|  |
| --- |
|  |
| ResNet50 – Transfer learning report |
|  |
|  |

Deep Learning Report ADeep Learning Report A

**פירוט המשימה:**

**לבנות רשת נוירונים שיודעת להבדיל בין100 קלאסים שונים של אתרים בעולם.**

**המשימה התחחלקה לשניים:**

**1. ליצור Deep Neural Network שתדע לסווג את התמונות לקלאסים השונים.**

**2. ליצור מסווג KNN שלפי הפיצ’רים של הרשת העמוקה שייצרנו תמצא את K התמונות הקרובות לה ביותר ותציג אותן.**

**פירוט הניסוי:**

**בעיה זו של הוצאת פיצ’רים מתמונה וסיווג התמונה לקלאס המתאים, היא בעיה דומה מאוד לאתגר ש imageNet מציג שמסווגת בעיקר בעלי חיים צמחים ועצמים דוממים (לא נופים כמו באתגר שלנו).**

**מהסיבה הזו החלטנו שדרך טובה לפתור את האתגר שלנו תהיה ביצוע Transfer Learning מרשת שכבר אומנה באתגר של imageNet. הרשת שבחרנו לצאת ממנה היא רשת ResNet50**

**שלבי ההרצה של ניסוי בודד:**

**- טעינת התמונות של ה TrainingData וטעינת התמונות של ה ValidationData**

**- טעינת המודל של InceptionV3**

**- הוספת שכבות ה dense האחרונות בכדי ליצור את הפלט הרצוי (מערך של 100 קלאסים עם ציונים עבור כל אחד מהם לתמונה שהתקבלה)**

**- הגדרת פונקציית ה loss ה optimizer וה Hyper Parameters הרלוונטים לאותו optimizer.**

**(learning rate, momentum וכו’)**

**- יצירת ה callbacks עבור שמירת ההיסטוריה של תוצאות ה loss וה accuracy בסיום כל epoch + שמירת המשקלים.**

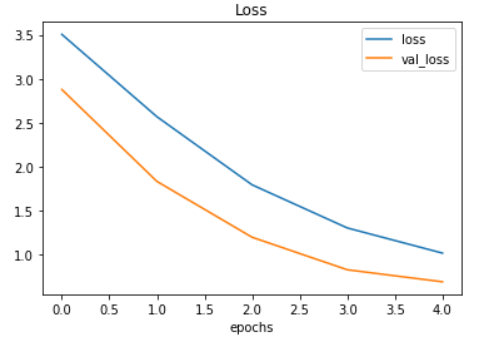
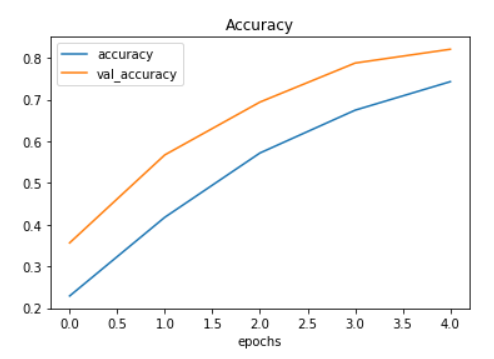
**- אימון המודל.**

**- יצירת הגרפים ומדידת תוצאות האימון.**

**חלק ראשון קיבוע ה Learning Rate:**

**ראשית, ננסה למצוא את הLearning Rate המתאים ביותר בהתייחס לאלגוריתם InceptionV3. ניקח את SGD כנקודת המוצא שלנו וממנו נתקדם למקומות אחרים.**

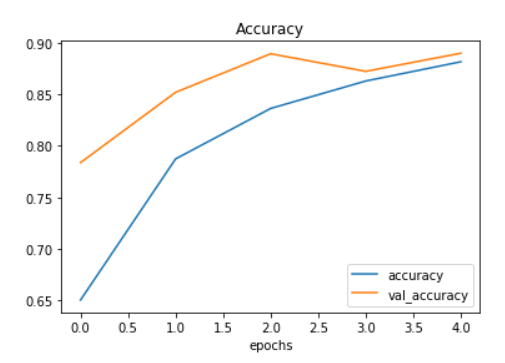
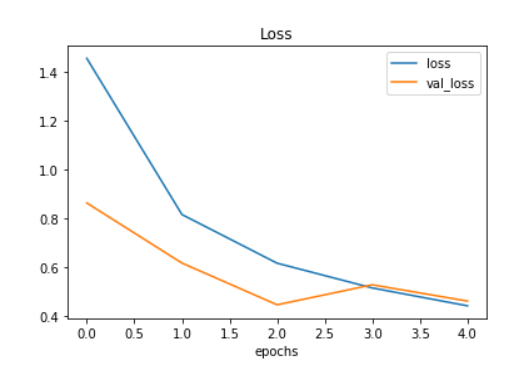
**נשתמש כמות קטנה יחסית של epochs מכיוון שבשלב זה נרצה בעיקר לראות מגמות של איך אנחנו מתפקדים עם learning rates שונים.**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Epochs | momentum | Batch Size | Learning Rate | Optimizer | Loss Function |
| 5 | 0.9 | 10 | 0.0001 | SGD - Momentum | categorical\_crossentropy |

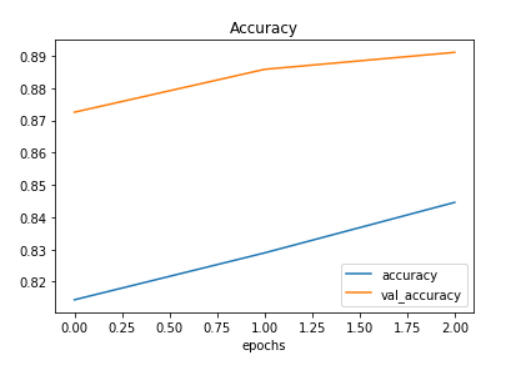
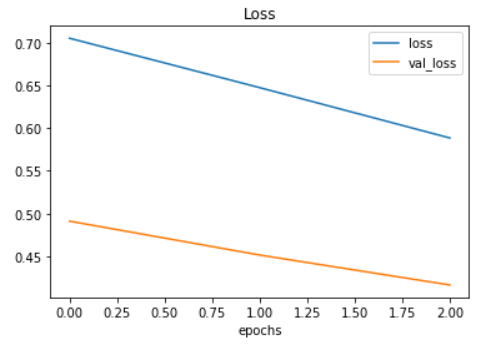
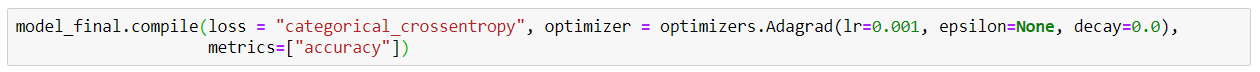
חלק שני: שינוי הlearning rate –

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Epochs | momentum | Batch Size | Learning Rate | Optimizer | Loss Function |
| 5 | 0.9 | 10 | 0.001 | SGD - Momentum | categorical\_crossentropy |



**Adagrad optimizer:**

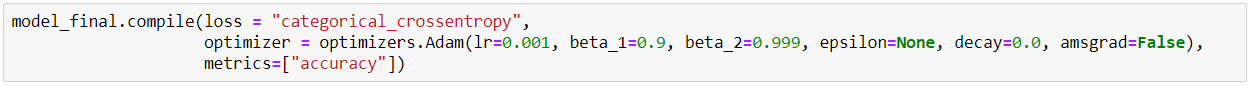
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Epochs | momentum | Batch Size | Learning Rate | Optimizer | Loss Function |
| 10 | 0.9 | 10 | 0.001 | Adagrad | categorical\_crossentropy |

