T.C.

FIRAT ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

GÖMÜLÜ SİSTEM PLATFORMU ÜZERİNDE GÖRÜNTÜ İŞLEME TEKNİKLERİNİN UYGULANMASI

Sertaç YAMAN

Doktora Tezi

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Elektronik Bilim Dalı

HAZİRAN 2021

T.C.

FIRAT ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Elektrik Elektronik Mühendisliği Anabilim Dalı

Doktora Tezi

GÖMÜLÜ SİSTEM PLATFORMU ÜZERİNDE GÖRÜNTÜ İŞLEME TEKNİKLERİNİN UYGULANMASI

Tez Yazarı

Sertaç YAMAN

Danışman

Dr. Öğr. Üyesi Yavuz EROL

HAZİRAN 2021

ELAZIĞ

BEYAN

Fırat Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım "Gömülü Sistem Platformu Üzerinde Görüntü İşleme Tekniklerinin Uygulanması" Başlıklı Doktora Tezimin içindeki bütün bilgilerin doğru olduğunu, bilgilerin üretilmesi ve sunulmasında bilimsel etik kurallarına uygun davrandığımı, kullandığım bütün kaynakları atıf yaparak belirttiğimi, maddi ve manevi desteği olan tüm kurum/kuruluş ve kişileri belirttiğimi, burada sunduğum veri ve bilgileri unvan almak amacıyla daha önce hiçbir şekilde kullanmadığımı beyan ederim.

18.06.2021

Sertaç YAMAN

ÖNSÖZ

"ONLAR HER YERDE" Gömülü sistemler insanların olduğu ve olmadığı her yerde, etrafımızda gördüğümüz tüm sistemlerin içinde bulunan ve sahip oldukları yazılımlar ile kontrol edebilme özelliği olan elektronik donanımlardan oluşan platformlardır. Ancak bu sistemlerin çoğu kullanıldıkları sürece işlevlerini değiştirmemektedir. Örneğin kullandığımız otomobillerde araç kumandası, dijital klimalar, otomatik park etme, araç arızalarını tek bir ekranda gösterme gibi sistemlerin kullanımı süresince aynı görevde hata yapmadan çalışmaktadırlar.

Gömülü sistemlerin görevlerini sorunsuzca yerine getirebilmeleri için donanım ve yazılımın daha etkin kullanımı ve uygulama alanlarının ihtiyaçlarına göre farklılaştırıp geliştirmek gerekmektedir. Günümüzde görüntü kaydetme ve işleme yeteneğine sahip, yüksek hızlı, düşük güç tüketen ve hızlı uygulama geliştirme ortamı sağlayan System on Chip (SoC) olarak adlandırılan çeşitli platformlar bulunmaktadır. Bu platformlar genellikle ARM işlemci ve ek olarak sayısal işaret işleyici (DSP) birimlerini içermektedir

Gömülü sistem platformlarında hafıza yetersizliği, çalışma hızı, ısınma problemleri ve farklı sistemlere/uygulamalara entegre edilebilme gibi gereksinimleri gerçek zamanlı (online) uygulamalarda başlıca geliştirilmesi ve üstesinden gelinmesi gereken problemler arasında yer almaktadır. Öğrenebilen makineler ile birlikte görüntü işleme algoritmaları, gömülü sistemler ve bunlar ile ilişkili gereksinimler ve problemlerin çözümü daha da fazla önem kazanmıştır. Bu tez projesi kapsamında, görüntü işleme algoritmalarının gömülü sistemler üzerine uygulamalarını içermektedir. Görüntü işleme algoritmaları farklı yazılım dilleri ve gömülü sistem platformları kullanılarak uygulamalar geliştirilmiştir.

Bu tez çalışmamda beni yönlendiren, yaptığım çalışmalarda beni destekleyen, tecrübeleriyle bana katkıda bulunan danışman hocam Dr. Öğr. Üyesi Yavuz Erol'a ve tüm desteklerinden dolayı aileme teşekkür ederim.

Sertaç YAMAN

ELAZIĞ, 2021

İÇİNDEKİLER

Sayfa

ÖNSÖZ iv

İÇİNDEKİLER v

ÖZET vii

ABSTRACT viii

ŞEKİLLER LİSTESİ ix

TABLOLAR LİSTESİ xiv

SİMGELER VE KISALTMALAR xvii

1. GİRİŞ 1

1.1. Literatür Özeti 2

1.2. Tezin Amacı ve Kapsamı 34

1.3. Tezin Organizasyonu 35

2. GÖMÜLÜ SİSTEM TEKNOLOJİLERİ 36

2.1. Gömülü Sistem Platformlarının Teknik Özellikleri 36

2.1.1. Alanda Programlanabilen Kapı Dizileri (FPGA) 39

2.1.2. Raspberry Pi 45

2.1.3. Jetson Gömülü sistem Platformları 46

2.1.4. Çip Üzerine Programlanabilir Sistem (PSoC) 48

2.2. Bölüm değerlendirmesi 49

3. YAPAY ZEKÂ VE DERİN ÖĞRENME İÇİN TEMEL ALGORİTMALAR 50

3.1. Yapay Sinir Ağı Model Yapısı ve Değişkenleri 50

3.1.1. Denetimli ve Denetimsiz Öğrenme 52

3.1.2. Model Öğrenme parametreleri 52

3.1.3. Nöron ve Gizli Katmanlar 61

3.2. Evrişimli Sinir Ağları 61

3.3. Bölüm değerlendirmesi 63

4. TEMEL GÖRÜNTÜ İŞLEME UYGULAMALARI 65

4.1. Raspberry Pi Kullanılarak Gerçekleştirilen Uygulamalar 65

4.1.1. Görüntü Filtreme Tekniklerinin Uygulanması 65

4.1.2. Gerçek Zamanlı Yüz Tanıma Uygulaması 67

4.1.3. Tensorflow ve Raspberry Pi Platformunda Yaya ve Araç Plaka Tespiti 68

4.2. FPGA Platformunda HDL Tabanlı Tasarımlar 70

4.2.1. Xilinx Sistem Generatörü ile Görüntü İşleme Tekniklerinin Uygulanması 70

4.2.2. Xilinx Sistem Generatörü ile Görüntüye Morfolojik İşlemlerin Uygulanması 75

4.2.3. Xilinx Sistem Generatörü İle Video Görüntüye Sobel Filtre Uygulanması 77

4.2.4. Renkli Video Görüntüsünün VHDL Kod İle Elde Edilmesi: 78

4.2.5. Gri Tonlamalı Video Görüntüsünün VHDL Kod İle Elde Edilmesi: 82

4.3. FPGA Platformunda Gerçek Zamanlı IP Çekirdek Tabanlı Tasarımlar 82

4.3.1. Renkli Video Görüntüsünün Elde Edilmesi 83

4.3.2. Gri Tonlamalı Video Görüntüsünün elde edilmesi 84
4.3.3. Gerçek Zamanlı Video Görüntüsüne Eşik Değer Uygulanması 94
4.3.4. Görüntünün Pürüzsüzleştirilmesi 95
4.3.5. Sobel Kenar Algılama Tekniğinin Video Görüntüye Uygulanması 98
4.3.6. Canny Kenar Algılama Tekniğinin Video Görüntüye Uygulanması 101
4.3.7. Morfolojik İşlemlerin Video Görüntüye Uygulanması 104
4.4. Bölüm değerlendirmesi 106
5. ÖNERİLEN MVSR NORMALİZASYON TEKNİĞİ İLE GÖRÜNTÜ İŞLEME UYGULAMALARI 110
5.1. MVSR Normalizasyon Tekniği 110
5.2. MVSR Normalizasyon Tekniğinin X-ışını Görüntülerine Uygulanması 111
5.3. MVSR Normalizasyon Tekniğinin CNN Model Yapısında Görüntü Ön İşleme Uygulanması 115
5.4. Bölüm değerlendirmesi 118
6. ARAÇ PLAKA GÖRÜNTÜSÜ ÜZERİNDE ÖZELLİK ÇIKARIMI 120
6.1. Standart Görüntü İşleme Teknikleri ile Araç Plaka Tespiti ve Karakterlerin Ayrıştırılması 120
6.2. Önerilen MVSR Tekniği ile Tespiti ve Karakterlerin Ayrıştırılması 122
6.3. Araç Plaka Karakterlerinin CNN Model Kullanılarak Sınıflandırılması 128
6.4. Bölüm değerlendirmesi 130
7. SONUÇLAR VE ÖNERİLER 132
KAYNAKLAR 135
ÖZGEÇMİŞ

ÖZET

Gömülü Sistem Platformu Üzerinde Görüntü İşleme Tekniklerinin Uygulanması

Sertaç YAMAN

Doktora Tezi

FIRAT ÜNİVERSİTESİ

Fen Bilimleri Enstitüsü

Elektrik Elektronik Mühendisliği Anabilim Dalı

Haziran 2021, Sayfa: xviii + 144

Gömülü sistemler, görüntü işleme teknikleri ve derin öğrenme algoritmaları, nesne tanıma, robotik uygulamalar, savunma sanayi için geliştirilen otonom ve uzaktan kontrol edilebilen sistemler, medikal uygulamalar, yüz tanıma ve araç plaka okuma gibi uygulamalarda kullanılmaktadır. Gömülü sistem uygulamalarında tasarıma bağlı olarak algoritmaların boyutu ve karmaşıklığı artmaktadır. Özellikle gerçek zamanlı görüntü işleme uygulamalarında işlem yükü, algoritmaların hafızada kapladığı alan ve güç tüketimi daha çok artmaktadır. Bu yüzden, tasarımın en uygun donanım yapısında gerçeklemesi büyük önem arz etmektedir.

Bu tez kapsamında uygun gömülü sistem platformlarının seçimleri ile gerçekleştirilebilecek görüntü işleme uygulamaları hakkında detaylı bilgiler verilmektedir. Bilgisayar tabanlı yazılım programları, Raspberry Pi ve FPGA gömülü sistem platformlarında, yapay zeka modelleri kullanılarak uygulamalar gerçekleştirilmiştir. İlk olarak temel görüntü işleme algoritmalarını farklı yazılım dilleri ve platformlarında geliştirilerek uygulamalar gerçekleştirilmiştir.

Tez kapsamında gerçekleştirilen uygulamaların bilimsel katkıları ve yenilikleri; bu tez kapsamında görüntü işleme algoritmalarının gömülü sistem platformlarında gerçek zamanlı uygulanabilirliği araştırılmış ve geliştirilmiştir. Bu uygulamaların akıllı sistemler haline dönüştürülmesi için derin öğrenme modelleri, karşılaştırmalı sonuçlar ile geliştirilmiştir. Görüntü üzerinde özellik çıkarımı ve akciğer x-ışını görüntülerinde virüslü bölgenin teşhisi için yeni bir normalizasyon tekniği önerilmiştir. X-ışını görüntülerinde Covid-19 virüsü tespitini derin öğrenme modelleri kullanarak, önerilen normalizasyon tekniği ile Covid-19 virüsü tespit etme oranı %83.01'den %96.16'ya çıkartılmıştır. Ayrıca, tez kapsamında geliştirilen görüntü işleme teknikleri ve derin öğrenme modelini kullanarak farklı koşullar altında elde edilen görüntüler üzerinde araç plaka tanıma işlemleri gerçekleştirilmiş ve bu uygulamaların performansları deneysel sonuçlar ile doğrulanmıştır.

Anahtar Kelimeler: Görüntü işleme, Gömülü sistemler, Derin öğrenme, Plaka tanıma

ABSTRACT

Image Processing Techniques on Embedded System

Sertaç YAMAN

Ph.D. Thesis

FIRAT UNIVERSITY

Graduate School of Natural and Applied Sciences

Department of Electrical and Electronics Engineering

June 2021, Pages: xviii + 144

Image processing and embedded systems are used in many applications such as object recognition, robotic applications, autonomous and remote control systems developed for the defense industry, medical applications, face recognition, and vehicle license plate recognition. The size and complexity of the design depend on the algorithms. Therefore, the design must be implemented in the most suitable hardware structure. Especially, Power consumption, desired memory space, and speed of processing are crucial in real-time image processing applications.

In the scope of this thesis, image processing applications that can be executed with appropriate embedded system selections have been developed. The Applications were carried out using artificial intelligence models on computer-based software programs, Raspberry Pi, and FPGA embedded system platforms. In the first step, basic image processing algorithms were implemented in different software languages and related embedded platforms.

Scientific contributions and innovations of the applications carried out within the scope of the thesis; real-time applicability of image processing algorithms on embedded system platforms was investigated and developed. Deep learning models have been developed with comparative results in order to transform these applications into smart systems. A new normalization technique is proposed for feature extraction on the image and diagnosis of the infected region on x-ray images. The rate of detecting Covid-19 virus was increased from 83.01% to 96.16% with the proposed normalization technique by using deep learning models to detect Covid-19 virus in X-ray images. In addition, using the image processing techniques and deep learning model developed within the scope of the thesis, car license plate recognition processes were performed on the images obtained under different conditions, and the performances of these applications were verified with experimental results.

Keywords: Image processing, embedded systems, deep learning, License plate recognition

ŞEKİLLER LİSTESİ

Sayfa

Şekil 1.1. Kullanılan kaynak miktarı (%) ve Enerji tüketimi (mJ) 3

Şekil 1.2. Sistem Tasarımı 4

Şekil 1.3. PSoC 5LP'de görüntü geliştirme algoritmasının uygulanması için tasarım akışı 7

Şekil 1.4. PSoC 5LP kaynak kullanımı 7

Şekil 1.5. Önerilen tasarım 8

Şekil 1.6. Dron kontrol 9

Şekil 1.7. AlexNet model yapısı 11

Şekil 1.8. Başlangıç modülü 11

Şekil 1.9. Model sonuçları 15

Şekil 1.10. Fire modülü 16

Şekil 1.11. Önerilen CNN modeli 17

Şekil 1.12. Beklenmedik olay tepkisi 21

Şekil 1.13. Önerilen sistem yapısı 23

Şekil 1.14. Çin otomobil plakası tanıma sisteminin sistem mimarisi 26

Şekil 2.1. Gömülü sitem yapısı 37

Şekil 2.2. Gömülü sistemler uygulama alanları 39

Şekil 2.3. FPGA platformu genel yapısı 40

Şekil 2.4. Xilinx XC400 FPGA Yapılandırılabilir mantık bloğu yapısı 41

Şekil 2.5. FPGA proje tasarım töntemleri 43

Şekil 2.6. XSG akış diyagramı 44

Şekil 2.7. Raspbery Pi geliştirme kartı 45

Şekil 2.8. NVIDIA Jetson TX2 geliştirme kartı 47

Şekil 2.9. PSoC 4 Geliştirme kartı 48

Şekil 3.1. Biyolojik sinir hücre yapısı [104] 51

Şekil 3.2. Yapay sinir hücresi matematiksel modeli 51

Şekil 3.3. Optimizasyon algoritmalarının öğrenmeye etkisi 54

Şekil 3.4. Hata ve ağırlığın karşılaştırılması 54

Şekil 3.5. Öğrenme oranının döngü sayısına göre güncellenmesi 56

Şekil 3.6. Ortalama ve maksimum ortaklama tekniği 57

Şekil 3.7. Basamak fonksiyonu ve türevi 58

Şekil 3.8. Sigmoid aktivasyon fonsiyonu ve türevinin grafiği 58

Şekil 3.9. Hiperbolik tanjant aktivasyon fonsiyonu ve türevinin grafiği 59

Şekil 3.10. ReLu aktivasyon fonsiyonu ve türevinin grafiği 59

Şekil 3.11. Leaky-ReLu aktivasyon fonsiyonu ve türevinin grafiği 60

Şekil 3.12. Softmax aktivasyon işlemi 60

Şekil 3.13. Yapay sinir ağı modeli 61

Şekil 3.14. Evrişimli sinir ağı yapısı 62

Şekil 3.15. Tam bağlantı dönüşümü 63

Şekil 4.1. Raspberry Pi ile gerçek zamanlı video görüntüsüne filtre uygulamaları, (A) gauss filtre, (B) sobel kenar algılama, (C) görüntünün tersi, (D) siyah-beyaz görüntü, (E) Morfolojik açma, (F) morfolojik kapama sonuçları 66

Şekil 4.2. Evrişim işlemi temsili kodları 67

Şekil 4.3. Raspberry Pi ile yüz tanıma uygulaması 68

Şekil 4.4. Tensorboard model eğitim sonucu 69

Şekil 4.5. Raspberry Pi gerçek zamanlı plaka ve yaya tespiti 70

Şekil 4.6. Renkli görüntünün çerçevelerine ayrılması 71

Şekil 4.7. Görüntünün yeniden boyutlandırılıp gösterilmesi 71

Şekil 4.8. Renkli görüntüden gri tonlu görüntünün elde edilmesi 72

Şekil 4.9. RGB to gray dönüşümü XSG alt sisteminin iç yapısı 72

Şekil 4.10. RGB to gray dönüşümü 72

Şekil 4.11. XSG Görüntü tersleme uygulaması 73

Şekil 4.12. Gri tonlamalı görüntüye eşik değer uygulanması 73

Şekil 4.13. XSG ile görüntünün kuvvetlendirilmesi 74

Şekil 4.14. XSG ile eşik değer uygulaması 74

Şekil 4.15. Gri tonlamalı görüntüye eşik değer ataması uygulaması 75

Şekil 4.16. Morfoloik işlemler için piksellerin kontrolü 75

Şekil 4.17. XSG ile morfolojik aşındırma ve yayma uygulamarı 76

Şekil 4.18. Morfolojik açma ve kapama uygulaması sonuçları 77

Şekil 4.19. XSG ile kenar belirleme uygulaması 77

Şekil 4.20. Kenar bulma uygulaması 78

Şekil 4.21. OV7670 Kamera modülü blok diyagramı 79

Şekil 4.22. OV7670 RGB565 çıkış diyagramı [125] 79

Şekil 4.23. VHDL dili kullanılarak renkli görüntünün Vivado 2018.1 ara yüzü gösterimi 80

Şekil 4.24. Zedboard FPGA uygulama düzeneği 81

Şekil 4.25. Renkli görüntünün sabit noktalı sayi kullanılarak gri tonlu görüntüye dönüştürülmesi 82

Şekil 4.26. IP Çekirdekleri ile görüntünün elde edilmesi 83

Şekil 4.27. FPGA, Renkli video görüntüsünün elde edilmesi 83

Şekil 4.28. Sabit noktalı ikili sayı sistemi gösterimi 85

Şekil 4.29. Kayan noktalı ikili sayı sistemi gösterimi 85

Şekil 4.30. Kayan noktalı sayı sistemi aritmetik çarpma işlemi 87

Şekil 4.31. Kayan noktalı sayı sisteminde aritmetik toplama işlemi aşamaları 88

Şekil 4.32. Gri tonlamalı görüntünün elde edilmesi 89

Şekil 4.33. RGB ve Gri tonlamalı video görüntüsü 89

Şekil 4.34. Analog Discovery 2 platformu 90

Şekil 4.35. Analog Discovery 2 platformunda sayısal verilerin elde edilmesi 91

Şekil 4.36. RGB-Gri tonlamalı görüntü dönüşümü piksel değerlerinin karşılaştırılması 91

Şekil 4.37. RGB-Gri tonlamalı dönüşümlerine ait (A)'da MATLAB, (B)'de sabit noktalı sayı ve (C)'de kayan noktalı sayı sistemi ile gerçekleştirilen piksel değerleri 92

Şekil 4.38. Siyah-Beyaz görüntünün elde edilmesi 94

Şekil 4.39. Piksel hafıza birimi 95

Şekil 4.40. Görüntünün pürüzsüzleştirilmesi 96

Şekil 4.41. Görüntünün pürüzsüzleştirilmesinde elde edilen sonuçlara ait piksel değerleri dağılımının karşılaştırılması 96

Şekil 4.42. Görüntünün pürüzsüzleştirilmesinde (A)'da MATLAB, (B)'de sabit noktalı sayı ve (C)'de kayan noktalı sayı sistemi ile gerçekleştirilen piksel değerleri 97

Şekil 4.43. Sobel filtre matrisleri 98

Şekil 4.44. IP bloklar ile gerçek zamanlı video görüntünün kenarlarının çıkartılması 99

Şekil 4.45. Sobel kenar algılama tekniğinin uygulanması ile elde edilen sonuçlara ait piksel değerleri dağılımının karşılaştırılması 99

Şekil 4.46. Sobel kenar algılama (A)'da MATLAB, (B)'de sabit noktalı sayı ve (C)'de kayan noktalı sayı sistemi ile gerçekleştirilen piksel değerleri 100

Şekil 4.47. FPGA Sobel kenar algılama VGA monitör görüntüsü 101

Şekil 4.48. Canny operatörü filtre matrisleri 101

Şekil 4.49. Canny kenar algılama tekniği ile görüntüde bulunan kenarların elde edilmesi 102

Şekil 4.50. Canny kenar algılama tekniğinin filtreleme işleminden sonra kenarların tespit edilmesi 102

Şekil 4.51. Yapısal elemanlar 105

Şekil 4.52. Morfolojik işlemlerin gerçek zamanlı video görüntüsüne uygulanması 105

Şekil 4.53. Morfolojik işlemlerin karşılaştırılması 107

Şekil 4.54. Sabit ve kayan noktalı sayı sistemlerinin piksel yoğunluğu hassasiyetliliğinin karşılaştırılması 107

Şekil 5.1. Covid-19 virüslü akciğer röntgeni orijinal ve MVSR uygulanmış görüntüsü 111

Şekil 5.2. Virüs tespit edilmeyen akciğer röntgeni orijinal ve MVSR uygulanmış görüntüsü 111

Şekil 5.3. Covid-19 virüslü farklı hastalara ait akciğer röntgeni orijinal ve MVSR uygulanmış görüntüler 112

Şekil 5.4. Virüs tespit edilmeyen farklı hastalara ait akciğer röntgeni orijinal ve MVSR uygulanmış görüntüler 112

Şekil 5.5. Covid-19 virüs tespit edilmeyen akciğer röntgeni orijinal ve MVSR uygulanmış görüntüsü FPGA sonucu 114

Şekil 5.6. Görüntü matrisi, piksel haritası ve renk paleti 114

Şekil 5.7. CNN Model sonuçlarının tahmin edilen ve hesaplanan değerlerinin karşılaştırılması 116

Şekil 5.8. Önerilen MVSR normalizasyon tekniği 118

Şekil 6.1. Araç plaka tanıma aşamaları 120

Şekil 6.2. (A) farklı açılarda ve (B) dik açıda elde edilen görüntülerde araç plakasının tespiti 121

Şekil 6.3. (A) Araç rekli görüntü, (B) gri tonlamalı görüntü ve (C) MVSR normalizasyonun uygulanması 122

Şekil 6.4. (A) Araç rekli görüntü, (B) gri tonlamalı görüntü ve (C) MVSR normalizasyonunda eşik değer uygulanması 122

Şekil 6.5. Plaka bölgesi karakterleri olabilecek kısımların tespiti 123

Şekil 6.6. Plaka bölgesi olabilecek yerlerin tespiti ve orijinal görüntü üzerinde gösterilmesi 124

Şekil 6.7. Aday plaka bölgeleri 124

Şekil 6.8. Farklı açılara sahip araç plakalarının MVSR normalizasyonu ile tespiti 125

Şekil 6.9. Plaka bölgesinin elde edilmesi 126

Şekil 6.10. Karakterlerin ayrıştırılması 126

Şekil 6.11. Plaka bölgesi yatay ve dikey yansıtma tekniği 127

Şekil 6.12. Plaka karakterlerinin ayrıştırılması 127

Şekil 6.13. Araç plaka karakterlerine ait model doğruluk grafiği 129

Şekil 6.14. Araç plaka karakterlerine ait model kayıp grafiği..... 130

TABLOLAR LİSTESİ

Sayfa

Tablo 1.1. Karşılaştırma tablosu 2
Tablo 1.2. Tasarım karşılaştırma sonuçları 4
Tablo 1.3. Karşılaştırma tablosu 5
Tablo 1.4. GPU ve FPGA ile gerçekleştirilmiş uygulamalar 5
Tablo 1.5. CNN modelinin performans karşılaştırması 6
Tablo 1.6. FPGA kaynak kullanımı sonuçları 8
Tablo 1.7. LeNET-5 CNN model 10
Tablo 1.8. FPGA Kaynak kullanımı 12
Tablo 1.9. Modellerin karşılaştırmalı sonuçları 12
Tablo 1.10. Sınıflandırma sonuçları 14
Tablo 1.11. Model karşılaştırma sonuçları 16
Tablo 1.12. Model sonuçlarının karşılaştırılması 17
Tablo 1.13. Model sonuçlarının karşılaştırılması 18
Tablo 1.14. Değiştirilen katman sayısı ve doğruluk oranları 18
Tablo 1.15. Önerilen CNN modellerinin sonuçları 19
Tablo 1.16. Önerilen sistemin sonuçları 20
Tablo 1.17. Önerilen sistemin sonuçları 23
Tablo 1.18. Önerilen Sistemin sonuçları 24
Tablo 1.19. Kullanılan kaynak miktarı 24
Tablo 1.20. Karakterlerin ayrıştırılması sonuçları 26
Tablo 1.21. FPGA platformunda kullanılan kaynak miktarı 27
Tablo 1.22. Farklı veri setlerinde plaka okuma 28
Tablo 1.23. Plaka bölgesinin belirlenme oranları 29
Tablo 1.24. Farklı veri setlerine ait önerilen modelin sonuçları 30
Tablo 1.25. Farklı veri setlerinde plaka belirleme sonuçları 31
Tablo 1.26. Önerilen sistemin doğruluk analizi 31
Tablo 1.27. Araç plaka tanıma sistemlerine ait literatürde gerçekleştirilen uygulamalar 33

Tablo 1.28. Akciğer X-ışını görüntülerinin farklı normalizasyon teknikleri ile sınıflandırılması 34

Tablo 2.1. FPGA platformlarının üretim teknolojileri 41

Tablo 2.2. FPGA platformlarının karşılaştırılması 42

Tablo 2.3. Jetson gömülü sistem platformlarının karşılaştırılması 47

Tablo 2.4. PSoC gömülü sistem platformlarının karşılaştırılması 48

Tablo 4.1. FPGA kaynak kullanımı 81

Tablo 4.2. FPGA Güç Tüketimi Miktarının Karşılaştırılması 92

Tablo 4.3. FPGA Kaynak Kullanımı Karşılaştırma Tablosu 93

Tablo 4.4. Renkli ve gri tonlamalı görüntü için kullanılan FPGA kaynak miktarları 93

Tablo 4.5. Siyah-Beyaz görüntü için FPGA Kaynak kullanımı 94

Tablo 4.6. Görüntünün pürüzsüzleştirilmesi işleminde kullanılan FPGA kaynak miktarları 97

Tablo 4.7. FPGA Sobel filtre kaynak kullanımları 100

Tablo 4.8. Canny kenar algılama tekniği FPGA kaynak kullanımı miktarı 103

Tablo 4.9. Kenar algılama tekniği FPGA kaynak kullanımı miktarlarının karşılaştırılması 103

Tablo 4.10. Kenar algılama tekniği FPGA kaynak kullanımları 104

Tablo 4.11. Sayı sistemlerinin performanslarının karşılaştırılması 104

Tablo 4.12. Morfolojik işlemlerin gerçekleştirilmesinde kullanılan FPGA kaynak miktarları 106

Tablo 4.13. FPGA Güç Tüketimi Miktarının Karşılaştırılması 108

Tablo 4.14. FPGA Kaynak Kullanımı Karşılaştırma Tablosu 108

Tablo 4.15. FPGA platformunda HDL ve IP Bloklar ile gerçekleştirilen uygulamaların kıyaslanması 109

Tablo 4.16. Kenar algılama tekniklerinde kullanılan FPGA kaynak kullanımı 109

Tablo 5.1. VHDL sabit noktalı sayı sistemi bölme işlemi temsili kod gösterimi 113

Tablo 5.2. Sabit noktalı sayı sisteminde karekök alma işlemi VHDL kodu 113

Tablo 5.3. MVSR Normalizasyonu FPGA kaynak kullanımı 115

Tablo 5.4. MVSR normalizasyonu için tasarlanan CNN modeli 115

Tablo 5.5. Covid-19 F1-Skor tablosu 117

Tablo 5.6. Farklı normalizasyon teknikleri ile gerçekleştirilen Covid-19 virüsü sınıflandırma uygulamalarının karşılaştırılması 117

Tablo 5.7. Farklı normalizasyon teknikleri kullanılarak gerçekleştirilen çalışmaların önerilen sistem ile karşılaştırılması 119

Tablo 6.1. Plaka tanıma oranları 127

Tablo 6.2. Farklı görüntü işleme teknikleri ile araç plaka bölgesinin tespit edilmesinin karşılaştırılması 128

Tablo 6.3. Plaka karakterleri için oluşturulan CNN modelinin tasarımı 129

SİMGELER VE KISALTMALAR

Simgeler

$\hat{y}_i$	: Hesaplanan değer
w_i	: Ağırlık
b_i	: Bias
H	: Gizli katmandaki nöronlar
σ	: Aktivasyon fonksiyonu
∂J	: Hata fonksiyonu değişim miktarı
α	: Öğrenme katsayısı

Kısaltmalar

CPU	: Merkezi işlem birimi
ASIC	: Özel tümleşik devreler
FPGA	: Alanda programlanabilen kapı dizileri
DSP	: Dijital sinyal işleme
MAC	: Çarpma-Toplama birimi
HDL	: Donanım tanımlama dili
GPU	: Grafik İşlemci Ünitesi
DNN	: Derin yapay sinir ağı
SIFT	: Ölçekle değişmeyen özellik dönüşümü
BRIEF	: İkili Bağımsız Temel Özellikler
CNN	: Evrişimli yapay sinir sağı
VGG	: Görsel Geometri Grubu
MR	: Manyetik Rezonans
HOG	: Yönelimli eğimlerin dağılımı
KNN	: K-en yakın komşuluk
SVM	: Destek vektör makineleri
RCNN	: Bölge tabanlı Evrişimli Sinir Ağı
OpenCV	: Açık Kaynak Bilgisayar Görüşü
OCR	: Optiksel karakter tanıma
CCD	: Şarj-bağlı Cihaz (Charge-coupled Device)
RGB	: Kırmızı Yeşil Mavi
I2C	: Entegre devreler
YSA	: Yapay Sinir Ağları
SGD	: Stokastik eğim azalması
Adam	: Uyarlanabilir Moment Tahmini
MSE	: Ortalama kare hatası
ReLU	: Doğrultulmuş Doğrusal Birim
LR	: Öğrenme katsayısı
ResNet	: Artık Derin Öğrenme
FM	: Özellik Haritası
BN	: Küme normalizasyonu (Batch-Normalization)
CLB	: Yapılandırılabilir Mantık blokları
LUT	: Arama Tablosu (LookUp Table)
IP	: Entelektüel Özellik

XSG	: Xilinx sistem generatörü
MVSR	: Ortalama Varyans Softmax Ölçeklendirme
TP	: Gerçek Pozitifler
TN	: Gerçek Negatifler
FP	: Yanlış Pozitifler
FN	: Yanlış Negatifler

1. GİRİŞ

Görsel, yazılı ve işitsel bilgiler her saniye üretilmekte ve farklı uygulama alanlarında işlenmektedirler. Günümüzde en yaygın olarak görsel, görüntü işleme tabanlı akıllı uygulamalar kullanılmaktadır. Günümüz teknolojilerinde robotik, savunma sanayi, tıp ve akıllı otonom sistemler (insansız kara, deniz ve hava araçları vb.) gibi alanlarda görüntü işleme algoritmalarının gömülü sistem platformlarına uygulanması ve giderek pek çok alanda kullanılması problemin önemini oluşturmaktadır.

Bu uygulamaları gömülü sistem platformlarında inşa ederken, algoritmanın uzunluğu, hafızada kapladığı alan, güç tüketimi ve veri işleme hızı gibi etkenler göz önünde bulundurulması gerekmektedir. İnşa edilen her sistem beraberinde çözülmesi gereken pek çok problemi de beraberinde getirecektir. Uygun algoritmaların oluşturulması ve tasarımları uygulamaya dönüştürecek doğru gömülü sistem platformunun seçimi gerekmektedir.

Gömülü sistemler kullanılarak gerçekleştirilen uygulamalarda, oluşturulacak uygulamaya göre algoritmaların boyutu, niteliği ve karmaşıklığı değişmektedir. Bu uygulamalar için kullanılacak karmaşık algoritmaların en uygun donanım yapısında gerçeklemesi gerekmektedir. Özellikle gerçek zamanlı görüntü işleme uygulamalarında işlem yükü, algoritmaların hafızada kapladığı alan ve güç tüketimi daha çok artmaktadır. Bu nedenlerden dolayı, kullanılacak platformların her birinin farklı ram yapısı ve güç tüketimi dikkate alınarak görüntü işleme algoritmalarının geliştirilmesi gerekmektedir.

Renk dönüşümü, kenar bulma, morfolojik işlemler ve filtreleme gibi görüntü işleme aşamaları için kullanılacak algoritmalar amaçlanan hedefe kadar birbirini takip eden görüntü işleme süreçlerinden oluşmaktadır. Algoritmaların kapsamı, uygun senaryolar içeren alternatif akış diyagramları, farklı platformların kodlama dilleri arasındaki farklılıklar ve kullanılan ara yüzlerde mevcut kütüphane fonksiyonlarının kullanımı gibi etkenler hafıza sorunu (kullanılan kaynaklar), güç tüketimi ve veri işleme hızı gibi tasarım değişkenlerini etkileyen başlıca konular arasında yer almaktadır. Bu kapsamda, gerçek zamanlı uygulamalar için görüntü işleme algoritmalarını geliştirmek ve bu algoritmaların uygun gömülü sistem platformlarında çalıştırılmasını sağlamak amaçlanmıştır.

Teknolojinin bu yöne doğru evirilmesinin nedeni tarihin başlangıcından yaşadığımız ana kadar doğayı taklit etmemizden kaynaklanmaktadır. Doğada görebilen canlı varlıkların gözleriyle görüp ve bu gördüklerini beyninde anlamlandırarak işleme olayı görüntü işleme, yapay zekâ ve gömülü sistemlerin ortaya çıktığı yerdir. Bu olaylar bilgisayar görüşü alanına girmektedir. İnsan zekâsının makineler tarafından taklit edilebilmesi şimdilik kodlama, yapay zekâ algoritmaları ve gömülü sistemler aracılığı ile yapılmaktadır. Denetimli (insanlar aracılığıyla) veya kendi kendilerine öğrenebilen algoritmaların yaptığı işlere “Makine Öğrenimi” denilmektedir. Gömülü

sistem platformlarının akıllı görüntü işleme uygulamaları özellikle de son yıllarda araştırma ve proje konularının başında yer almaktadır. Aynı zamanda ülkemizde öncelikli alanlar arasında bulunmaktadır.

1.1. Literatür Özeti

Video analizi ve bilgisayarlı görme sistemlerinde görüntü özelliklerinin algılanması ve istenilen özelliklere göre eşleştirilmesi veya sınıflandırılması yaygın olarak araştırılmaktadır. Gerçekleştirilen araştırmalar genel olarak farklı gömülü sistem platformlarının etkisi, görüntü işleme algoritmalarının tasarımlara katkısı ve derin öğrenme modelleri ile evirilmiş akıllı sistemlerin tasarlanması üzerine gerçekleştirilmektedir.

Kuon ve çalışma grubu tarafından gerçekleştirilen çalışmada[1], sistem tasarımcıları için standart özel tümleşik devreler (ASIC: Application Specific Integrated Circuit) ile alanda programlanabilen kapı dizilerini (FPGA: Field Programmable Gate Array) sahip oldukları mantık bloklarının yoğunluğu, işlem hızları ve güç tüketimleri bakımından karşılaştırmışlardır. Bu çalışma için Altera Stratix II çipi TSMC's Nexsys 90-nm üretim teknolojisi kullanılarak üretilmiş ve standart hücrelerin üretiminde STMicroelectronic'in CMOS090 Tasarımı kullanılmıştır. Karşılaştırmalı değerlendirme için açık kaynak Opencores'dan yapılar seçilmiş ve deneysel olarak karşılaştırma yapılmıştır. Bu karşılaştırma için kullanılan bazı yapılar şunlardır,

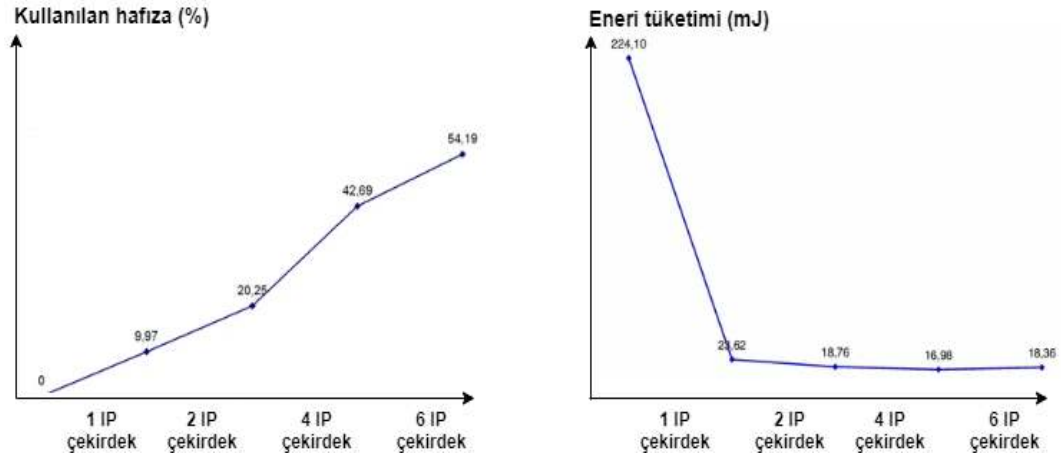
- Mac1: Ethernet Media Access Control (MAC) blok, Opencores'dan
- rs encoder: Reed Solomon kodlayıcı (encoder), Opencores'dan
- cordic8: 8-bit CORDIC algoritması, Opencores'dan
- diffeq: Diferansiyel denklem çözücü, Opencores'dan

Tablo 1.1. Karşılaştırma tablosu[1]

Dizayn	Güç Tüketimi (W)	
	FPGA	**ASIC**
Rs_encoder	1.31×10−02	2.61×10−04
Cordic18	4.43×10−02	5.73×10−04
Des_area	1.14×10−02	1.25×10−04
Des_pref	5.52×10−02	1.08×10−03
Fir_restruct	1.40×10−02	2.03×10−04
Mac1	3.52×10−02	4.08×10−04
Aes192	1.61×10−02	1.90×10−04
Diffeq2	1.15×10−02	3.63×10−04
Molecular	1.27×10−01	1.83×10−03

Tablo 1.1'de FPGA ve ASIC çiplerinin güç tüketimin bakımından kıyaslanması için gerçekleştirilen yapılara ait deneysel sonuçlar verilmektedir. Bu sonuçlara göre FPGA yapılarında güç tüketiminin daha fazla olduğu görülmektedir. Tamamen LUT tabanlı mantık öğeleri kullanılarak uygulanan devreler için FPGA'nın yaklaşık 35 kat daha büyük olduğu ve 3.4s ile 4.6s kat arasında daha yavaş oldukları tespit edilmiştir.

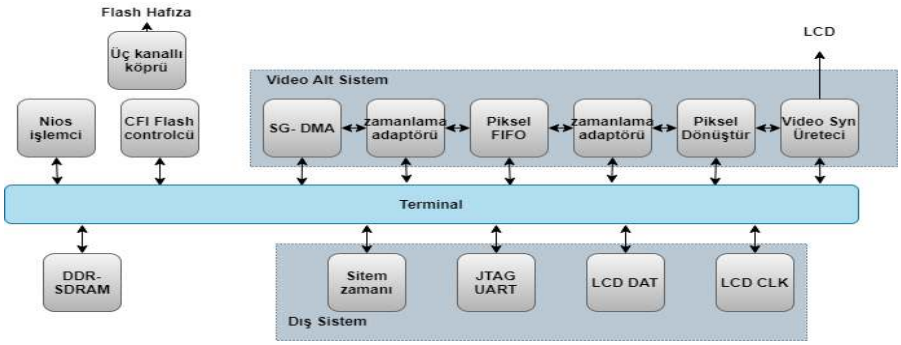
Pedre ve çalışma grubu tarafından gerçekleştirilen çalışmada[2], gerçek zamanlı görüntü işleme uygulamaları geliştirilirken karşılaşılacak kısıtlamaları (güç tüketimi ve hafıza sorunu vb.) minimize etmeye çalışmışlardır. Donanım ve yazılım ortak tasarım sistemler ile sadece yazılım ara yüzleri kullanarak oluşturulan projelere benzer geliştirme süresine ulaşmayı hedeflemişlerdir. Nesne tabanlı tanıma, tümleşik ve detaylı yazılım dili ve C dili ile kod üretme (HDL) araçlarını kullanarak gerçekleştirmişlerdir. Bu tasarım, görüntü sistemlerinde çoklu robot lokalizasyonu için yeni bir algoritmaya uygulanmış ve gömülü gerçek zamanlı görüntü işleme uygulamalarında kullanılabilir olduğunu göstermektedirler.



Şekil 1.1. Kullanılan kaynak miktarı (%) ve Enerji tüketimi (mJ)[2]

Şekil 1.1'de bu çalışma için tasarımı farklı sayıda IP çekirdekleri ile gerçekleştirip bu IP çekirdeklerini paralel çalıştırarak tasarlamışlardır. Gerçekleştirilen çalışmada, kullanılan kaynak miktarının artmasına karşılık enerji tüketimi azalmıştır ve veri işleme hızını 16 kat ve %92 oranında daha az enerji tüketerek gerçekleştirmişlerdir.

Fradi tarafından gerçekleştirilen çalışmada[3], Quartus II'nin Qsys araçlarını kullanarak programlanabilir çip dizayn ederek, bu sistemin performansını ve güç tüketimini test etmek için genel olarak kullanılan görüntü işleme algoritmaları kullanılmıştır. **Şekil 1.2**'de tasarlanan sistem gösterilmektedir.



Şekil 1.2. Sistem Tasarımı[3]

Şekil 1.2'de tasarım verilen bu sistem ile Altera Cyclone III ve Stratix II platformunda karşılaştırma yapılarak görüntü işleme uygulamaları kaynak kullanımı ve güç tüketimi karşılaştırılmıştır.

Tablo 1.2. Tasarım karşılaştırma sonuçları[3]

	Cyclone III	**Stratix II**
DSP	0	8
Lojik element	1415	6873
Hafıza Alanı	43.008/608.256	1154,816 / 2544,192
Frekans	102 MHz	120 MHz

Tablo 1.2'de verilen sonuçlarda tasarlanan sistemde kaynak kullanımının daha az olduğu ve tasarımı yapılan sistemin başarısı görülmektedir. Bu sistemin üzerinde bulunan LCD ekran sayesinde görüntü işlemede yapılan işlemler (filtreleme, kenar bulma algoritmaları gibi) anlık olarak görüntülenebilmektedir.

Falsafi ve çalışma grubu[4], dört farklı panel programında (Bilgisayar Mimarisi Araştırma Yönergeleri) veri merkezlerinde GPU'lara karşı FPGA yapılarını kıyaslanmışlardır. Programlanabilirlik açısından GPU'ların kütüphane fonksiyonlarına sahip olmaları, farklı programlama dilleri ile programlanabilir olmaları açık kaynak kod kullanımına (OpenCV, OpenMP vb.) elverişli olmaları GPU'ları daha efektif yapılar oluşmaktadır. Caffe ve Theano gibi çok hızlı GPU bağlantı noktalarına sahip olduklarından, derin sinir ağları geliştiricilerin programlama kütüphaneler aracılığı ile tasarımlarını gerçekleştirebilmektedirler. Bu nedenle yazılımı temelde web'den indirebilir ve anında çalıştırılabilir. Sabit ve kayan noktalı sayı aritmetikleri, kullanılarak gerçekleştirilen uygulamalarda, sabit noktalı sayı aritmetiğinin büyük verilerde daha az kaynak kullanımı ve daha hızlı işlem süreçlerinin gerçekleştirildiğini vurgulamışlardır. GPU'lar kayan noktalı sayı formatları için tasarlanmıştır.

HajiRassouliha ve diğerleri[5] bu çalışmalarında, donanım hızlandırıcıları seçerken göz önünde bulundurulması gereken önemli noktaları vurgulamışlardır. Merkezi işlem birimleri

(CPU'lar), hesaplamaları yeterli hızda gerçekleştiremedikleri için, hesaplama süresini azaltacak algoritmalar, dijital sinyal işlemcileri (DSP'ler), FPGA platformları ve GPU'lar gibi donanım hızlandırıcılar hakkında pratik bilgiler sunmaktadırlar. Aynı zamanda, DSP'lerin, FPGA'ların ve GPU'ların göreceli avantajları ve dezavantajlarını açıklamışlardır.

Tablo 1.3. Karşılaştırma tablosu[5]

DSP	FPGA	GPU
-Düşük güç tüketimi	-Düşük güç tüketimi	- Nispeten düşük güç tüketimi
-Düşük maliyetli	-Taşınabilir	- Hızlı bir geliştirme süresi gerektirir
-Taşınabilir	-Hesaplama açısından basit ve karmaşık algoritmalar	- Hesaplama açısından basit ve karmaşık algoritmalar
-Hesaplama açısından basit algoritmalar	-Esnek veya özel tasarlanmış donanımdan yararlanabilen algoritmalar	- Veri-paralel hale gelebilen algoritmalar.

Tablo 1.3'te DSP, FPGA ve GPU'ların bazı üstün yönleri verilmiştir. Bu çalışmada, Tasarımcılara kullanılacakları matematiksel aritmetiklere bağlı olarak uygun DSP'nin seçimi için önerilerde bulunmuşlardır. Örneğin sabit noktalı sayı aritmetiği için C5000 veya C64x DSP'ler kullanılabilir. Kayan noktalı sayı aritmetiği işlemlerini tercih edecek kişiler, C671x, C672x DSP ailesinden seçimler gerçekleştirmelidirler. Ayrıca, her iki sayı formatında da DM81x, C66x gibi DSP'ler önerilmektedir.

Tablo 1.4. GPU ve FPGA ile gerçekleştirilmiş uygulamalar

Uygulama	FPGA	GPU
-2Boyutlu filtre, stereo Görme, k-ortalama[6]	-Xilinx Virtex-4: Stereo görüş ve k-ortalama kümeleme için GPU'dan daha iyi performanslı.	- NVidia GTX 280: 11'den küçük filtre boyutları için FPGA'dan daha iyi performanslı.
-2Boyutlu Evrişim[7]	-Altera Stratix-3: 1280x720 piksel görüntü boyutu ve 25 çekirdek için 80 fps.	- NVidia GTX 295: 1280x720 piksel görüntü boyutu ve 25 çekirdek için 120 fps.
-1Boyutlu Evrişim[7]	-Altera Stratix-3: Küçük evrişimli çekirdek boyutlarına sahip büyük sinyal boyutları için GPU'dan daha iyi performanslı.	- NVidia GTX 295: Büyük evrişimli çekirdek boyutlarına sahip küçük veya büyük sinyal boyutları için FPGA'lardan daha iyi performanslı.
-3B ultrason bilgisayarlı tomografi görüntü işleme[8]	-Xilinx Virtex-7: Uygulama, GPU'dan 1.86 daha hızlı.	- NVidia Tesla K20: Uygulama FPGA'dan 1.86 daha yavaş.

Tablo 1.4'te GPU ve FPGA platformlarını birbirilerine göre kıyaslayarak gerçekleştirilmiş olan görüntü işleme uygulamalarını karşılaştırmışlardır. 1 boyutlu evrişim işleminin gerçekleştirilmesinde büyük sinyal gruplarında FPGA (Stratix-3) daha iyi performans gösterirken, küçük sinyal ise gruplarında NVIDIA GTX295 daha iyi performans göstermiştir.

Mittal tarafından gerçekleştirilen çalışmada[9], gömülü sistemlerde popüler olarak kullanılan Jetson geliştirme kartlarını kullanarak, sahip oldukları özellikler bakımından FPGA ve Raspberry Pi ile üstün yönlerini kıyaslamıştır. Jetson platformlarında açık kaynak kütüphaneler (OpenCL, cuDNN:Deep Neural Network library vb.) kullanılabildiği için kodlama aşamaları daha sade ve güç tüketimi daha az olacağını göstermiştir. Bu platformun sahip olduğu GPU C programlama dili ile programlanabilen NVIDIA'nın kullanıcılarına sunduğu CUDA[10] ile programlanabilmektedir. Geleneksel olarak kullanılan tüm derin öğrenme modelleri açık kod kütüphaneler aracılığı ile kolaylıkla kullanılabilmektedir. Kullanım kolaylığı ve boyutlarının da küçük olmasından dolayı IoT cihazlarda oldukça tercih edildiğini göstermiştir.

Suzen ve arkadaşları tarafından yürütülen çalışmada[11], Raspberry Pi ve Jetson geliştirme kartı ile farklı boyutlarda veriler için sinir ağı modeli kullanılarak sınıflandırma işlemi yapıldığında, sinir ağının sınıflandırmadaki doğruluk oranının, veri işleme hızının, kullanılan hafıza ve güç tüketimlerinin karşılaştırılması sonucunda Jetson kartlarının daha iyi sonuçlar verdiği gösterilmiştir. Veri işleme hızlarının daha yüksek, güç tüketimlerinin daha az ve artan veri kapasitesinde daha da iyi sonuçlar alındığı belirtilmiştir.

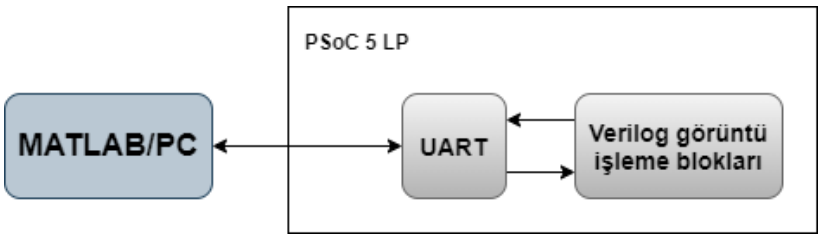
Tablo 1.5. CNN modelinin performans karşılaştırması[11]

Veri Seti	Doğruluk(%)			CPU(Güç/W)			GPU(W)		
	TX2	Nano	RasP	TX2	Nano	RasP	TX2	Nano	RasP
5K	87.6	87.5	87.2	2.23	1.5	3.5	5.27	2.23	-
10K	93.8	93.9	91.6	2.78	2.32	3.6	5.32	3.25	-
20K	94.6	94.5	-	3.76	-	3.9	5.22	-	-
30K	96.4	-	-	4.25	-	-	5.74	-	-
45K	97.8	-	-	4.92	-	-	6.29	-	-

Moda veri kümesi sınıflandırması için geliştirilen tek kartlı bilgisayarlarda CNN modelinin performans testlerinin sonuçları **Tablo 1.5**'te verilmiştir. Jetson TX2'nin yüksek donanım özelliklerinden dolayı daha yüksek kaynak ve güç tüketimine sahip olduğu gösterilmiştir. 20K veri setinden düşük olan veriler için, CPU güç tüketimi Raspberry Pi de en yüksek, Jetson nano da ise en düşük olarak test edilmiştir. Ancak yüksek boyutlu veri setleri kullanılarak gerçekleştirilecek uygulamalar için TX2 platformunun seçilmesi gerekmektedir.

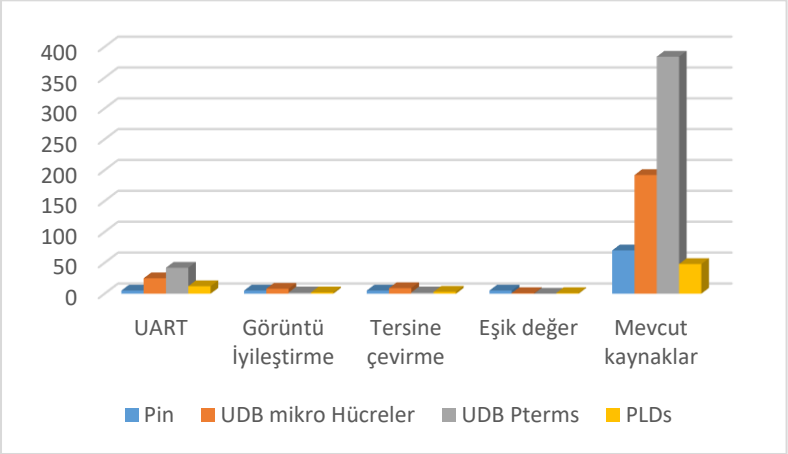
Hussain ve arkadaşları[12], temel görüntü geliştirme tekniklerini PSoC (Programmable System on Chip) gömülü sistem platformu ve MATLAB programını kullanarak yapılandırılabilir

bloklar kullanarak gerçekleştirmişlerdir. MATLAB benzetim ortamında görüntüler önce gri tonlamalı görüntüye dönüştürülür ve daha sonra bir boyutlu matris olarak, UART seri haberleşme ile PSoC 5LP kartına gönderilerek görüntü işleme tekniklerini uygulamışlardır. Cypress PSoC ara yüz programında Verilog programının kullanılabilmesi için geliştirilen yapılar kullanılarak, görüntü işleme gerçekleştirilmiştir.



Şekil 1.3. PSoC 5LP'de görüntü geliştirme algoritmasının uygulanması için tasarım akışı

Şekil 1.3'te PSoC 5LP platformu kullanılarak gerçekleştirilen görüntü geliştirme algoritmasının uygulanması için tasarım akışı gösterilmektedir.

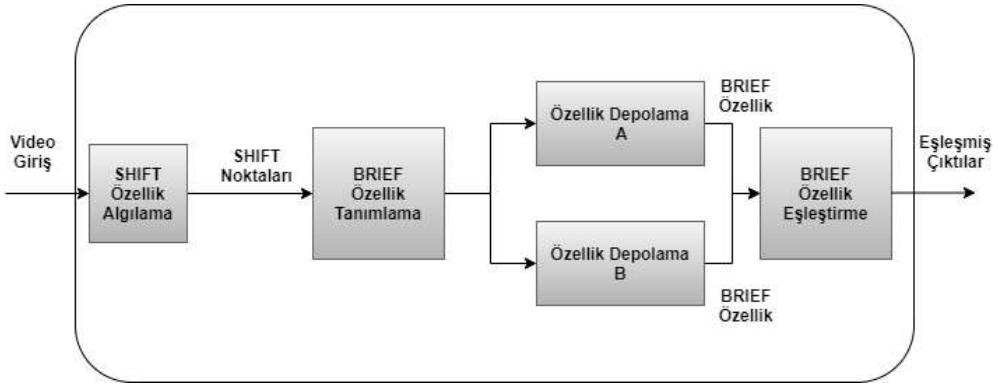


Şekil 1.4. PSoC 5LP kaynak kullanımı

Şekil 1.4'te PSoC 5LP platformu kullanılarak gerçekleştirilen görüntü geliştirme uygulamalarında kullanılan kaynak miktarları gösterilmektedir. Bu çalışmada[12], görüntü işleme algoritmasının aritmetik ve mantıksal işlemleri, C dili ile CPU çekirdeği ve Verilog bileşenlerinin dijital bloklar arasında uygun bir şekilde kullanılarak PSoC 5LP platformunda çok etkili sonuçlar elde edilebileceğini göstermişlerdir.

Wang ve çalışma grubu tarafından gerçekleştirilen çalışmada[13], Özellik algılama ve eşleştirme için yeni bir FPGA tabanlı gömülü sistem mimarisi önererek ölçekle değişmeyen özellik dönüşümü (SIFT:Scale-Invariant Feature Transform) ile özellik algılamanın yanı sıra ikili sağlam bağımsız temel özellikler (BRIEF:Binary Robust Independent Elementary Features) ile özellik açıklaması ve eşleştirmeden oluşan yöntemleri kullanmışlardır. Önerilen bu sistem ile video için

özellik algılama ve eşleştirme sağlamaktadır. Bu sistem, algılanan SIFT özellikleri için BRIEF tanımlayıcısını iki görüntü için eşleşen BRIEF özellikleri kullanarak tanıma işlemi FPGA platformunda gerçekleştirilmiştir.



Şekil 1.5. Önerilen tasarım[13]

Şekil 1.5'te Wang ve diğ. tarafından önerilen sistemin tasarımı verilmektedir. Giriş videosu 1280x720 piksel ve 60 frame/saniye SHIF özellik algılama bloğunda görüntü üzerinde bazı noktaları tespit edip işaretleme yapar ve BRIEF özellik tanımlama bölümü, tespit edilen her SIFT anahtar noktası için BRIEF tanımlayıcıları çıkarır. Noktaların tespit edilmesi giriş görüntüsü ile gauss filtrelerinin evrişim işleminin sonucunda elde edilen noktalardır. Daha sonra bu özellikler depolama bloğunda tutulur. Son olarak, BRIEF özellik eşleştirme bölümü, eşleşen çiftleri bulmak için özellik deposu A ve B'den (sırasıyla iki görüntü için) BRIEF özellik tanımlayıcılarını okur ve eşleştirme sonuçları elde edilir.

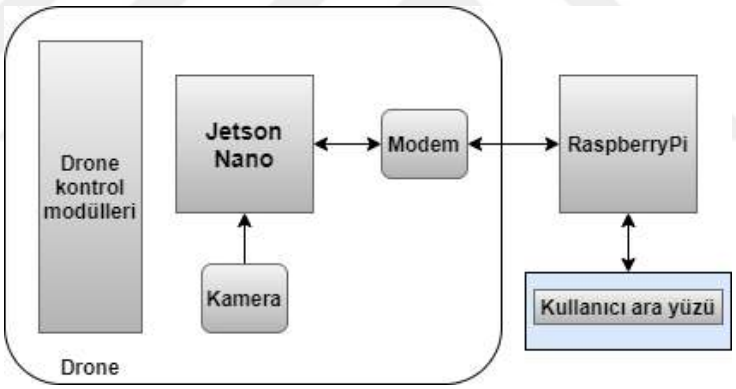
Tablo 1.6. FPGA kaynak kullanımı sonuçları[13]

Kaynaklar	Kaynak kullanımı
LUT	18.437 (%26)
FF	13.007 (%18)
DSP48E	52 (%81)
RAM (kbit)	4.932 (%92)
Frekans	159.16 MHz

Tablo 1.6'da Xilinx XC5VLX110T FPGA gömülü sistem platformunda Verilog yazılım dili kullanılarak gerçekleştirilen sistemin kaynak kullanımı sonuçları verilmektedir. Hafızada, videodan elde edilen iki ayrı görüntü saklandığı için RAM kullanımı ve SHIFT özellik algılama bloğunda gerçekleştirilen evrişim işlemlerinden kaynaklı DSP48E kullanm yüzdeleri yüksek çıkmıştır.

Alareqi tarafından yapılan çalışmada[14], sayısal görüntü işleme tekniklerini xilinx sistem generatörü (FPGA kullanarak) ve matlab simulink programını beraber çalıştırarak görüntü işleme uygulamaları gerçekleştirmiştir. Bu çalışmada görüntü işleme ile ilgili pek çok temel teknik (parlaklık kontrol, çözünürlük ayarlama, görüntü dönüşümleri gibi) açıklanmaktadır. Bu çalışmadaki amaç, insan elinde damarların XSG ve MATLAB benzetim blokları ile oluşturulan sistem ile tespit işleminin gerçekleştirilmesidir. Orijinal görüntüye sırasıyla, XOR, NOT, Addsub, parlaklık kontrol ve görüntü kalitesinin arttırılması işlemleri uygulanmıştır. Sonuç olarak bu teknik kullanılarak, damar saptanabilir, daha net görüntülenebilir ve daha fazla analiz kolaylığı sağlamaktadır.

Gur ve çalışma grubu tarafından gerçekleştirilen çalışmada[15], drone üzerine entegre edilmiş Jetson nano gömülü sistem platformu ve kamera aracılığıyla insanlar tarafından ulaşılamayacak bölgelerde araç kazaları tespit edilmiştir. Bu çalışmada, SSD MobileNet modeli kullanılarak tanıma ve sınıflandırma işlemleri gerçekleştirilmiştir. Önerilen bu sistemde, olay yeri, otonom dronlar, derin öğrenme ve iletişim ara yüzüne göre nesnelerin sınıflandırılması ve tespit edilmesinden oluşmaktadır. Otonom drone kontrolü ve derin öğrenme tabanlı görüntü işleme için NVIDIA Jetson Nano ana kart ve bilgisayar ile iletişim için Raspberry Pi kullanılmaktadır.



Şekil 1.6. Dron kontrol[15]

Şekil 1.6'te önerilen otonom dron kontrol yapısı verilmektedir. Dron'nun tüm yönetimi NVIDIA Jetson Nano kartı ile yapılmaktadır. Taşınabilir modem sayesinde dron ve kullanıcının haberleşmesi Raspberry Pi kullanılarak gerçekleştirilmektedir. Bu sistemde, kamera aracılığı ile Jetson Nano platformunda yakalanan görüntüler, MobileNet modeli üzerinden işlendikten sonra, nesne sınıflandırma ve tespit sonuçları kullanıcı ara yüzüne aktarılır. MobileNet modeli düşük gecikmeli sistemlerde, kolayca eşleştirilebilir mobil ve yerleşik görüntü uygulamaları için tasarlanmıştır[16].

İlk olarak Warren S. McCulloch ve Walter Pitts 1943 yılında[17], Yapay Sinir Ağı (YSA) matematiksel modelini, nöronların nasıl çalışabileceğini elektrik devreleriyle basit bir sinir ağı modeli oluşturmuşlardır. 1956 Dartmouth Yapay Zeka Yaz Araştırma Projesi[18] ile düşünebilen

makineler başlığı altında yapay zekanın kurucuları tarafından yaz çalıştayı gerçekleştirilmiştir. Bu sürecin sonuçlarından biri, hem endüstride yapay zeka hem de beynin çok daha düşük seviyeli sinir işleme kısımlarında araştırmayı teşvik etmeyi amaçlamışlardır.

Yann leCun ve çalışma grubu tarafından[19], geriye yayılım algoritması ile verilerin güncellenebilmesi makinelerin insansı hale gelmesini öne sürmüştür. Daha sonra Yann leCun tarafından 1998 yılında evrişimsel sinir ağını ilk defa kullanarak rakamların sınıflandırma işlemi gerçekleştirildi bu çalışma bilgisayarlı görünün temelini oluşturmaktadır[20].

