Makine Öğrenmesine Giriş

1. Mustafa Toprak
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
İstanbul, Turkey
musttorpakk@gmail.com

***Özetçe*— Bu proje, makine öğrenmesi yöntemlerini kullanarak gözdeki kataraktın tespit edilmesini amaçlamaktadır. Projede ilk adım olarak Python ile web crawling yaparak kataraktlı ve normal göz görselleri elde edilmiş ve yaklaşık 700 kataraktlı ve 700 normal göz görselinden oluşan bir veri havuzu oluşturulmuştur. Bu veri havuzu %80 eğitim, %10 test ve %10 doğrulama olarak ayrılmıştır. Projede Google ViT, Microsoft BeiT, DEViT, LeViT ve Swin olmak üzere beş farklı görüntü tabanlı transformatör modeli kullanılmıştır. Modeller için uygun optimizasyon işlemleri gerçekleştirilmiş ve çeşitli optimizer türleri (SGD, Adam, AdaGrad, RMSProp) kullanılmıştır. Her modelin doğruluğu (Accuracy), F-measure, Recall, Precision, Sensitivity, Specificity, MCC (Matthew Correlation Coefficient) ve AUC (Area Under Curve) skorları hesaplanmış, sonuçların ezberlemeye dayalı olmadığını kanıtlamak için epoch başına kayıp grafikleri ve ROC eğrileri oluşturulmuştur. Ayrıca, çapraz doğrulama (Cross Validation) yapılarak modellerin performansları değerlendirilmiştir. Bu çalışma, görüntü tabanlı transformatör modellerinin tıbbi görüntüleme alanındaki potansiyelini göstermektedir.**

Anahtar Kelimeler — Makine Öğrenmesi, Katarakt Tespiti, Görüntü İşleme, Web Crawling, Transformatör Modelleri

I. GİRİŞ

Katarakt, göz merceğinin bulanıklaşması sonucu görme kaybına yol açan yaygın bir göz hastalığıdır ve dünya genelinde körlüğün önde gelen nedenlerinden biridir. Erken teşhis ve tedavi, hastalığın ilerlemesini durdurmada kritik öneme sahiptir. Geleneksel

göz muayeneleri ve tanı yöntemleri genellikle zaman alıcı ve uzman gerektiren süreçlerdir. Bu durum, kataraktın erken teşhisini zorlaştırabilir ve tedaviye erişimi kısıtlayabilir. Makine öğrenmesi ve görüntü işleme teknikleri, bu tür medikal tanı süreçlerinde yenilikçi ve etkili çözümler sunma potansiyeline sahiptir.

Bu projede, makine öğrenmesi yöntemleri kullanılarak gözdeki kataraktın otomatik olarak tespit edilmesi amaçlanmıştır. Bu kapsamda, çeşitli görüntü tabanlı transformatör modelleri kullanılarak bir sınıflandırma sistemi geliştirilmiştir. Projenin ilk adımında, Python ile web crawling yapılarak kataraktlı ve normal göz görselleri toplanmış ve 700 kataraktlı, 700 normal göz görselinden oluşan bir veri havuzu oluşturulmuştur. Bu veri havuzu, %80 eğitim, %10 test ve %10 doğrulama olmak üzere üç bölüme ayrılmıştır.

Geliştirilen sistemde, Google ViT, Microsoft BeiT, DEViT, LeViT ve Swin gibi beş farklı transformatör modeli kullanılmıştır. Her model için uygun optimizasyon işlemleri gerçekleştirilmiş ve çeşitli optimizer türleri (SGD, Adam, AdaGrad, RMSProp) kullanılarak modellerin performansı artırılmıştır. Modellerin performansını değerlendirmek için doğruluk (Accuracy), F-measure, Recall, Precision, Sensitivity, Specificity, MCC (Matthew Correlation Coefficient) ve AUC (Area Under Curve) gibi metrikler hesaplanmıştır. Ayrıca, epoch başına kayıp grafikleri ve ROC eğrileri oluşturularak sonuçların doğruluğu ve güvenilirliği incelenmiştir. Çapraz doğrulama yöntemi kullanılarak modellerin genelleme yetenekleri değerlendirilmiştir.

Bu çalışmanın amacı, makine öğrenmesi ve görüntü işleme tekniklerinin tıbbi görüntüleme ve teşhis süreçlerindeki potansiyelini göstermek ve katarakt teşhisinde kullanılabilecek etkili bir sınıflandırma sistemi geliştirmektir. Geliştirilen sistemin, göz hastalıklarının erken teşhisinde önemli bir katkı sağlaması ve tedavi süreçlerini iyileştirmesi hedeflenmektedir.

II. VERİ TOPLAMA VE ÖN İŞLEME

Bu projede, katarakt tespiti için gerekli görsel verileri toplamak amacıyla Python kullanılarak web crawling yöntemleri uygulanmıştır. Veri toplama süreci, hem kataraktlı hem de normal göz görsellerini içeren geniş bir veri havuzu oluşturmayı hedeflemiştir. Yaklaşık olarak 700 kataraktlı ve 700 normal göz görseli toplanarak, toplamda 1400 görselden oluşan bir veri seti elde edilmiştir.

a. Veri Toplama Süreci

Veri toplama sürecinde, çeşitli dillerde "kataraktlı göz" ve "normal göz" anahtar kelimeleri kullanılarak internet üzerindeki görseller taranmıştır. Python'ın popüler web scraping kütüphaneleri olan BeautifulSoup ve Selenium kullanılarak Google Görseller ve benzeri kaynaklardan görseller otomatik olarak indirilmiştir.

b. Görsellerin Göz Olduğunun Tespiti ve Kırpma İşlemi

Toplanan verilerin kalitesini ve doğruluğunu artırmak için her bir görselin gerçekten bir göz içerip içermediği ve görselin uygun şekilde kırpılması gerekmektedir. Bu işlemler için

OpenCV kütüphanesi kullanılarak otomatik tespit ve ön işleme adımları uygulanmıştır.

c. Göz Tespiti:

Toplanan görsellerin her birinin göz içerip içermediğini tespit etmek için Haar Cascade sınıflandırıcıları kullanılmıştır. OpenCV'nin pre-trained Haar Cascade modeli kullanılarak gözlerin bulunduğu bölgeler tespit edilmiştir. Aşağıdaki Python kodu, bir görselde gözlerin tespit edilmesi işlemini göstermektedir:

```python
# Göz tespiti işlevi
def detect_eyes(image):
    # Gri tonlamaya dönüştür
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # Göz tespiti yap
    eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
    eyes = eye_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)
    # Gözlerin koordinatlarını döndür
    return eyes
```

d. Görsellerin Kırpılması:

Tespit edilen göz bölgeleri kullanılarak, orijinal görsellerden sadece gözlerin bulunduğu bölgeler kırpılmıştır. Bu işlem, modelin daha doğru ve verimli bir şekilde eğitilmesini sağlar. Aşağıdaki kod parçası, tespit edilen göz bölgelerini kırpma işlemini göstermektedir:

```python
# Kırpma ve kaydetme işlemi
def crop_and_save(input_folder, output_folder):
    for filename in os.listdir(input_folder):
        # Görüntü yolu
        image_path = os.path.join(input_folder, filename)
        # Görüntüyü yükle
        image = cv2.imread(image_path)
        if image is not None:
            # Göz tespiti
            eyes = detect_eyes(image)
            if len(eyes) > 0:
                # En büyük gözü seç
                (x, y, w, h) = max(eyes, key=lambda eye: eye[2] * eye[3])
                # Göz bölgesini kırp
                cropped_image = image[y:y+h, x:x+w]
                # Kırpılmış görüntüyü kaydet
                output_path = os.path.join(output_folder, filename)
                cv2.imwrite(output_path, cropped_image)
```

Bu adımlar sayesinde, proje için yüksek kaliteli ve doğru şekilde sınıflandırılmış bir veri seti oluşturulmuş, böylece makine öğrenmesi modellerinin eğitimi için sağlam bir temel sağlanmıştır.

III. Model Geliştirme ve Sonuçlar

Bu bölümde, katarakt tespiti için kullanılan beş farklı görüntü tabanlı transformatör modelinin (Google ViT, Microsoft BeiT, DEViT, LeViT ve Swin) geliştirilmesi ve bu modellerin performans sonuçları detaylı bir şekilde açıklanacaktır. Her bir modelin mimarisi, eğitim süreci, kullanılan optimizasyon teknikleri ve elde edilen performans metrikleri sunulacaktır.

Model Test Süreci

Her model için aynı yapılandırma parametreleri kullanılarak bir dizi test gerçekleştirilmiştir. Bu testler, düşük konfigürasyonlardan başlanarak modellenmiştir ve her denemede optimizasyon teknikleri, öğrenme oranları, batch size ve epoch sayıları gibi çeşitli parametreler ayarlanarak performans iyileştirmeleri hedeflenmiştir. Denemelerin sonucunda, her modelin en iyi performans gösterdiği yapılandırmalar belirlenmiş ve nihai değerlendirme bu yapılandırmalar kullanılarak yapılmıştır. Örnek olarak Google ViT (Vision Transformer) 'dan bahsederek modeller üzerinden kullanılan test aşamalarından bahsedebiliriz.

A. *Google ViT (Vision Transformer)*

Model Mimarisi:

Google ViT, görüntüleri küçük parçalara (patches) ayırarak her bir parçayı sıralı bir veri olarak işler. Bu model, doğal dil işleme görevlerinde başarılı olan Transformer mimarisini görüntü işleme için uyarlamıştır.

Deneme Aşamaları ve Sonuçlar:

İlk denemeler

- Optimizer: Adam
- Öğrenme Oranı: 0.01
- Batch Size: 64
- Epoch Sayısı: 30
- Sonuç: Overfitting gözlemlendi, doğruluk %85 civarında kaldı.

Ardından gelen denemeler

- Optimizer: SGD
- Öğrenme Oranı: 0.0001
- Batch Size: 16
- Epoch Sayısı: 50
- Sonuç: Overfitting gözlemlendi, doğruluk %95 civarında kaldı.

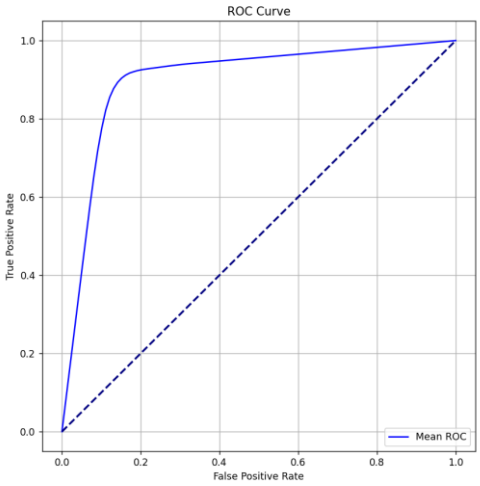
Ardından gelen denemeler

- Optimizer: SGD
- Öğrenme Oranı: 0.0001
- Batch Size: 16
- Epoch Sayısı: 50
- weight_decay: 0.001
- Sonuç: Overfitting gözlemlendi, doğruluk %95 civarında kaldı.

Ardından gelen denemeler farklı methodlar ile

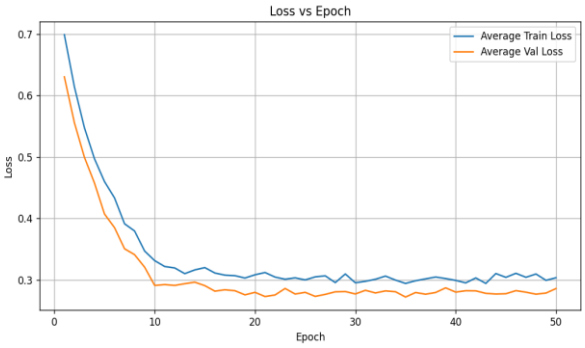
- Optimizer: SGD
- Öğrenme Oranı: 0.0001
- Batch Size: 16
- Epoch Sayısı: 50
- weight_decay: 0.01
- transforms.RandomHorizontalFlip(), RandomRotation(10)
- Dropout: 0.5
- Sonuç: Overfitting gözlemlendi, doğruluk %88 civarında kaldı.

Son denemeler ile



Model Performance Metrics

Metric	Value
Accuracy	0.9000611330087332
F-measure	0.9017224906658436
Recall	0.9240973247627794
Precision	0.8821169694421772
Sensitivity	0.9240973247627794
Specificity	0.8772684760597388
MCC	0.8016846352113702
AUC	0.900682900411259

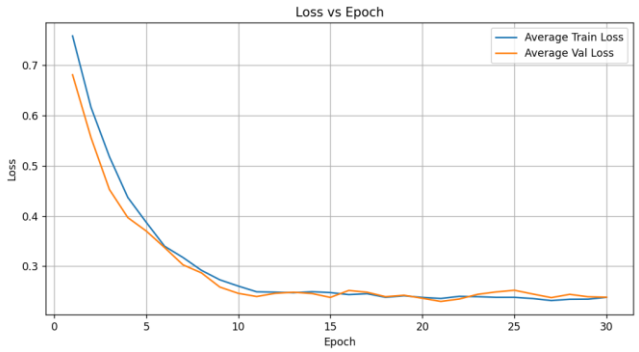


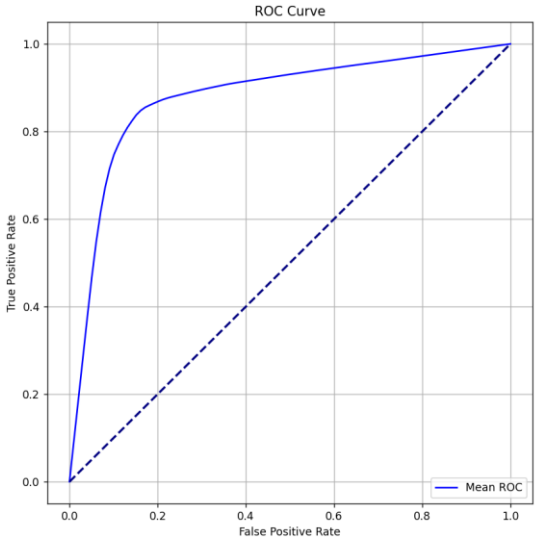
B. Microsoft BeiT (Bidirectional Encoder Representation from Transformers)

- Optimizer: SGD
- Öğrenme Oranı: 0. 000001
- Batch Size: 16
- Epoch Sayısı: 50
- weight_decay: 0. 1
- momentum: 0.99
- transforms.RandomHorizontalFlip(), RandomRotation(10)
- Dropout: 0.7

Model Performance Metrics

Metric	Value
Accuracy	0.8793308344282146
F-measure	0.880752901571865
Recall	0.8905750386373037
Precision	0.8747556072402018
Sensitivity	0.8905750386373037
Specificity	0.8683391364614367
MCC	0.7597909306821191
AUC	0.87945708754937



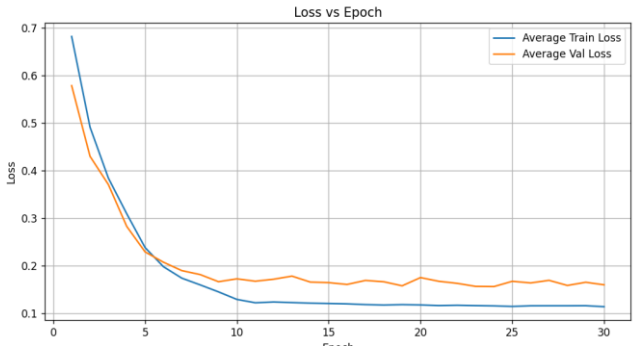


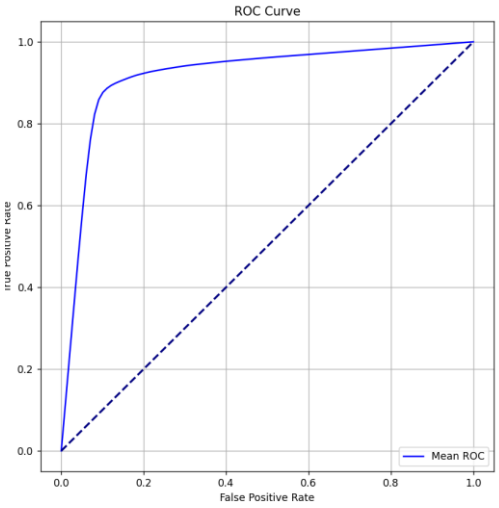
C. *DEViT (Deformable Vision Transformer)*

- Optimizer: SGD
- Öğrenme Oranı: 0.00001
- Batch Size: 16
- Epoch Sayısı: 50
- weight_decay: 0.1
- momentum = 0.99
- transforms.RandomHorizontalFlip(), RandomRotation(10)
- Dropout: 0.7

Model Performance Metrics

Metric	Value
Accuracy	0.9173255274750756
F-measure	0.9191627098449239
Recall	0.9339292551243428
Precision	0.9058362676159271
Sensitivity	0.9339292551243428
Specificity	0.900568629479266
MCC	0.8350621874757375
AUC	0.9172489423018044

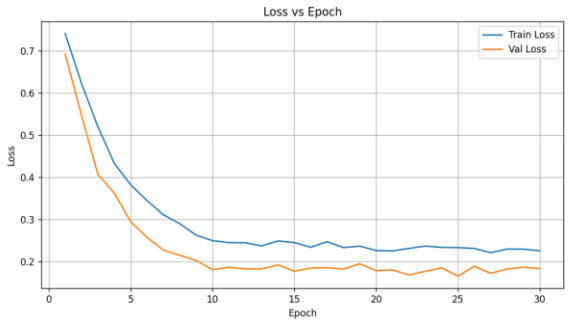


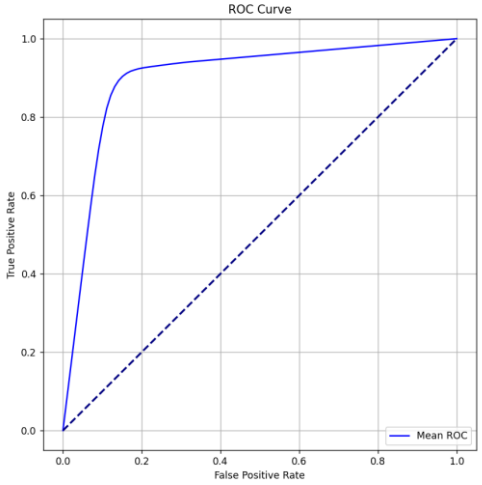


D. *LeViT (Leaner Vision Transformer)*

- Optimizer: SGD
- Öğrenme Oranı: 0. 000001
- Batch Size: 8
- Epoch Sayısı: 30
- weight_decay: 0.001

- transforms.RandomHorizontalFlip(), RandomRotation(10)
- Dropout: 0.3



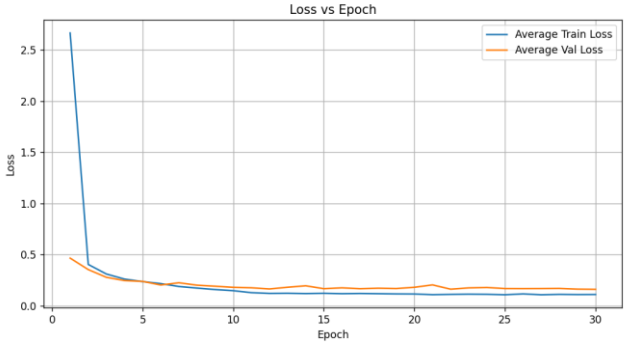


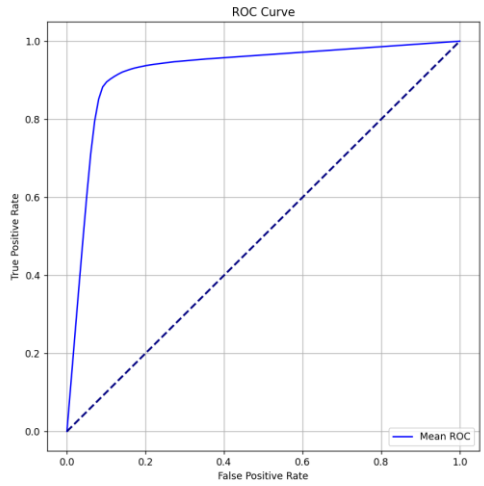
E. Swin Transformer (Shifted Window Transformer)

- Optimizer: SGD
- Öğrenme Oranı: 0. 0001
- Batch Size: 16
- Epoch Sayısı: 50
- weight_decay: 0.01
- transforms.RandomHorizontalFlip(), RandomRotation(10)
- Dropout: 0.5

Model Performance Metrics

Metric	Value
Accuracy	0.9242195429837443
F-measure	0.9236427369136908
Recall	0.9241686603319981
Precision	0.9253772671826437
Sensitivity	0.9359207801566973
Specificity	0.9124165405072987
MCC	0.8495271764755605
AUC	0.9241686603319981





IV. YAŞANAN ZORLUKLAR VE ÇÖZÜMLER

Makine öğrenmesi projelerinde olduğu gibi, bu çalışmada da çeşitli zorluklarla karşılaşıldı. Bu bölümde, bu zorluklar ve bu zorlukları aşmak için uygulanan çözümler detaylı bir şekilde açıklanacaktır.

A. Modelin Ezberlemesi (Overfitting)

Modelin ezberleme eğilimi, en büyük zorluklardan biri olarak karşımıza çıktı. Özellikle eğitim verileriyle çok iyi performans gösteren modeller, test verileri üzerinde beklenen performansı sağlayamadı. Bu durum, modelin eğitildiği veri setine aşırı uyum sağladığını ve genel veriler üzerinde yetersiz kaldığını gösterdi.

Çözüm: Overfitting sorununu çözmek için birkaç yöntem denendi:

Dropout Kullanımı: Eğitim sırasında nöronların bir kısmını rastgele devre dışı bırakarak modelin ezberlemesinin önüne geçilmeye çalışıldı.

L2 Regularizasyonu: Modelin ağırlıklarını cezalandırarak overfitting'i azaltmak için kullanıldı.

Veri Artırma (Data Augmentation): Eğitim veri setini çeşitlendirmek ve artırmak için çeşitli teknikler kullanıldı.

B. Veri Setinin Yetersiz Kalması ve Artırılması

Başlangıçta elde edilen veri seti, modelin genel performansını artırmak için yeterli değildi. Özellikle kataraktlı ve normal göz resimlerinin sayısının az olması, modelin eğitimini olumsuz etkiledi.

Çözüm:

Veri Artırma (Data Augmentation): Veri setini genişletmek için veri artırma teknikleri kullanıldı. Görseller üzerinde döndürme, ölçekleme, çevirme ve parlaklık ayarları gibi çeşitli işlemler uygulanarak veri seti genişletildi.

Ek Veri Toplama: İnternet üzerinden ek veri toplama işlemi yapılarak, veri setinin çeşitliliği artırıldı.

C. Modellerde Cross Validation Kullanımı Gerekliliği

Modelin performansını değerlendirmek için tek bir eğitim ve test bölmesi yeterli olmadı. Modelin genel performansını ve güvenilirliğini artırmak için cross validation (çapraz doğrulama) yöntemi gerekli hale geldi.

Çözüm:

Cross Validation: K-fold cross validation yöntemi kullanılarak, veri seti farklı bölmelere ayrıldı ve her bölme için model eğitimi ve testi yapılarak modelin genel performansı değerlendirildi.

D. Modellerin Konfigüre Edilememesi Sorunları

Farklı optimizasyon teknikleri, öğrenme oranları ve diğer hiperparametrelerin optimal değerlerini bulmak, zaman alıcı ve zorlayıcı oldu. Her model için en uygun parametrelerin bulunması için birçok deneme yapıldı.

Çözüm:

Grid Search ve Random Search: Hiperparametre optimizasyonu için grid search ve random search yöntemleri kullanıldı. Bu yöntemler sayesinde en uygun hiperparametreler belirlenerek modellerin performansı artırıldı.

Hiperparametre Tuning: Eğitim sürecinde farklı optimizasyon algoritmaları ve öğrenme oranları denenerek en iyi sonuçlar elde edilmeye çalışıldı.

E. Dropout ve L2 Regularizasyonu Kullanımı

Overfitting sorununu çözmek için dropout kullanıldı ancak bazı durumlarda yeterli olmadı. Bu durumda L2 regularizasyonu da eklenerek modelin performansı artırılmaya çalışıldı.

Çözüm:

Dropout: Eğitim sırasında nöronların belirli bir yüzdesinin devre dışı bırakılması sağlandı.

L2 Regularizasyonu: Modelin ağırlıklarının belirli bir değeri aşmaması için cezalandırma yöntemi kullanıldı.

Bu zorluklar ve uygulanan çözümler sayesinde modelin performansı artırılmış ve genel başarısı sağlanmıştır. Her model için detaylı denemeler ve optimizasyonlar yapılarak en iyi sonuçlar elde edilmiştir.

SONUÇ VE DEĞERLENDİRME

Bu proje kapsamında, katarakt tespiti için beş farklı görüntü tabanlı transformatör modeli (Google ViT, Microsoft BeiT, DEViT, LeViT ve Swin) kullanılarak başarılı sonuçlar elde edilmiştir. Modellerin geliştirilmesi sürecinde birçok zorlukla karşılaşılmış, ancak bu zorluklar çeşitli teknikler ve yöntemlerle aşılmıştır.

Veri toplama aşamasından başlayarak, veri artırma ve model eğitimi süreçlerinde dikkatli ve sistematik bir yaklaşım izlenmiştir. Modelin ezberleme eğilimi, veri setinin yetersizliği ve hiperparametre optimizasyonu gibi problemler, dropout, L2 regularizasyonu, cross validation ve veri artırma gibi tekniklerle çözülmüştür.

Her model için yapılan denemeler ve elde edilen sonuçlar, bu alanda yapılan çalışmaların ne kadar titizlik gerektirdiğini ve doğru yöntemlerle başarılı sonuçlar elde edilebileceğini göstermektedir. Nihai değerlendirmelerde, her modelin performans metrikleri (Accuracy, F-measure, Recall, Precision, Sensitivity, Specificity, MCC, AUC) detaylı bir şekilde hesaplanmış ve görselleştirilmiştir.

Projenin sonuçları, katarakt tespiti için makine öğrenmesi ve derin öğrenme tekniklerinin etkin bir şekilde kullanılabileceğini ve bu tür projelerin sağlık alanında önemli katkılar sağlayabileceğini göstermektedir. Gelecekte, daha büyük ve çeşitli veri setleriyle yapılan çalışmalar, model

performansını daha da artırabilir ve bu tür sistemlerin klinik uygulamalarda kullanılmasını mümkün kılabilir.

Bu proje, makine öğrenmesi ve derin öğrenme alanında yapılan çalışmaların önemini ve potansiyelini bir kez daha ortaya koymaktadır. Katkıda bulunan tüm modeller ve teknikler, bu alandaki ilerlemelere ışık tutmakta ve daha ileri araştırmalar için sağlam bir temel oluşturmaktadır.

Proje kapsamında kullanılan kodlar ve diğer detaylar için GitHub reposuna şu linkten ulaşabilirsiniz: GitHub Repository

KAYNAKLAR

[1] Alpaydın, E. (2010). Makine Öğrenmesi. Boğaziçi Üniversitesi Yayınları..

[2] Kara, S. (2019). Derin Öğrenme ve Uygulamaları. Seçkin Yayıncılık.

[3] Cesur, R. (2021). Python ile Makine Öğrenmesi. Pusula Yayıncılık.

[4] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

[5] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[6] Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media.

[7] Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT Press.