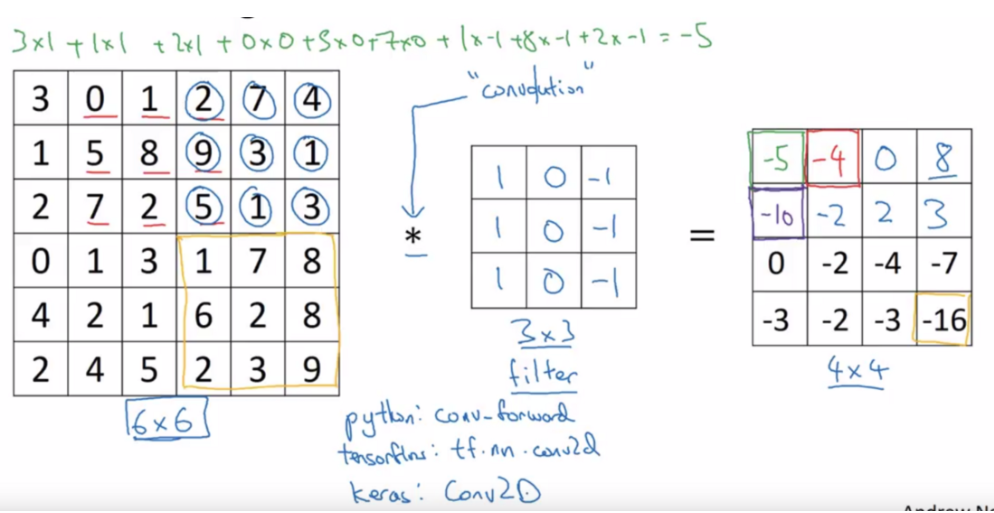
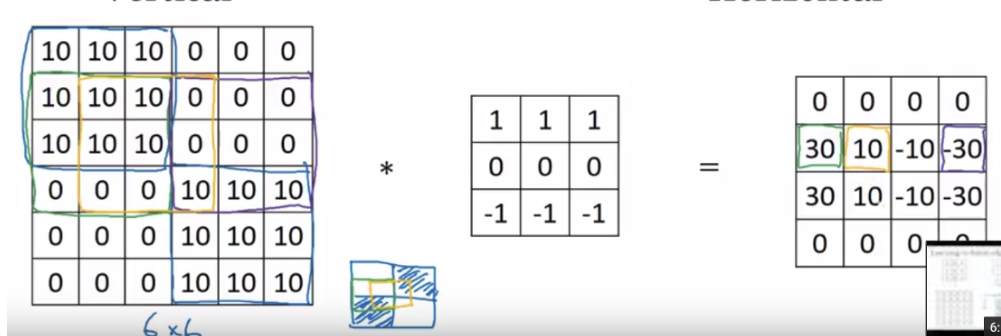
**Convolutional Neural Network Terminoloji – Filtreler – Python Kodları**

**Vertical Kenar Filtresi:**

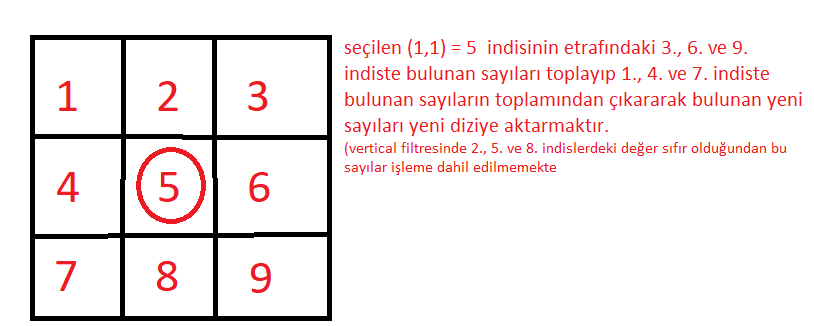
****

**Horizontal Kenar Filtresi:**



**Vertical Filtresi:**

Bir görüntüye vertical şekilde filtre uygulamak;