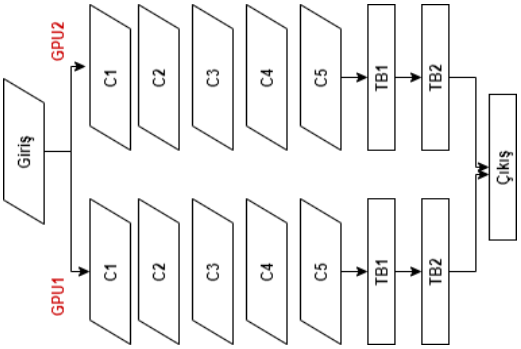
Tablo 1.7. LeNET-5 CNN model[20]

Katmanlar	Katman Konfigürasyonları			
Giriş	Görüntü		32	32
C1	Evrişim	5x5x6	28	28
S2	Özellik azaltma	2x2x6	14	14
C3	Evrişim	5x5x16	10	10
S4	Özellik azaltma	2x2x16	5	5
C5	Evrişim	5x5x120	1	1
F6	Tam bağlantı		1	84
Çıkış	Tam bağlantı		1	10

Tablo 1.7'de LeNET-5 CNN modeline ait yapı gösterilmektedir. Bu modelde eğitim on tane sınıfı bulunan 32x32 piksel boyutlarında gri tonlamalı resimlerden oluşan MNIST veri seti üzerinde % 0,9 hata ile sınıflandırma işlemi gerçekleştirilmiştir.

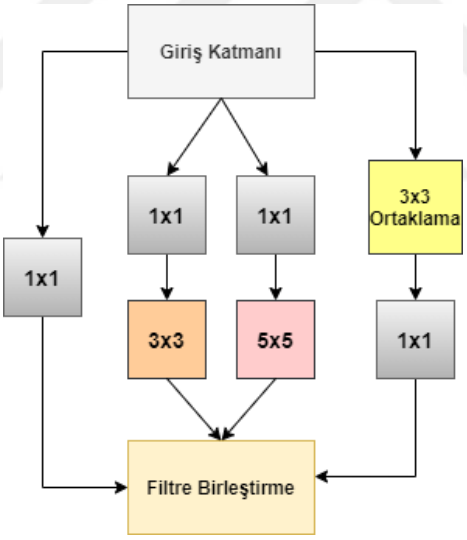
Deng ve çalışma grubu tarafından[21] ImageNet görüntü veri seti, yaklaşık olarak 22.000 farklı kategoride görüntülerden oluşturulmuştur. Bu görüntü kümesinde 15 milyondan fazla etiketli yüksek çözünürlüklü görüntü bulunmaktadır. Pascal Görsel Nesnesinin (Pascal Visual Object Challenge) bir uzantısı olarak 2010 yılından itibaren gerçekleştirilen yıllık bir yarışma olan ImageNet Büyük Ölçekli Görsel Tanıma Yarışmasında (ILSVRC: ImageNet Large-Scale Visual Recognition Challenge) bu veri seti kullanılmaktadır. Bu yarışmada, 1000 kategori ve her birinde yaklaşık 1000 görüntü içeren bir ImageNet alt veri kümesi kullanılmaktadır. Toplamda 1,2 milyon eğitim görüntüsü, 50 bin doğrulama görüntüsü ve 150 bin test görüntüsünden oluşmaktadır.

Krizhevsky ve çalışma grubu[22] tarafından 2012 yılında ILSVRC yarışmasında ImageNet veri setini, önerdikleri AlexNet CNN modeli ile birinci olmuşlardır. 650,000 nöronlu ve toplamda 3,2 milyon görüntü kullanarak çıkış katmanında 1000 sınıfın olduğu, beş adet evrişim ve üç adet tam bağlantı katmanı kullanılarak model geliştirmişlerdir. Modelin eğitimi için 2 adet GTX 580 3GB GPU grafik kartı ile CUDA üzerinde eş zamanlı kullanılarak parametrelerin eğitimleri gerçekleştirilmiştir.



Şekil 1.7. AlexNet model yapısı[21]

Şekil 1.7'de AlexNeT model yapısı gösterilmektedir. Aktivasyon fonksiyonu çıkış katmanında softmax, diğer katmanlarda ise ReLU kullanılmıştır. İlk katmanda evrişim işlemi 11x11 ve diğer evrişim katmanlarında 3x3'lük filtre matrisi kullanılarak gerçekleştirilmiştir[22]. ILSVRC yarışmasını 2014 yılında GoogleNet kazanmıştır[23,24]. Bu model daha önce oluşturulan modellere göre derin ve daha az parametre kullanarak Çince el yazısını okumak için gerçekleştirmiştir.



Şekil 1.8. Başlangıç modülü[24]

Bu modelde 22 katman bulunmakta ve bu modelde ara katmanlarda, farklı boyutlara sahip filtreler kullanılarak diğer modellerden farklı bir model geliştirmişlerdir. **Şekil 1.8**'de bu yapıya ait blok şema gösterilmektedir. Bir önceki özellik katmanı ve farklı boyutlarda filtre matrisi ile evrişim ve ortaklama işlemlerinden sonra bunlara ait sonuçların birleştirilmesi ile bir sonraki tabakada kullanılacak özellik matrisi elde edilmektedir.

Geleneksel CNN modellerinin (AlexNet, ResNet vb.) veya özel tasarımların FPGA platformlarında tasarlanması kaynak kullanımından dolayı gerçekleştirilememektedir. Sun ve çalışma grubu[25] Bu sorunun üstesinden gelmek için, her biri ağ modelindeki bir katmanın

hesaplanmasından sorumlu olan, çoklu veri işleme elemanından (VİE) oluşan FPGA tabanlı bir hızlandırıcı önermektedir. Bu önermede;

- Tüm VİE'ler FPGA çipi üzerinde eşleştirilir, böylece farklı katmanlar eşzamanlı olarak ardışık düzenlenmiş tarzda çalıştırılabilmektedir.
- Hızlandırıcının verimini arttırmak için Tam bağlantı katmanlarında, ağırlıkların sayısını azaltmak için bir budama yöntemi kullanılmasını önermektedir [26]. Bu tekniği kullanarak tam bağlantı katmanındaki parametrelerin sayısı 59M'den 6M'ye düşürülmüştür.

Bu çalışmada[25], AlexNet derin öğrenme modelini Xilinx Virtex-7 FPGA VC707 platformunda kullanmışlardır. Bu tasarım Vivado HLS (v2016.3) ile uygulanmıştır. HLS, C dilini donanım tanımlama diline (HDL) çevirmeyi ve tasarımı Vivado programına IP çekirdeği olarak dışa aktarmayı sağlamaktadır.

Tablo 1.8. FPGA Kaynak kullanımı[25]

Kaynaklar	Kaynak kullanımı
LUT	287396(%94,66)
FF	493210 (%81,22)
DSP	2387 (%85,25)
RAM	1806 (%87,67)

Bu modele[25] ait FPGA kaynak kullanımı **Tablo 1.8**'de gösterilmektedir ve FPGA'ya ait donanım kaynaklarının neredeyse tamamı kullanılmıştır.

CPU'lar sıralı işlemler için tasarlanmış bir/birçok çekirdekten oluşmaktadır. Datta ve çalışma grubu tarafından gerçekleştirilen çalışmada[26] paralelleştirilmiş CPU'ların performansını karşılaştırmayı amaçlamaktadırlar. Önerdikleri sistemi karşılaştırmak için üç farklı yapay sinir ağı (AlexNet, VGG16 ve kendi önerdikleri model) kullanarak görüntü sınıflandırma işlemi gerçekleştirmişlerdir. Aynı anda bir çekirdek, iki çekirdek, üç çekirdek ve dört çekirdek CPU kullanarak CNN modellerin doğruluk oranlarını ve işlem sürelerini karşılaştırmışlardır.

Tablo 1.9. Modellerin karşılaştırmalı sonuçları[26]

Çekirdek Sayısı	AlexNet		VGG16		Önerilen sistem	
	Doğruluk (%)	İşlem süresi(sn)	Doğruluk (%)	İşlem süresi(sn)	Doğruluk (%)	İşlem süresi(sn)
1	%87.38	519.45	%87.27	5830.96	%90.14	1215.23
2	%87.48	498.45	%87.32	5742.08	%91.6	1176.07
3	%87.68	485.25	%87.57	5647.21	%88.47	1162.06
4	%87.56	469.51	%87.38	5622.79	%95.44	1150.76

Yüksek boyutlu veri setleri kullanırken paralel olarak çalışabilen CPU çekirdeklerinin kullanımının daha iyi bir yol olduğunu göstermişlerdir. **Tablo 1.9**'da bu çalışmada kullanılan CNN modellerinin MNIST veri setine uygulanarak sınıflandırma sonuçlarının karşılaştırılması verilmektedir. Kullanılan CPU çekirdeklerinin paralel çalıştırılması her üç modelde de işlem süresini azaltmıştır. Görüntü işlemeyi hızlandırmanın daha iyi yolu, birden çok CPU çekirdeğini paralel hale getirmek olduğunu göstermişlerdir.

Manikandan ve çalışma grubu[27] evrişimsel sinir ağları (CNN: Convolutional Neural Network) gibi çok fazla işlem yükü olan ve donanımsal tarımlarda hafızada geniş yer tutan algoritmaların veri işleme hızlarını arttırmak için çalışmalar gerçekleştirilen çalışmaları derlemişlerdir. Yaklaşık Hesaplama (YH) donanım hızlandırma ile birlikte kullanılan CNN modeller, donanım uygulamalarında sistem performansı (hız, alan ve güç) ve verimliliğini arttırmak için kullanılmaktadır. CNN yapısında kullanılan YH tekniği, ağırlık budama ve ağırlık paylaşımı ile gerçekleştirilmektedir. Sinir ağı yapısındaki önemsiz ağırlıklar kaldırılarak ağ yoğunluğu azaltılır. Böylelikle, enerji ve bellek kullanımı açısından performans artırılabilmektedir. Budama tekniği, özellik çıkartma matrislerinde, rastgele bazı ağırlıklarda, sıralı ağırlıklarda ve bazı filtre ağırlıklarının tamamında yapılabilmektedir.

Wang ve çalışma grubu tarafından gerçekleştirilen çalışmada[28], son yıllarda kullanılan ve geliştirilen derin öğrenme modellerinin daha önce kullanılan modellere göre üstünlüklerini kıyaslamıştır. Veri setlerinin büyük olması, karmaşık analizler, görüntü boyutlarının büyük ve istenmeyen özellikleri (gürültü vb.) içermesi gibi etkenlerin görüntü sınıflandırmadaki önemi üzerine gerçekleştirilen çalışmaları karşılaştırmışlardır. Kiranyaz ve çalışma grubu tarafından gerçekleştirilen çalışmada[29], bir boyutlu ve iki boyutlu evrişimsel sinir ağları (CNN) kullanılarak gerçekleştirilen çalışmaları karşılaştırmıştır. Aynı şartlar altında bir boyutlu CNN yapısının hesaplama karmaşıklığının ve kullanılan parametrelerin sayısı daha az olduğunu göstermişlerdir. İki boyutlu derin CNN yapılarının eğitim aşamaları için GPU yapısına ihtiyaç duyulduğunu ve gerçek zamanlı uygulamalarda bir boyutlu yapıların kullanımının daha uygun olduğunu göstermişlerdir.

Song ve çalışma grubu tarafından gerçekleştirilen araştırmada[30], yüksek çözünürlüklü bilgisayarlı tomografi görüntülerinde gabor filtresi ve HOG dönüşümü algoritmalarından esinlenerek görüntüde özellik tanıma için iki farklı model oluşturmuşlardır. Bu modelleri kullanarak akciğer dokusunun beş kategorisi için yeni bir sınıflandırma yöntemi üzerine çalışmalar gerçekleştirmişlerdir. Önerilen bu çalışma üç aşamada gerçekleştirilmektedir

- İlk olarak, her bir görüntü bölgesi/yama (tümörlü bölge vb.)için bir dizi doku, yoğunluk ve gradyan (T-I-G) özelliklerinin çıkartıldıktan sonra iki adet özellik tanımlayıcısı belirlenir.

1. Çok ölçekli (Multi-Scale) Gabor filtrelerini ve LBP histogramlarını entegre eden görüntüde zengin doku özelliklerini çıkartmak için yeni bir Gabor-LBP (RGLBP) özellik tanımlayıcısı

2. Gradyan özelliklerini çıkarmak için çok koordinatlı (Multi-Coordinate) HOG (MCHOG) tanımlayıcısı

- Daha sonra her bir görüntü bölgesi, yeni bir bölge uyarlamalı seyrek yaklaşım (PASA) algoritmasına göre sınıflandırılır/işaretlenir.

1. Görüntü yaması etiketlenmesi işlemi istatistiksel yaklaşımlar kullanılarak seyrek olarak işaretlenen bölgelerin etkileri geliştirilir.

2. Ayırt ediciliği arttırmak için, referans noktaların özellik değerlerini değiştirerek ikili özellik mesafelerine dayalı, yamaya özgü bir uyarlama yöntemi ile tasarlamışlardır

- Son olarak, odaklanılan bölgede olasılık tahminine dayalı açıklamalı etiketleme "Açıklamalı ROI'nin (AROI)" işlemi gerçekleştirilmiş olur.

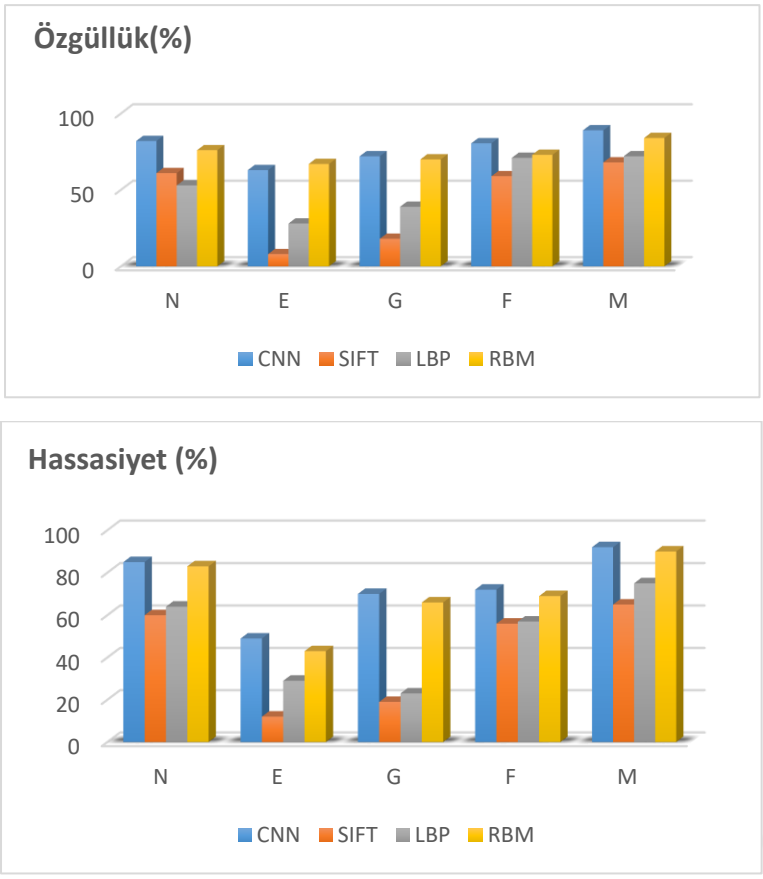
Bu çalışmada[30], önerilen özellik tanımlayıcılar (RGLBP ve MCHOG) ve yaklaşık görüntü sınıflandırma algoritması (PASA), problem alanıyla ilgili varsayımlara dayalı olarak tasarlandığından, diğer medikal uygulamalar için de gerçekleştirilebileceğini öne sürmektedirler.

Tablo 1.10. Sınıflandırma sonuçları[30]

Beklenen Değer	Hesaplanan Değer (%)				
	T_N	T_E	T_G	T_F	T_M
T_N	87.6	3.9	1.9	0.4	6.1
T_E	12.4	80.6	0.4	4.9	1.7
T_G	3.5	0.2	82.7	10.5	3.1
T_F	3.3	4.0	9.3	81.2	2.2
T_M	12.3	0.5	4.3	1.8	81.2

Tablo 1.10'da İnterstisyel Akciğer Hastalığı (ILD) teşhisi için önemli olan beş akciğer dokusu kategorisi "normal", "emphysema", "ground glass", "fibroz" ve "mikronodüller" yapılarının önerilen sisteme göre sınıflandırma sonuçları verilmektedir. Bu doku kategorilerinin f1-skor değerleri yüzde olarak sırası ile 56.2, 79.8, 73.1, 82.2 ve 83.4'tür. Li ve çalışma grubu[31], medikal görüntü işleme uygulamalarında sıklıkla kullanılan akciğer üzerinde leke tayini (beş akciğer dokusu) için CNN sınıflandırma modeli oluşturmuşlardır. Bu yöntem kullanılarak hastalık tespiti gerçekleştirmeyi amaçlamışlardır. CNN model eğitim sürecinde, eğitim için kullanılacak görüntü verileri eşit olarak 10 gruba ayırmışlardır. Her test aşamasında, bir gruptan örnekler test verisi olarak seçilir ve kalan 9 gruptan diğer tüm örnekler ise o modelde eğitim için kullanılır. Böylece 10 adet model eğitim seti oluşturup bu modellerin birleşimi ile kendi CNN eğitim

modellerini oluşturmuşlardır. Bu yaklaşımı kullanarak aşırı öğrenme (overfit) olayını engellemeye çalışmışlardır.



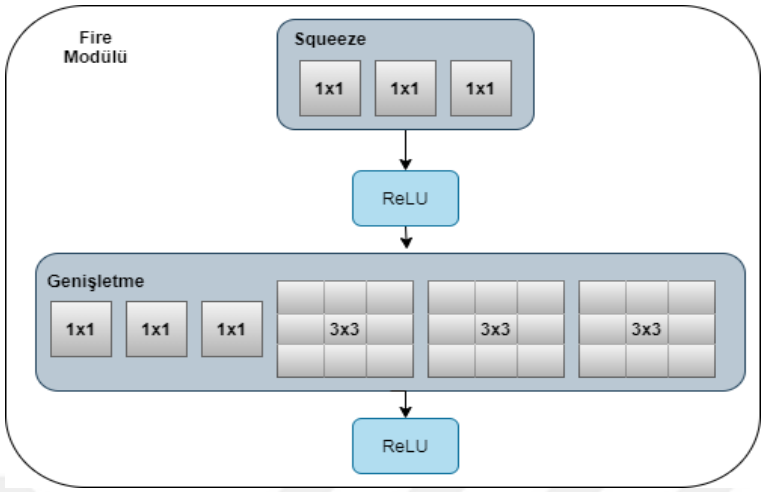
Şekil 1.9. Model sonuçları[31]

Şekil 1.9'da bu çalışmaya[31] ait beş akciğer dokusunun CNN model kullanılarak sınıflandırılması gösterilmektedir. Bu sınıflandırma sonucunu Hassasiyet (Precision) ve Özgüllük (Recall) ile göstermişlerdir.

Hosseini-Asl ve çalışma grubu tarafından gerçekleştirilen çalışmada[32], üç boyutlu CNN sınıflandırma yapısını beyin Manyetik Rezonans (MR) görüntüleme cihazı ile elde edilen görüntüler üzerinden Alzheimer hastalığının teşhisi için kullanmışlardır. Beyin MRI'larından elde edilen özellik vektörleri çok gürültülü oldukları için pürüzsüzleştirme (smooting) ve kümelemeden sonra sınıflandırma için kullanılabilir ve çıkarılan özellik vektörlerinin uygunluğu, büyük ölçüde kayıt hataları ve gürültü nedeniyle görüntü ön işlemesine bağlıdır. Bu etkilerin ortadan kaldırılması ve özelliklerin çıkarımı için yeni bir derin üç boyutlu evrişimli sinir ağı (3D-CNN) önermektedirler.

Zhang ve çalışma grubu tarafından gerçekleştirilen çalışmada[33], mide kanseri için erken teşhisi sağlamak için CNN yapısını kullanarak Gastric Precancerous Disease Network (GPDNet) modeli ile üç kategoride (polip, erozyon ve ülser) sınıflandırma işlemini gerçekleştirmişlerdir.

SqueezeNet modelinde kullanılan fire-modülleri bu modelde kullanılarak, sınıflandırma hızı artırırmış, model boyutunu ve kullanılan parametre sayısını yaklaşık 10 kat azaltmışlardır.



Şekil 1.10. Fire modülü[34]

GPDNet modelinde ortaklama katmanlarından sonra **Şekil 1.10**'da gösterilen Fire-modül yapısı kullanılmaktadır. Çıkış katmanında Softmax ve diğer katmanlarda ReLU aktivasyon fonksiyonu ile bu model küçük kümeler (mini-batch) stokastik eğim düşümü kullanılarak sınıflandırma işlemi gerçekleştirilmiştir.

Tablo 1.11. Model karşılaştırma sonuçları[34]

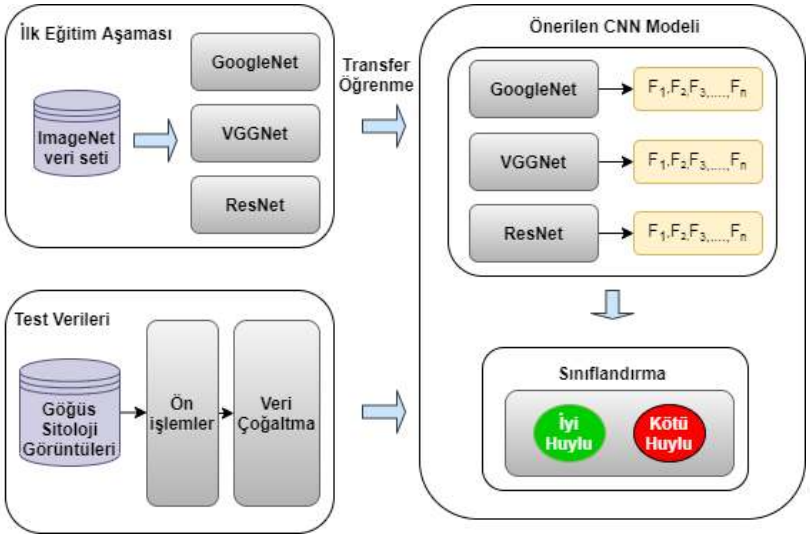
	GPDNet	GPDNeT+Fire
Fire-modül	0	2
Tam bağlantı katmanı	Var	Yok
Kullanılan parametre	144928	14368
Hafıza(Coffe model)	568kb	60kb
Doğruluk	%87.39	%79.74
Doğruluk (+reinforced)	%88.47	%88.47

Tablo 1.11'de GPDNet modeli ve fire modülü kullanımı ile elde edilen sonuçlar gösterilmektedir. GPDNet modeli eğitim süreci tamamlandıktan sonra, döngüsel reinforced öğrenme modeli parametreleri ile ekstra 6 döngü daha gerçekleştirilerek sonuçta tüm modelin doğrululuk oranı %88,9 olmuştur.

Esteva ve çalışma grubu tarafından gerçekleştirilen çalışmada[35], GoogleNet CNN modelini kullanarak 757 farklı deri kanseri türünün sınıflandırılmasını gerçekleştirmişlerdir. Bu çalışmada 21 farklı dermatolojistten uzman görüşü alınarak oluşturulan CNN modeli karşılaştırılmıştır. Bu modelde imageNet kullanılarak transfer öğrenme modeli Google Tensorflow ağını kullanmışlardır. Model doğruluk oranı % 93,33 olarak bulunmuştur. Bu çalışmada, Google'ın

Inception v3 CNN mimarisini, ImageNet veri setini 1000 nesne sınıfında (1.28 milyon görüntü) % 93.33 doğrulukta önceden eğitilmiş olarak kullandıktan sonra, son sınıflandırma katmanını ağdan kaldırıp ve tüm katmanlardaki parametreleri bu çalışmadaki veri setini kullanarak model eğitimini gerçekleştirmişlerdir. Bu model eğitim stratejisi, transfer öğrenimi olarak da bilinmektedir.

Khan ve çalışma grubu tarafından gerçekleştirilen çalışmada[36], GoogleNet, ResNet ve VGGNet modellerinin parametreleri kullanılarak transfer öğrenme modelini gerçekleştirmişlerdir. Bu çalışma da meme sitolojisi görüntülerini kullanarak göğüs kanseri teşhisi ve kanser evresi sınıflandırma işlemi gerçekleştirmişlerdir. Kullanılan bu hibrit model ile %97,52 doğruluk oranında sınıflandırma işlemi gerçekleştirmişlerdir.



Şekil 1.11. Önerilen CNN modeli[36]

Önerilen Transfer öğrenme CNN[36] modeli **Şekil 1.11**'de Gösterilmektedir. Genel olarak kullanılan GoogleNet, VGGNet ve ResNet modelleri ile CNN yapısında özellik çıkartma matrislerini oluşturmak için kullanılmıştır. Önerilen modelin güvenirliğini kanıtlamak için, veri seti sırasıyla GoogleNet, VGGNet ve ResNet olmak üzere üç farklı CNN mimarisi üzerinde tek tek eğitilmiştir.

Tablo 1.12. Model sonuçlarının karşılaştırılması[36]

	GoogleNet	**VGGNet**	**ResNet**	**Önerilen**
Doğruluk(%)	93.5	94.15	94.35	97.52

Tablo 1.12'de gösterildiği gibi önerilen CNN modeli gerçekleştirilen sınıflandırmada, % 97,52 doğruluk oranı ile en yüksek sonuç elde edilmiştir. Bu çalışmanın sonuçları, önerilen sistemin, diğer üç mimari ile karşılaştırıldığında meme kanseri tümörünün saptanması ve sınıflandırılmasında doğruluk açısından yüksek performansa ulaştığını göstermektedir.

Shanthi ve Sabeenian tarafından gerçekleştirilen çalışmada[37], Diyabetik retinopati görüntülerini CNN kullanarak sınıflandırma işlemi gerçekleştirmişlerdir. Bu çalışmada önerilen

model için, başlangıçta üç boyutlu renkli (RGB) görüntüler ayrıştırılır ve yeşil (G) boyuttaki pikseller kullanılarak model eğitimini Alexnet CNN modeli ile gerçekleştirilmiştir.

Tablo 1.13. Model sonuçlarının karşılaştırılması[37]

Kategoriler	Sağlıklı Retina	DR Seviye-1	DR Seviye-2	DR Seviye-3	Doğruluk (%)
Sağlıklı Retina	102	1	2	2	96.6
DR Seviye-1	3	51	2	2	88.0
DR Seviye-2	1	3	78	4	90.1
DR Seviye-3	1	0	1	51	96.0

Tablo 1.13'te Göz retinasında glikoz oranına bağlı olarak oluşan hastalık seviyelerine ait toplam 303 adet Messidor veri seti ile AlexNet CNN yapısı kullanılarak elde edilen sınıflandırma sonuçları verilmektedir.

Lu ve çalışma grubu tarafından gerçekleştirilen çalışmada[38], MR görüntülerini AlexNet ve transfer öğrenme metodunu kullanarak beyinde meydana gelen hasar ve hastalık teşhisini sınıflandırma işlemini gerçekleştirmişlerdir. Bu çalışmada kullanılan beyin görüntüsü veri seti derin bir ağını eğitmek için uygun değildir. Bu sorunu çözmek için transfer öğrenimi kullanılmışlardır. AlexNet modelinin son üç katmanını kendi tasarımladıkları katman ile değiştirmişlerdir. Orijinal modelin geri kalan parametrelerini değiştirmemiştirler ve özellik çıkartma katmanlarında bulunan parametrelerin başlangıç değeri olarak kullanılmıştır. Transfer öğrenimi, yeterli etiketli örnek olmadığında derin sinir ağını eğitmek için çok uygun ve etkili bir yöntem olduğunu göstermişlerdir. Önceden farklı veri setleri ile eğitilmiş ağdaki tüm parametreleri başlangıç koşulu olarak kullanmak, büyük görüntülerden öğrenilen özelliklerden faydalanılmasını sağlamaktadır.

Tablo 1.14. Değiştirilen katman sayısı ve doğruluk oranları[38]

Katman Sayısı	Eğitim süresi	Doğruluk (%)
3	2 dk. 17s	100
6	3 dk. 29s	100
9	3 dk. 45s	60.92

Tablo 1.14'te aynı eğitim veri seti kullanılarak, transfer öğrenmede daha fazla katman değiştirildiğinde doğruluğun azalabileceği ve eğitim süresinin arttığını göstermişlerdir. Bundan dolayı, bu çalışmada son üç katman değiştirilerek gerçekleştirilmiştir. Derin ağları eğitmek için yüksek performanslı GPU ve CPU yapılarına sahip bilgisayarlar gerektirmektedir, ancak transfer

öğrenme ile sıradan kişisel bilgisayarlarda da uygulanabilmektedir. Önerilen bu sistemin eğitimi, i7- 4770CPU ve NVIDIA Quadro K2000 GPU ile MATLAB R2018a'da uygulanmıştır. Bu yapıyı sınıflandırma hatasını azaltmak için kullanmışlardır ve %100 başarı elde edilmiştir.

Hammad ve çalışma grubu tarafından gerçekleştirilen çalışmada[39], elektrokardiyogram (ECG) sinyallerinden elde edilen kalp atışı grafiği (paterni) gibi biyometrik verilere dayalı yeni kimlik doğrulama işlemini CNN ve ResNet modellerini kullanarak gerçekleştirmişlerdir. Bu çalışmada, önerilen CNN modeli beş farklı model arasından karşılaştıralı sonuçlarına bakılarak seçilmiştir. Kullanılan modeller, 3.30 GHz CPU, 32GB RAM ve GPU NVIDIA GeForce GTX 1080 özelliklerine sahip iş istasyonunda eğitildi ve tüm algoritmalar MATLAB R2017a yazılımı ile açık kaynaklı derin öğrenme araç kutusu (toolbox) kullanılarak öğrenme süreci gerçekleştirilmiştir.

Tablo 1.15.Önerilen CNN modellerinin sonuçları[39]

Veri seti + doğruluk oranı	Öneri-1	Öneri-2	Öneri-3	Öneri-4	Öneri-5
PTB	96.9	90.9	94.3	97.9	98.5
CYBHI	94.7	90.2	96.3	98.8	99.7

Tablo 1.15'te İki farklı veri seti (PTB ve CYBHI) kullanarak beş farklı CNN modelinin sınıflandırma doğruluk oranları verilmiştir. Öneri-5 ile oluşturulan CNN modelinde bir boyutlu evrişim işlemi kullanılmaktadır.

En iyi sınıflandırma sonucu veren Öneri-5 şu katmanlardan oluşmaktadır;

1. Evrişim Katmanı-1
2. Evrişim Katmanı-2
3. Maksimum Ortaklama
4. Seyreltme (Drop)
5. Evrişim Katmanı-3
6. Evrişim Katmanı-4
7. Maksimum Ortaklama
8. Seyreltme (Drop)
9. Tam Bağlantı + Seyreltme (Drop)
10. Tam Bağlantı + Seyreltme (Drop)
11. SoftMax

Öztürk ve çalışma grubu[40], akciğer X-ışını görüntülerini kullanarak COVID-19 vakalarının erken tespiti için bir DarkCovidNet modelini önermektedirler. Bu çalışmada DarkCovidNet modelini, çoklu ve ikili sınıflandırma ile X-ışını görüntülerini kullanarak COVID-

19'u sınıflandırmışlardır. Önerilen modelin performansını kanıtlamak için, ikili ve çoklu sınıflandırma problemini 5 katlı çapraz doğrulama (Cross-Validation) prosedürü kullanılarak gerçekleştirmişlerdir. Eğitim veri seti beş kısma ayrılır ve her defasında farklı bir kısım test için seçilerek geri kalan görüntüler eğitim için kullanılır ve böylelikle DarkCovidNet beş farklı görüntü ile sınıflandırma işlemini gerçekleştirmiştir.

Tablo 1.16.Önerilen sistemin sonuçları[40]

	Veri-1	**Veri -2**	**Veri -3**	**Veri -4**	**Veri -5**	**Ortalama**
Çoklu sınıflandırma (%)	89.33	84.89	85.78	87.11	88.0	87.02
İkili sınıflandırma (%)	100	97.6	96.8	97.6	97.6	98.08

Tablo 1.16'da çoklu ve ikili sınıflandırmaya ait DarkCovidNet modelinin doğruluk oranları verilmiştir. İkili sınıflandırmada daha iyi doğruluk oranına sahip sınıflandırmalar gerçekleştirmişlerdir. Bhattacharya ve çalışma grubu tarafından gerçekleştirilen derleme çalışmada[41], görüntü tabanlı tıbbi teşhis ve medikal teknolojilerde kullanılan tomografi, MRI ve X-ışını görüntüleri kullanılarak görüntü işleme teknikleri ve yapay zeka modeli ile tespit edilen Covid-19 virüsü çalışmalarını göstermişlerdir. Daha önce gerçekleştirilmiş çalışmalarda karşılaşılan zorluklar ve öneriler;

- Sınırlı sayıda COVID-19 X-ray görüntüsünün kullanılması
- Laboratuvar test yöntemleri farklı hastanelerde farklı sonuçların üretilmesi
- COVID-19'un erken teşhis/tahmin edilememesi
- Uzman ve model arasında tutarlı sonuçlar elde etmek
- Göğüs Tomografisi veri kümelerinden COVID-19'un hızlı ve güvenilir tespiti
- Koronavirüs tespiti ve hastalık yükünün ölçümü ve takibinde yüksek doğruluk elde etmek

zorluğu gibi vurguları bu yayında göstermektedirler.

Karim ve çalışma grubu tarafından gerçekleştirilen çalışmada[42], Gerçek zamanlı görüntü işleme tekniklerine dayalı olarak sokak suçlularını izlemek ve hedeflemek için drone uçağı geliştirmiştirler. Bu dron'da iki adet işlemci tasarlanmıştır. Birincisi görüntü işleme algoritmalarını kullanarak tespit etme ve ikincisi ise kontrol, izleme ve hedefleme için kullanılmıştır. Xu ve çalışma grubu tarafından gerçekleştirilen çalışmada[43], gerçek zamanlı suç davranışının (araba çalma, insanların yabancılar tarafından takip edilmesi, suikast, banka soygunu vb.) tespiti için uygulama geliştirmişlerdir. Bu çalışmada, veri toplama yoluyla birçok suç davranışı özelliğini elde ettikten sonra suç davranışı tanımanın doğruluğunu artırmak için, seçilen insan gruplarında suç davranışları için tanımlama yöntemi incelenmiş ve hızlı işlem yapabilen iş istasyonlarını kullanarak bireysel suçları ve bu bölgede bulunan kişileri ayırt etmek/tanımlamak için görüntü işleme teknolojisini kullanmışlardır. Kalabalığın davranışını analiz etmek için, elde edilen görüntü parçalara (blok)

bölünür ve görüntü alanındaki nüfus yoğunluğuna bağlı olarak bloğun boyutu belirlenir. Bu çalışmada, K-ortalamalı kümelemeye dayalı bir algoritma kullanılarak tespit edilecek kişilerin görüntü üzerinde bulunduğu alan ve görüntü parçalama için bloğun boyutu belirlenmektedir.

Şekil 1.12. Beklenmedik olay tepkisi[43]

Şekil 1.12'de beklenmedik olaylar karşısında verilebilecek tepkiler gösterilmektedir. Gerçek video görüntülerinde, kalabalık normal davranışlar sergileyerek gitmek istediği yöne doğru hareketini sürdürür veya sabit kalır. Ancak beklenmedik olaylar gerçekleştiğinde insanlar içgüdüsel olarak panikleyeceklerdir, bu nedenle normal olmayan davranışlar sergileyip olay yerinden uzaklaşmak için rastgele koşuşturacaklardır. Bu araştırma algoritmasının gerçek suç tespitindeki etkisini test etmek için, 60 set suç gözetleme videosu kullanılarak, suç eylemlerinin tanımlama işlemlerini gerçekleştirmişlerdir. Suç tanıma doğruluk oranı ortalama %87.78 olarak tespit etmişlerdir.

Asıl ve Nasibov tarafından gerçekleştirilen çalışmada[44], Jetson nano gömülü sistem platformu kullanılarak, kazara insanların vurulmasını önlemek için görüntü işleme ve yapay zeka tabanlı uygulama geliştirmişlerdir. Atış poligonunda insan ve hedefi ayırt edip, insana doğrultulan silahı servo motor ile güvenlik kademesine alan bir çalışma gerçekleştirmiştirler. Bu sistem, Python, Open CV kütüphanesi ve SSD inceptionV2 modeli kullanılarak oluşturulmuştur. SSD inception-V2, bilgisayar görüşündeki nesneleri algılamak için kullanılan derin sinir ağı modelidir. Bu model;

- Model giriş görüntüsüne göre bir nesnenin var olabileceği bölgeleri tespit eder
- Belirlenen bölgelerde buluna nesnelerin hangi sınıfta olduğunu tahmin eder.

Bu çalışmada[44], NVIDIA Jetson Xavier NX geliştirme kartı kullanılarak SSD inceptionV2 derin öğrenme algoritması ile gerçekleştirilmiştir. Kamera aracılığı ile görüntüler 20 FPS elde edilmekte ile 0.05 sn gecikmeli, servo motor 0.10 sn gecikmeli ve kamera 0.12 sn gecikmeli işlem gerçekleştirmekte ve toplamda kameranın gördüğü hedef ile kullanılan mekanizmanın devreye geçme süresi 0.27 saniye sistem gecikmesi ile gerçekleştirilmiştir.

Priya ve çalışma grubu tarafından gerçekleştirilen çalışmada[45], tekstil endüstrisinde görüntü işleme uygulamaları gerçekleştirmiştirler. Kumaş zerinde defolu kısımları belirlemek için morfolojik işlemler ve temel görüntü işleme algoritmalarını (renk piksellerine ayırmak, griye dönüştürmek ve morfolojik işlemler vb.) uygulamışlardır. Manuel olarak tespit edilen sistem ile karşılaştırdıklarında, görüntü işleme uygulamalarının daha iyi sonuçlar verdiğini gözlemlemişlerdir. Liu ve çalışma grubu[46], Çok ölçekli bir evrişimli sinir ağı modeli ile kumaş hatası algılama algoritması önermektedirler. Düşük sıralı ayrıştırma (LRD) tekniğini kullanarak bir görüntü matrisini iki bölüme ayırarak, düşük sıralı matris ve seyrek matris, burada düşük sıralı matris arka planı gösterir ve seyrek matris belirgin bölgeleri gösterir. Bu çalışmada, Derin öğrenme modeli kullanılarak görüntüde bulunan özellik matrislerini çıkarmak ve LRD modeli, arka plan ile kusur bölgesini ayırmak için kullanılmışlardır. Bu çalışma, 3.30 GHz, Intel (R) Core (TM) i3-2120 CPU PC'de MATLAB R2016a kullanılarak gerçekleştirilmiştir.

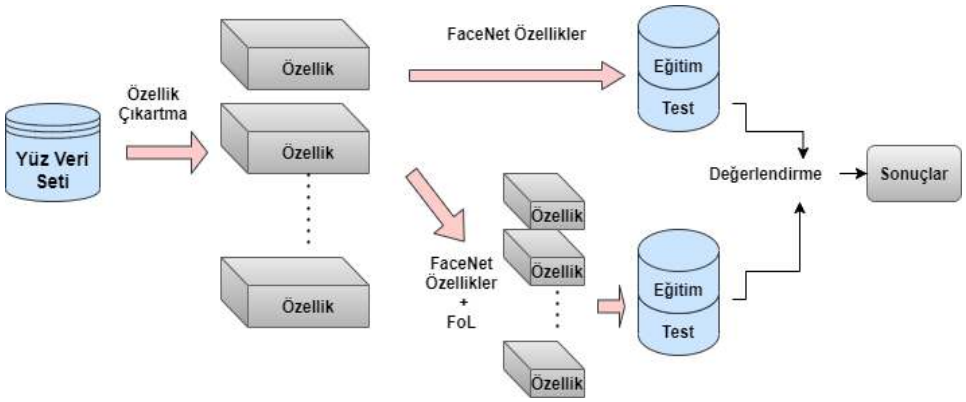
Revathi ve çalışma grubu tarafından gerçekleştirilen çalışmada[47], görüntü işleme algoritmalarını kullanarak tarım sektöründe uygulamalar geliştirmişlerdir. Bu çalışmada, bitki yaprağında bulunan hastalıkların teşhisi, Sobel ve Canny gibi kenar bulma algoritmaları ile hastalığın bulunduğu alanı, meyve ve sebzelerin boyut şekil analizi gibi işlemleri gerçekleştirmişlerdir.

- Hastalıklı yaprak, gövde, meyve tespiti
- Etkilenen alanı hastalığa göre ölçmek
- Etkilenen bölgenin sınırlarını bulmak
- Etkilenen bölgenin rengini belirlemek
- Meyvelerin boyut ve şeklini belirlemek
- Nesneyi doğru bir şekilde tanımlamak için görüntü analizi uygulamalarını gerçekleştirmişlerdir.

Naik ve Patel'in geçekleştirdiği çalışmada[48] farklı görüntü işleme YSA yapıları kullanarak tarım ürünlerinin sınıflandırılması ve büyüme kontrolünü gerçekleştiren çalışmalar hakkında detaylı bilgiler sunmaktadırlar. Bazı makine öğrenmesi, K-NN, SVM, YSA ve CNN gibi yaklaşımları kullanarak sınıflandırma işlemini meyvelerin renk, boyut, şekil, doku gibi dışsal özelliklerini kullanarak gerçekleştirilen çalışmalar hakkında karşılaştırmalı bilgiler verilmiştir. Zhu ve çalışma grubu tarafından gerçekleştirilen araştırmada[49], akıllı tarımda derin öğrenme tabanlı uygulamaların analizini gerçekleştirmişlerdir. Mahsulün verimliliğini artırmak, bitki hastalıklarını azaltmak ve tarımı veya tarımsal sanayiyi otomatikleştirmek için görüntü işleme ve gömülü sistemler üzerine uygulamalar gerçekleştirmişlerdir.

Peixoto ve çalışma grubu tarafından gerçekleştirilen çalışmada[50], Floor of Log (FoL) algoritmasını kullanarak yüz tanıma işlemi için KNN ve SVM sınıflandırma modelini kullanmışlardır. Bu yöntemin avantajı, modelin doğruluğunu koruyarak depolama ve enerjinin

azaltılmasıdır. Bu çalışmanın önerdiği yöntem (FoL), yüz tanıma uygulamaları için basit ama etkili bir özellik sıkıştırma algoritması sunmaktadır.



Şekil 1.13. Önerilen sistem yapısı[50]

Şekil 1.13'te yüz tanıma için önerilen sistem yapısı gösterilmektedir. Kullanılan bu yapıda, verilerin hafızada az yer kaplaması ve enerji tüketiminin az olmasından dolayı veri aktarımı için mümkün olan IoT cihazları arasında daha hızlı iletişim kurmayı sağlayabilmektedir. Çapraz doğrulama (Cross-validation) kullanılarak, FoL algoritmasının en iyi parametresini öğrenmek için K-En Yakın Komşular ve Destek Vektör Makinesi algoritması uygulanmıştırlar. Model eğitim ve test için, 16GB RAM ve NVIDIA GTX 1060 grafik kartına sahip 3.4 GHz AMD Ryzen 51.600 işlemciye sahip bilgisayar kullanılmıştır.

Tablo 1.17.Önerilen sistemin sonuçları[50]

	Model	Kullanılan yapı	Doğruluk (%)	Zaman (s)	Hafıza (kb)
Normal	SVM	HOG	97.37	0.095	536.9
		CNN	93.5	0.16	280.7
	K-NN	HOG	97.7	0.027	536.9
		CNN	94.75	0.046	280.7
+FoL	SVM	HOG	98.3	0.106	61.9
		CNN	96.75	0.15	32.5
	K-NN	HOG	97.0	0.03	61.9
		CNN	95.0	0.046	32.5

Tablo 1.17'de bu çalışmada[50], önerilen sisteme ait AR-scarf veri setine ait sonuçlar verilmektedir. Önerilen sistemin kullanılması ile modelin doğruluğunu koruyarak depolama/hafıza miktarı oldukça düşürülmüştür.

Nagare bu çalışmasında[51], araç plaka numaralarını belirlemek için iki farklı yapay sinir ağını kullanarak gerçekleştirmiştir. Bu çalışmada, görüntünün kameradan elde edilmesinden sonra

bazı ön işlemler (griye dönüştürme, morfolojik işlemler vb.) uygulandıktan sonra araç plaka bölgesi elde edilmektedir. Bu aşamadan sonra elde edilen plaka bölgesinde, aynı sütunda bulunan pikseller beyaz ise görüntü kesim işlemi gerçekleştirilir. Böylelikle içinde sadece bir adet plaka karakterlerinin bulunduğu resimler elde edilmiş olur. Bu aşamadan sonra, plaka karakterlerinin sınıflandırılması işlemi önerilen LVQ (Learning Vector Quantization) sinir ağı modeli ile gerçekleştirilmektedir.

Tablo 1.18.Önerilen Sistemin sonuçları[51]

Model	Numerik (%)	Alfabetik (%)	Doğruluk (%)
NN	70	62	66.67
LVQ NN	90	96.15	94.44

Tablo 1.18'de Geriye yayılımlı öğrenme modelinde % 66.67 ve bu çalışmada önerilen LVQ modelde ise % 94.44 doğruluk oranında plaka karakterlerinin tespiti gerçekleştirilmiştir.

Zhai ve çalışma grubu tarafından gerçekleştirilen çalışmada[52], FPGA gömülü sistem platformunu kullanarak görüntüde plaka bölgesinin tespiti gerçekleştirilmiştir. Bu çalışmada önerilen algoritma ilk olarak MATLAB ortamında gerçekleştirildikten sonra RC240 FPGA geliştirme kartında gerçekleştirilmiştir. İlk aşama olarak, üç boyutlu renkli görüntü, gri tonlamalı görüntüye dönüştürülmüştür, daha sonra bu görüntüye dikey 3x3'lük Sobel filtresi uygulanmıştır. Bu aşamadan sonra, 3x22'lik yapısal elemanlar ile morfolojik kapama ve ardından 3x5'lik yapısal eleman kullanılarak morfolojik açma işlemlerini uygulamışlardır.

Plaka bölgesi belirlerken üç adet ölçüt göz önünde bulundurulmuştur;

1. Plaka boyutları, genişlik (W) ve uzunluk (H) arasındaki oran
2. W ve H piksel boyutları aralığı
3. Alan

Tablo 1.19. Kullanılan kaynak miktarı[52]

	Kullanılan	Mevcut	Yüzde
Kullanılan dilimler	5.195	18.432	28
LUTs	7.168	36.854	19
BRAM	17	96	17

Tablo 1.19'da Araç plaka bölgesini tespit etmek için FPGA platformunda kullanılan kaynak miktarları verilmektedir. Bu çalışmada[52], %99.1 doğruluk oranında ve her bir plaka bölgesini ortalama 3.8 ms'de görüntüde tespit etmişlerdir.

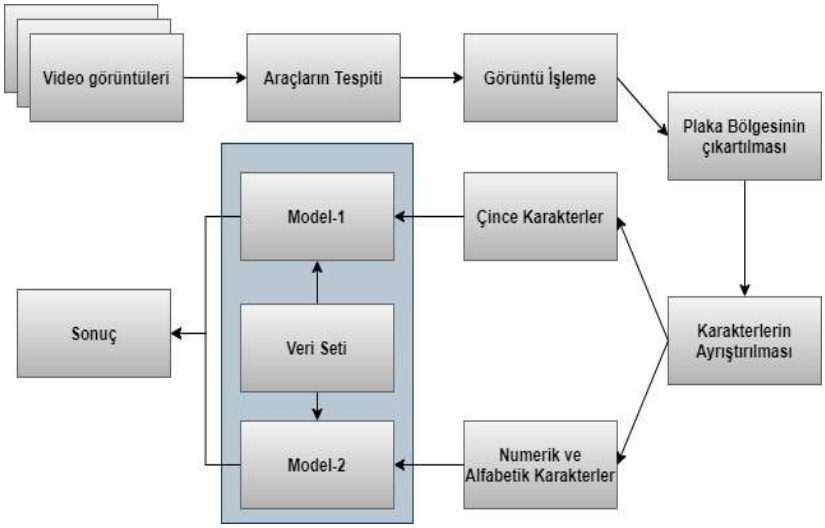
Indravadanbhai ve çalışma grubu tarafından gerçekleştirilen çalışmada[53], OCR metodu kullanarak plaka okuma uygulamaları gerçekleştirmiştir. Bu yöntemin en büyük dezavantajları karakterlerin birbirine yakın olması, görüntüde yazı bulunan kısımların gölgede kalması ve bazı karakterlerin benzerliği örneğin B ve 8, O ve sıfır, I ve 1, A ve 4, K ve X, C ve G, D ve O gibi karakterler okuma doğruluğunu etkilemektedir. Bu çalışmada bu etkileri azaltmak için görüntü ön işlemleri (filtreleme, görüntü iyileştirme, morfolojik işlemler vb.) ve karakter ayrıştırma aşamaları üzerinde çalışmalar gerçekleştirmiştirler.

Plaka standartları, boyutları, renkleri ve kullanılan dilleri (latince, arapça, çince karakterler vb.) farklılık göstermektedir. Du, Massaud ve çalışma grubu tarafından gerçekleştirdikleri çalışmalarda[54,55], Mısır'a ait Arapça harflerden oluşan plaka karakterlerinin tanınması gerçekleştirilmiştir. Plaka bölgesinin tespiti için,

- Görüntünün kamera aracılığı ile elde edilmesi
- Renkli görüntünün, gri tonlamalı görüntüye dönüştürülmesi
- Sobel kenar bulma algoritmasının uygulanması
- Morfolojik açma ve kapama işlemlerinin gerçekleştirilmesi
- Görüntüde bulunan kapalı bölgelerin içlerinin doldurulması
- 5x5 Ortanca filtresinin görüntüye uygulanması
- Standart plaka boyutlarına uymayan kapalı bölgelerin elenmesi ile plaka bölgesi tespit gerçekleştirilmiştir.

Plaka bölgesi tespit edildikten sonra, plaka karakterlerinin uygun bir şekilde ayrıştırılması için morfolojik yayma (dilation) uygulanarak karakterler arasındaki mesafeleri arttırmıştırlar. Plaka karakterleri ayrıştırıldıktan sonra, her bir karaktere ait görüntü, belirlenen standart boyutlara dönüştürülmüştür. Şablon eşleştirme tekniğini kullanarak, elde edilen karakter görüntüleri istatistiksel ilgileşim (Korelasyon) yöntemini kullanılarak tanıma işlemini gerçekleştirmiştirler. Sırasıyla aranacak görüntüyü ve şablonu belirten iki ayrı görüntüyü temsil eden görüntü çifti arasındaki normalleştirilmiş çapraz ilgileşim tekniği ile tanımlamışlardır. Bu sistem, MATLAB yazılımı altında GUI ve UDP kullanılarak tasarlanmıştır. Önerilen sistem 100 adet görüntü seti ile %91 doğruluk oranında gerçekleştirilmiştir.

Zhang ve çalışma grubu tarafından gerçekleştirilen çalışmada[56], Destek Vektör Makinesi (SVM:Support Vektor Machine) Vektör uzayı tabanlı sınıflandırma işlemi yaparak, sınıfları vektörler aracılığı ile bir birinden ayıran makine öğrenmesi modelini kullanmışlardır. SVM modelini kullanarak plaka bölgesinin tespiti yapıldıktan sonra iz düşüm tekniği kullanılarak karakterler birbirinden ayrıştırılıp plaka numaraları belirlenmiştir. Evrişimli sinir ağı ve çok kanallı işleme için vektör makinesini destekleyen iki sınıflandırıcı, sırasıyla Çince karakterleri, sayıları ve alfabe harflerini tanımak için tasarlanmışlardır.



Şekil 1.14. Çin otomobil plakası tanıma sisteminin sistem mimarisi[56]

Şekil 1.14'te bu çalışmada[56], plaka karakterlerinin belirlenmesinde kullanılan sistem mimarisi verilmektedir. Model-1 (CNN) ve Model-2 (SVM) veri seti kullanılarak eğitimi gerçekleştirilmiştir. Kameradan elde edilen görüntülerde, görüntü işleme algoritmaları kullanılarak plaka bölgesinin çıkartılması sağlanmıştır. Karakterlerin ayrıştırılması işlemi, iz düşüm tekniği kullanılarak gerçekleştirilmiştir. CNN modeli kullanılarak Çince ve SVM modeli ile numerik ve alfabetik karakterlerin sınıflandırılması gerçekleştirilmiştir. Bu çalışmada, plaka karakterlerinin sınıflandırılması % 99,2 doğruluk oranında gerçekleştirilmiştir.

Zhai ve Bensaali tarafından gerçekleştirilen çalışmada[57], araç plakasını daha hızlı ve doğru okumak için görüntüde plaka bölgesi elde edildikten sonra karakterlerin ayrıştırma işlemini geliştirmişlerdir. Bu çalışmada piksel yansıtma ve morfolojik işlemlere dayanan gelişmiş bir karakter ayrıştırma algoritmasını kullanmışlardır.

Tablo 1.20. Karakterlerin ayrıştırılması sonuçları[57]

Başarı yüzdesi	1.Veri Seti	2.Veri Seti	3.Veri Seti	Ortalama
MATLAB	205/212(%96,7)	337/351(%96)	420/437(%96,1)	%96,2
FPGA	208/212 (%98,1)	343/351 (%97,7)	426/437 (%97,5)	%97,7

Tablo 1.20'de üç farklı veri seti kullanılarak karakterlerin ayrıştırılması başarı yüzdesi verilmektedir. Önerilen sistemde görüntü işleme algoritmalarının uygulanması ile plaka bölgesinin tespiti ve iz düşüm tekniği ile karakterlerin ayrıştırılma işlemlerini farklı veri setleri ile gerçekleştirmişlerdir. Karakter ayrıştırma işleminden sonra OCR tekniğini kullanarak plaka bölgesinde bulunan karakterlerin belirlenmesini sağlamışlardır.

Tablo 1.21.FPGA platformunda kullanılan kaynak miktarı[57]

	Kullanılan	Mevcut	Yüzde
Kullanılan dilimler	2100	18432	11
LUTs	3011	36864	8
BRAM	2	96	2
DSP48s	1	64	1

Bu sistem[57] ilk olarak MATLAB ortamında gerçekleştirilmiştir, daha sonra FPGA platformunda sistemin gerçekleştirilmesi için C programlama dilini, HLS ara yüzünü kullanarak gerçekleştirmişlerdir. Xilinx Virtex-4 LX40 ile Mentor Graphics RC240 FPGA geliştirme kartı kullanılmıştır. **Tablo 1.21**'de gerçekleştirilen sisteme ait FPGA kaynak kullanımı gösterilmektedir. Virtex-4 FPGA platformunda tüm işlemler, 74,5 MHz frekansla ve bir görüntüyü 0,2-1,4 ms'de, % 97,7'lik başarı ve 913 mW güç tüketimi ile gerçekleştirmişlerdir.

Fan ve arkadaşları tarafından yapılan çalışmada[58], yol şerit algılama, otomatik park ve insansız araçlar için gelişmiş Sürücü Asistanlık Sistemleri (ADAS) geliştirilmiştir. Bu çalışmada görüntü işleme algoritmaları TMS320C6678 SoC üzerine gömülerek şerit tespit ve ADAS sistemleri üzerine çalışmalar yapmışlardır. Önerilen doğrusal şerit algılama sistemi iki ana bölümden oluşmaktadır; görüntü ön işleme uygulamaları ve Hough dönüşümü kullanılmıştır. Ön işleme aşamasında, Gauss filtresini görüntüyü bulanıklaştırarak gürültüyü ortadan kaldırarak küçük ayrıntıları azaltmak için kullanılmıştır. Daha sonra Sobel filtresi uygulayarak kenar belirleme işlemleri gerçekleştirilmiştir. Algoritma, SoC DSP TMS320C6678 çok çekirdekli işlemcide, 1242×375 görüntü çözünürlüğü ile 81 fps hız ve %94,1 doğruluk oranıyla tol şerit çizgileri tespit edilmiştir. Calderon ve çalışma grubu tarafından gerçekleştirilen çalışmada[59], bilgisayar ve FPGA kartını kullanarak plaka belirleme işlemini gerçekleştirmişler. Kolombiya plakalarına ait özel bir algoritma kullanarak plaka bölgesini HSV düzleminde sarı rengin tanınmasıyla gerçekleştirilmiştir.

Promrit ve Wannarat tarafından yapılan çalışmada[60], Gerçek zamanlı yol şerit belirleme FPGA (Zybo board) gömülü sistem platformu kullanılarak belirlenmiştir. Bu çalışmada HT ve Hyperbola, Bézier Curve, Dot plot, Dot plot ve Hyperbola algoritmalarını karşılaştırarak şerit belirleme işlemini gerçekleştirmiştir. Gerçek zamanlı uygulamalarda hangi algoritmanın daha uygun olduğu belirlenmiştir. FPGA Zybo kartında 1920×1080 piksel çözünürlükte ve 60 fps olarak elde edilen görüntüler gerçek zamanlı şerit tespiti için uygun algoritmayı bulmak için dört algoritma deneysel olarak karşılaştırılmıştır. Nokta grafiği ve Hiperbol tekniği diğer iki algoritmaya göre daha iyi sonuçlar elde edilmiştir. Bu iki teknik kullanılarak, yol şeridi % 0 hata oranıyla tespit edilmiştir. FPGA platformunda gerçekleştirilecek uygulamalarda gerçek zamanlı şerit tespiti için Nokta grafiği ve Hiperbol algoritmalarının daha iyi olduğunu göstermişlerdir.

Jagannatan ve çalışma grubu bu çalışmada[61] araç, insan, trafik işaret levhaları gibi nesneleri sınıflandırmak için adaBoost, Convolution Neural Network (CNN) ve histograf of oriented gradients (HOG) algoritmalarını TDA3x SoC gömülü sistem platformu ile gerçeklemişlerdir. AdaBoost kullanarak nesne tespiti için, 3 düğümlü ve ağaç başına 4 yapraklı 2 seviyeli kademeli ağaçlardan oluşan yapı kullanılmıştır. Trafik levhalarını belirlemek için AdaBoost kullanarak %79,6 ve CNN kullanarak %89,6 doğruluk ile sınıflandırmıştırlar.

Li ve çalışma grubu tarafından gerçekleştirilen çalışmada[62], plaka bölgesinin tespitinde ve karakterlerin belirlenmesinde iki adet ardışık CNN modeli kullanılmıştır. Bu metodun avantajı plaka bölgesi (plaka var/yok) tespit edildikten sonra karakter ayrıştırma işleminde gerçekleşebilecek hatalardan dolayı eksik karakter tanımanın ortadan kalkacağını öne sürmektedirler. İki farklı CNN modeli kullanarak kademeli bir sınıflandırma işlemi gerçekleştirilmiştir. İlk CNN modeli ile görüntü üzerinde bulunan karakterlerin yerleri ve ikici CNN modeli ile de plaka bölgesi olmayan kısımlar elenerek, plaka bölgesi tespit edilmiştir. Bu çalışmada, MatConvNet kullanılarak 6GB belleğe sahip NVIDIA Tesla K40c GPU üzerinde sınıflandırma işlemleri gerçekleştirilmiştir. Ancak bu yöntem gerçek zamanlı araç plaka karakterlerinin tespitinde kullanılmak için veri işleme hızından dolayı uygun değildir. Bu çalışmada kullanılan eğitim veri tabanı, algılama ve tanıma için farklı üç alt gruba ayrılmıştır:

- AC: Kamera sabit ve görüntüdeki araç da sabit veya düşük hızda
- LE: Görüntüdeki araç trafik ihlali nedeniyle çekilmiş
- RP: Görüntüdeki araç hareketli kamera ile çekilmiş

Tablo 1.22. Farklı veri setlerinde plaka okuma[62]

	AC Veri Seti	**LE Veri Seti**	**RP Veri Seti**
Hassasiyet (%)	98.53	97,75	95,28
Özgüllük (%)	98,38	97,62	95,58

Tablo 1.22'de bu çalışmada[62], önerilen sistemin farklı veri setlerinde gerçekleştirilen sınıflandırma sonuçları verilmektedir. Bu veri setinde her bir görüntü için sınıflandırma süresi ortalama olarak 2-3 saniyede gerçekleştirilmiştir. Rafique ve çalışma grubu tarafından gerçekleştirilen çalışmada[63], iki farlı algoritma kullanarak plaka bölgesinin tespiti gerçekleştirilmiştir. SVM ve bölge tabanlı evrişimsel sinir ağı[64] (RCNN: Region-based Convolutional Neural Network) modeli kullanılarak gerçekleştirilen bu çalışmada plaka bölgesi tespitinin R-CNN kullanılarak daha iyi sonuçlar verdiği gözlemlenmiştir. Deney için kullanılan programlama kodları, OpenCV kütüphaneleri ile C ++ kullanılarak uygulanmıştır. SVM, orijinal MATLAB koduyla eğitilmiştir. RCNN de bu çalışmadaki yazarlar tarafından sağlanan kodla eğitilmiştir.

Xie ve çalışma grubu tarafından gerçekleştirilen çalışmada[65], Araç görüntülerinin uzaktan alınması veya plaka bölgesinin kameraya dik olmaması gibi sorunlar CNN tabanlı YOLO (You Only Look Ones) YSA modeli önerilerek plaka bölgesinin kameraya göre açısı belirlenip dik konuma getirildikten sonra karakter ayrıştırma metodu ile plaka bölgesindeki karakterler belirlenmeye çalışılmıştır. Orijinal YOLO modelinde, yalnızca her nesnenin merkez koordinatını, yüksekliğini ve genişliğini tahmin etmektedir. Bu çalışmada, önerilen yöntem açı bilgisini ve belirli bir araba plakası görüntüsünün dönme açısını belirlemesi için kullanılmaktadır. Araç plakası testleri için kullanılan üç farklı veri setinde sırasıyla %99,51-%99.43 ve %99.46 doğruluk oranlarında sınıflandırma işlemleri gerçekleştirilmiştir.

Hendry ve Chen tarafından gerçekleştirilen çalışmada[66], YOLO-darknet öğrenme modelini kullanarak araç plaka karakterleri tespit edilmeye çalışılmıştır. Altı karakterli tayvan plakası okuma işlemi gerçekleştirilmiştir. Karakterler kayan pencere yöntemi ile görüntü üzerinde kaydırılarak sıra ile belirlenmektedir.

Tablo 1.23. Plaka bölgesinin belirlenme oranları[66]

	AC Veri Seti	**LE Veri Seti**	**RP Veri Seti**
Hassasiyet (%)	100	99.01	95.67
Özgüllük (%)	99.53	98.62	95.71

Tablo 1.23'te önerilen model[66] kullanılarak üç farklı görüntü veri setine ait sonuçlar karşılaştırılmaktadır ve en iyi sonucu AC veri setinde elde etmişlerdir. Her bir plaka için veri işleme hızı 800ms-1s ve sistem plaka algılamada ortalama %98,22 doğruluk ve plaka karakterlerinin tanınmasında %78 oranında plaka karakterleri belirlenmişlerdir.

