**פרויקט מסכם**

מבוא לראייה ממוחשבת

22928

****



**מגיש: אביעד טוויג**

**ת.ז: 319090171**

**תאריך הגשה: 20.1.2021**

**מנחה: ד"ר אמיר אגוזי**

**מבוא**

לפני שאתחיל לדבר על הפרוייקט וכיצד יישמתי את המודל אני אספר קודם על עצמי.

התחום של בינה מלאכותית משך אותי מאז שהייתי ילד (בערך גיל 5), לצערי עד השנה הנוכחית לא באמת התעסקתי בתחום. מדי פעם הייתי רוצה הרצאות ב-TED וב-Youtube, אבל לא באמת יצא לי ללמוד את התחום ולהתעסק בו.

אני כרגע במהלך התואר הראשון אותו התחלתי בגיל 19, וידעתי מהרגע הראשון שאני עומד ללכת לכיוון הבינה המלאכותית. מהסמסטר הראשון חיפשתי קורסים בתואר השני שמתקשרים לתחום הבינה המלאכותית. שני הקורסים שהכי עניינו אותי היו מבוא לעיבוד שפה טבעית ומבוא לראייה ממוחשבת. וכמובן שלבסוף לקחתי את הקורס מבוא לראייה ממוחשבת, כל האנשים מסביבי אמרו לי "מה אתה לוקח קורס של תואר שני במהלך התואר הראשון?", התשובה שלי הייתה ברורה מאוד מבחינתי. אני לוקח קורסים בנושאים שמעניינים אותי.

שהתחלתי את הקורס הייתי מאוד בשוק (ניתן לראות את זה במטלה הראשונה בה קיבלתי 57), לא ידעתי פייתון לפני הקורס, ובעצם לא היה לי שום ידע מקדים שמתקשר לראייה ממוחשבת חוץ מזה שזה עניין אותי. במטלה השנייה כבר יותר הבנתי את החומר וקיבלתי ציון בהתאם 98. (חייב לציין שקצת השתגעתי ובסמסטר הנוכחי אני לוקח 5 קורסים במקביל).

אבל עדיין לא הצלחתי להבין עד הסוף מה זה אומר deep learning, neural networks וכו'. והחלטתי לעשות קורס נוסף במקביל במסגרת udemy כדי להבין את הנושאים הנ"ל.

היום אני מרגיש שאני בעל יכולת להתחיל באמת לעבוד בתחום שמתקשר ב-deep learning, מכיוון שהקורס הזה לימד אותי הרבה, על יכולות הלמידה שלי האישיות, על בינה מלאכותית בכללי ועל התחום של ראייה ממוחשבת בפרט.

**כל קבצי ה-h5, py וה-csv של הפרוייקט שמורים בתוך ה-google drive בכתובת:**

[**https://drive.google.com/drive/folders/1\_TfToihaOpBerTbu78urajNSUpFiOZQW?usp=sharing**](https://drive.google.com/drive/folders/1_TfToihaOpBerTbu78urajNSUpFiOZQW?usp=sharing)

**בתיקייה: python files and relevant h5**

קיימים כל קבצי הקוד בפייתון וכל קבצי h5 שהשתמשתי במהלך התוכנית.

**בתיקייה: results on test set**

קיימים כל קבצי ה-csv שיצרתי כתוצאה מהחיזוי על סט הבדיקה (כפי שרשום בע"מ האחרון של תיק הפרוייקט).

**בתיקייה: results on train set**

קיימים קבצי ה-csv על סט האימון החדש (כפי שציינתי בע"מ 8 בתיק הפרוייקט).

**תהליך הפרוייקט**

**מטרה –**

המטרה של הפרוייקט היא לסווג כל תו בתמונות לפונט מבין שלושת הפונטים:

1. Skylark
2. Ubuntu Mono
3. Sweet Puppy

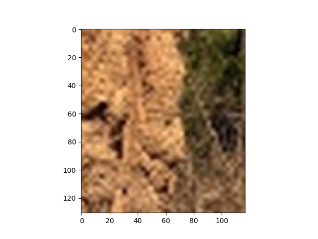
**בניית סט האימון –**

מכיוון שהמטרה הסופית היא לזהות את הפונט של כל תו בכל התמונות, הגעתי למסקנה שאין לי שום צורך בלשמור את התמונה המלאה בסט שלי ומספיק לי לכל תמונה לשמור רק את תמונה שמייצגת את התו עצמו תוך כדי שימוש ב-bounding box של כל תו.

