



הפקולטה למדעי המחשב

הטכניון

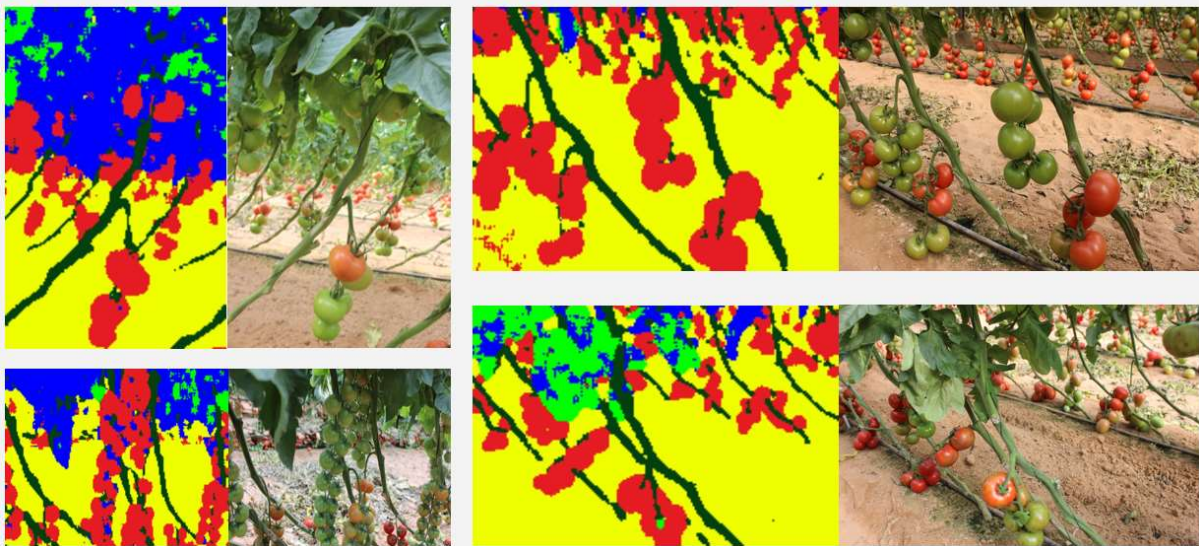


דוח פרויקט בעיבוד תמונה

Tomato Classification Project

איתן קוסמן ושחק מורג

מנחים: אלון זבירין, ירון חונן



קיץ 2018

תוכן עניינים

2	תקציר
3	מבוא
6	תיאוריה
9	פרטים טכניים
10	DATASET
12	מודלים ראשוניים
18	השפעה של טרנספורמציות צבע וגדלי התמונות על אחוזי הדיוק
19	בדיקה ויזואלית
21	ENCODER DECODER
21	AUTO ENCODER
22	מטרות
23	מימוש
25	תוצאות
26	מלפפונים ואוטומציה
29	References

הפרויקט נערך במסגרת מחקר לזיהוי פנוטיפים בצמחים, הנערך במספר מוסדות מחקר. אבחנה של פנוטיפים בצמחים היא בעיה נרחבת שמשפיעה על חיי המונים ברחבי העולם. המטרה העיקרית בפרויקט היא לבנות מערכת אמינה ומהירה שתבצע הפרדה של חלקי צמחים. השיטות שלנו לפתרון הבעיה משתמשות ביישומים של רשתות נוירונים.

בפרויקט בחנו מספר ארכיטקטורות של רשתות, בעיקר רשתות המסווגות קלטים ורשתות מסוג encoder-decoder. על-ידי שימוש ברשתות הסיווג השגנו אחוזי דיוק גבוהים (97%-98%) בסיווג תמונות של חלקי צמחים, אך זמני הריצה היו ארוכים והגיעו למספר דקות עבור תמונה בודדת. לאחר מכן, השימוש בארכיטקטורת encoder-decoder קיצר את הזמן הדרוש להפרדה של תמונה מדקות לשניות בודדות.

זיהוי עצמים בתמונה על-ידי רשתות נוירונים זה תחום מחקר נרחב. מלבד זאת, לרשתות נוירונים קיימים שימושים רבים במשימות של עיבוד תמונה. אחד השימושים הנפוצים הוא סיווג (classification) של תוכן תמונות. לטכניקה זו יש מגוון רחב של יישומים, לדוגמא: רפואה (זיהוי גידולים סרטניים בתצלום של איברים פנימיים), חיזוי מזג-אוויר (על-ידי ניתוח תצלומי לוויין), נהיגה אוטונומית (שליטה על כלי-הרכב על-פי תמונת מצלמה) ועוד.

רשתות הנמצאות בשימוש בתחומים אלה נדרשות לעמוד באחוזי דיוק גבוהים במיוחד, מאחר ולפלט שלהן יש השפעה מכרעת על קבלות החלטות (לדוגמא, האם כלי רכב אוטונומי יבלום בזמן הופעת הולך רגל מולו). לכן, כל רשת כזו צריכה להתאמן על מידע רב, זמן רב, כדי שתוכל לדעת לסווג מגוון רחב של מצבים על-פי נתונים עתידיים.

בתהליך האימון יש משתנים רבים הקובעים את אחוזי ההצלחה של פעולת הרשת: מבנה הרשת, Hyper-Parameters (פרמטרים הנקבעים בתחילת תהליך האימון), המידע עליו הרשת מתאמנת, ועוד מגוון רחב של משתנים.

בפרויקט זה נציג מימוש ארכיטקטורות שונות של רשתות שמטרתן להשיג הפרדה איכותית של חלקי צמחים בתמונה, לפי מחלקות שהוגדרו מראש. המחלקות מוגדרות על-פי תיוגים שקיבלנו ממנחי הפרויקט. אנו נתמקד ברשתות שמטרתן לסווג תמונה, ולאחר מכן נמשיך לסוג אחר – Encoder Decoder – שמטרתן להפריד אזורים בתמונה.

Patch based Neural Network

מאחר ואין ברשותנו Dataset מספיק גדול של תמונות מופרדות, הוחלט לייצר patches מתוך תמונות שצולמו בשדות ולתייג אותם. ה-Dataset הוכן ע"י מנחי הפרוייקט והוא כולל פאטצ'ים הלקוחים מתוך תמונות של 7 זנים עם 34 תתי-זנים של עגבניות.

Encoder-Decoder Neural Network

לאחר שהשגנו תוצאות מספקות מהרשת הראשונה, היה בידנו יכולת לייצר ground-truth מספיק גדול כדי לקבל הפרדה מלאה ומהירה של תמונות באמצעות רשת-encoder-decoder. האימון שלה מסתמך על תוצאות הרשת הראשונה, והיא שואפת להשיג תוצאות כמו של הרשת הראשונה באופן מהיר יותר. ה-Dataset של רשת זו הוכן על-ידינו. היצירה שלו כללה גזירת פאטצ'ים מתוך התמונות של העגבניות והפעלה של הרשת הראשונה עליהם.

מודלים ותוצאות שנבנו על-ידי מנחי הפרויקט:

לפני תחילת הפרויקט, נבנו מספר מודלים על-מנת לסווג את התמונות באמצעות רשתות דלילות, ולהשוות את התוצאות לרשת עמוקה:

- Fully-connected (4 שכבות)
- CNN (2-4 שכבות)
- רשת עמוקה – Inception V3

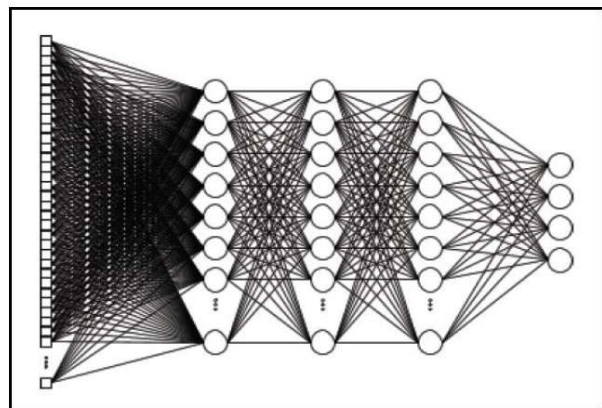
(Reference: <https://arxiv.org/pdf/1512.00567v3.pdf>)

Appendix A – Network configuration

Network architecture details:

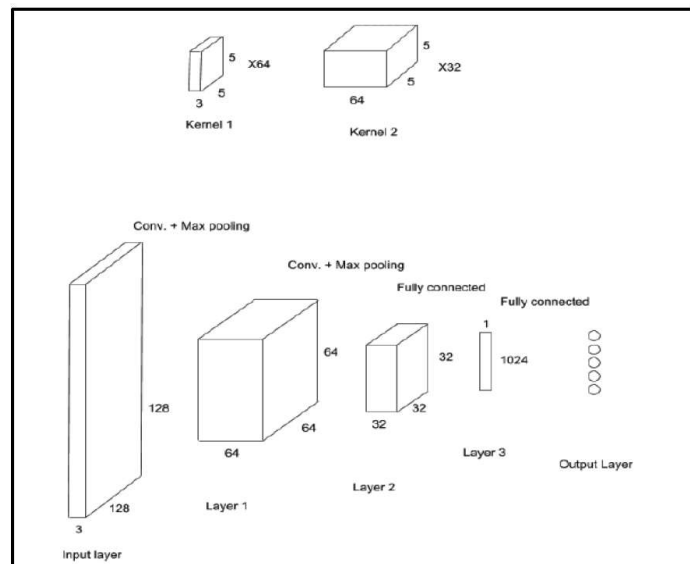
Fully connected neural network
4 Layers:

- Sigmoid [100]
- Tanh [100]
- Sigmoid [100]
- Sigmoid & [4] Softmax



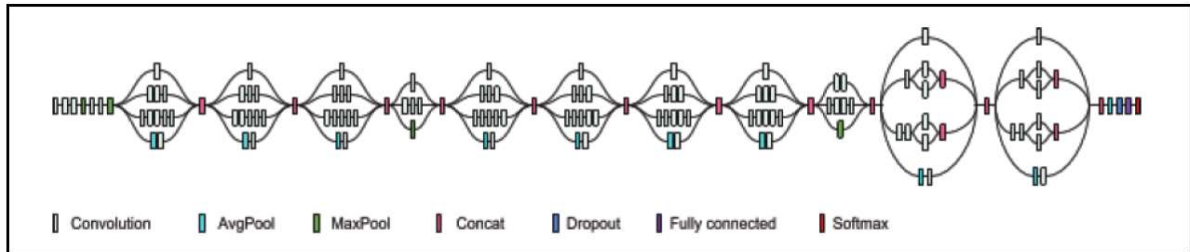
Convolution neural network

- 2 Convolutional layers:
 - 64 kernels of size 5x5x3
 - 32 kernels of size 5x5x64After each convolution layer there is stride of 2 max poolings
- Fully connected (with 50% dropout) from 32768 (32x32x32) to 1024 (ReLU activation function)
- Fully connected from 1024 to 5 (ReLU activation function)
- Softmax



Deep neural network

- Performed with retrain (transfer learning)
- Model: Inception V3 (48 layers)
- Originally trained on ImageNet



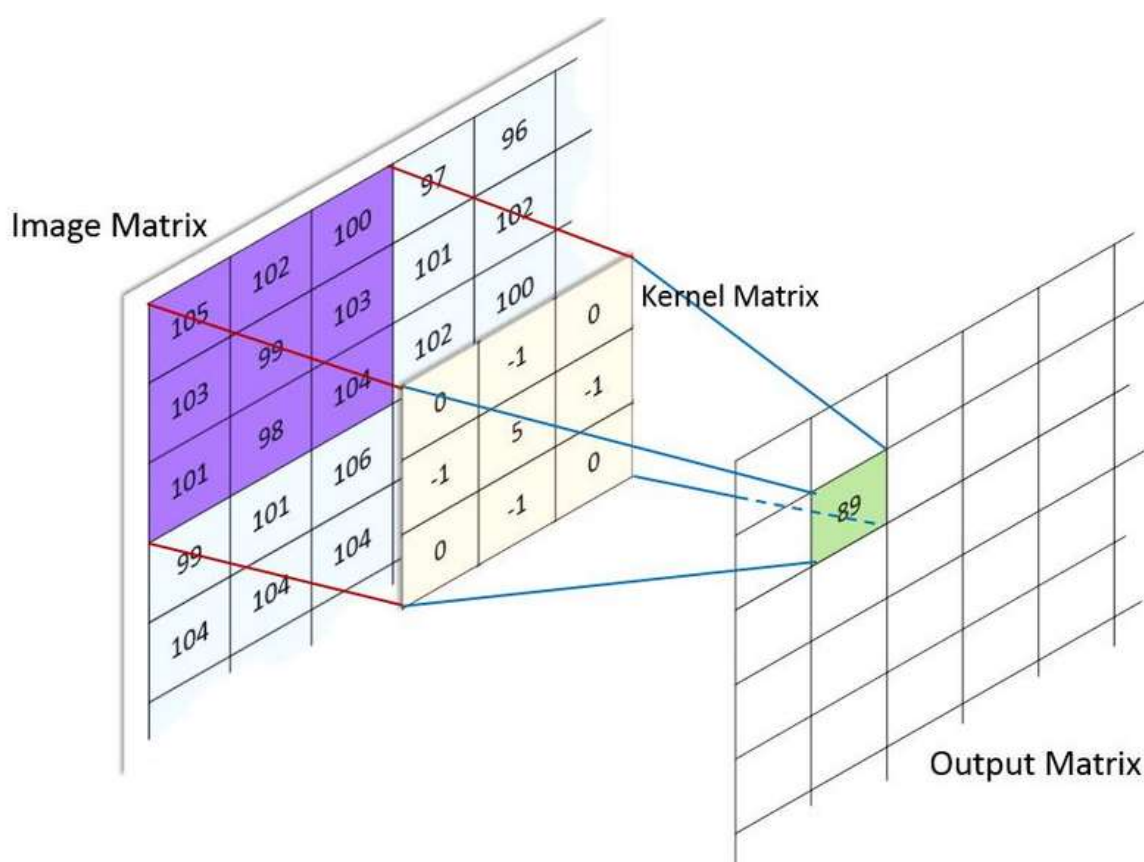
ביצועים:

Tomatoes				64	128	128 resized	256
Deep NN	Train	input	steps	100000	100000	100000	100000
		output	accuracy	95.70%	90.50%	89.10%	90.30%
			learning time	~2 hours	~8 hours	~8 hours	~4 hours
	Test	time per image	~7 hours	~7 hours	~7 hours	~7 hours	
CNN	Train	input	steps	60000	60000	60000	60000
		output	accuracy	95.44%	91.66%	90.30%	82.00%
			learning time	30 minutes	~9 hours	~9 hours	~30 hours
	Test	time per image	4-5 minutes	4-5 minutes	4-5 minutes	4-5 minutes	

בפרויקט זה מימשנו רשתות קונבולוציה. לכן, נספק הסבר קצר על פעולת הרשת.

רשתות קונבולוציה (ConvNets) בנויות להשתמש בצורה מינימאלית בעיבוד מקדים של התמונה. בנוסף, המבנה שלהן מושפע מתהליכים ביולוגיים, בהם חיבורים בין נוירונים מייצגים את הראיה.

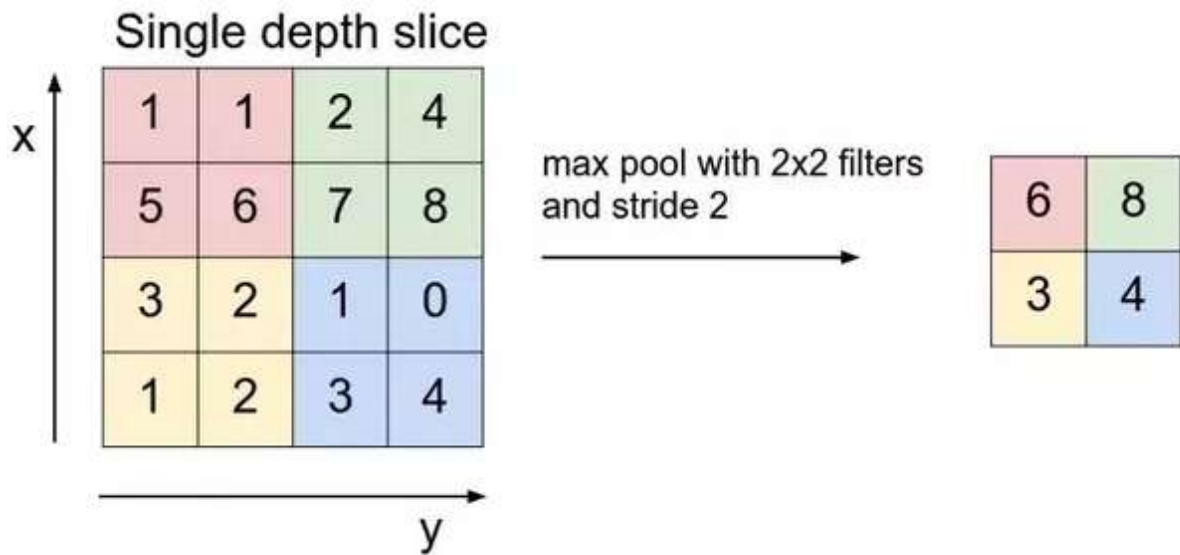
בזמן הלמידה, הרשת לומדת פרמטרים הנקראים "פילטרים" או "קרנלים". באלגוריתמים קלאסיים, פרמטרים אלה מהונדסים ידנית. היכולת ללמוד פרמטרים אלה מהווה יתרון גדול בכך שהיא מאפשרת לגלות פרמטרים אופטימליים ללא ידע מקדים.



