**מטלה סוף קורס רשתות נוירונים**

**CNN – Classification of MRI**

**מגישים :**

**אלמוג בורה 206295115**

**אלמוג סיסו 204307516**

**פלג שובל 316485549**

# תקציר

## מקור הנתונים :

למעבר לחץ [כאן](https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri)

## המשימה:

לאבחן ולבצע קלסיפיקציה של גידולים מוחיים מסוגים שונים לפי דאטה סט של תמונות.

## התהליך שביצענו תוצאות מרכזיות:

במהלך הפרויקט לקחנו קוד מ-Kaggle שבודק שאלת קלסיפיקציה לגידול במוח על ידי סריקות תמונות מסוג MRI. בהתחלה חקרנו את שלבי הקוד, הבנו איך הוא מעבד את הנתונים עצמם ומכין אותם להרצת המודל. המודל שהשתמש בו כותב הקוד הוא EfficientNetB3 שהוא מודל מאוד חזק, אך בהתחלה התוצאות שלו היו סביב ה-80%.

במהלך החקירה שינינו את היפר-פרמטרים של המודל המקורי על מנת להגיע לדיוק גבוה יותר, הרצנו בדיקות על מעל 30 הרצות שונות, עקבנו אחר התוצאות ואם היה שינוי לטובה אימצנו את השינוי.

לאחר מכן עברנו למודלים שונים בעזרת למידת העברה, השתמשנו במודל VGG16 ובמודל MobileNetV1. גם במודלים אלו שינינו את הפרמטרים ורשמנו את התוצאות הטובות ביותר.

לבסוף, יצרנו דאטה גדול מאוד על מנת לאזן בין המחלקות השונות, הגדלנו משמעותית את נפח סט האימון שלנו והרצנו על המודל הבסיסי. קיבלנו תוצאות גבוהות משמעותית עם רמת דיוק גבוהה מהרגיל.

## רקע לבעיה

## תיאור הבעיה

גידולי מוח התגלו לראשונה במאה ה-17, אך זוהו כגידולים רק במאה ה-19. כיום, גידולי מוח הם סוג סרטן נפוץ, עם כ-300,000 מקרים חדשים מדי שנה בעולם. השכיחות השנתית עומדת על כ-22 מקרים לכל 100,000 איש, והסיכון לגידולי מוח ראשוניים לאורך החיים הוא כ-1%. שיעור התמותה הכולל מגידולים אלה עומד על כ-30%.

**סוגי גידולים נפוצים**

רוב גידולי המוח הראשוניים (כ-80%) הם ממאירים ויכולים להתפשט לרקמות סמוכות. כ-20% הם גידולים שפירים שבדרך כלל אינם מסכני חיים, אך עדיין עלולים לגרום לבעיות בהתאם למיקומם וגודלם. שני סוגים נפוצים הם:

* גליומות (Gliomas) - קבוצת גידולים ממאירים הנובעים מתאי גליה תומכים במוח.
* גידולי בלוטת יתר המוח (Pituitary Tumors) - גידולים שפירים וממאירים הגדלים בבלוטת היחידה המווסתת הורמונים חשובים.
* מננגיומות (Meningiomas) הן גידולים שפירים ושכיחים היוצאים מקרומי המוח. הם עלולים לגרום ללחץ ונזק למוח אך אינם מתפשטים ספונטנית.

**תהליך האבחנה**

תהליך גילוי גידולים מוחיים כולל מספר שלבים חשובים:

1. אבחון ראשוני - זיהוי סימפטומים כמו כאבי ראש, בחילות, התקפים ושיבושים תפקודיים.  
2. בדיקות הדמיה - סריקות CT או MRI לזיהוי הגידול, מיקומו, גודלו וצורתו.   
3. ביופסיה - לעיתים נדרשת כדי לאבחן בדיוק את סוג הגידול ולקבוע טיפול מתאים.  
4. בדיקות משלימות - בדיקות דם, הערכות נוירולוגיות ותפקודיות להבנת השפעות הגידול.

**חשיבות האבחון המוקדם והמדויק**

אבחון מוקדם של גידולים מוחיים חיוני משום שגידולים מתקדמים עלולים לגרום נזק בלתי הפיך. זיהוי מוקדם מאפשר טיפול מדויק ומשפר סיכויי החלמה. יש חשיבות רבה לזיהוי המדויק של סוג הגידול, שכן יש שונות במידת הממאירות, תגובה לטיפולים. אבחנה מדויקת מכוונת את הגישה הטיפולית המתאימה.

**שילוב למידת מכונה**

בשנים האחרונות, למידת עמוקה (deep learning) מוטמעת בתהליך האבחון באמצעות ניתוח ממוכן של תמונות הדמיה, זיהוי וסיווג גידולים ודפוסים, חיזוי תגובה לטיפול וניטור התקדמות. כלים אלו מהווים כלי עזר משלים לרופאים, אך אינם מחליפים את התפקיד האנושי הקריטי בשלב זה.   
תהליכי למידה עמוקה יכולים לזהות באופן מדויק יותר מרופאים מאפיינים מסוימים בתמונות שמצביעים על גידול ממוקם. הם עשויים לסייע בזיהוי מוקדם יותר של גידולים חשודים ובכך לשפר את בטיחות ויעילות תהליך האבחון.

## אתגרים

במהלך הפרויקט התגלנו לנו מס אתגרים:

1. **בחירת קוד מתאים:**

עם תחילת העבודה ב-Kaggle, נחשפנו למגוון רחב של קודים ומודלים שלא הכרנו, כמו גם שיטות עבודה מגוונות. לבסוף, בחרנו בקוד יחסית פשוט בעל דיוק של כ-80%, מתוך כוונה לשפרו בהמשך.

1. **הכנת נתוני תמונה:**

גילינו שתי גישות עיקריות להכנת נתוני תמונה:  
שמירה כרשימה בתוך Data Frame: כתובת התמונה נשמרת בעמודה אחת, והמחלקה אליה היא שייכת בעמודה בשם "labels". כתוצאה מכך, נוצר משתנה אחד המחזיק הן את ערכי התמונה והן את ערכי המחלקה (ראה נספחים)

המרת התמונה למטריצה: באמצעות ספריית NumPy, התמונה מומרת למספרים בגודל הפיקסלים המוגדר. תהליך זה יוצר שני משתנים נפרדים: X ו-Y, עבור סט האימון/מבחן/ולידציה (ראה נספחים). הבנו שעבודה עם פורמט מטריצה (NP) נוחה ויעילה יותר ורוב הקודים מוכנים לעבודה בפורמט הזה, אולם הכנת הנתונים של הקוד שלנו הייתה בפורמט הראשון מה שהקשה ליצור את ההתאמה בין הקודים .בנוסף פורמט זה מאפשר ביצוע קל יותר של פעולות נרמול או עיבוד נתונים, והביא לנו תוצאות טובות יותר על אותו מודל.

1. **סוג התמונות:**

בניגוד למשימות קלסיפיקציה "קלאסיות" (כגון זיהוי כלב או חתול), עבדנו עם תצלומי MRI ומטרתנו הייתה לחזות את סוג הגידול המופיע בהם. כיוון שאין לנו ידע רפואי מקצועי, לא ידענו להבדיל בין סוגי הגידולים השונים בתמונה. יצירת דאטה מחודשת גרמה לחשש שלא נדע אם המידע שאנו יוצרים איכותי מספיק, או שאף "נמחק" בטעות את הגידול, מה שיגרום למחשב לזהות את התמונה כבריאה.

