חלק ראשון – קבוצת דאטה ראשונה

- תיאור רשת

רשת נוירונים קונבולוציונית (CNN) לביצוע סיווג בינארי (smoking / not smoking) על תמונות בגודל 256X256

מבנה

- שכבות קונבולוציה(Convolutional Layers)
  - 3שכבות קונבולוציה עם ReLU להפעלת מאפיינים מהתמונות.
  - כל שכבה משתמשת ב-32, 64 ו-128 פילטרים בגודל 3×3עם padding=1 כדי לשמור על גודל הפלט.
- שכבות MaxPooling
  - לאחר כל שכבת קונבולוציה יש MaxPooling בגודל 2X2 להקטנת המימדים בחצי.
- שכבות(Fully Connected (Dense Layers
  - שכבה ראשונה עם 128נוירונים ,המשמשת ללמידת ייצוגים עמוקים יותר.
  - שכבת פלט עם נוירון 1 בלבד עמSigmoid, כי מדובר בסיווג בינארי.
- Dropout עבור מניעת Overfitting
  - התחלתי ב (0.5) Dropout -להקטנת תופעת Overfitting.
- קצב למידה של 0.001
- Epochs – 10 (בגלל מגבלת CPU והסקות מתרגילים קודמים)
- Batch size של 32

עיבוד גודל התמונות

- התמונות עוברות שינוי גודל ל-256X256
- אחרי שלוש שכבות קונבולוציה עם MaxPooling מתקבל גודל פיצ'רים 32X32.
- לכן, הפלט מועבר ל Fully Connected-בגודל 131072 נוירונים (128X32X32).

פונקציית transform

- התאמת גודל התמונות
- מעבר למבנה של tensor
- אפשרות של נורמליזיית טווח ערכי הפיקסלים ל [-1,1] וaugmentation על ידי RandomRotation של pythorch שנשתמש בהן במידת הצורך

הוספת דאטה – בקובץ קוד נפרד TestingData.py יצרתי תיקייה של דאטה משולש לטובת data augmentation במהלך הניסויי Training_stageA_Augmented עבור קבוצת הtrain

מטרה

המודל מיועד ללמוד להבדיל בין תמונות של מעשנים (1) ולא מעשנים (0)

תיקון חשוב

Validation

בתוך הפונקציה train_model, אחרי שהמודל מסיים ללמוד מ X_train -אנחנו רוצים לבדוק **כמה טוב הוא עובד על** X_val.

המטרה :לראות אם המודל מצליח לנבא נכון את הy_val (התוויות של הוולידציה).

איך הוולידציה משתלבת באימון המודל?

1. תחילה המודל מתאמן עם X_train, y_train.
2. לאחר כל אפוק (epoch) בודקים **כמה טוב הוא מצליח על X_val, y_val**
3. אם התוצאה עלX_val גרועה בהרבה מזו של X_train כנראה שיש **Overfitting**
4. אם נראה שהדיוק משתפר גם ב X_train-וגם ב X_val, -סימן שהמודל לומד טוב.

מסקנות מהאימון הראשוני -

- הדיוק (accuracy) על **הסט של האימון** עלה מהר מאוד, מ **57.3%** -**ל-96.82%** תוך מספר אפוקים בודדים.
לעומת זאת ,הדיוק על **הסט של הוולידציה** משתפר הרבה פחות (מ-37.5% ל-75%), מה שמעיד על .**Overfitting**
- בהרצה ראשונה התחלנו עם accuracy של 57.3% וברצה שלישית כבר הגענו ל- 96.82% זהו קצב גבוה לעומת התוצאות על validation הן 37.5% בראשונה ו75% בהרצה שלישית קצב למידה איטי יותר.

