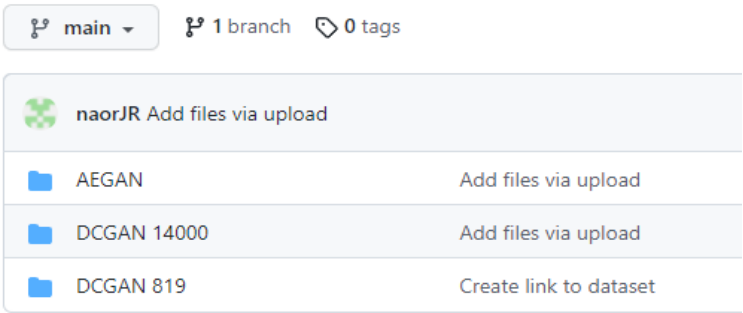
POKEMON GAN

Naor Cohen 208560086 Sagi Gov 204123681

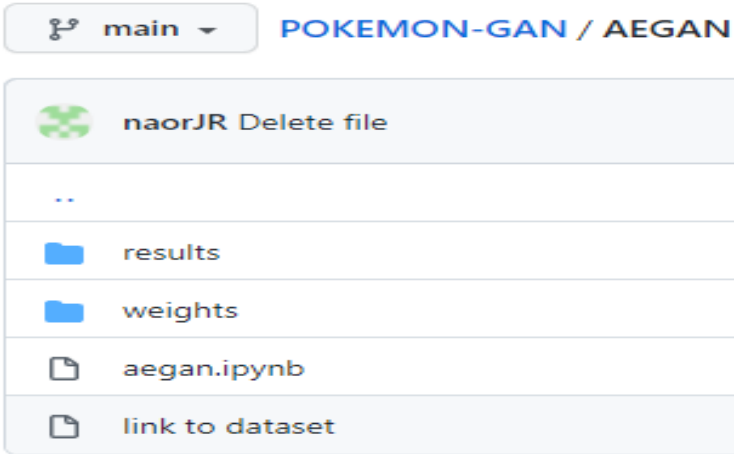
בפרויקט זה אנו ייצרנו פוקימונים סינטטיים באמצעות GAN . בדוח זה אנו הולכים לפרט על מה שעשינו כסוג של תרשים זרימה המתאר את ההתקדמות שלנו ואת העבודה לאורך הפרויקט. אנו ניסינו 3 מודלים שונים של רשת שכל אחד מהם מוגש בקובץ ipynb נפרד (כך למשל הארכיטקטורה הראשונה DCGAN819 תוגש בקובץ dcgan819.ipynb). נציין שבדוח זה נתמקד בדברים היותר מהותיים ולא נרד ממש לפרטים קטנים כמו איזה אופטימייזר בחרנו וכו' (Adam) אבל את כל זה יש כמובן בקבצי ipynb.

למען נוחות סידרנו הכל בגיט לפי התמונה הבאה -



כאשר כל תיקייה מכילה סוג שונה של מודל שניסינו ועליהם אנחנו הולכים להרחיב בדוח זה.

הערה – בכל תיקייה אפשר לראות בצורה מסודרת את הדברים הבאים :

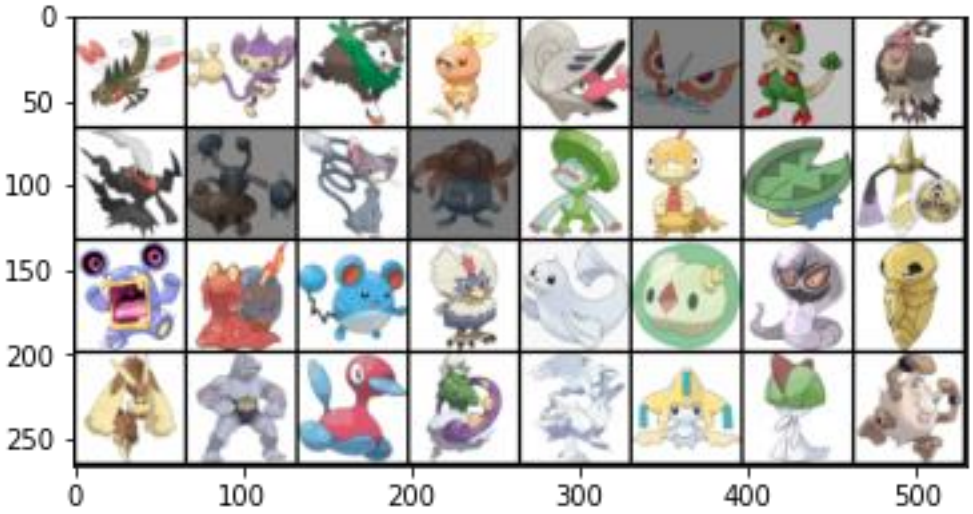


ככה שאם תרצה אפשר לראות גם את הקוד וגם את התוצאות בצורה ברורה יותר – רק צריך שאני אוסיף אותך, הכל בכל מקרה נמצא בקבצי ipynb שמוגשים וכעת נצלול לארכיטקטורה הראשונה בה התחלנו.

