מבוא לראייה ממוחשבת(22928)

פרויקט סיום:

Font Recognition

נדב מידן

ת.ז. 200990240

מנהלות הגשה:

בתיקיית ה-Zip קיימים הקבצים הבאים(בנוסף כמובן לקובץ PDF זה):

- README ובו הוראות הרצה מדויקות של קוד האימון והחיזור, כולל לינקים למודל ולקבצים אחרים.
- קובץ Python ובו מומשה התוכנית, מסודרת להפעלה פשוטה ע"י Run.

תיאור תהליך החשיבה:

פיתרון CV קלאסי

בשלב הראשון, תכננתי לתכנן איזשהו Base Line קלאסי, שממנו אצא לכיוון פיתרון הבעיה.
כלומר, יש לנו תמונות, עם טקסט מסונטז במניפולציות שונות(הטיות, כתב מראה, בולטות משתנה וכו') אך המיקומים נתונים לנו והטקסט, ולכן זו הופכת לבעיית קלאסיפיקציה שניתן לפתור בעזרת חילוץ פיצ'רים ל- Features Map והכנסה למסווג קלאסי כלשהו.

לכן, תכננתי ליצור Features Map ע"י הרצה של Sift descriptor על כל התווים האפשריים(אותיות אנגלית- גדולות וקטנות, ספרות ותווים נוספים) - על גזרי תמונות של אותיות מתוך הדאטה(אך חששתי שרעש ולכלוך יפריעו לי לחלץ כמות מספקת של פיצ'רים) או על דאטה ב"תנאי מעבדה" שאכין(לדוגמא לגזור טקסט בWord בכל אחד מהפונטים). בעזרת האלגוריתם, שנחשב כלי מאוד חזק, נחלץ נק' מפתח בתמונה ונחשב וקטור תיאורי- וקטור פיצ'רים.
ע"H חילוץ וקטור כזה עבור כל תו מכל פונט, אוכל להרכיב מפת פיצ'רים שמקיפה את ה"שפה" שבה חי הטקסט המסונטז שאת הפונט שלו אני מבקש לזהות.
בשלב השני, בהינתן מפת הפיצ'רים הזו, אוכל ע"י Knn classifier, עבור דגימת kים אי זוגיים עולים שאינם כפולה של 3(מספר הפונטים אליהם אנו מסווגים), להבין מהו ערך הK האופטימלי לסווג איתו את הטקסט לפונטים השונים.

כלומר, **האלגוריתם שתכננתי כלל:**

1- <u>עיבוד מקדים על הדאטה:</u> לבצע Parsing לתמונות המקוריות כך שייוצר לי דאטה סט של תמונות- תווים חתוכות(בהתחשב בהפעלת מניפולציות על כולן ליצירת אחידות: Rotation, Horizontal Flip, Normalize, Resize, GrayScale,, ניקוי וכו').
את הדאטה, מעובד וחתוך, ניתן לראות בלינק הבא:
https://drive.google.com/drive/folders/1sfIs-a-UbPBDYh6uh39xc7tW878oNToF?usp=sharing
2- עבור כל תמונה מקורית- נעבוד על רשימה של האותיות המופיעות בה.
3- עבור כל תמונה של אות- נריץ Sift ונסווג.
4- כך, ניצור Features Map של כל התווים האפשריים בכל שלושת הפונטים.
5- עבור כל מילה/ מחרוזת בתמונה, והכל התמונות, נריץ את אלג' Knn, ונבנה פרדיקציה לסיווג עבור כל תו במחרוזת.
6- לאחר מכן, נוכל להשתמש במידע על המילה(wordBB מסמן גבולות של מחרוזת בה כל התווים שייכים לאותו הפונט) וניישם 'מודל הצבעה' שעל פיו נכריע את פונט המילה.
7- כמובן שנריץ זאת עבור k עולים כאמור, וארכז את הנתונים לדו"ח, וכך נוכל להכריע עבור איזה ערך של k מתקבל אחוז הדיוק הגבוה ביותר.

אבל.. צללתי לתוך הדאטה המעובד, והבנתי שהוא מכיל גם תוים קשים ביותר לזיהוי, לדוגמא:

ולכן ליצור מפת פיצ'רים מתוך הדאטה ואז לסווג תספק לדעתי ביצועים בינוניים(70-80%), ואילו יצירת הפיצ'רים מדאטה נקי מהWord תהיה רחוקה מדי מהתווים איתם תיפגש בדאטה סט.

