**Assignment 3**

* שיר כהן, 315805168
* עמית שקרצ'י, 313278889

# Data

*ה-*data *נמצא* [*בקישור*](https://www.robots.ox.ac.uk/~vgg/data/flowers/102/)*.*

*ה-*data *מכיל 8189 קבצי תמונות של פרחים הנפוצים בבריטניה. קיימים 102 סיווגים שונים של סוגי פרחים.*

## SPLIT TRAIN, VALIDATION TEST

*השתמשנו ב-data set שניתן על ידי ה-*api *של* TensorFlow*. על מנת לבצע חלוקה של הנתונים בצורה שונה הגדרנו מיקום אחר של נקודת הפיצול לשלושת הקבוצות: train, test ו- validation.*

|  |  |  |
| --- | --- | --- |
| **Percentage of data** | **Number instance** |  |
| *50%* | *4094* | **Train** |
| *25%* | *2047* | **Validation** |
| *25%* | *2048* | **Test** |

## Preprocessing

*תהליך ה-*preprocessing *לתמונות הוגדר בצורה הבאה:*

1. ***Resize*** *– שינוי גודל התמונה לגודל קבוע על מנת להתאים את הגודל לארכיטקטורה. הגודל הוגדר כ- (224, 224).*
2. ***נרמול (אופציונאלי)-*** *ביצענו נרמול על ידי חלוקה ב- 255 על מנת לייצר תמונה המכילה טווח ערכים של [0,1]. מטרת התהליך הנ"ל הוא לאפשר לרשת להתכנס בצורה מהירה יותר עם ערכים מנורמלים. על מנת לקבוע האם לנרמל את התמונות או לא, ביצענו ניסויים בהם בחנו האם הנרמול עוזר לתהליך הלמידה או לא על סמך ביצועי המודלים.*

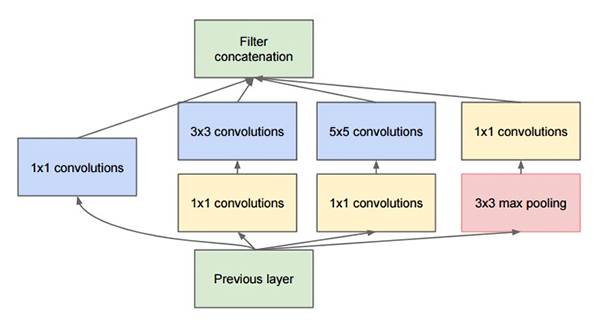
*בנוסף, ניסינו לבצע חיתוך של התמונות הקיימות סביב מרכז התמונה, על מנת למקד את התמונה אל הפרח עצמו ולא אל העלים שמסביבו. חיתוך התמונות גרם לביצועים ירודים יותר של המודלים שבחנו, ולכן בחרנו שלא להשתמש בו.*

# MODELS

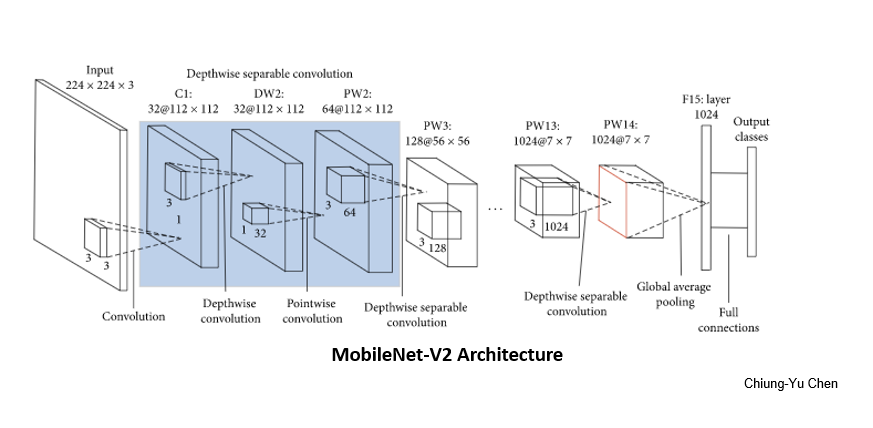
*Transfer Learning- התמקדות בשימור הידע הנצבר בפיתרון של בעיה אחת, ויישומו לפתרון בעיה אחרת הקשורה אליה. במקרה שלנו, מדובר בשימוש בארכיטקטורות קיימות של רשתות נוירונים שאומנו על סיווג של תמונות לקטגוריות- לצורך סיווג תמונות של פרחים ל-102 קטגוריות שונות.*

