### Pythn ile GörüntüOkumada En hızlı Modüller



Photo by [Caleb George](https://unsplash.com/@seemoris?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com?utm_source=medium&utm_medium=referral)

Görüntü 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 kullanışlılar.Tek satırla nerdeyse çoğu işlemi halletmeyi mümkün kılıyor.

Genel tabloya bakıldığında görüntü işleme **Matlab** ve **Python** üzerinde yoğunlaşmış gibi ama hızlı işlem yapma ihtiyacı özellikle şirketler için C**++** kullanmaya yönlendiriyor.

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

**Hız mı istiyorsun, yoksa daha rahat yazmak mı? Karar verip sonrasında projelere başla derim.**

Bu bölümde rahat yazmayı seçenler **(Python kullananlar )**için şu sorunun cevabını arayacağız .

**Hangi modül resim okumak için en ideal ?**

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.**

Photo by [OpticalNomad](https://unsplash.com/@opticalnomad?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com?utm_source=medium&utm_medium=referral)

Herkesin erişebileceği 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 T**ensorflow** 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** gibiyerlerden çekeceğiniz dataset pytorch veya tensorflow un sizlere sunduğu datasetten çok farklı olacak.

**Nasıl bir fark derseniz ,**

* Temiz olmayacak yani sizin bir ön işlemden geçirmeniz gerekecek.
* Boyutları aynı olmayacak . Boyutlarını sabitlemeniz gerekecek.(Model kurmada sabit boyut şart )
* Sınıflandırma problemi üzerine çalışıyorsanız bazen sınıfları olduğundan farklı olacak

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

Genelde **torch** kütüphanelerini kullanmayı seviyorum. Akademik alandada çok sık kullanılıyor ve araştırma temelli bir modül diyebilirim.

Bi göz atın derim pytorch ile ilgili yapılan çalışmalara.

Resimlerin indirilip jpeg formatında kaydedildiği kod ile başlayalım.

import torchvision.datasets as datasets

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

Ne indi acaba derseniz ,

import numpy as np

import matplotlib.pyplot as plt

def show\_imgs(dataset):

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

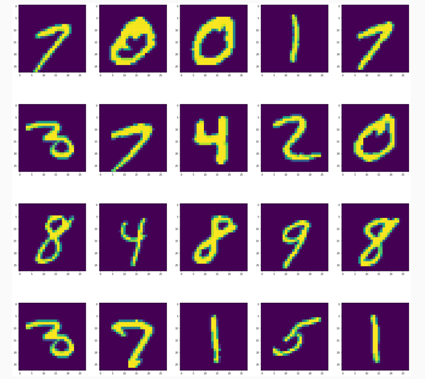
for i in range(1,21):

idx = np.random.randint(100)

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

plt.imshow(dataset.data[idx])

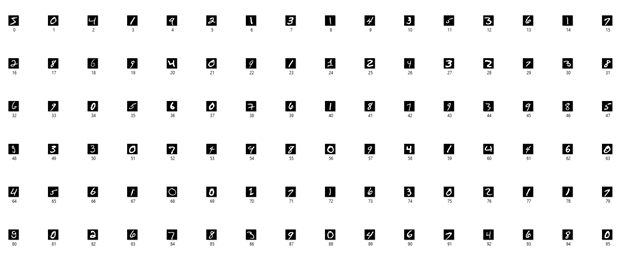
plt.show()

(show\_imgs fonksiyonunun 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())

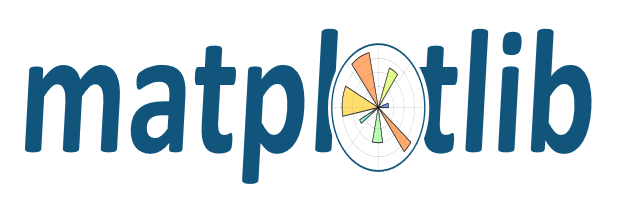
(Klasörde kaydedilmiş hali)

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 Durağımız **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):

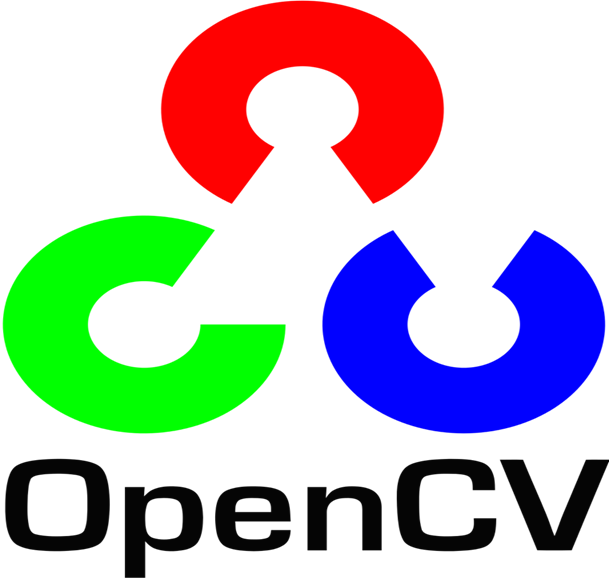
img\_array.append(mpimg.imread(os.path.join(img\_path, img)))

print(“The time difference is :”, timeit.default\_timer() — starttime)

#### The time difference is : 2.2727211

Yani Matplotlib ile 2.27 saniye de 10000 resmi okumuşuz.

Sonraki Durağımız **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)

**The time difference is : 0.952221600000000**

Sonraki Durağımız **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**

Sonuncu olarak **Scikit-Image**,

<https://scikit-image.org>

# IO IMREAD FROM SKIMAGE

import skimage.io as io

starttime = timeit.default\_timer()

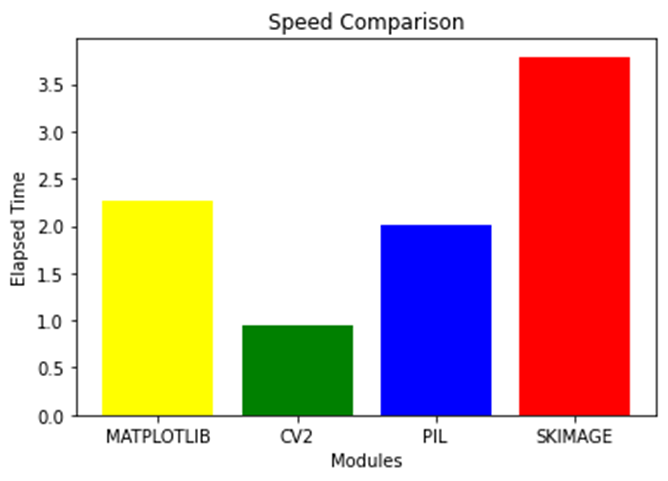
for img in os.listdir(img\_path):

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.

(Harcanan zamanın genel karşılaştırlması)

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

Resim okuma ile alakalı yazımızın sonuna geldik. Diğer yazılarımızda deep learning ile alakalı daha derin noktalara ineceğiz.

Bu yazının kodları githubta ([mntalha](https://github.com/mntalha" \t "_blank)/[Fastest-Approach-to-Read-Images](https://github.com/mntalha/Fastest-Approach-to-Read-Images)) mevcut. Bakmak isterseniz aşağıdaki linkten ulaşabilirsiniz.

Github’ta beni ve çalışmalarımı takip etmek isterseniz çok memnun olurum.

[**mntalha - Overview**  
Electronic Engineering MSc. mntalha has 13 repositories available. Follow their code on GitHub.github.com](https://github.com/mntalha)

Bu konuyla ilgili herhangi bir sorunuz varsa, linked-in yoluyla bana kolayca ulaşabilirsiniz.

[**Muhammed Nur Talha KILIÇ - İstanbul Teknik Üniversitesi - Istanbul, Istanbul, Turkey | LinkedIn**  
View Muhammed Nur Talha KILIÇ's profile on LinkedIn, the world's largest professional community. Muhammed Nur Talha has…www.linkedin.com](https://www.linkedin.com/in/mntalhakilic)