MARMARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

FPGA TABANLI PCB HATA TESPİTİNDE DERİN ÖĞRENME UYGULAMASI

İZEMNUR BUDAK

YÜKSEK LİSANS TEZİ
Elektrik-Elektronik Mühendisliği Anabilim Dalı
Elektrik-Elektronik Mühendisliği (Türkçe) Programı

DANIŞMAN
Prof. Dr. Hayriye KORKMAZ

EŞ-DANIŞMAN
Öğr. Gör. Dr. Sezen BAL

İSTANBUL, 2024

MARMARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

FPGA TABANLI PCB HATA TESPİTİNDE DERİN ÖĞRENME UYGULAMASI

İZEMNUR BUDAK
523121009

YÜKSEK LİSANS TEZİ
Elektrik-Elektronik Mühendisliği Anabilim Dalı
Elektrik-Elektronik Mühendisliği (Türkçe) Programı

DANIŞMAN
Prof. Dr. Hayriye KORKMAZ

EŞ-DANIŞMAN
Öğr. Gör. Dr. Sezen BAL

İSTANBUL, 2024

MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES IN PURE AND APPLIED SCIENCES

İzemnur BUDAK a Master of Science student of Marmara University Institute for Graduate Studies in Pure and Applied Sciences, defended her thesis entitled "**FPGA Based Deep Learning Acceleration for PCB Fault Detection**", on ……. and has been found to be satisfactory by the jury members.

Jury Members

Prof. Dr. Hayriye KORKMAZ (Advisor)
Marmara University ....

Assoc. Prof. Mustafa Cem KASAPBAŞI (Jury Member)
Marmara University ....

Asst. Prof. Ali SARIKAŞ (Jury Member)
Marmara University ....

APPROVAL

Marmara University Institute for Graduate Studies in Pure and Applied Sciences Executive Committee approves that İzemnur BUDAK be granted the degree of Master of Science in Department of Electrical-Electronics, Electrical-Electronics Engineering Program on ……. (Resolution no: …..).

Director of the Institute
Prof. Dr. Bülent EKİCİ

İÇİNDEKİLER

ÖZET ..... **ii**
ABSTRACT ..... **iii**
KISALTMALAR ..... **iv**
ŞEKİL LİSTESİ ..... **v**
TABLO LİSTESİ ..... **vii**
1. GİRİŞ **1**
1.1 Hedefler ve Amaçlar 4
1.2. Literatür Taraması 5
2. GENEL BİLGİLER **9**
2.1. Derin Öğrenme ve Nesne/Kusur Tespiti 9
2.1.1. YOLO Tabanlı Derin Öğrenme Modeli 11
2.1.1.1. YOLOv4 13
2.1.1.2. YOLOv8 13
2.2. FPGA Tabanlı Geliştirme Araçları ve Platformlar 16
2.2.1. Vitis HLS 16
2.2.2. Vitis AI 16
2.2.3. PYNQ Platformu 17
2.2.4. Jupyter Notebook 18
2.3. FPGA Donanımı 19
2.4. Veri Seti Etiketleme Yöntemleri 21
2.4.1. Sınırlayıcı Kutu Etiketlemesi 21
2.4.2. Poligon Etiketlemesi 21
2.4.3. Anlamsal Segmentasyon 21
3. MATERYAL VE YÖNTEM **22**
3.1. Veri Seti ve Etiketleme 22
3.2. Make Sense 24
3.3. Derin Öğrenme Modelinin Geliştirilmesi 27
3.3.1. YOLOv4 Algoritması ile Model Eğitimi 27
3.3.2. YOLOv8 Algoritması ile Model Eğitimi 35
4. DERİN ÖĞRENME MODELİNİN FPGA ÜZERİNDE UYGULANMASI **43**
5. SONUÇLAR VE DEĞERLENDİRME **49**
5.1. Gelecek Çalışmalar 50
6. KAYNAKLAR **52**

ÖZET

FPGA Tabanlı PCB Hata Tespitinde Derin Öğrenme Uygulaması

Elektronik devre elemanlarının, devre yollarının ve kart boyutlarının oldukça küçülmesi, tümleşik devre elemanlarının bacak yapılarının ve montaj yöntemlerinin değişmesi; diğer taraftan birden fazla katmana sahip plakaların varlığı baskılı devre tasarımlarında oldukça karmaşık yapılar oluşturmaktadır. Bu karmaşıklıklar, PCB üretim sürecinde devre yollarının oluşturulmasından montaj aşamasına kadar her adımda çeşitli hatalara yol açabilmektedir.

PCB hataları, Plaka Üretimi Hataları ve Montaj Hataları olmak üzere iki ana kategoriye ayrılmaktadır. Geleneksel yöntemde, uzman operatörler plaka ve montaj hatalarını çıplak gözle veya mikroskop kullanarak kontrol ederler. Ancak bu yöntem (görsel inceleme), zaman alıcıdır, hatalara açıktır ve hız açısından seri üretime uygun değildir. Bu hataların tespit edilememesi cihaz arızalarına, performans sorunlarına ve ürün güvenilirliğinde düşüşlere yol açabilmektedir. Bu nedenle, daha etkili ve hızlı hata tespit sistemleri kullanmak çok önemlidir. Bu soruna çözüm olarak, PCB montaj hattının sonuna yüksek çözünürlüklü kameralar ile gelişmiş görüntü işleme ve derin öğrenme modellerinin entegre edilmesi önerilebilir. Bu önerilen sistem, hataları otomatik olarak tespit eder, insan hatalarını en aza indirir ve süreci hızlandırır.

Bu tez çalışmasında, PCB üretimi sürecinde karşılaşılan olası kusurları otomatik olarak tespit etmek için görüntü işleme tekniklerini ve makine öğrenme algoritmalarını kullanan bir yöntem önerilmektedir. Çalışmanın temelini oluşturan açık kaynaklı DeepPCB veri seti, çeşitli kusur türlerini barındıran 1194 PCB görüntüsünü içermektedir ve bu kusurlu görüntülerin üzerinde etiketlenmiş 7.899 kusurdan 6.385'i eğitim ve 1.514'ü test için kullanılmıştır (%80'i eğitim, %20'si test için). YOLOv4 ve YOLOv8 modelleri kullanılmış ve sonuçlar karşılaştırılmıştır. YOLOv8, Kria KV260 kartında yüksek doğrulukla üstün performans göstermiştir. Özellikle, delik hattı sınıfında %78,0 mAP@50 (IoU eşiğinde ortalama hassasiyet 0,50) ortalama hassasiyet gösteren YOLOv4 modeliyle karşılaştırıldığında, %98,99'luk etkileyici bir mAP değerine ulaşmıştır. Ayrıca, deneyler sonucunda YOLOv8 modelinin genel doğruluğunun %98,21 seviyesinde korunduğu belirlenmiştir.

ABSTRACT

FPGA Based Deep Learning Implementation for PCB Fault Detection

The significant reduction in electronic circuit elements, circuit paths and card sizes, the change in the pin structures of integrated circuit elements and their assembly methods; on the other hand, the existence of plates with more than one layer create quite complex structures in printed circuit designs. These complexities can lead to various errors at every step from the creation of circuit paths to the assembly stage in the PCB production process.

PCB defects are divided into two main categories: Plate Production Defects and Assembly Defects: In the traditional method, expert operators check the plate and assembly defects using the naked eye or microscope. However, this method (visual inspection) is time-consuming, prone to errors, and not suitable for mass production in terms of speed. Failure to detect these errors can lead to device failures, performance problems, and product reliability reductions. Therefore, it is very important to use more effective and faster defect detection systems. As a solution to this problem, it can be suggested to integrate high-resolution cameras, advanced image processing technics and deep learning models at the end of the PCB assembly line. This proposed system automatically detects errors, minimizes human errors, and speeds up the process.

In this thesis, a method that uses image processing techniques and machine learning algorithms to automatically detect possible defects encountered in the PCB manufacturing process is proposed. The open-source DeepPCB dataset that forms the basis of the study includes 1194 PCB images containing various defect types, and 6385 of 7899 defects labeled on these defective images were used for training and 1514 for testing (80% for training, 20% for testing). YOLOv4 and YOLOv8 models were used and results were compared. YOLOv8 showed superior performance with high accuracy on the Kria KV260 board. In particular, it achieved an impressive mAP value of 98.99% in the hole line class when compared with YOLOv4 model showing %78,0 mAP@50 (mean Average Precision at IoU threshold 0.50) average precision. In addition, as a result of the experiments, it was determined that the overall accuracy of the model was maintained at 98.21%.

KISALTMALAR

AI : Artificial intelligence

AOI : Automatic Optical Inspection

ANN : Artificial Neural Network

CM : Confusion matrix

CNN : Convolutional Neural network

CPU : Central Processing Unit

CUDA : Compute Unified Device Architecture

DNN : Deep Neural Network

FPGA : Field Programmable Gate Array

HDL : High Level Design

HLS : High Level Synthesis

IoU : Intersection over Union

PCB : Printed Circuit Board

SOM : System on Module

YOLO : You Only Look Once

ML : Machine Learning

mAP : Mean average precision

SSD : Single shot multibox detector

VGG : Visual geometry group

Faster R-CNN : Region-based convolutional neural network

TN : True Negative

TP :True Positive

FN : False Negative

FP : False Positive

FPN : Feature pyramid network

ŞEKİL LİSTESİ

Şekil 1: Geleneksel PCB Üretim Süreci Şeması 2
Şekil 2: Önerilen PCB Kusur Tespit Sisteminin Geleneksel PCB Üretim Sürecine Entegre Edilecek Kontrol Noktaları 4
Şekil 3: Yapay Zeka, Makine Öğrenmesi ve Derin Öğrenme Kavramları 9
Şekil 4: Yapay Zeka Uygulamaları a) Klasik Makine Öğrenmesi b) Derin Öğrenme 11
Şekil 5: YOLO Evrim Zaman Çizelgesi 12
Şekil 6: YOLOv8 Mimarisi 13
Şekil 7: Kria KV260 Uygulama Geliştirme Seçenekleri 16
Şekil 8: VITIS AI Yapısı 17
Şekil 9: Kria KV260 FPGA Kartı 19
Şekil 10: Uygulama Genel Mimarisi 20
Şekil 11: Sınıflara Göre Örnek Sayısı 23
Şekil 12: PCB Görselleri Üzerinde İşaretlenmiş Sınırlayıcı Kutuların Dağılımı a) Sınırlayıcı Kutuların Normalize Edilmiş Konum Verileri b) Sınırlayıcı Kutuların Normalize Edilmiş Boyut Değerleri 23
Şekil 13: PCB Görselleri üzerinde işaretlenmiş Sınırlayıcı Kutuların Dağılımına İlişkin Detaylı İstatistiksel Görselleştirme 24
Şekil 14: Make Sense ile Etiket Tanımlama 25
Şekil 15: Make Sense ile PCB Kusur İşaretlemesi 25
Şekil 16: Make Sense ile İşaretlenmiş bir görüntüye ait Txt Dosyası 26
Şekil 17: Kusurlu bölgenin Koordinatları a) X Koordinat Başlangıcı b) X Koordinat Sonu 26
Şekil 18: Google Drive ve Google Colab Klasör Yapısı 28
Şekil 19: Google Colab Veri Seti Eğitim Kodu 28
Şekil 20: Chart.png Dosyası 29
Şekil 21: PCB Veri Seti Eğitilirken İzlenen Ekran Görüntüsü 29
Şekil 22: PCB Veri Seti Eğitimi Tamamlandığında İzlenen Ekran Görüntüsü 30
Şekil 23: Eğitilmiş Veri Seti Ağırlık Dosyaları 30
Şekil 24: YOLOv4 Model Test Komutu 31
Şekil 25: Kusur Türü, Adedi ve Yüzde Doğruluk Oranı 32
Şekil 26: Test Sonucunu Görselleştirmek İçin Kullanılan Kod 32
Şekil 27: Test Sonucu 33
Şekil 28: Eğitim Görselleştirmesi 36
Şekil 29: Doğrulama Görselleştirmesi (Batch size=16 için) a) Etiketlenen Görseller (a1. validation _batch0, a2. validation _batch1, a3. validation _batch2) b) Tahmin Edilen Görseller (b1. validation_predict_batch0, b2. validation_predict_batch1, b3. validation_predict_batch2) 37
Şekil 30: Etiketlenen Kusurlu Bölgeler ile Tahmin Edilen Kusurlu Bölgelerin Detaylı Karşılaştırılması a) Etiketlenen Görseller b) Tahmin Edilen Görseller 39

Şekil 31: Çok Sınıflı Çoklu Kusur Tespiti İçin Karışıklık Matrisi a) Karışıklık Matriksi b) Normalize Edilmiş Karışıklık Matriks 40
Şekil 32: YOLOV8m Model Mimarisinin Görselleştirilmesi 40
Şekil 33: YOLOv8 Performans Metrikleri a)R-C Eğrisi b)P-C Eğrisi c)F1-C Eğrisi d) P-R Eğrisi 41
Şekil 34: YOLOv8 Modelinin Sonuçları a)train/box_loss Eğrisi b) train/cls_loss Eğrisi c) train/dfl_loss Eğrisi d)metrics/precision Eğrisi e)metrics/recall Eğrisi f) val/box_loss Eğrisi g) val/cls_loss Eğrisi h) metrics /map50 h) metrics /map50-95 42
Şekil 35: balenaEtcher ile SD Karta İmaj Dosyası Yükleme 43
Şekil 36: Kria KV260 Arayüz Bassğlantıları 44
Şekil 37: Jupyter Notebook Kod Hücreleri 45
Şekil 38: YOLOv8 Modelinde Sınıflara Ait Güven Skorları a) Kısa Devre b) Açık Devre c) Aşırı Aşınma d) Yetersiz Aşınma e) Sahte Bakır Parçalar f) Delik Hatlar 46
Şekil 39: Sınıfların Piksel Bazında Güven Skorları 47
Şekil 40: Uygulama Kurulumu 48
Şekil 41: YOLOv8 Modelinin Kria KV260 kartı Üzerinde Çalıştırıldığında Tahmin Sonuçları 48

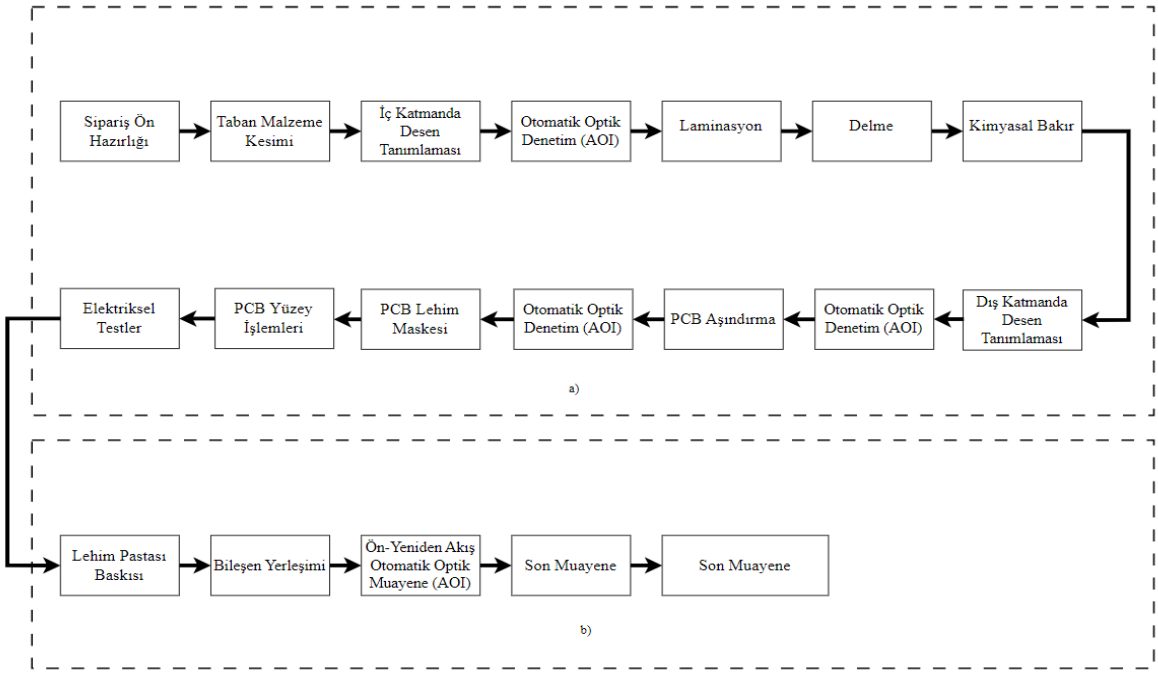
TABLO LİSTESİ

Tablo I. Veri Setindeki Sınıf Kusur Matrisi 7
Tablo II. Derin Öğrenme Tabanlı PCB Kusur Tespiti Üzerine Odaklanan Çalışmalar 7
Tablo III: Alternatif Derin Öğrenme Modelleri ve Özellikleri 12
Tablo IV. YOLO Sürümlerini Destekleyen Çerçeveler 15
Tablo V. PCB Kusur Tipleri ve Etiket Numaraları 22
Tablo VI. Deneysel Çevre Parametreleri 27
Tablo VII. YOLOv4 Modelinin Sınıflara göre Performans Metrikleri 33
Tablo VIII. YOLOv4 Model Sonuçları 34
Tablo IX. YOLOv8 Derin Öğrenme Modeli Hiper Parametreleri 36
Tablo X. FPGA Deneysel Ortam Yapılandırması 45
Tablo XI. Yolov8m Modelinin Platforma Göre Değerlendirilmesi 46
Tablo XII. YOLOv8 Modelinin Sınıflara Göre Performans Metrikleri 47
Tablo XIII. Kria KV260 Kartı Üzerinde Modelin Değerlendirme Parametreleri 50

1. GİRİŞ

Baskı Devre Kartları (Printed Circuit Board, PCB), güncel teknolojinin her alanında var olan elektronik cihazların en önemli bileşenlerinden biridir ve kullanımları artarak devam etmektedir. Bu nedenle sağlam ve hatasız PCB'lerin kullanımı, cihazların performansı, güvenilirliği ve ömrü için vazgeçilmezdir.

PCB üretim süreci, tasarımın karmaşıklığına bağlı olarak çeşitli kusurlara açıktır. Bu kusurlar bakır yolların plaka üzerinde aşındırma sürecinde olabileceği gibi, plaka üretimi tamamlandıktan sonra dizgi sürecinde de gözlemlenebilmektedir. Bu sebeple PCB üretim kusurları iki grup altında incelenebilir: 1) Plaka üretimi sırasındaki Kısa Devre, Açık Devre, Aşırı Aşındırma, Yetersiz Aşındırma, Sahte Bakır Parçalar, Delik Hatlar gibi kusurlar ve 2) Dizgi montaj hattında bileşenlerin yanlış yerleştirilmesi, istenmeyen lehim köprülerinin oluşumu gibi kusurlar. Plaka üretimi sırasındaki kusurlar, aynı zamanda 2 boyutlu; dizgi montaj hattındaki kusurlar ise 3 boyutlu kusurlar olarak da tanımlanabilmektedir. 2B görüşe dayalı kusur tespiti yalnızca iki boyutlu kusurları tespit edebilir ve bu da onu PCB çıplak kart kusurlarını yani plaka kusurlarını tespit etmek için daha uygun hale getirir. Şekil 1.a'da [81] nolu kaynaktan alıntı yapılan görsellerle PCB plaka üretim süreci gösterilmiştir. 3B görüşe dayalı PCB kusur tespiti, daha fazla derinlik bilgisi gereksinimleri olan kusurların tespiti için uygundur ve 2B görüşe dayalı denetimi tamamlayıcıdır [1]. Şekil 1.b'de ise, [3] nolu kaynaktan alıntı yapılan görseller kullanılarak PCB üretim sürecinin aşamaları verilmiştir.



Şekil 1: Geleneksel PCB Üretim Süreci Şeması a) Plaka Üretim Süreci b) Dizgi Üretim Süreci

Geleneksel yöntemde, PCB kusurlarının tespiti iki aşamalı bir süreç olarak gerçekleştirilmektedir. İlk aşamada, plaka üretimi tamamlandıktan sonra uzman operatörler, çıplak gözle veya mikroskop yardımıyla plakadaki potansiyel kusurları değerlendirirler. Bu kontrol aşamasını başarıyla geçen plakalar, montaj hattına yönlendirilir. İkinci aşamada ise, operatörler elektronik bileşenlerin yerleşimini ve yönlendirilmesini kontrol ederken, lehim miktarını da inceleyerek olası temassızlık veya kısa devre sorunlarını önlemeye çalışırlar.

Geleneksel yöntem, deneyimli personel gerektirir ve seri üretim hatlarında yavaş iş ve insan hatalarına karşı hassasiyetle eleştirilir. Bu nedenle bu metotla tespit edilemeyen kusurlar cihaz arızalarına, performans kaybına ve ürün güvenilirliğinin azalmasına neden olabilir. Bu nedenle PCB üretim sürecinde etkin, hızlı ve güvenilir kusur tespit metodolojilerinin kullanımı önem taşır.

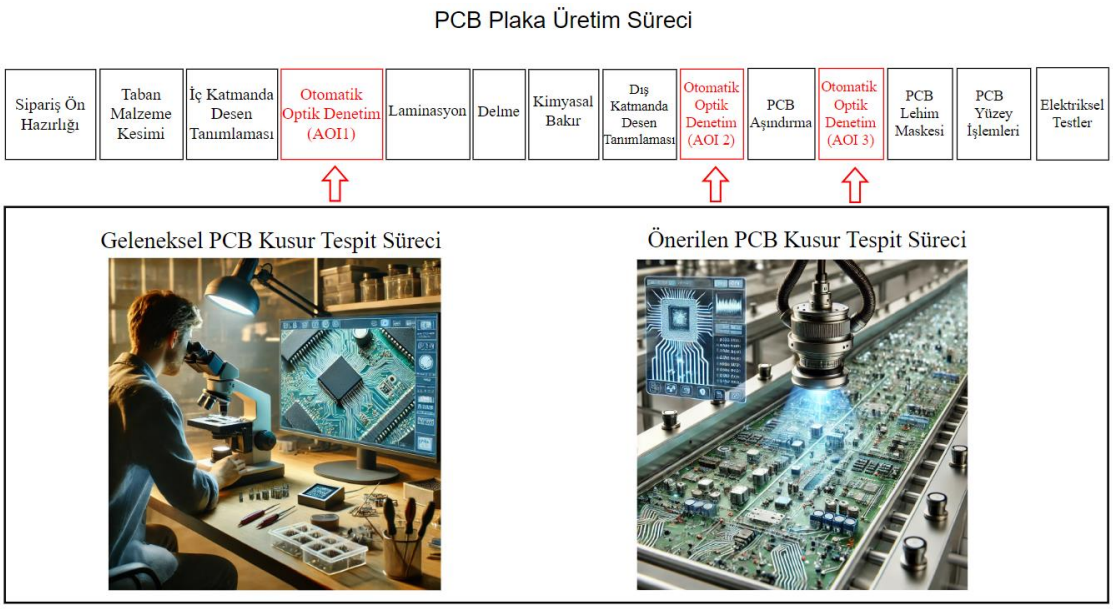
PCB plaka üretimi ve montaj hattına, yüksek çözünürlüklü kameralar, ileri düzey görüntü işleme algoritmaları ve derin öğrenme modelleri gibi teknolojik çözümlerin dahil edilmesi, insan kaynaklı hataları minimize ederken süreci otomize etmekte ve hızlandırmaktadır. Makine öğrenimi ve robotik otomasyon gibi teknolojiler, çalışanların iş verimliliğini artırma ve müşterilere daha hızlı, yüksek değerli ürünler sunma potansiyeline sahiptir. Operatörlerin uzman bilgisi ve deneyimi ile birleştirilen bu gelişmiş özellikler PCB kusur algılama sürecini daha da iyileştirecektir.

Sahada programlanabilir kapı dizileri (Field Programable Gate Array, FPGA) belirli bir görevin gereksinimlerini karşılamak üzere donanım düzeyinde yeniden yapılandırılabilir devrelerdir. Bu özelleştirilebilir yapıları ve paralel işleme yetenekleri nedeniyle derin öğrenme algoritmalarını uygulamak ve hızlandırmak için de uygun platformlardır.

Bu tez çalışmasında, plaka üretimi sırasında karşılaşılan Kısa Devre, Açık Devre, Aşırı Aşındırma, Yetersiz Aşındırma, Sahte Bakır Parçalar ve Delik Hatlar gibi kusurları tespit etmek için bir derin öğrenme modeli geliştirilmiş ve uygulanmıştır. Otomatik denetim (AOI - Automatic Optical Inspection) süreçlerini iyileştirmek amacıyla, üretim aşamasında gelişmiş görüntü işleme ve derin öğrenme teknikleri entegre edilebilir. Bu teknikler, PCB üzerindeki kusurların tespiti için kullanılan temel araçlar olup, verimlilik ve doğruluğu artırmayı hedeflemektedir. Bu bağlamda, yapay zeka tabanlı nesne tespit modelleri kritik bir rol oynamaktadır. Bu çalışmada, son teknoloji nesne tespit modeli YOLOv8, daha eski bir model olan YOLOv4 ile karşılaştırılmıştır. Yapılan değerlendirmeler sonucunda, YOLOv8 modelinin daha yüksek doğruluk oranları ile üstün performans gösterdiği belirlenmiştir. Ayrıca, YOLOv8 modelinin FPGA üzerinde uygulanabilmesi için gerekli optimizasyonlar yapılmış ve FPGA'ya uyumlu bir tasarım gerçekleştirilmiştir.

Şekil 2'de kullanılan geleneksel ve önerilen PCB kusur tespit süreci görselleri, OpenAI'nin Image Creator [2] modeli tarafından oluşturulmuş ve şuan kullanılan geleneksel yöntemin yerine önerilen derin öğrenme tabanlı kusur tespit sisteminin, geleneksel PCB plaka üretim sürecinde hangi aşamalara (denetim noktalarına) entegre edileceği gösterilmiştir.

