**למידה חישובית יישומית- פרויקט מסכם**

העבודה מתבססת על המאמר  [SGDR: STOCHASTIC GRADIENT DESCENT WITH WARM RESTARTS](https://arxiv.org/abs/1608.03983)שכתבוIlya Loshchilov & Frank Hutter ב2016.

המאמר מתמקד בStochastic Gradient Descent with Warm Restarts או בקיצור SGDR.

מוטיבציה:

תחום הAI הינו תחום מתפתח אשר צובר תאוצה בזמן האחרון.

ML הינו תת תחום של ה AL אשר צובר תאוצה גם הוא.

עולם ה Deep learning שייך לתחום ML ומהווה חלק משמעותי מתחום זה, הרבה משימות סיווג שמבוצעות באופן אוטומטי משתמשות ב Deep learning .

בעולם ה Deep learning קיימים אופטימייזרים שונים שמטרתם להאיץ את תהליך אימון הרשת ושיפור איכות התוצאות של הרשת.

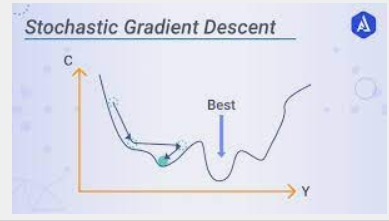
הם עושים זאת על ידי שינוי המשקלים של המשתנים ברשת , שינוי קצב הלמידה ועוד.

המאמר שלנו מספק חידוש באופטימייזר מסוג SGD שמביא לשיפור משמעותי ביכולות האופטימייזר ובמהירות אימון הרשת.

מאמר זה נחשב כפורץ דרך בתחום האופטימייזירים ומספק יתרון משמעותי באימון מודלים השייכים ל Deep learning.

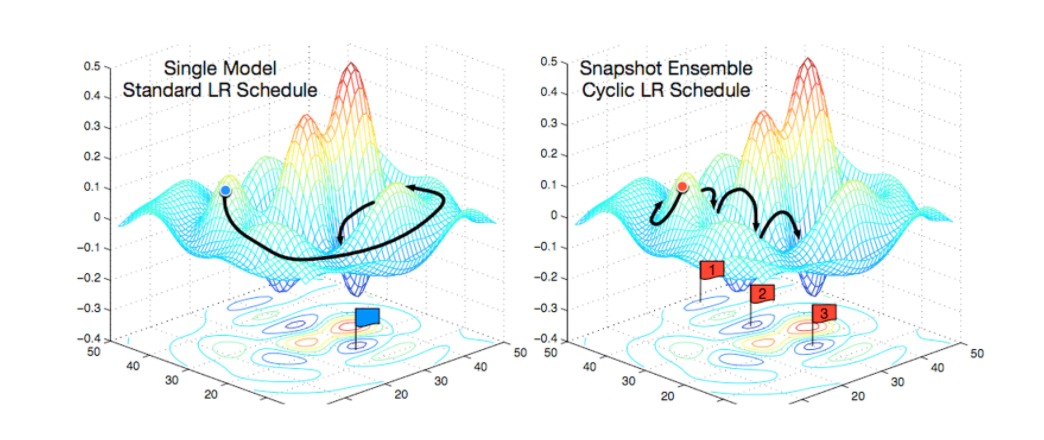
**בכדי להבין את SGDR ראשית נסביר על GD וSGD:**

GD: האופטימייזר הבסיסי נקרא gradient descent- עוברים על כל הנתונים שיש לנו, מחשבים את הנגזרת בנקודה שאנחנו נמצאים בה ונעים בניגוד לגרדיאנט. קיים משתנה שמשפיע על כמות התזוזה בניגוד לגרדיאנט. אם הוא מאוד קטן אנחנו נתכנס מאוד מאוד לאט. אם הוא מאוד גדול יכול להיות שלא נתכנס ואף נתבדר כלומר מצעד לצעד נלך לפתרון פחות טוב מהקודם). בשיטה זו אנו נתקדם רק לאחר מעבר על כל הנתונים ולכן יהיה מאוד קל להיתקע במינימום מקומי. ואין אקראיות כיוון שעוברים תמיד על כל הנתונים.

SGD(Stochastic gradient descent)- שיטה המשמשת לאופטימזציה של פרמטרים במודל למידה עמוקה וקיצור זמני הריצה של האלגוריתם. המודל בוחר בכל איטרציה שלו דוגמאות רנדומליות בדאטה סט במקום את כל הדאטה סט. SGD יעשה הרבה יותר ריצות ממודל GD רגיל אך עדיין מכיוון וכל ריצה הינה קצרה משמעותית, הוא יהיה משמעותית יותר מהיר. באופן פרקטי נעדכן את משקולות המודל אחרי כל batch (שזה אומר חלק מהנתונים).  מאפשר מהירות ואקראיות בבחירה של הנתונים באותו batch. בGD רגיל קיימת בעיה של התכנסות למינימום מקומי, SGD מהווה פתרון לבעיה ע"י כך שהוא נעזר במומנטום. אם הוא צובר תאוצה ועובר את המינימום המקומי, הוא בעצם לא יתקע במינימום המקומי ויחקור מקומות נוספים בדאטה. SGD מתחשב בגרדיאנט הנוכחי בתוספת משקול של הגרדיאנטים הקודמים.