<u>ניתוח גורמים שונים</u>

- <u>עומק גדול מידי?</u> - יכול להיות ש 3 שכבות זה עמוק מידי עבור כמות כזו של תמונות וכן כמות פרמטרים, יש יותר מידי וזה יכול להשפיע על למידת מאפיינים כלליים. נקודה לשינוי ובדיקה
- <u>אין מספיק Augmentation ?</u> – אם התמונות דומות מידי הרשת יכולה ללמוד דפוסים ספציפיים ולא כלליים ואולי דרושה הוספת דאטה. מאידך **כשבוחנים את התמונות** הן די מגוונות עם הרבה פריטים ברקע/דיוקן/תנועות ידיים שונות/מנחי גוף שונים/קירובים שונים/תאורות שונות ועוד. אפשר להוסיף הטיה רנדומלית של התמונות בכל הרצה וכן להכפיל ולהטות את הדאטה (עבור train בשני המקרים)
- <u>חוסר ב Regularization-משמעותי ?</u> – אולי שכבת dropout(0.5) אחת לא מספיקה כדי למנוע overfitting חזק. כמו כן ניתן להוסיף בpytorch (weight decay) **L2 Regularization** בזמן האימון כך שיהיה תיעדוף למשקלים קטנים יותר ומונע מהמקטעים של הרשת להפוך לגדולים מידי ולהגיע לoverffiting
נקודה לשינוי ובדיקה
- <u>קצה למידה גבוה מידי?</u> – אולי הקצב גבוה מידי וזה קופץ לנקודת מינימום מהירה יותר נקודה לשינוי ובדיקה
- <u>קבוצות batch?</u> – אולי הקטנת הקבוצות תוסיף מעבר תחת relu נוספות וזה יעזור עם המשקלים

- שינויים מוצעים:
  - Normalization לגודל הפיקסלים
  - הוספת augmentation על ידי RandomRotation של pytorch (מוסיף הטיה לתמונה של X מעלות ללא שכפול) ניתן לשקול הוספת תמונות מוטות
  - הוספת data עי שכפול והטיית השכפולים
  - הגדלת dropout קיים
  - הוספת dropout נוספת אחרי fc1
  - הקטנת קצב למידה
  - להוסיף (weight_decay) **L2 Regularization** לאופטימיזר.
  - בגלל מגבלות CPU לא אוסיף עצירה מוקדמת אם אין שיפור על ואלידציה כי אנחנו רק על 10 אפוקים
  - אימונים נוספים לקובץ המשקולות

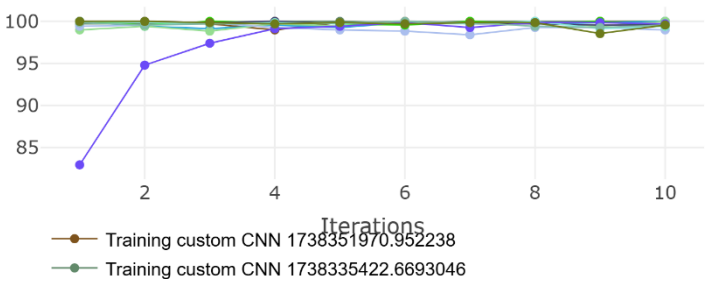
<u>בדיקה מקדימה על 2 תמונות</u>

- 100% על 2 תמונות שבחרתי כsanity check לראות שאני בכיוון הנכון
- התוצאה נשמרה לאורך כל חלק זה כולל הוספת שינויים ברשת

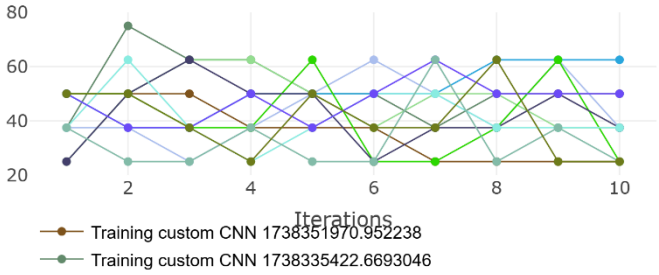
<u>בדיקה ראשונית על טסט</u> – 70% - התוצאה נשמרה גם אחרי אימון ממושך, הייתה רידה בשלב 4 ועלייה חזרה ל805 בשלב 5

תחילה לפני סבבי השינויים עשיתי נורמליזציה לנתונים

- בפונקציית ה – transform הדלקנו את האופציה שמעבירה את גודל הפיקסלים להיות בין -1 ל1, זה אמור להשפיע על המשקולות שמחושבות כך שערכיהן לא "יתפוצצו"
- חל שיפור caccuracy של הvalidation אבל לא משמעותי



Training accuracy initial runs1 Figure



Validation accuracy initial runs2 Figure

<u>ניסויים ושינויים</u>

- לאורך כל האימונים והשינויים הtest עמד על 60%-70% עד שלב 5
- המיקוד היה להשיג גדילה מתואמת של accuracy של train וvalidation כדי לוודא שאין overfitting ובמקבים תוצאות טובו על הtest
- בכל שלב השינויים הקודמים נשארו אלא אם צויין אחרת