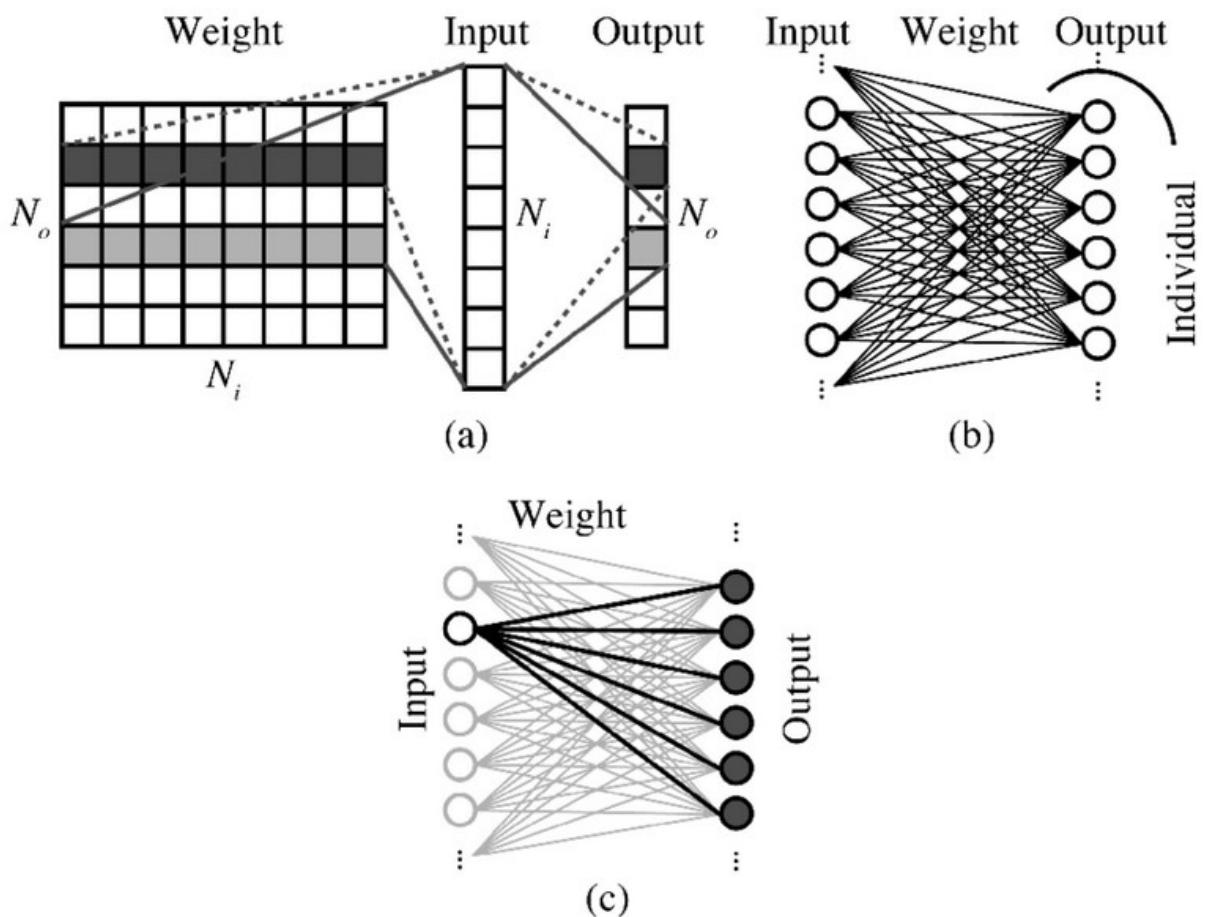
בין שכבות הקונבולוציה נמצאות שכבות נוספות המשמשות להורדת מימד:

- Max-Pooling
- Average-Pooling
- Etc...

הפעולה מתבססת על שילוב פלט מנוירונים שונים משכבה אחת כקלט לשכבה הבאה. לדוגמה, max-pooling משתמש בערך מקסימאלי בשכבה הקודמת. Average-pooling משתמש בממוצע על ערכי השכבה הקודמת.





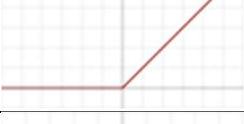
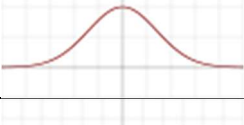
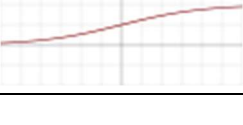
סוג נוסף של שכבה זה ה-fully-connected. בשכבה זו, כל פלט של נירון בשכבה אחת מחובר קקלט של נירון בשכבה הבאה. הפעולה ממומשת ע"י כפל של וקטור הפלט במטריצה.



Activation functions

בנוסף לשכבות המתוארות לעיל, ניתן להפעיל על פלטי השכבות פונקציות שונות. למעשה, תמיד תופעל פונקציית ברירת-מחדל שהיא פונקציית הזהות. עם זאת, רק פונקציות לא-ליניאריות מאפשרות לרשתות לפתור בעיות פחות טריוויאליות על-ידי שימוש במספר מועט של שכבות.

דוגמאות לפונקציות אקטיבציה:

המחשה	פונקציה
	$f(x) = x$
	$\begin{aligned} & \tanh \\ & f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \end{aligned}$
	$\begin{aligned} & \text{Rectified linear unit} \\ & f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \end{aligned}$
	$\begin{aligned} & \text{Gaussian} \\ & f(x) = e^{-x^2} \end{aligned}$
	$\begin{aligned} & \text{Sigmoid} \\ & f(x) = \frac{1}{1 + e^{-x}} \end{aligned}$

סביבת העבודה שבה נכתב הפרויקט היא Pycharm 2018 במערכת הפעלה Windows .10

שפת התכנות – Python 3.5

הספריות שבשימוש הן:

- OpenCV – עיבוד מקדים של התמונות לפני אימון ויצירת תמונות לאחר קבלת פלט מהרשת.
- Keras (with TensorFlow as backend) – מספקת High-level API המאפשר לבנות רשתות נוירונים בקלות
- Numpy – מאפשרת חישובים נומריים יעילים

DATASET

מאחר ולא קיים ground-truth, הגישה ההתחלתית היא סיווג של פאטצ'ים קטנים מתוך תמונה. לאחר מספר סימונים על תמונות, יצור כמות גדולה של פאטצ'ים מתבצעת יחסית בקלות. הפאטצ'ים נוצרים מתוך שרבוטים על התמונה.

ה-Dataset של העגבניות, המסופק ע"י "הזרע", מכיל בערך 4700 תמונות משדות גידול, ומכיל 7 זנים עם 34 תתי-זנים. בנוסף, קיימות גם תמונות שצולמו באמצעות מכשירים ניידים בשדות.

Tomato Species	Big Specialties	Pink	Plum	Canario Cluster	Cherry	Cherry Specialties	Roma
Tomato	24301	3628	48228	29636	46041	Honey Drop	3866
Sub-species	3714	3685	Luci Plus	Guarapo	46102	Solana	3868
	ALY 180	3691	Lucienne	Marchante	Kaori	Summer Sun	3869
	Maggie Star	HTP	Tyrex	Olympicus	Senalda		3872
	Pink Roma 61	11	Whitney	Orpheus	Shiren		
	Pink Roma 66			Sonia	Kaucana		

Table 1 – Tomato species and sub-species used for classification and segmentation

מכיוון שקשה לסמן bounding-boxes בתמונות כאלה (גבעולים דקים, פירות קטנים), יצירת הפאטצ'ים מתבצעת ע"י שרבוטים על התמונות, ולאחר מכן נוצרים פאטצ'ים במרווחים קבועים סביב הקווים. תהליך היצירה ניתן להמחשה בתרשימים 3 ו-4.

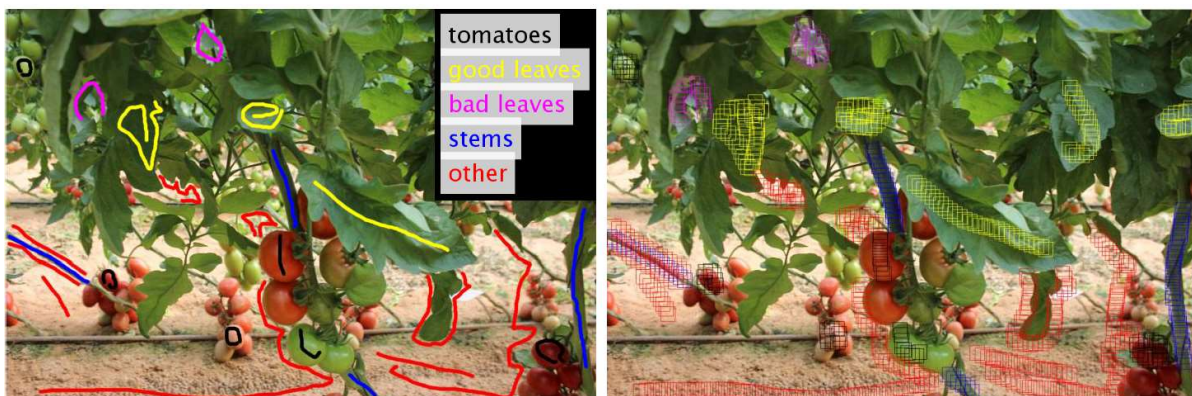


Figure 3 - Manual annotations (scribbles) for tomato part classification. Left – original scribbles, right – derived patches.



Figure 4 - Examples of patches for tomato part classification.

מודלים ראשוניים

במסגרת ה-Patches based neural networks מימשנו 2 סוגים של מודלים. המודל הראשון הוא מודל קלאסי, שבו מתבצע סיווג בין כל המחלקות המוגדרות בסיווג, והמודל השני הוא "2-round".

הרשת הראשונה מבוססת על מבנה שכבות מיוחד שנמצא בשימוש במודל מפורסם ששמו MobileNet¹. זה מודל יעיל שמטרתו לרוץ על מכשירים ניידים ו-embedded. לשם כך, המודל משתמש ב-Depth-wise separable convolutions² (DSC).

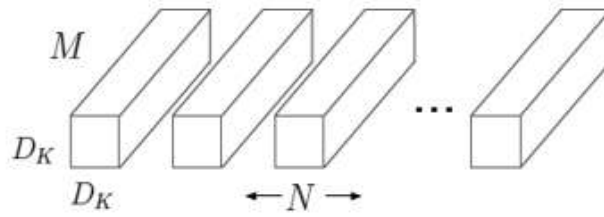
DSC מפריד את שכבת הקונבולוציה ל-2:

- Depth-wise convolution – משתמשת בקרנל אחד בלבד לכל שכבה בקלט.
- Point-wise convolution – משתמשת בקרנל בגודל 1x1 שמשלב את התוצאות של שכבת ה-Depth-wise.

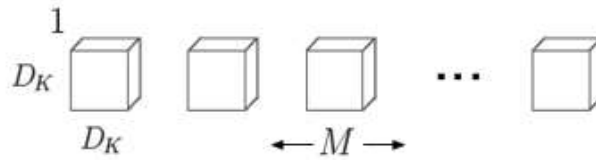
שימוש בשכבה זו במקום שכבת קונבולוציה רגילה מורידה את מספר הפרמטרים הנלמדים ואת סיבוכיות זמן הריצה עם פגיעה מזערית בדיוק הרשת. כתוצאה מכך, משיגים רשת מהירה יותר לעומת רשתות קונבולוציה רגילות.

