בס"ד

Deep Learning – Ex2

Submitters:

Avidan Menashe 207812421

Shoham Galili 208010785

בס"ד

תוכן עניינים

Background

בתרגיל זה מימשנו multi-class classification עבור 10 classes בשני שלבים: ראשית, בעזרת Logistic Regression שנית, בעזרת Neural Network with one hidden layer. השתמשנו בDataset – Fashion-MNIST אשר מכיל 70,000 samples, כך שלכל sample ישנם 28x28 features.
על מנת לממש את המודל בצורה מיטבית חילקנו את הData הנתון לשלוש קבוצות: Train, Test Validation.

The steps of running the code:

ראשית, על מנת להריץ את הקוד שלנו יש לוודא שבקובץ פייתון של הקוד הראשי (אצלנו הוא נקרא exc2.py) מותקנות הספריות הבאות :
numpy(בספרייה זו השתמשנו לצורך חישובים מתמטיים וכתיבת המודל שלנו)
matplotlib.ppylot(בספריה זו השתמשנו לצורך יצירת הגרפים)
panda(בספריה זו השתמשנו על מנת לטעון את הנתונים של ה data set שלנו --Fashion MNIST)
pyarrow (בספריה זו השתמשנו על מנת להציג את השגיאות בCONSOL בצורה קריאה יותר במהלך תהליך העבודה)
Logistic_regression_class(מחלקה שבה כתבנו את הקוד המממש את multi-class logistic regression classifier)
FashionNet(מחלקה שבה כתבנו את הקוד המממש את Neural Network (NN) with one hidden layer)

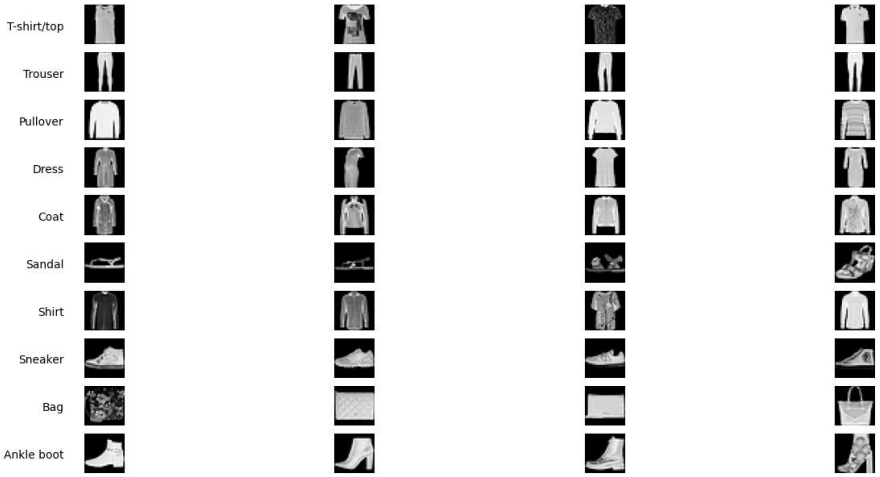
<u>הערה:</u> במחלקות Logistic_regression_class, FashionNet יש להתקין רק את הספריות numpy,sys כאשר sys זוהי ספרייה שמאפשרת לנו הדפסה בזמן הרצה בצורה נוחה יותר (אנו מדפיסים תמיד שורה אחת שמתעדכנת לconsol את התוצאות של הEPOCH הנוכחי במקום להדפיס שורה לכל EPOCH בנפרד).

שנית, יש לוודא שבתיקייה של הקוד מופיעים הקבצים test.csv, train.csv על מנת שנוכל לטעון את המידע של הdata set אל הקוד.

לבסוף, יש ללחוץ על כפתור RUN בקובץ של הקוד הראשי (exc2.py).

