Дигитално процесирање на слика

Тема: Image Super-Resolution

Марија Аврамовска 221044

Ева Беќарска 221153

1. Вовед

Во дигиталната обработка на слики, подобрувањето на резолуцијата е важна задача која наоѓа примена во различни области како што се медицинската дијагностика, сателитските снимки, безбедноста и мултимедијалните апликации. Image Super-Resolution (ISR) е процес на реконструкција на висококвалитетна слика (HR) од нејзината нискорезолуциска верзија (LR). Овој процес може значително да ја подобри визуелната содржина и деталите во сликите.

2. Традиционални методи за подобрување на резолуцијата

Традиционалните методи за подобрување на резолуцијата на сликите главно се базираат на интерполација:

- **Nearest Neighbor Interpolation**: Наједноставен метод каде новиот пиксел ја добива вредноста на најблискиот соседен пиксел.
- **Bilinear Interpolation**: Користи линеарна интерполација помеѓу соседните пиксели во двете насоки (x и y).
- **Bicubic Interpolation**: Подобрен метод кој користи 16 соседни пиксели за да ја пресмета новата вредност, обезбедувајќи посмазнети резултати.
- **Lanczos Resampling**: Користи синусна функција за подобрување на квалитетот, но е побавен во споредба со другите методи.

Овие методи даваат солидни резултати, но не можат да ги надминат ограничувањата на оригиналните слики во однос на детали и текстури.

3. Длабоки невронски мрежи за ISR

Современите пристапи користат длабоки конволуциски невронски мрежи (CNN) за реконструкција на висококвалитетни слики. Некои од најпознатите методи вклучуваат:

- **SRCNN (Super-Resolution Convolutional Neural Network)**: Едноставен модел кој користи три конволуциски слоеви за подобрување на резолуцијата.
- **VDSR (Very Deep Super-Resolution Network)**: Користи многу подлабока невронска мрежа (20+ слоеви) за да постигне подобри резултати.
- **ESRGAN (Enhanced Super-Resolution GAN)**: Генеративна мрежа која создава фотореалистични детали во супер-разрешени слики.
- **RCAN (Residual Channel Attention Network)**: Модел кој користи внимание на каналите за подобрување на реконструираните слики.

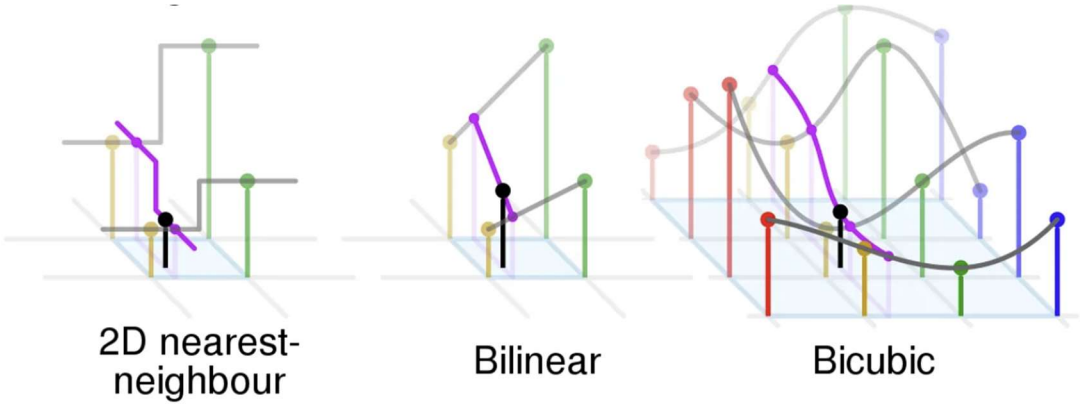
Во овој проект, ќе се имплементира **SRCNN** како основен пристап во ISR.

4. Имплементација на демо-апликација

Демо-апликацијата ќе вклучува:

- Учитање на слика со ниска резолуција
- Подобрување на резолуцијата со Bicubic Interpolation
- Користење на SRCNN за дополнително подобрување

4.1 Bicubic Interpolation



Bicubic интерполацијата се користи за почетно зголемување на резолуцијата на сликата. Оваа техника обезбедува поквалитетни резултати во споредба со Bilinear и Nearest Neighbor методите. Bicubic интерполацијата ги зема предвид вредностите на интензитетот на повеќе соседни пиксели за да создаде помазна крива со користење дополнителни информации за создавање нелинеарни функции кои ја намалуваат појавата на недосатоци добиени од билинеарниот метод.

Во 1d:

Bilinear користи 2 пиксели за екстраполирање на вредноста на пикселите

Bicubic користи 4 пиксели за екстраполирање на вредноста на пикселите

Во 2d:

Билинеар користи 2 * 2 = 4 пиксели за екстраполирање на вредноста на пикселите

Bicubic користи 4 * 4 = 16 пиксели за екстраполирање на вредноста на пикселите

Оттука, Bicubic е поскап од билинерната интерполација за зголемување на сликите бидејќи користи пресметки на повеќе вредности како средни вредности

Формула:

$$p(\text{unknown}) = p1q1 + p2q2 + p3q3 + p4q4$$

каде што p2 и p3 се точките кои се најблиску до непознатата точка ,а p1 и p4 се другите точки до p2 и p3 соодветно.

Единственото нешто што се менува е основната функција

$q1 = (-t^3 + 2t^2 — t) / 2$

$q2 = (3t^3 — 5t^2 + 2) / 2$

$q3 = (-3t^3 + 4t + t) / 2$

$q4 = (t^3 — t^2) / 2$

Можеби ќе треба да интерполираме вредности на повеќе точки или пиксели пред конечно да ја интерполираме вредноста на пикселот што го посакуваме.

Примена на Bicubic интерполација на слика:

При примена на Bicubic интерполација на 2d слики, ќе ги разгледаме следните случаи:

I. Пикселот се наоѓа на истата хоризонтална или вертикална линија со познатите пиксели
II. Пикселот не се наоѓа на истата хоризонтална или вертикална линија со познатите пиксели
III. Случаи како аголни и рабни пиксели за кои немаме доволно информации за интерполирање на нивните вредности

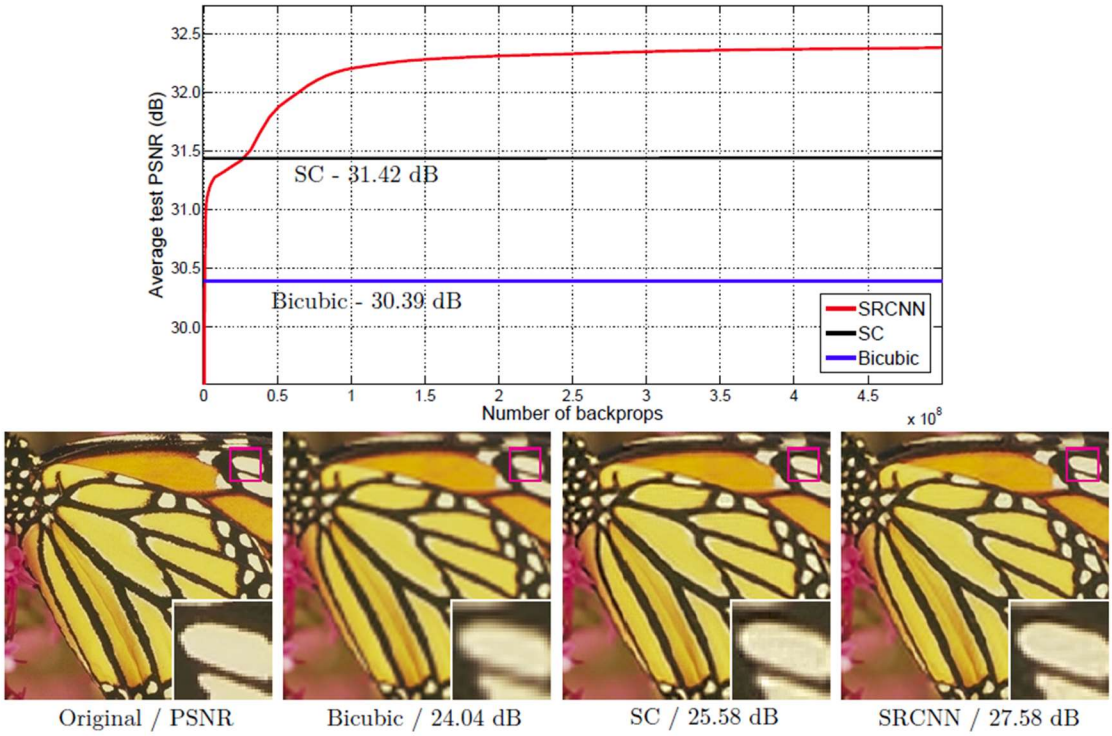
Случајот 1 може да се реши со едно користење на формулата, бидејќи имаме само отстапување по x или у оската, т.е. еднодимензионално

За случајот 2 ќе мора да се извршат повеќе интерполации (и/или одржување табела за оптимизирање на пресметката) за да се интерполира вредноста на саканиот пиксел

И за случај 3, тој може да се реши со примена на NN или Билинеарни интерполации или други методи.

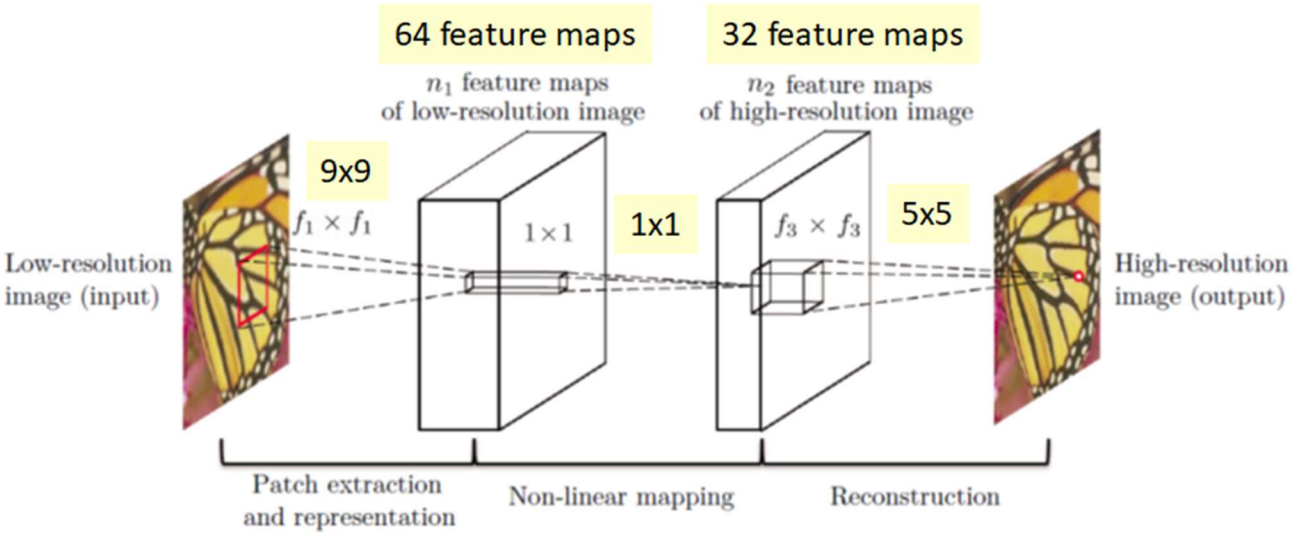
4.2 SRCNN

Во длабокот учење или конволутивната невронска мрежа (CNN), обично се користи CNN за класификација на слики. Во SRCNN, истата се користи за супер резолуција на една слика (SR) што е вообичаен проблем во компјутерската визија. Накратко, со подобар SR пристап, може да добиеме подобар квалитете на поголема слика иако првично имаме само мала слика.



Во горенаведената слика може да приметиме, со користење SRCNN, PSNR од 27.58dB, што е многу подобро од класичното бикубично кодирање, кое не се базира на длабоко учење.

Во SRCNN моделот, мрежата технички не е длабока и таа е составена од три главни слоеви: patch extraction and representation, non-linear mapping, and reconstruction, како што е прикажано и во сликата подолу.



I. **Patch extraction and representation**:

Издвојува мали делови од сликата и ги репрезентира како карактеристики. Важно е да се знае дека инпутот со ниска резолуција прво се зголемува до сканата големина со помош на бикубна интерполација пред да се внесе во мрежата на SRCNN. Според тоа:

X: Основна слика со висока резолуција

Y: Бикубна нетестирана верзија на слика со ниска резолуција

Првиот слој извршува стандардна конверзија со Relu за да добие F1(Y).

$F1(Y)=max(0 , W1 * Y + B1)$

Големина на W1: c×f1×f1×n1

Големина на B1: n1

Каде што c е бројот на канали на сликата, f1 е големина на филтерот и n1 е бројот на филтери. B1 е n1-димензионален вектор на пристрасност кој се користи само за зголемување на степенот на слобода за 1.

Во овој случај: c=1,f1=9,n1=64.

II. **Non-linear mapping**:

Користи конволуциски слој за мапирање на влезните карактеристики на излезни карактеристики со повисока резолуција.Овој чекор се извршува после patch extraction and representation.

$F2(Y)=max(0 , W2 *F1(Y) + B2)$

Големина на W2: n1×1×1×n2

Големина на B2: n2

Претставува мапирање на n1-димензионален вектор во n2-димензионален вектор. Кога n1>n2, може да замислиме нешто слично на PCA, но на нелинеарен начин.

Во овој случај, n2=32.

Овој 1×1 всушност е исто така конволуција 1×1 предложена во Network In Network (NIN). Кај неа е предоложено воведување поголема нелинеарност за подобрување на прецизноста. Исто така е предложена во GoogLeNet за намалување на бројот на конекции. Во овој случај е користено за мапирање на вектори со ниска резолуција во вектори со висока резолуција.

III. **Reconstruction**:

Генерира финална слика со висока резолуција. После мапирањето треба да се реконструира сликата. Оттука повторно правиме конверзија

$F(Y) = W3 * F2(Y) + B3$

Големина на W3: n2×f3 ×f3×c

Големина на B3: c

Функција на губење

$$L(\Theta) = \frac{1}{n}\sum_{i=1}^{n}\|F(Y_i; \Theta - X_i\ \|^2$$

За супер резолуција, функцијата за загуба L е просекот на квадратниот корен на средната грешка (MSE) за примероците за обука (n), што е еден вид стандардна функција на загуба.

Споредба со најсовремени пристапи

91 слика за тренирање обезбедува приближно 24.800 подслики со чекор 14 и Гаусово заматување. Потребни се 3 дена за обука на GTX 770 GPU со 8×10^8 задни пропагации.

Различни скали помеѓу 2 и 4 биле тестирани

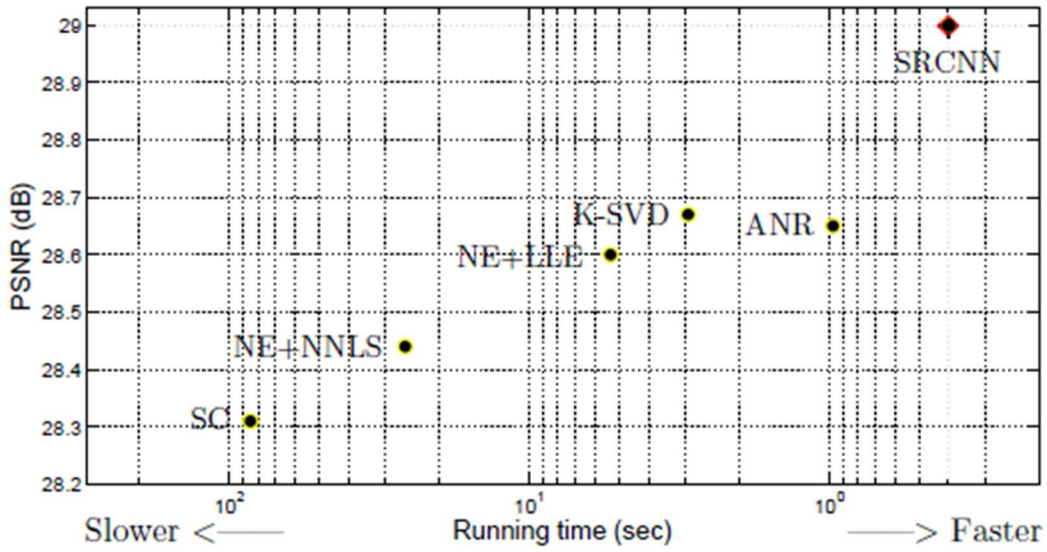
Set5 [2]		Bicubic		SC [26]		K-SVD [28]		NE+NNLS [2]		NE+LLE [4]		ANR [20]		SRCNN	
images	Scale	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
baby	2	37.07	-	-	-	38.25	7.0	38.00	68.6	38.33	13.6	**38.44**	2.1	38.30	0.38
bird	2	36.81	-	-	-	39.93	2.2	39.41	22.5	40.00	4.2	40.04	0.62	**40.64**	0.14
butterfly	2	27.43	-	-	-	30.65	1.8	30.03	16.6	30.38	3.3	30.48	0.50	**32.20**	0.10
head	2	34.86	-	-	-	35.59	2.1	35.48	19.2	35.63	3.8	**35.66**	0.57	35.64	0.13
woman	2	32.14	-	-	-	34.49	2.1	34.24	19.3	34.52	3.8	34.55	0.57	**34.94**	0.13
average	2	33.66	-	-	-	35.78	3.03	35.43	29.23	35.77	5.74	35.83	0.87	**36.34**	0.18
baby	3	33.91	-	34.29	76.0	35.08	3.3	34.77	28.3	35.06	6.0	**35.13**	1.3	35.01	0.38
bird	3	32.58	-	34.11	30.4	34.57	1.0	34.26	8.9	34.56	1.9	34.60	0.39	**34.91**	0.14
butterfly	3	24.04	-	25.58	26.8	25.94	0.81	25.61	7.0	25.75	1.4	25.90	0.31	**27.58**	0.10
head	3	32.88	-	33.17	21.3	33.56	1.0	33.45	8.2	33.60	1.7	**33.63**	0.35	33.55	0.13
woman	3	28.56	-	29.94	25.1	30.37	1.0	29.89	8.7	30.22	1.9	30.33	0.37	**30.92**	0.13
average	3	30.39	-	31.42	35.92	31.90	1.42	31.60	12.21	31.84	2.58	31.92	0.54	**32.39**	0.18
baby	4	31.78	-	-	-	33.06	2.4	32.81	16.2	32.99	3.6	**33.03**	0.85	32.98	0.38
bird	4	30.18	-	-	-	31.71	0.68	31.51	4.7	31.72	1.1	31.82	0.27	**31.98**	0.14
butterfly	4	22.10	-	-	-	23.57	0.50	23.30	3.8	23.38	0.90	23.52	0.24	**25.07**	0.10
head	4	31.59	-	-	-	32.21	0.68	32.10	4.5	32.24	1.1	**32.27**	0.27	32.19	0.13
woman	4	26.46	-	-	-	27.89	0.66	27.61	4.3	27.72	1.1	27.80	0.28	**28.21**	0.13
average	4	28.42	-	-	-	29.69	0.98	29.47	6.71	29.61	1.56	29.69	0.38	**30.09**	0.18

PSNR за Set15 податоци

Set14 [28]		Bicubic		SC [26]		K-SVD [28]		NE+NNLS [2]		NE+LLE [4]		ANR [20]		SRCNN	
images	scale	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
baboon	3	23.21	-	23.47	126.3	23.52	3.6	23.49	29.0	23.55	5.6	23.56	1.1	**23.60**	0.40
barbara	3	26.25	-	26.39	127.9	**26.76**	5.5	26.67	47.6	26.74	9.8	26.69	1.7	26.66	0.70
bridge	3	24.40	-	24.82	152.7	25.02	3.3	24.86	30.4	24.98	5.9	25.01	1.1	**25.07**	0.44
coastguard	3	26.55	-	27.02	35.6	27.15	1.3	27.00	11.6	27.07	2.6	27.08	0.45	**27.20**	0.17
comic	3	23.12	-	23.90	54.5	23.96	1.2	23.83	11.0	23.98	2.0	24.04	0.42	**24.39**	0.15
face	3	32.82	-	33.11	20.4	33.53	1.1	33.45	8.3	33.56	1.7	**33.62**	0.34	33.58	0.13
flowers	3	27.23	-	28.25	76.4	28.43	2.3	28.21	20.2	28.38	4.0	28.49	0.81	**28.97**	0.30
foreman	3	31.18	-	32.04	25.9	33.19	1.3	32.87	10.8	33.21	2.2	33.23	0.44	**33.35**	0.17
lenna	3	31.68	-	32.64	68.4	33.00	3.3	32.82	29.3	33.01	6.0	33.08	1.1	**33.39**	0.44
man	3	27.01	-	27.76	111.2	27.90	3.4	27.72	29.5	27.87	6.1	27.92	1.1	**28.18**	0.44
monarch	3	29.43	-	30.71	112.1	31.10	4.9	30.76	43.3	30.95	8.8	31.09	1.6	**32.39**	0.66
pepper	3	32.39	-	33.32	66.3	34.07	3.3	33.56	28.9	33.80	6.6	33.82	1.1	**34.35**	0.44
ppt3	3	23.71	-	24.98	96.1	25.23	4.0	24.81	36.0	24.94	7.8	25.03	1.4	**26.02**	0.58
zebra	3	26.63	-	27.95	114.4	28.49	2.9	28.12	26.3	28.31	5.5	28.43	1.0	**28.87**	0.38
average	3	27.54	-	28.31	84.88	28.67	2.95	28.44	25.87	28.60	5.35	28.65	0.97	**29.00**	0.39

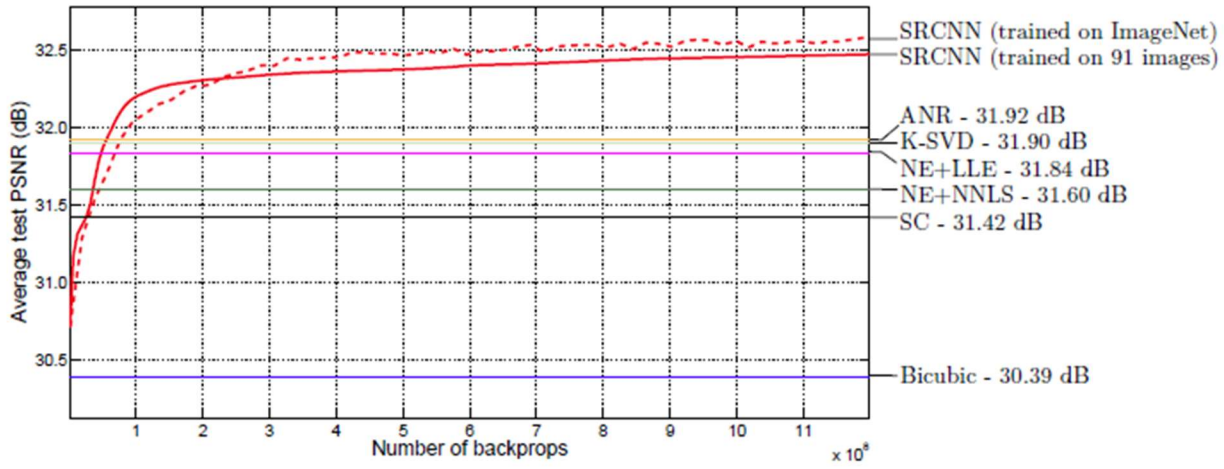
PSNR за Set14 податоци

SRCNN го држи највисокиот просек за PSNR



Подобриот,побрзиот, и моделот со повисок и подобар квалитет. Затоа SRCNN е во горниот десен раб што покажува дека има најдобри перформанси.

Истражување



Доколку тренирање со SRCNN се користат 395,909 слики кои се делумно од базата на обука за откривање ImageNet на ILSVRC 2013 година, резултатот е подобар од само тренирање на 91 слика.

Set5 [2]	$n_1 = 128,\ n_2 = 64$		$n_1 = 64,\ n_2 = 32$		$n_1 = 32,\ n_2 = 16$	
images	PSNR	Time	PSNR	Time	PSNR	Time
	32.60	0.60	32.52	0.18	32.26	0.05

Колку се поголеми n1 и n2, толку е поголем PSNR. Затоа е неромално со користење на повеќе филтри, да се добијат подобри резулати.

4.3 Тестирање на моделот

Во продолжение ќе се објасни кодот за применување на SRCNN моделот и резултатите добиени од него.

```
import os
import cv2
import numpy as np
from tqdm import tqdm
from tensorflow.keras.preprocessing.image import img_to_array
```

Најпрво се вчитуваат потребните библиотеки како што е os за работа со датотеки и директориуми, cv2 за користење OpenCV библиотека за обработка на слики, numpy за работа со нумерички податоци и сл.

```
def sorted_alphanumeric(data):
    convert = lambda text: int(text) if text.isdigit() else text.lower()
    alphanum_key = lambda key: [convert(c) for c in re.split('([0-
9]+)',key)]
    return sorted(data,key = alphanum_key)
```

Оваа функција ги сортира имињата на датотеките со природно подредување на бројките, бидејќи во податочнот множество сликите се именувани со бројка (пр. 1.png, 2.png, ..., 10.png)

```
SIZE = 256
high_img = []
path = '/content/dateset/Raw Data/high_res'
files = os.listdir(path)
files = sorted_alphanumeric(files)
for i in tqdm(files):
    if not i.endswith('.png'):
        continue
    else:
        img = cv2.imread(path + '/'+i,1)
        # open cv reads images in BGR format so we have to convert it to
RGB
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        #resizing image
        img = cv2.resize(img, (SIZE, SIZE))
        img = img.astype('float32') / 255.0
        high_img.append(img_to_array(img))
```

За вчитување и обработка на слики се дефинирани големината на сликите, односно во овој случај 256, листа за складирање(во овој дел користиме високорезолуциски слики, но се обработуваат подолу и нискорезолуциски слики), ја дефинираме патеката до директориумот и листа на имињата на датотеките во диреткриумот, сортинрани алфанумерички.

Во функцијата, во циклусот, секоја слика се вчитува, конверира од BGR во RGB, се променува на 256x256 пиксели, нормализира (вредности од 0 до 1) и се додава во листата high_img.

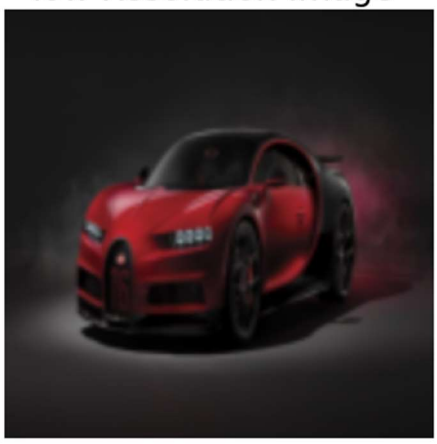
Во продолжение се бираат случајно 4 парови на нискорезолуциски и високорезолуциски слики може да се воочи разликата меѓу оригиналните и низкорезолуциските слики пред да ги подобри со SRCNN

High Resolution Imge

low Resolution Image

High Resolution Imge

low Resolution Image



High Resolution Imge

low Resolution Image

Следен чекор е подлеба на податоците за тренинирање,валидација и тест сетови

```
split_train = int(len(high_img) * 0.7)
split_val = int(len(high_img) * 0.85)
split_test = len(high_img)
train_high_image = high_img[:split_train]
validation_high_image = high_img[split_train:split_val]
test_high_image = high_img[split_val:split_test]
```

- **split_train**: Индекс кој означува 70% од податоците за тренинг.
- **split_val**: Индекс кој означува 85% од податоците (70% тренинг + 15% валидација).

Податоците се делат на:

- Сет за тренирање: Првите 70% од сликите

- Валидационен сет: Следните 15% од сликите

- Сет за тестирање:Последните 15% од сликите

```
from keras import layers
from tensorflow.keras.utils import plot_model


#SRCNN=tf.keras.models.Sequential([tf.keras.layers.Conv2D(64,9,padding='sa
me',activation='relu'),
 #      tf.keras.layers.Conv2D(64,1,padding='same',activation='relu'),
 #      tf.keras.layers.Conv2D(3,5,padding='same',activation='relu')])

input_img=Input(shape=(256,256,3))
l1=tf.keras.layers.Conv2D(64,9,padding='same',activation='relu')
(input_img)
l2=tf.keras.layers.Conv2D(32,1,padding='same',activation='relu') (l1)
l3=tf.keras.layers.Conv2D(3,5,padding='same',activation='relu') (l2)


SRCNN=Model(input_img,l3)
```

```
def pixel_mse_loss(x,y):
    return tf.reduce_mean( (x - y) ** 2 )
SRCNN.compile(optimizer=tf.keras.optimizers.Adam(0.001),loss=pixel_mse_los
s)
SRCNN.summary()
plot_model(SRCNN, to_file ='super_res.png',show_shapes=True)
```

Наредно се дефинира SRCNN моделот. Се користи Conv2D за 2D конволуциони слоеви.

Моделот се состои од три кнволуциони слоеви:

1) Прв слој: 64 филтри со големина 9x9, ReLU активација
2) Втор слој: 32 филтри со големина 1x1, ReLU активација
3) Трет слој: 3 филтри со големина 5x5, линеарна активација

Моделот се компајлира со Adam оптимизатор и MSE (Mean Squared Error) загуба.

```
SRCNN.fit(train_low_image, train_high_image, epochs = 20, batch_size = 1,
          validation_data = (validation_low_image,validation_high_image))
```

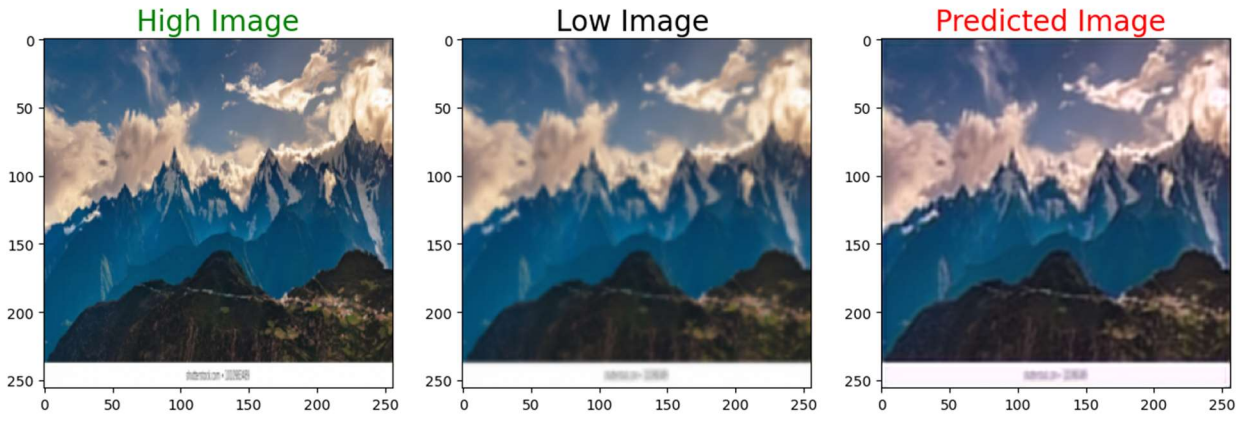
Се тренира со 50 епохи и големина на батч од 1

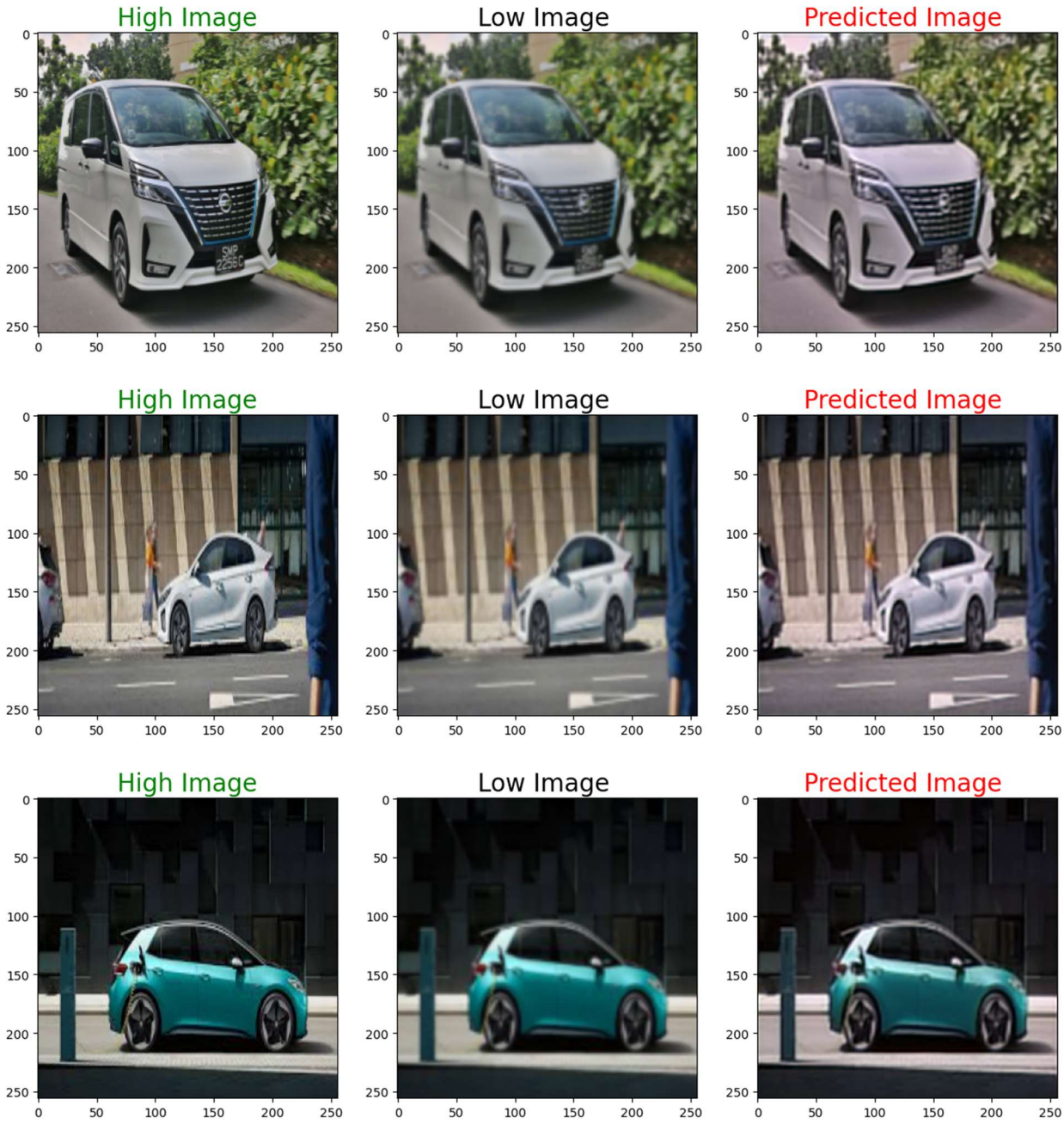
```
for i in range(16,25):

    predicted = np.clip(SRCNN.predict(test_low_image[i].reshape(1,SIZE,
SIZE,3)),0.0,1.0).reshape(SIZE, SIZE,3)
    plot_images(test_high_image[i],test_low_image[i],predicted)
    print('PSNR',PSNR(test_high_image[i],predicted),'dB',
"SSIM",tf.image.ssim(test_high_image[i],predicted,max_val=1))
```

За тестирање на моделот, се користи предвидувње на високорезолуциски слики од нискорезолуциски слики

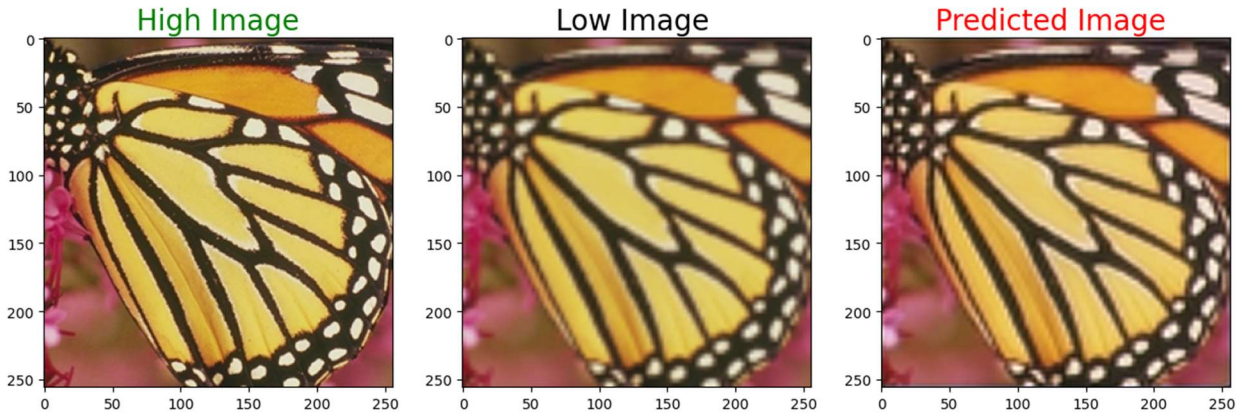
```
for i in range(16,25):

    predicted = np.clip(SRCNN.predict(test_low_image[i].reshape(1,SIZE,
SIZE,3)),0.0,1.0).reshape(SIZE, SIZE,3)
    plot_images(test_high_image[i],test_low_image[i],predicted)
    print('PSNR',PSNR(test_high_image[i],predicted),'dB',
"SSIM",tf.image.ssim(test_high_image[i],predicted,max_val=1))
```

се прикажуваат 9 слики од тест примерот, каде што првата колона е вискорезулуцискта слика, втората колона е нискорезолуциска и третата е слики генерирани од SRCNN моделот. Ке приказеме 4 примери:

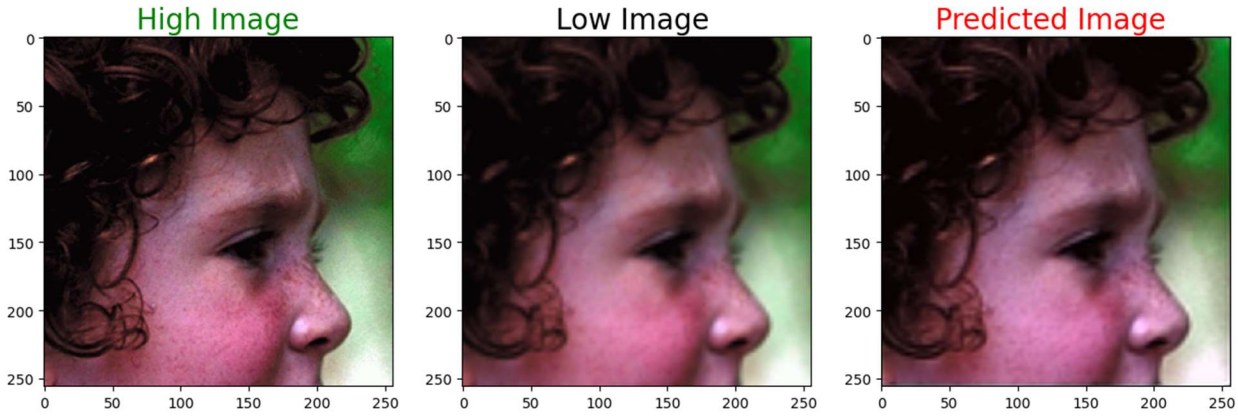




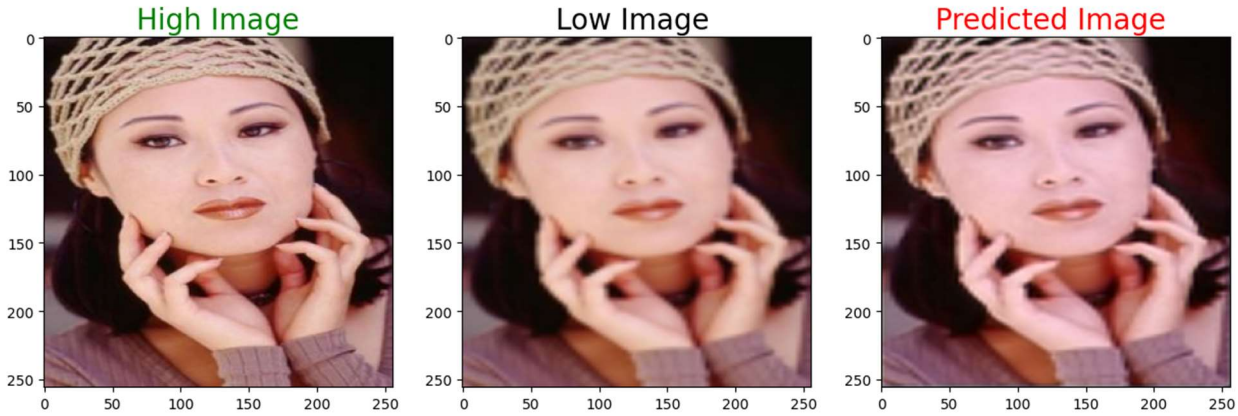
Во наредниот дел од кодот повторно се користи SRCNN моделот на друго податочно множество од слики, но дополнително премсетува PSNR(мерење на разликата меѓу оригиналната и предвидената слика) и SSIM(мерење на структурната сличност меѓу двете слики) за мерење на квалитетот. Еве пример од добиените резултати:



PSNR tf.Tensor(21.158062, shape=(), dtype=float32) dB SSIM tf.Tensor(0.7595427, shape=(), dtype=float32)



PSNR tf.Tensor(29.152067, shape=(), dtype=float32) dB SSIM tf.Tensor(0.8038142, shape=(), dtype=float32)



PSNR tf.Tensor(25.659296, shape=(), dtype=float32) dB SSIM tf.Tensor(0.871438, shape=(), dtype=float32)

5. Заклучок

ISR е важно поле кое се развива брзо благодарение на длабокото учење. SRCNN е едноставен, но ефикасен модел кој покажува значителни подобрувања во споредба со традиционалните методи. Понатамошните подобрувања во оваа област вклучуваат користење на GAN-мрежи, внимателни механизми и хибридни модели за уште повисок квалитет на супер-разрешените слики.

Идните истражувања може да се насочат кон оптимизација на мрежната архитектура, подобрување на брзината на обработка и намалување на потребата од огромни податочни множества за тренинг. Освен тоа, развојот на нови техники за пост-обработка може дополнително да го зголеми квалитетот на реконструираните слики. Со напредокот во хардверските технологии и алгоритмите, супер-резолуцијата има потенцијал да се примени во различни индустрии, како што се медицинската дијагностика, сателитската анализа и обработката на видеа со висока резолуција.

6. Користена литература

1. Dong, C., Loy, C. C., He, K., & Tang, X. (2016). "Image Super-Resolution Using Deep Convolutional Networks". IEEE Transactions on Pattern Analysis and Machine Intelligence.
2. Lim, B., Son, S., Kim, H., Nah, S., & Mu Lee, K. (2017). "Enhanced Deep Residual Networks for Single Image Super-Resolution". CVPR Workshops.
3. Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., & Shi, W. (2017). "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network". CVPR.
4. Chao Dong , Chen Change Loy , Kaiming He and Xiaoou Tang."Learning a Deep Convolutional Network for Image Super-Resolution"