¹ MobileNet - <https://arxiv.org/abs/1704.04861>

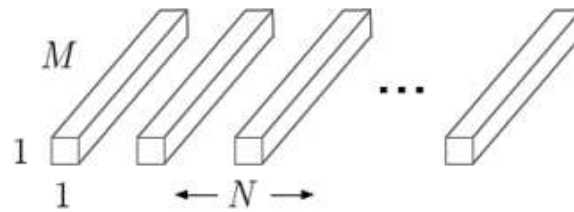
² DSC - <https://arxiv.org/pdf/1706.03059.pdf>



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figure 2. The standard convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c) to build a depthwise separable filter.

Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

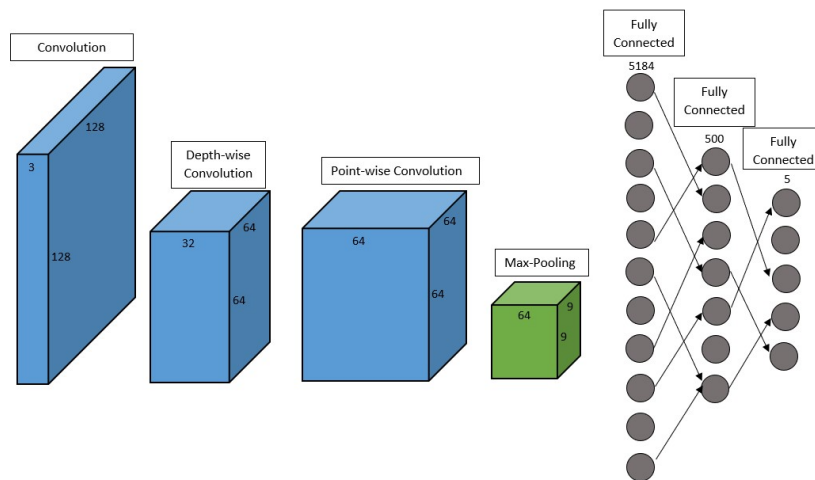
יצירת הרשת

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32 \text{ dw}$	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64 \text{ dw}$	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128 \text{ dw}$	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128 \text{ dw}$	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256 \text{ dw}$	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256 \text{ dw}$	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
Conv dw / s1	$3 \times 3 \times 512 \text{ dw}$	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512 \text{ dw}$	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024 \text{ dw}$	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×64	$1 \times 1 \times 64$
Sigmoid	In place	$1 \times 1 \times 64$
FC / s1	1024×162	$1 \times 1 \times 162$
Softmax	Classifier	$1 \times 1 \times 162$

MobileNet Architecture

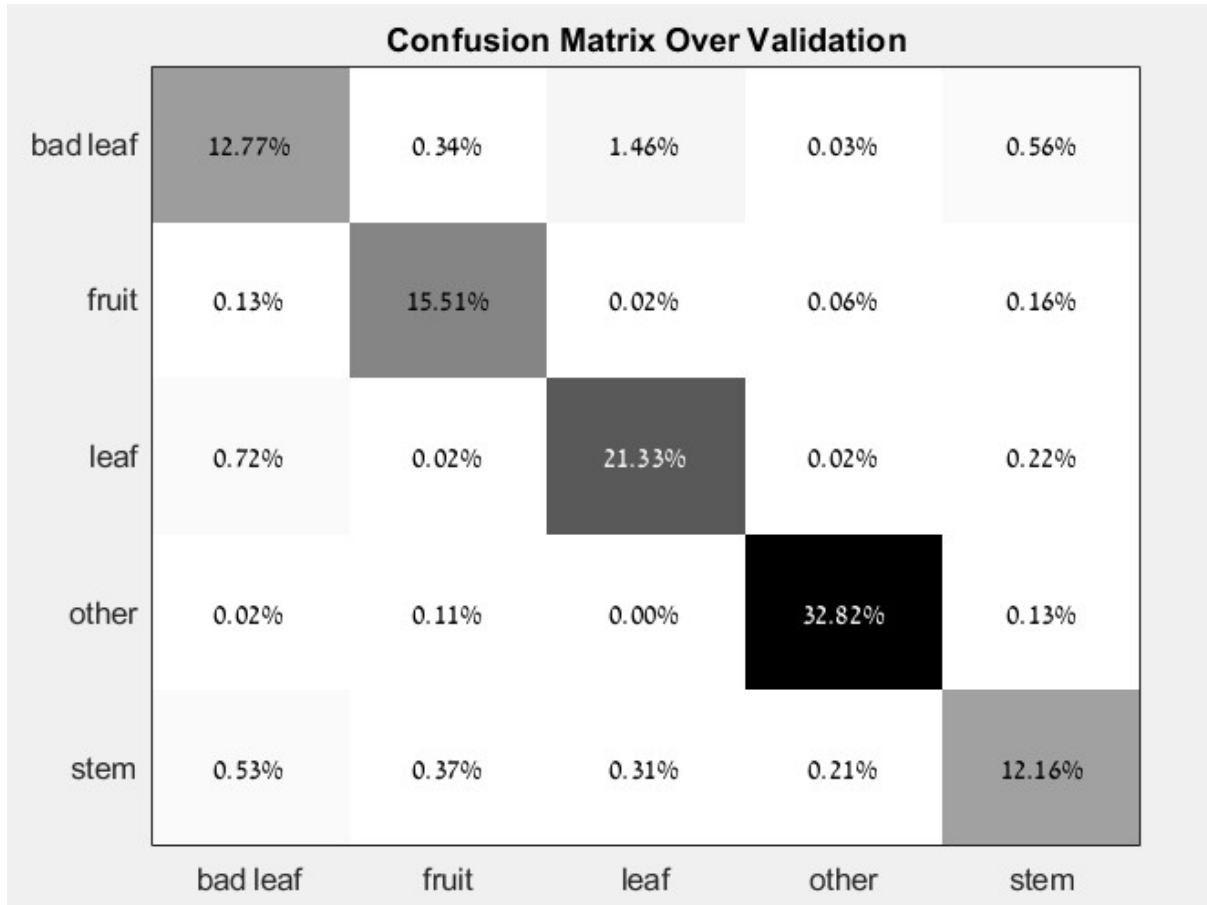
בתחילה ניסינו לאמן רשת Mobilenet מוכנה שאומנה על Dataset ImageNet. זו רשת עמוקה שמסוגלת לסווג בדיוק גבוה 1000 מחלקות. ביצענו לרשת Fine-tuning ע"י הסבת מספר הקטגוריות למספר הרצוי (5 במקרה של העגבניות) ואימון מחדש על ה-Dataset שלנו. מרשת זו קיבלנו אחוזי דיוק גבוהים מאוד – כמעט 99 אחוזי הצלחה בסיווג תמונות משדות העגבניות.

בהמשך, רצינו לייעל ולזרז את התהליך ובנינו רשת רדודה יותר, המתוארת בדיאגרמה הבאה:



2-round classification

בזמן ניסיונות האימון, הבחנו בכך שקיימות קטגוריות שביניהן המודל מתבלבל. האפשרות להבחנה זו מתאפשרת ע"י יצירה של confusion-matrix:



ניתן לשים לב שבאופן אידיאלי, המטרה היא לשאוף למטריצה אלכסונית. נתונים אלה הביאו אותנו למחשבה על מודל חדש שיסווג את התמונות ב-2 שלבים:

- סיווג לפי מחלקות שביניהם יש בלבול לבין מחלקות שאין ביניהן בלבול. בתמונה לעיל ניתן לראות שהמודל טועה ב-1.46% מהדגימות על "עלים", "עלים לא טובים" ו"אחר", לכן בשלב הראשון נסווג בין המחלקות:

Fruit ○

Stem ○

Other (consists of "leaf", "bad leaf", "other") ○

- סיווג מחדש של המחלקות הכלולות ב-Other של השלב הראשון ותיקון לפי הפלט החדש

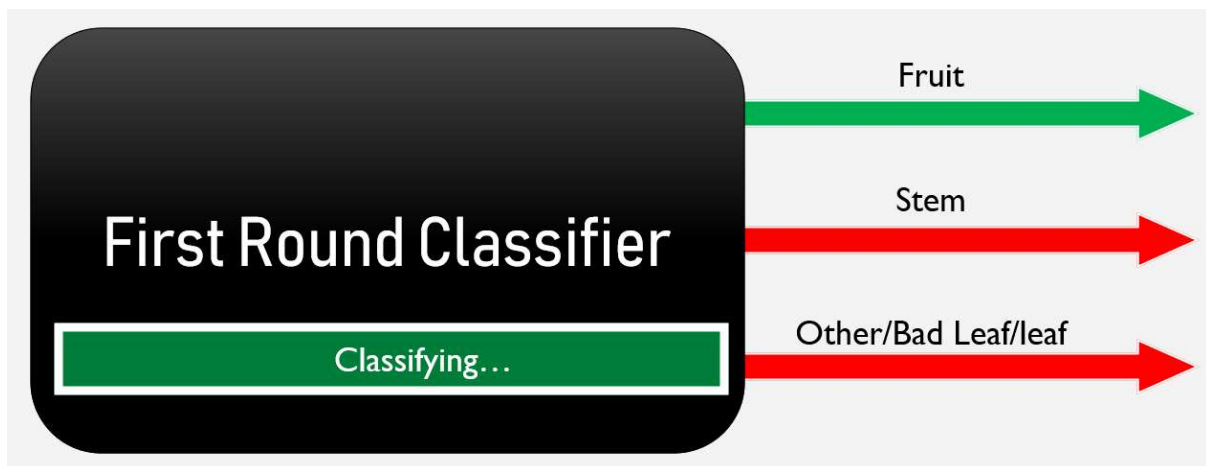
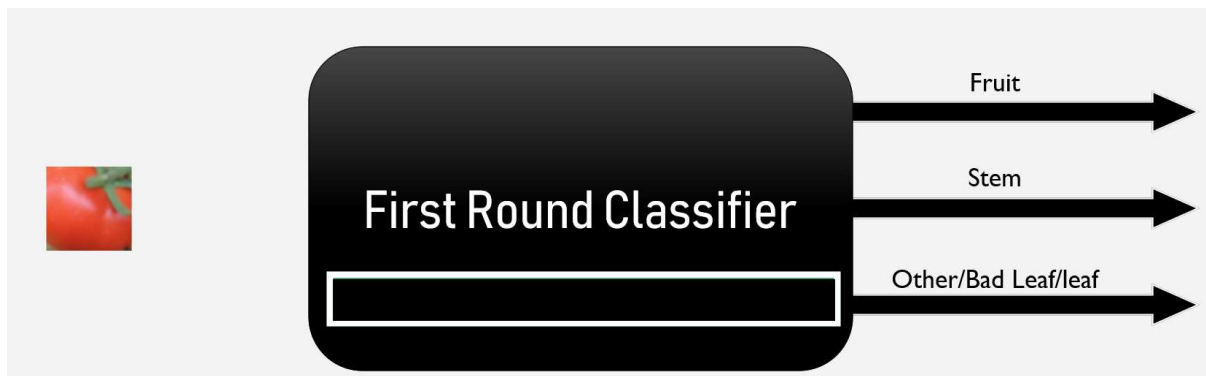
למודל זה מספר יתרונות:

1. זמן אימון קצר יותר – משתמשים בכלב שלב במודל קטן יותר מהמודל הראשון שהוצג. בכך קיצרנו את זמן האימון ולא זו בלבד אלא שניתן לאמן את 2 המודלים במקביל.

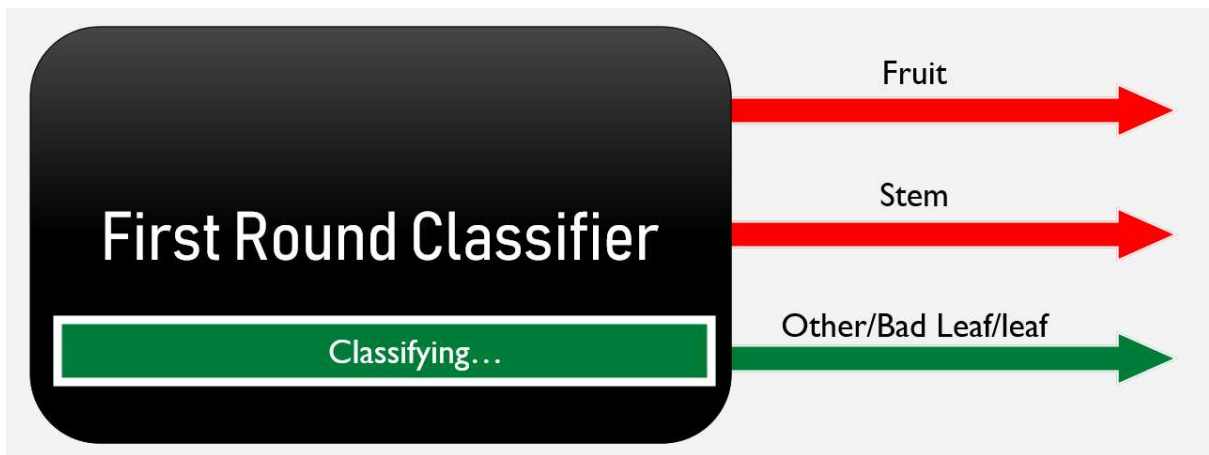
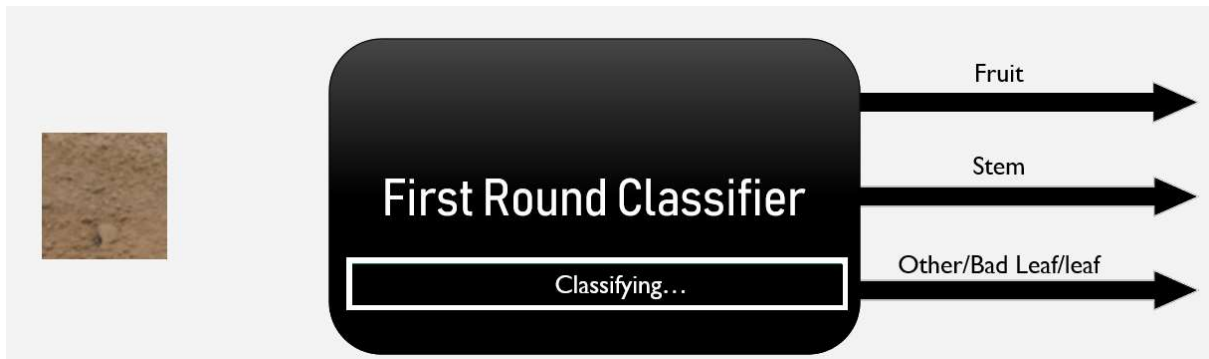
2. סיווג מהיר – השלב השני רץ רק על קלטים שסוגנו בשלב הראשון. בנוסף, ניתן להריץ את 2 שלבי הסיווג במקביל.
3. אחוזי דיוק גבוהים – לאחר אימון המודל הראשון, מייצרים confusion matrix על validation set, כך שיש באפשרותנו לאמן ולהפעיל את השלב השני במצבים שבהם יש בלבול.

המחשת 2-round

במקרה הראשון נרצה לסווג תמונה של עגבנייה. קיימת מחלקת fruit ולכן לא נצטרך להפעיל את שלב 2.



כעת, נרצה לסווג תמונה של אדמה ששייכת למחלקת other. כידוע, יש בלבול בין עלים ל-other ולכן נרצה להפעיל את שלב 2:



השפעה של טרנספורמציות צבע וגדלי התמונות על אחוזי הדיוק

במסגרת הניסיונות לשפר את אחוזי הדיוק, ביצענו מספר ניסיונות אימון של המודל.

בטבלה הבאה ניתן לראות את התוצאות:

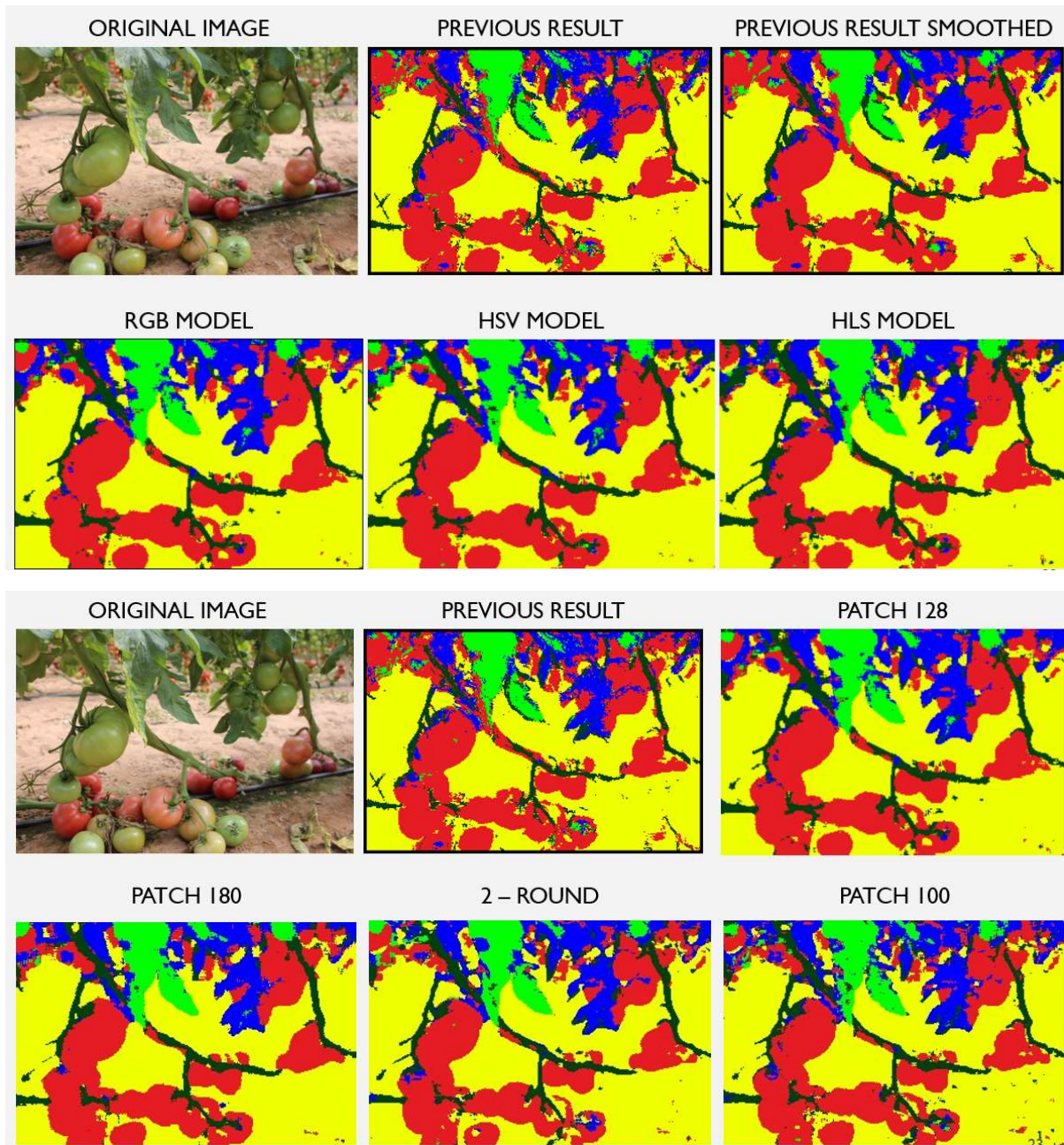
Color Space	Classifier Type	Training Time (Hours)	Validation Test Accuracy	Test Time Stride 13 (Minutes) [patches]	Test Time Stride 16 (Minutes) [patches]	Test Time Stride 20 (Minutes) [patches]
RGB	Patch 100	1:33	97.3%	3:05 [101528]	1:59 [66780]	1:18 [42840]
RGB	Patch 128	2:28	98.8%	3:11 [99584]	2:03 [65728]	1:18 [42251]
RGB	Patch 180	9:55	99.7%	5:34 [97020]	3:00 [64165]	1:25 [41164]
RGB	Patch 128 2 – round	2:44	97.4%	2:15 [99584]	1:48 [65728]	1:11 [42251]
HSV	Patch 128	2:25	98.3%	3:10 [99584]	2:03 [65728]	1:20 [42251]
HLS	Patch 128	2:17	97.6%	3:08 [99584]	2:06 [65728]	1:20 [42251]

מסקנות

- אימון עם פאט'צים גדולים יותר משיג אחוזי דיוק גדולים יותר. ההנחה שלנו היא שפאט'צים גדולים מכילים יותר מידע, ולכן יש יותר אינפורמציה שמאפשר לסווג בדיוק גבוה יותר.
- מרחבי צבע שונים לא הביאו לשיפור ניכר - מלבד קיצור זמן האימון ב-HLS, שהסיבה לכך יכולה להיות קשורה לאתחול של האימון שמתבצע בדרך-כלל באופן רנדומאלי.

בדיקה ויזואלית

להלן השוואה של תמונות כפליטים של הרשתות השונות. 2 התמונות העליונות (הימניות) הן פליטים של הרשתות של המנחים.

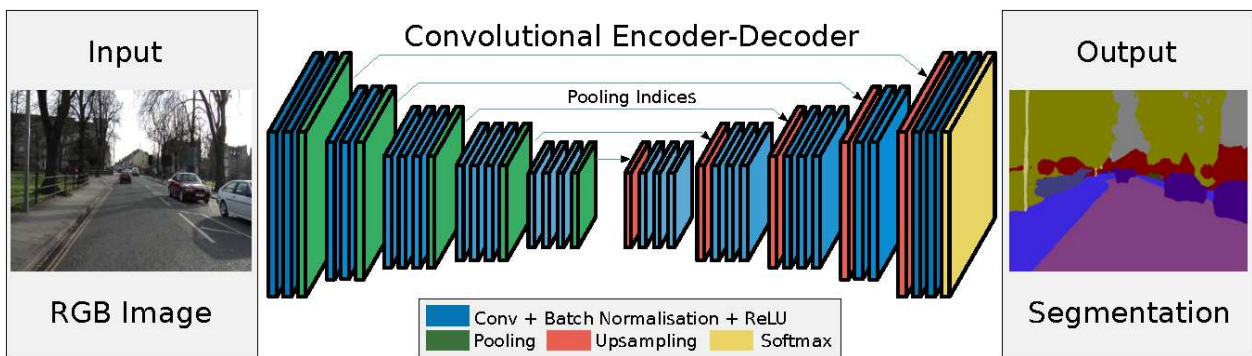


מסקנות:

- הצלחנו להשיג שיפור ביחס לתוצאות הקודמות. התמונות יותר חלקות (ללא עיבוד התמונה) וניתן בקלות לזהות צורות של חלקי צמחים.
- שימוש במרחבי צבע שונים לא הביא לשיפור משמעותי בסיווג למעט מקרים מסוימים. לדוגמא, הענף במרכז התמונה קיבל את ההפרדה הכי טובה במרחב HLS. למעט זאת, לא הבחנו בהבדלים משמעותיים.
- שימוש בגדלים שונים של פאט'צים השפיע באופן מכריע על התמונות. ניתן בבירור לראות שעם גודל 100x100 יש הרבה רעש בתמונה. נציין גם כי ניתן לתקן רעשים אלה על-ידי אלגוריתמים שונים. בנוסף, עם גודל 180x180 הבחנו בצורות פחות ברורות של חלקי צמחים – בלבול בין גבעולים ופירות, עגבניות לא עגולות ועוד. נסכם בכך שהגודל 128x128 סיפק לנו את התוצאות הכי טובות מרוב הבחינות – זמן אימון, אחוזי דיוק ותוצאות ויזואליות.

ENCODER DECODER

זוהי רשת שמורכבת משכבות קונבולוציה, המבנה שלה הוא קרנלים ברזולוציה שקטנה עד לנקודה מסוימת וממנה הרזולוציה גדלה. היא משמשת בעיקר לצורכי הפרדה של חלקים בתמונה כך שהשכבה האחרונה שלה בדרך כלל באותה רזולוציה כמו של השכבה הראשונה אך עם עומק שונה שנקבע לפי כמות המחלקות אותן נרצה להפריד. לדוגמא אם יש 5 מחלקות שונות ביניהן נרצה להפריד נשתמש בעומק של 5 כך שכל רכיב אומר כמה המחלקה מתאימה לפיקסל שסיווגנו. בדרך כלל מה שעושים כדי לאמן רשת כזו זה לקחת תמונה ואת התמונה לאחר הסיווג (שמבוצע באופן ידני) ולהעביר את התמונה המקורית ברשת ולחשב loss לפי ground truth של התמונה לאחר סיווג.



AUTO ENCODER

Auto-Encoder זוהי רשת עם ארכיטקטורה דומה לזו של Encoder-Decoder. ההבדל הוא שכדי לאמן את הרשת התמונות שנכנסות לרשת הן ground truth. המטרה באימון של הרשת זה שהיא תצליח לקודד בשכבה האמצעית הקטנה דברים על השכבה המקורית כדי שהDecoder יצליח להוציא ממנה את התמונה המקורית חזרה. משתמשים ברשת הזאת כדי להוציא בסוף השכבה של encoder כל מיני מאפיינים ותכונות של התמונה שיהיה אפשר להשתמש בהם הלאה.

מה המטרה שלנו הייתה בחלק השני של הפרויקט?

לאחר שהצלחנו להגיע לאחוזי זיהוי גבוהים מאוד בחלק הראשון של הפרויקט עם mobilenet החלטנו שאנו רוצים ללכת לכיוון אחר שאינו קשור רק באיכות של התמונה שיוצאת.

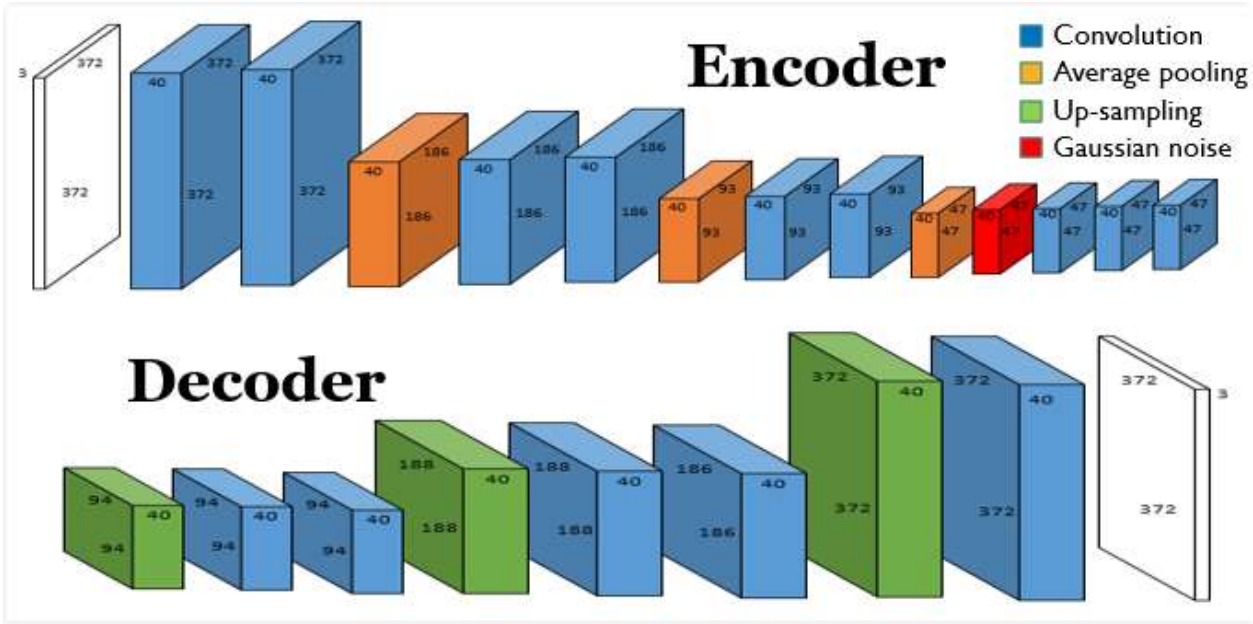
כיוון שהפרדה של תמונה לוקחת כמה דקות חיפשנו דרך להאיץ את התהליך וכדי להגיע למצב שב real time נוכל להפריד תמונה ושנוכל להריץ את הרשת על סרט בזמן סביר. לפי מה ששמענו מההרצאות של Stanford³ ונתקלנו בו באינטרנט משתמשים ב-Encoder Decoder כדי ליצור הפרדה של תמונות בזמן יעיל ולכן המטרה שלנו הייתה לאמן רשת של Encoder-Decoder כדי להפריד את התמונות.

