שימוש ב GAN ליצירת תמונה פוטוריאליסטית מתמונה סמנטית

איתי בר – 325839710

לידי שרית לולב

אפריל 2022

ליאו באק

LEO BAECK

Learning | Community | Leadership

תוכן עניינים

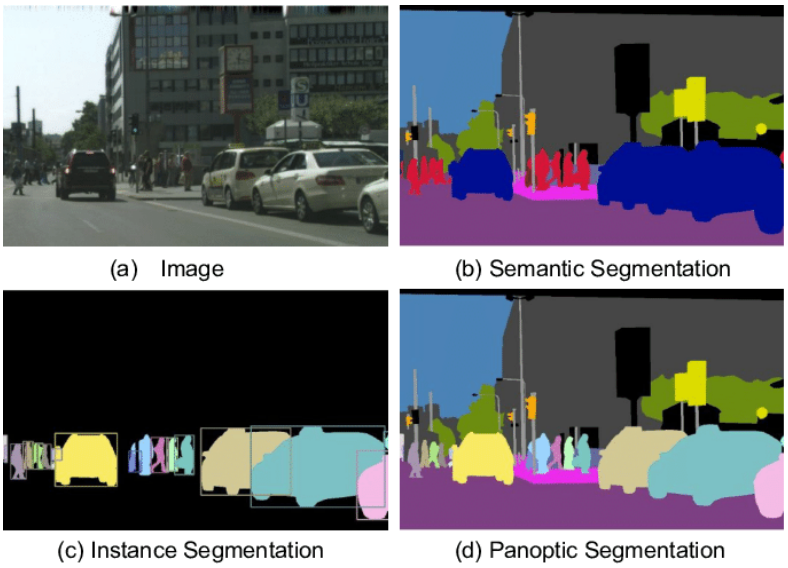
מבוא

מטרת הפרוייקט היא ליצור תמונה פוטוריאליסטית מתוך תמונה סמנטית.

כיום, הכיוון השני של המשימה שהוא ליצור תמונה סמנטית מתוך תמונה פוטוריאליסטית יחסית מוכר ויש מחקרים רבים ומגוונים בנושא. לעומת זאת, עבור מה שאני רוצה לעשות קיימים מספר מחקרים אשר מימשו את המשימה[1], אך לדעתי הם עדיין מעטים ויש מקום לשיפור ולכן הייתי רוצה לנסות להמשיך את עבודתם ולנסות גישות חדשות ומעניינות נוספות.

ישנם שלושה דרכים שונות לסמן את התמונות[2]:

1. Semantic Segmentation – כל פיקסל בתמונה מקבל סיווג למחלקה אליה הוא משתייך, בלי קשר למספר האובייקטים מאותה המחלקה שנמצאים בתמונה. יסווג למשל כרכב, עץ, כביש, בן אדם.
2. Instance Segmentation – כל פיקסל בתמונה מקבל סיווג עבור האובייקט המסויים אליו הוא משתייך. למשל אם קיימים בתמונה מספר רכבים, אז הפיקסל יכול להיות מסווג כרכב 1, רכב 2, וכו'.
3. Panoptic Segmentation – שילוב של שתי הגישות הקודמות. כל פיקסל מקבל גם סיווג של המחלקה אליה הוא משתייך, וגם סיווג אל האובייקט הספציפי.



תמונה הממחישה את ההבדל בין הדרכים לסימון תמונות

הייתי רוצה לנסות לממש את הפרוייקט שלי על תמונות שמסווגות באופן פנופטי, משום שלדעתי כך אני אקבל את התוצאות הטובות ביותר. במחקר (Wang, 2017)[1], השתמשו בתמונות סמנטיות רגילות, והם קיבלו סימני מריחה בין רכבים. הם הראו שכאשר הם מספקים למודל גם את הגבולות של הרכבים, מקבלים תמונות איכותיות יותר.

[1] High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. Retrieved from arXiv: https://doi.org/10.48550/arXiv.1711.11585

[2] *Semantic vs Instance vs Panoptic: Which Image Segmentation Technique To Choose*. Retrieved from analyticsindiamag: https://analyticsindiamag.com/semantic-vs-instance-vs-panoptic-which-image-segmentation-technique-to-choose/

בנוסף לכך, קיימים מחקרים ומודלים קודמים אשר יודעים ליצור תמונה פנופטית מתוך תמונה רגילה באיכות גבוהה, למשל EfficientPS[3].

בחרתי בנושא זה מכיוון שבאופן כללי סיקרן אותי הנושא של GAN-ים והאפשרויות הטמונות בהן. בין היתר, נחשפתי לכל מיני מודלים שונים שהצליחו לעשות דברים מדהימים, כמו Dall-e2 שמסוגל ליצור תמונות איכותיות מאוד מתוך טקסט. החשיפה לפרוייקטים שונים פורצים בתחומם במיוחד בתחום זה, הותירה בי רושם ומוטיבציה ליצירת פרוייקט משלי בתחום על מנת להגיע לתוצאות מרשימות.

ספציפית בחרתי ברעיון לקיחת סימונים והפיכתם לתמונות אמיתיות מפני שהמשימה ההפוכה לכך הינה בעיה מוכרת ועל כן קיימים מאגרי מידע רבים ואיכותיים שאוכל להשתמש בהם גם עבור המשימה שלי. בנוסף מצאתי שקיימים מחקרים ומודלים שניסו לעשות זאת, אך הרגשתי שהבעיה לא מוצתה מפני שהתמונות לא היו מספיק איכותיות כפי שהיה אפשר לצפות, כלומר יהיו לי חומרים שיוכלו לעזור לי בכתיבת הפרוייקט מצד אחד אך עדיין יהיה לי מספיק מקום למחקר משלי במטרה לשפר ביצועים מצד שני.

תהליך הלמידה והמחקר

כחלק מהלמידה לקראת הפרוייקט, השלמתי את הקורס הראשון והשני מתוך ההתמחות של DeepLearning.ai באתר קורסרה: Generative Adversarial Networks (GANs) Specialization.

הקורס הראשון עסק בעקרונות ובבסיס של GAN-ים, כמו פונקציית המחיר השימושית Wasserstein Loss, בנוסף ל Conditional GAN ו- Controllable Generation.

הקורס השני הרחיב על הדרכים שבהן ניתן להעריך את האיכות של המודל, בנוסף לדרכים נוספות לשפר את ה GAN-ים.

בנוסף לכך, קראתי מחקרים נוספים באינטרנט הקשורים ל GAN-ים, ובפרט לנושא הספציפי של הפרוייקט.

מה זה GAN (Generative Adversarial Network)?

GAN הוא סוג של מודל רגנרטיבי ללמידת מכונה.

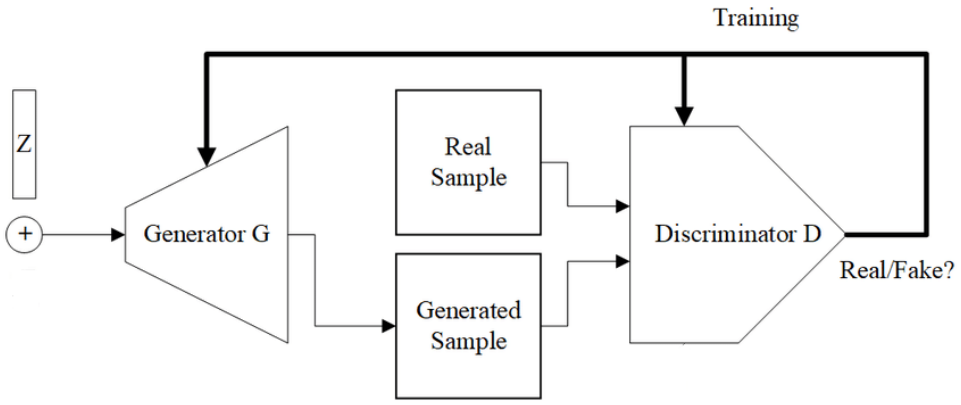
בהינתן סט אימון, ניתן להשתמש ב GAN בשביל ללמוד את ההתפלגות שממנה מגיעים הנתונים, וכתוצאה מכך ניתן להשתמש בו בשביל ליצור דגימות חדשות שנראות כמו דגימות ששייכות למאגר הנתונים. לדוגמה, GAN שאומן על מאגר תצלומים של בני אדם, יוכל ליצור תצלומים חדשים שנראים אותנטיים ומכילים מאפיינים מציאותיים רבים.

GAN מורכב משני חלקים מרכזיים: רשת discriminator (מאבחן) ורשת generator (מחולל). המחולל יוצר דגימות מזויפות והמאבחן מסווג האם הן מזויפות או אמיתיות.

המאבחן, מטרתו לסווג תמונות האם הן אמיתיות ונדגמו מתוך התפלגות מדגם האימון או שהן מזויפות, כלומר נוצרו על ידי המחולל.

המחולל, מטרתו לייצר תמונות מזויפות שנראות אמיתיות ככל הניתן. על מנת שלא ייצר את אותה תמונה כל פעם, המחולל מקבל כקלט וקטור רעש Z, אותו ניצור באופן רנדומלי. רשת המחולל אמורה ללמוד למפות בין התפלגות וקטור הרעש לבין התפלגות המידע הנכון וכך להוציא כפלט דוגמאות שנדגמו מאותה ההתפלגות של המידע.

[3] *EfficientPS: Efficient Panoptic Segmentation*. Retrieved from arXiv: https://arxiv.org/pdf/2004.02307v3.pdf



ארכיטקטורה כללית של GAN

אתגרים מרכזיים

במהלך העבודה על הפרוייקט נתקלתי במספר אתגרים מרכזיים שאיתם נאלצתי להתמודד.

1. זמן ריצה ארוך ויקר של המודל – מודלים גנרטיביים כדוגמת GAN הם לרוב מודלים גדולים ומורכבים אשר דורשים משאבים מרובים וזמן ריצה ארוך.
2. התאמה של המחקר עליו התבססתי לריצה כמחברת – קוד המקור פותח על מנת לרוץ כסקריפט על המחשב האישי. כחלק מהפרוייקט, לקחתי את קוד המקור ושיניתי אותו כך שהחלקים של אימון והרצה של המודל יהיו באותה מחברת שתייבא את המחלקות והקבצים הנוספים הנחוצים מקוד המקור ומקבצי פייתון נוספים שאני יצרתי. דרך זו מאפשרת קונפיגורציה נוחה וברורה יותר של המודל.
3. ייצוג תמונות פנופטיות כקלט עבור המודל – כאשר נרצה להעביר למודל קלט של תמונה סמנטית, נוכל ליצור mask עבור כל מחלקה סמנטית (כביש, שמיים וכו') בנפרד, כלומר מטריצה של אפסים בגודל התמונה כך שעבור כל פיקסל ששייך למחלקה הסמנטית הספציפית יופיע ב 1. mask שיטה זו מסתבכת כאשר נרצה להפריד גם בין מופעים שונים של אותה מחלקה סמנטית. פירוט בהמשך.
4. עבודה עם חבילת pytorch שלא הכרתי לפני כן - במהלך השנה, למדנו ליצור ולאמן מודלים באמצעות tensorflow ו keras, אך קוד המקור כמו גם מרבית החומרים שנחשפתי אליהם (בין היתר הקורס מקורסרה), משתמשים ב pytorch. על כן, נאלצתי ללמוד ולהכיר את החבילה תוך כדי העבודה על הפרוייקט.
5.

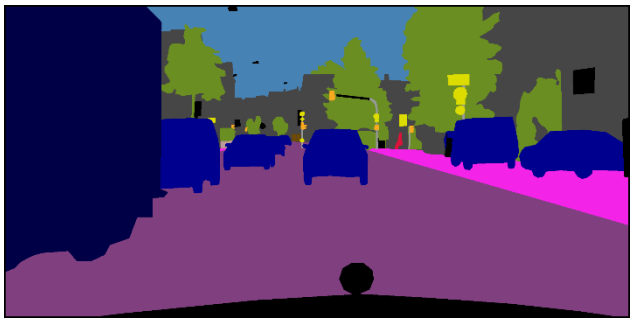
מבנה/ארכיטקטורה

איסוף, הכנה וניתוח הנתונים

מבנה הנתונים בו בחרתי להשתמש נקרא cityscapes[4].

זהו מבנה נתונים מוכר ופופולארי המשמש בעיקר למשימות סגמנטציה, ובין היתר מכיל 5000 תמונות שצולמו מתוך רכב ברחובות של 50 ערים שונות בגרמניה.

leftImage8bit/train/sttugart/stuttgart_000001_000019_leftImg8bit.png



gtFine/train/sttutgart/stuttgart_000001_000019_gtFine_color.png

על מנת להוריד ולהשתמש במבנה הנתונים יש צורך בהירשמות באתר.