Fotoğraf 1.1

Bu filtreyi python ile şu şekilde kodlayabiliriz: Fotoğraf 1.2

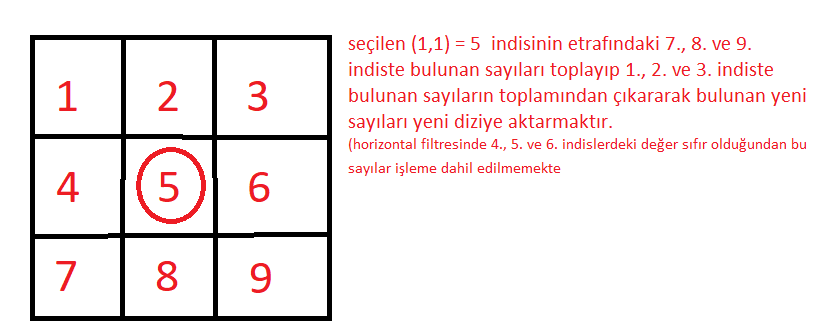
Görüntünün bir katmanını, satır ve sütun değerini aldığımız bu fonksiyonda, for döngüsünü 1’den başlatıp satir ve sütun değerlerinden bir önceki değere kadar çalıştırıyoruz çünkü başlamasını istediğimiz nokta (1,1) indisi, durmasını istediğimiz nokta ise en son indis değil, bir önceki indis. Fotoğraf 1.1. örnekte gördüğümüz anlatımdaki gibi indis değerlerini, numaralarla belirttiğimiz değişkenlere atıyoruz ve topluyoruz. Toplamı daha sonra bir diziye atıyoruz çünkü bu toplam bizim çıkarımımızda yeni eleman olacak.

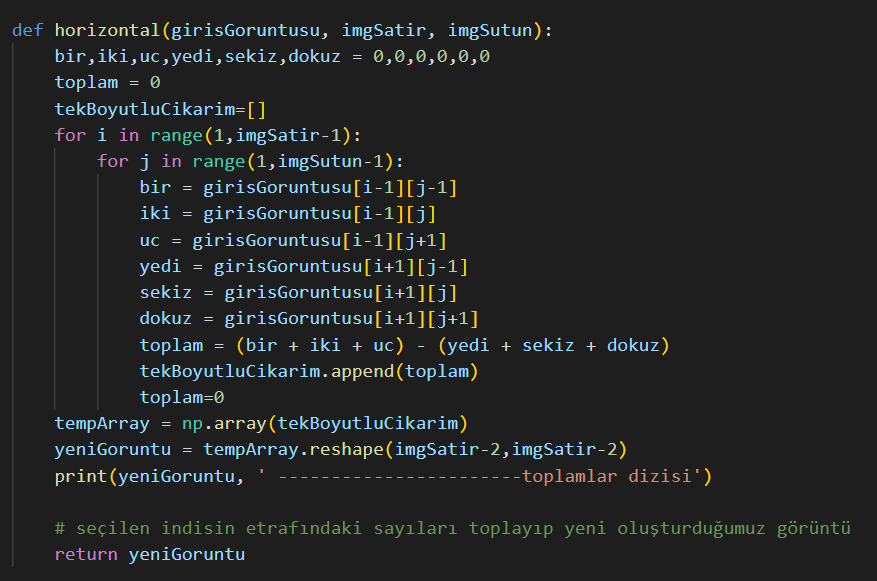
Döngünün sonunda bu tek boyutlu dizimizi gelen görüntü dizisi gibi, iki boyutlu bir diziye çeviriyoruz. Burada numpy metodunu kullandık. .reshape metodu ile yeni satır ve sütun değerlerini verdik: -2 dememizin sebebi oluşan her çıkarımın bize gelen dizinin satır ve sütunundan 2 eksik sayıda satın ve sütuna sahip olması. (Kayıp var)

Daha sonra oluşan yeni görüntüyü return ediyoruz.