הבנתי שאו שמודל עבודה כזה יספק תוצאות בינוניות נמוכות- או שידרוש עיבוד מקדים משמעותי מאוד על הדאטה, ובחרתי לנסות ולנצל את האפשרות להתנסות בעולם רשתות הנוירונים והדיפ לרנינג, שבו לא התנסיתי עד היום, ולבחון פיתרון המשלב שיטה קלאסית ולמידת מכונה.

פיתרון המשלב CV קלאסי עם רשתות נוירונים

אני סטודנט תואר ראשון, ולא התנסיתי בעבר בעבודה עם מודלים של רשתות נוירונים- מלבד בקורס חישוביות ביולוגית, אך שם הסקירה היא יותר היסטורית ותיאורטית ופחות יישומית.
לכן, לאחר שכבר ביצעתי עבודה מקדימה על הדאטה, חיפשתי דרך להכניס את הדאטה למודל עובד, כדי להתרשם מאיך תהליך האימון נראה ועובד. מצאתי מימוש מעניין למודל VGG16 בKeras שמאוד נוח לעבוד איתו למי שלא מבין, ובטח גם לכאלה שכן. בכל אופן, לצעדים ראשונים זה היה מצוין, והמודל הכבד והענק הזה השיג תוצאות די טובות באימון- באיזור ה90%.

```
Train Epoch: 20 [0/12238 (0%)]  Loss: 0.096836
Train Epoch: 20 [1920/12238 (16%)]      Loss: 0.058279
Train Epoch: 20 [3840/12238 (31%)]      Loss: 0.042356
Train Epoch: 20 [5760/12238 (47%)]      Loss: 0.043608
Train Epoch: 20 [7680/12238 (62%)]      Loss: 0.214342
Train Epoch: 20 [9600/12238 (78%)]      Loss: 0.061622
Train Epoch: 20 [11520/12238 (94%)]     Loss: 0.039173

Test set: Avg. loss: 0.2694, Accuracy: 7408/8198 (90%)
```

לאחר, מכן, קראתי על מודל שנקרא Feature Extraction, הדומה מעט לTransfer Learning, בכך שלוקחים מודל מאומן, "מפשירים" את הרשת האחרונה ומתאימים אותה לצורך שלנו.
16VGG שלקחתי מאומן כבר על הדאטה סט imageNet המכיל מעל מליון תמונות, של 1000 classים.
כל תמונה של תו, נכנסת לVGG המאומן, ויוצא עבורה וקטור פיצ'רים בגודל 4096 שמייצג אותה(וזו הסיבה שנפטרים מהשכבה האחרונה של הclassifier של המודל).
לאחר שמתקבל וקטור כזה עבור כל תמונה, שוב בדומה לרעיון הראשון שלי, נוכל להרכיב מעין מפת פיצ'רים ולהריץ עליה Knn ולבחור את הערך האופטימלי של k עבורו מתקבל אחוז הדיוק הגבוה ביותר.
ניתן להתרשם מהמודל בלינק הבא:
https://colab.research.google.com/drive/1dv1ArxNNSPX2blqExTM2aB9HoL42gCRc?usp=sharing

עם זאת, הרצה של מודל כה כבד ולאחר מכן גם Knn הייתה נורא כבדה עבור הקולאב והריצה נתקעה לאחר k=9, אך זה הספיק כדי להבין את הכיוון.

```
#Use K nearest neighbor to classify the test image from their features (K=[1,2,...10])
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
k_range = range(1, 21)
scores = []
for k in k_range:
 knn = KNeighborsClassifier(n_neighbors=k)
 knn.fit(vgg16_feature_list_np, charListLablesTrain)
 y_pred = knn.predict(vgg16_feature_test_list_np)
 print('for {} neighbors the score is {}'.format(k,metrics.accuracy_score(charListLablesVal, y_pred)))
 scores.append(metrics.accuracy_score(charListLablesVal, y_pred))
```

```
for 1 neighbors the score is 0.8558184923151988
for 2 neighbors the score is 0.8273969260795316
for 3 neighbors the score is 0.8541107587216394
for 4 neighbors the score is 0.8392290802634789
for 5 neighbors the score is 0.8448402049280312
for 6 neighbors the score is 0.83996096608929
for 7 neighbors the score is 0.8471578433764333
for 8 neighbors the score is 0.843620395218346
for 9 neighbors the score is 0.8447182239570628
```

תוצאה מיטבית התקבלה עבור k=3, אך כלל התוצאות היו יחסית אחידות בטווח ה82-85%. כלומר, פחות טובות מאימון של VGG לבדו, וכבד מדי למשימה.

לכן, שוב ניצלתי הזדמנות, וניגשתי ללמוד איך לבנות רשת בסביבת Pytorch ולהתנסות בעצמי בעיצוב של רשת, חישובי המעברים בין הרשתות(מס' הפרמטרים) והרבה ניסוי וטעיה.

פיתרון הכולל בנייה ואימון של רשת נוירונים מותאמת

לינק למודל-

https://drive.google.com/file/d/17OZikIZ78iKKNMGXHuqmPUzqT7rgcHi8/view?usp=sharing

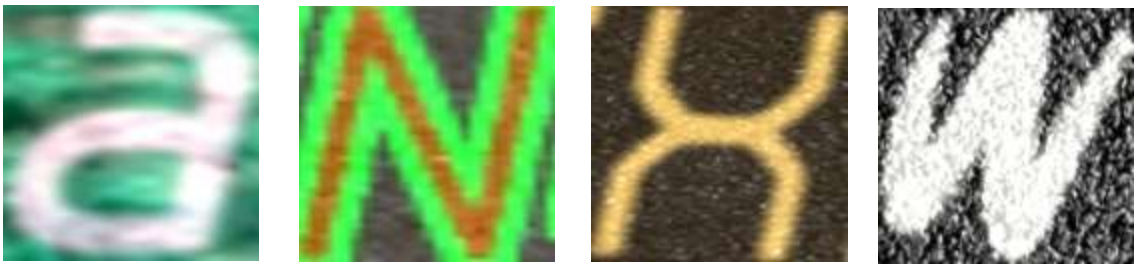
כאמור, בכל אחד מהרעיונות ביצעתי קודם כל את השלב העיבוד המקדים על הדאטה.

במסגרתו, : ביצעתי Parsing לתמונות המקוריות כך שייוצר לי דאטה סט של תמונות- תווים חתוכות(בהתחשב בהפעלת מניפולציות על כולן ליצירת אחידות: Rotation, Resize, GrayScale ,Horizontal Flip, Normalize, ניקוי וכו') וכמובן חיתוך ע"י טרנספורמציה פרספקטיבית.

את הדאטה, מעובד וחתוך, ניתן לראות בלינק הבא:

https://drive.google.com/drive/folders/1sfIs-a-UbPBDYh6uh39xc7tW878oNToF?usp=sharing

אציג מספר דוגמאות:



אז אחרי קריאה של מספר טוטוריאלס ומאמרים, הבנתי שבעיית הסיווג שלנו די דומה לסיווג על הדאטה סט MNist, רק שבמקום לסווג ל10 קלאסים(ספרות 0-9), יש לסווג ל-3 בלבד, כמספר הפונטים.

לאחרונה אני לומד איזשהו קורס מבוא לDeep Learning והתחלתי באמצעותו לכתוב את הרשת הפשוטה שלי.

בהתחלה, עיצבתי רשת פשוטה בעלת 3 שכבות ועוד 3 שכבות Fully Connected עם פונקציית אקטיביה Relu כשמבצעים batchNorm-Relu-dropOut ואת הכל עוטף max pooling כך בשלוש השכבות הראשונות, באותו אופן שניתן לראות בהרבה מימושים של רשתות לסיווג על MNist.

זה לא כ"כ צלח, והרשת שלי לא למדה, ונתקעה על 60% מה Epoch הראשון ועד העשירי, עם אותה תוצאה מספרית של סיווגים נכונים. כמובן שהנסיון הראשון כלל hyper Parameters נפוצים(ממקדם למידה 0. 1, מומנטום 0.9, ואופטימייזר SGD).

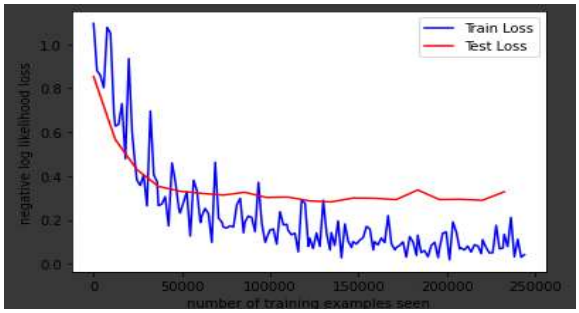
חשבתי שהבעיה היא שאני לא מכניס קלט מגוון מספיק מגוון וניסיתי להכניס אותו יחד עם composition של טרנספורמציות- horizontal flip, rotation, Gausian Noise, אך הרשת עדיין קפאה במקום ולא למדה. מכך הסקתי שהדאטה מספיק מגוון במניפולציות שלו על האותיות כך

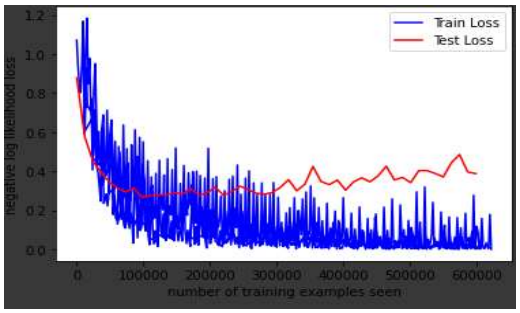
```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 512, kernel_size=3 ,padding = (1,1))
        self.conv2 = nn.Conv2d(512, 256, kernel_size=3, padding = (1,1))
        self.conv3 = nn.Conv2d(256, 128, kernel_size=3, padding = (1,1))
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(32768, 64)
        self.fc2 = nn.Linear(64, 64)
        self.fc3 = nn.Linear(64, 3)
        self.batchNorm1 = nn.BatchNorm2d(512)
        self.batchNorm2 = nn.BatchNorm2d(256)
        self.batchNorm3 = nn.BatchNorm2d(128)
        # self.noise = GaussianNoise()
    def forward(self, x):

        x = F.dropout(F.relu(self.batchNorm1(self.conv1(x))),0.25)
        x = F.max_pool2d(x , 2)
        x = F.dropout(F.relu(self.batchNorm2(self.conv2(x))),0.15)
        x = F.max_pool2d(x , 2)
        x = F.dropout(F.relu(self.batchNorm3(self.conv3(x))),0.3)
        x = F.max_pool2d(x , 2)
        x = x.view(-1, 32768)
        x = F.dropout(F.relu(self.fc1(x)),0.4)
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return F.log_softmax(x)
```

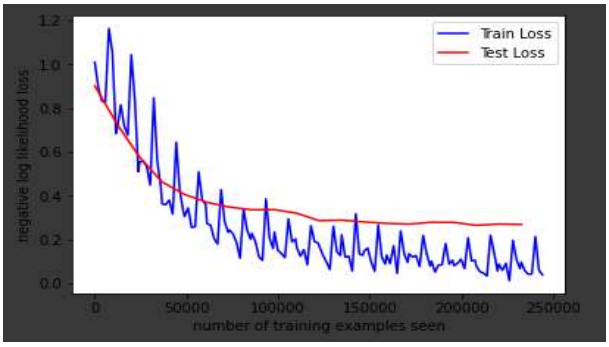
שהטרנפרמציות עליהן לא נצרכות באופן מלאכותי ולא משפרות ביצועים.

אז ניסיתי לשנות- החלפתי בין ה max pooling לdropOut, והרשת התחילה ללמוד- 63% בepoch הראשון ועולה ועולה. ראיתי שערך ה loss מאוד הפכפך אז שיחקתי מעט עם ערכי dropout ועם ערך המומנטום, כך שלא ייוצר 'מסגר' בירידה לאורך הגרדיאנט, אלא התכנסות כמה שיותר רציפה וחלקה, כפי שאפשר להתרשם מהתרשימים:

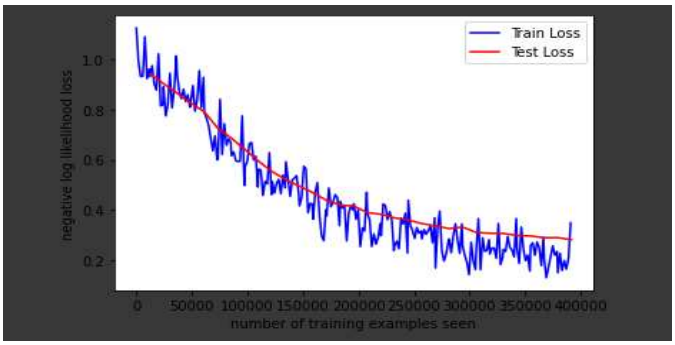




הבנתי שהמודל שלי אולי לומד מהר מדי, ועלול להיווצר over fitting ולכן המשכתי לנסות ולעדן את מקדם הלמידה, את ערכי הdropOuts המשתנים לאורך הרשת, ואת המומנטום, ולאט לאט הצלחתי מעט לאזן אותו, כולל להוריד את מס' האפוקים.