DCGAN 819

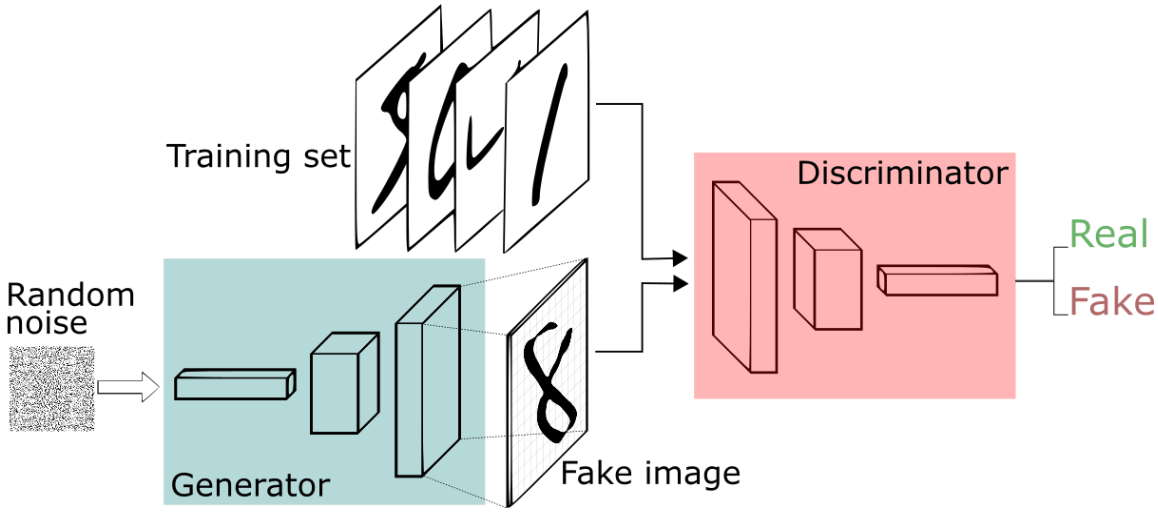
הארכיטקטורה הראשונה שהכנו בפרויקט זה נקראת dcgan 819 על שם סוג הרשת איתה בחרנו לעבוד - Deep convolutional Generative Adversarial Network וגודל ה dataset (819).

השיטה שבה קראנו את ה dataset היא באמצעות Imagefolder מודל מאוד נוח בו אתה יכול לקרוא קבצים מתיקייה מסוימת ותוך כדי להפעיל עליהם טרנספורמציות שונות וככה זה מאוד נוח וקל לעבד את מאגר התמונות שלנו. 3 עיבודים שבוצעו על התמונות היו – חיתוך התמונה מהמרכז לגודל 64 על 64 , המרה לטנזור והמרת הערכים מ [0,255] ל [-1,1]. בנוסף מפני שיש לנו מספר קטן של דגימות הגדלנו את כמות התמונות פי 3 על ידי שיקוף אנכי של התמונות ומשחק עם בהירות, ניגודיות וסטורציה.

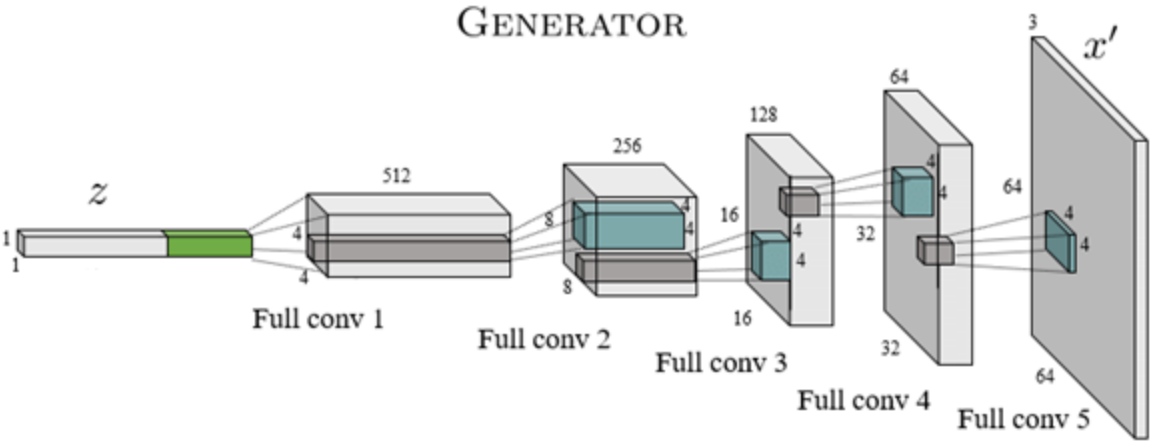


דגימה של 32 תמונות מתוך ה dataset

ארכיטקטורה :

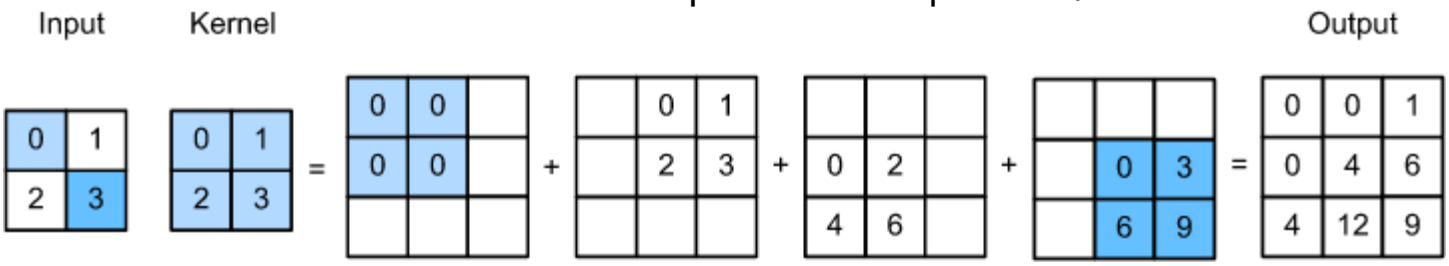


בצורתו הכללית ביותר המודל שממומש בחלק זה הינו GAN פשוט בו הגנרטור יוצר תמונה מרעש והדיסקרימנטור מסווג האם התמונה אמיתית או סינטטית, אין פה יותר מדי מה להרחיב בנושא כעת נצלול יותר למטה ונתמקד בנקודות המעניינות והחשובות ונתחיל בלפרט על מבנה הגנרטור.



המודל שמימשנו הוא בדיוק כמו בתרשים כאשר המעבר משכבה לשכבה נעשה על ידי - ConvTranspose2d – שכבה זו בעצם לוקחת מטריצת כניסה ומפעילה פילטר על כל איבר במטריצה כך שכל פיקסל יומר למטריצה בגודל הקרנל שתכיל את הערך שלו כפול הקרנל עצמו וזה ימוקם במטריצה יותר גדולה בהתאם למיקום הפיקסל, לבסוף מחברים את כל התוצאות של כל הפיקסלים לקבלת מוצא השכבה.

