

This Fish Does Not Exist

Final Project
Advanced Deep Learning

מרכה

ד"ר משה בוטמן

מגישיים

תומר יעקב 316593722
דניאל אידליס 205907298
ספיר מוסאצ'ו 318654001
עד קנפו 207040783

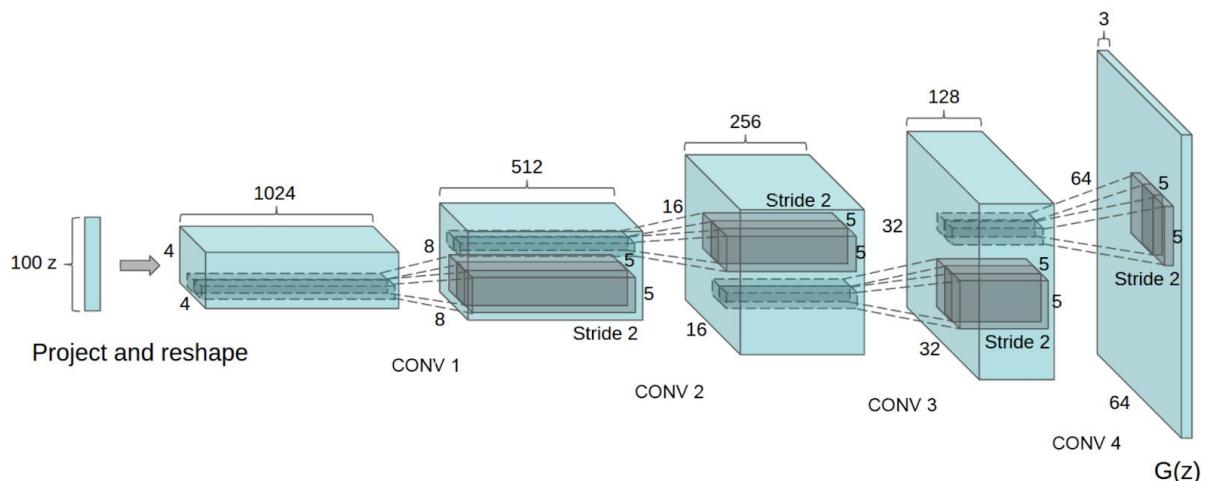
ארQUITטורה ראשונה- (DCGAN)

הקדמה:

בארכיטקטורה זו ניסינו למשה להבין איך לשפר את ביצועי ה GAN "הקלואס" בלי פונקציות Loss מתחכחות יתר על המידה. העיקרון המנחה היה להיצמד לקונבולוציות עם כמה עקרונות מפתח:

- הימנעות מ Pooling ושימוש ב Stride במקום
- שימוש ב Upsampling Transposed Convolutions לצורך
- הימנעות משכבות FC
- שימוש ב Batch Normalization

דוגמא ל Generator מבוסס קונבולוציה:



עיקר הניסויים היה סיבב קונפיגורציות שונות של שכבות הקונבולוציה הן בראשת ה Generator והן בראשת ה Discriminator.

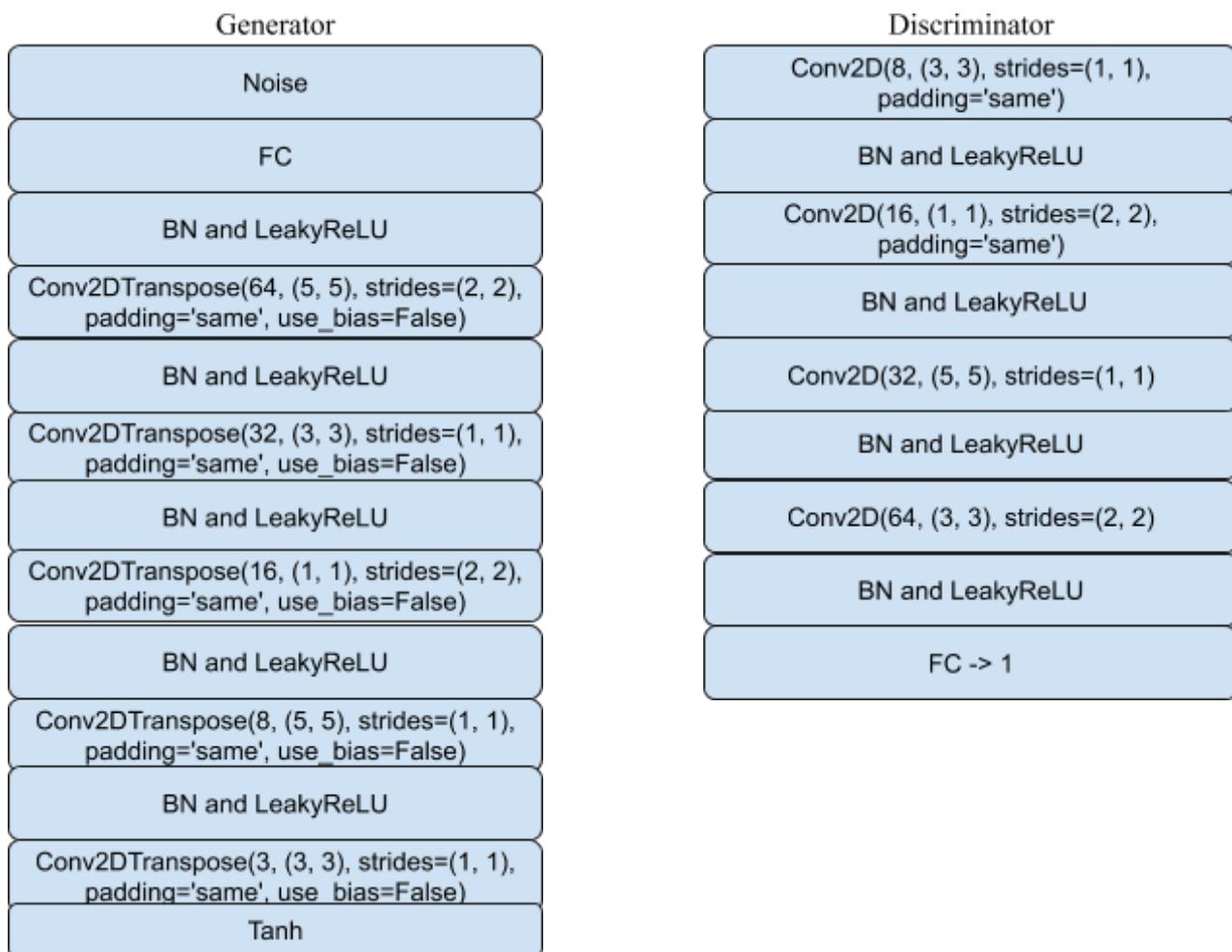
בנוסף, ניסינו לשחק עם תוספות ושיפורים מתוך המאמרים שקראננו, בין היתר הוספה של רעש ל Feature Label Smoothing, שימוש Dropout, Generator Matching.

ניסוי ראשון: קונבולוציה רדודה

<https://www.kaggle.com/dannyidlis/shallowgan>

בניסוי זה ניסינו להשתמש בקונבולוציה "חלשה" יחסית: מספר קטן של שכבות קונבולוציה בעומק של עד 32 שכבות. מטרת ניסוי זה היא לראות האם ניתן להגיע לתוצאות טובות גם ללא רשת عمוקה.

תיאור הרשת



שתי הרשתות השתמשו ב- ADAM עם הפרמטרים הבאים:

Generator: learning_rate=1e-4, beta_1=0.5, beta_2=0.999, epsilon=1e-07

Discriminator: learning_rate=2e-4, beta_1=0.5, beta_2=0.999, epsilon=1e-07

אוף אימון וביצועים

מספר Epochs היה 405

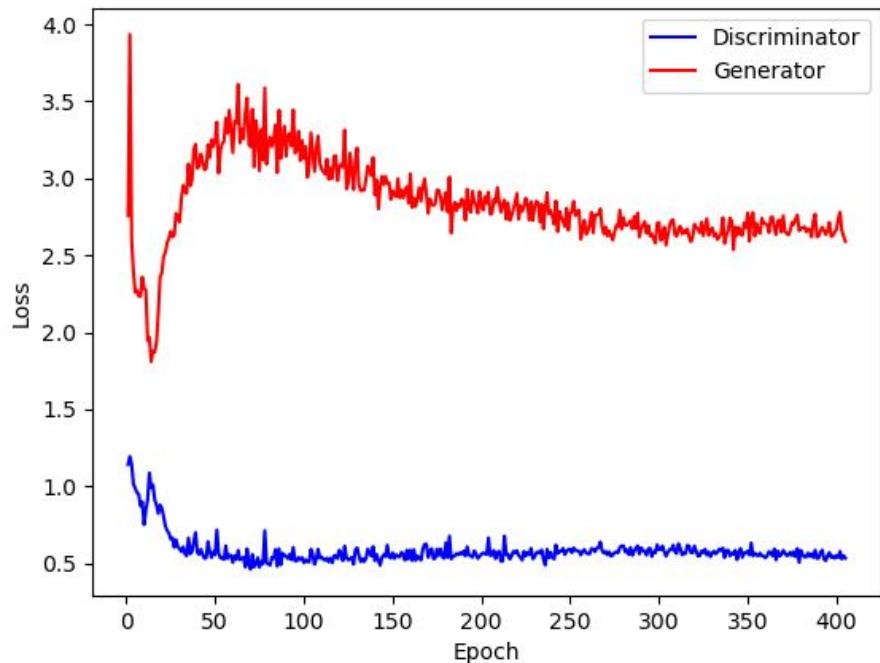
זמן האימון הממוצע על Epoch היה כ-30 שניות.

פונקציית loss Loss בזמן האימון הייתה BCE הסטנדרטית, עם Label Smoothing שマגריל שמאית אמת כמספר רנדומלי בין 0.9 ל 1 ומגריל Label זיופ כמספר רנדומלי בין 0 ל 0.1 כפי שניתן לראות בתמונה:

```
def discriminator_loss(real_output, fake_output):
    real_loss = cross_entropy(tf.random.uniform(real_output.shape, 0.9, 1), real_output)
    fake_loss = cross_entropy(tf.random.uniform(fake_output.shape, 0, 0.1), fake_output)
    total_loss = (real_loss + fake_loss) / 2
    return total_loss

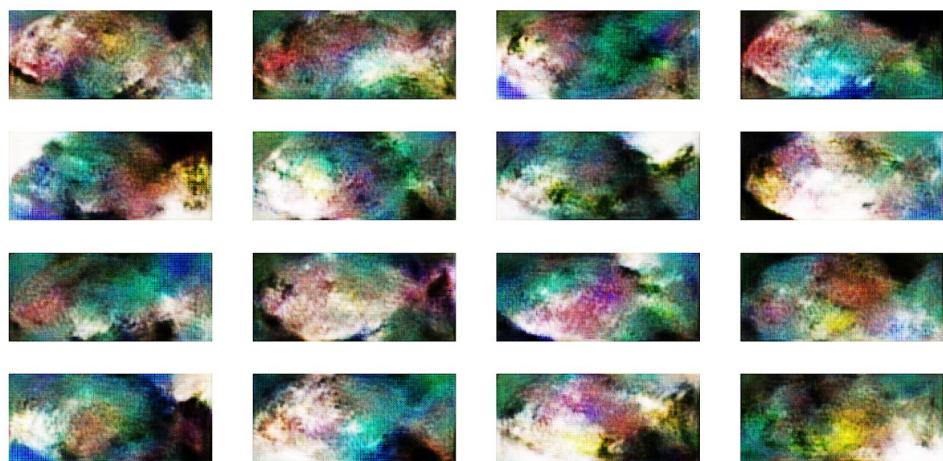
def generator_loss(fake_output):
    return cross_entropy(tf.random.uniform(fake_output.shape, 0.9, 1), fake_output)
```

להלן תרשים המתאר את loss Loss לאורך האימון:

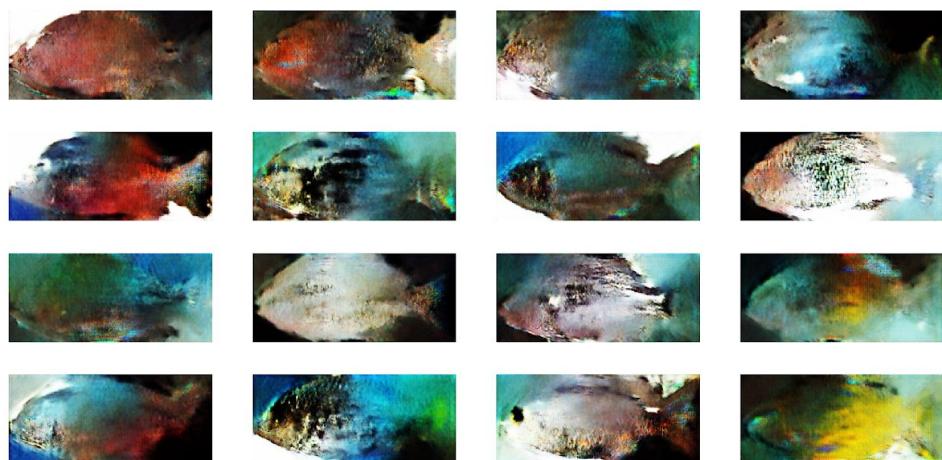


דוגמאות

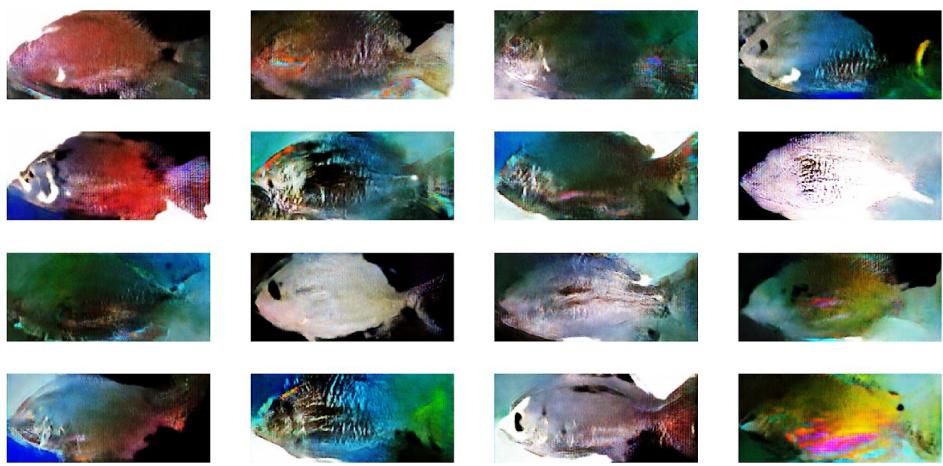
Epoch 10



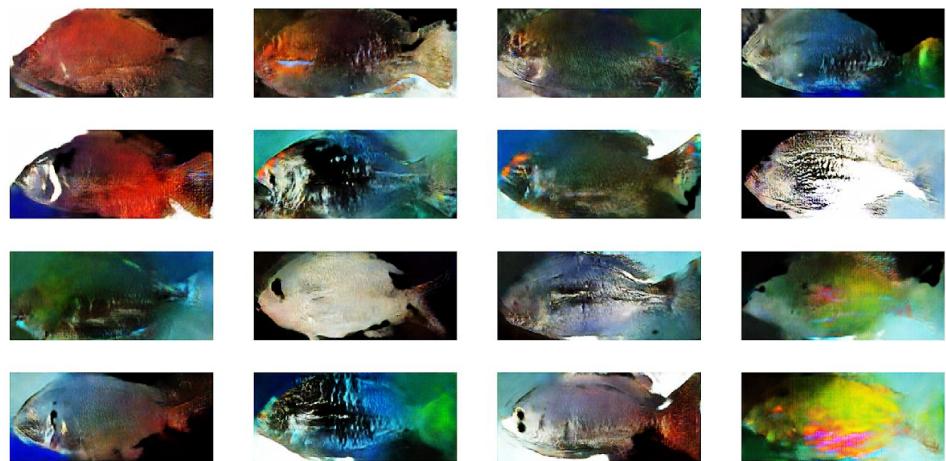
Epoch 150



Epoch 300



Epoch 405

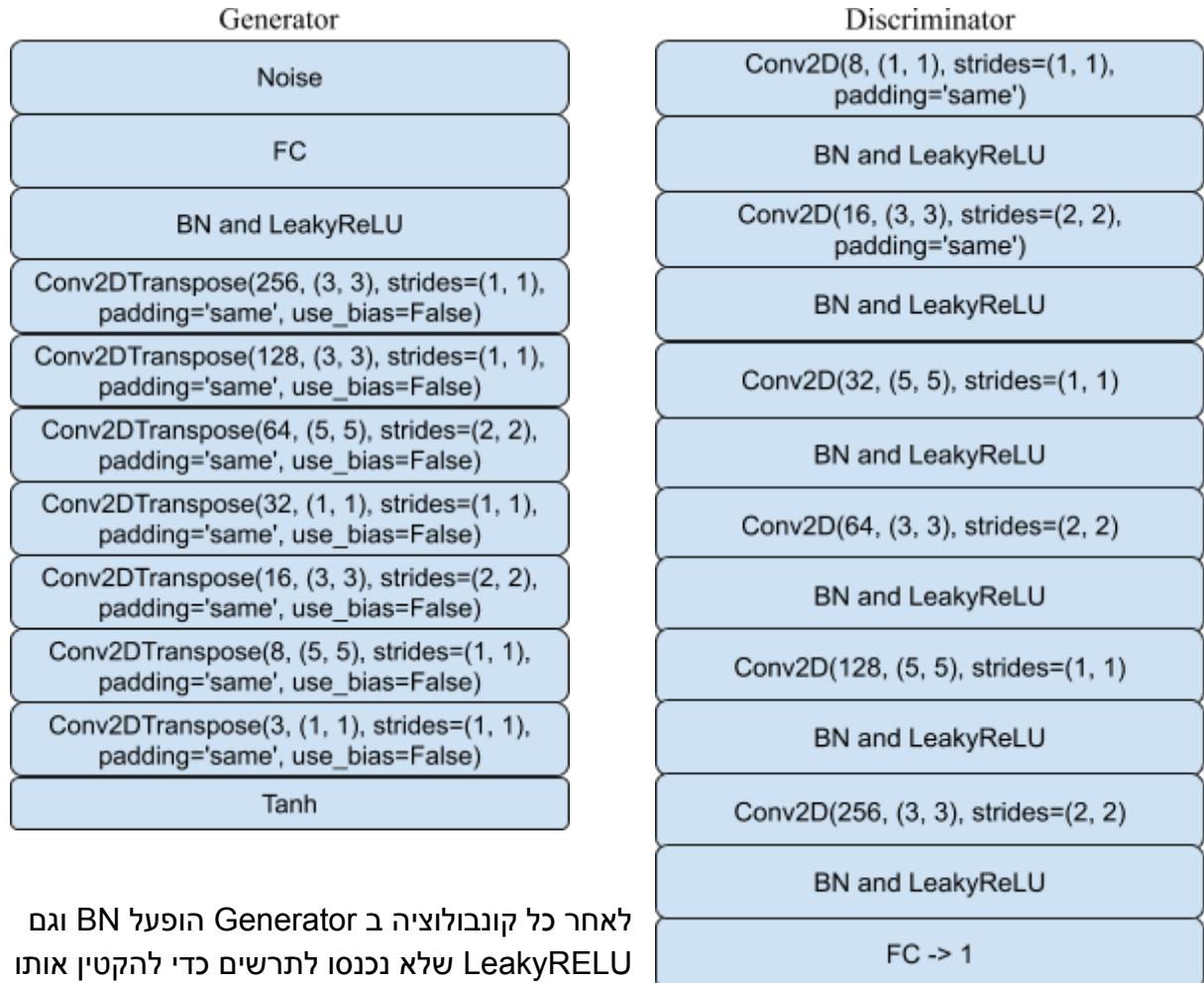


בסיסי שני: קונבולוציה عمוקה

<https://www.kaggle.com/dannyidlis/dcgan>

בניסוי זה חיזקנו את הרשות במספר גדול יותר של קונבולוציות וניסינו להגיע לעומק רב יותר בשכבות הביניים. המטרה הייתה לראות האם הוספת שכבות והעמקת הקונבולוציות תביא לתוצאה סופית טוביה יותר.

תיאור הרשות



שתי הרשותות השתמשו ב- ADAM עם הפרמטרים הבאים:

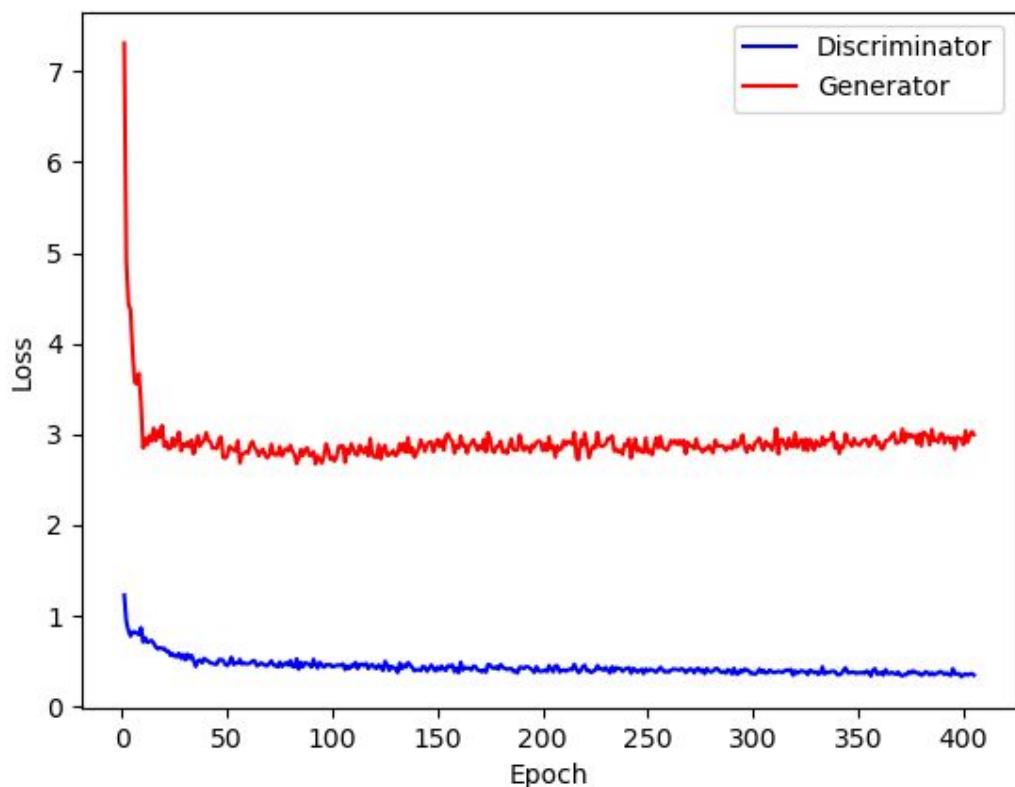
Generator: learning_rate=1e-4, beta_1=0.5, beta_2=0.999, epsilon=1e-07

Discriminator: learning_rate=2e-4, beta_1=0.5, beta_2=0.999, epsilon=1e-07

אוף אימון וביצועים

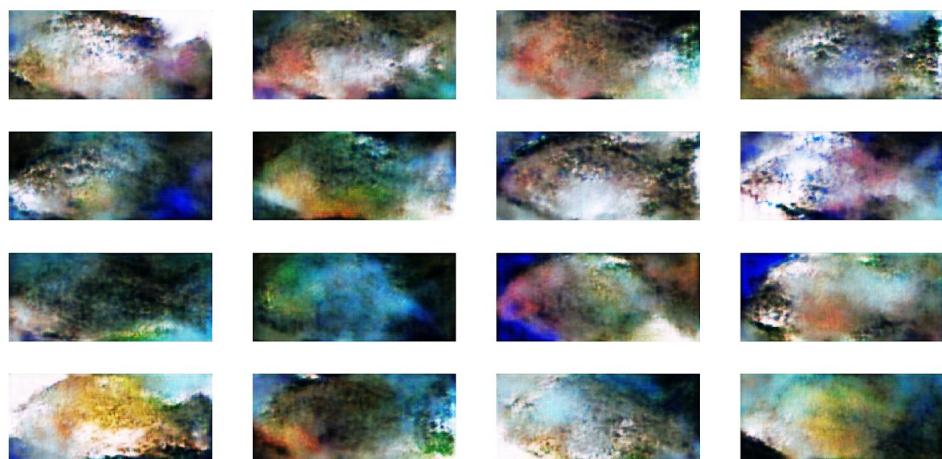
זהה לניסוי הראשון

להלן תרשימים המתאר את ה- Loss לאורך האמון.