כעת ננסה לאתגר מעט את אלגוריתם הלמידה ולהתאים אותו למאגר התמונות. ניתן לראות בברור שישנן תמונות באיכות נמוכה יותר מאחרות, לכן למרות שאנחנו מבצעים נורמליזציה ומגדירים את העיבוד על התמונות בגודל זהה, עדיין מתקבלות תמונות באיכויות שונות. לכן, במקום להשתמש ב Optimizer של SGD, נשתמש ב Adagrad. Adagrad ידוע ביכולת שלו להתמודד עם מידע מדולל, בכך שמבצע עדכונים עם שיעור למידה נמוך עבור פרמטרים הבאים לידיי ביטוי בפיצ'רים בתדירות גבוהה ושיעור למידה גבוהה עבור פיצ'רים בתדירות נמוכה. באופן זה האלגוריתם צריך לדעת להתמודד עם מידע דליל יותר.

3541/3541 [==============================] - 1050s 296ms/step - loss: 0.3668 - acc: 0.8985 - val\_loss: 0.3195 - val\_acc: 0.9237

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epochs | Batch Size | Learning Rate | Optimizer | Loss Function |
| 5 | 10 | 0.001 | Adagrad | categorical\_crossentropy |

**Adam Optimizer:**

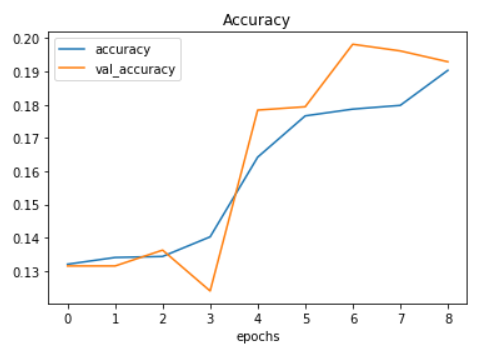
כעת, נרצה לבדוק Optimizer שונה.

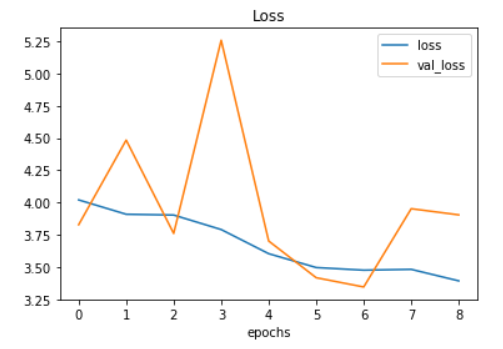
עפ"י בדיקות נראה שAdam הוא Optimazer בעל ביצועים גבוהים.

נאתחל אותו בערכי ברירת המחדל וננסה להריץ את אלגוריתם InceptionV3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epochs | Batch Size | Learning Rate | Optimizer | Loss Function |
| 10 | 10 | 0.001 | Adam | categorical\_crossentropy |

נראה את תוצאות האלגוריתם :



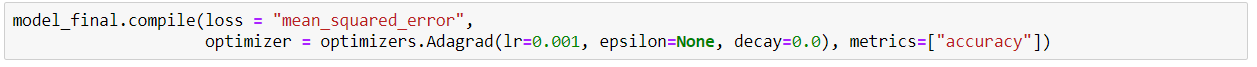
תוצאות האלגוריתם אינן מספקות כלל ומראות חוסר יציבות גבוהה.

ככל הנראה הנושא נוצר עקב שילוב בעייתי של hyperparameters או ביצוע התאמות ברמת הרשת הנדרשות להרצה של Adam כ Optimizer של InceptionV3.