שלב ראשון -

אנחנו רואים פער גדול בין דיוק האימון לדיוק הולידציה ,זה סימן טוב לOverfitting-

נתחיל עם:

- הוספת **L2 Regularization** (weight decay) עם 0.0001 בזמן האימון כדי לגרום לרשת להיות יותר "זהירה" בלמידה. המודל יעדיף מקלים קטנים יותר ("ענישה" לשמקלים גדולים מידי) דבר שאמור להתמודד עם overfitting.
- הגדלת ה Dropout ל0.6

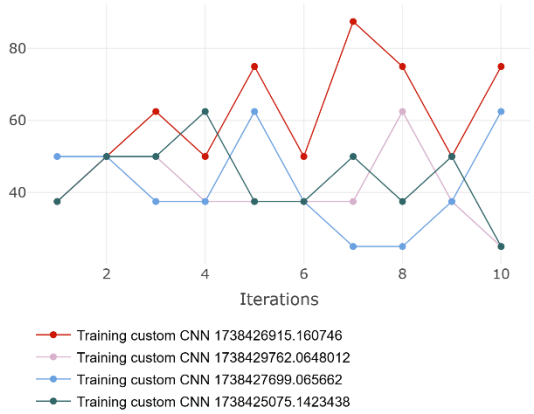
תוצאות-

- Loss של validation צנח מערכים של 4-5 ל1.2 בהרצה ראשונה, אבל לא היה ניכר שיפור בaccuracy או בתוצאות של הtest שעדיין עומדות על 70%
- בהרצה נוספת ניכר שינוי מהותי בaccuracy של הvalidation מערכים בין 20%-50% עברנו ל50%-90% בהרצה נוספת בלבד

הרצות נוספות לא הניבו עלייה בaccuracy של הvalidation והיה נראה שעדיין יש עלייה חדה בaccuracy של הtrain לעומת קבוצת הvalidation מה שמעיד שוב על overfitting ולכן עברתי לשלב הבא של dataAugmantation

- תוצאות accuracy ו-loss לאורך השלב הראשון – יש שיפור אבל עדיין מעיד על overfitting

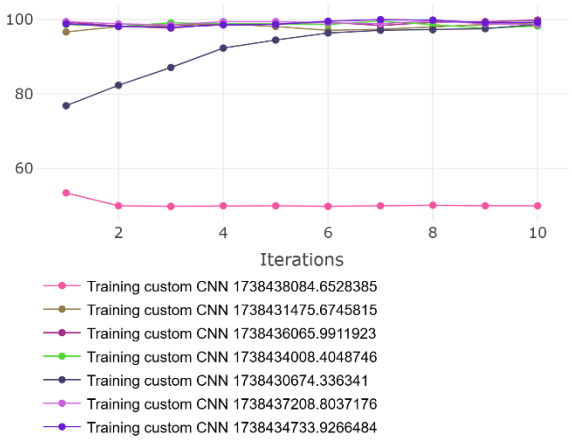
*Training accuracy **Loss: 0.09-0.0143** Figure*



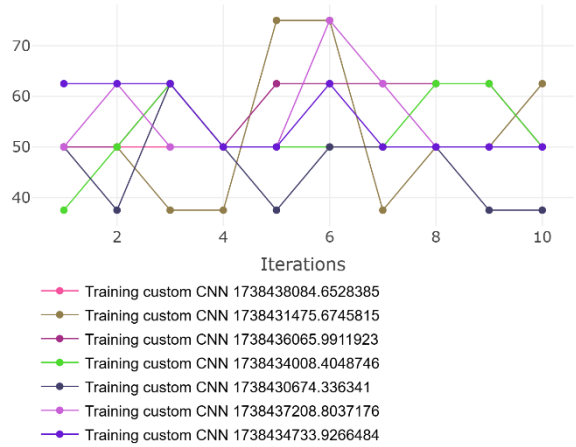
*Validation accuracy **Loss: 6.44-0.894** Figure*