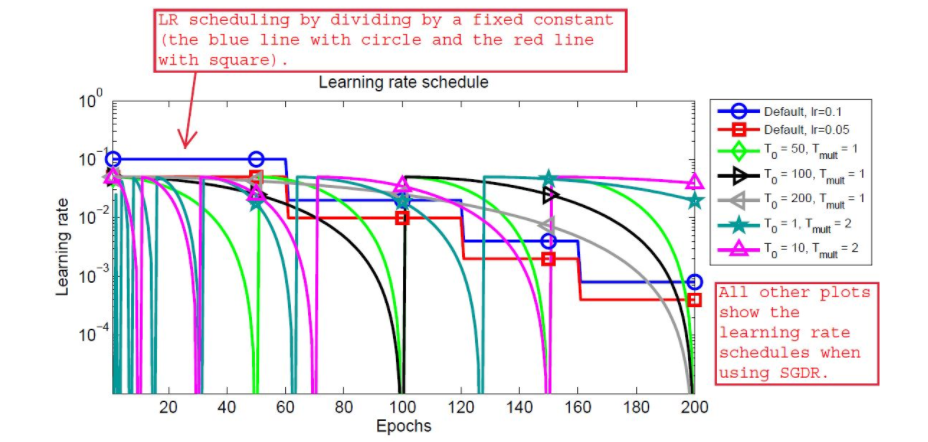
כעת נסביר מה הוא -SGDR אלגוריתם SGD שמשתמש ב.warm restart SGDR מאפשר קצב למידה (learning rate) משתנה תוך כדי האימון, זאת ועוד, משמש לתזמון קצב הלמידה של המודל בזמן האימון, האלגוריתם מתחיל מקצב למידה מסויים ומוריד אותו בהדרגה עד epoch מסוים, כאשר הוא מגיע לepoch קצב הלמידה חוזר לערך ההתחלתי וכך חוזר חלילה. SGDR עוזר למודל הרשת הנוירונים להתכנס הרבה יותר מהר מאשר בשימוש בקצב למידה סטטית.

Warm restart בדרך כלל מאפשר את גיוון תוצאות המודל מאחר שכל פעם אנחנו מתחילים מנקודה אחרת ולא נשאבים למינימום מקומי. בSGDR נעשה שימוש בwarm restart בצורה מגוונת אך מבוקרת, שימוש זה מאפשר לעבוד באזור המינימום המקומי ובכל זאת מאפשרת התכנסות למינימום הגלובלי.



ניתן לראות שבשתי התמונות קיימת התכנסות למינימום הגלובלי, אך בכל זאת נראה שבתמונה השמאלית ההתכנסות איטית יותר. בתמונה הימנית אנו רואים את השימוש בwarm restart מה שמאפשר להיות באזור המינימום המקומיים ו"לדלג" עליהם עד התכנסות למינימום הגלובלי בצורה יעילה.

דוגמת מימוש מהמאמר:



בתמונה זו ניתנה דוגמא להתפלגות ירידה/עלייה של קצב הלמידה באלגוריתמי תזמון לינאריים ומבוססי SGDR. ניתן לראות שני פרמטרים מרכזיים באלגוריתם, tmult ו.t0

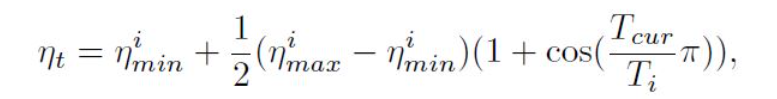
Tmult- אחראי על הכפלת טווח הepochs שקצב הלמידה ישתנה בהם.

T0(בקוד המקור נקרא first decay steps) - הערך ההתחלתי המגדיר את כמות הepochs שקצב הלמידה יירד בהם עד שיעלה חזרה לערך ההתחלתי.

דוגמא- אם ניקח את הגרף הורוד למעלה בו T0=10 וTmult=2(ניתן לראות מימין לגרף), נראה שהוא משלים את ירידת קצב הלמידה תוך 10 epochs ובכל איטרציה הוא מכפיל את המספר פי 2. כלומר הוא משלים בפעם הראשונה את האיטרציה ב10 epochs, בפעם השנייה ב20, בפעם הבאה ב40 וכך הלאה..

Momentum- משתנה זה מתייחס לגודל השינוי של קצב הלמידה בהתחשבות במצבו הנוכחי של האלגוריתם והגדרות קצב הלמידה המינימלי והמקסימלי במודל.

