

המחלקה להנדסת תוכנה

פרויקט סוף קורס יישומים ברשתות נוירונים

עמוקות – תשפ"ב

סגמנטציה מגילות ים המלח
Segmentation of Dead Sea Scrolls

מאת

שם הסטודנט/ית: נתנאל ראובן ומאור בן יאיר

ת.ז סטודנט/ית: 205463938 312597784

תקציר

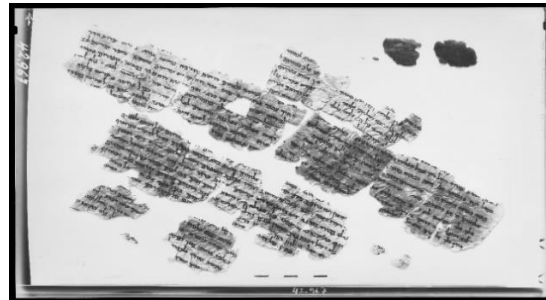
ביצענו סגמנטציה למגילות ים המלח. ערכנו השוואה בין ארבעת רשתות סגמנטציה שנמצאו כטובות ביותר עבור מטרתנו. בחנו כל אחת מהן בכדי למצוא את הפרמטרים הטובים ביותר. כאמור כל ההשוואות נעשו בסביבה הטרוגנית ובכך הבטחנו השוואה נכונה ותקינה. גילינו כי הרשת הטובה ביותר היא UNet++ והפרמטרים אשר הביאו לתוצאה זו הם Backbone RegNetY320 עם Adam(lr=0.0001) Optimizer, DiceLoss, Loss Criteria- Encoder- Decoder בעומק 4 ו-Batch Size בגודל 10. אנו משתמשים בממד ה-IOU ונבחין כי המודל קיבל ציון 88.9% שזהו ציון גבוה בבעיית הסגמנטציה מאחר ואנו מענישים עבור כל פיקסל שלא נצבע בצבע הנכון. מבחינת הערכת השגיאות המודל קיבל ציון טוב מאוד עבור זיהוי המגילות של 96.4% ועבור זיהוי הרקע 95.0%. ניתן לראות שרוב הרשתות טעו באותן המקומות שהן: מגילות בהן יש רווח קטן בין כל מגילה ומגילה, כאשר יש מגילות גדולות וכאשר ישנם רעשים שנראים כמו מגילות.

מבוא

מגילות ים המלח המכונות גם מגילות מדבר יהודה (המגילות הגנוזות ומגילות קומראן). מגילות אשר התגלו במערות קומראן, נחל חבר, במצדה שבמדבר יהודה ובאתרים נוספים באזור בין השנים 1947–1956. גילוי המגילות נחשב לאחד הממצאים הארכאולוגיים החשובים בארץ ישראל. בקורס עיבוד תמונה נחשפנו למגילות אלו דרך מוזיאון ישראל ורשות העתיקות של ישראל. כפרויקט סוף קורס התבקשנו לבצע זיהוי (לתמונות שצולמו בשנות ה-50) עבור המגילות. דוגמא:



לאחר סגמנטציה (Mask)



תמונה מקורית

כאמור, עלתה בנו המחשבה כלומדי הקורס יישומים ברשתות נוירונים עמוקות ובשל ההצלחה של רשתות נוירונים, החלטנו לנסות לבצע את המשימה הזאת בעזרת רשתות סגמנטציה. מכאן מטרתנו בפרויקט סוף קורס של יישומים ברשתות נוירונים עמוקות – סגמנטציה למגילות ים המלח. לזהות כל אובייקט בתמונה אשר מייצג חלק ממגילות ולצבוע אותה בלבן כשאר שאר הרקע יצבע בשחור לרבות, סרגל המדידה, מספרי המגילות ועוד.

עבודה קודמות

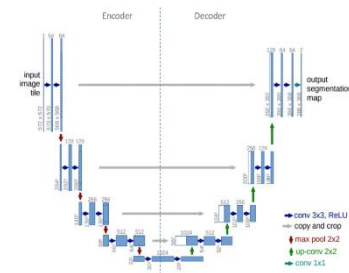
- (A1) [A Method for Segmentation, Matching and Alignment of Dead Sea Scrolls](#) •
מאמר מאוניברסיטת תל אביב המבצע התאמה בין כל אחת מחלקי המגילות בתמונות הישנות (משנות ה-50) לבין התמונות החדשות של המגילות שצולמו לאחרונה בצבע ובאיכות גובהה בהרבה. כחלק ממשימתם נאצלו לבצע סגמנטציה על המגילות הישנות.
- (A2) [UNet](#) •
- (A3) [UNet++](#) •
- (A4) [PAN](#) •
- (A5) [MANet](#) •

נתונים

הנתונים ההתחלתיים שקיבלנו היו התמונות בלבד (של חלקי המגילות אשר צלמו בשנות ה-50 ב-PAM). את ה-Data קיבלנו כחלק מפרויקט בקורס בעיבוד תמונה מרשות העתיקות של ישראל (Original_Data). בכדי לבצע סגמנטציה צריך את תוצאת הפרדיקציה הרצויה. היינו צריכים לייצר את הסגמנטציה האופטימלית באופן ידני. השתמשנו באלגוריתם Region Filling מכיוון עבור כל תמונה ואת התיקונים ביצענו באופן ידני על כל Mask ע"י Photoshop (Extra_files/BuildMask.py) יצרנו קוד שמייצר את התמונות באותו הגודל ומסובב את התמונות האופקיות ב-90° עם כיוון השעון (Extra_files/fixImage.py) חלק נכבד מהתמונות שקיבלנו הכילו חלקי מגילות קטנות והכילו כמות גדולה של חלקי מגילות כאלו. הרווחים בין חלקי המגילות היו קטנים מאוד ולכן בחרנו לאמן את המודל שלנו על תמונות ברזולוציות (Masks and DataSets) 640x360-1280x720 מאגר הנתונים שלנו קטן (107 תמונות). השתמשנו ב-Cross Validation על מנת לבחון את המודל בצורה טובה יותר (פירוט בשיטות).

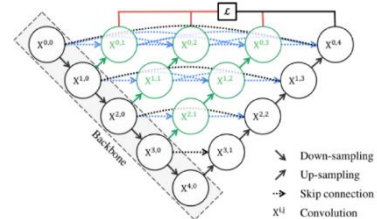
שיטות

** את הניסויים ערכנו על המחשבים הפרטיים שלנו ולכן היו הגבלות.
** בכדי שהתוצאות תהוונה השוואה נכונה בין המודלים. קיבענו את seed ל-42 ובחנו על אותן קבוצות Test-Validation.
ראשית כדי להרגיש נוח עם רשתות נזירונים עבור סגמנטציה ממשנו רשת UNet שהינה מהווה State-of-the-art accuracy. השתמשנו בספריית segmentation models pytorch. ספריה המציעה Transfer Learning למודלים של סגמנטציה, חישוב Losses מותאמים לסגמנטציה ועוד.
בחרנו 3 מודלים שלאחר העמקה בנושא נמצאו המודלים ה"טובים ביותר" לבעיית המגילות בכך שיש ביכולתם להתייחס להרבה אובייקטים בתמונה אחת. המודלים הם UNet, UNet++ ו-PAN. בנוסף בחנו גם על המודל האחרון שפורסם MANet (מתוך ספריה זו).
חילקנו את ה-Data שלנו לשלוש קבוצות – Train, Validation and Test. עבור כל Hyper Parameter אימנו מודל על קבוצת ה-Train ובדקנו את הציון על קבוצת ה-Validation, את הציון הטוב ביותר והפרמטרים שעזרו לנו לקבל אותו שמרנו ובצענו למידה חדשה על קבוצת ה-Train + Validation. לאחר מכן הצגנו את התוצאות שלנו על קבוצת ה-Test.

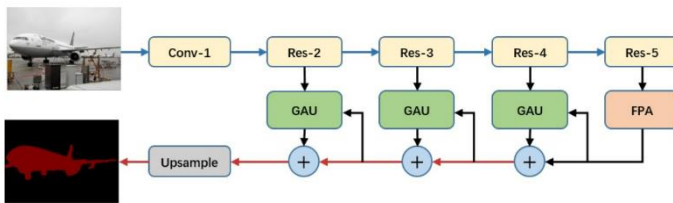


מבנה הרשת UNet

UNet++ - שיפור של UNet בכך שבמקום להעביר למידע של השלב ה-i של ה-Encoder לשלב ה-i של ה-Decoder הוא מעביר את כל הציורים של כל שכבות ה-Encoder עד לרמה ה-i ל-Decoder. ובכך משתמש ביותר מידע Up Sampling של ה-Decoder. כלומר יותר Features עבור פיקסל.