שלב שני – קבוצת דאטה ראושנה

- **Data Augmentation** – שינוי פונקציית transform שאני מעבירה על הdata והוספת הטיה לתמונות (בכל הרצה) להקשות על המודל "לשנן" תמונות על ידי transforms.RandomRotation(10) (10 מעלות), מוסיף שונות לסט האימון ועוזר למודל ללמוד דפוסים כלליים יותר. לא משכפל א ,התמונות אלא רק מטה בכל הרצה (מהסביבה הפשוטה שבחינת הדאטה הראתה גיוון מאוד גדול בין התמונות רקע\עומק\קירוב\פרטים\אור וכו ולכן בשלב זה לא מיהרתי לשכפל את התמונות)
- תוצאות
  - בתחילה ניכרר שיפור (accuracy של validation נע בין 50%-80%), אימונים נוספים החזירו למצב קודם (30%-50%) במקביל accuracy של train עומד על 99% אך הloss של הvalidation במגמת ירידה
  - הורגש שיש צורך באימון המשקלים בצורה יותר אינטסנסיבית כי הייתה ירידה קבועה בloss של הvalidation ויותר אחוזים גבוהים בaccuracy שלו למרות הירידות, על פני 10 epochs.
  - למרות זאת עדיין מצב מובהק של overfitting



Training accuracy ***Loss: 0.044-0.010*** *5 Figure*



Validation accuracy ***Loss: 2.11-1.32*** *6 Figure*

שלב שלישי –

- הוספת dropout של 0.4 אחרי fc1
- שינוי קצב למידה מ0.001 ל-0.01
  - ניכר שaccuracy של הvalidation והtrain באותו קצב, אמנם יחסית נמוך (התחיל מ50%) אבל זה סימן טוב של ניקוי overfitting וייתכן שעוד אימונים יביאו את הרשת למצב טוב יותר
  - **מצד שני נתוני loss לא הגיוניים עבור שניהם הloss עומד על 50 לערך**

שלב רביעי – עבודה עם lr

- הקטנת lr ל0.0005 עדיין הloss של שתי הקבוצות עומד על 50
- שינוי חזרה ל0.001 ואימונים חוזרים כדי להתייצב על loss מקורי – חוזר לנתונים התחלתיים של ספרות בודדות ומתחת ל1 עבור train
- מה שהתהליך כן יצר זה **קצב גדילה תואם בין הtrain לvalidation**
- לראשונה השגנו 100% accuracy על validation וloss של 0.5

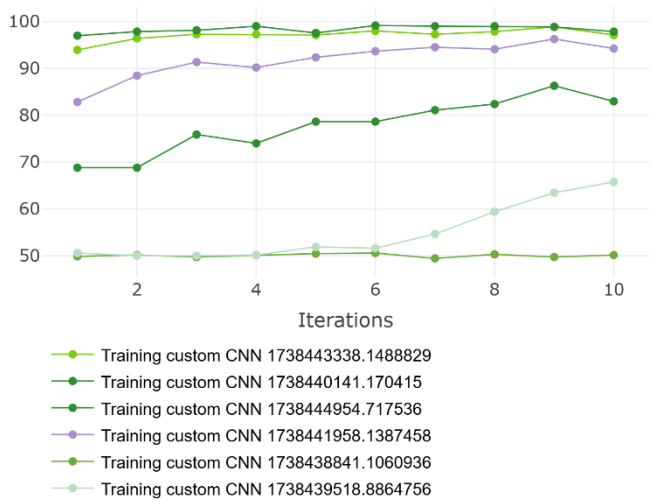
```
Epoch 7/10, Loss: 0.0095, Accuracy: 100.00%
Validation Loss: 1.7270, Validation Accuracy: 50.00%
Model weights saved as 'cnn_model.pth'
Epoch 8/10, Loss: 0.0080, Accuracy: 99.86%
Validation Loss: 1.7888, Validation Accuracy: 50.00%
Model weights saved as 'cnn_model.pth'
Epoch 9/10, Loss: 0.0142, Accuracy: 99.13%
Validation Loss: 1.8053, Validation Accuracy: 50.00%
Model weights saved as 'cnn_model.pth'
Epoch 10/10, Loss: 0.0202, Accuracy: 99.28%
Validation Loss: 1.8113, Validation Accuracy: 50.00%
Model weights saved as 'cnn_model.pth'
Training complete! Model weights saved as 'cnn_model.pth'
```

Figure 7 לפני, אימונים קודמים לפני שינוי בlr חוסר תאימות מצב של overfitting

```
Epoch 7/10, Loss: 0.4408, Accuracy: 81.07%
Validation Loss: 0.4423, Validation Accuracy: 87.50%
Model weights saved as 'cnn_model.pth'
Epoch 8/10, Loss: 0.4032, Accuracy: 82.37%
Validation Loss: 0.4003, Validation Accuracy: 87.50%
Model weights saved as 'cnn_model.pth'
Epoch 9/10, Loss: 0.3298, Accuracy: 86.27%
Validation Loss: 0.3588, Validation Accuracy: 100.00%
Model weights saved as 'cnn_model.pth'
Epoch 10/10, Loss: 0.3734, Accuracy: 82.95%
Validation Loss: 0.4594, Validation Accuracy: 75.00%
Model weights saved as 'cnn_model.pth'
Training complete! Model weights saved as 'cnn_model.pth'
```