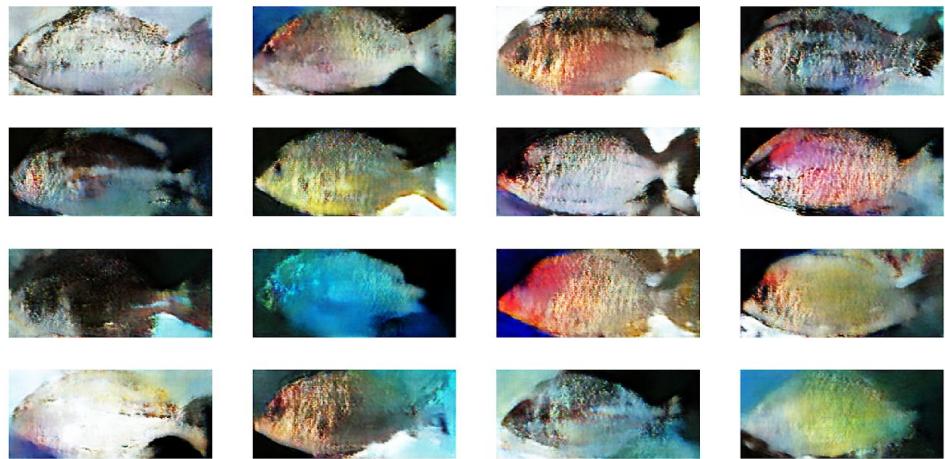


Epoch 10

דוגמאות



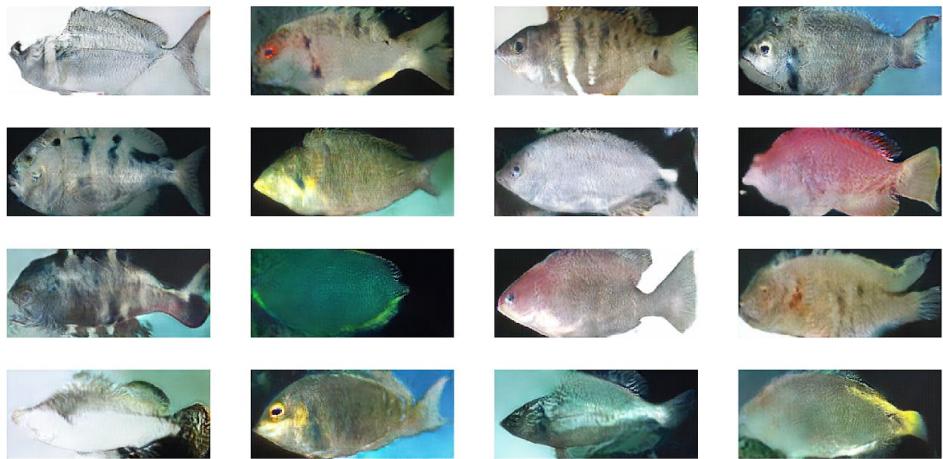
Epoch 50



Epoch 200



Epoch 300



Epoch 405

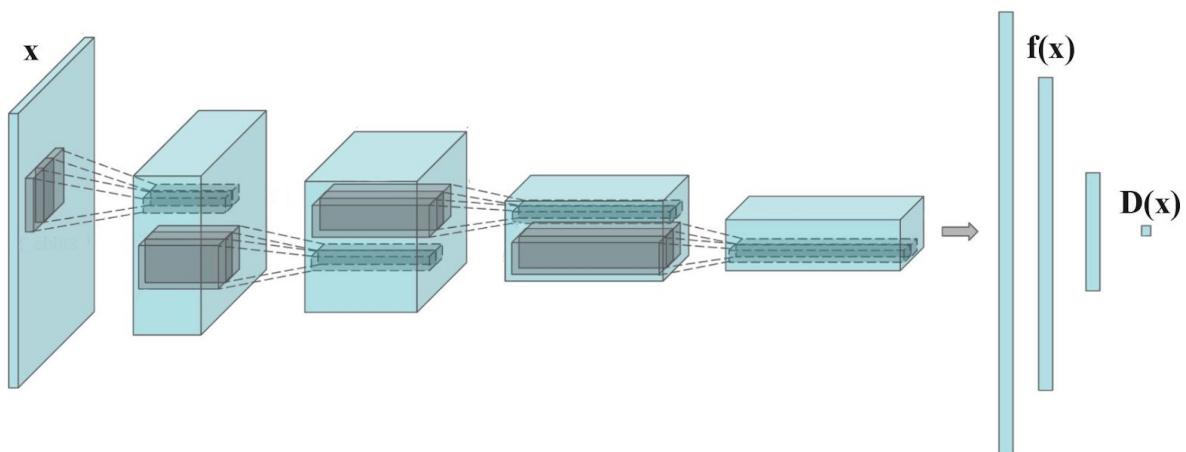


בסיסי שלישי: קונבולוציה عمוקה + Feature Matching
<https://www.kaggle.com/dannyidlis/feature-matching>

בניסוי זה ניסינו לשפר את הרשת מניסוי מ' 2 ע"י שימוש ב Feature Matching העיקרון של Feature Matching הוא להעניש את ה Generator במידה והוא הוציא תמונה שהפיצ'רים שחולצו ממנה ע"י ה "רחוקים" Discriminator מוד"י מהפיצ'רים שה Discriminator מחלץ מתוך תמונה אמיתית. מדובר בחישוב Distance L2 בין הגורמים הנ"ל:

$$\|\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \mathbf{f}(\mathbf{x}) - \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} \mathbf{f}(G(\mathbf{z}))\|_2^2$$

פרקטיית, علينا ללקח את שכבת ה Features (שכבה Flatten Discriminator שארה הkonvolוציות) המתוארת בתרשים ולהשוות את ה Output שלה על תונות אמיתיות ל Output שלה על תונות מזויפות:



תוצרת הרשת
זהה לניסוי השני

אופן אימון וביצועים

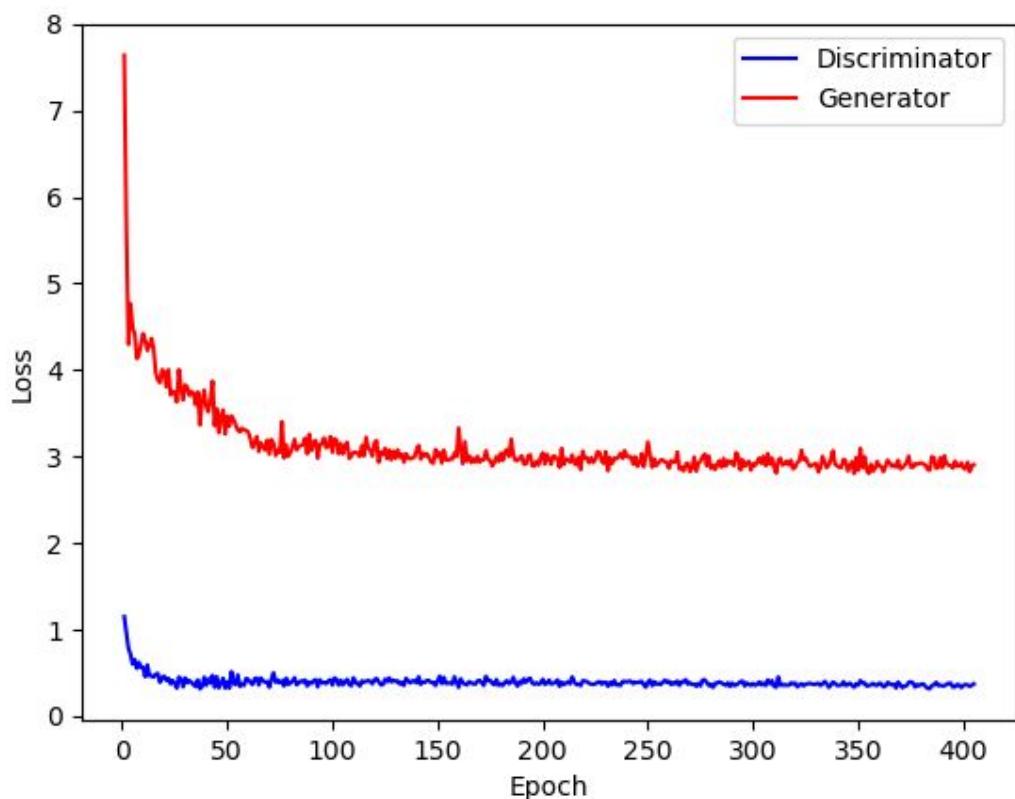
זהה לניסוי הראשון, כמעט ללא התוספת של Feature Matching :

```
def feature_matching(fake_fx, real_fx):
    return tf.sqrt(tf.square(tf.reduce_mean(fake_fx) - tf.reduce_mean(real_fx)))
```

והוסף התוצאה ל Loss של ה Generator :

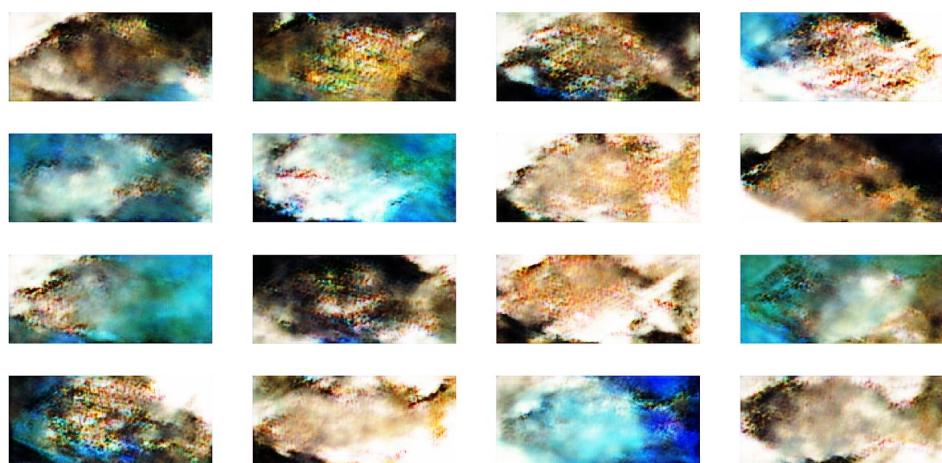
```
gen_loss = generator_loss(fake_output) + feature_matching(fake_fx, real_fx)
```

להלן תרשימים המתאר את ה- Loss לאורך האמון.

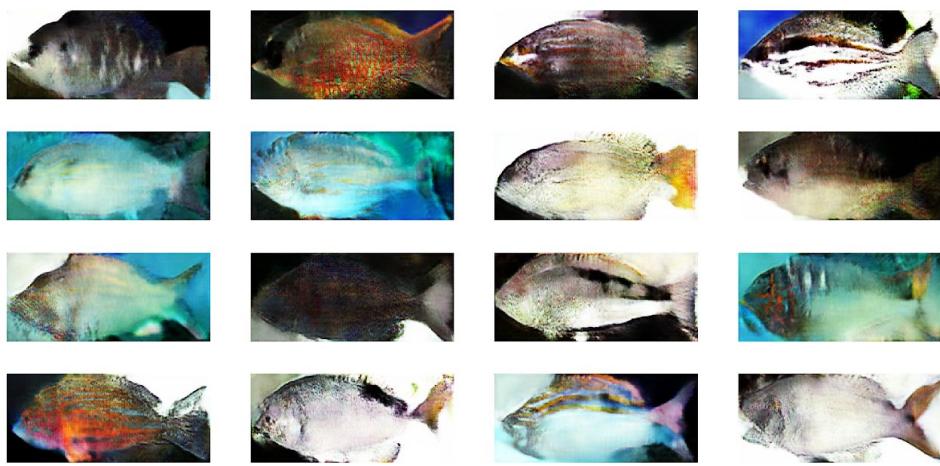


דוגמאות

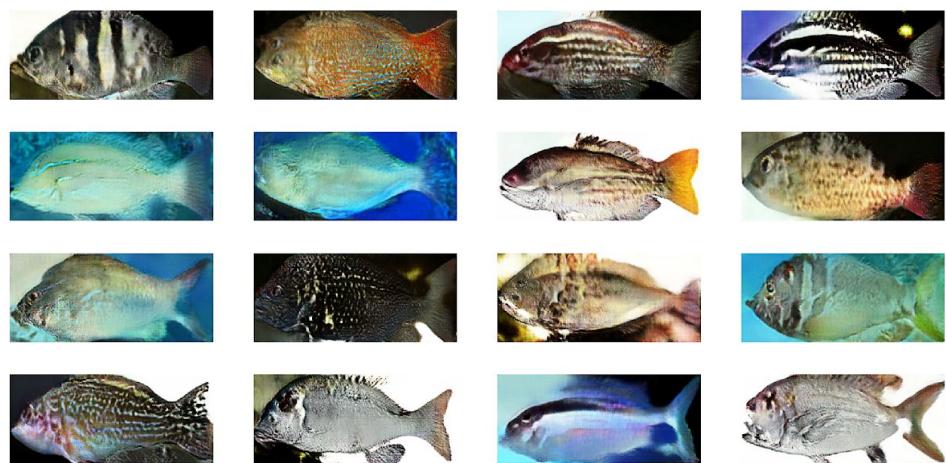
Epoch 30



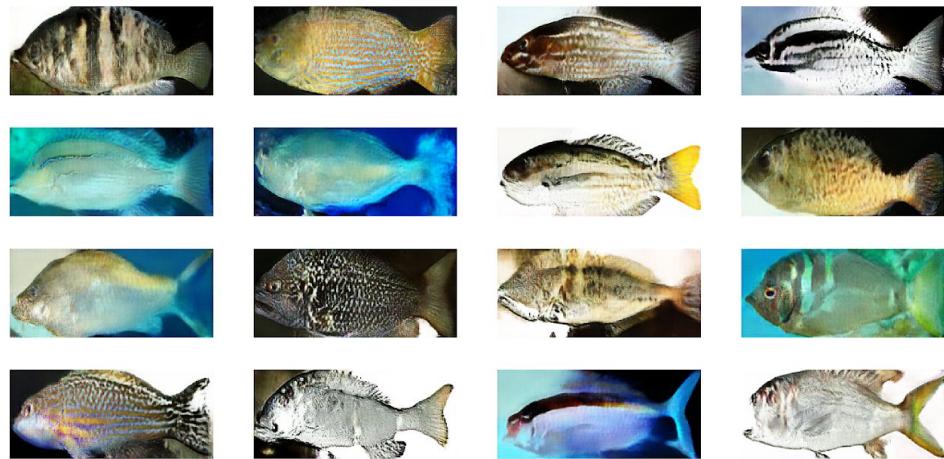
Epoch 120



Epoch 300



Epoch 405



מקורות

https://medium.com/@jonathan_hui/gan-a-comprehensive-review-into-the-gangsters-of-gans-part-2-73233a670d19

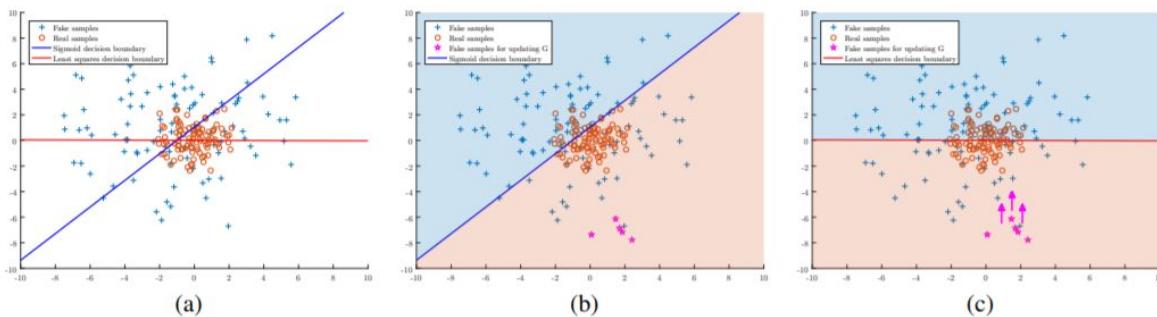
<https://towardsdatascience.com/gan-ways-to-improve-gan-performance-acf37f9f59b>

ארכיטקטורה שנייה - (LSGAN)

LSGAN היא ארכיטקטורת רשת שבאה לייצר תמונות באיכות גבוהה עם תשומת לב רבה לפרטים כדי שההתמונות יהיו נאמנות למציאות כלל הניתן.

