Darknet YOLOv3 ile Windows Üzerinde Nesne Algılama

Görüntü işleme konusuna ilgi duyduğum için ara ara Python ve OpenCV ile elimi kirleterek ufak bir kaç proje yapmıştım. Bu yazımda da merak edenler için birkaç saatlik bir hobi projesi olarak YOLO ile resim ve videolar içerisinde nesne algılamanın nasıl yapılabileceğine değineceğim. YOLO (You Only Look Once) Joseph Redmon tarafından geliştirilen açık kaynak kodlu ve yüksek performanslı bir nesne algılama algoritması. YOLO'nun en güncel versiyonu olan v4 ben bu yazıyı yazarken yayınlandı. İlerideki bir yazımda onu da deneyimlerim, şimdilik stabil olduğundan emin olduğum v3 ile yoluma devam ediyorum. YOLO'nun üzerinde çalıştığı Darknet framework'ü sayesinde Nvidia marka bir GPU'ya sahipseniz aşağıdaki adımları takip ederek siz de kendi yüksek performanslı nesne algılayıcınızı çalıştırabilirsiniz. Nvidia GPU'nuz yoksa Nvidia CUDA'yı yüklemenize ve CMake ile derlerken CUDA seçeneklerini atlayarak ta çalıştırmanız mümkün.

Gereksinimler

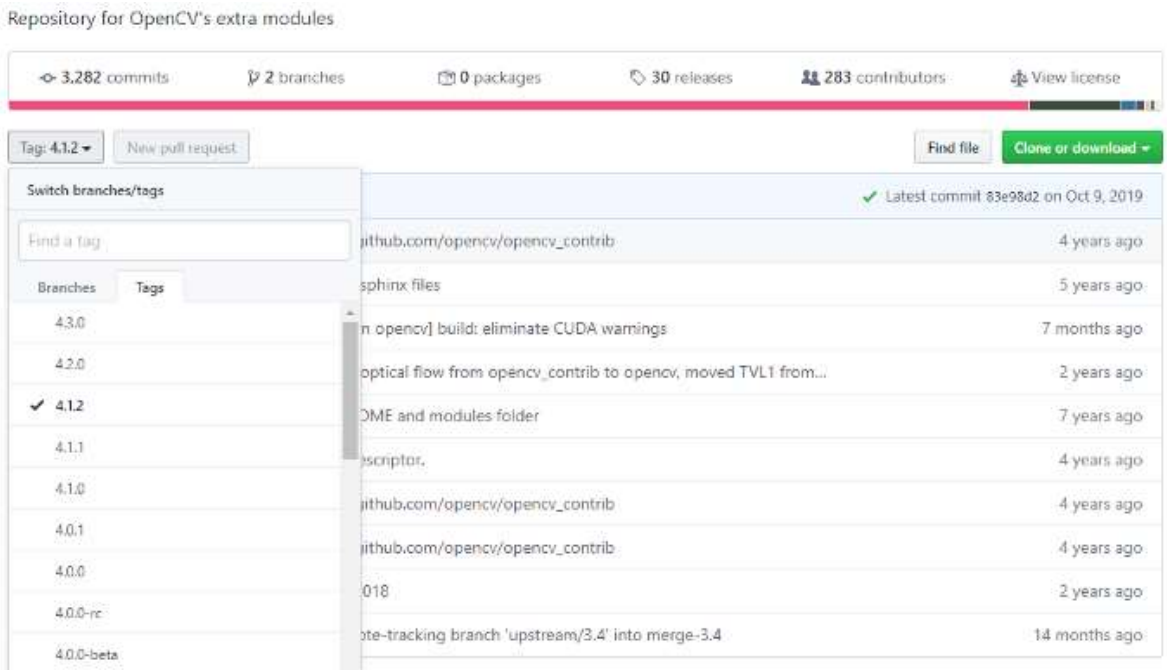
- **Azim:** Bu işlemler süresince yolunda gitmeyen bir çok şey olması çok normal, aldığımız hata kodlarıyla internette yapacağımız aramalar sonucunda yapbozun parçalarını birer birer yerine koyup devam etmek hobilerin doğasında var

Öncelikle aşağıdakileri indiriyor ve kuruyoruz;

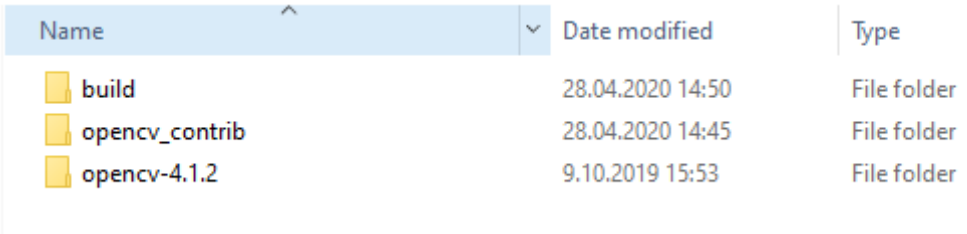
- **Python:** Kurulum sihirbazı aracılığı ile güncel versiyonu kurmanız yeterli https://www.python.org/downloads/
- **Cmake:** Farklı geliştirme ortamları için dosya yapısı derleme aracı https://cmake.org/download/
- **Visual Studio 2019:** IDE. Kurulum esnasında "Desktop development with C++"ın işaretli olduğundan emin olun. https://visualstudio.microsoft.com
- **OpenCV:** Açık kaynak kodlu görüntü işleme kütüphanesi (Windows üzerinde Darknet'in stabil olarak çalıştığı versiyon 4.1.2'yi kullanacağız, kurulum dosyası yerine "sources" a tıklayarak zip dosyası indiriyoruz ve arşivi çıkarıyoruz. https://opencv.org/releases/
- **(Opsiyonel) Nvidia CUDA:** GPU'yu kullanarak daha verimli çalışması için Nvidia'nın CUDA Toolkit'ini kullanacağız. https://en.wikipedia.org/wiki/CUDA adresinden grafik kartınızla uyumlu versiyonu kontrol edebilir ve https://developer.nvidia.com/cuda-downloads adresinden indirebilirsiniz.

CMake ile OpenCV Kurulumu

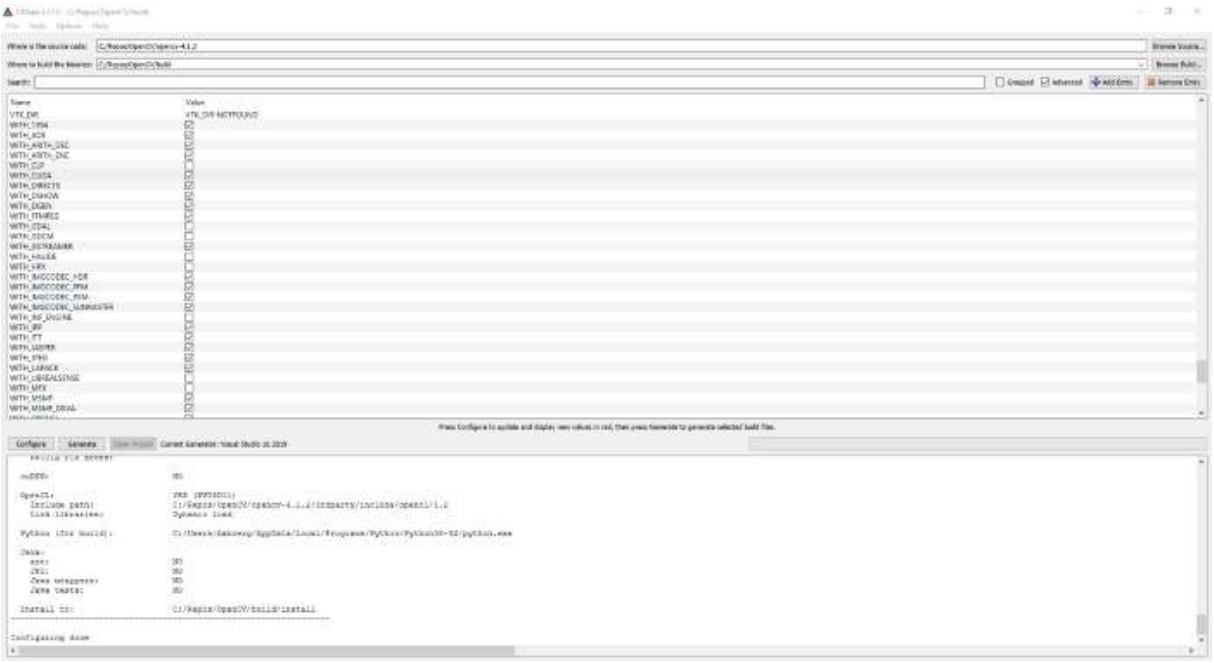
OpenCV için indirilen zip dosyasını bir klasöre açıyoruz. Ardından OpenCV'nin sitesinden indirdiğimiz klasör içinde olmayan fonksiyonları barındıran contrib repo'sunun 4.1.2. versiyonunu ediniyoruz. https://github.com/opencv/opencv_contrib adresinden zip halinde indiriyoruz ve arşivi açıyoruz. (github repoları üzerinde çalışmak için gitbash'i de kullanabilirsiniz)



Bu işlemin ardından iki OpenCV klasörümüz oluyor, birisi temel bileşenleri içeren diğeri ise ek modüllerin olduğu klasör. Üçüncü klasörü “build” adıyla oluşturuyoruz.

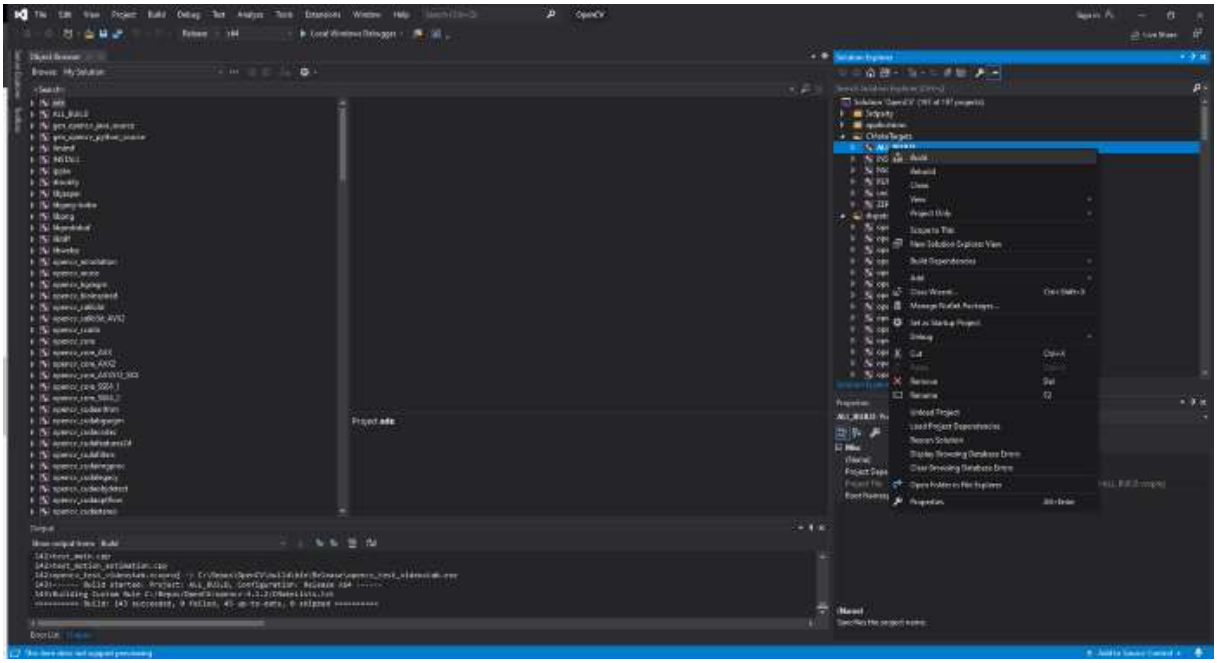


Şimdi CMake ile OpenCV’nin temel ve ek modüllerini birlikte derleyeceğiz, bunun için CMake’i çalıştırıyoruz. Browse source ile temel klasör olan “opencv-4.2.1”’ı seçiyoruz, browse build ile çıktıların olacağı “build” klasörünü seçiyoruz. (Contrib klasörü ile ilgili seçimi bir sonraki adımda yapacağız.) Configure’e tıklayarak mimariyi seçiyoruz (ben x64 kullanıyorum) ve “finish” ile derlemeyi başlatıyoruz. Bu işlemin sonunda aşağıdaki gibi bolca seçenek ile karşılanıyoruz.



OPENCV_EXTRA_MODULES_PATH değerini de contrib klasörü içerisindeki modules klasörünü gösterecek şekilde güncelliyoruz; C:/Users/xx/Downloads/OpenCV/opencv_contrib/modules.
GPU'muzun gücünden faydalanmak için WITH_CUDA seçeneğini işaretleyip tekrar "build"e tıklıyoruz. Tekrar Finish'e tıkladıktan sonra OpenCV'nin sorunsuz bir şekilde configure oluyor ve Generate'e tıklıyoruz.

Build klasöründeki OpenCV.sln solution dosyasını Visual Studio ile açıyoruz. Release modunda iken CmakeTargets'in altındaki ALL_BUILD öğesini build ediyoruz. Bittikten sonra kurulumu tamamlamak için INSTALL öğesini build ediyoruz.

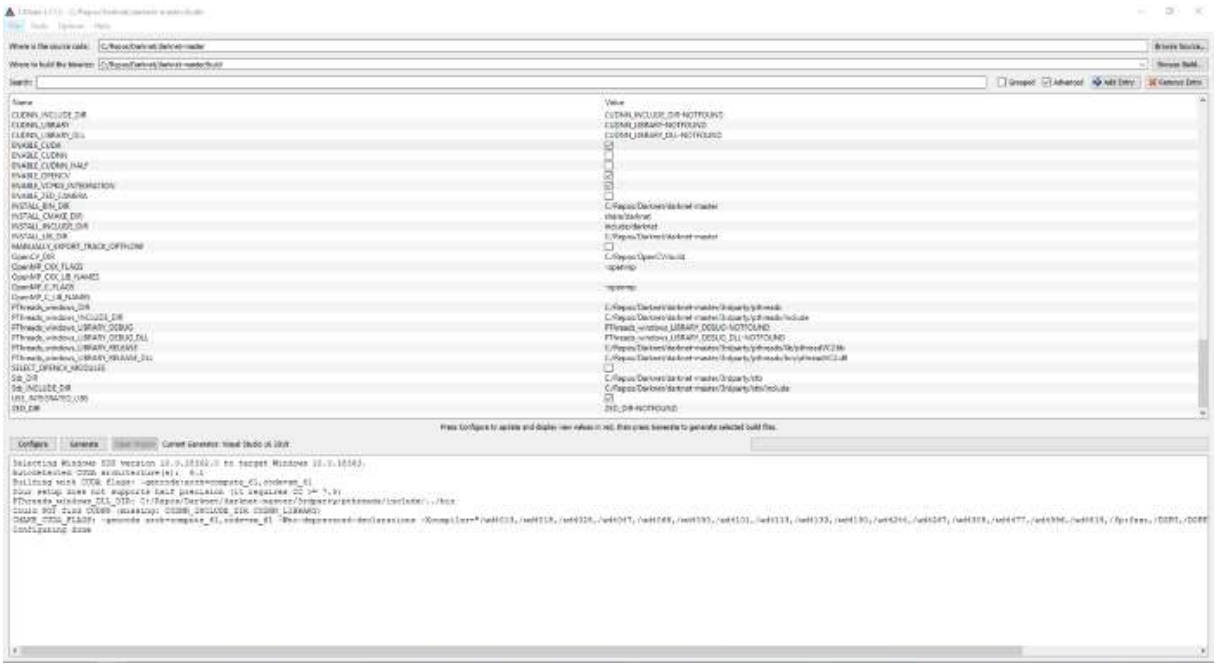


CMake ile Darknet YOLO Kurulumu

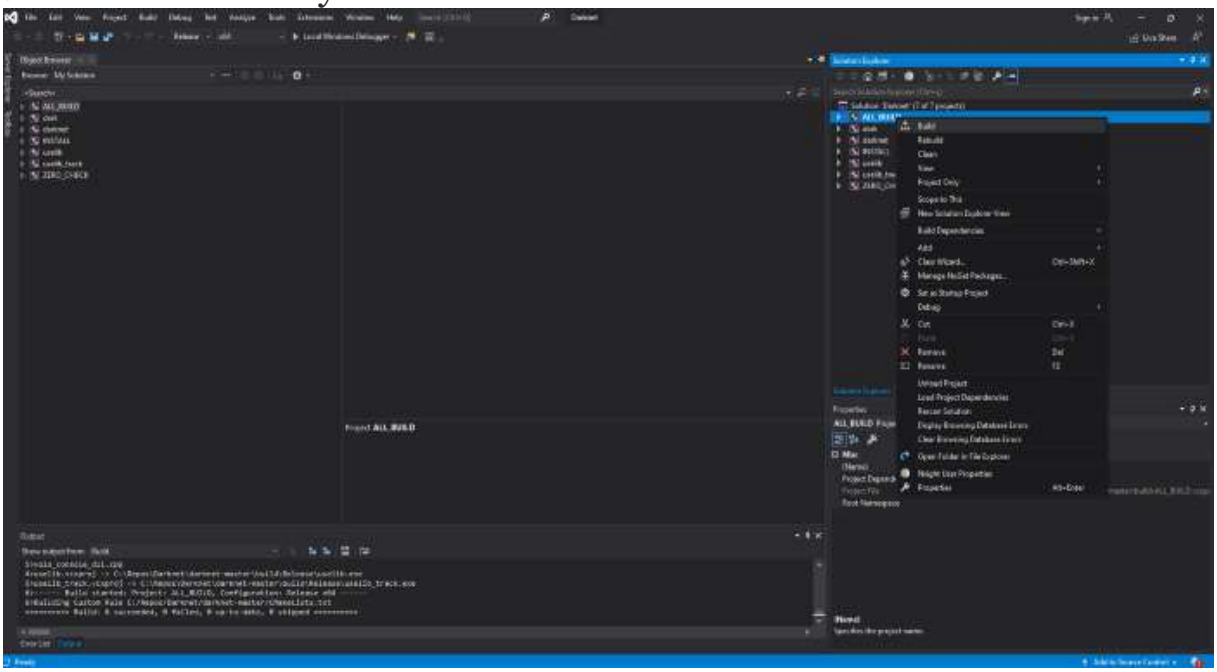
Github'dan orijinal Darknet YOLO repo'sunu ediniyoruz. https://github.com/AlexeyAB/darknet adresinden zip halinde indiriyoruz

Darknet repo'su içerisinde kendi eğitilmiş modelimiz olmadığı için ön-eğitimli ağırlıkları içeren modeli ediniyoruz. https://pjreddie.com/media/files/yolov3.weights adresinden indirip darknet klasörü içerisine kaydediyoruz

Ardından CMake'i açarak Source olarak "darknet" klasörünü, build olarak ise darknet içerisindeki "build" klasörünü seçerek Configure'e tıklıyoruz. Yine mimarimizi seçerek Finish'e tıklıyoruz, Hatalar aldıktan sonra OpenCV_DIR satırına daha önce oluşturduğumuz OpenCV build klasörünü işaret ediyoruz ve ENABLE_CUDA'yı seçerek tekrar configure dedikten sonra hata almadığımızdan emin olup Generate'e tıklıyoruz.



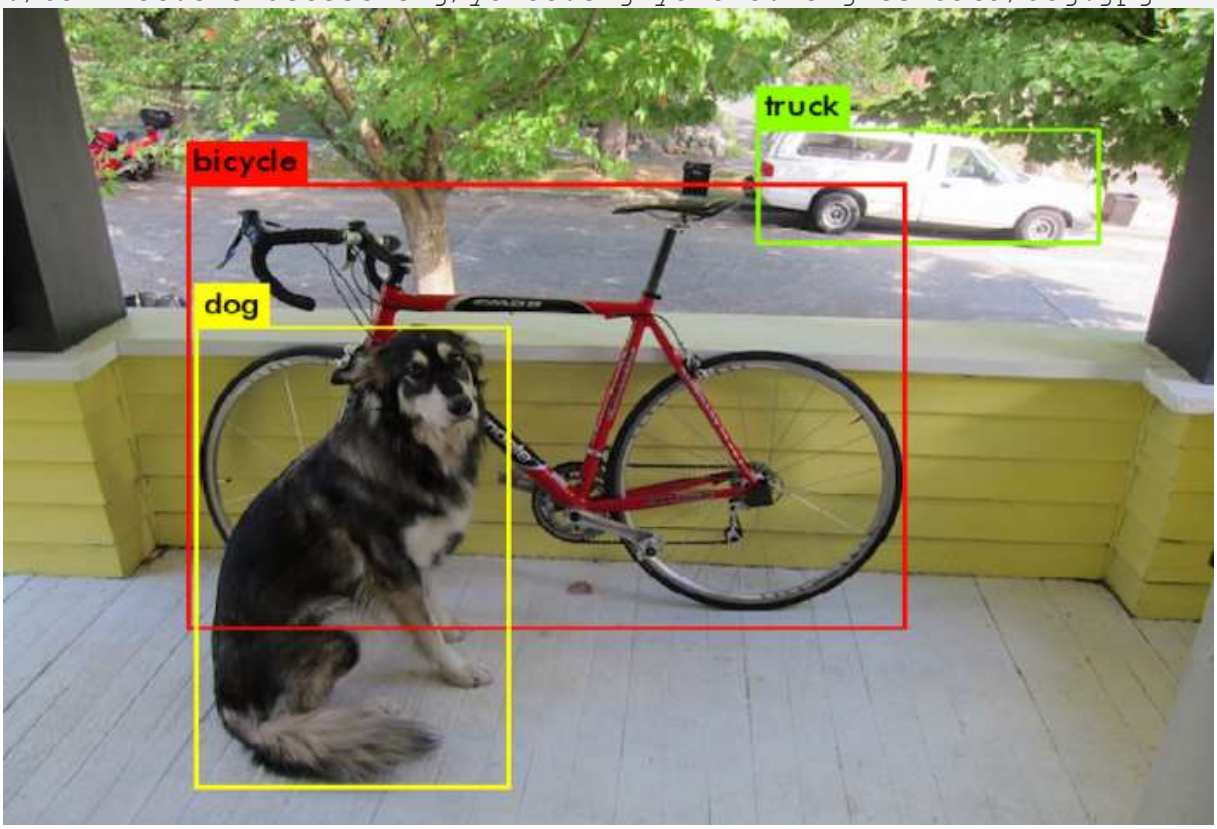
Ardından Darknet'in build klasöründeki Darknet.sln solution dosyasını Visual Studio ile açıyoruz. ALL_BUILD öğesini build ediyoruz ardından aynı işlemi INSTALL öğesi için de gerçekleştiriyoruz. Şu ana kadar herşey yolunda gittiyse hata almadan build'leri tamamlıyoruz



Deneme Bir… İki…

Modeli denemek için bir komut satırı veya PowerShell açtıktan sonra darknet klasörümüzün içerisine gelerek aşağıdaki komutu çalıştırdığımızda, indirdiğimiz ön eğitimli modeli data klasörü içindeki hazır resimlerden biri olan dog.jpg dosyası üzerinde çalıştırmış oluyoruz.

```
./darknet.exe detect cfg/yolo3.cfg yolov3.weights data/dog.jpg
```



Artık aynı klasör içerisindeyken ön eğitimli modelin çalışabileceği ve çok spesifik objeler içermeyen (ön eğitimli model kullanıyor olmamız sebebiyle) kayıtlı bir video üzerinde de nesne algılaması yapabiliriz.

```
./darknet.exe detector demo cfg/coco.data cfg/yolov3.cfg
yolov3.weights -ext_output videos/car.mp4 -out_filename output.avi
```

İlerleyen dönemde görüntü işleme denemelerimi tekrar Linux ortamına taşıyıp, yüksek FPS'li ve canlı görüntüler üzerinde yapmayı planlıyorum. Bunu yaparken işin detayına biraz daha girip

üniversitedeki bilgilerimi tazelemeyi amaçlıyorum. Ayrıca YOLOv4 ile ilgili başka deneyimlerimi de yine buradan paylaşmaya çalışacağım.