*ביצענו השוואה בין שתי ארכיטקטורות שונות:*

1. ***inceptionv3****– ארכיטקטורת CNN שפורסמה במאמר "* *Rethinking the Inception Architecture for Computer Vision". המימוש נלקח מהקישור* [*הבא*](https://www.tensorflow.org/api_docs/python/tf/keras/applications/inception_v3)*, כחלק מהארכיטקטורות המוכנות המוצעות ב-Keras. הארכיטקטורה אומנה על ImageNet dataset, עם 1,000 קטגוריות סיווג שונות. . המטרה של המודל היא להוות feature extractor על ידי חישוב של קונבולציות 1x1, 3x3, 5x5. הוספנו שכבת batch norm בתחילת הארכיטקטורה ושכבת GlobalAveragePooling2D לאחר המודל.*



1. ***MobileNet V2*** *– המימוש נלקח מהקישור* [*הבא*](https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4)*. המימוש מתבסס על המאמר "* "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation","*. הארכיטקטורה אומנה על ImageNet dataset, עם 1,000 קטגוריות סיווג שונות. הורדנו את השכבה האחרונה של הארכיטקטורה.*



*עבור כל ארכיטקטורה הוספנו שלוש שכבות נוספות:*

1. ***Dense*** *– עם 128 נויורנים.*
2. ***Dropout*** *– עם rate משנה לפי ניסויים.*
3. ***Dense******– output layer*** *– עם מספר נוירונים של 102 (מספר ה-classes) ואקטיבציה של 'softmax'.*

פרמטרים קבועים –

|  |  |
| --- | --- |
| **Value** | **Parameters** |
| 32 | Batch size |
| 'sparse\_categorical\_crossentropy' | Loss function |
| 10 | Epochs |
| After 2 epoch val loss doesn’t change | Early stopping |

# EXPERIMENTS

תחילה, חשוב לנו להדגיש כי **לא היו לנו משאבים** על מנת להריץ כמות ניסויים מספקת. כתוצאה מכך הרצנו את המודלים עבור 10 epochs בלבד.

השתמשנו ב- [tensorflow\_hub](https://www.tensorflow.org/hub) על מנת להגדיר ארכיטקטורות רשת שונות. הערכים –

1. **Noramlization** – האם לבצע נרמול של ה-data לטווח בין 0 ל-1.
2. **Dropout** – rate של 0 או 0.3.
3. **Optimizer** – adam או RMSprop.

## Normalization

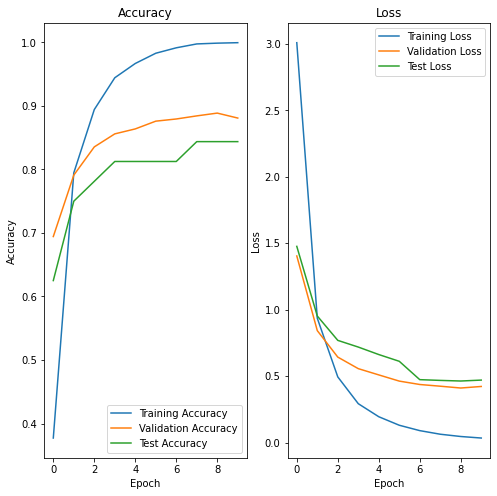
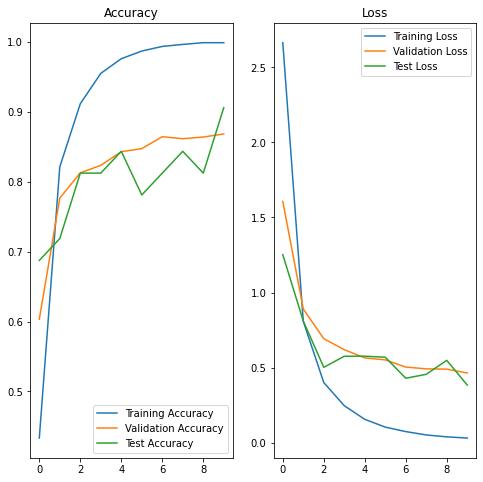
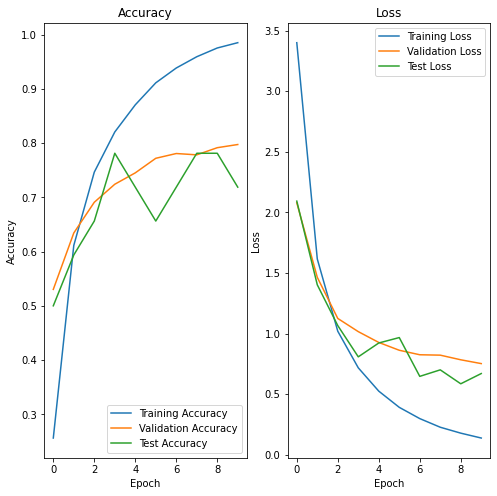
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Number run** | **Model** | **Normalization** | **Number unit dense** | **dropout\_rate** | **Way split** | **Accuracy Test** |
| **1** | mobilenet | T | 128 | 0 | 1 | 0.889 |
| **2** | inceptionv3 | T | 128 | 0 | 1 | 0.780 |
| **3** | mobilenet | T | 128 | 0 | 2 | 0.865 |
| **4** | inceptionv3 | T | 128 | 0 | 2 | 0.734 |
| **5** | mobilenet | F | 128 | 0 | 1 | 0.389 |
| **6** | inceptionv3 | F | 128 | 0 | 1 | 0.495 |
| **7** | mobilenet | F | 128 | 0 | 2 | 0.34 |
| **8** | inceptionv3 | F | 128 | 0 | 2 | 0.61 |