1. **: Data augmentation**   
   יצרת מחלקות מאוזנות: כאשר הסתכלנו ב Data שלנו ראינו כי היא לא מאוזנת בין המחלקות ולא מאוזנת ביחס בין אימון למבחן. יצרנו קוד שרץ על כל הדאטה ומשכפל תמונות בעזרת Data augmentation על מנת לאזן בין המחלקות ולאזן בין האימון למבחן
2. **בחירת מודל מתאים:**

לאחר מחקר מקיף, הבנו שעבור משימות קלסיפיקציה של תמונות MRI מומלץ להשתמש בטכניקת סגמנטציה (segmentation). טכניקה זו מאפשרת בידוד של הגידול באמצעות משחק עם פילטרים. גילינו שזוהי משימה מורכבת יותר ממה שחשבנו בתחילה, ולכן בחרנו להתמקד במודלים מוכרים לנו. מודל נוסף מומלץ ל-MRI הוא UNET, שעובד עם סגמנטציה ולכן יעיל מאוד לבעיה מסוג זה. עם זאת, גם בשחזור מודל זה נתקלנו בקשיים ונאלצנו לוותר עליו בשלב זה.

## בעיות דומות

כאשר חקרנו על המודל שאיתו אנחנו עובדים EfficientNetB3 הרצנו בדיקות בגוגל על מנת למצוא מידע על המודל , י לא הכרנו אותו לפני. נתקלנו במאמר שנקרא:

[Image classification: ResNet vs EfficientNet vs Effici](https://medium.com/@enrico.randellini/image-classification-resnet-vs-efficientnet-vs-efficientnet-v2-vs-compact-convolutional-c205838bbf49)[entNet\_v2 vs Compact Convolutional Transformers""](https://medium.com/@enrico.randellini/image-classification-resnet-vs-efficientnet-vs-efficientnet-v2-vs-compact-convolutional-c205838bbf49)

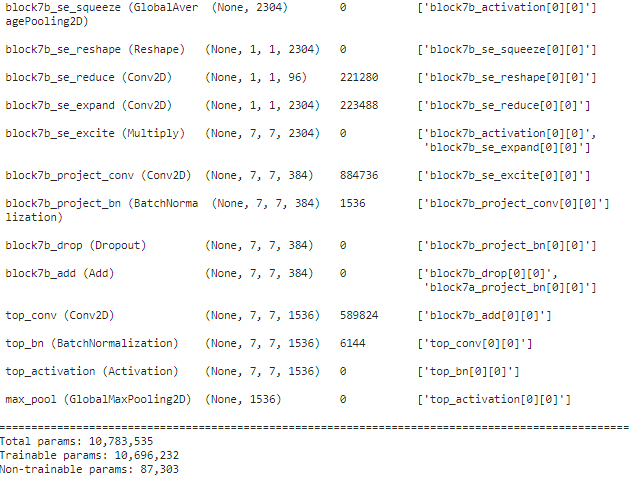
במאמר מסבירים על המודלים ועל ההבדלים בניהם, בהמשך להפתעתנו גילנו שהמאמר לצורך השוואה בין המודלים משתמש באותה דאטה סט שיש לנו ובאותה שאלת מחקר. נעזרנו במאמר לטובת קבלת רעיונות וכיוני חקירה איך לעבוד עם הדאטה סט שלנו.  
כאשר החלטנו לחולל תמונות בעזרת Data augmentation נעזרנו בפרמטרים של המאמר ועל מנת לא לפגוע באיכות התמונה יותר מידי, מה שעזר לנו להתמודד עם האתגר השלישי.

## הסבר על המודלים שהשתמשנו

**EfficientNetB3**

מודל שפותח על ידי חוקרים בגוגל בשנת 2019. הוא מורכב מ-12 שכבות שונות של מסננים וחיבורים, , שכבות נורמליזציה ופעולות לא לינאריות. שתוכננו באופן יעיל במיוחד כדי לשפר את הביצועים תוך צמצום העומס החישובי.   
מודל זה טוב במיוחד לזיהוי ולסיווג תמונות במגוון רחב של מטלות, כולל זיהוי פנים, זיהוי תנועה וזיהוי חפצים או במקרה שלנו MRI . הארכיטקטורה החדשנית שלו, המשלבת טכניקות כגון מנגנון "מסלול מדורג" מאפשר לשלב באופן יעיל מסננים מרובים בגדלים שונים, כך שהמודל יכול לתפוס תבניות ברזולוציות שונות של התמונה. היתרון המרכזי של EfficientNetB3 הוא היכולת שלו לשלב ביצועים גבוהים עם יעילות חישובית, דבר שהופך אותו לפופולרי בהרבה יישומים המצריכים זיהוי תמונות בזמן אמת על מכשירים ניידים או מוגבלי משאבים. אחד החסרונות של המודל הוא שהוא דורש כמות גדולה של נתונים לאימון יעיל, ועלול להתקשות כאשר כמות הנתונים קטנה. לרוב הוא מתאים יותר לקבוצות נתונים גדולות.

תמונה שמכילה טקסט, צילום מסך, מספר, גופן

התיאור נוצר באופן אוטומטיחלק ממבנה הארכיטקטורה של המודל

**MobileNet**

בניגוד למודלים הכבדים יותר, כמו VGG16 או EfficientNetB3. הוא מתמקד בספקטור היעילות והמהירות. המודל משתמש בטכניקות מתקדמות כדי להבטיח תוצאות מדויקות על תמונות, גם תחת תנאי חוסר משאבים. בזכות העיצוב המיוחד שלו, הוא יכול לזהות ולסווג תמונות במהירות ובדיוק, והוא מיועד ליישומים כמו זיהוי פנים, זיהוי תנועה, או סיווג תמונות מתוך מגוון רחב של מטרות.

A screenshot of a computer

Description automatically generated

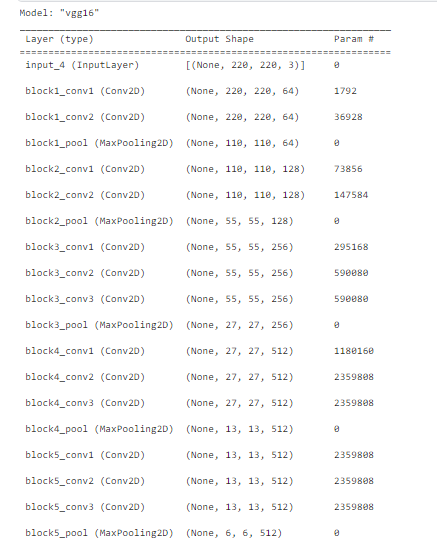
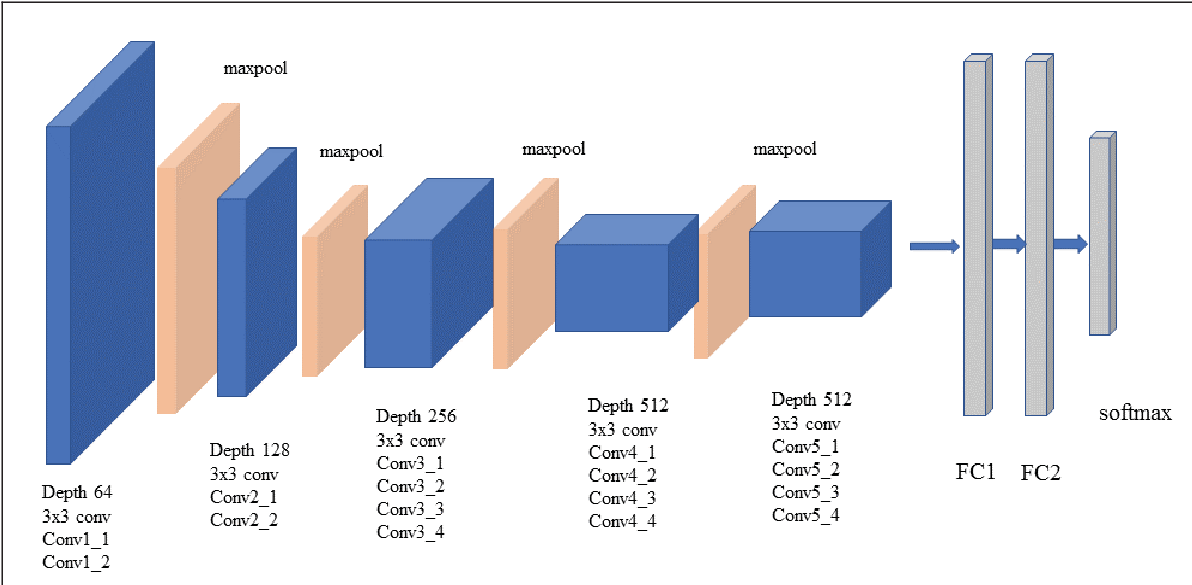
A screenshot of a computer code

