**פרויקט ראייה ממוחשבת**

**מגיש**: ניצן קרני, 208939215

**קריאת הנתונים:**

את הנתונים קראתי מתוך קובץ h5 אשר היה נתון. הקובץ כולו הכיל את הנתונים בהיררכיה הבאה:

הנתונים מחולקים פר תמונה.

להלן תמונה לדוגמא:



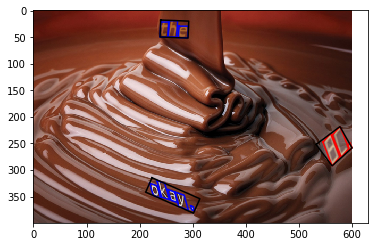
ממבט ראשון קל להבחין כי התמונות מכילות מילים שונות בפונטים שונים. ליתר דיוק כל מילה מורכבת מאותיות, המילה כולה מוצגת באחד משלושת הפונטים שאותם בעצם אנו צריכים לדעת לסווג.

עבור כל תמונה קיימות מסגרות משני סוגים, מסגרות אשר מסמנות את גבולות המופע של מילים. ומסגרת פר אות שמעידה על מיקומה וגבולותיה של כל אות בתמונה כולה.

עבור כל תמונה קיים מערך של טקסט המילים המופיעות בתמונה. למשל עבור התמונה המוצגת המערך הבא קיים:

[b'okay.', b'the', b'BC,']

בנוסף עבור התמונה כולה קיים מערך של פונטים. המערך מסודר כך שהפונט הראשון מעיד על הפונט של האות הראשונה במילה הראשונה במערך המילים וכן הלאה באותו אופן. לשם הבחנה בגבולות הנתונים נציג את הגבולות על גבי התמונה:



**עיבוד הנתונים**:

לאחר שקראתי את הנתונים החלטתי שזה יהיה רעיון טוב להתחיל דבר ראשון בלהצליח לחלץ תמונה של כל אות בנפרד מתוך כל התמונות וכך ליצור בסיס נתונים שיותר מתאים לעיבוד. את תהליך החילוץ ביצעתי בעזרת הפונקציה characterCrop אשר עושה את הדבר הבא:

הפונקציה בכלליות מקבלת את התמונה וגבול של ארבעה נקודות על גבי התמונות ומחלצת ממנו את המצולע שנתון על ידי ארבעת הנקודות הנתונות.

קודם כל אני בודק שאין נקודות אשר חורגות מגבולות התמונה. ( על מנת שאוכל לחתוך את התמונה כמו שצריך ולא להתקל בשגיאות של נסיון "לקחת" יותר ממה שיש ).

לאחר מכן אני יוצר מצולע של OPENCV מארבעת הנקודות. ראשית אני חותך בפשטות מלבן מתוך התמונה. נמשיך בליצור MASK לתמונה המלבנית שנחתכה אשר מכיל בלבן את החלק בו המצולע נתון במלבן ובשחור את החלק מחוץ למצולע במלבן, את הMASK ניצור מתוך הנקודות שבין המצולע הנתון לבין המלבן. לבסוף נפעיל פעולת AND בין התמונה לבין הMASK ונצבע בממוצע הצבע של התמונה את הגבולות שבין המצולע למרובע שנחתך.

**האות לאחר חיתוך:**



**האות לאחר צביעת הרקע:**

****

מהשוואה קלה בין שתי התמונות קל להבין את המוטיבציה לצבוע את הרקע ולא להשאיר חיתוך מלבני. נשים לב שהאות y מעט "פולשת" לתמונה של האות a , קיימים הרבה מופיעים של התופעה הזאת לכן כדי לשמור על כל אות בפני עצמה העדפתי להפטר מהמידע העודף הזה שעשוי לבלבל את המודל שלנו.

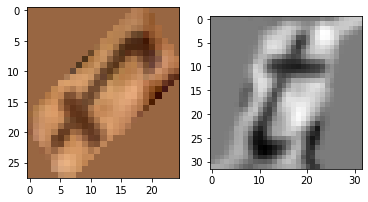
את הרקע צבעתי בממוצע על מנת להמנע ממצב בו המודל יזהה את המעבר החד בין הרקע השחור לבין האות בתור edge שמהווה חלק מהאות עצמה. שימוש בממוצע עוזר לרכך את הedge.

את האותיות נייצג בתמונות גווני אפור בגודל 32X32 על מנת ליצור מסד נתונים אחיד ככל שאפשר אשר ההבדל היחיד בין התמונות שלנו יהיה הצורה של האות. נשיג זאת על ידי פונקציות resize, cvtColor של openCV

בנתונים שלנו קיימות אותיות שנראות מיושרות וקריאות בקלות כמו “the” בתמונה הנתונה אך קיימות גם מילים שמופיעות באוריינטציה שמקשה על זיהוי המילה ויכולה להוות נקודה חלשה של המודל. כדי לעזור למודל שלנו כמה שאפשר עוד לפני שיקבל בכלל את הנתונים נשתדל ליישר את האותיות כמה שאפשר. נחשב את הזווית ביחס לציר האופקי של האות. ( באמצעות נקודות המסגרת של התמונה ) נסובב את התמונה בזווית הנגדית לזווית שחושבה.

