KUMAŞ ÖRÜNTÜLERİNDE BULUNAN TEKLİ VE ÇOKLU DEFOLARIN MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE TESPİTİ VE KARŞILAŞTIRILMASI

ABDELRAHMAN ELSAYED ALY ELKASAS

HAZİRAN 2025

DİYARBAKIR

DİCLE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KUMAŞ ÖRÜNTÜLERİNDE BULUNAN TEKLİ VE ÇOKLU DEFOLARIN MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE TESPİTİ VE KARŞILAŞTIRILMASI

ABDELRAHMAN ELSAYED ALY ELKASAS

DİCLE ÜNİVERSİTESİ LİSANSÜSTÜ EĞİTİM-ÖĞRETİM VE SINAV YÖNETMELİĞİNİN BİR PARÇASI OLARAK
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANA BİLİM DALINDA
YÜKSEK LİSANS TEZİ
OLARAK HAZIRLANMIŞTIR

HAZİRAN 2025

DİYARBAKIR

KUMAŞ ÖRÜNTÜLERİNDE BULUNAN TEKLİ VE ÇOKLU DEFOLARIN MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE TESPİTİ VE KARŞILAŞTIRILMASI

Abdelrahman Elsayed Aly ELKASAS tarafından Dicle Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği'nin bir parçası olarak hazırlanan bu çalışma, aşağıda bilgileri yazılı jüri üyeleri tarafından değerlendirilerek **Elektrik Elektronik Mühendisliği Ana Bilim Dalı'nda Yüksek Lisans Tezi** olarak kabul edilmiştir.

Prof. Dr. Neslihan DALKILIÇ
Müdür, **Fen Bilimleri Enstitüsü** _____

Prof. Dr. Mehmet Siraç ÖZERDEM
Danışman, Elektrik Elektronik Mühendisliği Bölümü, _____
Dicle Üniversitesi

Sınav Jürisi:

Prof. Dr. Mehmet Siraç ÖZERDEM (*,**)
Elektrik Elektronik Mühendisliği Bölümü, _____
Dicle Üniversitesi

Doç. Dr. Muhammed Ali ARSERİM
Elektrik Elektronik Mühendisliği Bölümü, _____
Dicle Üniversitesi

Doç. Dr. Kerim KARADAĞ
Elektrik Elektronik Mühendisliği Bölümü, _____
Harran Üniversitesi

ONAY

Savunma Tarihi: 20 / 06 / 2025

(*) Jüri Başkanı.
(**)Tez Danışmanı.

Anneme ve Babama…

Dicle Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırlanan bu tez çalışmasında yer alan tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu beyan ederim. Ayrıca, bahse konu bu kural ve ilkelerin gerektirdiği üzere, bu çalışmada özgün olmayan tüm bilimsel içerikleri kurallara uygun biçimde alıntılayıp kaynak gösterdiğimi beyan ederim. Beyanımla çelişen herhangi bir delil bulunduğu takdirde tüm sorumluluğu üstleneceğimi kabul ederim.

Ad, Soyad: ABDELRAHMAN ELSAYED ALY ELKASAS

İmza: .....

TEŞEKKÜR

Öncelikle danışmanım Prof. Dr. Mehmet Siraç ÖZERDEM 'e çalışmanın daha fikir aşamasından, örnekleme, deney setlerini oluşturma, analizlerin yorumlanması ve tez taslağındaki düzeltmeler konularına kadar gösterdikleri destek ve sabırdan dolayı teşekkür ederim. Jüri üyelerime de bu tezin daha nitelikli bir hale gelmesi için yaptıkları eleştiri ve önerileri için teşekkür ederim.

Deney düzeneğinin kurulmasında büyük katkı sağlayan Ünteks Grup'a, özellikle tekstil sektöründeki deneyimleri ve kaynaklarıyla sağladıkları destek için teşekkür ederim.

En önemlisi, beni araştırma hevesiyle coşturdukları ve kendime bazı zamanlar inanmasam da bana inandıkları için anneme ve babama teşekkür ederim.

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

KISALTMALAR LİSTESİ

Kısaltma	Açıklama
AI	Artificial İntelligence
OD	Object Detection
TL	Transfer Learning
CNN	Convolutional Neural Network
YOLO	You Only Look Once
FPN	Feature Pyramid Network
mAP	Mean Average Precision
PAN	Path Aggregation Network
SLP	Single-Layer Perceptron
ANN	Artificial Neural Networks
NN	Neural Network
MLP	Multi-Layer Perceptron
FFNN	Feed-Forward Neural Networks
BFNN	Back-Forward Neural Networks
FC	Fully Connected
Adam	Adaptive Moment Estimation
SGD	Stochastic Gradient Descent
LR	Learning Rate
VGG16	Visual Geometry Group
ResNet	Residual Networks
CSP	Cross Stage Partial
CSPNet	Cross Stage Partial Network
SPPF	Spatial Pyramid Pooling Fast

ÖZET

KUMAŞ ÖRÜNTÜLERİNDE BULUNAN TEKLİ ve ÇOKLU DEFOLARIN MAKİNE ÖĞRENMESİ YÖNTEMLERİ ile TESPİTİ ve KARŞILAŞTIRILMASI

ABDELRAHMAN ELSAYED ALY ELKASAS
Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü
Danışman: Prof.Dr. Mehmet Siraç ÖZERDEM
Haziran 2025, 108 sayfa

Tekstil endüstrisinde kalite kontrol süreçleri, üretim verimliliği ve ürün kalitesi için çok önemlidir. Geleneksel yöntemlerle defo tespiti, özellikle dokusuz kumaşlarda küçük ve düzensiz defolar nedeniyle zordur. Dokusuz kumaşlarda defo tespiti sorununa derin öğrenme tabanlı yenilikçi bir çözüm bu çalışma tarafından sunulmuştur. Yapay Zeka (Artificial Intelligence, AI) tabanlı Nesne Tespiti (Object Detection, OD) ve Transfer Öğrenimi (Transfer Learning, TL) yöntemlerinin kullanılması, bu durumda kumaş defolarının otomatik ve yüksek doğrulukla sınıflandırılmasını sağlamayı amaçlamaktadır.

Çalışmada, Roboflow platformunda etiketlenen ve özgün çektiğimiz bir veri seti kullanılmıştır. Toplam 1239 görüntüden oluşan veri setinde dört farklı defo sınıfı (yag_damlasi, LYC, igne_kirigi ve patlak) yer almaktadır. Yapılan 2584 etiketleme ile ortalama 2.1 defo bulunmuştur. OD yaklaşımında YOLOv5, YOLOv8, YOLOv9 ve YOLOv11 modelleri kullanılırken, TL yaklaşımında VGG16, ResNet50, EfficientNet_B0, DenseNet121, MobileNet_V2 ve Inception_V3 modelleri tercih edildi. Modeller, doğruluk, kesinlik, duyarlılık ve F1 skorları dahil olmak üzere çeşitli metriklerle değerlendirilmiştir.

YOLOv5m, tüm sınıflarda 0.244 güven seviyesinde ve 0.81 F1 skoru ile bir performans sergilemiştir. YOLOv8m 0.268 güven seviyesi ve 0.75 F1 skoru, YOLOv9-c 0.286 güven seviyesi ve 0.73 F1 skoru, YOLOv11-m ise 0.290 güven seviyesi ve 0.74 F1 skoru elde etmiştir. TL modelleri, her imgede tek defo içeren düzenlenmiş veri setinde %99-100 test doğruluğu ile yüksek performans göstermiş; VGG16, DenseNet121 ve MobileNet_V2 %100 doğrulukla öne çıkarken, ResNet50 %99.04 ile en düşük sonucu vermiştir. Ancak, her imgede birden fazla defo içeren orijinal veri setinde TL modellerinin performansları düşmüş; VGG16 %87.30, EfficientNet_B0 %77.38, Inception_V3 %57.54, ResNet50 %47.22 ve MobileNet_V2 %37.30 doğruluk oranları elde edilmiştir. Deneyler, kumaşların bir inceleme makinesi üzerinden geçtiği ve modellerin gerçek zamanlı analiz yaptığı bir sistem kullanılarak yürütülmüştür.

Bu çalışma, dokusuz kumaşlardaki defoları belirlemek için derin öğrenme tekniklerini kullanmanın yararlı olduğunu göstermiştir. Ayrıca, tekstil endüstrisindeki kalite kontrol prosedürlerine yardımcı olmak amacıyla bu teknikleri geliştirmeyi hedeflemiştir. Gelecekte, sistemin çeşitli kumaş türlerinde test edilmesi ve daha fazla defo sınıfının eklenmesi, yöntemin daha genelleştirilebilir hale gelmesine yardımcı olabilir.

Anahtar Kelimeler: Makine öğrenmesi, Dokusuz Kumaş, Defo Tespiti, Nesne Tespiti, Transfer Öğrenimi, YOLO Modelleri

ABSTRACT

DETECTION AND COMPARISON OF SINGLE AND MULTIPLE DEFECTS IN FABRIC PATTERNS USING MACHINE LEARNING METHODS

ABDELRAHMAN ELSAYED ALY ELKASAS
Master of Science in Department of Electrical And Electronic Engineering
Supervisor: Prof.Dr. Mehmet Siraç ÖZERDEM
June 2025, 108 pages

In the textile industry, quality control processes are very important for production efficiency and product quality. Defect detection with traditional methods is difficult, especially in nonwoven fabrics, due to small and irregular defects. An innovative deep learning-based solution to the problem of defect detection in nonwoven fabrics is presented by this study. The use of Artificial Intelligence-Based Object Detection (OD) and Transfer Learning (TL) methods aims to provide automatic and high-accuracy classification of fabric defects in this case.

In the study, an original dataset labeled on the Roboflow platform and captured by us was used. In the dataset consisting of a total of 1239 images, four different defect classes (yag_damlasi, LYC, igne_kirigi and patlak) and 2584 labelings, an average of 2.1 defects were found. In the Object Detection approach, YOLOv5, YOLOv8, YOLOv11 and YOLOv9 models were used, while in the Transfer Learning approach, VGG16, ResNet50, EfficientNet_B0, DenseNet121, MobileNet_V2 and Inception_V3 models were preferred. The models were evaluated with various metrics including accuracy, precision, sensitivity and F1 scores.

YOLOv5m demonstrated a notable performance with a confidence level of 0.244 and an F1 score of 0.81 across all classes. YOLOv8m achieved a confidence level of 0.268 and an F1 score of 0.75, YOLOv9-c a confidence level of 0.286 and an F1 score of 0.73, and YOLOv11-m a confidence level of 0.290 and an F1 score of 0.74. Transfer Learning (TL) models exhibited high performance on the curated dataset with a single defect per image, achieving test accuracies between 99% and 100%; VGG16, DenseNet121, and MobileNet_V2 stood out with 100% accuracy, while ResNet50 recorded the lowest result at 99.04%. However, on the original dataset with multiple defects per image, the performance of TL models declined, yielding accuracy rates of 87.30% for VGG16, 77.38% for EfficientNet_B0, 57.54% for Inception_V3, 47.22% for ResNet50, and 37.30% for MobileNet_V2. The experiments were conducted using a system where fabrics were passed through an inspection machine, enabling real-time analysis by the models.

This study has shown that using deep learning techniques to identify defects in nonwoven fabrics is useful. It also aims to develop these techniques to assist in quality control procedures in the textile industry. In the future, testing the system on various types of fabrics and adding more defect classes can help make the method more generalizable.

Keywords: Deep Learning, Nonwoven Fabrics, Defect Detection, Object Detection, Transfer Learning, YOLO Models

1. GİRİŞ

Tekstil sektöründe kalite kontrol, hem üretimin sorunsuz ilerlemesi hem de ürünlerin yüksek standartta olması için çok önemlidir. Kumaşlardaki defoları tespit etmek de bu sürecin önemli bir parçasıdır. Üretim sürecinde ortaya çıkabilecek problemleri erkenden tespit etmek, ürün kalitesini yükseltir ve maliyetleri düşürür. Özellikle dokusuz kumaşlarda defo bulmak daha zordur, bunun nedeni ise bu kumaşların homojen olması ve defoların (örneğin iğne delikleri, yırtıklar veya yağ lekeleri) küçük veya farklı şekillerde olabilmesidir. Bu tür defoları tespit etmede; makineyi durdurarak kumaşı incelemek, üretim sonrası elle kontrol etmek veya basit yazılımlar kullanmak gibi yöntemlerin kullanıldığı görülmüş ancak bu tür defoların tespitinde yetersiz kaldığı görülebilmiştir. Bu nedenle, daha gelişmiş ve otomatik sistemlere ihtiyaç duyulmaktadır.

Son zamanlarda, AI ile çalışan sistemler, kumaşlardaki defoların tespitinde iyi bir çözüm olduğu ve özellikle Nesne Tespiti (Object Detection, OD) ve Transfer Öğrenimi (Transfer Learning, TL) gibi yaklaşımların, eski yöntemlere kıyasla daha başarılı olduğu görülmektedir. Örneğin, Evrişimsel Sinir Ağı (Convolutional Neural Network, CNN) ve Bir Kez Bakarsın (You Only Look Once, YOLO) gibi modeller, kumaş defolarını hızlı ve doğru bir şekilde tespit ettiği için fabrikalarda kullanılmaktadır. Ancak, dokusuz kumaşlarla ilgili çalışmalar henüz yeterli olmamakla birlikte, Kahraman ve Durmuşoğlu, bu konuda daha fazla derin öğrenme çalışmasına ve özel verilere ihtiyaç olduğunu belirtmiştir [1]. Ayrıca, dokusuz kumaşların düz yapısı, defoların tespitini zorlaştırdığı ve bu nedenle, dokusuz kumaşlardaki defoları tespit etmek için yeni ve farklı çözümlerin geliştirilmesi gerektiği görülmüştür.

Bu tez çalışması, AI kullanarak dokusuz kumaşlardaki defoların tespitine odaklanmıştır. Derin öğrenme ile nesne tanıma ve bilgi aktarımı teknikleri birleştirilerek, kumaşlardaki defoların otomatik olarak ve doğru bir şekilde tespit edilmesine çalışılmıştır. Bu tez kapsamında hazırlanmış bir veri seti kullanılmıştır. Bu veri seti, dokusuz kumaşların tarafımızdan kayda alınan ve Roboflow platformın'da etiketlenen imgelerden oluşmaktadır. Toplam 1239 imgeden oluşan bu veri setinde, dört farklı defo türü (yağ damlası, LYC, iğne kırığı, patlak) bulunmaktadır ve her

imgede ortalama 2.1 defo olacak şekilde 2584 işaretleme yapılmıştır. Bu veri seti, dokusuz kumaşlardaki defo bulma çalışmalarına önemli bir destek sağlamaktadır.

Çalışmada kullanılan veri setinin kapsamındaki örüntüler tek defo veya çoklu defo içerebilmektedir. Tez kapsamında önerilen yaklaşımlara göre söz konusu veri seti üzerinden iki farklı veri seti elde edilmiştir. Bunlar; a) Her örüntü tek defo içerecek şekilde veri seti oluşturulmuş ve TL yöntemleri (VGG16, ResNet50, EfficientNet_B0, DenseNet121, MobileNet_V2 ve Inception_V3) uygulanmıştır. b) Her imgede çoklu defo olabilecek şekilde etiketlendirilerek, OD yöntemler (OLOv5, YOLOv8, YOLOv9 ve YOLOv11) uygulanılmıştır. OD modelleri doğruluk açısından endüstriyel uygulamalarda avantaj sağlarken, TL modelleri ise sınırlı veri setlerinde bile yüksek genelleştirme kapasitesi sunması önemli bir bulgudur ki bu yaklaşımların tez kapsamında kullanılmasının temel sebebidir [2].

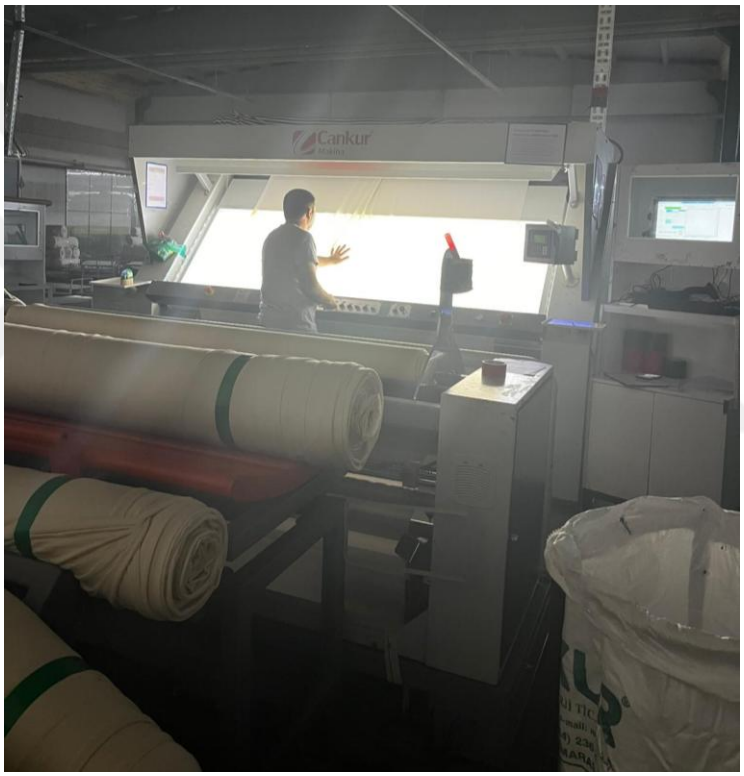
Örüntülerin anlık olarak kayda alınması ve alınan örüntüde defo olup olmadığına ilişkin tespitin online olarak yapılması şeklinde kurulan deney düzeneği, tez kapsamında başarıyla kurulmuştur. Elde edilen özgün veri seti kullanılarak OD ve TL yöntemleri ile defoların tespiti ve yöntemlerin karşılaştırılması amaçlanmıştır. Çalışma, tekstil sektöründe kalite kontrolünü iyileştirmeyi ve otomatik defo bulma sistemlerinin kullanılabilirliğini artırmayı hedeflemektedir.

2. ÖNCEKİ ÇALIŞMALAR

Bu bölümde, dokusuz kumaşlardaki defoların tespiti üzerine yapılan araştırmalar iki ayrı aşamada ayrıntılı olarak incelenmiştir: (1) defoların tespiti için kullanılan araçlar ve düzenekler, (2) defoların tespiti için kullanılan derin öğrenme yöntemleridir. İlk aşama, üretim sırasında ve sonrasında çalışan düzenekler şeklinde alt başlıklara ayrılmıştır. İkinci aşamada ise TL ve OD yöntemleri ayrı ayrı ele alınmıştır.

2.1 Defoların Tespiti İçin Kullanılan Düzenekler

Tekstil işinde kumaşlardaki defoları tespit etmek, üretimin kalitesini ve hızını doğrudan etkileyen önemli bir adımdır.



Şekil 2.1 İnsanların denetlediği kumaş kontrol düzenleri

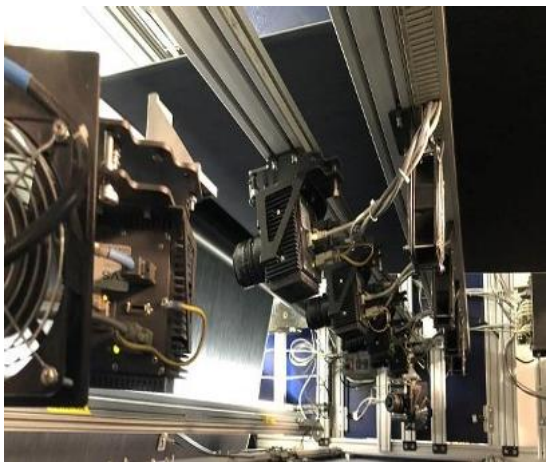
Kumaş fabrikalarında, Şekil 2.1'de görüldüğü üzere, defoların tespitinin genellikle insanlar tarafından gerçekleştiği görülmektedir. Ancak bu yöntem genellikle yeterince iyi sonuç vermediği ve yaklaşık %70 başarı oranı elde edildiği firma yetklilerinde öğrenilebilmektedir. Örneğin göz yanılması, dikkatin dağılması, uzun süre bakmaktan kaynaklanan baş ağrıları gibi sağlık sorunlarından dolayı otomatik sistemlere ihtiyaç duyulmaktadır [3]. Otomatik kumaş inceleme sistemleri, bu sorunları çözmek için yapılmış ve genellikle dört ana parçadan oluşmakatdır: ışık, kamera, mercek ve yazılımdır [4].

Üretim sırasında anlık olarak inceleme yapan sistemler ve üretim bittikten sonra kontrol amaçlı çalışan sistemler olmak üzere iki farklı yaklaşım vardır [5]. Bilimsel çalışmalarda, otomatik kumaş inceleme sistemleri genellikle sabit ve hareketli kamera sistemleri olarak sınıflandırılmıştır. Sabit kameralı sistemler, daha net görüntüler alma konusunda başarılı olduğu ancak daha maliyetli ve kurulumunun daha zor olduğu görülmektedir [6]. Hareketli kamera içeren sistemler daha ucuz ve üretim esnasında kontrole olanak tanır. Ancak kumaş üretildikten sonra defoların tespitinin sağlanması, bu sistemlerin kumaş kalitesi artırma konusunda yetersiz kaldığının bir göstergesidir [5]. Bu bölümde, defoların tespitinde kullanılan sistemler hakkında bilgi verilmiştir.

2.1.1 Üretim esnasında defo tespit makineleri

Tekstil sektöründe, üretim sırasında anlık olarak defoları bulan sistemler, kayıpları azaltmak ve kaliteyi yükseltmek için çok önemlidir. Bu sistemler, kumaş üretim hızıyla birlikte defoları anlık fark edebilir ve üretimde bir sorun olduğunda hızlıca müdahale etme şansı verir. Örneğin, üretim sırasında bir iğnenin kırıldığının fark edilmesi, defonun tüm kumaşa yayılmasını engelleyebilir ve kalitesiz kumaş üretiminin önüne geçebilir [7].

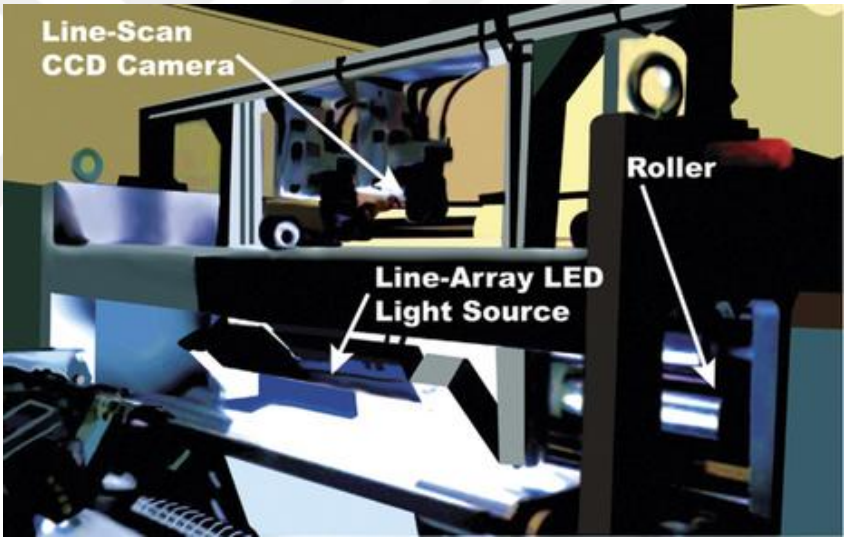
Çalışan düzenekler hakkında yazılanlarda, üretimde kullanılan düzenekler hem hareket edebilen hem de sabit donanımlı sistemleri içerir. Talu ve Varjovi (2023), kot kumaşların dokuma makinelerinde anlık incelenmesi için hareketli ve kamera içeren bir düzenek geliştirmiştir. Bu sistem, düz bir ray üzerinde giden bir kameradan oluşur ve kamera hareketini kumaş üretim hızına göre ayarlayabilimektedir.



Şekil 2.2 Cyclops hareketli kamera sistemi [5]

Şekil 2.2'de görüldüğü gibi, sistem bir dokuma makinesine takılı düz bir ray ve bu ray üzerinde hareket eden bir kameradan oluşur [5]. Ancak, bu sistemin en büyük sorunu, sadece aralıklı dokunmuş kumaşlarda işe yaraması ve ayrı bir bilgisayara ihtiyaç duymasıdır. Cyclops (BMSvision, 2019) gibi hareketli kamera sistemleri de benzer problemlerle karşılaşmıştır; bu sistem sadece bazı kumaş çeşitlerinde çalışabilir ve pahalı olduğu için çok fazla kullanılmamaktadır. Şekil 2.2'de Cyclops sistemine uygun bir örnek verilmiştir; bu sistem bir dokuma makinesi üzerinde çalışan hareketli bir kamera düzeninden oluşmaktadır [8].

Sabit kamera sistemleri, üretim sırasında da kullanılabilir. Örneğin, Schneider ve Aach (2012), dokuma makinelerinde defoları anlık bulmak için sabit bir çizgi kamera sistemi önermiştir. Bu sistem; hızlı üretilen kumaşları incelemek için çizgi ışıklarla desteklenmiş ve defolar %85 oranında başarıyla tespit edilebilmiştir.



Şekil 2.3 Çizgi kamera sistemi kurulumu [9]

Şekil 2.3'te görüldüğü gibi, sistem bir dokuma makinesine takılmış bir çizgi kamera ve ona ek olarak LED ışıklardan oluşmaktadır [9]. Ancak, bu tür sistemler pahalı ve çok sayıda kamera gerektirdiğinden, kullanım alanları kısıtlıdır. Ayrıca, sabit kamera sistemlerinin kumaş üretim hızına ayak uydurmakta zorlanması, küçük ve düzensiz defoların (örneğin, dokusuz kumaşlardaki yağ lekesi veya yırtık gibi) gözden kaçmasına yol açabilmektedir [10].

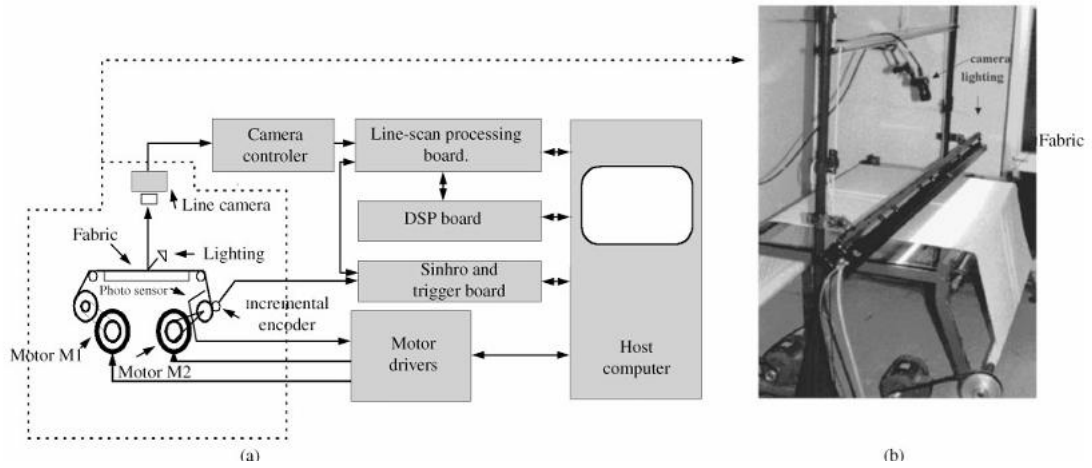
Bu tez kapsamında önerilen düzenek, üretim sırasında anlık inceleme yapabilen sabit bir kamera düzeneğidir. Bu düzen, özellikle dokusuz kumaşlar için yapılmış bir

kontrol cihazında çalışır ve birçok yüksek çözünürlüklü kamera ile kumaşın yüzeyini inceler. Düzeneğin en iyi yanı, defoları tespit etme başarısının yüksek olmasıdır. Ek olarak, düzenek kumaşın üretim hızına ayak uydurabildiği ve anlık tespit edebilme ve defoları anlık tespit ederek üretim kayıplarını en aza indirebilmektedir. Bu özellikleriyle, çalışan düzenek, alandaki hareketli kamera düzeneklerinin kumaş türüne bağlı kısıtlamalarını ve sabit düzeneklerinin yüksek fiyat sorunlarını ortadan kaldırmaktadır. Özellikle dokusuz kumaşların benzer yapısı ve rastgele defolar düşünüldüğünde, önerilen düzeneğin yüksek çözünürlüklü inceleme yeteneği ve yüksek tepit başarımı ile alana önemli bir katkı sağlayacağı düşünülmektedir.

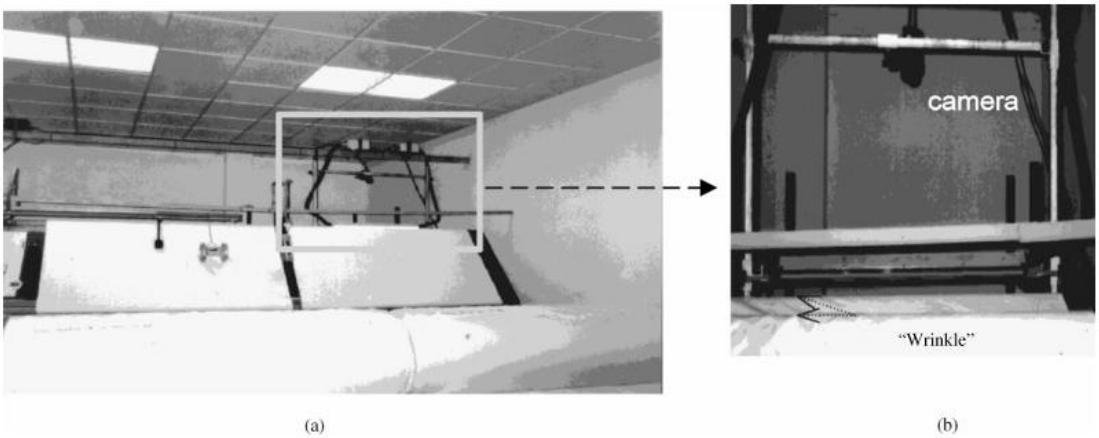
2.1.2 Üretimden sonra defo tepit makineleri

Tekstil sektöründe, tamamlanmış ürünlerin kontrolü için kullanılan sistemler geçmişte daha çok tercih edildiği görülmüştür. Bu sistemler, kumaş üretildikten sonra kalite kontrol bölümüne gelindiğinde kumaşın incelenmesi sağlanır. Ancak bu yöntemin en büyük sorunu, defoların üretim sırasında bulunamaması ve bu nedenle üretim kayıplarının önüne geçilememesidir. Örneğin, üretim sırasında bir iğne kırığının fark edilmesi durumunda, tüm kumaş standartların altında kalitesiz sayılabilir ve bu da üreticiye büyük maddi zararlar verebilir.

Literatürde, üretim sonrası analiz yapan sistemler genellikle hareketsiz kamera sistemleri üzerine odaklanmıştır.



Şekil 2.4 Gerçek zamanlı Kumaş denetim sistemi mimarisi [3]



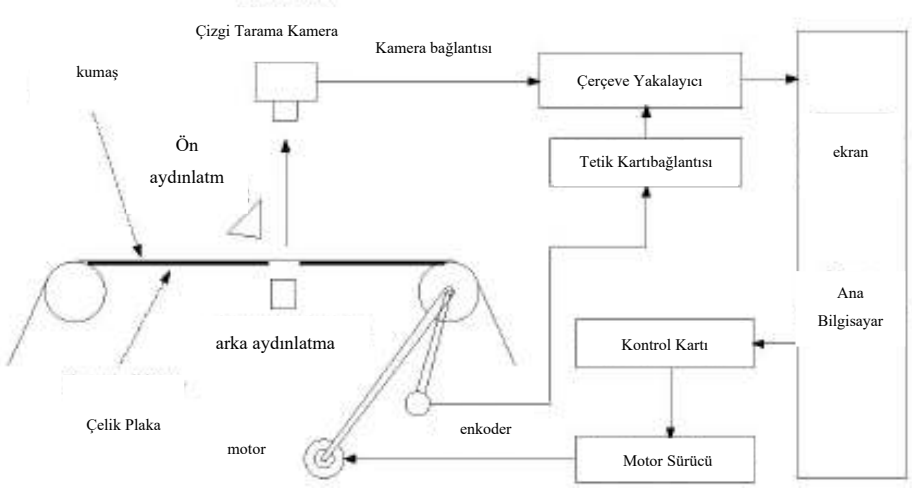
Şekil 2.5 Denetim masasına monte edilmiş görüş sistemi [3]

Stojanovic ve arkadaşları (2001), kumaşları üretildikten sonra incelemek için sabit bir kamera sistemi oluşturmuş ve defoları %70 oranında tespiti sağlanmıştır. Çalışmada önerilen sistem Şekil 2.4'te gösterilmiş, Şekil 2.5'te ise kontrol masasına takılı görüntüleme sistemi yer almaktadır [3]. Bu çalışma, insanların yaptığı kontrollere göre daha iyi sonuçlar vermiş, ancak incelemenin üretimden sonra yapılmasından dolayı defoları üretim sırasında engelleme konusunda yetersiz kalmıştır.

Şekil 2.6 Hareketsiz kamera sistemi kurulumu[11]

Aynı şekilde, Uster Vision gibi sistemler, yan yana çalışan birçok sabit kamerayla geniş kumaş alanlarını tarayarak yüksek çözünürlüklü görüntüler elde edebilir. Şekil 2.6'da görüldüğü gibi, Uster Vision sistemi bir kontrol masası üzerinde sıralı kameralardan ve bu kameralara ışık sağlayan ünitelerden oluşmaktadır [11].

Mak ve arkadaşları (2005), kumaşlardaki defoları tespiti için anlık çalışan bir bilgisayar sistemi geliştirmişlerdir.



Şekil 2.7 Bilgisayar görüş sistemi kurulumu [10]

Benzer bir sistem, üretim bittikten sonraki incelemeler için yapılmış ve defolar %82 oranında tespit edilmiştir. Çalışmada önerilen sistem, bir kamera düzeneği ve kontrol masası içerir (Şekil 2.7) [10]. Üretim sonrası sistemlerin bir dezavantajı da kumaş çeşitlerine göre değişim göstermesidir. Örneğin dokusuz kumaşların aynı tür yapısı ve farklı türden defoları, bu sistemlerin başarısını düşürebilmektedir [1]. Ayrıca, üretim bittikten sonra inceleme yapan sistemler, defoların raporlarını hazırlamada iyi olsalar da, defoların üretim sırasında engellenmesi için bir çözüm sunmaz. Bu durum, özellikle çok ürün üreten şirketler için büyük bir sorundur [12].

Bu tez kapsamında önerilen çalışma, üretim sırasında anlık tespit yapabilen bir sistem sunarak yukarıda belirtilen sorunları çözmeyi amaçlamaktadır. Üretim sonrası sistemlerden farklı olarak, önerilen sistem defoları anlık olarak bulması ve üretimdeki kayıpları engellemesidir.

2.2 Algoritmik Yaklaşımlara göre Defo Tespiti

Derin öğrenme yöntemleri, karmaşık görüntü işleme sorunlarında, özellikle de kumaş defolarının tespitinde son zamanlarda önemli bir çözüm olarak ortaya çıkmıştır. Kenar bulma, eşik belirleme veya histogram gibi eski görüntü işleme yöntemleri, kumaş yüzeyindeki defoların tespiti için yetersiz kalmıştır [13]. Bu konvansiyonel yöntemlerin, özellikle desensiz kumaşların aynı türdeki yapısı ve düzensiz defoları (örneğin, yağ lekesi veya yırtık gibi) söz konusu olduğunda iyi sonuç üretemediği görülmüştür. Derin öğrenme yaklaşımı, bu eksiklikleri gidererek daha doğru sonuçlar üretmiş ve özellikle kumaştaki defonun tespit edilmesinde TL ve OD modelleri sıkça tercih edilmiştir. Ayrıca kumaş türü fark etmeksizin derin öğrenmeye dayalı yöntemler

ile genel kullanıma uygun modellerin sunulabilme potansiyeli söz konusudur. Bu bölümde, hem TL ve hem de OD yaklaşımı ile yapılan tekstil çalışmalarına yer verilmiştir.

2.2.1 TL yaklaşımı ile defo tespiti

TL, önceden eğitilmiş derin öğrenme modellerinin farklı bir görev ya da iş için yeniden kullanılması anlamına gelir. Bu yöntem, özellikle sınırlı veri setleriyle çalışırken etkili bir çözüm sunar ve kumaşlardaki defoların tespiti gibi işlerde sıkça kullanılır. TL, genellikle ImageNet gibi büyük veri setleri üzerinde önceden eğitilmiş modellerin (örneğin, VGG16, ResNet50, Inception_V3) ince ayar (fine-tuning) yapılarak yeni bir görev için uyarlanması şeklinde uygulanır [4].

Literatürde, CNN ile kumaş defolar tespiti konusunu farklı şekillerde ve farklı veri setleri kullanarak incelemiştir. Senthilkumar ve Suguna (2021), dokuma kumaşlardaki defoların tespiti için VGG16 ve ResNet50 modellerini TL ile kullanmış ve %92 oranında doğru sonuçlar almıştır. Ancak, bu çalışmada sadece dokuma kumaşlara odaklanılmış ve dokusuz kumaşların düz yapısından kaynaklanan sorunlara değinilmemiştir [13]. Kumar (2008), normal bilgisayar görme yöntemlerinin kumaş defosu bulmada yetersiz kaldığını ve derin öğrenmeye dayalı TL gibi yöntemlerin bu konuda işe yarayabileceğini belirmiştir [14]. Goyal (2018), tekstil defolarını bulmak için DenseNet121 modelini TL ile kullanmış ve %93 doğruluk oranı yakalamıştır. Ancak, bu çalışmada kullanılan veri setinin çeşitliliği ve kumaş türleri hakkında yeterli bilgi verilmemesi eksiklik olarak görülmüştür [11].

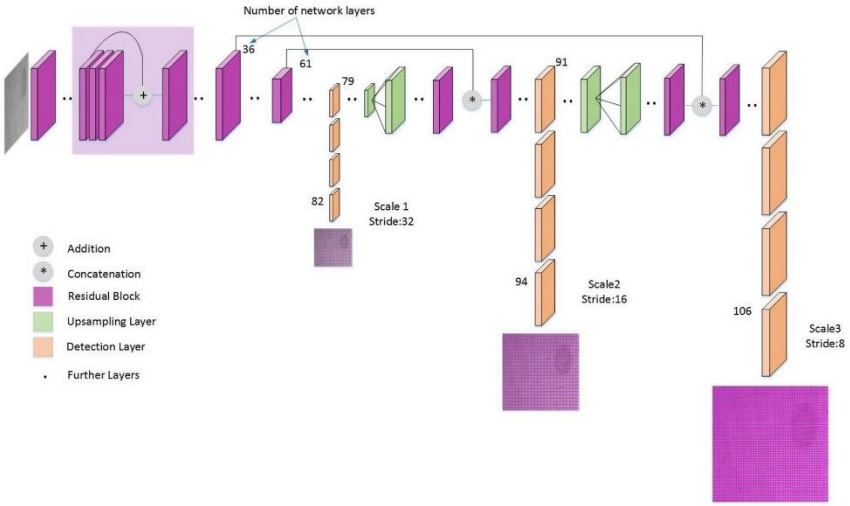
2020 ve sonrası çalışmalar, TL yöntemlerinin kumaş defo tespiti alanındaki uygulamalarını daha da ileriye taşımıştır. Jeyaraj ve Nadar (2020), derin öğrenmeyi kullanarak bir sistem tasarlamış ve kumaş defolarını tespit etme konusunda bir çalışma yapmışlardır. YOLO ile desteklenen bir derin öğrenme yaklaşımı ile %93 doğruluk oranı sağlamışlardır. Bu çalışma, özellikle üretim sonrası kontrollerde derin öğrenmenin ne kadar iyi olduğunu göstermiştir [2]. Hu ve arkadaşları (2020), gözetimsiz öğrenme ile kumaş defolarını bulmak için derin evrişimli üretken çekişmeli ağları (Deep Convolutional Generative Adversarial Network - DCGAN) kullanmış ve %90 F1 skoru elde etmiştir. Bu yöntem, etiketlenmemiş veri setleriyle çalışabilme avantajı sunmuş ve özellikle küçük işletmeler için uygun fiyatlı bir çözüm sağlanmıştır

[15]. Siegmund ve ekibi (2020), önemli noktaları bulma ve derin öğrenme ile kumaş defolarını bulma üzerine bir araştırma yapmış ve %91 doğruluk elde etmiştir. [16] nolu çalışma, derin öğrenmenin çok küçük elyaf defolarını bulmada işe yarayabileceğini göstermiştir. Kahraman ve Durmuşoğlu (2023), derin öğrenmeye dayalı yöntemlerin dokusuz kumaşlardaki eksikliklerine değinmiş ve bu kumaşların aynı türden yapısının derin öğrenme modelleri için daha zor olduğunu belirtmiştir [1].

Literatürdeki TL araştırmaları, daha çok dokuma kumaşlara yoğunlaşmış ve dokusuz kumaşlar için yapılan uygulamalar az sayıda kalmıştır. Ek olarak, TL'ye dayalı araştırmaların çoğu, çok fazla veriye ihtiyaç duyan ResNet50 gibi modelleri kullanmış ve az veriyle çalışan şirketler için uygun bir çözüm olmamıştır. Bu tez çalışmasında, dokusuz kumaşlara özel bir veri seti üzerinde VGG16, ResNet50, EfficientNet_B0, DenseNet121, MobileNet_V2 ve Inception_V3 modellerini TL yöntemiyle kullanarak bu eksikliğin giderilmesi hedeflenmiştir.

2.2.2 OD yaklaşımı ile defo tespiti

OD, kumaşlardaki defoların tespiti için kullanılan bir yöntemdir, bu defoların konumunu ve türünü belirler. YOLO modelleri, hız ve doğruluk açısından bu alanda sıklıkla tercih edilmiştir. YOLO modelleri, tek bir sinir ağı ile hem sınıflandırma hem de konum belirleme yapabilme kapasitesine sahiptir [17]. Faster R-CNN ve Fast R-CNN gibi modeller de OD alanındaki önemli yaklaşımlardandır [18, 19]. Bu modeller, özellikle karışık kumaşlarda aynı anda birçok defo bulmak gerektiğinde iyi bir çözüm sunabilmektedir [14]. YOLO modellerinin tekstil defolarını bulmada nasıl kullanıldığı çeşitli araştırmalarda gösterilmiştir. Jing ve arkadaşaları (2020), tekstil üretimindeki defoların tespiti için geliştirilmiş bir YOLOv3 modelini kullanmış ve %5'ten düşük bir defo oranı elde etmiştir.



Şekil 2.8 YOLOv3 ağının yapısı [20]

YOLOv3 modelinin yapısı Şekil 2.8'de gösterilmektedir [20]. Bu çalışma, YOLOv3'ün hızlı çalışabildiğini ve geliştirilmiş versiyonuyla tekstil defo tespitinde yüksek başarı sağladığını vurgulamıştır. Yang ve arkadaşları (2022), Transformer temelli bir model (ACCTNet) kullanarak kumaş defolarını tespit etmiş ve %94 F1 skoru elde etmiştir [21]. ACCTNet, özellikle kumaş yüzeyindeki bağlantıları anlama konusunda başarılı olmuş ve diğer YOLO modellerine göre daha iyi sonuçlar vermiştir. Prunella ve arkadaşları (2023), fabrikalardaki yüzey defolarını otomatik olarak bulmak için derin öğrenme yöntemlerini araştırmış ve YOLO modellerinin bu konudaki gücüne dikkat çekmiştir. Bu inceleme, özellikle yüksek çözünürlüklü imgeler üzerinde yapılan çalışmalarda derin öğrenme modellerinin ne kadar başarılı olduğunu göstermiştir [22].

OD alanında, YOLO modellerinin yanı sıra, Faster R-CNN ve Fast R-CNN gibi AI modelleri de kullanılmıştır. Ren ve ekibi (2015), Faster R-CNN modelini sunarak, gerçek zamanlı defo bulma işlemlerinde bir yenilik yaparak, bölge öneri ağlarının OD için kullanılmasını önermiştir [18]. Aynı şekilde, Girshick (2015), Fast R-CNN modelini daha iyi hale getirerek OD işlemlerini hızlandırmış ve kumaşlardaki defoların tespiti gibi alanlarda işe yarar bir çözüm ortaya koymuştur [19]. Ancak, bu modellerin kumaş defo tespiti alanında kullanımı az olmuş ve genellikle daha çok veriye ihtiyaç duymaları sebebiyle fabrikalardaki uygulamalarda kullanışlılıkları sorgulanmıştır [9]. Bu tez çalışmasında, dokusuz kumaşlarla ilgili özel bir veri

kullanarak YOLOv5, YOLOv8, YOLOv11 ve YOLOv9 modelleri eğitilmiş ve bu konudaki başarımları karşılaştırılmıştır.

2.3 Tez Çalışmasının Katkısı

Literatür incelediğinde, dokusuz kumaşlardaki defoların tespitine yönelik çalışmaların yeterince araştırılmadığı görülmüştür. Çalışma prensibi olarak, şu anki sistemler daha çok dokuma kumaşlarına uygun olup, dokusuz kumaşların düzgün yapısından kaynaklanan sorunlara çözüm üretemediği görülmüştür. Üretim sırasında anlık olarak inceleme yapabilen sistemlerin her kumaşa uygun olmaması ve pahalı olması, yaygınlaşmalarını zorlaştırmaktadır [5]. Örneğin, Talu ve Varjovi (2023) tarafından sadece aralıklı dokunmuş kumaşlarda işe yarayan bir sistem önermiş ve bu sistemin dokusuz kumaşlar için uygun olmadığı belirtilmiştir [5]. Üretim bittikten sonra inceleme yapan sistemler ise defoların üretim sırasında engellenmesini sağlamadığı için üretim kayıplarına neden olduğu görülmüştür [9]. Ayrıca üretim sonrası defo tespit sistemlerinin kumaş türüne göre değişmesi, özellikle dokusuz kumaşlardaki defoları (örneğin, yağ lekesi veya yırtık) bulunmasını zorlaştırmaktadır [1]. Literatürde var olan çalışmaların büyük çoğunluğu konvansiyonel yaklaşımlar ile dokuma kumaşlar üzerine yoğunlaşılmıştır. Diğer bir taraftan derin öğrenmeyle yapılan uygulamalar ise dokusuz kumaşlar için çok az sayıda çalışıldığı görülmüştür. TL tabanlı çalışmalar, genellikle büyük veri setlerine ihtiyaç duyan modelleri (örneğin, ResNet50) kullanmış ve küçük veri setleriyle çalışan işletmeler için pratik bir çözüm sunmamıştır [9]. Geleneksel yöntemler ise genellikle örgülü kumaşlardaki kolayca görülebilen defoları (iplik kopması gibi) bulmaya odaklanmış, örülmemiş kumaşların düzgün yapısından kaynaklanan sorunlara değinmemiştir [20]. Ayrıca, Faster R-CNN ve Fast R-CNN gibi modellerin çok fazla veriye ihtiyaç duymaları ve yavaş çalışmaları yüzünden, kumaş defosunun tespitinde kullanılması kısıtlı kalmıştır [18, 19].

Literatürde dikkat çeken diğer bir tespit; kumaş defo sınıflandırma çalışmalarında kullanılan veri setlerinin çeşitliliği ve erişilebilirliği ile ilişkilidir. Birçok çalışmada kullanılan veri setinin paylaşılmadığı görülmüştür. Bu durum, sonuçlara ilişkin doğruluğun kontrolünü ve farklı çalışmalar ile karşılaştırılmasını zorlaştırır [12]. Örneğin, Hu ve arkadaşları (2020), DCGAN temelli bir yöntem kullanmış ancak çalışmada kullanılan veri seti hakkında yeterince bilgi verilmemiştir [15]. Aynı

şekilde, Siegmund ve arkadaşları (2020), elyaf düzeyindeki defoları bulmuş ancak veri setinin büyüklüğü ve nasıl oluşturulduğu hakkında yeterli bilgi sunmamıştır [16].
Yukarıda belirtilen çerçevede, tez kapsamında yapılan çalışmanın katkıları aşağıda listelenmiştir.

- Öncelikle, dokusuz kumaşlar için özel bir örme makinesi üzerinde çalışan, sabit bir kamera düzeneği oluşturulmuştur ve bu sistemin üretim sırasında gerçek zamanlı olarak sınıflandırma yeteneği test edilmiştir. Önerilen sistemin defo kaçırma oranı %5 oranından az olduğu, literatürdeki hareketli ve hareketsiz sistemlere kıyasla daha avantajlı bir çözüm sunduğu tespit edilmiştir. Özellikle önerilen sistemin, dokusuz kumaşların homojen yapısına uygunluğu ve yüksek çözünürlükte tarama kapasitesi, bu tez çalışmasının farklılığını ortaya koymaktadır.

- Dokusuz kumaşlara özgü bir veri seti oluşturulmuş ve bu veri seti üzerinde hem TL hem de OD yöntemleri uygulanmıştır. Oluşturulan veri seti, 1239 görüntüden oluşmakta ve dört farklı defo sınıfını (yag_damlasi, LYC, igne_kirigi, patlak) içermektedir. Tez kapsamında önerilen OD modelleri ile %99-100 doğruluk aralığında performans elde edilirken, önerilen YOLO modelleri ile 0.75-0.881 F1 skoru aralığında başarı performansı elde edilmiştir. Bu sonuçlar, özellikle dokusuz kumaşlara yönelik derin öğrenme tabanlı yöntemlerin potansiyelini ortaya koymaktadır. Ayrıca, çalışmamızda kullanılan veri seti, açık kaynak olarak paylaşılabilecek şekilde tasarlanmıştır.

- Tez çalışmasında, OD ve TL yöntemleri aynı veri seti üzerinde karşılaştırmalı olarak değerlendirmiştir. Bu yaklaşım, literatürde sıklıkla ayrı ayrı tartışıldığı görülmektedir. Örneğin, YOLO modellerine kıyasla TL modellerin performansının yüksek olduğu; (VGG16, DenseNet121, MobileNet_V2 ve Inception_V3) ancak YOLOv5'in 0.881 F1 skoru, defoların konumunu bulmadaki başarısını göstermiştir. Bu karşılaştırma, dokusuz kumaş defolarını bulmak için hangi yöntemin daha iyi çalıştığını göstermektedir.

- Tez çalışmasında, kumaş defolarının tespiti konusunda endüstriyel uygulamalar için pratik bir çözüm üretilmesi hedeflenmiştir. Büyük veri setlerine ve yüksek işlem gücüne ihtiyaç duyan modelleri kullanan araştırmalar, küçük ve orta ölçekli şirketler için uygun değildir [9]. Önerilen çalışmada kullanılan EfficientNet_B0 ve MobileNet_V2 gibi küçük modeller, düşük işlem gücüyle bile yüksek doğruluk oranları sağladığı görülmüştür. Sistemin üretim sırasında çevrim içi analiz kapasitesi, üretim sırasında meydana gelen zararları azaltarak endüstriyel uygulamalarda önemli bir avantaj sağlayacağı düşülmektedir.

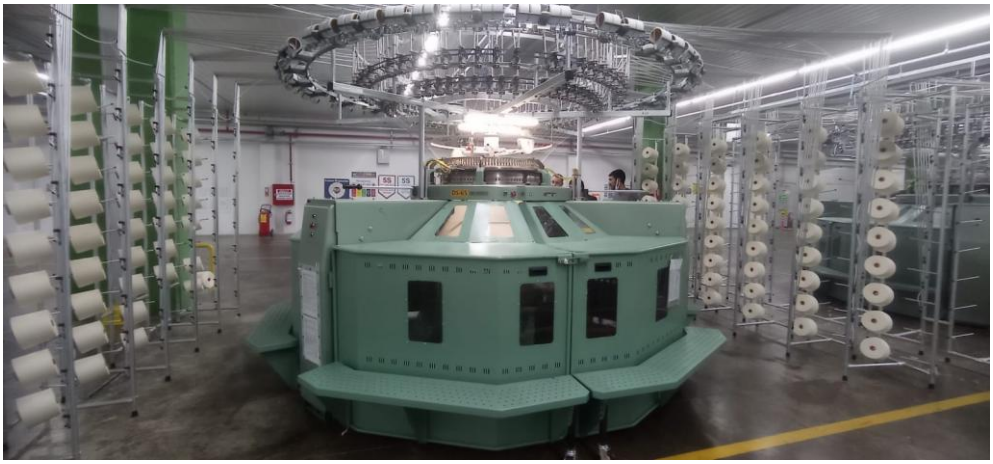
3. MATERYAL VE METOD

Bu çalışma, tekstil sektöründe sıkça rastlanan ve üretimi doğrudan etkileyen dokusuz kumaşlardaki defo bulma sorununa, son zamanlarda popülerleşen derin öğrenme teknikleriyle yeni bir çözüm getirmeyi hedeflemektedir. Tekstil üretiminde kalite kontrolünün çok önemli olduğu günümüzde, insanların yaptığı defo bulma yöntemleri yavaş, pahalı ve kişiden kişiye değişebilir. Bu nedenle, bu tez çalışmasında AI tabanlı, özellikle de sinir ağları (Neural Network, NN) ve TL gibi derin öğrenme yöntemleri kullanılarak, kumaş defolarının otomatik, hızlı ve doğru bir şekilde sınıflandırılması amaçlanmıştır. Bu amaca ulaşması ile, üretim sırasında oluşabilecek defoları erkenden tespit ederek; malzeme kaybını azaltılması, üretim harcamalarını düşürmesi ve ürün kalitesinin yükseltilmesi sağlanacaktır.

Bu amaca yönelik tez kapsamında, dokusuz kumaş üretim hattına eklenebilen, özel olarak tasarlanmış bir düzenekle donatılmış, anlık görüntü analizi yapabilen akıllı bir sistem geliştirmiştir. Bu sistem, kumaşın yüzeyini sürekli tarayarak, olası defoları anlık tespit edebilecek özelliktedir. Sistem sadece defo bulmakla kalmayıp, aynı zamanda bulunan defoların çeşidini (yırtık, leke, delik vb.) ve kumaştaki konumunu da belirleyerek, düzeltme işlemlerinin kolaylaştırılması hedeflenmiştir.

3.1 Veri Seti için Deney Düzeneği

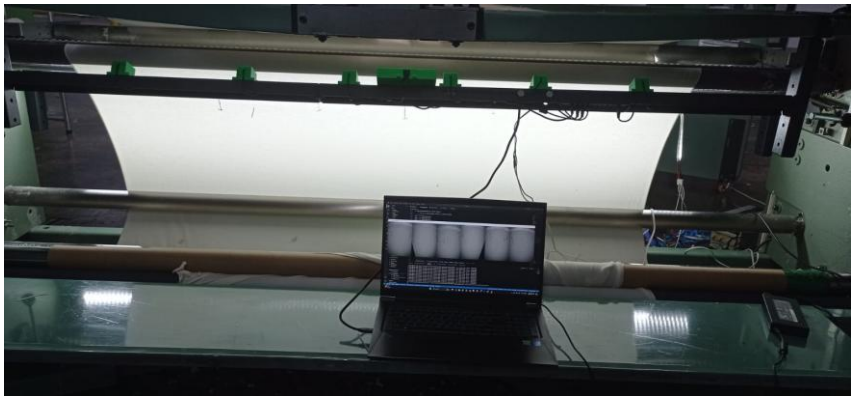
Bu tez kapsamında, dokusuz kumaşlardaki üretim defoları otomatik olarak bulmayı hedeflenmiştir. Mevcut veri setlerinin yeterli olmaması nedeniyle, özel bir veri seti oluşturulmuştur. Bu veri seti, piyasada bulunan genel veri setlerinden ayrı olarak, belirli üretim şartlarını ve defo çeşitlerini içerecek şekilde hazırlanmıştır. Veri setinin oluşturulmasında, ekibimizin geliştirdiği özel bir sistem kullanılmıştır. Bu sistem, yüksek çözünürlüklü imgeler elde etmeyi sağlamış ve dokusuz kumaş yüzeyindeki en ufak ayrıntıların bile yakalanmasına imkan vermiştir.



Şekil 3.1 Yuvarlak örme makinesi (Circular knitting machine)

İmgelerin alınması, Şekil 3.1'de gösterildiği gibi, endüstriyel bir yuvarlak örme makinesi (circular knitting machine) içerisinde gerçekleştirilmiştir. Bu makine, dokusuz kumaş yapımının temel aracıdır ve iplikleri karmaşık bir şekilde örerek kumaşın şeklini oluşturur. Tez kapsamında oluşturulan düzenek ile kumaş üretilirken kumaşın yüzeyi taranmış ve imgeler kayda alınmıştır. Böylece kumaşın üretim aşamasındaki anlık doku formu kayda alınmıştır. Tez kapsamındaki önerilen düzenek, üretim hattının bir parçası olarak çalışabilir ve sürekli bilgi toplayabilir. Bu durum, anlık defo tespit etme ve kalite kontrolü için büyük bir fayda sağlamıştır.

Veri seti, tekstil sektöründe kalite kontrolünü artırmak için dikkatle tasarlanmıştır. 6 yüksek çözünürlüklü sabit kamera, dairesel örgü makinesine stratejik olarak yerleştirilmiş ve bir dizi seri LED ışıklarla desteklenmiştir. Bu aydınlatma, kumaş yüzeyini homojen şekilde aydınlatarak ince iplik kopmaları gibi az belirgin defoları belirginleştirdiği görülmüştür. 2MP Aptina AR0234 sensörlü küresel deklanşörlü kamera ile yüksek hızlı hareketler kayda alınabilmiştir. Ayrıca kullanılan kamera ile 1920x1200 çözünürlükte MJPEG formatında 90 fps'ye kadar kare hızı sunabilmektedir. UVC uyumluluğu sayesinde tak-çalıştır özelliğiyle bilgisayar'a bağlantısı yapılabilmektedir. 25mm lens kullanılarak bozulmasız ve detaylı görüntüler elde edilmiştir. Sistem, Beelink Mini S12 Intel N95 (8GB RAM, 256GB SSD, Windows 11 Pro) mini PC ile desteklenerek veri işleme hızını optimize edilmiştir (Şekil 3.2).

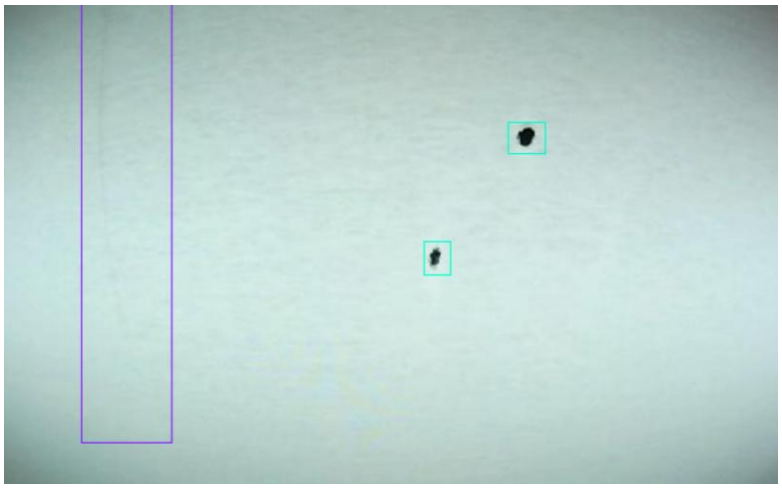


Şekil 3.2 Dairesel örgü makinesinde Kameraların ve LED Işıkların Yerleşimi

3.2 Veri Seti

Oluşturulan veri seti, 1239 adet (1920x1200) boyutlarında imgelerden oluşmaktadır. Bu imgeler, desensiz kumaşlardaki çeşitli defo tiplerini içerir. Veri seti etiketlemek için Roboflow platformu kullanılmış. Roboflow, AI modellerini eğitmek için gereken etiketleme işini kolaylaştıran ve hızlandıran bir araçtır. Platformun kolay kullanılır yapısı ve birlikte çalışma özellikleri sayesinde, etiketleme işi verimli bir şekilde yapılmış.

Veri setinde, dokusuz kumaşlarda yaygın olarak görülen dört çeşit defo belirlenmiştir: "yag_damlasi", "LYC" (LYC lifinden kaynaklanan sorunlar), "igne_kirigi" ve "patlak" (kumaşta yırtık veya delik). Bu defolar, dokusuz kumaş üretiminde en çok rastlanan ve kaliteyi en çok etkileyen defolardır. Veri setinde toplam 2584 etiketleme yapılmıştır, bu da her imgede ortalama 2.1 defo olduğu anlamına gelmektedir.



Şekil 3.3 Dokusuz kumaş örneği ve tespit edilen defolar

Roboflow kullanılarak etiketlenmiş, dokusuz bir kumaş üzerindeki defo örnekleri şekil 3.3'de gösterilmiştir. Şekildeki imgede hem patlak hem de iğne kırığı defoları içerir.

3.2.1 Veri setinin dağılımı ve analizi

Veri setimiz öncelikle 5 sınıftan oluşmaktadır. Oluşturulan defo sınıflarındaki dağılım tablo 3.12de gösterilmiştir.

Tablo 3.1 Dokusuz kumaş defo Dağılımı (Başlangıç Verisi)

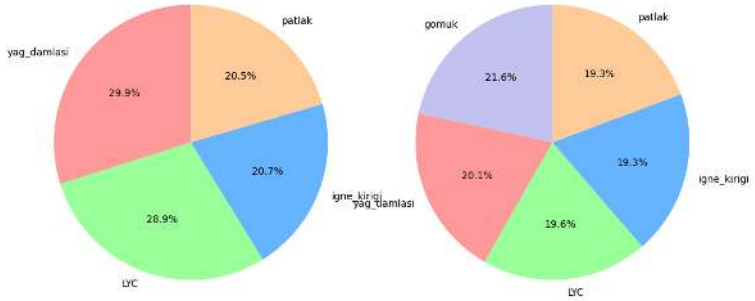
Defo Sınıfı	Etiket Sayısı	Yüzde (%)
gömük	1531	21.6%
igne_kirigi	1425	20.1%
yag_damlasi	1387	19.6%
patlak	1369	19.3%
LYC	1369	19.3%

Oluşturulan veri seti için yapılan ön analizlerde, gömük sınıfının kumaş üzerinde oluşturduğu küçük ölçekli defo, gömük sınıfının tespitini zorlaştırdığı ve genel sistemin başarı performansını düşürdüğü görülmüştür. Bun sebepten dolayı gömük sınıfı değerlendirme dışı bırakmıştır. Gömük türünün veri setinden çıkarılması ile elde edilen yeni veri setinin sınıf dağılımı tablo 3.2'de göstermiştir.

Tablo 3.2 Optimize Edilmiş Dokusuz Kumaş Veri Seti Dağılımı

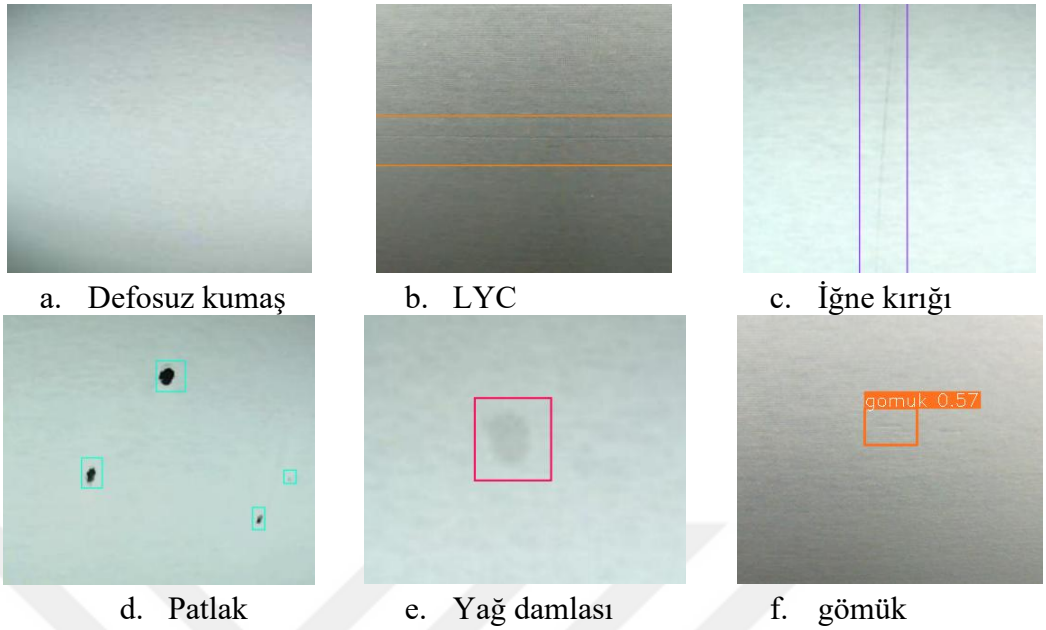
Defo Sınıfı	Etiket Sayısı	Yüzde (%)
yag_damlasi	773	29.9%
LYC	748	28.9%
igne_kirigi	534	20.7%
patlak	529	20.5%

Şekil 3.4'da, başlangıç ve optimize edilmiş veri setlerindeki defo sınıflarının oranları iki ayrı pasta grafiği ile görselleştirilmiştir.



Şekil 3.4 Veri setlerindeki defo sınıflarının dağılımını gösteren pasta grafikleri

Oluşturulan defoların birer örneği şekil 3.5te verilmiştir.



a. Defosuz kumaş b. LYC c. İğne kırığı

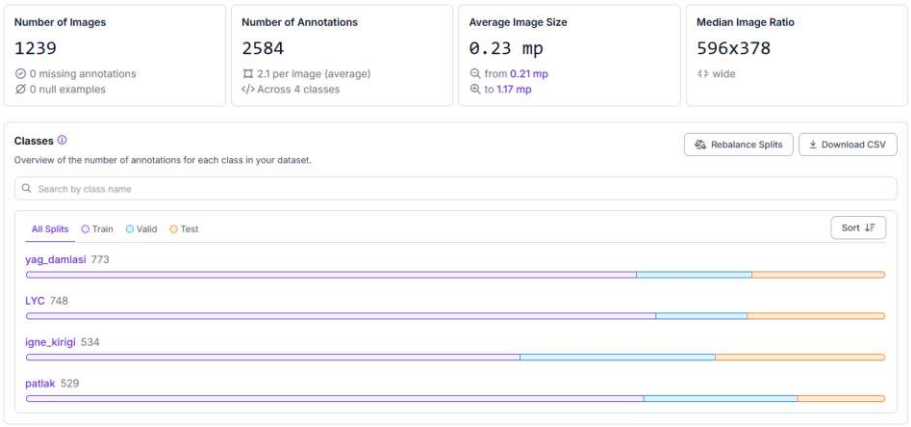
d. Patlak e. Yağ damlası f. gömük

Şekil 3.5 Defo örnekleri

3.2.2 Veri setinin etiketlenmesi

Kameralardan alınan işlenmemiş imgeler Roboflow'a yüklenerek etiketleme işlemlerini uygulanmıştır. Roboflow, makine öğrenimi projeleri için özel olarak geliştirilmiş, imge etiketleme, veri hazırlama ve model eğitme işlemleri kolaylaştıran geniş kapsamlı bir platformdur [23]. Etiketleme sürecinde, her bir defo sınıfı için sınırlayıcı kutular (bounding boxes) çizilmiş ve defolar sınıflandırılmıştır.

Tez kapsamında kullanılan veri seti, toplam 1239 görüntü ve 2584 etiket içermektedir. Dört farklı defo sınıfını kapsamaktadır: yag_damlasi, LYC, igne_kirigi ve patlak.



Şekil 3.6 Roboflow platformunda veri seti istatistikleri

Şekil 3.6'da, Roboflow platformunda veri setin detaylı istatistikleri yer almaktadır. Toplam 1239 görüntü ve 2584 etiketleme bulunmakta olup, her görüntüde ortalama 2.1 etiket yer almaktadır. Ortalama görüntü boyutu 0.23 megapikseldir ve görüntüler 596x378 piksel boyutundadır. Sınıf dağılımları şu şekildedir: yag_damlasi (773 etiket), LYC (748 etiket), igne_kirigi (534 etiket) ve patlak (529 etiket).

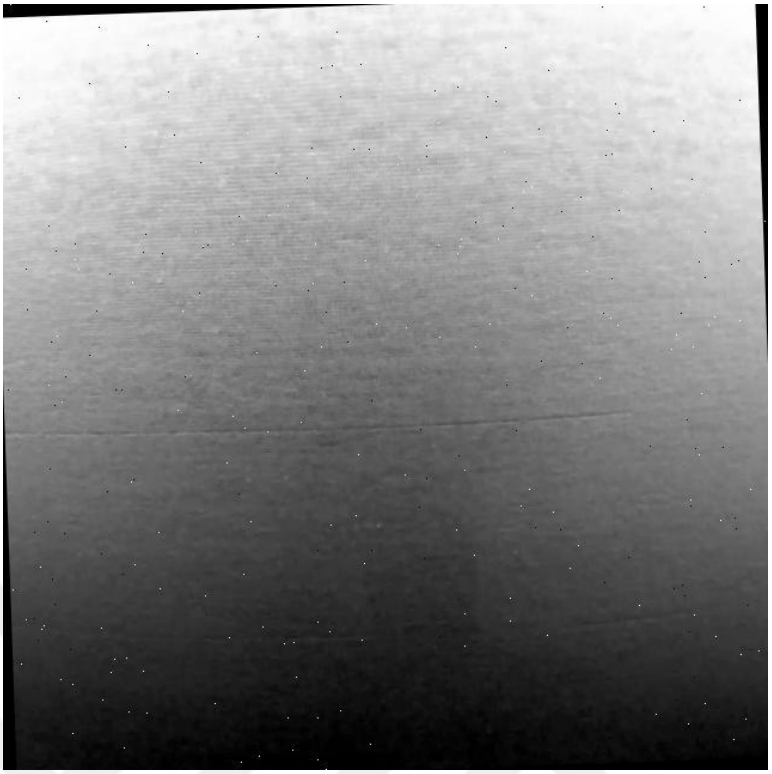
3.2.3 Veri ön işleme ve veri artırımı

Veri seti; eğitim, doğrulama ve test olmak üzere üç ayrı kümeye ayrılmıştır. İmgelerin %70'i (867 imge) eğitim için, %15'i (186 imge) doğrulama için ve kalan %15'i (186 imge) ise test için ayrılmıştır. Veri ön işleme aşamasında, imgeleri modellerin ihtiyaçlarına göre uygun olarak yeniden boyutlandırılmıştır. OD modelleri (örneğin, YOLO modelleri) için görüntüler 640x640 piksel boyutuna, TL modelleri için ise 224x224 piksel boyutuna getirilmiştir. Ayrıca, tüm imgelere normalizasyon uygulanmış ve piksel değerleri [0, 1] aralığına ölçeklendirilmiştir.

Veri çoğaltma (data augmentation) yöntemleri kullanılarak tez kapsamında veri seti zenginleştirilmiştir. Bu teknikler, modellerin genelleştirme kapasitesini artırmayı ve overfitting riskini azaltmayı amaçlamıştır [3]. Uygulanan veri artırımı işlemleri şunlardır:

- Döndürme (Rotation): Görüntüler rastgele olarak ±30 derece döndürülmüştür.
- Çevirme (Flipping): Görüntüler hem yatay hem de dikey olarak çevrilmiştir.
- Parlaklık ve Kontrast Ayarı: Görüntülerin parlaklık ve kontrast değerleri %10 oranında değiştirilmiştir.
- Gürültü Ekleme (Noise Addition): Görüntülere Gaussian gürültüsü eklenerek modellerin gürültüye karşı dayanıklılığı artırılmıştır.
- Kesme (Cropping): Görüntülerin rastgele bölümleri kesilerek farklı perspektifler oluşturulmuştur.

Bu işlemler, özellikle küçük ve az belirgin defoların tespiti için yapılmıştır. Şekil 3.7'de, veri çoğaltma yapılmış bir imge örneği verilmiştir.



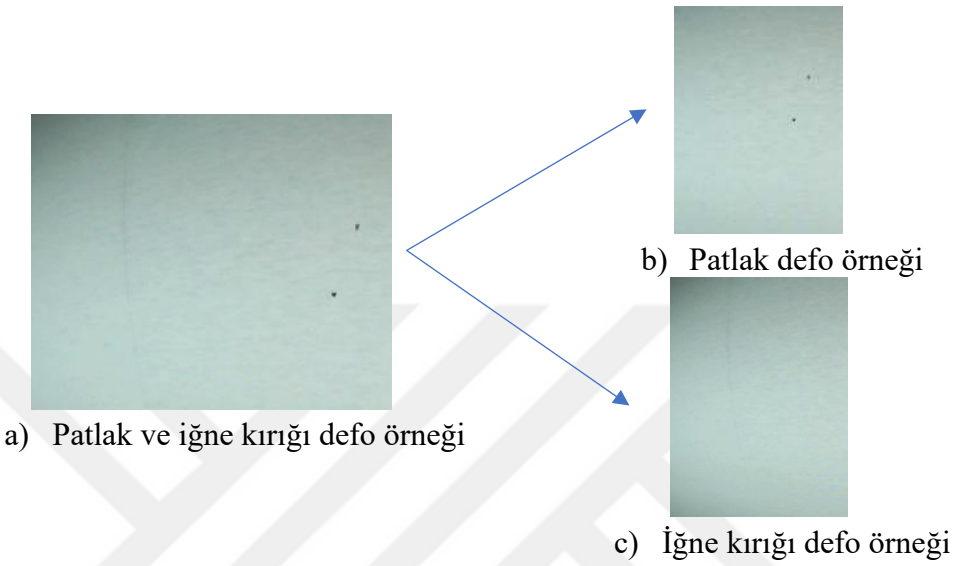
Şekil 3.7 Veri artırımı uygulanmış bir örnek görüntü

Şekil 3.7'de, bir dokusuz kumaş imgesine ön işleme ve veri artırımı işlemleri uygulanmış bir imge gösterilmektedir. Ön işleme aşamasında, görüntüye şu işlemler yapılmıştır: Auto-Orient ile görüntünün yönü otomatik olarak düzeltilmiş, Resize ile görüntü 640x640 piksel boyutuna yeniden boyutlandırılmış ve Auto-Adjust Contrast ile histogram eşitleme (histogram equalization) kullanılarak kontrast otomatik olarak ayarlanmıştır. Veri artırımı aşamasında ise her bir eğitim örneği için 3 farklı çıktı oluşturulmuştur: Flip ile görüntü yatay olarak çevrilmiş, Crop ile görüntü %4 minimum ve %16 maksimum yakınlaştırma oranlarıyla kırpılmış, Saturation ile renk doygunluğu %-13 ile %+13 arasında değiştirilmiş ve Noise ile piksellerin %0.1'ine kadar gürültü eklenmiştir.

3.2.4 TL için veri setinin düzenlenmesi

Transfer Öğrenimi (TL) modelleri, derin öğrenmede yaygın ve etkili bir yaklaşımdır. Bu modeller, önceden eğitilmiş bir ağın (örneğin, VGG16 veya ResNet50) bilgisini ve yetkinliklerini, yeni bir veri setine uyarlayarak daha az veriyle hızlı ve yüksek doğrulukta sonuçlar elde etmeyi amaçlar. Bu yöntem, özellikle sınırlı veri kaynaklarıyla çalışırken zaman ve hesaplama maliyetlerini azaltır.

TL modellerinin başarısı, veri setinin özelliklerine ve defolu kısımların dağılımına doğrudan bağlıdır. Farklı veri setlerinde performansın değişkenlik göstermesi, bu bağımlılığın bir sonucudur. Tez kapsamında kullanılan dokusuz kumaş veri setinde, tek defolu imgeler üzerinde yoğunlaşarak TL modellerinin genelleme yeteneğinin artırılması hedeflenmiştir.



Şekil 3.8 TL için veri setinin düzenlenmesi

Veri içindeki örüntüler birden fazla farklı defo içerebilmektedir. Örneğin, şekil 3.8(a)'da gösterildiği gibi bir örüntüde hem "iğne kırığı" hem de "patlak" defoları yer alabilmektedir. Ancak, yapmış olduğumuz deneysel çalışmalar sonucunda, TL modellerinin bu tür birden fazla defo sınıfı içeren imgeler üzerinde zorlandığı ve dolayısıyla başarı oranının gözle görülür bir şekilde düştüğü tespit edilmiştir. Modeller, defoların iç içe geçtiği, birbirini etkilediği veya gölgelediği durumlarda, hangi defonun mevcut olduğunu doğru bir şekilde belirlemekte güçlük çekildiği görülmüştür. Örneğin, bir imgede hem "iğne kırığı" hem de "patlak" defoları eş zamanlı olarak bulunduğunda, TL modellerinin doğruluk oranı %65.5 gibi düşük bir seviyeye kadar gerilemiştir. Bu düşüş, TL modellerinin karmaşık ve çok sınıflı defo tespiti senaryolarında genelleştirme kapasitesinin sınırlı olduğunu açıkça ortaya koymuştur [25]. Bu durum, TL modellerinin eğitildiği veri setinin kalitesinin ve yapısının, modelin performansı üzerindeki kritik rolünü bir kez daha göstermiştir.

Bu önemli sorunu çözmek ve TL modellerinin performasını tam olarak ortaya çıkarmak için, veri seti üzerinde kapsamlı bir yeniden yapılandırma çalışması gerçekleştirilmiştir. Yeniden yapılıandırma, her bir imgenin yalnızca tek ve net bir

defo sınıfını içermesi sağlanmıştır. Her bir imge tek tek kontrol edilerek kırpılmış ve her örüntüde yalnızca tek bir defo sınıfı kalacak şekilde düzenlenmiştir (şekil 3.8(b) ve şekil 3.8(c)).

3.3 Derin Öğrenme ve Temel Kavramlar

Bu bölümde, dokusuz kumaş görüntülerinde defo tespiti ve sınıflandırması için kullanılan derin öğrenme modellerinin temel kavramları ve teorik altyapısı ele alınmıştır. Derin öğrenme, Yapay Sinir Ağları (Artificial Neural Networks, ANN) tabanlı bir makine öğrenimi alt dalı olup, özellikle büyük veri setleriyle karmaşık desenleri öğrenme kapasitesiyle dikkat çeker [1]. Bu çalışmada kullanılan TL yöntemlerinin temelini oluşturan derin öğrenme modelleri, hiperparametreler, aktivasyon fonksiyonları ve optimizasyon algoritmaları detaylı bir şekilde açıklanmıştır.

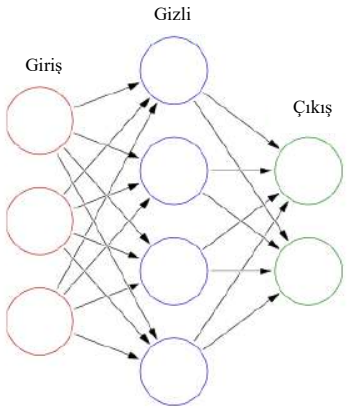
3.3.1 Derin öğrenme modelleri

Derin öğrenme modelleri, çok katmanlı yapılar aracılığıyla karmaşık veri desenlerini öğrenmek için tasarlanır. Bu modeller, özellikle görüntü işleme ve sınıflandırma görevlerinde, geleneksel yöntemlere kıyasla üstün performans gösterir [13]. Dokusuz kumaş görüntülerinde defo tespiti gibi endüstriyel uygulamalarda, derin öğrenme modelleri hem hızlı hem de yüksek doğruluklu sonuçlar sunduğu litratürde görülür [20].

3.3.2 ANN

ANN, insan beynindeki nöronlardan esinlenerek tasarlanmış matematiksel modellerdir. ANN, giriş katmanı, gizli katmanlar ve çıkış katmanından oluşan bir yapıya sahiptir [13]. Her bir katman, önceki katmandan gelen bilgiyi işler ve ağırlıklar aracılığıyla bir sonraki katmana aktarır. Bu çalışma kapsamında, ANN'nin çeşitli türleri kullanılmıştır.

Bir sinir ağındaki "nöron benzeri" birimler düğümlerdir. Sinirsel hesaplama, sınıflandırma, optimizasyon, kontrol teorisi ve regresyon problemlerini çözme gibi alanlarda kullanılmaktadır.



Şekil 3.9 Sinir ağları

ANN mimarisi, çok sayıda düğüm ve bunların arasındaki bağlantılardan oluşur ve örnek bir mimari şekil 3.9'da gösterilmiştir. Çoklu girişli bir nörona 'R' sayıda giriş vardır. Her bir giriş, ağırlık matrisi W'nin ilgili elemanları $w_{1,1}$ $w_{1,2}$ …, $w_{1,R}$ ile ağırlıklandırılır.

$$n = w_{1,1}p_1 + w_{1,2}p_2 + \cdots + w_{1,R}p_R + b \tag{3.1}$$

Bağlantı (3.1), matris formunda bağlantı (3.2)'de gösterildiği şekilde yazılabilir.

$$n = Wp + b \tag{3.2}$$

Nöronun gizli katmanın 1. Nöron için hesap yapılır ise giriş nöronlarından gelen veriler ağırlıklar ile çarpılır ve bağıntı (3.1)'de gösterildiği şekilde n değeri hesaplanır ise bağlantı (3.3)'de gösterildiği şekilde tanımlanır:

$$a = f(Wp + b) \tag{3.3}$$

Bağlantıda yer alan f() fonksiyonu, aktivasyon fonksiyonu alarak tanımlanır ve ANN'de farkılık türde fonksiyonlar kullanmılabilmektedir bu fonksiyonlardan bir tanesi, log-sigmoid transfer fonksiyonudur. Bu fonksiyon, girişi (artı ve eksi sonsuz arasında herhangi bir değer) alır ve 0 ile 1 arasında sonuçlar üretir, bağlantı (3.4)'de göserildiği şekilde hesaplanır:

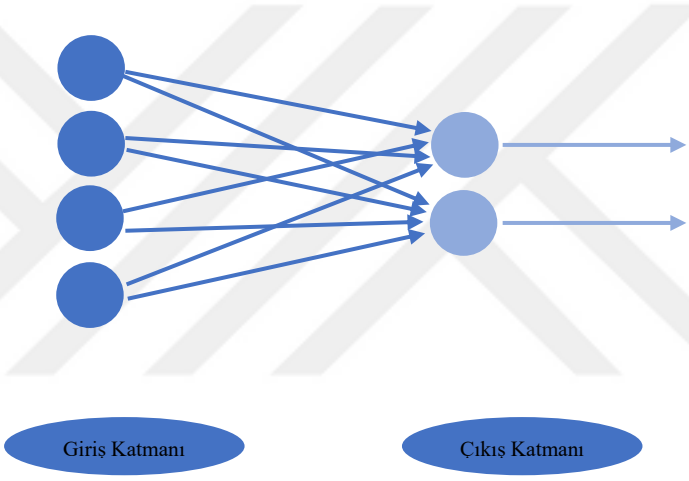
$$y = \log \mathrm{sig}(n) = \frac{1}{1 + e^{-n}} \tag{3.4}$$

Belirli bir aşamada, düğümler bir "katman" oluşturur. İlk katman giriş katmanı, son katman ise çıkış katmanı olarak adlandırılır. Giriş ve çıkış katmanları arasında yer alan katmanlar gizli katman olarak bilinir. ANN'nın performansı, gizli katmanların sayısının artmasıyla iyileşir. Her bir düğüm, ağdaki tüm girişlerin toplamını yapar. Bir düğümün girişi, önceki düğümün çıktısıdır. Her nöronun basit modeli bir algılayıcıdır

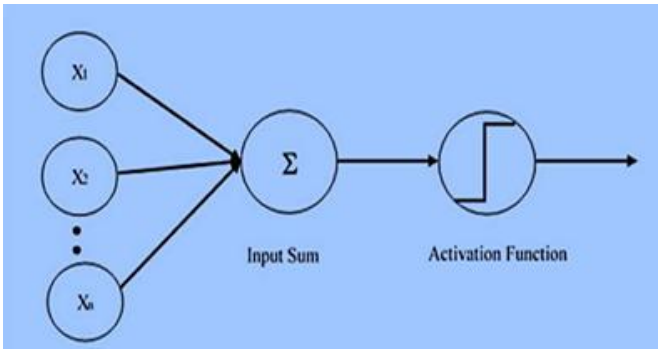
(perceptron). Bu model, giriş katmanı ve çıkış katmanından oluşan iki katmanlı bir yapıya sahiptir. En basit nöron, yani algılayıcı, doğrusal olarak ayrılabilir bilgileri çözmek için kullanılır. Bilgi doğrusal olarak ayrılamıyorsa, geri yayılım (back-propagation) yöntemi kullanılır [26].

a) Tek Katmanlı Algılayıcılar (Single-Layer Perceptron, SLP)

Tek katmanlı algılayıcı (SLP), ANN'nin ilk ve en temel modelidir. İleri beslemeli sinir ağı bunun başka bir adıdır [27]. SLP'nin çalışması için temel görevi ise düğümler arasındaki eşik aktarımıdır. Bu, makine öğrenmesinde doğrusal problemleri çözmek için kullanılan en basit yapay sinir ağı türüdür. Tek katmanlı algılayıcılarda sadece girdi ve çıktı katmanları bulunur şekil 3.10 ve 3.11'de göstermektedir.



Şekil 3.10 Tek Katmanlı Algılayıcı Sınıflarında Giriş ve Çıkış Katmanları [27].



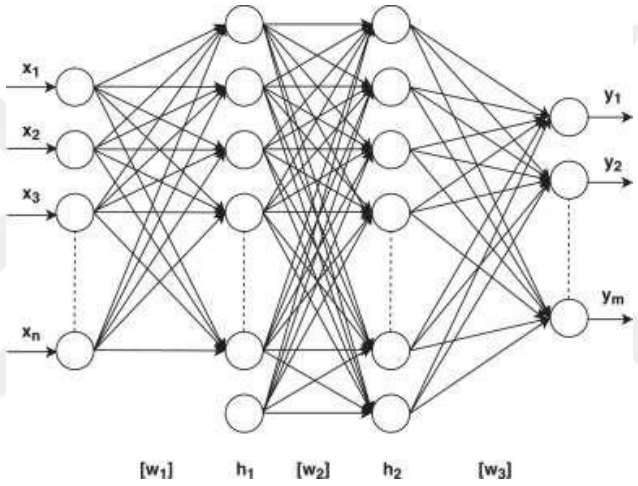
Şekil 3.11 Tek Katmanlı Algılayıcının Bileşenleri.[28]

b) Çok Katmanlı Algılayıcılar (Multi-Layer Perceptron, MLP)

Denetimli görevler için geleneksel olarak kullanılan çok katmanlı bir algılayıcı, MLP'dir; her bir katman düğümlerden oluşan birbirine bağlı çoklu katmanlardan oluşur. Ağırlık matrisleri, katman ara bağlantılarını ifade eder. Eğitim veri setindeki

sınıf etiketleri, çıktı katmanının düğümleri tarafından gösterilir. Ağırlık matrisleri rastgele başlar, ancak gradyan iniş tekniğini kullanarak tahmin edilen ve gerçek etiketler (kayıp fonksiyonu) arasındaki defonun yinelemeli minimizasyonu yoluyla güncellenir. Doğrusal olmayan dönüşümler yoluyla girdi verilerinin gizli temsillerini öğrenmek için girdi ve çıktı katmanları arasındaki ağırlık matrisleri kullanılır.

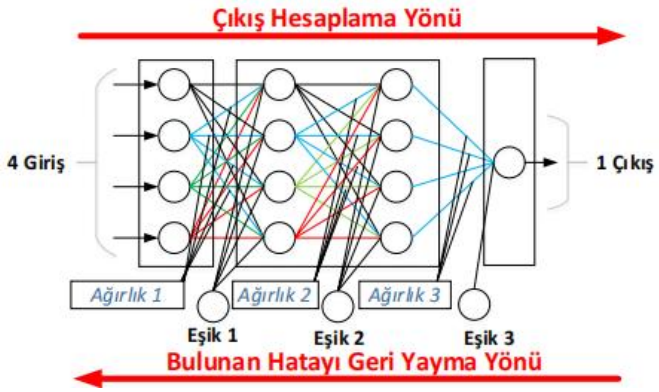
Temel dağılımlar bu katmanlarda tanımlanır ve çıktı katmanında sınıfları gösteren bir düğümle eşleştirilir. Gizli katmanlar özellik çıkarımına yardımcı olur. Çok katmanlı algılayıcı yapısı şekil 3.12'de gösterilmiştir [29].



Şekil 3.12 Çok katmanlı algılayıcı modeli- MLP

c) İleri ve Geri Beslemeli Yapay Sinir Ağları (Feed / Back Forward Neural Networks)

Verinin yalnızca ileri yönde aktığı bir yapıda olan İleri Beslemeli Sinir Ağı (FFNN), sınıflandırma ve regresyon problemleri için kullanılır. Biyolojik sinir hücrelerinden ilham alınarak tasarlanmış bir nöron, FFNN'nin ana işlevsel bileşenidir. N-boyutlu bir giriş vektörüne karşılık her nöron tek bir çıkış sinyali üretir. Sinir ağları, nöronlar arasındaki bağlantı değerlerini değiştirerek belirli işlevleri gerçekleştirmek için eğitilir. Bu bağlantılar bir döngü oluşturmaz ve bağlantı katsayıları ağırlıklar olarak bilinir [30].

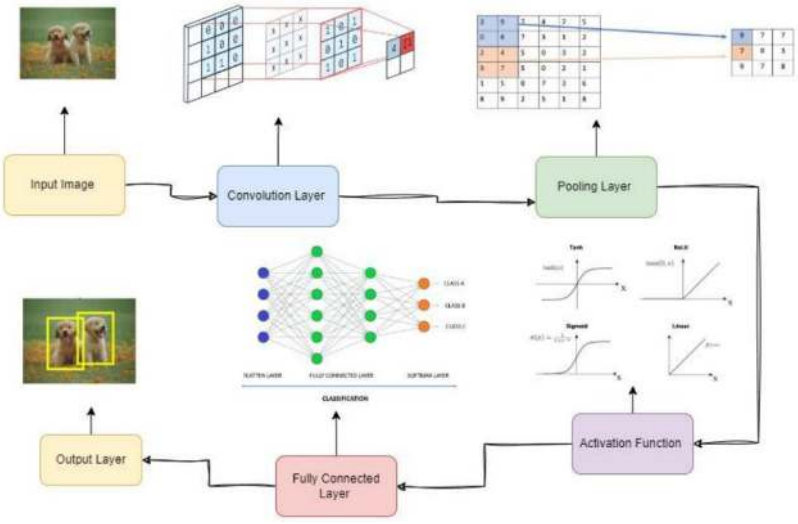


Şekil 3.13 Geri beslemeli yapay sinir ağı [31].

Geri beslemeli yapay sinir ağları (Back-Forward Neural Networks, BFNN), ağın tüm çıkışlarından elde edilen defonun, çıkış katmanından giriş katmanına doğru ağırlıklarının güncellenmesi yoluyla giriş katmanına aktarılması prensibine dayanır. Doğrusal olmayan sınıflama, eğri uydurma ve tahminleme çalışmalarında BFNN yapısı, uygun çözümler üretebilmektedir. Şekil 3.13'te üç ara katmanlı geri beslemeli örnek bir ANN modeli gösterilmiştir. Bu yapıda çıkıştan elde edilen defonun, ağırlıklara yansıtılması için bir geri yayılım algoritması kullanılır.

d) CNN'nin Temel İlkeleri

CNN yönteminin öğrenilebilmesi için, CNN'yi oluşturan bileşenlere ilişkin fonksiyonlarının bilinmesi oldukça önemlidir. Şekil 3.14'da CNN'nin farklı bileşenleri gösterilmektedir.



Şekil 3.14 CNN Bileşenleri. [32]

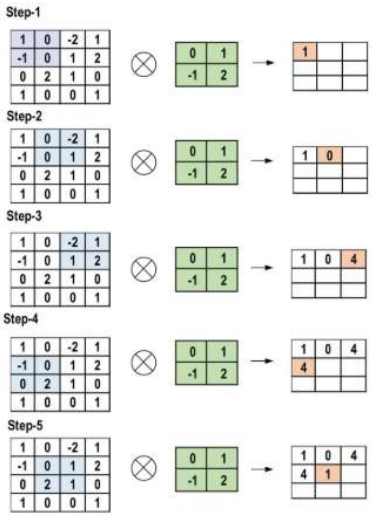
❖ CNN Katmanları

Bir CNN tipik olarak dört tür katmandan oluşur:

- Evrişim (Convolutional);
- Havuzlama (Pooling);
- Aktivasyon Fonksiyonu (Function of Activation);
- Tam Bağlantılı Birim (Fully Connected, FC).

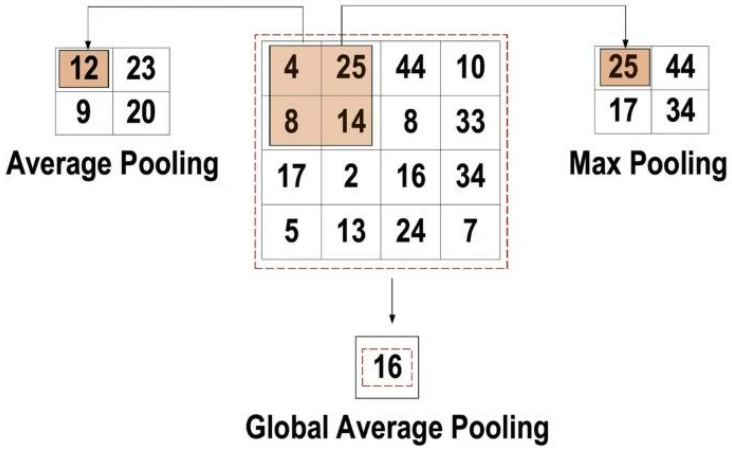
1. Evrişimli Katman: Evrişim filtreleri veya çekirdeklerden oluşan evrişimli katman, CNN mimarisinin temel unsurudur. Çıkış özellik haritası oluşturmak için giriş görüntüsü bu filtrelerle birleştirilir.

- Bir çekirdek, ayrık değerlerden oluşan bir ızgaradır ve her değerin çekirdek ağırlığı olarak adlandırılır. Eğitimin başlangıcında ağırlıklar rastgele atanır ve çeşitli başlatma teknikleri kullanılabilir. Eğitim sürecinde ağırlıkları ayarlamak, önemli özelliklerin çıkarılmasına yardımcı olur.
- Evrişim İşlemi: CNN, çok kanallı görüntüleri (örneğin, RGB üç kanal veya gri tonlamalı tek kanal) alır. Örneğin, 4x4 gri tonlamalı bir görüntü üzerinden işlem yapılır ve 2x2 rastgele ağırlıklı bir çekirdek üzerinden işlem yapılır. Görüntü üzerinde kaydırarak çekirdeğin nokta çarpımını hesaplamak mümkündür. Bunun için, karşılık gelen değerler çarpılır ve toplanarak bir skaler değer elde edilir. Bu, tüm görüntü taranana kadar tekrarlanır ve özellik haritasını oluşturur. Şekil 3.15, bu süreci grafiksel olarak gösterir; 2x2 çekirdek (açık yeşil) ve giriş görüntüsü (açık mavi) çarpılır, sonuç (açık turuncu) özellik haritasına eklenir.



Şekil 3.15 Evrişimli katmanın her adımında yapılan temel hesaplamalar [33].

2. Havuzlama Katmanı: Özellik haritalarının alt örneklemesi, havuzlama katmanının temel görevidir. Evrişimli işlemler bu haritayı oluşturur. Başka bir deyişle, bu yöntem, büyük boyutlu özellik haritalarını küçülterek daha küçük özellik haritaları oluşturur. Havuzlama aşamasının her adımında, aynı zamanda baskın bilgilerin veya özelliklerin çoğunu korur. Havuzlama işlemi, evrişimli işleme benzer şekilde başlangıçta hem adım hem de çekirdek boyutlarını belirler. Havuzlama için çeşitli katmanlar mevcuttur. Ağaç havuzu, geçitli havuzlama, ortalama havuzlama, minimum havuzlama, maksimum havuzlama, genel ortalama havuzlama (GAP) ve genel maksimum havuzlama bu yöntemlerden bazılarıdır. Max, min ve GAP havuzlama yöntemleri en yaygın ve popüler olanlardır. Bu üç havuzlama işlemi Şekil 3.16'da gösterilmektedir.



Şekil 3.16 Üç havuzlama işlemi

3. Aktivasyon Fonksiyonu: Girdiyi çıkışa eşlemek, tüm sinir ağı türlerindeki tüm aktivasyon fonksiyonu türlerinin temel işlevidir. Nöron girişinin ağırlıklı toplamının yanlılığı (varsa) hesaplanarak giriş değeri bulunur. Bu, aktivasyon fonksiyonunun, karşılık gelen çıktıyı üreterek belirli bir girdiye atıfta bulunarak bir nöronun ateşlenip ateşlenmeyeceğini belirlediği anlamına gelir.

   CNN mimarisinde, doğrusal olmayan aktivasyon katmanları, ağırlıkları olan tüm katmanlardan sonra, (FC katmanları ve evrişimli katmanlar gibi) kullanılır. Aktivasyon katmanlarının bu doğrusal olmayan performansı, girdinin çıktıya eşlenmesinin doğrusal olmayacağı anlamına gelir. Sonuç olarak, bu katmanlar CNN'ye daha karmaşık şeyler öğrenme yeteneği verir. Aktivasyon işlevi, ağı eğitmek için defo geri yayılımının kullanılmasına izin verdiği için ayırt etme yeteneğine de sahip olmalıdır. Bu son derece önemli bir özelliktir.

- Sigmoid: Bu aktivasyon fonksiyonunun girişi gerçek sayılardır, çıktı ise sıfır ile bir arasında sınırlıdır. Sigmoid fonksiyon eğrisi S şeklindedir ve matematiksel olarak denklem 3.5 ile temsil edilebilir.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.5}$$

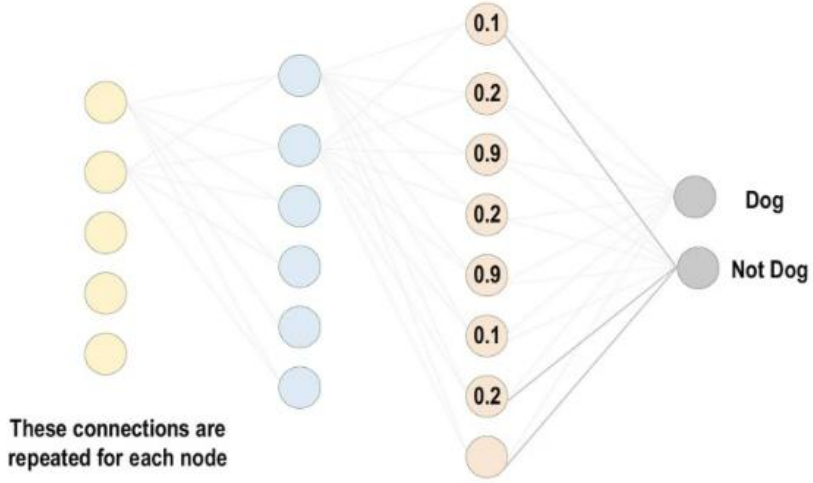
- Tanh: Sigmoid fonksiyonuna benzer, çünkü girdisi gerçek sayılardır, ancak çıktı -1 ile 1 arasında sınırlıdır. Matematiksel gösterimi Denklem 3.6'tedir.

$$f(x)_{tanh} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.6}$$

- ReLU: CNN bağlamında en yaygın kullanılan işlev. Girişin tüm değerlerini pozitif sayılara dönüştürür. Daha düşük hesaplama yükü, ReLU'nun diğerlerine göre ana avantajıdır. Matematiksel gösterimi Denklem 3.7'dedir.

$$f(x)_{\text{ReLU}} = \max(0, x) \tag{3.7}$$

4. Tam Bağlantılı Birim (FC): Her CNN mimarisinin sonunda bu katman bulunur. FC yaklaşım olarak bilinen önceki katmanın tüm nöronlarına bu katmanın her nöronu bağlanır. CNN, sınıflandırma için kullanılır. Bir tür ileri beslemeli ANN olduğu için, geleneksel çok katmanlı algılayıcı sinir ağlarının temel prensiplerine uygun olarak çalışır. Son havuzlama veya evrişimli katman, FC katmanının girişini oluşturur. Düzleştirme işleminin ardından bu girdi, özellik haritalarından oluşturulan bir vektör biçimindedir. Şekil 3.17, FC katmanının çıktısını nihai CNN çıktısını gösterir[34].



Şekil 3.17 Tam Bağlı Katman[33].

❖ CNN Kayıp Fonksiyonları

Kayıp fonksiyonları, CNN mimarisinin son katmanı olan çıktı katmanında, modelin eğitim örnekleri üzerindeki tahmin hatasını hesaplamak için kullanılır. Bu hata, gerçek çıktı ile tahmin edilen çıktı arasındaki farkı gösterir ve CNN öğrenme süreciyle optimize edilir. Kayıp fonksiyonu, hatayı hesaplamak için iki parametre kullanır: CNN'in tahmini çıktısı ve gerçek çıktı. Farklı problem türleri için çeşitli kayıp fonksiyonları kullanılır. Aşağıda bazı kayıp fonksiyonu türleri kısaca açıklanmıştır.

(a) Çapraz Entropi veya Softmax Kayıp Fonksiyonu

Çıktı sınıfı olasılığının matematiksel ifadesi (denklem 3.8)'de gösterilmiştir:

$$P(y_i) = \frac{e^{z_i}}{\sum_{i=1}^{N} e^{z_i}} \tag{3.8}$$

Burada, z_i önceki katmandan gelen normalize edilmemiş çıktıyı, N ise çıktı katmanındaki nöron sayısını temsil eder.

Çapraz entropi kayıp fonksiyonunun matematiksel ifadesi (denklem 3.9)'da gösterilmiştir:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y_i}) + (1 - y_i)\log(1 - \hat{y_i})] \tag{3.5}$$

Burada, y_i gerçek etiketi, $\hat{y_i}$ ise tahmin edilen olasılığı gösterir.

(b) Kare Hata Kayıp Fonksiyonu

Tahmin edilen kare hata kaybının matematiksel ifadesi (denklem 3.10):

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2 \tag{3.6}$$

Burada, y_i gerçek çıktıyı , $\hat{y_i}$ tahmin edilen çıktıyı, N ise örnek sayısını temsil eder.

(c) Kenar Boşluğu Kayıp Fonksiyonu

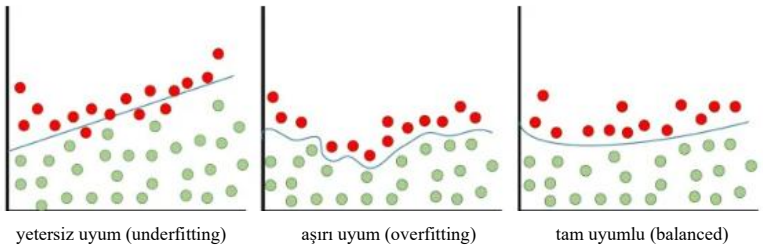
Kenar boşluğu kayıp fonksiyonunun matematiksel formülü (denklem 3.11)'de gösterilmiştir:

$$L = \sum_{i=1}^{N} \max(0, m - y_i \cdot \hat{y_i}) \tag{3.7}$$

Burada, y_i istenen çıktıyı, $\hat{y_i}$ tahmin edilen çıktıyı, m kenar boşluğunu (genellikle 1) ve N örnek sayısını temsil eder.

❖ CNN'nin Düzenlenmesi

Aşırı uyum (over-fitting), CNN modellerinde iyi genelleştirmenin ana sorunudur. Model, özellikle eğitim verileri üzerinde iyi çalışıp test verileri (görülmemiş veriler) üzerinde başarısız olduğunda çok uyumlu hale gelir. Bu durum sonraki bölümlerde daha ayrıntılı olarak tartışılmıştır. Aşırı uyumun tersi, yetersiz uyum (under-fitting) durumudur; bu durumda model, eğitim verilerinden yeterli öğrenme sağlayamaz. Model, eğitim ve test verilerinde iyi performans gösterirse "tam uyumlu"dur. Şekil 3.20 bu üç kategoriyi göstermektedir. Düzenlileştirme, aşırı uyumu önlemek için kullanılır.



Şekil 3.18. Aşırı uyum ve yetersiz uyum sorunları [36]

1. Rastgele nöronların devre dışı bırakılması (Dropout)

Bu, genelleştirme için yaygın bir yöntemdir. Nöronlar her eğitim döneminde (epoch) rastgele devre dışı bırakılır. Bu yaklaşım, özellik seçme yeteneğini tüm nöron grubu arasında eşit bir şekilde dağıtarak modeli çeşitli bağımsız özellikleri öğrenmeye zorlar. Eğitim sürecinde devre dışı bırakılan nöronlar geri yayılım veya ileri yayılım süreçlerine katılmaz. Bununla birlikte, test sürecinde tam ölçekli ağ tahmininde kullanılır [35][36].

2. Rastgele bağlantıların pasıy edilmesi (Drop-Weights)

Bu yöntem, ayrılmaya çok benzer. Her eğitim döneminde, nöronlar yerine nöronlar arasındaki bağlantılar (ağırlıklar) devre dışı bırakılır. Bu, düşen ağırlıklarla düşen ağırlıklar arasındaki tek farktır.

3. Veri Artırma (Data Augmentation)

Modeli çok sayıda veri üzerinde eğitmek, aşırı uyumdan kaçınmanın en basit yoludur. Bunu yapmak için veri artırma kullanılır. Eğitim veri setini yapay olarak büyütmek için birçok yöntem vardır [37][38].

4. Yığın Normalizasyonu (Batch Normalization)

Bu yaklaşım, çıktı aktivasyonlarının başarısını garanti eder [81]. Bu performans, birimlerin normal dağılımını temsil eder. Her katmanın çıktısını normalleştirmek için

ortalama hesaplanır ve standart sapmaya bölünür. Bu, ağdaki her katmanda ön işleme görevi olarak düşünülebilir ve diğer ağlarla farklılaştırılabilir ve entegre edilebilir. Aktivasyon katmanlarının iç kovaryans kayması (internal covariate shift) sorununu da azaltır. İç kovaryans kayması, her katmandaki aktivasyon dağılımındaki değişiklikten kaynaklanır. Eğitim verilerinin çeşitli kaynaklardan (örneğin gündüz ve gece görüntüleri) toplanması durumunda, sürekli ağırlık güncellemeleri nedeniyle bu kayma oldukça yüksek olabilir. Bu nedenle, modelin yakınsaması için daha fazla zaman gerekir, bu da eğitim süresini uzatır. Bu sorunu çözmek için CNN'nin mimarisine yığın (batch) normalizasyonu işlemini gerçekleştiren bir katman dahil edilir [39][40].

Yığın Normalizasyonunun Avantajları Şunlardır:

- Gradyan kaybolma (vanishing gradient) sorununu önler.
- Kötü ağırlık başlatma (weight initialization) sorununu etkili bir şekilde kontrol eder.
- Ağ yakınsama süresini önemli ölçüde azaltır (büyük ölçekli veri setlerinde bu son derece faydalıdır).
- Hiperparametreler üzerindeki eğitim bağımlılığını azaltmakta zorlanır.
- Düzenlileştirme üzerinde küçük bir etkisi olduğu için aşırı uyum olasılığını azaltır.

❖ Optimizasyon Algoritması (Optimizer)

Yukarıdaki bölüm, CNN mimarisindeki veri akışını açıklamaktadır. Ağın eğitiminin, gradyan güncellemesinin temel adımına dayandığını, amaç fonksiyonu veya kayıp fonksiyonu gradyanını hesaplamak için ağ parametrelerine göre birinci dereceden bir türev kullanmaktır. Ardından, her bir ağ katmanının öğrenme parametrelerinin güncellenmesini sağlamak için kısmi diferansiyel hesaplama yoluyla gradyan bilgileri önceki ağ katmanına aktarılır. Optimize edicinin amacı, gradyan güncellemesini daha makul hale getirmenin bir yolunu sağlamaktır. Makroskopik performans, tüm ağın daha hızlanmasını, daha düşük yerel optimal değeri (yani daha az kayıp), daha ucuz hesaplamayı ve diğer faktörleri tahmin etmesini sağlamaktır. Tablo 3.3, yöntemler ve özellikleri içeren yaygın olarak kullanılan birkaç optimize ediciyi özetlemektedir.

Tablo 3.3 CNN Modeli İçin Farklı Optimizasyon Algoritmaları

Optimizasyon Algoriması	Yöntemi	Avantajı	Dezavantajı
Adaptive Moment Estimation (Adam)	Modeldeki her parametre için uyarlanabilir bir öğrenme oranı hesaplar	1-Seyrek verilerde yakınsama oranını artırır. 2- Daha hızlı özellik ayırma sağlar. 3- Momentum, AdaGrad ve RMSprop'un avantajlarını birleştirir	1- Kötü genelleştirme ve keskin minimumlara yakınsama olasılığı yüksektir.
Stochastic Gradient Descent (SGD)	Eğitim örneklerinin bir kısmını rastgele seçerek örnekler verir.	1- Eğitimden sonra yeterli genelleştirme sağlar.	1- Yerel veya eyer noktalarına (saddle points) yatkınlık gösterir. 2- Başlangıç hiperparametrelerine duyarlıdır ve öğrenme oranının ayarlanması zordur. 3- Hiperparametre başlatılmasına karşı hassastır.
Batch Gradient Descent (BGD)	Tüm eğitim örneklerinin gradyanını hesaplar ve ardından parametreleri güncellemek için bu gradyanı kullanır.	1. Seyrek verilerde yakınsama oranını artırır. 2. BGD kullanarak ekstra kararlı bir gradyan oluşturur.	1. Genellikle büyük bir eğitim veri kümesi için uygun değildir 2. Önemli miktarda kaynak gerektirir.
AdaGrad	Her adımda öğrenme katsayısını günceller.	1-Öğrenme oranı uyarlanabilir şekilde ayarlanır. 2- Seyrek özelliklerle verilerde yakınsama oranı daha hızlı	1- Keskin minimumlara (sharp minima) yatkınlık gösterir ve yakınsama olasılığı yüksektir. 2- Gradyentlerin agresif ölçeklenmesi nedeniyle kaybolma (vanishing gradient) sorunu yaşanabilir.
Mini-batch Gradient Descent	Eğitim örneklerini birkaç küçük partiye böler ve ardından gradyan hesaplandıktan sonra parametreleri günceller.	1- Yakınsama hem güvenilir hem de etkilidir. 2- Ekstra bellek etkinliği, SGD'nin teknik avantajlarından biridir.	Kayıp değeri yüksek çıkabilir ve gürültüye eğilimlidir.

❖ Hiperparametre (Hyperparameter)

Aktivasyon fonksiyonu, kayıp fonksiyonu ve optimize edici seçimi gibi, CNN modellerinin performansını önemli ölçüde etkileyen diğer hiperparametrelerin de dikkatle ayarlanması gerekir. Bilindiği gibi, her durum için en iyi çözümü garanti

edebilecek sabit bir hiperparametre seti yoktur. Sonuç olarak, hiperparametrelerin belirlenmesinde rehber ilkeler ve deneysel bir yaklaşım önemli bir rol oynamaktadır. Bazı yaygın hiperparametrelerin model performansına etkileri Tablo 4'te gösterilmektedir.

Tablo 3.4 CNN Modellerinde Yaygın Hiperparametreleri

Hiperparametre	Tanım	Strateji	Açıklama
Öğrenme Oranı (Learning Rate - LR)	LR, ağın ağırlıklarını güncellemek için kullanılan adım büyüklüğünü belirler. Değişken olabilir; genellikle küçük bir LR tercih edilir. Çok küçük bir LR yakınsamayı yavaşlatırken, çok büyük bir LR ise yakınsamayı kararsız hale getirebilir.	Genellikle, LR'nin başlangıç değeri deneysel olarak belirlenir. LR, eğitim sırasında kademeli olarak azaltılabilir; örneğin, başlangıçta büyük bir LR ile hızlı eğitim yapılır ve daha sonra küçük bir LR ile ince ayar gerçekleştirilir.	LR'yi ayarlamak için bir dizi deneme yapılır. İlk değer, optimal yakınsamayı sağlayacak şekilde seçilir.
Epochs (İterasyon Sayısı)	Epoch sayısı , tüm eğitim verisinin ağ üzerinden bir kez geçirildiği eğitim döngülerinin sayısını ifade eder. Epoch Sayısı, modelin veri üzerindeki öğrenme derinliğini belirler; az epoch sayısı yetersiz öğrenmeye, çok epoch sayısı ise aşırı öğrenmeye yol açabilir.	Epoch sayısı, hesaplama kaynaklarına bağlı olarak mümkün olduğunca büyük seçilebilir. Ancak, modelin doğrulama performansı izlenerek aşırı öğrenme riski önlenmelidir; genellikle 50-500 arasında bir aralıkta belirlenir.	Epoch Sayısı'nın optimizasyonu, eğitim süresini ve genelleştirme performansını etkiler. Optimal epoch sayısı, doğrulama seti üzerindeki sonuçlara göre ayarlanmalıdır.
Image boyutu	İmge boyutu, modelin işleyeceği giriş görüntülerinin piksel boyutunu (örneğin, 224x224, 640x640) ifade eder. Image boyutu, modelin hesaplama karmaşıklığını ve özellik ekstraksiyonunu etkiler.	IMGSZ, donanım kapasitesine ve modelin gereksinimlerine göre seçilir. Daha büyük IMGSZ, daha fazla detayı yakalar ancak hesaplama maliyetini artırır; genellikle 256x256 veya 640x640 gibi standart boyutlar tercih edilir.	Image boyutunun doğru seçimi, modelin hem hızını hem de doğruluğunu dengeler. Veri setine ve donanım kaynaklarına bağlı olarak optimize edilmelidir.
Batch (yığın Boyutu - BATCH)	Yığın boyutu (BATCH), her eğitim iterasyonunda işlenen örneklerin sayısını ifade eder. BATCH, gradyan güncellemelerinin stabilitesini ve eğitim hızını etkiler; küçük BATCH gürültülü gradyanlara, büyük	BATCH, genellikle 8, 16, 32 gibi güçler şeklinde seçilir ve GPU belleğine göre ayarlanır. Büyük veri setlerinde 64 veya 128 gibi değerler kullanılabilir.	BATCH'in seçimi, modelin yakınsama hızını ve genelleştirme performansını etkiler. Optimizasyon algoritmasına ve veri setinin büyüklüğüne göre belirlenmelidir.

	BATCH ise daha stabil gradyanlara yol açar.		
Workers (İşçi Sayısı)	Workers, veri yükleme ve ön işleme için kullanılan paralel iş parçacıklarının sayısını ifade eder. Workers, eğitim sürecindeki veri işleme hızını artırır.	Workers, CPU çekirdek sayısına bağlı olarak 2 ile 16 arasında seçilebilir. Çok yüksek workers, sistem kaynaklarını tüketebilir ve kararsızlığa yol açabilir.	Workers, büyük veri setlerinde eğitim süresini kısaltır, ancak donanım kapasitesine uygun seçilmelidir.
Initial Learning Rate (Başlangıç Öğrenme Oranı - LR0)	Başlangıç öğrenme oranı (LR0), eğitimin ilk aşamasında kullanılan LR değeridir. LR0, modelin başlangıçtaki öğrenme hızını belirler ve eğitim boyunca dinamik olarak ayarlanabilir.	LR0 genellikle 0.001 ile 0.1 arasında seçilir ve transfer öğrenimi için daha küçük değerler (örneğin, 0.0001) tercih edilir.	LR0'nin doğru ayarlanması, modelin hızlı ve stabil bir şekilde yakınsaması için kritiktir.
Final Learning Rate (Son Öğrenme Oranı - LRF)	Son öğrenme oranı (LRF), eğitimin son aşamalarında kullanılan LR değeridir. LRF, öğrenme sürecinin ince ayarlama aşamasında stabilite sağlar.	LRF, LR0'dan çok daha küçük bir değer (örneğin, 0.00001) olarak ayarlanır ve genellikle öğrenme oranı çürümesi (learning rate decay) ile belirlenir.	LRF, modelin yerel minimumlara takılmasını önler ve genelleştirme performansını artırır.
Momentum	Momentum, gradyan inişi algoritmasında hareket yönünü koruma katsayısını ifade eder. Momentum, gradyanın yönünü stabilize ederek yakınsamayı hızlandırır.	Momentum genellikle 0.9 ile 0.99 arasında seçilir. Optimizasyon algoritmasına (örneğin, SGD) bağlı olarak ayarlanır.	Momentum, özellikle düz alanlarda yakınsamayı hızlandırır ve yerel minimumlardan kaçınmayı kolaylaştırır.
Weight Decay (Ağırlık azalması - WD)	WD, modelin ağırlıklarının büyümesini sınırlamak için uygulanan bir düzenlileştirme tekniğidir. WD, aşırı öğrenmeyi önler.	WD genellikle 0.0001 ile 0.01 arasında küçük bir değer alır ve optimizasyon algoritmasıyla uyumlu olmalıdır.	WD, modelin genelleştirme yeteneğini artırır ve karmaşık ağırlık yapılarını caydırır.
Optimizer (Optimizasyon Algoritması)	Optimizer, kayıp fonksiyonunu minimize etmek için kullanılan yöntemdir (örneğin, SGD, Adam, RMSprop). Optimizer, modelin öğrenme dinamiklerini belirler.	Optimizer, veri setinin özelliklerine ve modelin karmaşıklığına göre seçilir. Adam, seyrek verilerde tercih edilirken, SGD daha basit modeller için uygundur.	Optimizer'nin doğru seçimi, yakınsama hızını ve model performansını doğrudan etkiler.
Patience (Sabır)	Sabır (PAT), erken durdurma (early stopping) mekanizmasında modelin doğrulama kaybının iyileşmesini bekleme epoch sayısını ifade eder. PAT, aşırı öğrenmeyi önler.	PAT genellikle 10 ile 50 epoch arasında seçilir ve doğrulama kaybı sabit kaldığında eğitimi durdurur.	PAT, modelin gereksiz yere uzun süre eğitilmesini önler ve hesaplama kaynaklarını optimize eder.

3.4 Kullanılan Modeller ve Yöntemler

Bu çalışmada, dokusuz kumaşlardaki defoların otomatik olarak tespit edilmesi amacıyla iki farklı makine öğrenimi yaklaşımı izlenmiştir: TL ve OD. Sanayi üretim süreçlerinde, özellikle tekstil sektöründe, dokusuz kumaşların kalitesi son derece önemlidir. Üretim defolarından kaynaklanan defolar, ürünün performansını ve müşteri memnuniyetini olumsuz yönde etkileyebilir. Bu nedenle, bu defoların erken ve doğru bir şekilde tespit edilmesi büyük önem taşımaktadır. İşte bu noktada, TL ve OD yöntemleri, geleneksel manuel denetim yöntemlerine kıyasla daha hızlı, daha tutarlı ve daha verimli bir alternatif sunmaktadır.

Bu alt bölümde, bu iki ana yöntem ve bu yöntemler kapsamında kullanılan spesifik modeller, ilgili mimarileri, eğitim süreçleri ve elde edilen performans sonuçları detaylı bir şekilde incelenerek açıklanmıştır. Her bir yöntemin avantajları ve dezavantajları karşılaştırılarak, dokusuz kumaş defo tespiti için en uygun yaklaşımın belirlenmesine yönelik bir değerlendirme sunulacaktır. Ayrıca, bu yöntemlerin gelecekteki potansiyeli ve geliştirilebilme alanları da tartışılmıştır. Bu çalışmada, dokusuz kumaş endüstrisindeki kalite kontrol süreçlerini optimize etmek ve üretim verimliliğini artırmak için kullanılabilecek yenilikçi bir çözüm sunmayı hedefler.

3.4.1 TL model yaklaşımı

TL model yaklaşımı, önceden büyük veri setleri üzerinde eğitilmiş derin öğrenme modellerinin yeni bir görev için küçük verilerle değiştirilmesidir. Dokusuz kumaş defolarının tespiti gibi özel alanlarda bu yöntem tercih edilmiştir. Bu tez kapsamında kullanılan TL model yaklaşımları ve karşlaştırma analizi tablo 3.5'te gösterilmiştir. Bu çalışmada kullanılan TL modelleri ve ilgili bilgiler aşağıda belirtilmiştir.

Tablo 3.5 TL model yaklaşımları ve karşlaştırma analizi

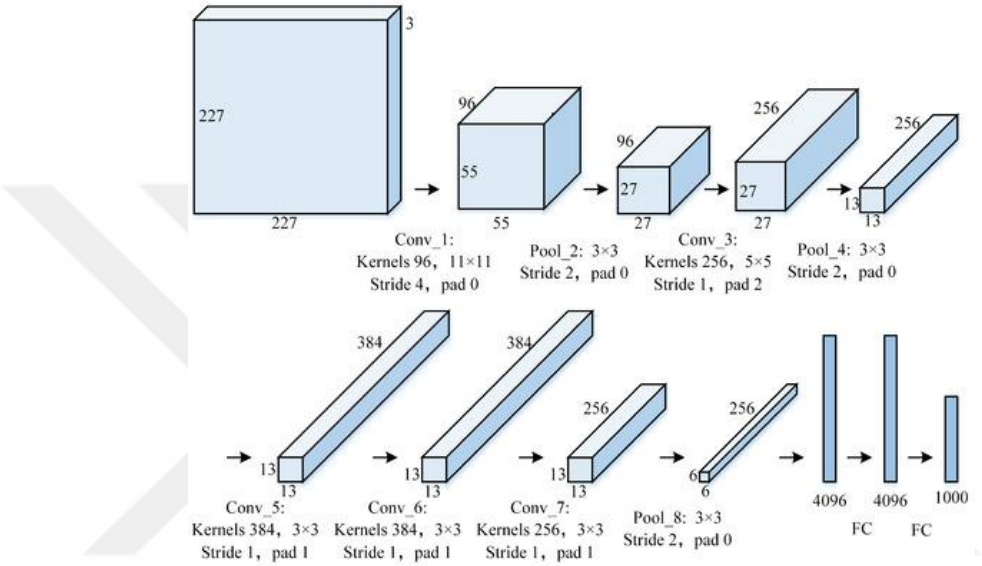
Model	Parametre Sayısı	ImageNet Top-1 Doğruluk	Avantajlar	Dezavantajlar
VGG16	~140 milyon	-	1. Basit ve homojen topoloji. 2. Küçük (3×3) filtrelerle verimlilik. 3. Derinlik avantajı.	1. Yüksek hesaplama maliyeti. 2. Aşırı parametre kullanımı
ResNet50	~25.6 milyon	-	1. Artık bağlantılarla gradyan akışı 2. Daha az parametreyle yüksek doğruluk 3. Derinlikte verimlilik	1. Derin yapının optimize edilmesi zor

EfficientNet-B0	~5.3 milyon	77.1%	1. Bileşik ölçeklendirme ile denge. 2. MBConv ile düşük hesaplama maliyeti. 3. Transfer öğrenimi için hızlı adaptasyon	1. Derin özellik ekstraksiyonunda sınırlamalar
DenseNet121	~8 milyon	75.0%	1. Yoğun bağlantılarla özellik yeniden kullanımı. 2. Gradyan kaybolma azalması. 3. Düşük parametre sayısı	1. Yüksek bellek kullanımı. 2. Eğitim süresi uzun.
MobileNetV2	~3.5 milyon	72%	1. Düşük hesaplama maliyeti (300M MACS). 2. Gerçek zamanlı uygulamalara uygunluk. 3. Hafif yapı.	1. Karmaşık desenlerde hassasiyet düşük. 2. Derinlik sınırlaması
InceptionV3	~24 milyon	78.8%	1. Farklı ölçeklerde özellik öğrenme. 2. 1×1 evrişimlerle hesaplama optimizasyonu	1. Yüksek hesaplama ve bellek maliyeti. 2. Eğitim süresi uzun

a) VGG16

Mimari: Simonyan ve Zisserman, görüntü tanıma alanında CNN'nin etkili olduğunu belirledikten sonra basit ve verimli bir CNN tasarım ilkesini önerdiler. Görsel geometri grubu (Visual geometry group, VGG16) bu yaratıcı tasarımın adıdır. VGG16, ZefNet [41] ve AlexNet [42]'e kıyasla on dokuz katman daha fazla içeren çok katmanlı bir model [43] olan ağın temsili kapasitesinin derinlik ile ilişkisini temsil eder. Bununla birlikte, 2013-ILSVRC yarışmasında öne çıkan ağ olan ZefNet, küçük boyutlu filtrelerin CNN'nin performansını artırabileceğini belirtti. Bu bulgulara dayanarak VGG16, ZefNet'te 7×7 ve 11×11 filtreler yerine küçük boyutlu (3×3) filtrelerden oluşan bir katman eklemiştir. Bir dizi deney, bu küçük boyutlu filtrelerin büyük boyutlu filtrelerle aynı etkiye sahip olabileceğini göstermiştir. Başka bir deyişle, bu küçük boyutlu filtreler, büyük boyutlu filtreler kadar verimli olduğu sonucuna verilmiştir. Parametre sayısını azaltarak hesaplama karmaşıklığını azaltmak da ek bir avantaj; Bu bulgular, CNN'de küçük boyutlu filtreler kullanmak için yeni bir araştırma trendi başlatmıştır. Ek olarak, VGG16, sonraki özellik haritalarının lineer bir gruplandırmasını öğrenmek ve ağ karmaşıklığını düzenlemek için 1×1 evrişimleri evrişim katmanlarının ortasına ekler. Ağ ayarında, evrişim katmanının ardından maksimum havuzlama katmanı veya havuzlama katmanı eklenmiş ve mekansal çözünürlüğü korumak için doldurma uygulanmıştır [44]. Genel olarak, VGG16, görüntü sınıflandırması ve yerelleştirme ile ilgili sorunları çözdü. 2014-ILSVRC yarışmasında birinci olamamasına rağmen, basitliği, homojen topolojisi ve artırılmış

derinliği sayesinde ün kazandı. Bununla birlikte, yaklaşık 140 milyon parametre kullanımı nedeniyle VGG16'nın hesaplama maliyeti aşırıdır, bu da onu temel eksikliği olarak öne çıkarıyor. Ağın yapısı Şekil 3.19'da gösterilmektedir. Şekilde, modelin katman yapısı detaylı bir şekilde gösterilmiş; giriş katmanından itibaren evrişim (convolution), maksimum havuzlama (max pooling, MP) ve tam bağlı FC katmanlarının sıralı düzeni açıkça belirtilmiştir. Bu yapı, VGG16'nın homojenlik ve derinlik avantajlarını vurgular.[45]

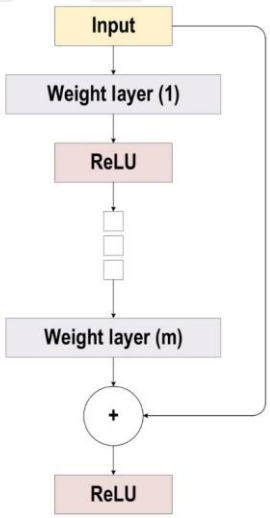


Şekil 3.19 VGG-16 Mimarisi

b) ResNet50

Mimari: ResNet50, 50 katmanlı bir CNN olup, Microsoft Research ekibi tarafından geliştirilmiştir . Bu model, He ve diğerleri tarafından 2016 yılında "Deep Residual Learning for Image Recognition" başlıklı makalede önerilen Artık Ağlar (Residual Networks - ResNet) ailesinin bir üyesidir [46]. Makale, ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2015'te üstün performans sergileyerek birinci olan ResNet mimarisini tanıtmış ve derin ağların eğitilmesinde karşılaşılan temel sorunlara yenilikçi bir çözüm sunmuştur. ResNet'in temel katkısı, atlama bağlantıları (skip connections) veya artık bağlantılar (residual connections) olarak adlandırılan bir yapıdır. Geleneksel CNN'lerde, ağ derinliği arttıkça gradyan kaybolma problemi nedeniyle eğitim performansı düşer ve modelin doğruluğu saturasyona uğrar. He ve diğerleri, bu sorunu çözmek için her katmanın girişini çıkışına ekleyen bir artık öğrenme (residual learning) yaklaşımı önermiştir. Matematiksel olarak, bir katmanın girişi x ve çıktısı $H(x)$ ise, ResNet mimarisi $F(x)=H(x)-x$ olarak artık fonksiyonu $F(x)$

öğrenir ve çıkış $H(x)=F(x)+x$ şeklinde hesaplanır. Bu yapı, gradyan akışını iyileştirerek derin ağların eğitilmesini mümkün kılar ve modelin genelleştirme kapasitesini artırır. ResNet50, beş evrişim bloğundan (conv blocks) oluşur: conv1, conv2, conv3, conv4 ve conv5. İlk katman (conv1), 7×7 filtrelerle 64 kanal oluştururken, maksimum havuzlama (max pooling) ile mekansal boyutlar azaltılır. Ardından, her blokta 3×3 filtreler ve bottleneck tasarımı (1×1, 3×3, 1×1 konvolüsyonlar) kullanılarak derinlik ve parametre verimliliği artırılmıştır. Bottleneck tasarımı, önce 1×1 konvolüsyonla kanal sayısını azaltır, ardından 3×3 konvolüsyonla işlem yapar ve son olarak 1×1 konvolüsyonla kanal sayısını artırır; bu, hesaplama maliyetini düşürürken derinliği korur. Toplamda, ResNet50 yaklaşık 25.6 milyon parametreye sahiptir ve son katman, ImageNet için 1000 sınıflı bir tam bağlı katman (fully connected layer) içerir. Bu çalışmada, son katman veri setindeki defo sınıflarına göre uyarlanmıştır. Şekil 3.20 , ResNet50'nin katman yapısını ve atlama bağlantılarını görselleştirmektedir. He ve diğerleri [46], ResNet mimarisinin, daha az parametreyle daha yüksek doğruluk elde ettiğini ve geleneksel ağlara kıyasla daha verimli olduğunu deneysel olarak göstermiştir.

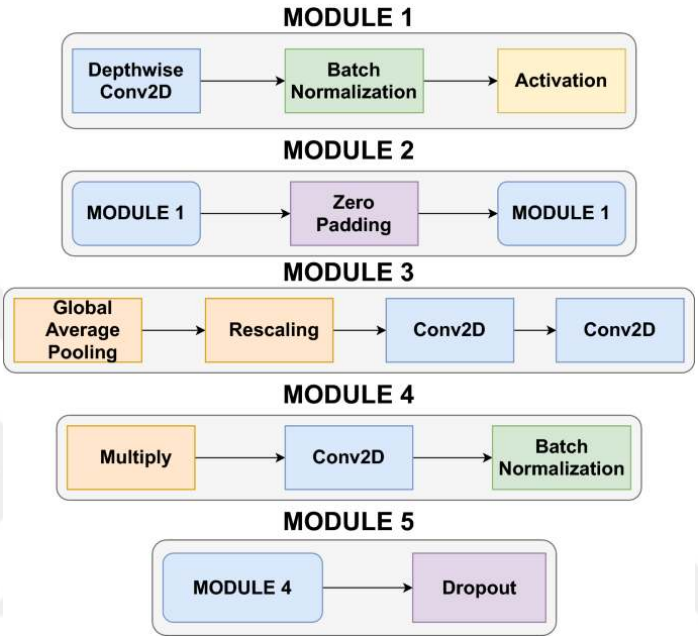


Şekil 3.20 ResNet Temel Çalışması [46]

c) EfficientNet-B0

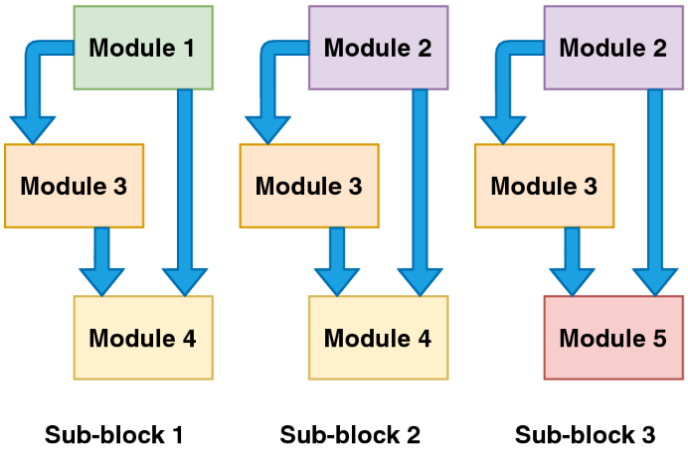
Mimari: EfficientNet, Google AI Araştırma ekibi tarafından geliştirilen bir Evrişimli Sinir Ağı CNN model ailesidir ve 2019'da Mingxing Tan ve Quoc V. Le tarafından "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" başlıklı makalede tanıtılmıştır [47]. EfficientNet ailesi, B0'dan B7'ye kadar sekiz farklı model içerir. Her yeni model daha fazla ayar ve daha iyi sonuçlar sunar. EfficientNet-B0, bu

ailenin temel modelidir ve bileşik ölçeklendirme (compound scaling) yöntemiyle derinlik, genişlik ve çözünürlük boyutlarını dengeli bir şekilde ölçeklendirir. EfficientNet-B0'nin temel yapı taşı, MobileNetV2'den türetilen ters bottleneck (inverted residual) blokları olan MBConv katmanlarıdır. Model, 7 ana bloktan ve toplamda 237 katmandan oluşur. Bu bloklar, 5 temel modülle oluşturulan alt bloklardan (sub-blocks) meydana gelir.



Şekil 3.21 EfficientNet-B0 Alt Bloklarının Modül Yapısı [48].

Şekil 3.21, bu alt blokların 5 temel modülle (Module 1-5) nasıl yapılandırıldığını gösterir: Module 1, alt blokların başlangıç bloğudur; Module 2, 1. blok hariç diğer ana blokların ilk alt bloğunu başlatır; Module 3, tüm alt bloklarda atlama bağlantısı (skip connection) olarak kullanılır; Module 4, ilk alt bloklardaki atlama bağlantılarını birleştirir; ve Module 5, her alt bloğu bir önceki alt bloğa atlama bağlantısıyla bağlar.

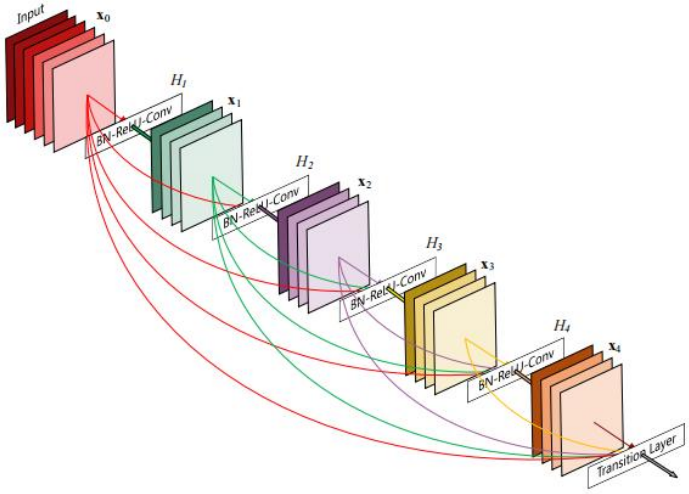


Şekil 3.22 EfficientNet-B0'nin MBConv Katmanlarının Mimari Yapısı [48].

Şekil 3.22, EfficientNet-B0'nin MBConv katmanlarının yapısını detaylandırır: MBConv bloğu, 1×1 konvolüsyonla kanal sayısını artırır, 3×3 veya 5×5 derinlikten ayrılmış evrişim (depthwise separable convolution) ile özellik çıkarımı yapar, sıkıştırma-uyarma (squeeze-and-excitation) modülü ile kanal bazlı dikkat mekanizması uygular ve 1×1 konvolüsyonla kanal sayısını azaltır. Her MBConv bloğunun son katmanında doğrusal aktivasyon (linear bottleneck) kullanılarak ReLU'nun bilgi kaybı önlenir. İlk katman (stem), 3×3 evrişimle 32 kanal üretir ve maksimum havuzlama ile boyutlar 112×112'ye düşürülür. Model, 224×224 piksel giriş boyutuna sahiptir, yaklaşık 5.3 milyon parametre içerir ve ImageNet'te 77.1% top-1 doğruluk oranı sunar. Bu çalışmada, son tam bağlı katman dokusuz kumaş defo sınıflarına uyarlanmıştır [48].

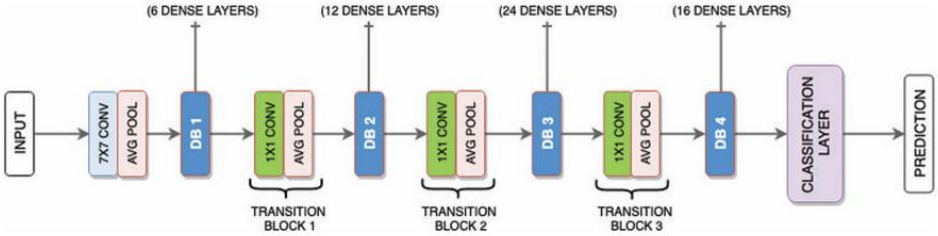
d) DenseNet121

Mimari: DenseNet121, Dense Convolutional Network (DenseNet) ailesinin bir üyesidir ve 2017 yılında Huang ve diğerleri tarafından "Densely Connected Convolutional Networks" başlıklı makalede tanıtılmıştır [35]. DenseNet, her katmanın önceki tüm katmanlarla doğrudan bağlantı kurduğu yoğun bağlantı (dense connectivity) yapısıyla öne çıkar. Bu yapı, özelliklerin yeniden kullanımını teşvik eder ve gradyan kaybolma (vanishing gradient) sorununu azaltır. DenseNet121, 121 katmandan oluşur ve 4 yoğun blok (dense block) içerir. Her yoğun blokta, katmanlar arasında yoğun bağlantılar bulunur; yani her katman, önceki tüm katmanların özellik haritalarını (feature maps) alır ve kendi özellik haritalarını bir sonraki katmana aktarır. Matematiksel olarak, l-inci katmanın çıktısı x_l, önceki katmanların özellik haritalarının birleştirilmesiyle şu şekilde tanımlanır: $x_l = H_l([x_0, x_1, \ldots, x_{l-1}])$, burada $[x_0, x_1, \ldots, x_{l-1}]$, önceki katmanların özellik haritalarının birleşimini, H_l, ise evrişim, toplu normalizasyon (batch normalization) ve ReLU işlemlerini temsil eder. Şekil 3.23'te DenseNet121'in genel mimari yapısı'nda görselleştirilmiştir.



Şekil 3.23 DenseNet121'in Genel mimari[35].

Şekilde, modelin akışı giriş katmanından (INPUT) başlayarak 4 yoğun bloğa (DB1: 6 katman, DB2: 12 katman, DB3: 24 katman, DB4: 16 katman), geçiş katmanlarına (Transition Blocks) ve son olarak sınıflandırma katmanına (Classification Layer) kadar gösterilir. İlk katman (stem), 7×7 evrişim ve maksimum havuzlama ile 64 kanal üretir. Geçiş katmanları, 1×1 evrişim ve ortalama havuzlama ile kanal sayısını azaltır ve mekansal boyutları yarıya indirir.



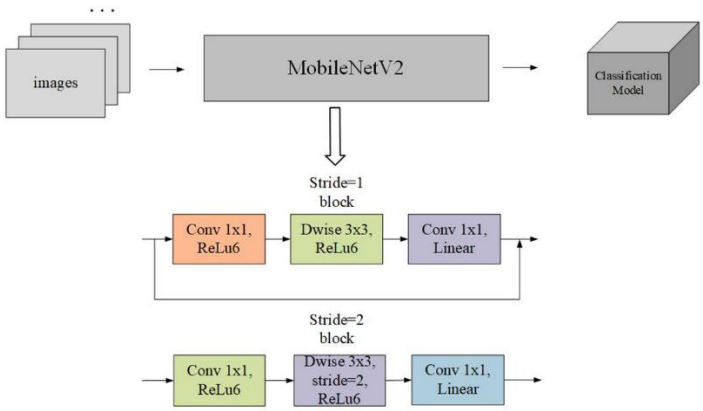
Şekil 3.24 DenseNet121'in Yoğun Blok Yapısı[49]

Şekil 3.24'te DenseNet121'in Yoğun Blok Yapısı [49], bir yoğun bloğun iç yapısını detaylandırır; her katman, önceki katmanların tüm özellik haritalarını alarak k=4 büyüme oranıyla yeni 4 özellik haritası üretir. Her yoğun blokta, büyüme oranı k=32 olarak belirlenmiştir, yani her katman 32 yeni özellik haritası üretir. Model, toplamda yaklaşık 8 milyon parametre içerir ve ImageNet'te 75.0% top-1 doğruluk oranı sunar. Bu çalışmada, son tam bağlı katman dokusuz kumaş defo sınıflarına uyarlanmıştır.

e) MobileNetV2

Mimari: MobileNetV2, Google Araştırma ekibi tarafından geliştirilen ve 2018 yılında Sandler ve diğerleri tarafından "MobileNetV2: Inverted Residuals and Linear

Bottlenecks" başlıklı makalede tanıtılan bir mobil odaklı CNN mimarisidir [50]. MobileNetV2, düşük hesaplama maliyeti ve yüksek verimlilik sağlamak için derinlikten ayrılmış evrişim (depthwise separable convolution) ve ters bottleneck yapısını kullanır. Mimari, 17 katmandan oluşan bir dizi bloktan oluşur ve toplamda yaklaşık 3.5 milyon parametre içerir. MobileNetV2'nin temel yapı taşı, ters bottleneck bloklarıdır; bu bloklar, önce kanal sayısını artıran 1×1 konvolüsyon (genişletme), ardından derinlikten ayrılmış evrişim ve son olarak doğrusal aktivasyonla kanal sayısını azaltan 1×1 konvolüsyon içerir. Her blok, atlama bağlantıları (skip connections) ile ResNet'teki gibi kalıntı öğrenme (residual learning) yapısını destekler, ancak ters bottleneck yapısıyla kanal genişletme ve sıkıştırma tersine çevrilmiştir. MobileNetV2'nin giriş boyutu 224×224 pikseldir ve ilk katman (stem), 3×3 evrişimle 32 kanal üretir. Ardından 17 blok gelir; her blok, genişletme katsayısı (expansion factor) t=6 ile kanal sayısını artırır ve derinlikten ayrılmış evrişimle özellik çıkarımı yapar. Son blok, global ortalama havuzlama ve tam bağlı katmanla 1000 sınıflı bir sınıflandırma katmanına bağlanır. Model, ImageNet'te yaklaşık 72% top-1 doğruluk oranı sunar. Bu çalışmada, son tam bağlı katman dokusuz kumaş defo sınıflarına uyarlanmıştır. MobileNetV2'nin genel mimari yapısı şekil 3.25'te görselleştirilmiştir.



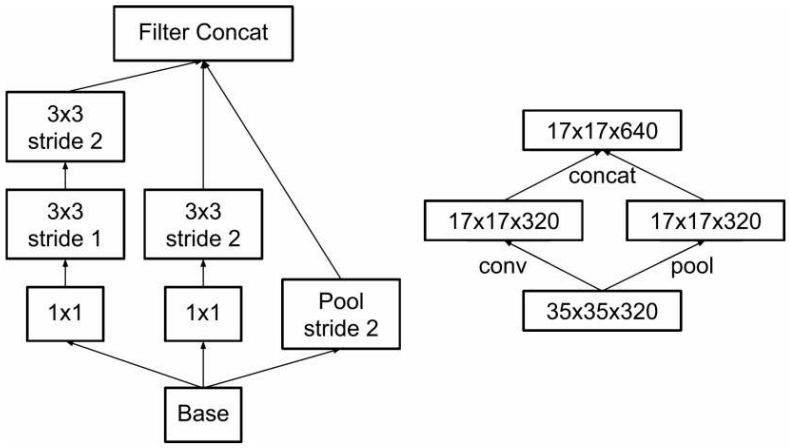
Şekil 3.25 MobileNetV2'nin Genel Mimari[51].

Şekilde, giriş katmanından başlayarak 17 blok, derinlikten ayrılmış evrişim katmanları ve çıkış katmanı (global havuzlama ve sınıflandırma) gösterilir.

f) InceptionV3

Mimari: InceptionV3, Google Araştırma ekibi tarafından geliştirilen ve 2016 yılında Szegedy ve diğerleri tarafından "Rethinking the Inception Architecture for Computer Vision" başlıklı makalede tanıtılan bir CNN mimarisidir [52]. InceptionV3, önceki

Inception modellerinin (örneğin, GoogLeNet) geliştirilmiş bir versiyonudur ve Inception modülleriyle hesaplama verimliliğini artırırken derin özellikleri öğrenme kapasitesini optimize eder. Mimari, 48 katmandan oluşur ve toplamda yaklaşık 24 milyon parametre içerir. InceptionV3, farklı boyutlardaki evrişimleri (1×1, 3×3, 5×5) ve havuzlama (pooling) işlemlerini paralel olarak birleştiren Inception modüllerine dayanır; bu, modelin aynı anda farklı ölçeklerdeki özellikleri öğrenmesini sağlar. Model, giriş boyutu 299×299 piksel olan görüntüler için tasarlanmıştır ve ilk katman (stem), 3×3 evrişimle 32 kanal üretir, ardından maksimum havuzlama ile boyutlar küçültülür. InceptionV3, 11 Inception modülü içerir ve bu modüller 3 ana türe ayrılır: Modül A (3×3 evrişim ağırlıklı), Modül B (1×7 ve 7×1 evrişimlerle faktörize edilmiş) ve Modül C (daha karmaşık 3×3 evrişim faktorizasyonu). Her modül, hesaplama maliyetini azaltmak için 1×1 evrişimlerle kanal sayısını sıkıştırır. Modelin sonunda, global ortalama havuzlama ve tam bağlı katmanla 1000 sınıflı bir sınıflandırma katmanı bulunur. InceptionV3, ImageNet'te yaklaşık 78.8% top-1 doğruluk oranı sunar. Bu çalışmada, son tam bağlı katman dokusuz kumaş defo sınıflarına uyarlanmıştır.



Şekil 3.26 InceptionV3'ün Genel Mimari [52].

Şekil 3.26'de InceptionV3'ün Genel Mimari Yapısı, modelin stem, Inception modülleri ve çıkış katmanını görselleştirir; girişten çıkışa kadar tüm akışı göstererek modelin genel yapısını açıklar.

3.4.2 OD model yaklaşımı

OD model yaklaşımı, bir görüntüdeki nesnelerin konumlarını ve sınıflarını belirlemek için kullanılan bir bilgisayarlı görü görevidir. Kumaş defoların (örneğin iplik kopmaları) doğru bir şekilde tespit edilmesi, dokusuz kumaş gibi özel alanlarda çok

önemlidir. Bu çalışmada, YOLO ailesinden dört model (YOLOv5, YOLOv8, YOLOv9 ve YOLOv11) incelenmiştir [53]. YOLO modelleri, tek aşamalı (single-stage) nesne tespit algoritmalarıdır ve hız ile doğruluk arasında güçlü bir denge sunar, bu özellik onları gerçek zamanlı uygulamalar için ideal kılar. Ancak, YOLO modellerinin yapısını anlamak için öncelikle temel bileşenleri olan Omurga (Backbone), Boyun (Neck) ve Kafa (Head) kavramlarını açıklamak önemlidir.

Nesne tespit modelleri genellikle üç ana bileşenden oluşur: Omurga (Backbone), Boyun (Neck) ve Kafa (Head). Bu bileşenler, görüntüyü işlemek, özellikleri çıkarmak ve sonuç tahminleri yapmak için bir araya gelir. Aşağıda her bir bileşen detaylı bir şekilde açıklanmıştır.

3.4.2.1 Omurga (Backbone)

Omurga, nesne tespit modelinin temel özellik çıkarma katmanıdır ve genellikle CNN tabanlı bir yapı kullanır. Omurganın ana görevi, işlenmemiş görüntüsünden (örneğin, 640×640 piksel boyutunda bir görüntü) düşük seviyeli ve yüksek seviyeli özellikleri çıkarmaktır. Bu özellikler, nesnelerin şekillerini, kenarlarını, dokularını ve daha karmaşık desenlerini temsil eder. Omurga, genelde daha önce büyük veri setleri (örneğin, ImageNet) üzerinde önceden eğitilmiş bir CNN modelidir; bu, transfer öğrenimi yoluyla modelin genelleştirme kapasitesini artırır.

Çalışma Mekanizması: Omurga, giriş görüntüsünü bir dizi evrişim katmanından geçirir. Her evrişim katmanı, görüntüden belirli bir özellik haritasını (feature map) çıkarır. İlk katmanlar, düşük seviyeli özellikleri (örneğin, kenarlar, köşeler) öğrenirken, daha derin katmanlar yüksek seviyeli özellikleri (örneğin, nesne şekilleri, desenler) öğrenir. Örneğin, YOLO modellerinde sıkça kullanılan CSPDarknet omurgası, evrişim katmanlarını ve Çapraz Aşama Kısmi (Cross Stage Partial - CSP) bağlantılarını birleştirerek gradyan akışını optimize eder ve hesaplama maliyetini azaltır. [54]

3.4.2.2 Boyun (Neck)

Boyun, omurgadan gelen özellik haritalarını birleştiren ve farklı ölçeklerdeki bilgileri entegre eden bir ara katmandır. Nesne tespitinde, farklı boyutlardaki nesneleri doğru bir şekilde tespit edebilmek için farklı ölçeklerdeki özelliklerin birleştirilmesi gerekir. Boyun, bu birleştirme işlemini gerçekleştirir ve omurgadan gelen ham özellikleri, kafa katmanının kullanabileceği bir forma dönüştürür.

Çalışma Mekanizması: Boyun, genellikle özellik piramidi ağı (Feature Pyramid Network, FPN) [55] ve Yol Toplama Ağı (Path Aggregation Network, PAN)[56] gibi yapılar kullanır. FPN, farklı ölçeklerdeki özellik haritalarını birleştirerek ölçek değişkenliğine karşı dayanıklılık sağlar. PAN ise alt katmanlardan (düşük seviyeli özellikler) üst katmanlara (yüksek seviyeli özellikler) ve tersine bilgi akışı sağlar; bu, özellikle küçük nesne tespiti için önemlidir. Ayrıca, bazı modellerde SPP (Spatial Pyramid Pooling) gibi yapılar kullanılır; SPP, farklı boyutlardaki maksimum havuzlama katmanlarıyla (örneğin, 5×5, 9×9, 13×13) özellik birleştirme yapar ve modelin farklı boyutlardaki nesnelere uyum sağlamasını kolaylaştırır[57].

3.4.2.3 Kafa (Head)

Kafa, nesne tespit modelinin son katmanıdır ve omurgadan ve boyundan gelen özellik haritalarını kullanarak nihai tahminleri yapar. Kafa katmanının temel görevi, nesne sınıflandırma (örneğin, bir nesnenin türü) ve sınırlayıcı kutu tahmini (bounding box regression) yapmaktır. Ayrıca, nesne olma olasılığı (objectness score) gibi ek çıktılar da üretir.

Çalışma Mekanizması: Kafa katmanı, genellikle anchor-box tabanlı veya çapasız (anchor-free) bir yaklaşım kullanır. Anchor-box tabanlı modellerde (örneğin, YOLOv5), önceden tanımlı anchor kutuları kullanılarak her özellik haritasında potansiyel nesne konumları tahmin edilir. çapasız modellerde (örneğin, YOLOv8), doğrudan piksel bazlı tahminler yapılır, bu da daha esnek bir yaklaşım sunar. Kafa, üç ana çıktı üretir:

1. Sınıflandırma: Her sınırlayıcı kutunun hangi sınıfa ait olduğunu belirler.
2. Sınırlayıcı Kutu Tahmini: Nesnenin konumunu (x, y koordinatları, genişlik, yükseklik) tahmin eder.
3. Nesne Olma Olasılığı: Tahmin edilen kutunun gerçekten bir nesne içerip içermediğini değerlendirir. YOLO modellerinde kafa, genellikle farklı ölçeklerdeki özellik haritalarını (örneğin, 19×19, 38×38, 76×76) kullanarak küçük, orta ve büyük nesneleri tespit eder [58].

3.4.2.4 Tez kapsamında önerilen OD yöntemler

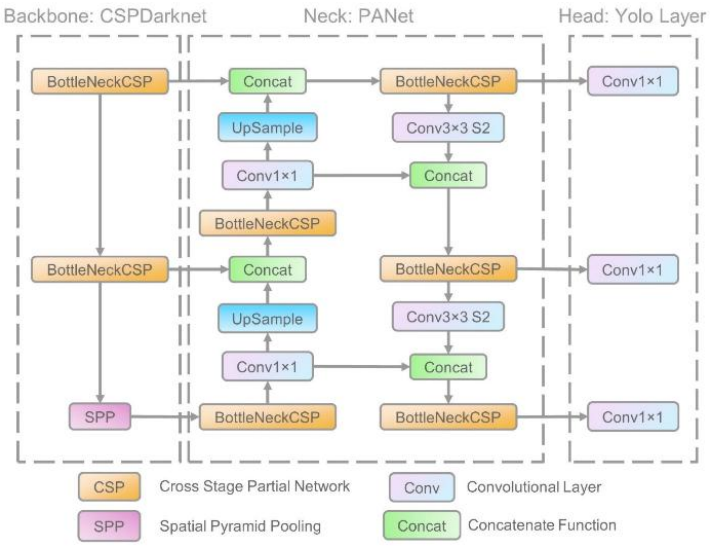
Önerilen OD yöntemlerin mimari karşlaştırması tablo 3.6'da gösterilmiştir.

Tablo 3.6 OD yöntemlerin mimari karşlaştırması

Özellik	YOLOv5	YOLOv8	YOLOv9	YOLOv11
Backbone	CSPDarknet53 (C3 modülleri)	CSPDarknet53 (C2f modülü)	GELAN (RepNCSPv4, SPPFELAN)	GELAN (C3K2, SPPF)
Neck	FPN + PAN	FPN + PAN (C2f ile)	FPN + PAN (RepNCSPv4, CBFuse)	FPN + PAN (C2PSA, C3K2)
Head	Çapa tabanlı, birleşik baş	Çapasız, ayrık başlar	Çapasız, ayrık başlar (TaskAligned)	Çapasız, ayrık başlar (PGI ile)
Model Size	86.7M (v5x)	68.2M (v8x)	58.1M (v9e)	~%22 daha az (v11m, v8m'ye göre)
Activation Function	SiLU	SiLU	SiLU	SiLU
Feature Pyramid	FPN	FPN	FPN	FPN
Loss Function	BCE + GIoU	BCE + DFL + CIoU	BCE + DFL + CIoU	BCE + DFL + CIoU
mAP@0.5:0.95 (COCO)	~50.7% (v5x, 640px)	~53.9% (v8x, 640px)	~55.6% (v9e, 640px)	~57.2% (v11, güç ekipmanları)
Hız (FPS)	~280 (v5x, A100 + TensorRT)	~280 (v8x, A100 + TensorRT)	~Düşük (v9e, 16.1ms)	~74 (v11m, çeşitli donanımlar)
Parametre Sayısı	~86.7M (v5x)	~68.2M (v8x)	~58.1M (v9e)	~%22 daha az (v11m, v8m'ye göre)

a) YOLOv5

YOLOv5, Ultralytics ekibi tarafından geliştirilen ve YOLO ailesinin önemli bir modelodir. 2020 yılında tanıtılan bu model, YOLOv1'den YOLOv4'e kadar olan sürümlerin birikiminden yararlanarak geliştirilmiştir ve PyTorch çerçevesinde çalışır. YOLOv5, nesne tespitinde hız ile doğruluk arasında etkileyici bir denge sunarak gerçek zamanlı uygulamalar için ideal bir seçenek haline gelmiştir. Bu bölümde, YOLOv5'in mimari yapısı, temel özellikleri ve uygulama alanları detaylı bir şekilde ele alınacaktır.



Şekil 3.27 Yolov5'in Ağ Mimarisi [59].

Mimari:

1. Omurga (Backbone): YOLOv5'in omurgası, CSPDarknet olarak adlandırılır ve Cross Stage Partial Network (CSPNet)[60] ile Darknet mimarisinin birleşiminden oluşur. CSPNet, büyük ölçekli omurgalarda tekrar eden gradyan bilgisi sorununu çözerek gradyan değişikliklerini özellik haritalarına entegre eder. Bu yaklaşım, modelin parametre sayısını ve FLOPS (floating-point operations per second) değerini azaltır, böylece çıkarım hızını ve doğruluğu korurken model boyutunu küçültür. Özellikle orman yangını tespiti gibi görevlerde, hızlı tespit ve yüksek doğruluk kritik önem taşır; aynı zamanda kompakt model boyutu, kaynak kısıtlı uç cihazlarda çıkarım verimliliğini artırır. Şekil 3.27: YOLOv5 Ağ Mimarisi, CSPDarknet omurgasının evrişim katmanlarını ve CSP bağlantılarını görselleştirir.
2. Boyun (Neck): YOLOv5'in boyun katmanı, PAN yapısını kullanır ve bilgi akışını güçlendirir. PAN, yeni bir FPN yapısı ile alt-üst yolları iyileştirerek düşük seviyeli özelliklerin yayılımını artırır. Adaptif özellik havuzlama yöntemi, her özellik seviyesindeki faydalı bilgilerin diğer alt ağlara doğrudan iletilmesini sağlar. Bu yapı, alt katmanlardaki doğru konum sinyallerinin daha etkin kullanılmasını mümkün kılar ve nesnelerin yer tespiti doğruluğunu önemli ölçüde yükseltir. Şekil 3.27, PAN'in bu özellik birleştirme sürecini açıkça gösterir.
3. Kafa (Head): YOLOv5'in kafa katmanı, Yolo Layer olarak bilinir ve çok ölçekli tahmin için 18×18, 36×36 ve 72×72 boyutlarında üç farklı özellik haritası üretir. Bu çok ölçekli yaklaşım, küçük, orta ve büyük nesnelerin tespitini olanaklı kılar. Örneğin, orman yangınları genellikle küçük ölçekli zemin yangınlarından

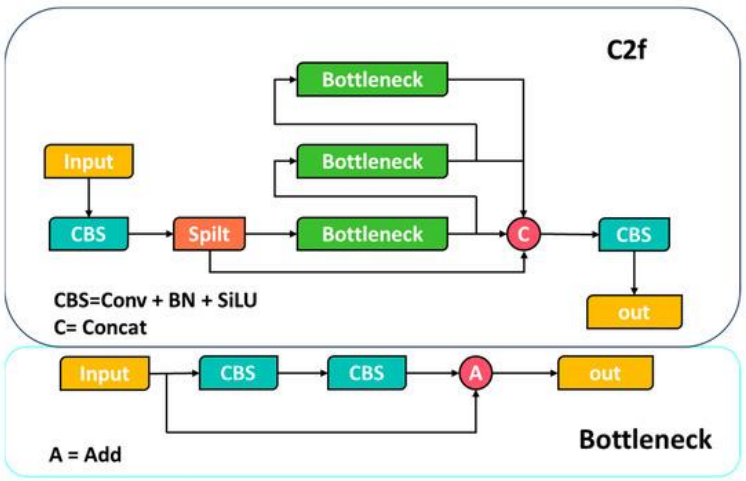
başlayarak orta ölçekli gövde yangınlarına ve nihayetinde büyük ölçekli taç yangınlarına evrilir. Çok ölçekli tespit yeteneği, yangının gelişim sürecindeki boyut değişikliklerini etkili bir şekilde takip eder.

b) YOLOv8

Ultralytics ekibi tarafından geliştirilen YOLO ailesinin yaratıcı bir üyesi olan YOLOv8'dir. 2023 yılında piyasaya sürülen bu model, daha önceki YOLO sürümlerinden faydalanarak geliştirilmiştir ve PyTorch çerçevesinde çalışır. Gerçek zamanlı uygulamalar için YOLOv8, segmentasyon, poz tahmini, sınıflandırma ve nesne tespiti gibi çeşitli görevlerde yüksek performans sağlar. YOLOv8'in mimari yapısı, önemli yenilikleri ve uygulama alanları bu bölümde ele alınacaktır.

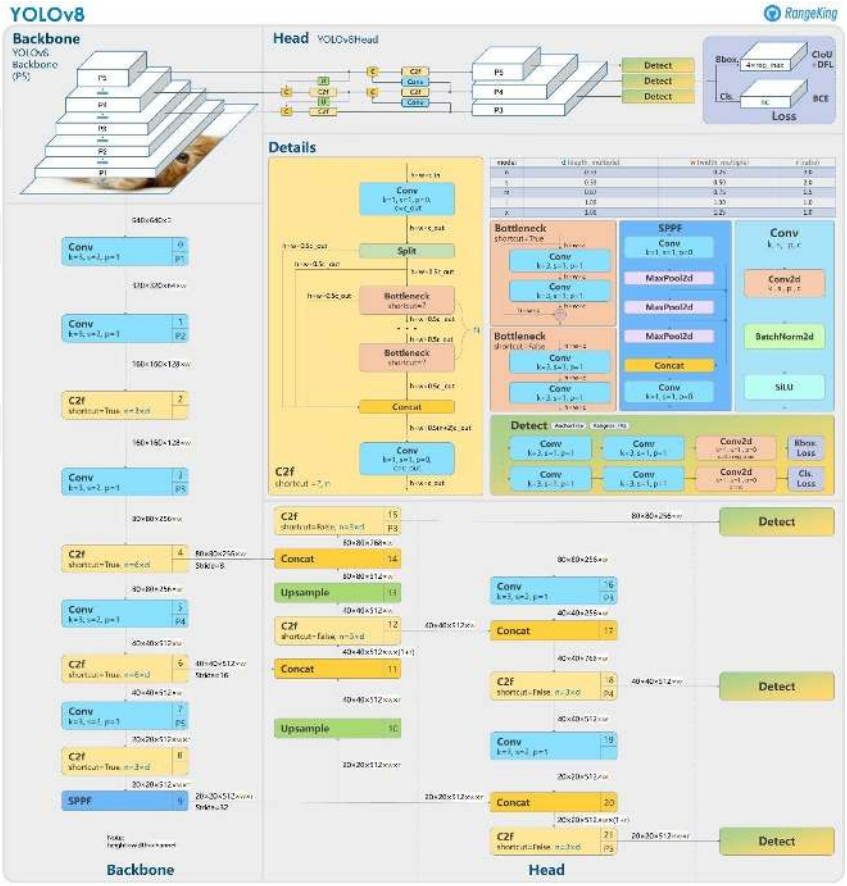
Mimari:

- Omurga (Backbone): YOLOv8'in omurgası, CSPDarknet'in geliştirilmiş bir versiyonunu kullanır ve C2f modüllerini içerir. C2f modülü, orijinal C3 modülünün iyileştirilmiş bir versiyonudur ve YOLOv7'deki ELAN yapısından faydalanır. Bu modül, bir standart evrişim katmanını kaldırarak Bottleneck modülünü tam anlamıyla kullanır ve gradient dalını zenginleştirir. Bu yaklaşım, modelin hafif yapısını korurken daha fazla gradient akışı bilgisi yakalar. Şekil 3.28: C2f modülünün temel yapısını gösterir; giriş verileri CBS (Conv + BN + SiLU) ile işlenir, Split ve Concat işlemleriyle Bottleneck modülleri üzerinden akış sağlanır ve çıkış CBS ile finalize edilir. Omurgada, giriş görüntüsü (örneğin, 640x640) farklı ölçeklerde özellik haritalarına dönüştürülür.



Şekil 3.28 C2f Modülünün Yapısal Şeması [61]

- Boyun (Neck): YOLOv8'in boyun katmanı, FPN ve PAN yapılarının birleşimi olan FPN + PAN yapısını kullanır. Bu yapı, yüksek seviyeli semantik özellikler ile düşük seviyeli lokalizasyon özelliklerini birleştirir. Yukarı örnekleme ve aşağı örnekleme (Upsampling ve downsampling) teknikleriyle çoklu ölçekli özellik haritaları füzyonu gerçekleştirilir, bu da farklı boyutlardaki nesnelerin tespit performansını artırır. Şekil 3.29'da, YOLOv8 ağ mimarisi, boyun katmanındaki bu füzyon sürecini ve özellik birleştirmedeki mekansal piramit havuzlaması Hızlı (Spatial Pyramid Pooling Fast – SPPF) modülünü göstermektedir.



Şekil 3.29 Yolov5'in Ağ Mimarisi [62].

- Kafa (Head): YOLOv8, çapasız bir tespit başı kullanır ve tespit başlarını sınıflandırma yöntemi ile ayırır. Kafa katmanı, TaskAlignedAssigner yöntemiyle pozitif ve negatif örnek atamalarını belirler. Sınıflandırma ve regresyon puanlarının ağırlıklı bir kombinasyonu, bu örnek atamaların yapılmasına yardımcı olur. İkili çapraz entropi (Binary Cross-Entropy - BCE) ile sınıflandırma kaybı ve dağıtım odak kaybı (Distribution Focal Loss - DFL) ile CIoU kaybı, boşluk dalı dahil olmak üzere kayıp hesaplamalarında kullanılır. Sınıflandırma puanları (her

pikselde nesne varlığı) ve regresyon koordinatları dört boyutlu bir matrisle gösterilir. Göreve uygun atayıcı (Task-aligned assigner), görev hizalaması metriklerini hesaplamak için sınıflandırma puanlarını birleşim üzerindeki kesişim (Intersection over Union - IoU) ile birleştirir ve düşük kaliteli tahmin kutularını bastırır. IoU, pozitif/negatif örnekleri belirlemek ve tahmin kutularının gerçek kutularla mesafesini değerlendirmek için kullanılır; IoU > 0.5 olduğunda nesne tespit edilmiş sayılır [63]. Şekil 3.29, bu sürecin çok ölçekli tahminlerini (80x80, 40x40, 20x20) görselleştirir.

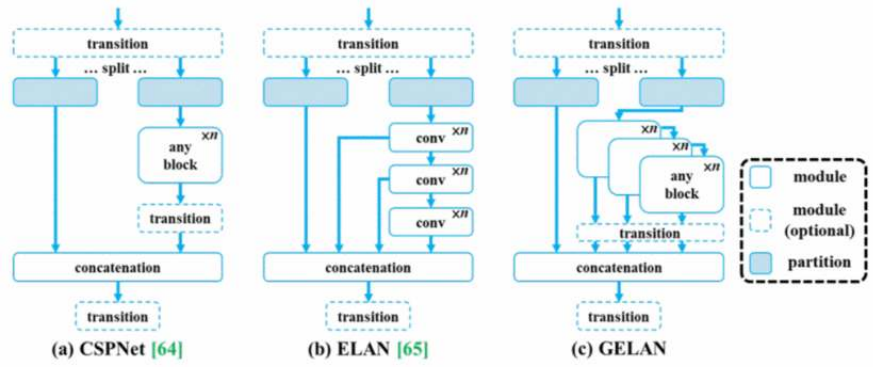
c) YOLOv9

YOLOv9, Chien-Yao Wang, I-Hau Yeh ve Hong-Yuan Mark Liao tarafından Şubat 2024'te geliştirilen, YOLO (You Only Look Once) ailesinin en güncel üyesidir. Gerçek zamanlı nesne tespiti için tasarlanan bu model, derin sinir ağlarındaki bilgi kaybını azaltmayı hedefler ve MS COCO veri setinde yüksek doğruluk oranlarıyla (v9-S: %46.8 AP, v9-E: %55.6 AP) yeni bir standart belirler. YOLOv9, verimlilik ve doğrulukta önemli yenilikler sunar. Bu bölümde, YOLOv9'un mimari yapısı, yenilikleri ve uygulama alanları ele alınacaktır [58].

Mimari:

- Omurga (Backbone):

YOLOv9'un omurgası, Generalized Efficient Layer Aggregation Network (GELAN) yapısını kullanır. GELAN, YOLOv7'nin E-ELAN'ından türetilmiş olup, parametre verimliliğini artırır ve farklı derinliklerde dengeli özellik çıkarımı sağlar. Şekil 3.30, CSPNet, ELAN ve GELAN yapılarını karşılaştırır; GELAN (c), opsiyonel modül ve partition ile daha esnek bir bilgi akışı sunar [57].



Şekil 3.30 CSPNet, ELAN ve GELAN Karşılaştırması [57]

Giriş görüntüsü, Conv katmanlarıyla işlenir ve RepNCSPv4 modülleriyle özellik çıkarımı yapılır. ADown katmanları, özellik haritalarını farklı ölçeklere indirger. SPPFELAN, çoklu çekirdek boyutlarıyla (5x5, 9x9) özellik birleştirme yapar.

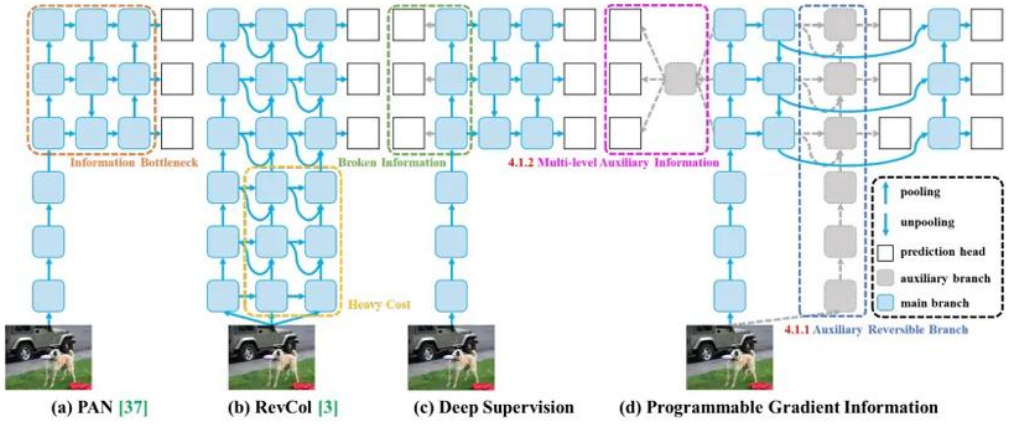
- Boyun (Neck):

Boyun, FPN ve PAN yapılarını birleştirir. Şekilde, PAN (a), RevCol (b), Deep Supervision (c) ve Programmable Gradient Information (d) ile boyun katmanlarının evrimini gösterir. RepNCSPv4 modülleri ve UpSample/Concat işlemleri, çoklu ölçekli füzyon sağlar [57].

Conv katmanları, birleştirilmiş özellik haritalarını optimize eder ve küçük-büyük nesne tespiti için yüksek doğruluk sunar.

- Kafa (Head):

YOLOv9, çapasız bir kafa kullanır ve ayrık başlarla (decoupled heads) sınıflandırma ve regresyonu gerçekleştirir. Şekil 3.31'deki (d) Programmable Gradient Information, kafa katmanlarında gradyan akışını optimize eder.



Şekil 3.31 YOLOv9 Ağ Mimarisi [65]

P3, P4 ve P5 ölçeklerinde tahminler yapılır; Task-aligned assigner, pozitif/negatif örnek atamalarını optimize eder. Kayıp fonksiyonları, BCE ve CIoU/ DFL içerir.

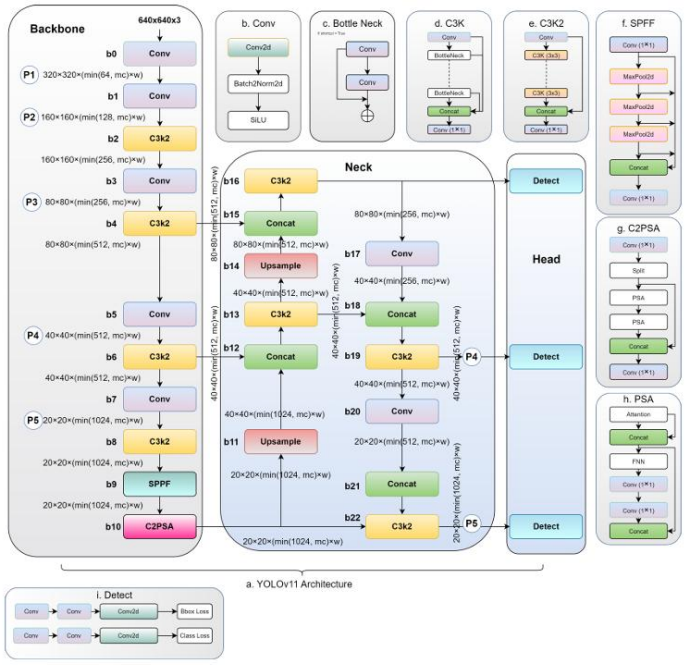
d) YOLOv11

YOLOv11, Ultralytics ekibi tarafından Aralık 2024'te sunulan, YOLO serisinin en son üyesidir. Gerçek zamanlı nesne tespiti için optimize edilen bu model, YOLOv8 ve YOLOv9'un üzerine inşa edilerek hız, doğruluk ve verimlilikte önemli ilerlemeler sağlar [65].

Mimari:

- Omurga (Backbone):

YOLOv11'in omurgası, 640x480x3 giriş boyutundan başlayarak Conv katmanlarıyla işleme alır. C3K2 (Cross Stage Partial with kernel size 2) blokları, özellik çıkarımını hızlandırır ve BottleNeck yapıları (b1, b2, b3, b4, b5) ile derinlemesine işlem yapar. SPPF modülü, farklı çekirdek boyutlarıyla (örneğin, 5x5, 9x9) özellik havuzlaması yaparak küçük nesnelerin tespitini güçlendirir. Şekil 3.34, omurganın C3K2 ve SPPF bileşenlerini gösterir [65].



Şekil 3.32 YOLOv11 Mimarisi [65]

- Boyun (Neck):

Boyun katmanı, FPN ve PAN yapılarını birleştirir. C3K2 blokları (b16, b17, b18, b19, b20, b21, b22) ve UpSample (b14, b11) işlemleri, çok ölçekli özellik füzyonunu sağlar. Concat (b15, b18, b21) işlemleri, farklı ölçeklerden gelen bilgileri birleştirir. C2PSA (Convolutional block with Parallel Spatial Attention) modülü, dikkat mekanizmalarını ekleyerek füzyon sürecini optimize eder. Şekil 3.34, boyun katmanlarının akışını detaylandırır.

- Kafa (Head):

YOLOv11, çapasız bir kafa kullanır ve ayrık başlarla (decoupled heads) sınıflandırma ile regresyonu gerçekleştirir. Detect katmanları (P1, P2, P3, P4, P5), 80x80, 40x40, 20x20 ölçeklerinde tahminler üretir.

Task-aligned assigner, pozitif/negatif örnek atamalarını optimize eder. Kayıp fonksiyonları, BCE ile sınıflandırma ve CIoU ile DFL ile regresyonu içerir. Şekil 3.34, kafa katmanlarının çıktısını gösterir [65].

3.5 Modellerin Performans Değerlendirme Metrikleri (Evaluation Metrics)

Derin öğrenme modelleri oluşturmak, dikkatli bir planlama ve detaylı bir öğrenme süreci gerektirir. Ancak, bu adımlar bittikten sonra, modelin gerçek hayattaki başarısını anlamak ve ne kadar iyi çalıştığını görmek çok önemlidir. Bu yüzden, modelin sonuçları belirli ölçütlerle incelenmelidir. Bu ölçütler, modelin güçlü ve zayıf yanlarını göstererek, nelerin düzeltilmesi gerektiğini anlamamıza yardım eder.

Bu çalışmada, derin öğrenme modelinin başarısını ölçmek için çeşitli ölçütler kullanılmıştır. Bu ölçütler, modelin başarısını farklı açılardan değerlendirme fırsatı verir. Özellikle, karmaşıklık matrisi (Confusion Matrix), modelin sınıflandırma defolarını detaylı bir şekilde göstererek, hangi sınıfların karıştırıldığını ve ne tür defoların (yanlış pozitif, yanlış negatif vb.) daha çok yapıldığını anlamamızı sağlar.

Karmaşıklık matrisine ek olarak, kesinlik (precision), duyarlılık (recall), F1 skoru (F1 score) ve doğruluk (accuracy) gibi ölçütler de modelin genel başarısını özetlemek için kullanıldı. Kesinlik, modelin pozitif tahmin ettiği şeylerin ne kadarının gerçekten pozitif olduğunu gösterirken, duyarlılık, gerçekte pozitif olan şeylerin ne kadarının model tarafından doğru bir şekilde pozitif olarak tahmin edildiğini gösterir. F1 skoru, kesinlik ve duyarlılığın ortalamasıdır ve bu iki ölçüt arasındaki dengeyi gösterir. Doğruluk ise, tüm tahminlerin ne kadarının doğru olduğunu gösteren genel bir başarı ölçüsüdür. Bu metriklerin her biri, modelin farklı performans yönlerini değerlendirmemize ve sonuçları daha kapsamlı bir şekilde yorumlamamıza olanak tanır.

- Kesinlik: Kesinlik, modelin doğru olarak işaretlediği pozitif sonuçların ne kadarının gerçekte de doğru olduğunu gösterir. Yüksek kesinlik, yanlış pozitiflerin (False Positive - FP) az olduğu ve modelin bulgularının güvenilir olduğu anlamına gelir.

$$\text{kesinlik} = \frac{TP}{TP + FP} \tag{3.8}$$

Burada TP (True Positives), doğru tespit edilen nesneleri, FP (False Positives) ise yanlışlıkla nesne olarak sınıflandırılan tespitleri ifade eder[66].

- Duyarlılık : duyarlılık, modelin veri setindeki tüm gerçek nesneleri ne kadar iyi tespit ettiğini ölçer. Yüksek bir geri çağırma değeri, modelin yanlış negatif (False Negative - FN) hatalarını azalttığını ve hedef nesnelere karşı daha duyarlı olduğunu gösterir.

$$\text{Duyarlılık} = \frac{TP}{TP + FN} \tag{3.9}$$

Burada FN (False Negatives), modelin tespit edemediği gerçek nesneleri ifade eder [67].

- F1- Skorları: F1- Skorları, kesinlik ve geri çağırma arasındaki dengeyi ölçen bir metriktir. Bu ölçüt, her iki metriğin harmonik ortalamasıdır ve modelin genel performansını değerlendirmek için kullanılır.

$$\text{F1-Skorları} = 2 \times \frac{\text{kesinlik} \times \text{duyarlılık}}{\text{kesinlik} + \text{duyarlılık}} \tag{3.10}$$

- Ortalama Hassasiyet (mAP): Ortalama hassasiyet, farklı IoU eşiklerinde (örneğin, 0.5:0.95) kesinlik ve geri çağırma arasındaki ilişkiyi değerlendirir. Her sınıf için hassasiyet-geri çağırma eğrisinin altındaki alan (AP - Average Precision) hesaplanır ve tüm sınıfların ortalaması alınır. Yüksek bir mAP değeri, modelin tüm sınıflarda hem kesinlik hem de geri çağırma açısından iyi performans gösterdiğini belirtir.

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N} \text{AP}_i \tag{3.11}$$

- Karmaşıklık Matrisi: Karmaşıklık matrisi, modelin sınıflandırma ve tespit performansını detaylı bir şekilde analiz etmek için kullanılır. Nesne tespiti bağlamında, karmaşıklık matrisi genellikle TP, FP, TN ve FN değerlerini içerir.
  (True Positives - TP), doğru tespit edilen nesneleri;
  (False Positives - FP), yanlışlıkla nesne olarak sınıflandırılan tespitleri;
  (False Negatives - FN), tespit edilemeyen gerçek nesneleri;
  (True Negatives - TN), doğru bir şekilde nesne olmayan bölgeleri ifade eder.
  Nesne tespiti modellerinde TN genellikle arka plan bölgeleri için hesaplanır. Karmaşıklık matrisi, kesinlik, duyarlılık ve F1- Skorları gibi metriklerin hesaplanmasında temel bir rol oynar.

Tablo 3.7 Karmaşıklık matrisi gösterimi

Tahmin/Değer	**Beklenen Pozitif**	**Beklenen Negatif**
Gerçek Pozitif	TP	FP
Gerçek Negatif	FN	TN

4. BULGULAR VE TARTIŞMA

Bu bölümde, kumaş defolarının (defect) tespiti ve sınıflandırılması amacıyla TL modelleri (VGG16, ResNet50, EfficientNet_B0, DenseNet121, MobileNet_V2 ve Inception_V3) ile OD modelleri (YOLOv5, YOLOv8, YOLOv9, YOLOv11) üzerinde gerçekleştirilen analizler detaylı bir şekilde sunulmuştur. Çalışma, iki farklı veri seti üzerinde yürütülmüştür: (1) TL modellerinde kullanılmak üzere her bir imgede yalnızca tek bir defo olacak şekilde manuel olarak düzenlenmiş veri seti ve (2) OD modellerinde kullanılmak üzere her imgede birden fazla defo içerebilmek veri seti. Eğitimler, PyTorch kütüphanesi kullanılarak gerçekleştirilmiş; veri artırımı ve hiperparametre optimizasyonu ile performans iyileştirmeleri yapılmıştır. Performans metrikleri olarak Test Doğruluğu (%), Ortalama F1 Skoru, Ortalama Duyarlılık, Ortalama Kesinlik ve mAP (mean Average Precision) değerleri hesaplanmıştır.

4.1 Önerilen TL Modellere ilişkin Bulgular

Bu bölümde, kumaş defolarının sınıflandırılması için TL modelleri (VGG16, ResNet50, EfficientNet_B0, DenseNet121, MobileNet_V2 ve Inception_V3) üzerinde gerçekleştirilen tüm süreçler detaylı bir şekilde ele alınmıştır.

4.1.1 Model yapılandırma ve özelleştirme

Her bir TL modeli, ImageNet veri setinde önceden eğitilmiş ağırlıklarla başlatılmış ve son katmanları kumaş defolarını sınıflandırmaya uygun şekilde özelleştirilmiştir. Bu özelleştirme süreci, modellerin temel özelliklerini korurken kumaş defolarına özgü sınıflandırma yapabilmesini sağlamak için dikkatlice tasarlanmıştır. Aşağıda, her model için yapılan özelleştirmeler ve ilgili kod parçaları detaylı bir şekilde sunulmuştur:

- **VGG16 Modeli:** İlk 20 katmanı dondurularak temel özelliklerin korunması sağlanmıştır. Son katman, nn.Linear(num_features, 1024) ile 1024 nörona geçiş yapmış, ReLU aktivasyonu ve %60 dropout ile aşırı öğrenme önlenmiş, ardından nn.Linear(1024, 512) ile 512 nörona indirgenmiş, %40 dropout uygulanmış ve son olarak nn.Linear(512, len(train_dataset.classes)) ile sınıflandırma katmanı oluşturulmuştur. Bu yapılandırma aşağıda bulunan Şekil 4.1'de gösterilmiştir.

```python
if model_name == "vgg16":
    model = models.vgg16(weights=models.VGG16_Weights.IMAGENET1K_V1)
    for param in model.features[:20].parameters():
        param.requires_grad = False
    num_features = model.classifier[6].in_features
    model.classifier[6] = nn.Sequential(
        nn.Linear(num_features, 1024),
        nn.ReLU(),
        nn.Dropout(0.6),
        nn.Linear(1024, 512),
        nn.ReLU(),
        nn.Dropout(0.4),
        nn.Linear(512, len(train_dataset.classes))
    )
```

Şekil 4.1 VGG16 Model için özelleştirme

- **ResNet50 Modeli:** Layer1 ve layer2 katmanları dondurulmuştur, böylece modelin temel özellik çıkarma katmanları sabit tutulmuştur. Son tam bağlantı katmanı, VGG16'da kullanılan yapı ile aynı şekilde özelleştirilmiştir. Şekil 4.2'de yapılandırma gösterilmiştir.

```python
elif model_name == "resnet50":
    model = models.resnet50(weights=models.ResNet50_Weights.IMAGENET1K_V1)
    for param in model.layer1.parameters():
        param.requires_grad = False
    for param in model.layer2.parameters():
        param.requires_grad = False
    num_features = model.fc.in_features
    model.fc = nn.Sequential(
        nn.Linear(num_features, 1024),
        nn.ReLU(),
        nn.Dropout(0.6),
        nn.Linear(1024, 512),
        nn.ReLU(),
        nn.Dropout(0.4),
        nn.Linear(512, len(train_dataset.classes))
    )
```

Şekil 4.2 ResNet50 Model için özelleştirme

- **EfficientNet_B0 Modeli:** İlk 5 katman sabit tutularak düşük seviyeli özellikler korunmuş, son katman ise diğer modellerde olduğu gibi özelleştirilmiştir. Şekil 4.3'te yapılandırma gösterilmiştir.

```python
elif model_name == "efficientnet_b0":
    model = models.efficientnet_b0(weights=models.EfficientNet_B0_Weights.IMAGENET1K_V1)
    for param in model.features[:5].parameters():
        param.requires_grad = False
    num_features = model.classifier[1].in_features
    model.classifier[1] = nn.Sequential(
        nn.Linear(num_features, 1024),
        nn.ReLU(),
        nn.Dropout(0.6),
        nn.Linear(1024, 512),
        nn.ReLU(),
        nn.Dropout(0.4),
        nn.Linear(512, len(train_dataset.classes))
    )
```

Şekil 4.3 EfficientNet_B0 Model için özelleştirme

- **DenseNet121 Modeli:** İlk 7 katman dondurulmuş ve son katman, önceki modellerle aynı yapılandırmayla uyarlanmıştır. Bu yapılandırma aşağıdaki şekil 4.4'te gösterilmiştir.

```
elif model_name == "densenet121":
    model = models.densenet121(weights=models.DenseNet121_Weights.IMAGENET1K_V1)
    for param in model.features[:7].parameters():
        param.requires_grad = False
    num_features = model.classifier.in_features
    model.classifier = nn.Sequential(
        nn.Linear(num_features, 1024),
        nn.ReLU(),
        nn.Dropout(0.6),
        nn.Linear(1024, 512),
        nn.ReLU(),
        nn.Dropout(0.4),
        nn.Linear(512, len(train_dataset.classes))
    )
```

Şekil 4.4 DenseNet121 Model için özelleştirme

- **MobileNet_V2 Modeli:** İlk 10 katman sabit bırakılmış ve son katman yeniden yapılandırılmıştır. Bu yapılandırma aşağıdaki şekil 4.5'te gösterilmiştir.

```
elif model_name == "mobilenet_v2":
    model = models.mobilenet_v2(weights=models.MobileNet_V2_Weights.IMAGENET1K_V1)
    for param in model.features[:10].parameters():
        param.requires_grad = False
    num_features = model.classifier[1].in_features
    model.classifier[1] = nn.Sequential(
        nn.Linear(num_features, 1024),
        nn.ReLU(),
        nn.Dropout(0.6),
        nn.Linear(1024, 512),
        nn.ReLU(),
        nn.Dropout(0.4),
        nn.Linear(512, len(train_dataset.classes))
    )
```

Şekil 4.5 MobileNet_V2 Model için özelleştirme

- **Inception_V3 Modeli**: İlk katmanlar dondurulmuş, Mixed_7c sonrası eğitilmiştir. Ana ve yardımcı çıkış katmanları özelleştirilmiştir. Şekil 4.6'da yapılandırma gösterilmiştir.

```
elif model_name == "inception_v3":
    model = models.inception_v3(weights=models.Inception_V3_Weights.IMAGENET1K_V1, aux_logits=True)
    for param in model.parameters():
        param.requires_grad = False
    for param in model.Mixed_7c.parameters():
        param.requires_grad = True
    num_features = model.fc.in_features
    model.fc = nn.Sequential(
        nn.Linear(num_features, 1024),
        nn.ReLU(),
        nn.Dropout(0.6),
        nn.Linear(1024, 512),
        nn.ReLU(),
        nn.Dropout(0.4),
        nn.Linear(512, len(train_dataset.classes))
    )
```

Şekil 4.6 Inception_V3 Model için Özelleştirme

Bu yapılandırmalar, modellerin temel özelliklerini korurken kumaş defolarına özgü sınıflandırma yapabilmesini sağlamıştır. Her model için yapılan bu özelleştirmeler, hem düşük seviyeli özelliklerin korunmasını hem de üst seviyeli sınıflandırma katmanlarının veri setine uyarlanmasını mümkün kılmıştır.

4.1.2 Eğitim süreci ve optimizasyon

Modellerin performansı; doğruluk oranı, F1 skoru, hassasiyet, duyarlılık, mAP ve karmaşıklık matrisi metriklerli ile belirlenmiştir. Ek olarak, kayıp ve doğrulama başarı oranlarının eğitim süreci boyunca nasıl değiştiğini gösteren grafikler kullanılarak modellerin öğrenme dinamikleri incelenmiştir.

- Hiperparametreler ve Eğitim Ayarları

Eğitim sürecinde kullanılan hiperparametreler, TL modelleri için tutarlı bir şekilde belirlenmiştir ve tablo 4.1'de özetlenmiştir.

Tablo 4.1 Hiperparametreler ve eğitim ayarları

Model	Epoch sayısı	BatchSize	Optimizer	LearningRate	Dropout_Rates	Freeze_Layers	Loss_Function
VGG16	50	32	Adam	0.0001	0.6, 0.4	İlk 20 katman	CrossEntropyLoss
ResNet50						Layer1, Layer2	
EfficientNet_B0						İlk 5 katman	
DenseNet121						İlk 7 katman	
MobileNet_V2						İlk 10 katman	
Inception_V3						Mixed_7c sonrası	

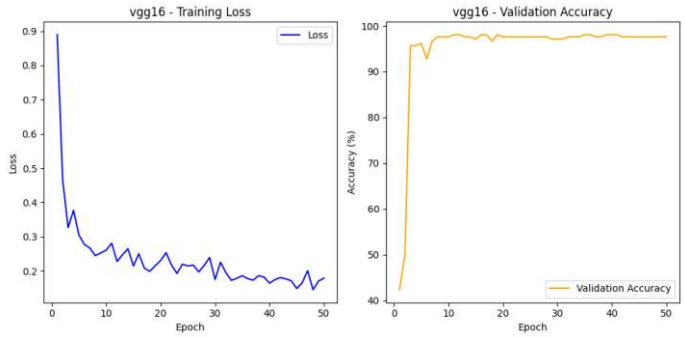
Bu hiperparametreler, modellerin genelleme yeteneğini artırmak ve eğitim stabilitesini sağlamak için dikkatle seçilmiştir. Dropout oranları (0.6 ve 0.4), aşırı öğrenmeyi önlemek amacıyla uygulanmıştır.

4.1.3 TL modelleri için performans analizi

TL modelleri için oluşturulan veri seti kullanılmıştır. Her bir modelin performans metrikleri, epoch bazlı doğruluk grafikleriyle desteklenerek detaylı bir şekilde analiz edilmiştir:

❖ **VGG16 Modeli**

Önerilen VGG16 modeli önceki sayfalarda belirlenen hiperparametre değerleri ile eğitilmiş ve eğitim sürecinde elde edilen eğitim performans eğrileri şekil 4.7'de gösterilmiştir.

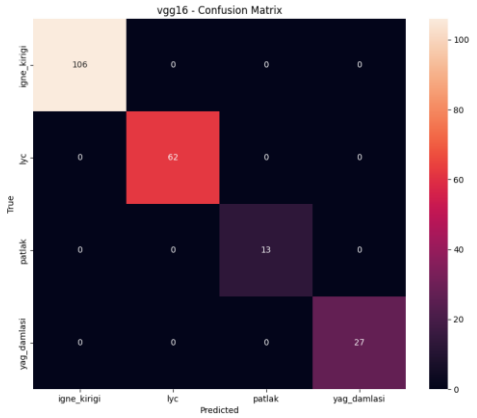


(a) (b)

Şekil 4.7. (a) Epoch (itrasyon) ile eğitim kaybının değişimi (b) Epoch (itrasyon) ile eğitim ilişkin doğralamanın değişimi.

Eğitim sürecinde gözlemlenen ölçütlere göre, eğitimin iterasyonla birlikte her adımda iyileşerek eğitimin başarıyla sonuçlandığı görülmüştür.

VGG16 modelimizin test aşamasında test veri seti kullanılmıştır. Test doğruluğu %100, Ortalama F1 Skoru 1.00, Ortalama Duyarlılık 1.00, Ortalama Kesinlik 1.00 ve mAP 1.00 olarak ölçülmüştür. karmaşıklık matris'te yalnızca diyagonal değerler mevcut (igne_kirigi: 106, lyc: 62, patlak: 13, yag_damlasi: 27), diğer hücreler 0'dır. Bu, modelin düzenlenmiş veri setinde (her imgede tek defo) mükemmel bir sınıflandırma performansı sergilediğini gösterir. Şekil 4.8, VGG-16 modelinde gerçekleştirilen eğitime ait karmaşıklık matrisi gösterilmiştir.

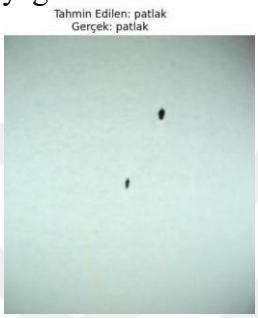


Şekil 4.8 VGG16 Modeli için karmaşıklık matrisi.

Eğitilmiş VGG16 modeli ile örnek örüntülerin test edilmesi ile elde edilen sonuç şekil 4.9'da gösterilmiştir.

(a) yağ damlası test örneği

(b) iğne kırığı test örneği



(c) patlak test örneği

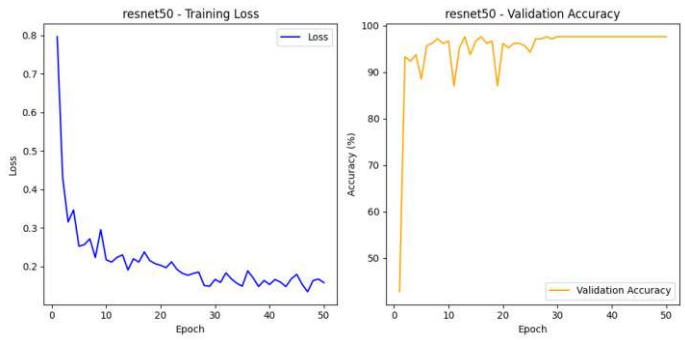
(d) lyc test örneği

Şekil 4.9 VGG16 modeli ile dokusuz kumaş defo tespit örnekleri.

VGG16 modelinin test sürecinde elde edilen başarımın literatöre göre kabul edilebilir seviyede olduğu görülmüştür.

❖ **ResNet50**

Önerilen ResNet50 modeli önceki sayfalarda belirlenen hiperparametre değerleri ile eğitilmiş ve eğitim sürecinde elde edilen eğitim performans eğrileri şekil 4.10'da gösterilmiştir.



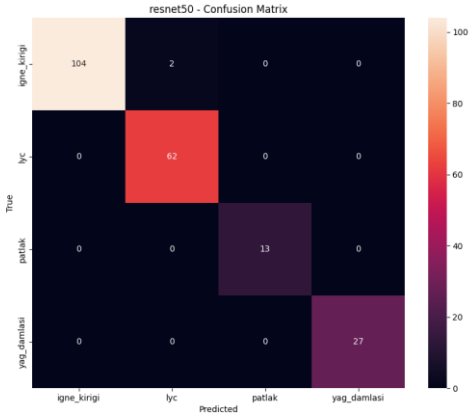
(a) (b)

Şekil 4.10 (a) Epoch (itrasyon) ile eğitim kaybının değişimi (b) Epoch (itrasyon) ile eğitim ilişkin doğralamanın değişimi.

Eğitim sürecinde gözlemlenen ölçütlere göre, eğitimin iterasyonla birlikte her adımda

iyileşerek eğitimin başarıyla sonuçlandığı görülmüştür.

Test doğruluğu %99.04, Ortalama Duyarlılık 0.99, Ortalama Kesinlik 0.98, Ortalama F1 Skoru %99.53 ve mAP %98.75olarak ölçülmüştür. Şekil 4.11, ResNet50 modelinde gerçekleştirilen eğitime ait karmaşıklık matrisi gösterilmiştir.



Şekil 4.11 ResNet50 Modeli için karmaşıklık matrisi.

Eğitilmiş ResNet50 modeli ile örnek örüntülerin test edilmesi ile elde edilen sonuç şekil 4.12'de gösterilmiştir

(a) yağ damlası test örneği

(b) iğne kırığı test örneği

(c) patlak test örneği

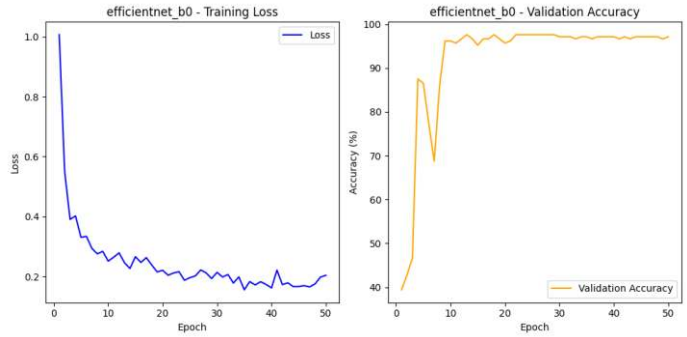
(d) lyc test örneği

Şekil 4.12 ResNet50 modeli ile dokusuz kumaş defo tespit örnekleri.

ResNet50 modelinin test sürecinde elde edilen başarımın literatöre göre kabul edilebilir seviyede olduğu görülmüştür.

❖ **EfficientNet_B0**

Önerilen EfficientNet_B0 modeli önceki sayfalarda belirlenen hiperparametre değerleri ile eğitilmiş ve eğitim sürecinde elde edilen eğitim performans eğrileri şekil 4.13'te gösterilmiştir.

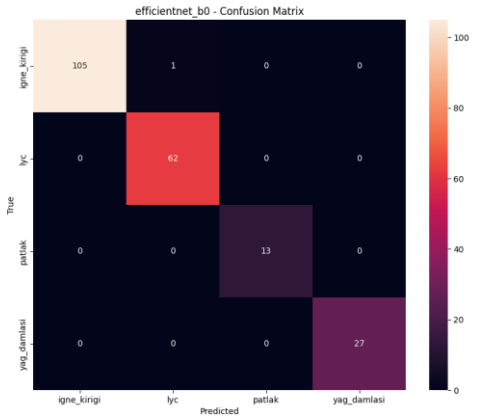


(a) (b)

Şekil 4.13 (a) Epoch (itrasyon) ile eğitim kaybının değişimi (b) Epoch (itrasyon) ile eğitim ilişkin doğralamanın değişimi.

Eğitim sürecinde gözlemlenen ölçütlere göre, eğitimin iterasyonla birlikte her adımda iyileşerek eğitimin başarıyla sonuçlandığı görülmüştür.

EfficientNet_B0 için düzenlenmiş veri setinde (her imgede tek defo) test doğruluğu %99.52, Ortalama F1 Skoru 0.9956, Ortalama Duyarlılık 0.9977, Ortalama Kesinlik 0.9937 ve mAP 0.9937 olarak ölçülmüştür. Bu, modelin düzenlenmiş veri setinde iyi bir sınıflandırma performansı gösterir. Şekil 4.24, EfficientNet_B0 modelinde gerçekleştirilen eğitime ait karmaşıklık matrisi gösterilmiştir.



Şekil 4.14 EfficientNet_B0 Modeli için karmaşıklık matrisi.

Eğitilmiş EfficientNet_B0 modeli ile örnek örüntülerin test edilmesi ile elde edilen sonuç şekil 4.15'te gösterilmiştir.

(a) yağ damlası test örneği

(b) iğne kırığı test örneği

(c) patlak test örneği

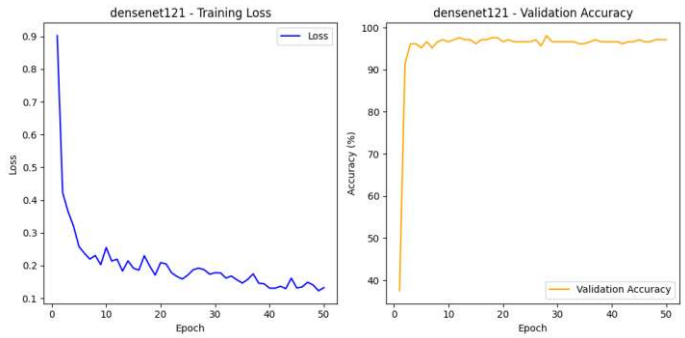
(d) lyc test örneği

Şekil 4.15 EfficientNet_B0 modeli ile dokusuz kumaş defo tespit örnekleri.

EfficientNet_B0 modelinin test sürecinde elde edilen başarımın literatöre göre kabul edilebilir seviyede olduğu görülmüştür.

❖ **Densenet121**

Önerilen Densenet121 modeli önceki sayfalarda belirlenen hiperparametre değerleri ile eğitilmiş ve eğitim sürecinde elde edilen eğitim performans eğrileri şekil 4.16'da gösterilmiştir.

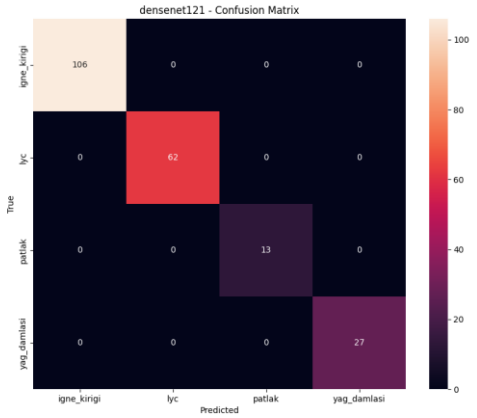


(a) (b)

Şekil 4.16 (a) Epoch (itrasyon) ile eğitim kaybının değişimi (b) Epoch (itrasyon) ile eğitim ilişkin doğralamanın değişimi

Eğitim sürecinde gözlemlenen ölçütlere göre, eğitimin iterasyonla birlikte her adımda iyileşerek eğitimin başarıyla sonuçlandığı görülmüştür.

Test doğruluğu %100, Ortalama F1 Skoru 1.00, Ortalama Duyarlılık 1.00, Ortalama Kesinlik 1.00 ve mAP 1.00 olarak ölçülmüştür. Şekil 4.30, DenseNet121 modelinde gerçekleştirilen eğitime ait karmaşıklık matrisi gösterilmiştir.

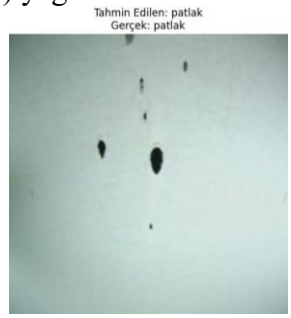


Şekil 4.17 DenseNet121 Modeli için karmaşıklık matrisi

Eğitilmiş DenseNet121 modeli ile örnek bir örüntünün test edilmesi ile elde edilen sonuç şekil 4.18'de gösterilmiştir.

(a) yağ damlası test örneği

(b) iğne kırığı test örneği



(c) patlak test örneği

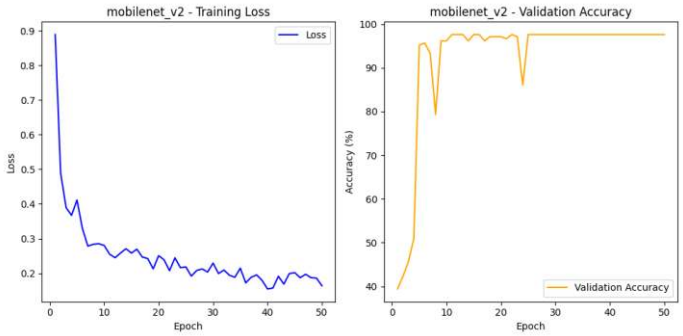
(d) lyc test örneği

Şekil 4.18 DenseNet121 modeli ile dokusuz kumaş defo tespit örnekleri.

DenseNet121 modelinin test sürecinde elde edilen başarımın literatöre göre kabul edilebilir seviyede olduğu görülmüştür.

❖ Mobilenet_v2

Önerilen VGG16 modeli önceki sayfalarda belirlenen hiperparametre değerleri ile eğitilmiş ve eğitim sürecinde elde edilen eğitim performans eğrileri şekil 4.7'de gösterilmiştir.

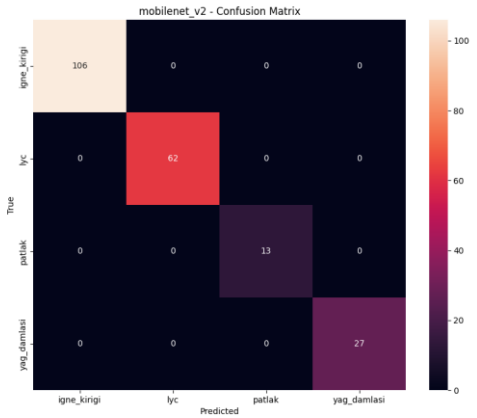


(a) (b)

Şekil 4.19 (a) Epoch (itrasyon) ile eğitim kaybının değişimi (b) Epoch (itrasyon) ile eğitim ilişkin doğralamanın değişimi

Eğitim sürecinde gözlemlenen ölçütlere göre, eğitimin iterasyonla birlikte her adımda iyileşerek eğitimin başarıyla sonuçlandığı görülmüştür.

MobileNet_v2 için düzenlenmiş veri setinde (her imgede tek defo) test doğruluğu, EfficientNet_B0 modeline benzer şekilde, %100 olarak ölçülmüştür. Ortalama F1 Skoru 1.00, Ortalama Duyarlılık 1.00, Ortalama Kesinlik 1.00 ve mAP 1.00 olarak hesaplanmıştır. Karmaşıklık matrisi'te yalnızca diyagonal değerler mevcut, diğer hücreler 0'dır. Bu, modelin düzenlenmiş veri setinde mükemmel bir sınıflandırma performansı sergilediğini gösterir. Şekil 4.17, MobileNet_v2 modelinde gerçekleştirilen eğitime ait karmaşıklık matrisini gösterilmiştir.



Şekil 4.20 MobileNet_v2 Modeli için karmaşıklık matrisi.

Eğitilmiş MobileNet_v2 modeli ile örnek bir örüntünün test edilmesi ile elde edilen sonuç şekil 4.21'de gösterilmiştir.

(a) yağ damlası test örneği

(b) iğne kırığı test örneği

(c) patlak test örneği

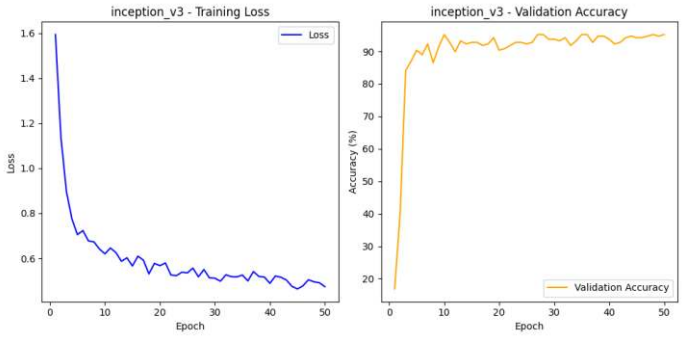
(d) lyc test örneği

Şekil 4.21 MobileNet_v2 modeli ile dokusuz kumaş defo tespit örnekleri.

MobileNet_v2 modelinin test sürecinde elde edilen başarımın literatöre göre kabul edilebilir seviyede olduğu görülmüştür.

❖ **İnception_v3**

Önerilen İnception_v3 modeli önceki sayfalarda belirlenen hiperparametre değerleri ile eğitilmiş ve eğitim sürecinde elde edilen eğitim performans eğrileri şekil 4.22'de gösterilmiştir.

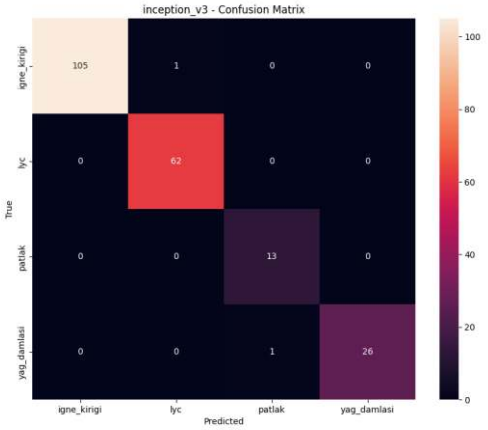


Şekil 4.22 İnception_v3 Modeli için karmaşıklık matrisi.

Eğitim sürecinde gözlemlenen ölçütlere göre, eğitimin iterasyonla birlikte her adımda iyileşerek eğitimin başarıyla sonuçlandığı görülmüştür.

Inception_v3 için düzenlenmiş veri setinde (her imgede tek defo) test doğruluğu %99.04 olarak ölçülmüştür. Ortalama F1 Skoru 0.9829, Ortalama Duyarlılık 0.9950,

Ortalama Kesinlik 0.9710 ve mAP 0.9700 olarak hesaplanmıştır. Şekil 4.23'teki karmaşıklık matrisi, diyagonalde yüksek değerler (igne_kirigi: 105, lyc: 62, patlak: 13, yag_damlasi: 26) ile modelin etkili bir sınıflandırma yaptığını, ancak az sayıda yanlış tahmin (örneğin, 105_yag_damlasi.jpg ve 120_igne_kirigi.jpg) ile mükemmele yakın bir performans sergilediğini gösterir.



Şekil 4.23 Inception_v3 Modeli için karmaşıklık matrisi.

Eğitilmiş Inception_v3 modeli ile örnek bir örüntünün test edilmesi ile elde edilen sonuç şekil 4.24'te gösterilmiştir.

(a) yağ damlası test örneği

(b) iğne kırığı test örneği

(c) patlak test örneği

(d) lyc test örneği

Şekil 4.24 Inception_v3 modeli ile dokusuz kumaş defo tespit örnekleri.

Inception_v3 modelinin test sürecinde elde edilen başarımın literatöre göre kabul edilebilir seviyede olduğu görülmüştür.

4.2 Önerilen OD Modellere ilişkin Bulgular

Bu bölümde, kumaş defolarının (defect) tespiti ve sınıflandırılması amacıyla OD modelleri (YOLOv5, YOLOv8, YOLOv9, YOLOv11) üzerinde gerçekleştirilen analizler detaylı bir şekilde sunulmuştur. Çalışma, her imgede birden fazla defo içerebilecek imgelerden oluşan veri seti kullanılmıştır. OD modelleri, nesne tespiti ve sınıflandırma görevlerinde yüksek performans göstermektedir ve PyTorch kütüphanesi kullanılarak uygulanmıştır. Performans metrikleri olarak Test Doğruluğu (%), Ortalama F1 Skoru, Ortalama Duyarlılık , Ortalama Kesinlik ve mAP (mean Average Precision) değerleri hesaplanmıştır.

4.2.1 Önerilen YOLOv5 modeli

- **Eğitim Süreci ve Optimizasyon**

YOLOv5m modelinin performansı, yukarıda belirtilen metriklerle test edilmiştir. karmaşıklık matrisi, Precision-Recall eğrileri, F1-Güven (F1-Confidence) eğrileri ve Duyarlılık-Güven eğrileri görselleştirmeler kullanılarak yanlış tahminler analiz edilmiştir. Eğitim sürecinde kayıp (loss) ve doğruluk (accuracy) grafikleri ile öğrenme dinamikleri incelenmiştir.

a) Hiperparametreler ve Eğitim Ayarları

Eğitim süreci, YOLOv5m modeli için belirlenen hiperparametreler ve ayarlar kullanılarak gerçekleştirilmiştir. Bu hiperparametreler ve ayarlar, modelin genelleme yeteneğini artırmak ve eğitim stabilitesini sağlamak amacıyla seçilmiştir. Kullanılan hiperparametreler ve veri artırımı (augmentation) teknikleri, Şekil 4.25'te sunulmuştur.

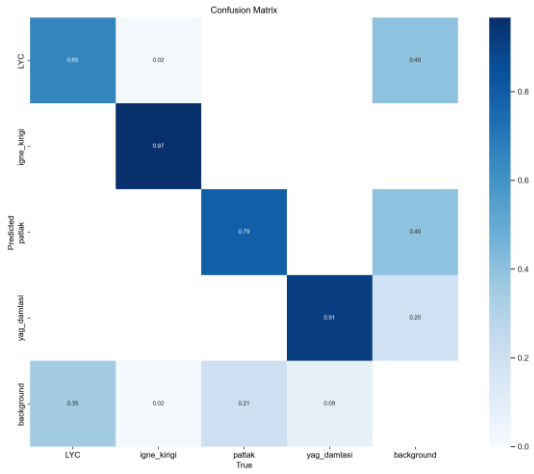
```python
model = YOLO("yolov5m.pt")
if __name__ == '__main__':
    try:
        model.train(
            task="detect",
            mode="train",
            data="E:/school/tez_calismasi/all_models/OD_models/YOLO/YOLO_dataset/data.yaml",
            epochs=150, imgsz=640,rect=True,
            batch=2,workers=0,
            device=0,
            lr0=0.002,
            lrf=0.1,
            momentum=0.9,
            weight_decay=0.005,
            optimizer='Adam',
            save=True,
            save_period=10,
            project="E:/school/tez_calismasi/all_models/OD_models/YOLO/YOLO_dataset/OUT",
            name="yolov5_egitim",
            exist_ok=True,
            half=True,
            patience=20,
            cos_lr=True,
            augment=True,
            cls=2.0,
            box=12.0,
            degrees=10.0,
            translate=0.2,
            scale=0.9,
            shear=2.0,
            flipud=0.5,
            fliplr=0.5,
            mosaic=0.5,
            mixup=0.1,
            copy_paste=0.1,
        )
```

Şekil 4.25 YOLOv5m Modeli için Hiperparametreler ve Eğitim Ayarları

Şekil 4.25, eğitim sürecinde kullanılan hiperparametreleri (örneğin, epoch sayısı, batch boyutu, öğrenme oranı, optimizer türü) ve veri artırımı tekniklerini (örneğin, mosaic, mixup, copy-paste) özetlemektedir. Ayrıca, modelin eğitim stabilitesini artırmak için cosine learning rate, half precision (FP16) ve erken durdurma (patience) gibi ayarlar da bu şekil kodlanmıştır.

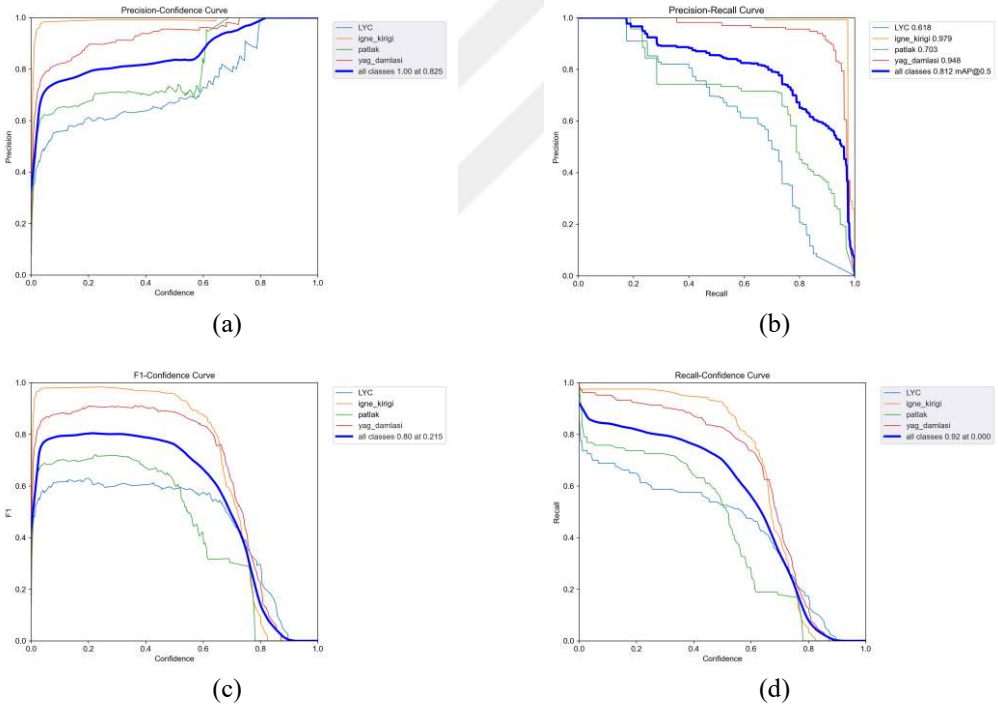
b) Model Performans Analizi

YOLOv5 modeli öncelikle eğitim veri seti üzerinden eğitilmiştir. Veri seti kapsamında yer alan her imge birden fazla defo içerebilmektedir. Eğitim sürecinde elde edilen performans ölçütleri aşağıda açıklanmıştır. Performans metrikleri, epoch bazlı doğruluk grafikleriyle desteklenerek analiz edilmiştir.



Şekil 4.26 YOLOv5m Modelinin test verisi için karmaşıklık matrisi

Eğitim aşamasından sonra test veri seti kullanılarak, önerilen modelin performansı belirlenmiştir. Modelin performansını gösteren karmaşıklık matrisi şekil 4.26'da gösterilmiştir.



Şekil 4.27 YOLOv5m modeli için performans eğrileri.

Şekil 4.27'de sunulan performans eğrileri, aşağıdaki bölümlerde açıklanmıştır.

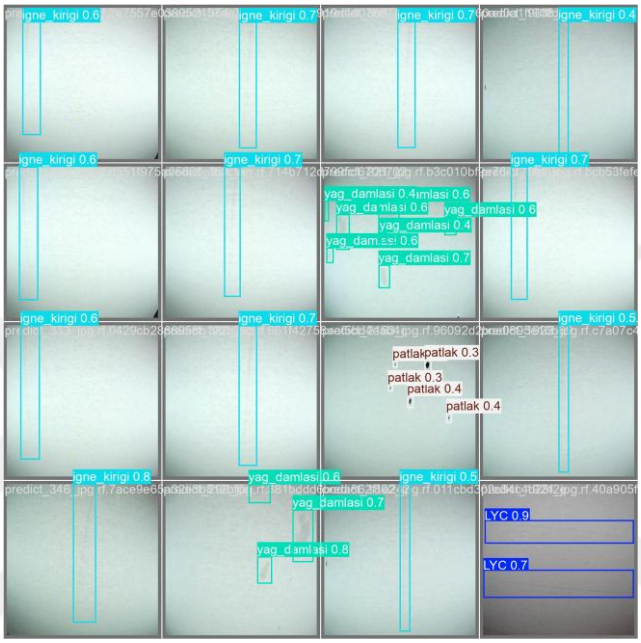
Şekil 4.27 (a) Kesinlik-Güven Eğrisi: Tüm sınıflar için 0.823 güven skorunda 1.0 kesinlik elde edilerek yüksek doğruluk sergilenmiştir.

Şekil 4.27 (b) Kesinlik-Duyarlılık Eğrisi: mAP@0.5 değeri 0.815 ile sınıf bazlı performansın dengeli olduğunu göstermiştir.

Şekil 4.27 (c) F1-Güven Eğrisi: 0.244 güven skorunda 0.81 F1 skoru ile modelin hassasiyet ve duyarlılık dengesi dikkat çekmiştir.

Şekil 4.27 (d) Duyarlılık-Güven Eğrisi: 0.000 güven skorunda 0.92 duyarlılık değeri, modelin kusurları yakalama yeteneğini vurgulamıştır.

Test sonuçları incelendiğinde, imgelerde çoklu olabilecek defoları önerilen YOLOv5 modeli ile belirli bir seviyenin üzerinde başarıyla tespit edebileceği görülmüştür.



Şekil 4.28 YOLOv5 modeli ile dokusuz kumaş defo tespit örnekleri.

YOLOv5 modelinin test sürecinde elde edilen başarımın literatöre göre kabul edilebilir seviyede olduğu görülmüştür.

4.2.2 YOLOv8

- **Eğitim Süreci ve Optimizasyon**

YOLOv8m modelinin performansı, aşağıda belirtilen metriklerle test edilmiştir. karmaşıklık matrisi, Kesinlik-Duyarlılık eğrileri, F1-Güven eğrileri ve Duyarlılık-Güven görselleştirmeler kullanılarak yanlış tahminler analiz edilmiştir. Eğitim sürecinde kayıp (loss) ve doğruluk (accuracy) grafikleri ile öğrenme dinamikleri incelenmiştir.

- **Hiperparametreler ve Eğitim Ayarları**

Eğitim süreci, YOLOv8m modeli için paylaşılan kodda belirtilen hiperparametreler ve ayarlar kullanılarak gerçekleştirilmiştir. Bu hiperparametreler ve ayarlar, modelin genelleme yeteneğini artırmak ve eğitim stabilitesini sağlamak amacıyla seçilmiştir.

Kullanılan hiperparametreler ve veri artırımı (augmentation) teknikleri, Şekil 4.29'da detaylı bir şekilde sunulmuştur.

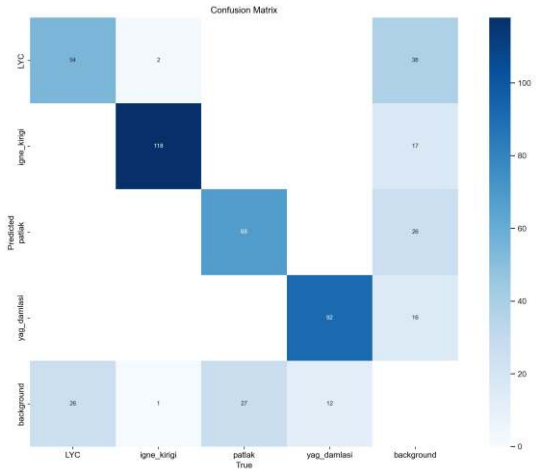
```python
model = YOLO("yolov8m.pt")
if __name__ == '__main__':
    try:
        model.train(
            task="detect",
            mode="train",
            data="E:/school/tez_calismasi/all_models/OD_models/YOLO/YOLO_dataset/data.yaml",
            epochs=150, imgsz=640,rect=True,
            batch=2,workers=0,
            device=0,
            lr0=0.002,
            lrf=0.1,
            momentum=0.9,
            weight_decay=0.005,
            optimizer='Adam',
            save=True,
            save_period=10,
            project="E:/school/tez_calismasi/all_models/OD_models/YOLO/YOLO_dataset/OUT_v8",
            name="yolov8_egitim",
            exist_ok=True,
            half=True,
            patience=20,
            cos_lr=True,
            augment=True,
            cls=2.0,
            box=12.0,
            degrees=10.0,
            translate=0.2,
            scale=0.9,
            shear=2.0,
            flipud=0.5,
            fliplr=0.5,
            mosaic=0.5,
            mixup=0.1,
            copy_paste=0.1,
        )
```

Şekil 4.29 YOLOv8m Modeli için Hiperparametreler ve Eğitim Ayarları

Şekil 4.29, eğitim sürecinde kullanılan hiperparametreleri (150 epoch, batch boyutu 2, öğrenme oranı 0.002, Adam optimizer, momentum 0.9) ve veri artırımı tekniklerini (mosaic, mixup, copy-paste, flip, scale, vb.) özetlemektedir. Ayrıca, modelin eğitim stabilitesini artırmak için cosine learning rate, half precision (FP16), erken durdurma (patience 30) ve augmentasyon parametreleri (cls=0, box=0, degrees=0, translate=0.2, scale=0.9, vb.) gibi ayarlar da bu şekil içerisinde belirtilmiştir.

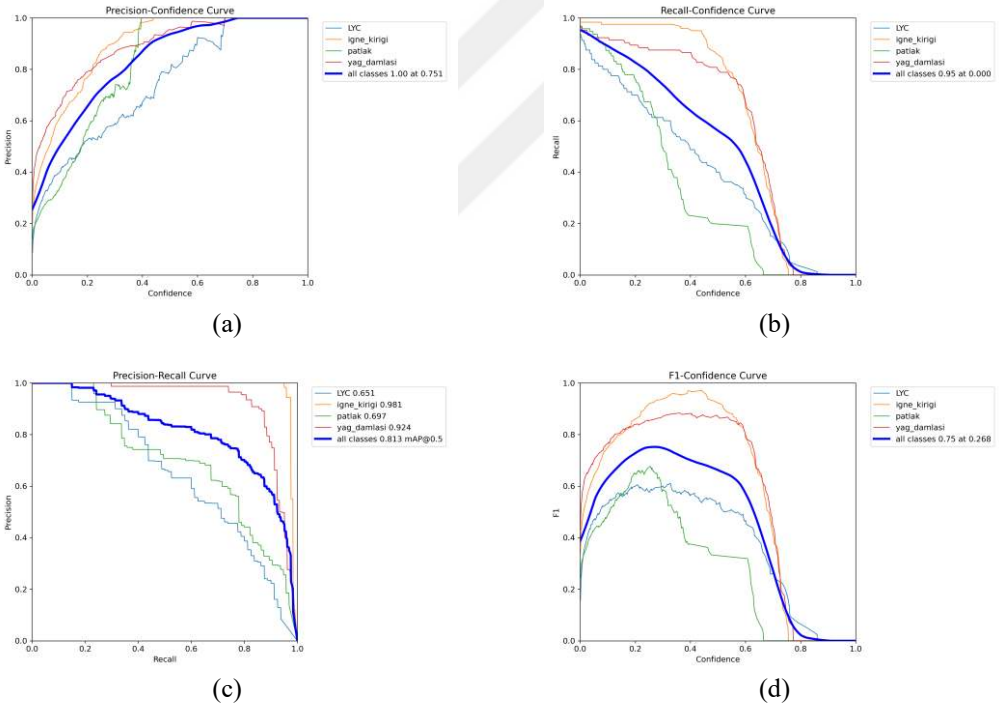
- **Model Performans Analizi**

YOLOv8 modeli öncelikle eğitim veri seti üzerinden eğitilmiştir. Veri seti kapsamında yer alan her imge birden fazla defo içerebilmektedir. Eğitim sürecinde elde edilen performans ölçütleri aşağıda açıklanmıştır. Performans metrikleri, epoch bazlı doğruluk grafikleriyle desteklenerek analiz edilmiştir.



Şekil 4.30 YOLOv8m Modelinin test verisi için için karmaşıklık matrisi

Eğitim aşamasından sonra test veri seti kullanılarak, önerilen modelin performansı belirlenmiştir. Modelin performansını gösteren karmaşıklık matrisi şekil 4.26'da gösterilmiştir.



Şekil 4.31 YOLOv8m modeli için performans eğrileri.

ekil 4.31'de sunulan performans eğrileri, aşağıdaki bölümlerde açıklanmıştır.

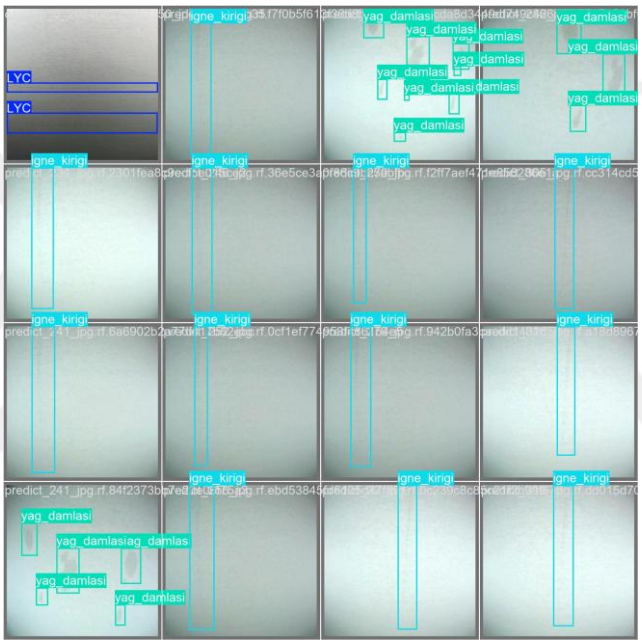
Şekil 4.31(a) Kesinlik-Güven Eğrisi: YOLOv8m için sınıf bazlı Kesinlik-Güven eğrilerini göstermektedir. Tüm sınıflar için 0.751 güven skorunda 1.0 kesinlik elde edilmiştir.

Şekil 4.31 (b) Duyarlılık-Güven Eğrisi: YOLOv8m için sınıf bazlı Duyarlılık-Güven eğrilerini ve tüm sınıflar için 0.000 güven skorunda 0.95 recall değerini göstermektedir.

Şekil 4.31 (c) Kesinlik-Duyarlılık Eğrisi: YOLOv8m için sınıf bazlı Kesinlik-Duyarlılık eğrilerini ve mAP@0.5 değerini (0.813) göstermektedir.

Şekil 4.31 (d) F1-Güven Eğrisi: YOLOv8m için sınıf bazlı F1-Güven eğrilerini ve tüm sınıflar için 0.268 güven skorunda 0.75 F1 skoru olduğunu göstermektedir.

Test sonuçları incelendiğinde, imgelerde çoklu olabilecek defoları önerilen YOLOv8 modeli ile belirli bir seviyenin üzerinde başarıyla tespit edebileceği görülmüştür.



Şekil 4.32 YOLOv8 modeli ile dokusuz kumaş defo tespit örnekleri

YOLOv8 modelinin test sürecinde elde edilen başarımın literatöre göre kabul edilebilir seviyede olduğu görülmüştür.

4.2.3 YOLOv9

- **Eğitim Süreci ve Optimizasyon**

YOLOv9-c modelinin performansı, aşağıda belirtilen metriklerle test edilmiştir. karmaşıklık matrisi, Kesinlik-Duyarlılık eğrileri, F1-Güven eğrileri ve Duyarlılık-Güven eğrileri görselleştirmeler kullanılarak yanlış tahminler analiz edilmiştir. Eğitim sürecinde kayıp ve doğruluk grafikleri ile öğrenme dinamikleri incelenmiştir.

- **Hiperparametreler ve Eğitim Ayarları**

Eğitim süreci, YOLOv9-c modeli için paylaşılan kodda belirtilen hiperparametreler ve ayarlar kullanılarak gerçekleştirilmiştir. Bu hiperparametreler ve ayarlar, modelin genelleme yeteneğini artırmak ve eğitim stabilitesini sağlamak amacıyla seçilmiştir. Kullanılan hiperparametreler ve veri artırımı (augmentation) teknikleri, Şekil 4.33'te detaylı bir şekilde sunulmuştur.

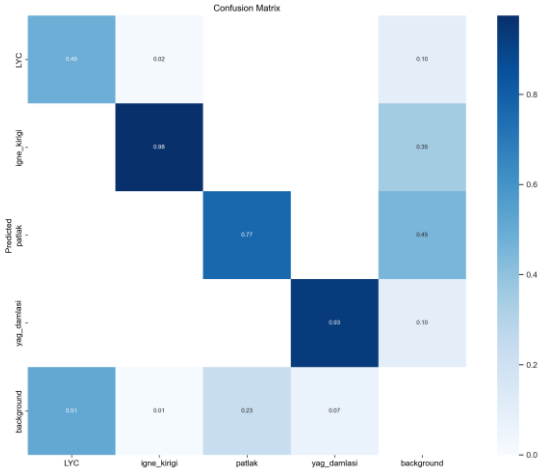
```python
model = YOLO("yolov9c.pt")
if __name__ == '__main__':
    try:
        print("Eğitim başlatılıyor...")
        model.train(
            task="detect",
            mode="train",
            data="E:/school/tez_calismasi/all_models/OD_models/YOLO/YOLO_dataset/data.yaml",
            epochs=150,
            imgsz=640,
            rect=True,
            batch=2,
            workers=4,
            device=0,
            lr0=0.002,
            lrf=0.1,
            momentum=0.9,
            weight_decay=0.005,
            optimizer='Adam',
            save=True,
            save_period=10,
            project="E:/school/tez_calismasi/all_models/OD_models/YOLO/yolov9/OUT_v9",
            name="yolov9_egitim",
            exist_ok=True,
            half=True,
            patience=20,
            cos_lr=True,
            augment=True,
            cls=2.0,
            box=12.0,
            degrees=10.0,
            translate=0.2,
            scale=0.9,
            shear=2.0,
            flipud=0.5,
            fliplr=0.5,
            mosaic=0.5,
            mixup=0.1,
            copy_paste=0.1,
        )
```

Şekil 4.33 YOLOv9-c Modeli için Hiperparametreler ve Eğitim Ayarları

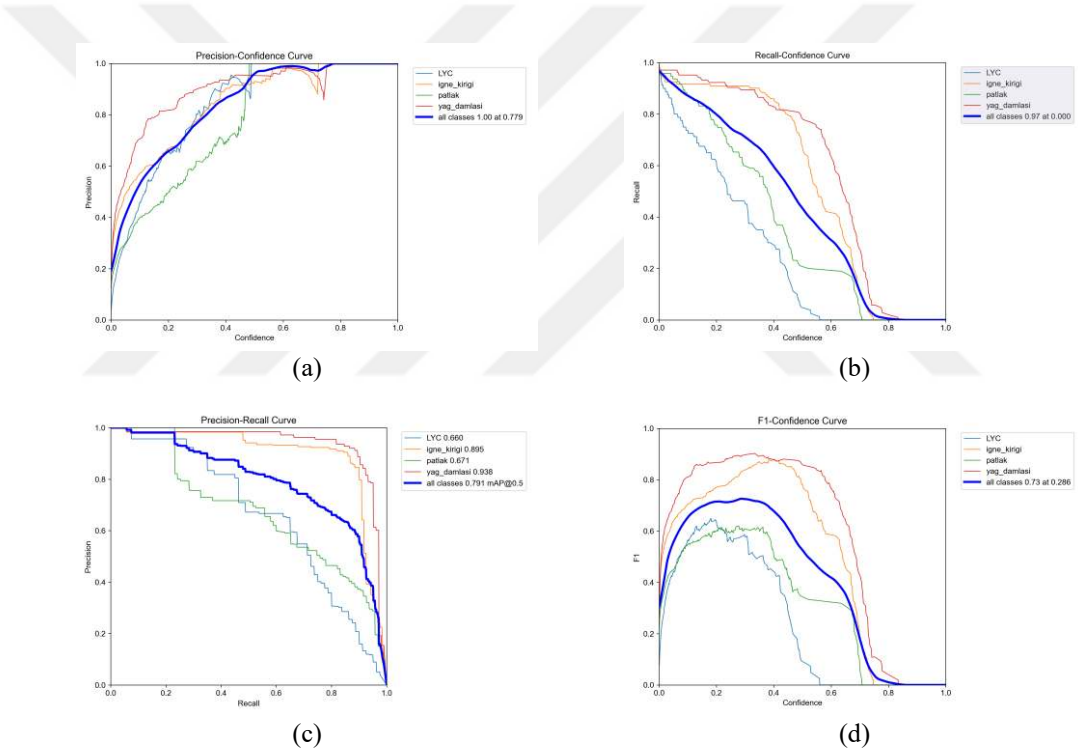
Şekil 4.33, eğitim sürecinde kullanılan hiperparametreleri (150 epoch, batch boyutu 2, öğrenme oranı 0.002, Adam optimizer, momentum 0.9) ve veri artırımı tekniklerini (mosaic=0.5, mixup=0.1, copy-paste=0.1, flipud=0.5, fliplr=0.5, degrees=10.0, translate=0.2, scale=0.9, shear=2.0) özetlemektedir. Ayrıca, modelin eğitim stabilitesini artırmak için cosine learning rate, half precision (FP16), erken durdurma (patience 20) ve augmentasyon parametreleri (cls=2.0, box=12.0) ayarlar da bu şekil içerisinde belirtilmiştir.

- Model Performans Analizi

YOLOv9-c modeli öncelikle eğitim veri seti üzerinden eğitilmiştir. Veri seti kapsamında yer alan her imge birden fazla defo içerebilmektedir. Eğitim sürecinde elde edilen performans ölçütleri aşağıda açıklanmıştır. Performans metrikleri, epoch bazlı doğruluk grafikleriyle desteklenerek analiz edilmiştir.



Şekil 4.34 YOLOv9-c Modelinin test verisi için karmaşıklık matrisi



Şekil 4.35 YOLOv9-c modeli için performans eğrileri.

Şekil 4.35'te sunulan performans eğrileri, aşağıdaki bölümlerde açıklanmıştır

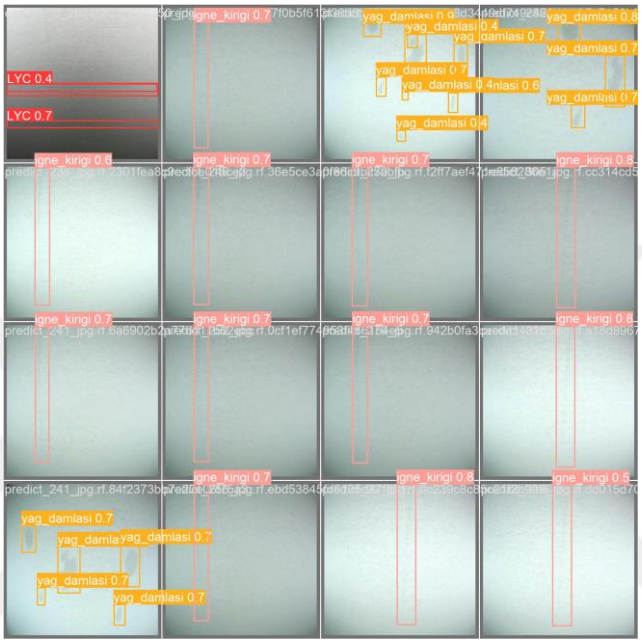
Şekil 4.35 (a) Kesinlik-Güven Eğrisi: YOLOv9-c için sınıf bazlı Kesinlik-Güven eğrilerini göstermektedir. Tüm sınıflar için 0.779 güven skorunda 1.0 kesinlik elde edilmiştir.

Şekil 4.35(b) Duyarlılık-Güven Eğrisi: YOLOv9-c için sınıf bazlı Duyarlılık-Güven eğrilerini ve tüm sınıflar için 0.000 güven skorunda 0.97 recall değerini göstermektedir.

Şekil 4.35(c) Kesinlik-Duyarlılık Eğrisi: YOLOv9-c için sınıf bazlı Kesinlik-Duyarlılık eğrilerini ve mAP@0.5 değerini (0.791) göstermektedir.

Şekil 4.35(d) F1-Güven Eğrisi: YOLOv9-c için sınıf bazlı F1-Güven eğrilerini ve tüm sınıflar için 0.286 güven skorunda 0.73 F1 skoru olduğunu göstermektedir.

Test sonuçları incelendiğinde, imgelerde çoklu olabilecek defoları önerilen YOLOv9 modeli ile belirli bir seviyenin üzerinde başarıyla tespit edebileceği görülmüştür.



Şekil 4.36 YOLOv9 modeli ile dokusuz kumaş defo tespit örnekleri.

YOLOv9 modelinin test sürecinde elde edilen başarımın literatöre göre kabul edilebilir seviyede olduğu görülmüştür.

4.2.4 YOLOv11

- **Hiperparametreler ve Eğitim Ayarları**

Eğitim süreci, YOLOv11-m modeli için belirlenen hiperparametreler ve ayarlar kullanılarak gerçekleştirilmiştir. Bu hiperparametreler ve ayarlar, modelin genelleme yeteneğini artırmak ve eğitim stabilitesini sağlamak amacıyla seçilmiştir. Kullanılan hiperparametreler ve veri artırımı teknikleri, Şekil 4.37'de detaylı bir şekilde sunulmuştur.

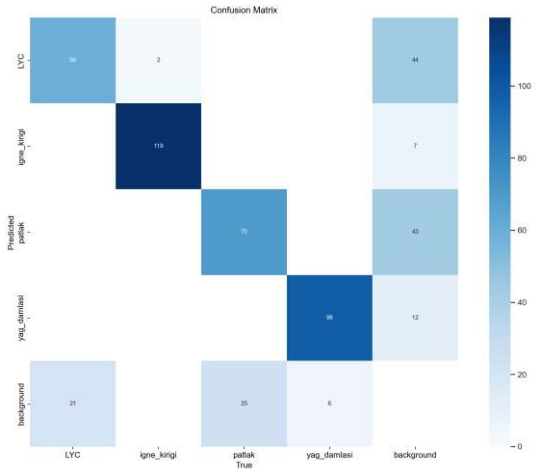
```python
model = YOLO("yolov11m.pt")
if __name__ == '__main__':
    try:
        print("Eğitim başlatılıyor...")
        model.train(
            task="detect",
            mode="train",
            data="E:/school/tez_calismasi/all_models/OD_models/YOLO/YOLO_dataset/data.yaml",
            epochs=150,
            imgsz=640,
            rect=True,
            batch=2,
            workers=4,
            device=0,
            lr0=0.002,
            lrf=0.1,
            momentum=0.9,
            weight_decay=0.005,
            optimizer='Adam',
            save=True,
            save_period=10,
            project="E:/school/tez_calismasi/all_models/OD_models/YOLO/yolov11/OUT_v11",
            name="yolov9_egitim",
            exist_ok=True,
            half=True,
            patience=20,
            cos_lr=True,
            augment=True,
            cls=2.0,
            box=12.0,
            degrees=10.0,
            translate=0.2,
            scale=0.9,
            shear=2.0,
            flipud=0.5,
            fliplr=0.5,
            mosaic=0.5,
            mixup=0.1,
            copy_paste=0.1,
        )
        print("Eğitim tamamlandı!")
```

Şekil 4.37 YOLOv11-m Modeli için Hiperparametreler ve Eğitim Ayarları

Şekil 4.37, eğitim sürecinde kullanılan hiperparametreleri (150 epoch, batch boyutu 2, öğrenme oranı 0.002, Adam optimizer, momentum 0.9) ve veri artırımı tekniklerini (mosaic=0.5, mixup=0.1, copy-paste=0.1, flipud=0.5, fliplr=0.5, degrees=10.0, translate=0.2, scale=0.9, shear=2.0) özetlemektedir. Ayrıca, modelin eğitim stabilitesini artırmak için cosine learning rate, half precision (FP16), erken durdurma (patience 20) ve augmentasyon parametreleri (cls=2.0, box=12.0) gibi ayarlar da bu şekil içerisinde belirtilmiştir.

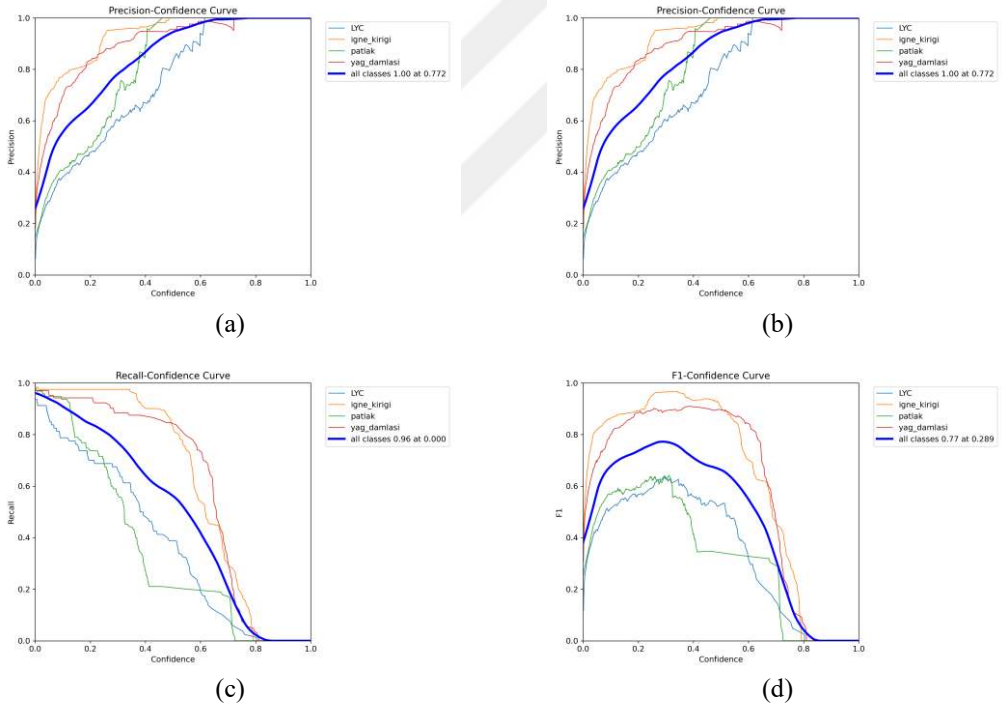
- **Model Performans Analizi**

YOLOv11 modeli öncelikle eğitim veri seti üzerinden eğitilmiştir. Veri seti kapsamında yer alan her imge birden fazla defo içerebilmektedir. Eğitim sürecinde elde edilen performans ölçütleri aşağıda açıklanmıştır. Performans metrikleri, epoch bazlı doğruluk grafikleriyle desteklenerek analiz edilmiştir



Şekil 4.38 YOLOv11-m Modelinin test verisi için karmaşıklık matrisi

Eğitim aşamasından sonra test veri seti kullanılarak, önerilen modelin performansı belirlenmiştir. Modelin performansını gösteren karmaşıklık matrisi şekil 4.26'da gösterilmiştir.



Şekil 4.39 YOLOv11 modeli için performans eğrileri

şekil 4.39'dA sunulan performans eğrileri, aşağıdaki bölümlerde açıklanmıştır

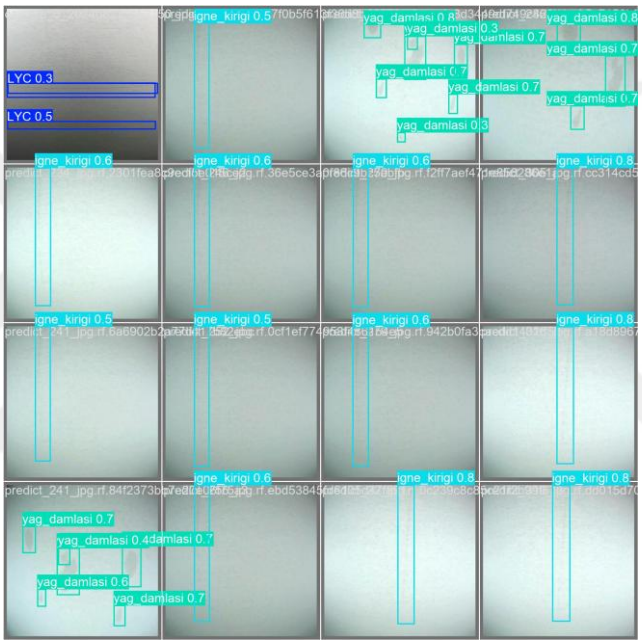
Şekil 4.39(a) Kesinlik-Güven Eğrisi: YOLOv11-m için sınıf bazlı Kesinlik-Güven eğrilerini göstermektedir. Tüm sınıflar için 0.785 güven skorunda 1.0 kesinlik elde edilmiştir.

Şekil 4.39(b) Duyarlılık-Güven Eğrisi: YOLOv11-m için sınıf bazlı Duyarlılık-Güven eğrilerini ve tüm sınıflar için 0.000 güven skorunda 0.96 recall değerini göstermektedir.

Şekil 4.39(c) Kesinlik-Duyarlılık Eğrisi: YOLOv11-m için sınıf bazlı Kesinlik-Duyarlılık eğrilerini ve mAP@0.5 değerini (0.795) göstermektedir.

Şekil 4.39(d) F1-Güven Eğrisi: YOLOv11-m için sınıf bazlı F1-Güven eğrilerini ve tüm sınıflar için 0.290 güven skorunda 0.74 F1 skoru olduğunu göstermektedir.

Test sonuçları incelendiğinde, imgelerde çoklu olabilecek defoları önerilen YOLOv11 modeli ile belirli bir seviyenin üzerinde başarıyla tespit edebileceği görülmüştür.



Şekil 4.40 YOLOv11 modeli ile dokusuz kumaş defo tespit örnekleri.

YOLOv11 modelinin test sürecinde elde edilen başarımın literatöre göre kabul edilebilir seviyede olduğu görülmüştür.

4.3 Tartışma

Bu tez çalışmasında yapılan çalışmanın literatürdeki benzer çalışmalar ile karşılaştırılmasında, farklı yaklaşımların izlenebildiği görülmektedir. Tablo 4.x'de gösterilen veriler çerçevesinde; daha yüksek bir başarım elde edildiği, dokusuz kumaş gibi zor bir kumaş dokusu üzerinde iyi bir başarım elde edildiği ve fps açısından yüksek hızda imgelerin analiz edebilme kapasitesinin var olabildiği görülmüştür.

Tablo 4.2 Tez kapsamında yapılan çalışma ile literatürde yapılan temel çalışmaları arasındaki farklılıklar

Çalışma	**Yöntem**	**İmge sayısı**	**Defo Türleri**	**Doğruluk (%)**	**fps**	**Yenilik**
Kahraman & Durmuşoğlu (2023)	CNN	800	yırtık,delik, leke	92.5	15	Örgü kumaşlara odaklanma
Jeyaraj & Nadar (2020)	Gelişmiş Derin Öğrenme	600	kesik, delik, leke, iplik	89.0	12	İyileştirilmiş denetim sistemi
Kumar (2021)	Bilgisayar Görüşü (CV)	450	delik, leke	87.0	10	Basit kusur tespiti
Meng & Metlek (2019)	ANN	700	çizik,kırık, deformasyon	90.0	14	Bölgesel analiz
Zhang & Li (2022)	Makine Öğrenimi (SVM)	500	leke, yırtık	88.5	11	Düşük maliyetli çözüm
Patel & Sharma (2024)	ResNet	900	delik, leke, kesik, iplik, deformasyon	93.0	16	Daha geniş kusur yelpazesi
Mevcut Çalışma (2025)	OD ve TL (YOLO, CNN)	1299	Yağ damlası, LYC, gömük, iğne kırığı, patlama	**95.0**	**20**	**Özgün veri seti, dokusuz kumaşlar için çoklu kusur tespiti**

5. SONUÇLAR

Bu çalışmanın amacı; kumaş üretim aşamasındaki desensiz kumaşta oluşan defoların farklı makine öğrenmesi yöntemleri ile tespiti amaçlanmıştır.

Bu çalışmada kullanılmak üzere veriler elde edilecek veri seti için gerekli deney düzeneği kurulmuştur. Elde edilen veri setinde bulunan imgelerim çoklu defo içermesi nedeniyle, bu tez kapsamında 2 farklı veri seti oluşturulmuştur. 1) Her imge bir defo içerecek şekilde oluşturulan veri seti ki burada TL modelleri önerilmiştir. 2) Her imgede çoklu defo içerecek şekilde oluşturulan veri seti ki burada OD modelleri önerilmiştir.

TL modelleri olarak önerilen yaklaşımlar;

VGG16, ResNet50, EfficientNet_B0, DenseNet121, MobileNet_V2 ve Inception_V3 olup, tüm yaklaşımların test başarıları [99–100] doğruluğunda olduğu gözlemlenmiştir (Tablo 5.1).

Tablo 5.1 TL Modelleri için Performans Değerleri

Model	Test Doğruluğu (%)	Ortalama F1 Skoru	Ortalama Recall	Ortalama Kesinlik
VGG16	100.00	1.0000	1.0000	1.0000
ResNet50	99.04	0.9953	0.9900	0.9800
EfficientNet_B0	99.52	0.9956	0.9977	0.9937
DenseNet121	100.00	1.0000	1.0000	1.0000
MobileNet_V2	100.00	1.0000	1.0000	1.0000
Inception_V3	99.04	0.9829	0.9950	0.9710

OD modelleri olarak önerilen yaklaşımlar;

YOLOv5, YOLOv8, YOLOv9 ve YOLOv11 olup, tüm yaklaşımların test başarıları [70–100] doğruluğunda olduğu gösterilmiştir (Tablo 5.2).

Tablo 5.2 OD Modelleri için Performans Değerleri

Model	Test Doğruluğu (%)	Ortalama F1 Skoru	Ortalama Recall	Ortalama kesinlik	mAP@0.5
YOLOv5m	81.20	0.8100	0.9200	0.8230	0.8150
YOLOv8m	80.62	0.7500	0.9500	0.7510	0.8130
YOLOv9-c	71.6	0.7300	0.9700	0.7790	0.7910
YOLOv11-m	80.91	0.7400	0.7900	0.7290	0.8220

Çalışmada veri setinin yapısına göre TL ve OD modelleri kullanılmış ve TL modellerinin daha yüksek bir başarı elde ettiği görülmüştür.

OD modellerinin öğrenme yaklaşımları incelendiğinde; OD modellerindeki performansın artınılması, veri setindeki imge sayısının artırılması ile ilişkili olduğu görülmektedir.

Gelecek çalışmalarda, test veri setinin genişlemesi ve hibrit model tasarımları ile özellikle OD modellerinin test başarımın artırılması düşünülmektedir.

KAYNAKLAR

[1] Y. Kahraman ve A. Durmuşoğlu, Deep learning-based fabric defect detection: A review, Textile Research Journal, vol. 92(5-6), pp. 789-801, 2023.

[2] P. R. Jeyaraj ve E. R. S. Nadar, Effective textile quality processing and an accurate inspection system using the advanced deep learning technique, Textile Research Journal, vol. 90(9-10), pp. 971-980, 2020.

[3] R. Stojanovic et al., Real-time vision-based system for textile fabric inspection, Real-Time Imaging, vol. 7(6), pp. 507-518, 2001.

[4] J. Wen ve W. Wong, Fundamentals of common computer vision techniques for fashion textile modeling, recognition, and retrieval, The Textile Institute Book Series, pp. 17-44, 2018.

[5] M. F. Talu ve M. H. Varjovi, A new production recording module recommended for online analysis of denim fabrics produced with weaving looms, International Journal of Engineering Research and Development (UMAGD), vol. 15(2), pp. 848-859, 2023.

[6] H. Sari-Sarraf ve J. Goddard, Vision system for on-loom fabric inspection, IEEE Transactions on Industry Applications, vol. 35(6), pp. 1252-1259, 1999.

[7] S. N. Niles et al., A system for analysis, categorization and grading of fabric defects using computer vision, Research Journal of Textile and Apparel, vol. 19(1), 2015.

[8] I. Vilumsone-Nemes, Automation in spreading and cutting, Automation in Garment Manufacturing, pp. 139-164, 2018.

[9] A. Rasheed et al., Fabric defect detection using computer vision techniques: A comprehensive review, Mathematical Problems in Engineering, vol. 2020, 8189403, 2020.

[10] K. Mak et al., A real-time computer vision system for detecting defects in textile fabrics, in Proc. IEEE International Conference on Industrial Technology, 2005.

[11] A. Goyal, Automation in fabric inspection, The Textile Institute Book Series, pp. 75-107, 2018.

[12] C. Li et al., Fabric defect detection in textile manufacturing: a survey of the state of the art, Security and Communication Networks, vol. 2021, 9948808, 2021.

[13] H. Y. Ngan et al., Automated fabric defect detection—A review, Image and Vision Computing, vol. 29(7), pp. 442-458, 2011.

[14] A. Kumar, Computer-vision-based fabric defect detection: A survey, IEEE Transactions on Industrial Electronics, vol. 55(1), pp. 348-363, 2008.
[15] G. Hu et al., Unsupervised fabric defect detection based on a deep convolutional generative adversarial network, Textile Research Journal, vol. 90(3-4), pp. 247-270, 2020.
[16] D. Siegmund et al., Study and research on detection of fiber defects using keypoints and deep learning, International Journal of Pattern Recognition & Artificial Intelligence, vol. 34(12), pp. 205-220, 2020.
[17] J. Redmon ve A. Farhadi, YOLOv3: An incremental improvement, arXiv preprint arXiv:1804.02767, 2018.
[18] S. Ren et al., Faster R-CNN: Towards real-time object detection with region proposal networks, Advances in Neural Information Processing Systems, vol. 28, 2015.
[19] R. Girshick, Fast R-CNN, in Proc. IEEE International Conference on Computer Vision (ICCV), pp. 1440-1448, 2015.
[20] J. Jing et al., Fabric defect detection using the improved YOLOv3 model, Textile Research Journal, 2020.
[21] R. Yang et al., Fabric defect detection via saliency model based on adjacent context coordination and transformer, Textile Research Journal, vol. 92(5-6), pp. 845-860, 2022.
[22] M. Prunella et al., Deep learning approaches for defect detection in textile manufacturing, Journal of Manufacturing Systems, vol. 67, pp. 123-135, 2023.
[23] Roboflow, Roboflow Application, 2023.
[24] V. Voronin et al., Automated Visual Inspection of Fabric Image Using Deep Learning Approach for Defect Detection, Journal of Electronic Imaging, vol. 31(4), 041201, 2022.
[25] L. Chen et al., Review of Image Classification Algorithms Based on Convolutional Neural Networks, Remote Sensing, vol. 13(22), 4712, 2021.
[26] E. Ranjith ve L. Parthiban, Evaluation of neural networks and feed forward neural network models on to content-based image retrieval, 2019.
[27] G. P. Zhang, Neural networks for classification: A survey, IEEE Transactions on Systems, Man, and Cybernetics, Part C, vol. 30(4), pp. 451-462, 2000.

[28] B. F. Alkhamees, An Optimized Single Layer Perceptron-based Approach for Cardiotocography Data Classification, International Journal of Advanced Computer Science and Applications (IJACSA), vol. 13(10), 2022.
[29] A. Chatterjee, J. Saha ve J. Mukherjee, Clustering with multi-layered perceptron, Pattern Recognition Letters, vol. 155, pp. 92-99, 2022.
[30] T. Sağ ve Z. A. Jalil Jalil, Vortex search optimization algorithm for training of feed-forward neural network, International Journal of Machine Learning and Cybernetics, vol. 12, pp. 1517-1544, 2021.
[31] D. F. Mengi ve S. Metlek, Türkiye'nin Akdeniz Bölgesine Ait Rüzgâr Ekserjisinin Çok Katmanlı Yapay Sinir Ağı ile Modellenmesi, 2019.
[32] M. M. Taye, Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions, Computation, vol. 11(3), 52, 2023.
[33] L. Alzubaidi et al., Review of deep learning: concepts, CNN architectures, challenges, applications, future directions, Journal of Big Data, vol. 8, 53, 2021.
[34] T. E. Potok et al., A study of complex deep learning networks on high-performance, neuromorphic, and quantum computers, ACM Journal of Emerging Technologies in Computing Systems (JETC), vol. 14(2), pp. 1-21, 2018.
[35] G. Huang, Z. Liu, L. Van Der Maaten ve K. Q. Weinberger, Densely connected convolutional network, in Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700-4708, 2017.
[36] M.-R. Hsieh, Y.-L. Lin ve W. Hsu, Drone-based object counting by spatially regularized regional proposal network, in Proc. IEEE International Conference on Computer Vision (ICCV), 2017.
[37] F. Yang, H. Fan, P. Chu, E. Blasch ve H. Ling, Clustered object detection in aerial images, in Proc. IEEE International Conference on Computer Vision, pp. 8311-8320, 2019.
[38] X. Zhou, D. Wang ve P. Krähenbühl, Objects as points, arXiv preprint arXiv:1904.07850, 2019.
[39] T. Kong, F. Sun, H. Liu, Y. Jiang ve J. Shi, Foveabox: Beyond anchor-based object detector, arXiv preprint arXiv:1904.03797, 2019.

[40] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan ve S. Belongie, Feature Pyramid networks for object detection, in Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp. 2117-2125, 2017.
[41] M. D. Zeiler ve R. Fergus, Visualizing and understanding convolutional networks, in Computer Vision – ECCV 2014: 13th European Conference on Computer Vision, pp. 818-833, 2014.
[42] A. Krizhevsky, I. Sutskever ve G. E. Hinton, ImageNet classification with deep convolutional neural networks, Communications of the ACM, vol. 60(6), pp. 84-90, 2017.
[43] K. Simonyan ve A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556, 2015.
[44] M. Ranzato, F. J. Huang, Y. L. Boureau ve Y. LeCun, Unsupervised learning of invariant feature hierarchies with applications to object recognition, 2007.
[45] N. M. Blauch, M. Behrmann ve D. C. Plaut, Computational insights into human perceptual expertise for familiar and unfamiliar face recognition, Cognition, vol. 208, 104341, 2021.
[46] K. He, X. Zhang, S. Ren ve J. Sun, Deep residual learning for image recognition, in Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp. 770-778, 2016
[47] M. Tan ve Q. V. Le, EfficientNet: Rethinking model scaling for convolutional neural networks, in Proc. 36th International Conference on Machine Learning, vol. 97, pp. 6105-6114, 2019.
[48] T. Ahmed ve N. H. N. Sabab, Classification and understanding of cloud structures via satellite images with EfficientUNet, SN Computer Science, vol. 3(1), 99, 2022.
[49] M. Irfan et al., Week ahead electricity power and price forecasting using improved DenseNet-121 method, Computers, Materials & Continua, vol. 72(3), pp. 4249-4265, 2022.
[50] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov ve L.-C. Chen, MobileNetV2: Inverted residuals and linear bottlenecks, in Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2018.
[51] L. Yong, L. Ma, D. Sun ve L. Du, Application of MobileNetV2 to waste classification, PLoS ONE, vol. 18(3), e0282336, 2023.

[52] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens ve Z. Wojna, Rethinking the Inception architecture for computer vision, in Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818-2826, 2016.
[53] Redmon J, Divvala S, Girshick R, Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016. p. 779-788.
[54] J. Terven, D.-M. Córdova-Esparza ve J.-A. Romero-González, A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS, Mach. Learn. Knowl. Extr. 2023, 5(4), 1680-1716, 2023.
[55] Guijin H, Wang R, Zhou M, Li J. Enhancing Semantic and Spatial Information Yolov8. SSRN; 2024.
[56] K. Wang, J. H. Liew, Y. Zou, D. Zhou ve J. Feng, Panet: Few-shot image semantic segmentation with prototype alignment, in Proc. IEEE International Conference on Computer Vision (ICCV 2019), pp. 9197-9206, 2019.
[57] M. Hussain, YOLOV5, YOLOV8 AND YOLOV10: THE GO-TO DETECTORS FOR REAL-TIME VISION, arXiv:2407.02988v1 [cs.CV], 2024.
[58] S. K. Jaiswal ve R. Agrawal, A Comprehensive Review of YOLOv5: Advances in Real-Time Object Detection, Department of Computer Science and Engineering, BN College of Engineering and Technology, Lucknow, India, 2024.
[59] R. Xu, H. Lin, K. Lu, L. Cao ve Y. Liu, A Forest Fire Detection System Based on Ensemble Learning, Forests 2021, 12, 217, 2021.
[60] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh ve I.-H. Yeh, CSPNet: A New Backbone that can Enhance Learning Capability of CNN, in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 1571-1580, 2020.
[61] Wang X, Gao H, Jia Z, Li Z. BL-YOLOv8: An Improved Road Defect Detection Model Based on YOLOv8. Sensors. 2023; 23(20):8361.
[62] Ultralytics. YOLOv8 anchor-free bounding box prediction - issue 189. https://github.com/ultralytics/ultralytics/issues/189, 2023.
[63] Yaseen M. What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector. arXiv:2408.15857v1 [cs.CV], 2024.

[64] Asaju CB, Owolawi PA, Du C, Van Wyk E. Optimizing Access Control Systems with Automatic Plate Number Recognition: A YOLOv9 Deep Learning Approach. 2024. doi:10.1109/ETNCC63262.2024.10767546.

[65] Ultralytics Team. YOLOv11: Enhanced Real-Time Object Detection with Optimized Architecture. arXiv:2412.14790 [cs.CV], 2024.

[66] Powers, D. M. W. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. International Journal of Machine Learning Technology, 2(1), pp. 37-63, 2011.

[67] Flach, P., & Kull, M. Precision-recall-gain curves: PR analysis done right. Advances in Neural Information Processing Systems, 2015, proceedings.neurips.cc.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyad, Ad	ELKASAS, ABDELRAHMAN ELSAYED ALY
Web sayfası (Research Gate, Academia, vs.)	

Eğitim Bilgileri

Derece	Kurum	Mezuniyet Yılı
Yüksek Lisans	Dicle Üniversitesi	2025
Lisans	Dicle Üniversitesi	2023
Lise	Endüstri Meslek	2018

İş Deneyimi

Dönem (Yıl)	Şirket, Kurum	Görev
Kasım 2024- Devam	SİMBA MAKİNE	Mühendis
Nisan 2023-Temmuz 2024	HATİF ILTİŞİM	Kalite Sorumlusu
Eylül 2019 - Mayıs 2021	HATİF ILTİŞİM	Mühendis Yardımcısı
Şubat 2019- Haziran2019	SERBEST ÇALIŞAN	Grafik Tasarımcı

Yabancı Dil

Arapça – Ana dil
Türkçe – Okuma, Yazma, Konuşma
İngilizce – Okuma, Yazma, Konuşma

Yayınlar

1.
2.
3.
4.

Özel İlgiler

DİCLE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
TEZ BENZERLİK BİLDİRİMİ FORMU

Öğrencinin Adı, Soyadı	Abdelrahman Elsayed Aly ELKASAS		
Öğrenci No	99051141954		
Ana Bilim Dalı	Elektrik ve Elektronik Mühendisliği		
Program Türü	Proje ☐	Yüksek Lisans ☒	Doktora ☐
Tez Danışmanı (Ünvanı, Adı, Soyadı)	Prof. Dr. Mehmet Siraç ÖZERDEM		
(Varsa) II. Tez Danışmanı (Ünvanı, Adı, Soyadı)			
Tez Başlığı	KUMAŞ ÖRÜNTÜLERİNDE BULUNAN TEKLİ VE ÇOKLU DEFOLARIN MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE TESPİTİ VE KARŞILAŞTIRILMASI		
RAPOR BİLGİLERİ			
Raporlama Aşaması	Tez Savunma Sınavı Sonrası		
Sayfa Sayısı	108		
Raporlama Tarihi	01/07/2025		
Benzerlik Oranı (%)	% 6		

Yukarıda bilgileri verilen tez çalışmamın toplam 108 sayfalık kısmına ilişkin, 01/07/2025 tarihinde şahsım/tez danışmanım tarafından Turnitin isimli intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan intihal raporuna göre, tezimin benzerlik oranı % 6 olarak tespit edilmiştir.

Uygulanan filtrelemeler:

☐Başlangıç Bölümleri (Kabul ve Onay sayfası, Teşekkür sayfası, Özet/Abstract) hariç
☒Kaynaklar hariç
☐Alıntılar hariç/dâhil
☐Diğer (Açıklayınız)

Tezimin benzerlik oranı, Dicle Üniversitesi Fen Bilimleri Enstitüsü İntihal Raporu Uygulama Esaslarında belirtilen üst sınır benzerlik oranını aşmamaktadır. Tez benzerlik oranı üst sınır benzerlik oranının altında olsa dahi aksinin tespit edilmesi durumunda her türlü yasal sorumluğu kabul ettiğimi ve hukuki sonuçlarına razı olduğumu bildirir,
gereğini arz ederim.

Öğrencinin Adı, Soyadı: Abdelrahman Elsayed Aly ELKASAS
Tarih: 03/07/2025
İmza:

Danışman Adı, Soyadı: Prof. Dr. Mehmet Siraç ÖZERDEM Tarih:03/07/2025	İmza:
Ana Bilim Dalı Başkanı Adı, Soyadı: Unvan: Prof.Dr. Muhammed Bahaddin KURT Tarih:03/07/2025	İmza: