HACETTEPE ÜNİVERSİTESİ

27.12.2024

İSTATİSTİKSEL VERİ MADECİLİĞİ FİNAL RAPORU

HAZIRLAYANLAR

Melike Zeynep IŞIKTAŞ 2210329109
Sidar Deniz TOPALOĞLU 2210329107

DERS SORUMLUSU

Dr. Öğr. Üyesi Onur Toka

İÇİNDEKİLER

Veriye Ait Genel Bilgiler

Bu proje kapsamında kullanılan "Wonders of the World Image Classification" veri seti, dünyanın yedi harikasını ve bazı diğer ünlü yapıları sınıflandırmak amacıyla derlenmiştir. Kaggle platformunda Balabaskar tarafından sağlanan bu veri seti, JPG formatında 12,160 görselden oluşmaktadır ve görseller yedi temel sınıfa (Great Wall of China, Taj Mahal, Machu Picchu, Christ the Redeemer, Petra, Colosseum, Chichen Itza) ayrılmıştır. Her bir sınıf, ilgili yapıya ait görselleri içermektedir ve görseller sınıflar arasında dengeli şekilde dağıtılmıştır. Etiketli yapısıyla denetimli öğrenme algoritmaları için uygundur ve çözünürlük farklılıkları nedeniyle ön işleme gerektirebilir. Bu veri seti, kültürel mirasların tanınması ve görüntü işleme temelli sınıflandırma projelerinde değerli bir kaynak sunmaktadır. Veri setinin %87'sine karşılık gelen 10,620 görsel train set , %8'ine karşılık gelen 1,008 görüntü validation set ve %4'lük bir orana denk gelen 532 görüntü test seti olarak kullanılmıştır.

Çalışmanın Amacı

Bu çalışmanın amacı, görüntü işleme ve makine öğrenimi yöntemlerini kullanarak dünyanın yedi harikasını ve diğer seçilmiş yapıları tanıyabilen bir sınıflandırma modeli geliştirmektir. Dünyanın yedi harikasını görsel veriler üzerinden otomatik olarak tanımlayabilen bir sistem oluşturmak, hem kültürel mirasların korunması ve tanıtımına katkı sağlamak hem de görüntü işleme teknolojilerinin uygulama alanlarını genişletmek açısından önem taşımaktadır. Çalışma kapsamında, "Wonders of the World Image Classification" veri seti kullanılarak derin öğrenme tabanlı bir model eğitilmiş ve bu modelin doğruluk oranları ile sınıflandırma başarımı değerlendirilmiştir. Bu amaç doğrultusunda, veriler üzerinde ön işleme adımları uygulanmış, model eğitimi gerçekleştirilmiş ve sonuçlar analiz edilerek sistemin etkinliği test edilmiştir. Çalışma, gelecekteki benzer projeler için referans niteliği taşımayı ve kültürel eserlerin dijital ortamlarda tanınmasına yönelik çözüm önerileri sunmayı hedeflemektedir.

Kullanılan Araçlar ve Yöntemler

Bu çalışmada, Python programlama dili kullanılarak veri analizi ve model eğitimi gerçekleştirilmiştir. Kullanılan başlıca araçlar arasında Ultralytics ve NVIDIA GPU bulunmaktadır. Notebook üzerinde GPU desteği sağlamak amacıyla NVIDIA GPU kontrolü yapılmıştır. Ultralytics kütüphanesi, derin öğrenme modellerinin eğitimi ve değerlendirilmesinde etkin olarak kullanılmıştır. Çalışmada verilerin işlenmesi, görselleştirilmesi ve analiz edilmesi için modern veri bilimi araçları kullanılmıştır.

Çalışmanın Yapılışı

Bu çalışmada, veri ön işleme ve veri çoğaltma (augmentation) yöntemleri detaylı ve sistematik bir şekilde uygulanmıştır. Ön işleme aşamasında, veri setindeki görüntülerin yönlendirilmesi otomatik olarak ayarlanmış (Auto-Orient), bu sayede yanlış yönlendirilmiş görüntülerin yol açabileceği hatalar ortadan kaldırılmıştır. Görseller, model girdilerinin sabit bir boyuta sahip olması ve işlem performansının artırılması amacıyla 640x640 piksel boyutuna ölçeklendirilmiştir (Resize). Ayrıca, veri setinin renk bilgisine ihtiyaç duymaması nedeniyle görseller gri tonlamaya dönüştürülmüş (Grayscale), böylece gereksiz bilgilerin filtrelenmesi sağlanmıştır. Sınıf düzenleme adımları kapsamında veri setindeki 20 sınıf yeniden düzenlenmiş ve bir sınıf çıkarılmıştır. Bu işlem, modelin belirli bir görev için özelleştirilmesini sağlamıştır. Veri çoğaltma aşamasında, her eğitim örneği için üç yeni örnek üretilmiştir. Bu yöntem, modelin overfitting'e karşı dayanıklılığını artırmış ve genelleme yeteneğini güçlendirmiştir. Yatay ve dikey çevirme (Flip) işlemleri, modelin görüntüleri farklı perspektiflerden algılayabilmesine olanak sağlamıştır. Görseller -20° ile +20° arasında döndürülmüş (Rotation) ve bu sayede modelin nesneleri farklı açılarda tanıma yeteneği geliştirilmiştir. Görüntülere 1.4 piksel değerine kadar bulanıklık (Blur) uygulanarak düşük kaliteli görsellerle karşılaşıldığında modelin performansının korunması hedeflenmiştir. Ayrıca, %1.88 oranında gürültü (Noise) eklenmiş, bu işlem gerçek dünyadaki bozuk veya düşük kaliteli görüntülerle başa çıkma kapasitesini artırmıştır. Nesne tespiti görevleri için kritik bir işlem olarak, etiketli nesne çerçevelerine (Bounding Box) yatay ve dikey çevirme uygulanmış ve veri artırımı sırasında etiketlerin doğruluğu korunmuştur. Bu adımların tümü, modelin farklı veri koşullarında yüksek performans göstermesi ve genelleme kabiliyetinin artırılması amacıyla gerçekleştirilmiştir.

Çalışmanın Yapılışı

↘

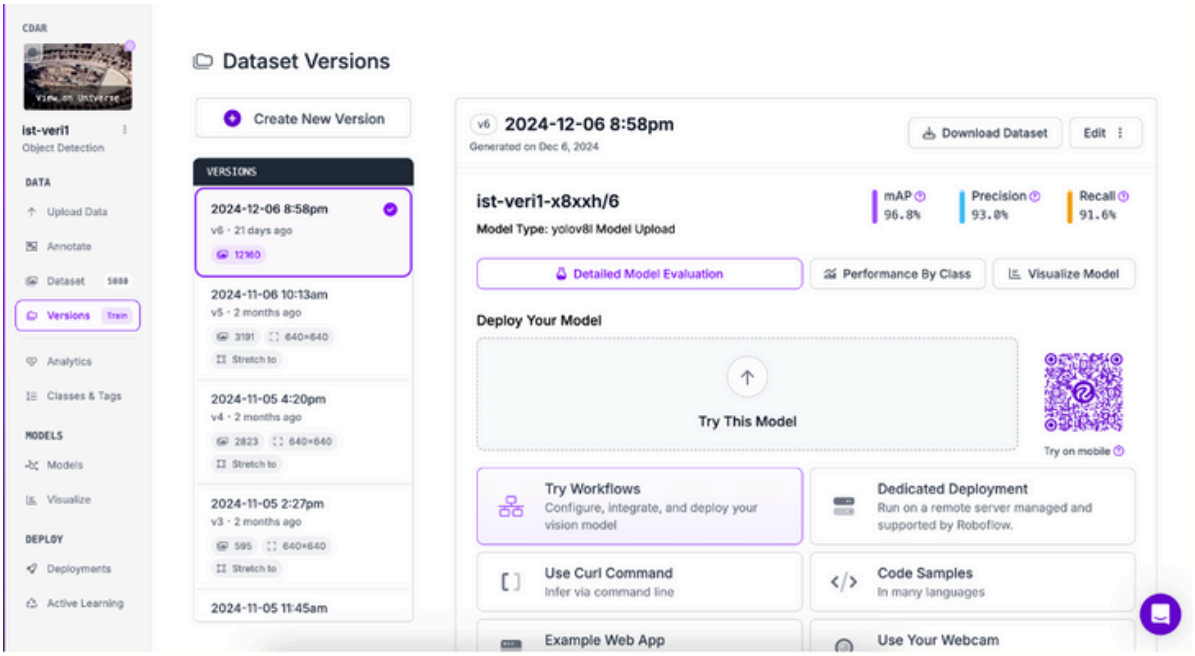
```
[ ]  from ultralytics import YOLO

     from IPython.display import display, Image

[ ]  !pip install roboflow

     from roboflow import Roboflow
     rf = Roboflow(api_key="lOEo0JaDaqQkXIdXNkeA")
     project = rf.workspace("cdar-jrqva").project("ist-veri1-x8xxh")
     version = project.version(6)
     dataset = version.download("yolov8")
```

Bu kısımda mevcut verimiz roboflow platformu kullanılarak etiketlenmiştir ve platformun sunduğu kütüphane ile Python ortamına aktarılmıştır. Aşağıda ise roboflow üzerinde verimizin deploy olduğunu görüyoruz ve artık denemeler yapabilir ya da modelle ilgili skorları inceleyebiliriz. Sonraki adım olarak bu veri setini download kısmından yolo v8 formatından çıkartarak artık model eğitimine başlayabiliriz.



Çalışmanın Yapılışı

↘

```
!yolo task=detect mode=train model=yolov8l.pt data={dataset.location}/data.yaml epochs=20 imgsz=640
```

Yukarıdaki, kodda görüldüğü üzere detect modunda çalıştırıyoruz ve model olarak yolov8l modelini kullanıyoruz. Modele verinin yolunu verip artık son işleme geçiyoruz. Burada kaç epoch ile modelin eğitileceğini seçiyoruz ve modele girecek görsellerin boyutunun kararını veriyoruz. Her epoch bitiminde bize modelin başarı metrikleri ile ilgili skorlar sunuluyor. Bu bir train sürecidir ve bu süreç tamamlandığında aşağıda verilen çıktıları alıyoruz.

```
Starting training for 20 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       1/20      10.3G     0.9694      1.891      1.504         30        640: 100% 664/664 [09:07<00:00,  1.21it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 32/32 [00:22<00:00,  1.40it/s]
                   all       1008       1008      0.647      0.371      0.389      0.202

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       2/20      10.3G      1.009      1.571      1.518         33        640: 100% 664/664 [09:02<00:00,  1.22it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 32/32 [00:20<00:00,  1.53it/s]
                   all       1008       1008      0.761      0.471      0.567      0.292

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       3/20      10.4G     0.9512      1.404      1.463         35        640: 100% 664/664 [08:59<00:00,  1.23it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 32/32 [00:20<00:00,  1.52it/s]
                   all       1008       1008      0.801      0.496      0.539      0.324

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       4/20      10.4G     0.8689      1.257      1.397         35        640: 100% 664/664 [08:57<00:00,  1.24it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 32/32 [00:21<00:00,  1.51it/s]
                   all       1008       1008      0.792      0.561      0.682        0.4
```

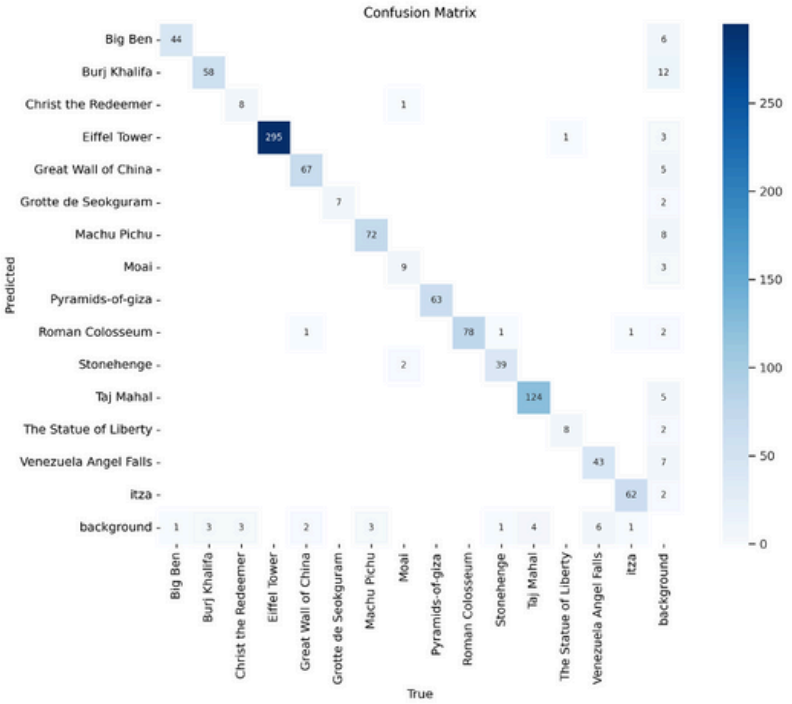
```
Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.2.103 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 268 layers, 43,618,173 parameters, 0 gradients, 164.9 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 32/32 [00:23<00:00,  1.37it/s]
                   all       1008       1008      0.931      0.915      0.968      0.743
                Big Ben         45         45      0.916      0.889      0.962       0.68
           Burj Khalifa         60         61      0.843      0.869       0.94      0.791
   Christ the Redeemer         11         11      0.985      0.727      0.953      0.774
           Eiffel Tower        292        295       0.99      0.998      0.993      0.896
   Great Wall of China         70         70      0.953      0.957      0.963      0.706
   Grotte de Seokguram          7          7      0.847          1      0.995      0.787
          Machu Pichu          75         75      0.936       0.92      0.969      0.596
                 Moai          12         12      0.852      0.833      0.947      0.764
      Pyramids-of-giza         63         63      0.989          1      0.995      0.831
      Roman Colosseum         78         78       0.94      0.987      0.986      0.872
           Stonehenge         41         41      0.951      0.951      0.991      0.739
            Taj Mahal        124        128      0.978      0.969      0.971       0.81
 The Statue of Liberty          9          9      0.898      0.976      0.984      0.635
 Venezuela Angel Falls         49         49      0.918      0.683      0.882      0.421
                 itza         64         64      0.969      0.972      0.987      0.843
Speed: 0.2ms preprocess, 16.7ms inference, 0.0ms loss, 1.8ms postprocess per image
```

Çalışmanın Sonuçları ve Yorumları

↘

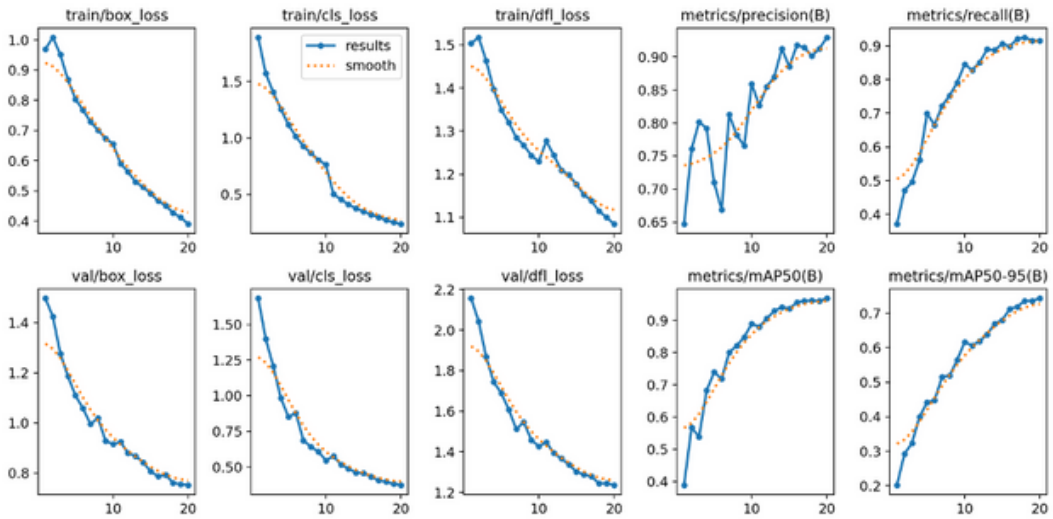
Bu sonuçlara bakıldığında, modelin genel olarak oldukça iyi bir performans sergilediğini söyleyebiliriz. Özellikle "Eiffel Tower" ve "Taj Mahal" gibi sınıflarda neredeyse mükemmel sonuçlar elde edilmiştir; bu sınıflarda accuracy ve recall oranları çok yüksektir ve MAP50 değerleri sırasıyla %99.8 ve %96.6 gibi üst düzey performansı yansıtmaktadır. Bu durum, modelin bu tür sınıfları ayırt etmede son derece başarılı olduğunu göstermektedir. Ancak, bazı sınıflarda performansın diğerlerine kıyasla daha düşük olduğu gözlenmiştir. Örneğin, "Big Ben" için doğruluk oranı %88.9 iken "Christ the Redeemer" sınıfında %72.7'ye kadar düşmüştür; bu da modelin bu nesneleri sınıflandırırken daha fazla hata yaptığını göstermektedir. Benzer şekilde, "Moai" ve "Grotte de Seokguram" gibi sınıflarda da hem doğruluk hem de MAP50-95 değerleri düşük kalmış, bu durum modelin bu sınıflar için iyileştirmeye ihtiyaç duyduğunu işaret etmiştir. Özellikle "Christ the Redeemer" ve "Grotte de Seokguram" gibi sınıflar için veri setine daha fazla örnek eklenmesi veya veri çeşitliliğinin artırılması faydalı olabilir. Model genel olarak %96 civarında bir MAP50 oranı elde etmiş, bu oldukça iyi bir sonuçtur, ancak MAP50-95 oranının %74 seviyesinde kalması, daha sıkı bir doğruluk eşiğinde model performansının düştüğünü göstermektedir. Bu durum, modelin genel performansının iyileştirilmesi ve sıkı doğruluk eşiklerinde daha dayanıklı hale getirilmesi gerektiğine işaret etmektedir. Genel olarak, model güçlü bir performans sergilemekle birlikte, belirli sınıflarda ek veri ve optimizasyon ile daha da geliştirilme potansiyeline sahiptir.

Çalışmanın Sonuçları ve Yorumları



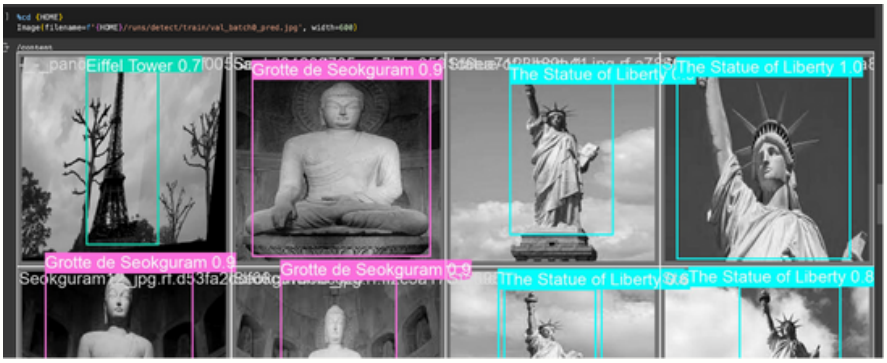
Bu confusion matrix, modelin sınıflandırma performansını detaylı bir şekilde incelememizi sağlar. Matriste, her bir gerçek sınıfın ve modelin tahmin ettiği sınıfın (Predicted) değerleri yer almaktadır. Diyagonal çizgi üzerindeki değerler, modelin doğru sınıflandırdığı örnekleri ifade ederken, diğer hücrelerde yer alan değerler yanlış sınıflandırmaları temsil etmektedir. Model, "Eiffel Tower" sınıfında 295 örneği doğru sınıflandırmış, bu da modelin bu sınıfı tanımada oldukça başarılı olduğunu göstermektedir. Benzer şekilde, "Taj Mahal" 124 doğru tahmin, "Roman Colosseum" 78 doğru tahmin ve "Great Wall of China" 67 doğru tahmin sınıflarında da yüksek doğruluk oranları görülmektedir. Bu durum, modelin bu nesneleri ayırt etmede güçlü olduğunu göstermektedir. Bazı sınıflar için modelin karışıklık yaşadığı ve yanlış tahminler yaptığı dikkat çekmektedir. Örneğin, "Big Ben" sınıfında 6 örnek "Burj Khalifa" olarak, "Burj Khalifa" sınıfında 12 örnek "Eiffel Tower" olarak yanlış tahmin edilmiştir. Benzer şekilde, "Christ the Redeemer" sınıfında 1 örnek yanlışlıkla "background" olarak sınıflandırılmıştır. Bu, modelin bazı sınıflar arasındaki benzerliklerden etkilenebileceğini ve daha fazla çeşitlendirilmiş veri ile eğitilmesi gerektiğini göstermektedir. Bazı sınıflarda tahmin sayılarının düşük olması, bu sınıfların model tarafından yeterince öğrenilmediğini veya veri setinde bu sınıflardan yeterli örnek bulunmadığını işaret etmektedir. Örneğin, "Moai" sınıfında yalnızca 9 doğru tahmin yapılmış ve 3 örnek başka sınıflara yanlış sınıflandırılmıştır. Benzer şekilde, "Grotte de Seokguram" sınıfında 7 doğru tahmin yapılırken, 3 örnek "background" olarak Model genel olarak güçlü bir performans sergilese de, bazı sınıflarda, özellikle birbirine benzer nesnelerin olduğu durumlarda, yanlış sınıflandırma oranı artmaktadır. "Burj Khalifa" ve "Eiffel Tower" arasında yaşanan karışıklık, bu nesnelerin benzer görsel özelliklere sahip olmasından kaynaklanabilir. Ayrıca, az sayıda örneği olan "Moai" ve "Grotte de Seokguram" gibi sınıflarda düşük doğruluk oranları, modelin bu sınıfları daha iyi öğrenebilmesi için veri setinin zenginleştirilmesi gerektiğini göstermektedir.

Çalışmanın Sonuçları ve Yorumları



Modelin eğitim süreci boyunca kayıp değerlerinin sürekli azalması ve performans metriklerinin düzenli olarak iyileşmesi, eğitimin başarılı bir şekilde ilerlediğini göstermektedir. Eğitim ve doğrulama kayıplarının birbirine yakın seyretmesi, modelin overfitting (aşırı uyum) yapmadığını ve genelleme yeteneğinin güçlü olduğunu ortaya koymaktadır. Ayrıca, mAP ve precision gibi metriklerdeki yükseliş, modelin nesne algılama ve sınıflandırma görevlerinde güvenilir bir performans sergilediğini ifade etmektedir. Bu bulgular, modelin hem eğitim hem de doğrulama verileri üzerinde sağlam bir performans sergilediğini ve süreç boyunca etkili bir şekilde optimize edildiğini göstermektedir.

Modelin kendi içinde nasıl göründüğüne dair bir görsel.



Çalışmanın Sonuçları ve Yorumları

↘

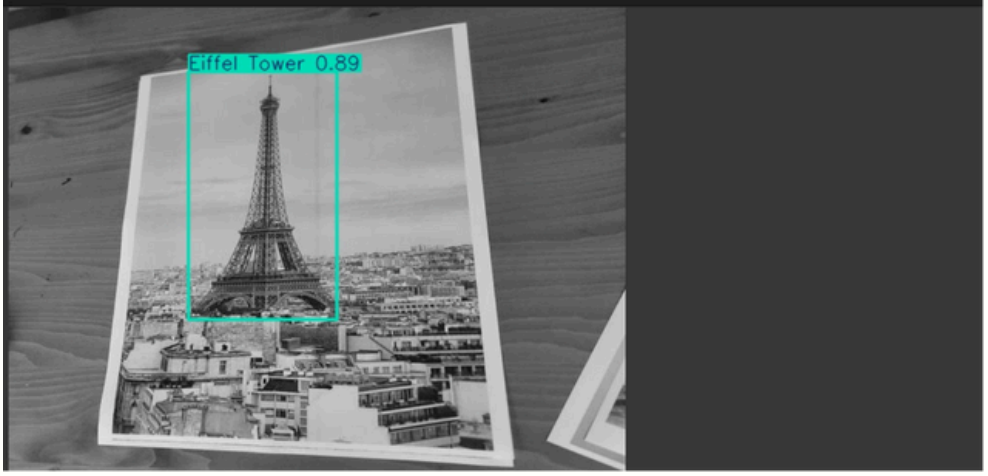
Modeli test verilerini ne kadar iyi tahmin edebildiğini görmek için çalıştırıyoruz.

```
%cd {HOME}
!yolo task=detect mode=predict model={HOME}/runs/detect/train/weights/best.pt conf=0.25 source={dataset.location}/test/images save=true
```

Modeli predict moduna aldık ve trainden oluşan ağırlık dosyasını da uzantı olarak verdik. Sonrasında "conf" ile bizim için bir alt sınır belirledik. Alt sınırı daha daha sağlam tahminler yapabilmek için ekledik.

```
import glob
from IPython.display import Image, display

for image_path in glob.glob(f'{HOME}/runs/detect/predict/*.jpg')[:26]:
      display(Image(filename=image_path, height=600))
      print("\n")
```



İlk predict sonuçlarına bakıldığında modele verilen yapı görsellerinden, yapıları tanıdığını görebiliyoruz. Modelin gayet başarılı çalıştığı sonucuna varabiliriz.

KAYNAKÇA

↘

https://www.kaggle.com/datasets/balabaskar/wonders-of-the-world-image-classification