למשל עבור מטריצה בגודל 2 על 2 אם נרצה להגדיל אותה במימד אחד נבחר פילטר בגודל 2 על 2 ואז נקבל את מה שניתן לראות פה :

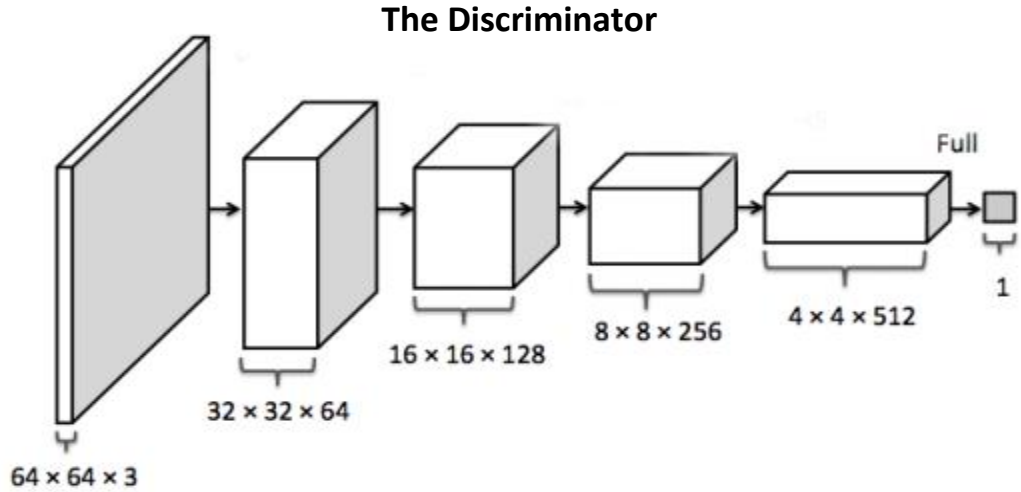


ניתן עוד דוגמא למשל איך עברנו מ $512x4x4$ ל $256x8x8$ - קודם כל ה 256 זה מספר הפילטרים שאנו רוצים להפעיל בדיוק כמו קונבולוציה רגילה (מספר ערוצי המוצא) וה 512 זה מספר ערוצי הכניסה (כאמור כמו קונבולוציה רגילה כל ערוץ כניסה עובר פילטר באופן בלתי תלוי בשאר הערוצים ואז תוצאת קונבולוציה של כל הערוצים נסכמים וזה תקף גם פה). כעת מה שמעניין זה המעבר מ 4 על 4 ל 8 על 8 – זה נעשה על ידי בחירת קרנל בגודל $4x4$ ו stride = 2 ו padding = 1 בהינתן ערוץ כניסה בגודל $4x4$ נקבל מוצא 8 על 8 לפי הנוסחה הבאה :

$$H_{out} = (H_{in} - 1) \times \text{stride} - 2 \times \text{padding} + \text{kernel\_size}$$

כל שכבה עוברת batch normalization והפונקציה היא relu פרט לשכבה האחרונה שם השתמשנו ב tanh על מנת לקבל ערכים בתחום [-1,1]. וכעת נמשיך לארכיטקטורה המקבילה.

The Discriminator



רשת זו כאמור היא רשת קונבולוציה פשוטה שמחזירה הסתברות האם התמונה סינטטית או אמיתית , זו רשת בסיסית,המעבר בין שכבה לשכבה נעשה על ידי פעולת קונבולוציה ואין מה להרחיב עליה יותר מדי אולי מלבד שהפונקציה שמופעלת לאחר כל שכבה פרט לאחרונה היא :

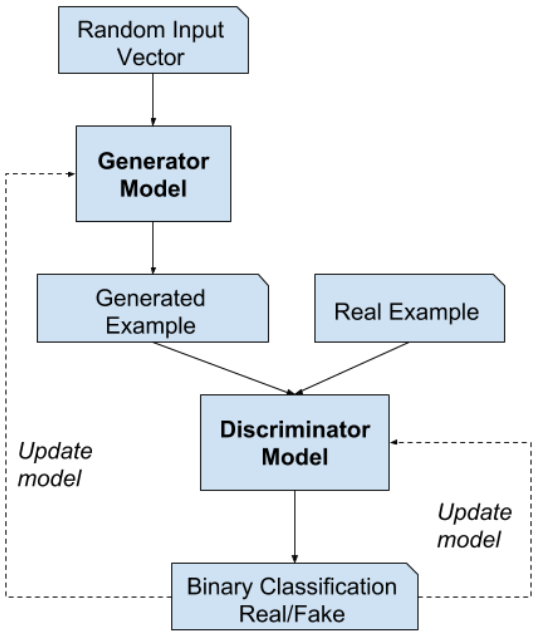
$$\text{LeakyReLU}(x) = \max(0, x) + 0.2 * \min(0, x)$$

זו בעצם פונקציה דומה ל *relu* רק שעבור ערכים קטנים מ 0 לא לוקחים 0 כמו שקורה ב *relu* -

$$\mathrm{ReLU}(x) = \max(0, x)$$

בנוסף הפונקציה של השכבה האחרונה היא sigmoid כדי להחזיר הסתברות, בקיצור רשת קונבולוציה סטנדרטית בדיוק כמו שרואים באיור הפרט היחידי שאולי חשוב לציין הוא שהורדת רוחב ואורך התמונה נעשו על ידי $\mathrm{stride} = 2$ (באחת העבודות בית השתמשנו ב maxpooling למטרה זו).

כעת נסביר על האימון :



הרעיון באימון הפונקציה הוא לאמן את שתי הרשתות יחד – כלומר שתיהן משפרות אחת את השנייה , התפקיד של הדיסקרימנטור הוא להבדיל בין תמונה אמיתית לתמונה שהגנרטור יצר ולכן כדי לאמן אותו נתייג תמונה אמיתית כ *true* ותמונה סינטטית כ *false* וכמובן נשתמש בפונקציית השגיאה *BCE*.

$$BCE = binary\ cross\ entropy\ = y * \log(x) + (1 - y) * \log\,(1 - x)$$

התפקיד של הגנרטור הוא לרמות את הדיסקרימנטור ולגרום לו לחשוב שהתמונה שייצר היא אמיתית ולכן פה נתייג את התמונות שהגנרטור ייצר כ *true* נכניס אותם לדיסקרימנטור ושוב בעזרת *BCE* נבדוק כמה טוב הוא הצליח לעבוד עליו. כאמור בכל שלב אנו מעדכנים רק את המשקולות של הרשת אותה אנו מלמדים

כלומר באימון הגנרטור אנו כמובן משתמשים בדיסקרימנטור אך לא נעדכן את המשקולות שלו (כלומר נעשה רק *opt_generator.step()*) וכנ"ל הפוך.

כעת לפני שעוברים על התוצאות נסביר על 3 דברים שביצענו על מנת לשפר את הארכיטקטורה שלנו :

1.באימון של ה *discriminator* השתמשנו ב - *smoothing label* – במקום לתייג *true* כ '1' ו *false* כ '0' מה שעשינו זה תייגנו בצורה הבאה :

$$true = 0.85 + 0.15 * N(0,1)\ ;\ N = normal\ distribution$$

$$false = 0.15 + 0.15 * N(0,1)$$

הסיבה שעשינו זו הינה שזה מקשה על ה *discriminator* וגורם לו לא להתכנס מהר מדי באיטרציות הראשונות (כאמור כל המטרה פה זה הגנרטור ולא הוא)

2. הוספת רעש – שוב באימון ה *discriminator* בשביל אותה מטרה מה שעשינו זה לקחת 5 אחוז מהדגימות ולהפוך אותן לשגויות (כלומר לקחת למשל דגימות של תמונות אמיתיות ולתייג אותן כסינטטיות)

3. *spectral norm* – גם פה זה בארכיטקטורה עצמה של ה *discriminator* בכל שכבת קונבולוציה

$$\boldsymbol{W}SN = \frac{\boldsymbol{W}}{\sigma(\boldsymbol{W})}, \sigma(\boldsymbol{W}) = \max_{h:\, h \neq 0} \frac{||Wh||_2}{||h||_2}$$

זה נרמול המשקולות כפי שניתן לראות וגם פה זה כדי לייצב את הדיסקרימנטור (אמנם בנימה אישית אנחנו לא הרגשנו שזה תרם הרבה אבל השארנו את זה).

מבחינת ההיפר פרמטרים אפשר לראות בטבלה עם מה בחרנו לעבוד:

learning rate	0.0002
batch size	32
latent shape	128
kernel size	4

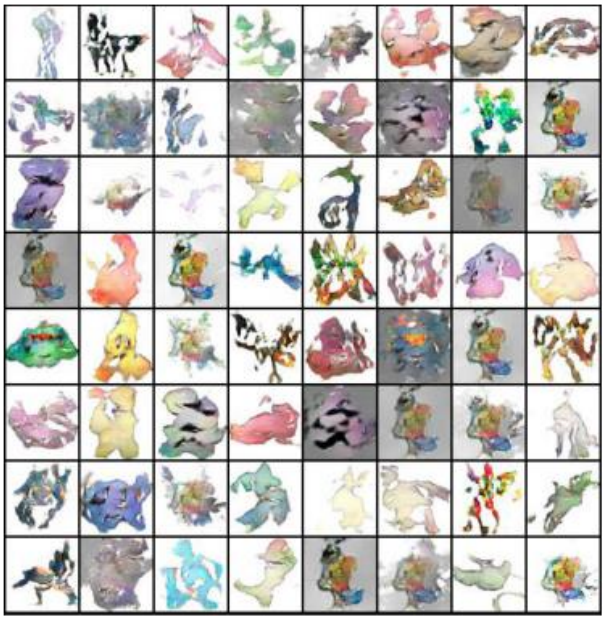
מבחינת גודל הקרנל זה אותו גודל גם לגנרטור וגם לדיסקרימנטור, לא בוצע חיפוש היפר פרמטרים (בסוף נסביר למה).

תוצאות :