מבנה הנתונים מגיע גם עם חבילה הכוללת סקריפטים המשמשים להורדה ולניתוח הנתונים. ניתן למצוא את קוד המקור של החבילה בגיטהאב: https://github.com/mcordts/cityscapesScripts

ניתן להתקין את החבילה בבאמצעות pip:

```
1.  ! pip install cityscapesscripts
```

לאחר שמתקינים את החבילה, ניתן להוריד את מבנה הנתונים כך:

```
1.  download_dataset = True
2.  if download_dataset:
3.    cityscapes_dir = root_project_directory + "/datasets/cityscapes"     #destination for
    the dataset
4.    ! csDownload leftImg8bit_trainvaltest.zip -d {cityscapes_dir}
5.    ! csDownload gtFine_trainvaltest.zip -d {cityscapes_dir}
6.    ! unzip -q {cityscapes_dir + "/gtFine_trainvaltest.zip"} -d {cityscapes_dir}
7.    ! unzip -q {cityscapes_dir + "/leftImg8bit_trainvaltest.zip"} -d {cityscapes_dir}
8.
```

התיקייה leftImg8bit מכילה את כל התמונות האמיתיות כאשר היא כוללת חלוקה ל train|val|test ולערים השונות. התיקייה gtFine מכילה את כל הסימונים המדוייקים עבור התמונות וגם היא מחולקת ל train|val|test ולערים השונות. התיקייה מגיעה עם 3 סימונים שונים עבור כל תמונה:

- *_color.png – זוהי תמונה המראה את המחלקות הסמנטיות באופן צבעוני (כפי שניתן לראות בתמונה הימנית שבראש הדף)

[4] Cordts, M. a. (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding: https://www.cityscapes-dataset.com/

- *_labelIds.png – זוהי תמונה המייצגת מטריצה שמכילה עבור כל פיקסל את הקידוד המתאים עבור המחלקה הסמנטית שלו. הקידוד עבור כל מחלקה סמנטית מפורט מטה.
- *_instanceIds.png – זוהי תמונה המייצגת מטריצה שמכילה עבור כל פיקסל ששייך למחלקה סמנטית שיכולים להיות ממנה מספר עצמים שונים (למשל רכב, בן אדם), את מספר האובייקט הספציפי אליו הוא שייך.
- *_polygons.json – קובץ json המכיל פוליגונים שמייצגים את המחלקות הסמנטיות השונות בתמונה.

בסופו של דבר, מבנה הנתונים אמור להיראות כך:

```
cityscapes
+-- gtFine
|   |
|   +-- train
|   |   |
|   |   +-- aachen
|   |       |
|   |       +-- *_color.png
|   |       +-- *_instanceIds.png
|   |       +-- *_labelIds.png
|   |       +-- *_polygons.json
|   |   ...
|
|   +-- val
|   +-- test
|
+-- leftImg8bit
     |
     +-- train
     +-- val
     +-- test
```

במבנה הנתונים קיימים סימונים ל 34 מחלקות סמנטיות שונות.
הקידוד (id) עבור כל מחלקה סמנטית נתון בקובץ labels.py אשר בחבילה cityscapesscripts, יחד עם המידע על צבע המחלקה, האם יכולים להיות ממנה עצמים מרובים ועוד הגדרות נוספות.

```
labels = [
    #       name                     id    trainId   category            catId     hasInstances   ignoreInEval   color
    Label(  'unlabeled'            ,  0 ,      255 , 'void'            , 0       , False        , True         , (  0,  0,  0) ),
    Label(  'ego vehicle'          ,  1 ,      255 , 'void'            , 0       , False        , True         , (  0,  0,  0) ),
    Label(  'rectification border' ,  2 ,      255 , 'void'            , 0       , False        , True         , (  0,  0,  0) ),
    Label(  'out of roi'           ,  3 ,      255 , 'void'            , 0       , False        , True         , (  0,  0,  0) ),
    Label(  'static'               ,  4 ,      255 , 'void'            , 0       , False        , True         , (  0,  0,  0) ),
    Label(  'dynamic'              ,  5 ,      255 , 'void'            , 0       , False        , True         , (111, 74,  0) ),
    Label(  'ground'               ,  6 ,      255 , 'void'            , 0       , False        , True         , ( 81,  0, 81) ),
    Label(  'road'                 ,  7 ,        0 , 'flat'            , 1       , False        , False        , (128, 64,128) ),
    Label(  'sidewalk'             ,  8 ,        1 , 'flat'            , 1       , False        , False        , (244, 35,232) ),
    Label(  'parking'              ,  9 ,      255 , 'flat'            , 1       , False        , True         , (250,170,160) ),
    Label(  'rail track'           , 10 ,      255 , 'flat'            , 1       , False        , True         , (230,150,140) ),
    Label(  'building'             , 11 ,        2 , 'construction'    , 2       , False        , False        , ( 70, 70, 70) ),
    Label(  'wall'                 , 12 ,        3 , 'construction'    , 2       , False        , False        , (102,102,156) ),
    Label(  'fence'                , 13 ,        4 , 'construction'    , 2       , False        , False        , (190,153,153) ),
    Label(  'guard rail'           , 14 ,      255 , 'construction'    , 2       , False        , True         , (180,165,180) ),
    Label(  'bridge'               , 15 ,      255 , 'construction'    , 2       , False        , True         , (150,100,100) ),
    Label(  'tunnel'               , 16 ,      255 , 'construction'    , 2       , False        , True         , (150,120, 90) ),
    Label(  'pole'                 , 17 ,        5 , 'object'          , 3       , False        , False        , (153,153,153) ),
    Label(  'polegroup'            , 18 ,      255 , 'object'          , 3       , False        , True         , (153,153,153) ),
    Label(  'traffic light'        , 19 ,        6 , 'object'          , 3       , False        , False        , (250,170, 30) ),
    Label(  'traffic sign'         , 20 ,        7 , 'object'          , 3       , False        , False        , (220,220,  0) ),
    Label(  'vegetation'           , 21 ,        8 , 'nature'          , 4       , False        , False        , (107,142, 35) ),
    Label(  'terrain'              , 22 ,        9 , 'nature'          , 4       , False        , False        , (152,251,152) ),
    Label(  'sky'                  , 23 ,       10 , 'sky'             , 5       , False        , False        , ( 70,130,180) ),
    Label(  'person'               , 24 ,       11 , 'human'           , 6       , True         , False        , (220, 20, 60) ),
    Label(  'rider'                , 25 ,       12 , 'human'           , 6       , True         , False        , (255,  0,  0) ),
    Label(  'car'                  , 26 ,       13 , 'vehicle'         , 7       , True         , False        , (  0,  0,142) ),
    Label(  'truck'                , 27 ,       14 , 'vehicle'         , 7       , True         , False        , (  0,  0, 70) ),
    Label(  'bus'                  , 28 ,       15 , 'vehicle'         , 7       , True         , False        , (  0, 60,100) ),
    Label(  'caravan'              , 29 ,      255 , 'vehicle'         , 7       , True         , True         , (  0,  0, 90) ),
    Label(  'trailer'              , 30 ,      255 , 'vehicle'         , 7       , True         , True         , (  0,  0,110) ),
    Label(  'train'                , 31 ,       16 , 'vehicle'         , 7       , True         , False        , (  0, 80,100) ),
    Label(  'motorcycle'           , 32 ,       17 , 'vehicle'         , 7       , True         , False        , (  0,  0,230) ),
    Label(  'bicycle'              , 33 ,       18 , 'vehicle'         , 7       , True         , False        , (119, 11, 32) ),
    Label(  'license plate'        , -1 ,       -1 , 'vehicle'         , 7       , False        , True         , (  0,  0,142) ),
]
```

החבילה cityscapesscripts מאפשרת בנוסף לפתוח את האפליקציה שבעזרתה ערכו את הסימונים עבור התמונות. באפליקציה זו ניתן לעבור על תמונות מבנה הנתונים ולשנות את הסימונים עבורן.

הנתונים הגולמיים נטענים למחברת באמצעות הפונקציה get_dataloaders שנמצאת בקובץ my_dataloaders.py. קטעי קוד אלה נלקחו מקוד המקור של OASIS אך שונו על מנת להתאים ליצירה של תמונות פנופטיות. ניתן למצוא את תוכן הקובץ בחלק המדריך למפתח.

הפונקציה get_dataloaders יוצרת שתי מחלקות מבני נתונים עבור דוגמאות האימון ועבור דוגמאות הבדיקה. לאחר מכן, הפונקציה יוצרת DataLoader עבור כל אחד ממבני הנתונים האלו. DataLoader היא מחלקה מובנית בחבילת pytorch המאפשרת להעביר למודל נתונים מתוך מבנה הנתונים תוך חלוקה לbatch-ים ועוד אפשרויות.

המחלקה CityscapesDataset משמשת על מנת לאסוף את כל הנתונים ולהוציא אותם לאחר הפיכתם לטנסורים. ראשית, היא מחפשת ושומרת את כל הכתובות של הדוגמאות בזיכרון. לאחר מכן, כאשר נרצה להוציא דוגמא ממבנה הנתונים, תיקרא הפונקציה __getitem__ אשר תפתח את התמונות, תהפוך את הסימון לפנופטי במקרה הצורך, ותפעיל את הפונקציה transform על הדוגמאות. הפונקציה transform תשנה את גודל התמונות ל 256x512, תהפוך את הדוגמאות אופקית חצי מהפעמים (data augmentation), תמיר את התמונות לטנסורים ותנרמל את התמונה.

נשים לב שטעינת מבנה הנתונים מתאימה גם עבור סימונים סמנטיים בלבד וגם עבור סימונים פנופטיים.
את הקידוד של הסימונים הפנופטיים יוצרים בשיטה הבאה:

```
panoptic ID = semantic ID * 1000 + instance ID
```

לבסוף, לפני הכניסה למודל אנו מפעילים על הטנסורים את הפונקציה preprocess_input, המקבלת טנסור מגודל [batch_size,1,256,512] המכיל עבור כל פיקסל את קידוד המחלקה הסמנטית המתאים לו, ומייצרת טנסור מגודל [batch_size, 35, 256, 512], כאשר במבנה הנתונים cityscapes קיימים 35 מחלקות סמנטיות שונות, כולל "unknown". למעשה, הפונקציה מייצרת עבור כל מחלקה סמנטית מעין mask, שהוא מטריצה בגודל התמונה של אפסים, חוץ מהפיקסלים השייכים לאותה מחלקה סמנטית בהם יסומן 1. דבר זה נעשה באמצעות הפונקציה torch.scatter_().

כאשר מדובר בסימונים פנופטיים, עשיתי שהפונקציה תשים את ה instance ID של העצם במקום 1 בתוך ה mask כשמדובר במחלקה סמנטית שיש ממנה עצמים מרובים. שיטה זו אינה עבדה לי, והמודל לא הצליח ללמוד את הקלט הזה. להבא, הייתי מנסה לעשות כך שלכל עצם בנפרד יהיה mask משלו.

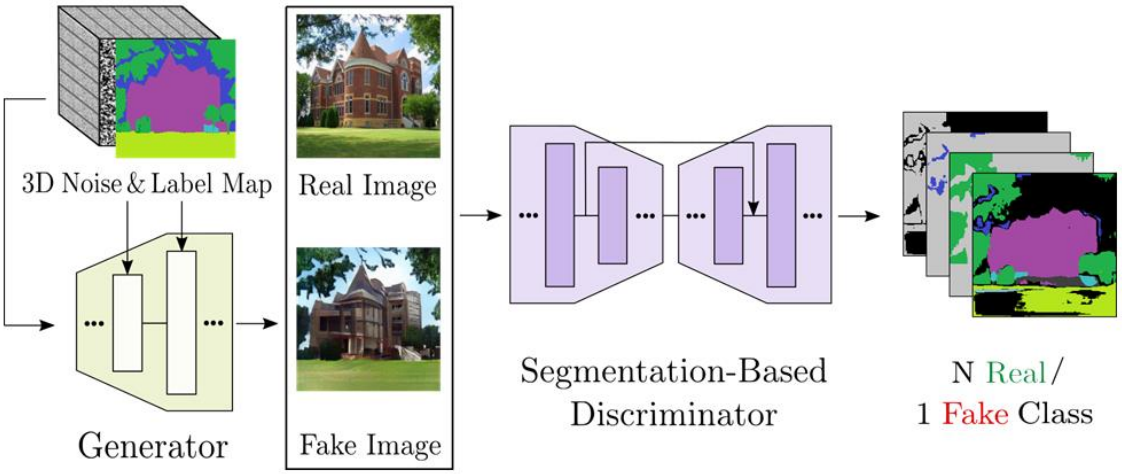
להלן הפונקציה preprocess_input:

```
def preprocess_input(opt, label):
    label = label.long()
    if opt.gpu_ids != "-1":
        label = label.cuda()
    label_map = label
    bs, _, h, w = label_map.size()
    nc = opt.semantic_nc
    if opt.gpu_ids != "-1":
        input_label = torch.cuda.FloatTensor(bs, nc, h, w).zero_()
    else:
        input_label = torch.FloatTensor(bs, nc, h, w).zero_()
    if (opt.segmentation == "panoptic"):
      semantic_map = torch.div(label_map, 1000, rounding_mode="floor")
      semantic_map = semantic_map.masked_fill_(semantic_map==255, 19)
      instance_map = torch.fmod(label_map, 1000.0) + 1.0
      input_semantics = input_label.scatter_(1, semantic_map, instance_
map)
    else:
      input_semantics = input_label.scatter_(1, label_map, 1.0)
    return input_semantics
```

בנייה ואימון המודל

אני משתמש בפרוייקט במודל OASIS אשר נחשב נכון למועד זה כ state-of-the-art. המודל נבנה על ידי צוות מחקר בחברת bosch וקוד המקור שלו נמצא בגיטהאב (https://github.com/boschresearch/OASIS) לו אני עושה clone על מנת להשתמש בו ישירות.

המודל עצמו בנוי משני מודלים נפרדים: generator ו discriminator.



תיאור סכמטי של המבנה הכללי של המודל

המחולל מקבל את הסימון של המחלקות הסמנטי לאחר ה preprocessing יחד עם רעש תלת מימדי שמוסיפים לו בדמות שכבות נוספות בטנסור, ומייצר טנסור מגודל [batch_size, 3, 256, 512] המייצג את התמונות המזוייפות.