PCB plaka üretim sürecinde, plaka üzerinde iç katman tasarımının işlenmesi sonrası (AOI1), dış katman tasarımının işlenmesi sonrası (AOI2) ve PCB aşındırma işlemi sonrasında (AOI3) olmak üzere her 3 denetim noktasında bu uygulama kullanılabilir. Kusurlar üretim sürecindeki aşamalara göre farklılık gösterebilmektedir. Bu kusurların tespiti için her aşama için ayrı modeller oluşturulabilir ya da tüm aşamaları kapsayacak tek bir model de kullanılabilir. Bu çalışmada oluşturulan model, kullanılan veri setindeki görüntüler tüm aşamaları kapsadığından her denetim noktasında uygulanabilir.



Şekil 2: Önerilen PCB Kusur Tespit Sisteminin Geleneksel PCB Üretim Sürecine Entegre Edilecek Kontrol Noktaları

Bu tezde ele alınan derin öğrenme tabanlı PCB kusur tespit çalışması, aşağıda sıralanan aşamalardan oluşmaktadır:

1) kusur özelliklerinin çıkarılması ve kategorize edilmesi
2) veri seti hazırlanması,
3) seçilen bir derin öğrenme algoritmasıyla derin öğrenme modelinin oluşturulması,
4) derin öğrenme modelin FPGA üzerinde çalıştırılması,
5) FPGA üzerinde çalışan modelin doğruluk, hız ve donanım kaynaklarının değerlendirilmesi ve iyileştirilmesi
6) PCB üretim hattına entegrasyon ve kontrol noktası olarak uygulanması.

1.1 Hedefler ve Amaçlar

Bu tez çalışmasında, PCB kusur tespiti için bir derin öğrenme modeli oluşturulup, bu modelin Xilinx Kria KV260 FPGA kartı üzerinde gerçeklenmesi amaçlanmıştır. Kria KV260, derin öğrenme çıkarımı için optimize edilmiş, performans ve güç verimliliği sağlayan bir FPGA kartıdır. Bu çalışmada öncelikle, PCB görüntüleri Make Sense uygulaması kullanılarak işaretlenerek bir veri seti oluşturulmuştur. Daha sonra bu veri seti, eğitim (%80) ve test seti (%20) olarak ikiye ayrılmıştır. Derin öğrenme modeli eğitiminde, YOLOv4 ve YOLOv8 nesne tespit algoritmaları kullanılmıştır. Her iki model için farklı bir FPGA gerçeklemesi önerilmiş; ancak model performansı daha iyi olduğu için sadece YOLOv8 modeli FPGA kartı üzerinde gerçeklenmiştir.

1.2. Literatür Taraması

Kusur tespiti için görsel tabanlı yaklaşım, endüstride kullanılan en yaygın prosedürlerden biridir. Ancak geleneksel görsel inceleme, değişken ve öznel sonuçlara sahip ölçülemeyen bir süreçtir. Bu durum araştırmacıları çözülmesi gereken belirli bir sorunun kendine has karmaşıklığı ve benzersizliği nedeniyle yeni otomatik kusur tespit sistemleri geliştirmeye yöneltmiştir [3]. Derin öğrenme ile geliştirilen otomatik kusur tespit sistemleri üretimi, optimize edilmiş akıllı tesislere dönüştürmektedir.

Bu tez çalışması için literatür taraması Web of Science (WOS) veri tabanında "object detection", defect detection" ve "deep learning" anahtar kelimeleri kullanılarak yapılmıştır. İlk filtrelemede 2021 ve sonrası tarihler için taranmış ve 720 çalışma olduğu görülmüştür. Bu çalışmalar incelendiğinde tekstil alanında kumaş hataları [4], [5], [6], [7], [8]; su altı organizmaların tespiti [9], [10]; tren raylarındaki yüzey kusurları [11], [12], [13], [14], [15]; yol yüzeylerindeki çatlak benzeri kusurların tespitinde [16], [17], [18]; tarımsal alanda meyvelerin ayrılması veya meyve üzerindeki çürüme gibi bozuklukların tespitinde [19], [20], [21], [22]; metal yüzeylerdeki kusur tespitinde [23], [24], [25], [26], [27]; kanalizasyon boruları veya çelik borular içerisindeki kusurlu bölgelerin tespitinde [28], [29], [30], [31], [32], [33], [34], [35]; trafikteki araba veya bisiklet gibi obje tespitinde [36] ve araba parçalarının kalite kontrol süreci [37] gibi farklı alanlarda yapılmış çalışmaların olduğu görülmüştür.

Bu tez çalışmasında amaç aynı zamanda geliştirilen derin öğrenme modelinin FPGA üzerinde çalıştırılmasıdır. Bu yüzden "defect detection", "deep learning" ve "FPGA" anahtar kelimeleri kullanılarak 2021 ve sonrası tarihler için ikinci bir tarama yapılmış ve bu defa sadece 7 çalışma olduğu görülmüştür. Bu çalışmalar incelendiğinde Fotovoltaik (PV) panel yüzeyindeki kusur tespitinde [38], sürücü destek sistemlerinde yavaşlama rampası, su çukuru, çukur, rögar kapağı, su birikintisi gibi yol üzerindeki nesnelerin tespitinde [39], yarı iletken parçaların yüzey kusurlarında [40] , tren raylarındaki ray bağlantı elemanı kusur tespitinde [41], meyve dış kusurlarının tespitinde [42], ultrasonik kılavuzlu dalgalarda veri odaklı hasar değerlendirme yöntemlerinde [43] ve ultrasonik test sistemi kullanılarak bir çelik blok içindeki kusurların tespitinde [44] olmak üzere bir kaç çalışmanın olduğu görülmüştür. Yapılan tarama "defect detection", "deep learning", "FPGA" ve "PCB" anahtar kelimeleriyle özelleştirildiğinde ise hiç çalışma olmadığı görülmüştür.

Bu yüzden “object detection”, defect detection”, “deep learning” ve “PCB” anahtar kelimelerinin kullanıldığı üçüncü bir tarama yapılmış ve 2021 yılından itibaren 25 adet çalışma bulunmuştur. PCB yüzey kusurlarını tespit eden [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57] ve [58] numaralı çalışmalar incelenmiştir. Tablo I ’de literatürde yapılan PCB kusur tespit çalışmaları, çalışmanın yapıldığı yıl, tüm sınıflar için ortalama kesinlik (mAP) değerleri ve tespit edilen kusurlu nesne karşılaştırması sunulmuştur. Ayrıca [59] ve [60] numaralı çalışmalar WOS veri tabanında yer almakla beraber yukarıda belirtilen anahtar kelimeler ile sorgulama yapıldığında bulunamamaktadır.

Literatürdeki tüm çalışmalar incelendiğinde nesne (kusur) tespiti için sınıf sayısına ve nesne(kusur) miktarına bağlı performans farklılıkları olduğu görülmüştür. Bu tez çalışmasında kullanılan DeepPCB veri setinde, her bir PCB görüntüsünde aynı veya farklı sınıflara ait birden fazla kusur bulunmaktadır [62].

Tablo I’de kusur tespit süreçlerinde karşılaşılan, bir görselde yer alabilecek sınıf sayısı ile bir sınıfa ait kusur miktarını açıklayan veri seti sınıf-kusur matrisi oluşturulmuştur. Bu matrise göre derin öğrenme tabanlı kusur tespitleri, genellikle dört ana kategori altında incelenebilir: tek sınıf-tek kusur tespiti, tek sınıf-çok kusur tespiti, çok sınıf-tek kusur tespiti, çok sınıf-çok kusur tespiti. Tek sınıf tek kusur tespiti (Single-Class, Sinle-Instance Defect Detection) daha basit senaryolar için kullanılır, burada algoritma mevcut görüntüde var olan yalnızca tek bir kusur sınıfına (türüne) ait bir kusuru tespit etmektedir. Yani görüntüde etiketlenmiş sadece bir tane kusur (kusurlu bölge) bulunmaktadır. Tek sınıf çok kusurlu tespitte ise (Single-Class, Multi-Instance Defect Detection), bir görüntüde aynı kusur türünden birden fazla kusur (birden fazla işaretlenmiş kusurlu bölge) bulunabilmektedir. Çok sınıf tekli kusur tespitinde (Multi-Class, Sinle-Instance Defect Detection), model birden fazla kusur tespiti için eğitilmiş olsa bile bir görüntüde sadece tek bir kusurun olmasına karşılık gelmektedir. Çok sınıflı-çoklu kusur tespiti (Multi-Class, Multi-Instance Defect Detection) ise model yine birden fazla kusur sınıfının tespitine göre eğitilmiştir ve bir görüntüde aynı anda birden fazla aynı veya farklı sınıfa ait kusurlar bulunabilmektedir.

Sınıf-kusur matrisini bir meyve tabağı ve içindeki meyvelerin tespit edilmesi örneği üzerinde incelemek, bu durumu anlamaya yardımcı olacaktır. Bir meyve tabağı görselinde yalnızca elmaların olduğunu varsayalım. Burada elma sayısının bir veya birden fazla olması sınıf sayısını değiştirmez ve bu durum “tek sınıf” olarak adlandırılır. Elmaların üzerinde sadece

çürük olarak işaretlenmiş kusur algılanmaya çalışılıyorsa “tek sınıf-tekli kusur (nesne)” denir. Ancak elmaların üzerinde böcek yemesi, çürüme, lekeli gibi kusurlar işaretlenmişse bu durum, “tek sınıf-çoklu kusur (nesne)” kategorisine girmektedir.

Diğer bir durumda, meyve tabağında hem elma hem de kivilerin olduğunu varsayalım. Bu durum "çok sınıf" kategorisinde değerlendirilir. Bu kategoride, böcek yemesi, çürüme, leke gibi kusurlardan sadece bir tanesi bulunuyorsa "çok sınıf-tekli kusur (nesne); bahsedilen kusurlardan en az 2 tanesi bulunuyorsa bu duruma da "çok sınıf-çoklu kusur" denir. Bu ayrımlar, kusur tespit sürecinde hem sınıf sayısının hem de kusur miktarının etkilerini daha ayrıntılı analiz edebilmek adına önemli bir çerçeve sunmaktadır.

Tablo I. Veri Setindeki Sınıf Kusur Matrisi

Sınıf Sayısı	**Görseldeki Kusur Miktarı**
Tek Sınıf	Tekli Kusur
Tek Sınıf	Çoklu Kusur
Çok Sınıf	Tekli Kusur
Çok Sınıf	Çoklu Kusur

Tablo II. Derin Öğrenme Tabanlı PCB Kusur Tespiti Üzerine Odaklanan Çalışmalar

Referans	**Yıl**	**Veri Seti**	**Görsel Boyutu (piksel)**	**Sınıf Sayısı**	**Görseldeki Kusur Durumu**	**Model**	**mAP^{val} @50 (%)**	**mAP @50:95 (%)SSS**
[45]	2021	274 görüntü üzerinde 324 kusur bulunmaktadır.	3840 x2160 pikselden 50x50 piksele düşürülmüştür.	6	Çoklu Kusur	CNN	96.40	
[46]	2021	23.000 görüntü kullanılmış olup, eğitim amaçlı 20.700 görüntü ve test amaçlı 2300 görüntü olmak üzere ayrılmıştır.	-	2 sınıf var. Kusurlu veya kusursuz	Çoklu Kusur	YOLOv5l	99.74	
[49]	2021	1455 eğitim görüntüsü, 624 test görüntüsü içerir.	-	6	Çoklu Kusur	Feature Pyramid Network (FPN)	44.3	-
[59]	2024	İnternet kaynakları ve Roboflow'un geniş PCB arıza görüntüleri koleksiyonunun birleşiminden derlenmiştir.	-	5	Tekli Kusur	YOLOv8x	98.9	-
[60]	2022	2654 orijinal PCB yüzey kusuru görüntüsü veri artırma işleminden sonra, 18.572 görüntüye çıkarılmıştır. 18.572 görüntü üzerinde 37675 adet kusur bulunmaktadır.	960x960 800x800 640x640 480x480	6	Tekli Kusur	YOLOX	99.13	-
Bu çalışma	2024	1194 görüntü üzerinden etiketlenmiş 7.899 adet kusur bulunmaktadır.	640x640	6	Çoklu Kusur	YOLOv8m	90.4	38.29

[18] numaralı çalışma, Çok sınıf tek kusurlu türden olup; mAP@50 değeri %97'dir. [10] numaralı çalışma ise, Çoklu sınıf çoklu kusur özelliğiyle bu tez çalışmasındakine benzer sınıf-kusur durumuna sahiptir ve mAP@50 değeri %84,7 dir. Bu filtreleme yöntemi dışında kalan ama YOLOv8 ile yol kusur tespiti yapan [61] nolu çalışma da incelenmiştir.

Tablo II'de ise tez çalışmasının da konusu olan PCB kusur tespiti üzerine yapılan çalışmalar incelenmiştir.

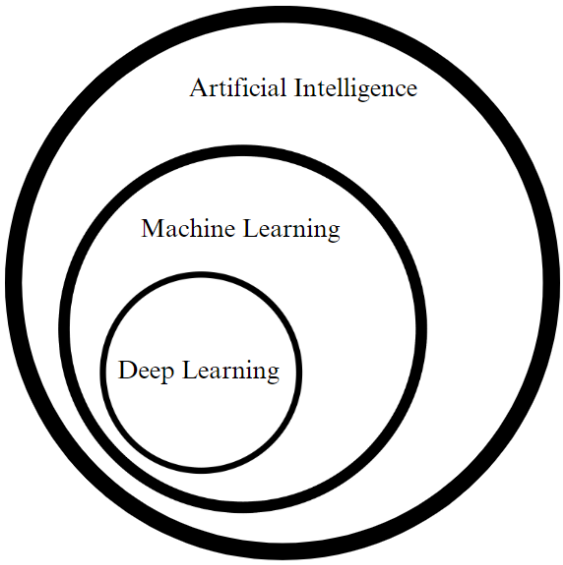
Bu tez çalışmasında kullanılan DeepPCB veri setinde, her bir PCB görüntüsünde aynı veya farklı sınıflara ait birden fazla kusur bulunmaktadır [62].

2. GENEL BİLGİLER

Bu bölüm, derin öğrenme tabanlı kusur algılama yöntemlerini, PCB yüzey kusurlarını belirlemek için seçilecek sinir ağı modellerini ve özelliklerini içerir. Bu bölümde YOLOv4 ve YOLOv8 derin öğrenme algoritmaları ayrıntılı olarak incelenmiş ve seçilen algoritmaya göre FPGA üzerinde modelin nasıl çalıştırılacağı açıklanmıştır. Bu tez çalışması başlangıçta, YOLOv4 algoritması ile oluşturulan modelin Vitis AI kullanılarak Kria KV260 kartı üzerinde çalıştırılması şeklinde planlanmış olsa da, YOLOv4 modelinin performansının düşük olduğu görüldükten sonra YOLOv8 ile model eğitimi yapılmıştır. Ancak Vitis AI, YOLOv8 i desteklememektedir. Bu nedenle, çalışmada YOLOv8'in etkin bir şekilde kullanılabilmesi için alternatif bir yaklaşım olarak PYNQ platformunun kullanılması tercih edilmiştir. Bu bölümde her iki durum için gerekli olan yazılım ve donanım araçlarının genel tanımları yapılmıştır. Uygulamaya yönelik adımlar ise 3.bölümde açıklanmıştır.

2.1. Derin Öğrenme ve Nesne/Kusur Tespiti

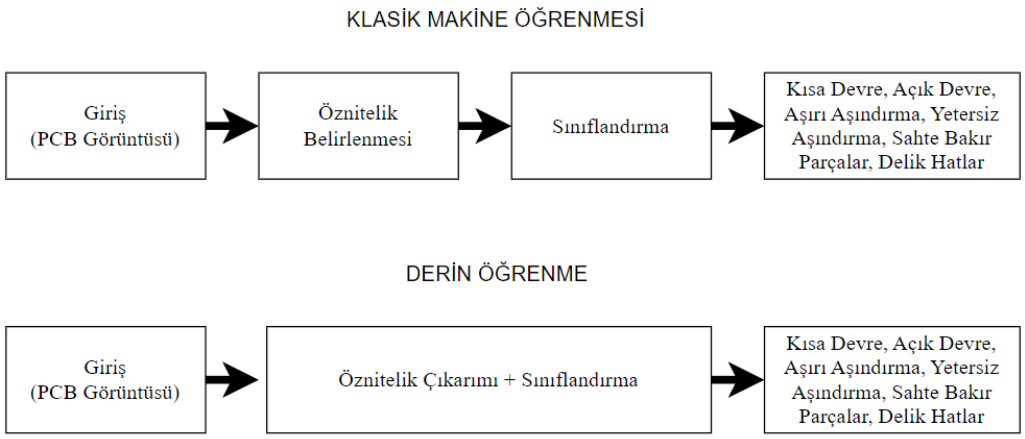
PCB görselleri üzerinde bulunan 6 farklı kusuru sınıflandırmak için 3 farklı uygulama geliştirilebilir. İlk uygulama türü, insan zekası ile kural temelli olarak türetilen yazılımla problem çözümü yapmaktır. Bu çözümde her kusur için belirli koşullar tanımlamanız gerekir. Örneğin 2 desen arası boşluk varsa açık devre, değilse diğer durumlar şeklinde bir algoritma kurulmalıdır. Ancak bu, bir yapay zeka uygulaması değildir. Yapay zeka kuralları kendisi keşfeder. Şekil 4'de gösterildiği gibi yapay sinir ağı sistemi (YSA) üzerinde makine öğrenmesi ve derin öğrenme teknolojisi çalışır [63].



Şekil 3: Yapay Zeka, Makine Öğrenmesi ve Derin Öğrenme Kavramları [64]

PCB görselleri üzerindeki kusurları sınıflandırmak için kullanılabilecek diğer yöntemler sırasıyla klasik makine öğrenmesi ve derin öğrenmedir. Makine Öğrenmesi, bilgisayarların verilerden anlam çıkarıp öğrenmesi ve zamanla performanslarını iyileştirebilmesini sağlayan bir yapay zeka alt dalıdır. Makine öğrenimi özellik çıkarma ve model inşasını farklı bir şekilde gerçekleştirir ve her modül adım adım oluşturulur. Burada el yapımı özellikler, öncelikle ham verileri farklı bir alana (örneğin istatistiksel, frekans ve zaman-frekans alanı) dönüştürerek uzman alan bilgisi gerektiren temsili bilgileri almak için çıkarılır [65]. Derin Öğrenme ise, klasik makine öğrenmesinin daha ileri bir alt dalı olup, çok katmanlı yapay sinir ağları kullanarak verilerden otomatik olarak özellik çıkarabilmektedir ve bu süreçte insan müdahalesini en aza indirmektedir [64]. Minimum insan çıkarımına sahip uçtan uca bir öğrenme yapısıdır. Bu yüzden bu tez çalışmasında makine öğrenimi yerine derin öğrenme modeli tercih edilmiştir.

Şekil 4 'te de gösterildiği üzere, klasik makine öğrenmesinde öznitelik çıkarımı insan tarafından yapılarak etiketli veri ile eğitilir, sınıflandırma aşaması makine öğrenmesinin kendisi tarafından yapılır. Derin öğrenmede ise öznitelik çıkarımı ve sınıflandırmayı derin öğrenme modeli kendisi yapar. Makine öğreniminden farklı olarak büyük veri kümeleri üzerinde eğitilir ve görüntü verilerindeki karmaşık örüntüleri ve anomalileri otomatik olarak öğrenebilir, bu da derin öğrenmeyi PCB kusur tespitinde kullanmak için ideal hale getirir. Bir başka avantajı ise derin öğrenme algoritmalarında problem zorlaştıkça katman sayısı arttırılabilir. Derin öğrenme algoritmalarında girişe yakın katmanlarda öznitelikleri tespit etmeye yani öznitelikleri anlamaya yönelik katmanlar yer alır. Girişte ilk öznitelik çıkarım katmanı genel öznitelikleri keşfeder. Sonraki katmanlar girişten uzaklaştıkça daha detaylı özellikleri çıkarır. Sonrasında ise sınıflandırma katmanları başlar. Bu özellik büyük veri setleri ve karmaşık problemler üzerinde makine öğrenmesine kıyasla daha yüksek performans sunmasını sağlamaktadır.



Şekil 4: Yapay Zeka Uygulamaları a) Klasik Makine Öğrenmesi b) Derin Öğrenme

Nesne tespiti, bir görüntüdeki nesneleri bularak bunların konum ve sınıf olasılıklarını tanımlamayı amaçlar. Derin öğrenmeye dayalı nesne dedektörleri ise iki ana kategoriye ayrılmaktadır: 1) Bir aşamalı dedektörler, 2) İki aşamalı dedektörler (Örneğin R-CNN). YOLO (You Only Look Once) serisi modeller, (YOLO, YOLOv2, YOLOv3, YOLOv4) yaygın olarak kullanılan tek aşamalı dedektörlerdir [66].

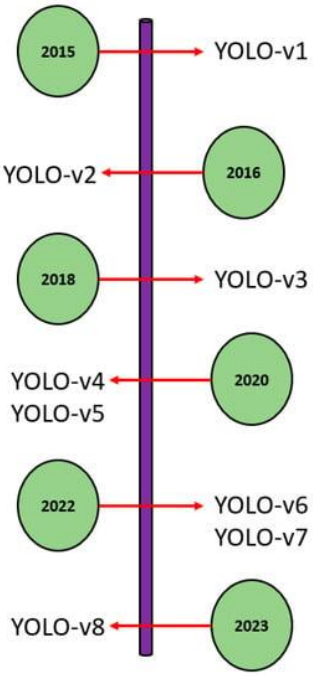
2.1.1. YOLO Tabanlı Derin Öğrenme Modeli

YOLO tabanlı derin öğrenme modelleri nesne tespiti ve sınıflandırma görevlerinde üstün performans göstermektedir ve gerçek zamanlı nesne algılama uygulamaları için çok daha hızlıdır [67],[68]. Ayrıca YOLO, hem CPU hem de GPU'larda çalışacak şekilde optimize edilebilir. Bundan dolayı bu çalışmada PCB kusurlarının otomatik tespiti için YOLO tabanlı derin öğrenme modelleri kullanılmıştır.

YOLO nesne dedektörleri serisi, bilgisayarlı görüş alanında önemli bir etki yapmıştır. Joseph Redmon ve arkadaşları tarafından 2015'te tanıtılan YOLO-v1, sınırlayıcı kutu (bounding box) tahmini ve sınıflandırma işlemlerini tek bir süreçte birleştiren yenilikçi bir mimari ile piyasaya sürüldü. Ardından, YOLO-v2'de toplu normalleştirme, yüksek çözünürlük ve çapa kutuları (anchor boxes) gibi mimari gelişmeler tanıtıldı. Çapa kutuları, belirli yükseklik ve genişlik değerlerine sahip önceden tanımlanmış kutulardır. Bu kutuların yükseklik ve genişlik değerleri veri kümesinde bulunan ve algılanmak istenilen nesnelerin büyüklüklerine göre değişebilmektedir. YOLO-v3, 2018'de, nesnellik puanları ve çok ölçekli tahmin yetenekleri gibi ilerlemelerle piyasaya sürüldü ve özellikle küçük nesnelerde performansı artırıldı.

2020'de, Joseph Redmon'un katkıda bulunmadığı ilk model olan YOLO-v4, gelişmiş özellik

toplama ve yeni aktivasyon fonksiyonları ile geldi. YOLO-v5, PyTorch tabanlı olarak kısa süre sonra yayınlandı ve GitHub üzerinden geniş bir kitleye ulaştı. YOLO-v6 ve YOLO-v7, sırasıyla donanım verimliliğine ve mimari yeniliklere odaklanarak, performansı artırmak için devam eden iyileştirmeler sundu. Son olarak, 2023'te Ultralytics tarafından piyasaya sürülen YOLO-v8, benzer donanım parametreleriyle üstün performans sergileyerek, serinin en son gelişmelerini temsil etmektedir. Şekil 5, YOLO'nun zaman içindeki evrimini gösteren bir zaman çizelgesini içermektedir.



Şekil 5: YOLO Evrim Zaman Çizelgesi [69]

YOLO dışında kullanılabilecek diğer derin öğrenme modelleri de bulunmaktadır. Alternatif olarak önerilebilecek bazı modeller Tablo III ’de gösterilmiştir.

Tablo III: Alternatif Derin Öğrenme Modelleri ve Özellikleri

Alternatif Derin Öğrenme Modeli	**Avantajı**	**Dezavantajı**
Faster R-CNN (Region-based Convolutional Neural Network)	Yüksek doğrulukta nesne tespiti yapar ve nesne sınırlayıcı kutularını daha kesin olarak ayarlar.	YOLO 'ya göre daha yavaştır, bu da gerçek zamanlı uygulamalarda sınırlayıcı olabilir.
SSD (Single Shot MultiBox Detector)	YOLO gibi tek seferde tespit yapar ve hızlı çalışır. Çeşitli nesne ölçeklerini desteklemek için çoklu çözünürlüklerde özellik haritaları kullanır.	YOLO kadar hızlı olmayabilir, ancak genellikle daha doğru tespitler sağlar.
Mask R-CNN	Nesne tespitinin yanı sıra nesne maskelerini de sağlar.	Yüksek hesaplama gücü gerektirir.