Part 1- Visualize the Data

בסעיף זה בחרנו בצורה אקראית 4 תמונות מכל class והצגנו זאת כך שכל שורה מייצגת class אחר, וכל עמודה מייצגת תמונה נוספת מאותו class. בתחילת כל שורה הצגנו את הlabel המייצג כל class כפי שניתן לראות בתמונה הבאה:



Part 2- Logistic Regression Classifier

על מנת להגיע לסט הפרמטרים אשר מניב את התוצאות המיטביות השתמשנו בשיטת "ניסוי וטעיה". בתחילה, אימנו את המודל עם סט פרמטרים התחלתי, קיבלנו ערך התחלתי ולא מספיק טוב, על כן המשכנו לאמן את המודל כך שבכל הרצה בדקנו את השפעת שינוי אחד הפרמטרים על תוצאות ה Accuracy עבור הValidation. המשכנו בשלבים עד אשר הגענו לרמת דיוק טובה מספיק.

להלן פירוט התהליך שעשינו ושינוי הפרמטרים בהתאמה:

Accuracy	Num of Epochs	Batch size	Lr	L2
70.812%	100	64	0.001	0.001
76.116%	100	64	0.003	0.001
81.948%	100	64	0.01	0.001
84.955%	100	64	0.03	0.001
83.812%	100	64	0.03	0.002
85.393%	150	64	0.03	0.001
85.304%	200	64	0.03	0.001
85.482%	200	128	0.03	0.001
83.509%	200	256	0.03	0.001
84.946%	200	128	0.03	0.002
84.705%	250	128	0.03	0.001
84.018%	200	128	0.05	0.001

פירוט השפעת hyperparameters על רמת דיוק המודל:

תחילה בחרנו שרירותית את הנתונים ההתחלתיים על מנת לקבל איזושהי תמונה כללית על המודל ונקודה התחלתית. ניתן לראות שהגענו לערך: Accuracy = 70.812%

- השפעת ערך ה Learning Rate על דיוק המודל.

כידוע קצב הלמידה משפיע ישירות על המהירות וההתכנסות של אלגוריתם האופטימיזציה המשמש לאימון, בחירת קצב למידה מתאים יכולה להשפיע משמעותית על הדיוק וביצועי המודל. Learning Rate נמוך מדי עלול להביא לפתרון לא אופטימלי נוסף על זמן התכנסות ארוך- Underfitting. מאידך, Learning rate גבוה מדי עלול לגרום למודל להתנודד סביב הפתרון האופטימלי ולהוביל לOverfitting. שמנו לב כי בהעלאת ערך ה Learning Rate רמת הדיוק עלתה עד לערך lr = 0.03 ומעבר לכך קיבלנו ירידה משמעותית בדיוק המודל – זאת בהתאם למצופה שהlr לא צריך להיות גבוה מדי.

- לאחר מכן, בדקנו את השפעת ערך L2 regularization על דיוק המודל.

פרמטר זה "מעניש" משקלים גדולים מדי ומנמיך את ערכם ובכך מונע Overfitting, מביא לשליטה מיטבית במורכבות המודל ושיפורו. תחילה קבענו L2=0.001 ניסינו להעלות את ערכו ב0.001 ונוכחנו לגלות כי רמת דיוק המודל ירדה- לכן חזרנו לערך ההתחלתי.

- בשלב הבא, בדקנו את השפעת גודל Num of Epochs על דיוק המודל.

פרמטר זה מייצג את מספר הפעמים שה training dataset מועבר קדימה ואחורה דרך המודל בתהליך האימון. כמות נמוכה של epochs עלולה לגרום לUnderfitting כיוון שהמודל לא מספיק ללמוד מהdataset וגורם לדיוק נמוך יותר. מאידך, כמות גבוהה של epochs עלולה לגרום לOverfitting כיוון שהמודל לומד רעש מהdataset ולכן ישנה חוסר הצלחה בהכללת הלמידה לdata חדש אשר לא נראה קודם. על כן, התחלנו בepochs 100 והעלינו את הערך בהתאם לרמת הדיוק עד שקיבלנו בepochs 200 את רמת הדיוק המיטבית עבור סט פרמטרים זה.

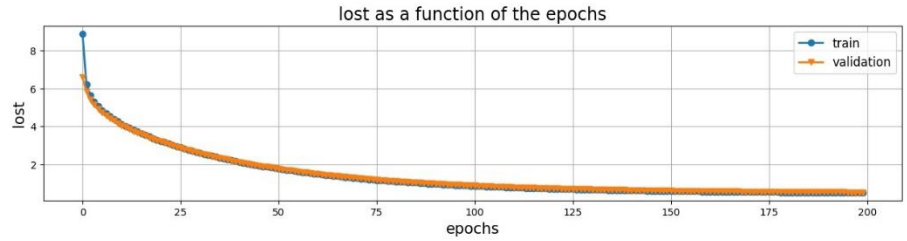
- בשלב הבא, בדקנו את השפעת גודל הBatch Size על דיוק המודל.