להלן תקציר של ה generator כפי שנוצר על ידי הפונקציה summary שבחבילה torchinfo.

```
==========================================================================================
======
Layer (type:depth-idx)                                  Output Shape              Param #
==========================================================================================
======
OASIS_Generator                                         --                        --
├─ModuleList: 1-1                                       --                        --
├─Conv2d: 1-2                                           [2, 1024, 8, 16]          185,344
├─ModuleList: 1-1                                       --                        --
│    └─ResnetBlock_with_SPADE: 2-1                      [2, 1024, 8, 16]          --
│    │    └─SPADE: 3-1                                  [2, 1024, 8, 16]          --
│    │    │    └─SynchronizedBatchNorm2d: 4-1           [2, 1024, 8, 16]          --
│    │    │    └─Sequential: 4-2                        [2, 128, 8, 16]           --
│    │    │    │    └─Conv2d: 5-1                       [2, 128, 8, 16]           23,168
│    │    │    │    └─ReLU: 5-2                         [2, 128, 8, 16]           --
│    │    │    └─Conv2d: 4-3                            [2, 1024, 8, 16]          1,180,672
│    │    │    └─Conv2d: 4-4                            [2, 1024, 8, 16]          1,180,672
│    │    └─LeakyReLU: 3-2                              [2, 1024, 8, 16]          --
│    │    └─Conv2d: 3-3                                 [2, 1024, 8, 16]          9,438,208
│    │    └─SPADE: 3-4                                  [2, 1024, 8, 16]          --
│    │    │    └─SynchronizedBatchNorm2d: 4-5           [2, 1024, 8, 16]          --
│    │    │    └─Sequential: 4-6                        [2, 128, 8, 16]           --
│    │    │    │    └─Conv2d: 5-3                       [2, 128, 8, 16]           23,168
│    │    │    │    └─ReLU: 5-4                         [2, 128, 8, 16]           --
│    │    │    └─Conv2d: 4-7                            [2, 1024, 8, 16]          1,180,672
│    │    │    └─Conv2d: 4-8                            [2, 1024, 8, 16]          1,180,672
│    │    └─LeakyReLU: 3-5                              [2, 1024, 8, 16]          --
│    │    └─Conv2d: 3-6                                 [2, 1024, 8, 16]          9,438,208
├─Upsample: 1-3                                         [2, 1024, 16, 32]         --
├─ModuleList: 1-1                                       --                        --
│    └─ResnetBlock_with_SPADE: 2-2                      [2, 1024, 16, 32]         --
│    │    └─SPADE: 3-7                                  [2, 1024, 16, 32]         --
│    │    │    └─SynchronizedBatchNorm2d: 4-9           [2, 1024, 16, 32]         --
│    │    │    └─Sequential: 4-10                       [2, 128, 16, 32]          --
│    │    │    │    └─Conv2d: 5-5                       [2, 128, 16, 32]          23,168
```

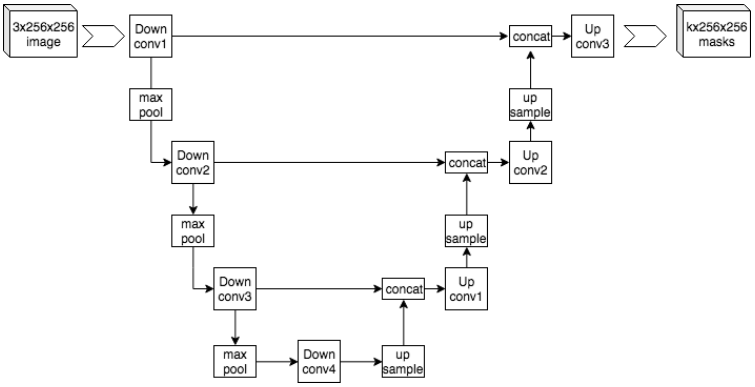
```
│   │   │   │   └─ReLU: 5-6                    [2, 128, 16, 32]          --
│   │   │   └─Conv2d: 4-11                     [2, 1024, 16, 32]         1,180,672
│   │   │   └─Conv2d: 4-12                     [2, 1024, 16, 32]         1,180,672
│   │   └─LeakyReLU: 3-8                       [2, 1024, 16, 32]         --
│   │   └─Conv2d: 3-9                          [2, 1024, 16, 32]         9,438,208
│   │   └─SPADE: 3-10                          [2, 1024, 16, 32]         --
│   │   │   └─SynchronizedBatchNorm2d: 4-13     [2, 1024, 16, 32]         --
│   │   │   └─Sequential: 4-14                  [2, 128, 16, 32]          --
│   │   │   │   └─Conv2d: 5-7                   [2, 128, 16, 32]          23,168
│   │   │   │   └─ReLU: 5-8                     [2, 128, 16, 32]          --
│   │   │   └─Conv2d: 4-15                     [2, 1024, 16, 32]         1,180,672
│   │   │   └─Conv2d: 4-16                     [2, 1024, 16, 32]         1,180,672
│   │   └─LeakyReLU: 3-11                      [2, 1024, 16, 32]         --
│   │   └─Conv2d: 3-12                         [2, 1024, 16, 32]         9,438,208
├─Upsample: 1-4                                [2, 1024, 32, 64]         --
├─ModuleList: 1-1                              --                        --
│   └─ResnetBlock_with_SPADE: 2-3              [2, 512, 32, 64]          --
│   │   └─SPADE: 3-13                          [2, 1024, 32, 64]         --
│   │   │   └─SynchronizedBatchNorm2d: 4-17     [2, 1024, 32, 64]         --
│   │   │   └─Sequential: 4-18                  [2, 128, 32, 64]          --
│   │   │   │   └─Conv2d: 5-9                   [2, 128, 32, 64]          23,168
│   │   │   │   └─ReLU: 5-10                    [2, 128, 32, 64]          --
│   │   │   └─Conv2d: 4-19                     [2, 1024, 32, 64]         1,180,672
│   │   │   └─Conv2d: 4-20                     [2, 1024, 32, 64]         1,180,672
│   │   └─Conv2d: 3-14                         [2, 512, 32, 64]          524,288
│   │   └─SPADE: 3-15                          [2, 1024, 32, 64]         --
│   │   │   └─SynchronizedBatchNorm2d: 4-21     [2, 1024, 32, 64]         --
│   │   │   └─Sequential: 4-22                  [2, 128, 32, 64]          --
│   │   │   │   └─Conv2d: 5-11                  [2, 128, 32, 64]          23,168
│   │   │   │   └─ReLU: 5-12                    [2, 128, 32, 64]          --
│   │   │   └─Conv2d: 4-23                     [2, 1024, 32, 64]         1,180,672
│   │   │   └─Conv2d: 4-24                     [2, 1024, 32, 64]         1,180,672
│   │   └─LeakyReLU: 3-16                      [2, 1024, 32, 64]         --
│   │   └─Conv2d: 3-17                         [2, 512, 32, 64]          4,719,104
│   │   └─SPADE: 3-18                          [2, 512, 32, 64]          --
│   │   │   └─SynchronizedBatchNorm2d: 4-25     [2, 512, 32, 64]          --
│   │   │   └─Sequential: 4-26                  [2, 128, 32, 64]          --
│   │   │   │   └─Conv2d: 5-13                  [2, 128, 32, 64]          23,168
│   │   │   │   └─ReLU: 5-14                    [2, 128, 32, 64]          --
│   │   │   └─Conv2d: 4-27                     [2, 512, 32, 64]          590,336
│   │   │   └─Conv2d: 4-28                     [2, 512, 32, 64]          590,336
│   │   └─LeakyReLU: 3-19                      [2, 512, 32, 64]          --
│   │   └─Conv2d: 3-20                         [2, 512, 32, 64]          2,359,808
├─Upsample: 1-5                                [2, 512, 64, 128]         --
├─ModuleList: 1-1                              --                        --
│   └─ResnetBlock_with_SPADE: 2-4              [2, 256, 64, 128]         --
│   │   └─SPADE: 3-21                          [2, 512, 64, 128]         --
│   │   │   └─SynchronizedBatchNorm2d: 4-29     [2, 512, 64, 128]         --
│   │   │   └─Sequential: 4-30                  [2, 128, 64, 128]         --
│   │   │   │   └─Conv2d: 5-15                  [2, 128, 64, 128]         23,168
│   │   │   │   └─ReLU: 5-16                    [2, 128, 64, 128]         --
│   │   │   └─Conv2d: 4-31                     [2, 512, 64, 128]         590,336
│   │   │   └─Conv2d: 4-32                     [2, 512, 64, 128]         590,336
│   │   └─Conv2d: 3-22                         [2, 256, 64, 128]         131,072
│   │   └─SPADE: 3-23                          [2, 512, 64, 128]         --
│   │   │   └─SynchronizedBatchNorm2d: 4-33     [2, 512, 64, 128]         --
│   │   │   └─Sequential: 4-34                  [2, 128, 64, 128]         --
│   │   │   │   └─Conv2d: 5-17                  [2, 128, 64, 128]         23,168
│   │   │   │   └─ReLU: 5-18                    [2, 128, 64, 128]         --
│   │   │   └─Conv2d: 4-35                     [2, 512, 64, 128]         590,336
│   │   │   └─Conv2d: 4-36                     [2, 512, 64, 128]         590,336
│   │   └─LeakyReLU: 3-24                      [2, 512, 64, 128]         --
│   │   └─Conv2d: 3-25                         [2, 256, 64, 128]         1,179,904
│   │   └─SPADE: 3-26                          [2, 256, 64, 128]         --
│   │   │   └─SynchronizedBatchNorm2d: 4-37     [2, 256, 64, 128]         --
│   │   │   └─Sequential: 4-38                  [2, 128, 64, 128]         --
│   │   │   │   └─Conv2d: 5-19                  [2, 128, 64, 128]         23,168
│   │   │   │   └─ReLU: 5-20                    [2, 128, 64, 128]         --
│   │   │   └─Conv2d: 4-39                     [2, 256, 64, 128]         295,168
│   │   │   └─Conv2d: 4-40                     [2, 256, 64, 128]         295,168
│   │   └─LeakyReLU: 3-27                      [2, 256, 64, 128]         --
│   │   └─Conv2d: 3-28                         [2, 256, 64, 128]         590,080
├─Upsample: 1-6                                [2, 256, 128, 256]        --
├─ModuleList: 1-1                              --                        --
│   └─ResnetBlock_with_SPADE: 2-5              [2, 128, 128, 256]        --
│   │   └─SPADE: 3-29                          [2, 256, 128, 256]        --
```

```
│    │    │    └─SynchronizedBatchNorm2d: 4-41     [2, 256, 128, 256]        --
│    │    │    └─Sequential: 4-42                  [2, 128, 128, 256]        --
│    │    │    │    └─Conv2d: 5-21                 [2, 128, 128, 256]        23,168
│    │    │    │    └─ReLU: 5-22                   [2, 128, 128, 256]        --
│    │    │    └─Conv2d: 4-43                      [2, 256, 128, 256]        295,168
│    │    │    └─Conv2d: 4-44                      [2, 256, 128, 256]        295,168
│    │    └─Conv2d: 3-30                           [2, 128, 128, 256]        32,768
│    │    └─SPADE: 3-31                            [2, 256, 128, 256]        --
│    │    │    └─SynchronizedBatchNorm2d: 4-45     [2, 256, 128, 256]        --
│    │    │    └─Sequential: 4-46                  [2, 128, 128, 256]        --
│    │    │    │    └─Conv2d: 5-23                 [2, 128, 128, 256]        23,168
│    │    │    │    └─ReLU: 5-24                   [2, 128, 128, 256]        --
│    │    │    └─Conv2d: 4-47                      [2, 256, 128, 256]        295,168
│    │    │    └─Conv2d: 4-48                      [2, 256, 128, 256]        295,168
│    │    └─LeakyReLU: 3-32                        [2, 256, 128, 256]        --
│    │    └─Conv2d: 3-33                           [2, 128, 128, 256]        295,040
│    │    └─SPADE: 3-34                            [2, 128, 128, 256]        --
│    │    │    └─SynchronizedBatchNorm2d: 4-49     [2, 128, 128, 256]        --
│    │    │    └─Sequential: 4-50                  [2, 128, 128, 256]        --
│    │    │    │    └─Conv2d: 5-25                 [2, 128, 128, 256]        23,168
│    │    │    │    └─ReLU: 5-26                   [2, 128, 128, 256]        --
│    │    │    └─Conv2d: 4-51                      [2, 128, 128, 256]        147,584
│    │    │    └─Conv2d: 4-52                      [2, 128, 128, 256]        147,584
│    │    └─LeakyReLU: 3-35                        [2, 128, 128, 256]        --
│    │    └─Conv2d: 3-36                           [2, 128, 128, 256]        147,584
├─Upsample: 1-7                                    [2, 128, 256, 512]        --
├─ModuleList: 1-1                                  --                        --
│    └─ResnetBlock_with_SPADE: 2-6                 [2, 64, 256, 512]         --
│    │    └─SPADE: 3-37                            [2, 128, 256, 512]        --
│    │    │    └─SynchronizedBatchNorm2d: 4-53     [2, 128, 256, 512]        --
│    │    │    └─Sequential: 4-54                  [2, 128, 256, 512]        --
│    │    │    │    └─Conv2d: 5-27                 [2, 128, 256, 512]        23,168
│    │    │    │    └─ReLU: 5-28                   [2, 128, 256, 512]        --
│    │    │    └─Conv2d: 4-55                      [2, 128, 256, 512]        147,584
│    │    │    └─Conv2d: 4-56                      [2, 128, 256, 512]        147,584
│    │    └─Conv2d: 3-38                           [2, 64, 256, 512]         8,192
│    │    └─SPADE: 3-39                            [2, 128, 256, 512]        --
│    │    │    └─SynchronizedBatchNorm2d: 4-57     [2, 128, 256, 512]        --
│    │    │    └─Sequential: 4-58                  [2, 128, 256, 512]        --
│    │    │    │    └─Conv2d: 5-29                 [2, 128, 256, 512]        23,168
│    │    │    │    └─ReLU: 5-30                   [2, 128, 256, 512]        --
│    │    │    └─Conv2d: 4-59                      [2, 128, 256, 512]        147,584
│    │    │    └─Conv2d: 4-60                      [2, 128, 256, 512]        147,584
│    │    └─LeakyReLU: 3-40                        [2, 128, 256, 512]        --
│    │    └─Conv2d: 3-41                           [2, 64, 256, 512]         73,792
│    │    └─SPADE: 3-42                            [2, 64, 256, 512]         --
│    │    │    └─SynchronizedBatchNorm2d: 4-61     [2, 64, 256, 512]         --
│    │    │    └─Sequential: 4-62                  [2, 128, 256, 512]        --
│    │    │    │    └─Conv2d: 5-31                 [2, 128, 256, 512]        23,168
│    │    │    │    └─ReLU: 5-32                   [2, 128, 256, 512]        --
│    │    │    └─Conv2d: 4-63                      [2, 64, 256, 512]         73,792
│    │    │    └─Conv2d: 4-64                      [2, 64, 256, 512]         73,792
│    │    └─LeakyReLU: 3-43                        [2, 64, 256, 512]         --
│    │    └─Conv2d: 3-44                           [2, 64, 256, 512]         36,928
├─Conv2d: 1-8                                      [2, 3, 256, 512]          1,731
==========================================================================================
======
Total params: 68,923,331
Trainable params: 68,923,331
Non-trainable params: 0
Total mult-adds (G): 479.78
==========================================================================================
======
Input size (MB): 20.97
Forward/backward pass size (MB): 4415.03
Params size (MB): 275.69
Estimated Total Size (MB): 4711.69
```