ואז



בסה"כ, את הניסיונות של ריכזתי במסמך הבא, והתרשמתי שגודל הבאטצ' משמעותי לפוטנציאל הלמידה, ולמידה איטית יותר תייצר מודל יציב יותר ומאומן, גרפים חלקים יותר והלכתי בכיוון זה.

ניסיתי גם להחליף אופטימייזר ל- Adam אבל הpreformance רק נחלשו.

ניתוח ביצועי מודל שבניתי ע"פ היפר פרמטרים

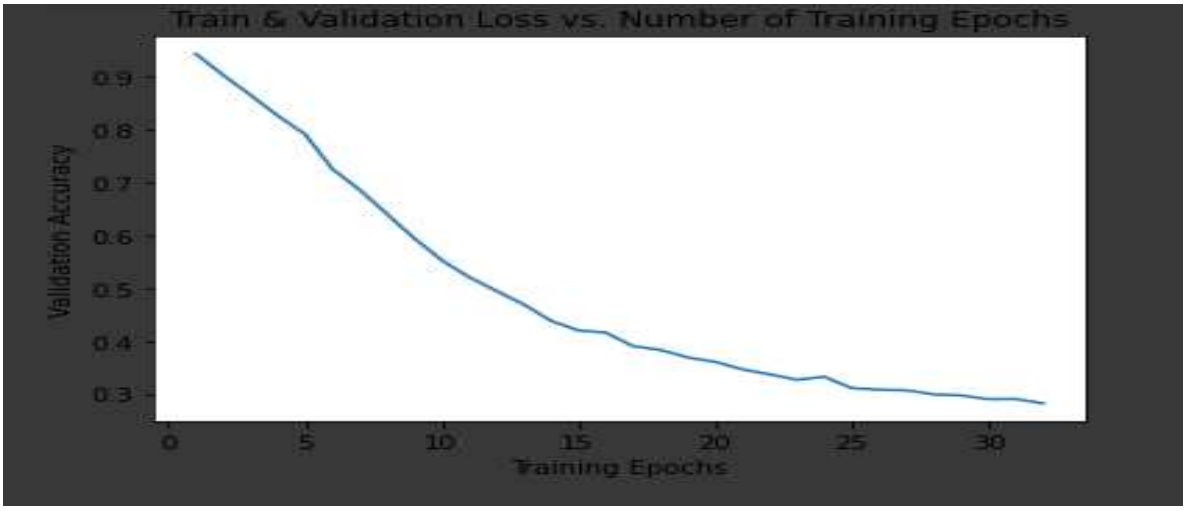
ניסיון	Learning Rate	Momentum	n_epochs	F.dropout	batch_size	נתונים אחרים	Accuracy	הערות
1	0.01	0.9	50	0.15-0.5	32	dropOut מעל max pooling	60%	
2	0.01	0.9	50	0.15-0.5	32	dropOut מעל max pooling	63%	
3	0.001	0.9	50	0.15-0.5	32	dropOut מעל max pooling	90%	
4	0.001	0.7	50	0.15-0.5	32	dropOut מעל max pooling	91%	
5	0.001	0.8	20	0.15-0.5	32	dropOut מעל max pooling	91%	
6	0.001	0.6	20	0.15-0.5	32	dropOut מעל max pooling	90-91%	
7	0.001	0.9	40	כולם ב-0.5	32	dropOut מעל max pooling	85%,	
8	0.001	0.9	20	0.15-0.5	64	dropOut מעל max pooling	91%,	
9	0.001	0.9	20	0.15-0.6	64	dropOut מעל max pooling	81%	
10	0.0008	0.7	20	0.15-0.6	64	dropOut מעל max pooling	90%	
11	0.001	0.9	32	0.15-0.5	32	והכנסת dropOut מעל max pooling טתספורמס(נרמול וטימור)	94%+	עולה מהר, לוס יחסית יציב
12	0.001	0.9	32	0.15-0.5	32	והכנסת dropOut מעל max pooling טתספורמס(נרמול וטימור ורעש)	94%	עולה מהר, לוס טיפה תנודתי
13	0.001	0.9	32	0.15-0.5	32	והכנסת dropOut מעל max pooling טתספורמס(נרמול וטימור ורעש) וללא מנויאל סיד	93%	עולה מהר, לוס טיפה יותר תנודתי
14	0.001	0.9	32	0.15-0.5	32	והכנסת dropOut מעל max pooling טתספורמס(נרמול וטימור ורעש והוריזונטל פלים) ועם מנויאל סיד	91%	עולה פחות מהר, לוס ריצרצ החל מאפוק 10
15	0.001	0.9	32	0.15-0.5	32	והכנסת dropOut מעל max pooling טתספורמס(נרמול וטימור ורעש ורוטציית 60 מעלות) ועם מנויאל סיד	85%	עולה לאט, לוס הפכפך
16	0.001	0.9	32	0.15-0.5	32	והכנסת dropOut מעל max pooling טתספורמס(נרמול וטימור והוריזונטל פלים) ועם מנויאל		עולה מהר מאוד
17	0.001	0.9	64	0.15-0.5	32	והכנסת dropOut מעל max pooling טתספורמס(נרמול וטימור ורעש)	חלש ולוס מאוד לא יציב	Adam
18	0.000008	0.9	64	0.15-0.5	110	והכנסת dropOut מעל max pooling טתספורמס(נרמול וטימור)		
19	0.00005	0.9	64	0.15-0.5	96	והכנסת dropOut מעל max pooling טתספורמס(נרמול וטימור)		
20	0.0001	0.7	64	0.15-0.5	96	והכנסת dropOut מעל max pooling טתספורמס(נרמול וטימור)		נכשל, OF
21	0.0000095	0.9	32	0.15-0.5	120	והכנסת dropOut מעל max pooling טתספורמס(נרמול וטימור)	90%	
22	0.0001	0.8	32	0.15-0.5	32	והכנסת dropOut מעל max pooling טתספורמס(נרמול וטימור)	93%	
23	0.00095	0.9	32	0.15-0.5	64	והכנסת dropOut מעל max pooling טתספורמס(נרמול וטימור)	94%	
24	0.0001	0.9	32	0.15-0.5	64	והכנסת dropOut מעל max pooling טתספורמס(נרמול וטימור)	94%	
25	0.00095	0.9	32	0.15-0.5	64	והכנסת dropOut מעל max pooling טתספורמס(נרמול וטימור)	94%	loss 0.19 7726