Description automatically generated

**VGG16**

מכיל 16 שכבות של רשת נוירונים, כאשר השכבות מתחלקות לשכבות מסננים ושכבות "Fully" – שכבות בה כל נוירון נראה לכל הנוירונים ולכן כל קלט של נוירון בשכבה הקודמת משפיע על הנוכחית. בניית המודל התבססה על הרעיון הפשוט של קיבוץ רב רמות של מסננים קטנים (3x3) עם פעולת חישובים ביניים, המשתמש ברעיון זה ביעילות גבוהה לזיהוי תכונות מורכבות בתמונות. המודל מצריך יותר משאבים וזמן לאימון, וכמו המודל המקורי שלנו לעיתים יכול להתקשות להתמודד עם משימות מסוימות כמו זיהוי תמונות MRI עקב גודל המודל וצורת האימון שדורשת כמות גדולה של נתונים**.**

מבנה הארכיטקטורה של המודל :



# ניתוח נתונים ראשוני

קובץ הנתונים שלנו הגיע כבר מחולק לסט של אימון וסט של מבחן. כל תיקיה הייתה מחולקת ל 4 מחלוקות.

סט האימון מכיל : 2870 תמונות  
סט המבחן מכיל : 394 תמונת

ניתן לראות מתצוגת הגרף כי הדאטה אינה מאוזנת בין המחלקות גם בסט אימון וגם בסט מבחן, וכמו כן אינה מאוזנת בין היחס של האימון מבחן עצמו

**חלוקה של הנתונים**

תמונה שמכילה טקסט, תרשים, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

**תצוגה מקדימה של סט התמונות**

תמונה שמכילה הדמיה רפואית, רדיולוגיב, סרט רנטגן, רדיוגרפיה רפואית

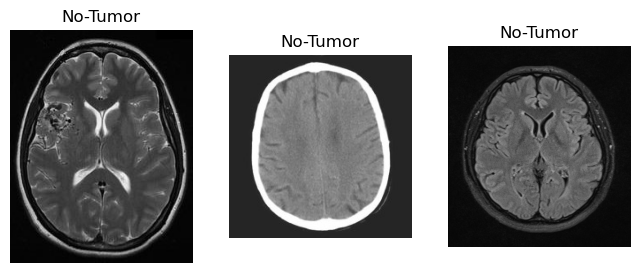
התיאור נוצר באופן אוטומטי

תמונה שמכילה הדמיה רפואית, רדיולוגיב, סרט רנטגן

התיאור נוצר באופן אוטומטי

תמונה שמכילה הדמיה רפואית, רדיולוגיב, סרט רנטגן

התיאור נוצר באופן אוטומטי



# תהליך החקירה :

הקוד [**המקורי**](https://www.kaggle.com/code/timeisfly1/brain-classification-with-transfer-learning) שנבחר נלקח מהאתר של Kaggle . הקוד נבחר מכיוון שבהתחלה ראינו כי ציון הדיוק של הוולידציה עומד על 0.786 וכי שם המודל שאיתו עובד כותב הקוד ובעזרתו ביצע למידת העברה " EfficientNetB3" אינו מוכר לנו , ולכן חשבנו כי זה הזדמנות מצוינת לנסות ולעבוד עם הקוד ולשפר אותו.

בהתחלה הרצנו את הקוד כמו שהוא בלי שינוים בכלל. והצלחנו לשחזר את התוצאות.

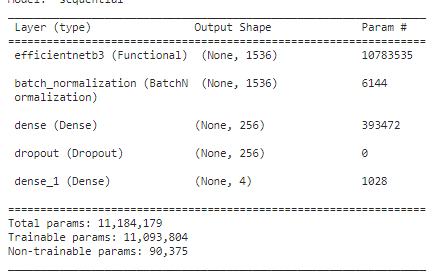
**גרף 1 - קוד בסיסי:**

**תמונה שמכילה טקסט, תרשים, עלילה, קו

התיאור נוצר באופן אוטומטי**

התוצאה הכי גבוהה:

ארכיטקטורת המודל:



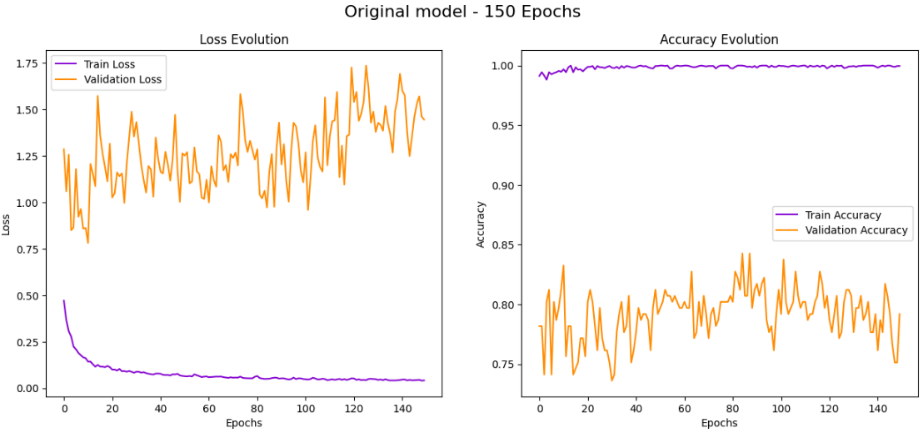
ריכזנו היפר – פרמטרים שאנחנו יודעים שעושים להשפיע על הדיוק, בנוסף לפרמטרים של כותב הקוד:

* **Epochs** – מספר הריצות על כל סט התמונות של המודל, בקוד המקורי:  **8**
* **Regularization** - טכניקות לצמצום התאמת יתר ע"י הוספת קנסות על משקלי הפרמטרים  
  L1 מטרתו "לחסל" פיצ'רים חלשים ו2L לפזר משקלים באופן שווה, בקוד המקורי: **L1 = 0.006 ,L2=0.016**
* **Batch** **Size** – גודל של קבוצת הדוגמאות שמועברות יחד דרך הרשת בכל שלב של האימון, בקוד המקורי: **15.**
* **Image** **Size** – גודל התמונה שהמודל מקבל כקלט, נמדד בפיקסלים: **220**
* **Dropout** – טכניקה להפחתת התאמת יתר ע"י השבת נוירונים באופן אקראי, בקוד המקורי: **0.4**
* **Activation** **Function** – פונקציה שמופעלת על ידי כל נוירון, בקוד המקורי: **RelU**

בהמשך חקירת המודל כל פעם שינינו את אחד הפרמטרים ובדקנו האם הוא שיפר את תוצאת המודל, אם כן רשמנו לעצמנו ושמרנו אותו. אם לא עברנו לפרמטר אחר.