המאבחן (discriminator), מקבל תמונות אמיתיות ומזוייפות ומטרתו ליצור סימון סמנטי עבור התמונה שהוא מקבל. המאבחן מנסה לקבוע עבור כל פיקסל את המחלקה הסמנטית אליו הוא שייך כאשר מזוייף על ידי המחולל היא

מחלקה נוספת. כלומר קיימות 35 מחלקות אמיתיות ו1 של זיוף. העובדה שהמאבחן מנסה לזהות גם את המחלקות הסמנטיות האמיתיות בתוך התמונה, מאשר רק לקבוע בינארית אם התמונה אמיתית או מזוייפת, מסייעת למחולל לקבל מידע רב וספציפי יותר על הטעויות שלו. ניתן ליצור אנלוגיה לכך שכאשר נרצה לתת חוות דעת שלנו על משהו מסויים שמישהו יצר, נרצה לתת ביקורת בונה אשר מפרטת גם הצלחות ומקומות לשימור וגם את המקומות לשיפור, ולא רק אם הוא עשה טוב או לא טוב. ביקורת אינפורמטיבית ומדוייקת תסייע לאותו מישהו ליצור משהו טוב יותר בפעם הבאה.

הארכיטקטורה עצמה של המאבחן היא כשל U-Net, מודל מוכר המשמש ליצירת סגמנטציה של תמונות. המאבחן קודם מייצא את הפיצ'רים של התמונה, ומתוכם הוא בונה את הסגמנטציה של התמונה, כאשר הוא מקבל גם מידע משלבים מוקדמים של ניתוח התמונה.



איור סכמטי של מודל U-Net

שמו נגזר מצורת ה-U שהוא מקבל, כפי שניתן לראות באיור.

ספציפית המאבחן של OASIS מכיל 6 residual blocks, בניגוד ל 3 המתוארים באיור.

להלן תקציר של ה discriminator כפי שנוצר על ידי הפונקציה summary שבחבילה torchinfo.

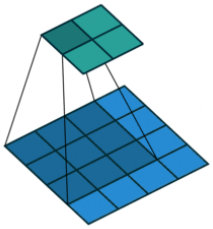
```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
OASIS_Discriminator                      --                        --
├─ModuleList: 1-1                        --                        --
├─ModuleList: 1-2                        --                        --
│    └─residual_block_D: 2-1             [2, 128, 128, 256]        --
│    │    └─AvgPool2d: 3-1               [2, 3, 128, 256]          --
│    │    └─Conv2d: 3-2                  [2, 128, 128, 256]        512
│    │    └─Sequential: 3-3              [2, 128, 256, 512]        --
│    │    │    └─Conv2d: 4-1             [2, 128, 256, 512]        3,584
│    │    └─Sequential: 3-4              [2, 128, 256, 512]        --
│    │    │    └─LeakyReLU: 4-2          [2, 128, 256, 512]        --
│    │    │    └─Conv2d: 4-3             [2, 128, 256, 512]        147,584
│    │    └─AvgPool2d: 3-5               [2, 128, 128, 256]        --
│    └─residual_block_D: 2-2             [2, 128, 64, 128]         --
│    │    └─AvgPool2d: 3-6               [2, 128, 64, 128]         --
│    │    └─Sequential: 3-7              [2, 128, 128, 256]        --
│    │    │    └─LeakyReLU: 4-4          [2, 128, 128, 256]        --
│    │    │    └─Conv2d: 4-5             [2, 128, 128, 256]        147,584
│    │    └─Sequential: 3-8              [2, 128, 128, 256]        --
│    │    │    └─LeakyReLU: 4-6          [2, 128, 128, 256]        --
│    │    │    └─Conv2d: 4-7             [2, 128, 128, 256]        147,584
│    │    └─AvgPool2d: 3-9               [2, 128, 64, 128]         --
│    └─residual_block_D: 2-3             [2, 256, 32, 64]          --
│    │    └─Conv2d: 3-10                 [2, 256, 64, 128]         33,024
│    │    └─AvgPool2d: 3-11              [2, 256, 32, 64]          --
│    │    └─Sequential: 3-12             [2, 256, 64, 128]         --
│    │    │    └─LeakyReLU: 4-8          [2, 128, 64, 128]         --
│    │    │    └─Conv2d: 4-9             [2, 256, 64, 128]         295,168
│    │    └─Sequential: 3-13             [2, 256, 64, 128]         --
│    │    │    └─LeakyReLU: 4-10         [2, 256, 64, 128]         --
│    │    │    └─Conv2d: 4-11            [2, 256, 64, 128]         590,080
```

```
│   │   └─AvgPool2d: 3-14              [2, 256, 32, 64]          --
│   └─residual_block_D: 2-4            [2, 256, 16, 32]          --
│   │   └─AvgPool2d: 3-15              [2, 256, 16, 32]          --
│   │   └─Sequential: 3-16             [2, 256, 32, 64]          --
│   │   │   └─LeakyReLU: 4-12          [2, 256, 32, 64]          --
│   │   │   └─Conv2d: 4-13             [2, 256, 32, 64]          590,080
│   │   └─Sequential: 3-17             [2, 256, 32, 64]          --
│   │   │   └─LeakyReLU: 4-14          [2, 256, 32, 64]          --
│   │   │   └─Conv2d: 4-15             [2, 256, 32, 64]          590,080
│   │   └─AvgPool2d: 3-18              [2, 256, 16, 32]          --
│   └─residual_block_D: 2-5            [2, 512, 8, 16]           --
│   │   └─Conv2d: 3-19                 [2, 512, 16, 32]          131,584
│   │   └─AvgPool2d: 3-20              [2, 512, 8, 16]           --
│   │   └─Sequential: 3-21             [2, 512, 16, 32]          --
│   │   │   └─LeakyReLU: 4-16          [2, 256, 16, 32]          --
│   │   │   └─Conv2d: 4-17             [2, 512, 16, 32]          1,180,160
│   │   └─Sequential: 3-22             [2, 512, 16, 32]          --
│   │   │   └─LeakyReLU: 4-18          [2, 512, 16, 32]          --
│   │   │   └─Conv2d: 4-19             [2, 512, 16, 32]          2,359,808
│   │   └─AvgPool2d: 3-23              [2, 512, 8, 16]           --
│   └─residual_block_D: 2-6            [2, 512, 4, 8]            --
│   │   └─AvgPool2d: 3-24              [2, 512, 4, 8]            --
│   │   └─Sequential: 3-25             [2, 512, 8, 16]           --
│   │   │   └─LeakyReLU: 4-20          [2, 512, 8, 16]           --
│   │   │   └─Conv2d: 4-21             [2, 512, 8, 16]           2,359,808
│   │   └─Sequential: 3-26             [2, 512, 8, 16]           --
│   │   │   └─LeakyReLU: 4-22          [2, 512, 8, 16]           --
│   │   │   └─Conv2d: 4-23             [2, 512, 8, 16]           2,359,808
│   │   └─AvgPool2d: 3-27              [2, 512, 4, 8]            --
├─ModuleList: 1-1                      --                        --
│   └─residual_block_D: 2-7            [2, 512, 8, 16]           --
│   │   └─Upsample: 3-28               [2, 512, 8, 16]           --
│   │   └─Sequential: 3-29             [2, 512, 8, 16]           --
│   │   │   └─LeakyReLU: 4-24          [2, 512, 4, 8]            --
│   │   │   └─Upsample: 4-25           [2, 512, 8, 16]           --
│   │   │   └─Conv2d: 4-26             [2, 512, 8, 16]           2,359,808
│   │   └─Sequential: 3-30             [2, 512, 8, 16]           --
│   │   │   └─LeakyReLU: 4-27          [2, 512, 8, 16]           --
│   │   │   └─Conv2d: 4-28             [2, 512, 8, 16]           2,359,808
│   └─residual_block_D: 2-8            [2, 256, 16, 32]          --
│   │   └─Upsample: 3-31               [2, 1024, 16, 32]         --
│   │   └─Conv2d: 3-32                 [2, 256, 16, 32]          262,400
│   │   └─Sequential: 3-33             [2, 256, 16, 32]          --
│   │   │   └─LeakyReLU: 4-29          [2, 1024, 8, 16]          --
│   │   │   └─Upsample: 4-30           [2, 1024, 16, 32]         --
│   │   │   └─Conv2d: 4-31             [2, 256, 16, 32]          2,359,552
│   │   └─Sequential: 3-34             [2, 256, 16, 32]          --
│   │   │   └─LeakyReLU: 4-32          [2, 256, 16, 32]          --
│   │   │   └─Conv2d: 4-33             [2, 256, 16, 32]          590,080
│   └─residual_block_D: 2-9            [2, 256, 32, 64]          --
│   │   └─Upsample: 3-35               [2, 512, 32, 64]          --
│   │   └─Conv2d: 3-36                 [2, 256, 32, 64]          131,328
│   │   └─Sequential: 3-37             [2, 256, 32, 64]          --
│   │   │   └─LeakyReLU: 4-34          [2, 512, 16, 32]          --
│   │   │   └─Upsample: 4-35           [2, 512, 32, 64]          --
│   │   │   └─Conv2d: 4-36             [2, 256, 32, 64]          1,179,904
│   │   └─Sequential: 3-38             [2, 256, 32, 64]          --
│   │   │   └─LeakyReLU: 4-37          [2, 256, 32, 64]          --
│   │   │   └─Conv2d: 4-38             [2, 256, 32, 64]          590,080
│   └─residual_block_D: 2-10           [2, 128, 64, 128]         --
│   │   └─Upsample: 3-39               [2, 512, 64, 128]         --
│   │   └─Conv2d: 3-40                 [2, 128, 64, 128]         65,664
│   │   └─Sequential: 3-41             [2, 128, 64, 128]         --
│   │   │   └─LeakyReLU: 4-39          [2, 512, 32, 64]          --
│   │   │   └─Upsample: 4-40           [2, 512, 64, 128]         --
│   │   │   └─Conv2d: 4-41             [2, 128, 64, 128]         589,952
│   │   └─Sequential: 3-42             [2, 128, 64, 128]         --
│   │   │   └─LeakyReLU: 4-42          [2, 128, 64, 128]         --
│   │   │   └─Conv2d: 4-43             [2, 128, 64, 128]         147,584
│   └─residual_block_D: 2-11           [2, 128, 128, 256]        --
│   │   └─Upsample: 3-43               [2, 256, 128, 256]        --
│   │   └─Conv2d: 3-44                 [2, 128, 128, 256]        32,896
│   │   └─Sequential: 3-45             [2, 128, 128, 256]        --
│   │   │   └─LeakyReLU: 4-44          [2, 256, 64, 128]         --
│   │   │   └─Upsample: 4-45           [2, 256, 128, 256]        --
│   │   │   └─Conv2d: 4-46             [2, 128, 128, 256]        295,040
```

```
│    │    └─Sequential: 3-46              [2, 128, 128, 256]        --
│    │    │    └─LeakyReLU: 4-47          [2, 128, 128, 256]        --
│    │    │    └─Conv2d: 4-48             [2, 128, 128, 256]        147,584
│    └─residual_block_D: 2-12             [2, 64, 256, 512]         --
│    │    └─Upsample: 3-47                [2, 256, 256, 512]        --
│    │    └─Conv2d: 3-48                  [2, 64, 256, 512]         16,448
│    │    └─Sequential: 3-49              [2, 64, 256, 512]         --
│    │    │    └─LeakyReLU: 4-49          [2, 256, 128, 256]        --
│    │    │    └─Upsample: 4-50           [2, 256, 256, 512]        --
│    │    │    └─Conv2d: 4-51             [2, 64, 256, 512]         147,520
│    │    └─Sequential: 3-50              [2, 64, 256, 512]         --
│    │    │    └─LeakyReLU: 4-52          [2, 64, 256, 512]         --
│    │    │    └─Conv2d: 4-53             [2, 64, 256, 512]         36,928
├─Conv2d: 1-3                             [2, 36, 256, 512]         2,340
==========================================================================================
Total params: 22,251,364
Trainable params: 22,251,364
Non-trainable params: 0
Total mult-adds (G): 18.39
==========================================================================================
Input size (MB): 3.15
Forward/backward pass size (MB): 1633.68
Params size (MB): 89.01
Estimated Total Size (MB): 1725.83
==========================================================================================
```

המודל משלב בתוכו שכבות רבות:

- שכבת קונבולוציה (Conv2d) –

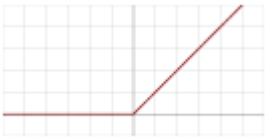


קונבולוציה על תמונה מגודל 4x4 עם ליבה בגודל 3x3

שכבת קונובלוציה לוקחת את הטנזור ומכפילה אותו בליבה (kernel) שהיא בעצמה טנזור מגודל קטן יותר. מטרת המודל היא ללמוד את ערכי הליבה.