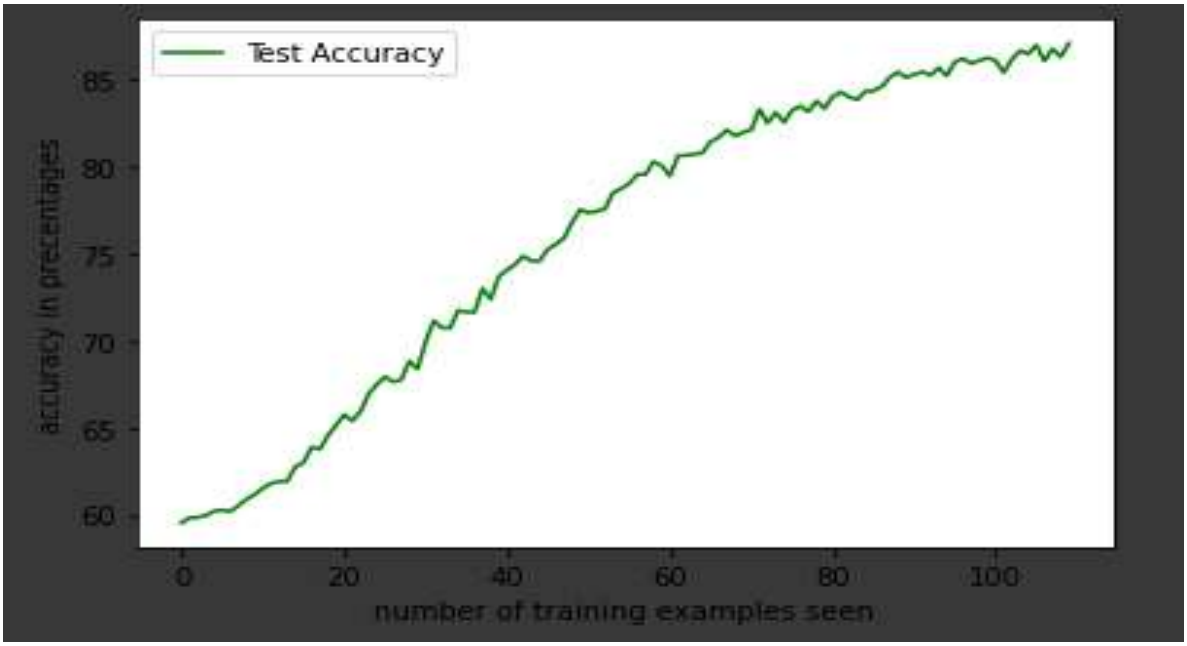
המודל עם הפרמטרים המנצחים, המסומן בירוק בטבלה, השיג התכנסות די חלקה ואיכותית, ובאימון לא ארוך מדי, 32 אפוקים:

```
Train Epoch: 32 [0/12237 (0%)]  Loss: 0.196752
Train Epoch: 32 [960/12237 (8%)]        Loss: 0.325932
Train Epoch: 32 [1920/12237 (16%)]      Loss: 0.018214
Train Epoch: 32 [2880/12237 (23%)]      Loss: 0.045675
Train Epoch: 32 [3840/12237 (31%)]      Loss: 0.008933
Train Epoch: 32 [4800/12237 (39%)]      Loss: 0.010871
Train Epoch: 32 [5760/12237 (47%)]      Loss: 0.027397
Train Epoch: 32 [6720/12237 (55%)]      Loss: 0.058207
Train Epoch: 32 [7680/12237 (63%)]      Loss: 0.194628
Train Epoch: 32 [8640/12237 (70%)]      Loss: 0.013162
Train Epoch: 32 [9600/12237 (78%)]      Loss: 0.039581
Train Epoch: 32 [10560/12237 (86%)]     Loss: 0.092798
Train Epoch: 32 [11520/12237 (94%)]     Loss: 0.105992

Test set: Avg. loss: 0.2050, Accuracy: 7718/8197 (94%)
```

(בשלב מסוים קצת השתבשו לי הכותרות והמספרים על הצירים, לכן נראה כאילו אין תאימות)





המודל הזה השיג performance של 94% באימון על הסט ולידציה, כשאני עבור כל סט פרדיקציות שמיוצר לכל אות- בוחר את המקסימלית ללא תלות בתוצאות עבור האותיות האחרות במילה.

בהינתן שהדאטה כ"כ לא מאוזן (מתחלק בערך 50%, 30%,20% בין הפונטים) זו תוצאה טובה.

לאחר, מכן, כאשר אני משתמש במודל המאומן לסיווג סט הבדיקה.

הרצתי תחילה את הבדיקה על סט הולידציה המכיל לייבלים, וזיהיתי שהרבה מן הטעויות הינן אות או שתיים במילה, וללא ספק ניתן לבצע מעין ניתוח אחר לתוצאות, בהינתן אחידות הפונט בכל תמונה.

לכן, בשלב הטסט, השתמשתי ב'מודל הצבעה' כך שאני מחשב mostFrequent מכל הפרדיקציות לאותיות במילה, ואם ישנו רוב ברור- אני עושה מעין ולידציה ומתקן את האות ה"סוררת" להיות בפונט כמו כל חברותיה במילה. ולכן, כבשבדקתי את הטסט על סט הולידציה, מודל ההצבעה העלה את אחוזי הדיוק ל98%(עשיתי diff בין שני הפלטים של הcsv).

בנוסף, בחרתי לבצע את המעבר על הדאטה בשלב הבדיקה באופן שונה.

אם באימון המודל, אני מעבד את כל הדאטה ומרכז אותו, ומכניס את כולו לרשת בבאצ'ים של 64, בבדיקה(השימוש במודל)- אני מבצע את כל התהליך על מילה אחת בתמונה בכל פעם. ז"א, בזמן שהמילה 'have' בתמונה הראשונה, כבר עובדה, נכנסה לרשת, קיבלה פרדיקציה עבור כל אות עברה "ולידציה" ואף נכתב כבר לcsv- המילה 'the' אשר איתה באותה התמונה, לא נחתכה אפילו מהתמונה המקורית. כלומר, אני מכניס את התמונות לרשת בבאצ'ים של 'מילה' אחת.