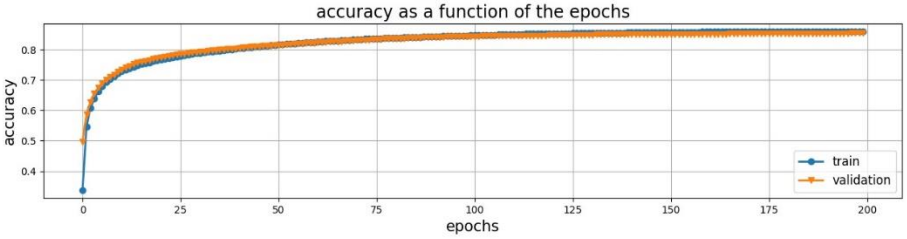
פרמטר זה קובע את מספר הדגימות המעובדות לפני עדכון הפרמטרים של המודל במהלך תהליך האימון. הוא ממלא תפקיד מכריע בקביעת היעילות והאפקטיביות של תהליך האופטימיזציה. הBatch size משפיע ביחס ישר על התכנסות המודל- זאת משום שגדלי קבוצות גדולים יותר גורמים בד"כ לעדכונים יציבים יותר והתכנסות מהירה יותר. כיוון שהן מספקות הערכות מדויקות יותר של הגראדינט של פונקציית הloss. התחלנו בערך Batch Size=64 הגדלנו אותו על מנת להגיע לביצועים טובים יותר של המודל כמצופה, אך שמנו לב כי קיים מעין חסם עליון בערך 256 בו רמת הדיוק החלה לרדת מכיוון שערך Batch Size גדול מדי עלול לגרום לאילוצי זיכרון ולהתכנסות איטית יותר. על כן הערך האופטימלי מבחינתנו הוא Batch Size = 128 כפי שמוצג בטבלה.

בסה"כ הגענו לרמת הדיוק המיטבית Accuracy validation = 85.482% בשילוב סט הפרמטרים הבא: Num of Epochs=200 Batch Size=128 Lr=0.03 L2=0.001 כדלהלן:

```
C:\Users\avidan\Desktop\DeepLearning_exc2\.venv\Scripts\python.exe C:\Users\avidan\Desktop\DeepLearning_exc2\exc2.py
Epoch: 200Train Loss: 0.504 | Train Accuracy: 85.991% Validation Loss: 0.525 | Validation Accuracy: 85.482%
Process finished with exit code 0
```

מצורף להלן גרפים המתארים את ערכי הloss וה accuracy כפונקציה של מספר ה epochs עבור סט הTrain והValidation:





מהגרפים לעיל ניתן לראות כי ככל שמספר הepochים עולה ← loss יורד, accuracy עולה.
בהתאם לתיאוריה שהסברנו לעיל וכפי שציפינו שיקרה. עם הגדלת מספר הepochים המודל הופך מאומן יותר, מכיר את הdata בצורה טובה. הloss פוחת כיוון שמכל epoch המודל לומד מנתוני האימון ומעדכן את המשקלים בהתאם על מנת למזער את פונקציית הloss. לכן ככל שיש יותר epochים המודל מקבל יותר הזדמנויות להתאים את הפרמטרים שלו לנתוני האימון וכך להיות מדויק יותר עד לפתרון האופטימלי.

וכן ישנה עלייה בaccuracy כיוון שככל שהמודל לומד יותר מנתוני האימון במהלך כל epoch הוא משתפר בביצוע התחזיות. השיפור הנ"ל ביכולות החיזוי מוביל לדיוק גבוה יותר, כמצופה.