הנקודות במסגרת האות מתחילות בפינה השמאלית העליונה של האות וממשיכות עם כיוון השעון. נבדוק האם הנקודה הראשונה במסגרת שמעידה על הפינה השמאלית עליונה של האות נמצאת מתחת לנקודת הפינה השמאלית תחתונה של האות (הנקודה האחרונה), אם כן זו אינדיקציה שהאות מופיעה באוריינטציה הפוכה ולכן גם נסובב את האות ב180 מעלות וגם נסובב את הכיוון שלה.

לדוגמא:



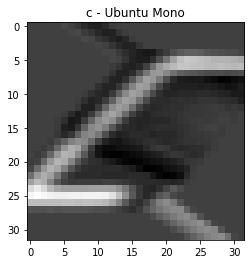
כעת ניתן לומר שהאותיות שלנו חתוכות, מיושרות ובגודל אחיד. ניתן להמשיך בהכנת המידע למודל.

לטובת ניווט קל בין הנתונים לצורך מחקר יצרתי מכל המידע מילון אחד שמכיל את כל התמונות, עבור כל תמונה את כל המידע הנחוץ כמו תמונות המילים, אותיות, הטקסט עצמו, הגבולות ועוד.

**נקודת המכשול**

כשבניתי את המילון שאפשר ניווט נוח במידע עשיתי טעות קריטית ביותר שעלתה לי בכמעט שבועיים של עבודה ודריכה במקום. יצאתי מנקודת הנחה שמערך הפונטים מתאר פונט אחד פר מילה. לכן את המילון בניתי באופן הבא:

כך נוצר מצב שבניתי את המידע שלי כך שאת הפונטים, כלומר היעד של הסיווג אותו אני אמור לבצע היה לי נתון בצורה שגויה ומטעה. עבור כל אות התאמתי את הפונט של האות במקום i כאשר i מייצג את המיקום של המילה במערך המילים בטקסט. כלומר עבור התמונה הראשונה שלנו אני אשתמש במידע בצורה הבאה:



אני השתמשתי במידע כי האות בפונט Ubuntu Mono למרות שברור כי היא בפונט Skylark .

עקב התקלה הזאת שחלפה מעל אפי מצאתי את עצמי מתחבט בתוצאות נוראיות של מודלים מתוחכמים שונים ומשונים. תוצאות אשר גורמות לי לחשוב שאולי עדיף פשוט לבחור אקראית פונט לכל אות מאשר להשתמש במודל.

בין היתר מצאתי את עצמי מתחיל לנסות להתמודד עם הבעיה באמצעות BoW , ממשיך לזיהוי features עם SIFT והזנה בNN מתוחכם. שימוש בVGG16 (שהמחשב שלי לא בדיוק התמודד איתו יפה בלשון המעטה), ImageNet ובכל טכניקה אפשרית כדי להצליח להתגבר על המכשול.

הייתי בטוח שהבעיה היא במודל ואני פשוט לא השתמשתי עד עכשיו במודל הנכון אבל כשאני אשתמש בו אני סוף כל סוף אראה תוצאות מבטיחות.

ככה ביליתי לא מעט זמן בקריאת מאמרים, צפייה בחומר על tensorflow, keras נסיון להגדיל את כוח העיבוד עם כרטיס גרפי, מייצר עוד כמויות של מידע מתוך הפונטים הנתונים ושיטות data augmentation .

כאשר יצרתי אותיות בפונטים שונים אוריינטציות שונות על רקעים שונים עם פילטר salt&pepper התחלתי להתקרב לגילוי שורש הבעיה.

אימנתי את המודל עם המידע הסינטטי ובדקתי אותו על התמונות עצמן. פתאום החל לעלות הtraining accuracy כמובן שtest accuracy היה נוראי. בהתחלה הרגשתי שאולי מדובר במקרה פשוט של overfitting. ניסיתי להגדיל את הכמויות המידע הסינטטי עד שהגיע ליותר מפי 2 מכמות המידע האמיתי בסט האימון. ושוב, הtraining accuracy אפילו עלה אך הtest accuracy נשאר כשהיה אם לא יותר גרוע. החלטתי שאולי כדי רק לנסות לראות מה קורה אם אני משתמש אך ורק במידע סינטטי. מיותר לציין שקיבלתי תוצאות נהדרות.

התחלתי לחקור את המידע עצמו. הצגתי בתרשים כמות גדולה של אותיות מתוך המידע האמיתי והבעיה האמיתית נגלתה לנגד עיני.

נתקלתי במספר לא מבוטל של מופעים בעייתיים של אותיות אשר הפונט שהותאם להן היה לא נכון.