Figure 8 אחרי, מצב של תאימות בין הtrain לvalidation

- בהתחלה זה היה נראה שיש עלייה מתואמת בין הaccuracy של הtrain והvalidation וחשבתי לאמן עוד **כי התוצאה על הtest ירדה ל60%** אבל **אימונים נוספים** התחילו להחזיר למצב שראינו קודם שהוא accuracy נורא גבוה לtrain שבעוד שהvalidation לא עולה באותו הקצב
- ניתן לראות שבהרצה האחרונה הaccuracy של הvalidation חוזר לאזור ה50% למרות שהיה בעל ערכים גבוהים בסשן הנוכחי

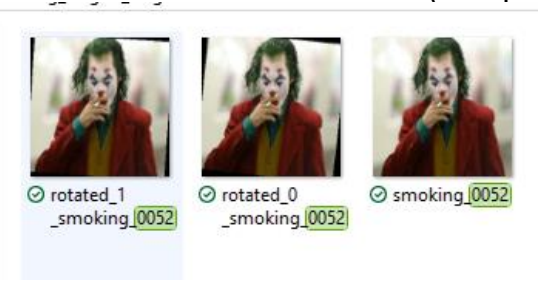


*Training accuracy **Loss: 0.59-0.061**9 Figure*

*Validation Accuracy **Loss: 0.83-0.459**10 Figure*

Figure 11תרשים המתאם תוצאות כלליות להמשך האימונים

- **אימונים נוספים העלו את הtest ל70% ללא התקדמויות נוספות**
- **המסקנה הכללית** היא שאולי **מאוד קשה להשיג יציבות עם מעט דאטה ולכן ננסה data augmentation** כאשר נשכפל ונטה מעט את את התמונות המשוכפלות (בשונה ממה שעשינו בהתחלה שהטינו בכל הרצה ב10 מעלות את התמונות הקיימות)



שלב חמישי –

- Data augmentation כמפורט
- הטיה של 10 מעלות והוספת 2 גרסאות מסובבות לכל תמונה
- סה"כ 2067 תמונות
- הבדל משמעותי
  - Loss של validation קטן
  - Accuracy של validation מתייצב על ערכים גבוהים יותר (75% במקום 50%-60%) (קבוצת train ללא שינוי, מה שמצביע על כך שיש הצלחה מול overfitting)
  - Test עולה ל80%!!!

Validation accuracy12 Figure

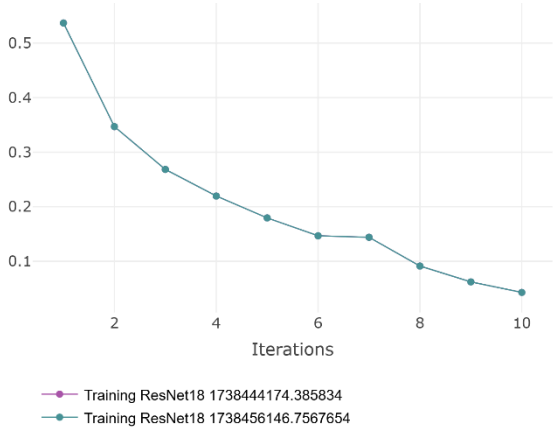
חלק שני –

תיאור המודל והרצה ראשונה, שלב ראשון-

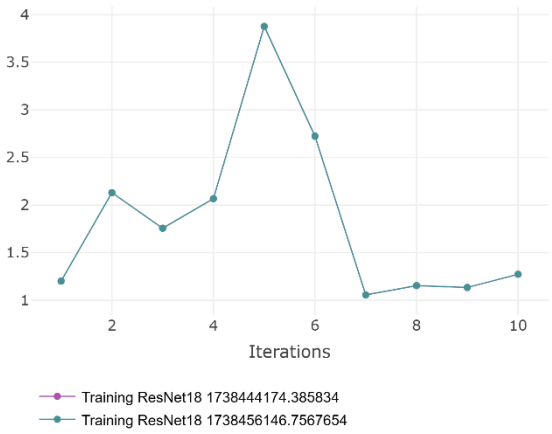
- שימוש בresnet18 ושמירה על מאפיינים שלמדנו במודל הראשון
  - נורמליזציה שלעדכי התמונה לפי resnet18 mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]
  - Batch size של 32 ו10 epochs
  - הערה – שימוש בresnet18 מחזיר את אותם ערכים עבור הרצות בודדות מכיוון שהרשת כבר מאומנת, הסתפקתי ב2 הרצות לכל שינוי ובדיקה

השוואה -

- תוצאות התחלתיות הרבה יותר טובות מהמודל בחלק 1 ,
- **loss נמוך מההתחלה ובאופן כללי עבור קבוצת הvalidation** (~1.2 בעוד שבחלק 1 ראינו loss מעל 5 בהרצות הראשוניות)
-
- **loss נמוך מההתחלה ובאופן כללי עבור קבוצת ה train** (התחלה עם 0.5 וירידה ל0.04)

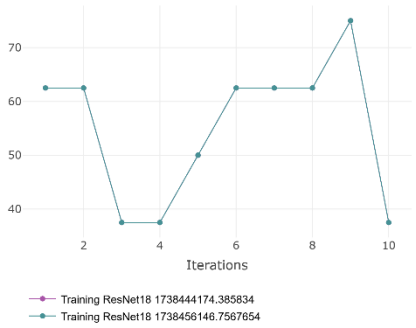


Training Loss round #113 Figure

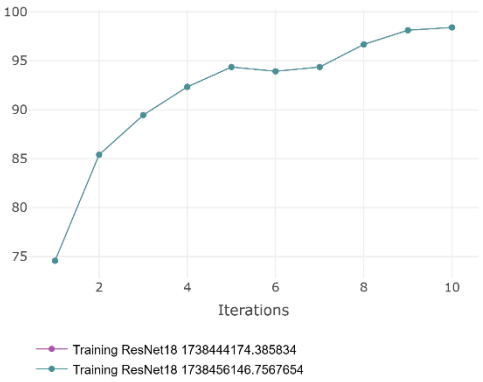


Validation Loss round #1 14 Figure

- **Accuracy של הvalidation והtrain ברוב המקרים מתואם** ועולה באותו קצב (יותר טוב באופן כללי מחלק 1)
- ישנו **שיפור בloss של ה validation וה train לאורך האימון** עם זאת השיפור המשמעותי שייר לtrain



validation Accuracy round #1 15 Figure



Training Accuracy round #1 16 Figure

- **תוצאות ראשונית של 90% על ה test**

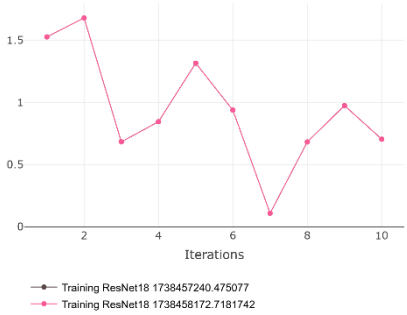
שלב שני-

- Data augmentation של הטיית התמונה בכל הרצה לא שכפול בשלב התחלתי

השוואה -

- התנהגות דומה של עליה בaccuracy של הvalidation ובמקביל ירידה בaccuracy של הtest
  - ייתכן בגין התפלגות שונה בtest
- תוצאות הרבה יותר טובות של הvalidation לעומת ללא הטיה שהוספנו
  - Loss: 1.27 to 0.7 , Accuracy: average of 55% to average of 67.5%
- על קבוצת הtrain אין שינוי משמעותי בloss והaccuracy

Validation accuracy round #2 18 Figure



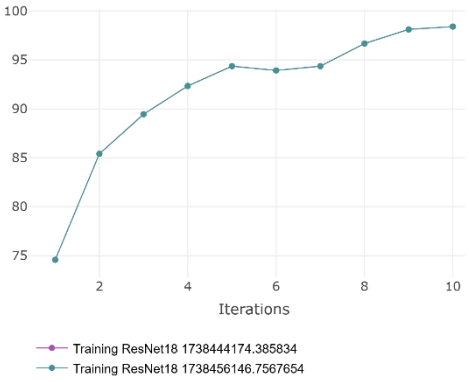
Validation loss round #217 Figure

שלב שלישי –

- שימוש בתיקיית התמונות של **data augmentation** המכילה שכפולים מוטים כמו בחלק 1 (בשונה ממה שעשינו בהתחלה שהטינו בכל הרצה ב10 מעלות את התמונות הקיימות)