מבנה הרשת UNet++



מבנה הרשת PAN

ומעבירה ל-Decoder ה-Decoder מבצע FPA+GAU 3 פעמים ולאחר מכן מבצע Up Sampling.

MANet - מבוסס Encoder-Decoder. ה-Encoder מבצע 5 שלבים של בלוקי קונבולוציה, בכל שלב מבצע גם Down Sampling. תוצאת הקונבולוציה החמישית לאחר Down Sampling מועברת ל-Decoder. ה-Decoder מבצע בכל שלב שרשרת של תוצאת ה-Down Sampling של השלב ה-i עם תוצאת ה-Down Sampling של השלב ה-i-1. מבצע בלוק של Attention וחוזר שוב ושוב עד תוצאת ה-Attention האחרונה ועושה אינטרפולציה Bilinear.

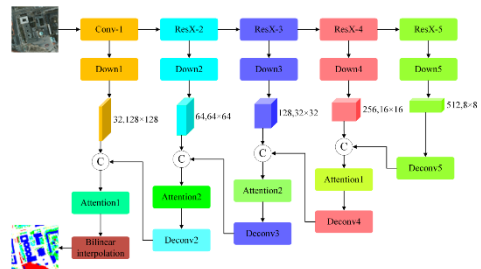


Fig. 5. The structure of the proposed MANet.

מבנה הרשת MANet

תוצאות של ניסויים

**** את כל הקודים למציאת ה-Hyper Parameters הטובים ביותר כולל הרצת הניסוי ניתן למצוא בתיקיית models.**
**** את כל תוצאות המודלים שבחנו ניתן למצוא בתיקיית Predict.**

תיאור הניסוי -

בחנו ארבעה מודלים UNet, UNet++, MANet ו-PAN. כל אחד מהמודלים בחנו עם היפר פרמטרים שונים לרבות Backbone (ארכיטקטורה שעליה מתבסס ה-Encoder), Optimizer, Batch size, גדלים שונים של תמונות ועוד. ה-Backbones שבחנו היו ResNet34, ResNet101 (לחלק מהארכיטקטורות מפאת קוצר הזמן שהיה להכנת הפרויקט), ו-RegNety320. בחרנו ב-3 המודלים האלו להיות לנו כ-Backbones מאחר ורצינו לראות השפעת ארכיטקטורה מורכבת יותר לעומת פחות. דוגמא לארכיטקטורה מורכבת - RegNetY320.

עבור מודלים אלו בחנו גם את ה-Hyper Parameters הבאים כך שבכל ריצה מספר Epochs היה קבוע ל-10:

Batch Sizes – [5, 10, 20] 640×352 [1, 2, 5] 1280×720

Loss Criteria – [DiceLoss, TverskyLoss]

Optimizer – [Adam $lr = 0.0001$, Adam $lr = 0.001$, AdaMax $lr = 0.0001$, AdaDelta $lr = 0.0001$, SGD $lr = 0.0001$, momentum = 0.9]

בנוסף עבור מודלים UNet, UNet++, ו-MANet בחנו עם שכבת Batch Normalization ו-Activation אחרי כל שכבה.

עבור UNet, UNet++, ו-MANet בדקנו גם:

Encoder Depth – [3, 4, 5]

Decoder Depth – [[64, 32, 16], [16, 32, 64, 128], [256, 128, 64, 32, 16]]

עבור PAN:

Stride – [8, 16, 32]

Decoder Depth – [16, 32, 64]

תוצאות הניסוי – (עבור 50 Epochs)

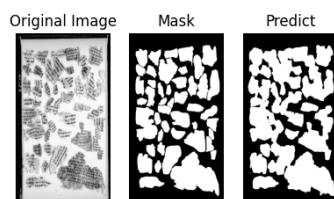
** השורה הראשונה מתייחסת לתמונות בגודל 640×352 . שורה שניה מתייחסת לתמונות בגודל 1280×720 . עבור כל שורה בטבלה.

