Python’da En hızlı şekilde Resim Okuma

Resim işleme son dönemlerde oldukça yaygın bir şekilde çeşitli programlama dilleri kullanılarak yapılan uygulama türlerinin başında geliyor.

Neden mi?

Çünkü görsel bir çıktı vermesi doğruluğunun kontrol edilmesi açısından oldukça kolay. Yani resim ile oynarken keskinlik ekleme, filtre ekleme ya da gürültü giderme gibi işlemler yapıldıktan sonra kolaylıkla kontrol edilebiliyor.

OpenCV , Sckit-image gibi modüller özellikle kullanım rahatlığı açısından oldukça güzel modüller .Tek satırla nerdeyse çoğu işlemi halletmeyi mümkün kılıyor.

Diğer taraftan genel mana da görüntü işleme matlab ve python üzerinde yoğunlaşmış gibi görünüyor ama hızlı işlem yapma ihtiyacı özellikle şirketler için c++’a yönlendiriyor.

Çünkü alt seviye diller üst seviye dillere göre daha hızlıdır. Makine diline yakın olmak hız kazandırsa da üst seviye dillere göre yazımı çok zaman kaybettirir. Yollar burada ikiye ayrılıyor demek yanlış olmaz.

Hız mı istiyorsun, yoksa daha rahat yazmak mı? Karar verip sonrasında projelere başlamak en mantıklısı olur.

Bu bölümde rahat yazmayı seçenler (python ) için şu soruların cevabını arayacağız .

Bu dili kullanarak nasıl daha hızlı yazarız ?

Hangi modüller daha kullanışlı bu soruların ?

Benim bu konu hakkında yazma sebebim ise özellikle son dönemlerde yaygınlaşan derin öğrenme için (örnek olarak CNN) gerekli resim sayısının binlerce olması.

Durum böyle olunca veri setindeki binlerce resim en hızlı şekilde okunup öyle derin öğrenme ağına gönderilmeli. Bu sayede zaten oldukça zaman alan derin öğrenmenin en azında resim ön işleme kısmını bir nebze hızlandırmış oluruz. Geri kalan kısımları için neler yapıldığını sonraki yazılarım da bahsetmeye çalışacağım.

Amacımı ve gerekçeleri sıraladıktan sonra hafif hafif başlayabiliriz?

Herkesin erişebilecepği bir dataset kullanmak en iyi olur diye düşündüm bu yüzden Mnist biçilmez bir kaftan.Fakat indirirken ubyte ve zipli bir şekilde indiriyor .Şayet Pytorch ve tensorflow un dataset sınıfını kullanarak indiriyorsanız.

Bende öyle yaptım ve inen tüm resimleri jpeg formatına çevirerek kaydettim. Sebebim elinizde resimlerin olduğu herhangi bir dataset’e neler yapıyorsanız buna da aynısını yapmak.Çünkü deep learning de uğraşırken genelde kaggle vb yerlerden çekeceğiniz dataset pytorch veya tensorflow un sizlere sunduğu datasetten çok farklı olacak.

Yeri gelmişken Nasıl farklardan bahsediyorum.

1-Temiz olmayacak .Yani sizin bir ön işlemden geçirmeniz gerekecez.

2-Boyutları aynı olmayacak . Boyutlarını sabitlemeniz gerekecek.(Model kurmada sabit boyut şart )

3- Sınıflandırma problemi üzerine çalışıyorsanız bazen sınıfları olduğundan farklı olacak

Şimdilik aklıma gelen bunlar ☺ .İçine girdikçe daha çok sorunla karşılacağınıza emin olabilirsiniz.

Resimlerin indirilip jpeg formatında kaydedildiği kod ile başlayalım.

Genelde torch kütüphanesini kullanmayı seviyorum. Akademik alandada çok sık kullanılıyor ve araştırma temelli bir modül diyebilirim. Bi göz atın buna

import torchvision.datasets as datasets

mnist\_trainset = datasets.MNIST(root='.', train=True, download=True, transform=None)

Spyder üzerinde gösterecek olursak

Kodu ,

import numpy as np

import matplotlib.pyplot as plt

def show\_imgs(dataset):

fig = plt.figure(figsize=(28, 28))

for i in range(1,21):