**Horizontal Kenar Filtresi:**

Bir görüntüye horizontal şekilde filtre uygulamak ise;

Fotoğraf 1.3

Bu filtreyi python ile şu şekilde kodlayabiliriz: 

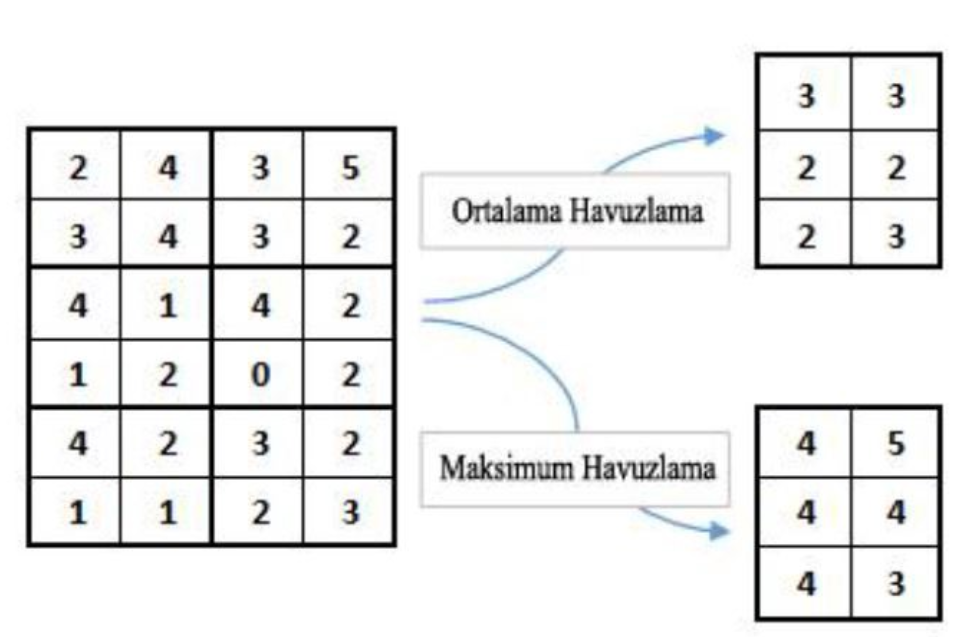
Fotoğraf 1.4

Veritcal fonksiyonundaki gibi görüntünün bir katmanını, satır ve sütun değerini aldığımız bu fonksiyonda, for döngüsünü 1’den başlatıp satir ve sütun değerlerinden bir önceki değere kadar çalıştırıyoruz çünkü başlamasını istediğimiz nokta (1,1) indisi, durmasını istediğimiz nokta ise en son indis değil, bir önceki indis. Fotoğraf 1.3. örnekte gördüğümüz anlatımdaki gibi indis değerlerini, numaralarla belirttiğimiz değişkenlere atıyoruz ve topluyoruz. Toplamı daha sonra bir diziye atıyoruz çünkü bu toplam bizim çıkarımımızda yeni eleman olacak.

Döngünün sonunda bu tek boyutlu dizimizi gelen görüntü dizisi gibi, iki boyutlu bir diziye çeviriyoruz. Burada numpy metodunu kullandık. .reshape metodu ile yeni satır ve sütun değerlerini verdik: -2 dememizin sebebi oluşan her çıkarımın bize gelen dizinin satır ve sütunundan 2 eksik sayıda satın ve sütuna sahip olması. (Kayıp var)

Daha sonra oluşan yeni görüntüyü return ediyoruz.

**Havuzlama:**

Fotoğraf 1.3

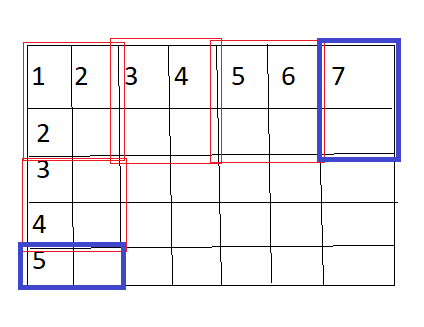
Havuzlama ile görüntüden 2x2 lik matrisler alıp bu matrislerden en büyük elemanı yeni diziye ekliyoruz. Bu işlem ile görüntü 1/3 oranında bi küçülme daha yaşıyor.

Havuzlama işleminin Python kodu:

 Fotoğraf 1.4

Filtre uygulanmış görüntüyü alıp 2x2 ‘lik matrisler seçmemiz gerekiyor. Burada for döngümüzde i ve j değerlerini ikişerli şekilde arttırarak ilerleyecek for döngüsü kurarak bunu yapabiliriz. Bu nedenle range fonksiyonunu 3 parametre ile başlatıyoruz; (başlangıç değeri, bitiş değeri, artış). Bu döngü bizim görüntümüzün satır ve sütun değerleri içindi. İçerisinde ek olarak 2x2 ‘lik matris seçebilmek için, bu matrisin for döngülerini tanımlıyoruz. Bu for döngüsünden önce gördüğümüz secilenDortEleman isimli dizinin olma sebesi ise 4 elemanı seçip, üzerinde işlem yapabilmek ve sonrasında yeni dörtlü kısım için içini temizlemek.

Bu for döngüsünün içindeyse iki adet if değerleri var. Bunlar; satır veya sütun sayısı eğer 2x2 olarak ayrılabilecek oranda değilse bunları 1x2 veya 2x1 şeklinde ayırabilmek için var. Bunu daha iyi anlayabilmek için şu şekil üzerinde gidelim.

Fotoğraf 1.5

Şekilde gördüğümüz gibi 5 satır 7 sütun var ve 2x2 lik matrisler var. Bizim ikinci tanımladığımız for döngüsü ile bu matrisleri seçiyoruz.

**Dizi indis değerleri: 2x2 ‘lik dizi değerleri:**

[00 01] [02 03] [04 05 ] [06] [00 01]

[10 11] [12 13] [14 15] [16] [10 11]

Bizim ilk döngümüzde verdiğimiz for değerleri ikişerli olarak arttığı için yalnızca ilk for döngüsünü çalıştırdığımızda elde edeceğimiz indis değerleri: [00] [02] [06] oluyor. İkinci matris ile aradaki değerlere ulaşabiliyoruz. Bunu da indis değerlerini toplayarak yapıyoruz:

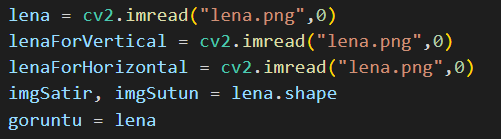
verticalFiltre [i+l] [j+m] => [02] seçildi diyelim, ikinci for çifti ile bu 0 ve 2 indislerine ekleme yaparak sağa ve aşağı gidebilyoruz. Yani [02][03] , [12][13] değerlerini seçebiliyoruz.

Koşulda (i + l), (j+m) değerleri son elemandan büyük olduğu durumda hata almamak ve son elemanları da seçebilmek için l ve m değerlerinden 1 çıkarıyoruz. Çünkü bu değerleri zaten yalnızca 0 ve 1 değerlerini alabiliyordu (range(2) => 0 ve 1 çalışır, 2 dahil değildir) . Bu nedenle en fazla +1 fazlalık olabilir. Örneğin i==6 iken l==1 olursa 7 çıktısı ‘out of range’ hatasını verir ve son elemanı alamayız. Bu nedenle l -=1 yapıp i’yi tekrar 0 ile toplattırıyoruz ve hatadan kurtuluyoruz.

Sonrasında bu seçilen dört elemanınn olduğu diziye sort() işlemi uygulayıp, koşul içerisinde maximum elemanını seçiyoruz. Ve havuzlama sonucu oluşan tek boyutlu diziyi de geri dönderiyoruz.

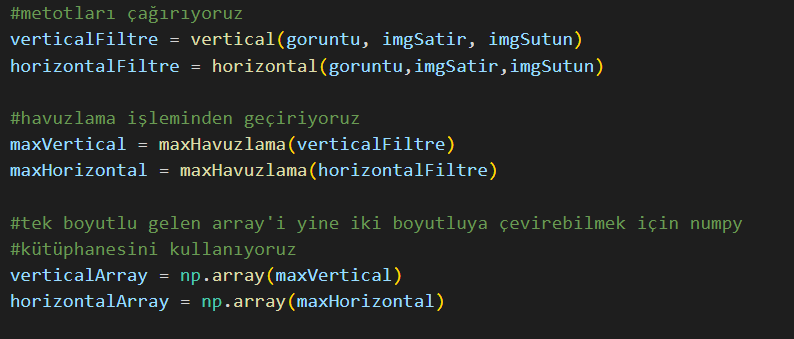
**Sonuç:**

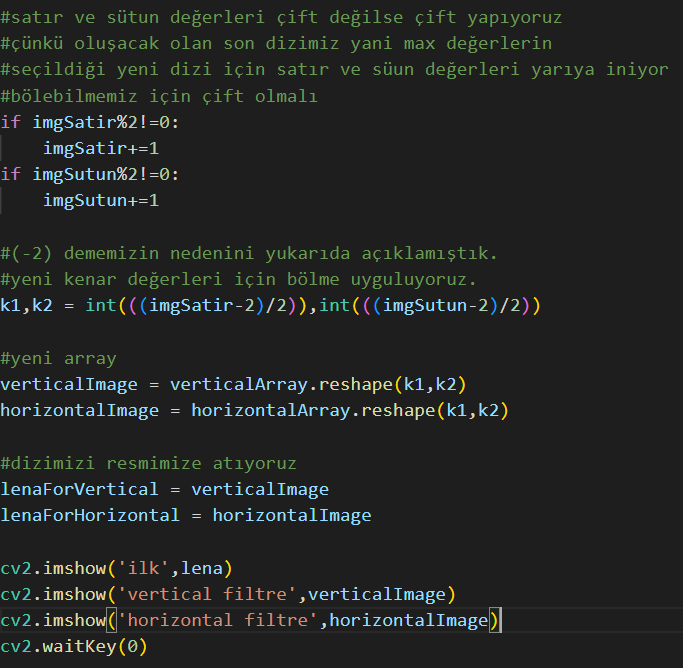
Şimdi kodumuzu çalıştırıp sonuçları görebilmek için görüntü matrisini ve değerleri gönderelim.



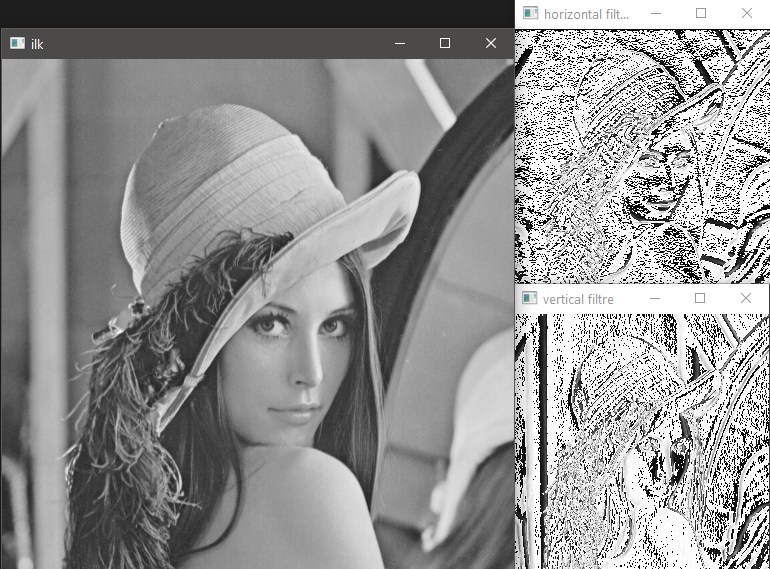
Fotoğraf 1.6

Son kısımda vertical ve horizontal filtrelerinin resimde nasıl göründüğünü görebilmek için lenaForVertical ve lenaForHorizontal tanımladık.

Fotoğraf 1.7

Fotoğraf 1.8

**Ve Resimler:**

****