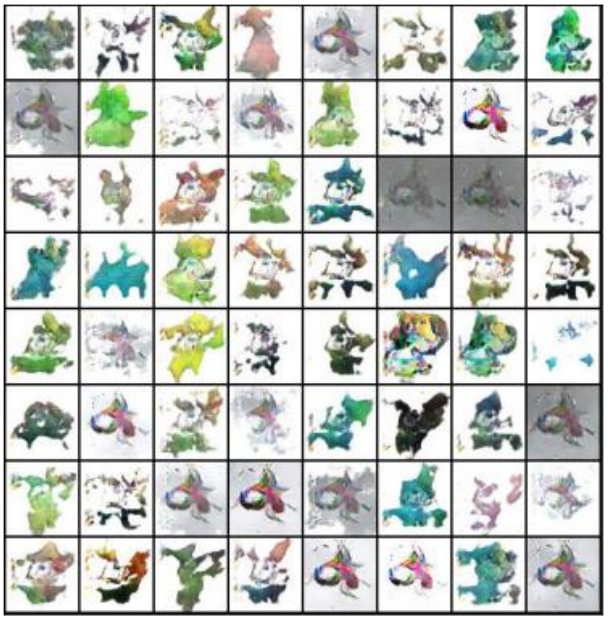
100 Epoch

200 Epoch

300 Epoch



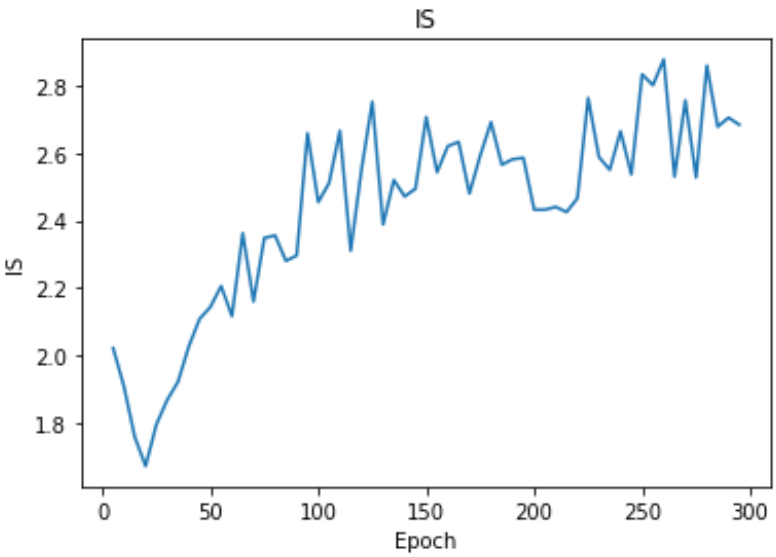
400 Epoch

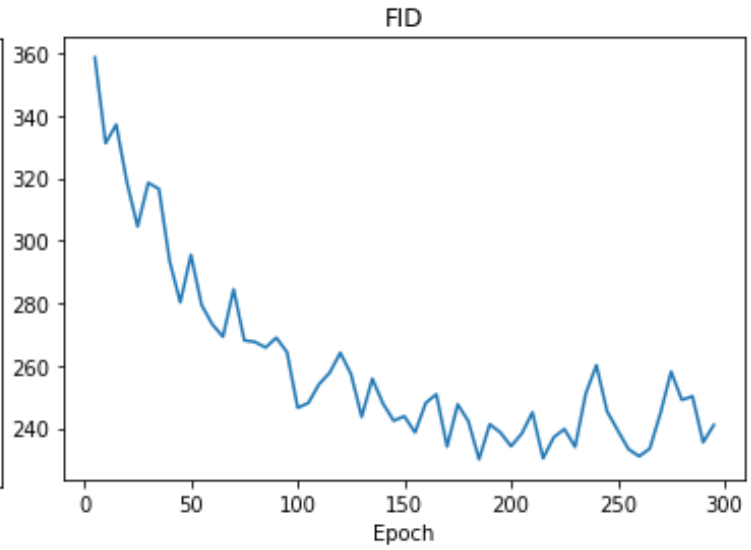


(קצת קשה לראות את התוצאות ב pdf ,בכל מקרה בקובץ dcgan819 ipynb
אפשר לראות אותם יותר גדול וגם כמו שציינו בהתחלה ב git יש הכל מסודר)

באופן אישי לא כל כך התרשמנו מהתוצאות האלו מבחינת מראה , כן היה שיפור לעומת מה שהתחלנו איתו אבל לא מספק.

נוסיף ניתוח גרפי של FID ו Inception score (מדדים מוכרים ומיהמנים בתחום הערכת ביצועי GAN):





מדד ה FID מחושב בצורה הבאה :

$$d^2 = \left|\left|mu_1 - mu_2\right|\right|^2 + Tr\left(C_1 + C_2 - 2 * \sqrt{(C_1 * C_2)}\right)$$

כאשר $mu_1\, mu_2$ אלו התוחלת של התמונות האמתיות והסינטטיות בהתאמה ו C זוהי מטריצת הקוואריאנס.

כן רואים שיפור בשני המדדים – נציין שחישוב שני המדדים נעשה באמצעות רשת עזר Inception v3 כי כך משתמשים במדד זה – במקום להשוות פיקסל פיקסל ה FID משווה את הממוצע והשונות של שכבה אחת לפני הסוף של רשת זו. ה Inception v3 הינה רשת מוכרת שמבצעת סיווג תמונה לפי 1000 קטגוריות שונות ובגלל זה לוקחים שיכבה אחת לפני הסוף כי אנו לא רוצים את הפרדיקציה של הרשת (השכבה הסופית היא ווקטור הסתברות בהתאם למספר הקטגוריות – 1000).

לגבי מדד IS אנו כן רוצים את כל הרשת כולל השכבה האחרונה כי מדד זה בודק כמה התמונה "איכותית" ותמונה איכותית זו תמונה שהרשת מסוגלת לחזות אותה בשלמות כלומר שמכילה אובייקט מסוים (ואז הווקטור התפלגות של מוצא הרשת

יהיה [1,0,0...] וקטור יחידה) ובנוסף הוא בודק כמה הרשת מגוונת – האם היא חוזה מספר אובייקטים או כל פעם אותו אובייקט- כך למשל אם כל פעם נקבל את אותו ווקטור הסתברות - זה לא טוב כי זה אומר שאנו מייצרים את אותו אובייקט. כלומר IS מושלם של N תמונות יהיה שווה ל N וזה אומר שהרשת inception v3 חזתה ווקטור יחידה שונה לכל תמונה.

לגבי מימוש ונוסחאות ניתן לראות בקוד – לא השתמשנו במשהו מובנה אבל כן מצאנו מימוש ידני שמחשב את זה טוב ובצורה מפורטת!

עוד נוסיף שמדד ה IS של התמונות האמיתיות הוא בעצמו נמוך : 2.7 מתוך 128 שזה מספר התמונות האמיתיות שבחרנו לבדוק כל פעם ולכן הוא לא באמת נותן לנו משהו פה – אפשר לשמוח כביכול שמבחינת מדד זה הצלחנו להשתוות בערכו למדד של התמונות האמתיות (בגרף רואים שהגענו ל 2.8) אבל זה לא נותן לנו כלום כי כאמור גם בתמונות האמתיות מדד זה נמוך – מה שכן זה מעיד על הקושי במשימה.

ערך נמוך במדד FID אומר שהתפלגות מאפייני התמונה זהים (כאמור אנחנו לא מודדים את זה על התמונות עצמן אלא על המוצא של השכבה לפני אחרונה של הרשת inception v3 כפי שציינו למעלה). במקרה שלנו אנו מתחילים בערך ממש גבוה (הגיוני) כי אנחנו משווים רעש בהתחלה. כן ניתן לראות שיפור וירידה לערך של 240 אחרי 200 epochs ולאחר מכן זה מתנדנד. עד לפה זה היה החלק הראשון של הפרויקט וכעת מה שרצינו לעשות זה לחקור מה יקרה עם dataset יותר גדול – יש הרבה דברים שיכולים להפיל Gan ודבר אחד זה גודל ה dataset. המוטיבציה לכך היה זה שניסינו לעבוד עם מאגר גדול של תמונות אנימה ואחרי מספר קטן של epochs קיבלנו תוצאות טובות :

מאגר המידע של תמונות האנימה הכיל 64000 תמונות ובנוסף ייצור פרצופים הרבה יותר קל מפוקימונים כי לפרצופים יש מאפיינים דומים (עיניים, אף פה וכו') בניגוד לפוקימונים.