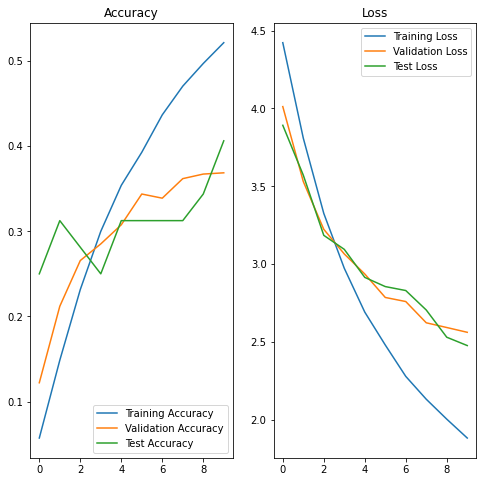
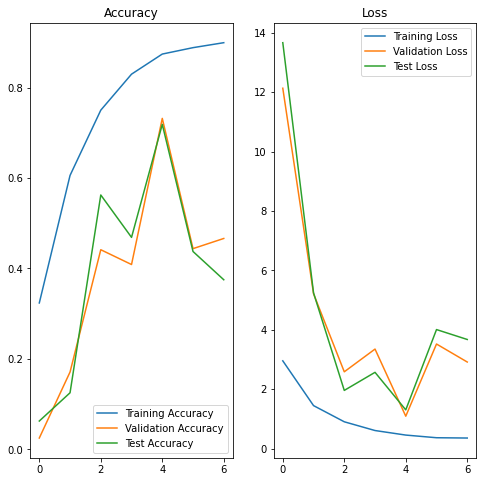
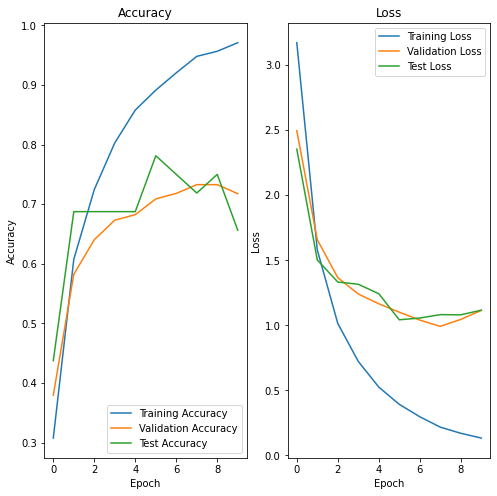
בהרצה הראשונה רצינו לבדוק את ההשפעה של תהליך ה-preprocessing שמבצע נרמול של התמונה לטווח ערכים בין 0 ל-1. מאחר ורשתות נוירונים פועלות על הפיקסלים ההיגיון מאחורי ביצוע הנרמול שהוא יאפשר לרשת **להתכנס בצורה מהירה** יותר. הסיבה לכך, שפיקסלים עם ערכים מאוד גבוהים לא ישפיעו בצורה "משמעותית" על הרשת.