Part 3-Neural Network with One Hidden Layer

בחלק זה אימנו Dataset זהה Fashion-MNIST בצורה שונה- על ידי רשת נוירונים בעלת one hidden layer. באופן דומה לאימון בשלב הקודם, על מנת להגיע לסט הפרמטרים אשר מניב את התוצאות המיטביות השתמשנו בשיטת "ניסוי וטעיה". בתחילה, אימנו את המודל עם סט פרמטרים התחלתי, קיבלנו ערך התחלתי ולא מספיק טוב, על כן המשכנו לאמן את המודל כך שבכל הרצה בדקנו את השפעת שינוי אחד הפרמטרים על תוצאות ה Accuracy עבור הValidation. המשכנו בשלבים עד אשר הגענו לרמת דיוק טובה מספיק.

להלן פירוט התהליך שעשינו ושינוי הפרמטרים בהתאמה:

accuarcy	Drop_rate	Size_hiden_layer	Num_epochs	Batch_size	Learning_rate	L2_layer2	L2_layer 1
84.679	0	100	130	128	0.01	0.01	0.01
84.232	0	100	100	128	0.01	0.01	0.01
84.392	0	100	130	128	0.01	0.02	0.02
84.789	0	100	130	128	0.01	0.005	0.005
85.482	0	150	130	128	0.01	0.005	0.005
86.277	0	150	150	128	0.01	0.005	0.005
86.589	0	150	150	64	0.01	0.005	0.005
86.580	0	150	150	32	0.01	0.005	0.005
86.179	0.2	150	150	64	0.01	0.005	0.005
86.611	0.05	150	150	64	0.01	0.005	0.005
87.250	0.05	150	200	64	0.01	0.005	0.005
86.04	0.05	150	200	64	0.03	0.005	0.005
87.446	0.05	150	200	64	0.01	0.001	0.001
85.482	0.05	150	200	64	0.007	0.001	0.001
86.583	0.05	180	200	64	0.01	0.001	0.001
86.875	0.05	150	200	64	0.01	0.0007	0.001
88.205	0.05	150	200	64	0.01	0.005	0.001
87.518	0.05	150	200	64	0.01	0.008	0.001
88.303	0.05	150	250	64	0.01	0.005	0.001
87.929	0.08	150	250	64	0.01	0.005	0.001
87.786	0.02	150	250	64	0.01	0.005	0.001
88.321	0.05	150	250	64	0.01	0.005	0.0007
88.232	0.05	150	250	64	0.01	0.005	0.0005
88.545	0.05	150	300	64	0.01	0.005	0.0007
88.205	0.05	150	350	64	0.01	0.005	0.0007

פירוט השפעת hyperparameters על רמת דיוק המודל:

תחילה בחרנו שרירותית את הנתונים ההתחלתיים על מנת לקבל איזושהי תמונה כללית על המודל ונקודה התחלתית. ניתן לראות שהגענו לערך: Accuracy = 84.679%

- השפעת ערך ה Learning Rate על דיוק המודל.

כידוע קצב הלמידה משפיע ישירות על המהירות וההתכנסות של אלגוריתם האופטימיזציה המשמש לאימון, בחירת קצב למידה מתאים יכולה להשפיע משמעותית על הדיוק וביצועי המודל. Learning Rate נמוך מדי עלול להביא לפתרון לא אופטימלי נוסף על זמן התכנסות ארוך- Underfitting. מאידך, Learning rate גבוה מדי עלול לגרום למודל להתנודד סביב הפתרון האופטימלי ולהוביל לOverfitting. שמנו לב כי בהעלאת ערך ה Learning Rate לערך 0.03 והורדת ערך ה Learning Rate לערך 0.007 קיבלנו ירידה משמעותית בדיוק

המודל, רמת הדיוק היתה מטיבית בערך lr = 0.01 - זאת בהתאם למצופה שהlr לא צריך להיות גבוה מדי אך לא נמוך מדי.

- לאחר מכן, בדקנו את השפעת ערך L2 regularization של כל אחת מהשכבות על דיוק המודל.

פרמטר זה "מעניש" משקלים גדולים מדי ומנמיך את ערכם ובכך מונע Overfitting, מביא לשליטה מיטבית במורכבות הדגם ושיפורו. תחילה קבענו L2_layer1 =0.01 =L2_layer2 ניסינו להעלות את ערכו ב0.01 ונוכחנו לגלות כי רמת דיוק המודל ירדה- לכן ניסינו להקטין אותו לערך 0.005 שם קיבלנו רמת דיוק גבוהה יותר ועל כן נשארנו סביב ערך זה.
בהמשך הבדיקות נוכחנו לגלות כי הערך L2_layer2=0.005 , L2_layer1=0.0007 מניבים יחד את רמת הדיוק הגבוהה ביותר.

- בשלב הבא, בדקנו את השפעת גודל Num of Epochs על דיוק המודל.

פרמטר זה מייצג את מספר הפעמים שה training dataset מועבר קדימה ואחורה דרך המודל בתהליך האימון. כמות נמוכה של epochs עלולה לגרום לUnderfitting כיוון שהמודל לא מספיק ללמוד מהdataset וגורם לדיוק נמוך יותר. מאידך, כמות גבוהה של epochs עלולה לגרום לOverfitting כיוון שהמודל לומד רעש מהdataset ולכן ישנה חוסר הצלחה בהכללת הלמידה לdata חדש אשר לא נראה קודם. על כן, התחלנו בepochs 130 והעלינו את הערך בהתאם לרמת הדיוק עד שקיבלנו בepochs 300 את רמת הדיוק המיטבית עבור סט פרמטרים זה.

- בשלב הבא, בדקנו את השפעת גודל הBatch Size על דיוק המודל.

פרמטר זה קובע את מספר הדגימות המעובדות לפני עדכון הפרמטרים של המודל במהלך תהליך האימון. הוא ממלא תפקיד מכריע בקביעת היעילות והאפקטיביות של תהליך האופטימיזציה. הBatch size משפיע ביחס ישר על התכנסות המודל- זאת משום שגדלי קבוצות גדולים יותר גורמים בד"כ לעדכונים יציבים יותר והתכנסות מהירה יותר. כיוון שהן מספקות הערכות מדויקות יותר של הגראדינט של פונקציית הloss. התחלנו בערך Batch Size=128 הקטנו אותו לערך 64 אך שמנו לב כי קיים מעין חסם תחתון בערך 32 בו רמת הדיוק החלה לרדת. על כן הערך האופטימלי מבחינתנו הוא Batch Size = 64 כפי שמוצג בטבלה.

- בשלב הבא, בדקנו את השפעת גודל הSize_hiden_layer על דיוק המודל.

פרמטר זה קובע את מספק הנוירונים בשכבת הhidden layer שברשת הנוירונים שבנינו עבור אימון המודל. מספר קטן של נוירונים בשכבת הhidden עלול לגרום לUnderfitting כיוון שהמודל לא מספיק ללמוד מהdataset את הדפוסים הבסיסיים מהנתונים מה שגורם לדיוק נמוך יותר. מאידך, מספר גבוה של נוירונים בשכבת הhidden עלול לגרום לOverfitting כיוון שהמודל עלול ללמוד לשנן את נתוני האימון הללו ולא להכליל מהם דפוס התנהגות למקרים הכלליים יותר.
כמו כן זמן האימון מושפע מפרמטר זה. Hidden layer גדולה מדי דורשת יותר חישוב במהלך האימון משום שיש המון פרמטרים לעדכן בזמן האימון- מה שגורם להארכת זמן האימון במיוחד ברשתות עמוקות. מאידך, Hidden layer נמוכה מדי דורשת פחות חישוב אומנם אך עשויות להביא לרמות דיוק נמוכות כיוון שלא מספיקות ללכוד את התכונות הרלוונטיות בdataset.
מהטבלה ניתן לראות שהתחלנו בערך Size_hiden_layer=100 העלנו את ערכו עד אשר קיבלנו רמת דיוק נמוכה יותר, והתקבענו על ערך של 150 נוירונים בשכבה הנסתרת.

- בשלב האחרון, בדקנו את השפעת גודל Drop_rate כחלק מתהליך הDropout בשלב האימון על דיוק המודל.

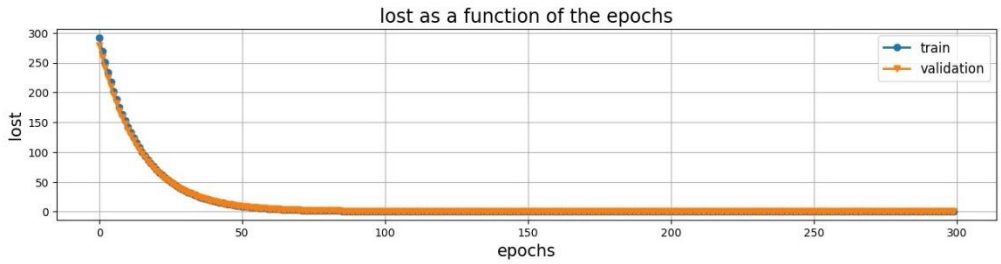
פרמטר זה קובע את ההסתברות לניתוק נוירון X מרשת הנוירונים. על מנת לאמן מספר רב של רשתות בו זמנית , בשלב האימון אנו מייצרים מספר רב של רשתות בו זמנית כך שבכל פעם מנתקים נוירונים אחרים מהרשת(לכל נוירון אנו מגרילים בכל mini batch מספר רנדומלי בין 0 ל 1 ובמידה ומספר זה קטן יותר מהDrop_rate אזי באותו mini batch אנו מנתקים אותו מהרשת).
באופן זה, בכל mini batch נוצרת רשת חדשה אשר דומה לרשת המקורית בשינויים קלים. דרך זו מאפשרת למנוע Overfitting בשלב אימון המודל כיוון שזה מונע מהמודל הסתמכות וקיבעון לסט מסוים של תכונות. כמו כן, שיעור drop out גבוה מדי עלול להוביל לחוסר התאמה- כיוון שיש יותר מדי יחידות שמנותקות בשלב האימון מה שמגביל את יכולת המודל ללמוד את ה datasetבצורה נכונה. מאידך, שיעור drop out נמוך מדי עלול שלא לספק רגולריזציה מספקת. על כן התחלנו בשיעור אפסי של drop out ועם התקדמות תהליך האימון העלנו את ערכו על מנת לקבל רמת דיוק גבוהה יותר, עד שהגענו לערך Drop_rate=0.05 שהניב תוצאה טובה מספיק.

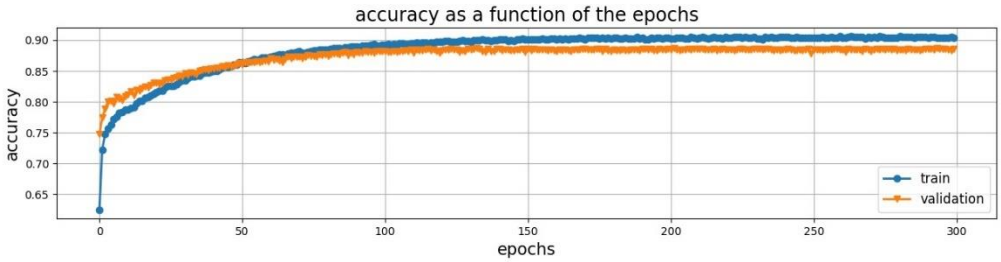
- במהלך קביעת סט הפרמטרים שמניב רמת דיוק מיטבית בדקנו את השפעת ביצועי פונקציית אקטיבציה שונות כגון: tanh וReLU, הגענו למסקנה כי פונקציית הReLU מניבה את הביצועים הטובים ביותר. על כן השארנו אותה בלבד בקוד שלנו.

בסה"כ הגענו לרמת הדיוק המיטבית Accuracy validation = 88.545% בשילוב סט הפרמטרים הבא: Num of Epochs=300 Batch Size=64 Lr=0.01 L2_layer2=0.005
L2_layer1=0.0007 Drop_rate=0.05 Size_hiden_layer=150 כדלהלן:

```
C:\Users\avidan\Desktop\Deeplearning_exc2\.venv\Scripts\python.exe C:\Users\avidan\Desktop\Deeplearning_exc2\exc2.py
Epoch: 300 Train Loss: 0.438 | Train Accuracy: 90.386% Validation Loss: 0.479 | Validation Accuracy: 88.545%
```

מצורף להלן גרפים המתארים את ערכי הloss וה accuracy כפונקציה של מספר ה epochs עבור סט הTrain והValidation:





מהגרפים לעיל ניתן לראות כי ככל שמספר הepochים עולה ← loss יורד, accuracy עולה. בהתאם לתיאוריה שהסברנו לעיל וכפי שציפינו שיקרה. עם הגדלת מספר הepochים המודל הופך מאומן יותר, מכיר את הdata בצורה טובה. הloss פוחת כיוון שמכל epoch המודל לומד מנתוני האימון ומעדכן את המשקלים בהתאם על מנת למזער את פונקציית הloss. לכן ככל שיש יותר epochים המודל מקבל יותר הזדמנויות להתאים את הפרמטרים שלו לנתוני האימון וכך להיות מדויק יותר עד לפתרון האופטימלי.

וכן ישנה עלייה בaccuracy כיוון שככל שהמודל לומד יותר מנתוני האימון במהלך כל epoch הוא משתפר בביצוע התחזיות. השיפור הנ"ל ביכולות החיזוי מוביל לדיוק גבוה יותר, כמצופה.

כמו כן, נשים לב כי אחוז הaccuracy הנמדד עבור הtrain הינו גבוה יותר מאשר אחוז הaccuracy הנמדד עבור סט הvalidation, הסיבה המרכזית לכך הינה שהמודל רואה את סט הtrain הכי הרבה (כל mini batch ולא כלepoch כמו הvalidation) ולכן המודל יותר מותאם (overfit) עבורו ומצליח לדייק באופן מירבי יותר מאשר סט הvalidation.

Summary

ניתן לראות כי על ידי שימוש ברשת נוירונים בשלב האימון הגענו לרמת דיוק טובה יותר של המודל. הדבר נובע מכמה סיבות:

1. מורכבות המודל- Neural Networks מורכבות יותר, בנויות משכבות של נוירונים המחוברים זה לזה, מה שעזר למודל ללמוד גבולות החלטה לא ליניאריים ולקלוט דפוסי התנהגות מורכבים בDataset הנתון. זאת לעומת Logistic Regression- מודל ליניארי שלומד גבולות החלטה ליניאריים בלבד.
2. ייצוג תכונות מורכבות- Neural Networks מסוגלות ללמוד תכונות בצורה טובה יותר הודות לשכבות המרובות שבהן. יש ביכולתן לגלות ולחלץ תכונות רלוונטיות ומורכבות מנתונים גולמיים- דבר שמסייע מאוד בdataset מורכב. זאת לעומת Logistic Regression- אשר מסתמך על שילובים ליניאריים של תכונות שעלולים להיות לא יעילים בהבנת מבנה הנתונים כאשר הdataset הינו מממדים גבוהים או לא ליניאריים.
3. קיבולת וגמישות- Neural Networks בעלות קיבולת וגמישות גבוהה יותר בזכות המבנה המורכב והגמיש שלהן שמסייעים להם בקליטת דפוסים מורכבים, בניגוד ל Logistic Regression שהיא מוגבלת לייצוג נתונים בעלי קשרים ליניאריים בלבד.

אי לכך ובהתאם למצופה קיבלנו כי רמת הדיוק הגבוהה יותר התקבלה בייצוג ע"י Neural Networks .

הערה לחלק 2 ו3:

במהלך פונקציית הTRAIN שבה אימנו את NN ואת ה multi-class logistic regression classifier
בכל סוף כל EPOCH בדקנו אם ערך הACCUARCY על סט הVALIDATION גדול מהערך המקסימלי של הACCUARCY שקיבלנו עד עתה, אם כן אנו שומרים את המשקלים הנוכחיים במשתנה עזר.
בסוף פונקציית הTRAIN של המודל אנו שומרים בתוך קובץ את המשקלים הטובים ביותר של המודל שקיבלנו בתהליך האימון.
לפני שאנו מבצעים את הforward pass בשלב הTEST אנו טוענים את ערכי המשקלים הטובים מתוך הקובץ אל תוך המשתנים המתאימים במודל שלנו ורק לאחר מכן מפעילים את ה forward pass ומקבלים את הפרדיקציות, אנו בוחרים לבצע זאת על מנת לאתחל את המודל עם המשקלים הטובים ביותר שהתקבלו בשלב האימון, שהם אינם דווקא אלו שהתקבלו מהEPOCH האחרון שבצענו בשלב האימון.