בארכיטקטורה זו עושים שימוש ב Loss Mean Square Error כפונקציית Loss של ה Discriminator . הרעיון העיקרי עליו מבוססת הארכיטקטורה הוא שעל ידי שימוש ב MSE כפונקציית Loss ניתן להעניק על המרחק של הדוגמאות המזויפות מקום קבלת ההחלטה.

כלומר, במידה ועל דוגמה מוגנרטת ה Discriminator אמראמת, פונקציית ה Loss תעניש אותו על המרחק מקום קבלת ההחלטה וכן ניתן יהיה ללמד את המודל לייצר תמונות דומות למציאות.



.Sigmoid Cross Entropy & MSE: בתמונה מעלה ניתן לראות את ההתנהגות של שתי פונקציות: בתרשים (a) רואים את קו קבלת ההחלטה של שתי הפונקציות.

(b) קו קבלת ההחלטה של sigmoid cross entropy , האזור הכתום הוא האזור של התמונה המציאותיות והאזור הכהול הוא האזור של התמונה המזויפות. במקרה זה, דוגמאות מציאות חדשות (כמו אלו המסומנות בוורוד), מקבלות ערך שגיאה מאוד נמוך בגל שנן לצד של התמונה

המציאותיות ולכן העדכון של ה generator יהיה קטן מאוד.

(c) קו קבלת ההחלטה של MSE . MSE מענישה את הדוגמאות המזויפות לפי המרחק מהקו וכך היא מכrichtה את generator לייצר דוגמאות קרובות יותר לקו ומצביעות יותר.

1. ניסוי ראשון

<https://www.kaggle.com/adiknafo/lsgan-adi?scriptVersionId=30067993>

* ניתן לראות את תוצאות הניסוי ב commit 5

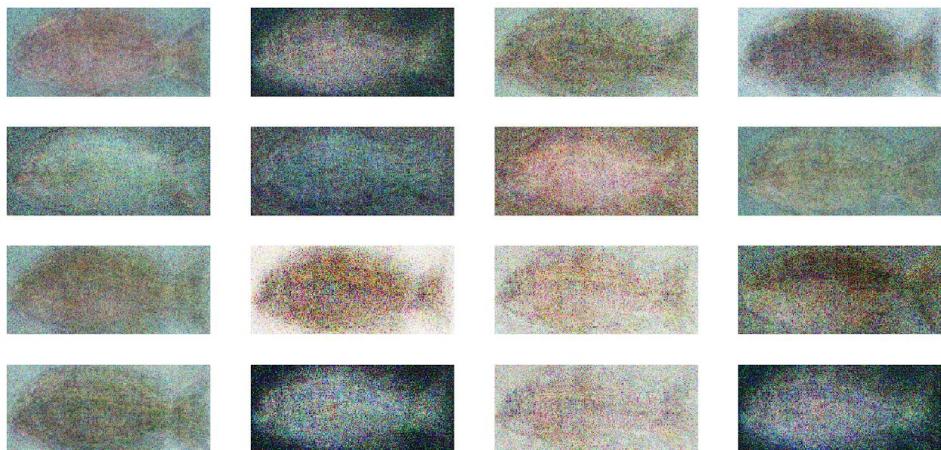
,Convolutional, השתמשנו בארכיטקטורה ללא שכבות pooling, רק שכבות Fully connected .

תיאור הארכיטקטורה: .1.1

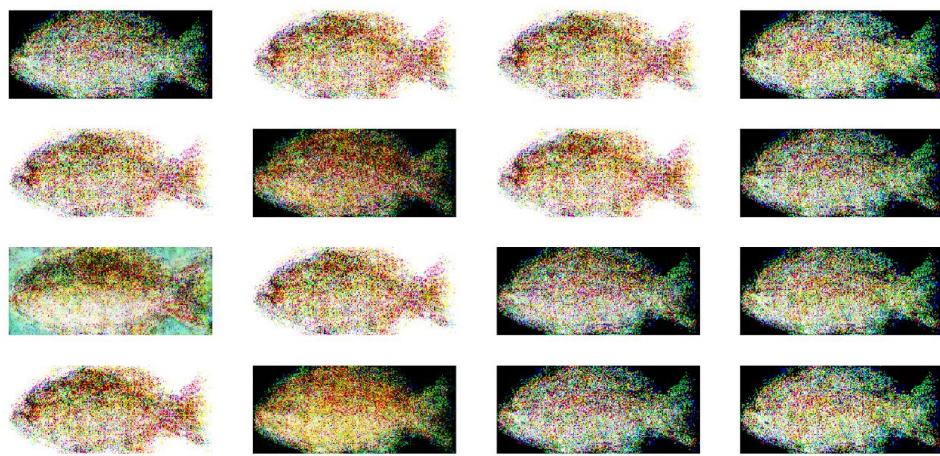


1. תוצאות

Epoch 10



Epoch 400



ה Loss של ה discriminator הוכנס סבב 0.04
ה Loss של ה generator הוכנס סבב 1.06

מסקנות . 1.2

ניתן לראות שדי מהר ה Generator מבין מה הם קווים המתאר של דג, כבר באפקן 10 אפשר לראות מסגרת של דגים. אחרי 400 אפקנים ממש ניתן לראות בבירור צורה של דגים, אף הדגים מאד מפוקסלים لكن החלטנו להוסיף רשותת קונבולוציה כפתרון.

ניסוי שני - . 2

<https://www.kaggle.com/yardenm/lsgan-adi/edit/run/32247058>

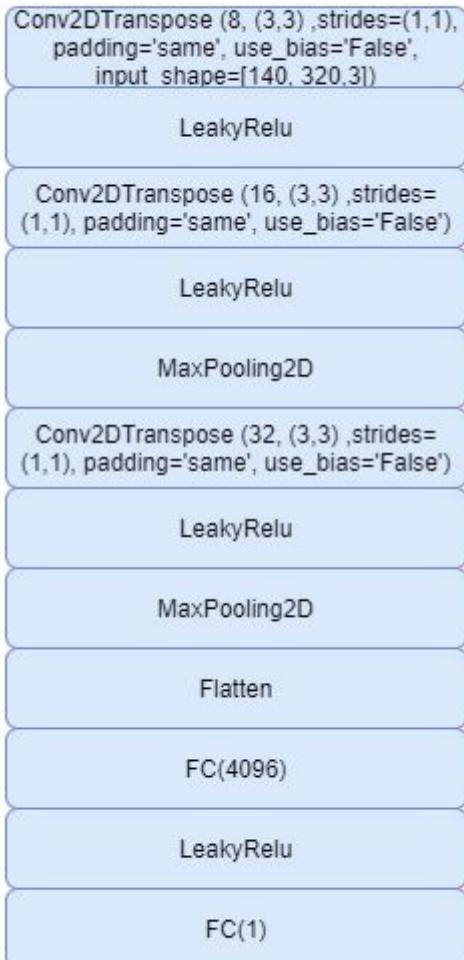
בניסוי השני השתמשנו בארכיטקטורה המכילה קונבולוציות. במהלך העבודה על הארכיטקטורה נעשה המון ניסיונות שונים, להגדיל את כמות הפילטרים, לשנות את גודל הקרNEL, שימוש ב Kernel Initializer, הוספה שכבות קונבולוציה. אחרי הרבה ניסיונות נציג כאן את ארכיטקטורת LSGAN שיצרה את התוצאות הטובות ביותר.

תיאור הארכיטקטורה .2.1

Generator



Discriminator



שתי הרשותות השתמשו ב- ADAM עם הפרמטרים הבאים:
`adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-03)`

אופן האימון .2.2

מספר ה-Epochs היה 400

זמן האימון הממוצע על Epoch היה כ-30 שניות.

פונקציית loss-Loss בזמן האימון MSE

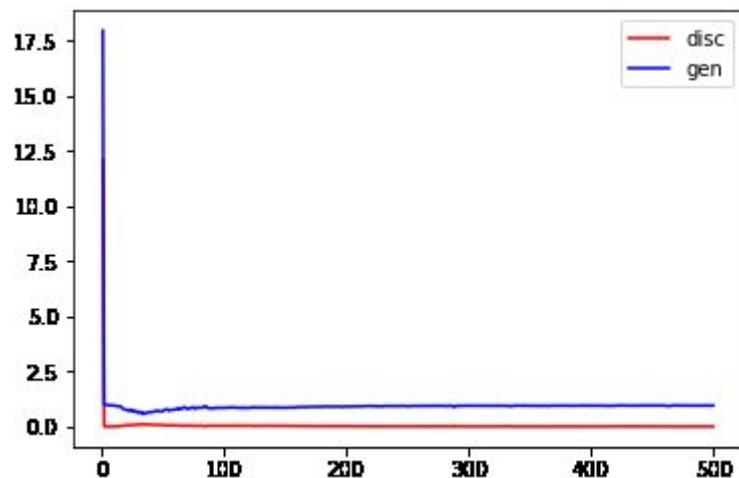
```

mse = tf.keras.losses.MeanSquaredError()

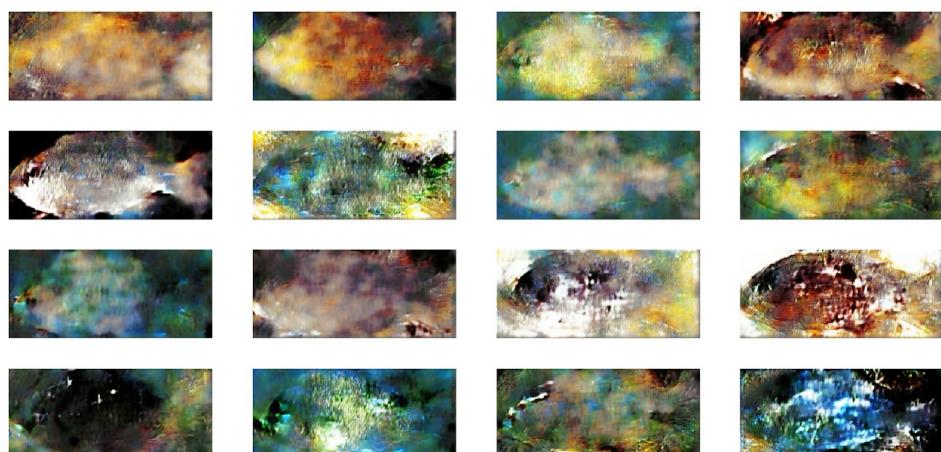
def discriminator_loss(real_output, fake_output):
    real_loss = mse(real_output, tf.ones_like(real_output) * 0.9)
    fake_loss = mse(fake_output, tf.zeros_like(fake_output))
    total_loss = 0.5 * (real_loss + fake_loss)
    return total_loss

def generator_loss(fake_output):
    return mse(fake_output, tf.ones_like(fake_output))
  