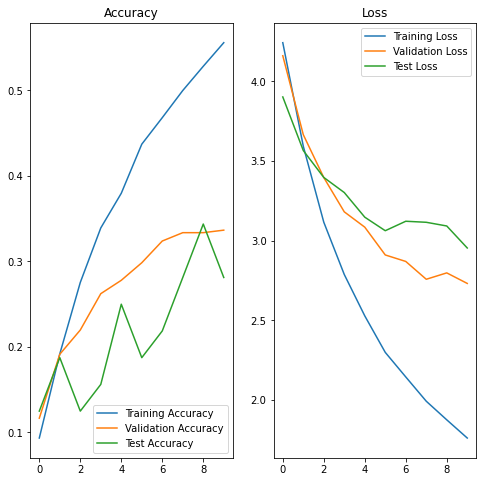
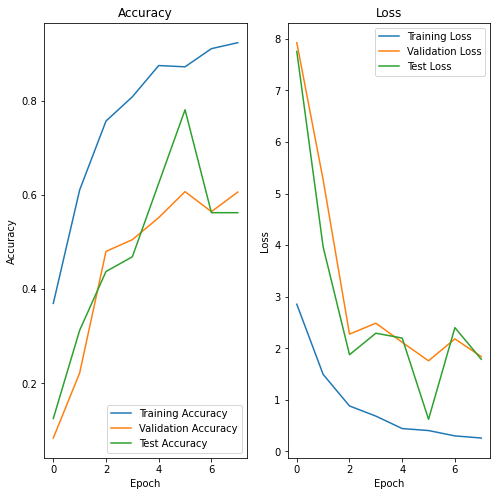
מסקנות מהטבלה–

* **השפעת הנרמול** – ניתן לראות כי בכל הניסויים (הן במודלים השונים והן בפיצול ה-train, validation test בצורה שונה) ה-accuracy על ה-test set גבוהות יותר כאשר ביצענו נרמול. כלומר, פעולת הנרמול כחלק מתהליך ה-preprocessing תורמת ללמידה של המודל.
* **השפעת ה-split –** ניתן לראות כי החלוקות השונות של ה-data אינן משפיעות בצורה רבה על ה-accuracy על ה-test. דבר זה מעיד כי הרשת עמידה לחלוקות שונות **והתוצאות אמינות** יותר.
* **מודלים –**ניתן לראות כי המודל עם הארכיטקטורה של mobilenet בעל ביצועים טובים יותר מהארכיטקטורה של inceptionv3.

גרפים מסודרים משמאל לימין לפי סדר הניסויים –







מסקנות מהגרפים–

* **עצירה מוקדמת –** מרבית הריצות סיימנו לבצע את 10 ה-epoch שהוגדרו.
* **Loss –** בכל הגרפים ניתן לראות מגמת ירידה של ה-loss. כמו כן, ה-loss על ה-train נמוך יותר מה-loss על ה-test וה-validation כמצופה.
* **Accuracy –** קיימת מגמת עלייה של ה-accuracy על ה-train בכל הגרפים. דבר זה מעיד כי מתקיימת למידה על ה-train set. בנוסף, במרבית הגרפים ה-train מגיע ל-accuracy קרוב ל-1 דבר המעיד על overfitting מסוים. לעומת ה-accuracy של ה-train ה-accuracy של ה-test וה-validation אינו במגמת עלייה מתמדת ו"סובל" מתנודות מסוימות. ניתן להבחין כי ה-accuracy של ה-test וה-validation יחסית קרובים בכל הגרפים כמצופה.

## DROPOUT

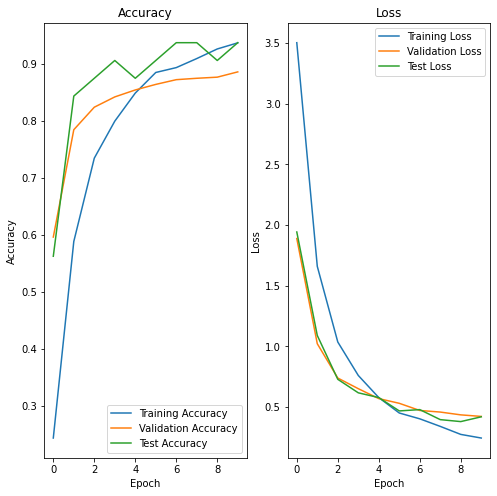
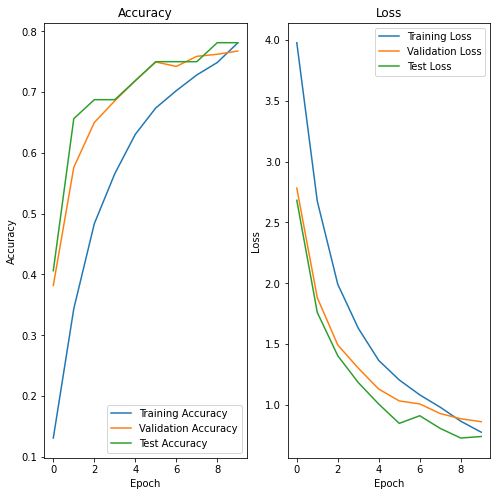
שכבת ה-Dropout מאפשר ל"כבות" אחוז מסוים מהנוירונים בשכבת ה-Dense וכך מאפשרת לרשת לבצע למידה רק עם חלק מהנוירונים. הסיבה לביצוע שכבת dropout היא להגביר את ה-robustness של הרשת על מנת שלא תהייה **רגישה** ל"רעשים" או data לא נכון בתמונות. כך, הרשת לא מסתמכת על מספר נוירונים ספציפיים בלבד.

