בס״ד

המכללה האקדמית להנדסה
אורט בראודה
ORT BRAUDE COLLEGE

פרויקט גמר (סטאז׳) מחקרי

שיפור איכות תמונות באמצעות אלגוריתמי למידה עמוקה

קדוש עמית

בהנחיית: ד״ר אדלר אמיר חתימה: נחתם דיגיטלית

מקום ביצוע ההתמחות: המכללה האקדמית להנדסה אורט בראודה

הוגש בתאריך: 13.09.2020, כ״ד באלול תש״פ

הוגש לשם מילוי חלקי של הדרישות לקבלת התואר
״בוגר במדעים B.Sc. בהנדסת חשמל ואלקטרוניקה״

תקציר:

המחקר עוסק בשיפור איכות תמונות באמצעות אלגוריתמי למידה עמוקה (Deep Learning).
המחקר יכלול בתוכו פיתוח אלגוריתמי למידה עמוקה לשיפור תמונות בדגש על הפעולות הבאות:

א. ניקוי רעשים (de-noising).
ב. סופר-רזולוציה (super-resolution).
ג. תיקון פיקסלים חסרים (salt & pepper).

מימוש האלגוריתמים יתבצע בשתי שפות תכנות: Matlab ו-Python.
ב-Matlab תבוצע העבודה הישירה עם התמונות וכן יצירת GUI נוח למשתמש לביצוע האלגוריתמים שנכתבו בעבודה.
ב-Python יבוצעו בניות ארכיטקטורות של רשתות נוירונים שונות ואימונן.

תהליך השיפור יתבצע באמצעות המרת התמונה למרחבי ייצוג יתירים, כדוגמת ייצוג DCT יתיר (Discrete Cosine Transform), ביצוע פעולות מתמטיות ע״י רשת נוירונים עמוקה במרחב ההתמרה, וביצוע התמרה הופכית בחזרה למרחב התמונה.

בחלקו הראשון של המחקר, תמומש מערכת אשר מרכזה הינו למידת פונקציות כיווץ סקרליות. לאחר מכן, המערכת תומר למערכת אשר מרכזה מתבסס על לימוד פונקציית כיווץ וקטורית. בהמשך המחקר, ביצוע ההתמרה למרחב DCT וההתמרה ההפוכה יבוצעו גם הם בתוך רשת הנוירונים העמוקה.

בדיקת תוצאות המערכת יבוצעו לאורך כל שלבי המחקר על מאגרי תמונות גדולים שקיימים באינטרנט ויבססו על הפרמטרים PSNR ו-SSIM. בנוסף לכך, יוצגו תוצאות ויזואליות להמחשת התוצאות.

כיום כבר קיימות מערכות המבצעות פעולות אלה באמצעות למידה עמוקה **אך היתרון ההנדסי המהותי של פרויקט מחקר זה הוא בעובדה שבפרויקט זה אבצע את הפעולות הנ״ל באמצעות שכבות Dense בלבד ובכך אצור פתרון פשוט מאוד חישובית לבעיות מורכבות אלה. פשטות הרשת תאפשר מימוש של המערכת על רכיבים בעלי פונקציות חישוב בסיסיות ביותר (חיבור וכפל, ואפשר להגיד גם רק חיבור).**

תודות:

ברצוני להודות לד״ר אדלר אמיר, מנחה הפרויקט, על ההכוונה והתמיכה בתכנון וביצוע הפרויקט. הדרכתו הייתה מקצועית ומעשירה לאורך כל הדרך ולמדתי ממנו המון גם בעת הפרויקט וגם בקורס למידה עמוקה.

תודה נוספת למר משה שדה, מרכז הפרויקטים, על התייעצות והכוונה בתחילת המחקר.

תוכן עניינים:

רשימת טבלאות:

רשימת איורים:

<u>רשימת קיצורים:</u>

Adam – Adaptive Moment
DC - Direct current
DCT - Discrete Cosine Transform
GB – Gigabyte
GUI - Graphical User Interface
IDCT – Inverse Discrete Cosine Transform
NN – Neural Network
PSNR - Peak Signal-to-Noise Ratio
RAM - Random Access Memory
SGD - Saccharomyces Genome Database
SR – Super Resolution
SSIM - Structural Similarity Index Measure
std - Standard Deviation

1. מבוא

שיפור איכות תמונות הינו תחום מחקר פעיל ומרכזי עם יישומים אזרחיים וצבאיים, בין היתר עבור מצלמות דיגיטליות, פלאפונים חכמים, מערכות ראיית לילה, הדמיה רפואית, מכ"מים ועוד. נדון בשלושה מנגנוני קלקול נפוצים בתהליך רכישת תמונה ותיקונם:

ניקוי רעשים: עקב מנגנונים שונים, כמו למשל המגבלות הקוונטיות של תהליך הצילום האלקטרוני, העברת התמונה דרך מערכת תקשורת רועשת, וגם הקוונטיזציה של רמות האפור, אין אנו מקבלים את רמות האפור הנכונות והמדויקות בכל מקום. המודל המקובל לאפיון תהליכים אלו הוא תוספת של אותו טפילי, המכונה באופן מסורתי רעש, לאות התמונה. שימוש בלמידה עמוקה לפתרון בעיה זו צפוי להביא לתוצאות טובות בהרבה מהאלגוריתמים הקיימים שלא משתמשים ברשתות נוירונים.

סופר רזולוציה: תחום הסופר רזולוציה נועד על מנת לאפשר לנו לעשות "זום" על תמונות מבלי לאבד מאיכות התמונה. אם תמונה צולמה באיכות גבוהה אפשר לעשות לה "זום" בקלות, אבל במקרה הסטנדרטי, למשל כאשר מסתכלים על מצלמות אבטחה, לא ניתן פשוט לעשות "זום" ולקבל תמונה באיכות טובה. תחום הסופר רזולוציה אשר מבוסס על רשתות נוירונים נועד על מנת לפתור בעיה זו.

המנגנון השלישי הינו שילוב בין שני המנגנונים הראשונים והוא ניקוי רעשים וסופר רזולוציה יחד, נבחן כמה שיטות ונבדוק איזו שיטה מניבה את התוצאות הטובות ביותר.

2. תיאור המערכת

2.1 מפרט פונקציונלי

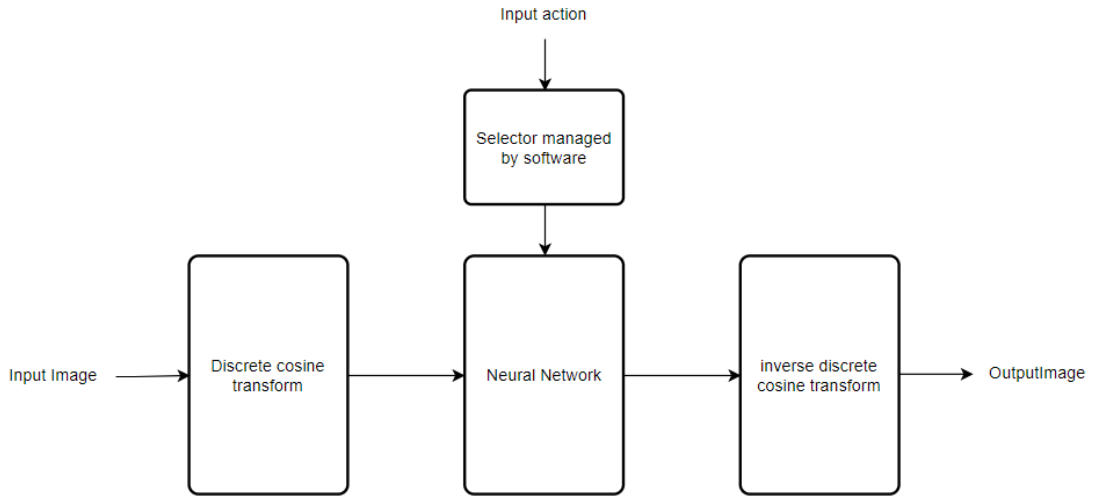
המערכת מבצעת שיפור תמונות ומטפלת בשלוש בעיות שונות בתחום עיבוד התמונה:
ניקוי רעשים, סופר רזולוציה וניקוי רעשים וסופר רזולוציה יחד.
זמן פעולת המערכת בכל תחום הינו שניות בודדות והפעולות לא מצריכות חישובים מסובכים.

יכולות המערכת:

א. ניקוי רעשים מתמונה בגווני אפור תוך שימוש ב-81 רשתות נוירונים סקלריות.
ב. ניקוי רעשים מתמונה בגווני אפור במהירות תוך שימוש ברשת וקטורית אחת.
ניקוי הרעשים מתבצע ברמות שונות של סטיית תקן: 1,2,5,10,15,20,25.
ג. סופר רזולוציה ביחסי הגדלה של 2 ו-3 לתמונות צבעוניות (וכן תמונות בגווני אפור) בגודל 510X510.
ד. מערכת לניקוי רעשים בשילוב סופר רזולוציה. מערכת זו פועלת על רעשים בסטיית תקן של 5 או 10 ועם יחס הגדלה של 2.

2.2 מפרט טכני

תרשים מלבנים של המערכת



איור 1 : תיאור תרשים מלבנים כללי של המערכת

הערה : בחלק של הסופר רזולוציה שני המכלולים האחראים על ביצוע ההתמרות יוכנסו לתוך מכלול ה-Neural Network.

טבלת פירוט מכלולי המערכת:

מספר סעיף	מכלול המערכת	תפקיד המכלול
1	Discrete cosine transform	המכלול מקבל בכניסתו תמונה צבעונית (או בגווני אפור) בהתאם לאלגוריתם, ומוציא במוצאו את התמרת ה-DCT של התמונה. מוצא המערכת הינו 81 "תמונות" עבור 81 פסי תדר שונים של התמונה בכניסה.
2	Selector managed by software	המכלול מקבל בכניסתו מהמשתמש את הפעולה הרצויה לביצוע והפרמטרים הרלוונטיים ומוציא במוצאו הוראה באיזה רשת נוירונים להשתמש.
3	Neural Network	תפקיד הרשת היא לבצע את הפעולה הרצויה באמצעות אלגוריתמי Deep Learning שונים המבוססים על פונקציית כיווץ וקטורית ומבנה רב שכבתי המורכב משכבות Dense.
4	מערכת התמרה הפוכה למרחב התמונה	המכלול מקבל בכניסתו 81 "תמונות" במרחב ה-DCT מתוקנות לאחר מעבר דרך הרשת, ומוציא במוצאו את התמונה הסופית במרחב התמונה לאחר ביצוע הפעולה הרצויה.

טבלה 1 : טבלת פירוט מכלולי המערכת

עקרון פעולת המערכת

בעת הפעלת המערכת המשתמש יוכל לבחור תמונה מהמחשב באמצעות סייר קבצים.
במקביל יוצג למשתמש תפריט שבו הוא יוכל לבחור את הפעולה הרצויה מהפעולות הבאות:

- ניקוי רעש בסטיית תקן של 1 (תמונות בגווני אפור).
- ניקוי רעש בסטיית תקן של 2 (תמונות בגווני אפור).
- ניקוי רשע בסטיית תקן של 5 (תמונות בגווני אפור).
- ניקוי רעש בסטיית תקן של 10 (תמונות בגווני אפור).
- ניקוי רעש בסטיית תקן של 15 (תמונות בגווני אפור).
- ניקוי רעש בסטיית תקן של 20 (תמונות בגווני אפור).
- ניקוי רעש בסטיית תקן של 25 (תמונות בגווני אפור).
- סופר רזולוציה ביחס הגדלה של 2 לרזולוציה מלאה של 510X510 (כולל תמונות צבעוניות).
- סופר רזולוציה ביחס הגדלה של 3 לרזולוציה מלאה של 510X510 (כולל תמונות צבעוניות).
- ניקוי רעש בסטיית תקן של 5 בשילוב עם סופר רזולוציה ביחס הגדלה של 2 לרזולוציה מלאה של 510X510 (כולל תמונות צבעוניות).
- ניקוי רעש בסטיית תקן של 10 בשילוב עם סופר רזולוציה ביחס הגדלה של 2 לרזולוציה מלאה של 510X510 (כולל תמונות צבעוניות).

בשלב זה המערכת תיכנס לפעולה תבצע את התיקון הדרוש לתמונה בכניסה ותציג למסך את התמונה ביציאה מהמערכת.
כל פעולת המערכת מבוצעת מאחורי הקלעים: הרשתות אומנו ב-Python במשך ימים שלמים לקבלת המשקלים והמקדמים המתאימים לפעולת כל מערכת.
כמובן שהרשתות בממשק זה הן הרשתות שהניבו את התוצאות הטובות ביותר.

3. מטלות הנדסיות:

3.1 מטלות הנדסיות ברמת מפרט הדרישות

- למידת נושא התמרת DCT דו ממדית (והתמרת DCT דו ממדית הפוכה) לצורך התמרת התמונה למרחב ייצוג DCT.
- לימוד אלגוריתם קיים מבוסס פונקציית כיווץ סקלרית לניקוי רעשים מתמונה בגווני אפור וגודל לא קבוע.
- מימוש אלגוריתם מבוסס פונקציית כיווץ סקלרית תוך אימון 81 רשתות נוירונים שונות עבור כל פס תדר.
- לימוד ומימוש אלגוריתם חדש מבוסס פונקציית כיווץ וקטורית לניקוי רעשים מתמונה על מנת לאפשר פעולת מערכת בשניות בודדות וללא צורך בכוח חישובי רב.
- אימון רשתות וקטוריות שונות ב-Python ובחינת תוצאותיהן לקבלת הרשת עם התוצאות הטובות ביותר.
- בדיקת השפעה של נרמול הערכים בכניסה לרשת על תוצאות הרשת.
- מילוי טבלת תוצאות עבור הרשת הטובה ביותר בבחינה על סטיות תקן שונות.
- מעבר לביצוע תהליך Super Resolution ביחס הגדלה של 2, כאשר כעת ההתמרות בין המרחבים יבוצעו כולן בתוך ה-Neural Network.
- למידה וביצוע של הקפאת שכבות מהרשת בסביבת Keras על מנת לקבע את המסננים שבעזרתם מבוצעת ההתמרה.
- אימון רשתות שונות למציאת הרשת הטובה ביותר עבור Super Resolution.
- אימון מערכת הסופר רזולוציה על יחס הגדלה נוסף של 3.
- בניית מערכת אחת לביצוע שתי פעולות יחד (ניקוי רעשים וסופר רזולוציה), והשוואת המערכת לביצוע של ניקוי רעשים ורק אחר כך ברשת אחרת סופר רזולוציה.
- יצירת ממשק משתמש ב-Matlab לטעינת תמונה, בחירת פעולה לביצוע ופרמטרים מתאימים וקבלת התמונה ביציאה.
- כתיבת ספר פרויקט והכנת מצגת.

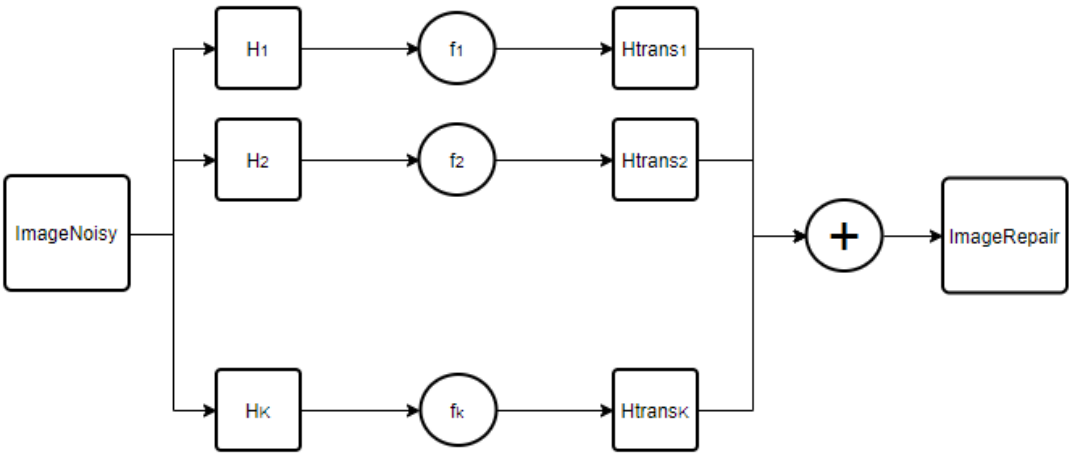
3.2 ביצוע המטלות:

אופן ביצוע המטלות יתואר לפי הסדר, כאשר בכל חלק יופיע תכן מפורט, ביצוע בדיקות וסימולציות ובחינת התוצאות.
מכיוון שהמחקר כולל בתוכו מערכות רבות, נבצע את התכן והבדיקות לפי פרקים עבור כל מערכת בנפרד.

