקת, לגרום לכך שלא יהיו הרבה תמונות של תווים בהם התו נראה על הצד, באמצעות כך שהפכתי כל מלבן כך שהצלע הקצרה שלו תהא אופקית - זאת מכיוון שאותיות על פי רוב הן יותר גבוהות מרחבות. באשר לבעיית הגוונים וההיפוך לא מצאתי פתרון בשיטה זו.

2. אוגמנטציה - זו השיטה בה השתמשתי לבסוף והיא הובילה לשיפור ניכר בביצועים. בשיטה זו, מזינים לרשת עוד עותקים של אותה תמונה (וכמובן עם אותו תיוג), עם עיוותים מסוימים. במקרה שלנו, עבור כל תמונה של תו אני מזין לרשת 4 תמונות: את התמונה המקורית, תמונה הפוכה $180°$, תמונת negative ותמונת negative של התמונה ההפוכה. תמונות ה־negative מאפשרות לרשת לזהות את גופן התו הן כאשר הוא בהיר ביחס לרקע והן כאשר כהה. התמונות ההפוכות מאפשרות לרשת לזהות תוים גם כשהם הפוכים.

איור 3: מופעים של האות T בתמונות

חלק II

מימוש מסווג באמצעות רשת נוירונים

DNN - Deep Neural Network

זו הרשת הראשונה שבחנתי, מדובר ברשת פשוטה בה כל השכבות הן Fully-Connected, ובעצם הפרמטרים המשמעותיים ביותר במבנה שלה הם מספר השכבות וגודל כל שכבה. מניסוי ותהייה הגעתי יחסית מהר לרשת שהניבה תוצאות ראשוניות שהיו יותר טובות מ־20% דיוק, כלומר יותר טובות מניחוש. מבנה הרשת היה:

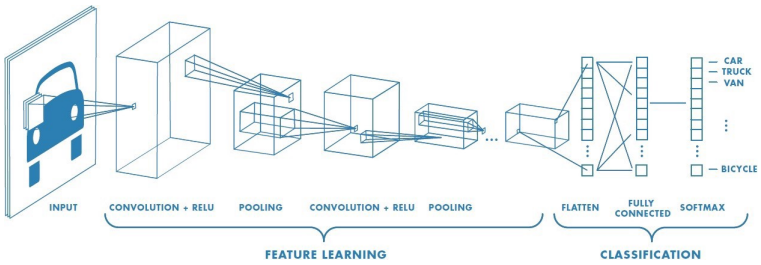
$$\texttt{Linear} \rightarrow \texttt{ReLU} \rightarrow \texttt{Dropout} \rightarrow \ldots \rightarrow \texttt{LogSoftmax}$$

בעצם שלושת אבני הבניין המרכזיים היו שכבות Linear, Relu, Dropout והמבנה שלהם והפרמטרים הפנימיים שלהם הובילו אותי לתוצאות טובות יחסית לרשת פשוטה כמו זו.

CNN - Convolutional Neural Network

המוטיבציה למעבר לרשת CNN היה שהרי אנחנו עוסקים בתמונות של אותיות, ורשתות קונבולוציה הן השולטות בתחום הזה, מכיוון שהן מותאמות ללמידה ממידע שמאורגן באופן מרחבי (כמו תמונה) ומחלצות Features אשר מעידים על תכונות מסוימות של התמונה. ההתאמה למקרה שלנו מיידית - נרצה ללמוד אילו פיצ'רים מאפיינים כל גופן, ולהתשמש בהם לסיווג.

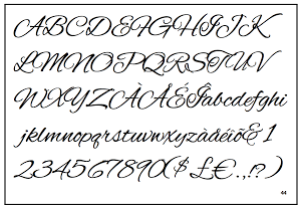
תחילה התוצאות שקיבלתי לא היו מזהירות, ושינויים רבים שעשיתי ברשת לא הובילו לשיפור ניכר, מסיבה זו חזרתי להשתמש ב־DNN למשך תקופה אך לבסוף חזרי לרשתות קונבולוציה כאשר פריצת הדרך הייתה שינוי ארכיטקטורת השכבות האחרונות ברשת - אלה אשר אחראיות על קבלת וקטור הפיצ'רים כקלט וסיווג לגופנים. מה שעשיתי היה בעצם להדביק את רשת ה־DNN בתוך המודל הנוכחי ושימוש בו בתור ב־Classifier של הרשת (ראה איור 4) - שיטה זו של שילוב בין העוצמות של שני סוגי הרשתות הוא שהוביל לתוצאות המיטביות ובו השתמשתי גם בגרסה הסופית של הפיתרון.



איור 4: מבנה רשת מסוג CNN, במקרה שלי ה־Classifier הוא בדיוק רשת ה־DNN בה השתמשתי קודם למעבר.