**ראשית**, שמנו לב כי המודל המקורי עשה רק 8 Epochs, וראינו כי המודל עצר בעליה ולכן ביצענו ריצה עם 150 Epochs

**גרף 2 – מספר Epochs ל 150**



התוצאה הכי גבוהה:

מגרף 2 ניתן לראות כי יש שיפור ב Val accuracy עצמו לאורך הריצות אולם הוא מתחיל להישאר אותו מספר וניתן לראות התאמת יתר מאוד גדולה ולכן נבדוק אותו מודל עם 50 Epochs

**גרף 3 – מספר Epochs ל 50**

A graph of different colored lines

Description automatically generated with medium confidence

התוצאה הכי גבוהה:

ישנה ירידה קלה ביחס למודל הקודם אבל עדין יש עלייה בAccuracy ביחס למודל המקורי גם פה הבחנו בהתאמת יתר ולכן נוריד את מס' Epochs

**גרף 4 - מספר Epochs ל 25**

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

****התוצאה הכי גבוהה:

כאשר הורדנו ל 25 Epochs ראינו כי ישנה עלייה בAccuracy וכרגע זו התוצאה הטובה ביותר, ואת שאר ההרצות נרוץ על אותו מספר Epoch

שמנו לב כי המודל נמצא ב **Overfitting** ורצינו לטפל בכך על ידי הגדלת אחוז ה Dropout

A graph of different colored lines

Description automatically generated**גרף 5 - שינוי הDropout מ0.4 ל0.5**

****התוצאה הכי גבוהה:

ראינו כי ישנה ירידה ב Accuracy ולא חל שינוי ב **Overfitting** ולכן נישאר עם Dropout = 0.4

ניסנו לטפל ב Overfitting על ידי הענשה גדולה יותר בשיטת ה **Regularization** במתן קנסות גדולים יותר על 1L וגם על 2L

**גרף 6 - הגדלת 2L מ0.016 ל0.2**

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

התוצאה הכי גבוהה:****

ישנה ירידה ב Accuracy ולא חל שינוי ב **Overfitting** לכן הגדלת L2 באחוז נמוך לא מועילה לשיפור המודל

**גרף 7 – הגדלת 1L מ0.006 ל0.2**

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

****התוצאה הכי גבוהה:

ישנה ירידה ב Accuracy ולא חל שינוי ב **Overfitting** ולכן הגדלת L1 באחוז נמוך לא מועילה לשיפור המודל

**גרף 8 – הקטנת L1 מ0.006 ל0.001**

**A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated**

****התוצאה הכי גבוהה :

ישנה ירידה ב Accuracy ולא חל שינוי ב **Overfitting** ולכן הקטנת L1 באחוז נמוך לא מועילה לשיפור המודל

**גרף 9 – הגדלת 2L מ0.016 ל0.08**

**A graph of a line and a line

Description automatically generated with medium confidence**

התוצאה הכי גבוהה:

****

ישנה ירידה ב Accuracy ולא חל שינוי ב **Overfitting** ולכן הגדלת L2 ל0.08 לא מועילה לשיפור המודל

בהרצה שמתחת רצינו לראות האם הענשה גדולה תעזור לשיפור המודל או שמא תבטל המון פיצ'רים כך שהAccuracy ירד.

**גרף 10 – הגדלת 1L ו 2L ל0.8**

A screenshot of a graph

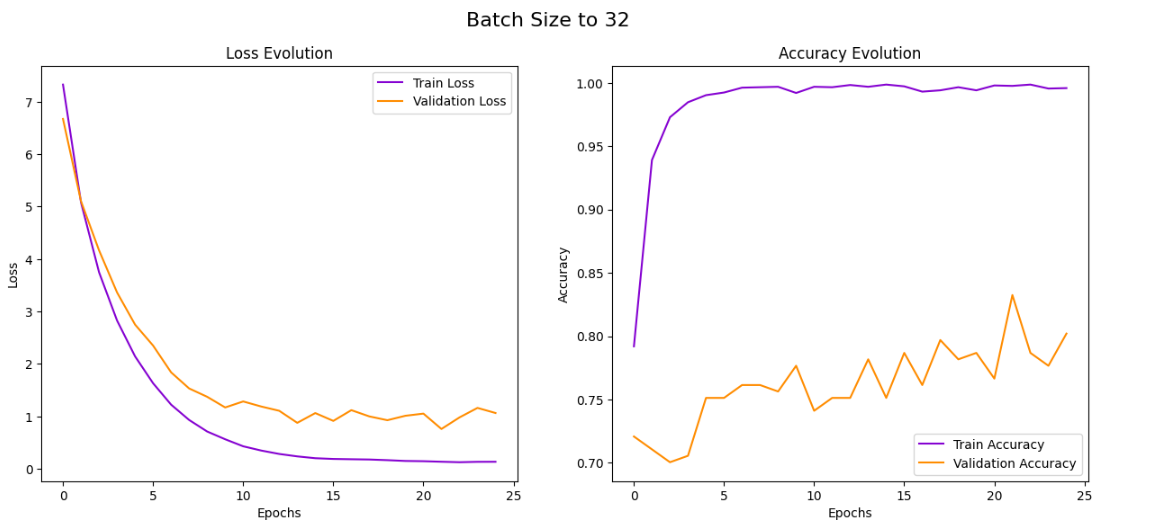
Description automatically generated

התוצאה הכי גבוהה:

כפי שציפינו לראות, קיבלנו תוצאות ממש נמוכות ולכן ניתן להבין כי הגדלה משמעותית של L1ושל L2 משפיעה על התוצאות של המודל.

החלטנו לשנות את הBatch size של המודל, אפילו שיש לנו סט אימונים קטן יחסית והקטנתו אמורה להביא לתוצאות יותר נמוכות ,רצינו לראות איך שינוי שלו ישפיע על המודל מתוך ידיע כי הדבר ישפיע משמעותית על זמן הריצה

**גרף 11 – הגדלת Batch Size ל-32**



התוצאה הכי גבוהה :



ישנה עלייה בAccuracy ביחס למודל הבסיסי אבל לא התוצאה הכי גבוהה, ראינו כי המודל נעצר בעליה ולכן ביצענו אותה ריצה עם 50 Epochs

**גרף 12 – הגדלת Batch Size ל-32 ומס' Epochs ל 50**

A graph of different colored lines

Description automatically generated with medium confidence

 התוצאה הכי גבוהה :

כעת אנו רואים כי המודל מפתח לעצמו דפוס של שיני מסור ועדין נמצא בהתאמת יתר מאוד גבוהה ולכן נחזיר את ה **Batch Size לגודלו המקורי**

ניסנו להקטין את ה Dense האחרון ל 128 ובכך "להקטין" את פתח היציאה ולראות איך זה משפיע על המודל

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated**גרף 13 – שינוי הDense ל128**

התוצאה הכי גבוהה :



ניתן לראות כי הדיוק קטן מן הדיוק המקורי וכי שינוי בגודל שכבת ה Dense לא עוזר לדיוק המודל במקרה הנ"ל.

אנו יודעים שסט האימונים שלנו מורכב מתמונות MRI ולפעמים התמונות קשות לאבחון, במהלך הכנת הדאטה אנו מכווצים את התמונות לגודל של 220 פיקסלים . לכן מתוך מחשבה על איכות התמונה העלנו את גודל הפיקסלים ל 512 בתקווה שהמודל יצליח להבחין טוב יותר בין המחלקות.

**גרף 14 – הגדלת Image Size ל512**

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

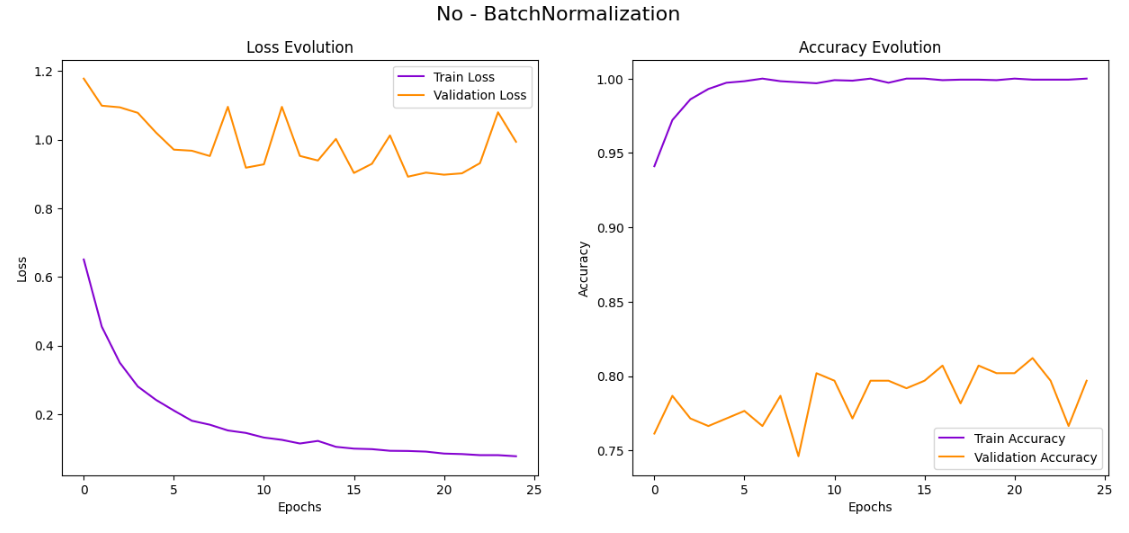
התוצאה הכי גבוהה :



ניתן לראות כי המודל לא שיפר את התוצאה שלו ביחס למודל המקורי

כפי שניתן לראות במבנה של Classifer , יש פרמטר בשם BatchNormalization שתפקידו ליצור נרמול בין השכבות השונות של המודל, לכן רצינו לראות אם הורדת הפרמטר תשפר את תוצאות המודל.

**גרף 15 – הורדה של BatchNormalization**



התוצאה הכי גבוהה:



ניתן לראות שישנה ירידה לאחר נטרול הפרמטר גורמת לרשת ולמודל להיות פחות יציבים ולאחוז דיוק נמוך יותר

רצינו לראות האם שינוי בפונקציית האקטיבציה ל TanH ישפר את הדיוק של המודל, מכיוון שהמודל שלנו הוא קלסיפיקציה של 4 מחלקות אולי יהיה לו יותר גל להבדיל בין המחלקות בעזרת TanH

**גרף 16– שינוי פונקציית האקטיבציה לTanH**

A graph of different colored lines

Description automatically generated

**** התוצאה הכי גבוהה :

ניתן לראות שישנה עלייה בAccuracy ולכן שינוי לפונקציית טנגנס היפרבולי תורמת לשיפור המודל וכי **כרגע הוא עם התוצאה הכי טובה**

**גרף 17 – שינוי פונקציית האקטיבציה לTanH והורדת הDropout ל0.1**

A graph of a function

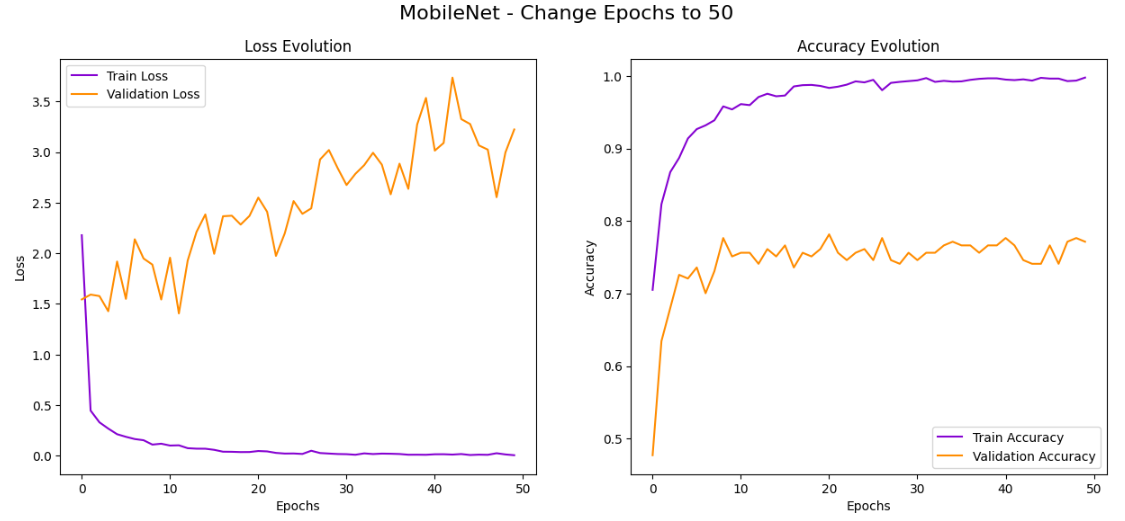
Description automatically generated with medium confidence

התוצאה הכי גבוהה :

ישנה ירידה בAccuracy ולכן הורדת הDropout לא תורמת לשיפור המודל עם פונקצייה מסוג TanH ,ניתן לשים לב שהגרף עוד בעלייה

**סיכום מודל EfficientNetB3  
לאחר שביצענו יותר מ 15 הרצות שונות על המודל . כל פעם שינינו פרמטר אחד במודל וקיבלנו תוצאות דומות יחסית בין המודלים והחלטנו לבדוק מודלים נוספים, אנו מעריכים כי תוצאות המודל זהות ומתקבעות סביב ה 82 אחוז דיוק עם התאמת יתר מאוד גבוהה הן כתוצאה מסט תמונות מאוד קטן ללמידה של המודל, ובנוסף מחלקות לא מאוזנות. את השערה זאת נבדוק בהמשך החקירה.**בהרצה הנוכחית רצינו להשתמש במודל **MobileNetV1** מכיוון שהוא דומה למודל המקורי, שניהם יודעים להבחין בין תמונות בצורה טובה כאשר המשאבים מוגבלים וחשבנו להשוות בין הביצועים של 2 המודלים

**גרף 18 – מודל** **MobileNetV1**

****

התוצאה הכי גבוהה:

****

ציפינו לראות ממודל MobilenetV1 תוצאות דומות למודל הקודם מכיוון שמודל זה פותח ע"מ להתמודד עם משאבים מוגבלים וגם נחשב מודל אמין עם איכות תוצאה טובה. ראינו שזמן הריצה נמוך יותר אך גם הדיוק נמוך יותר מהמודל המקורי

ראינו שיש 2 שיטות עיבוד הנתונים, הראשונה לשמור את הנתונים בDataframe שיש עמודה של הנתיב ועמודה של סוג המחלקה, וישנה שיטה להעביר את הנתונים לתצורה מטריציונית שבה יש שני מערכי מספרים בשם X\_train\_np, y\_train\_np בהתאמה לכל קבוצה מבחן/אימון/ולדיצה. החלטנו לעבד את הנתונים שלנו בצורה הזאת ולבדוק את המודל המקורי מבלי לשנות פרמטרים ולשמור על אותו יחס אימון מבחן כמו בקוד המקורי.

**גרף 19 – מודל EfficientNetB3 format in NP**

תמונה שמכילה טקסט, תרשים, עלילה, קו

התיאור נוצר באופן אוטומטי

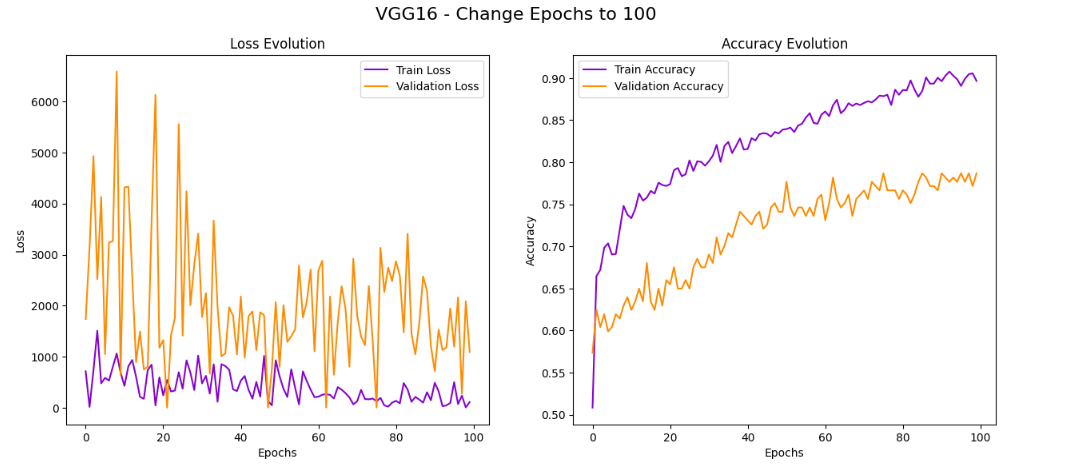
התוצאה הכי גבוהה:



כפי שניתן לראות , קיבלנו אחוז דיוק מאוד גבוהה מה שניתן להסיק כי יש הבדל בדרך שמכינים את הדאטה שלנו, וכי אולי בתמונות מסוג מסוים כמו MRI יש עדיפות להכנה בשיטה הזאות.

המשנו בחקירה , ועברנו למודל המפורסם VGG16 המודל מאופיין ב 16 שכבות עמוקות של קונבולוציה ורצינו לראות איך הוא משפיע על הסט אימון שלנו

**גרף 20 – מודל VGG16**

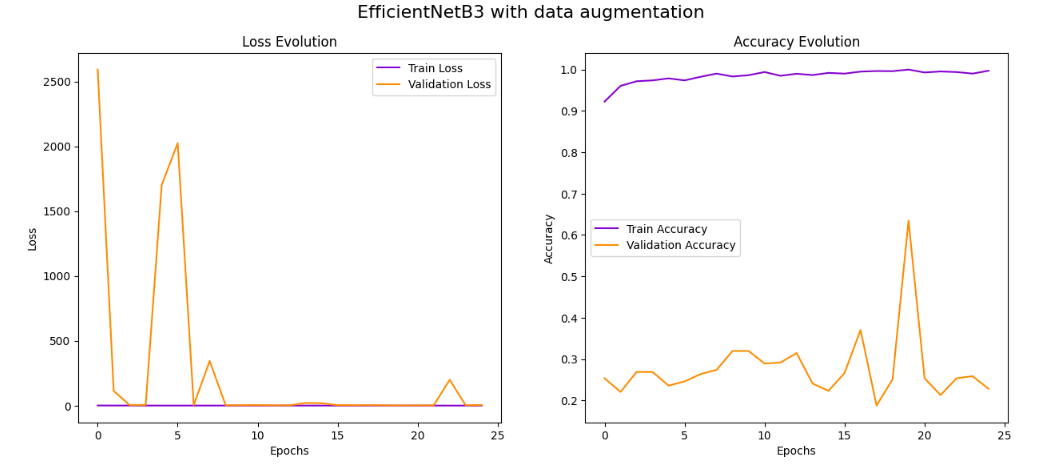


התוצאה הכי גבוהה:



בהתחלה בדקנו את המודל על 50 Epochs מכיוון שראינו שהמודל נעצר במגמת עליה ולאחר מכן העברנו ל 100 Epochs. כפי שניתן לראות ,המודל נראה שהוא במגמת עליה, אך לקראת הסוף הוא מתיישר. כמו כן אפשר לראות כי פונקציה העלות שלו מאוד לא יציבה בנוסף לעלות עצמה שמאוד מאוד גבוה, ולכן למורות שהמודל מביא לנו תוצאות קרובות למודל המקורי נאלצנו לוותר עליו.

**גרף 20 – מודל מקורי עם Data Augmentation**

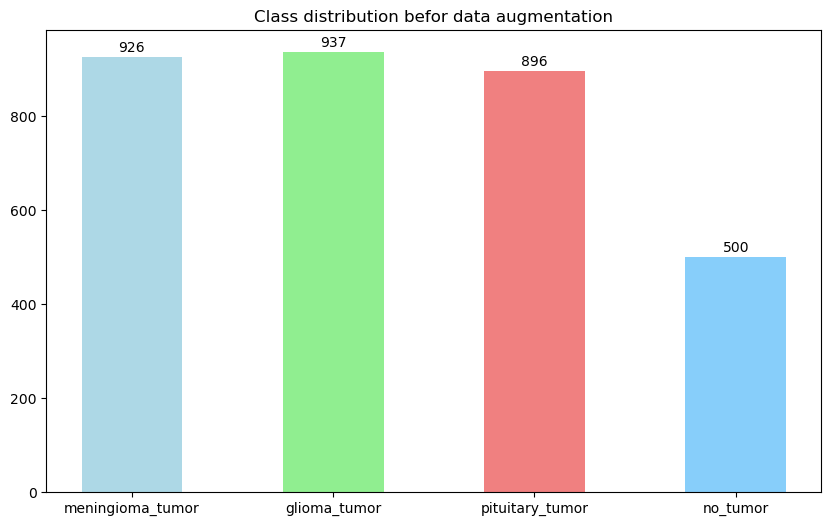
 התוצאה הכי גבוהה:

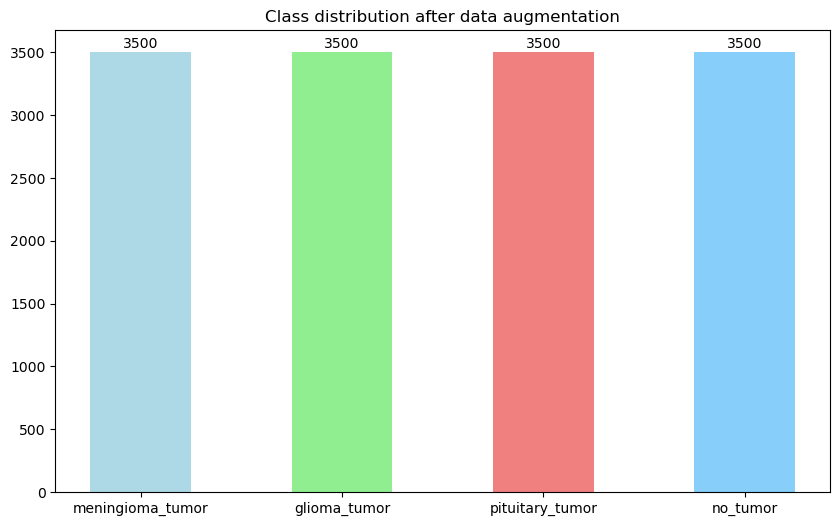


הבנו שהמודל שאנחנו חוקרים צריך לעבוד עם קבצי דאטה גדולים וראינו כי סט התמונות שלנו קטן מאוד ולכן ביצענו Data Augmentaion בתוך הקוד על מנת לשפר את תוצאות המודל. ניתן לראות כי ציון הדיוק לא השתפר כלל ולכן נאלצנו לעשות Data Augmentaion עוד לפני שלב הכנת הנתונים.

**לבסוף** סברנו כי תוצאות המודל נמוכות לא בגלל טיב המודל או ארכיטקטורה של המודל אלא בגלל סט התמונות של המודל. כפי שהצגנו בסעיף ניתוח נתונים ראשונים, ראינו כי הדאטה סט שלנו לא מאוזנת בין המחלקות.  
ראשית, איחדנו בין הסט תמונות של האימון ושל המבחן לאחר מכן יצרנו פונקציה שנכנסת לכל תיקייה ובודקת כמה תמונות יש בכל תיקיה, במידה וחסר לה כמות תמונות ממה שהגדרנו היא משלימה בעזרת Data Augmentation כמות תמונות לתיקיה, ובכך השלמנו תמונות ואיזנו בין המחלקות .כעת יש בכל מחלקה כ3500 תמונות, מה שאומר כ 14,000 תמונות בכל הדאטה סט שלנו לעומת 3,264 תמונות מהדאטה סט הקודם

**כמות דוגמאות מכל מחלקה – לפני Data Augmentation**



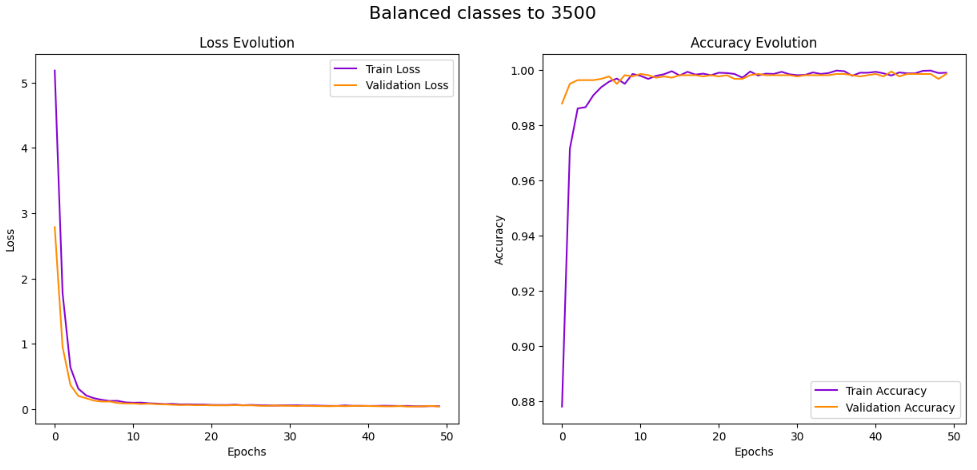
**כמות דוגמאות מכל מחלקה – אחרי Data Augmentation**

לאחר מכן חילקנו את המודל ביחס של 20/80 אימון מבחן, וסט האימון שלנו חילקנו 80/20 לאימון וולידציה ככה שיש לנו :

* אימון 8960
* מבחן 2800
* ולידציה 2240

הכנסו את הדאטה לתוך המודל המקורי שלנו EfficientNetB3 מבלי לשנות אף פרמטר והרצנו אותו עם 50 Epochs ולהלן התוצאות הנפלאות :

**גרף 19 – EfficientNetB3 עם מחלקות מאוזנות**



כפי שניתן לראות מן הגרף אחוז הדיוק עלה משמעותית בצורה מפליאה וכי הוא מתכנס סביב ה 0.998



אין ספק שזה המודל עם הציון הכי גבוה ולכן נבדוק עליו את סט המבחן שלנו:

תמונה שמכילה טקסט, צילום מסך, גופן, תצוגה

התיאור נוצר באופן אוטומטי

לאחר בדיקה על סט המבחן קיבלנו את התוצאות :  
**דיוק 99.89%  
הפסד 0.0385**

זמן לביצוע הערכה : 8.6 שניות

תמונה שמכילה טקסט, גופן, כלי, מדריך

התיאור נוצר באופן אוטומטיבנוסף לדיוק בדקנו גם את המדדים הבאים :

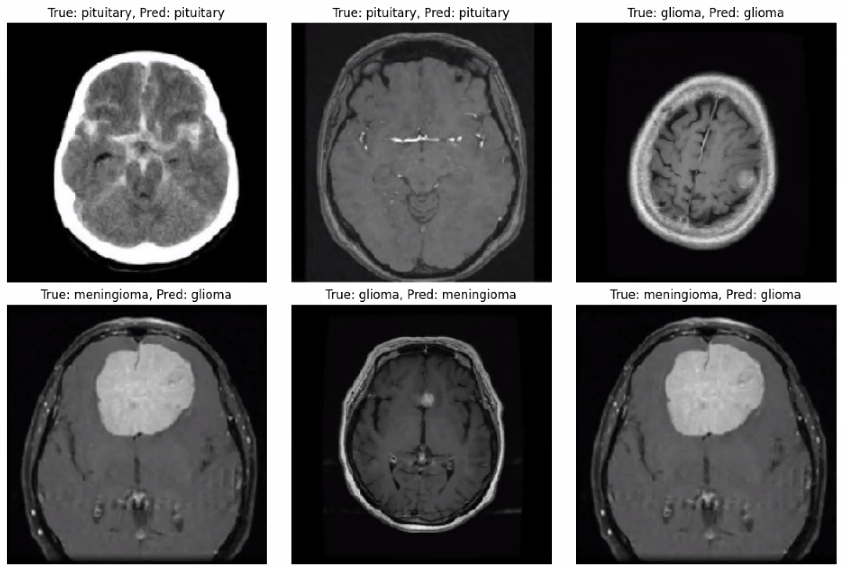
תמונה שמכילה טקסט, צילום מסך, מלבן, תרשים

התיאור נוצר באופן אוטומטיביצענו בדיקה עם confusion matrix

זמן לבדיקת התוצאות

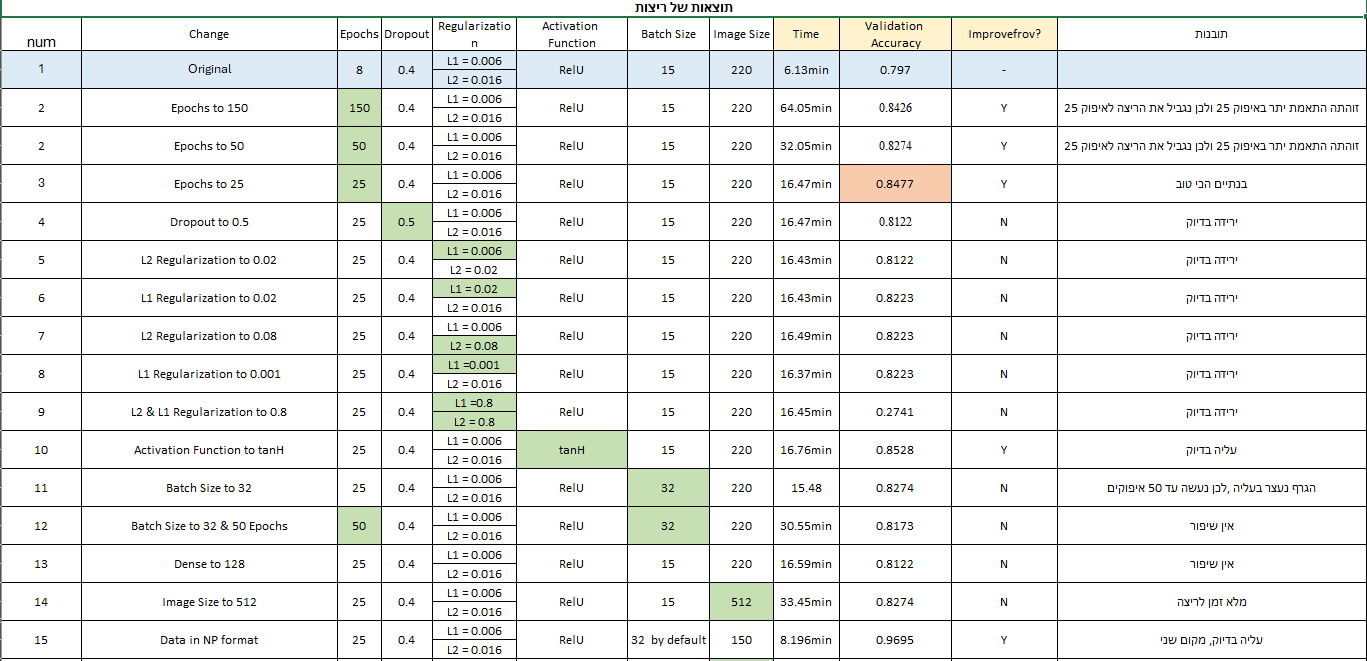


מהמטריצה למעלה ניתן לראות כי יש רק 3 תמונות שהמחשב זיהה כסוג גידול שונה מהגידול שהיה בפועל.  
כמו כן ניתן לראות כי לא היו כלל הבחנה של גידול שבפועל לא היה גידול מה שיכול לגרום לטיפולים מסוכנים, אבל חשוב עוד יותר לא היה הבחנות שקבעו "לא גידול" לתמונות שהיו בהן גידול מסוים, מה שיכול להיות מסוכן מאוד וזה הטעות שכולנו רוצים למזער כמה שיותר.

**התמונות בהן המודל זיהה נכון ולא נכון:**

**ניתן לראות התאמה בין התמונות שמוצגות פה confusion matrix.**

# תמונה שמכילה טקסט, קו, מספר, צילום מסך התיאור נוצר באופן אוטומטילהלן תוצאות של סוגי הריצות השונות



# סיכום, מסקנות ומה למדנו :

## סיכום

במהלך הפרויקט לקחנו קוד שלא הכרנו מ-Kaggle, והתחלנו לחקור אותו. את הקוד בחרנו באופן אקראי מבלי לדעת באיזה מודל משתמש כותב הקוד. המודל שבו בחר להשתמש כותב הקוד הוא EfficientNetB3 – מודל עמוק וחזק לזיהוי תמונות שפותח על ידי גוגל. בדיעבד התגלה כמודל מאוד עוצמתי עם תוצאות גבוהות מאוד.

לאחר שקראנו את הקוד, התחלנו לשנות את היפר-פרמטרים של המודל על מנת לשפר את הדיוק. ביצענו מעל 30 הרצות שונות של המודל ורשמנו את התוצאות. בכל פעם שינינו פרמטר אחד בלבד והתבוננו כיצד הוא משפיע על הביצועים. במידה והייתה השפעה חיובית, שמרנו את השינוי. במסמך זה כתבנו רק את השינויים המשמעותיים ביותר.

במהלך החקירה הבנו כי הדרך בה כותב הקוד הכין את הנתונים שונה קצת מרוב הקודים שראינו. הבנו שיש שתי דרכים עיקריות להכנת נתוני תמונות: האחת לשמור את התמונה כנתיב בצורת DataFreme, והשנייה להציג אותה כמטריצה באמצעות ספריית NumPy. כותב הקוד בחר בדרך הראשונה. בהמשך גילינו כי הדרך השנייה יותר נוחה לעבוד ואפשר להגיע איתה לתוצאות יותר טובות.

לאחר שחקרנו את המודל הבסיסי, עברנו לחקור מודלים נוספים באמצעות למידת העברה (Transfer Learning). חקרנו את המודלים MobileNetV1 ו-VGG16. גם איתם לא הגענו לתוצאות מרשימות כמו שציפנו.

לבסוף, שמנו לב שוב שהמחלקות בנתונים שלנו לא היו מאוזנות, ולכן ביצענו Data Augmentation לפני שלב עיבוד התמונות. הגדלנו משמעותית את גודל סטי האימון והמבחן שלנו. לאחר שהגדלנו את סט התמונות לסדר גודל של 14,000 תמונות (במקום 3,200) ואיזנו בין המחלקות כך שבכל מחלקה יש 3,500 תמונות, חילקנו את קובץ הנתונים ליחס 80/20 והרצנו את המודלים.

התוצאות בסוף דיברו בעד עצמן, וסוף סוף יכולנו לראות את עוצמתו של המודל EfficientNetB3. הגענו לתוצאות דיוק של 99.8%.

## מסקנות ודברים שלמדנו

* למדנו לעבוד עם Kaggle ועם מגוון קבצי נתונים
* למדנו לקרוא קודים שאנשים אחרים כתבו, גם אם הם שונים ממה שלמדנו בכיתה
* למודלים מורכבים ועמוקים כמו EfficientNet יש צורך בכמות גדולה של נתוני אימון (תמונות) כדי להגיע לביצועים מיטביים
* שיטות שונות לשלב עיבוד הנתונים לפני הכנסת המודל
* חשיבות האיזון בין המחלקות בקבוצת הנתונים על מנת למנוע הטיה
* על מנת להכניס תמונה חדשה לבחינת תוצאת המודל יש להעביר אותה כל תהליך עיבוד הנתונים מהתחלה
* יתרונות וחסרונות של טכניקות שונות להצגת נתוני תמונה

**הפרויקט חשף אותנו לאתגרים חדשים ולמדנו כיצד להתמודד עם קושי ופערי ידע באמצעות חקר עצמי ולמידה ממקורות שונים באינטרנט.**

# נספחים:

סקריפט של הקוד הטוב ביותר

ארכיטקטורת המודל

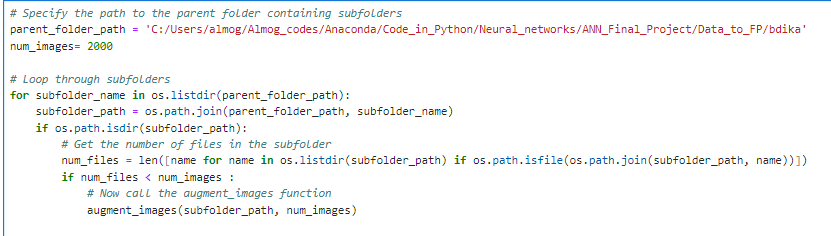
A screenshot of a computer

Description automatically generated

סקריפטים של Data Augmentation

תמונה שמכילה טקסט, צילום מסך, מסמך, גופן

התיאור נוצר באופן אוטומטי



קריאה תמונות ושמירה בדאטה פריים:



קריאה תמונות ושמירה בתצורה מטריציונית:

תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

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