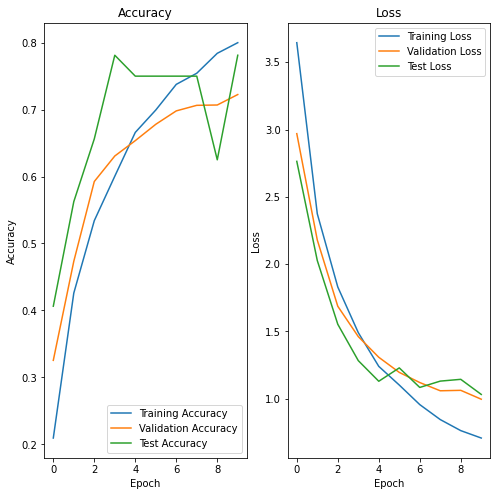
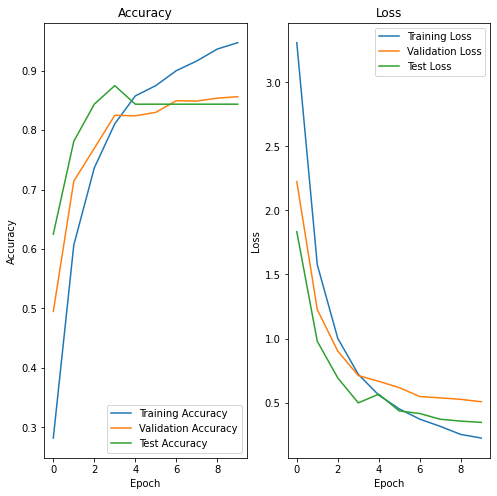
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Number run** | **Model** | **Normalization** | **Number unit dense** | **dropout\_rate** | **Way split** | **Accuracy Test** |
| **1** | mobilenet | T | 128 | 0.3 | 1 | 0.889 |
| **2** | inceptionv3 | T | 128 | 0.3 | 1 | 0.77 |
| **3** | mobilenet | T | 128 | 0.3 | 2 | 0.865 |
| **4** | inceptionv3 | T | 128 | 0.3 | 2 | 0.73 |
| **5** | mobilenet | T | 128 | 0 | 1 | 0.889 |
| **6** | inceptionv3 | T | 128 | 0 | 1 | 0.780 |
| **7** | mobilenet | T | 128 | 0 | 2 | 0.865 |
| **8** | inceptionv3 | T | 128 | 0 | 2 | 0.734 |

מסקנות מהטבלה–

* **השפעת ה-dropout** – ניתן לראות כי אין השפעה רבה לביצוע dropout. אנחנו מסיקות כי הסיבה לכך שהתמונות אינן מכילות רעשים מיותרים או מידע לא נכון (למשל תמונות שאינן צולמו טוב וכדומה).
* **השפעת ה-split –** ניתן לראות כי החלוקות השונות של ה-data אינן משפיעות בצורה רבה על ה-accuracy על ה-test. דבר זה מעיד כי הרשת עמידה לחלוקות שונות **והתוצאות אמינות** יותר.
* **מודלים –**ניתן לראות כי המודל עם הארכיטקטורה של mobilenet בעל ביצועים טובים יותר מהארכיטקטורה של inceptionv3.