והתחלתי בתהליך של חיתוך התמונות לפי ה- bounding box של כל תו בתמונה.

חיתוך התמונה לפי התווים -

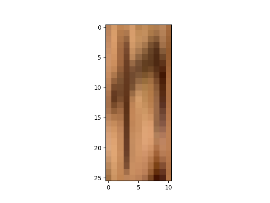
התהליך של חיתוך התמונה, היה לי בהתחלה מאוד מסובך.

ראשית ניסיתי להשתמש בקודקודים של הפינה השמאלית העליונה ביחד עם הרוחב והאורך כדי להגיע לחיתוך המתאים אבל זה נתן לי תוצאות מאוד שגויות ולא ברורות.

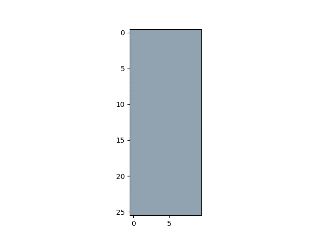
למשל, כאן היה אמור להופיע האות t מהתמונה הראשונה.

ואז הבנתי שעליי להיעזר באינטרנט כדי להגיע לתוצאה הנכונה. השתמשתי במספר מקורות אבל עדיין שום מקור לא נתן לי את החיתוך אותו רציתי.

לבסוף מצאתי באתר pyimagesearch, [בקישור הזה](https://www.pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/), יישמתי את דרך זו בקוד ואכן ראיתי שבהתחלה הקוד עזר לי לחתוך את התמונות בצורה נכונה. אבל אז שמתי לב שבתו הרביעי של התמונה הראשונה הוא התחיל להציג מקומות בתמונה שבכלל אין שם תווים.



למשל כאן ניתן לראות שמופיעה האות t (קצת הפוכה אבל ניתן להבין שזה אכן זה).



לעומת זאת, אין כאן בכלל את האות h שהיא התו הרביעי בתמונה הראשונה.

לבסוף הצלחתי להגיע לתוצאה הנדרשת בה כל תו מופיע במלבן נפרד, בעזרת הפעולות getPerspectiveTransform ו-warpPerspective של opencv. כל תמונה שמרתי בגודל של 32 על 32 פיקסלים, שמרתי את כל התמונות ברשימה עדיין בצבעם המקורי.

באופן מקביל על כל תו שהוספתי לסט האימון שלי הוספתי לו תגית מ-0 עד 2 שמסמן את הפונט שבו הוא נמצא ( 0 = skylark, 1 = ubuntu mono, 2 = sweet puppy).

את סט האימון

**בניית המודל והתאמה עם סט הוולדיזציה –**

בהתחלה השתמשתי רק בחיתוכים של התמונות עבור ה-dataset, לא הגדלתי אותו, לא הוספתי רעשים וכו'. והשתמשתי במודל בסיסי, שמכיל את השכבות הבאות:

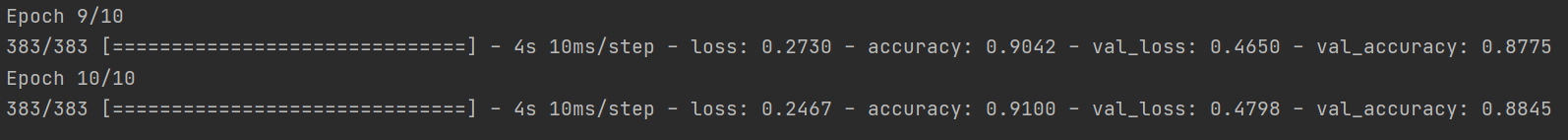
קפצו לי שגיאות רבות ולאט לאט הצלחתי לפתור אותם שגיאה לאחר שגיאה:

1. שמרתי את סט התמונות שלי כרשימה במקום כ-np.array.
2. שמרתי את התמונות בסט לפי BGR במקום גווני אפור.
3. שמרתי את הלייבלים כרשימה (כדי לשנות את זה השתמשתי בפונקציה to\_categorical של keras.utils).

לאחר שסיימתי לבנות את סט האימון בצורה נכונה ביצעתי את אותם פעולות עבור סט הוולידציה על מנת שאוכל לעשות fit למודל ולראות את התוצאות של ה-val\_loss ו-val\_accuracy.