Momentum- מוגדר במאמר ע"י הנוסחה הבאה:



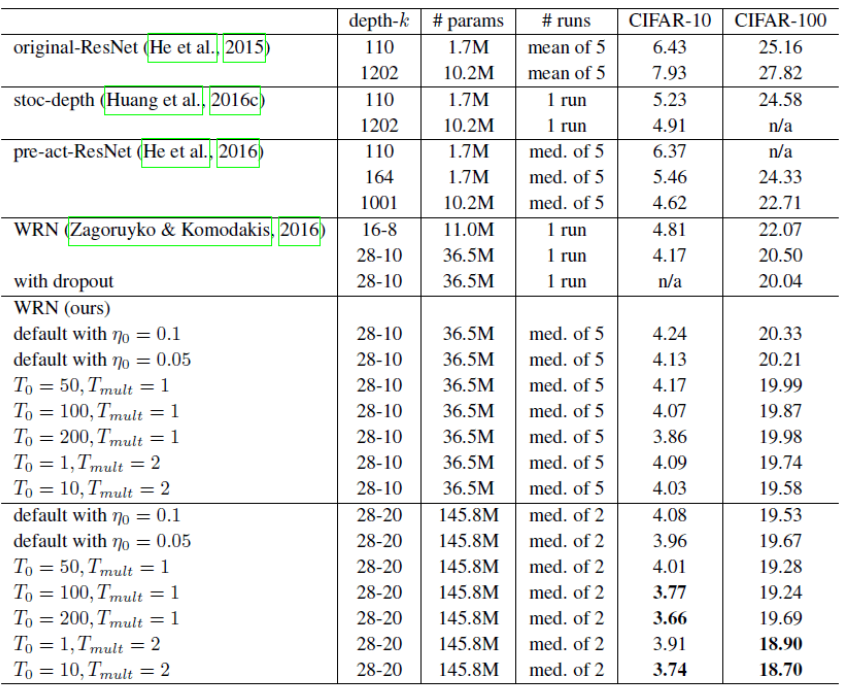
Nmin מייצג את קצב הלמידה המינימלי שהוגדר למודל.

Nmax מייצג את קצת הלמידה המקסימלי שהוגדר למודל.

Tcur- עוקב אחרי מספר הepochs שעברו מאז תחילת האיטרציה(warm restart) ומתעדכן בכל איטרציה.

Ti- מייצג את מספר הepochs של הריצה ה-i.

במאמר ניתן היה לראות שהכותבים הגיעו למסקנה מהניסויים שעשו עם שינוי הפרמטרים, ההתחלה עם T0 נמוך וTmult גדול מ1, נתנו את טעות הבדיקה הנמוכה ביותר על מירב הדאטה סטים.



צירפנו את התוצאות מהניסוי במאמר, ניתן לראות את השיפור הברור מול אלגוריתמים סטנדרטיים.

יתרונות מרכזיים באלגוריתם SGDR:

1. SGDR משפר את SGD מבחינת התכנסות והן מבחינת איכות התוצאות.
2. כאשר משתמשים בSGDR, המודל יתכנס יותר מהר מאשר עם אלגוריתמי תזמון אחרים. זאת במידה ונבצע תהליך hyper parameter optimization איכותי, כך המודל יוכל להשיג שגיאת מבחן(test error) נמוכה כבר באפוקים הראשונים של הריצה.
3. SGDR יעיל מאוד בביצוע תהליך ensemble של מודלים.

חסרונות באלגוריתם SGDR:

1. החיסרון היחידי שעלול להופיע באלגוריתם זה הוא תהליך היפר פרמטר לא איכותי עם טווח ערכים נמוך. לצורך העניין אם ניקח קצב למידה גבוה מאוד בהתחלה וקצב ההתחלות החדשות יהיה מהיר מאוד אז בעצם נתקבע על טווח גבוה של קצב למידה כאשר קצב זה לא בהכרח מתאים לכל מאגרי המידע.
2. SGDR מתאים ספציפית לבעיות סיווג של תמונות ולא בהכרח לכל בעיה אחרת, כלומר הוא בהכרח ישפר מודלים שנועדו לסיווג תמונות אך לא בהכרח ישפר מודלים שנועדו לסוג מידע אחר.
3. SGDR אינו בהכרח ישפיע לטובה על מידע מסוג multi-modality.

תהליך המימוש של קוד המקור(source code):

הקדמה-

נתאר את התהליך שעברנו בכדי להגיע לקוד המקור הסופי-

1. ניסיון לעבוד עם הקוד המקורי של המאמר. הניסיון לא צלח, לא הצלחנו להריץ את הקוד אז עזבו אותו לאחר יומיים של ניסיונות.
2. ניסיון לממש את המאמר בעצמנו. הניסין לא צלח עקב מורכבות האלגוריתם, לאחר מספר ניסיונות שונים זנחנו רעיון זה.
3. ניסיון לעבוד עם מימוש של המאמר תחת חבילת pytorch. גם הפעם הניסיון לא צלח עקב התקלות בקשיים טכניים ועבודה עם קוד מסובך להבנה של המממש.
4. **לבסוף מצאנו מימוש של המאמר בספריית keras.**

<https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/schedules/CosineDecayRestarts>

מימוש הקוד לפי שלבים(pseudo code):

\*הנחנו שהכוונה בpseudo code הייתה לכתוב את שלבי האלגוריתם(גם במאמר אין פסודו קוד).

1. יצירת משתנים גלובליים למטרת עבודה גנרית עם הdatasets השונים.
2. טעינת סט האימון וסט הבדיקה.
3. תהליך pre-processing- בהתחלה מימשנו את התהליך pre-processing כפי שתואר במאמר עם data augmentation, resizing, rescaling אך בשל אילוצי זמן ריצה החלטנו לוותר על הdata augmentation למטרת סיום המשימה בזמן ומכיוון שהמאמר עוסק בעיקר באופטימייזר ולא בתהליך הpre-processing.
4. בניית המודל- כמו במאמר, השתמשנו ברשת מסוג resnet המממשת את עיקרון הresidual blocks(תואר בהרחבה בעבודה 3). גם אנחנו השתמשנו ברשת מסוג resnet שטענו אותה מאומנת מראש(pre-trained) מספריית keras. מבנה הרשת אותה יצרנו:

* שכבת קלט.
* המודל המאומן אותו טענו.
* שכבת batchnormalizion.
* שכבת dropout.
* שכבת סיווג כגודל מספר הקלאסים שצריך לחזות.

לאחר מכן הפשרנו(unfreeze) מספר שכבות מהמודל המאומן, לטובת אימון הרשת.

1. אופטימייזר: לטובת מימוש SGDR השתמשנו באופטימייזר מסוג SGD של חבילת keras.

האופטימייזר מממש gradient descent עם מומונטום. על מנת לבצע warm restart וקצב למידה משתנה בחרנו להשתמש במתזמן קצב למידה אשר מבוסס על cosine decay עם warm restart. קישור לחבילה המממשת SGD- <https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/SGD>.

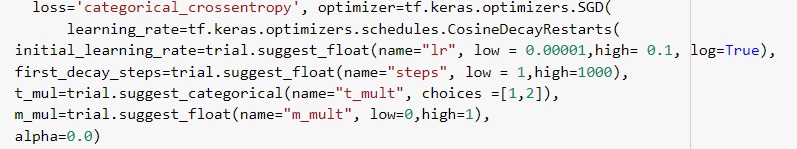
קישור לחבילה המממשת את cosine with warm restart- <https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/schedules/CosineDecayRestarts#returns>

1. תהליך הhyperparameter optimization: על מנת לבצע את תהליך זה בחרנו להשתמש בספריית optuna. ספרייה חכמה המאפשרת בחירת היפר פרמטר באופן דינמי ומושכל.

על מנת להשתמש בספרייה זו מימשנו את פונקציית objective.

בפונקציה זו, יצאנו את המודל והרצנו אותו. כאשר יצרנו את המודל, הגדרנו לו משתנים אשר להם הגדרנו טווח מינימום ומקסימום המשתנה בין הרצות המודל.

דוגמאות למשתנים מסוג float וקטגורי שביצענו עליהם אופטימיזציה:

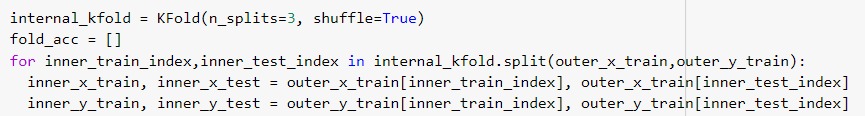
הפונקציה אשר מפעילה את objective הינה study. פונקציה זו יוצרת ניסוי אשר מטרתו למצוא את הפרמטרים הטובים ביותר. אנו מגדירים לstudy האם מטרתה הוא למקסם את ה"רווח" או להביא אותו למינימום. בסיום הריצה של study אנו יכולים לקבל את הפרמטרים הטובים ביותר שהיא מצאה. את study אנו מריצים עם מספר trials מסוים אשר מגדיר את מספר הקומבינציות המקסימלי בין פרמטרים שהפונקציה תריץ.

C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\WhatsApp Image 2021-07-17 at 11.08.29 AM.JPEG

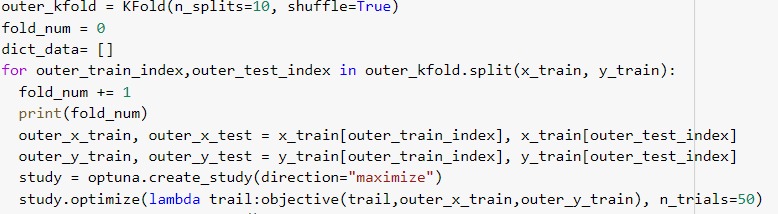
המשתנים אשר בחרנו לעשות להם אופטימיזציה הינם: קצב הלמידה ההתחלתי, T0, tmult וmomentum.

1. לולאות cross-validation:

* לולאה פנימית- נמצאת בתוך הפונקציה objective ומבצעת את תהליך האופטימיזציה על שלוש חלוקות שונות של הדאטה.



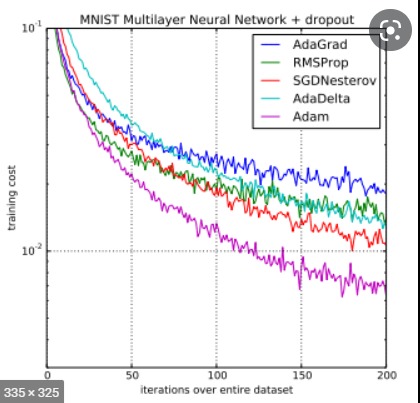
* לולאה חיצונית- לולאה אשר מריצה את הניסוי על עשר חלוקות שונות של הדאטה.