Dcgan 14000

כעת שני האתגרים שעומדים לפנינו הם בדיקת הרשת עם מאגר מידע יותר גדול ולנסות למיין את הפוקימונים לפי מאפיינים דומים – ראינו שייצור דמויות אנימה עבד טוב וכאמור זה כי היה לנו הרבה תמונות וכל התמונות מכילות פחות או יותר מאפיינים דומים ולכן אולי אם נחלק פוקימונים לקבוצות של פוקימונים דומים זה יוביל לשיפור ביצועים (ספוילר – לצערנו אצלנו זה לא עזר : לא הצלחנו לחלק את הפוקימונים לקבוצות עם מאפיינים דומים נרחיב בהמשך בקצרה מה ניסינו).

לגבי מבנה הרשת ניהול ה dataset הכל אותו מבנה כמו הרשת למעלה (חוץ מהעובדה שכעת לא ביצענו מניפולציות כדי להגדיל את מאגר הנתונים כמו תמונת מראה וכו' שביצענו קודם) מה ששונה זה ה dataset עצמו המכיל 14000 תמונות, וכתוצאה מכך הגדלנו גם את ה batch size ל 64 לכן לא נסביר שוב ונעבור ישר לתוצאות :

25 Epoch

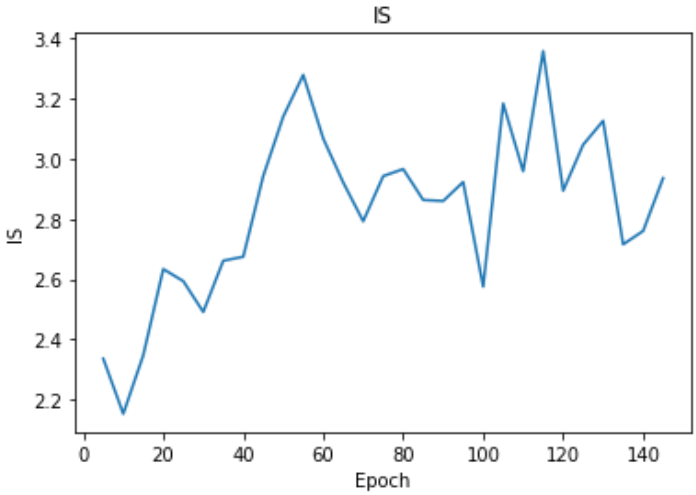
50 Epoch

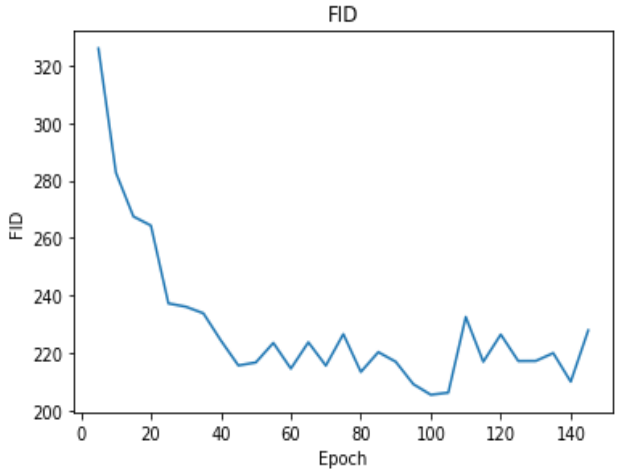
100 Epoch

150 Epoch

מבחינת העין אכן רואים שיפור בתוצאות – התמונות פחות מרוחות, הצורה יותר ברורה ויש פחות רעש. כאמור כל התוצאות נמצאות באופן מסודר ב Git וניתן לראות אותם בצורה יותר טובה, כולל וידאו שמראה את ההתקדמות לפי מספר ה epoch.

לגבי המדדים FID ו IS :





אנו רואים שיפור בשני המדדים האלו כמו שציינו מקודם לדעתינו ה FID מדד יותר טוב מה IS כי כאמור הוא מניב תוצאות טובות גם על מאגר הנתונים האמיתי. אנו רואים שהגענו ל FID של 210 שזה יותר טוב וכאמור גם בתמונות עצמם לדעתינו הייצור הרבה יותר טוב.

כעת נסביר בקצרה מה לא עבד עם חלוקת הפוקימונים, על מנת להמחיש נסתכל על פוקימונים מסוג מים :

אם נחלק את המאגר לפי סוג הפוקימון אז כפי שניתן לראות הדבר הבולט ביותר שנקבל זה צבע דומיננטי במקרה של פוקימוני מים – כחול. אבל מבחינת צורה אין שום קשר בין הצורות של הפוקימונים מאותו הסוג ולכן עדייו נקבל אותו דבר כמו מקודם או אפילו יותר גרוע כי בבחירת סוג אחד של פוקימון אנו מקטינים את מאגר הנתונים שלנו וחוץ מזה שכל התמונות שייצאו יהיו עם צבע דומיננטי לא נקבל הרבה. גם חלוקה לפי צבע לא תועיל מאותה סיבה חוץ מזה שאנחנו כן רוצים שילובי צבע בייצור הפוקימונים אבל בכל מקרה זה לא יועיל.

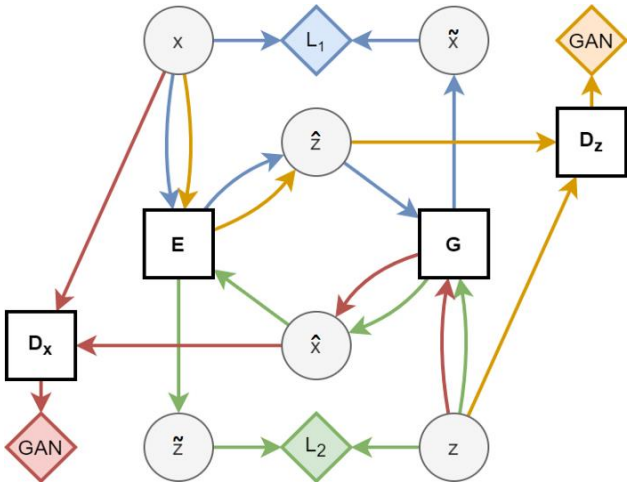
אנו חושב שהתמונה של פוקימוני המים ממחישה למה כל כך קשה לייצר פוקימונים בניגוד לתמונות של פרצופי אנימה מאחר ועבורם לכל המאגר נתונים יש צורה קבועה של פרצוף (שיער אוזניים פה ואף).

AEGAN

כעת בחרנו לעבוד עם ארכיטקטורה נוספת - Autoencoding Generative Adversarial Network.

כאמור ב Gan רגיל אנו אמורים לייצר תמונה מרעש כלומר כל ערך בווקטור הרעש תואם לערך יחיד כלשהו במישור התמונה. אך להפך זה לא מתקיים יש פיקסלים במישור התמונה שאין להם ערך תואם במישור הרעש כמו כן, יכולות להיות הרבה נקודות בווקטור הרעש שתואמות לאותו פיקסל וככה אין קשר חד חד ערכי דו כיווני, כלומר פיקסל במישור התמונה לא יהיה תואם לערך יחיד במישור הרעש אלא לכמה ערכים.

לכן ניצור את הארכיטקטורה הבאה :



x are the real images.

z is random noise.

$\hat{x} = G(z)$

$\hat{z} = E(X)$

$\tilde{x} = G(\hat{z}) = G(E(x))$

$\tilde{z} = G(\hat{x}) = E(G(z))$

$\boldsymbol{G}$ *is the generator network. It takes a latent vector z as input and returns an image x as output*

$\mathbf{E}$ *is the encoder network. It takes an image x as input and returns a latent vector z as output.*

$\boldsymbol{D_x}$ *is the image discriminator network. It takes an image x as input and returns the probability that x was drawn from the original dataset as output*

$\boldsymbol{D_z}$ *is the latent discriminator network. It takes a latent vector z as input and returns the probability that z was drawn from the latent distribution as output*

בעצם המטרה פה היא שכל נקודה במישור התמונה $x \in X$ תתאים לנקודה ייחודית במישור הרעש $z \in Z$ ולהפך כמובן (מה שקורה ב *Gan* רגיל).

על מנת לבצע זו מה שאנו עושים זה מוסיפים מקודד שיעשה בדיוק הפוך מהגנרטור, ייקח תמונה אמיתית ויהפוך אותה לרעש סינטטי וככה אנו יוצרים קשר ממישור התמונה למישור הרעש כלומר –

ב *Gan* הקודם שעבדנו איתו היה לנו $G: Z \to X$ גנרטור שממיר רעש לתמונה כעת יש לנו גם $E: X \to Z$ מקודד שממיר תמונה לרעש. וכעת בנוסף יש לנו שתי דיסקרימנטורים אחד לרעש (שמבדיל האם הרעש הוא רעש אמיתי או רעש סינטטי- רעש שנוצר כתוצאה מהפעלת המקודד על התמונה) ואחד רגיל לתמונה כמו שעבדנו עד עכשיו.

מבני הרשת –

לגבי מבנה הרשת הגנרטור והדיסקרימנטור של התמונה לא השתנו.

לגבי המקודד הוא מכיל שכבות קונבולוציה ולאחר מכן *fully connected* על מנת להמיר תמונה לווקטור רעש בגודל המתאים. הדיסקרימנטור של התמונה לוקח רעש ועל ידי רשת *fully connected* מחזיר הסתברות אם הרעש אמיתי או לא.

לטעמינו אין מה להרחיב על המבנים יותר מדי הם סטנדרטיים וניתן לראות בקוד בצורה יותר מפורטת וברורה.

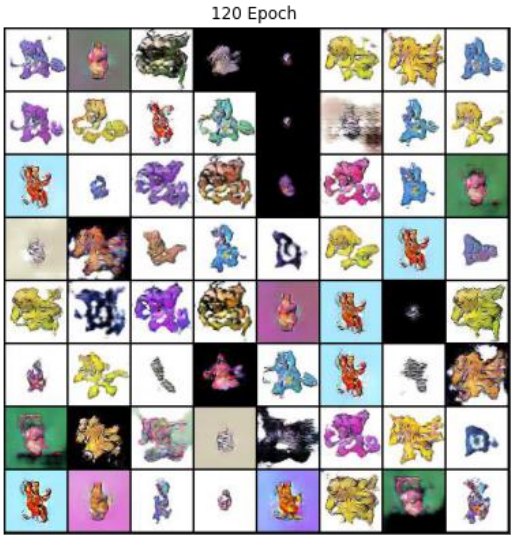
אימון :

אנחנו מאמנים ברעיון דומה לארכיטקטורה הקודמת כדי לאמן את D_x נשתמש ב *BCE* אך כעת אנו מתייגים כ *false* את $\hat{x}$ וגם את $\tilde{x}$ ואת x כמובן משאירים *true*. כמובן שכל הטכניקות לשיפור גם שהשתמשנו כגון *label smoothing* נשארות אותו דבר. על מנת לאמן את D_z זה אותו דבר רק עם ווקטור הרעש כלומר ווקטור הרעש שייצרנו באמצעות פונקציית *random* יהיה מתויג כ *true* ו $\hat{z}$ $\tilde{z}$ יהיו *false*.