איך הגענו למודל הסופי שלנו?

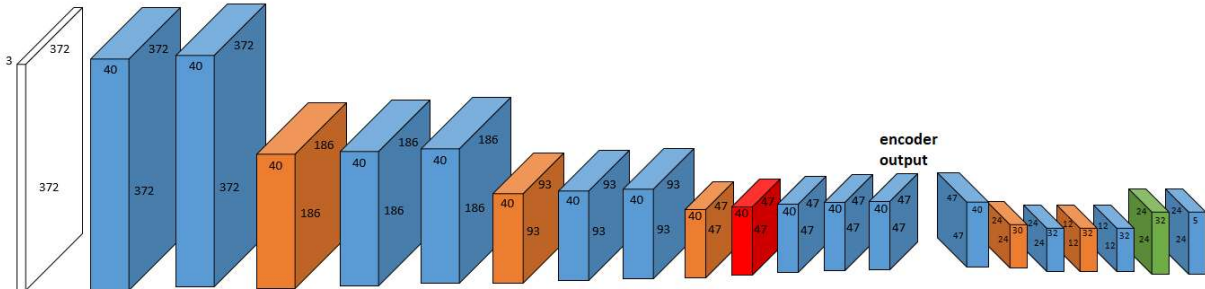
לאחר ששימוש במודלים שונים של Encoder-Decoder נכשלו (יצרו תמונות עם הפרדה שהייתה ממש גרועה בעין עבור התמונות ושהייתה עם loss מאוד גבוה באימונים) חשבנו על שימוש ב-Encoder מתוך Auto-Encoder כדי שיוציא כל מיני תכונות ומאפיינים שיעזרו לרשת של Encoder-decoder שלנו ללמוד יותר טוב איך לסווג את התמונה.

³ Stanford lectures - <https://youtu.be/ByjaPdWXXKJ4>

המבנה של Auto-Encoder:



המבנה של המודל הסופי שלנו הוא Encoder מתוך Auto-Encoder ורשת של Encoder-Decoder שמקבלת את הoutput של Encoder.



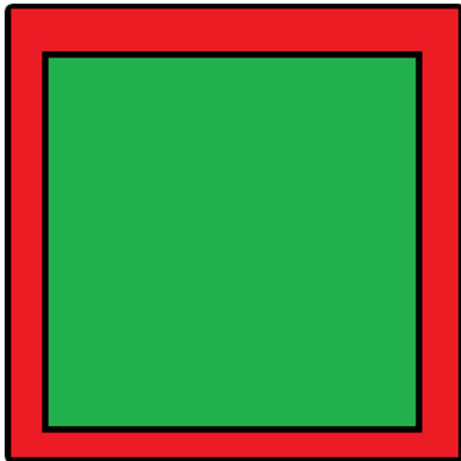
איך אימנו את המודל שלנו?

לאימון היו כמה שלבים:

1. יצירת הground truth עבור כל המודל שלנו.

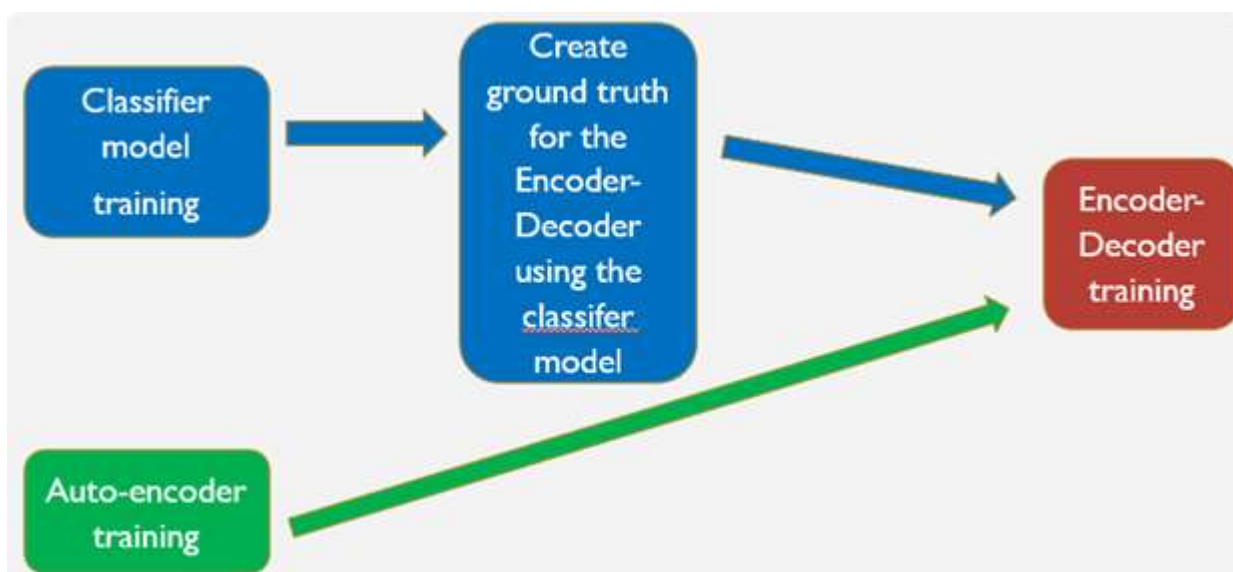
הרעיון הוא ליצור סיווגים עבור כל פיקסל אמצעי בpatch בגודל 128 על 128 שנחתך מתוך patch בגודל של 500 על 500 (500 נבחר שרירותית ו128 על 128 זה גודל הכניסה של mobilenet) כמו שעושים סיווג לתמונה בגודל המקורי מהחלק הראשון של הפרויקט. אז יצרנו חלקי תמונה בגודל של 500 על 500 והרצנו את mobilenet על כל אחד עם קפיצות בגודל של 16 (כיוון שקפיצות בגודל אחד היו יוצרות mobilenet מלא תמונות שצריך לסווג מה שהיה לוקח לנו כמות זמן לא הגיונית) ולכן גודל התמונה שנוצרה לאחר ההפרדה היא 24 על 24 $(\frac{500 \cdot 64}{16})$.

2. אימון Auto-Encoder: כדי להתחיל בכלל את השלב הזה נדרשנו ליצור מתוך התמונות פטצ'ים בגודל של 372×372 (כיוון שהמבנה של mobilenet לא מתחילה לקחת את הסיווג שלה מנקודה בצד של התמונה אלא מנקודה שבמרחק של 64 מהקצה ומסיימת בנקודה שהמרחק שלה מהקצה הוא גם 64) אותם יצרנו מחיתוך של 64 פיקסלים מהקצוות של התמונות של 500 על 500.



התמונה המקורית זה כל מה שהיה במסגרת השחורה החיצונית והחלק האדום זה מה שירד מהתמונה והחלק הירוק זה מה שנשאר מהתמונה. וכדי לאמן את ה auto-encoder מה שעשינו זה לקחת את התמונות שנוצרו בגודל של 372 על 372 ולהשתמש בהם גם כקלט לרשת וגם כ ground truth כמו שמאמנים רשת של Auto-Encoder.

3. אימון המודל הסופי: לאחר שסיימנו את שני השלבים הראשונים אפשר להתחיל באימון של המודל הסופי בכך שניקח את ה Encoder מתוך ה Auto-Encoder בתור חלק שאינו ניתן לאימון עבור הרשת הסופית ו"נדביק" אליו את ה Encoder-Decoder והקלט שייכנס למודל זה התמונות בגודל של 372×372 על 372×372 שיצרנו בשלב השני וה ground truth יהיה התוצאות שיצאו מהשלב הראשון (אנו משתמשים בתוצאות של השלב הראשון בתור ground truth כיוון שאין בידנו סיווג שאדם עשה עבור הקלט שלנו ואחד הרעיונות המרכזיים בפרויקט זה איך רק מקלסיפיקציה שנוצרת מסיווג חלקי של תמונות לקבל הפרדה).

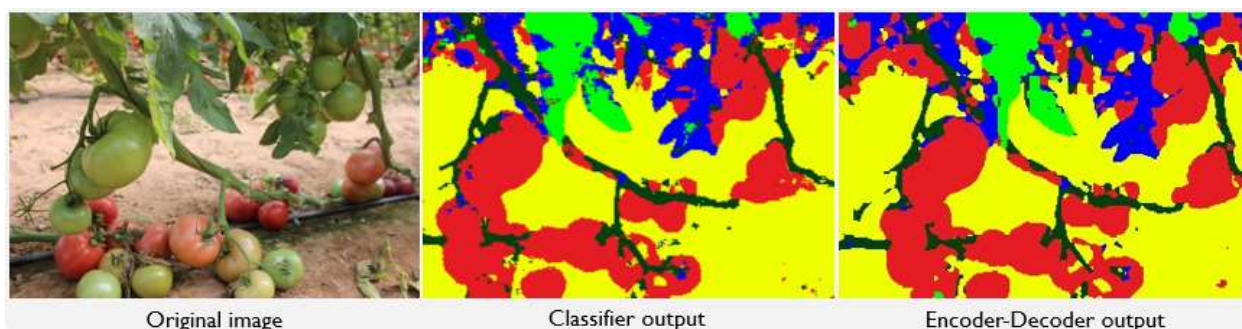


תוצאות

הזמן שלקח למobilenet לעשות הפרדה של תמונה (כך שהיא מבצעת קפיצות של 16 פיקסלים כל פעם ומתחילה מהפיקסל ה-64 ומסיימת בפיקסל ה-64 מהסוף) הוא 2 דקות. לעומת זאת למודל הסופי שבנינו לקח זמן שקטן משניה אחת כלומר הצלחנו עם המודל הסופי להגיע לשיפור של **יותר מפי 120** בזמן מאשר המobilenet. לצורך השוואה לקח לנו להריץ את המודל הסופי על 4700 תמונות חצי שעה ולפי החישובים שלנו יקח למobilenet $4700 \cdot 2 \text{ minutes} = 6.52 \text{ days}$.

הencoder-decoder אינו פולט משהו שניתן להגיד את הדיוק שלו כיוון שאין לנו ground truth רציני (אין לנו הפרדה מלאה של תמונות) ולכן לא השתמשנו בaccuracy ואין לנו דרך להשוות את אחוזי הדיוק של המודל החדש יחסית למודל של mobilenet פרט להשוואה של איך התמונות שיוצאות מהפרדה של כל אחד מהם.

לאחר הסתכלות על כמה תמונות נראה שהתוצאה שיוצאת מהmobilenet היא תמונה פחות חלקה מאשר זו שיוצאת מהEncoder-Decoder שעשינו בסוף.



השוואה בין הפלטים של שני המודלים

מלפפונים ואוטומציה

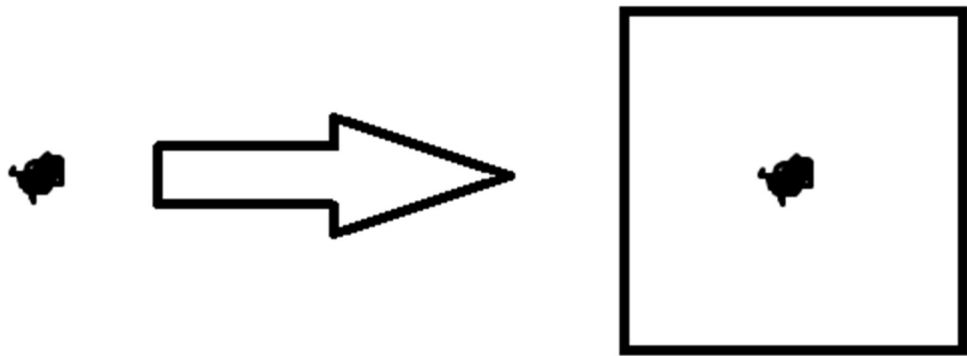
לאחר ההצלחה בשיפור בזמנים של המודל האחרון שיצרנו אלון פנה אלינו בבקשה שנפעיל את המודל שלנו גם כדי ללמוד איך לסווג מלפפונים.

אוטומציה

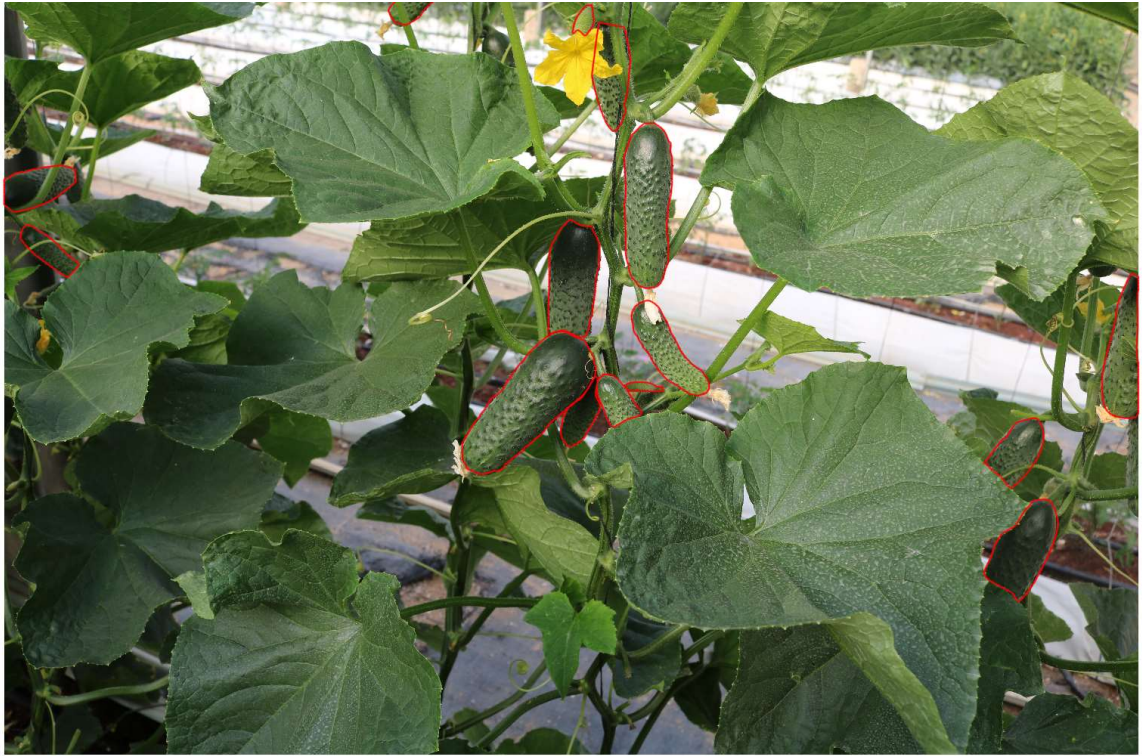
הדרישה שאלון הביא לנו גרמה לנו להחליט שאנו רוצים לעשות את כל תהליך האימון אוטומטית, כלומר שיצירת הפטצ'ים מכל סוגי הגדלים תקרא לבד וכך גם יצירת התמונות והעברה שלהן למודל הסופי בשביל האימון וגם הדבקת המודלים.

איך יוצרו הפטצ'ים?

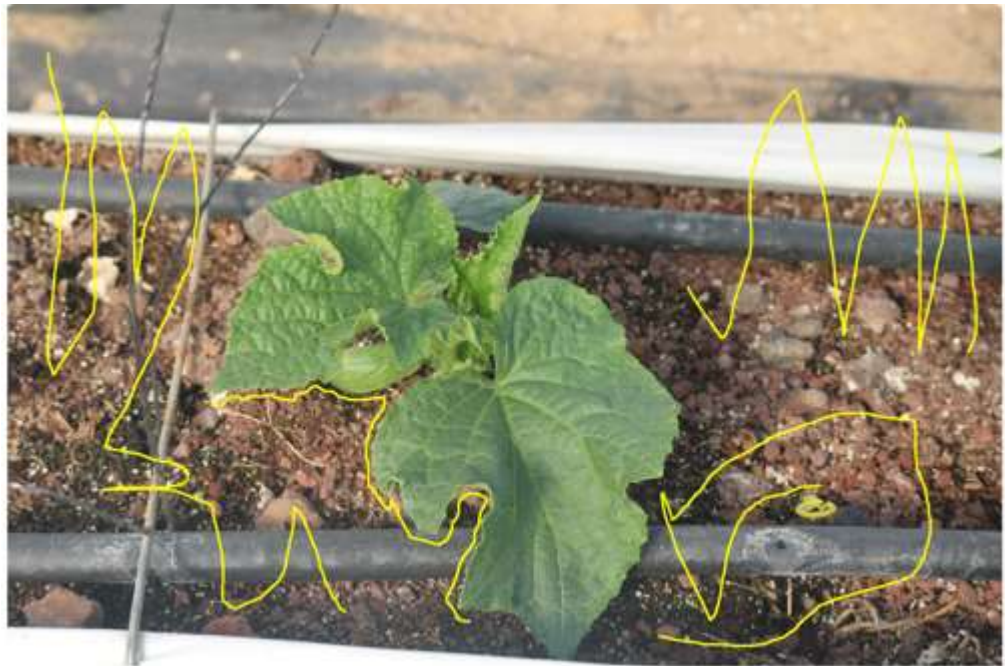
עבור המלפפונים הפטצ'ים יוצרו באופן שונה מאיך שיוצרו הפטצ'ים של העגבניות כיוון שאלון קיבל סוג שונה של תמונות מסווגות בהם סימנו קווי מתאר רק מסביב לאובייקט שרצו להפריד ולא על כל התמונה. כדי להתאים את הסוג השונה הזה של הסימונים אל הדרישות שלנו אלון השתמש באלגוריתם של מטלאב שבוחר נקודות באקראי בתוך הפוליגון שסומן ולוקח פטצ'ים בגודל של 128 מסביבם באופן הבא:



הסימונים שהוא קיבל לא כללו את הקטגוריה של other ולכן הוא סימן באופן ידני באותו אופן כמו שהוא עשה עבור העגבניות את הקטגוריה הזאת.



תמונה שמציגה סימונים שאלון קיבל עבור המלפפונים לדוגמא.



תמונה של קווי מתאר שאלון צייר עבור הקטגוריה other.



תוצאות שיצאו לנו מסיווג שהמודל שלנו עשה. אפשר לראות שתוצאות הסיווג אינו טובות ואנו משערים שהבעיה מכך שהסימונים שאלון קיבל לא היו מספיק טובים בשביל שנוכל לאמן את הרשת שלנו עליהם.

REFERENCES

- <https://arxiv.org/abs/1704.04861> – Mobilenet .1
- <https://arxiv.org/pdf/1706.03059.pdf> - Depthwise seperable convolution (DSC) .2
- <https://youtu.be/T7o3xvJLuHk> - DSC video .3
- <https://youtu.be/NfnWJUyUJYU> - Stanford DL course videos (CS231n) .4
- [/https://www.tensorflow.org/tutorials](https://www.tensorflow.org/tutorials) - MNIST tutorial for ML begginers .5
- <https://youtu.be/RznKVRTFkBY> - Keras training .6