Architecture (Backbone)	Encoder Depth	Encoder Stride	Decoder Depth	Batch Size	Loss Criteria	Optimizer	Accuracy (IOU Score)
Unet(resnet34)	4	X	4	10 5	DiceLoss	Adam(lr=0.001)	79.5% 85.6%
Unet(regnet320)	4	X	4	5	TverskyLoss	Adam(lr=0.001)	87.0% ***
Unet++(resnet34)	4	X	4	10 5	DiceLoss	Adam(lr=0.001)	84.9% 82.1%
Unet++(regnet320)	4	X	4	10	DiceLoss	Adam(lr=0.0001)	88.9% ***
PAN(resnet34)	X	16	32	20 5	DiceLoss	Adam(lr=0.001)	87.1% 87.8%
PAN(regnet320)	X	16	16	20 5	DiceLoss	Adam(lr=0.001)	87.0% 86.9%
PAN(resnet101)	X	16	32	20 5	DiceLoss	Adam(lr=0.001)	84.7% 78.1%
MANet(resnet34)	4	X	4	20 5	TverskyLoss	Adam(lr=0.001)	80.1% 87.8%
MANet(regnet320)	4	X	4	5	DiceLoss	Adam(lr=0.001)	85.9% ***

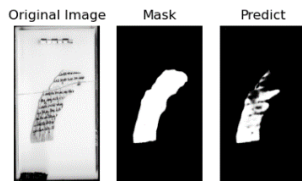
טבלה 1

*** חסרים נתונים בטבלה מפאת אורך זמן הריצות. regnet320 רשת כבדה מאוד שזמן הריצה הוא בערך 24 שעות

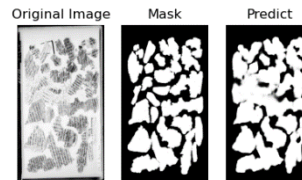
מצבי כשל נפוצים –



- עבור תמונות בהן חלקי המגילות צמודות, התגלה קושי להפריד את המגילות לחלקים אינדיבידואליים. הציג קובץ של מגילות קטנות כמגילה אחת גדולה יותר. מאחר והמרווח בין כל 2 חלקי מגילות הוא כמה פיקסלים בודדים. סטייה קטנה וזיהוי המגילות מתחבר.



- עבור תמונות בהן הייתה רק מגילה אחת גדולה, הרשת זיהתה בה חורים מאחר ורוב הדאטה היו מגילות קטנות. ביצענו ניסוי עם אוגמנטציה והתוצאות היו פחות טובות עבור כל קבוצת ה-Test אך המסכות נראו טובות יותר עבור התמונות עם חלקי המגילה הגדולות. (בדקנו רק על חלק מהרשתות מפאת קוצר הזמן).



- ישנם חלקים בתמונה בהם יש רעש שנראה כמו מגילה. הרשת זיהתה את הרעש כמגילה וזו תופעה שחזרה אצלנו בכל הרשתות שבחנו. זהו הרעש היחיד שהרשתות לא הצליחו להתעלם ממנו באופן גורף. שאר הרעשים הורדו ברוב המוחלט של הרשתות שבדקנו.

ניתוח פרמטרים -

UNet / UNet++ / MANet – שימוש ב-Batch Normalization **חשוב מאוד!** ללא השימוש ב-BN המודל מתכנס למסכות לא הגיוניות וחוזות תמונות אפורות (עבור 10 Epochs) ועבור הלמידה עם Batch Normalization מגיע למסכות הגיוניות על חלק נכבד מקבוצת ה-Validation. ככל שהארכיטקטורת ה-Encoder הייתה מורכבת יותר ככה גם **הציון היה גבוה יותר** וזה בא על חשבון זמן ריצה שהתארך. (עבור חלק מהמודלים בחנו גם את השפעת resnet101 אשר בכל הבדיקות חזרה תוצאות פחות טובות מ-resnet34). התמונות הגדולות היו עדיפות ללמידה עפ"י הציונים שלהם. ניתן לראות כי עבור כל המודלים כי גדלי Encoder-Decoder משפיעים אבל לא משמעותיים. הטוב ביותר היה בעומק 4 בכל המקרים. AdaDelta Optimizer ו-AdaMax הן Optimizers **גרועים** לבעייתנו, הטוב ביותר היה Adam. שימוש ב-Loss Criteria השפיע על הציון אבל לא באופן משמעותי והטוב ביותר היה DiceLoss. ככל שה-Batch Size **גדול יותר** ככה המודל חוזה בציון טוב יותר וכך גם למספר ה-Epochs.

ניתוח שגיאות

**** השורה הראשונה מתייחסת לתמונות בגודל 640x352. השורה שניה מתייחסת לתמונות בגודל 1280x720. עבור כל שורה בטבלה.**

$$\frac{TP}{TP + FP + FN}$$

את הציונים שהצגנו בחלק הקודם חשבנו בעזרת מדד IOU שאנו ממישנו. מדד זה מחושב

כמה צבענו בלבן וצדקנו

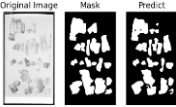

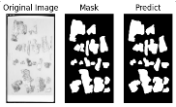
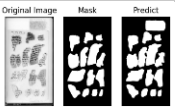
כלומר

כמה צבענו בלבן וצדקנו + כמה צבענו בלבן וטעינו + כמה צבענו בשחור וטעינו

מדד זה נותן הערכה נוקשה לשגיאה (ציונים של 80% מפיץ מסכות טובות מאוד). מדד זה נוקשה מאחר ואנחנו מענישים עבור כל פיקסל!

את השגיאה הערכנו בשני פרמטרים Background Success Rate ו-Scroll Success Rate כאשר

הם מחושבים $\frac{TP}{TP + FP}$ ו- $\frac{TN}{TN + FN}$ בהתאמה. כאמור ניתן לראות כי:

Architecture (Backbone)	Scroll Success Rate	Background Success Rate	Example of Easy Predict 640x352	Example of Hard Predict 640x352
Unet(resnet34)	86.4% 91.1%	95.1% 96.0%		
Unet(regnety320)	96.8% ***	93.4% ***		

<i>Unet++(resnet34)</i>	93.1% 87.7%	94.9% 96.1%		
<i>Unet++(regnety320)</i>	96.4% ***	95.0% ***		
<i>PAN(resnet34)</i>	92.6% 95.3%	94.6% 94.9%		
<i>PAN(regnety320)</i>	95.4% 95.1%	94.5% 94.1%		
<i>PAN(resnet101)</i>	95.6% 85.3%	94.2% 94.7%		
<i>MANet(resnet34)</i>	96.1% 94.3%	88.1% 95.4%		
<i>MANet(regnety320)</i>	96.5% ***	90.1% ***		

טבלה 2

מסקנות

מטבלה 1 ניתן לראות כי המודל שקיבל את הציון הטוב ביותר הוא 88.9% והוא התקבל מ-UNet++ עם Backbone כ-RegNetY320 המשתמש ב-141 מיליון פרמטרים. למרות הציונים הטובים שמודל זה קיבל הן בהערכת שגיאות והן כתוצאה על קבוצת ה-Test ניתן לראות כי עדיין נכשל באותן המקומות בהן כל המודלים נכשלו (פירוט במצבי כשל נפוצים). ראשית הכנת הדאטה היוו מכשול עבור הפרויקט. המסכות שיצרנו הן אינן הטובות ביותר שניתן לייצר עבור בעיה זו, כאמור, אם המסכות ישתפרו אנחנו מאמינים כי התוצאות יהיו טובות יותר מאחר ואין "רעש" במסכות האופטימליות. מהתוצאות שהתקבלו ניתן לראות כי ככל שה-Batch Size יהיה גדול יותר נקבל תוצאות טובות יותר. בחנו את התוצאות על המחשבים האישיים שלנו שהיו מגבלה ולא יכלנו לבדוק Batch Size גדולים יותר. ככל הנראה עם כוח חישוב גדול יותר ניתן להגיע לתוצאות טובות יותר. רוב הרשתות שבחנו היו צריכות הרבה פחות מ-50 Epochs כדי להגיע לתוצאה קרובה מאוד לתוצאה שהתקבלה ע"י 50 Epochs. לדוגמא PAN עם ResNet34 עם 20 Epochs הגיעה ל~85%. התמונות הגיעו ברזולוציה גבוהה. וניתן לראות שרוב המודלים שאומנו על התמונות הגדולות יותר קיבלו גם ציונים טובים יותר. וזה מאחר וישנם תמונות בהן הרווח בין המגילות קטן מאוד.

מצורף תיקיית Zip המכילה:

- Original Data – מכיל את כל תמונות המקור שקיבלנו.
- DataSets – שמכיל 2 תיקיות דאטה בגדלים שונים.
- MaskSets – שמכיל 2 תיקיות מסכות בגדלים שונים.
- Extra Files – בהם הקודים שהשתמשנו על מנת לסדר את הדאטה ולייצר מסכות ראשוניות.
- Models – ובה כל הקודים של מודלים שבחנו ותהליך האימון.
- Predict – תיקייה ובה כל תוצאות כל המודלים שהתקבלו לאחר חיפוש ה-Hyper Parameters הטובים ביותר.