Viju tarafından gerçekleştirilen çalışmada[67], K-Means (KM) kümeleme algoritması ve Optik Karakter Tanıma (OCR) tekniğini kullanarak araç plaka tanıma sistemini gerçekleştirmiştir. Ortanca filtre gibi temel görüntü işleme algoritmalarını ve KM'yi kullanarak plaka bölgesini tespit ettikten sonra OCR tekniği ile plaka karakterlerini belirlemiştirler. Yousif ve diğerleri[68], görüntü işleme algoritmalarında plaka tanıma için yeni bir yöntem ve genetik algoritmaya (GA) dayalı optimize edilmiş bir nötronofik set (NS) önermişlerdir. NS setini kullanarak, plaka görüntülerini daha iyi belirleyebilmişlerdir. Plaka bölgesindeki karakterleri bölümlere ayırmak için k-ortalamalı kümeleme algoritması ve her bir karakteri etkili bir şekilde çıkarmak için uygun pikselleri gruplayarak, bağlantılı bileşen etiketleme analizi (CCLA) algoritması uygulamışlardır. Plaka karakterleri ayrıştırıldıktan sonra tüm resimler belirlenen boyuta dönüştürüldükten sonra istatistiksel çapraz ilgileşim yöntemini kullanarak karakterlerin eşleştirilmesi ve plakanın belirlenmesi işlemlerini gerçekleştirmişlerdir. Yüksel çözünürlüklü görüntülerde %96.67 ve düşük

çözünürlüklü görüntülerde ise %94.27 doğruluk oranlarında plaka karakterlerinin belirlenmesi işlemi gerçekleştirilmiştir.

Zhang ve çalışma grubu tarafından gerçekleştirilen çalışmada[69], CycleGAN (Generative adversarial networks) modelini ve LSTM (Long short-term memory) tabanlı dizi kod çözücü kullanarak plaka belirleme işlemini gerçekleştirmiştirler. Bu model ile plaka bölgesinin tam olarak belirlenemeyen görüntülerde, görüntü benzerliği özelliğini kullanarak karakterler tespit edilmeye çalışılmıştır. Önerilen modelin aşamaları;

- Giriş görüntüsüne eğitilmiş YOLOv2 modeli uygulanarak plaka bölgesinin tespiti
- Tespit edilen plaka bölgeleri, otuz katmanlı bir Xception ağı ile özellik matrisleri elde edilir
- Ayrıca plaka karakterlerinin tespiti için, Xception modeli ile bir ara özellik haritası çıkartılır
- Bu iki özellik matrisi ile plaka karakterlerinin tespiti için bir LSTM modeli kullanılır.

Tablo 1.24. Farklı veri setlerine ait önerilen modelin sonuçları[69]

	AC Veri Seti	**LE Veri Seti**	**RP Veri Seti**
Doğruluk (%)	97.3 (681 görüntü)	98.3 (757 görüntü)	91.9 (611 görüntü)

Tablo 1.24'te önerilen[69] sistemin üç faklı veri setine ait sonuçları listelenmiştir.

Agbeyangi ve çalışma grubu tarafından gerçekleştirilen çalışmada[70], Raspberry Pi kullanarak Open Computer Vision (OpenCV) açık kaynak kütüphaneleri kullanarak Optical Character Recognition (OCR) tekniği ile araç plaka karakterleri belirlenmiştirler. Fernandes ve çalışma grubu tarafından gerçekleştirilen çalışmada[71], plaka bölgesinin tespit edilmesi için Tiny YOLOV3 ve plaka bölgesindeki karakterlerin belirlenmesi için OCR tekniğini kullanmışlardır. İki farklı veri seti için Jetson TX2 geliştirme kartı kullanarak %96.87 ve %90.56 doğruluk oranında plaka karakterlerini tespit etmişlerdir. Izidio ve çalışma grubu tarafından gerçekleştirilen çalışmada[72], CNN modeli kullanarak brezilya için araç plaka tanıma sistemi geliştirmişler. Araç plaka bölgesinin belirlenmesinde Tiny YOLOV3 ve plaka karakterlerinin %98.43 doğruluk ile ikinci bir CNN model kullanılmıştır.

Castro ve çalışma grubu tarafından gerçekleştirilen çalışmada[73], plaka okuma işleminin gerçekleştirmek için Intel tarafından geliştirilen görüntü işleme tabanlı yapay zeka platformu olan OpenVINO'yu kullanmışlardır. Bu çalışmada, Evrişimsel sinir ağı ile görüntüde bulunan özelliklerin çıkartılması ile tek atış detektörü (SSD)'nü kendi çalışmaları için farklı bir şekilde kullanarak plaka karakterlerinin ayrıştırılması ve tanıma işlemlerini gerçekleştirmişlerdir. Bu yöntemde, daha az parametre kullanarak daha hızlı plaka karakterlerinin belirlenmesi sağlanmıştır.

Tablo 1.25. Farklı veri setlerinde plaka belirleme sonuçları[73]

	Caltech-Cars Veri Seti	UCSD-Stills Veri Seti
Karakter ayrıştırma(%)	96.46	99.79
Plaka tanıma(%)	96.23	99.79

Tablo 1.25'te Önerilen modele[73] ait plaka tanıma ve karakterlerin ayrıştırılması işlemlerinin sonuçları verilmektedir. On iki çekirdekli Intel Xeon CPU (2.60 GHz) bilgisayar ve OpenVINO ile 14 ms'de, Raspberry Pi kullanılarak 66ms'de plaka karakterlerini tespit etmiştirler.

Weihong ve Jiaoyang tarafından gerçekleştirilen çalışmada[74], kullanılan kamera, düzensiz aydınlatma, sabitlenmemiş çekim açısı, farklı hava koşulları (sis, yağmur, kar vb.), görüntünün elde edildiği saat (gece, gündüz), hareket bulanıklığı ve plaka bölgesinin kirlilik oranı gibi kötü şartlarda elde edilen resimlerde plaka okuma işleminin gerçekleştirilmesinde öneriler ve gerçekleştirilen çalışmalar karşılaştırılmıştır. Ma ve Zhang tarafından gerçekleştirilen çalışmada[75], araç plaka bölgesi olabilecek yerler belirlenirken oluşabilecek hataları ortadan kaldırmak için, CNN yapısı kullanarak araç logosunu tespit etmişlerdir. Böylelikle plaka bölgesine odaklanıp plaka numaralarının analizini gerçekleştirmişler.

Tablo 1.26.Önerilen sistemin doğruluk analizi[75]

Kullanılan Görüntü Sayısı	Özgüllük (%)	Doğruluk (%)
20	57	57
40	51	62
60	65	67
80	73	70
100	88	94

Tablo 1.26'da CNN yapısı kullanılarak araç plaka bölgesinin tespiti için araca ait marka sembolü tespiti sonuçları gösterilmektedir. Bu çalışmaya[75] ait deneysel sonuçlar, kullanılan yöntemin %94 yüksek doğruluk oranına sahip olduğunu ve görüntü veri bloğu için %88 özgüllük (Recall) oranına sahip olduğunu göstermişlerdir.

Heidari ve çalışma grubu[76], akciğer X-ışını görüntüleri (AXG) kullanarak iki adet veri seti oluşturmuşlardır. Birinci veri seti, diyafram bölgesini tespit etmek ve görüntüden kaldırmak için bir ön işleme algoritması oluşturulmuş ve ikinci set ise orijinal x-ışını görüntülerini işlemek için histogram eşitleme algoritması ve iki taraflı filtre (Bilateral) uygulamışlardır. AXG'inde diyafram bölgesi çıkartıldıktan sonra, üç boyutlu (RGB) görüntü elde edebilmek için sırasıyla, histogram eşitleme, bilateral filtre ve diyafram bölgesi çıkartılmış gri tonlamalı AXG kullanılmıştır. Derin

öğrenme model eğitimi için ImageNet veri seti ile daha önceden eğitilmiş VGG16 yapısını kullanarak transfer öğrenme gerçekleştirmişlerdir. Kullanılan yöntem ile AXG görüntülerinin sınıflandırılması işlemi %94,5 (normalizasyon kullanmadan %88) doğruluk oranında gerçekleştirilmiştir.

Wang ve çalışma grubu[77], AXG'ne normalizasyon tekniği uygulamak için, orijinal görüntü gri tonlamalı görüntüye dönüştürülmüş ve OTSU eşik değer uygulaması ile siyah beyaz görüntü elde ettikten sonra arka planda bulunan siyah bölgeler beyaz renk ile değiştirildikten sonra tersleme işlemi gerçekleştirilmiştir. M-inception modeli kullanılarak transfer öğrenme ile AXG verilerinin sınıflandırılması işlemi gerçekleştirilmiştir. 1065 adet AXG görüntüsünün sınıflandırılması normalizasyon uygulanmadan önce %82,5 ve normalizasyonun etkisi ile %93 doğrulukta gerçekleştirilmiştir. Bu çalışmanın uzman görüşleri ile 745 adet görüntünün değerlendirme sonuçları, birinci radyolojist %55,8 ve ikinci radyolojist %55,4 doğruluk oranlarında tespitler gerçekleştirmişlerdir. Nayak ve çalışma grubu[78], genel olarak kullanılan sekiz model (VGG16, ResNet, AlexNet vb.) ile transfer öğrenme gerçekleştirmişlerdir. Öğrenme işleminden önce veri setine normalizasyon uygulanmıştır. Öğrenme sürecini hızlandırmak için, görüntüye ait piksel değerleri 0-1 sayı aralığında normalize edilmiştir.

Rahman ve çalışma grubu[79], akciğer x-ışını görüntülerinde virüs tespiti için gerçekleştirmiş oldukları sınıflandırma uygulamasında beş farklı görüntü işleme, Histogram eşitleme, Negatif dönüşüm, Gama iyileştirme, Kontrast geliştirme tekniklerini giriş görüntülerine uygulayarak model eğitimini gerçekleştirmişlerdir. Farklı görüntü işleme tekniklerinde covid-19 virüsünün tespit edilmesi altı farklı CNN model, ResNet18, ResNet50, ResNet101, InceptionV3, DenseNet201 ve ChexNet kullanılarak araştırılmıştır. Aynı öğrenme modeller, U-Net modeli ile akciğer bölgesinin görüntüden çıkartılmasından sonra öğrenme sonuçları karşılaştırılmıştır. Model eğitim süreci; X-ışını görüntülerinde, U-Net aracılığı ile akciğer bölgesinin görüntüden çıkartılması ve orijinal görüntülere beş farklı görüntü işleme tekniğinin uygulanması ile iki farklı veri seti oluşturulmuştur. Ardından, Altı farklı CNN model yapısı bu verilere ayrı ayrı uygulanır ve sonuçlandırma işlemi gerçekleştirilmiştir. Gama iyileştirme tekniğine normalize edilmiş ChexNet modeli, görüntü ayrıştırma olmadan en iyi performansın sağlandığını, aynı iyileştirme ile DenseNet201, ayrıştırılmış akciğerler için daha iyi performans elde edildiğini göstermişlerdir.

Literatürde, FPGA, GPU ve CPU yapıları gerçekleştirilen uygulamalar bakımından kıyaslanarak birbirilerine göre üstünlükleri gözlemlenebilmektedir [1-13]. Genel olarak kullanılan derin öğrenme modellerinin (NN, CNN vb.) farklı uygulamaları gösterilmektedir[16-22]. Tıp alanında derin öğrenme modelleri ile uygulamalar gerçekleştirilmiştir [28-42]. Gömülü sistem platformları ve derin öğrenme modelleri kullanılarak, güvenlik tedbirleri ve suç önleme

uygulamaları [43-45], tarım ve tekstil uygulamaları [46-50], Araç plaka okuma ve karakterlerin sınıflandırılması [52-75] ve x-ışını görüntülerine farklı normalizasyon teknikleri uygulayarak sınıflandırma [76-79] çalışmaları gerçekleştirilmiştir. Araç plaka tanıma için literatürde kullanılan yöntemler **Tablo 1.27**'de ve farklı normalizasyon teknikleri ile derin öğrenme modellerinin x-ışını görüntülerinde sınıflandırma işlemi Araç plaka tanıma için literatürde kullanılan yöntemler kıyaslandığında, açık kaynak kütüphaneler ve GPU yapısına sahip gömülü sistem platformlarında plaka belirleme oranlarının daha yüksek olduğu gözlemlenmektedir. Plaka bölgesinin belirlenmesi işlemlerinin derin öğrenme modelleri (YOLO, R-CNN, CNN vb.) ile gerçekleştirilmesi doğruluk oranını arttırmaktadır.

Tablo 1.28'de gösterilmektedir.

Tablo 1.27. Araç plaka tanıma sistemlerine ait literatürde gerçekleştirilen uygulamalar

Plaka bölgesinin tespiti	Karakterlerin ayrıştırılması	Karakterlerin tanıma işlemi	Kullanılan platformlar	Plaka tanıma Doğruluk oranı	Kaynaklar
Renk dönüşümü, morfolojik işlemler	İz düşüm tekniği	NN, LVQ NN	-	%66, %94,44	[51]
Renk dönüşümü, kenar algılama, morfolojik işlemler,	Bağlı Bileşen Analizi (CCA)	Otiksel karakter tanıma	MATLAB, FPGA	%99	[52]
Renk dönüşümü, kenar algılama, morfoloji işlemler	Morfoloji, iz düşüm tekniği	Şablon eşleştirme	MATLAB	%91	[54,55]
Temel görüntü işleme teknikleri	İz düşüm tekniği	CNN, SVM	-	%99.2	[56]
Renk dönüşümü, morfolojik işlemler	İz düşüm tekniği	OCR	MATLAB, FPGA	%96.2, %97.7	[57]
CNN	İz düşüm tekniği	CNN	NVIDIA Tesla K40c	%97.75	[62]
R-CNN	-	-	MATLAB	%100	[63]
YOLO-darknet	-	-	GPU K40	%99,5 %99,43, %99,46	[65]
YOLO-darknet	-	-	Nvidia GTX970 GPU	%98,2	[66]
K-ortalama	İz düşüm tekniği	OCR			[67]
Temel görüntü işleme teknikleri, Morfolojik işlemler	Neutrosophic set, K-ortalama,	Karakter eşleştirme	MATLAB	%96.67, %94.27	[68]
YOLO	Xception ağı	LSTM	-	%97.3, %98.3, %91.9	[69]
YOLO	-	OCR	Jetson TX2	%96.87,	[70–72]

				%90.56	

Araç plaka tanıma için literatürde kullanılan yöntemler kıyaslandığında, açık kaynak kütüphaneler ve GPU yapısına sahip gömülü sistem platformlarında plaka belirleme oranlarının daha yüksek olduğu gözlemlenmektedir. Plaka bölgesinin belirlenmesi işlemlerinin derin öğrenme modelleri (YOLO, R-CNN, CNN vb.) ile gerçekleştirilmesi doğruluk oranını arttırmaktadır.

Tablo 1.28. Akciğer X-ışını görüntülerinin farklı normalizasyon teknikleri ile sınıflandırılması

Kullanılan teknik	Öğrenme modeli	Kullanılan veri miktarı	Normalizasyon öncesi-sonrası sonuçlar		Kaynaklar
Akciğer x-ışını görüntülerinde diyafram bölgesi çıkartılmış, Histogram eşitleme ve Bilateral filtre uygulanmıştır	VGG16 modeli, ImageNet verileri ile Transfer öğrenme	8474 görüntüde, 415 covid-19 virüslü, 5179 diğer virüslere sahip ve 2880 normal	%88,0	%94,5	[76]
AXG görüntü işleme teknikleri uygulanmıştır	GoogleNetM-inception modeli, ImageNet verileri ile transfer öğrenme	1065 görüntü	%82,5	%93	[77]
AXG görüntü piksel değerlerinin 0-1 sayı aralığında normalizasyonu	AlexNet, VGG-16, GoogleNet, MobileNet-V2, SqueezeNet, ResNet-34, Inception-V3 modelleri ile transfer öğrenme	406 görüntü, 203 covid-19 virüslü, 203 normal		%98.33, %97.50, %96.67, %95.83, %97.50, %95.83, %92.50, %96.67	[78]
Histogram eşitleme, Negatif dönüşüm, Gama düzeltmesi, Kontrast geliştirme teknikleri uygulanmıştır	ResNet18, ResNet50, ResNet101, InceptionV3, DenseNet201 ve ChexNet	18479 görüntü, 3616 adet Covid-19 virüslü	%93,22	Ortalama %96,29	[79]

Normalizasyon tekniği kullanılarak gerçekleştirilen literatürde kullanılan yöntemler ile akciğer x-ışını görüntülerinde covid-19 virüsü için gerçekleştirilen uygulamaların daha iyi sonuçlar verdiği **Tablo 1.28**'de gösterilmektedir.

1.2. Tezin Amacı ve Kapsamı

Bu tez çalışmasında, gömülü sistem platformlarında görüntü işleme yöntemleri ve derin öğrenme modelleri kullanılarak uygulamalar geliştirilmiştir. Bu çalışmadaki genel amaç, gerçek zamanlı görüntü işleme uygulamalarında uygun gömülü sistem platformlarının seçimi ve yeni uygulamaların geliştirilmesini kapsamaktadır. Bu uygulamalar geliştirilirken, bilgisayar tabanlı

(MATLAB, Anaconda ara yüzü), Raspberry Pi ve FPGA gömülü sistem platformları kullanılmıştır. Bu kapsamda gerçekleştirilen çalışmalar genel olarak;

- Kameralardan elde edilen görüntülerin uygulama geliştirebilmek için gömülü sistem platformlarına uygun veriler haline dönüştürülmesi,
- Gömülü sistem platformlarına uygun yazılım dilleri ile görüntü işleme ve derin öğrenme algoritmalarının geliştirilmesi,
- Bilgisayar tabanlı platformlarda gerçekleştirilen görüntü tanıma uygulamalarının gömülü geliştirme kartları kullanılarak gerçekleştirilmesi ve geliştirilmesi,
- Görüntü işleme uygulamalarının derin öğrenme modelleri ile gerçek zamanlı gerçekleştirilmesi

Bu tez çalışması kapsamında, gömülü sistem platformu, kamera ve ekrandan oluşan bir sistem kurulmuştur. Derin öğrenme model eğitim aşamaları için veri setleri oluşturulmuştur. Gerçekleştirilen sistem ile eğitilmiş modeller kullanılarak gerçek zamanlı uygulamalar için kullanılmıştır. Bu tezde görüntü işleme algoritmalarının geliştirilmesi ve bu geliştirilen kodların uygun platformlarda çalıştırılması etkin fayda sağlayacaktır. Ayrıca düşük maliyetli, yüksek performanslı gömülü sistemlerin gerçek zamanlı görüntü işleme uygulamalarında çalışacak araştırmacılar için faydalı bir kaynak olacaktır. Bu tezde gerçekleştirilen uygulamalar farklı gömülü sistem platformlarında ve bu platformlara ait yazılım dilleri ile gerçekleştirildiğinden araştırmacıları farklı bakış açılarına yönlendirecektir.

1.3. Tezin Organizasyonu

Bu tez çalışması yedi bölümden oluşmaktadır.

Birinci bölümde giriş, tezin amacı/kapsamı ve literatürde daha önce gömülü sistem platformları kullanılarak gerçekleştirilen görüntü işleme çalışmaları ve kullanılan teknikler incelenmiştir.

İkinci bölümde, genel olarak kullanılan gömülü sistem teknoloileri açıklanmış ve karşılaştırılmıştır. Gömülü sistem platformları ve uygulama alanları anlatılmaktadır.

Üçüncü bölümde, Yapay sinir ağı modelleri ve model değişkenlerinin eğitime katkısı ve kullanılabilecek farklı sinir ağı modellerinin sahip olması gereken özellikler açıklanmaktadır.

Dördüncü bölümde, tez çalışması kapsamında gerçekleştirilen temel görüntü işleme uygulamalarının, gömülü sistem platformlarında gerçeklemeleri sunulmaktadır.

Beşinci bölümde, tez kapsamında geliştirilen normalizasyon tekniğinin akciğer x-ışını görüntülerine uygulanması ve bu normalizasyon tekniğinin, CNN modelde kullanılması ile gerçekleştirilen sınıflandırma işlemleri sunulmaktadır.

Altıncı bölümde, araç plaka bölgelerinin belirlenmesi, standart görüntü işleme teknikleri ve önerilen MVSR normalizasyon tekniğinin görüntülere uygulanması ile gerçekleştirilen tanıma işlemleri aktarılmaktadır. Plaka bölgesinin tespitinden sonra bu bölgelerde bulunan karakterlerin ayrıştırılması ve karakter tanıma aşamaları anlatılmıştır. Son olarak, tez çalışması kapsamında elde edilen sonuçlar değerlendirilmiş ve öneriler yedinci bölümde sunulmaktadır.

2. GÖMÜLÜ SİSTEM TEKNOLOJİLERİ

Gömülü sistemler, donanım ve yazılım alt birimlerine sahip, giderek daha karmaşık ve ağ bağlantılı yapılar hale gelen, tasarlanan sistemin kontrolünü gerçekleştiren yapılardır. Devam eden teknolojik evrim (Nesnelerin interneti, Akıllı evler, Güvenli taşımacılık, Otonom sistemler vb.) gömülü sistemleri daha yaygın hale getirmiştir. Bununla birlikte, etkili, verimli ve güvenilir gerçek dünya çözümleri tasarlamak için veri toplama ve işleme aşamalarının yanı sıra karar verme aşamalarında da farklı düzeylerde zekâ ile donatılmaları gerekmektedir. Bu nedenle gömülü sistemlerin gerçek zamanlı performansı ve ağa gömülü cihazların görevlerini sorunsuzca yerine getirebilmeleri için donanım ve yazılımın daha etkin kullanımı ve geliştirilmesi gerekmektedir[80].

Gömülü sistemleri uygulama alanlarına göre dört gruba ayırabiliriz.

1. Bağımsız Sistemler
2. Gerçek zamanlı Sistemler
3. Ağ yapısına bağlı sistemler
4. Harekete duyarlı sistemler

Bağımsız sistemlerde, giriş verilerini dış ortamdan elde edip, sahip oldukları yazılımlar ile başka sistemlerden bağımsız olarak çalışabilen sistemlerdir. Bu sistemlerin çıkış bilgileri, bağımsız olarak çıkış birimlerine aktarılmaktadır. Üretim sektöründe kullanılan otonom robotik sistemler bağımsız çalışabilen gömülü sistemlere örnek olarak verilebilir[81]. Gerçek zamanlı olarak çalışabilen gömülü sistemler, sahip oldukları yazılım ve donanım birimleri ile giriş verilerini dışarıdan alıp karşılaştıran ve eş zamanlı olarak içinde bulunduğu sistemleri kontrol edilebilen yapılardır. Ağa bağlı sistemler, birden fazla kullanıcının ulaşabildiği veya ortak ağ üzerinden iletişim kurulabildiği yapılardır. Bu yüzden ağ yapısının desteklendiği donanımlara sahip gömülü sistemler üretilmektedirler. Bu yapılar aynı zamanda internet erişimi ile de kontrol edilebilmektedirler[82]. Mikro şebekeler, Akıllı şehirler veya binalar bu sistemlere örnek olarak verilebilir.

2.1. Gömülü Sistem Platformlarının Teknik Özellikleri

Gömülü sistemler farklı uygulama alanlarında kullanılmak üzere üretilen işletim sistemine sahip yapılardır. Sistem tasarımlarında zaman kararlılığı, enerji tüketimi ve hafıza alanı gibi faktörler düşünüldüğünde, gerçek zamanlı çalışan platformlarının yerine getirmesi gereken en temel görevler bile zorlu hale gelmektedir. Yeni bir sistem inşa edilirken, planlanan iş paketlerinin miktarı ve bu görevleri yerine getirmesini sağlayan yazılımların niteliği arttıkça farklı gömülü sistem platformlarına ihtiyaç duyulmaktadır. Yarı iletken teknolojisinin gelişimi ile günümüzde

kullanılan hemen hemen tüm cihazların içerisinde bir ve birden fazla gömülü sistem platformu bulunmakta ve bir çok işlevi yerine getirmektedirler.

Zaman kararlılığı yüksek olan FPGA platformları ile bilgisayar ve GPU tabanlı sistemler karşılaştırıldığında, FPGA işlemleri donanımsal olarak çok hızlı ve aynı anda bu donanımsal blokların paralel olarak çalışabilmesini sağlayan programlanabilir mantık bloklarından oluşmaktadır. Bu donanım yapısı tekrardan düzenlenebildiği için daha çok tercih edilmektedirler. Gömülü sistemlerde, özel tümleşik devreler (ASIC:Application Specific Integrated Circuit), özel tasarımlara özel çözümler ve sadece belirli bir fonksiyonu/görevi yerine getirmek için üretildikleri için farklı yapılar için kullanılamayacaktır[2,3]. Ancak FPGA'lar üretimden sonra, tasarımcının fonksiyonuna bağlı olarak yeniden yapılandırılabilindiği için alanda programlanabilir kapı dizisi olarak adlandırılmıştır. Bu özelliklerinden dolayı, yeniden programlanabilen mantık kapıları büyük bir öneme sahiptir.

Raspberry Pi ve Jetson geliştirme kartı kullanılarak, farklı boyutlarda veriler ile sinir ağı modeli kullanıp sınıflandırma işlemi yapıldığında sinir ağının sınıflandırmadaki doğruluk oranı, veri işleme hızı, kullanılan hafıza ve güç tüketimlerinin karşılaştırılması yapıldığında Jetson kartlarının daha iyi sonuçlar verdiği gözlemlenebilmektedir. Veri işleme hızları daha yüksek, güç tüketimleri daha az ve artan veri kapasitesinde daha da iyi sonuçlar vermektedir[11].

Gömülü sistemler genelde, tasarıma bağlı olarak bir veya çoklu görevleri yerine getirmek ve kontrol etmek için programlanabilen elektronik platformlardır. Bu platformlar mikroişlemci veya mikrodenetleyici tabanlı, gerçek zamanlı işlemler/hesaplamalar gerçekleştirmek için I2C (Entegre devreler), bellek, RAM, A/D dönüştürücüler ve seri haberleşme modüllerine sahiptirler.

Şekil 2.1. Gömülü sitem yapısı

Şekil 2.1'de genel bir gömülü sisteme ait yapı gösterilmektedir. Bu yapı donanım, yazılım, giriş-çıkış, diğer sistemlerle bağlantı ve kullanıcı arayüzünden oluşmaktadır. Gömülü sistemlerin tasarımı veya seçimi yapılırken dikkat edilmesi gereken başlıca noktalar şunlardır:

- Veri işleme hızı
- Hafıza
- Zamansal kararlılık
- Maliyet
- Boyut
- Kullanım süresi

DSP'ler, sinyal işleme görevlerini gerçekleştirmek için özel olarak tasarlanmış bir mimariye sahip mikro işlemcilerdir. Texas Instruments (TI) ve Analog Devices (AD), DSP üretim pazarındaki iki büyük şirkettir. Texas Instruments'in Ürettiği DSP'ler, bilgisayarla görme ve görüntü işleme uygulamalarında daha yaygın olarak kullanılmaktadır. Ultra düşük güçlü üretilen DSP'ler uygun maliyetlidirler, fakat düşük hesaplama gücüne sahip oldukları için genellikle basit bilgisayar görüşü ve temel görüntü işleme algoritmalarıyla kullanılmaktadırlar.

DSP'lerdeki kayan nokta operasyonel desteği, algoritma uygulamasını kolaylaştırır ve sabit noktaya kıyasla hassasiyeti artırır. Ancak, sabit nokta işlemleri daha az bit ile işlemler gerçekleştirebilir, ancak programcının her matematiksel işlemden sonra ondalık noktayı dikkatlice yeniden konumlandırması gerekir. Bununla birlikte, sabit noktalı DSP'ler genellikle kayan noktalı DSP'lerden daha ucuzdur. DSP'lerde, aritmetik hesaplamalar çarpan-toplayıcı (MAC) birimlerinde gerçekleştirilir. DSP'leri, mikro denetleyicileri ve ARM işlemcileri programlamak için Texas Instruments firması tarafından geliştirilen SYS/BIOS (eski ismiyle DSP/BIOS) gerçek zamanlı işletim sistemi kullanılabilmektedir[83].

GPU'lar ilk olarak iş istasyonları için üretilmiştir. birçok işlemci çekirdeğinden oluşan ve paralel olarak hızlı matris hesaplamaları gerçekleştirebilen optimize edilmiş hızlandırıcılardır. Oyun endüstrisi tarafından geliştirildiklerinden, bu cihazlar uygun fiyatlı ve yaygın olarak kullanılmaktadır. Bu nedenle GPU'lar, son yıllarda oyunlar dışında görüntü işleme uygulamalarında da yaygın olarak kullanılmaktadır. NVidia GPU serisi sahip oldukları kütüphaneler ve bir çok çekirdek mikro mimriye sahip oldukları için, görüntü işleme ve derin öğrenme uygulamalarında yaygın olarak kullanılmaktadır[84].

GPU kullanımının avantajları;

• GPU'lar, görüntü ve video işleme için özel olarak tasarlanmıştır.

• GPU'larda kod geliştirmek ve hata ayıklamak FPGA platformlarına göre daha hızlı ve kolaydır.

• GPU kartları ve ana bilgisayar arasındaki PCI (Çevresel bileşen ara bağlantı) arabirimi, programcılar tarafından kolayca kullanılabilir.

GPU kullanımının dezavantajları;

• GPU'lar, aynı performans sınıfındaki FPGA platformlarına kıyasla önemli ölçüde daha fazla güç tüketirler. Bu nedenle güç tüketiminin önemli olduğu sistemler için uygun değillerdir.

• GPU'lar, veri paralelliği olan ve gerektiren uygulamalar için tasarlanmıştır. Ancak, GPU'ların performansı, verileri beklemeleri gerektiğinde veya verilerin işlenmesi zaman alıcı ve yavaşsa önemli ölçüde azalacaktır.

Günümüzde gömülü sistemlerin kullanımı her geçen gün daha fazla artmakta ve önem kazanmaktadır. En yaygın olarak akıllı ulaşım, bilgisayar insan etkileşimi, akıllı tarım, robotik, savunma sanayi, tıp ve insansız kara, deniz ve hava araçları vb. gibi alanlarda kullanılmaktadır.

Şekil 2.2. Gömülü sistemler uygulama alanları

Şekil 2.2'de genel olarak kullanılan gömülü sistem platformlarının uygulama alanları gösterilmektedir.

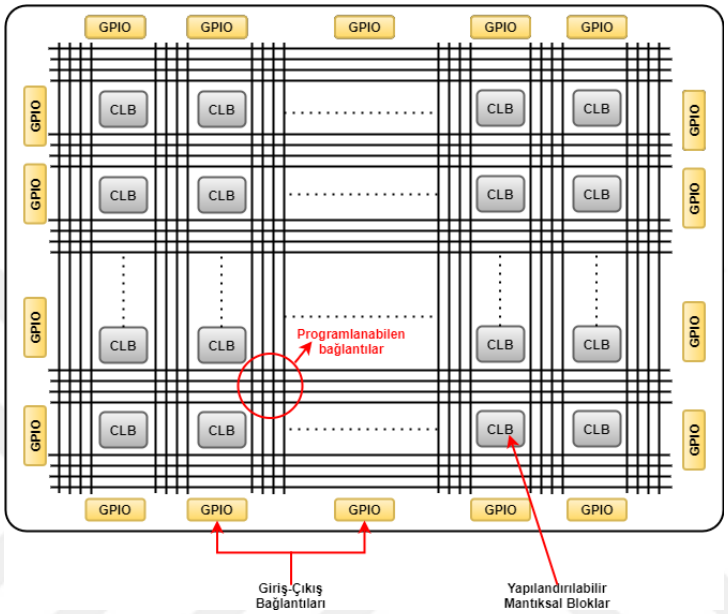
2.1.1. Alanda Programlanabilen Kapı Dizileri (FPGA)

Mantık devrelerinin tasarımında kullanılan ilk programlanabilir mantık devre elemanı PROM (Salt okunabilir devre)'dur. Bu yapılar genel olarak, birkez programlanabilen (PROM, PLA vb.) ve birden azla programlanabilen yapılar (EPROM, EEPROM) olarak iki gruba ayrılmaktadır. Ancak bu yapılar, gelişmiş uygulamalar için yetersiz kalmaktadırlar. Karmaşık yapılarda kullanılabilen mantıksal fonksiyonlar dört bölümde incelenebilmektedir[85].

- SPLD: Basit programlanabilir mantıksal devreler
- CPLD: Karmaşık programlanabilir mantıksal devreler

- MPGA: Maske programlanabilir mantıksal kapı dizileri
- FPGA: Alan programlamalı kapı dizileri

Programlanabilir yapıların (CPLD ve MPGA) avantajlarını kullanarak Xilinx firması tadarfından FPGA platformu 1985 yılında uygulama geliştiricilerine sunulmuştur. FPGA platformları genel olarak; yeniden yapılandırılabilir mantıksal bloklar, ara bağlantılar ve giriş çıkış birimlerinden oluşmaktadır.



Şekil 2.3. FPGA platformu genel yapısı

Şekil 2.3'te FPGA platformunun genel yapısı gösterilmektedir. FPGA donanım mimarisi, belirli bir görevi gerçekleştirmek için FPGA mantık kapılarını birbirine bağlayarak yapılandırılır ve her yeni algoritma için yeniden yapılandırma gerektirir. FPGA'lar bu nedenle genellikle yeniden yapılandırılabilir cihazlar olarak adlandırılır. CPU'lar, DSP'ler ve GPU'ların aksine, FPGA yapılarının önceden yapılandırılmış bir çip mimarisi veya merkezi bir işlem birimi yoktur. Bu nedenle, FPGA üzerinde bulunan yapılandırılabilir mantık blokları (CLB), başvuru tabloları (LUT'lar), çoklayıcılar (Multiplexer) ve Flip-flop gibi bileşenler kullanarak, özel donanım mimarisi tasarlanmaktadır.

FPGA platformlarında programlama geliştirme için kullanılan yazılım dilleri (VHDL/Verilog) CPU'lar, DSP'ler ve GPU'lar için kullanılanlardan oldukça farklıdır. FPGA'lerin programlanmasını kolaylaştırmak için, C,C++ benzeri yazılım dilleri (HLS gibi yapılarda) kullanılmakta ancak paralel çalışma mantığına fazla uymadığı için çok nadir tercih edilmektedir. Xilinx firması tarafından üretilen FPGA platformları, Spartan serisi, genel olarak basit uygulamalar için tasarlanmış düşük maliyetli FPGA'lardır. Virtex serisi, sinyal işleme uygulamalarını gerçekleştirmek için özel olarak tasarlanmıştır ve diğer FPGA platformlarına kıyasla daha pahalıdır.

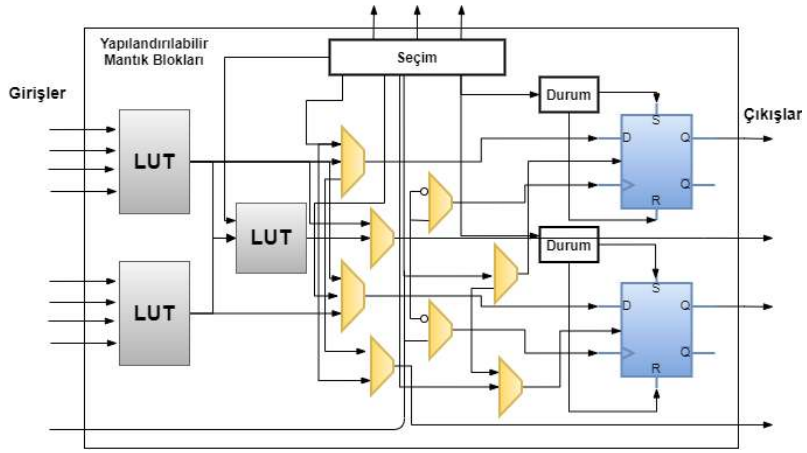
Kintex ve Artix serileri, düşük performanslı ve Virtex-7'nin ucuz versiyonlarıdır. Zynq serisi üretilen platformlar orta seviye olarak üretilen (Artix veya Kintex) ve bir ARM işlemci içeren orta ila yüksek performanslı gömülü sistem platformudur. UltraScale ve UltraScale + aileleri, en yeni Xilinx teknolojileridir. Virtex UltraScale + FPGA, Xilinx firması tarafından üretilen en yüksek performanslı FPGA'sıdır [87]. Yüksek performanslı ve yüksek hızlı uygulamalar için tasarlanmıştır.**Tablo 2.1**'de FPGA platformları ve bunlara ait düzlemsel üretim teknolojileri/süreçleri gösterilmektedir.

Tablo 2.1. FPGA platformlarının üretim teknolojileri

FPGA Platformları	Üretim Teknolojileri
Virtex-4, Spartan-3E	90nm
Virtex-6, Spartan-6	40nm
Virtex-7, Kintex, Artix, Zynq	28nm
UltraScale (Virtex, Kintex, Zynq)	20nm
UltraScale+(Virtex, Kintex, Zynq)	16nm

Bilgisayarla görme ve görüntü işleme uygulamalarında uygun FPGA seçimi için önemli bazı özellikler aşağıda özetlenmiştir.

- CLB: Yapılandırılabilir mantık bloklarının ara bağlantıları, kullanıcıların tanımladığı donanımı temsil eden mantık devrelerini oluşturur. Çok sayıda CLB'ye sahip olan FPGA'lar, karmaşık algoritmalar için uygun bir seçimdir[86]. **Şekil 2.4**'te Xilinx XC400 FPGA Yapılandırılabilir mantık bloğu yapısı gösterilmektedir.



Şekil 2.4. Xilinx XC400 FPGA Yapılandırılabilir mantık bloğu yapısı[87]

- DSP: FPGA'da bulunan DSP blokları temel matematik işlemleri ve sinyal işleme için tasarlanmıştır. Bu bloklar, Xilinx FPGA'larda DSP48 olarak adlandırılır ve Xilinx platformlarında

farklı sayıda bitlere sahip olabilirler. DSP'lerin performansı, tek bir DSP diliminin saniyede gerçekleştirebileceği maksimum matematiksel işlem sayısı ile ölçülmektedir. Gerçek zamanlı uygulamalarda DSP'lerin performanslarına bağlı olarak karmaşık yapılar gerçekleştirilebilir veya problem olarak karşımıza çıkabilmektedir.

- BRAM: Blok RAM'ler, verilerin depolanması veya arabelleğe alınması için SRAM mimarisiyle tasarlanmıştır ve özellikle görüntü verilerini FPGA'ya depolamak için önemlidir. Yazma etkinleştir girişi mantıksal bir değerini aldığında, saat darbesi ile adres bilgisi ve veriler hafızaya yüklenmektedir. BRAM tasarımına bağlı olarak ne kadar verinin hafızada tutulacağı veya hafızadan okunacağı tasarımcıya bağlı olarak değişmektedir. Geleneksel görüntü işleme algoritmaları sıralı işlemciler için tasarlanmıştır. FPGA platformlarında bu algoritmalar, paralel işleme için değiştirildiğinde ve optimize edildiğinde daha iyi bir performans elde etmek mümkündür[88].

Tablo 2.2. FPGA platformlarının karşılaştırılması

	Spartan-7	**Artix-7**	**Kintex-7**	**Virtex-7**
Mantık blokları	102k	215k	478k	1955k
BRAM	4.2 Mb	13 Mb	34 Mb	68 Mb
DSP hücresi	160	740	1920	3600
I/O pinleri	400	500	500	1200
I/O gerilimleri	1.2-3.3V	1.2-3.3V	1.2-3.3V	1.2-3.3V

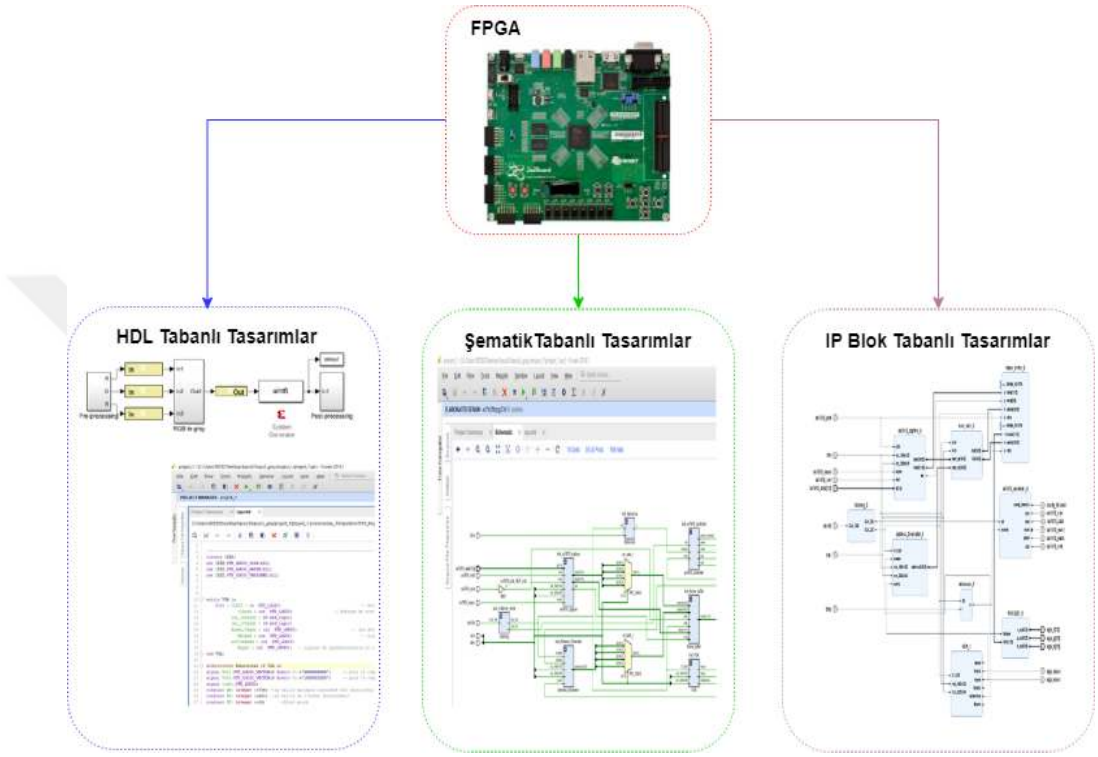
Tablo 2.2'de Xilinx firmasının ürettiği yedi serisi FPGA platformlarının karşılaştırılması verilmektedir. Xilinx-7 serisi FPGA'lar, düşük maliyetli ve tüm sistem gereksinimlerini karşılayan dört FPGA ailesinden oluşmaktadır. Spartan-7 grubu FPGA platformu; Düşük maliyetli ve diğer platformlarla kıyaslandığında en düşük hafıza ve DSP birimlerine sahiptirler. Gerçekleştirilecek uygulamaya göre FPGA platformlarının seçimleri büyük önem teşkil etmektedir. Tasarım gerçekleştirilirken kullanılacak hafızaya, mantık bloklarına, DSP hücre sayılarına dikkat edilerek FPGA platformlarının seçimleri gerçekleştirilmelidir.

FPGA gömülü sistem platformu kullanarak tasarım gerçekleştirmek için üç genel tasarım yöntemi kullanılmaktadır.

- ✓ HDL, Donanım Tanımlama Dili (Hardware Description Language)
- ✓ Şematik diyagramlar
- ✓ Özel IP Çekirdek tabanlı tasarımlar

HDL tabanlı tasarımlar genel olarak MATLAB ve XSG benzetim blokları ile veya sadece FPGA programlama dilleri kullanılarak gerçekleştirilmektedir. Şematik diyagramlar ile oluşturulan tasarımlar, hiyerarşi ve sinyal ara bağlantısını göstermek için güçlü bir tasarım yöntemidir. Bu tür

tasarımlarda, basit mantıksal kapılardan çok işlevli blok yapılarına kadar eklenebilmekte ve düzenlenebilmektedir. Üçüncü yöntemde kullanılan IP çekirdeği veya IP bloğu, tasarım için mantık bilgilerini üretmek ve yapılandırılabilir mantık bloklarında bölümler oluşturmaktadır. IP çekirdekleri başka bir tasarım için lisanslanabilir veya tek bir tasarıma ait olabilir ve kullanılabilir. Özel bütünleşmiş devreler (ASIC) ve FPGA sistemleri tasarımcıları, IP çekirdeklerini yapı taşları olarak kullanmaktadırlar[89].



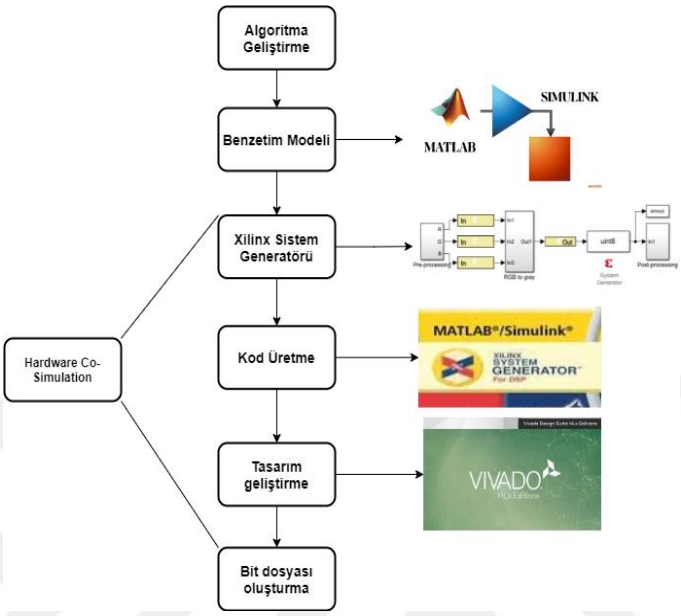
Şekil 2.5. FPGA proje tasarım töntemleri[90]

Şekil 2.5'te genel olarak tercih edilen FPGA proje tasarım yöntemleri gösterilmektedir. HDL tabanlı tasarımlar genel olarak XSG blokları veya sadece FPGA programlama dilleri (Verilog/VHDL/SystemVerilog) kullanılarak dillerini kullanarak sentezlenebilmektedir. Şematik diyagramlar ile oluşturulan tasarımlar, hiyerarşi ve sinyal ara bağlantısını göstermek için güçlü bir tasarım yöntemidir. Bu tür tasarımlarda, basit mantıksal kapılardan çok işlevli bloklara kadar eklenebilir ve düzenlenebilmektedir.

XSG-MATLAB/Benzetim platformu, ileri seviyelerde HDL bilgisi olmadan FPGA tabanlı uygulamaları doğrudan kullanılabilecek otomatik HDL kod oluşturma işlevini gerçekleştirebilmeyi sağlamaktadır. XSG kullanılarak, herhangi bir uygulamanın oluşturulması için benzetim ortamında gerçekleştirilen tasarımın tamamı JTAG donanım eş benzetimi için dönüştürülür[91].

Xilinx Sistem Üreticisi, Benzetim bloklarını kullanarak sahada programlanabilir kapı dizilerinde yüksek performanslı DSP sistemlerini modelleme ve uygulama geliştirilmesini sağlamaktadır. Xilinx benzetim blokları, dijital filtreleme, spektral analiz ve dijital iletişim için

aritmetik ve mantıksal işlemler (OR, NOT vb.), hafızaların ve DSP fonksiyonlarının gerçek modellerini içermektedir. XSG, benzetim bloklarından oluşturulmuş modelleri, sentezlenebilir VHDL/Verilog dillerinde ve FPGA platformlarında verimli bir şekilde çalıştırılmak üzere hazırlanmış blokları bir donanım uygulamasına dönüştürebilmektedir[92].



Şekil 2.6. XSG akış diyagramı

Şekil 2.6'da Xilinx sistem generatörüne ait tasarım aşamaları akış diyagramı verilmştir. Tasarımı gerçekleştirilcek algoritma oluşturulduktan sonra, MATLAB'da benzetim bloları ve XSG generatörü blokları ile tasarım benzetimi gerçekleştirilir. Bu aşamadan sonra, XSG aracılığı ile tasarıma ait otomatik kod üretimi gerçekleştirilebilmektedir. Vivado veya ISE program arayüzü ile .bit uzantılı dosya implementation aşamasından sonra oluşturulur ve FPGA platformunda çalıştırılmak üzere yükleme işlemi tamamlanarak, gerçekleştirilen tasarım benzetimten donanıma aktarılmış olur.

IP çekirdekleri önceden tasarlanmış ve önceden doğrulanmış karmaşık işlevleri yerine getiren bloklardır. FPGA da gerçekleştirilecek tasarımlarda IP çekirdeklerinin kullanımı önerilen bir uygulama geliştirme yöntemidir. Aynı mimari yapıya sahip uygulamalarla kıyaslandığında, IP çekirdekleri ile gerçekleştirilen uygulamalarda FPGA kaynak kullanımı da azalmaktadır[93]. Özelliklerine göre IP çekirdekleri üç gruba ayrılabilir.

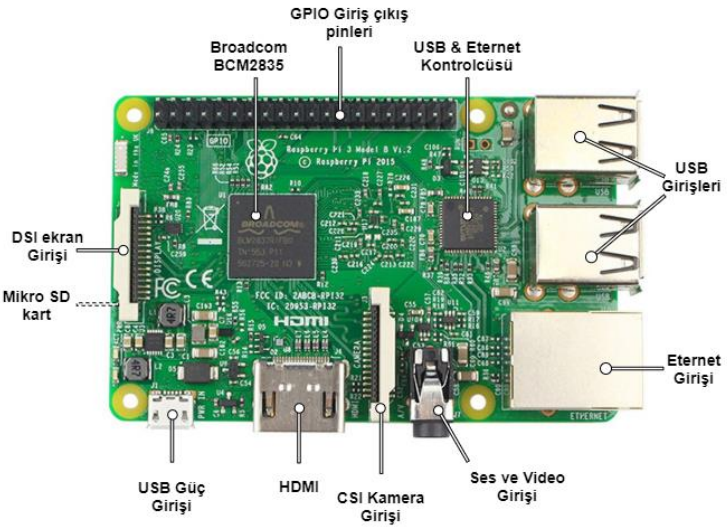
- Yumuşak (Soft) IP çekirdekler : En yüksek derecede değişiklik esnekliği ile yeniden sentezlenebilir mimari modüllerdir. Sentezlenebilir IP çekirdekleri, HDL dosyalarına (kodlamaya) bağlı olarak tasarlanabilmektedir. Bu IP çekirdekleri kulanıcıya ve tasarıma özgü olarak gerçekleştirileblen yapılardır.

- Sabit (Firm) IP çekirdekler: Tasarım akışına entegre edilen şifreli kara kutulardır. kütüphane öğeleriyle aynı şekilde sabit değiştirilemeyen IP çekirdeklerdir. Program dosyası içinde bulunmakta ve tasarıma göre mevcut yapıları değiştirilmeden kullanılabilmektedirler.
- Sert (Hard) IP çekirdekler: Sert IP çekirdekleri ara yüz programının versiyonuna ve teknolojiye bağlı modüllerdir. Daha önce tasarımcı şirket tarafından geliştirilen IP bloklarının performansı ve fiziksel özellikleri kullanıcıya sunulmuştur.

2.1.2. Raspberry Pi

Raspery Pi 32 bit ARM işlemciye sahip geliştirme kartına yüklenebilen linux işletim sistemi sayesinde pek çok yazılım dili (C/C++, Python, Java vb.) ile uygulama geliştirme imkanı sunmaktadır. Raspberry Pi'yi, ucuz, esnek, tamamen özelleştirilebilir ve programlanabilir küçük bir bilgisayar kartı ve bir bilgisayarın avantajlarını sensör ağı alanına getirebilmektedir. bu özellik onu çok çeşitli harici çevre birimleriyle arayüz oluşturmak için taşınabilir bir platform haline getirmektedir. Raspberry Pi'nin sensör ağı alanında çok başarılı kullanımı ve çeşitli araştırma uygulamaları ile ucuz bir bilgisayar olarak kaldığını göstermiştir[94].

Raspberry Pi küçük boyutlarda üretilebildiği için, Fiziksel boyut, uygulama tasarımını etkilediği için daha fazla konuma yerleştirilebilir ve daha fazla senaryoda kullanılabilmektedirler. Raspberry Pi'nin sahip olduğu CPU, matematiksel ve mantıksal işlemler yoluyla bir bilgisayar programının talimatlarını yerine getirmekten sorumlu ana bileşenidir. Raspberry Pi'nin işlemcisi, ARM mimarisi üzerine inşa edilen bir Yonga Üzerinde Sistem (SoC) yapısıdır.



Şekil 2.7. Raspbery Pi geliştirme kartı

Şekil 2.7'de Raspberry Pi 3+B geliştirme kartı gösterilmektedir. Broadcom firması tarafından üretilen ARM tabanlı BCM2835 çipi sayesinde, düşük güç tüketiminde çalışabilen

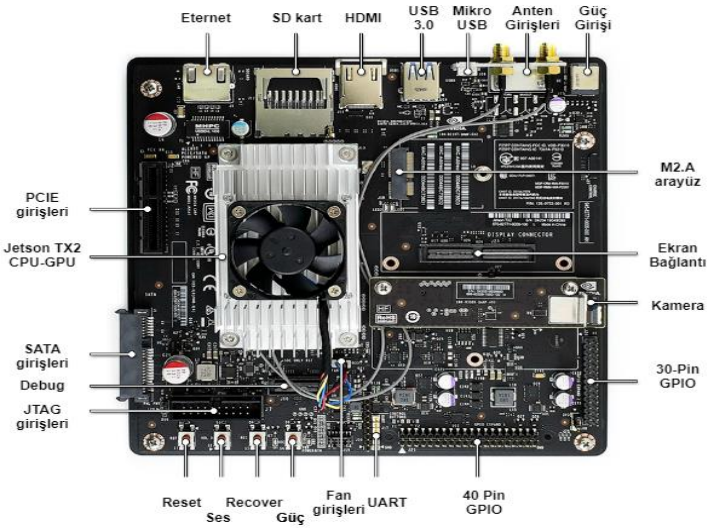
Raspberry Pi kartları, bilgisayarların USB portlarına, 5v 1A güç üretebilen harici güç ünitelerine ve dış ortamda güneş panelleri aracılığı ile çalıştırılabilmektedirler. Genel olarak Raspberry Pi;

- Bir mikro USB girişi ile Raspberry Pi çalıştırılabilir ve kartın üzerinde bulunan USB bağlantıları ile çevre birimlerinin (klavye, fare, kamera vb.) ve harici depolama aygıtlarının bağlanmasına izin vermektedir.
- Analog ses girişleri sayesinde, kulaklık veya hoparlörler Raspberry Pi'ye bağlanabilmektedir. Bu özellikler, ses ve medya oynatıcı tabanlı projelerin geliştirilmesine olanak sağlamaktadır.
- Yüksek Çözünürlüklü Çoklu Ortam Arabirimi (HDMI) bağlantısı sayesinde, Raspberry Pi'nin televizyonlara ve monitörlere bağlanmasını sağlamaktadır.
- CSI Desteği (Kamera Seri Ara yüzü) Raspberry Pi'ye direk olarak kamera bağlanabilmektedir.
- DSI Desteği (Ekran Seri Ara yüzü) sayesinde Raspberry Pi ekran bağlantısı gerçekleştirilebilmektedir.

Günümüzde robotik uyulmalarda en popüler gömülü bilgisayarlardan biri Raspberry Pi'dir ve nesne algılama vb. uygulamalarda da kullanılmaktadır. Düşük güçlü CPU'ya ve GPU'nun olmamasına rağmen Raspberry Pi, bazı CNN tabanlı uygulamalar için de kullanılmaktadır ancak 0,5-1,5 FPS kare hızlarında çalışabilmektedir[95].

2.1.3. Jetson Gömülü sistem Platformları

Yakın zamanda üretilen gömülü sistem platformları yapay zekanın gelişimi ile, derin öğrenme uygulamalarını göz önünde bulundurularak üretildiler. Bu platformlardan biri, NVidia Jetson gömülü sistem platformudur. Bu platform, nesne algılama, görüntü sınıflandırma vb. gerçek zamanlı GPU hızlandırmalı görüntü işleme uygulamaları için kullanılmaktadır. Jetson'un düşük güç tüketimi ve küçük boyutu sayesinde mobil robotik uygulamalar için uygun ve GPU'ya sahip olmasından dolayı genellikle tercih edilmektedir.



Şekil 2.8. NVIDIA Jetson TX2 geliştirme kartı

Şekil 2.8'de NVIDIA Jetson TX2 geliştirme kartına gösterilmektedir. Jetson gömülü sistem platformlarında GPU, CUDA ile programlanabilmektedir. Bu özellik sayesinde, üst düzey GPU'lar üzerinde eğitilen derin sinir ağları uygulamaları da CUDA (cuDNN) kullanılarak kodlandığından, bu uygulamalar taşınabilir sistemler (Robotlar, IHA'lar vb.) için doğrudan Jetson platformlarında kullanılabilmektedir. Raspberry Pi ve FPGA gibi diğer platformlarda, cuDNN benzeri yapılar mevcut olmadığından, derin öğrenme uygulamalarının geliştirilmesi çok kısıtlı olarak gerçekleştirilebilmektedir[96].

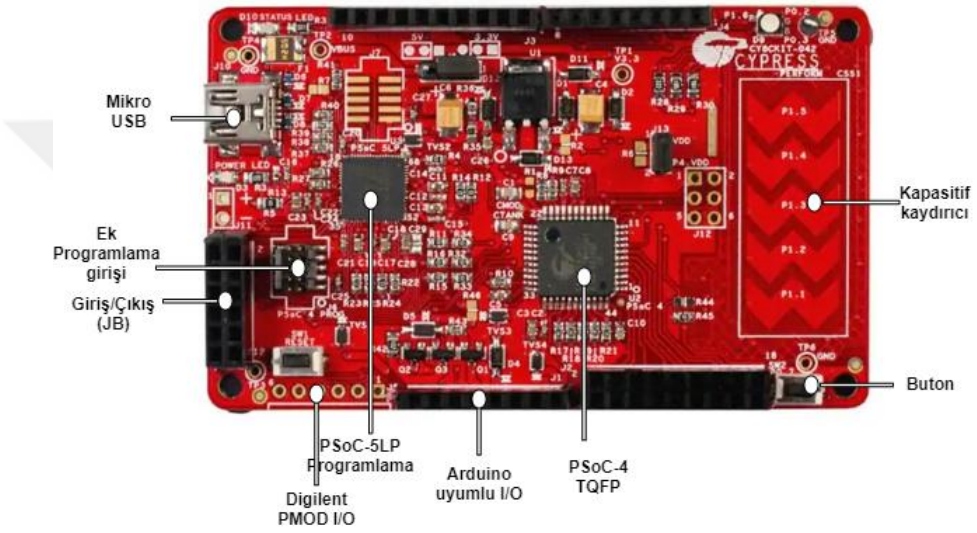
Tablo 2.3. Jetson gömülü sistem platformlarının karşılaştırılması

	Jetson-Nano	**TK1**	**TX1**	**TX2**	**AGX Xavier**
Performans	472GFLOPS	326 GFLOPS	1 TeraFLOPS	1.33TFLOPS	32 TOPS
Üretim		28nm	20nm	16nm	16nm
GPU	128 çekirdekli Maxwell	192 çekirdekli Keplerr	256 çekirdekli Maxwell	256 çekirdekli Paskal	512 Çekirdekli NVIDIA Volta GPU
CPU	Dört çekirdekli ARM Cortex-A57	2.32 GHz ARM Cortex A12	1.73 GHz ARM Cortex A57	2 GHz ARM Cortex A52+NVIDIA	Carmel ARM v8.2 64-bit CPU 8MB L2 + 4MB L3
Memory	4GB 64 bit LPDDR4	2GB 64 bit DDR3L	4GB 64 bit LPDDR4	8GB 128 bit LPDDR4	32 GB 256-bit LPDDR4x 136.5GB/s

Tablo 2.3'te Jetson gömülü sistem platformlarının karşılaştırılması verilmektedir. Performans açısından verimli ve düşük güçlü GPU çekirdeklerinden dolayı Jetson gömülü sistemleri özellikleri nedeniyle popüler hale gelmiştir. Genel olarak, GPU'lar FPGA'lardan daha yüksek verim sağladığından dolayı, Jetson gömülü sistem platformlarını derin öğrenme uygulamaları için daha uygun olmaktadır. En yüksek performans Jetson AGX Xavier modelinde elde edilmektedir. Bu modelde 32TOPS (operations per second) saniyede 32 tensor işlem gerçekleştirilebilmektedir.

2.1.4. Çip Üzerine Programlanabilir Sistem (PSoC)

PSoC (Programmable System on Chip), yeniden yapılandırılabilir analog ve dijital alt sistemleri tek bir yonga üzerinde bütünleştiren düşük maliyetli yeniden yapılandırılabilir gömülü sistem platformudur. PSoC ayrıca bir Harici Bellek ara yüzü içerir. Dahili olarak bir mikro işlemci çekirdeğinden, yapılandırılabilir analog ve dijital bloklardan, bir yazılımla yapılandırılmış ve programlanabilir yönlendirme ve ara bağlantılardan oluşan ve bir elektronik sistemin tasarımını kolaylaştıran bir yapıdır. Bu sayede, analog sinyali dijitalleştirme ve bu sinyali tek bir cihaz içinde işlemenin gerçekleştirilebileceği uygulamaların gerçekleştirilebileceği anlamına gelmektedir[97].



Şekil 2.9. PSoC 4 Geliştirme kartı

Şekil 2.9'da PSoC 4 geliştirme kartı gösterilmektedir. PSoC gömülü sistem platformları genelde; bir PSoC çekirdeği (CPU), dijital (PWM, SPI vb.) ve analog (filtreler, ADC, DAC vb.) sistem ve arabirimlerden (MAC, I2C vb) oluşmaktadır.

Tablo 2.4. PSoC gömülü sistem platformlarının karşılaştırılması

	PSoC-1	**PSoC-3**	**PSoC-4**	**PSoC-5**
ADC	6-14 bit Delta-sigma	8-20 bit Delta-sigma	12 bit SAR	8-20 bit Delta-sigma 2 ad. 12 bit SAR
DAC	2 ad. 6-8bit	4 ad. 8bit	2 ad.8bit	4 d.8-12bit
Çalışma Akımı	2mA	1.2 mA	1.6 mA	2 mA
CPU	8-bit M8C 24 MHz, 4 MIPS	8-bit 8051 CPU 67 MHz, 33 MIPS	32-bit ARM Cortex-M0 CPU 48 MHz, 100 MIPS	32-bit ARM Cortex- M3 7 MHz, 84 MIPS
SRAM	256b-2Kb	2-8Kb	4Kb	16-64Kb

Tablo 2.4'te PSoC gömülü sistemlerinin karşılaştırılması verilmektedir. Geliştirilecek uygulamaya göre; kullanılacak ADC, DAC sayısının önemi veya tasarımın kapsamına göre SRAM

seçimi gibi değişkenler PSoC gömülü sistem platformunun seçimini belirleyen faktörler arasında olacaktır.