אשר בכל חלוקה של הדאטה, מוציאה נתונים אמפיירים על הניסוי עם המשתנים שהביאו לאופטימיזיציה מקסימילית. 

**אלגוריתם מוכר**

בחרנו להשתמש באופטימייזר מסוג Adam. Adam הינו אופטימייזר מוכר מאוד ובשימוש רב בעולם הDeep learning. אופטימייזר זה מתחשב בהיסטוריה של קצב הלמידה הקודם, כלומר אם קיימים כמה קצבי למידה קודמים הוא ייתן לקרובים יותר משקל יותר גבוה. Adam הינו הרחבה של SGD ויש לו מספר יתרונות מרכזיים כמו: מימוש פשוט, יעיל בזמן החישוב שלו, לא דורש הרבה זיכרון ועוד.

מימוש האלגוריתם: מימוש זהה לקוד המקור שתואר לעיל מלבד החלפת האופטימייזר מSGD לAdam, ובנוסף הורדת תזמון קצב הלמידה המשתנה כלומר עבודה עם קצב למידה קבוע. בהתאם לכך גם

המשתנים אשר ביצענו להם אופטימיזציה השתנו.

קישור לספריית Adam של keras בה השתמשנו- <https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam>

מימוש הקוד לפי שלבים(pseudo code), השינויים מקוד המקור מסומנים באדום:

1. יצירת משתנים גלובליים למטרת עבודה גנרית עם הdatasets השונים.
2. טעינת סט האימון וסט הבדיקה.
3. תהליך pre-processing- בהתחלה מימשנו את התהליך pre-processing כפי שתואר במאמר עם data augmentation, resizing, rescaling אך בשל אילוצי זמן ריצה החלטנו לוותר על הdata augmentation למטרת סיום המשימה בזמן ומכיוון שהמאמר עוסק בעיקר באופטימייזר ולא בתהליך הpre-processing.
4. בניית המודל- כמו במאמר, השתמשנו ברשת מסוג resnet המממשת את עיקרון הresidual blocks(תואר בהרחבה בעבודה 3). גם אנחנו השתמשנו ברשת מסוג resnet שטענו אותה מאומנת מראש(pre-trained) מספריית keras. מבנה הרשת אותה יצרנו:

* שכבת קלט.
* המודל המאומן אותו טענו.
* שכבת batchnormalizion.
* שכבת dropout.
* שכבת סיווג כגודל מספר הקלאסים שצריך לחזות.

לאחר מכן הפשרנו(unfreeze) מספר שכבות מהמודל המאומן, לטובת אימון הרשת.

1. אופטימייזר: לטובת מימוש Adam בחבילה keras.

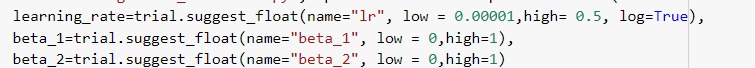
<https://keras.io/api/optimizers/adam/>

1. תהליך הhyperparameter optimization: על מנת לבצע את תהליך זה בחרנו להשתמש בספריית optuna. ספרייה חכמה המאפשרת בחירת היפר פרמטר באופן דינמי ומושכל.

על מנת להשתמש בספרייה זו מימשנו את פונקציית objective.

בפונקציה זו, יצאנו את המודל והרצנו אותו. כאשר יצרנו את המודל, הגדרנו לו משתנים אשר להם הגדרנו טווח מינימום ומקסימום המשתנה בין הרצות המודל.

דוגמאות למשתנים שביצענו עליהם אופטימיזציה:



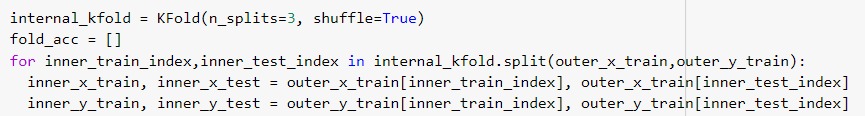
הפונקציה אשר מפעילה את objective הינה study. פונקציה זו יוצרת ניסוי אשר מטרתו למצוא את הפרמטרים הטובים ביותר. אנו מגדירים לstudy האם מטרתה הוא למקסם את ה"רווח" או להביא אותו למינימום. בסיום הריצה של study אנו יכולים לקבל את הפרמטרים הטובים ביותר שהיא מצאה. את study אנו מריצים עם מספר trials מסוים אשר מגדיר את מספר הקומבינציות המקסימלי בין פרמטרים שהפונקציה תריץ.

C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\WhatsApp Image 2021-07-17 at 11.08.29 AM.JPEG

המשתנים אשר בחרנו לעשות להם אופטימיזציה הינם: קצב הלמידה, beta1, beta2.