3.2.1 ניקוי רעשים (Denoising) - מערכת מבוססת רשתות סקלריות לא מנורמלות

הצגת הרעיון הכללי

את ניקוי הרעשים נבצע תחילה ע"י אלגוריתם מבוסס פונקציית כיווץ סקלרית.
לשם כך עלינו לתכנן רשת סקלרית שתלמד את פונקציית הכיווץ, כלומר רשת שמקבלת במבואה סקלר ומוציאה במוצאה גם כן סקלר.



איור 2 : תרשים מלבנים של מערכת ניקוי רעשים מבוססת רשתות סקלריות לא מנורמלות

בכניסה למערכת מוכנסת תמונה רועשת ברעש גאוסי עם סטיית תקן כלשהי, במוצא המערכת תתקבל תמונה נקייה מרעשים (בהתאם ליכולות המערכת).
הרעיון מסתמך על התמרת DCT עם פונקציות בסיס בגודל 9x9.
תחילה, נמיר את התמונה הרועשת ל-81 "תמונות" בפסי תדר שונים, פעולה זו מתבצעת ע"י כך שהתמונה עוברת קונבולוציה עם 81 מסננים שונים ובכל פס תדר מתקבלת "תמונה במרחב DCT".
לאחר מכן נתקן את ה"תמונות" בכל פסי התדר באמצעות רשתות נוירונים עמוקות המורכבות משכבות Dense בלבד. עבור כל פס תדר נאמן רשת אחרת שתלמד מקדמים שונים. מטרת הרשתות היא למידת פונקציות הכיווץ הדרושות לסינון הרעשים בכל פס תדר.
לאחר התיקון במישור ה-DCT נתמיר בחזרה את ה"תמונות" בכל פסי התדר למרחב התמונה וע"י חיבור התמונות המתקבלות בכל פסי התדר נקבל את התמונה ביציאה במטרה שהיא תהיה נקיה מרעשים.

הערה : נדגיש כי הרשת הינה רשת סקלרית (מקבלת סקלר ומוציאה סקלר). למרות שבתרשים המלבנים הדבר נראה כאילו אנו מכניסים לרשת "תמונה במישור התדר", זהו לא המצב. מתבצע שיטוח של התמונה לווקטור ומכניסים פיקסל אחרי פיקסל.

יצירת ה-Datasets עבור כל הרשתות

כאמור מבוצע אימון של 81 רשתות שונות ולכן ניצור 81 Datasets שונים - Dataset עבור כל פס תדר. את יציאת ה-Datasets לאימון הרשתות ניצור באמצעות Matlab. הקוד המלא מצורף לפרויקט. כאן נתאר את מבנה הקוד:

נקרא תמונה מתוך תמונות ה-Dataset שהורדנו מהאינטרנט (אם התמונה צבעונית נדאג להמירה לגווני אפור).

נרעיש את התמונה ע"י הוספת רעש גאוסי עם תוחלת 0 וסטיית תקן של sigma אותה הגדרנו ל-25. נשים לב שלאחר תוספת הרעש התקבלו ערכים גדולים מ-255 ולכן נמיר אותם ל-255 ובנוסף התקבלו ערכים קטנים מ-0 ולכן אותם נמיר ל-0. בנוסף הערכים המתקבלים לא שלמים ומכיוון שאנו רוצים לדמות תמונה אמיתית עם רעש, נעגל את כל הערכים בעזרת uint8.

עבור כל פס תדר:

יצירת ה-data:

נבצע קונבולוציה של התמונה הרועשת עם המסנן הנוכחי. את התמונה במוצא נשטח לווקטור עמודה – זה יהיה ה-data המתקבל מהתמונה הנוכחית. את הוקטור המתקבל נשרשר ל-data של המסנן הנוכחי.

יצירת ה-label:

נבצע קונבולוציה של התמונה הנקייה מרעשים עם המסנן הנוכחי. את התמונה במוצא נשטח לווקטור עמודה – זה יהיה ה-label המתקבל מהתמונה הנוכחית. את הוקטור המתקבל נשרשר ל-label של המסנן הנוכחי.

לבסוף עבור כל מסנן נקבל cell בגודל 1X2 שבעמודה הראשונה שלו מופיע וקטור עמודה גדול של data ובעמודה השנייה שלו מופיע וקטור עמודה גדול של label. כל cell כזה נשמור בשם מתאים בהתאם לכל מסנן. נדגיש שכל איבר בוקטור ה-data שיצרנו הוא כניסה לרשת ולכן ממעט תמונות מקבלים data עצום.

כעת, עם ה- Datasets שיצרנו נאמן רשתות סקלריות בעלות ארכיטקטורות שונות על מנת למצוא את הרשת שמספקת את התוצאות הטובות ביותר.

הערה – הערכת זמן גסה: בשיטה זו אנו חייבים לאמן 81 רשתות, אימון כל רשת לוקח בערך שעתיים וחצי (במחשב שברשותנו) ולכן אימון שלם של כל המערכת על מחשב אחד לוקח בערך 8 ימים. לכן, נאמן את הרשתות בכמה מחשבים במקביל, כאשר בכל מחשב אנו נאמן רשתות שונות. בעיה זו, של האימון הממושך והמפוצל תיפתר בהמשך כשנעבור לרשת וקטורית.

אימון הרשתות ב-Python ושמירת המודלים

את אימון 81 הרשתות נבצע ב-Python. הקוד המלא מצורף לפרויקט. נתאר את מבנה הקוד והחלקים המרכזים בו:

תחילה טוענים את ה-modules הדרושים לקוד ביניהן Keras ליצירת מבנה הרשת. מכיוון שנדרש לאמן 81 רשתות אנו עוברים בלולאה על 81 ה-Datasets שיצרנו עבור 81 פסי התדר השונים.

טוענים את ה-data וה-label הרלוונטיים לאותו מסנן. נשים לב שאנו משתמשים ב-numpy.array ולא ב-list. הסיבה היא המיקום בזיכרון. בהמשך נשתמש ברשימה של רשימות אשר תופסת הרבה יותר מקום בזיכרון מאשר מערך דו ממדי של numpy (זאת מכיוון שכל list שאנו יוצרים תופסת מקום נוסף בזיכרון שנועד למימוש תכונות ה-list).

ארכיטקטורת הרשת:

```
model = Sequential();
model.add(Dense(256, input_shape=(1,), activation='sigmoid'));
model.add(Dense(256, activation='sigmoid'));
model.add(Dense(256, activation='sigmoid'));
model.add(Dense(256, activation='sigmoid'));
model.add(Dense(1));
model.compile(optimizer='adam', loss='mse')
print(model.summary())
numOfEpochs = 5;
trained_model = model.fit(trainData, trainLabel, validation_data=(validationData,
validationLabel), epochs=numOfEpochs)
```

איור 3: יצירת ארכיטקטורת רשת מבוססת פונקציית כיווץ סקלרית ואימונה ב-Keras

כאמור, לעת עתה אנו עובדים ברשת סקלרית, כלומר הרשת מקבלת במבואה סקלר לכן input_shape=(1,), בנוסף רשת סקלרית מוציאה במוצאה גם כן סקלר ולכן הוספנו בסוף שכבה עם פרספקטרון אחד.

השתמשנו בתור הרשת הראשונה לאימון בשתי שכבות Dense של 256 פרספקטרונים כל אחת (בתמונה לעיל מוצגת הרשת עם ארבע שכבות Dense של 256).

נשתמש ב-optimizer של adam ונאמן את הרשת על 5 Epochs (בדיעבד, אימנו תחילה על יותר Epochs אך ראינו שאין שינוי ברשת אחרי בערך 5 Epochs. לכן זו כמות האפוקים שנבחרה).

אנו מעוניינים כרגע להציג את ה-loss של ה-Train ושל ה-Validation באחוזים, ולכן נחשב אותם מתוך האנרגיה של ה-Train וה-Validation בהתאמה.

את הגרף נשמור לקובץ בשם plotResults<numFilter>.pdf יחד עם מספר המסנן וכן את המודל עם המקדמים נשמור לקובץ model<numFilter>.h5 יחד עם מספר המסנן.

בחינת התוצאות

את התוצאות אנו בוחנים על 6 תמונות, אלו הן 6 התמונות שבחנו עליהן במאמר שאליו נשווה את התוצאות שלנו [1]. תחילה נציג את התוצאות על פי PSNR, לבסוף נציג גם תמונות של התוצאות עבור הרשת הטובה ביותר שנמצא מבין הרשתות שנבחן.
ראשית, נרעיש את התמונה בדיוק באותו אופן שהוסבר לעיל.
לאחר מכן נעבור על כל המסננים ועבור על מסנן נבצע את הפעולות הבאות:

- נבצע קונבולוציה של התמונה בכניסה עם המסנן הנוכחי.
- נשטח את התוצאה לקבלת וקטור.
- את התוצאה נכניס לרשת אך במקום להשתמש בפעולת activation החלטנו לממש בעצמנו את פעולת הרשת ע״י קריאה ערכי ה-Weights וה-Bias של כל שכבה. **כאן נכנס היתרון המרכזי של המחקר (שהוסבר בתקציר) שנגזר מהעובדה שהרשת שמכילה רק שכבות Dense – קלות המימוש.**
- את הוקטור המתקבל במוצא הרשת נמיר חזרה לממדי התמונה.
- נוסיף את התוצאה לתוך ImageRepair – התמונה המשוחזרת (נקיה מרעשים עד כמה שהצלחנו).

בחינת התוצאות תבוצע בהתאם השלבים הבאים:

- הורדת קצוות שנוספו ל-ImageRepair בעקבות פעולת הקונבולוציה.
- חישוב PSNR ו-SSIM על מנת להשוות בין התמונה שהתקבלה לתמונה המקורית.
- נציג ב-figure אחד את שלושת התמונות: התמונה המקורית, התמונה הרועשת והתמונה ביציאה מהרשת. בנוסף נשמור את כל ה-figures.
- את ערכי PSNR ו-SSIM נשמור בקבצי mat בשמות PSNR ו-SSIM בהתאמה.

באותו אופן נבחן תוצאות של שתי רשתות נוספות שיתוארו בטבלה להלן:

	תוצאות מאמר (להשוואה)	adam, 2layers, 256-256, sigmoid	adam, 4layers, 256-256-256-256, sigmoid	SGD, 4layers 256-256-256-256, sigmoid
barbara	29.09	24.60	26.89	13.11
boat	29.11	26.87	28.20	13.18
fingerprint	27.15	25.48	26.94	12.43
house	30.95	27.21	29.24	12.49
lena	31.02	28.16	29.24	13.39
peppers256	29.04	26.43	27.89	13.01
הפרש ממוצע מתוצאות מאמר [1]		-2.94	-1.33	-16.46

טבלה 2: תוצאות מערכת מבוססת רשתות סקלריות לא מנורמלות

מסקנות ממערכת מבוססת רשתות סקלריות לא מנורמלות

ברשת הראשונה, כאשר השתמשנו רק בשתי שכבות של 256, התוצאות שקיבלנו היו רחוקות מאוד מתוצאות המאמר (הפרש ממוצע של 2.94dB לעומת תוצאות המאמר). לכן ניסינו להוסיף לרשת שתי שכבות של 256 פרספקטרונים כל אחת, על מנת שנוכל לראות את ההשפעה של הוספת שכבות. כפי שרואים בטבלה לעיל התוצאות השתפרו בצורה משמעותית בעזרת הוספת שתי שכבות נוספות.

בנוסף בחנו את ההשפעה של שינוי optimizer : שינינו מ-adam ל-SGD. התוצאות שקיבלנו משינוי זה היו גרועות מאוד וגם התמונות ביציאה התקבלו בצורה מעוותת. לכן חשוב לשים לב שההשפעה של ה-optimizer גם היא, גדולה מאוד.

3.2.2 ניקוי רעשים (Denoising) - מערכת מבוססת רשתות סקלריות מנורמלות

הצגת רעיון המעבר לרשתות מנורמלות

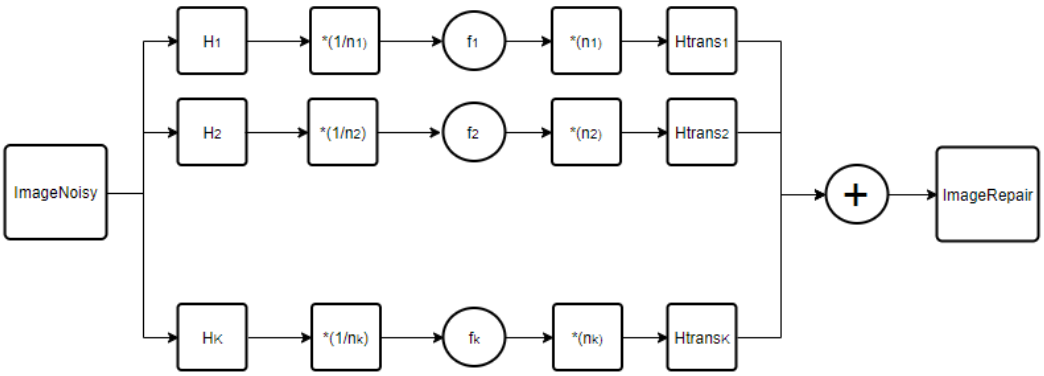
עד עכשיו, בכל הרשתות שבחנו, הכניסה לרשת לא הייתה מנורמלת. נרצה לבחון מה ההשפעה של נרמול הערכים בכניסה לכל רשת. האם נרמול הערכים בכניסה לטווח של [-1,1] ישפר את התוצאות או שמא ההפך.
לשם כך, לכל פס תדר אנו מוסיפים מקדם נרמול. זהו בעצם מקדם שנחלק בו לפני הכניסה לרשת ונכפיל בו ביציאה ממנה. כך בעצם בכניסה לרשת תהיה בטווח ערכים [-1,1] כפי שאנו רוצים.
עבור על פס תדר (עבור כל רשת) קיים מקדם נרמול שונה.

ביצוע השינויים הדרושים למעבר

ב-createDataSet.m נמצא את הערך המקסימלי בערך מוחלט מתוך ה-data וה-label של כל פס תדר. נשמור את כל הערכים האלה בקובץ בשם normalized.mat.
את כל ה-data וה-label של כל מסנן נחלק במקדם זה, כך הנתונים של הכניסה והיציאה מהרשת איתם הרשת מתאמנת יהיו בטווח ערכים [-1,1].

את RepairImage.m נפצל לשתי אופציות המתוארות להלן:
אופציה א - RepairImageWithH1.m:
פשוט נחלק במקדם הנרמול של כל פס תדר לפני הכניסה לרשת ונכפול בו ביציאה מהרשת.

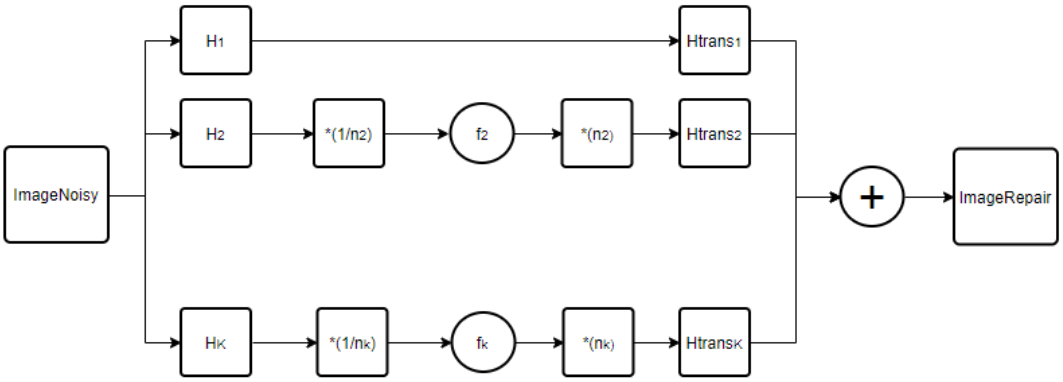
תרשים בלוקים של המערכת עם הנרמול (עם המסנן הראשון):



איור 4: תרשים מלבנים של מערכת לניקוי רעשים המבוססת על רשתות סקלריות מנורמלות (הנרמול גם במסנן הראשון)

המסנן הראשון הוא פס תדר ה-DC, זהו מסנן שמבצע ממוצע. הממוצע של הרעש הוא בעצם התוחלת, כלומר 0. לכן תיאורטית פס התדר הראשון כלל לא צריך תיקון של הרשת.
אנו רוצים לבדוק טענה זו ולראות האם לדבר זה יש השפעה ממשית על התוצאות.