2.2. Bölüm değerlendirmesi

Gömülü sistem platformları genel olarak, yazılım, donanım, giriş çıkış arabirimleri ve kullanıcı ara yüzlerinden oluşmaktadır. Gerçekleştirilecek uygulamaya göre, Hafıza, veri işleme hızı, boyut ve zamansal kararlılık gibi değişkenler göz önünde bulundurulmalıdır. Gerçek zamanlı ve bilgisayardan bağımsız çalıştırılabilen gömülü sistem platformlarında FPGA, Jetson ve Raspberry Pi platformları tercih edilmektedir. Farklı özelliklerle üretilen Jetson ve Raspberry Pi modellerinde, açık kaynak kütüphane fonksiyonların kullanımı, çatı yapılar ile makine öğrenmesi uygulamalarının desteklenmesi ve boyutlarının küçük olması tercih edilme nedenlerini arttıran özelliklerdendir. Ayrıca, Xilinx firması tarafından üretilen UltraScale ve UltraScale + aileleri, en yüksek performanslı FPGA platformlarıdır. Yüksek performanslı ve yüksek hızlı uygulamalar için tasarlanmışlardır. Vitis AI veya HLS4ML gibi çatı yapılar ile makine öğrenmesi uygulamaları gerçekleştirilebilmektedir.

Zaman kararlılığı yüksek olan FPGA platformları ile bilgisayar ve CPU tabanlı sistemler karşılaştırıldığında, FPGA işlemleri donanımsal olarak çok hızlı ve aynı anda bu donanımsal blokların geliştirilen uygulamalara göre paralel olarak çalıştırılabilmesini sağlamaktadır.

DSP'lerdeki kayan nokta operasyonel desteği, algoritma uygulamasını kolaylaştırır ve sabit noktaya kıyasla hassasiyeti artırır. Görüntü veya video işleme için GPU yapılarının kullanımı önerilmektedir. GPU'larda kod geliştirmek ve hata ayıklamak FPGA platformlarına göre daha hızlı ve kolaydır. Ancak, GPU'lar, aynı performans sınıfındaki FPGA platformlarına kıyasla daha fazla güç tüketirler. FPGA platformlarında programlama geliştirme için kullanılan yazılım dilleri (VHDL/Verilog) CPU'lar, DSP'ler ve GPU'lar için kullanılanlardan oldukça farklıdır.

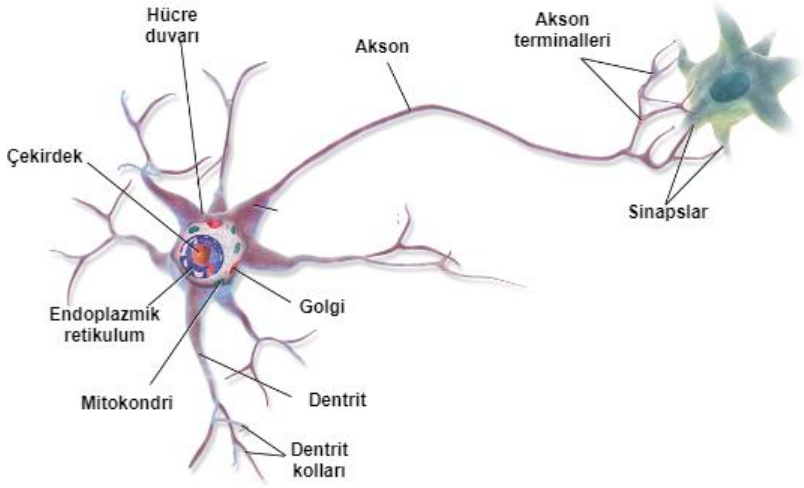
3. YAPAY ZEKÂ VE DERİN ÖĞRENME İÇİN TEMEL ALGORİTMALAR

İnsan zekâsının, makineler tarafından taklit edilebilmesi şimdilik kodlama, yapay zekâ algoritmaları ve gömülü sistemler aracılığı ile yapılmaktadır. Çünkü gömülü sistem platformları algıladıklarını işlemede insan beyninden daha kötüdür ve büyük miktarlarda veri işleme alanına ve enerjiye ihtiyaç duymaktadır[98]. Doğadaki canlıların davranışlarını yapay olarak gerçekleştirmeyi amaçlayan ve insan beynini model olarak alan yapay sinir ağları, kümeleme, tahmin etme ve sınıflandırma gibi işlemleri denetimli (Supervised) ve denetimsiz (Unsupervised) olarak yapmaktadır. Uzman görülerinden elde edilen sistemler, bulanık mantık (fuzzy), genetik algoritmalar, yapay sinir ağları ve makine öğrenmesi gibi uygulamalar yapay zekâ olarak adlandırılmaktadır[99].

Makine öğrenmesi, eğitim verilerini ve geliştirilen sinir ağı algoritmalarını kullanarak öğrenen ve gömülü sistem platformlarında otonom davranışlar sergileyen modellerdir. Genel olarak bir yapay sinir ağı modeli, girdilerden çıktılara kadar bir dizi işlemlerin yapıldığı matematiksel bir süreçtir. Bu modellerde elde edilen sonuçlara göre beklenen değerlerin karşılaştırılması ile hata miktarları elde edilmektedir. Eğitim veri kümesinde bu hatalara yanıt olarak ağın ağırlıklarını güncelleyerek hata miktarı azaltılır. Güncellemeler, bu hatayı kabul edilebilir seviyelere kadar sürekli olarak azaltmak için yapılır ve yapay sinir ağı modeli elde edilir veya öğrenme süreci takılır ve durur[100].

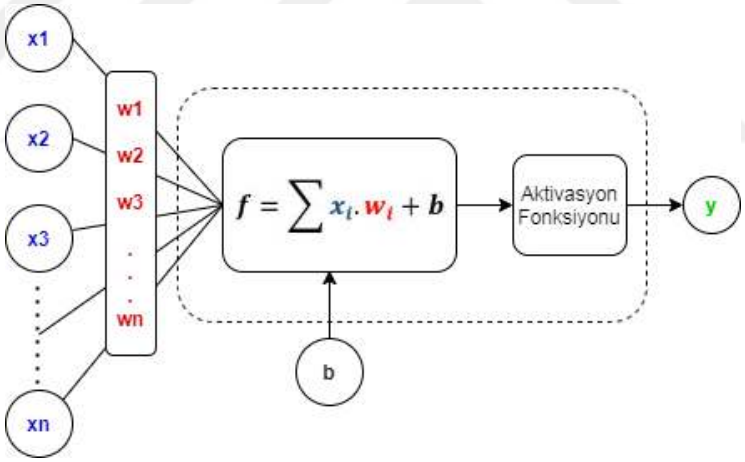
3.1. Yapay Sinir Ağı Model Yapısı ve Değişkenleri

Güdüm bilimi (Sibernetik) sayesinde doğadaki canlıların davranışlarını gözlemleyerek, benzer yapay modellerin matematiksel fonksiyonları elde edilebilmektedir[101]. Bu bilim dalı kullanılarak insan beyni yapısı öğrenme modeli oluşturulmaya çalışılmaktadır. Makinelerin yapay sinir ağı modelleri kullanılarak eğitilmesi ve öğrenmesi amaçlanmaktadır.



Şekil 3.1. Biyolojik sinir hücre yapısı [102]

Bir nöron veya sinir hücresinde akson boyunca kimyasal taşıyıcılar ile sinyaller iletilmektedir. Her hangi bir sinir hücresine ait akson ucunun (çıkış), başka bir sinir hücresinin dentritleri (giriş) ile bağlantıyı sinaps'lar aracılığı ile gerçekleştirilmektedir[102].



Şekil 3.2. Yapay sinir hücresi matematiksel modeli

Şekil 3.1Hata! Başvuru kaynağı bulunamadı.'de biyolojik sinir yapısı ve bu sinir yapısının matematiksel olarak benzetimi **Şekil 3.2**'de gösterilmektedir. Yapay sinir hücresinde bir önceki katmanın çıktıları (yapay dentrit yapısı) veya giriş katmanı (x1, x2, … xn) ile ağırlıkların (w1, w2, … wn) birbirileri ile çarpılıp (yapay sinaps) toplanması ve bias (b) değeri eklenerek aktivasyon fonksiyonundan geçirilmesi ile çıkış (akson terminalleri) elde edilmektedir.

Bir eğitim veri setine uygun bir sinir ağı modelini veya öğrenme algoritmasını oluşturmak ya da mevcut modeli iyileştirmek ve hızlandırmak için model değişkenlerinin tasarıma göre kontrol edilmesi gerekmektedir. Bu bölümde bu değişkenlerin YSA modeline etkileri açıklanacaktır.

3.1.1. Denetimli ve Denetimsiz Öğrenme

Denetimli öğrenme (Supervised), YSA modelleri sınıflandırma işlemi gerçekleştirilirken, eğitim veri setindeki her bir örneklemenin hangi sınıfa ait olduğunun bilindiği durumdur. Model parametreleri (ağırlıklar ve bias) bu bilgiye göre güncellenmektedir. Yani beklenen değerin bilindiği ve model sonucunun bu bilgiye göre değiştirildiği sistemlerdir.

Denetimsiz öğrenme(Unsupervised), verilerin kümeleme işlemleri ile bazı özelliklere göre kendi aralarında sınıflandırma veya kümeleme işlemini etiketlenmemiş (belirsiz) veriler ile gerçekleştirmektedir. Bu öğrenme modelinde geri bildirim veya hata hesaplama işlemleri bulunmamaktadır. Denetimli öğrenmede model doğruyu bulmak için eğitilir ancak denetimsiz öğrenme modelinde doğruya en yakın sınıflandırma gerçekleştirilir[103].

3.1.2. Model Öğrenme parametreleri

Gizli katmanlardaki yapay sinir hücrelerinin ve ağdaki gizli katmanların sayısını ağ topolojisi ile ifade edilmektedir. Ağ topolojisinden YSA modelinin boyutu, derinliği ve kullanılan parametrelerin miktarını genel hatları ile belirlenmektedir. Bir YSA modeli oluştururken aşağıda verilen parametrelerin veri setine ve modele uygun olarak seçilmesi gerekmektedir.

1. Eğitime girecek veriler (Batch-Size)
2. Optimizasyon
3. Hata (Loss veya Cost) fonksiyonu
4. Parametrelerin (ağırlıklar ve bias) başlangıç değerleri
5. Öğrenme oranı (Learning rate)
6. Döngü sayısı (Epoch)
7. Normalizasyon
8. Ortaklama (Pooling)
9. Seyreltme (Dropout)
10. Aktivasyon fonksiyonu

Model parametrelerini güncellemeden önce hata miktarını hesaplamak için kullanılan örneklerin sayısıdır. Kullanılacak model için her bir döngüde eğitime girecek veri miktarını temsil etmektedir. Model eğitilirken tüm verilerin eğitime dahil edilmesi eğitim sürecini yavaşlatacaktır. Bundan dolayı verilerin parçalar halinde eğitilmesi eğitimi hızlandıracaktır. Küçük kümeleme (mini-batch) yöntemini kullanarak, aynı anda ne kadar verinin işleneceği belirlenmektedir. Yani giriş verilerinin tüm veriden oluşan paketçikler ile model eğitimine iletilmesi olarak düşünülebilir. Küçük kümeler seçilirken paketçiklerin (mini-batch) boyutu ve genel olarak veriyi kapsaması gerekmektedir. Model eğitimi için tüm durumlarda en iyi sonuçlar mini-batch sayısı 32 veya daha küçük boyutlar ile elde edilmektedir[104]. Genel olarak veriyi kapsayan kümelerde hata miktarı az

veya kapsamayan kümelerde hata miktarı çok fazla olabilmektedir. Kümeleme seçimi eğitimi uzatabileceğinden, bu yöntemi kullanırken optimizasyon yöntemlerini de kullanmamız gerekmektedir[105].

Derin sinir ağları derinleştikçe ve veri kümeleri büyüdükçe bu ağların optimizasyonu daha zor hale gelmektedir. Bundan dolayı, farklı optimizasyon teknikleri geliştirilmiştir. Optimizasyon teknikleri sinir ağı modellerinin eğitimini hızlandırmak, hatayı (Loss, Cost) azaltmak ve mümkün olan en doğru sonuçları sağlamaktan için kullanılmaktadır. Genel olarak kullanılan optimizasyon teknikleri;

1. Eğim azaltma (Gradient descent)
2. Adagrad
3. AdaDelta
4. Adam
5. RMSprop

En temel ancak en çok kullanılan optimizasyon tekniği Eğim azaltma algoritmasıdır. Doğrusal regresyon (linear regression) ve sınıflandırma algoritmalarında yoğun olarak kullanılmaktadır. Eğim Azaltma, hata fonksiyonunun birinci dereceden türevine bağlı olan birinci dereceden bir optimizasyon algoritmasıdır. Fonksiyonun minimum değere ulaşabilmesi için ağırlıkların hangi yönde değiştirilmesi gerektiğini hesaplar[106]. Stokastik Eğim Azaltma (SGD: Stochastic Gradient Descent) modelin parametreleri (ağırlık ve bias) her eğitim örneğinde hata hesaplandıktan sonra güncellenir. Örneğin veri kümesi 300 elemandan meydana geliyorsa, SGD model parametrelerini 300 kez güncelleyecektir.

$$\mathrm{w} = \mathrm{w} - \nabla \mathrm{H}(w, x_i, y_i) \tag{3.1}$$

Denklem 3.1'de ağırlıkların stokastik eğim azaltma işlemi ile YSA modeli ağırlıklarının (w) güncellenmesi gösterilmektedir. ∇H Hata fonksiyonunun değişimini ve x_i, y_i eğitim verilerini temsil etmektedir.

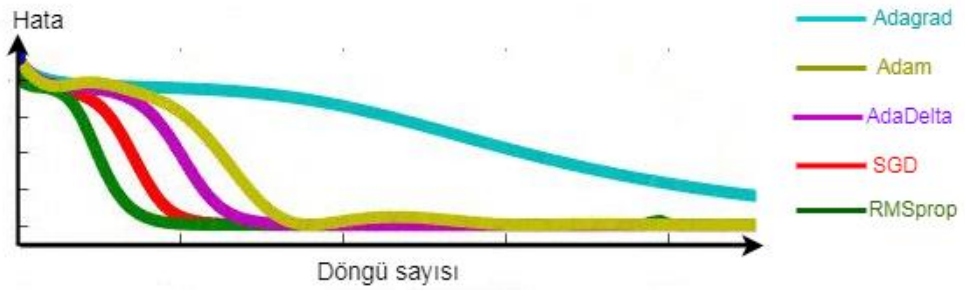
Genel olarak optimizasyon tekniklerinin dezavantajlarından biri, öğrenme katsayısının (Learning rate) sabit olmasıdır. Adagrad tekniğinde öğrenme katsayısı her parametre için ve her döngüde değiştirilir. Ancak bu teknik ikinci derecede optimizasyon işlemi gerçekleştirdiği için hesaplama süresi artmakta ve öğrenme oranı her döngüde azaltıldığı için öğrenme süresi uzamaktadır.

AdaDelta tekniğinde öğrenme oranının azaltılması problemini ortadan kaldırmak için Adagrad tekniğinin geliştirilmesi ile elde edilmiş bir optimizasyon tekniğidir. Adadelta, benzerlik oranı katsayısı (genellikle 0,9 kullanılır, momentum katsayısı) ile geçmiş hata değişimlerinin kareleri kullanılarak parametrelerin güncellenmesi gerçekleştirilmektedir.

Adam (Adaptive Moment Estimation), birinci ve ikinci dereceden momentumların kullanılması ile edilen optimizasyon tekniğidir. Bu teknik ile bir önceki hata fonksiyonunun

değişimini β_1=0.9 ve β_2=0.99 momentumları ile mevcut hesaplanan hata değişimi ile birlikte parametrelerin güncellenme işlemi gerçekleştirilmektedir.

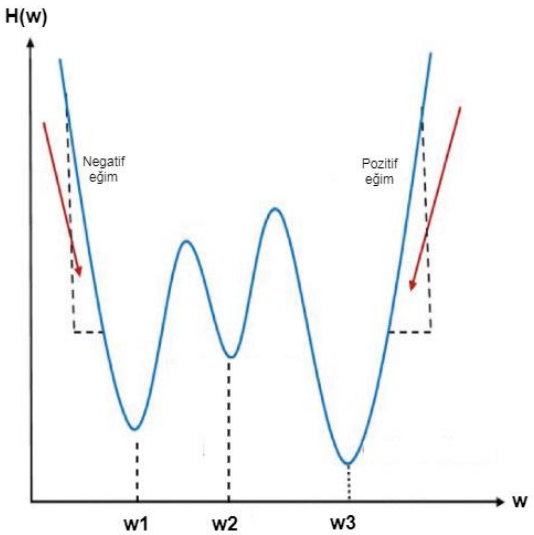
RMSprop model parametreleri güncellenirken öğrenmenin hızlandırılması için kullanılan bir tekniktir. Bu teknikte küçük kümeleme yönteminde hesaplanan hata değişim miktarının karekökünü kullanarak parametrelerin güncellenmesi işlemi gerçekleştirilmektedir.



Şekil 3.3. Optimizasyon algoritmalarının öğrenmeye etkisi[107]

Şekil 3.3'te farklı optimizasyon tekniklerinin hata miktarına bağlı artan döngü sayısına göre karşılaştırılması gösterilmektedir. Adagrad tekniğinin öğrenme sürecinin en yavaş olduğu teknik olduğu görülmektedir. Bunun nedeni her döngüde öğrenme oranının azaltılmasıdır. SGD ve RMSprop tekniklerinde hata miktarını daha hızlı azalmakta ve modelin öğrenme hızına etkisi diğer optimizasyon tekniklerine göre daha fazla olduğu görülmektedir.

Hata fonksiyonu, yapay zeka modeli eğitim veri kümesindeki parametrelerin performansını değerlendirmek için kullanılan bir fonksiyondur. Genel olarak kullanılabilen pek çok hata fonksiyonu bulunmaktadır. Veri setinize ve modelinize uygun fonksiyonun seçilmesi eğitimi hızlandıracak ve model doğruluk oranını arttıracaktır. Sinir ağı modellerini eğitirken, çapraz entropi (Cross-entropy) ve ortalama kare hatası (MSE: Mean Square Error) genel olarak kullanılan iki ana hata fonksiyonudur[108].



Şekil 3.4. Hata ve ağırlığın karşılaştırılması

Güncellenen hata değerinin ağırlık ile karşılaştırılması **Şekil 3.4**'te verilmiştir. Burada w1 ve w2 ile gösterilen noktalarda ağırlıklar kısmi olarak en düşük değeri göstermektedir. Fakat bu noktalarda eğitim durabilir ve modelin doğruluk oranı düşük seviyelerde olacaktır. Grafikte w3 ile gösterilen ağırlık değeri ise tasarlanan modele göre en düşük ağırlık değerini temsil etmekte ve istenilen değerdedir. Bu noktaya evrensel, w2 ve w3'ün bulunduğu noktalar ise bölgesel minimum denmektedir. Hatanın eğimi eğer pozitif ise ağırlıklar azalır, negatif eğime sahip ise ağırlık değeri büyür.

Ortalama kare hatası denetimli öğrenme modellerinde kullanılan hata fonksiyonudur. Bu fonksiyon model eğitim sürecinde beklenen değer (y) ile hesaplanan değerin ($\hat{y}$) farkının toplamlarının ortalamasına eşittir. Bu eşitlik denklem 3.2'de verilmektedir.

$$\mathrm{MSE} = \frac{1}{2.n}\sum(y - \hat{y})^2 \tag{3.2}$$

Çapraz entropi hatası, logaritmik hata fonksiyonu olarak da bilinmektedir. Modelin çıkışında tek sınıf var ise binary çapraz entropi veya çoklu sınıflandırma işlemi yapılıyorsa kategorik çapraz entropi hatası hesaplanır.

$$\mathrm{ÇE} = -\frac{1}{n}\sum_i(y_i.\log(\hat{y}_i) + (1 - y_i).\log(1 - \hat{y}_i)) \tag{3.3}$$

Eğitim sürecinin başında model parametrelerine küçük rasgele değerlerin verilmesi aşamasıdır. Her bir nöron farklı bir özelliği öğreneceğinden ağırlıkların giriş değeri rastgele verilmektedir. Model eğitim sürecinde ağırlıkların değeri sıfırdan farklı seçilmelidir. Eğer bu değerler sıfır ile başlatılırsa nöronlar güncellenemeyecektir.

Başlangıç parametrelerinin seçimi yapılırken girişe rastgele değerler vermek yerine Transfer öğrenme metodu da kullanılabilmektedir. Transfer öğrenme tekniği kullanılırken, daha önce benzer problemlerde kullanılmış model parametrelerini yeni eğitilecek modele uygulayarak başlangıç koşulları ve model parametreleri belirlenmektedir. Bu yöntemi sinir ağlarının eğitimini hızlandırmak için kullanılmaktadır[109].

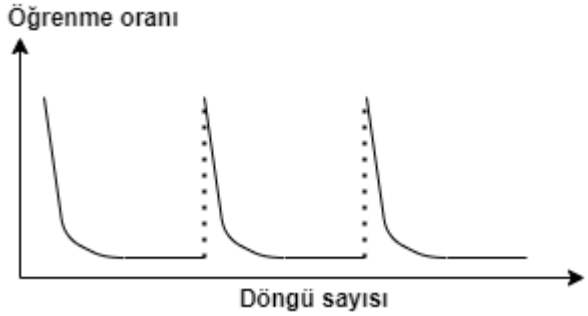
Model öğrenme sürecinde iyi performans elde etmek için modeldeki değişiklik miktarı, adım boyutu veya öğrenme hızı olarak da adlandırılır ve en önemli hiper parametrelerden biridir. Modelin öğrenme sürecinde her yineleme için parametrelerin güncellenme miktarını belirleyen katsayıdır ve bir sinir ağı modelinin bir sorunu ne kadar hızlı veya yavaş öğrendiğini kontrol eder.

Öğrenme oranının çok büyük seçilmesi modelin öğrenmemesini sağlayacaktır. **Şekil 3.4**'te bu durumu temsil etmek istersek eğer, eğim bir pozitif tarafta, bir negatif tarafta olacaktır ve osilasyona girip parametrelerin güncellenmesi sağlanamayacaktır. Bunun aksine öğrenme oranının çok küçük seçilmesi de öğrenme sürecini çok yavaşlatacaktır. Öğrenme işleminin gerçekleşmesi için çok fazla döngüye ihtiyaç duyulacaktır ve öğrenme için çok fazla zamana ihtiyaç duyulacak ve enerji tüketimi artacaktır.

Öğrenme hızındaki azalma ile öğrenme oranı, her güncellemede (örn. her mini-batch grubun sonu) aşağıdaki gibi hesaplanabilir.

$$LR = iLR * \frac{1}{1+dr*do} \quad (3.4)$$

Denklem 3.4'te *LR* Öğrenme oranı, *iLR* öğrenme oranı ilk değeri, *dr* azalma miktarı ve *do* döngü sayısını temsil etmektedir.



Şekil 3.5. Öğrenme oranının döngü sayısına göre güncellenmesi

Şekil 3.5'te öğrenme oranının döngü sayısına göre değişimi gösterilmektedir. Öğrenme oranı döngü sayısına göre azaltılabilir ve sonra tekrardan ilk değerine getirilip yeniden azaltılarak model öğrenme performansı arttırılabilir[110].

Bir Epoch, veri kümesinin yalnızca bir kez sinir ağı üzerinden ileri ve geri aktarılmasıdır. Eğitim veri setini kullanarak modelin eğitimi için güncellenen parametrelerin meydana geldiği sürecin tekrarlama sayısını temsil etmektedir. Döngü sayısının az olması durumunda model eğitimi gerçekleşmemiş veya döngü sayısının fazla olması eğitim süreci bitmiş model için zaman ve güç kaybı olacaktır. Epoch ve iterasyon kavramları birbirinden farklıdır. Örneğin 1000 adet verinin bulunduğu bir veri setini 50'lik küçük kümeleme (mini-batch) gruplar halinde böldüğümüzde, 1 Epoch tamamlanması için 20 iterasyonun tamamlanması gerekmektedir ve her bir döngüde 50 defa veriler güncellenmektedir.

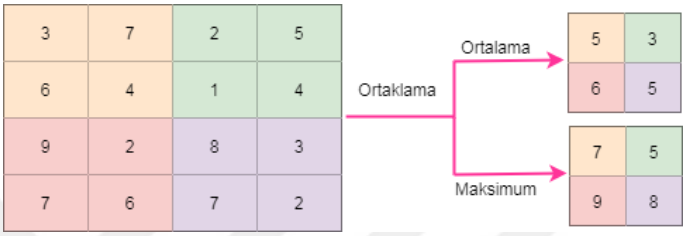
Bir sinir ağı modelini eğitmeden önce veri hazırlama veya yeniden ölçeklendirmek normalleştirme ve standardizasyon gibi tekniklerin kullanılmasını içermektedir. Ölçeklendirilmemiş giriş değişkenleri, yavaş veya dengesiz bir öğrenme sürecine neden olabilmektedir. Veri setinde bulunan her bir özelliğin birimi farklı olabileceğinden (kilometre, kilogram, saniye vb.) ölçeklendirme veya normalizasyon yapılmayan verilerde eğitilen parametrelerin çok büyük olmasını sağlayabilmektedir. Bu yüzden model eğitim öncesinde veri ölçeklendirme işleminin yapılmasını gerektirmektedir.

Normalizasyon işlemi giriş, çıkış verilerine yapıldığı gibi aynı zamanda gizli tabakalarda da yapılmaktadır. Bu tekniklerden biri de Batch-Normalizasyon yöntemidir. Bu tekniği kullanarak, derin ağları eğitmek için gereken öğrenme sürecini stabilize ederek ve döngü sayısını önemli ölçüde azaltma etkisine sahiptir. Bu yöntem değişken kayması (internal covariate shift) problemini de

ortadan kaldırmaktadır ve aynı zamanda karmaşık sınıflandırma işlemlerini de düzenlemektedir. Örneğin kutup ayısı ve boz ayıyı birbirinden ayrı sınıflara ait olduğunu belirleyebilmektedir[111].

Evrişimli sinir ağı modellerinde kullanılan boyut (özellik) indirgeme işlemini gerçekleştiren bir tekniktir. Ortaklama katmanında, evrişim katmanından sonra elde edilen matrislerin her küçük bölgesinde bilgi toplamak için ortaklama operatörü uygulanarak elde edilir. Genellikle kullanılan iki tip ortaklama tekniği bulunmaktadır[112,113].

- Ortalama (Average-Pooling) Ortaklama
- Maksimum (Max-Pooling) Ortaklama



Şekil 3.6. Ortalama ve maksimum ortaklama tekniği

Şekil 3.6'da Ortalama ve maksimum ortaklama tekniği gösterilmektedir. Bu teknik ile ortaklama matrisinin bulunduğu bölgedeki değerlerin ortalama değeri alınır veya maksimum değeri seçilerek ortaklama işlemi gerçekleştirilmiş olur. Ortaklama matrisinin boyutuna göre bir sonraki tabakanın boyutu indirgenir.

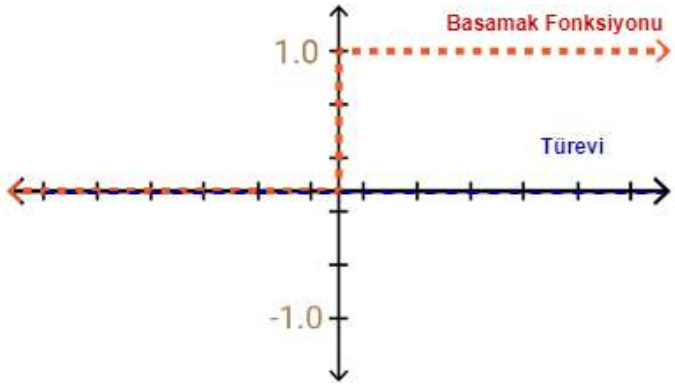
Bir sinir ağı modelinin eğitim aşamasında bazı nöronların aktif, bazılarının ise pasif olarak eğitim sürecinin gerçekleştirilmesi işlemidir. Bu teknik genellikle çok derin sinir ağı modellerinde katmandaki nöronları seyrelterek öğrenme hızını arttırmak ve aşırı öğrenme (overfit) olayını engellemek için kullanılmaktadır[114]. Seyreltme işleminin kullanılmasındaki amaç bazı nöronları pasif ederek model eğitimini hızlandırmak amaçlanmıştır.

YSA model öğrenme sürecinde aktivasyon fonksiyonlarının seçimi öğrenme sürecine yardımcı olur ve ayrıca doğrusal olmayan özelliklerin sınıflandırılmasını sağlamaktadır. Aktivasyon kelimesi bu fonksiyonların çıkışına bakarak bir nöronun aktif olup olmadığını belirlemesi işleminden gelmektedir. Model öğrenme sürecinde, geri yayılım kısmında aktivasyon fonksiyonlarının türevinin sıfır olması istenilmeyen durumlardan biridir. Türevin sıfır olması durumunda parametrelerin güncellenmesi gerçekleşmeyecektir. Aktivasyon fonksiyonlarında üstel fonksiyonların bulunması da diğer dezavantajlardan biridir. Hesaplanan yüksek değerlerde üstel ifade sonsuza gideceğinden eğitim sürecinin durdurulmasına neden olacaktır.

Basamak fonksiyonu geriye yayılım algoritmasında türevinin sıfır olmasından dolayı nöronların güncellenmesini sağlayamayacağı için basit ve sadece ileri yayılımlı modellerde kullanılabilir ama genellikle tercih edilmektedir.

$f(x) = 0, x < 0$ Ve $f(x) = 1, x > 0$ (3.5)

$f(x)' = 0, x < 0$ Ve $f(x)' = 0, x \geq 0$ (3.6)



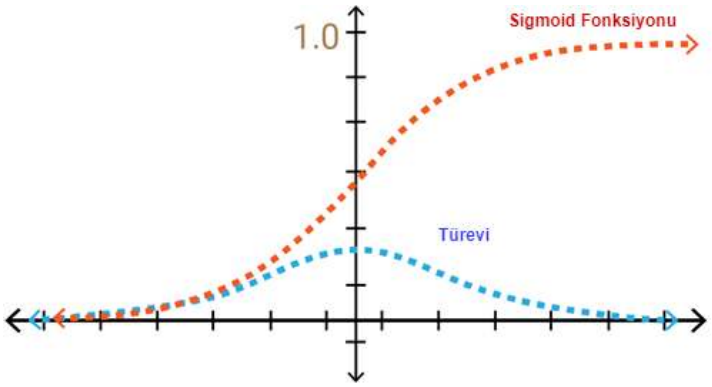
Şekil 3.7. Basamak fonksiyonu ve türevi

Şekil 3.7'de basamak fonksiyonu ve türevinin grafiği, denklem 3.5 ve 3.6'da basamak fonksiyonunun matematiksel ifadesi gösterilmektedir.

Sigmoid aktivasyon fonksiyonu 0-1 sayı aralığında değerler üretebilen "S" şeklinde bir eğriye sahiptir. Bu fonksiyonun dezavantajı belli bir değerden sonra doyuma ulaşmasıdır. Doyum nedeni ile eğimi sıfıra gideceğinden geriye doğru yayılmada parametrelerin güncellenememesi probleminden dolayı modelin öğrenme süresi uzamaktadır. Bu olaya yok olan gradyan denilmektedir.

$$f(\mathrm{x}) = \frac{1}{1+e^{-x}} \quad (3.7)$$

$$f(\mathrm{x})' = \left(\frac{1}{1+e^{-x}}\right) * \left(1 - \frac{1}{1+e^{-x}}\right) \quad (3.8)$$



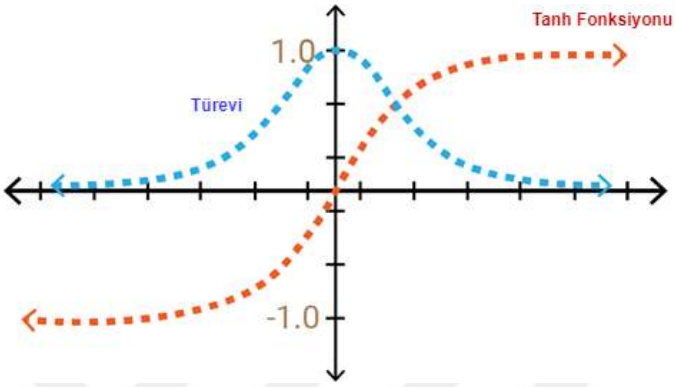
Şekil 3.8. Sigmoid aktivasyon fonsiyonu ve türevinin grafiği

Şekil 3.8'de sigmoid aktivasyon fonksiyonu ve türevinin grafiği, denklem 3.7 ve 3.8'de Sigmoid fonksiyonunun matematiksel ifadesi gösterilmektedir.

Hiperbolik tanjant fonksiyonu -1, +1 sayı aralığında sigmoid fonksiyonuna benzer sonuçlar üretmektedir. Üstel fonksiyon yapısında olduğu için sigmoid fonksiyonun sahip olduğu dezavantajlar bu fonksiyonda da mevcuttur.

$$f(\mathrm{x}) = \frac{1-e^{-2x}}{1+e^{+2x}} \tag{3.9}$$

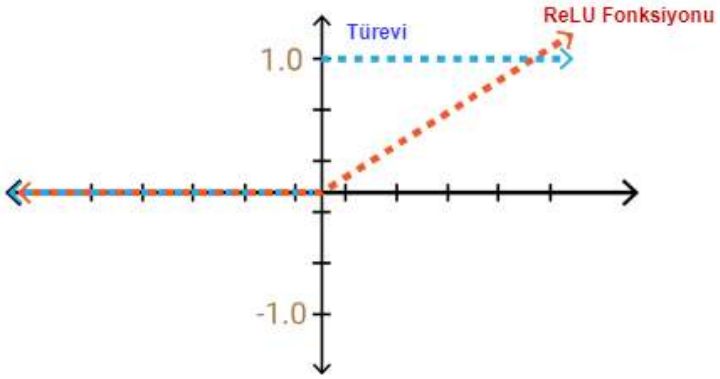
$$f(\mathrm{x})' = \frac{4}{(e^{+x}+e^{-x})^2} \tag{3.10}$$



Şekil 3.9. Hiperbolik tanjant aktivasyon fonsiyonu ve türevinin grafiği

Şekil 3.9'da hiperbolik tanjant aktivasyon fonksiyonu ve türevi, denklem 3.9 ve 3.10'da Hiperbolik Tanh fonksiyonunun matematiksel ifadesi gösterilmektedir.

ReLU aktivasyon fonksiyonu negatif değerlerde sıfır değerini ve pozitif değerlerde ise değerin aynısını çıkışa iletmektedir. Bu teknik kullanılarak model öğrenme hızı arttırmaktadır fakat ileri yayılım sürecinde negatif değerlerin etkisi olmayacaktır ve geriye yayılım sürecinde bu negatif nöronlarda yok olan gradyan etkisi meydana gelecektir. Hesaplama açısından işlem yükü sağlamadığından ve evrensel minimum değerine sigmoid ve tanh'tan daha hızlı ulaştığı için tercih edilmektedir.



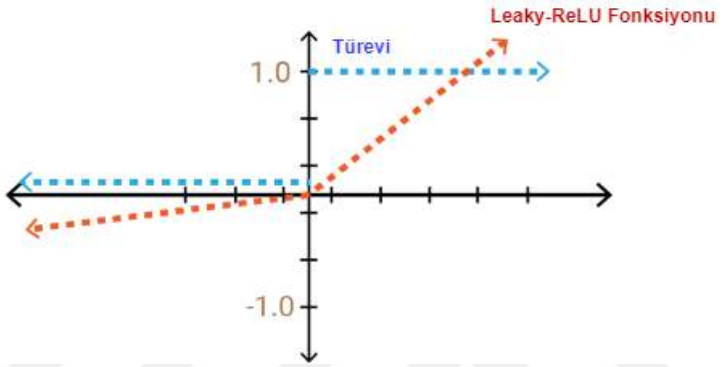
Şekil 3.10. ReLu aktivasyon fonsiyonu ve türevinin grafiği

Şekil 3.10'da ReLU aktivasyon fonksiyonu ve türevi gösterilmektedir. ReLU fonksiyonu sıfırdan büyük sayılarda; sayının kendisini ve sıfırdan küçük sayılarda ise çıkışa sıfır değerini iletmektedir.

Leaky-ReLu aktivasyon fonksiyonu pozitif değerlerde ReLu fonksiyonuna benzemektedir fakat negatif değerlerde ise değer belli bir katsayı (genellikle 0.01x) ile çarpılmaktadır. Böylelikle yok olan gradyan etkisi ortadan kalkmakta ve aynı zamanda ReLu fonksiyonu gibi hızlı işlem gerçekleştirilebilmektedir.

$$f(x) = 0{,}01 * x,\ x < 0 \quad (3.11)$$

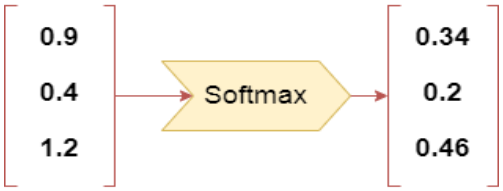
$$f(x) = x,\ x > 0 \quad (3.12)$$



Şekil 3.11. Leaky-ReLu aktivasyon fonsiyonu ve türevinin grafiği

Şekil 3.11'de Leaky-ReLU aktivasyon fonksiyonu ve türevi, denklem 3.11 ve 3.12'de Leaky-ReLU fonksiyonunun matematiksel ifadesi gösterilmektedir.

Softmax, üstel fonksiyonlardan oluşan ve girişlerin tüm değerleri arasında (pozitif ve negatif) birbirileri ile ilişkisini olasılıksal olarak veren aktivasyon fonksiyonudur. Bu aktivasyon fonksiyonunun çıkışı 0-1 değer aralığındadır ve çıkışların toplamı bire eşittir. Bu özelliklerinden dolayı genellikle çıkış katmanında ve birden fazla sınıflandırma işlemi gerçekleştirilen modellerde kullanılmaktadır. Çıkış katmanında tek bir nöron bulunan modellerde kullanılmaz, çünkü çıkış sürekli bir değerine eşit olacaktır ve model öğrenmeyecektir. Bu yüzden sigmoid fonksiyonunun kullanılması önerilmektedir.



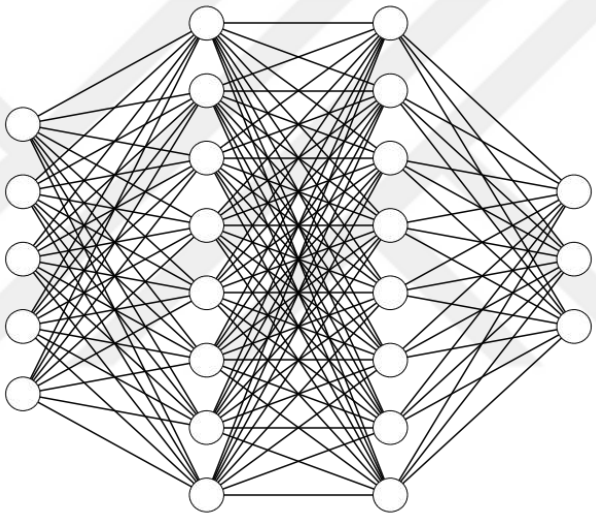
Şekil 3.12. Softmax aktivasyon işlemi

Şekil 3.12'de sigmoid fonksiyonunun kullanılması ile elde edilebilecek sonuçların gösterilmesi için örnek verilmiştir.

3.1.3. Nöron ve Gizli Katmanlar

Bir sinir ağı modelinin kapasitesi gizli katmanların sayısı ve bu katmanlarda bulunan nöronların miktarına bağlıdır. Bir YSA modelinin uygun şartlar altında eğitiminin sağlanması için kullanılan tekniklerden biri de nöron ve gizli katman sayısını değiştirmektir. Çok az kapasiteye sahip bir model eğitim veri setini öğrenemeyebilir (underfit) veya çok fazla kapasiteye sahip bir model de eğitimi ezberleyebilir (overfit)[115].

YSA'ların tasarımı gerçek bir beynin biyolojik yapısından ilham alınarak birçok sinir ağı türü kullanılarak oluşturulmuştur. Ancak temel olarak benzer ilkeler içermektedirler. Ağdaki her nöron giriş sinyallerini işler ve bir çıkış sinyali üretir. Her nöron en az bir nöronla bağlantılıdır ve her bağlantı, nöral ağda verilen bağlantının önem derecesini yansıtan ağırlık katsayısı (w) ve bias (b) adı verilen sayılar ile gerçekleştirilmektedir.



Şekil 3.13. Yapay sinir ağı modeli

Şekil 3.13'te yapay sinir ağı modeli gösterilmektedir. Bu model giriş katmanında beş, gizli katmanların her birinde sekizer ve çıkış katmanında üç adet nöron bulunmaktadır. En basit YSA yapısında tek katman bulunmaktadır. Kullanılacak eğitim veri seti eğer kolaylıkla birbirinden ayrılabiliyor ise tek katmanlı YSA yapıları kullanılabilir. Fakat beyin hücrelerinin modellemesini kullanmak ve sınıflandırma işlemi gerçekleştirmek için çok katmanlı YSA modellerinin kullanılması gerekmektedir. Ayrıca lineer olmayan sınıflandırma işlemlerinde de çok katmanlı YSA modelleri kullanılmaktadır[116].

3.2. Evrişimli Sinir Ağları

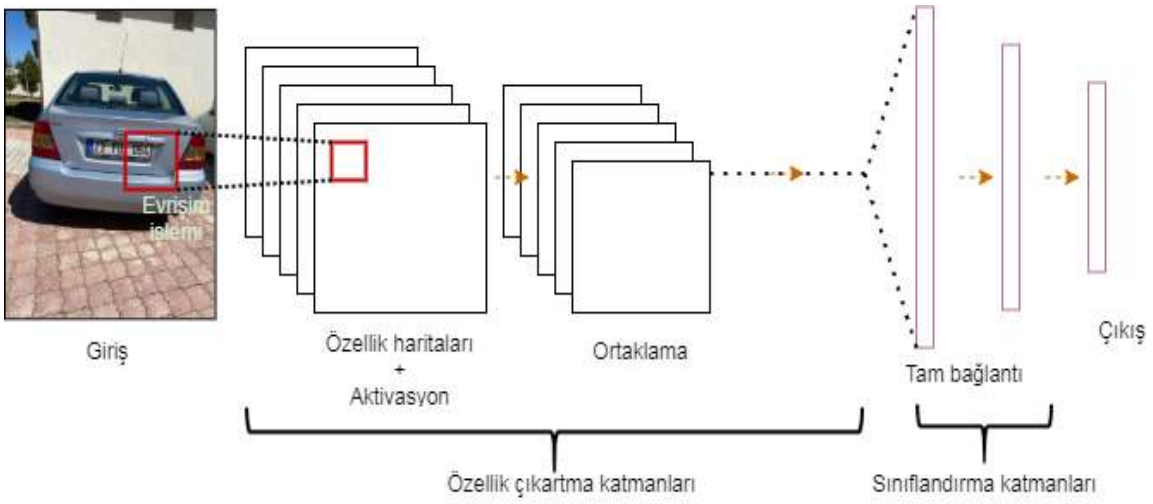
Evrişimli sinir ağı yapısı ilk olarak Fukushima ve çalışma grubu tarafından[117] kullanılmaya başlandı fakat bu modelin öğrenme algoritması çok karmaşık olduğundan tercih

edilmemiştir. Daha sonra LeCun[118] tarafından geriye yayılım algoritması ile evrişimli sinir ağının kullanımı yaygınlaşmıştır. LeCun'un oluşturduğu bu evrişimli sinir ağı LeNet-5 ilk olarak rakam tanıma için kullanıldı. Daha sonra geliştirilen evrişimli sinir ağları genellikle görüntü sınıflandırması (hayvanlar, arabalar, sebzeler vb.) nesne algılama, nesne tanıma, optiksel karakter tanıma, yüz tanıma ve plaka tanıma gibi uygulamalar için kullanılmaktadır.

Evrişim işleminde ağırlıklar nxn boyutunda filtre matrisleri ile tüm pikseller için ortak olarak kullanıldığı için tasarımı yapılan öğrenme modelinde kullanılan parametre sayısı da azalmaktadır. Bu durum evrişimli sinir ağlarının üstün yönlerindendir. Evrişim işleminin ardından oluşan her bir matrise özellik (Feature Map) veya aktivasyon haritası denilmektedir. Bu özellik haritaları girişe uygulanan filtre matrisleri ile elde edilmektedir. Özellik algılayıcıları veya filtreler, bir görüntüde bulunan kenarları, dikey çizgileri, yatay çizgileri, kıvrımları vb. farklı özellikleri belirlemeye yardımcı olur. Genel olarak evrişimli sinir ağı modellerinde,

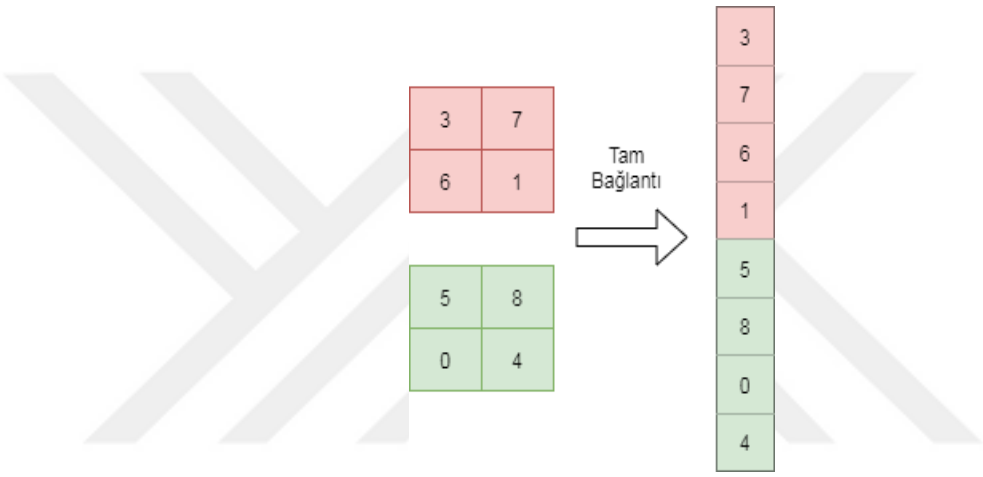
- ✓ Evrişim katmanı
- ✓ Normalizasyon katmanı (batch-Norm)
- ✓ Aktivasyon katmanı
- ✓ Ortaklama (maksimum veya ortalama)
- ✓ Tam bağlantı katmanları (çok katmanlı yapay sinir ağı) gibi katmanlar bulunmaktadır.

Evrişimli sinir ağları insan görüşünü modellemek için ve insan beyni tepkileri ile genel uyuşmaları nedeniyle nesne sınıflandırma uygulamalarında daha fazla kullanılmaktadır[119]. Evrişimli sinir ağları gerçekleştirilmesi planlanan uygulamalara göre farklı tasarımlarla kullanılabilmektedir. Genel olarak, görsel tanıma için Evrişimli Sinir Ağları (CNN, R-CNN[120] vb.) veya ResNet (Deep Residual Learning), konuşma tanıma için RNN (Recurrent Neural Networks) veya Derin Sinir Ağları (DNN) ve diyaloğu anlamak için RL (Represent Learning) gibi teknikler genel olarak kullanılmaktadır[98].



Şekil 3.14. Evrişimli sinir ağı yapısı

Şekil 3.14'te genel olarak kullanılan evrişimli sinir ağı (ESA) mimarisi bulunmaktadır. Giriş resmi ile model parametrelerine (filtreler) evrişim işleminin sonucunda özellik haritaları (matrisleri) elde edilmektedir. Bu matrislere aktivasyon, seyreltme ve normalizasyon gibi işlemler uygulanarak bir sonraki katmana ait girdi matrisleri oluşturulmaktadır. Bu işlemler model tasarımına bağlı olarak gerçekleştirilmesi planlanan evrişim katmanlarının sayısı kadar tekrar yapılmaktadır. Evrişim katmanlarının sonucunda elde edilen matrisler 1xn boyutlarına dönüştürüldükten sonra tek bir matris haline getirilmektedir. En son 1xm boyutunda elde edilen bu matrise tam bağlantılı (Fully connected) matris denilmektedir. Tam bağlantı matrisi dönüşümü **Şekil 3.15**'te gösterilmektedir. Bu aşamadan sonra yapay sinir ağlarında meydana gelen süreçler burada da uygulanıp sınıflandırma işlemi gerçekleştirilmektedir.



Şekil 3.15. Tam bağlantı dönüşümü

Bir ESA mimarisinde özellik çıkartma katmanlarında (Feature Learning) seyreltme (Dropout) kullanılmamaktadır bunun yerine normalizasyon işlemleri (Batch Normalization vb.) tercih edilmektedir. Sınıflandırma katmanlarında ise model eğitim sürecini hızlandırmak için seyreltme işlemi genel olarak kullanılmaktadır[121].

3.3. Bölüm değerlendirmesi

Gerçekleştirilecek uygulamaya göre derin öğrenme modellerinin tasarımı farklılaşabilmektedir. Model eğitim sürecinde eğitilecek verilerin çok yüksek miktarlarda hazırlanması gerekmektedir. Ancak, veri seti miktarlarının arttırılamayacağı uygulamalarda alternatif olarak, daha önceden büyük veri setleri ile yüksek doğruluk oranlarında eğitimi tamamlanmış modellerin parametreleri kullanılarak transfer öğrenme ile model geliştirme süreci hızlandırılabilmektedir.

Derin öğrenme modeli oluştururken aşağıda verilen parametrelerin veri setine ve modele uygun olarak seçilmesi gerekmektedir.

- Eğitime girecek veriler (Batch-Size)
- Optimizasyon
- Hata (Loss veya Cost) fonksiyonu
- Parametrelerin (ağırlıklar ve bias) başlangıç değerleri
- Öğrenme oranı (Learning rate)
- Döngü sayısı (Epoch)
- Normalizasyon
- Ortaklama (Pooling)
- Seyreltme (Dropout)
- Aktivasyon fonksiyonu

Veri seti, eğitim aşamasında küçük gruplar (mini-batch) halinde eğitilmesi model öğrenme hızını arttıracaktır. Optimizasyon tekniği, hata fonksiyonu ve öğrenme oranının seçimi model parametrelerinin güncellenmesinde önemli rol oynamaktadır. Normalizasyon tekniklerinin kullanımı, eğitimin daha yüksek doğrulukta ve hızda gerçekleşmesini, aşırı öğrenmenin engellenmesini sağlayacaktır. Aktivasyon fonksiyonunun seçiminde, kaybolan gradyan etkisini ortadan kaldırmak için Leaky-ReLU tekniği kullanımı önerilmektedir. Çıkış katmanında, tekli (binary) sınıflandırma için sigmoid ve birden fazla sınıflandırma işlemi için softmax fonksiyon önerilmektedir.

4. TEMEL GÖRÜNTÜ İŞLEME UYGULAMALARI

Günümüzde insanların yerini alan otonom sistemlerin hemen hemen her alanda kullanılması ile gömülü sistemler ve görüntü işleme uygulamalarının önemi giderek daha fazla artmaktadır. Görüntüler, üç boyutlu nesnelerin iki boyuta gösterilmesi olarak tanımlanmaktadır. FPGA, mantık blokları ve ara bağlantılar gibi mantık bileşenlerinden oluşan kullanıcının tasarımına bağlı olarak içyapısının değiştirilebildiği yeniden programlanabilen bir gömülü sistem platformudur. Bunun dışında, daha karmaşık mantıksal işlemleri gerçekleştirmek için yapılandırılabilir mantık blokları (CLB: Configuration Logic blocks), herhangi bir girdi için çıkışın ne olduğunu belirleyen tablolar (LUT: LookUp Table), ve hafıza birimlerinden (Blok Ram vb.) oluşur.

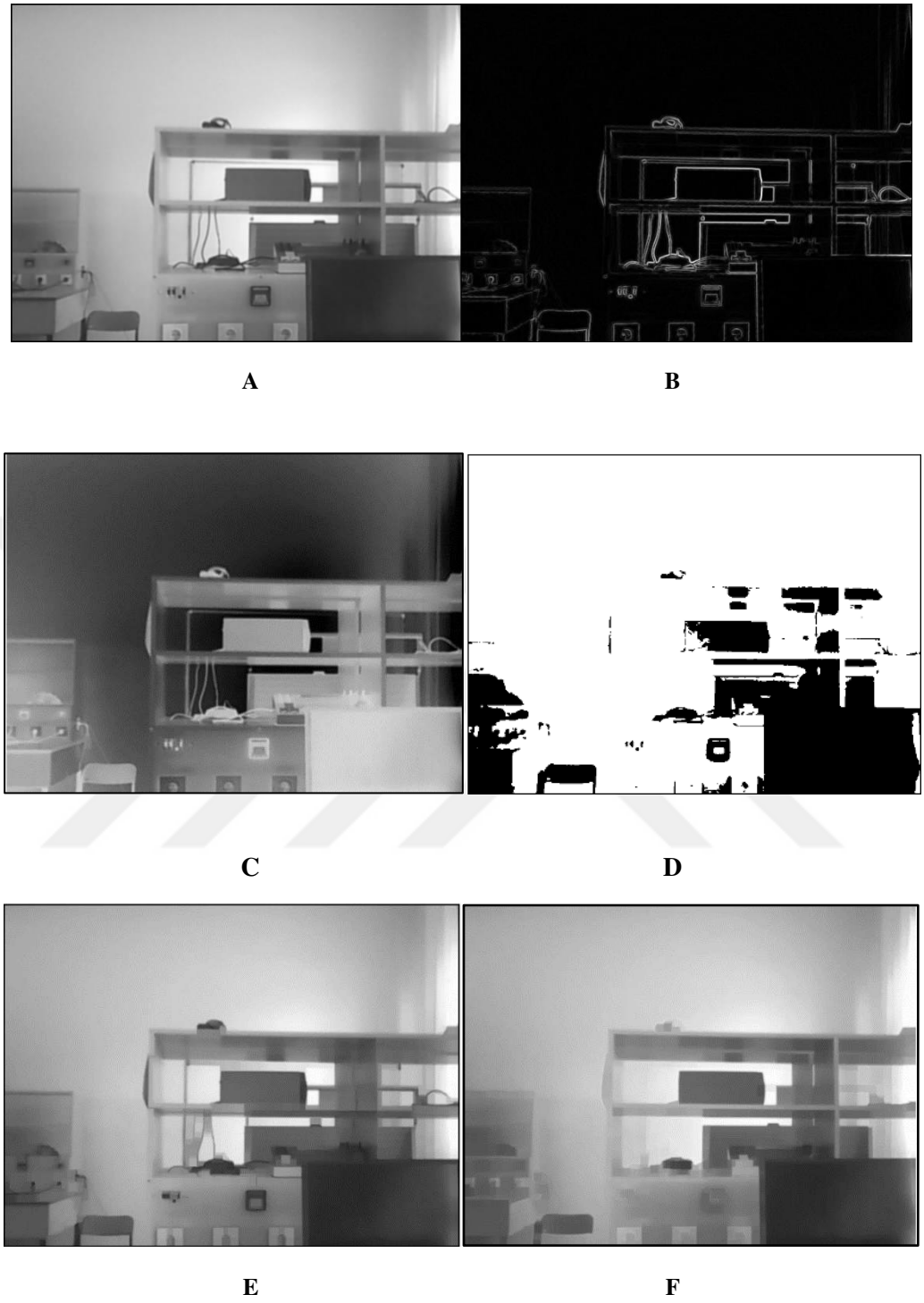
Görüntü işleme için gerçekleştirilecek uygulamalarda kullanılabilecek kütüphanelerin gömülü sistem platformunda bulunması çok önemli bir yer tutmaktadır. Görüntü işlemeye ait bu kütüphaneler, birden fazla adımda gerçekleştirilecek işlemleri tek aşamada, daha hızlı ve daha az karmaşık yapılar ile uygulamalar gerçekleştirebilme olanağı sağlamaktadır. Bu durum zamandan kazanç sağlayıp güç tüketimini ve işlem yükünü azaltmaktadır. Gömülü sistem platformlarında farklı uygulamalar için kullanılabilecek pek çok görüntü işleme kütüphaneleri bulunmaktadır. Bu kütüphaneler arasında en yaygın olarak, Intel firması tarafından geliştirilen, OpenCV (Open Source Computer Vision) açık kaynak ve ücretsiz olarak pek çok yazılım dilinde (C++, Python, Java vb.) kullanılabilmektedir[122].

4.1. Raspberry Pi Kullanılarak Gerçekleştirilen Uygulamalar

Raspberry Pi, Broadcom firması tarafından üretilen ARM tabanlı BCM2835 çipi sahip olduğu USB portları aracılığıyla depolama aygıtları, klavye, mouse, kamera vb. bağlanabilen, HDMI portu ile bir monitöre veya TV'ye bağlanabilen, Ethernet portu veya Wifi bağlantısı ile bir ağa bağlanabilen, üzerinde güçlü bir işlemci ve ram barındıran bir bilgisayardır. Bu özelliklerinden dolayı görüntü işleme alanında kullanılan gömülü sistem platformları arasında yer almaktadır.

4.1.1. Görüntü Filtreme Tekniklerinin Uygulanması

Görüntü filtreleme işlemleri, görüntü işlemede her zaman önemli bir rol oynamaktadır. Gerçekleştirilecek uygulamalar çerçevesinde genellikle gürültü giderme, boyut indirgeme, renk değişimleri, kenar algılama gibi uygun bir filtre kullanılarak görüntü ön işleme gerçekleştirilebilir ve asıl gerçekleştirilecek ana uygulama işlemleri aşamasındaki birçok zorluğu da ortadan kaldıracaktır.



Şekil 4.1. Raspberry Pi ile gerçek zamanlı video görüntüsüne filtre uygulamaları, (A) gauss filtre, (B) sobel kenar algılama, (C) görüntünün tersi, (D) siyah-beyaz görüntü, (E) Morfolojik açma, (F) morfolojik kapama sonuçları

Şekil 4.1'de gerçek zamanlı uygulamalarda Raspberry Pi ve USB kamera aracılığı ile elde edilen video görüntüye temel görüntü filtreleme uygulamalarına sonuçları gösterilmektedir. (A)

Gauss, (B) Sobel kenar algılama, (C) gri tonlamalı görüntünün tersini alma, (D) gri tonlamalı görüntünün siyah-beyaz görüntüye dönüştürülmesi, (E) Morfolojik açma ve (F) ‘de Morfolojik kapama işlemlerine ait sonuçlar gösterilmektedir. **Şekil 4.1**’de gösterilen filtreleme uygulamalarının gerçekleştirilmesi için kullanılan evrişim işlemi fonksiyon yapısı gösterilmektedir.

```
Fonksiyon Evrisim_islemi(Görüntü matrisi, Filtre matrisi)
        G_satır, G_sütun = boyut(Görüntü matrisi)
        F_satır, F_sütun = boyut(Filtre matrisi)
        H=(F_satır-1)/2
        W=(F_sütun-1)/2
        döngü i H'tan  G_satır-H'a kadar
                döngü j W'dan  G_sütun-W'ya kadar
                        Toplam=0
                        döngü k -H'tan  H+1'e kadar
                                döngü l -W'dan  W+1'e kadar
                                        a=Görüntü matrisi[i+k,j+l]
                                        b=Filtre matrisi[H+k,W+l]
                                        Toplam=Toplam+(a*b)
                        Çıkış matrisi[i,j]=Toplam
```

Şekil 4.2. Evrişim işlemi temsili kodları

Şekil 4.2’de Raspberry Pi platformunda kullanılan evrişim işlemine ait temsili kodlar gösterilmektedir. Genel olarak görüntü filtrelemede evrişim işlemi kullanıldığından, uygulanacak görüntü ve filtre matrisinin fonksiyona giriş olarak verilmesi ile çıkışta kullanıcının tanımladığı filtreleme işlemi gerçekleştirilmiş olacaktır.

Raspberry Pi platformunda açık kaynak kütüphanelerin kullanımı işlem süresini oldukça kısaltmaktadır. **Şekil 4.1**’de gerçekleştirilen filtreleme işlemlerinde açık kaynak kütüphaneler kullanılmadan gerçekleştirilmiştir. Ancak, işlem süresi görüntü boyutuna bağlı olarak çok uzun zamanda gerçekleşmektedir. Bu platformda, ısınma probleminden dolayı ileri seviye uygulamalar gerçekleştirildiğinde, platform hata vermekte ve kendini kapatarak işlemlerin gerçekleşmemesi meydana gelmektedir. İleri seviye uygulamalar için Jetson platformlarının veya Python ile programlanabilen FPGA (PYNQ) kullanımı önerilmektedir.

4.1.2. Gerçek Zamanlı Yüz Tanıma Uygulaması

OpenCV kütüphaneleri kullanarak Raspberry Pi gömülü sistem platformunda gerçekleştirilen, görüntü filtreleme, morfolojik işlemler, kenar bulma algoritmaları, insan yüzü ve araç tanıma gibi uygulamalar gerçekleştirilebilmektedir. Yüz tanıma işlemi genel olarak, yüz bölgesinde bulunan bazı kısımların (göz, kaş, burun vb.) etiketlenmesi ile elde edilen harita aracılığı ile yüzün belirli noktalarının tespit edilmesi ile gerçekleştirilmektedir. Yüz tanıma için oluşturulan xml uzantılı dosya, kameradan elde edilen resimde yüz bölgesi olabilecek bölgeleri tespit etmek

için kullanılmaktadır. Bu algoritmaya göre tanımlanan yüzler bir dikdörtgen çerçeve içerisine alınarak gösterilmektedir.

Şekil 4.3. Raspberry Pi ile yüz tanıma uygulaması

Şekil 4.3'te USB kamera ve 1.4 GHz dört çekirdek 64 bit Raspberry Pi 3B+ modeli kullanılarak gerçek zamanlı video görüntüde bulunan yüz bölgesinin tespiti, haar-cascade özellikler ve OpenCV kütüphaneleri ile gerçekleştirilmiştir.

Yüz veya nesne tanıma gibi uygulamalarda, örneğin yüz tanıma ile kişilerin birbirinden ayrılması vb. daha ileri seviyelerde uygulamaların gerçekleştirilmesi için derin öğrenme modelleri kullanımı ile Raspberry Pi veya özellikle Jetson platformlarında uygulamaların geliştirilmesi önerilmektedir.

4.1.3. Tensorflow ve Raspberry Pi Platformunda Yaya ve Araç Plaka Tespiti

Tensorflow, makine öğrenmesi veya derin öğrenme modellerinin kendi verilerinizle ya da açık kaynak kullanılabilen veri setleri ile model geliştirilmesine olanak sağlayan açık kaynak kütüphanedir. Bu kütüphaneyi kullanarak uygulama geliştirmek için kendi modelinizi oluşturabilir, android, IOS ve Raspberry Pi gibi tanışabilir cihazlarda veya bilgisayarlarda eğitim ve sınıflandırma işlemleri gerçekleştirilebilmektedir. Tensorflow kütüphanesi ile daha önceden geliştirilmiş modeller kullanılabilir veya yeni bir model oluşturarak yeniden eğitilebilir, eğitilmiş modellerin mobil gömülü sistem platformlarında çalıştırılabilmesi için genellikle Tensorflow-lite kullanılmaktadır.

