מגיש : שי ארץ קדושה – 203276258

דו"ח עבודה 4 -Super Resolution

במסגרת הקורס קיבלנו את עבודה זו בנושא Super resolution שמטרתה היא להכיר ולאמן Fully convolutional networks בארכיטקטורות שונות כדי לחזות את איכות הרזולוציה של תמונות שונות באיכות תמונה נמוכה לאיכות תמונה גבוהה יותר ולהשוות בין התוצאות המתקבלות של כל מודל.

שלבים מקדימים:

- **הכנת מאגר התמונות**

לצורך ביצוע המשימה נעזרנו במאגר התמונות של PascalVOC 2007 .

מתוך מאגר תמונות זה התבקשנו ליצור dataset המכיל את התמונות בשלושה גדלים שונים והם:

72X72, 144X144, 288X288. לביצוע שלב זה טענו את התמונות של המאגר וביצענו עליהם resize לגדלים שנתבקשנו ושמרנו אותם בשלוש רשימות בהתאם לגודל של כל תמונה.

מאגר התמונות של PascalVOC 2007 מכיל 5011 תמונות שונות ומכיוון שהרצת מודלים על כמות תמונות גדולה כזו עשויה לקחת זמן רב בחרתי תחילה להשתמש רק ב-100 תמונות מתוך המאגר לצורך בדיקת תקינות המודלים באופן ראשוני. לאחר שראיתי שהמודלים רצים בצורה טובה על כמות קטנה של תמונות נעזרתי ב-Data Generator לטעון את כל מאגר התמונות לשימוש בכל אחד מהמודלים.

- **כתיבת מחלקות שונות ופונקציות עזר**

PSNR (Peak signal to noise ratio)- מדד המשמש לבדיקת איכות שחזור האות בין המידע המקורי למידע שהתקבל או במקרה שלנו בין התמונה המקורית לתמונה שהתקבלה לאחר הרצת המודל. בעזרת מדד אוכל להראות את ההבדלים בין התוצאות המתקבלות בין כל מודל. בהמשך הגדרתי את המדד הזה גם ב-loss שלפיו המודל יתאמן ולשם כך הייתי צריך לשים את הסימן מינוס בערך שמחזירה הפונקציה כי רק כך המודל הבין שעליו למזער את ה-loss כאשר הוא מוגדר כ-PSNR.

המחלקה PredictionCallback- מחלקה זו יורשת את מחלקה Callback של Keras והיא נועדה לכך שבעת סיום epoch של כל מודל יתבצע predict על 12 תמונות כך שבהמשך אוכל ליצור gif שיראה איך איכות התמונה השתפרה בין כל epoch. לצורך זה הגדרתי מילון שהמפתח שלו יהיה שם המודל שיינתן לו בהרצה של כל אחד מהמודלים והערך של אותו מפתח יהיה רשימה של התמונות שנשמרו עבור כל epoch. כך בהמשך אוכל להציג gif לכל אחת מ-12 התמונות לכל אחד מהמודלים לפי שם המודל.

פונקציה Create_callbacks- פונקציה זו תיקרא כ-callback בעת הרצה של כל מודל. פונקציה זו מכילה שימוש ב-early_stopping במקרה שהמודל מפסיק להשתפר לפי ה- Loss שהגדרנו, check point לצורך שמירת המשקולות הטובות ביותר מבין ה-epochs בעת הרצה של כל מודל לפי שם המודל וקריאה למחלקה PredictionCallback שיצרתי כדי לשמור את התמונות בין כל epoch לצורך יצירת gif.

פונקציה Create gif- פונקציה זו מקבלת את שם המודל לפי איך שהוגדר במהלך ההרצה שלו, מס' שמייצג את גודל התמונה (0 עבור 144X144 ו-1 עבור 288X288) ואינדקס של תמונה (מתוך 12 תמונות שבחרתי לבצע עליהן predict בסוף כל epoch) ויוצרת gif בעזרת כל תמונות שנשמרו במהלך כל epoch בעת הרצת המודל. לצורך יצירת ה- gif אני נעזר בספריית imageio כך שאוכל להציג את השינוי בתמונה בין כל epoch באמצעות ה-gif.

פונקצייה plot multi imgs pred- פונקציה זו מקבלת אינדקס של התמונה אותה נרצה להציג, רשימה של התמונות שהוכנסו כ-input למודל בגודל 72X72, רשימה של תמונות בגודל המקורי אותו רצינו לחזות (144X144 או 288X288) והרשימה של התמונות אותם המודל הביא כפלט ותציג את התמונה לפי האינדקס ובהתאם לגדלים כך יהיה אפשר לראות אם נעשה שינוי בין הקלט של המודל לפלט שהתקבל וביחס לתוצאה אותה רצינו לשאוף להגיע.

מגיש : שי ארץ קדושה – 203276258

פונקצייה history to plot- פונקציה זו מקבלת משתנה המייצג את התהליך שעבר המודל בהתאם ל- loss והמטריקות שהוגדרו עבורו (שהתקבל מביצוע fit), רשימה שהוגדרה מראש של השמות עבור הפלטים של PSNR ל-train ול-validation ורשימה של שמות של ה-loss שהוגדרו מראש עבור ה-train וה-validation ותציג שני גרפים לאורך ה-epochs בעת תהליך האימון של המודל. גרף אחד עבור ה- loss שהוגדר וגרף שני עבור מטריקה של PSNR. בעזרת גרפים אלו יהיה אפשר לדעת האם המודל השתפר ולמד לפי ה- Loss והמטריקות שהוגדרו עבורו.

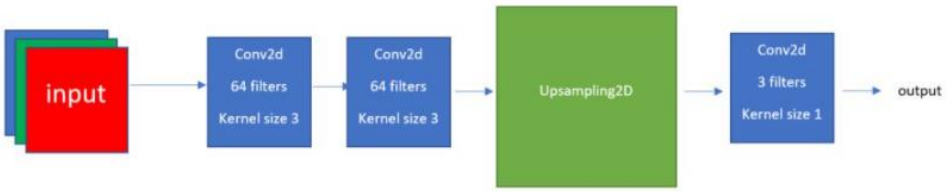
כתיבת Data generator- לצורך טעינת כלל התמונות לאימון המודלים יצרתי מחלקה UnionDataGenerator שיורשת מהמחלקה Sequence של Keras. במחלקה זו בעצם כביכול יצרתי שלושה Data generators שבעת ביצוע fit על המודלים הם יבצעו resize על התמונות וכך יהיה אפשר לטעון את כל התמונות בעת אימון כל מודל. במחלקה הגדרתי משתנים בבנאי שהם batch size לצורך שינוי ה-batch size באופן דינמי ו-mode שמגדיר אם התמונות הם לצורך train או validation. את המשתנה shuffle הגדרתי False על מנת לשמור על הסדר של התמונות שלא ישתנה בין המודלים שאוכל לראות את השינויים ואת הערך שמשפיע לדעתי הכי על התוצאות והוא interpolation הגדרתי תחילה ל-bucubic (בהמשך שינתי אותו כשניסיתי לנתח את ההשפעה של ה-interpolation על טיב התוצאה).

יצירת המודלים

כלל השכבות של המודלים הבסיסיים נבנו בהתאם לארכיטקטורה שקיבלנו מראש באיורים (מצורפים בדוח בעת הצגה של כל מודל). ישנם מודלים שניסיתי לשפר בהם שינתי את השכבות של הרשת (ארחיב עליהם בהמשך). בנוסף תחילה הגדרתי את ה-loss של כל המודלים להיות לפי MSE בהמשך נשנה אותו ל-PSNR ואעשה השוואה בין התוצאות.

- **שלב א'**

בשלב הראשון התבקשנו ליצור מודל ראשון פשוט לפי ארכיטקטורה מוגדרת שקיבלנו שמקבל תמונה ברזולוציה 72X72 ומוציאה כפלט תמונה ברזולוציה 144144.



מכיוון שמודל זה מוציא רק פלט אחד של תמונה בגודל 144X144 בשונה מכל המודלים האחרים שאבנה בהמשך עבור שני פלטים בחרתי לא להשתמש ב-Data generator. החלטה זו נובעת מכיוון שהגדרתי את ה-data generator עבור שני פלטים בגלל כל המודלים האחרים והוא אינו מתאים למודל עם פלט אחד (בנוסף רציתי להראות על מודל אחד איך השתמשתי במעט תמונות בשלב המקדים לצורך אימון המודל לפני שביצעתי שימוש בכל התמונות ולכן בחרתי להראות זאת על המודל הפשוט ביותר-כמו שעשינו בביתה). הגדרתי את מס' התמונות לאימון להיות 80 ומס' התמונות לוולידיציה 20.

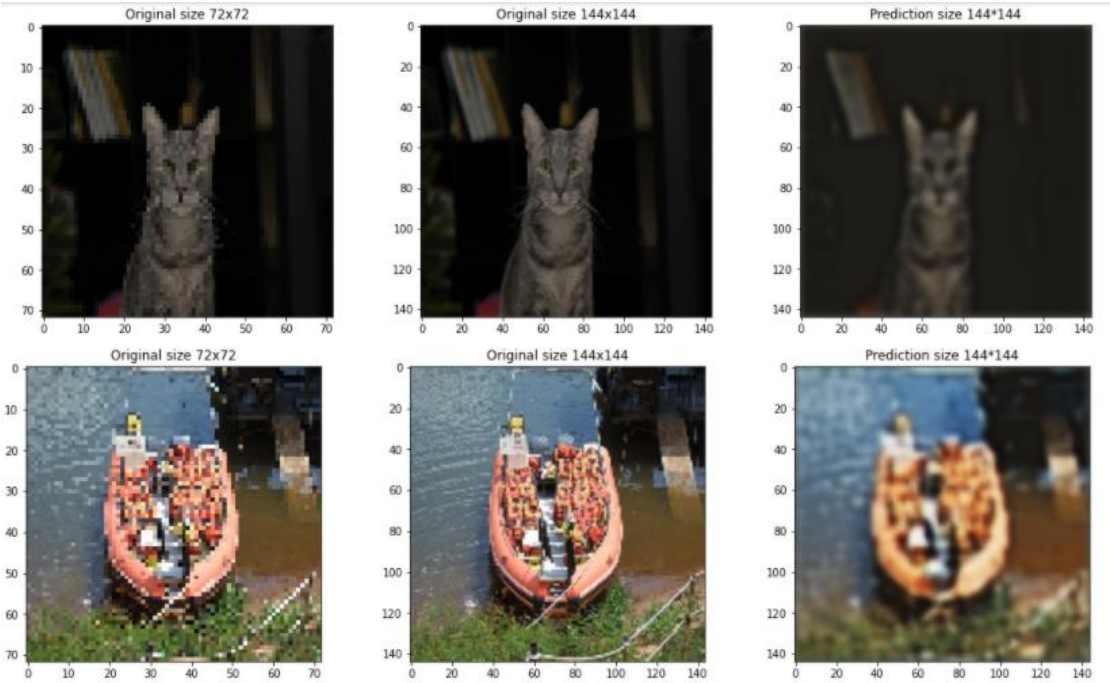
המודל:

```
def get_simple_model():
  input_shape = (72,72,3)
  inputs = Input(shape=input_shape)
  x = Conv2D(64,(3,3),padding='same',activation='relu')(inputs)
  x = Conv2D(64,(3,3),padding='same',activation='relu')(x)
  x = UpSampling2D(size=2,interpolation='bilinear')(x)
  output_1 = Conv2D(3,kernel_size=1,activation='sigmoid')(x)

  model = Model(inputs=inputs, outputs=[output_1])
  model.compile(loss='mse',optimizer='adam',metrics=['acc',PSNR])
  return model

model = get_simple_model()
model.summary()
```

מגיש : שי ארץ קדושה – 203276258

אימנתי את המודל לאורך 20 epochs.

להלן חלק מהתמונות שהתקבלו כתוצאה ממודל זה כשניסיתי לשפר בעזרתו את הרזולוציה של התמונות מ-72X72 ל-144X144.



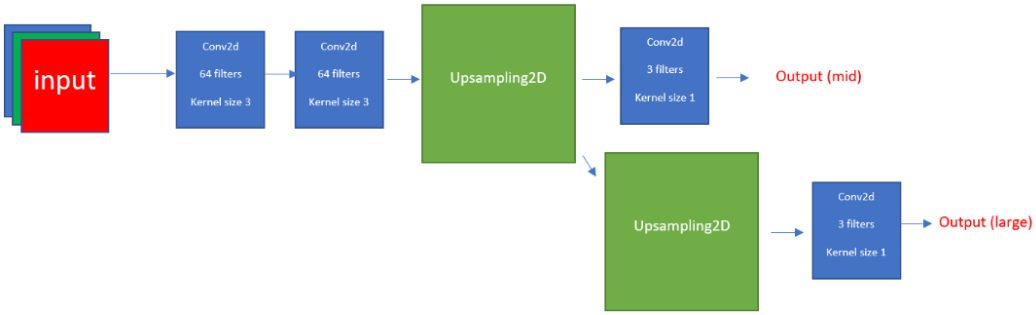
ניתן לראות שהתוצאות שמתקבלות עבור המודל הבסיסי הזה הם יחסית נמוכות, אומנם בתמונות לא רואים את הפיקסלים אבל הן די מרוחות ומטושטשות. ייתכן שזה נובע מכיוון את המודל הזה אימנתי ללא שימוש ב-data generator על מעט תמונות ולכן התמונות נטענות ב-interpolation ברירת מחדל שהוא nearest שהוא יחסית פשוט ולוקח בחשבון רק פיקסל אחד כל פעם לעומת bicubic ו-bilinear. את ההנחה הזו אבדוק בהמשך כשאשווה בין interpolations שונים בעת שימוש ב-data generator על כל התמונות.

<u>את המודלים הבאים אימנתי על 80% מכלל התמונות הנמצאות במאגר ועבור 20% הנותרים השתמשתי לצורך ולידציה. לשם כך נעזרתי ב-Data generator שהגדרתי.
בנוסף לפי בדיקה שערכתי ממאמרים ואתרים שונים ראיתי שה-interpolation שלרוב מביא את התוצאות הטובות ביותר הוא bicubic שלוקח בחשבון 16 פיקסלים לחישוב הפלט ולכן השתמשתי בו ב-data generator על המודלים שהתבקשנו לממש. בהמשך ביצעתי השוואה עם Interpolations שונים.</u>

- <u>שלב ב'</u>

בשלב השני התבקשנו ליצור מודל לפי ארכיטקטורה מוגדרת שמקבל תמונה ברזולוציה 72X72 ומוציאה שני פלטים כפלט תמונה ברזולוציה 144X144 ופלט שני ברזולוציה 288X288. בשונה מהמודל הקודם ביצעתי שוב Up Sampling כדי לקבל גם פלט נוסף ברזולוציה של 288X288.

מגיש : שי ארץ קדושה – 203276258

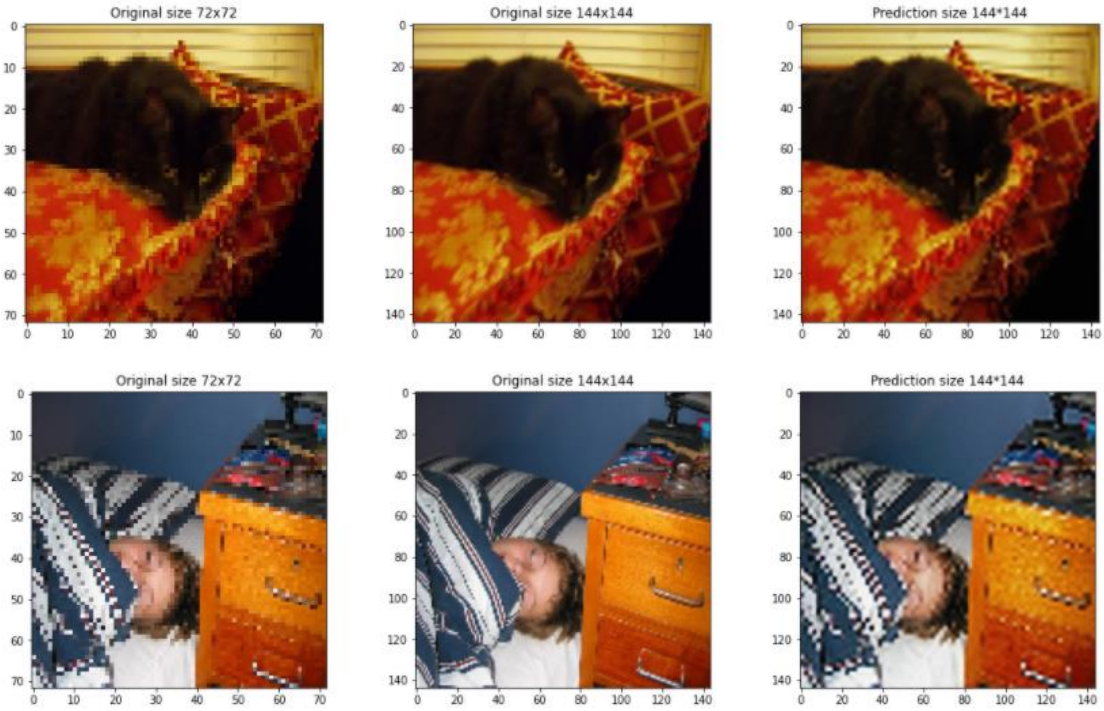


המודל:

```
def get_dual_model():
  input_shape = (72,72,3)
  inputs = Input(shape=input_shape)
  x = Conv2D(64,(3,3),padding='same',activation='relu')(inputs)
  x = Conv2D(64,(3,3),padding='same',activation='relu')(x)
  x = UpSampling2D(size=2,interpolation='bilinear')(x)
  out1 = Conv2D(3,kernel_size=1,activation='sigmoid',name = 'Out_Med')(x)

  out2 = UpSampling2D(size=2,interpolation='bilinear')(x)
  out2 = Conv2D(3,kernel_size=1,activation='sigmoid',name = 'Out_Large')(out2)

  model = Model(inputs=inputs, outputs=[out1,out2])
  model.compile(loss='mse',optimizer='adam',metrics=['acc',PSNR])
  return model

dual_model = get_dual_model()
dual_model.summary()
```

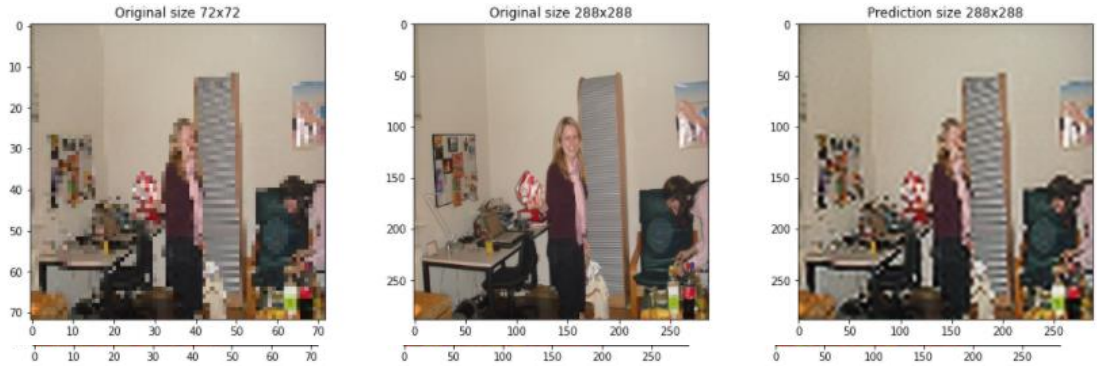
אימנתי את המודל לאורך 20 epochs.

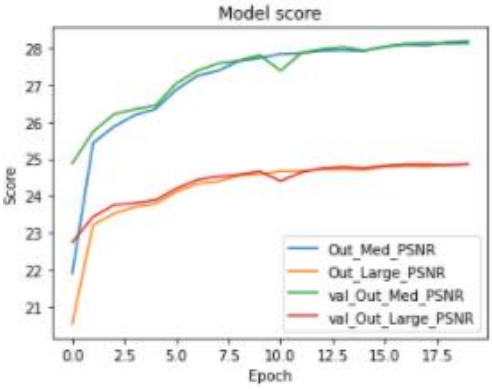
להלן חלק מהתמונות שהתקבלו כתוצאה ממודל זה כשניסיתי לשפר בעזרתו את הרזולוציה של התמונות מ-72X72 ל-144X144.



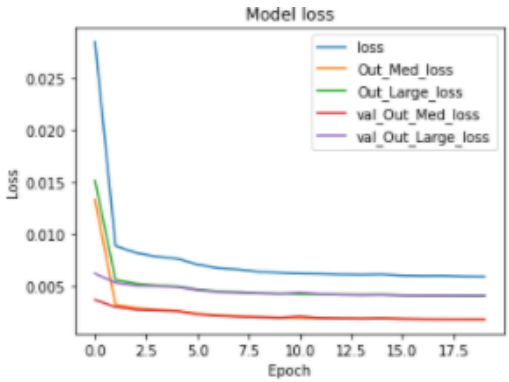
מגיש : שי ארץ קדושה – 203276258

להלן חלק מהתמונות שהתקבלו כתוצאה ממודל זה כשניסיתי לשפר בעזרתו את הרזולוציה של התמונות מ-72X72 ל-288X288.





גרף המראה את השיפור במטריקה PSNR לאורך 20 אפוקים.



גרף המראה את הירידה ב-loss כשהוא מוגדר לפי MSE לאורך 20 אפוקים.

ניתן לראות מהתמונות שקיבלנו כפלט מהמודל עבור שני הגדלים שבעת התמונות יותר מפוקסות אבל עדיין רואים את רוב הפיקסלים כמו התמונה שמודל קיבל כפלט ייתכן שזה נובע בעקבות כך שהגדרתי את ה-interpolation להיות Bicubic.

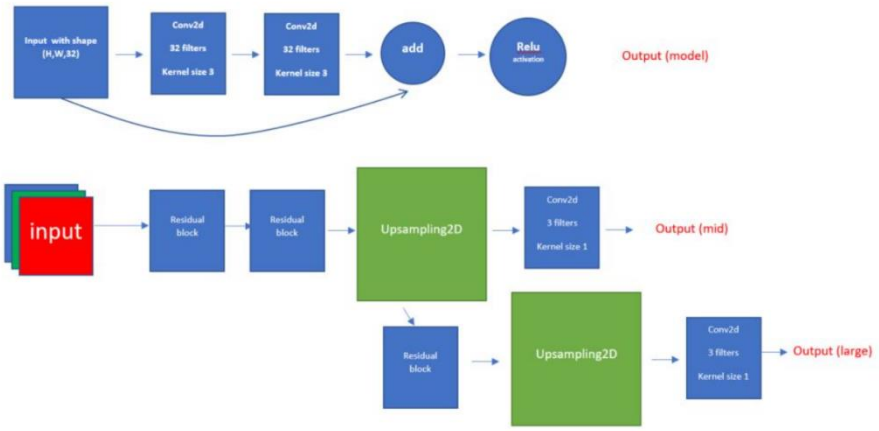
בנוסף בגרף שמציג את ה-PSNR ניתן לראות שגם עבור התמונות בגודל 144X144 וגם 288X288 אין פער בין סט האימון לולידציה ונראה שיש יציבות לאורך ההתקדמות במס' האפוקים ואין overfitting, אבל מצד שני עבור התמונות בגודל 144X144 יש התכנסות לערך PSNR של 29 ועבור התמונות בגודל 288X288 יש התכנסות לערך PSNR של 25. להערכתי ניתן יהיה לשפר את זה במודל הבאים ולעלות את ערך ה-PSNR כאשר המודלים יהיו בנויים בארכיטקטורות מורכבות יותר.

גם עבור ה-loss ניתן לראות שישנה התכנסות לערכים מסויימים לאורך ההתקדמות במס' האפוקים ולהערכתי יהיה ניתן לרדת גם מערכים אלו בעזרת שילוב טכניקות אחרות במודלים הבאים.

- <u>שלב ג'</u>

בשלב השלישי התבקשנו ליצור מודל לפי ארכיטקטורה מוגדרת שמקבל תמונה ברזולוציה 72X72 ומוציאה שני פלטים כפלט תמונה ברזולוציה 144X144 ופלט שני ברזולוציה 288X288. במודל הזה השתמשתי ב-3 residual block שמכילים בתוכם שני שכבות קונבולוציה בעלי 32 פילטרים עם גודל קרנל 3.

מגיש : שי ארץ קדושה – 203276258



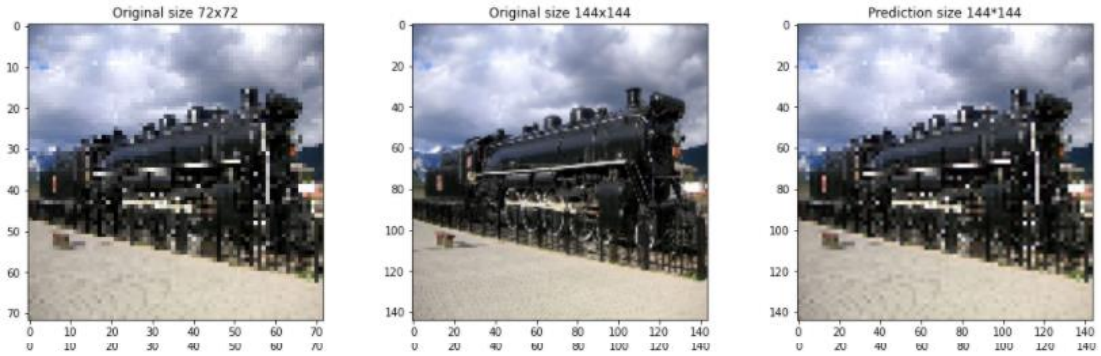
המודל:

```
def get_model():
  inp = Input(shape = (None,None,3))
  x = Conv2D(32,1)(inp)
  x = Activation(LeakyReLU(alpha=0.2))(x)
  x = residual_block(32)(x)
  x = residual_block(32)(x)
  x = UpSampling2D()(x)

  y1 = Conv2D(3,1, padding='same')(x)
  y1 = Activation(LeakyReLU(alpha=0.2),name = 'Out_Med')(y1)

  x = residual_block(32)(x)
  x = UpSampling2D()(x)

  y2 = Conv2D(3,1, padding='same')(x)
  y2 = Activation(LeakyReLU(alpha=0.2),name = 'Out_Large')(y2)



  model_res = Model(inp,[y1,y2])
  if use_psnr:
    model_res.compile(loss=PSNR,optimizer='adam',metrics=['acc','mse'])
  else:
    model_res.compile(loss='mse',optimizer='adam',metrics=['acc',PSNR])
  return model_res

res_model = get_model()
res_model.summary()
```

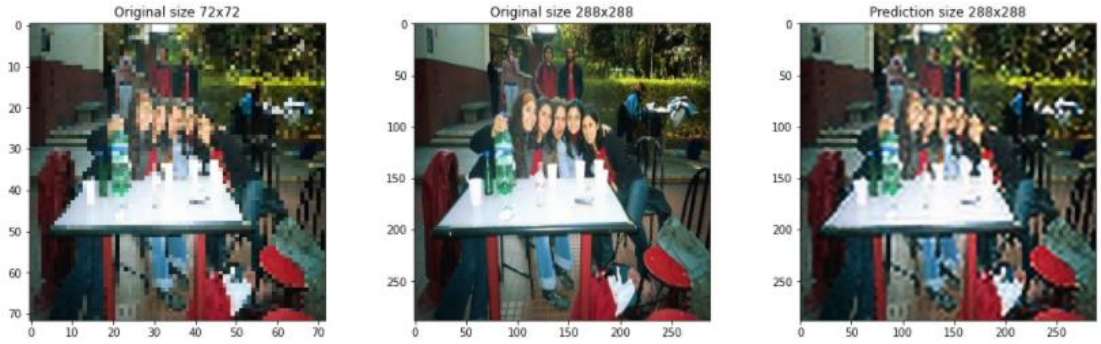
המודל אומן לאורך 20 epochs על 80 אחוז מהתמונות בעזרת שימוש ב-data generator באשר ה-Interpolation מוגדר bicubic.

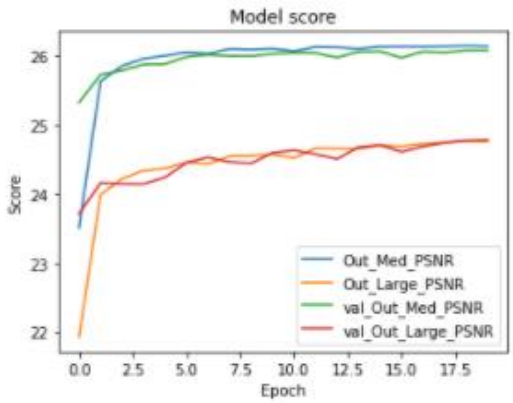
מגיש : שי ארץ קדושה – 203276258

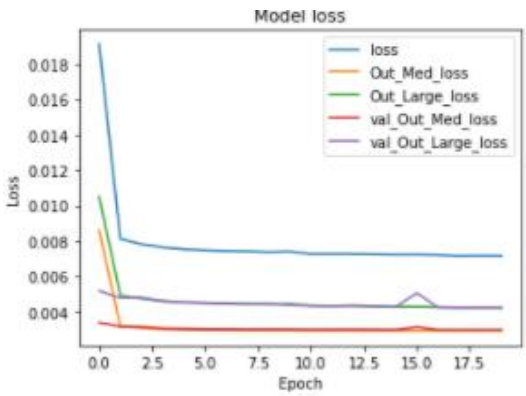
להלן חלק מהתמונות שהתקבלו כתוצאה ממודל זה כשניסיתי לשפר בעזרתו את הרזולוציה של התמונות מ-72X72 ל-144X144.



להלן חלק מהתמונות שהתקבלו כתוצאה ממודל זה כשניסיתי לשפר בעזרתו את הרזולוציה של התמונות מ-72X72 ל-288X288.







גרף המראה את השינוי בנשמרת ה-PSNR לאורך 20 אפוקים.

מגיש : שי ארץ קדושה – 203276258

ניתן לראות שעל אף השימוש בטכניקת residual block בארכיטקטורה הנ"ל, רואים עדיין די בירור בתמונות המתקבלות כפלט ממודל זה את הפיקסלים באופן די דומה לפלט שהוכנס למודל והשיפור לא נראה יחסית

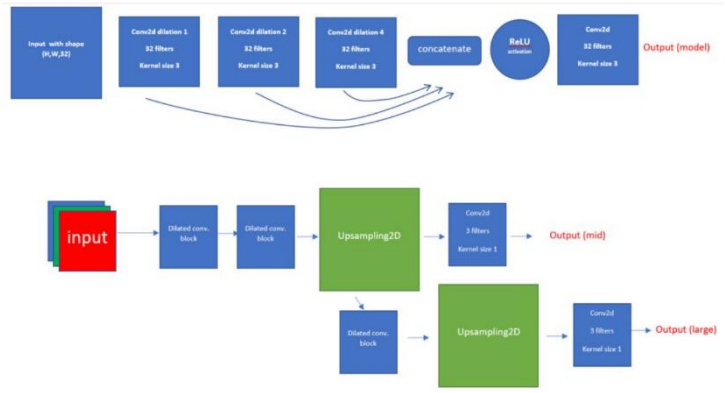
יש לציין שבתמונות קרובות יותר עם פריטים גדולים יותר (לדוגמה הרכבת) ניתן לראות יותר את השיפור בעין לעומת תמונות רחוקות יותר עם פריטים בעלי גודל קטן יותר (לדוגמה הפרצופים של האנשים).

ביחס למודל הקודם ניתן לראות ירידה קלה בהתכנסות של ה-PSNR שבודל זה נע סביב הערך 26 אך לעומת זאת ה-loss מעט יותר נמוך מהמודל הקודם. ההבדלים הללו עדיין קטנים מאוד ולא ניתן כמעט להבחין בהם בעין בתוצאות המתקבלות.

- **שלב ד'**

בשלב הרביעי התבקשנו ליצור מודל לפי ארכיטקטורה מוגדרת שמקבל תמונה ברזולוציה 72X72 ומוציאה שני פלטים כפלט תמונה ברזולוציה 144X144 ופלט שני ברזולוציה 288X288. במודל הזה השתמשתי ב- dilated blocks.

dilated blocks- טכניקה שבה אנו מרחיבים את ההסתכלות על הפיקסלים הנבחרים בהתאם ל - dilation rate שנגדיר. ההנחה הסמויה שעלייה אנו מסתמכים היא שפיקסלים סמוכים הם לפעמים דומים מידי ולכן עדיף להסתכל לאו דווקא על פיקסלים סמוכים אלא על הפיקסל הבא בכל איטרציה.



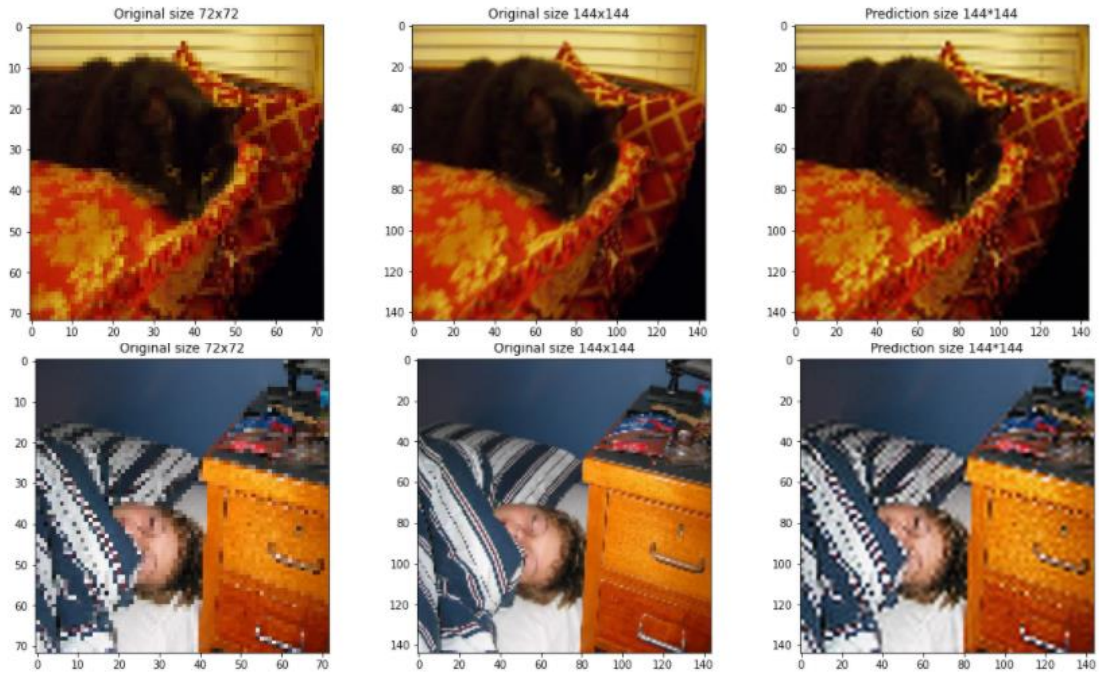
המודל:

```
def dilated_conv(cnl):
  inp = Input(shape = (None,None,cnl))
  d1 = Conv2D(cnl,3,activation=LeakyReLU(alpha=0.2),padding='same',dilation_rate=1)(inp)
  d2 = Conv2D(cnl,3,activation=LeakyReLU(alpha=0.2),padding='same',dilation_rate=2)(inp)
  d3 = Conv2D(cnl,3,activation=LeakyReLU(alpha=0.2),padding='same',dilation_rate=4)(inp)
  c = concatenate([d1,d2,d3],axis=-1)
  a = LeakyReLU(alpha=0.2)(c)
  out = Conv2D(cnl,3,padding='same')(a)
  return Model(inp,out)
```

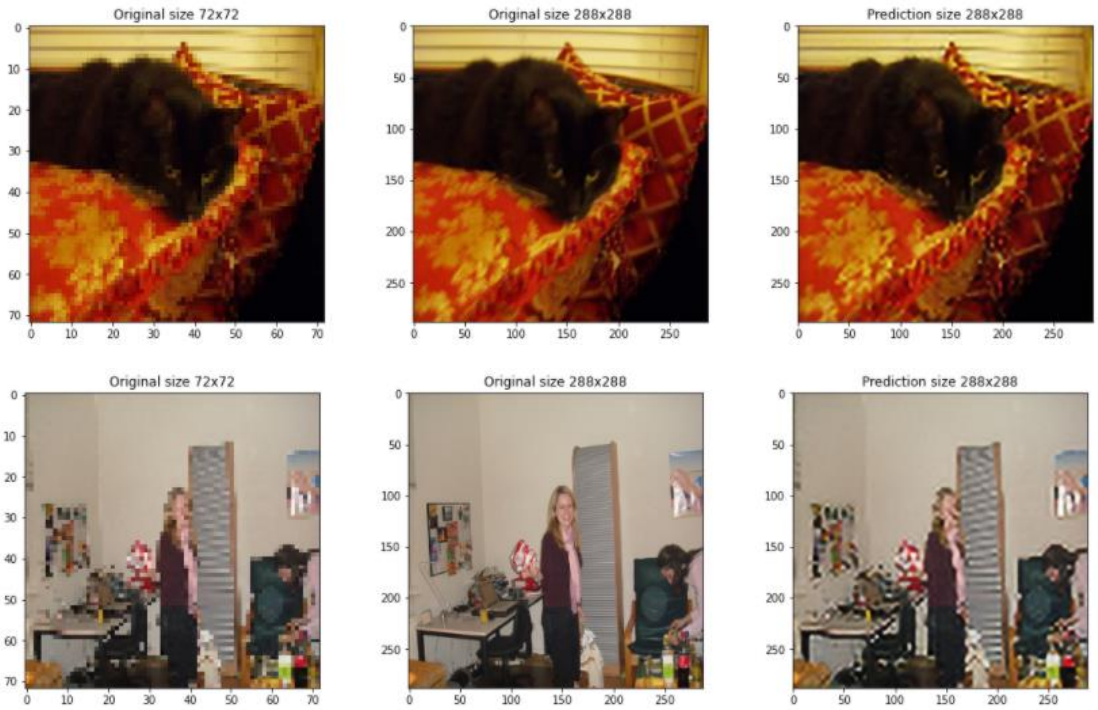
```
def dilated_model():
  inp = Input(shape = (None,None,3))
  x= Conv2D(32,1)(inp)
  x = Activation(LeakyReLU(alpha=0.2))(x)
  x = dilated_conv(32)(x)
  x = dilated_conv(32)(x)
  x = UpSampling2D(interpolation='bilinear')(x)
  y1 = Conv2D(3,1,padding='same')(x)
  y1 = Activation('sigmoid',name = 'Out_Med')(y1)
  x = dilated_conv(32)(x)
  x = UpSampling2D(interpolation='bilinear')(x)
  y2 = Conv2D(3,1,padding='same')(x)
  y2 = Activation('sigmoid',name = 'Out_Large')(y2)
  model_del = Model(inp,[y1,y2])
  if use_psnr:
    model_del.compile(loss=PSNR,optimizer='adam',metrics=['acc','mse'])
  else:
    model_del.compile(loss='mse',optimizer='adam',metrics=['acc',PSNR])
  return model_del

model_del = dilated_model()
model_del.summary()
```

מגיש : שי ארץ קדושה – 203276258

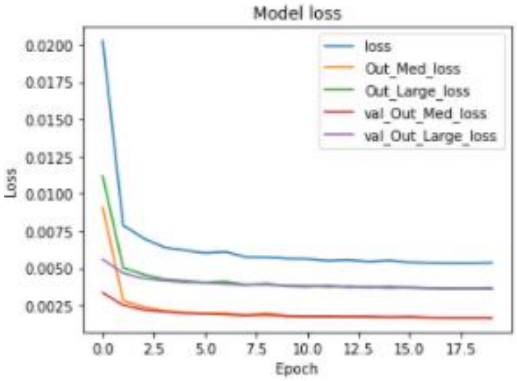
להלן חלק מהתמונות שהתקבלו כתוצאה ממודל זה כשניסיתי לשפר בעזרתו את הרזולוציה של התמונות מ-72X72 ל-144X144.



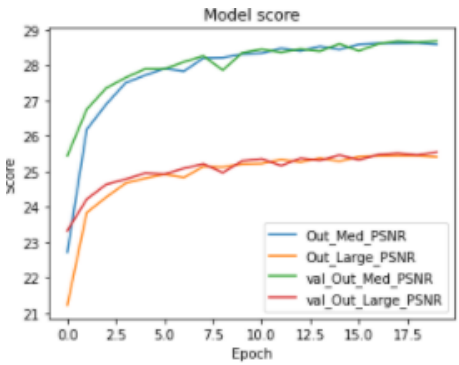
להלן חלק מהתמונות שהתקבלו כתוצאה ממודל זה כשניסיתי לשפר בעזרתו את הרזולוציה של התמונות מ-72X72 ל-288X288.



מגיש : שי ארץ קדושה – 203276258



גרף המראה את הירידה ב-loss כשהוא מוגדר לפי MSE לאורך 20 אפוקים.



גרף המראה את השיפור במטריקה PSNR לאורך 20 אפוקים.

לפי התמונות המתקבלות ממודל זה ניתן לראות שיש שיפור קטן בחלק מהתמונות ביחס למודלים קודמים. בחלק מהתמונות יש מקומות בתמונה בהם הפיקסלים קצת טיפה פחות נראים לעין צריך להסתכל ממש טוב כדי לשים לב לזה, ממבט ראשוני זה די נראה אותו דבר.

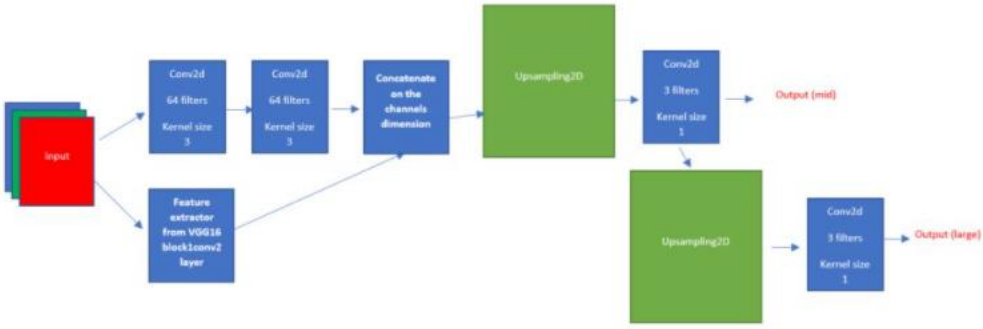
מבחינת ה-PSNR ניתן לראות שיש עלייה קטנה לעומת המודלים הקודמים ועבור המודל הזה הערך נע באזור 29 עבור תמונות בגודל 144X144 כאשר במודלים הקודמים הוא נע בין 26-28. גם מבחינת ה- loss ניתן לראות ירידה מאוד קטנה ביחס למודלים הקודמים וכעט הוא מתכנס לאזור 0.0075 שזה מעט פחות מהתוצאה שקיבלתי במודלים הקודמים.

על פניו נראה שכרגע זהו המודל שנותן את התוצאה הטובה ביותר ביחס ל-PSNR וה-Loss אבל עדין למראית עין קשה לראות את ההבדלים בתמונות המתקבלות ביחס לתוצאות שקיבלתי במודלים הקודמים.

- <u>שלב ה</u>

בשלב החמישי התבקשנו ליצור מודל לפי ארכיטקטורה מוגדרת שמקבל תמונה ברזולוציה 72X72 ומוציאה שני פלטים כפלט תמונה ברזולוציה 144X144 ופלט שני ברזולוציה 288X288. במודל זה בשונה מהמודלים הקודמים טענתי את המודל VGG16 וייצרתי feature extractor שמקבל קלט של VGG ומוציא את הפלט של השכבה השנייה. בחרתי את השכבה השנייה כדי להישאר באותם מימדים של הקלט לפני ביצוע Max polling וכך אני משתמש בפילטרים שקיבלתי מהמודל המאומן על כלל Imagenet. הגדרתי לו Include_top שיהיה false כי לא רציתי לעבוד בקלט בגודל של 224X224. בנוסף הגדרתי לו לא להתאמן על המשתנים של המודל.

את ה- feature extractor מהVGG16 שילבתי במודל לפי הארכיטקטורה המוגדרת כאשר הכנסתי לו את ה-input שאלו התמונות בגודל 72X72 וביצעתי concatenate עם הפלט שקיבלתי משני השכבות הקונבולוציה הראשונות.



מגיש : שי ארץ קדושה – 203276258

המודל:

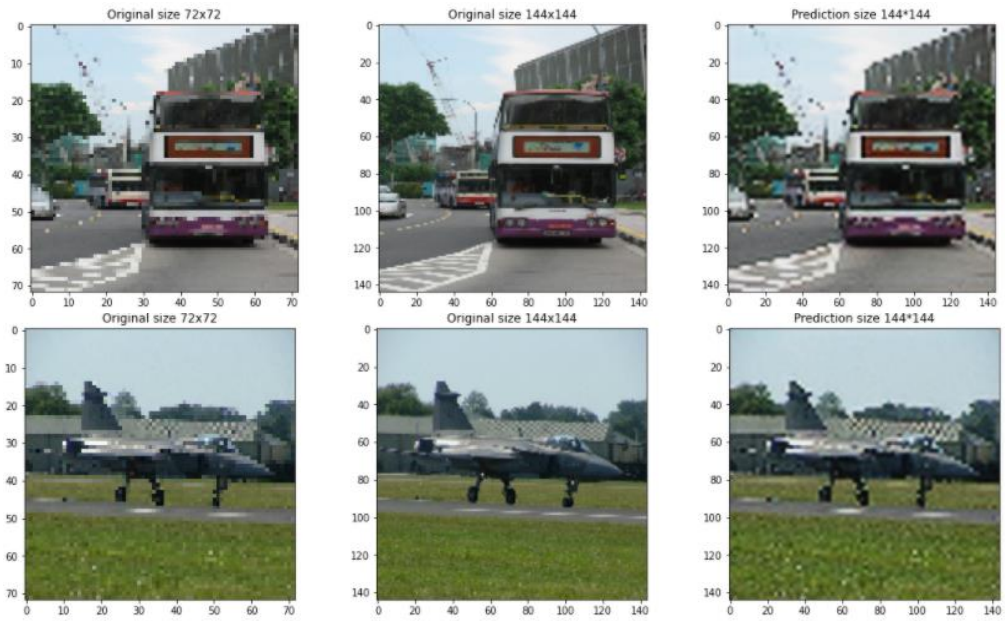
```
from tensorflow.keras.applications.vgg16 import VGG16
```

```
vgg = VGG16(weights = 'imagenet', include_top = False)
fe_model = Model(vgg.input,vgg.layers[2].output)
fe_model.summary()
```

```
def get_model_with_vgg():
  inp = Input(shape=(None,None,3))

  x = Conv2D(64,3,activation=Activation(LeakyReLU(alpha=0.2)), padding = 'same')(inp)
  x = Conv2D(64,3,activation=Activation(LeakyReLU(alpha=0.2)), padding = 'same')(x)

  y = fe_model(inp)
  x = concatenate([x,y],axis=-1)

  x = UpSampling2D(interpolation='bilinear')(x)
  y1 =Conv2D(3,1, padding='same')(x)
  y1 = Activation('sigmoid',name = 'Out_Med')(y1)

  x = Conv2D(32,1)(x)
  x = UpSampling2D(interpolation='bilinear')(x)

  y2 = Conv2D(3,1,padding='same')(x)
  y2 = Activation('sigmoid',name = 'Out_Large')(y2)

  model_with_vgg = Model(inp,[y1,y2])
  if use_psnr:
    model_with_vgg.compile(loss=PSNR,optimizer='adam',metrics=['acc','mse'])
  else:
    model_with_vgg.compile(loss='mse',optimizer='adam',metrics=['acc',PSNR])
  return model_with_vgg
```

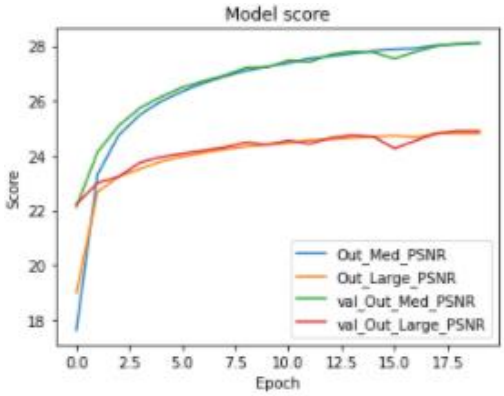
```
Total params: 82,054
Trainable params: 43,334
Non-trainable params: 38,720
```

להלן חלק מהתמונות שהתקבלו בתוצאה ממודל זה בשניסיתי לשפר בעזרתו את הרזולוציה של התמונות מ-72X72 ל-144X144.

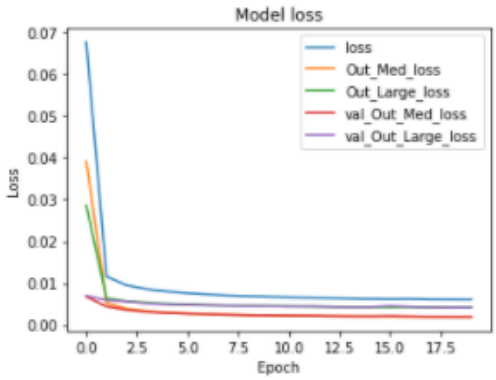


מגיש : שי ארץ קדושה – 203276258

להלן חלק מהתמונות שהתקבלו כתוצאה ממודל זה כשניסיתי לשפר בעזרתו את הרזולוציה של התמונות מ-72X72 ל-288X288.



גרף המראה את השיפור במטריקה PSNR לאורך 20 אפוקים.



גרף המראה את הירידה ב-loss כשהוא מוגדר לפי MSE לאורך 20 אפוקים.

כמו כל המודלים עד כה לא ניתן לראות בעין הבדלים מהותיים בין התמונות המתקבלות ולכן ההערכה של המודל תבוצע בעיקר לפי תוצאת המטריקה PSNR וה-Loss שמוגדר לפי MSE.

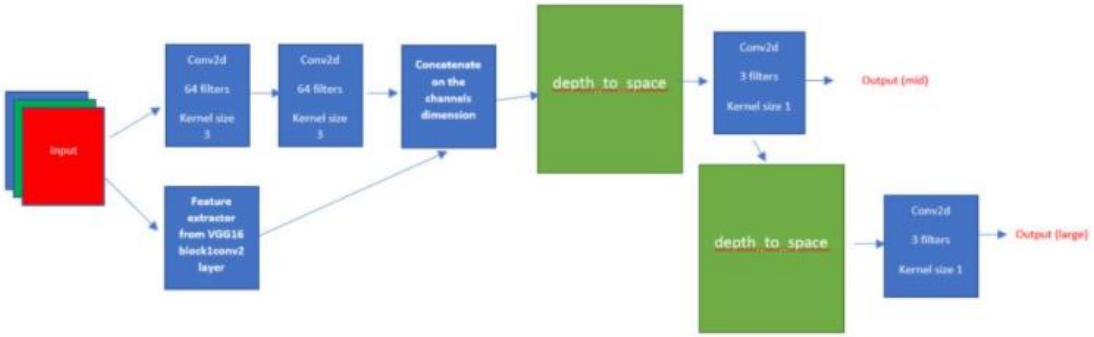
ניתן לראות שמדד ה-PSNR הוא די יציב ביחס לתמונות של סט האימון לעומת התמונות של סט הולידציה למרות שהערך עצמו ירד מעט ביחס למודל הקודם שבמודל זה נע עבור תמונות בגודל 144X144 סביב 28 ועבור תמונות בגודל 288X288 סביב 25.

לעומת זאת ה-Loss יחסית עלה לעומת המודל הקודם שהיה 0.0075 וכרגע הוא התכנס לאזור הערך 0.01.

מגיש : שי ארץ קדושה – 203276258

- שלב ו'

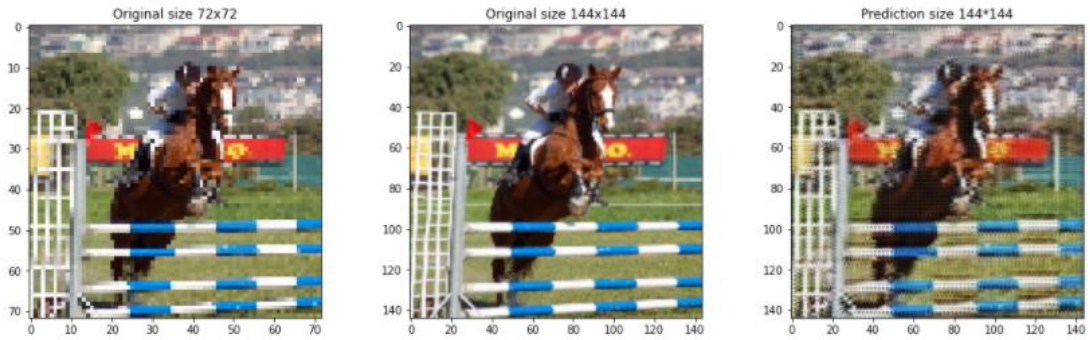
בשלב החמישי התבקשנו ליצור מודל לפי ארכיטקטורה מוגדרת שמקבל תמונה ברזולוציה 72X72 ומוציאה שני פלטים כפלט תמונה ברזולוציה 144X144 ופלט שני ברזולוציה 288X288. כעת במודל זה אשלב טכניקה שבה נחליף את up sampling בשיטה שנקראת-depth to space. בשיטה זו, לומדים פילטרים שונים שאמורים להיות סמוכים אחד לשני במה שהם לומדים. כלומר נייצר 4 feature maps כדי להגדיל את התמונה פי 2 ונרחיב את הפיקסלים בהתאם לאותו פיקסל מכל אחד מה-feature maps שנוצרו.



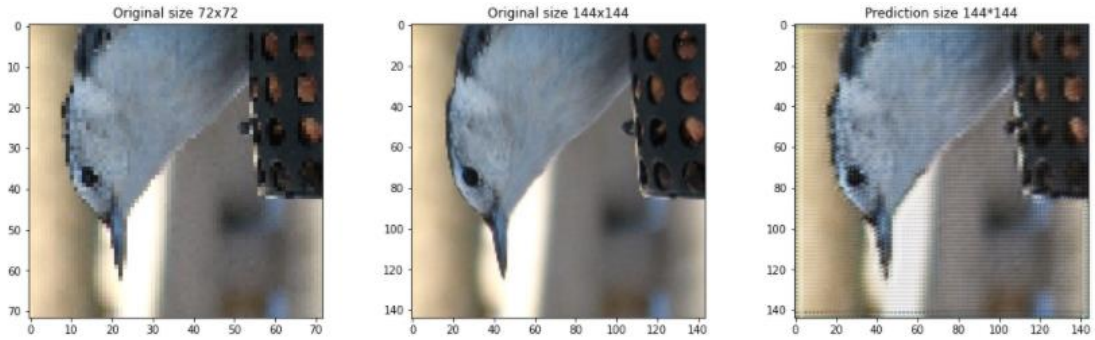
המודל

```
def get_model_depth_to_space():
  inp = Input(shape=(None,None,3))

  x = Conv2D(64,3,activation=Activation(LeakyReLU(alpha=0.2)), padding = 'same')(inp)
  x = Conv2D(64,3,activation=Activation(LeakyReLU(alpha=0.2)), padding = 'same')(x)

  y = fe_model(inp)
  x = concatenate([x,y],axis=-1)

  x = tf.nn.depth_to_space(x,2)
  y1 =Conv2D(3,1, padding='same')(x)
  y1 = Activation('sigmoid',name = 'Out_Med')(y1)

  x = Conv2D(32,1)(x)
  x = tf.nn.depth_to_space(x,2)

  y2 = Conv2D(3,1,padding='same')(x)
  y2 = Activation('sigmoid',name = 'Out_Large')(y2)

  model_with_dts = Model(inp,[y1,y2])
  if use_psnr:
    model_with_dts.compile(loss=PSNR,optimizer='adam',metrics=['acc','mse'])
  else:
    model_with_dts.compile(loss='mse',optimizer='adam',metrics=['acc',PSNR])
  return model_with_dts
```

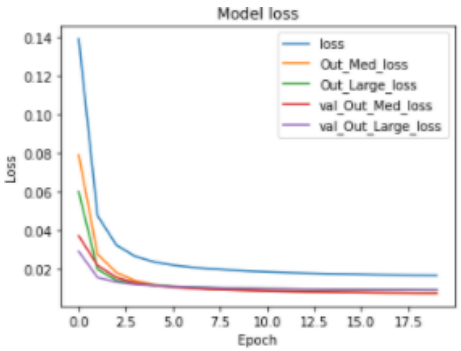
להלן חלק מהתמונות שהתקבלו כתוצאה ממודל זה כשניסיתי לשפר בעזרתו את הרזולוציה של התמונות מ-72X72 ל-144X144.



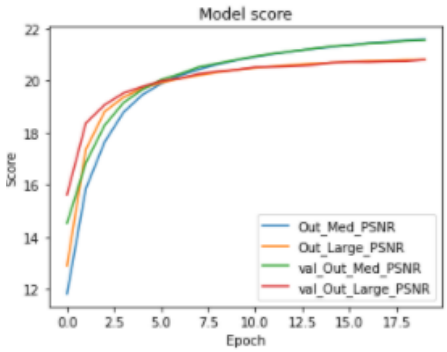
מגיש : שי ארץ קדושה – 203276258



להלן חלק מהתמונות שהתקבלו כתוצאה ממודל זה כשניסיתי לשפר בעזרתו את הרזולוציה של התמונות מ-72X72 ל-288X288.



גרף המראה את הירידה ב-loss כשהוא מוגדר לפי MSE לאורך 20 אפוקים.



גרף המראה את השיפור במטריקה PSNR לאורך 20 אפוקים.

מגיש : שי ארץ קדושה – 203276258

ניתן לראות מהרצת מודל זה כי יש ירידה באיכות התמונות המתקבלות כפלט. בעין רואים בבירור כי ישנה מטריצה של נקודות שמתווספת על התמונה המתקבלת ומורידה את איכות הפלט המתקבל עבור שני הגדלים. לאחר בדיקה נראה שהשילוב של ה- feature extractor של ה-VGG16 עם ה-depth to space הוא זה שמביא ליצירת הנקודות האלו וכאשר ניסיתי להריץ את המודל ללא השילוב של ה-feature extractor אומנם הנקודות נעלמו אבל עדיין לא היה שיפור משמעותי בתוצאות.

מבחינת מדד ה-PSNR ניצן לראות שיש התכנסות לערך שנע בין 20-22 עבור שני הגדלים דבר שלא היה במודלים הוקדמים, אך עם זאת הערך המספרי יחסית די נמוך.

עבור ה-Loss יש התכנסות לערך 0.02 שזהו גם ערך יחסית גבוהה לעומת ה-loss שקיבלתי במודלים קודמים.

אפשר לומר ששילוב הטכניקה של depth to space לא הביאה לעלייה באיכות התמונה וגם במדדים שלפיהם הערכנו את טיב המודל ביחס למודלים קודמים.

- <u>שלב ז'</u>

כעת לאחר שבניתי את כלל המודלים המפורטים בתרגיל לפי הארכיטקטורת המוגדרות מראש והצגתי את התוצאות שלהם בחרתי להתמקד בשני מודלים שהביאו את התוצאות הטובות ביותר הן לפי מראית עין והן לפי איכות המטריקות וניסיתי לנסות לשפר את התוצאות שלהם.

המודלים שנבחרו:

- מודל מס' 4 עם dilated blocks
- מודל מס' 5 המשלב feature extractor ממודל VGG16.

הדרכים בהם ניסיתי לשפר את התוצאות:

- שינוי מבנה הארכיטקטורה של המודלים
- שינוי שיטת ה-interpolation בעת טעינת התמונות בשימוש ה-Data generator.
- שינוי חישוב ה-Loss לפי PSNR.

המודלים המשופרים:

- מודל מס' 4.1 משופר עם dilated blocks

בשונה מהמודל המקורי המוצג בארכיטקטורה שקיבלנו בסעיף ד' הוספתי שתי שכבות נוספות של dilated blocks לפני ביצוע up sampling השני וניסיתי לבדוק האם יש שינוי בתמונות המתקבלות כפלט ובמטריקות שאגדיר בהמשך.

```
def dilated_model_2_loss_psnr():
  inp = Input(shape = (None,None,3))
  x= Conv2D(32,1)(inp)
  x = Activation(LeakyReLU(alpha=0.2))(x)

  x = dilated_conv(32)(x)
  x = dilated_conv(32)(x)

  x = UpSampling2D(interpolation='bilinear')(x)
  y1 = Conv2D(3,1,padding='same')(x)
  y1 = Activation('sigmoid',name = 'Out_Med')(y1)

  x = dilated_conv(32)(x)
  x = dilated_conv(32)(x)
  x = dilated_conv(32)(x)

  x = UpSampling2D(interpolation='bilinear')(x)
  y2 = Conv2D(3,1,padding='same')(x)
  y2 = Activation('sigmoid',name = 'Out_Large')(y2)
  model_del = Model(inp,[y1,y2])
  model_del.compile(loss=PSNR_as_loss,optimizer='adam',metrics=['acc','mse'])
  return model_del

model_del_2 = dilated_model_2_loss_psnr()
model_del_2.summary()
```

מגיש : שי ארץ קדושה – 203276258

בעת הרצתי את המודל המשופר שיצרתי מספר פעמים עם קומבינציות שונות של Interpolation שמוגדר בטעינה של התמונות ב-data generator (nearest, bilinear, bicubic) ו-loss שמוגדר פעם לפי MSE ופעם אחרת לפי PSNR, סה"כ 6 קומבינציות אפשריות עבור המודל.

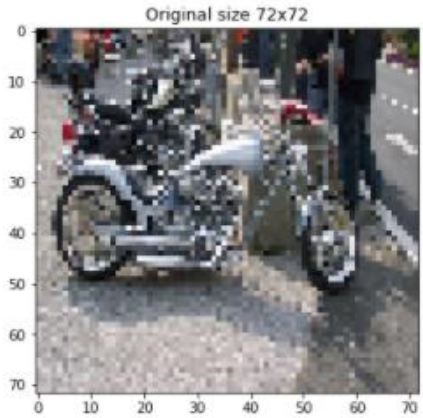
מתוך כלל התוצאות המתקבלות כתוצאה משינוי ה-interpolation וה-loss ניסיתי לערוך השוואה ולהבין מהי שיטת ה-interpolation הטובה ביותר שמוגדרת ב-data generator שיחד עם ה-loss שמוגדר בין האופציות MSE ל-PSNR תביא לתוצאות הטובות ביותר.

ניתוח של התוצאות המתקבלות

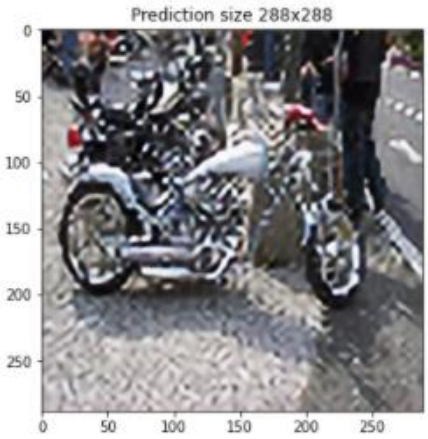
אנסה להציג את ההבדלים ביחס לתמונה אחת בגודל 288X288 עבור כלל הקומבינציות במודל זה.

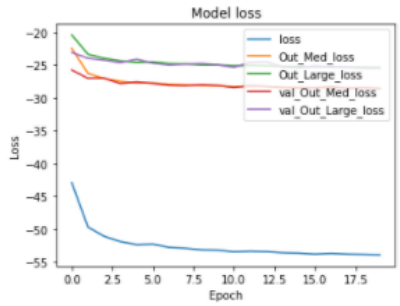
תמונה מקורית

תמונת קלט לאחר ביווץ לגודל 72X72

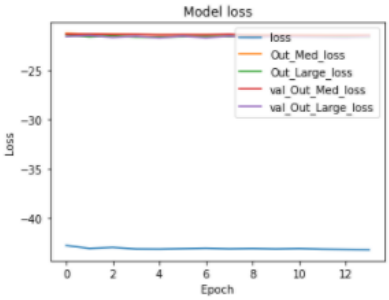


loss- PSNR ,interpolation- bicubic

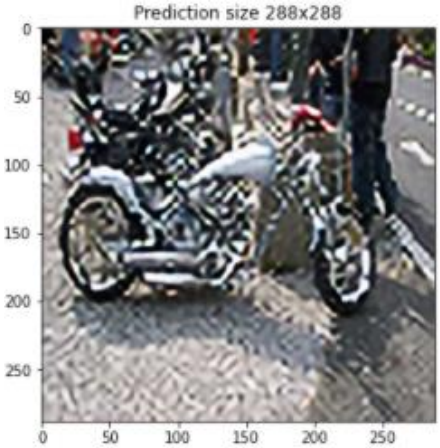


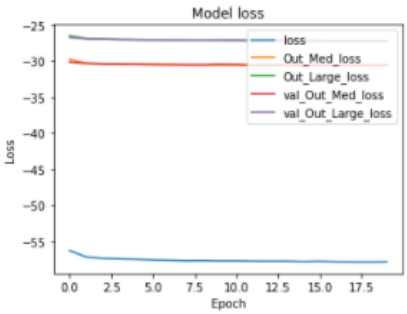


loss- PSNR ,interpolation-nearest



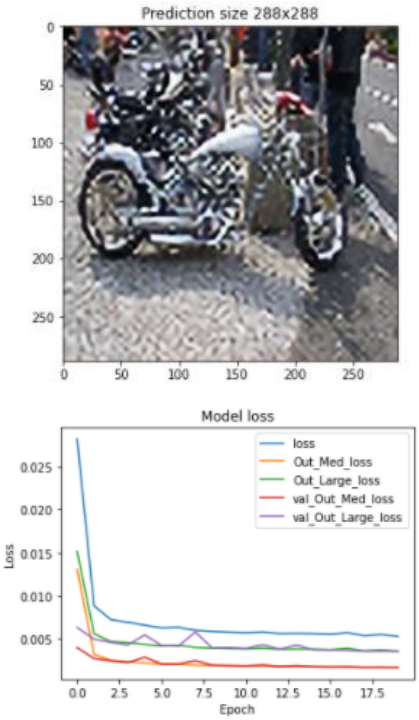
loss-PSNR ,interpolation- bilinear



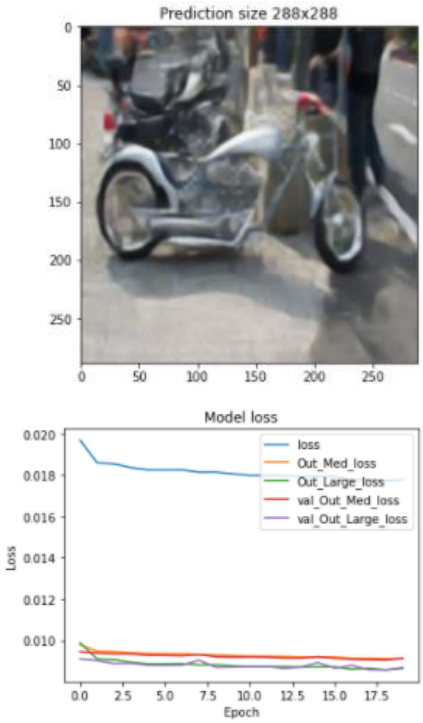


מגיש : שי ארץ קדושה – 203276258

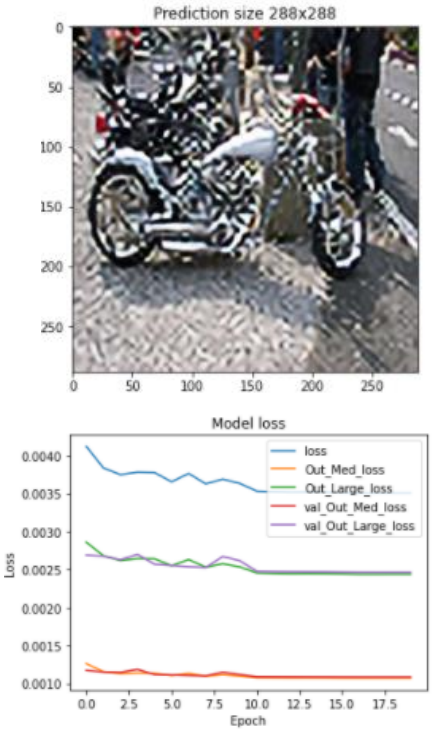
loss- MSE ,interpolation- bicubic



loss- MSE ,interpolation- nearest



loss- MSE ,interpolation- bilinear



ניתן לראות הבדל משמעותי בתוצאות בין כל אחת מהקומבינציות המוצגות.

למראית העין נראה שעבור interpolation- nearest שנחשבת לפשוטה ביותר התמונה המתקבלת כפלט מהמודל היא ברורה יותר לעומת התמונות עם שיטות interpolations אחרות כאשר היא כמעט ללא פיקסלים אך גם מרוחה ולא מפוקסת. בניגוד לכך ניתן לראות שה-loss גם כשהוא מוגדר לפי PSNR וגם לפי MSE הוא הכי גבוה כאשר ה-interpolation מוגדר כ-nearest. לכן ניתן להבין שלמרות שתמונה מתקבלת כמעט ללא פיקסלים המריחה, הטשטוש וחוסר הפוקוס משפיעים לרעה על ה-Loss המוגדר.

לעומת זאת, ניתן לראות שכאשר ה-interpolation מוגדרת כ-bilinear שנחשבת ליותר מתקדמת מ-nearest ניתן לראות בבירור בתמונה פיקסלים רבים כמעט כמו בקלט, אך למרות זאת התמונה כביכול יותר מפוקסת והצבעים נראים בצורה עזה יותר. כאשר מסתכלים על ה-loss גם עבור MSE וגם עבור PSNR כשה-interpolation מוגדרת כ-bilinear ניתן לראות שה-loss הוא הנמוך ביותר ביחס לאופציות האחרות של ה-interpolation מכאן אני מניח שה-loss לוקח בחשבון את החדות והשיפור של התמונה באופן איטי ביחס לכל פיקסל לעומת תמונה המתקבלת מרוחה ולא מפוקסת.

מגיש : שי ארץ קדושה – 203276258

- מודל מס' 5.1 משופר עם feature extractor של VGG16 בשילוב dilated blocks ו-3 Inputs.

בשונה מהמודל המקורי המוגדר בארכיטקטורה בסעיף ה ניסיתי לשנות את הארכיטקטורה בכך שיצרתי שלוש שכבות קונבולוציה עם גדלי קרנל שונים (8,4,2) אליהם הכנסתי את הקלט של המודל והעברתי את ה-output המתקבל בשני dilated blocks עבור כל אחד משלושת ה-inputs. בנוסף לכך ביצעתי feature extractor ממודל VGG16 וביצעתי בין ארבעת ה- outputs את פעולת concatenate. בחרתי לעשות כך את הארכיטקטורה המשופרת מתוך מחשבה שאם אעביר את הקלט כמה פעמים בגדלי קרנלים שונים ייתכן בשילוב dilated blocks תהיה לכך השפעה על חישוב הפיקסלים של התמונה.

```
def get_model_with_vgg_v2_loss_psnr():

    inp = Input(shape=(None,None,3))
    x1 = Conv2D(64,8,activation=Activation(LeakyReLU(alpha=0.2)), padding = 'same')(inp)
    x1 = dilated_conv(64)(x1)
    x1 = dilated_conv(64)(x1)

    x2 = Conv2D(64,4,activation=Activation(LeakyReLU(alpha=0.2)), padding = 'same')(inp)
    x2 = dilated_conv(64)(x2)
    x2 = dilated_conv(64)(x2)

    x3 = Conv2D(64,2,activation=Activation(LeakyReLU(alpha=0.2)), padding = 'same')(inp)
    x3 = dilated_conv(64)(x3)
    x3 = dilated_conv(64)(x3)


    y = fe_model(inp)
    x = concatenate([x1,x2,x3,y],axis=-1)

    x = UpSampling2D(interpolation='bilinear')(x)

    y1 =Conv2D(3,1, padding='same')(x)
    y1 = Activation('sigmoid',name = 'Out_Med')(y1)

    x = Conv2D(64,1)(x)

    x = UpSampling2D(interpolation='bilinear')(x)

    y2 = Conv2D(3,1,padding='same')(x)
    y2 = Activation('sigmoid',name = 'Out_Large')(y2)

    model_with_vgg_v2 = Model(inp,[y1,y2])
    model_with_vgg_v2.compile(loss=PSNR_as_loss,optimizer='adam',metrics=['acc','mse'])
    return model_with_vgg_v2
```

כעת הרצתי גם את המודל המשופר הזה שיצרתי מספר פעמים, עם קומבינציות שונות של Interpolation שניתן להגדיר בטעינה של התמונות ב-data generator (nearest, bilinear, bicubic) ו-loss שמוגדר פעם לפי MSE ופעם אחרת לפי PSNR, גם כאן סה"כ 6 קומבינציות אפשריות עבור המודל.

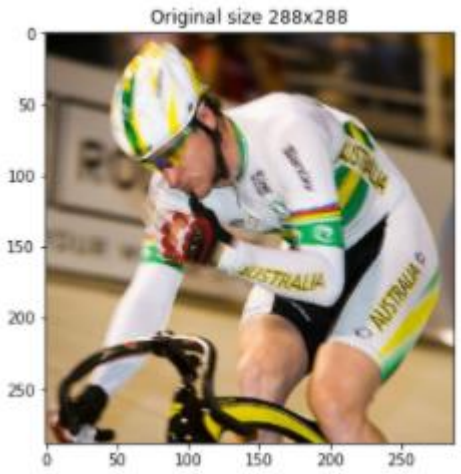
מתוך כלל התוצאות המתקבלות כתוצאה משינוי ה-interpolation וה-loss ניסיתי לערוך השוואה ולהבין מהי שיטת ה-interpolation הטובה ביותר שמוגדרת ב-data generator שיחד עם ה-loss שמוגדר בין האופציות MSE ל-PSNR תביא לתוצאות הטובות ביותר.

מגיש : שי ארץ קדושה – 203276258

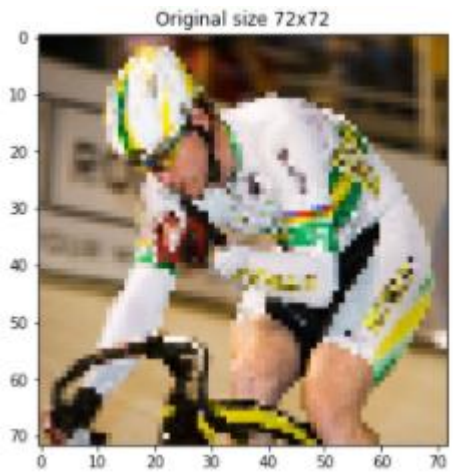
ניתוח של התוצאות המתקבלות

אנסה להציג את ההבדלים ביחס לתמונה אחת בגודל 288X288 עבור כלל הקומבינציות במודל זה.

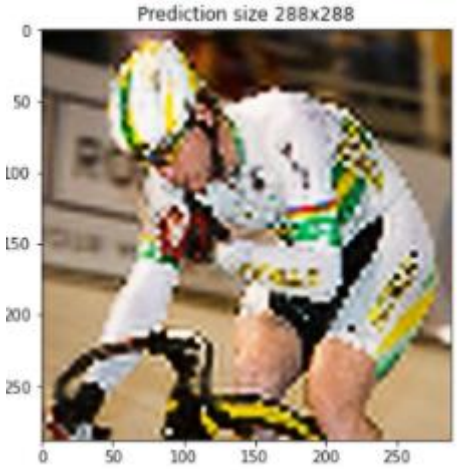
תמונה מקורית



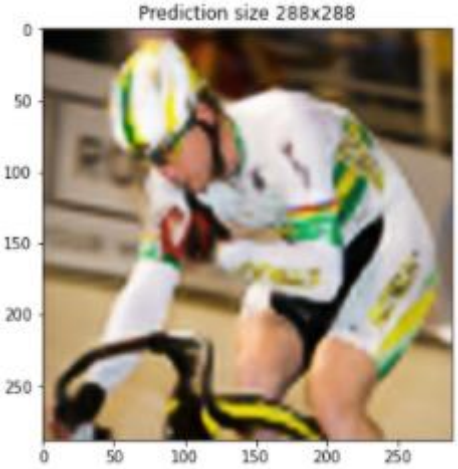
תמונת קלט לאחר ביווץ לגודל 72X72



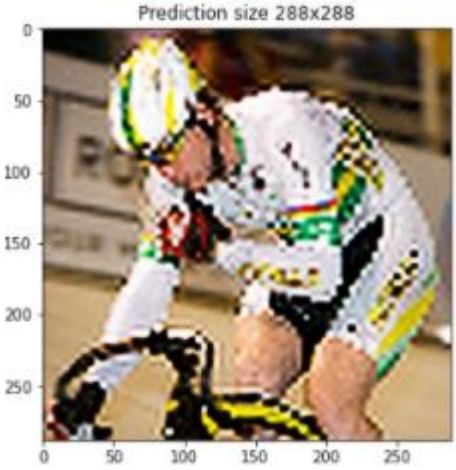
loss- PSNR ,interpolation- bicubic

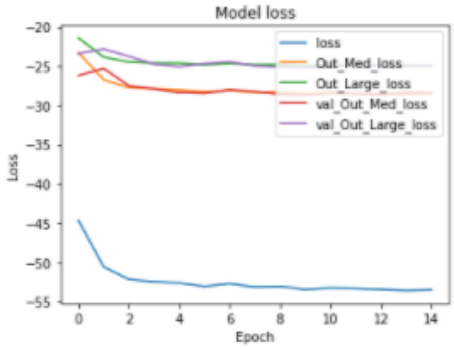


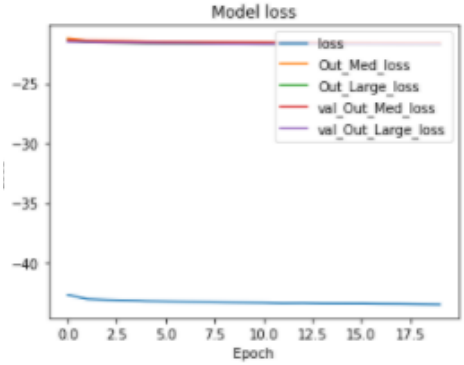
loss- PSNR ,interpolation-nearest

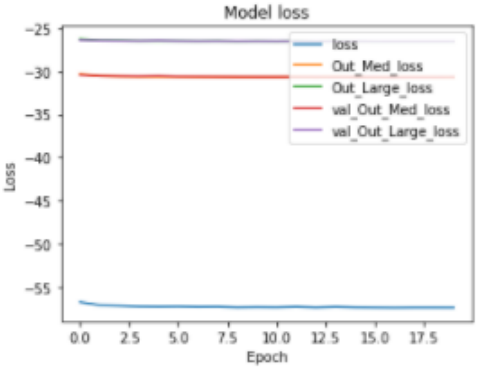


loss-PSNR ,interpolation- bilinear



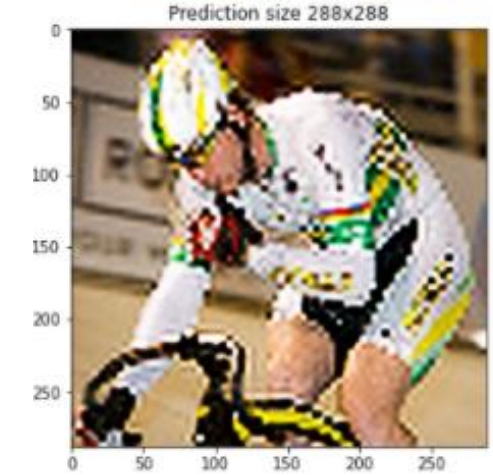


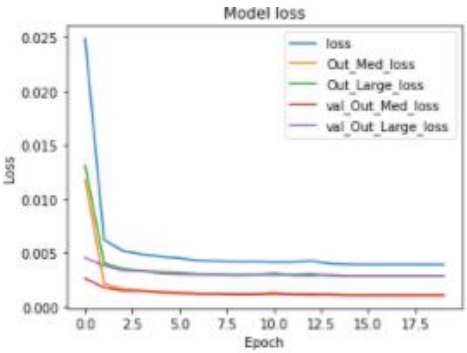




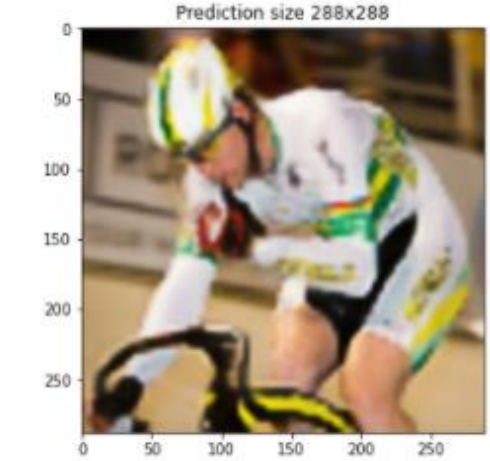
מגיש : שי ארץ קדושה – 203276258

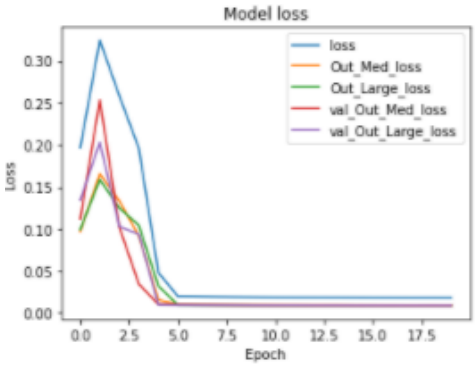
loss- MSE ,interpolation- bilinear



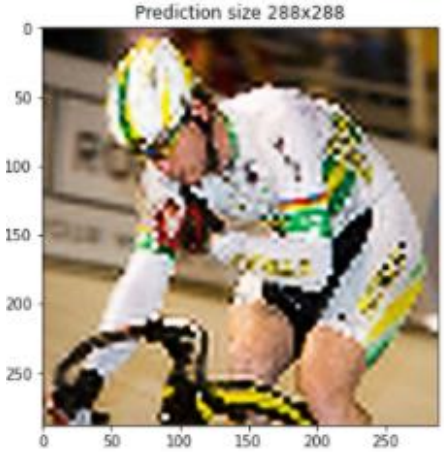


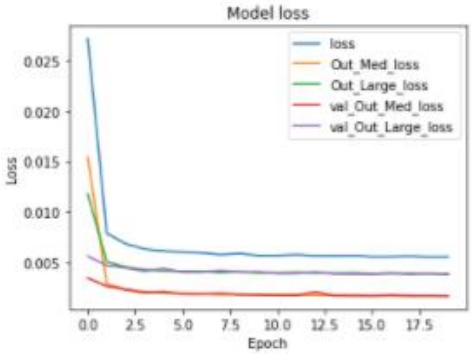
loss- MSE ,interpolation- nearest





loss- MSE ,interpolation- bicubic





ניתן לראות שגם עבור המודל המשופר הזה יש הבדל משמעותי בתוצאות בין כל אחת מהקומבינציות המוצגות אבל ביחס למודל הקודם התוצאות הן די דומות.

גם עבור המודל הזה השילוב של interpolation- nearest הביא לקבלת תמונה במעט ללא פיקסלים אך מטושטשת ולא מפוקסת שהיא למראית עין נראית טובה יותר מהתמונות האחרות. לעומת זאת במדדים גם במודל זה נראה שכאשר משתמשים ב- interpolation- nearest ה-loss מגיע לערך הגבוה ביותר גם עבור MSE וגם עבור PSNR. תוצאה זו מחזקת את ההשערה שהערכתי קודם שכאשר התמונה מרוחה ולא מפוקסת זה משפיע לרעה על קצב הירידה ב-Loss.

במודל זה גם עבור interpolation- bilinear ניתן לראות שה-Loss המתקבל הוא הנמוך ביותר גם עבור MSE וגם עבור PSNR אך התמונה גם כאן יוצאת עם פיקסלים רבים ונראה שהשינוי שנעשה בה הוא מינורי לעומת תמונת הקלט שהוכנסה למודל.

מגיש : שי ארץ קדושה – 203276258

סיכום:

לאחר סיום מימוש כלל המודלים לפי הארכיטקטורות השונות שהוגדרו בעבודה ובנוסף השיפורים והניתוחים שביצעתי לפי אפשרויות שונות של interpolations ו-loss ניתן להבין כי על אף כלל השינויים והווריאציות השונות התוצאות שמוצגות למראית לאחר כל מודל לא מראות שיפור ניכר ברזולוציה של התמונה ביחס לתמונת הקלט בכלל המודלים. אני מעריך שבעת כיווץ התמונה לגודל 72X72 התמונה מאבדת מידע רב וכך למרות שאנחנו מנסים באמצעות מודלים שונים ושיטות שונות ישנו קושי רב לשחזר את התמונה לרזולוציה המתבקשת עקב המידע שאבד בכיווץ. יחד עם זאת, אם מסתכלים היטב על תמונות שונות ניתן לראות שינויים קטנים בין מודל למודל בתמונות שהאובייקטים בהן גדולים וקרובים יותר אך אם התמונה רחוקה והפריטים קטנים קשה לראות את ההבדלים בעין.

העבודה הזו עניינה אותי מאוד ובמהלכה צברתי ידע רב וטכניקות שונות להתמודד עם בעיות שיפור רזולוציה של תמונות שונות ואת מידת ההשפעה של משתנים שונים על התוצאה המתקבלת. זהו נושא מאוד מורכב שקשה מאוד להגיע לתוצאות איכותיות ומצריך יכולת חישובית גבוהה שזהו היה הקושי העיקרי בעבודה על המחשב האישי או הקולב. ייתכן ואם המשאבים שהיו עומדים לרשותנו היו בעלי יכולת חישובית טובים יותר ייתכן והיינו מגיעים לתוצאות טובות יותר.