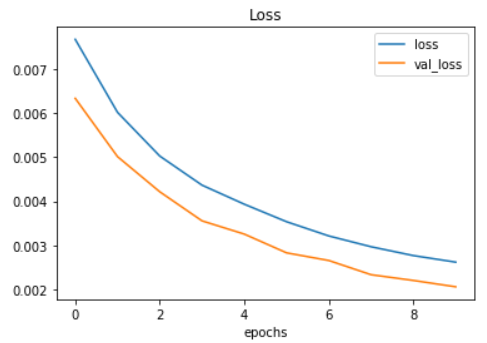
**שינוי פונקציות Loss :**

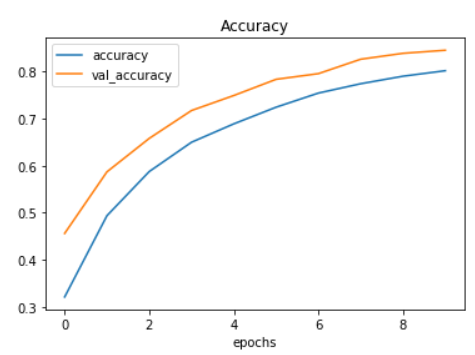
כעת נתבנון על ההשפעה של פונקציות ה-Loss השונות על האלגוריתם שלנו.

אומנם mean\_squared\_error היא פונק' הפסד המשמשת לבעיות רגרסיה, אך היא מוכרת מאוד. מתוך סקרנות מחקרית, נבדוק מה יהיו הביצועים שלה על המודל שבנינו.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epochs | Batch Size | Learning Rate | Optimizer | Loss Function |
| 10 | 10 | 0.001 | Adagrad | mean\_squared\_error |



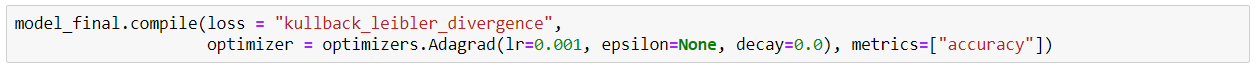


לא הצלחנו להתעלות על התוצאות הקודמות שקיבלנו.

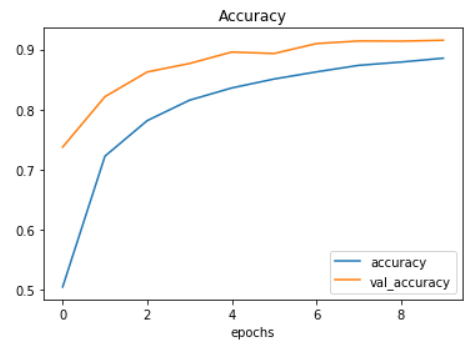
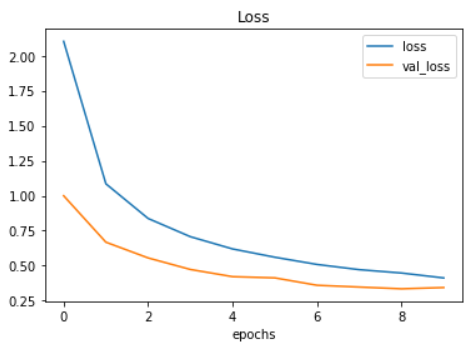
ננסה לחשוב על פונק' הפסד אחרת, שיתכן ותשפר את ביצועי המודל על ה- validation\_data.

**kullback\_leibler\_divergence:**

כעת ננסה להשתמש בפונק' הפסד שונה. פונק' זו דומה מאוד בהתנהגותה לפונק' ההפסד המוכרת cross-entropy, אך זו משמשת ללמידה וסיווג של בעיות מורכבות יותר מסיווג מרובה (multi-class classification). מעניין לראות כיצד תשפיע על פתרון בעיית האופטימיזציה בפניה אנו עומדים.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epochs | Batch Size | Learning Rate | Optimizer | Loss Function |
| 10 | 10 | 0.001 | Adagrad | kullback\_leibler\_divergence |



3541/3541 [==============================] - 973s 275ms/step - loss: 0.4124 - acc: 0.8854 - val\_loss: 0.3438 - val\_acc: 0.9152

Out[72]: