**T.C.**
**ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ**
**SAĞLIK BİLİMLERİ ENSTİTÜSÜ**
**BİYOİSTATİSTİK ANABİLİM DALI**

# OKÜLER HASTALIKLARIN SINIFLANDIRILMASINDA DERİN KONVOLÜSYONEL SİNİR AĞI MODELİ

**Büşra EMİR**

**DOKTORA TEZİ**

**DANIŞMAN**
**Prof. Dr. Ertuğrul ÇOLAK**

**Eskişehir**
**2021**

**T.C.**
**ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ**
**SAĞLIK BİLİMLERİ ENSTİTÜSÜ**
**BİYOİSTATİSTİK ANABİLİM DALI**

# OKÜLER HASTALIKLARIN SINIFLANDIRILMASINDA DERİN KONVOLÜSYONEL SİNİR AĞI MODELİ

**Büşra EMİR**

**DOKTORA TEZİ**

**DANIŞMAN**
**Prof. Dr. Ertuğrul ÇOLAK**

**Eskişehir**
**2021**

# TEŞEKKÜR

Doktora eğitimim boyunca kıymetli bilgi, birikim ve tecrübeleri ile bana yol gösterici ve destek olan değerli danışman hocam Prof. Dr. Ertuğrul ÇOLAK'a teşekkürlerimi sunarım.

Tez çalışmasında desteğini esirgemeyen, kıymetli öneri ve eleştirileriyle beni yönlendiren ve tez izleme komitesinde yer alan değerli hocalarım Prof. Dr. Setenay ÖNER ve Doç. Dr. Ünal ERKORKMAZ'a teşekkür ederim.

Tüm hayatım boyunca bana her zaman inanan, güvenen, destek olan, bu günlere gelmemi sağlayan, emeklerini asla ödeyemeyeceğim ve minnettarlığımı kelimelerle dile getiremeyeceğim aileme sonsuz teşekkür ederim.

Büşra EMİR
27.04.2021

# ÖZET


## OKÜLER HASTALIKLARIN SINIFLANDIRILMASINDA DERİN KONVOLÜSYONEL SİNİR AĞI MODELİ

**Amaç:** Oftalmolojide, insan gözündeki anatomik yapıları ve anomalilikleri yakalamak için bir retina görüntüleme yöntemi olan fundus görüntüleme ve erken fundus taraması, oftalmolojik hastalıkların neden olduğu körlüğü önlemenin etkili ve ekonomik bir yoludur. Klinik olarak, tıbbi kaynakların yetersizliği nedeniyle manuel teşhis zaman alıcıdır. Hastalığa ait tanı, teşhis ve tedavi durumunu geciktirebilir. Günümüzde hızlı ve otomatik hastalık tespiti, oftalmologların iş yükünü azaltmak için kritik ve acildir. Oftalmologlar, gözün ve çevresindeki yapıların doğrudan veya dolaylı olarak görselleştirilmesi yoluyla örüntü tanımaya dayalı olarak hastalıkları teşhis eder. Oftalmoloji alanının hastalık tespitinde fundus görüntülerine olan bu bağlılığı, derin öğrenme mimarilerinden yararlanmak için mükemmel bir zemin hazırlamıştır. Bu tez çalışmasının temel amacı, yapay zekâ odaklı oküler hastalıkların sınıflandırma probleminin çözümlenmesini sağlamaktır. Bu kapsamda, hastaların sağ ve sol gözlerinden alınan renkli fundus görüntüleri ve her bir görüntüye ait hastalık teşhis anahtar kelimeleri olan eğitim verisi kullanılarak oküler hastalık sınıflandırması için farklı derin konvolüsyonel sinir ağı modelleri oluşturmak, bu modellerin eğitimini gerçekleştirmek, model eğitimi sonrası, test fundus görüntü kümesi kullanılarak oluşturulan modellerin hastalık sınıflandırma performans ölçütleri olan kappa değeri, F1 skoru, Eğri altında kalan alan (Area Under Curve-AUC) ve bu üç ölçütün ortalaması olan Final skoru değerlerinin hesaplanması, modellerin sınıflandırma performansının bu ölçütlere göre değerlendirilmesi, en iyi skora sahip modelin oküler hastalıkların sınıflandırılmasında kullanılması ve literatüre önerilmesi amaçlanmıştır.

**Yöntem:** Konvolüsyonel sinir ağları (CNN), güçlü özellik öğrenme yeteneği ile fundus görüntüleri üzerinde dikkate değer bir başarı elde etmiştir ve derin öğrenmenin gelişmesiyle birlikte, oftalmoloji alanında hastalıklar üzerine yapılan araştırmalar hız kazanmıştır. Yapılan araştırmaların çoğu sadece tek bir hastalığa odaklanmıştır. Göz dibi taraması sırasında, oftalmologlar genellikle binoküler fundus görüntüsünde çoklu hastalık teşhisi verirler. Gerçek tıbbi senaryoyu karşılayabilmek için 2019 yılında ilk kez 8 farklı hastalık sınıfı içeren binoküler fundus görüntülerinden oluşan halka açık Oküler Hastalık Akıllı Tanıma (ODIR) veri seti yayınlanmıştır.

**Bulgular:** Bu tez çalışmasında, oftalmolojik hastalıklara ait görüntülerinin sınıflandırılması için literatürde State of the Art Modelleri olarak adlandırılan ve üç başarılı model olan Inceptionv3, VGG16 ve ResNet50 CNN mimarisi oluşturulmuştur. Ön-eğitimli ImageNet ağırlıkları kullanılarak, her bir model için öznitelik çıkarımı elde edildikten sonra modellerin hastalık sınıflandırma performansları değerlendirilmiştir. Uluslararası Oküler Hastalık Akıllı Tanıma Yarışması tarafından sağlanan veri seti üzerinde eğitilen ve test edilen modeller için sırası ile Final skoru değerleri VGG16 modeli için, 0.677, Inceptionv3 modeli için 0.669, ResNet50 modeli için 0.628 bulunmuştur.

**Sonuç:** Elde edilen yüksek doğruluk, AUC, F1 skoru, kappa değeri ve son üç ölçütün ortalaması olan en yüksek Final skor değerine sahip CNN modeli olan VGG16 modelinin fundus görüntülerinin sınıflandırılmasında kullanılabileceğini; Tıp Fakültelerinin Oftalmoloji alanında uzmanlara tanı aşamasında yardımcı bir destek rolü üstlenebileceği ve bu alanda gelecekte kullanılabilir sistemler tasarlanabileceğini göstermektedir.

**Anahtar Kelimeler:** Görüntü Sınıflandırma, Fundus görüntüleri, Derin öğrenme, Konvolüsyonel Sinir Ağları, Ön-eğitimli ağlar, Inceptionv3, VGG16, ResNet50.

## SUMMARY

## A DEEP CONVOLUTIONAL NEURAL NETWORK MODEL FOR CLASSIFICATION OF OCULAR DISEASES

**Aim:** Fundus imaging and early fundus scanning, a retinal imaging method to capture anatomical structures and anomalies in the human eye in ophthalmology, is an effective and economical way to prevent blindness caused by ophthalmologic diseases. Clinically, manual diagnosis is time consuming due to the scarcity of medical resources. Diagnosis, diagnosis and treatment of the disease may delay. Fast and automated disease detection today is critical and urgent to reduce the workload of ophthalmologists. Ophthalmologists diagnose diseases based on pattern recognition through direct or indirect visualization of the eye and surrounding structures. This commitment of the field of ophthalmology to fundus images in disease detection has laid the perfect groundwork for taking advantage of deep learning architectures. The main purpose of this thesis is to solve the classification problem of ocular diseases focused on artificial intelligence. In this context, creating different deep convolutional neural network models for ocular disease classification by using color fundus images taken from the right and left eyes of the patients and the education data, which are the disease diagnosis keywords of each image, to train these models, after the model training, the test fundus image set Calculation of the disease classification performance criteria such as kappa value, F1 score, Area Under Curve (AUC) and Final score values which are the average of these three criteria, evaluation of the classification performance of the models according to these criteria, It is aimed to be used in the classification of diseases and suggested to the literature.

**Method:** Convolutional neural networks (CNN) have achieved remarkable success on fundus images with their powerful feature learning ability, and with the development of deep learning, research on diseases in the field of ophthalmology has gained momentum. Most of the research has focused on only one disease. During fundus scanning, ophthalmologists usually diagnose multiple diseases on a binocular fundus image. In order to meet the real medical scenario, a public Ocular Disease Intelligent Recognition (ODIR) data set consisting of binocular fundus images containing eight different disease classes was published for the first time in 2019.

**Results:** In this thesis, three successful models Inceptionv3, VGG16 and ResNet50, which are called State of the Art Models in the literature, were created for the classification of images of ophthalmologic diseases. The disease classification performances of the models were evaluated after feature extraction was obtained for each model using pre-trained ImageNet weights. Final score values for the models trained and tested on the data set provided by the International Ocular Disease Intelligent Recognition Competition were 0.677 for the VGG16 model, 0.669 for the Inceptionv3 model, and 0.628 for the ResNet50 model, respectively.

**Conclusion:** The obtained high accuracy, AUC, F1 score, kappa value and the highest Final score value of the CNN model, which is the average of the last three criteria, can be used in classification of fundus images; It shows that Medical Faculties can play an auxiliary role to support specialists in the field of ophthalmology at the diagnosis stage and those systems can be designed in the future.

**Keywords:** Image Classification, Fundus images, Deep learning, Convolutional Neural Networks, Pre-trained networks, Inceptionv3, VGG16, ResNet50.

# İÇİNDEKİLER

# TABLO DİZİNİ

# ŞEKİL DİZİNİ

# ŞEKİL DİZİNİ (Devam Ediyor)

# ŞEKİL DİZİNİ (Devam Ediyor)

# SİMGE VE KISALTMALAR DİZİNİ

## Simgeler

| | | |
|---|---|---|
| $S$ | : | Softmax fonksiyonu |
| $\nabla$ | : | Gradyan |
| $\partial$ | : | Kısmi türev |
| $L$ | : | Kayıp (loss) fonksiyonu |
| $y$ | : | Gerçek (bilinen hedef) değer |
| $\hat{y}$ | : | Tahmini çıktı değeri |
| $\eta$ | : | Öğrenme katsayısı |
| $\nabla_\theta J(\theta_t)$ | : | $\theta$ model parametresine bağlı hata fonksiyonu gradyanı |
| $v_t$ | : | $t$ iterasyondaki momentum değeri |
| $\gamma$ | : | Bir önceki iterasyondaki momentum değeri katsayısı |
| $\varepsilon$ | : | Öğrenme katsayısının 0'a bölünmemesi için atanan sabit değer |
| $G_{t,ii}$ | : | $\theta_i$ parametresine göre, $t$. iterasyona kadar hesaplanmış gradyan değerlerinin kareleri toplamı, Her biri $i, i$ konumundaki köşegen elemanı |
| $g_{t,i}$ | : | $t$ anında, $\theta_i$ parametresine göre hesaplanmış, hata fonksiyonunun gradyan değeri |
| $g\ x\ g$ | : | Giriş matrisi boyutu |
| $a\ x\ a$ | : | Ağırlık matrisi boyutu |
| $n_{in}$ | : | Ağırlık matrisindeki girdilerin sayısı |
| $n_{out}$ | : | Ağırlık matrisindeki çıktıların sayısı |
| $\sigma$ | : | Standart sapma |
| $a^{[l]}$ | : | $l$. katmandan gelen artık değer |
| $w_i$ | : | $i$. girdi birimine ait ağırlık değeri |
| $x_i$ | : | i. girdi birimi |
| $\Delta x$ | : | $x$ eksenindeki değişim |
| $\Delta y$ | : | $y$ eksenindeki değişim |

## Kısaltmalar

| | | |
|---|---|---|
| OCT | : | Optik Koherens Tomografi |
| D | : | Diyabetik Retinopati |
| A | : | Yaşa bağlı Makula Dejenerasyonu |
| MIT | : | Massachusetts Teknoloji Enstitüsü |
| AI | : | Yapay Zeka |
| ILSVRC | : | ImageNet Büyük Ölçekli Görüntü Tanıma Yarışması |
| KVKK | : | Kişisel Verileri Koruma Kanunu |
| GDPR | : | Genel Veri Koruma Yönetmeliği |
| GPU | : | Grafik İşlem Birimi |
| TPU | : | Tensör İşlem Birimi |
| MR | : | Manyetik Rezonans |

| | | |
|---|---|---|
| ML | : | Makine Öğrenmesi |
| DL | : | Derin Öğrenme |
| CNN | : | Konvolüsyonel Sinir Ağları |
| LSDM | : | Uzun Vadeli Kısa Dönem Bellek Modeli |
| GSYİH | : | Gayri Safi Yurtiçi Hasıla |
| WHO | : | Dünya Sağlık Örgütü |
| N | : | Normal |
| G | : | Glokom |
| C | : | Katarakt |
| H | : | Hipertansiyon |
| M | : | Miyop |
| O | : | Diğer |
| DMÖ | : | Diyabetik Makuler Ödem |
| VGG | : | Visual Geometry Group |
| DRIVE | : | Digital Retinal Images for Vessel Extraction |
| STARE | : | Structured Analysis of the Retina |
| ODIR | : | Ocular Disease Intelligent Recognition |
| PKU | : | Pekin Üniversitesi |
| SG | : | Shanggong Medical Technology Co. Ltd. |
| ResNet | : | Residual Net |
| AUC | : | Eğri Altında Kalan Alan (Area Under the Curve) |
| DCNN | : | Derin Konvolüsyonel Sinir Ağları |
| ReLU | : | Düzleştirilmiş Doğrusal Aktivasyon Fonksiyonu |
| SGD | : | Stokastik Gradyan İniş |
| NAG | : | Nesterov Hızlandırılmış Gradyan |
| Adagrad | : | Adaptif Gradyan |
| Adadelta | : | Adaptif Delta |
| Rmsprop | : | Root Mean Square Error Probability |
| Adam | : | Adaptif Moment |
| Nadam | : | Nesterov Adaptif Moment |
| ICO | : | Uluslararası Sınıflandırma Standardı |
| CCRG | : | Amerikan Kooperatif Katarakt Araştırma Grubu |
| TP | : | Gerçek Pozitif (True Positive) |
| FP | : | Yanlış Pozitif (False Positive) |
| TN | : | Gerçek Negatif (True Negative) |
| FN | : | Yanlış Negatif (False Negative) |
| ROC | : | Receiver Operating Characteristic |
| FDA | : | Gıda ve İlaç Dairesi (Food and Drug Administration) |

# 1. GİRİŞ VE AMAÇ

Yapay zekâ ve derin öğrenme mimarileri son yıllarda sağlık alanında popüler çalışma alanıdır. Oftalmoloji, Cerrahi Tıp Bilimleri bölümleri içerisinde teknolojinin en çok kullanıldığı bölümlerden biridir. Dünyada sık görülen göz hastalıklarının tanı, takip ve sınıflamasında derin öğrenme mimarileri kullanılarak klinik sonuçlar elde edilmeye çalışılmaktadır. Fundus fotoğrafları, görme alanı testleri, optik koherens tomografi (Optik Koherens Tomografi-OCT) görüntülemeleri ile diyabetik retinopati (D), yaşa bağlı makula dejenerasyonu (A), glokom, hipertansiyon, miyop, katarakt, maküler ödem ve retina segmentasyonu gibi konularda derin öğrenme mimarileri üzerine çalışılmaktadır (Elangovan & Nath, 2020; Gulshan vd., 2016; Lee vd., 2017; Maji, Santara, Mitra & Sheet, 2016; Saranya & Prabakaran, 2020; Sogawa vd., 2020; Tan vd., 2018; Zhang vd., 2020). Erken tanı ve doğru zamanda yapılacak hasta yönlendirmesi ile insanların yaşam kalitesinin artırılması ve önlenebilir körlüğün önüne geçilmesi hedeflenmektedir (Karaküçük & Eker, 2020).

Bu bölümde yapay zekâ, tarihçesi ve kilometre taşları, veri ve yapay zekâ ile ilişkisi, derin öğrenme ve tarihçesi, tez çalışmasına ait literatür bilgisi, problemin belirlenmesi, çalışma hipotezi ve tezin amacı açıklanmıştır.

## 1.1. Yapay Zekâ nedir, Tarihçesi ve Kilometre Taşları

Yapay zeka, “normalde insanlar tarafından yerine getirilen düşünsel faaliyetleri otonom hale getirme” olarak tanımlanabilir (Chollet, 2017). Tarihte sıfırın bulunması bilgisayar programlamanın ilk adımı olarak düşünülebilir. İkili (binary) sistem bu bilgi üzerine kurulmuştur. Sıfır rakamının mucidi, El Harezmî (780-850), algoritmanın temellerini atmıştır. El-Cezerî (1136-1206), sibernetiğin ilk adımlarını atan ve ilk robotu yaptığı kabul edilen Müslüman alim ve mühendistir (Çırak & Yörük, 2016). Robot tanımını yaparken Leonardo da Vinci’ye ilham olmuştur.

Leonardo da Vinci (1400-1500), felsefe, sanat ve bilim alanlarında önemli eserler bırakmıştır. İnsana benzeyen, oturma, yürüme, ayakta durma işlevlerini yapan mekanik robot çalışmaları vardır (Hamet & Tremblay, 2017). Gottfried Leibniz (1600-1700), sayısal elektronik ve haberleşmenin temelini atmıştır. İntegral sembolü ve ilk mekanik hesap makinesini tasarlamıştır. Descartes düşünen makineler üzerine çalışmalar yapmıştır.

Descartes makinelerin insana benzer işler yapmayı başarsa da insan gibi bilinçli bir şekilde yapmayacağı için insan gibi düşünemeyeceklerini ortaya atmıştır (Haenlein & Kaplan, 2019).

Charles Babbage (1800-1900), endüstri devriminin ilk adımlarını atması ile birlikte artık kas gücünün yerini makineler almaya başlamıştır. Bilgisayar biliminin temelleri bu dönem atılmıştır diyebiliriz. Çünkü bugünkü merkezi işlem birimlerinin (CPU) atası sayılabilecek mekanik bilgisayarlar da bu dönemde ortaya çıkmıştır. Ada Lovelace tarihteki ilk programcı olarak anılır ve bernoulli sayılarını Babbege'in mekanik bilgisayarı üzerinde çalıştırmayı başaran programı yazmıştır. George Boole, bugün de günümüzde yapay zekanın babalarından biri olarak tanımlanan Geoffrey Hinton'un dedesi olan İngiliz bilim adamı, matematikçidir. Boole mantığının tasarımını yapmıştır ve otomasyonun önünü açmıştır (İnik & Ülker, 2017).

Massachusetts Teknoloji Enstitüsünden (MIT) bir mühendis ve bilim adamı olan Vannevar Bush [1900-1950], insanın ürettiği makinelerin gelecekte düşünebileceği konusunda eserler yayınlamıştır. Douglas Engelbart ise Stanford Üniversitesinde "İnsan Aklının Artırılması" isimli makalesi ile "Artırılmış zekâ" tanımını yapmıştır. 1945-1950'li yıllarda kriptograf, matematikçi ve enformasyon kuramcısı olan Claude Elwood Shannon, evrendeki tüm bilginin ikili sistemde ifade edilebileceğini savunmuştur ki bu aslında bugün bilim adamlarının o halde canlının zekası da ikili sistemde ifade edilebilir mi sorusunu sağlamıştır ve bu konu üzerine düşünmeye itmiştir. Bu yüzden de her ne kadar istatistik, enformasyon ve haberleşme teorisi alanlarında ünlü olsa da yapay zekâ konusunda da önemli isimlerden biridir. 1950'ye kadar geçen bu sürece yapay zekanın tomurcuklanma dönemi adı verilmiştir (Nilsson, 2014).

1950'lerin başında Alan Turing ile birlikte artık tomurcuklanma döneminin bittiği ve ilk keşiflerin başladığı dönem başlamıştır. İkinci dünya savaşı sırasında tasarlamış olduğu Turing makinesi, kestirim yapabilme özelliğine sahipti ve bu makine yapay zekanın ilk adımı olarak kabul edilmiştir. Alan Turing'in 1950 yılında Mind dergisinde yayınlanan "Computing Machinery and Intelligence" isimli makalesinde "Can Machines think?", makineler düşünebilir mi sorusunu soran ve aynı zamanda makinenin öğrenmeyi öğrenmesi ve evrimleşmesi üzerine çalışmıştır (Turing, 1950).

1956 yılında Amerika, New-Hampshire'da Dartmouth Kolejinde John McCarthy, Marvin Minsky, Claude Shannon gibi önemli bilim insanlarının yer aldığı iki aylık yaz çalıştayı yapılmıştır ve ilk kez "Yapay Zekâ- Artificial Intelligence" kavramı bu çalıştayda kullanılmıştır ve resmileşmiştir. 1955-1965 yıllarında, ilk sohbet robotu olan ELIZA büyük ses getirmiştir. Çok sohbet ettiği söylenemeyen bu sohbet robotu taklitten öteye geçememiştir.

Cahit Arf, 1959'da "Makineler Düşünebilir mi ve Nasıl Düşünebilir?" isimli çalışmasını Atatürk Üniversitesi'nde yayınlamıştır (Arf, 1959). Bu süreçte, 1958'de Psikolog Frank Rosenblatt, Bir biyolojik sinir hücresinin matematiksel tanımını yapmıştır ve ancak "AND-OR" gibi mantık kapılarını karşılayabilirken bir sinir hücresinin "XOR" problemini çözemediğini ve birden fazla sinir hücresi ve paralel seri bağlantılara ihtiyaç duyulduğunu, Marvin Minsky ve Seymour, 1969'da yayınladığı "Perceptrons" isimli kitabında, akıllı sistemler için birden fazla sinir hücresi ve katmana ihtiyaç olduğunu savunmuştur (Minsky, & Seymour, 1969). 1986 yılında Boole'un torunu olan Geoffrey Hinton, günümüzde kullandığımız öğrenme algoritmalarının temelini atmıştır ve 2000'li yıllarda insansı başarılara erişmekte önemli adımların atılmasını sağlayan birçok bilim insanının akıl hocasıdır (Nilsson, 2014).

Türkiye'de, 1992 yılında "1. Türk Yapay Zekâ ve Yapay Sinir Ağları Sempozyumu" Bilkent Üniversitesinde gerçekleştirilmiştir. Bugün Facebook-AI direktörü olan Yann LeCun bilgisayarlı görü konusunda önemli başarılara imza atmıştır. El yazısı rakamların görüntülerden tanınmasını ve konvolüsyonel sinir ağı modellerinin yeniden popüler hale gelmesini sağlayan kişidir.

Veri kısıtı işlemci kapasite gücünün yeterli olmaması o yıllarda biraz daha yavaş ilerlenmesine sebep olmuştur. 1997 yılında, yapay zekâ çok popüler olan konu haline gelmişti. Çünkü, 30 bilgisayarı daha önce yenebilmiş olan dünya satranç şampiyonu Garry Kasparov IBM'in geliştirdiği Deep Blue yazılımına yenilmişti ve daha sonra insanlığı ele geçiren yapay zekâ bilim kurgu yayınlarında büyük patlama yaşandı. Aslında yine bugünkü gibi çok popüler bir dönem geçirmiştir (Campbell, Hoane & Hsu, 2002).

Türkiye'de 2000'li yıllarda akıllı sistemleri ve sinyal işleme konularını odak alan akademik konferanslar düzenlenmeye ve yaygınlaşmaya başladı. Stanford Üniversitesi Prof. Dr. Fei-Fei Li önderliğinde, 2009 yılında en büyük açık ve ücretsiz görsel veri seti ImageNet, büyük ölçekli görüntü tanıma yarışmasında (ILSVRC) yayınlanmıştır (Deng, Dong, Socher, Li, Jia-Li & Li-Fei, 2009).

ImageNet veri setinin yayınlanması ile birlikte bilgisayarlı görü alanında özellikle nesne tanıma konusunda kısa sürede insan başarısının üzerine çıkılması sağlanmış oldu. 2000'li yılların başında başlayan otonom araçlarla ilgili 2010 yılında önemli şirketler test sürüşlerine başladıklarını açıklamışlardı. 2010 yılından bu yana otonom araçlar konusunda da hızlı bir gelişme olmuştur. Tesla 2014 yılında, yapay zekâ destekli otopilot çalışmalarını duyurmuştur (Rosenzweig & Bartl, 2015). Aynı yıl, Future of Life enstitüsü, yapay zekanın etik, hukuki ve sosyal boyutlarını ele aldı. Ian Goodfellow ve çalışma arkadaşları, sentetik veri üretimi ile gerçekçi insan yüzleri üretmiştir (Goodfellow, Abadie, Mirza, Xu, Warde-Farley, Ozair & Bengio, 2014). 2017 yılında, Google DeepMind yazılımı AlphaGo ile dünya şampiyonlarını Go oyununda yendi (Sang-Hun, 2017). Aynı yıl, OpenAI, DOTA oyununda insan yarışmacıları, 2019 yılında da dünya şampiyonlarını yendi. 2018 yılında, Amazon Go kasiyersiz akıllı alışveriş ile stok takibi gibi temassız alışverişin önü açılmış oldu (Polacco & Backes, 2018). Sesli asistan ve sohbet robotları alanında da önemli gelişmeler yaşanmıştır. Siri, Alexa, Google, ayrıca tavsiye sistemleri, bununla birlikte sosyal medya ve kişisel verinin önemi de bu yıllarda daha net anlaşılmış oldu. Brexit, Cambridge Analytica, Facebook gibi veri paylaşımları ile ilgili skandallar da yine bu yıllarda gündeme gelmiştir (Lopez, Quesada & Guerrero, 2017).

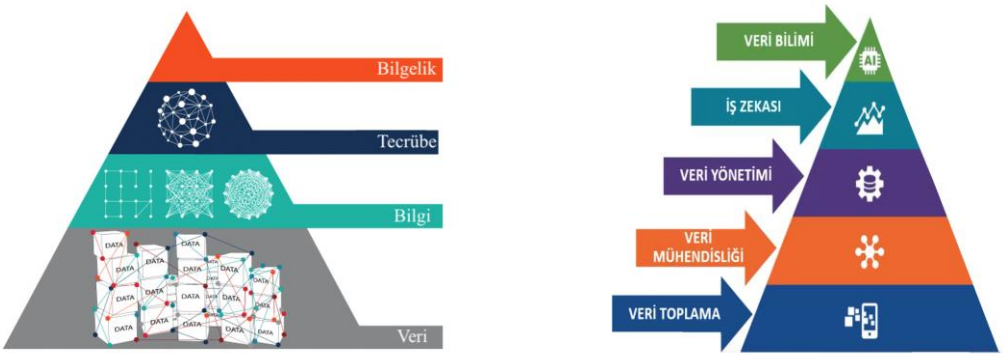
Kişisel Verileri Koruma Kanunu (KVKK-GDPR) üzerine kişisel mahremiyet ve toplumsal etkisi göz önüne alınarak son yıllarda çalışılmaya başlandı. Tüm bu yapay zekâ çalışmalarının gelişmesinde, verinin çokluğu, paylaşılabilir olması, grafik işlem birimi (GPU), tensör işlem birimi (TPU) gibi donanımların gelişmesi, bulut ortamlarına erişilebilirliğin kolaylaşması gibi imkanların itici güç olduğu bilinmektedir (Kızrak, 2020).

## 1.2. Veri nedir ve Yapay Zekâ ile ilişkisi

Veri, bir ölçüm, deney, gözlem, sayım ya da bir araştırma yolu ya da ham/işlenmemiş bilgi parçacığı olarak tanımlanabilir. Türk Dil Kurumu, veriyi; bir araştırmanın, bir tartışmanın, bir muhakemenin temeli olan ana öge, muta, done olarak tanımlamıştır. Bir veri tek başına bir anlam ifade etmezken birden fazla veri veya birden fazla konudaki bir grup veri veya ilişkili veriler bize basit ya da karmaşık bazı problemleri çözmek için yol gösterebilir (Silahtaroğlu, 2008; TDK, 2021).

Özellikle 18. ve 19. yüzyıllarda maden, tarım ve hava olayları gibi konularda insanlar gözlemler yapmaya başlamış ve yaptıkları bu gözlemleri kaydetmişlerdir. Bu gözlemler sonucunda aldıkları kayıtlarla birlikte geleceğe yönelik tahminlerde bulunarak örneğin gelecek yıl hangi tohumları toprağa ekmeleri gerektiğini, madeni nerede aramaları gerektiğini, hava durumunun nasıl olacağını kestirmeye başlamışlardır (Kızrak, 2020).

Günümüzde veriler dijital ortamda, kolay toplanabilir, kaydedilebilir ve paylaşılabilir durumdadır. Bu yüzden artık veri grupları yerine daha büyük veriler ile karşılaşmaktayız.




**Şekil 1.1.** Bilgi hiyerarşisi ve veri bilimi piramidi (SenseCorp, 2021)

Bilgi hiyerarşisi ve veri bilimi piramidi Şekil 1.1'de yer almaktadır. Piramidin temelinde veri yer almaktadır. Bu veriden bilgi elde edilmektedir. Kullanacağımız derin öğrenme yöntemleri ile bir tecrübe çıkarılmaktadır. İnsanlar veriye maruz kaldıkça o veriden bazı bilgiler çıkararak tecrübe sahibi olmaktadır ve bu bilgi, tecrübe arttıkça bilgeliğe doğru ilerleyen bir yolu temsil etmektedir.

Manyetik Rezonans (MR) görüntülerinden oluşan bir veri setimiz olduğunu düşünelim. Bu, piramidin tabanındaki veriyi temsil etmektedir. Akciğer tomografisi mi beyin tomografisi mi olduğunu anlayabildiğimiz noktada o veriden bilgi çıkardığımız ikinci aşamaya geçmekteyiz. Beyin tomografisi tespit ettiğimizi düşünelim. Bu beyin tomografisinde anomali olup olmadığı gibi bir bilgi daha çıkarabiliyor isek buna artık tecrübe dediğimiz noktaya ulaşmış oluyoruz. Ardından o anomalinin nasıl tedavi edildiği edilebileceğine dair öneride de bulunabiliyorsak bu aşamaya bilgelik denir. Veriden anlam çıkardıkça yapay zekanın istediğimiz noktası en gelişmiş bilgelik noktasına ulaşmış olacaktır (Kızrak, 2020).

Günümüzde verinin hacmi, üretimi, transferi giderek kolaylaşmakta ve yaygınlaşmaktadır. Bu sayede, büyük veri ile birçok hedefe ulaşmak mümkün hale gelmektedir. Büyük verinin tanımı, 3 karakteristik özelliği olan İngilizcede yer alan Volume, Velocity ve Variety kelimelerinden türeyen (3V) bir kavramdır (Aktan, 2018).

Hacim (Volume), büyük verinin günlük bazda oluşturulan verinin sayıca ne kadar fazla olduğunu gösteren bir parametredir. Veri hacmi, verilerden çıkarılacak değer ile ilgili bir fikir verdiği için çok önemlidir. Ayrıca, belirli bir veri kümesinin büyük veri olarak değerlendirilmesi, verinin hacmine bağlıdır. Verilerin boyutu terabayt ve petabayt arasında değişebilir. Bu nedenle hacim, büyük veri ile çalışırken dikkate alınması gereken önemli bir parametredir.

Hız (Velocity), verilerin alınması ve verilerin işlenmesi için geçen süre olarak düşünülebilir. Büyük veriyi geleneksel yöntemlerle incelemek istediğimizde günler ve haftalar sürebilir. Bu nedenle hız ihtiyacının olması büyük veri için en önemli parametrelerdendir. Hız, verilerin ne kadar hızlı üretildiğini ve işlendiği, veriden elde edilecek değeri belirler. Veri akışının büyüklüğü ve kapladığı hacim inanılmaz büyüktür. Bu noktada verinin alınma ve işlenme hızı önem kazanmaktadır.

Çeşitlilik (Variety), verilerin farklı formatlarda oluşturulması ve farklı gruplardan elde edilmesini ifade etmektedir. Gelişen teknoloji ve yeni kullanıcı alışkanlıkları ile birlikte veri düşünüldüğünde, sadece düz metinler veya veri tabanları aklımıza gelmemelidir.

Veri, yapılandırılmış, yapılandırılmamış, sayısal, ses, video, fotoğraf, finansal işlemler gibi çok çeşitli veri türleri anlamına gelmektedir. Tüm bu farklı veri türleri, bir anlamda birbiri ile ilişkisiz ve işlenmeyi beklemektedir.

Çok çeşitli ve büyük hacimli verileri depolayıp, işlemek için yeni bir çözüm olmalıdır. Uygulamalar ve kullanıcı alışkanlıkları değiştiği için geleneksel yöntemler yetersiz kalmaktadır. Bu nedenle büyük veri problemleri 3V ile tanımlanabilir (O'leary, 2013).

Büyük veriyi tanımlayan 3V'ye ek olarak, Doğruluk (Veracity) ve Değer (Value) bileşenleri de dahil edilerek 3V yerine 5V'den söz edilmektedir.

Doğruluk (Veracity), verinin güvenilir ve doğru bilgiler içermesi gerektiğini ifade eder. Veriler içinde doğru olmayan ve anlamsız kayıtların sağlıklı sonuç alabilmek adına temizlenmesi gerekmektedir. Doğru olmayan veri, anlamlı bilgiye dönüştürülemez.

Değer (Value), kayıt altına alınan datanın büyük veri olarak kabul edilebilmesi için başka bir kriter de verilerin işlendikten sonra artı değer sağlamasıdır. Toplanan ve içerisinden işe yarayacak anlamlı bilgi elde edilemeyen bir veri tabanı, hiçbir işe yaramaz. Bu nedenle mevcut verilerin doğru bir şekilde işlenmesi ve anlamlı sonuçlar üretilebilmesi önemlidir (Uddin & Gupta, 2014).

## 1.3. Derin Öğrenme nedir, Tarihçesi

Derin öğrenme, 2006 yılından sonra popüler olup bu ismi almış olsa da yapay zekâ tarihi boyunca farklı tanımlamalar altında çalışılan bir araştırma alanıdır. Özellikle yapay sinir ağlarının eğitimi için verinin ve model boyutlarının artması, donanımların bu işlem gücünü karşılayabilir olması, araştırmaların üniversite laboratuvarlarından çıkıp projelendirilip gerçek hayatta karşılık bulmasına olanak tanımaktadır. Karmaşık problemlerde klasik yaklaşımlardan daha başarılı sonuçlar verdiği sayısız araştırmayla ortaya çıkmıştır. Geçtiğimiz yıllarda da yapay zekanın babaları olarak adlandırılan Geoffrey Hinton, Joshua Bengio ve Yann LeCun Turing ödülünü almaya hak kazanmıştır (Hinton, Bengio &, LeCun, 2019). Bu üç isim de bugün derin öğrenme konusunda önemli eserlere sahip bilim insanlarıdır.

Yapay zekâ araştırmalarının bir alt dalı olan makine öğrenmesi, bilgisayar sistemlerine akıl yürütme, bilgi sunumu, planlama ve öğrenme gibi işlevleri sağlamak için istatistiksel teknikleri kullanan bir bilgisayar bilimi alanıdır (Alpaydın 2011). Makine öğrenmesi (Machine Learning-ML), bilgisayarların örnek veri ya da geçmiş deneyimi kullanarak başarımlarını arttıracak biçimde programlanmasıdır.

Elimizde bazı parametrelere bağlı olarak tanımlanmış bir model, veri ya da geçmiş deneyim üzerinde modelin başarımını ölçmek için bir ölçüt tanımlıdır. Amaç, modelin parametrelerini bir başarım ölçütüne göre en iyi yapan parametre değerlerini bulmaktır. Model, gelecekle ilgili tahmin yapmak için kullanılabilen, tahminci veya veriden bilgi çıkarmaya yönelik açıklayıcı bir model olabilir. Makine öğrenmesi ile bilgisayarların karar verebilmesi için öncesinde modelin öğrenimini tamamlaması gerekmektedir. Bu öğrenme işlemi doğrudan sonucu olan verileri kullanarak, verilerdeki kalıpları arayarak ya da keşfederek gerçekleşmektedir. Makinenin öğrenme sürecinde verilere ait özelliklerin ML algoritmaları için önceden hazırlanması gerekmektedir. Başka bir deyişle ML algoritmaları kendisine gönderilen ham verileri doğrudan kullanamamaktadır (Alpaydın 2011).

Ham verilerden anlamlı verileri (veri özelliklerini) ortaya çıkarabilmek için belirli teknikler yardımıyla özellik çıkarımı yapılması gerekmektedir. Elde edilen özellikler, çeşitli algoritmalar üzerinde çalıştırılarak makine öğrenimi gerçekleştirilerek veri üzerinde ayrıştırma yapılmaktadır. İşlenmesi istenen mevcut verilerin türüne göre öğrenme yöntemi değişiklik göstermektedir. Bu öğrenme yöntemleri denetimli, denetimsiz ve takviyeli öğrenme olarak üç ana kategoriye ayrılmaktadır (Alpaydın 2011).

Denetimli öğrenme, belirli bir veri kümesi üzerinde, $x$ giriş değerlerine karşılık bilinen $y$ çıkış değerleri kullanılarak gerçekleştirilen eğitim yöntemidir. Bu tür yöntemler verilerden öğrenerek yeni veri kümesi için sınıflandırma ya da tahminleme yapar (Marsland 2015).

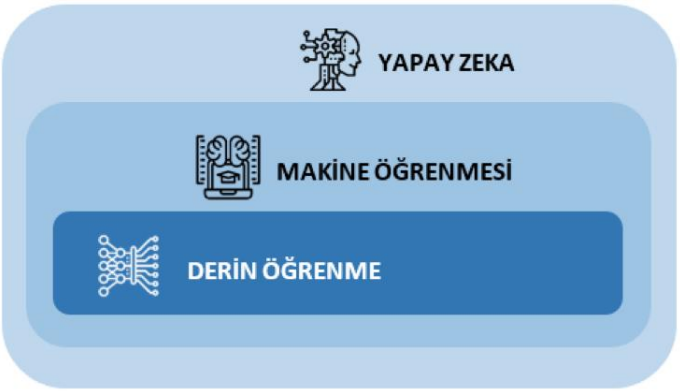
Denetimsiz öğrenme, belirli bir veri kümesi üzerinde $x$ giriş verilerine karşılık hiçbir çıkış değerine sahip olmayan verilerin hata sinyali vermeden kendi kendine öğrenmesi olarak tanımlanmaktadır.

Öğrenme işlemini verilerin benzerliklerinden, ilişkilerinden ve dağılımından keşfederek gerçekleştirmektedir. Kümeleme, doku segmentasyonu, konuşma tanıma gibi uygulamalarda kullanılmaktadır (Goodfellow, Bengio, ve Courville 2016; Saravanan ve Sujatha 2018; Sathya ve Abraham 2013).

Takviyeli öğrenme, belirli bir veri kümesi üzerinde $x$ giriş değerlerine karşılık $y$ çıkış değerleri bilinmektedir. Fakat öğrenme sırasında çıkış değerleri ağa gösterilmemektedir. Bunun yerine $f(x)$ değerlerine karşılık üretilen çıkış değerlerinin gerçek $y$ çıkış değerine göre doğru ya da yanlış olduğunu gösteren bir sinyal göndermektedir (Öztemel 2008). Başka bir deyişle model ortamlarda deneme yanılma yoluyla keşfederek öğrenimini gerçekleştirmektedir. Belirli zamanlarda gelen sinyaller ajanlar tarafından algılanır, algılanan sinyale karşılık bir eylem üretilir ve bu eyleme karşılık çevreden ödül değeri olarak adlandırılan tepki üretilerek ajana geri dönüş gerçekleştirilir (Schmidhuber 2015).

Makine öğrenmesi alanının bir alt dalı olan Derin öğrenme (Deep Learning-DL) ile özellik çıkarma işleminde uzmana ihtiyaç duymadan veri içindeki bilgiye erişilebilir ve sınıflandırma yapılabilir. Derin Öğrenme ile görüntülerin sınıflandırılmasının geleneksel yöntemlere göre çok daha başarılı olduğu görülmektedir.

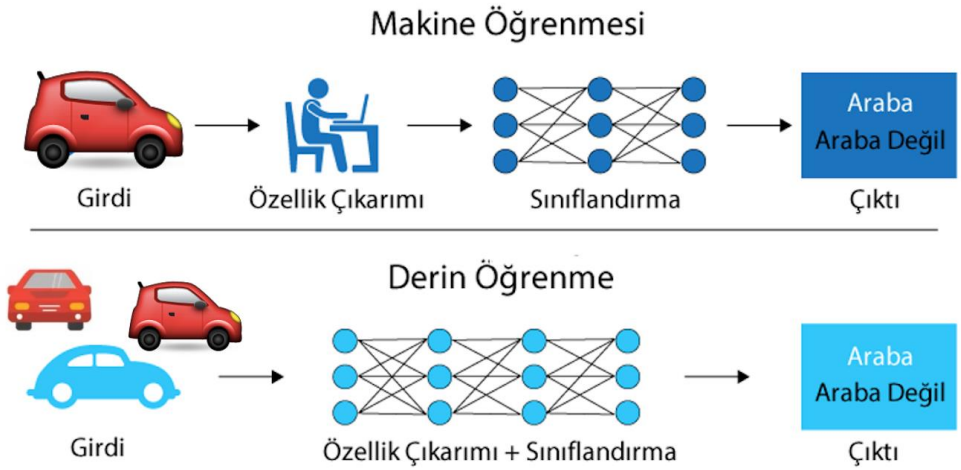
Konvolüsyonel Sinir Ağları (Convolutional Neural Network-CNN) derin öğrenmenin temel mimarisi olarak kabul edilmektedir (Krizhevsky vd., 2017). Bu ağlar özellikle görüntü sınıflandırma alanında sıklıkla kullanılmaktadır (İnik & Ülker, 2017). Şekil 1.2'de yapay zekâ, makine öğrenmesi ve derin öğrenme alanlarının kapsayıcılığı gösterilmektedir.




**Şekil 1.2.** Yapay zekâ, makine öğrenmesi ve derin öğrenme (Ucuzal, 2020)

Derin öğrenmenin temel anlamda beyin benzetimi yapmak gibi bir amacı yoktur. Ancak sinir bilim ve hesaplamalı sinir bilim kaynakları bazı derin öğrenme çalışmalarının da ilgi odağı halindedir. Çağdaş derin öğrenme, bilgi kuramı, olasılık ve istatistik, lineer cebir gibi birçok alandan beslenmektedir.

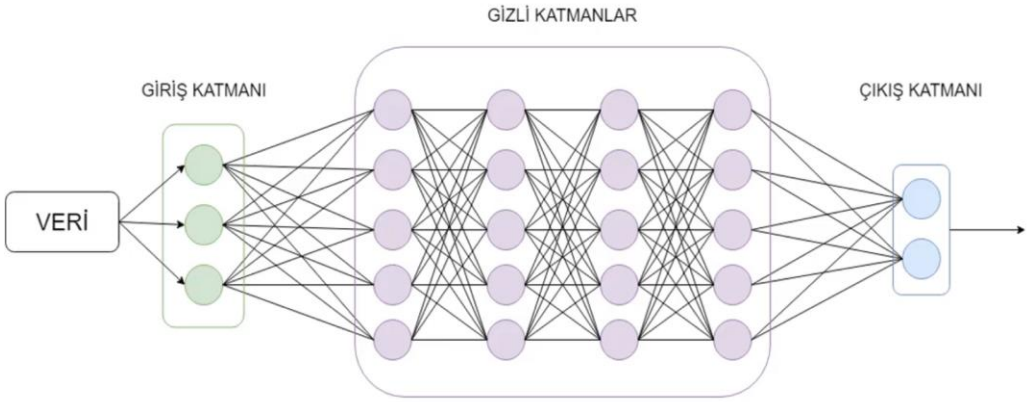
Derin öğrenmenin makine öğrenmesinden ayrılan en önemli özelliği öznitelik çıkarma durumudur. Şekil 1.3'de araç görüntüsünün sınıflandırılmasında makine öğrenmesi ve derin öğrenme arasındaki farklılıklar gösterilmiştir. Makine öğrenmesinde bir arabayı tanımaya çalıştığımızda öznitelik çıkaran bir uzman kişiye ihtiyaç duyulur. Arabanın özelliklerine dair bilgiler çıkarılır ve bu bilgiler sinir ağı ya da makine öğrenmesi modeline verilir. Bunun için çok fazla araba görüntüsüne ihtiyaç yoktur. Bu özellik çıkaran kişi bu konuda uzman biridir. Derin öğrenmede öznitelik çıkarma işlemi bir kişi ya da uzman tarafından gerçekleştirilmez. Ancak girişte eskisinden çok daha fazla çeşitliliğe sahip araba görüntülerine ihtiyaç duyulur. Bunun için on ya da yüz adet yerine binlerce belki milyonlarca veriye ihtiyaç duyulmaktadır. Bu durumda uzman kişiye ihtiyaç ortadan kalkmaktadır. Öznitelik çıkarma işlemi, derin öğrenmenin sinir ağları boyunca gerçekleşir. Özellikle büyük veri varlığında makine öğrenmesi problemlerinden çok daha başarılı sonuç verdiği kanıtlanmıştır (Taylor, 2017).




**Şekil 1.3.** Makine öğrenmesi ve derin öğrenme arasındaki farklılıklar (Ucuzal, 2020)

Günümüzde uygulama yöntemlerinin başında gelen derin öğrenme, kısaca insanların çözebildiği zekâ gerektiren karmaşık problemleri çözmeyi hedefleyen veri, algoritma, model ve donanım bilgisi gerektiren sistemlerin oluşturulması üzerine disiplinler arası bir çalışma alanıdır. Derin öğrenme, uygulandıkları alana göre çeşitlilik göstermektedir.

Veri, giriş katmanı, gizli katmanlar ve çıkış katmanından oluşan ikili sınıflandırma problemi için basit bir derin sinir ağı gösterimi Şekil 1.4'de verilmiştir. Veri giriş katmanlarına alınır. Öznitelikler gizli katmanlar aracılığı ile çıkarılır. İki sınıflı çıkış katmanı için bilgiler olasılıksal yaklaşımlar ile incelenir (Sewak, Karim & Pujari, 2018).




**Şekil 1.4.** İkili sınıflandırma problemi için basit bir derin sinir ağı gösterimi (Kinsley & Kukiela, 2020)

Geoffrey Hinton, öğrencileri Alex Krizhevsky ve Ilya Sutskever 2012 yılında yayınladıkları makalede AlexNet mimarisi ile aslında 1980'li yıllarda temeli atılan derin sinir ağlarını, ImageNet veri seti üzerinde denediğinde çok başarılı sonuçlar elde etmiştir. Daha önceki yıllarda uygulanan destek vektör makineleri gibi klasik makine öğrenmesi modelleri ile elde edilen sınıflandırma sonuçlarının yaklaşık iki katı kadar daha başarılı sonuçlar elde edilmiştir (Krizhevsky, Sutskever & Hinton, 2012). 2012 yılından sonra derin öğrenme yöntemleri daha sık kullanılmıştır. Derin öğrenme mimarisinde gizli katmanların sayısı arttıkça eğitim hatası da azalmıştır. Ancak her zaman katman sayısının fazla olması eğitim hatasının az olacağı anlamına gelmemektedir. Özellikle son yıllarda EfficientNet gibi daha güncel derin öğrenme modellerinde daha az katman ile daha başarılı sonuçlar elde edilmeye odaklanılmıştır (Tan & Le, 2019).

Farklı derin öğrenme modelleri farklı veriler üzerinde daha başarılı sonuçlar elde etmektedir. Örneğin görüntüler için örüntüleri çıkarmada konvolüsyonel sinir ağları, zaman serisi ve bellek bilgisinin gerekli olduğu veriler için yinelemeli sinir ağları olarak genelleştirdiğimiz uzun vadeli kısa dönem bellek modelleri (LSTM) kullanılmaktadır.

2014 yılında geliştirilen üretici çekişmeli ağlar (GAN), üretici ağların son yıllarda yeniden popüler olmasını sağlamıştır. Stil transferi konusunda sanatçılardan esinlenerek resim, müzik ve şiir gibi eserlerin yeniden üretilmesi ile çok konuşulur hale gelmiştir. Buna rağmen sentetik veri üretme, düşük çözünürlüklü görüntüleri iyileştirme konularında oldukça başarılıdır ve insan yüzü üretme konusunda insanları aldatabilecek noktaya ulaşmıştır (Goodfellow vd., 2014).

## 1.4. Tez Çalışması Literatür Bilgisi

Oftalmoloji, tıp bilimlerinde teknolojinin en çok kullanıldığı bölümlerden biridir. Teknolojik gelişmeler, göz hastalıklarının tanı ve tedavi yaklaşımlarını değiştirebilmektedir. Gerek tanı koyma gerek tedavi yolu belirleme aşamasında oftalmologlar yapay zekâ alanındaki gelişmeleri hevesle takip etmektedir. Yaşlanan dünya popülasyonu ile erken tanı giderek önem kazanmaktadır. Farklı teknolojik cihazlarla alınan görüntülemeler zaman zaman bilgi kirliliğine sebep olsa da bu verilerin sınıflandırılması ve işlenmesi ile faydalı bilgiye ulaşmak yapay zekâ sistemleri ile erişilmek istenenler arasındadır (Karaküçük & Eker, 2020).

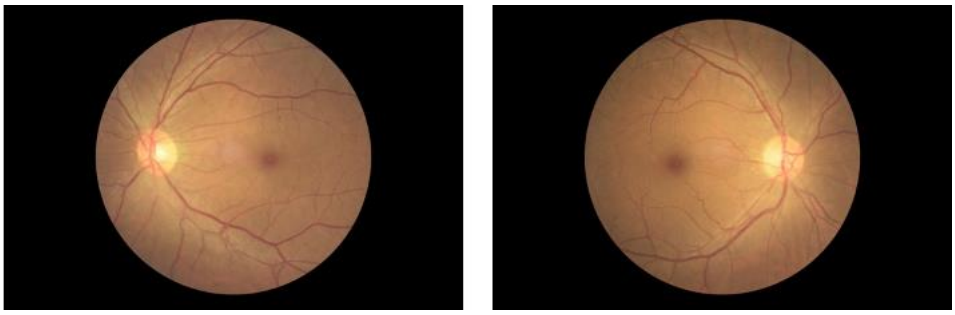
Retina, gözümüzün arkasında bulunan, gelen ışığı algılayan ve beynimize görüntü gönderen bir doku tabakasıdır. Merkezinde makulanın bulunduğu, net görüş sağlayan, okuma, yazma ve araç kullanma, sürüş yeteneği gibi günlük hayatta çok önemli ve hayati görevlerde bize yardımcı olan sinir dokusudur. Bu sinir dokusu, görmeyi etkileyebilecek çeşitli bozukluklardan veya hastalıklardan etkilenebilir. Günümüzde retina patolojileri, dünya çapında çocukluk çağı körlüğünün en yaygın nedenlerinden biri olarak görülmektedir. Ülkeler zenginleştikçe ve kişi başına düşen gelir arttıkça, körlük prevalansı azalır, körlüğe sebep olan nedenler değişir. Dünyanın en fakir ülkelerinde körlüğün önde gelen nedeni katarakttır.

Latin Amerika gibi ortalama gayri safi yurtiçi hasılaya (GSYİH) sahip bir ülkede ekonomik iyileşme sayesinde katarakt ameliyatı çoğunlukla erişilebilir ve uygulanabilir olduğu için körlüğün önde gelen nedeni glokom ve diyabetik retinopatidir. Yüksek GSYİH' ye sahip ülkelerde glokom ve katarakt çok yaygın ve önemli patolojiler olmaya devam etmektedir. Ancak körlük, önlenebilir diyabetik retinopati gibi diğer retina hastalıklarından kaynaklanmaktadır ve erken evrelerinde tedavi edilebilir (Gilbert, 2001).

Diyabet, gelişmekte olan ülkelerde giderek artan bir sorundur. Diyabeti olan kişilerde en sık hasar gören organlardan biri gözdür ve diyabetik retinopati (D), diyabet hastalığının erken dönem komplikasyonu olarak karşımıza çıkar. D körlüğe sebep olabileceğinden erken tespit edilmesi önem taşımaktadır. Dünya Sağlık Örgütü'ne (WHO) göre Hindistan'da Diyabetten etkilenen 31,7 milyon insan vardır ve 2030'da 79,4 milyona çıkması beklenmektedir (Gadkari, Maskati, ve Nayak 2016). Dünya çapında 2030 yılına kadar 400 milyondan fazla diyabetik retinopati hastası olacağı, 2020 yılında 80 milyon kişinin glokom hastası olacağı belirtilmiştir (Costagliola, Dell'Omo, Romano, Rinaldi, Zeppa & Parmeggiani, 2009).

Tedavisi olmayan birçok retina dejenerasyonu vardır. Nitelikli ve iyi donanımlı bir retina cerrahı olduğu sürece, hastalara ayrıntılı bir açıklama ve net bir prognoz ile doğru teşhis yapılır ve bu komplikasyonlar tedavi edilebilir. Gözlenebilir eğilim göz önüne alındığında, retina hastalıklarının dünyanın her yerinde büyüyen bir sorun olduğu bilinmektedir (Yorston, D. 2003).

Gelişmiş ülkelerde hastalıkların tespiti için oftalmologlar "retinopati veya fundus fotoğrafçılığı" adı verilen standart bir tıbbi görüntüleme aracı kullanırlar. Hızlı ve kolay bir prosedürle uzman, morfolojisinin ve yapılarının (optik sinir, kan damarları, makula, retina vb.) açıkça görülebildiği göz fundusunun yüksek kaliteli renkli bir fotoğrafını elde edebilir. Şekil 1.5'de görüldüğü gibi medikal görüntüleme ile hastaya ait fundus görüntüleme elde edilir (Saine & Tyler, 2002).

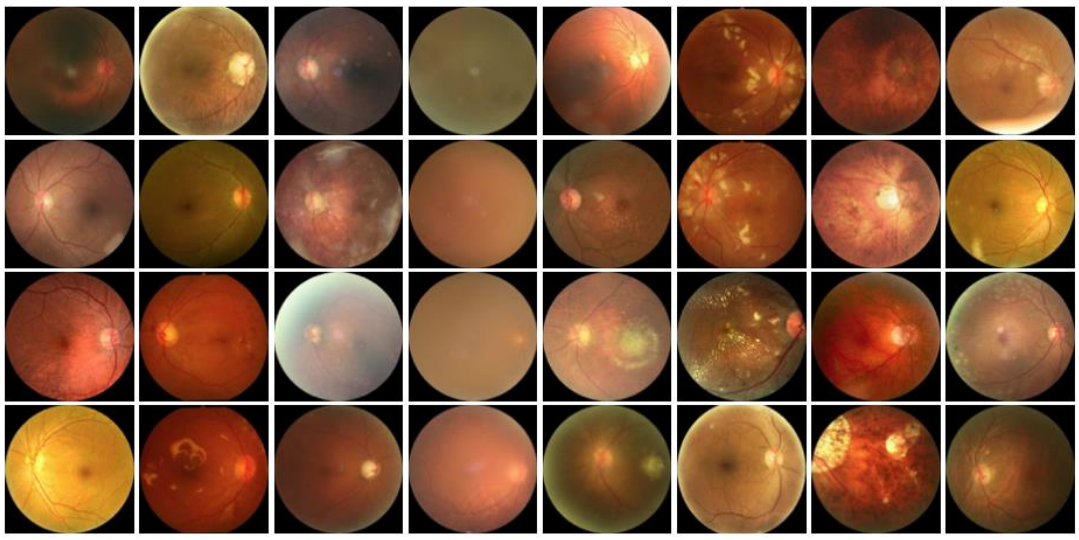


**Şekil 1.5.** Herhangi bir anomali olmayan sol ve sağ göz retinografisi (ODIR, 2019)

Retina hastalıklarının tedavisi için teknolojik ilerlemeler sayesinde ekipman pahalı olsa da günümüzde gelişmekte olan ülkelerde daha çok kullanılmaktadır. Yaşlanan nüfusla birlikte nitelikli personele olan ihtiyaç artmaktadır. Bu durum, retina hastalıklarında uzmanlaşmış daha fazla göz doktoruna ihtiyacımız olduğunu göstermektedir. Görüntüleme prosedürü bir retina kamera kullanılarak gerçekleştirilir (Aptel, Denis, Rouberol & Thiyolet, 2008).

Diyabet, glokom, katarakt ve makula dejenerasyonu gibi hastalıkları ve körlüğü önlemek için etkili, güvenli ve ekonomik yol her yıl yapılması mümkün olan göz kontrol muayenesidir. Bu temel prosedür gerçekleştirilerek hastalıkların hızlı tespit edilmesi, olası lezyon oluşumu ve ilerlemesinin önüne geçilebilir.

Fundus testi ile tespit edilebilen yaygın patolojilere ait fundus görüntüleri Şekil 1.6'da yer almaktadır. Sırası ile, 1) Normal (N), 2) Diyabetik Retinopati (D), 3) Glokom (G), 4) Katarakt (C), 5) Yaşa Bağlı Makula Dejenerasyonu (A), 6) Hipertansiyon (H), 7) Miyop (M), 8) Diğer hastalıklar (O) ifade etmektedir.



1 2 3 4 5 6 7 8

**Şekil 1.6.** Yaygın patolojilere ait fundus görüntüleri (ODIR, 2019)

Şekil 1.6'da yer alan patolojilerin bazıları, hastalık ilerlerse körlüğe yol açabilir. Diyabetli ve yaşlılarda Diyabetik Retinopati gelişme olasılığı daha yüksektir ve tedavi edilmezse körlüğe sebep olabilir. Glokom, göz içi basıncında sürekli bir artış ile optik sinire ilerleyen hasara neden olan bir hastalıktır. Glokomun kesin bir tedavisi yoktur. Ancak, cerrahi müdahale ile büyük ölçüde tedavi edilebilir. Bu patoloji, dünyadaki görme kaybının önde gelen nedenlerinden biridir (Bourne, 2021).

Katarakt, yaşlı yetişkinleri etkiler ve görmeyi zorlaştıran bir dizi yamanın geliştiği gözün iç kısmındaki fizyolojik değişiklik nedeni ile görme azalmasına neden olur. Makula, merkezi görüşten sorumludur ve içinde sıvı birikirse görme bozulur. Makula dejenerasyon genellikle 70 yaşın üzerindeki kişilerde görülür ve genellikle hastalığın erken aşamalarında herhangi bir semptom göstermez. Bu da göz fundusunu erken teşhis etmede önemli bir süreç haline getirir.

Zhou ve ark. (2019) tarafından yayınlanan çalışmada, tıbbi fundus görüntülerinin tedavisinde iki önemli araştırma alanı bulunmaktadır. Birincisi, hastalık şiddetine göre sınıflandırma, ikincisi ise görüntüdeki pikselleri analiz ederek lezyon segmentasyonuna dayanan sınıflandırma işlemidir. Her iki alan da günümüzde hem oftalmolojide hem de diğer medikal görüntüleme alanlarında derin öğrenme yöntemlerinin uygulandığı genel bir sınıflandırma problemi olarak görülmektedir. Oftalmologlar, gözün ve çevresindeki yapıların doğrudan veya dolaylı görselleştirilmesi yolu ile örüntü tanımaya dayalı hastalıkları teşhis etmektedirler. Teşhis teknolojileri, oftalmoloğun kararını vermesine yardımcı olan eşlik eden bilgileri sağlar. Göz hastalıklarının fundus fotoğrafı ile değerlendirilmesi oftalmoloji alanını derin öğrenme mimarilerinden faydalanmak için mükemmel hale getirmiştir.

Derin öğrenme mimarilerinin dahil edilmesi, oftalmolojinin farklı alanlarında uygulanmaya başlanmıştır (Abramoff, Lou, Clarida, Amelon, Folk & Niemeijer, 2016; Grewal, Oloumi, Rubin & Tennant, 2018; Lee vd., 2017). Bu alanlar, fundus görüntüleme, optik koherens tomografi (OCT) gibi medikal görüntüleme yöntemleri ile diyabetik retinopati, erken retinopati, glokom, makula ödem ve makula dejenerasyonun saptanmasında kullanılmaktadır.

Konvolüsyonel Sinir Ağları (Convolutional Neural Network-CNN) mimarisi kullanan derin öğrenme modelleri ile ağlar, ağın patolojik işaretleri dikkate aldığı çeşitli özellik türlerini belirlemek için bir dizi görüntü işleme filtresi kullanır. Özellik çıkarma, bir eğitim seti öğrenildikten sonra gerçekleşir ve örüntü sınıflandırma yöntemlerinin ayrılmaz bir parçasıdır. Dolayısı ile derin öğrenme, farklı hastalıkların çeşitli özelliklerini ölçebilen en uygun görüntü işleme filtrelerini ve araçlarını belirleyebilen bir algoritma olarak görülebilir (Hijazi, Kumar & Rowen, 2015).

Derin öğrenme modelleri ile alanında uzman kişiler görüntüleme üzerinde çalışarak karşılaştırma yapmaktadır. AlexNet (2012), sekiz katmanlı CNN mimarisi, (Krizhevsky vd., 2012), yirmi iki katmanlı GoogLeNet (Szegedy vd., 2014), 16-19 katmanlı VGGNet (Simonyan & Zisserman, 2014) ve 18-152 katmanlı Deep Residual Net (He, Zhang, Ren & Sun,2016) büyük ölçekli görüntü tanıma yarışması olan ImageNet (Large Scale Visual Recognition Challenge-(ILSVRC)) sınıflandırma problemi üzerinde çalışan derin konvolüsyonel sinir ağı mimarilerdir. Bu sinir ağları, görüntü sınıflandırma konusunda sıklıkla kullanılan derin öğrenme mimarileridir.

Derin öğrenmenin oftalmolojide ilk uygulama alanı olarak tarama testlerinde kullanılması beklenmektedir. Glokom, Diyabetik retinopati ve prematüre retinopatisi gibi hastalıkların tarama ve uzun dönem takiplerinde kullanılabilecektir. Tarama testleri gelişmiş veya gelişmemiş ülkeler fark etmeksizin büyük finansal yüklere sebep olmaktadır. Derin öğrenme sistemleri ile birincil göz sağlığı gözetiminde kolaylık sağlayacağı şüphesizdir.

Retina fundus fotoğraflarında mikroanevrizma, hemoraji, eksuda, pamuk yün nokta bulguları otomatize sistemlere eğitilerek derin öğrenme konusunda çalışmalar yapılmıştır. Diyabetik retinopati tespiti için Gulshan ve ark. retina fotoğrafları ile derin öğrenme mimarisi üzerine çalışan ilk isimlerdir. İki tabanlı sistemdeki 9963 ve 1748 görüntüyle, orta şiddet veya kötü durum Diyabetik retinopati ve öngörülebilir diyabetik maküler ödem (DMÖ) üzerine çalışmışlardır. Daha sonradan Ting ve ark. Diyabetik retinopati, yaşa bağlı makula dejenerasyonu (A) ve glokom üzerine ilişkili göz hastalıkları üzerine derin öğrenme sistemleri kurmuşlardır. Kermany ve ark. DMÖ ve A için tarama programı hedefleyerek derin öğrenme modeli geliştirmişlerdir (Gulshan vd., 2016).

Tufail ve ark. otomotize olan Diyabetik retinopati görüntüleme değerlendirme sistemleri (ARIAS) ile insanların taramasının yerine geçmesini hedefleyen çalışma yapmışlardır. Diyabetik retinopati, öngörülen Diyabetik retinopati ve proliferatif Diyabetik retinopati duyarlılıklarını EyeArt için sırayla %94,7, %93,8 ve %99,6 olarak saptamış, Retmarker için %73, %85, ve %97,9 tespit etmişlerdir. Kabul edilebilir düzeylerde olan bu duyarlılık değerleri ile ARIAS modelinin Diyabetik retinopatiyi erken saptamada insan uygulamasının yerine geçebileceği bildirilmiştir. Bu Diyabetik retinopati tarama modelinin uzak sağlık hizmetleri bünyesinde kullanılabileceğini öngörmüşlerdir (Tufail vd., 2017).

Abramoff ve ark. yaptığı çalışmada öngörülebilir Diyabetik retinopatiyi saptamak için retina görüntülerinin analizine dayanarak 'Iowa Detection Program' üzerine çalışmışlardır. Bu sistemin hastalığı saptamada yüksek duyarlılık ve özgüllük değerlerine sahip olduğu ve bu modelin Diyabetik retinopatinin erken tanısında kullanılabileceği, görüşte azalmanın önüne geçilebileceği vurgulanmıştır. Yaptıkları çalışma, Nisan 2018'de yapay zekâ sistemi olarak IDxDR, FDA tarafından onaylanan ilk sistemdir (Abramoff vd., 2016).

Li ve ark. Optik koherens tomografi ile alınmış 207130 görüntü kullanarak VGG16 (Visual Geometry Group-VGG) mimarisi ile oluşturdukları sinir ağı modelinde, A ve DMÖ sınıflamasında %98,6 duyarlılık değerine ulaşan başarılı sonuçlar elde etmiştir. Bu sistemin, Diyabetik retinopati taramalarında kullanılabilecek yüksek duyarlılığa sahip güvenilir bir araç olduğunu bildirmişlerdir. Yapılan bu çalışmalar kayda değer sonuçlarla karşımıza çıksa da pratikte Diyabetik retinopati taramasında test konusunda eksiklikler mevcut olduğunu, farklı etnik yapılar, popülasyon bazlı derin öğrenme sistemlerinin genellenebilirlik sorunsalını ortaya çıkarttığını, görüntülemede kullanılan cihazların farklılığı da farklı bir sorun teşkil ettiğini belirtmişlerdir (Li, Liu, Zhang & Wu, 2019).

Ting ve ark. derin öğrenme modelini Singapur Diyabetik retinopati programında beş yıl test etmiş ve modeli geliştirmiştir. Bu çalışmada Hong Kong, Çin, Meksika, Amerika ve Avustralya olmak üzere çok merkezli çalışma yürütülmüştür. VGG19 modeliyle geliştirilmiş olan model sonuçlarına göre Diyabetik retinopatiyi tespit etmede %90,5 duyarlılık ve %91,6 özgüllük değerine ulaşmışlardır (Ting vd., 2017).

Poly ve ark. derin öğrenme modellerini değerlendiren 706922 fundus fotoğrafı içeren sekiz çalışmanın sonuçlarını birleştirerek meta analizi çalışması gerçekleştirmişlerdir ve Diyabetik retinopati saptamada %74 duyarlılık değeri ve %95 özgüllük değerine ulaşmışlardır (Poly vd., 2019).

Yaşa Bağlı Makula Dejenerasyonu (A), retinanın arka bölümünde makula denen bölgeyi tutan bir göz hastalığıdır. Makula renkli görmeyi sağlayan reseptörlerin en çok bulunduğu ve görmenin en iyi olduğu retina merkezidir. Yaşa Bağlı Makula Dejenerasyonu hastalığının erken evrelerinde retina üzerinde pigment değişiklikleri ve büyük drusen oluşumları meydana gelir. Hastalık geç evrelerde eksudatif olmayan 'kuru tip' ve eksudatif/neovasküler formu olan 'yaş tip' olarak görülmektedir. Eksudatif olmayan formu bu hastalığın %90'ını oluşturmakta olup coğrafik atrofilerle karakterize edilmiştir. Yaş tip formuna göre daha iyi prognozludur ve görme keskinliğindeki azalma daha yavaştır. Eksudatif tip olan neovasküler form ise bu hastalığın %10'unu oluşturur. Koroidde yeni damarlar oluşumu görülmektedir. Bu damarlardan makulaya sızıntı ve kanamalar olabilmektedir (Karaküçük & Eker, 2020).

Yaşa bağlı makula dejenerasyonu 60 yaş üzeri bireylerde görme azlığının ve körlüğün önde gelen bir sebebidir ve Dünya Sağlık Örgütünün raporuna göre %8,7 prevalansı vardır. Dünya Sağlık Örgütünün görme kusurları raporunda, katarakt ve glokomdan sonra %5 oranında görülen üçüncü körlük sebebi olarak karşımıza çıkmaktadır. 2014 yılında yapılmış sistematik meta analizi çalışması sonucunda, dünya çapında %8,4'ünde Yaşa Bağlı Makula Dejenerasyonu saptanmıştır. 2012 yılında 170 milyon, 2020 yılında 198 milyon, 2040 yılında 288 milyon kişinin bu hastalıktan etkilenmesi beklenmektedir. A prevalansının artıyor olması ve yaşlanan dünya nüfusu A hastalığının önemini giderek artırmaktadır (Wong vd., 2014).

Burlina ve ark. renkli fundus fotoğrafları kullanarak Yaşa Bağlı Makula Dejenerasyonu tarama ve saptamada kullanılan CNN modelini geliştirmişlerdir. Sonuçların sağlık profesyonellerinin değerlendirilmeleriyle uyumlu olduğu görülmüştür (Burlina vd., 2016). Grassmann ve ark. ise fundus fotoğrafları kullanarak Yaşa Bağlı Makula Dejenerasyonu derecelendirmesi yapan bir derin öğrenme mimarisi oluşturmuşlardır (Grassmann, 2018).

Peng ve ark. fundus fotoğrafları kullanarak DeepSeeNet derin öğrenme modeliyle Yaşa Bağlı Makula Dejenerasyonu hastalık sınıfının otomatize sınıflamasını yapmış ve sonuçlarını retina uzmanlarının sonuçlarıyla kıyaslamışlardır. DeepSeeNet modelinin, büyük druzen ve pigment anomalilerini saptamada retina uzmanlarına göre daha iyi sonuçlar verdiği görülürken, geç Yaşa Bağlı Makula Dejenerasyonu tespit etmede daha kötü sonuçlar çıkardığı bildirilmiştir. Bu algoritmanın, erken ve geç Yaşa Bağlı Makula Dejenerasyonu gelişimi için risk öngörmede yardımcı olacağı vurgulanmıştır (Peng vd., 2019).

Glokom, optik sinirin progresif hasarı ile seyreden görme kaybına sebep olan ve dünyada körlük nedenleri arasında ikinci sırada yer alan oküler hastalıktır. Glokomun erken tanısı ve altta yatan sebebe yönelik erken müdahale yapılması bu olumsuz sonuçların önüne geçecektir (Karaküçük & Eker, 2020).

Li ve ark. glokom tanısı koyma üzerine LabelMe adlı CNN modeli geliştirmişlerdir. Yüksek duyarlılık ve özgüllük değerlerine sahip olsalar da mevcut Diyabetik Retinopati, Yaşa Bağlı Makula Dejenerasyonu ve yüksek miyopi gibi patolojik durumların sonuçlarını etkilediklerini belirtmişlerdir (Li vd., 2018).

Asaoka ve ark. görme alanı testini baz alarak açık açılı glokom için derin öğrenme modeli üzerinde çalışmışlardır (Asaoka, Murata, Iwase & Araie, 2016).

Niemeijer ve ark. tarafından 2004 yılında Diyabetik Retinopati tarama programı yapmak amacıyla fundus fotoğrafları ile retinal damarlarda segmentasyon yapabilen DRIVE (Digital Retinal Images for Vessel Extraction) veri tabanı kurulmuştur (Niemeijer, Staal, Van-Ginneken, Loog & Abramoff). Benzer şekilde Hoover ve ark. STARE (Structured Analysis of the Retina) veri tabanı üzerinde çalışmışlardır. Bu görüntü temelli veri tabanları, farklı CNN modelleri üzerine çalışan bilim insanlarına kaynak oluşturmuştur (Hoover, 1975).

## 1.5. Problemin Belirlenmesi ve Çalışma Hipotezi

Göz hastalıklarının fundus fotoğrafı ile değerlendirilmesi oftalmoloji alanını derin öğrenme mimarilerinden faydalanmak için mükemmel bir fırsat haline getirmiştir. Literatür taraması sonucunda, oftalmoloji alanında hastalıkları sınıflandırmada derin konvolüsyonel sinir ağı mimarilerinin oluşturulduğu, bu mimarilerin performanslarının değerlendirildiği görülmektedir.

Hastalıkların erken evresinde az sayıda semptom görülebildiği için yaygın oküler hastalıkların erken teşhisi oldukça zordur. Daha uzun süreli diyabeti olan kişilerin, Diyabetik Retinopati (D) ve Diyabetik Makula Ödemi (DMÖ) geliştirme olasılığı daha yüksektir. Diyabetik göz hastalığının erken belirtileri, küçük ve tespit edilmesi zor olan mikroanevrizmalardır. Makula merkezi görüşten sorumludur. Sıvı birikimli Makula görüşü bozacaktır. Yaşlı insanlarda makula dejenerasyonunun erken evrede hiçbir semptomu yoktur. Katarakt, yaşlılarda görülmektedir ve insanın görüşünü azaltmaktadır. Glokom, geri dönüşü olmayan optik sinir başlığına zarar veren bir grup hastalığı ifade etmektedir. Ayrıca, hipertansiyon, kan damarlarının morfolojik yapılarını, örneğin çap değişikliklerini etkilemekte ve inme, kalp krizi gibi kalp-beyin damar hastalıkları oluşturabilmektedir. Progresif retinal pigment epitel incelmesi ve zayıflaması olan miyop insanlarda görme kaybı da yüksek risk altındadır. Bu hastalıkların erken teşhisi görme hasarını ve eşlik edebilecek hastalıkları önleyebilir.

ODIR-2019 (Ocular Disease Intelligent Recognition), Pekin Üniversitesi (PKU), Shanggong Medical Technology Co. Ltd. (SG) ve Sağlık ve Tıpta Veri Bilimi Merkezi (CDSHM) ile birlikte düzenlenen Oküler Hastalık Akıllı Tanıma Yarışması, uluslararası katılım için başvuruya açılan ilk oküler hastalık sınıflandırma yarışmasıdır.

Shanggong Medical Technology Co. Ltd. (SG), yarışma katılımcılarına hastanın yaşı, cinsiyeti, her iki göz için kullanılan renkli fundus (binocular color fundus) görüntüleri ve doktorların teşhis raporlarından oluşmak üzere 5.000 hastaya ait oftalmolojik görüntü seti sağlamıştır. Bu veri setinde Oftalmoloji alanında sıklıkla karşılaşılan Diyabetik Retinopati, Glokom, Katarakt, Yaşa bağlı Makula Dejenerasyonu, Hipertansiyon, Miyop, Diğer Anomaliler olmak üzere yedi farklı hastalık sınıfı ve sağlıklı bireylere ait fundus görüntüleri olmak üzere toplam sekiz kategoriden oluşan fundus görüntüleri yer almaktadır.