השוואה-

- גם פה למרות הוספת data נותני קבוצת train נותרו דומים (השוואה accuracy/train)



train accuracy round #1 19 Figure

train accuracy round #2 21 Figure

train accuracy round #3 20 Figure

- אין שיפור של השינויים על קבוצת validation,
  - Loss: 1.27 to 0.7 **to 1.4** , Accuracy: average of 55% to average of 67.5% **to average of 65%**
- על קבוצת test קיבלתי 90%
- נראה שגם שינויים של שילוש תמונות פחות משפיע על מדול מאומן היטב (די הגיוני)

מסקנות כלליות –

(קבוצת תמונות ראשונה)

- **הוספת דאטה היה הפרמטר עם המשמעות הגדולה ביותר**
- **אפשר לשלב** גם הוספה של דאטה וגם שימוש בכלי ההטייה של pytorch, RandomRotation(10) שמוסיף גם אפקט אקראי כי הוא עושה את זה בכל הרצה, הגעתי לתוצאות טובות ללא זב אבל יש אינספור שינויים שאפשר לעשות ובגלל מגבלת זמן התמקדתי בכמה שינויים
- **מודל מאומן מתחיל מנקודת פתיחה הרבה יותר טובה** מהמודל הראשון ולמרות שעשינו עליו הרבה שינויים והתחלנו עם מבנה שנחשב "טוב" – dropouts, נורמליזציה וכו
- **המודל המאומן מגיע לתוצאות טובות יותר מהר יותר עם כל שינוי שעשינו** - ללא שינויים-> הוספת הטייה--> הוספת דאטה
- בחלק הראשון יש משמעות לכל היפר פרמטר שמשנים, המודל מאוד רגיש במיוחד אם ישנה קבוצת תמונות קטנה יחסית
- **במודל המאומן** היו תוצאות כלליות יותר טובות לtrain אבל לאורך השינויים שעשינו לא היה שינוי משמעותי, לעומת זאת ראינו **שיפור בנתוני הvalidation לאור חלק השינויים** שעשינו במודל המאומן
- **הוספת דאטה לא בהכרח משפיעה על מודל מאומן היטב**

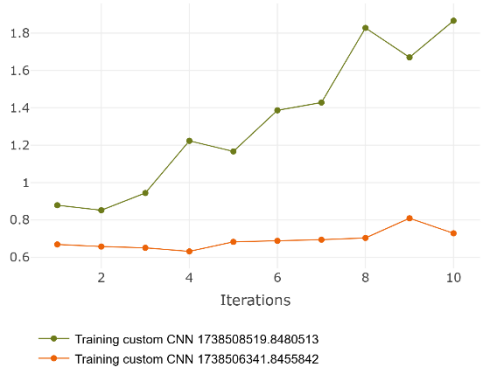
<u>בחינת קבוצת דאטה שניה</u>

בחלק זה אבחן את המודל שלי לפי הפרמטרים שהשגנו בחלק הראשון שהראו שהם עובדים טוב (lr=0.001, הטיה של התמונות בהרצה עצמה (סוג של data augmentation), normalization של הנתונים, הגדלת dropout ל0.6, שימוש בdropout אחרי fc1, שימוש ב**Regularization** לאופטימייזר, הנתונים בחלק זה עברו data augmentation (הטיות/סיבובי מראה/צבעים וכו) ולכן לא אעשה שימוש בתיקייה המורחבת של דאטה שיצרתי עבור קבוצת הדאטה הראשונה

מודל שלי:

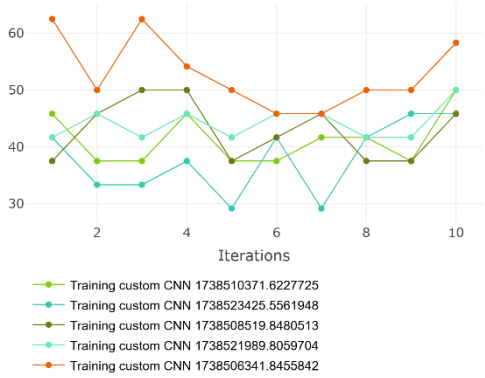
- **אנחנו רואים שיש התמודדות עם overfitting אבל יש צורך באימון ושינוי**
- הנתונים בהתחלה לא הכי טובים אבל לפחות מתואמים (accuracy סביב ה50% וloss סביב ה0.6 לשניהם) בין הtrain לvalidation וקצב הצמיחה זהה מה שמעיד שאנחנו מתמודדים טוב יותר עם overfitting עם קבוצת התמונות החדשה
- לעומת הוספת התמונות שאני עשיתי בחלק הקודם שכולל רק שכפול והטייה של 10 ו20 מעלות, **הגיוון בקבוצה הנוכחית (סיבובים, סיבוב מראה, הטיות, צבעים) הרבה יותר אפקטיבי לאימון** ונראה שיש התמודדות טובה יותר עם Overfitting
- התקבלו 46.67% על קבוצת הtest בהרצה ראושנית, העליתי dropout שאחרי fc1 ל0.6 וביטלתי הטיה רנדומלית של התמונות מאחר שהדאטה כבר קשה הרבה יותר ולקח לאימון יותר מ20 דק. הנתונים היו טובים יותר והתקבל 56.67% על הtest אבל עם סימן לoverfitting – החל פעם בקצב בין הtrain שטיפס ל90% לvlaidation שירד ל45% בממוצע כפי שניתן לראות:

Validation accuracy22 Figure

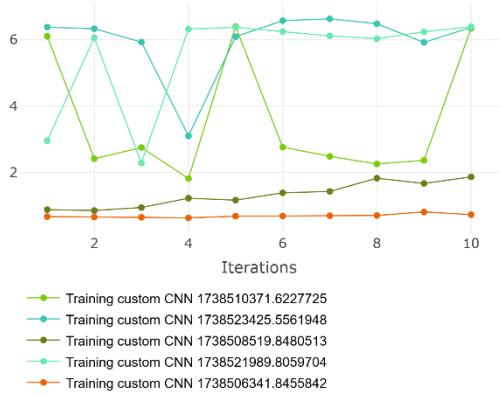


Validation loss23 Figure

- הורדתי dropout אחרי fc1 ושמתי dropout כללי על 0.5 – ללא שינוי מהותי, הגענו למצב ממוצע בין שני השינויים האחרונים ו53.33% על הtest
- נראה שלמודל ממש קשה ללמוד על קבוצת הdata השניה
- בחינת גודל batch
  - גודל של 54 השיג את אותה תוצאה על הtest אבל נראה שהloss והaccuracy של הvalidation והtrain טובים יותר לעומת זאת לא ניכר שיפור בגודל של 64 ואפשר להניח שזה גדול מידי וגרם למודל להכליל את מה שלמד ולא ללמוד מאפיינים. נתבונן בנתוני הvalidation (נתוני הטסט יותר מידי טובים ומעידיפ על Overfitting ללא שינוי משמעותי 99% וloss=0.02 עבור 64)



Validation accuracy 24 Figure



Validation loss 25 Figure

ירוק בהיר- גודל 54, תכלת- גודל 64

:Resnet18

- התחלתי עם אותם מאפיינים ממקודם של normalization שמתאימה לresnet18
  - std, [0.485, 0.456, 0.406]=mean=[0.229, 0.224, 0.225]
- ו-RandomRotation(10) של 10 מעלות בכל הרצה
- קיבלתי תוצאות טובות יותר, אך לא בהרבה
- המודל המאומן מצליח להתמודד טוב ולמנוע overfitting, קצת הגדילה של נתוני הtrain וה-validation ללא הפרשים מאוד גדולים כמו במודל הראשון
- התוצאות הכלליות לא גבוהות וקיבלנו 56.67% בתחילה על הtest
- עבודה עם גודל batch
  - יכול להיות שגודל batch של 32 כאשר יש לנו 2076 תמונות הוא קטן מידי ומייצר רעש גדול מידי
  - הגדלתי batch ל54 ואז ל64 התוצאות טובות יותר וקיבלנו 63.33% על הtest
  - המודל המאומן מתמודד טוב יותר עם Overfitting אבל עדיין הורגש שיש, הדאטה בחלק השני קשה יותר אבל **שינוי גודל הbatch מתגלה כמשמעותי** (עד כמה שנוכל, עצרתי ב64 בגלל מגבלת CPU) בתמונות 32-אפור, 64- ירוק בהיר, ניתן לראות את ההבדל

Validation loss per batch sizes27 Figure

Validation accuracy per batch sizes 26 Figure

  - ההנחה היא שעבודה עם batch sizes תניב תוצאות מוצלחות יותר
  - התוצאות על הtrain עלו בהתאמה

train loss per batch sizes28 Figure

train validation per batch sizes 29 Figure

שם repository - https://github.com/Ayo1a/CNN-implementation המכיל את כל הקבצים