לאחר עוד קצת מחקר הגעתי לשורש הבעיה. וכמובן שלאחר שינוי של שורת קוד בודדה, האחראית על התאמת הפונט לאות. הרצתי שוב והגעתי לתוצאות הaccuracy המצויינות של 86% על סט הבדיקה.

כעת, אפשר להמשיך.

**סידור נתונים**

המידע עצמו שאותו נזין למודל מורכב ממערך המכיל את כל התמונות החתוכות של כל האותיות מכל התמונות שהיו נתונות לנו במסד הנתונים בקובץ.

בנוסף לכל תמונה מותאמת תגית המתארת את הפונט של האות שמוצגת בתמונה.

זהו מערך המחלקות:

['Ubuntu Mono', 'Skylark', 'Sweet Puppy']

כל מחלקה מייצגת פונט. את תגיות היעד שלנו נייצג בתור מערך באורך 3 כך שאם הפונט הוא הפונט שנמצא במקום הi במערך המחלקות אז המערך התגית יופיע במקום הi 1 ואילו בשאר המקומות יופיע 0.

כלומר, אם הפונט הוא Skylark התגית תיוצג כך: [0,1,0]

נחלק את הנתונים ל50% אימון ו50% בדיקה.

**סינטזת נתונים**

כדי שנוכל לאמן את המודל על יותר נתונים כדי שיוכל להיות כללי יותר ולהמנע יותר מoverfitting נסנתז בעצמנו כמויות גדולות של מידע ונשתמש בו כדי לאמן את המודל.

במהלך קריאת הנתונים ניצור סט אשר מציג לנו את כל האותיות שמופיעות בנתונים.

**['!','"','#','$','%',"'",'(',')','\*','+',',','-','.','/','0','1','2','3','4','5','6','7','8','9',':',';','<','=','>','?','@','A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','[',']','\_','`','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','|']**

עבור כל אות, נצייר אותה בשחור על תמונת רקע בגודל של 32X32 בשלושת הפונטים השונים.

במודל הנתון הזה השתמשתי ב100 תמונות רקע אשר מורכבות מהריבוע ה32X32 העליון שמאל של התמונות הראשיות ממסד הנתונים. באופן הבא יצרתי 88\*3\*100=26400 תמונות שונות של אותיות שונות.

כדי להגדיל את השונות בין התמונות ביצעתי עליהם סיבוב רנדומלי של זווית בין 45 ל-45 מעלות, היפוך האות בכיוון אופקי עם סיכוי של 1 ל10 ופילטר של salt and pepper. וכמובן שכדי להתאים לסט האמיתי נמיר את התמונה לצבעי אפור.

את הסט המסונטז נשרשר לסט האימון שלנו ונערבב בין הדגימות על מנת שלא יהיה "גוש" מסונתז ו"גוש" אמיתי.

**מודל 1**

בתור מודל בחרתי לאמן Convolutional neural network. את הרשת מימשתי באמצעות ספריית keras של tensorflow. להלן סיכום שכבות הרשת:

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d (Conv2D) (None, 30, 30, 32) 320

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

layer\_normalization (LayerNo (None, 30, 30, 32) 60

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d (MaxPooling2D) (None, 15, 15, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_1 (Conv2D) (None, 13, 13, 64) 18496

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 6, 6, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_2 (Conv2D) (None, 5, 5, 128) 32896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_2 (MaxPooling2 (None, 2, 2, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 88) 45144

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 264) 23496

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 3) 795

=================================================================

Total params: 121,207

Trainable params: 121,207

Non-trainable params: 0

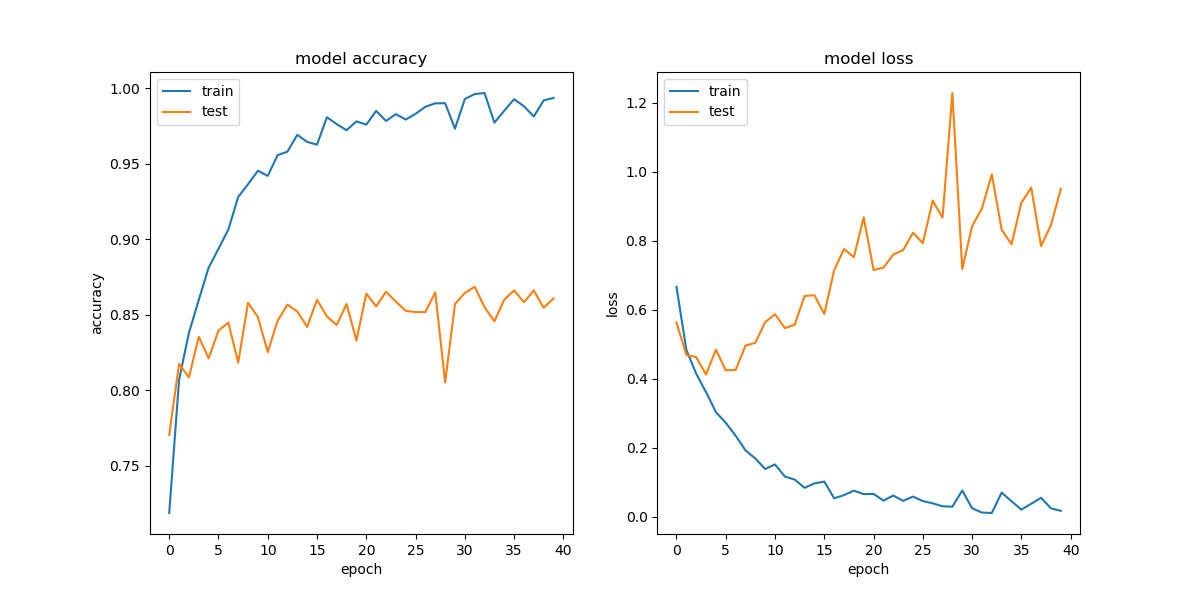
בשכבה הראשונה נשתמש ב32 פילטרים בגודל 3X3 עם פונקציית אקטיבציה relu שיעברו על התמונה. לאחר מכן נעבור לשכבת נורמליזציה שתנרמל את התוצאות.

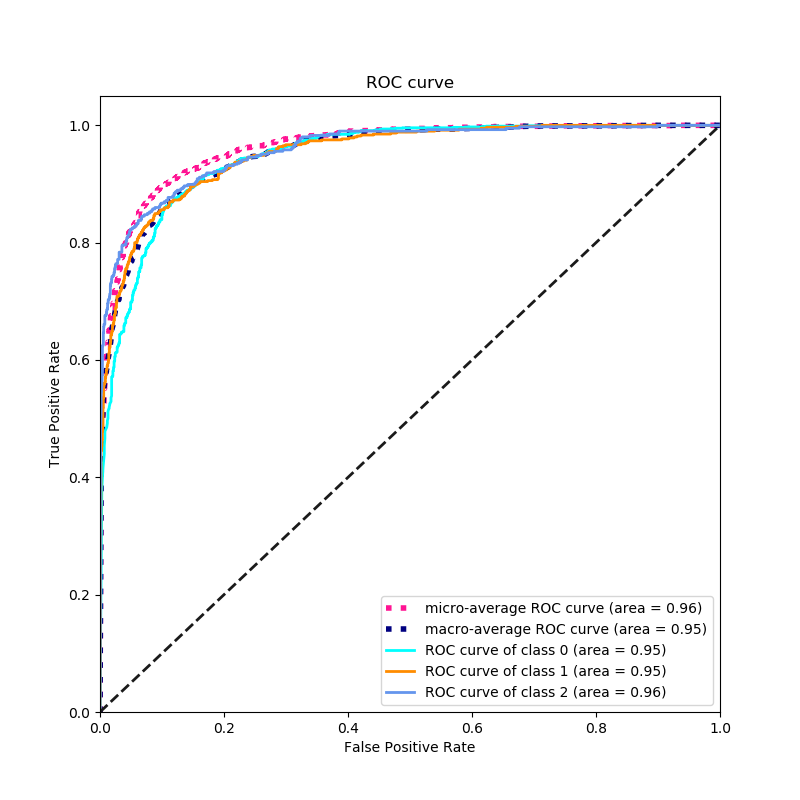
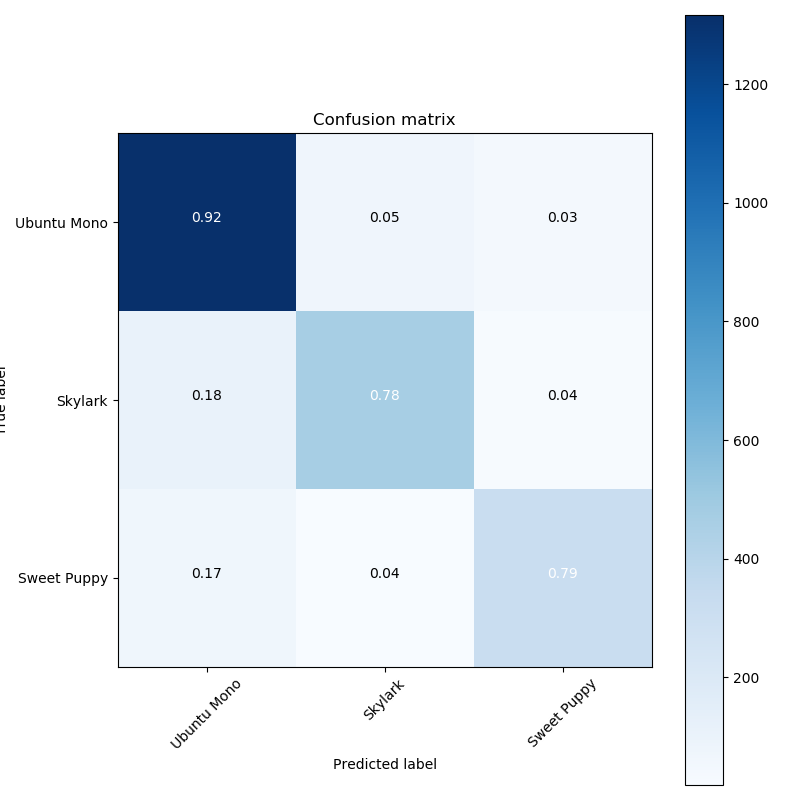
לאחר מכן נבצע max\_pooling, וכך נחזור על התהליך עבור שתי השכבות קונבולוציה הבאות כאשר מספר הפילטרים הולך וגדל על חזקותיו של 2. לאחר מכן נשטח את הנתונים ונעביר אותם לשכבה נסתר עם 88 נוירונים ( נוירון לכל אות ), לעוד שכבה עם 264 נוירונים ( נוירון פר אות פר פונט ) ולבסוף נצמצם לפלט של 3 נוירונים בלבד כאשר נשתמש בsoftmax.

בפונקציה ההפסד נשתמש ב’categorical\_crossentropy’ ובoptimizer נשתמש בadam .

את האימון נריץ על 40 איטרציות של מעבר על כל הדגימות. תהליך האימון כולו לקח כשעה.

מפרט טכני: מעבד: intel-i5-8250U ,זיכרון RAM: 8GB

**תוצאות**



מטריקות:

Accuracy: 0.8754085

Recall: 0.872549

Persision: 0.8815518

AUC: 0.9617097

**מודל 2**

ניכר מגרף הלמידה כי המודל שלנו סובל מover fitting אחת הדרכים להלחם בכך היא באמצעות שימוש במודל פשוט יותר, נסיר את השכבות conv2d\_2, max\_pooling2d\_2, dense\_1 מהרשת.

Model: "sequential\_9"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_20 (Conv2D) (None, 30, 30, 32) 320

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

layer\_normalization\_8 (Layer (None, 30, 30, 32) 60

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_20 (MaxPooling (None, 15, 15, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_21 (Conv2D) (None, 13, 13, 64) 18496

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_21 (MaxPooling (None, 6, 6, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_9 (Flatten) (None, 2304) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_20 (Dense) (None, 88) 202840

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_21 (Dense) (None, 3) 267

=================================================================

Total params: 221,983

Trainable params: 221,983

עוד בעיה שהייתה לנו היא זמן האימון הארוך שלקח למודל. על מנת לצמצם אותו נקטין את מספר האיטרציות ל20 וננסה לראות מה קורה כאשר לא משתמשים כלל בנתונים במסונתזים.

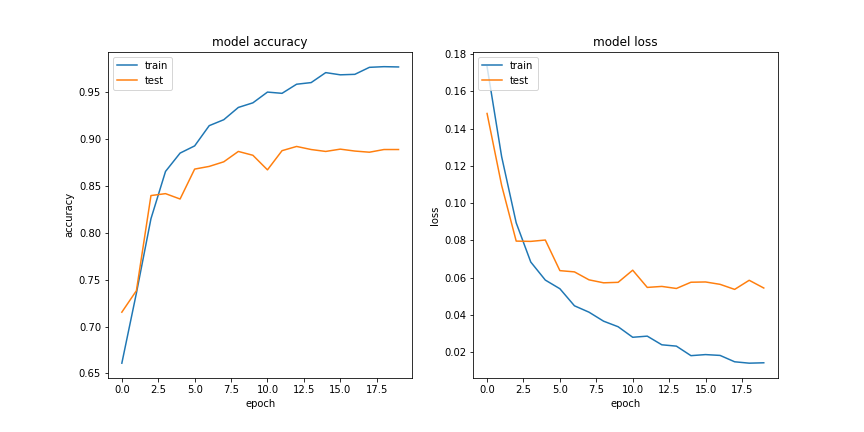
כלומר: סט אימון 80% מהמידע שקיבלנו (9790 דגימות) , סט בדיקה 20% מהמידע שקיבלנו (2448 דגימות).

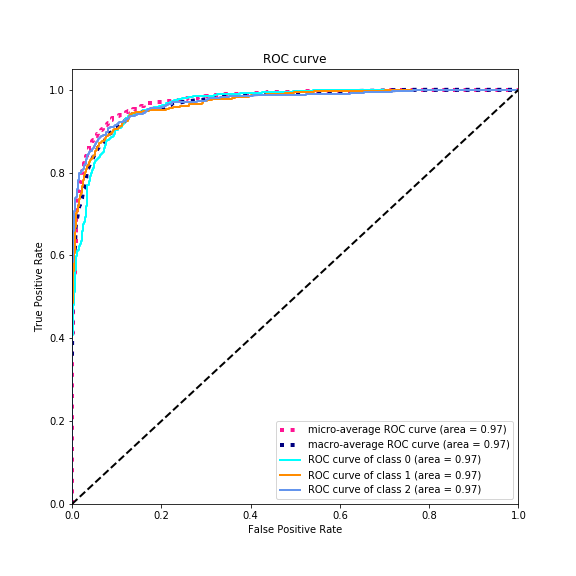
נשמור את המודל מהאיטרציה המוצלחת ביותר בממקסמת את validation\_accuracy.

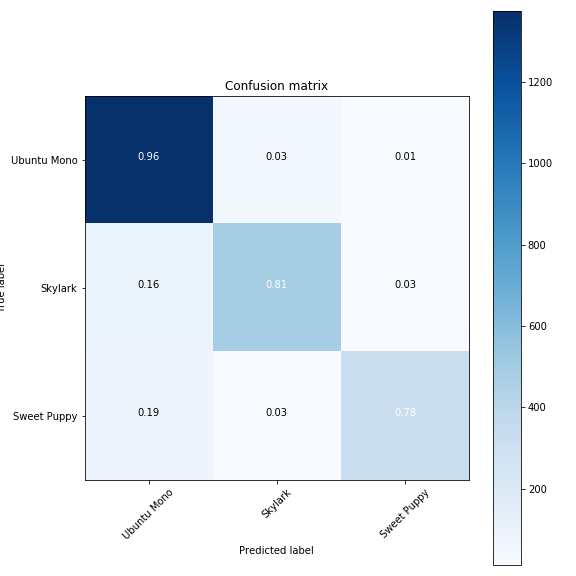
**תוצאות**

ניכר כי התוצאות במעט יותר טובות על ביצועים יותר טובים. לקח לי כ7 דקות לאמן את המודל (כמעט פי 10 מהר יותר)

בנוסף ניתן להבחין כי המודל פחות ניחן בoverfitting.







Accuracy: 0.89174837

Recall: 0.88398695

Persision: 0.89718074

AUC: 0.97307014

**מודל 3**

בעוד נסיון להעלות את אחוזי הדיוק של המודל ננסה לנצל את העובדה שיש פיסת מידע מאוד חשובה שאנחנו כלל לא משתמשים בה באימון המודל שלנו. הטקסט של האות. ננסה לשלב את הידע שיש לנו על כל תמונת אות ועל הטקסט שלה בהתאם.

הרעיון יהיה כזה: ניצור 88 מודלים שונים, כל מודל יהיה אחראי לסווג אך ורק תמונות של האות עליה המודל אחראי.

באופן דומה אותו מודל גם יקבל אך ורק את אותה האות בפונטים ותצורות שונות. בשל חסך של מידע באותיות שונות שמופיעות לעיתים תכופות פחות ( כמו למשל ';' ) נשתמש בכמות גדולה של אותיות מסונתזות בסט האימון שלנו.

נחלק את המידע האמיתי שלנו ל50/50 סט אימון וסט בדיקה. נאחד את סט האימון עם הסט המסונטז. וכעת נעבור על מערך האותיות. עבור כל אות נאמן מודל חדש עם אותן הגדרות כמו במודל 2, נחשב את מטריקות הבדיקה הנחוצות ונכניס למילון המחולק על פי אותיות.

**תוצאות**

תהליך האימון כוללו לקח כרבע שעה דקות.

**מטריקות**

Total Accuracy: 0.8710074159977871

Total Recall: 0.8626810145831688

Total Persision: 0.8754866569975543

Total AUC: 0.9452160020040177

ניכר כי המודל מראה ביצועים פחות טובים ויעילות פחותה ממודל 2, לכן נוותר עליו כבר בשלב זה.

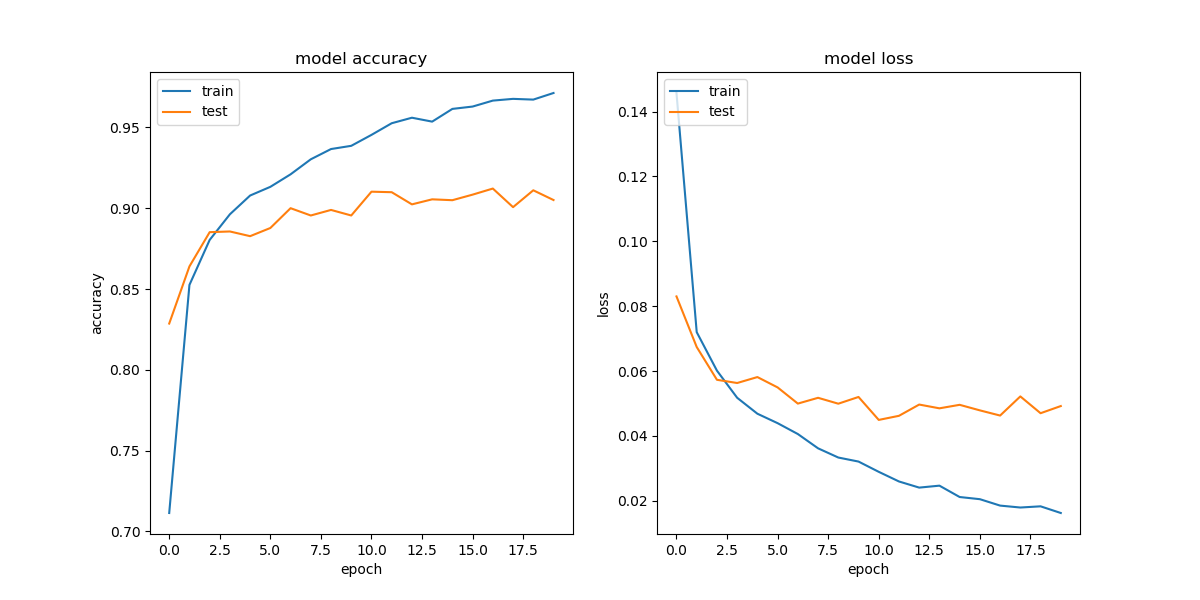
מודל 2 הפשוט ביותר מראה את התוצאות הטובות ביותר מבין שלושת המודלים ולכן נבחר בו בתור המודל שלנו. הרעיון הוא 2 שכבות גדלות של קונבולוציה עם שכבת נורמליזציה, שכבות SOFTMAX בינהם ולבסוף שכבה נסתרת המכילה 88 נוירונים, אחד לכל אות. ויציאה החוצה ל3 מחלקות בהתאם.

89% דיוק בסיווג של 2448 דגימות הינה תוצאה לא רעה כלל ואני מאמין בעזרת עוד נתוני אמת נוכל להגדיל את דיוק המודל ואת הכלליות שלו שכן המודל יחשף לעוד מידע ובכך יהיה פחות נתון לoverfitting ממנו הוא סובל כרגע.

**תוספת מידע**

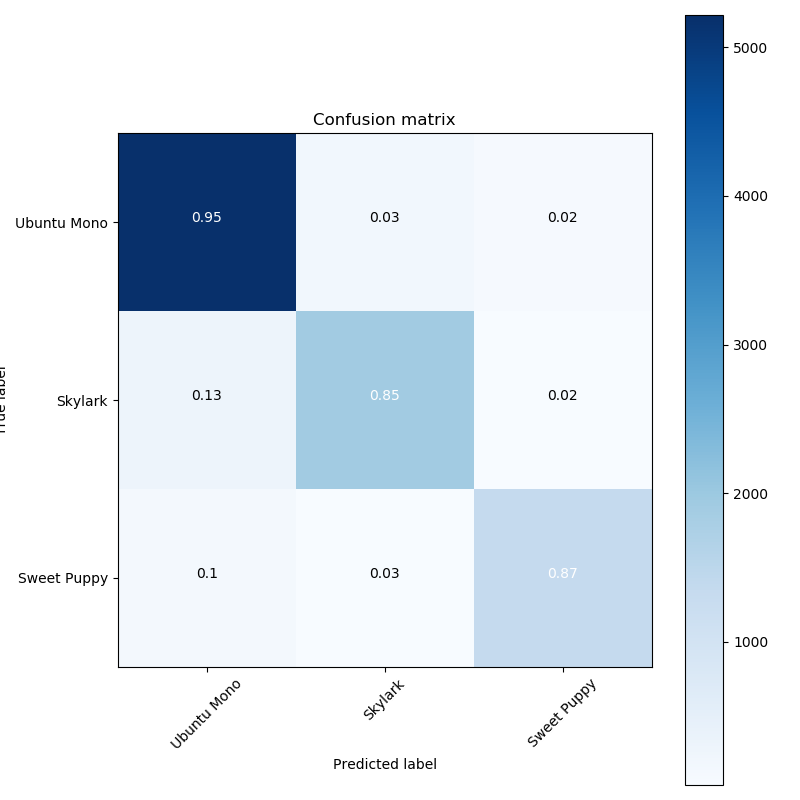
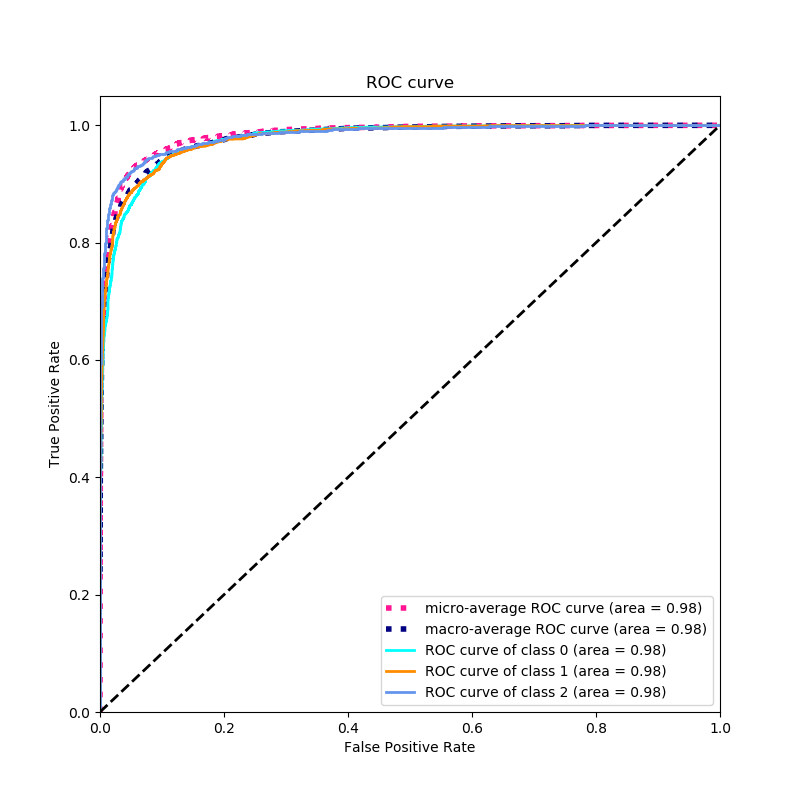
לאחר שהוספנו כ30000 דגימות ניתן לראות שיפור לא מבוטל בביצועים כפי שחזינו. זאת לאחר שהוספנו לסט האימון שלנו את סט הוולידציה ואת סט האימון הנוסף.

Accuracy: 0.912

Recall: 0.908

Persision: 0.917

AUC: 0.979

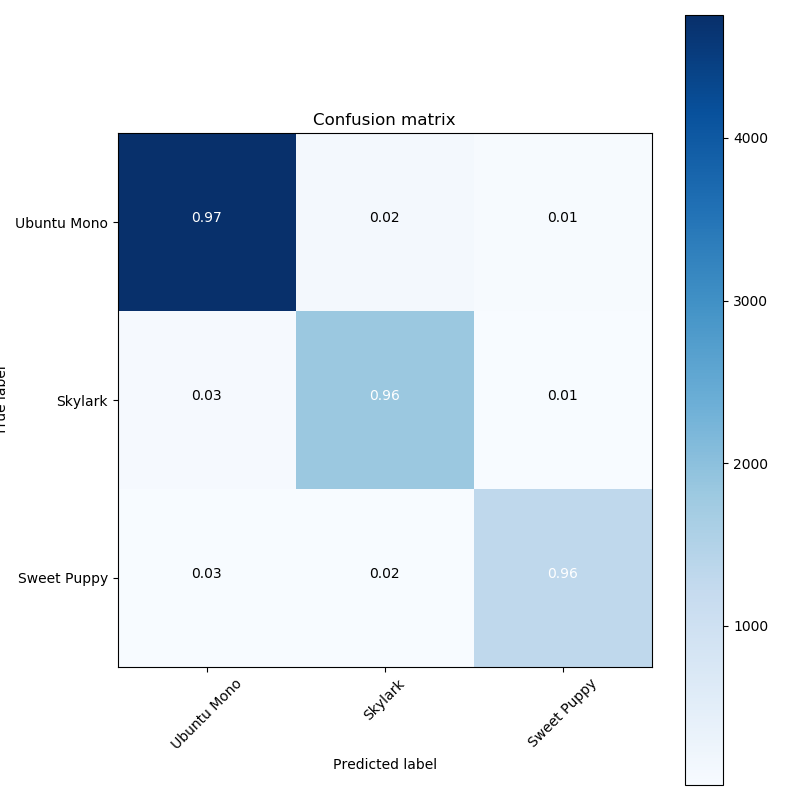


בנוסף נשים לב לconfusion matrix שקיבלנו מהמודל, ניתן לראות כי המודל יותר כללי ומזהה הרבה יותר טוב מהמודל הקודם פונטים שאינם Ubuntu Mono דבר שהמודל הקודם לא היה מאוד טוב בו.

**Fine tuning**

לאחר שקיבלתי את סט הבדיקה העפתי מבט בתוצאות ולהפתעתי הרבה היו לא מעט מילים בהן רוב המילה סומנה כפונט מסויים וכמה אותיות חריגות סומנו כפונט אחר. על פי הנתונים ידוע כי כל מילה חייבת להכיל בהכרח פונט אחד בלבד. וכל האותיות במילה הן מאותו הפונט. לכן קיימתי "הצבעת רוב" עבור כל מילה שחזרה מהמודל וסימנתי את כל האותיות שבמילה בפונט שהוחלט על המילה.

התוצאות על סט הולידציה לאחר השינוי הקטן הזה מאוד מבטיחות. הצלחנו לעלות ל96.3% accuracy.

Accuracy: 0.96377164

Recall: 0.95401317

Persision: 0.96627134

AUC: 0.9784279

**לסיכום**

לאחר המסע המפרך הצלחנו להגיע למודל בעל ביצועים מרשימים מאוד, המודל אינו מסובך עד כדי כך. CNN פשוט בעל 2 שכבות קונבולוציות ו2 שכבות נסתרות. כמובן שכמות המידע השפיעה במידה לא מבוטלת על הצלחתנו. בנוסף בחירת רוב היוותה כאן נקודת מפנה מצוינת שנתנה את האקסטרא מייל למודל שלנו.