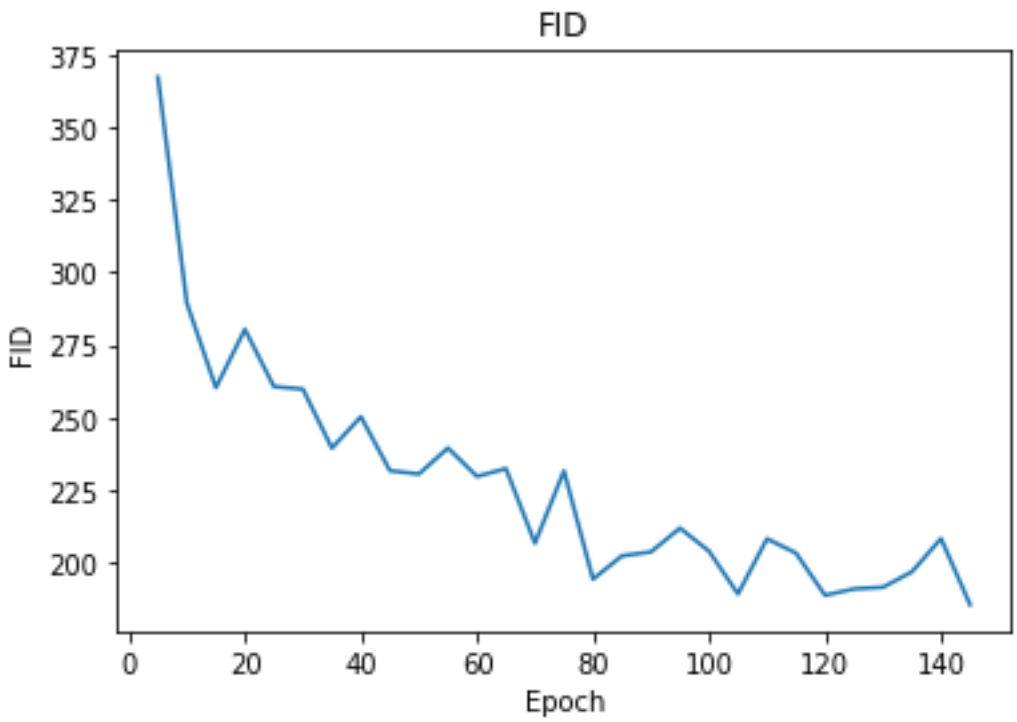
לגבי אימון הגנרטור – הפעם נתייג את $\tilde{x}$ ואת $\hat{x}$ כ *true* ונבדוק כמה טוב הם מרמים את D_x ובנוסף נוסיף שגיאת $\left\|\tilde{x} - x\right\|_1$ *L1* כי בארכיטקטורה מושלמת נרצה ש $\tilde{x} = G(E(x)) = x$

לגבי אימון המקודד – הפעם נתייג את $\tilde{z}$ ואת $\hat{z}$ כ *true* ונבדוק כמה טוב הם מרמים את D_z ובנוסף נוסיף שגיאת $\left\|\tilde{z} - z\right\|_2$ *L2* כי בארכיטקטורה מושלמת נרצה ש $E(G(Z)) = Z$

נציג את התוצאות :



קיבלנו שיפור מהארכיטקטורה הקודמת *dcgan* – התמונות נראות יותר חדות והצורה טיפה יותר ברורה.



גם במדד ה FID אכן רואים שיפור והגענו לערך מינימלי של 185 עבור ארכיטקטורה זו כאשר בקודם המינימום היה 210 וזה אחרי 150 epochs.

לגבי ה IS אותו דבר קיבלנו כמו מקודם וכמו שאמרנו להערכתנו מדד זה לא כל כך מועיל לנו.

לסיכום, בפרויקט זה בנינו Gan לייצור פוקימונים התחלנו עם ארכיטקטורה מוכרת -DCGAN לאחר מכן הוספנו שיטות שונות לשיפור Gan ועבדנו עם dataset יותר גדול. לבסוף ניסינו את הארכיטקטורה האחרונה שדיברנו עליה Aegan וכן שמנו לב לשיפור לאורך התקדמות הפרויקט כפי שפירטנו בהרחבה.
בארכיטקטורות השונות שעבדנו איתם לא התבצע חיפוש של היפר פרמטרים שונים מהסיבה שגם ככה הזמן ריצה היה כל כך ארוך (סדר גודל של שעה ל epoch) ולא היה מספיק כוח חישובי על מנת לבחון פרמטרים שונים לכל רשת.
בנוסף ב Gan קשה לתת מדד מדויק לשאלה האם הפרמטרים שבחרנו טובים כי פה המטרה היא לא התכנסות ומינימום פונקציית שגיאה או מקסימום דיוק. כן היה אפשר לנסות להשתמש במדד ה FID ולבחור פרמטרים שמביאים למינימום מדד זה אך בהינתן חוסר כוח חישובי זה לצערינו לא היה אפשרי (אם לאמן רשת ל 150 epochs לקח לנו 3 ימים אז עבור כל קומבינציה של היפר פרמטרים הזמן יוכפל) לכן העדפנו לנסות ארכיטקטורות שונות ולא פרמטרים שונים לאותה ארכיטקטורה כדי להנות יותר מהעבודה (מה הסיכוי שהיינו בוחרים קומבינציה מושלמת של פרמטרים שמשפרת פלאים את התוצאות?! ,ככה לפחות גם למדנו משהו מהתהליך). עוד נוסיף כי יכול להיות שאם נמשיך להריץ מעבר ל 150

אולי נקבל תוצאות יותר טובות אבל שוב עקב מגבלת כוח חישוב epochs
ומוגבלות המשתמש החינמי בקולאב הקשו עלינו עד מאוד

References:

https://towardsdatascience.com/autoencoding-generative-adversarial-networks-16082512b583

https://blog.jovian.ai/pokegan-generating-fake-pokemon-with-a-generative-adversarial-network-f540db81548d

https://machinelearningmastery.com/how-to-implement-the-inception-/score-from-scratch-for-evaluating-generated-images

https://scholar.smu.edu/cgi/viewcontent.cgi?article=1193&context=datasciencereview

https://becominghuman.ai/generating-new-pokemons-using-gans-ceba1c6dc676