MASİF PANEL ÜRETİMİNDE KULLANILAN LAMEL PARÇALARI ÜZERİNDE NESNE TESPİTİ VE SINIFLANDIRILMASI

Merve ÖZKAN

2022
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

Tez Danışmanı
Dr. Öğr. Üyesi Caner ÖZCAN

MASİF PANEL ÜRETİMİNDE KULLANILAN LAMEL PARÇALARI ÜZERİNDE NESNE TESPİTİ VE SINIFLANDIRILMASI

Merve ÖZKAN

Tez Danışmanı
Dr. Öğr. Üyesi Caner ÖZCAN

T.C.
Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalında
Yüksek Lisans Tezi
Olarak Hazırlanmıştır

KARABÜK
Aralık 2022

Merve ÖZKAN tarafından hazırlanan "MASİF PANEL ÜRETİMİNDE KULLANILAN LAMEL PARÇALARI ÜZERİNDE NESNE TESPİTİ VE SINIFLANDIRILMASI" başlıklı bu tezin Yüksek Lisans Tezi olarak uygun olduğunu onaylarım.

Dr. Öğr. Üyesi Caner ÖZCAN .....
Tez Danışmanı, Bilgisayar Mühendisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından Oy Birliği ile Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir. 27/12/2022

Ünvanı, Adı SOYADI (Kurumu) İmzası

Başkan : Dr. Öğr. Üyesi Emel SOYLU (SAMÜ) .....

Üye : Dr. Öğr. Üyesi Caner ÖZCAN (KBÜ) .....

Üye : Dr. Öğr. Üyesi Kürşat Mustafa KARAOĞLAN (KBÜ) .....

KBÜ Lisansüstü Eğitim Enstitüsü Yönetim Kurulu, bu tez ile, Yüksek Lisans derecesini onamıştır.

Doç. Dr. Müslüm KUZU .....
Lisansüstü Eğitim Enstitüsü Müdürü

"Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim."

Merve ÖZKAN

ÖZET

Yüksek Lisans Tezi

MASİF PANEL ÜRETİMİNDE KULLANILAN LAMEL PARÇALARI ÜZERİNDE NESNE TESPİTİ VE SINIFLANDIRILMASI

Merve ÖZKAN

Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Dr. Öğr. Üyesi Caner ÖZCAN
Aralık 2022, 76 sayfa

Bu çalışmada, masif panel üretimi esnasında lamel parçalarının kalite sınıflandırması aşamasında kullanılan lamel görüntüleri üzerinde nesne tespit işlemi ve nihai sınıflandırma işlemi gerçekleştirilmiştir. Masif panel üretilirken aynı kalitede lamel parçalarının aynı masif panel levhası üzerinde bir arada olması hedeflenir. Masif panel sektöründe üretim gerçekleştiren firmaların birçoğu ilgili aşamada lamel parçalarının sınıflandırılması işlemini kalite kontrol ekibine bırakır. Kalite kontrol ekibi insan gücüne dayalı çalışmaktadır. Lamellerin kalitelerine göre sınıflandırılması işlemi dikkat gerektiren bir süreçtir. Bu önemli işlemin insan gücüne dayalı olması birçok problemi de beraberinde getirmektedir. Üretimde verim kaybı, zaman planlamasında sapma, üretilen panelin maliyetinin altında satılması, müşteriye karşı güven kaybı gibi sorunlar bunlardan başlıcalarıdır. Bu sebepten tam bu noktada lamel sınıflandırma işlemini derin öğrenme ağı vasıtasıyla otomatikleştirecek uzman bir sisteme ihtiyaç duyulmuştur. Çalışma kapsamında lamel görüntüleri üzerindeki sınıflandırmayı

etkileyen faktörlerden budak, yatay desen ve çatlak nesnelerinin tespiti sonrasında başarım metriklerinin elde edilmesi, nihai sınıflandırma işleminin gerçekleştirilmesi aşamaları yer almaktadır. Toplam 2972 adet görüntü kullanılmıştır. Bu görüntülerden 2480 adeti budak, yatay desen ve çatlak modellerine ait eğitim işlemi esnasında kullanılmıştır. Bu görüntüler üzerinde toplam erişilen etiket sayısı 12044'tür. Eğitim sonrasında modellerin başarım metriklerini ölçebilmek için 360 adetlik bir test veri seti oluşturulmuştur. Bir sonraki aşama olan nihai sınıflandırma işleminde kullanılacak test veri setinin sayısı ise 132'dir. Tamamı toplandığında elde edilen rakam 2972 olmaktadır. Görüntüler üzerinde Mask R-CNN ağı kullanılarak eğitim işlemi gerçekleştirilmiştir. Bu tez çalışması kapsamında eğitim işlemi için i9 10980XE işlemcili ve NVIDIA Quadro RTX 5000 ekran kartına sahip bir bilgisayar kullanılmıştır. Görüntüler ise Fujifilm X-S1 12MP kamera vasıtasıyla alınmıştır. Python diliyle birlikte TensorFlow ve Keras kütüphaneleri kullanılmıştır. Mask R-CNN'de seçilen omurga ağı ResNet-101'dir. Deneysel çalışmalar sonucunda gerekli görülen kod blokları, şekil ve çizelgeler yardımıyla daha detaylı analiz gerçekleştirilerek sonuçlar verilmiştir. Öneriler kısmında ise gelecek çalışmalarda hangi problemlerin çözüleceği, hazırda tahmin edilemeyen ya da yanlış tahmin edilen görsellerdeki problemlerin neden kaynaklandığına ve bir sonraki adımda neler yapılabileceğine dair bilgiler sunulmaktadır.

Anahtar Sözcükler : Lamel veri seti, masif panel üretimi, bilgisayar destekli tespit, lamel sınıflandırma, Mask R-CNN, segmentasyon, derin öğrenme.

Bilim Kodu : 92414.

ABSTRACT

M. Sc. Thesis

OBJECT DETECTION AND CLASSIFICATION ON LAMELLA PIECES USED IN SOLID PANEL PRODUCTION

Merve ÖZKAN

Karabük University
Institute of Graduate Programs
Department of Computer Engineering

Thesis Advisor:
Assist. Prof. Dr. Caner ÖZCAN
December 2022, 76 pages

In this study, object detection and final classification processes were carried out on the lamella images used in the quality classification of lamella pieces during the production of massive panels. While producing the massive panel, it is aimed to have the same quality lamella pieces together on the same solid panel plate. Most of the companies producing in the solid panel sector leave the classification of the lamella parts to the quality control team. The quality control team works based on manpower. The process of classifying lamellas according to their quality is a process that requires attention. The fact that this important process is based on manpower brings with it many problems. Problems such as loss of efficiency in production, deviation in time planning, sale of the produced panel below its cost, loss of trust towards the customer are the main ones. For this reason, at this point, there was a need for an expert system to automate the lamella classification process through a deep learning network. Within the scope of the study, there are stages of obtaining performance metrics after the

detection of knots, horizontal patterns and crack objects, which are among the factors affecting the classification on the lamella images, and performing the final classification process. A total of 2972 images were used. During the training of knot, horizontal pattern and crack models, 2480 images were used. The total number of tags accessed on these images is 12044. After the training, a 360 test data set was created to measure the performance metrics of the models. The number of test data set to be used in the final classification process, which is the next step, is 132. When all are added together, the resulting figure is 2972. Training was performed on the images using Mask R-CNN network. In this thesis, a computer with i9 10980XE processor and NVIDIA Quadro RTX 5000 graphics card was used for the training process. Images were taken with the Fujifilm X-S1 12MP camera. TensorFlow and Keras libraries are used with Python language. The backbone network chosen in Mask R-CNN is ResNet-101. As a result of the experimental studies, more detailed analysis was carried out with the help of code blocks, figures and tables, which were deemed necessary, and the results were given. In the suggestions section, information on which problems will be solved in future studies, why the problems in the images that cannot be predicted or incorrectly estimated, and what can be done in the next step are presented.

Key Word **:** Lamella dataset, massive panel production, computer aided detection, lamella classification, Mask R-CNN, segmentation, deep learning

Science Code : 92414.

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın hocam Dr. Öğr. Üyesi Caner ÖZCAN'a sonsuz teşekkürlerimi sunarım.

Çalışmamın tüm zorlu süreçleri boyunca yanımda olan, duraksadığımda bana yeniden başlama azmi veren, tüm yoğunluklarıma rağmen sabır ile yaklaşan ve benden desteğini hiç esirgemeyen eşim Emre ÖZKAN'a teşekkür ederim.

Çalışma kapsamında kullanılan veri kümesinin anlamlandırılması esnasında yardımlarını esirgemeyen Kastamonu Ticaret Borsası Genel Sekreteri Orman Yüksek Mühendisi Emre UZER'e teşekkürlerimi sunarım.

Ek olarak bu tez çalışmamızı "KBÜBAP-22-YL-096" proje numarası ile destekleyen Karabük Üniversitesi Bilimsel Araştırma Projeleri Birimi'ne teşekkürlerimi sunarım.

Sevgili aileme manevi hiçbir yardımı esirgemeden yanımda oldukları için tüm kalbimle teşekkür ederim.

İÇİNDEKİLER

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

SİMGELER

$\mathscr{L}$	: kayıp fonksiyonu değeri
$\mathscr{L}cls$	: sınıf kayıp değeri
$\mathscr{L}box$	: sınıflandırmadan alınan sınırlayıcı kutucuk değeri
$\mathscr{L}mask$	: segmentasyon esnasında alınan maske değeri
W	: ankor en değeri
H	: ankor boy değeri
k	: kutucuk sayısı
$\sum_{i=1}^{N} APi$	: ortalama kesinlik değerleri için toplam sembolü

KISALTMALAR

API : Application Programming Interface (Uygulama Programlama Arayüzü)

CNN : Convolutional Neural Network (Evrişimli Sinir Ağları)

CUDA : Compute Unified Device Architecture (Birleşik Aygıt Mimarisi Hesaplama)

cuDNN : CUDA Deep Neural Network Library (CUDA Derin Sinir Ağları Kütüphanesi)

DN : Doğru Negatif

DÖ : Derin Öğrenme

DP : Doğru Pozitif

DVM : Destek Vektör Makineleri

Faster R-CNN : Faster Region Based Convolutional Neural Network (Daha Hızlı Bölge Tabanlı Evrişimli Sinir Ağı)

Fast R-CNN : Fast Region Based Convolutional Neural Network (Hızlı Bölge Tabanlı Evrişimli Sinir Ağı)

FPN : Feature Pyramid Network (Özellik Piramit Ağı)

GPU : Graphics Proccesing Unit (Grafik İşlem Birimi)

IoU : Intersection Over Union (Birleşim Üzerinden Kesişim)

mAP : Mean Average Precision (Ortalama Hassasiyet Ortalaması)

Mask R-CNN : Mask Region Based Convolutional Neural Network (Maske Bölge Tabanlı Evrişimli Sinir Ağı)

MÖ : Makine Öğrenimi

SSD : Single Shot Detector (Tek Atış Dedektörü)

R-CNN : Region Based Convolutional Neural Network (Bölge Tabanlı Evrişimli Sinir Ağı)

RoI : Region of Interest (İlgi Bölgesi)

RoI Align : Region of Interest Align (İlgi Bölgesi Hizalama)

RoI Pooling : Region of Interest Pooling (İlgi Bölgesi Havuzlama)

RPN : Region Proposal Network (Bölge Teklif Ağı)

YN : Yanlış Negatif

YOLO : You Only Look Once (Yalnızca Bir Kez Bak)

YP : Yanlış Pozitif

YSA : Yapay Sinir Ağları

BÖLÜM 1

GİRİŞ

Türkiye'deki orman varlığı çeşitliliği sebebiyle orman endüstri sektörü her geçen gün gelişim yaşamaktadır. Orman endüstrisi alanında pazar payına sahip firmalar müşterilerine çok çeşitli ürün yelpazesi sunmaktadır. Bu ürünlerden biri de masif paneldir. Masif paneller kalitelerine göre müşterilere pazarlanmaktadır. Farklı kalitede masif panellerin farklı alanlarda kullanımı gerçekleşir.

Masif panel üretimi esnasında aynı kalitedeki lamellerin aynı kalite masif panelde birleştirilmesi gerekir. Belirlenen kalitelerin içerisinde yer alan yapıların haricinde başka bir yapı içeren lamelin ilgili kalite kategorisinde yer almaması zorunludur. Lamellerin kalite ayırma işlemi masif panel üretim zincirinde yer alan kalite kontrol birimi tarafından gerçekleştirilmektedir. Burada işlem tamamıyla insan gücüne dayanmaktadır. Masif panel üretimi gerçekleştiren tesislerin kalite ayırma birimlerindeki çalışanlar işlemi hızlı gerçekleştirebilmek, konsantrasyon eksikliği, deneyimsizlik gibi etkenler nedeniyle hataya düşmektedir. Bu aşamada yapılan hata müşteri memnuniyetsizliği, üretimde verim kaybının yaşanması, üretilen panelin maliyetinin düşmesi gibi durumlara sebep olmaktadır. Tüm bu sebepler kalite kontrol işleminin yapay zeka destekli uzman bir sistem tarafından gerçekleştirilmesi gerektiğini ortaya koymaktadır.

Yapılan bu tez çalışmasında kaliteyi etkileyen budak, yatay desen ve çatlak yapılarının segmentasyonu ve tespiti ilk aşamada gerçekleştirilmiştir. Elde edilen modeller üzerinde başarım metriklerine ait değerler ölçülmüştür. Segmente edilen yapılara göre kalite sınıflarına ayrılması ikinci aşamada uygulanmıştır. Bu çalışma sayesinde masif panel üretimi esnasında tamamen insan gücüne dayalı işlem yapan kalite kontrol biriminin ortadan kaldırılması, sürecin makine tarafından kontrol edilmesi, insan

tarafından oluşan hataların yok edilmesi, üretimde verimliliğin artırılması hedeflenmiştir. Ayrıca bu sayede teknolojik ilerlemeler açısından yetersiz olan orman endüstri tesislerinin teknolojik gelişim kaydetmesi sağlanacaktır.

Yapılan tez çalışmasında lameller üzerinde yer alan yapılarda segmentasyon işlemi gerçekleştirildi. Bu yapıların tespitinin gerçekleşmesi aşamasında yer alan ilk adım maskeleme işlemidir. Maskeleme işlemi en önemli adımlardan biridir. Budak, yatay desen ve çatlak nesneleri poligonal olarak etiketlenmiştir. Etiketleme işlemi web tabanlı uygulama vasıtasıyla gerçekleştirilmiştir. Sonrasında ise evrişimli sinir ağı mimarisi olan Mask R-CNN kullanılmıştır.

Gerçekleştirilen bu çalışmada yer alan ilk bölüm olan giriş bölümünde çalışmanın kısa özeti yer almaktadır. İkinci bölümde ise masif panel üretiminde kullanılan ağaç türlerinden ve üretim sürecinden kısaca bahsedilmiştir. Veri kümesi ve ilgili veri setinin açıklamasına bu bölümde ayrıca yer verilmiştir. Orman endüstri sektöründe kaydedilen gelişmeler geniş bir literatür taramasıyla sunulmuştur. Bir sonraki bölüm olan üçüncü bölümde Mask R-CNN detaylı bir şekilde anlatılmıştır. Dördüncü bölümde ise budak, yatay desen ve çatlak yapılarının tespiti gerçekleştirilmiş ve ikinci aşamada temsili bir sınıflandırma işlemi oluşturulmuştur. Beşinci bölümde Mask R-CNN vasıtasıyla gerçekleştirilen deneysel çalışma sonucunda ortaya çıkan veriler yer almaktadır. Çalışmanın son bölümü olan altıncı bölümde ise beşinci bölümde sunulan bulgular çalışmanın hedefine uygun bir şekilde yorumlanarak sonuçlandırılmıştır. Budak, yatay desen, çatlak nesnelerinin analizi ve verilerin anlamlandırılması tablo ve grafiksel sonuçlar üzerinden detaylı bir şekilde sunulmuştur.

BÖLÜM 2

MASİF PANEL

Masif panel aynı kalitede yer alan küçük kesitli lamellerin uç uca ya da solid halde eklenmesiyle oluşturulan latalardan elde edilen küçük ahşap kesitlerdir. Üretim başlangıcında hangi tür ağaçtan yapılacağına karar verilmesi gerekmektedir. Yerli ya da yabancı ülke kaynaklarından birinci sınıf ya da ikinci sınıf olarak seçilir. Masif panelin oluşumu seçilen ağaçların kurutma fırınına girmesiyle başlamaktadır. Üretim esnasında uzun soluklu bir aşamadan geçmekte ve üretim sürecini tamamladıktan sonra müşteriye sunulmaktadır. Ahşap mutfak, banyo tezgahı, mobilya, oyun parkları, duvar, kolon ve tavan kaplama, ofis mobilyaları, okul mobilyaları, ahşap kiriş gibi birçok alanda kullanılmaktadır [1]. Diğer ahşap ürünlere göre sahip oldukları avantajlar sayesinde piyasada daha çok kullanım alanına sahiptir. Kullanım açısından ömrünün uzun olması, makinelerde işlenebilirliğinin kolay olması bu avantajlardan bir kaçıdır [2]. En önemli etkenlerden biri de ağaç malzemenin mekanik özellikleri ve kullanımı açısından kuvvete karşı dayanıklılığıdır [3]. Tüm bu sebeplerden masif panel orman ürünleri sektöründe giderek artan bir öneme sahip olmuştur.

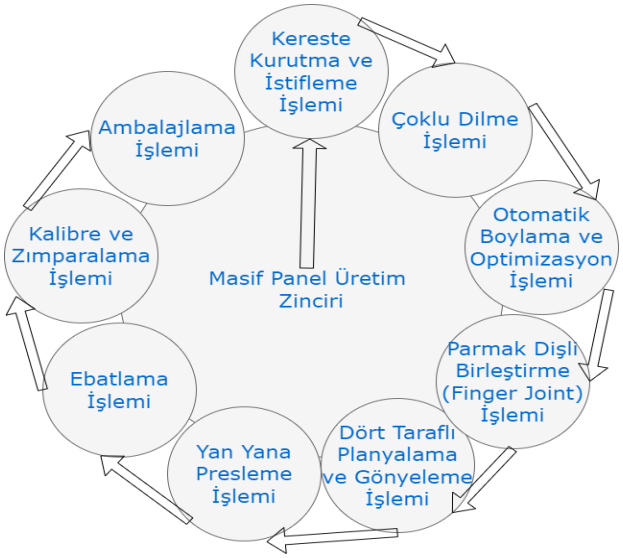
Masif panel üretimi gerçekleştiren tesislerde ürün talepleri yüksek olmasına rağmen, buradaki üretim zincirini otomatikleştirecek, insan gücüne olan ihtiyacı tamamen ortadan kaldırabilecek bir sistem henüz ülkemizde yer almamaktadır. Masif panel üretim sürecinde ağaç malzeme lamellere ayrılır. Aynı kalitede lameller birleştirilerek aynı kalitede paneller elde edilir. Lamellerin aynı kalitede olduğuna karar verilmesi durumu ülkemizdeki birçok tesiste kalite kontrol ekibi tarafından tamamen insan gücüne dayalı olarak gerçekleşmektedir. Lamellerin kalite ayrımı esnasında herhangi bir hata meydana gelmemesi zorunludur. Bu işlemin insan gücüne dayalı olması üretimdeki verimliliği kötü yönde etkilemektedir. Ayrıca insani sebeplerden dolayı yapılan hatalar panelin satış fiyatını düşürmektedir. Masif panele talebin fazla olması ve üretim sürecinde yaşanan bu negatif etkinin ortadan kaldırılması adına kalite

kontrol ekibine destek olacak ya da yerine geçebilecek yapay zeka destekli uzman bir sistem tasarımı bu noktada gereklidir.

Tüm bu durumlar göz önünde bulundurulduğunda bu çalışma lamel parçalarının ilk aşamada üzerindeki yapıların tespitine ikinci aşamada ise temsili sınıflandırılması işlemine odaklanmıştır.

2.1. MASİF PANEL ÜRETİM SÜRECİ

Kaliteli ağacın seçilmesiyle başlayan üretim süreci fırınlama işlemiyle devam eder [4].



Şekil 2.1. Masif panel üretim aşamaları

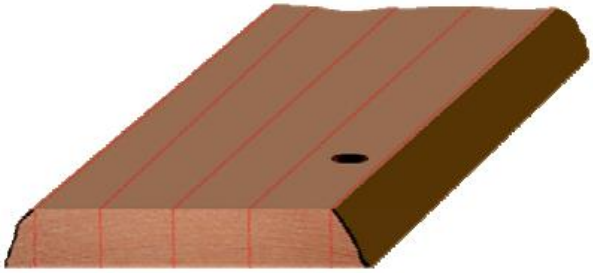
Şekil 2.1'de masif panel üretim aşamaları listelenmiş ve detayları aşağıdaki başlıklarda verilmiştir.

2.1.1. Kurutma İşlemi

Her ağacın olması gerektiği nem oranı farklıdır. İmalat için kullanılacak olan kereste (lata, tahta, kalas) ağaç türlerine göre belirlenen nem oranına kurutma ve buharlama fırınları vasıtasıyla ulaştırılır.

2.1.2. Çoklu Dilme İşlemi

Ham malzemenin bir sonraki aşamaya geçmek için uygun hale getirildiği kısımdır. Bu aşamada latanın dilimleme işlemi sonrasındaki görüntüsüne göre tasnif yapılmaktadır.

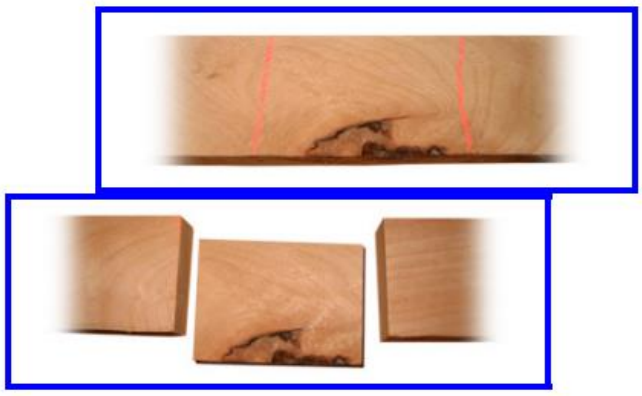


Şekil 2.2. Sulamalı tahtanın masif panel üretimi için boylanması işlemi [5]

Üretilecek olan panelin kalınlık bilgilerine göre çoklu dilme Şekil 2.2'de gösterildiği şekilde gerçekleştirilir.

2.1.3. Otomatik Boylama ve Optimizasyon İşlemi

Bu aşamada kullanılan makineye göre kusur tespiti ya makine tarafından otomatik ya da insan tarafından el ile gerçekleştirilmektedir. Budak tespitini otomatik olarak yapan makineler günümüzde yer almaktadır [6]. Otomatik olarak kusur tespiti yapan makineleri kullanmayan tesislerde ise bu işlem insan gücüne dayalıdır.



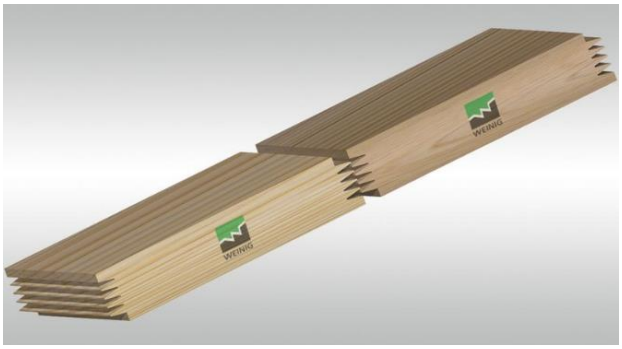
Şekil 2.3. Latalarda kusurlu kısımların tespit edilip ayrılması işlemi [5]

Makinenin giriş kısmında yer alan çalışan, kusurlu yapıyı tespit edip fosforlu çizim yapan bir araçla ilgili kısmı ayırıp makineye verir. Makine bu işareti görüp kusurlu kısmı hatasız parçadan ayırır. Bu işlem Şekil 2.3'te detaylı bir şekilde gösterilmektedir.

Sonrasında elde edilen lamel parçaları üzerinde yer alan yapılara göre kalite kontrol ekibi tarafından sınıflandırma işlemi gerçekleştirilir. Kaliteye ayrılmaya uygun olan parçalar belirlenir ve kalite kontrol ekibi tarafından incelenmek üzere bir sonraki aşamaya geçer. Kaliteye ayrılmaya uygun görülmeyen parçalar en ya da kalınlık olarak üretilecek olan masif panele uygun olmayan parçalardır. Bu parçalar aynı süreçlerden geçirilmek üzere ilgili kısımlara gönderilir. Kalitelendirilmeye uygun görülen parçalar üzerinde kalite kontrol ekibinin ele aldığı yapılar kayın ağacını örnek verecek olursak budak, yatay desen, yeşillenme, mantarlanma ve çatlak nesneleridir. Lamellerin üzerinde bu yapıların bulunup bulunmaması durumuna ya da nesnelerin boyutunun lamel üzerinde kapladığı alana göre hangi kaliteye ayrılacağına karar verilir. Bu aşamadaki işlem birçok fabrikada tamamıyla insan gücüne dayalıdır. Ayrıca kalite sınıfları müşterinin isteğine göre değişkenlik göstermektedir.

2.1.4. Parmak Dişli Birleştirme (Finger Joint) İşlemi

Ağaç en yüksek direncini daima liflere paralel olarak sergilemektedir. İki lamel parçasını liflerine paralel birleştirmek oldukça zordur. Finger joint işlemi sayesinde Şekil 2.4'te görüldüğü gibi açılan dişler tutkal yardımıyla birleştirilir.



Şekil 2.4. Finger joint işlemi ile birleştirilecek olan lamel örneği [7]

Aynı kalitede lamellerde açılan dişler birbirine eklenerek boy uzatma işlemi gerçekleştirilir. Şekil 2.5'te örneğine yer verilmiştir. Literatürde ahşap panellerin kenarlarındaki tutkal kalıntı kusurlarının tespiti ile alakalı yapılan çalışma 0,97 hassasiyet değeri ile başarıya ulaşmıştır [8]. Makalede önerilen model sayesinde tutkal kalıntılarının başarılı tespiti gerçekleştirilebilmektedir.

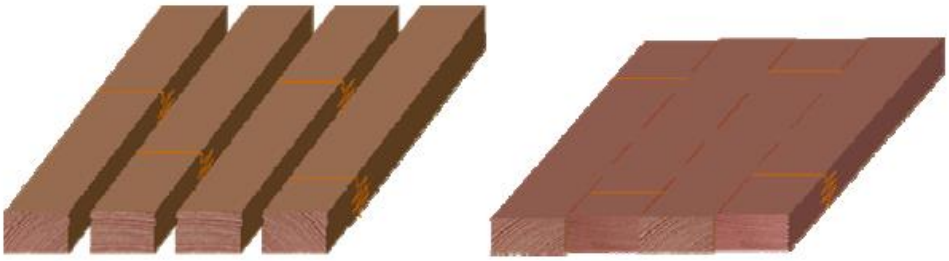
Şekil 2.5. Finger joint işlemi sonrası oluşan latalardan örnek görüntü

2.1.5. Planyalama İşlemi

Presleme işlemine geçmeden önce parmak dişli sonrasında elde edilen latalar tüm yüzeylerinin pürüzsüz olması amacıyla dört taraflı planyalama işlemi gerçekleştirilmelidir. Latalar yan yana preslenirken yüzeylerin tutkalı kabul edebilmesi açısından bu aşamada yapılan işlem önemlidir. Bu aşamada lata en ya da kalınlık olarak incelmektedir. Hangi yüzeyinde pürüz varsa işlem yapan makine o yüzeye odaklanır ve ilgili kısmı inceltir.

2.1.6. Presleme İşlemi

Planyalama işlemi sonrasında oluşan her lataya eşit ısı ve basınç uygulanarak lataların boydan birbiriyle tutunması sağlanır. Şekil 2.6'da lataların birleştirilmesi örneğine yer verilmiştir.



Şekil 2.6. Lataların presleme vasıtasıyla birleştirilmesi işlemi [5]

2.1.7. Ebatlama İşlemi

Oluşan panelin kenarlarındaki pürüzler kesilerek ebatlanmaktadır. Ebatlama işlemi yatay ve dikey olabilir.

2.1.8. Kalibre ve Zımparalama İşlemi

Oluşan panelin yüzeyinde pürüz varsa pürüzün giderilebileceği üretimin son aşamasıdır. Üretilen panel mobilya endüstrisinde birçok yerde kullanılacağından bu yüzeyinin düzgün, pürüzsüz ve temiz olması zorunludur.

Şekil 2.7. Masif panelde sorunlu yüzey tipleri a) Pürüzlü b) Çukur içeren

Şekil 2.7'de yer alan sorunların tamamıyla giderilmesi gerektiği bu aşama masif panel üretimi için çok önemlidir.

2.1.9. Ambalajlama İşlemi

Zımpara aşamasını geçen paneller paketleme aşamasına geçer. Bu aşamada vakumlu paketleme makinesi yardımıyla paneller paketlenir ve satış için hazır hale getirilir.

2.2. MASİF PANEL ÜRETİMİNDE KULLANILAN AĞAÇ TÜRLERİ

Masif panel birçok çeşitte ağaç türünden meydana gelir. Ağaç türüne göre kalitelendirme aşamasında kalite ayırma biriminin dikkat ettiği yapılar değişmektedir. Literatürde ormanlardaki ağaç türlerinin sınıflandırılması derin öğrenme algoritmaları kullanılarak etkili bir şekilde gerçekleştirilmiştir [9].

Ağaç malzemenin cinsi kullanılacağı yere göre seçilmelidir. Bu şekilde bir seçim yapıldığında malzemeden yüksek oranda verim alınır. Naive Bayes sınıflandırma teknikleri kullanılarak literatürde ağaç türlerinin tahmini yapılmış ve diğer tahmin yöntemlerine göre hem zamandan hem de maliyetten tasarruf sağlanmıştır [10]. 14 farklı Avrupa ağaç türü kullanılarak yapılan ağaç cinsi tahmin çalışmasında %93 oranında başarı elde edilmiştir [11]. Ayrıca ağaçların fiziko-mekanik özellikleri bakımından sınıflandırılması ağaçlarının kuvvete karşı dayanıklılığının ne düzeyde olduğunu göstermek için kullanılır [12].

Çizelge 2.1. Masif panel üretimi aşamasında kullanılan ağaç cinsleri [5]

Yapraklı Ağaç Türleri	**İğne Yapraklı Ağaç Türleri**	**Tropik Ağaç Türleri**
kayın, kestane, meşe, ceviz, dişbudak, kavak, kiraz, armut, ıhlamur, akçaağaç	ladin, sarıçam, göknar, larix	sapelli, iroko, teak, limba, wenge, bubinga

Çizelge 2.1'de masif panel yapımında kullanılabilen ağaç çeşitleri yer almaktadır. Bu çalışmada kayın ağacı üzerinden elde edilen görüntülerde sınıflandırma işlemi gerçekleştirilmiştir. Kayın ağacı üzerinde yer alan ve sınıflandırmayı etkileyen yapılar;

budak, yatay desen, yeşillenme, mantarlanma, çatlaktır. Bu çalışma için ele alınan yapılar ise budak, yatay desen ve çatlaktır.

2.3. LAMEL VERİ KÜMESİ VE VERİ ETİKETLEME AŞAMALARI

Bölüm 2.2'de masif panel üretiminde kullanılan ağaç cinslerinin detayları verilmiştir. Her ağaç cinsine göre kaliteyi etkileyen nesneler değişiklik göstermektedir. Bu çalışmada yapraklı ağaç türlerinden olan kayın ağacı ele alınmıştır. Sert ağaç türlerinden biri olan bu ağaç tipi üzerinde nesne tespiti yapmak şekilleri sebebiyle oldukça zordur. Literatürde yapılan çalışmalardan mermer yüzeylerinin sınıflandırılmasını bu kısımda ele alacak olursak ilgili yüzeylerin sınıflandırılması da oldukça zordur, buna rağmen evrişimli sinir ağları çözüm üretebilmiştir [13,14].

Orman endüstri alanında yapay zeka sistemlerine üretim verimliliği ve zamandan tasarruf için ihtiyaç vardır. Sektördeki çoğu yönetici bu fikir altında birleşmektedir. Masif panel üretimi yapan tesislerden yola çıkacak olursak bu alanda ilerlemenin kaydedilebilmesi için daha çok araştırmacılara açık görüntülere ihtiyaç vardır. Literatürde ülkemizde lamel görüntüleri üzerine yapılan çalışmalar yurtdışında yapılan çalışmalardan oldukça azdır. Bu alanda veriye ulaşılmasının güç olması bu çalışmaların az olmasının sebeplerinden biridir.

Bu tez çalışmasında hem literatürdeki bu açığın yerini doldurmak hem de orman endüstri sektöründe yapay zeka uygulamalarında gelişme kaydedebilmek hedeflenmiştir. Bu hedef kapsamında orman ürünleri sektöründe üretim gerçekleştiren bir tesisin masif panel üretim hattından bir uzman yardımıyla seçilen özel lamel görüntüleri üzerinde çalışılmıştır. İlerleyen bölümlerde bu görüntüler üzerinde yer alan yapıların detaylı bilgileri açıklanmış ve elde edilen parametreler paylaşılmıştır.

2.3.1. Lamel Veri Kümesi

Bu tez kapsamında Kastamonu Ticaret Borsası Ağaç ve Orman Ürünleri San. A.Ş. firmasının masif panel üretim hattından alınan özel lamel görüntüleri üzerinde çalışılmıştır. Firmadan gerekli izinler sağlanmıştır. Görüntüler toplanırken FujiFilm

X-S1 12MP kamera kullanılmıştır. Kameranın piksel yoğunluğu 326 dpi ve ekran çözünürlüğü ise 783 x 587'dir. Görüntülerin anlamlandırılması işlemi bir uzman yardımıyla gerçekleştirilmiş ve görüntülerde kullanılan ağaç tipi yapraklı ağaç türlerinden olan kayın ağacı olarak seçilmiştir.

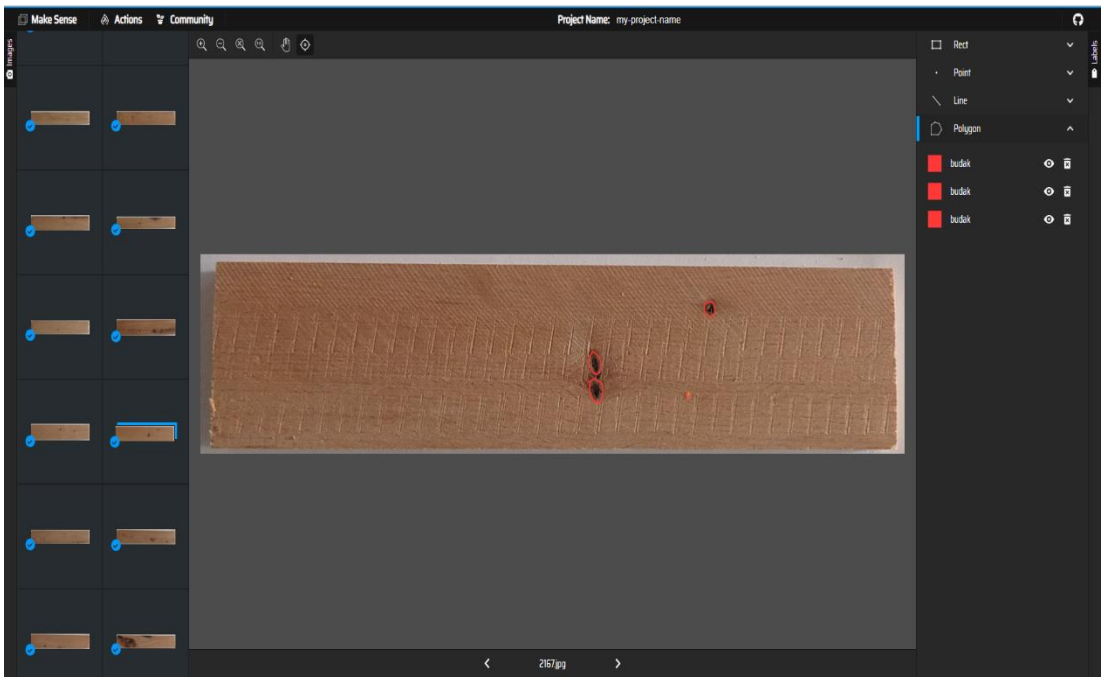
Çalışmada kullanılan toplam görüntü sayısı 2972'dir. Çalışmada kullanılan veri kümesinde 3 ayrı nesne tespit işlemi yapılmıştır. Budak sınıfına ait toplam görüntü sayısı 656 adettir. Eğitim için ayrılan görüntü sayısı 584, test için ayrılan görüntü sayısı ise 72'dir. Bir lamel görüntüsü üzerinde birden fazla budak yer alabilir. Eğitim için ayrılan toplam etiket sayısı 889 iken, test için ayrılan toplam etiket sayısı 117'dir. Yatay desen sınıfına ait toplam görüntü sayısı 237'dir. Eğitim için ayrılan görüntü sayısı 211 iken test için ayrılan görüntü sayısı 26 olmuştur. Bir lamel görüntüsü üzerinde genelde çok sayıda yatay desen yer alır. Yatay desen sınıfına ait toplam etiket sayısı 3103'tür. Bu sayının 2816'sı eğitim için ayrılırken 287'si test için ayrılmıştır. Çatlak sınıfına ait görüntü sayısı 1587'dir. 1587 adet görselin 1411 tanesi eğitim, 176 tanesi test için ayrılmıştır. Çatlak sınıfına ait toplam etiket sayısı 7935 iken, bu sayının 7213'ü eğitim 722'si test için ayrılmıştır. 3 ayrı nesne için 3 ayrı evrişimli sinir ağı (Convolutional Neural Network, CNN) ağı eğitilmiş ve 3 ayrı model oluşturulmuştur. Sınıflandırma işleminde kullanılan görsellerin haricinde toplam kullanılan görüntü sayısı 2480 iken, toplam erişilen etiket sayısı 12044'tür.

Oluşturulan modeller eğitim işlemi sonrası başarı metriklerini elde edebilmek için var/yok şeklinde sınıflandırmaya tabi tutulmuştur. Bu aşamada test kümesinde 120'şer adet olmak üzere çatlak, yatay desen ve budak modeli için toplamda 360 adetlik veri seti oluşturulmuştur. Bu veri seti üzerinde modeller test edilerek başarı metrikleri sonuçları elde edilmiştir.

Sonraki aşamada nihai sınıflandırmada kullanılan veri setinde toplam sayı 132 adettir. Bu görseller kullanılarak nihai sınıflandırmada modellerin başarısı ölçümlenmiştir. Bu bilgiler ışığında maske bölge tabanlı evrişimli sinir ağında (Mask Region-Convolutional Neural, Mask R-CNN) 3 ayrı modelin eğitimi gerçekleştirilmiştir. İlerleyen bölümlerde eğitim ve testte kullanılan görüntülere ait detaylı açıklamalara yer verilmiştir.

2.3.2. Lamel Veri Etiketleme

CNN ağına veriler üzerinde yer alan anlamlı parçaları tanıtma işlemini veri etiketleme olarak açıklayabiliriz. Denetimli makine öğrenmesi yöntemleri ya da derin öğrenme yöntemleri kullanılarak çalışılan projelerde veri etiketleme işlemi en kritik adımlardan biridir. Bu tip çalışmalar verinin doğru anlamlandırılmasıyla başlar ve eğitim aşamasında model anlamlı veriye göre öğrenmesini gerçekleştirir. Kullanılan veri setine ait yapıların tanınması tecrübe gerektirdiği için bu tez çalışmasında verinin anlamlandırılması aşaması, alanında uzman bir kişi yardımıyla gerçekleştirilmiştir. Bu tez çalışmasında Make Sense web tabanlı ücretsiz etiketleme aracı kullanılmıştır [15]. Etiketleme işlemine ait örnek bir görüntü Şekil 2.8'de yer almaktadır [15].



Şekil 2.8. Make Sense etiketleme aracı çalışma alanı örneği [15]

Bu çalışma platformunda Şekil 2.8'de sağ üst köşede görüldüğü üzere 4 ayrı etiketleme çeşidi vardır. Etiket işlemleri budak, çatlak ve yatay desen yapıları üzerinden ayrı ayrı gerçekleştirilmiştir. Bu nesneler net çizgilerle ayrılan özelliklere sahip olmadığından poligonal etiketleme kullanılmıştır. Poligonal etiketleme kıvrımları fazla olan yapılar için uygun bir etiketleme çeşididir. Yalnızca Bir Kez Bak (You Only Look Once, YOLO) gibi ağlar poligonal etiketlemeyi desteklemezler. Bu ağ çeşitlerinde

dikdörtgen etiketleme yöntemi kullanılır [16]. Bu etiketleme yönteminde kullanılan ağlar ağ maskesi oluşturmamakta ve nesnenin koordinatları üzerinden tespit işlemi yapmaktadır.

Bu tez çalışmasında lameller üzerindeki 3 adet yapı ele alınmıştır. İlerleyen bölümlerde ele alınan yapılarla alakalı görseller detaylarıyla birlikte verilmiş ve açıklanmıştır.

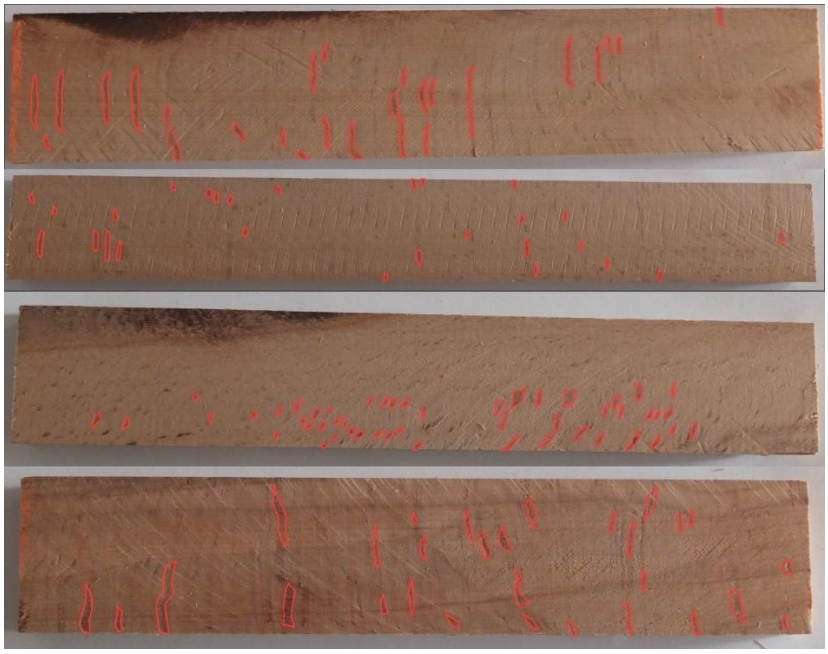
2.3.3. Budak Nesnesi

Şekil 2.9'da lamellerin sınıflandırılmasında etkili olan budak nesnesine yer verilmiştir. Budak yapısı lamel için kusurdur. Gerçek hayat sınıflandırmasında büyük boyutta budakların yüksek kaliteli sınıflarda yer alması kabul edilemez. Literatürde bununla alakalı birçok çalışma yer almaktadır. Kılıç vd.[17]'nin yaptığı çalışma 100 adet veri içermektedir. Bu görseller halka açık hale getirilmiştir. Bu çalışmada meşe lameller kullanılmıştır. Görüntüler üzerinde görüntü yumuşatma, gri tonlama, görüntü erozyonu, düğüm hatası adaylarını belirlemek için histogram eşikleme, düğüm hatası sınıflandırması ve son olarak da düğüm etiketleme kullanmışlardır. Yapılan çalışma sonucunda %81 başarı elde etmişlerdir.

Şekil 2.9. Lamel veri setinden rastgele alınan budak etiketi içeren görseller

2.3.4. Yatay Desen Nesnesi

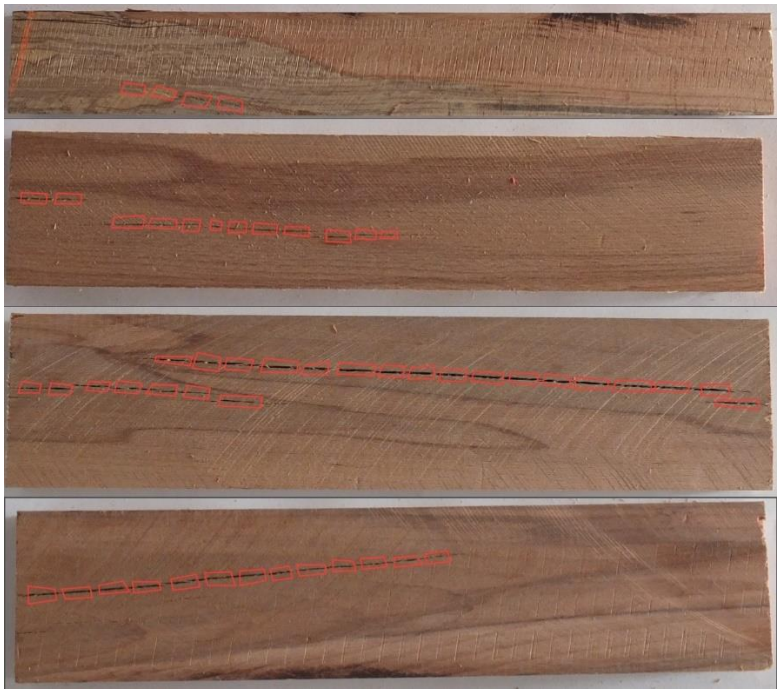
Şekil 2.10'da sınıflandırmayı etkileyen yapılardan yatay desen nesnesine yer verilmiştir. Oluşacak olan panel üzerinde en iyi kalite sınıfında yatay desen nesnesinin yer alması müşteriler ve üreticiler tarafından istenmez. Genelde yatay desen nesnesi bir lamel üzerinde hep çoklu yapıda tespit edilir. Bu sebepten bu nesne tespiti gerçekleştiğinde kalite bakımından üst sıralarda yer alan sınıfa koymak doğru değildir.



Şekil 2.10. Lamel veri setinden rastgele alınan yatay desen etiketi içeren görseller

2.3.5. Çatlak Nesnesi

Şekil 2.11'de yer alan nesne ise çatlak nesnesidir. Masif panel üretiminin ilk aşamasının kurutma ve buharlama adımında her ağaç cinsine göre farklı bir nem oranı olduğu belirtilmiştir. Ağaçtaki nem oranı bu aşamada doğru ayarlanmadığında ya da sonraki aşamalarda gerçekleşen yanlış uygulamalardan sonra lamel üzerinde Şekil 2.11'de olduğu gibi çatlaklar görülmektedir. Çatlak içeren yapının kalite sınıflarından en kötüsünde yer alması beklenir.

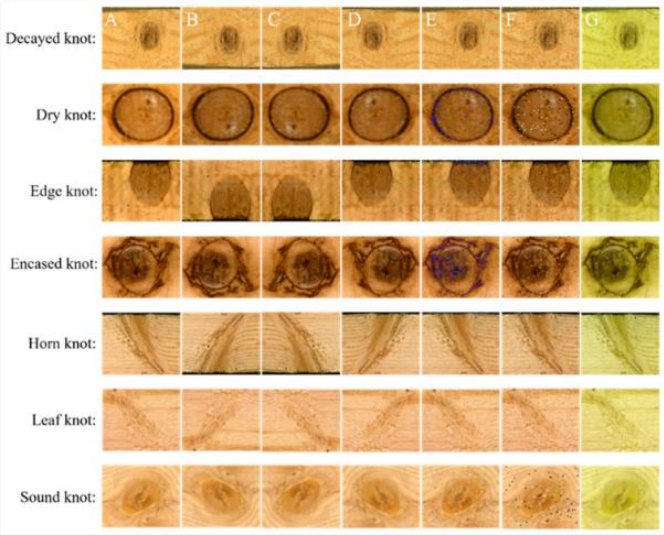


Şekil 2.11. Lamel veri setinden rastgele alınan çatlak etiketi içeren görseller

Bu tez çalışmasında lamel ve masif ahşap levhalar üzerindeki yapıları tanımlayan ve başarıya ulaşan literatürdeki tüm çalışmalarla alakalı detaylı bir inceleme gerçekleştirilmiştir. Lameller üzerindeki renk tonu benzerlikleri ayrıca kaliteyi etkileyen etmenler arasındadır. Çok koyu bir renk tonuyla çok açık renk tonu aynı panel üzerinde birleşmemelidir. Bu kapsamda denetimsiz öğrenme yöntemi olan K-Means kullanılarak panel parçası üzerinde renk sınıflandırma işlemi yapılmıştır. 200 adet görsel içeren veri seti Avrupa kayınına ait panellerdir [18].

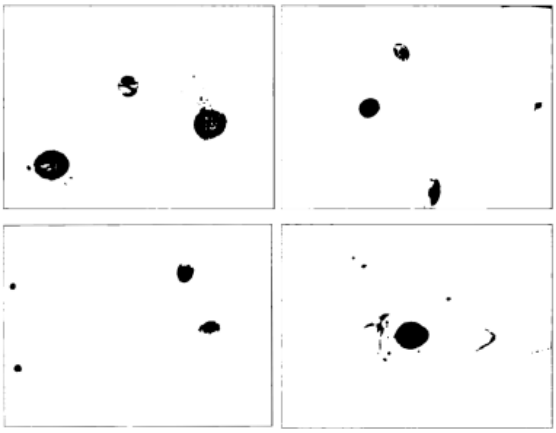
Şekil 2.12. Renk sınıflandırma işleminden sonra panellerin görüntüsü [18]

Kayın solid paneller üzerinde renk tonları sınıflandırılırken renk histogramı elde edilerek ilerlenmiştir. Bu çalışmada etiketleme maliyeti fazla olduğundan denetimsiz öğrenme yöntemi kullanılmıştır. K değerini belirlemek için ise kontur katsayısı yöntemini tercih etmişlerdir. Çeşitli k değerlerini ele alarak (k=7, k=8, k=9) sınıflandırma işlemini gerçekleştirmişler ve yüksek oranda başarı elde etmişlerdir [18]. Zhuang vd. [19]'nin yaptığı araştırmada 108 adet görüntü XGBoost yöntemi kullanılarak %97,22 başarı oranını elde etmiştir. Bu çalışmada VGG16, DenseNet121, XGBoost derin öğrenme yöntemleri birleştirilerek kullanılmıştır. Solid ahşap panel yüzeyleri üzerinde renk sınıflandırılmasına gidilmiştir. Model eğitim süresi 5,27 ve her bir resim için ortalama test süresi 51 milisaniye olarak ölçülmüştür. Zhengguang vd. [18]'nin yaptığı araştırmanın aksine Zhuang vd. [19]'nin yaptığı bu çalışmada açık, orta ve koyu şeklinde sınıflandırmaya gidilmiştir. Solid panellerin renklerinin sınıflandırılması üzerinde yapılan bu iki çalışmada da yüksek başarı elde edilmiştir.
Çam, göknar ve dişbudak ağacı kullanılarak Aşırı Öğrenme Makineleri sınıflandırma yöntemi %96,72 doğruluk sağlamıştır. Ele alınan yapılar ölü budak, canlı budak ve çürümedir [20]. Ladin ağaç cinsinden oluşan 448 görsel içeren veri seti üzerinde ResNet34'ü transfer öğrenme ile birleştiren TL-ResNet34 yöntemi önerilmiştir. Bu çalışmada budak tespiti yapılmıştır. Budak alt sınıfları; çürümüş budak, ölü budak, kenar budak, dağılmamış budak, boynuz budak, düğüm budak, sabit budaktır. Elde edilen başarı oranı %98'dir [21].



Şekil 2.13. Budak sınıfının alt dallarına ayrılarak tespit edilmesi işlemi [21]

Denetimli öğrenme yöntemlerinden YOLO algoritması kullanılarak hasarsız yüzey solucan deliği, budak ve çatlak sınıflarında nesne tespit işlemi gerçekleştirilmiş ve %80 Ortalama Hassasiyet Ortalaması (Mean Average Precision, mAP) değerine ulaşılmıştır [22]. 1800 adet görsel içeren veri setinde kullanılan ağaç cinsleri meşe, sarıçam ve huş ağacıdır [22]. Çatlak eğitim sınıfına ait ayrıca ulaşılan mAP değeri ise %68,75'tir [22]. Diğer bir çalışmada ise çin göknar ve sarıçamından oluşan 5000 adet veri setine ait ResNet yöntemi kullanılarak elde edilen mAP değeri %89,7 olarak ölçülmüştür. Bu çalışmada ele alınan yapılar dağılır budak, sabit budak, çatlak, küflenme, çürüme ve iğne deliğidir [23]. 500 adet görüntü üzerinde yapılan başka bir çalışmada ise Tek Atış Dedektörü (Single Shot Detector, SSD) yönteminin kullanılması önerilmiştir. Dağılır budak, sabit budak ve kontrol sınıflarını içeren bu çalışma %96,1 oranında başarıya ulaşmıştır [24]. Nesne tespiti işlemlerinde ön işlem aşamasında kullanılan Hilbert dönüşümü ve Gabor filtresi yöntemleri sayesinde doğruluk oranının artması hedeflenmiştir [25]. Yang vd. [26]'nin yaptığı çalışma budak büyüklüğünü tespit etmede Matlab araçlarının yeterli olduğunu göstermektedir.



Şekil 2.14. Budak büyüklüklerinin belirlenmesi [26]

Ahşap mobilya panellerinde farklı birden fazla farklı filtre kullanılarak budak nesnesinin tespiti gerçekleştirilmiş en iyi sonuç olarak %97,89 oranı belirlenmiştir [27]. Ayrıca tomruklar üzerinde yapılan çalışmada Mask R-CNN'in detaylı segmentasyonu desteklemediği öne sürülmüştür [28].

BÖLÜM 3

EVRİŞİMLİ SİNİR AĞLARI VE MASK R-CNN

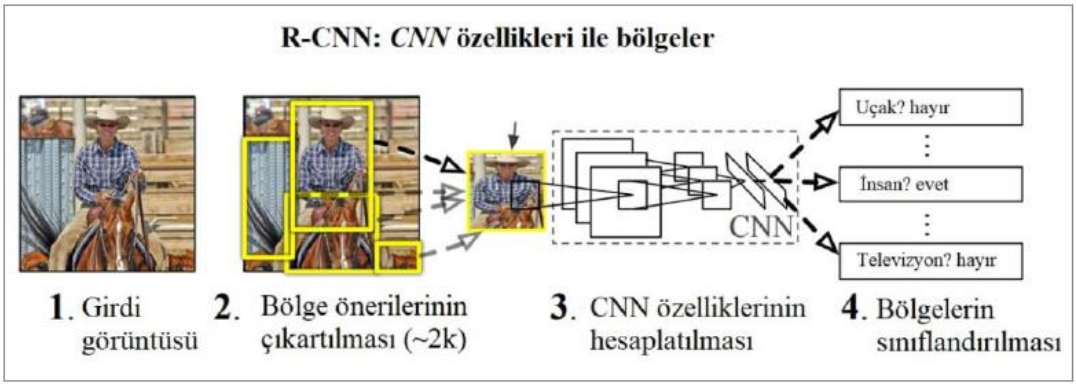
CNN'ler görüntü işleme üzerine kullanılan sınıflandırma ve segmentasyon işlemleri yapabilen en güçlü ve en verimli yapıdır [29]. CNN girdi olarak görüntü alır ve çeşitli nesneleri birbirinden ayırabilmeyi hedefler. Bu tez çalışması kapsamında Make Sense web tabanlı etiketleme aracı yardımıyla görüntüler üzerindeki nesnelerin işaretleme işlemi gerçekleştirilmiştir. Budak, yatay desen ve çatlak nesnelerinin tespit edilme aşamasında ise Yapay Sinir Ağları (YSA) kullanılmıştır. YSA, insan beyninde yer alan özelliklerden biri olan öğrenmenin taklit edilerek yeni bilgiler oluşturabilme, keşfedebilme ve türetebilme gibi yetenekleri otomatik bir şekilde uygulayabilmek kapsamında geliştirilen sistemlerdir [30,31]. Bu tez çalışmasında oluşturulan modellerin alt yapısı Mask R-CNN yapay sinir ağı mimarisi kullanılarak oluşturulmuştur. Mask R-CNN ağının tarih içerisindeki ilerleme aşamaları aşağıdaki bölümlerde detaylı olarak verilmiştir.

3.1. EVRİŞİMLİ SİNİR AĞI

Derin öğrenme mantığını kavrayabilmek için CNN'lerin alt yapısını öncelikle ele almak gerekir. Bir evrişimli sinir ağı girdi olarak görüntü alır, bu görüntüler üzerindeki nesneleri ağırlık ile anlamlandırır ve eğitim sonrasında görüntü üzerindeki nesnelerin birbirlerinden ayırt edilebilmesini sağlar, ek olarak nesnelerin birbiriyle ilişkilerini ortaya koyan bir derin öğrenme algoritmasıdır [32]. CNN'ler temelde görüntü sınıflandırma, benzerlik kümeleme ve görüntüler üzerinde nesne tespiti işlemleri için kullanılırlar. Yapısı gereği girdi olarak resim ya da video alırlar. Girdilerin matris formatında ağa sunulması gerekmekte ve gerekli filtreler uygulanarak görüntü üzerinden verilerin elde edilmesi hedeflenmektedir [33]. CNN'ler görüntü üzerindeki özellikleri çıkartır ve ilgili özellikler yok olmadan alt boyuta dönüştürme işlemini gerçekleştirir.

3.1.1. Bölge Tabanlı Evrişimli Sinir Ağı

Bölge tabanlı evrişimli sinir ağı (Region Based Convolutional Neural Network, R-CNN), bölge bazlı nesne tanıma işleminin gerçekleştirilmesinde kullanılmaktadır. Bu ağ çeşidi yüksek derecede nesne algılama doğruluğu vermektedir. Bir görüntü CNN ağına girdi olarak verildiğinde çok fazla sayıda sınırlayıcı kutucu oluşur. Bu sınırlayıcı kutucukların bazıları doğru bazıları yanlıştır. R-CNN doğru sınırlayıcı kutucukları bulabilmeyi hedefler [33]. R-CNN'ler tespit için örnek bir görüntü alır, görüntü üzerindeki tüm nesnelerde sınırlayıcı kutucuklar çizerek işleme başlar.



Şekil 3.1. R-CNN ağı çalışma mimarisi örneği [34]

Şekil 3.1'de R-CNN ağının çalışma diyagramında seçici arama kullanılmaktadır. Ağ aldığı görüntü üzerinde seçici arama işlemi uygular, görüntü yaklaşık 2000 bölge teklifine bölünür. Sonrasında bölünen 2000 adet bölgenin tamamı CNN ağı mimarisine gönderilir. CNN ağı gönderilen bölgelerdeki özellikleri çıkarır. Seçici arama işlemi yardımıyla belirlenmiş olan bu bölge CNN ağının çıkardığı özellikler ile birlikte Destek Vektör Makineleri (DVM)'ne girdi halinde verilir. DVM nesneleri ve gelen özelliklerini karşılaştırır, yaptığı hesaplamalar ile sonuca ulaşır. Bu süreçler görüntü üzerinde gerçekleşirken seçici arama ile belirlenen 2000 adet bölgenin devamlı bir şekilde CNN ağıyla işleme sokulması eğitim ve tahmin süresini oldukça uzatmaktadır. Bu sebepten R-CNN ağları gerçek zamanlı olarak yapılacak nesne tespit işlemleri için önerilmezler [35]. Çok fazla disk alanı gerektirmesi diğer bir dezavantajıdır. R-CNN mimarisine ait başarı değerlerine PASCAL VOC 2005-2012 [36] ve ILSVRC2013

[37] yarışmalarında ortaya çıkan sonuçlar örnek gösterilebilir. R-CNN'in yarışmalarda elde edilen sonuçları dezavantajlarıyla birlikte ele alındığında performanslı fakat oldukça yavaş çalıştığı gözlemlenir. R-CNN mimarisindeki bu problem Hızlı Bölge Tabanlı Evrişimli Sinir Ağının (Fast Region Based Convolutional Neural Network, Fast R-CNN) oluşturulmasına sebep olmuştur [38, 39]

3.1.2. Hızlı Bölge Tabanlı Evrişimli Sinir Ağı

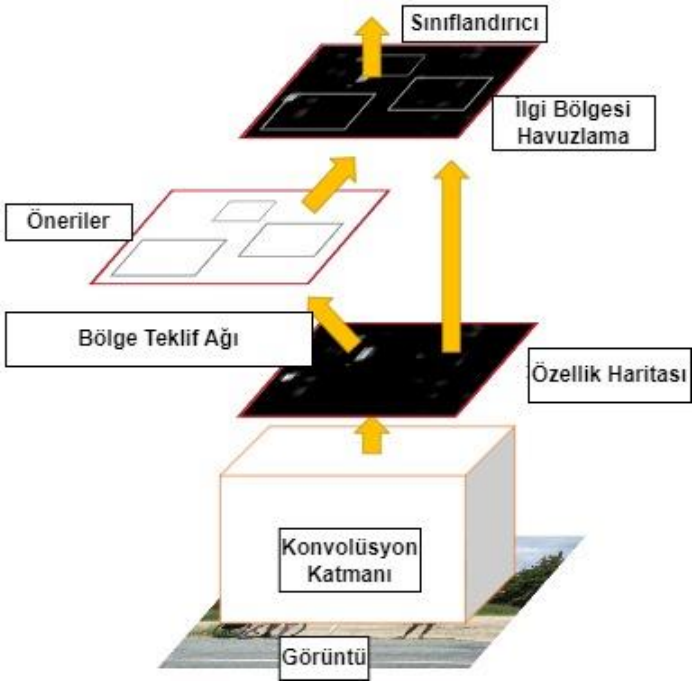
R-CNN mimarisinde görüntü 2000 bölge önerisine bölünmektedir. Bu sebepten R-CNN ağını eğitmek uzun zaman sürer ve maliyeti daha yüksektir. R-CNN mimarisinden kaynaklanan bu problemi çözmek için Fast R-CNN önerilmiştir [38]. Fast R-CNN tüm görüntüyü ve bölge önerilerini CNN mimarisine girdi olarak tek bir seferde alır. R-CNN ağından olduğu gibi öncesinde görüntüyü bölge önerilerine ayırmaz.

R-CNN mimarisinin aksine Fast R-CNN görüntüyü tek bir seferde CNN mimarisine almaktadır [40]. Görüntüyü CNN ağına gönderdikten sonra ağın oluşturduğu özellik haritası sayesinde bölge önerme işlemi gerçekleşmektedir. Her bölge önerisi için gereken öznitelikler bir araya getirilir. Ardından maksimum havuzlama işlemi ile belirli boyutlara indirgenir. Bu havuzlama katmanına ise İlgi Bölgesi (Region of Interest, RoI) adı verilir. Sonrasında öznitelik belirlemesi yapılan haritalar tek boyutlu vektör haline dönüştürülür. Oluşturulan vektörler sinir ağında işleme tabi tutulur. Softmax fonksiyonu ile o bölgede bulunan nesneye ait sınıf, sınırlayıcı kutucuk regresörü ile nesnenin sınırlayıcı kutusu belirlenir [40].

Fast R-CNN'in R-CNN'den diğer bir farkı ise R-CNN'in kullanmış olduğu DVM yerine softmax sınıflandırma yöntemini kullanmasıdır. Bütün bu anlatılan işlemler boyunca CNN yalnızca bir sefer kullanıldığından eğitim ve tespit işlemi için ayrılan süre büyük ölçüde azalmıştır. Ayrıca R-CNN ile karşılaştırıldığında mAP değerini iyileştirir [39].

3.1.3. Daha Hızlı Bölge Tabanlı Evrişimli Sinir Ağı

Daha Hızlı Bölge Tabanlı Evrişimli Sinir Ağı (Faster Region Based Convolutional Neural Network, Faster R-CNN) ağı R-CNN mimarilerinin üçüncüsüdür. Fast R-CNN ağının eksiğini iyileştirebilmek adına bu mimari tasarlanmıştır.



Şekil 3.2. Faster R-CNN örnek akış mekanizması [41]

Şekil 3.2'de Faster R-CNN'in örnek bir akış mekanizmasına yer verilmiştir. Burada görüldüğü gibi Fast R-CNN temelde iki ana kısımdan meydana gelmektedir. Bunlardan ilki Bölge Teklif Ağı (Region Proposal Network, RPN) ikincisi ise İlgi Bölgesi Havuzlama (Region of Interest Pooling, RoI Pooling) katmanlarıdır. Şekil 3.2'de verilen mimariye göre, ağa verilen örnek bir görüntünün RPN ve RoI Pooling katmanlarından geçirilmesi gösterilmektedir. Faster R-CNN'in döndürdüğü sonuçlar arasında sınırlayıcı kutucuklar, etiketler ve tespit edilen nesnelerin olasılık değerleri yer almaktadır [42].

Fast R-CNN'de ilgilenilen bölgeleri seçmek için herhangi bir yöntem olmamasına rağmen Faster R-CNN bölge kümelerini oluşturmak için RPN yöntemini kullanır. İki mimari arasındaki temel fark budur [43]. Burada bölge öneri yönteminden kaynaklanan fark ile zamandan kazanç sağlanır ve tahmin süresi Çizelge 3.1'de de görüldüğü üzere yaklaşık 0,2 saniyeye düşürülür.

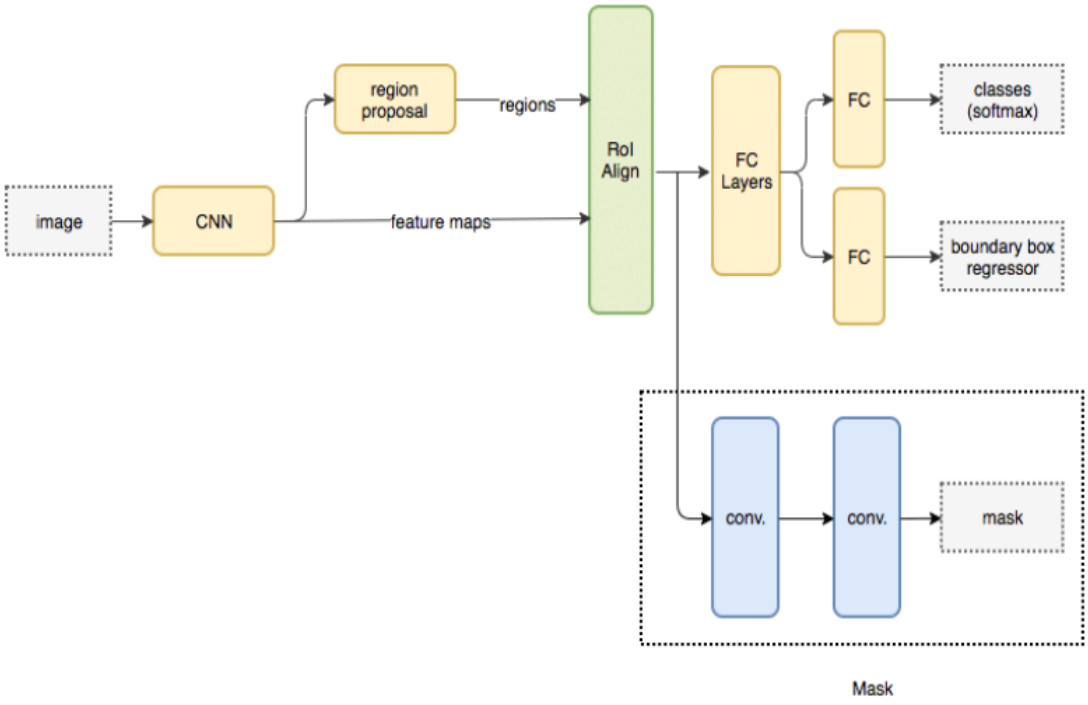
Çizelge 3.1. Anlatılan mimarilerin bazı özelliklerinin kıyaslanması [44]

	R-CNN	**Fast R-CNN**	**Faster R-CNN**
Bölge önerileri yöntemi	Seçici arama	Seçici arama	RPN
Tahmin süresi	40-50 saniye	2 saniye	0,2 saniye
Hesaplama	Yüksek hesaplama süresi	Yüksek hesaplama süresi	Düşük hesaplama süresi
Pascal VOC 2007 test veri setindeki mAP (%)	58,5	66,9	69,9
Pascal VOC 2012 test veri setindeki mAP (%)	53,5	65,7	67,0

R-CNN mimarisinin eksiklerinin üzerine eklemeler yapılarak geliştirilen ve Faster R-CNN'e kadar uzanan bu ilerlemelerin sonucunda Faster R-CNN mimarisi nesne tespitinin temel yapı taşı haline dönüşmüştür. Sonraki çalışmalarda bu mimarinin çeşitli eksikleri geliştirilmiştir [43]. Geliştirilen özelliklerin yanı sıra nesneleri segmente edebilen ve yüksek kaliteli maskeler oluşturabilen Mask R-CNN mimarisi oluşturulmuştur.

3.1.4. Mask R-CNN

Nesne tespit işlemlerinin yanı sıra segmentasyon problemine çözüm getirmeyi hedefleyen derin bir sinir ağıdır. Bu ağ mimarisi bir görüntü ve videodaki nesneleri ayırabilmektedir. Mask R-CNN sınıflandırma ve sınırlayıcı kutucuk işlemlerinin yanı sıra Faster R-CNN'deki RoI aşaması sırasında segmentasyon maskelerini tahmin etme görevini üstlenir [45].



Şekil 3.3. Mask R-CNN örnek mimari [46]

Mask R-CNN iki aşamadan oluşmaktadır. İlk adımda görüntüyü tarar ve bir nesneyi içerme olasılığı yüksek önerilerde bulunur. İkinci adımda ise bu önerileri sınıflandırır ve sınırlayıcı kutucuklarla birlikte maskeleme işlemini de gerçekleştirir. Faster R-CNN'den farklı olarak bulunan maske dalı küçük bir hesaplama yükünü gerçekleştirdiği işlemlere dahil ederek hızlı bir deney gerçekleştirir [45].

Mask R-CNN'in temelini oluşturan her iki aşamada omurga yapısına bağlıdır. Bu omurga özellik belirleyici olarak görev yapan ortalama bir evrişimli sinir ağıdır. İlk katmanlar basit seviyedeki özellikleri sonraki katmanlar ise yüksek seviyeli özellikleri algılamaktadır. Şekil 3.3'te de görülen İlgi Bölgesi Hizalama (Region of Interest Align, RoI Align) Faster R-CNN'de yer almamaktadır. Mask R-CNN bu adımda nesnelerin hizalanmasını sağlar. Omurga ağından geçen görüntü özellik haritasına dönüşür. Bu özellik haritası bu aşamadan sonra gerçekleşecek olan işlemlere girdi olarak sunulur. Mask R-CNN ağı segmentasyon işlemini gerçekleştirmek için evrişimli sinir ağlarının bir uzantısını içermektedir. Mask R-CNN omurgasında MS COCO ağırlıklarıyla işlem yapmaktadır [47]. ResNet-101'den özellikler çıkarıldıktan

sonra Özellik Piramit Ağı (Feature Pyramid Network, FPN) oluşturulur ve bu haritada ankorların tanımlaması yapılır. FPN ilk piramitten üst düzey özellikleri alır ve bunları alt katmanlara geçirerek standart özellik çıkartma piramidini oluşturur.

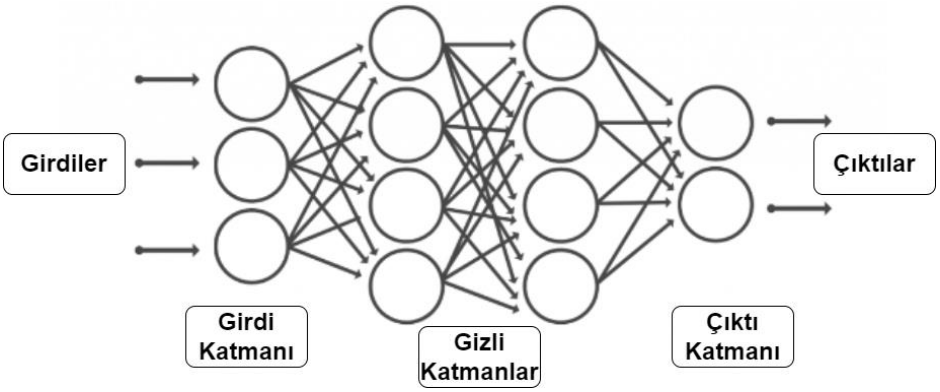
Ayrıca Mask R-CNN maskeleme işlemi esnasında örnek bölütleme özelliğini kullanmaktadır. Bu işlemde eğer görüntü üzerinde aynı türden birden fazla nesne varsa bu nesneler farklı renk kombinasyonları ile birbirinden ayrılır. Algoritma kayıpları L sınıflandırma kayıpları $Lcls$, maskeleme kayıpları $Lmask$ ve çerçeve kayıpları $Lbox$ olmak üzere 3 ayrı gruptan oluşmaktadır. Eşitllik (3.1)'de gösterilmiştir [48].

$$L = Lcls + Lbox + Lmask \tag{3.1}$$

Bu tez çalışmasında Mask R-CNN mimarisinin omurga ağında ResNet-101 tercih edilmiştir.

3.2. DERİN ÖĞRENME KÜTÜPHANELERİ

Derin öğrenme (DÖ) verilen bir set üzerinde tahmin etme işlemi yapan ve birden fazla katmandan oluşan makine öğreniminin (MÖ) bir alt bölümüdür. Derin öğrenme makine öğrenmesinin; makine öğrenmesi ise yapay zekanın alt dalı olarak açıklanmaktadır. Derin öğrenme çok sayıda katmandan oluşmaktadır ve bu art arda gelen katmanlar önceki katmandan gelen çıktıyı girdi olarak almaktadır [49,50]. Bir derin öğrenme çalışması verileri eğitme ve tecrübelerden öğrenme aşamalarını içerir.



Şekil 3.4. Örnek derin öğrenme ağı çalışma mimarisi [51]

Şekil 3.4'te sergilenen derin öğrenme ağında işlemler ilk katman olan girdi katmanının, girdi verilerini alarak bunu ilk gizli katmana iletmesiyle başlamaktadır. Nöronlar arasındaki her bağlantı ağırlıklardan meydana gelir, bu değerlerin önemini belirtir. Gizli katmanlar alınan veriler üzerinde hesaplama işlemleri gerçekleştirmektedir. Son olarak çıktı katmanı gerekli sonucu elde ederek akışı tamamlamaktadır. Çıkışları standart hale getirmek için ise aktivasyon fonksiyonları tercih edilir. Derin öğrenme ağlarında başlangıçtaki en önemli nokta ağı eğitmek için büyük bir veri seti oluşturmaktır [51].

Bu tez çalışmasında kullanılan sinir ağının eğitilmesi için i9 10980XE işlemcili ve NVIDIA Quadro RTX 5000 ekran kartına sahip bilgisayar tercih edilmiştir. Proje kodlanırken Python dili ve TensorFlow ile Keras gibi kütüphaneler de kullanılmıştır.

3.2.1. Keras Kütüphanesi

Keras, Python dili kullanılarak yazılmış her türlü derin öğrenme modelini tanımlayan üst düzey sinir ağları Uygulama Programlama Arayüzü (Application Programming Interface, API)'dür [51,52]. Keras'ın geliştirilmesinin altında yatan temel fikir hızlı bir ilk örnek oluşturarak deneyleri kolaylaştırmaktır. Ortaya atılan bir fikirden mümkün olduğunca en az gecikme ile sonuca ulaşma yeteneği vardır [51]. Keras'ın sağladığı bu fayda hem bilim insanları hem de geliştiriciler için büyük bir avantajdır. Keras temel olarak derin öğrenme projelerini az çabayla test etmeye yardımcı olmaktadır.

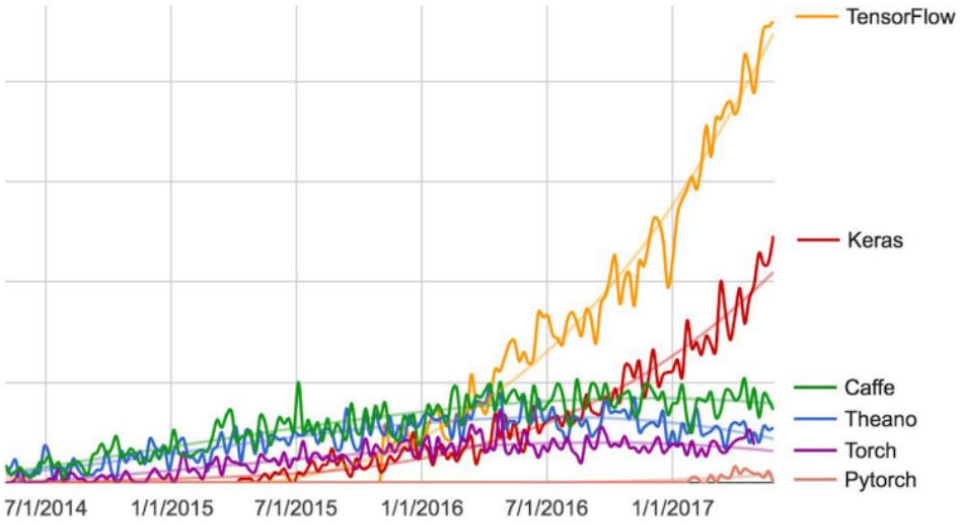
Keras ile hızlı bir şekilde birkaç satırdan oluşan basit bir ağ modeli ortaya çıkarılabilir. Keras'ı kullanan birçok yapay zeka topluluğu vardır bu sayede ağ oluşturulurken ya da sonrasında alınan hatalarda kolayca çözüm bulabilme imkanı sağlamaktadır. Keras'a TensorFlow, Theano [53], Microsoft Cognitive Toolkit [54], PlaidML [55], gibi birçok arka uç bağlanabilir. Her arka ucun da kendine özgü bir avantajı vardır. Birçok sayıda platformu destekler. Keras'ta tek bir Grafik İşlem Birimi (Graphics Proccesing Unit ,GPU) ya da birden fazla GPU kullanılabilir.

Keras'ın birçok avantajı olsa bile üst düzey bir API kullanıldığının unutulmaması gerekir. Bu sebepten Keras ile düşük seviyeli işlemler gerçekleştirilmez [56]. Ayrıca arka uçta gerçekleşen her ayrıntıyı bilmeyi zorunluluğu kılmadığı için Keras'ı anlamak kolaydır. Karışık olmayan derin öğrenme modeli kullanılacak ise Keras tercih edilmelidir. Kullanıcı dostu ve uygulaması kolay bir Python kütüphanesidir. Sağladığı bu özellikler nedeniyle bu çalışmada Keras kullanımı tercih edilmiştir.

3.2.2. TensorFlow Kütüphanesi

TensorFlow derin öğrenmeye dayalı açık kaynak kodlu bir kütüphanedir. Başlangıçta derin öğrenme yerine büyük hesaplama işlemleri için oluşturulmuştur. Kısa süre sonra TensorFlow'un derin öğrenme uygulamalarına olan yararı keşfedilmiş ve sinir ağlarına dayalı özel bir sistem olarak açık kaynaklı bir çözüm haline getirilmiştir [57].

TensorFlow çoklu makine öğrenimi, derin öğrenme modellerini bir araya getirmektedir. Python dilini makine öğrenmesi alanında kullanmaya imkan verir, bunun için bir ön yüz API sunmaktadır [58]. Python haricinde C++ ile de kullanım olanağı yer almaktadır. Temel olarak matematikte yapılması gereken karmaşık işlemleri uygulamak için düşük seviyeli bir araç takımıdır. Deneysel mimarileri öğrenmek, değiştirmek ve bu mimarileri çalışan bir yazılıma dönüştürmeyi isteyen araştırmacıları hedefler. Çeşitli MÖ uygulamaları içerisinde sinir ağları eğitilebilir [59]. Kullanımı kolay olduğundan giriş düzeyindeki MÖ bilgisine sahip kişilerin tüm yapay zeka modellerini sıfırdan oluşturmak yerine güçlü bir kütüphane desteği vermesi sebebiyle popüler hale gelmiştir. Şekil 3.5'te görüldüğü üzere farklı kütüphanelerin TensorFlow'a kıyasla daha az tercih edildiği görülmektedir.



Şekil 3.5. Farklı derin öğrenme kütüphanelerinin popülerliği [56]

BÖLÜM 4

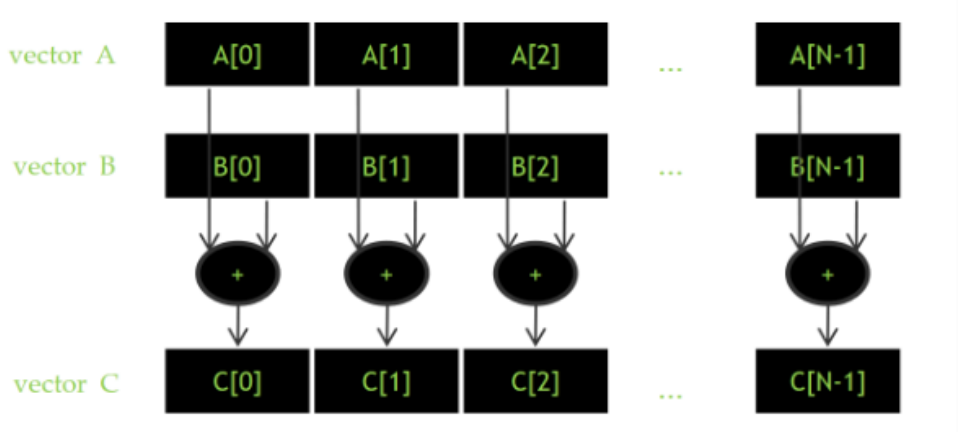
MASİF PANEL ÜRETİMİNDE KULLANILAN LAMEL PARÇALARI ÜZERİNDE NESNE TESPİTİ VE SINIFLANDIRILMASI

Bu tez çalışmasında modelin eğitimi esnasında Mask R-CNN kullanılacağından önceki bölümlerde detaylı olarak bahsedilmiştir. Bu bölümde derin öğrenme ortamının ayarlanması, sanal ortam kurulumu, Mask R-CNN mimarisi için gerekli konfigürasyon ayarlarına ait parametreler yer almaktadır. Derin öğrenme işlemi gerçekleştiren bir ağ her yanlış tahminde elindeki ağırlık değerlerini değiştirerek öğrenme işlemi için tutarlı katsayıları elde etmeye çalışır. Bir taraftan da her bir veri için yeniden kontrol gerçekleştirip gelen bilgiye göre bazı değerleri azaltırken bazı değerleri artırır. Günümüzde büyük veriler işlenip doğru ve tutarlı modeller kurmak mümkün hale gelmiştir fakat bu işlem için güçlü bir makineye ihtiyaç duyulmaktadır. Aksi takdirde bilgisayar başında çok fazla beklemek gerekir. Bu aşamada bekleme süresini kısaltan donanım GPU'dur. Her model eğitimi için GPU zorunlu ya da olması gereken bir donanım değildir. Özellikle projede küçük veri setleriyle uğraşılacaksa, görüntü işleme ya da büyük veri kullanılmayacaksa GPU kullanmak gözle görülür bir fayda sağlamayacaktır. Fakat görüntü sayısı ve görüntüler üzerindeki etiket sayıları arttıkça zamanı verimli kullanabilmek adına GPU kullanımı gereklidir.

Bu tez çalışmasında görüntü sayısı ve görüntüler üzerindeki etiketlerin fazla olması sebebiyle merkezi işlem birimi yerine GPU tercih edilerek zamanı verimli kullanabilmek hedeflenmiştir [60]. Aşağıdaki bölümlerde GPU kullanımı için gereksinimlerin kurulumu Keras ve TensorFlow kütüphanelerinin kurulan GPU ile uyum sağlayabilmesi adına hangi versiyonlarının uygun olabileceği ile alakalı detaylara yer verilmiştir.

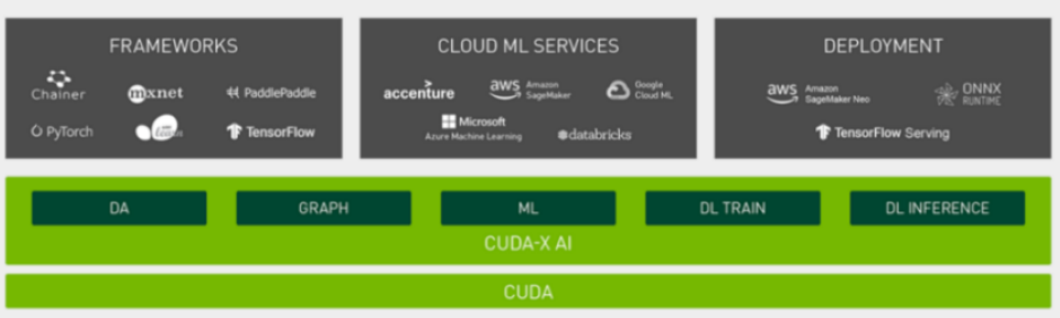
4.1. MODEL EĞİTİMİ ESNASINDA GPU KULLANIMI İÇİN GEREKENLER

Önceki bölümlerde derin öğrenme mimarisini oluştururken kullanılacak olan derin öğrenme kütüphaneleri TensorFlow ve Keras olarak saptanmıştır. GPU zamanı verimli kullanabilmek adına bu projede tercih edilmiştir [60]. GPU kullanımı için Birleşik Aygıt Mimarisi Hesaplama (Compute Unified Device Architecture, CUDA) kurulumuna ihtiyaç vardır. CUDA, NVIDIA tarafından oluşturulan bilgisayarın işlem performansını artıran bir paralel programlama platformudur [61]. GPU'da gerçekleşen hesaplama işlemlerinde çekirdeklerin kullanılmasına olanak sağlar. C dilini kullanır ve basitçe kurulabilecek olan eklentilerle birlikte kodu rastgele erişimli belleğe işleyebilir.



Şekil 4.1. CUDA'da verinin paralelleştirilmesi işleminin bir örneği [62]

Şekil 4.1'de CUDA'nın veriler üzerinde paralel çalışmasına örnek verilmiştir. Şekilde yer alan vektörlerin toplanması işlemi gerçekleştirilecektir. CUDA bu kısımda her bir vektörün farklı elemanlarını tek tek toplayıp elde etmek yerine vektörlerin bütün elemanlarını aynı anda toplayarak paralel işlem sağlamaktadır [62]. CUDA mimarisinde C[0], B[0] ve A[0] elemanları toplanırken, C[1], B[1] ve A[1] elemanlarının da aynı zamanda toplanmasına engel teşkil edecek bir durum yer almamaktadır. Engelin aksine CUDA mimarisi bu aşamaları paralel bir şekilde gerçekleştirerek hızlanmayı sağlamaktadır.

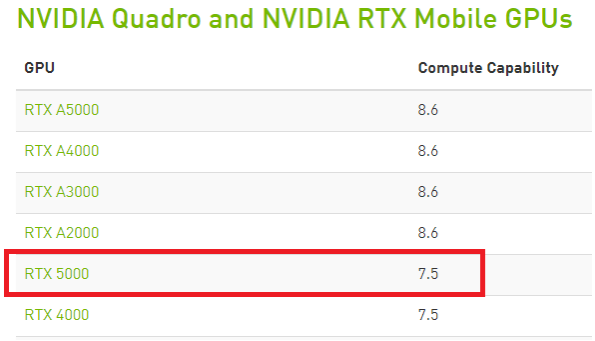


Şekil 4.2. NVIDIA CUDA servisleri [63]

Şekil 4.2'de NVIDIA CUDA'ya ait MÖ kütüphaneleri yer almaktadır. Burada yer alan kütüphaneler arasından TensorFlow kütüphanesi seçilmiştir. CUDA kurulumu yapılırken aynı zamanda CUDA Derin Sinir Ağları Kütüphanesi (CUDA Deep Neural Network Library, cuDNN) kurulumunun da yapılması gereklidir. CuDNN, CUDA ortamının hazırlanması için gerekli olan bir altyapıdır. CuDNN ileri ve geri evrişim, havuzlama, normalizasyon ve aktivasyon katmanları gibi işlemler için uygulamalar sunmaktadır.CUDA, herhangi bir CNN mimarisindeki işlemleri hızlandırmayı hedeflemektedir. Diğer temel hedefi ise sinir ağı çerçeveleri topluluğunun API'lerinden eşit olarak yararlanabilmek için gerekli alt yapıyı sağlamaktır [62]. Şekil 4.3'te cuDNN için hızlandırılmış çerçeveler yer almaktadır [64].

Şekil 4.3. CuDNN için hızlandırılmış çerçeveler [64]

Projede kullanılacak olan bilgisayar Windows 10 Pro işletim sistemine sahiptir. Bilgisayarda bulunan ekran kartı ise NVIDIA Quadro RTX 5000'dir. NVIDIA kendi sitesinde GPU destekli ekran kartlarına özel indirme linklerine yer vermiştir.



NVIDIA Quadro and NVIDIA RTX Mobile GPUs

GPU	Compute Capability
RTX A5000	8.6
RTX A4000	8.6
RTX A3000	8.6
RTX A2000	8.6
RTX 5000	7.5
RTX 4000	7.5

Şekil 4.4. NVIDIA Quadro RTX ekran kartının hesaplama kapasitesi [65]

NVIDIA Quadro RTX 5000 ekran kartının hesaplama kapasitesi NVIDIA'nın kendi sitesinde yer almaktadır [65]. Projede kullanılan bilgisayarda yer alan ekran kartının GPU destekli olduğu ve hesaplama kapasitesinin üstünlüğü Şekil 4.4'te görülmektedir. İşletim sistemi windows, mimari x86_64, versiyon 10, indirme tipi exe(network) olarak işaretlendiğinde uygun CUDA versiyonu 11.7 olarak indirilecektir. CUDA ile uyumlu cuDNN versiyon ise 8.5 olarak belirlenmiş ve gerekli indirmelerden sonra kurulumlar gerçekleştirilmiştir. CUDA ve cuDNN kurulumlarından sonra Anaconda [66] terminal ekranı kullanılarak TensorFlow ve Keras paketlerinin kurulumu sağlanmıştır. Anaconda üzerinde projelerde çalışılan kütüphane versiyonları farklılık gösterebilir, bu sık karşılaşılan bir durumdur. Bu sebepten Anaconda üzerinden Python kullanımı gerçekleştirileceği zaman her proje için ayrı bir sanal ortam kurulup sonrasında projenin gerektirdiği kütüphane versiyonları eklenmelidir. Bu önlem projeler arasında geçiş yaptıktan sonra kodların versiyon kaynaklı hata verme sebeplerini minimuma düşürecektir.

Aşağıda projede kullanılan Keras ve TensorFlow'un ilgili versiyonlarının yüklenmesini sağlayan kod satırlarına yer verilmiştir.

conda install tensorflow-gpu==1.14.0

conda install keras==2.2.5

Bu aşamada derin öğrenme ağı kurulacak bir sistemin gerekli alt yapısı inşa edilmiştir. Sonraki aşamada model eğitiminde kullanılacak olan Mask R-CNN sinir ağı için gerekli konfigürasyon parametreleri yer almaktadır.

4.2. MASK R-CNN KONFİGÜRASYON PARAMETRELERİ

Bölüm 4.1'de anlatılan aşamalar sayesinde gerekli GPU kurulumu tamamlanmıştır. Bu aşamada GPU'nun çalışma mimarisine dahil edilip edilmediğinin kontrolü gerçekleştirilmiştir [67]. Şekil 4.5'te kurulan GPU'nun sistem kullanımı için hazır olduğu görüntülenmektedir.

```
from tensorflow.python.client import device_lib

device_lib.list_local_devices()
```

```
[name: "/device:CPU:0"
 device_type: "CPU"
 memory_limit: 268435456
 locality {
 }
 incarnation: 3130643251426480278,
 name: "/device:GPU:0"
 device_type: "GPU"
 memory_limit: 15132721152
 locality {
   bus_id: 1
   links {
   }
 }
 incarnation: 7137459545454281573
 physical_device_desc: "device: 0, name: Quadro RTX 5000, pci bus id: 0000:17:00.0, compute capabilit
y: 7.5",
 name: "/device:GPU:1"
 device_type: "GPU"
 memory_limit: 15132721152
 locality {
   bus_id: 1
   links {
   }
 }
 incarnation: 13681897084781345669
 physical_device_desc: "device: 1, name: Quadro RTX 5000, pci bus id: 0000:65:00.0, compute capabilit
y: 7.5"]
```

Şekil 4.5. GPU'nun kullanıma hazır olup olmadığını kontrol eden kod bloğu

```
import os
import sys
import random
import math
import re
import time
import numpy as np
import cv2
import matplotlib
import matplotlib.pyplot as plt
import json
import os
import shutil
import zipfile

```

Şekil 4.6. Mask R-CNN kullanımı için gerekli kütüphanelerin listesi

Şekil 4.6'da Mask R-CNN ağının eğitim, tahmin ve görselleştirme aşamalarında kullanılacak olan Python kütüphanelerinin içe aktarılması işlemi bu adımda gerçekleştirilmiştir. Bu kütüphaneler arasında Matplotlib kütüphanesi de yer almaktadır [68]. Bu kütüphane sayesinde eğitim sonrasında test veri seti ile oluşturulan karmaşıklık matrisinin görselleştirilmesi işlemleri gerçekleştirilmektedir.

```
class CustomConfig(Config):
    def __init__(self, num_classes):


        classes_number = num_classes
        super().__init__()

    # eğitilecek modeldeki nesne ismi
    NAME = "budak"

    GPU_COUNT = 1
    IMAGES_PER_GPU = 1

    # Toplam sınıf sayısına +1 eklenir arka plan için
    NUM_CLASSES = 1+1

    IMAGE_MIN_DIM = 1024
    IMAGE_MAX_DIM = 1024

    STEPS_PER_EPOCH = 500

    VALIDATION_STEPS = 5

    DETECTION_MIN_CONFIDENCE = 0.7
```

Şekil 4.7. Model eğitiminde kullanılan gerekli parametre değerleri

Mask R-CNN model eğitimi için başlangıçta kullanılacak olan parametre değerleri Şekil 4.7'de gösterilmiştir. Model için gerekli omurga ağı görüntü özelliklerini ayıklayan ResNet-50 ya da ResNet-101 [69] olarak seçilebilir. Bu çalışmada omurga ağı ResNet-101 seçilmiştir. Mask R-CNN arka plan için eğitilecek olan sınıf sayısını YOLO'nun aksine bir arttırmaya ihtiyaç duyar [70]. Bu sebepten NUM_CLASSES parametresi 2 olarak belirlenmiştir.

Bu proje kapsamında 3 ayrı model oluşturulmuştur. Oluşturulan modeller budak, yatay desen ve çatlaktır. Oluşturulan modeller için DETECTION_MIN_CONFIDENCE, STEPS_PER_EPOCH, IMAGE_MIN_DIM, IMAGE_MAX_DIM, IOU_THRESHOLD, STEPS_PER_EPOCH parametrelerinin belirlenmesi için denemeler yapılmıştır. Her deneme esnasında farklı değerler verilerek doğru konfigürasyon ayarlarının yapılması sağlanmıştır. Örneğin, STEPS_PER_EPOCH

parametresine 100 atanarak model eğitildiğinde çıkan mAP değeri 500 atanarak eğitildiğinde çıkan mAP değerinden düşük olduğundan STEPS_PER_EPOCH değerine olması gereken atama 500 olarak belirlenmiştir.

```
Configurations:
BACKBONE                        resnet101
BACKBONE_STRIDES                [4, 8, 16, 32, 64]
BATCH_SIZE                      1
BBOX_STD_DEV                    [0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE          None
DETECTION_MAX_INSTANCES         100
DETECTION_MIN_CONFIDENCE        0.7
DETECTION_NMS_THRESHOLD         0.3
FPN_CLASSIF_FC_LAYERS_SIZE      1024
GPU_COUNT                       1
GRADIENT_CLIP_NORM              5.0
IMAGES_PER_GPU                  1
IMAGE_CHANNEL_COUNT             3
IMAGE_MAX_DIM                   1024
IMAGE_META_SIZE                 14
IMAGE_MIN_DIM                   1024
IMAGE_MIN_SCALE                 0
IMAGE_RESIZE_MODE               square
IMAGE_SHAPE                     [1024 1024    3]
LEARNING_MOMENTUM               0.9
LEARNING_RATE                   0.001
LOSS_WEIGHTS                    {'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0,
'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE                  14
MASK_SHAPE                      [28, 28]
MAX_GT_INSTANCES                100
MEAN_PIXEL                      [123.7 116.8 103.9]
MINI_MASK_SHAPE                 (56, 56)
NAME                            budak
NUM_CLASSES                     2
POOL_SIZE                       7
POST_NMS_ROIS_INFERENCE         1000
POST_NMS_ROIS_TRAINING          2000
PRE_NMS_LIMIT                   6000
ROI_POSITIVE_RATIO              0.33
RPN_ANCHOR_RATIOS               [0.5, 1, 2]
RPN_ANCHOR_SCALES               (32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE               1
RPN_BBOX_STD_DEV                [0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD               0.7
RPN_TRAIN_ANCHORS_PER_IMAGE     256
STEPS_PER_EPOCH                 500
TOP_DOWN_PYRAMID_SIZE           256
TRAIN_BN                        False
TRAIN_ROIS_PER_IMAGE            200
USE_MINI_MASK                   True
USE_RPN_ROIS                    True
VALIDATION_STEPS                5
WEIGHT_DECAY                    0.0001
```

Şekil 4.8. Budak modelinin eğitimi için ayarlanan parametre değerleri

Şekil 4.8'de görülen budak modelinin konfigürasyon parametrelerinden DETECTION_MIN_CONFIDENCE değerini ele alırsak, %70 ve üzerinde skor puanıyla tespit ettiği nesneleri sonuçlara dahil etmesi gerekmektedir. Bu model için IOU_THRESHOLD 0,3 olarak belirlenmiştir [71]. Bu şu anlama gelir; maskeleme işlemi sonrasında elde ettiği değer 0,3 ve üzerinde ise maskelenen bu nesneyi sonuçlarda dahil etmesi gerekmektedir. Şekil 4.9'da yatay desen modeli için belirlenen parametrelere, Şekil 4.10'da çatlak modeli için belirlenen parametrelere ayrıca yer verilmiştir.

```
Configurations:
BACKBONE                       resnet101
BACKBONE_STRIDES               [4, 8, 16, 32, 64]
BATCH_SIZE                     1
BBOX_STD_DEV                   [0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE         None
DETECTION_MAX_INSTANCES        100
DETECTION_MIN_CONFIDENCE       0.7
DETECTION_NMS_THRESHOLD        0.3
FPN_CLASSIF_FC_LAYERS_SIZE     1024
GPU_COUNT                      1
GRADIENT_CLIP_NORM             5.0
IMAGES_PER_GPU                 1
IMAGE_CHANNEL_COUNT            3
IMAGE_MAX_DIM                  1024
IMAGE_META_SIZE                14
IMAGE_MIN_DIM                  1024
IMAGE_MIN_SCALE                0
IMAGE_RESIZE_MODE              square
IMAGE_SHAPE                    [1024 1024    3]
LEARNING_MOMENTUM              0.9
LEARNING_RATE                  0.001
LOSS_WEIGHTS                   {'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0,
'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE                 14
MASK_SHAPE                     [28, 28]
MAX_GT_INSTANCES               100
MEAN_PIXEL                     [123.7 116.8 103.9]
MINI_MASK_SHAPE                (56, 56)
NAME                           yatay_desen
NUM_CLASSES                    2
POOL_SIZE                      7
POST_NMS_ROIS_INFERENCE        1000
POST_NMS_ROIS_TRAINING         2000
PRE_NMS_LIMIT                  6000
ROI_POSITIVE_RATIO             0.33
RPN_ANCHOR_RATIOS              [0.5, 1, 2]
RPN_ANCHOR_SCALES              (32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE              1
RPN_BBOX_STD_DEV               [0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD              0.7
RPN_TRAIN_ANCHORS_PER_IMAGE    256
STEPS_PER_EPOCH                500
TOP_DOWN_PYRAMID_SIZE          256
TRAIN_BN                       False
TRAIN_ROIS_PER_IMAGE           200
USE_MINI_MASK                  True
USE_RPN_ROIS                   True
VALIDATION_STEPS               5
WEIGHT_DECAY                   0.0001
```

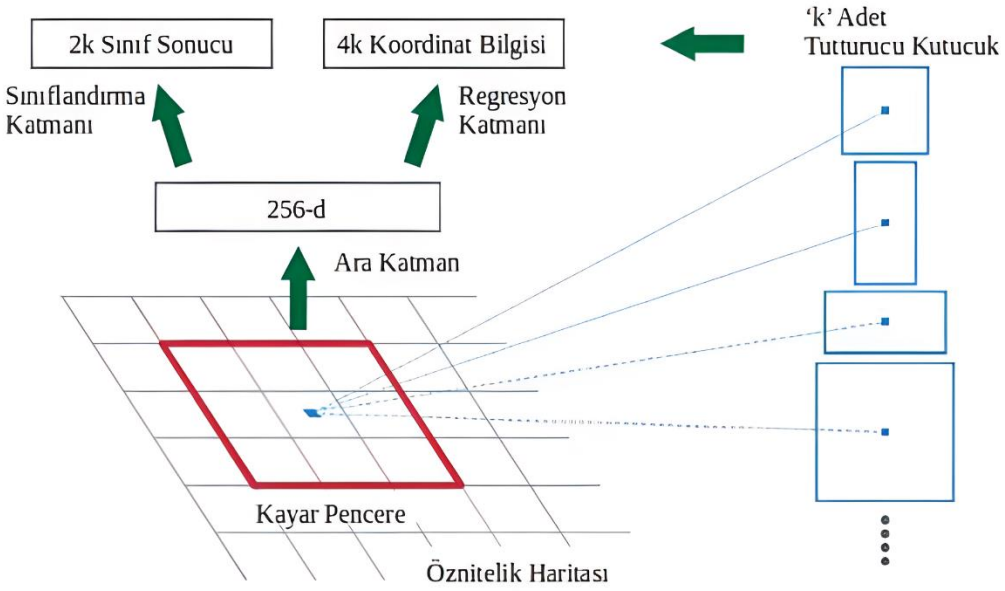
Şekil 4.9. Yatay desen modelinin eğitimi sırasında kullanılan parametreler

```
Configurations:
BACKBONE                       resnet101
BACKBONE_STRIDES               [4, 8, 16, 32, 64]
BATCH_SIZE                     1
BBOX_STD_DEV                   [0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE         None
DETECTION_MAX_INSTANCES        100
DETECTION_MIN_CONFIDENCE       0.7
DETECTION_NMS_THRESHOLD        0.3
FPN_CLASSIF_FC_LAYERS_SIZE     1024
GPU_COUNT                      1
GRADIENT_CLIP_NORM             5.0
IMAGES_PER_GPU                 1
IMAGE_CHANNEL_COUNT            3
IMAGE_MAX_DIM                  1024
IMAGE_META_SIZE                14
IMAGE_MIN_DIM                  1024
IMAGE_MIN_SCALE                0
IMAGE_RESIZE_MODE              square
IMAGE_SHAPE                    [1024 1024    3]
LEARNING_MOMENTUM              0.9
LEARNING_RATE                  0.001
LOSS_WEIGHTS                   {'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0,
'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE                 14
MASK_SHAPE                     [28, 28]
MAX_GT_INSTANCES               100
MEAN_PIXEL                     [123.7 116.8 103.9]
MINI_MASK_SHAPE                (56, 56)
NAME                           çatlak
NUM_CLASSES                    2
POOL_SIZE                      7
POST_NMS_ROIS_INFERENCE        1000
POST_NMS_ROIS_TRAINING         2000
PRE_NMS_LIMIT                  6000
ROI_POSITIVE_RATIO             0.33
RPN_ANCHOR_RATIOS              [0.5, 1, 2]
RPN_ANCHOR_SCALES              (32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE              1
RPN_BBOX_STD_DEV               [0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD              0.7
RPN_TRAIN_ANCHORS_PER_IMAGE    256
STEPS_PER_EPOCH                500
TOP_DOWN_PYRAMID_SIZE          256
TRAIN_BN                       False
TRAIN_ROIS_PER_IMAGE           200
USE_MINI_MASK                  True
USE_RPN_ROIS                   True
VALIDATION_STEPS               5
WEIGHT_DECAY                   0.0001
```

Şekil 4.10. Çatlak modelinin eğitimi sırasında kullanılan parametreler

4.3. MASK R-CNN MODELİ BÖLGE TEKLİF AĞI ÇALIŞMA PRENSİBİ

Kayar pencerelerin her birinin konumu için k sayıda eş zamanlı bölge teklif önerisinde bulunulur. Bu sebepten regresyon katmanı çıktı olarak önerilen bölgelere ait k adet tutturucu kutucuk için 4k koordinat noktası üretmektedir. Önerilen bölgelerin bir nesneye ait olup olmadığının tahmininin çıktısını ise sınıflandırıcı katmanı üretmektedir. Üretilmiş olan k adet bölge önerisi k adet referans kutucuğa bağlı biçimde incelenmektedir. Referans olarak adlandırılan kutucukların her biri tutturucu (ankor) olarak adlandırılmaktadır. Kayar pencerenin merkez noktasına denk gelebilecek şekilde üretilen bir ankor farklı en, boy ve ölçek bilgisine sahip olabilmektedir.



Şekil 4.11. Bölge teklif ağı çalışma mimarisi [72]

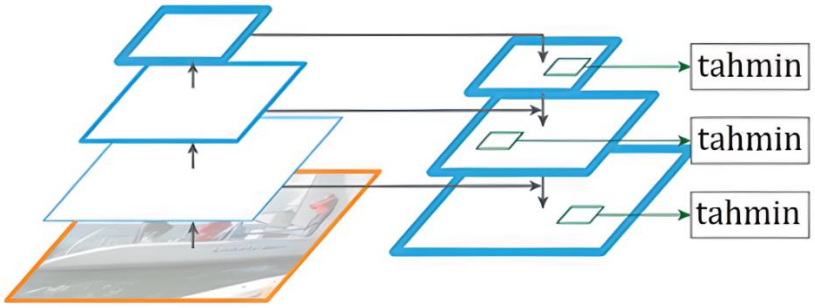
Kayar pencerenin her bir konumu için $W\ x\ H$ boyutlu evrişimli öznitelik haritası oluşur. Oluşan öznitelik haritası için kullanılacak olan tutturucu kutucuk sayısı eşitlik 4.1'de verilmiştir.

$$toplam\ sayı = W\ x\ H\ x\ k \tag{4.1}$$

Şekil 4.11'de verilen görsel yukarıdaki açıklamaları destekler niteliktedir. Bu yöntemde görüntü üzerindeki herhangi bir nesne ötelenirse buna ek olarak öneride ötelenmektedir. Ağın eğitiminde sınıflandırma katmanı tarafından tutturucu kutucukların her birine nesne içerip içermeme durumuna göre ikili etiket belirlenir. Birleşim Üzerinden Kesişim (Intersection Over Unıon, IoU) değeri yüksek olana pozitif, düşük olana ise negatif etiket atanır. Negatif etiket atanması önerilen bölgede herhangi bir nesne olmadığını göstermektedir.

4.4. MASK R-CNN MODELİ ÖZELLİK PİRAMİT AĞI ÇALIŞMA PRENSİBİ

Özellik piramit ağı Şekil 4.12'de görüldüğü üzere ilk piramitten üst düzey özellikleri alıp alt katmanlara iletmekte ve ikinci bir piramidi üstüne koyarak standart bir özellik çıkarma piramidini geliştirmektedir. Bu işlem esnasında her düzeydeki özelliklerin hem alt düzeyde bulunan özelliklere hem de üst düzeyde bulunan özelliklere erişmesine imkan sağlar.



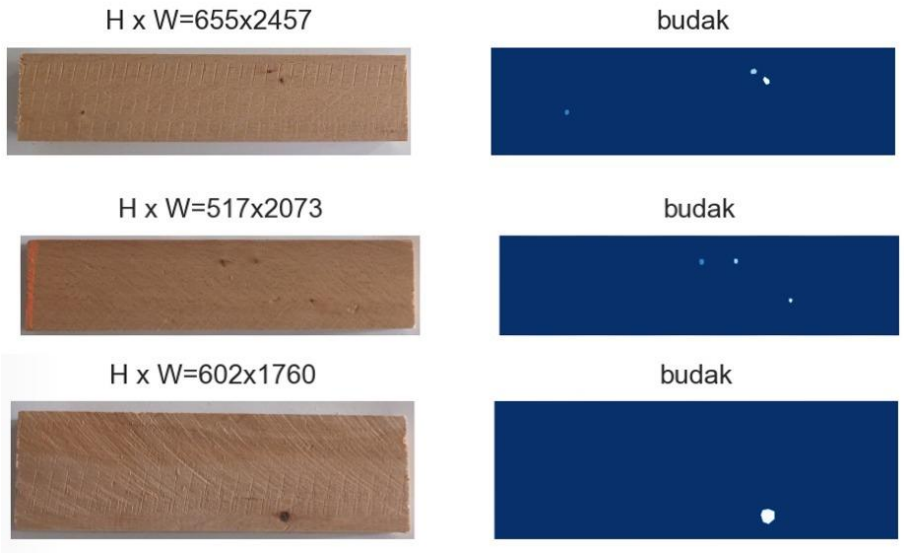
Şekil 4.12. Nesne tespiti esnasında özellik piramit ağının kullanımı [73]

Değişken ölçekli görüntüler ele alındığında görüntü farklı ölçeklerde yeniden boyutlandırma işlemiyle katmanlar haline dönüşmektedir. Bu esnada Şekil 4.12'de aşağıdan yukarıya doğru çözünürlüğün azaldığı görsel piramitleri oluşturulur. Bu yol nesneler üzerinde farklı ölçeklerdeki özelliklerin çıkarılmasına olanak sağlar fakat çözünürlüğün düşmesiyle küçük budak, çatlak ya da yatay desen nesnelerinin gözden kaçırılması mümkün olabilir. Şekil 4.12'de görülen katmanlardaki görüntülerin tamamında, CNN'e özellik çıkarımı için girdiği zaman CNN içerisinde oluşturulan

özellik haritaları derinlik olarak artarken çözünürlük olarak düşmektedir. Ayrıca Şekil 4.12'de görülen üst katmanlar semantik olarak alt katmanlardan daha zengin bilgiye sahiptir.

4.5. LAMEL VERİ SETİ MASKELİ NESNELER

Şekil 4.13'te budak nesnesine ait orijinal görüntüler ve nesnelerin poligon olarak etiketlenmiş ve maskesi oluşturulmuş hali yer almaktadır. Bir görüntü üzerinde birden fazla budak nesnesi yer alabilir ve boyutları birbirinden farklı olabilir, sabit budaklar hariç genellikle ağaç renginde koyu bir oluşuma giderek ayırt edilebilir hale gelirler. Model üzerinden maskelenen budak grupları arasında nokta sabit budak, kuşgözü sabit budak, nokta dağılır budak, kuşgözü dağılır budak yer almaktadır.

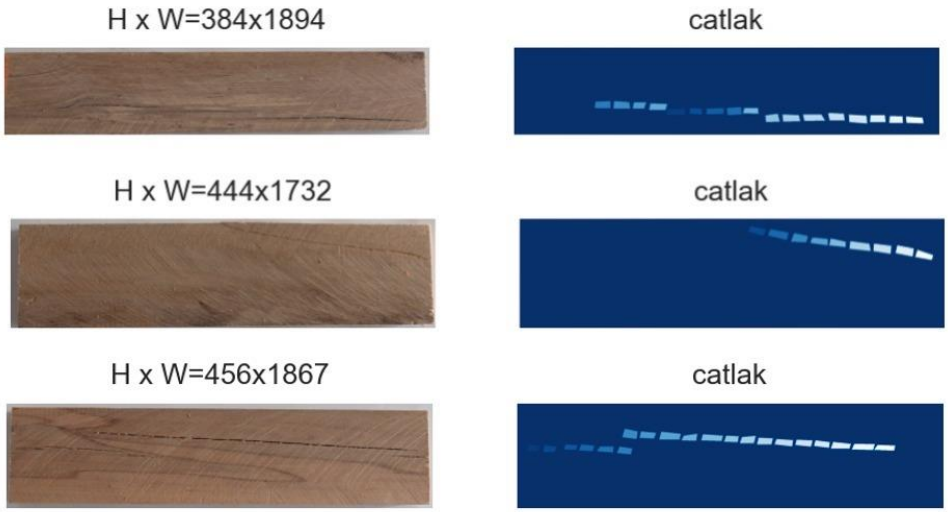


Şekil 4.13. Budak nesnesine ait maskelenmiş görüntü örnekleri

Budaklar literatürdeki çalışmalarda ve gerçek hayat uygulamalarında kusur olarak adlandırılır. Boyutuna ve türüne göre yok edilmesi hedeflenir. Görüntü üzerinde piksel yoğunluğu baz alınarak nesneler üzerinde koyu renkten açık renge gösterim sağlanmaktadır.

Şekil 4.14. Yatay desen nesnesine ait maskelenmiş görüntü örnekleri

Şekil 4.14'te yatay desen nesnesine ait orijinal ve poligonal etiketleme çeşidi kullanılarak maskelenmiş görüntüleri yer almaktadır. Yatay desen nesnesi genelde bir ağaç üzerinde çoklu yapıda yer alır. Yeterince düz olmayan eğimli yapıdaki tomruklar biçme işlemi esnasında lif yönlerinin biçme yönüne paralel olmamasından kaynaklı olarak oluşan nesne çeşididir. Genellikle ağaç cinsi bakımından kayın ve meşe türlerinde görülmektedir.

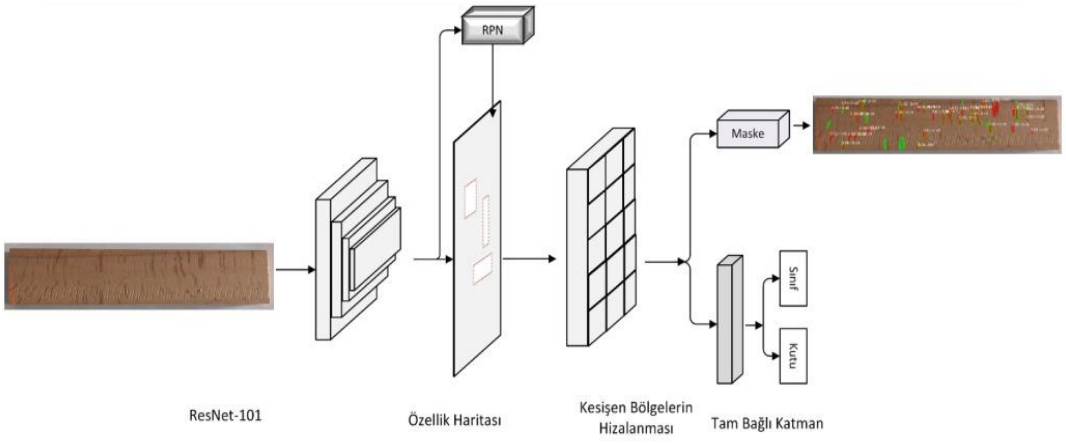


Şekil 4.15. Çatlak nesnesine ait maskelenmiş görüntü örnekleri

Şekil 4.15’te çatlak nesnesine ait orijinal görüntüler ve poligonal etiketleme ile maskelenmiş görüntüler yer almaktadır. Lamel üzerinde çatlak nesnesinin boydan boya, kenarda ya da lamelin köşelerinde oluştuğu görülmektedir. Çatlaklar ağacın görüntü üzerinde görülmeyen iç kısımlarında da yer alabilmektedirler. Ham malzeme kesimi gerçekleştikten sonra uzun süre beklerse çatlama gerçekleşir. Ayrıca bekletildiği alanın güneşli olması da çatlamaya pozitif yönde etki edecektir. Masif panel üretim zincirinde ham malzeme kesildikten sonra biçme ve kurutma işlemlerinden geçer [4]. Bu aşamadan sonra uzun süre herhangi bir işlem görmeden bekletilse bile çatlama olmayacaktır.

4.6. MASK R-CNN MİMARİSİNİN LAMEL VERİ SETİNDE UYGULANMASI

Mask R-CNN mimarisi, Şekil 4.16’da görüldüğü gibi iki ayrı aşama olarak değerlendirilmektedir. Mask R-CNN lamel görüntülerini almakta ve sonrasında görüntü üzerinde tespit ettiği nesneleri maskelemektedir. Mask R-CNN ilk aşamada bölge teklif ağında lamel görüntüsü üzerinde aday nesneler ve aday bölgeler çıkarır. Sonrasında aday olarak belirlenen ankorların özellik haritaları çıkartılır. RPN tarafından iki farklı sonuç oluşturulur. Bu sonuçlardan biri lamel üzerindeki nesnenin hangi sınıfa ait olduğu diğeri ise nesnenin içerisinde bulunduğu sınırlayıcı kutucuğun boyut ve konumudur.



Şekil 4.16. Mask R-CNN mimarisinin lamel veri seti üzerindeki akış diyagramı

RPN'in oluşturduğu sınırlayıcı kutucuklar farklı boyutlarda olabilmektedir. Farklı boyutlarda konumlanan görüntüleri işlemenin zor olması sebebiyle sabit bir girdi boyutu atanmalıdır. Mask R-CNN ağı RoI Pooling işlemi bu aşamada kullanır. RoI Pooling işlemi sonucunda sınıflandırma ve sınırlayıcı kutucuk oluşur. Bu işlemin yanı sıra lamel üzerinde tespit edilen budak, yatay desen ya da çatlak nesnelerinin maskelenmesi de gerçekleştirilir.

4.7. LAMEL VERİ SETİ ÜZERİNDEKİ NESNELERDE SEGMENTASYONU

Mask R-CNN ağı nesne tespiti işleminde varsayılan olarak MS COCO ağırlıklarını kullanır. Model ağırlıkları yüklenirken Şekil 4.17'de MS COCO ağırlıklarıyla işlem yapılmaktadır. Eğer herhangi bir eğitim sonucunda oluşan model varsa son oluşan modelin ağırlıklarının yüklenmesi için gerekli olan kod bloğu oluşturulmuştur.

```
def load_training_model(config):
    model = modellib.MaskRCNN(mode="training", config=config,
                              model_dir=MODEL_DIR)

   #varsayılan olarak coco ağırlıklarını alır
    init_with = "coco"

    if init_with == "imagenet":
        model.load_weights(model.get_imagenet_weights(), by_name=True)
    elif init_with == "coco":

        #coco ağırlıklarının yüklenmesi
        print(COCO_MODEL_PATH)
        model.load_weights(COCO_MODEL_PATH, by_name=True,
                           exclude=["mrcnn_class_logits", "mrcnn_bbox_fc",
                                    "mrcnn_bbox", "mrcnn_mask"])
    elif init_with == "last":
        #son kaydedilen modelin ağırlıkların yüklenmesi
        model.load_weights(model.find_last(), by_name=True)

    return model
```

Şekil 4.17. Son oluşturulan model ağırlıklarının algoritmaya dahil edilmesi adımı

Şekil 4.17'de Mask R-CNN klasörü içerisine logs dizini oluşturulmakta ve kaydedilen modellerin ağırlıklarının tamamı buraya eklenmektedir. Modele yüklenecek olan ağırlık parametrelerinde eğitim işlemi esnasında mrcnn_class_logits, mrcnn_bbox_fc, mrcnn_box değerlerinin de yer alması gerektiğinden Şekil 4.17'de alttaki kod satırı eklenmiştir.

```
MODEL_DIR = os.path.join(ROOT_DIR, "logs")
model = modellib.MaskRCNN(mode="training", config=config, model_dir=MODEL_DIR)
print(COCO_MODEL_PATH)
model.load_weights(COCO_MODEL_PATH, by_name=True,exclude=["mrcnn_class_logits", "mrcnn_bbox_fc", "mrcnn_bbox", "mrcnn_mask"])
```

Şekil 4.18. Modele yüklenecek olan ağırlıkların yolu ve parametreleri

Bu işlem sayesinde model eğitime başladıktan sonraki örnek görüntü Şekil 4.18'de olduğu gibidir.

```
Epoch 1/100
500/500 [==============================] - 221s 442ms/step - loss: 0.7304 - rpn_class_loss: 0.0191 - rpn_bbox_loss: 0.1389 -
mrcnn_class_loss: 0.0512 - mrcnn_bbox_loss: 0.2350 - mrcnn_mask_loss: 0.2863 - val_loss: 0.7403 - val_rpn_class_loss: 0.0163
- val_rpn_bbox_loss: 0.1996 - val_mrcnn_class_loss: 0.0958 - val_mrcnn_bbox_loss: 0.2327 - val_mrcnn_mask_loss: 0.1960
WARNING:tensorflow:From c:\users\windows10\anaconda3\envs\newmaskrcnn\lib\site-packages\keras\callbacks.py:1265: The name tf.
Summary is deprecated. Please use tf.compat.v1.Summary instead.

Epoch 2/100
500/500 [==============================] - 210s 420ms/step - loss: 0.2677 - rpn_class_loss: 0.0042 - rpn_bbox_loss: 0.0472 -
mrcnn_class_loss: 0.0203 - mrcnn_bbox_loss: 0.0586 - mrcnn_mask_loss: 0.1374 - val_loss: 0.7271 - val_rpn_class_loss: 0.0159
- val_rpn_bbox_loss: 0.1963 - val_mrcnn_class_loss: 0.0961 - val_mrcnn_bbox_loss: 0.1436 - val_mrcnn_mask_loss: 0.2751
Epoch 3/100
500/500 [==============================] - 210s 421ms/step - loss: 0.1724 - rpn_class_loss: 0.0023 - rpn_bbox_loss: 0.0308 -
mrcnn_class_loss: 0.0131 - mrcnn_bbox_loss: 0.0282 - mrcnn_mask_loss: 0.0980 - val_loss: 1.0439 - val_rpn_class_loss: 0.0202
- val_rpn_bbox_loss: 0.1716 - val_mrcnn_class_loss: 0.1908 - val_mrcnn_bbox_loss: 0.2285 - val_mrcnn_mask_loss: 0.4329
```

Şekil 4.19. Model eğitimi sırasında ekrana yazdırılan parametreler ve değerleri

Şekil 4.19'da her bir iterasyonun işleminin 500 batch işleminden meydana geldiği görülmektedir. Bir iterasyon işleminin yaklaşık süresinin 221 saniye olduğu görselde yer almaktadır. Bu bilgi sayesinde başlattığımız modelin eğitiminin yaklaşık ne kadar süre sonra tamamlanacağı saptanabilir. Ayrıca burada yer alan genel kayıp değeri, maske kaybı değeri, sınıf kaybı değeri gibi parametreler ele alınarak bir yorumlamaya gidilebilir. Kayıp grafiğinin çizdirilmesi işlemi görsel olarak TensorBoard [74] ile desteklenebilir. TensorBoard, modellediğimiz sinir ağları üzerinde birçok parametrenin grafikler yardımıyla görselleştirilmesine olanak sağlar. TensorFlow kurulumu TensorBoard kullanımı için yeterlidir [74].

```
model.train(dataset_train, dataset_val, learning_rate=config.LEARNING_RATE, epochs=100, layers='heads')
```

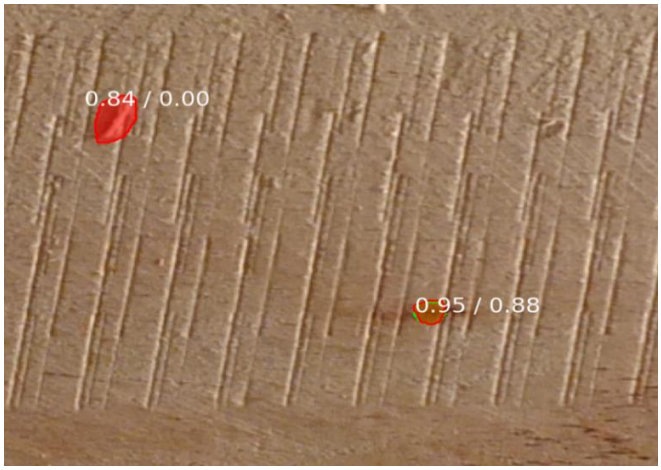
Şekil 4.20. Modelin oluşturulması için eğitim işleminin başlatılması

Model için gerekli konfigürasyon parametrelerinin ayarlanması, model yollarının verilmesi, ilk ağırlıkların atanması işlemlerinden sonra Şekil 4.20'de eğitim işlemi

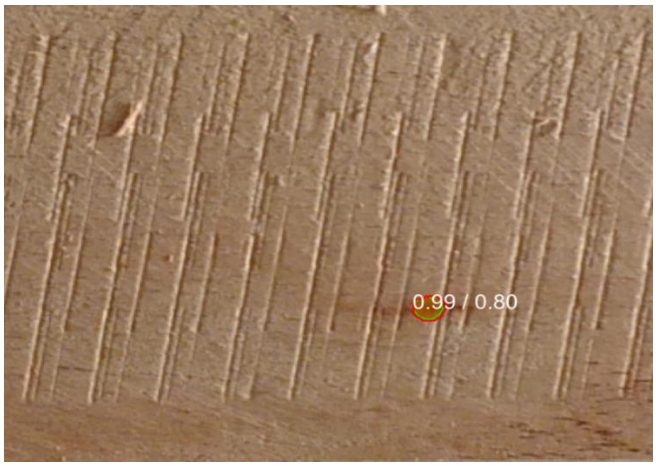
tetiklenmiştir. Öğrenme oranı olan “LEARNING_RATE” parametresine varsayılan değer olarak 0,001 atanmıştır, yapılan testler sonucunda bu parametre değerinde herhangi bir değişikliğe ihtiyaç duyulmamıştır. Şekil 4.20’de kod satırında örnek bir eğitimdeki iterasyon sayısı 100 olarak başlatılmıştır. Bu tez çalışması kapsamında üç ayrı model oluşturulmuştur. Oluşturan modeller için en iyi doğruluk değerine farklı iterasyon sayıları sonucunda ulaşılmıştır.

4.7.1. Budak Nesnesinin Segmente Edilmesi

Bu bölümde budak nesnesi için en iyi doğruluk değerine sahip olan modeli elde etmek hedeflenmiştir. Doğruluk değerinin yanı sıra duyarlılık, hassasiyet, f1-skoru metrikleri de ele alınmıştır. Tüm bu değerler göz önünde bulundurulduğunda budak modeli için oluşturulan en iyi iterasyon sayısının 250 olması gerektiğine karar verilmiştir. Modelin oluşturulması için geçen zaman 15 saat 58 dakika olarak ölçülmüştür. Kayıp 0,3 ile başlayıp 0,1’e düşürülmüştür. Şekil 4.20’de 50 iterasyon sonucunda eğitilen model ağaç üzerine düşen bir ufak bir parçayı budak olarak tespit etmiştir. Yani 50 iterasyonluk işlem sonrasında henüz yeterli öğrenme gerçekleşememiştir. Şekil 4.20’de tespit edilen doğru budak nesnesinin güven skoru 1 üzerinden 0,95 olarak belirlenmiş, IoU skoru ise 0,88 olarak hesaplanmıştır. IoU skoru 1 değerine ne kadar yakın ise tespit edilen nesnenin alanı o kadar güvenilirdir. 250 iterasyon sonrasında oluşan modelde öğrenme gerçekleşmiş ve hedeflenen doğruluk değerine ulaşılmıştır. Şekil 4.21’de yanlış tespit ettiği nesnenin Şekil 4.22’de yer almadığı görülmektedir. Bu da öğrenmenin bu modelde diğer modelden daha iyi olduğunu destekler niteliktedir. Ayrıca Şekil 4.21’de 0,95 olarak belirlenen güven skoru Şekil 4.22’de 0,99’a çıkartılmıştır.



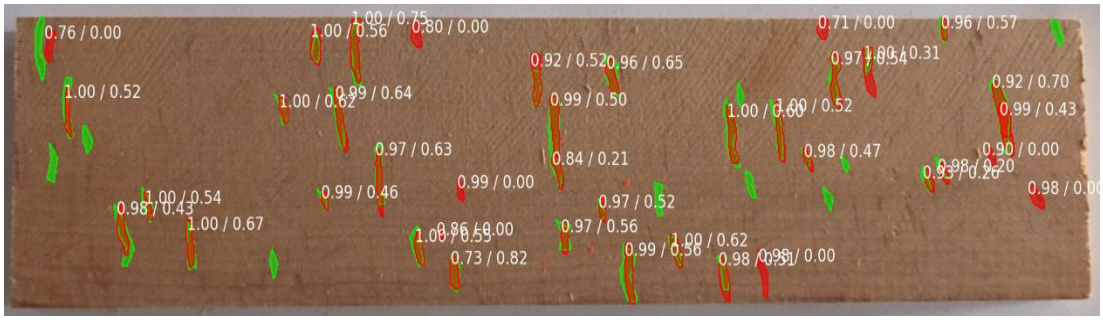
Şekil 4.21. Budak modelinin 50 iterasyon sonrası yaptığı tahmin ve maskeleme



Şekil 4.22. Budak modelinin 250 iterasyon sonrası yaptığı tahmin ve maskeleme

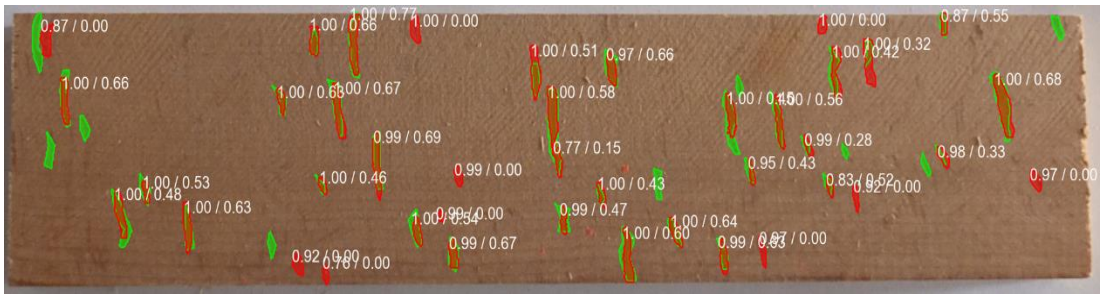
4.7.2. Yatay Desen Nesnesinin Segmente Edilmesi

Yatay desen nesnelerini tespit eden modelin eğitim süresi 21 saat 6 dakika olarak ölçülmüştür. En iyi hassasiyet, duyarlılık, f1-skoru ve doğruluk değerlerine sahip iterasyon sayısı 200 olarak belirlenmiştir. Genel kayıp değeri 0,62'den 0,29'a düşürülmüştür. 50 iterasyon sonucunda oluşan modelin yaptığı tahminler ve IoU skorları Şekil 4.23'te verilmiştir.



Şekil 4.23. Yatay desen modelinin 50 iterasyon sonrası yaptığı tahmin ve maske

Şekil 4.24'te ise 200 iterasyon sonucunda oluşan modelin yaptığı tahminler ve IoU skorları yer almaktadır. Her iki görsel arasında nesne tespiti anlamında aşırı farklılıklar bulunmamaktadır. Modelin görsel üzerindeki yatay desen nesnelerini verimli bir şekilde tespit ettiği görülmektedir. Şekil 2.24'te tespit edilen nesnelerin birçoğunun güven skorunun diğer şekilde tespit edilen nesnelere göre artışta olduğu saptanmıştır.



Şekil 4.24. Yatay desen modelinin 200 iterasyon sonrası yaptığı tahmin ve maske

4.7.3. Çatlak Nesnesinin Segmente Edilmesi

Çatlak modeli için eğitimde doldurulması gereken süre 11 saat 15 dakika olarak ölçülmüştür. İterasyon sayısı 50 ile başlatılıp 50'şer artış yapılarak 250'ye kadar gerçekleştirilen işlemlere ait tespit sonuçları ele alınmıştır. Genel kayıp değeri 1,4 ile başlayarak 0,59'a kadar düşürülmüştür. Şekil 4.25'te 50 iterasyon işlemi sonrasındaki modelin tespit ettiği güven skoru ve IoU skoru yer almaktadır. Şekil 4.26'da ise 150 iterasyon sonrasındaki güven ve IoU skor değerlerine ait görsele yer verilmiştir. Her iki görselde de yer alan çatlak nesnelerinin verimli bir şekilde tespit edildiği görülmektedir.

Şekil 4.25. Çatlak modelinin 50 iterasyon sonrası yaptığı tahmin ve maskeleme

Şekil 4.26. Çatlak modelinin 150 iterasyon sonrası yaptığı tahmin ve maskeleme

Bu tez çalışması kapsamında yatay desen ve çatlak ve budak modellerinin budak, yatay desen ve çatlak içeren görsellerde sayısına bakılmaksızın nesne tespiti gerçekleştirmesi yeterli olacaktır. Eğitilen modellerin bu gereksinimi karşıladığı görülmektedir.

4.8. LAMEL VERİLERİNİN SINIFLANDIRILMASI İŞLEMİ

Lamel verilerinin sınıflandırılması ile aynı sınıfa ait lamellerin aynı masif panel levhası üzerinde yer alması hedeflenmiştir. Bir masif panel levhası üzerinde farklı kalitelere sahip lameller yer almamalıdır. Bu işlemin aksinin gerçekleşmesi panellerde kaliteyi bozmaktadır. Bu alanda üretim gerçekleştiren birçok tesiste bu işlem insan gücüne dayalı olduğundan sık sık hatalarla karşılaşılmaktadır. Sınıflandırmada yaşanan bu problem üretilen masif panelin maliyetinin altında rakamlara satılmasına sebep olmaktadır.

Masif panellerin sınıflandırılmasında firmalara göre çok çeşitli sınıflar yer almaktadır. Masif panel üretimi gerçekleştiren bir firma panellerini AA, AB, AC, BB, BC, CC

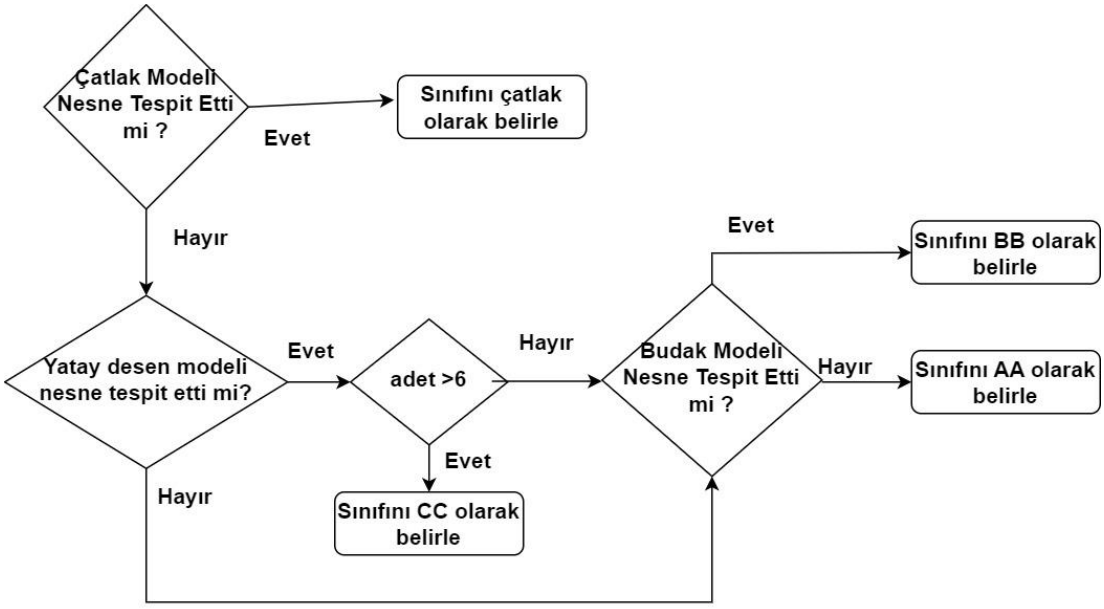
şeklinde sınıflandırmaktadır. Başka bir firma ise AA, BB, CC şeklinde sınıflandırmaktadır. Yani masif panellerin sınıflandırılmasında firmalar arasında belirlenmiş sabit bir sınıflandırma şekli yoktur. Bu sınıflar üretim dönemi esnasında müşterinin isteğine göre belirlenmektedir. Sınıflandırma işleminde firmalar tarafından benimsenen sabit sınıf sayısı olmamasına rağmen, sınıflandırma yaparken lamel üzerinde dikkate alınan yapılar hep aynıdır. Bu tez çalışması kapsamında bu yapılardan budak, yatay desen ve çatlak nesneleri ele alınmıştır.

Bölüm 4.7'de lamel veri seti üzerinde budak, yatay desen ve çatlak nesnelerine ait tespit işleminin kod bloklarında nasıl gerçekleştirildiğinin ve tespit edilen nesnelerin ne anlama geldiğinin detayları yer almaktadır. Bu nesnelerin görüntü üzerinde var olup olmama durumuna göre sınıflandırmaya gidilmiş ve başarım metrikleri ölçülmüştür. Oluşturulan üç modele görüntü sorularak ilgili yapıları tespit etmesi beklenmiştir. Tespit ettiği nesne sayısının herhangi bir önemi bulunmamaktadır. Model bir görüntü üzerinde on adet çatlak yer alsa bile bir adet çatlak tespit ettiği durumda bekleneni karşıladığı anlamına gelir. Budak modeline ait tespitler içinde aynı durum geçerlidir. Bir adet budak nesnesinin tespit edilmesi sınıflandırma için yeterlidir. Yatay desen nesnesi içinde aynı durum geçerlidir fakat nihai sınıflandırmada başarı yüzdesini artırabilmek için farklı bir kural uygulanmıştır. Aşağıda bu kuralın detayına yer verilmiştir.

Çizelge 4.1. Kalite sınıfları ve bu sınıflarda yer alması beklenen yapılar

Kalite Sınıfları	**Sınıflarda Yer Alan Yapılar**
AA	Herhangi bir yapı yer almamalı
BB	Budak yer almalı
CC	Yatay desen zorunlu, budak seçimli
Çatlak	Çatlak zorunlu, budak ve yatay desen seçimli

Çizelge 4.1'de bu tez çalışması kapsamında kalite sınıflarında yer alması beklenen yapılar listelenmiştir. Budak, yatay desen ve çatlak nesneleri tespit edildiğinde yukarıdaki listeye göre kalite sınıflandırması işlemi gerçekleşmektedir.



Şekil 4.27. Nihai sınıflandırmaya ait algoritma akış diyagramı

Bu yapıların yalnızca bir adet tespit edilmesinin sınıflandırma işlemi için yeterli olacağına değinilmiştir. Fakat sınıflandırma başarımını iyileştirmek için yatay desen sınıfında farklı bir yöntem izlenmiştir. Yatay desen nesnesi bir lamel üzerinde genelde çok fazla sayıda yer alır. Yatay desen lif yönlerine bağlı olarak oluşan bir yapı olduğundan az sayıda olması beklenmez. Görüldüğü durumlarda olur fakat nadir rastlanır. Bu sebepten eğer nesne üzerinde altı ya da daha fazla yatay desen nesnesi tespit edildiyse algoritma ilgili görseli CC sınıfına koymaktadır. Bu koşul CC sınıfında genel bir düşüşe sebep olmasına rağmen diğer sınıflardaki doğruluğu artırmıştır. Bu çalışma kapsamından yapılan nesne tespiti ve sonrasındaki kalite sınıflandırması işlemlerine ait nihai doğruluk %90,90 olarak saptanmıştır. Modellerin tahminlemesi sonrasında gidilen nihai sınıflandırma işlemine ait akış diyagramı Şekil 4.27'de verilmiştir.

BÖLÜM 5

DENEYSEL ÇALIŞMALAR

Mask R-CNN derin öğrenme ağı kullanılarak 3 ayrı model eğitilmiş ve eğitim sonrası da test kümesi bir sınıflandırma işlemine tabi tutulmuştur. Modellerin ulaştığı en iyi hassasiyet, duyarlılık, doğruluk ve f1-skor değerleri göz önünde bulundurularak 50, 100, 150, 200, 250 iterasyon sonucunda oluşan modellerden hangisinin proje kapsamında kullanılacağına karar verilmiştir. İterasyonlar sonucunda modelin elde ettiği mAP parametresi göz önünde bulundurulsa da tek başına yeterli olmayacağı değerlendirilmiştir. Modelin eğitimler sonrasında hesapladığı mAP değerinin bu tez çalışması kapsamında önemi yoktur, çünkü görsel üzerinde tespit edilecek yapının konumu, kaç tane olduğu önemli değildir. mAP gerçek referans etiketleri üzerinden hesaplama yapan bir değerdir ve etiketlenen tüm nesneleri bire bir doğru bulduğunda yüksek yüzdelik dilimlere ulaşır. Fakat bu çalışma kapsamında gerçek referans değerli tüm etiketlerin tespit edilmesine gerek yoktur. Çatlak bulunan nesne üzerinden birden fazla çatlak yer almasına rağmen modele sorulduğunda çatlaklardan bir tanesini bile bulsa yeterli olmaktadır. Bir adet çatlak nesnesinin tespiti ya da bir adet yatay desen nesnesinin tespiti lamel parçasının sınıfını belirlemede yeterlidir. Bu sebepten modelin eğitim sonrasında oluşturduğu mAP değerlerinin dışında sınıflandırmaya gidilmiş ve hassasiyet, doğruluk, duyarlılık ve f1-skoru parametreleri test verileri üzerinden yeniden hesaplanmış ve en doğru değerler elde edilmiştir. Sonrasında da nihai sınıflandırma işlemi gerçekleştirilmiştir. Aşağıdaki tabloda Mask R-CNN yönteminde, var yok sınıflandırmasında ve nihai sınıflandırmada kullanılan görüntü ve etiket sayıları tablo halinde verilmiştir.

Çizelge 5.1. Projede kullanılan toplam görüntü ve etiket sayıları

	Kullanılan Toplam Görüntü Sayısı	**Toplam Etiket Sayısı**
Budak Modeli Eğitimi	656	1006
Çatlak Modeli Eğitimi	1587	7935
Yatay Desen Modeli Eğitimi	237	3103
Var/ Yok Sınıflandırılması	360	-
Nihai Sınıflandırma	132	-
Toplam	2972	12044

Başarım metrikleri oluşturulan modelin test kümesi üzerinde doğruluğunu saptamak için kullanılmıştır. Modelin performans parametreleri hassasiyet, duyarlılık, f1-skoru ve doğruluk olarak belirlenmiştir. Literatürde sıkça kullanılan parametrelerin değerleri sırasıyla ölçümlenmiştir.

Çizelge 5.2. Karmaşıklık matrisi

		TAHMİN	
GERÇEK		**VAR**	**YOK**
	VAR	Doğru Pozitif	Yanlış Negatif
	YOK	Yanlış Pozitif	Doğru Negatif

Başarım metrikleri hesaplamasında yer alan terimler doğru negatif (DN), doğru pozitif (DP), yanlış negatif (YN), yanlış pozitif (YP)'dir. Bu değerler kullanıcı tarafından test öncesinde etiketlenen verinin sınıfı ve model tarafından yapılan tahmin etme işleminin sonucunda gereken uyuşmanın sağlanıp sağlanmaması durumuna göre oluşturulmaktadır. Çizelge 5.1'de var ve yok olarak adlandırılan hücreler, tespit edilmesi istenen nesnenin görselde yer alıp almama durumuna göre şekillenmektedir. Bu çizelgede var yok olarak yazılan hücreleri budak var ya da yok olarak değerlendirmeliyiz. Bu durumda doğru pozitif terimi gerçekte budak olarak sınıflandırılan görüntünün model tarafından da budak olarak sınıflandırılmasıdır. Yanlış negatif gerçekte budak olarak sınıflandırılan görüntünün model tarafından

budak değil olarak sınıflandırılmasıdır. Yanlış pozitif gerçekte budak değil olarak sınıflandırılan görüntünün model tarafından budak olarak sınıflandırılmasıdır. Doğru negatif gerçekte budak değil olarak sınıflandırılan görüntünün model tarafından da budak değil olarak sınıflandırılmasıdır. Ele alınan terimler hassasiyet, duyarlılık, f1-skoru ve doğruluk hesaplamalarında kullanılacaktır.

Eşitlik 5.1, 5.2, 5.3, ve 5.4'te hassasiyet, duyarlılık, f1-skoru, ve doğruluk denklemlerine yer verilmiştir.

$$Hassasiyet = \frac{DP}{DP+YP} \tag{5.1}$$

Doğru olarak tahmin edilenlerin toplam var tahminlerine oranıdır. Bu değerin yüksek olması model seçimlerinde önemli bir kriterdir. Eşitlik 5.1 hassasiyet parametresinin formülüdür. Modelin ne sıklıkla doğru tahmin ettiğinin ölçüsüdür.

$$Duyarlılık = \frac{DP}{DP+YN} \tag{5.2}$$

Eşitlik 5.2'de duyarlılık parametresinin formülüne yer verilmiştir. Pozitif olarak tahmin edilmesi gereken görsellerin ne kadarının pozitif tahmin edildiğini gösterir. Modelin doğruları bilme konusundaki etkinliği olarak da adlandırılabilir.

Bir model yüksek duyarlılığa sahip ancak hassasiyeti düşük ise pozitif örneklerin çoğunu doğru bir şekilde sınıflandırır, ancak birçok yanlış pozitife sahiptir. Bir modelin hassasiyeti yüksek ancak duyarlılığı düşük ise model bir örneği pozitif olarak sınıflandırdığında doğrudur ama pozitif örneklerin yalnızca bir kısmını sınıflandırabilir.

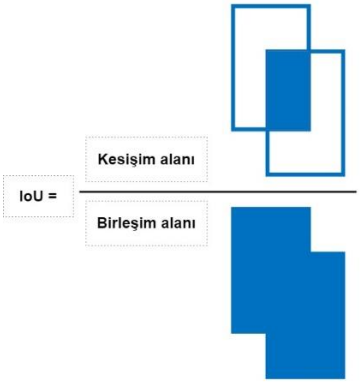
$$F1 - Skoru = \frac{(2\ x\ Hassasiyet\ x\ Duyarlılık)}{(Hassasiyet+Duyarlılık)} \tag{5.3}$$

F1- skor hassasiyet ve duyarlılık parametrelerinin harmonik ortalamasıdır. Aritmetik yerine harmonik ortalama olmasının sebebi uç durumların göz ardı edilmemesi gerekliliğinden kaynaklanır. Hassasiyet değeri 1 ve duyarlılık değeri 0 olsaydı f1-skor

0,5 olacaktı ve bu yanıltıcı bir durum oluşturur. Eşit dağılmayan veri kümelerinde hatalı model seçimi yapmayı engeller. Eşitlik 5.3, f1-skor metriğinin formülünü içermektedir.

$$Doğruluk = \frac{(DP+DN)}{(DP+YP+DN+YN)} \tag{5.4}$$

Doğruluk doğru sınıflandırmanın toplam rakama bölünmesiyle elde edilir. Eşitlik 5.4'te doğruluk metriği formülize edilmiştir.



Şekil 5.1. IoU skor hesaplama gösterimi [75]

Şekil 5.1'de IoU skoruna ait hesaplama işlemine görsel bir örnek vasıtasıyla yer verilmiştir. Örtüşen bölgelerin kesişim alanına, bölgelerin birleşimi bölünerek IoU skor hesaplanmaktadır.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} APi \tag{5.5}$$

mAP değeri, hassasiyet, duyarlılık, f1-skor gibi değerlerin tek bir formül ile değerlendirilmesi adına tasarlanan bir parametredir. Hassasiyet ve duyarlılık arasındaki dengeyi içerir ve hem YP hem de YN'leri dikkate almaktadır.

mAP değeri IoU eşiğine bağlı olarak değişiklik gösterir. Modelin doğruluk değeri her farklı eşik değerinde değişebileceğinden doğru eşik değeri her model için farklılık göstermektedir.

Mask R-CNN ağında eğitilen çatlak, budak ve yatay desen modelleri nesne tespit etmekten ziyade ilgili nesnenin var yok durumuna göre işlemlere devam edecektir. Bu sebepten Çizelge 5.3'te budak sınıfına ait karmaşıklık matrisine yer verilmiştir.

5.1. MODELLERE AİT KARMAŞIKLIK MATRİSLERİ, BAŞARIM METRİKLERİ VE DEĞERLERİ

Çizelge 5.3. Budak modeline ait karmaşıklık matrisi

		TAHMİN		
		Budak	**Budak Değil**	**Toplam**
GERÇEK	**Budak**	61 (DP)	0 (YN)	61
	Budak Değil	5 (YP)	54 (DN)	59
	Toplam	66	54	120

Bölüm 4.7.1'de budak nesnesinin segmente edilmesi başlığı altında en iyi iterasyon Mask R-CNN'de oluşturulan modelin 250 iterasyon sonucunda ortaya çıktığı bilgisine ve detayların yer verilmiştir. Modelin asıl başarısı test edilen görseller üzerinde budak sınıfına ait mi değil mi sorusuna verdiği cevaplar üzerinde ölçümlenecektir.

Budak modeli için test işlemi esnasında 120 adet test görseli toplanmıştır. Bu veri setinin içerisinde 61 adet budak sınıfına ait görsel, 59 adet budak sınıfına ait olmayan görsel yer almaktadır. 61 adetlik sayıyı DP ve YN grubuna ait rakamlar oluştururken, 59 adetlik sayı YP ve DN grubuna ait rakamlardan meydana gelmektedir. Çizelge 5.3'te verilen budak modeline ait karmaşıklık matrisi yukarıdaki bilgileri destekler niteliktedir.

Çizelge 5.4. Budak modeli başarım metrikleri ve değerleri

Budak modeli başarım metrikleri	**Parametre değerleri**
Hassasiyet	0,92424242
Duyarlılık	1
F1-skor	0,96062992
Doğruluk	0,95833333

Çizelge 5.4'te budak modeli için başarım metriklerine ait değerlere yer verilmiştir. Doğruluk metriğinin %95,83 olarak ölçülmesi budak modelinin sınıflandırma yeteneği adına yeterli olarak değerlendirilmiştir.

Çizelge 5.5. Çatlak modeline ait karmaşıklık matrisi

		TAHMİN		
		Çatlak	**Çatlak Değil**	**Toplam**
GERÇEK	**Çatlak**	64 (DP)	14 (YN)	78
	Çatlak Değil	4 (YP)	38 (DN)	42
	Toplam	68	52	120

Mask R-CNN ağı vasıtasıyla eğitilen en iyi modelin 200 iterasyon sonucunda oluştuğu bilgisine bölüm 4.7.2'de çatlak nesnesinin segmente edilmesi başlığı altında yer verilmiştir. Bu aşamada çatlak modeli için ayrılan test kümesine ait görsellerde çatlak sınıfına ait ya da çatlak sınıfına ait değil şeklinde bir sınıflandırmaya gitmesi beklenmiştir. Çatlak modelinin sınıflandırılması aşaması için 120 adet görüntü test işlemine ayrılmıştır. Bu veri setinde 78 adet çatlak sınıfına ait görsel 42 adet çatlak olmayan sınıfa ait görsel yer almaktadır. Modele ait sınıflandırma işlemi sonucunda Çizelge 5.5'te yer alan karmaşıklık matrisi ortaya çıkmıştır. Bu matriste 78 adet görsel DP ve YN gruplarının toplamından oluşmuştur. Bu toplam çatlak sınıfına ait görsellerin tamamıdır. Çatlak sınıfına ait olmayan görsellerin grubu ise YP ve DN'dir. Çizelge 5.5'te yer alan veriler bu verilen bilgileri doğrular niteliktedir.

Çizelge 5.6. Çatlak modeli başarım metrikleri ve değerleri

Çatlak modeli başarım metrikleri	Parametre değerleri
Hassasiyet	0,94117647
Duyarlılık	0,82051282
F1-skor	0,87671233
Doğruluk	0,85

Çizelge 5.6'da çatlak modeline ait başarım metriklerine yer verilmiştir. Buradaki doğruluk değerinin %85 olarak ölçümlenmesi çatlak sınıflandırma işlemini gerçekleştiren modelin bir sonraki aşamada yapılacak olan sınıflandırma işlemi için hazır olduğunu göstermektedir.

Çizelge 5.7. Yatay desen modeline ait karmaşıklık matrisi

		TAHMİN		
		Yatay Desen	**Yatay Desen Değil**	**Toplam**
GERÇEK	**Yatay Desen**	60 (DP)	5 (YN)	65
	Yatay Desen Değil	23 (YP)	32 (DN)	55
	Toplam	83	37	120

Bölüm 4.7.3'te yatay desen nesnesine ait segmentasyon işlemi başlığı altında en iyi iterasyon sayısının 150 olduğuna karar verilmiş ve detaylı bilgi sunulmuştur. Bu aşamada elde edilen modelin test kümesinin içerisinde yer alan görsellere ait bir sınıflandırma yapması beklenmiştir. Burada yatay desen sınıfı mı yoksa değil mi sorusunun cevabını veren bir sınıflandırma aşaması uygulanmaktadır. Bu sınıflandırma sonrasında Çizelge 5.7'de yer alan veriler elde edilmiştir.

Yatay desen modelinin yapacağı sınıflandırma işlemi için 120 adet görsel ayarlanmıştır. Bu veri setinde 65 adet yatay desen sınıfına ait görüntüye yer verilirken, 55 adet yatay desen olmayan sınıfa ait görüntüye yer verilmiştir. DP ve YN toplamı yatay desen sınıfına ait görsellerin toplamına yani 65'e eşittir. YP ve DN gruplarının toplamı yatay desen olmayan sınıfa ait görsellerin toplamına yani 55'e eşittir. Çizelge

5.7’da verilen değerler bu rakamları destekler niteliktedir. Buradaki elde edilen değerlerden yatay desen nesnesi için gerekli başarım metriklerinin değerlerinin hesaplanması sağlanmıştır.

Çizelge 5.8. Yatay desen modeli başarım metrikleri ve değerleri

Yatay desen modeli başarım metrikleri	**Parametre değerleri**
Hassasiyet	0,72289157
Duyarlılık	0,92307692
F1-skor	0,81081081
Doğruluk	0,7666667

Çizelge 5.7’de elde edilen değerlerden Çizelge 5.8’deki modelin performansını ölçen değerlere geçilmiştir. Buradaki parametrelerinin her birinin ayrı bir önemi vardır. Esas olan hepsinin belirli bir aralık bandından yer almasıdır. Yani çok uç rakamların aynı grupta yer almaması çok önemlidir. Yatay desen modeline ait sınıflandırma işleminde doğruluk %76,66 olarak ölçülmüştür. Bir sonraki aşamada gerçekleştirilecek olan nihai sınıflandırma işlemi için %76,66 değeri yeterli bir sonuç olarak değerlendirilmiştir.

5.2. NİHAİ SINIFLANDIRMA SONUÇLARI

Tez çalışması kapsamında AA, BB, CC ve Çatlak isimli 4 adet nihai sınıf belirlenmiştir. Her sınıfa ait 33 adet görsel kullanılmış toplamda veri setinde 132 görüntü sayısına ulaşılmıştır. Çizelge 5.9, 5.10, 5.11 ve 5.12’de tek tek görseller üzerinde yer alan yapılar, modelin tespit ettiği yapılar, görselin gerçekte ait olduğu sınıf ve algoritmanın görselin hangi sınıfa ait olması gerektiğine dair verdiği karar yer almaktadır.

Çizelge 5.9. AA sınıfına ait veri setinde modelin tespiti ve algoritmanın kararı

no	Var Olan Yapı	Tespit Ettiği Yapı	Kendi Sınıfı	Algoritmanın Karar Verdiği Sınıf
a_1	-	-	AA	AA
a_2	-	1 yatay desen	AA	AA
a_3	-	2 yatay desen	AA	AA
a_4	-	-	AA	AA
a_5	-	-	AA	AA
a_6	-	4 yatay desen	AA	AA
a_7	-	-	AA	AA
a_8	-	1 yatay desen	AA	AA
a_9	-	-	AA	AA
a_10	-	-	AA	AA
a_11	-	6 yatay desen	AA	AA
a_12	-	12 yatay desen	**AA**	**CC**
a_13	-	3 yatay desen	AA	AA
a_14	-	-	AA	AA
a_15	-	-	AA	AA
a_16	-	-	AA	AA
a_17	-	-	AA	AA
a_18	-	1 yatay desen	AA	AA
a_19	-	-	AA	AA
a_20	-	-	AA	AA
a_21	-	-	AA	AA
a_22	-	-	AA	AA
a_23	-	-	AA	AA
a_24	-	-	AA	AA
a_25	-	1 yatay desen	AA	AA
a_26	-	-	AA	AA
a_27	-	-	AA	AA
a_28	-	-	AA	AA

a_29	-	1 budak	**AA**	**BB**
a_30	-	1 budak	**AA**	**BB**
a_31	-	-	AA	AA
a_32	-	-	AA	AA
a_33	-	-	AA	AA

Çizelge 5.9. (devam ediyor)

AA sınıfına ait 33 adet görsel kullanılmıştır. Bu veri setindeki görseller üzerinde hiçbir yapı yer almamaktadır. Buna rağmen, veri setinde model 11 adet görselde yatay desen ve budak tespiti gerçekleştirmiştir. Model eğer 6 adet üzerinde yatay desen nesnesini tespit ettiyse CC sınıfında olmalıdır. Toplamda bir görsel üzerinde 6 adet ve altında yatay desen nesnesinin tespit edildiğinde nihai sınıflandırmada dikkate alınmamaktadır. Bu nihai sınıflandırma algoritmanın başarısını artırabilmek için belirlenmiş bir kuraldır. AA, BB, CC ve Çatlak sınıflarının tamamında bu kural dikkate alınarak algoritma çalıştırılmaktadır. Bu kural kapsamında Çizelge 5.9'da yer alan görsellerden yalnızca a_12, a_29 ve a_30 numaralarına sahip olan satırlarda yanlış sınıflandırma gerçekleşmiştir. 30 doğru 3 yanlış ile AA sınıfına ait başarı oranı %90,90 olarak ölçülmüştür.

Çizelge 5.10. BB sınıfına ait veri setinde modelin tespiti ve algoritmanın kararı

no	Var Olan Yapı	Tespit Ettiği Yapı	Kendi Sınıfı	Algoritmanın Karar Verdiği Sınıf
b_353	budak	1 budak	BB	BB
b_373	budak	1 budak	BB	BB
b_388	budak	1 yatay desen, 1 budak	BB	BB
b_389	budak	1 yatay desen, 1 budak	BB	BB
b_412	budak	1 budak	BB	BB
b_443	budak	1 budak	BB	BB
b_445	budak	1 budak	BB	BB

b_458	budak	1 budak	BB	BB
b_1572	budak	1 budak	BB	BB
b_2063	budak	3 budak	BB	BB
b_2183	budak	1 budak	BB	BB
b_2222	budak	2 budak	BB	BB
b_2288	budak	3 budak	BB	BB
b_2325	budak	1 yatay desen, 5 budak	BB	BB
b_2377	budak	1 budak	BB	BB
b_2412	budak	1 budak	BB	BB
b_2440	budak	2 budak	BB	BB
b_2465	budak	2 budak	BB	BB
b_2470	budak	1 budak	BB	BB
b_2492	budak	3 budak	BB	BB
b_2577	budak	3 budak	BB	BB
b_2607	budak	1 budak	BB	BB
b_2611	budak	4 budak	BB	BB
b_2772	budak	1 budak	BB	BB
b_2776	budak	1 budak	BB	BB
b_2490	budak	3 budak	BB	BB
b_2968	budak	1 budak	BB	BB
b_3198	budak	1 budak	BB	BB
b_3450	budak	2 budak	BB	BB
b_3592	budak	1 yatay desen, 1 budak	BB	BB
b_3878	budak	1 budak	BB	BB
b_4025	budak	3 budak	BB	BB
b_4146	budak	3 yatay desen, 1 budak	BB	BB

Çizelge 5.10. (devam ediyor)

BB sınıfının test verisi kümesinde 33 adet görsel kullanılmıştır. Bu veri setindeki görseller üzerinde yalnızca budak nesnesi yer almaktadır. Buna rağmen yatay desen modeli 5 görselde yatay desen nesne tespiti gerçekleştirmiştir. Bölüm 4.8'de verilen lamel verilerinin sınıflandırılması işlemi bölümü altında nihai sınıflandırma işleminin akış diyagramı verilmiştir. Burada da belirtildiği üzere 6 adet ve altında yatay desen nesnesi tespit edildiyse bu sınıflandırmada dikkate alınmamaktadır. Bu koyulan kural sayesinde 5 görselde yanlış tespit edilen yatay desen nesnesi sınıflandırmada başarı oranını düşürmemiştir. BB sınıfına ait sınıflandırma işleminde 33 görselde 33'ü de algoritma tarafından doğru sınıfa ayrılmıştır. Test verisi üzerinde BB sınıfına ait elde edilen başarı oranı %100 olarak belirlenmiştir. Çizelge 5.10'da bu bilgilere ait detaylar yer almaktadır.

Çizelge 5.11. CC sınıfına ait veri setinde modelin tespiti ve algoritmanın kararı

no	Var Olan Yapı	Tespit Ettiği Yapı	Kendi Sınıfı	Algoritmanın Karar Verdiği Sınıf
c_2038	yatay desen	41 yatay desen	CC	CC
c_2109	yatay desen	9 yatay desen	CC	CC
c_2114	yatay desen	33 yatay desen	CC	CC
c_2126	yatay desen	32 yatay desen	CC	CC
c_2136	yatay desen	6 yatay desen, 2 budak	**CC**	**BB**
c_2141	yatay desen	9 yatay desen	CC	CC
c_2156	yatay desen	23 yatay desen	CC	CC
c_2157	yatay desen	43 yatay desen	CC	CC
c_2126	yatay desen	32 yatay desen	CC	CC
c_2190	yatay desen	8 yatay desen	CC	CC
c_2202	yatay desen	1 çatlak, 8 yatay desen	**CC**	**Çatlak**
c_2210	yatay desen	2 çatlak, 8 yatay desen	**CC**	**Çatlak**

c_2251	yatay desen	32 yatay desen	CC	CC
c_2253	yatay desen	11 yatay desen, 1 budak	CC	CC
c_2260	yatay desen	1 çatlak, 27 yatay desen, 2 budak	**CC**	**Çatlak**
c_2266	yatay desen	28 yatay desen	CC	CC
c_2285	yatay desen	9 yatay desen, 2 budak	CC	CC
c_2321	yatay desen	23 yatay desen	CC	CC
c_2327	yatay desen	8 yatay desen	CC	CC
c_2356	yatay desen	10 yatay desen	CC	CC
c_2381	yatay desen	12 yatay desen	CC	CC
c_2400	yatay desen	14 yatay desen	CC	CC
c_2408	yatay desen	23 yatay desen	CC	CC
c_2418	yatay desen	18 yatay desen	CC	CC
c_2447	yatay desen	19 yatay desen	CC	CC
c_2538	yatay desen	29 yatay desen	CC	CC
c_2583	yatay desen	32 yatay desen, 3 budak	CC	CC
c_2832	yatay desen	10 yatay desen	CC	CC
c_3701	yatay desen	15 yatay desen	CC	CC
c_3705	yatay desen	6 yatay desen	**CC**	**AA**
c_3710	yatay desen	7 yatay desen	CC	CC
c_4282	yatay desen	8 yatay desen	CC	CC
c_4797	yatay desen	3 yatay desen	**CC**	**AA**

Çizelge 5.11. (devam ediyor)

CC sınıfının test verisi kümesinde 33 adet görsel yer almaktadır. Bu veri setinde yalnızca yatay desen nesnesine yer verilmiştir. Görseller üzerinde modeller tarafından budak, çatlak ve yatay desen tespit edilmiştir. C_2136 numaralı görselde 6 adet yatay desen ve 2 adet budak tespiti sağlanmıştır. Yatay desen nesnesinin sayısı 6'ya eşit ve

6’dan küçük ise sınıflandırmada göz ardı edilmektedir. Bu sebepten görsel budak nesnesinden dolayı BB sınıfına ayrılmıştır. Bunun haricinde 3 adet görsel çatlak tespit edildiğinden sınıfı çatlak olarak belirlenmiş ve 1 adet görsel 6 adet yatay desen tespitinden dolayı AA sınıfına ayrılmıştır. 33 adet görselden 27 doğru 6 yanlış sınıflandırma ile CC sınıfına ait başarı oranı %81,81 olarak belirlenmiştir. Çizelge 5.11 bu bilgileri destekler niteliktedir.

Çizelge 5.12. Çatlak sınıfına ait veri setinde modelin tespiti ve algoritmanın kararı

no	Var Olan Yapı	Tespit Ettiği Yapı	Kendi Sınıfı	Algoritmanın Karar Verdiği Sınıf
ca_1122	çatlak	1 çatlak	Çatlak	Çatlak
ca_2073	çatlak ,yatay desen	1 çatlak, 25 yatay desen	Çatlak	Çatlak
ca_6010	çatlak	8 çatlak, 1 yatay desen	Çatlak	Çatlak
ca_6016	çatlak	5 çatlak, 1 yatay desen	Çatlak	Çatlak
ca_6017	çatlak, budak	3 çatlak, 1 yatay desen,1 budak	Çatlak	Çatlak
ca_6019	çatlak	8 çatlak, 1 yatay desen	Çatlak	Çatlak
ca_6020	çatlak	27 çatlak, 1 yatay desen,2 budak	Çatlak	Çatlak
ca_6024	çatlak	4 çatlak, 1 budak	Çatlak	Çatlak
ca_6027	çatlak	3 çatlak, 1 budak	Çatlak	Çatlak
ca_6028	çatlak, yatay desen	1 çatlak,4 yatay desen,3 budak	Çatlak	Çatlak
ca_6033	çatlak, yatay desen, budak	3 çatlak, 10 yatay desen,3 budak	Çatlak	Çatlak
ca_6051	çatlak, yatay desen	2 çatlak, 18 yatay desen	Çatlak	Çatlak

ca_6053	çatlak, yatay desen	8 yatay desen	**Çatlak**	**CC**
ca_6055	çatlak	2 çatlak	Çatlak	Çatlak
ca_6062	çatlak	1 çatlak	Çatlak	Çatlak
ca_6092	çatlak	15 çatlak	Çatlak	Çatlak
ca_6095	çatlak	5 çatlak, 1 yatay desen, 1 budak	Çatlak	Çatlak
ca_6097	çatlak	3 budak	**Çatlak**	**BB**
ca_6098	çatlak, budak	7 çatlak, 1 budak	Çatlak	Çatlak
ca_6114	çatlak	1 yatay desen	**Çatlak**	**AA**
ca_6178	çatlak	1 çatlak	Çatlak	Çatlak
ca_6182	çatlak, budak	2 çatlak, 2 yatay desen, 2 budak	Çatlak	Çatlak
ca_6215	çatlak, budak	4 çatlak, 1 budak	Çatlak	Çatlak
ca_6216	çatlak	10 çatlak	Çatlak	Çatlak
ca_6274	çatlak, budak	11 çatlak, 10 yatay desen,2 budak	Çatlak	Çatlak
ca_6275	çatlak	18 çatlak, 1 yatay desen, 1 budak	Çatlak	Çatlak
ca_6276	çatlak	6 çatlak	Çatlak	Çatlak
ca_6280	çatlak	14 çatlak, 1 yatay desen	Çatlak	Çatlak
ca_6281	çatlak	10 çatlak	Çatlak	Çatlak
ca_6288	çatlak	4 çatlak, 1 yatay desen	Çatlak	Çatlak
ca_6289	çatlak, yatay desen	2 çatlak, 8 yatay desen	Çatlak	Çatlak
ca_6290	çatlak	8 çatlak, 3 yatay desen	Çatlak	Çatlak
ca_6295	çatlak, budak	4 çatlak, 1 budak	Çatlak	Çatlak

Çizelge 5.12. (devam ediyor)

Çatlak sınıfına ait test kümesi içerisinde 33 adet görsel bulunmaktadır. Bu görsellerdeki çatlak, budak, yatay desen nesnelerinin tamamı karışık bir şekilde yer almaktadır. ca_6053 numaralı görüntüde çatlak nesnesi tespit edilememiş yerine 8 adet yatay desen tespit edildiğinden algoritma görüntüyü CC sınıfına ayırmıştır. Ca_6097 görüntüsü üzerinde çatlak yerine budak tespiti gerçekleşmiş bu sebepten BB sınıfına ayrılmıştır. ca_6114 numaralı görselde 1 yatay desen tespiti gerçekleştiğinden algoritma görüntüyü AA sınıfına ayırmıştır. 33 adet görüntüden 30'u doğru 3'ü yanlış sınıflandırılmıştır. Çizelge 5.12'de bu bilgilere ait detaylar yer almaktadır. Çatlak sınıfına ait başarı oranı %90,90 olarak belirlenmiştir. Çizelge 5.13'te ise sınıflara ait başarı değerlerine yer verilmiştir.

Çizelge 5.13. Tüm sınıflara ve genel başarıya ait yüzdelik dilimler

Nihai Sınıflar	**Başarı Oranları**
AA	%90,90
BB	%100
CC	%81,81
Çatlak	%90,90
Genel Başarı	%90,90

BÖLÜM 6

SONUÇLAR VE ÖNERİLER

Bu çalışma kapsamında masif panel üretiminde kullanılan lamel parçaları üzerinde nesne tespiti ve sınıflandırılması işlemi gerçekleştirilmiştir. Ağaç cinsi olarak kayın ağacı kullanılmıştır. Segmentasyon aşamasında Mask R-CNN ağı tercih edilmiştir. Diğer derin öğrenme ağlarının yaptığı işlemin yanı sıra maskeleme işlemi ile tespit edilmesi istenen nesne arka plan görüntüsünden segmente edilmektedir. Modeli eğitirken Mask R-CNN ağı kullanılmış, bunun yanı sıra Python dilinde kodlama esnasında Keras ve TensorFlow kütüphanelerinden yararlanılmıştır. Görüntüler toplanırken FujiFilm X-S1 12MP kamera kullanılmıştır. Kameranın piksel yoğunluğu 326 dpi, ekran çözünürlüğü ise 783 x 587'dir. Modeller Windows 10 Pro işletim sistemli Quadro RTX 5000 ekran kartına sahip bilgisayar kullanılarak eğitilmiştir. Mask R-CNN modelinde kullanılan omurga ağı ise ResNet-101 olarak belirlenmiştir. Çatlak, budak ve yatay desen nesnelerine ait model eğitimi gerçekleştirilmiştir. Bu nesnelerin görsel üzerindeki konumunun, büyüklüğünün ya da adedinin bir önemi yoktur. Lamel üzerinde yer alan 3 çatlaktan 1 adedini tespit ettiği durumda bile nihai sınıflandırmada doğru sonuca varılmaktadır. Bu sebepten modellere ait başarım metrikleri hassasiyet, duyarlılık, f1-skoru ve doğruluk olarak belirlenmiştir. Bu aşamada metriklerin değerlendirilmesi konusunda çatlak var yok, budak var yok, yatay desen var yok şeklinde bir sınıflandırılmaya gidilmiştir. Bu sınıflandırma sonucunda Mask R-CNN ağında eğitilen en iyi budak modelinin doğruluk değeri %95,83, çatlak modelinin doğruluk değeri %85, yatay desen modelinin doğruluk değeri %76,66 olarak ölçümlenmiştir.

Bu aşamadan sonra tespit edilen nesnelere göre nihai sınıflandırma gerçekleştirilmiştir. Bu sınıflandırma gruplarında AA sınıfında lamel üzerinde herhangi bir nesnenin yer almaması, BB sınıfında yalnızca budak nesnesinin yer alması beklenmektedir. CC sınıfında yatay desen nesnesi kesinlikle yer almalı ve ek

olarak budak nesnesi de yer alabilir. Çatlak sınıfında ise çatlak nesnesi kesinlikle yer almalı bunun haricinde budak ve yatay desen de yer alabilir. Nihai sınıflandırmada başarı oranını artırmak için tespit edilen yatay desen nesne sayısı 6'ya eşit ya da daha küçük ise sınıflandırmada görmezden gelinecektir. Yatay desen nesnesi normalde lamel parçası üzerinde çoklu sayıda yer alır. Bu kural, bu bilgiden üretilen sınıflandırma performansını artırıcı bir süreçtir. Tüm bu bilgiler göz önünde bulundurularak yapılan nihai sınıflandırmada AA sınıfına ait başarı oranı %90,90, BB sınıfına ait başarı oranı %100, CC sınıfına ait başarı oranı %81,81, çatlak sınıfına ait başarı oranı %90,90 olarak ölçülmüştür. Genel sınıflandırmada elde edilen başarı oranı ise %90,90'dır.

Nihai sınıflandırma sonucunca budak modeline ait doğruluk değeri %95,83'ten %100'e yükseltilmiştir. Çatlak modeline ait doğruluk değeri %85'ten %90,90'a yükseltilmiştir. Yatay desen modeline ait doğruluk değeri %76,66'dan %81,81'e yükseltilmiştir.

Lamel görüntüleri üzerinde budak yapısının oluştuğu yerler genelde ağacın renginden daha koyu olmaktadır. Bazı koyuluklar ise budak olmamasına rağmen lamelin belirli kısımlarında oluşmaktadır. Budak modeli bu koyu halkaları da budak olarak tespit etmiştir. Budak modeli yüksek oranda başarı elde etmiştir. Çünkü budak olmamasına rağmen budak olarak işaretlediği yapılar, gerçek budağa aşırı derecede benzemektedir. Üretim esnasında bazı lamel parçalarında enine testere diş izi oluşmaktadır. Yatay desen modeli bu izleri yatay desen olarak işaretlemiştir.

Çatlak modeli boyuna olan koyu çizgileri çatlak olarak algılamaktadır. Genelde boyuna olan koyu çizgilerin içerisinde çatlak etiketi yer aldığından model boyuna koyu çizgide çatlak yer almasa bile çatlak olarak işaretlemektedir. Ayrıca ağacın deseni şeklinde olan koyu çizgilerde de model çatlak tespiti gerçekleştirmektedir.

Gelecek çalışmalarda çatlak, yatay desen ve budak modeline ait kusurlu bulunan yapıların iyileştirilmesi sağlanarak doğruluğun daha da artırılması hedeflenmektedir. Ayrıca gerçek hayattaki lamel parçalarının sınıflandırılması probleminde ele alınan diğer yapıların da derin öğrenme ağlarına tanıtımı ve tespiti ile ilgili çalışılacaktır. Bu

çalışma kapsamında yapraklı ağaç türlerinden kayın ağacı ile çalışılmıştır. Bunun haricinde masif panel üretiminde kullanılan yapraklı, iğne yapraklı ve tropik ağaç türlerinin de derin öğrenme ağları yardımıyla üzerindeki yapıların tespiti ve sınıflandırılması işleminin yapılması planlanmıştır.

KAYNAKLAR

1. Hatipoğlu, M. "Makroskopik görünüşleri ve mikroskopik yapılarına göre doğal fosilleşmiş ağaçların (SiO2) sınıflandırılması; Güdül-Ankara bölgesi fosil ağaç ormanı örneği", ***International Journal of Interdisciplinary and Intercultural Art***, 3 (4) : 109–127 (2018).

2. Mengeloğlu, F. ve Kurt, R., "Mühendislik ürünü ağaç malzemeler 1 tabakalanmış kaplama kereste (TAK) ve tabakalanmış ağaç malzeme (TAM)", ***KSÜ Fen ve Mühendislik Dergisi***, 7 (1) : 39-44 (2004).

3. İlkuçar, M, Kaya, A. İ. ve Çifci, A., "Mekanik özelliklere göre ağaç türlerinin yapay sinir ağları ile tahmini" ***GÜFBED/GUSTIJ,*** 8 (1): 75-83 (2018).

4. Donatello, S.,Cordella, M., Kaps R., Kowalska, M. and Wolf, O., "Are the existing eu ecolabel criteria for furniture products too complex? An analysis of complexity from a material and a supply chain perspective and suggestions for ways ahead", ***The International Journal of Life Cycle Assessment,*** (2019).

5. Bilgin, Y., "Türkiye'de masif panel sektörünün yapısal durumu ve ağaç işleri endüstrisindeki kullanım olanakları" , Yüksek Lisans Tezi, ***İstanbul Üniversitesi Fen Bilimleri Enstitüsü***, İstanbul, 1-123 (2010).

6. İnternet : Weining Dimter Youtube, "Weining Otomatik Budak Ayırma Makinesi", **https://www.youtube.com/watch?v=Z14xySOT6Rk** (2022).

7. İnternet : Weining, "Weining Ultra TT 1000 : Ultra Finger Jointing Line", **https://www.weinig.co.uk/en/products/finger-jointing-lines/short-wood-lines/ultra-ultra-tt-1000.htm** (2022).

8. Chen, L.C., Pardeshi, M.S., Lo, W.T., Sheu, R.K., Pai, K.C., Chen, C.Y., Tsai, P.Y., and Tsai, Y.T., "Edge-glued wooden panel defect detection using deep learning" ***Wood Science and Technology***, 56: 477-507(2022).

9. Xi, Z., Hopkinson, C., Rood, S., and Peddle, D., "See the forest and the trees: effective machine and deep learning algorithms for wood filtering and tree species classification from terrestrial laser scanning", ***ISPRS Journal of Photogrammetry and Remote Sensing***, 168 : 1-6 (2020).

10. Kılıçarslan, Ş., Türker, Y. ve İnce, M., "Temas açısı değerlerine bağlı ağaç türlerinin farklı sınıflandırma yöntemleri ile tahmini", ***Bartın Orman Fakültesi Dergisi***, 22 (3): 861-870 (2020).

11. Fabijanska, A., Danek, M. and Barniak, J., "Wood species automatic identification from wood core images with a residual convolutional neural network" ***Computer and Electronics in Agriculture,*** 181 (2021).

12. As, N., Dündar, T. ve Büyüksarı, Ü., "Türkiye'de yetişen ağaç türlerinin bazı fiziko-mekanik özellikleri bakımından sınıflandırılması", ***Journal of the Faculty of Forestry Istanbul University,*** 66 (2) : 727-735 (2016).

13. Öktem, M., Akosman, Ş., Moral, Ö. ve Kılıç, V., "Derin öğrenme tabanlı mermer yüzeylerinin otomatik sınıflandırılması", ***Avrupa Bilim Sanayi ve Teknoloji Dergisi,*** 26 : 73-77 (2021).

14. Gurkan, C. ve Palandoken, M., "SMARfacTory-Net:mermerin sınıflandırılması için bilgisayarlı görü, qr kod ve android tabanlı teknolojilerle desteklenen sistem tasarımının geliştirilmesi", ***Journal of Computer Science,*** 347-358 (2021).

15. İnternet : Make Sense Web Sayfası, "Make Sense", ***https://www.makesense.ai/*** (2022).

16. Özel M., Baysal S. ve Şahin M., "Derin öğrenme algoritması (YOLO) ile dinamik test süresince süspansiyon parçalarında çatlak tespiti", ***Avrupa Bilim ve Teknoloji Dergisi***, 26 : 1-5 (2021).

17. Kılıç, Ö., Susuz, D. ve Süzek, B., "A quality control system prototype for detecting knot defects in the wooden panel manufacturing", ***Mugla Journal of Science and Technology***, (2018).

18. Wang, Z., Zhuang, Z., Liu, Y., Ding, F. and Tang, M., "Color classification and texture recognition system of solid wood panels", ***Forests*** (2021).

19. Zhuang, Z., Liu, Y., Ding, F. and Wang, Z., "Online color classification system of solid wood flooring based on characteristic features", ***Sensors,*** 2-13 (2021).

20. Yang, Y., Zhou, X., Liu, Y., Hu, Z. and Ding, F., "Wood defect detection based on depth extreme learning machine", ***Applied Sciences***, 2-14 (2020).

21. Gao, M., Qi, D., Mu, H. and Chen, J., "A transfer residual neural network based on ResNet-34 for detection of wood knot defects", ***Forests,*** 12 (212) : 2-15 (2021).

22. Wang, B., Yang, C., Ding, Y. and Qin, G., "Detection of Wood Surface Defects Based on Improved YOLOv3 Algorithm" , ***BioResources,*** 16 (4) : 6766-6780 (2021).

23. Yang, Y., Wang, H., Jiang, D. and Hu, Z., "Surface detection of solid wood defects based on SSD improved with ResNet", ***Forests***, 12 (149) : 2-11 (2021).

24. Ding, F., Zhuang, Z., Liu, Y., Jiang, D., Yan, X. and Wang, Z., "Detecting defects on solid wood panels based on an improved SSD algorithm", ***Sensors***, 20 (5315) : 2-17 (2020).

25. Mohan, S. and Venkatachalapathy, K., "Wood knot classification using bagging", ***International Journal of Computer Applications,*** 51 (18) : 50-53 (2012).

26. Yang, J., Xiao, J., Fu, W., Chen, J., Yan, L. and Wang, S., "Image processing and identification of lumber surface knots", ***Acta Technica***, 1A (62) : 99-108 (2017).

27. Augustauskas, R., Lipnickas, A. and Surgailis, T., "Segmentation of drilled holes in texture wooden furniture panels using deep neural network", ***Sensors,*** 21 (3633) : 2-26 (2021).

28. Decelle, R. and Jalilian, E., "Neural networks for cross-section segmentation in raw images of log ends", ***HAL,*** (2020).

29. Valueva, M. V., Nagornov, N. N., Lyakhov, P. A., Valuev, G. V., and Chervyakov, N. I., "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation", ***Mathematics and Computers in Simulation***, *177*, 232-243 (2020).

30. Ataseven, B., "Yapay sinir ağları ile öngörü modellemesi", ***Dergi Park,*** 101 (115) : 101-115 (2013).

31. Öztürk, K. ve Şahin, M., "Yapay sinir ağları ve yapay zekaya genel bir bakış", ***Takvim-i Vekayi,*** 6 (2) : 25-36 (2018).

32. Singh, M.K., Baluja, G.S. and Sahu, D.P., "Understading the convolutional neural network it's research aspects in deep learning", ", ***International Journal for Research in Applied Science & Engineering Technology,*** (IJRASET), 5 (5) : 867-871 (2017).

33. Zhang, W., Itoh, K., Tanida, J. and Ichioka, Y., "Parallel distributed model with local space-invariant interconnections and its optical architecture", ***Optica Publishing Group***, 29 : 4790-4797 (1990).

34. Girshick, R., Donahue, J., Darrell, T., and Malik, J., "Rich feature hierarchies for accurate object detection and semantic segmentation", ***In Proceedings of the***

IEEE Conference on Computer Vision and Pattern Recognition pp. 580-587 (2014).

35. Yılmaz, O., Aydın, H. ve Çetinkaya, A., "Faster R-CNN evrişimsel sinir ağı üzerinde geliştirilen modelin derin öğrenme yöntemleri ile doğruluk tahmini ve analizi: nesne tespiti uygulaması", ***Avrupa Bilim ve Teknoloji Dergisi,*** 783-795 (2020).

36. İnternet: The PASCAL Visual Object Classes Homepage, "Pascal VOC Challenges", **http://host.robots.ox.ac.uk/pascal/VOC/** (2022).

37. İnternet: ILSVRC2013, "ImageNet Large Scale Visual Recognition Challenge 2013", **https://image-net.org/challenges/LSVRC/2013/** (2022)

38. Girshick, R., "Fast r-cnn***", In Proceedings of the IEEE International Conference on Computer Vision***, 1440-1448 (2015).

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

40. İnternet: Fast R-CNN ile İlgili Teknoloji Web Sayfası, "Fast R-CNN Nedir?" **https://teknoloji.org/**, (2022)

41. Ren, S., He, K., Girshick R., and Sun J., "Faster r-cnn: Towards real-time object detection with region proposal networks", ***Advances in Neural Information Processing Systems***, *28*, 91-99 (2015).

42. İnternet: Tryolabs Machine Learning, "Faster R-CNN: Down the rabbit hole of modern object detection", **https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/** (2021).

43. Cai, Z., and Vasconcelos, N., "Cascade r-cnn: High quality object detection and instance segmentation", ***IEEE Transactions on Pattern Analysis and Machine Intelligence*** (2019).

44. İnternet: Comparative Guide, "R-CNN vs Fast R-CNN vs Faster R-CNN", **https://analyticsindiamag.com/r-cnn-vs-fast-r-cnn-vs-faster-r-cnn-a-comparative-guide/,** (2022).

45. He, K., Gkioxari, G., Dollár, P., and Girshick, R., "Mask r-cnn", ***In Proceedings of the IEEE international conference on computer vision***, pp. 2961-2969 (2017).

46. Challapalli, P., Teja, C.S.B., Chandra, P.N., Revanth, V. and Babu, B.S., "Implementation of machine learning techniques for defect detection in real-time", ***International Research Journal of Engineering and Technology (IRJET),*** 07 (10) : 409-414 (2020)

47. Lin, T. Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L. and Dollár, P., "Microsoft coco: Common objects in context", ***In European Conference on Computer Vision***, pp. 740-755. Springer, Cham (2014).

48. Albayrak, E., Yayla R. ve Yüzgeç U., "Mask R-CNN ile iha görüntülerinden araç tespiti", ***International Symposium of Scientific Research and Innovative Studies***, 326-335 (2021).

49. He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition", ***In Proceedings of the IEEE Conference On Computer Vision and Pattern Recognition*** pp. 770-778 (2016).

50. L. Deng and D. Yu, "Deep Learning: Methods and Applications", ***Foundations and Trends in Signal Processing***, vol. 7, no. 3–4, pp. 197–387 (2014).

51. İnternet: Keras Web Sitesi, "Keras", **https://keras.io** (2022).

52. Gulli, A., and Pal, S., "Deep learning with Keras", ***Packt Publishing Ltd*** (2017)

53. Sivalingam, K., and Mujkanovic, N., "Graph compilers for AI training and inference" (2019)

54. Etaati, L., "Deep Learning Tools with Cognitive Toolkit (CNTK)", ***In Machine Learning with Microsoft Technologies***, pp. 287-302 Apress, Berkeley, CA (2019).

55. Varouqaux, G., Walt, S.J. and Millman, K.J., "Python for scientific computing", ***Proccedings of the eighth Python in Science Conference***, (2009)

56. Chollet, F., "*Python ile Derin Öğrenme"*, Birol Kuyumcu, ***Buzdağı Yayınevi***, 978-605-69024-2-0 (2019)

57. İnternet: TensorFlow Web Sitesi, "TensorFlow", **https://www.tensorflow.org** (2022)

58. İnternet: Kaggle Web Sitesi, "Kaggle", **https://www.kaggle.com** (2022).

59. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., and Zheng, X., "Tensorflow: A system for large-scale machine learning", ***In 12th {USENIX} symposium on operating systems design and implementation***, pp. 265-283 (2016).

60. Kalaiselvi, T., Sriramakrishnan, P., and Somasundaram, K., "Survey of using GPU CUDA programming model in medical image analysis", ***Informatics in Medicine Unlocked,*** 133-144 (2017)

61. Kirk, D., "NVIDIA Cuda Software and Gpu Parallel Computing Architecture" ***Proceedings of the 6th international symposium on Memory management***, (2007)

62. İnternet: Cuda Web Sitesi, "Cuda Paralleşmesi ve İş Parçacıkları", ***https://docs.truba.gov.tr/education/cuda/02_cuda_c_giris/02_03.html*** (2022)

63. İnternet: NVIDIA Web Sitesi, "NVIDIA", **https://www.nvidia.com/en-us/,** (2022)

64. İnternet : NVIDIA Web Sitesi, "NVIDIA cuDNN", **https://developer.nvidia.com/cudnn *(***2022)

65. İnternet : NVIDIA Web Sitesi "Your GPU Compute Capability", **https://developer.nvidia.com/cuda-gpus** (2022)

66. İnternet : Anaconda Web Sayfası, "Anaconda Kurulum Seçenekleri" **https://www.anaconda.com/** (2022)

67. İnternet : TensorFlow Web Sitesi, "TensorFlow Uygun Fiziksel Cihazların Sorgulanması", **https://www.tensorflow.org/api_docs/python/tf/config/list_physical_devices** (2022)

68. İnternet : Matplotlib Web Sitesi "Matplotlib Kütüphanesi", **https://matplotlib.org/,** (2022)

69. İnternet : Kaggle Web Sitesi "ResNet-101", **https://www.kaggle.com/datasets/pytorch/resnet101,** (2022)

70. Garg, D., Goel, P., Pandya, S., Ganatra, S. and Kotecha, A., "A deep learning approach for face detection using YOLO", ***IEEE Punecon***, (2018)

71. Chen, J., Xie, M., Xing, Z., Chen, C., Xu, X., Zhu, L., and Li, G., "Object detection for graphical user interface: old fashioned or deep learning or a combination?", ***ResearchGate***, 8-13, (2020)

72. Ren, S., He, K., Girshick, R. and Sun, J., "Faster R-CNN: towards ral-time object detection with region proposal networks", ***Computer Vision and Pattern Recognition,*** (2015)

73. İnternet: Wiki, "Feature Pyramid Networks", ***https://wikidocs.net/162976, (2022)***

74. Nguyen, V., Dang T. and Jin F.," Predict Saturated Thickness using TensorBoard Visualization", ***Workshop on Visualisation in Environmental Sciences,*** (2018)

75. Kıyıkçı, F., Cunedioğlu H., Koşar E., Bekin M., Abut F. and Akay M., "Assessing household damages using multi-model deep learning pipeline", ***European Mechanical Science,*** 2587-1110 (2022)

ÖZGEÇMİŞ

Merve ÖZKAN, ilk ve orta öğrenimini aynı şehirde tamamladı. Kastamonu'da yer alan Saime İnal Savi Anadolu Lisesi'nden mezun oldu. 2012 yılında Yalova Üniversitesi Bilgisayar Mühendisliği Bölümü'nde öğrenime başlayıp 2017 yılında mezun oldu. 2021 Ocak ayında Karabük Üniversitesi Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda başlamış olduğu yüksek lisans programında eğitimini sürdürmektedir. 2018 yılında Kastamonu Ticaret Borsası Ağaç ve Orman Ürünleri San. A.Ş. şirketinde Yazılım ve Veri Tabanı Uzmanı olarak göreve başlamıştır. 4 yıldır halen aynı yerde çalışmaya devam etmektedir. Aynı zamanda Monybyte şirketinde Yazılım Geliştiricisi ünvanıyla görev almaktadır.