**למידה עמוקה – תרגיל 3**

מגישים

נועם דיאמנט - 208520262

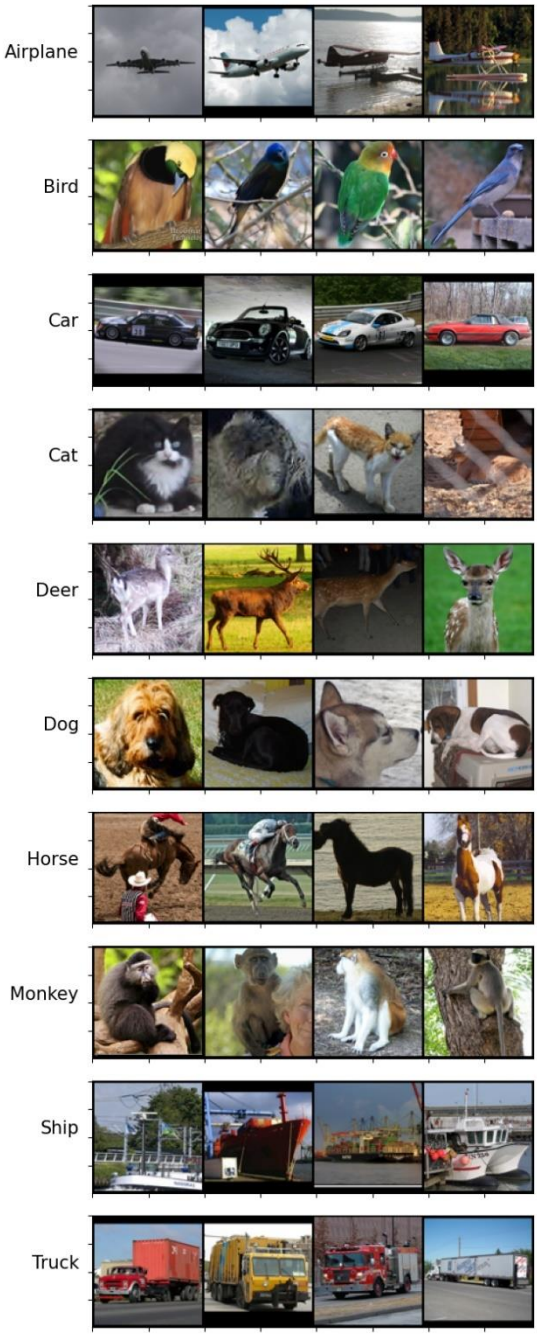
יונתן גולן – 208980888

חלק א

## Part 1 - Visualize the Data (10 Pts)

First, you will need to visualize the data using the Matplotlib package. Plot 4 different examples from each class in a grid of $10 \times 4$ (i.e., 10 rows, each contains 4 different examples from the same class). Label each row with the class name. Add the figure to a report that should be submitted along with your code.

בחלק זה התבקשנו להציג 4 תמונות מכל קלאס עם כותרת הקלאס לידן. למטה ניתן לראות את הפלט שהוצאנו עבור חלק זה, כדרוש.



חלק ב

## Part 2 - Classification with Various Networks (90 Pts)

Here, you will experiment with different types of networks. To reduce the computational complexity we will be working with images of size $3 \times 64 \times 64$. During training apply random cropping for images to size 64, and for test images center cropping to size 64. These operations can be done easily using the PyTorch Transforms class. So, to clarify, the input to the network (both during training and testing) should be an image of size $3 \times 64 \times 64$. Also, in all experiments, during training, you need to apply data augmentation techniques (such as random horizontal flipping). You may choose whichever augmentations you want, but you have to apply at least two augmentations. Describe in the report which augmentations were selected and add one image that was transformed by each augmentation.

Implement the following networks:

1. Logistic regression over flattened version of the images
2. Fully-connected NN with at least 3 hidden layers over flattened version of the images followed by a classification layer. Apply batch normalization and dropout to all hidden layers.
3. CNN with at least two convolution layers and two pooling layers followed by two fully connected layers and a classification layer. Apply batch normalization to the convolution layers and dropout to the fully connected layers.
4. A fixed pre-trained MobileNetV2 as feature extractor followed by two fully connected layers and an additional classification layer. To clarify, you need to learn only the parameters of the task head.
5. A learned pre-trained MobileNetV2 as feature extractor followed by two fully connected layers and an additional classification layer. Here, you need to learn the parameters of the MobileNetV2 as well.

בחלק זה של המטלה, התבקשנו לבצע אוגמנטציות שונות על סט ה – train וה – test לפי ההוראות שמופיעות במטלה. השינויים שביצענו על סט ה – train הם:

1. סיבוב רנדומלי – סיבוב רנדומלי של התמונה בפלוס / מינוס מעלות.

```
torchvision.transforms.RandomRotation(20)
```

2. טרנספורמציה אפינית רנדומלית – טרנספורמציה אפינית רנדומלית של התמונה כאשר מרכז התמונה נותר ללא שינוי.

```
torchvision.transforms.RandomAffine(5)
```

תוצאות הטרנספורמציות הן:

RandomAffine

RandomRotation

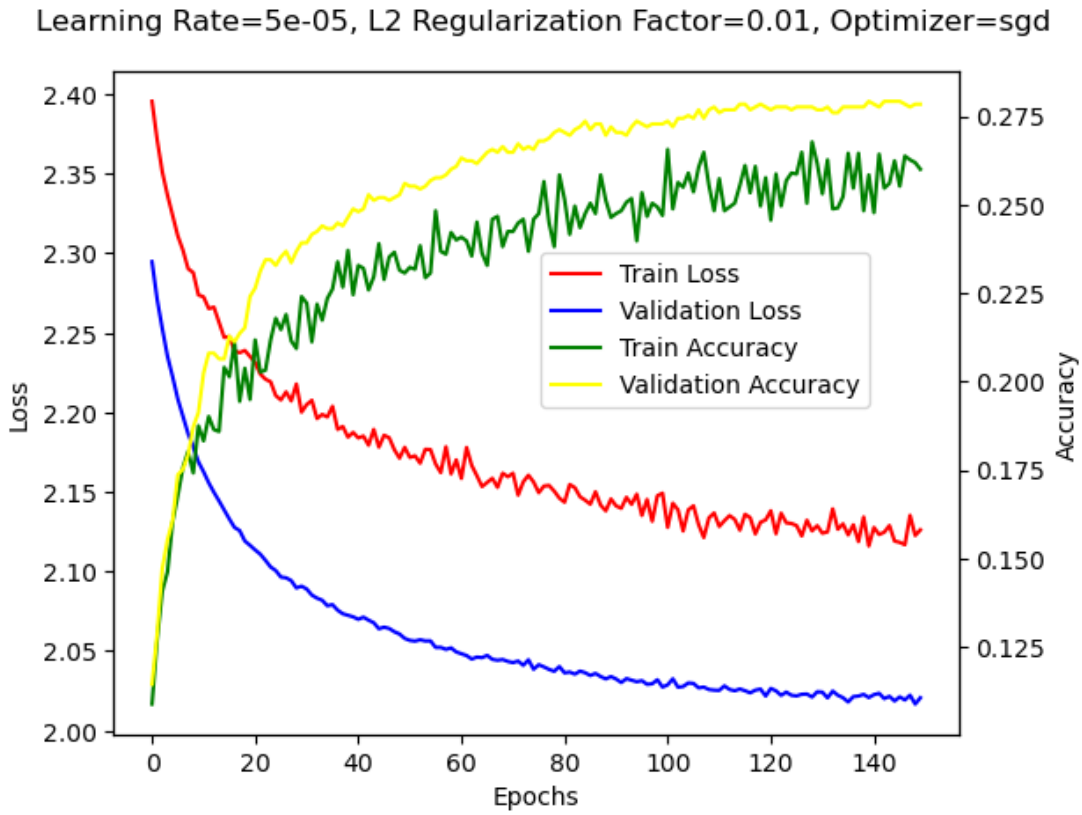
כעת, עברנו על מקרים שונים עבור ההיפר – פרמטרים של הרשתות השונות, התחשבנו בשיקולים השונים המפורטים מטה, וקיבלנו עבור כל רשתות את הפרמטרים הטובים ביותר (הפרמטרים המדויקים בהם השתמשנו עבור כל מודל נמצאים בקוד תחת החלק שבו כתבנו בפירוש את הפרמטרים עבור כל אחד מהמודלים):

1. גודל שכבה במודל (Layer Size) – השיקולים בבחירה זו היו שמצד אחד ככל שהשכבות ברשת יהיו יותר גדולות הרשת תהיה יותר מדויקת, ומנגד אם השכבות גדולות מדי אז יהיה קשה לאמן את הרשת כי יהיו פרמטרים רבים מדי עבור דאטא סט קטן מדי.
בסופו של דבר בחרנו את הגדלים הבאים עבור כ"א מהמודלים הללו:
Fully connected – we chose [200,100,50]
CNN – for the channel we chose [20,64], for the FC we chose: [computation, 100, 50]
ועבור הרשת mobileNetV2 קיבלנו גודל קבוע כאשר היכולת לשנות אותו (לעשות לו tuning) הייתה תלויה בחלק של המטלה.
2. אופטימייזר – כאן כתלות במודל וכפי שמפורט בקוד בחלק הנ"ל, השתמשנו או ב SGD עם מומנטום ו – L2 regularization שונים, או ב – Adam, כפי שניתן לראות בקוד המצורף עבור כל אחד מהמודלים.
3. גודל ה – Batch – כאן יש שיקולים לשני הצדדים. מצד אחד batch גדול יתן תוצאות יותר טובות בדיוק של הפרמטרים ושל הרשת כי אנחנו מעדכנים את הפרמטרים יותר טוב, אבל האימון יקח יותר זמן. מצד שני, btach קטן אמנם יתרום לאימון יותר מהיר, אבל התוצאות שלו יהיו בדיוק יותר נמוך. בסופו של דבר החלטנו להשתמש ב – batch בגודל 128, למרות שגם עבור הגדלים של 64 או 256 קיבלנו תוצאות טובות.
4. קצב למידה (Learning rate) – כמו שראינו בהרצאה ובתרגול, קצב למידה איטי דורש זמן רב עד שהוא מתכנס, ומנגד קצב למידה מהיר מדי עלול לא להתכנס כלל. כתלות במודל וכפי שמפורט בקוד בחלק הנ"ל, השתמשנו בקצב למידה של $0.01 - 5 \cdot 10^{-5}$.
5. מקדם רגולריזציה (Regularization Coefficient) – כאן השיקולים הוא שמצד אחד אנחנו רוצים למנוע תופעה של Overfitting, אך מצד שני ככל שנגדיל את מקדם הרגולריזציה אמנם נמנע אוברפיטינג אבל גם נפגע בדיוק של המודל. גם כאן, כתלות במודל וכפי שמפורט בקוד בחלק הנ"ל, השתמשנו בגדלים שנעו בטווח של $0.01 - 4 \cdot 10^{-5}$.

התוצאות עבור כל אחד מהמודלים השונים:

1. מודל Logistic Regression:

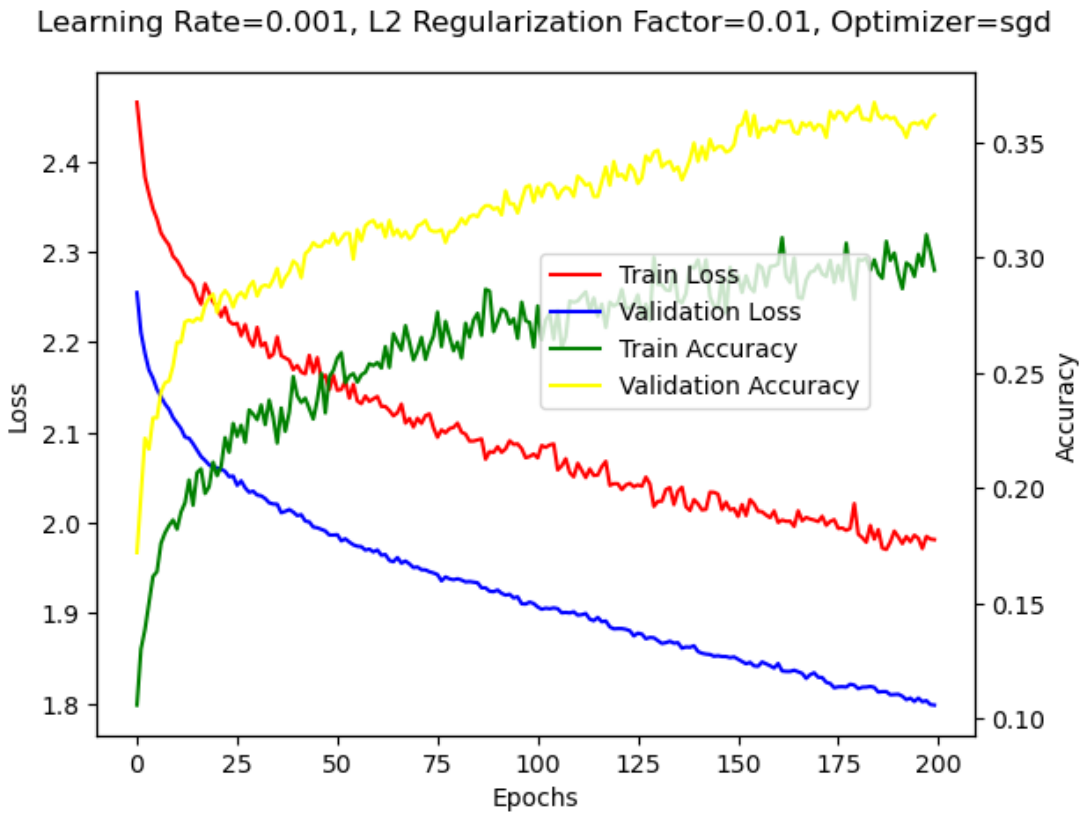
```
Learning Rate: 5e-05, Optimizer: sgd, L2 Regularization: 0.01, Epochs:
150
Train Accuracy: 0.260
Train Loss: 2.126
Validation Accuracy: 0.278
Validation Loss: 2.021
Test Accuracy: 0.281
Test Loss:2.015
```




ניתן לראות שמודל זה לא השיג דיוק גבוה במיוחד, שכן אינו אופטימלי עבור קלאסיפיקציה של תמונות כידוע. זאת ועוד, כאן התבקשנו להשטיח את התמונה ולהפוך אותה לוקטור (כלומר לא התייחסנו לייחודיות של הדאטה). לסיכום, זה אינו מודל כל כך טוב כי הוא לא מתייחס לייחודיות של הדאטה.

2. Fully Connected NN:

```
Learning Rate: 0.001, Optimizer: sgd, L2 Regularization: 0.01, Epochs:
200
Train Accuracy: 0.294
Train Loss: 1.981
Validation Accuracy: 0.362
Validation Loss: 1.798
Test Accuracy: 0.354
Test Loss:1.805
```




בחלק זה של המטלה מימשנו רשת עם 3 שכבות נסתרות כאשר לקחנו גם כאן את התמונות, השטחנו אותן והפכנו אותן לוקטור. בסוף הרשת יש שכבת קלאספיקציה, ובמקומות הנדרשים מימשנו גם נרמול של ה - batch ו – dropout. בניגוד למודל הקודם, ששם הייתה שכבה אחת בלבד, כאן היו 3 שכבות ולכן קיבלנו תוצאות יותר מדויקות ביחס למודל הקודם, כי תהליך הלמידה עבור 3 שכבות יותר טוב מאשר עבור שכבה אחת.

בנוסף, כפי שהתבקשנו במטלה, נציג את החישוב עברו מספר הפרמטרים הנלמדים ברשת זו:

פרמטרים בשכבת הקלט של הרשת:

הקלט הוא בגודל של $64 * 64 * 3$ והפלט בגודל 200, ולכן מספר הפרמטרים בחלק זה הוא:

$$64 * 64 * 3 * 200 + 200 + 200 * 2 = 2458200$$

פרמטרים בשכבה השנייה של הרשת:

הקלט הוא בגודל של 200 והפלט בגודל 100, ולכן מספר הפרמטרים בחלק זה הוא:

$$200 * 100 + 100 + 100 * 2 = 20300$$

פרמטרים בשכבה השלישית של הרשת:

הקלט הוא בגודל של 100 והפלט בגודל 50, ולכן מספר הפרמטרים בחלק זה הוא:

$$100 * 50 + 50 + 50 * 2 = 5150$$

פרמטרים בשכבה הפלט של הרשת:

הקלט הוא בגודל של 50 והפלט בגודל 10, ולכן מספר הפרמטרים בחלק זה הוא:

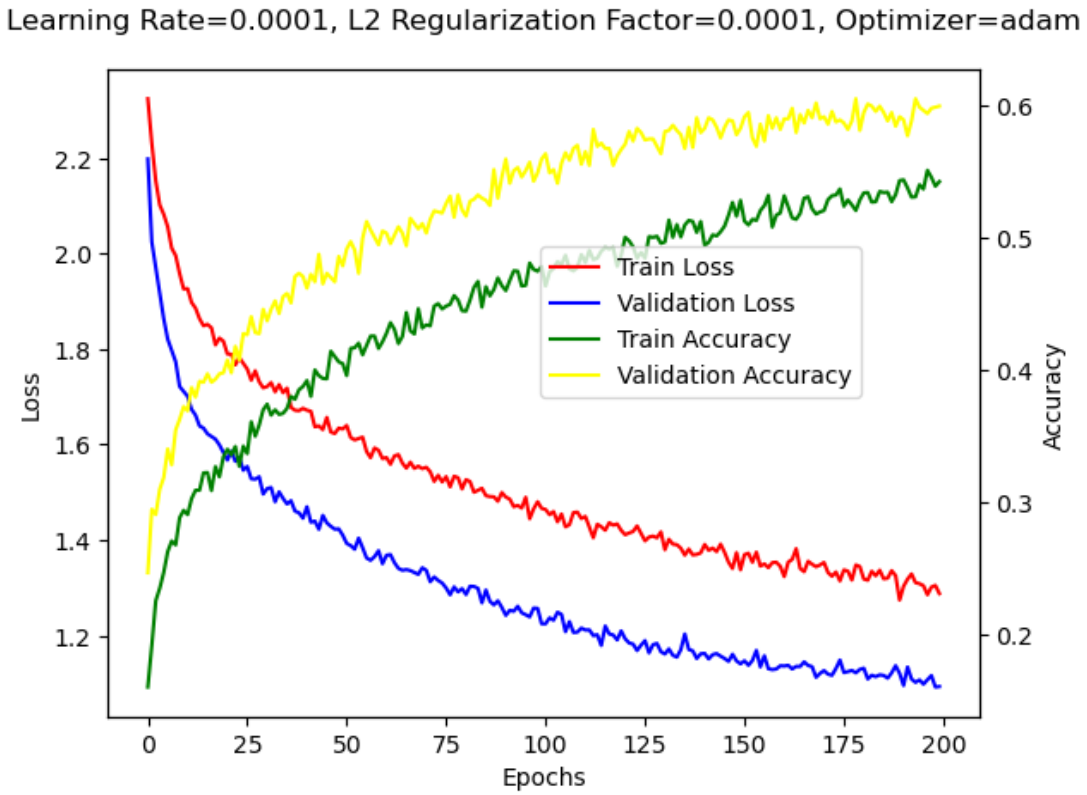
$$50 * 10 + 10 = 510$$

סה"כ פרמטרים ברשת:

$2{,}484{,}160\ Parameters$

3. CNN:

```
Learning Rate: 0.0001, Optimizer: adam, L2 Regularization: 0.0001,
Epochs: 200
Train Accuracy: 0.543
Train Loss: 1.288
Validation Accuracy: 0.600
Validation Loss: 1.093
Test Accuracy: 0.597
Test Loss:1.129
```




כאן בנינו רשת CNN שמכילה 2 שכבות של קונבולציה, לאחריהן שתי שכבות של FC, ובסוף הרשת יש שכבה של קלאספיקציה. כל שכבת קונבלוציה מכילה את פונקציית ה – conv2d, נרמול ה – batch ופונקציית pooling. בנוסף, מימשנו גם dropout עבור שכבות ה - FC, כדרוש. כאן גם קיבלנו תוצאות יותר טובות מהמודל הקודם, ככה"נ בשל העבודה שרשת CNN הינה מודל שמיועד עבור תמונות ובנוסף ישנו אימון ארוך לרשת זו שהיטיב עם הדיוק של הרשת.

בנוסף, כפי שהתבקשנו במטלה, נציג את החישוב עברו מספר הפרמטרים הנלמדים ברשת זו:

עבור כל אחד משכבות הקונבלוציה, החישוב מבוצע כדלהלן:

$$kernel[0] * kernel[1] * n_{input\ filters} * n_{output\ filters} * n_{output\ filters} + n_{output\ filters} + 2 * n_{output\ filters}$$

פרמטרים בשכבת הקלט של הרשת:

הקלט הוא בגודל של (64,64,3) והקרנל בגודל של (3,3), הפלט הוא בגודל של:

*((64-3+1)//2, (64-3+1)//2, 20) = (31, 31, 20)*

סה"כ מספר הפרמטרים בשכבה זו:

*3*3*3*20+20+20*2 = 600*

פרמטרים בשכבה השנייה של הרשת:

הקלט הוא בגודל של (3131,3) והקרנל בגודל של (3,3), הפלט הוא בגודל של:

*((31-3+1)//2, (6313+1)//2, 63) = (14, 14, 64)*

סה"כ מספר הפרמטרים בשכבה זו:

*3*3*20*64+64+64*2 = 11712*

פרמטרים בשכבה השלישית של הרשת (שכבה ראשונה של FC):

הקלט הוא בגודל של $14 * 14 * 64 = 12544$ והפלט בגודל 100, ולכן מספר הפרמטרים בחלק זה הוא:

$$12544 * 100 + 100 = 1254500$$

פרמטרים בשכבה הרביעית של הרשת (השנייה של ה - FC):

הקלט הוא בגודל של 100והפלט בגודל 50, ולכן מספר הפרמטרים בחלק זה הוא:

$$100 * 50 + 50 = 5050$$

פרמטרים בשכבה הפלט של הרשת:

הקלט הוא בגודל של 50 והפלט בגודל 10, ולכן מספר הפרמטרים בחלק זה הוא:

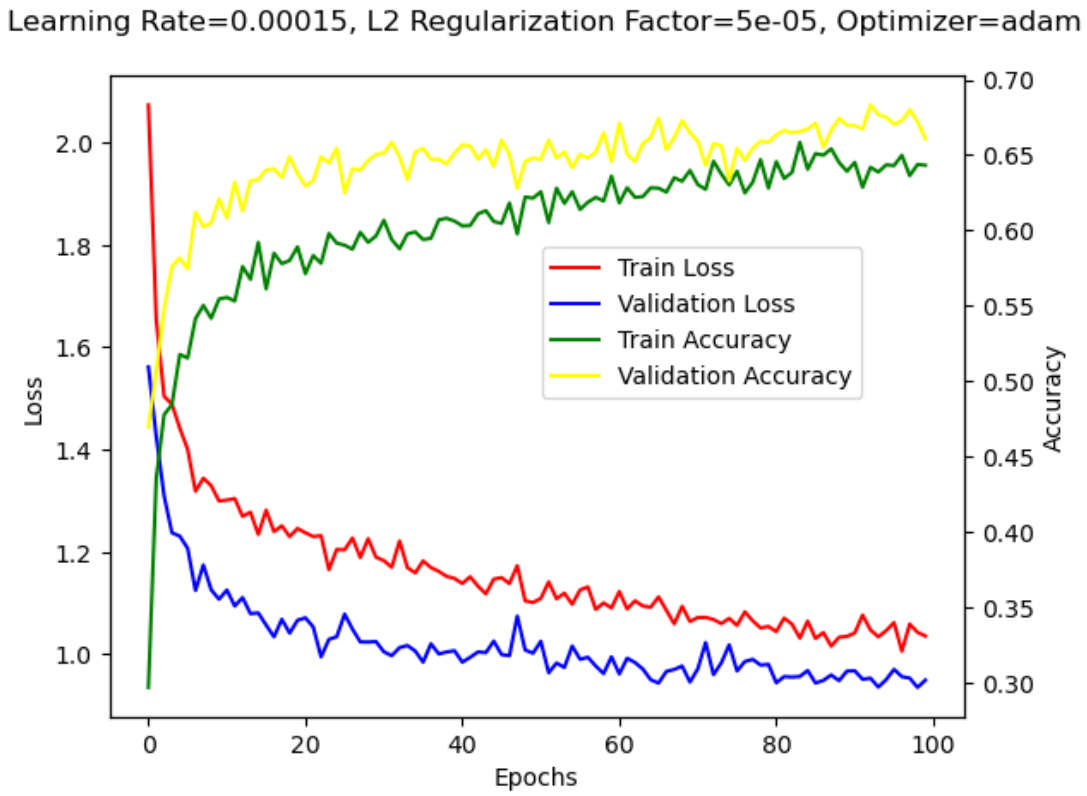
$$50 * 10 + 10 = 510$$

סה"כ פרמטרים ברשת:

$$1{,}272{,}372\ Parameters$$

4. Fixed MobileNetV2:

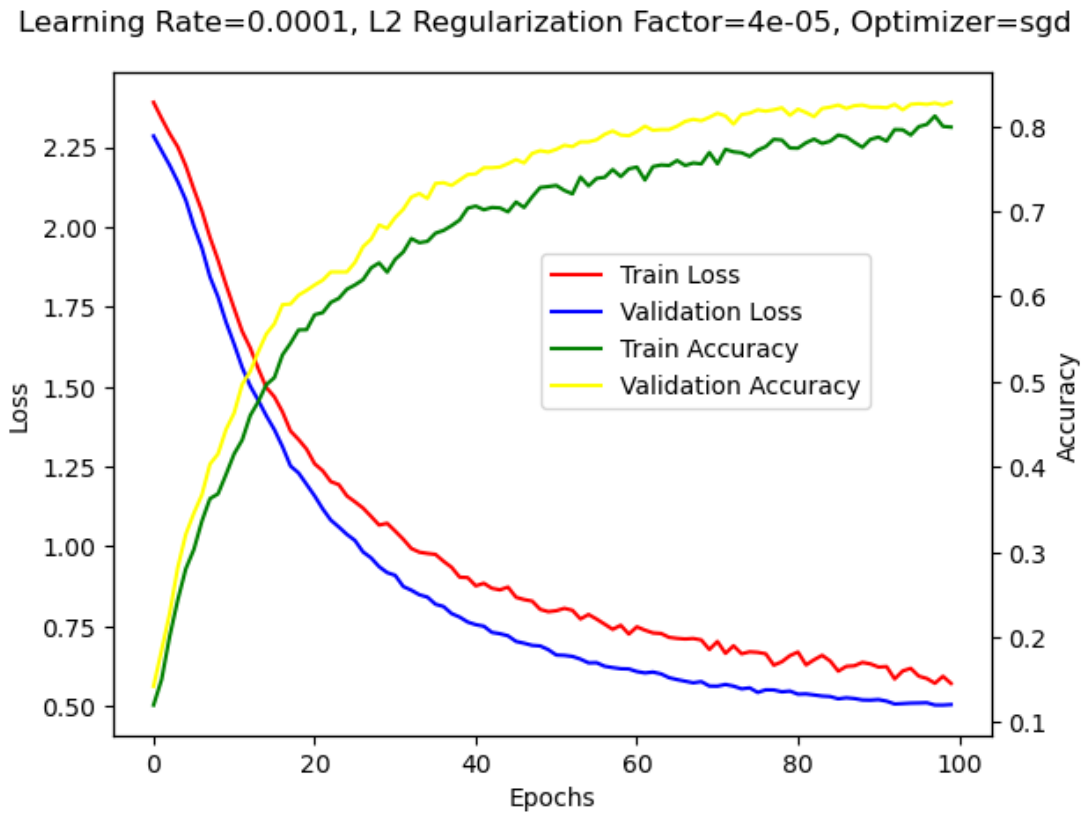
```
Learning Rate: 0.00015, Optimizer: adam, L2 Regularization: 5e-05,
Epochs: 100
Train Accuracy: 0.643
Train Loss: 1.035
Validation Accuracy: 0.661
Validation Loss: 0.949
Test Accuracy: 0.643
Test Loss:1.013
```




בחלק זה של המטלה השתמשנו ברשת מוכנה ומאומנת מראש, כאשר הוספנו לה בסופה שתי שכבות של FC ושכבת קלאספיקציה בסוף הרשת. ניתן לראות שכיוון שזו רשת שמאומנת מראש קיבלנו תוצאות טובות יותר מהמודלים הקודמים.

5. Learned MobileNetV2:

```
Learning Rate: 0.0001, Optimizer: sgd, L2 Regularization: 4e-05,
Epochs: 100
Train Accuracy: 0.799
Train Loss: 0.571
Validation Accuracy: 0.828
Validation Loss: 0.505
Test Accuracy: 0.809
Test Loss:0.560
```




בחלק זה של המטלה השתמשנו באותה הרשת מחלק הקודם, רק שכאן אימנו את הראש ועדכנו את הפרמטרים שלה בעצמנו, ואכן קיבלנו תוצאות יותר טובות, כצפוי.

בסופו של דבר, ניתן לראות שהרשת האחרונה נתנה את התוצאות הטובות ביותר.

<u>כיצד להריץ את הקוד שלנו</u>

הקוד שלנו הוא בקובץ ipynb כלומר צריך מחברת Jupiter כדי להריץ אותו.

בנוסף הרצנו את הקוד שלנו עם פייתון בגרסה 3.8.18.

בנוסף, השתמשנו בספריות הבאות (צריך להוריד אותן כדי להריץ את הקוד):

Numpy, torch, torchvision, matplotlib, sklearn