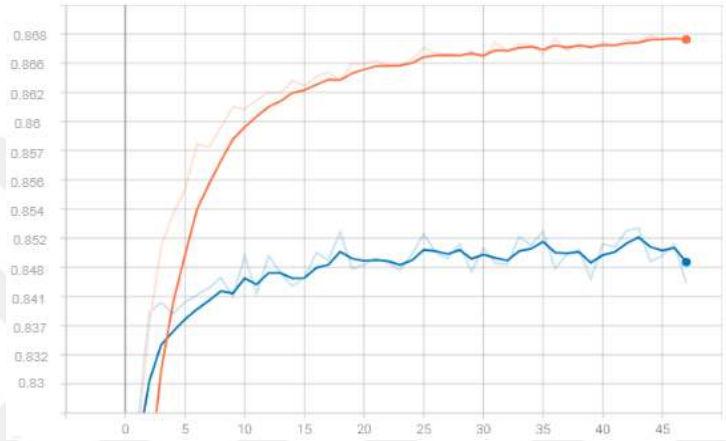
Tensorflow-lite kullanılarak gerçekleştirilen eğitim çalışmalarının aşamaları;

1. Sınıflandırma yapılacak nesnelere ait görüntüler elde ediliri: Elde edilen görüntü sayısı ile modelin eğitim sonunda modelin değerlendirilmesi ile doğru orantılıdır. Görüntü sayısının çok olması model doğruluk oranını arttıracaktır.

2. Etiketleme işlemi: Her bir görüntüde bulunan sınıflandırılacak nesnelere LabelImg, pyqt ve Anaconda ara yüzü ile sınıf bilgisi etiketleme işlemi gerçekleştirilir. Bu işlemin sonunda her bir görüntü için .xml uzantılı dosyalar elde edilmektedir.

3. Eğitim ve test için ayrılan görüntü .xml dosyaları anaconda ve Python dili (isteğe bağlı olarak farklı dillerde de gerçekleştirilebilir) ile model eğitimi gerçekleştirilebilir duruma getirilir ve bu aşamadan sonra Tensorflow için. record uzantılı dosta elde edilmektedir.

4. Modelin doğruluk oranının yüksek olması için daha önce eğitimi tamamlanmış modellerin parametreleri kullanılarak (Transfer öğrenme, Fine_Tuning) model eğitimi gerçekleştirilecektir.



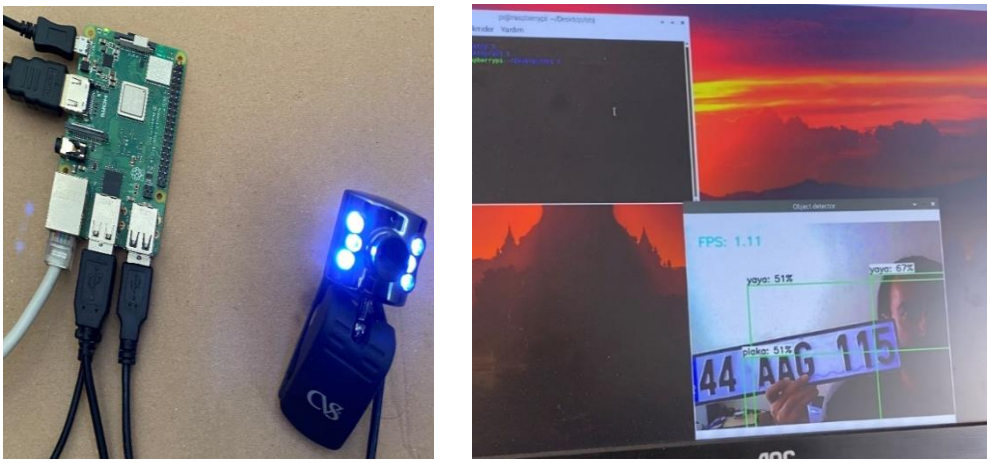
Şekil 4.4. Tensorboard model eğitim sonucu

Şekil 4.4'te Anaconda Navigator (veya Google colab) ve Tensorboard kullanılarak eğitilen modelin sonucu gösterilmektedir. Turuncu ile eğitim ve mavi ile test sonuçları temsil edilmektedir.

Tensorflow-lite eğitim aşamalarından sonra RasperryPi ile çalıştırılması için gerekli işlem adımları;

1. Gerekli kütüphanelerin (OpenCV, Tensorflow vb.) kurulumu gerçekleştirilir,

2. Tensorflow derleyicisinin kurulumu gerçekleştirildikten sonra modelin çalıştırılması gerçekleştirilebilir.



Şekil 4.5. Raspberry Pi gerçek zamanlı plaka ve yaya tespiti

Şekil 4.5'te Raspberry Pi 3B+ modeli ve Tensorflow 2.5.0 kütüphanesi kullanılarak gerçek zamanlı yaya ve plaka tespiti uygulaması MobileNet modeli kullanılarak gerçekleştirilmiştir. Gerçek zamanlı nesne tespiti için kullanılan Raspberry Pi uygulama düzeneğinde bir adet 12 mega piksel CVS DN-981USB kamera, VGA (HDMI bağlantılı) monitör, Ethernet, klavye ve fare bulunmaktadır. MobileNet parametreleri kullanılarak kendi görüntü veri setimiz ile Fine_Tuning öğrenme gerçekleştirilmiştir.

Görsel veri setleri ile çalışıyorsak küçük veri setleri ile gerçekleştirilen modellerin doğruluk oranları düşük ve tanıma performansları için veri miktarı yeterli değildir. Bu yüzden, model eğitim sürecinde her bir sınıfa ait binlerce verinin kullanılması gerekmektedir. Küçük veri seti ile gerçekleştirilecek derin öğrenme modelleri yapay veriler (agumentasyon) ile veri setinde bulunan örneklemeler artırılmalıdır veya transfer öğrenme, fine tuning yöntemleri ile önceden eğitilmiş modeller parametreleri kullanılabilir.

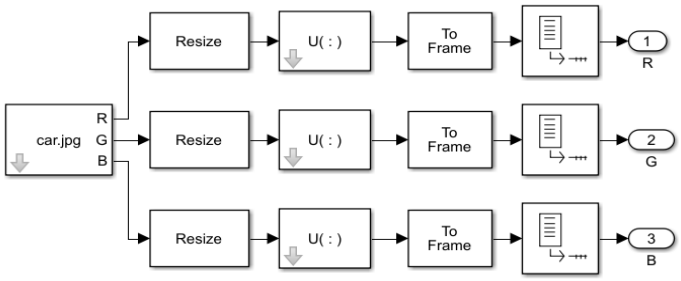
4.2. FPGA Platformunda HDL Tabanlı Tasarımlar

XSG, model tabanlı MATLAB/Benzetim tasarımının FPGA'ya uygulanmasını sağlayan Xilinx firmasının üretmiş olduğu bir Sayısal işaret işleme (DSP: Digital signal processing) tasarım aracıdır. Benzetim uygulamalarının kolaylıkla gerçekleştirilebilmesi için XSG'de, toplayıcılar, çarpanlar, kaydediciler (yazmaçlar), hata düzeltme blokları, FFT'ler ve farklı filtrelerin bulunduğu birçok DSP bloğu bulunmaktadır.

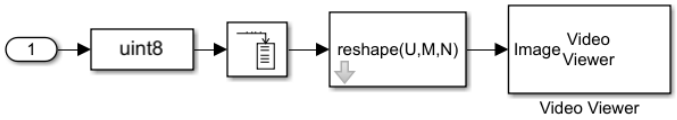
4.2.1. Xilinx Sistem Generatörü ile Görüntü İşleme Tekniklerinin Uygulanması

Bu bölümde, MATLAB ve Xilinx Sistem Generatör (XSG) kullanılarak gerçekleştirilen bazı görüntü ve video işleme uygulamaları, renkli görüntünün gri tonlu görüntüye dönüştürülmesi,

görüntünün tersini alma işlemi, görüntü kalitesinin güçlendirilmesi, morfolojik işlemler, eşik değer atama ve kenar bulma algoritmaları gösterilmektedir. XSG de bu algoritmaları kullanabilmek için başlangıç ve son işlem (pre and post processing) basamaklarının arasında gerçekleştirilmesi gerekmektedir. Bu işlemler **Şekil 4.6** ve **Şekil 4.7** de gösterilmektedir.



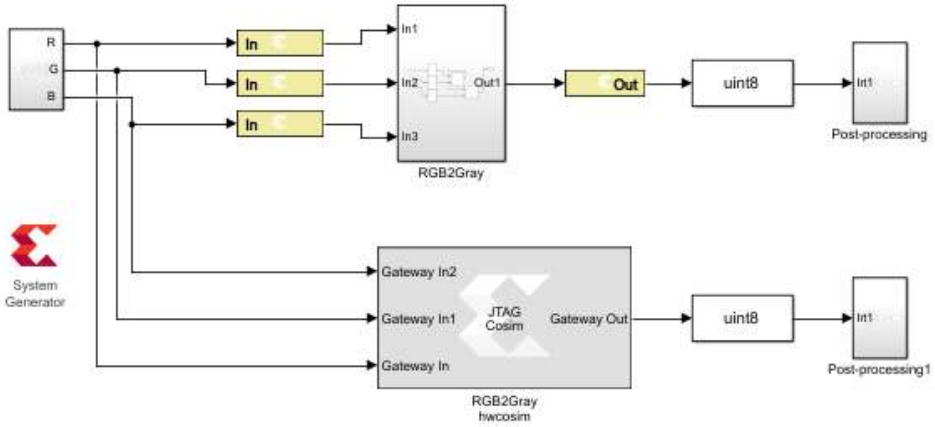
Şekil 4.6. Renkli görüntünün çerçevelerine ayrılması



Şekil 4.7. Görüntünün yeniden boyutlandırılıp gösterilmesi

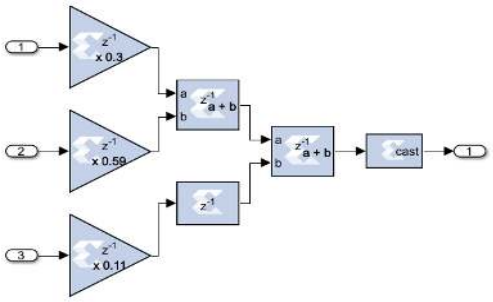
RGB görüntünü Gri tonlamalı görüntüye dönüştürülmesi:

XSG kullanılarak gerçekleştirilen uygulamalardan biri olan üç boyutlu renkli görüntünün bir boyutlu gri tonlamalı görüntüye (RGBtogray) dönüştürülmesi **Şekil 4.8**'de gösterilmektedir. Uygulama Zedboard Zynq (xc7z020-1c1g484) geliştirme kartı ile System Generator 2018.1 ve MATLAB R2018a platformları kullanılarak gerçekleştirilmiştir. JTAG Cosim bloğu XSG tarafından kullanıcının tanımladığı benzetim programını kullanarak üretilmektedir. Elde edilen JTAG Cosim bloğu ile üç boyutlu renkli görüntünün gri tonlamalı görüntüye dönüştürülme işlemleri gerçekleştirilebilmektedir. **Şekil 4.8**'de gösterilen her iki görüntü son işlemler bloğunun çıkış görüntüleri aynı sonucu vermektedir.



Şekil 4.8. Renkli görüntüden gri tonlu görüntünün elde edilmesi

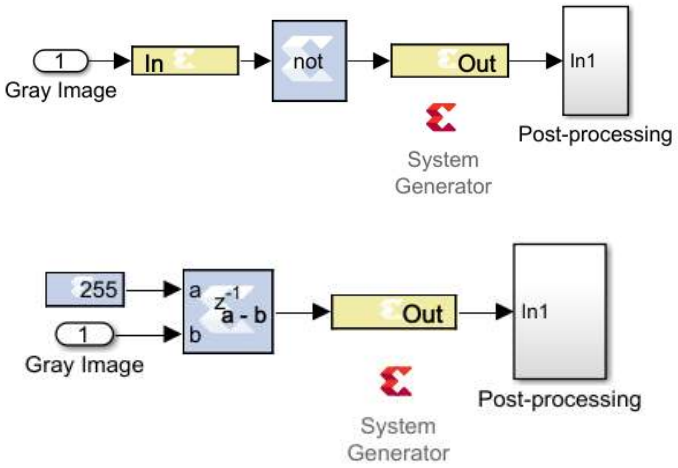
XSG benzetim ön işlemlerinde üç boyutlu renkli görüntü, kırmızı, yeşil ve mavi bileşenlerine ayrılarak denklem 2.2'deki matematiksel işlemler kullanılarak gri tonlamalı görüntü elde edilmiştir ve bu benzetim blokları detayları ile birlikte **Şekil 4.9**'da gösterilmektedir. XSG kullanılarak renkli görüntünün gri tonlamalı görüntüye dönüştürülmesi ile elde edilen görüntü **Şekil 4.10**'da gösterilmektedir.



Şekil 4.9. RGB to gray dönüşümü XSG alt sisteminin iç yapısı

Şekil 4.10. RGB to gray dönüşümü

Görüntü tersleme:



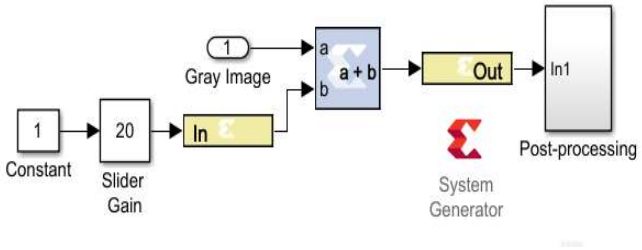
Şekil 4.11. XSG Görüntü tersleme uygulaması

Şekil 4.11'de XSG kullanılarak gri tonlamalı görüntünün tersleme algoritmasının uygulanması iki farklı şekilde gösterilmiştir.

Şekil 4.12. Gri tonlamalı görüntüye eşik değer uygulanması

Bu uygulamaların ilkinde, giriş görüntüsünün mantıksal değil (NOT) kapısı ile gerçekleştirilmiştir. İkinci uygulamada, giriş görüntüsü sekiz bitlik (0-255 piksel değer aralığı) gri tonlamalı görüntü olduğundan, her bir piksel değeri 255'ten çıkartılarak görüntü terleme işlemleri gerçekleştirilmiştir. **Şekil 4.12**'de gri tonlamalı görüntüye eşik değer uygulaması gösterilmektedir.

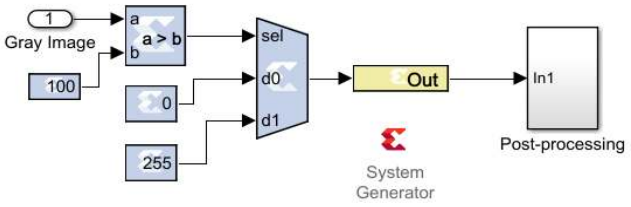
Görüntünün kuvvetlendirilmesi:



Şekil 4.13. XSG ile görüntünün kuvvetlendirilmesi

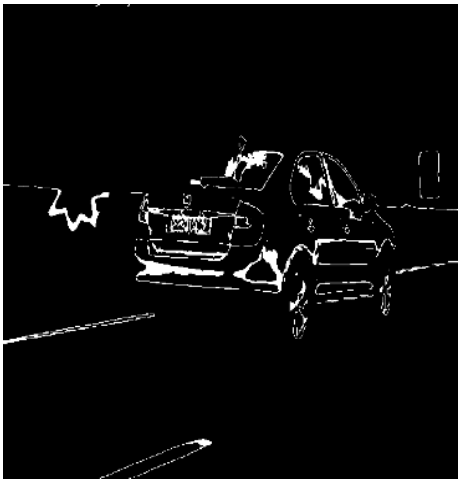
Şekil 4.13'te XSG kullanılarak gri tonlamalı görüntünün kuvvetlendirilmesi işlemi gerçekleştirilmiştir. Başka bir ifade ile görüntü piksellerinin niteliklerinin arttırılmasıdır. Bu uygulamada, gri tonlamalı giriş görüntüsünün her bir piksel değerine belirli bir katsayı eklenmesi veya çarpılması ile gerçekleştirilmektedir.

Görüntüye eşik değer uygulanması:



Şekil 4.14. XSG ile eşik değer uygulaması

Şekil 4.14'te XSG ve MATLAB benzetim blokları kullanılarak gri tonlamalı görüntüye eşik değer ataması uygulaması gösterilmektedir. Gri tonlamalı giriş görüntüsüne ait pikseller belirlenen eğiş değeri (100 alınmıştır) ile karşılaştırma işlemi gerçekleştirilmektedir. Piksel değeri belirlenen eşik değerinden büyük olduğu durumlarda 255 ve küçük olduğu durumlar için piksel değeri ne sıfır değeri atanmaktadır. Böylelikle sadece 0 ve 255 piksel değerlerine ait görüntü elde edilmiş olacaktır.

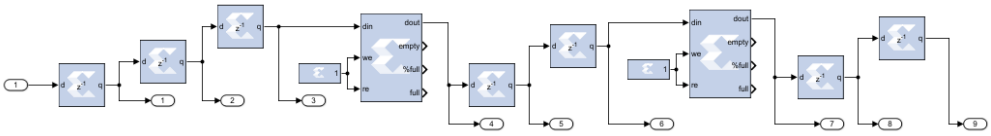


Şekil 4.15. Gri tonlamalı görüntüye eşik değer ataması uygulaması

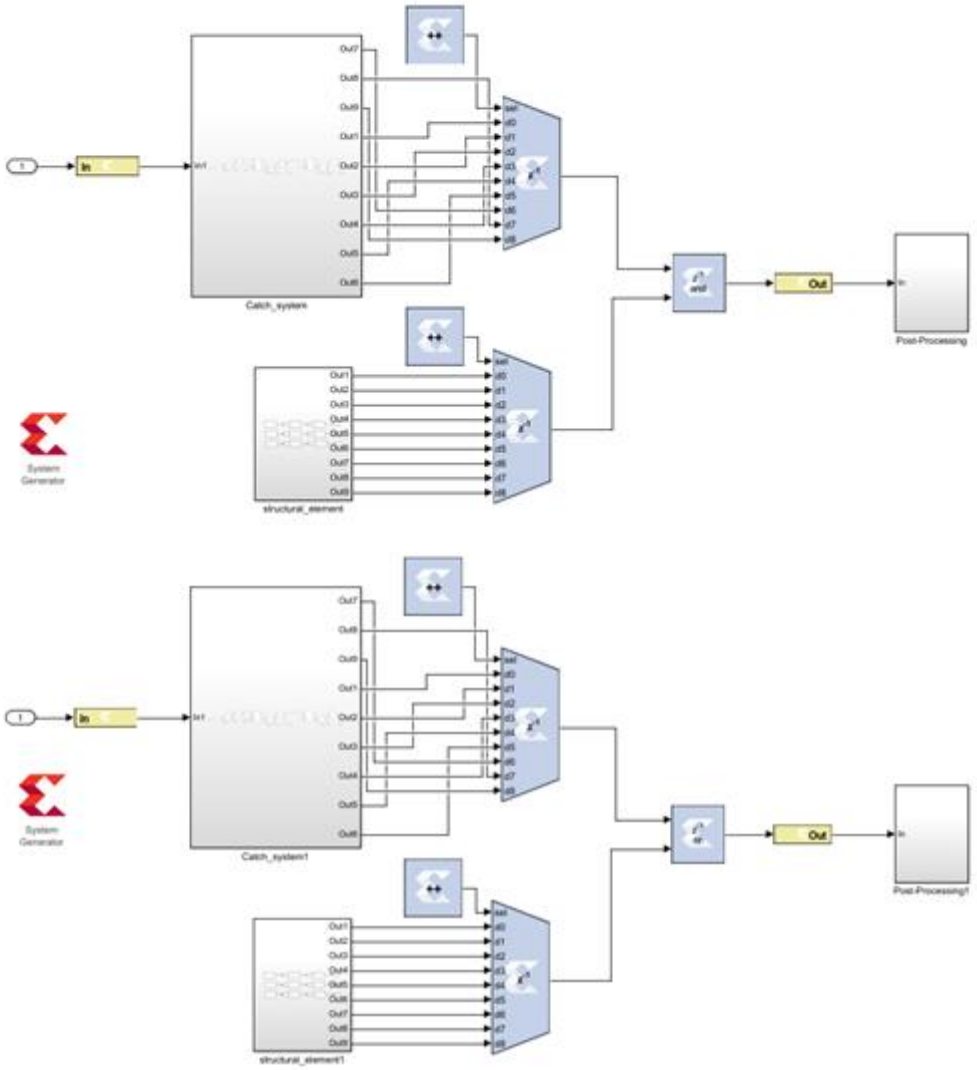
Şekil 4.15'te XSG kullanılarak gri tonlamalı görüntüye eşik değer ataması uygulamasına ait çıkış görüntüsü gösterilmektedir. Bu uygulama sayesinde görüntü üzerinde bulunan kullanılmayacak kısımlar filtrelenmiş olacaktır. Gri tonlamalı görüntüye eşik değer uygulama blok diyagramında, görüntü üzerinden hesaplanabilen (ortalama değer, ortanca değer vb.) değer ile karşılaştırılarak yeni piksel değerlerinin elde edilmesi ile adaptif eşik değer oluşturulması görüntüde nesnelere ait bazı özelliklerin bu uygulama sonucunda ortadan kaldırılmasını engelleyecektir. Örnek uygulamada aracın bir tekerleğinin ve yol işaret levhasının ortadan kaldırılması engellenmiş olacaktır.

4.2.2. Xilinx Sistem Generatörü ile Görüntüye Morfolojik İşlemlerin Uygulanması

XSG/MATLAB-Benzetim programını kullanılarak görüntüye morfolojik işlemlerin uygulanabilmesi için, görüntü üzerinde sıralı üç adet satır bilgisinin (piksellerin) elde edilmesi gerekmektedir. Bunun için **Şekil 4.16**'da gösterilen yapı ile piksellerin kontrolü, zamansal gecikmeleri ve kayıt edilip yeniden kullanılmaları sağlanmıştır. Bu blok sayesinde XSG-MATLAB benzetim programı ile genel olarak kullanılan filtreleme işlemleri gerçekleştirilebilmektedir.



Şekil 4.16. Morfoloik işlemler için piksellerin kontrolü



Şekil 4.17. XSG ile morfolojik aşındırma ve yayma uygulamarı

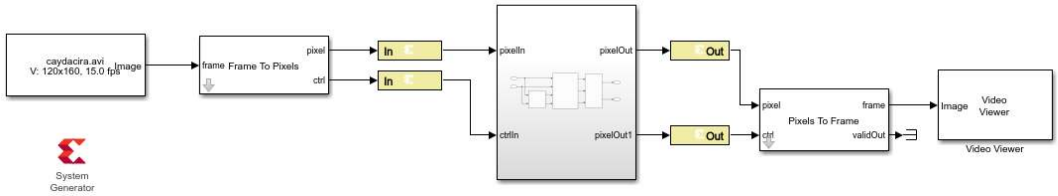
Şekil 4.17'de XSG kullanılarak gri tonlamalı görüntüye morfolojik açma ve kapama uygulamaları blok diyagramı gösterilmektedir. "*Catch_system*" bloğu ile girişe uygulanan görüntüye ait pikseller hafızada tutularak, yapısal elamana ait filtre bloğu kullanılarak morfolojik işlemler gerçekleştirilmektedir. Mux yapısı ile görüntü piksel yakala bloğunda bulunan pikseller ve yapısal elemanın filtre katsayıları çıkışa aktarılarak morfolojik işlemlerin gerçekleşmesi sağlanacaktır. Gri tonlamalı giriş görüntüsü üzerinde yapısal elemana ait filtre matrisinin, evirişim işlemi gibi gezdirilerek görüntü üzerinde bulunan tüm piksel değerleri ile yapısal elemana ait piksel değerlerinin tam eşleşmesi ile aşındırma (erosion) işlemi, mantıksal OR işleminin uygulanması ile yayma (dilation) işlemi gerçekleştirilmiş olacaktır. Gri tonlamalı görüntüye önce, yayma ve ardından yayma işlemi uygulanmış görüntüye aşındırma işlemi uygulanması ile morfolojik kapama ve bu işlemin tam tersi uygulama ile morfolojik açma işlemleri uygulanmış olacaktır.

Şekil 4.18. Morfolojik açma ve kapama uygulaması sonuçları

Şekil 4.18'de XSG/MATLAB-Benzetim blok yapıları ile oluşturulmuş morfolojik uygulamalara ait sonuçlar gösterilmektedir. Sol tarafta morfolojik açma işlemi ile görüntüde arka planda bulunan bazı kısımlar filtrelenmiş ve sağ tarafta morfolojik kapama işlemi ile görüntüde bulunan nesnelerin üzerinde bulunan istenilmeyen gürültüler filtrelenmiştir.

4.2.3. Xilinx Sistem Generatörü İle Video Görüntüye Sobel Filtre Uygulanması

Görüntüde bulunan kenarların tespit edilmesinde en yaygın olarak kullanılan Sobel filtresi ile XSG/MATLAB-Benzetim programı kullanılarak aşağıda gösterilen uygulama gerçekleştirilmiştir.



Şekil 4.19. XSG ile kenar belirleme uygulaması

Şekil 4.19'da XSG/MATLAB-Benzetim blok diyagramları kullanılarak video görüntü üzerinde bulunan kenarların tespit edilmesi işlemi gerçekleştirilmiştir. Frame To Pixel kısmında giriş video formatına göre satır ve sütunda bulunan piksellerin konumları ve çıkışta elde edilecek piksellerin kontrolü sağlanmaktadır. XSG giriş-çıkış blokları arasında kalan bölgede sobel kenar algılama sistemi bulunmaktadır. Kenar bulma filtreleme işlemleri için ise giriş görüntüsüne ait pikseller FIFO yardımı ile ilk iki satır piksel değeri hafızada saklanmakta ve üçüncü satırın son piksel değeri ile işlemler gerçekleştirildikten sonra dördüncü ve beşinci satırdaki piksel değerleri

hafızada saklanmaktadır ve bu işlemler tüm görüntü piksellerine uygulanmaktadır. PixelOut kısmında kenarları tespit edilmiş piksellere ait çıkışlar Pixels To Frame ile çıkışta gözlemlenebilecek video formatında sonuçların elde edilmesini sağlamaktadır.

Şekil 4.20. Kenar bulma uygulaması

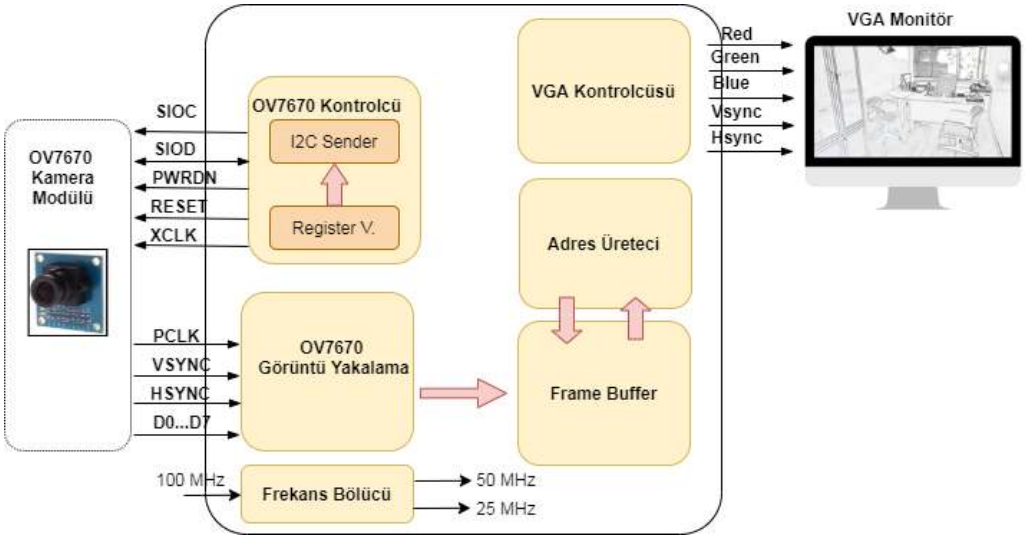
Gerçek zamanlı akan trafikten alınan videoya, XSG/MATLAB-benzetim blok diyagramları aracılığı ile görüntüde bulunan kenarların tespit edilmesi sonuçları **Şekil 4.20**'de gösterilmektedir. Bu uygulama için Sobel kenar bulma algoritması uygulanmıştır.

4.2.4. Renkli Video Görüntüsünün VHDL Kod İle Elde Edilmesi:

VHDL programlama dili ile renkli görüntünün kameradan elde edilmesi aşamaları Vivado 2018.1 sürümü ve Basys3 ve XC7Z020-CLG484 Zedboard FPGA gömülü sistem platformları ile gerçekleştirilmiştir. Bu Tasarımda, OV7670 kameradan, VGA monitöre RGB renk formatında video görüntülemek için video yakalama, kamera kontrol ve denetleyici, çerçeve arabelleği (frame buffer) vb. alt modüller bulunmaktadır. Kamerayı çalıştırmak için, saat darbesi (SIOC), piksellerin (SIOD) kontrolü ve piksel saat darbesi (XCLK) OV7670 kontrolcü alt bileşenleri ile gönderilerek OV7670 görüntü yakalama algoritmaları kullanılarak kameradan görüntüler (D0…D7) elde edilebilmektedir.

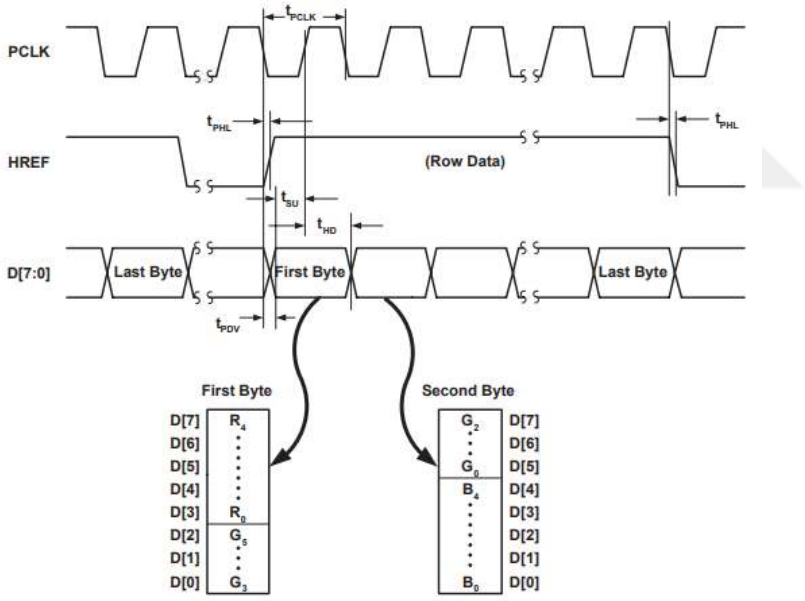
OV7670 Kamera:

OV7670 kamera modülü, RBG formatında 8 bit veri çıkışına sahiptir. Bu veriler piksel saat darbeleri ile yakalanmaktadır. Ardışık iki saat darbesinde piksel verileri birleştirilerek 16 bit RGB565 formatına dönüştürülmektedir. Böylelikle bir piksel için kırmızı, yeşil ve mavi renk tonlama değerleri elde edilmiş olur.



Şekil 4.21. OV7670 Kamera modülü blok diyagramı

Şekil 4.21'de OV7670 kamera modülünün kontrolü ve görüntünün elde edilme aşamaları ve bu aşamalara ait blok diyagram gösterilmektedir.

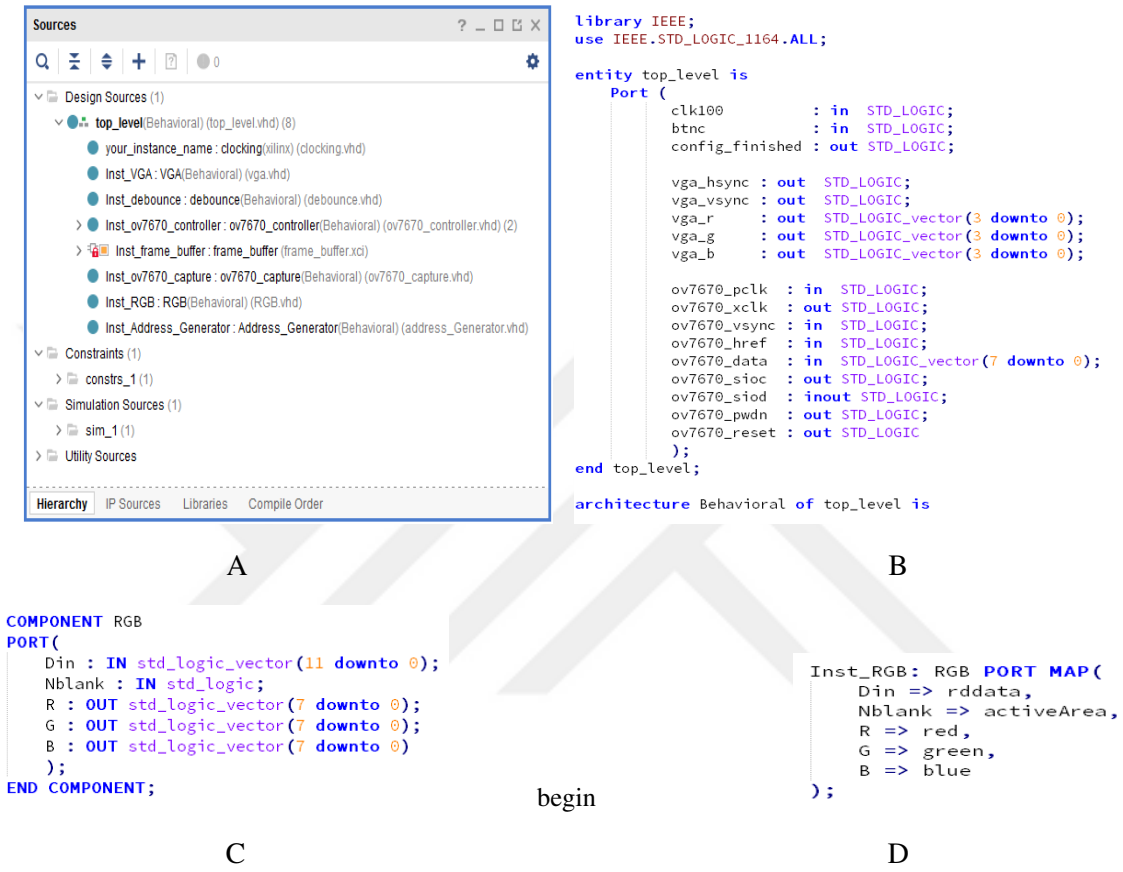


Şekil 4.22. OV7670 RGB565 çıkış diyagramı [125]

Şekil 4.22'de OV7670 kamerası RGB565 formatında veri çıkışına ait diyagramı gösterilmektedir. Altı adet yeşi, beşer adet kırmızı ve mavi piksel değerlerinin oluşması için iki adet saat zamanının (PCLK) geçmesi ile 16 bitlik renkli görüntü kameradan elde edilebilmektedir.

Piksel değerlerinin eşzamanlı bir şekilde VGA monitöründe gösterilebilmesi için okuma, yazma çerçeve arabelleği ve adres üreteci kullanılmaktadır. Böylelikle monitörde titreşimler ve gürültüler azaltılmış olacaktır. Veri okuma ve yazma kısmı FIFO (First In First Out) olarak

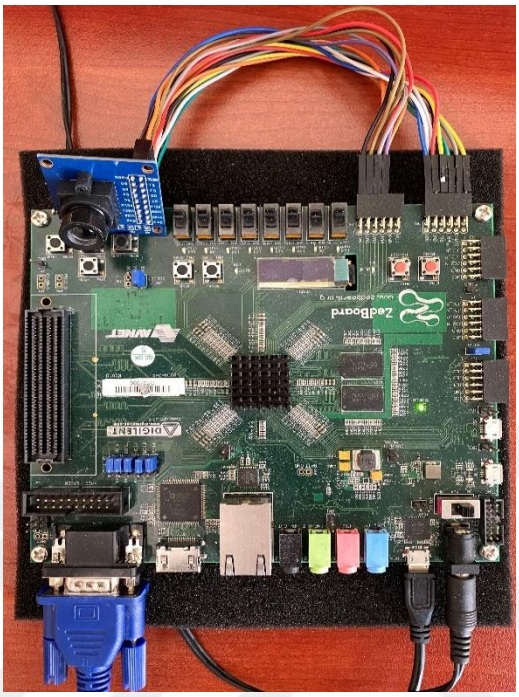
çalışmaktadır. Buraya gelen pikseller FIFO boyutuna göre belli sayılarda ve gelen sıra ile hafızada tutulmaktadır. FIFO hafızasına alınan pikseller, saat darbesi ile eş zamanlı olarak VGA kontrolcüsüne aktarılmaktadır. VGA kontrolcüsü ekranda (yatay ve dikeyde) hangi pikselin nerede bulunması gerektiğini organize etmektedir. Böylelikle kameranın gördüğü kısım monitörde gösterilebilmektedir.



Şekil 4.23. VHDL dili kullanılarak renkli görüntünün Vivado 2018.1 ara yüzü gösterimi

Şekil 4.23'te VHDL dili kullanılarak renkli görüntünün Vivado 2018.1 ara yüzü gösterilmektedir. Bu kısımda vivado 2018.1 ve OV7670 kamera kullanılarak renkli görüntünün VGA ekranda gösterilmesine kadar olan süreçler gösterilmektedir. Source kısmında **Şekil 4.23** (A)'da rekli görüntünün elde edilmesi için gerekli tüm kodlama dosyaları bulunmaktadır. Kod parçaları ile gösterilen kısımda **Şekil 4.23** (B, C ve D)'de kodlamaya ait üst ana modül (top level) ve alt modüllere ait örnek kısımları göstermektedir. Yapı (Architecture) kısmından sonra her bir kod dosyasını top modül ile çalıştırabilmek için bileşen (component) olarak tanımlamamız gerekmektedir ve yapıya ait başla (begin) komutundan sonra bileşen olarak atanan kodlama dosyalarına ait giriş çıkış pinleri, FPGA giriş-çıkış pinleri ile ilişkilendirilerek tek bir parça halinde çalıştırılmasını sağlamaktadır. Buradaki önemli noktalardan biri, top modül ile alt modüllerde verilen kod yazım dilleri farklı olabilmektedir. Yani; biri Verilog iken diğeri VHDL ile tasarlanmış

olabilir. Böylelikle daha önceden gerçekleştirmiş olduğunuz projelerden kesitler alıp farklı uygulamalarda da kullanılabilmektedir. Bu aşamaya kadar olan tüm süreçler görüntünün elde edilmesi ayarlamaları olarak adlandırılabilir ve daha sonraki görüntü işleme uygulamaları için ortak olarak kullanılacaktır.



Şekil 4.24. Zedboard FPGA uygulama düzeneği

Şekil 4.24'te Zedboard geliştirme kartı kullanılarak gerçekleştirilen uygulama düzeneği gösterilmektedir. OV7670 kamera kullanılarak gerçekleştirilen uygulamaların sonuçları, RS232 VGA girişi ile monitörde anlık olarak gösterilmektedir. Kamera bağlantıları XC7Z020-CLG484 geliştirme kartının PMOD dijital girişleri kullanılarak gerçekleştirilmiştir.

Tablo 4.1. FPGA kaynak kullanımı

Kaynaklar	VHDL kod ile RBG Video	Mevcut
LUTs	350	53200
Slice Register	230	106400
BRAM	44	140
IO	35	200

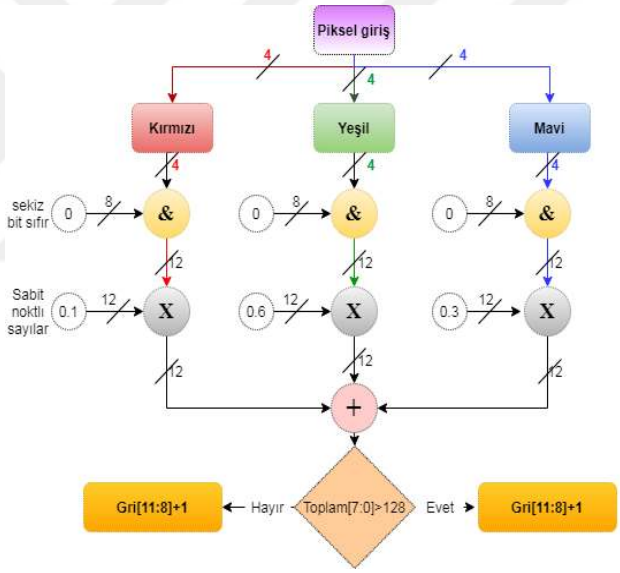
Tablo 4.1'de Zedboard geliştirme kartı kullanılarak renkli video görüntüsü uygulamasının kaynak kullanımı miktarları verilmektedir.

4.2.5. Gri Tonlamalı Video Görüntüsünün VHDL Kod İle Elde Edilmesi:

Kameradan RGB565 formatında elde edilen 6640x480 renkli görüntünün gri tonlamalı görüntüye dönüştürülmesinde, renkli görüntünün elde edilmesi için gerçekleştirilen süreçlerden sonra gri tonlamalı dönüşümün gerçekleştirilmesi için denklem 4.1'de bulunan matematiksel işlemler gerçekleştirilmiştir.

$$Gri(i,j)=R(i,j)*0.114+G(i,j)*0.587+B(i,j)*0.299 \quad (4.1)$$

RGB-Gri tonlamalı görüntü dönüşümü işleminin aşamaları **Şekil 4.25**'te gösterilmektedir. OV7670 kamerasından elde edilen her bir piksel FPGA hafıza biriminden okunarak, Kırmızı, Yeşil ve Mavi ve üç boyutlu renkli görüntüye ait pikseller sabit noktalı ikili sayı sistemine dönüştürülmesi için bu sayıların sağ kısmına sekiz bit sıfır değeri eklenerek gerçekleştirilmiştir. Çerçevelerine ayrılmış pikseller, denklem 4.1'de verilen ilgili katsayılar ile çarpıldıktan sonra toplama işleminin gerçekleştirilmesi ile gri tonlamalı görüntü elde edilebilmektedir.



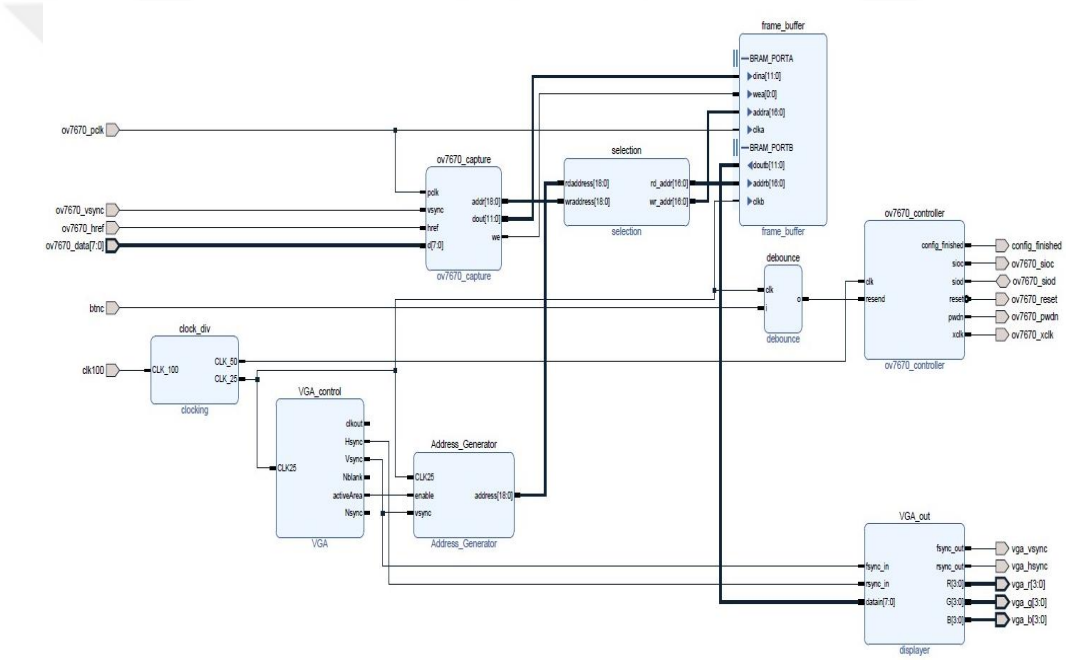
Şekil 4.25. Renkli görüntünün sabit noktalı sayi kullanılarak gri tonlu görüntüye dönüştürülmesi

4.3. FPGA Platformunda Gerçek Zamanlı IP Çekirdek Tabanlı Tasarımlar

FPGA görüntü işleme uygulamaları görüntünün bir kamera aracılığıyla elde edilmesi veya görüntünün hafızadan çekilmesi ile başlamaktadır. Kameradan görüntü yakalama ve işleme mimarisi, kamera görüntü yakalama ve kontrolcüsü, frekans bölücü, çerçeve arabelleği (Frame Buffer), adres üreteci ve VGA kontrol kısımlarından oluşmaktadır. FPGA platformunda görüntü işleme için, kamera aracılığı ile elde edilen pikseller uygulamaya göre gerçekleştirilecek aritmetik işlemler ve piksellerin temsil edilmesi işlemleri sabit noktalı sayı formatı ile gerçekleştirilmiştir.

4.3.1. Renkli Video Görüntüsünün Elde Edilmesi

Kamera aracılığı ile renkli görüntünün elde edilmesine ait VHDL programlama dili ile gerçekleştirilen uygulamaya ait kodlar kullanılarak IP bloklar oluşturulmuştur. Bu uygulama kodlama ve IP çekirdekleri ile elde edilen görüntü için kullanılan kaynak miktarını karşılaştırmak amaçlanmıştır. IP bloklar ile renkli görüntünün oluşturulması **Şekil 4.26**'da gösterilmektedir. OV7670 kamera kullanılarak gerçek zamanlı renkli görüntüler; FPGA platformunda üretilen saat darbesi ile OV7670 kamera kontrolcüsü piksellerin elde edilmesi için SIOC ve SIOD bilgisini göndermektedir. Piksel senkronizasyonu sağlamak için, adresleme IP bloğu ve RGB565 formatında piksellerin oluşumunu gerçekleştirecek yakalama IP bloğu ile kameradan elde edilen piksel değerleri FPGA'nın hafızasına saat darbesi ile işlenmektedir. Okuma saat darbesi ile pikseller hafızadan çekilerek VGA ekrana aktarılmaktadır.



Şekil 4.26. IP Çekirdekleri ile görüntünün elde edilmesi

Şekil 4.27. FPGA, Renkli video görüntüsünün elde edilmesi

Şekil 4.26'da bulunan IP blokları ile oluşturulmuş uygulamanın, **Şekil 4.27**'de video görüntüsü sonucu gösterilmektedir. VGA çıkış IP bloğunda hafızadan okunan pikseller, adres ve ekran pozisyon bilgisine göre oluşturulan yatay ve dikey sinyaller (vga_vsync ve vga_hsync) ile ekrana aktarılmaktadır.

4.3.2. Gri Tonlamalı Video Görüntüsünün elde edilmesi

Renkli görüntüden gri tonlamalı görüntünün elde edilmesi işlemi için kameradan elde edilen her bir piksel değerinin (Kırmızı, Yeşil ve Mavi) sabit noktalı ikili sayı sistemine dönüştürülmesi için bu sayıların sağ kısmına sekiz bit sıfır değeri eklenir. Renkli görüntünün gri tonlamalı görüntüye dönüştürülmesi için kullanılan denklem 4.1'de verilen formül kullanılarak gri tonlamalı görüntü elde edilebilmektedir.

Sabit ve kayan noktalı sayı sistemleri ile gerçekleştirilecek uygulamalarda kaynak kullanımı, güç tüketimi ve hassasiyetlik değerlendirmelerinin gerçekleştirilebilmesi için gerçek zamanlı renkli video görüntüsünün gri tonlamalı görüntüye dönüştürülmesinde iki sayı sistemi ayrı ayrı kullanılarak karşılaştırılmıştır. Denklem 4.1'de verilen matematiksel işlemler ile renkli görüntünün gri tonlamalı görüntüye dönüştürülmesi gerçekleştirilebilmektedir. Bu denklem sisteminin sabit veya kayan noktalı sayı sisteminde kullanılması için;

1. İkilik sayı sistemindeki pikseller, sabit veya kayan noktalı sayı formatına dönüştürülmesi
2. Sabit veya Kayan noktalı sayı sisteminde çarpma işleminin gerçekleştirilmesi
3. Sabit veya Kayan noktalı sayı sisteminde toplama işleminin gerçekleştirilmesi

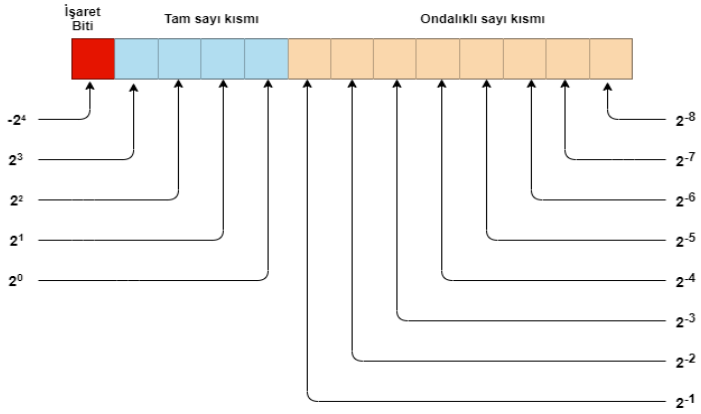
Aşamalarının gerçekleştirilmesi gerekmektedir.

İkili Sayı Sistemleri

Gömülü sistem platformlarında ikili sayıların gösterim türlerinden biri sabit noktalı ikili sayı ve diğeri de kayan noktalı sayı formatıdır.

Sabit noktalı ikili sayı (Binary) sistemi:

Sayı sistemlerinin gösteriminde kullanılan tekniklerden biri de sabit noktalı sayı sistemidir. Sabit noktalı sayı sistemi Qm.n formatında değiştirilen değerler için "m" tamsayı kısmını ve "n" ise virgülden sonraki sayıları temsil etmektedir. Örneğin onluk sayı sisteminde 3.5 sayısı, Q4.8 formatındaki sabit noktalı sayı sisteminde "001110000000" olarak temsil edilecektir. **Şekil 4.28**'de kesirli sayının, işaretli sabit noktalı bir sayıya dönüştürülmesi gösterilmektedir.



Şekil 4.28. Sabit noktalı ikili sayı sistemi gösterimi

Sabit noktalı ikili sayı sisteminde, kesirli kısım için kullanılan bit sayısı bize bu tekniğin hassasiyetini vermektedir. **Şekil 4.28**'de sekiz bit olarak seçilen formatın virgülden sonra 0,996 ile $2^{-8} = 3{,}906.10^{-3}$ sayısına kadar değer alabildiğini göstermektedir. Sabit noktalı sayı sisteminde, noktanın bulunduğu yer değişmediğinden, aritmetik işlemlerin gerçekleştirilmesi kayan noktalı sayı sistemine göre daha erken süreçlerde tamamlanabilmektedir.

Kayan noktalı ikili sayı (Binary) sistemi:

Bilgisayar ortamında ikili sayıların gösterim türlerinden biri de kayan noktalı sayı formatıdır. Aynı bit uzunluğunda sabit noktalı ikili sayı sisteminden daha geniş sayı aralığı gösterimine sahiptirler.

Şekil 4.29. Kayan noktalı ikili sayı sistemi gösterimi

$$bias = 2^{n-1} - 1 \tag{4.2}$$

$$e - bias = yuvarla(log2(mutlak(sayı))) \tag{4.3}$$

$$sayı = (-1)^{s}.(1.f).2^{e-bias} \tag{4.4}$$

Şekil 4.29'da kayan noktalı ikili sayı sistemine ait sayıların genel olarak gösterimi temsil edilmektedir. Sabit noktalı sayı sistemi için, Üs değeri "e", çarpan bilgisi "f" ve işaret biti "s" ile gösterilmektedir.

Aynı bit uzunluğuna sahip ikilik sayılarda, kayan noktalı sayı sisteminde daha geniş sayıların temsil edilebilmektedir. Donanım aritmetiği geleneksel olarak tamsayı veya sabit noktalı aritmetik sayı sistemleri ile temsil edilmektedir. Ancak, Gömülü sistem platformu kaynaklarının önemli

ölçüde artması ile artık kayan nokta veya logaritmik temsiller gibi daha karmaşık aritmetik formatların kullanılması mümkündür.

16 bit uzunluğunda yarı duyarlı kayan noktalı sayı sisteminde en düşük anlamlı bitinin değişimi ile onluk tabanda bu sayı $9{,}765x10^{-19}$ hassasiyette değişmektedir. Aynı bit uzunluğuna sahip sabit noktalı sayı sisteminde ise $1x10^{-10}$ hassasiyette değişebilmektedir. Onluk tabanda 263,3 sayısı, sabit noktalı sayı sisteminde: 100000111_0100110011 (9 bit tam 10 bit ondalıklı) 19 bit uzunluğunda, kayan noktalı sayı sisteminde: 1111_00001110100 (4 bit tam 10 bit ondalık) 14 bit uzunluğunda temsil edilebilmektedir.

Sabit noktalı sayı sistemi ile kayan noktalı sayı sistemi karşılaştırıldığında, kayan noktalı sayı gösterimi, gerçek sayılar için açık ara en yaygın kullanılan gösterimdir. Eşit bit sayılarına sahip sabit noktalı sayı temsili ile karşılaştırıldığında, kayan noktalı sayılar, üs biti genişliği ile katlanarak artan daha geniş aralıklarda sayılar temsil edilebilmektedir[123]. Öte yandan, kayan noktalı sayı gösteriminin kullanıldığı tasarımlarda donanımsal olarak kaynak kullanımı ve güç tüketimi artmaktadır. Ses tanıma, görüntü işleme vb. uygulamaları, kayan noktalı sayı veya sabit noktalı sistemler ile daha geniş dinamik aralıklarda uygulamalar gerçekleştirilebilmektedir[124]. Bu özelliklerden yararlanılarak gerçekleştirilecek uygulamanın minimum sayıda kesir ve üs bitleri kısmının kullanımı ile gerçekleştirilecek programın genel doğruluğu korurken güç tüketimini önemli ölçüde azaltılabilmektedir.

Sabit nokta aritmetiğinden kaynaklanan hesaplama hatasına rağmen, gömülü sistemler için maliyet ve güç tüketimi en aza indirilebilmektedir. Bu nedenle, veri işleme süresini en aza indiren ve verilen hesaplama doğruluğu kısıtlamalarını karşılayan optimize edilmiş sabit noktalı sayı sistemleri kullanılabilmektedir[125]. Sabit noktalı sayı sistem aritmetiği ile gerçekleştirilecek işlemlerde enerji ve hafıza alanı maliyetlerinin, kayan noktalı sayı sistemi ile gerçekleştirilecek işlemlerde daha az olduğu gösterilmektedir. FPGA tabanlı gerçekleştirilecek gelişmiş uygulamalarda (CNN vb.) sabit nokta aritmetiği, kayan nokta aritmetiğine kıyasla daha az kaynak ve enerji tüketimi ile gerçekleştirilmiştir[126].

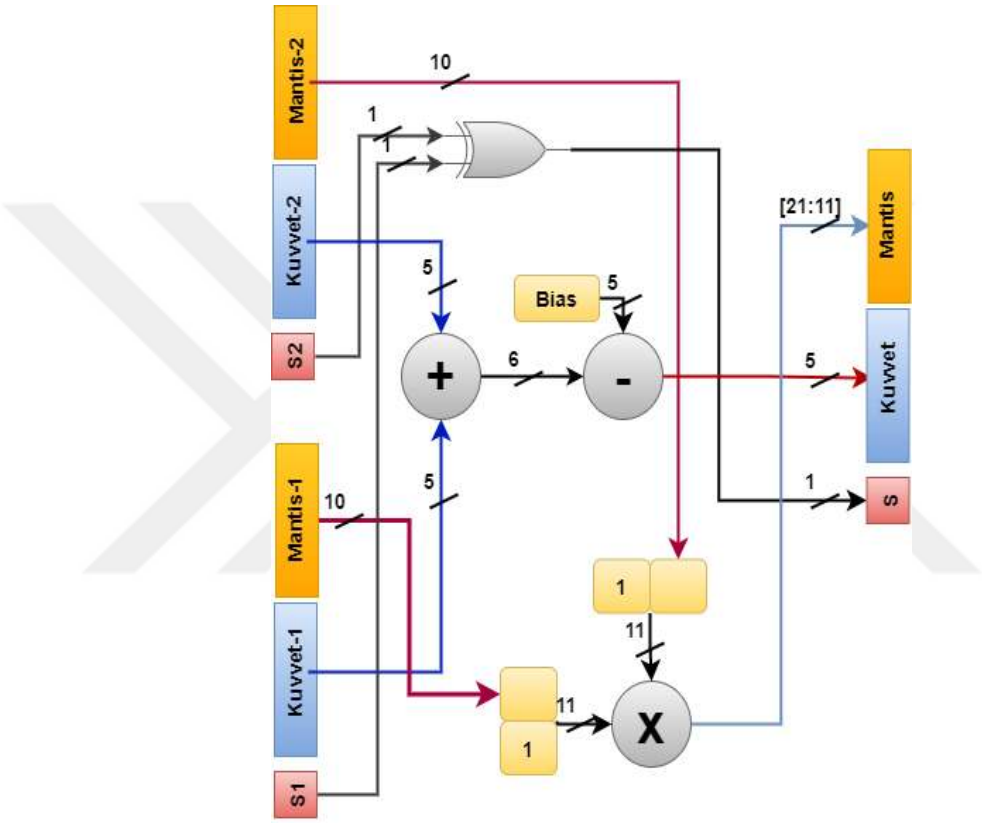
Kayan noktalı sayı sisteminde aritmetik işlemlerin gerçekleştirilmesi

Kayan noktalı sayı sisteminde, iki sayının birbiri ile cebirsel çarpımı işlemi için kayan noktalı sayı formatı, işaret biti, kuvvet biti ve mantis biti olmak üzere üç gruba ayrılır. Eğer kuvvet değeri sıfıra eşit ise mantis değerine bakılmasına gerek duyulmadan sayının sıfıra eşit olduğu söylenebilmektedir.

İki sayının birbiri ile aritmetik çarpıma işleminde iki sayıya ait işaret bitlerinin mantıksal özel veya işlemi ile gerçekleştirilmektedir. İki sayıya ait işaret bitlerinin birbirinden farklı olması durumunda çarpma işleminin sonunda negatif işaret (işaret biti=1) ve işaret bitlerinin aynı olması

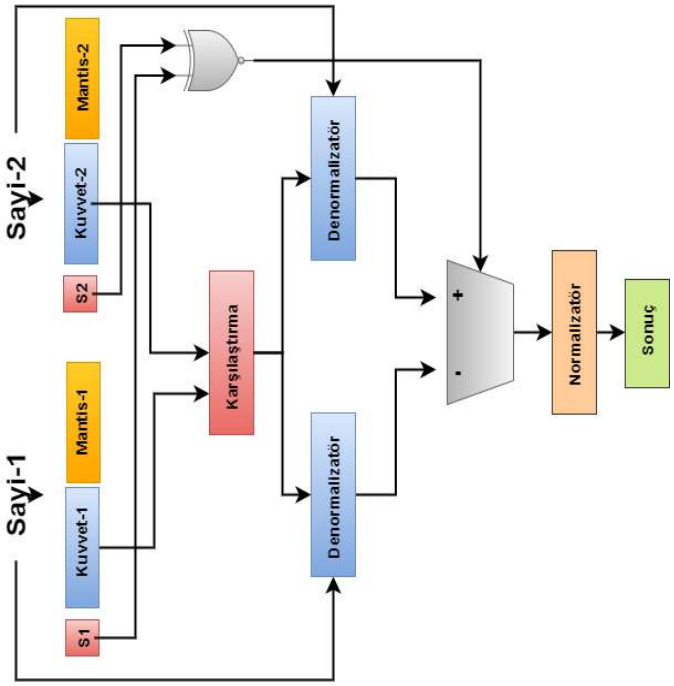
durumunda ise pozitif işaretli (işaret biti=0) sayı elde edilmektedir. Çarpma işlemi gerçekleştirilecek her iki sayının kuvvetlerinin sıfırdan farklı olması durumunda, kuvvetlerin cebirsel toplama işlemi ve iki sayının mantislerinin çarpımı ile de sonuca ait mantis ifadesi elde edilebilmektedir. Denklem 4.5'te çarpma işleminin matematiksel ifadesi gösterilmektedir.

$$\left((-\mathbf{1})^{s_1}.\,\mathbf{2}^{e_1+bias}.\{\mathbf{1},\boldsymbol{M_1}\}\right) * \left((-\mathbf{1})^{s_2}.\,\mathbf{2}^{e_2+bias}.\{\mathbf{1},\boldsymbol{M_2}\}\right) = \left((-\mathbf{1})^{s_1+s_2}.\,\mathbf{2}^{e_1+e_2+2.bias}.\left\{\mathbf{1},\boldsymbol{M_{çarpım}}\right\}\right) \quad \textbf{(4.5)}$$



Şekil 4.30. Kayan noktalı sayı sistemi aritmetik çarpma işlemi

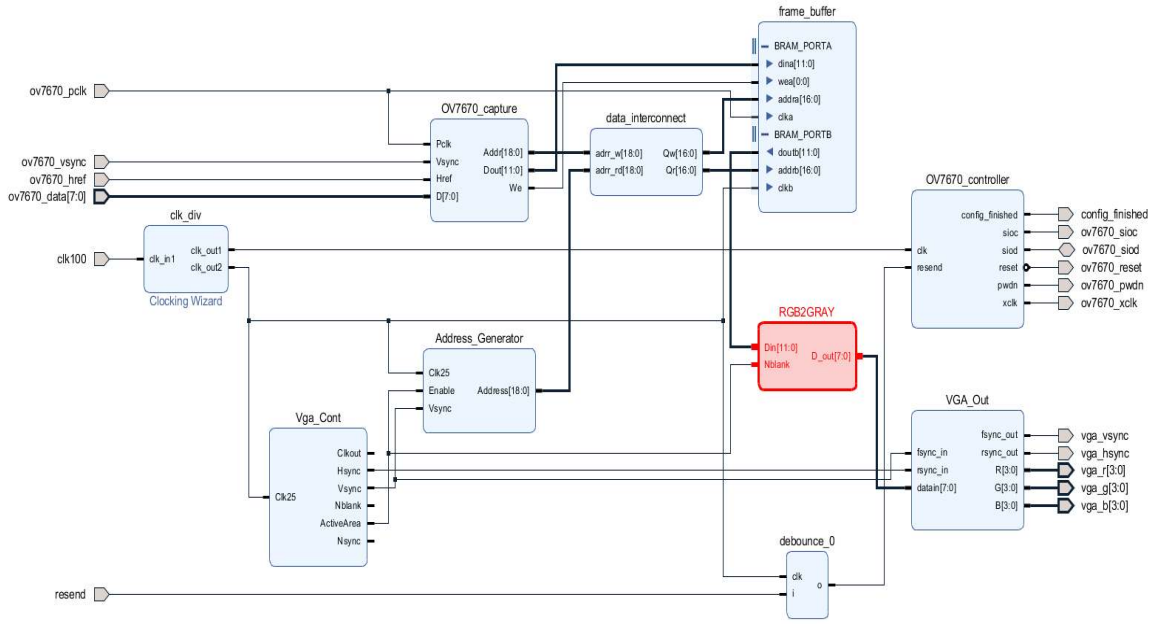
Şekil 4.30'da kayan noktalı sayı sisteminde aritmetik çarpma işleminin aşamaları gösterilmektedir. Gösterilen 16 bit kayan noktalı sayı sistemi temsilinde, pozitif gösterilebilecek en büyük sayı 65504 ve en küçük sayı ise $6{,}1035.10^{-5}$ bu aralığın dışındaki değerlerde "0111110000000000" ve "0000000000000000" değeri verilmektedir.



Şekil 4.31. Kayan noktalı sayı sisteminde aritmetik toplama işlemi aşamaları

Şekil 4.31'de kayan noktalı sayı sisteminde aritmetik toplama işleminin aşamaları gösterilmektedir. Kayan noktalı sayı sistemlerinde, toplama işleminin gerçekleştirileceği sayılarda noktaların yerinin farklı olması durumunda iki sayının kuvvet ve mantis ifadelerinde noktaların bulunduğu kısımların birbirine eşitlenmesi gerekmektedir. Toplama işleminde kuvvetler birbirine eşitlendikten sonra (küçük kuvvet büyüğe eşitlenir), toplanacak sayılar için denormalizasyon işlemi gerçekleştirilir ve ardından mantis kısımları işaret bitine bağlı olarak toplandıktan/çıkartıldıktan sonra tekrar normalize edilerek sonuç elde edilir.

Renkli görüntünün kayan noktalı sayı sistemi ile gerçekleştirilmesi için ilk aşamada, kameradan elde edilen RGB565 formatındaki piksel değerleri kayan noktalı sayı sistemine dönüştürülür. Daha sonra denklem 4.1'deki matematiksel ifade de bulunan katsayılar ile kayan noktalı sayı sistemine dönüştürülmüş pikseller, çarpma işlemi ve ardından toplama işleminin uygulanması ile renkli görüntünün gri tonlamalı görüntüye dönüştürülmesi gerçekleştirilmektedir.



Şekil 4.32. Gri tonlamalı görüntünün elde edilmesi

Şekil 4.32'de Kamera aracılığı ile elde edilen piksel değerleri hafızadan okunduktan sonra kırmızı renkli FPGA platformunda IP çekirdeğinin tasarımı ile RGB-Gri tonlamalı görüntü dönüşümü VGA monitörüne aktarılmaktadır. RGB-Gri tonlamalı görüntünün dönüşümü için gerçekleştirilen IP çekirdeği tabanlı uygulama, sabit ve kayan noktalı sayı sistemleri için ayrı ayrı tasarlanmıştır.

Şekil 4.33. RGB ve Gri tonlamalı video görüntüsü

Şekil 4.33'te FPGA platformunda IP bloklar kullanılarak gerçek zamanlı (Real time) elde edilen renkli ve gri tonlamalı videonun görüntüsü 640x480 çözünürlükte VGA ekrana aktarılarak gösterilmektedir.

Tez kapsamında yürütülen çalışmalarda hangi sayı sistemi ile çalışmanın daha uygun olacağını tespit etmek üzere, RGB- gri dönüşüm uygulaması üzerinde iki farklı sayı sistemi kullanılmış ve kıyaslamalar yapılmıştır. Sabit noktalı sayı sisteminin yeterli hassasiyette çözüm üretmesi durumunda, tasarımlarda sabit noktalı sayı sisteminin tercih edilebilmesi mümkün olacaktır.

Kayan ve sabit noktalı sayılar ile tasarlanan uygulamaların veri analizlerinin gerçekleştirilmesi için Analog Discovery 2 ölçüm ve test cihazı kullanılmıştır.

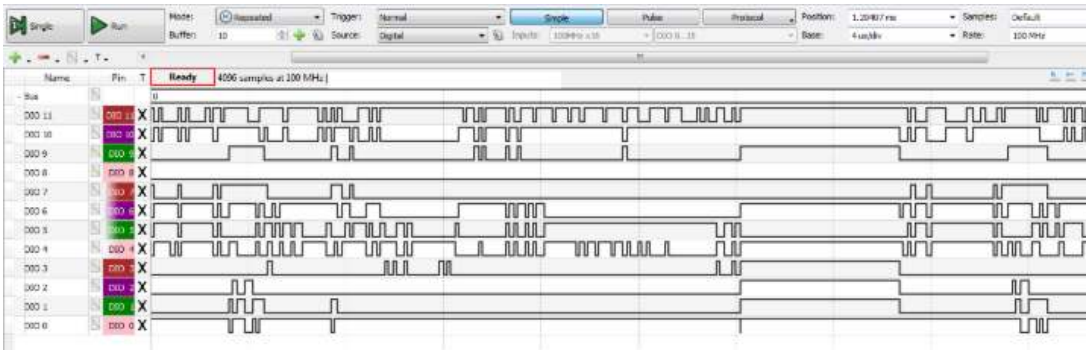
Analog Discovery 2

Analog Dicovery 2 (AD2) ölçüm cihazı, Osiloskop, sinyal jeneratörü, lojik analizör, dijital desen üretici, spektrum analizörü, voltmetre ve programlanabilir güç kaynağı gibi modüllerin bir araya getirilmesi ile üretilmiştir. Bu platformun sahip olduğu analog ve dijital giriş-çıkışlar sayesinde, tasarım devrelerinin sinyal ölçümleri, bu sinyallerin görüntülenmesini ve kaydedilebilmesini sağlamaktadır. Tüm işletim sistemlerinde WaveForm ücretsiz ara yüzü ile kullanılmaktadır.

Şekil 4.34. Analog Discovery 2 platformu[127]

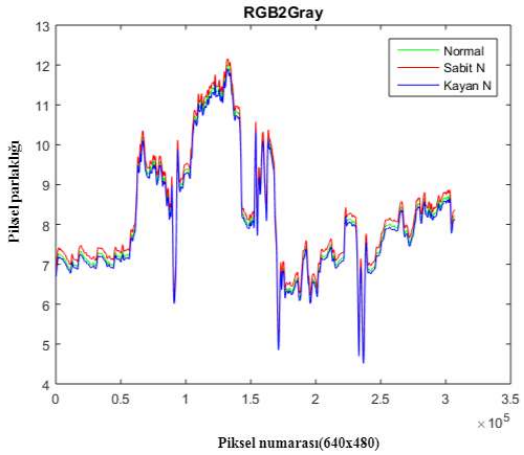
Şekil 4.34'te gösterilen analog Discovery 2 kullanılarak, FPGA platformunda gerçek zamanlı uygulamalarda veri analizinin gerçekleştirilebilmesi için 12 adet sayısal giriş-çıkış birimleri kullanılarak veriler bilgisayar ortamına aktarılmıştır. Kayan ve sabit noktalı sayı

sisteminde gerçekleştirilen uygulamaların kıyaslanması için kameradan elde edilen renkli görüntüye ait piksel değerleri AD2 kullanılarak kaydedilmiştir. Daha sonra bu veriler MATLAB ortamında kullanılarak, RGB-Gri tonlamalı dönüşümü, Filtre uygulamaları ve Sobel kenar algılama tekniği uygulamaları gerçekleştirilmiştir.



Şekil 4.35. Analog Discovery 2 platformunda sayısal verilerin elde edilmesi

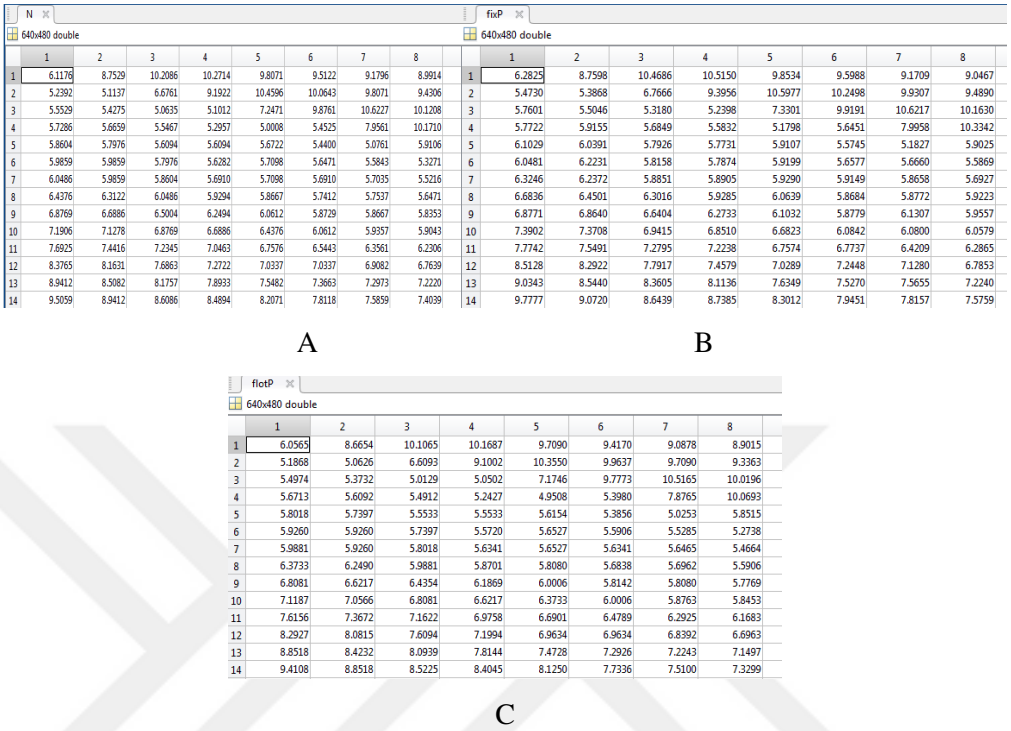
Şekil 4.35'te Analog Discovery 2 kullanılarak, FPGA platformundan elde edilen sayısal 12 bit görüntü piksel verileri gösterilmektedir. Program ara yüzünde tekrarlı elde edilen veriler maksimum 4096 adet gösterilebilmektedir. Ancak, kayıt ile 100 milyon örneklemeye kadar alınabilmektedir. Tez kapsamında gerçekleştirilecek uygulamalarda, kameradan 640x480 boyutlarında elde edilmektedir. 10x640x480 adet örnekleme alınarak, uygulamalar ortalama piksel değerleri ile gerçekleştirilmiştir.



Şekil 4.36. RGB-Gri tonlamalı görüntü dönüşümü piksel değerlerinin karşılaştırılması

Şekil 4.36'da Analog Discovery 2 kullanılarak, FPGA platformundan elde edilen 12 bit görüntü piksel verilerinin karşılaştırılması için MATLAB platformunda gerçekleştirilen görüntü ile sabit ve kayan noktalı sayı sistemi ile FPGA platformunda gerçekleştirilen uygulamaların sonuçları karşılaştırılmaktadır. X-ekseni 640x480 piksel ve y-ekseni bu piksellere ait yoğunlukları temsil

etmektedir. Aynı grafik üzerinde verilen üç ayrı sonuç, piksel eşleştirme için gerçekleştirilmiştir. Kayan noktalı sayı sisteminde gerçekleştirilen uygulamada elde edilen piksel yoğunlukları, sabit noktalı sayı sistemine göre daha hassas sonuçlar elde edilmiştir.



N — 640x480 double

	1	2	3	4	5	6	7	8
1	6.1176	8.7529	10.2086	10.2714	9.8071	9.5122	9.1796	8.9914
2	5.2392	5.1137	6.6761	9.1922	10.4596	10.0643	9.8071	9.4306
3	5.5529	5.4275	5.0635	5.1012	7.2471	9.8761	10.6227	10.1208
4	5.7286	5.6659	5.5467	5.2957	5.0008	5.4525	7.9561	10.1710
5	5.8604	5.7976	5.6094	5.6094	5.6722	5.4400	5.0761	5.9106
6	5.9859	5.9859	5.7976	5.6282	5.7098	5.6471	5.5843	5.3271
7	6.0486	5.9859	5.8604	5.6910	5.7098	5.6910	5.7035	5.5216
8	6.4376	6.3122	6.0486	5.9294	5.8667	5.7412	5.7537	5.6471
9	6.8769	6.6886	6.5004	6.2494	6.0612	5.8729	5.8667	5.8353
10	7.1906	7.1278	6.8769	6.6886	6.4376	6.0612	5.9357	5.9043
11	7.6925	7.4416	7.2345	7.0463	6.7576	6.5443	6.3561	6.2306
12	8.3765	8.1631	7.6863	7.2722	7.0337	7.0337	6.9082	6.7639
13	8.9412	8.5082	8.1757	7.8933	7.5482	7.3663	7.2973	7.2220
14	9.5059	8.9412	8.6086	8.4894	8.2071	7.8118	7.5859	7.4039

fixP — 640x480 double

	1	2	3	4	5	6	7	8
1	6.2825	8.7598	10.4686	10.5150	9.8534	9.5988	9.1709	9.0467
2	5.4730	5.3868	6.7666	9.3956	10.5977	10.2498	9.9307	9.4890
3	5.7601	5.5046	5.3180	5.2398	7.3301	9.9191	10.6217	10.1630
4	5.7722	5.9155	5.6849	5.5832	5.1798	5.6451	7.9958	10.3342
5	6.1029	6.0391	5.7926	5.7731	5.9107	5.5745	5.1827	5.9025
6	6.0481	6.2231	5.8158	5.7874	5.9199	5.6577	5.6660	5.5869
7	6.3246	6.2372	5.8851	5.8905	5.9290	5.9149	5.8658	5.6927
8	6.6836	6.4501	6.3016	5.9285	6.0639	5.8684	5.8772	5.9223
9	6.8771	6.8640	6.6404	6.2733	6.1032	5.8779	6.1307	5.9557
10	7.3902	7.3708	6.9415	6.8510	6.6823	6.0842	6.0800	6.0579
11	7.7742	7.5491	7.2795	7.2238	6.7574	6.7737	6.4209	6.2865
12	8.5128	8.2922	7.7917	7.4579	7.0289	7.2448	7.1280	6.7853
13	9.0343	8.5440	8.3605	8.1136	7.6349	7.5270	7.5655	7.2240
14	9.7777	9.0720	8.6439	8.7385	8.3012	7.9451	7.8157	7.5759

A B

flotP — 640x480 double

	1	2	3	4	5	6	7	8
1	6.0565	8.6654	10.1065	10.1687	9.7090	9.4170	9.0878	8.9015
2	5.1868	5.0626	6.6093	9.1002	10.3550	9.9637	9.7090	9.3363
3	5.4974	5.3732	5.0129	5.0502	7.1746	9.7773	10.5165	10.0196
4	5.6713	5.6092	5.4912	5.2427	4.9508	5.3980	7.8765	10.0693
5	5.8018	5.7397	5.5533	5.5533	5.6154	5.3856	5.0253	5.8515
6	5.9260	5.9260	5.7397	5.5720	5.6527	5.5906	5.5285	5.2738
7	5.9881	5.9260	5.8018	5.6341	5.6527	5.6341	5.6465	5.4664
8	6.3733	6.2490	5.9881	5.8701	5.8080	5.6838	5.6962	5.5906
9	6.8081	6.6217	6.4354	6.1869	6.0006	5.8142	5.8080	5.7769
10	7.1187	7.0566	6.8081	6.6217	6.3733	6.0006	5.8763	5.8453
11	7.6156	7.3672	7.1622	6.9758	6.6901	6.4789	6.2925	6.1683
12	8.2927	8.0815	7.6094	7.1994	6.9634	6.9634	6.8392	6.6963
13	8.8518	8.4232	8.0939	7.8144	7.4728	7.2926	7.2243	7.1497
14	9.4108	8.8518	8.5225	8.4045	8.1250	7.7336	7.5100	7.3299

C

Şekil 4.37. RGB-Gri tonlamalı dönüşümlerine ait (A)'da MATLAB, (B)'de sabit noktalı sayı ve (C)'de kayan noktalı sayı sistemi ile gerçekleştirilen piksel değerleri

Şekil 4.37'de RGB-Gri tonlamalı görüntüye ait piksel değerleri gösterilmektedir. Analog Discovery 2 kullanılarak renkli görüntünün piksel değerleri bilgisayar platformuna aktarıldıktan sonra, denklem 4.1 kullanılarak RGB-Gri tonlamalı görüntü dönüşümü gerçekleştirilmiştir. Bu dönüşüm aynı zamanda FPGA platformunda, sabit ve kayan noktalı sayılar için gerçekleştirilip, uygulama çıktıları AD2 ile dış ortama aktarılmıştır. Elde dilen piksel değerlerinin kıyaslanmasında, kayan noktalı sayı sisteminde elde edilen değerler, sabit noktalı sayı sistemine göre MATLAB platformunda elde edilen piksellere daha yakın değerlere ve hassasiyete sahiptir.

Tablo 4.2. FPGA Güç Tüketimi Miktarının Karşılaştırılması

Kaynaklar	Sabit noktalı (Watt)	Kayan noktalı (Watt)
Signals	2,580	2,960
Logic	0,776	1,713
BRAM	0,907	0,907
DSP	0,802	1,165

Tablo 4.2'de RGB-Gri tonlamalı dönüşümü uygulamasının, kayan ve sabit noktalı sayı sistemleri ile gerçekleştirilmesi ile elde edilen güç tüketimi miktarları karşılaştırılmıştır. Aynı bit uzunluğunda (16 bit) ve 640x480 boyutlarında video görüntünün gri tonlamalı görüntüye dönüştürülmesinde sabit noktalı sayı sistemlerinde daha az güç tüketimi ile gerçekleştirilmektedir. Kayan noktalı sayı sistemlerinde işlem yükünün sabit noktalı sayı sistemlerine göre daha fazla olmasından dolayı, gerçekleştirilen uygulamada daha fazla güç tüketimi ile uygulama gerçekleştirilmektedir.

Tablo 4.3. FPGA Kaynak Kullanımı Karşılaştırma Tablosu

Kaynaklar	Sabit noktalı	Kayan noktalı
LUTs	319	433
Slice Register	218	218
BRAM	45.5	44,5
DSP	3	3

Tablo 4.3'te RGB-Gri tonlamalı dönüşümü uygulamasının, kayan ve sabit noktalı sayı sistemleri ile gerçekleştirilmesi ile kullanılan FPGA kaynak miktarları karşılaştırılmıştır. Aynı bit uzunluğunda (16 bit) ve 640x480 boyutlarında gri tonlamalı görüntüye dönüşümde sabit ve kayan noktalı sayı sistemlerinde aynı miktarda BRAM kullanılmıştır. Eşit miktarda DSP kullanılmış fakat DSP kullanımında güç tüketimi sabit noktalı sayı sistemlerinde daha az miktardadır. Her iki sayı sisteminde kaynak kullanımının az olması, uygulamanın IP çekirdekleri kullanılarak gerçekleştirilmesinin etkisi de mevcuttur.

Tablo 4.4. Renkli ve gri tonlamalı görüntü için kullanılan FPGA kaynak miktarları

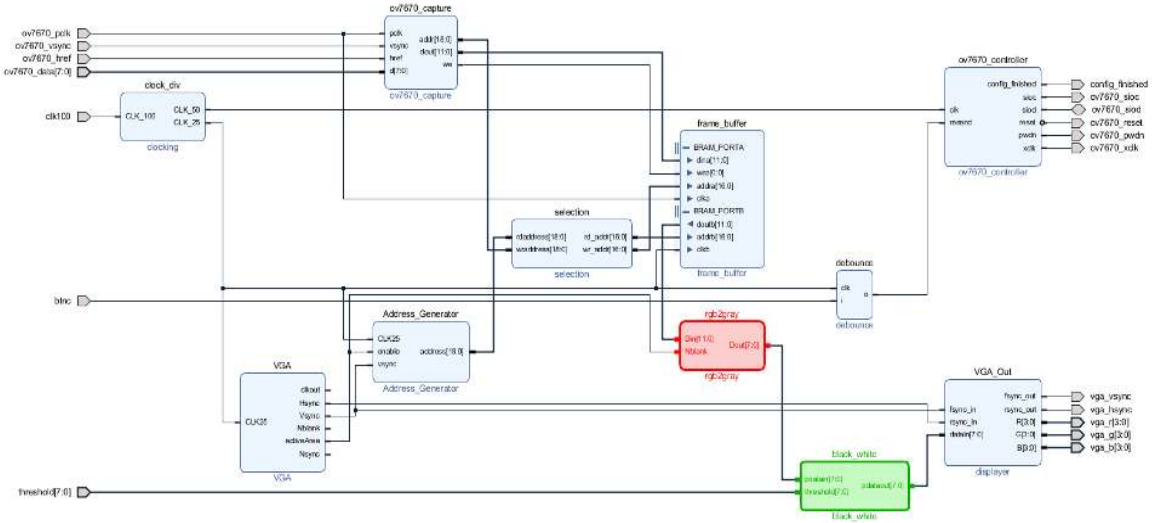
Kaynaklar	VHDL kod ile RBG Video	IP Blok ile RBG Video	VHDL kod ile gri tonlamalı Video	IP Blok ile gri tonlamalı Video
LUTs	350	295	307	304
Slice Register	230	225	225	225
BRAM	44	44	44	44
IO	35	35	35	35

Tablo 4.4'te Renkli ve renkli görüntünün gri tonlamalı görüntüye dönüştürülmesinde sabit noktalı sayı sistemi kullanılarak gerçekleştirilen FPGA kaynak kullanımı verilmektedir. VHDL ve IP çekirdekler ile renkli ve gri tonlamalı görüntüler için oluşturulan çalışmalarda, IP bloklar ile oluşturulan projelerde kullanılan FPGA kaynak miktarlarının daha az olduğu görülmektedir. Aynı zamanda IP blokları ile oluşturulan projelerde, IP çekirdeklerinin giriş çıkışlarının görsel olarak da görüntülenebilmesinden dolayı, tasarımcıya ara bağlantılarda kolaylıklar sağlayacaktır.

VHDL/Verilog dillerinde gerçekleştirilen uygulamalarda olduğu gibi IP çekirdekleri de aynı tasarım içerisinde VHDL veya Verilog dillerinde de oluşturulabilmektedir.

4.3.3. Gerçek Zamanlı Video Görüntüsüne Eşik Değer Uygulanması

Gri tonlamalı görüntünün elde edilmesi işleminden sonra, kullanıcının online olarak belirleyebileceği bir değer ile kameradan alınan görüntüler bu sayı ile karşılaştırılarak siyah ve beyaz (0 yada 255) piksel değerlerine sahip video görüntüsü elde edilmiştir. Bu eşik değerin uygulanması sensörler veya farklı yapılar ile kontrol edilerek kullanılabilmektedir.



Şekil 4.38. Siyah-Beyaz görüntünün elde edilmesi

Şekil 4.38'de FPGA IP çekirdekleri kullanılarak siyah-beyaz görüntünün elde edilmesine ait tasarım gösterilmektedir. Kırmızı IP çekirdeği renkli görüntünün gri tonlamalı görüntüye ve yeşil IP çekirdeği ile de gri tonlamalı görüntü siyah beyaz görüntüye dönüştürülmektedir.

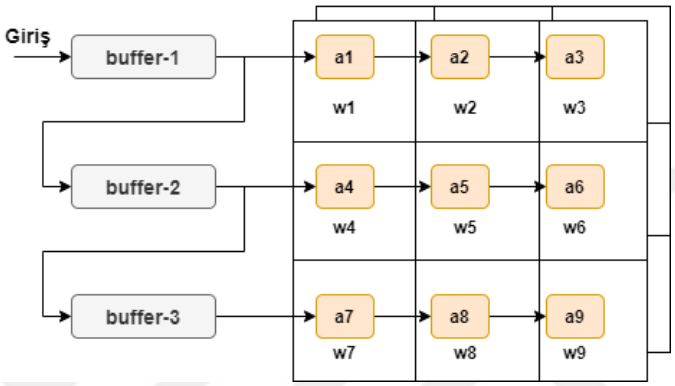
Tablo 4.5. Siyah-Beyaz görüntü için FPGA Kaynak kullanımı

Kaynaklar	Kullanılan	Mevcut
LUTs	326 (%0,61)	53200
Slice Register	218(%0,2)	106400
BRAM	44,5(%31,79)	140
IO	41(%20,5)	200

Tablo 4.5'te OV7670 kamera ve FPGA Zedboard platformu kullanılarak gerçekleştirilen, siya-beyaz görüntünün elde edilmesinde kullanılan FPGA kaynak kullanımı gösterilmektedir.

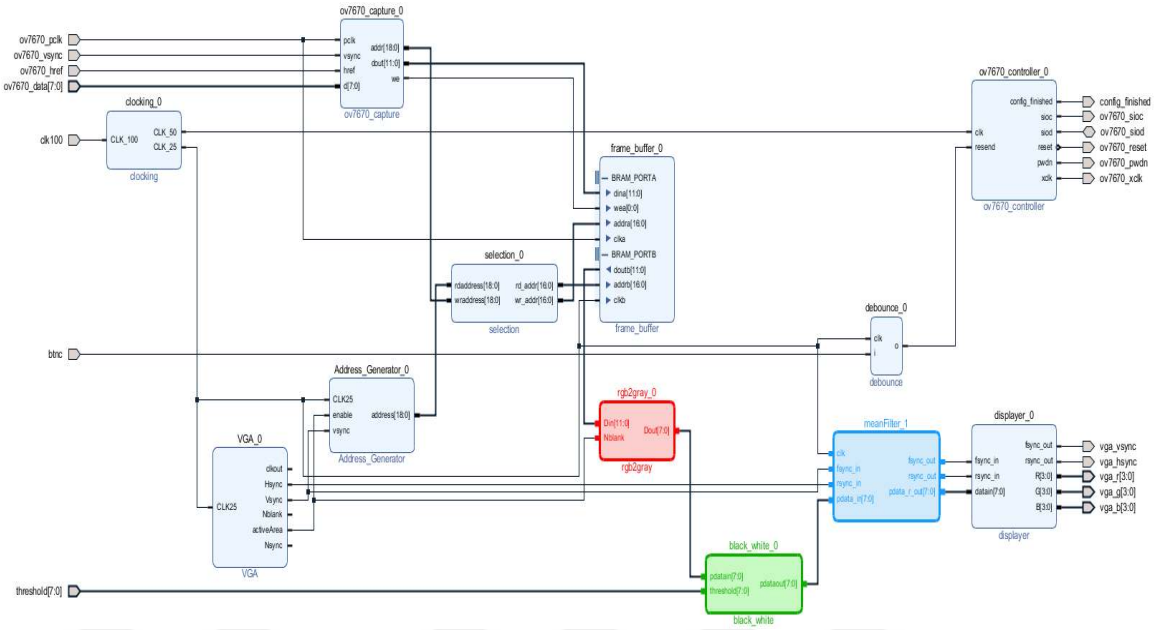
4.3.4. Görüntünün Pürüzsüzleştirilmesi

Pürüzsüzleştirme (smooting) veya yumuşatma olarak da adlandırılan görüntü filtreleme işlemidir. Genellikle filtre matrisi görüntüye uygulanarak resim üzerinde bulunan gürültünün yok edilmesi için kullanılmaktadır. **Şekil 4.40**'da kameradan elde edilen görüntüler, gri tonlamalı görüntüye dönüştürüldükten sonra eşik değer uygulaması gerçekleştirilmiştir. Bu işlemlerin ardından filtre matrisinin (3x3 boyutlu) görüntü üzerinde buffer yapıları kullanılarak hafızada tutulan piksel değerlerine uygulanması ile gerçekleştirilmektedir.



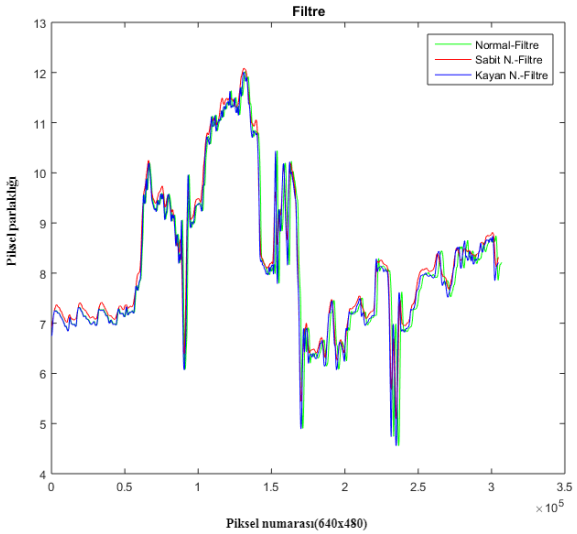
Şekil 4.39. Piksel hafıza birimi

Şekil 4.39'da satır bilgisine ait piksel değerlerine ait buffer yapısı gösterilmektedir. Buffer yapısı ile iki adet satır (2x640) bilgisi hafıza tutulmakta ve üçüncü satırın ilk üç piksel değeri ile filtreleme işlemleri gerçekleştirilmektedir. Gerçekleştirilen işlem ile her piksel için yeni bir renk değeri hesaplanmaktadır. Üçüncü satır pikselleri ile diğer iki hafıza biriminde bulunan pikseller eş zamanlı olarak elde edilecektir. Böylelikle üç satıra ait piksel değerleri (a1…a9) ile Sobel filtre matris katsayıları (w1…w9) kullanılarak işlem yapılabilmektedir. Bu işlem ile görüntüye; Medyan, Ortalama (Mean), Gauss ve bilateral filtreler gibi en çok kullanılan bulanıklaştırma filtreleri uygulanabilmektedir.



Şekil 4.40. Görüntünün pürüzsüzleştirilmesi

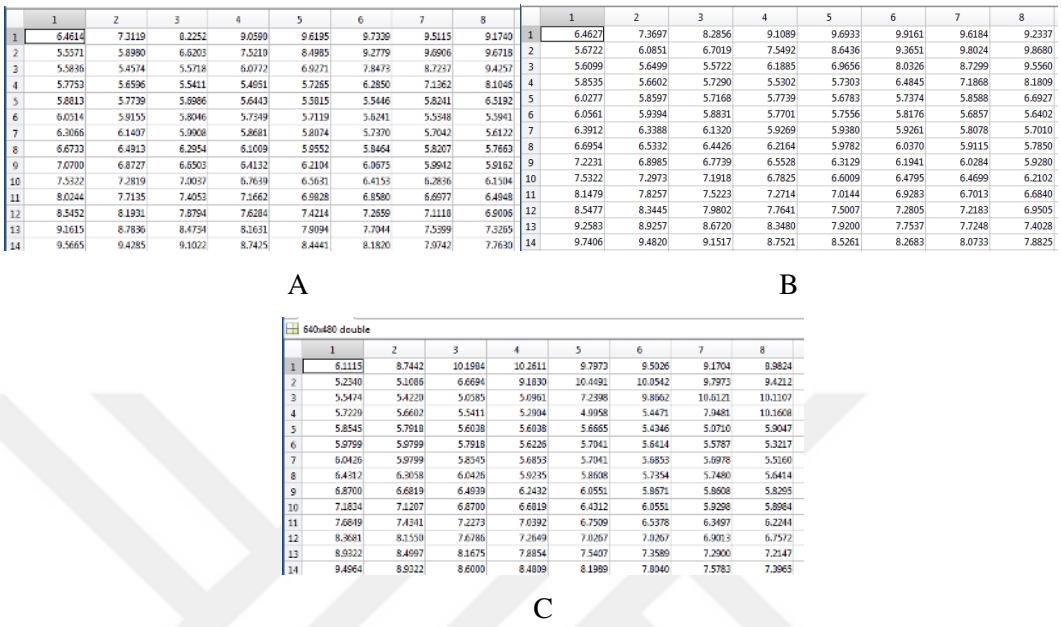
Görüntü pürüzsüzleştirme işleminin gerçekleştirilmesi için gerçekleştirilen **Şekil 4.40**'ta bulunan tasarımda ortalama filtre tekniği uygulanmıştır. Gerçek zamanlı eşik değer tekniği uygulandıktan sonra video görüntüsüne ortalama filtre tekniği uygulanmıştır. Bu çalışma ile gerçek zamanlı kameradan elde edilen video görüntülerinde gürültülerin giderilmesi amaçlanmıştır.



Şekil 4.41. Görüntünün pürüzsüzleştirilmesinde elde edilen sonuçlara ait piksel değerleri dağılımının karşılaştırılması

Şekil 4.41'de AD2 ile elde FPGA platformundan bilgisayar ortamına aktarılan görüntü pürüzsüzleştirilmesinde sabit ve kayan noktalı sayı sisteminin karşılaştırılmasında elde edilen

sonuçlar karşılaştırılmıştır. X-ekseni 640x480 piksel ve y-ekseni bu piksellere ait yoğunlukları temsil etmektedir. Normal ile etiketlendirilen sonuçlar MATLAB ortamında karşılaştırma amacıyla RGB görüntüye ait piksellerin FPGA platformundan elde edilmesi ile gerçekleştirilmiştir.



A

	1	2	3	4	5	6	7	8
1	6.4614	7.3119	8.2252	9.0590	9.6195	9.7339	9.5115	9.1740
2	5.5571	5.8980	6.6203	7.5210	8.4985	9.2779	9.6906	9.6718
3	5.5836	5.4574	5.5718	6.0772	6.9271	7.8473	8.7237	9.4257
4	5.7753	5.6596	5.5411	5.4951	5.7265	6.2850	7.1362	8.1046
5	5.8813	5.7739	5.6986	5.6443	5.5815	5.5446	5.8241	6.5192
6	6.0314	5.9155	5.8046	5.7549	5.7119	5.6241	5.5548	5.5941
7	6.3066	6.1497	5.9908	5.8681	5.8074	5.7370	5.7042	5.6122
8	6.6733	6.4913	6.2954	6.1009	5.9552	5.8464	5.8207	5.7663
9	7.0700	6.8727	6.6503	6.4132	6.2104	6.0675	5.9942	5.9162
10	7.5322	7.2819	7.0037	6.7639	6.5631	6.4153	6.2836	6.1504
11	8.0244	7.7135	7.4053	7.1662	6.9828	6.8580	6.6977	6.4948
12	8.5452	8.1931	7.8794	7.6284	7.4214	7.2659	7.1118	6.9006
13	9.1615	8.7836	8.4734	8.1631	7.9094	7.7044	7.5399	7.3265
14	9.5665	9.4285	9.1022	8.7425	8.4441	8.1820	7.9742	7.7630

B

	1	2	3	4	5	6	7	8
1	6.4627	7.3697	8.2856	9.1089	9.6933	9.9161	9.6184	9.2337
2	5.6722	6.0851	6.7019	7.5492	8.6436	9.3651	9.8024	9.8680
3	5.6099	5.6499	5.5722	6.1885	6.9656	8.0326	8.7299	9.5560
4	5.8535	5.6602	5.7290	5.5302	5.7303	6.4845	7.1868	8.1809
5	6.0277	5.8597	5.7168	5.7739	5.6783	5.7374	5.8588	6.6927
6	6.0561	5.9394	5.8831	5.7701	5.7556	5.8176	5.6857	5.6402
7	6.3912	6.3388	6.1320	5.9269	5.9380	5.9261	5.8078	5.7010
8	6.6954	6.5332	6.4426	6.2164	5.9782	6.0370	5.9115	5.7850
9	7.2231	6.8985	6.7739	6.5528	6.3129	6.1941	6.0284	5.9280
10	7.5322	7.2973	7.1918	6.7825	6.6009	6.4795	6.4699	6.2102
11	8.1479	7.8257	7.5223	7.2714	7.0144	6.9283	6.7013	6.6840
12	8.5477	8.3445	7.9802	7.7641	7.5007	7.2805	7.2183	6.9505
13	9.2583	8.9257	8.6720	8.3480	7.9200	7.7537	7.7248	7.4028
14	9.7406	9.4820	9.1517	8.7521	8.5261	8.2683	8.0733	7.8825

C

640x480 double

	1	2	3	4	5	6	7	8
1	6.1115	8.7442	10.1984	10.2611	9.7973	9.5026	9.1704	8.9824
2	5.2340	5.1086	6.6694	9.1830	10.4491	10.0542	9.7973	9.4212
3	5.5474	5.4220	5.0585	5.0961	7.2398	9.8662	10.6121	10.1107
4	5.7229	5.6602	5.5411	5.2904	4.9958	5.4471	7.9481	10.1608
5	5.8545	5.7918	5.6038	5.6038	5.6665	5.4346	5.0710	5.9047
6	5.9799	5.9799	5.7918	5.6226	5.7041	5.6414	5.5787	5.3217
7	6.0426	5.9799	5.8545	5.6853	5.7041	5.6853	5.6978	5.5160
8	6.4312	6.3058	6.0426	5.9235	5.8608	5.7354	5.7480	5.6414
9	6.8700	6.6819	6.4939	6.2432	6.0551	5.8671	5.8608	5.8295
10	7.1834	7.1207	6.8700	6.6819	6.4312	6.0551	5.9298	5.8984
11	7.6849	7.4341	7.2273	7.0392	6.7509	6.5378	6.3497	6.2244
12	8.3681	8.1550	7.6786	7.2649	7.0267	7.0267	6.9013	6.7572
13	8.9322	8.4697	8.1675	7.8854	7.5407	7.3589	7.2900	7.2147
14	9.4964	8.9322	8.6000	8.4809	8.1989	7.8040	7.5783	7.3965

Şekil 4.42. Görüntünün pürüzsüzleştirilmesinde (A)'da MATLAB, (B)'de sabit noktalı sayı ve (C)'de kayan noktalı sayı sistemi ile gerçekleştirilen piksel değerleri

Şekil 4.42'de AD2 ile elde FPGA platformundan bilgisayar ortamına aktarılan görüntü pürüzsüzleştirilmesinde sabit ve kayan noktalı sayı sisteminin karşılaştırılmasında elde edilen piksel değerlerine ait sonuçlar gösterilmektedir.

Tablo 4.6. Görüntünün pürüzsüzleştirilmesi işleminde kullanılan FPGA kaynak miktarları

Kaynaklar	**Kullanılan**	**Mevcut**
LUTs	530 (%0,9)	53200
Slice Register	354 (%0,3)	106400
BRAM	44,5 (%31,8)	140
IO	41 (%20,5)	200

Tablo 4.6'da FPGA platformun görüntünün pürüzsüzleştirilmesi için kullanılan kaynak miktarları gösterilmektedir. Çok az sayıda FPGA kaynak kullanımı ile ortalama filtre işlemleri gerçekleştirilmiştir.

4.3.5. Sobel Kenar Algılama Tekniğinin Video Görüntüye Uygulanması

Kenar algılama, görüntü işlemede en önemli uygulamalardan biridir. Kenar algılama, görüntü ayrıştırma (segmentasyon), özellik çıkarma, nesne veya sınır tanımlaması için kullanılabilen, bir görüntüde kenar veya çizginin varlığını belirleyen bir filtreleme tekniğidir. Kenar algılama uygulamalarında genelde kullanılan üç adet yöntem bulunmaktadır.

- Sobel
- Canny
- Prewit Kenar Algılama algoritmalarıdır.

-1	0	1
-2	0	2
-1	0	1

Yatay Sobel

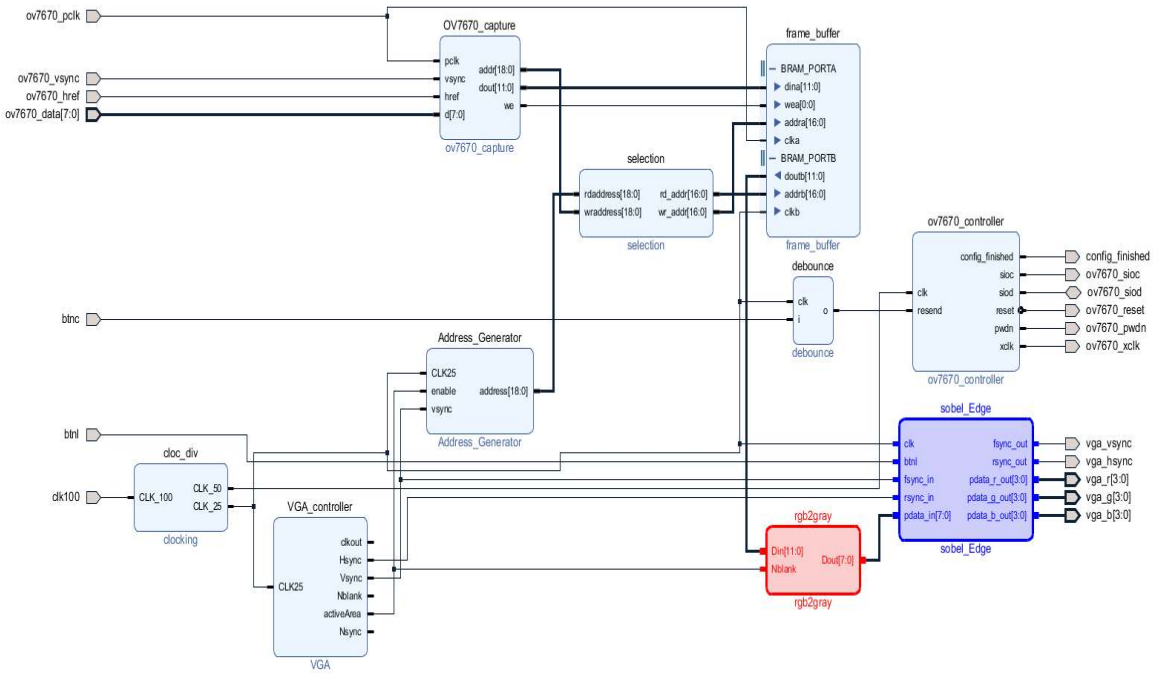
-1	-2	-1
0	0	0
1	2	1

Dikey Sobel

Şekil 4.43. Sobel filtre matrisleri

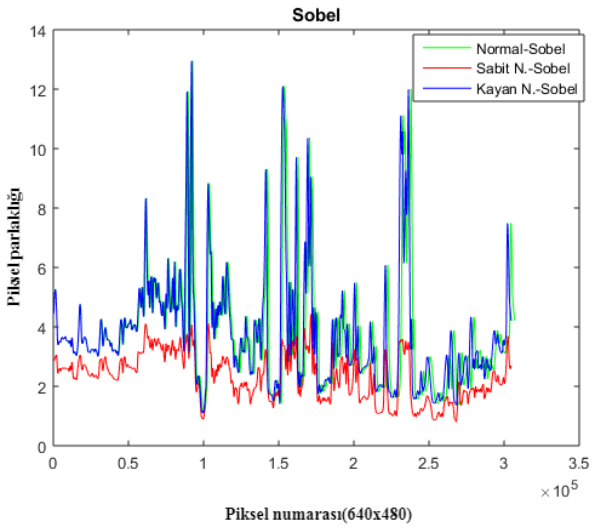
$$|S| = \sqrt{{S_X}^2 + {S_Y}^2} \tag{4.6}$$

Yatay ve dikey Sobel filtrelerinin görüntüye uygulanması ile Sx ve Sy bileşenleri elde edilmektedir. **Şekil 4.43**'te yatay ve dikey Sobel filtreleri matrisleri gösterilmektedir. Her iki filtre matrisinin görüntüye uygulanmasından sonra denklem 4.6 aracılığı ile Sobel filtreleme işlemi gerçekleştirilmiş olmaktadır. Sobel operatörü görüntü işleme ve bilgisayarla görmede kullanılır, özellikle kenarları vurgulayan bir kenar algılama tekniğidir. RGB verilerini gri tonlamaya dönüştürme ve ardından gerçek zamanlı videonun kenarlarını tespit etmek için Sobel filtresi uygulanması için 3x3 boyutunda yatay ve dikey piksel değerlerinin elde edilmesi gerekmektedir. Gri tonlamalı görüntü elde edildikten sonra ilk iki satır değeri (VGA monitör dizilimi) geçici hafızada saklanır (line buffer) ve üçüncü satırın piksel değerleri ile eş zamanlı olarak Sobel filtre matrisleri uygulanır. Üçüncü satırın son piksel değeri işlem gördükten sonra hafızada tutulan satırlar güncellenir dördüncü ve beşinci satırlar hafızaya alınır ve altıncı satırdaki piksellerin gelmesi ile filtreleme işlemi gerçekleştirilir ve tüm satırlar bitene kadar bu işlemler sürdürülür.



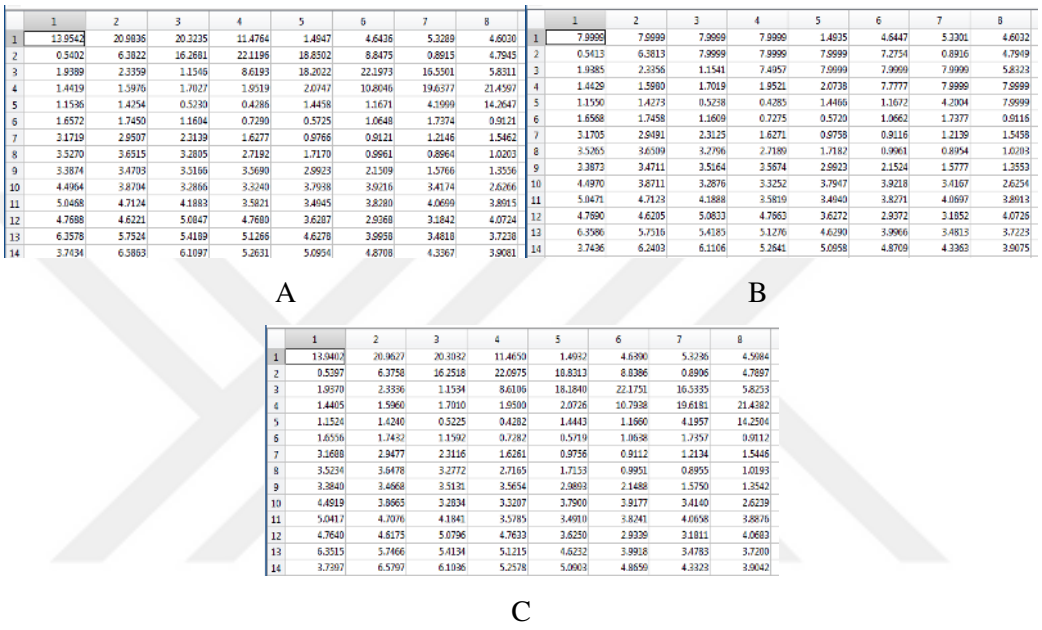
Şekil 4.44. IP bloklar ile gerçek zamanlı video görüntünün kenarlarının çıkartılması

Kenar algılama, görüntü işlemede en önemli uygulamalardan biridir. IP çekirdekler ile renkli görüntü, ardından gri tonlamalı görüntüye dönüştürme ve gerçek zamanlı video akışının kenarlarının VGA monitöründe gösterilmesi için gerçekleştirilen tasarım **Şekil 4.44**'te gösterilmektedir. Sobel filtreleme işlemi **Şekil 4.39**'da bulunan buffer yapısı kullanılarak gerçekleştirilmiştir.



Şekil 4.45. Sobel kenar algılama tekniğinin uygulanması ile elde edilen sonuçlara ait piksel değerleri dağılımının karşılaştırılması

Şekil 4.45'te Sobel kenar algılama tekniğinin uygulanması ile elde edilen sonuçlara ait piksel değerlerinin dağılımı karşılaştırılmıştır. Elde edilen değerlerde sabit noktalı sayı sistemi ile MATLAB ortamında ve kayan nokta temsiline göre farklı sonuçlar elde edilmiştir. X-ekseni 640x480 piksel ve y-ekseni bu piksellere ait yoğunlukları temsil etmektedir. Piksel eşleştirme ile kayan noktalı sayı sisteminin kullanımında, sabit noktalı sayı sistemine göre daha fazla olduğu görülmektedir. Bu sonuçlar, ileri seviye ve sayı hassasiyetinin önemli olduğu uygulamalarda, kayan noktalı sayı sisteminin kullanılması önerilmektedir.



	1	2	3	4	5	6	7	8
1	13.9542	20.9836	20.3235	11.4764	1.4947	4.6436	5.3289	4.6020
2	0.5402	6.3822	16.2681	22.1196	18.8502	8.8475	0.8915	4.7945
3	1.9389	2.3359	1.1546	8.6193	18.2022	22.1973	16.5501	5.8311
4	1.4419	1.5976	1.7027	1.9519	2.0747	10.8046	19.6377	21.4597
5	1.1536	1.4254	0.5280	0.4286	1.4458	1.1671	4.1999	14.2647
6	1.6572	1.7450	1.1604	0.7290	0.5725	1.0648	1.7374	0.9121
7	3.1719	2.9507	2.3139	1.6277	0.9766	0.9121	1.2146	1.5462
8	3.5270	3.6515	3.2805	2.7192	1.7170	0.9961	0.8964	1.0203
9	3.3874	3.4703	3.5166	3.5690	2.9923	2.1509	1.5766	1.3556
10	4.4964	3.8704	3.2866	3.3240	3.7938	3.9216	3.4174	2.6266
11	5.0468	4.7124	4.1883	3.5821	3.4045	3.8280	4.0699	3.8915
12	4.7688	4.6221	5.0847	4.7680	3.6287	2.9368	3.1842	4.0724
13	6.3578	5.7524	5.4189	5.1266	4.6278	3.9958	3.4818	3.7238
14	3.7434	6.5863	6.1097	5.2631	5.0954	4.8708	4.3367	3.9081

A

	1	2	3	4	5	6	7	8
1	7.9999	7.9999	7.9999	7.9999	1.4935	4.6447	5.3301	4.6032
2	0.5413	6.3813	7.9999	7.9999	7.9999	7.2754	0.8916	4.7949
3	1.9385	2.3356	1.1541	7.4957	7.9999	7.9999	7.9999	5.8323
4	1.4429	1.5980	1.7019	1.9521	2.0738	7.7777	7.9999	7.9999
5	1.1550	1.4273	0.5298	0.4285	1.4466	1.1672	4.2004	7.9999
6	1.6568	1.7458	1.1609	0.7275	0.5720	1.0662	1.7377	0.9116
7	3.1705	2.9491	2.3125	1.6271	0.9758	0.9116	1.2139	1.5458
8	3.5265	3.6509	3.2796	2.7189	1.7182	0.9961	0.8954	1.0203
9	3.3873	3.4711	3.5164	3.5674	2.9923	2.1524	1.5777	1.3553
10	4.4970	3.8711	3.2876	3.3252	3.7947	3.9218	3.4167	2.6254
11	5.0471	4.7123	4.1888	3.5819	3.4040	3.8271	4.0697	3.8913
12	4.7690	4.6205	5.0833	4.7663	3.6272	2.9372	3.1852	4.0726
13	6.3586	5.7516	5.4185	5.1276	4.6290	3.9966	3.4813	3.7225
14	3.7436	6.2403	6.1106	5.2641	5.0958	4.8709	4.3363	3.9075

B

	1	2	3	4	5	6	7	8
1	13.9402	20.9627	20.3032	11.4650	1.4932	4.6890	5.3236	4.5084
2	0.5397	6.3758	16.2518	22.0975	18.8313	8.8386	0.8906	4.7897
3	1.9370	2.3336	1.1534	8.6106	18.1840	22.1751	16.5335	5.8253
4	1.4405	1.5960	1.7010	1.9500	2.0726	10.7938	19.6181	21.4382
5	1.1524	1.4240	0.5225	0.4282	1.4443	1.1660	4.1957	14.2504
6	1.6556	1.7432	1.1592	0.7282	0.5719	1.0638	1.7357	0.9112
7	3.1688	2.9477	2.3116	1.6261	0.9756	0.9112	1.2134	1.5446
8	3.5234	3.6478	3.2772	2.7165	1.7153	0.9951	0.8955	1.0193
9	3.3840	3.4668	3.5131	3.5654	2.9893	2.1488	1.5750	1.3542
10	4.4919	3.8665	3.2834	3.3207	3.7900	3.9177	3.4140	2.6239
11	5.0417	4.7076	4.1841	3.5785	3.4910	3.8241	4.0658	3.8876
12	4.7640	4.6175	5.0796	4.7633	3.6250	2.9339	3.1811	4.0683
13	6.3515	5.7466	5.4134	5.1215	4.6232	3.9918	3.4783	3.7200
14	3.7397	6.5797	6.1036	5.2578	5.0903	4.8659	4.3323	3.9042

C

Şekil 4.46. Sobel kenar algılama (A)'da MATLAB, (B)'de sabit noktalı sayı ve (C)'de kayan noktalı sayı sistemi ile gerçekleştirilen piksel değerleri

Şekil 4.46'da Sobel kenar algılama tekniğinin (A)'da MATLAB, (B)'de sabit noktalı sayı ve (C)'de kayan noktalı sayı sistemi ile uygulanması ile elde edilen sonuçlara ait piksel değerleri gösterilmektedir. Elde edilen sonuçlara göre, ileri seviye uygulamalarda sayı hassasiyetinin kayan noktalı sayı sistemlerinde daha uygun olduğu görülmektedir.

Tablo 4.7. FPGA Sobel filtre kaynak kullanımları

Kaynaklar	Bu çalışmada	[131]	[132]	[133]
LUTs	544	5677	7301	2335
Slice Register	366	7919	12311	3647
BRAM	45.5	7	12	125
IO	35	52	32	32

Tablo 4.7'de FPGA zedboard (XC7Z020-CLG484) platformu kullanılarak gerçekleştirilen çalışma ile literatürde aynı platformu kullanıp gerçekleştirilen Sobel filtre uygulamalarının kaynak kullanımı verilmektedir. IP çekirdekler ve sabit noktalı sayı formatını kullanılarak gerçekleştirilen uygulama, diğer çalışmalara kıyasla daha az FPGA kaynak kullanımı ile gerçekleştirilmiştir.

Şekil 4.47. FPGA Sobel kenar algılama VGA monitör görüntüsü

Şekil 4.47'de FPGA IP çekirdekleri ile sobel kenar algılama için oluşturulmuş uygulamaya ait video görüntüsü gösterilmektedir. Bu uygulama ile gerçek zamanlı uygulamalarda, görüntünün ayrıştırılması (segmentasyon), özellik çıkarma, nesne veya sınır tanımlaması, görüntüde kenarların veya çizgilerin belirlenmesi için kullanılabilmektedir.

4.3.6. Canny Kenar Algılama Tekniğinin Video Görüntüye Uygulanması

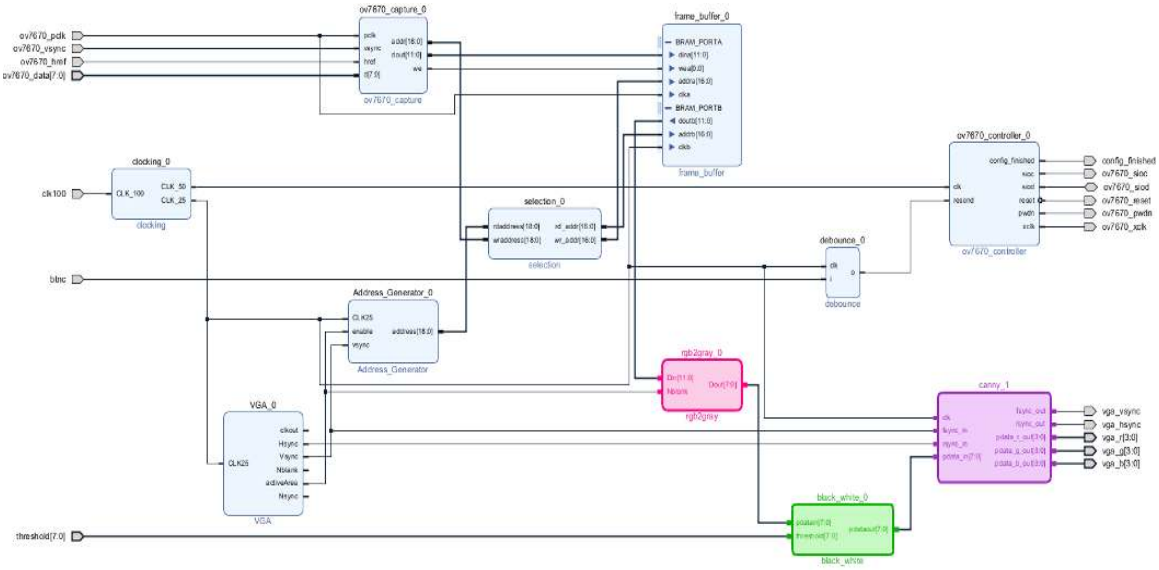
Gerçek zamanlı video görüntüde Canny kenar algoritmasının uygulanması Sobel kenar operatörüne göre kullanılan filtre farklıdır. Canny, görüntülerdeki çok çeşitli kenarları algılamak için kullanılan bir kenar algılama operatörüdür. Canny kenar algılama işlemi 2x2'lik yatay ve dikey uygulanan filtre matrislerinin uygulanması ile gerçekleştirilmektedir.

1	-1
0	-C

1	0
-1	-C

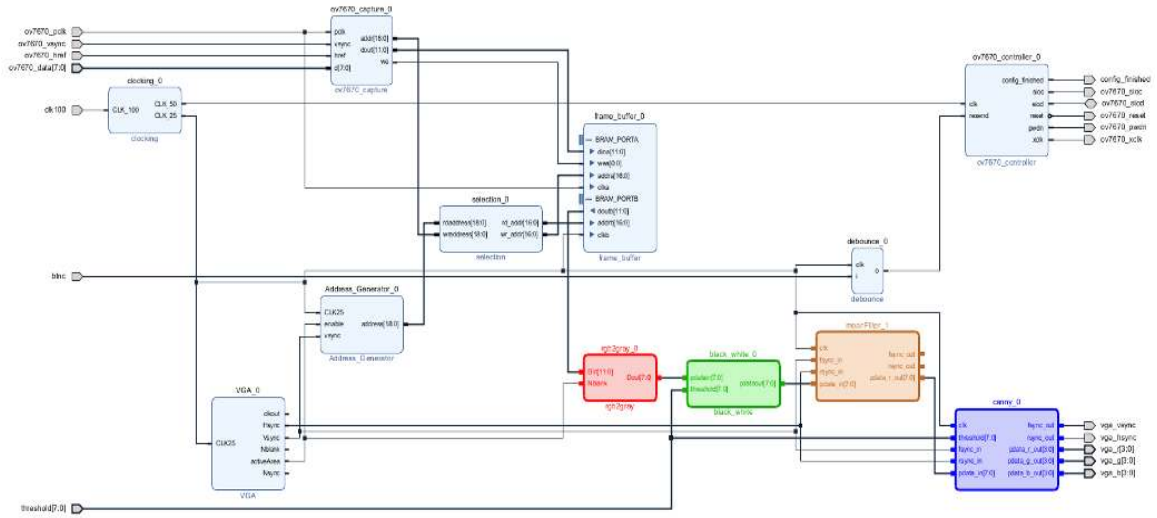
Şekil 4.48. Canny operatörü filtre matrisleri

Şekil 4.48'de gösterilen "C" değeri uygulamaya göre istenilen değerler verilebileceği gibi genellikle 0.707 değeri kullanılmaktadır[128]. Canny operatörü ile filtreleme işlemi Sobel filtreleme yapısına benzemektedir. Sadece filtre matrisinin boyutu ve değerleri farklıdır. Canny kenar algılama tekniği için **Şekil 4.39**'da bulunan yapı tek satır bilgisini hafızada tutacak şekilde değiştirilip uygulanmıştır.



Şekil 4.49. Canny kenar algılama tekniği ile görüntüde bulunan kenarların elde edilmesi

Şekil 4.49'da gerçekleştirilen uygulama ile gerçek zamanlı video görüntüsüne Canny kenar algılama tekniğinin uygulanmasına ait tasarım gösterilmektedir. Kenar algılama işlemi için ilk önce eşik değer uygulaması gerçekleştirilmiştir.



Şekil 4.50. Canny kenar algılama tekniğinin filtreleme işleminden sonra kenarların tespit edilmesi

Şekil 4.50'de kenar algılama işlemi gerçekleştirilmeden önce görüntüye 3x3'lük gauss filtre işlemleri uygulanmıştır. Canny kenar algılama tekniğini diğer kenar algılama teknikleri ile kıyasladığımızda, en çok detay bilgisi Canny operatöründe elde edilmektedir. Sobel operatörü iki filtreye göre ortalama bir kenar algılama işlemi gerçekleştirmektedir. 3x3'lük (sigma =0.85) Gauss filtrenin uygulanması ile bazı detayların filtrelenmesi amaçlanmıştır. Denklem 4.7'de kullanılan gauss filtre matrisi gösterilmektedir.

$$G(x,y,\sigma)_{2D} = \begin{bmatrix} 0{,}0625 & 0{,}125 & 0.0625 \\ 0{,}125 & 0{,}25 & 0{,}125 \\ 0{,}0625 & 0{,}125 & 0{,}0625 \end{bmatrix} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad (4.7)$$

Tablo 4.8. Canny kenar algılama tekniği FPGA kaynak kullanımı miktarı

Kaynaklar	Filtre uygulanmamış	Filtre uygulanmış	Mevcut
LUTs	522	736	53200
Slice Register	354	485	106400
BRAM	45.5	46,5	140
IO	33	41	200

Tablo 4.8'de Canny tekniği kullanılarak, pürüzsüzleştirme işlemi için Gauss filtrenin uygulanması ve uygulanmadan elde edilen kenar tespiti durumları için gerçekleştirilen tasarımlara ait FPGA kaynak kullanım miktarları verilmektedir. Filtre tekniği uygulandıktan sonra görüntü üzerinde küçük detaylara (gürültüler vb.) ait kenarlar elimine edilmiştir. Ancak kullanılan FPGA kaynak miktarı artmıştır.

Tablo 4.9. Kenar algılama tekniği FPGA kaynak kullanımı miktarlarının karşılaştırılması

	Canny	Sobel	Frekans (MHz)
Spartan3E-Slices [135]	2613	1054	120
Spartan6-Slices [135]	2418	1391	201
Virtex5-Slices [135]	2409	1389	292
Zynq7000-Slices	736	544	100

Tablo 4.9'da Canny kenar algılama tekniğini, tez kapsamında gerçekleştirilen çalışmanın farklı FPGA platformlarında gerçekleştirilen çalışma ile karşılaştırılması gösterilmiştir. Tezde gerçekleştirilen kenar algılama uygulamaları en az FPGA kaynak kullanımı ile gerçekleştirilmiştir.

Tablo 4.10. Kenar algılama tekniği FPGA kaynak kullanımları

Kaynaklar	Canny [136]	Sobel [136]	Canny [137]	Canny	Sobel
LUTs	5131	2335	7894	736	544
Slice Register	3647	1643	-	485	366
BRAM	155	125	5	46,5	45.5
IO	-	-	-	41	35

Tablo 4.10'da Canny kenar algılama tekniği kullanılarak gerçekleştirilen çalışmalarda kullanılan FPGA kaynak kullanımları karşılaştırılmıştır. Literatürde bulunan çalışmada[129], 5x5 filtre matrisi ile gerçekleştirilmiştir. Aritmetik işlemlerin LUT yapıları üzerinden gerçekleştirilmesinden dolayı, kaynak kullanımının arttığı düşünülmektedir. Tez kapsamında gerçekleştirilen çalışma ve [130]'te 3x3'lük gauss filtresinin ardından Canny kenar algılama tespiti uygulanmıştır. Literatürde bulunan çalışmalara ile gerçekleştirilen çalışmaların karşılaştırılmasında, tez kapsamında gerçekleştirilen kenar algılama çalışmalarda FPGA kaynak kullanım miktarları oldukça az sayıdadır.

Canny kenar algılama tekniği uygulandıktan sonra, görüntüye eşik değer algoritması uygulanmıştır, daha sonra morfolojik işlemler gerçekleştirildikten sonra Analog discovery 2 ile veriler bilgisayar ortamına taşınmıştır. Sayı sistemlerinin karşılaştırılması için MATLAB platformu kullanılmıştır. Piksel eşleştirme yöntemi kullanılarak sayı sistemlerinin performansları karşılaştırılmıştır.

Tablo 4.11. Sayı sistemlerinin performanslarının karşılaştırılması

	Performans (%)
Sabit noktalı sayı sistemi	67,22
Kayan noktalı sayı sistemi	72,79

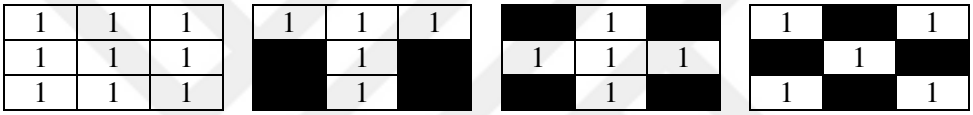
Tablo 4.11'de FPGA platformunda kayan ve sabit noktalı sayı sistemleri ile gerçekleştirilen aynı uygulamanın performansları karşılaştırılmıştır. İki farklı sayı sistemi arasındaki piksel değerlerinin eşleşme sonuçlarını karşılaştırmak için gerçekleştirilmiştir. Kayan noktalı sayı sistemlerinde daha hassas sonuçlar elde edilebilmektedir.

4.3.7. Morfolojik İşlemlerin Video Görüntüye Uygulanması

Görüntü işlemede morfoloji, resim üzerinde bulunan nesnelerin şekil ve yapı olarak incelenerek görüntüde sınırların, iskelet yapısının tanımlanması veya çıkartılması, gürültü giderme ve bölümleme gibi uygulamaların gerçekleştirilmesini sağlayan bir tekniktir. Genel olarak görüntünün temizlenmesi ve iyileştirilmesi için aşağıdaki morfolojik filtreler kullanılmaktadır.

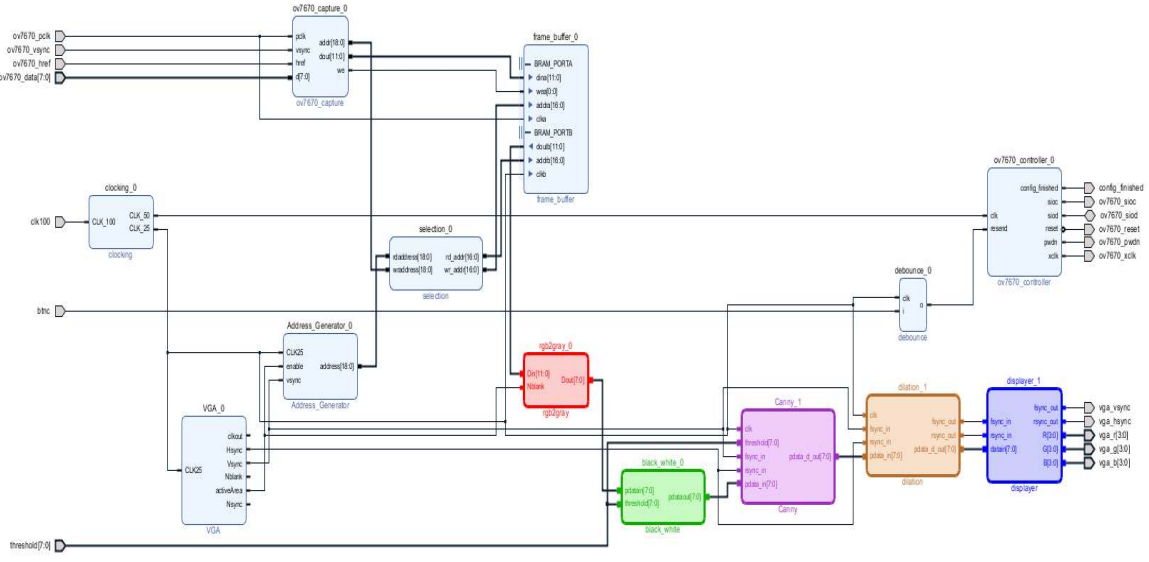
- Aşındırma (Erosion),
- Yayma (Dilation),
- Açma (Opening),
- Kapama (Closing)

Morfolojik işlemler ikili (binary) renk tonu ve gri tonlamalı görüntüler için kullanılabilmektedir. Bu tekniğin uygulanması için görüntüde yapı değişimini meydana getiren yapısal elemanlar kullanılmaktadır.



Şekil 4.51. Yapısal elemanlar

Şekil 4.51'de genel olarak kullanılan yapısal elemanlar için örnek filtreler verilmiştir. Gerçekleştirilecek uygulamaya göre yapısal elemanın boyut ve katsayıları tasarımcıya bağlı olarak değiştirilebilmektedir. Bu tez kapsamında, IP çekirdekleri ile gerçekleştirilen uygulamada 3x3'lük tüm değerleri "1" olan yapısal eleman matrisi kullanılmıştır.



Şekil 4.52. Morfolojik işlemlerin gerçek zamanlı video görüntüsüne uygulanması

Şekil 4.52'de gerçek zamanlı video görüntüsüne morfolojik işlemlerin IP çekirdekleri kullanılarak gerçekleştirilen tasarım gösterilmektedir. Kameradan elde edilen görüntü RGB-Gri tonlamalı dönüşümü gerçekleştirilir. Eşik değer uygulandıktan sonra Canny kenar belirleme tekniği kullanılarak görüntü üzerinde kenarların tespiti gerçekleştirilir. Bu aşamadan sonra görüntüye morfolojik işlemler uygulanarak, görüntü üzerindeki gürültüler filtrelenmiştir.

Tablo 4.12. Morfolojik işlemlerin gerçekleştirilmesinde kullanılan FPGA kaynak miktarları

Kaynaklar	Kullanılan	Mevcut
LUTs	674	53200
Slice Register	467	106400
BRAM	46.5	140
IO	41	200

Tablo 4.12'de morfolojik işlemler için, IP çekirdekleri kullanılarak gerçekleştirilen tasarımda kullanılan FPGA kaynak kullanımı verilmektedir.

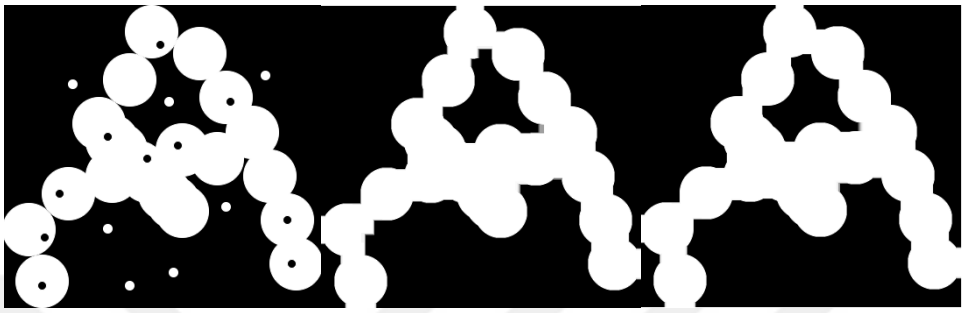
4.4. Bölüm değerlendirmesi

Tez çalışmasının bu bölümünde günümüzde kullanılan görüntü işleme tekniklerine ait yapılar açıklanmıştır. Görüntü işleme uygulamaların da kullanılan Raspberry Pi 3+b modeli, açık kaynak kütüphane fonksiyonları gerçekleştirilen uygulamalarda, işlem süresi görüntünün boyutuna göre çok fazla olacaktır ve ısınma probleminden dolayı işlemler gerçekleştirilemeyecek veya hata mesajları ile program sonlandırılacaktır. Bu yönler ile kıyaslandığında kullanımı önerilmemektedir. Transfer öğrenme ile yaya ve plaka bölgesi tespiti, Raspberry Pi ve USB kamera kullanılarak gerçek zamanlı olarak gerçekleştirilmiştir. Ancak, saniyede 0.8-1.5 görüntü elde edilebilmektedir. Raspberry Pi kartında ek olarak Coral USB benzeri yapılar kullanılarak fazladan TPU (Tensor Processing Unit) eklenerek saniye başına elde edilen görüntü sayısı (yaklaşık 30 FPS) arttırılabilmektedir. Nesne tespiti veya tanıma işlemleri gerçekleştirilirken, doğruluk oranını arttırmak ve daha iyi sonuçlar elde edebilmek için daha fazla etiketlenmiş görüntü ile model eğitimi gerçekleştirilmesi önerilmektedir. Ayrıca, transfer öğrenme veya model uyarlama (fine tuning) işlemleri kullanılarak gerçekleştirilecek çalışmalarda iyi sonuçlar elde edilebilmektedir.

FPGA platformunda, XSG/MATLAB-benzetim blokları ile gerçekleştirilecek tasarımlar, benzetim blokları ile sınırlı olduğu için, çok karmaşık yapılara sahip olmayan gerçek zamanlı temel görüntü işleme uygulamaları tasarlanabilmektedir. Aynı zamanda, XSG ile gerçekleştirilen uygulamaların tasarım aşamalarında MATLAB programına ihtiyaç duymaktadır. FPGA uygulamaları için IP blok tasarımları ile uygulamaların gerçekleştirilmesi önerilmektedir. Çünkü

IP bloklar ile gerçekleştirilen uygulamalarda daha az FPGA kaynak kullanımı ile gerçekleştirilebilmektedir.

Görüntünün elde edilmesi ve görselleştirilmesi işlemlerinden sonra, görüntü filtreleme, nesnelere ait sınırların belirlenmesi, görüntüde istenilmeyen kısımların çıkartılması ve görüntü üzerinde uygulamanın geliştirileceği kısımlara odaklanabilmek için bir dizi görüntü işleme tekniği uygulanmıştır.

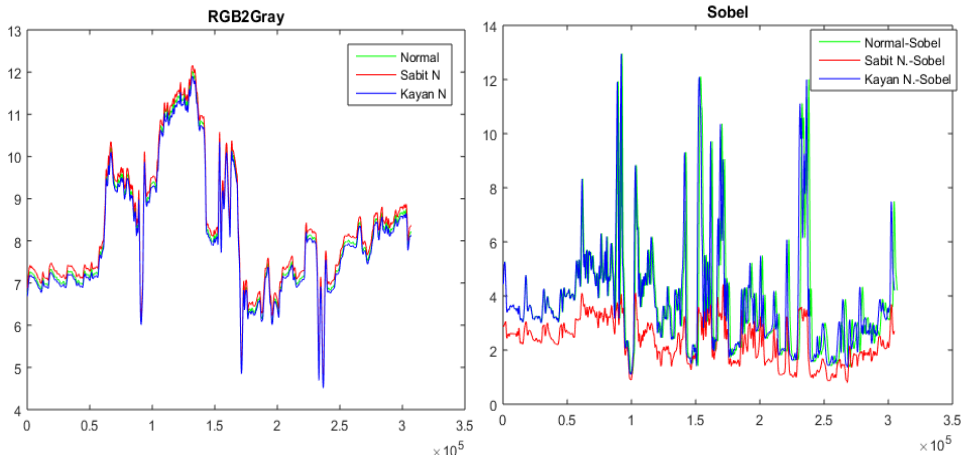


Orijinal görüntü MAK MKA

Şekil 4.53. Morfolojik işlemlerin karşılaştırılması

Morfolojik işlemler ile genel olarak görüntü üzerinde arka planda veya nesnelerin üzerinde bulunan gürültülerin giderilmesi için kullanılmaktadır. Tez kapsamında, morfolojik işlemlere farklı bir bakış açısı getirilerek, hem arka planda hem de nesnelerin üzerinde bulunan gürültüler giderilmiştir. **Şekil 4.53**'te orijinal görüntüye açma ve kapama işlemleri uygulanmıştır. MAK (Morfolojik Açma Kapama) ve MKA (Morfolojik Kapama Açma) işlemleri ile görüntüde bulunan istenilmeyen gürültüler filtrelenmiştir. Görüntüde nesneye ait sınırların en fazla korunduğu MKA tekniğinin kullanımı önerilmektedir.

FPGA platformunda sabit ve kayan noktalı sayı aritmetiği kullanılarak gerçekleştirilecek uygulamalarda, FPGA platformunda kullanılan kaynak miktarları ve güç tüketimleri karşılaştırılmıştır.



Şekil 4.54. Sabit ve kayan noktalı sayı sistemlerinin piksel yoğunluğu hassasiyetliliğinin karşılaştırılması

Şekil 4.54'te RGB-Gri tonlamalı görüntü dönüşümü ve Sobel kenar algılama tekniğinin FPGA platformunda sabit ve kayan noktalı sayı sistemlerinin sayı hassasiyetliklerinin karşılaştırılması için gerçekleştirilen uygulamalara ait sonuçlar gösterilmektedir. X-ekseni 640x480 piksel ve y-ekseni bu piksellere ait yoğunlukları temsil etmektedir. Gelişmiş uygulamalarda, kayan noktalı sayı aritmetiğinin sayı hassasiyetinin sabit noktalı sayılara göre daha uygun sonuçlarda elde edildiği görülmektedir. FPGA platformunda kayan ve sabit noktalı sayı sistemleri ile gerçekleştirilen Canny kenar algılama uygulaması için piksel eşleşme yöntemi kullanılarak satı sistemlerine göre karşılaştırılması gerçekleştirilmiştir. Kayan noktalı sayı sistemlerinde daha yüksek performansta sonuçlar elde edilebilmektedir.

Tablo 4.13.FPGA Güç Tüketimi Miktarının Karşılaştırılması

Kaynaklar	Sabit noktalı (Watt)	Kayan noktalı (Watt)
Signals	2,580	2,960
Logic	0,776	1,713
BRAM	0,907	0,907
DSP	0,802	1,165

Tablo 4.13'te RGB-Gri tonlamalı görüntü dönüşümünün FPGA platformunda, kayan ve sabit noktalı sayı sistemleri ile gerçekleştirilmesi ile elde edilen güç tüketimi miktarları karşılaştırılmaktadır. Aynı bit uzunluğunda (16 bit) ve 640x480 boyutlarında video görüntünün gri tonlamalı görüntüye dönüştürülmesinde, kayan noktalı sayı sistemlerinde işlem yükünün sabit noktalı sayı sistemlerine göre daha fazla olmasından dolayı sabit noktalı sayı sisteminin kullanılması ile daha az güç tüketimi ile uygulama gerçekleştirilmiştir.

Tablo 4.14.FPGA Kaynak Kullanımı Karşılaştırma Tablosu

Kaynaklar	Sabit noktalı	Kayan noktalı
LUTs	319	433
Slice Register	218	218
BRAM	45.5	44,5
DSP	3	3

Tablo 4.14'te RGB-Gri tonlamalı dönüşümü uygulamasının, kayan ve sabit noktalı sayı sistemleri ile gerçekleştirilmesi ile kullanılan FPGA kaynak miktarları karşılaştırılmıştır. Aynı bit

uzunluğunda (16 bit) ve 640x480 boyutlarında gri tonlamalı görüntüye dönüşümde sabit ve kayan noktalı sayı sistemlerinde aynı miktarda BRAM kullanılmıştır. Eşit miktarda DSP kullanılmış fakat DSP kullanımında güç tüketimi sabit noktalı sayı sistemlerinde daha az miktardadır.

Tablo 4.15.FPGA platformunda HDL ve IP Bloklar ile gerçekleştirilen uygulamaların kıyaslanması

Kaynaklar	VHDL kod ile RBG Video	IP Blok ile RBG Video	VHDL kod ile gri tonlamalı Video	IP Blok ile gri tonlamalı Video
LUTs	350	295	307	304
Slice Register	230	225	225	225
BRAM	44	44	44	44
IO	35	35	35	35

Tablo 4.15'te FPGA platformunda HDL ve IP çekirdek tabanlı gerçekleştirilen uygulamaların kaynak kullanımı karşılaştırılmaktadır. Aynı kodlar ile renkli görüntünün elde edilmesi ve gri tonlamaya dönüştürmesi ile gerçekleştirilen uygulamalarda, IP çekirdeklerinin kullanımı daha az miktarlarda FPGA kaynak kullanımı gerçekleştirilmiştir.

Görüntü üzerinde kenar algılama tekniğinde, yatayda ve dikeyde uygulanan iki adet filtre matrisinin görüntüye uygulanması sonucunda gerçekleştirilmektedir. Fakat filtreleme işleminde kenarların tespit edilmesinde belirleyici rol filtre katsayılarına aittir.

Tablo 4.16. Kenar algılama tekniklerinde kullanılan FPGA kaynak kullanımı

Kaynaklar	Sobel	Canny	Canny+ pürüzsüzleştirme	[131]	[132]	[133]
LUTs	544	522	736	5677	7301	2335
Slice Register	366	354	485	7919	12311	3647
BRAM	45.5	45.5	46,5	7	12	125
IO	35	33	41	52	32	32

Tablo 4.16'da kenar algılama tekniklerinde FPGA kaynak kullanımı ve literatürde gerçekleştirilen uygulamalar ile karşılaştırılması verilmektedir. Tez kapsamında Sobel ve Canny algoritmaları kullanılarak kenar algılama işleminin gerçekleştirilmesinde Canny kenar algılamada FPGA kaynak kullanım miktarı daha azdır. Ancak, pürüzsüzleştirme algoritması ile birlikte kullanılacak Canny tekniğinde, Sobel filtreye kıyasla daha fazla FPGA kaynak kullanılmaktadır. Gerçekleştirilecek tasarıma göre, detaylı kenar bilgisinin istenildiği uygulamalarda Canny algoritmasının kullanımı önerilmektedir.

Tez kapsamında gerçekleştirilen Sobel kenar algılama tekniği ve literatürde bulunan aynı FPGA platformu ile gerçekleştirilen çalışmaların karşılaştırılması **Tablo 4.16**'da verilmektedir. IP çekirdeklerinin kullanımı, kullanılan kaynak miktarını düşüreceği için kullanımı önerilmektedir.

5. ÖNERİLEN MVSR NORMALİZASYON TEKNİĞİ İLE GÖRÜNTÜ İŞLEME UYGULAMALARI

Normalizasyon, görüntü işleme sürecinde yaygın olarak kullanılan tekniklerden biridir. Bu teknikler genellikle görüntüyü iyileştirmek, görüntüde üzerinde bulunan gürültüleri azaltmak ve renk dağılımını değiştirip hafızada daha az yer kaplamasını sağlamak gibi uygulamalarda görüntü ön işleme basamağı olarak kullanılmaktadır.

5.1. MVSR Normalizasyon Tekniği

Genel olarak kullanılan normalizasyon tekniklerinde, görüntünün normalize edilmiş piksel değerleri, pozitif veya negatif piksel değerlerine sahip olabilmektedirler ve negatif piksel değerleri bazı durumlarda (uint8 dönüşümü vb.) elimine edilmektedir. Normalizasyon işleminin sonucunda negatif piksellerin etkisinin ortadan kaldırılmaması için MVSR normalizasyon tekniği önerilmektedir.

Ortalama değer ve varyans ile normalleştirme, sinyal işleme, görüntü ve konuşma işleme, yapay zeka, büyük veri, optimizasyon algoritmaları vb. Birçok uygulamada kullanılan bir gürültü / hata telafi yöntemidir. Bu uygulamamızda, MVSR (Ortalama-Varyans-Softmax-Rescale) olarak adlandırılan yeni bir normalleştirme algoritması önerilmiştir. Bu teknikte ortalama değer, varyans, Softmax aktivasyon fonksiyonu ve yeniden ölçeklendirme süreçleri kullanılmaktadır.

$$\mu_{MVSR} = \frac{1}{M}\sum_{i=1}^{M} x_i \tag{5.1}$$

$$\sigma^2_{MVSR} = \frac{1}{M}\sum_{i=1}^{M}\left(x_i - \mu_{MVSR}\right)^2 \tag{5.2}$$

$$x_i = \frac{x_i - \mu_{MVSR}}{\sqrt{\sigma^2_{MVSR}} + \delta} \tag{5.3}$$

$$S\left(x\right)_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{5.4}$$

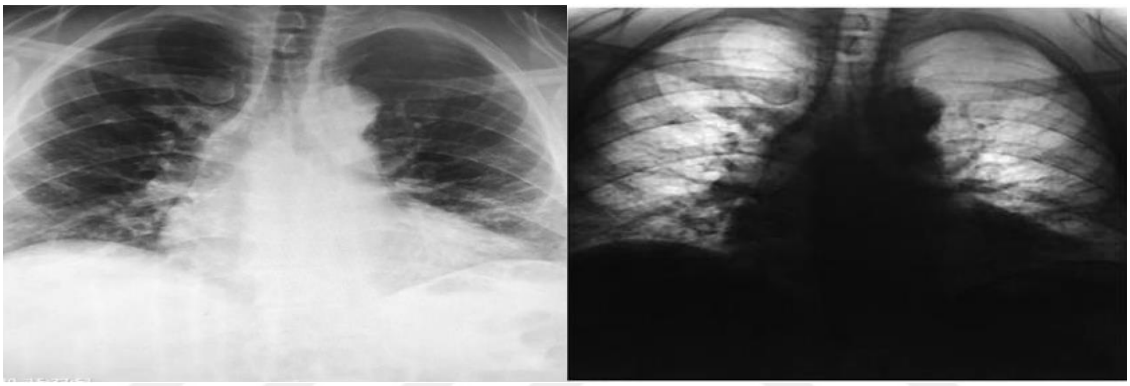
$$r_i = \frac{S\left(x\right)_i}{\max(S\left(x\right)_i)} * 255 \tag{5.5}$$

MVSR Normalizasyon süreçleri denklem 5.1-.5.5'de gösterilmektedir. Görüntülere tersleme işlemi uygulandıktan sonra, denklem 5.1'de piksellere ait ortalama değer, denklem 5.2'de varyans hesabı, denklem 5.3'te ortalama ve varyans değerleri kullanılarak yeni piksel değerleri elde edilmektedir. Elde edilen yeni piksellerde negatif değerli piksellerin elimine edilmemesi için,

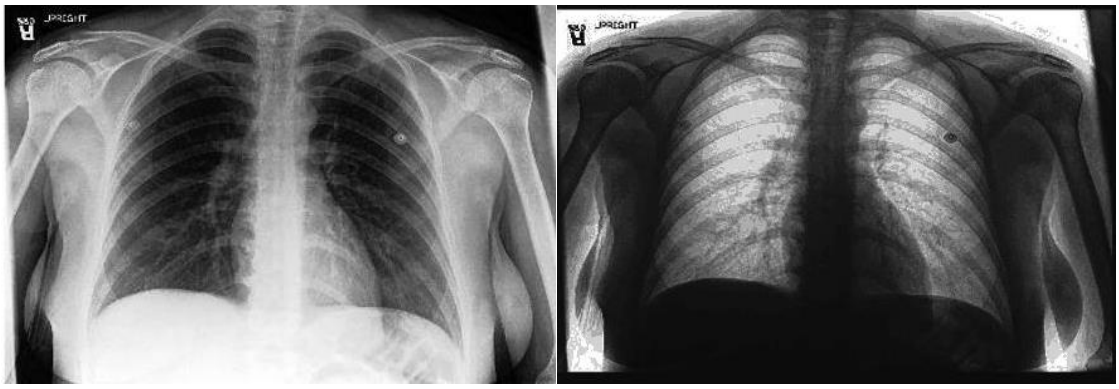
denklem 5.4'te verilen softmax aktivasyon fonksiyonu kullanılarak tüm piksel değerlerinin birbirileri ile ilişkilendirilmesi amaçlanmıştır. Softmax uygulandıktan sonra elde edilen değerler 0-1 sayı aralığından 0-255 aralığına, denklem 5.5 kullanılarak yeni görüntü için piksel değerleri elde edilmiştir.

5.2. MVSR Normalizasyon Tekniğinin X-ışını Görüntülerine Uygulanması

MVSR normalizasyon tekniğinin görüntü işleme üzerindeki etkilerinin incelenmesi amacıyla, bu teknik Akciğer X-ışını görüntülerine uygulanmıştır. Bu teknik kullanılarak, Covid-19 hastalığının teşhisini ve ön değerlendirmesini kolaylaştırmak için uygulanmıştır.

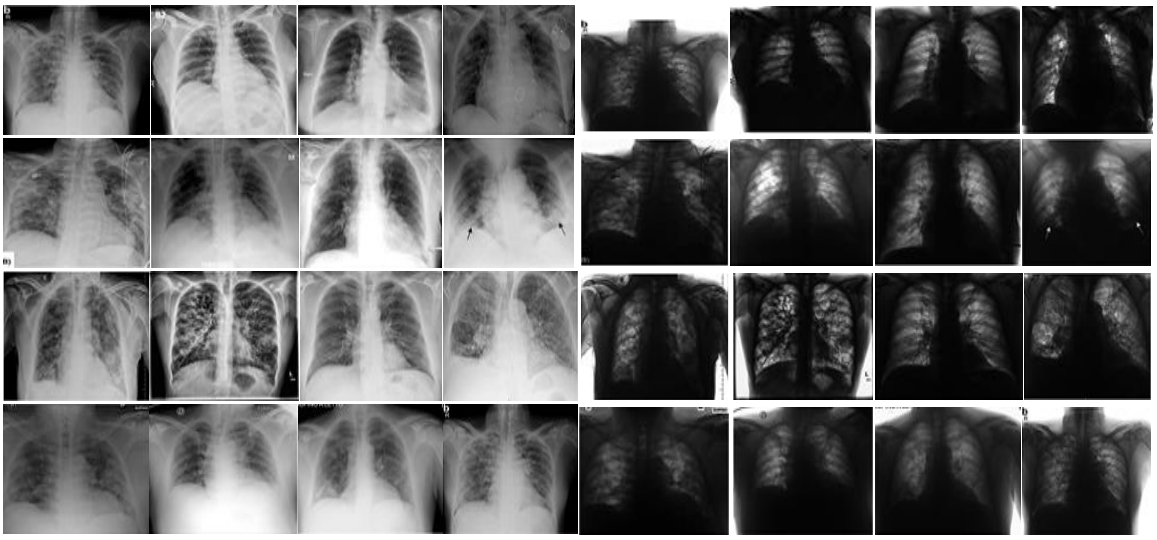


Şekil 5.1. Covid-19 virüslü akciğer röntgeni orijinal ve MVSR uygulanmış görüntüsü

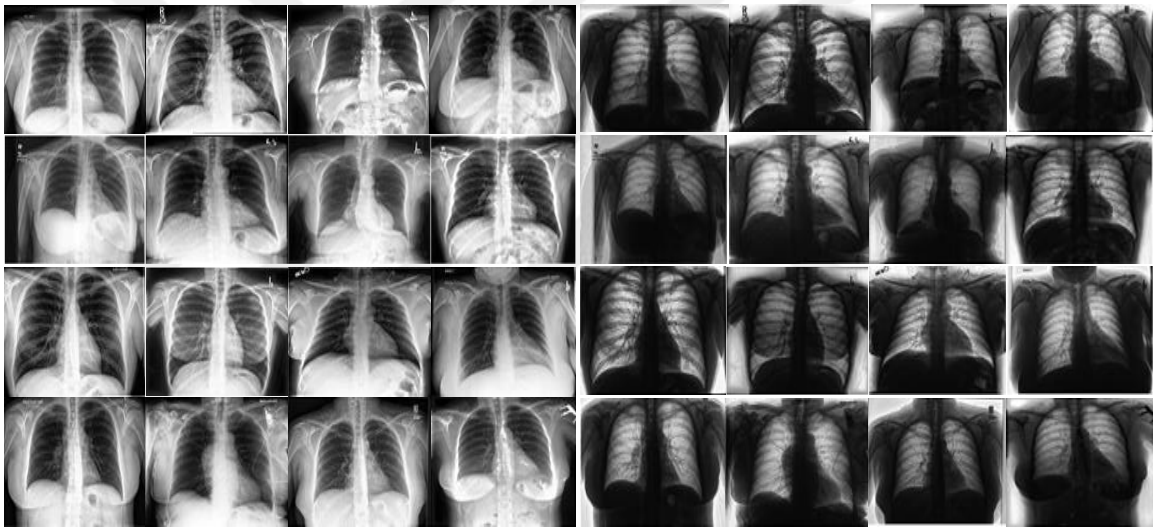


Şekil 5.2. Virüs tespit edilmeyen akciğer röntgeni orijinal ve MVSR uygulanmış görüntüsü

MATLAB platformu kullanılarak MVSR normalizasyon tekniğinin akciğer X-ışını görüntülerine uygulanması ile elde edilen sonuçlar **Şekil 5.1** ve **Şekil 5.2**'de gösterilmektedir. Elde edilen sonuçlar uzman görüşüne (Enfeksiyon Hastalıkları Doktoru) göre "MVSR normalizasyonun uygulandığı akciğer röntgen sonuçları görüntülerinde enfeksiyon daha net görülebilmektedir" yorumu yapılmıştır.



Şekil 5.3. Covid-19 virüslü farklı hastalara ait akciğer röntgeni orijinal ve MVSR uygulanmış görüntüler



Şekil 5.4. Virüs tespit edilmeyen farklı hastalara ait akciğer röntgeni orijinal ve MVSR uygulanmış görüntüler

MVSR Normalizasyon tekniğinin etkilerini görüntüler üzerinde görebilmek için Covid-19 teşhis edilen ve bu virüse sahip olmayan hastalardan çekilmiş akciğer x-ışını görüntüleri kullanılmıştır. Bu uygulama sonuçları, **Şekil 5.3** ve **Şekil 5.4**'te gösterilmektedir. Bu teknik kullanılarak elde edilen sonuçların daha iyi olduğunu göstermek için evrişimli sinir ağı kullanılarak doğruluğu kanıtlanmıştır. Orijinal ve MVSR normalizasyonu uygulanmış görüntüler aynı model ile eğitildikten sonra, orijinal görüntülerin sinir ağı model doğruluk sonucu %83.01 iken normalizasyon uygulanmış görüntülerin doğruluk oranı %96.16 olarak tespit edilmiştir.

MVSR normalizasyonu etkilerini MATLAB programı ile tespit edilmesinin ardından FPGA platformunda uygulanabilmesi için normalizasyonun tüm aşamaları sabit noktalı sayı formatında

tasarlanıp, VHDL dili kullanılarak gerçekleştirilmiştir. MVSR Normalizasyon tekniğinde kullanılan bölme aritmetiğini **Tablo 5.1**'de ve karekök alma işlemi **Tablo 5.2**'de gösterilmektedir. Bu işlemler sabit noktalı sayı formatında gerçekleştirildiği için kesirli sayıların bölme ve karekök alma işlemleri için ondalık sayı sistemine göre, ondalıklı sayı kısmında virgülden sonra iki basamak hassasiyet ile işlemler gerçekleştirilmiştir. Bölme işleminde "LUT_operation" ile adlandırılan kısımda virgülden sonraki kısım için elde edilen sayının sabit noktalı gösterimdeki karşılığı bulunmaktadır.

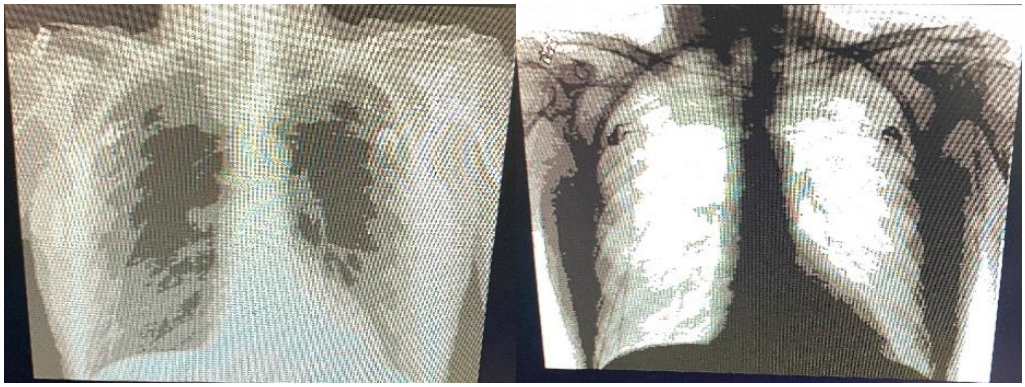
Tablo 5.1. VHDL sabit noktalı sayı sistemi bölme işlemi temsili kod gösterimi

```
If Dividend < 0
   Dividend = Not Dividend +1
else if Divisor < 0
   Divisor = Not Divisor +1
else if Divisor or Dividend = 0
   Quotient=0
   Remainder=0
else
   Remainder = Dividend rem Divisor
   Quotient= Dividend / Divisor
   nRemainder=100* Remainder/ Divisor
   mRemainder = LUT_operation (nRemainder)
End
Output= Quotient& mRemainder
```

MVSR normalizasyonu karekök alma işleminde, sabit noktalı sayılar soldan sağa doğru 2,3,4,5…n bit değerinde gruplara ayrılmıştır. İlk grup iki bitlik sayı "01" ile karşılaştırıldıktan sonra sonuç '0' ya da '1' değerini alacaktır. Bu değer köklü sayının ilk çıkışı olacaktır ve aynı zamanda karşılaştırma yapılan değere eklenerek döngü bitene kadar bu şekilde tekrarlanacaktır. Bu detaylara ait VHDL kod **Tablo 5.2**'de verilmiştir.

Tablo 5.2. Sabit noktalı sayı sisteminde karekök alma işlemi VHDL kodu

```
datau:=unsigned(datain);
expectdata (1 downto 0):="01";
for i in 0 to length loop
   if (datau(size downto size-1-2*i)< expectdata (2*i+1
downto 0)) then
     dtu(length-i):='0';
   else
     dtu(length-i):='1';
   end if;
     expectdata (2*i+3 downto 0) :=(dtu(length downto
length-i)&'1')*(dtu(length downto length-i)&'1');
      dt:=std_logic_vector(dtu);
 end loop;
 return dt;
```



Şekil 5.5. Covid-19 virüs tespit edilmeyen akciğer röntgeni orijinal ve MVSR uygulanmış görüntüsü FPGA sonucu

Şekil 5.5'te FPGA kullanılarak, Covid-19 virüsüne sahip olmayan hastalardan alınan akciğer röntgen filminin orijinali ve MVSR normalizasyonu uygulanmış halinin sonucu gösterilmektedir. Orijinal görüntünün piksel renk paletleri görüntüden çıkartılarak, piksellere numaralar atanmıştır. Aynı piksel renk paletlerine sahip piksellere aynı değerler verilmiştir. Bu teknik kullanılarak görüntünün hafızada kapladığı alan minimum seviyelere indirilmiştir ve işlem hızını da arttırmıştır. Bu teknik histogram tekniğine benzemektedir. Histogram tekniğinde her bir pikselden kaç adet olduğu belirlenirken, bu teknikte aynı piksel değerleri aynı isim ile adlandırılmaktadır. Her bir pikselin üç boyutlu renk değeri, renk paletine aktarılarak aynı renk tonları aynı sayı ile piksel haritasında isimlendirilmiştir. Böylelikle üç boyutlu renkli görüntü yine üç boyut bilgisi ile bir boyutta temsil edilmektedir. Bu uygulama sayesinde gömülü sistemde hafıza kullanımı oldukça düşürülmüştür.

Şekil 5.6. Görüntü matrisi, piksel haritası ve renk paleti

Şekil 5.6'da görüntü matrisi üzerinde piksel haritası ve renk paletinin çıkartılması temsili olarak gösterilmektedir. MVSR normalizasyonu, Xilinx Zynq-7000 AP SoC XC7Z020-CLG484 FPGA geliştirme kartı kullanılarak gerçekleştirilmiştir. Bu tasarıma ait FPGA kaynak kullanımı **Tablo 5.3**'te Gösterilmektedir.

Tablo 5.3. MVSR Normalizasyonu FPGA kaynak kullanımı

Kaynaklar	Kullanılan	Mevcut	% Olarak kullanılan
LUTs	193	53200	0.36
Slice Register	136	106400	0.13
BRAM	30	140	21.43
IO	18	200	9

5.3. MVSR Normalizasyon Tekniğinin CNN Model Yapısında Görüntü Ön İşleme Uygulanması

MVSR normalizasyon tekniği kullanılarak elde edilen sonuçların doğruluğunu ispatlamak için evrişimsel sinir ağı (CNN) modeli kullanıldı. Bu model, orijinal ve MVSR uygulanmış akciğer resimlerinin Covid-19 virüsünün tespitinde kullanılmıştır. Model giriş resimlerinin boyutları (W, H) ve her bir katmanda kullanılan filtrelerin boyutları (S) ve sayıları (N) ile temsil edilmektedir.

Tablo 5.4. MVSR normalizasyonu için tasarlanan CNN modeli

Katmanlar	Katman Konfigürasyonları			
	S	N	W	H
Giriş	-	-	150	150
Conv1	3x3	32	150	150
Conv2	3x3	32	150	150
Conv3	3x3	64	150	150
Max-Ortak.	2x2	-	75	75
Conv4	3x3	128	75	75
Max-Ortak.	2x2	-	37	37
Conv5	3x3	64	37	37
Max-Ortak.	2x2	-	18	18
Conv6	3x3	128	18	18
Max-Ortak.	2x2	-	9	9
Conv7	3x3	128	9	9
Max-Ortak.	2x2	-	4	4
Tam bağl.1		2048		
Tam bağl.2		512		
Çıkış		1		

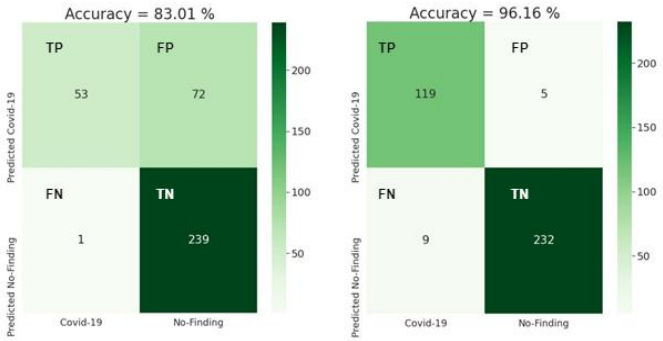
Tablo 5.4'te her iki veri seti için kullanılan CNN yapısının her bir katmanında gerçekleştirilen işlemler ve bu işlemlere ait kullanılan parametrelerin miktarları verilmiştir. Giriş 150x150 akciğer x-ışını görüntüsü ve Covid-19 veya virüse sahip olmadığını gösteren tek bir çıkış

bulunmaktadır. Her iki model için, 0.01 ve her 300 döngüde 0.95 olarak azalan öğrenme katsayı ve Adam optimizasyon tekniği ile 48 tur (Epoch) ikili çapraz entropi (binary cross entropy) hata fonksiyonu kullanılmıştır. Maksimum ortaklama tekniğinden önce Batch-Norm ve Tam bağlantı katmanlarından sonra 0.25 oranında seyreltme (Dropout) kullanılmıştır. Son katmanda sigmoid ve diğer katmanlarda ReLU aktivasyon fonksiyonları kullanılmıştır.

Model eğitim sürecinde özellik çıkartma katmanlarında, ezberleme (aşırı öğrenme: overfitting) olayını engellemek için Batch-Norm tekniği ve sınıflandırma katmanlarında ise seyreltme ile modelin hızının arttırılması planlanmıştır.

Genel olarak YSA modellerinin sınıflandırma sonuçlarını tespit etmek ve değerlendirmek için dört farklı tahmin yaklaşımı kullanılmaktadır. Tahmin edilen ve hesaplanan değerlerin dört farklı kombinasyonu vardır. Bunlar Gerçek Pozitifler (TP), Gerçek Negatifler (TN), Yanlış Pozitifler (FP) ve Yanlış Negatifler (FN) olarak adlandırılırlar.

- ✓ TP: Klinik testlerle Covid-19 virüsü teşhisi konan kişilerdir ve CNN modeli, onları Covid-19 enfekte olarak doğrular.
- ✓ TN: Klinik testlerle Covid-19 virüsü teşhisi konulmayan kişilerdir ve CNN modeli onları enfekte olmadıklarını doğrular.
- ✓ FP: Klinik testlerle Covid-19 hastalığı teşhisi konulmayan kişilerdir ancak CNN modeli onları Covid-19 ile enfekte olarak sınıflandırmıştır.
- ✓ FN: Klinik testlerle Covid-19 hastalığı teşhisi konan kişilerdir, ancak CNN modeli onları enfekte değil olarak sınıflandırmıştır.



Şekil 5.7. CNN Model sonuçlarının tahmin edilen ve hesaplanan değerlerinin karşılaştırılması

Şekil 5.7'de MVSR normalizasyonu uygulanmış ve uygulanmamış akciğer X-ışını görüntülerine ait CNN modelinin, tahmin edilen ve hesaplanan değerlerinin karşılaştırılması gösterilmektedir. Bu dört farklı tahmin yaklaşımı kullanılarak modelin doğruluğunu değerlendirecek bazı parametreler elde edilebilmektedir. Bu parametreler, Hassasiyet (Precision), Özgüllük (Recall) ve bu iki parametrenin harmonik ortalaması olan F1 skor değeridir.

Tablo 5.5. Covid-19 F1-Skor tablosu

	F1-Score Normal	**F1-Score** MVSR
Covid-19	0.51	0.92
Normal	0.85	0.96

Tablo 5.5'te Evrişimli sinir ağının Covid-19 virüsüne sahip olan bu virüsü taşımayan akciğer X-ışını görüntüleri ile gerçekleştirilen CNN sonuçları gösterilmektedir. Virüse sahip olan görüntülerin belirlenmesine ait f1-skor %51'den %92'ye ve virüs olmayan akciğer görüntülerinin tespiti ise %85'ten %96'ya çıkarılmıştır. MVSR normalizasyon sonuçları dolaylı bir şekilde virüslerin yayılmasına da engel olduğu söylenebilir. Çünkü eski modelde Covid-19 virüsüne sahip olduğu halde 72 hastaya ait sonuçlar negatif olarak sınıflandırılmıştır.

Tablo 5.6. Farklı normalizasyon teknikleri ile gerçekleştirilen Covid-19 virüsü sınıflandırma uygulamalarının karşılaştırılması

	[76]	[77]	[78]	[79]	**Bu çalışmada**
Kullanılan yöntem	AXG görüntülerinde diyafram bölgesi çıkartılmış, Histogram eşitleme ve Bilateral filtre uygulanmıştır	AXG görüntü işleme teknikleri uygulanmıştır	AXG görüntü piksel değerlerinin 0-1 sayı aralığında normalizasyonu	Histogram eşitleme, Negatif dönüşüm, Gama düzeltmesi, Kontrast geliştirme teknikleri uygulanmıştır	MVSR normalizasyon
Öğrenme modeli	VGG16 modeli, ImageNet verileri ile Transfer öğrenme	GoogleNet M-inception modeli, ImageNet verileri ile transfer öğrenme	AlexNet, VGG-16, GoogleNet, MobileNet-V2, SqueezeNet, ResNet-34, Inception-V3 modelleri ile transfer öğrenme	ResNet18, ResNet50, ResNet101, InceptionV3, DenseNet201 ve ChexNet	CNN
Normalizasyon öncesi sonuçlar	%88	%82,5		%93,22	%83.01
Normalizasyon sonrası sonuçlar	%94,5	%93	%96.35 (ortalama)	%96.29 (ortalama)	%96.16
Kullanılan veri miktarı	8474 görüntüde, 415 covid-19 virüslü, 5179 diğer virüsler ve 2880 normal	1065 görüntü	406 görüntü, 203 covid-19 virüslü, 203 normal	18479 görüntü, 3616 adet Covid-19 virüslü	565 görüntü 240 covid-19, 120 normal, 200 test (Covid ve Normal)

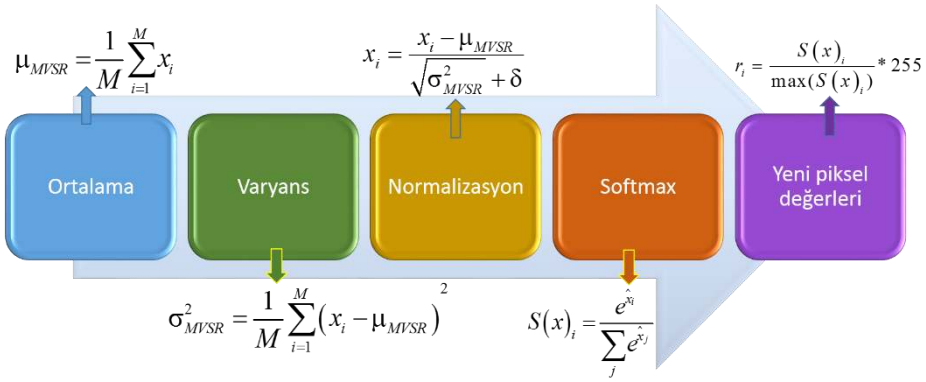
Tablo 5.6'da Tıbbi görüntü işleme ve makine öğrenimine dayalı gömülü sistem platformları ile hastalık tespit veya teşhis işlemlerinin geliştirilmesi, hastalığa özelliklerin otomatik

olarak analiz edilmesi ve uzmanlara (Doktor, radyolog vb.) daha doğru karar vermeleri için destek araçları sağlamayı amaçlayan çalışmalar bulunmaktadır. Bu çalışmalarda, model eğitim sürecinde aşırı (overfit) veya eksik (underfit) öğrenme sonuçlarından kaçınmak için önceden büyük veri setleri (ImageNet vb.) ile eğitilmiş modeller ile transfer öğrenme tekniği kullanılarak model eğitimlerini, Alexnet, VGGNet, ResNet gibi CNN yapıları kullanarak gerçekleştirmişlerdir.

Bu tez kapsamında gerçekleştirilen MVSR normalizasyon tekniği akciğer x-ışını görüntülerinde Covid-19 virüsünün teşhisini kolaylaştırmak için uygulanmıştır. Kullanılan normalizasyon tekniğinin performansını değerlendirmek için oluşturulan CNN modelinin sonuçlarının doğruluğunun arttırılmasını sağlayacak farklı teknikler (Transfer öğrenme vb.) kullanılmamıştır. Böylelikle sadece normalizasyonun sınıflandırmaya etkisi araştırılmak istenmiştir. Bu çalışma literatürde olan farklı normalizasyon teknikleri ile gerçekleştirilen diğer araştırmalar ile karşılaştırıldığında, MVSR normalizasyon tekniği ile gerçekleştirilen uygulamanın performansının iyi sonuçlar verdiği gözlemlenmektedir.

5.4. Bölüm değerlendirmesi

Önerilen MVSR normalizasyon tekniği, virüslü ve virüssüz akciğer x-ışını görüntülerine uygulanması ile hastalığın ön teşhisinin kolaylaştırılması gerçekleştirilmiştir. Uzman görüşüne göre "MVSR normalizasyonu, Covid-19 virüsü ile enfekte olan ve olmayan hastalara ait X-ışını görüntülerinin farkı ön değerlendirme aşamasında daha net görülebilmektedir.". önerilen normalizayon tekniğinin etkinliği, uzman görüşü, MATLAB, Anaconda Navigator ara yüzü ve FPGA gömülü sistemi içeren farklı değerlendirme yöntemleri kullanılarak kanıtlanmıştır.



Şekil 5.8. Önerilen MVSR normalizasyon tekniği

Şekil 5.8'de MVSR normalizasyon tekniğine ait işlem basamakları gösterilmektedir. Önerilen tekniğin FGPA platformunda uygulanabilmesi için akciğer x-ışını görüntülerine Histogram dağılımı tekniğine benzer bir teknik kullanılmıştır. Orijinal görüntünün, RGB üç boyutlu

renk piksel değerleri numaralandırılarak renk paleti oluşturulmuştur. Aynı piksel renk paletlerine sahip piksellere aynı değerler verilmiştir. Bu teknik kullanılarak görüntünün hafızada kapladığı alan minimum seviyelere indirilmiştir ve işlem hızını da arttırmıştır. Bu teknik histogram tekniğine benzemektedir. Histogram tekniğinde her bir pikselden kaç adet olduğu belirlenirken, bu teknikte aynı piksel değerleri aynı isim ile adlandırılmaktadır. Böylelikle üç boyutlu renkli görüntü yine üç boyut bilgisi ile bir boyutta temsil edilmektedir. Bu uygulama sayesinde gömülü sistemde hafıza kullanımı oldukça düşürülmüştür.

Evrişimli sinir ağı modeli oluşturularak, Covid-19 virüsüne sahip olan ve olmayan akciğer x-ışını görüntülerinin sınıflandırılması gerçekleştirilmiştir. Önerilen normalizasyon tekniğinin akciğer x-ışını görüntülerine uygulanması ile virüse sahip olan görüntülerin belirlenmesinde elde edilen f1-skor değerleri %51'den %92'ye ve virüs olmayan akciğer görüntülerinin tespiti ise %85'den %96'ya çıkarılmıştır. Model doğruluk oranı ise %83,01 iken, önerilen normalizasyon tekniği ile bu oran %96,16 elde edilmiştir.

Tablo 5.7. Farklı normalizasyon teknikleri kullanılarak gerçekleştirilen çalışmaların önerilen sistem ile karşılaştırılması

	[76]	[77]	[78]	[79]	**Bu çalışmada**
Normalizasyon öncesi sonuçlar	%88	%82,5	-	%93,22	%83.01
Normalizasyon sonrası sonuçlar	%94,5	%93	%96.35 (ortalama)	%96.29 (ortalama)	%96.16

Önerilen MVSR normalizasyon tekniğinin literatürde bulunan faklı normalizasyon teknikleri ile gerçekleştirilmiş uygulamaları ile karşılaştırılması **Tablo 5.7**'de gösterilmektedir. Önerilen normalizasyon tekniği ile akciğer x-ışını görüntülerinde Covid-19 virüsü tespitinin doğruluk oranı arttırılmıştır. Bu çalışmada kullanılan normalizasyon tekniğinin, literatürde bulunan tekniklere göre sınıflandırma doğruluk oranı daha yüksek miktarlardadır.

6. ARAÇ PLAKA GÖRÜNTÜSÜ ÜZERİNDE ÖZELLİK ÇIKARIMI

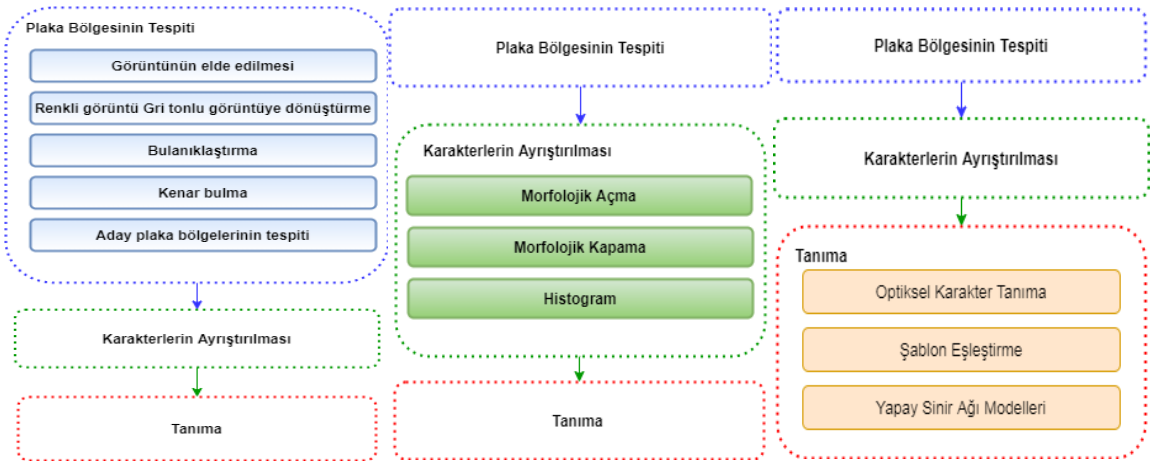
Görüntüde özellik çıkarımı uygulamalarından biri olan araç plaka tanıma sistemleri genel olarak, bir görüntüden veya bir grup görüntüden (video) araç plaka karakterlerinin çıkarılması için tasarlanan yazılım ve donanım tabanlı uygulamalardan biridir. Araç plaka bilgileri, otomatik elektronik ödeme sistemleri (otoyol geçiş ücreti veya park ücreti ödemeleri vb.) ve trafik akış kontrolü için otoyol ve arter izleme sistemleri gibi birçok uygulamada bir veri tabanıyla (Çalıntı araç tespiti, trafik ışığı ihlalleri vb.) veya veri tabanı olmadan kullanılmaktadır. Bu yüzden araç plaka okuma sistemlerinin verimli olarak uygulanabilmesi önemli yer tutmaktadır. Araç plaka karakterlerinin belirlenmesi genel olarak üç aşamada gerçekleştirilmektedir. Bu aşamalar, Plaka bölgesinin bulunduğu yerin tespiti, Plaka bölgesindeki karakterlerin ayrıştırılması ve bu karakterlerin tanınmasıdır.

6.1. Standart Görüntü İşleme Teknikleri ile Araç Plaka Tespiti ve Karakterlerin Ayrıştırılması

Araç plaka tanıma uygulamalarının dezavantajları ve plaka tanıma için kullanılması gereken sistemler (donanım ve yazılım) göz önüne alındığında oldukça maliyetli sistemlerdir. Kamera aracılığı ile elde edilen görüntülerde, plaka ve kamera arasındaki mesafe azaldıkça plaka tanıma sistemlerinin maliyeti de azalmaktadır.

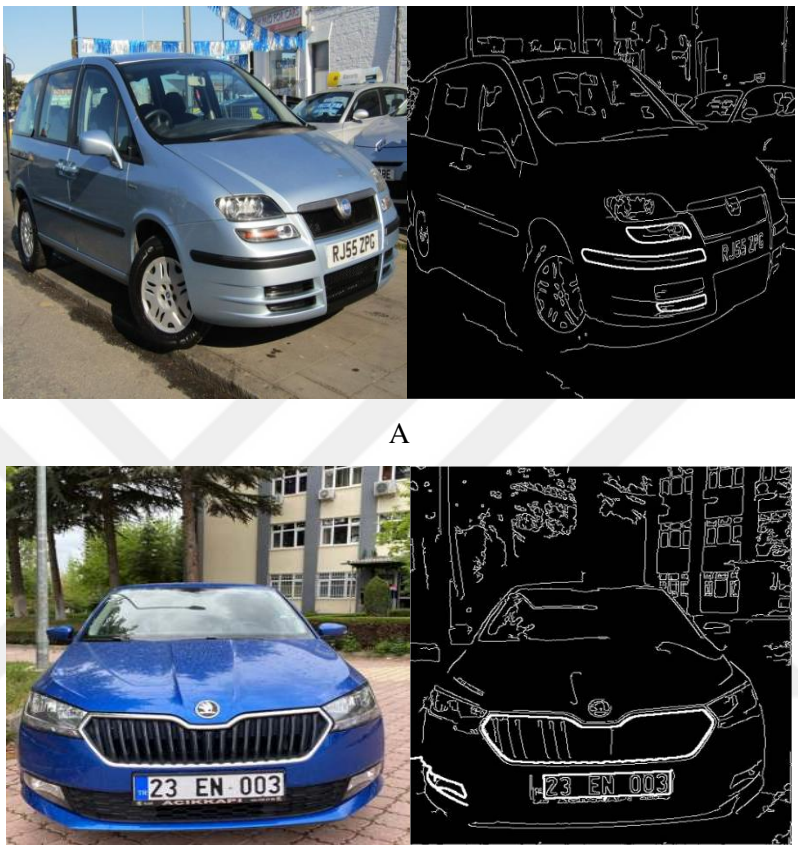
Plaka tanıma sistemlerinde, araçların okuma alanına girmeleri ile plaka belirleme süreçleri başlamaktadır. Araç plakasının belirlenme işlemi üç temel süreçten oluşmaktadır. Araç plakasının doğru bir şekilde tespit edilmesi bu süreçlerin başarılı bir şekilde gerçekleştirilmesine bağlıdır.

- Plaka bölgesinin tespiti
- Plaka bölgesindeki karakterlerin ayrıştırılması
- Plaka karakterlerinin tanınması



Şekil 6.1. Araç plaka tanıma aşamaları

Şekil 6.1'de görüntü işleme algoritmaları kullanılarak gerçekleştirilen araç plaka tanıma sisteminin aşamaları verilmektedir. plaka bölgesinin tespitinin gerçekleştirilmesi için, RGB-Gri tonlamalı renk dönüşümü, pürüzsüzleştirme, görüntüde bulunan kenarların tespiti aşamalarından sonra morfolojik işlemler ve plaka en-boy oranına göre aday bölgelerin tespiti aşamalarından oluşmaktadır.



A

B

Şekil 6.2. (A) farklı açılarda ve (B) dik açıda elde edilen görüntülerde araç plakasının tespiti

Şekil 6.2 (A)'da kameraya göre farklı açılarda bulunan ve (B)'de ise kameraya göre dik açı ile elde edilmiş görüntüler üzerinde, görüntü işleme teknikleri uygulanarak araç plaka bölgesinin tespiti gerçekleştirilmiştir. Standart görüntü işleme tekniklerinin kullanımı ile araç plakasının tespitinin gerçekleştirilmesinde kamera ile plaka bölgesinin arasındaki mesafe ve açı bilgsine bağlı olarak doğru tespitler gerçekleştirilebilmektedir. **Şekil 6.2**'de iki farklı açı ile elde edilmiş görüntü üzerinde standart görüntü işleme teknikleri kullanılarak gerçekleştirilmiştir. Ancak araç plaka bölgesinin tespitinde, temel görüntü işlem tekniklerinin kullanılması yeterli değildir. Bu yüzden farklı teknikler (derin öğrenme tabanlı) kullanılarak gerçekleştirilmesi önerilmektedir.

6.2. Önerilen MVSR Tekniği ile Tespiti ve Karakterlerin Ayrıştırılması

Araç plaka tanıma uygulamalarının dezavantajları ve plaka tanıma için kullanılması gereken yapıların (donanım ve yazılım) oldukça maliyetli sistemler olduğu göz önünde bulundurulmalıdır. Kamera aracılığı ile elde edilen görüntülerde, plaka ve kamera arasındaki mesafe azaldıkça plaka tanıma sistemlerinin maliyeti de azalmaktadır.

A B C

Şekil 6.3. (A) Araç rekli görüntü, (B) gri tonlamalı görüntü ve (C) MVSR normalizasyonun uygulanması

Araç plaka bölgesinin tespiti için, üç boyutlu renkli görünü bir boyutlu gri tonlamalı görüntüye dönüştürülmüş ve ardından MVSR normalizasyon tekniği uygulanmıştır. **Şekil 6.3**'te bu dönüşüme ait sonuçlar gösterilmektedir.

A B C

Şekil 6.4. (A) Araç rekli görüntü, (B) gri tonlamalı görüntü ve (C) MVSR normalizasyonunda eşik değer uygulanması

Şekil 6.3'te elde edilen görüntülere eşik değer algoritması uygulanarak **Şekil 6.4**'te bulunan siyah-beyaz görüntüler elde edilmiştir. **Şekil 6.4** (B)'de bulunan görüntüye eşik değer tekniğinden

önce Bilateral filtre uygulanarak, görüntü pürüzsüzleştirme işlemi gerçekleştirilmiştir. Bu tekniklerin uygulanması ile görüntü üzerinde bulunan gürültülerin ve küçük ayrıntıların elimine edilmesi amaçlanmıştır.

A B C

Şekil 6.5. Plaka bölgesi karakterleri olabilecek kısımların tespiti

Şekil 6.5'te plaka bölgesi olabilecek dikdörtgen bölgeler tespit edilmiştir. Kamera ve araç arasındaki mesafe arttıkça bu dikdörtgen bölgelerin sayısı da artacaktır ve plaka bölgesinin tespiti zorlaşacaktır. Plaka bölgesi olabilecek dikdörtgen bölgeler tespit edilmiştir. Kamera ve araç arasındaki mesafe arttıkça bu dikdörtgen bölgelerin sayısı da artacaktır ve plaka bölgesinin tespiti zorlaşacaktır.

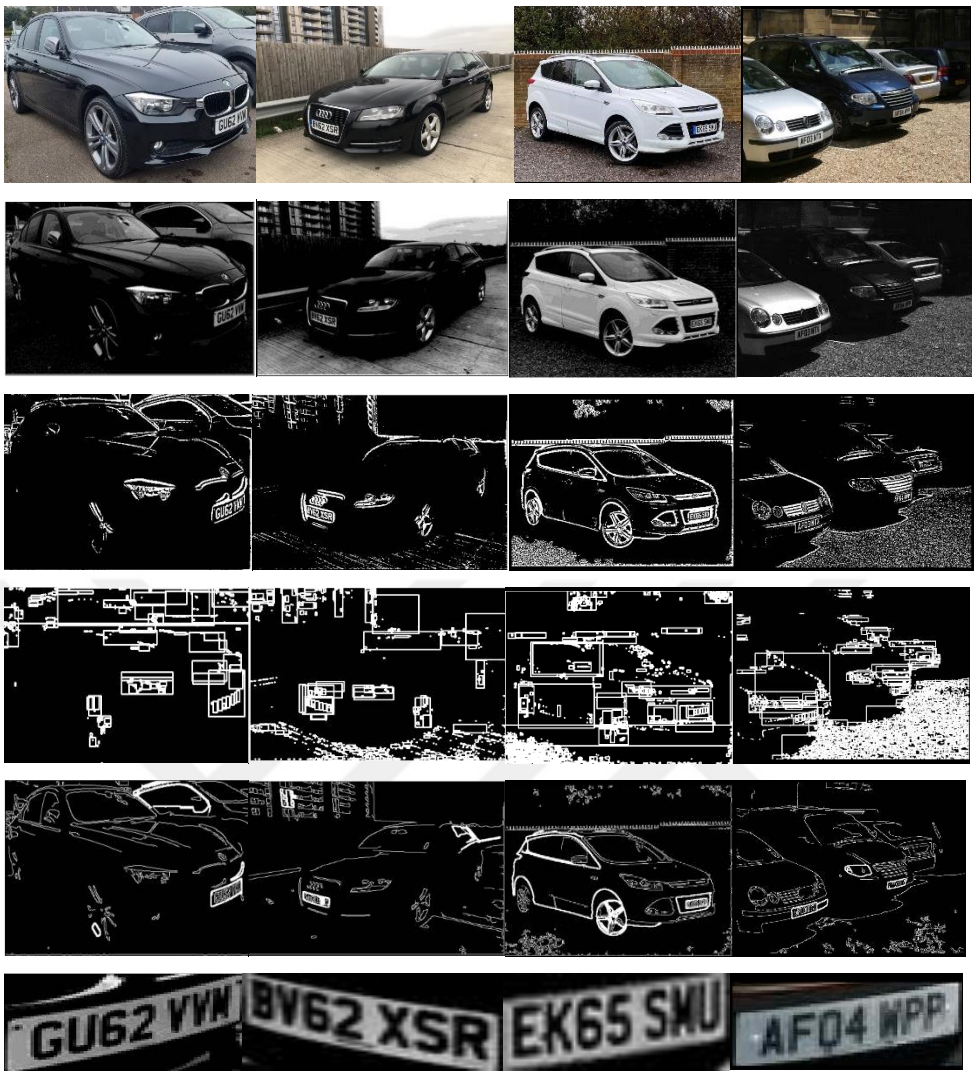
Gri tonlamalı görüntü elde edildikten sonra, gauss filtresi, bilateral filtre ve MVSR normalizayon tekniği uygulandıktan sonra her üç görüntüye ayrı ayrı eşik değer uygulanmıştır. Bu uygulamalardan sonra sırasıyla **Şekil 6.4**'(A-B-C) elde edilmiştir. Görüntü üzerinde araç plaka bölgesi olabilecek kısımların elde edilmesi için bu üç görüntü üzerindeki dikdörtgen bölgelerin tespiti **Şekil 6.5** (A-B-C)'de gösterilmektedir. Görüntü üzerinde tespit edilen plaka bölgesi olmayacak bölgelern sayısı, işlem yükünü, eneri tüketimini ve işlem süresini arttıracaktır. Bundan dolayı, MVSR normalizasyonu uygulanmış görüntüde araç plaka bölgesinin tespiti işlemi daha iyi sonuçlar ile elde edilmiştir.Plaka bölgesi olabilecek aday bölgeler arasından plaka en ve boy oranına göre gerçekleştirilecek işlemler ile plaka boyutlarına sahip olamayacak bölgeler elenmektedir.

Şekil 6.6. Plaka bölgesi olabilecek yerlerin tespiti ve orijinal görüntü üzerinde gösterilmesi

Şekil 6.6'da araç plaka bölgesi olabilecek kapalı dikdörtgen bölgeler tespit edildikten sonra görüntüden bu bölümler kesip çıkartıldıktan sonra aday plaka bölgeleri elde adilmiş olmaktadır. Bu görüntüye göre üç adet plaka bölgesi olabilecek alt görüntüler elde edilmiştir.

Şekil 6.7. Aday plaka bölgeleri

Şekil 6.7'de plaka aday bölgeleri elde edildikten sonra, bu bölgeler orijinal görüntüden RGB renkli olarak çıkartılır. RGB-Gri tonlamalı dönüşüm, görüntü iyileştirme ve adaptif eşik değer algoritması kullanıldıktan sonra morfolojik işlemler uygulanarak, aday plaka bölgesinde plaka karakterlerinin ayrıştırılması için karakterler arasındaki boşluklar belirginleştirilmiş ve görüntü üzerinde bulunan gürültüler elimine edilmiştir.



Şekil 6.8. Farklı açılara sahip araç plakalarının MVSR normalizasyonu ile tespiti

Microsoft Research Cambridge Nesne Tanıma Görüntü Veri tabanından[131] alınan, Farklı açılarda elde edilmiş araç görüntüleri üzerinden plaka bölgesinin tespit edilmesi ve aşamaları **Şekil 6.8**'de gösterilmektedir. Plaka bölgesinin tespit edilmesinin aşamaları;

1. Orijinal görüntünün gri tonlamalı görüntüye dönüştürülmesi
2. MVSR normalizasyon tekniğinin gri tonlamalı görüntüye uygulanması
3. Eşik değer uygulaması ile siyah-beyaz görüntünün elde edilmesi
4. Aday araç plaka bölgesi olabilecek kısımların dikdörtgen bölgeler içine alınması
5. Plaka en boy oranına göre, aday plaka bölgelerinin tespiti
6. Plaka bölgesinin görüntüden çıkartılması işlemleri gerçekleştirilmiştir.

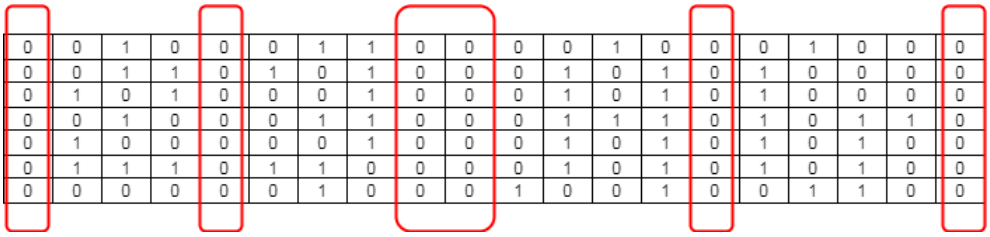
Yukarıda aşamalı ile gösterilen araç plaka belirleme işlemlerinde 2. Basamak olan MVSR normalizasyonu tekniğini uygulamadan araç plaka bölgesi tespit işlemleri gerçekleştirildiğinde,

plaka bölgesinin tespiti gerçekleştirilememiştir. Ancak, MVSR normalizasyonu uygulandıktan sonra, farklı açılardan çekilmiş araç görüntülerinde plaka bölgelerinin tespiti işlemi gerçekleştirilebilmektedir. Farklı açılardan elde edilen görüntülerde plak bölgesinin belirleme %95 oranında gerçekleştirilmiştir.

Şekil 6.7 de aday plaka bölgeleri belirlendikten sonra, bu aşamada, elde edilen aday plaka bölgelerinin boyutları eşitlenir. Eşik değerinin görüntüye göre değişen adaptif otsu algoritması aday bölgelere uygulandıktan sonra siyah ve beyaz piksellerden oluşan bir görüntü oluşacak ve böylelikle bulanıklaştırma ortadan kaldırılacaktır. Bu bölgelere morfoloik açma işlemi uygulayarak görüntüde bulunan objeler arasındaki mesafeler arttırılır (bağlantılar kesilir) ve daha sonra satır ve sütunda birbirinden ayrılmayan beyaz pikseller dikdörtgen içerisine alınır. Türkiye özel araç plakalarında[110] bulunan maksimum karakter sayısından (yedi) az ya da çok dikdörtgen içeren görüntüler elenir ve plaka bölgesi tespit edilmiş olur ya da plaka bölgesi olamayacak şekillere sahip görüntü(ler) elde edilmiş olacaktır.

Şekil 6.9. Plaka bölgesinin elde edilmesi

Şekil 6.9'da elde edilen plaka bölgesi ve bu bölgeye görüntü tersleme uygulanmış halinin elde edilmesi gösterilmektedir. Plaka bölgesi belirlendikten sonra karakterlerin ayrıştırılması işlemi birkaç farklı şekilde gerçekleştirilebilmektedir. Plaka bölgesinde, yatay ve dikey izdüşüm yansıtma tekniği kullanılarak veya bu bölgede **Şekil 6.10**'da gösterildiği gibi düşey doğrultuda (karakterler beyaz arka plan siyah) tüm piksel değeri sıfır (siyah) olan bölgeler karakterler arasındaki boşluklar olarak kabul edilip, karakterlerin ayrıştırılması gerçekleştirilebilir.



Şekil 6.10. Karakterlerin ayrıştırılması

Şekil 6.11. Plaka bölgesi yatay ve dikey yansıtma tekniği

Şekil 6.12. Plaka karakterlerinin ayrıştırılması

Şekil 6.12 ve **Şekil 6.12**'de plaka bölgesindeki karakterlerin ayrıştırılması için yatay ve dikey iz düşüm tekniği ile sadece plaka karakterlerinin bulunduğu bölgeler bulunarak karakterlerin ayrıştırılması gösterilmektedir. Plaka bölgesindeki karakterler, dikdörtgen içerisine alınarak her bir dikdörtgen bölge farklı bir resim olarak alındığında karakterler ayrıştırılmış olacaktır.

Tablo 6.1. Plaka tanıma oranları

	Görüntü sayısı	**Yüzdelik oran**	**MVSR sonuçları**
Plaka yerini saptama	266/221	83	19/20
Karakterlerin tanımlanması	266/201	76	

Tablo 6.1'de plaka tanıma oranları ve belirlenme yüzdeleri sonuçları verilmektedir. standart görüntü işleme teknikleri ile tespit edilemeyen araçlara ait görüntülere, MVSR normalizasyon tekniği uygulanmıştır. Farklı açılardan çekilmiş görüntülerde plaka bölgesinin tespit edilmesi işlemleri %95 oranında gerçekleştirilmiştir.

Tablo 6.2. Farklı görüntü işleme teknikleri ile araç plaka bölgesinin tespit edilmesinin karşılaştırılması

Plaka bölgesinin tespiti	Karakterlerin ayrıştırılması	Plaka tanıma Doğruluk oranı	Kaynaklar
Renk dönüşümü, morfolojik işlemler	İz düşüm tekniği	%66	[52]
Renk dönüşümü, kenar algılama, morfolojik işlemler,	Bağlı Bileşen Analizi (CCA)	%99	[53]
Renk dönüşümü, kenar algılama, morfoloji işlemler	Morfoloji, iz düşüm tekniği	%91	[55, 56]
Temel görüntü işleme teknikleri	İz düşüm tekniği	%99.2	[57]
Renk dönüşümü, morfolojik işlemler	İz düşüm tekniği	%96.2	[58]
MVSR Tekniği	İz düşüm tekniği	%95	Bu çalışmada

Tablo 6.2'de literatürde bulunan, farklı görüntü işleme teknikleri ile araç plaka bölgesinin tespit edilmesinin uygulamalarının karşılaştırılması verilmektedir. plaka bölgesinin tespitinde standart görüntü işleme tekniklerinin kullanılması ile daha düşük oralarda plaka bölgesinin tepiti gerçekleştirilmiştir. Yüksek oranlarda aday plaka bölgesinin gerçekleştirildiği uygulamalarda, kamera ile plaka bölgesi arasında çok az mesafeler bulunmakta ve arka planda bulunan gürültüler bulunmamaktadır.

YOLO, R-CNN ve CNN vb. Derin öğrenme modelleri kullanılarak gerçekleştirilen uygulamalarda araç plaka bölgesinin tespitide oldukça yüksek oranlarda gerçekleştirilebilmektedir. Bu açıdan kıyaslandığında, sadece görüntü normalizasyon tekniği ile gerçekleştirilen litertürdeki uygulamalara ile önerilen MVSR normalizasyon tekniği kullanılarak gerçekleştirilen plaka bölgesinin tespit edilmesi kıyaslandığında, önerilen tekniğin etkinliğinin daha fazla olduğu görülmektedir. Araç plaka bölgesinin tespit edilmesi işleminde, standart yöntemlerde tespiti edilemeyen plaka bölgeleri önerilen yöntem ile tespit edilmiştir.

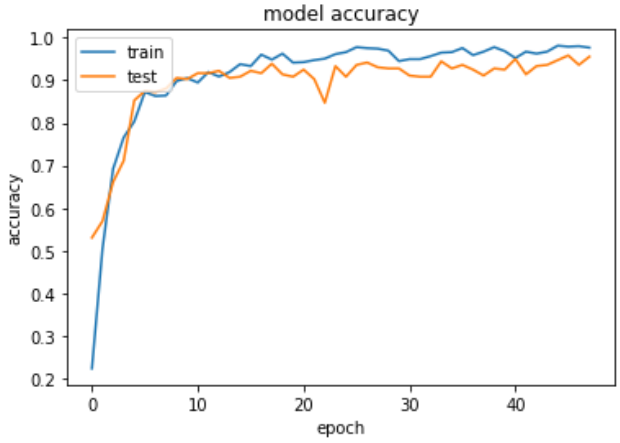
6.3. Araç Plaka Karakterlerinin CNN Model Kullanılarak Sınıflandırılması

Araç plaka tanıma sisteminde, plaka bölgesinin bulunması ve karakterlerin ayrıştırılmasının ardından plaka bölgesinde bulunan karakterlerin belirlenebilmesi için evrişimli sinir ağı (CNN) modeli kullanılmıştır. Bu model eğitimi için görüntüler araç resimlerinden alınmıştır. Elde edilen her bir görüntünün boyutları 32x32x3 olarak yeniden düzenlenmiştir.

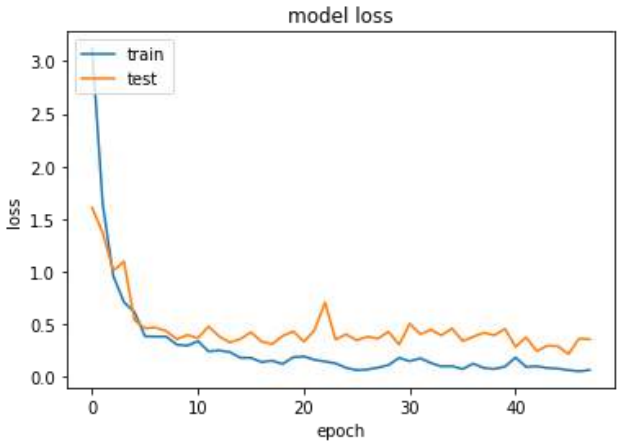
Tablo 6.3. Plaka karakterleri için oluşturulan CNN modelinin tasarımı

Katmanlar	Katman Konfigürasyonları			
	S	N	W	H
Giriş	-	-	32	32
Conv1	3x3	32	32	32
Conv2	3x3	32	32	32
Conv3	3x3	64	32	32
Max-Ortak.	2x2	-	16	16
Conv3	3x3	128	16	16
Max-Ortak.	2x2	-	8	8
Conv4	3x3	64	8	8
Max-Ortak.	2x2	-	4	4
Tam bağl.1		1024		
Tam bağl.2		512		
Tam bağl.3		256		
Çıkış		33		

Araç plaka bölgesinde bulunabilecek otuz üç adet farklı karakterin (on adet rakam ve 23 adet harf) eğitiminde kullanılan CNN modelinin yapısı **Tablo 6.3**'te gösterilmektedir. Evrişim katmanlarında (Conv1…) evirişim işleminin ardından Batch-Norm tekniği ve ReLU aktivasyon fonksiyonu kullanılmıştır. Çıkış katmanında sınıflandırma işleminin softmax aktivasyon fonksiyonu kullanılarak gerçekleştirilmiştir. Hata fonksiyonu hesaplama işleminde ortalama kare hatası kullanılmıştır.



Şekil 6.13. Araç plaka karakterlerine ait model doğruluk grafiği



Şekil 6.14. Araç plaka karakterlerine ait model kayıp grafiği

Araç plaka bölgesi karakterleri ayrıştırma işleminin sonunda, her bir karakterin tek tek modelden geçirilerek tespit işlemi gerçekleştirilmiş olacaktır. Oluşturulan CNN modelinin doğruluk oranı aynı zamanda araç plaka tanıma sisteminin sonucunu da belirleyecektir. Bu yüzden model eğitim sonucunun yüksek olması gerekmektedir. CNN modeli harf ve rakamlardan meydana gelen 1240 adet 32x32 piksel boyutlarında görüntü, 48 döngü (Epoch) ile sınıflandırma işlemi gerçekleştirilmiştir. **Şekil 6.13** ve **Şekil 6.14**'te araç plaka karakterlerine ait model döngü sayısına bağlı, doğruluk ve kayıp grafikleri gösterilmektedir. Eğitim sonucu, model doğruluk oranı % 98.54 ve kayıp 0.074 olarak elde edilmiştir.

6.4. Bölüm değerlendirmesi

Araç plaka bölgesinin tespiti işlemlerinin gerçekleştirilmesinde, standart görüntü işleme tekniklerini kullanılarak gerçekleştirilen uygulamalarda, zorlu koşullarda plaka tespit işlemlerinin gerçekleştirilmesi bazı durumlarda gerçekleştirilememektedir.

Önerilen normalizasyon tekniği, araç plaka bölgesinin belirlenmesi aşamalarında kullanılan gauss ve bilateral filtreleme tekniklerinin yerine kullanılmıştır. Önerilen yöntemin kullanılması ile araç plaka bölgesi olmayan aday bölgelerin sayısı oldukça azaltılmıştır. Böylelikle gerçekleştirilecek işlem yükü azaltılmış ve daha az enerji tüketimi ve daha hızlı araç plaka bölgesinin tespiti işlemleri gerçekleştirilmiştir. Ayrıca zorlu koşullarda (kameranın plaka bölgesini farklı açılarda görmesi), araç plaka bölgesinin elde edilmesi oranı arttırılmıştır.

Araç plaka bölgesinin belirlenmesi sürecinde filtreleme (gauss, bilateral vb.) aşamasında MVSR normalizasyonu tekniğinin uygulaması ile daha önceden belirlenemeyen bazı plaka bölgeleri de başarılı bir şekilde tespit edilmiştir. Böylelikle önerilen MVSR normalizasyonu uygulandıktan sonra, plaka bölgesinin kameraya göre uzaklığı, açısı gibi olumsuz etkiler ile belirlenemeyen bazı görüntülerde de plaka bölgesinin tespiti %95 doğruluk oranında

gerçekleştirilmiştir. MVSR normalizasyonunun uygulanması ile elde edilen sonuçlar Literatürde bulunan diğer uygulamalar ile kıyaslandığında, sadece standart görüntü işleme teknikleri ile gerçekleştirilen uygulamalardan daha iyi sonuçlar elde edilmiştir.

Transfer öğrenme ile yaya ve plaka bölgesi tespiti, Raspberry Pi ve USB kamera kullanılarak gerçek zamanlı olarak gerçekleştirilmiştir. Ancak, saniyede 0.8-1.5 görüntü elde edilebilmektedir. Raspberry Pi kartında ek olarak Coral USB benzeri yapılar kullanılarak fazladan TPU (Tensor Processing Unit) eklenerek saniye başına elde edilen görüntü sayısı (yaklaşık 30 FPS) arttırılabilmektedir. Nesne tespiti veya tanıma işlemleri gerçekleştirilirken, doğruluk oranını arttırmak ve daha iyi sonuçlar elde edebilmek için daha fazla etiketlenmiş görüntü ile model eğitimi gerçekleştirilmesi önerilmektedir. Ayrıca, transfer öğrenme veya model uyarlama (fine tuning) işlemleri kullanılarak gerçekleştirilecek çalışmalarda iyi sonuçlar elde edilebilmektedir.

.

7. SONUÇLAR VE ÖNERİLER

Bu tez kapsamında, gömülü sistem platformlarında görüntü işleme ve derin öğrenme uygulamaları kullanılarak üç farklı dilde (Python, C/C++, VHDL/Verilog) tasarımlar gerçekleştirilmiştir. Raspberry Pi platformu ile gerçekleştirilen uygulamalarında kütüphanelerin kullanımı ile gerçekleştirilen uygulamalarda, zamandan kazanç sağlayıp güç tüketimini ve işlem yükünü azaltmaktadır. Ancak kartın ısınma probleminden dolayı ileri seviye uygulamalar için uygun bulunmamaktadır. İleri seviye tasarımlarda programda donmalar, hatalar ve yanlış sonuçlar verebilmektedir. Daha ileri seviye uygulamalar için Jetson platformları veya Python dili ile programlanabilen PYNQ (Python productivity for Zynq) FPGA kartları kullanılabilir.

Zaman kararlılığı yüksek olan FPGA gömülü sistem platformları ile gerçekleştirilen uygulamalarda, XSG ile gerçekleştirilecek tasarımlar, benzetim blokları ile sınırlı olduğu için, çok karmaşık tasarımlara sahip olmayan gerçek zamanlı temel görüntü işleme uygulamaları tasarlanabilmektedir. XSG ile gerçekleştirilen uygulamalarda aynı zamanda MATLAB programına ihtiyaç duymaktadır.

FPGA uygulamaları için IP blok tasarımları ile uygulamaların gerçekleştirilmesi uygun görülmektedir. Çünkü IP bloklar ile gerçekleştirilen uygulamalarda kaynak kullanımın daha az olduğu gerçekleştirilen uygulamalarda gösterilmiştir. Aynı zamanda birbirinden bağımsız olarak tasarlanan IP çekirdekleri, farklı uygulamalar için farklı konfigürasyonlarda kullanılabilmektedir. Gerçek zamanlı video akışı üzerinde Sobel ve Canny kenar algılama teknikleri gerçekleştirilmiştir. Bu çalışmada, kenar algılama yöntemini uygulamadan önce, kameradan elde edilen RGB565 renkli video akışı gri tonlamalı video formatına dönüştürülmüştür. Elde edilen gri tonlamalı pikseller kullanılarak nesnelerin kenarları tespit edilmiştir. Ayrıca tasarımımız gerçek zamanlı video modu çalıştırma kabiliyeti anlamında diğer çalışmalardan öne çıkmaktadır ve daha az FPGA kaynak kullanımı ile gerçekleştirilmiştir.

Gerçek zamanlı video görüntülerinin FPGA platformunda gerçekleştirilebilmesi için aynı koşullar altında ve aynı uygulamada sabit ve kayan noktalı sayı aritmetiği kullanılarak FPGA kaynak kullanımı ve güç tüketimi karşılaştırılmıştır. Kayan noktalı sayı sistemi üs biti genişliği ile katlanarak artan daha geniş aralıklarda sayılar temsil edilebilmektedir ve sabit noktalı sayı sistemlerine göre sayı hassasiyetleri oldukça yüksektir. FPGA platformunda gerçekleştirilecek uygulamalarda, kayan noktalı sayı sistemine ait aritmetik işlemler, sabit noktalı sayı sistemine göre daha yoğundur. Öte yandan, kayan noktalı sayı gösteriminin kullanıldığı tasarımlarda donanımsal olarak kaynak kullanımı ve güç tüketimi artmaktadır. Ancak, sayı hassasiyetliğinin önemli olduğu uygulamalarda kayan noktalı sayı sisteminin uygulanması önerilmektedir.

Tasarlanan MVSR normalizasyon tekniğinin akciğer X-ışını görüntülerine uygulanması ile Covid-19 hastalığının ön değerlendirmesi kolaylaştırılmıştır. Ayrıca, bu yeni normalleştirme

tekniğinin etkinliği, uzman görüşü, MATLAB, Anaconda Navigator ara yüzü ve FPGA gömülü sistemi içeren farklı değerlendirme yöntemleri kullanılarak kanıtlanmıştır. Uzman görüşüne göre MVSR normalizasyonu, Covid-19 virüsü ile enfekte olan ve olmayan hastalara ait X-ışını görüntülerinin farkı ön değerlendirme aşamasında daha net görülebilmektedir.

MVSR normalizasyonu, akciğer X-ışını röntgen görüntüleri gibi karmaşık görüntülere uygulanabilir. Önerilen MVSR normalizasyon tekniğinin etkisini kanıtlamak ve akciğer röntgeni görüntülerinde Covid-19 virüsünün tespitini gerçekleştirmek için tasarlanan evrişimli yapay sinir ağı modelinin doğruluk oranı önemli artmıştır. Normalizasyon uygulanmayan görüntülerde % 83.01 ve uygulanan görüntülerde ise % 96.16 olarak tespit edilmiştir.

Normalizasyon için oluşturulan model eğitimi için 200 ve test için 365 adet akciğer x-ışını röntgen görüntüsü kullanılmıştır. Özellik çıkartma katmanlarında yedi adet evirişim ve iki sınıflandırma katmanında iki adet tam bağlantı katmanı kullanılmıştır. Evrişim işleminden sonra elde edilen özellik matrislerinde bulunan negatif sayıların öğrenmeye katkılarının ortadan kaldırılmaması için önce batch-normalizasyon ve daha sonra ReLU aktivasyon fonksiyonu kullanılmıştır. Küçük veri seti ile gerçekleştirilecek derin öğrenme modelleri yapay veriler (agumentasyon) ile veri setinde bulunan örneklemeler artırılmalıdır veya transfer öğrenme, fine tuning yöntemleri ile önceden eğitilmiş modeller parametreleri kullanılabilir. Alternatif olarak, SVM gibi modeller küçük veriler için kullanılabilmektedir.

Araç plaka bölgesini tespitinde, temel görüntü işleme algoritmalarının kullanımı önerilmemektedir. Kamera ile araç plakası arasındaki mesafe arttıkça plaka bölgesi olabilecek bölgelerin sayısı artacağından, gerçek plaka bölgesinin tespit edilmesi zorlaşacaktır ve bulunamayacaktır. Önerilen normalizasyon tekniği ile araç plaka bölgesinin belirlenmesi aşamalarında, gauss ve bilateral filtreleme tekniklerinin yerine kullanılmıştır. Önerilen yöntemin kullanılması ile araç plaka bölgesi olmayan aday bölgelerin sayısı oldukça azaltılmıştır. Böylelikle gerçekleştirilecek işlem yükü azaltılmış ve daha az enerji tüketimi ile daha hızlı araç plaka bölgesinin tespiti işlemleri gerçekleştirilmiştir. Önerilen MVSR normalizasyonu uygulandıktan sonra, plaka bölgesinin kameraya göre uzaklığı, açısı gibi olumsuz etkiler ile standart görüntü işleme teknikleri ile belirlenemeyen görüntülerde de plaka bölgesinin tespiti %95 doğruluk oranında gerçekleştirilmiştir. Ancak, plaka bölgesinin belirlenmesi işlemlerinin derin öğrenme modelleri (R-CNN, CNN vb.) kullanılarak tespit edilmesi önerilmektedir.

VHDL/Verilog yazılım dilleri ile oluşturulacak derin öğrenme modelleri, büyük hafıza birimlerine ve FPGA kaynaklarına ihtiyaç duyduğu için bu modellerin kullanımı yalnızca benzetim olarak gerçekleştirilebilmektedir. FPGA'ların programlanmasını kolaylaştırmak için C,C++ benzeri yazılım dilleri (HLS gibi yapılarda) uygulamalar gerçekleştirilebilmektedir. Fakat paralel çalışma mantığına fazla uymadığı için çok nadir tercih edilmektedir. Alternatif olarak Ultrascale + MPSoC

zcu102, zcu104 vb. FPGA platformlarında kullanılabilen Vitis AI veya HLS4ML gibi çatı yapılar (framework) ile makine öğrenmesi uygulamaları gerçekleştirilebilmektedir.

Transfer öğrenme ile yaya ve plaka tespiti, Raspberry Pi ve USB kamera kullanılarak gerçek zamanlı olarak gerçekleştirilmiştir. Ancak elde edilen sonuçlarda saniyede 0.8-1.5 görüntü elde edilebilmektedir. Raspberry Pi kartı ile Coral USB benzeri yapılar ile fazladan TPU (Tensor Processing Unit) eklenerek saniye başına elde edilen görüntü sayısı (yaklaşık 30 FPS) arttırılabilir. Nesne tespiti veya tanıma işlemleri gerçekleştirilirken, doğruluk oranını arttırmak ve daha iyi sonuçlar elde edebilmek için daha fazla etiketlenmiş görüntü ile model eğitimi gerçekleştirilmesi gerekmektedir.

Bu tez çalışmasının kapsamında gerçekleştirilen çalışmalar ile gömülü sistemler, görüntü işleme ve derin öğrenme modelleri kullanımı için gerçekleştirilecek çalışmalara büyük katkılar sağlayacağı düşünülmektedir.

KAYNAKLAR

[1] I. Kuon, J. Rose, Measuring the gap between FPGAs and ASICs, IEEE Trans. Comput. Des. Integr. Circuits Syst. 26 (2007) 203–215. https://doi.org/10.1109/TCAD.2006.884574.

[2] S. Pedre, T. Krajník, E. Todorovich, P. Borensztejn, Accelerating embedded image processing for real time: a case study, J. Real-Time Image Process. (n.d.). https://doi.org/10.1007/s11554-013-0353-2.

[3] M. Fradi, W.E. Youssef, M. Mohsen, The design of an embedded system (SOPC) for an image processing application, in: 2017 Int. Conf. Control. Autom. Diagnosis, ICCAD 2017, Institute of Electrical and Electronics Engineers Inc., 2017: pp. 511–515. https://doi.org/10.1109/CADIAG.2017.8075711.

[4] B. Falsafi, B. Dally, D. Singh, D. Chiou, J.J. Yi, R. Sendag, FPGAs versus GPUs in Data centers, IEEE Micro. 37 (2017) 60–72. https://doi.org/10.1109/MM.2017.19.

[5] A. HajiRassouliha, A.J. Taberner, M.P. Nash, P.M.F. Nielsen, Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms, Signal Process. Image Commun. 68 (2018) 101–119. https://doi.org/10.1016/j.image.2018.07.007.

[6] S. Asano, T. Maruyama, Y. Yamaguchi, Performance comparison of FPGA, GPU and CPU in image processing, in: FPL 09 19th Int. Conf. F. Program. Log. Appl., 2009: pp. 126–131. https://doi.org/10.1109/FPL.2009.5272532.

[7] J. Fowers, G. Brown, P. Cooke, G. Stitt, A performance and energy comparison of FPGAs, GPUs, and multicores for sliding-window applications, in: ACM/SIGDA Int. Symp. F. Program. Gate Arrays - FPGA, 2012: pp. 47–56. https://doi.org/10.1145/2145694.2145704.

[8] J. Choi, R.A. Rutenbar, Video-Rate Stereo Matching Using Markov Random Field TRW-S Inference on a Hybrid CPU+FPGA Computing Platform, IEEE Trans. Circuits Syst. Video Technol. 26 (2016) 385–398. https://doi.org/10.1109/TCSVT.2015.2397198.

[9] S. Mittal, A Survey on optimized implementation of deep learning models on the NVIDIA Jetson platform, J. Syst. Archit. 97 (2019) 428–442. https://doi.org/10.1016/j.sysarc.2019.01.011.

[10] CUDA Toolkit 11.4 Downloads | NVIDIA Developer, (n.d.). https://developer.nvidia.com/cuda-downloads (accessed July 8, 2021).

[11] A.A. Suzen, B. Duman, B. Sen, Benchmark Analysis of Jetson TX2, Jetson Nano and Raspberry PI using Deep-CNN, in: HORA 2020 - 2nd Int. Congr. Human-Computer Interact. Optim. Robot. Appl. Proc., Institute of Electrical and Electronics Engineers Inc., 2020. https://doi.org/10.1109/HORA49412.2020.9152915.

[12] S.S. Hussain, S.S.H. Zaidi, Digital configurable block based implementation of image enhancement algorithm using PSoC 5LP, in: 17th IEEE Int. Multi Top. Conf. Collab. Sustain. Dev. Technol. IEEE INMIC 2014 - Proc., Institute of Electrical and Electronics Engineers Inc., 2015: pp. 182–186. https://doi.org/10.1109/INMIC.2014.7097334.

[13] J. Wang, S. Zhong, L. Yan, Z. Cao, An embedded system-on-chip architecture for real-time visual detection and matching, IEEE Trans. Circuits Syst. Video Technol. 24 (2014) 525–538. https://doi.org/10.1109/TCSVT.2013.2280040.

[14] M. Alareqi, R. Elgouri, M. Tarhda, K. Mateur, A. Zemmouri, A. Mezouari, L. Hlou, Design and FPGA implementation of Real-Time Hardware Co-Simulation for image enhancement in biomedical applications, in: 2017 Int. Conf. Wirel. Technol. Embed. Intell. Syst. WITS

2017, 2017. https://doi.org/10.1109/WITS.2017.7934601.

[15] A. Gur, M. Erim, M. Karakose, Image Processing Based Approach for Crime Scene Investigation Using Drone, in: 2020 Int. Conf. Data Anal. Bus. Ind. W. Towar. a Sustain. Econ. ICDABI 2020, Institute of Electrical and Electronics Engineers Inc., 2020. https://doi.org/10.1109/ICDABI51230.2020.9325606.

[16] H. Andrew G., M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, ArXiv. (2017). https://paperswithcode.com/paper/mobilenets-efficient-convolutional-neural (accessed May 22, 2021).

[17] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (1943) 115–133. https://doi.org/10.1007/BF02478259.

[18] J. McCarthy, M.L. Minsky, N. Rochester, C.E. Shannon, A proposal for the Dartmouth summer research project on artificial intelligence, AI Mag. 27 (2006) 12–14.

[19] Y. LeCun, J. Clifford, D.S. Reisinger, A. Hinton, A Theoretical Framework for Back-Propogation, Proc. 1988 Connect. Model. Summer Sch. (1988) 5–26. https://doi.org/10.2307/j.ctv7cjw41.5.

[20] LECUN, Y., THE MNIST DATABASE of handwritten digits, Http://Yann.Lecun.Com/Exdb/Mnist/. (n.d.). https://ci.nii.ac.jp/naid/10027939599 (accessed April 1, 2021).

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, Li Fei-Fei, ImageNet: A large-scale hierarchical image database, in: Institute of Electrical and Electronics Engineers (IEEE), 2010: pp. 248–255. https://doi.org/10.1109/cvpr.2009.5206848.

[22] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, 2017. https://doi.org/10.1145/3065386.

[23] Z. Zhong, L. Jin, Z. Xie, High performance offline handwritten Chinese character recognition using GoogLeNet and directional feature maps, in: Proc. Int. Conf. Doc. Anal. Recognition, ICDAR, IEEE Computer Society, 2015: pp. 846–850. https://doi.org/10.1109/ICDAR.2015.7333881.

[24] C. Szegedy, W. Liu, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, 2014.

[25] F. Sun, C. Wang, L. Gong, C. Xu, Y. Zhang, Y. Lu, X. Li, X. Zhou, A high-performance accelerator for large-scale convolutional neural networks, in: Proc. - 15th IEEE Int. Symp. Parallel Distrib. Process. with Appl. 16th IEEE Int. Conf. Ubiquitous Comput. Commun. ISPA/IUCC 2017, Institute of Electrical and Electronics Engineers Inc., 2018: pp. 622–629. https://doi.org/10.1109/ISPA/IUCC.2017.00099.

[26] D. Datta, D. Mittal, N.P. Mathew, J. Sairabanu, Comparison of Performance of Parallel Computation of CPU Cores on CNN model, in: Int. Conf. Emerg. Trends Inf. Technol. Eng. Ic-ETITE 2020, Institute of Electrical and Electronics Engineers Inc., 2020. https://doi.org/10.1109/ic-ETITE47903.2020.142.

[27] N. Manikandan, M. Priyanka, Sasikumar, R. Muthaiah, Approximation Computing Techniques to Accelerate CNN Based Image Processing Applications – A Survey in Hardware/Software Perspective, Int. J. Adv. Trends Comput. Sci. Eng. 9 (2020). https://doi.org/10.30534/ijatcse/2020/202932020.

[28] X. Wang, Y. Zhao, F. Pourpanah, Recent advances in deep learning, Int. J. Mach. Learn.

Cybern. 11 (2020) 747–750. https://doi.org/10.1007/s13042-020-01096-5.

[29] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, D.J. Inman, 1D convolutional neural networks and applications: A survey, Mech. Syst. Signal Process. 151 (2021) 107398. https://doi.org/10.1016/j.ymssp.2020.107398.

[30] Y. Song, W. Cai, Y. Zhou, D.D. Feng, Feature-based image patch approximation for lung tissue classification, IEEE Trans. Med. Imaging. 32 (2013) 797–808. https://doi.org/10.1109/TMI.2013.2241448.

[31] Q. Li, W. Cai, X. Wang, Y. Zhou, D.D. Feng, M. Chen, Medical image classification with convolutional neural network, in: 2014 13th Int. Conf. Control Autom. Robot. Vision, ICARCV 2014, Institute of Electrical and Electronics Engineers Inc., 2014: pp. 844–848. https://doi.org/10.1109/ICARCV.2014.7064414.

[32] E. Hosseini-Asl, R. Keynton, A. El-Baz, Alzheimer's disease diagnostics by adaptation of 3D convolutional network, in: Proc. - Int. Conf. Image Process. ICIP, IEEE Computer Society, 2016: pp. 126–130. https://doi.org/10.1109/ICIP.2016.7532332.

[33] X. Zhang, W. Hu, F. Chen, J. Liu, Y. Yang, L. Wang, H. Duan, J. Si, Gastric precancerous diseases classification using CNN with a concise model, PLoS One. 12 (2017) e0185508. https://doi.org/10.1371/journal.pone.0185508.

[34] F.N. Iandola, S. Han, M.W. Moskewicz, K. Ashraf, W.J. Dally, K. Keutzer, SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, (2016). http://arxiv.org/abs/1602.07360 (accessed May 23, 2021).

[35] A. Esteva, B. Kuprel, R.A. Novoa, J. Ko, S.M. Swetter, H.M. Blau, S. Thrun, Dermatologist-level classification of skin cancer with deep neural networks, Nature. 542 (2017) 115–118. https://doi.org/10.1038/nature21056.

[36] S.U. Khan, N. Islam, Z. Jan, I. Ud Din, J.J.P.C. Rodrigues, A novel deep learning based framework for the detection and classification of breast cancer using transfer learning, Pattern Recognit. Lett. 125 (2019) 1–6. https://doi.org/10.1016/j.patrec.2019.03.022.

[37] T. Shanthi, R.S. Sabeenian, Modified Alexnet architecture for classification of diabetic retinopathy images, Comput. Electr. Eng. 76 (2019) 56–64. https://doi.org/10.1016/j.compeleceng.2019.03.004.

[38] S. Lu, Z. Lu, Y.D. Zhang, Pathological brain detection based on AlexNet and transfer learning, J. Comput. Sci. 30 (2019) 41–47. https://doi.org/10.1016/j.jocs.2018.11.008.

[39] M. Hammad, P. Pławiak, K. Wang, U.R. Acharya, ResNet-Attention model for human authentication using ECG signals, in: Expert Syst., Blackwell Publishing Ltd, 2020. https://doi.org/10.1111/exsy.12547.

[40] T. Ozturk, M. Talo, E.A. Yildirim, U.B. Baloglu, O. Yildirim, U. Rajendra Acharya, Automated detection of COVID-19 cases using deep neural networks with X-ray images, Comput. Biol. Med. 121 (2020) 103792. https://doi.org/10.1016/j .compbiomed.2020.103792.

[41] S. Bhattacharya, P.K. Reddy Maddikunta, Q.V. Pham, T.R. Gadekallu, S.R. Krishnan S, C.L. Chowdhary, M. Alazab, M. Jalil Piran, Deep learning and medical image processing for coronavirus (COVID-19) pandemic: A survey, Sustain. Cities Soc. 65 (2021) 102589. https://doi.org/10.1016/j.scs.2020.102589.

[42] S. Karim, Y. Zhang, A.A. Laghari, M.R. Asif, Image processing based proposed drone for detecting and controlling street crimes, in: Int. Conf. Commun. Technol. Proceedings,

ICCT, Institute of Electrical and Electronics Engineers Inc., 2018: pp. 1725–1730. https://doi.org/10.1109/ICCT.2017.8359925.

[43] Z. Xu, C. Cheng, V. Sugumaran, Big data analytics of crime prevention and control based on image processing upon cloud computing, J. Surveillance, Secur. Saf. 1 (2020) 16–33. https://doi.org/10.20517/jsss.2020.04.

[44] U. Asıl, E. Nasibov, USING IMAGE PROCESSING TECHNIQUES TO INCREASE SAFETY IN SHOOTING RANGES, Int. J. Comput. Sci. Eng. Appl. 11 (2021). https://doi.org/10.5121/ijcsea.2021.11103.

[45] S. Priya, T. Ashok Kumar, V. Paul, A novel approach to fabric defect detection using digital image processing, in: 2011 - Int. Conf. Signal Process. Commun. Comput. Netw. Technol. ICSCCN-2011, 2011: pp. 228–232. https://doi.org/10.1109/ICSCCN.2011.6024549.

[46] Z. Liu, B. Wang, C. Li, M. Yu, S. Ding, Fabric defect detection based on deep-feature and low-rank decomposition, J. Eng. Fiber. Fabr. 15 (2020). https://doi.org/10.1177/1558925020903026.

[47] P. Revathi, M. Hemalatha, Classification of cotton leaf spot diseases using image processing edge detection techniques, in: IEEE Proc. Int. Conf. Emerg. Trends Sci. Eng. Technol. Recent Adv. Sci. Eng. Innov. INCOSET 2012, Institute of Electrical and Electronics Engineers Inc., 2014: pp. 169–173. https://doi.org/10.1109/incoset.2012.6513900.

[48] S. Naik, B. Patel, Machine Vision based Fruit Classification and Grading-A Review, 2017.

[49] N. Zhu, X. Liu, Z. Liu, K. Hu, Y. Wang, J. Tan, M. Huang, Q. Zhu, X. Ji, Y. Jiang, Y. Guo, Deep learning for smart agriculture: Concepts, tools, applications, and opportunities, Int. J. Agric. Biol. Eng. 11 (2018) 32–44. https://doi.org/10.25165/ijabe.v11i4.4475.

[50] S.A. Peixoto, F.F.X. Vasconcelos, M.T. Guimarães, A.G. Medeiros, P.A.L. Rego, A. V. Lira Neto, V.H.C. de Albuquerque, P.P. Rebouças Filho, A high-efficiency energy and storage approach for IoT applications of facial recognition, Image Vis. Comput. 96 (2020) 103899. https://doi.org/10.1016/j.imavis.2020.103899.

[51] A.P. Nagare, License Plate Character Recognition System using Neural Network, 2011. http://www.photocop.com/recognition.htm. (accessed April 10, 2021).

[52] X. Zhai, F. Bensaali, S. Ramalingam, Real-time license plate localisation on FPGA, in: IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work., IEEE Computer Society, 2011: pp. 14–19. https://doi.org/10.1109/CVPRW.2011.5981739.

[53] C. Indravadanbhai Patel, D. Patel, C. Patel Smt Chandaben Mohanbhai, A. Patel, S. Chandaben Mohanbhai, D. Patel Smt Chandaben Mohanbhai, Optical Character Recognition by Open source OCR Tool Tesseract: A Case Study, Int. J. Comput. Appl. 55 (2012) 975–8887. https://doi.org/10.5120/8794-2784.

[54] S. Du, M. Ibrahim, M. Shehata, W. Badawy, Automatic license plate recognition (ALPR): A state-of-the-art review, IEEE Trans. Circuits Syst. Video Technol. 23 (2013) 311–325. https://doi.org/10.1109/TCSVT.2012.2203741.

[55] M.A. Massoud, M. Sabee, M. Gergais, R. Bakhit, Automated new license plate recognition in Egypt, Alexandria Eng. J. 52 (2013) 319–326. https://doi.org/10.1016/j.aej.2013.02.005.

[56] D. Zang, Z. Chai, J. Zhang, D. Zhang, J. Cheng, Vehicle license plate recognition using visual attention model and deep learning, J. Electron. Imaging. 24 (2015) 033001. https://doi.org/10.1117/1.JEI.24.3.033001.

[57] X. Zhai, F. Bensaali, Improved number plate character segmentation algorithm and its efficient FPGA implementation, J. Real-Time Image Process. 10 (2015) 91–103. https://doi.org/10.1007/s11554-012-0258-5.

[58] R. Fan, V. Prokhorov, N. Dahnoun, Faster-than-real-time linear lane detection implementation using SoC DSP TMS320C6678, in: IST 2016 - 2016 IEEE Int. Conf. Imaging Syst. Tech. Proc., 2016: pp. 306–311. https://doi.org/10.1109/IST.2016.7738242.

[59] J.A.F. Calderon, J.S. Vargas, A. Perez-Ruiz, License plate recognition for Colombian private vehicles based on an embedded system using the ZedBoard, in: 2016 IEEE Colomb. Conf. Robot. Autom. CCRA 2016 - Conf. Proc., Institute of Electrical and Electronics Engineers Inc., 2017. https://doi.org/10.1109/CCRA.2016.7811426.

[60] P. Promrit, W. Suntiamorntut, Design and development of lane detection based on FPGA, in: Proc. 2017 14th Int. Jt. Conf. Comput. Sci. Softw. Eng. JCSSE 2017, Institute of Electrical and Electronics Engineers Inc., 2017. https://doi.org/10.1109/JCSSE.2017.8025909.

[61] S. Jagannathan, K. Desappan, P. Swami, M. Mathew, S. Nagori, K. Chitnis, Y. Marathe, D. Poddar, S. Narayanan, Efficient object detection and classification on low power embedded systems, in: 2017 IEEE Int. Conf. Consum. Electron. ICCE 2017, 2017: pp. 233–234. https://doi.org/978-1-5090-6389-5/17/$31.00.

[62] H. Li, P. Wang, M. You, C. Shen, Reading car license plates using deep neural networks, Image Vis. Comput. 72 (2018) 14–23. https://doi.org/10.1016/j.imavis.2018.02.002.

[63] M.A. Rafique, W. Pedrycz, M. Jeon, Vehicle license plate detection using region-based convolutional neural networks, Soft Comput. 22 (2018) 6429–6440. https://doi.org/10.1007/s00500-017-2696-2.

[64] R. Girshick, J. Donahue, T. Darrell, J. Malik, Region-Based Convolutional Networks for Accurate Object Detection and Segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 38 (2016) 142–158. https://doi.org/10.1109/TPAMI.2015.2437384.

[65] L. Xie, T. Ahmad, L. Jin, Y. Liu, S. Zhang, A New CNN-Based Method for Multi-Directional Car License Plate Detection, IEEE Trans. Intell. Transp. Syst. 19 (2018) 507–517. https://doi.org/10.1109/TITS.2017.2784093.

[66] Hendry, R.C. Chen, Automatic License Plate Recognition via sliding-window darknet-YOLO deep learning, Image Vis. Comput. 87 (2019) 47–56. https://doi.org/10.1016/j.imavis.2019.04.007.

[67] V.R. Viju, Radha, License plate recognition based on K-means clustering algorithm, in: Intell. Syst. Ref. Libr., Springer, 2019: pp. 23–29. https://doi.org/10.1007/978-3-030-32644-9_3.

[68] B.B. Yousif, M.M. Ata, N. Fawzy, M. Obaya, Toward an Optimized Neutrosophic k-Means with Genetic Algorithm for Automatic Vehicle License Plate Recognition (ONKM-AVLPR), IEEE Access. 8 (2020) 49285–49312. https://doi.org/10.1109/ACCESS.2020.2979185.

[69] L. Zhang, P. Wang, H. Li, Z. Li, C. Shen, Y. Zhang, A Robust Attentional Framework for License Plate Recognition in the Wild, ArXiv. (2020). https://doi.org/10.1109/tits.2020.3000072.

[70] A.O. Agbeyangi, O.A. Alashiri, A.E. Otunuga, Automatic Identification of Vehicle Plate Number using Raspberry Pi, in: 2020 Int. Conf. Math. Comput. Eng. Comput. Sci. ICMCECS 2020, Institute of Electrical and Electronics Engineers Inc., 2020.

https://doi.org/10.1109/ICMCECS47690.2020.246983.

[71] L.S. Fernandes, F.H.S. Silva, E.F. Ohata, A. Medeiros, A.V.L. Neto, Y.L.B. Nogueira, P.A.L. Rego, P.P.R. Filho, A Robust Automatic License Plate Recognition System for Embedded Devices, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), Springer Science and Business Media Deutschland GmbH, 2020: pp. 226–239. https://doi.org/10.1007/978-3-030-61377-8_16.

[72] D.M.F. Izidio, · Antonyus, P.A. Ferreira, H.R. Medeiros, E.N. Da, S. Barros, An embedded automatic license plate recognition system using deep learning, Des. Autom. Embed. Syst. 24 (2020) 23–43. https://doi.org/10.1007/s10617-019-09230-5.

[73] R.D. Castro-Zunti, J. Yépez, S.B. Ko, License plate segmentation and recognition system using deep learning and OpenVINO, IET Intell. Transp. Syst. 14 (2020) 119–126. https://doi.org/10.1049/iet-its.2019.0481.

[74] W. Weihong, T. Jiaoyang, Research on License Plate Recognition Algorithms Based on Deep Learning in Complex Environment, IEEE Access. 8 (2020) 91661–91675. https://doi.org/10.1109/ACCESS.2020.2994287.

[75] L. Ma, Y. Zhang, Research on vehicle license plate recognition technology based on deep convolutional neural networks, Microprocess. Microsyst. 82 (2021) 103932. https://doi.org/10.1016/j.micpro.2021.103932.

[76] M. Heidari, S. Mirniaharikandehei, A.Z. Khuzani, G. Danala, Y. Qiu, B. Zheng, Improving the performance of CNN to predict the likelihood of COVID-19 using chest X-ray images with preprocessing algorithms, Int. J. Med. Inform. 144 (2020) 104284. https://doi.org/10.1016/j.ijmedinf.2020.104284.

[77] S. Wang, B. Kang, J. Ma, X. Zeng, M. Xiao, J. Guo, M. Cai, J. Yang, Y. Li, X. Meng, B. Xu, A deep learning algorithm using CT images to screen for Corona virus disease (COVID-19), Eur. Radiol. (2021) 1–9. https://doi.org/10.1007/s00330-021-07715-1.

[78] S.R. Nayak, D.R. Nayak, U. Sinha, V. Arora, R.B. Pachori, Application of deep learning techniques for detection of COVID-19 cases using chest X-ray images: A comprehensive study, Biomed. Signal Process. Control. 64 (2021) 102365. https://doi.org/10.1016/j.bspc.2020.102365.

[79] T. Rahman, A. Khandakar, Y. Qiblawey, A. Tahir, S. Kiranyaz, S. Bin Abul Kashem, M.T. Islam, S. Al Maadeed, S.M. Zughaier, M.S. Khan, M.E.H. Chowdhury, Exploring the effect of image enhancement techniques on COVID-19 detection using chest X-ray images, Comput. Biol. Med. 132 (2021) 104319. https://doi.org/10.1016/j.compbiomed.2021.104319.

[80] G. Berthou, K. Marquet, T. Risset, G. Salagnac, Accurate Power Consumption Evaluation for Peripherals in Ultra Low-Power embedded systems, in: GIoTS 2020 - Glob. Internet Things Summit, Proc., Institute of Electrical and Electronics Engineers Inc., 2020. https://doi.org/10.1109/GIOTS49054.2020.9119593.

[81] D. Iqbal, A. Abbas, M. Ali, M.U.S. Khan, R. Nawaz, Requirement Validation for Embedded Systems in Automotive Industry through Modeling, IEEE Access. 8 (2020) 8697–8719. https://doi.org/10.1109/ACCESS.2019.2963774.

[82] H. Shahinzadeh, J. Moradi, G.B. Gharehpetian, H. Nafisi, M. Abedi, IoT Architecture for smart grids, in: Int. Conf. Prot. Autom. Power Syst. IPAPS 2019, Institute of Electrical and Electronics Engineers Inc., 2019: pp. 22–30. https://doi.org/10.1109/IPAPS.2019.8641944.

[83] SYSBIOS Operating system (OS) | TI.com, (n.d.). https://www.ti.com/tool/SYSBIOS

(accessed May 20, 2021).

[84] GPU Applications | High Performance Computing | NVIDIA, (n.d.). https://www.nvidia.com/en-us/gpu-accelerated-applications/ (accessed May 20, 2021).

[85] S. Brown, J. Rose, FPGA and CPLD architectures: A tutorial, IEEE Des. Test Comput. 13 (1996) 42–56. https://doi.org/10.1109/54.500200.

[86] Configurable Logic Blocks (CLBs) on an FPGA - LabVIEW Communications System Design Suite 5.0 Manual - National Instruments, (n.d.). https://www.ni.com/documentation/en/labview-comms/5.0/fpga-targets/configurable-logic-blocks/ (accessed May 20, 2021).

[87] Xilinx, Inc, XC4000E and XC4000X Series Features • System featured Field-Programmable Gate Arrays-SelectRAM TM memory: on-chip ultra-fast RAM with-synchronous write option-dual-port RAM option-Fully PCI compliant (speed grades-2 and faster), n.d. http://www.xilinx.com/xlnx/xweb/xil_publications_index.jsp (accessed May 29, 2021).

[88] Y.K. Lim, L. Kleeman, T. Drummond, Algorithmic methodologies for FPGA-based vision, Mach. Vis. Appl. 24 (2013) 1197–1211. https://doi.org/10.1007/s00138-012-0474-9.

[89] Site Keyword Search, (n.d.). https://www.xilinx.com/search/site-keyword-search.html#q=ip core (accessed May 11, 2021).

[90] S. Yaman, B. Karakaya, Y. Erol, Customized IP-Core based Video Processing on FPGA, in: Int. Conf. Power Control Embed. Syst., 2019.

[91] M.A. ARSERIM, C. HAYDAROĞLU, H. ACAR, A. UÇAR, Forming and Co-simulation of Square and Triangular Waveforms by Using System Generator, Balk. J. Electr. Comput. Eng. 7 (2019).

[92] Xilinx, Inc, Vivado Design Suite Tutorial: Model-Based DSP Design using Add-on for MATLAB and Simulink, (n.d.). www.xilinx.com (accessed July 9, 2021).

[93] M. Geier, M. Brandle, D. Faller, S. Chakraborty, Debugging FPGA-accelerated Real-time Systems, in: Proc. IEEE Real-Time Embed. Technol. Appl. Symp. RTAS, Institute of Electrical and Electronics Engineers Inc., 2020: pp. 350–363. https://doi.org/10.1109/RTAS48715.2020.00010.

[94] V. Vujović, M. Maksimović, Raspberry Pi as a Wireless Sensor node: Performances and constraints, in: 2014 37th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2014 - Proc., IEEE Computer Society, 2014: pp. 1013–1018. https://doi.org/10.1109/MIPRO.2014.6859717.

[95] D. Pena, A. Forembski, X. Xu, D. Moloney, Benchmarking of CNNs for Low-Cost, Low-Power Robotics Applications, in: n.d.

[96] D. Kang, D.H. Kang, J. Kang, S. Yoo, S. Ha, Joint optimization of speed, accuracy, and energy for embedded image recognition systems, in: Proc. 2018 Des. Autom. Test Eur. Conf. Exhib. DATE 2018, Institute of Electrical and Electronics Engineers Inc., 2018: pp. 715–720. https://doi.org/10.23919/DATE.2018.8342102.

[97] M. El Fezazi, A. Achmamad, M. Aqil, A. Jbari, PSoC-Based Embedded Instrumentation and Processing of sEMG Signals, Analog Integr. Circuits Signal Process. (2021) 1–16. https://doi.org/10.1007/s10470-021-01850-x.

[98] H. Lu, Y. Li, M. Chen, H. Kim, S. Serikawa, Brain Intelligence: Go beyond Artificial

Intelligence, Mob. Networks Appl. 23 (2018) 368–375. https://doi.org/10.1007/s11036-017-0932-8.

[99] Y. Lecun, Y. Bengio, G. Hinton, Deep learning, Nature. 521 (2015) 436–444. https://doi.org/10.1038/nature14539.

[100] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, 2016. https://www.deeplearningbook.org/ (accessed April 22, 2021).

[101] N. Wiener, Cybernetics or Control and Communication in the Animal and the Machine, MIT Press, 2009. https://books.google.com.tr/books?hl=tr&lr=&id=s-mvDwAAQBAJ&oi=fnd&pg=PR7&dq=Cybernetics&ots=1y5XB-x2Sb&sig=YonZxhYMWKQH0hC756-Nr_7HmRc&redir_esc=y#v=onepage&q=Cybernetics&f=false (accessed April 22, 2021).

[102] Washington, History of Neuroscience, (n.d.). http://faculty.washington.edu/chudler/hist.html (accessed April 22, 2021).

[103] I. Sutskever, R. Jozefowicz, K. Gregor, D. Rezende, T. Lillicrap, O. Vinyals, Towards Prıncıpled Unsupervısed Learnıng, in: Int. Conf. Learn. Represent., 2016.

[104] D. Masters, C. Luschi, Revisiting Small Batch Training for Deep Neural Networks, ArXiv. (2018). http://arxiv.org/abs/1804.07612 (accessed April 24, 2021).

[105] I. Kandel, M. Castelli, The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset, ICT Express. 6 (2020) 312–315. https://doi.org/10.1016/j.icte.2020.04.010.

[106] K.A. Sankararaman, S. De, Z. Xu, W.R. Huang, T. Goldstein, The Impact of Neural Network Overparameterization on Gradient Confusion and Stochastic Gradient Descent, PMLR, 2020. http://proceedings.mlr.press/v119/sankararaman20a.html (accessed April 24, 2021).

[107] S. Doshi, Various Optimization Algorithms For Training Neural Network | by Sanket Doshi | Towards Data Science, (2019). https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6 (accessed April 25, 2021).

[108] Cross entropy - Wikipedia, (n.d.). https://en.wikipedia.org/wiki/Cross_entropy (accessed April 25, 2021).

[109] Y. Bengio, Deep Learning of Representations for Unsupervised and Transfer Learning, 2012. http://www.causality.inf.ethz.ch/unsupervised-learning.php (accessed April 26, 2021).

[110] Y. Bengio, Practical Recommendations for Gradient-Based Training of Deep Architectures, 2012. http://deeplearning.net/software/pylearn2 (accessed June 9, 2021).

[111] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: 32nd Int. Conf. Mach. Learn. ICML 2015, International Machine Learning Society (IMLS), 2015: pp. 448–456. https://arxiv.org/abs/1502.03167v3 (accessed April 26, 2021).

[112] Z. Mushtaq, S.F. Su, Environmental sound classification using a regularized deep convolutional neural network with data augmentation, Appl. Acoust. 167 (2020) 107389. https://doi.org/10.1016/j.apacoust.2020.107389.

[113] M. Sun, Z. Song, X. Jiang, J. Pan, Y. Pang, Learning Pooling for Convolutional Neural Network, Neurocomputing. 224 (2017) 96–104. https://doi.org /10.1016/j.

neucom.2016.10.049.

[114] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, n.d.

[115] E.J. Teoh, K.C. Tan, C. Xiang, Estimating the number of hidden neurons in a feedforward network using the singular value decomposition, IEEE Trans. Neural Networks. 17 (2006) 1623–1629. https://doi.org/10.1109/TNN.2006.880582.

[116] G. Panchal, A. Ganatra, Y.P. Kosta, D. Panchal, Behaviour Analysis of Multilayer Perceptrons with Multiple Hidden Neurons and Hidden Layers, Int. J. Comput. Theory Eng. 3 (2011). http://www.ijcte.org/papers/328-L318.pdf (accessed April 22, 2021).

[117] K. Fukushima, S. Miyake, Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition, in: Springer, Berlin, Heidelberg, 1982: pp. 267–285. https://doi.org/10.1007/978-3-642-46466-9_18.

[118] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE. 86 (1998) 2278–2323. https://doi.org/10.1109/5.726791.

[119] Y. Xu, M. Vaziri-Pashkam, Limits to visual representational correspondence between convolutional neural networks and the human brain, Nat. Commun. 12 (2021) 1–16. https://doi.org/10.1038/s41467-021-22244-7.

[120] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, Comput. Vis. Found. (2014). http://arxiv. (accessed May 8, 2021).

[121] C. Garbin, X. Zhu, O. Marques, X. Zhu, Dropout vs. batch normalization: an empirical study of their impact to deep learning, Multimed. Tools Appl. 79 (2020) 12777–12815. https://doi.org/10.1007/s11042-019-08453-9.

[122] Home - OpenCV, (n.d.). https://opencv.org/ (accessed May 15, 2021).

[123] E. Ternovoy, M.G. Popov, D. V. Kaleev, Y. V. Savchenko, A.L. Pereverzev, Comparative Analysis of Floating-Point Accuracy of IEEE 754 and Posit Standards, in: Proc. 2020 IEEE Conf. Russ. Young Res. Electr. Electron. Eng. EIConRus 2020, Institute of Electrical and Electronics Engineers Inc., 2020: pp. 1883–1886. https://doi.org/10.1109/EIConRus49466.2020.9039521.

[124] J.Y.F. Tong, D. Nagle, R.A. Rutenbar, Reducing power by optimizing the necessary precision/range of floating-point arithmetic, IEEE Trans. Very Large Scale Integr. Syst. 8 (2000) 273–286. https://doi.org/10.1109/92.845894.

[125] D. Menard, D. Chillet, O. Sentieys, Floating-to-fixed-point conversion for digital signal processors, EURASIP J. Appl. Signal Processing. 2006 (2006) 1–19. https://doi.org/10.1155/ASP/2006/96421.

[126] X. Lian, Z. Liu, Z. Song, J. Dai, W. Zhou, X. Ji, High-performance fpga-based cnn accelerator with block-floating-point arithmetic, IEEE Trans. Very Large Scale Integr. Syst. 27 (2019) 1874–1885. https://doi.org/10.1109/TVLSI.2019.2913958.

[127] Analog Discovery 2 - Digilent Reference, (n.d.). https://reference.digilentinc.com/test-and-measurement/analog-discovery-2/start (accessed June 30, 2021).

[128] W. Rong, Z. Li, W. Zhang, L. Sun, An improved Canny edge detection algorithm, in: 2014 IEEE Int. Conf. Mechatronics Autom. IEEE ICMA 2014, IEEE Computer Society, 2014: pp. 577–582. https://doi.org/10.1109/ICMA.2014.6885761.

[129] Y. Jang, J. Mun, J. Kim, Resource-efficient FPGA architecture of canny edge detector, in: ISOCC 2016 - Int. SoC Des. Conf. Smart SoC Intell. Things, Institute of Electrical and Electronics Engineers Inc., 2016: pp. 299–300. https://doi.org/10.1 109/ISOCC. 2016.7799796.

[130] Y.H. Kwon, J.W. Jeon, Comparison of FPGA Implemented Sobel Edge Detector and Canny Edge Detector, in: 2020 IEEE Int. Conf. Consum. Electron. - Asia, ICCE-Asia 2020, Institute of Electrical and Electronics Engineers Inc., 2020. https://doi.org/10.1109/ICCE-Asia49877.2020.9277425.

[131] Download Microsoft Research Cambridge Object Recognition Image Database from Official Microsoft Download Center, (n.d.). https://www.microsoft.com/en-us/download/confirmation.aspx?id=52644 (accessed June 20, 2021).