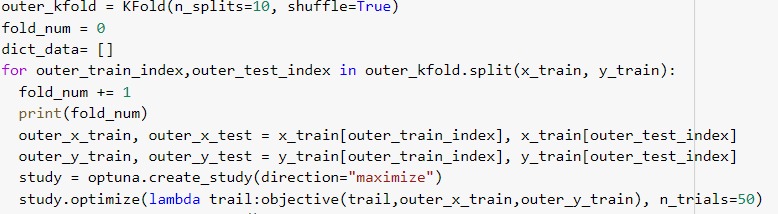
1. לולאות cross-validation:

* לולאה פנימית- נמצאת בתוך הפונקציה objective ומבצעת את תהליך האופטימיזציה על שלוש חלוקות שונות של הדאטה.



* לולאה חיצונית- לולאה אשר מריצה את הניסוי על עשר חלוקות שונות של הדאטה.

אשר בכל חלוקה של הדאטה, מוציאה נתונים אמפיירים על הניסוי עם המשתנים שהביאו לאופטימיזיציה מקסימילית.



**שיפור קוד המקור**

לאחר קריאה מעמיקה של המאמר, שמנו לב שבנקודות לדיון שנשארו פתוחות, כותבי המאמר העלו את האופציה של שימוש באופטימייזר מסוג Adam עם warm restart. החלטנו לבדוק באינטרנט האם באמת ההשערה של הכותבים הגיונית ואכן Adam יוכל לשפר את קוד המקור הקיים.

ראינו שבכמעט כל בדיקה שעשו adam מייצר תוצאות טובות יותר מsgd כאשר הניסוי בוצע על סיווג תמונות. החלטנו לבדוק בדיקה אמפירית בין שני האופטימייזרים באמצעות מודל סיווג פשוט. ראינו שבבדיקה אכן adam נתן תוצאות איכותיות יותר מSGD ולכן החלטנו לממש את Adam עם warm restarts. בנוסף במהלך החיפוש נתקלנו בניסויים שונים אשר ביצעו אופטימיזציה כלשהיא לאדם, חלקם אף שינו את האלגוריתם הבסיסי והוסיפו מתזמנים שונים. רעיונות אלו חיזקו את עמדתינו ששילוב של adam עם warm restart יוכלו להשתלב טוב ולהניב תוצאות טובות.

מימוש בפועל של Adam עם warm restarts: מימוש זהה לקוד המקור שתואר לעיל מלבד החלפת האופטימייזר מSGD לAdam ובניגוד לאלגוריתם המוכר השארנו את קצב הלמידה המשתנה ואת הwarm restart.

מימוש הקוד לפי שלבים(pseudo code):

1. יצירת משתנים גלובליים למטרת עבודה גנרית עם הdatasets השונים.
2. טעינת סט האימון וסט הבדיקה.
3. תהליך pre-processing- בהתחלה מימשנו את התהליך pre-processing כפי שתואר במאמר עם data augmentation, resizing, rescaling אך בשל אילוצי זמן ריצה החלטנו לוותר על הdata augmentation למטרת סיום המשימה בזמן ומכיוון שהמאמר עוסק בעיקר באופטימייזר ולא בתהליך הpre-processing.
4. בניית המודל- כמו במאמר, השתמשנו ברשת מסוג resnet המממשת את עיקרון הresidual blocks(תואר בהרחבה בעבודה 3). גם אנחנו השתמשנו ברשת מסוג resnet שטענו אותה מאומנת מראש(pre-trained) מספריית keras. מבנה הרשת אותה יצרנו:

* שכבת קלט.
* המודל המאומן אותו טענו.
* שכבת batchnormalizion.
* שכבת dropout.
* שכבת סיווג כגודל מספר הקלאסים שצריך לחזות.

לאחר מכן הפשרנו(unfreeze) מספר שכבות מהמודל המאומן, לטובת אימון הרשת.

1. אופטימייזר: לטובת מימוש Adam עם warm restart השתמשנו באופטימייזר מסוג Adam של חבילת keras.

על מנת לבצע warm restart וקצב למידה משתנה בחרנו להשתמש במתזמן קצב למידה אשר מבוסס על cosine decay עם warm restart. קישור לחבילה המממשת Adam- <https://keras.io/api/optimizers/adam/>

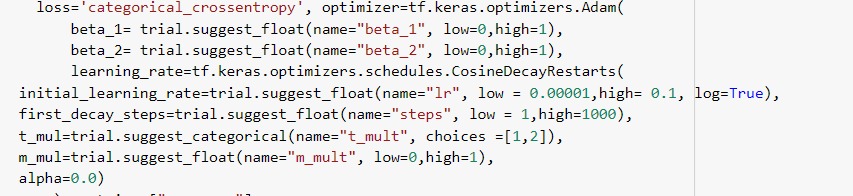
קישור לחבילה המממשת את cosine with warm restart- <https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/schedules/CosineDecayRestarts#returns>

1. תהליך הhyperparameter optimization: על מנת לבצע את תהליך זה בחרנו להשתמש בספריית optuna. ספרייה חכמה המאפשרת בחירת היפר פרמטר באופן דינמי ומושכל.

על מנת להשתמש בספרייה זו מימשנו את פונקציית objective.

בפונקציה זו, יצאנו את המודל והרצנו אותו. כאשר יצרנו את המודל, הגדרנו לו משתנים אשר להם הגדרנו טווח מינימום ומקסימום המשתנה בין הרצות המודל.

דוגמאות למשתנים מסוג float וקטגורי שביצענו עליהם אופטימיזציה:



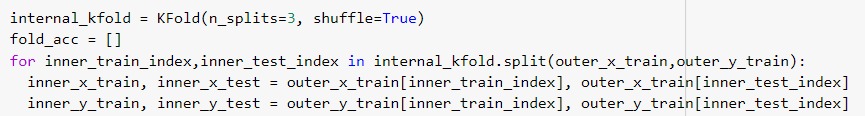
הפונקציה אשר מפעילה את objective הינה study. פונקציה זו יוצרת ניסוי אשר מטרתו למצוא את הפרמטרים הטובים ביותר. אנו מגדירים לstudy האם מטרתה הוא למקסם את ה"רווח" או להביא אותו למינימום. בסיום הריצה של study אנו יכולים לקבל את הפרמטרים הטובים ביותר שהיא מצאה. את study אנו מריצים עם מספר trials מסוים אשר מגדיר את מספר הקומבינציות המקסימלי בין פרמטרים שהפונקציה תריץ.

C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\WhatsApp Image 2021-07-17 at 11.08.29 AM.JPEG

המשתנים אשר בחרנו לעשות להם אופטימיזציה הינם: קצב הלמידה ההתחלתי, T0, tmult וmomentum, beta1, beta2.

1. לולאות cross-validation:

* לולאה פנימית- נמצאת בתוך הפונקציה objective ומבצעת את תהליך האופטימיזציה על שלוש חלוקות שונות של הדאטה.



* לולאה חיצונית- לולאה אשר מריצה את הניסוי על עשר חלוקות שונות של הדאטה.

אשר בכל חלוקה של הדאטה, מוציאה נתונים אמפיירים על הניסוי עם המשתנים שהביאו לאופטימיזיציה מקסימילית.

**מאגר המידע**

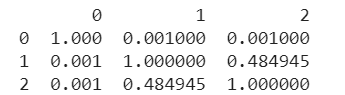
דאטה סטים בהם השתמשנו (כל הדאטה סטים הינם דאטה סטים מובנים לסיווג תמונות של tensor):

1. oxford\_flowers102
2. beans
3. cassava
4. cassava\_Part\_B
5. cats\_vs\_dogs
6. cat\_vs\_dogs\_part\_b
7. cat\_vs\_dogs\_part\_c
8. cifar10\_1
9. cycle\_gan
10. cycle\_gan\_part\_b
11. colorectal\_histology
12. colorectal\_histology\_part\_B
13. dtd
14. dtd\_part\_b
15. horses\_or\_humans
16. oxford\_iiit\_pet
17. rock\_paper\_scissors
18. tf\_flowers
19. tf\_flowers\_part\_b
20. tf\_flowers\_part\_c

**מבחן פרידמן**

ביצענו מבחן פרידמן על ממד TPR: על פי המבחן קיים הבדל מובהק בין האלגוריתמים כאשר אלפא=0.05.



בכדי למצוא איזה אלגוריתמים שונים זה מזה עברנו למבחן מסוג **HOC**.

כאשר 0 מייצג את קוד המקור, 1 את האלגוריתם המוכר(Adam) ו2 את שיפור קוד המקור.

כפי שניתן לראות קוד המקור הניב תוצאות שונות בצורה מובהקת מהאלגוריתם המוכר ומהשיפור.



הוצאנו את ממוצעי FTP של כל אלגוריתם על כל הdatasets.

כפי שניתן לראות, אכן קיים שוני בין קוד המקור לשתי המימושים האחרים והשינוי בין השיפור לקוד המוכר אינו מובהק. ניתן לראות שקוד המקור הניב את התוצאות הטובות ביותר לפי מדד זה והשינוי בין האלגוריתם המוכר לאלגוריתם המשופר אינו מובהק.

**תוצאות הניסוי**

**(הפקת תוצאות הניסוי לקחה לנו 5 ימים)**

מאחר שהוצאנו את מבחן פרידמן והוק והממוצעים בעזרת קוד. בחרנו להציג את התוצאות עבור כל המדדים וכך נוכל לגזור מסקנות רבות לפי כל מדד במידה ואכן יצא מובהק לפי פרידמן.

(ה Pseudo code של כל אחד מהאלגוריתמים נמצא למעלה, ומותאר בפירוט רב)

|  |  |  |  |
| --- | --- | --- | --- |
| מדד/  קטגוריה | מבחן פרידמן | מבן הוק | ממוצעים |
| Accuracy | קיים הבדל מובהק בין האלגוריתמים |  |  |
| Precision | אין הבדל מובהק בין האלגוריתמים |  |  |
| AUC | אין הבדל מובהק בין האלגוריתמים |  |  |
| FPR | אין הבדל מובהק בין האלגוריתמים |  |  |
| Training Time | קיים הבדל מובהק בין האלגוריתמים |  |  |
| Inference Time | קיים הבדל מובהק בין האלגוריתמים |  |  |
| PR-Curve | קיים הבדל מובהק בין האלגוריתמים |  |  |
| TPR | קיים הבדל מובהק בין האלגוריתמים |  |  |