תרשים בלוקים של המערכת עם הנרמול (ללא תיקון למסנן הראשון) :



איור 5 : תרשים מלבנים של מערכת לניקוי רעשים המבוססת על רשתות סקלריות מנורמלות (ללא נרמול במסנן הראשון)

בחינת התוצאות

Normalized, **WithoutH1**, adam, 4layers, 256-256-256-256, sigmoid	Normalized, **WithH1**, adam, 4layers, 256-256-256-256, sigmoid	תוצאות מאמר (להשוואה)	
27.62	27.65	29.09	barbara
28.27	28.27	29.11	boat
27.01	27.00	27.15	fingerprint
29.22	29.15	30.95	house
29.37	29.34	31.02	lena
28.06	27.99	29.04	peppers256
-1.14	-1.16		הפרש ממוצע מתוצאות המאמר [1]

טבלה 3 : תוצאות מערכת מבוססת רשתות סקלריות מנורמלות

מסקנות ממערכת מבוססת רשתות סקלריות מנורמלות

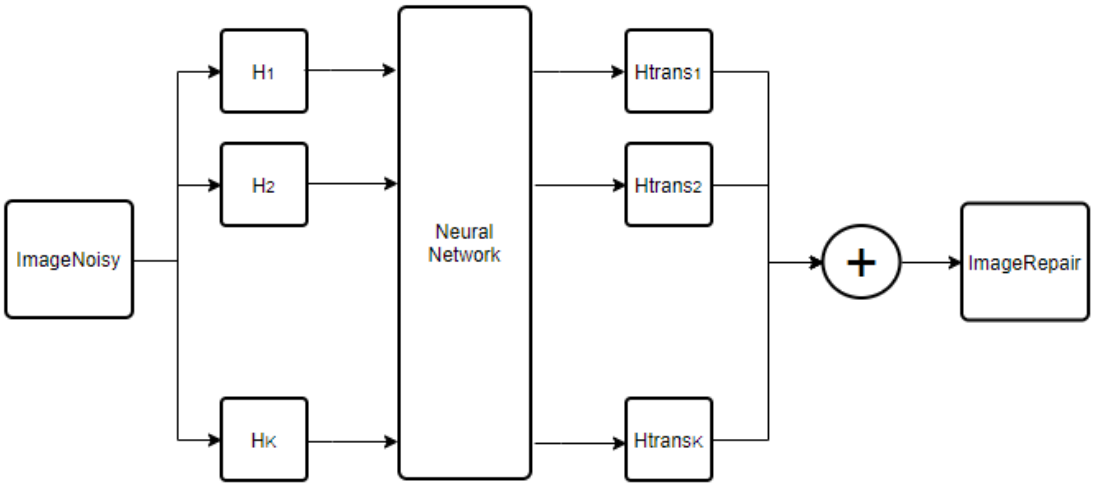
ראשית הנרמול ברשת הסקלרית אכן שיפר מעט את התוצאות (כמובן שעוד נבדוק את זה ברשת וקטורית שתתואר בהמשך).

הורדת התיקון במסנן H1 אכן סיפק תוצאות שמתיישבות עם התיאוריה שהצגנו.
הרשת לתיקון H1 לא משפיעה. אין משמעות להבדל המינורי של 0.02db.

3.2.3 ניקוי רעשים (Denoising) - מערכת מבוססת רשת וקטורית

הצגת הרעיון הכללי

בדיוק כמו שביצענו ברשת הסקלרית, גם ברשת הוקטורית נציג את הרעיון הכללי, את החלקים המרכזיים בתוכנות למימוש המערכת ולאחר מכן נסכם את התוצאות (כל זאת נעשה על סטיית התקן הגבוהה ביותר שנבדוק בפרויקט זה שהיא כמו קודם 25).
לאחר מכן, נבחר את הרשת שמספקת את התוצאות הטובות ביותר עבור std=25, ונבחן את התוצאות שלה עבור סטיות תקן נוספות ונשווה למאמר.
בחלק זה נציג את התוצאות גם בצורה ויזואלית.



איור 6 : תרשים מלבנים של מערכת לניקוי רעשים המבוססת על רשת וקטורית (לא מנורמלת)

רשת וקטורית כשמה כן היא – רשת שמקבלת בכניסתה וקטור ומוציאה במוצאה גם כן וקטור.
הרעיון הוא להחליף את 81 הרשתות הסקלריות לרשת אחת וקטורית.

המעבר לרשת וקטורית מאפשר אימון של רשת אחת בלבד וכן בשלב בחינת התוצאות אנו משתמשים כמובן ברשת אחת בלבד. יש לכך יתרונות גדולים מאוד מבחינת הזמן :

- נדרש אימון רק לרשת אחת, לצורך השוואה מדובר בסדר גודל של 3 שעות לעומת 8 ימים למערכת.
- בחינת התוצאות מתבצעת במהירות. מכיוון שברשת סקלרית היה צורך במעבר ב-81 רשתות הדבר היה דורש לא מעט זמן (בערך 5 דקות לתמונה), לעומת זאת ברשת הוקטורית מדובר בכמה שניות לתמונה שזה גם מה שרצינו מלכתחילה.
- בזכות הזמן הנמוך להרצת תמונה במערכת שיטה זו אכן רלוונטית ליישומים שמשתמשים בניקוי רעש מתמונות.
- מכיוון שלוקח רק כמה שניות להרצת תמונה במערכת נוכל לבצע כמה הרצות ולראות את ערכי ה-PSNR וה-SSIM הממוצעים ולא רק תוצאות מריצה אחת של המערכת.

התוצאות בטבלאות מטה הן תוצאות ממוצעות על 10 ריצות של המערכת.

יצירת ה-Dataset

למשתנה שישמור את ה-data וה-label שאנו יוצרים למערכת אנו קוראים DataAndLabel.

ה-DataAndLabel שאנו רוצים כעת שונה במבנהו מה-DataAndLabel שיצרנו ברשת הסקלרית. הסיבה היא שכעת יש רשת אחת והיא מקבלת וקטור בגודל 81 שבו כל איבר מגיע ממסנן אחר.

נתבונן על "הטבלה" שיצרנו ברשת הסקלרית כאשר יצרנו את 81 קבצי ה-mat:

	data	label
פס תדר 1	mX1	mX1
פס תדר 2	mX1	mX1
...	...	...
פס תדר 81	mX1	mX1

הערה: כל שורה בטבלה מייצגת קובץ mat שיצרנו, כאשר m זהו מספר הדוגמאות שקיבלנו מכל התמונות ב-Dataset.

ברשת הוקטורית מבנה זה לא מתאים מכיוון שבמבנה זה אין וקטורים של 81X1 כפי שאנו צריכים. לכן נבצע המרה של המבנה לעיל למבנה הבא:

data	label
81X1	81X1
81X1	81X1
...	...
81X1	81X1

הערה: כאשר כמות השורות במבנה זה היא כמות הדוגמאות.

נסביר ע"י שימוש במספרים אמיתיים מה-Dataset שלנו:

ברשת הסקלרית יצרנו טבלה עם 81 שורות שבכל שורה יש 2 עמודות שבכל אחת מהן ישנו וקטור עמודה בגודל 12,438,400. כעת אנו מבצעים המרה שתשמור על כל הנתונים אך בסדר אחר:

אנו רוצים טבלה עם 12,438,400 שורות שבכל שורה 2 עמודות שבכל שורה מהן ישנו וקטור עמודה בגודל 81.

כלומר כל שורה במבנה החדש תהווה דוגמה לרשת הוקטורית שאנו מאמנים.

עד כאן זה היה בתאוריה, אך במציאות הטבלאות לעיל גדולות מאוד ודורשות המון זיכרון, לכן גם המעבר עליהן דורש המון זיכרון (ולעיתים אף גורם לקריסת התוכנית).

נפתור בעיה זו ע"י אלגוריתם שלוקח בכל פעם 5 תמונות מה-Dataset ומבצע עליו את הפעולות שהוסברו.
בסופו של דבר נקבל כמה קבצים קטנים יותר (בערך 1.5GB כל אחד) במקום קובץ אחד גדול שגם אם היינו מחכים הרבה זמן ויוצרים אותו הייתה נוצרת בעיה בקריאתו ב-Python עקב גודלו.

אימון רשתות שונות ב-Python

נטען את הנתונים בקבצי ה-mat שיצרנו לתוך ארבעה מערכי numpy מסוג float32.
באותו אופן כמו ברשת הסקלרית 80% מהנתונים משמשים עבור ה-Train ו-20% נוספים משמשים עבור ה-Validation.

בבניית ארכיטקטורת הרשת נשים לב שכעת משום שהרשת וקטורית אנו משנים את ה-input_shape ל-81 וכן את שכבת ה-dense האחרונה נשנה גם כן ל-81 פרספקטרונים.

```
model = Sequential()
model.add(Dense(256, input_shape=(81,), activation='sigmoid'));
model.add(Dense(256, activation='sigmoid'));
model.add(Dense(256, activation='sigmoid'));
model.add(Dense(256, activation='sigmoid'));
model.add(Dense(81));
model.compile(optimizer='adam', loss='mse')
print(model.summary())
numOfEpochs = 5;|
trained_model = model.fit(trainData, trainLabel, validation_data=(validationData,
validationLabel), epochs=numOfEpochs)
```

איור 7: יצירת ארכיטקטורת רשת לניקוי רעשים מבוססת פונקציית כיווץ וקטורית ואימונה ב-Keras

בחינת התוצאות

נקודות חשובות על השינויים במעבר לרשת הוקטורית ב-RepairImage.m:
כאמור הכניסה לרשת היא וקטור שכל איבר בו מגיע מקונבולוציה עם מסנן אחר. על מנת לממש זאת, יצרנו inputMatrix – מטריצה שבה 81 שורות (שורה עבור כל מסנן) ובכל שורה מופיעה "התמונה בתדר" לאחר קונבולוציה עם המסנן המתאים ושיטוח לוקטור שורה.

כאמור את ה-PSNR וה-SSIM אנחנו מחשבים בעזרת ממוצע של 10 ריצות, ולכן נוספה לולאה חיצונית עם המשתנה iteration.

ה-PSNR וה-SSIM נשמרים עם תוצאות כל הריצות ובעמודה האחרונה (ה-11) מופיעות התוצאות הממוצעות של כל הריצות יחד.

3.2.4 ניקוי רעשים (Denoising) - מערכת מבוססת רשת וקטורית מנורמלת

הצגת רעיון המעבר לרשת מנורמלת

המעבר מרשת וקטורית לרשת וקטורית מנורמלת קצת יותר בעייתי מהמעבר מרשת סקלרית לרשת סקלרית מנורמלת, נסביר:
ברשת הסקלרית כל קובץ mat היה מיועד עבור מסנן מסויים. פשוט מצאנו את המקסימום בקובץ זה וחילקנו בו. כעת, ברשת הוקטורית, לא ניתן למצוא את מקדם הנרמול לפני יצירת כל קבצי ה-Dataset החדש, מכיוון שבכל פעם נוספות תמונות חדשות.

הפתרון לכך מעט מסובך. נשמור בוקטור normalized (שהוא וקטור בגודל 81) את מקדמי הנרמול המקסימליים שאנו מוצאים. נייצר את ה-newDataAndLabel כרגיל ללא חילוק במקדמי הנרמול.
הסבר לביצוע: במקום לשמור את מקדם הנרמול המתקבל מריצה על כל 5 תמונות נוספות בכל פעם, נבדוק אם הוא גדול ממקדם הנרמול הקודם עבור אותו מסנן ואם כן נשמור אותו.
בדרך זו באמת נשמור רק את מקדמי הנרמול המקסימליים עבור כל מסנן שיבטיחו להכניס את **כל הכניסות** לרשת לטווח [-1,1] בצורה נכונה.

לאחר מכן נעבור על כל קבצי ה-mat שיצרנו, נזכיר שקבצים אלו מכילים טבלה עם המון שורות שבכל שורה 2 עמודות שבכל עמודה וקטור עמודה בגודל 81. נחלק כל וקטור כזה בוקטור מקדמי הנרמול. איבר חלקי איבר. כך נבצע חילוק במקדם נרמול תלוי מסנן.
חילוק וקטורים איבר-איבר ב-Matlab מתבצע ע"י הוספת אופרטור הנקודה (.) לפני אופרטור החילוק (/).

ביצוע השינויים הדרושים בקוד ליצירת ה-Dataset

כאמור, נוסיף מציאה של מקדמי הנרמול המקסימליים לפי ההסבר לעיל.
לאחר מכן נבצע את החילוק בין הוקטורים המוסבר לעיל גם הוא.

ביצוע השינויים הדרושים בקוד לבחינת התוצאות

נזכור שברשת הוקטורית יש לנו את המטריצה input_matrix שבנינו שבה 81 שורות, שורה עבור כל מסנן. אנו צריכים לפני הכניסה לרשת לחלק כל שורה במטריצה במקדם הנרמול המתאים לאותו מסנן, ומיד ביציאה מהרשת להכפיל במקדמים אלה.
ב-Matlab ניתן לבצע את החלוקה הבאה בפשטות בצורה הבאה למשל:
[1,2; 3,4]./[2; 6]=[0.5,1; 0.5,0.6667]

אימון רשתות שונות ובחינת התוצאות

בחינת תוצאות של רשתות שונות על std=25 לזיהוי הרשת הטובה מבין הרשתות						
Normalized, adam, 2layers, 1024-512, sigmoid	adam, 2layers, 1024-512, sigmoid	adam, 6layers, 256-256-256-256-256-256 , sigmoid	Normalized, adam, 4layers, 256-256-256-256 ,sigmoid	adam, 4layers, 256-256-256-256 ,sigmoid	תוצאות מאמר	
26.01	27.28	25.04	27.83	26.32	29.09	barbara
28.51	29.30	28.93	29.35	29.22	29.11	boat
26.79	27.23	27.38	27.09	27.23	27.15	fingerprint
29.74	31.24	31.22	31.37	31.40	30.95	house
29.76	31.09	31.12	31.17	31.13	31.02	lena
28.68	29.61	29.24	29.67	29.55	29.04	peppers256
-1.15	-0.10	-0.57	+**0.02**	-0.25		ההפרש הממוצע מתוצאות המאמר

טבלה 4 : תוצאות מערכות מבוססות רשת וקטורית (גם לא מנורמלת וגם מנורמלות)

מילוי טבלת תוצאות עבור סטיות תקן שונות עם הרשת הטובה

מצאנו את הרשת הטובה ביותר מבין הרשתות שבחנו עבור סטיית התקן המקסימלית שנבחן – 25. התוצאות הטובות ביותר הגיעו מהרשת :

Normalized, adam, 4layes, 256-256-256-256, sigmoid

ניסינו לקבל תוצאות טובות יותר ע״י העלאת מספר האפוקים ל-10 אך ההפך קרה. כשמעלים את מספר האפוקים מ-5 ל-10 נכנסים למצב של overfitting. לכן, נישאר על 5 Epochs ונמלא טבלה שמסתכמת את התוצאות עבור רשת זו שמתאמנת כל פעם על סטיית תקן שונה.

Normalized, adam, 4layes, 256-256-256-256, sigmoid							
Noise s.t.d	barbara	boat	fingerprint	house	lena	peppers 256	ההפרש הממוצע מתוצאות המאמר
1	44.69	45.16	45.26	45.17	45.12	44.95	-3.55
2	41.65	42.09	42.16	42.37	42.28	42.23	-1.32
5	36.77	36.55	36.05	37.97	37.97	37.23	-0.72
10	32.83	33.23	31.92	34.86	34.79	33.75	-0.55
15	30.66	31.56	29.74	33.46	33.25	32.01	-0.25
20	29.15	30.36	28.23	32.47	32.19	30.77	-0.03
25	27.83	29.35	27.09	31.37	31.17	29.67	+0.02

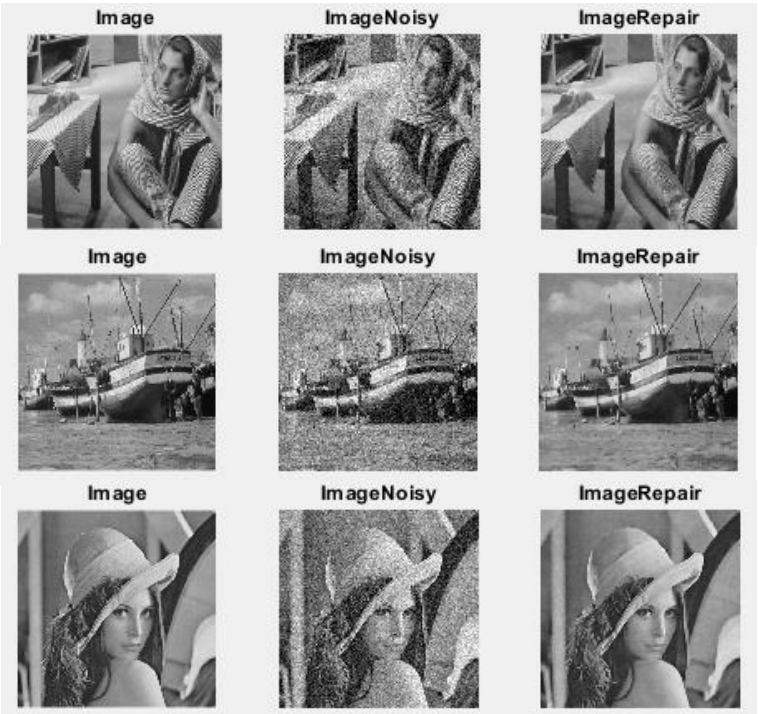
טבלה 5 : טבלת תוצאות סופית לניקוי רעשים בסטיות תקן 1,2,5,10,15,20,25

מסקנות והצגת תוצאות ויזואליות עבור סטיית תקן 25

נשים לב לתכונה מעניינת עבור ארכיטקטורת הרשת הנוכחית : ככל שה-std קטן יותר התוצאות פחות טובות.

נכון שב-std נמוכים מאוד של 1 ו-2 תוצאות ה-PSNR ממש לא טובות אך יש לקחת בחשבון שמדובר ברעש בעוצמה נמוכה מאוד שלרוב עין אנושית כלל לא תבחין בו.

נציג תמונות של התוצאות עבור הרעש החזק ביותר שבחנו (סטיית תקן של 25) :



איור 8 : ניקוי רעש גאוסי בסטיית תקן 25 על תמונת lena בעזרת הרשת שנלמדה

3.2.5 סופר רזולוציה (Super Resolution) - תמונות בגווני אפור בגודל 100X100 עם יחס הגדלה של 2

סופר רזולוציה זהו תהליך ליצירת תמונה ברזולוציה גבוהה (High resolution) מתמונה ברזולוציה נמוכה (Low resolution). תהליך סופר רזולוציה זהו תהליך מאתגר מכיוון שלא ניתן לשחזר את תוכן התמונה המקורי בתדרים הגבוהים מתוך התמונה ברזולוציה הנמוכה.

בפרויקט זה נתכנן Neural network לביצוע סופר רזולוציה.
הרשת לומדת את המיפוי בין התמונה ברזולוציה הנמוכה לתמונה ברזולוציה הגבוהה.
מיפוי זה אפשרי מכיוון שלתמונה ברזולוציה הנמוכה ולתמונה ברזולוציה הגבוהה יש תוכן תמונה דומה והן נבדלות בעיקר בפרטי התדירות הגבוהה.

הרשת שאנו מתכננים משתמשת בטכניקה ללמידת residual, כלומר הרשת לומדת להעריך את תמונת השרידים (residual image).
בהקשר של תהליך הסופר רזולוציה ההפרש בין התמונה ברזולוציה גבוהה לבין התמונה ברזולוציה נמוכה שהוגדלה (upscale) ע"י אינטרפולציית ביקיוביק (bicubic interpolation) להתאמת הגודל לתמונת הרפרנס (reference image).
תמונה ה-residual מכילה את המידע על הפרטים בתדירות הגבוהה של התמונה.

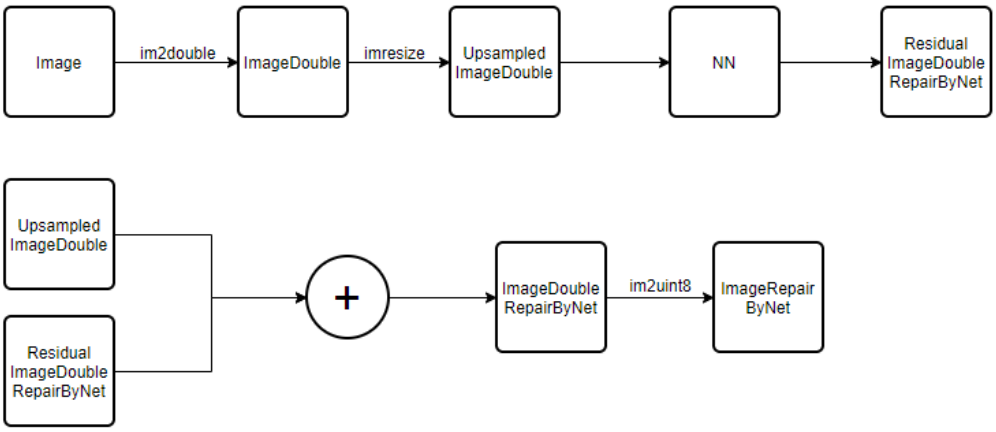
נתחיל את העבודה בתנאים הבאים :

- תמונות בגודל קבוע של 100X100
- תמונות בגווני אפור

נשתמש במסקנות מהרשת לניקוי רעשים ולפני הכניסה לרשת נבצע נרמול לערכי התמונה על מנת לשפר את תוצאות הרשת.

במערכת הבאה יש שוני מהותי מהפתרון שהצענו לניקוי רעשים והוא העובדה שעברנו לעבוד במישור התמונה, כלומר הכניסה לרשת היא תמונה והיציאה ממנה היא תמונה.

כעת נתאר בתרשים בלוקים כיצד מבוצע תהליך שלם של סופר רזולוציה על תמונה אחת :



איור 9 : תרשים מלבנים למערכת לביצוע סופר רזולוציה לתמונה (כולל שימוש בתמונה המקורית לבחינת התוצאות)

הערות:

1. בפעולת ה-imresize הכוונה להקטנת התמונה לפי היחס הרצוי והגדלתה בחזרה לפי אותו היחס. כך נדמה תמונה ברזולוציה לא טובה שנכנסה למערכת ונוכל להשוות את התוצאה לתמונה ברזולוציה הגבוהה.
2. השמות המתוארים בדיאגרמה הם גם השמות שבהם השתמשתי בקוד ה-Matlab הממומש את התהליך.

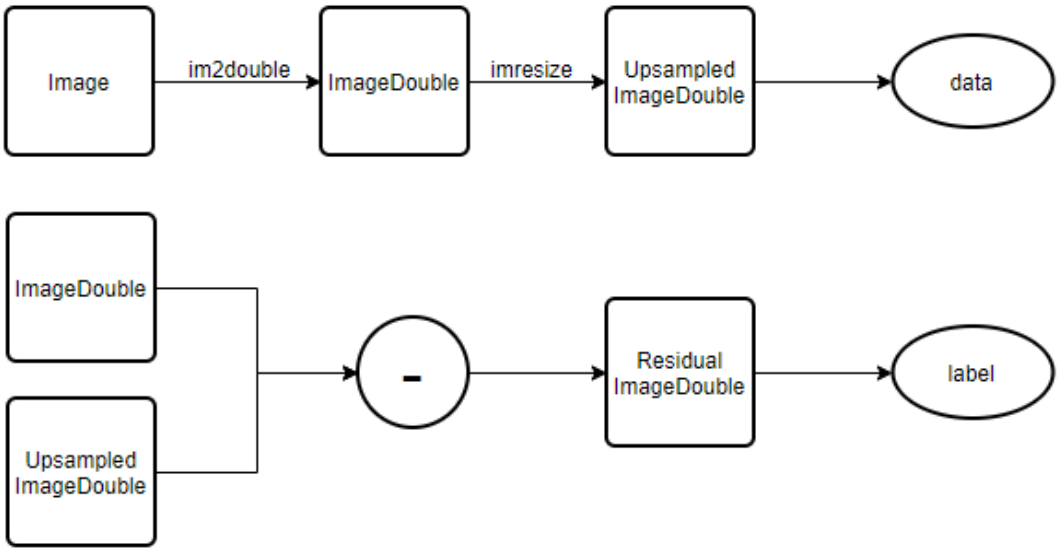
יצירת ה-Dataset

הרעיון כללי:

קבלת תיקייה עם תמונות והמרתה ל-4 תיקיות נפרדות השמות:
trainUpsampledImages, trainResidualImages, validationUpsampledImages, validationResidualImages

הערה:

1. מכיוון שכרגע אנו עובדים עם תמונות בגווני אפור, נשנה כל תמונה שאנו קוראים מה-Dataset שלנו לתמונות בגווני אפור.
2. תמונת ה-residual כוללת בתוכה גם ערכים שליליים ובנוסף לכך אנו גם מנרמלים את ערכי התמונה ולכן לא ניתן לשמור אותה כתמונה (עם סיומת סטנדרטית של תמונה).
לכן נשמור את המידע בקבצי mat



איור 10: תרשים מלבנים המתאר ליצירת ה-data וה-"label" של הרשת

תכנון ובניית הרשת תוך הטמעת התמרות המרחבים ברשת

את בניית הרשת נבצע בשפת Python ע״י שימוש בספריית Keras. זהו מרכיב חשוב מאוד בחלק זה של ה-Super Resolution ולכן נציג את כל חלקי הקוד ונסביר על כל חלק בנפרד בהרחבה.

<u>חלק א – טעינת קבצי ה-mat של ה-Dataset שיצרנו מ-Matlab:</u>

הערה: כאן הצגתי טעינה של תיקייה אחת, אך יש לבצע עבור כל ארבעת התיקיות שיצרנו.

נסביר על טעינת קבצי ה-mat מתיקייה מסוימת:
הרעיון הכללי הוא להכניס את המידע מכל תיקייה לרשימה של תמונות (כלומר, רשימה של רשימות דו ממדיות).
אמנם יש לשים לב לדבר חשוב מאוד:
רשימות רב ממדיות (list) תופסות הרבה יותר מקום בזיכרון לעומת מערכי numpy. זה דבר מאוד משמעותי מכיוון שזיכרון ה-RAM מוגבל כמובן. אנו חייבים לצמצם כמה שיותר בצריכת זיכרון זה. לשם המחשה טעינת תיקייה של תמונות ששוקלת בערך 2GB, צורכת בערך 14GB כאשר שומרים אותה ברשימה תלת ממדית כפי שתיארנו. לעומת זאת אם נשמור אותה ב-numpy array בעל שלושה ממדים היא תצרוך בדיוק 2GB.

נגדיר מערך תלת ממדי כאשר הגודל החיצוני הוא מספר קבצי ה-mat, הממד השני הוא מספר השורות בכל תמונה והממד השלישי הוא מספר העמודות בכל תמונה. נעבור על כל הקבצים בתיקייה ונמלא את המערך התלת ממדי. בסיום עבודה על כל קובץ נדפיס את שמו למסך לשם נוחות ודיבוג הקוד בזמן הריצה.

הערה:
נשים לב שלאחר מילוי המערך התלת ממדי נוסיף ממד של 1 בסוף לשם התאמה לשכבת הכניסה של הרשת. זאת ניתן להבין מאחר וללא הוספת הממד מתקבלת הודעת השגיאה הבאה:

```
ValueError: Input 0 of layer conv2d is incompatible with the layer:
expected ndim=4, found ndim=3. Full shape received: [None, 100, 100]
```

נשים לב שזה לא קורה בגלל ה-1 שכתוב בשכבה הראשונה של הרשת - גם אם מורידים אותו נשארים עם אותה שגיאה בדיוק. בנוסף גם אם מורידים אותו וגם את הממד הנוסף שהוספנו לכל מערך תלת ממדי נשארים עם אותה השגיאה. לאחר קריאה אודות שגיאה זו הבנתי שהוספת ממד של 1 במקום המתאים זהו הפתרון הנדרש.

חלק ב – טעינת המשקלים (Weight) שיצרנו בקוד שמייצר את המסננים :

הסבר על יצירת המסננים :

בכניסה לרשת ישנה שכבת קונבולוציה שמטרתה לפרק את התמונה ל-81 תמונות בפסי תדר שונים. ההמרה מתבצעת ע״י קונבולוציה של התמונה בכניסה עם 81 מסננים שונים עבור 81 פסי התדר. גודל כל מסנן הוא 9X9, והמסננים הם בעצם פונקציות בסיס של התמרת DCT. זוהי ההתמרה שבניקוי הרעש ביצענו מחוץ לרשת – עכשיו היא מתבצעת בתוך הרשת.

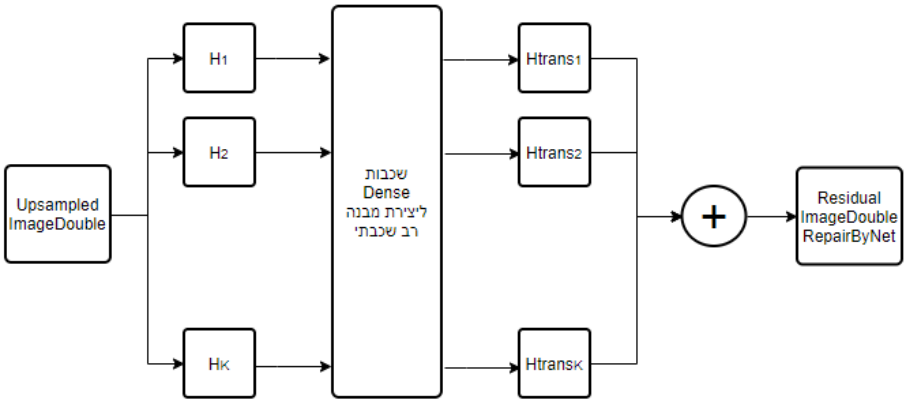
הקוד המחשב את 81 המסננים ושומר אותם לקובץ mat מתואר בנספח א לפרויקט.
אמנם לקוד זה עלינו להוסיף עוד חלק קטן של התאמת ממדים.
התאמת הממדים של המשקלים בצורה נכונה :
ראשית בניתי את השכבה הראשונה והאחרונה ללא אתחול וקיבוע של המשקלים, כך ע״י שימוש בפקודה model.layer[i].get_weight()[0] קיבלתי את המבנה בו שמורים ה-Weight, וע״י הפקודה model.layer[i].get_weight()[1] קיבלתי את המבנה בו שמורים ה-Biases.

מסקנות :
בשכבה הראשונה :
המשקלים : shape(9,9,1,81) – כלומר הממד השלישי קבוע 1 והממד הרביעי הוא מספר המסנן.
הביאסים : shape(81,)
בשכבה האחרונה :
המשקלים : shape(9,9,81,1) – כלומר בממד השלישי הוא מספר המסנן וממד הרביעי קבוע 1.
הביאסים : shape(1,)
מכאן ידעתי איך לבצע את שינוי הממדים ב-Matlab לממדים הנכונים כפי שמבוצע הקוד הבא שהוספנו לקוד מהנספח :

הערה :
נשים לב שב-Matlab אין משמעות לממד של 1 בממד האחרון, אלא רק במעבר ל-Python.
לכן במשקלים של השכבה האחרונה הוספנו ממד של 1 בעזרת שורת הקוד :
WeightsForLast = numpy.expand_dims(WeightsForLast, axis=3)

הבדיקה שהאתחול והקיבוע של הפרמטרים הצליח היא ע״י הכנסת תמונה לרשת, וחיסור בין התמונה במוצא לתמונה בכניסה. באמת הכל מתאפס חוץ מ-4 פיקסלים בכל דופן (בגלל הקונבולוציה עם מסנן בגודל 9X9).

חלק ג – יצירת ארכיטקטורת הרשת ואימונה :



איור 11 : תרשים מלבנים לתיאור ארכיטקטורת הרשת לביצוע סופר רזולוציה במישור התמונה

כאמור, בכניסה לרשת ישנה שכבת קונבולוציה שמטרתה לפרק את התמונה ל-81 תמונות בפסי תדר שונים. ההמרה מתבצעת ע"י קונבולוציה של התמונה בכניסה עם 81 מסננים שונים עבור 81 פסי התדר. גודל כל מסנן הוא 9X9, והמסננים הם בעצם פונקציות בסיס של התמרת DCT.