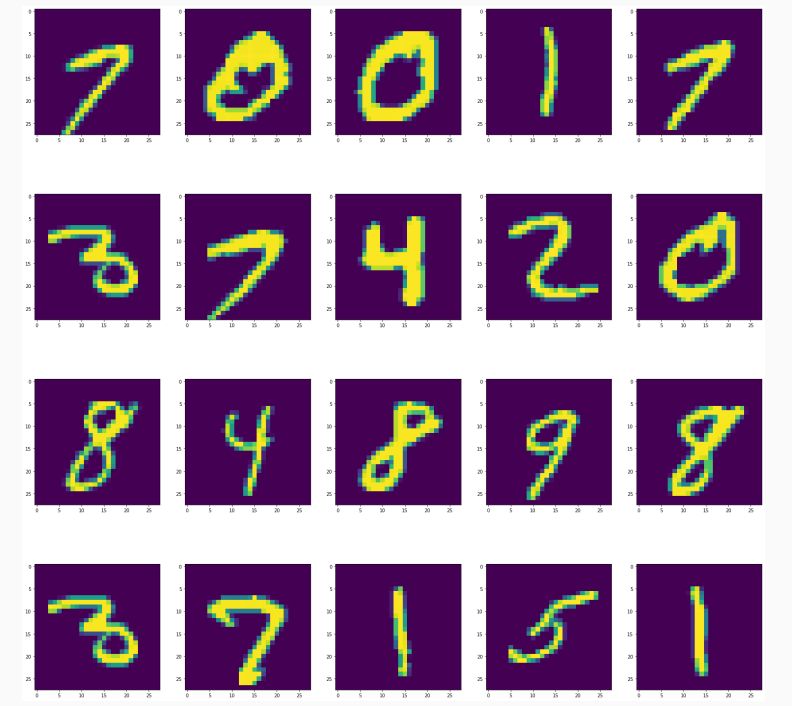
idx = np.random.randint(100)

fig.add\_subplot(4,5,i)

plt.imshow(dataset.data[idx])

plt.show()

Çıktısı ,

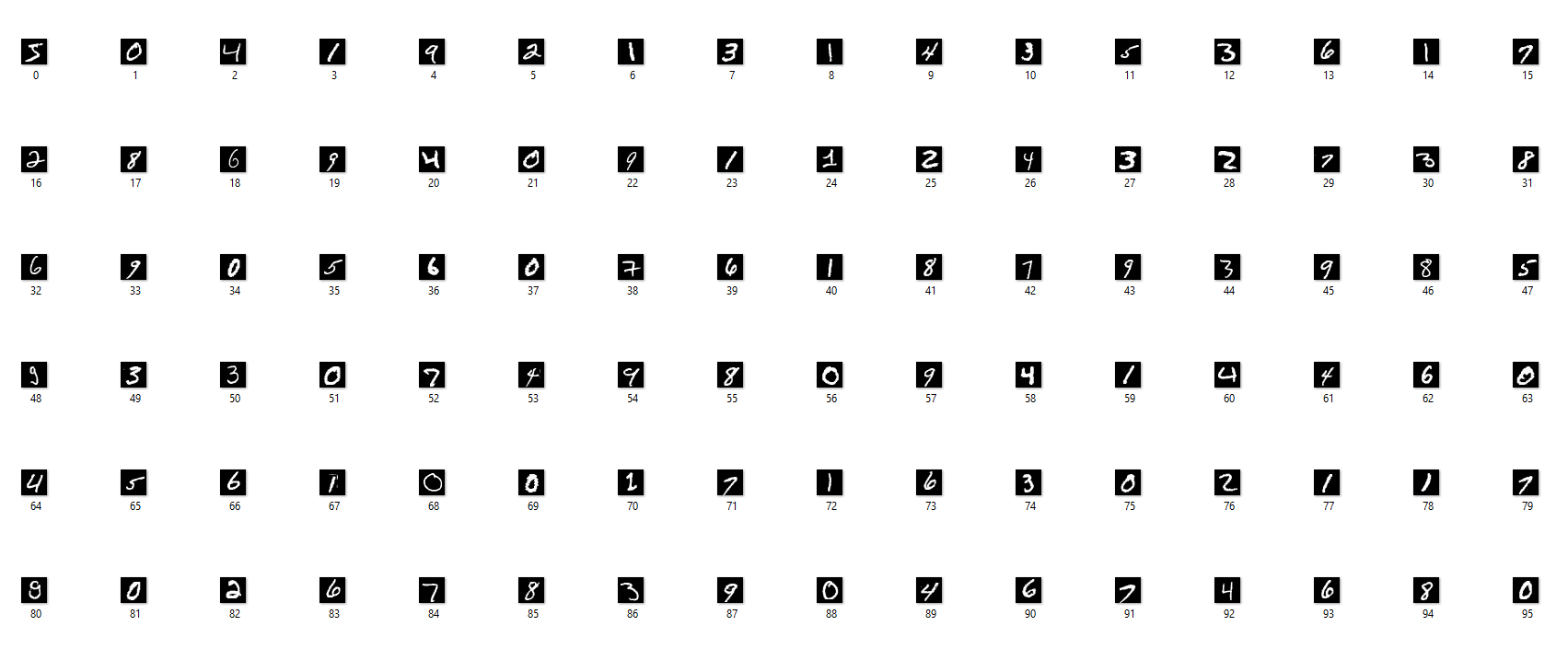


Şimdilik train kısmındaki resimleri indirsek yeterli olur .Çünkü 60,000 görüntü var .Biz sadece 10000 ini alalım ve bunu opencv kütüphanesini kullanarak kaydedelim.

for i in range(10000):

cv2.imwrite(str(i)+".jpg", mnist\_trainset.data[i].numpy())

Klasörümüz içini açtığımızda hiç fena durmuyor ☺

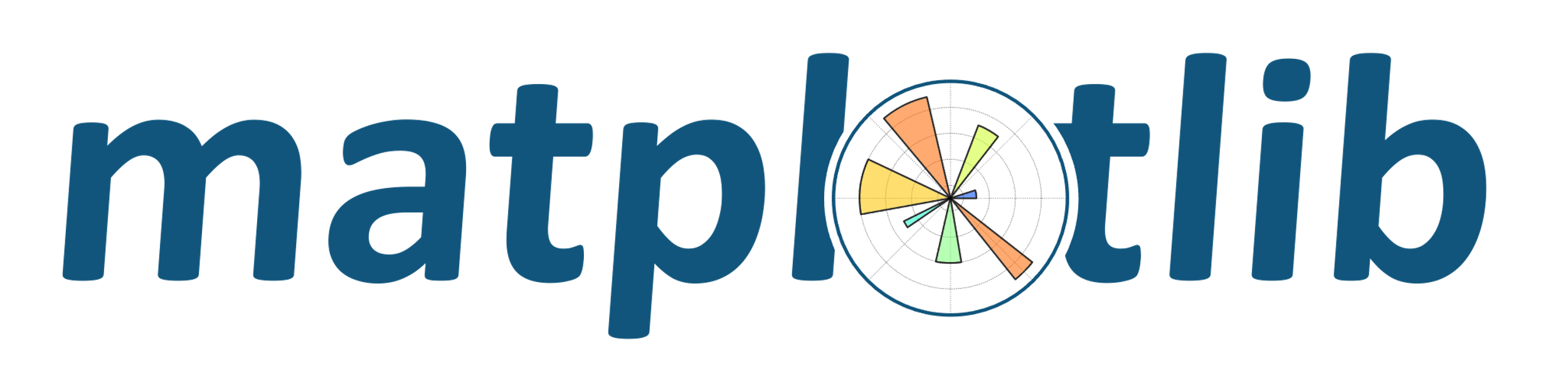


Şimdi bu resimleri tamamını farklı modüller kullanarak bir dizine kaydetmek istiyoruz ki sonrasında kullanabilelim.

Bu projede belirten süreler gerçeği çok yansıtmamaktadır. Yanlış olduğunu düşünmeyin aynı işlemi ardı ardına çalıştırınca bilgisayar sık kullanıyor diye düşünüp cache atıyor.Bu sayede çok daha hızlı bir şekilde okuyor .Genel aldığım sonuçlar 1 ile 5 saniye arası ama olması gereken 20-30 saniye arasıdır.Sizde ilk denemenizde ve 2. Kez aynı kodu çalıştırınca farkı göreceksiniz.

Sadece yapmamız gereken aradaki farklara bakıp en uygununu seçmek.

İlk Durağımızı Matplolib



<https://matplotlib.org>

Resimlerin gösterilmesinde çok sık kullanan bu kütüphane okuma işlemi içinde kullanıyor.

#MATPLOTLIB READ

import matplotlib.image as mpimg

starttime = timeit.default\_timer()

for img in os.listdir(img\_path):

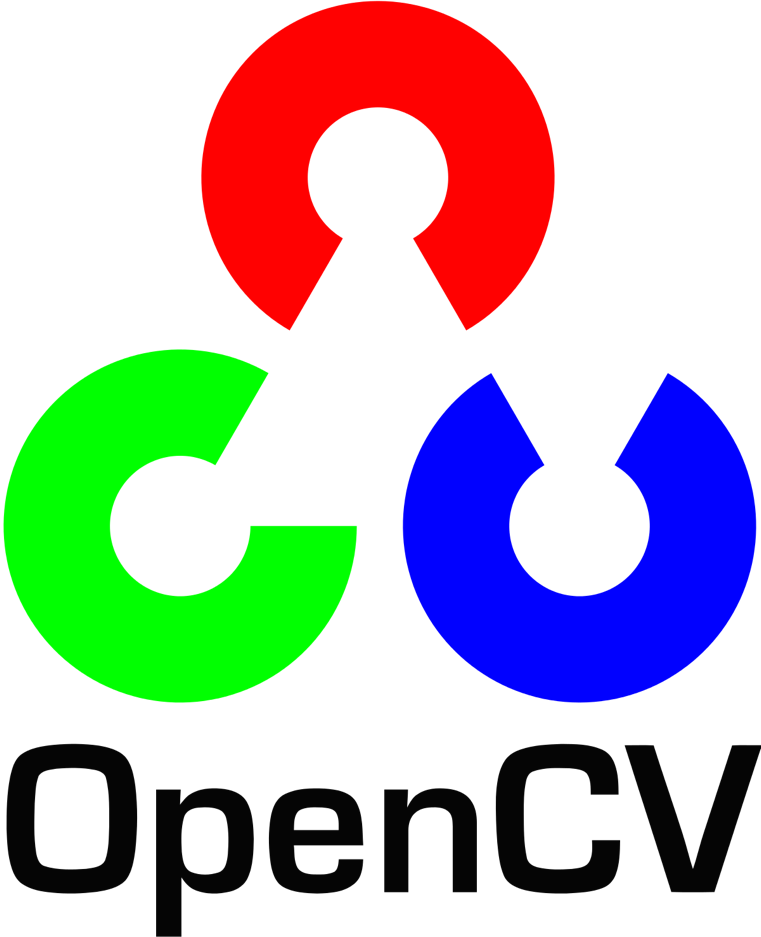
img\_array.append(mpimg.imread(os.path.join(img\_path, img)))

print("The time difference is :", timeit.default\_timer() - starttime)

Çıktısı : #The time difference is : 2.2727211

Matplotlib ile 2.27 saniye de 10000 resmi okumuşuz.

Sonraki Durağımızı Opencv



<https://tr.wikipedia.org/wiki/OpenCV>

# CV2 IMREAD

import cv2

starttime = timeit.default\_timer()

for img in imgs\_name:

img\_array.append(cv2.imread(os.path.join(img\_path, img)))

print("The time difference is :", timeit.default\_timer() - starttime)

Çıktısı : #The time difference is : 0.952221600000000

Opencv ile 0.95 saniye de 10000 resmi okumuşuz.Yani matplotlib in 2 katı az sürede.

Sonraki Durağımızı PIL



https://github.com/python-pillow/Pillow

# PIL READ

from PIL import Image

import numpy as np

starttime = timeit.default\_timer()

for img in imgs\_name:

img\_array.append(np.asarray(Image.open(os.path.join(img\_path, img))))

print("The time difference is :", timeit.default\_timer() - starttime)

#The time difference is : 2.0121553



https://scikit-image.org

# IO IMREAD FROM SKIMAGE

import skimage.io as io

starttime = timeit.default\_timer()

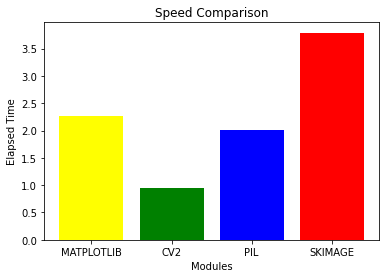
for img in os.listdir(img\_path):

img\_array.append(io.imread(os.path.join(img\_path, img)))

print("The time difference is :", timeit.default\_timer() - starttime)

# The time difference is : 3.799766

En son bir karşılaştırma yaparsak , aşağıdaki grafikte de göreleceği gib opencv en kısa sürede resimleri okuyarak 1. Sırada yerini aldı.



Bu karşılaştırma sayesinde projelerimizde toplu bir resim okuma sürecinde hangisinin daha hızlı bir şekilde okuyacağını artık biliyoruz .