גרפים מסודרים משמאל לימין לפי מספר הריצה –





מסקנות מהגרפים–

* **Loss –** בכל הגרפים ניתן לראות מגמת ירידה של ה-loss. בהרצות ללא ה-dropout הבחנו כי ה-loss על ה-train בעל שיפוע גדול יותר ויותר קרוב ל-0 מאשר ה-loss של ה-test וה-validation. ואילו כאשר הוספנו את ה-dropout ניתן לראות כי ה-loss יחסית דומים ויורדים במגמה אחידה. מכאן אנחנו מסיקים כי ה-dropout מאפשר לרשת להתכנס בצורה אחידה יותר
* **Accuracy –** קיימת מגמת עלייה של ה-accuracy על ה-train, test וה-validation בכל הגרפים. ניתן לראות כי במריבת הגרפים לקראת ה-epochs האחרונים ה-accuracy של ה-train עולה יותר מה-accuracy של ה-test וה-validation דבר המעיד על overfitting. אך, קיימת מגמת עלייה חיובית בכל ה-set השונים.

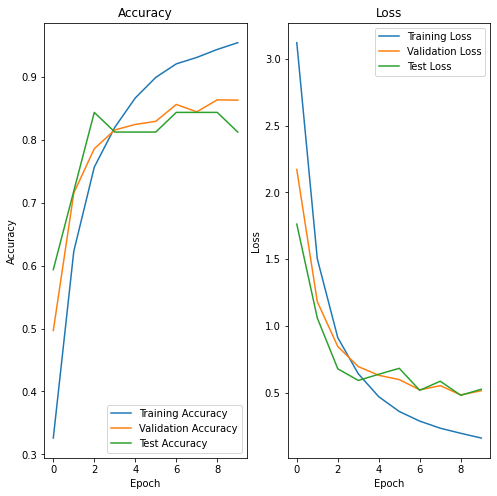
## OPRIMIZER

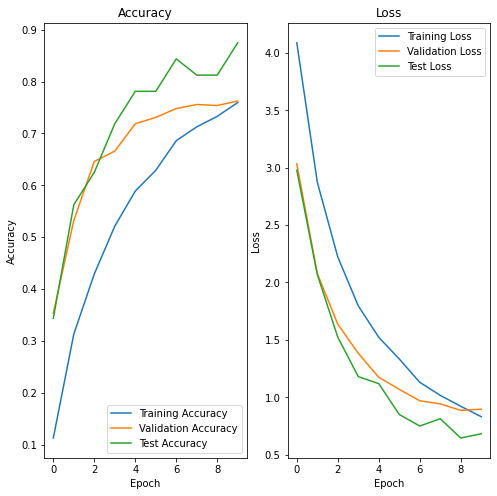
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Number run** | **Model** | **Normalization** | **Number unit dense** | **dropout\_rate** | **Way split** | **Optimizer** | **Accuracy Test** |
| **1** | mobilenet | T | 128 | 0.3 | 1 | adam | 0.889 |
| **2** | inceptionv3 | T | 128 | 0.3 | 1 | adam | 0.77 |
| **3** | mobilenet | T | 128 | 0.3 | 2 | adam | 0.865 |
| **4** | inceptionv3 | T | 128 | 0.3 | 2 | adam | 0.73 |
| **1** | mobilenet | T | 128 | 0.3 | 1 | RMSprop | 0.88 |
| **2** | inceptionv3 | T | 128 | 0.3 | 1 | RMSprop | 0.753 |
| **3** | mobilenet | T | 128 | 0.3 | 2 | RMSprop | 0.868 |
| **4** | inceptionv3 | T | 128 | 0.3 | 2 | RMSprop | 0.768 |

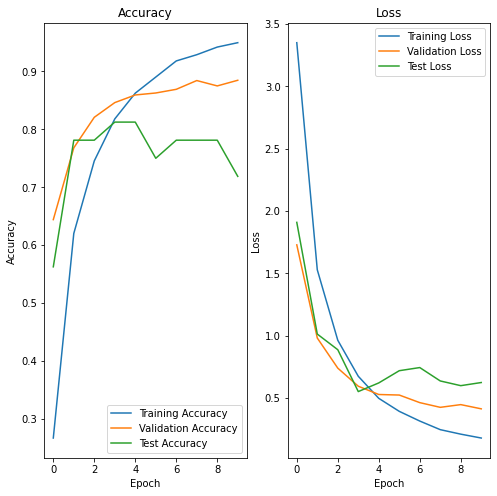
עד כה את הניסויים ביצענו עם ה-optimizer Adam אשר משתמש בשיטה של stochastic gradient. רצינו לבצע השוואה עם optimizer אחר בשם RMSprop אשר שומר על ממוצע נע של ריבועי השיפועים ומחלק את ה-gradient לפי השורש של הממוצע.

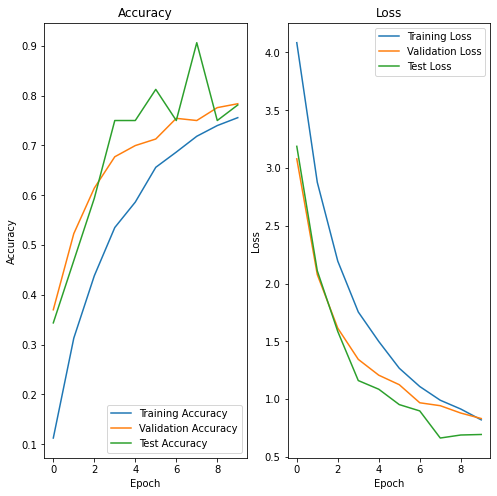
מסקנות מהטבלה–

* **השפעת ה-optimizer** – ניתן לראות כי ל-optimizer אין השפעה רבה על ה-accuracy. אנחנו מניחים כי מאחר והשינוי במשקולות נעשה רק בשכבות האחרונות (dense ו-dropout) אז ההשפעה של ה-optimizer על התכנסות המשקולות היא זניחה.

גרפים מסודרים משמאל לימין לפי מספר הריצה –







## TRAIN LAYER

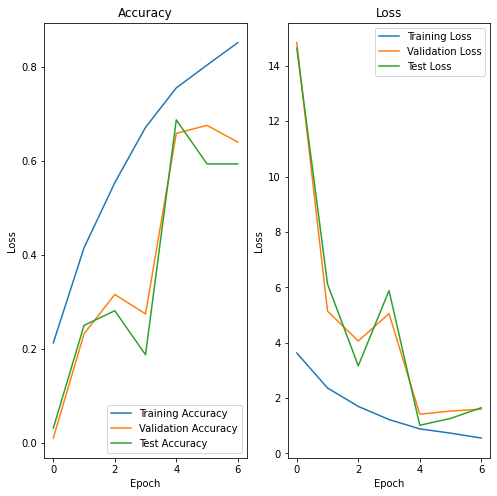
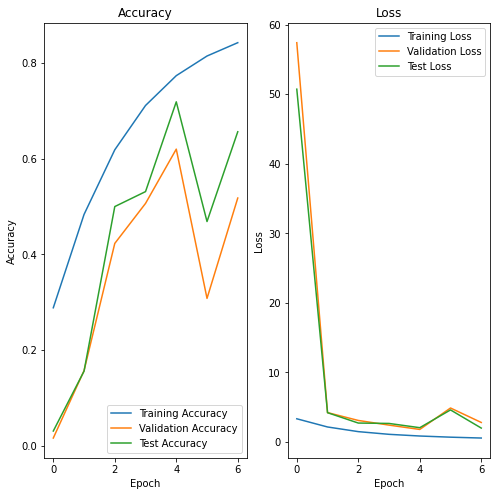
מאחר ואנחנו משתמשים במודלים מוכנים (mobilenet ו- inceptionv3) אשר אומנו על data set שונים המשקולות כבר "הותאמו" מראש לבעיות שונות. כתוצאה מכך, כשאר בנינו את הארכיטקטורה לא אימנו את המודלים שאומנו כבר מתוך הבנה שהמשקולות שהוגדרו עבורם הם האופטימליים עבור data set רבים.

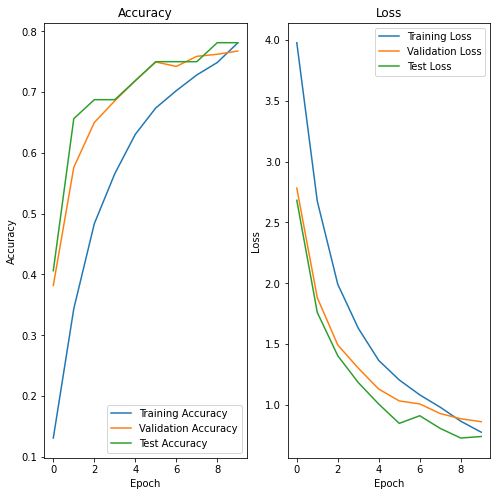
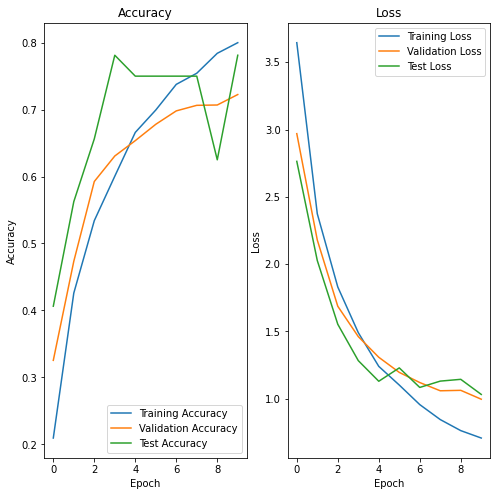
רצינו לבחון את ההשפעה של מצב בו נחליט לאמן את המודלים שבחרנו וכך להתאים את המשקולות ל-data שלנו. ביצענו את הניסויים עבור המודל inceptionv3.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Number run** | **Model** | **Normalization** | **Number unit dense** | **dropout\_rate** | **Way split** | **Train layer** | **Accuracy Test** |
| **1** | inceptionv3 | T | 128 | 0.3 | 1 | T | 0.63 |
| **2** | inceptionv3 | T | 128 | 0.3 | 2 | T | 0.5291 |
| **3** | inceptionv3 | T | 128 | 0.3 | 1 | F | 0.77 |
| **4** | inceptionv3 | T | 128 | 0.3 | 2 | F | 0.73 |