לבסוף מופיעה שכבת קונבולוציה אשר מטרתה לבצע את התמרת DCT ההפוכה, ע"י ביצוע קונבולוציה של כל תמונה בפס תדר כלשהו עם המסנן המשוחלף באותו פס תדר, ולבסוף לחבר את כל פסי התדר לקבלת תמונה אחת.

הערה חשובה :

השכבה בכניסה – שאחראית על פירוק התמונה לפסי תדר שונים והשכבה ביציאה – שאחראית על הרכבת התמונה חזרה מתמונות פסי התדר לא מתאמנות כלל. נבצע זאת ע"י קביעת פרמטר trainable בשכבות אלה לערך false.

בין שתי שכבות אלה יופיעו "שכבות Dense" שכרגע גם הם ממומשות עם שכבות "Conv2D", ונסביר מדוע :

אנו רוצים בעצם להשתמש בדיוק באותו רעיון אשר ביצענו ברשת הוקטורית של ניקוי רעשים (De-Noise) – אנו רוצים לעבור על וקטורים בגודל 81 המורכבים מפיקסל אחד מכל פס תדר.

על מנת לבצע זאת נשתמש בקונבולוציה דו ממדית עם גודל מסנן 1X1.
הפעלת מסנן כזה שקולה לנוירון אחד שפועל על כל 81 השכבות וסורק את כל הפיקסלים.
אם נקצה N מסננים כאלה הדבר שקול ל-N פרספקטרונים.
בצורה זו אפשר ליצור מבנה רב שכבתי שזהה למבנה שבנינו ברשת הוקטורית של הפחתת רעש איתו אנו רוצים לעבוד.

נשים לב שלפני שכבה זו חובה לשים שכבה עם 81 פרספקטרונים מכיוון שאם נכנסים וקטורים בגודל 81 למבנה הרב שכבתי אנו רוצים שגם יצאו ממבנה זה וקטורים בגודל 81.

בקוד המוצג להלן שמנו באמצע 4 שכבות בעלות 256 פרספקטרונים כל אחת (בהתאם לרשת הטובה ביותר שקיבלנו מניקוי הרעשים בחלק הראשון של הפרויקט) ופונקציית האקטיבציה של כל שכבה היא relu.

הערה:
חשוב לציין שתחילה ניסיתי עם פונקציית אקטיבציה של sigmoid (כמו שביצענו בניקוי הרעשים) והתוצאות היו פחות טובות בצורה מאוד משמעותית – התוצאות שהתקבלו ע"י שימוש באינטרפולציית Bicubic היו טובות יותר מאשר התוצאות עם הרשת. מכן ניתן ללמוד על חשיבותה הרבה של פונקציית האקטיבציה.

```
model = Sequential()
model.add(Conv2D(81,(9,9),padding='same', input_shape=(imageDim,imageDim,1),
weights=[WeightsForFirst,numpy.zeros(81,)], trainable=False))
model.add(Conv2D(256,(1,1),activation='relu'))
model.add(Conv2D(256,(1,1),activation='relu'))
model.add(Conv2D(256,(1,1),activation='relu'))
model.add(Conv2D(256,(1,1),activation='relu'))
model.add(Conv2D(81,(1,1)))
model.add(Conv2D(1,(9,9),padding='same', weights=[WeightsForLast,numpy.zeros(1,)],
trainable=False))
model.compile(optimizer='adam', loss='mse')
print(model.summary())
numOfEpochs = 15
trained_model = model.fit(trainData, trainLabel, validation_data=(validationData,
validationLabel), epochs=numOfEpochs, batch_size=32)
```

איור 12 : יצירת ארכיטקטורת רשת לביצוע סופר רזולוציה הפועלת במישור התמונה ואימונה ב-Keras

<u>חלק ד – יצירת גרף loss עבור ה-train וה-validation ושמירתו וכן שמירת מודל הרשת:</u>

כאן יש דבר אחד שחשוב לשים אליו לב:
אנו שומרים את המודל בפורמט h5 (זו הדרך הנפוצה לשמירת מודלים של רשתות).
המטרה היא שנוכל לקרוא אותו ב-Matlab ולהשתמש בו.
כאן חובה להשתמש בגרסת tensorflow של 2.1.0 ובכך לקבל גרסת keras של 2.2.4.
במידה ומשתמשים ב-visual studio ניתן להוריד את הספרייה בקלות ע"י הרצת השורה:
pip install tensorflow==2.1.0 –user
אם לא משמשים בגרסה זו, לא ניתן לקרוא ב-Matlab את המודל.

בחינת התוצאות על Datasets שונים

נכתוב קוד ב-Matlab שמבצע בדיוק את המתואר באיור מס' 12.

בקוד מבוצע חישוב של שני פרמטרים חשובים לקביעת איכות התוצאות שהתקבלו והם: PSNR ו-SSIM. פרמטרים אלה מחושבים גם עבור התמונה המתקבלת מאינטרפולציית

bicubic וגם עבור התמונה המתקבלת לאחר הרשת שבניתי.

אנו רוצים לבחון את התוצאות על Datasets שונים ומוכרים אשר מכילים תמונות רבות והם: Set5, Set14, BSDS100, Urban100, Manga109.

אז נוסיף בתחילת הקוד קטע שבונה שתי טבלאות שישמשו עבור התוצאות:

על מנת לבחון את תוצאות המערכת על תמונה ספציפית גם באופן חזותי וגם עם PSNR ו-SSIM יש להריץ את הקוד RepairOneImage.m המצורף לתיקייה.

הערה: סוף הקוד כתוב בצורה שמטלב יאפשר להציג תמונות בגדלים שונים באותו חלון.

אימון הרשת ובחינת התוצאות:

רזולוציה מלאה – 100X100, יחס הגדלה של 2, 15 epochs, ה-Dataset לאימון: תיקיות 00-10 של iaprtc12, train – 5159 תמונות, validation – 1291.

	Set5	Set14	BSDS100	Urban100	Manga109	שיפור ממוצע
Bicubic	32.77	31.18	34.21	Inf(34.48)	35.44	
Network	35.25	32.65	35.85	36.58	39.83	
שיפור ב-dB	2.48	1.47	1.64	2.1	4.39	**2.41**

טבלה 6: טבלת סיכום תוצאות סופר רזולוציה ביחס הגדלה של 2, כאשר הרזולוציה המלאה היא 100X100 והשוואת התוצאות לאינטרפוצליית Bicubic

3.2.6 מעבר לתמונות בגווני אפור בגודל 510X510 עם יחס הגדלה של 2

המעבר כולל בתוכו כמה שלבים עיקריים :

- מציאת Dataset מתאים לאימון ובחינת התוצאות
- שינוי הקודים
- אימון הרשתות ובחינת התוצאות

מציאת Dataset חדש לאימון ובחינת התוצאות

בתמונות של 100X100 יכולנו להשתמש בתמונות מה-Dataset של iaprtc12, אך התמונות שם בגודל 480X360 או 360X480 ולכן הן לא מספיקות לאימון לתמונות ברזולוציה 510X510.

פתרתי בעיה זו ע"י שימוש בשני Dataset אחרים : DIV2K ו-Flickr1024.
ב-DIV2K יש : 800 – Train תמונות, 100 – Validation תמונות.
ב-Flickr1024 יש : 1600 – Train תמונות, 224 – Validation תמונות, 224 – Test תמונות.
נאחד את התמונות של DIV2K-Train, Flickr1024-Train, Flickr1024-Validation לתוך תיקיה בשם : imagesDirForTrainAndValidation510x510

רוב התמונות האלה גדולות מ-510X510 אבל כדי להישאר רק עם תמונות שגדולות מ-510X510 נריץ קוד Matlab שמוחק תמונות שקטנות מ-510X510 בתיקיה imagesDirForTrainAndValidation510x510 ונריץ אותו. הקוד מצורף לפרויקט.

לאחר ריצת הקוד, בתיקייה נשארו 2362 תמונות שיתחלקו ל : 1889 – Train, 473 – Validation. על מנת לבחון את התוצאות נשתמש ב-Datasets הבאים :

- DIV2K_valid_HR (100 תמונות)
- Flickr1024\Test (184 תמונות לאחר מחיקה של תמונות לא בגודל המתאים)
- Urban100 (100 תמונות)
- Manga109 (109 תמונות)

הערה : על Flickr1024\Test הרצנו את הקוד לעיל להורדת תמונות שלא בגודל המתאים.

שינוי הקודים הדרוש למעבר

נשנה בכל שלושת הקודים את המספר 100 ל-510 בכל מקום.
באימון הרשת, את ה-batch_size נשנה ל-4 על מנת לפתור את שגיאת הזיכרון שקפצה.
בנוסף בקוד זה נשנה את הממד הראשון של המערכים בהתאם למספר הקבצים.
ביציאת טבלת התוצאות נשנה את שמות התיקיות בהתאם ל-Dataset שאנו בוחנים כעת.

אימון רשתות שונות ב-Python ובחינת התוצאות

כעת בשורה מ-100X100 מכיוון שמדובר בחלק מרכזי, אימנו ובחנו את התוצאות של כמה רשתות שונות אשר מתוארות בטבלה להלן:

רזולוציה מלאה – 510X510, יחס הגדלה של 2, 15 epochs, ה-Dataset לאימון: תיקיית imagesDirForTrainAndValidation510x510 שבנייתה תוארה לעיל, train – 1889 תמונות, validation – 473.

שיפור ממוצע	Flickr1024/Test	Manga109	Urban100	DIV2K_valid_HR	
	26.29	29.66	25.91	36.01	Bicubic
	28.44	34.68	28.81	37.96	adam,4layers,256-256-256-256, relu
3.01	2.15	5.02	2.90	1.95	שיפור ב-dB
	28.47	34.62	28.83	38.03	adam,2layers,1024-512, relu
3.02	2.18	4.96	2.92	2.02	שיפור ב-dB
	28.47	34.52	28.88	37.93	adam,4layers,256-256-256-256, relu+endTanh
2.98	2.18	4.86	2.97	1.92	שיפור ב-dB
	27.20	31.66	26.93	36.97	adam,4layers,256-256-256-256, tanh
1.22	0.91	2.00	1.02	0.96	שיפור ב-dB

טבלה 7: טבלת סיכום תוצאות של רשתות שונות לביצוע סופר רזולוציה ביחס הגדלה של 2, כאשר הרזולוציה המלאה היא 510X510 והשוואת התוצאות לאינטרפולציית Bicubic

מסקנות סופר רזולוציה ביחס הגדלה של 2

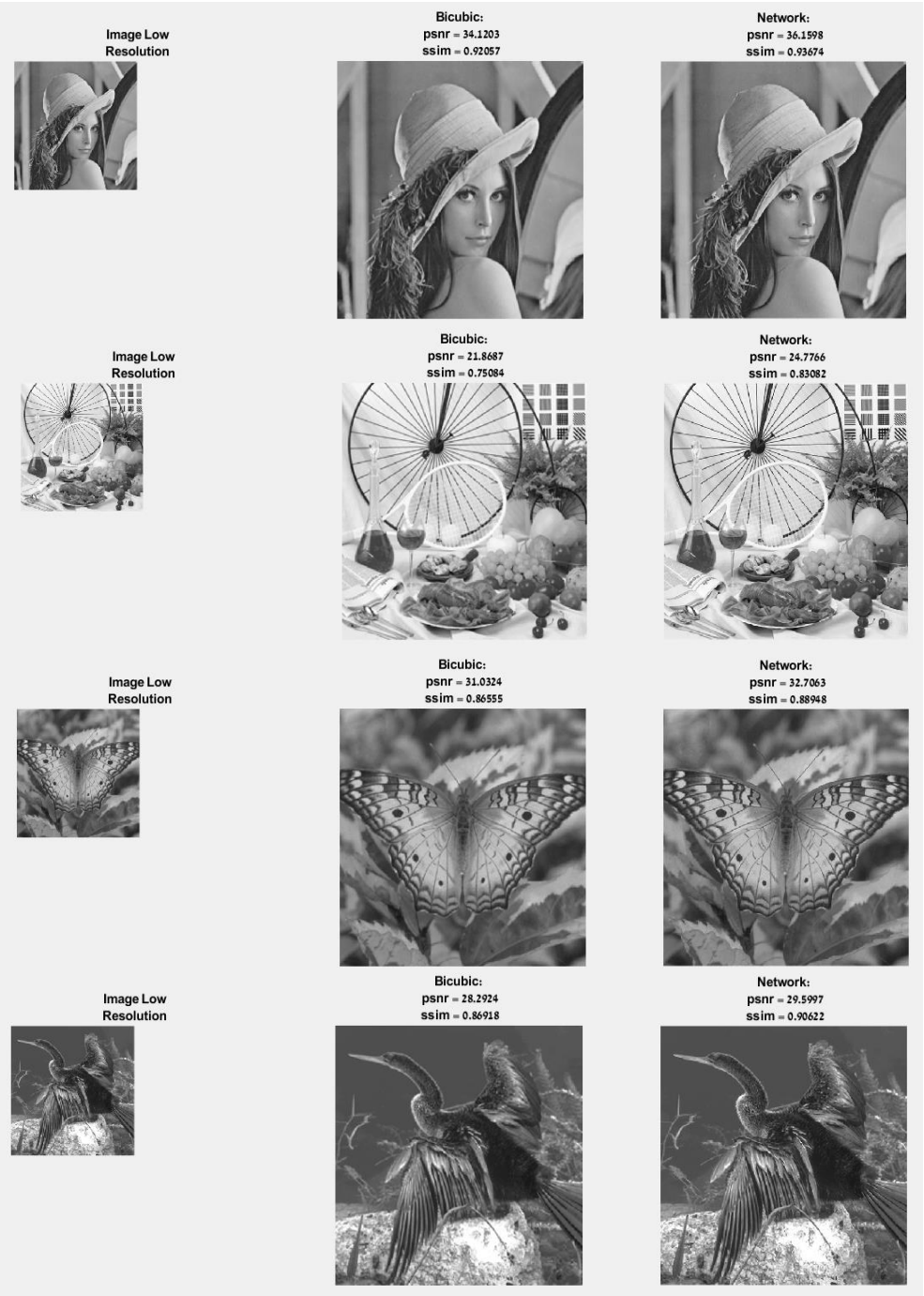
הרשת שהייתה הכי טובה בניקוי רעשים ממשיכה להיות הרשת הכי טובה ומצטרפת אליה רשת נוספת בעלת 2 שכבות של 1024-512 (כמובן שלא נתייחס להפרש 0.01dB בין הרשתות).

ניסינו להוסיף activation של tanh לשכבה האחרונה על מנת לוודא שמוצא הרשת נשאר בטווח של [-1,1] כפי שאנו מצפים. פעולה זו לא שיפרה את התוצאות אך היא לא פגעה בהן.

ניסינו פונקציית אקטיבציה שונה – tanh, במקום relu בכל ארבעת השכבות הקיימות. התוצאות שקיבלנו ממש לא טובות. נזכיר שבעבר בדקנו גם את sigmoid והתוצאות היו גם לא טובות. לכן נישאר סופית עם relu.

נחליט להמשיך עם הרשת המקורית של adam,4layers,256-256-256-256, relu.

הצגה ויזואלית של התוצאות על הרשת הטובה



איור 13 : הצגה ויזואלית של תוצאות הרשת לסופר רזולוציה ביחס הגדלה של 2 כאשר הרזולוציה המלאה היא 510X510.

3.2.7 מעבר לתמונות בגווני אפור בגודל 510X510 עם יחס הגדלה של 3

שינוי הקודים הדרוש למעבר

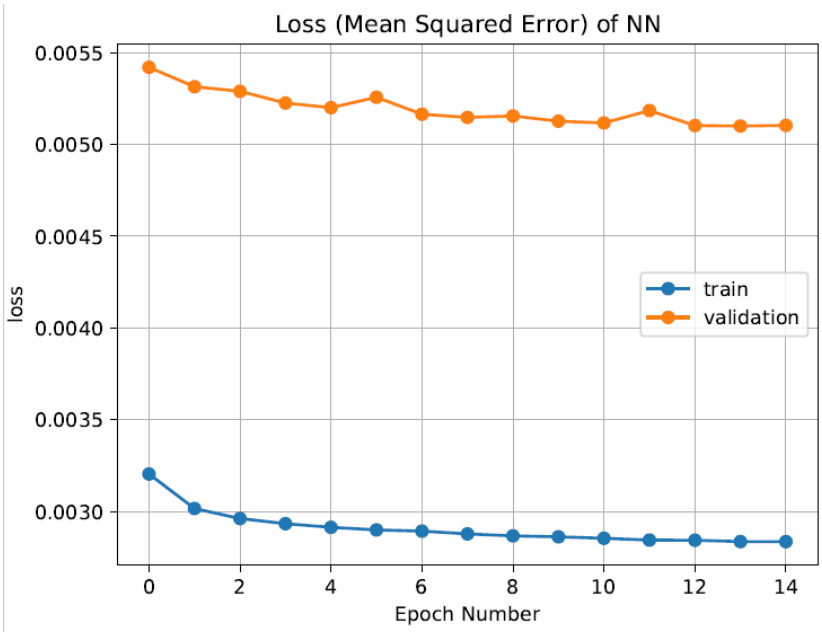
פשוט בקודים createDataSet.m ו-RepairImage.m נשנה את הפרמטר scaleFactor ל-3, ניצור שוב קבצי mat לאימון ונאמן את הרשת שלנו שהביאה עד עכשיו את התוצאות הטובות ביותר שקיבלנו.

בחינת התוצאות עם הרשת הטובה

רזולוציה מלאה – 510X510, יחס הגדלה של 3, 15 epochs, ה-Dataset לאימון: תיקיית imagesDirForTrainAndValidation510x510 שבנייתה תוארה לעיל, train – 1889 תמונות, validation – 473.

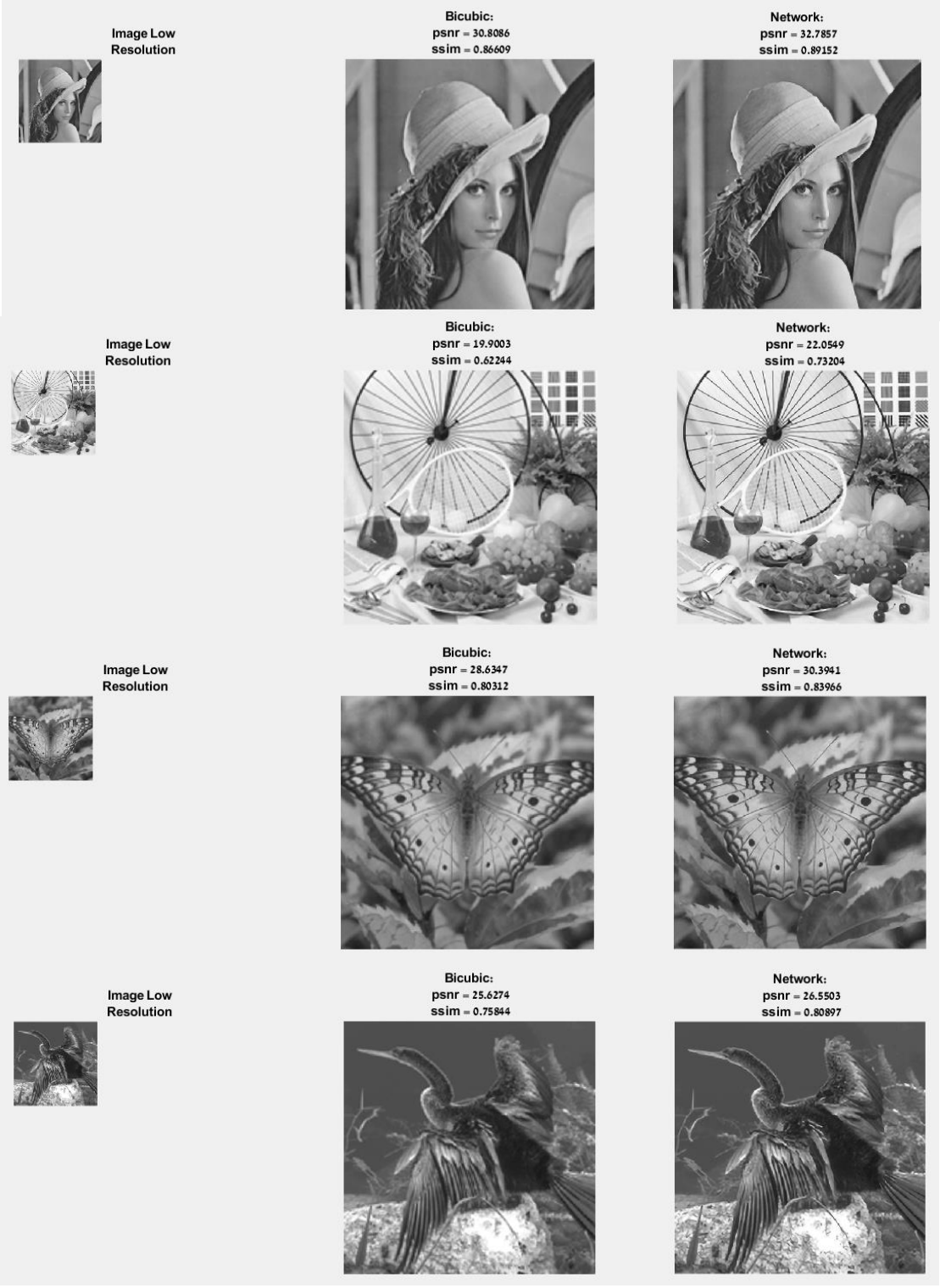
	DIV2K_valid_HR	Urban100	MANGA109	Flickr1024/Test	שיפור ממוצע
Bicubic	33.27	23.50	25.78	24.15	
adam,4layers,256-256-256-256, relu	34.53	25.35	29.49	25.37	
שיפור ב-dB	1.26	1.85	3.71	1.22	**2.01**

טבלה 8: תוצאות הרשת לביצוע סופר רזולוציה ביחס הגדלה של 3, כאשר הרזולוציה המלאה היא 510X510



איור 14: גרף ה-Loss של ה-Train וה-Validation בהתאם למספר האפוקים

הצגה ויזואלית של התוצאות על הרשת הטובה



איור 15 : הצגה ויזואלית של תוצאות הרשת לסופר רזולוציה ביחס הגדלה של 3 כאשר הרזולוציה המלאה היא 510X510.

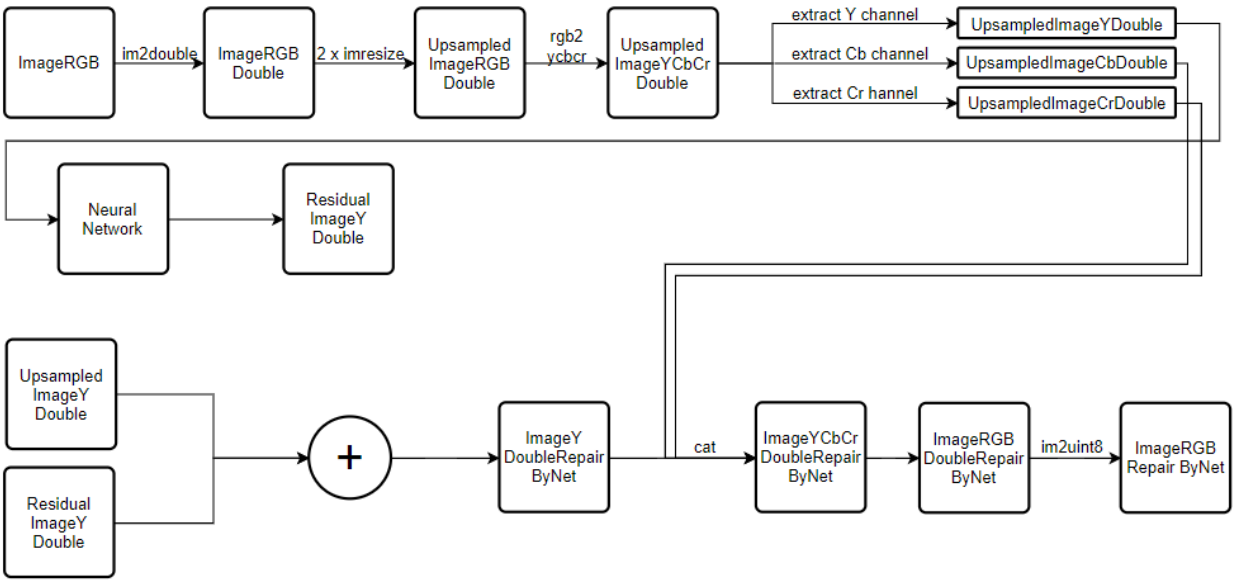
3.2.8 מעבר לתמונות צבעוניות בגודל 510X510 ביחסי הגדלה של 2 ו-3

הצגת הרעיון הכללי למעבר לתמונות צבעוניות

על מנת לבצע סופר רזולוציה על תמונה צבעונית עלינו לעבור ממרחב RGB למרחב YCbCr. על פי המוצג במאמר [2], כאשר עוברים למרחב YCbCr מספיק לבצע את תיקון הרשת רק על ערוץ ה-Y ואילו את ערוצים Cb ו-Cr להעביר פשוט באינטרפולציית Bicubic.

המעבר למרחב YCbCr חוסך לנו את השימוש ב-3 רשתות שונות עבור R,G,B בנפרד.

תרשים מלבנים המציג את פעולת ה-Super Resolution על תמונה צבעונית:



איור 16: תרשים מלבנים המציג את פעולת ה-Super Resolution על תמונה צבעונית

נסביר את השלבים:

1. קבלת תמונה RGB (לשם חישוב PSNR זו תהיה התמונה ברזולוציה הגבוהה).
2. נרמול התמונה לתחום [0,1] בעזרת im2double.
3. ביצוע הקטנה ולאחר מכן הגדלה בהתאם לפרמטר scale (התמונה רק לאחר הקטנה היא התמונה בכניסה לרשת).
4. המרת התמונה המוגדלת למרחב YCbCr.
5. חילוץ ערוצי ה - Y, Cb, Cr מהתמונה המוגדלת.
6. העברת ערוץ ה-Y ברשת שאימנו וקבלת תמונה ה-Residual של ערוץ Y.
7. חיבור ערוץ Y של התמונה המוגדלת עם תמונה ה-Residual של ערוץ Y לקבלת ערוץ Y המתוקן.
8. חיבור ערוצי ה - Y, Cb, Cr מהתמונה המוגדלת (כאשר ה-Y לאחר תיקון של הרשת) לקבלת התמונה המוגדלת המרחב YCbCr.
9. המרת התמונה למרחב RGB.
10. המרת התמונה ל-uint8 על מנת שניתן יהיה להציג את התמונה.

הערות:

את תוצאות הרשת במדדי PSNR ו-SSIM נבחן על ערוץ ה-Y, כך נבדוק את תוצאות הרשת עצמה. כך גם מבצעים במאמרים ברחבי האינטרנט העוסקים בסופר רזולוציה.

נבחר לעבוד עם scale3 וכמובן נמשיך עם אותה הרשת: adam,4layers,256-256-256-256,relu.

לגביי יצירת ה-Dataset, תחילה אימנתי מחדש את הרשת על תמונות Y מכיוון שאולי כך נקבל תוצאות טובות יותר אך אין בכך צורך מכיוון שה-Dataset שקיים כעת הוא על תמונות אפור מנורמלות ועובדה זו תורמת לכך שאין שום הבדל ב-Dataset איתו נאמן את הרשת.

בחינת התוצאות עבור יחס הגדלה של 2

	DIV2K_valid_HR	Urban100	MANGA109	Flickr1024/Test	שיפור ממוצע
Bicubic	37.37	27.23	30.98	27.62	
adam,4layers,256-256-256-256, relu	39.23	30.11	35.97	29.74	
שיפור ב-dB	1.86	2.88	4.99	2.12	**2.96**

טבלה 9: תוצאות הרשת לביצוע סופר רזולוציה על תמונות צבעוניות ביחס הגדלה של 2, כאשר הרזולוציה המלאה היא 510x510

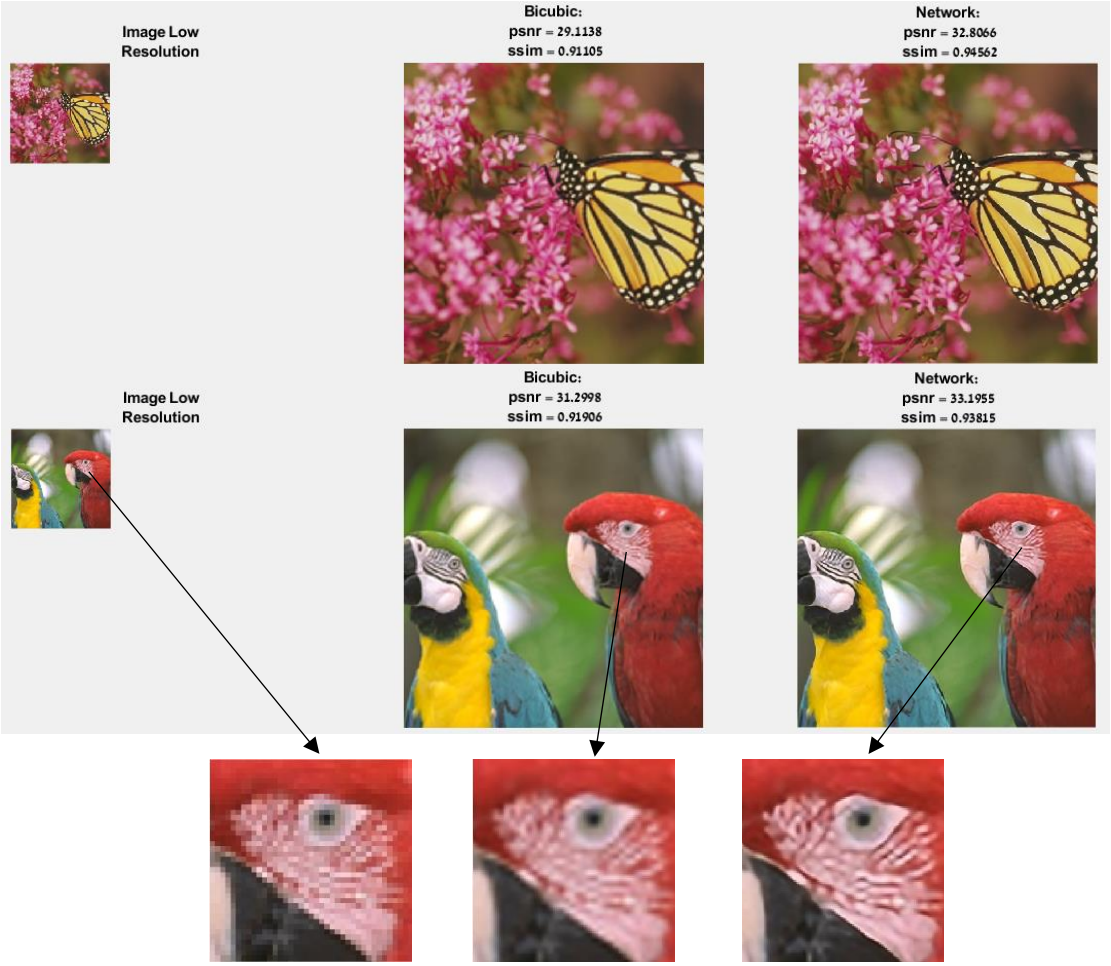
בחינת התוצאות עבור יחס הגדלה של 3

	DIV2K_valid_HR	Urban100	MANGA109	Flickr1024/Test	שיפור ממוצע
Bicubic	34.61	24.83	27.11	25.48	
adam,4layers,256-256-256-256, relu	35.85	26.68	30.85	26.70	
שיפור ב-dB	1.24	1.85	3.74	1.22	**2.01**

טבלה 10: תוצאות הרשת לביצוע סופר רזולוציה על תמונות צבעוניות ביחס הגדלה של 3, כאשר הרזולוציה המלאה היא 510x510

נשים לב שקיבלנו בדיוק את אותו שיפור ממוצע (וגם את אותו השיפור בכל תמונה) ולכן המעבר לתמונה צבעונית בוצע בהצלחה.

הצגה ויזואלית של התוצאות עבור יחס הגדלה של 3

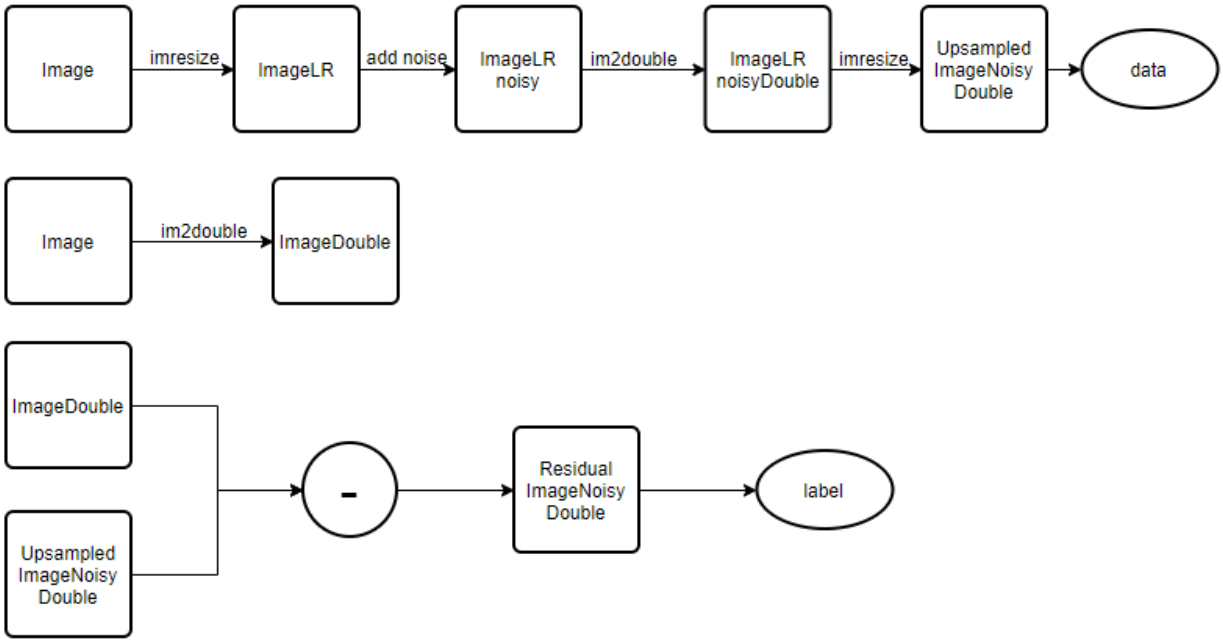


איור 17 : הצגה ויזואלית של תוצאות המערכת לביצוע לסופר רזולוציה על תמונות צבעוניות ביחס הגדלה של 3 כאשר הרזולוציה המלאה היא 510X510.

3.2.9 ניקוי רעשים וסופר רזולוציה – Denoising and Super Resolution

כעת נבצע ניקוי רעשים וסופר רזולוציה באותה הרשת כאשר את הרעש נוסיף כמובן לתמונה המוקטנת.

יצירת ה-Dataset



איור 18 : תרשים מלבנים להסבר יצירת ה-Dataset עבור הרשת לביצוע ניקוי רעשים וסופר רזולוציה יחד

נבנה שתי רשתות :

- רשת אחת שמתקנת רעש עם סטיית תקן של 5
- רשת שניה שמתקנת רעש עם סטיית תקן של 10

אנו עדיין עובדים עם תמונות בגודל 510X510, נבחר יחס הגדלה של 2 ונעבוד עם הרשת שהניבה את התוצאות הטובות יותר (adam,4layers,256-256-256-256, relu).

הצגת אלגוריתמים להשוואה

כעת בהשוואת התוצאות ישנה בעיה, לא נכון להשוות את תוצאות הרשת לאינטרפולציית Bicubic מכיוון שאינטרפולציית Bicubic לא מפחיתה רעש אלא רק מגדילה את התמונה.

לכן, על מנת לקבל תמונה רחבה על אופציות שונות לביצוע פעולה של הפחתת רעש וסופר רזולוציה נשווה בין שלושה אלגוריתמים שונים:

1. ניקוי רעשים מהתמונה הרועשת באמצעות הרשת שנלמדה מהחלק הראשון בפרויקט ולאחר מכן ביצוע של אינטרפולציית Bicubic להגדלת התמונה.
2. ניקוי רעשים מהתמונה הרועשת באמצעות הרשת שנלמדה מהחלק הראשון בפרויקט ולאחר מכן הגדלה ע"י Super Resolution עם הרשת מהחלק השני.
3. שימוש ברשת שאנו מאמנים עכשיו שמבצעת גם ניקוי רעשים וגם Super Resolution.

על מנת שנוכל לבצע השוואה זו כתבנו שתי פונקציות ב-Matlab. אחת מקבלת תמונה עם רעש, מתקנת אותה לפי החלק הראשון של הפרויקט ומחזירה אותה - DeNoiseOneImageFunction, והשנייה מקבלת תמונה ומבצעת עליה תהליך של Super Resolution כפי שביצענו בחלק השני של הפרויקט - SuperResolutionOneImageFunction.

פונקציות אלה שימושיות מאוד מכיוון שהן מאפשרות להריץ בהן תמונה במערכת ללא שום צורך בידע קודם על המערכת, הן פונקציות שניתן בקלות להכניס לממשק משתמש.

בחינת התוצאות עבור יחס הגדלה של 2 וסטיית תקן 5

פרטים מלאים על ההרצה:

רזולוציה מלאה – 510X510, **יחס הגדלה של 2 + רעש גאוסי עם סטיית תקן 5**, 15 epochs, ה-Dataset לאימון: תיקיית imagesDirForTrainAndValidation510x510, train – 1889 תמונות, validation – 473.

Std5	DIV2K_valid_HR	Urban100	MANGA109	Flickr1024/Test	ממוצע
אלגוריתם 1: רשת לניקוי רעשים מהחלק הראשון ואז Bicubic	33.07	25.26	28.54	25.69	28.14
אלגוריתם 2: ניקוי רעשים עם רשת מהחלק הראשון ואז הגדלה עם רשת מהחלק השני	33.68	27.18	31.41	27.20	29.87
אלגוריתם 3: רשת אחת לניקוי רעשים ולסופר רזולוציה	34.70	27.70	32.43	27.50	30.58

טבלה 11: השוואת תוצאות של שלושה אלגוריתמים שונים לביצוע ניקוי רעשים וספור רזולוציה (יחס הגדלה של 2 וסטיית תקן של 5)

בחינת התוצאות עבור יחס הגדלה של 2 וסטיית תקן 10

פרטים מלאים על ההרצה:

רזולוציה מלאה – 510X510, **יחס הגדלה של 2 + רעש גאוסי עם סטיית תקן 10**, 15 epochs, ה-Dataset לאימון: תיקיית imagesDirForTrainAndValidation510x510, train – 1889 תמונות, validation – 473.

Std5	DIV2K_valid_HR	Urban100	MANGA109	Flickr1024/Test	ממוצע
אלגוריתם 1: רשת לניקוי רעשים מהחלק הראשון ואז Bicubic	31.61	24.63	27.51	25.08	27.21
אלגוריתם 2: ניקוי רעשים עם רשת מהחלק הראשון ואז הגדלה עם רשת מהחלק השני	31.85	25.88	29.20	26.10	28.26
אלגוריתם 3: רשת אחת לניקוי רעשים ולסופר רזולוציה	32.78	26.53	30.32	26.47	29.03

טבלה 12: השוואת תוצאות של שלושה אלגוריתמים שונים לביצוע ניקוי רעשים וספור רזולוציה (יחס הגדלה של 2 וסטיית תקן של 10)

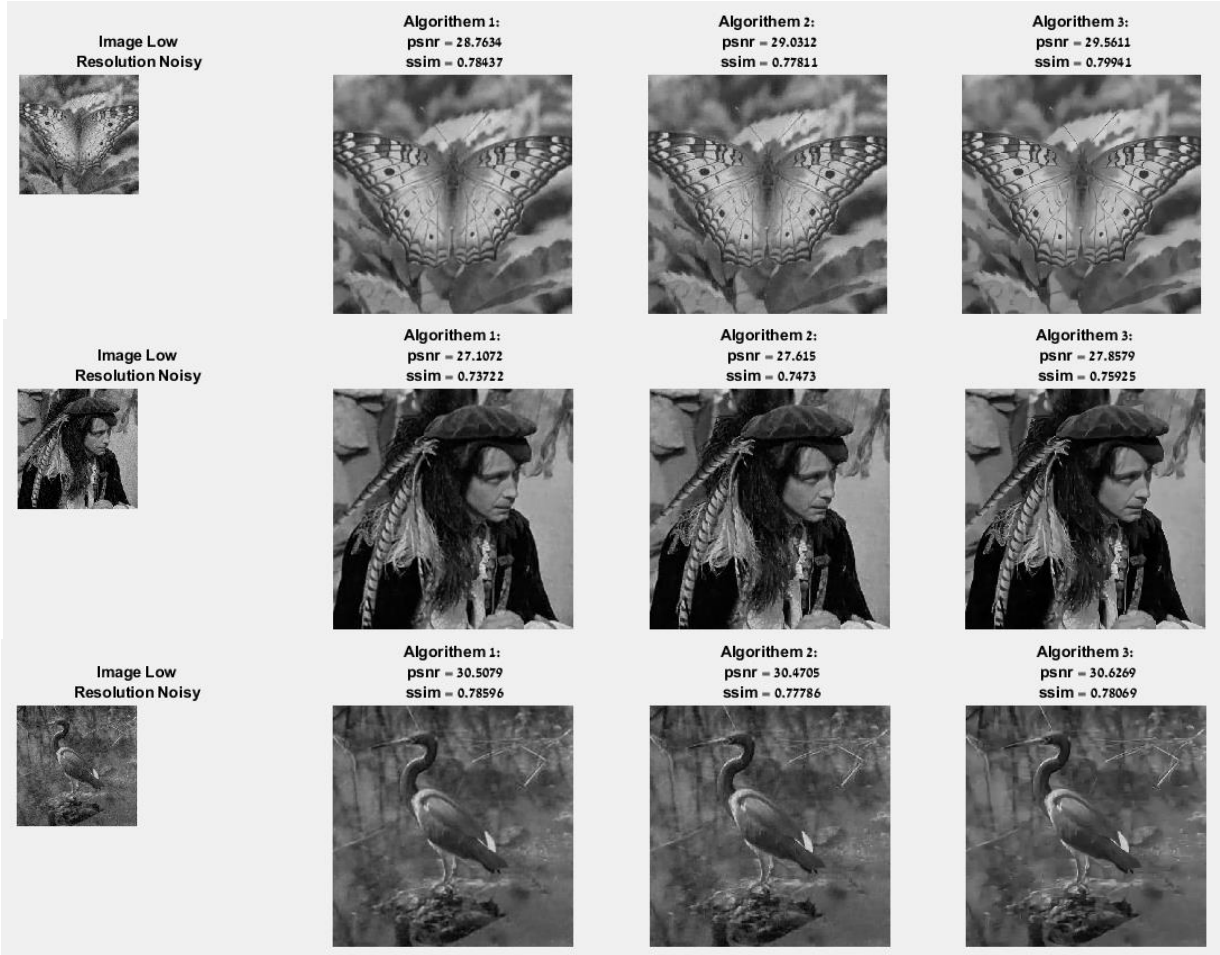
מסקנות

ניתן להסיק מכאן מסקנה ברורה מאוד ויחד עם זאת מאוד מעניינת. אלגוריתם 2 טוב יותר מאלגוריתם אחד מכיוון שאנו מחליפים את אינטרפולציית Bicubic בביצוע Super Resolution.

אבל אם נשים לב, אלגוריתם 3 טוב יותר מאלגוריתם 2. כלומר רשת אחת שמבצעת את שתי הפעולות טובה יותר משתי רשתות אחת אחרי השנייה.

את התוצאות הויזואליות של כמה תמונות נראה דווקא על רעש עם סטיית תקן של 10, מכיוון שמדובר באותו עיקרון בדיוק רק שנבחין יותר בהבדלים.

הצגה ויזואלית של התוצאות עבור יחס הגדלה 2 וסטיית תקן 10



איור 19 : תוצאות ויזואליות להשוואה בין שלושה אלגוריתמים שונים לביצוע ניקוי רעשים וסופר רזולוציה

3.2.10 בניית ממשק גרפי ב-Matlab לשימוש כללי במערכת

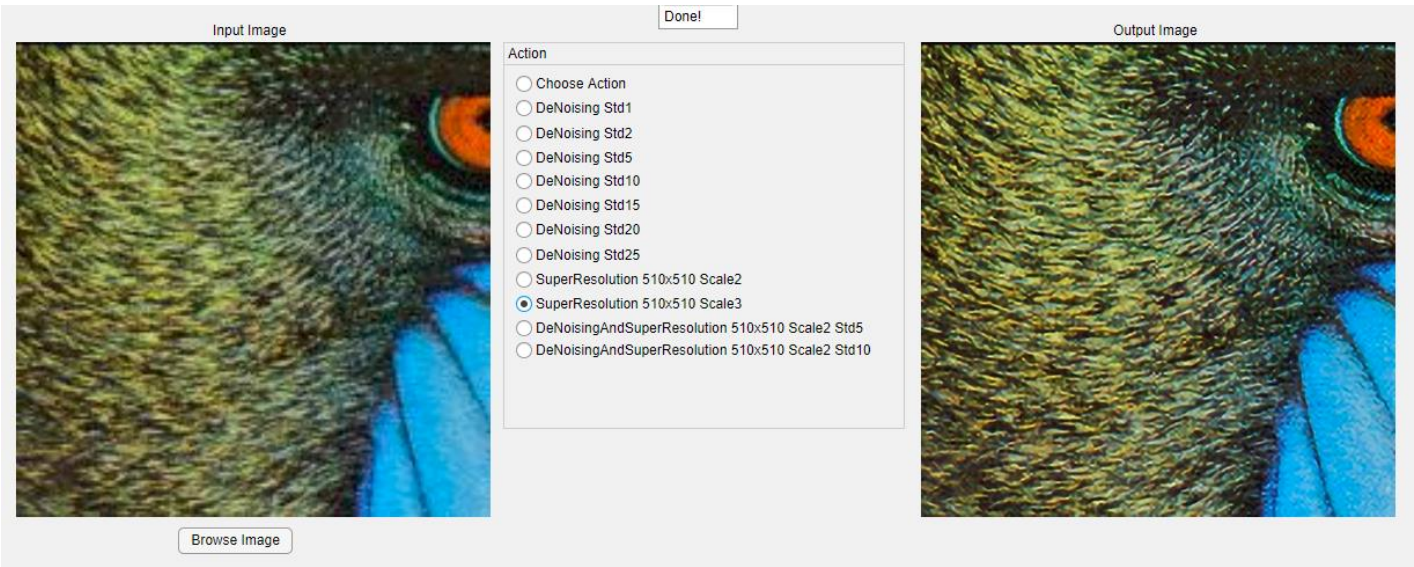
על מנת להנגיש את תוצאות המחקר ל"קהל הרחב" בלי שהמשתמש יבין באלגוריתמי עיבוד תמונה, למידה עמוקה או שפות תכנות כלשהן עליי ליצור ממשק גרפי שישמש כמעין אפליקציה למשתמש.

את האפליקציה אצור ב-Matlab ולכן נדרש Matlab במחשב על מנת להריץ את האפליקציה.
הערה: על מנת ליצור אפליקציה ב-Matlab שתתאים גם למחשבים ללא Matlab יש צור ברישיון עסקי של Matlab.

האפליקציה תכלול את כל האפשרויות הקיימות במחקר וכמובן שהרשתות שבהן יתבצע שימוש הן רק הרשתות שבסופו של דבר החלטנו שהן הטובות ביותר לביצוע הפעולה הדרושה.

הערה: במידה ויוכנסו למערכת תמונות שלא מתאימות לאותה הפעולה, המערכת תשנה את התמונה בכניסה.

נצרף תמונה להמחשת האפליקציה:



כפי שניתן לראות אין שום צורך בידע כלשהו בתכנות, פשוט בוחרים תמונה מהמחשב, בוחרים פעולה רצויה ותוך כמה שניות מתקבלת תמונת המוצא מהמערכת.

4. סיכום ודיון

4.1 עמידה בדרישות

נדרש בהצעת הפרויקט	בוצע בפרויקט
יצירת Dataset גדול לאימון הרשתות	בוצע – בפועל נדרשו כמה Dataset שונים, לעיתים קבצי mat ולעיתים תמונות בגדלים שונים בהתאם לפעולה הרצויה
מימוש מערכת מבוססת פונקציית כיווץ סקלרית	בוצע – בפועל נוספה מערכת סקלרית מנורמלת המתוארת לעיל וגם היא מומשה
אימון רשתות נוירונים עמוקות באמצעות ה-Dataset שהכנו לקבלת מערכת סקלרית	בוצע
בדיקת התוצאות המערכת הסקלרית לניקוי רעשים מתמונות בסטיית תקן קבועה	בוצע
תיקון הרשת הסקלרית עד לקבלת רשת סקלרית אופטימלית לעבודה	בוצע
תכנון מערכת מבוססת פונקציית כיוון וקטורית	בוצע
הרחבת הפתרון לביטול טשטוש	לא בוצע – במהלך המחקר בשיתוף המנחה הוחלט להתמקד עוד יותר במערכת הסופר רזולוציה ולוותר על ביטול הטשטוש
הרחבת הפתרון לסופר רזולוציה	בוצע – לא בוצע רק הרחבת הפתרון אלא מעבר לאלגוריתם חדש שעובד בממד התמונה על מנת להתנסות בתחום נוסף של עיבוד תמונה בלמידה עמוקה
יצירת ממשק משתמש לטעינת תמונות, בחירת הפעולה הרצויה וקבלת התמונה המתוקנת	בוצע
השוואת ביצועים מול שיטות מובילות בעולם	בוצע בניקוי הרעשים והוצע לבצע זאת עבור הסופר רזולוציה בעתיד לאחר הרחבת המערכת לגודל תמונה לא קבוע
כתיבת ספר פרויקט + הכנת מצגת	בוצע
	הרחבה לתמונות צבעוניות עבור רשת הסופר רזולוציה

איור 20 : טבלת עמידה בדרישות

4.2 הצעות להרחבות עתידיות של המחקר

המחקר נפתח להמשך בהנחיית ד״ר אמיר אדלר גם בשנה הקרובה, כעת אתאר את המלצותיי להרחבת העבודה:

- אימון ובניית רשת אחת לביצוע ניקוי רעשים עבור כלל סטיות התקן והשוואת המערכת לתוצאות המתקבלות בפרויקט זה.
- ביצוע סופר רזולוציה עם רשת שמקבלת בכניסה וקטורים (כפי שביצענו בניקוי רעשים) ולא עם רשת שמקבלת בכניסה תמונה, והשוואה בין התוצאות שיתקבלו לתוצאות בפרויקט זה. פעולה זו תאפשר להכניס למערכת תמונות בגודל לא קבוע (בניגוד למבוצע בפרויקט זה).
- לאחר המעבר לביצוע סופר רזולוציה על תמונות בגודל לא קבוע, להשוות את התוצאות המתקבלות על Datasets שונים לתוצאות המפורסמות במאמרים אחרים ברחבי האינטרנט העוסקים גם הם בסופר רזולוציה באמצעות למידה עמוקה.
- לאחר ביצוע המעברים שתוארו לעיל ניתן לאמן את הרשת לביצוע סופר רזולוציה על Dataset גדול יותר של תמונות ולבחון את ההשפעה של הגדלת ה-Dataset.
- ביצוע הפעולות שביצעתי בפרויקט רק עם פונקציות בסיס של התמרת DCT בגודל 16X16 (ניתן ליצור את פונקציות הבסיס בעזרת הנספח שצירפתי).
- הוספת אפשרות לתיקון פיקסלים חסרים: כאן מדובר ברעש מסוג אחר הקרוי רעש שיא או בשפה המסורתית salt & pepper. ברעש זה, פיקסלים משנים את ערכם לשחור או לבן מוחלט באופן אקראי. גם כאן האלגוריתמים הקיימים שלא מתבססים על למידה עמוקה מביאים תוצאות סבירות בתיקון התמונה, אני מציע להשתמש בלמידה עמוקה לצורך טיפול בקלקול זה.
- כמובן שאת כל התוספות יש להוסיף גם לממשק הגרפי ב-Matlab.

5. סימוכין

[1] A Discriminative Approach for Wavelet Denoising, by Yavov Hel-Or and Doron Shaked.
Available: https://users.soe.ucsc.edu/~milanfar/IS08/HelOrPaper.pdf

[2] Image Super-Resolution Using Deep Convolutional Networks, by Chao Dong, Chen Change Loy, Member, IEEE, Kaiming He, Member, IEEE, and Xiaoou Tang, Fellow, IEEE, 31 July 2015.
Available: https://arxiv.org/pdf/1501.00092.pdf

[3] Discrete Cosine Transform, MathWorks.
Available: https://www.mathworks.com/help/images/discrete-cosine-transform.html

[4] Fast and Accurate Image Super-Resolution with Deep Laplacian Pyramid Networks. by Wei-Sheng Lai, Jia-Bin Huang Narendra Ahuja and Ming-Hsuan Yang.
Available: http://vllab.ucmerced.edu/wlai24/LapSRN/#download

[5] Dataset of standard 512x512 grayscale test image.
Available: http://decsai.ugr.es/cvg/CG/base.htm

[6] DIV2K dataset: DIVerse 2K resolution high quality images as used for the challenges @ NTIRE (CVPR 2017 and CVPR 2018) and @ PIRM (ECCV 2018), by Radu Timofte, Eirikur Agustsson, Shuhang Gu, Jiqing Wu, Andrey Ignatov, Luc Van Gool.
Available: https://data.vision.ee.ethz.ch/cvl/DIV2K/

[7] Fundamentals of Digital Image Processing, by Roger L. Easton, Jr, 22 November 2010.
Available: https://www.cis.rit.edu/class/simg361/Notes_11222010.pdf

[8] 20 Free Image Datasets for Computer Vision, by Meiryum Ali, 22 May 2019.
Available: https://lionbridge.ai/datasets/20-best-image-datasets-for-computer-vision/

6. נספחים

6.1 נספח א – הוספת רעש גאוסי עם תוחלת 0 וסטיית תקן רצויה לתמונה

שורת הקוד שמבצעת פעולה זו היא:

```
ImageLRnoisy = double(ImageLR) + randn(DIM1_ImageLR,DIM2_ImageLR)
* sigma
```

כאשר sigma מציין את סטיית התקן הרצויה.

נבצע בדיקה של מנת לבחון את הרעש ולוודא שאכן התווסף לתמונה רעש בסטיית תקן של sigma.

נבודד את הרעש:

```
Noise = ImageLRnoisy - double(ImageLR);
```

נחשב את הממוצע של הרעש בצורה הבאה ונוודא שהיא אכן 0 (תוחלת 0):

```
avgNoise = mean(Noise(:));
```

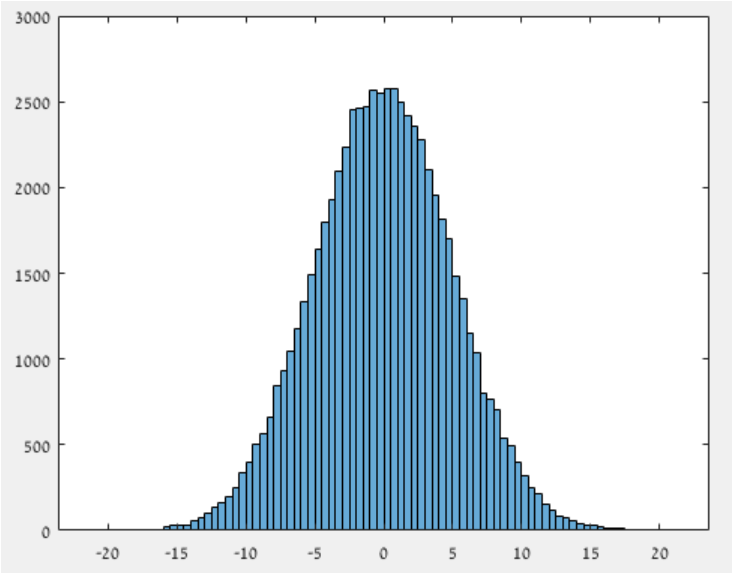
נחשב את השונות:

```
varNoise = var(Noise(:));
```

הערה: בבדיקה כתבו במקום sigma את המספר 5.
נראה שאכן מקבלים שונות של 25 ולכן סטיית התקן היא באמת 5 כמו שרצינו.

ניתן גם לראות את ההתפלגות בגרף:

```
histogram(Noise)
```



איור 21: גרף התפלגות רעש גאוסי עם תוחלת 0 וסטיית תקן 5 לשם הוכחת נכונות הפקודה להוספת רעש לתמונה ב-Matlab

אמצע הפעמון הוא 0 שזו התוחלת ורוחב כל צד בפעמון הוא קצת יותר מ-3sigma, קצת יותר מ-15.

הערה:

על מנת להפוך את התמונה המורעשת לתמונה אמיתית נשנה כל ערך שגדול מ-255 ל-255 וכל ערך שקטן מ-0 ל-0, לאחר מכן נעגל את הערכים. כל זה מבוצע בשלוש השורות הבאות:

```
ImageLRnoisy(ImageLRnoisy<0) = 0;
ImageLRnoisy(ImageLRnoisy>255) = 255;
ImageLRnoisy = uint8(ImageLRnoisy);
```

6.2 נספח ב – חישוב פונקציות הבסיס של התמרת DCT

הקוד המחשב את 81 המסננים ושומר אותם לקובץ mat מתואר להלן:

```
clc; clear; close;

N = 9; % NxN DCT
K = N^2; % number of sub-bands
B = zeros(N^2); % DCT basis functions image
H = zeros(N^2,N,N); % 81x9x9 matrix containing all basis functions
H_trans = H; % % 81x9x9 matrix containing all transposed basis functions

T=dctmtx(N); % 1D DCT Matrix
k=0;
% Build DCT Basis
for n2=1:N
    for n1=1:N
        k = k+1;
        I = zeros(N);
        I(n1,n2) = 1;
        dct_basis_function = single(T'*I*T)/N;
        B((n1-1)*N+1:(n1-1)*N+N,(n2-1)*N+1:(n2-1)*N+N) = dct_basis_function;
        H(k,:,:) = dct_basis_function;
        H_trans(k,:,:) = fliplr(flipud(dct_basis_function));
    end
end
```

איור 22: הקוד ב-Matlab למציאת 81 המסננים להתמרה ולהתמרה ההפוכה, כלומר 81 פונקציות הבסיס של התמרת ה-DCT.ההתמרה ההפוכה IDCT

בקוד לעיל מייצרים 81 מסננים בגודל 9x9 עבור התמרת DCT, ובשורת הקוד האחרונה בכל לולאה משתמשים בפקודה fliplr על מנת לקבל את 81 המסננים עבור התמרת IDCT.

הנוסחאות בקוד הינן הנוסחאות מתוך [3].

במידה ומעוניינים לעבור לכמות אחרת של מסננים כפי שהצעתי בהצעות להמשך המחקר, פשוט יש לשנות את N לגודל המסנן הרצוי.

7. קורות חיים – עמית קדוש

ארץ לידה: ישראל
כתובת: בית שאן, נוף הגלבוע, האירוס 5א'
פלאפון: 052-5949371
דוא"ל: amitkadosh7@gmail.com

תמצית:

- מהנדס חשמל ואלקטרוניקה, מצטיין דיקן במשך שלוש שנים, בעל ממוצע 89.59.
כיום חייל צה"ל בשירות חובה בחיל האוויר ביחידת כטמ"מ מודיעין ול"א בענף מיוחדים.
- פרויקט גמר מחקרי מורחב בנושא "שיפור איכות תמונות באמצעות אלגוריתמי Deep Learning".
- בעל ניסיון רב בעמידה מול קהל והרצאות, יכולת להעמקה וללמידה עצמית ויכולת עבודה עצמאית ובצוות. אחראי, יסודי ומקצועי, בעל סדר וארגון, משימתי וממוקד, בעל יכולת קידום משימות בצורה מקצועית ובעל יחסי אנוש טובים.
- שליטה בשפות התכנות: C, C++, Python, Matlab, VHDL, Assembly8051, C51, Arduino.
- פיתוח בסביבות: Multisim, Visual studio, PyCharm, Spider, PyScripter, Matlab, Quartus, ModelSim, Arduino.

השכלה:

2016-2020: תואר ראשון B.Sc בהנדסת חשמל ואלקטרוניקה, התמחות בחומרה ותוכנה, מכללת אורט בראודה בכרמיאל. **מצטיין דיקן במשך שלוש שנים, בעל ממוצע 89.59. מצטיין ראש מחלקה בשנה השנייה והשלישית ללימודים מטעם חברת KLA.**

2004-2016: בוגר 12 שנות לימוד בתיכון "אורט פסגות - בית שאן" במגמת אלקטרו-פיזיקה עם הרחבה נוספת במערכות תקשורת. בגרות מלאה עם סה"כ 38 יחידות לימוד **בממוצע 112.**
תעודת הצטיינות מטעם תוכנית "מצטייני פריפריה" המוענקת לחמשת התלמידים בעלי ממוצע הבגרות הגבוה ביותר.

ניסיון תעסוקתי:

2020-היום: כיום חייל צה"ל בשירות חובה בחיל האוויר ביחידת כטמ"מ מודיעין ול"א בענף מיוחדים – מסווג.

2019-2020: מדריך בקורס "מבוא לתכנות בשפת C" בתוכנית "מגשימים" - תכנית הסייבר הלאומית של ישראל, מדריך רובוטיקה בתוכנית "קדימה מדע" – בניית רובוטים ע"י שימוש בערכת EV3 ובנוסף חונך אישי לסטודנט שנה א' בהנדסת חשמל ואלקטרוניקה במסגרת תוכנית "פרח לסטודנטים".

2017-2018: מתרגל לבגרויות באלקטרוניקה, מתמטיקה ופיזיקה ברמת 5 יחידות לימוד בתיכון "אורט פסגות" בבית שאן. בנוסף, מורה למתמטיקה ברמת 5 יחידות לימוד בתוכנית "נחשון" הכפופה למשרד החינוך.

פרויקטים:

- פרויקט גמר מחקרי מורחב בנושא "שיפור איכות תמונות באמצעות אלגוריתמי deep learning". הפרויקט כולל מימוש אלגוריתמי deep learning בשתי שפות תכנות שונות – Matlab ו-Python, וכתיבת ספר פרויקט.
- זכיה במקום השני בהאקתון "רפואה והנדסה" בזכות תכנון ופיתוח אב טיפוס למערכת לניטור מטופלים בבתי חולים – מערכת בשם MIS.
- בניית רובוטים שונים במסגרת הדרכה ברובוטיקה – רובוט שעולה מדרגות, רובוט שאוסף חפצים, מכונית על שלט ועוד.
- מימוש תוכנה להצגת gif במסכי VGA תוך שימוש בשפת VHDL.
- תכנון "עגלה אוטונומית" שמטרתה לזהות מטרה משדרת ולהגיע אליה תוך התחמקות ממכשולים בדרך. הפרויקט מבוסס Arduino.
- פרויקט במיקרו בקר 8051 בשפת תכנות C51 – בקרת טמפרטורה והדלקת מאוורר לקירור או לוח קרמי לחימום בהתאם לצורך.
- פיענוח פרטים מתוך טפסים סרוקים של רשות המיסים ע"י שימוש ב-deep learning בשפת Matlab. כולל GUI ב-Matlab.
- פרויקט תכנון בשפת C++ לקריאה וכתיבה של קובץ בינארי בפורמט bmp ופרויקטי תכנות נוספים כגון מימוש TAKI בשפת C++, מימוש תוכנה ליצירת GIF ותוכנה שפותרת סודוקו בכל הרמות בשפת C.

כלים וטכנולוגיות:

שפות תכנות: C, C++, Python, Matlab, VHDL, Assembly8051, C51.

חומרה: FPGA, Arduino, Microcontrollers 8051, ARM Microcontroller, EV3.

מערכות הפעלה: Windows, Linux.

סביבות עבודה: Multisim, Visual studio, PyCharm, Spider, PyScripter, Matlab, Quartus, ModelSim, Arduino.

פעילות קהילתית:

2017-2018: התנדבות בארגון "פרח"- חונכות לילדים הזקוקים לתמיכה לימודית ורגשית.

2015-2016: התנדבות קהילתית בספרייה העירונית בבית שאן, הכוללת עזרה לימודית לתלמידים ואחריות על ניהול שוטף של הספרייה. השתייכות לעמותת "אחריי! לצה"ל- נוער מוביל שינוי" תוך שיתוף פעולה עם ארגון "לתת"

שפות: עברית: רמת שפת אם. אנגלית: קריאה, כתיבה ודיבור ברמה טובה.

*המלצות יינתנו לפי דרישה