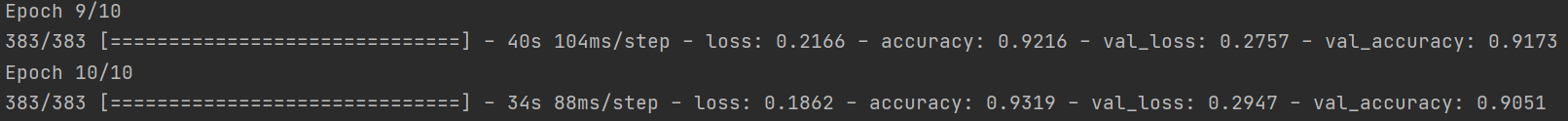
המודל עצמו –

הפונקציית התאמה שלי (model.fit) כללה את הדברים הבאים: x\_train- מכיל את התמונות בסט האימון, y\_train- הלייבלים של התמונות, epoch שווה ל-10 ואת סט הוולדיזציה (גם כאן x\_val ו-y\_val).

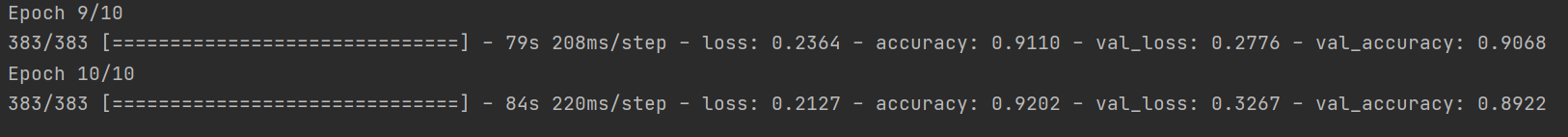
השתמשתי תחילה במודל הבסיסי שהזכרתי למעלה. והתוצאות כמובן לא היו טובות. התוצאה בשני ה-epochs האחרונים:

לאחר מכן שיניתי את המודל, ובניתי אותו מחדש בכך שהוספתי עוד שכבות, הוספתי עוד קונוולוציה, dropout. המודל החדש היה בנוי בצורה הבאה:

והתוצאות אכן השתפרו:



רציתי לשפר אותם עוד יותר, לטובת זאת הגדלתי את כמות הפילטרים בקונוולציות. בהתחלה הגדלתי את הקונוולציה הראשונה להיות עם 128 פילטרים. וראיתי שבאופן מפתיע (עבורי לפחות) במקום שהתוצאות יעלו הם דווקא ירדו וגם זמן הריצה של המודל עלה באופן משמעותי, לכן הבנתי שעליי לעשות משהו שונה.



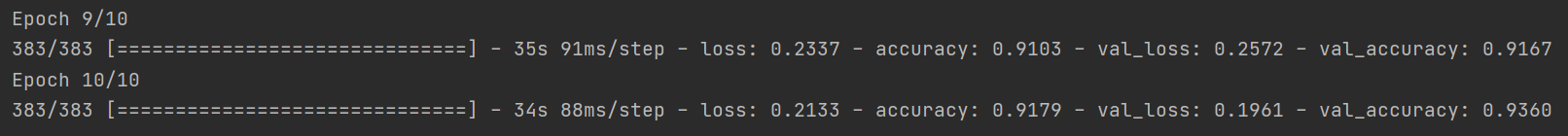
לאחר מכן עשיתי הרבה ניסוי וטעייה, וגיליתי את הדברים הבאים:

1. ככל שהגדלתי את כמות הפילטרים במודל המודל שלי רץ יותר זמן, ברמה של מספר שעות. מכיוון שהיו לי יותר פרמטרים. ומשום מה גם התוצאות שלי נהיו פחות טובות כי ככל הנראה היה overfitting, ראיתי זאת מכיוון שה-val\_loss עלה בכל epoch.
2. ככל שהוספתי יותר hidden layers, גם כאן נהיה לי overfitting.

דוגמאות נוספות לדברים שניסיתי:

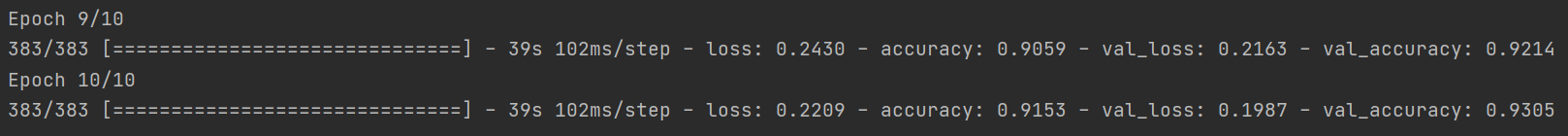
* המודל:

תוצאות של שני epoch אחרונים:

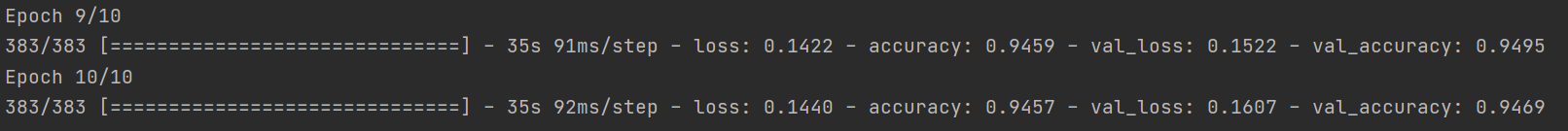


* המודל:

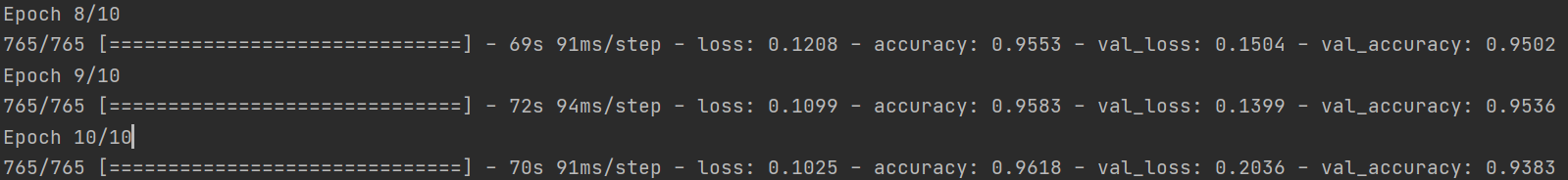
תוצאות של שני epoch אחרונים:



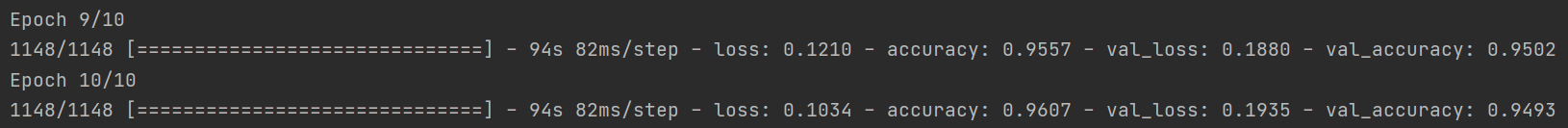
בסוף המודל שהניב לי את התוצאות הטובות ביותר בזמן קצר למדי הוא:



מכיוון שהתוצאות יכולות עוד להשתפר החלטתי בשלב זה להוסיף רעש לסט האימון שלי על מנת לשפר את הדיוק. הוספתי רעשים שונים וראיתי אם התוצאות השתפרו או לא. (הרעש נוסף בתוך הפונקציה שיוצרת את סט האימון ועבור כל תמונה הוספתי רעש ואת הלייבל המתאים).

1. Gaussian Noise – ניתן היה לראות שרעש גאוסיאני לא משנה כל כך את התוצאות, זה נובע מכך שכלל הנראה מכיוון שהתו החתוך הוא בכל מקרה מטושטש רעש גאוסיאני לא מוסיף לו מידע. והתוצאות נשארו די זהות.
2. Salt and Pepper Noise – את הרעש בניתי על התווים החתוכים שהם היו בגודל 400 על 400 פיקסלים, ואז עשיתי resize לתמונה המורעשת. ניתן היה לראות שאכן התוצאות השתפרו במקצת אבל ב-epoch ה-10 נוצר overfitting.

לאחר מכן, ניסיתי לעשות עוד שינוי בסט והוספתי את זה שעבור כל תמונה נוסיף לתמונה גם חדות, sharpe. אבל גם כאן התוצאות לא השתנו. אבל הזמן גדל לכן החלטתי לוותר על התמונות האלה.



לאחר שעשיתי את כל הדברים האלה רציתי לדעת כיצד אני יכול לשפר עוד יותר, ותחילה שיניתי את כמות ה-epochs, הרצתי 30 epochs כדי לראות את השינוי בערכים. וראיתי את הדברים הבאים:

1. ה-loss המינימלי היה ב-epoch האחרון, שערכיו הם:

loss: 0.0583 - accuracy: 0.9792 - val\_loss: 0.2714 - val\_accuracy: 0.9468

1. ה-accuracy המקסימלי היה ב-epoch ה-29 והאחרון, שערכיו הם:

loss: 0.0629 - accuracy: 0.9792 - val\_loss: 0.1981 - val\_accuracy: 0.9516

1. ה-val\_loss המינימלי היה ב-epoch ה-7, שערכיו הם:

loss: 0.1404 - accuracy: 0.9480 - val\_loss: 0.1690 - val\_accuracy: 0.9449

1. ה-val\_accuracy המקסימלי היה ב-epoch ה-18, שערכיו הם:

loss: 0.0806 - accuracy: 0.9701 - val\_loss: 0.1746 - val\_accuracy: 0.9585

1. בנוסף ראיתי שה-val\_loss הולך בצורת זיגזג וב-epoch האחרון הוא הכי גבוה מכאן הסקתי שנוצר overfitting מכיוון ששם גם ה-loss הוא המינימלי.
2. בכל epoch ה-accuracy עלה וה-loss ירד, כי המודל ניסה להתאים את עצמו כמה שיותר לסט האימון.

ואז שיניתי את ה-learning\_rate של המודל לראות כיצד זה ישפיע על התוצאות. במודל הנוכחי בחרתי כ-optimazier את Adam, שלו ה-learning rate הוא 0.001. וראיתי שכאשר הגדלתי אותו ל-0.01 התוצאות נהיו הרבה יותר גרועות, ה-val\_loss הגיע ל-0.9. ואז שיניתי אותו ל-0.0001, והתוצאות אכן נהיו יותר טובות והגיעו ב-epoch ה-25 ל-val\_accuracy של 0.9660, val\_loss של 0.1413, accuracy של 0.9777 ו-loss של 0.0574. ואז הקטנתי עוד יותר את ה-loss\_rating ל-, וראיתי שהתוצאות נהיו יותר גרועות.

מכל מה שראיתי והסקתי הגעתי למסקנה ש-25 epochs זה יתאים למודל שאותו בחרתי, עם learning rate של .

מכל מה שראיתי והסקתי הגעתי למסקנה שהמודל הכי מתאים הוא:

ב-compile:

ב-fit: כמות הepochs שווה ל-25.

**שלב החיזוי –**

ביום ראשון 17.1.2021, הועלה לאתר הקורס. סט בדיקה ועוד סט אימון. לפני שביצעתי חיזוי על סט הבדיקה רציתי לבדוק את החיזוי שלי על מול סט האימון הנוסף.

ביצעתי את החיזוי באופן הבא:

שמרתי את המודל כקובץ h5 והכנסתי בו את המשקלים. לאחר מכן, טענתי את המודל ואת המשקלים ששמרתי. ועשיתי predict על סט האימון. הpredict החזיר לי רשימה עבור כל תמונה ובו יש את ההסתברות שהתו הנ"ל הוא בפונט המתאים (המיקומים ברשימה: 0 = skylark, 1 = ubuntu mono, 2 = sweet puppy).

ואז עברתי במקביל על כל איבר ברשימה מול כל איבר בסט, ושמרתי את זה כ-csv. כך שעבור כל תו קיימת לי שורה בקובץ והעמודות הם: SN (serial number, מס"ד), image (שם התמונה), char (התו שעליו אני מבצע את החיזוי), real skylark, real sweet puppy, real ubuntu mono, predict skylark, predict sweet puppy, predict ubuntu mono (כך שעבור העמודות של הreal יש לי רק עמודה אחת שמכילה 1 עבור כל הפונט האמיתי, ובעמודות ה-predict יש לי רק עמודה אחת שמכילה 1 עבור הפונט שחזיתי).

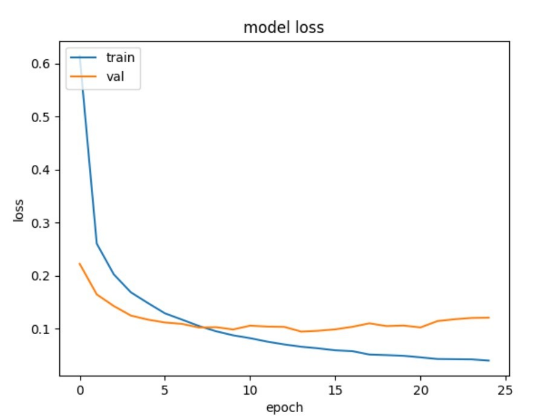
לבסוף אחוזי הנכונות שבו צדקתי בחיזוי שווה ל-93.62%.

שמתי לב שרוב הטעויות שלי היו יכולות להימנע אם הייתי מתייחס לכל המילה שאני יודע שהיא מאותו פונט. לכן החלטתי לעשות את הדבר הבא – אנחנו יודעים שכל מילה היא מאותו פונט, לכן אני יכול "ללכת על פי הרוב". למשל, במידה ויש לי מילה שמכילה 7 תווים מתוכם עבור 4 תווים חזיתי skylark, עבור 2 תווים חזיתי sweet puppy ועבור תו אחד חזיתי ubuntu mono, אז אני יכול להגיד שככל הנראה כל המילה היא מפונט של skylark. במידה ותהיה לי תיקו בין שני תווים עבור המקום הראשון, אני אעדיף להשאיר זאת לפי מה שחוזה. האופן שבו עשיתי זאת הוא לשמור מערך של המילים ו-counter עבור כל חיזוי התווים במילה ספציפית. לאחר מכן ביצעתי השוואה בין ה-counter לראות מי יותר גדול או האם יש תיקו על ה"מקום הראשון".

לאחר שעשיתי את "ההליכה על פי רוב" אחוזי הנכונות שבו צדקתי בחיזוי היה שווה ל-96.54%.

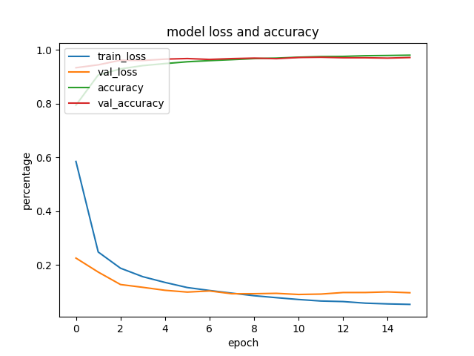
שני הקבצים הנ"ל שמורים בתוך drive של הפרוייקט בתיקייה 'results on train set'.

הגעתי לתוצאות שמהם אני מרוצה באופן כללי, תוצאות שנותנות לי חישוב הגיוני. לכן החלטתי להוסיף את סט האימון הנוסף לסט שלי ולבנות את המודל מחדש ביחד עם הסט החדש.

ובסוף התוצאות היו:

loss: 0.0467 - accuracy: 0.9820 - val\_loss: 0.0974 - val\_accuracy: 0.9734. והריצה של המודל הייתה בזמן של שעה וחצי.

לכן הבנתי שמתחיל להיווצר overfitting. אז בניתי את המודל מחדש, רק שהפעם הוספתי callback של earlyStopping, שהמדד שלו הוא val\_loss, מינימום והסבלנות שלו (patience) היא של 5 איטרציות. בנוסף הוספתי callback של check point, כדי שישמור לי את המודל הטוב ביותר, שבו ה-val\_accaurcy הוא המקסימלי.

הרצתי עוד פעם את המודל, הוא עצר ב-epoch ה-16. המודל רץ במשך כשעה. והגרף loss ו- accuracyהוא זה:

המודל שנשמר הוא המודל הטוב ביותר.

שעשיתי evaluate, ראיתי שה-val\_accaurcy הוא 0.972.

וה-val\_loss הוא 0.091.

אחרי שעשיתי את כל הדברים שהזכרתי לעיל, הרצתי חיזוי על סט הבדיקה. ובניתי קובץ csv שמפרט את החיזויים שלי עבור כל תו, בנוסף בניתי csv נוסף שמפרט את החיזויים שלי עבור כל מילה (אני יודע שזה לא נדרש אבל הרגשתי צורך לבצע את זה בכל זאת).

ה-csv של התווים מכיל את העמודות הבאות: SN (serial number, מס"ד), image (שם התמונה), char (התו שעליו אני מבצע את החיזוי), skylark, sweet puppy, ubuntu mono (כאשר עבור כל שורה רק עמודה אחת מכילה 1, עבור החיזוי, וכל השאר 0-ים).

ה-csv של המילים מכיל את אותם עמודות רק שבמקום העמודה char יש לי את העמודה word עבור המחרוזת של המילה. אם במילה אחת יש תיקו על המקום הראשון בין שניים כל אחד מהפונטים שבתיקו מקבל 0.5, אם יש תיקו בין שלושתם כל אחד מקבל 0.333.

**לסיכום:**

התהליך של הפרוייקט היה חוויה מעניינת מאוד ומלמדת, אני ממש מצפה ליום שבו אני אוכל לעבוד בתחום של deep learning ולעשות פרוייקטים כאלו במסגרת יומיומית.