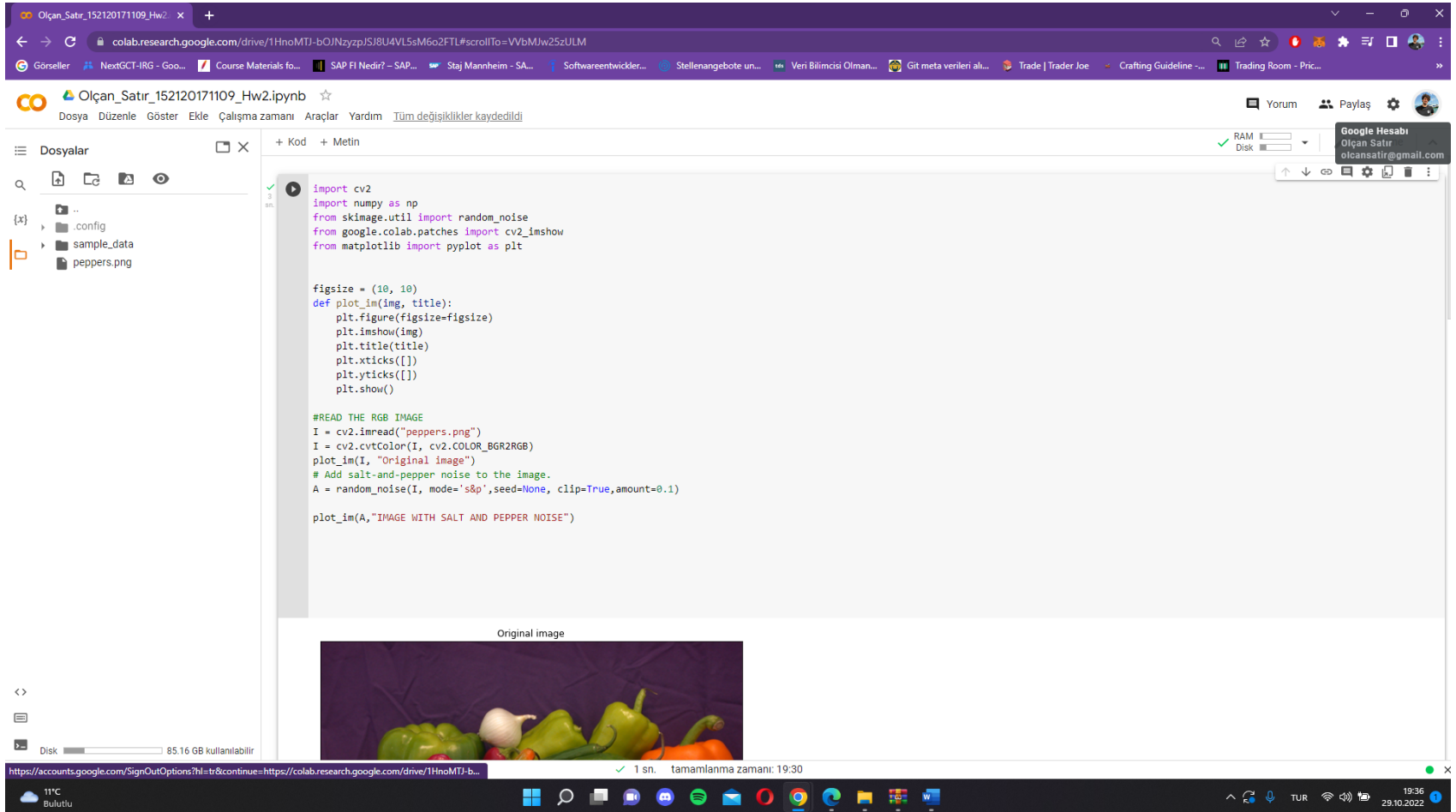


OLÇAN SATIR

152120171109

Deep Learning Homework-2 Report

Here we read the image and make the BGR image RGB. We print the original photo. Then we press it by applying salt and pepper.



Here we see the original image and salt and pepper noise image

Olçan_Satır_152120171109_Hw2 Homework02

colab.research.google.com/drive/1HnoMTJ-bOJNzyzJSJ8U4VL5sM6o2FTL#scrollTo=VvbmJw25zULM

Olçan_Satır_152120171109_Hw2.ipynb

Dosya Düzenle Göster Ekle Çalışma zamanı Araçlar Yardım Tüm değişiklikler kaydedildi

Dosyalar

- ..
- .config
- sample_data
- peppers.png

Original image





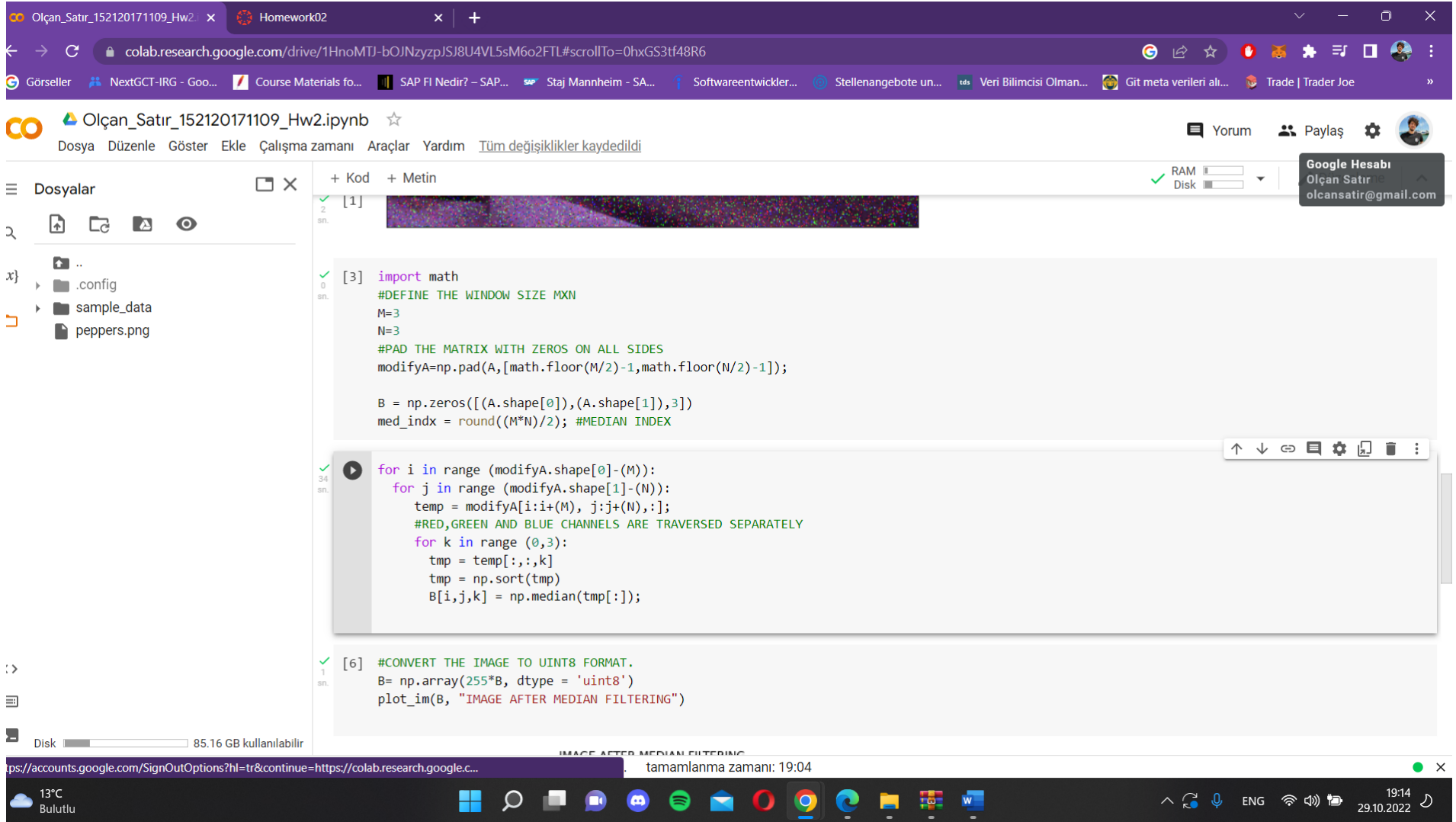
IMAGE WITH SALT AND PEPPER NOISE



13°C Bulutlu


19:12 29.10.2022

Here, zero padding is applied for the 3x3 filtering process. In this way, the kernel hovering over the picture does not create a size mismatch. The outputs obtained in the padding process were thrown into the zero matrix, and then the median filter was applied on the relevant values.



The screenshot displays a Google Colab notebook titled "Olçan_Satr_152120171109_Hw2.ipynb". The notebook is open in a web browser, showing the file explorer on the left with folders like ".config", "sample_data", and "peppers.png". The main area shows the code execution process for a median filter.

The code is as follows:

```
[1] 

[3] import math
#DEFINE THE WINDOW SIZE MXN
M=3
N=3
#PAD THE MATRIX WITH ZEROS ON ALL SIDES
modifyA=np.pad(A,[math.floor(M/2)-1,math.floor(N/2)-1]);

B = np.zeros([(A.shape[0]),(A.shape[1]),3])
med_indx = round((M*N)/2); #MEDIAN INDEX

for i in range (modifyA.shape[0]-(M)):
    for j in range (modifyA.shape[1]-(N)):
        temp = modifyA[i:i+(M), j:j+(N),:];
        #RED, GREEN AND BLUE CHANNELS ARE TRAVERSED SEPARATELY
        for k in range (0,3):
            tmp = temp[:, :,k]
            tmp = np.sort(tmp)
            B[i,j,k] = np.median(tmp[:]);

[6] #CONVERT THE IMAGE TO UINT8 FORMAT.
B= np.array(255*B, dtype = 'uint8')
plot_im(B, "IMAGE AFTER MEDIAN FILTERING")
```

The notebook interface includes a "Dosyalar" (Files) sidebar, a "Kod" (Code) tab, and a "Metin" (Text) tab. The code is executed in a Jupyter environment, with the output of the first cell showing the "Peppers" image. The second cell shows the padding and median filtering process, and the third cell shows the final result, "IMAGE AFTER MEDIAN FILTERING".

The bottom of the screen shows the Windows taskbar with the date and time: 19:14, 29.10.2022.

We print the median filtered image by converting it to uint8 format.

Olçan_Satir_152120171109_Hw2 x Homework02

colab.research.google.com/drive/1HnoMTJ-bOJNzyzpjSJ8U4VL5sM6o2FTL#scrollTo=0hxGS3tf48R6

Görseller NextGCT-IRG - Goo... Course Materials fo... SAP FI Nedir? - SAP... Staj Mannheim - SA... Softwareentwickler... Stellenangebote un... tds Veri Bilimcisi Olman... Git meta verileri ali... Trade | Trader Joe

Olçan_Satir_152120171109_Hw2.ipynb ☆

Dosya Düzenle Göster Ekle Çalışma zamanı Araçlar Yardım Tüm değişiklikler kaydedildi


Dosyalar

- ..
- .config
- sample_data
- peppers.png

+ Kod + Metin

```
[6] #CONVERT THE IMAGE TO UINT8 FORMAT.  
B= np.array(255*B, dtype = 'uint8')  
plot_im(B, "IMAGE AFTER MEDIAN FILTERING")
```

IMAGE AFTER MEDIAN FILTERING



1 sn. 85.16 GB kullanılabilir

https://accounts.google.com/SignOutOptions?hl=tr&continue=https://colab.research.google... 1 sn. tamamlanma zamanı: 19:04

13°C Bulutlu

19:14 29.10.2022