```

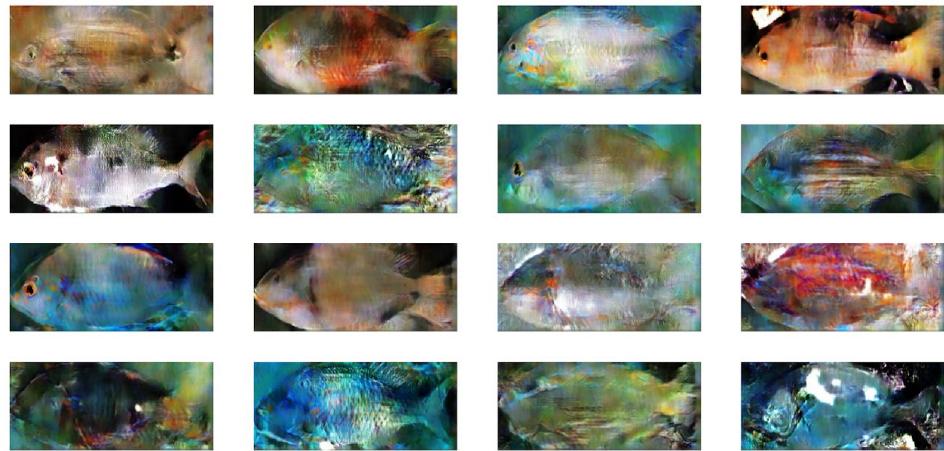
תוצאות .2.3



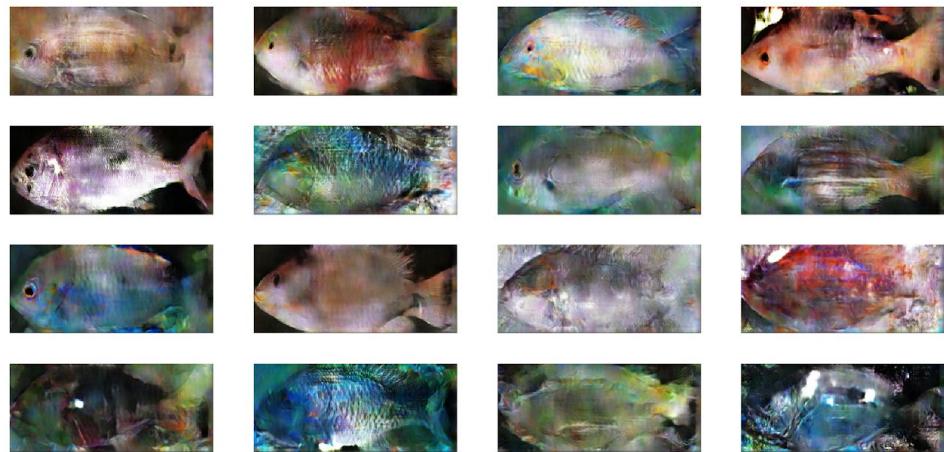
Epoch 100



Epoch 300



Epoch 400



מקורות :

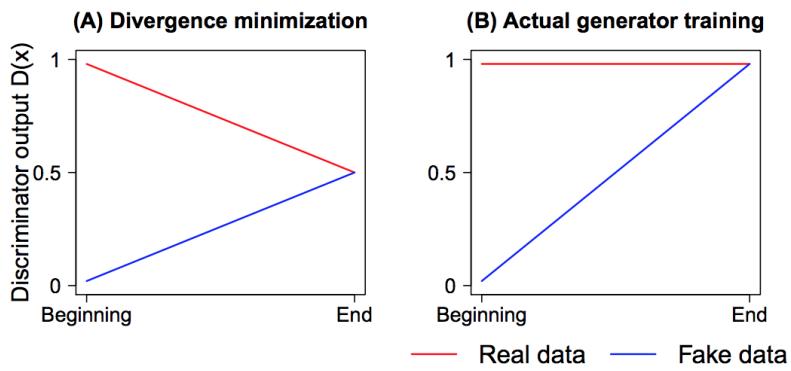
1. https://medium.com/@jonathan_hui/gan-lsgan-how-to-be-a-good-helper-62ff52dd3578
2. http://openaccess.thecvf.com/content_ICCV_2017/papers/Mao_Least_Squares_Generative_ICCV_2017_paper.pdf

ארכיטקטורה שלישית - RaGAN

<https://www.kaggle.com/tomeryacov/ralsgan-with-adam>

הבדיל מארכיטקטורות אחרות של GAN, אשר בהן ה-Discriminator מנסה להזות באופן הסתברותי האם תמונה היא אמיתית או לא. RaGAN / RGAN מנסה לדעת מה המרחק היחסי בין תמונה אמיתית לבין תמונה אשר נוצרה ע"י ה-Generator. או במלים אחרות, לדעת כמה תמונה היא אמיתית אל מול אחת מזויפת ולהפוך.

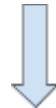
במקום שנשאף שה-Discriminator יתן את ההסתברות 1 לתמונה מזויפת, נרצה שהוא יתן את ההסתברות חצי בין תמונה אמיתית לאחת מזויפת. זאת אומרת שהבחירה שלו תהיה שווה לבחירה רנדומלית.



מכאן שפונקציית loss שלנו צריכה להשתנות בהתאם על מנת לייצג את המרחק.

RGAN

$$\min_D V(D) = -\mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] - \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$



בצורה כללית

$$\begin{aligned} L_D^{RSGAN} &= -\mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [\log(\text{sigmoid}(C(x_r) - C(x_f)))]. \\ L_G^{RSGAN} &= -\mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [\log(\text{sigmoid}(C(x_f) - C(x_r)))]. \end{aligned} \quad \begin{array}{c} \equiv \\ \equiv \end{array} \quad \begin{aligned} L_D^{RGAN*} &= \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_r) - C(x_f))]. \\ L_G^{RGAN*} &= \mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q})} [f_1(C(x_f) - C(x_r))]. \end{aligned}$$

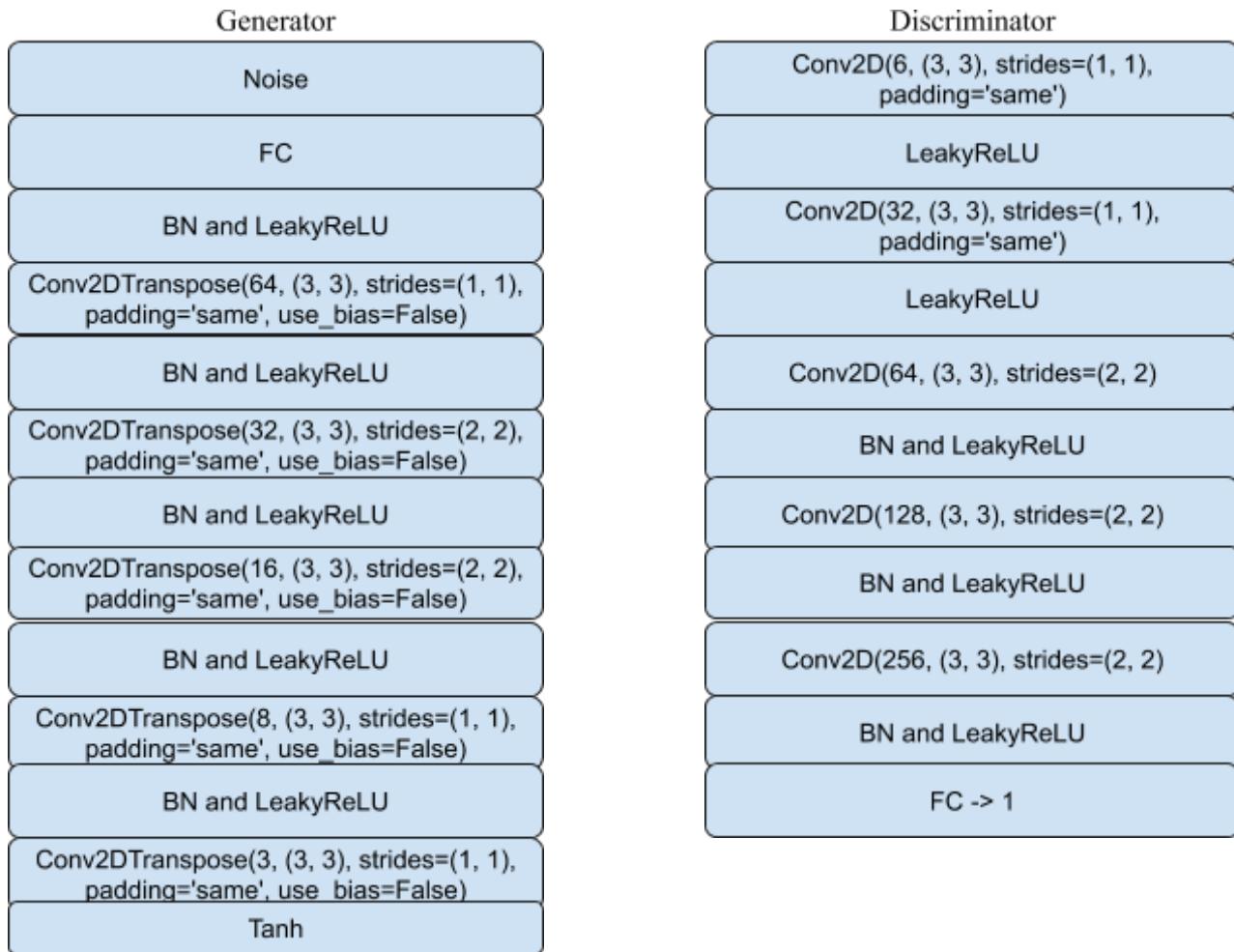
כאשר x_r מסמל את התמונות האמיתיות ו- x_f את המזויפות.

RaGAN

בצורה זאת אנו מודדים את המרחק היחסי אל מול ממוצע הצד השני.

$$\begin{aligned} L_D^{RaGAN} &= \mathbb{E}_{x_r \sim \mathbb{P}} [f_1(C(x_r) - \mathbb{E}_{x_f \sim \mathbb{Q}} C(x_f))] + \mathbb{E}_{x_f \sim \mathbb{Q}} [f_2(C(x_f) - \mathbb{E}_{x_r \sim \mathbb{P}} C(x_r))]. \\ L_G^{RaGAN} &= \mathbb{E}_{x_r \sim \mathbb{P}} [g_1(C(x_r) - \mathbb{E}_{x_f \sim \mathbb{Q}} C(x_f))] + \mathbb{E}_{x_f \sim \mathbb{Q}} [g_2(C(x_f) - \mathbb{E}_{x_r \sim \mathbb{P}} C(x_r))]. \end{aligned}$$

תצורת הרשת



שתי הרשתות השתמשו ב- ADAM עם הפרמטרים הבאים:
 learning_rate=0.0002, beta_1=0.5, beta_2=0.999, epsilon=1e-07

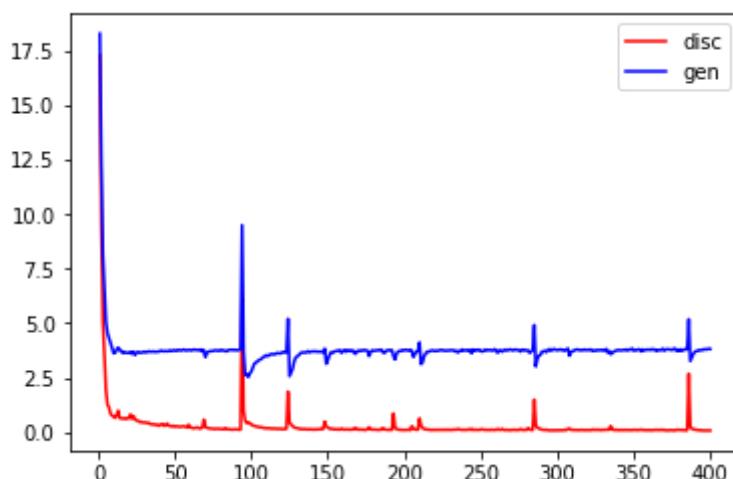
אוף אימון וביצועים
 מספר Epochs היה 400
 זמן האימון הממוצע על Epoch היה כ-40 שניות.
 פונקציית loss Loss בזמן האימון ליקחת את הקונספט של MSE, עם התאמת לארכיטקטורת RaGAN

RaLSGAN

$$L_D^{RaLSGAN} = \mathbb{E}_{x_r \sim \mathbb{P}} [(C(x_r) - \mathbb{E}_{x_f \sim \mathbb{Q}} C(x_f) - 1)^2] + \mathbb{E}_{x_f \sim \mathbb{Q}} [(C(x_f) - \mathbb{E}_{x_r \sim \mathbb{P}} C(x_r) + 1)^2]$$

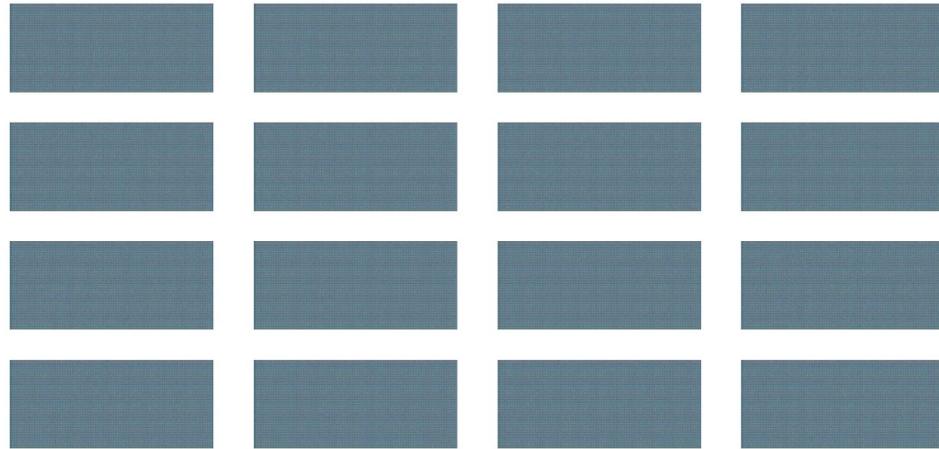
$$L_G^{RaLSGAN} = \mathbb{E}_{x_f \sim \mathbb{P}} [(C(x_f) - \mathbb{E}_{x_r \sim \mathbb{P}} C(x_r) - 1)^2] + \mathbb{E}_{x_r \sim \mathbb{P}} [(C(x_r) - \mathbb{E}_{x_f \sim \mathbb{Q}} C(x_f) + 1)^2]$$

להלן תרשים המתאר את ה- Loss לאורך האימון.

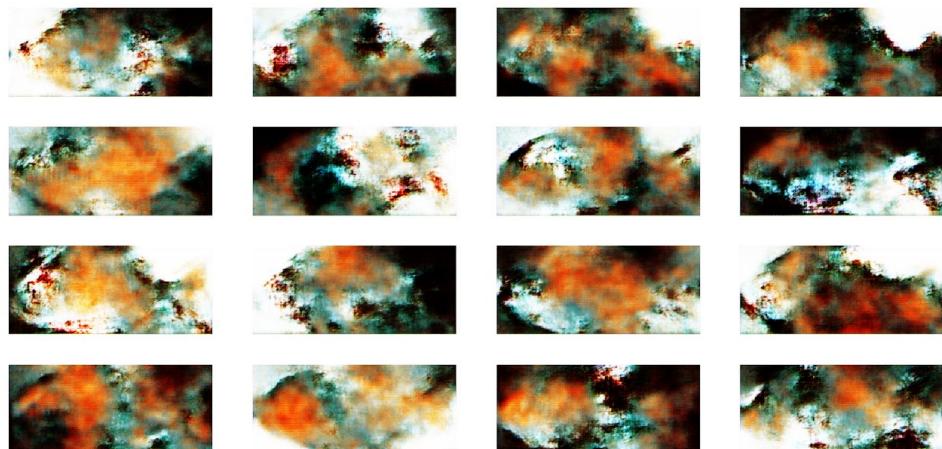


דוגמאות

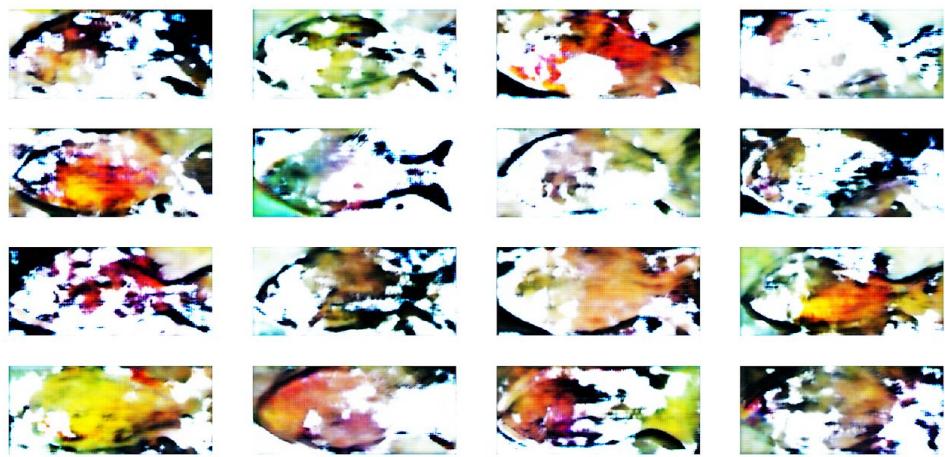
Epoch 1



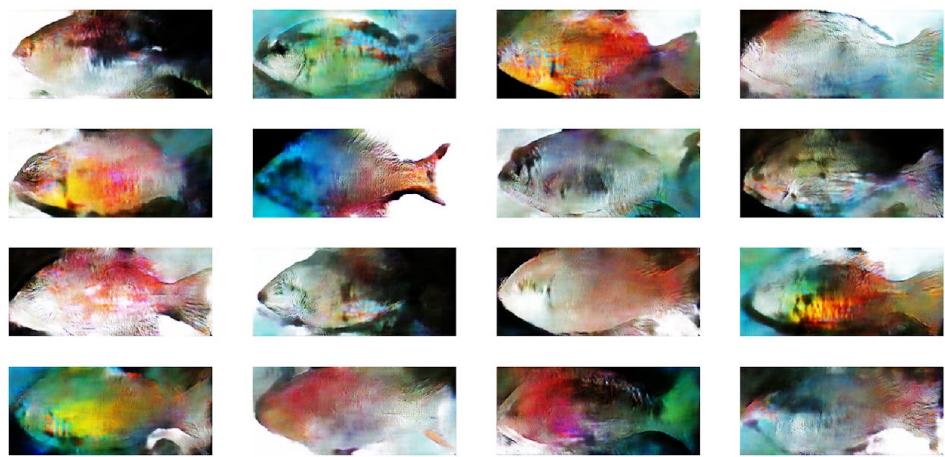
Epoch 10



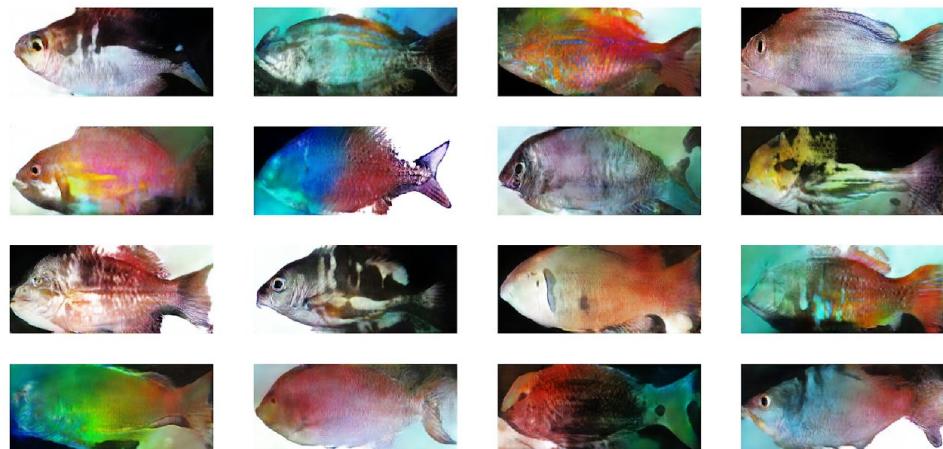
Epoch 94



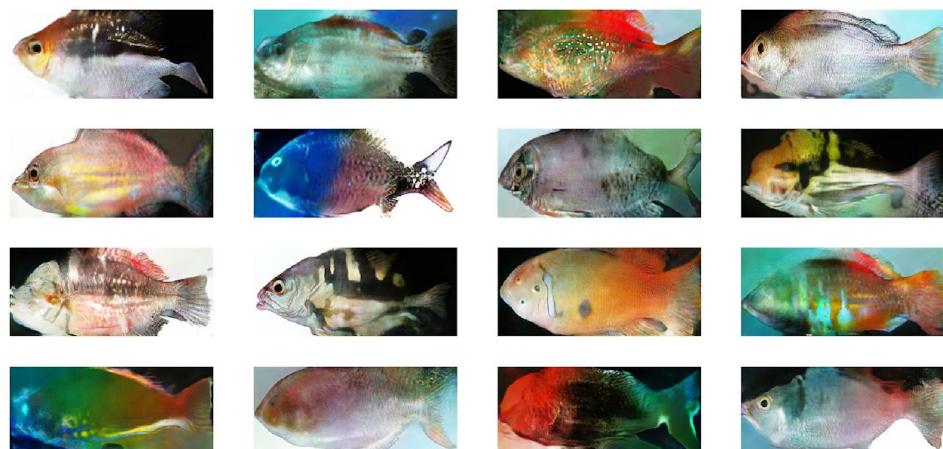
Epoch 100



Epoch 250



Epoch 400



מקורות

1. https://medium.com/@jonathan_hui/gan-rsgan-ragan-a-new-generation-of-cost-function-84c5374d3c6e
2. <https://arxiv.org/pdf/1807.00734.pdf>

висויים נוספים

RaGAN עם התאמת לאריטקטורת BinaryCrossentropy

אוף אימון וביצועים

שתי הרשתות השתמשו ב- SGD

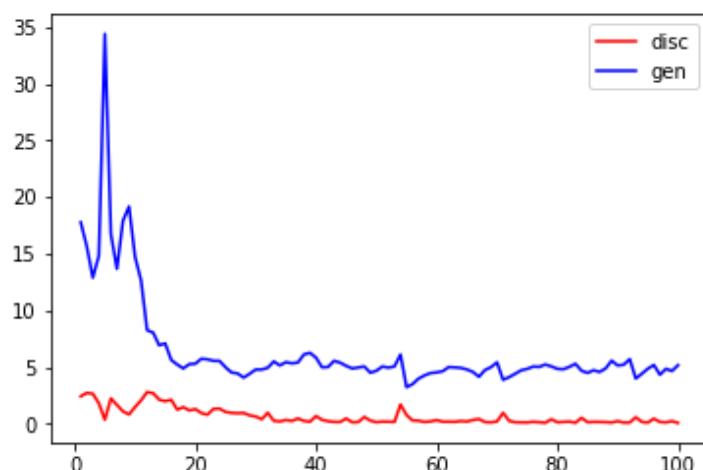
מספר ה-Epochs היה 100 זמן האימון הממוצע על Epoch היה כ-40 שניות.

פונקציית ה-Loss בזמן האימון, BinaryCrossentropy RaGAN,

$$L_D^{RaGAN} = \mathbb{E}_{x_r \sim \mathbb{P}} [f_1(C(x_r) - \mathbb{E}_{x_f \sim \mathbb{Q}} C(x_f))] + \mathbb{E}_{x_f \sim \mathbb{Q}} [f_2(C(x_f) - \mathbb{E}_{x_r \sim \mathbb{P}} C(x_r))].$$

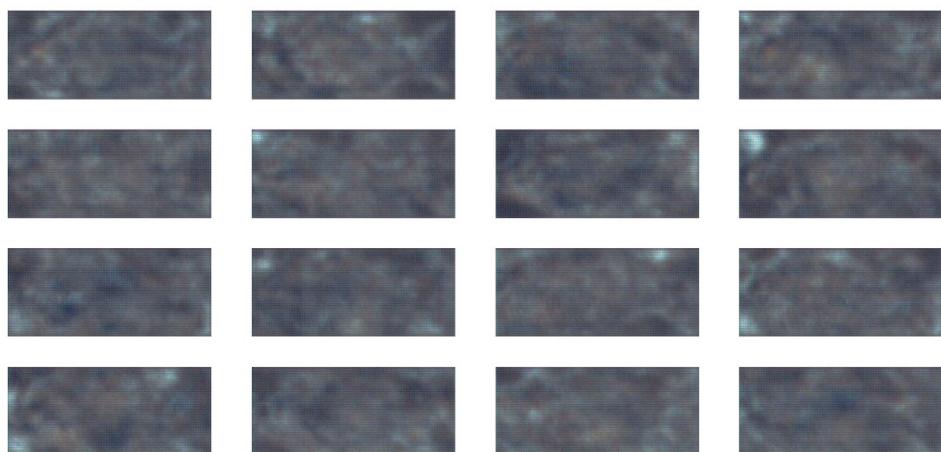
$$L_G^{RaGAN} = \mathbb{E}_{x_r \sim \mathbb{P}} [g_1(C(x_r) - \mathbb{E}_{x_f \sim \mathbb{Q}} C(x_f))] + \mathbb{E}_{x_f \sim \mathbb{Q}} [g_2(C(x_f) - \mathbb{E}_{x_r \sim \mathbb{P}} C(x_r))].$$

להלן תרשימים המתאר את ה-Loss לאורך האימון.

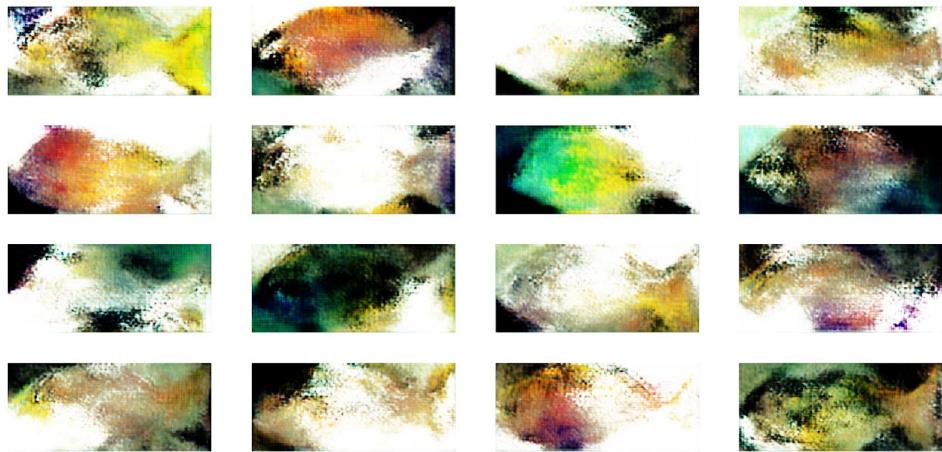


דוגמאות

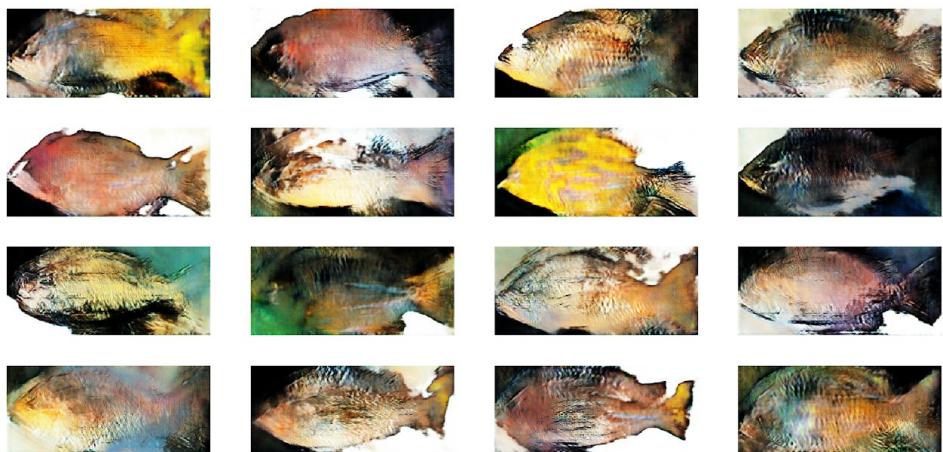
Epoch 1



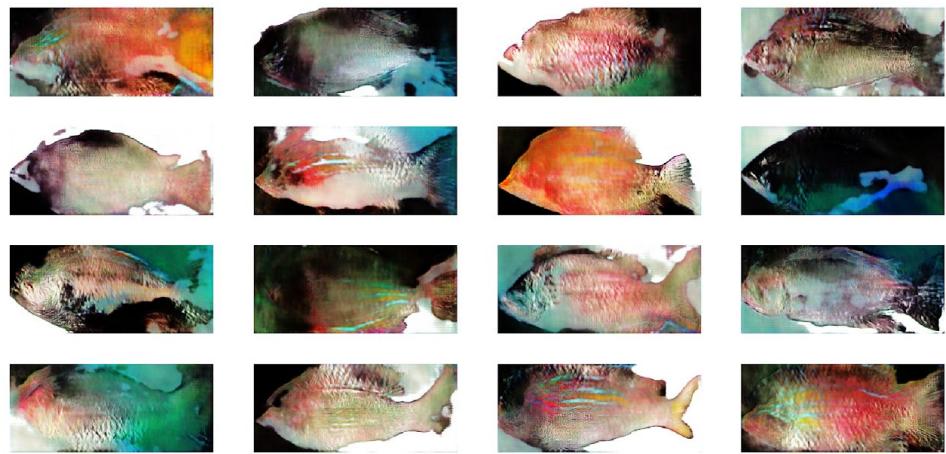
Epoch 20



Epoch 60



Epoch 100



גודל יותר LR

בחלק מהניסיונות נעשה שימוש במקדמי למידה שונים אשר נתנו תוצאות לא איקוטיות.
הארcitקטורה הינה אותה ארcitקטורה כמו שמצוור למטה.

learning_rate=0.002

לאחר epoch 50 היה נראה כי אין התכונות והניסוי הופסק



BEGAN: Boundary Equilibrium - Generative Adversarial Networks

מחברת הרצה: <https://www.kaggle.com/sapman97/began-fish>

הארQUITטורה BEGAN היא ARQUITטורה GAN המנסה להגדיר לרשות נקודת איזון, ולأمان עד הגעה אליה. הרשות נותנת כלים גם להגדיר את האיזון בין שונות התמונות המיוצרות לבין איכותן.

הרשות היא מעין הכלאה בין WGAN ל EBGAN, כאשר היא מכילה עקרונות משניהם. WGAN היא ARQUITטורה בה ה-zDiscriminator מסתנה ל-Critic, שמטרתו היא למצוא את מרחק Wasserstein בין התפלגות התמונות האמיתיות לבין התפלגות התמונות המזויפות. גם ב WGAN קיימת נקודת איזון מוגדרת לרשות, שבאה על חשבון זמן אימון גדול. EBGAN היא ARQUITטורה המשתמשת ב Autoencoder על מנת להבדיל בין תמונות מזויפות לאמיתיות.

Autoencoder היא ARQUITטורת רשות שמרתה, בדרך כלל, לייצר פיצ'רים טובים וアイcotים לבעה מסוימת. היא מורכבת משתי רשותות: encoder ו-decoder. ה-decoder מקבל כקלט את הקלט ומטרתו ממנו סט פיצ'רים בגודל קבוע, וה-decoder מקבל כקלט את הפיצ'רים שיצאו מה encoder ומטרתו לשחזר את תמונה הקלט. מטרת ה Autoencoder בכלל היא להצליח לשחזר את הקלט בפלט, ובעצם, אם ניתן לשחזר מהפיצ'רים שהרשות מצאה את הקלט, אז נראה הפיצ'רים איקוטיים. ניתן לחשב על זה כמעין CIוז איקוטי.

נרצה להגיע לעיבור reconstruction נמוך עבור תמונות אמיתיות, וגובהו עבור תמונות מזויפות. ה reconstruction מוגדר על ידי מרחק L1 או L2:

$$D(x) = \|Dec(Enc(x)) - x\|$$

חשוב מאד לבחור נכון את כמות הפיצ'רים שיוציאו ה-Autoencoder. מצד אחד, אם המספר יהיה קטן מדי, למשל פיצ'ר אחד עבור תמונה בגודל 140X320, ה-Autoencoder לא יוכל למצוא פיצ'רים איקוטיים מספיק וכנראה יהיה לו loss reconstruction גבוה. אך, מצד שני, גם מספר גובה מדי יוביל לתוצאות לא טובות. למשל, אם ניקח עבור תמונה באותו גודל, 35X80 פיצ'רים, נראה שהוא חזק דיו כדי להתקרב פונקציית הזהות. במקרה הזה, ה loss reconstruction יהיה נמוך, אך לא יהיה ניתן ללמידה כלום מכך. בעיה נוספת של ערך גובה היא יותר פרמטרים, שכן ב-Autoencoder משתמש בעיקר בקובולוציות, כמעט כל הפרמטרים של הרשות נמצאים בשכבה ה FC המובילה לפיצ'רים ב encoder, ובשכבה ה FC ש"פורסט" אותם ב decoder.

פונקציית loss ב EBGAN מוגדרת כך, כאשר הסימן $+$ מתייחס ל hinge loss:

$$\begin{aligned}\mathcal{L}_D(x, z) &= D(x) + [m - D(G(z))]^+ \\ \mathcal{L}_G(z) &= D(G(z))\end{aligned}$$

$$\text{where } [u]^+ = \max(0, u)$$

כאשר m הוא Hyperparameter שמנגדיר מה המרחק שאנו רוצים בין תמונה מצויפת לפולט עבורו. ועכשו ל BEGAN, כמו ב BEGAN, גם ב BEGAN משתמשים ב Autoencoder. למעשה, ל Discriminator יש שתי מטרות:
 1. להיות Autoencoder טוב, קלומר להצלחה לשחרר بصورة טובה תמונות מה-Dataset.
 2. להיות Critic טוב, קלומר להצלח "להרים" כמה שאפשר תמונות מצויפות.

רשת BEGAN מנסה כמו WGAN למצוא על ידי מרחק Wasserstein את המרחק בין התפלגות התמונות האמיתיות לבין התפלגות התמונות המצוייפות.
 נקודת האיזון היא הנקודה שמקיימת:

$$\mathbb{E}[\mathcal{L}(x)] = \mathbb{E}[\mathcal{L}(G(z))]$$

בנקודת האיזון זו ה Discriminator לא יוכל להבדיל כלל בין התמונות. על מנת לשלוט באיזון בין שתי מטרותיו של ה Discriminator, הארכיטקטורה מגדרה Hyperparameter חדש γ , שמייצג את היחס בין התפלגותיות.

$$\gamma = \frac{\mathbb{E}[\mathcal{L}(G(z))]}{\mathbb{E}[\mathcal{L}(x)]}$$

כאשר γ בין 0 ל 1. ערכי γ נומקיים יתנו תמונות יותר טובות במחair של שונות נומוכה בין התמונות, כלומר mode collapse, וערך γ גבוהים יתנו שונות גבוהה יותר בתמונות. במאמר מוצגות הרצאות עם ערכי γ שונים.

לבסוף, הנוסחה הסופית של BEGAN:

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x) - k_t \cdot \mathcal{L}(G(z_D)) & \text{for } \theta_D \\ \mathcal{L}_G = \mathcal{L}(G(z_G)) & \text{for } \theta_G \\ k_{t+1} = k_t + \lambda_k (\gamma \mathcal{L}(x) - \mathcal{L}(G(z_G))) & \text{for each training step } t \end{cases}$$

על מנת שהאימון יהיה יציב, משתמשים בפרמטר k , שמאזן על פי ערך γ וערך ה Loss הנוכחיים את היחס בין מטרותיו של ה Autoencoder. את k מתחילה ב 0, והוא חסום בין 0 ל 1. יתרון נוסף

של k , שהוא נותן לנו Autoencoder להתאמן בהתחלה כמעט רק על תמונות אמיתיות כדי שיוכל ללמידה תחילה את ההתפלגות שלהן. המשטנה α הוא מעין מועד למידה לא.

אחת הבעיות של GAN היא שקשה לקבוע מתי הוא התכנס, כי מדובר במשחק סכום אפס, כלומר אם ערך Loss אחד ירד, השני כנראה עלה ולא נדע מתי הגענו לנקודת התכנסות. BEGAN מגדירה את הערך M שהוא מודד להתקנסות הרשת.

$$\mathcal{M}_{global} = \mathcal{L}(x) + |\gamma \mathcal{L}(x) - \mathcal{L}(G(z_G))|$$

לסיכום, הרשת מגדירה את ה z s Hyperparameters הבאים, בנוסף לטריויאלים (מקדם למידה וכו'):

- ערך γ - יחס ההתפלגות.
- ערך α - מקדם הלמידה של k .
- ערך h_z - גודל הרעש בקלט של ה Generator.
- ערך hx - גודל וקטור הפיצרים ב Autoencoder. בדרך כלל נרצה ש $hx = hz$.

הניסוי שבוצע:

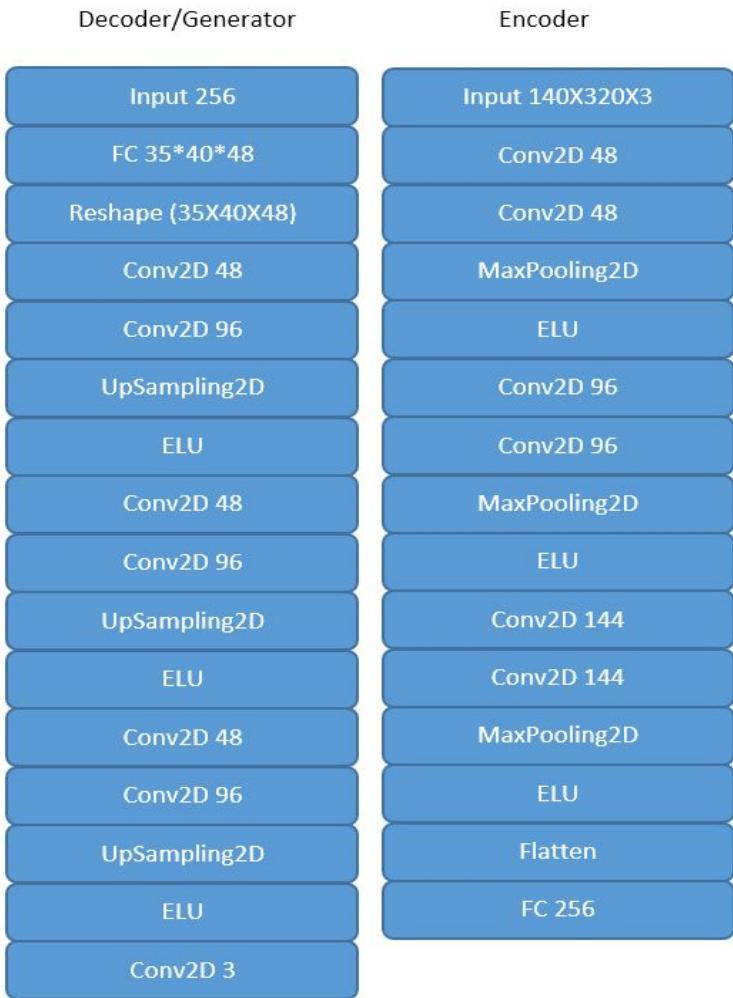
לאחר ניסיונות לא מוצלחים רבים, שכללו משחקים על כל ה-Hyperparameters הרבים שמנגדירה הארכיטקטורה הזאת, ארכיטקטורת הניסוי הסופית:

- $\gamma = 0.7$
- $\lambda k = 0.003$
- $hx = hz = 256$
- Adam lr = 1e-4
- Batch size 16
- ELU activation
- MAE (Mean Absolute Error)

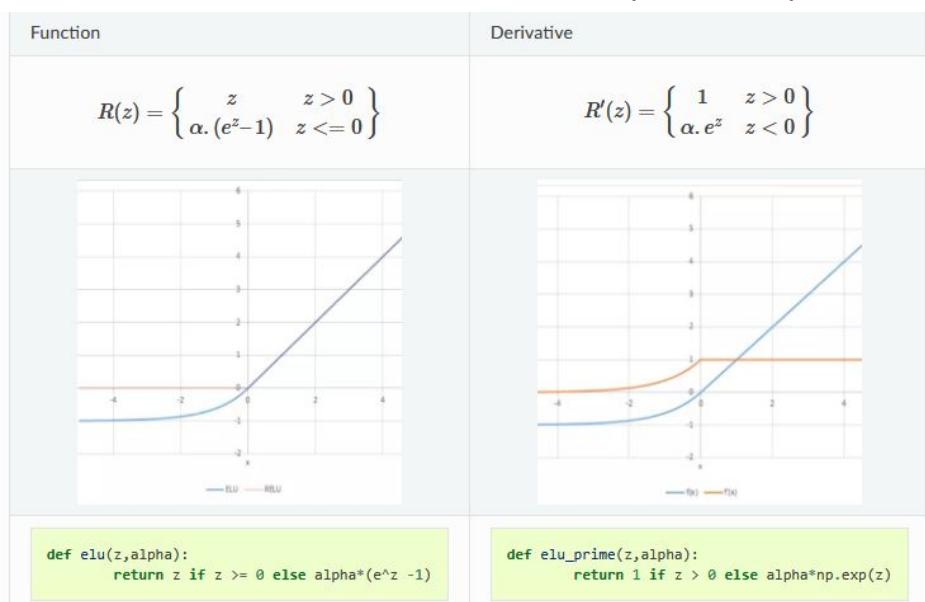
כאמור ה Discriminator מורכב משני חלקים, encoder ו decoder, באופן הבא:

Layer (type)	Output Shape	Param
<hr/>		
input_1 (InputLayer)	[(None, 140, 320, 3)]	0
encoder (Sequential)	(None, 256)	25,525,840
decoder (Sequential)	(None, 140, 320, 3)	17,434,323
<hr/>		
Total params: 42,960,163		

למעשה, מבחינת גודלי הפלט והקלט, ה decoder generator והוא זהים. לכן, הם חולקים את אותה הארכיטקטורה, אך הם לא חולקים משקלות או פרמטרים אחרים.



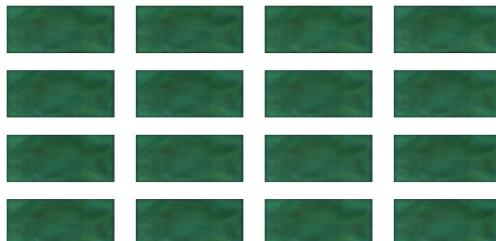
בניגוד להמלצות DCGAN, כמו לא להשתמש ב MaxPooling ו UpSampling, במאמר נעשה בהם שימוש. הכוונה ב UpSampling זו nearest neighbor שימוש בפונקציית ELU, שהוא מעין גרסה מוחלקת בעזרת אקספוננט של U, כאשר היא גזירה לכל R.



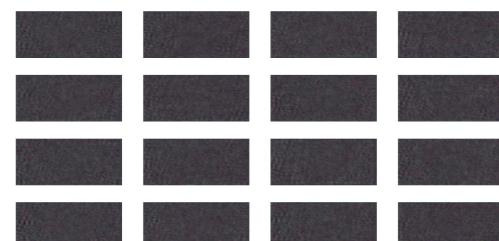
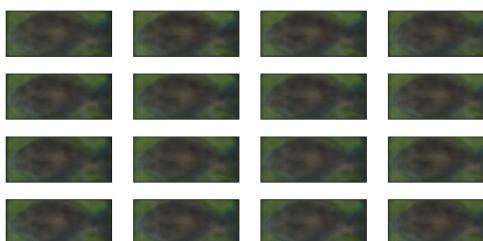
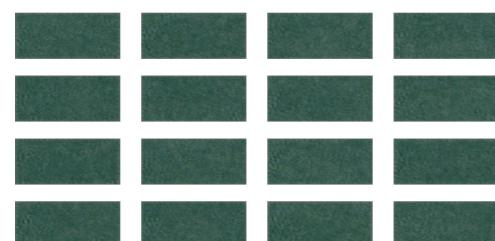
תוצאות:

מעניין לראות שכבר לאחרEpochים הראשונים, ה Autoencoder מצליח להבין מה זה דג,
ולמעשה הופך כל דבר למעין דג:

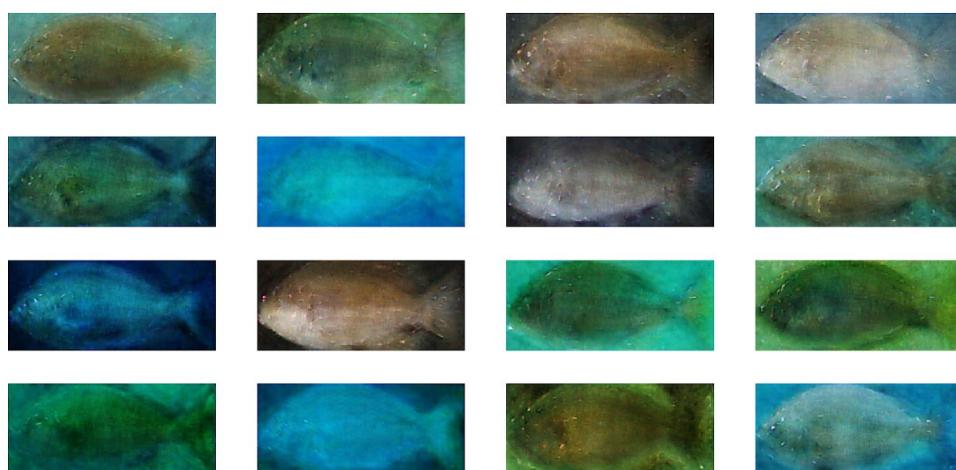
אחריו



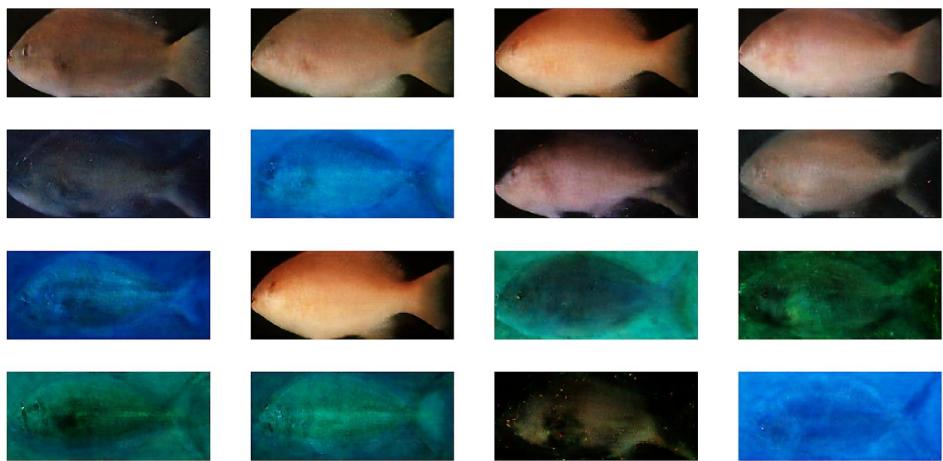
לפני (פלט הGenerator)



:50 אפק



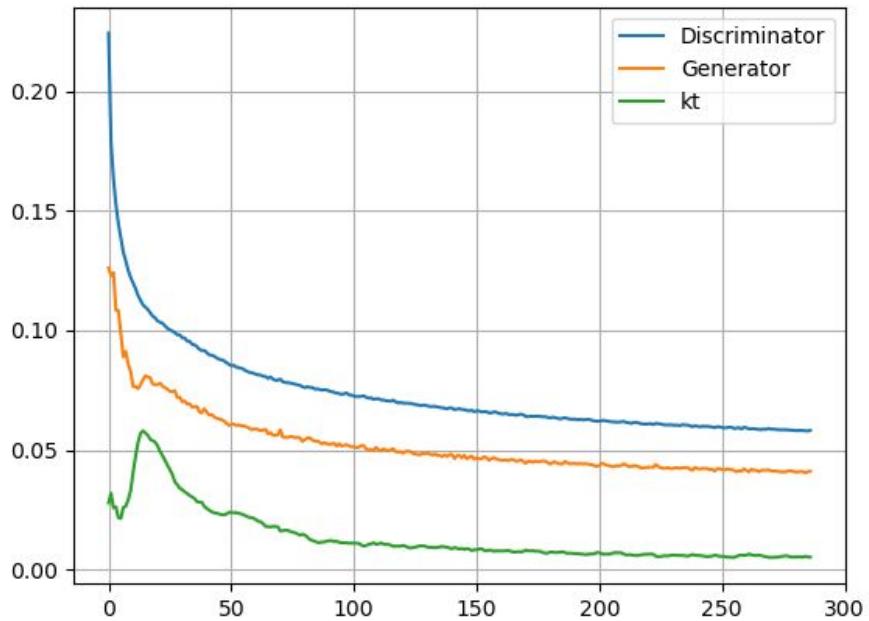
:200 אפיק



:260 אפיק

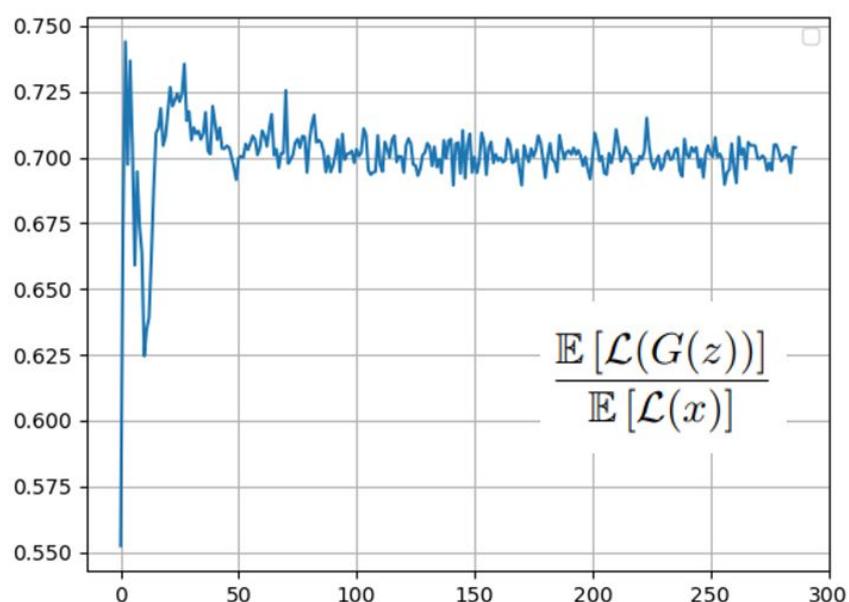


גרף התוצאות Loss:

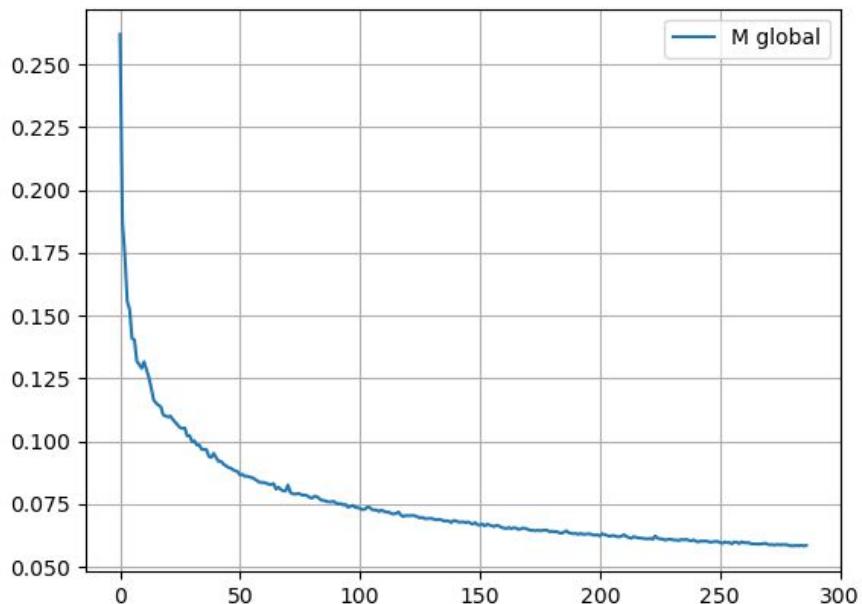


ניתן לראות בגרף את הקשר בין k ל loss של generator. לדוגמה, בערך בסביבות 20 epoch מתרחש ערך ה generator ירד, מה שגרם לעלייה בערך k , שגרמה לעלייה ב generator. זהה הדגמה של האיזון שמבצע k .

גרף מעניין נוסף הוא היחס בין ה Lossים. כאמור, ערך ה χ שנבחר הוא 0.7, כלומר היחס בין Loss ה Generator ל Loss ה Discriminator צריך להיות 0.7, ואכן ניתן לראות זאת בגרף, שmagiu בערך ל 0.7 בסוף האימון.



ולבסוף, גרפּ M כפונקציה של epoch:



לסיכום, התוצאות יכולו להיות טובות הרבה יותר, אך מחסום המשאבים מנע את ההתקדמות. לשם השוויה, במאמר נעשה שימוש בקונבולוציות של 256 כדי לאמן על A,Celeb, תמונות קטנות בהרבה מהדגים. זמן האימון במאמר היה יומיים רצופים על שני כרטיסים גרפים. פה נעשה שימוש ברשת רדודה בהרבה וצרה בהרבה לתמונות גדולות בהרבה, אם כי פשוטות בהרבה מתונות פנים. גודל הרשות כפה שימוש ב Batch size נמוך יחסית. זמן האימון ל Epoch היה כ-3 דקות. נעשו ניסיונות יותר עמוקים, אך זמן האימון הגיע לכ-5 דקות.

מקורות:

1. https://medium.com/@jonathan_hui/gan-energy-based-gan-ebgan-boundary-equilibrium-gan-began-4662cceb7824
2. <https://arxiv.org/pdf/1703.10717.pdf>