**מסקנות הניסוי**

קיצורי שמות: קוד המקור – SGDR ,קוד משופר – AdamWR ,קוד של אלגוריתם מוכר - Adam

מסקנות מטבלת תוצאות הניסוי :

1. קיים הבדל משמעותי ומובהק בין מהירות הריצה של SGDR לעומתAdam ו AdamWR, **נראה ש SGDR יעיל יותר מהאלגוריתמים האחרים.**  
   (מסקנה נו נגזרה על פי התוצאות בטבלה של תוצאות הניסוי)
2. קיים הבדל משמעותי ומובהק בין היכולת לחזות true positive בין SGDR ל Adam ו AdamWR. **נראה ש SGDR מדוייק יותר בחיזוי true positive מהאלגוריתמים האחרים.** (מסקנה נו נגזרה על פי התוצאות בטבלה של תוצאות הניסוי)
3. קיים הבדל משמעותי ומובהק בין היכולת לחזות PR-Curve בין SGDR ל Adam ו AdamWR. **נראה ש SGDR מדוייק יותר בחיזוי PR-Curve מהאלגוריתמים האחרים.** (מסקנה נו נגזרה על פי התוצאות בטבלה של תוצאות הניסוי)
4. קיים הבדל משמעותי ומובהק בין היכולת לחזות Accuracy בין SGDR ל Adam ו AdamWR. **נראה ש SGDR פחות מדוייק בחיזוי Accuracy מהאלגוריתמים האחרים.** (מסקנה נו נגזרה על פי התוצאות בטבלה של תוצאות הניסוי)
5. נראה שאין הבדל מובהק בין 3 האלגוריתמים במדדים הבאים : **Precision**, **AUC,FPR**
6. נראה ש AdamWR אינו משפר את קוד המקור ובנוסף אין הבדל מובהק בינו לבין Adam, לכן **השיטה לשיפור קוד המקור שהצענו לא באמת מניבה שיפור.**

**נקודות לדיון**

* לגבי מסקנה מספר 4 ו 5 – נראה שSGDR הניב תוצאות נמוכות יותר במדד accuracy מהאלגוריתמים השונים ובנוסף בחלק מהמדדים לא היה הבדל מובהק, זאת בניגוד לנאמר במאמר.אנחנו מניחים שתופעה זאת התרחשה מאחר שמספר ה epochs היה סטטי וקטן מאוד יחסית למאמר (מספר האפוקים שלנו היה 5 בגלל אילוצי זמן**) . נאלצנו לבחור מספר אפוקים מאוד קטן**, שעלול היה לפגוע בטיב התוצאות ואינו משקף **הרצה במספר אפוקים גדול יותר, כאשר בריצה כזאת יכולות SGDR באות לידי ביטוי בצורה טובה יותר.**
* לגבי מסקנה מספר 6- בה מצאנו שהשיפור שהצענו לא הניב שיפור באלגוריתם המקור ואף לא הניב שיפור מובהק באלגוריתם המוכר, שכן הוא מבוסס על האלגוריתם המוכר ומשלב את יכולות אלגוריתם המקור. במהלך הריצות שלקחו מספר ימים סרקנו את האינטרנט עבור מימוש השיטה AdamWR וגילנו שכותבי המאמר אכן מימשנו שיטה זאת בשנת 2019 אך על מנת שתפיק תוצאות טובות הם שינו את האלגוריתם של Adam ל Adam עם Weight decay ואז הוסיפו את ה warm restarts with cosine annealing , ואכן הגיעו **לתוצאות מרשימות ביותר** בעזרת האלגוריתם המשופר , מצורף קישור למאמר עבור אלגוריתם זה <https://openreview.net/forum?id=rk6qdGgCZ>  
  לצערנו ככל הנראה כותבי המאמר מחזיקים בסט כישורים רחב יותר משלנו ובעלי זמן לא מוגבל ולכן המימוש שלהם לשיטה AdamWR היה נכון יותר משלנו.
* בנוסף ניתן לראות שהמאמר צוטט בהמון מאמרים אחרים וקיימת התרשמות גדולה ברשת מתוצאות והחידושים שהמאמר מביא עימו.

Text, letter

Description automatically generated

**דעות ומאמרים בנושא:**

1. במאמר FIXING WEIGHT DECAY REGULARIZATION IN ADAM"" מנסים להפעיל את Adam ביחד עם warm restarts. עד המאמר לא פורסם אלגוריתם שהצליח להפעיל את Adam ביחד עם warm restarts כך שהוא יצליח להתחרות בSGDR. המאמר מנסה לעשות זאת באמצעות שינוי שיטת המשקול באלגוריתם.
2. במאמר "DECOUPLED WEIGHT DECAY REGULARIZATION", מוזכר שעד 2017 SGDR היה אלגוריתם מוביל בעולם סיווג התמונה.
3. במאמר " Stochastic Gradient Descent with Hyperbolic-Tangent Decay on Classification" משווים בין SGDR לבין step decay ומציגים את היתרונות והחסרונות של כל אחד מהם. לדוגמא חוסר הגמישות של SGDR(מזינים min וmax) לעומת המהירות שלו.