DERİN ÖĞRENME İLE BEYİN TÜMÖR TESPİTİ

YAZILIM MÜHENDİSLİĞİ

ŞULE KARTAL 22040301049

EMİNE ESRA ÇETİN 22040301014

DİLARA KURTUL 22040301061

DANIŞMAN: CEVAT RAHEBİ

NİSAN 2025

İÇİNDEKİLER

ÖZET

Bu çalışma, manyetik rezonans görüntüleme (MRI) verileri kullanılarak beyin tümörlerinin otomatik bir şekilde tespit edilmesini ve dört temel kategoriye ayrılarak sınıflandırılmasını amaçlamaktadır. Bu kategoriler glioma, meningioma, pituitary tümörleri ile tümör içermeyen (notumor) beyin görüntülerinden oluşmaktadır. Çalışma kapsamında kullanılan derin öğrenme yaklaşımı, Konvolüsyonel Sinir Ağları (CNN) tabanlı bir mimari üzerine inşa edilmiş ve model, %92.7 oranında doğruluk ile sınıflandırma gerçekleştirmiştir. Projenin bir diğer önemli yönü olan model açıklanabilirliği ise Grad-CAM (Gradient-weighted Class Activation Mapping) tekniğiyle sağlanmıştır. Grad-CAM, modelin karar verirken dikkate aldığı alanların görsel olarak sunulmasına imkân tanımış, bu da hem güvenilirliği artırmış hem de modelin karar sürecini daha anlaşılır kılmıştır. Ayrıca, geliştirilen masaüstü grafik kullanıcı arayüzü (GUI) ile kullanıcılar bir MRI görüntüsü yükleyip sınıflandırma sonucunu hem sayısal hem de görsel olarak kolaylıkla gözlemleyebilmiştir. Rapor kapsamında veri setinin hazırlanması, modelin eğitimi, değerlendirme metrikleri, Grad-CAM analizleri ve arayüz tasarımı detaylı bir şekilde ele alınmıştır.

ABSTRACT

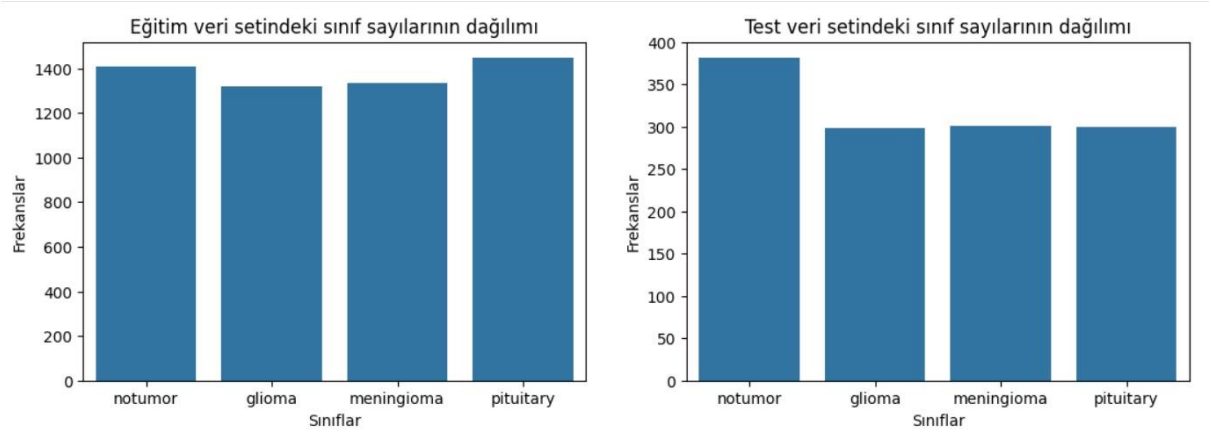
This study aims to automatically detect and classify brain tumors into four basic categories using magnetic resonance imaging (MRI) data. These categories consist of glioma, meningioma, pituitary tumors and non-tumor (notumor) brain images. The deep learning approach used in the study was built on a Convolutional Neural Network (CNN) based architecture and the model performed classification with an accuracy of 92.7%. Another important aspect of the project, model explainability, was achieved with the Grad-CAM (Gradient-weighted Class Activation Mapping) technique. Grad-CAM allowed the model to visually present the areas that the model takes into account when making decisions, which both increased reliability and made the model's decision process more understandable. Furthermore, a desktop graphical user interface (GUI) allowed users to upload an MRI image and easily observe the classification result both numerically and visually. The preparation of the dataset, training of the model, evaluation metrics, Grad-CAM analysis and interface design are discussed in detail in this report.

1.GİRİŞ

Beyin tümörleri, insan sağlığı üzerinde hayati etkileri olabilen karmaşık ve çoğu zaman ölümcül olabilen hastalıklardır. Erken tanı, hastalığın tedavi sürecinde oldukça önemli bir rol oynar. Ancak klasik görüntüleme ve değerlendirme yöntemleri genellikle zamana bağlıdır ve uzman yorumuna ihtiyaç duyar. Bu durum, insan hatası riskini artırmakta ve tanı sürecini uzatabilmektedir. Yapay zeka ve özellikle derin öğrenme tekniklerinin gelişmesiyle birlikte, tıbbi görüntülemede otomatik tanı sistemleri ön plana çıkmıştır. Derin öğrenme algoritmaları, büyük veri kümelerinden öğrenme kabiliyetleri sayesinde karmaşık desenleri tanıyabilir ve yüksek doğruluk oranlarıyla sınıflandırmalar yapabilir. Bu projede, beyin MRI görüntüleri kullanılarak geliştirilen CNN tabanlı bir model ile dört farklı sınıfa ait beyin tümörü görüntülerinin otomatik olarak sınıflandırılması hedeflenmiştir. Bu sınıflandırma sürecine ek olarak, modelin karar alma sürecinin şeffaf hale getirilmesi amacıyla Grad-CAM yöntemi uygulanmış ve kullanıcıya karar mantığı görsel biçimde sunulmuştur. Ayrıca, sistemin erişilebilirliğini ve kullanılabilirliğini artırmak amacıyla bir grafik kullanıcı arayüzü (GUI) geliştirilmiştir. Böylece kullanıcı, herhangi bir kod bilgisine sahip olmadan modeli kullanabilir hale gelmiştir. Proje, veri hazırlığı, model eğitimi, değerlendirme, görselleştirme ve kullanıcı arayüzü tasarımı adımlarını içeren bütüncül bir sistem sunmaktadır.

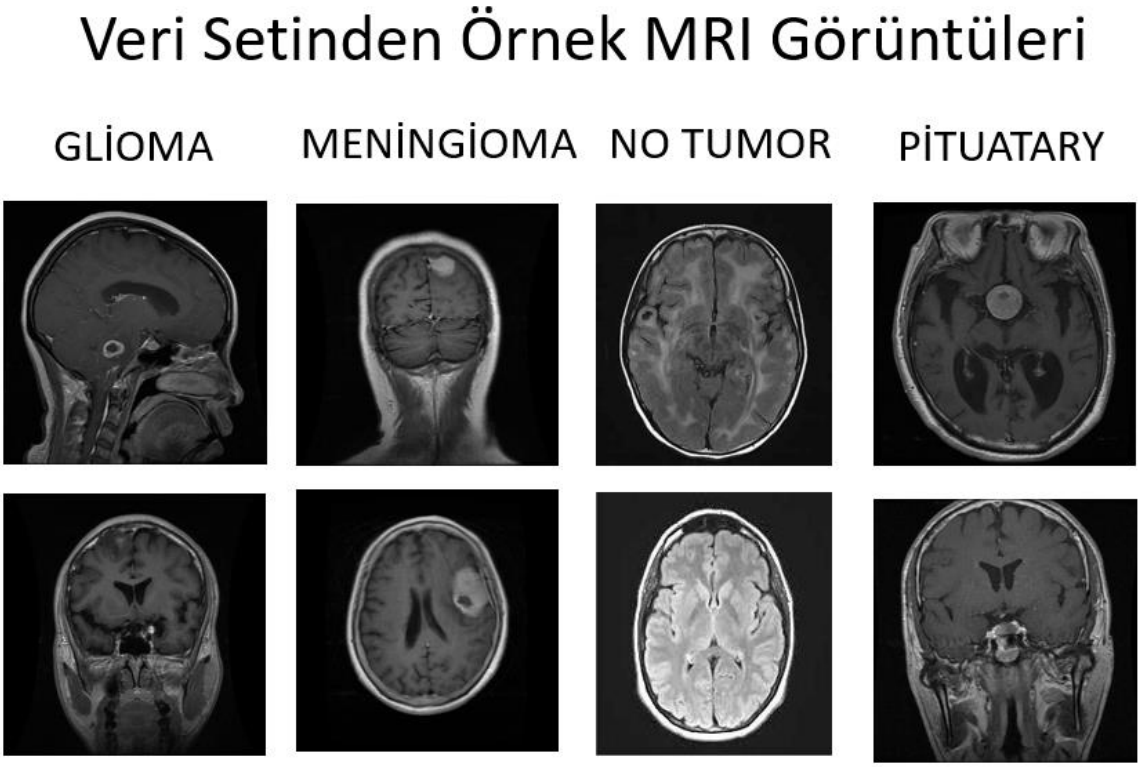
2.MATERİAL & METHODS

Bu çalışmada kullanılan veri seti, dört farklı beyin durumu içeren MRI görüntülerinden oluşmaktadır. Bu sınıflar; notumor (tümör içermeyen beyin), glioma, meningioma ve pituitary tümörleri şeklindedir. Görüntüler Kaggle platformu üzerinden elde edilmiş olup, eğitim ve test seti olarak ikiye ayrılmıştır. Eğitim setinde 5712, test setinde ise 1311 adet görsel yer almaktadır. Görsellerin tamamı 200x200 piksel boyutunda olacak şekilde yeniden boyutlandırılmıştır ve RGB renk formatında kullanılmadan önce normalize edilmiştir.



Yukarıdaki çubuk grafiklerde, eğitim ve test veri setlerinde yer alan dört tümör sınıfının (notumor, glioma, meningioma, pituitary) örnek sayıları gösterilmektedir. Eğitim verileri, sınıflar arasında oldukça dengeli dağılmış olup her bir sınıf yaklaşık 1300–1450 arası görüntü içermektedir. Test veri setinde de benzer bir denge korunmuştur.

Bu dağılım sayesinde model, her sınıftan yeterli sayıda örnek görerek öğrenme sürecini dengeli bir şekilde gerçekleştirmiştir. Veri setindeki bu denge, modelin doğruluk ve genelleme kabiliyetini artıran önemli bir unsurdur.



Modelin doğru ve verimli şekilde eğitilebilmesi için veri ön işleme adımları büyük önem taşımıştır. Öncelikle yinelenen görseller veri setinden çıkarılmış, ardından görseller sayısal tensör formatına dönüştürülerek modele uygun hale getirilmiştir. Etiketler 'one-hot encoding' yöntemi ile dört sınıfa ayrılarak sinir ağında kullanılabilir hale getirilmiştir.

Modelin oluşturulmasında derin öğrenme temelli konvolüsyonel sinir ağı (CNN) mimarisi tercih edilmiştir. Model üç adet Conv2D katmanı, MaxPooling katmanları, Dropout ve Dense katmanlardan oluşmaktadır. ReLU aktivasyon fonksiyonları ara katmanlarda, softmax aktivasyon fonksiyonu ise çıkış katmanında kullanılmıştır.

2.1 Kullanılan Python Kütüphaneleri ve Amaçları

1. TensorFlow

- Sürüm: 2.16.1
- Amaç: Derin öğrenme modelinin kurulması, eğitilmesi ve değerlendirilmesi için kullanılmıştır. Modelin arka planındaki matematiksel işlemleri GPU destekli gerçekleştirme imkânı sağlar. keras API'si TensorFlow'un içinde entegre olarak çalışır.

2. Keras

- Amaç: Yüksek seviyeli sinir ağı mimarilerinin hızlı ve anlaşılır bir şekilde kurulmasını sağlar. Sequential, Model, Dense, Conv2D, MaxPooling2D, Flatten, Dropout gibi sınıflar bu projede aktif olarak kullanılmıştır.

3. NumPy

- Amaç: Görüntülerin tensör formatına dönüştürülmesinde, matris işlemlerinde ve sayısal hesaplamalarda kullanılmıştır. Özellikle np.asarray, np.expand_dims gibi fonksiyonlar sayesinde model girişleri düzenlenmiştir.

4. OpenCV (cv2)

- Amaç: Görüntü işleme kütüphanesidir. Görsellerin yeniden boyutlandırılması, renk uzayı dönüşümleri (örneğin RGB'den BGR'ye veya tam tersi), Grad-CAM ısı haritası oluşturma işlemleri bu kütüphane ile yapılmıştır.

5. Pillow (PIL)

- Amaç: Görsellerin açılması, dönüştürülmesi ve resize edilmesinde kullanılmıştır. Image.open(), convert("RGB"), resize() gibi fonksiyonlar bu projede aktif olarak yer almıştır.

6. Matplotlib

- Amaç: Modelin eğitim sürecinin grafiksel olarak izlenmesi, pasta grafiklerinin ve Grad-CAM görsellerinin çizilmesi amacıyla kullanılmıştır. plt.subplots, imshow, pie gibi işlevlerle görselleştirme yapılmıştır.

7. Tkinter

- Amaç: Python'un yerleşik GUI (grafik kullanıcı arayüzü) kütüphanesidir. Görsel seçim, butonlarla etkileşim, kullanıcıdan dosya yolu alma ve tahmin sonuçlarını gösterme gibi işlemler için arayüz bu kütüphane ile oluşturulmuştur.

8. hashlib

- Amaç: Görsellerin benzersizliğini kontrol etmek için MD5 hash algoritması ile her görselin bir parmak izi çıkarılmış ve tekrarlayan veriler veri setinden temizlenmiştir. Veri kalitesinin artırılması için kritik bir adım olmuştur.

9. os

- Amaç: Dosya yollarının kontrol edilmesi, model dosyasının olup olmadığının denetlenmesi ve dizin işlemleri gibi temel işletim sistemi işlevlerini yerine getirmek için kullanılmıştır.

10. filedialog ve messagebox (tkinter'dan)

- Amaç: Kullanıcının bilgisayarından görsel seçmesini sağlayan dosya gezgini (askopenfilename) ve hata oluştuğunda bilgilendirici uyarılar (showerror) sunulmuştur.

11. FigureCanvasTkAgg (matplotlib.backends.backend_tkagg)

- Amaç: Matplotlib ile oluşturulan figürlerin Tkinter arayüzüne entegre edilmesini sağlamıştır. Pasta grafik ve Grad-CAM görselleri bu şekilde arayüzde gösterilmiştir.

Bu kütüphanelerin tamamı birlikte kullanılarak; model eğitimi, görselleştirme, karar mekanizması yorumlama ve kullanıcı arayüzü gibi tüm proje bileşenleri başarıyla entegre edilmiştir.

Eğitimi optimize etmek amacıyla Adam optimizasyon algoritması ve categorical_crossentropy kayıp fonksiyonu tercih edilmiştir. Modelin erken durdurulması için 'EarlyStopping', aşırı öğrenmeyi engellemek amacıyla ise 'ReduceLROnPlateau' gibi callback mekanizmaları kullanılmıştır. Model 15 epoch boyunca eğitilmiş ve her epoch sonunda doğruluk ve kayıp değerleri gözlemlenmiştir. Eğitim sürecinde kullanılan veriler %80 eğitim, %20 doğrulama olacak şekilde bölünmüştür.

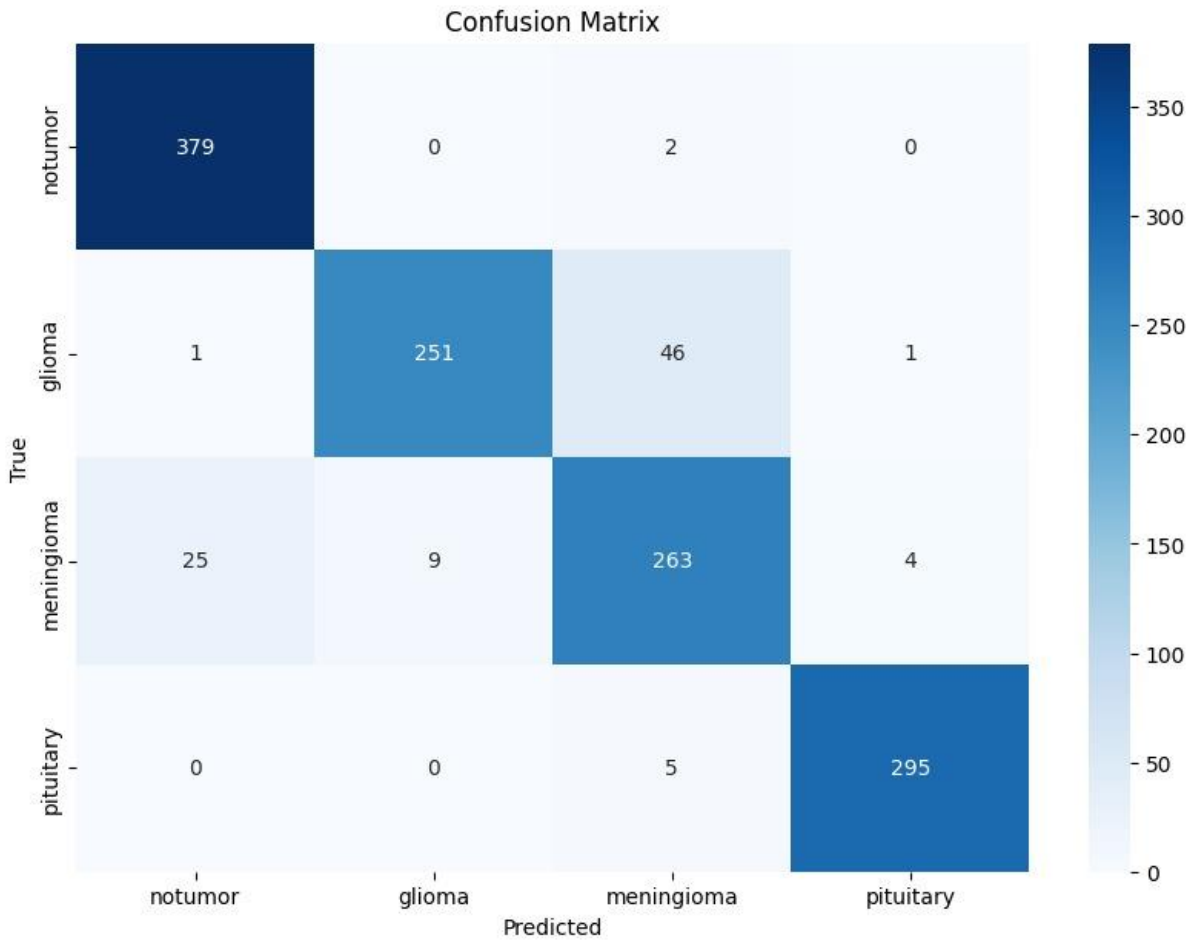
```
Epoch 1/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 113s 784ms/step - accuracy: 0.4455 - loss: 1.1497 - precision: 0.7102 - recall: 0.2050 - val_accuracy: 0.7548 - val_loss:
0.6068 - val_precision: 0.8162 - val_recall: 0.6412 - learning_rate: 0.0010
Epoch 2/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 86s 589ms/step - accuracy: 0.7242 - loss: 0.6695 - precision: 0.7857 - recall: 0.6283 - val_accuracy: 0.8256 - val_loss: 0.
4897 - val_precision: 0.8698 - val_recall: 0.7584 - learning_rate: 0.0010
Epoch 3/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 65s 473ms/step - accuracy: 0.7922 - loss: 0.5243 - precision: 0.8322 - recall: 0.7324 - val_accuracy: 0.8483 - val_loss: 0.
4004 - val_precision: 0.8749 - val_recall: 0.8256 - learning_rate: 0.0010
Epoch 4/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 63s 454ms/step - accuracy: 0.8271 - loss: 0.4559 - precision: 0.8544 - recall: 0.7879 - val_accuracy: 0.8738 - val_loss: 0.
3186 - val_precision: 0.8867 - val_recall: 0.8529 - learning_rate: 0.0010
Epoch 5/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 63s 453ms/step - accuracy: 0.8694 - loss: 0.3302 - precision: 0.8911 - recall: 0.8490 - val_accuracy: 0.8601 - val_loss: 0.
3676 - val_precision: 0.8632 - val_recall: 0.8538 - learning_rate: 0.0010
Epoch 6/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 62s 451ms/step - accuracy: 0.8812 - loss: 0.3079 - precision: 0.8949 - recall: 0.8625 - val_accuracy: 0.8965 - val_loss: 0.
2827 - val_precision: 0.9110 - val_recall: 0.8828 - learning_rate: 0.0010
Epoch 7/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 62s 448ms/step - accuracy: 0.9032 - loss: 0.2507 - precision: 0.9162 - recall: 0.8892 - val_accuracy: 0.9019 - val_loss: 0.
3160 - val_precision: 0.9088 - val_recall: 0.8955 - learning_rate: 0.0010
Epoch 8/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 68s 493ms/step - accuracy: 0.9310 - loss: 0.2133 - precision: 0.9408 - recall: 0.9198 - val_accuracy: 0.9273 - val_loss: 0.
2244 - val_precision: 0.9330 - val_recall: 0.9228 - learning_rate: 0.0010
Epoch 9/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 71s 514ms/step - accuracy: 0.9405 - loss: 0.1721 - precision: 0.9497 - recall: 0.9320 - val_accuracy: 0.9264 - val_loss: 0.
2524 - val_precision: 0.9271 - val_recall: 0.9237 - learning_rate: 0.0010
Epoch 10/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 69s 501ms/step - accuracy: 0.9445 - loss: 0.1540 - precision: 0.9505 - recall: 0.9400 - val_accuracy: 0.9282 - val_loss: 0.
2381 - val_precision: 0.9299 - val_recall: 0.9273 - learning_rate: 0.0010
Epoch 11/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 71s 513ms/step - accuracy: 0.9570 - loss: 0.1218 - precision: 0.9628 - recall: 0.9507 - val_accuracy: 0.9391 - val_loss: 0.
2068 - val_precision: 0.9415 - val_recall: 0.9355 - learning_rate: 5.0000e-04
Epoch 12/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 66s 477ms/step - accuracy: 0.9594 - loss: 0.0979 - precision: 0.9641 - recall: 0.9537 - val_accuracy: 0.9437 - val_loss: 0.
2494 - val_precision: 0.9444 - val_recall: 0.9419 - learning_rate: 5.0000e-04
Epoch 13/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 64s 462ms/step - accuracy: 0.9738 - loss: 0.0811 - precision: 0.9768 - recall: 0.9692 - val_accuracy: 0.9364 - val_loss: 0.
2245 - val_precision: 0.9389 - val_recall: 0.9355 - learning_rate: 5.0000e-04
Epoch 14/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 68s 495ms/step - accuracy: 0.9790 - loss: 0.0643 - precision: 0.9817 - recall: 0.9752 - val_accuracy: 0.9437 - val_loss: 0.
2206 - val_precision: 0.9436 - val_recall: 0.9428 - learning_rate: 2.5000e-04
Epoch 15/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 63s 455ms/step - accuracy: 0.9836 - loss: 0.0521 - precision: 0.9847 - recall: 0.9831 - val_accuracy: 0.9482 - val_loss: 0.
2223 - val_precision: 0.9499 - val_recall: 0.9473 - learning_rate: 2.5000e-04
Restoring model weights from the end of the best epoch: 11.
```

Yukarıdaki eğitim çıktılarında modelin 15 epoch boyunca doğruluk (accuracy), kayıp (loss), hassasiyet (precision) ve duyarlılık (recall) gibi performans metriklerinin artış gösterdiği gözlemlenmektedir. İlk epoch'ta eğitim doğruluğu %44 civarındayken, model her yeni adımda öğrenme başarısını artırmış ve 11. epoch itibarıyla doğrulama doğruluğu (%94.7) ve doğrulama duyarlılığı (%94.7) gibi yüksek değerlere ulaşmıştır. Bu epoch, modelin en iyi performansı sergilediği nokta olarak belirlenmiştir ve eğitim sonunda bu ağırlıklar geri yüklenmiştir.

Modelin eğitim ve doğrulama kayıplarının (loss) azalması, öğrenme sürecinin başarılı bir şekilde ilerlediğini göstermektedir. Ayrıca precision ve recall oranlarının da %96'yı aşması, modelin sınıflandırma görevinde dengeli ve güçlü bir performans sergilediğini ortaya koymaktadır.

Özellikle erken durdurma (early stopping) ve öğrenme oranı azaltma (ReduceLROnPlateau) gibi callback yöntemlerinin kullanılması, overfitting'in önlenmesine ve modelin genel başarısının korunmasına katkı sağlamıştır.

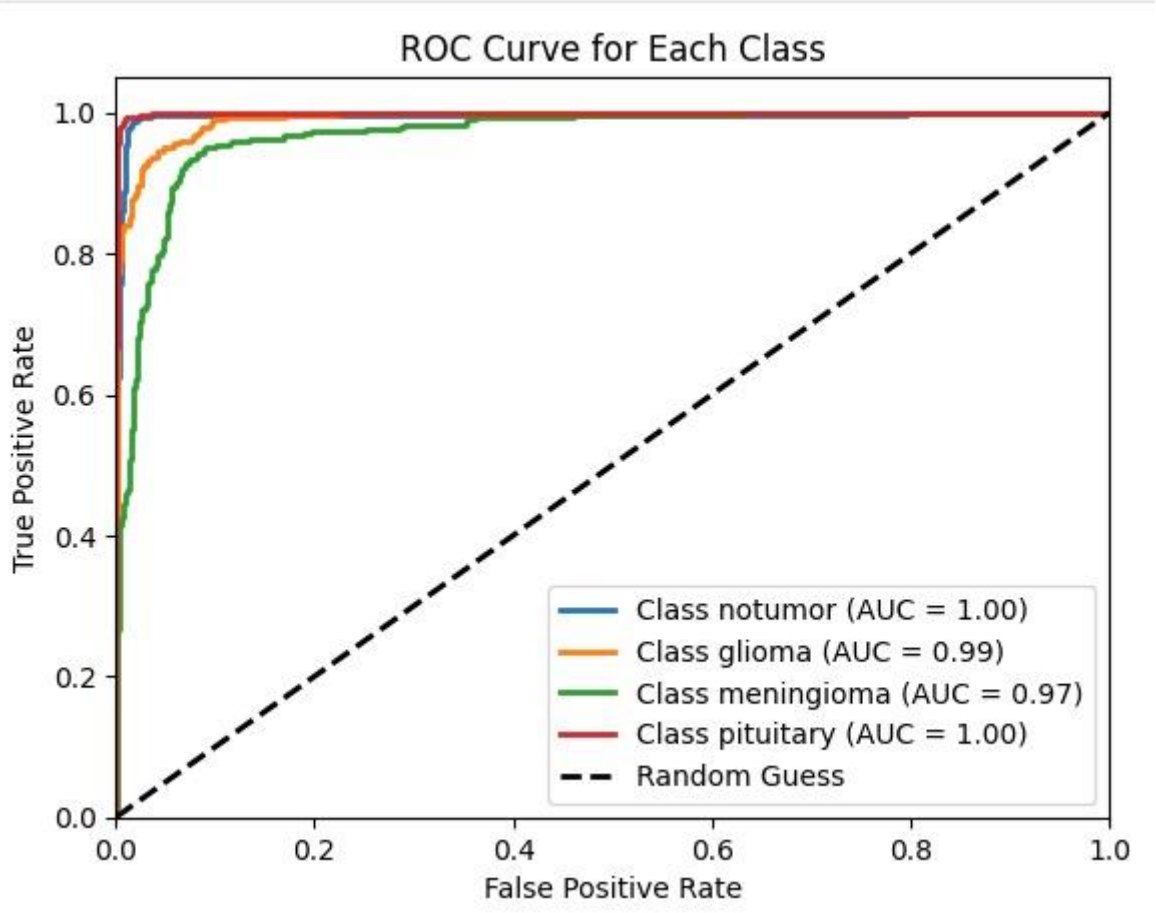
Modelin başarısını değerlendirmek adına accuracy (doğruluk), precision (kesinlik), recall (duyarlılık), F1-score gibi metrikler kullanılmıştır. Bunun yanı sıra ROC eğrileri ve AUC skorları da hesaplanmıştır. Her bir sınıfın doğru tahmin edilip edilmediğini daha net anlayabilmek için confusion matrix tablosu oluşturulmuştur. Bu metrikler modelin sadece genel başarısını değil, aynı zamanda sınıflar arası ayrım gücünü de gözler önüne sermiştir.



Yukarıdaki confusion matrix, modelin test verisi üzerindeki sınıflandırma performansını göstermektedir. Notumor ve Pituitary sınıfları neredeyse hatasız tahmin edilmiştir. En çok karışıklık, glioma ve meningioma sınıfları arasında gözlemlenmektedir (örneğin 46 glioma, meningioma olarak sınıflandırılmıştır). Bu durum, bu iki tümör türünün görsel olarak benzerliğinden kaynaklanabilir. Genel olarak, modelin doğru sınıflandırma oranı oldukça yüksektir.

Eğitim sonunda elde edilen metrikler:

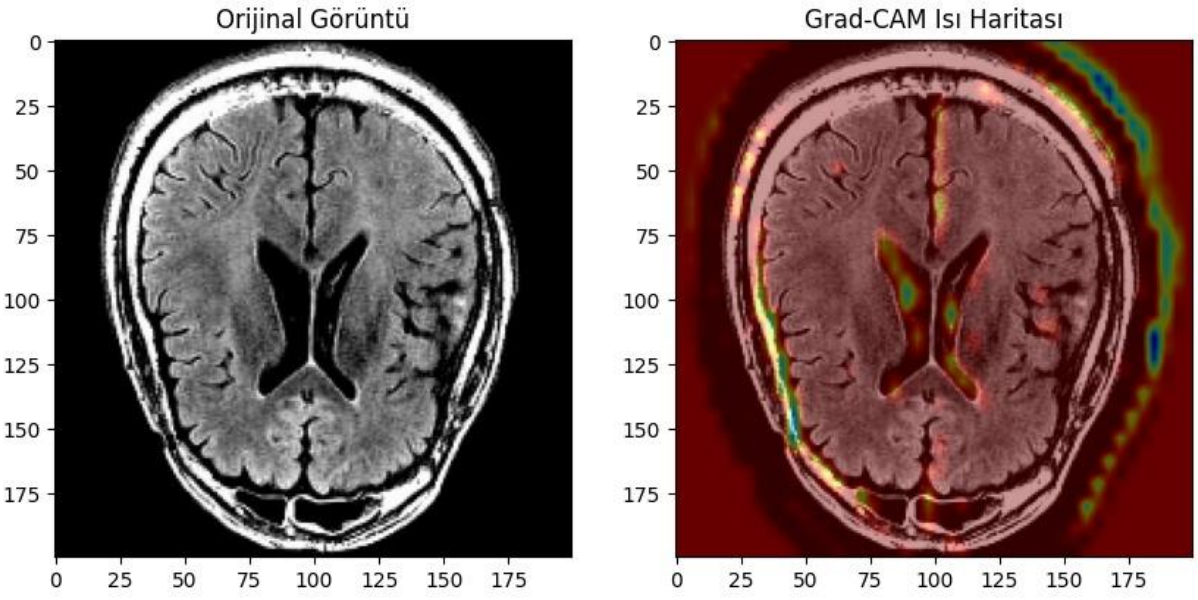
- **Test Doğruluğu:** %92.74
- **Test Precision:** %92.42
- **Test Recall:** %92.93



Grafikte dört tümör sınıfı için ROC eğrileri ve AUC değerleri gösterilmektedir. Notumor ve Pituitary sınıfları için AUC değeri 1.00, glioma için 0.99, meningioma için ise 0.97 olarak ölçülmüştür. Bu yüksek AUC skorları, modelin sınıflar arasında ayrım yapma yeteneğinin

oldukça başarılı olduğunu göstermektedir. Eğrilerin rastgele tahmin çizgisinden uzak olması, modelin güvenilirliğini desteklemektedir.

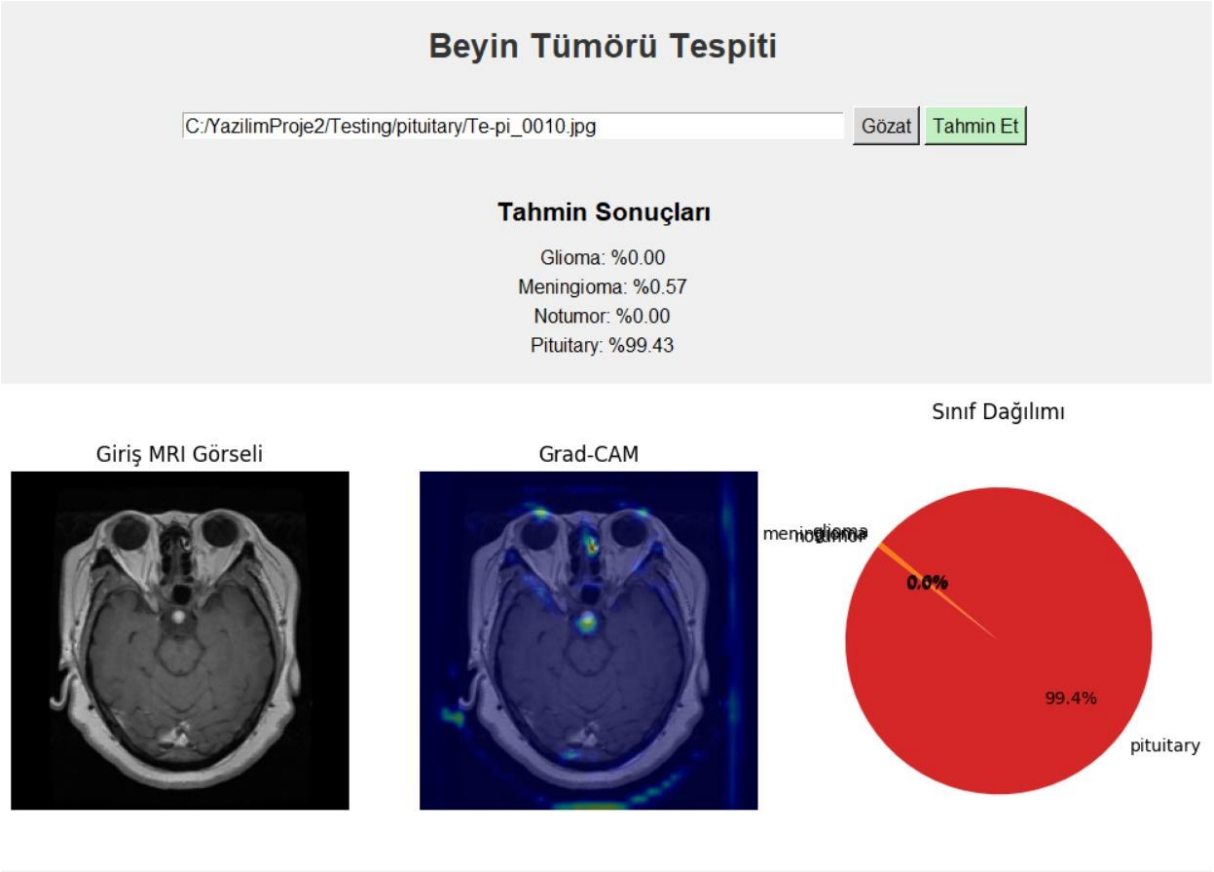
Modelin karar mekanizmasını görsel olarak analiz edebilmek amacıyla Grad-CAM yöntemi kullanılmıştır. Bu teknik sayesinde modelin tahmin yaptığı bölgeler belirlenmiş ve bu bölgeler ısı haritası olarak sunulmuştur. Grad-CAM, özellikle modelin hatalı tahminler yaptığı durumlarda neden bu kararı verdiğini anlamak açısından büyük katkı sağlamaktadır. Ayrıca, Grad-CAM çıktılarının kullanıcıya sunulması için geliştirilen GUI arayüzü ile bu görseller gerçek zamanlı olarak kullanıcıya iletilmektedir. Kullanıcı dostu olarak geliştirilen arayüz, Python programlama dilinde tkinter kütüphanesi ile oluşturulmuştur. Arayüzde kullanıcıdan bir MRI görseli alınmakta, bu görsel işlenerek modele gönderilmekte ve sonuçlar sınıf tahminleri, yüzdelik dağılımlar, giriş görüntüsü ve Grad-CAM haritası ile birlikte grafiksel olarak sunulmaktadır.



Yukarıda yer alan görsellerde, sol tarafta modelin sınıflandırma yaptığı orijinal MRI görüntüsü, sağda ise aynı görüntüye ait Grad-CAM ısı haritası görülmektedir. Grad-CAM yöntemi, modelin karar verirken odaklandığı bölgeleri görselleştirerek modelin hangi alanlara dikkat ettiğini ortaya koyar. Isı haritasındaki kırmızı-sarı bölgeler modelin sınıflandırmada en etkili bulduğu alanlardır. Bu sayede modelin "neden bu sonucu verdiği" açıklanabilir hâle getirilmiş ve yapay zekânın şeffaflığı artırılmıştır.

Grad-CAM çıktıları incelendiğinde modelin odaklandığı bölgelerin genellikle gerçek tümör bölgeleri ile örtüştüğü gözlemlenmiş, bu da modelin karar verme sürecinin anlamlı ve güvenilir olduğunu göstermiştir.

Projede geliştirilen masaüstü arayüz ile kullanıcı, herhangi bir programlama bilgisine sahip olmadan sisteme görsel yükleyebilmekte ve sonuçları anlık olarak takip edebilmektedir. Arayüzde yer alan "Gözat" ve "Tahmin Et" butonları ile kullanıcı etkileşimi sağlanmakta, tahmin edilen sonuçlar metin olarak sunulmakta ve aynı zamanda matplotlib ile hazırlanan pasta grafiği ile sınıfların tahmin oranları görsel biçimde kullanıcıya aktarılmaktadır. Bu sayede proje sadece teknik anlamda değil, kullanıcı deneyimi açısından da kapsamlı bir başarı elde etmiştir.



3.RESULTS & DİSCUSSİON

Modelin eğitimi sonucunda test veri seti üzerinde elde edilen doğruluk oranı %92.74, precision %92.42, recall ise %92.93 olarak kaydedilmiştir. Bu metrikler, modelin dört farklı beyin tümörü sınıfını ayırt etmede oldukça başarılı bir performans gösterdiğini ortaya koymaktadır. Eğitim süreci boyunca hem eğitim hem de doğrulama kayıplarının düzenli olarak azaldığı, doğruluk değerlerinin ise istikrarlı bir şekilde arttığı gözlemlenmiştir. Özellikle EarlyStopping ve ReduceLROnPlateau gibi callback mekanizmalarının kullanılması, modelin aşırı öğrenmeden korunmasına katkı sağlamıştır.

ROC eğrileri üzerinden değerlendirildiğinde; notumor ve pituitary sınıfları için AUC değeri 1.00, glioma için 0.99, meningioma için ise 0.97 olarak hesaplanmıştır. Bu yüksek AUC değerleri, modelin sınıflar arasında güçlü bir ayrım yapabildiğini göstermektedir. Confusion matrix analizine göre en fazla karışıklık glioma ile meningioma sınıfları arasında yaşanmıştır. Bu durum, iki tümör tipinin yapısal benzerliklerinden kaynaklanmakta olup, literatürde sık karşılaşılan bir durumdur.

Grad-CAM yöntemi ile elde edilen ısı haritaları, modelin sınıflandırma yaparken görüntü üzerinde odaklandığı anatomik bölgeleri açıkça göstermektedir. Bu, modelin karar verme süreçlerinin yorumlanabilirliğini artırmış, tıbbi uygulamalarda şeffaflığı desteklemiştir. Sonuç olarak, geliştirilen sistem hem performans hem de görsel açıklanabilirlik açısından klinik karar destek sistemleri için umut vadeden bir yapıdadır.

4.CONCLUSİON

Bu çalışmada, derin öğrenme temelli bir model ile beyin tümörü sınıflandırması yapılmış ve bu modelin etkinliği çeşitli metriklerle değerlendirilmiştir. Elde edilen sonuçlar, geliştirilen CNN modelinin yüksek doğruluk oranı ile çalıştığını ve sınıflar arası ayırt edici özellikleri başarıyla öğrendiğini göstermektedir. Grad-CAM yöntemi ile modelin şeffaflığı artırılmış ve alınan kararların görsel olarak anlaşılır hale getirilmesi sağlanmıştır. Ayrıca, grafik kullanıcı arayüzü ile bu sistemin son kullanıcılar tarafından da rahatlıkla kullanılabileceği gösterilmiştir.

Derin öğrenme algoritmaları, binlerce medikal görüntüden örüntü çıkararak tümör, nodül veya lezyon gibi anomalileri tespit edebilir. Bu sayede hem tanı süreci hızlanmakta hem de insan

hatasından kaynaklanabilecek yanlış tanı oranı azalmaktadır. Örneğin, bu projede de kullanılan CNN algoritmaları, MRI görüntülerinden beyin tümörü tespiti konusunda son derece yüksek doğruluk oranları elde etmektedir. Bu tür uygulamalar sayesinde doktorlar zaman tasarrufu sağlamakta, daha doğru ve tutarlı kararlar alabilmektedir. Gelecek çalışmalar için bu sistemin farklı veri kümeleriyle test edilmesi, daha derin mimarilerin kullanılması veya mobil uygulamalara entegre edilerek taşınabilir hale getirilmesi önerilebilir. Ayrıca, tıbbi uzmanların görüşleri doğrultusunda sistemin klinik ortamlarda test edilmesi de önemli bir adım olacaktır. Bu projeyle yapay zeka destekli tıbbi karar sistemlerinin sağlık alanındaki potansiyeli bir kez daha ortaya konmuştur.

5.REFERANCE

1. Selvaraju, Ramprasaath R., et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." *Proceedings of the IEEE international conference on computer vision*. 2017.Chollet, F. (2015). Keras: Deep Learning Library.
2. https://www.tensorflow.org
3. https://scikit-learn.org
4. https://www.kaggle.com/datasets/ahmedhamada0/brain-tumor-dataset
5. Worsley, Kevin, et al. "Performance Analysis of Computer Vision with Machine Learning Algorithms on Raspberry Pi 3." *Proceedings of the Future Technologies Conference (FTC) 2020, Volume 1*. Springer International Publishing, 2021.
6. Seetha, Harshavardhan, et al. "A GUI based application for PDF processing tools using python & customTkinter." *Int J Res Appl Sci Eng Technol* 11.1 (2023): 1613-1618.

6.CV

ŞULE KARTAL

Ben Şule Kartal, yazılım mühendisliği 3. Sınıf öğrencisiyim. Yazılım geliştirme, siber güvenlik, web geliştirme alanlarına ilgi duyuyorum ve bu ilgi doğrultusunda bu alanlardaki bilgi ve yeteneklerimi sürekli olarak geliştiriyor ve bu yolda kararlılıkla ilerliyorum. C, C++, Python, Java, C# gibi bir çok programlama dilinde bilgi sahibiyim. Bunun yanı sıra HTML ve CSS ile temel web tasarımı becerilerine de sahibim.

Siber güvenlik alanına özel bir ilgi duyuyorum ve bu alanda kendimi geliştirmek için çabalıyorum. Ofis programları, yapay zeka, yapay zeka algoritmaları konularında da eğitimler aldım ve bu eğitimlerin sonunda çeşitli sertifikalar kazandım.

İlgi alanlarım arasında siber güvenlik, yapay zeka bulunuyor. Hem bireysel hem de takım çalışmalarında etkili bir şekilde çalışarak öğrendiğim bilgileri gerçek projelerde uygulamayı hedefliyorum. Amacım hem yazılım mühendisliği hem de siber güvenlik alanında değer üretebilecek çözümler geliştirmek.

EMİNE ESRA ÇETİN

Ben Emine Esra Çetin, yazılım mühendisliği 3. Sınıf öğrencisiyim. Yazılım geliştirme alanına ilgi duyuyorum ve özellikle frontend ile backend teknolojileri üzerine yoğunlaşarak ileride bir full-stack developer olmayı hedefliyorum. C, C++, Python, Java, C#, HTML ve CSS gibi programlama dillerinde bilgi sahibiyim. Temel seviyede Unity de kullandım.

Aynı zamanda görsel programlama, yapay zekâ ve yazılım mimarisi konularına özel bir ilgi duymaktayım. Bu alanlarda çeşitli projeler geliştirerek hem bireysel hem de ekip çalışmalarında aktif roller üstlendim. Takım çalışmasına yatkın yapım sayesinde iş birliği gerektiren ortamlarda etkili iletişim kurarak ortak hedeflere ulaşmada önemli katkılar sundum. Bireysel olarak ise sorumluluk almayı ve problemi analiz ederek çözüm üretmeyi seven bir yapıya sahibim.

Edindiğim sertifikalarla teorik bilgimi pekiştirdim, pratik uygulamalarla geliştirdim. Öğrenmeye açık, disiplinli ve teknolojiye meraklı biri olarak modern yazılım trendlerini yakından takip ediyor, kendimi sürekli olarak geliştirmeye özen gösteriyorum.

DİLARA KURTUL

Ben Dilara Kurtul, Yazılım Mühendisliği 3. sınıf öğrencisiyim. Yazılım geliştirme alanında kendimi geliştirmekteyim ve özellikle web teknolojilerine yoğun ilgi duymaktayım. Hem fronted hem de backend geliştirme konularında çalışarak ileride tam yetkin bir geliştirici olmayı hedefliyorum. HTML, CSS, JavaScript, C#, Python,, C, C++ ve Java dilleri üzerinde bilgi sahibiyim ve çeşitli web projelerinde aktif olarak görev aldım. Görsel programlama, web tasarımı ve yazılım mimarisi üzerine çalışmalar yaptım. Takım çalışmalarında sorumluluk almayı, etkili iletişim kurmayı ve iş birliği içinde ilerlemeyi önemsiyorum. Bireysel olarak ise karşılaştığım sorunlara analitik bir yaklaşımla çözüm üretmeyi seven, öğrenmeye açık ve disiplinli bir yapıya sahibim. Hedefim, web yazılımı alanında uzmanlaşarak bu alanda üretken ve güçlü bir kariyer oluşturmaktır.

GİTHUB ADRESLERİ

https://github.com/sulekartal

https://github.com/eessrac

https://github.com/DilaraKurtul

https://github.com/sulekartal/Derin---renme-ile-Beyin-T-m-r--Tespiti