מסקנות מהטבלה–

* **השפעת אימון השכבות** – ניתן לראות בצורה מובהקת כי אימון השכבות מקטין בצורה משמעותית את אחוזי הדיוק על ה-test. אנחנו סבורות כי הסיבה לכך שמספר ה-epochs שאנחנו מבצעים הוא יחסית קטן ולכן הרשת לא מספיקה להגיע למצב יציב. בנוסף, ייתכן כי הדבר נובע מכך שהארכיטקטורה אומנה על data set רבים והגיע למצב יציב כך שכל שינוי במשקולות הרשת פוגע מהרשת להיות בנקודה האופטימלית.

גרפים מסודרים משמאל לימין לפי מספר הריצה –



# Conclusion

**מסקנות–**

1. **השוואה בין המודלים** – לאורך כל הניסויים הארכיטקטורה של ה- MobileNetהשיגה אחוזי דיוק טובים יותר על ה-test מאשר הארכיטקטורה של ה- inceptionv3. בעוד ב-inception מבצעים חלוקה של הקלט לשכבה במודל למספר יחידות עצמאיות של מידע, דוחסים (באמצעות קונבולוציה) ומאחדים ע"י שכבת concatenate, MobileNet ממפה קשרים מרחביים לכל output channel בצורה נפרדת לחלוטין ובסוף מבצעת 1X1 depth-wise convolution, כדי "לתפוס" cross-channel correlations. ארכיטקטורת MobileNet הראתה ביצועים טובים יותר על הדאטא סט של ImageNet, כפי שהיא מראה ביצועים טובים יותר עבר המשימה שלנו.
2. **השוואה בין החלוקה הרנדומלית בצורה שונה** – ראינו בכל הניסויים כי החלוקה של ה-train, test וה-validation בצורה שונה לא השפיע בצורה משמעותית על אחוזי הדיוק של הרשת. ניתן להקביל את התהליך לביצוע cross validation עבור 2-fold. דבר זה מראה כי הרשת אמינה יותר וכי התוצאות שקיבלנו אינן מושפעות מחלוקה מסוימת של ה-data.
3. **השפעת preprocessing** – תהליך ה-preprocessing הינו תהליך חשוב במיוחד כאשר המודל הינו רשת נוירונים. רצינו לבצע תהליכי preprocessing נוספים כמו ביצוע crop לתמונה על מנת למרכז את המיקום של הפרח אך מפאת זמני ריצה ארוכים לא ביצענו זאת.
   1. השפעת normalization – ראינו בצורה מובהקת כי תהליך הנרמול משפיע על הרשתות להתכנסות מהירה יותר ולאחוזי דיוק גבוהים יותר.

# SUMMARY

בעבודה זאת השתמשנו ברשתות קונבולוציה מאומנות וביצענו להן transfer learning על מנת להתאים אותם ל-dataset אחר. מצאנו את ההשפעה של תהליך preprocessing על התמונות וביצענו השוואה בין ארכיטקטורות שונות. כל אלה, הראו את ההשפעה הגדולה של פרמטרים שונים על תהליך הלמידה. ביצענו ניסויים שונים על מנת להבין את ההשפעה של הפרמטרים השונים של הרשת. הגרפים שהראו את ה-loss וה-accuracy סייעו לנו להבין כיצד הפרמטרים השונים משפיעים על הרשת בתהליך הלמידה.