FCN - Fully Convolutional Neural Network

רשת FCN דומה לרשת מסוג CNN אך היא אינה כוללת חלק האחראי על Classification, למעשה ברשת כולה אין שכבה לינארית, במקום זאת, השכבות האחרונות הן קונבולוציות בגודל 1×1 המסמלצות שכבות FC. הדבר מאפשר תכונה רצויה של הרשת שהיא אגנוסטיות לגודל הקלט - כלומר, רשתות מסוג FCN יכולות לקבל כקלט תמונות מגדלים שונים. הדבר נראה רצוי בעיניי שכן חשבתי לנצל את העובדה שמילים שלמות הן בעלי אותו גופן בכך שאזין לרשת תמונות של מילים שלמות במקום תמונות של תווים בודדים - האינטואיציה מאחורי רעיון זה היא שכל גופן הוא בעל "סגנון" מסוים משלו, והסגנון הזה נשמר במידה רבה גם בין אותיות שונות, דוגמה לכך היא בגופן Alex Brush (ראה איור 5) - ניתן להבחין כי אותיות שונות משמרות את הסגנון של הגופן, אשר מאופיין בין היתר ע"י ספירלות בקצוות האותיות - פיצ'ר שקל לרשת קונבולוציה לזהות. אם כן, הרציונל להזין לרשת תמונה של מילים שלמות היה שבאופן זה ישנה חזרה גדולה יותר על סגנון הגופן, כלומר יותר מידע למסווג ומכאן דיוק גבוה יותר.



איור 5: הגופן Alex Brush

הכיוון הזה ננטש עוד בטרם התחלתי לממש את הרשת, זאת בעצת חבר לקורס אשר הצביע על כך שרשתות מסוג FCN אינן מתאימות לבעיות סיווג, וישנן דרכים אחרות לייצר מודלים שאגנוסטיים לגודל הקלט - אך הם מסובכים מאוד. לכן, החלטתי להישאר עם רשת מסוג CNN ולנצל בצורה אחרת את העובדה שלמילים יש גופן אחיד, על כך ארחיב בחלק השלישי: **שיפור ביצועים**.

חלק III

שיפור ביצועים

בשלב האחרון של הפיתוח, לאחר שהיה בידי מודל "גלובלי" אשר מסוגל לסווג כל אות לגופן כלשהו בדיוק מרשים (שעומד על 76%). התחלתי לכתוב את הקוד אשר אחראי על ביצוע הסיווג בפועל, כלומר לא למטרת אימון. לשם כך השתמשתי בשתי שיטות:

אימון מסווגים ייחודיים לתווים

הגיוני לחשוב שאם אנו מתבוננים על אות מסוימת שאינה ברורה, יהיה לנו קל יותר להכריע מה הגופן שלה אם יגידו לנו מה האות הזאת. לכן, בחנתי את האפשרות של אימון מסווגים נוספים - מסווג לכל אות.

בשיטה זו ישנה בעיה מרכזית והיא חוסר במידע, שכן מסווג לאות x יוכל להתאמן רק על תמונות של האות x - זה הרבה פחות מידע ממה שניתן לנו תחילה. בנוסף, לאותיות נפוצות פחות יש אפילו פחות מידע.

הפתרון שלי לבעיה מורכב משלושה חלקים.

1. אוגמנטציה - כבר בשלב העיבוד המקדים הכפלתי כך את כמות המידע פי 4.
2. fine-tuning על המודל הגלובלי להתאמתו לאות מסוימת.
3. עבור תווים עם פחות מ־50 תמונות לא אימנתי מסווג כלל, השתמשתי במסווג הגלובלי.

לאחר כל אלו, היו בידי מסווגים להרבה תווים, מתוכם השארתי רק את המסווגים אשר השיגו אחוז דיוק גבוה יותר מהמסווג הגלובלי. כעת, כאשר נידרש לסווג אות מסוימת - נבדוק האם קיים לה מסווג ספציפי, אם קיים - נשתמש בו, אחרת - נשתמש בגלובלי.

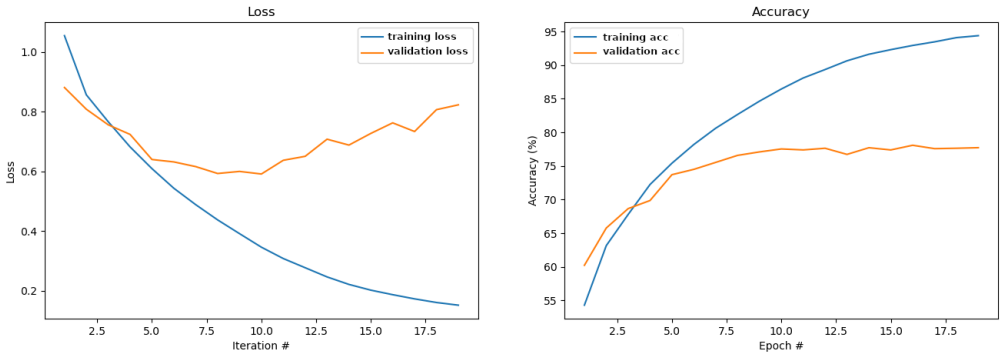
שימוש במידע מקדים

מאחר והסיווג אינו למטרת אימון, הייתי חופשי להשתמש בכל המידע המקדים שברשותי לשיפור דיוק הסיווג (אינטואיטיבית, אם הייתי מבצע שימוש במידע מקדים בשלב אימון המודל, הדבר היה רק פוגע ביכולת שלו לסווג אותיות - בחרתי "להסתיר" מהמודל כל מידע קודם שהיה לי על מנת שיוכל לתת סיווג באופן לא מוטה). המידע המקדים שברשותי הוא כמובן העובדה שלאותיות אשר מרכיבות את אותה מילה יש אותו גופן. אזי, עלינו לסווג לכל תווי מילה כלשהי את אותו גופן, לכן נרצה לבחור גופן שמהווה קונסנזוס.

האופן בו בחרתי לבצע זאת הוא כזה: נתון לנו עבור כל תו במילה את וקטור ההסתברות של הגופנים, שהוא הפלט של המסווג (בין אם המסווג הגלובלי או מסווג ספציפי) - הגיוני אם כן לקחת ממוצע של כל וקטורי ההסתברות הללו, ואז לבחור את הגופן עם הסבירות הגבוהה ביותר. שיפור של שיטה זו הוא לקחת ממוצע משוקלל, כאשר נרצה "להתחשב יותר" בפלט של מסווג מסוים אם ידוע לנו שהוא מדויק יותר. אכן - זו השיטה בה השתמשתי לבסוף, למסווג הגלובלי ולכל מסווג אחר של תו מסוים, שמרתי את הדיוק האמפירי שלו אשר מדדתי בתהליך האימון, כעת, כאשר רציתי לקחת בחשבון את הפלט שלו - הכפלתי אותו בדיוק המסווג. בשיטה זו, מסווגים מדויקים קיבלו יותר "say" בהכרעה על גופן המילה ובהתאם ראיתי שיפור בדיוק.

תוצאות

השיטות הנ"ל איפשרו למודל להגיע לאחוז דיוק מרשים מאוד של 99.2% על מאגר ה־test. השיפור הוא ניכר בהשוואה ל־76% דיוק של המודל הגלובלי על ה־validation set בזמן האימון - כאשר לא השתמשנו במידע המקדים ובמסווגים המותאמים. את תהליך האימון של המודל הגלובלי (שהוא העיקרי) ניתן לראות באיור 76:



איור 6: דיוק ו־loss בזמן אימון המודל הגלובלי

חלק IV
מבנה הפרויקט

הפרויקט הסופי מחולק ל־11 קבצי קוד ו־3 תיקיות, ראה איור 7.

```
.
├── best_models
├── checkpoints
├── const.py
├── data
├── dataset.py
├── hdf5_utils.py
├── image_utils.py
├── models.py
├── plot.py
├── predict.py
├── preprocess.py
├── training.py
├── train.py
└── types_.py
```

איור 7: היררכיית קבצי המקור של הפרויקט

best_models בתיקייה זו מאוחסנים קבצי .pth הניתנים לקריאה ע"י torch.load() ־ אלו הם הגרסאות ־ לאחר אימון ־ של המודלים בפרויקט: המודל הגלובלי והמודלים של כלל התווים. גם מאוחסן בה קובץ accuracies.json אשר מכיל את הדיוקים של כל מודל, כפי שנמדד בזמן האימון.

checkpoints בתיקייה זו נשמרים הסנאפשוטים של הגרסא הכי טובה של כל מודל בזמן האימון.

const.py קבועים למיניהם.

data בתיקייה זו מאוחסן כל המידע על הוריאציות השונות שלו ־ עליו המודל מתאמן.

dataset.py קוד האחראי על המרת המידע בפורמט HDF5 למחלקות CharacterDataset אשר ממשות את הממשק Dataset של pytorch ומחזיקות את המידע שניתן לרשתות בזמן אימון.

hdf5_utils.py קוד שמספק אבסטרקציה לעבודה מול קבצי HDF5.

image_utils פעולות על תמונה, לרבות הומוגרפיה.

models.py המודלים של רשתות הנוירונים למינהם.

plot.py קוד ליצירת גרפים.

predict.py הסקריפט המרכזי של הפרויקט, מקבל קובץ HDF5 וכותב לתוך results.csv את תוצאות הסיווג.

preprocess.py בגרסה הקודמת של הפורמט, קוד זה ביצע את ההמרה.

training.py מחלקת Trainer אשר מספקת פעולות לאימון רשת.

train.py קוד אשר מבצע שימוש במחלקה Trainer שהוגדרה ב־training.py ומבצע בפועל את אימון כל המסווגים, תוך שמירת גרסאות שלהם בתיקייה checkpoints.

types_.py סוגי מבני נתונים למיניהם.