Bu veri seti Çin, Pekin şehri kooperatif hastanelerinde ve sağlık kurumlarında oküler hastalık sınıflandırması gerçekleştirmek için gerçek klinik ortama sahip oküler sağlık muayenesi yapılan hastalardan elde edilmiştir.

Oftalmolojide fundus taraması, diyabetik retinopati, glokom, katarakt, yaşa bağlı makula dejenerasyonu ve diğer birçok nedenin sebep olduğu körlüğü mümkün olduğunca erken zamanda önlemenin ekonomik ve etkili bir yoludur. Bu bağlamda, oküler hastalıkların sınıflandırılması için klasik konvolüsyonel sinir ağı mimarilerinden olan Inceptionv3, VGG16, ResNet50 ön eğitimli model mimarilerinin kullanılabilirliğini araştırmak, bu derin konvolüsyonel sinir ağı modeli mimarilerinin sınıflandırma performanslarını değerlendirmek bu tez çalışmasının problemi olarak tanımlanmıştır. Böylelikle çalışmanın ana hipotezini, oküler hastalıkların sınıflandırılması için oluşturulan Inceptionv3, VGG16 ve ResNet50 CNN mimarilerinin hastalık sınıflandırma performans ölçütlerinin karşılaştırılması oluşturmaktadır.

## 1.6. Tezin Amacı

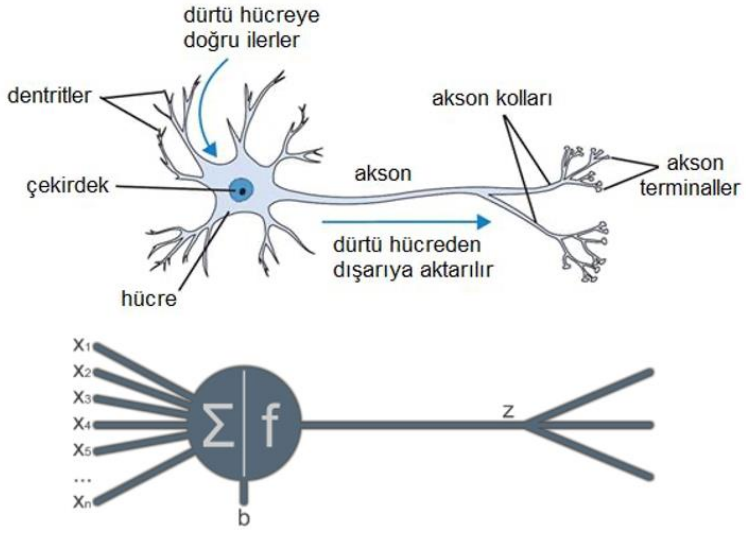
Bu tez çalışmasının temel amacı, yapay zekâ odaklı oküler hastalıkların sınıflandırma probleminin çözümlenmesini sağlamaktır. Bu kapsamda, hastaların sağ ve sol gözlerinden alınan renkli fundus görüntüleri ve her bir görüntüye ait hastalık teşhis anahtar kelimeleri olan eğitim verisi kullanılarak oküler hastalık sınıflandırması için farklı derin konvolüsyonel sinir ağı modelleri oluşturmak, bu modellerin eğitimini gerçekleştirmek, model eğitimi sonrası, test fundus görüntü kümesi kullanılarak oluşturulan modellerin hastalık sınıflandırma performans ölçütleri olan kappa değeri, F1 skoru, Eğri altında kalan alan (Area Under the Curve-AUC) ve bu üç ölçütün ortalaması olan Final skoru değerlerinin hesaplanması, modellerin sınıflandırma performansının bu ölçütlere göre değerlendirilmesi, en iyi skora sahip modelin oküler hastalıkların sınıflandırılmasında kullanılması ve literatüre önerilmesi amaçlanmıştır.

# 2. GENEL BİLGİLER

## 2.1. Derin Öğrenmenin Matematiksel Temelleri

### 2.1.1. *Yapay sinir ağları*

Yapay sinir ağları, insan beyninden ilham alınarak geliştirilmiştir. İnsan beyninin temel işlem elemanı ve sinir sisteminin en basit elemanı olan nöron ve bu nöronlar arası bağlantılara şekilsel ve işlevsel olarak benzeyen bir yapay sinir ağı, biyolojik sinir sisteminin basit bir simülasyonudur. Biyolojik sinir sisteminin matematiksel bir modeli olarak da tanımlanabilecek olan yapay sinir ağı, birbirleri ile bağlantılı yapay sinir hücrelerinin oluşturduğu bir sistem ile biyolojik sinir sisteminin bilgiyi depolama, kullanma ve işleme yeteneklerini taklit etmeyi ve insan gibi karar verebilen ve muhakeme yeteneği olan zeki sistemler elde etmeyi amaçlar Şekil 2.1'de bir biyolojik nöron ile yapay nöron karşılaştırması gösterilmiştir (Kinsley & Kukiela, 2020).



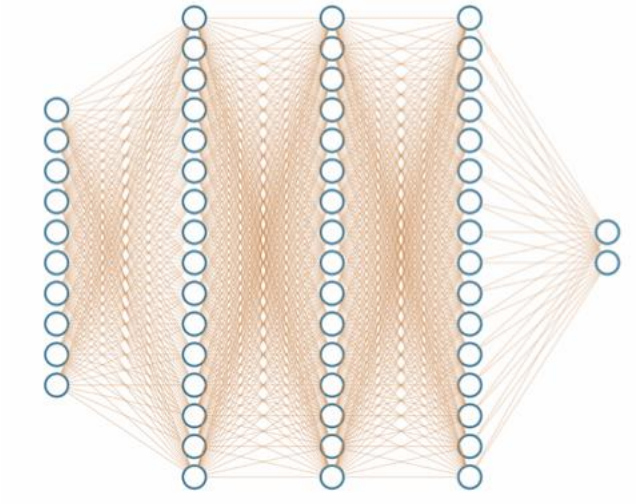

**Şekil 2.1.** Bir biyolojik nöron ile yapay nöron karşılaştırması (Kızrak, 2018; Kinsley & Kukiela, 2020)

Tek bir nöronun kendi başına göreceli olarak faydası yoktur, ancak yüzlerce veya binlerce (veya daha fazla) başka nöronla birleştiğinde, diğer makine öğrenmesi yöntemlerinden daha iyi performans gösteren ilişkiler ve sonuçlar üretir. Bir yapay sinir ağının temel yapısında; nöronlar, nöronlardan oluşan katmanlar, nöronlar arasında bağlantı kuran ağırlık değerleri, bağlantı ağırlıklarının belirlenmesini sağlayan öğrenme algoritmaları ve transfer (aktivasyon) fonksiyonu bulunmaktadır (Kinsley & Kukiela, 2020).

Yapay sinir ağında katmanlar girdi katmanı, çıktı katmanı ve bu ikisinin arasında bulunan gizli katmanlardır. Her bir katman nöronlardan oluşmaktadır. Girdi ve çıktı katmanındaki nöron sayısı, bağımsız ve bağımlı değişkenlerinin sayısı ile belirlenmekte iken, gizli katmandaki katman sayısı ve her bir gizli katmanda bulunacak nöron sayıları, en iyi performansı verecek şekilde kullanıcı tarafından belirlenmektedir. Girdi katmanı verilerin yapay sinir ağına alınmasını, çıktı katmanı ise ağ içerisinde işlenen bilginin sonucunu verir. Bir yapay sinir ağında gizli katman olması gerekmediği gibi, birden fazla gizli katman da bulunabilir. Girdi katmanı, ağı adlandırırken katman sayısına dâhil edilmemektedir. Bundan dolayı, girdi katmanı ile birlikte dört katmandan oluşan bir yapay sinir ağı üç katmanlı bir ağ olarak adlandırılır. (Fausett, 2006).

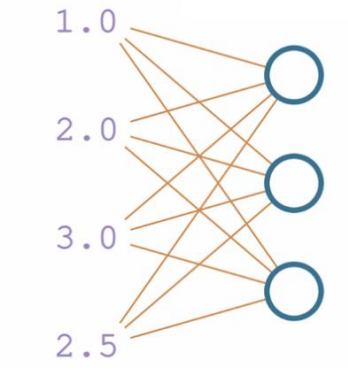
Nöronlar arasındaki her bağlantının kendisiyle ilişkili bir ağırlığı vardır. Bu ağırlık, girdinin ne kadarının kullanılacağının eğitilebilir bir faktörüdür. Tüm girdi ve ağırlık değerleri çarpımı nöronumuza aktığında, bu değerler toplanır ve başka bir eğitilebilir parametre olan bir bias değeri eklenir (Çıktı = (Girdi x Ağırlık) + bias). Bias değerinin amacı, çıktıyı olumlu ya da olumsuz olarak dengelemektir, bu da daha gerçek dünya dinamik veri türlerini değerlendirmemize yardımcı olur. Ağırlıklar ve bias kavramı, modelimizi verilere uydurmak için ayarlayabileceğimiz "düğümler" olarak düşünülebilir. Bias ve ağırlıkların her ikisi de ayarlanabilir parametrelerdir ve her ikisi de nöronların çıktılarını etkiler ancak bunu farklı şekilde yaparlar. Ağırlıklar çarpıldığı için, yalnızca çıktının büyüklüğünü, fonksiyonun eğimini değiştirecektir ve hatta işareti pozitiften negatife veya tam tersine çevirecektir. Bias ise genel olarak fonksiyonu dengeleyecektir. Örneğin, bias arttıkça, fonksiyon çıktısı genel olarak pozitif yöne doğru hareket edecektir.

Katman büyüklükleri katmandaki nöron sayıları ile ifade edilmektedir. Şekil 2.2'deki sinir ağına ait katman büyüklüğü (10,16,16,16,2) olarak belirtilir. Bu sinir ağına ait ağırlık değerlerinin sayısı, giriş katmanı 10 nöron ile ilk gizli katman 16 nöron arasında 10*16=160, ilk gizli katman ile ikinci gizli katman arasında 16*16=256, ikinci gizli katman ile üçüncü gizli katman arasında 16*16=256, son gizli katman ile çıkış katmanı arasında 16*2=32 bağlantıların sayısı olmak üzere toplam hesaplanan ağırlık değeri 704, giriş katmanı haricinde her bir katmandaki nöronlara bias değeri eklendiği göz önüne alındığında, 16+16+16+2=50 bias değeri olmak üzere toplam 754 öğrenilebilir parametre hesabı gerçekleştirilmektedir (Kinsley & Kukiela, 2020).



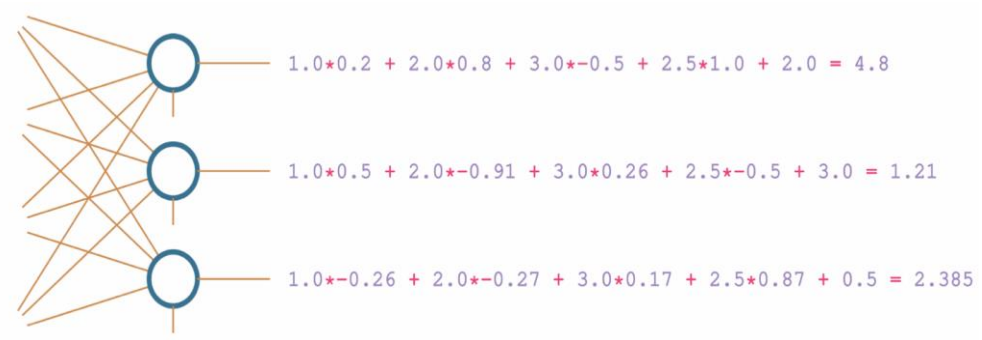
**Şekil 2.2.** Her biri 16 nörondan oluşan 3 gizli katmana sahip sinir ağı örneği (Kinsley & Kukiela, 2020)

Sinir ağları tipik olarak birden fazla nörondan oluşan katmanlara sahiptir. Bir katmandaki her nöron tam olarak katmana verilen (bu eğitim verisi veya önceki katmandan çıktı olabilir) girdiyi alır. Her nöron kendi ağırlık kümesini ve kendine ait biası içerir ve kendi eşsiz çıktısını üretir. Katmanın çıktısı, her bir nöron için bir tane olmak üzere bu çıktıların her birinin bir kümesidir. Örneğin, girdi [1,2,3,2.5] olmak üzere 4 girdiden oluşan ve bir katmanda 3 nöronun olduğu bir senaryo olsun. Her bir nörona ait ağırlık ve bias değerleri sırası ile ağırlık1= [0.2, 0.8, -0.5, 1], bias1 = 2; ağırlık2 = [0.5, -0.91, 0.26, -0.5], bias2 = 3; ağırlık3 = [-0.26, -0.27, 0.17, 0.87], bias3 = 0.5 olsun. Şekil 2.3'de 4 girdi ve bir katmanda 3 nörondan oluşan örnek senaryo gösterilmiştir.




**Şekil 2.3.** Dört girdi ve bir katmanda 3 nörondan oluşan örnek senaryo (Kinsley & Kukiela, 2020)

Şekil 2.4'de bir nöron katmanının arkasındaki matematik ve görselleştirilmesi verilmiştir. Bu senaryoda üç nöronu tanımlayan üç ağırlık değeri ve üç bias değeri vardır. Katmanın çıktısı, [4.8, 1.21, 2.385] olmak üzere 3 değerden oluşan bir liste olacaktır. Her nöron aynı girdilere bağlanır. Aradaki fark, her bir nöronun girdiye uyguladığı farklı ağırlık ve bias değeridir. Mevcut katmandaki her nöronun önceki katmandaki her nöronla bağlantısı vardır. Bu sinir ağı yapısına tam bağlantı (Fully Connected-FC) katmanlı sinir ağı adı verilir. Bu yapı çok yaygın bir sinir ağı türüdür. Ancak her yapay sinir ağı modeli bu yapı gibi tam bağlı katmanlardan oluşmamaktadır.




**Şekil 2.4.** Bir nöron katmanının arkasındaki matematik ve görselleştirilmesi (Kinsley & Kukiela, 2020)

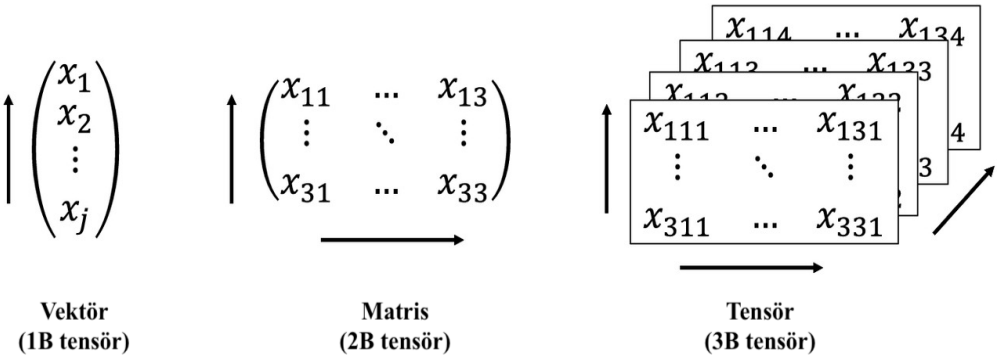
Çok daha fazla katmanlı ve daha fazla nöronu olan bir ağ olduğunu düşünelim. Bu durumda mevcut yöntemleri kullanarak hesaplamak oldukça zorlaşacaktır. Bunun yerine, dinamik boyutlu girdileri ve katmanları ölçeklemek ve işlemek için bir döngü kullanılabilir. Farklı ağırlık değişkenleri, üzerinde yineleme yapabilmek için ağırlık listesine dönüştürülür ve döngüler kullanılır.

Döngüler kullanan bilgisayar programlama dili komutları ile, katmandaki girdi veya nöron sayısını değiştirebilir ve döngü bu işlemleri kolaylıkla hesaplayabilir hale gelmektedir. Bu noktada Python programlama dili tek başına matris/tensör/dizi matematiksel işlemlerini verimli bir şekilde gerçekleştirmez. NumPy kütüphanesi sayesinde bu matematiksel işlemler kolaylıkla gerçekleştirilebilir. Öncelikle, Python'daki en popüler derin öğrenme kütüphanesinin "TensorFlow" olarak adlandırılmasının temel nedeni, tamamen tensörler üzerinde işlem yapmasıdır (Kinsley & Kukiela, 2020).

### 2.1.2. *Tensörler, diziler, vektörler*

Tensörler, dizilerle (arrays) yakından ilişkilidir. Derin öğrenme uygulamalarında tensör/dizi/matris temel kavramları karıştırılmaktadır. Bu kavramlar arasında önemli farklılıklar bulunmaktadır.

Tensör, çok boyutlu verinin simgelenebildiği geometrik bir nesnedir. Sadece bir eleman taşıyan tensörlere skaler (skaler tensör, 0 boyutlu tensör veya 0B tensör) denir. Skaler aynı zamanda yönsüz nicel büyüklüklerdir. Birden fazla elemanı olan skalerlerden oluşan sayı dizilerine ise vektör ya da 1B tensör denir. 1B bir tensör bir eksene sahiptir. Bir vektör dizisi matris veya 2B tensördür. Bir matrisin genellikle satır ve sütun olarak adlandırılan iki ekseni vardır. Tensör, tüm bu nesnelerin genelleştirilmiş halidir ve çok boyutlu veri kümeleri için kullanılmaktadır. Nesnenin kaç boyut ile ifade edildiğine tensörün derecesi denir. Bir skalerin derecesi sıfır, bir vektörün derecesi bir, bir matrisin derecesi ise ikidir. Tensörler üç ve üzeri dereceye sahip olabilir. Vektör, matris ve tensörün matematiksel gösterimi Şekil 2.5'de verilmiştir (Kolecki, 2002).




**Şekil 2.5.** Vektör, matris ve tensör matematiksel gösterimi (Brownlee, 2018)

Görüntülerin yükseklik, genişlik ve kanal sayısı olmak üzere üç boyutu vardır. Örneğin, genişliği ve yüksekliği 256x256 boyutlarında, Kırmızı, Yeşil ve Mavi (RGB) olmak üzere 3 renk derinliğine sahip kanal sayısı, 1000 görüntüden oluşan tensörün ifadesi (1000, 256, 256, 3) olacaktır.

### 2.1.3. *İç çarpım ve vektör toplama işlemleri*

Girdi ve ağırlık vektörlerimizdeki her bir elementi element bazında çarpma işlemini iç çarpım (dot product) kullanarak Eşitlik 2.1'deki gibi elde edebiliriz. İki vektörün iç çarpımı, ardışık vektör elemanlarının çarpımlarının toplamıdır. Her iki vektör de aynı boyutta olmalıdır (eşit sayıda öğeye sahip olmalıdır).

$$\vec{a}.\vec{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \tag{2.1}$$

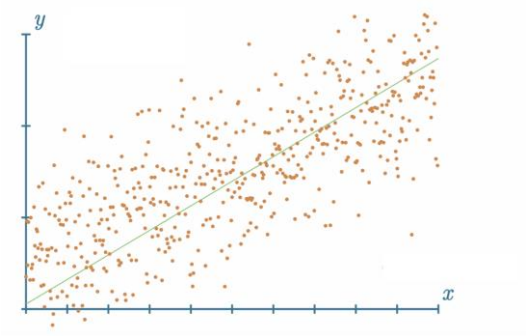
*a* vektörünü girdiler ve *b* vektörünü "ağırlıklar" olarak adlandırdığımızda, bu iç çarpım işlemi, ihtiyacımız olan işlemleri gerçekleştirmenin kısa ve öz bir yoludur. Bu sonuca bias değerini eklemek için ise Eşitlik 2.2'deki gibi iki vektörün toplanması işlemine ihtiyaç duyulur. Bu işlem, her iki vektörün de aynı boyutta olmasını gerektirir ve element bazında gerçekleştirilen bir işlemdir (Kinsley & Kukiela, 2020).

$$\vec{a} + \vec{b} = [a_1 + b_1, a_2 + b_2, \cdots, a_n + b_n] \tag{2.2}$$

### 2.1.4. *Toplu veri (A Batch of Data)*

Yapay sinir ağları, model eğitimi aşamasında verileri toplu olarak alma eğilimindedir. Şu ana kadar örnek girdi verileri, özellik kümesi adı verilen çeşitli özelliklerin yalnızca bir örneği (veya gözlemi) olmuştur. Örneğin girdinin [1, 2, 3, 2.5] verilerinden oluştuğunu düşünelim. Burada [1, 2, 3, 2.5] verileri bir şekilde anlamlı ve arzu edilen çıktı için açıklayıcıdır. Her bir değeri, farklı bir sensörden elde edilen bir değer olarak düşünelim. Bu değerlerin her biri, bir özellik gözlem verisidir ve aynı zamanda bir gözlem veya en yaygın olarak örnek olarak adlandırılan bu özellikler birlikte, bir özellik seti örneği oluştururlar. Genellikle, paralel işlemede gruplar halinde eğitimin daha hızlı olması ve grupların eğitim sırasında genellemeye yardımcı olması sayesinde yapay sinir ağları aynı anda birçok örnek alır.

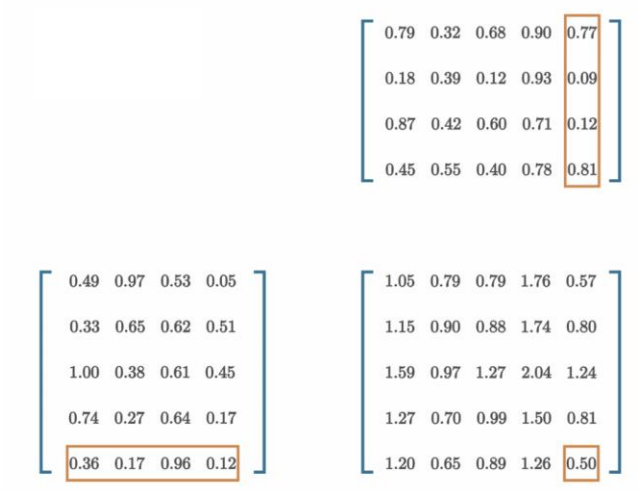
Bir eğitim sürecinin bir adımını gerçekleştirirken her seferinde tek bir örneklem alınırsa tüm veri kümesine uyan ağırlıkları ve biası yavaşça genel olarak güncellemek yerine, büyük olasılıkla model alınan tek örneğe uymaya devam eder. Gruplar halinde eğitim, ağırlık ve bias değerlerinde daha anlamlı değişiklikler yapma şansını artırır. Tek seferde bir örnek yerine gruplar halinde örneklem parçalarının alınması kavramı için Şekil 2.6'da seçilen gözlemlerden oluşan doğrusal denklem örneği gösterilmiştir.




**Şekil 2.6.** Seçilen gözlemlerden oluşan doğrusal model uyumu (Kinsley & Kukiela, 2020)

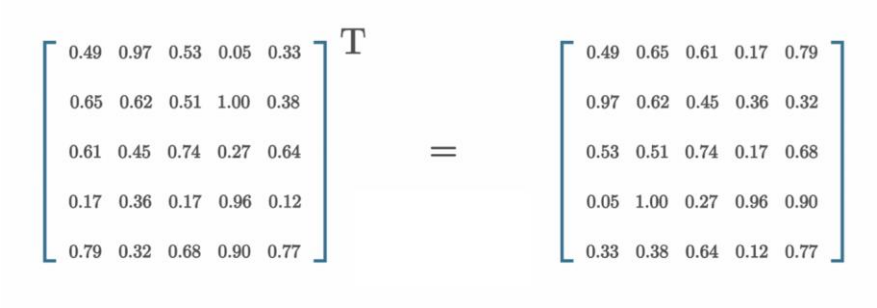
### 2.1.5. *Matris çarpımı*

Matris çarpımı, 2 matrise sahip olduğumuzda gerçekleştirilen bir işlemdir ve ilk matristeki tüm satır kombinasyonlarının, 2. matrisin sütunlarının iç çarpımlarını gerçekleştirerek bu iç çarpımların bir matrisi Şekil 2.7'deki gibi elde edilir.




**Şekil 2.7.** Matris çarpımı gösterimi (Kinsley & Kukiela, 2020)

Bir matris çarpımı gerçekleştirmek için, birinci matrisin sütun boyutu ile ikinci matrisin satır boyutu ile aynı olmalıdır. Matris çarpımı işleminde bu kural sağlanması için genellikle matris devriği alma işlemi uygulanır. Devrik alma işlemi, bir matrisin, satırları sütun ve sütunları satır olacak şekilde değiştirilerek uygulanır. Şekil 2.7'de yer alan ilk matrisin devriği alınarak Şekil 2.8 gösterimi elde edilmiştir.




**Şekil 2.8.** Bir matrisin devriği (Kinsley & Kukiela, 2020)

### 2.1.6. *Aktivasyon fonksiyonları*

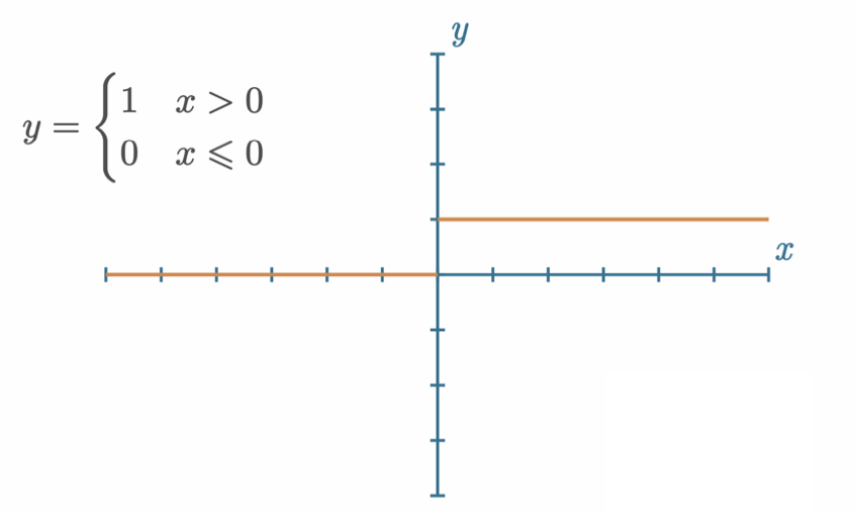
Bir sinir ağındaki nöron, *n* sayıda girdi alan ve tek bir çıktı üreten temel hesaplama birimidir. Bir yapay nöron *x* girdilerin, *w* ağırlıkları ile çarpımları toplamını hesaplar ve bias değeri *b* ekleyerek Eşitlik 2.3'deki gibi bir *y* çıktısı üretir:

$$y = f\left(\sum_{i=1}^{n} w_i . x_i + b\right) \tag{2.3}$$

Aktivasyon fonksiyonu burada $y$ çıktı değerini kontrol etmek için yani bir nöronun aktif olup olmayacağına karar vermek için kullanılmaktadır. Aktivasyon fonksiyonları, bu yönüyle derin sinir ağları için önemli bir özelliktir (Chollet, 2017).

En basit aktivasyon fonksiyonu adım (step) fonksiyonu Şekil 2.9'da gösterilmektedir. Tek bir nöronda, $(w.x + b) > 0$ ise nöron aktif olacak ve 1 değeri üretecek, aksi durumda 0 çıktısı verecektir. İkili değer alan bir fonksiyondur ve Genellikle çıkış katmanlarında ikili sınıflandırıcı olarak kullanılır. Gizli katmanlarda öğrenme değeri temsil etmediği ve türevlenebilir bir yapısı olmadığı için kullanılmamaktadır.

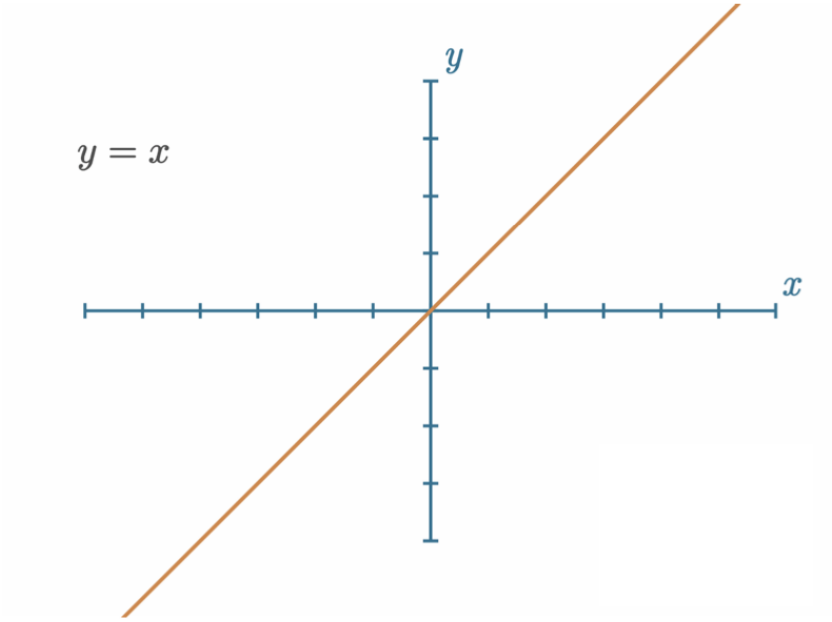
Adım fonksiyonunun günümüzde tercih edilmemesinin sebebi, modelin eğitimi ve ağ optimizasyonu sürecinde, her bir nörondan aktarılan ağırlıkların ve bias'ın ağın çıktısı üzerindeki bireysel etkilerini değerlendirme aşamasında etkisinin büyük olduğu göz önüne alındığında adım fonksiyonu ile ilgili sorun, bu etkilerin ne olduğu hakkında çok az bilgi vermesidir. Çünkü bu fonksiyondan toplanan çok az bilgi vardır (0 ya da 1).




**Şekil 2.9.** Adım (step) aktivasyon fonksiyonu (Kinsley & Kukiela, 2020)

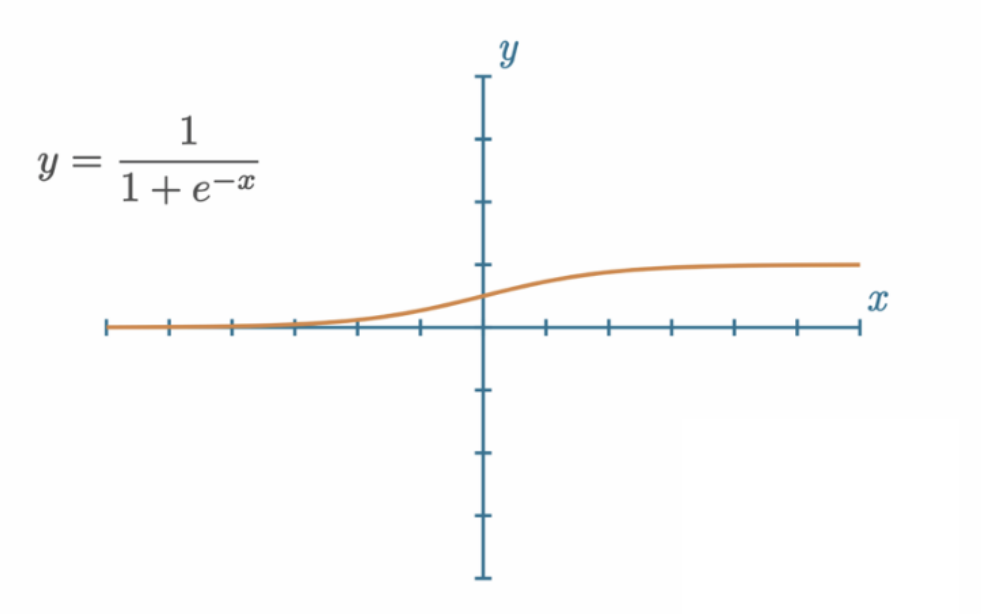
Şekil 2.10'da doğrusal aktivasyon fonksiyonu gösterilmiştir. Basitçe bir doğru denklemini ifade etmektedir. Burada $y = x$ ve çıktı değeri girdiye eşittir. Bu aktivasyon fonksiyonu genellikle bir doğrusal regresyon modeli için son katmanın çıktısına uygulanmaktadır.

Doğrusal aktivasyon fonksiyonun türevinin sabit olması, modelin eğitimi sürecinde her bir nörondaki ağırlık ve bias değerleri geriye yayılım (backpropagation) algoritması ile güncelleme ve böylece öğrenme işlemi gerçekleştirilmektedir. Bu algoritmanın temeli türev alan bir sistemden oluşmaktadır. $y = c.x$, fonksiyonunun $x$'e göre türevi alındığında $c$ sonucuna erişiriz. Bu $x$ girdi değişkenleri ile bir ilişkinin kalmadığı anlamına gelmektedir. Türevi sürekli olarak sabit bir değer çıkan bir yapıda öğrenme ve ağırlık güncelleme işlemi gerçekleştirilemez. Ayrıca, tüm katmanlarda doğrusal aktivasyon fonksiyonu kullanıldığında giriş katmanı ile çıkış katmanı arasında hep aynı doğrusal sonuca ulaşılır. Doğrusal fonksiyonların doğrusal bir şekilde birleşimi yine bir başka doğrusal fonksiyondur. Bu durum da gizli katmanların işlevsiz kaldığı sonucunu doğurmaktadır (Kinsley & Kukiela, 2020).




**Şekil 2.10.** Doğrusal aktivasyon fonksiyonu (Kinsley & Kukiela, 2020)

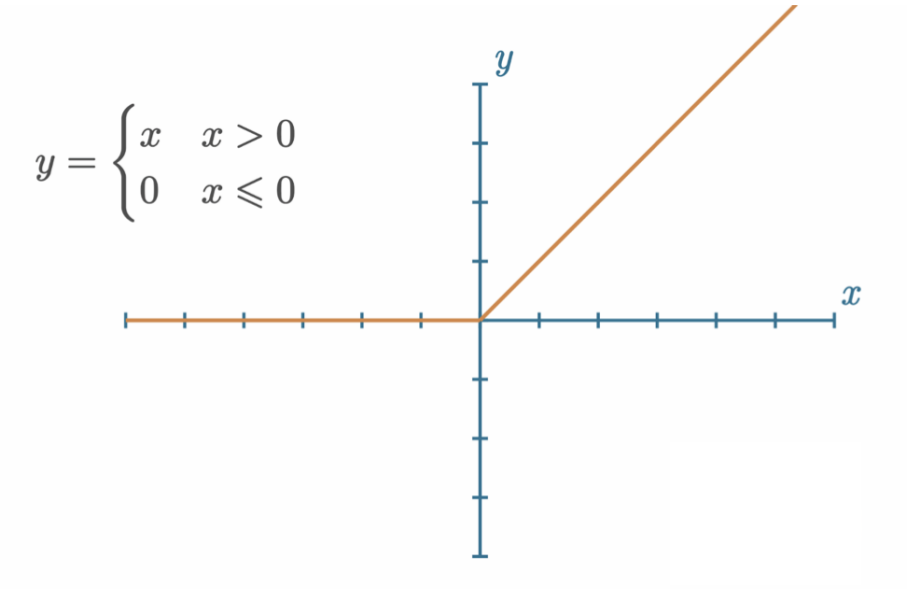
Yapay sinir ağları için kullanılan orijinal, daha ayrıntılı, Şekil 2.11'de gösterilen aktivasyon fonksiyonu Sigmoid aktivasyon fonksiyonudur. Sigmoid aktivasyon fonksiyonu, girdi $x$ değerlerinin $(-\infty, +\infty)$ değer aralığına karşılık her zaman (0,1) aralığında çıktı değeri üretmektedir. Adım fonksiyonundan farklı olarak fonksiyon türevlenebilirdir. Model eğitimi ve optimizasyonu aşamasında ağırlıklar güncellenebilir ve öğrenme işlemi gerçekleştirilebilir. Sigmoid fonksiyonu en sık kullanılan aktivasyon fonksiyonu olmakla birlikte birçok başka ve daha verimli alternatifleri de vardır. Şekil 2.11 incelendiğinde, $y$ değerleri $x$'deki değişikliğe çok az tepki vermektedir. Bu problem bu noktalarda türev değerlerinin düşük olmasına ve 0'a yakınsamasına neden olur. Bu duruma gradyanların kaybolması (vanishing gradient) denir ve öğrenme durumu bu noktalarda minimum düzeyde gerçekleşir. Hatayı minimize eden optimizasyon algoritması yerel minimum değerine takılabilir ve yapay sinir ağı modelinden alınabilecek maksimum performans elde edilemeyebilir.




**Şekil 2.11.** Sigmoid aktivasyon fonksiyonu (Kinsley & Kukiela, 2020)

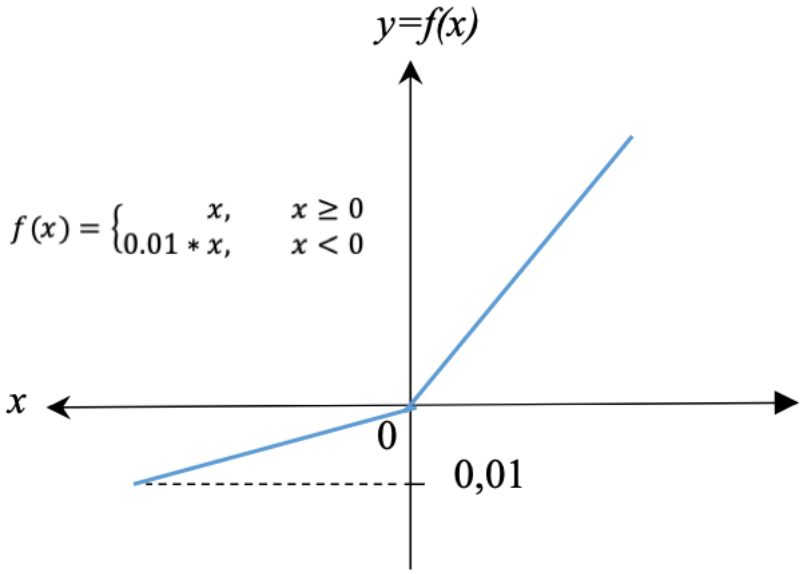
Düzleştirilmiş doğrusal birim aktivasyon fonksiyonu (Rectified Linear Unit-ReLU) Şekil 2.12'de gösterilmiştir. Sigmoid fonksiyonunun daha basitleştirilmiş bir yapısıdır. $x \leq 0$ ise, $y = 0$ olup, aksi durumda $y = x$'dir. İlk bakışta pozitif eksende doğrusal fonksiyon ile aynı özelliklere sahip olduğu görülmektedir. ReLU [0, +∞) aralığında değer almaktadır.

Çok fazla nörondan oluşan çok katmanlı büyük bir sinir ağı hayal ettiğimizde sigmoid aktivasyon fonksiyonu neredeyse tüm nöronların aynı şekilde aktifleşmesine sebep olur. Bu durum aktivasyon fonksiyonunun yoğun işlem gerektirdiği anlamına gelmektedir. Ağdaki bazı nöronların aktif olup aktivasyonun verimli bir hesaplama yükü ile gerçekleştirilmesi istenir. ReLU ile bu durum sağlanmaktadır. Aynı zamanda negatif eksen boyunca 0 değerlerini alması ağın daha hızlı çalışacağı anlamına gelmektedir. Hesaplama yükünün sigmoid fonksiyonuna göre az olması çok katmanlı sinir ağlarında bu fonksiyonun daha çok tercih edilmesini sağlamıştır (Zhang vd., 2017).




**Şekil 2.12.** Düzleştirilmiş doğrusal aktivasyon fonksiyonu (ReLU) (Kinsley & Kukiela, 2020)

Kendisine gelen $x$ değerleri pozitif ise x değerini, negatif ise ReLU'dan farklı olarak $0.01 * x$ çıkış değerini üreten aktivasyon fonksiyonu Leaky ReLU fonksiyonudur. Negatif değerleri 0.01 değeri çarpması sayesinde veri kaybı önlenerek öğrenmenin daha başarılı bir şekilde gerçekleşmiş olması beklenmektedir (Zhang vd, 2017). Bu fonksiyonun grafiği Şekil 2.13'de gösterilmektedir.




**Şekil 2.13.** Leaky ReLU aktivasyon fonksiyonu (Kinsley & Kukiela, 2020)

Softmax aktivasyon fonksiyonu, bir sinir ağının çıkış değerlerini normalize ederek bu değerleri olasılık değerlerine dönüştürür. Sınıflandırma problemlerinde kullanılan aktivasyon fonksiyonudur.

$$S_{i,j} = \frac{e^{z_{i,j}}}{\sum_{l=1}^{L} e^{z_{i,l}}} \tag{2.4}$$

Eşitlik 2.4'de, $S_{i,j}$ $j$-inci Softmax'ın $i$-inci örneğinin çıktısını, $z$ - girdi vektörlerinin bir listesi olan girdi dizisini (önceki katmandan çıktı vektörleri), $z_{i,j}$ $j$-inci Softmax'ın $i$ -inci örnek girdisi, $L$ - girdi sayısını, $z_{i,l}$- $l$-inci Softmax'ın $i$-inci örnek girişini ifade etmektedir. Derin sinir ağı çıktısı, girdinin hangi sınıfa ait olduğuna dair olasılıksal bir tahmin sonucu sunar. Softmax aktivasyon fonksiyonu tarafından hesaplanan bu dağılım, her sınıf için olasılık değerlerini temsil eder ve sınıf olasılık değerleri toplamı 1'dir. Tahmin edilen sınıf, en büyük olasılık değerini veren çıktı nöronuyla ilişkilidir. Girdinin belirli sınıfa ait olma olasılığını 0–1 aralığında değerler üreterek belirlenmesini sağlamaktadır. Bu sayede olasılıksal yorumlamaya imkân sağlar (Goodfellow, Bengio & Courville, 2016).

### 2.1.7. *Türev ve gerekliliği*

Rasgele ağırlık ve bias değerleri veya farklı başlatma teknikleri kullanarak farklı yaklaşımlar ile başlatılmış bir derin sinir ağı modeli ile amaç, modeli zaman içinde eğitmek ve ağın optimizasyonunu gerçekleştirmektir. Bir modeli eğitmek için, modelin doğruluğunu ve güvenini artırmak için ağırlıkları ve bias değerlerini güncelleme işlemi yapılır. Bunu yapmak için, modelin ne kadar hatası olduğu hesaplanır. Maliyet fonksiyonu olarak da adlandırılan kayıp fonksiyonu bir modelin ne kadar yanlış olduğunu ölçen algoritmadır. Kayıp, bu metriğin ölçüsüdür. Kayıp modelin hatası olduğundan, ideal olarak 0 olmasını isteriz. Kayıp fonksiyonu, gerçek değerler ile tahmin değerleri arasındaki farkı ifade eder. Bu fonksiyon ağırlık ve bias içermez. Bu fonksiyonun girdisi modelin çıktısıdır.

Nöronların ağırlıkları ve bias değerleri model çıktısını etkiler. Dolayısıyla, ağırlıkları, bias değerlerini değil, modelin çıktısından kaynaklanan kaybı hesaplama işlemi gerçekleştirsek bile, bu ağırlıklar ve bias değerleri, kaybı doğrudan etkiler.

Olası ağırlık ve bias kombinasyonlarının sayısı sonsuz olduğu için en uygun ağırlık ve bias değerini rasgele olarak değiştirmek ve aramak uygun değildir (Kinsley & Kukiela, 2020).

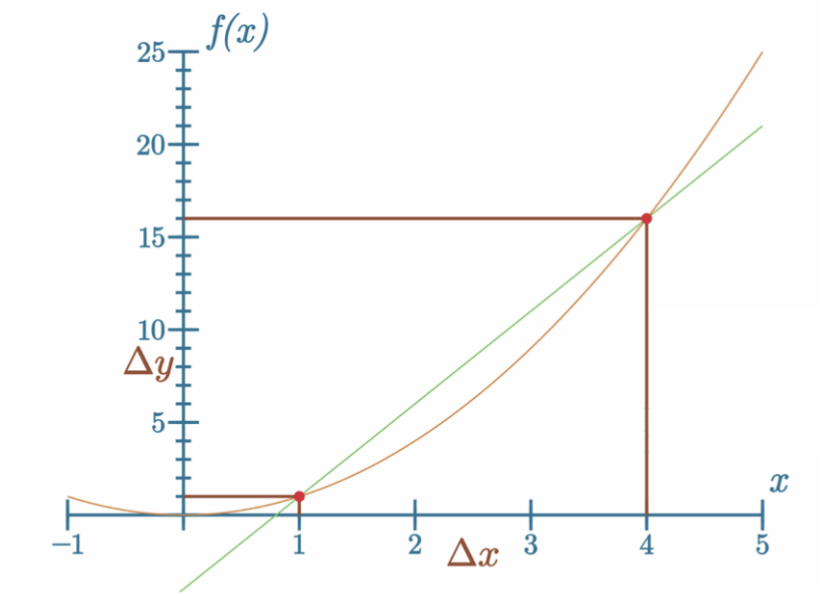
Her bir ağırlık ve bias değeri, kayıp üzerinde farklı derecelerde etkiye sahip olabilir. Bu etki, birinci katmana girdi olarak verilen mevcut örneklemin yanı sıra parametrelerin kendisine de bağlıdır. Bu girdi değerleri daha sonra ağırlık değerleri ile çarpılır. Böylece girdi verileri nöronun çıktısını etkiler ve ağırlıkların kayıp üzerindeki etkisini etkiler. Aynı durum, önceki katmanın çıktılarını girdi olarak alarak diğer katmanlardaki ağırlık ve bias parametreleri için de geçerlidir. Bu, çıktı değerleri üzerindeki etki, parametrelere olduğu kadar örneklere de bağlıdır. Her örnek için kayıp değeri ayrı ayrı hesaplanır. Ayrıca, bir ağırlık veya bias değerinin genel kaybı nasıl etkilediğinin fonksiyonu doğrusal olmayabilir. Ağırlık ve bias değerlerinin nasıl ayarlanacağını bilmek için, öncelikle bu parametrelerin kayıp üzerindeki etkileri anlaşılmalıdır (Rashid, 2016).

Temel olarak, her bir tekil ağırlığın ve bias'ın verilen bir örnek için kayıp değeri üzerinde ne kadar etkisi olduğu, kayıp değerini ne kadar değiştirdiği ve bu değerin düşmesi için ağırlık ve bias değerlerinin nasıl güncellendiği (üzerinde ne kadar etkisi olduğunu) hesaplanır. Burada amaç, kaybı azaltmaktır ve bu işlem gradyan, kısmi türev, gradyan iniş kullanılarak gerçekleştirilir.

Gradyan, tensör işleminin türevi olup kısmi türevlerin hesaplanmasının bir sonucudur. Tüm ağırlıkları ve bias değerlerini güncellemek için zincir kuralı kullanılarak, geriye yayılması işlemi gerçekleştirilir. Kısmi türevleri anlamak için, kısmi türevlerin özel bir durumu olan türev işlemini hatırlamak gerekir. Türev, tek parametre alan fonksiyonlar için hesaplanır (Chollet, 2017).

Bir fonksiyonun nasıl hareket ettiğini belirleyebilmek, fonksiyonun hangi noktalarda artış ya da azalış gösterdiği gibi durumları inceleyebilmek için eğimlerden yararlanılmaktadır. Bu hareket fonksiyondaki değişkende oluşacak değişimin fonksiyonda oluşturacağı duruma, değişkendeki değişime oranıdır. Fonksiyona teğet bir $d$ doğrusu çizilecek olursa koordinat düzleminde $x$'teki değişim $f(x)$ fonksiyonunda da değişime neden olacaktır.

$f(x)$ fonksiyonunda oluşturacağı değişim $x$'teki değişime oranlandığında $d$ teğetinin eğimi elde edilir. Bu da $f(x)$ fonksiyonunun $x$ noktasındaki türevidir. Yani fonksiyonun o noktadaki türevi, o noktada ilgili eğriye teğet olan doğrunun eğimidir. Bu durumda fonksiyonun hangi yönde arttığının anlaşılabilmesi açısından eğim kullanılabilir. $y = f(x)$ eğrisinin herhangi bir noktasında dururken en tabana inmek istenirse o noktada fonksiyonun türevi, $x$'teki değişim hangi yönde olursa yukarı ya da aşağı hareketi belirler. Türev ile elde edilen eğim hareketi artırmaya yönelik olursa yani değişkeni eğimle artırılırsa üste çıkılabilir. Ayrıca eğim hareketi azaltmaya yönelik olursa yani değişkeni eğimin tersi yönünde artırılırsa minimuma ulaşılabilir (Kinsley & Kukiela, 2020).




**Şekil 2.14.** Türevin geometrik olarak gösterimi (Kinsley & Kukiela, 2020)

### 2.1.8. *Kısmi türev, gradyan ve zincir kuralı*

Kısmi türev, tek bir girdinin bir fonksiyonun çıktısı üzerinde ne kadar etkisi olduğunu ölçmektedir. Kısmi türevi hesaplama yöntemi, türevlerle aynıdır; bağımsız girdilerin her biri için bu işlemi tekrarlamamız yeterlidir. Etki 0 olsa bile, fonksiyon girdilerinin her birinin bu fonksiyonun çıktısı üzerinde bir etkisi vardır. Bu etkileri bilmemiz gerekir; bu, her biri hakkında bilgi edinmek için her girdiye göre türevi ayrı ayrı hesaplamamız gerektiği anlamına gelir. Verilen girdiye göre bu kısmi türevleri bu tekil bir girdiyle ilişkili olarak türevin bir kısmı hesaplanır. Kısmi türev tek bir eşitliktir.

Gradyan olarak adlandırılan çok değişkenli fonksiyonun kısmi türevi ise bir dizi denklemden oluşur. Gradyan, girdilerin her birine göre kısmi türev çözümleri içeren girdi boyutunun bir vektörüdür. Gradyan bir fonksiyonun tüm kısmi türev bilgilerini içerir ve fonksiyonun değişken sayısı ne olursa olsun tüm bilgileri tek vektörde bir araya getirir.

Gradyan boyutsal koordinat düzleminde en uzun yüksekliği tercih etmesinden dolayı artışın en çok olduğu yönü temsil eder. Bu nedenle gradyan yönünde hareket edilirse maksimuma ve benzer şekilde gradyanın tersi yönünde hareket edilirse de minimuma ulaşılır.

Kısmi türevi verilen girdinin çıktıya etkisini bulmaya çalışırken diğer tüm girdileri sabitler olarak ele aldığımız bir durum olan kısmi türevi belirtmek için Euler $\partial$ gösterimini kullanalım. Modeldeki amaç parametreleri güncellemek olduğu için tekil girdilerin etkisi ile ilgilenilir. Kısmi türev Eşitlik 2.5'deki gibi ifade edilir.

$$f(x,y,z) \rightarrow \frac{\partial}{\partial x} f(x,y,z), \frac{\partial}{\partial y} f(x,y,z), \frac{\partial}{\partial z} f(x,y,z) \tag{2.5}$$

Girdi vektörü $x$'i alıp çıktı $\hat{y}$ üreten bir sinir ağı oluşturduğumuzda, bilgi ağ içerisinde ileriye doğru akar. $x$ girdisi ilk bilgiyi sağlar. Daha sonra bu bilgi bütün gizli katmanlardaki gizli birimlere yayılır ve son olarak $\hat{\boldsymbol{y}}$ çıktısını üretir. Buna ileri yayılım adı verilir. İleri yayılım sırasında, girdi verileri nöronlardan, ardından aktivasyon fonksiyonundan, daha sonra bir sonraki katmandaki nöronlardan, ardından başka bir aktivasyon fonksiyonundan çıktı katmanına ulaşana kadar geçirilir. Girdi parametresi olan bir fonksiyon çağırılır, bir çıktı alınır ve bu çıktı başka bir fonksiyona girdi olarak verilir. Basit bir örnek için $f$ ve $g$ olmak üzere Eşitlik 2.6 ve Eşitlik 2.7'deki gibi iki fonksiyon düşünelim.

$$z = f(x) \tag{2.6}$$

$$y = g(z) \tag{2.7}$$

Eşitlik 2.6'de $x$ girdi verisidir, $z$ $f$ fonksiyonunun bir çıktısıdır. Eşitlik 2.7'de belirtildiği gibi $z$ aynı zamanda $g$ fonksiyonu için bir girdidir ve $y$, $g$ fonksiyonunun bir çıktısıdır. Aynı hesaplama, Eşitlik 2.8'deki gibi ifade edilebilir.

$$y = g(f(x)) \tag{2.8}$$

Eşitlik 2.8'de, $g$ fonksiyonu $f$ fonksiyonunun çıktısını doğrudan girdi olarak aldığı gösterilerek, ara $z$ değişkeni kullanılmamaktadır. Bu, Eşitlik 2.9'daki denklemden çok farklı değildir. Ancak bu şekilde zincirlenmiş fonksiyonların önemli bir özelliğini gösterir. Çünkü $x$, $f$ fonksiyonunun bir girdisidir ve sonra $f$ fonksiyonunun çıktısı, $g$ fonksiyonunun girdisidir, $g$ fonksiyonu bir şekilde $x$'den etkilenir.

Derin sinir ağları modelinde ileri yayılım, bu şekilde bir fonksiyonlar zinciridir. Zincirleme fonksiyonların türevi, sonraki fonksiyonların kısmi türevlerinin çarpımına eşittir.

$$\frac{d}{dx}f(g(x)) = \frac{d}{dg(x)}f(g(x)).\frac{d}{dx}g(x) = f'\big(g(x)\big).g'(x) \tag{2.9}$$

Aynı durum Eşitlik 2.10'da verilen kısmi türevler için de geçerlidir.

$$\frac{\partial}{\partial x}f(g(y,h(x,z))) = f'(g(y,h(x,z))).g'(y,h(x,z)).h'(x,z) \tag{2.10}$$

Zincir kuralı, türevi bilinen fonksiyonların birleşiminden oluşan fonksiyonların türevlerini hesaplamada kullanılır. Geri yayılım da çok etkili olan işlemleri özel bir sırayla kullanarak bu zincir kuralını hesaplayan bir algoritmadır. Geri yayılım terimi ile ilgili genel bir yanlış anlaşılma vardır. Geri yayılım, çok katmanlı sinir ağlarındaki tüm öğrenme algoritması gibi düşünülür. Aslında geri yayılım, sadece gradyan hesaplama metoduna verilen addır. Gradyan ($\nabla$) bir fonksiyonun tüm olası kısmi türevlerinden oluşan bir vektördür. Zincir kuralını kullanarak model eğitiminin bir parçası olan geriye doğru yayılımı gerçekleştirmek için tek parametreli fonksiyonların türevleri ve çok değişkenli fonksiyonların gradyanları kullanılır. Gradyan, tüm olası kısmi türevlerin vektörüdür. Üç girdi biriminden oluşan fonksiyona bir örnek Eşitlik 2.11'de verilmiştir (Kinsley & Kukiela, 2020).

$$\nabla f(x,y,z) = \begin{bmatrix} \frac{\partial}{\partial x}f(x,y,z) \\ \frac{\partial}{\partial y}f(x,y,z) \\ \frac{\partial}{\partial z}f(x,y,z) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{bmatrix} f(x,y,z) \tag{2.11}$$

### 2.1.9. *Kayıp fonksiyonu*

Amaç ya da maliyet fonksiyonu olarak da adlandırılan kayıp fonksiyonu (loss function), derin sinir ağları modeli kurulduktan sonra model eğitimi aşamasında her bir adımda ağın ürettiği sonuç ile gerçek değer arasındaki farkın değerlendirilmesi olarak tanımlanabilir (Haykin, 2009).

Derin sinir ağlarında, farklı problemler için farklı kayıp fonksiyonları kullanılmaktadır. Regresyon analizi problemlerinde derin sinir ağlarında kullanılan hata kareler ortalaması (Mean Squared Error-MSE) kayıp fonksiyonu kullanılmaktadır. Sınıflandırma probleminde model, çıktı katmanı için Softmax aktivasyon fonksiyonunu kullanır ve bir olasılık dağılımı çıktısı verir. Kategorik çapraz-entropi (categorical cross-entropy) fonksiyonu, temel gerçek (ground-truth) olasılığı ($y$) ve tahmin edilen dağılım olasılığı ($\hat{y}$) ile karşılaştırmak için kullanılır. Kayıp fonksiyonu olarak sınıflandırma problemlerinde, Softmax aktivasyon fonksiyonu ile en sık kullanılan kayıp fonksiyonudur.

$y$ (gerçek / hedef) ve $\hat{y}$ (tahmin) kategorik çapraz entropi kayıp fonksiyonu Eşitlik 2.12'de verilmiştir.

$$L_i = -\sum_j y_{i,j} \log\left(\hat{y}_{i,j}\right) \tag{2.12}$$

Eşitlik 2.12'de, $L_i$ kayıp değerini, $i$ setteki $i$'inci örneği, $j$ çıktı indisini, $y$ hedef değerlerini ve $\hat{y}$ tahmin edilen değerleri ifade etmektedir.

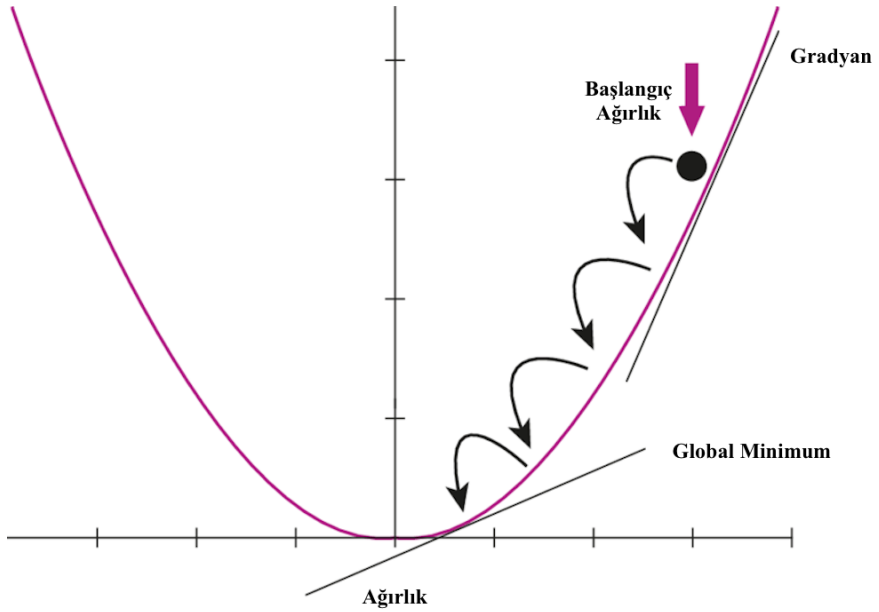
Derin sinir ağının çıktı katmanının üç sınıftan oluştuğunu ve Softmax sınıflandırma katmanı sonucunda [0.7, 0.1, 0.2] olasılık dağılım sonuçlarını elde ettiğimizi varsayalım. Hedeflenen sınıflandırma değeri (ikili- one-hot encoded) sonuç ise [1,0,0] olsun. Bu durumda kategorik çapraz entropi kayıp fonksiyonu Eşitlik 2.13 ve 2.14'deki gibi hesaplanır (Kinsley & Kukiela, 2020).

$$L_i = -\sum_j y_{i,j} \log\left(\hat{y}_{i,j}\right) = -(1.\log(0.7)) + 0.\log(0.1) + 0.log(0.2)\,)) \tag{2.13}$$

$$= -(-0.3566749393873245 + 0 + 0) = 0.35667494393873245 \tag{2.14}$$

### 2.1.10. *Gradyan inişi ve türleri*

Optimizasyon algoritmaları, ağın ürettiği çıkış değeri ile gerçek değer arasında farkı en küçüklemek için kullanılan yöntemdir (Ruder, 2016). Derin sinir ağlarını en iyilemek için en popüler yaklaşımlardan biri gradyan inişidir (Gradient Descent). Şekil 2.15'de Gradyan optimizasyonu gösterilmektedir.




**Şekil 2.15.** Gradyan optimizasyonu (Han, Kim W, Kim S & Youn, 2018)

Tek iterasyonda kullanılan veri setinin büyüklüğüne göre üç gradyan inişi yöntemi (Batch Gradient Descent, Stochastic Gradient Descent, Mini-Batch Gradient Descent) vardır (Ruder, 2016).

#### 2.1.10.1. *Batch gradyan iniş (Batch gradient descent)*

Vanilya Gradyan İnişi olarak da adlandırılmaktadır. Hata fonksiyonunun gradyan inişini tüm eğitim veri seti üzerinde hesaplamaktadır. Tüm veri setini tek seferde kullanmasından dolayı bellek yetersizliği problemi ortaya çıkmaktadır. Hesaplama maliyeti yönünden yüksek olmasına rağmen diğer gradyan iniş türlerine göre daha kararlıdır. Hata fonksiyonunun $\theta$ parametresine bağlı olarak eğiminin hesaplanması Eşitlik 2.15'de gösterilmiştir.

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta_t) \tag{2.15}$$

Burada, $\theta \in \mathbb{R}^d$: model parametreleri; $\eta$: Öğrenme katsayısı; $\nabla_\theta J(\theta_t)$ parametrelere bağlı olarak hata fonksiyonun gradyanı olarak ifade edilmektedir.

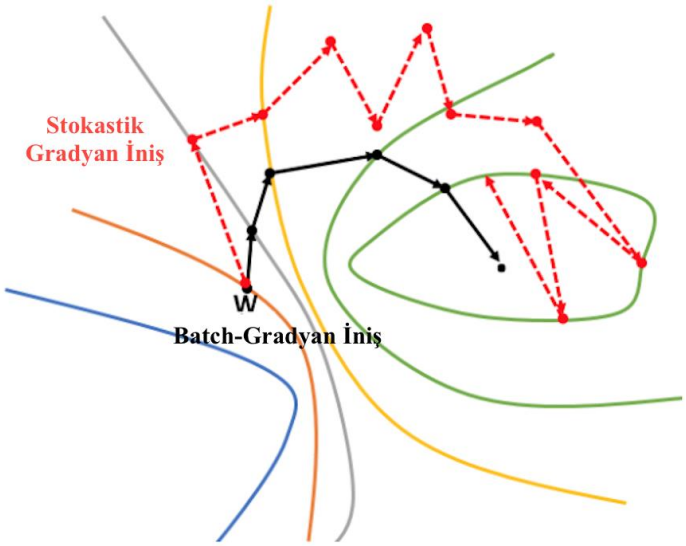
Tek seferde tüm veri seti ele alınarak eğimler hesaplandığı için bu yöntem oldukça yavaştır ve hafızaya sığmayacak büyüklükteki veri setleri için uyarlanması güçtür (Ruder, 2016).

#### 2.1.10.2. *Stokastik gradyan iniş (Stochastic gradient descent-SGD)*

Olasılıksal gradyan iniş olarak da adlandırılan Stokastik gradyan inişi, batch gradyan iniş algoritmasının aksine her bir eğitim örneği $x^{(i)}$ ve $y^{(i)}$ etiketi için bir parametre güncellemesi yapmaktadır. Alınan her eğitim verisi rasgele seçilmektedir. Eşitlik 2.16'daki gibi ifade edilebilir.

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta_t; x^{(i)}; y^{(i)}) \tag{2.16}$$

Bu yöntemde, global minimum noktasına giden yol, Şekil 2.16'da iki boyutlu çizilen paket eğim düşümünde olduğu gibi doğrudan giden bir yol değil, zikzak çizerek ilerleyen bir yoldur. Zikzak çizmesine rağmen yine de global minimum noktasına ulaşmaktadır. Bu yöntem bir önceki yöntemin aksine, tüm eğitim verisi yerine sadece eğitim örneğini alarak ağırlık güncellemesi yaptığı için daha verimlidir (Soni, 2018).




**Şekil 2.16.** Stokastik ve batch gradyan iniş gösterimi (Kurt, 2018)

#### 2.1.10.3. *Mini-batch gradyan iniş (Mini-batch gradient descent)*

Mini-batch gradyan iniş, eğitim setindeki her bir mini parça grup için bir güncelleme gerçekleştirir. Eğitim setinden $n$ adet mini parçalar halinde gruplar seçmek parametre güncellemelerinin varyansını azaltır. Bu sayede daha kararlı yakınsama sağlanabilir.

Mini-batch gradyan inişi derin sinir ağı uygulamalarında sıklıkla tercih edilmektedir. Bu yöntem, yerel minimum noktalarından kaçınarak global minimum noktasına hızlı bir yakınsama sağlamaktadır. Hesaplama işlemi Eşitlik 2.17'de olduğu gibi ifade edilebilir (Ruder, 2016):

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta_t; x^{(i,i+n)}; y^{(i,i+n)}) \tag{2.17}$$

### 2.1.11. *Gradyan iniş algoritmalarında kullanılan fonksiyonlar*

#### 2.1.11.1. *Momentum*

Hata (kayıp) fonksiyonunun küçültülmesi aşamasında Stokastik gradyan inişinin en küçük değere ulaşmasını kolaylaştırmak için kullanılır. Hata fonksiyonunun geri yayılım yapılarak ağırlık ve bias değerlerine göre küçültülmesi sürecinde her bir $t$ adımında hesaplanan hata değerleri farklıdır. Bu da her bir adımda farklı türev değerinin oluşması anlamına gelmektedir. Momentum yöntemi ile belirlenen sayıda daha önceden hesaplanmış türev değerlerinin üssel olarak ağırlıklandırılmış ortalaması alınmaktadır. Bu değer daha sonra ağırlık ve bias güncelleme eşitliğinde çarpan olarak kullanılır.

Stokastik gradyan iniş tek başına uygulandığında, başlangıçta belli bir büyüklüğe sahip olan hata değeri, her adımda hesaplanan farklı türev değerlerine göre en küçük noktaya çok fazla salınım yaparak ilerler. Bu şekilde bir kullanımda en küçük değere ulaşmak uzun sürmektedir (Qian, 1999). Bu salınımların daha az olması ve bu sayede en küçük değere daha kısa sürede ulaşmasına imkan sağlayan momentumlu gradyan inişi formülü Eşitlik 2.18 ve 2.19'da verilmiştir.

$$v_{t+1} = \gamma v_t + \eta \nabla_\theta J(\theta) \tag{2.18}$$

$$\theta_{t+1} = \theta_t - v_{t+1} \tag{2.19}$$

Burada, $v_t$: momentum değeri, $\gamma$: önceki momentum değeri katsayısı, $\eta$: öğrenme katsayısıdır (Ruder, 2016).

#### 2.1.11.2. *Nesterov hızlandırılmış gradyan (Nesterov accelerated gradient-NAG)*

Nesterov hızlandırılmış gradyan, hata fonksiyonunun küçültülmesinde aşırı hızlı bir şekilde küçülmeyi önlemek, kontrollü bir şekilde ilerlemek ve daha hızlı bir çözüm üretilmesinde katkı sağlayan bir yöntemdir (Bengio, Boulanger & Pascanu, 2013).

$\theta$ parametre değerlerinin momentumu $\gamma v_t$ ile güncelleneceği bilinmektedir. Buna göre $(\theta - \gamma v_t)$ değerini hesaplamak, parametrelerin bir sonraki yaklaşık değerlerini verir. Formülü, Eşitlik 2.20 ve 2.21'de verilmiştir (Botev, Lever & Barber, 2017) .

$$v_{t+1} = \gamma v_t + \eta \nabla_\theta J(\theta - \gamma v_t) \tag{2.20}$$

$$\theta_{t+1} = \theta_t - v_{t+1} \tag{2.21}$$

#### 2.1.11.3. *Adagrad (Adaptive Gradient)*

Adagrad, her $t$ adımda farklı öğrenme katsayısı kullanarak, her bir parametre için farklı güncelleme yapmaktadır. Öğrenme katsayısı manuel olarak ayarlama ihtiyacını ortadan kaldırmıştır. Formülü, Eşitlik 2.22 ve Eşitlik 2.23'de verilmiştir. Adagrad'da her parametrenin kendi öğrenme hızı vardır ve güncelleme işleminde öğrenme katsayısı değerinin bölündüğü ifadenin eğitim süresince büyümeye devam etmesi sonucunda öğrenme katsayısı aşırı küçülmektedir.

$$g_{t,i} = \nabla_{\theta_t} J(\theta_{t,i}) \tag{2.22}$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} g_{t,i} \tag{2.23}$$

$G_{t,ii}$: $\theta_i$ parametresine göre, $t$. iterasyona kadar hesaplanmış gradyan değerlerinin kareleri toplamı, Her biri $i, i$ konumundaki köşegen elemanı,

$g_{t,i}$: $t$ anında, $\theta_i$ parametresine göre hesaplanmış, hata fonksiyonunun gradyan değerini,

$\varepsilon$: Öğrenme katsayısının 0'a bölünmemesi için atanan sabit değeri $(10^{-8})$ ifade etmektedir (Çarkacı, 2018).

#### 2.1.11.4. *Adadelta*

Adagrad fonksiyonundaki öğrenme oranının sonsuz derecede küçük olma problemine çözüm getirmesi için geliştirilmiş optimizasyon fonksiyonudur. Öğrenme katsayısı parametresine ihtiyaç duymamaktadır. Adadelta yöntemi Adagrad fonksiyonundan farklı olarak önceki gradyanların karelerinin toplamının tamamını almak yerine, belirli bir oranını almaktadır. Eşitlik 2.24'de gösterilmektedir.

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \tag{2.24}$$

$E[g^2]_t$: geçmiş eğim karesi değerleri ortalamasını

$g_t^2$: $t$ anında, $\theta$ parametresine göre hesaplanmış, hata fonksiyonunun gradyan değeri karesini ifade etmektedir (Zeiler, 2012).

#### 2.1.11.5. *Rmsprop*

Rmsprop, Adagrad algoritmasında öğrenme katsayısının aşırı küçültme sorununa bir çözüm yolu olarak geliştirilmiştir. Adagrad algoritmasındaki geçmiş bütün eğimlerin karelerinden elde edilen değerlerin tamamını kullanmak yerine değer miktarını belli bir çerçeve boyutu ile kısıtlamıştır (Ruder, 2016; Kurt, 2018). Eşitlik 2.25, Eşitlik 2.26, Eşitlik 2.27'de verilmiştir.

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1 g_t^2 \tag{2.25}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}} g_t \tag{2.26}$$

$$g_t = \nabla_{\theta_t} J(\theta_t) \tag{2.27}$$

#### 2.1.11.6. *Adam (Adaptive Moment)*

Bu yöntemde, Rmsprop algoritmasında olduğu gibi geçmiş gradyanların karelerinin üssel olarak ağırlıklandırılmış ortalamalarının $(v_t)$ kullanılmasının yanında, momentum değişikliklerini $(m_t)$ dahil edilmiştir. Yani RMSProp ve momentumu birleştirir. Varsayılan değerler $\beta_1$ için 0.9, $\beta_2$ için 0.999 ve $\varepsilon$ için $10^{-8}$ olarak belirtilmiştir (Çarkacı, 2018). Eşitlik 2.28, Eşitlik 2.29, Eşitlik 2.30 ve Eşitlik 2.31'de verilmiştir.

$$Em_t = \beta_1 \mathrm{m}_{t-1} + (1-\beta_1)\mathrm{g}_t \tag{2.28}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{2.29}$$

$$m_t' = \frac{m_t}{1-\beta_1^t};\ v_t' = \frac{v_t}{1-\beta_2^t} \tag{2.30}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t' + \varepsilon}} m_t' \tag{2.31}$$

#### 2.1.11.7. *Nadam*

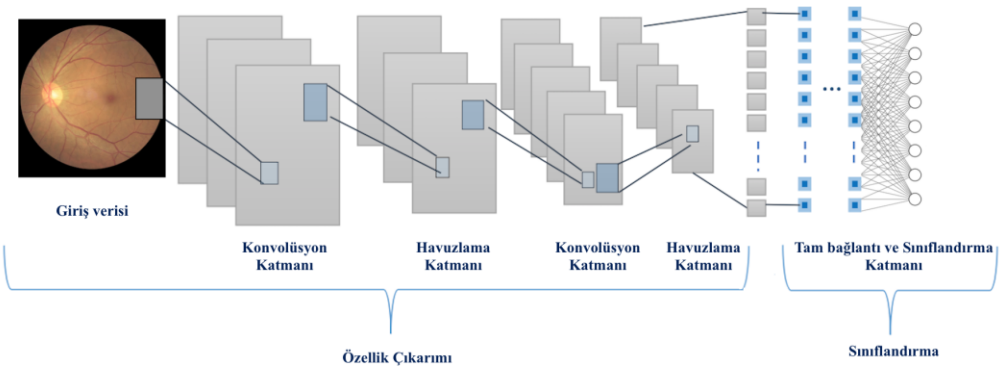
Nadam, Adam ve Nesterov hızlandırılmış gradyan (NAG) yöntemlerinin birleşiminden oluşmaktadır. NAG algoritmasını Adam algoritmasına dahil etmek için, momentum ifadesinde değişiklik yapılarak sonuçta mevcut güncelleştirme kuralı Eşitlik 2.32'deki gibi elde edilmiştir (Ruder, 2016)

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t' + \varepsilon}}\left(\beta_1 m_t' + \frac{(1-\beta_1)g_t}{1-\beta_1^t}\right) \tag{2.32}$$

## 2.2. Konvolüsyonel Sinir Ağları

Derin öğrenme, farklı problemlerin çözümü için oluşturulan farklı mimarilere sahiptir. Bu mimari yapıların başında bilgisayarlı görü araştırma ve uygulama alanında Konvolüsyonel Sinir Ağları (Convolutional Neural Networks-CNN) gelmektedir. Son yıllarda, tıbbi görüntülerden hastalıkların sınıflandırılması, nesne tanımlama, görüntü bölütleme, yüz bilgisi çıkarma, otomatik video sınıflandırma gibi alanlarda CNN mimarileri önemli bir gelişme göstermiştir (Chollet, 2017; Goodfellow vd., 2016).

Konvolüsyonel sinir ağları, insan görme sisteminin modellemesiyle oluşturulmuştur. Geleneksel CNN mimarisi girdi katmanı, konvolüsyon katmanı, havuzlama katmanı, tam bağlantı katmanı (fully-connected layer) ve çıktı katmanı olmak üzere beş ana katman içermektedir. Araştırmacılar, bu beş katmanın farklı dizilişleriyle oluşturulmuş AlexNet, ResNet, Inception, VGG gibi CNN mimarileri ile geliştirme çalışmaları yapmaya devam etmektedir. Bu bölümde CNN mimarisinde bulunan beş katman daha detaylı olarak incelenecektir. Örnek bir CNN mimari yapısı Şekil 2.17'de görülmektedir.




**Şekil 2.17.** Konvolüsyonel sinir ağı mimarisi gösterimi

### 2.2.1. *Giriş katmanı*

Bu katman mimarideki ilk katmandır. Bu katmanda girdi verileri olan görüntüler ağa verilmektedir. Tasarlanacak modelin başarımı için bu katmandaki verinin boyutu önemlidir. Giriş görüntü boyutunun yüksek seçilmesi hem yüksek bellek ihtiyacını hem eğitim süresini hem de görüntü başına düşen test süresini uzatabilir. Bunun yanında ağ başarısını arttırabilir.

Giriş görüntü boyutunun düşük seçilmesi bellek ihtiyacını azaltır ve eğitim süresini kısaltır. Ancak kurulacak ağın derinliği azalır ve performansı düşük olabilir. Görüntü analizinde hem ağ derinliği hem donanımsal hesaplama maliyeti hem de ağ başarısı için uygun bir giriş görüntü boyutu seçilmelidir (Chollet, 2017; Kinsley & Kukiela, 2020).

### 2.2.2. *Konvolüsyon katmanı*

Konvolüsyon katmanı ile öznitelikler çıkartılır ve çoğaltılır. Bu katman basit bir filtreleme işlemi gerçekleştirir. Filtreler görüntü boyunca kaydırılır. Kaydırma sırasında filtrenin üzerinde bulunduğu görüntü piksel değerleri ile filtredeki değerler çarpılır ve elde edilen değerler toplanır ve net sonuç bulunur. Bu işlemi tüm görsele uyguladığımızda yeni bir görüntü edilir. Elde edilen görüntüye konvolüsyon özniteliği (Convolved Feature) adı verilir (Goodfellow vd., 2016).

Her bir filtre aslında yapay sinir ağlarına ait bir nörondur. Bu nöronun ağırlıkları ise filtre içerisindeki değerlerdir. Filtre boyutu büyüdükçe çıkış matrisi küçülecektir. Bu küçülme giriş matrisindeki bilgileri kaybetme anlamına gelmektedir. Genellikle $3x3$, $5x5$, $7x7$ boyutunda filtreler giriş matrisi üzerinde kullanılmaktadır. Kaydırma işlemi uygulanırken filtre boyutunun dışında kaydırma adımı (stride) ve piksel ekleme (padding) işlemlerine de ihtiyaç duyulmaktadır.

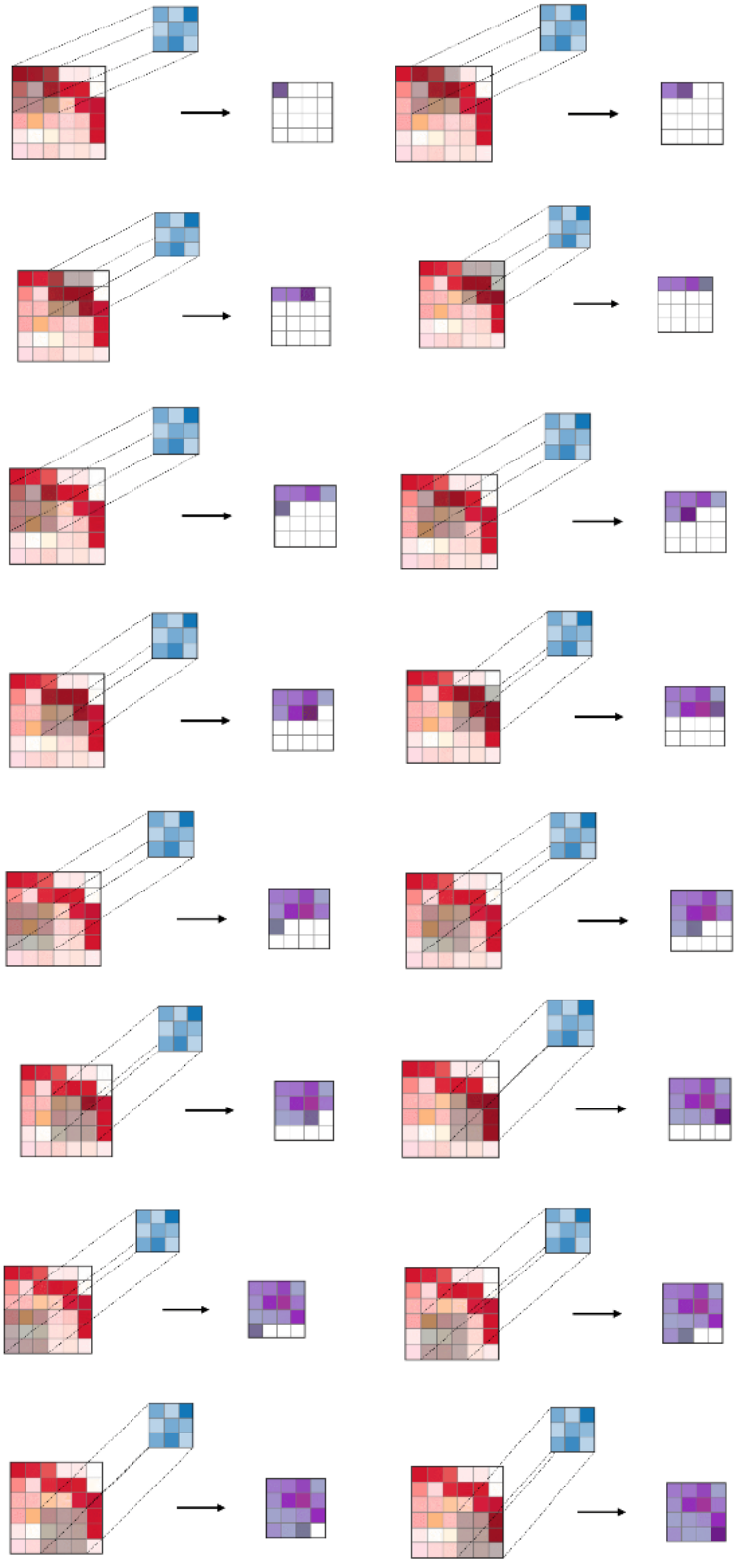
Adım işlemi, filtre uygulanırken filtrenin ne kadar aralıklarla kaydırılacağını belirtmektedir. Boşluk işlemi ise kenar değerlerinin işleme nasıl katılacağını ifade eder. Her bir filtre iki boyutlu giriş matrisinde ayrı bir özelliğe odaklanmaktadır. Uygulanan filtreler ağırlıklardan oluşmaktadır. Ağırlıklar ise ağın eğitimi ile öğrenilmektedir (Kinsley & Kukiela, 2020).

Konvolüsyon işleminden sonra giriş matrisinin boyutunda bir azalma olur. Giriş matrisinin boyutunun azalmasının istenmediği durumlarda matrisin eksik boyutlarına değer atanır. Bu atanan değer, boşluk doldurma (zero padding) işlemi ile gerçekleştirilir. Giriş matrisinin boyutu $g\ x\ g$ ağırlık matrisinin boyutu $a\ x\ a$ olduğunda, çıkış matrisi $(g-a+1)x(g-a+1)$ şeklinde hesaplanır.

Kaydırma adımı (stride) konvolüsyon işlemi için ağırlık matrisi olan filtreyi görüntü üzerinde birer piksellik adımlarla ya da daha büyük adımlarla kaydıracağının bilgisini verir. Bu da doğrudan çıkış boyutunu etkileyen diğer bir parametredir. Çıkış matrisinin boyutu Eşitlik 2.33'deki gibi hesaplanır (Kızrak, 2020).

$$[(\frac{g+2p-a}{s})+1]x[(\frac{g+2p-a}{s})+1] \tag{2.33}$$

Şekil 2.18'de $6x6$ boyutundaki giriş matrisine $3x3$ boyutundaki filtrenin kullanımı, $4x4$ boyutunda konvolüsyon özniteliğinin elde edilmesi gösterilmiştir.

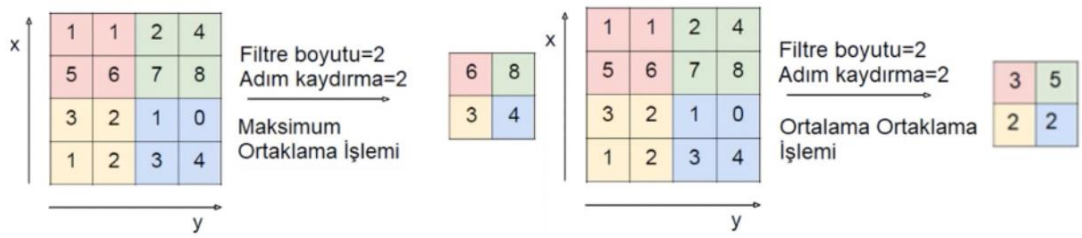


**Şekil 2.18.** Konvolüsyon özniteliğinin elde edilmesi (Amidi & Amidi, 2021)

### 2.2.3. *Havuzlama (Ortaklama) katmanı*

Konvolüsyonel Sinir Ağlarında boyut azaltmak için kullanılan bir diğer işlem havuzlama ya da ortaklama (pooling) işlemidir. Bu katmanda maksimum ortaklama (max-pooling) yöntemi yaygın olarak kullanılmaktadır. Ortaklama katmanı konvolüsyon katmanından sonra belirlenen aktivasyon fonksiyonu ile üretilen sonuçtan sonra kullanılır. Amacı girdi olarak verilen görüntünün alt örnekleme işlemini gerçekleştirmektir. Bu işlem ile görüntünün alt örneklemi elde edilerek modelin ezberleme yapmasını engellemek, yani aşırı uyum (overfitting) probleminden kaçınmak ve bir sonraki katman için hesap yükünü azaltmak amaçlanır (Dumoulin & Visin, 2016).

Ortaklama katmanı, konvolüsyon katmanında olduğu gibi işlemleri filtreler üzerinden gerçekleştirmektedir. En sık kullanılan ortaklama işlemi görüntü üzerinde filtreye karşılık gelen değerler içinden en büyük değeri (max pooling) alabildiği gibi, tüm değerlerin ortalaması (mean pooling) ya da en küçük değeri (min pooling) alarak uygulanabilir (İnik ve Ülker, 2017; Amini, 2021; Friedman, 2020). Şekil 2.19'da filtre boyutu 2, kaydırma adımı 2 olarak alındığı maksimum ve ortalama ortaklama işlemlerinin gerçekleştirilmesi gösterilmektedir.

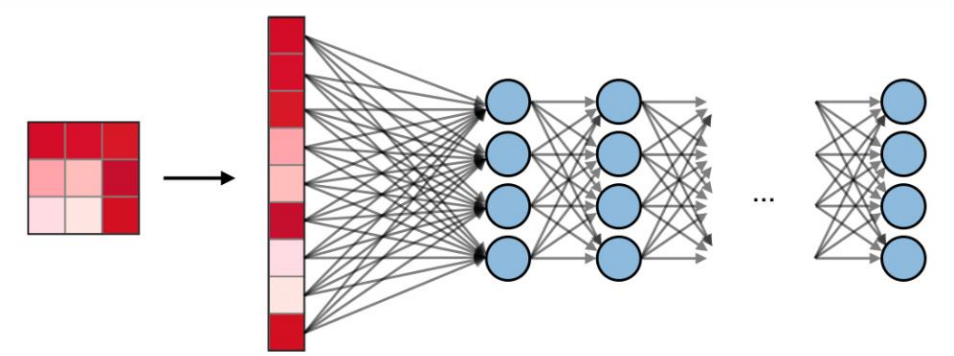



**Şekil 2.19.** Maksimum ve ortalama ortaklama işlemi (Piette, 2018)

### 2.2.4. *Tam bağlantı katmanı*

Tam Bağlantı katmanı, kendinden önceki katmandan gelen verilere tam bağlı nöronları içermektedir. Bu katmandan önce matris girişlerini tek boyutlu bir yapıya dönüştüren düzleştirme (flattening) işlemi gerçekleştirilir. Ağın çıkışında elde edilecek değerler sayı değerleri olacağı için çok boyutlu veriler (matrisler) ile işlem yapıldıktan sonra bu katmanda boyut indirgeme yapılarak öznitelik haritasından elde edilen veriler tek boyutlu hale getirilir. Tek boyutlu veri sınıflandırma katmanına girdi olarak verilir.

Şekil 2.20'de tam bağlantı katmanına giren matrisin tek boyutlu bir yapıya dönüştürüldüğü düzleştirme (flattening) işlemi ve tam bağlantı katman yapısı gösterilmiştir (Amidi & Amidi, 2021).



**Şekil 2.20.** Örnek tam bağlantı katmanı (Amidi & Amidi, 2021)

### 2.2.5. *Sınıflandırma katmanı*

Sınıflandırma katmanı, öznitelikler belirlendikten sonra probleme uygun bir sınıflandırıcı ile sınıflandırma işleminin yapıldığı katmandır. Softmax sınıflandırıcısı derin konvolüsyonel sinir ağları sınıflandırma katmanında sıklıkla tercih edilmektedir. Bu nedenle sınıflandırma katmanı Softmax adımı olarak da adlandırılmaktadır (Goodfellow vd., 2016).

Softmax adımı, $x \in \mathbb{R}^n$ skorlarının bir vektörünü girdi olarak alan ve mimarinin sonunda Softmax fonksiyonundan $p \in \mathbb{R}^n$ çıkış olasılık vektörünü oluşturan genelleştirilmiş bir lojistik fonksiyon olarak görülebilir. Eşitlik 2.34 ve Eşitlik 2.35'de tanımlanmıştır (Chollet, 2017).

$$p_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{2.34}$$

$$p = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix} \tag{2.35}$$

## 2.3. Hiperparametreler

Derin sinir ağlarının eğitiminin daha az maliyetli ve yüksek başarımla gerçekleşmesini sağlamak için uygun hiperparametre seçimi önemli rol oynar. Bu tez çalışmasında derin sinir ağlarında kullanılan hiperparametreler 11 başlık altında incelenmiştir.

### 2.3.1. *Öğrenme katsayısı*

Öğrenme katsayısı, gradyan iniş algoritmalarının yakınsamasını sağlayan bir katsayıdır. Hata düzeltme katsayısı olarak da bilinmektedir. Öğrenme katsayısı, fazla büyük olduğunda global minimum noktasının etrafında takılabilir, istenen hedefe ulaşılamayabilir. Benzer şekilde öğrenme katsayısı, çok küçük seçildiğinde, algoritma her iterasyonda çok küçük adımlarla ilerleyeceğinden yakınsama çok uzun sürmektedir (Amini, 2021).

### 2.3.2. *Aktivasyon fonksiyonu*

Derin sinir ağlarında, aktivasyon fonksiyonlarının kullanımı Bölüm 2.1.6'da anlatılmaktadır. Probleme uygun oluşturulan derin sinir ağı modelinin çözümüne uygun olan aktivasyon fonksiyonu seçilmelidir. (Friedman, 2020).

### 2.3.3. *Optimizasyon algoritmasının belirlenmesi*

Derin sinir ağlarında, ağın ürettiği çıkış değeri ile gerçek değer arasındaki farkı en küçüklemede kullanılan optimizasyon algoritmaları ve gradyan inişi fonksiyonları Bölüm 2.1.10 ve 2.1.11'de anlatılmıştır. Bu bilgiler doğrultusunda oluşturulan derin sinir ağı mimarisinde en yüksek başarıma ve en düşük kayıp fonksiyonu sonucuna ulaşan optimizasyon algoritmasının tercih edilmesi önerilmektedir (Ruder, 2016).

### 2.3.4. *Derin sinir ağının genişliği ve derinliği*

Gizli katmandaki gizli düğümlerin miktarı derin sinir ağının genişliğini, ağda bulunan gizli katmanların sayısı ise ağın derinliğini ifade etmektedir (Kinsley & Kukiela, 2020).

### 2.3.5. *Epok (döngü) sayısı ve batch büyüklüğü*

Epok (döngü) sayısı, tüm veri setinin bir ileri yönde bir de geri yönde tüm ağdan geçmesidir. İterasyon, belirlenen batch büyüklüğü kadarlık verinin ileri ve geri yönde ağdan geçmesidir.

Batch büyüklüğü, ileri ve geri yönde yayılım için veri setinden alınan veri miktarıdır. Yüksek batch büyüklüğü seçilmesi her iterasyon için kullanılacak eğitim örneğinin büyümesi anlamına gelmektedir.

Epok (döngü) sayısı, eğitim algoritmasının tüm eğitim verisini toplamda kaç defa kullanacağını belirlemektedir. Doğru epok sayısını seçmek için dikkat edilmesi gereken metrik "doğrulama hatası"dır. Derin öğrenmede doğrulama hatasının ilk epoklarda yüksek, epoklar arttıkça ise düşük olması beklenmektedir, çünkü ağdaki ağırlık değerleri adım adım güncellenmektedir. Sezgisel olarak doğrulama hatası azalmaya devam ettiği sürece modelin iterasyonlarına devam etmesi beklenmektedir. Modelin ne zaman durdurulacağını belirlemek için "Erken Durdurma (Early Stopping)" adlı teknik kullanılmaktadır. Bu teknikte, eğitim setindeki hata ile doğrulama setindeki hata arasındaki fark belirlenen eşik değerinin üzerine çıkmaya başladığında eğitim süreci durdurulmaktadır (Goodfellow vd., 2016; Amini, 2021).

Epok ve mini-batch parametrelerinin kullanımına bir örnek şu şekilde verilebilir: 500 örnekten oluşan bir veri kümesi, mini-yığın büyüklüğü 5 ve epok sayısı 1000 olacak şekilde eğitilmek istensin. Bu durumda veri kümesi 100 adet yığına bölünecektir (500/5). Her bir yığında 5 örnek yer alacaktır. Her bir 5 örnek için ağırlıklar hesaplandıktan sonra hata hesaplanacak ve ağırlık güncellemeleri yapılacaktır. Bir epok 100 yığın yani model için 100 güncelleme içerecektir. Dolayısıyla 1000 epok için model tüm veri kümesinden 1000 kere yararlanacak ve tüm eğitim sürecinde toplam 100.000 güncelleme yapılmış olacaktır.

### 2.3.6. *Düzenlileştirme (Regularization)*

Düzenlileştirme yöntemi ile ağın girdideki gürültüleri öğrenerek daha fazla parametreye sahip olması engellenmektedir. Ağın ezberlemesi durumu, girdinin gürültüsü ile birlikte öğrenilmesiyle meydana gelir. Bu yöntem, ağın ezberlemesini önler.

Düzenlileştirme, aynı zamanda algoritmanın genelleştirilmesini de sağlar ve ağın karmaşıklığı arttıkça katsayıların girdi ile mükemmel uyuşmasını önler (Murugan & Durairaj, 2017). Ezberlemeyi önlemek için veri setinin arttırılması bir çözüm olarak karşımıza çıkmaktadır. Seyreltme ya da Sönümleme (Drop-out), Ağırlık Sıfırlama ve veri arttırma yöntemleri düzenlileştirme aşamasında kullanılmaktadır.

Tam bağlantılı (Fully-Connected) katmanlarda belli eşik değerin altındaki düğümlerin seyreltilmesinin ya da sönümlenmesinin başarımı arttırdığı gözlenmiştir. Yani zayıf bilgilerin unutulması ile öğrenimin arttığı gözlenmiştir. Seyreltme (Drop-out) değeri olarak genelde 0.5 kullanılmaktadır. Farklı kullanıldığı durumlar da yaygındır. Probleme ve veri setine göre değişiklik göstermektedir (Goodfellow vd., 2016). Tüm katmanlarda aynı Drop-out değeri kullanılması zorunlu değildir; farklı seyreltme (Drop-out) değerleri de kullanılabilir. Ağırlık Sıfırlama işlemi de seyreltme ile benzerdir. Ancak ağırlık sıfırlamada nöron çıktı değerleri yerine ağırlıkların bazıları sıfırlanarak modelin eğitim başarısı arttırılmaya çalışılır (Chollet, 2017).

### 2.3.7. *Ağırlık başlatma teknikleri*

Ağırlık başlangıç değerleri, ağın başarısını ve eğitim süresini doğrudan etkiler. Ağırlık değerlerinin başlangıçta 0 olarak atanması, bütün nöronların aynı çıktı değerini hesaplamasını sağlayacağından geri yayılım işlemi sırasında türev ve gradyan değerlerinin aynı çıkmasına, dolayısıyla tüm ağırlıkların aynı parametre değeriyle güncellenmesine sebep olacaktır (Ng, 2021).

Başlangıçta rasgele değerler olarak atanan ağırlıklar, tam olarak 0 değerine sahip olmamakla birlikte küçük rasgele değerlere sahip olmalıdır (Gustafson & Sartoris, 1998). Bu şekilde rastgele ve farklı ağırlıkların olması, farklı güncelleme değerlerinin üretilmesini ve ağırlıkların birbirlerinden farklı şekillerde ağa adapte olmalarını sağlayacaktır (Ng, 2021).

Çok küçük değerlerin daha iyi sonuçlar vereceği kesin değildir. Çünkü çok küçük değerler almış herhangi bir ağırlık katmanı, geri yayılım algoritması sırasında çok küçük gradyan değerlerinin hesaplanmasına yani gradyanların yok olması (vanishing gradient) adı verilen soruna neden olur (Srivastava, Greff & Schmidhuber, 2015).

Ağırlık başlatma teknikleri:

**I.** Bir: Başlangıç ağırlıklarına 1 değeri atanır.

**II.** Sabit: Başlangıç ağırlıkları olarak elle girilmiş sabit bir değer atanır.

**III.** Rasgele Normal: Ağırlık birimlerinin değerleri normal dağılım ile belirlenir.

**IV.** Kesikli Normal: Ağırlık birimlerinin değerleri kesikli normal dağılım ile belirlenir. Rastgele normal dağılıma benzer sonuçlar üretilir ama standart sapması 2'den büyük olan değerler hesaba katılmaz ve bu değerler için yeniden hesaplama yapılır. Bu yöntem, sinir ağları ve filtreler için önerilen yöntemdir.

**V.** Rasgele Uniform: Ağırlık birimlerinin değerleri Uniform dağılım ile belirlenir.

**VI.** Varyans Ölçeklendirme: Ağırlık başlangıç değerlerinin belirlenmesi, ağırlık matrisinin boyutlarına göre ölçeklendirmeyle yapılır. Değer belirleme işlemi normal dağılım seçeneğiyle yapılıyorsa, ağırlıklar sıfır merkezli kesikli normal dağılıma göre değer alırlar.

**VII.** Dikey (Orthogonal): Ağırlıklar matrisi, elemanları rastgele değerlerden oluşan bir kare matristir (Saxe, McClelland & Ganguli, 2013).

**VIII.** Birim (Identity): Ağırlıklar matrisi, birim matristir.

**IX.** Xavier Normal: Glorot Normal başlangıç ağırlık atama tekniği olarak adlandırılmaktadır. Ağırlıklar 0 ortalamalı, kesikli dağılıma göre değer alır. Atanacak değerlerin sapması Eşitlik 2.36'daki gibi ifade edilir.

$$\sigma = \sqrt{\frac{2}{n_{in} + n_{out}}} \tag{2.36}$$

$n_{in}$: Ağırlıklar matrisindeki girdilerin sayısı

$n_{out}$: Ağırlıklar matrisindeki çıktıların sayısı

**X.** Xavier Uniform: Glorot Uniform başlangıç ağırlık atama tekniği olarak adlandırılmaktadır. Ağırlıklar, [-limit, limit] aralığında Uniform dağılıma göre değer alır. Limit değeri Eşitlik 2.37'deki gibi ifade edilir.

$$\text{Limit} = \sqrt{\frac{6}{n_{in} + n_{out}}} \tag{2.37}$$

**XI.** He Normal: Ağırlıklar, 0 ortalamalı kesikli dağılıma göre değer ataması geçekleştirilir. Standart sapma Eşitlik 2.38'daki gibi ifade edilir.

$$\sigma = \sqrt{\frac{2}{n_{in}}} \tag{2.38}$$

**XII.** He Uniform: Ağırlıklar, [-limit, limit] aralığında uniform dağılıma göre değer alır. Limit değeri Eşitlik 2.39'deki gibi ifade edilir.

$$\text{Limit} = \sqrt{\frac{6}{n_{in}}} \tag{2.39}$$

**XIII.** Lecun Normal: Ağırlıklar, 0 ortalamalı kesikli dağılıma göre değer ataması geçekleştirilir. Standart sapma Eşitlik 2.40'daki gibi ifade edilir.

$$\sigma = \sqrt{\frac{1}{n_{in}}} \tag{2.40}$$

**XIV.** Lecun Uniform: LeCun düzgün başlangıç, değer tanımlayıcısında ağırlıklar, [-limit, limit] aralığında uniform dağılıma göre değer alır. Burada limit değeri Eşitlik 2.41'e göre belirlenir. $n^{l-1}$ son katmandan önceki katmanda yer alan ağırlıklar matrisindeki girdi sayısını ifade eder (LeCun, Bottou, Orr & Müller, 2012).

$$\text{Limit} = \sqrt{\frac{3}{n^{l-1}}} \tag{2.41}$$

### 2.3.8. *Konvolüsyon filtre derinliği, boyutu, adım sayısı ve dolgu (Padding)*

Filtre sayısı, filtre derinliğini ifade etmektedir. Her filtre, girdideki farklı bir özelliği aramaktadır. Adım sayısı, filtrenin girdi üzerinde ne kadar piksel kaydırılacağını belirten hiperparametredir. Adım sayısının 2 veya daha fazla bir değere sahip olması, çıktı boyutunun girdiden daha düşük olmasına neden olur (Goodfellow vd., 2016). Girdi görüntüsü ile filtreleme işlemi sonucu elde edilen görüntünün aynı boyutta olması için boşluk doldurma (padding) işlemi gerçekleştirilir (Kızrak, 2020).

### 2.3.9. *Ortaklama yönteminin belirlenmesi*

Ortaklama ya da havuzlama katmanında hangi yöntemin tercih edileceği belirlenir. Ortalama, minimum ya da maksimum ortalama yöntemi olmak üzere üç yöntem vardır (Lecun vd., 1990; Lecun, Bottou, Bengio & Haffner, 1998).

### 2.3.10. *Batch normalizasyonu*

Derin sinir ağlarında, gizli katmanlar arasındaki kovaryansı azaltmak amacıyla kullanılan normalizasyon tekniğidir. Her katmanın girişi kendisinden önceki katmanın çıkış vektörü normalize edilerek beslenir. Bunun için de ilgili çıkış vektöründen; önce ortalama değeri çıkartılır, daha sonra standart sapma değerine bölünür. Elde edilen normalize vektör sonraki katmana giriş olarak verilir ve prosedür katmanlar arasında bu şekilde devam eder. Modeli düzenli hale getirir ve seyreltme (Drop-out) düzenlileştirme tekniğine olan ihtiyacı azaltmıştır (Goodfellow vd., 2016).

### 2.3.11. *Transfer öğrenme*

Öğrenme yöntemleri, Gözetimli Öğrenme (Supervised Learning), Gözetimsiz Öğrenme (Unsupervised Learning), Pekiştirmeli Öğrenme (Reinforcement Learning) ve Transfer Öğrenme (transfer learning) olmak üzere dörde ayrılır. Gözetimli öğrenmede eğitim verisindeki sınıflandırmalar belirlidir. Sinir ağları, girdileri ve onların sınıflarını kullanarak ağırlıklarını ayarlamaktadır.

Gözetimli öğrenme yönteminde sinir ağları, girdiler ve çıktılar arasındaki ilişkiyi görmektedir ve bu ilişkiye göre girdilerden çıktıları oluşturacak olan fonksiyonu öğrenmektedir.

Gözetimsiz öğrenme yönteminde, eğitim verisi herhangi bir sınıflandırma belirtilmeksizin girdi olarak verilmektedir. Sinir ağları, bu veri üzerinde belirli desenler ya da örüntüler bulup onları sınıflandırmaya çalışmaktadır.

Pekiştirmeli öğrenme yönteminde, sinir ağları her bir iterasyonda oluşturduğu sonuç hakkında bir geri bildirim alır bu sayede doğru sonuca ne kadar yakınsadığını hesaplayabilmektedir. Bu yöntem ödül ve ceza yöntemidir. Sinir ağları, ağırlıklarını bu geri bildirime göre ayarlayarak maksimum ödüle ulaşmaya çalışmakta ve bu şekilde öğrenmektedir (Alpaydın, 2011).

İlk olarak 1998 yılında ortaya çıktığı bilinmesine rağmen, etkisi ilk kez 2012 yılında ortaya çıkan Derin Öğrenme nesne tanımlama alanındaki Büyük Ölçekli Görüntü Tanıma Yarışması (ImageNet) ile adını daha çok duyuran Transfer Öğrenme, (Aktarımlı öğrenme), belirli bir alanda eğitilmiş bir modelin başka bir işlemde kullanılmak için belirli modifikasyonlar ile yeni görüntü kümesi üzerinde tekrar eğitilmesi anlamına gelmektedir.

Medikal görüntüleme alanında bir konvolüsyonel sinir ağı modelini sıfırdan oluşturup eğitebilmek için yeterli miktarda veri bulunmamaktadır. Milyonlarca görüntü ile eğitilmiş hazır ağların yapılandırılarak kullanılması ihtiyaç duyulan veri miktarını önemli ölçüde azaltmaktadır. Daha önce ImageNet görüntü kümesi üzerinde klasik konvolüsyonel sinir ağları modelleri ile öğrenilen bilgilerin transfer edilerek yeni sınıflandırma problemini destekleyecek şekilde son katmanı dondurup tam bağlantı katmanının sınıflandırma kategorilerinin yeni sınıflandırma problemine uygun olarak tasarlanması ve eğitilmesi transfer öğrenme ya da ince ayar (fine-tuning) olarak adlandırılmaktadır (Petrovska, Stojanovic & Atanasova, 2018). Önceden eğitilen CNN modellerinden en son adımda yer alan tam bağlantılı katman çıkarılır. Hedef veri kümesindeki sınıf sayısı dikkate alınarak yeni bir tam bağlantılı katman eklenir. Yeni tam bağlantılı katmanın ağırlıkları rastgele verilir ve önceden eğitilmiş olan CNN'deki ağırlıklar kullanılır. Ağın yeni eklenen son katmanı eğitilerek bu katmandaki ağırlıklar öğrenilir (Shin vd., 2016).

Transfer öğrenme, Derin Konvolüsyonel Sinir Ağı Modellerinde sınıflandırma problemlerinde sıklıkla kullanılmaktadır. Uygulanan transfer öğrenmesi yöntemiyle tasarımcılar hem zamandan kazanç sağlarken hem de yüksek doğruluk oranları elde etme imkânına sahip olmuşlardır. Konvolüsyonel Sinir Ağlarında ön eğitimli ağırlıklar kullanılarak oluşturulan modelin yeniden eğitilmesi ile beraber ince ayar yöntemiyle ağırlık güncelleme yöntemi gerçekleştirilir. Transfer öğrenme yöntemiyle ağın rastgele başlangıç değerleri ile eğitilmesi yerine transfer öğrenmesi ile önceden eğitilmiş ağırlıklar kullanılır. Bu yöntem ile daha az veri ile konvolüsyonel sinir ağı modellerinin eğitimi etkin bir şekilde sağlanmış olur.

Bu tez çalışmasında transfer öğrenme yönteminden faydalanılarak daha önceden ImageNet görüntü kümesi üzerinde eğitilmiş klasik konvolüsyonel sinir ağı mimarileri olan Inceptionv3, ResNet50 ve VGG16 modelleri oluşturulmuştur ve modellere ait ön eğitimli ağırlıklar kullanılarak öznitelikler elde edilmiştir.

## 2.4. Klasik Konvolüsyonel Sinir Ağları

### 2.4.1. *LeNet-5*

LeNet modeli posta kodları, el yazısı uygulamalarında başarılı sonuçlar elde etmiş bir konvolüsyonel sinir ağı modelidir. 1995 yılında Yann LeCun ve Yoshua Bengio tarafından yayınlanan "Convolutional networks for images, speech, and time series" isimli çalışmada LeNet-5 modeli oluşturulmuştur. Bu model 3 konvolüsyon, 2 ortalama ortaklama ve 1 tam bağlantı katman olmak üzere toplam 6 katmandan oluşmaktadır (LeCun & Bengio, 1995). Giriş katmanında yer alan görüntü 32x32x1 boyutlarındadır. Tablo 2.1'de LeNet-5 Model Mimarisi yer almaktadır.

**Tablo 2.1.** LeNet-5 model mimarisi

| Katman | Katman Tipi | Özellik Haritası | Filtre Boyutu | Girdi Boyutu | Adım Sayısı | Aktivasyon Fonksiyonu |
|---|---|---|---|---|---|---|
| **Girdi** | Görüntü | 1 | - | 32x32 | - | - |
| **1** | Konvolüsyon | 6 | 5x5 | 28x28 | 1 | Tanh |
| **2** | Ortalama Ortaklama | 6 | 2x2 | 14x14 | 2 | Sigmoid |
| **3** | Konvolüsyon | 16 | 5x5 | 10x10 | 1 | Tanh |
| **4** | Ortalama Ortaklama | 16 | 2x2 | 5x5 | 2 | Sigmoid |
| **5** | Konvolüsyon | 120 | 5x5 | 1x1 | - | Tanh |
| **6** | Tam bağlantı | - | - | 84 | - | Tanh |
| **Çıktı** | Tam bağlantı | - | - | 10 | - | Softmax |

### 2.4.2. *AlexNet*

AlexNet 2012 yılında konvolüsyonel sinir ağlarını popülerleştiren bir çalışma olmuştur. ILSVRC-2012 yarışmasında sunulan AlexNet modeli, Alex Krizhevsky, Ilya Sutskever ve Geoffrey Hinton tarafından geliştirilmiştir. Geliştirilen bu model, en iyi 5 test hata oranında (top-5 error) %15,3'lük bir değer elde etmiştir.

Bilgisayarlı görü alanında derin öğrenmenin ilk popüler kullanımı AlexNet mimarisi ile başlamıştır. 10 milyon görüntü ve 1000 farklı görüntü kategorisi olan ImageNet veri tabanındaki görüntüleri sınıflandırmayı amaçlamıştır (Krizhevsky vd., 2012). ImageNet yarışmasını, 2012 yılında derin öğrenme mimarisi ile tasarlanan AlexNet mimarisi kazanmıştır. AlexNet'in bu başarısı herkesi heyecanlandırmıştır. Görüntü sınıflandırma performansı önceki yöntemlerden daha üstündür. Tablo 2.2'de AlexNet Model Mimarisi yer almaktadır.

**Tablo 2.2.** AlexNet model mimarisi

| Katman | Katman Tipi | Özellik Haritası | Filtre Boyutu | Girdi Boyutu | Adım Sayısı | Aktivasyon Fonksiyonu |
|---|---|---|---|---|---|---|
| **Girdi** | Görüntü | 1 | - | 227x227 | - | - |
| **1** | Konvolüsyon | 96 | 55x55 | 11x11 | 4 | ReLU |
| **2** | Maksimum Ortaklama | 96 | 27x27 | 3x3 | 2 | ReLU |
| **3** | Konvolüsyon | 256 | 27x27 | 5x5 | 1 | ReLU |
| **4** | Maksimum Ortaklama | 256 | 13x13 | 3x3 | 2 | ReLU |
| **5** | Konvolüsyon | 384 | 13x13 | 3x3 | 1 | ReLU |
| **6** | Konvolüsyon | 384 | 13x13 | 3x3 | 1 | ReLU |
| **7** | Konvolüsyon | 256 | 13x13 | 3x3 | 1 | ReLU |
| **8** | Maksimum Ortaklama | 256 | 6x6 | 3x3 | 2 | ReLU |
| **9** | Tam bağlantı | - | - | 4096 | - | ReLU |
| **10** | Tam bağlantı | - | - | 4096 | - | ReLU |
| **Çıktı** | Softmax | - | - | 1000 | - | Softmax |

AlexNet Model Mimarisi 10 katmandan oluşmaktadır. Bu Derin Konvolüsyonel Sinir Ağı (DCNN)'de 5 adet konvolüsyon katmanı, 3 adet maksimum ortaklama katmanı, 2 adet tam bağlantı katmanından oluşur. Giriş katmanında yer alacak olan görüntü 227x227x3 boyutlarındadır. Son katmanda ise sınıflandırma yapılarak giriş görüntüsündeki sınıflandırma sayısı değeri verilir. AlexNet mimarisi, ImageNet veri tabanı için %80 doğruluğu yakalamış bir derin öğrenme mimarisidir. AlexNet, 60 milyon parametre ve 650.000 nöron içeren büyük bir ağ yapısıdır. Bu parametreleri eğitmek için Krizhevsky birçok iyileştirme yapmıştır. AlexNet ayrıca ReLU aktivasyon fonksiyonu ve Drop-out tekniğinin derin sinir ağlarında kullanımının öncüsü olmuştur (Krizhevsky vd., 2012).

### 2.4.3. *VGG-16*

VGG16 derin öğrenme mimarisi daha derin ağların daha iyi ağlar olduğu fikrine dayanarak tasarlanmıştır (Simonyan & Zisserman, 2014). AlexNet'e göre daha yüksek doğruluk performansı sağlasa da çok fazla parametresi olduğundan (yaklaşık 140 milyon) çok fazla bellek kullanım ihtiyacı olmuştur. Diğer yandan AlexNet'e göre daha küçük filtreler kullanılmıştır. Bu mimari tüm konvolüsyon katmanlarında değişken sayıda 64, 128, 256 filtre sayısı ile sabit 3x3 boyutlu filtreler kullanmaktadır.

VGG16, 13 konvolüsyon, 3 tam bağlı katmanından oluşan bir ağdır. Girdi katmanında yer alacak görüntü 224x224x3 boyutundadır. Son katman ise sınıflandırma katmanıdır. Tablo 2.3'de VGG16 Model Mimarisi gösterilmiştir.

VGG19 mimarisinde Tablo 2.3'deki VGG16 mimarisinden farklı olarak 5., 7. ve 10. adımlarda 4 kez konvolüsyon işlemi uygulanmıştır. Böylece, 16 konvolüsyon katmanı ve üç tam bağlantı katmanından oluşmaktadır (Simonyan & Zisserman, 2014).

**Tablo 2.3.** VGG16 model mimarisi

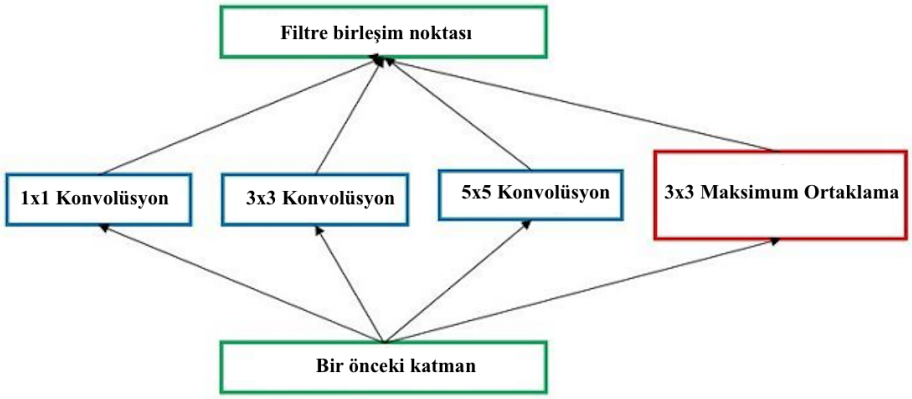
| Katman | Katman Tipi | Özellik Haritası | Filtre Boyutu | Girdi Boyutu | Adım Sayısı | Aktivasyon Fonksiyonu |
|---|---|---|---|---|---|---|
| **Girdi** | Görüntü | 1 | - | 224x224 | - | - |
| **1** | 2xKonvolüsyon | 64 | 224x224 | 3x3 | 1 | ReLU |
| | Maksimum Ortaklama | 64 | 112x112 | 3x3 | 2 | ReLU |
| **3** | 2xKonvolüsyon | 128 | 112x112 | 3x3 | 1 | ReLU |
| | Maksimum Ortaklama | 128 | 56x56 | 3x3 | 2 | ReLU |
| **5** | 3xKonvolüsyon | 256 | 56x56 | 3x3 | 1 | ReLU |
| | Maksimum Ortaklama | 256 | 28x28 | 3x3 | 2 | ReLU |
| **7** | 3xKonvolüsyon | 512 | 28x28 | 3x3 | 1 | ReLU |
| | Maksimum Ortaklama | 512 | 14x14 | 3x3 | 2 | ReLU |
| **10** | 3xKonvolüsyon | 512 | 14x14 | 3x3 | 1 | ReLU |
| | Maksimum Ortaklama | 512 | 7x7 | 3x3 | 2 | ReLU |
| **13** | Tam bağlantı | - | - | 25088 | - | ReLU |
| **14** | Tam bağlantı | - | - | 4096 | - | ReLU |
| **15** | Tam bağlantı | - | - | 4096 | - | ReLU |
| **Çıktı** | Softmax | - | - | 1000 | - | Softmax |

### 2.4.4. *Inception (GoogLeNet)*

ILSVRC 2014 yılının kazananı olan bu konvolüsyonel sinir ağı modeli Christian Szegedy ve arkadaşları tarafından geliştirilmiştir. GoogleNet yapısındaki Inception modüllerinden dolayı karmaşık bir mimaridir. GoogLeNet 22 katmanlı ve en iyi 5 test hata oranında (top-5 error) %6,7'lik bir değer elde etmiştir. VGG16 derin öğrenme algoritmasından daha hızlı olduğu görülmüştür. Daha önceki klasikleşmiş konvolüsyon ve ortaklama katmanlarının artarda dizilmesiyle oluşturulan modellerden farklı bir yaklaşım getirmiştir. Önceki modellerde görüldüğü gibi, her şeyin sıralı olarak gerçekleşmediği görülmektedir.

GoogleNet modelinde paralel olan ağ parçaları mevcuttur. Bu parçalar ya da modüller "inception" olarak adlandırılmıştır (Szegedy vd., 2015).

Temel Inception Modülü, ağ içinde ağ yapısı ile birlikte Google çalışanları, 2012 yılında gösterime giren 'Inception' filminden esinlenerek isimlendirdikleri Inception modüllerinin birleşiminden oluşan GoogLeNet'i yayınladılar (Szegedy vd., 2015). Inception modüllerindeki genişleme etkisi konvolüsyon katmanlarındaki (1×1), (3×3), (5×5) filtrelerin ve (3×3) maksimum ortaklama işleminin paralel olarak gerçekleştirilmesiyle oluşturulmaktadır. Bu yapı "Naive Inception module" olarak isimlendirilmektedir. Temel bir Inception modülü Şekil 2.21'de gösterilen diyagramdaki gibidir.

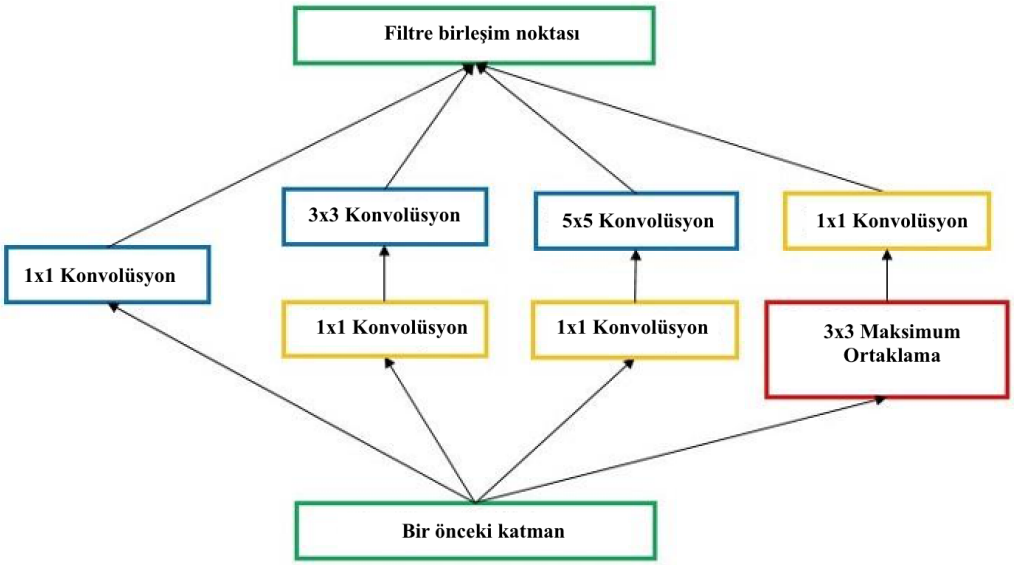



**Şekil 2.21.** Basit Inception modülü (Szegedy vd., 2015; Kızrak, 2020)

İşlem karmaşıklığı, çıkış boyutunun büyüklüğü ve parametre sayısı, paralel işlemler nedeniyle çok büyümektedir. Bu sorunun üstesinden gelebilmek için (1×1) konvolüsyon katmanları paralel Naive Inception konvolüsyon katmanlarının önüne eklenerek boyut azaltma işlemi gerçekleştirilmiştir. Bu Inception modüllerinin ilk versiyonunu oluşturmuştur. (1×1) konvolüsyon derinliği olmayan x ve y düzlemine sahip bir matriste çarpma etkisine sahip olsa da bu derinlik bilgisi olan bir tensör haline geldiğinde konvolüsyon işleminin parametre hesaplamada önemli derecede etkisi olmaktadır.

Giriş matrisi 100 kanallıysa ve buna 50 kanallı bir (1×1) konvolüsyon filtresi hacimsel olarak uygulandığında çıkış matrisinin kanal sayısı filtre sayısına eşit ve 50 olur. Bu durumda (1×1) konvolüsyon katmanı derinlikte boyut azaltmak demektir. Hesaplama karmaşıklığına ve büyüklüğüne bulunan bu çözüm beraberinde hız ve başarımı getirmiştir.

Inception ağ modeli modüllerden oluşmaktadır. Her bir modül, farklı boyutlu konvolüsyon ve maksimum ortaklama işlemlerinden meydana gelmiştir. Şekil 2.22'de GoogleNet modelinin içindeki bir Inception modülü yapısı sunulmuştur.




**Şekil 2.22.** GoogLeNet modelinin içindeki bir Inception modülü (Szegedy vd., 2015; Kızrak, 2020)

Inception modülü, (5×5) konvolüsyon işlemi için (28×28×32)× (5×5×192) = 120 milyon parametre hesabı yapmaktadır. Bunun gibi diğer konvolüsyon ve maksimum ortaklama katmanlarını da aynı şekilde hesaplamak gerekmektedir. Bu da beraberinde büyük bir işlem yükü getirir. Szegedy ve ekibi her konvolüsyon katmanından önce (1×1) konvolüsyon katmanı kullanılarak işlem yükünü azaltmaya çalışmıştır. Böylece daha karmaşık bir ağ modeli ile daha az hesap ve daha hızlı bir tasarım yapılmaktadır. Bu koşulda; (1×1) konvolüsyon katmanında: (28×28×16) × (1×1×192) = 2,4 milyon parametre ve (5×5) konvolüsyon katmanında: (28×28×32) × (5×5×16) = 10 milyon parametre olmak üzere toplamda 12,4 milyon parametre hesaplanmaktadır. İlk duruma göre yaklaşık 10 kat daha az parametre hesabı son derece çarpıcı bir etkiye sahiptir. Bu (1×1) konvolüsyon işlemi 'darboğaz' (bottleneck) olarak tanımlanmıştır. Bu işleme öznitelik ortaklama (pooling of feature) denir. Çünkü tensör derinliği yani katmanın kanal sayısı azaltılmaktadır. Aktivasyon fonksiyonu olarak ReLU kullanılmaktadır (Szegedy vd., 2015). Tüm ağ, 9 inception modülünden oluşmakta ve toplamda 100'den fazla katman bulunmaktadır. Tüm bağlantı katmanlarında ortalama ortaklama kullanılmaktadır. Böylece (7×7×1024) adet tensör (1×1×1024) boyutlu bir vektöre dönüştürülmektedir. Parametre maliyeti bakımından büyük avantaj sağlanmaktadır. GoogLeNet 5 milyon parametre ile AlexNet'e göre 12 kat daha az işlem yüküne sahiptir (Kızrak & Bolat, 2018).

AlexNet'ten 12 kat daha fazla parametre içeren bu mimarinin en büyük katkısı toplam 22 katman ile parametre sayısı 60 milyon olan AlexNet mimarisine karşılık

parametre sayısını 5 milyona düşürmesidir (yaklaşık 12 kat daha az parametre). Girdi katmanında yer alacak görüntü 224x224x3 boyutundadır. Konvolüsyon katmanında 1x1, 3x3 ve 5x5 boyutunda filtreler kullanılmaktadır. 3x3 boyutunda maksimum ortaklama kullanılmaktadır. Aktivasyon fonksiyonu olarak ReLU kullanılmaktadır. Inception modelinin dört farklı geliştirilmiş versiyonu bulunmaktadır. Bu tez çalışması kapsamında Inceptionv3 mimarisi oluşturulmuştur.

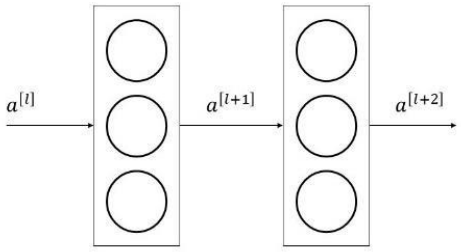
**Tablo 2.4.** Inceptionv3 model mimarisi

| | **Katman Tipi** | **Filtre Boyutu** | **Girdi Boyutu** | **Adım Sayısı** |
|---|---|---|---|---|
| **Girdi** | Görüntü | - | 299x299x3 | - |
| **1** | Konvolüsyon | 3x3 | 299x299x3 | 2 |
| **2** | Konvolüsyon | 3x3 | 149x149x32 | 1 |
| **3** | Konvolüsyon | 3x3 | 147x147x32 | 1 |
| **4** | Ortaklama | 3x3 | 147x147x64 | 2 |
| **5** | Konvolüsyon | 3x3 | 73x73x64 | 1 |
| **6** | Konvolüsyon | 3x3 | 71x71x80 | 2 |
| **7** | Konvolüsyon | 3x3 | 35x35x192 | 1 |
| **8** | Inception modülü A | 3 modül | 35x35x288 | - |
| **9** | Inception modülü B | 4 modül | 17x17x768 | - |
| **10** | Inception modülü C | 2 modül | 8x8x1280 | - |
| **11** | Ortaklama | 8x8 | 8x8x2048 | - |
| **12** | Linear | Lojit | 1x1x2048 | - |
| **Çıktı** | Softmax | 1000 | 1x1x1000 | - |

### 2.4.5. *ResNet*

Daha önceki modellerden farklı olarak artık değerlerin (Residual value) sonraki katmanları beslediği blokların (Residual Block) modele eklenmesiyle oluşan ve yeni bir yaklaşım getiren ResNet mimarisi, 18, 34, 50, 101 ya da 152 katmandan oluşan bir konvolüsyonel sinir ağı modelidir.

ImageNet veri seti için 2015 yılında %3,57'lik bir hata oranı elde eden ILSVRC 2015 yılının kazananı olan bu derin konvolüsyonel sinir ağı modeli, Microsoft Asya Araştırma ekibi tarafından geliştirilmiştir (He, Zhang, Ren & Sun, 2016). Şekil 2.23'de ResNet blok modülü gösterilmiştir.




**Şekil 2.23.** ResNet blok modülü (Kızrak, 2018)

Doğrusal ve ReLU aktivasyon fonksiyonlarına sahip katmanların arasında iki katmanda bir eklenen artık değer Eşitlik 2.42, 2.43, 2.44 ve 2.45'de belirtildiği gibi hesaplama işlemine dahil edilmiştir. Önceki katmandan gelen $a^{[l]}$ değeri, $a^{[l+2]}$ değerine eklenmektedir.

$$z^{[l]} = W^{[l+1]}a^{[l]} + b^{[l+1]} \tag{2.42}$$

$$a^{[l+1]} = g(z^{[l+1]}) \tag{2.43}$$

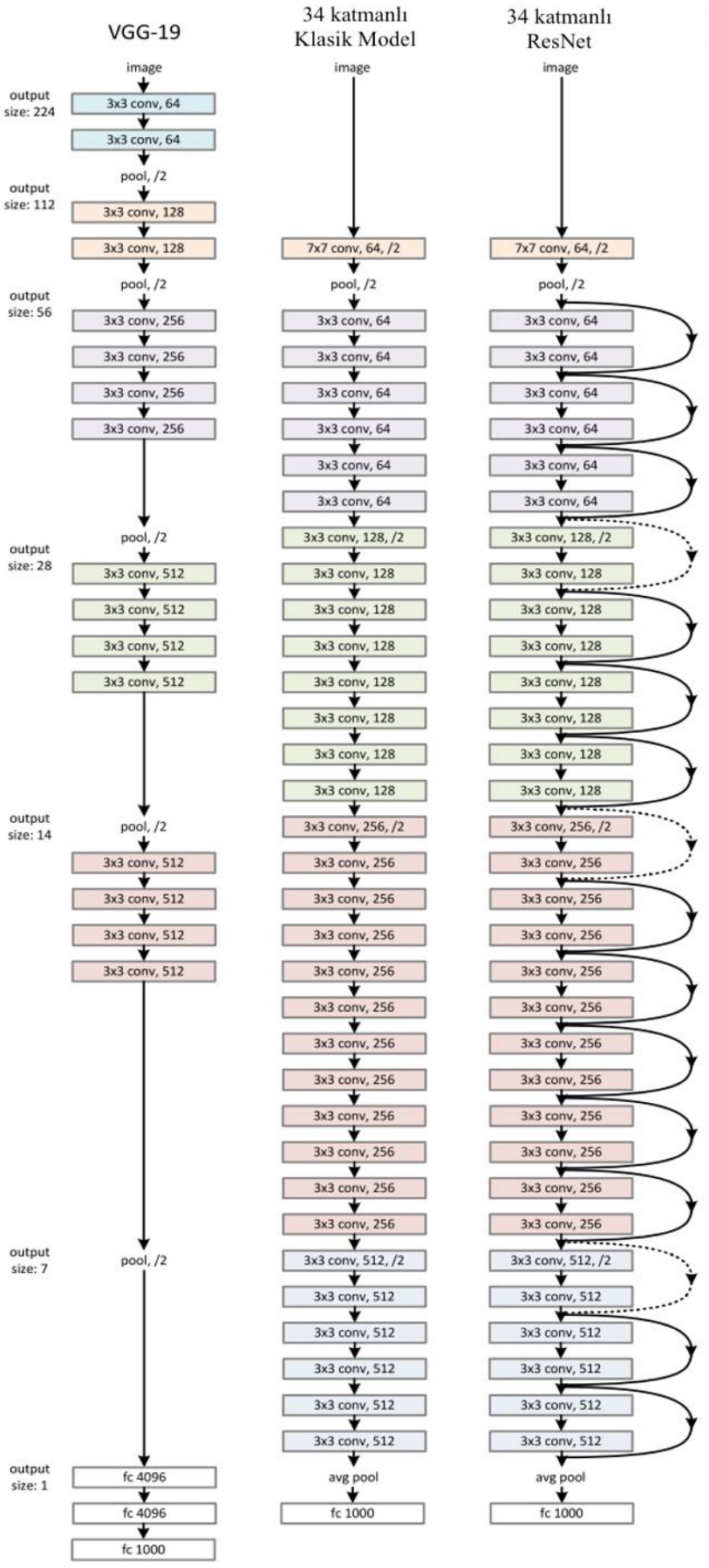
$$z^{[l+2]} = W^{[l+2]}a^{[l+2]} + b^{[l+2\}} \tag{2.44}$$

$$a^{[l+2]} = g(z^{[l+2]}) \tag{2.45}$$

Derin öğrenme üzerine araştırma yapan uzmanlar derin konvolüsyonel sinir ağı mimarisi oluştururken katman sayısından kaynaklı problemler ile karşılaşmışlardır. Önceden tasarlanan derin öğrenme mimarilerinde katmanlar eklendikçe belirli bir seviyeye kadar mimarilerin performansları artarken bir noktadan sonra performanslarında hızlı bir düşüş görülmüştür.

Görüntü tanımadaki en son teknolojilerden biri olan ResNet mimarisi önceki model mimarileri gibi "ağ ne kadar derin olursa performans o kadar artar" fikri üzerine kurulmuştur. Ancak artan ağ derinliği ile birlikte, her katmanın gradyanı zincir kuralı ile hesaplandığından katman sayısı arttıkça gradyan değerleri küçülmüş, sıfıra yaklaşmıştır ve gradyanların yok olması (vanishing gradient) problemi artmıştır.

Artık değer (Residual value) beslemesi, yeni çıkış eşitliğini iki önceki katmandan gelen $a^{[l]}$ değeri o anki ağırlık sıfır olsa bile öğrenme hatasını optimize etmiştir ve ağ daha hızlı eğitilmiştir. Şekil 2.24'de VGG19 modeli, 34 katmandan oluşan klasik model ve 34 katmanlı ResNet Model Mimarisi yer almaktadır (He vd., 2016). Bu tez çalışmasında 50 katmanlı ResNet Modeli Mimarisi oluşturulmuştur.




**Şekil 2.24.** VGG19, Klasik model ve ResNet mimarisi (He vd., 2016)

# 3. GEREÇ VE YÖNTEMLER

## 3.1. Fundus Görüntülerinin Elde Edilmesi

Bu tez çalışmasında kullanılan fundus görüntüleri, Pekin Üniversitesi'nin sponsor olduğu Oküler Hastalık Akıllı Tanıma (Ocular Disease Intelligent Recognition-ODIR) üzerine gerçekleştirilen uluslararası bir yarışmadan elde edilmiştir. Veri kümesi, Çin'deki 26 farklı şehirden 487 hastaneden Shanggong Medical Technology Co. Ltd. tarafından toplanan "gerçek" hasta verileridir.

Her bir fundus görüntüsü, çeşitli hastalıkların neden olduğu fundusun farklı bölümlerinde yer alan anomalilikler içermektedir. Bu farklı bölümlerin arasında makula, optik kap, optik disk, kan damarları ve tüm fundus arka planı yer almaktadır. Her bir görüntüyü etiketlemek için belirli bir hastalık kategorisinin daha ayrıntılı bir sınıflandırması tek bir kategoride toplanmıştır. Örneğin 1, 2, 3 ve 4. seviye Diyabetik Retinopatisi (D), tek bir kategori olarak birleştirilmiştir. Fundus görüntülerinin yüksek kalitede olmasını sağlamak için, özel veri tabanının görüntülerini, yinelenen görüntüleri ve düşük kaliteli görüntüler filtrelenerek temizlenmiştir. Daha sonra kalan fundus görüntülerinden eğitim ve test için uygun oranda örnekler seçilmiştir. Son olarak, ODIR veri seti, 5.000 hastanın sağ ve sol gözlerinden 8 sınıf ve ek açıklama içeren 10.000 fundus görüntüsünden oluşur. Etiketli kategoriler arasında Normal (N), Diyabetik Retinopati (D), Glokom (G), Katarakt (C), Yaşa bağlı Makula Dejenerasyonu (A), Hipertansif Retinopati (H), Miyop (M) ve diğer hastalıklar/anomalilikler (O) bulunmaktadır. Bilgisayar destekli oküler hastalık tanıma algoritmalarını değerlendirmek için veri seti eğitim, doğrulama ve test seti olmak üzere sırası ile 3500, 500 ve 1000 hastaya ait sol ve sağ fundus görüntüleri içermektedir. Eğitim seti ile test setinin oranı 7:3'tür. Derin konvolüsyonel sinir ağları modelini eğitmek için eğitim seti, model seçimi için doğrulama seti, genelleştirilme yeteneği ise, test seti ile değerlendirilmiştir.

### 3.1.1. *Fundus görüntülerine ait ek açıklamalar*

Fundus görüntülerine ait hastalıkların etiketlenmesi işleminin tamamlanması yaklaşık 10 ay süren profesyonel açıklama personeli ve hakem ekibi tarafından yapılmıştır. Açıklama ve hakem ekibi, iki yıldan fazla klinik deneyime sahip üç oftalmolog ve oftalmolojide on yıldan fazla klinik deneyime sahip üç oftalmologdan oluşmaktadır. Veri açıklama sürecinde ilgili standardizasyon ve prosedürler takip edilmiştir. İlk olarak, açıklama ekibi sırasıyla aynı grup fundus görüntülerine açıklama eklemiştir ve sonuçları kaydetmiştir. Bu ekip arasında herhangi bir anlaşmazlık söz konusu olduğu durumda, hakem ekibi nihai sonucu belirlemiştir. Bu sonuç iki veya daha fazla uzman hakemin fikir birliğine dayanmıştır. ODIR veri seti, Binoküler fundus görüntüsüne dayalı olarak uluslararası alanda başlatılan ilk büyük ölçekli çoklu tip hastalık tespit veri setidir. Aynı alandaki diğer fundus görüntü veri kümeleriyle karşılaştırıldığında, ODIR veri kümesi aşağıda açıklanan önemli özelliklere sahiptir.

**1. Çoklu Hastalık:** Yalnızca bir oftalmik hastalığa odaklanan mevcut fundus görüntü veri setlerinin çoğunun aksine, ODIR görüntü kümesi birden fazla oftalmolojik hastalık içermektedir. Bu hastalıklar fundusun farklı bölgelerindeki lezyonlarla birlikte anormallikleri içerir. Uluslararası Sınıflandırma Standardı ICO' ya (Wong vd., 2017) göre D, dört aşamaya ayrılmıştır. D'nin erken evresi retinadaki çeşitli anomaliliklerle karakterizedir. Örneğin fundus görüntüsünde sert eksuda, yumuşak eksuda, kanama ve neovaskülarizasyon gibi lezyonlar vardır. Gelişmiş D ile karşılaştırıldığında, erken evre klinik tedavi için ciddi ve önemlidir (Sengupta, Singh, Leopold, Gulati & Lakshminarayanan, 2020). Glokom için, göz doktoru genellikle optik kabın optik diske oranını hesaplar. Oran 0.5'ten büyük olduğunda, hastanın glokomu olduğuna karar verilir. Son yıllarda, oftalmologlar ayrıca nöroretinal kenar kaybı, görme alanı ve retina sinir lifi tabakasındaki bozukluk teşhisi koymaktadır. Klinik olarak, Amerikan Kooperatif Katarakt Araştırma Grubu (CCRG) yöntemi gibi protokoller genellikle kataraktı sınıflandırmak için kullanılır. Deneyimli oftalmologlar, hastaların katarakt şiddetini belirlemek için fundus görüntülerini standart katarakt resimleri ile karşılaştırırlar (Zhou, Li, & Li, 2019).

Günümüzde A, ileri yaştaki insanlarda da yaygın bir körlük nedenidir ve açık bir şekilde maküler alandaki drusen ile ilişkilidir (Floriano, Ferreira, Camacho, & Marquez, 2019). Oftalmologlar, A hastalık sınıfının ciddiyetini drusenlerin boyutuna ve sayısına göre teşhis eder.

Hipertansiyonlu bir hastanın fundus görüntüsünde, arteriyovenöz çapın daha büyük bir orana sahip olduğunu görebiliriz. Patolojik miyopi için hastanın fundus bölgesinde net leopar baskı şekilleri vardır. Oftalmoloji araştırmalarında, mevcut veri setlerinin çoğu fundus hastalığına dayanmaktadır. Bu durum fundus hastalığına dayalı ilgili çalışmanın diğer hastalıkların tespitine uygulanmasını zorlaştırmaktadır. Klinik olarak oftalmologlar bir fundus görüntüsünü gözlemleyerek çeşitli hastalıkların tanı sonuçlarını verebilirler. Bu nedenle, birden fazla fundus hastalığı içeren bir veri seti olarak ODIR, klinik uygulama senaryolarına daha yakındır.

**2. Binoküler-yapı:** Mevcut oftalmolojik hastalık tespit çalışmalarının çoğu fundus görüntüsüne dayalıdır. Ancak gerçek klinik senaryolarda oftalmologlar genellikle hastaları her iki gözden gelen bilgilerle teşhis eder. Veri seti üzerinde yapılan ilgili çalışmaların gerçekçi sahnelere daha iyi uygulanabilmesi için veri seti hastaların sol ve sağ gözlerinin fundus görüntülerini içermektedir. Yalnızca tek bir göz fundus görüntüsü tespiti ile karşılaştırıldığında, her iki göz fundus görüntüsü olan hastalar için tarama hem kapsamlı hem de karmaşıktır. Çünkü görüntünün öznitelik çıkarma sürecinde, iki göz ve ilgili özellikleri arasındaki korelasyonu dengelemeliyiz. Bu, bu veri kümesindeki sınıflandırma görevini zorluklarla dolu hale getirir. Veri setinin nihai amacı, fundus görüntülerini kullanarak hastaların çok etiketli sınıflandırmasını yapmaktır.

**3. Çok modlu veri (Multi-Modal data):** Veri seti, hastaların çoklu bilgilerini entegre etmiştir. Araştırmacılara hastaların sol ve sağ gözlerinin fundus görüntülerini sağlamanın yanı sıra, her hastanın yaşı ve cinsiyeti ile her fundus görüntüsü için göz doktorlarının tanısal anahtar kelimelerini sağlamıştır. Bu bilgiler, araştırmacıların çoklu hastalık sınıflandırma görevlerini daha iyi gerçekleştirmelerine yardımcı olabilir ve ayrıca fundus görüntülerine dayalı metinsel tanı oluşturma (Cheleramani vd., 2019) ve yaş tahmini (Poplin vd., 2018) gibi veri setine dayalı daha ayrıntılı araştırmalar yapmalarına yardımcı olabilir.

**4. Ölçek (Scale):** Günümüzün veriye dayalı derin öğrenme araştırmasında, büyük ölçekli veri kümeleri, bir araştırmanın gerçek dünya senaryolarına doğru olarak uygulanmasını sağlamanın temel yapı taşıdır. ODIR veri seti, 5.000 klinik hastanın sol ve sağ gözlerinden alınan 10.000 fundus görüntüsünü içermektedir. Bu görüntüler, Çin'in çeşitli bölgelerindeki çok sayıda oftalmik merkezde farklı kameralar (Canon, Zeiss, Kowa, vb.) tarafından elde edilmiştir.

ODIR fundus görüntü kümesi için özel açıklamalar:

**1.** Açıklamalı sınıflandırma etiketleri aşağıdaki kurallara göre belirlenmiştir.

**a)** Bir hastanın sınıflandırma etiketleri, sol ve sağ fundus görüntülerine ve ilgili tanı anahtar kelimelerine bağlıdır.

**b)** Bir hasta hem sol hem de sağ tanı anahtar kelimelerinin "normal fundus" olması durumunda normal olarak sınıflandırılır.

**c)** Sınıflandırma etiketi, fundus görüntülerinden biri "normal fundus" olarak belirlenmiş ise diğer fundus görüntüsü tarafından sınıflandırma yapılmıştır.

**d)** Tüm şüpheli hastalıkları veya anomalilikleri diğer teşhis edilmiş hastalıklar veya anomalilikler olarak değerlendirilir.

**2.** Teşhis anahtar kelimelerinde görülen özel kelimeler

**a)** "Ön segment görüntüsü" ve "fundus görüntüsü yok" gibi iki tanı anahtar kelimesi, bu yarışmada sekiz kategoriden hiçbirine sınıflandırılmamıştır. Örneğin, 1706_left.jpg ve 1710_right.jpg isimli iki ön segment görüntüsü vardır. Bu durumda, hastanın sınıflandırma etiketleri yalnızca aynı hastanın diğer fundus görüntüsü ile değerlendirilmiştir. Ayrıca, 4580_left.jpg görüntüsü için tanı anahtar kelimesinin "fundus resmi yok" olduğuna dikkat edilmelidir. Bu görüntü aslında bu hastanın sol fundus görüntüsü olmadığı için, sağ fundus görüntüsünün dönüştürülmesi ile oluşturulmuştur.

**b)** "Mercek tozu (lens dust)", "Optik disk fotografik olarak görülmez (optic disk photographically invisible)", "Düşük görüntü kalitesi (low image quality)" ve "Görüntü kayması (image offset)" tanı kelimeleri sınıflandırmada belirleyici bir rol oynamamıştır.

**c)** Bazı görüntülerin arka planı diğerlerinden oldukça farklıdır. Bu görüntülerin ön işlenmiş olduğu bilinmektedir. Modelde bu görüntülerin eğitilip eğitilmemesi araştırmacıya bırakılmıştır. Bu görüntüler:

2174_right.jpg; 2175_left.jpg; 2176_left.jpg; 2177_left.jpg; 2177_right.jpg; 2178_right.jpg; 2179_left.jpg; 2179_right.jpg; 2180_left.jpg; 2180_right.jpg; 2181_left.jpg; 2181_right.jpg; 2182_left.jpg; 2182_right.jpg; 2957_left.jpg; 2957_right.jpg.

## 3.2. Python Geliştirme Ortamı ve Kullanılan Kütüphaneler

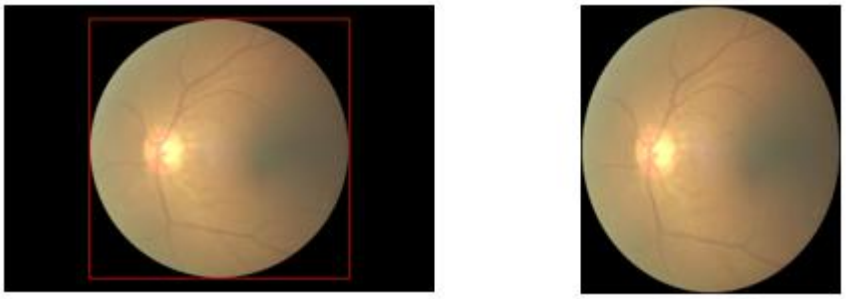
Bu tez çalışması için oluşturulan programlama kodları, Python Programlama Dili (Version: 3.6.13) ile Visual Studio Code (Version: 1.54.3) geliştirme ortamı ile gerçekleştirilmiştir. Oluşturulan Derin Öğrenme Mimarisi Modelleri eğitiminde donanım aracı olarak Google'ın geliştiricilere sunduğu bulut servisi olan Google Colaboratory (Colab-Pro) platformunda, NVIDIA GPU (Grafik İşleme Birimi) işlemcisine sahip Tesla P100 işlemcili makine kullanılmıştır. NVIDIA Tesla-P100 GPU hızlandırıcı kullanılarak yapılan çalışma için yaklaşık 25 GB (Giga Byte) rastgele erişilebilir bellek (Random Access Memory – RAM) kullanılmıştır. Her bir epokta eğitim, doğrulama ve test setine ait modellerin ve sonuçların kaydedilmesi için 100 GB depolama alanı kullanılmıştır.

Tez Çalışması için Kullanılan Python Kütüphaneleri:

- PyTorch-3.9 ve yan kurulumları torch-1.3.1, torchvision-0.4.2
- Matplotlib-3.1.1
- SciKit-Learn-0.21.3
- Pillow-7.1.2
- NumPy-1.18.4
- python-csv-0.0.13
- tqdm-4.46.0
- requests-2.23.0
- xlrd-1.2.0
- Pandas-0.25.1
- Augmentor-0.2.8

## 3.3. Veri Ön İşleme

ODIR fundus görüntü setinde yer alan fundus görüntüleri ön işlemlerden geçirilmiştir. Şekil 3.1'de görüldüğü gibi öncelikle renkli piksellerin bulunduğu koordinatlar belirlenerek bir çerçeve oluşturulup bu çerçeve kesilerek yeni bir eğitim, doğrulama ve test görüntü seti oluşturulmuştur. Çerçeve dışında kalan siyah piksel alan böylece eğitime dahil edilmemiştir. Bu işlemler için pre_preprocess.py ve pre_preprocess_image_crop.py uzantılı dosya oluşturulmuştur.



a) İşlem öncesi (663_left.jpg) b) İşlem sonrası (663_left.jpg)

**Şekil 3.1.** Renkli piksellerin koordinat alanı ile seçilmesi

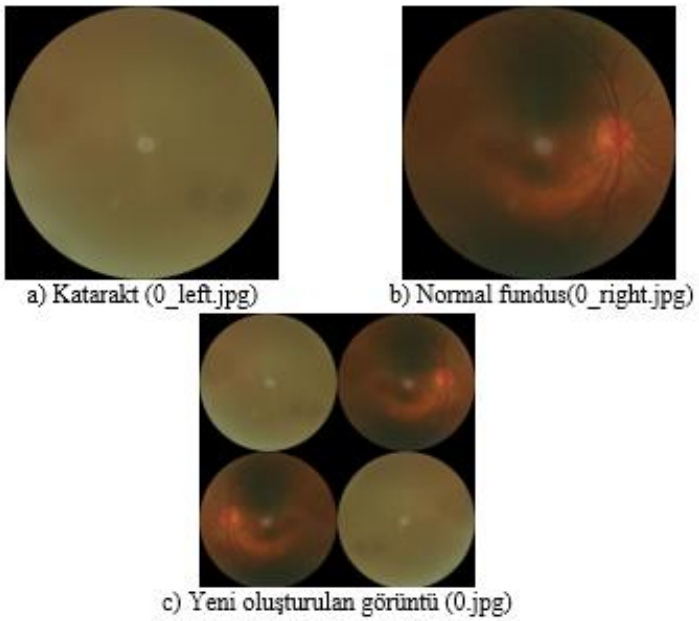
Bölüm 3.1.1'de belirtilen ODIR fundus görüntü setine ait özel açıklamalarda yer alan ikinci maddede belirtilen teşhis anahtar kelimelerinde görülen özel kelimeler incelendiğinde, a, b ve c alt maddelerinde belirtilen özelliğe sahip tüm görüntüler, eğitim setinde bulunan 3500 hastanın 302'si, doğrulama setinde 500 hastanın 29'u, test setinde ise 1000 hastanın 70'i çalışmaya dahil edilmemiştir. Böylece eğitim, doğrulama ve test seti dağılımı, sırası ile 3198, 930 ve 471 hastaya ait fundus görüntüsünden oluşmaktadır. Eğitim, doğrulama ve test setine ait her bir hastanın sağ ve sol fundus teşhis anahtar kelimelerinin yer aldığı excel formatındaki dosya csv uzantılı dosyaya dönüştürülerek okunması işlemi gerçekleştirildi. Daha sonra, sağ ve sol fundus teşhis anahtar kelimeleri bir sözlük oluşturularak toplam 117 farklı etiket olduğu belirlendi.

Bu işlemler için utils.py, preprocess_annotations.py uzantılı dosyalar oluşturulmuştur. Eğitim, doğrulama ve test fundus görüntülerinin sekiz sınıflandırma kategorisine dağılımı Tablo 3.1'de verilmiştir.

**Tablo 3.1.** ODIR veri setinde yer alan eğitim, doğrulama ve test fundus görüntülerinin sekiz sınıflandırma kategorisine dağılımı

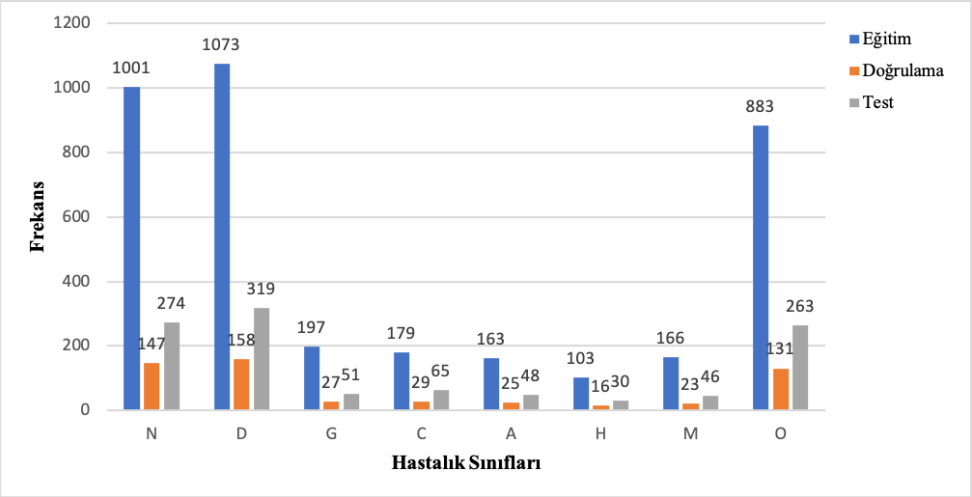
| **Sınıf** | **Eğitim (1/0)** | **Doğrulama (1/0)** | **Test (1/0)** | **Toplam (1/0)** |
|---|---|---|---|---|
| N | 1001/2197 | 147/324 | 274/656 | 1422/3177 |
| D | 1073/2125 | 158/313 | 319/611 | 1550/3049 |
| G | 197/3001 | 27/444 | 51/879 | 275/4324 |
| C | 179/3019 | 29/442 | 65/865 | 273/4326 |
| A | 163/3035 | 25/446 | 48/882 | 236/4363 |
| H | 103/3095 | 16/455 | 30/900 | 149/4450 |
| M | 166/3032 | 23/448 | 46/884 | 235/4364 |
| O | 883/2315 | 131/340 | 263/667 | 1277/3322 |
| **Toplam** | 3198 | 471 | 930 | 4599 |

Hastaların sol ve sağ fundus görüntü piksel genişlikleri yatay eksende toplandı. Birleştirilen sol ve sağ fundus görüntüleri aynalama işlemi yapıldı. Yatay eksende piksel genişlikleri toplanan görüntülerin yüksekliklerine dikey eksende aynalama işlemi sonrası elde edilen görüntülerinin yüksekliği eklenerek Şekil 3.2'deki gibi elde edilmiştir. Bu işlem utils.py uzantılı dosya ile oluşturulmuştur.




**Şekil 3.2.** Bir hastanın sol ve sağ fundus görüntülerinin birleştirilmesi ve aynalama işlemi sonrası elde edilen görüntü

Eğitim, doğrulama ve test setinde yer alan her bir hastalık sınıfına yeni bir dizin oluşturma işlemi gerçekleştirilmiştir. Hastalık sınıflarına ait dizin oluşturulduktan sonra hastalık yok "0" ve hastalık var "1" olmak üzere iki alt dizin oluşturulmuştur. Bu dizinlere hastalık sınıfını içeren tüm yeni oluşturulan görüntüler eklenmiştir. Bu işlemler için collect_dataset.py uzantılı dosya oluşturulmuştur.

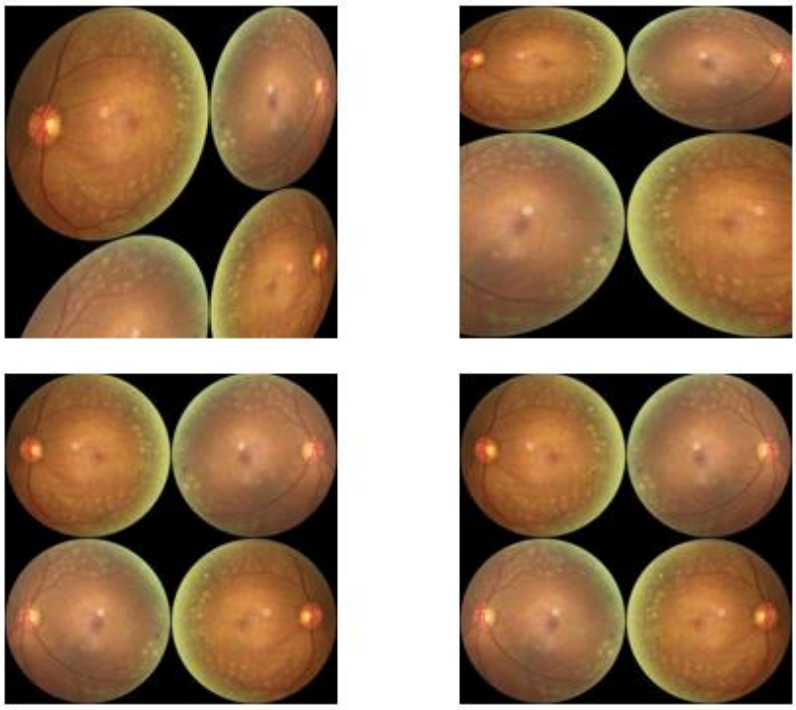



**Şekil 3.3.** Eğitim, doğrulama ve test seti fundus görüntülerinin hastalık sınıflarına göre frekans dağılımı

Şekil 3.3 incelendiğinde Glokom, Katarakt, Yaşa bağlı makula dejenerasyonu, Hipertansiyon, Miyop hastalık sınıflarının sıklığının çok az olduğu görülmektedir. Sınıf dağılımı dengesizliğinin ortadan kaldırılması için eğitim setinde yer alan her bir hastalık sınıfına ait fundus görüntülerinin arttırılması (augmentation) işlemi gerçekleştirilmiştir.

Fundus görüntülerinin arttırılmasında yeni oluşturulan dizinlerde her bir hastalık sınıfı kategorisinde hastalık yok "0" ve hastalık var "1" olmak üzere her bir alt kategoride eşit sayıda olmak üzere 10000 adet yeni fundus görüntüsü elde edilmiştir. Öncelikle görüntü yüksekliği ve genişliğindeki bozulma derecesi 3, 0.25 olasılık ile rasgele bozulma ve gaussian bozulma işlemi, daha sonra 0.05 olasılık ile eğim, sol ve sağ, alt ve üst, köşe bölümlerinden eğim işlemleri Augmentor paketi, random_distortion, gaussian_distortion, skew, skew_tilt, skew_left_righ, skew_top_bottom ve skew_corner fonksiyonları ile gerçekleştirilmiştir. Bu işlemler için augment.py uzantılı dosya oluşturulmuştur.

Glokom hastalık sınıfına ait 30. hastaya ait görüntü artırma işlemi sonrası bir örnek gösterim Şekil 3.4'de yer almaktadır.



**Şekil 3.4.** Glokom hastalık sınıfına ait 30.hastanın sol ve sağ fundus görüntülerinin arttırılması işlemi

Görüntü ön işleme ve veri artırma adımları sonrasında VGG16, Inceptionv3 ve ResNet50 model mimarileri oluşturulmuştur. Bu modellerin giriş katmanlarına hazırlanan fundus görüntüleri verilmiştir. Her bir oküler hastalık sınıfı için üç farklı model olmak üzere toplam 24 model mimarisi eğitimi gerçekleştirilmiştir.

## 3.4. Derin Öğrenme Mimarisi

Ön-eğitimli ağlar CNN modellerinde öğrenmenin daha hızlı sağlanması, işlemlerin daha hızlı yapılması ve doğruluk başarımının artırılması için oldukça önemlidir. Bu ağlar benzer bir problem üzerinde önceden kullanılarak buna bağlı ağırlıklar alan modeller olarak bilinir. Bu ağlar, ImageNet görsel nesne tanıma üzerine kurulmuş çok büyük bir veri tabanı üzerinde önceden kullanılmıştır.

Bu tez çalışmasında, ODIR eğitim setinin sol ve sağ fundus görüntüleri, veri ön işleme adımı sonrası transfer öğrenme yöntemi ile ImageNet görüntü seti üzerinde elde edilen ön eğitimli model ağırlıkları kullanılarak her bir oküler hastalık sınıfı için VGG16, Inceptionv3 ve ResNet50 model mimarileri oluşturuldu. Oluşturulan model mimarilerine ince ayar (fine tuning) yöntemi uygulandı.

Veri ön işleme adımı sonrası elde edilen giriş görüntüleri VGG16 ve ResNet50 mimarileri için 224 × 224 × 3, Inceptionv3 için 299 × 299 × 3 olarak modellerin veri giriş katmanlarında kullanılan boyutlara uygun hale getirildi. Bu model mimarilerinde 0,001 öğrenme oranı, değerler $\beta_1$ için 0.9, $\beta_2$ için 0.999 ve $\varepsilon$ için $10^{-8}$ olarak belirlendi ve Adam (Adaptif Moment) optimizasyon algoritması kullanıldı.

Modellerin eğitimi sırasında eğitim hatası ile doğrulama hatası arasındaki fark büyüdüğünde model ezberleme sorunu oluştuğu bilinmektedir. Bu problemden kaçınmak için erken durdurma işlemi uygulanmıştır. Adım sayısı yedi epok olarak belirlenmiştir. Yedi epok boyunca doğrulama seti doğruluk değeri en iyi değerinden daha yüksek değere ulaşamadığı durumda model eğitimi durdurulmuştur.

Oxford Üniversitesi Görsel Geometri Grubu (Visual Geometry Group) tarafından geliştirilen VGG16 derin öğrenme mimarisi, 2014 yılında ILSVRC'de görüntü sınıflandırmada ikinci ve yerelleştirme kategorisinde birinci olmuştur. VGG16 model mimarisi on üç konvolüsyon katmanı, üç tam bağlantı katmanı dahil olmak üzere 16 ağırlık katmanı kullanır. Bu tez çalışmasında, VGG16 model mimarisinin iki kategorili sınıflandırma yapabilmesi için daha önce nesne sınıflandırılmasında 1000 sınıf olan tam bağlantı katmanındaki sınıf sayısı, 2 sınıfa uyarlandı. Ön eğitimli katman ağırlıkları kullanarak mimarinin son üç tam bağlantı katmanı dışındaki tüm katmanlar donduruldu. Mimari oküler hastalıkların sınıflandırılması için ODIR fundus görüntü setinde ince ayar (fine-tuning) yöntemi ile eğitildi.

ILSVRC 2014 yılının kazananı olan Inceptionv3 modeli nesne tanımlama için oluşturan ve eğitilen bir derin öğrenme mimarisidir. Bu mimari konvolüsyonel sinir ağları, havuzlama katmanları ve tam bağlantı katmanlarından oluşur. Bu katmanlar inception-v3 mimarisinin görüntülerin özelliklerinin çıkarımına ve nesne tanımlamasına olanak sağlar. Bu tez çalışmasında, inceptionv3 model mimarisinin iki kategorili sınıflandırma yapabilmesi için daha önce nesne sınıflandırılmasında 1000 sınıf olan tam bağlantı katmanındaki sınıf sayısı, 2 sınıfa uyarlandı. Inceptionv3 mimarisinin 2. Inception C modülü, InceptionAux, Inception D, Inception E ve çıktı olan tam bağlı katman dışındaki tüm katmanlar dondurulduğu. Mimari oküler hastalıkların sınıflandırılması için ODIR fundus görüntü setinde ince ayar (fine-tuning) yöntemi ile eğitildi.

Microsoft araştırmacıları tarafından geliştirilen ResNet50 modeli, artık ağ (Residual Network) mimarileri 2015 yılında ILSVRC'de görüntü sınıflandırmada, nesne tespitinde ve yerelleştirilmesi alt başlıklarında birinci olmuştur. Resnet50 mimarisi 50 katmandan oluşan artık ağ mimarisidir. Bu tez çalışmasında öncelikle Resnet50 mimarisinin iki kategorili sınıflandırma yapabilmesi için daha önce nesne sınıflandırılmasında 1000 sınıf olan tam bağlantı katmanındaki sınıf sayısı, 2 sınıfa uyarlandı. Ön eğitimli katman ağırlıkları kullanarak mimarinin yedinci katmanına kadar olan ağırlıkları dondurduldu. Mimari oküler hastalıkların sınıflandırılması için ODIR fundus görüntü setinde ince ayar (fine-tuning) yöntemi ile eğitildi.

## 3.5. Performans Değerlendirme Ölçütlerinin Belirlenmesi

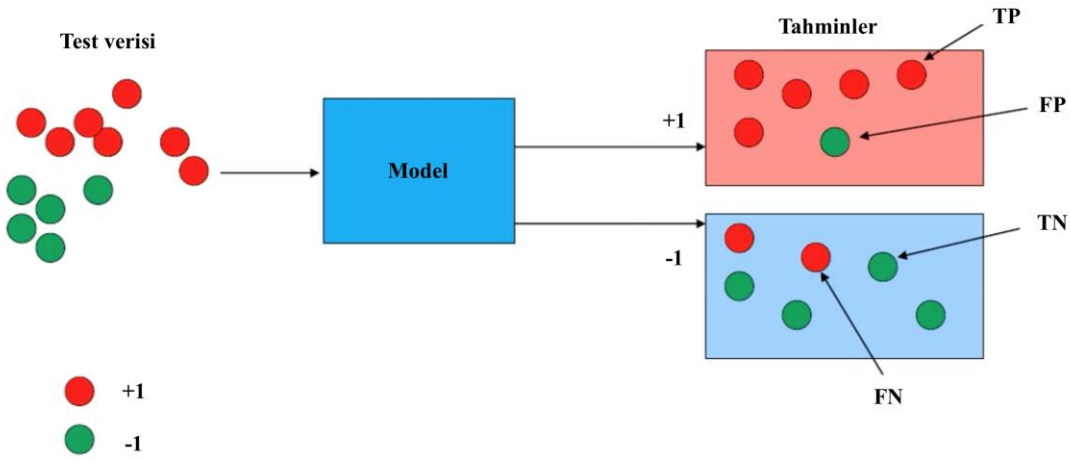
Modellerin sınıflandırma performanslarının karşılaştırılabilmesi için öncelikle ikili sınıflandırma için oluşturulan Tablo 3.2'de karmaşıklık matrisinde (Confusion Matrix) bulunan Gerçek Pozitif (True Positive-TP), Yanlış Pozitif (False Positive-FP), Gerçek Negatif (True Negative-TN) ve Yanlış Negatif (False Negative-FN) değerleri hesaplanır ve karmaşıklık matrisi elde edilir.

**Tablo 3.2.** Karmaşıklık Matrisi (Confusion Matrix)

| | | **Kestirim Sınıfı (Beklenen)** | |
|---|---|---|---|
| | | **Pozitif** | **Negatif** |
| **Gerçek Sınıf (Gözlem)** | **Pozitif** | TP (True Positive) Gerçek Pozitif | FN (False Negative) Yanlış Negatif |
| | **Negatif** | FP (False Positive) Yanlış Pozitif | TN (True Negative) Gerçek Negatif |

Tablo 3.2'deki karmaşıklık matrisi elde edilebilmesi için her bir hastalık sınıfına ait Gerçek Pozitif (True Positive-TP), Yanlış Pozitif (False Positive-FP), Gerçek Negatif (True Negative-TN) ve Yanlış Negatif (False Negative-FN) değerlerini ifade etmektedir.

Şekil 3.5'de örnek bir test verisinden model sınıflandırma sonucunda tahmin edilen değerler göz önüne alınarak TP, FP, TN ve FN değerlerinin elde edilmesine dair görsel bir sınıflandırma örneği yer almaktadır.




**Şekil 3.5.** Sınıflandırma sonucu TP, FP, TN ve FN değerlerinin elde edilmesi (Kızrak, 2019)

Karmaşıklık matrisi elde edildikten sonra doğruluk (Accuracy), kesinlik (Precision), duyarlılık (Recall, Sensitivity), özgüllük (Specificity) değerleri elde edilir.

Modellerin sınıflandırma performansını değerlendirmek için Kappa, F1 skoru (F1), ROC (Receiver Operating Characteristic Curves) eğrisi altında kalan alan (AUC) ve bu üç ölçütün ortalaması ile Final skoru dahil olmak üzere dört değerlendirme ölçütü kullanılmıştır.

Kappa katsayısı iki değerlendirici arası karşılaştırmalı uyumun güvenilirliğini ölçen bir istatistiksel yöntemdir ve -1 ile 1 arasında değişmektedir. $p_0$, gözlenen uyumluluk oranı, $p_e$ şansa bağlı ya da tesadüfi uyum oranı olmak üzere, Kappa istatistiği hesaplanır.

F1 ölçütü, kesinlik ve duyarlılık oranlarının her ikisi de yüksek olduğunda yüksek olan duyarlılık ve kesinlik değerlerinin harmonik ortalamasıdır.

Eğri altında kalan alan AUC, 1'e ne kadar yakınsa, modelin sınıflandırma performansı o kadar iyi olur. Genellikle modelin kararlılığını ölçmek için kullanılır. Bu dört değerlendirme ölçütü sklearn paketi kullanılarak hesaplanmıştır. Bu değerlendirme ölçütleri Eşitlik 3.1-3.8'de verilmiştir.

$$\text{Doğruluk (Accuracy)} = \frac{TP+TN}{TP+TN+FP+FN} \quad (3.1)$$

$$\text{Kesinlik (Precision)} = \frac{TP}{TP+FP} \quad (3.2)$$

$$\text{Duyarlılık (Sensitivity, Recall)} = \frac{TP}{TP+FN} \quad (3.3)$$

$$\text{Özgüllük (Specificity)} = \frac{TN}{TN+FP} \quad (3.4)$$

$$\text{F1 Skoru} = \frac{2\ x\ Kesinlik\ x\ Duyarlılık}{Kesinlik+Duyarlılık} \quad (3.5)$$

$$\text{Kappa } (\kappa) = \frac{p_0-p_e}{1-p_e} \quad (3.6)$$

$$\text{AUC} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \quad (3.7)$$

$$\text{Final Skoru} = \frac{F1\ skoru+Kappa+AUC}{3} \quad (3.8)$$

## 3.6. Python Programlama Kodları

# pre_preprocess.py

```
import cv2
import numpy as np
import logging
import os
from odir_image_crop import ImageCrop


class ImageCrop:
    def __init__(self, source_folder, destination_folder, file_name):
        self.logger = logging.getLogger('odir')
        self.source_folder = source_folder
        self.destination_folder = destination_folder
        self.file_name = file_name

    def remove_black_pixels(self):
        file = os.path.join(self.source_folder, self.file_name)
        image = cv2.imread(file)

        # Mask of coloured pixels.
        mask = image > 0

        # Coordinates of coloured pixels.
        coordinates = np.argwhere(mask)

        # Binding box of non-black pixels.
        x0, y0, s0 = coordinates.min(axis=0)
        # slices are exclusive at the top
        x1, y1, s1 = coordinates.max(axis=0) + 1

        # Get the contents of the bounding box.
        cropped = image[x0:x1, y0:y1]
        # overwrite the same file
        file_cropped = os.path.join(self.destination_folder, self.file_name)
        cv2.imwrite(file_cropped, cropped)
```

# pre_preprocess_image_crop.py

```
    import logging
    import logging.config
    from os import listdir
    from os.path import isfile, join
    from odir_image_crop import ImageCrop
    def process_all_images():
        files = [f for f in listdir(source_folder)
                 if isfile(join(source_folder, f))]
        for file in files:
            logger.debug('Processing image: ' + file)
            ImageCrop(source_folder, destination_folder,
                      file).remove_black_pixels()
    if __name__ == '__main__':
        source_folder = r'/Users/bemir/ODIR-5K_Training_Dataset'
        destination_folder = r'/Users/bemir/ODIR-5K_Training_Dataset_cropped'
        # create logger
        logging.config.fileConfig('logging.conf')
        logger = logging.getLogger('odir')
        process_all_images()
```

# utils.py

```
import os
import sys
from tqdm import tqdm
import requests
from zipfile import ZipFile
import json
from PIL import Image, ImageOps
import matplotlib.pyplot as plt
import csv
import json

# returns csv data for the given path.
def read_csv_file(path):
    l = []
    with open(path, mode='r') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=';')
        for row in csv_reader:
            l.append(row)
    return l

# returns PIL image for the given path.
def load_image(path):
    return Image.open(path)

# shows the given input PIL image.
def show_image(img):
    img.show()

# returns rgb image of the given input image.
def gray_to_rgb(im):
    return im.convert('RGB')

# returns resized image for the given image.
def resize(im):
    return im.resize((299, 299))

# returns collected unique disease for the given annotations and save them into a json file.
def create_dict(annotations):
    d = {}
    index = 0
    for ann in annotations[1:]:
        left_eye = ann[5]
        right_eye = ann[6]
        splitted_left_eye = left_eye.split(',')
        splitted_right_eye = right_eye.split(',')
        for a in splitted_left_eye:
            if a not in d:
                d[a] = index
                index += 1
        for a in splitted_right_eye:
            if a not in d:
                d[a] = index
                index += 1
    with open('classes.json', 'w') as file:
        json.dump(d, file)
    return d

```

```
58 # returns saved disease classes from the json file.
59 def load_dict():
60     with open('classes.json', 'r') as file:
61         d = json.load(file)
62     return d
63 
64 # returns the mirrored image of the given image.
65 def mirror_image(im):
66     return ImageOps.mirror(im)
67 
68 # returns horizontal concatenation of the given two images.
69 def concat_horizontal(im1, im2):
70     dst = Image.new('RGB', (im1.width + im2.width, im1.height))
71     dst.paste(im1, (0, 0))
72     dst.paste(im2, (im1.width, 0))
73     return dst
74 
75 # returns vertical concatenation of the given two images.
76 def concat_vertical(im1, im2):
77     dst = Image.new('RGB', (im1.width, im1.height + im2.height))
78     dst.paste(im1, (0, 0))
79     dst.paste(im2, (0, im1.height))
80     return dst
81 
82 # returns the input image for the deep learning architectures.
83 def create_input_image(im1, im2):
84     concataneted = concat_horizontal(im1, im2)
85     mirrored_conc = mirror_image(concataneted)
86     output_image = concat_vertical(concataneted, mirrored_conc)
87     return output_image
```

# preprocess_annotations.py

```
from utils import *
train_annotations = read_csv_file('Annotations/train.csv')
val_annotations = read_csv_file('Annotations/val.csv')
test_annotations = read_csv_file('Annotations/test.csv')
# Extract invalid data from the dataset.
spoiled_keys = ["2174", "2175", "2176", "2177", "2178", "2179", "2180", "2181",
"2182", "2957"]
spoiled_classes = ["anterior segment image", "no fundus image"]
trivial_classes = ["lens dust", "optic disk photographically invisible", "low image
quality", "image offset"]
inappropriate_classes = spoiled_classes + trivial_classes
def check_data_appropriate(annotation):
    l = annotation.split(',')
    for c in l:
        if c in inappropriate_classes:
            return False
    return True


def check_person_valid(annotation):
    if annotation[0] in spoiled_keys:
        return False
    if not check_data_appropriate(annotation[5]):
        return False
    if not check_data_appropriate(annotation[6]):
        return False
    return True
# Extract invalid data and save the remaining valid data into a csv file.
def process_annotations(annotations, fn="processed_annotations"):

    processed_annotations = []
    processed_annotations.append(annotations[0])

    for ann in annotations[1:]:

        validity_of_person = check_person_valid(ann)
        if validity_of_person:
            processed_annotations.append(ann)

    with open(f"{fn}.csv", 'w') as f:
        write = csv.writer(f, delimiter=";")
        write.writerows(processed_annotations)

    return processed_annotations

process_annotations(annotations=train_annotations,
fn="Annotations/processed_train_annotations")
process_annotations(annotations=val_annotations,
fn="Annotations/processed_val_annotations")
process_annotations(annotations=test_annotations,
fn="Annotations/processed_test_annotations")
```

# classes.json


```
{
"cataract": 0,
"normal fundus": 1,
"laser spot": 2,
"moderate non proliferative retinopathy": 3,
"branch retinal artery occlusion": 4,
"macular epiretinal membrane": 5,
"mild nonproliferative retinopathy": 6,
"epiretinal membrane": 7,
"drusen": 8,
"vitreous degeneration": 9,
"hypertensive retinopathy": 10,
"retinal pigmentation": 11,
"pathological myopia": 12,
"myelinated nerve fibers": 13,
"depigmentation of the retinal pigment epithelium": 14,
"abnormal pigment ": 15,
"post laser photocoagulation": 16,
"glaucoma": 17,
"macular hole": 18,
"wet age-related macular degeneration": 19,
"dry age-related macular degeneration": 20,
"epiretinal membrane over the macula": 21,
"central retinal artery occlusion": 22,
"pigment epithelium proliferation": 23,
"diabetic retinopathy": 24,
"atrophy": 25,
"chorioretinal atrophy": 26,
"white vessel": 27,
"retinochoroidal coloboma": 28,
"atrophic change": 29,
"retinitis pigmentosa": 30,
"retina fold": 31,
"suspected glaucoma": 32,
"branch retinal vein occlusion": 33,
"optic disc edema": 34,
"retinal pigment epithelium atrophy": 35,
"severe nonproliferative retinopathy": 36,
"spotted membranous change": 37,
"proliferative diabetic retinopathy": 38,
"refractive media opacity": 39,
"suspected microvascular anomalies": 40,
"severe proliferative diabetic retinopathy": 41,
"central retinal vein occlusion": 42,
"tessellated fundus": 43,
"maculopathy": 44,
"oval yellow-white atrophy": 45,
"suspected retinal vascular sheathing": 46,
"macular coloboma": 47,
"vessel tortuosity": 48,
"idiopathic choroidal neovascularization": 49,
"optic nerve atrophy": 50,
"wedge white line change": 51,
"old chorioretinopathy": 52,
"punctate inner choroidopathy": 53,
"myopia retinopathy": 54,
"old choroiditis": 55,
"myopic maculopathy": 56,
```

```
"chorioretinal atrophy with pigmentation proliferation": 57,
"congenital choroidal coloboma": 58,
"optic disk epiretinal membrane": 59,
"morning glory syndrome": 60,
"retinal pigment epithelial hypertrophy": 61,
"old branch retinal vein occlusion": 62,
"asteroid hyalosis": 63,
"retinal artery macroaneurysm": 64,
"suspicious diabetic retinopathy": 65,
"suspected diabetic retinopathy": 66,
"glial remnants anterior to the optic disc": 67,
"diffuse chorioretinal atrophy": 68,
"optic discitis": 69,
"intraretinal hemorrhage": 70,
"arteriosclerosis": 71,
"silicone oil eye": 72,
"retinal vascular sheathing": 73,
"choroidal nevus": 74,
"suspected retinitis pigmentosa": 75,
"old central retinal vein occlusion": 76,
"diffuse retinal atrophy": 77,
"fundus laser photocoagulation spots": 78,
"suspected abnormal color of  optic disc": 79,
"myopic retinopathy": 80,
"vitreous opacity": 81,
"macular pigmentation disorder": 82,
"pigmentation disorder": 83,
"suspected moderate non proliferative retinopathy": 84,
"suspected macular epimacular membrane": 85,
"peripapillary atrophy": 86,
"retinal detachment": 87,
"central serous chorioretinopathy": 88,
"post retinal laser surgery": 89,
"age-related macular degeneration": 90,
"intraretinal microvascular abnormality": 91,
"moderate nonproliferative retinopathy": 92,
"suspected abnormal macular pigment ": 93,
"nerve fiber layer defect": 94,
"sever hypertension": 95,
"mild hypertension": 96,
"vascular loops": 97,
"suspected fibrous proliferation membrane ": 98,
"stripe-like change": 99,
"suspected moderate nonproliferative retinopathy": 100,
"suspected pathological myopia": 101,
"suspected choroidal neovascularization": 102,
"geographic atrophy": 103,
"Bietti crystalline corneoretinal dystrophy": 104,
"bull's eye maculopathy": 105,
"multifocal choroiditis": 106,
"optic disc anomaly": 107,
"abnormal camera exposure ": 108,
"ischemic optic neuropathy": 109,
"microaneurysm": 110,
"macualar vascular abnormality ": 111,
"Coats disease": 112,
"MAC": 113,
"branch retinal occlusion": 114,
"post photocoagulation of old branch retinal vein occlusion": 115,
"congenital optic disc anomaly": 116 }
```

# collect_datasets.py

```
from utils import *
from tqdm import tqdm

val_annotations = read_csv_file('Annotations/processed_val_annotations.csv')
val_annotations = val_annotations[1:]
eye_diseases = ["N", "D", "G", "C", "A", "H", "M", "O"]
eye_diseases_index = [-8, -7, -6, -5, -4, -3, -2, -1]

# Dataset -> {eye_diseases} -> 0 and 1 | Create folders of annotations for binary classification.
try:
    os.mkdir("Dataset_Val")
    print("Directory Dataset_Val Created ")
except FileExistsError:
    print("Directory Dataset_Val already exists")
for folder_name in eye_diseases:
    try:
        os.mkdir(f"Dataset_Val/{folder_name}")
        os.mkdir(f"Dataset_Val/{folder_name}/0")
        os.mkdir(f"Dataset_Val/{folder_name}/1")
        print(f"Directory {folder_name} Created ")
    except FileExistsError:
        print(f"Directory {folder_name} already exists")
# Move the images into binary folders.
for disease_class, disease_index in tqdm(zip(eye_diseases, eye_diseases_index)):
    print(disease_class)
    for ids, ann in tqdm(enumerate(val_annotations)):
        disease_class_of_person = int(ann[disease_index])
        person_id = int(ann[0])
        left_eye = gray_to_rgb(load_image(f'Val/{person_id}_left.jpg'))
        right_eye = gray_to_rgb(load_image(f'Val/{person_id}_right.jpg'))
        left_eye = resize(left_eye)
        right_eye = resize(right_eye)
        output_image = create_input_image(left_eye, right_eye)
        if disease_class_of_person:
            # save to 1 folder
            output_image.save(f'Dataset_Val/{disease_class}/1/{ids}.jpg')
        else:
            # save to 0 folder
            output_image.save(f'Dataset_Val/{disease_class}/0/{ids}.jpg')
```

# augment.py

```
import Augmentor

paths = ['N', 'D', 'G', 'C', 'A', 'H', 'M', 'O']
classifications = ['0', '1']

# Augment dataset for all given class images.
def augment(path, classification):
    pth = '/Users/bemir/Desktop/new/Dataset_Train/' + path + '/' + classification + '/'
    p = Augmentor.Pipeline(pth)
    p.random_distortion(probability=0.25, grid_width=3, grid_height=3,
magnitude=3)
    p.gaussian_distortion(probability=0.25, grid_width=3, grid_height=3,
magnitude=3, corner='bell', method='in')
    p.skew(probability=0.05)
    p.skew_tilt(probability=0.05)
    p.skew_left_right(probability=0.05)
    p.skew_top_bottom(probability=0.05)
    p.skew_corner(probability=0.05)
    p.sample(10000)


from tqdm import tqdm

for path in tqdm(paths):
    for classification in classifications:
        augment(path=path, classification=classification)
```

# train.py

```
import torch
import torchvision
import torchvision.transforms as transforms

import torch.optim as optim
from tqdm import tqdm
import torchvision.models as models
torch.set_default_tensor_type('torch.cuda.FloatTensor')
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

batch_size = 32

import matplotlib.pyplot as plt
import numpy as np
import csv
torch.manual_seed(7) # Allows deterministic behaviour and reproducibility

torch.cuda.manual_seed(7)
torch.cuda.manual_seed_all(7)
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False

def imshow(img):
    img = img / 2 + 0.5
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.show()


classes = ('0', '1')
# imshow(torchvision.utils.make_grid(images))
# print(' '.join('%5s' % classes[labels[j]] for j in range(batch_size)))

import torch.nn as nn
import torch.nn.functional as F

# returns inception_v3 for binary classification.
def get_inception():
    net = models.inception_v3(pretrained=True)
    net.AuxLogits.fc = nn.Linear(768, 2)
    net.fc = nn.Linear(2048, 2)
    counter = 0
    for child in net.children():
        if counter < 13:
            for param in child.parameters():
                param.requires_grad = False
        else:
            break
        counter += 1
    return net
# returns resnet50 for binary classification.
def get_resnet():
    net = models.resnet50(pretrained=True)
    net.fc = nn.Linear(2048, 2)
    counter = 0
    for child in net.children():
        if counter < 7:
            for param in child.parameters():
```

```
            param.requires_grad = False
        else:
            break
        counter += 1
    return net
# returns vgg16 for binary classification.
def get_vgg():
    net = models.vgg16(pretrained=True)
    net.classifier[6] = nn.Linear(4096, 2)
    for i, child in enumerate(net.children()):
        if i == 0:
            for c in child.parameters():
                c.requires_grad = False
    return net
# trains the model for the given model and dataset.
def train(model_name, class_name): # Given the model name get the model and
transform.
    if model_name == "inception":
        net = get_inception()
        transform = transforms.Compose([
            transforms.Resize((299, 299)),
            transforms.ToTensor(),
            transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])
    elif model_name == "resnet":
        net = get_resnet()
        transform = transforms.Compose([
            transforms.Resize((224, 224)),
            transforms.ToTensor(),
            transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])
    else:
        net = get_vgg()
        transform = transforms.Compose([
            transforms.Resize((224, 224)),
            transforms.ToTensor(),
            transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

    net = net.to(device) # move the model to gpu.
    result_list = [] # initialize result file.
    labels = ["Epoch", "Train Loss","Train Accuracy", "Val Loss", "Val Accuracy",
"Test Loss", "Test Accuracy"]
    result_list.append(labels) # Initialize Train, Val, and Test dataset.
    train_set = torchvision.datasets.ImageFolder(root=f'/content/oia-odir-augmented-
dataset/{class_name}',
                                                 transform=transform)
    train_loader = torch.utils.data.DataLoader(train_set, batch_size=batch_size,
                                               shuffle=True)

    val_set = torchvision.datasets.ImageFolder(root=f'/content/oia-odir-val-
dataset/{class_name}',
                                               transform=transform)
    val_loader = torch.utils.data.DataLoader(val_set, batch_size=batch_size,
                                             shuffle=True)

    test_set = torchvision.datasets.ImageFolder(root=f'/content/oia-odir-test-
dataset/{class_name}',
                                                transform=transform)
    test_loader = torch.utils.data.DataLoader(test_set, batch_size=batch_size,
                                              shuffle=True)

    criterion = nn.CrossEntropyLoss() # Initialize loss function.
```

```
114     optimizer = optim.Adam(net.parameters()) # Initialize optimizer.
115     best_val_loss = None
116     best_val_accuracy = None
117     counter = 0
118     for epoch in range(200):  # loop over the dataset multiple times
119 
120         net.train()
121         running_loss = 0.0
122         total = 0
123         correct = 0
124         for i, data in enumerate(train_loader, 0):
125             # get the inputs; data is a list of [inputs, labels]
126             inputs, labels = data[0].to(device), data[1].to(device)
127             # zero the parameter gradients
128             optimizer.zero_grad()
129 
130             # forward + backward + optimize
131             if model_name == "inception": # if model is optimization consider aux logits
for training.
132                 outputs, aux = net(inputs)
133                 loss = 0.0
134                 loss += criterion(outputs, labels)
135                 loss += 0.4 * criterion(aux, labels)
136 
137             else:
138                 outputs = net(inputs)
139                 loss = criterion(outputs, labels)
140             loss.backward()
141             optimizer.step()
142             _, predicted = torch.max(outputs.data, 1)
143             correct += (predicted == labels).sum().item()
144             total += labels.size(0)
145 
146             # print statistics
147             running_loss += loss.item() * labels.size(0)
148         print(f'Train Accuracy : {correct / total}')
149         train_loss = running_loss / total
150         train_accuracy = correct / total
151         net.eval()
152         running_loss = 0.0
153         total = 0
154         correct = 0
155         for i, data in enumerate(val_loader, 0):
156             # get the inputs; data is a list of [inputs, labels]
157             inputs, labels = data[0].to(device), data[1].to(device)
158             # zero the parameter gradients
159 
160             # forward + backward + optimize
161             with torch.no_grad():
162                 outputs = net(inputs)
163             _, predicted = torch.max(outputs.data, 1)
164             total += labels.size(0)
165             correct += (predicted == labels).sum().item()
166             # loss = 0.0
167             loss = criterion(outputs, labels)
168             # loss += criterion(outputs, labels)
169             # loss += 0.4*criterion(aux, labels)
170 
171             # print statistics
172             running_loss += loss.item() * labels.size(0)
```

```
            batch_loss = loss.item()
        print(f'Val Accuracy : {correct / total}')
        val_accuracy = correct / total
        val_loss = running_loss / total

        running_loss = 0.0
        total = 0
        correct = 0
        for i, data in enumerate(test_loader, 0):
            # get the inputs; data is a list of [inputs, labels]
            inputs, labels = data[0].to(device), data[1].to(device)
            # zero the parameter gradients

            # forward + backward + optimize
            with torch.no_grad():
                # outputs, aux = net(inputs)
                outputs = net(inputs)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
            # loss = 0.0
            loss = criterion(outputs, labels)
            # loss += criterion(outputs, labels)
            # loss += 0.4*criterion(aux, labels)
            # print statistics
            running_loss += loss.item() * labels.size(0)
            batch_loss = loss.item()
        print(f'Test Accuracy : {correct / total}')
        test_accuracy = correct / total
        test_loss = running_loss / total
        result = [epoch + 1, train_loss, train_accuracy, val_loss, val_accuracy, test_loss,
    test_accuracy]
        result_list.append(result)
        if best_val_accuracy is None:
            best_val_accuracy = val_accuracy
            counter = 0

        elif val_accuracy <= best_val_accuracy:
            counter += 1
            if counter >= 7:
                break
        else:
            best_val_accuracy = val_accuracy
            counter = 0
        torch.save(net, f"/content/drive/MyDrive/oia-odir/oia-odir-best-
models/{model_name}_{class_name}_{epoch+1}.pth") # save models

    with open(f'/content/drive/MyDrive/oia-odir/oia-odir-csv-
results/{model_name}_{class_name}.csv', 'w') as f:
        writer = csv.writer(f)
        writer.writerows(result_list) # save results into a csv file
class_names = ["N", "D", "G", "C", "A", "H", "M", "O"]
# model_names = ["inception", "resnet", "vgg"]
model_names = ["resnet"]
for model_name in tqdm(model_names): # Train each models for each classes.
  print(model_name)
  for class_name in class_names:
    print(class_name)
    train(model_name=model_name, class_name=class_name)
```

# test.py

```
from utils import *
import torch
import torch.nn as nn
import csv
import torchvision
import torchvision.transforms as transforms
from tqdm import tqdm
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
# device = torch.device("cpu")
test_annotations = read_csv_file('/content/drive/MyDrive/oia-odir/Annotations/processed_test_annotations.csv')
test_annotations = test_annotations[1:]

models = ["vgg", "inception", "resnet"]
# models = ["vgg"]

softmax = nn.Softmax(dim=1)
for model in models: # loop over models and calculate the probability of having the disease of each classes and save results into a csv file.
    if model == "inception":
        transform = transforms.Compose([
            transforms.Resize((299, 299)),
            transforms.ToTensor(),
            transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])
    else:
        transform = transforms.Compose([
            transforms.Resize((224, 224)),
            transforms.ToTensor(),
            transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

    N = torch.load(f"/content/drive/MyDrive/oia-odir/oia-odir-best-models/{model}_N_1.pth", map_location=device)
    D = torch.load(f"/content/drive/MyDrive/oia-odir/oia-odir-best-models/{model}_D_1.pth", map_location=device)
    G = torch.load(f"/content/drive/MyDrive/oia-odir/oia-odir-best-models/{model}_G_1.pth", map_location=device)
    C = torch.load(f"/content/drive/MyDrive/oia-odir/oia-odir-best-models/{model}_C_1.pth", map_location=device)
    A = torch.load(f"/content/drive/MyDrive/oia-odir/oia-odir-best-models/{model}_A_1.pth", map_location=device)
    H = torch.load(f"/content/drive/MyDrive/oia-odir/oia-odir-best-models/{model}_H_1.pth", map_location=device)
    M = torch.load(f"/content/drive/MyDrive/oia-odir/oia-odir-best-models/{model}_M_1.pth", map_location=device)
    O = torch.load(f"/content/drive/MyDrive/oia-odir/oia-odir-best-models/{model}_O_1.pth", map_location=device)
    N.eval()
    D.eval()
    G.eval()
    C.eval()
    A.eval()
    H.eval()
    M.eval()
    O.eval()
    result_list = []
    labels = ['ID', 'N', 'D', 'G', 'C', 'A', 'H', 'M', 'O']
    result_list.append(labels)
```

```
    for ann in tqdm(val_annotations):
        ann_id = ann[0]
        ann_left_path = ann[3]
        ann_right_path = ann[4]
        left_im = resize(gray_to_rgb(load_image(f"Test/{ann_id}_left.jpg")))
        right_im = resize(gray_to_rgb(load_image(f"Test/{ann_id}_right.jpg")))
        input_tensor = create_input_image(left_im, right_im)
        input_tensor = transform(input_tensor)
        input_tensor = input_tensor.unsqueeze(0).to(device)
        with torch.no_grad():
            N_result = softmax(N(input_tensor)).data[0, 1] # calculate the probability of
        being normal.
            D_result = softmax(D(input_tensor)).data[0, 1]
            G_result = softmax(G(input_tensor)).data[0, 1]
            C_result = softmax(C(input_tensor)).data[0, 1]
            A_result = softmax(A(input_tensor)).data[0, 1]
            H_result = softmax(H(input_tensor)).data[0, 1]
            M_result = softmax(M(input_tensor)).data[0, 1]
            O_result = softmax(O(input_tensor)).data[0, 1]
        result = [ann_id, N_result.item(), D_result.item(), G_result.item(),
C_result.item(), A_result.item(), H_result.item(), M_result.item(), O_result.item()]
        result_list.append(result)
# repeated val ve train files
    with open(f'/content/drive/MyDrive/oia-
odir/test/{model}_processed_test_result_1.csv', 'w') as f: # save results into csv file.
        writer = csv.writer(f)
        writer.writerows(result_list)
```

# evaluation.py

```
import matplotlib.pyplot as plt
from sklearn import metrics
import numpy as np
import sys
import xlrd
import csv
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from utils import read_csv_file
import os


# read the ground truth from xlsx file and output case id and eight labels
def importGT(filepath):
    data = xlrd.open_workbook(filepath)
    table = data.sheets()[0]
    data = [[int(table.row_values(i, 0, 1)[0])] + table.row_values(i, -8) for i in range(1,
table.nrows)]
    return np.array(data)


# read the submitted predictions in csv format and output case id and eight labels
def importPR(gt_data, filepath):
    with open(filepath, 'r') as f:
        reader = csv.reader(f)
        header = next(reader)
        pr_data = [[int(row[0])] + list(map(float, row[1:])) for row in reader]
    pr_data = np.array(pr_data)

    # Sort columns if they are not in predefined order
    order = ['ID', 'N', 'D', 'G', 'C', 'A', 'H', 'M', 'O']
```

```
    order_index = [0, 1, 2, 3, 4, 5, 6, 7, 8]
    order_dict = {item: ind for ind, item in enumerate(order)}
    sort_index = [order_dict[item] for ind, item in enumerate(header) if item in
order_dict]
    wrong_col_order = 0
    if (sort_index != order_index):
        wrong_col_order = 1
        pr_data[:, order_index] = pr_data[:, sort_index]

        # Sort rows if they are not in predefined order
    wrong_row_order = 0
    order_dict = {item: ind for ind, item in enumerate(gt_data[:, 0])}
    order_index = [v for v in order_dict.values()]
    sort_index = [order_dict[item] for ind, item in enumerate(pr_data[:, 0]) if item in
order_dict]
    if (sort_index != order_index):
        wrong_row_order = 1
        pr_data[order_index, :] = pr_data[sort_index, :]

    # If have missing results
    missing_results = 0
    if (gt_data.shape != pr_data.shape):
        missing_results = 1
    return pr_data, wrong_col_order, wrong_row_order, missing_results


# for classes calculate confusion matrix accuracy, precision, sensitivity, specificity,
          kappa, F-1 score and AUC value. Save confusion matrix into a plot and results into a
          csv file.
def ODIR_Metrics_for_classes(gt_data, pr_data, name):
    classes = ['N', 'D', 'G', 'C', 'A', 'H', 'M', 'O']
    th = 0.5
    # gt = gt_data.flatten()
    # pr = pr_data.flatten()
    result_list = []
    labels = ["accuracy", "precision", "sensitivity", "specificity", "kappa", "f1",
                        "auc", "final score"]
    result_list.append(labels)
    for i, classification in enumerate(classes):
        gt = gt_data[:, i]
        pr = pr_data[:, i]
        classified_pr = [1 if i >= th else 0 for i in pr]
        cm = confusion_matrix(gt, classified_pr)
        disp = ConfusionMatrixDisplay(confusion_matrix=cm)
        plt.figure()
        disp.plot()
        title_name = name.replace('_', ' ')
        plt.title(f'{title_name} {classes[i]}')
        plt.savefig(f'{name}/{name}_{classes[i]}.png')
        tn, fp, fn, tp = confusion_matrix(gt, classified_pr).ravel()
        if tn + fp + fn + tp == 0:
            accuracy = 0
        else:
            accuracy = (tn + tp) / (tn + fp + fn + tp) # calculate accuracy
        if fp + tp == 0:
            precision = 0
        else:
            precision = (tp) / (fp + tp) # calculate precision
        if tp + fn == 0:
            sensitivity = 0
```

```
85        else:
86            sensitivity = (tp) / (tp + fn) # calculate sensitivity
87        if tn + fp == 0:
88            specificity = 0
89        else:
90            specificity = (tn) / (tn + fp) # calculate specificity
91        kappa = metrics.cohen_kappa_score(gt, pr > th) # calculate kappa
92        f1 = metrics.f1_score(gt, pr > th, average='micro') # calculate f1 score
93        auc = metrics.roc_auc_score(gt, pr) # calculate auc score
94        final_score = (kappa + f1 + auc) / 3.0 # calculate final score
95
96        result = [accuracy, precision, sensitivity, specificity, kappa, f1, auc, final_score]
97        result_list.append(result)
98
99    with open(f'{name}/{name}_class_results.csv', 'w') as f:
100       writer = csv.writer(f)
101       writer.writerows(result_list) # write results into csv file.
102   # return kappa, f1, auc, final_score
103
104
105 # calculate metrics for the overall model.
106 def ODIR_Metrics(gt_data, pr_data, name):
107     th = 0.5
108     gt = gt_data.flatten()
109     pr = pr_data.flatten()
110     classified_pr = [1 if i >= th else 0 for i in pr]
111     tn, fp, fn, tp = confusion_matrix(gt, classified_pr).ravel()
112     if tn + fp + fn + tp == 0:
113         accuracy = 0
114     else:
115         accuracy = (tn + tp) / (tn + fp + fn + tp)
116     if fp + tp == 0:
117         precision = 0
118     else:
119         precision = (tp) / (fp + tp)
120     if tp + fn == 0:
121         sensitivity = 0
122     else:
123         sensitivity = (tp) / (tp + fn)
124     if tn + fp == 0:
125         specificity = 0
126     else:
127         specificity = (tn) / (tn + fp)
128     kappa = metrics.cohen_kappa_score(gt, pr > th)
129     f1 = metrics.f1_score(gt, pr > th, average='micro')
130     auc = metrics.roc_auc_score(gt, pr)
131     final_score = (kappa + f1 + auc) / 3.0
132
133     result_list = []
134     labels = ["accuracy", "precision", "sensitivity", "specificity", "kappa", "f1",
                "auc", "final score"]
135     result_list.append(labels)
136     result = [accuracy, precision, sensitivity, specificity, kappa, f1, auc, final_score]
137     result_list.append(result)
138     with open(f'{name}/{name}_results.csv', 'w') as f:
139         writer = csv.writer(f)
140         writer.writerows(result_list) # save results into a csv file.
141
142
143 # calculate confusion matrix for the model performance
```

```
144 def confusion_matrix_for_all(gt_data, pr_data, name, matrix_name, title_name):
145     gt_data = np.argmax(gt_data, axis=-1)
146     pr_data = np.argmax(pr_data, axis=-1)
147     cm = confusion_matrix(gt_data, pr_data)
148     disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['N', 'D',
'G', 'C', 'A', 'H', 'M', 'O'])
149     plt.figure()
150     disp.plot()
151     title_name = title_name.replace('_', ' ')
152     plt.title(f'{title_name}')
153     plt.savefig(f'{name}/{matrix_name}.png')
154     # np.savetxt('inception_overall_processed_test_confusion_matrix.csv', cm,
delimiter=',', fmt='%i')
155 
156     # return kappa, f1, auc, final_score
157 
158 # calculate scores for the model and classes and save the results into csv file and
confusion matrix into png file.
159 def calculate_score(GT_filepath, PR_filepath, name):
160     try:
161         os.mkdir(f"{name}")
162         print(f"Directory {name} Created ")
163     except FileExistsError:
164         print(f"/Directory {name} already exists")
165     val_annotations = read_csv_file(GT_filepath)
166     val_annotations = val_annotations[1:]
167     gt_data = []
168     for ann in val_annotations:
169         gt_data.append([int(ann[0]), float(ann[-8]), float(ann[-7]), float(ann[-6]),
          float(ann[-5]), float(ann[-4]),
170                         float(ann[-3]), float(ann[-2]), float(ann[-1])])
171     gt_data = np.array(gt_data)
172     pr_data, wrong_col_order, wrong_row_order, missing_results =
importPR(gt_data, PR_filepath)
173 
174     ODIR_Metrics(gt_data[:, 1:], pr_data[:, 1:], name=name)
175     ODIR_Metrics_for_classes(gt_data[:, 1:], pr_data[:, 1:], name=name)
176     confusion_matrix_for_all(gt_data=gt_data[:, 1:], pr_data=pr_data[:, 1:],
name=name, matrix_name=name, title_name=name)
177 
178 
179 calculate_score('Annotations/processed_test_annotations.csv',
          'vgg_processed_overall_test_result.csv', name='vgg_test')
```

# 4. BULGULAR

Bu tez çalışmasında, sol ve sağ fundus görüntülerinden sekiz farklı hastalığın sınıflandırılması amaçlandı. Kullanılan veri setinde görüntüler sağlıklı, diyabetik retinopati, glokom, katarakt, miyop, yaşa bağlı makula dejenerasyonu, hipertansiyon ve diğer hastalık sınıfları olmak üzere sekiz farklı kategoriye ayrılmış olup her bir fundus görüntüsü farklı teşhis anahtarına sahiptir. ODIR veri seti eğitim, doğrulama ve test olmak üzere %70-%20-%10 oranında ayrıldı ve modellerin sınıflandırma performansları incelendi.

Derin özellik çıkarımı ImageNet veri tabanı ile ön eğitimli VGG16, Inceptionv3, ResNet50 model mimarileri ile gerçekleştirildi. CNN modelleri kullanılarak çıkarılan derin özellikler ince ayar (fine tuning) yöntemi ile tam bağlantı katmanına aktarıldı. Model eğitimi gerçekleştirildi. Doğrulama ve test setinde her bir hastalık sınıfı için CNN modellerinin sınıflandırma performansları değerlendirildi. Daha sonra sekiz farklı hastalık sınıf kategorisi için modellerin sınıflandırma performansları değerlendirildi. Her bir hastalık sınıfı için karmaşıklık matrisi oluşturuldu. Doğruluk (Accuracy), Kesinlik (Precision), Duyarlılık (Recall, Sensitivity), Özgüllük (Specificity) değerleri elde edildi. Performans değerlendirmesi için Kappa, F1 skoru, ROC (Receiver Operating Characteristic Curves) eğrisi altında kalan alan (AUC) ve bu üç ölçütün ortalaması ile Final skoru dahil olmak üzere dört değerlendirme ölçütü kullanıldı.

VGG16, Inceptionv3 ve ResNet50 model mimarileri ile her bir hastalık sınıfına ait model sınıflandırma performans bulguları ve sekiz hastalık sınıf kategorisine ait derin konvolüsyonel sinir ağı mimarisi modelleri sınıflandırma performans bulguları bu bölümde yer almaktadır.

## 4.1. Normal Sınıf Kategorisine ait Derin Konvolüsyonel Sinir Ağı Mimari Modelleri Sınıflandırma Performans Bulguları

Eğitim, doğrulama ve test seti için VGG16, Inceptionv3, ResNet50 modellerine ait her bir epokta (epoch) hata (loss) ve doğruluk (accuracy) değerleri Tablo 4.1'deki gibi elde edilmiştir.

Eğitim, doğrulama ve test seti doğruluk değerleri sırası ile VGG16 modeli, 14 epok sonunda 0.719, 0.597 ve 0.631; Inceptionv3 modeli, 20 epok sonunda 0.997, 0.660 ve 0.649; ResNet50 modeli, 12 epok sonunda 0.991, 0.648 ve 0.666 olarak bulunmuştur. ResNet50 modelinin test seti için en yüksek doğruluğa sahip olduğu görülmektedir.

Eğitim, doğrulama ve test seti hata değerleri sırası ile VGG16 modeli, 14 epok sonunda 0.476, 1.193 ve 0.978; Inceptionv3 modeli, 20 epok sonunda 0.016, 2.263 ve 2.536; ResNet50 modeli, 0.028, 0.991 ve 1.812 olarak bulunmuştur. VGG16 modelinin test seti için en düşük hata değerine sahip olduğu görülmektedir.

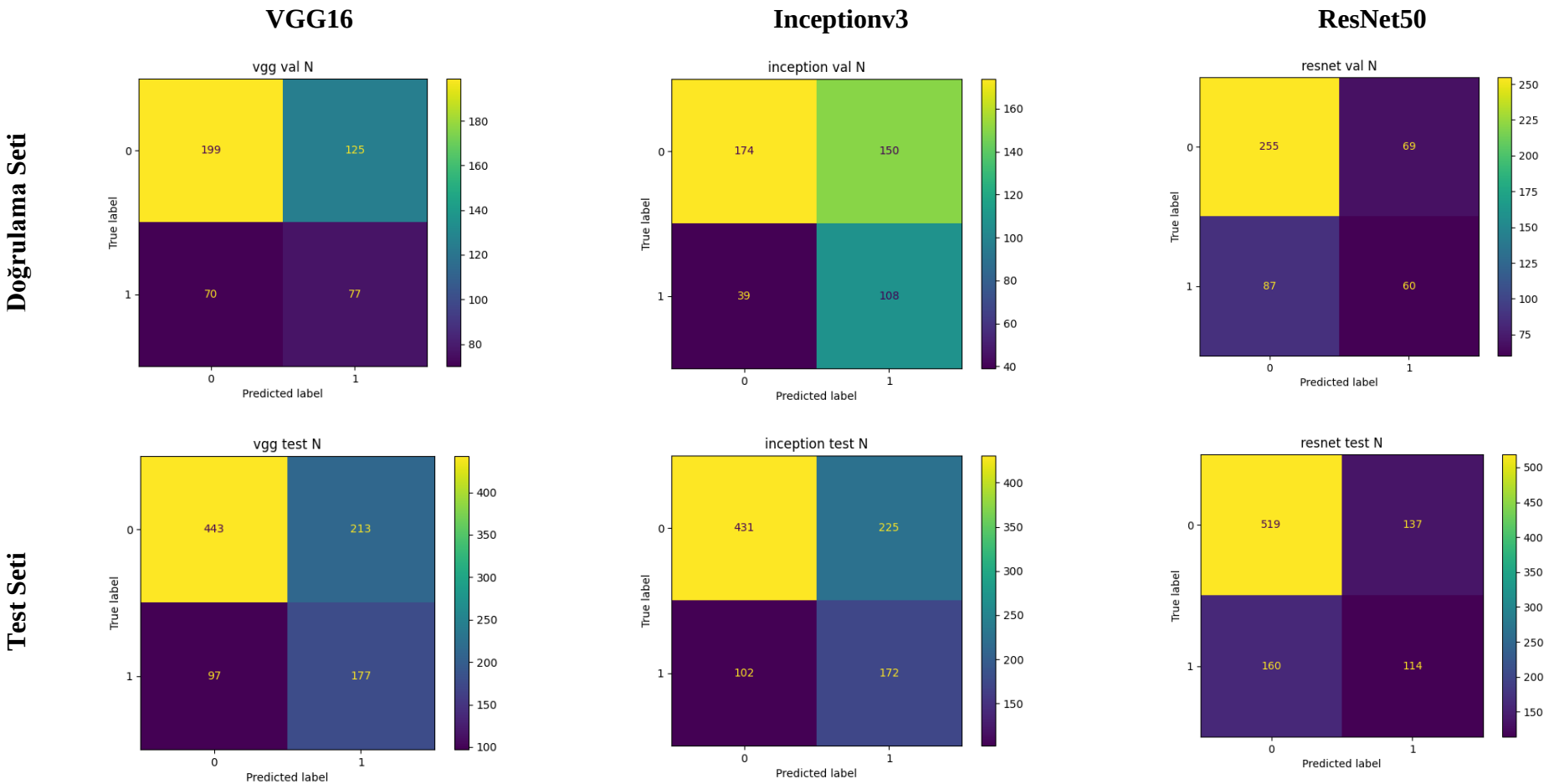
Doğrulama ve test seti için normal sınıf kategorisine ait karmaşıklık matrisi Şekil 4.1'de yer almaktadır. Doğrulama setinde toplam 471 fundus görüntüsü bulunmaktadır. Bu fundus görüntülerinin 147'sinin normal sınıfına ait olduğu bilinmektedir. VGG16 modeli, 77; Inceptionv3 modeli, 108; ResNet50 modeli, 60'ını doğru olarak sınıflandırmıştır. Normal olmayan sınıf kategorisine sahip 324 fundus görüntüsü bulunmaktadır. VGG16 modeli, 199; Inceptionv3 modeli, 174; ResNet50 modeli, 255'ini doğru olarak sınıflandırmıştır. Test setinde toplam 930 fundus görüntüsü bulunmaktadır. Bu fundus görüntülerinin 274'ünün normal sınıfına ait olduğu bilinmektedir. VGG16 modeli, 177; Inceptionv3 modeli, 172; ResNet50 modeli, 114'ünü doğru olarak sınıflandırmıştır. Normal olmayan sınıf kategorisine sahip 656 fundus görüntüsü bulunmaktadır. VGG16 modeli, 434; Inceptionv3 modeli, 431; ResNet50 modeli, 519'unu doğru olarak sınıflandırmıştır.

Karmaşıklık matrisi elde edildikten sonra her bir modelin sınıflandırma başarı ölçütleri hesaplanarak Tablo 4.2'de sunulmuştur. VGG16, Inceptionv3, ResNet50 modelleri için her bir epokta doğruluk, hata, kesinlik, duyarlılık, Eğri altında kalan alan (AUC), F1 skoru, Kappa ve Final değeri sonuçları grafiksel olarak Şekil 4.2, Şekil 4.3, Şekil 4.4'de sunulmuştur.

Normal sınıf kategorisi için test setinde VGG16, Inceptionv3, ResNet50 modelleri için Final skor değerleri sırası ile 0.557, 0.537 ve 0.532 olarak hesaplanmıştır. En yüksek Final skoru VGG16 modeli ile elde edilmiştir.

**Tablo 4.1.** Eğitim, doğrulama ve test seti için Normal Sınıf Kategorisine ait VGG16, Inceptionv3 ve ResNet50 Modeli Hata ve Doğruluk değerleri

| | | Eğitim seti | | Doğrulama seti | | Test seti | |
|---|---|---|---|---|---|---|---|
| Model | Epok | Hata (Loss) | Doğruluk (Accuracy) | Hata (Loss) | Doğruluk (Accuracy) | Hata (Loss) | Doğruluk (Accuracy) |
| VGG16 | 1 | 0,817 | 0,547 | 0,634 | 0,635 | 0,628 | 0,653 |
| | 2 | 0,702 | 0,576 | 0,618 | 0,605 | 0,584 | 0,656 |
| | 3 | 0,672 | 0,595 | 0,609 | 0,603 | 0,579 | 0,658 |
| | 4 | 0,631 | 0,638 | 0,720 | 0,633 | 0,622 | 0,669 |
| | 5 | 0,633 | 0,631 | 0,673 | 0,514 | 0,654 | 0,538 |
| | 6 | 0,577 | 0,675 | 0,745 | 0,637 | 0,597 | 0,665 |
| | 7 | 0,562 | 0,682 | 0,662 | 0,66 | 0,590 | 0,673 |
| | 8 | 0,517 | 0,690 | 0,653 | 0,607 | 0,588 | 0,638 |
| | 9 | 0,513 | 0,696 | 0,966 | 0,645 | 0,707 | 0,709 |
| | 10 | 0,518 | 0,705 | 0,867 | 0,597 | 0,707 | 0,638 |
| | 11 | 0,513 | 0,701 | 0,862 | 0,626 | 0,670 | 0,667 |
| | 12 | 0,501 | 0,712 | 1,026 | 0,639 | 0,770 | 0,695 |
| | 13 | 0,465 | 0,719 | 1,036 | 0,616 | 0,758 | 0,647 |
| | 14 | 0,476 | 0,719 | 1,193 | 0,597 | 0,978 | 0,631 |
| Inceptionv3 | 1 | 0,723 | 0,741 | 0,777 | 0,626 | 0,732 | 0,639 |
| | 2 | 0,343 | 0,896 | 1,345 | 0,614 | 1,393 | 0,633 |
| | 3 | 0,174 | 0,952 | 1,396 | 0,652 | 1,389 | 0,652 |
| | 4 | 0,120 | 0,968 | 1,722 | 0,658 | 1,593 | 0,666 |
| | 5 | 0,087 | 0,978 | 1,863 | 0,645 | 1,724 | 0,644 |
| | 6 | 0,071 | 0,982 | 2,102 | 0,643 | 1,960 | 0,632 |
| | 7 | 0,057 | 0,986 | 2,066 | 0,639 | 1,865 | 0,659 |
| | 8 | 0,048 | 0,989 | 2,201 | 0,643 | 1,753 | 0,672 |
| | 9 | 0,049 | 0,988 | 2,205 | 0,677 | 2,058 | 0,683 |
| | 10 | 0,043 | 0,991 | 2,240 | 0,652 | 2,013 | 0,661 |
| | 11 | 0,035 | 0,992 | 2,481 | 0,675 | 1,970 | 0,685 |
| | 12 | 0,04 | 0,992 | 2,197 | 0,662 | 1,914 | 0,691 |
| | 13 | 0,032 | 0,993 | 2,301 | 0,684 | 1,901 | 0,680 |
| | 14 | 0,026 | 0,994 | 2,750 | 0,669 | 2,274 | 0,681 |
| | 15 | 0,030 | 0,993 | 2,517 | 0,669 | 2,417 | 0,662 |
| | 16 | 0,021 | 0,996 | 2,544 | 0,633 | 2,262 | 0,671 |
| | 17 | 0,019 | 0,996 | 2,399 | 0,660 | 2,212 | 0,661 |
| | 18 | 0,028 | 0,993 | 2,384 | 0,652 | 2,204 | 0,686 |
| | 19 | 0,025 | 0,994 | 2,289 | 0,667 | 2,571 | 0,646 |
| | 20 | 0,016 | 0,997 | 2,263 | 0,660 | 2,536 | 0,649 |
| ResNet50 | 1 | 0,507 | 0,734 | 1,638 | 0,554 | 1,805 | 0,539 |
| | 2 | 0,221 | 0,904 | 1,367 | 0,584 | 1,468 | 0,575 |
| | 3 | 0,114 | 0,953 | 1,455 | 0,650 | 1,302 | 0,672 |
| | 4 | 0,080 | 0,970 | 1,720 | 0,650 | 1,592 | 0,641 |
| | 5 | 0,059 | 0,977 | 1,612 | 0,673 | 1,435 | 0,690 |
| | 6 | 0,048 | 0,982 | 1,941 | 0,616 | 1,941 | 0,643 |
| | 7 | 0,034 | 0,987 | 2,445 | 0,611 | 2,460 | 0,623 |
| | 8 | 0,043 | 0,984 | 1,828 | 0,622 | 1,674 | 0,633 |
| | 9 | 0,030 | 0,989 | 2,228 | 0,624 | 1,764 | 0,686 |
| | 10 | 0,030 | 0,990 | 2,226 | 0,639 | 2,046 | 0,676 |
| | 11 | 0,028 | 0,991 | 2,054 | 0,626 | 2,015 | 0,630 |
| | 12 | 0,028 | 0,991 | 1,890 | 0,648 | 1,812 | 0,666 |

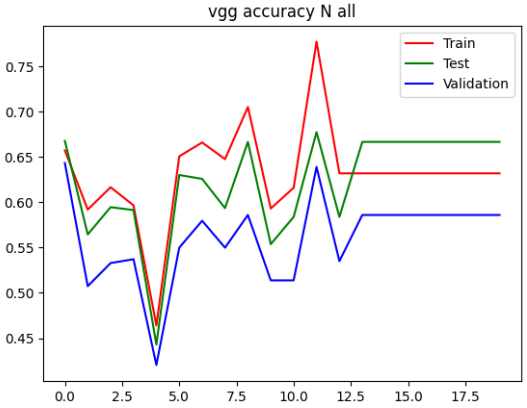



**Şekil 4.1.** Doğrulama ve test seti için VGG16, Inceptionv3 ve ResNet50 Modeli Normal Sınıf Kategorisine ait Karmaşıklık Matrisi

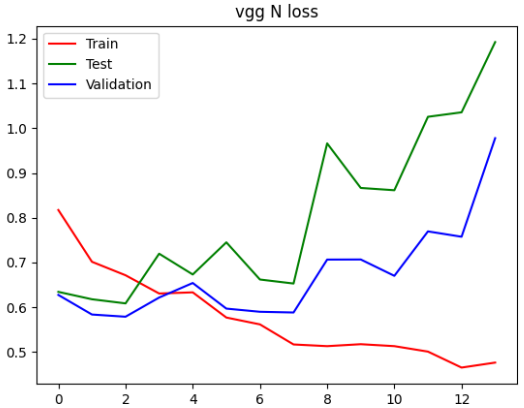
**Tablo 4.2.** Doğrulama ve test seti için Normal Sınıf Kategorisine ait VGG16, Inceptionv3 ve ResNet50 Modeli sınıflandırma başarı ölçütleri

| Model | Performans ölçütleri | Doğrulama seti | Test seti |
|---|---|---|---|
| VGG16 | Accuracy | 0,586 | 0,667 |
| | Precision | 0,381 | 0,454 |
| | Sens | 0,524 | 0,646 |
| | Spec | 0,614 | 0,675 |
| | $\kappa$ | 0,125 | 0,286 |
| | F1 | 0,586 | 0,667 |
| | AUC | 0,617 | 0,717 |
| | Final | 0,443 | 0,557 |
| Inceptionv3 | Accuracy | 0,599 | 0,648 |
| | Precision | 0,419 | 0,433 |
| | Sens | 0,735 | 0,628 |
| | Spec | 0,537 | 0,657 |
| | $\kappa$ | 0,225 | 0,252 |
| | F1 | 0,599 | 0,648 |
| | AUC | 0,691 | 0,711 |
| | Final | 0,505 | 0,537 |
| ResNet50 | Accuracy | 0,669 | 0,681 |
| | Precision | 0,465 | 0,454 |
| | Sens | 0,408 | 0,416 |
| | Spec | 0,787 | 0,791 |
| | $\kappa$ | 0,202 | 0,212 |
| | F1 | 0,669 | 0,681 |
| | AUC | 0,682 | 0,703 |
| | Final | 0,518 | 0,532 |

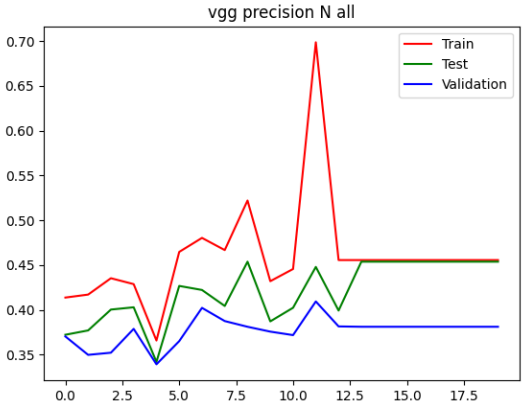
Accuracy: Doğruluk; Precision: Kesinlik; Sens: Duyarlılık; Spec: Özgüllük; $\kappa$: Kappa; AUC: Eğri altında kalan alan; Final: F1, AUC ve $\kappa$ değeri ortalaması



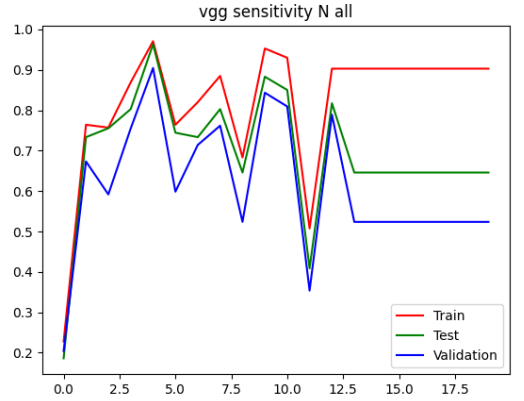

a) Doğruluk (Accuracy)




b) Hata (Loss)

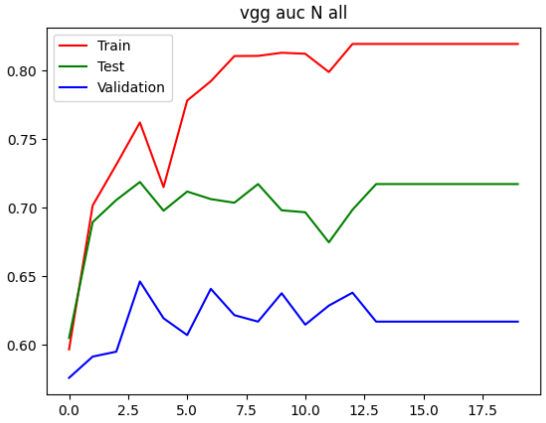



c) Kesinlik (Precision)

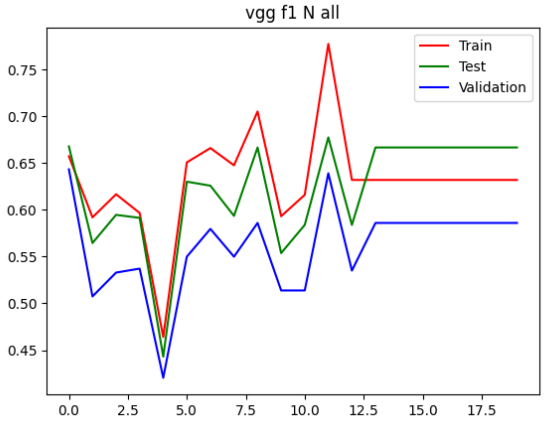



d) Duyarlılık (Recall)

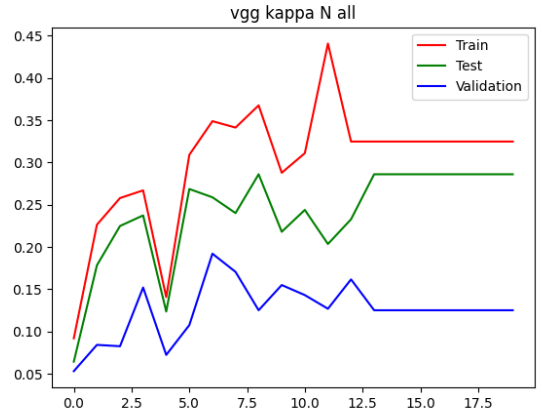
**Şekil 4.2.** Normal sınıf kategorisine ait VGG16 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi



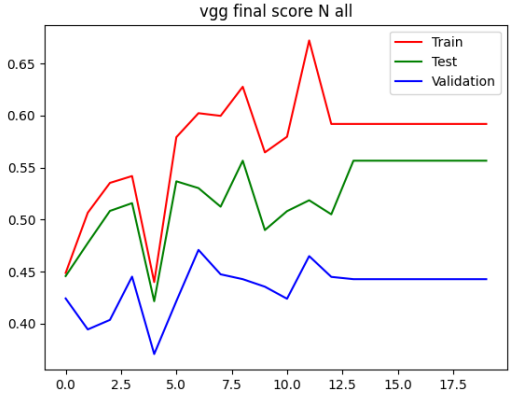

e) Eğri altında kalan alan (AUC)




f) F1 skoru

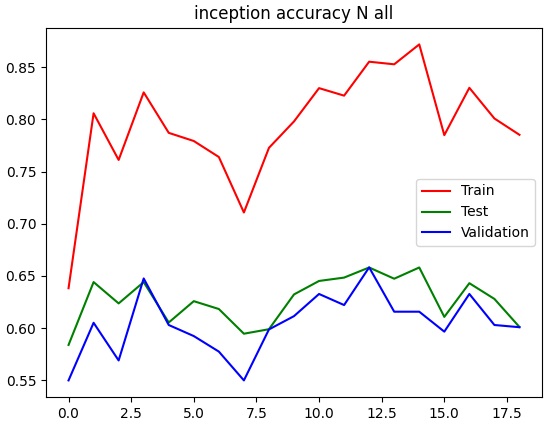



g) Kappa

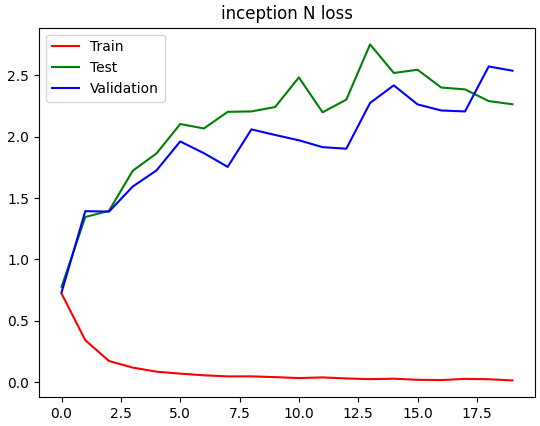



h) Final Skoru

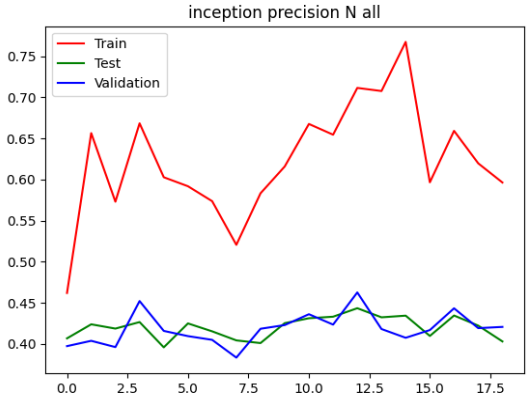
**Şekil 4.2.** Normal sınıf kategorisine ait VGG16 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)



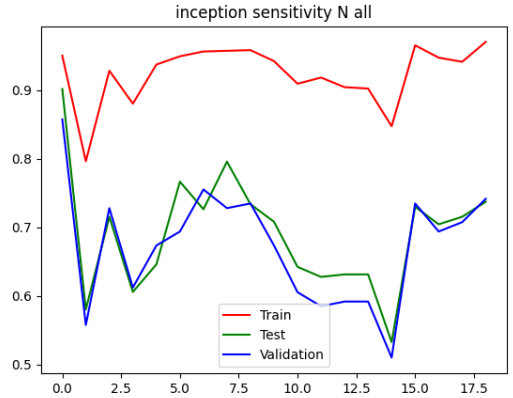

a) Doğruluk (Accuracy)




b) Hata (Loss)

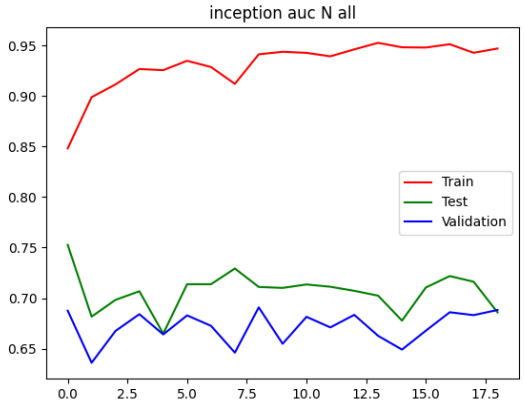



c) Kesinlik (Precision)

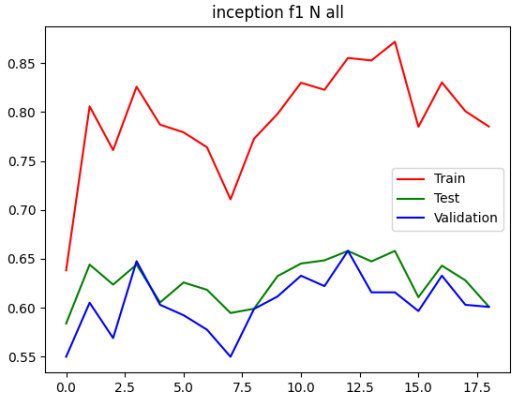



d) Duyarlılık (Recall)

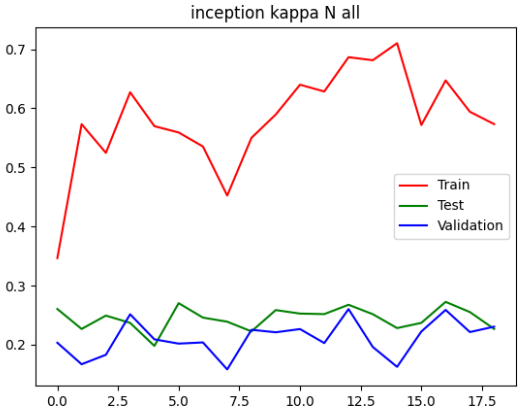
**Şekil 4.3.** Normal sınıf kategorisine ait Inceptionv3 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi



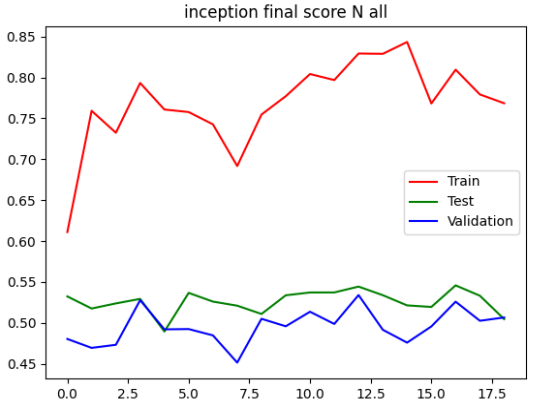

e) Eğri altında kalan alan (AUC)




f) F1 skoru

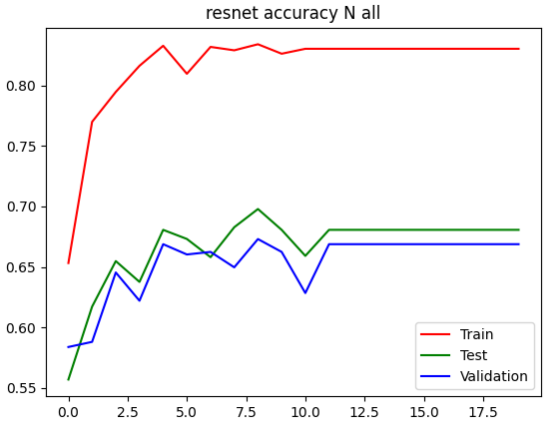



g) Kappa



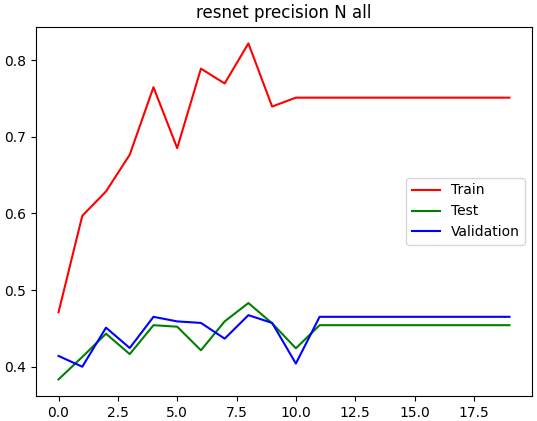

h) Final Skoru

**Şekil 4.3.** Normal sınıf kategorisine ait Inceptionv3 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)

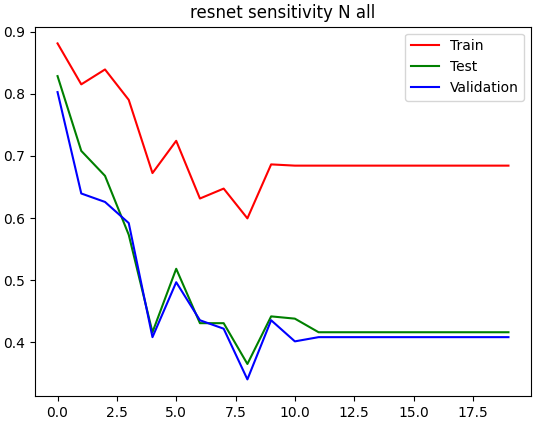



a) Doğruluk (Accuracy)


b) Hata (Loss)



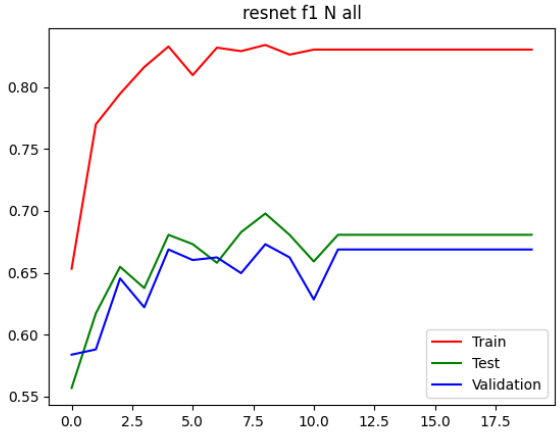

c) Kesinlik (Precision)



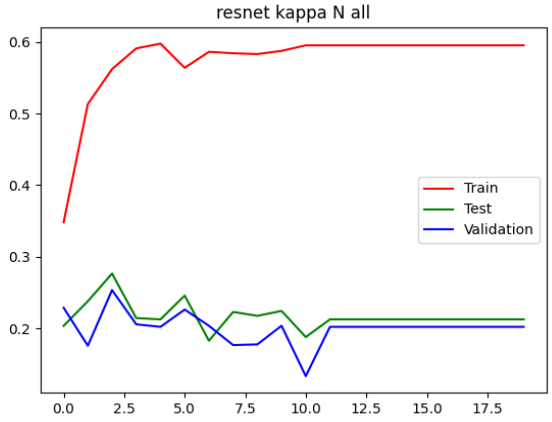

d) Duyarlılık (Recall)

**Şekil 4.4.** Normal sınıf kategorisine ait ResNet50 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi

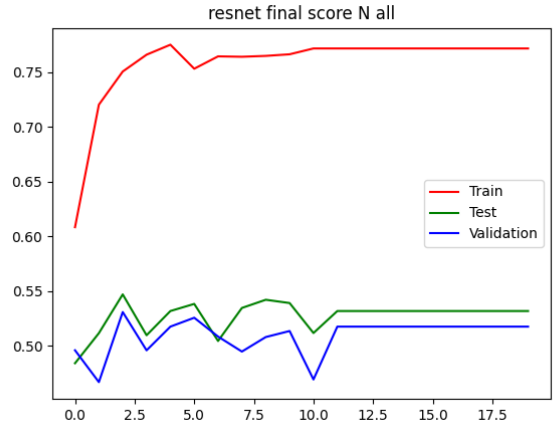

e) Eğri altında kalan alan (AUC)




f) F1 skoru




g) Kappa




h) Final Skoru

**Şekil 4.4.** Normal sınıf kategorisine ait ResNet50 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)

## 4.2. Diyabetik Retinopati Sınıf Kategorisine ait Derin Konvolüsyonel Sinir Ağı Mimari Modelleri Sınıflandırma Performans Bulguları

Eğitim, doğrulama ve test seti için VGG16, Inceptionv3, ResNet50 modellerine ait her bir epokta (epoch) hata (loss) ve doğruluk (accuracy) değerleri Tablo 4.3'deki gibi elde edilmiştir.

Eğitim, doğrulama ve test seti doğruluk değerleri sırası ile VGG16 modeli, 21 epok sonunda 0.735, 0.684 ve 0.668; Inceptionv3 modeli, 14 epok sonunda 0.994, 0.720 ve 0.686; ResNet50 modeli,15 epok sonunda 0.992, 0.701 ve 0.698 olarak bulunmuştur. ResNet50 modelinin test seti için en yüksek doğruluğa sahip olduğu görülmektedir.

Eğitim, doğrulama ve test seti hata değerleri sırası ile VGG16 modeli, 21 epok sonunda 0.461, 0.999 ve 1.218; Inceptionv3 modeli, 14 epok sonunda 0.024, 1.721 ve 2.211; ResNet50 modeli, 0.022, 2.263 ve 2.703 olarak bulunmuştur. VGG16 modelinin test seti için en düşük hata değerine sahip olduğu görülmektedir.

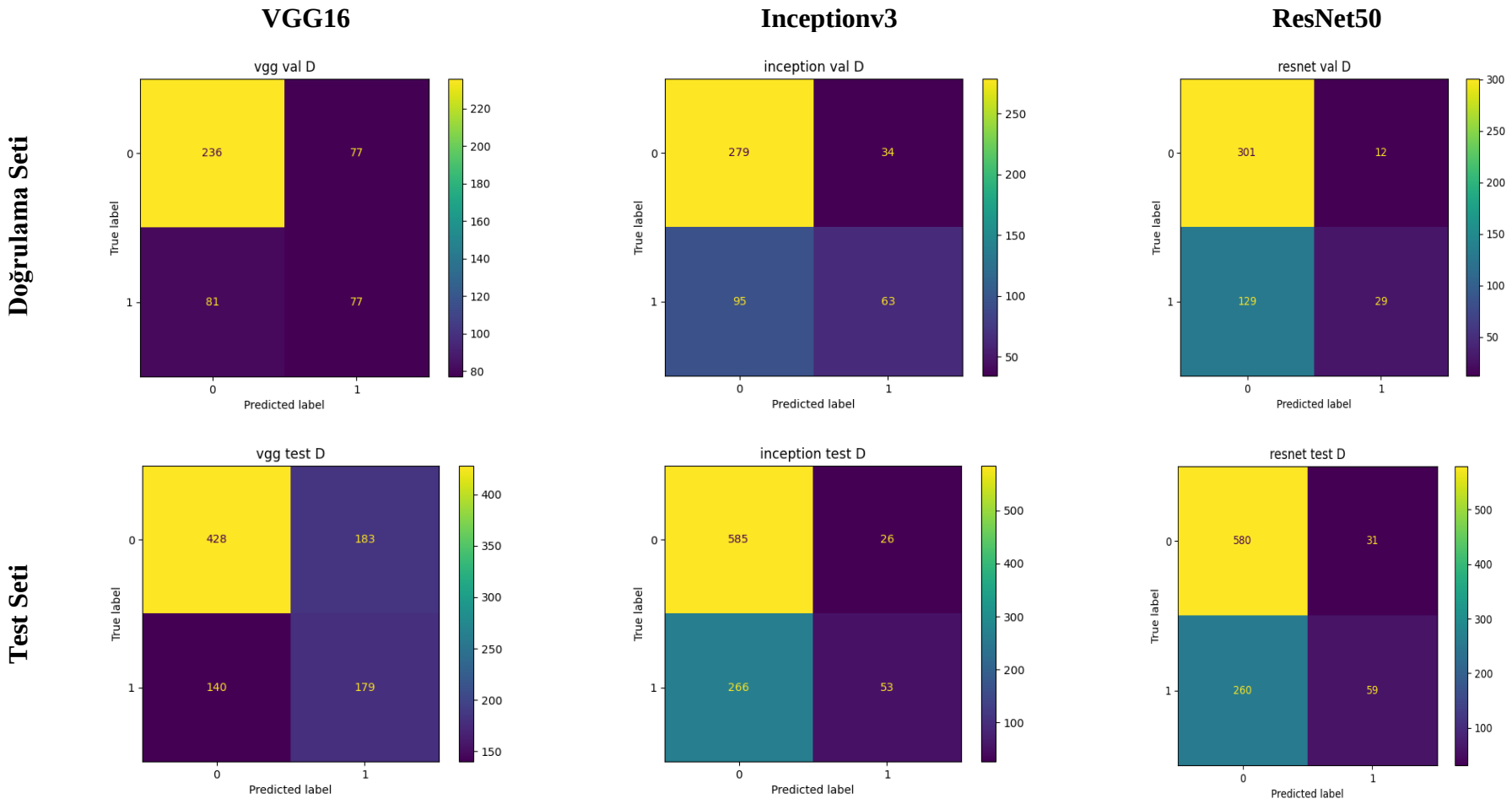
Doğrulama ve test seti için normal sınıf kategorisine ait karmaşıklık matrisi Şekil 4.5'de yer almaktadır. Doğrulama setinde toplam 471 fundus görüntüsünden 158'inin diyabetik retinopati sınıfına ait olduğu bilinmektedir. VGG16 modeli, 77; Inceptionv3 modeli, 63; ResNet50 modeli, 29'unu doğru olarak sınıflandırmıştır. Diyabetik retinopati olmayan sınıf kategorisine sahip 313 fundus görüntüsü bulunmaktadır. VGG16 modeli, 236; Inceptionv3 modeli, 279; ResNet50 modeli, 301'ini doğru olarak sınıflandırmıştır. Test setinde toplam 930 fundus görüntüsünden 274'ünün diyabetik retinopati sınıfına ait olduğu bilinmektedir. VGG16 modeli, 179; Inceptionv3 modeli, 53; ResNet50 modeli, 59"unu doğru olarak sınıflandırmıştır. Diyabetik retinopati olmayan sınıf kategorisine sahip 656 fundus görüntüsünden, VGG16 modeli, 428; Inceptionv3 modeli, 585; ResNet50 modeli, 580'ini doğru olarak sınıflandırmıştır.

Karmaşıklık matrisi elde edildikten sonra her bir modelin sınıflandırma başarı ölçütleri hesaplanarak Tablo 4.4'de sunulmuştur. VGG16, Inceptionv3, ResNet50 modelleri için her bir epokta doğruluk, hata, kesinlik, duyarlılık, Eğri altında kalan alan (AUC), F1 skoru, Kappa ve Final değeri sonuçları grafiksel olarak Şekil 4.6, Şekil 4.7, Şekil 4.8'de sunulmuştur.

Diyabetik retinopati sınıfı için test setinde VGG16, Inceptionv3, ResNet50 modelleri için Final skor değerleri sırası ile 0.528, 0.521 ve 0.497 olarak hesaplanmıştır. En yüksek Final skoru VGG16 modeli ile elde edilmiştir.

**Tablo 4.3.** Eğitim, doğrulama ve test seti için Diyabetik Retinopati Sınıf Kategorisine ait VGG16, Inceptionv3 ve ResNet50 Modeli Hata ve Doğruluk değerleri

| | | Eğitim seti | | Doğrulama seti | | Test seti | |
|---|---|---|---|---|---|---|---|
| Model | Epok | Hata (Loss) | Doğruluk (Accuracy) | Hata (Loss) | Doğruluk (Accuracy) | Hata (Loss) | Doğruluk (Accuracy) |
| VGG16 | 1 | 0,804 | 0,551 | 0,625 | 0,671 | 0,605 | 0,653 |
| | 2 | 0,708 | 0,588 | 0,629 | 0,616 | 0,620 | 0,625 |
| | 3 | 0,662 | 0,617 | 0,648 | 0,701 | 0,607 | 0,676 |
| | 4 | 0,646 | 0,637 | 0,587 | 0,641 | 0,619 | 0,646 |
| | 5 | 0,635 | 0,636 | 0,640 | 0,658 | 0,610 | 0,633 |
| | 6 | 0,592 | 0,666 | 0,649 | 0,656 | 0,631 | 0,629 |
| | 7 | 0,575 | 0,682 | 0,646 | 0,662 | 0,686 | 0,637 |
| | 8 | 0,560 | 0,691 | 0,672 | 0,682 | 0,745 | 0,637 |
| | 9 | 0,509 | 0,715 | 0,660 | 0,707 | 0,796 | 0,670 |
| | 10 | 0,514 | 0,714 | 0,652 | 0,715 | 0,757 | 0,670 |
| | 11 | 0,485 | 0,728 | 0,648 | 0,713 | 0,720 | 0,660 |
| | 12 | 0,48 | 0,734 | 0,783 | 0,679 | 0,935 | 0,655 |
| | 13 | 0,486 | 0,737 | 0,903 | 0,694 | 1,043 | 0,659 |
| | 14 | 0,469 | 0,740 | 0,737 | 0,720 | 0,882 | 0,670 |
| | 15 | 0,473 | 0,727 | 0,891 | 0,709 | 1,212 | 0,667 |
| | 16 | 0,460 | 0,745 | 0,888 | 0,675 | 1,107 | 0,652 |
| | 17 | 0,435 | 0,746 | 1,094 | 0,703 | 1,348 | 0,681 |
| | 18 | 0,450 | 0,738 | 0,828 | 0,673 | 0,887 | 0,645 |
| | 19 | 0,436 | 0,748 | 1,217 | 0,711 | 1,592 | 0,680 |
| | 20 | 0,440 | 0,747 | 1,021 | 0,665 | 1,140 | 0,637 |
| | 21 | 0,461 | 0,735 | 0,999 | 0,684 | 1,218 | 0,668 |
| Inceptionv3 | 1 | 0,703 | 0,751 | 0,689 | 0,718 | 0,812 | 0,714 |
| | 2 | 0,342 | 0,894 | 0,857 | 0,699 | 1,097 | 0,668 |
| | 3 | 0,177 | 0,950 | 1,016 | 0,684 | 1,201 | 0,678 |
| | 4 | 0,120 | 0,967 | 1,268 | 0,713 | 1,628 | 0,678 |
| | 5 | 0,087 | 0,977 | 1,501 | 0,701 | 1,866 | 0,687 |
| | 6 | 0,072 | 0,982 | 1,486 | 0,722 | 2,115 | 0,687 |
| | 7 | 0,062 | 0,986 | 1,369 | 0,749 | 2,066 | 0,699 |
| | 8 | 0,052 | 0,988 | 1,635 | 0,724 | 1,939 | 0,682 |
| | 9 | 0,049 | 0,989 | 1,549 | 0,749 | 2,121 | 0,685 |
| | 10 | 0,040 | 0,991 | 1,667 | 0,747 | 2,542 | 0,704 |
| | 11 | 0,043 | 0,990 | 1,777 | 0,718 | 2,298 | 0,690 |
| | 12 | 0,041 | 0,990 | 1,976 | 0,720 | 2,363 | 0,688 |
| | 13 | 0,035 | 0,992 | 2,007 | 0,737 | 2,423 | 0,698 |
| | 14 | 0,024 | 0,994 | 1,721 | 0,720 | 2,211 | 0,686 |
| ResNet50 | 1 | 0,506 | 0,748 | 0,774 | 0,722 | 0,898 | 0,704 |
| | 2 | 0,202 | 0,915 | 1,103 | 0,705 | 1,301 | 0,710 |
| | 3 | 0,114 | 0,954 | 1,358 | 0,711 | 1,636 | 0,711 |
| | 4 | 0,078 | 0,970 | 1,340 | 0,696 | 1,587 | 0,700 |
| | 5 | 0,057 | 0,980 | 1,527 | 0,709 | 1,861 | 0,696 |
| | 6 | 0,046 | 0,984 | 1,497 | 0,730 | 1,750 | 0,695 |
| | 7 | 0,038 | 0,987 | 1,475 | 0,720 | 1,817 | 0,694 |
| | 8 | 0,040 | 0,985 | 1,674 | 0,735 | 1,844 | 0,710 |
| | 9 | 0,031 | 0,989 | 1,673 | 0,728 | 2,014 | 0,725 |
| | 10 | 0,033 | 0,989 | 1,927 | 0,711 | 2,137 | 0,710 |
| | 11 | 0,021 | 0,993 | 1,698 | 0,724 | 1,911 | 0,704 |
| | 12 | 0,025 | 0,991 | 1,789 | 0,690 | 1,872 | 0,687 |
| | 13 | 0,025 | 0,992 | 1,975 | 0,656 | 1,990 | 0,663 |
| | 14 | 0,021 | 0,993 | 1,818 | 0,711 | 2,039 | 0,712 |
| | 15 | 0,022 | 0,992 | 2,263 | 0,701 | 2,703 | 0,698 |

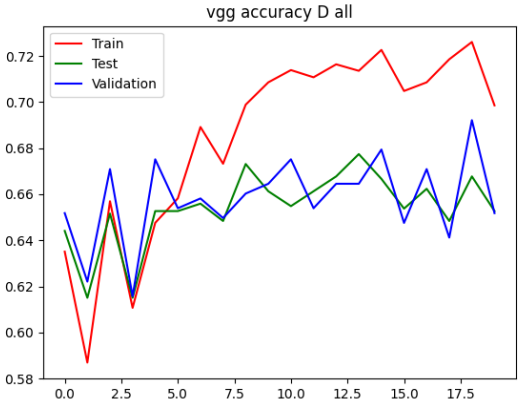



**Şekil 4.5.** Doğrulama ve test seti için VGG16, Inceptionv3 ve ResNet50 Modeli Diyabetik Retinopati Sınıf Kategorisine ait Karmaşıklık Matrisi

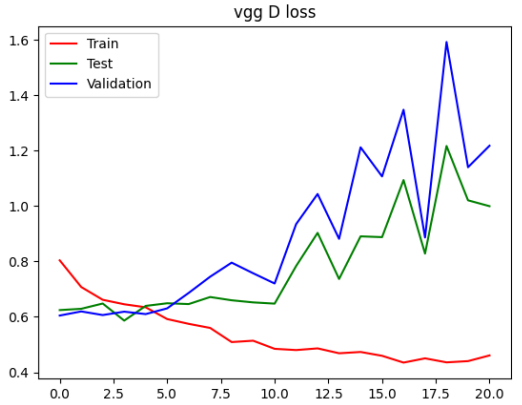
**Tablo 4.4.** Doğrulama ve test seti için Diyabetik Retinopati Sınıf Kategorisine ait VGG16, Inceptionv3 ve ResNet50 Modeli sınıflandırma başarı ölçütleri

| Model | Performans ölçütleri | Doğrulama seti | Test seti |
|---|---|---|---|
| VGG16 | Accuracy | 0,665 | 0,653 |
| | Precision | 0,500 | 0,494 |
| | Sens | 0,487 | 0,561 |
| | Spec | 0,754 | 0,700 |
| | $\kappa$ | 0,243 | 0,253 |
| | F1 | 0,665 | 0,653 |
| | AUC | 0,709 | 0,677 |
| | Final | 0,539 | 0,528 |
| Inceptionv3 | Accuracy | 0,726 | 0,686 |
| | Precision | 0,649 | 0,671 |
| | Sens | 0,399 | 0,166 |
| | Spec | 0,891 | 0,957 |
| | $\kappa$ | 0,321 | 0,151 |
| | F1 | 0,726 | 0,686 |
| | AUC | 0,758 | 0,726 |
| | Final | 0,602 | 0,521 |
| ResNet50 | Accuracy | 0,701 | 0,687 |
| | Precision | 0,707 | 0,656 |
| | Sens | 0,184 | 0,185 |
| | Spec | 0,962 | 0,949 |
| | $\kappa$ | 0,178 | 0,162 |
| | F1 | 0,701 | 0,687 |
| | AUC | 0,695 | 0,643 |
| | Final | 0,525 | 0,497 |

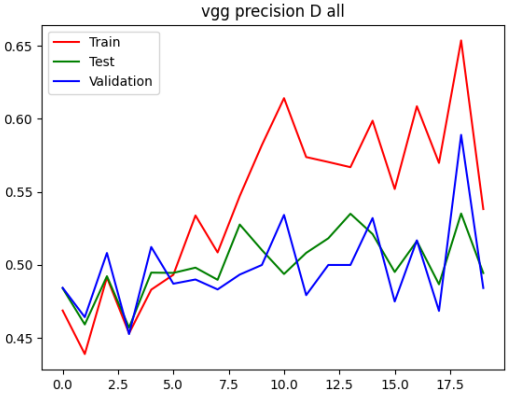
Accuracy: Doğruluk; Precision: Kesinlik; Sens: Duyarlılık; Spec: Özgüllük; $\kappa$: Kappa; AUC: Eğri altında kalan alan; Final: F1, AUC ve $\kappa$ değeri ortalaması



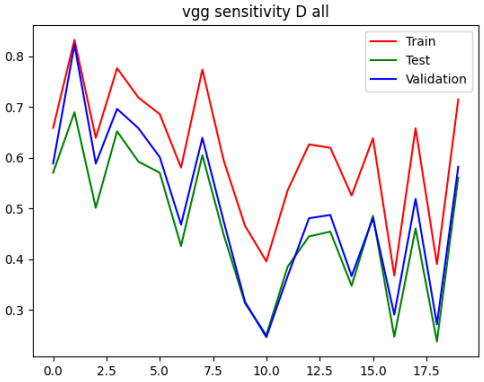

a) Doğruluk (Accuracy)




b) Hata (Loss)

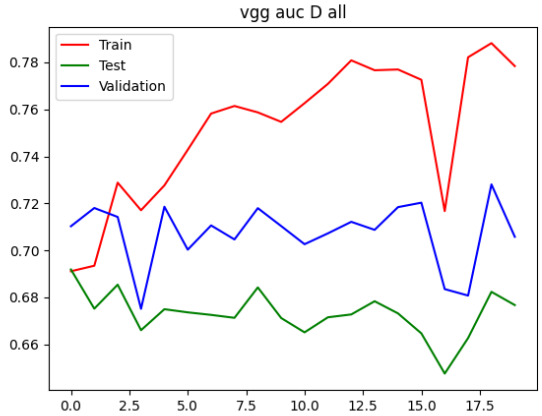



c) Kesinlik (Precision)

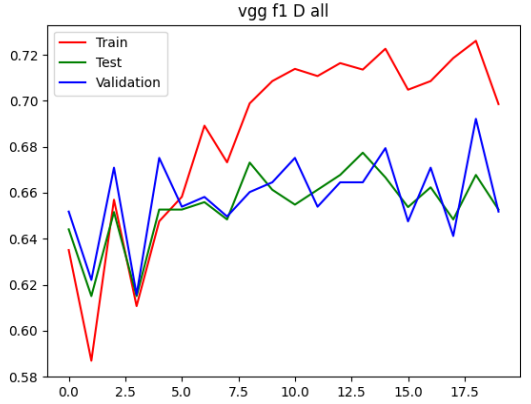



d) Duyarlılık (Recall)

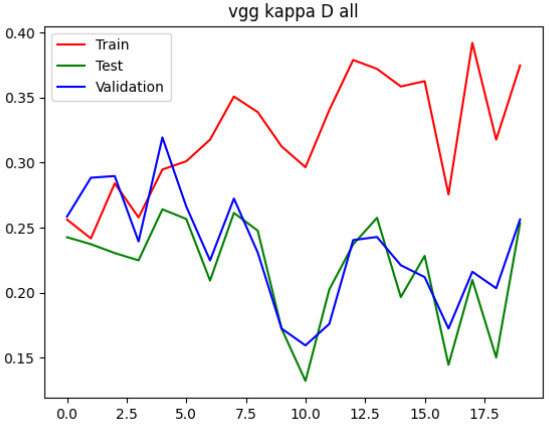
**Şekil 4.6.** Diyabetik Retinopati sınıf kategorisine ait VGG16 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi



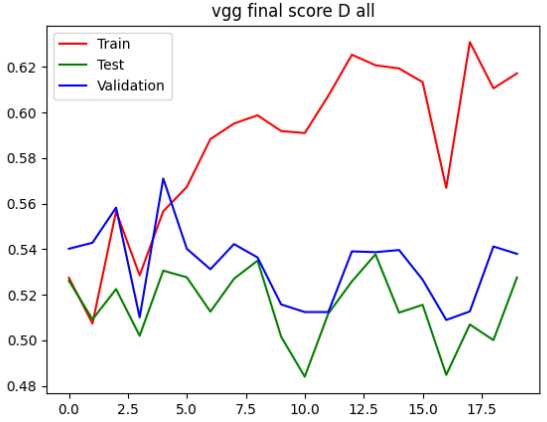

e) Eğri altında kalan alan (AUC)




f) F1 skoru

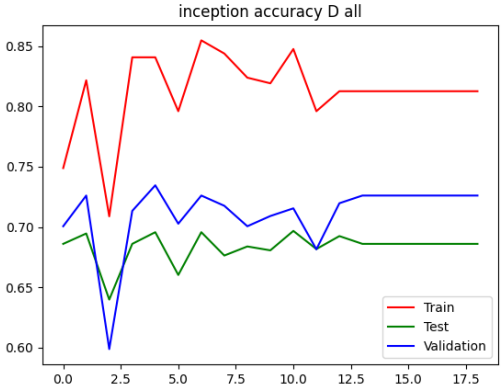



g) Kappa

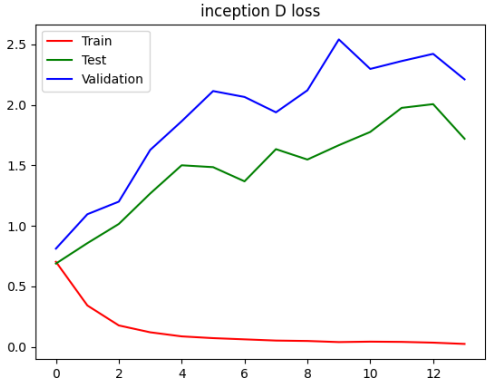



h) Final Skoru

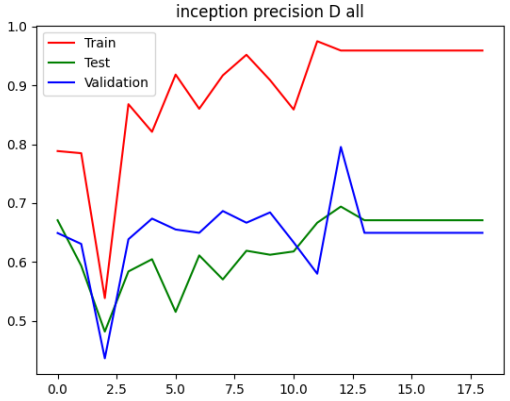
**Şekil 4.6.** Diyabetik Retinopati sınıf kategorisine ait VGG16 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)



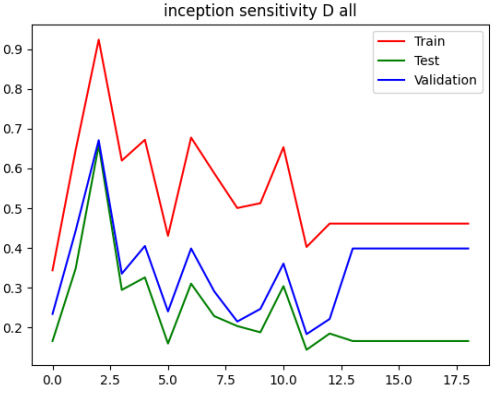

a) Doğruluk (Accuracy)




b) Hata (Loss)

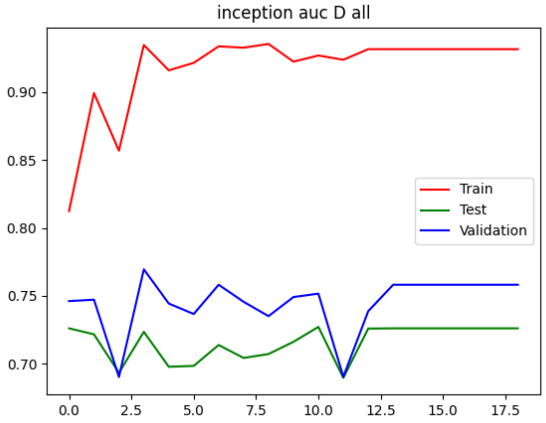



c) Kesinlik (Precision)

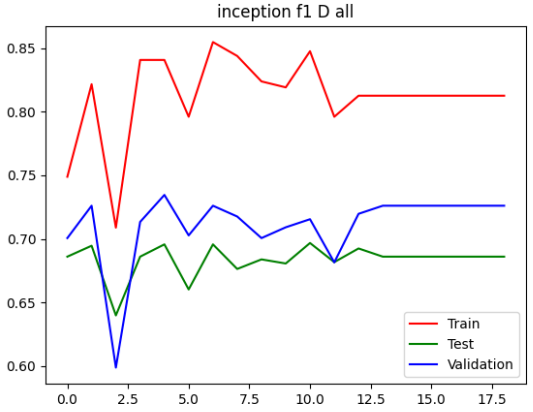



d) Duyarlılık (Recall)

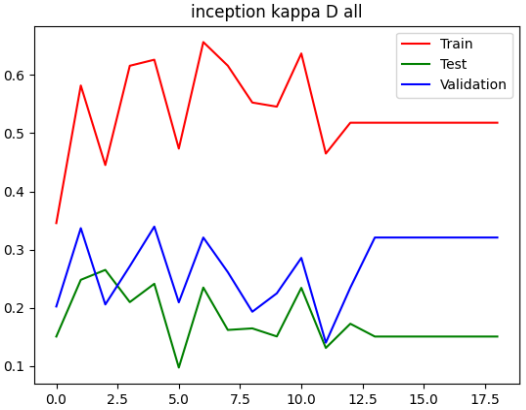
**Şekil 4.7.** Diyabetik Retinopati sınıf kategorisine ait Inceptionv3 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi



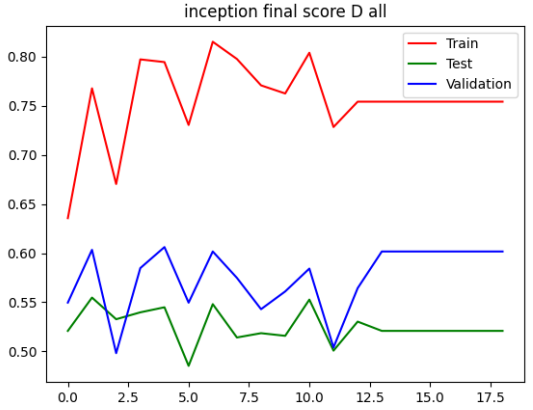

e) Eğri altında kalan alan (AUC)




f) F1 skoru

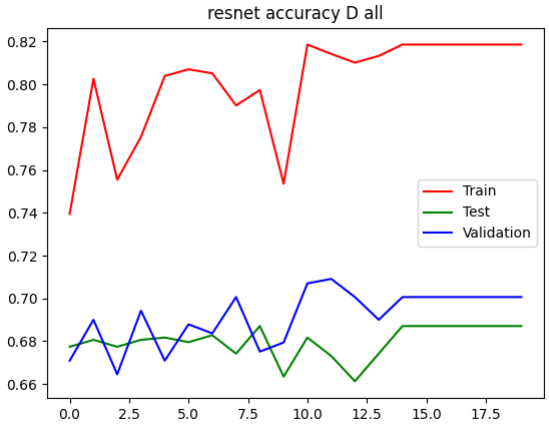



g) Kappa

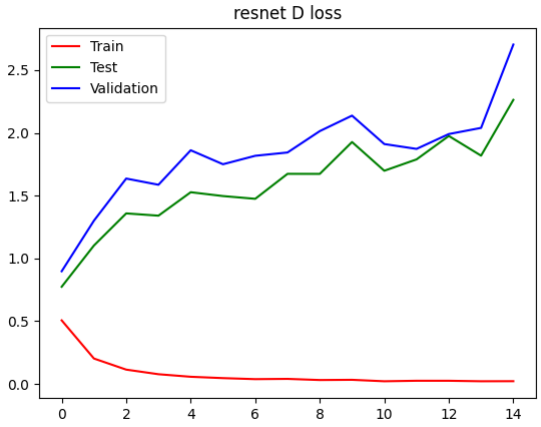



h) Final Skoru

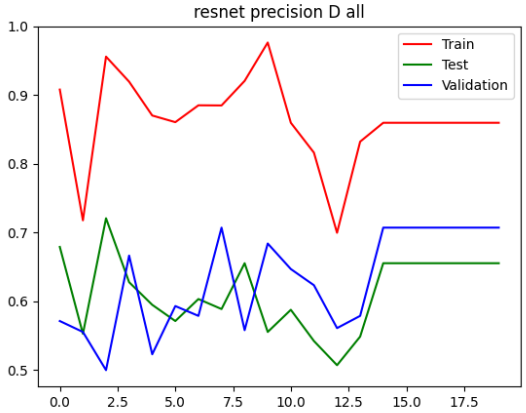
**Şekil 4.7.** Diyabetik Retinopati sınıf kategorisine ait Inceptionv3 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)



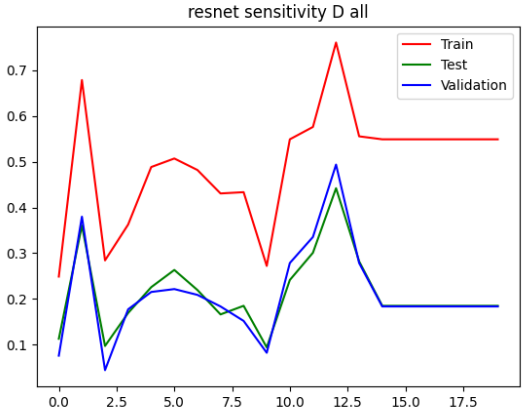

a) Doğruluk (Accuracy)




b) Hata (Loss)

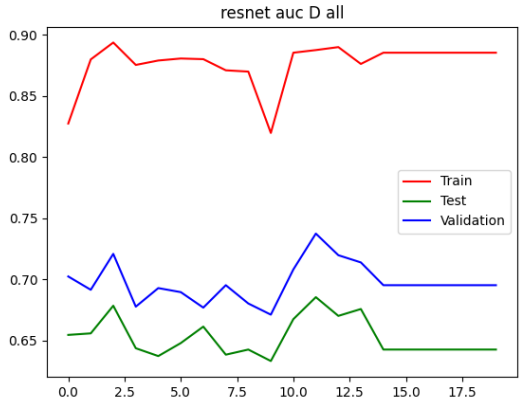



c) Kesinlik (Precision)

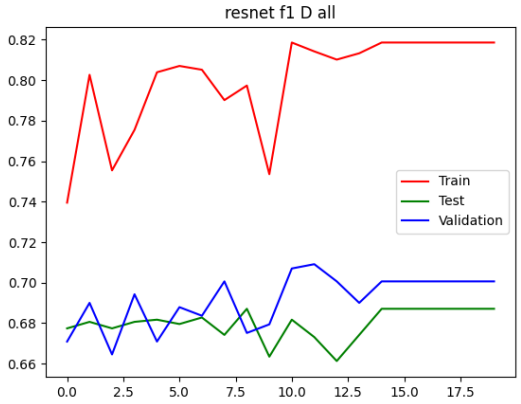



d) Duyarlılık (Recall)

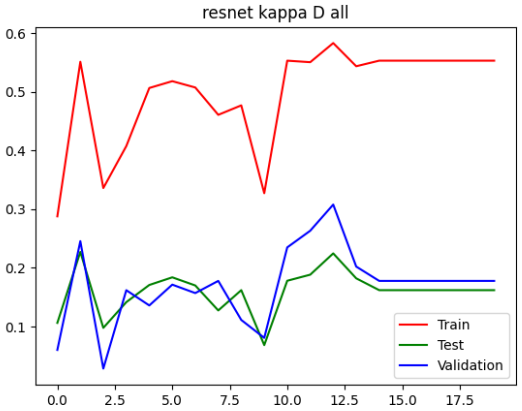
**Şekil 4.8.** Diyabetik Retinopati sınıf kategorisine ait ResNet50 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi



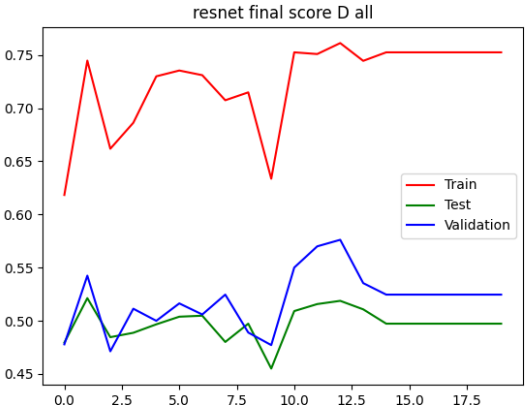

e) Eğri altında kalan alan (AUC)




f) F1 skoru




g) Kappa




h) Final Skoru

**Şekil 4.8.** Diyabetik Retinopati sınıf kategorisine ait ResNet50 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)

## 4.3. Glokom Sınıf Kategorisine ait Derin Konvolüsyonel Sinir Ağı Mimari Modelleri Sınıflandırma Performans Bulguları

Eğitim, doğrulama ve test seti için VGG16, Inceptionv3, ResNet50 modellerine ait her bir epokta (epoch) hata (loss) ve doğruluk (accuracy) değerleri Tablo 4.5'deki gibi elde edilmiştir.

Eğitim, doğrulama ve test seti doğruluk değerleri sırası ile VGG16 modeli, 14 epok sonunda 0.954, 0.892 ve 0.913; Inceptionv3 modeli, 17 epok sonunda 0.997, 0.915 ve 0.915; ResNet50 modeli, 21 epok sonunda 0.999, 0.943 ve 0.941 olarak bulunmuştur. ResNet50 modelinin test seti için en yüksek doğruluğa sahip olduğu görülmektedir.

Eğitim, doğrulama ve test seti hata değerleri sırası ile VGG16 modeli, 14 epok sonunda 0.171, 0.845 ve 0.831; Inceptionv3 modeli, 17 epok sonunda 0.013, 0.863 ve 1.209; ResNet50 modeli, 0.004, 0.876 ve 0.809 olarak bulunmuştur. ResNet50 modelinin test seti için en düşük hata değerine sahip olduğu görülmektedir.

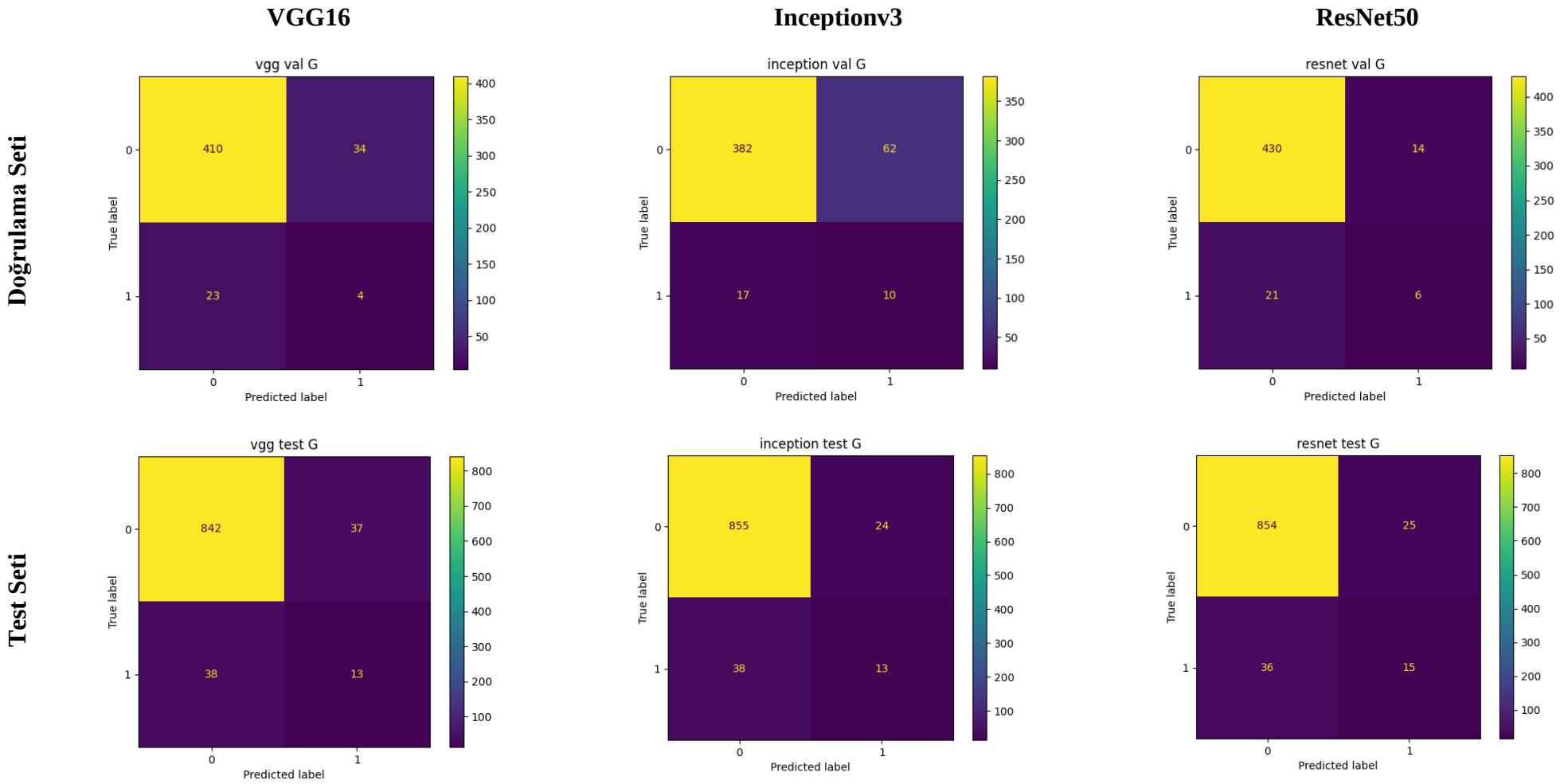
Doğrulama ve test seti için glokom sınıf kategorisine ait karmaşıklık matrisi Şekil 4.9'da yer almaktadır. Doğrulama setinde toplam 471 fundus görüntüsünden 27'sinin glokom sınıfına ait olduğu bilinmektedir. VGG16 modeli, 4; Inceptionv3 modeli, 10; ResNet50 modeli, 6'sını doğru olarak sınıflandırmıştır. Glokom olmayan sınıf kategorisine sahip 444 fundus görüntüsü bulunmaktadır. VGG16 modeli, 410; Inceptionv3 modeli, 382; ResNet50 modeli, 430'unu doğru olarak sınıflandırmıştır. Test setinde toplam 930 fundus görüntüsünden 51'inin glokom sınıfına ait olduğu bilinmektedir. VGG16 modeli, 13; Inceptionv3 modeli, 13; ResNet50 modeli, 15"ini doğru olarak sınıflandırmıştır. Glokom olmayan sınıf kategorisine sahip 879 fundus görüntüsünden, VGG16 modeli, 842; Inceptionv3 modeli, 855; ResNet50 modeli, 854'ünü doğru olarak sınıflandırmıştır.

Karmaşıklık matrisi elde edildikten sonra her bir modelin sınıflandırma başarı ölçütleri hesaplanarak Tablo 4.6'da sunulmuştur. VGG16, Inceptionv3, ResNet50 modelleri için her bir epokta doğruluk, hata, kesinlik, duyarlılık, Eğri altında kalan alan (AUC), F1 skoru, Kappa ve Final değeri sonuçları grafiksel olarak Şekil 4.10, Şekil 4.11, Şekil 4.12'de sunulmuştur.

Glokom sınıfı test setinde VGG16, Inceptionv3, ResNet50 modelleri için Final skor değerleri sırası ile 0.620, 0.626 ve 0.664 olarak hesaplanmıştır. En yüksek Final skoru ResNet50 modeli ile elde edilmiştir.

**Tablo 4.5.** Eğitim, doğrulama ve test seti için Glokom Sınıf Kategorisine ait VGG16, Inceptionv3 ve ResNet50 Modeli Hata ve Doğruluk değerleri

| | | Eğitim seti | | Doğrulama seti | | Test seti | |
|---|---|---|---|---|---|---|---|
| Model | Epok | Hata (Loss) | Doğruluk (Accuracy) | Hata (Loss) | Doğruluk (Accuracy) | Hata (Loss) | Doğruluk (Accuracy) |
| VGG16 | 1 | 0,554 | 0,785 | 0,653 | 0,688 | 0,636 | 0,700 |
| | 2 | 0,393 | 0,852 | 0,412 | 0,860 | 0,415 | 0,881 |
| | 3 | 0,296 | 0,886 | 0,423 | 0,839 | 0,436 | 0,871 |
| | 4 | 0,282 | 0,902 | 0,518 | 0,870 | 0,478 | 0,908 |
| | 5 | 0,264 | 0,906 | 0,488 | 0,866 | 0,441 | 0,890 |
| | 6 | 0,230 | 0,922 | 0,551 | 0,851 | 0,518 | 0,878 |
| | 7 | 0,217 | 0,927 | 0,493 | 0,902 | 0,432 | 0,929 |
| | 8 | 0,226 | 0,921 | 0,851 | 0,858 | 0,812 | 0,901 |
| | 9 | 0,216 | 0,937 | 0,913 | 0,883 | 0,844 | 0,908 |
| | 10 | 0,215 | 0,934 | 0,618 | 0,902 | 0,616 | 0,922 |
| | 11 | 0,203 | 0,939 | 1,025 | 0,875 | 0,955 | 0,910 |
| | 12 | 0,178 | 0,945 | 0,910 | 0,883 | 0,903 | 0,902 |
| | 13 | 0,215 | 0,941 | 0,691 | 0,881 | 0,694 | 0,918 |
| | 14 | 0,171 | 0,954 | 0,845 | 0,892 | 0,831 | 0,913 |
| Inceptionv3 | 1 | 0,256 | 0,924 | 0,452 | 0,898 | 0,413 | 0,920 |
| | 2 | 0,088 | 0,976 | 0,475 | 0,898 | 0,437 | 0,922 |
| | 3 | 0,057 | 0,986 | 0,399 | 0,932 | 0,415 | 0,938 |
| | 4 | 0,047 | 0,988 | 0,838 | 0,890 | 0,785 | 0,904 |
| | 5 | 0,037 | 0,991 | 0,588 | 0,924 | 0,639 | 0,926 |
| | 6 | 0,033 | 0,993 | 0,704 | 0,907 | 0,647 | 0,930 |
| | 7 | 0,021 | 0,995 | 0,698 | 0,921 | 0,620 | 0,938 |
| | 8 | 0,020 | 0,995 | 0,811 | 0,909 | 0,769 | 0,926 |
| | 9 | 0,022 | 0,995 | 0,708 | 0,919 | 0,707 | 0,942 |
| | 10 | 0,018 | 0,996 | 0,769 | 0,934 | 0,684 | 0,951 |
| | 11 | 0,021 | 0,996 | 0,653 | 0,917 | 0,584 | 0,935 |
| | 12 | 0,010 | 0,998 | 0,874 | 0,915 | 0,795 | 0,939 |
| | 13 | 0,010 | 0,998 | 0,671 | 0,917 | 0,821 | 0,928 |
| | 14 | 0,012 | 0,998 | 0,735 | 0,926 | 0,677 | 0,943 |
| | 15 | 0,011 | 0,998 | 0,908 | 0,930 | 1,013 | 0,943 |
| | 16 | 0,008 | 0,998 | 0,962 | 0,921 | 0,891 | 0,935 |
| | 17 | 0,013 | 0,997 | 0,863 | 0,915 | 1,209 | 0,915 |
| ResNet50 | 1 | 0,200 | 0,916 | 0,405 | 0,913 | 0,326 | 0,929 |
| | 2 | 0,067 | 0,973 | 0,392 | 0,917 | 0,379 | 0,930 |
| | 3 | 0,046 | 0,983 | 0,418 | 0,917 | 0,402 | 0,933 |
| | 4 | 0,033 | 0,989 | 0,473 | 0,938 | 0,441 | 0,939 |
| | 5 | 0,027 | 0,99 | 0,621 | 0,904 | 0,514 | 0,920 |
| | 6 | 0,018 | 0,994 | 0,742 | 0,917 | 0,734 | 0,912 |
| | 7 | 0,014 | 0,996 | 1,502 | 0,830 | 1,205 | 0,857 |
| | 8 | 0,021 | 0,993 | 0,661 | 0,913 | 0,639 | 0,941 |
| | 9 | 0,014 | 0,996 | 0,504 | 0,909 | 0,568 | 0,923 |
| | 10 | 0,012 | 0,996 | 0,603 | 0,941 | 0,599 | 0,947 |
| | 11 | 0,015 | 0,995 | 0,728 | 0,934 | 0,761 | 0,945 |
| | 12 | 0,010 | 0,997 | 0,495 | 0,921 | 0,56 | 0,937 |
| | 13 | 0,010 | 0,996 | 0,715 | 0,921 | 0,648 | 0,927 |
| | 14 | 0,009 | 0,997 | 0,642 | 0,943 | 0,636 | 0,946 |
| | 15 | 0,014 | 0,996 | 0,541 | 0,926 | 0,525 | 0,948 |
| | 16 | 0,005 | 0,998 | 0,668 | 0,941 | 0,67 | 0,945 |
| | 17 | 0,007 | 0,998 | 1,504 | 0,862 | 1,433 | 0,884 |
| | 18 | 0,010 | 0,997 | 0,886 | 0,917 | 0,749 | 0,929 |
| | 19 | 0,003 | 0,999 | 0,813 | 0,938 | 0,776 | 0,943 |
| | 20 | 0,006 | 0,998 | 0,933 | 0,943 | 0,842 | 0,933 |
| | 21 | 0,004 | 0,999 | 0,876 | 0,943 | 0,809 | 0,941 |

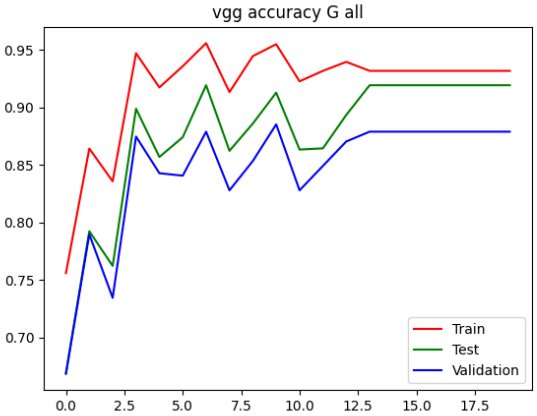



**Şekil 4.9.** Doğrulama ve test seti için VGG16, Inceptionv3 ve ResNet50 Modeli Glokom Sınıf Kategorisine ait Karmaşıklık Matrisi

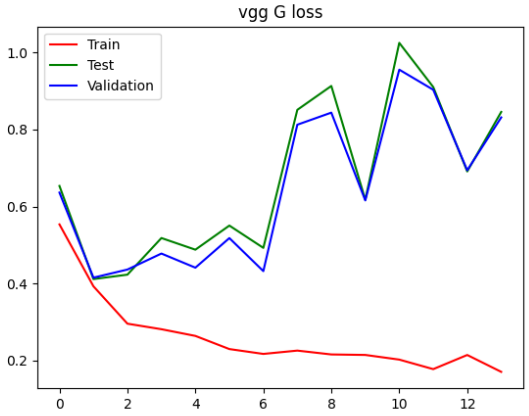
**Tablo 4.6.** Doğrulama ve test seti için Glokom Sınıf Kategorisine ait VGG16, Inceptionv3 ve ResNet50 Modeli sınıflandırma başarı ölçütleri

| Model | Performans ölçütleri | Doğrulama seti | Test seti |
|---|---|---|---|
| VGG16 | Accuracy | 0,879 | 0,919 |
| | Precision | 0,105 | 0,260 |
| | Sens | 0,148 | 0,255 |
| | Spec | 0,923 | 0,958 |
| | $\kappa$ | 0,060 | 0,215 |
| | F1 | 0,879 | 0,919 |
| | AUC | 0,719 | 0,726 |
| | Final | 0,553 | 0,620 |
| Inceptionv3 | Accuracy | 0,832 | 0,933 |
| | Precision | 0,139 | 0,351 |
| | Sens | 0,370 | 0,255 |
| | Spec | 0,860 | 0,973 |
| | $\kappa$ | 0,129 | 0,261 |
| | F1 | 0,832 | 0,933 |
| | AUC | 0,662 | 0,685 |
| | Final | 0,541 | 0,626 |
| ResNet50 | Accuracy | 0,926 | 0,934 |
| | Precision | 0,300 | 0,375 |
| | Sens | 0,222 | 0,294 |
| | Spec | 0,968 | 0,972 |
| | $\kappa$ | 0,217 | 0,296 |
| | F1 | 0,926 | 0,934 |
| | AUC | 0,678 | 0,762 |
| | Final | 0,607 | 0,664 |

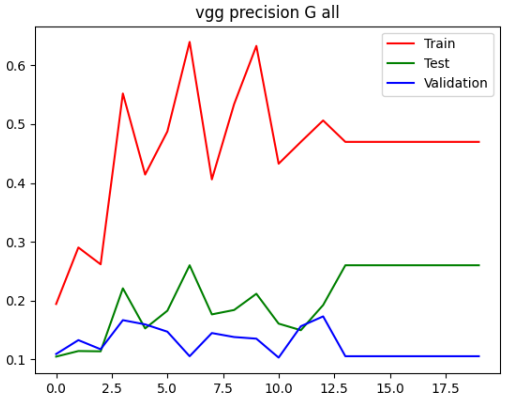
Accuracy: Doğruluk; Precision: Kesinlik; Sens: Duyarlılık; Spec: Özgüllük; $\kappa$: Kappa; AUC: Eğri altında kalan alan; Final: F1, AUC ve $\kappa$ değeri ortalaması



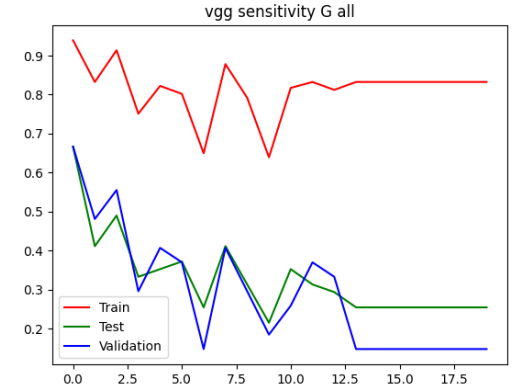

a) Doğruluk (Accuracy)




b) Hata (Loss)

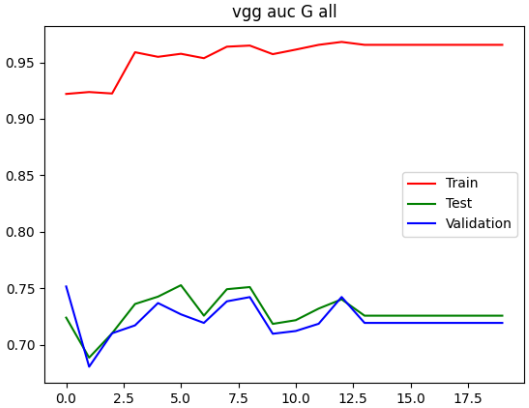



c) Kesinlik (Precision)

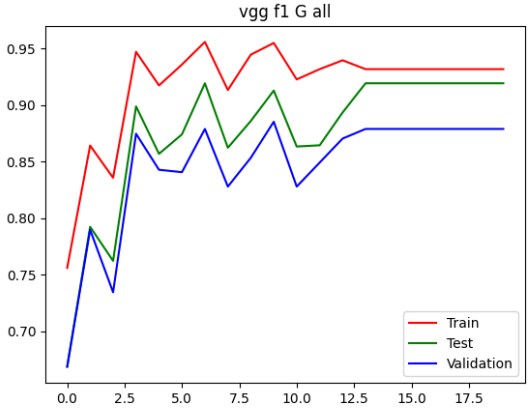



d) Duyarlılık (Recall)

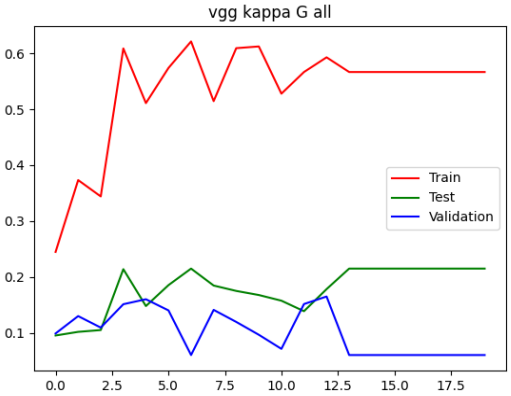
**Şekil 4.10.** Glokom sınıf kategorisine ait VGG16 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi



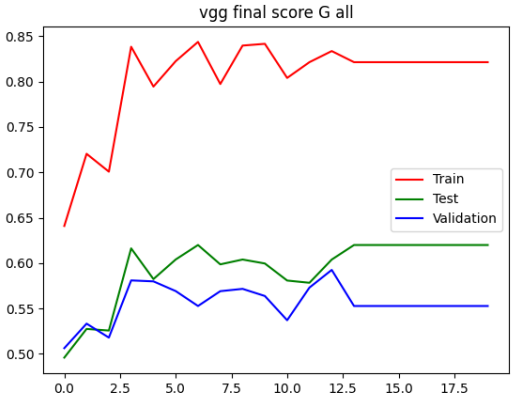

e) Eğri altında kalan alan (AUC)




f) F1 skoru

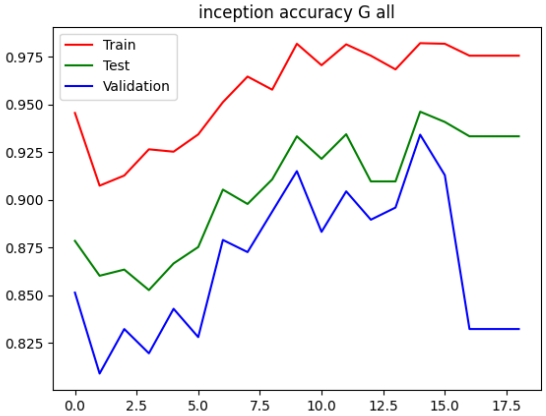



g) Kappa



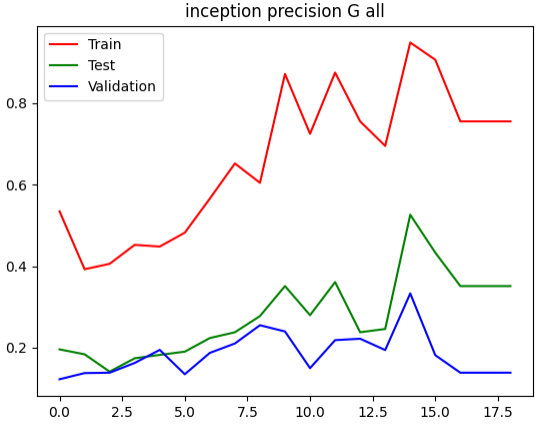

h) Final Skoru

**Şekil 4.10.** Glokom sınıf kategorisine ait VGG16 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)

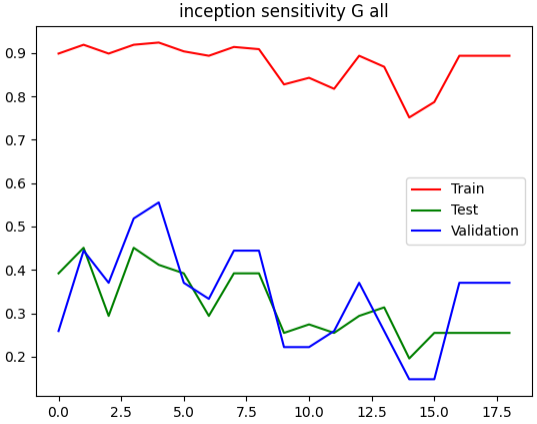



a) Doğruluk (Accuracy)


b) Hata (Loss)

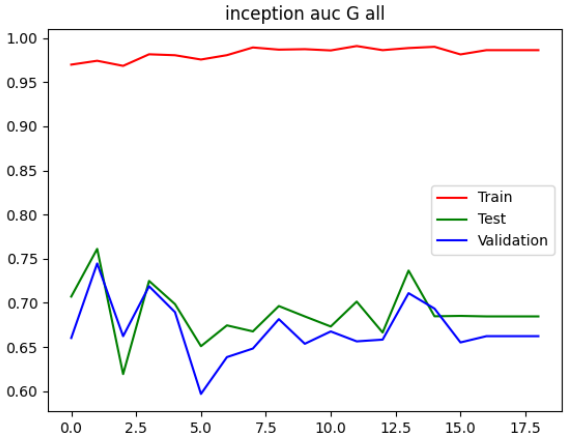



c) Kesinlik (Precision)

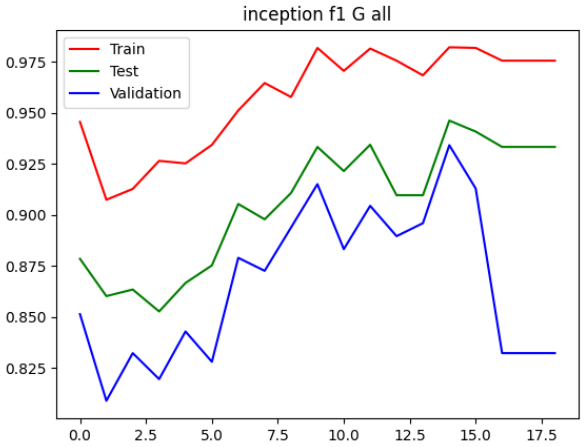



d) Duyarlılık (Recall)

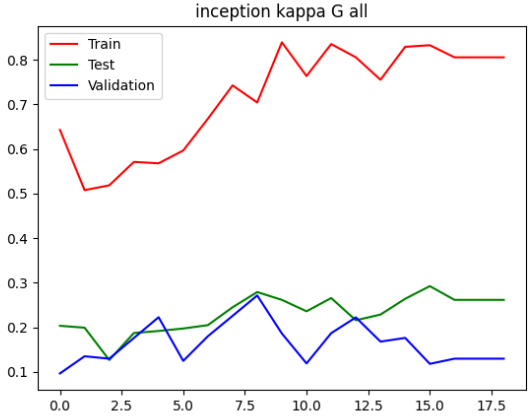
**Şekil 4.11.** Glokom sınıf kategorisine ait Inceptionv3 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi



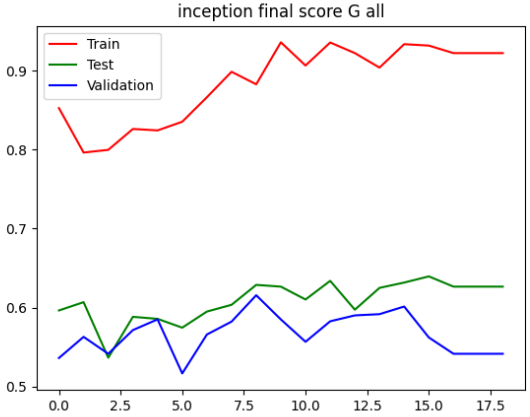

e) Eğri altında kalan alan (AUC)




f) F1 skoru

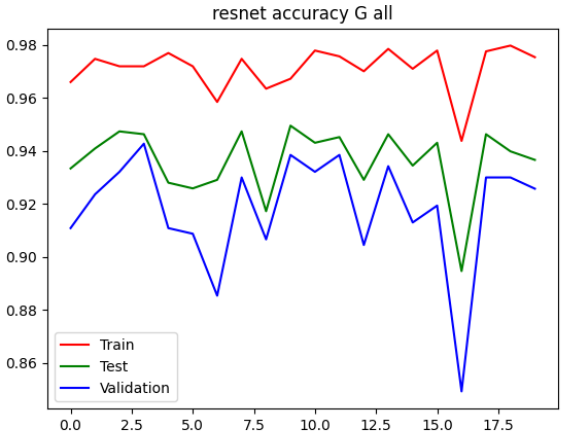



g) Kappa

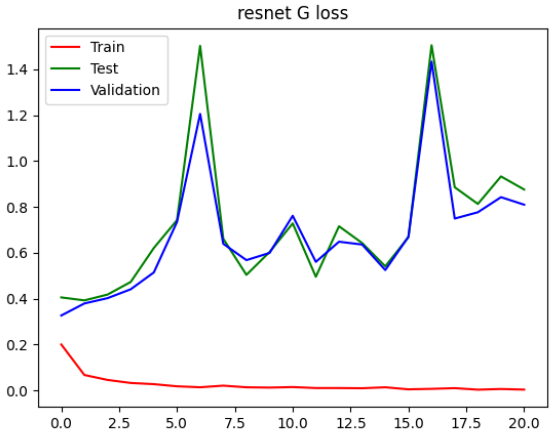



h) Final Skoru

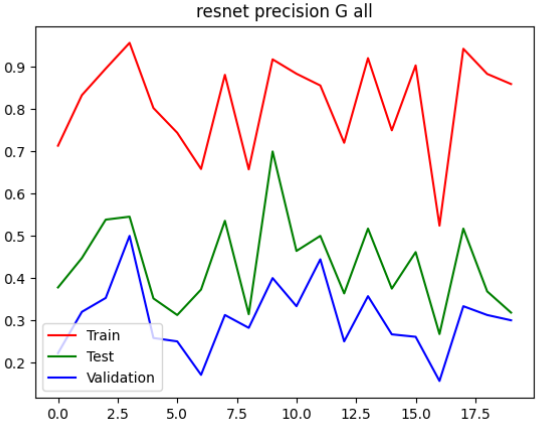
**Şekil 4.11.** Glokom sınıf kategorisine ait Inceptionv3 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)



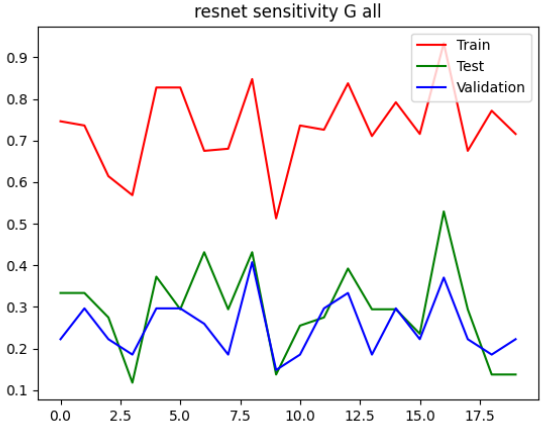

a) Doğruluk (Accuracy)




b) Hata (Loss)

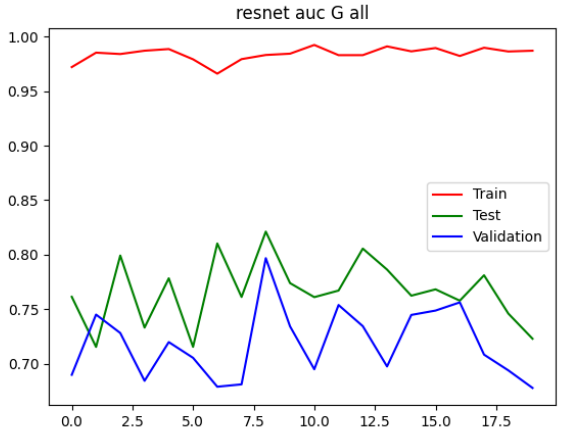



c) Kesinlik (Precision)

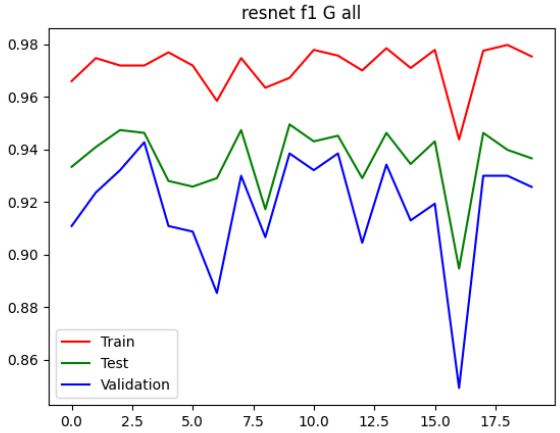



d) Duyarlılık (Recall)

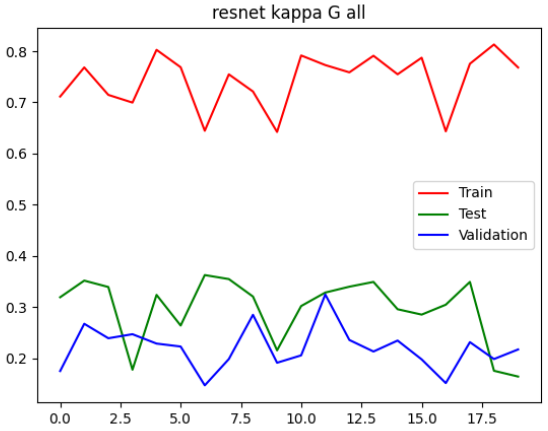
**Şekil 4.12.** Glokom sınıf kategorisine ait ResNet50 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi




e) Eğri altında kalan alan (AUC)




f) F1 skoru




g) Kappa


h) Final Skoru

**Şekil 4.12.** Glokom sınıf kategorisine ait ResNet50 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)

## 4.4. Katarakt Sınıf Kategorisine ait Derin Konvolüsyonel Sinir Ağı Mimari Modelleri Sınıflandırma Performans Bulguları

Eğitim, doğrulama ve test seti için VGG16, Inceptionv3, ResNet50 modellerine ait her bir epokta (epoch) hata (loss) ve doğruluk (accuracy) değerleri Tablo 4.7'deki gibi elde edilmiştir. Eğitim, doğrulama ve test seti doğruluk değerleri sırası ile VGG16 modeli, 14 epok sonunda 0.981, 0.968 ve 0.962; Inceptionv3 modeli, 10 epok sonunda 0.998, 0.968 ve 0.970; ResNet50 modeli, 10 epok sonunda 0.999, 0.981 ve 0.967 olarak bulunmuştur. Inceptionv3 modelinin test seti için en yüksek doğruluğa sahip olduğu görülmektedir.

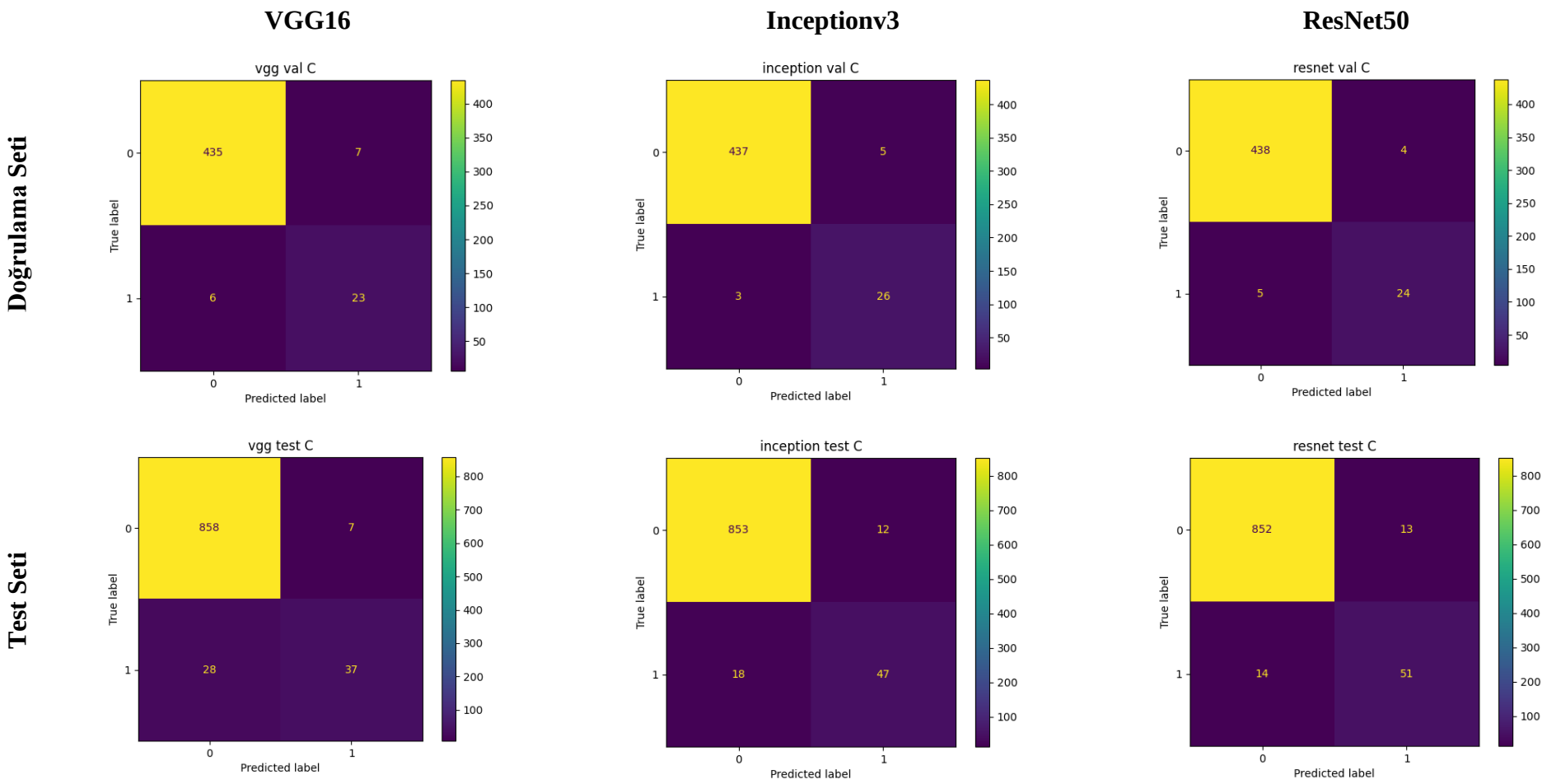
Eğitim, doğrulama ve test seti hata değerleri sırası ile VGG16 modeli, 14 epok sonunda 0.154, 0.129 ve 0.688; Inceptionv3 modeli, 10 epok sonunda 0.006, 0.237 ve 0.278; ResNet50 modeli, 0.001, 0.428 ve 0.407 olarak bulunmuştur. Inceptionv3 modelinin test seti için en düşük hata değerine sahip olduğu görülmektedir.

Doğrulama ve test seti için katarakt sınıf kategorisine ait karmaşıklık matrisi Şekil 4.13'de yer almaktadır. Doğrulama setinde, toplam 471 fundus görüntüsünden 29'unun katarakt sınıfına ait olduğu bilinmektedir. VGG16 modeli, 23; Inceptionv3 modeli, 26; ResNet50 modeli, 24'ünü doğru olarak sınıflandırmıştır. Katarakt olmayan sınıf kategorisine sahip 442 fundus görüntüsü bulunmaktadır. VGG16 modeli, 435; Inceptionv3 modeli, 437; ResNet50 modeli, 438'ini doğru olarak sınıflandırmıştır. Test setinde toplam 930 fundus görüntüsünden 65'inin katarakt sınıfına ait olduğu bilinmektedir. VGG16 modeli, 37; Inceptionv3 modeli, 47; ResNet50 modeli, 51"ini doğru olarak sınıflandırmıştır. Katarakt olmayan sınıf kategorisine sahip 865 fundus görüntüsünden, VGG16 modeli, 858; Inceptionv3 modeli, 853; ResNet50 modeli, 852'sini doğru olarak sınıflandırmıştır.

Karmaşıklık matrisi elde edildikten sonra her bir modelin sınıflandırma başarı ölçütleri hesaplanarak Tablo 4.8'de sunulmuştur. VGG16, Inceptionv3, ResNet50 modelleri için her bir epokta doğruluk, hata, kesinlik, duyarlılık, Eğri altında kalan alan (AUC), F1 skoru, Kappa ve Final değeri sonuçları grafiksel olarak Şekil 4.14, Şekil 4.15, Şekil 4.16'da sunulmuştur. Katarakt sınıfı test setinde VGG16, Inceptionv3, ResNet50 modelleri için Final skor değerleri sırası ile 0.857, 0.885 ve 0.903 olarak hesaplanmıştır. En yüksek Final skoru ResNet50 modeli ile elde edilmiştir.

**Tablo 4.7.** Eğitim, doğrulama ve test seti için Katarakt Sınıf Kategorisine ait VGG16, Inceptionv3 ve ResNet50 Modeli Hata ve Doğruluk değerleri

| | | Eğitim seti | | Doğrulama seti | | Test seti | |
|---|---|---|---|---|---|---|---|
| Model | Epok | Hata (Loss) | Doğruluk (Accuracy) | Hata (Loss) | Doğruluk (Accuracy) | Hata (Loss) | Doğruluk (Accuracy) |
| VGG16 | 1 | 0,367 | 0,919 | 0,177 | 0,962 | 0,250 | 0,951 |
| | 2 | 0,225 | 0,954 | 0,179 | 0,962 | 0,200 | 0,958 |
| | 3 | 0,162 | 0,963 | 0,213 | 0,968 | 0,473 | 0,966 |
| | 4 | 0,177 | 0,965 | 0,163 | 0,968 | 0,289 | 0,962 |
| | 5 | 0,154 | 0,973 | 0,149 | 0,966 | 0,384 | 0,969 |
| | 6 | 0,133 | 0,975 | 0,171 | 0,975 | 0,387 | 0,954 |
| | 7 | 0,139 | 0,972 | 0,325 | 0,977 | 0,769 | 0,962 |
| | 8 | 0,169 | 0,974 | 0,171 | 0,962 | 0,682 | 0,952 |
| | 9 | 0,129 | 0,978 | 0,108 | 0,972 | 0,515 | 0,967 |
| | 10 | 0,141 | 0,980 | 0,210 | 0,970 | 0,542 | 0,961 |
| | 11 | 0,123 | 0,979 | 0,181 | 0,955 | 0,475 | 0,948 |
| | 12 | 0,097 | 0,981 | 0,175 | 0,966 | 0,975 | 0,961 |
| | 13 | 0,086 | 0,985 | 0,605 | 0,958 | 0,903 | 0,953 |
| | 14 | 0,154 | 0,981 | 0,129 | 0,968 | 0,688 | 0,962 |
| Inceptionv3 | 1 | 0,098 | 0,977 | 0,077 | 0,979 | 0,142 | 0,968 |
| | 2 | 0,030 | 0,993 | 0,085 | 0,979 | 0,180 | 0,970 |
| | 3 | 0,019 | 0,996 | 0,134 | 0,981 | 0,162 | 0,973 |
| | 4 | 0,017 | 0,997 | 0,124 | 0,972 | 0,166 | 0,970 |
| | 5 | 0,006 | 0,999 | 0,288 | 0,962 | 0,240 | 0,961 |
| | 6 | 0,012 | 0,998 | 0,234 | 0,966 | 0,176 | 0,969 |
| | 7 | 0,004 | 0,999 | 0,161 | 0,981 | 0,239 | 0,974 |
| | 8 | 0,009 | 0,998 | 0,125 | 0,977 | 0,238 | 0,971 |
| | 9 | 0,007 | 0,998 | 0,200 | 0,975 | 0,348 | 0,972 |
| | 10 | 0,006 | 0,998 | 0,237 | 0,968 | 0,278 | 0,970 |
| ResNet50 | 1 | 0,080 | 0,971 | 0,163 | 0,979 | 0,182 | 0,970 |
| | 2 | 0,024 | 0,991 | 0,196 | 0,977 | 0,173 | 0,965 |
| | 3 | 0,017 | 0,995 | 0,115 | 0,983 | 0,170 | 0,969 |
| | 4 | 0,012 | 0,996 | 0,167 | 0,981 | 0,206 | 0,972 |
| | 5 | 0,008 | 0,997 | 0,188 | 0,975 | 0,232 | 0,969 |
| | 6 | 0,009 | 0,998 | 0,162 | 0,966 | 0,185 | 0,968 |
| | 7 | 0,008 | 0,997 | 0,217 | 0,970 | 0,192 | 0,968 |
| | 8 | 0,006 | 0,998 | 0,275 | 0,979 | 0,232 | 0,974 |
| | 9 | 0,007 | 0,998 | 0,247 | 0,981 | 0,239 | 0,967 |
| | 10 | 0,001 | 0,999 | 0,428 | 0,981 | 0,407 | 0,967 |

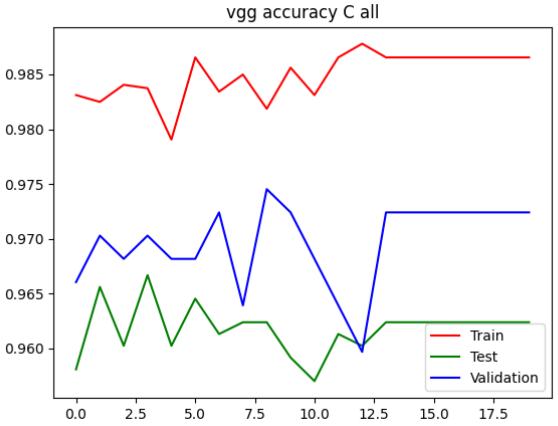



**Şekil 4.13.** Doğrulama ve test seti için VGG16, Inceptionv3 ve ResNet50 Modeli Katarakt Sınıf Kategorisine ait Karmaşıklık Matrisi

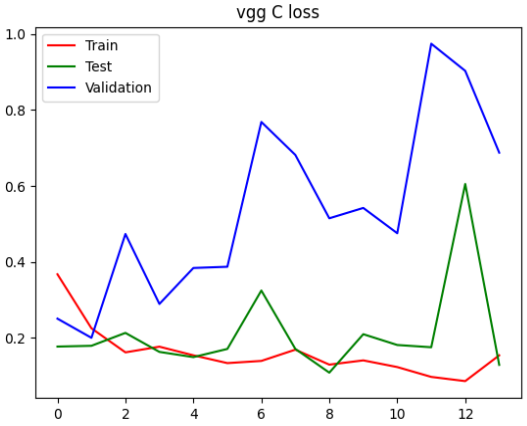
**Tablo 4.8.** Doğrulama ve test seti için Katarakt Sınıf Kategorisine ait VGG16, Inceptionv3 ve ResNet50 Modeli sınıflandırma başarı ölçütleri

| Model | Performans ölçütleri | Doğrulama seti | Test seti |
|---|---|---|---|
| VGG16 | Accuracy | 0,972 | 0,962 |
| | Precision | 0,767 | 0,841 |
| | Sens | 0,793 | 0,569 |
| | Spec | 0,984 | 0,992 |
| | $\kappa$ | 0,765 | 0,660 |
| | F1 | 0,972 | 0,962 |
| | AUC | 0,988 | 0,949 |
| | Final | 0,908 | 0,857 |
| Inceptionv3 | Accuracy | 0,983 | 0,968 |
| | Precision | 0,839 | 0,797 |
| | Sens | 0,897 | 0,723 |
| | Spec | 0,989 | 0,986 |
| | $\kappa$ | 0,858 | 0,741 |
| | F1 | 0,983 | 0,968 |
| | AUC | 0,987 | 0,948 |
| | Final | 0,943 | 0,885 |
| ResNet50 | Accuracy | 0,981 | 0,971 |
| | Precision | 0,857 | 0,797 |
| | Sens | 0,828 | 0,785 |
| | Spec | 0,991 | 0,985 |
| | $\kappa$ | 0,832 | 0,775 |
| | F1 | 0,981 | 0,971 |
| | AUC | 0,988 | 0,964 |
| | Final | 0,933 | 0,903 |

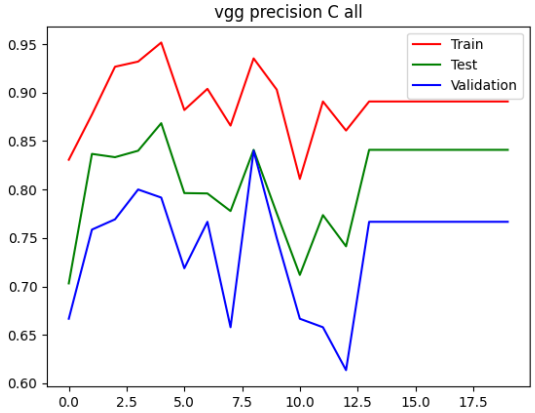
Accuracy: Doğruluk; Precision: Kesinlik; Sens: Duyarlılık; Spec: Özgüllük; $\kappa$: Kappa; AUC: Eğri altında kalan alan; Final: F1, AUC ve $\kappa$ değeri ortalaması



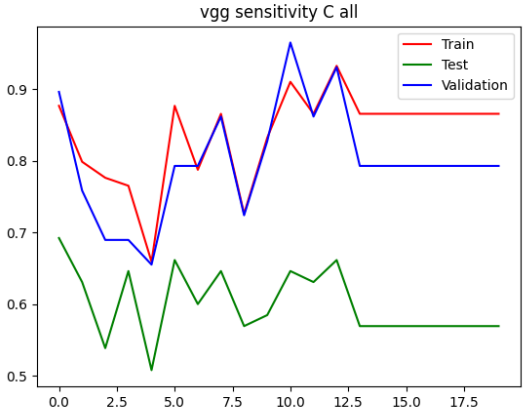

a) Doğruluk (Accuracy)




b) Hata (Loss)

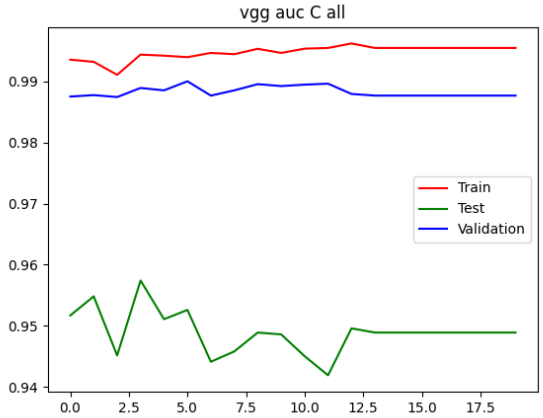



c) Kesinlik (Precision)

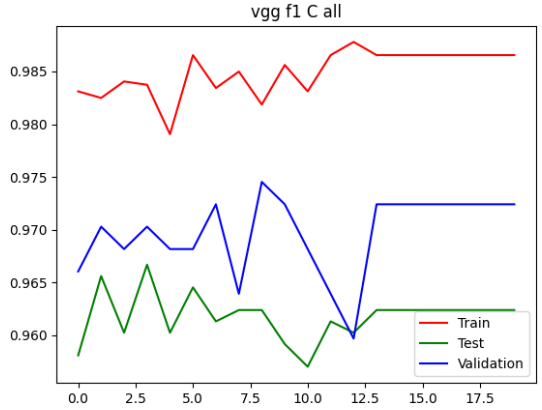



d) Duyarlılık (Recall)

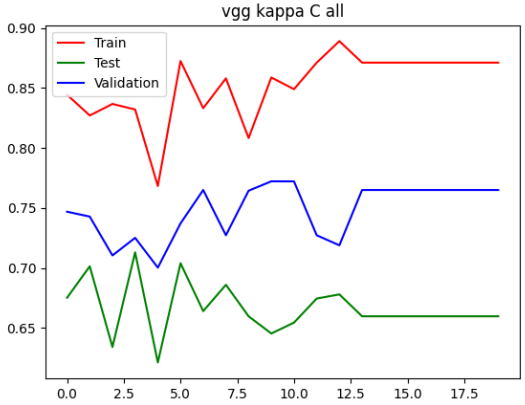
**Şekil 4.14.** Katarakt sınıf kategorisine ait VGG16 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi



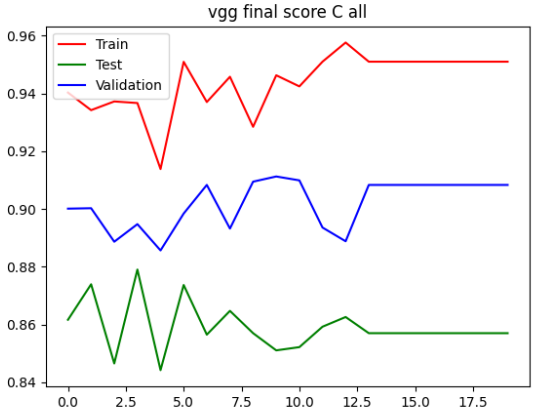

e) Eğri altında kalan alan (AUC)




f) F1 skoru

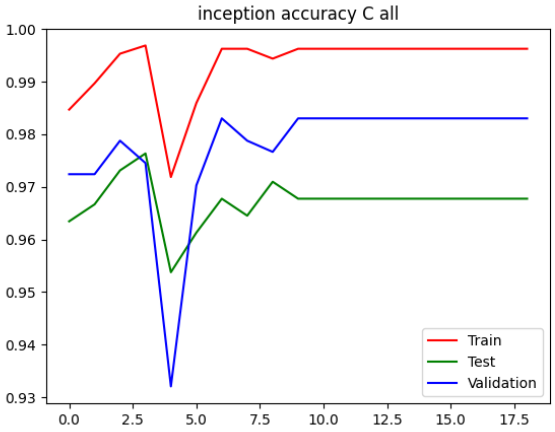



g) Kappa

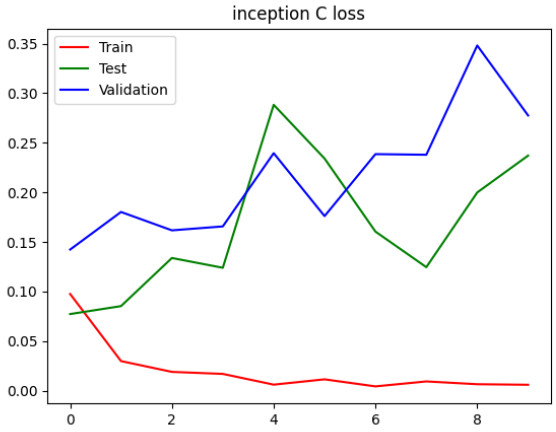



h) Final Skoru

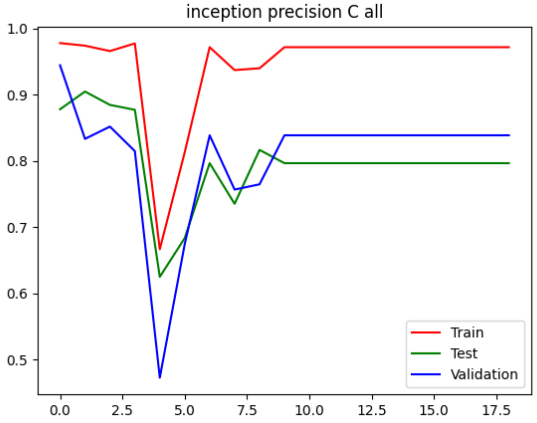
**Şekil 4.14.** Katarakt sınıf kategorisine ait VGG16 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)



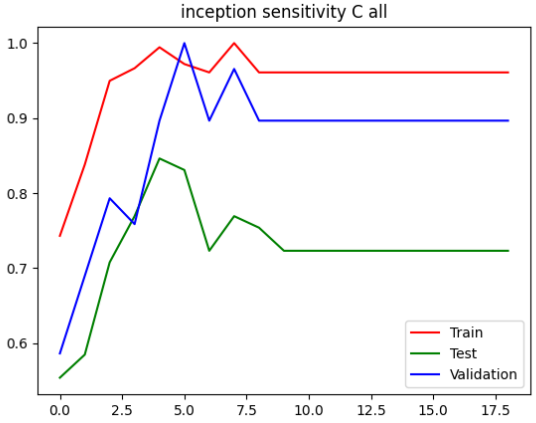

a) Doğruluk (Accuracy)




b) Hata (Loss)

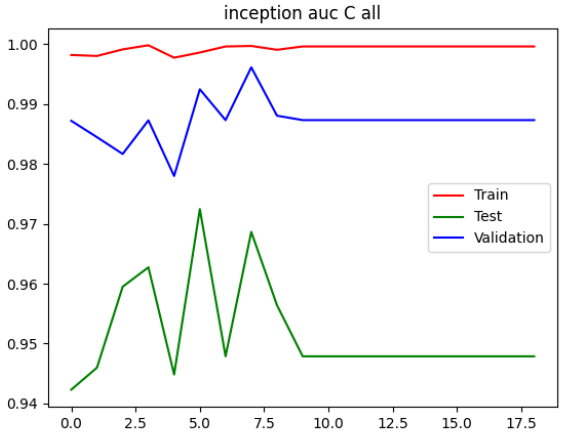



c) Kesinlik (Precision)

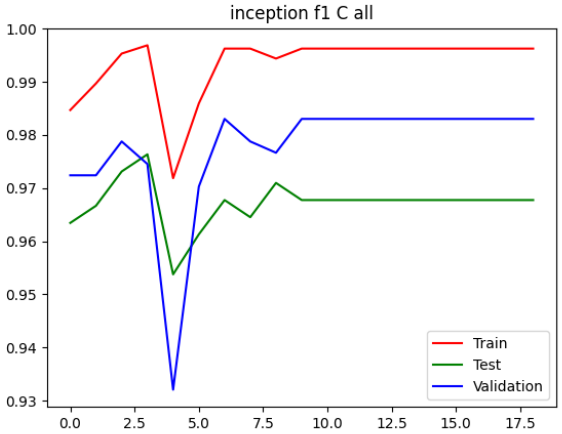



d) Duyarlılık (Recall)

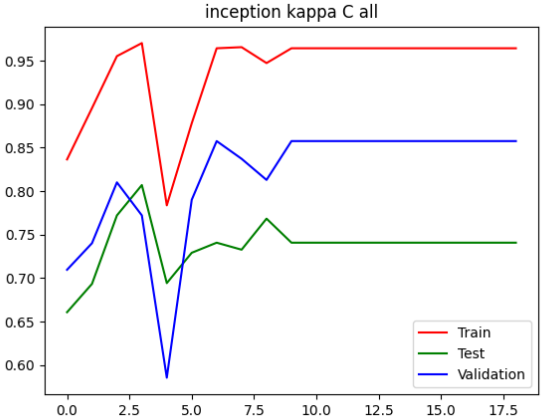
**Şekil 4.15.** Katarakt sınıf kategorisine ait Inceptionv3 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi



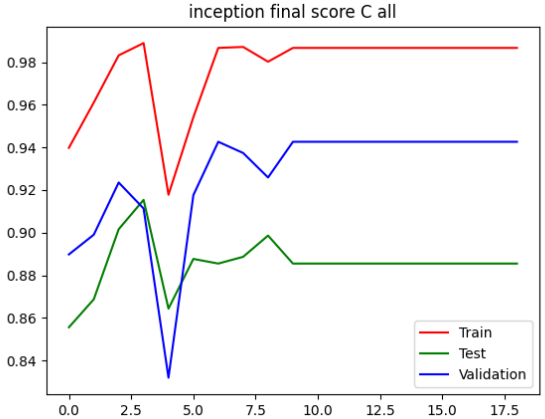

e) Eğri altında kalan alan (AUC)




f) F1 skoru

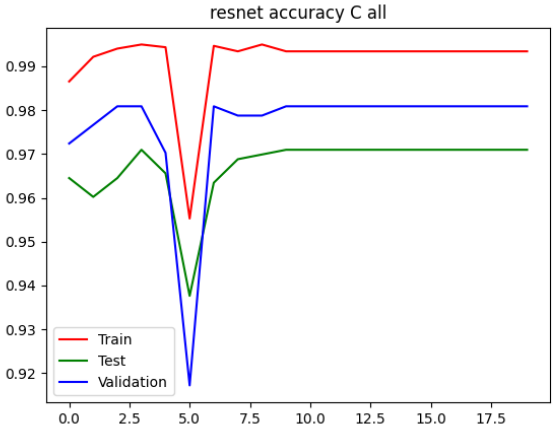



g) Kappa

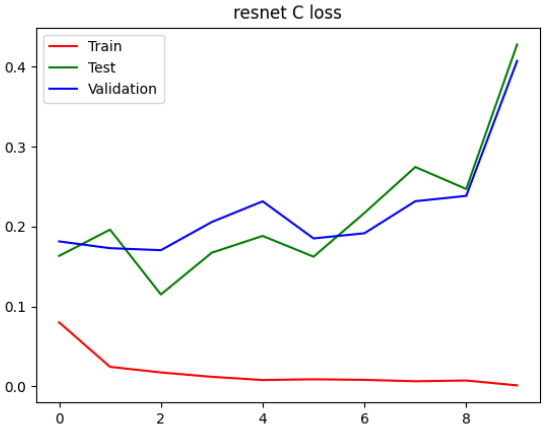



h) Final Skoru

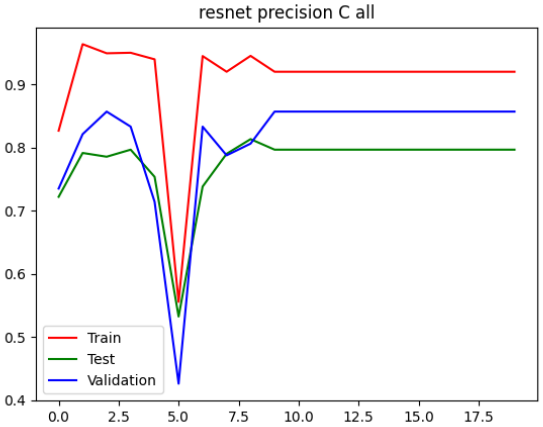
**Şekil 4.15.** Katarakt sınıf kategorisine ait Inceptionv3 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)



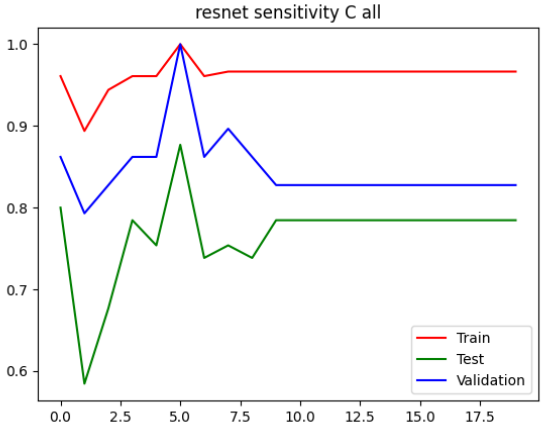

a) Doğruluk (Accuracy)




b) Hata (Loss)



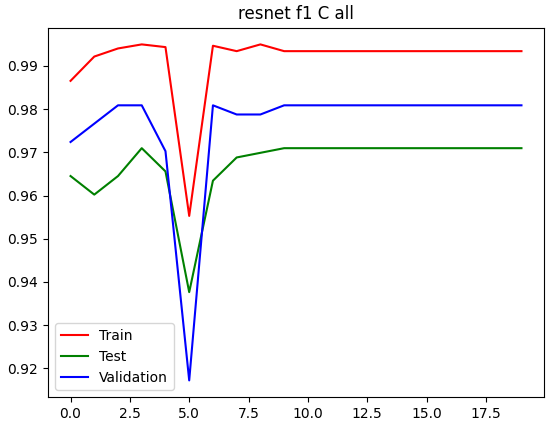

c) Kesinlik (Precision)



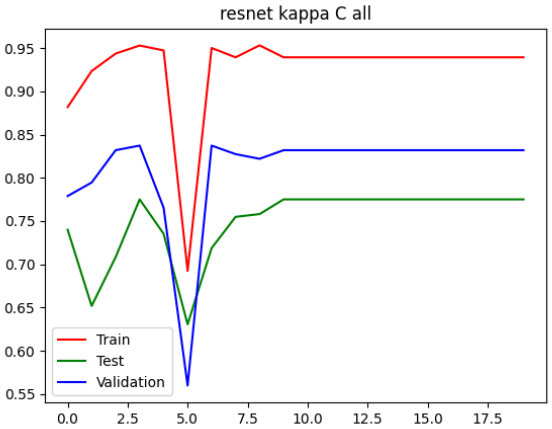

d) Duyarlılık (Recall)

**Şekil 4.16.** Katarakt sınıf kategorisine ait ResNet50 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi

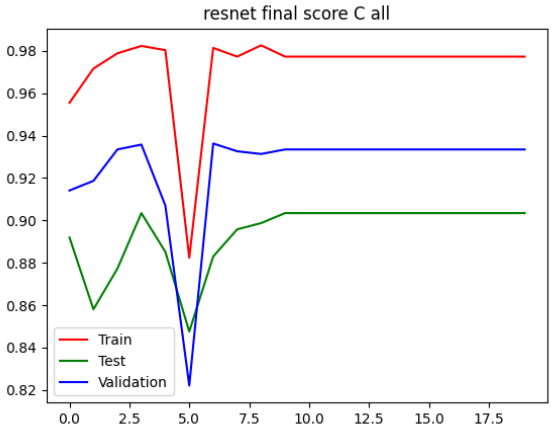

e) Eğri altında kalan alan (AUC)




f) F1 skoru




g) Kappa




h) Final Skoru

**Şekil 4.16.** Katarakt sınıf kategorisine ait ResNet50 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)

## 4.5. Yaşa bağlı Makula Dejenerasyonu Sınıf Kategorisine ait Derin Konvolüsyonel Sinir Ağı Mimari Modelleri Sınıflandırma Performans Bulguları

Eğitim, doğrulama ve test seti için VGG16, Inceptionv3, ResNet50 modellerine ait her bir epokta (epoch) hata (loss) ve doğruluk (accuracy) değerleri Tablo 4.9'daki gibi elde edilmiştir.

Eğitim, doğrulama ve test seti doğruluk değerleri sırası ile VGG16 modeli, 16 epok sonunda 0.935, 0.945 ve 0.927; Inceptionv3 modeli, 17 epok sonunda 0.999, 0.962 ve 0.944; ResNet50 modeli, 17 epok sonunda 0.998, 0.960 ve 0.942 olarak bulunmuştur. Inceptionv3 modelinin test seti için en yüksek doğruluğa sahip olduğu görülmektedir.

Eğitim, doğrulama ve test seti hata değerleri sırası ile VGG16 modeli, 16 epok sonunda 0.225, 0.548 ve 0.771; Inceptionv3 modeli, 17 epok sonunda 0.004, 0.597 ve 0.919; ResNet50 modeli, 0.006, 0.448 ve 0.654 olarak bulunmuştur. ResNet50 modelinin test seti için en düşük hata değerine sahip olduğu görülmektedir.

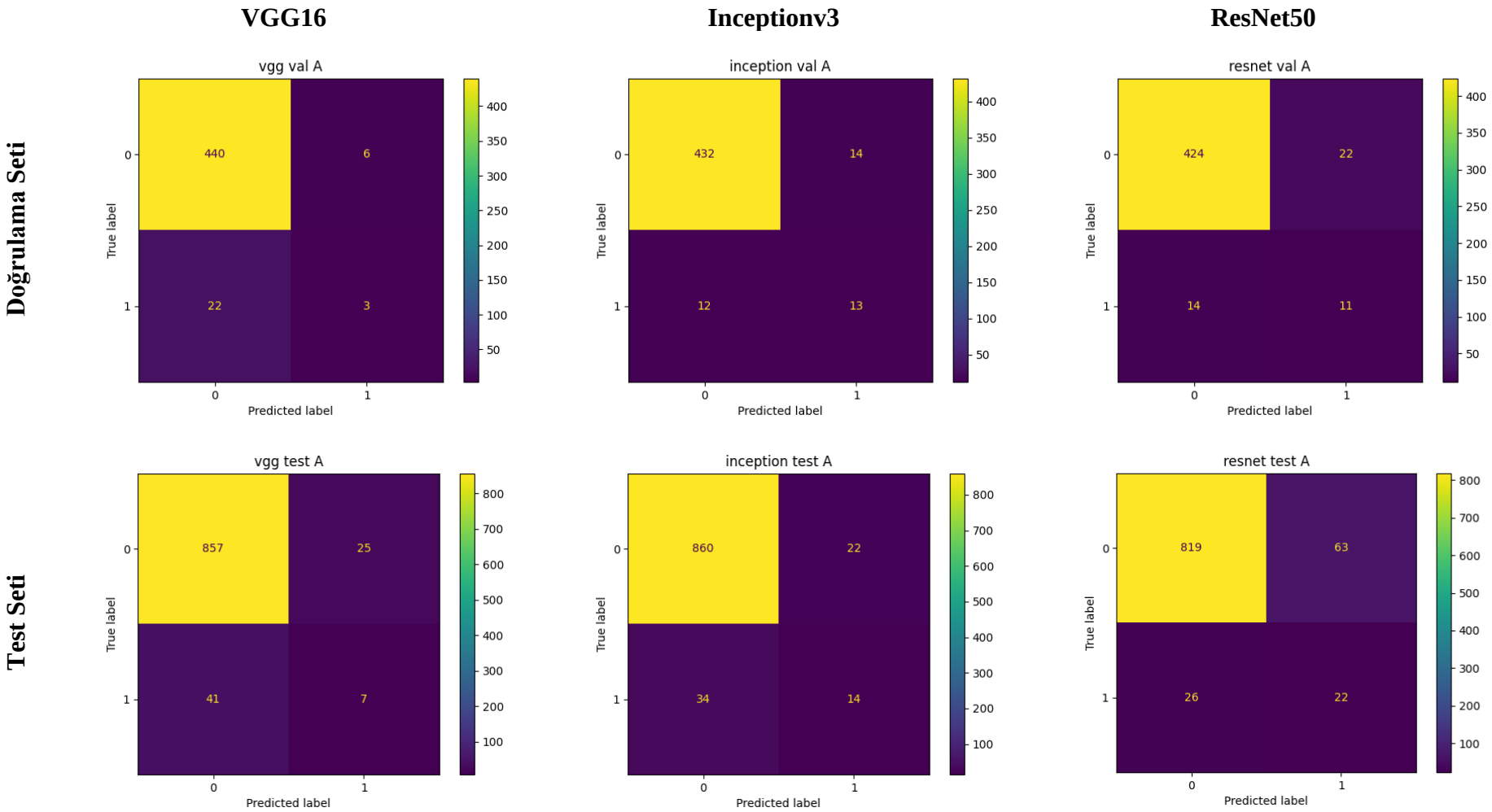
Doğrulama ve test seti için yaşa bağlı makula dejenerasyonu sınıf kategorisine ait karmaşıklık matrisi Şekil 4.17'de yer almaktadır. Doğrulama setinde, toplam 471 fundus görüntüsünden 25'inin yaşa bağlı makula dejenerasyonu sınıfına ait olduğu bilinmektedir. VGG16 modeli, 3; Inceptionv3 modeli, 13; ResNet50 modeli, 11'ini doğru olarak sınıflandırmıştır. Yaşa bağlı makula dejenerasyonu olmayan sınıf kategorisine sahip 446 fundus görüntüsü bulunmaktadır. VGG16 modeli, 440; Inceptionv3 modeli, 432; ResNet50 modeli, 424'ünü doğru olarak sınıflandırmıştır. Test setinde toplam 930 fundus görüntüsünden 48'inin yaşa bağlı makula dejenerasyonu sınıfına ait olduğu bilinmektedir. VGG16 modeli, 7; Inceptionv3 modeli, 14; ResNet50 modeli, 22'sini doğru olarak sınıflandırmıştır. Yaşa bağlı makula dejenerasyonu olmayan sınıf kategorisine sahip 882 fundus görüntüsünden, VGG16 modeli, 857; Inceptionv3 modeli, 860; ResNet50 modeli, 819'unu doğru olarak sınıflandırmıştır.

Karmaşıklık matrisi elde edildikten sonra her bir modelin sınıflandırma başarı ölçütleri hesaplanarak Tablo 4.10'da sunulmuştur. VGG16, Inceptionv3, ResNet50 modelleri için her bir epokta doğruluk, hata, kesinlik, duyarlılık, Eğri altında kalan alan (AUC), F1 skoru, Kappa ve Final değeri sonuçları grafiksel olarak Şekil 4.18, Şekil 4.19, Şekil 4.20'de sunulmuştur.

Yaşa bağlı makula dejenerasyonu sınıfı test setinde VGG16, Inceptionv3, ResNet50 modelleri için Final skor değerleri sırası ile 0.549, 0.663 ve 0.630 olarak hesaplanmıştır. En yüksek Final skoru Inceptionv3 modeli ile elde edilmiştir.

**Tablo 4.9.** Eğitim doğrulama ve test seti için Yaşa bağlı makula dejenerasyonu Sınıf Kategorisine ait VGG16, Inceptionv3 ve ResNet50 Modeli Hata ve Doğruluk değerleri

| | | Eğitim seti | | Doğrulama seti | | Test seti | |
|---|---|---|---|---|---|---|---|
| Model | Epok | Hata (Loss) | Doğruluk (Accuracy) | Hata (Loss) | Doğruluk (Accuracy) | Hata (Loss) | Doğruluk (Accuracy) |
| VGG16 | 1 | 0,603 | 0,738 | 0,356 | 0,866 | 0,366 | 0,859 |
| | 2 | 0,351 | 0,861 | 0,252 | 0,943 | 0,286 | 0,929 |
| | 3 | 0,306 | 0,878 | 0,399 | 0,909 | 0,494 | 0,897 |
| | 4 | 0,269 | 0,902 | 0,329 | 0,917 | 0,466 | 0,909 |
| | 5 | 0,264 | 0,900 | 0,346 | 0,896 | 0,393 | 0,882 |
| | 6 | 0,305 | 0,898 | 0,495 | 0,943 | 0,594 | 0,932 |
| | 7 | 0,281 | 0,901 | 0,324 | 0,898 | 0,404 | 0,890 |
| | 8 | 0,237 | 0,915 | 0,331 | 0,941 | 0,523 | 0,915 |
| | 9 | 0,227 | 0,922 | 0,411 | 0,947 | 0,573 | 0,928 |
| | 10 | 0,224 | 0,925 | 0,465 | 0,934 | 0,609 | 0,922 |
| | 11 | 0,217 | 0,925 | 0,434 | 0,930 | 0,489 | 0,926 |
| | 12 | 0,213 | 0,932 | 0,339 | 0,924 | 0,529 | 0,908 |
| | 13 | 0,233 | 0,927 | 0,435 | 0,938 | 0,578 | 0,926 |
| | 14 | 0,188 | 0,933 | 0,44 | 0,915 | 0,659 | 0,908 |
| | 15 | 0,186 | 0,939 | 0,403 | 0,924 | 0,774 | 0,906 |
| | 16 | 0,225 | 0,935 | 0,548 | 0,945 | 0,771 | 0,927 |
| Inceptionv3 | 1 | 0,224 | 0,932 | 0,282 | 0,945 | 0,453 | 0,930 |
| | 2 | 0,070 | 0,982 | 0,282 | 0,953 | 0,480 | 0,932 |
| | 3 | 0,045 | 0,989 | 0,404 | 0,945 | 0,507 | 0,940 |
| | 4 | 0,034 | 0,992 | 0,470 | 0,953 | 0,618 | 0,942 |
| | 5 | 0,027 | 0,993 | 0,483 | 0,951 | 0,799 | 0,931 |
| | 6 | 0,023 | 0,995 | 0,425 | 0,958 | 0,647 | 0,941 |
| | 7 | 0,016 | 0,996 | 0,402 | 0,947 | 0,676 | 0,938 |
| | 8 | 0,026 | 0,994 | 0,410 | 0,960 | 0,668 | 0,941 |
| | 9 | 0,008 | 0,998 | 0,450 | 0,951 | 0,796 | 0,933 |
| | 10 | 0,019 | 0,996 | 0,453 | 0,962 | 0,762 | 0,938 |
| | 11 | 0,014 | 0,997 | 0,553 | 0,958 | 0,725 | 0,945 |
| | 12 | 0,007 | 0,999 | 0,544 | 0,953 | 0,836 | 0,940 |
| | 13 | 0,015 | 0,997 | 0,477 | 0,953 | 0,762 | 0,939 |
| | 14 | 0,010 | 0,999 | 0,530 | 0,951 | 0,780 | 0,937 |
| | 15 | 0,012 | 0,998 | 0,497 | 0,953 | 0,786 | 0,939 |
| | 16 | 0,006 | 0,999 | 0,454 | 0,960 | 0,725 | 0,939 |
| | 17 | 0,004 | 0,999 | 0,597 | 0,962 | 0,919 | 0,944 |
| ResNet50 | 1 | 0,180 | 0,924 | 0,185 | 0,951 | 0,309 | 0,935 |
| | 2 | 0,054 | 0,979 | 0,307 | 0,951 | 0,415 | 0,940 |
| | 3 | 0,037 | 0,986 | 0,348 | 0,958 | 0,491 | 0,938 |
| | 4 | 0,032 | 0,989 | 0,307 | 0,955 | 0,461 | 0,949 |
| | 5 | 0,019 | 0,993 | 0,346 | 0,945 | 0,674 | 0,915 |
| | 6 | 0,019 | 0,993 | 0,356 | 0,962 | 0,485 | 0,947 |
| | 7 | 0,016 | 0,995 | 0,332 | 0,953 | 0,521 | 0,940 |
| | 8 | 0,014 | 0,995 | 0,333 | 0,962 | 0,544 | 0,945 |
| | 9 | 0,012 | 0,996 | 0,324 | 0,949 | 0,478 | 0,942 |
| | 10 | 0,009 | 0,997 | 0,368 | 0,964 | 0,670 | 0,931 |
| | 11 | 0,009 | 0,997 | 0,294 | 0,964 | 0,590 | 0,943 |
| | 12 | 0,011 | 0,996 | 0,382 | 0,949 | 0,786 | 0,928 |
| | 13 | 0,011 | 0,997 | 0,350 | 0,960 | 0,706 | 0,938 |
| | 14 | 0,004 | 0,999 | 0,423 | 0,964 | 0,782 | 0,941 |
| | 15 | 0,008 | 0,998 | 0,452 | 0,964 | 0,695 | 0,939 |
| | 16 | 0,008 | 0,998 | 0,435 | 0,958 | 0,605 | 0,941 |
| | 17 | 0,006 | 0,998 | 0,448 | 0,960 | 0,654 | 0,942 |

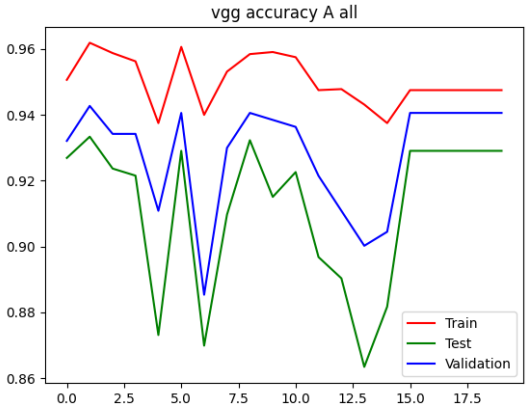



**Şekil 4.17.** Doğrulama ve test seti için VGG16, Inceptionv3 ve ResNet50 Modeli Yaşa bağlı makula dejenerasyonu Sınıf Kategorisine ait Karmaşıklık Matris

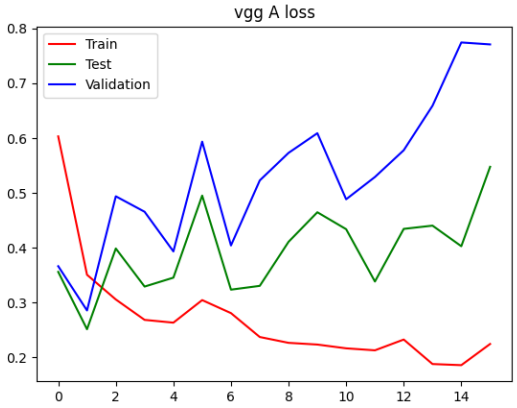
**Tablo 4.10.** Doğrulama ve test seti için Yaşa bağlı makula dejenerasyonu sınıf kategorisine ait VGG16, Inceptionv3 ve ResNet50 Modeli sınıflandırma başarı ölçütleri

| Model | Performans ölçütleri | Doğrulama seti | Test seti |
|---|---|---|---|
| VGG16 | Accuracy | 0,941 | 0,929 |
| | Precision | 0,333 | 0,219 |
| | Sens | 0,12 | 0,146 |
| | Spec | 0,987 | 0,972 |
| | $\kappa$ | 0,153 | 0,139 |
| | F1 | 0,941 | 0,929 |
| | AUC | 0,72 | 0,578 |
| | Final | 0,605 | 0,549 |
| Inceptionv3 | Accuracy | 0,945 | 0,940 |
| | Precision | 0,481 | 0,389 |
| | Sens | 0,52 | 0,292 |
| | Spec | 0,969 | 0,975 |
| | $\kappa$ | 0,471 | 0,302 |
| | F1 | 0,945 | 0,940 |
| | AUC | 0,796 | 0,748 |
| | Final | 0,737 | 0,663 |
| ResNet50 | Accuracy | 0,924 | 0,904 |
| | Precision | 0,333 | 0,259 |
| | Sens | 0,44 | 0,458 |
| | Spec | 0,951 | 0,929 |
| | $\kappa$ | 0,339 | 0,284 |
| | F1 | 0,924 | 0,904 |
| | AUC | 0,815 | 0,702 |
| | Final | 0,693 | 0,630 |

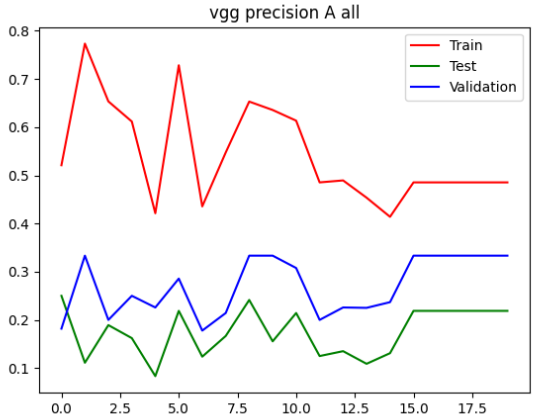
Accuracy: Doğruluk; Precision: Kesinlik; Sens: Duyarlılık; Spec: Özgüllük; $\kappa$: Kappa; AUC: Eğri altında kalan alan; Final: F1, AUC ve $\kappa$ değeri ortalaması



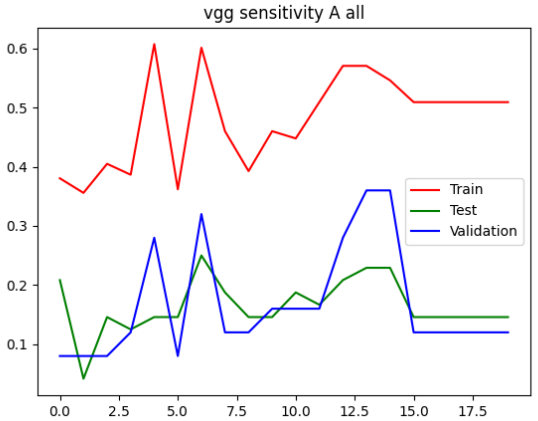

a) Doğruluk (Accuracy)




b) Hata (Loss)

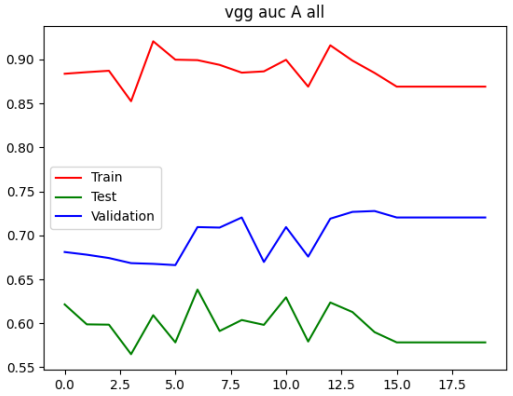



c) Kesinlik (Precision)

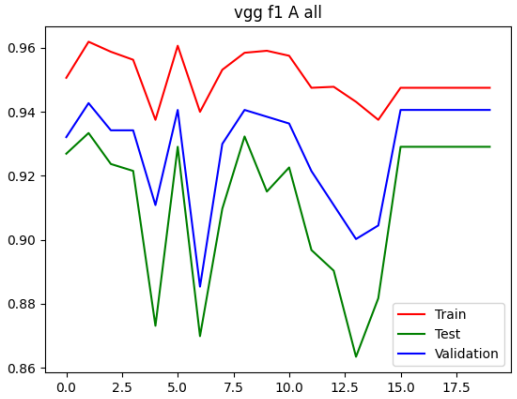



d) Duyarlılık (Recall)

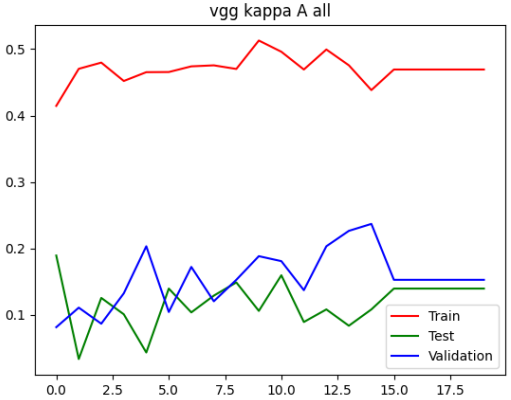
**Şekil 4.18.** Yaşa bağlı makula dejenerasyonu sınıf kategorisine ait VGG16 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi



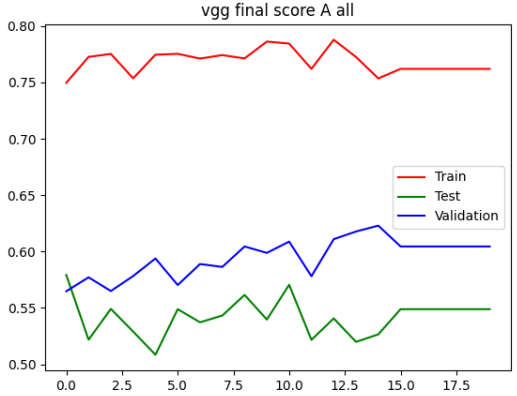

e) Eğri altında kalan alan (AUC)




f) F1 skoru

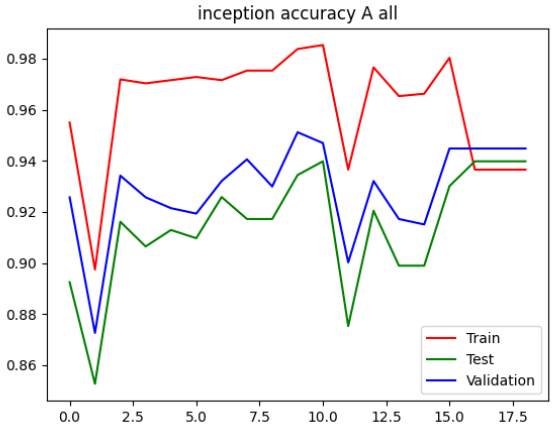



g) Kappa

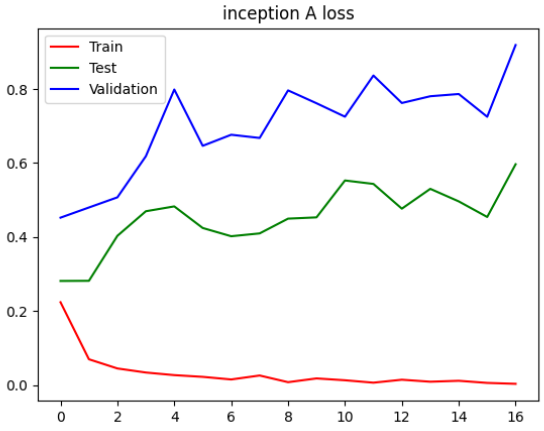



h) Final Skoru

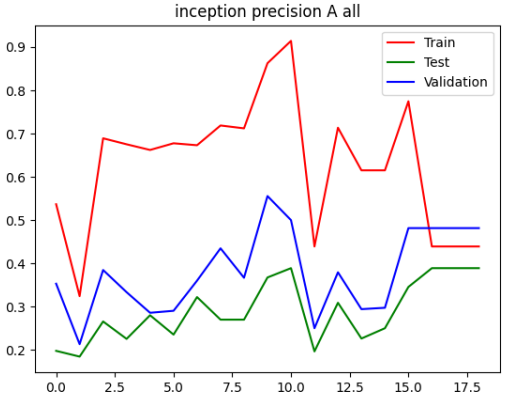
**Şekil 4.18.** Yaşa bağlı makula dejenerasyonu sınıf kategorisine ait VGG16 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)



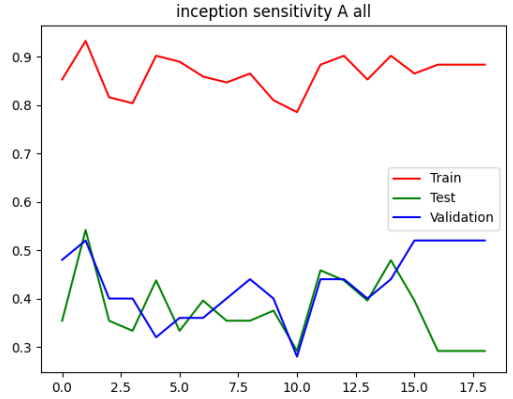

a) Doğruluk (Accuracy)




b) Hata (Loss)

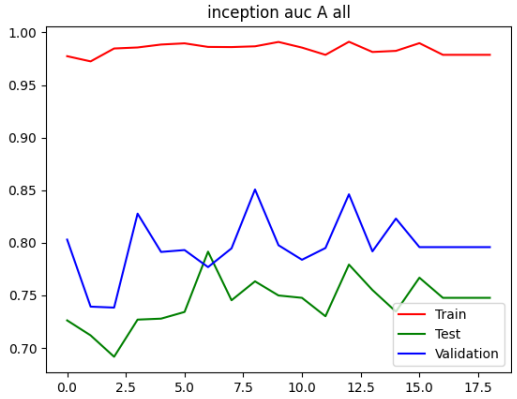



c) Kesinlik (Precision)

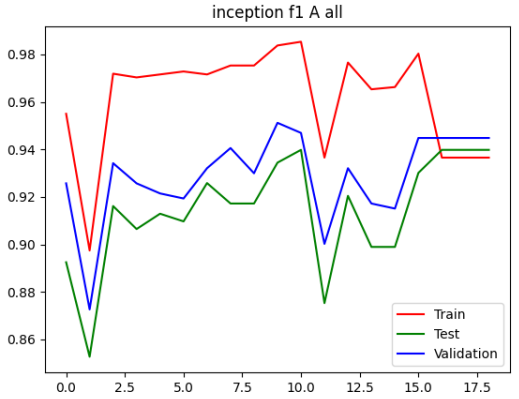



d) Duyarlılık (Recall)

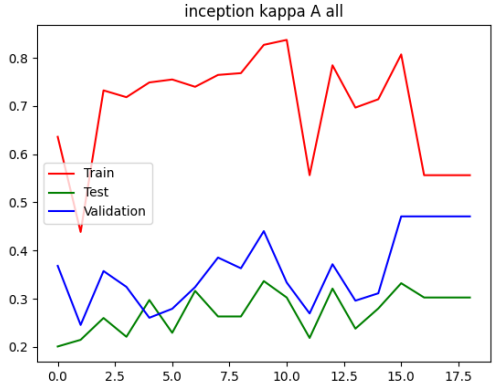
**Şekil 4.19.** Yaşa bağlı makula dejenerasyonu sınıf kategorisine ait Inceptionv3 modeli sınıflandırma başarı ölçütleri değerlendirilmesi



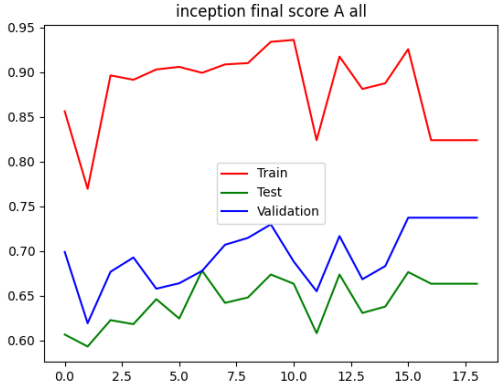

e) Eğri altında kalan alan (AUC)




f) F1 skoru

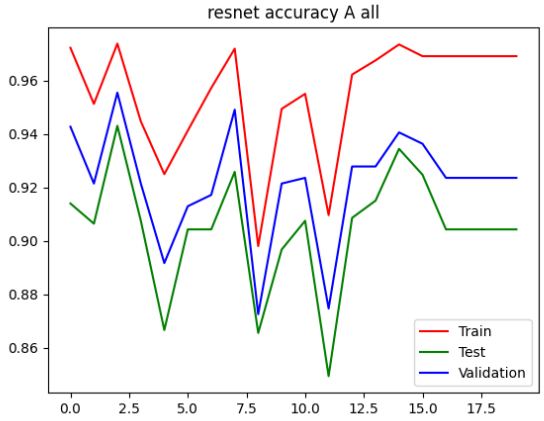



g) Kappa



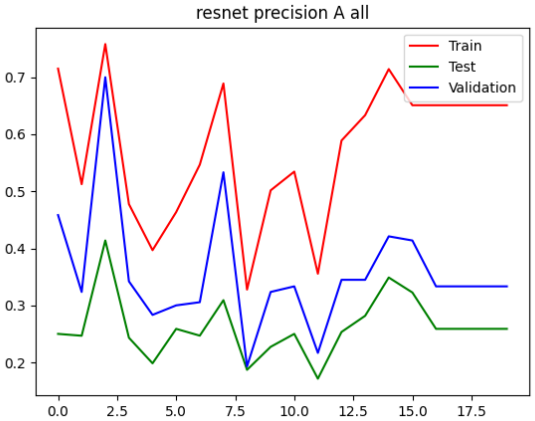

h) Final Skoru

**Şekil 4.19.** Yaşa bağlı makula dejenerasyonu sınıf kategorisine ait Inceptionv3 modeli sınıflandırma başarı ölçütleri değerlendirilmesi (Devam Ediyor)

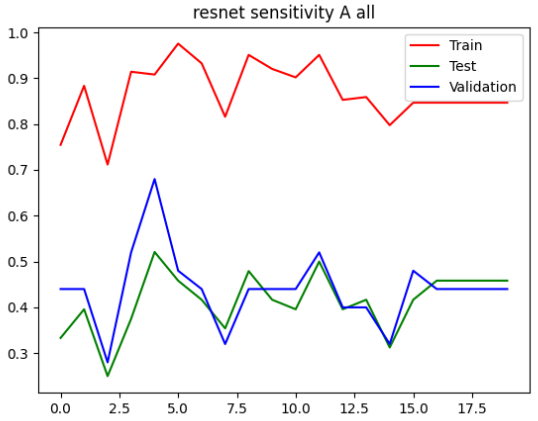



a) Doğruluk (Accuracy)


b) Hata (Loss)

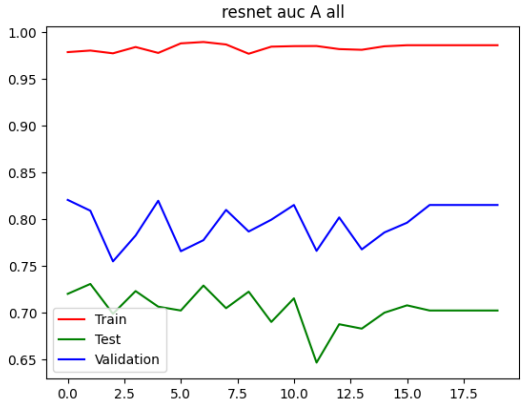



c) Kesinlik (Precision)

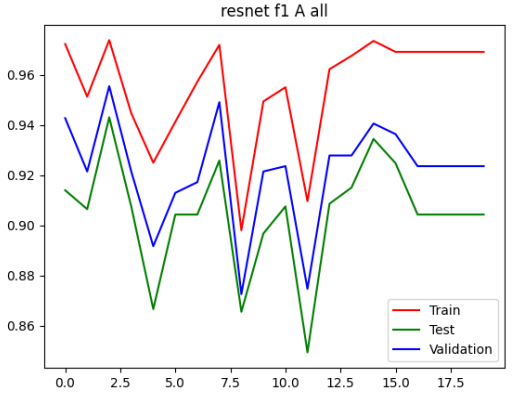



d) Duyarlılık (Recall)

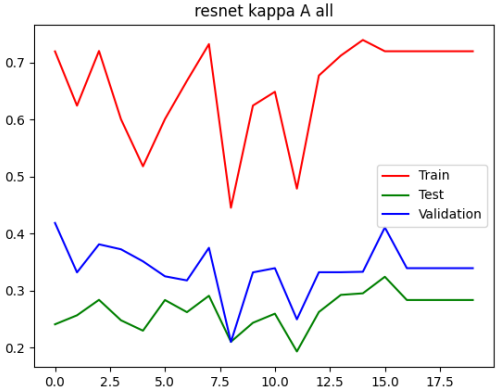
**Şekil 4.20.** Yaşa bağlı makula dejenerasyonu sınıf kategorisine ait ResNet50 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi




e) Eğri altında kalan alan (AUC)




f) F1 skoru




g) Kappa


h) Final Skoru

**Şekil 4.20.** Yaşa bağlı makula dejenerasyonu sınıf kategorisine ait ResNet50 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)

## 4.6. Hipertansiyon Sınıf Kategorisine ait Derin Konvolüsyonel Sinir Ağı Mimari Modelleri Sınıflandırma Performans Bulguları

Eğitim, doğrulama ve test seti için VGG16, Inceptionv3, ResNet50 modellerine ait her bir epokta (epoch) hata (loss) ve doğruluk (accuracy) değerleri Tablo 4.11'deki gibi elde edilmiştir. Eğitim, doğrulama ve test seti doğruluk değerleri sırası ile VGG16 modeli, 15 epok sonunda 0.946, 0.955 ve 0.966; Inceptionv3 modeli, 13 epok sonunda 0.996, 0.955 ve 0.957; ResNet50 modeli, 12 epok sonunda 0.998, 0.960 ve 0.966 olarak bulunmuştur. VGG16 ve ResNet50 modelinin test seti için en yüksek doğruluğa sahip olduğu görülmektedir.

Eğitim, doğrulama ve test seti hata değerleri sırası ile VGG16 modeli, 15 epok sonunda 0.203, 0.461 ve 0.511; Inceptionv3 modeli, 13 epok sonunda 0.020, 0.798 ve 0.756; ResNet50 modeli, 0.007, 0.859 ve 0.763 olarak bulunmuştur. VGG16 modelinin test seti için en düşük hata değerine sahip olduğu görülmektedir.

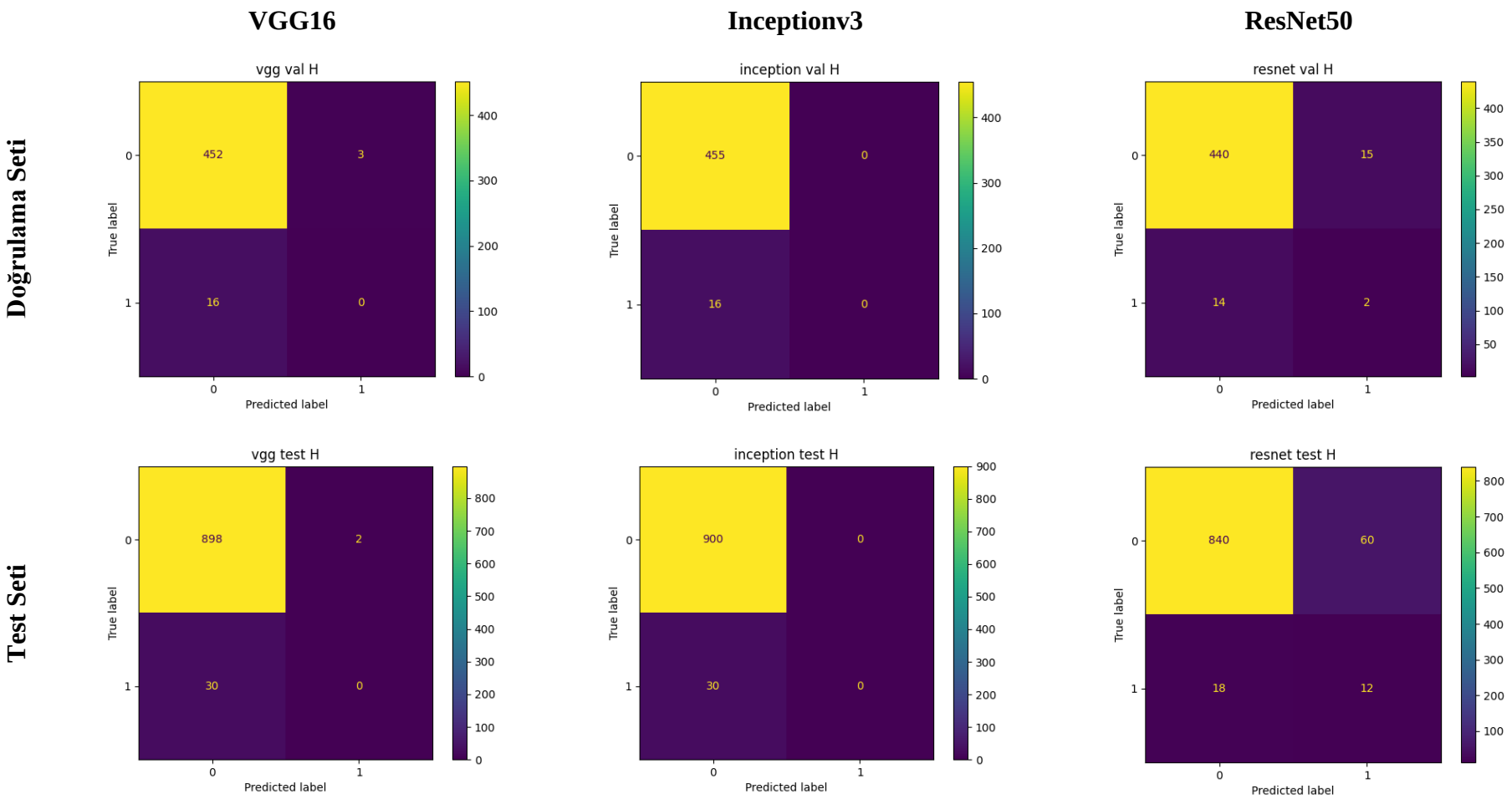
Doğrulama ve test seti için hipertansiyon sınıf kategorisine ait karmaşıklık matrisi Şekil 4.21'de yer almaktadır. Doğrulama setinde, toplam 471 fundus görüntüsünden 16'sının hipertansiyon sınıfına ait olduğu bilinmektedir. VGG16 modeli, 0; Inceptionv3 modeli, 0; ResNet50 modeli, 2'sini doğru olarak sınıflandırmıştır. Hipertansiyon olmayan sınıf kategorisine sahip 455 fundus görüntüsü bulunmaktadır. VGG16 modeli, 452; Inceptionv3 modeli, 455; ResNet50 modeli, 440'ını doğru olarak sınıflandırmıştır. Test setinde toplam 930 fundus görüntüsünden 30'unun hipertansiyon sınıfına ait olduğu bilinmektedir. VGG16 modeli, 0; Inceptionv3 modeli, 0; ResNet50 modeli, 12'sini doğru olarak sınıflandırmıştır. Hipertansiyon olmayan sınıf kategorisine sahip 900 fundus görüntüsünden, VGG16 modeli, 898; Inceptionv3 modeli, 900; ResNet50 modeli, 840'ını doğru olarak sınıflandırmıştır.

Karmaşıklık matrisi elde edildikten sonra her bir modelin sınıflandırma başarı ölçütleri hesaplanarak Tablo 4.12'de sunulmuştur. VGG16, Inceptionv3, ResNet50 modelleri için her bir epokta doğruluk, hata, kesinlik, duyarlılık, Eğri altında kalan alan (AUC), F1 skoru, Kappa ve Final değeri sonuçları grafiksel olarak Şekil 4.22, Şekil 4.23, Şekil 4.24'de sunulmuştur.

Hipertansiyon sınıfı test setinde VGG16, Inceptionv3, ResNet50 modelleri için Final skor değerleri sırası ile 0.538, 0.558 ve 0.626 olarak hesaplanmıştır. En yüksek Final skoru ResNet50 modeli ile elde edilmiştir.

**Tablo 4.11.** Eğitim, doğrulama ve test seti için Hipertansiyon Sınıf Kategorisine ait VGG16, Inceptionv3 ve ResNet50 Modeli Hata ve Doğruluk değerleri

| | | Eğitim seti | | Doğrulama seti | | Test seti | |
|---|---|---|---|---|---|---|---|
| Model | Epok | Hata (Loss) | Doğruluk (Accuracy) | Hata (Loss) | Doğruluk (Accuracy) | Hata (Loss) | Doğruluk (Accuracy) |
| VGG16 | 1 | 0,551 | 0,774 | 0,197 | 0,941 | 0,207 | 0,956 |
| | 2 | 0,307 | 0,888 | 0,260 | 0,953 | 0,255 | 0,958 |
| | 3 | 0,276 | 0,898 | 0,311 | 0,941 | 0,326 | 0,955 |
| | 4 | 0,239 | 0,920 | 0,310 | 0,943 | 0,276 | 0,951 |
| | 5 | 0,244 | 0,918 | 0,337 | 0,938 | 0,283 | 0,954 |
| | 6 | 0,204 | 0,935 | 0,431 | 0,949 | 0,414 | 0,961 |
| | 7 | 0,210 | 0,941 | 0,566 | 0,955 | 0,511 | 0,962 |
| | 8 | 0,192 | 0,941 | 0,724 | 0,962 | 0,673 | 0,967 |
| | 9 | 0,208 | 0,94 | 0,385 | 0,958 | 0,376 | 0,963 |
| | 10 | 0,184 | 0,942 | 0,420 | 0,947 | 0,429 | 0,963 |
| | 11 | 0,186 | 0,949 | 0,623 | 0,962 | 0,637 | 0,967 |
| | 12 | 0,161 | 0,956 | 0,731 | 0,958 | 0,727 | 0,961 |
| | 13 | 0,177 | 0,953 | 0,347 | 0,958 | 0,372 | 0,960 |
| | 14 | 0,133 | 0,965 | 0,645 | 0,958 | 0,675 | 0,965 |
| | 15 | 0,203 | 0,946 | 0,461 | 0,955 | 0,511 | 0,966 |
| Inceptionv3 | 1 | 0,228 | 0,930 | 0,367 | 0,964 | 0,350 | 0,968 |
| | 2 | 0,072 | 0,981 | 0,421 | 0,964 | 0,330 | 0,968 |
| | 3 | 0,042 | 0,989 | 0,454 | 0,962 | 0,377 | 0,962 |
| | 4 | 0,036 | 0,993 | 0,572 | 0,962 | 0,518 | 0,966 |
| | 5 | 0,026 | 0,994 | 0,831 | 0,962 | 0,660 | 0,968 |
| | 6 | 0,021 | 0,996 | 0,856 | 0,966 | 0,668 | 0,968 |
| | 7 | 0,020 | 0,995 | 0,761 | 0,966 | 0,714 | 0,969 |
| | 8 | 0,012 | 0,998 | 0,732 | 0,960 | 0,639 | 0,965 |
| | 9 | 0,018 | 0,996 | 0,602 | 0,960 | 0,495 | 0,967 |
| | 10 | 0,013 | 0,997 | 0,569 | 0,955 | 0,522 | 0,963 |
| | 11 | 0,010 | 0,998 | 0,661 | 0,962 | 0,606 | 0,968 |
| | 12 | 0,008 | 0,999 | 0,748 | 0,962 | 0,677 | 0,967 |
| | 13 | 0,020 | 0,996 | 0,798 | 0,955 | 0,756 | 0,957 |
| ResNet50 | 1 | 0,175 | 0,928 | 0,314 | 0,958 | 0,244 | 0,963 |
| | 2 | 0,060 | 0,978 | 0,270 | 0,951 | 0,254 | 0,955 |
| | 3 | 0,033 | 0,988 | 0,493 | 0,960 | 0,418 | 0,965 |
| | 4 | 0,032 | 0,988 | 0,473 | 0,962 | 0,392 | 0,968 |
| | 5 | 0,025 | 0,992 | 0,438 | 0,966 | 0,494 | 0,967 |
| | 6 | 0,015 | 0,995 | 0,514 | 0,962 | 0,510 | 0,968 |
| | 7 | 0,019 | 0,993 | 0,439 | 0,960 | 0,431 | 0,966 |
| | 8 | 0,016 | 0,995 | 0,642 | 0,960 | 0,605 | 0,967 |
| | 9 | 0,010 | 0,996 | 0,691 | 0,964 | 0,787 | 0,968 |
| | 10 | 0,010 | 0,997 | 0,443 | 0,958 | 0,543 | 0,962 |
| | 11 | 0,013 | 0,995 | 0,504 | 0,951 | 0,553 | 0,959 |
| | 12 | 0,007 | 0,998 | 0,859 | 0,960 | 0,763 | 0,966 |

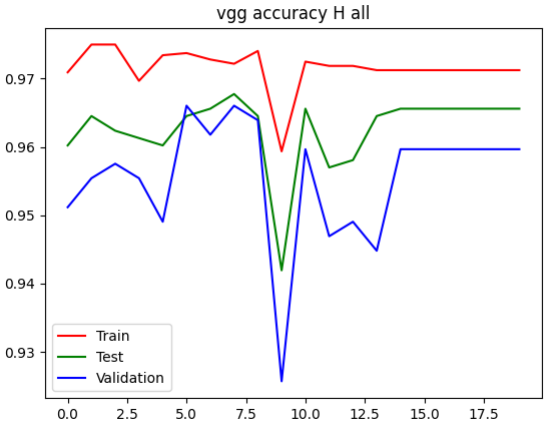



**Şekil 4.21.** Doğrulama ve test seti için VGG16, Inceptionv3 ve ResNet50 Modeli Hipertansiyon Sınıf Kategorisine ait Karmaşıklık Matrisi

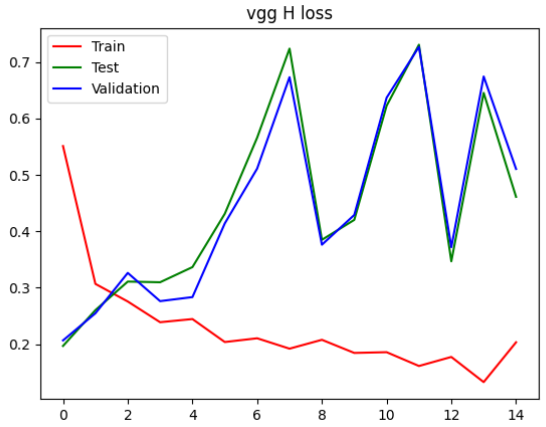
**Tablo 4.12.** Doğrulama ve test seti için Hipertansiyon Sınıf Kategorisine ait VGG16, Inceptionv3 ve ResNet50 Modeli sınıflandırma başarı ölçütleri

| Model | Performans ölçütleri | Doğrulama seti | Test seti |
|---|---|---|---|
| VGG16 | Accuracy | 0,960 | 0,966 |
| | Precision | 0,000 | 0,000 |
| | Sens | 0,000 | 0,000 |
| | Spec | 0,993 | 0,998 |
| | $\kappa$ | -0,011 | -0,004 |
| | F1 | 0,960 | 0,966 |
| | AUC | 0,604 | 0,654 |
| | Final | 0,518 | 0,538 |
| Inceptionv3 | Accuracy | 0,966 | 0,968 |
| | Precision | 0,000 | 0,000 |
| | Sens | 0,000 | 0,000 |
| | Spec | 1,000 | 1,000 |
| | $\kappa$ | 0,000 | 0,000 |
| | F1 | 0,966 | 0,968 |
| | AUC | 0,690 | 0,706 |
| | Final | 0,552 | 0,558 |
| ResNet50 | Accuracy | 0,938 | 0,916 |
| | Precision | 0,118 | 0,167 |
| | Sens | 0,125 | 0,400 |
| | Spec | 0,967 | 0,933 |
| | $\kappa$ | 0,089 | 0,199 |
| | F1 | 0,938 | 0,916 |
| | AUC | 0,745 | 0,764 |
| | Final | 0,591 | 0,626 |

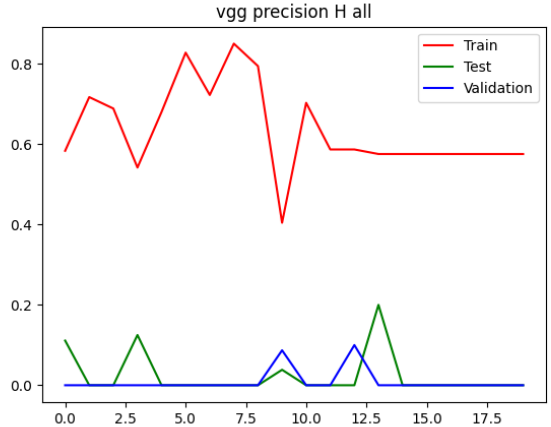
Accuracy: Doğruluk; Precision: Kesinlik; Sens: Duyarlılık; Spec: Özgüllük; $\kappa$: Kappa; AUC: Eğri altında kalan alan; Final: F1, AUC ve $\kappa$ değeri ortalaması



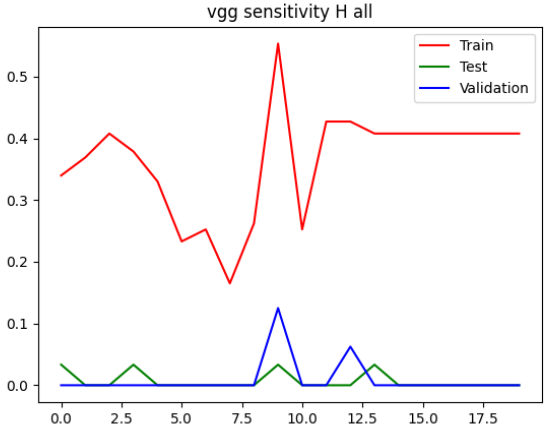

a) Doğruluk (Accuracy)




b) Hata (Loss)

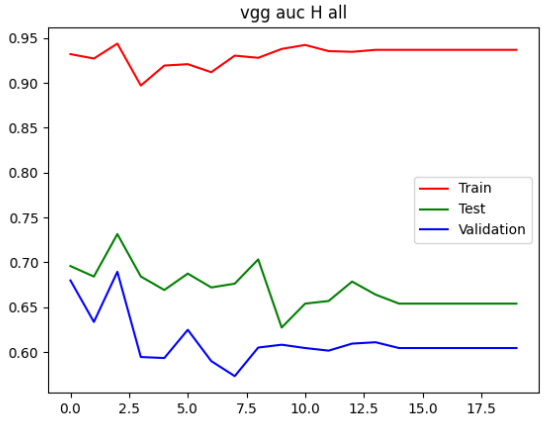



c) Kesinlik (Precision)

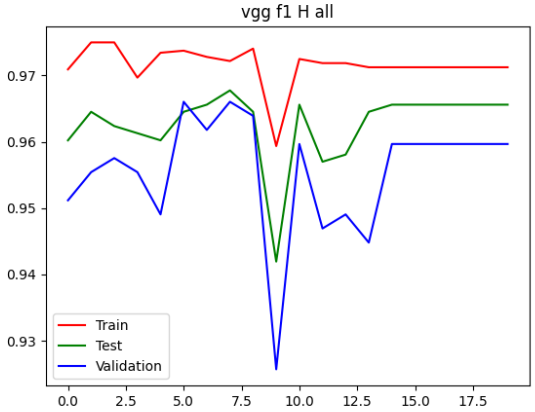



d) Duyarlılık (Recall)

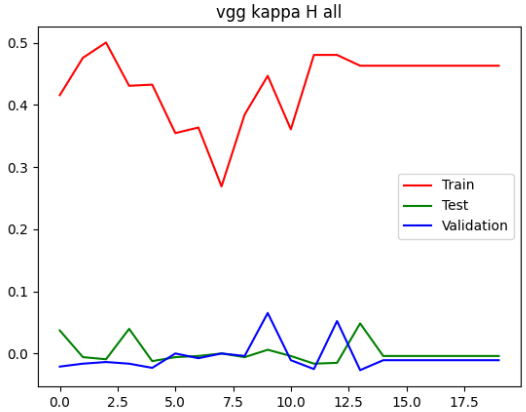
**Şekil 4.22.** Hipertansiyon sınıf kategorisine ait VGG16 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi



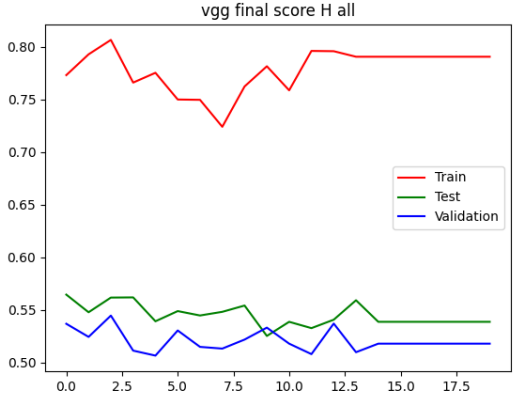

e) Eğri altında kalan alan (AUC)




f) F1 skoru

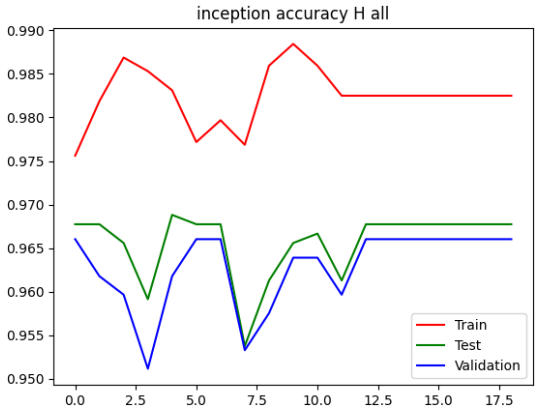



g) Kappa

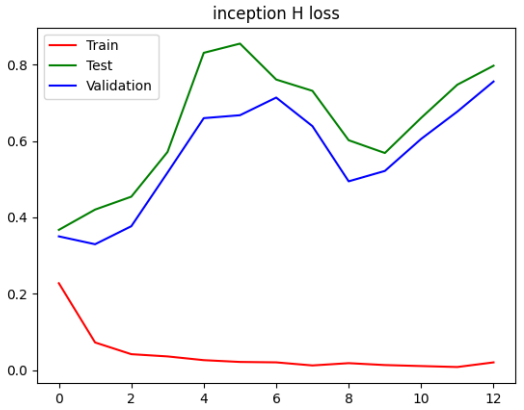



h) Final Skoru

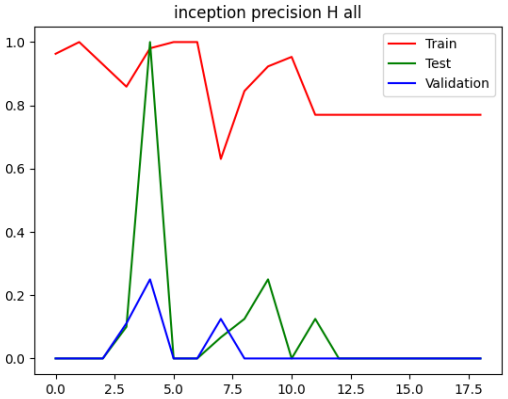
**Şekil 4.22.** Hipertansiyon sınıf kategorisine ait VGG16 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)



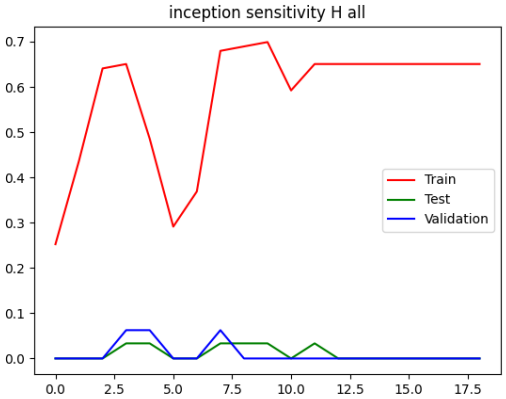

a) Doğruluk (Accuracy)




b) Hata (Loss)

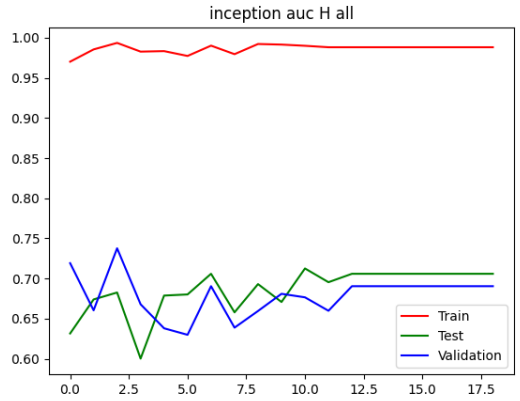



c) Kesinlik (Precision)

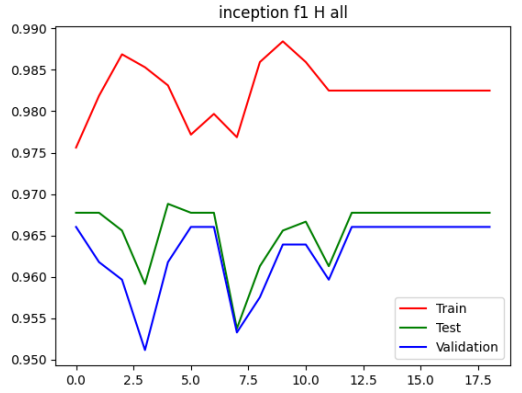



d) Duyarlılık (Recall)

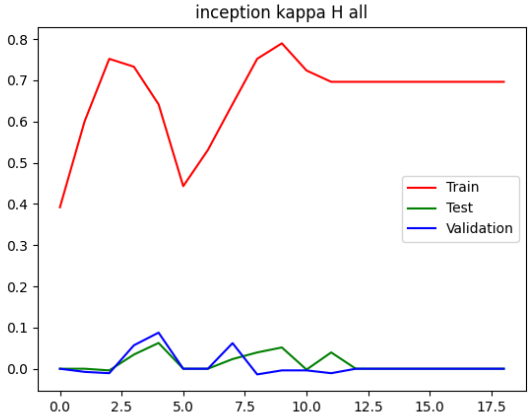
**Şekil 4.23.** Hipertansiyon sınıf kategorisine ait Inceptionv3 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi



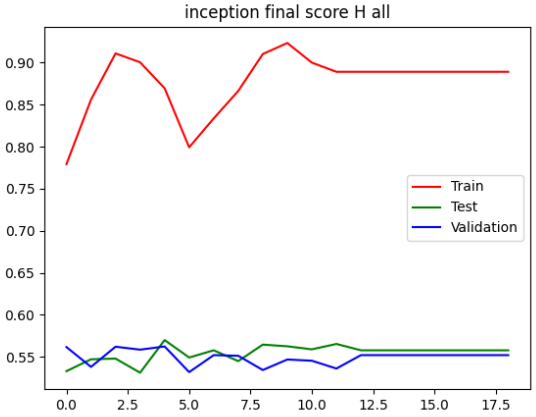

e) Eğri altında kalan alan (AUC)




f) F1 skoru

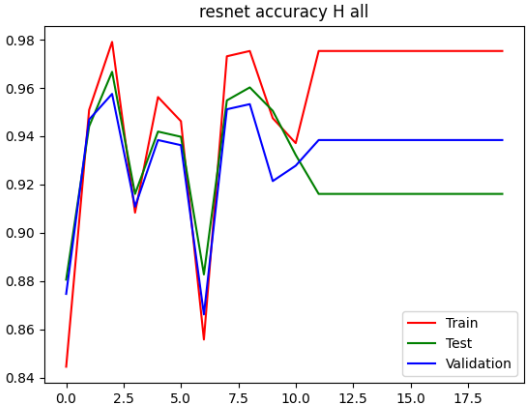



g) Kappa

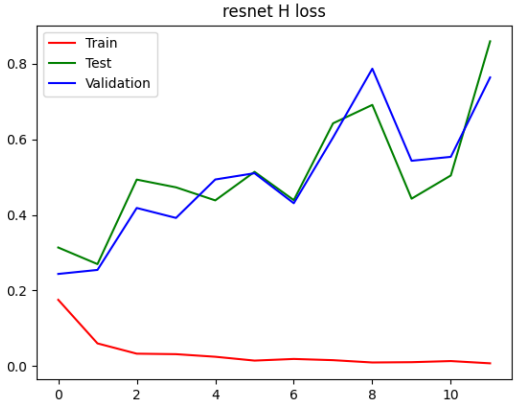



h) Final Skoru

**Şekil 4.23.** Hipertansiyon sınıf kategorisine ait Inceptionv3 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)

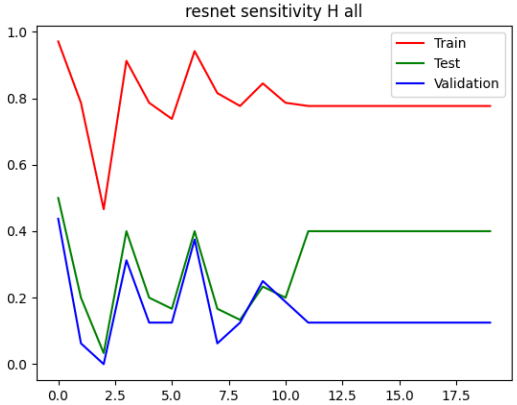



a) Doğruluk (Accuracy)




b) Hata (Loss)

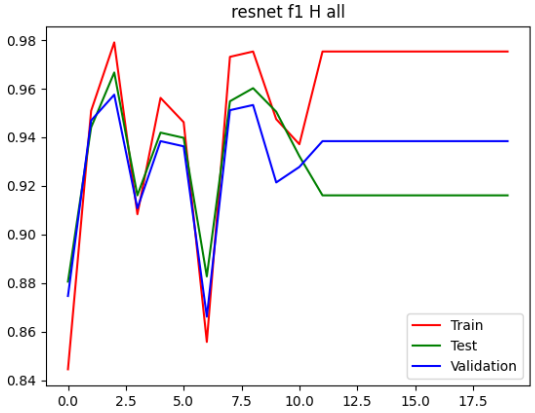

c) Kesinlik (Precision)



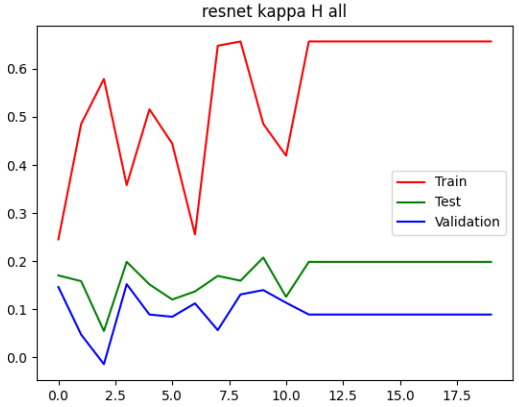

d) Duyarlılık (Recall)

**Şekil 4.24.** Hipertansiyon sınıf kategorisine ait ResNet50 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi

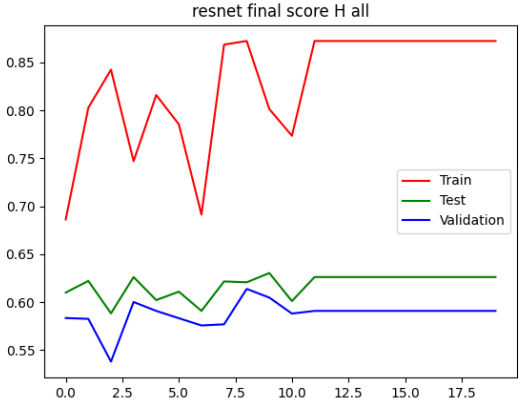

e) Eğri altında kalan alan (AUC)




f) F1 skoru




g) Kappa




h) Final Skoru

**Şekil 4.24.** Hipertansiyon sınıf kategorisine ait ResNet50 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)

## 4.7. Miyop Hastalık Sınıf Kategorisine ait Derin Konvolüsyonel Sinir Ağı Mimari Modelleri Sınıflandırma Performans Bulguları

Eğitim, doğrulama ve test seti için VGG16, Inceptionv3, ResNet50 modellerine ait her bir epokta (epoch) hata (loss) ve doğruluk (accuracy) değerleri Tablo 4.13'deki gibi elde edilmiştir. Eğitim, doğrulama ve test seti doğruluk değerleri sırası ile VGG16 modeli, 16 epok sonunda 0.995, 0.972 ve 0.920; Inceptionv3 modeli, 15 epok sonunda 0.999, 0.972 ve 0.945; ResNet50 modeli, 17 epok sonunda 1.000, 0.977 ve 0.937 olarak bulunmuştur. Inceptionv3 modelinin test seti için en yüksek doğruluğa sahip olduğu görülmektedir.

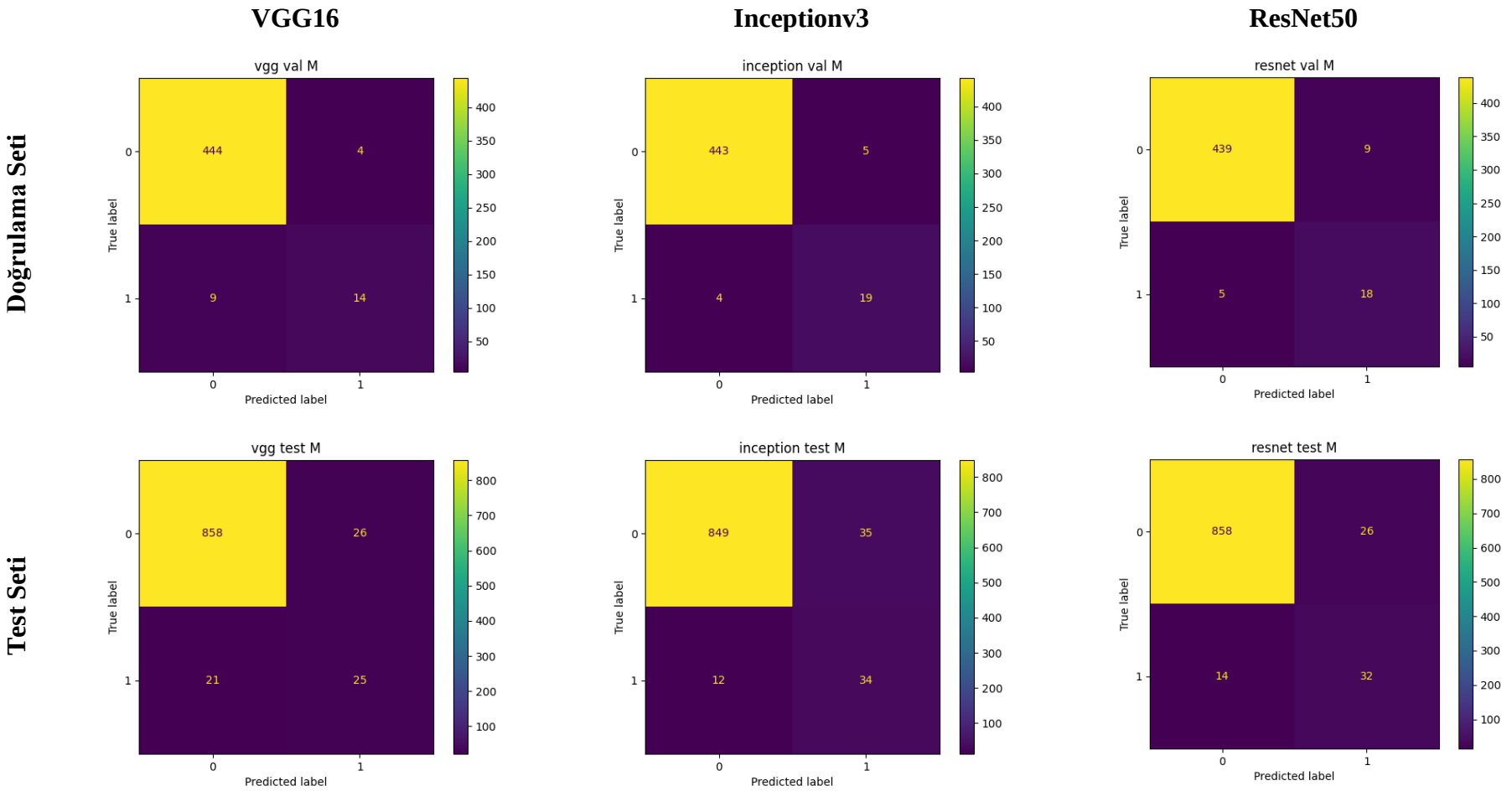
Eğitim, doğrulama ve test seti hata değerleri sırası ile VGG16 modeli, 16 epok sonunda 0.273, 3.324 ve 10.660; Inceptionv3 modeli, 15 epok sonunda 0.005, 0.289 ve 0.902; ResNet50 modeli, 0.000, 0.321 ve 0.976 olarak bulunmuştur. Inceptionv3 modelinin test seti için en düşük hata değerine sahip olduğu görülmektedir.

Doğrulama ve test seti için miyop sınıf kategorisine ait karmaşıklık matrisi Şekil 4.25'de yer almaktadır. Doğrulama setinde, toplam 471 fundus görüntüsünden 23'ünün miyop sınıfına ait olduğu bilinmektedir. VGG16 modeli, 14; Inceptionv3 modeli, 19; ResNet50 modeli, 18'ini doğru olarak sınıflandırmıştır. Miyop olmayan sınıf kategorisine sahip 448 fundus görüntüsü bulunmaktadır. VGG16 modeli, 444; Inceptionv3 modeli, 443; ResNet50 modeli, 439'unu doğru olarak sınıflandırmıştır. Test setinde toplam 930 fundus görüntüsünden 46'sının miyop sınıfına ait olduğu bilinmektedir. VGG16 modeli, 25; Inceptionv3 modeli, 34; ResNet50 modeli, 32'sini doğru olarak sınıflandırmıştır. Miyop olmayan sınıf kategorisine sahip 884 fundus görüntüsünden, VGG16 modeli, 858; Inceptionv3 modeli, 849; ResNet50 modeli, 858'ini doğru olarak sınıflandırmıştır.

Karmaşıklık matrisi elde edildikten sonra her bir modelin sınıflandırma başarı ölçütleri hesaplanarak Tablo 4.14'de sunulmuştur. VGG16, Inceptionv3, ResNet50 modelleri için her bir epokta doğruluk, hata, kesinlik, duyarlılık, Eğri altında kalan alan (AUC), F1 skoru, Kappa ve Final değeri sonuçları grafiksel olarak Şekil 4.26, Şekil 4.27, Şekil 4.28'de sunulmuştur. Miyop sınıfı test setinde VGG16, Inceptionv3, ResNet50 modelleri için Final skor değerleri sırası ile 0.767, 0.818 ve 0.836 olarak hesaplanmıştır. En yüksek Final skoru ResNet50 modeli ile elde edilmiştir.

**Tablo 4.13.** Eğitim, doğrulama ve test seti için Miyop Sınıf Kategorisine ait VGG16, Inceptionv3 ve ResNet50 Modeli Hata ve Doğruluk değerleri

| | | Eğitim seti | | Doğrulama seti | | Test seti | |
|---|---|---|---|---|---|---|---|
| Model | Epok | Hata (Loss) | Doğruluk (Accuracy) | Hata (Loss) | Doğruluk (Accuracy) | Hata (Loss) | Doğruluk (Accuracy) |
| VGG16 | 1 | 0,346 | 0,946 | 1,417 | 0,928 | 3,831 | 0,856 |
| | 2 | 0,254 | 0,973 | 0,436 | 0,970 | 1,572 | 0,915 |
| | 3 | 0,207 | 0,982 | 0,590 | 0,972 | 1,735 | 0,938 |
| | 4 | 0,219 | 0,986 | 0,451 | 0,972 | 1,833 | 0,925 |
| | 5 | 0,155 | 0,989 | 1,769 | 0,970 | 5,646 | 0,910 |
| | 6 | 0,229 | 0,989 | 1,377 | 0,968 | 5,246 | 0,910 |
| | 7 | 0,264 | 0,990 | 3,694 | 0,966 | 10,875 | 0,895 |
| | 8 | 0,192 | 0,991 | 1,170 | 0,975 | 3,234 | 0,928 |
| | 9 | 0,129 | 0,995 | 1,543 | 0,977 | 4,355 | 0,939 |
| | 10 | 0,212 | 0,992 | 2,445 | 0,972 | 7,426 | 0,918 |
| | 11 | 0,146 | 0,995 | 2,040 | 0,975 | 6,850 | 0,927 |
| | 12 | 0,202 | 0,994 | 2,119 | 0,970 | 7,079 | 0,919 |
| | 13 | 0,153 | 0,996 | 3,952 | 0,968 | 12,119 | 0,911 |
| | 14 | 0,127 | 0,996 | 2,989 | 0,970 | 9,726 | 0,924 |
| | 15 | 0,157 | 0,996 | 3,371 | 0,970 | 8,487 | 0,931 |
| | 16 | 0,273 | 0,995 | 3,324 | 0,972 | 10,660 | 0,920 |
| Inceptionv3 | 1 | 0,065 | 0,985 | 0,147 | 0,966 | 0,289 | 0,937 |
| | 2 | 0,021 | 0,995 | 0,208 | 0,966 | 0,472 | 0,931 |
| | 3 | 0,008 | 0,998 | 0,184 | 0,97 | 0,433 | 0,937 |
| | 4 | 0,008 | 0,998 | 0,182 | 0,968 | 0,611 | 0,935 |
| | 5 | 0,01 | 0,998 | 0,215 | 0,96 | 0,859 | 0,917 |
| | 6 | 0,008 | 0,998 | 0,449 | 0,968 | 1,177 | 0,925 |
| | 7 | 0,005 | 0,999 | 0,305 | 0,968 | 0,787 | 0,933 |
| | 8 | 0,005 | 0,999 | 0,190 | 0,977 | 0,669 | 0,948 |
| | 9 | 0,003 | 0,999 | 0,293 | 0,972 | 0,875 | 0,935 |
| | 10 | 0,007 | 0,999 | 0,274 | 0,97 | 0,767 | 0,943 |
| | 11 | 0,001 | 0,999 | 0,32 | 0,968 | 0,892 | 0,944 |
| | 12 | 0,006 | 0,999 | 0,315 | 0,968 | 0,796 | 0,946 |
| | 13 | 0,003 | 0,999 | 0,233 | 0,97 | 0,632 | 0,949 |
| | 14 | 0,000 | 1,000 | 0,26 | 0,968 | 0,672 | 0,943 |
| | 15 | 0,005 | 0,999 | 0,289 | 0,972 | 0,902 | 0,945 |
| ResNet50 | 1 | 0,051 | 0,983 | 0,121 | 0,979 | 0,418 | 0,920 |
| | 2 | 0,015 | 0,995 | 0,219 | 0,972 | 0,559 | 0,942 |
| | 3 | 0,009 | 0,997 | 0,251 | 0,975 | 0,773 | 0,941 |
| | 4 | 0,007 | 0,998 | 0,182 | 0,972 | 0,572 | 0,924 |
| | 5 | 0,006 | 0,998 | 0,288 | 0,970 | 0,830 | 0,912 |
| | 6 | 0,005 | 0,999 | 0,301 | 0,977 | 0,768 | 0,919 |
| | 7 | 0,004 | 0,999 | 0,281 | 0,983 | 0,838 | 0,934 |
| | 8 | 0,006 | 0,998 | 0,323 | 0,981 | 0,989 | 0,923 |
| | 9 | 0,002 | 1,000 | 0,369 | 0,970 | 1,233 | 0,914 |
| | 10 | 0,001 | 1,000 | 0,275 | 0,985 | 0,893 | 0,947 |
| | 11 | 0,003 | 0,999 | 0,193 | 0,979 | 1,104 | 0,929 |
| | 12 | 0,005 | 0,998 | 0,237 | 0,979 | 0,887 | 0,935 |
| | 13 | 0,002 | 0,999 | 0,252 | 0,981 | 0,745 | 0,941 |
| | 14 | 0,004 | 0,999 | 0,313 | 0,981 | 0,833 | 0,934 |
| | 15 | 0,003 | 0,999 | 0,271 | 0,972 | 0,671 | 0,941 |
| | 16 | 0,002 | 0,999 | 0,254 | 0,979 | 0,826 | 0,939 |
| | 17 | 0,000 | 1,000 | 0,321 | 0,977 | 0,976 | 0,937 |

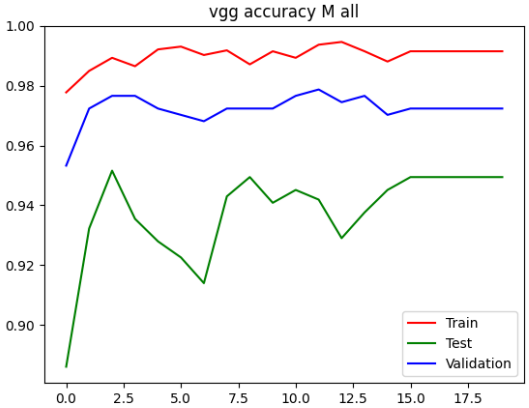



**Şekil 4.25.** Doğrulama ve test seti için VGG16, Inceptionv3 ve ResNet50 Modeli Miyop Sınıf Kategorisine ait Karmaşıklık Matrisi

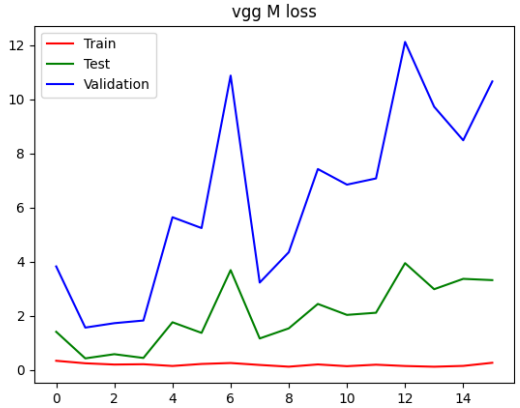
**Tablo 4.14.** Doğrulama ve test seti için Miyop Sınıf Kategorisine ait VGG16, Inceptionv3 ve ResNet50 Modeli sınıflandırma başarı ölçütleri

| Model | Performans ölçütleri | Doğrulama seti | Test seti |
|---|---|---|---|
| VGG16 | Accuracy | 0,972 | 0,949 |
| | Precision | 0,778 | 0,490 |
| | Sens | 0,609 | 0,543 |
| | Spec | 0,991 | 0,971 |
| | $\kappa$ | 0,669 | 0,489 |
| | F1 | 0,972 | 0,949 |
| | AUC | 0,967 | 0,861 |
| | Final | 0,870 | 0,767 |
| Inceptionv3 | Accuracy | 0,981 | 0,949 |
| | Precision | 0,792 | 0,493 |
| | Sens | 0,826 | 0,739 |
| | Spec | 0,989 | 0,960 |
| | $\kappa$ | 0,798 | 0,566 |
| | F1 | 0,981 | 0,949 |
| | AUC | 0,991 | 0,940 |
| | Final | 0,923 | 0,818 |
| ResNet50 | Accuracy | 0,970 | 0,957 |
| | Precision | 0,667 | 0,552 |
| | Sens | 0,783 | 0,696 |
| | Spec | 0,980 | 0,971 |
| | $\kappa$ | 0,704 | 0,593 |
| | F1 | 0,970 | 0,957 |
| | AUC | 0,989 | 0,959 |
| | Final | 0,888 | 0,836 |

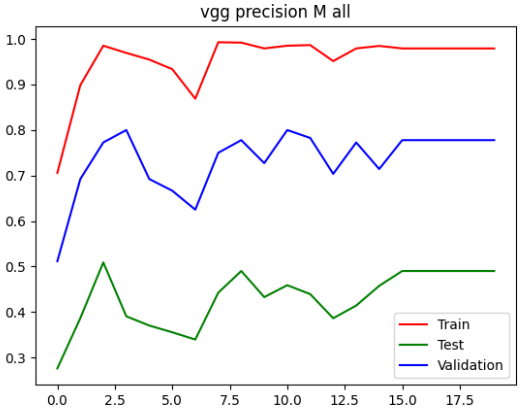
Accuracy: Doğruluk; Precision: Kesinlik; Sens: Duyarlılık; Spec: Özgüllük; $\kappa$: Kappa; AUC: Eğri altında kalan alan; Final: F1, AUC ve $\kappa$ değeri ortalaması



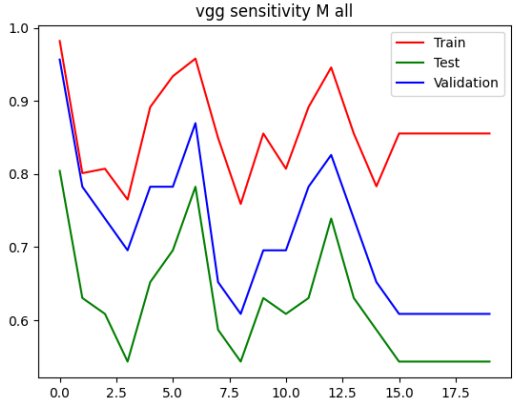

a) Doğruluk (Accuracy)




b) Hata (Loss)

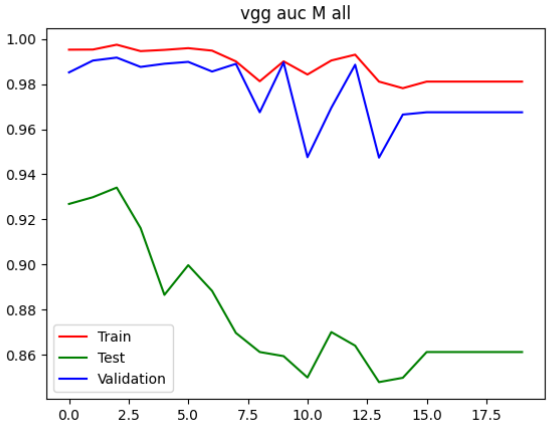



c) Kesinlik (Precision)

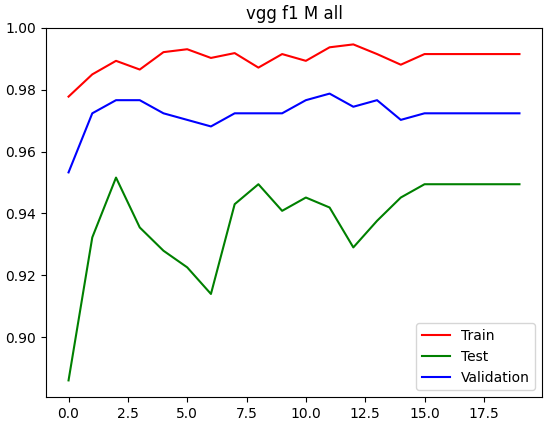



d) Duyarlılık (Recall)

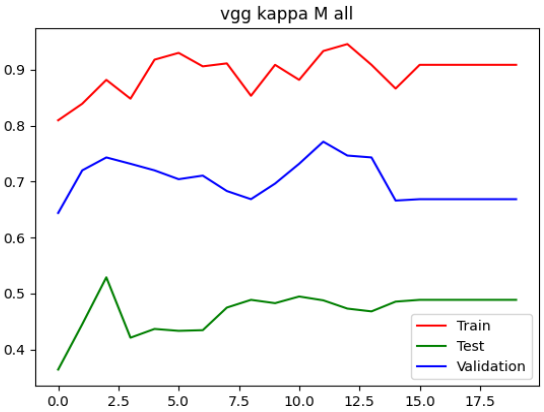
**Şekil 4.26.** Miyop sınıf kategorisine ait VGG16 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi



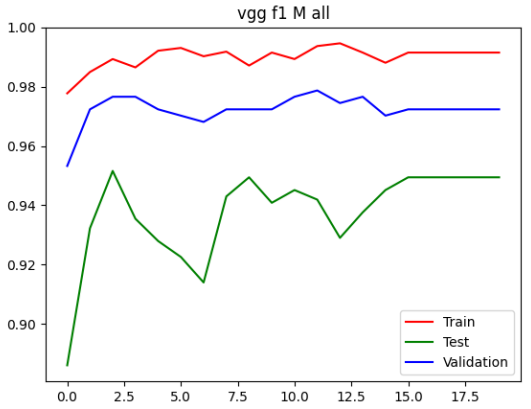

e) Eğri altında kalan alan (AUC)




f) F1 skoru

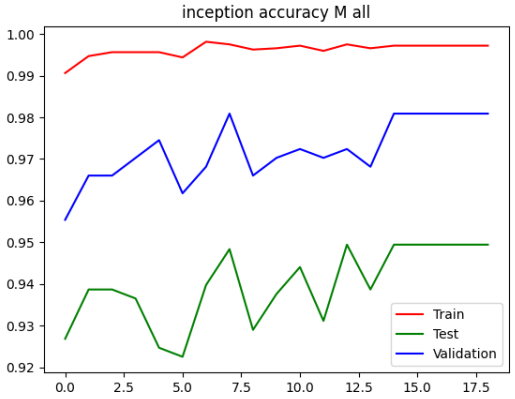



g) Kappa

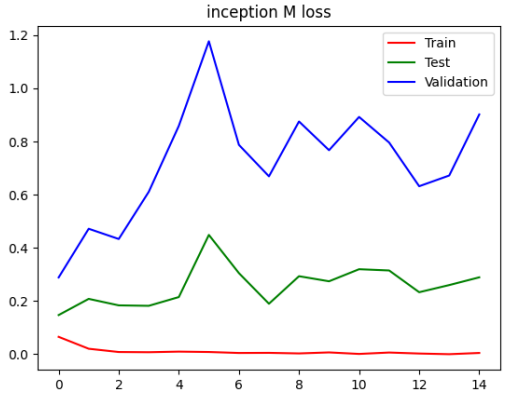



h) Final Skoru

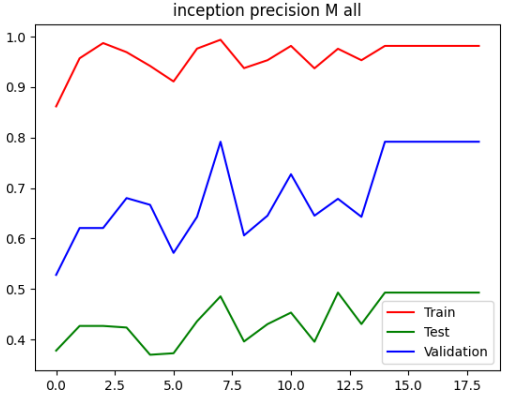
**Şekil 4.26.** Miyop sınıf kategorisine ait VGG16 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)



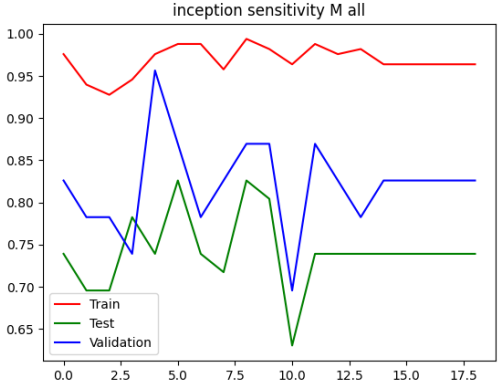

a) Doğruluk (Accuracy)




b) Hata (Loss)

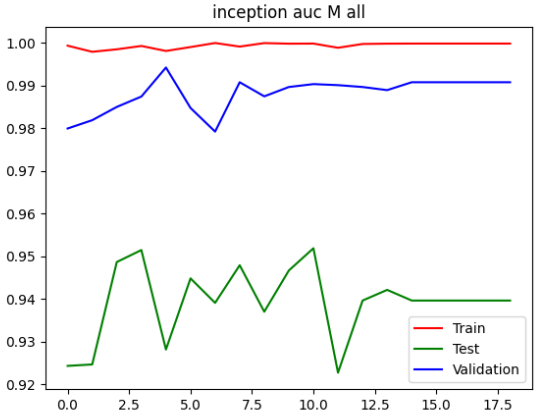



c) Kesinlik (Precision)

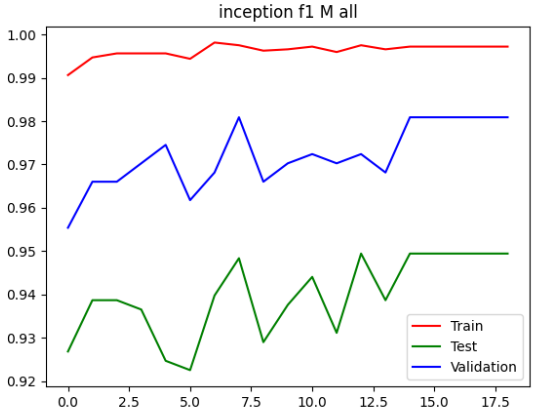



d) Duyarlılık (Recall)

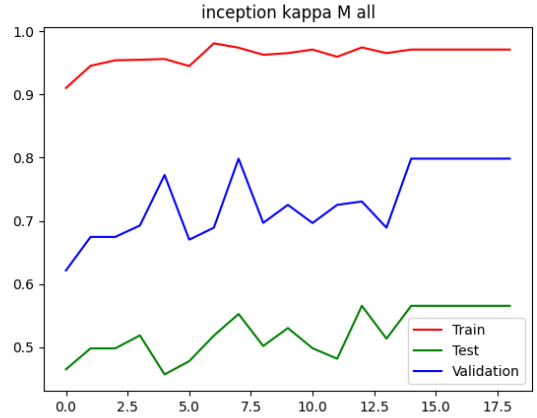
**Şekil 4.27.** Miyop sınıf kategorisine ait Inceptionv3 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi



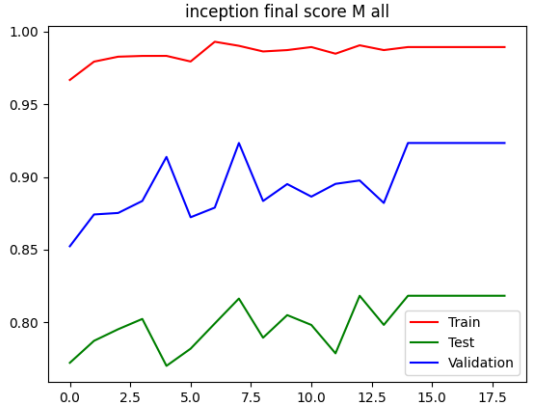

e) Eğri altında kalan alan (AUC)




f) F1 skoru

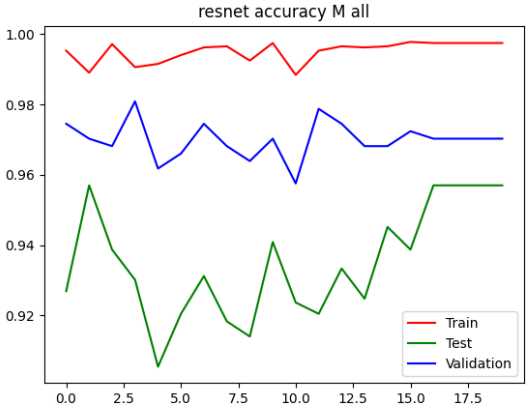



g) Kappa

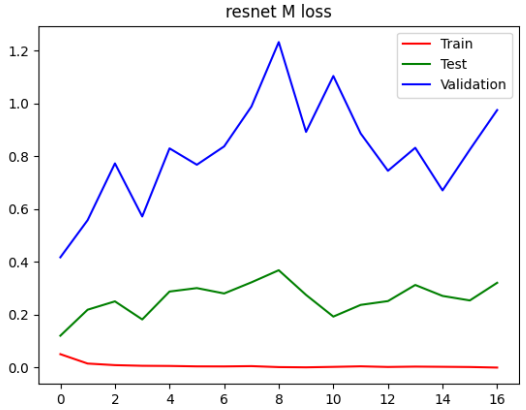



h) Final Skoru

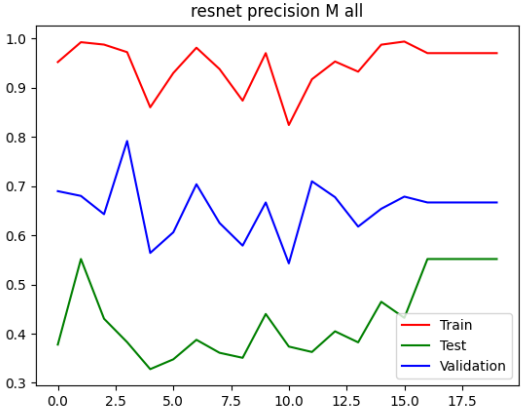
**Şekil 4.27.** Miyop sınıf kategorisine ait Inceptionv3 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)



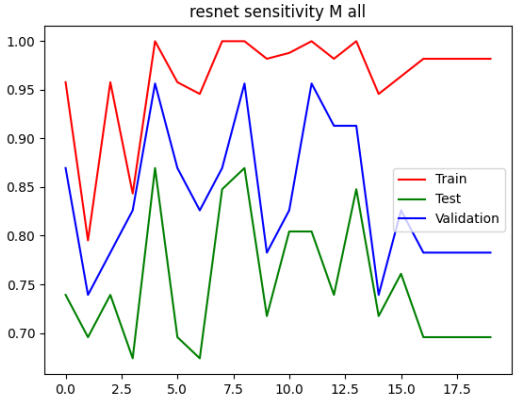

a) Doğruluk (Accuracy)




b) Hata (Loss)



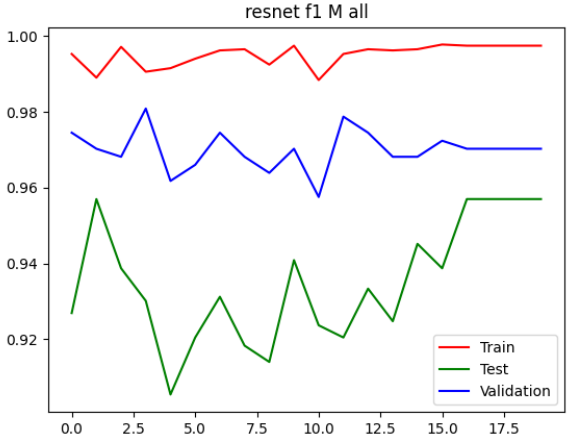

c) Kesinlik (Precision)



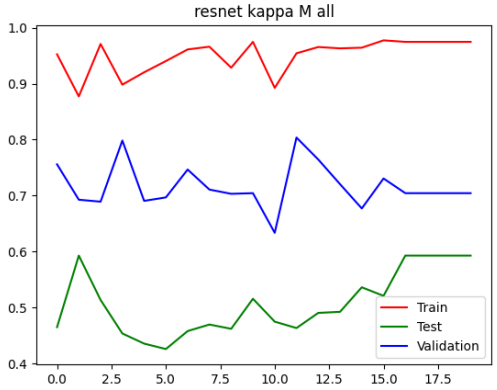

d) Duyarlılık (Recall)

**Şekil 4.28.** Miyop sınıf kategorisine ait ResNet50 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi

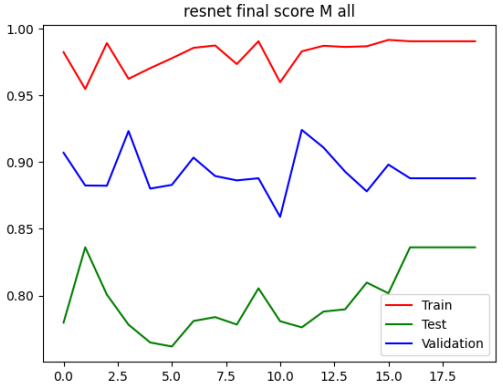

e) Eğri altında kalan alan (AUC)




f) F1 skoru




g) Kappa




h) Final Skoru

**Şekil 4.28.** Miyop sınıf kategorisine ait ResNet50 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)

## 4.8. Diğer Hastalıklar Sınıf Kategorisine ait Derin Konvolüsyonel Sinir Ağı Mimari Modelleri Sınıflandırma Performans Bulguları

Eğitim, doğrulama ve test seti için VGG16, Inceptionv3, ResNet50 modellerine ait her bir epokta (epoch) hata (loss) ve doğruluk (accuracy) değerleri Tablo 4.15'deki gibi elde edilmiştir. Eğitim, doğrulama ve test seti doğruluk değerleri sırası ile VGG16 modeli, 10 epok sonunda 0.645, 0.501 ve 0.556; Inceptionv3 modeli, 10 epok sonunda 0.992, 0.667 ve 0.643; ResNet50 modeli, 16 epok sonunda 0.995, 0.709 ve 0.710 olarak bulunmuştur. ResNet50 modelinin test seti için en yüksek doğruluğa sahip olduğu görülmektedir.

Eğitim, doğrulama ve test seti hata değerleri sırası ile VGG16 modeli, 10 epok sonunda 0.593, 1.016 ve 0.957; Inceptionv3 modeli, 10 epok sonunda 0.039, 2.455 ve 3.048; ResNet50 modeli, 0.016, 2.423 ve 2.829 olarak bulunmuştur. VGG16 modelinin test seti için en düşük hata değerine sahip olduğu görülmektedir.

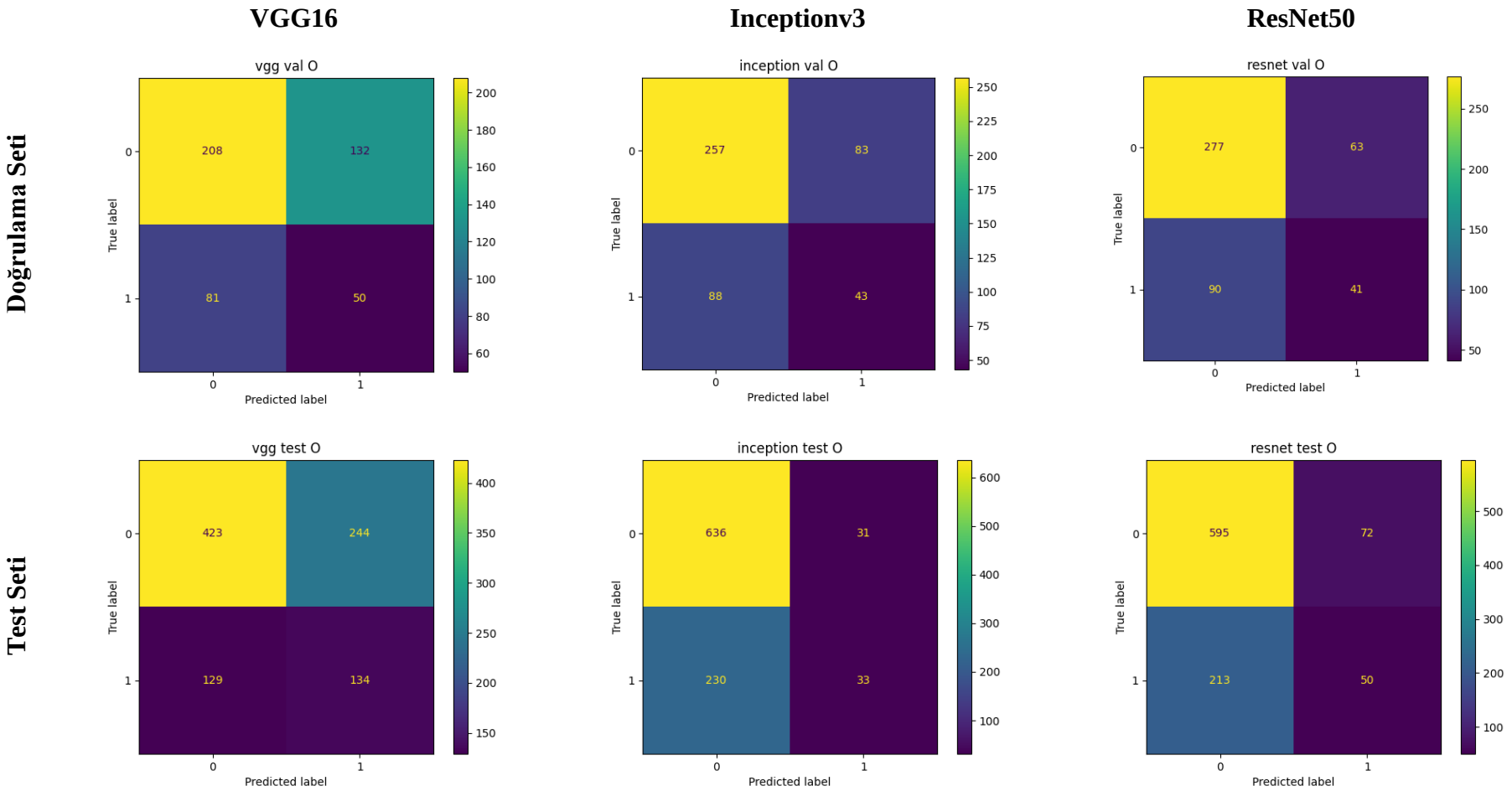
Doğrulama ve test seti için diğer hastalıklar sınıf kategorisine ait karmaşıklık matrisi Şekil 4.29'da yer almaktadır. Doğrulama setinde, toplam 471 fundus görüntüsünden 131'inin diğer hastalıklar sınıfına ait olduğu bilinmektedir. VGG16 modeli, 50; Inceptionv3 modeli, 43; ResNet50 modeli, 41'ini doğru olarak sınıflandırmıştır. Diğer hastalıklar sınıfına dahil olmayan 340 fundus görüntüsü bulunmaktadır. VGG16 modeli, 208; Inceptionv3 modeli, 257; ResNet50 modeli, 277'sini doğru olarak sınıflandırmıştır. Test setinde toplam 930 fundus görüntüsünden 263'ünün diğer hastalıklar sınıfına ait olduğu bilinmektedir. VGG16 modeli, 134; Inceptionv3 modeli, 33; ResNet50 modeli, 50'sini doğru olarak sınıflandırmıştır. Diğer hastalıklar sınıfına dahil olmayan 667 fundus görüntüsünden, VGG16 modeli, 423; Inceptionv3 modeli, 636; ResNet50 modeli, 595'ini doğru olarak sınıflandırmıştır.

Karmaşıklık matrisi elde edildikten sonra her bir modelin sınıflandırma başarı ölçütleri hesaplanarak Tablo 4.16'da sunulmuştur. VGG16, Inceptionv3, ResNet50 modelleri için her bir epokta doğruluk, hata, kesinlik, duyarlılık, Eğri altında kalan alan (AUC), F1 skoru, Kappa ve Final değeri sonuçları grafiksel olarak Şekil 4.30, Şekil 4.31, Şekil 4.32'de sunulmuştur.

Diğer hastalıklar sınıfı test setinde VGG16, Inceptionv3, ResNet50 modelleri için Final skor değerleri sırası ile 0.437, 0.481 ve 0.456 olarak hesaplanmıştır. En yüksek Final skoru Inceptionv3 modeli ile elde edilmiştir.

**Tablo 4.15.** Eğitim, doğrulama ve test seti için Diğer Sınıf Kategorisine ait VGG16, Inceptionv3 ve ResNet50 Modeli Hata ve Doğruluk değerleri

| | | Eğitim seti | | Doğrulama seti | | Test seti | |
|---|---|---|---|---|---|---|---|
| Model | Epok | Hata (Loss) | Doğruluk (Accuracy) | Hata (Loss) | Doğruluk (Accuracy) | Hata (Loss) | Doğruluk (Accuracy) |
| VGG16 | 1 | 0,818 | 0,523 | 0,680 | 0,520 | 0,661 | 0,570 |
| | 2 | 0,720 | 0,533 | 0,686 | 0,454 | 0,666 | 0,512 |
| | 3 | 0,699 | 0,562 | 0,655 | 0,673 | 0,630 | 0,682 |
| | 4 | 0,680 | 0,585 | 0,717 | 0,393 | 0,698 | 0,383 |
| | 5 | 0,673 | 0,610 | 0,780 | 0,497 | 0,688 | 0,580 |
| | 6 | 0,660 | 0,625 | 0,852 | 0,505 | 0,751 | 0,584 |
| | 7 | 0,639 | 0,623 | 0,945 | 0,565 | 0,842 | 0,614 |
| | 8 | 0,620 | 0,639 | 0,977 | 0,535 | 0,832 | 0,600 |
| | 9 | 0,589 | 0,646 | 0,991 | 0,556 | 0,840 | 0,599 |
| | 10 | 0,593 | 0,645 | 1,016 | 0,501 | 0,957 | 0,556 |
| Inceptionv3 | 1 | 0,782 | 0,688 | 0,919 | 0,669 | 1,073 | 0,688 |
| | 2 | 0,313 | 0,908 | 1,285 | 0,696 | 1,301 | 0,706 |
| | 3 | 0,162 | 0,954 | 1,435 | 0,718 | 1,576 | 0,700 |
| | 4 | 0,114 | 0,971 | 2,040 | 0,686 | 2,143 | 0,698 |
| | 5 | 0,080 | 0,979 | 2,240 | 0,677 | 2,382 | 0,696 |
| | 6 | 0,067 | 0,984 | 1,912 | 0,701 | 1,952 | 0,696 |
| | 7 | 0,055 | 0,987 | 2,081 | 0,711 | 2,506 | 0,695 |
| | 8 | 0,048 | 0,989 | 2,133 | 0,665 | 2,406 | 0,669 |
| | 9 | 0,049 | 0,988 | 2,258 | 0,692 | 2,426 | 0,660 |
| | 10 | 0,039 | 0,992 | 2,455 | 0,667 | 3,048 | 0,643 |
| ResNet50 | 1 | 0,558 | 0,701 | 1,061 | 0,718 | 1,030 | 0,723 |
| | 2 | 0,220 | 0,904 | 1,205 | 0,639 | 1,158 | 0,640 |
| | 3 | 0,118 | 0,951 | 1,787 | 0,724 | 1,811 | 0,730 |
| | 4 | 0,084 | 0,966 | 2,102 | 0,705 | 2,484 | 0,691 |
| | 5 | 0,061 | 0,978 | 1,978 | 0,658 | 1,837 | 0,668 |
| | 6 | 0,051 | 0,980 | 1,755 | 0,715 | 1,757 | 0,710 |
| | 7 | 0,040 | 0,986 | 2,278 | 0,699 | 2,059 | 0,709 |
| | 8 | 0,036 | 0,987 | 1,711 | 0,703 | 1,895 | 0,689 |
| | 9 | 0,032 | 0,988 | 2,263 | 0,743 | 2,363 | 0,716 |
| | 10 | 0,032 | 0,989 | 1,941 | 0,722 | 1,983 | 0,702 |
| | 11 | 0,026 | 0,991 | 2,543 | 0,607 | 2,450 | 0,596 |
| | 12 | 0,028 | 0,990 | 2,362 | 0,662 | 2,378 | 0,671 |
| | 13 | 0,022 | 0,993 | 2,000 | 0,718 | 2,026 | 0,702 |
| | 14 | 0,021 | 0,992 | 2,375 | 0,656 | 2,242 | 0,659 |
| | 15 | 0,019 | 0,993 | 2,430 | 0,665 | 2,448 | 0,663 |
| | 16 | 0,016 | 0,995 | 2,423 | 0,709 | 2,829 | 0,710 |

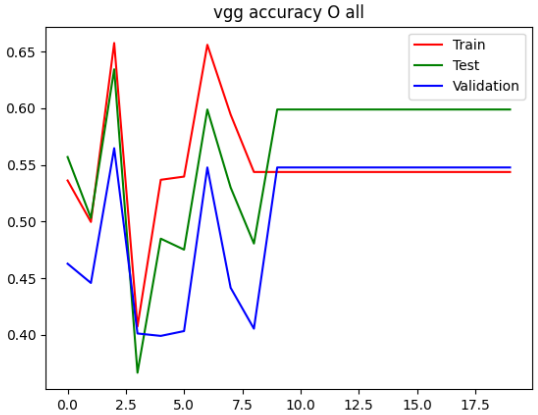



**Şekil 4.29.** Doğrulama ve test seti için VGG16, Inceptionv3 ve ResNet50 Modeli Diğer Sınıf Kategorisine ait Karmaşıklık Matrisi

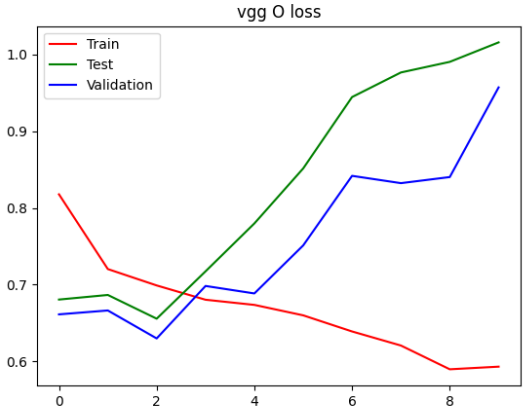
**Tablo 4.16.** Doğrulama ve test seti için Diğer Sınıf Kategorisine ait VGG16, Inceptionv3 ve ResNet50 Modeli sınıflandırma başarı ölçütleri

| Model | Performans ölçütleri | Doğrulama seti | Test seti |
|---|---|---|---|
| VGG16 | Accuracy | 0,548 | 0,599 |
| | Precision | 0,275 | 0,354 |
| | Sens | 0,382 | 0,51 |
| | Spec | 0,612 | 0,634 |
| | $\kappa$ | -0,006 | 0,127 |
| | F1 | 0,548 | 0,599 |
| | AUC | 0,504 | 0,586 |
| | Final | 0,349 | 0,437 |
| Inceptionv3 | Accuracy | 0,637 | 0,719 |
| | Precision | 0,341 | 0,516 |
| | Sens | 0,328 | 0,125 |
| | Spec | 0,756 | 0,954 |
| | $\kappa$ | 0,085 | 0,102 |
| | F1 | 0,637 | 0,719 |
| | AUC | 0,556 | 0,621 |
| | Final | 0,426 | 0,481 |
| ResNet50 | Accuracy | 0,675 | 0,694 |
| | Precision | 0,394 | 0,410 |
| | Sens | 0,313 | 0,190 |
| | Spec | 0,815 | 0,892 |
| | $\kappa$ | 0,136 | 0,098 |
| | F1 | 0,675 | 0,694 |
| | AUC | 0,608 | 0,575 |
| | Final | 0,473 | 0,456 |

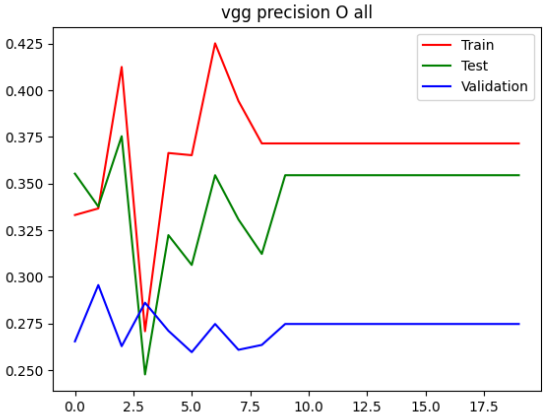
Accuracy: Doğruluk; Precision: Kesinlik; Sens: Duyarlılık; Spec: Özgüllük; $\kappa$: Kappa; AUC: Eğri altında kalan alan; Final: F1, AUC ve $\kappa$ değeri ortalaması



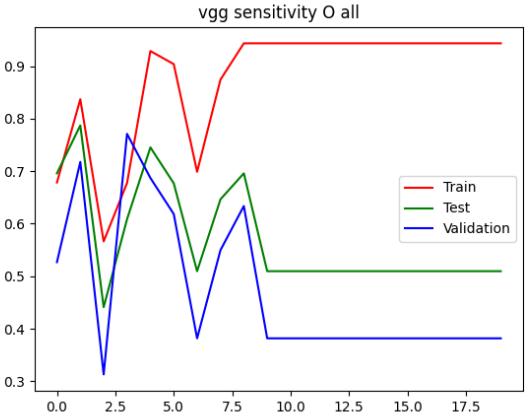

a) Doğruluk (Accuracy)




b) Hata (Loss)

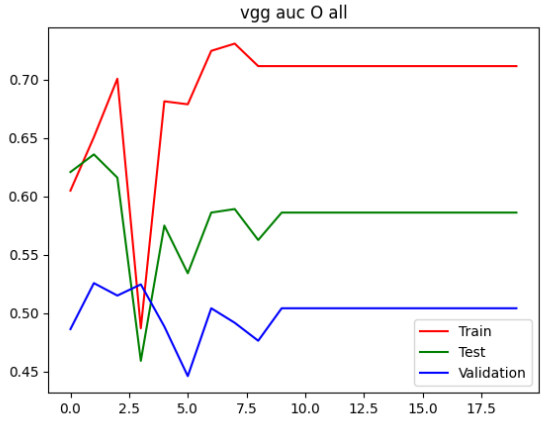



c) Kesinlik (Precision)

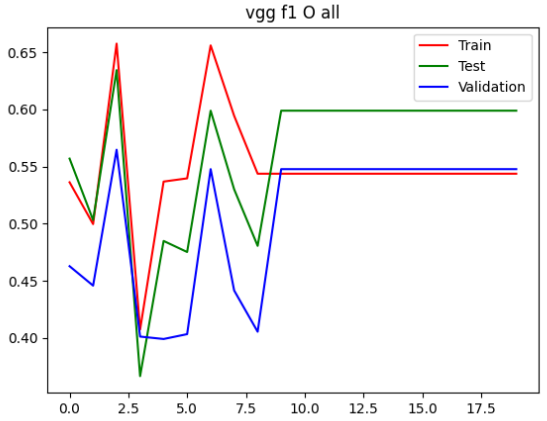



d) Duyarlılık (Recall)

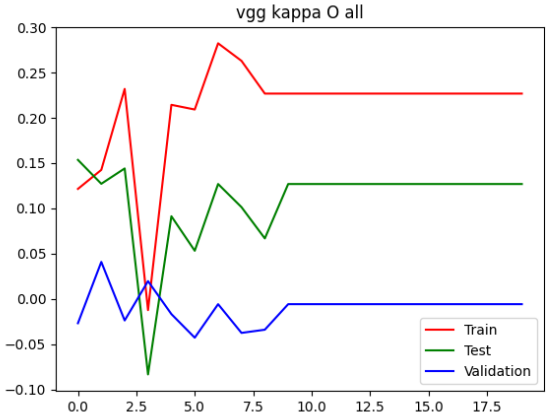
**Şekil 4.30.** Diğer sınıf kategorisine ait VGG16 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi



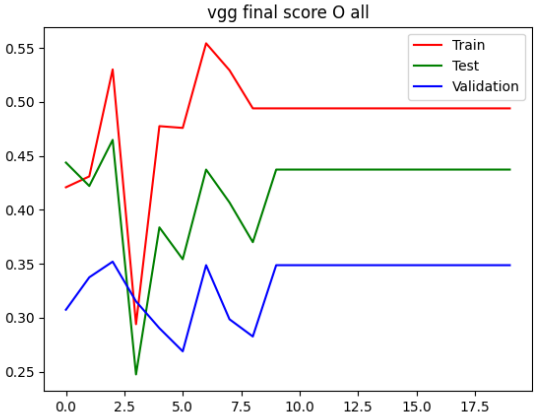

e) Eğri altında kalan alan (AUC)




f) F1 skoru

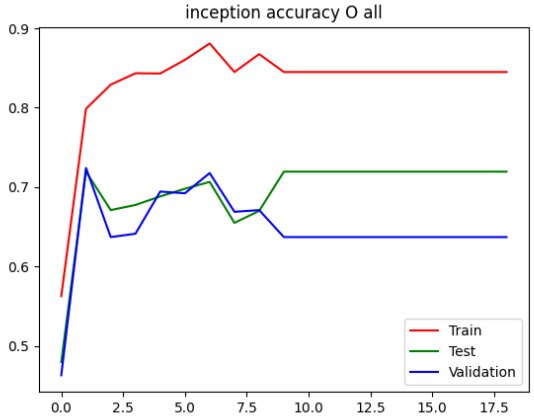



g) Kappa



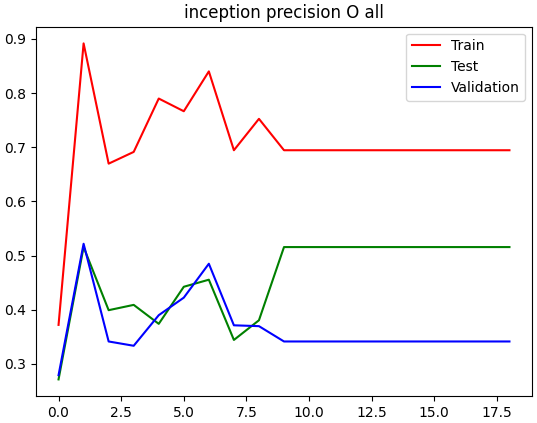

h) Final Skoru

**Şekil 4.30.** Diğer sınıf kategorisine ait VGG16 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)

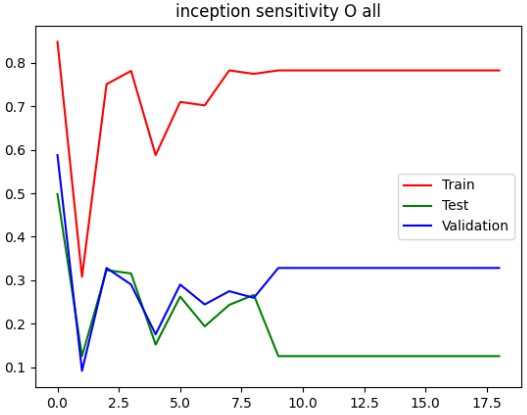



a) Doğruluk (Accuracy)


b) Hata (Loss)

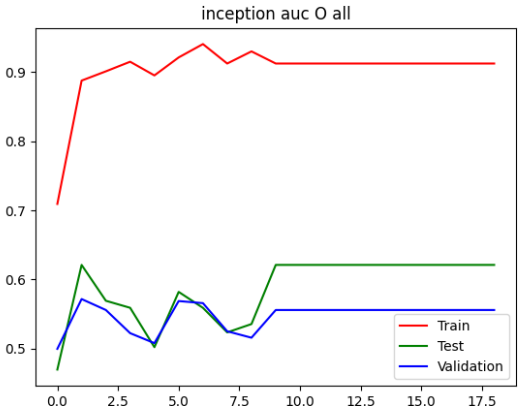



c) Kesinlik (Precision)

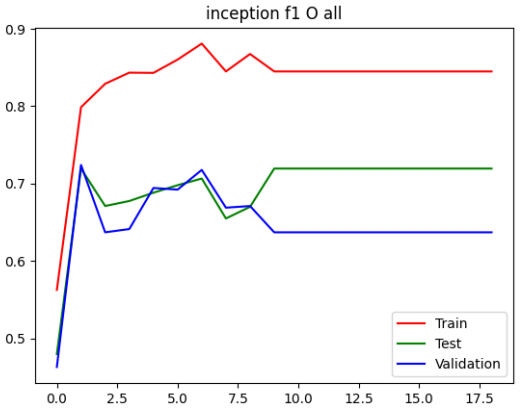



d) Duyarlılık (Recall)

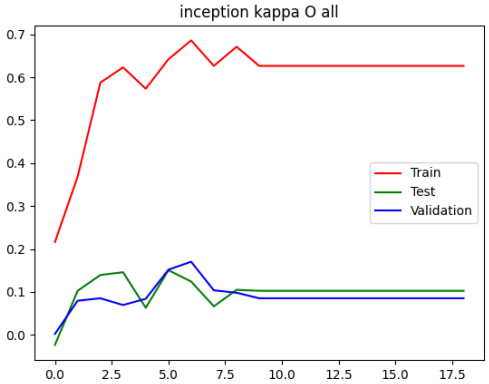
**Şekil 4.31.** Diğer sınıf kategorisine ait Inceptionv3 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi



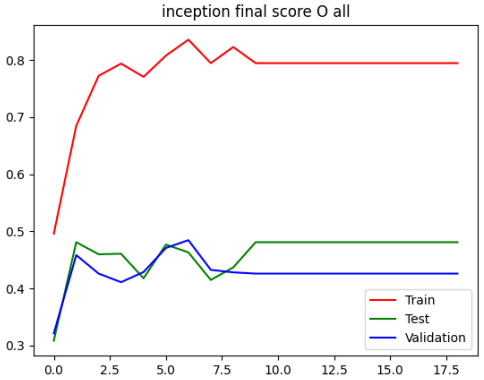

e) Eğri altında kalan alan (AUC)




f) F1 skoru

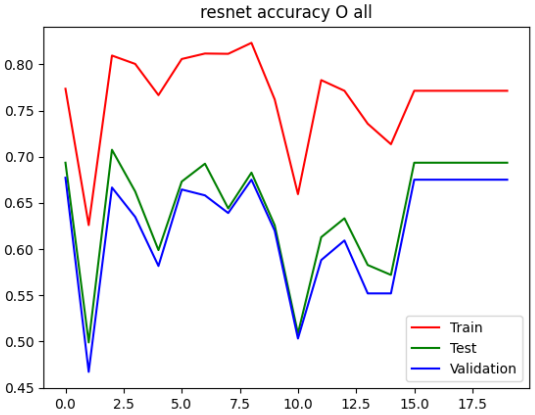



g) Kappa

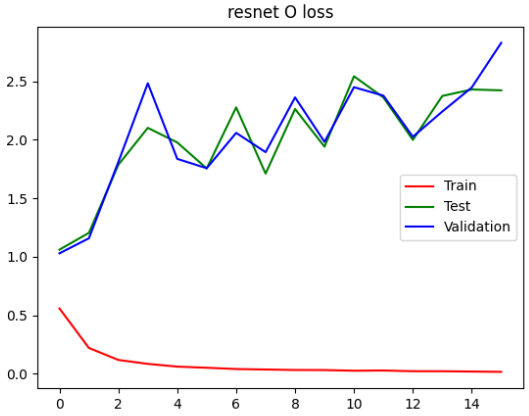



h) Final Skoru

**Şekil 4.31.** Diğer sınıf kategorisine ait Inceptionv3 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)

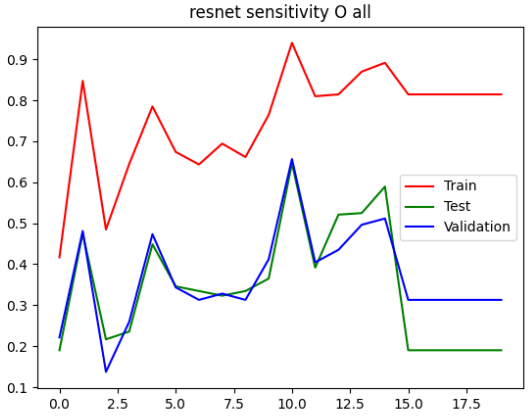



a) Doğruluk (Accuracy)



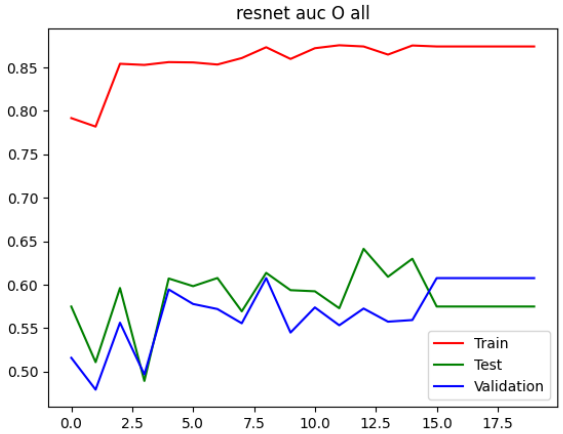

b) Hata (Loss)


c) Kesinlik (Precision)

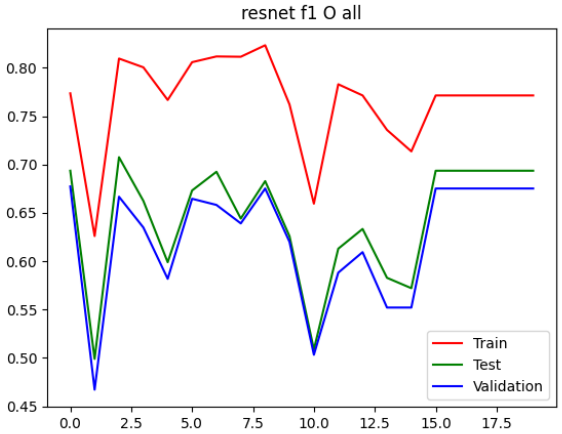



d) Duyarlılık (Recall)

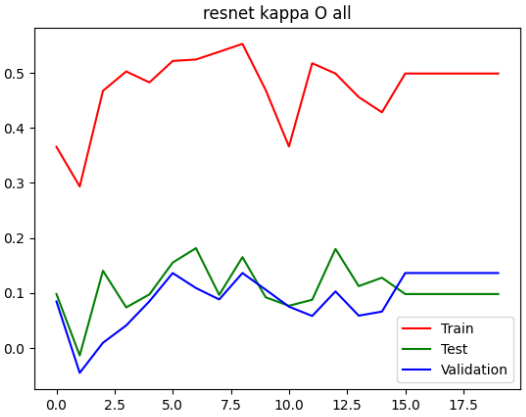
**Şekil 4.32.** Diğer sınıf kategorisine ait ResNet50 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi



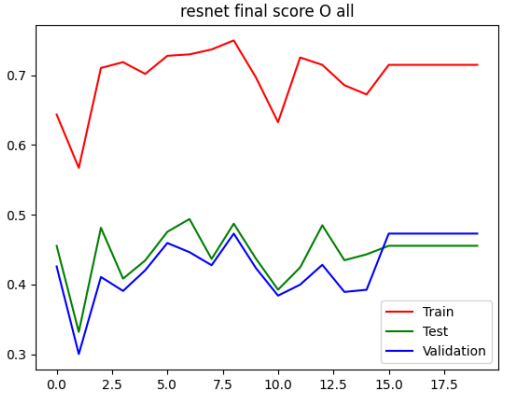

e) Eğri altında kalan alan (AUC)




f) F1 skoru




g) Kappa




h) Final Skoru

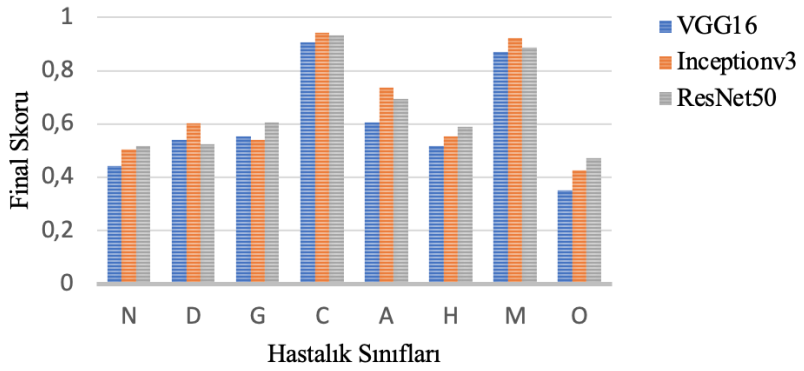
**Şekil 4.32.** Diğer sınıf kategorisine ait ResNet50 modeli sınıflandırma başarı ölçütlerinin değerlendirilmesi (Devam Ediyor)

**Tablo 4.17.** Derin Konvolüsyonel Sinir Ağı Modellerine ait Sınıflandırma Performanslarının Değerlendirilmesi

| | | Doğrulama Seti | | | | | | | | Test Seti | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | Kategori | Acc | Prec | Sens | Spec | $\kappa$ | F1 | AUC | Final | Acc | Prec | Sens | Spec | $\kappa$ | F1 | AUC | Final |
| VGG16 | N | 0,586 | 0,381 | 0,524 | 0,614 | 0,125 | 0,586 | 0,617 | 0,443 | 0,667 | 0,454 | 0,646 | 0,675 | 0,286 | 0,667 | 0,717 | 0,557 |
| | D | 0,665 | 0,500 | 0,487 | 0,754 | 0,243 | 0,665 | 0,709 | 0,539 | 0,653 | 0,494 | 0,561 | 0,700 | 0,253 | 0,653 | 0,677 | 0,528 |
| | G | 0,879 | 0,105 | 0,148 | 0,923 | 0,060 | 0,879 | 0,719 | 0,553 | 0,919 | 0,260 | 0,255 | 0,958 | 0,215 | 0,919 | 0,726 | 0,620 |
| | C | 0,972 | 0,767 | 0,793 | 0,984 | 0,765 | 0,972 | 0,988 | 0,908 | 0,962 | 0,841 | 0,569 | 0,992 | 0,660 | 0,962 | 0,949 | 0,857 |
| | A | 0,941 | 0,333 | 0,120 | 0,987 | 0,153 | 0,941 | 0,720 | 0,605 | 0,929 | 0,219 | 0,146 | 0,972 | 0,139 | 0,929 | 0,578 | 0,549 |
| | H | 0,960 | 0,000 | 0,000 | 0,993 | -0,011 | 0,960 | 0,604 | 0,518 | 0,966 | 0,000 | 0,000 | 0,998 | -0,004 | 0,966 | 0,654 | 0,538 |
| | M | 0,972 | 0,778 | 0,609 | 0,991 | 0,669 | 0,972 | 0,967 | 0,870 | 0,949 | 0,490 | 0,543 | 0,971 | 0,489 | 0,949 | 0,861 | 0,767 |
| | O | 0,548 | 0,275 | 0,382 | 0,612 | -0,006 | 0,548 | 0,504 | 0,349 | 0,599 | 0,354 | 0,510 | 0,634 | 0,127 | 0,599 | 0,586 | 0,437 |
| Inceptionv3 | N | 0,599 | 0,419 | 0,735 | 0,537 | 0,225 | 0,599 | 0,691 | 0,505 | 0,648 | 0,433 | 0,628 | 0,657 | 0,252 | 0,648 | 0,711 | 0,537 |
| | D | 0,726 | 0,649 | 0,399 | 0,891 | 0,321 | 0,726 | 0,758 | 0,602 | 0,686 | 0,671 | 0,166 | 0,957 | 0,151 | 0,686 | 0,726 | 0,521 |
| | G | 0,832 | 0,139 | 0,370 | 0,860 | 0,129 | 0,832 | 0,662 | 0,541 | 0,933 | 0,351 | 0,255 | 0,973 | 0,261 | 0,933 | 0,685 | 0,626 |
| | C | 0,983 | 0,839 | 0,897 | 0,989 | 0,858 | 0,983 | 0,987 | 0,943 | 0,968 | 0,797 | 0,723 | 0,986 | 0,741 | 0,968 | 0,948 | 0,885 |
| | A | 0,945 | 0,481 | 0,520 | 0,969 | 0,471 | 0,945 | 0,796 | 0,737 | 0,940 | 0,389 | 0,292 | 0,975 | 0,302 | 0,940 | 0,748 | 0,663 |
| | H | 0,966 | 0,000 | 0,000 | 1,000 | 0,000 | 0,966 | 0,690 | 0,552 | 0,968 | 0,000 | 0,000 | 1,000 | 0,000 | 0,968 | 0,706 | 0,558 |
| | M | 0,981 | 0,792 | 0,826 | 0,989 | 0,798 | 0,981 | 0,991 | 0,923 | 0,949 | 0,493 | 0,739 | 0,96 | 0,566 | 0,949 | 0,940 | 0,818 |
| | O | 0,637 | 0,341 | 0,328 | 0,756 | 0,085 | 0,637 | 0,556 | 0,426 | 0,719 | 0,516 | 0,125 | 0,954 | 0,102 | 0,719 | 0,621 | 0,481 |
| ResNet50 | N | 0,669 | 0,465 | 0,408 | 0,787 | 0,202 | 0,669 | 0,682 | 0,518 | 0,681 | 0,454 | 0,416 | 0,791 | 0,212 | 0,681 | 0,703 | 0,532 |
| | D | 0,701 | 0,707 | 0,184 | 0,962 | 0,178 | 0,701 | 0,695 | 0,525 | 0,687 | 0,656 | 0,185 | 0,949 | 0,162 | 0,687 | 0,643 | 0,497 |
| | G | 0,926 | 0,300 | 0,222 | 0,968 | 0,217 | 0,926 | 0,678 | 0,607 | 0,934 | 0,375 | 0,294 | 0,972 | 0,296 | 0,934 | 0,762 | 0,664 |
| | C | 0,981 | 0,857 | 0,828 | 0,991 | 0,832 | 0,981 | 0,988 | 0,933 | 0,971 | 0,797 | 0,785 | 0,985 | 0,775 | 0,971 | 0,964 | 0,903 |
| | A | 0,924 | 0,333 | 0,440 | 0,951 | 0,339 | 0,924 | 0,815 | 0,693 | 0,904 | 0,259 | 0,458 | 0,929 | 0,284 | 0,904 | 0,702 | 0,630 |
| | H | 0,938 | 0,118 | 0,125 | 0,967 | 0,089 | 0,938 | 0,745 | 0,591 | 0,916 | 0,167 | 0,400 | 0,933 | 0,199 | 0,916 | 0,764 | 0,626 |
| | M | 0,970 | 0,667 | 0,783 | 0,980 | 0,704 | 0,970 | 0,989 | 0,888 | 0,957 | 0,552 | 0,696 | 0,971 | 0,593 | 0,957 | 0,959 | 0,836 |
| | O | 0,675 | 0,394 | 0,313 | 0,815 | 0,136 | 0,675 | 0,608 | 0,473 | 0,694 | 0,410 | 0,19 | 0,892 | 0,098 | 0,694 | 0,575 | 0,456 |

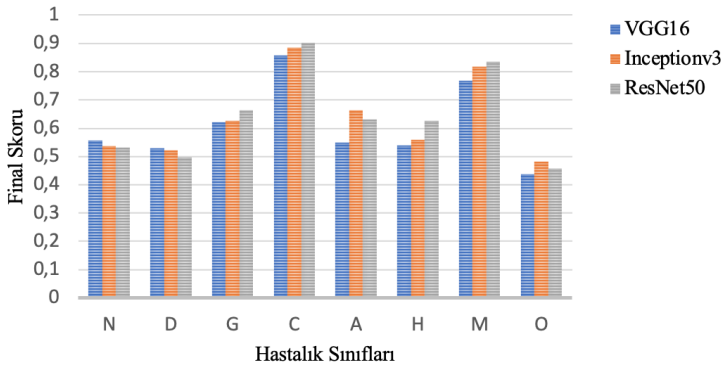
Acc: Doğruluk; Prec: Kesinlik; Sens: Duyarlılık; Spec: Özgüllük; $\kappa$: Kappa; AUC: Eğri altında kalan alan; Final: F1, AUC ve $\kappa$ değeri ortalaması

Doğrulama seti için oküler hastalık sınıflarına ait derin konvolüsyonel sinir ağı modelleri sınıflandırma performansı değerlendirme ölçütü olan Final skorları Şekil 4.33'de yer almaktadır. Normal, Glokom, Hipertansiyon ve diğer hastalıklar sınıf kategorileri için ResNet50 modeli, Diyabetik retinopati, Katarakt, Yaşa bağlı makula dejenerasyonu, Miyop sınıf kategorileri için Inceptionv3 modeli en yüksek final skor değerine ulaştığı görülmektedir.




**Şekil 4.33.** Doğrulama seti oküler hastalık sınıflarına ait derin konvolüsyonel sinir ağı modellerinin sınıflandırma performansı Final skorları

Test seti için oküler hastalık sınıflarına ait derin konvolüsyonel sinir ağı modelleri sınıflandırma performansı değerlendirme ölçütü olan Final skorları Şekil 4.34'de yer almaktadır. Normal ve Diyabetik retinopati sınıf kategorileri için VGG16 modeli, Glokom, Katarakt, Hipertansiyon ve Miyop sınıf kategorileri için ResNet50 modeli, Yaşa bağlı makula dejenerasyonu, diğer hastalıklar sınıf kategorileri için Inceptionv3 modeli en yüksek final skor değerine ulaştığı görülmektedir.




**Şekil 4.34.** Test seti oküler hastalık sınıflarına ait derin konvolüsyonel sinir ağı modellerinin sınıflandırma performansı Final skorları

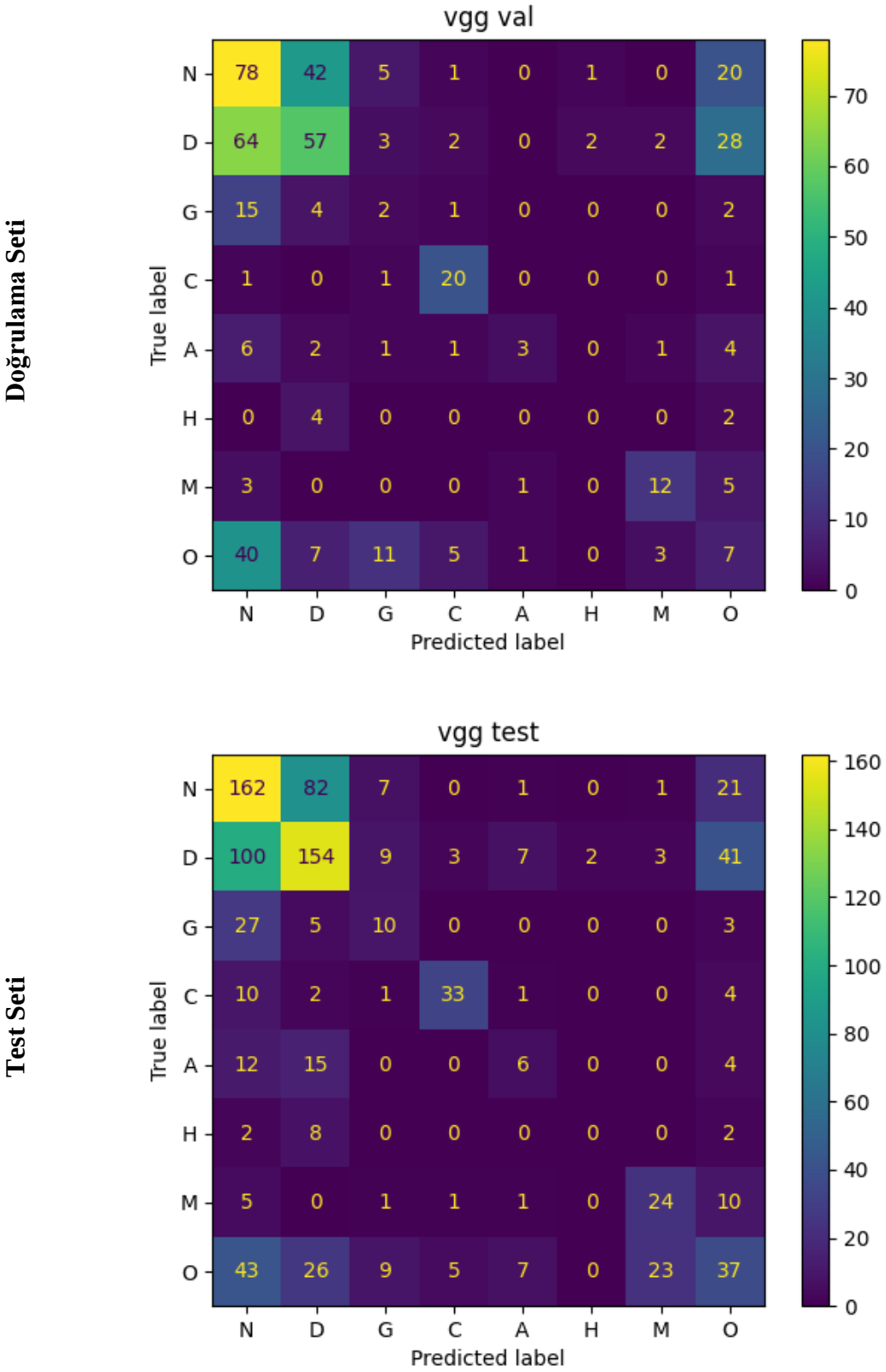
## 4.9. Derin Konvolüsyonel Sinir Ağı Mimari Modellerine ait Genel Sınıflandırma Performans Bulguları

İkili sınıflandırma sonrası doğrulama ve test seti için sekiz farklı hastalık sınıfına ait karmaşıklık matrisleri VGG16, Inceptionv3 ve ResNet50 modelleri için sırası ile Şekil 4.35, Şekil 4.36 ve Şekil 4.37'de yer almaktadır.

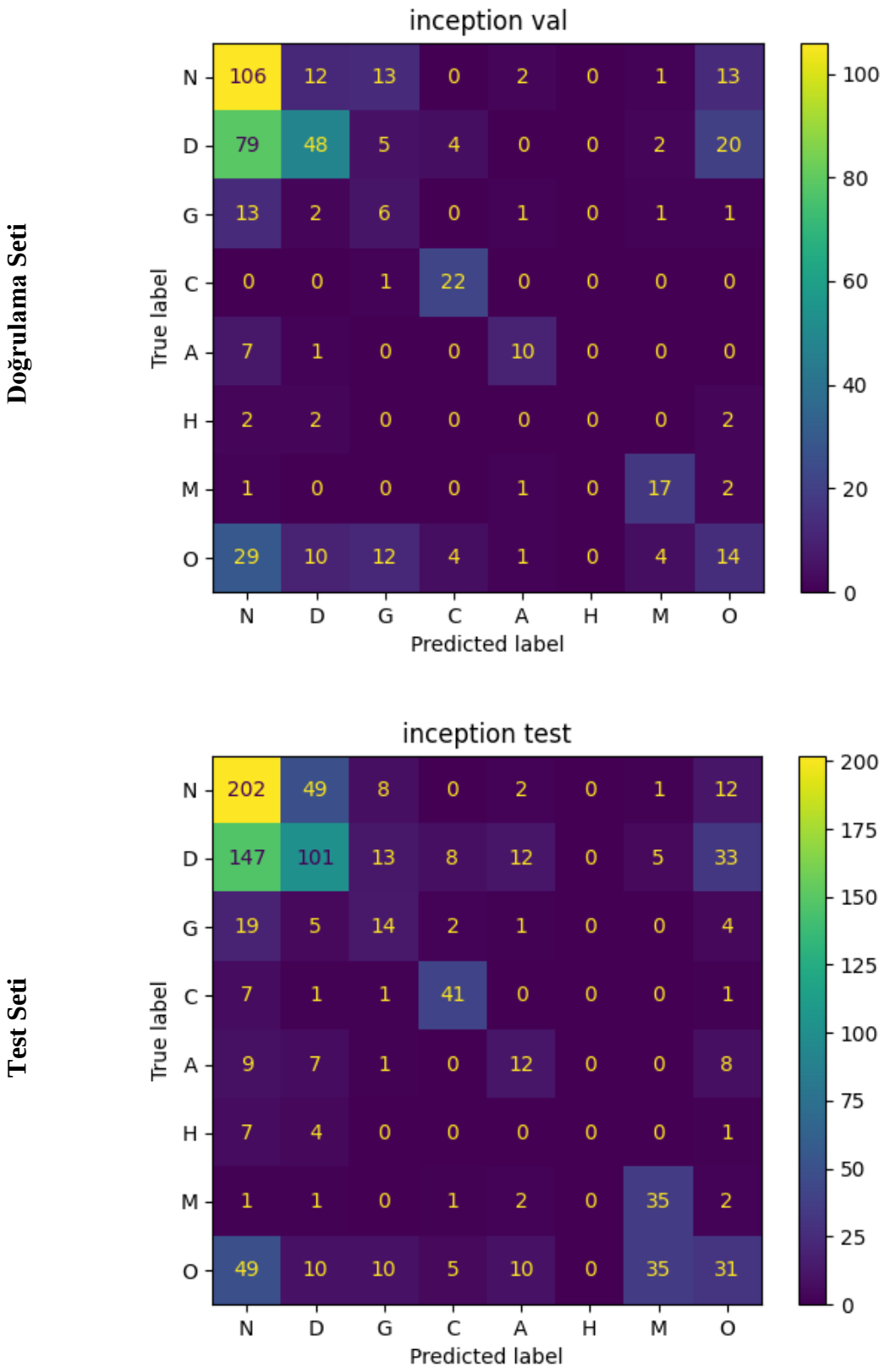
Doğrulama ve test seti için modellerin genel sınıflandırma başarı ölçütleri hesaplanarak Tablo 4.18'de sunulmuştur.

ODIR doğrulama setinde modellerin doğruluk, kesinlik, duyarlılık ve özgüllük değerleri sırası ile, VGG16 modeli için 0.815, 0.390, 0.446 ve 0.879; Inceptionv3 modeli için 0.834, 0.444, 0.507 ve 0.890; ResNet50 modeli için 0.848, 0.479, 0.344 ve 0.935 olarak hesaplanmıştır. Final skorunun belirlenmesinde önemli rol oynayan kappa, F1 ve AUC değerleri ve ortalaması alınarak elde edilen Final skoru ise sırası ile VGG16 modeli için 0.307, 0.815, 0.826 ve 0.649; Inceptionv3 modeli için 0.375, 0.834, 0.829 ve 0.679; ResNet50 modeli için 0.316, 0.848, 0.785 ve 0.649 olarak hesaplanmıştır.

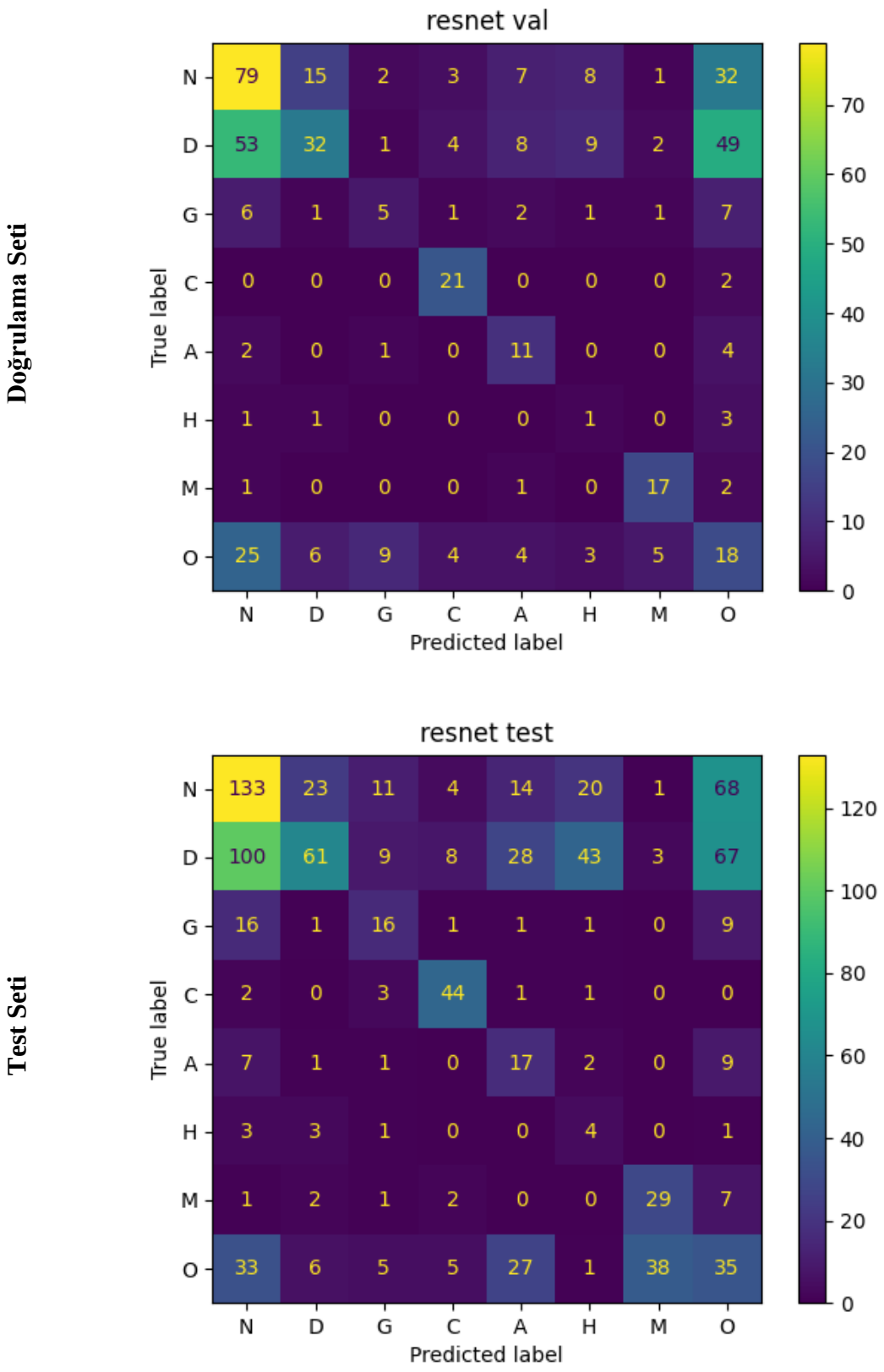
ODIR test setinde modellerin doğruluk, kesinlik, duyarlılık ve özgüllük değerleri sırası ile, VGG16 modeli için 0.831, 0.437, 0.522 ve 0.884; Inceptionv3 modeli için 0.851, 0.494, 0.334 ve 0.941; ResNet50 modeli için 0.843, 0.454, 0.324 ve 0.933 olarak hesaplanmıştır. Final skorunun belirlenmesinde önemli rol oynayan kappa, F1 ve AUC değerleri ve ortalaması alınarak elde edilen Final skoru ise sırası ile VGG16 modeli için 0.376, 0.831, 0.824 ve 0.677; Inceptionv3 modeli için 0.317, 0.851, 0.837 ve 0.669; ResNet50 modeli için 0.291, 0.843, 0.751 ve 0.628 olarak hesaplanmıştır. Bu üç farklı derin konvolüsyonel sinir ağı mimarisinden en yüksek Final skoru VGG16 modeli ile elde edilmiştir.




**Şekil 4.35.** Doğrulama ve test seti için VGG16 Modeli Sekiz Farklı Sınıf Kategorisine ait Karmaşıklık Matrisi




**Şekil 4.36.** Doğrulama ve test seti için Inceptionv3 Modeli Sekiz Farklı Sınıf Kategorisine ait Karmaşıklık Matrisi




**Şekil 4.37.** Doğrulama ve test seti için ResNet50 Modeli Sekiz Farklı Sınıf Kategorisine ait Karmaşıklık Matrisi

**Tablo 4.18.** Derin Konvolüsyonel Sinir Ağı Modellerine ait Genel Sınıflandırma Performanslarının Değerlendirilmesi

| | Performans ölçütleri | VGG16 | Inceptionv3 | ResNet50 |
|---|---|---|---|---|
| Doğrulama Seti | Accuracy | 0,815 | 0,834 | 0,848 |
| | Precision | 0,390 | 0,444 | 0,479 |
| | Sens | 0,446 | 0,507 | 0,344 |
| | Spec | 0,879 | 0,890 | 0,935 |
| | $\kappa$ | 0,307 | 0,375 | 0,316 |
| | F1 | 0,815 | 0,834 | 0,848 |
| | AUC | 0,826 | 0,829 | 0,785 |
| | Final | 0,649 | 0,679 | 0,649 |
| Test Seti | Accuracy | 0,831 | 0,851 | 0,843 |
| | Precision | 0,437 | 0,494 | 0,454 |
| | Sens | 0,522 | 0,334 | 0,324 |
| | Spec | 0,884 | 0,941 | 0,933 |
| | $\kappa$ | 0,376 | 0,317 | 0,291 |
| | F1 | 0,831 | 0,851 | 0,843 |
| | AUC | 0,824 | 0,837 | 0,751 |
| | Final | 0,677 | 0,669 | 0,628 |

Accuracy: Doğruluk; Precision: Kesinlik; Sens: Duyarlılık; Spec: Özgüllük; $\kappa$: Kappa; AUC: Eğri altında kalan alan; Final: F1, AUC ve $\kappa$ değeri ortalaması

# 5. TARTIŞMA

Görüntü tanıma problemlerinde ilerlemeyi sağlayan en önemli gelişme, 2009'da yayınlanan ve derin konvolüsyonel sinir ağı model mimarilerinin görüntüleri en düşük hata oranıyla sınıflandırabilmesi için her yıl düzenlenen bir yarışmaya dönüşen ImageNet veri tabanıdır (Deng vd., 2009). Derin öğrenme temelli görüntü sınıflandırma yöntemleri ile hem düşük düzey hem de yüksek düzey görüntü özellikleri öğrenilebilmektedir. Derin öğrenme teknikleriyle öğrenilen görüntü özelliklerinin uzman seçimli özelliklerden daha fazla temsil edici olduğu görülmektedir (Xiao vd., 2020). Özellik seçimi ve çıkarımı için en yaygın kullanılan derin öğrenme yöntemi konvolüsyonel sinir ağlarıdır. Konvolüsyonel sinir ağları kavramsal olarak otomatik özellik çıkarımı ve sınıflandırma olmak üzere iki gruba ayrılmaktadır. (Salvaris ve ark., 2018).

Derin konvolüsyonel sinir ağı mimarisi modelinin eğitimi ve parametrelerin öğrenme aşamasında büyük miktarda veri ve önemli hesaplama gücü gerektirmesi en önemli problemlerin başında gelmektedir. Bu nedenle CNN modellerinde sıklıkla transfer öğrenme yöntemi kullanılmaktadır. Sıfırdan öğrenmek, özellikle yüksek boyutlu medikal görüntüleme veri kümelerinde, hesaplama maliyeti, yakınsama problemi ve yeterli sayıda yüksek kaliteli etiketli örnek olmaması nedeniyle pratik bir yöntem değildir. Bilgi işlem kaynaklarına bağlı olarak, derin bir modelin eğitimi aylar sürebiliyorken, bu modeli hedef uygulamalar için transfer öğrenme yöntemiyle aktarmak çalışma zamanını önemli ölçüde azaltmaktadır. Transfer öğrenme, model eğitimi için çok sayıda örneğin gerekliliğini ortadan kaldırmış, medikal görüntüleme alanında yardımcı haline gelmiştir (Shin, 2016).

Bilgisayarlı görü görevlerinde dikkat çeken başarıya sahip CNN mimarileri, son zamanlarda oküler hastalıkların sınıflandırılması için oldukça etkili bir yöntem haline gelmiştir (Gour & Khanna, 2020; Ram & Aldasoro, 2020; Luo & Shen, 2020). Fundus görüntülerini kullanarak hastalıkların sınıflandırılması karmaşık olduğundan, bu görev için derin konvolüsyonel sinir ağı mimarileri ve transfer öğrenme yöntemi tercih edilmektedir.

Literatürde ODIR-2019 veri seti kullanılarak farklı derin konvolüsyonel sinir ağı mimarisi modellerinin kurulduğu ve en iyi doğruluğa sahip modelin önerildiği birkaç çalışma yer almaktadır. Corbilla tez çalışmasında, farklı veri ön işleme stratejileri uygulamıştır. Sol göze ait fundus görüntülerinin simetriğini alarak her bir sol göze ait fundus görüntüsünü sağ fundus görüntüsüne dönüştürerek eğitim kümesi için 7000 fundus görüntüsü elde etmiştir. Her bir hastalık sınıfı kategorisine ait dağılımın eşit olacağı şekilde veri artırımı gerçekleştirmiştir. Fundus görüntülerinin yükseklik ve genişliklerini 224x224 piksel olarak belirlemiştir. ImageNet ön-eğitimli ağırlıklar kullanarak transfer öğrenme yöntemi ve Stokastik Gradyan İniş optimizasyon algoritması ile Inceptionv3 ve VGG16 derin konvolüsyonel sinir ağı mimarisi modellerini oluşturmuştur. Hiperparametre konfigürasyonu için öğrenme oranı 0.001, momentum 0.90, erken durdurma kriteri 8 adım olarak belirlemiştir. Model eğitimini gerçekleştirirken ODIR-2019 veri setine ait doğrulama seti için sınıflandırma etiketleri paylaşılmadığı için doğrulama seti olarak eğitim setinden her sınıfta 50 fundus görüntüsü olmak üzere, toplamda 400 fundus görüntüsünü doğrulama seti olarak ayırmıştır. ODIR-2019 doğrulama setini de test seti olarak değerlendirmiştir. Çalışma sonucunda, Inceptionv3 derin konvolüsyonel sinir ağı mimarisinin sırası ile doğrulama veri seti için doğruluk, kesinlik, AUC ve kappa değerlendirme ölçütlerini 0.8984; 0.6021; 0.855; 0.5186 olarak elde etmiştir, F1, AUC ve Final skorunu ise sırası ile 0.8984; 0.8838; 0.7669 olarak hesaplamıştır. Inceptionv3 modelinin VGG16'dan daha yüksek performans gösterdiğini belirtmiştir (Corbilla, 2019).

Wang ve arkadaşları, çalışmalarında veri setini %90 eğitim, %10 doğrulama seti olarak ayırmıştır. Girdi görüntü boyutlarını 299x299 olmak üzere düzenlemişler. Görüntüleri 45, 90 derece rotasyon olmak üzere rasgele döndürme yöntemi ile veri artırımı işlemi gerçekleştirmişlerdir. Ayrıca tek kanallı ve üç kanallı olmak üzere görüntüleri iki farklı şekilde farklı ön-eğitimli modeller ile transfer öğrenme yöntemi kullanarak eğitim işlemini gerçekleştirmişlerdir. VGG16, VGG19, Xception, Inceptionv3, EfficientNet B3, ResNet50, MobileNet, InceptionResNetV2, DenseNet derin konvolüsyonel sinir ağı modellerini oluşturmuşlardır. EfficientNet B3 modelinin diğer modellerle karşılaştırıldığında en iyi performans ölçütlerine sahip olduğunu belirtmişlerdir. Bu model ile AUC, Kappa, F1 ve Final skorunu sırası ile 0.67, 0.43, 0.85 ve 0.65 olarak hesaplamışlardır (Wang vd., 2020).

Wang ve ark, ayrıca, girdi görüntü yükseklik ve genişlik büyüklüklerinin de model performansını etkileyebileceğini düşünerek girdi boyutunu 448x448 olarak güncellediklerinde, EfficientNet B3 modeli ile Xception modellerinin sınıflandırma performansının birbirine benzer ve diğer modellere göre daha başarılı sonuçlar elde ettiğini göstermişlerdir (Wang vd., 2020).

Gour ve Khanna, fundus görüntülerinin sekiz oküler hastalık kategorisine sınıflandırılması için transfer öğrenmeyi kullanan iki CNN tabanlı model önermişlerdir. Önerilen ilk modelde, eğitim setinin sol ve sağ fundus görüntüleri, girdi olarak önceden eğitilmiş CNN mimarilerine ayrı ayrı uygulanmıştır. İlk (girdi) katmanı kaldırılmıştır ve son beş katmanın ağırlığı, transfer öğrenimi için dondurulmuştur. Tam bağlantı katmanlarını kaldırmıştır. Bu paralel oluşturulan CNN mimarilerinden elde edilen iki özellik haritası elde edildikten sonra ortalama havuzlama katmanı ile birleştirilmiş ve sigmoid aktivasyon fonksiyonu kullanarak sınıflandırma işlemi gerçekleştirilmiştir. İkinci modelde ise sol ve sağ fundus görüntüleri birleştirilerek ön eğitimli ağ modellerine girdi olarak verilmiştir. ResNet, Inceptionv3, MobileNet ve VGG16 modelleri Adam ve Stokastik Gradyan İniş algoritmaları ile model sınıflandırma performans metrikleri değerlendirilmiştir. İki girdili önerilen ilk derin konvolüsyonel sinir ağı modelinin kullanılarak oluşturulduğu VGG16 modelinin InceptionV3, ResNet ve MobileNet CNN mimarileri ile karşılaştırıldığında en yüksek performans ölçütüne ulaştığı belirtilmiştir. SGD optimizasyon algoritması ile iki girdili yaklaşım için VGG16 ile elde edilen AUC ve F1 skorlarını sırasıyla 84,93 ve 85,57 olarak hesaplamışlardır (Gour & Khanna, 2020).

Luo ve Shen, retina hastalığının tespiti için damar farkındalığına sahip yeni bir mimari önermişlerdir. Önerilen VesselNet mimarisi, küresel fundus görüntüsünü, damarları ve damarla ilgili bölgeyi bütünleştiren üç aşamadan elde edilmiştir. ODIR görüntü kümesinde yaşa bağlı makula dejenerasyonu sınıfı için önerilen yöntem için performans ölçütü değerlerinin diğer CNN mimarilerinden daha iyi performans gösterdiğini ve damarla ilgili bölgelerin önemini vurgulamıştır (Luo & Shen, 2020).

İncelenen tüm modeller derin mimariye sahip olmalarından dolayı görüntü sınıflandırılmasında oldukça başarılı modellerdir. Araştırıcının deneyimleyerek veri setine ve problemin niteliğine göre belirlenen ve ayarlanan hiperparametreler, algoritma farklılıkları, eğitim-test veri setinin her çalışma için rastgele ve birbirinden bağımsız bir şekilde ayrılması sonucunda, elde edilen sonuçların farklılık gösterebileceği söylenebilir. Farklı sonuçlara rağmen CNN modellerinin derin mimarilerinden dolayı, sınıflandırma problemlerinde oldukça iyi performans gösterdikleri açıkça görülmektedir.

Bu tez çalışması ile elde edilen sonuçlar değerlendirildiğinde ODIR fundus görüntü setinde oküler hastalıkların sınıflandırılmasında, VGG16 modeli Final skoru 0.677, Inceptionv3 modeli Final skoru 0.669 ve ResNet50 modeli Final skoru 0.628 olarak bulunmuştur. Final skorları göz önüne alındığında en başarılı modelin VGG16 olduğu belirlenmiştir. Sonuçta, literatür çalışmaları ve bu çalışmada gerçekleştirilen tekrarlı analizlerle edinilen tecrübeye dayanarak VGG16 modelinin %83,1 doğruluk ve en yüksek Final skoru ile ODIR veri seti için başarılı bir model olduğu, Inceptionv3 ve ResNet50 modellerinin de bu çalışma için %85,1 ve %84,3 ile yüksek doğruluk değerlerine sahip olduğu tespit edilmiştir.

# 6. SONUÇ VE ÖNERİLER

Medikal görüntüler hastalıkların erken tanısında hayati öneme sahiptir. Oftalmoloji alanında medikal görüntüleme alanı olan fundus görüntüleme, oftalmolojik hastalıkların kesin tanısında kullanılmaktadır. Bu tez çalışmasında oküler hastalıkların teşhisinin doğru bir şekilde saptanması amacı ile fundus görüntüleri sınıflandırılmıştır. Oküler hastalıkların sınıflandırılmasında oftalmologların gerçek klinik ortama uygun gerçekleştirdiği her bir hastanın iki göz görüntüsünden elde edilen fundus görüntüleme ile çoklu hastalık tanılarının konduğu ve literatürde ilk kez çok sınıflı binoküler fundus görüntülerini oluşturulduğu ODIR görüntü kümesi kullanılmıştır. 5000 hastadan 10000 görüntüden oluşan veri seti eğitim, doğrulama ve test seti olmak üzere sırası ile 3500, 500 ve 1000 hastaya ait sol ve sağ fundus görüntüleri içermektedir. Eğitim seti ile test setinin oranı 7:3 olarak belirlenmiştir. Derin konvolüsyonel sinir ağları modelini eğitmek için eğitim seti, model seçimi için doğrulama seti, modellerin genelleştirilme yeteneği ise, test seti ile değerlendirilmiştir.

Günümüzde sayısal veri üretebilen makinelerin ve internet kullanımının artması, çok büyük verilerin kaydedilebilmesini sağlamıştır. Bu büyük verileri işlemek için yetersiz kalan CPU işlemciler yerine daha büyük hesaplama kapasitelerine sahip GPU işlemcili makineler kullanılmaktadır. Bununla birlikte karmaşık işlemler yapan derin öğrenme yöntemleri, büyük verileri sınıflandırma ve tahmin etme işlemlerinde en çok tercih edilen yöntemlerdir. Bu tez çalışmasında, Google'ın geliştiricilere sunduğu bulut servisi olan Google Colaboratory (Colab-Pro) platformunda, NVIDIA GPU (Grafik İşleme Birimi) işlemcisine sahip Tesla P100 işlemcili makine kullanılmıştır. NVIDIA Tesla-P100 GPU hızlandırıcı kullanılarak yapılan çalışma için yaklaşık 25 GB (Giga Byte) rastgele erişilebilir bellek (Random Access Memory – RAM) ve model eğitimlerinin ve sonuçlarının kaydedilebilmesi için 100 GB depolama alanı kullanılmıştır.

Konvolüsyon katmanlarındaki işlemler ile görüntü boyutları küçültülmekte ve konvolüsyon işlemleri tamamlandıktan sonra elde edilen yeni boyutlu görüntü verileri düzleştirilerek tam bağlı yapay sinir ağına sunulmaktadır. Konvolüsyon katmanlarında, görüntüler yeniden boyutlandırılırken filtre boyutu, filtre sayısı, kaydırma adımı, konvolüsyon katman sayısı ve ortaklama katman sayısı hiper-parametreleri kullanılmıştır.

Bu tez çalışması kapsamında, literatürde sınıflandırma performansları ve başarıları ile dikkat çeken Inceptionv3, VGG 16 ve ResNet50 derin konvolüsyonel sinir ağları mimari modelleri kullanılmıştır. Ön eğitimli ImageNet ağırlıkları kullanılarak sağ ve sol fundus görüntüleri ile üç farklı derin konvolüsyonel sinir ağları mimari modellerinin eğitim işlemi gerçekleştirilmiştir. Test görüntü seti üzerinde, oluşturulan modellerin belirlenen başarı ölçütlerine göre sınıflama performansları karşılaştırılmıştır.

Elde edilen yüksek doğruluk, AUC, F1 skoru, kappa değeri ve son üç ölçütün ortalaması olan en yüksek Final skor değerine sahip CNN modeli olan VGG16 modelinin fundus görüntülerinin sınıflandırılmasında kullanılabileceğini; Tıp Fakültelerinin Oftalmoloji alanında uzmanlara tanı aşamasında yardımcı bir destek rolü üstlenebileceği ve bu alanda gelecekte kullanılabilir sistemler tasarlanabileceğini göstermektedir.

Giderek daha fazla halka açık veri setine daha kolay ulaşılabilme imkânı olduğundan, gelecek çalışmalarda, Tıp Fakültesinin medikal görüntüleme yapan diğer farklı uzmanlık alanlarında da derin sinir ağları model eğitiminin artacağı ve kullanılabileceği tahmin edilmektedir. Ancak, günümüzde derin öğrenmenin klinik ortamda pratik bir şekilde uygulanması hala büyük bir sorundur. Özellikle, ODIR-2019 veri seti için, "O" (diğer hastalıklar/anomaliler) sınıfı çeşitli yaygın olmayan nadir görülen fundus hastalıklarını içermektedir. Bu gibi hastalıklar için veri miktarı oldukça sınırlıdır. Oftalmologlar bu sınıflara sahip hastalarla sıklıkla karşılaşmamaktadır. Bu durum, bir ağın performansını iyileştirmeyi oldukça zorlaştırmaktadır. Diğer bir temel sınırlama ise derin ağların doğasının kara kutusundan gelmektedir. Ağ, özellikleri görüntülerden otomatik olarak öğrenir. Ancak öğrenilen belirli özellikler bilinmemektedir. Sık görülmeyen hastalıkların sınıflandırma doğruluğunu ve genellemeyi geliştirmeye yardımcı olması için farklı kameralar aracılığıyla farklı hastanelerden on binlerce anomali ve nadir karşılaşılan vakayı içerecek büyük eğitim veri kümeleri elde edilerek literatüre katkı sağlanması ile Derin Konvolüsyonel Sinir Ağı Model Mimarilerinin daha farklı özellikler öğrenmesi sağlanabilir.

# KAYNAKLAR DİZİNİ


Abràmoff, M. D., Lou, Y., Erginay, A., Clarida, W., Amelon, R., Folk, J. C., & Niemeijer, M. (2016). Improved automated detection of diabetic retinopathy on a publicly available dataset through integration of deep learning. *Investigative ophthalmology & visual science*, 57(13), 5200-5206.

Aktan, E. (2018). *Büyük veri: Uygulama alanları, analitiği ve güvenlik boyutu*. Bilgi Yönetimi, 1(1), 1-22.

Alpaydın, Ethem. 2011. *Yapay öğrenme*. Boğaziçi Üniversitesi Yayınevi.

Amidi, Afshine A & Amidi, S. (2021) Vip cheatsheet: Convolutional Neural Networks. https://github.com/afshinea/stanford-cs-230-deep-learning (Nisan, 2021).

Amini, Alexander (2021). MIT 6S191.Introduction to Deep Learning. https://introtodeeplearning.com (Ocak, 2021).

Aptel, F., Denis, P., Rouberol, F., & Thivolet, C. (2008). Screening of diabetic retinopathy: effect of field number and mydriasis on sensitivity and specificity of digital fundus photography. *Diabetes & metabolism*, 34(3), 290-293.

Arf, C. (1959). Makine Düşünebilir Mi ve Nasıl Düşünebilir? Atatürk Üniversitesi, 1958-1959, Halk Konferansları I., *Erzurum: Atatürk Üniversitesi, Üniversite Çalışmalarını Muhite Yayma ve Halk Eğitimi Yayınları Konferanslar Serisi*, No:1.

Asaoka, R., Murata, H., Iwase, A., & Araie, M. (2016). Detecting preperimetric glaucoma with standard automated perimetry using a deep learning classifier. *Ophthalmology*, 123(9), 1974-1980.

Bengio, Y., Boulanger-Lewandowski, N., & Pascanu, R. (2013). Advances in optimizing recurrent networks. *IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 8624-8628). IEEE.

Botev, A., Lever, G., & Barber, D. (2017). Nesterov's accelerated gradient and momentum as approximations to regularised update descent. *International Joint Conference on Neural Networks* (IJCNN) (pp. 1899-1903). IEEE.

Bourne, R. R. (2021). Causes of blindness and vision impairment in 2020 and trends over 30 years, and prevalence of avoidable blindness in relation to VISION 2020: the Right to Sight: an analysis for the Global Burden of Disease Study. *The Lancet Global Health*, (2), e144-e160.

Brownlee, J. (2018). Basics of Linear Algebra for Machine Learning.

Brownlee, J. What is the Difference Between a Parameter and a Hyperparameter?, https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/ (Nisan, 2021).

Burlina, P., Freund, D. E., Joshi, N., Wolfson, Y., & Bressler, N. M. (2016, April). Detection of age-related macular degeneration via deep learning. *IEEE 13th International Symposium on Biomedical Imaging* (ISBI) (pp. 184-188).

## KAYNAKLAR DİZİNİ (Devam Ediyor)


Campbell, M., Hoane Jr, A. J., & Hsu, F. H. (2002). *Deep blue*. Artificial intelligence, 134(1-2), 57-83.

Chelaramani, S., Gupta, M., Agarwal, V., Gupta, P., & Habash, R. (2019, November). Multi-task Learning for Fine-Grained Eye Disease Prediction. *In Asian Conference on Pattern Recognition* (pp. 734-749). Springer, Cham.

Chollet, F. Keras Usage of initializers, https://keras.io/initializers/ (Nisan, 2021).

Chollet, F. 2017. *Deep learning with Python*. Manning Publications Company.

Coll Corbilla, J. (2019). Reconeixement intel· ligent de malalties oculars mitjançant arquitectures d'aprenentatge profund.

Costagliola, C., Dell'Omo, R., Romano, M. R., Rinaldi, M., Zeppa, L., & Parmeggiani, F.(2009). Pharmacotherapy of intraocular pressure: part I. Parasympathomimetic, sympathomimetic and sympatholytics. *Expert opinion on Pharmacotherapy*, 10(16), 2663-2677.

Çarkacı, N. (2018). Derin Öğrenme Uygulamalarında En Sık kullanılan Hiperparametreler. https://medium.com/deep-learning-turkiye/derin-ogrenme-uygulamalarinda-en-sik-kullanilan-hiper-parametreler-ece8e9125c4 (10.04.2020).

Çırak, B. ve Yörük, A. (2016). Mekatronik biliminin öncüsü İsmail El-Cezeri. Siirt Üniversitesi, *Sosyal Bilimler Enstitüsü Dergisi*, (4), 175-194.

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. *IEEE conference on computer vision and pattern recognition* (pp. 248-255). Ieee.

Dumoulin, V., & Visin, F. (2016). A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285.

Elangovan, Poonguzhali, ve Malaya Kumar Nath. 2020. "Glaucoma assessment from color fundus images using convolutional neural network". *International Journal of Imaging Systems and Technology* (March):1–17. doi: 10.1002/ima.22494.

Esmer, G. B. , (2019). *Yapay Zeka ve Girişimsel İşlemler*. Sağlık Bilimlerinde Yapay Zeka (pp.113-124), İstanbul: Çağlayan Kitapevi ve Eğitim Çözümleri Ticaret A.Ş.

Fausett, L. V. (2006). *Fundamentals of neural networks: architectures, algorithms and applications*. Pearson Education India.

Friedman, L (2020). Deep Learning: State of the Art. https://lexfridman.com/files/slides/2020_01_06_deep_learning_state_of_the_art.pdf (Ocak, 2020).

Gadkari, Salil S., Quresh B. Maskati, ve Barun Kumar Nayak. 2016. "Prevalence of diabetic retinopathy in India: The All India Ophthalmological Society Diabetic Retinopathy Eye Screening Study 2014". *Indian Journal of Ophthalmology* 64(1):38–44. doi: 10.4103/0301-4738.178144.

## KAYNAKLAR DİZİNİ (Devam Ediyor)

García-Floriano, A., Ferreira-Santiago, Á., Camacho-Nieto, O., & Yáñez-Márquez, C. (2019). A machine learning approach to medical image classification: Detecting age-related macular degeneration in fundus images. *Computers & Electrical Engineering*, 75, 218-229.

Glorot, X., & Bengio, Y. (2010, March). Understanding the difficulty of training deep feedforward neural networks. *In Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249-256). JMLR Workshop and Conference Proceedings.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial networks. arXiv preprint arXiv:1406.2661.

Goodfellow, Ian, Yoshua Bengio, ve Aaron Courville. 2016. "*Deep Learning*".

Gour, N., ve Khanna, P. (2020). Multi-class multi-label ophthalmological disease detection using transfer learning based convolutional neural network. *Biomedical Signal Processing and Control*, 102329.

Grassmann, F., Mengelkamp, J., Brandl, C., Harsch, S., Zimmermann, M. E., Linkohr, B., ... & Weber, B. H. (2018). A deep learning algorithm for prediction of age-related eye disease study severity scale for age-related macular degeneration from color fundus photography. *Ophthalmology*, 125(9), 1410-1420.

Grewal, P. S., Oloumi, F., Rubin, U., & Tennant, M. T. (2018). Deep learning in ophthalmology: a review. *Canadian Journal of Ophthalmology*, 53(4), 309-313.

Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., ... & Webster, D. R. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama*, 316(22), 2402-2410.

Gustafson, K., & Sartoris, G. (1998). Assigning initial weights in feedforward neural networks. *IFAC Proceedings Volumes*, 31(20), 1053-1058.

Haenlein, M., & Kaplan, A. (2019). A brief history of artificial intelligence: On the past, present, and future of artificial intelligence. *California management review*, 61(4), 5-14.

Hamet, P., & Tremblay, J. (2017). Artificial intelligence in medicine. *Metabolism*, 69, S36-S40.

Han, S. H., Kim, K. W., Kim, S., & Youn, Y. C. (2018). Artificial neural network: understanding the basic concepts without mathematics. *Dementia and neurocognitive disorders*, 17(3), 83

Haykin, S. S. (2009). Neural networks and learning machines/Simon Haykin.

He, J., Li, C., Ye, J., Qiao, Y., & Gu, L. (2021). Multi-label ocular disease classification with a dense correlation deep neural network. *Biomedical Signal Processing and Control*, 63, 102167.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

## KAYNAKLAR DİZİNİ (Devam Ediyor)


Hijazi, S.L., Kumar, R.R., & Rowen, C. (2015). Using Convolutional Neural Networks for Image Recognition By. https://ip.cadence.com/uploads/901/cnn_wp-pdf.

Hinton, G. E. (1986). Learning distributed representations of concepts. *In Proceedings of the eighth annual conference of the cognitive science society* (Vol. 1, p. 12).

Hinton, G., Bengio, Y., Yann LeCun (2019). *Computing Machinery*, Communications of the ACM.

Hoover, A. (1975). STARE database. Available: Available: http://www. ces. clemson. edu/~ ahoover/stare.

İnik. Ö., & Ülker, E. (2017). Derin öğrenme ve görüntü analizinde kullanılan derin öğrenme modelleri. *Gaziosmanpaşa Bilimsel Araştırma Dergisi*, 6(3), 85-104.

Karaküçük, Y., & Eker, S. (2020). *Sağlık Bilimlerinde Yapay Zeka, Oftalmolojide Yapay Zeka ve Derin Öğrenme Uygulamaları*, Bölüm 9. Editör: Şahin, Doğan, Sivri. Özyurt Matbaacılık.

Kızrak, M. A., & Bolat, B. (2018). Derin öğrenme ile kalabalık analizi üzerine detaylı bir araştırma. *Bilişim Teknolojileri Dergisi,* 11(3), 263-286.

Kızrak, M. A. (2020). Keras ile Derin Öğrenmeye Giriş, Bilgi Teknolojileri Kurumu, BTK Akademi.

Kolecki, J. C. (2002). An introduction to tensors for students of physics and engineering. *National Aeronautics and Space Administration*, Glenn Research Center.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 1097-1105.

Kurt, F. (2018). Evrişimli Sinir Ağlarında Hiper Parametrelerin Etkisinin İncelenmesi.

LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K. R. (2012). *Efficient backprop. In Neural networks: Tricks of the trade* (pp. 9-48). Springer, Berlin, Heidelberg.

LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10), 1995.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., & Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. *In Advances in neural information processing systems* (pp. 396-404).

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

Lee, C. S., Tyring, A. J., Deruyter, N. P., Wu, Y., Rokem, A., & Lee, A. Y. (2017). Deep-learning based, automated segmentation of macular edema in optical coherence tomography. *Biomedical optics express*, 8(7), 3440-3448.

Li, F., Chen, H., Liu, Z., Zhang, X., & Wu, Z. (2019). Fully automated detection of retinal disorders by image-based deep learning. *Graefe's Archive for Clinical and Experimental Ophthalmology*, 257(3), 495-505.

## KAYNAKLAR DİZİNİ (Devam Ediyor)


Li, N., Li, T., Hu, C., Wang, K., & Kang, H. (2021). A Benchmark of Ocular Disease Intelligent Recognition: One Shot for Multi-disease Detection. In Benchmarking, Measuring, and Optimizing: Third BenchCouncil International Symposium, Bench 2020, Virtual Event, November 15–16, 2020, Revised Selected Papers 3 (pp. 177-193). *Springer International Publishing*.

Li, Z., He, Y., Keel, S., Meng, W., Chang, R. T., & He, M. (2018). Efficacy of a deep learning system for detecting glaucomatous optic neuropathy based on color fundus photographs. *Ophthalmology*, 125(8), 1199-1206.

Luo, D., & Shen, L. (2020). Vessel-Net: A Vessel-Aware Ensemble Network For Retinopathy Screening From Fundus Image. *IEEE International Conference on Image Processing (ICIP)* (pp. 320-324). IEEE.

López, G., Quesada, L., & Guerrero, L. A. (2017, July). Alexa vs. Siri vs. Cortana vs. Google Assistant: a comparison of speech-based natural user interfaces. *In International Conference on Applied Human Factors and Ergonomics* (pp. 241-250). Springer, Cham.

Maji, Debapriya, Anirban Santara, Pabitra Mitra, ve Debdoot Sheet. 2016. "Ensemble of Deep Convolutional Neural Networks for Learning to Detect Retinal Vessels in Fundus Images".

Marsland, Stephen. 2015. *Machine learning: an algorithmic perspective*. CRC press.

Murugan P. & Durairaj S. (2017) "Regularization and Optimization strategies in Deep Convolutional Neural Network," ArXiv e-prints, https://ui.adsabs.harvard.edu/#abs/2017arXiv171204711M

Ng. A., Stanford University (2021). CS231n: Convolutional Neural Networks for Visual Recognition, https://github.com/cs231n/cs231n.github.io (Nisan, 2020).

Niemeijer, M., Staal, J., van Ginneken, B., Loog, M., & Abramoff, M. D. (2004, May). Comparative study of retinal vessel segmentation methods on a new publicly available database. *In Medical imaging 2004*: image processing (Vol. 5370, pp. 648-656). International Society for Optics and Photonics.

Nilsson, N. J. (2014). Principles of artificial intelligence. Morgan Kaufmann.

O'Leary, D. E. (2013). Artificial intelligence and big data. *IEEE intelligent systems*, 28(2), 96-99.

Peemen, M., Shi, R., Lal, S., Juurlink, B., Mesman, B., & Corporaal, H. (2016). The neuro vector engine: Flexibility to improve convolutional net efficiency for wearable vision. *Design, Automation & Test in Europe Conference & Exhibition (DATE)* (pp. 1604-1609). IEEE.

Peking university international competition on ocular disease intelligent recognition (ODIR-2019), 2020, (Erişim Tarihi: 28.04.2020). Erişim adresi: https://odir2019.grand-challenge.org/

Peng, Y., Dharssi, S., Chen, Q., Keenan, T. D., Agrón, E., Wong, W. T., ... & Lu, Z. (2019). DeepSeeNet: a deep learning model for automated classification of patient-based age-related macular degeneration severity from color fundus photographs. *Ophthalmology*, 126(4), 565-575.

## KAYNAKLAR DİZİNİ (Devam Ediyor)


Petrovska, B., Stojanovic, I., & Atanasova-Pacemska, T. (2018). Classification of Small Sets of Images with Pre-trained Neural Networks. *International Journal of Engineering and Manufacturing*, 8(4), 40.

Piette, L. 2018. Computer Vision with Convolutional Neural Networks, https://www.linkedin.com/pulse/computer-vision-convolutional-neural-networks-luke-piette. (Nisan 2021)

Polacco, A., & Backes, K. (2018). The amazon go concept: Implications, applications, and sustainability. *Journal of Business and Management*, 24(1), 79-92.

Poly, T. N., Islam, M. M., Yang, H. C., Nguyen, P. A., Wu, C. C., & Yu-Chuan (Jack) Li. (2019). Artificial Intelligence in Diabetic Retinopathy: Insights from a Meta-Analysis of Deep Learning. In *MedInfo* (pp. 1556-1557).

Poplin, R., Varadarajan, A. V., Blumer, K., Liu, Y., McConnell, M. V., Corrado, G. S., ... & Webster, D. R. (2018). Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning. *Nature Biomedical Engineering*, 2(3), 158-164.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1), 145-151.

Ram, A., ve Reyes-Aldasoro, D. C. C. (2020). The relationship between Fully Connected Layers and number of classes for the analysis of retinal images. arXiv preprint arXiv:2004.03624.

Raschka, S. (2015). *Python machine learning*. Packt Publishing Ltd.

Rashid, T. (2016). *Make your own neural network* (p. 222). CreateSpace Independent Publishing Platform.

Rosenzweig, J., & Bartl, M. (2015). A review and analysis of literature on autonomous driving. *E-Journal Making-of Innovation*, 1-57.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

Saine, P. J., & Tyler, M. E. (2002). Ophthalmic photography: retinal photography, angiography, and electronic imaging (Vol. 132). *Boston: Butterworth-Heinemann*.

Salvaris M, Dean D, Tok WH. (2018). *Deep learning with azure: building and deploying artificial intelligence solutions on the microsoft AI platform*. 1st Ed., Berkeley, Apress. 16-31.

Sang-Hun, C. (2017). Google's computer program beats Lee Sedol in Go tournament. New York Times.

Saranya, P., ve S. Prabakaran. (2020). "Automatic detection of non-proliferative diabetic retinopathy in retinal fundus images using convolution neural network". *Journal of Ambient Intelligence and Humanized Computing* (0123456789). doi: 10.1007/s12652-020-02518-6.

## KAYNAKLAR DİZİNİ (Devam Ediyor)


Saravanan, R., ve Pothula Sujatha. (2018). "Algorithms : A Perspective of Supervised Learning Approaches in Data Classification". *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)* (Iciccs):945–49.

Sathya, R., ve Annamma Abraham. 2013. "Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification". *International Journal of Advanced Research in Artificial Intelligence* 2(2):34–38. doi: 10.14569/ijarai.2013.020206.

Saxe, A. M., McClelland, J. L., & Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. arXiv preprint arXiv:1312.6120.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61, 85-117.

Sengupta, S., Singh, A., Leopold, H. A., Gulati, T., & Lakshminarayanan, V. (2020). Ophthalmic diagnosis using deep learning with fundus images–A critical review. *Artificial intelligence in medicine*, 102, 101758.

SenseCorp, Data Science Pyramid, https://sensecorp.com/data-science-pyramid/ (Nisan,2021)

Sewak, M., Karim, M. R., & Pujari, P. (2018). *Practical convolutional neural networks: implement advanced deep learning models using Python*. Packt Publishing Ltd.

Shin, H. C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., ... & Summers, R. M. (2016). Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5), 1285-1298.

Silahtaroğlu, G. (2008). *Veri madenciliği*. Papatya Yayınları, İstanbul.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Sogawa, T., Tabuchi, H., Nagasato, D., Masumoto, H., Ikuno, Y., Ohsugi, H., ... & Mitamura, Y. (2020). Accuracy of a deep convolutional neural network in the detection of myopic macular diseases using swept-source optical coherence tomography. *Plos one*, 15(4), e0227240.

Soni, D. (2018) Improving Vanilla Gradient Descent, https://towardsdatascience.com/improving-vanilla-gradient-descent f9d91031ab1d

Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Training very deep networks. arXiv preprint arXiv:1507.06228.

Subramanian, V. (2018). *Deep Learning with PyTorch: A practical approach to building neural network models using PyTorch*. Packt Publishing Ltd.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

## KAYNAKLAR DİZİNİ (Devam Ediyor)


Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

Tan, J. H., Bhandary, S. V., Sivaprasad, S., Hagiwara, Y., Bagchi, A., Raghavendra, U., ... & Acharya, U. R. (2018). Age-related macular degeneration detection using deep convolutional neural network. *Future Generation Computer Systems*, 87, 127-135.

Tan, M., & Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *In International Conference on Machine Learning* (pp. 6105-6114). PMLR.

Taylor, M. (2017). *Neural Networks: A Visual Introduction for Beginners*. Blue Windmill Media.

Ting, D. S. W., Cheung, C. Y. L., Lim, G., Tan, G. S. W., Quang, N. D., Gan, A., ... & Wong, T. Y. (2017). Development and validation of a deep learning system for diabetic retinopathy and related eye diseases using retinal images from multiethnic populations with diabetes. *Jama*, 318(22), 2211-2223.

Tufail, A., Rudisill, C., Egan, C., Kapetanakis, V. V., Salas-Vega, S., Owen, C. G., ... & Rudnicka, A. R. (2017). Automated diabetic retinopathy image assessment software: diagnostic accuracy and cost-effectiveness compared with human graders. *Ophthalmology*, 124(3), 343-351.

Turing, A. M. (1950). *Computing machinery and intelligence-AM Turing*. Mind, 59(236), 433.

Ucuzal, H. (2020). Yapay zekâya dayalı anlamsal video işleme yöntemlerinin tıpta kullanılabilirliğinin araştırılması (Master's thesis, İnönü Üniversitesi Sağlık Bilimleri Enstitüsü).

Uddin, M. F., & Gupta, N. (2014, April). Seven V's of Big Data understanding Big Data to extract value. *In Proceedings of the 2014 zone 1 conference of the American Society for Engineering Education* (pp. 1-5). IEEE.

Wang, J., Yang, L., Huo, Z., He, W., & Luo, J. (2020). Multi-Label Classification of Fundus Images With EfficientNet. *IEEE Access*, 8, 212499-212508.

Wong, T. Y., Aiello, L. P., Ferris, F., Gupta, N., Kawasaki, R., & Lansingh, V. (2017). Updated 2017 ICO guidelines for diabetic eye care. *Int Counc Ophthalmol*, 1-33.

Wong, W. L., Su, X., Li, X., Cheung, C. M. G., Klein, R., Cheng, C. Y., & Wong, T. Y. (2014). Global prevalence of age-related macular degeneration and disease burden projection for 2020 and 2040: a systematic review and meta-analysis. *The Lancet Global Health*, 2(2), e106-e116.

Xiao Y, Tian Z, Yu J, Zhang Y, Liu S, Du S, Lan X. (2020) A review of object detection based on deep learning. *Multimedia Tools and Applications* DOI:https://doi.org/10.1007/s11042-020-08976-6.

Yorston, D. (2003). Retinal diseases and vision 2020. *Community Eye Health*, 16(46), 19-20.

## KAYNAKLAR DİZİNİ (Devam Ediyor)

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.

Zhang, Li, Mengya Yuan, Zhen An, Xiangmei Zhao, Hui Wu, Haibin Li, Ya Wang, Beibei Sun, Huijun Li, Shibin Ding, Xiang Zeng, Ling Chao, Pan Li, ve Weidong Wu. (2020). "Prediction of hypertension, hyperglycemia and dyslipidemia from retinal fundus photographs via deep learning: A cross-sectional study of chronic diseases in central China". *PLoS ONE* 15(5):1–11. doi: 10.1371 /journal.pone.0233166.

Zhang, Linglin, Jianqiang Li, I. Zhang, He Han, Bo Liu, Jijiang Yang, ve Qing Wang. 2017. "Automatic cataract detection and grading using Deep Convolutional Neural Network". *Proceedings of the 2017 IEEE 14th International Conference on Networking, Sensing and Control, ICNSC*. pp: 60–65. doi: 10.1109/ICNSC.2017.8000068.

Zhou, Y., Li, G., & Li, H. (2019). Automatic cataract classification using deep neural network with discrete state transition. *IEEE transactions on medical imaging*, 39(2), 436-446.