- שכבת אקטיבציה (ReLU) –

שכבת האקטיבציה לוקחת את הטנזור ומפעילה את פונקציית ה ReLU(Rectified Linear Unit) הפופולארית המוגדרת להיות:



גרף הפונקציה ReLU

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

לפונקציה יתרונות רבים כאשר העיקרית שבהם היא הטיפול בבעיית הגרדיאנט הנעלם, בנוסף לחישוביות מהירה. הבעיה העיקרית של הפונקציה היא שנוירונים עשויים לעיתים להגיע למצב שבו הם אינם מגיבים עוד ולמעשה "מתים", מפני שערכם שלילי עבור כל קלט.

- שכבת אקטיבציה (LeakyReLU) –

שכבת האקטיבציה לוקחת את הטנזור ומפעילה את פונקציית ה LeakyReLU המוגדרת להיות :

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ 0.01x & \text{otherwise.} \end{cases}$$

פונקציה זו מטרתה לטפל בבעיה שבה נוירונים עשויים "למות", ומאפשרת גרדיאנט חיובי קטן כאשר הנוירון שלילי. אמנם, ישנה ירידה בביצועים לעומת ReLU.

- שכבת אגרגציה (AvgPool2d) –

שכבת האגרגציה תעבור על טנזור הקלט עם חלון בגודל מסויים, ועבור כל חלון שכזה תחשב את הממוצע של כל הערכים בחלון, כך שהממוצע ייצג את החלון בטנזור הפלט.
שכבות אגרגציה מאפשרות לנו להקטין את הטנזורים ובכך למעשה לייצא את הפיצ'רים החשובים.

- שכבת upsample (Upsample) –

שכבת ה upsample היא הפעולה ההופכית לפעולת האגרגציה, בכך שכל ערך בטנזור הקלט ישוכפל ויופיע 4 פעמים כחלון 2x2 בטנזור הקלט.
שכבה זו מאפשרת לנו לקחת את הפיצ'רים של התמונה ולהפוך אותם לפלט מפורט.

```
        1, 2
Input = (3, 4)

         1, 1, 2, 2
Output = (1, 1, 2, 2)
         3, 3, 4, 4
         3, 3, 4, 4
```

קלט ופלט של שכבת upsample
על מטריצה לדוגמא

- שכבת נורמליזציה (SynchronizedBatchNorm2d) –

שכבת הנורמליזציה מטרתה להתאים את הפרמטרים של המודל כך שהתפלגות הפרמטרים תתאים להתפלגות הנורמלית הסטנדרטית, כלומר התוחלת תהיה שווה ל 0 והשונות ל 1. על אף שעדיין אין הסכמה גורפת מדוע, ידוע כי הנורמליזציה מביאה להתכנסות מהירה ויציבה יותר של תהליך אימון המודל.

משמעות ה synchronized היא שכאשר נאמן את המודל על כמה GPU-ים שונים, הסטטיסטיקה לגבי התוחלת והשונות של המודל תחושב עבור כל ה mini-batch שעל כל המכשירים, בניגוד ל BatchNorm2d רגיל שיחשב את הסטטיסטיקה עבור כל מכשיר בנפרד, מה שיוביל לתוצאות לא מדויקות.

את הפרמטרים וההיפרפרמטרים של המודל ניתן לשנות בנוחות באמצעות הפונקציה custom_arguments שלתוכה נוסיף את הפרמטרים שאנחנו רוצים לשנות:

```
def custom_arguments(opt):
  #opt.your-argument-here = value
  opt.name = "my_cityscapes_train"
  opt.segmentation = "panoptic"
  opt.batch_size = 14
  pass
```

את כל הפרמטרים וההיפרפרמטרים השונים וערכיהם הדפולטיביים ניתן לראות בפונקציה get_default_opt שנמצאת בקובץ my_utils.py:

```
def get_default_opt(train):
  #do not change these parameters
  opt = SimpleNamespace(
      #--- general options ---gdfsgds
      name="oasis_cityscapes_pretrained",
   # name of the experiment. It decides where to store samples and models
      seed=42,
   # random seed
      segmentation="semantic",
   # use semantic or panoptic segmentation for training
      gpu_ids='0',
   # gpu ids: e.g. 0  0,1,2, 0,2. use -1 for CPU
      checkpoints_dir=root_project_directory+"/pretrained_checkpoints",
   # models are saved here
      no_spectral_norm=False,
   # this option deactivates spectral norm in all layers
      batch_size=20,
   # input batch size
      dataroot=root_project_directory+"/dataset",
   # path to dataset root
      dataset_mode="cityscapes",
   # this option indicates which dataset should be loaded
```

```
    no_flip=False,
 # if specified, do not flip the images for data argumentation
    #--- generator options ---
    num_res_blocks=6,
 # number of residual blocks in G and D
    channels_G=64,
 # number of gen filters in first conv layer in generator
    param_free_norm="syncbatch",
 # which norm to use in generator before SPADE
    spade_ks=3,
 # kernel size of convs inside SPADE
    no_EMA=False,
 # if specified, do *not* compute exponential moving averages
    EMA_decay=0.9999,
 # decay in exponential moving averages
    no_3dnoise=False,
 # if specified, do *not* concatenate noise to label maps
    z_dim=64)
 # dimension of the latent z vector

if train:
  opt_extra = SimpleNamespace(
      freq_print=1000,
 # frequency of showing training results
      freq_save_ckpt=20000,
 # frequency of saving the checkpoints
      freq_save_latest=10000,
 # frequency of saving the latest model
      freq_smooth_loss=250,
 # smoothing window for loss visualization
      freq_save_loss=2500,
 # frequency of loss plot updates
      freq_fid=5000,
 # frequency of saving the fid score (in training iterations)
      continue_train=True,
 # resume previously interrupted training
      which_iter='latest',
 # which epoch to load when continue_train
      num_epochs=10,
 # number of epochs to train
      beta1=0.0,
 # momentum term of adam
      beta2=0.999,
 # momentum term of adam
      lr_g=0.0001,
 # G learning rate
      lr_d=0.0004,
 # D learning rate

      channels_D=64,
 # number of discrim filters in first conv layer in discriminator
      add_vgg_loss=False,
 # if specified, add VGG feature matching loss
      lambda_vgg=10.0,
 # weight for VGG loss
      no_balancing_inloss=False,
 # if specified, do *not* use class balancing in the loss function
      no_labelmix=False,
 # if specified, do *not* use LabelMix
      lambda_labelmix=10.0
 # weight for LabelMix regularization
```

```
    )
  else:
    opt_extra = SimpleNamespace(
        results_dir=root_project_directory+"/results/",
   # saves testing results here.
        ckpt_iter='best'
   # which epoch to load to evaluate a model
    )
  opt = SimpleNamespace(**opt.__dict__, **opt_extra.__dict__)        # conac
tenate all options
  opt.phase = 'train' if train else 'test'
  opt.loaded_latest_iter=None;
  return opt
```

תהליך האימון, יחד עם כל הדוחות והמודלים המאומנים נשמרים בתיקייה opt.checkpoints_dir/opt.name.

את תוצאות האימונים הדפולטיביים ניתן למצוא ולהוריד כאן (שינוי ה0 ל1 בסוף הלינק יוביל להורדה מיידית של הקובץ): https://www.dropbox.com/sh/nf6of02pyk84zjg/AADUqkgPWsPKUZUPXHnaji-ma/oasis_cityscapes_pretrained.zip?dl=0

את תהליך האימון ניתן להפעיל באמצעות קטע הקוד הבא:

```
1.  #--- read options ---#
2.  opt = configure_arguments(train=True)
3.  print("configured arguments")
4.  #--- create utils ---#
5.  timer = utils.timer(opt)
6.  visualizer_losses = utils.losses_saver(opt)
7.  losses_computer = losses.losses_computer(opt)
8.  dataloader, dataloader_val = get_dataloaders(opt)
9.  im_saver = utils.image_saver(opt)
10. fid_computer = fid_pytorch(opt, dataloader_val)   #problem with tpus
11.
12. #--- create models ---#
13. print("creating models")
14. model = models.OASIS_model(opt)
15. model = models.put_on_multi_gpus(model, opt)
16.
17. #--- create optimizers ---#
18. optimizerG = torch.optim.Adam(model.module.netG.parameters(), lr=opt.lr_g, betas=(o
    pt.beta1, opt.beta2))
19. optimizerD = torch.optim.Adam(model.module.netD.parameters(), lr=opt.lr_d, betas=(o
    pt.beta1, opt.beta2))
20.
21. #--- the training loop ---#
22. already_started = False
23. start_epoch, start_iter = utils.get_start_iters(opt.loaded_latest_iter, len(dataloa
    der))
24. for epoch in range(start_epoch, opt.num_epochs):
25.     for i, data_i in enumerate(dataloader):
26.         if not already_started and i < start_iter:
27.             continue
28.         already_started = True
29.         cur_iter = epoch*len(dataloader) + i
30.         image, label = preprocess_input(opt, data_i)
31.
32.         #--- generator update ---#
33.         model.module.netG.zero_grad()
34.         loss_G, losses_G_list = model.forward(image, label, "losses_G", losses_comp
    uter)
35.         loss_G, losses_G_list = loss_G.mean(), [loss.mean() if loss is not None els
    e None for loss in losses_G_list]
```

```
36.         loss_G.backward()
37.         optimizerG.step()
38.
39.         #--- discriminator update ---#
40.         model.module.netD.zero_grad()
41.         loss_D, losses_D_list = model.forward(image, label, "losses_D", losses_comp
    uter)
42.         loss_D, losses_D_list = loss_D.mean(), [loss.mean() if loss is not None els
    e None for loss in losses_D_list]
43.         loss_D.backward()
44.         optimizerD.step()
45.
46.         #--- stats update ---#
47.         if not opt.no_EMA:
48.             utils.update_EMA(model, cur_iter, dataloader, opt)
49.         if cur_iter % opt.freq_print == 0:
50.             im_saver.visualize_batch(model, image, label, cur_iter)
51.             timer(epoch, cur_iter)
52.         if cur_iter % opt.freq_save_ckpt == 0:
53.             utils.save_networks(opt, cur_iter, model)
54.         if cur_iter % opt.freq_save_latest == 0:
55.             utils.save_networks(opt, cur_iter, model, latest=True)
56.         if cur_iter % opt.freq_fid == 0 and cur_iter > 0:
57.             is_best = fid_computer.update(model, cur_iter)
58.             if is_best:
59.                 utils.save_networks(opt, cur_iter, model, best=True)
60.         visualizer_losses(cur_iter, losses_G_list+losses_D_list)
61.
62. #--- after training ---#
63. utils.update_EMA(model, cur_iter, dataloader, opt, force_run_stats=True)
64. utils.save_networks(opt, cur_iter, model)
65. utils.save_networks(opt, cur_iter, model, latest=True)
66. is_best = fid_computer.update(model, cur_iter)
67. if is_best:
68.     utils.save_networks(opt, cur_iter, model, best=True)
69.
70. print("The training has successfully finished")
71.
```

עבור האימון עם הערכים הדפולטיביים, פלט תהליך האימון נראה כך:

```
configured arguments
Created CityscapesDataset, size train: 2975, size val: 500
/usr/local/lib/python3.7/dist-packages/torchvision/models/inception.py:48:
FutureWarning: The default weight initialization of inception_v3 will be
changed in future releases of torchvision. If you wish to keep the old
behavior (which leads to long initialization times due to
scipy/scipy#11299), please set init_weights=True.
  FutureWarning,
--- Now computing Inception activations for real set ---
/usr/local/lib/python3.7/dist-
packages/torchvision/transforms/functional.py:424: UserWarning: Argument
interpolation should be of type InterpolationMode instead of int. Please,
use InterpolationMode enum.
  "Argument interpolation should be of type InterpolationMode instead of
int. "
--- Finished FID stats for real set ---
creating models
Created OASIS_Generator with 68923331 parameters
Created OASIS_Discriminator with 22250389 parameters
/usr/local/lib/python3.7/dist-packages/torch/nn/functional.py:1933:
UserWarning: nn.functional.tanh is deprecated. Use torch.tanh instead.
```

```
  warnings.warn("nn.functional.tanh is deprecated. Use torch.tanh
instead.")
[epoch 0/200 - iter 0], time:0.000
[epoch 6/200 - iter 1000], time:5.281
[epoch 13/200 - iter 2000], time:4.933
[epoch 20/200 - iter 3000], time:5.129
[epoch 27/200 - iter 4000], time:4.912
[epoch 33/200 - iter 5000], time:4.916
[epoch 40/200 - iter 6000], time:4.992
[epoch 47/200 - iter 7000], time:4.912
[epoch 54/200 - iter 8000], time:5.139
[epoch 60/200 - iter 9000], time:4.915
[epoch 67/200 - iter 10000], time:4.914
[epoch 74/200 - iter 11000], time:4.994
[epoch 81/200 - iter 12000], time:4.912
[epoch 87/200 - iter 13000], time:5.141
[epoch 94/200 - iter 14000], time:4.914
[epoch 101/200 - iter 15000], time:4.917
[epoch 108/200 - iter 16000], time:4.992
[epoch 114/200 - iter 17000], time:4.906
[epoch 121/200 - iter 18000], time:5.133
[epoch 128/200 - iter 19000], time:4.910
[epoch 135/200 - iter 20000], time:4.908
[epoch 141/200 - iter 21000], time:4.985
[epoch 148/200 - iter 22000], time:4.911
[epoch 155/200 - iter 23000], time:5.131
[epoch 162/200 - iter 24000], time:4.905
[epoch 168/200 - iter 25000], time:4.906
[epoch 175/200 - iter 26000], time:4.985
[epoch 182/200 - iter 27000], time:4.911
[epoch 189/200 - iter 28000], time:5.132
[epoch 195/200 - iter 29000], time:4.904
```

```
def tens_to_im(tens):
  out = (tens + 1) / 2
  out.clamp(0, 1)
  out = out.detach().cpu().numpy()
  return np.transpose(out, (1, 2, 0))

label = Image.open("/gtFine/train/hanover/hanover_000000_000164_gtFine_labelIds.png")
label = transforms(opt, label)
label = label * 255
input_semantics = preprocess_input(opt, label)
tens = model(None, input_semantics, "generate", None)
img_np = tens_to_im(tens[0])
plt.figure()
plt.subplot()
plt.imshow(img_np)
plt.show()
```

השינוי שאני ניסיתי להכניס למודל היה לאפשר לו לקבל תמונות פנופטיות ובתקווה לקבל תוצאות טובות יותר. ניסוי זה לא צלח. הקונפיגורציה בה ניסיתי לאמן את המודל היא:

```
               EMA_decay: 0.999                                [default: 0.9999]
            add_vgg_loss: False
```

```
               batch_size: 2                                [default: 20]
                    beta1: 0.0
                    beta2: 0.999
               channels_D: 64
               channels_G: 64
          checkpoints_dir: /content/drive/MyDrive/ProjectGAN/pretrained_checkpoints
           continue_train: False                            [default: True]
                 dataroot: /content/drive/MyDrive/ProjectGAN/datasets/cityscapes
             dataset_mode: cityscapes
                 freq_fid: 1                                [default: 5000]
               freq_print: 100                              [default: 1000]
           freq_save_ckpt: 500                              [default: 20000]
         freq_save_latest: 500                              [default: 10000]
           freq_save_loss: 500                              [default: 2500]
         freq_smooth_loss: 250
                  gpu_ids: -1
          lambda_labelmix: 5.0                              [default: 10.0]
               lambda_vgg: 10.0
       loaded_latest_iter: 0                                [default: None]
                     lr_d: 0.0004
                     lr_g: 0.0004                           [default: 0.0001]
                     name: my_cityscapes_cool_panoptic      [default:
oasis_cityscapes_pretrained]
               no_3dnoise: True                             [default: False]
                   no_EMA: True                             [default: False]
      no_balancing_inloss: False
                  no_flip: False
                no_labelmix: True                           [default: False]
         no_spectral_norm: False
               num_epochs: 2                                [default: 10]
            num_res_blocks: 6
           param_free_norm: syncbatch
                     phase: train
                      seed: 42
             segmentation: semantic
                 spade_ks: 3
               which_iter: latest
                     z_dim: 64
```

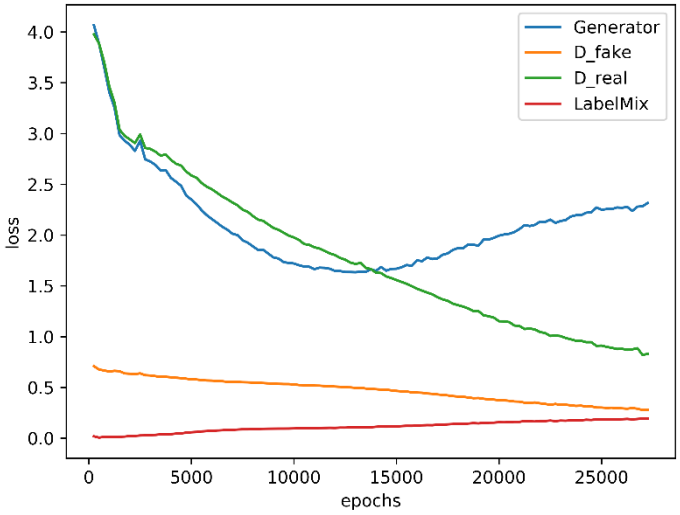
פלט תהליך האימון היה:

```
[epoch 0/200 - iter 0], time:0.000
[epoch 0/200 - iter 100], time:6.748
[epoch 0/200 - iter 200], time:6.768
[epoch 1/200 - iter 300], time:6.722
[epoch 1/200 - iter 400], time:6.697
[epoch 2/200 - iter 500], time:6.732
[epoch 2/200 - iter 600], time:6.714
[epoch 3/200 - iter 700], time:6.703
[epoch 3/200 - iter 800], time:6.699
[epoch 4/200 - iter 900], time:6.723
[epoch 4/200 - iter 1000], time:6.726
[epoch 5/200 - iter 1100], time:6.728
[epoch 5/200 - iter 1200], time:6.710
[epoch 6/200 - iter 1300], time:6.708
[epoch 6/200 - iter 1400], time:6.725
[epoch 7/200 - iter 1500], time:6.739
[epoch 7/200 - iter 1600], time:6.808
[epoch 8/200 - iter 1700], time:6.699
[epoch 8/200 - iter 1800], time:6.708
[epoch 8/200 - iter 1900], time:6.710
[epoch 9/200 - iter 2000], time:6.758
[epoch 9/200 - iter 2100], time:6.711
[epoch 10/200 - iter 2200], time:6.726
[epoch 10/200 - iter 2300], time:6.682
[epoch 11/200 - iter 2400], time:6.696
[epoch 11/200 - iter 2500], time:6.741
[epoch 12/200 - iter 2600], time:6.726
```

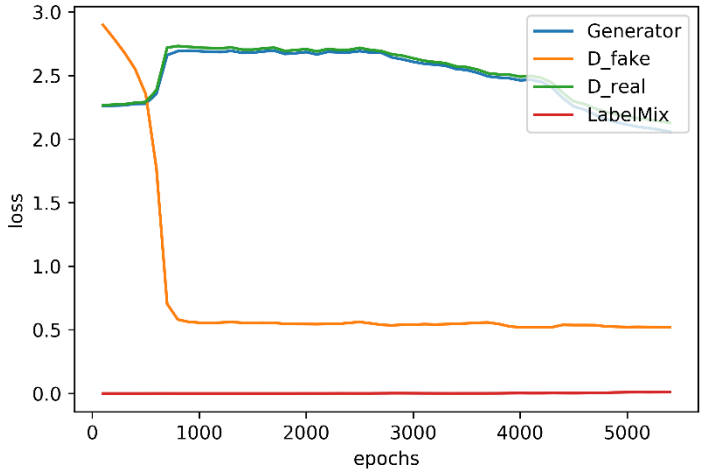
```
[epoch 12/200 - iter 2700], time:6.688
[epoch 13/200 - iter 2800], time:6.725
[epoch 13/200 - iter 2900], time:6.711
[epoch 14/200 - iter 3000], time:6.752
[epoch 14/200 - iter 3100], time:6.733
[epoch 15/200 - iter 3200], time:6.716
[epoch 15/200 - iter 3300], time:6.705
[epoch 16/200 - iter 3400], time:6.671
[epoch 16/200 - iter 3500], time:6.762
[epoch 16/200 - iter 3600], time:6.703
[epoch 17/200 - iter 3700], time:6.648
[epoch 17/200 - iter 3800], time:6.748
[epoch 18/200 - iter 3900], time:6.702
[epoch 18/200 - iter 4000], time:6.759
[epoch 19/200 - iter 4100], time:6.791
[epoch 19/200 - iter 4200], time:6.732
[epoch 20/200 - iter 4300], time:6.733
[epoch 20/200 - iter 4400], time:6.774
[epoch 21/200 - iter 4500], time:6.784
[epoch 21/200 - iter 4600], time:6.751
[epoch 22/200 - iter 4700], time:6.742
[epoch 22/200 - iter 4800], time:6.738
[epoch 23/200 - iter 4900], time:6.737
[epoch 23/200 - iter 5000], time:6.777
[epoch 24/200 - iter 5100], time:6.763
[epoch 24/200 - iter 5200], time:6.741
[epoch 25/200 - iter 5300], time:6.717
[epoch 25/200 - iter 5400], time:6.714
[epoch 25/200 - iter 5500], time:6.790
[epoch 26/200 - iter 5600], time:6.756
[epoch 26/200 - iter 5700], time:6.768
```

הזמן מייצג דקות.

עצרתי את תהליך האימון באמצע לאחר שהסתכלות בגרף המתאר את היסטוריית פונקציות המחירים הביאה אותי למסקנה שהמודל נתקע והאימון לא יצלח.



גרף המתאר את היסטוריית פונקציות המחיר עבור האימון הדפולטיבי



גרף המתאר את היסטוריית פונקציות המחיר עבור האימון הפנופטי

כפי שניתן לראות, המודל הפנופטי בתחילת האימון מגיע למצב בו הוא למעשה נתקע (עד איטרציה 2500 הוא אופקי כמעט לגמרי), ורק לאחר מכן נראה שיש שיפור קטן בתהליך האימון (יתכן שאימון ממושך יותר לבסוף כן יגיע לתוצאות רצויות, על אף שאיפשרתי לו להתאמן במהלך הלילה למשך כ 6 שעות על Nvidia RTX A6000 48GB).

אני סבור כי קיימות שתי סיבות הגיוניות היכולות להסביר למה המודל לא הצליח להתאמן.
ייתכן שזה מפני הדרך בה הכנתי את התמונות הפנופטיות כטנזורים לפני ההכנסה למודל. במקום ליצור masks כמספר המחלקות הסמנטיות כך שהערך ב mask יהיה ה instanceID, ניתן אולי ליצור masks כמספר המחלקות הסמנטיות + מספר העצמים השונים (השאלה היא איך המודל יגיב כאשר הוא יקבל טנזור שצורתו משתנה?).
סיבה נוספת יכולה להיות המאבחן שבנוי במקור ליצירה של תמונות סמנטיות. אולי התאמתו של המאבחן להצליח להבדיל גם בין עצמים שונים והתאמתו של פונקציית המחיר לקחת זאת בחשבון, תוביל לאימון יעיל ואיכותי יותר.

פונקציות המחיר

מטרתו של המאבחן היא לסווג כל פיקסל למחלקה הסמנטית המתאימה לו ולכן פונקציית המחיר של המאבחן היא פונקציית cross-entropy עבור N+1 מחלקות (N מחלקות סמנטיות + 1 עבור זיוף). מאחר וישנם מחלקות סמנטיות המיוצגות על ידי הרבה פיקסלים וישנם כאלה שפחות, נוצר מצב של imbalance, כלומר המחלקות הסמנטיות שתופסות שטח רחב יותר בתמונה יקבלו חשיבות גדולה יותר בפונקציית המחיר מה שעשוי לקפח את המחלקות הסמנטיות עם שטח קטן, אך חשובות לא פחות (למשל אנשים). על מנת לפתור זאת, מכפילים את המחיר של כל מחלקה סמנטית ב 1 חלקי השטח היחסי של המחלקה הסמנטית מהתמונה (תדירות הפיקסלים) . כלומר, ככל ששטח המחלקה קטן יותר, כך נכפיל את סכום מחירי הפיקסלים האינדיבידואלים שלה במשקל גדול יותר ולהיפך. כך נגיע למצב בו כל המחלקות הסמנטיות מקבלות ייצוג שווה בפונקציית המחיר. בייצוג מתמטי הפונקציה נראית כך:

$$\mathcal{L}_D = -\mathbb{E}_{(x,t)}\left[\sum_{c=1}^{N}\alpha_c \sum_{i,j}^{H\times W} t_{i,j,c}\log D(x)_{i,j,c}\right] - \mathbb{E}_{(z,t)}\left[\sum_{i,j}^{H\times W}\log D(G(z,t))_{i,j,c=N+1}\right]$$

אלפא מתארת את המשקל היחסי של המחלקה הסמנטית c בתמונה:

$$\alpha_c = \frac{H \times W}{\sum_{i,j}^{H \times W} E_t \left[\mathbb{1}[t_{i,j,c} = 1] \right]}.$$

המאבחן ינסה להקטין את הפונקציה הזו ככל הניתן, כאשר החלק השמאלי של הפונקציה מתארת את המחיר של התמונות האמיתיות והחלק הימני המחיר של התמונות המזויפות.

פונקציית המחיר של המחולל נגזרת מתוך פונקציית המחיר של המאבחן:

$$\mathcal{L}_G = -\mathbb{E}_{(z,t)} \left[\sum_{c=1}^{N} \alpha_c \sum_{i,j}^{H \times W} t_{i,j,c} \log D(G(z,t))_{i,j,c} \right],$$

המחולל ינסה למקסם את פונקציה זו, שקול לכך שהמחולל ינסה להערים על המאבחן.

באופן זה, המאבחן והמחולל משחקים משחק minmax וכל עוד הם נמצאים ברמה שווה ביכולת המשחק הם ילמדו זה מזה.

האופטימיזציה והעדכון של המודל נעשית באמצעות אלגוריתם Adam שנחשב נכון לעכשיו למהיר והמתקדם ביותר.

יישום המודל

בחרתי לפתח למודל יישום באמצעות anvil.

זוהי טכנולוגיה המאפשרת לפתח אתר אינטרנט בקלות על ידי גרירה של אלמנטים על גבי המסך, ותכנות של הפונקציונליות שלהם באמצעות פייתון.

בנוסף, anvil מאפשרת באמצעות טכנולוגיית uplink לחבר מחברת קולאב כך שניתן יהיה להריץ על גביה פונקציות. כלומר, מחברת קולאב יכולה לשמש כצד השרת של האתר.

לצורך כך, יצרתי מחברת חדשה שיש בה פונקציה שיודעת לקבל תמונה סמנטית, להריץ אותה על המודל ולהעביר חזרה לאתר תמונה פוטוריאליסטית שהתקבלה מהמודל. הפונקציה עצמה מבוססת על פי קטע הקוד שהוזכר למעלה.

ניתן למצוא את המחברת כאן: https://drive.google.com/file/d/1la8Aw-VsryxfJp_Tlk2-awP_W9B6IYHA/view?usp=sharing

חובה להריץ את המחברת על מנת שהאפליקציה תעבוד.

מדריך למפתח

קובץ my_utils.py

קובץ המכיל את כל פונקציות ה utilities שאני שיניתי והתאמתי.
נמצא תחת ה root_project_directory.

```
1.  import pickle
2.  from types import SimpleNamespace
3.  from constants import root_project_directory
4.  import utils.utils as utils
5.  import os
6.
7.  def load_options(opt):
8.      #load the options file of an already available model in the checkpoints directory.
9.      #gets the opt set by the user, and returns the new opt of the saved model.
10.     file_name = os.path.join(opt.checkpoints_dir, opt.name, "opt.pkl")
11.     new_opt = pickle.load(open(file_name, 'rb'))
12.     return new_opt
13.
14. def get_default_opt(train):
15.   #returns the default options object.
16.   #do not change these parameters here, do it in custom arguments.
17.   opt = SimpleNamespace(
18.       #--- general options ---gdfsgds
19.       name="oasis_cityscapes_pretrained",                                    # name of
    the experiment. It decides where to store samples and models
20.       seed=42,                                                               # random
    seed
21.       segmentation="semantic",                                               # use
    semantic or panoptic segmentation for training
22.       gpu_ids='0',                                                           # gpu ids:
    e.g. 0  0,1,2, 0,2. use -1 for CPU
23.       checkpoints_dir=root_project_directory+"/pretrained_checkpoints",      # models
    are saved here
24.       no_spectral_norm=False,                                                # this
    option deactivates spectral norm in all layers
25.       batch_size=20,                                                          # input
    batch size
26.       dataroot=root_project_directory+"/dataset",                # path to dataset root
27.       dataset_mode="cityscapes",                                             # this
    option indicates which dataset should be loaded
28.       no_flip=False,                                                         # if
    specified, do not flip the images for data argumentation
29.       #--- generator options ---
30.       num_res_blocks=6,                                                      # number
    of residual blocks in G and D
31.       channels_G=64,                                                         # number
    of gen filters in first conv layer in generator
32.       param_free_norm="syncbatch",                                           # which
    norm to use in generator before SPADE
33.       spade_ks=3,                                                            # kernel
    size of convs inside SPADE
34.       no_EMA=False,                                                          # if
    specified, do *not* compute exponential moving averages
35.       EMA_decay=0.9999,                                                      # decay in
    exponential moving averages
36.       no_3dnoise=False,                                                      # if
    specified, do *not* concatenate noise to label maps
37.       z_dim=64)                                                              #
    dimension of the latent z vector
38.
39.   if train:
40.     opt_extra = SimpleNamespace(
41.         freq_print=1000,                                                     #
    frequency of showing training results
```

```
42.         freq_save_ckpt=20000,                                                   #
    frequency of saving the checkpoints
43.         freq_save_latest=10000,                                                 #
    frequency of saving the latest model
44.         freq_smooth_loss=250,                                                   #
    smoothing window for loss visualization
45.         freq_save_loss=2500,                                                    #
    frequency of loss plot updates
46.         freq_fid=5000,                                                          #
    frequency of saving the fid score (in training iterations)
47.         continue_train=True,                                                   # resume
    previously interrupted training
48.         which_iter='latest',                                                    # which
    epoch to load when continue_train
49.         num_epochs=10,                                                         # number of
    epochs to train
50.         beta1=0.0,                                                              # momentum
    term of adam
51.         beta2=0.999,                                                            # momentum
    term of adam
52.         lr_g=0.0001,                                                            # G
    learning rate
53.         lr_d=0.0004,                                                            # D
    learning rate
54.
55.         channels_D=64,                                                          # number
    of discrim filters in first conv layer in discriminator
56.         add_vgg_loss=False,                                                     # if
    specified, add VGG feature matching loss
57.         lambda_vgg=10.0,                                                        # weight
    for VGG loss
58.         no_balancing_inloss=False,                                              # if
    specified, do *not* use class balancing in the loss function
59.         no_labelmix=False,                                                      # if
    specified, do *not* use LabelMix
60.         lambda_labelmix=10.0                                                    # weight
    for LabelMix regularization
61.     )
62.   else:
63.     opt_extra = SimpleNamespace(
64.         results_dir=root_project_directory+"/results/",                         # saves
    testing results here.
65.         ckpt_iter='best'                                                        # which
    epoch to load to evaluate a model
66.     )
67.
68.   opt = SimpleNamespace(**opt.__dict__, **opt_extra.__dict__)       # conactenate all
    options
69.   opt.phase = 'train' if train else 'test'
70.   opt.loaded_latest_iter=None;
71.
72.   return opt
73.
74.
75.
76.
77.
78. def set_dataset_default_lm(opt):
79.   #set default values based on given dataset
80.   #changes the options object
81.   if opt.dataset_mode == "ade20k":
82.       opt.lambda_labelmix=10.0
83.       opt.EMA_decay=0.9999
84.   if opt.dataset_mode == "cityscapes":
85.       opt.lr_g=0.0004
86.       opt.lambda_labelmix=5.0
87.       opt.freq_fid=2500
88.       opt.EMA_decay=0.999
89.   if opt.dataset_mode == "coco":
```

```
90.       opt.lambda_labelmix=10.0
91.       opt.EMA_decay=0.9999
92.       opt.num_epochs=100
93.
94. def load_iter(opt):
95.     #gets the iteration from which to load the saved model, based on whether to get the
    best or the latest model.
96.     #returns an int which represents the the iteration of the model to load.
97.     if opt.which_iter == "latest":
98.         with open(os.path.join(opt.checkpoints_dir, opt.name, "latest_iter.txt"), "r")
    as f:
99.             res = int(f.read())
100.             return res
101.      elif opt.which_iter == "best":
102.          with open(os.path.join(opt.checkpoints_dir, opt.name, "best_iter.txt"), "r")
    as f:
103.             res = int(f.read())
104.             return res
105.      else:
106.          return int(opt.which_iter)
107.
108.  def configure_arguments(custom_arguments,train=False):
109.    #get the options arguments based on whether to train the model, and the custom
    arguments function.
110.    #set default values
111.    opt = get_default_opt(train)
112.    custom_arguments(opt)
113.    if train:
114.      set_dataset_default_lm(opt)
115.      if opt.continue_train:
116.        update_options_from_file(opt)
117.      custom_arguments(opt)
118.
119.
120.    utils.fix_seed(opt.seed)
121.    print_options(opt, train)
122.    if train:
123.      opt.loaded_latest_iter = 0 if not opt.continue_train else load_iter(opt)
124.    if train:
125.      save_options(opt, train)
126.    return opt
127.
128.
129.  # need to change the other functions
130.  def update_options_from_file(opt):
131.    #update the options object based on the options of the saved model.
132.    #changes the opt variable.
133.    new_opt = load_options(opt)
134.    for k, v in sorted(vars(opt).items()):
135.      if hasattr(new_opt, k) and v != getattr(new_opt, k):
136.        new_val = getattr(new_opt, k)
137.        setattr(opt, k, new_val)
138.
139.
140.  def print_options(opt, train):
141.      #prints to the screen the current options object alongside the default values if
    they are not the same.
142.      message = ''
143.      message += '----------------- Options ---------------\n'
144.      default_opt = get_default_opt(train)
145.      for k, v in sorted(vars(opt).items()):
146.          comment = ''
147.
148.          default = getattr(default_opt, k)
149.          if v != default:
150.              comment = '\t[default: %s]' % str(default)
151.          message += '{:>25}: {:<30}{}\n'.format(str(k), str(v), comment)
152.      message += '----------------- End -------------------'
153.      print(message)
```

```
154.
155.  def save_options(opt, train):
156.      #saves the options object to the checkpoints directory as a txt file and as a
    pickle.
157.      path_name = os.path.join(opt.checkpoints_dir,opt.name)
158.      os.makedirs(path_name, exist_ok=True)
159.      default_opt = get_default_opt(train)
160.      with open(path_name + '/opt.txt', 'wt') as opt_file:
161.          for k, v in sorted(vars(opt).items()):
162.              comment = ''
163.              default = getattr(default_opt, k)
164.              if v != default:
165.                  comment = '\t[default: %s]' % str(default)
166.              opt_file.write('{:>25}: {:<30}{}\n'.format(str(k), str(v), comment))
167.
168.      with open(path_name + '/opt.pkl', 'wb') as opt_file:
169.          pickle.dump(opt, opt_file)
170.
171.
172.
```

קובץ my_dataloaders.py

קובץ המכיל את המחלקה שטוענת את מבנה הנתונים.

נמצא תחת ה root_project_directory.

```
1.  from OASIS.dataloaders.dataloaders import get_dataset_name
2.  from create_panoptic import _create_panoptic_label
3.  from create_panoptic import _read_segments
4.  import torch
5.  from constants import root_project_directory
6.  import os
7.  from PIL import Image
8.  from torchvision import transforms as TR
9.  import random
10. import numpy as np
11.
12.
13. def get_dataloaders(opt):
14.     #creates the dataset and returns the dataloaders for the train and validation sets.
15.     #only cityscapes dataset is currently supported.
16.     dataset_name   = get_dataset_name(opt.dataset_mode)
17.
18.     dataset_train = CityscapesDataset(opt, for_metrics=False)
19.     dataset_val   = CityscapesDataset(opt, for_metrics=True)
20.
21.     print("Created %s, size train: %d, size val: %d" % (dataset_name,
    len(dataset_train), len(dataset_val)))
22.
23.     dataloader_train = torch.utils.data.DataLoader(dataset_train, batch_size =
    opt.batch_size, shuffle = True, drop_last=True)
24.     dataloader_val = torch.utils.data.DataLoader(dataset_val, batch_size =
    opt.batch_size, shuffle = False, drop_last=False)
25.
26.     return dataloader_train, dataloader_val
27.
28.
29. class CityscapesDataset(torch.utils.data.Dataset):
30.     # The cityscapes dataset class is responsible for listing the the paths to all the
    images, and in turn give proper tensors created from the images.
31.     def __init__(self, opt, for_metrics):
32.         #initiates the dataset with all the constants, and lists all the images and
    labels paths.
33.         opt.load_size = 512
34.         opt.crop_size = 512
35.         opt.label_nc = 34
36.         opt.contain_dontcare_label = True
```

```
37.         opt.cache_filelist_read = False
38.         opt.cache_filelist_write = False
39.         opt.aspect_ratio = 2.0
40.         opt.semantic_nc = opt.label_nc + 1    # label_nc + unknown
41.
42.         self.opt = opt
43.         self.for_metrics = for_metrics
44.         self.images, self.labels, self.paths = self.list_images()
45.
46.
47.     def __len__(self,):
48.         #returns the number of examples in the dataset.
49.         return len(self.images)
50.
51.     def __getitem__(self, idx):
52.         #creates an example and returns it as a tensor.
53.         image = Image.open(os.path.join(self.paths[0], self.images[idx])).convert('RGB')
54.         label = Image.open(os.path.join(self.paths[1], self.labels[idx]))
55.         if self.opt.segmentation=="panoptic":
56.             #for panoptic segmentation, create panoptic IDs as described in the paper.
57.             label_arr = np.asarray(label, dtype=np.uint32)
58.             mask = label_arr < 1000
59.             mask = (mask * 999) + 1
60.             label_arr = mask * label_arr
61.             label = Image.fromarray(label_arr.astype(np.uint32))
62.
63.         image, label = self.transforms(image, label)
64.         if (not self.opt.segmentation=="panoptic"):
65.           label = label * 255
66.         return {"image": image, "label": label, "name": self.images[idx]}
67.
68.     def list_images(self):
69.         #lists all of the paths of the images and the labels.
70.         mode = "val" if self.opt.phase == "test" or self.for_metrics else "train"
71.         images = []
72.         path_img = os.path.join(self.opt.dataroot, "leftImg8bit", mode)
73.         for city_folder in sorted(os.listdir(path_img)):
74.             cur_folder = os.path.join(path_img, city_folder)
75.             for item in sorted(os.listdir(cur_folder)):
76.                 images.append(os.path.join(city_folder, item))
77.         labels = []
78.         path_lab = os.path.join(self.opt.dataroot, "gtFine", mode)
79.         file_extension = "_gtFine.png"
80.         for city_folder in sorted(os.listdir(path_lab)):
81.             cur_folder = os.path.join(path_lab, city_folder)
82.             for item in sorted(os.listdir(cur_folder)):
83.                 if item.find("labelIds") != -1 and self.opt.segmentation=="semantic":
84.                     #if semantic, get the labelIds annotation.
85.                     labels.append(os.path.join(city_folder, item))
86.                 if item.find("instanceIds") != -1 and self.opt.segmentation=="panoptic":
87.                     #if panoptic, get the instanceIds annotation.
88.                     labels.append(os.path.join(city_folder, item))
89.
90.         assert len(images)  == len(labels), "different len of images and labels %s - %s"
    % (len(images), len(labels))
91.         return images, labels, (path_img, path_lab)
92.
93.     def transforms(self, image, label):
94.         #transfroms a PIL Image object into an augmented, resized and normalized tensor.
95.         #return a tensor of the image and a tensor of the label.
96.         assert image.size == label.size
97.         # resize
98.         new_width, new_height = (int(self.opt.load_size / self.opt.aspect_ratio),
    self.opt.load_size)
99.         image = TR.functional.resize(image, (new_width, new_height), Image.BICUBIC)
100.          label = TR.functional.resize(label, (new_width, new_height), Image.NEAREST)
101.          # flip
102.          if not (self.opt.phase == "test" or self.opt.no_flip or self.for_metrics):
103.              if random.random() < 0.5:
```

```
                image = TR.functional.hflip(image)
                label = TR.functional.hflip(label)
        # to tensor
        image = TR.functional.to_tensor(image)
        label = TR.functional.to_tensor(label)
        # normalize
        image = TR.functional.normalize(image, (0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
        return image, label
```

קובץ constants.py

הקובץ מכיל רק את הכתובת של התיקייה הראשית בפרוייקט, ה root_project_directory.
כלל הקבצים מייבאים קובץ זה על מנת לקבל את מיקום התיקייה הראשית. כאשר נעביר את תיקיית הפרוייקט ממקום למקום, נצטרך לשנות רק את הערך בקובץ זה, במקום לשנות את המשתנה בכל אחד מהקבצים בנפרד.
נמצא תחת ה root_project_directory.

```
root_project_directory = "/content/drive/MyDrive/ProjectGAN"
```

קובץ simple_application.ipynb

הקובץ הינו מחברת אשר משמש כשרת עבור האפליקציה שנבנתה ב anvil.
נמצא תחת ה root_project_directory.

המחברת מכילה פונקציה אשר מקבלת מהאפליקציה תמונה סמנטית ויוצרת ממנה בעזרת המודל הדפולטיבי המאומן תמונה פוטוריאלסטית אותה היא שולחת חזרה לאפליקציה.

את המחברת ניתן גם למצוא כאן: https://drive.google.com/file/d/1la8Aw-VsryxfJp_Tlk2-awP_W9B6IYHA/view?usp=sharing

קובץ MyGoodPanopticProject.ipynb

הקובץ הינו מחברת אשר דרכה ניסיתי לאמן את המודל הפנופטי.
נמצא תחת ה root_project_directory.

הקובץ עצמו מכיל לא מעט קטעי קוד שאינם רלבנטיים, ובאופן כללי הוא כלל לא מסודר או מתועד באופן ברור. הוא שימש אותי בשלל ניסויים כדי להבין את קוד המקור ודרכי הפעולה שלו והוא לא בהכרח מעודכן עד הסוף בכל הפרטים.

ניתן להציץ אך לא מומלץ: https://colab.research.google.com/drive/1emL9ZF8JqqsmhAGu0Qw8TKruIYKOGV_c?usp=sharing

קובץ my_good_panoptic_project.ipynp

הקובץ הינו מחברת המאפשר אימון של כל מודל.
נמצא תחת ה root_project_directory.

המחברת עצמה היא מסודרת ומאפשרת להריץ את תוכנית האימון של המודל.
כל הקוד שנמצא במחברת כבר כתוב בתיק פרוייקט ואין בו שום דבר חדש.

תיקיית pretrained_checkpoints

תיקייה בה נשמרים כל המודלים שאנו מאמנים, יחד עם הסטטיסטיקות שלהם ודוגמאות לתמונות שנוצרו מתוך דוגמאות האימון. כל מודל ישמר בתיקייה משלו בשם שניתן לו.
נמצאת תחת root_project_directory.

תיקיית cityscapes

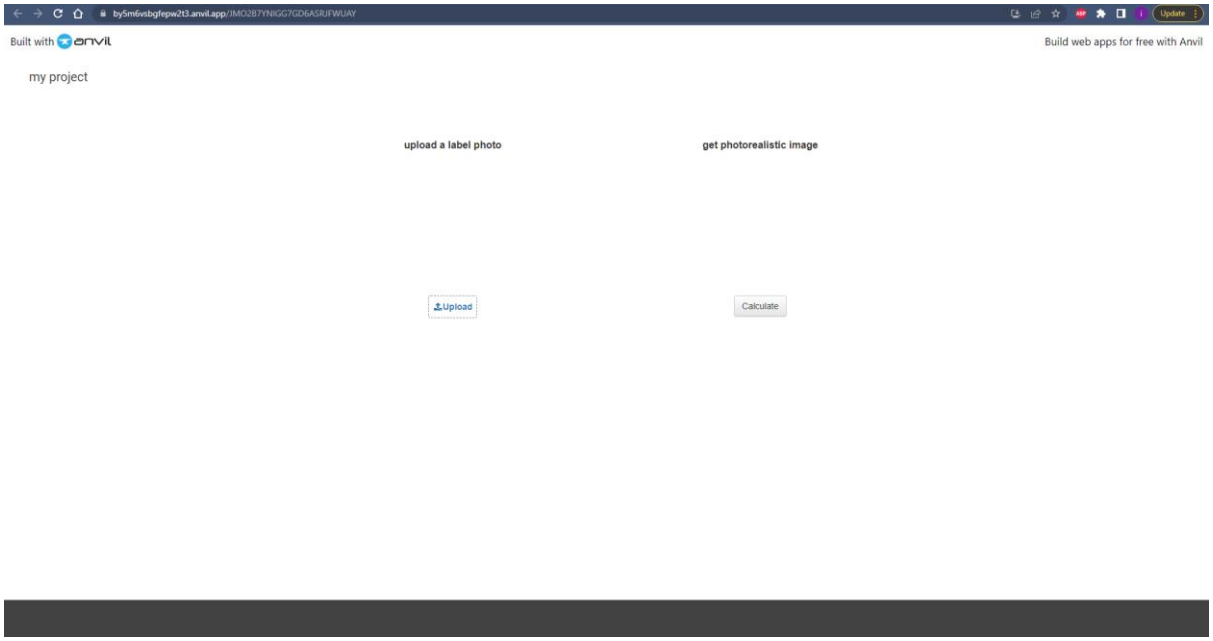
התיקייה בה נשמור את מבנה הנתונים cityscapes כפי שמתואר בפרק איסוף הנתונים.
נמצאת תחת תיקיית datasets שנמצאת תחת ה root_project_directory.

מדריך למשתמש

https://BY5M6VSBGFEPW2T3.anvil.app/JMO2B7YNIGG7GD6ASRJFWUAY

על מנת להשתמש באפליקציה, יש להריץ קודם את המחברת simple_application.ipynb על ידי פתיחתה בקולאב ולחיצה על Runtime->run all.

אחר כך, להיכנס לקישור שנמצא כאן למעלה, אז יפתח לנו החלון הבא:



נשים לב שמוצבים בפנינו שתי כפתורים. לחיצה על הכפתור השמאלי upload, תפתח לנו את האפשרות להעלות קובץ לבחירתנו. נרצה להעלות קובץ אשר מכיל את הקידודים של המחלקות הסמנטיות, באופן אידיאלי קובץ labelId מתוך מבנה הנתונים cityscapes.

לאחר מכן, לחיצה על הכפתור הימני calculate תציג עיגול מסתובב עד אשר יופיע על המסך תמונה פוטוריאליסטית שנוצרה על ידי המודל.

במקרה וישנה בעיה, נקבל הודעת שגיאה בצד ימין למטה. לחיצה עליה תספק פירוט של הבעיה.

במידה ונקבל הודעת שגיאה כזאת:

משמעותה היא שהמחברת אינה רצה. יש לחזור אל המחברת ולוודא שהיא מחוברת ל runtime וללחוץ שוב על run all.

סיכום אישי

הנושא באופן כללי של למידת מכונה ולמידה עמוקה בפרט, לטעמי מאוד מעניין ומסקרן. במהלך הפרוייקט נהניתי לעבור על כל מיני מחקרים שונים ולראות שיטות שונות שמסוגלות להגיע לתוצאות מדהימות. במיוחד בנושא של GAN, בולט המודל Dall-e 2 אשר מסוגל ליצור תמונות איכותיות ויצירתיות מתוך טקסט בלבד.

העבודה עצמה על הפרוייקט הייתה לא פשוטה ודרשה מאמץ והשקעה רבה. בייחוד היה קשה עבורי להבין את הקוד והשיטות בהם הם נקטו על מנת לייצג תמונה סמנטית ולהתאים אותה עבור המודל. לקח לי גם הרבה זמן להבין איך מבנה הנתונים בנוי, מפני שלדעתי לא היה מספיק תיעוד של הקוד והקידודים של התמונות.

בסופו של דבר, אני מעט מאוכזב שלא הצלחתי לאמן את המודל כפי שרציתי ואולי אם הייתי מתחיל לעבוד על הפרוייקט קודם, הייתי יכול לנסות דרכים נוספות. למרות זאת, אני לוקח איתי מהפרוייקט ידע ונסיון רב על מגוון נושאים רחב. בין היתר למדתי על GAN-ים (השלמתי קורס בקורסרה על הנושא), הכרתי יצירת אפליקציה עם anvil, למדתי את pytorch, ובעיקר נחשפתי למגוון מאמרים בכל מיני נושאים והבנתי שיכולים להיות מספר דרכים שונות לגשת לאותה הבעיה.

בנוסף, למדתי מהפרוייקט הזה שאני צריך לתכנן יותר טוב מה המטרות שלי ומה אני מנסה להשיג לפני שאני מתחיל לעבוד, מפני שמרוב כל המידע שמצאתי באינטרנט הייתי די מבולבל באשר לרעיון מאחורי הפרוייקט. אם הייתי מתחיל היום לעבוד על הפרוייקט, הייתי מתחיל לעבוד עליו מוקדם יותר, מפני שאימון המודלים לוקח זמן ממושך כמו גם הגיבוש של הרעיון ותהליך הלמידה.

העבודה עבורי הייתה יכולה להיות יעילה יותר לדעתי אם הייתה אפשרות לעבודה בקבוצות או בזוגות, מפני שאז גם יש יותר מוטיבציה לעבוד על הפרוייקט וגם יש מישהו שמבין יחד איתך את הפרוייקט ואפשר להתייעץ איתו ולגבש יחד איתו את הרעיון עבור הפרוייקט.

כחקר עצמי הייתי רוצה לבדוק את השיטות שהצעתי לבדוק על מנת לגרום למודל כן ללמוד תמונות פנופטיות. בנוסף לכך, יהיה מעניין לבדוק אם אפשר לשנות את המודלים לטרנספורמרים למשל. בכל אופן, אני בהחלט רואה את עצמי ממשיך לחקור בנושא הלמידה העמוקה, ואני אשמח להתעמק ולהתעסק בו בעתיד.

ביבליוגרפיה

Cordts, M. a. (2016). *The Cityscapes Dataset for Semantic Urban Scene Understanding*. Retrieved from Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.: https://www.cityscapes-dataset.com/

GAN — What is Generative Adversarial Networks GAN? (2018, June 19). Retrieved from medium: https://jonathan-hui.medium.com/gan-whats-generative-adversarial-networks-and-its-application-f39ed278ef09

GAN (למידה חישובית). אוחזר מתוך ויקיפדיה:
https://he.wikipedia.org/wiki/GAN_(%D7%9C%D7%9E%D7%99%D7%93%D7%94_%D7%97%D7%99%D7%A9%D7%95%D7%91%D7%99%D7%AA)

Mohan, R., & Valada, A. (2021). *EfficientPS: Efficient Panoptic Segmentation*. Retrieved from arXiv: https://arxiv.org/pdf/2004.02307v3.pdf

Shiledarbaxi, N. (2021, Februar 8). *Semantic vs Instance vs Panoptic: Which Image Segmentation Technique To Choose*. Retrieved from analyticsindiamag: https://analyticsindiamag.com/semantic-vs-instance-vs-panoptic-which-image-segmentation-technique-to-choose/

Wang, T.-C. a.-Y.-Y. (2017). *High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs*. Retrieved from arXiv: https://doi.org/10.48550/arXiv.1711.11585

Zhou, S. (n.d.). *Generative Adversarial Networks (GANs) Specialization*. Retrieved from Coursera: https://www.coursera.org/specializations/generative-adversarial-networks-gans