Bu çalışmada YOLO modelinin tercih edilmesinin temel sebepleri şu şekilde özetlenebilir:

- **Hız:** YOLO, görüntüleri tek bir işlemde değerlendirme yeteneği ile bilinir, bu da modelin hızlı çalışmasını sağlar. Bu özellik, seri üretim hatlarında gerçek zamanlı kusur tespiti yapılmasına olanak tanır ve üretim süreçlerinin verimliliğini artırır.
- **Doğruluk:** YOLO, çeşitlendirilmiş ve kapsamlı veri setleri üzerinde eğitildiğinde yüksek doğrulukta tespitler gerçekleştirebilir. Bu, PCB kusurlarının hassas bir şekilde sınıflandırılmasını ve yanlış pozitif oranlarının azaltılmasını sağlar, böylece ürün kalitesi ve güvenilirliği artar.
- **Esneklik:** YOLO, farklı boyut ve şekillerdeki nesneleri başarıyla tespit edebilme kapasitesine sahiptir. Bu esneklik, PCB üretiminde karşılaşılan çeşitli kusur türlerini (örneğin, kısa devreler ve lehim problemleri) efektif bir şekilde algılamak için idealdir.
- **Ölçeklenebilirlik:** YOLO, farklı çözünürlüklerdeki görüntülerle uyumlu şekilde çalışabilir. Böylece sistem gereksinimlerine göre kolayca özelleştirilebilir. Bu özellik, modelin farklı üretim ortamlarına ve değişen işlem kapasitelerine uyum sağlamasını mümkün kılar.

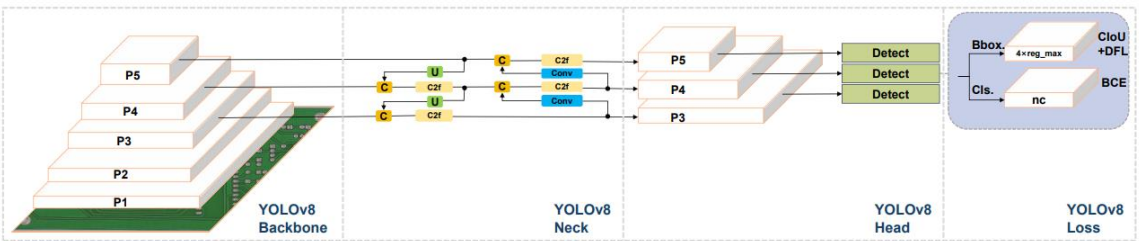
2.1.1.1. YOLOv4

YOLOv4, giriş görüntüsünü bir ızgaraya bölen ve her ızgara hücresi için birden çok sınırlayıcı kutu ve sınıf olasılığı tahmin eden tek aşamalı bir nesne detektörüdür [70].

Bu tez çalışmasında PCB kusur tespiti için YOLOv4 ün seçilmesinin nedeni, yüksek kare hızı ve düşük gecikme süresi ile bilinen gerçek zamanlı bir nesne detektörü olmasıdır ve bu da onu gerçek zamanlı PCB otomatik denetleme sistemleri için uygun hale getirmektedir. YOLOv4, YOLOv8'e kıyasla daha eski bir modeldir ve düşük donanım gereksinimlerine sahiptir. Eski model GPU'larda YOLOv8 modelinde uyumsuzluklar yaşanabilmektedir. Bu durumlarda YOLOv4 tercih edilmektedir.

2.1.1.2. YOLOv8

YOLOv8, giriş ön işleme, omurga ağı, boyun modülü ve çıktı işleme olmak üzere dört modüllü tek aşamalı nesne algılama yöntemidir. YOLOv8 mimarisi Şekil *6*'da gösterilmiştir.



Şekil 6: YOLOv8 Mimarisi [71]

Ultralytics tarafından YOLOv8, nesne algılama, segmentasyon ve poz tahmini için en gelişmiş performansı sunmak üzere tasarlanmıştır. YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l ve YOLOv8x gibi çeşitli model versiyonları bulunmaktadır. Uygulamanın hem performans gereksinimine hem de üzerinde çalıştırılacak gömülü sistemin hafıza kısıtlarına bağlı olarak uygun model tercihi yapılmaktadır.

YOLOv8, PyTorch, TorchScript, ONNX, OpenVINO, TensorRT, CoreML, TF SavedModel, TF GraphDef, TF Lite, TF Edge TPU, TF.js, PaddlePaddle ve NCNN gibi birçok farklı formatta modelin dışa aktarımını desteklemektedir [72]. Dışa aktarma, modelin farklı platformlar ve cihazlar üzerinde çalışabilmesi için çeşitli formatlara dönüştürülmesi anlamına gelmektedir. Bu sayede model, çeşitli donanım ve yazılım ortamlarında kullanılabilir hale getirilebilmektedir.

YOLOv8, YOLOv4'e göre tasarımcı için kolaylaştırılmış eğitim süreçlerine sahiptir. Derin öğrenme modeli oluşturulmadan önceki hazırlık aşamalarında tasarımcı elindeki veri setine göre daha az ayarlama ve hesaplama yapmaktadır. Örneğin Max_batches değeri oluşturulması ve filtre sayısı hesaplaması gibi. Benzer şekilde eğitim sonrasında Yolov4'te test ve eğitim görsellerindeki kusur örnek sayısı, karışıklık matrisi, doğruluk oranı, ortalama kesinlik, duyarlılık ve F1 Skoru kısmen manuel hesaplamalarla elde edilirken, YOLOv8'de daha kolay hesaplanmakta ve grafikleri daha kolay oluşturulmaktadır.

2.1.2. Derin Öğrenme için Kullanılan Çerçeveler

Çerçeve (framework), programcıların karmaşık uygulamalar geliştirmesini kolaylaştıran, bir dizi araç, kütüphane ve yapı taşlarından oluşan bir yazılım altyapısıdır. Derin öğrenme çerçeveleri, özellikle yapay sinir ağlarının tasarımı, eğitimi ve değerlendirilmesi için optimize edilmiş yazılımlar sunar. Bu çerçeveler, araştırmacıların ve mühendislerin veri kümelerini daha kolay işleyebilmesini, modelleri eğitebilmesini ve gerçek dünya uygulamalarına entegre edebilmesini sağlar. Bir derin öğrenme çerçevesi, sinir ağlarının oluşturulmasında ve model eğitimi sırasında kullanılan temel işlemleri optimize eder.

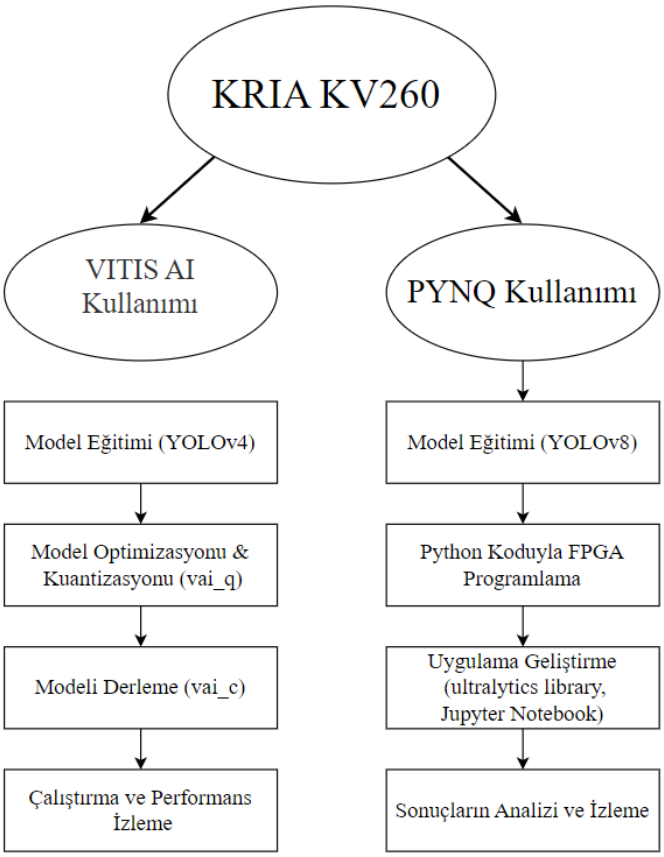
Bu bağlamda, derin öğrenme için kullanılan popüler çerçeveler Tablo IV'te listelenmiştir. Bu çerçeveler, farklı derin öğrenme modellerini geliştirmek ve eğitmek için yaygın olarak kullanılır. Bazı çerçeveler özellikle YOLO gibi nesne algılama modellerini desteklemek için öne çıkmıştır. Tablo IV, her çerçevenin YOLO modelleriyle olan uyumluluğunu ve Darknet ve PyTorch gibi popüler çerçevelerin hangi YOLO sürümlerini desteklediklerini açıkça göstermektedir.

Tablo IV. YOLO Sürümlerini Destekleyen Çerçeveler

Çerçeve	**Açıklama**	**YOLO Modelleri**
Darknet	C ve CUDA ile yazılmış, YOLO'nun ilk geliştirildiği çerçeve	YOLOv1, YOLOv2, YOLOv3, YOLOv4
PyTorch	Esnek ve dinamik hesaplama grafiği sağlayan popüler derin öğrenme çerçevesi	YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, YOLOv8
TensorFlow	Google tarafından geliştirilen geniş kapsamlı derin öğrenme çerçevesi	YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, YOLOv8
MXNet	Amazon tarafından geliştirilen, verimli ve dağıtık eğitim desteği sunan çerçeve	YOLOv3

FPGA'lar, paralel işleme yetenekleri sayesinde yoğun hesaplama gerektiren CNN işlemlerini hızlandırmak için idealdir. FPGA ve FPGA tabanlı Çip Üzerinde Sistem (System on Chip, SoC) teknolojisi, yoğun bilgi işlem yüklerini hızlandırabilmektedir. Kria KV260 kartı üzerinde CNN (Convolutional Neural Networks) kullanılarak gerçek zamanlı PCB kusur tespit uygulamaları geliştirilebilmektedir. Görüntü sınıflandırma, bölümleme ve nesne algılama gibi çeşitli uygulamalar için CNN'lerin düşük gecikme ve güç tüketimiyle gerçek zamanlı olarak uygulanması yapılmaktadır [73].

Çoğunlukla derin öğrenme modelleri GPU’lar üzerinde çalışırken, günümüzde gecikmeyi ve güç tüketimini azaltma avantajlarından dolayı bu modellerin FPGA ve benzeri gömülü sistemler üzerinde çalıştırılmasına başvurulduğu görülmektedir. Bu modellerin gömülü sistemler üzerine yüklenmesi için bazı ara süreçlerin izlenmesi gerekmektedir. Şekil 7’de görüldüğü gibi bu tezde kullanılan donanımın (Kria KV260) üzerinde modelin çalışabilmesi için izlenebilecek 2 farklı yol bulunmaktadır:1) Vitis AI, 2) PYNQ.



Şekil 7:Kria KV260 Uygulama Geliştirme Seçenekleri

2.2.FPGA Tabanlı Geliştirme Araçları ve Platformlar

Aşağıda bu tez çalışması kapsamında Kria KV260 FPGA kartının özelliklerini kullanarak Vitis ve PYNQ araçları 2 farklı şekilde yazılım geliştirilmesine olanak sağlar. Bu bölümde Vitis ve PYNQ'nun özellikleri açıklanmıştır.

2.2.1. Vitis HLS

Vitis HLS, Xilinx'in gelişmiş bir yüksek düzey sentez (HLS) aracıdır ve Xilinx'in Vitis teknoloji platformunun bir parçası olarak geliştirilmiştir. Yüksek Düzey Sentezleme (High-Level Synthesis, HLS), donanım tasarımcılarının geleneksel donanım tanımlama dilleri (HDL) yerine; C, C++ veya SystemC gibi yüksek seviyeli programlama dillerini kullanarak donanım tanımlamalarına olanak tanıyan bir yöntemdir. Vitis HLS, daha önce Xilinx tarafından sunulan Vivado HLS aracının yeniden markalanmış ve genişletilmiş bir versiyonudur [74].

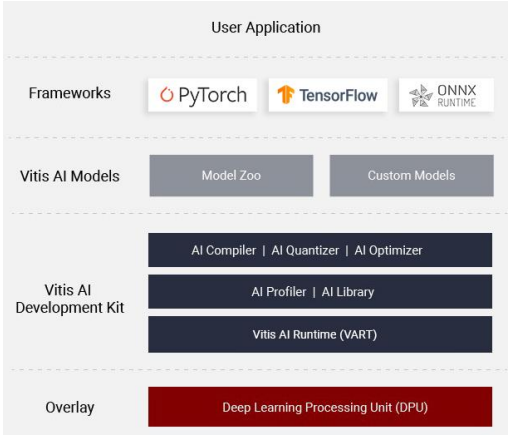
2.2.2. Vitis AI

Xilinx tarafından geliştirilen Vitis AI, yapay zeka ve makine öğrenimi modellerinin Xilinx donanımlarında uygulanması için optimize edilmiş bir yazılım platformudur. Ancak, AMD'nin

2021'de Xilinx'i satın almasından sonra, Vitis AI ve diğer Xilinx teknolojileri artık AMD bünyesinde yer almaktadır.

Vitis AI, Kria KV260 gibi AMD cihazları için kapsamlı bir yapay zeka çıkarım geliştirme çözümüdür. Yapay zeka modelleri için optimize edilmiş derin öğrenme işlemci birimi (DPU) çekirdekleri, araçları, kütüphaneleri, modelleri ve örnek tasarımlarından oluşan zengin bir setten oluşur [75].

Vitis AI, modelleri optimize etmek, derlemek ve bunları FPGA'da hızlandırılmış çıkarım için dağıtmak için bir dizi araç ve kütüphane sağlar. Bu platform Şekil 8'de gösterildiği gibi TensorFlow, PyTorch gibi popüler derin öğrenme çerçevelerinden ve ONNX Runtime üzerinden çalışan modelleri destekler ve bu modelleri FPGA üzerinde çalıştırılabilir hale getirir [76]. ONNX Runtime, farklı çerçevelerden eğitilmiş modellerin taşınabilirliğini sağlayan ve bu modellerin performansını optimize eden bir çalışma zamanı sistemidir. Vitis AI platformu, bu çerçevelerle uyumlu Model Zoo, özelleştirilmiş modeller ve çeşitli geliştirme araçları içerir. Bu araçlar arasında AI Compiler, AI Quantizer ve AI Optimizer bulunmakta olup, modellerin Xilinx'in donanım hızlandırıcılarında verimli bir şekilde çalıştırılmasını sağlar. Vitis AI Runtime (VART), uygulamaların Deep Learning Processing Unit (DPU) üzerinde hızlandırılmış çıkarım yapılmasını destekler. Bu entegre yaklaşım, Vitis AI'nın yapay zeka tabanlı projelerde kapsamlı bir çözüm sunmasını sağlar, endüstriyel uygulamalardan veri merkezlerine kadar geniş bir kullanım alanına hitap eder.



Şekil 8: VITIS AI Yapısı [76]

2.2.3. PYNQ Platformu

PYNQ, yani Python Productivity for Zynq, Xilinx tarafından geliştirilen ve özellikle Zynq SoC'ler ve FPGA'lar için tasarlanmış bir açık kaynak platformudur. PYNQ, FPGA'ların gücünden yararlanmak isteyen yazılım geliştiricileri ve araştırmacıları hedef alarak, donanım

tasarımı süreçlerini basitleştirmeyi amaçlar. Bu platform, Python programlama dilinin erişilebilirliği ve kolaylığı ile donanım tasarımının karmaşıklığını bir araya getirir.

PYNQ platformu, Python dili üzerinden FPGA donanımını programlama imkanı sunar. Böylelikle geliştiriciler, VHDL veya Verilog gibi donanım tanımlama dillerini öğrenmeden FPGA üzerinde uygulama geliştirme imkanına kavuşur. PYNQ, Jupyter Notebook arayüzünü kullanarak, web tabanlı bir geliştirme ortamı sunar ve kullanıcıların Python scriptleri ile FPGA üzerinde doğrudan IP bloklarını kontrol etmelerine olanak tanır.

PYNQ platformu, çeşitli kütüphaneleri ve IP entegrasyonlarını destekler, böylece kullanıcılar donanım hızlandırıcılarından, sensörlerden ve iletişim protokollerinden kolayca yararlanabilir. Ayrıca, PYNQ topluluğu tarafından sunulan pek çok önceden hazırlanmış projeye ve koda erişim sağlar, bu da geliştirme sürecini daha da hızlandırır ve kolaylaştırır.

Kria KV260 kartı gibi modern Xilinx donanımları, PYNQ platformu ile uyumlu çalışabilir [77],[78]. Bu da geliştiricilere AI, görüntü işleme ve IoT gibi alanlarda zengin uygulamalar geliştirme imkanı sunar. PYNQ'un bu esnek ve kullanıcı dostu yaklaşımı, FPGA teknolojisini daha geniş bir yazılım geliştirici kitlesine açar ve karmaşık donanım projelerinin üzerindeki engelleri azaltır.

2.2.4. Jupyter Notebook

Jupyter Notebook, açık kaynaklı, web tabanlı bir uygulama olup, özellikle veri analizi, makine öğrenimi ve bilimsel hesaplamalar gibi alanlarda kullanılır. Kullanıcılara hem kod yazma hem de çıktıları aynı platformda görme imkanı sunar. Jupyter Notebook'un en büyük avantajlarından biri, hücre tabanlı bir yapıya sahip olmasıdır. Bu yapısı sayesinde kodları adım adım çalıştırabilir, her bir adımın sonucunu anında görebilirsiniz. Bu hem hataları tespit etmeyi kolaylaştırır hem de işlemleri daha esnek hale getirir.

Jupyter Notebook ayrıca interaktif bir çalışma ortamı sağlar. Yani, kodların çıktıları doğrudan ekranda görüntülenir ve grafiksel veri görselleştirmeleri kolayca sunulabilir [17]. Bu da özellikle veri analizi ve makine öğrenimi gibi alanlarda büyük avantaj sağlar.

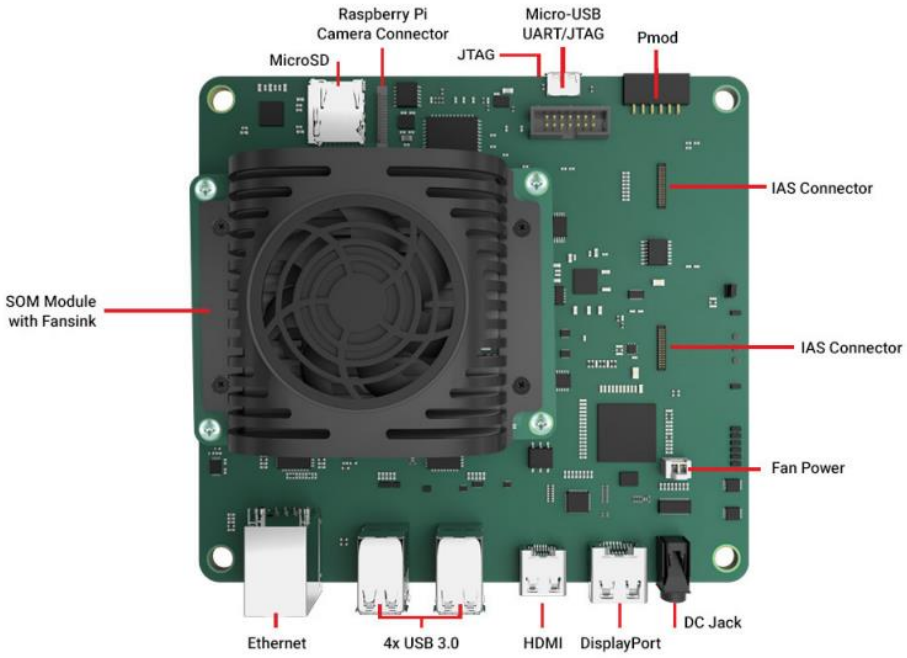
Bu tez çalışmasında, Python kodları Ubuntu 22.04 işletim sistemi üzerinde çalışan Jupyter Notebook ile yazılmış ve Kria KV260 FPGA kartına yüklenerek FPGA üzerinde koşturulmuştur. Jupyter'in interaktif ve hücresel yapısı, proje üzerinde yapılan denemeleri ve analizleri hızlandırmış ve esneklik kazandırmıştır.

2.3.FPGA Donanımı

Alanda programlanabilir kapı dizileri (FPGA) hazır satılan entegre devrelerdir. FPGA'ların enerji tüketimi açısından verimli olması bir avantaj oluştururken, düşük seviyeli donanım tasarımları nedeniyle de zorluk oluşturmaktadır.

Kria KV260 Vision AI Starter Kit, Xilinx tarafından geliştirilmiş, özellikle uç (edge) cihazlardaki yapay zeka uygulamalarını hızlandırmak için tasarlanmış bir FPGA kartıdır. Bu kart, AI ve görüntü işleme projelerinde kullanılmak üzere optimize edilmiştir ve geniş bir kullanım yelpazesi sunar. Özellikle endüstriyel, perakende ve şehir güvenlik uygulamalarında etkili çözümler sağlamaktadır [13]. KV260, güçlü bir işlem kapasitesine sahip olan Xilinx'in Kria SoM (System on Module) serisinin bir parçasıdır ve programlanabilir mantık birimi ile birleştirilmiş ARM Cortex işlemciler içerir. Bu bileşim, hem esneklik hem de yüksek performans sağlar, kullanıcılara çeşitli uygulama senaryoları için uygun bir platform sunar.

Kria KV260, Xilinx'in Vitis AI platformu ile uyumlu çalışır ve bu platformun sunduğu AI Compiler, AI Quantizer ve AI Optimizer gibi araçları kullanarak AI modellerini FPGA üzerinde etkin bir şekilde çalıştırmak için gerekli optimizasyonları yapabilir. Şekil 9'da Kria KV260 FPGA Kartı'nın fiziksel görünüşü yer almaktadır.



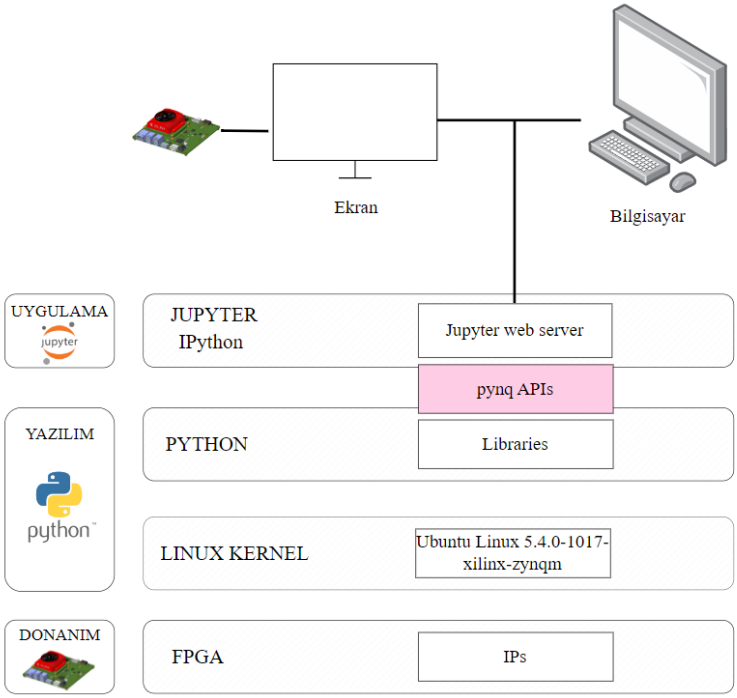
Şekil 9: Kria KV260 FPGA Kartı [79]

Daha önce de ifade edildiği gibi Kria KV260 FPGA kartında yapay zeka uygulamalarını çalıştırmak için iki temel yöntem bulunmaktadır: 1)Vitis AI ve 2) PYNQ.

Bu tez çalışmasında Kria KV260 FPGA kartı için PYNQ platformu tercih edilmiştir. Çünkü PYNQ, Python programlama dilini kullanarak FPGA üzerinde uygulamalar geliştirmeyi kolaylaştıran bir platformdur. Python, özellikle veri bilimi, makine öğrenimi ve yapay zeka toplulukları arasında popülerliği ve erişilebilirliği nedeniyle geniş bir kullanıcı kitlesine sahiptir. PYNQ platformu, bu popüler dili kullanarak FPGA'ların güçlü işlem kapasitesinden yararlanmayı mümkün kılar, böylece kullanıcılar karmaşık donanım tanımlama dillerini öğrenmeye gerek kalmadan doğrudan yüksek seviyede donanım tasarımları yapabilirler.

PYNQ ayrıca, Jupyter Notebook üzerinden interaktif bir geliştirme deneyimi sunar. Bu da, prototiplerin hızlı bir şekilde geliştirilmesine ve test edilmesine olanak tanır. Bu interaktif ortam, gerçek zamanlı olarak kod değişikliklerinin etkilerini gözlemlemeyi ve hızlı bir şekilde iterasyon yapmayı kolaylaştırır. Ayrıca, PYNQ geniş bir topluluk desteği ile birçok hazır IP bloğu ve kütüphanesine de sahiptir. Bu da geliştirme sürecini daha da hızlandırır ve kullanıcıların farklı ihtiyaçlarına uyum sağlar [80].

Bu tez çalışmasına ait mimari şema Şekil 10'da verilmiştir. Uygulamanın detayları bir sonraki bölümde açıklanmıştır.



Şekil 10: Uygulama Genel Mimarisi

2.4. Veri Seti Etiketleme Yöntemleri

Derin öğrenme modellerini eğitmek için, görsel verilerin kusur türü ve konumu dahil olmak üzere doğru bilgilerle etiketlenmesi gerekir. Bu işlem etiketleme (veya işaretleme) olarak bilinir ve derin öğrenme tabanlı PCB kusur tespit sisteminin geliştirilmesinde önemli bir adımdır. Etiketleme teknikleri, işaretlenecek görsele de bağlıdır. Bazı veri seti işaretleme/etiketleme yöntemleri aşağıda sıralanmıştır:

2.4.1. Sınırlayıcı Kutu Etiketlemesi

Bu teknik, her kusur örneğini çevreleyen dikdörtgen bir sınırlayıcı kutu çizmeyi ve ona karşılık gelen kusur türüyle etiketlemeyi içerir. Sınırlayıcı kutu işaretlemesi, uygulaması nispeten kolay olduğu için yaygın olarak kullanılan bir tekniktir. Bununla birlikte, düzensiz şekilli veya örtüşen kusurlar olduğunda doğru bir şekilde etiketleme yapılamayabilir. Bu tez çalışmasında da [51]'de olduğu gibi sınırlayıcı kutu etiketlemesi tercih edilmiştir.

2.4.2. Poligon Etiketlemesi

Poligon işaretlemesi, daha yüksek doğruluk gerektiren uygulamalar için daha uygun bir etiketleme yöntemidir. Bu teknik, her kusur örneğinin sınırını hassas bir şekilde izlemek için bir dizi nokta (köşeler) kullanmayı içerir. Poligon işaretlemesi, özellikle karmaşık şekilli veya örtüşen kusurlar için daha doğru işaretlemeler sağlar. Çeşitli coğrafyaları ve kentsel dokuları kapsayan [81] numaralı çalışma, poligon etiketlemesine örnek olarak verilebilir.

2.4.3. Anlamsal Segmentasyon

Bu teknik, görüntüdeki her pikseli karşılık gelen kusur türüyle etiketlemeyi içerir. Anlamsal segmentasyon, kusurun şekli ve boyutu hakkında ayrıntılı bilgiler sağlar ve bu da daha doğru kusur tespitine ve sınıflandırmasına imkan tanır. Bununla birlikte, diğer işaretleme tekniklerine göre daha fazla zaman alıcı ve hesaplama yükü açısından yoğundur.

3. MATERYAL VE YÖNTEM

Bu bölümde görsel veri seti tanımlama ve etiketleme süreci açıklanmıştır. Ardından derin öğrenme modeli için hem YOLOv4'ün hem de YOLOv8'in eğitim ve performans sonuçları incelenmiştir.

3.1. Veri Seti ve Etiketleme

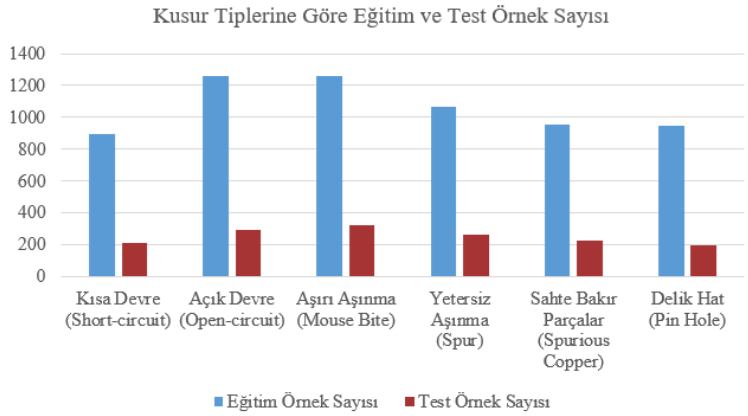
Bu tez çalışmasında açık kaynak DeepPCB veri kümesi kullanılmıştır [62]. Kullanılan veri setinde Tablo V'de isimleri verilen 6 farklı kusur türü bulunmaktadır. Şekil 11'de görüldüğü üzere 1194 görüntüdeki 7.899 etiketli kusurdan 6.385'i eğitim ve 1.514'ü test için kullanılmıştır. Endüstriyel standartlara uygun olarak toplanan 16k × 16k piksel boyutundaki yüksek çözünürlüklü ham görüntüler, 640 × 640 piksellik alt görüntülere dönüştürülüp hizalanmıştır. Görüntüler, aydınlatma kaynaklı bozulmaları azaltmak amacıyla ikili hale getirilmiştir.

Bu veri setinde bulunan görüntüler, %80'i eğitim ve %20'si test amaçlı olmak üzere iki kısma ayrılmıştır. Veri kümesi, Kısa devre, Açık devre, Aşırı Aşınma, Yetersiz Aşınma, Sahte Bakır Parça, Delik Hat olmak üzere 6 farklı türde kusur sınıfı içermektedir. Sınıfların farklı durumları şu şekilde açıklanabilir: Tek bir sınıftan bir veya birden fazla etiket, birden fazla sınıftan bir veya birden fazla etiket. Bu veri setinin bir başka özelliği de bir görüntünün birden fazla sınıfta kusur içermesi ve aynı sınıf kusurun görüntünün farklı koordinatlarında yer almasıdır.

Tablo V. PCB Kusur Tipleri ve Etiket Numaraları

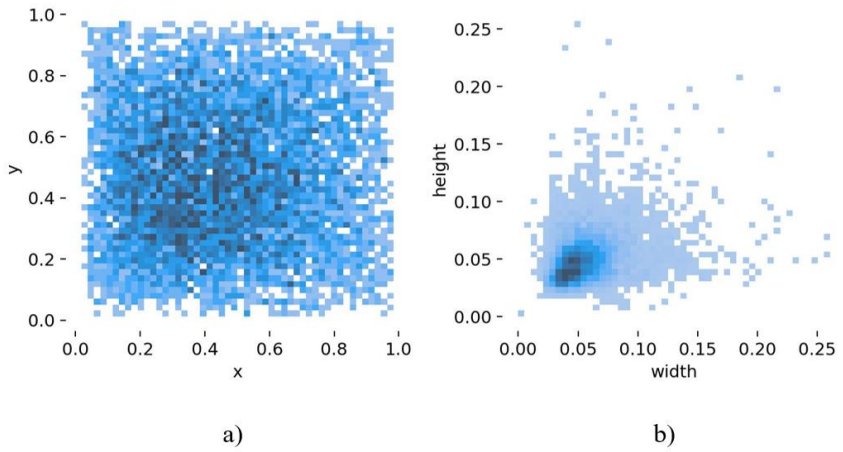
Kusur Tipi	**Etiket Numarası**
Kısa Devre (Short-circuit)	0
Açık Devre (Open-circuit)	1
Aşırı Aşınma (Mouse Bite)	2
Yetersiz Aşınma (Spur)	3
Sahte Bakır Parçalar (Spurious Copper)	4
Delik Hatlar (Pin Hole)	5

Veri setinin yapılandırılmasına ve sınıflandırılmasına ilişkin daha derinlemesine bir anlayış kazanmak amacıyla, Şekil 11'de farklı PCB kusur sınıflarına ayrılan eğitim örneklerinin ve test süreci için ayrılmış örneklerin sınıflar arası dağılımı gösterilmektedir.

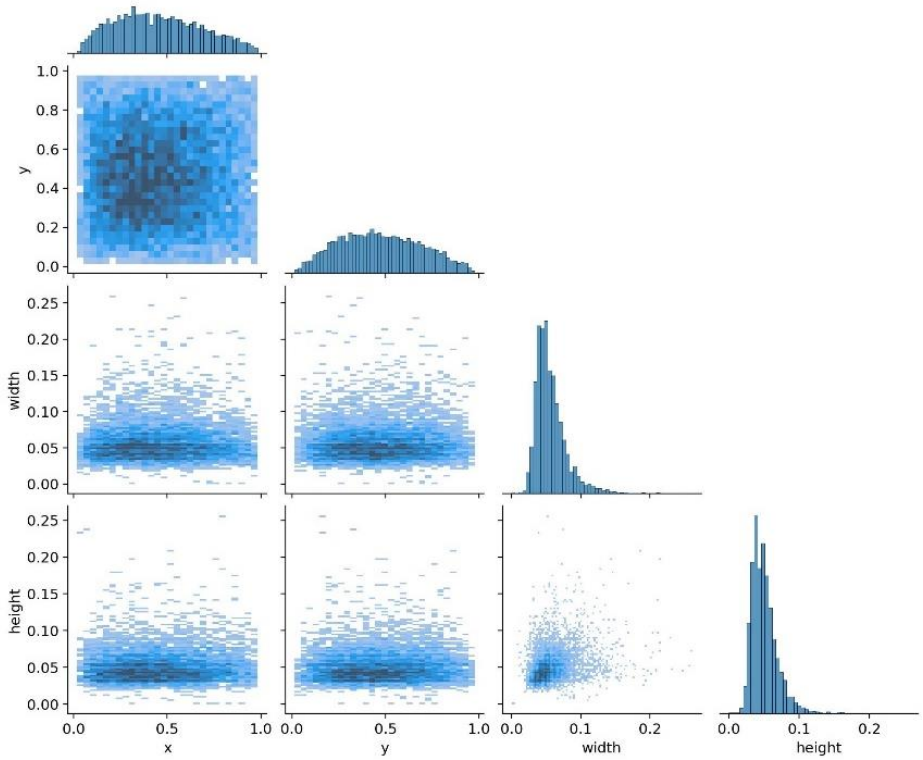


Şekil 11: Sınıflara Göre Örnek Sayısı

Şekil 12.a, mavi renkli işaretlenmiş PCB görüntülerinde sınırlayıcı kutuların (bounding boxes) yani kusurların, PCB kartlarının yüzeyinde oldukça homojen bir şekilde dağılmış olduğunu ortaya koymaktadır. Koyu mavi renk veri setinde ilgili piksel konumunda var olan kusur miktarının fazla olduğunu ifade etmektedir. 640 x 640 piksel boyundaki görüntüler 1,0 x 1,0 boyutundaki grafiğe normalize edilmiştir. Şekil 12.b'de ise işaretlenen sınırlayıcı kutuların boyutunu temsil etmektedir. Yani 1,0 x 1,0 piksel boyutundaki görsel için etiket boyutu genellikle 0,05 x 0,05 piksel boyutunda olacaktır. Bu da 640 x 640 piksellik bir görselde işaretlenen sınırlayıcı kutu boyutunun 32 x 32 piksel boyutuna sahip olacağı anlamına gelmektedir. Dolayısıyla bu tez çalışmasında yapılan işlem, küçük nesne/kusur tespiti olarak kategorize edilebilir. Şekil 13'de ise, Şekil 12'in detaylandırılmış ve sütun grafiği hali yer almaktadır.



Şekil 12: PCB Görselleri Üzerinde İşaretlenmiş Sınırlayıcı Kutuların Dağılımı a) Sınırlayıcı Kutuların Normalize Edilmiş Konum Verileri b) Sınırlayıcı Kutuların Normalize Edilmiş Boyut Değerleri

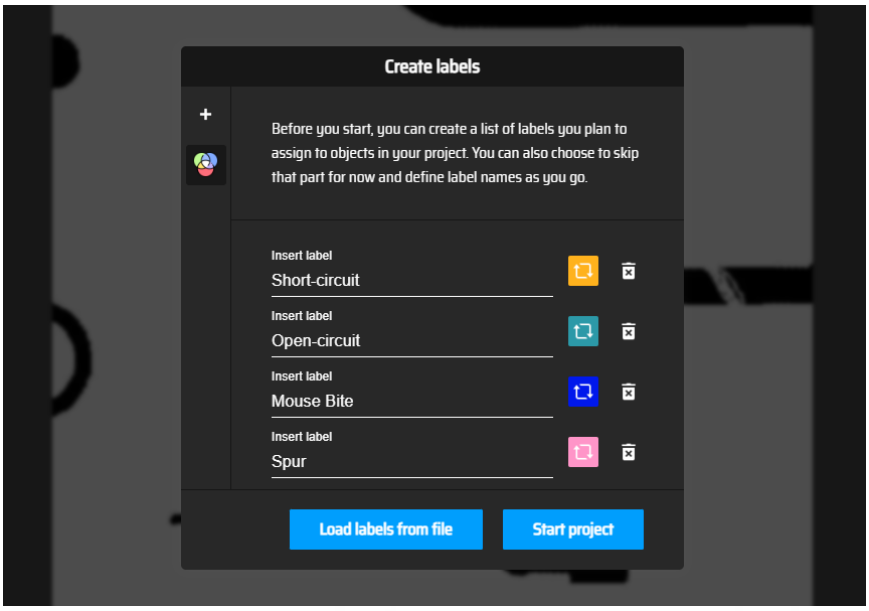


Şekil 13: PCB Görselleri üzerinde işaretlenmiş Sınırlayıcı Kutuların Dağılımına İlişkin Detaylı İstatistiksel Görselleştirme

3.2.Make Sense

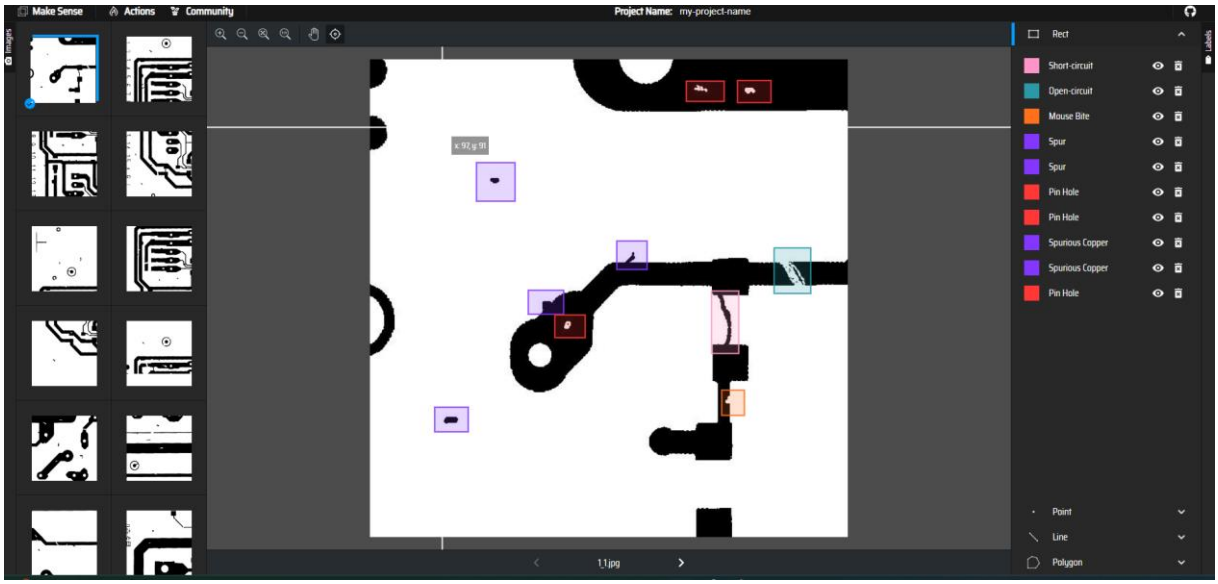
Görsel verilerin işaretlemesi için web tabanlı veya masaüstü uygulamalar mevcuttur. Web tabanlı uygulamalara Make Sense [2], Labelbox [82], SuperAnnotate [83], Roboflow [84], Supervisely [85], VIA (VGG Image Annotator) [86] gibi uygulamalar örnek verilebilirken; masaüstü uygulamalarına ise LabelImg [87], Labelme [88], RectLabel örnek olarak verilebilir. Web tabanlı araçlar, herhangi bir kurulum gerektirmeden hızlıca başlamak için idealdir. Masaüstü araçlar ise genellikle daha fazla özelleştirme ve çevrimdışı çalışma imkanı sunar. Bu tez çalışmasında web tabanlı olması ve ücretsiz kullanım sağlamasından dolayı Make Sense sınırlayıcı kutu işaretlemesi tercih edilmiştir.

Görsel işaretleme işlemi zaman aldığından veri setindeki görseller gruplara bölünerek işaretlenmiştir. İlk olarak “Get Started” tıklanarak işaretlenecek resimler seçilir. Ardından “Object Detection” ile işaretleme yapılacak sınıflar girilir. Burada önemli olan gruplandırılacak etiket isimleri aynı sıra ile ve İngilizce olarak yazılmalıdır. (Bakınız Şekil 14)



Şekil 14: Make Sense ile Etiket Tanımlama

Şekil 15'te görülen sayfanın sol tarafında veri setinden yüklenilen resimler ve sağ tarafında ise label bölümü yer almaktadır. 'Create labels' ekranında sizin nesnelerinizin hangi sınıfa ait olduğunu belirten etiketler olacaktır. Pencerenin sol tarafında bulunan '+' sembolüne basarak Short-circuit, Open-circuit, Mouse Bite, Spur, Spurious Copper, Pin Hole olacak şekilde altı ayrı etiket eklenmiştir. Ardından 'Start project' butonuna tıklanmalıdır. Proje açıldıktan sonra Şekil 15'te görüldüğü gibi görsel üzerindeki kusurlar işaretlenir.



Şekil 15: Make Sense ile PCB Kusur İşaretlemesi

MakeSense, YOLO, VOC XML, CSV, VGG JSON, COCO JSON formatlarını desteklemektedir. Bu, eğitim ve çıkarım süreçlerinde farklı performans ve kullanım senaryoları sağlamaktadır. Make Sense'nin YOLO çıktısı her bir resim için Şekil 16'daki gibi bir txt dosyasıdır.

Her bir satırda yer alan 5 değer sırasıyla "etiket tipi, x1, y1, x2, y2" formatında belirtilmiştir. YOLO v4 algoritmasında, işaretlenmiş nesnelerin konumları ve boyutları, görselin boyutlarına göre normalize edilir. Bu normalizasyon işlemi, nesnenin orta noktasının x koordinatı (x_center), y koordinatı (y_center), genişliği (width) ve yüksekliği (height) üzerinden yapılır. x_center ve y_center değerleri, sırasıyla, görselin genişliğine ve yüksekliğine bölünerek hesaplanır. Bu normalizasyon sonucu elde edilen koordinatlar, nesnenin sınırlayıcı kutusunun köşe koordinatlarını hesaplamak için kullanılır.

```
0 0.035714 0.595899 0.044974 0.054233
5 0.595899 0.359788 0.056878 0.055556
2 0.500000 0.398810 0.071429 0.067460
4 0.680556 0.481481 0.070106 0.071429
4 0.792989 0.418651 0.091270 0.091270
3 0.685185 0.615079 0.063492 0.050265
1 0.289683 0.248016 0.063492 0.038360
```

Şekil 16: Make Sense ile İşaretlenmiş bir görüntüye ait Txt Dosyası

Nesnenin sınırları Denklem (1), (2), (3) ve (4) kullanılarak hesaplanır:

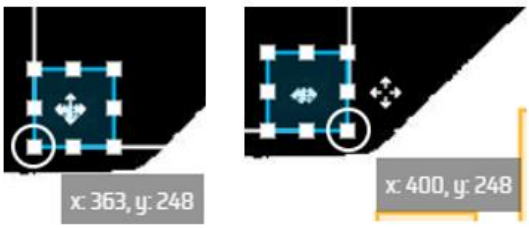
x1 = x_center - (width / 2) (1)

y1 = y_center - (height / 2) (2)

x2 = x_center + (width / 2) (3)

y2 = y_center + (height / 2) (4)

Bu yöntem, YOLO algoritmasının nesne tespiti için kullandığı görüntülerdeki nesnelerin tam olarak yerini ve boyutunu doğru bir şekilde yansıtmak üzere tasarlanmıştır.



Şekil 17: Kusurlu bölgenin Koordinatları a) X Koordinat Başlangıcı b) X Koordinat Sonu

Veri setinde 640 x 640 piksel boyutuna sahip Şekil 17'de verilen görselde, Şekil 16'da gösterilen etiket sınıfı 5 olan (2.satırdaki) nesnenin x koordinatının başlangıç ve bitiş değerleri Şekil 17 2.a ve 2.b' de gösterilmektedir. Bu nesnenin merkez x koordinatı x_center = [(400-363)/2]+363]/640= 0.6 ve width = (400-363)/640= 0.06 olarak hesaplanır. Buna göre, x1 yaklaşık olarak x1= 0,6- 0,03= 0,57 hesaplanır. Şekil 16'da, bu değerin daha kesin olarak 0,595899 olarak hesaplanmış olduğunu görülmektedir.

3.3.Derin Öğrenme Modelinin Geliştirilmesi

Derin öğrenme modellerinin verimli bir şekilde eğitilmesi ve çalıştırılması için yüksek performanslı bilgi işlem cihazlarına ihtiyaç duyulmaktadır. Bu cihazlar arasında GPU, NPU, TPU ve TensorRT gibi donanımlar yer alır. Google Colab, kullanıcıların web tarayıcıları aracılığıyla güçlü GPU'lar ve TPU'lar dahil olmak üzere Google'ın bulut tabanlı hesaplama kaynaklarına erişmesini sağlayan ücretsiz bir bulut platformudur. Kullanıcı dostu arayüzü, önceden yüklenmiş kütüphaneleri ve iş birliği özellikleri ile Google Colab, derin öğrenme modellerinin eğitimi ve deneysel çalışmaları için uygun bir ortam sunar.

3.3.1. YOLOv4 Algoritması ile Model Eğitimi

Bu çalışmada kullanılan deneysel ortam Google Colab, derin öğrenme çerçevesi ise Darknet'tir. Deneysel parametreler Tablo VI'da verilmiştir.

Tablo VI. Deneysel Çevre Parametreleri

Çevresel Parametre	**Değeri**
CUDA Versiyon	12.2.0
cuDNN Versiyon	8.9.6
GPU	Tesla T4

İlk işlem veri kümesini eğitim, doğrulama ve test kümelerine bölmektir. Eğitim kümesi, modelin ağırlıklarını ve önyargılarını eğitmek; doğrulama kümesi hiper parametreleri ayarlamak ve aşırı uyumu önlemek için; test kümesi ise modelin performansının tarafsız bir değerlendirmesini sağlamak için kullanılır.

Google Colab, varsayılan olarak Darknet çerçevesini içermez. Bu nedenle Darknet'in öncelikle kurulması gerekmektedir. Bu süreç, Darknet'in GitHub deposundan gerekli bağımlılıkların yüklenmesini ve kaynak kodunun derlenmesini içermektedir.

Darknet, YOLOv4 modelini yapılandırmak için bir yapılandırma dosyası (.cfg) kullanır. Bu dosya, ağ mimarisini, katman türlerini, filtre sayısını ve diğer hiper parametreleri tanımlar. PCB kusur tespiti görevini ve kullanılan veri kümesini eşleştirmek için yapılandırma dosyası ayarlanmıştır.

Google Colab klasör yapısı, Şekil 18'de görülmektedir. Google drive içerisine YOLOv4 yapılandırma dosyları zip halinde yüklenmiştir. Ardından Şekil 18'de gösterildiği gibi Google Drive içerisine Google Colab ortamı açılmıştır. Eğitim işlemi Google Colab ortamında Python dilinde yapılmaktadır. Drive hesabınızı colab ortamına bağlayarak ilk

önce zip dosyası extracted_files dosyasına çıkarılarak Google Drive'a kaydedilmiştir. Drive'a çıkarılmış dosyalar kaydedilmezse eğitim işlemi bittikten sonra oluşan ağırlık dosyaları kısa süre içinde silinmektedir.



Şekil 18: Google Drive ve Google Colab Klasör Yapısı

YOLOv4 sisteminin algılamaları CPU'dan 100 kat daha hızlı işleyebilmesi için Colab dizüstü bilgisayarında GPU hızlandırma etkinleştirilmiştir. Ardından aşağıdaki kodlar sırasıyla çalıştırılmıştır. Şekil 19'daki kodlar [89], [90] ve [91] nolu uygulamalardan derlenmiştir.

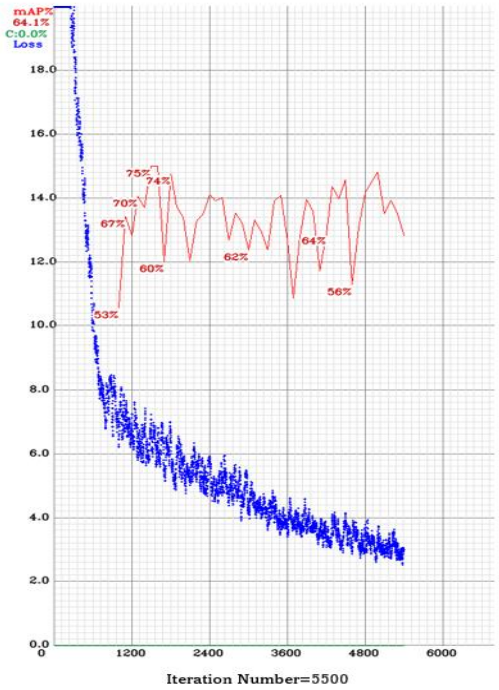
```
!apt-get update
!unzip "/content/drive/MyDrive/colabEgitim/darknet-master.zip"

******!cp "/content/output/results.txt" "/content/drive/My Drive/path/to/your/output/folder"******


%cd /content/drive/MyDrive/extracted_files/darknet-master
%pwd
!sudo apt install dos2unix
!find . -type f -print0 | xargs -0 dos2unix
!chmod +x /content/drive/MyDrive/extracted_files/darknet-master/build/darknet
!chmod +x /content/drive/MyDrive
!make
!./darknet detector train /content/drive/MyDrive/extracted_files/darknet-master/myData/myData.data /content/drive/MyDrive/extracted_files/darknet-master/yolov4.cfg /content/drive/MyDrive/extracted_files/darknet-master/yolov4.conv.137 -map -dont_show -backup /content/drive/MyDrive/extracted_files/darknet-master/backup
```

Şekil 19: Google Colab Veri Seti Eğitim Kodu

Google Colab'da bir veri seti eğitilirken oluşturulan chart.png dosyası, genellikle eğitim sürecinin görsel bir özetini sunar. Bu dosya, modelin performansını zamanla izlemek ve değerlendirmek için eğitim ve doğrulama metriklerini grafiksel olarak gösterir. Şekil 20'de eğitim sürecini özetleyen bir grafik verilmiştir. Kırmızı renkli eğri Ortalama Hassasiyet (mean Average Precision, mAP) eğrisidir. Modelin doğruluğunu ölçen bir metriktir ve yüksek bir mAP değeri, modelin iyi performans gösterdiğini belirtir.

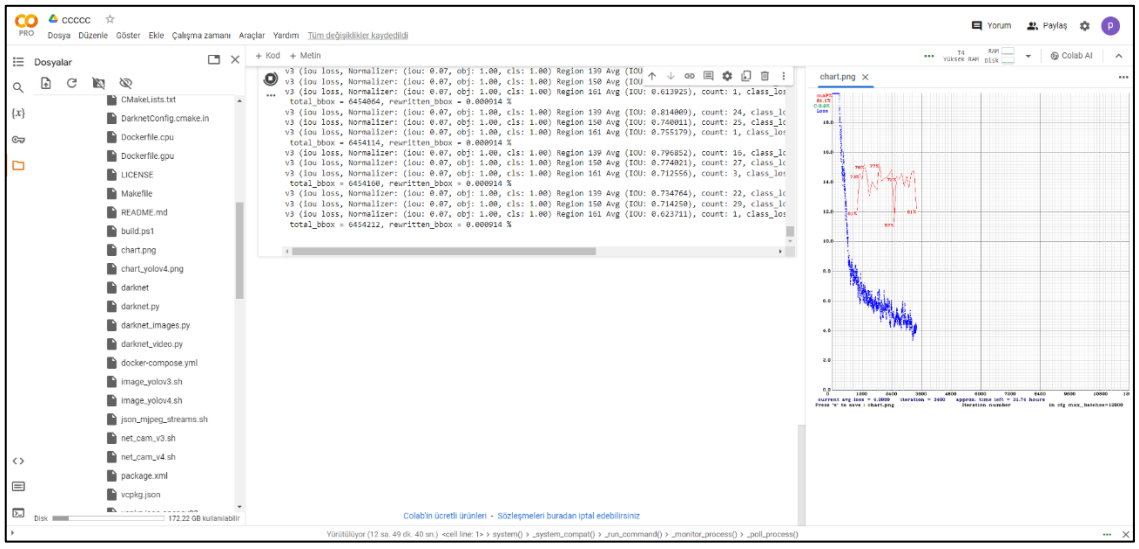
Modelin doğruluğunun eğitim süresi ile doğru orantılı olarak artması beklenirken, burada mAP değerinin zamanla dalgalandığı görülmektedir. Bunun sebebi, veri setindeki görüntü sayısının kısıtlı olmasıdır. Bu sorunun, veri artırma işlemi sonrasında aşırı öğrenme engellenilerek çözümlenebileceği beklenmektedir. Mavi renkli kayıp değişim eğrisi (Loss eğrisi) ise modelin ne kadar yanlış tahmin yaptığını gösterir ve bu değerin düşük olması beklenir. Loss eğrisinin zamanla azaldığı ve 0' a yaklaştığı görülmektedir. Her

iterasyon, modelin ağırlıklarının bir kez güncellendiği anlamına gelir. Örneğin, en sonuncu 5500. iterasyonda modelin kayıp değeri 25 olarak kaydedilmiştir ve mAP değeri % 64.1 olarak hesaplanmıştır.

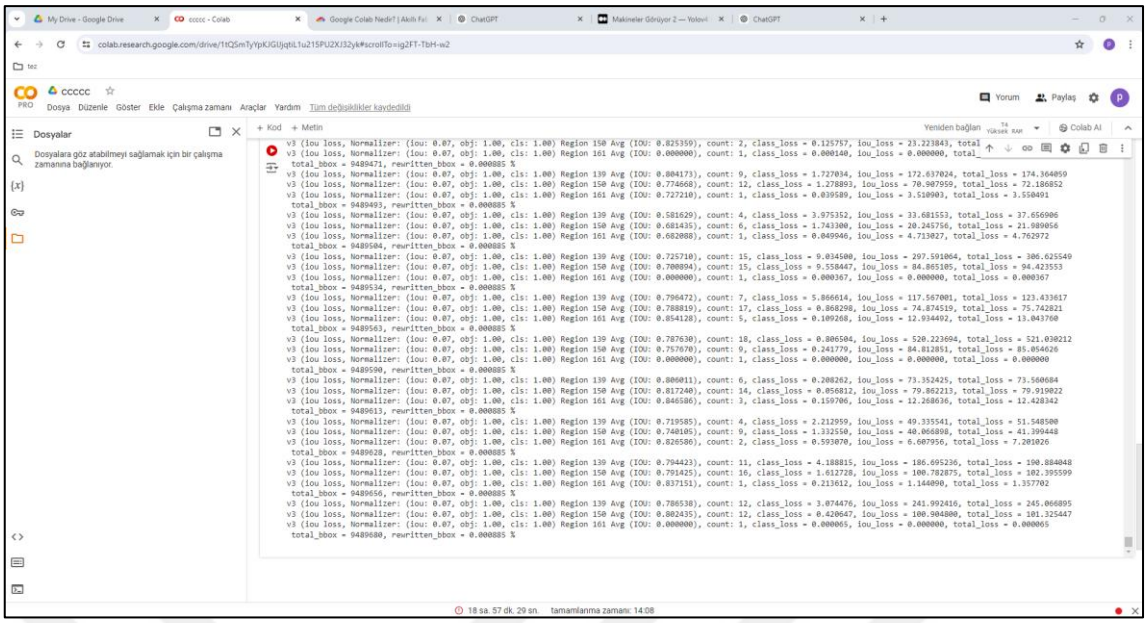


Şekil 20: Chart.png Dosyası

Eğitim devam ederken ve bittiği andaki görüntüler sırasıyla Şekil 21 ve Şekil 22'de gösterilmiştir.

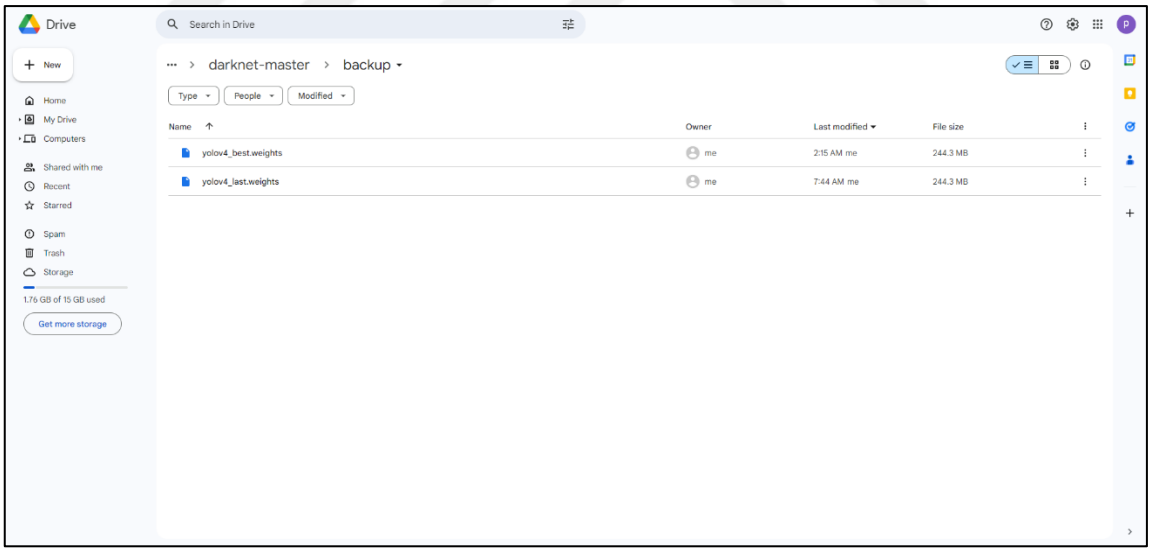


Şekil 21: PCB Veri Seti Eğitilirken İzlenen Ekran Görüntüsü



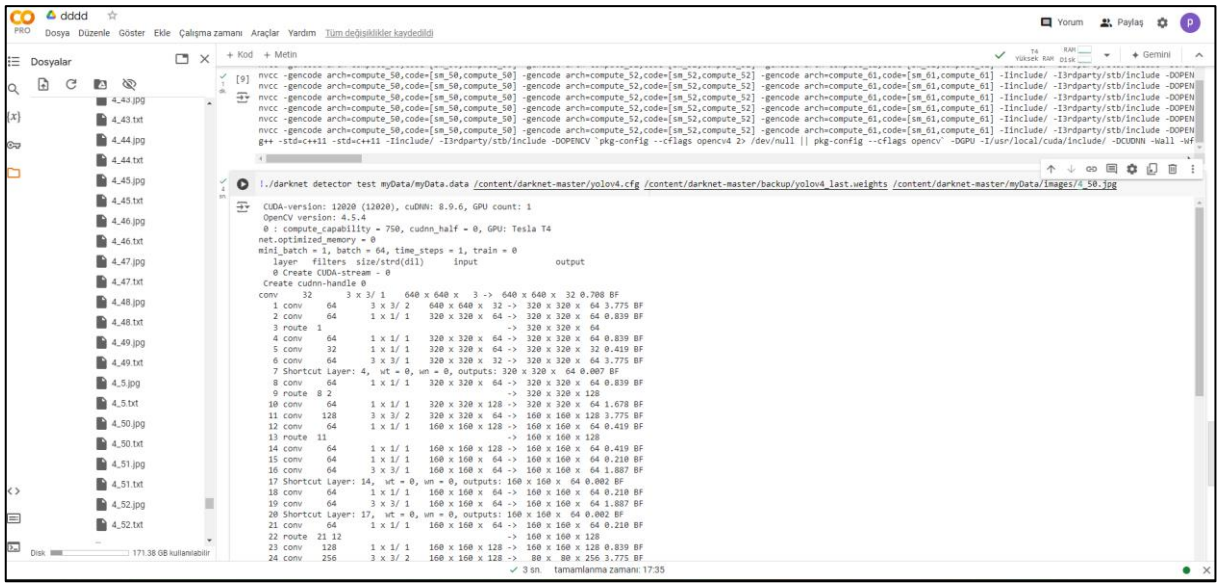
Şekil 22: PCB Veri Seti Eğitimi Tamamlandığında İzlenen Ekran Görüntüsü

YOLOv4 modelinin eğitimi tamamlandığında kullanıma hazır bir eğitimli ağırlık seti oluşmaktadır. Yolov4_best.weight ve Yolov4_last.weight dosyaları Şekil 23'de görselde mevcuttur.



Şekil 23: Eğitilmiş Veri Seti Ağırlık Dosyaları

Test kodu ise, Şekil 24'de verilmiştir.



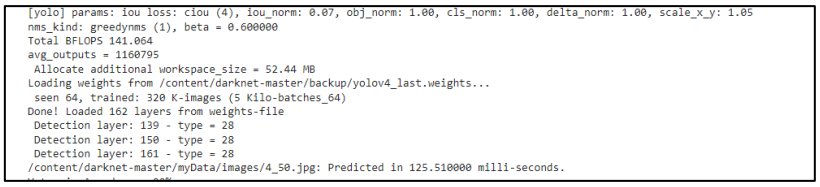
Şekil 24: YOLOv4 Model Test Komutu

Model, "Yetersiz Aşındırma", "Sahte Bakır Parçalar", "Delik Hatlar" ve "Açık Devre" gibi farklı kusur türlerini %94 - %99 aralığında yüksek doğruluk oranlarıyla tespit edebilmektedir. Şekil 25'de, Google Colab'a yüklenen test görselinde 1 adet yetersiz aşındırma, 1 adet sahte bakır parça, 2 adet delik hat ve 2 adet açık devre kusuru olmak üzere toplam 6 adet kusur olduğu görülmektedir. Görselin iki farklı yerinde bulunan iki farklı açık devre kusurunu doğru tespit etme oranları %99 ve %95 olmak üzere farklılaşabilmektedir. Çünkü açık devre kusurunun görseldeki konumu ve durumu farklılık göstererek modelin kusuru doğru bulma oranını değiştirmektedir.

"Predicted in 125,538000 milli-seconds" ifadesi, modelin ilgili görüntüdeki nesneleri yaklaşık 125 milisaniyede tespit ettiğini göstermektedir. Görüntü başına tahmin süresi oldukça düşüktür (125,538 ms), bu da modelin hızlı çalıştığını gösterir ve gerçek zamanlı uygulamalara uygun olduğuna işaret eder. Eğitim parametrelerinden iou loss (ciou): 4, iou_norm: 0,07, obj_norm: 1,00, cls_norm: 1,00, delta_norm: 1,00, scale_x_y: 1,05'dir. Toplam BFLOPS 141,064'dur. Bu değer, modelin hesaplama karmaşıklığını gösterir. Daha yüksek BFLOPS, daha karmaşık bir model olduğunu belirtir.

Şekil 25'de YOLOv4 modeline ait sonuçlar görülmektedir. YOLO gibi nesne algılama modelleri, her bir görüntüdeki tahmin edilen objeler için bounding box'lar (sınırlayıcı kutular) oluşturur. "Ortalama Çıktılar" burada modelin tahmin ettiği bounding box sayısının ortalamasıdır. 1160795 sayısı oldukça yüksek bir çıktı sayısıdır, bu da modelin ne kadar karmaşık bir işlem yaptığını gösterir.

Google Colab'da, Şekil 26'da verilen kodlar kullanılarak test görselinin PNG cinsinden sonucu, Şekil 27'deki gibi elde edilmiştir.



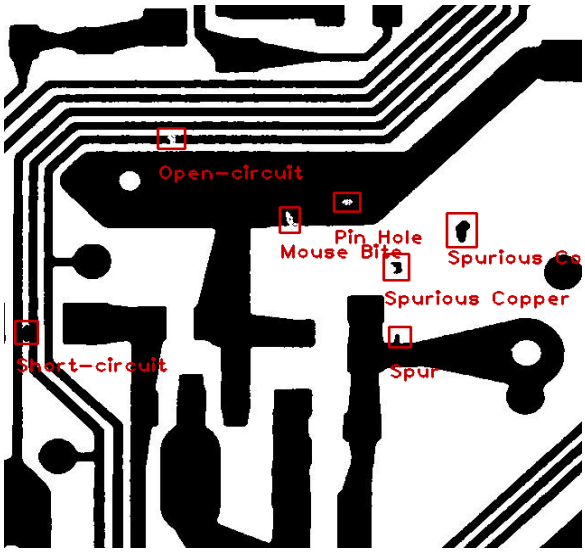
```
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 141.064
avg_outputs = 1160795
 Allocate additional workspace_size = 52.44 MB
Loading weights from /content/darknet-master/backup/yolov4_last.weights...
 seen 64, trained: 320 K-images (5 Kilo-batches_64)
Done! Loaded 162 layers from weights-file
 Detection layer: 139 - type = 28
 Detection layer: 150 - type = 28
 Detection layer: 161 - type = 28
/content/darknet-master/myData/images/4_50.jpg: Predicted in 125.510000 milli-seconds.
```

Şekil 25: Kusur Türü, Adedi ve Yüzde Doğruluk Oranı

```
******************
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# YOLOv4 modelinin yüklenmesi
net = cv2.dnn.readNet("/content/darknet-master/backup/yolov4_last.weights", "/content/darknet-master/yolov4.cfg")
classes = []
with open("/content/drive/MyDrive/myData.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]
layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]

# Resmin yüklenmesi ve boyutlarının alınması
img = cv2.imread("/content/darknet-master/myData/images/10_1.jpg")
height, width, channels = img.shape

# Resmi blob formatına dönüştürme
blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)

# YOLOv4 ile nesne tespiti yapma
net.setInput(blob)
outs = net.forward(output_layers)

# Tespit edilen nesnelerin koordinatlarını, etiketlerini ve doğruluklarını alma
class_ids = []
confidences = []
boxes = []
for out in outs:
    for detection in out:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.5:
            # Nesne merkez koordinatları ve genişlik-yükseklik değerlerinin hesaplanması
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)
            w = int(detection[2] * width)
            h = int(detection[3] * height)

            # Nesnenin sol üst koordinatlarının hesaplanması
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)

            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)
```

```
# Non-maximum suppression uygulanması
indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

# Tespit edilen nesnelerin çerçeve üzerine çizilmesi
font = cv2.FONT_HERSHEY_PLAIN
for i in range(len(boxes)):
    if i in indexes:
        x, y, w, h = boxes[i]
        label = str(classes[class_ids[i]])
        color = (0, 0, 200)


        cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
        cv2.putText(img, label, (x, y + 60), font, 1.5, color, 2)

# Çıktının gösterilmesi ve kaydedilmesi
cv2_imshow(img)
cv2.imwrite("output.jpg", img)
```

Şekil 26: Test Sonucunu Görselleştirmek İçin Kullanılan Kod



Şekil 27: Test Sonucu

Yüzey kusuru tespitinde yaygın olarak kullanılan farklı değerlendirme ölçütleri bulunmaktadır. Her piksel, zemin gerçeği ve tahmin sonuçlarının karşılaştırılmasına dayanarak dört kategori ile sınıflandırılmaktadır: 1) Gerçek pozitif (TP), gerçekte kusurlu olarak sınıflandırılmış kusur örneklerinin kusurlu olarak tahmin edilme sayısını belirtir; 2) Gerçek negatif (TN), gerçekte kusursuz örneklerin, kusursuz olarak tahmin edilme sayısını belirtir; 3) Yanlış pozitif (FP), gerçekte kusurlu örneklerin kusursuz örnekler olarak tahmin edilme sayısını belirtir; 4) Yanlış negatif (FN), gerçekte kusursuz örneklerin kusurlu olarak tahmin edilme sayısını belirtir [92]. Tablo VII'de değerleri gösterilmiştir.

Tablo VII. YOLOv4 Modelinin Sınıflara göre Performans Metrikleri

SINIF	**TP**	**FP**	**FN**
Kısa Devre	150	52	59
Açık Devre	218	76	74
Aşırı Aşınma	212	119	110
Yetersiz Aşınma	183	85	80
Sahte Bakır Parçalar	181	58	48
Delik Hatlar	168	30	31

Duyarlılık (Recall), modelin tespit etmesi gereken tüm nesnelerden kaç tanesini doğru bir şekilde tespit ettiğini gösterir. Hassasiyet (Precision), modelin tespit ettiği nesnelerden kaç tanesinin doğru olduğunu gösterir. Doğruluk (ACC), doğru sınıflandırılan örnek sayısının toplam örnek sayısına bölünmesiyle elde edilen değerdir.

Bu değerler, sırasıyla Denklem (5), (6), (7), (8) ve (9) ile hesaplanabilir. Hesaplanan

değerler Tablo VIII'e eklenmiştir.

$$FN = GT - TP \quad (5)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (6)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (7)$$

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (8)$$

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (9)$$

Tablo VIII. YOLOv4 Model Sonuçları

Sınıf	Sınıf No	Eğitim Örnek Sayısı	Test Örnek Sayısı	Doğruluk Oranı (%)	Ortalama Hassasiyet (%)	Duyarlılık (%)	F1 Skor (%)
Kısa Devre	0	898	209	57,4	62,8	71,7	66,9
Açık Devre	1	1259	292	59,2	65,8	74,6	69,9
Aşırı Aşınma	2	1262	322	48	51,1	65,8	57,5
Yetersiz Aşınma	3	1064	263	52,5	55,9	69,5	62
Sahte Bakır Parçalar	4	954	229	63	71,1	79	74,8
Delik Hatlar	5	948	199	73,3	78,0	84,4	81,1

Analiz sonuçlarına göre, "Delik Hatlar" sınıfı %73,3 doğruluk, %78,0 ortalama kesinlik, %84,4 duyarlılık ve %81,1 F1 puanı ile en başarılı tespit edilmiş kusur türüdür. Bu sonuçlar, modelin öne çıkan kusur özelliklerini etkili bir şekilde tespit edebildiğini göstermektedir. Öte yandan, %48 doğruluk, %51,1 ortalama hassasiyet, %65,8 geri çağırma oranı ve %57,5 F1 skoru ile "Aşırı Aşınma" sınıfı modelin zorlandığı bir alan olarak tespit edilmiş ve modelin bu sınıf için geliştirilmesi gerektiğini göstermiştir.

Çok sınıflı kusur tespiti için %64,1'lik mAP değeri (Tablo VIII'de verilen her sınıf için ortalama hassasiyetler), modelin aynı anda birden fazla kusur türünü tanıma becerisinde bir miktar başarı gösterdiğine işaret etmektedir. Bununla birlikte, bu değer modelin potansiyelini tam olarak yansıtmamakta ve performansında daha fazla iyileştirilmesi için hala yer olduğunu göstermektedir.

Benzer çalışmalarda elde edilen mAP değerleri genellikle, bu çalışmada elde edilen değerden daha yüksektir. Ancak bu çalışmada olduğu gibi çok sınıflı çoklu kusur tespitine yönelik çalışmalar devam etmektedir ve bu alan halen aktif bir araştırma konusudur. Bu tür karmaşık senaryolarda mAP değerlerinin daha düşük olması beklenmektedir. Çünkü modelin aynı anda

birden fazla sınıfı ve her sınıftaki birden fazla kusuru tespit etmesi gerekmektedir (Multi-Class, Multi-Instance Defect Detection). Bu durum mevcut modellerin sınırlarını zorlamakta ve algoritma geliştiricilerinin yeni iyileştirme yolları aramasını gerektirmektedir. Dolayısıyla bu çalışma, çok sınıflı çoklu kusur tespiti için derin öğrenme modellerinin geliştirilmesi ve iyileştirilmesi ihtiyacını vurgulamaktadır. Kusur sınıflandırması için %64,1, düşük bir mAP oranı gibi görünse de, literatürde çok etiketli, çok kusurlu durum için benzer durumlarda verilen orandan (<%50) daha yüksektir. [93] nolu çalışmada her kusur için farklı bir puan hesaplanmıştır. Ancak, ortalama mAP değeri hiçbiri için %50'yi aşmamıştır. Bu alanda yapılacak daha fazla araştırma, PCB üretim süreçlerindeki kusur tespit sistemlerinin genel performansını artıracak ve endüstriyel otomasyonu daha da ilerletecektir.

Model eğitiminin sonunda, eğitim sürecinin son döneminin ağırlıkları ve en iyi eğitim ağırlıkları elde edilmiştir. YOLOv4 için best.weights ve last.weights ağırlık dosyaları oluşturulmuştur.

Ancak YOLOv4 Tablo VIII'de de görüldüğü üzere yeterli performansı gösterememiştir. Bu yüzden ilk başta YOLOv4 modeli kullanılarak Vitis AI ile FPGA üzerinde uygulamanın çalıştırılması planlansa da daha yüksek performans elde edilmek istendiğinden YOLOv8 algoritması ile veri seti tekrar eğitilmiştir.

Yapay zeka ve makine öğrenimi modellerinin donanım üzerinde uygulanması, model ve platform arasındaki uyumluluk gereksinimleri nedeniyle çeşitli zorluklar içerebilmektedir. Bu çalışmada, ikinci aşamada Kria KV260 donanımında YOLOv8 algoritmasının kullanılması hedeflenmiştir. Ancak, KV260'ın desteklediği Vitis AI 3.0 sürümü, YOLOv8 için gerekli olan işlevselliği sağlamamaktadır. YOLOv8, Vitis AI'nin 3.5 sürümünde ve KV260 kartı tarafından desteklenmemektedir. Bu nedenle, çalışmada YOLOv8'in etkin bir şekilde kullanılabilmesi için alternatif bir yaklaşım olarak PYNQ platformunun kullanılması tercih edilmiştir.

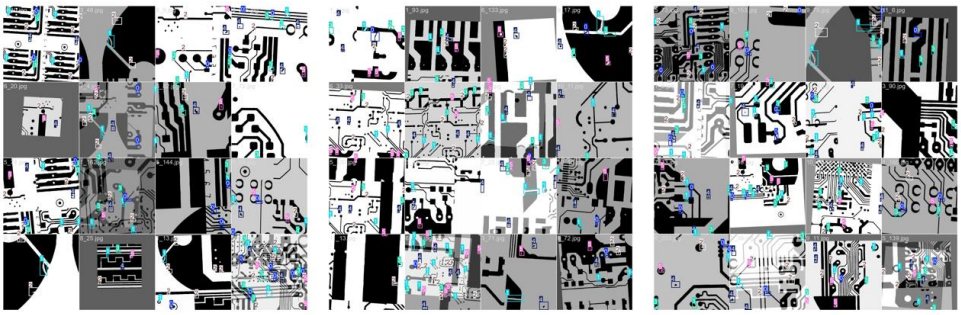
3.3.2. YOLOv8 Algoritması ile Model Eğitimi

YOLOv8m modeli aynı YOLOv4 gibi Google Colab üzerinde Ultralytics öğrenci lisansı ile eğitilmiştir. YOLOv8m eğitim sonucunda best.pt ve last.pt Pytorch ağırlık dosyaları oluşmuştur. En iyi eğitim ağırlıkları doğrulama için modelin çıktı ağırlıkları olarak kullanılmıştır. Şekil 28'de, eğitilen modelin eğitim sürecine ait görselleştirmesi sunulmaktadır. YOLOv8 derin öğrenme modeli için hiper parametreler Tablo IX'da verilmiştir. Ayrıca epoch değeri literatür taraması sonucu 100 olarak belirlenmiştir.

Tablo IX. YOLOv8 Derin Öğrenme Modeli Hiper Parametreleri

Hiper Parametre	**Değeri**
Görsel Boyutu (piksel)	640 x 640
Batch size	16
Epoch	100

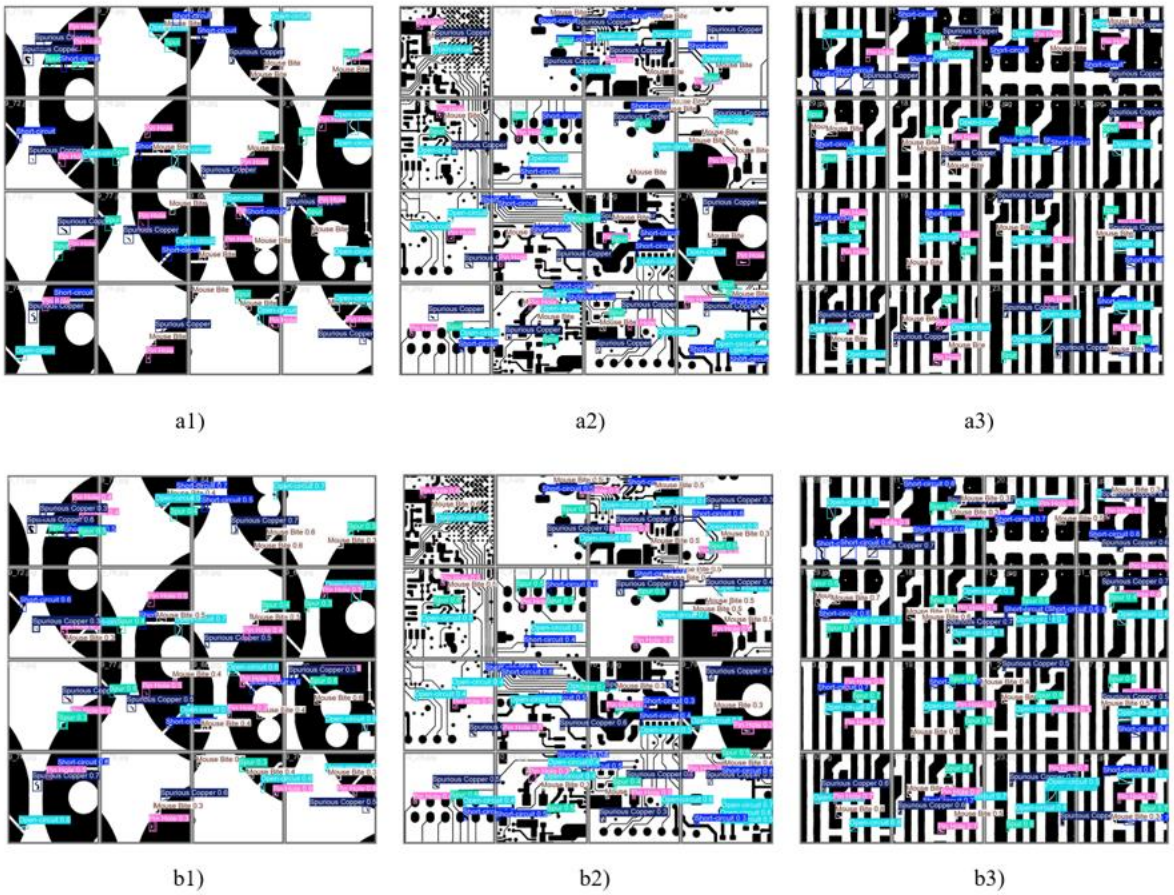
Batch size 16 olarak seçildiğinden her batch için modelin eğitimi sırasında kullandığı farklı görüntü sayısı 16 adettir. Eğitimde veya test sürecinde işlemin kaç batch ile tamamlanacağını bulmak için veri setindeki görüntü sayısı 16'ya bölünür.



a) train_batch0 b) train_batch1 c) train_batch2

Şekil 28: Eğitim Görselleştirmesi (Batch size=16 için)

Şekil 29.a (a1, a2, a3)'da rastgele seçilmiş 3 batch için doğrulama sürecine ait etiketlenmiş kusurlu bölgeler sunulmuştur. Şekil 29.b'de ise yine aynı batch'ler için modelin tahmin edip etiketlediği test görselleri yer almaktadır.



Şekil 29: Doğrulama Görselleştirmesi (Batch size=16 için) a) Etiketlenen Görseller (a1. validation _batch0, a2. validation _batch1, a3. validation _batch2) b) Tahmin Edilen Görseller (b1. validation_predict_batch0, b2. validation_predict_batch1, b3. validation_predict_batch2)

Şekil 30'da ise Şekil 29'da sunulan görsellerin içinden sadece 2 tanesi ele alınmış ve detaylı bir şekilde karşılaştırma yapılmıştır. Böylelikle modelin, nesneleri (kusurlu bölgeleri) doğru tespit edip edemediği kolaylıkla incelenebilmektedir. Şekil 30.a'da etiketlenen görseller (gerçek doğru, ground truth), Şekil 30.b'de ise modelin tahmin ettiği kusurlar ve kusurlu bölgeleri işaretleyen kutular yani "bounding box" lar ile tahmin edilen görsellerin IoU puanı (score) verilmiştir. IoU, tahmin edilen sınırlayıcı kutu ile etiketlenen sınırlayıcı kutu arasındaki örtüşmeyi ifade eden bir paremetredir ve 1 değeri %100'lük örtüşmeyi ifade etmektedir. Örneğin Şekil 30.b2'de Açık devre (Open-circuit) kusuru, nispeten yüksek bir değer olan 0,7'lik bir IoU puanı ile tespit edilmiştir. Diğer taraftan Şekil 30.b1'de Yetersiz Aşınma (Spur) kusuru, 0,4'lük bir IoU puanına sahiptir. Bu durumun sebebi kusurlu bölgenin yanlış tespit edilmesi değil, Şekil 30.a1'de görüldüğü gibi Yetersiz Aşınma (Spur) kusurunun etiketleme aşamasında gereğinden büyük bir kutu ile işaretlenmesidir.

Şekil 30.b2 de Kısa Devre (Short-circuit) kusurunun tespitinde ise üst üste binen 2 ayrı tahmin kutusu (bounding boxes) olduğu görülmektedir. Bunları ortadan kaldırmak için NMS (Non-

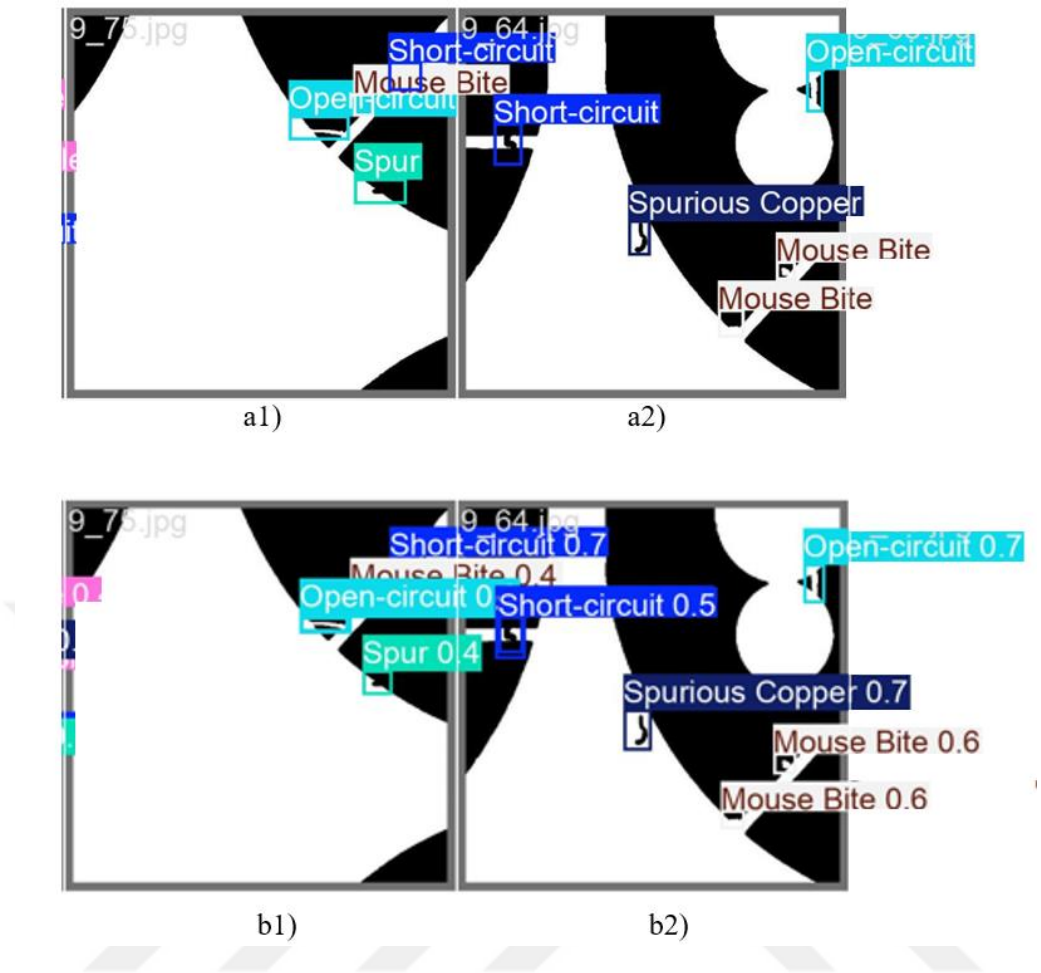
Maximum Suppression) tekniği kullanılmalıdır. NMS modelin çıkardığı tüm tahmin kutularını (bounding boxes) skorlarına göre sıralar (genellikle güven puanı ile) ardından aynı nesneye karşılık gelen kutular arasında, en yüksek skorlu olanı seçer ve diğerlerine baskı uygular, yani onları eler. Bu işlem, nesne algılamadaki gereksiz tekrarları azaltarak güven puanında (confidence score) iyileşmeler sağlayabilir.

Güven Puanı (confidence score), modelin tespit edilen her kusur sınıfı için kesinliğini ölçer. Bir kusurun belirli bir sınıfa ait olma olasılığı sınıf güveni olarak adlandırılır. Bu, denklem (10)'da verildiği şekilde hesaplanır:

$$C_i = \Pr(Class_i|Object) * \Pr(Object) * IoU(pred, truth) \quad (10)$$

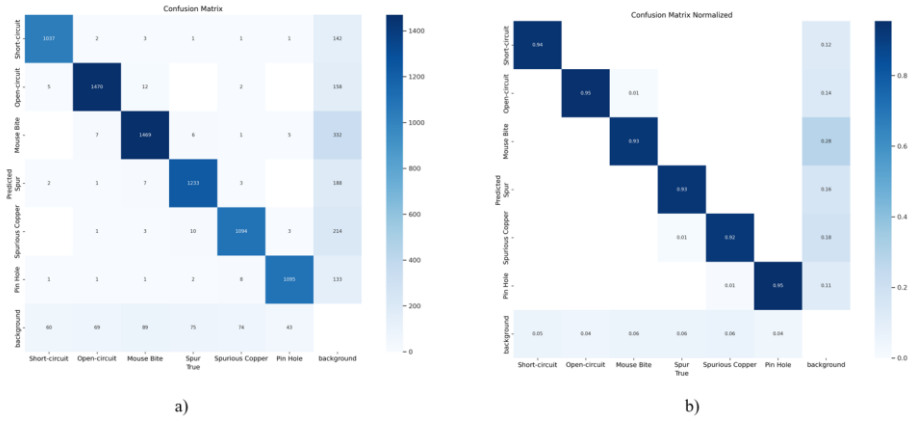
Burada C_i, Class_i'ye (6 kusur sınıfından birini temsil eder) ait bir güven puanıdır. Yani, bir kusurun sınıfının doğru olup olmadığının hesaplanmasıyla ilgili bir değerdir. Pr (Class_i|Object), Class_i sınıfının, belirlenen bir kusura ait olma olasılığını verir ve 1/6'ya eşittir. Pr (Object), Object'in (kusurun) genel olasılığıdır. Yani, bir görüntüde bir kusurun bulunma olasılığıdır.

Oluşturulan modelde her sınıfın ortalama güven skorları sırasıyla kısa devre 0,57; açık devre 0,56; fare ısırığı 0,49; yetersiz aşınma 0,51; sahte bakır parçalar 0,48; delik hat 0,54 olarak elde edilmiştir.



Şekil 30: Etiketlenen Kusurlu Bölgeler ile Tahmin Edilen Kusurlu Bölgelerin Detaylı Karşılaştırılması a) Etiketlenen Görseller b) Tahmin Edilen Görseller

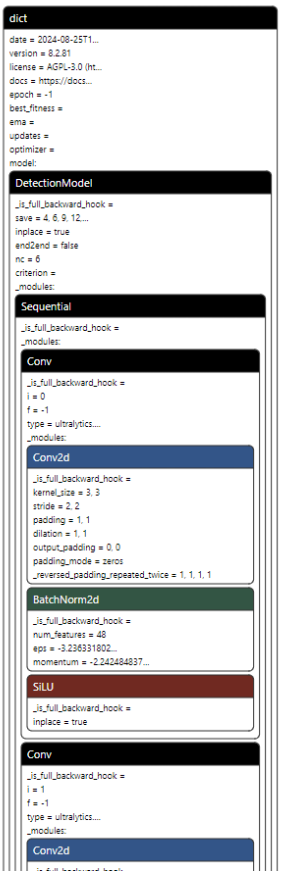
Modelin genel tahmin doğruluğu, tüm sınıf etiketleri arasındaki doğru tahmin sayısını bir matris biçiminde özetleyen bir karışıklık matrisi kullanılarak Şekil 31'de sunulmuştur. Burada, tahmin edilen kusur sınıfı (y ekseni) ile etiketlenen kusur sınıfı (x ekseni) arasındaki korelasyon sunulmaktadır. Köşegendeki koyu renkli hücrelerdeki elemanlar, doğru tahmin sayısını gösterirken; açık mavi renkli elemanlar ise, her sınıfa karşılık gelen yanlış tahminleri göstermektedir. Örneğin kısa devre sınıfı için 1045 örnekten 1037'si doğru tahmin edilmiş; geri kalan 8 tanesi ise yanlış sınıflandırılmıştır. Yanlış sınıflandırmalardan 5 tanesi açık devre, 2 tanesi yetersiz aşınma, 1 tanesi ise delik hattır.



a) b)

Şekil 31: Çok Sınıflı Çoklu Kusur Tespiti İçin Karışıklık Matrisi a) Karışıklık Matriksi b) Normalize Edilmiş Karışıklık Matriks

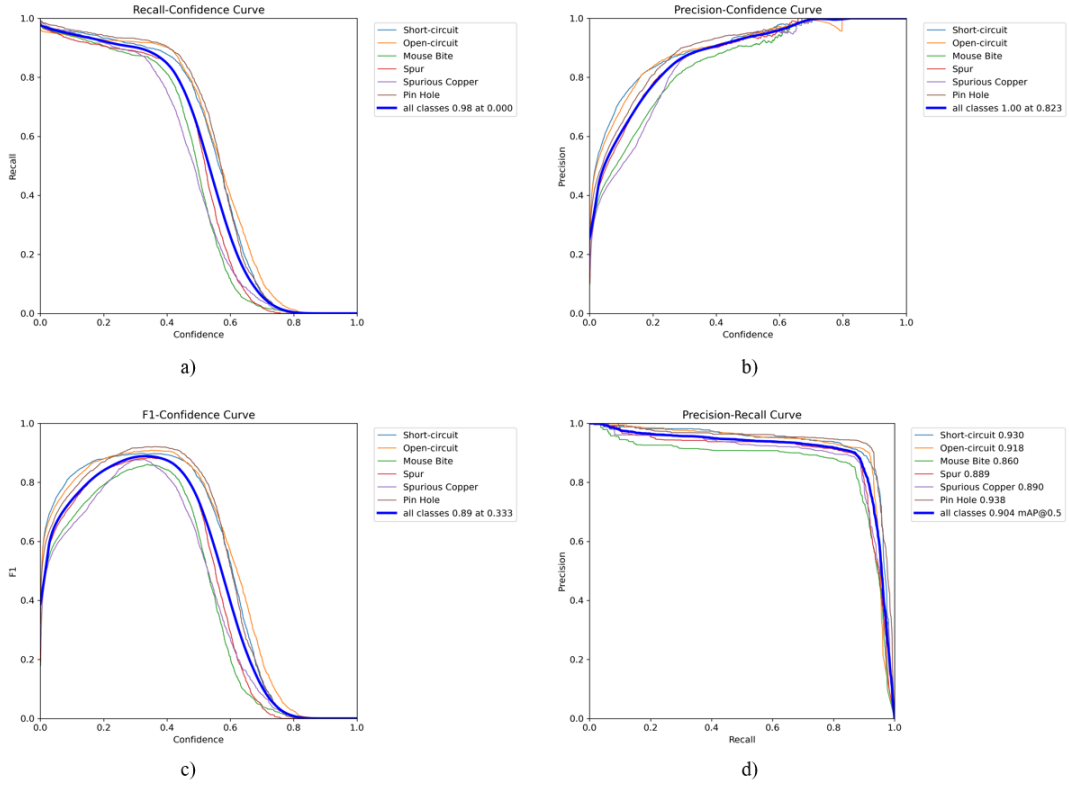
Netron yazılımı aracılığıyla YOLOv8 derin öğrenme modelinin mimarisi Şekil 32'deki gibi görselleştirilebilmektedir.



Şekil 32: YOLOV8m Model Mimarisinin Görselleştirilmesi

Şekil 33'de YOLOv8m modelinin 0-1 aralığındaki farklı güven (confidence) eşik değerlerinde gösterdiği performanslar verilmiştir.

Y ekseni kesinlik ve x ekseni duyarlılık oranı olmak üzere çizilen grafik, Kesinlik- Duyarlılık (P-R) eğrisi olarak adlandırılır ve eğitim sürecinin P-R eğrisi Şekil 33 .d'de gösterilmiştir. P-R eğrisi, farklı eşikler için kesinlik ve duyarlılık arasındaki dengeyi gösterir. Kesinlik ne kadar yüksekse, duyarlılık oranı da o kadar yüksek olur ve bu da model daha verimlidir anlamına gelir [23]. Yani, çizilen P-R eğrisi mümkün olduğunca sağ üst köşeye yakın olmalıdır. Şekil 33 .d'de her kusur sınıfı için ayrı ayrı P-R eğrileri çizilmiştir. Mavi renkle temsil edilen eğri tüm sınıfların P-R eğrisidir. Üst kısma yakın bir eğri, modelin hem yüksek Precision hem de yüksek Recall değerleri elde ettiğini gösterir, bu da iyi bir performansa işaret eder.

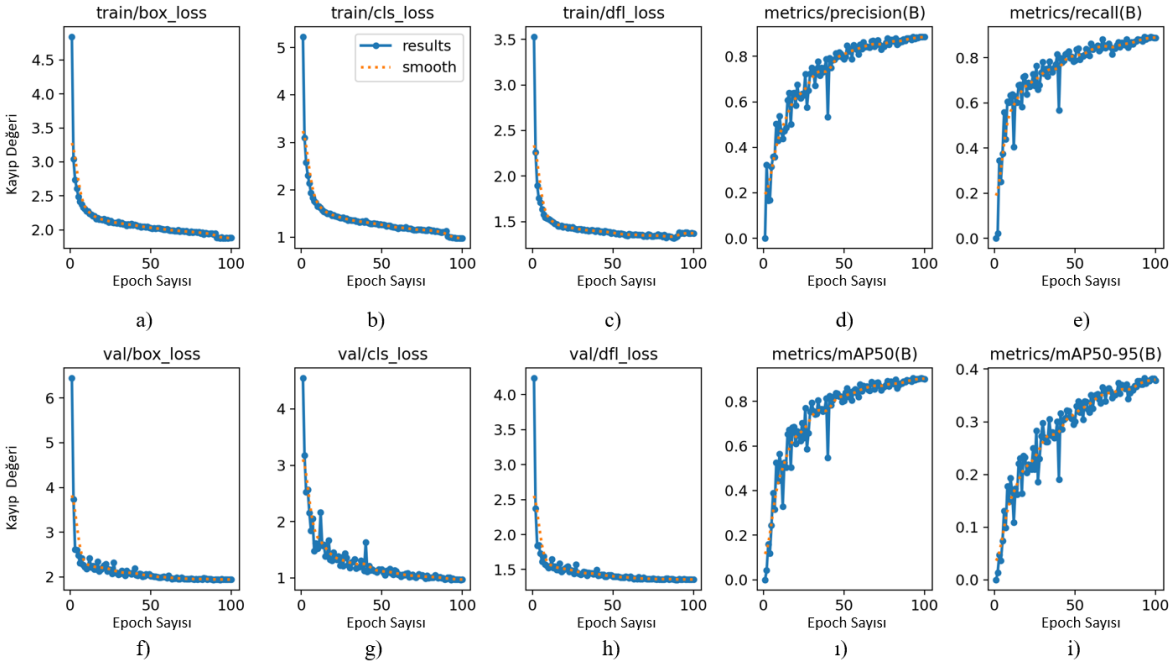


Şekil 33: YOLOv8 Performans Metrikleri a)R-C Eğrisi b)P-C Eğrisi c)F1-C Eğrisi d)P-R Eğrisi

Şekil 33.d'de görüldüğü gibi YOLOv8 modelinin genel mAP@50 değeri %90,4 olarak elde dilmiştir. Bu değer, YOLOv4'de elde edilen %64,1 değerinden daha yüksektir.

Şekil 34'de sunulan her bir eğri farklı performans ve kusur metriklerini temsil eder. Train/box_loss eğrisi, eğitim sırasındaki kutu kaybı olarak ifade edilir. Kutu kaybı nesne tespiti sırasında tahmin edilen kutuların gerçek kutularla olan farkını ölçen bir kayıp fonksiyonudur. Bu kayıp ne kadar düşükse modelin tahminleri o kadar doğru demektir. Train/cls_loss eğrisi eğitim sırasında sınıf kaybını incelemek için kullanılırken train/dfl_loss eğrisi sınırlayıcı kutuların (bounding boxes) konumlarını daha hassas şekilde optimize etmek için incelenen bir

kayıptır. Sınıf kaybı ise modelin tahmin ettiği sınıfın gerçek sınıf ile ne kadar iyi örtüştüğünü ölçen bir kayıp fonksiyonudur. Metrics/precision eğrisi, kesinlik (precision), metrics/recall eğrisi duyarlılık (recall) oranını ifade eder. Val/box_loss eğrisi doğrulama (validation) verisi üzerindeki kutu kaybını, val/cls_loss eğrisi ise doğrulama (validation) verisi üzerindeki sınıf kaybını ifade eder. metrics/map50 eğrisi mAP@50, ortalama doğruluk (precision) değeridir. mAP@50, modelin IoU 0.50'nin üzerinde olan tahminlerinin doğruluğunu ölçer. Bu eğri modelin genel başarı performansını gösterirken, metrics/map50-95 eğrisi mAP@50-95, IoU eşik değerlerinin 0.50'den 0.95'e kadar değiştiği bir aralıkta hesaplanan ortalama doğruluk değeridir. Bu daha genel bir performans ölçümüdür ve modelin farklı hassaslık seviyelerindeki doğruluğunu gösterir.



Şekil 34: YOLOv8 Modelinin Sonuçları a)train/box_loss Eğrisi b) train/cls_loss Eğrisi c) train/dfl_loss Eğrisi d)metrics/precision Eğrisi e)metrics/recall Eğrisi f)val/box_loss Eğrisi g) val/cls_loss Eğrisi h) metrics /map50 h) metrics /map50-95

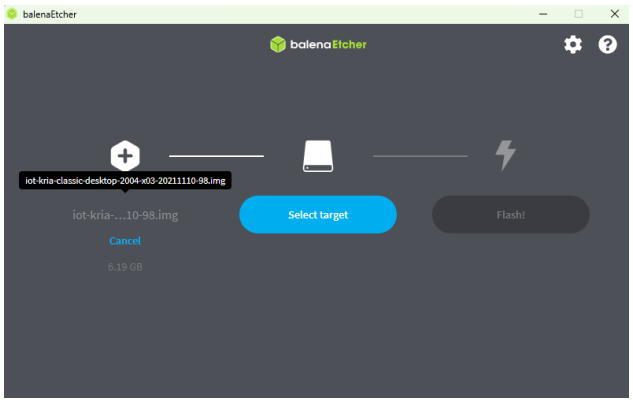
Bu model, FPGA tabanlı Kria KV260 platformunda test edilmek üzere hazırlanmıştır. Kria KV260, yüksek performanslı ve düşük güç tüketimli bir FPGA platformudur ve derin öğrenme modellerini hızlı bir şekilde çalıştırmak için idealdir [18].

4. DERİN ÖĞRENME MODELİNİN FPGA ÜZERİNDE UYGULANMASI

SoC (System-on-Chip) mevcut olan bir FPGA kartında, hem Processing System (PS) hem de PL (Programmable Logic) alanı bulunmaktadır ve uygulamalar her iki ortamdan faydalanılarak geliştirilebilir. Bu tez çalışmasında, YOLOv8'in Vitis AI 3.0' desteği olmamasından dolayı FPGA'nın PS kısmı üzerinde uygulama geliştirilmesi tercih edilmiştir.

PL kısmında tasarımlar daha zahmetli olsa da FPGA'nın sunduğu gerçek avantajlar ancak sadece burada kullanılabilmektedir. Mevcut durumda kullanılan geliştirme kartı bir bilgisayar olarak çalışmaktadır ve yük tamamen işlemeciye (4 adet Arm®Cortex®-A53 çekirdek) kaldığından tahmin süresi daha uzun olmaktadır.

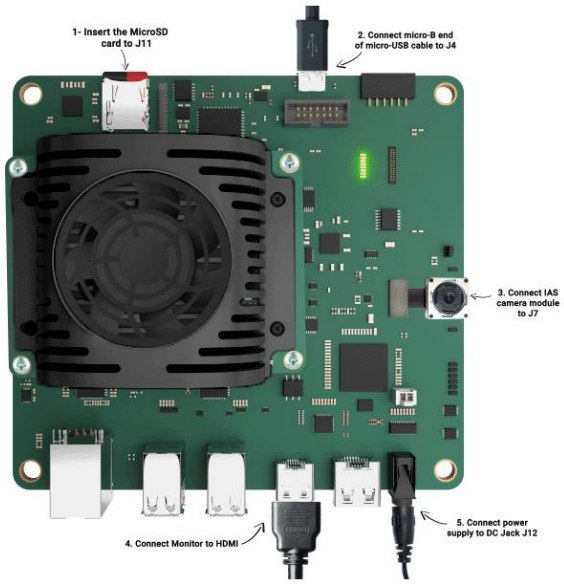
Xilinx MPSoC genellikle Petalinux kullanılarak geliştirilirken, Python ile ilgili kütüphanelerin genişletilebilirliği ve PYNQ'un olanakları nedeniyle bu tez çalışmasında Ubuntu 20.04 tercih edilmiştir. Tablo X'da verilen versiyonlarla uyumlu Ubuntu .img dosyası [83] önce bilgisayara indirilmiş, ardından balenaEtcher [84] aracılığıyla SD karta, Şekil 35'deki yöntemle yüklenmiştir. İmaj dosyasını SD karta yüklemek için bir alternatif olarak Win32DiskImager [85] de kullanılabilir.



Şekil 35: balenaEtcher ile SD Karta İmaj Dosyası Yükleme

Tüm arayüz bağlantıları kurulmuş ve test edilmiştir. Daha ayrıntılı bağlantı ve test prosedürleri için [86] numaralı kaynağa başvurulabilir. Kria KV260 kartının kullanım kolaylığını artırmak amacıyla, USB portları aracılığıyla klavye ve fare; HDMI portu kullanılarak ise ekran bağlantısı gerçekleştirilmiştir. Kria KV260'a uzaktan SSH ile erişim sağlamak için PuTTy [87] terminal programı konfigüre edilmiştir. Tüm arayüz bağlantıları tamamlandıktan sonra, Kria KV260

kartına SD kart yerleştirilmiş ve cihaza enerji verilmiştir. İnternet bağlantısının aktif olduğu, ping komutu ile doğrulanmıştır.



Şekil 36: Kria KV260 Arayüz Bassğlantıları [94]

Doğrulamalar tamamlandıktan sonra, eğitilen modelin uygulandığı Kria KV260 platformuna, Tablo X'da belirtilen versiyonlarla uyumlu yazılım paketleri terminal aracılığıyla yüklenmiştir. Ancak, versiyon uyumsuzlukları tespit edildiğinden, projenin gereksinimlerini karşılamak üzere özelleştirilmiş bir izole çalışma ortamı oluşturulmuştur. Bu izole ortamın kurulmasının ardından, ilgili yazılım paketleri aşağıda verilen komutlar kullanılarak izole ortama entegre edilmiş ve Jupyter Notebook ortamı başlatılmıştır.

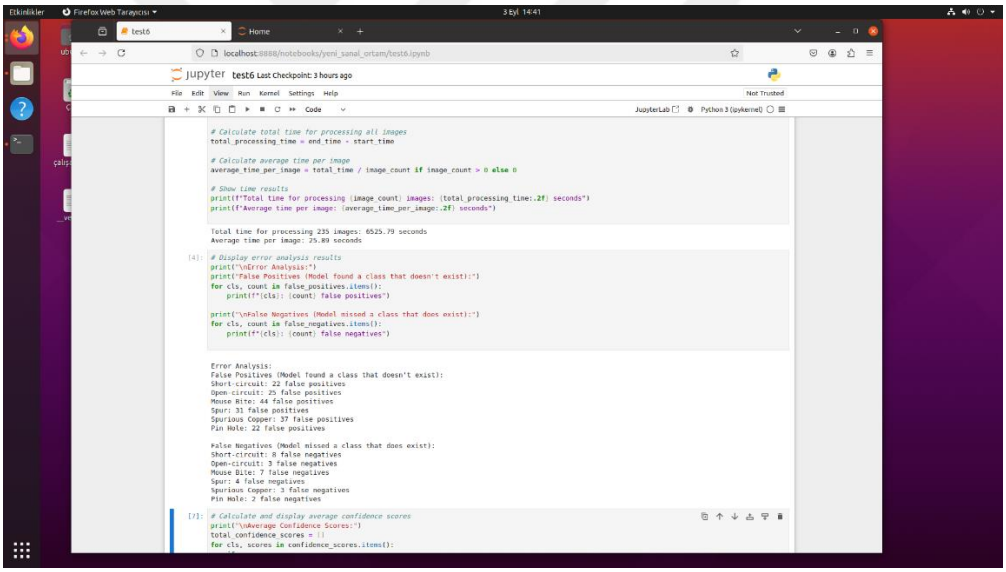
Kria KV260 Terminal Kodları

1: *ubuntu@kria:~$ python3 -m venv yeni_sanal_ortam*
2: *ubuntu@kria:~$ source yeni_sanal_ortam/bin/activate*
3: *(yeni_sanal_ortam) ubuntu@kria:~$ pip install --upgrade pip setuptools Wheel*
4: *(yeni_sanal_ortam) ubuntu@kria:~$ pip install torch --extra-index-url https://download.pytorch.org/whl/cpu*
5: *(yeni_sanal_ortam) ubuntu@kria:~$ pip install torch==1.10.0 torchvision==0.11.1 torchaudio==0.10.0*
6: *(yeni_sanal_ortam) ubuntu@kria:~$ pip install ultralytics*
7: *(yeni_sanal_ortam) ubuntu@kria:~$ pip install jupyter*
8: *(yeni_sanal_ortam) ubuntu@kria:~$ jupyter notebook*

Tablo X. FPGA Deneysel Ortam Yapılandırması

Çevresel Parametre	**Değer/Versiyon**
İşletim sistemi	Ubuntu 20.04
Derin Öğrenme Çerçevesi	PyTorch
Programlama Dili	Python 3.8.10
pip	24.2
torch	1.10.0
torchvision	0.11.1
Ultralytics	8.2.82
matplotlib	3.7.5
numpy	1.24.4

Jupyter Notebook ortamında Python dili kullanılarak derin öğrenme modelinin uygulanması için hazırlanan kodlar çalıştırılmış ve Şekil 37'de gösterildiği üzere model, hücre bazında adım adım başlatılmıştır.



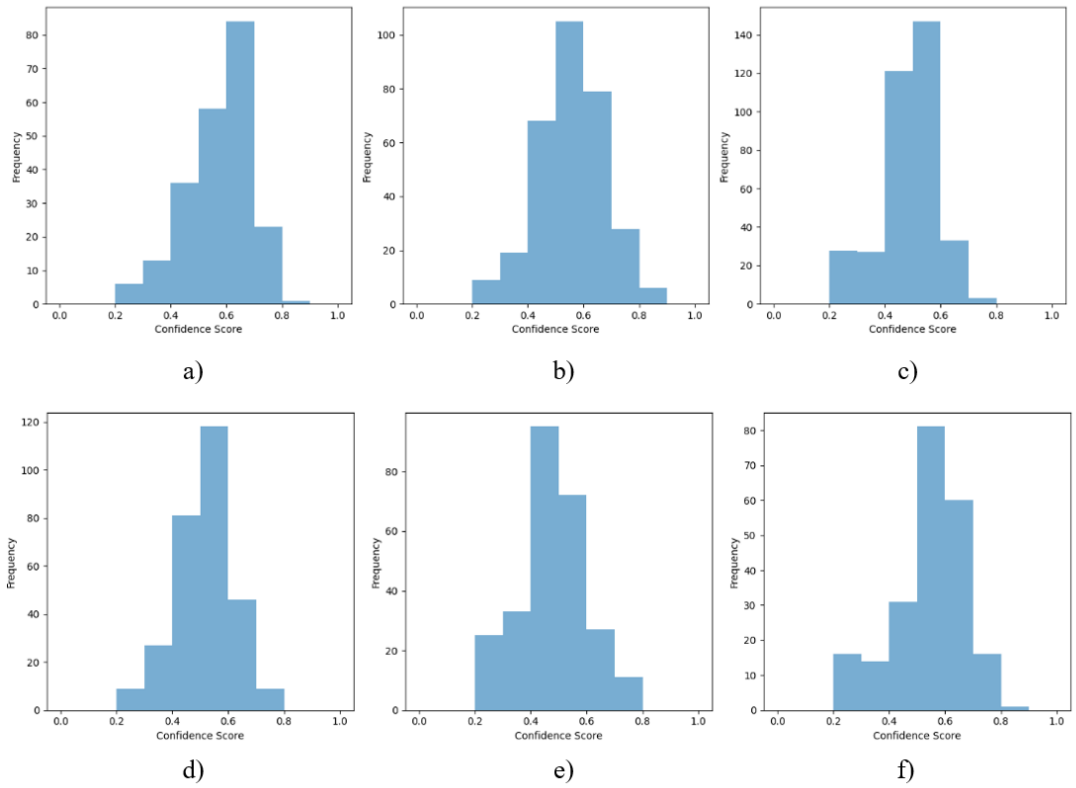
Şekil 37:Jupyter Notebook Kod Hücreleri

Google Colab ortamında eğitilen ve test edilen YOLOv8m modeli, Kria KV260 kartı üzerinde de kullanılmıştır. Karşılaştırma sonuçları Tablo XI'de sunulmuştur. Google Colab da eğitilen model ile bu modelin Kria KV260 kartı üzerinde çalıştırıldığı uygulama sonuçları aynı sonuçları vermiştir. Bu da, Kria KV260 kartı üzerinde modelin başarılı bir şekilde uygulandığını ve işlevselliğini doğrulamaktadır.

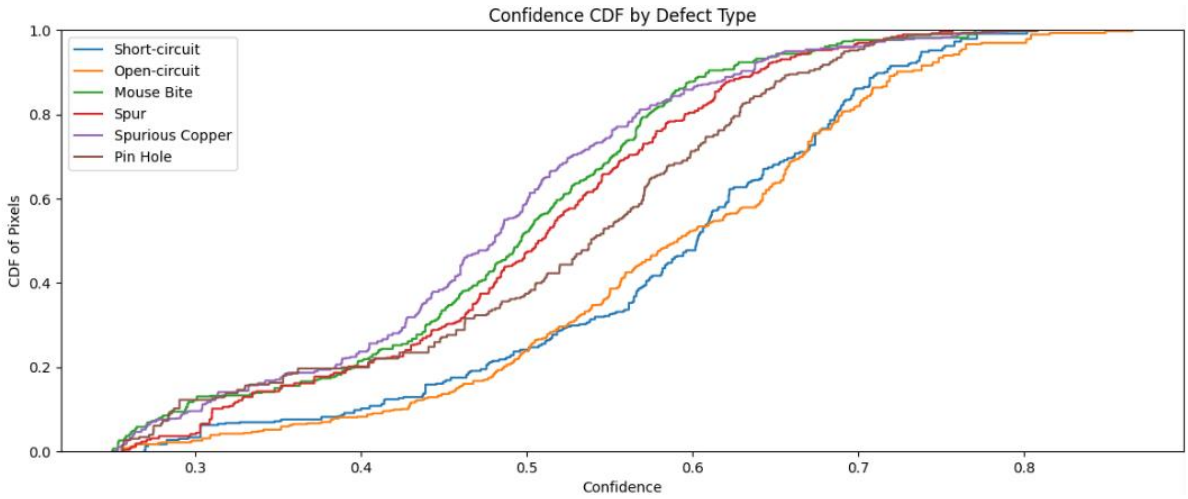
Tablo XI. Yolov8m Modelinin Platforma Göre Değerlendirilmesi

Sınıf	**Kria KV Üzerinde Doğruluk Oranı (%)**	**Google Colab Ortamında Doğruluk Oranı (%)**	**Kria KV Üzerinde Ortalama Güven Skoru**	**Google Colab Ortamında Ortalama Güven Skoru**
Kısa Devre	96.14	96.14	0.57	0.57
Açık Devre	98.97	98.97	0.56	0.56
Aşırı Aşınma	97.83	97.83	0.49	0.49
Yetersiz Aşınma	98.48	98.48	0.51	0.51
Sahte Bakır Parçalar	98.69	98.69	0.48	0.48
Delik Hatlar	98.99	98.99	0.54	0.54

YOLO çalışma şekli olarak, bir görüntüyü S × S hücre ızgarasına böler; burada her hücre, içindeki bir nesnenin varlığını tahmin etmekten sorumludur, ayrıca nesneleri kapsaması muhtemel bir dizi sınırlayıcı kutu üretir. Tahmin edilen sınırlayıcı kutuların her birinin bir güven skoru vardır [95]. YOLOv8 derin öğrenme modelindeki her sınıf için güven skoru-frekans histogram grafikleri Şekil 38'de sunulmuştur. Güven skoru en yüksek sınıf 0.57 ile kısa devre, en düşük sınıf 0.48 ile sahte bakır parçalar sınıfıdır. Şekil 39'da ise modelin bir görüntüdeki her bir piksel için yaptığı tahminlerin güven düzeyleri sunulmuştur.



Şekil 38: YOLOv8 Modelinde Sınıflara Ait Güven Skorları a) Kısa Devre b) Açık Devre c) Aşırı Aşınma d) Yetersiz Aşınma e) Sahte Bakır Parçalar f) Delik Hatlar

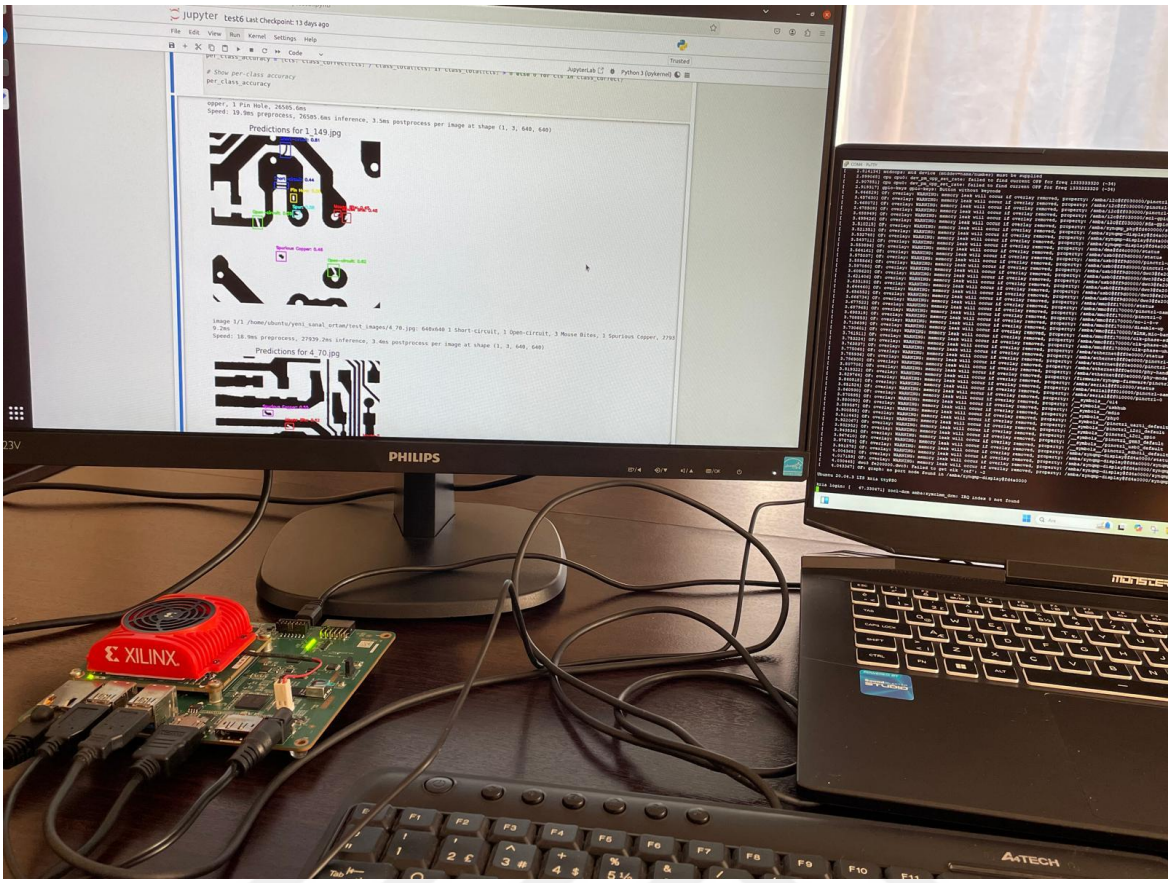


Şekil 39: Sınıfların Piksel Bazında Güven Skorları

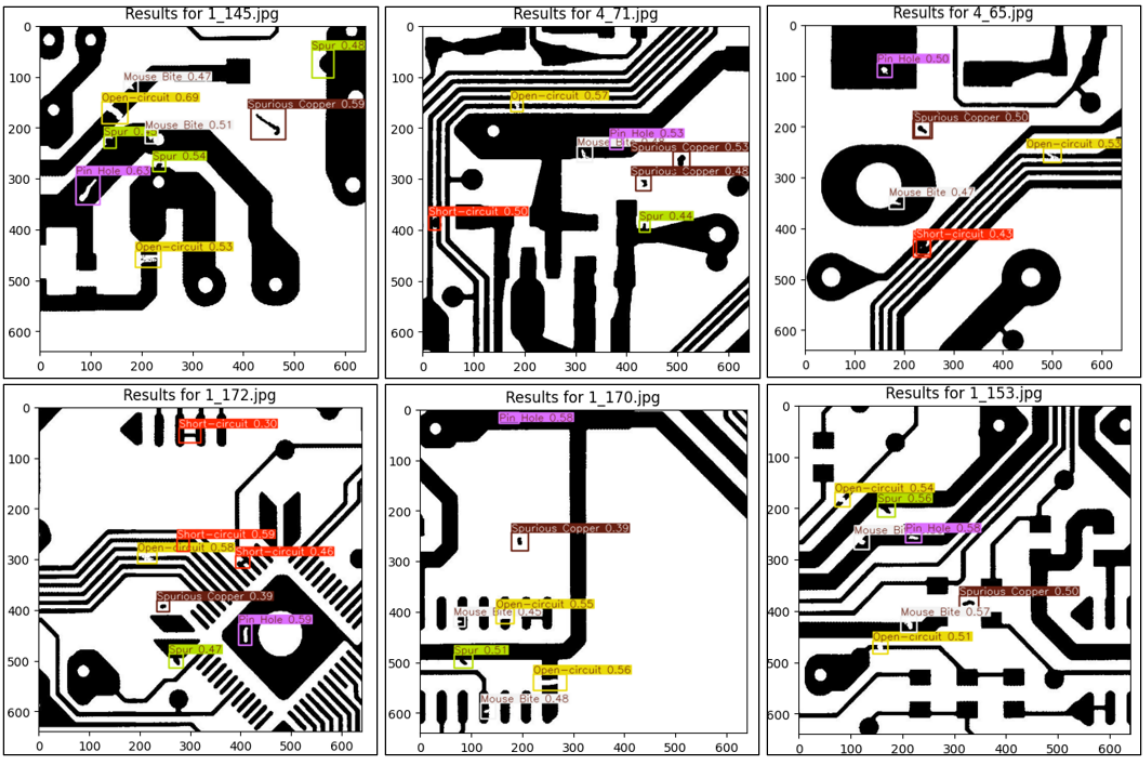
Tablo XII. YOLOv8 Modelinin Sınıflara Göre Performans Metrikleri

SINIF	**FP**	**FN**
Kısa Devre	22	8
Açık Devre	25	3
Aşırı Aşınma	44	7
Yetersiz Aşınma	31	4
Sahte Bakır Parçalar	37	3
Delik Hatlar	22	2

Şekil 40'da uygulamanın kurulum düzeneği, Şekil 41'de ise uygulama sonucunda elde edilen tahmin sonuçları yer almaktadır.



Şekil 40: Uygulama Kurulumu



Şekil 41: YOLOv8 Modelinin Kria KV260 kartı Üzerinde Çalıştırıldığında Tahmin Sonuçları

5. SONUÇLAR VE DEĞERLENDİRME

Pre, görüntü ölçekleme, dolgu ve kanal dönüşümü tarafından tüketilen zaman dahil olmak üzere modelin ön işleme süresidir. Inference (çıkarım), modelin çıkarım süresidir ve önceden işlenmiş görüntü girişinden model çıktı sonucuna kadar geçen süreyi ifade etmektedir. Post, modelin son işleme süresidir ve model çıktı sonuçlarının satır dönüşümünde harcanan süreyi ifade etmektedir [96]. Inference süresinin uzun sürmesi, modelin boyutu, karmaşıklığı ve donanımın yetersizliği ile ilgilidir. Bu ölçülen değerler gerçek zamanlı nesne(kusur) veya olayların tespiti için yeterli hızda olmamakla beraber, PCB kusurlarının geleneksel yöntemle tespit edilmesine karşılık avantajlar sunmaktadır. Yeterli hıza ulaşmadaki kısıtlamalar, modelin boyutu ve karmaşıklığı ile donanım kapasitesinin yetersizliği gibi faktörlere bağlıdır. Bu kısıtları azaltma önerileri gelecek çalışmalar bölümünde verilmiştir.

Gerçek zamanlı nesne(kusur) tespiti için yeterli hızda olmamakla beraber YOLOv8m modeli, PCB kusurlarını etkili bir şekilde tespit etme kapasitesine sahiptir. Modelin her sınıf için mAP değerleri %95'i aşmış, özellikle delik hatlar sınıfında maksimum %98,99 değerine ulaşmıştır. Bu sonuçlar, YOLOv8m modelinin, YOLOv4 modeline göre %26,3 daha yüksek mAP@50 ortalama hassasiyetine sahip olduğunu ve dolayısıyla daha iyi bir tespit performansı sunduğunu göstermektedir.

Sonuç olarak bu çalışma, Kria KV260 kartı üzerinde derin öğrenme modellerini çalıştırmak için yapılan denemelerin, modelin genel doğruluğunu %98,21 oranında başarılı bir şekilde koruduğunu ve görsel başına ortalama sürenin 29,4 saniye olduğunu göstermiştir. Tablo XIII'de modelin değerlendirme parametreleri sunulmuştur.

Modelin daha da iyileştirilmesi ve donanım üzerindeki performansının artırılması, gerçek zamanlı uygulamalar için önemli faydalar sunabilir. Kria KV260 kartında YOLOv8 algoritmasının uygulanmasında, FPGA'nın Processing System (PS) bölümünden faydalanılmış, Programmable Logic (PL) bölümündeki özelleştirilmiş donanım hızlandırma yetenekleri yeterince kullanılmamıştır. Bu durum, algoritmanın işlem süresinin gereksiz yere uzamasına ve FPGA'nın hızlandırma potansiyelinden tam anlamıyla faydalanılamamasına neden olmuştur. Bu sorunu çözmek için, modelin boyutunu ve karmaşıklığını azaltmaya yönelik optimizasyonlar önerilebilir. Örneğin, daha az hesaplama kaynağı gerektiren YOLO'nun Tiny gibi alternatif modellerin kullanılması işlem yükünü hafifletebilecektir. Ayrıca, modelin veri

tipi boyutunu küçülten niceleme (quantization) ve az kullanılan ağırlıkları kaldıran budama (pruning) teknikleri uygulanarak işlem süresi daha da kısaltılabilir.

Tablo XIII. Kria KV260 Kartı Üzerinde Modelin Değerlendirme Parametreleri

Model	Çerçeve	Backbone	Model Size (MB)	Size (pixels)	Pre-process (ms)	Inference (ms)	Post-process (ms)
YOLOv8m	PyTorch	YOLO-v8	50	640	29,2	29413,7	104,8

5.1. Gelecek Çalışmalar

Derin öğrenme modellerinin başarısının artırılması için donanımın iyileştirilmesi ve geniş veri setlerinin toplanması gerekmektedir. Model performansını ve genelleme yeteneklerini artırmak amacıyla veri artırma tekniklerinin kullanılması, bu konuda önemli araştırma alanları sunmaktadır. Ayrıca, çeşitlilik eksikliği ve veri kıtlığına yönelik mevcut zorluklar, veri artırma ve kusur simülasyonu yaklaşımlarının geliştirilmesi için potansiyel yeni bir çalışma alanı oluşturmaktadır. Gelecek çalışmalarda, aşağıda sıralanan konulara odaklanılması planlanmaktadır:

1. Veri setindeki görüntü sayısının artırılması, çeşitlendirilmesi ve bazı optimizasyon tekniklerinin uygulanması, mAP değerlendirme parametresinin iyileştirilmesi açısından büyük potansiyele sahiptir. Bu yaklaşımlar, modelin daha doğru tahminler yapmasını ve işlem sürelerinin hızlanmasını sağlayacak şekilde tasarlanabilir.
2. Gerçek zamanlı uygulamalar için zaman parametre değerlerini iyileştirecek ve daha hafif YOLO modelleri, örneğin Yolov8n ve YOLO tiny gibi modellerin geliştirilmesi gerekmektedir. Bu modellerin kullanılması, işlem hızını artırarak daha verimli sonuçlar alınmasını sağlayabilir.
3. PCB kusur tespit uygulamalarının başarısını artırmak için veri setinin her bir otomatik optik denetim (AOI) noktasına göre özelleştirilmesi mümkündür. Bu özelleştirme, farklı türdeki kusurları daha etkin bir şekilde tespit etmek için yapılabilir. Veri setinin özel ihtiyaçlara göre ayarlanması, modelin belirli kusurları daha doğru bir şekilde tanımasına ve genel olarak kusur tespit işleminin başarısını artırmasına yardımcı olabilir.

Son olarak, bu çalışma, derin öğrenme tabanlı PCB kusur tespitinin endüstriyel ortamda geniş bir uygulama potansiyeline sahip olduğunu göstermektedir.

Önerilen gelecek çalışmaların tamamlanması, kalite kontrol süreçlerinde ve üretim verimliliğinde önemli iyileştirmeler sağlayacaktır. Ayrıca, optimizasyon tekniklerinin entegrasyonu, kusur tespit süreçlerinin hızlandırılması için kritik bir rol oynayacaktır.

6. KAYNAKLAR

[1] Y. Zhou, M. Yuan, J. Zhang, G. Ding, and S. Qin, "Review of vision-based defect detection research and its perspectives for printed circuit board," *Journal of Manufacturing Systems,* vol. 70, pp. 557-578, 2023.

[2] https://chatgpt.com/g/g-fnPpt4WcB-image-creator (accessed.

[3] T. Czimmermann *et al.*, "Visual-based defect detection and classification approaches for industrial applications—a survey," *Sensors,* vol. 20, no. 5, p. 1459, 2020.

[4] X. Yue, Q. Wang, L. He, Y. X. Li, and D. Tang, "Research on Tiny Target Detection Technology of Fabric Defects Based on Improved YOLO," (in English), *Appl. Sci.-Basel,* vol. 12, no. 13, p. 16, Jul 2022, Art no. 6823, doi: 10.3390/app12136823.

[5] Y. B. Guo, X. J. Kang, J. F. Li, and Y. X. Yang, "Automatic Fabric Defect Detection Method Using AC-YOLOv5," (in English), *Electronics,* vol. 12, no. 13, p. 15, Jul 2023, Art no. 2950, doi: 10.3390/electronics12132950.

[6] Z. Y. Lu, Y. D. Zhang, H. Xu, and H. Chen, "Fabric defect detection via a spatial cloze strategy," (in English), *Text. Res. J.,* vol. 93, no. 7-8, pp. 1612-1627, Apr 2023, doi: 10.1177/00405175221135205.

[7] J. Wang, J. R. Yang, G. D. Lu, C. Zhang, Z. Y. Yu, and Y. Yang, "Adaptively Fused Attention Module for the Fabric Defect Detection," (in English), *Adv. Intell. Syst.,* vol. 5, no. 2, p. 10, Feb 2023, doi: 10.1002/aisy.202200151.

[8] Z. F. Liu, S. L. Liu, C. L. Li, and B. C. Li, "Fabric defect detection based on multi-source feature fusion," (in English), *Int. J. Cloth. Sci. Technol.,* vol. 34, no. 2, pp. 156-177, Mar 2022, doi: 10.1108/ijcst-07-2020-0108.

[9] J. F. Huang, T. J. Zhang, S. J. Zhao, L. Zhang, and Y. C. Zhou, "An Underwater Organism Image Dataset and a Lightweight Module Designed for Object Detection Networks," (in English), *ACM Trans. Multimed. Comput. Commun. Appl.,* vol. 20, no. 5, p. 23, May 2024, Art no. 147, doi: 10.1145/3640465.

[10] S. Qu, C. Cui, J. Duan, Y. Lu, and Z. Pang, "Underwater small target detection under YOLOv8-LA model," *Scientific Reports,* vol. 14, no. 1, p. 16108, 2024.

[11] Y. P. Wu, Y. Qin, Y. Qian, F. Guo, Z. P. Wang, and L. M. Jia, "Hybrid deep learning architecture for rail surface segmentation and surface defect detection," (in English), *Comput.-Aided Civil Infrastruct. Eng.,* vol. 37, no. 2, pp. 227-244, Feb 2022, doi: 10.1111/mice.12710.

[12] Y. P. Wu, P. Chen, Y. Qin, Y. Qian, F. Xu, and L. M. Jia, "Automatic Railroad Track Components Inspection Using Hybrid Deep Learning Framework," (in English), *IEEE Trans. Instrum. Meas.,* vol. 72, p. 15, 2023, Art no. 5011415, doi: 10.1109/tim.2023.3265636.

[13] C. C. Hsieh, T. Y. Hsu, and W. H. Huang, "An Online Rail Track Fastener Classification System Based on YOLO Models," (in English), *Sensors,* vol. 22, no. 24, p. 23, Dec 2022, Art no. 9970, doi: 10.3390/s22249970.

[14] A. Zhukov, A. Rivero, J. Benois-Pineau, A. Zemmari, and M. Mosbah, "A Hybrid System for Defect Detection on Rail Lines through the Fusion of Object and Context Information," (in English), *Sensors,* vol. 24, no. 4, p. 26, Feb 2024, Art no. 1171, doi: 10.3390/s24041171.

[15] S. C. Wang *et al.*, "Automated Identification and Localization of Rail Internal Defects Based on Object Detection Networks," (in English), *Appl. Sci.-Basel,* vol. 14, no. 2, p. 21, Jan 2024, Art no. 805, doi: 10.3390/app14020805.

[16] E. Eslami and H. B. Yun, "Attention-Based Multi-Scale Convolutional Neural Network (A plus MCNN) for Multi-Class Classification in Road Images," (in English), *Sensors,* vol. 21, no. 15, p. 27, Aug 2021, Art no. 5137, doi: 10.3390/s21155137.
[17] W. Zhou, Y. F. Zhan, H. C. Zhang, L. Zhao, and C. Wang, "Road defect detection from on-board cameras with scarce and cross-domain data," (in English), *Autom. Constr.,* vol. 144, p. 12, Dec 2022, Art no. 104628, doi: 10.1016/j.autcon.2022.104628.
[18] B. Cao, A. Jiang, J. Shen, and J. Liu, "Research on Rapid Recognition of Moving Small Targets by Robotic Arms Based on Attention Mechanisms," *Applied Sciences,* vol. 14, no. 10, p. 3975, 2024.
[19] J. Yao, J. M. Qi, J. Zhang, H. M. Shao, J. Yang, and X. Li, "A Real-Time Detection Algorithm for Kiwifruit Defects Based on YOLOv5," (in English), *Electronics,* vol. 10, no. 14, p. 13, Jul 2021, Art no. 1711, doi: 10.3390/electronics10141711.
[20] N. Ismail and O. A. Malik, "Real-time visual inspection system for grading fruits using computer vision and deep learning techniques," (in English), *Inform. Process. Agric.,* vol. 9, no. 1, pp. 24-37, Mar 2022, doi: 10.1016/j.inpa.2021.01.005.
[21] Y. H. Xie *et al.*, "ECLPOD: An Extremely Compressed Lightweight Model for Pear Object Detection in Smart Agriculture," (in English), *Agronomy-Basel,* vol. 13, no. 7, p. 30, Jul 2023, Art no. 1891, doi: 10.3390/agronomy13071891.
[22] Z. P. Wang, L. Y. Jin, S. Wang, and H. R. Xu, "Apple stem/calyx real-time recognition using YOLO-v5 algorithm for fruit automatic loading system," (in English), *Postharvest Biol. Technol.,* vol. 185, p. 11, Mar 2022, Art no. 111808, doi: 10.1016/j.postharvbio.2021.111808.
[23] Y. Wang, H. Y. Wang, and Z. H. Xin, "Efficient Detection Model of Steel Strip Surface Defects Based on YOLO-V7," (in English), *IEEE Access,* vol. 10, pp. 133936-133944, 2022, doi: 10.1109/access.2022.3230894.
[24] I. E. Parlak and E. Emel, "Deep learning-based detection of aluminum casting defects and their types," (in English), *Engineering Applications of Artificial Intelligence,* vol. 118, p. 18, Feb 2023, Art no. 105636, doi: 10.1016/j.engappai.2022.105636.
[25] M. Sharma, J. Lim, and H. Lee, "The Amalgamation of the Object Detection and Semantic Segmentation for Steel Surface Defect Detection," (in English), *Appl. Sci.-Basel,* vol. 12, no. 12, p. 18, Jun 2022, Art no. 6004, doi: 10.3390/app12126004.
[26] X. F. Zhu, J. Y. Liu, X. Y. Zhou, S. H. Qian, and J. H. Yu, "Detection of irregular small defects on metal base surface of infrared laser diode based on deep learning," (in English), *Multimed. Tools Appl.,* vol. 83, no. 7, pp. 19181-19197, Feb 2024, doi: 10.1007/s11042-023-16352-3.
[27] W. H. Li *et al.*, "Deep learning based online metallic surface defect detection method for wire and arc additive manufacturing," (in English), *Robot. Comput.-Integr. Manuf.,* vol. 80, p. 12, Apr 2023, Art no. 102470, doi: 10.1016/j.rcim.2022.102470.
[28] D. M. Yang, Y. R. Cui, Z. Y. Yu, and H. Q. Yuan, "Deep Learning Based Steel Pipe Weld Defect Detection," (in English), *Appl. Artif. Intell.,* vol. 35, no. 15, pp. 1237-1249, Dec 2021, doi: 10.1080/08839514.2021.1975391.
[29] Y. Tan, R. Y. Cai, J. R. Li, P. L. Chen, and M. Z. Wang, "Automatic detection of sewer defects based on improved you only look once algorithm," (in English),

Autom. Constr., vol. 131, p. 17, Nov 2021, Art no. 103912, doi: 10.1016/j.autcon.2021.103912.
[30] L. M. Dang, H. Wang, Y. Li, T. N. Nguyen, and H. Moon, "DefectTR: End-to-end defect detection for sewage networks using a transformer," (in English), *Constr. Build. Mater.,* vol. 325, p. 12, Mar 2022, Art no. 126584, doi: 10.1016/j.conbuildmat.2022.126584.
[31] D. W. Li, Q. Xie, Z. H. Yu, Q. Y. Wu, J. Zhou, and J. Wang, "Sewer pipe defect detection via deep learning with local and global feature fusion," (in English), *Autom. Constr.,* vol. 129, p. 13, Sep 2021, Art no. 103823, doi: 10.1016/j.autcon.2021.103823.
[32] X. Fang *et al.*, "Sewer defect instance segmentation, localization, and 3D reconstruction for sewer floating capsule robots," (in English), *Autom. Constr.,* vol. 142, p. 19, Oct 2022, Art no. 104494, doi: 10.1016/j.autcon.2022.104494.
[33] Z. Situ, S. Teng, X. T. Liao, G. F. Chen, and Q. Q. Zhou, "Real-time sewer defect detection based on YOLO network, transfer learning, and channel pruning algorithm," (in English), *J. Civ. Struct. Health Monit.,* vol. 14, no. 1, pp. 41-57, Jan 2024, doi: 10.1007/s13349-023-00681-w.
[34] Z. Situ *et al.*, "A transfer learning-based YOLO network for sewer defect detection in comparison to classic object detection methods," (in English), *Dev. Built Environ.,* vol. 15, p. 13, Oct 2023, Art no. 100191, doi: 10.1016/j.dibe.2023.100191.
[35] Q. Q. Zhou, Z. X. Situ, S. Teng, W. F. Chen, G. F. Chen, and J. H. Su, "Comparison of classic object-detection techniques for automated sewer defect detection," (in English), *J. Hydroinform.,* vol. 24, no. 2, pp. 406-419, Mar 2022, doi: 10.2166/hydro.2022.132.
[36] S. Y. Liang *et al.*, "Edge YOLO: Real-Time Intelligent Object Detection System Based on Edge-Cloud Cooperation in Autonomous Vehicles," (in English), *IEEE Trans. Intell. Transp. Syst.,* vol. 23, no. 12, pp. 25345-25360, Dec 2022, doi: 10.1109/tits.2022.3158253.
[37] Y. K. He, B. T. Wu, J. H. Mao, W. W. Jiang, J. Z. Fu, and S. Y. Hu, "An effective MID-based visual defect detection method for specular car body surface," (in English), *Journal of Manufacturing Systems,* vol. 72, pp. 154-162, Feb 2024, doi: 10.1016/j.jmsy.2023.11.014.
[38] H. X. Wang, H. A. Chen, B. Wang, Y. Jin, G. Q. Li, and Y. Kan, "High-efficiency low-power microdefect detection in photovoltaic cells via a field programmable gate array-accelerated dual-flow network," (in English), *Appl. Energy,* vol. 318, p. 13, Jul 2022, Art no. 119203, doi: 10.1016/j.apenergy.2022.119203.
[39] T. K. Chi *et al.*, "An Edge Computing System with AMD Xilinx FPGA AI Customer Platform for Advanced Driver Assistance System," (in English), *Sensors,* vol. 24, no. 10, p. 21, May 2024, Art no. 3098, doi: 10.3390/s24103098.
[40] L. Andrade, T. Baumela, F. Pétrot, D. Briand, O. Bichler, and M. Coppola, "Efficient Deep Learning Approach for Fault Detection in the Semiconductor Industry," in *ARTIFICIAL INTELLIGENCE FOR DIGITISING INDUSTRY: Applications*, O. Vermesan, R. John, C. DeLuca, and M. Coppola Eds., (River Publishers Series in Communications. Aalborg: River Publishers, 2021, pp. 131-146.
[41] Q. Yu, A. Liu, X. X. Yang, and W. M. Diao, "An Improved Lightweight Deep Learning Model and Implementation for Track Fastener Defect Detection with

Unmanned Aerial Vehicles," (in English), *Electronics,* vol. 13, no. 9, p. 15, May 2024, Art no. 1781, doi: 10.3390/electronics13091781.
[42] J. H. Zhang, D. M. Huang, T. T. Hu, R. Fuchikami, T. Ikenaga, and Ieee, "Critically Compressed Quantized Convolution Neural Network based High Frame Rate and Ultra-Low Delay Fruit External Defects Detection," in *17th International Conference on Machine Vision Applications (MVA)*, Nagoya, JAPAN, Jul 25-27 2021, NEW YORK: Ieee, 2021, doi: 10.23919/mva51890.2021.9511388. [Online]. Available: <Go to ISI>://WOS:000733621500044
[43] P. Kashyap *et al.*, "Unsupervised deep learning framework for temperature-compensated damage assessment using ultrasonic guided waves on edge device," (in English), *Scientific Reports,* vol. 14, no. 1, p. 14, Feb 2024, Art no. 3751, doi: 10.1038/s41598-024-54418-w.
[44] Y. Yuan, K. Virupakshappa, E. Oruklu, and Ieee, "Hardware Efficient Implementation of an Ultrasonic Non-Destructive Evaluation Algorithm based on Deep Learning," in *IEEE 65th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Fukuoka, JAPAN, Aug 07-10 2022, NEW YORK: Ieee, 2022, doi: 10.1109/mwscas54063.2022.9859349. [Online]. Available: <Go to ISI>://WOS:000861351300041
[45] A. Sezer and A. Altan, "Detection of solder paste defects with an optimization-based deep learning model using image processing techniques," (in English), *Solder. Surf. Mt. Technol.,* vol. 33, no. 5, pp. 291-298, Oct 2021, doi: 10.1108/ssmt-04-2021-0013.
[46] V. A. Adibhatla, H. C. Chih, C. C. Hsu, J. Cheng, M. F. Abbod, and J. S. Shieh, "Applying deep learning to defect detection in printed circuit boards via a newest model of you-only-look-once," (in English), *Math. Biosci. Eng.,* vol. 18, no. 4, pp. 4411-4428, 2021, doi: 10.3934/mbe.2021223.
[47] Q. Q. Lin, J. Z. Zhou, Q. R. Ma, Y. J. Ma, L. Kang, and J. J. Wang, "EMRA-Net: A Pixel-Wise Network Fusing Local and Global Features for Tiny and Low-Contrast Surface Defect Detection," (in English), *IEEE Trans. Instrum. Meas.,* vol. 71, p. 14, 2022, Art no. 2504314, doi: 10.1109/tim.2022.3151926.
[48] A. Bhattacharya and S. G. Cloutier, "End-to-end deep learning framework for printed circuit board manufacturing defect classification," (in English), *Scientific Reports,* vol. 12, no. 1, p. 13, Jul 2022, Art no. 12559, doi: 10.1038/s41598-022-16302-3.
[49] Y. Zhang, F. Xie, L. Huang, J. Shi, J. Yang, and Z. Li, "A lightweight one-stage defect detection network for small object based on dual attention mechanism and PAFPN," *Frontiers in Physics,* vol. 9, p. 708097, 2021.
[50] M. Jeon, S. Yoo, and S. W. Kim, "A Contactless PCBA Defect Detection Method: Convolutional Neural Networks With Thermographic Images," (in English), *IEEE Trans. Compon. Pack. Manuf. Technol.,* vol. 12, no. 3, pp. 489-501, Mar 2022, doi: 10.1109/tcpmt.2022.3147319.
[51] X. Wu, Y. X. Ge, Q. F. Zhang, and D. L. Zhang, "PCB Defect Detection Using Deep Learning Methods," in *24th IEEE International Conference on Computer Supported Cooperative Work in Design (IEEE CSCWD)*, Dalian, PEOPLES R CHINA, May 05-07 2021, NEW YORK: Ieee, in International Conference on Computer Supported Cooperative Work in Design, 2021, pp. 873-876, doi: 10.1109/cscwd49262.2021.9437846. [Online]. Available: <Go to ISI>://WOS:000716858200148

[52] Z. K. Zhang, M. L. Zhou, H. L. Wan, M. Li, G. Li, and D. L. Han, "IDD-Net: Industrial defect detection method based on Deep-Learning," (in English), *Engineering Applications of Artificial Intelligence,* vol. 123, p. 16, Aug 2023, Art no. 106390, doi: 10.1016/j.engappai.2023.106390.
[53] J. H. Liu, H. G. Li, F. Y. Zuo, Z. Zhao, and S. X. Lu, "KD-LightNet: A Lightweight Network Based on Knowledge Distillation for Industrial Defect Detection," (in English), *IEEE Trans. Instrum. Meas.,* vol. 72, p. 13, 2023, Art no. 3525713, doi: 10.1109/tim.2023.3300421.
[54] X. Chen, Y. L. Wu, X. Y. He, and W. Y. Ming, "A Comprehensive Review of Deep Learning-Based PCB Defect Detection," (in English), *IEEE Access,* Review vol. 11, pp. 139017-139038, 2023, doi: 10.1109/access.2023.3339561.
[55] X. Y. Hu, D. F. Kong, X. Y. Liu, J. W. Zhang, and D. D. Zhang, "FM-STDNet: High-Speed Detector for Fast-Moving Small Targets Based on Deep First-Order Network Architecture," (in English), *Electronics,* vol. 12, no. 8, p. 15, Apr 2023, Art no. 1829, doi: 10.3390/electronics12081829.
[56] Y. Z. Li, J. S. Lei, W. B. Shi, and S. Y. Yang, "HDTNet: PCB defect detection algorithm for tiny objects of high-density regions," (in English), *J. Intell. Fuzzy Syst.,* vol. 46, no. 1, pp. 1531-1541, 2024, doi: 10.3233/jifs-230150.
[57] P. Klco, D. Koniar, L. Hargas, K. P. Dimova, and M. Chnapko, "Quality inspection of specific electronic boards by deep neural networks," (in English), *Scientific Reports,* vol. 13, no. 1, p. 14, Nov 2023, Art no. 20657, doi: 10.1038/s41598-023-47958-0.
[58] S. Y. Luo *et al.*, "EC-YOLO: Improved YOLOv7 Model for PCB Electronic Component Detection," (in English), *Sensors,* vol. 24, no. 13, p. 22, Jul 2024, Art no. 4363, doi: 10.3390/s24134363.
[59] W. LI, "Detecting defects in PCB manufacturing: an exploration using Yolov8 deep learning," *International Journal on Interactive Design and Manufacturing (IJIDeM),* pp. 1-11, 2024.
[60] W. Xuan, G. Jian-She, H. Bo-Jie, W. Zong-Shan, D. Hong-Wei, and W. Jie, "A lightweight modified YOLOX network using coordinate attention mechanism for PCB surface defect detection," *IEEE sensors journal,* vol. 22, no. 21, pp. 20910-20920, 2022.
[61] G. Li and D. Zhu, "Research on road defect detection based on improved YOLOv8," in *2023 IEEE 11th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, 2023, vol. 11: IEEE, pp. 143-146.
[62] tangsanli5201. https://github.com/tangsanli5201/DeepPCB (accessed 10.10, 2023).
[63] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, "A survey of deep learning and its applications: a new paradigm to machine learning," *Archives of Computational Methods in Engineering,* vol. 27, pp. 1071-1092, 2020.
[64] R. Molina. https://indico.ictp.it/event/9644/session/7/contribution/29/material/slides/0.pdf (accessed 2024).
[65] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," *Journal of manufacturing systems,* vol. 48, pp. 144-156, 2018.
[66] V. K. Sharma and R. N. Mir, "A comprehensive and systematic look up into deep learning based object detection techniques: A review," *Computer Science Review,* vol. 38, p. 100301, 2020.

[67] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934,* 2020.
[68] J. Redmon, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
[69] M. Hussain, "YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection. Machines, 11 (7), 677," ed, 2023.
[70] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas," *Machine Learning and Knowledge Extraction,* vol. 5, no. 4, pp. 1680-1716, 2023.
[71] Z. H. Yuan, X. Y. Tang, H. Ning, and Z. Z. Yang, "LW-YOLO: Lightweight Deep Learning Model for Fast and Precise Defect Detection in Printed Circuit Boards," (in English), *Symmetry-Basel,* vol. 16, no. 4, p. 20, Apr 2024, Art no. 418, doi: 10.3390/sym16040418.
[72] "Object Detection." Ultralytics Inc. https://docs.ultralytics.com/tasks/detect/#what-formats-can-i-export-a-yolov8-model-to (accessed 28.08, 2024).
[73] Y. M. Xu, K. Zhang, and L. Wang, "Metal Surface Defect Detection Using Modified YOLO," (in English), *Algorithms,* vol. 14, no. 9, p. 14, Sep 2021, Art no. 257, doi: 10.3390/a14090257.
[74] A. Ushiroyama, M. Watanabe, N. Watanabe, A. Nagoya, and Ieee, "Convolutional neural network implementations using Vitis AI," in *IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, Electr Network, Jan 26-29 2022, NEW YORK: Ieee, 2022, pp. 365-371, doi: 10.1109/ccwc54503.2022.9720794. [Online]. Available: <Go to ISI>://WOS:000800243200057
[75] "AMD Xilinx Kria KV260 Vision AI Starter Kit : Using VITIS-AI." elementı4/AN AVNET COMMUNITY. https://community.element14.com/products/roadtest/b/blog/posts/amd-xilinx-kria-kv260-vision-ai-starter-kit-software (accessed 28.05, 2024).
[76] "Optimal Artificial Intelligence Inference from Edge to Cloud." AMD. https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html (accessed.
[77] "https://community.element14.com/products/roadtest/b/blog/posts/amd-xilinx-kria-kv260-vision-ai-starter-kit-pynq." https://community.element14.com/products/roadtest/b/blog/posts/amd-xilinx-kria-kv260-vision-ai-starter-kit-pynq (accessed 13.09, 2024).
[78] "https://github.com/Xilinx/Kria-PYNQ." https://github.com/Xilinx/Kria-PYNQ (accessed 13.09, 2024).
[79] "Kria™ KV260 Vision AI Starter Kit, PRODUCT BRIEF." https://www.amd.com/content/dam/amd/en/documents/products/som/kria/k26/kv260-product-brief.pdf (accessed 12.06, 2024).
[80] "Vision AI DPU-PYNQ." https://www.amd.com/en/developer/resources/kria-apps/vision-ai-dpu-pynq.html (accessed.
[81] S. Verma, A. Panigrahi, S. Gupta, and I. C. Soc, "QFabric: Multi-Task Change Detection Dataset," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Electr Network, Jun 19-25 2021, LOS ALAMITOS: Ieee Computer Soc, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2021, pp. 1052-1061, doi:

10.1109/cvprw53098.2021.00116. [Online]. Available: <Go to ISI>://WOS:000705890201015
[82] "labelbox." https://labelbox.com/ (accessed.
[83] "SuperAnnotate." https://superannotate.medium.com/introduction-to-object-detection-with-deep-learning-f4d2b9f4d1b5 (accessed.
[84] "Roboflow." https://roboflow.com/ (accessed.
[85] "Supervisely." https://supervisely.com/labeling-toolbox/images/ (accessed.
[86] "VGG Image." https://medium.com/@nikunjsaini37/get-ready-to-boost-your-yolov8-models-performance-by-creating-a-custom-dataset-with-the-vgg-image-292cac524db5 (accessed.
[87] "LabelImg." https://medium.com/@pat.x.guillen/setting-up-labelimg-1ad1887d91c5 (accessed.
[88] "Labelme." https://github.com/labelmeai/labelme (accessed.
[89] G. YILMAZ. "Google Colab Nedir? Nasıl Kullanılır?" MEDIUM, Akıllı Fabrikalar. https://akillifabrikalar.com.tr/google-colab-nedir-nasil-kullanilir/ (accessed 2024).
[90] O. Akköse. "Makineler Görüyor 2 — Yolov4 ile Kendi Nesne Tespit Modelinizi Oluşturun." MEDIUM. https://medium.com/deep-learning-turkiye/makineler-g%C3%B6r%C3%BCyor-2-yolov4-ile-kendi-nesne-tespit-modelinizi-olu%C5%9Fturun-c73b81c8a4ea (accessed 01.03, 2024).
[91] "YOLOv4_Training_Tutorial.ipynb." https://colab.research.google.com/drive/1_GdoqCJWXsChrOiY8sZMr_zbr_fH-0Fg?usp=sharing#scrollTo=imc0NP19hLuq (accessed 01.03, 2024).
[92] R. Ameri, C.-C. Hsu, and S. S. Band, "A systematic review of deep learning approaches for surface defect detection in industrial applications," *Engineering Applications of Artificial Intelligence,* vol. 130, p. 107717, 2024.
[93] Z. Chen, Q. Zhu, X. Zhou, J. Deng, and W. Song, "Experimental Study on YOLO-Based Leather Surface Defect Detection," *IEEE Access,* 2024.
[94] "Getting Started with Kria KV260 Vision AI Starter Kit (PetaLinux)." https://www.amd.com/en/products/system-on-modules/kria/k26/kv260-vision-starter-kit/getting-started/getting-started.html (accessed.
[95] M. Hatab, H. Malekmohamadi, and A. Amira, "Surface defect detection using YOLO network," in *Intelligent Systems and Applications: Proceedings of the 2020 Intelligent Systems Conference (IntelliSys) Volume 1*, 2021: Springer, pp. 505-515.
[96] Y. Cao *et al.*, "Improved yolov8-gd deep learning model for defect detection in electroluminescence images of solar photovoltaic modules," *Engineering Applications of Artificial Intelligence,* vol. 131, p. 107866, 2024.