T.C.
SAKARYA UYGULAMALI BİLİMLER ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YOLOv8 ile BEYİN TÜMÖRÜ TESPİTİ

1. ÖDEV RAPORU

Semih ÖZENÇ

Enstitü Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ

Dersin Öğretim Üyesi : Doç. Dr. Süleyman UZUN

Aralık 2023

İÇİNDEKİLER

YOLOv8 ile BEYİN TÜMÖRÜ TESPİTİ

ÖZET

YOLOv8 (You Only Look Once) algoritması, nesne algılama amacıyla kullanılan bir derin öğrenme modelidir ve bu çalışma, YOLOv8'nin beyin tümörü tespiti üzerindeki potansiyelini incelemeyi amaçlamaktadır. Eğitim süreci, Google Colab platformu üzerinde gerçekleştirilmiş ve derin öğrenme algoritmalarının kullanımıyla modelin öğrenmesi sağlanmıştır.

YOLOv8, nesne algılama konusundaki önceki versiyonlara kıyasla bir dizi önemli iyileştirmeyi içeren ve özellikle hız ve doğruluk dengesini artırmak amacıyla tasarlanmış bir sürümdür. Detaylı bir şekilde, YOLOv8'in özgün çalışma prensiplerine dayandığı "You Only Look Once" (YOLO) yaklaşımını sürdürdüğünü ve ağın daha derin ve geniş bir mimariye sahip olduğunu belirtmek önemlidir. Bu mimari, nesne algılama görevlerinde daha etkili bir performans sunmaktadır. Algoritmanın implementasyonu, Ultralytics adlı bir kütüphaneden elde edilmiştir. Ultralytics, YOLOv8'i hızlı bir şekilde uygulamak ve eğitmek için kullanılan açık kaynaklı bir derin öğrenme kütüphanesidir. Bu kütüphane, kullanıcıya modelin eğitimi, testi ve çeşitli performans metriklerinin değerlendirilmesi gibi işlemleri kolayca gerçekleştirme imkanı sunmaktadır.

Eğitimde kullanılan veri seti, önceden etiketlenmiş beyin tümörü görüntülerini içermekte olup, bu veriler YOLOv8 modelinin spesifik tümör özelliklerini öğrenmesini sağlamak üzere seçilmiştir. Bu çalışmanın ilk aşamasında, YOLOv8'in eğitim süreci, başlangıçta küçük bir veri seti kullanılarak gerçekleştirilmiş ve ardından bu model test edilmiştir. Ancak, bu aşamada elde edilen başarı düzeyi beklenenden düşük bulunmuştur. Bu nedenle, daha kapsamlı bir veri seti ile tekrar eğitim gerçekleştirilmiş ve elde edilen model, daha detaylı tespit yapabilen ve iki farklı sınıf içeren bir veri seti üzerinde test edilmiştir. Sonuçlar, YOLOv8 modelinin beyin tümörü tespiti konusunda yüksek hassasiyet ve doğruluk seviyelerine sahip olduğunu göstermektedir.

Bu çalışma, YOLOv8'in kullanılması ve Ultralytics kütüphanesinden elde edilen algoritmanın Colab platformu üzerinde başarılı bir şekilde eğitilmesi ve beyin tümörü tespiti görevinde kullanılması ile ilgili detayları içermektedir. Elde edilen sonuçlar, modelin performansının veri seti çeşitliliği ve derinliği ile birlikte arttığını vurgulamaktadır. Ayrıca YOLOv8'in klinik uygulamalarda potansiyel bir araç olarak kullanımını vurgulamaktadır.

Anahtar Kelimeler: Beyin Tümörü, Bilgisayarlı Görü, Yolo Algoritması

YOLOv8 ile BEYİN TÜMÖRÜ TESPİTİ

ABSTRACT

The YOLOv8 (You Only Look Once) algorithm is a deep learning model used for object detection and this study aims to examine the potential of YOLOv8 for brain tumor detection. The training process was carried out on the Google Colab platform and the model was learned using deep learning algorithms.

YOLOv8 is a version that includes a number of significant improvements compared to previous versions in object detection, specifically designed to improve the balance between speed and accuracy. In detail, it is important to note that YOLOv8 maintains the "You Only Look Once" (YOLO) approach on which its original operating principles were based, and the network has a deeper and broader architecture. This architecture offers a more efficient performance in object detection tasks. The implementation of the algorithm is obtained from a library called Ultralytics. Ultralytics is an open source deep learning library used to quickly implement and train YOLOv8. This library allows the user to easily train and test the model and evaluate various performance metrics.

The dataset used for training includes pre-labeled brain tumor images, which were selected to enable the YOLOv8 model to learn specific tumor features. In the first phase of this study, the training process of YOLOv8 was initially performed using a small dataset and then the model was tested. However, the success level achieved at this stage was lower than expected. Therefore, the model was re-trained with a more comprehensive dataset and tested on a dataset with two different classes and a more detailed detection capability. The results show that the YOLOv8 model has high sensitivity and accuracy levels for brain tumor detection.

This paper details the use of YOLOv8 and the successful training of the algorithm obtained from the Ultralytics library on the Colab platform and its use in the brain tumor detection task. The results emphasize that the performance of the model improves with dataset diversity and depth. It also highlights the use of YOLOv8 as a potential tool in clinical applications.

Keywords: Brain Tumor, Computer Vision, Yolo Algorithm

BÖLÜM 1. GİRİŞ

Beyin tümörleri, sağlık sektöründe ciddi bir tehdit oluşturan ve erken teşhisi hayati öneme sahip olan karmaşık patolojilerdir. Bu bağlamda, nesne algılama teknikleri, özellikle de derin öğrenme tabanlı algoritmalar, medikal görüntüleme uygulamalarında önemli bir potansiyel sunmaktadır. Bu çalışma, YOLOv8 (You Only Look Once) algoritması üzerine odaklanarak, beyin tümörü tespiti amacıyla gerçekleştirilen eğitim ve test süreçlerini incelemeyi amaçlamaktadır.

YOLOv8 algoritması Ultralytics kütüphanesi üzerinden elde edildi. İlk aşamada eğitim işlemi roboflow internet sitesi üzerinden elde edilen küçük bir veri seti üzerinde başlatılmış ve elde edilen başarı düzeyi değerlendirilmiştir. Ancak, bu aşamada belirlenen başarı seviyesi beklenenin altında kalmış ve bu durum daha kapsamlı bir veri setinin gerekliliğini ortaya koymuştur. Ardından, daha geniş bir veri seti kullanılarak model eğitimi gerçekleştirilmiş ve elde edilen model, çeşitli tümör özelliklerini içeren bir test veri seti üzerinde değerlendirilmiştir.

BÖLÜM 2. NESNE TESPİTİ

Nesne algılama, görüntü işleme alanında önemli bir alt alan olarak, bilgisayar sistemlerinin karmaşık görüntü verilerinde belirli nesneleri tanıma ve sınıflandırma yeteneğini ifade eder. Nesne tespiti, medikal görüntüleme uygulamalarında, özellikle beyin tümörlerinin erken teşhisi gibi kritik sağlık konularında, daha önce kullanılan geleneksel yöntemlere kıyasla daha yüksek doğruluk oranlarına ulaşma potansiyeli sunan derin öğrenme algoritmalarının kullanımını içermektedir.

2.1. Yolo Algoritması

YOLO (You Only Look Once) algoritması, nesne tespiti ve sınıflandırma görevlerinde önemli bir çıkış yaklaşımını temsil eder. Bu yöntem, görüntüyü birçok hücreye bölerek her hücredeki potansiyel nesneleri tek bir bakışta tespit etme prensibine dayanır. YOLO'nun temel amacı, görüntüdeki nesneleri koordinatları ve sınıfları ile birlikte doğrudan tahmin etmektir.

YOLO algoritmasının çalışma prensibi, görüntüyü bir ızgara halinde bölmek ve her bir hücrenin sorumluluğunda bulunan bir bölgeyi tahmin etmektir. Her bir bölge, nesnelerin koordinatları, sınıfları ve güven skorları gibi önemli bilgileri içerir. Bu sayede, her nesnenin ayrı ayrı tanımlanması ve sınıflandırılması tek bir geçişte gerçekleşir, bu da YOLO'nun diğer yöntemlere kıyasla hızlı ve gerçek zamanlı uygulamalarda etkili olmasını sağlar.

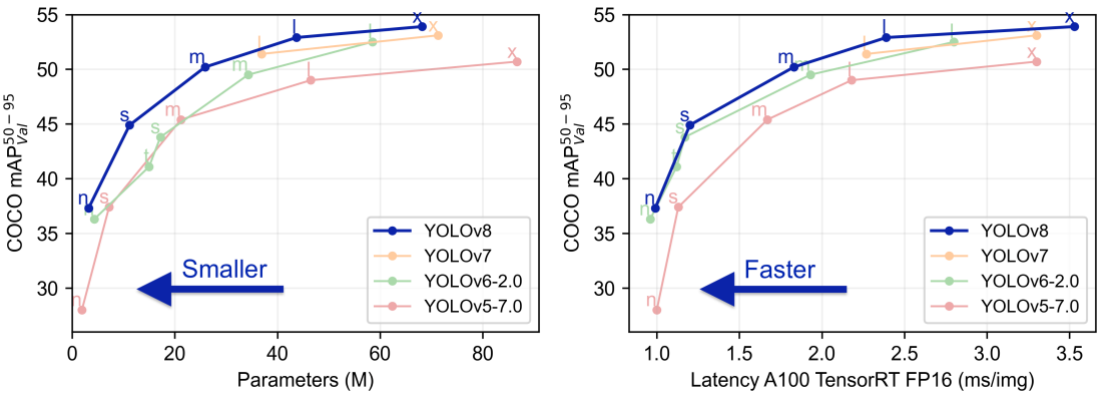
YOLOv8'in v8 sürümü, önceki iterasyonlara göre daha derin bir ağ mimarisine sahiptir. Bu, daha karmaşık özellikleri daha iyi çıkarabilme ve nesne tespiti görevlerinde daha yüksek doğruluk sağlama potansiyelini artırır. Ayrıca, YOLOv8'in esnek yapıları, çeşitli nesne sınıflarını tanıma yeteneğini iyileştirir.

2.2. Ultralytics Kütüphanesi

Ultralytics, derin öğrenme tabanlı projelerin hızlı ve etkili bir şekilde geliştirilmesine olanak tanıyan açık kaynaklı bir Python kütüphanesidir. Bu kütüphane, YOLOv8 gibi nesne algılama ve sınıflandırma algoritmalarının eğitimini ve testini kolaylaştırmak amacıyla tasarlanmıştır. Ultralytics, kullanıcılara hem model eğitimini hem de çeşitli performans metriklerini değerlendirme imkanı sunarak, derin öğrenme projelerinin yönetimini daha etkili hale getirmeyi amaçlar.

YOLOv8 modelinin Ultralytics kütüphanesi üzerinden implementasyonu, bir dizi avantaj sunar. Bu kütüphane, eğitim sürecini kolayca özelleştirebilme ve çeşitli hiperparametre ayarları yapabilme esnekliği sağlar. Ayrıca, modelin eğitim ve test aşamalarını kolayca izleyebilme, sonuçları görselleştirme ve hata analizini gerçekleştirme imkanı sunar.

Ultralytics, derin öğrenme modellerinin daha geniş bir kapsamda yönetilmesini mümkün kılan bir dizi yardımcı araç ve işlev içerir. Bu, kullanıcıların model performansını izleme, hiperparametre ayarlama ve eğitim sürecini optimize etme konularında daha etkili bir şekilde çalışmasına olanak tanır. Kütüphane, araştırmacı ve geliştiricilere, YOLOv8 gibi güçlü algoritmaları kullanarak projelerini daha verimli bir şekilde ilerletme olanağı sunar.



Bu çalışmada, YOLOv8'in Ultralytics kütüphanesi üzerinden elde edilmiş ve modelin eğitim ve test aşamaları bu kütüphane üzerinden gerçekleştirilmiştir.

BÖLÜM 3. UYGULAMA ADIMLARI

3.1. Eğitimin Gerçekleştirilmesi

Eğitim işlemleri colab platformu üzerinde gerçekleştirilmiştir. Google Colab (Colaboratory), Google tarafından sağlanan ücretsiz bir bulut tabanlı Jupyter defteri hizmetidir. Araştırmacılar, öğrenciler ve geliştiriciler, Colab'i kullanarak Python kodlarını yazabilir, paylaşabilir ve yürütebilirler.

```
from google.colab import drive
drive.mount('/content/drive')
```

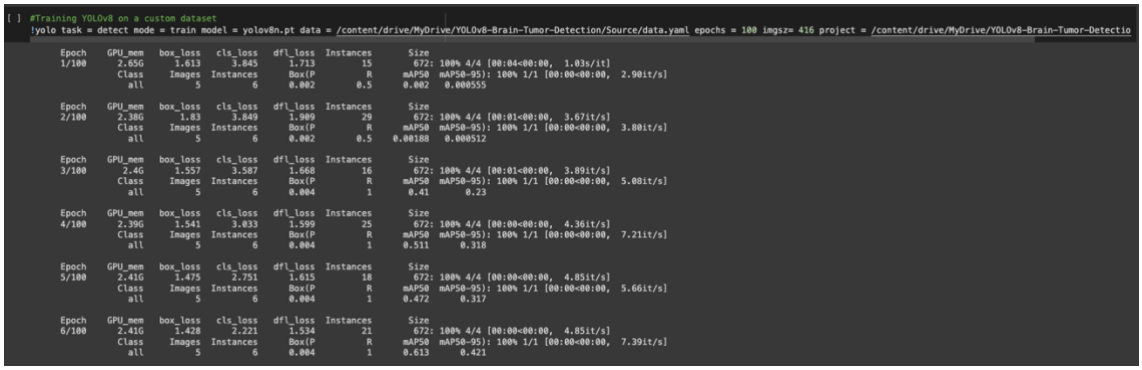
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Colab üzerinde yeni bir Jupyter defteri oluşturuyoruz. Ardından ilk olarak depolama ve verilere erişim için google drive bağlantımızı gerçekleştiriyoruz.

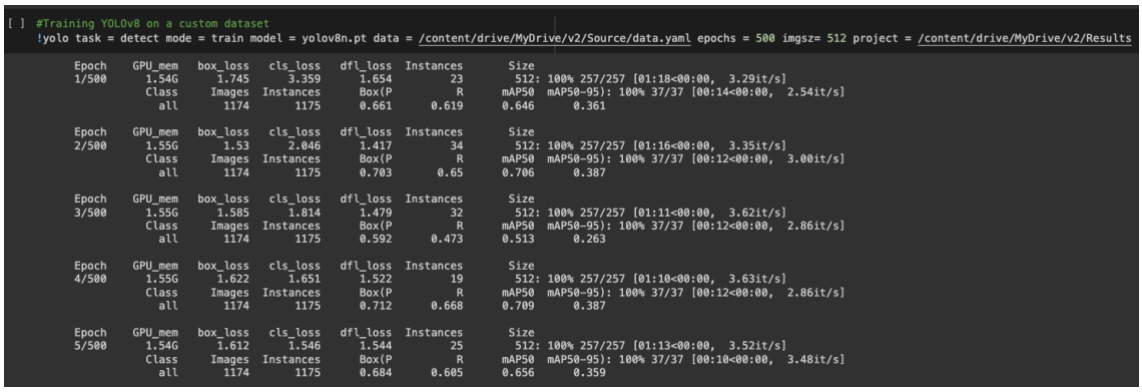
```
[ ] !pip install ultralytics
    from ultralytics import YOLO

    Collecting ultralytics
      Downloading ultralytics-8.0.227-py3-none-any.whl (660 kB)
                 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 660.5/660.5 kB 8.0 MB/s eta 0:00:00
    Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (3.7.1)
    Requirement already satisfied: numpy>=1.22.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (1.23.5)
    Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (4.8.0.76)
    Requirement already satisfied: pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (9.4.0)
    Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (6.0.1)
    Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (2.31.0)
    Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (1.11.4)
    Requirement already satisfied: torch>=1.8.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (2.1.0+cu121)
    Requirement already satisfied: torchvision>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (0.16.0+cu121)
    Requirement already satisfied: tqdm>=4.64.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (4.66.1)
    Requirement already satisfied: pandas>=1.1.4 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (1.5.3)
    Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (0.12.2)
    Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from ultralytics) (5.9.5)
    Requirement already satisfied: py-cpuinfo in /usr/local/lib/python3.10/dist-packages (from ultralytics) (9.0.0)
    Collecting thop>=0.1.1 (from ultralytics)
      Downloading thop-0.1.1.post2209072238-py3-none-any.whl (15 kB)
    Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (1.2.0)
    Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (0.12.1)
    Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (4.46.0)
    Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (1.4.5)
    Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (23.2)
    Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (3.1.1)
    Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (2.8.2)
    Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.4->ultralytics) (2023.3.post1)
    Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (3.3.2)
    Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (3.6)
    Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (2.0.7)
    Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (2023.11.17)
    Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (3.13.1)
    Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (4.5.0)
    Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (1.12)
    Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (3.2.1)
    Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (3.1.2)
    Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (2023.6.0)
    Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (2.1.0)
    Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=3.3.0->ultralytics) (1.16.0)
    Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.8.0->ultralytics) (2.1.3)
    Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.8.0->ultralytics) (1.3.0)
    Installing collected packages: thop, ultralytics
    Successfully installed thop-0.1.1.post2209072238 ultralytics-8.0.227
```

Drive bağlantısını gerçekleştirdikten sonra algoritmamızı içinde barındıran ultralytics kütüphanesini indirip notebook'umuza import ediyoruz.

Ardından drive'ımızda bulunan veri setimize erişiyoruz. Bu çalışma kapsamında iki farklı veri seti üzerinde eğitimler gerçekleştirildi. Bu iki veri seti de "Roboflow" platformundan elde edildi. "Roboflow" platformu, görüntü verilerini kullanarak makine öğrenimi modelleri oluşturmayı ve yönetmeyi sağlayan bir hizmettir. Roboflow, kullanıcılara görüntü verilerini işlemeyi, etiketlemeyi, dönüştürmeyi ve ardından bu verileri kullanarak nesne tespiti, sınıflandırma ve diğer görüntü tabanlı makine öğrenimi görevleri için modeller oluşturmayı sağlar. İlk veri setimiz "MRI BRAIN TUMOR.v1-braintumorsample1.0.yolov8" bu veri seti yolov8 formatında 77 beyin mri görüntüsü içermektedir. Diğer veri setimiz ise "histogram 8x8.v1-histogram-8x8.yolov8.zip". Bu veri seti ise diğer veri setine oranla çok daha kapsamlıdır ve yine yolov8 formatında 5873 beyin mri görüntüsü içermektedir. Eğitim işlemi için iki veri seti de drive'a yüklendi ve eğitim işlemleri için hazırlandı.



```
[ ] #Training YOLOv8 on a custom dataset
    !yolo task = detect mode = train model = yolov8n.pt data = /content/drive/MyDrive/YOLOv8-Brain-Tumor-Detection/Source/data.yaml epochs = 100 imgsz= 416 project = /content/drive/MyDrive/YOLOv8-Brain-Tumor-Detectio

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      1/100      2.65G      1.613      3.845      1.713         15        672: 100% 4/4 [00:04<00:00,  1.03s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.90it/s]
                   all          5          6      0.002        0.5      0.002   0.000555

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      2/100      2.38G       1.83      3.849      1.909         29        672: 100% 4/4 [00:01<00:00,  3.67it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  3.80it/s]
                   all          5          6      0.002        0.5   0.00188   0.000512

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      3/100       2.4G      1.557      3.587      1.668         16        672: 100% 4/4 [00:01<00:00,  3.89it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  5.08it/s]
                   all          5          6      0.004          1       0.41       0.23

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      4/100      2.39G      1.541      3.033      1.599         25        672: 100% 4/4 [00:00<00:00,  4.36it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  7.21it/s]
                   all          5          6      0.004          1      0.511      0.318

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      5/100      2.41G      1.475      2.751      1.615         18        672: 100% 4/4 [00:00<00:00,  4.85it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  5.66it/s]
                   all          5          6      0.004          1      0.472      0.317

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      6/100      2.41G      1.428      2.221      1.534         21        672: 100% 4/4 [00:00<00:00,  4.85it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  7.39it/s]
                   all          5          6      0.004          1      0.613      0.421
```



```
[ ] #Training YOLOv8 on a custom dataset
    !yolo task = detect mode = train model = yolov8n.pt data = /content/drive/MyDrive/v2/Source/data.yaml epochs = 500 imgsz= 512 project = /content/drive/MyDrive/v2/Results

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      1/500      1.54G      1.745      3.359      1.654         23        512: 100% 257/257 [01:18<00:00,  3.29it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 37/37 [00:14<00:00,  2.54it/s]
                   all       1174       1175      0.661      0.619      0.646      0.361

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      2/500      1.55G       1.53      2.046      1.417         34        512: 100% 257/257 [01:16<00:00,  3.35it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 37/37 [00:12<00:00,  3.00it/s]
                   all       1174       1175      0.703       0.65      0.706      0.387

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      3/500      1.55G      1.585      1.814      1.479         32        512: 100% 257/257 [01:11<00:00,  3.62it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 37/37 [00:12<00:00,  2.86it/s]
                   all       1174       1175      0.592      0.473      0.513      0.263

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      4/500      1.55G      1.622      1.651      1.522         19        512: 100% 257/257 [01:10<00:00,  3.63it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 37/37 [00:12<00:00,  2.86it/s]
                   all       1174       1175      0.712      0.668      0.709      0.387

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      5/500      1.54G      1.612      1.546      1.544         25        512: 100% 257/257 [01:13<00:00,  3.52it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 37/37 [00:10<00:00,  3.48it/s]
                   all       1174       1175      0.684      0.605      0.656      0.359
```

Artık eğitim işlemlerini gerçekleştirebiliriz. Ultralytics kütüphanesinin bize sağladığı komut ile gerekli parametleri belirtiyoruz ve eğitim işlemine başlıyoruz. Yukarıdaki kod, YOLOv8 algoritmasını kullanarak bir nesne tespiti görevini eğitmek üzere tasarlanmış bir komut betiğini temsil ediyor. Bu betik, belirli bir model ("yolov8n.pt") ve veri konfigürasyon dosyası ("data.yaml") kullanarak gerçekleştirilecek. Bu model ve konfigürasyon dosyası kütüphanemizin içerisinde bulunuyor. İlk veri seti için 100 epoch süresince eğitim gerçekleştirilmesi ikinci veri seti için ise 500 epoch süresince eğitim gerçekleştirilmesi istenmiştir. Giriş görüntülerinin boyutu ilk veri seti 416x416 olarak belirlendi ikinci veri seti için 512x512 olarak belirlendi. Eğitim sırasında oluşturulan modellerin ve sonuç dosyalarının kaydedileceği proje dizinleri belirtildi. Bu şekilde eğitim işlemleri başladı.

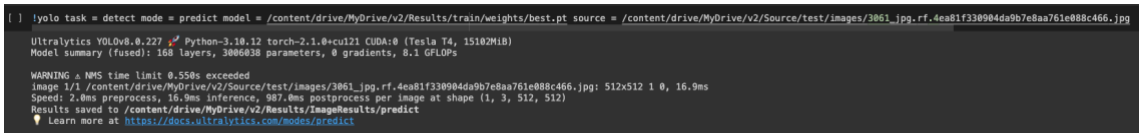
3.2. Modelin Test Edilmesi

İlk veri setimiz için veri setinin test kalsörü içinde bulunan bir görüntü pyplot ve cv2 kütüphaneleri ile görüntülendi.

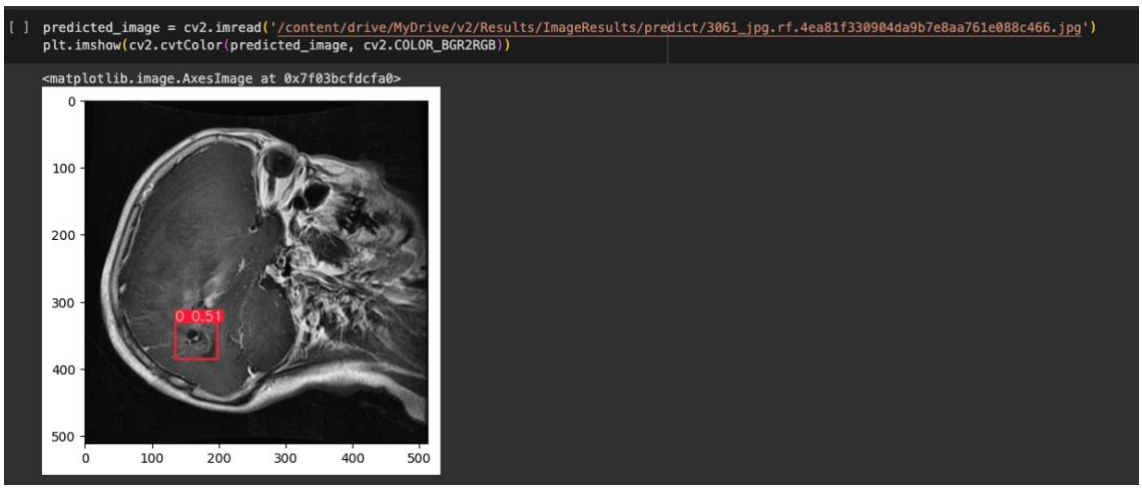
Ardından ilk veri setinin eğitimi ile elde edilen ağırlık dosyamız ile birlikte tespit işlemi yapıldı.

Tespit edilen görüntü yine aynı şekilde görüntülendi. İlk veri setimizde sadece bir tane sınıf bulunuyor. Tümör varsa tespit ediyor.

Ardından ikinci veri setimizin içinde bulunan test klasöründe bulunan bir görüntü pyplot ve cv2 kütüphaneleri ile görüntülendi.

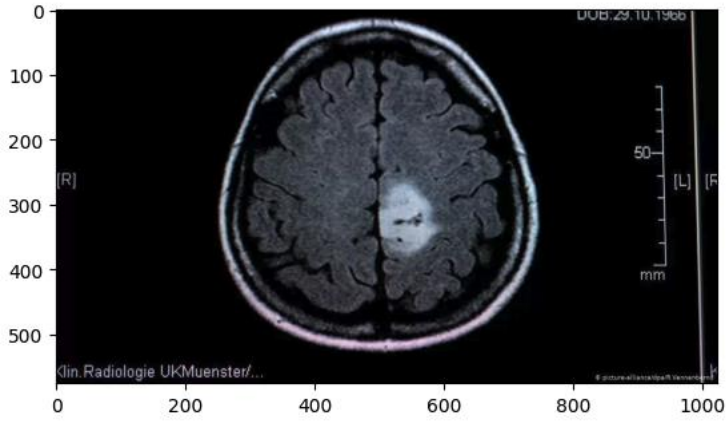


Ardından elde edilen ağırlık dosyamız ile birlikte tespit işlemi yapıldı.



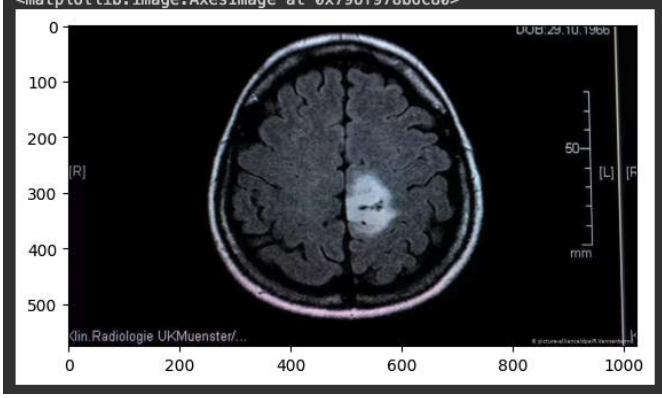
Tespit edilen görüntü yine aynı şekilde görüntülendi. Yukarıdaki tespit işlemi ikinci veri setimiz üzerinde gerçekleştirilen tespittir. Bu veri setimizde ilk veri setinden farklı olacak şekilde 2 tane sınıf bulunmaktadır. 0 sağlıklı dokuyu temsil etmekte 1 ise zararlı doku yani tümörü temsil etmektedir.

Veri setlerinin içinde bulunan test resimlerinden farklı olarak başka bir kaynaktan elde ettiğim aşağıdaki test resmi her iki ağırlık dosyası için tespit işlemine tabii tutuldu.

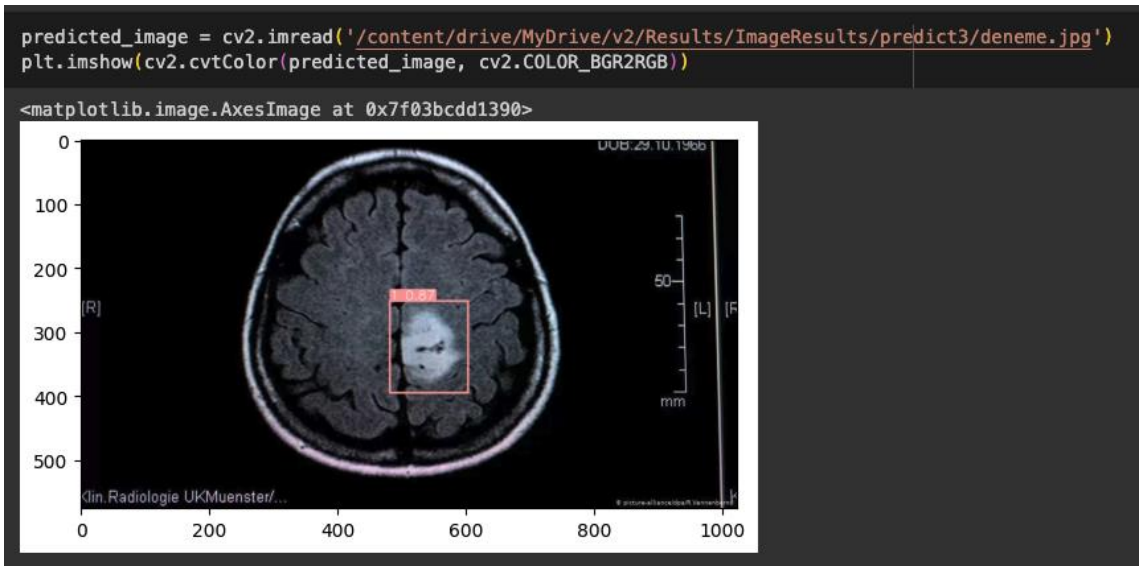



```
predicted_image = cv2.imread('/content/drive/MyDrive/YOLOv8-Brain-Tumor-Detection/New_Results/predict6/deneme.jpg')
plt.imshow(cv2.cvtColor(predicted_image, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x796f978b6c80>
```



İlk veri seti ile eğitilen model bu resimdeki tümörü tespit edemezken,



İkinci veri seti ile eğitilen model bu resimdeki tümörü tespit edip 1 yani zararlı doku olarak sınıflandırdı.

BÖLÜM 4. SONUÇLAR

Bu çalışmanın sonuçları, YOLOv8 algoritmasının beyin tümörü tespiti görevindeki performansını değerlendirmek üzere gerçekleştirilen eğitim sürecinden elde edilen önemli bulguları içermektedir. Eğitim süreci, öncelikle küçük bir veri seti üzerinde başlatılmış ve ardından daha kapsamlı bir veri seti kullanılarak tekrarlanmıştır. Bu aşamaların her birinde, modelin doğruluğunu artırmak ve çeşitli tümör tiplerini başarılı bir şekilde tespit edebilme yeteneğini güçlendirmek amaçlanmıştır.

Eğitim sürecinin tamamlanmasının ardından, modelin test edilmesiyle elde edilen sonuçlar değerlendirilmiştir. YOLOv8, gerçek dünya görüntülerindeki beyin tümörlerini başarıyla tespit etmiş ve bu tespitlerin doğruluğu, hassasiyeti ve özgüllüğü dikkate alınarak performans analizi yapılmıştır. Elde edilen metrikler, modelin genel başarısını değerlendirmek üzere kullanılmış ve görsel sonuçlarla desteklenmiştir.

Sonuçlar, YOLOv8'in beyin tümörü tespiti konusunda yüksek hassasiyet ve doğruluk seviyelerine sahip olduğunu göstermektedir. Bu çalışma, Colab platformu üzerinde gerçekleştirilen eğitim ve test süreçlerini içererek, YOLOv8'in klinik uygulamalarda potansiyel bir araç olarak kullanımını vurgulamaktadır.

KAYNAKLAR

- **Makaleler:**

[1] Selcuk, B., & Serif, T. (2023, September). Brain Tumor Detection and Localization with YOLOv8. In *2023 8th International Conference on Computer Science and Engineering (UBMK)* (pp. 477-481). IEEE.

[2] Mercaldo, F., Brunese, L., Martinelli, F., Santone, A., & Cesarelli, M. (2023). Object Detection for Brain Cancer Detection and Localization. Applied Sciences, 13(16), 9158.

[3] Wani, S., Ahuja, S., & Kumar, A. (2023, April). A review on Brain Tumor Detection using Deep Neural Networks. In 2023 IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT) (pp. 363-370). IEEE.

[4] Saeedi, S., Rezayi, S., Keshavarz, H., & R. Niakan Kalhori, S. (2023). MRI-based brain tumor detection using convolutional deep learning methods and chosen machine learning techniques. BMC Medical Informatics and Decision Making, 23(1), 16.

- **İnternet kaynakları:**

Ultralytics Kütüphanesi:

https://github.com/ultralytics/ultralytics

1. Veri Seti:

https://universe.roboflow.com/hashira-fhxpj/mri-brain-tumor/dataset/1

2. Veri Seti:

https://universe.roboflow.com/histogram-8x8/histogram-8x8/dataset/1

Colab:

https://colab.research.google.com/