



Teknoloji Fakültesi

MARMARA ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME PROJESİ

Görüntü İşleme Tekniği İle Satranç Oyun Hamlesi Tespiti

PROJE YAZARI

Emine Yiğit -100919015

Aslı Cennet Ercan -170421040

Yusuf Doğan -170421853

DANIŞMAN

Doç. Dr. ÖNDER DEMİR

İstanbul , 2025



Teknoloji Fakültesi

MARMARA ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME PROJESİ

Görüntü İşleme Tekniği İle Satranç Oyun Hamlesi Tespiti

PROJE YAZARI

Emine Yiğit -100919015

Aslı Cennet Ercan -170421040

Yusuf Doğan -170421853

DANIŞMAN

Doç. Dr. ÖNDER DEMİR

İstanbul , 2025

MARMARA ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Marmara Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Öğrencileri Emine Yiğit, Yusuf Doğan, Aslı Cennet Ercan' nın “Görüntü İşleme Tekniği İle Satranç Oyun Hamlesi Tespiti ” başlıklı bitirme projesi çalışması, 19/06/2025 tarihinde sunulmuş ve jüri üyeleri tarafından başarılı bulunmuştur.

Jüri Üyeleri

DOÇ. Dr. ÖNDER DEMİR (Danışman)

Marmara Üniversitesi (İMZA)

Arş. Gör. Dr. ABDULSAMET AKTAŞ (Üye)

Marmara Üniversitesi (İMZA)

İçindekiler

KISALTMALAR/ABBREVIATIONS	6
ŞEKİL LİSTESİ	7
TABLO LİSTESİ	8
ÖZET	9
1.GİRİŞ	10
1.1 Arka Plan Bilgisi	10
1.2 Problem, Amaç, Yaklaşım	10
1.3 Proje Kapsamı ve Organizasyonu	11
1.4 Proje Takvimi	12
2. Literatür Taraması	12
2.2 Nesne Algılama ve YOLO Kullanımı	13
2.3 FEN Formatı Kullanımı ve Satranç Motorları.....	13
2.4 Özgünlük ve Farklılıklar	13
3. Problem Tanımı.....	14
3.1 Problemin Açıklaması.....	14
3.2 Ortam Tanımı ve Varsayımlar.....	15
3.3 Proje Gereksinimleri ve Sınırlamalar	17
3.3.1 Özel Taşlar ve Tahta Tasarımı Bağımlılığı.....	18
3.3.2 Kamera Açısı ve Işık Koşullarının Etkisi.....	18
3.3.3 Gerçek Zamanlılık ve Hesaplama Performansı.....	19
4. METOT (YÖNTEM)	20
4.1 Sistem Mimarisi.....	20
4.2. Satranç Tahtasını Algılama Katmanı.....	20
4.3. Parçaların Tespiti.....	21
4.4. FEN Üretimi Katmanı	23
4.5. Hamle Önerisi Katmanı (Stockfish Entegrasyonu).....	25
4.6. Grafiksel Kullanıcı Arayüzü (GUI)	27
4.6.1 GUI Tasarımı ve Kullanıcı Etkileşimi	28
5. Uygulama Ortamı.....	29
5.1 Yazılım ve Kullanılan Kütüphaneler:	30
5.2 Geliştirme Süreci	31
5.2.1 Veri Toplama ve Etiketleme:.....	31
5.2.2 Veri Artırımı (Augmentation):.....	32
5.2.3 YOLOv8 Model Eğitimi:	32
5.2.4 GUI Geliştirme:	34

5.2.5 Stockfish Entegrasyonu:	34
5.3 Test Senaryoları ve Uygulama Adımları.....	36
5.3.1 Sabit Açık Testi.....	36
5.3.2 Işık Koşullarına Dayanıklılık Testi.....	36
5.3.3 Taş Eksikliği ve Yanlış Konum Testi.....	37
6.SONUÇ 6.1 Sistem Değerlendirmesi ve Başarılar.....	37
6.1.1 Ulaşılan Hedefler	37
6.1.2 Algılama Başarısı.....	38
6.1.3 FEN Üretimi ve Doğrulama Başarısı.....	38
6.1.4 Hamle Önerme Başarısı.....	38
6.2 Mevcut Sistem ile Karşılaştırma	38
6.3.2 Model Optimizasyonu ve Veri İşleme.....	40
7. GELECEK ÇALIŞMALAR İÇİN ÖNERİLER.....	40
7.1 Farklı Satranç Setleriyle Çalışabilen Genel Model	40
7.2 Mobil Cihazlarla Entegre Çalışma	41
7.3 LLM Destekli Pozisyon Analizi.....	41
8.KAYNAKLAR.....	42

KISALTMALAR/ABBREVIATIONS

- GUI → Grafiksel Kullanıcı Arayüzü
- YOLO → You Only Look Once (Sadece Bir Kez Bak)
- FEN → Forsyth Edwards Notasyonu
- API → Uygulama Programlama Arayüzü
- LLM → Büyük Dil Modeli
- GPU → Grafik İşlem Birimi
- CPU → Merkezi İşlem Birimi
- UCI → Evrensel Satranç Arayüzü
- SVG → Ölçeklenebilir Vektör Grafiği
- FPS → Saniye Başına Kare
- mAP→Mean Average Precision (mAP)

ŞEKİL LİSTESİ

Şekil.3.2: *Tasarlanan oyun izleme düzeneği*

Şekil.4.1: Önerilen sistemin iş akış şeması

Şekil.4.3: *Satranç Tahtasında Taşların Algılanması*

Şekil.4.4: *FEN ve Oyun Durumuna Bir Örnek*

Şekil.5.2.1: *Satranç Tahtasında İşaretlenen Taşlar*

Şekil.5.2.3: *Satranç Tahtasında Algılanan Köşeler*

Şekil.5.3: *Satranç Tahtasında Taşların Algılanması*

Şekil 5.4: *Satranç Taşı Algılama Modelinin Performans Ölçümleri*

Şekil 5.5: *Oyunun Arayüze Aktarımına ve Taş Konumlarının Ekrana Yazılmasına Bir Örnek*

TABLO LİSTESİ

Tablo.1.4.Proje Takvimi Açıklaması

Tablo.4.3.Sınıf Etiketlerinin Satranç Taşı Notasyonlarına Eşlenmesi

ÖZET

Chess Player Helper projesi, satranç oyunu esnasında satranç oyuncularının hamlelerini kamera yardımı ile gerçek zamanlı olarak taş ve taş konumlarını tespit edip analiz etmeyi amaçlayan bir destek sistemi olarak tasarlanmıştır. Bu proje kapsamında kamera görüntüsünden satranç taşlarının pozisyonları YOLO (You Only Look Once) nesne algılama modeli kullanılarak belirlenirken, OpenCV kütüphanesi ile görüntü ön işleme gerçekleştirilmiştir. Böylece, 8×8 kareli satranç tahtasındaki her bir hücrenin güncel durumu doğru bir şekilde çıkarılmıştır. Elde edilen taş konum verileri belirlendikten sonra, Stockfish satranç motoruna aktararak hamle öneri ve analiz mekanizması devreye sokulmuş; bu sayede oyuncuya en uygun hamle alternatifleri gerçek zamanlı olarak sunulmuştur. Kullanıcı dostu GUI (Graphical User Interface) tasarımı ise PyQt5 kütüphanesi ile geliştirilmiştir.

Projede çözülmek istenen temel problem, fiziksel satranç tahtlarında gerçekleşen hamlelerin anlık olarak dijital ortama aktarılması ve bu hamlelerin profesyonel bir satranç motoru aracılığıyla derinlemesine analiz edilmesidir. Geleneksel yöntemlerde, bir hamlenin fiziksel olarak not edilmesi ile yapılmakta olup bu durum oyun esnasında gecikmeye neden olmaktadır. Oysa Chess Player Helper, YOLO'nun hızlı algılama performansı ve OpenCV optimizasyonları sayesinde gerçek zamanlı izleme yapılarak anlık olarak tespit edilmesine olanak sağlamaktadır. Stockfish entegrasyonu ise hamle değerlendirme ve öneri sürecini hızlı şekilde tamamlayarak oyuncuya kesintisiz bir deneyim ve oyun esnasında analiz imkanı sunmaktadır. Chess Player Helper projesi esnasında kullanılan terimler ve teknolojiler aşağıda yer almaktadır.

Hamle Tespiti, kamera görüntüsünden taşların oyun tahtası üzerindeki pozisyonları tespit edilmesini sağlamaktadır. Bu aşamada YOLO (You Only Look Once) ise ,nesne algılamada tek aşamalı, hızlı ve yüksek doğruluklu derin öğrenme tabanlı bir model ailesi olarak satranç taşlarını tek karede gerçek zamanlı olarak tespit etmek ve sınıflandırmak için kullanılır. Gerekli taş tespiti yapıldıktan sonra OpenCV , görüntü düzeltme (perspektif transform), 8x8 grid oluşturma ve kare bölümlendirme işlemleriyle ham veri hazırlama hattını oluşturur. Ardından Stockfish ile ,açık kaynaklı olan ve yaygın olarak kullanılan satranç motoru; hamleleri derinlemesine analiz eder ve tespit edilen taş konumlarına dayanarak o andaki en iyi hamleleri hesaplar. Bu hesaplanan hamleler GUI (Graphical User Interface) , PyQt5 ile hazırlanmış, hamle tespiti ve Stockfish hamle tahminlerinin gösterildiği etkileşimli masaüstü arayüz uygulaması ile birlikte kullanıcıya sunulur.

1.GİRİŞ

Chess Player Helper projesinin detaylı olarak arka planında nasıl çalıştığı , proje kapsamı ve proje organizasyonu belirtilmektedir. Satranç oyununun genel özellikleri ve fiziksel oyunlarda gerçek zamanlı analiz eksikliği ile birlikte bulunan problemler, proje amacı olarak bu problemlerin çözümü net bir şekilde ifade edilmiştir.

1.1 Arka Plan Bilgisi

Satranç, strateji ve taktik üzerine kurulu iki oyunculu bir tahta oyunudur. Oyunun temel öğeleri, beyaz ve siyah taşlar ile 8x8 kareden oluşan bir satranç tahtasıdır. Her oyuncuya 16 taş (1 şah, 1 vezir, 2 kale, 2 at, 2 fil ve 8 piyon) dağıtılır ve amaç, rakibin şahını tehdit altında bırakarak mat etmektir. Günümüzde bilgisayar destekli analiz yöntemleri, satranç oyuncularının becerilerini geliştirmek, yeni açılış teorileri keşfetmek ve oyun sonu tekniklerini incelemek için yaygın şekilde kullanılmaktadır.

Fiziksel oyunlarda gerçek zamanlı analiz eksikliği Çevrimiçi platformlarda hamlelerin hemen kaydedilip güçlü motorlarla analiz edilebilmesi mümkün iken, fiziksel satranç maçlarında hamlelerin dijital ortama aktarılması genellikle manuel notasyon veya yayın sırasında insan gözetimi gerektirmektedir. Manuel notasyon, insan hatasına açıktır ve hamlelerin kaydedilme hızı sınırlı ve yavaştır. Bu durum beraberinde notasyon hatalarına ve oyuncunun performansına olumsuz bir durum olarak yansımaktadır.

1.2 Problem, Amaç, Yaklaşım

Problem:

Fiziksel satranç maçlarında oyuncular, gerçekleştirdikleri hamleleri manuel olarak not almak zorundadır. Bu durum, özellikle resmi turnuvalarda zorunlu olmasına rağmen, oyun akışını yavaşlatmakta ve dikkat dağınıklığına yol açabilmektedir. Ayrıca oyuncular, hamleler arasında uzun düşünme sürelerinde farklı stratejik varyantları değerlendirme konusunda zaman baskısı altında kalmakta ve potansiyel avantajları gözden kaçırabilmektedir.

Amaç:

Bu çalışmanın temel amacı, fiziksel bir satranç tahtasında gerçekleştirilen hamlelerin kamera destekli bir sistem ile gerçek zamanlı olarak algılanması ve dijital ortama aktarılmasıdır. Buna ek olarak, tespit edilen taş konumlarının güçlü bir satranç motoru olan Stockfish ile analiz edilerek oyuncuya en uygun hamle önerilerinin sunulması hedeflenmektedir. Bu sistem ile hem oyun kaydının otomatik olarak tutulması sağlanmakta hem de oyuncunun stratejik karar süreçlerine destek verilmektedir.

Yaklaşım:

Önerilen sistem dört ana teknolojik bileşen üzerine kuruludur:

- **Algılama:** Kamera görüntüleri üzerinde YOLO tabanlı derin öğrenme modeli kullanılarak taşların türü ve konumu belirlenmektedir.
- **Görüntü İşleme:** OpenCV kütüphanesi kullanılarak satranç tahtasının köşe koordinatları tespit edilmekte ve taşlar doğru karelerle eşleştirilmektedir.

- **Analiz:** Tespit edilen taş pozisyonları Stockfish motoruna gönderilerek en uygun hamle hesaplanmakta ve mevcut oyun durumuna göre değerlendirme yapılmaktadır.
- **Öneri ve Sunum:** Hamle önerileri PyQt5 tabanlı grafiksel kullanıcı arayüzü üzerinden kullanıcıya sunulmakta, aynı zamanda taş konumları gerçek zamanlı olarak görselleştirilmektedir.

Bu bütünsel yaklaşım sayesinde, satranç oyuncularının fiziksel oyun ortamında dijital destekle daha stratejik ve verimli kararlar almaları sağlanmaktadır.

1.3 Proje Kapsamı ve Organizasyonu

Chess Player Helper projesi fiziksel satranç tahtası ve taş seti üzerinde çalışabilen gerçek zamanlı algılama modülü, hamle analiz motoru (Stockfish)entegrasyonu, kullanıcı dostu masaüstü GUI ile birlikte proje kullanıcısı için simüle edilmiş oyun olarak sunmaktadır. Sistem farklı aydınlatma koşullarına ve farklı taş ve gölge durumlarına uyum sağlayacak şekilde veri seti ile eğitilmiş ve test edilmiştir.

Proje içerisinde oyuncu ve oyun odaklı bir yapı bulunmaktadır ve ağ üzerinden çoklu oyuncu desteği, çevrimiçi maç izleme entegrasyonu, mobil platformlara özgü uygulama desteği satranç problemleri veya özel varyant oyunları (Fischer Random, Üç Kişilik Satranç vb.) bu çalışmanın kapsamı dışındadır.

- Literatür Taraması ve Tasarım: Satranç algılama ve analiz yöntemlerinin incelenmesi, sistem mimarisinin planlanması.
- Veri Toplama ve Model Eğitimi: Farklı taş setleri ve aydınlatma koşullarında görüntü kayıtları alınarak YOLO eğitimi.
- Görsel Algılama Geliştirme: OpenCV ile ön işleme hattının ve perspektif dönüşüm işlemlerinin test edilmesi.
- Motor Entegrasyonu: Stockfish ile API bağlantısının kurulması ve veri alışveriş protokolünün tasarlanması.
- GUI Geliştirme: PyQt5 ile kullanıcı arayüzünün oluşturulması, modüller arası entegrasyonunun yapılması.
- Test ve Değerlendirme: Gerçek satranç maçları simülasyonu ile doğruluk ve gecikme ölçümlerinin yapılması.
- Rapor Yazımı: Bulguların detaylı sunumu, karşılaştırmalar ve gelecekteki çalışmalara yönelik öneriler.

1.4 Proje Takvimi

Aşama	Süre	Açıklama
Literatür Taraması	1.Hafta	Benzer sistemlerin incelenmesi
Sistem Tasarımı	2.-3. Hafta	Mimarinin ve alt modüllerin planlanması
Veri Toplama ve Etiketleme	4.–5. Hafta	Satranç tahtası ve taşlarının görüntülenmesi, etiketlenmesi
Model Eğitimi (YOLOv8)	6.–7. Hafta	Nesne tanıma modelinin eğitilmesi
Görüntü İşleme ve FEN	8.–9. Hafta	Perspektif düzeltme, taş eşleme, FEN üretimi
Stockfish Entegrasyonu	10. Hafta	Motor bağlantısı ve veri aktarımı
Grafik Arayüz (GUI)	11.–12.Hafta	Kullanıcıya hamle gösterimi ve etkileşim
Test ve Değerlendirme	13. Hafta	Farklı senaryolarda sistemin test edilmesi
Rapor Yazımı ve Düzenleme	14.–15. Hafta	Sonuçların yazılması, görsel ve belgelerin eklenmesi

Tablo.1.4 Proje Takvimi Açıklaması

2. Literatür Taraması

2.1 Satranç Tabanlı Bilgisayarla Görüntüleme Projeleri

Son yıllarda fiziksel satranç tahtı üzerinde hamleleri otomatik olarak tanıyan ve dijitalleştiren bilgisayarla görüntü sistemleri yaygın olarak kullanılmaktadır. ChessVision.ai, webcam veya sabit kameralar aracılığıyla tahtanın görüntüsünü işler, karelerin perspektifini düzelterek taşların koordinatlarını çıkarmaktadır [10]. YOLO veya benzeri nesne algılama modelleriyle taşları sınıflandırdıktan sonra, hamleleri FEN dizgesine dönüştürerek bir satranç motoruna aktarma imkânı sunmaktadır. Kullanıcı dostu arayüzüyle hem amatör oyunculara anında analiz hem de yayıncılar için canlı maç takibi sağlamaktadır.

Chess OCR yaklaşımı hamleleri tahtanın yanına yerleştirilmiş elektronik sensörlerden ya da kare altı matrislerine yerleştirilmiş optik karakter okuma (OCR) bileşenlerinden okumaktadır[11]. Bu yöntemde “a4-e4” tarzı yazılı hamle notasyonunu tarayıp dijitalleştirerek sisteme iletir; ancak tahtanın gerçek görüntüsünü anlamadığı için satranç seti görünümündeki problemler karşısında yanlış algılama sonucunda yanlış kararlar vermektedir.

ChessBotX gibi projeler, fiziksel bir robot kol aracılığıyla taşları hem tespit eder hem de hareket ettirir [12]. Görüntü işleme kısmında OpenCV tabanlı kare tespiti ve renk ayrıştırma kullanılırken, eklenmiş robot bileşeni, analizi kendine entegre edilmiş bir satranç motorunun en iyi hamlesini fiziksel olarak uygulamaktadır. Bu sayede satranç tahtası, tamamen otonom bir rakip haline gelir.

Bu projelerin ortak hedefi; fiziksel satranç deneyimini dijitalleştirerek oyuna yeni etkileşim katmanları eklemektir. Analiz, hamle kaydı, uzaktan yayın ve otonom oynatma gibi farklı ihtiyaçları karşılamayı amaçlayan her çalışma, algılama ve sınıflandırma doğruluğu ile gecikme süresi dengesi problemini farklı yöntemlerle çözmüştür. Chess Player Helper projesi ise benzer prensipleri gerçek zamanlı hamle tespitine odaklanarak analiz sağlar; bu sayede hem oyuncunun hem de izleyicinin anlık geri bildirim alabileceği bütüncül bir çözüm sunar.

2.2 Nesne Algılama ve YOLO Kullanımı

Nesne algılama, bir görüntüdeki tüm nesnelerin konum ve sınıf bilgisini aynı anda veren bir bilgisayarla görüntüleme sistemidir. Tek aşamalı yaklaşımlardan biri olan YOLO (You Only Look Once), görüntüyü sabit boyutlu hücrelere böler ve her hücreden eşzamanlı olarak sınıflandırma ile sınır kutusu (bounding box) tahmini çıkartır. Bu sayede yüksek hız ve makul doğruluk oranı sağlar.

YOLOv8 derin ve optimize edilmiş ağ yapısı kullanmaktadır; aktarım öğrenme (transfer learning) ile küçük veri setlerinde bile düşük hata ile eğitilebilir [15]. Chess Player Helper’da, taş setleri üzerinde toplanmış etiketli görüntüler ile birlikte YOLOv8 tabanlı bir model eğitilmiştir. Eğitilen model, taşı renk ve tip (örneğin “beyaz vezir = Q, siyah piyon =p”) olarak sınıflandırırken birbirinin üzerini örtebilecek senaryoları bile doğru tespit edebilmektedir.

Roboflow gibi platformlar, veri etiketleme sürecini hızlandırır ve model için otomatik augmentasyon (döndürme, parlaklık, keskinlik varyasyonu) imkânı sunar [13]. Chess Player Helper eğitim hattında Roboflow kullanılarak; zayıf aydınlatma, gölge ve taşların birbirinin görüntüsünü engelleme durumu gibi gerçek dünya koşullarına karşı dayanıklılığı artırılmıştır. YOLO ile taş tespiti sayesinde, algoritmik olarak her karedeki taş yerleşimi anlık olarak çıkarılır ve sonraki adıma (FEN oluşturma, motor sorgusu) hazır hâle gelir.

2.3 FEN Formatı Kullanımı ve Satranç Motorları

Forsyth–Edwards Notation (FEN), bir satranç tahtının her hücresindeki taş yerleşimini tek bir satır metin olarak tanımlar.. FEN, satranç motorlarına kolayca iletebilen standart bir protokoldür. Chess Player Helper’da, YOLO ve OpenCV katmanlarıyla her kareden elde edilen taş sınıfları ve koordinatlarından FEN satırını oluşturacak ara katman geliştirilmiştir. Önce, 8×8 hücreye bölünen görüntü üzerinde hücre indeksleri (a1’den h8’e) hesaplanır; ardından her hücreye karşılık gelen taş sembolü FEN dizgesine eklenir. Boş hücreler bir rakamla (örneğin “3” üç boş hücre) gösterilir. Taş tarafı, rok hakları, geçerken alma durumu ve yarı hamle sayısı gibi meta bilgileri de kodlayarak tam bir FEN oluşturulur.

Oluşturulan FEN, Stockfish motoruna gönderilir. Stockfish, FEN’den aldığı pozisyona göre pozisyon değerlendirmesi yapar, en iyi hamle ağacını derinlemesine tarar ve öneri listesi ile bir puan (centipawn) döner [14]. Chess Player Helper, bu puan ve öneri verilerini işleyerek GUI’ya aktarır. Böylece hem oyuncu hem de izleyici, o anki konumun stratejik değerlendirmesini ve en kuvvetli varyantları görebilir.

2.4 Özgünlük ve Farklılıklar

Chess Player Helper’ın literatürdeki muadillerinden ayrıldığı en temel nokta tam entegre gerçek zamanlı deneyim sunmasıdır. Mevcut projelerin birçoğu ya hamle kaydı (Chess

OCR), ya analiz (çevrimiçi platformlar), ya da otonom taş oynatma (ChessBotX) odaklıdır. Helper ise “algılama → analiz → öneri” zincirini uçtan uca kapsar:

- Gerçek Zamanlılık: YOLOv8 ile birlikte insan algı sınırının altında gecikme sağlar.
- GUI Entegrasyonu: PyQt5 tabanlı etkileşimli arayüzde, oyun taş dizimi , hamle geçmişi, varyant ağacı(Stockfish ile hamle tercihi aynı ekranda senkronize görülür.
- Özel Veri Seti: Proje için toplanan ve etiketlenen farklı taş seti ve aydınlatma koşulu bulunduran fotoğraflar ile birlikte özgün veri kümesi, modelin farklı fiziksel ortamlar ve tasarımlar arasında taşınabilirliğini artırmaktadır.
- Modüler Mimari: FEN oluşturma, motor entegrasyonu ve GUI bileşenleri modüler halde geliştirilmiş olup; ileride farklı tespit modelleri, motorlar veya arayüz çatıları kolayca entegre edilebilir durumda tasarlanmıştır.

Bu özellikler, Chess Player Helper’ı akademik bir prototipten çıkarıp hem turnuva ve maç ortamlarında hem de amatör eğitim platformlarına entegre edilebilir ve kullanılabilir gelişmiş gerçek hayat uyumluluğu yüksek bir proje haline getirmektedir.

3. Problem Tanımı

3.1 Problemin Açıklaması

Satranç, rekabetle birlikte bilişsel yeteneklerin sergilendiği, kuralların ve hamle sıralamasının titizlikle takip edildiği stratejik bir oyundur. Özellikle Türkiye Satranç Federasyonu (TSF) ve uluslararası düzeydeki organizasyonlar tarafından düzenlenen resmi turnuvalarda, her oyuncu yaptığı hamleleri oyun süresince özel olarak sağlanan notasyon kâğıtlarına kendileri elle kaydetmekle sorumludur. Bu uygulama sayesinde, oyun sürecinin adım adım belgelenmesini ve karşılaşma sırasında veya sonrasında çıkabilecek anlaşmazlıkların çözüme kavuşturulmasını sağlamakta kritik önem taşımaktadır. Notasyon kâğıdı, hakem müdahalesi gereken durumlarda oyunun doğruluğunu teyit etme açısından temel başvuru kaynağıdır.

Ancak bu yöntemin bazı pratik zorluklar doğurduğu gerçeği gözlemlenmiştir. Oyuncuların her hamle sonrası kâğıda manuel olarak not alma zorunluluğu, oyunun doğal akışını kesintiye uğratmakta ve zihinsel odaklanmayı olumsuz yönde etkileyebilmektedir. Özellikle yüksek düzeyde konsantrasyon gerektiren karşılaşmalarda bu durum, oyuncuların dikkatinin dağılmasına, hatalı hamle yapma olasılıklarının artmasına ve genel performansın düşmesine neden olabilmektedir. Ayrıca, fiziki olarak tutulan notasyonlar çevresel faktörlere karşı dayanıksızdır. Örneğin; yırtılma, kaybolma veya okunaksız yazım gibi sorunlar sürecin güvenilirliğini zedeleyebilmektedir.

Bu noktada oluşabilecek olumsuz durumları ve yetersiz doğruluk gerçekliklerini ortadan kaldırmak amacıyla, sabit açıyla yerleştirilmiş bir kamera aracılığıyla oyun sürecinin gerçek zamanlı olarak kaydedilmesi , hem teknolojik hem de organizasyonel açıdan daha sürdürülebilir bir çözüm olacağı düşünüldüğü projeye başlanılmıştır. Kamera ile alınan görüntülerin bilgisayarla görme teknikleri yardımıyla analiz edilmesiyle, oyuncuların yaptığı hamleler sistem tarafından otomatik olarak algılanabilir, dijital notasyon oluşturulabilir ve oyun sürecini eşzamanlı olarak kaydedebilmeyi sağlar. Bu sayede hem oyuncunun hamle kaydetme sorumluluğu ortadan kaldırılmış olur hem de turnuva

yöneticilerine ve hakemlere, gerektiğinde başvurulabilecek görsel temelli, zaman damgalı bir kayıt sağlanmış olur.

Düzenlenen turnuvalardaki mevcut sistemlerde bu tür teknolojik çözümlerin eksikliği, fiziksel satranç turnuvalarının dijital dönüşümünü sınırlamakta ve hâlen manuel prosedürlere bağımlı kalınmasına neden olmaktadır. Proje kapsamında geliştirilecek kamera destekli otomatik notasyon sistemi, hem oyuncu deneyimini iyileştirme hem de hakem kararlarını objektif verilerle destekleme potansiyeli taşımaktadır. Bu bağlamda, problem yalnızca bir teknolojik eksiklik değil, aynı zamanda organizasyonel verimlilik ve oyuncu konforu , dikkat dağılmasını engelleme açısından da kritik bir boşluğa işaret etmektedir.[8]

Tahta üzerinde oynanan satranç oyunlarında, dijital platformlardakine benzer gerçek zamanlı analiz yapılabilmesi büyük ölçüde bulunmamaktadır. Mevcut çözümler, satranç konumunu dijitalleştirme ve hamle analizini sağlama konusunda çeşitli kısıtlarla çalışırlar. Örneğin, bazı yaklaşımlar yalnızca sabit bir kamera açısı ve aydınlatma altında, hatta bazen özel işaretleyicilere sahip özel taş veya tahtalarla doğru sonuç verebilmektedir. Bu durum, genel kullanıma yönelik esnekliklerini kısıtlamakta ve gerçek zamanlı analiz ihtiyacını tam olarak karşılamamaktadır. Nitekim popüler bir uygulama olan *Chessify*, bir konumu dijital formata aktarmak için kullanıcının her hamleden sonra tahtanın fotoğrafını çekip sisteme yüklemesini şart koşmaktadır; bu yöntem canlı oyun sırasında pratik olmadığı gibi oyunun akışını olumsuz etkilemektedir. Benzer şekilde daha gelişmiş bir sistem olan *KnightVision*, video akışı üzerinden takip yapabilesine rağmen kameranın belirli bir açıyla sabit bir konumda (üçayak üzerinde) olmasını gerektirmektedir. Bu tür gereksinimler, gerçek zamanlı analiz hedefiyle çelişmekte ve oyuncuların maç esnasında anlık geri bildirim almasını güçleştirmektedir. Sonuç olarak, literatürde kapsamlı ve taşınabilir bir satranç görsel analiz sistemi henüz tam anlamıyla mevcut değildir. Bu eksiklikler, özellikle fiziksel tahtada oynayıp hamlelerinin anlık değerlendirmesini görmek isteyen oyuncular ve maçları çevrimiçi yayınlamak isteyen turnuva yöneticileri için önemli bir problem teşkil etmektedir. Dolayısıyla, kamera görüntülerini işleyerek hamleleri anında saptayabilecek ve bir satranç motoru yardımıyla stratejik analiz sunabilecek dijital bir sisteme ihtiyaç duyulmaktadır. Bu projede, OpenCV ve YOLOv8 tabanlı bilgisayarlı görü teknikleri ile Optical Flow tabanlı izleme ve Stockfish satranç motoru analizini bir araya getiren böyle bir gerçek zamanlı sistem ele alınmaktadır. [1]

3.2 Ortam Tanımı ve Varsayımlar

Görüntü işleme tabanlı modellerin doğruluğu, büyük ölçüde veri toplama sürecinde sağlanan ortam koşullarına bağlıdır. Bu çalışmada geliştirilen satranç hamlesi tanıma sisteminin eğitimi ve testi, kontrollü bir iç mekân ortamında, belirli şartlar altında gerçekleştirilmiştir. Modelin kararlılığını sağlamak ve dış etkenlerden kaynaklanabilecek doğruluk sapmalarını minimize etmek amacıyla hem eğitim hem de test süreçleri, aynı fiziksel ortamda yürütülmüş ve parametreler sabit tutulmuştur.



Şekil.3.2. Tasarlanan oyun izleme düzeneği

Eğitim sürecinde kullanılan görüntülerin alınmasında, sabit bir tripod üzerine kamera yerleştirilerek elde edilmiştir. Kamera, beyaz taş oyuncusunun oturur pozisyonda tahtaya baktığında sahip olacağı göz hizası dikkate alınarak belirli bir yükseklikten ve açıdan, uygun perspektife konumlandırılmıştır. Bu sayede modelin eğitildiği görüntülerin, gerçek kullanım senaryolarına yakın bir açıdan elde edilmesi hedeflenmiştir. Kamera açısı ve yüksekliği, tüm veri toplama süreci boyunca değiştirilmemiştir. Modelin eğitiminde kullanılan bu sabit açılardırmanın, algoritmanın genel doğruluğu üzerinde belirleyici olduğu tespit edilmiştir. Yapılan gözlemlerde, bu açının değiştirilmesi durumunda modelin taş tespiti ve kare eşleme performansında doğruluk değerinde düşme yaşanabileceği öngörülmüştür. Bu, literatürde sıklıkla vurgulanan “görüş açısı bağımlılığı” (viewpoint dependency) probleminin bir yansımasıdır. Derin öğrenme tabanlı nesne tanıma sistemleri, eğitildikleri perspektiften farklı açılara sahip görüntülerde tespit başarısını korumakta sınırlılıklara sahiptir. (Redmon et al., 2016).

Işıklandırma koşulları da sabit tutulmuştur. Eğitim ve test aşamaları, yalnızca tek bir yapay ışık kaynağı olan bir odada, zemin rengi sabit olacak bir masada gerçekleştirilmiştir. Aydınlatma, doğrudan değil, ortam aydınlatması şeklindedir ve değişken doğal ışık yani gün ışığı etkisini dışlamak için dış ortamdan gelen ışık kaynakları sınırlandırılmıştır. Bu koşullar, görüntülerde oluşabilecek gölgelenme, yansıma veya parlama gibi görsel gürültüleri minimize ederek modelin daha tutarlı ve dengeli bir veri seti üzerinde öğrenmesini sağlamıştır. Modelin dış ortam ışığı veya yansımadan kaynaklı ışıklar gibi farklı ışık koşullarında test edilmesi durumunda performansında sapmalar oluşabileceği varsayılmaktadır. Bu durum dikkate alınıp ,çevresel faktörlerden etkilenmemesi noktasında bir ortam hazırlanmıştır.

Satranç tahtası, düz ve homojen yüzeyli kahverengi bir masa üzerine yerleştirilmiştir. Bu yüzey seçimi, arka plan gürültüsünü azaltmak ve taşların görüntüde daha kolay ayrıştırılmasını sağlamak amacıyla yapılmıştır. Arka plan kontrastının bu tür tespit işlemlerinde önemli olduğu, daha önceki bilgisayarla görme uygulamalarında da

vurgulanmıştır (OpenCV Documentation, 2023). Görüntüde yalnızca taşlar ve kareler bulunacak şekilde çerçeveleme yapılmış, kamera dışındaki cisimlerin görünmemesi sağlanmıştır. Veri eğitimi sürecindeki ,etiketleme sürecinde de satranç zemini asıl çerçeve olacak şekilde eğitilmiştir.

Eğitimde kullanılan fiziksel taş seti ile testte kullanılan taş seti birebir aynı olup, modelin aynı görsel örüntüler üzerinden doğruluk değerlendirmesi yapılmıştır. Bu yaklaşım, test sırasında veri setinden bağımsız değişkenlerin model performansına etkisini sınırlandırmakta ve iç geçerliliği (internal validity) artırmaktadır. Ancak bu durum aynı zamanda modelin genelleme kapasitesini sınırlandırmaktadır; farklı tasarımlara, ışık koşullarına veya açılara sahip senaryolarda performansın düşebileceği öngörülmektedir.

Sonuç olarak, sistemin geliştirme ve test sürecinde kullanılan ortam koşulları, modelin belirli sabit varsayımlar altında yüksek doğruluk göstermesini mümkün kılmıştır. Ancak bu koşullardan sapma durumunda sistemin yeniden kalibrasyona ya da transfer öğrenme yöntemiyle adaptasyona ihtiyaç duyabileceği değerlendirilmektedir.

Bu çalışmada önerilen sistem, kontrollü bir iç mekân ortamında sabit bir kamera düzeni ile çalışacak şekilde tasarlanmıştır. Kamera, satranç tahtasına tepeden bakan sabit bir açıyla konumlandırılmıştır; bu sayede tek bir karede tüm tahtayı ve üzerindeki taşları kapsayan *panoramik* bir görüntü elde edilmektedir. Kamera yüksekliği ve açısı deney boyunca değiştirilmemiş, her karede tahtanın konumu görüntünün tutarlı bir bölgesine denk gelecek şekilde kalibrasyon yapılmıştır. Benzer akademik çalışmalarda da veri toplama aşamasında kameranın sabit açı ve mesafede konumlandırılması yaygın bir yaklaşımdır; böylece algoritmalar, değişken perspektiflerin getireceği karmaşıklıklardan arındırılmış verilerle eğitilir. Ortamın aydınlatması da deney süresince sabit tutulmuştur. Işık şiddeti ve yönü, yansıma veya gölge oluşturmayaacak biçimde ayarlanmış ve sabitlenmiştir. Modelin eğitimi ve testi, bu sabit kamera açısı ve aydınlatma koşulları altında gerçekleştirilmiştir. Dolayısıyla, geliştirilen nesne tanıma modeli (YOLOv8 tabanlı), yalnızca belirlenen perspektiften görünen standart bir satranç tahtasında yüksek başarı göstermektedir. Bu varsayım, gerçek dünyadaki tüm senaryoları kapsamasa da sistemi geliştirme aşamasında belirsizlikleri azaltmak ve tutarlı bir temel oluşturmak amacıyla bilinçli olarak yapılmıştır. Ortam koşullarının sabit olduğu bu senaryoda, OpenCV ile görüntü işleme (ör. perspektif düzeltme, renk filtreleme) adımları daha güvenilir uygulanabilmekte ve Optical Flow ile hareket takibi gürültüden arındırılmış bir şekilde gerçekleştirilebilmektedir. Eğitim verilerinin ve test ortamının bu kontrollü çerçevede olması, sonuçların tekrarlanabilirliğini de artırmıştır. Ancak, bu varsayımların dışında kalan durumlar ,örneğin; kamera açısının değişmesi veya ışığın aniden azalması bir sonraki bölümün devamında ele alınacak olan sınırlamaları beraberinde getirmektedir.

3.3 Proje Gereksinimleri ve Sınırlamalar

Görüntü işleme tabanlı bir satranç hamlesi tespit sisteminin başarılı şekilde çalışabilmesi, yalnızca yazılım altyapısına değil, aynı zamanda donanım ve ortam koşullarına da doğrudan bağlıdır. Bu bağlamda geliştirilen modelin kullanılabilirliği ve doğruluğu, çeşitli gereksinimlerle sınırlı kalmakta, bazı durumlarda performans düşüklükleri yaşanabilmektedir. Bu bölümde, sistemin karşılaştığı teknik ve operasyonel sınırlamalar kapsamlı biçimde ele alınmaktadır.

3.3.1 Özel Taşlar ve Tahta Tasarımı Bağımlılığı

Bu çalışmada kullanılan satranç tahtası ve taşları, Türkiye Satranç Federasyonu tarafından resmi turnuvalarda standart olarak kullanılan ekipmanlardır. Söz konusu takım, Staunton tarzı olarak bilinen klasik tasarımı esas almakta olup, hem ulusal hem de uluslararası turnuvalarda hakem kontrolünde kullanılan modeldir (TSF Ekipman Standartları, 2023). Bu ekipmanlar ile eğitilmiş olan YOLOv8 nesne tanıma modeli, yalnızca bu taşların ve tahtanın görsel özelliklerine (şekil, boyut, renk, orantı) göre optimize edilmiştir.

Ancak eğitim verisi dışında kalan, örneğin tematik tasarımlara sahip taş setleri (örneğin cam, metal, çizgi roman karakterli setler vb.) veya farklı ölçü ve renk varyasyonları içeren tahtalarda, sistemin doğruluğu ciddi oranda azalabileceği öngörülmektedir. YOLOv8 gibi evrişimli sinir ağı tabanlı modellerin görsel örüntülere duyarlı olduğu bilinmektedir; bu nedenle eğitimde yer almayan nesne görünümleriyle karşılaşıldığında hatalı sınıflandırmalar yapılması olasıdır. Bu durum, sistemin genellenebilirliğini sınırlandırmakta ve her yeni taş seti için yeniden eğitim gerekliliğini doğurmaktadır.

Sonuç olarak, model yeni bir taş stiliyle karşılaştığında o taşı bilinen sınıflardan birine güvenilir biçimde eşleyemeyebilir. Bu sınırlama, derin öğrenme tabanlı algılayıcıların eğitim verisine bağımlı olmasından kaynaklanır; modelin başarılı olabilmesi için farklı tasarımları kapsayacak şekilde veri artırımı veya yeniden eğitim gerekebilecektir.

3.3.2 Kamera Açısı ve Işık Koşullarının Etkisi

Modelin eğitildiği ortamda kamera, sabit bir tripoda yerleştirilmiş ve beyaz taş tarafında oynayan oyuncunun göz hizasına yakın bir yükseklikten, tahtaya tam tepeden bakacak şekilde konumlandırılmıştır. Bu sabit açının değiştirilmesi durumunda, örneğin kamera daha yatay bir perspektife alındığında ya da eğik açılardan görüntü sağlandığında, taşların karelerle olan konumsal ilişkisi bozulmakta ve tespit algoritmasında sapmalar oluşmaktadır. Nitekim nesne tanıma literatüründe kamera perspektifinin doğruluk üzerindeki etkisi sıkça vurgulanmakta ve sabit kamera geometrisinin performans için temel şart olduğu belirtilmektedir (Redmon et al., 2016; Bochkovskiy et al., 2020).

Benzer şekilde, sabit ortam aydınlatması dışına çıkıldığında, özellikle yetersiz ışık, gölge oluşumu veya parlak yüzey yansımaları, kontrol dışında gelebilecek gün ışığı taşların model tarafından doğru şekilde algılanmasını zorlaştırmaktadır. Düşük ışık koşullarında kamera görüntüsündeki kontrast azalmakta, taş ve zemin ayrımı bulanıklaşmakta, bu da hem taşın tanınmasını hem de doğru kareye eşlenmesini zorlaştırmaktadır. Aynı zamanda parlama, bazı taşların kenarlarını silikleştirerek modelin “bounding box” yerleşiminde kararsızlık yaratmaktadır. Bu tür optik sorunlar, taş tespiti sürecinin temel güvenilirliğini zedelemektedir.

Mevcut sistem de benzer şekilde, kamera perspektifinin ciddi biçimde değiştiği veya ortam ışığının eğitimdekinden farklılaştığı senaryolarda hatalı kare okuma, taş konumlandırma hataları ya da taşları hiç algılayamama gibi problemlerle karşılaşabilir. Bu sınırlama, sistemin gerçek dünyada yaygınlaştırılabilmesi için ortam koşullarına duyarlılığın azaltılması yönünde ileride ek çalışmalar yapılması gerektirdiğini göstermektedir.[5]

3.3.3 Gerçek Zamanlılık ve Hesaplama Performansı

Oluşturulan sistem, kullanıcıya gerçek zamanlı bir analiz deneyimi sunmak üzere tasarlanmıştır. Uygulama, kameradan sürekli olarak görüntü akışı almakta ve bu akıştan belirli zaman aralıklarında anlık görüntü “frame capture” olarak, bu kareyi nesne tespiti için YOLOv8 modeline göndermektedir. Tespit edilen taş pozisyonları ile oluşturulan FEN string, ardından Python-chess kütüphanesi aracılığıyla Stockfish motoruna iletilmekte ve elde edilen analiz GUI (Grafiksel Kullanıcı Arayüzü) üzerinde kullanıcıya yansıtılmaktadır.

Bu işlemler zinciri teorik olarak gerçek zamanlı görünse de, pratikte çeşitli performans kısıtları ortaya çıkmaktadır. Özellikle:

- Yüksek çözünürlüklü karelerin işlenmesi,
- YOLOv8 modelinin inference (çıkarım) süresi,
- Optical Flow algoritmasının eşzamanlı çalışması,
- Stockfish motorunun analiz süresi,
- GUI üzerinden çıktıların çizilmesi

gibi işlemler, belirli bir zaman aralığına yayılmakta ve toplam işlem süresi artabilmektedir. Bu da bazı sistemlerde kare başına gecikmeye (latency) sebep olmakta ve sistemin gerçek zamanlı dönüş üretmesinde sekteye uğrayabilmektedir. Ayrıca, CPU üzerinden çalışan sistemlerde FPS değeri düşük kalmakta ve kullanıcıya sunulan analiz gecikmeli olarak iletilebilmektedir. Yalnızca GPU destekli donanımlarda, bu işlem süreci ideal sınırlarda tutulabilmektedir. Bu nedenle sistemin tam anlamıyla gerçek zamanlı çalışabilmesi için donanım kapasitesi (özellikle GPU belleği ve işlem gücü) kritik bir gereksinim hâline gelmektedir.

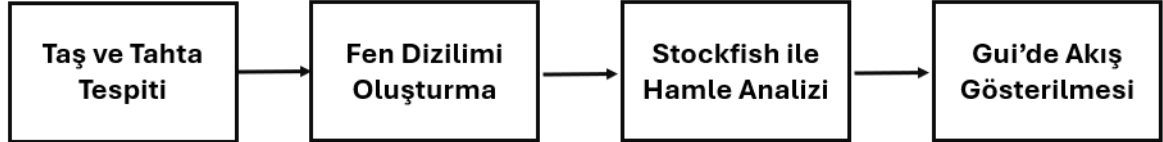
Sonuç olarak, geliştirilen sistemin kullanım başarısı; belirli taş setlerine, sabit kamera ve ışık koşullarına ve yüksek performanslı donanıma bağlıdır. Bu sınırlamalar, sistemin geniş çaplı dağıtımı ve farklı ortamlarda kullanımı öncesinde yeni veri setleri ile yeniden eğitimi ve sistem mimarisinde optimizasyon ihtiyacını gündeme getirmektedir. Ancak bu sınırlamalara rağmen, kontrollü ortamlarda modelin yüksek doğrulukla çalıştığı ve kullanıcı deneyimini iyileştirdiği görülmüştür.

Özellikle işlemci üzerinde çalışan derin öğrenme modellerinde çerçeve işleme hızının sınırlı kaldığı görülmektedir. Dolayısıyla, gerçek zamanlılık hedefi ile sistemin donanım/yazılım performansı arasında bir denge gözetmek gerekmektedir. Bu proje kapsamında mevcut donanımla yaklaşık gerçek zamanlı bir performans elde edilse de (örneğin saniyede birkaç kare işlenebilmesi), yüksek kare hızlarında tutarlı analiz için optimizasyon (modeli hızlandırma, donanım hızlandırıcılarından yararlanma gibi) ihtiyaçları devam etmektedir. Bu durum, gerçek zamanlı bir satranç analiz sisteminin ölçeklenebilirlik ve verimlilik açısından geliştirilmesi gereken yönleri olduğunu işaret etmektedir.

4. METOT (YÖNTEM)

4.1 Sistem Mimarisi

Bu çalışmada geliştirilen “Chess Player Helper” sistemi, fiziksel bir satranç tahtası üzerinde oynanan oyunu gerçek zamanlı olarak analiz etmeyi amaçlayan, bilgisayarla görme ve yapay zeka tabanlı çok katmanlı bir sistemdir. Önerilen sistemin adımlarını gösteren iş akış şeması Şekil-2’de verilmiştir. Sistemin genel mimarisi dört temel aşamadan oluşmaktadır: (1) taş ve tahta tespiti, (2) konum bilgileri ile FEN dizilimi oluşturulması, (3) Stockfish satranç motoru ile hamle analizinin yapılması ve (4) sonuçların kullanıcıya grafiksel bir arayüz (GUI) üzerinden sunulması.



Şekil.4.1.Önerilen sistemin iş akış şeması

Bu çok katmanlı mimari yapı, literatürdeki benzer çalışmalara dayanmaktadır. Görüntüden taşların tespiti, pozisyon analizi ve hamle önerisinin birlikte çalıştığı benzer sistemler örneğin KnightVision (Sharma et al., 2021) ve ChessVision.ai gibi uygulamalarda da benzer blok yapı ile modellenmiştir. Ayrı katmanlar hâlinde tasarlanan bu yapı sayesinde hata izolasyonu kolaylaşmakta, sistem modüler şekilde geliştirilebilmektedir. Özellikle, taşların algılanması ve FEN üretimi gibi süreçlerin birbirinden ayrılması, sistemin daha sürdürülebilir ve hata toleranslı olmasını sağlamaktadır.

4.2. Satranç Tahtasını Algılama Katmanı

İlk aşamada, kamera sabit tutularak alınan görüntü üzerinde YOLOv8 nesne tanıma algoritması kullanılarak satranç tahtasının dört köşesi ve taşların konumları tespit edilmektedir. Tahtanın köşeleri tespit edildikten sonra OpenCV yardımıyla perspektif düzeltme işlemi uygulanır ve 8x8 grid sistemi elde edilir. Daha sonra, taşların koordinatları bu grid yapısı ile eşleştirilerek her bir taşın hangi karede bulunduğu belirlenir. Bu işlemler, sonraki aşama olan FEN üretimi için temel teşkil eder.

Satranç tahtasını algılamak, sistemin iş akışındaki ilk ve kritik adımdır. Bu işlemde, sabit bir kamerayla alınan görüntü üzerinde YOLOv8 nesne tanıma algoritması kullanılarak satranç tahtasının dört köşesi ve taşların konumları otomatik olarak tespit edilmektedir. Köşe bilgileri, tahtanın hizalanması ve taşların doğru karelere eşleştirilmesi açısından büyük öneme sahiptir.

YOLO (You Only Look Once), gerçek zamanlı nesne tespiti konusunda literatürde yaygın olarak tercih edilen bir derin öğrenme mimarisidir. YOLOv8, Ultralytics tarafından geliştirilen en güncel versiyon olup, önceki sürümlere göre daha yüksek doğruluk (mAP) ve daha düşük gecikme süresi (latency) sağlamaktadır (Jocher et al., 2023). YOLO mimarisi, bir görüntü üzerinde nesneleri aynı anda sınıflandırma ve konumlandırma özelliğine sahiptir; bu özellik satranç gibi çok sayıda küçük ve benzer nesnenin bulunduğu ortamlarda büyük avantaj sağlayarak , tespit sağlamaktadır. Bu projede YOLOv8'in "s" (small) versiyonu tercih edilmiştir. Bu model, düşük donanım gereksinimleri ile yüksek doğruluk oranı arasında denge sağlamaktadır. Eğitim süreci sırasında elde edilen sonuçlar, modelin

mAP@0.5 oranında %97.8 gibi yüksek başarı düzeyine ulaştığını göstermiştir. Bu da taşların çoğunlukla doğru şekilde sınıflandırıldığını göstermektedir.

YOLO modeli bu görev için özel olarak eğitilmiş olup, eğitim sürecinde yaklaşık 400 görüntüden oluşan bir veri kümesi kullanılmıştır. Bu görüntüler, satranç tahtasının köşe bölgeleri elle anotasyonlanarak modele öğretilmiş, köşe tespiti doğruluğu arttırılmıştır. Köşelerin başarıyla algılanmasının ardından, OpenCV yardımıyla görüntüye perspektif dönüşüm uygulanmakta ve tahtanın üstten bakışla düz bir görünümü böylece elde edilmektedir. Bu görünüm üzerinden 8x8'lik bir grid sistemi oluşturulmakta ve YOLO ile tespit edilen her taşın koordinatları bu kare yapısıyla eşleştirilerek hangi karede hangi taşın bulunduğu belirlenmektedir. Bu işlem, sistemin sonraki aşaması olan FEN üretimi ve hamle analizi için temel ve gerekli bir aşamadır.



Şekil.4.2.Köşe tespiti

Satranç tahtası ve üzerindeki taşların doğru şekilde tespit edilmesi, Chess Player Helper sisteminin temel yapı taşıdır. Bu amaçla, YOLOv8 nesne tanıma algoritması kullanılmış ve model, özel olarak oluşturulmuş bir veri seti üzerinde eğitilmiştir. Algılama süreci iki temel bileşene ayrılmıştır: (1) satranç tahtasının dört köşesinin tespiti, (2) satranç taşlarının tür ve konumlarının sınıflandırılması.

4.3. Parçaların Tespiti

Satranç tahtasını algıladıktan ve ızgara konumlarını kaydettikten sonraki adım, tahtadaki satranç taşlarını tanımlamak ve sınıflandırmaktır. Satranç taşlarını tanımak ve siyah ve beyaz taşlar arasında ayırım yapmak üzere özel olarak eğitilmiş başka bir YOLO nesne algılama modeli kullanılır.

Satranç tahtasının konumu başarıyla belirlendikten ve 8x8'lik ızgara yapısı elde edildikten sonra, sistemin bir sonraki aşaması tahtadaki satranç taşlarının tespit edilmesi ve sınıflandırılmasıdır. Bu adım, oyunun mevcut durumunu doğru şekilde modelleyebilmek ve hamle analizlerinin yapılabilmesi için kritik öneme sahiptir.

Sistemde, satranç taşlarının türlerini (örneğin piyon, at, kale, vezir) ve renklerini (beyaz veya siyah) ayırt edebilecek şekilde özel olarak eğitilmiş ikinci bir YOLOv8 nesne algılama modeli kullanılmıştır. Bu model, satranç tahtası üzerinde konumlandırılmış taşları otomatik olarak tespit eder ve her taş için sınırlayıcı kutular (bounding boxes) oluşturur. Her kutu, taşın bulunduğu alanı kapsar ve buna karşılık elen sınıf etiketiyle birlikte gelir.[6]

Sınıf Kimlik Numarası	Etiketleme Harf Karşılığı	Taş İsmi
0	<u>b</u>	Siyah Fil
1	<u>k</u>	Siyah Şah
2	<u>n</u>	Siyah At
3	<u>p</u>	Siyah Piyon
4	<u>q</u>	Siyah Vezir
5	<u>r</u>	Siyah Kale
6	<u>B</u>	Beyaz Fil
7	<u>K</u>	Beyaz Şah
8	<u>N</u>	Beyaz At
9	<u>P</u>	Beyaz Piyon
10	<u>Q</u>	Beyaz Vezir
11	<u>R</u>	Beyaz Kale

Tablo.4.3.Sınıf Etiketlerinin Satranç Taşı Notasyonlarına Eşlenmesi




Şekil.4.3.Satranç Tahtasında Taşların Algılanması

Veri seti, sistemin test edileceği sabit açılı tripod düzeneği kullanılarak, kullanıcılar tarafından manuel olarak oluşturularak sisteme aktarılmıştır. Eğitimde kullanılmak üzere yaklaşık 400 adet orijinal görsel, satranç tahtası üzerinde farklı konumlandırılmış taşlarla birlikte çekilmiştir. Bu görseller, Roboflow platformu kullanılarak etiketlenmiş ve artırılmıştır. Augmentasyon sürecinde görüntüler üzerinde şu işlemler uygulanmıştır:

- Görüntülerin %30'una kadar grayscale (gri tonlama),
- Maksimum 4 piksele kadar blur (bulanıklaştırma) uygulanması,
- Boyutlandırma: tüm görüntüler 640x640 piksel olarak yeniden boyutlandırılmıştır.

Bu augmentasyonlar, modelin değişen ışık ve görüntü kalitesi koşullarında daha genel sonuçlar vermesini sağlamak için yapılmıştır. Sonuçta veri seti yaklaşık 1000 örneğe çıkarılmış ve bu örnekler eğitim (835), doğrulama (80) ve test (38) olmak üzere üçe ayrılmıştır .

Model toplamda 12 sınıfa göre eğitilmiştir. Bu sınıflar şunlardır: B, K, N, P, Q, R (beyaz taşlar) ve b, k, n, p, q, r (siyah taşlar). Her sınıf, satranç notasyonuna uygun olarak adlandırılmıştır ve FEN üretimi aşamasında doğrudan bu etiketlerle eşleştirilmiştir. Eğitim süreci 25 epoch boyunca yürütülmüş, optimizasyon için YOLOv8s model dosyası (yolov8s.pt) kullanılmıştır. Aşağıdaki komut dizisi ile eğitim gerçekleştirilmiştir:

- !yolo task=detect mode=train model=yolov8s.pt data=data.yaml epochs=25 imgs=640 plots=True

Bu eğitim sürecinin sonunda, model başarıyla doğrulama seti üzerinde test edilmiştir.

4.4. FEN Üretimi Katmanı

FEN (Forsyth–Edwards Notation), satranç oyunlarında bir konumun sistematik ve evrensel şekilde ifade edilmesini sağlayan standartlaştırılmış bir gösterim biçimidir. Bu notasyon, taşların tahtadaki konumlarını ve uygun önerilecek hamle önerisini yapmaktır. Bu yönüyle FEN, satranç motorları tarafından pozisyonun anlaşılması, analiz edilmesi ve hamle önerilmesi için kritik bir yapı olma görevini üstlenir.

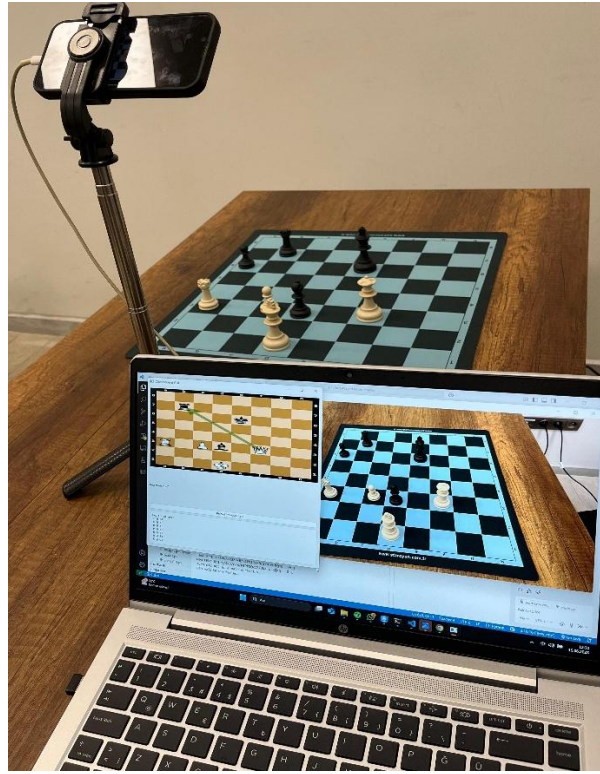
Geliştirilen bu sistemde, YOLOv8 algoritması tarafından tespit edilen taşların konumları, daha önce oluşturulan 8x8 boyutundaki grid sistemiyle eşleştirilir. Elde edilen bu eşleşmeler, sistemin FEN dizisini oluşturabilmesi için temel girdiyi oluşturur. Her taşın sınıf etiketi ilgili FEN sembolüne dönüştürülerek satranç tahtasının temsili görünümü sayısal olarak üretilir.

FEN üretim süreci aşağıdaki adımlarla gerçekleştirilir:

1. Taşların Grid Üzerine Haritalanması: YOLO modeli, hizalanmış ve normalize edilmiş satranç tahtası görüntüsü üzerinden taşları algılar. Her bir tespit edilen taşın alt-orta koordinatı, 8x8 karelik ızgara yapısı içerisinde belirli bir kareye eşlenir. Elde edilen bu eşleşmelerle taşlar, grid üzerine yerleştirilmiş olur.
2. Satır Bazında Notasyona Dönüşüm: Tahtanın her satırı (yani her yatay sıra), soldan sağa doğru taranarak boş ve dolu kareler ayrıştırılır. Boş kareler, ardışık şekilde sayılarla ifade edilirken; dolu kareler ilgili taşın notasyon harfi ile belirtilir (örneğin 'p' siyah piyon, 'K' beyaz şah gibi). Her bir satırın FEN karşılığı bu şekilde elde edilir.
3. FEN Dizesinin Oluşturulması: Sekiz satırın her biri eğik çizgi (/) ile ayrılarak birleştirilir. Bu, FEN dizisinin ilk bölümünü yani taş yerleşimini ifade eder.
4. Hatalı Tespitte Fallback Mekanizması: Eğer üretilen yeni FEN, geçerlilik kontrollerinden geçemezse ya da sistem tarafından doğrulanamazsa, devreye "Fallback Mekanizması" girer. Bu yapı sayesinde, en son geçerli olduğu doğrulanan FEN dizisi geri yüklenerek sistemin stabil çalışması sağlanır. Bu durum, özellikle kamera görüşündeki taşların anlık kaybı, aşırı parlama, tespit sapması gibi durumlarda devreye girerek sistemin sürekliliğini garanti eder.

Sonuç olarak, FEN üretimi katmanı yalnızca görsel tespitin metinsel bir temsili oluşturmakla kalmaz, aynı zamanda satranç motoru ile makine arasında anlamlı bir iletişim

kurulmasını sağlar. Bu bileşen sayesinde sistem, hamle önerileri, oyun durumu analizi ve hata düzeltme gibi ileri düzey işlemleri yerine getirebilecek bir yapıyı oluşturmuştur.



Şekil.4.4. FEN ve Oyun Durumuna Bir Örnek

Görüntü işleme ile tespit edilen taşların ve konumlarının, satranç motoru tarafından anlaşılabilir bir biçime dönüştürülmesi için FEN (Forsyth–Edwards Notation) kullanılmıştır. FEN formatı, satranç tahtasının mevcut durumunu standartlaştırılmış şekilde ifade eder ve satranç motorlarının (örneğin Stockfish) pozisyon analizi yapmasını sağlar. Bu çalışmada, YOLOv8 ile tespit edilen taşlar, grid yapısına göre karelere yerleştirilmiş ve FEN stringi oluşturularak hem analiz hem de hamle önerisi süreci başlatılmıştır.

FEN, Uluslararası Satranç Federasyonu (FIDE) tarafından da kabul gören, satranç konumlarını metinsel biçimde tanımlamak için kullanılan bir standarttır. Tahtadaki tüm taşların yerini, hangi oyuncunun hamle sırasının olduğunu, rok haklarını, en passant ihtimallerini, yarı hamle sayısını ve tam hamle numarasını tek bir satırda ifade eder. Bu format, hem hamle analizi hem de geçmiş oyun konumlarının kayıt altına alınmasında büyük kolaylık sağlar. Özellikle Stockfish gibi motorların FEN ile çalışıyor olması, bu yapının sistem mimarisine doğrudan entegre edilmesini zorunlu kılmıştır.

FEN oluşturma işlemi, aşağıdaki mantıksal adımlarla gerçekleştirilmiştir:

1. Grid Oluşturma ve Taş Eşleme

YOLO modeli tarafından tespit edilen taşların bounding box (sınırlayıcı kutu) alt orta noktaları alınarak 8x8'lik grid yapısındaki karelere eşlenmiştir. `map_pieces_to_grid()` fonksiyonu ile her taş doğru hücreye atanmış ve boş kareler işlenmek üzere bırakılmıştır.

2. FEN Satırlarının Oluşturulması

Tahta grid'i satır satır taranarak, her satır için FEN kurallarına uygun dizilim üretilmiştir. Art arda gelen boş kareler sayı ile temsil edilirken, taşlar ilgili harfleriyle yazılmıştır (örneğin RNBQKBNR veya pppppppp gibi). Her satır, / karakteri ile ayrılmıştır.

3. Aktif Oyuncunun Belirlenmesi

Oyuncu sırası tespiti, valid_fens listesinde kayıtlı olan son FEN verisine göre yapılmıştır. Eğer bir önceki pozisyon ile güncel pozisyon aynıysa, yeni bir FEN üretilmeden önceki FEN korunmuştur. Eğer farklıysa, hamle sırası değiştirilecek şekilde "w" veya "b" parametresi güncellenmiştir.

4. Castling, En Passant ve Sayaçlar

Rok hakları, tahtada K, Q, k, q taşları olup olmasına göre "KQkq" veya "-" olarak yazılmıştır. En passant ve sayaçlar şimdilik sadeleştirilmiş olarak "- 0 1" şeklinde sabit bırakılmıştır ancak yapının daha sonra genişletilebilir olması gözetilmiştir.

Aşağıda örnek bir FEN string örneği verilmiştir:

- rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1

Bu ifade, oyunun başlangıç pozisyonunu ve beyazın sırasını göstermektedir. FEN doğrulama, sistemin hata toleransını artırmak amacıyla validate_fen() fonksiyonu üzerinden yürütülmüştür. Bu mekanizma aşağıdaki kontrolleri içerir:

- **Aktif hamle sırası kontrolü:** Yeni FEN'in, önceki pozisyona göre sırası gelen oyuncuyu doğru yansıtıp yansıtmadığı denetlenmiştir.
- **Taş sayısı kontrolü:** Son geçerli pozisyonla karşılaştırıldığında, taş sayısındaki ani değişiklikler (örn. bir karede iki taş görünmesi, piyon fazlalığı gibi) durumları için tolerans sınırı belirlenmiştir (± 2 taş).
- **Hatalı FEN yapısı:** Satranç kurallarına aykırı veya eksik FEN ifadeleri oluşturulması durumunda, sistem hata mesajı vererek FEN'i kabul etmemektedir.

Bu kontrollerin sonucu başarılıysa, yeni FEN "valid_fens" listesine eklenir ve GUI ekranına aktarılır. Aksi durumda, kullanıcıya önceki geçerli FEN ile devam etme seçeneği sunulmaktadır. Bu yapı, sistemin istikrarını artırmakta ve kullanıcıya güvenli bir deneyim sağlamaktadır.

4.5. Hamle Önerisi Katmanı (Stockfish Entegrasyonu)

FEN dizesi üretildikten sonra sistemin üçüncü temel bileşeni olan hamle önerisi katmanı aşamasına geçilmiştir. Bu katman, tahta üzerindeki mevcut konumun değerlendirilerek en iyi hamlenin kullanıcıya önerilmesini amaçlar. Bu işlev, açık kaynaklı ve dünya genelinde en güçlü motorlardan biri olarak kabul edilen Stockfish satranç motorunun entegrasyonu ile gerçekleştirilmiştir.

İlk uygulamalarda, Stockfish motoru Python üzerinden subprocess komutu aracılığıyla başlatılmış ve bu şekilde analiz işlemleri yapılmıştır. Ancak bu yöntemin bazı önemli

dezavantajları tespit edilmiştir. Özellikle gerçek zamanlı sistemlerde subprocess kullanımı zamanla performans darboğazlarına neden olmuş, işlem sırasında motor çökmesi veya gecikmeli cevap gibi istikrarsızlıklar gözlemlenmiştir. Bu nedenle, sistemin daha kararlı ve hızlı çalışabilmesi amacıyla bu yapı terk edilmiştir.

Sistem, geliştirme sürecinin ilerleyen aşamalarında daha optimize ve modern bir yaklaşım olan API temelli motor kontrolü modeline geçirilmiştir. Bu yöntemle, Stockfish motoru arka planda sabit bir servis olarak çalışmakta ve analiz talepleri bir API arayüzü üzerinden gönderilmektedir. Bu mimari ile:

- Hamle analizi daha kararlı hale gelmiş,
- İşlem süreleri önemli ölçüde kısalmış,
- Kullanıcıya kesintisiz hamle önerileri sunulabilir hale gelmiştir.

FEN dizesi, sistem tarafından analiz amacıyla motorun API arayüzüne iletilmekte ve satranç motoru bu konuma ilişkin en iyi hamleyi üretmektedir. Geri dönen hamle verisi, GUI üzerinden görsel olarak kullanıcıya aktarılır. Arayüzde okla gösterilen hamle, hem kullanıcı deneyimini artırmakta hem de oyun stratejisi açısından yönlendirici bir görev üstlenmektedir.

Ayrıca sistem, önerilen hamleleri yalnızca beyaz oyuncu için sunmakta; siyah hamlelerin yapılmasını kullanıcıdan beklemektedir. Bu sayede oyuncular kendi kararlarını vererek sistemi yalnızca yardımcı bir araç olarak kullanmaktadır.

Sonuç olarak, hamle önerisi katmanı satranç analizinin merkezinde yer alarak sistemin zeka katmanını oluşturur. Stockfish motorunun doğru ve etkin kullanımı sayesinde sistem yalnızca hamleleri kaydetmekle kalmaz, aynı zamanda stratejik öneriler sunarak oyuncunun performansını iyileştirmeye katkı sağlar.

FEN formatı ile temsil edilen satranç tahtasının durumunun analiz edilebilmesi ve oyuncuya öneri sunulabilmesi için, bu çalışmada Stockfish satranç motoru entegre edilmiştir. Stockfish, açık kaynaklı ve dünyanın en güçlü hamle hesaplama motorlarından biri olup, FIDE tarafından turnuvalarda kullanılan motorlara da temel oluşturmaktadır. Sistem; tespit edilen FEN ifadesini, Python üzerinden Stockfish'e ileterek mevcut pozisyonu analiz ettirmekte ve en uygun hamleyi hesaplamaktadır.

Stockfish motoru, açık kaynaklı yapısı ve zengin API desteği sayesinde farklı sistemlere entegre edilebilir bir altyapı sunmaktadır. Aynı zamanda;

- Çok yüksek ELO derecesine sahiptir (3500+),
- Derin pozisyon analizi yapabilir,
- UCI (Universal Chess Interface) desteği ile birçok satranç uygulamasında kullanılabilir,
- Python ortamında python-chess kütüphanesi ile sorunsuz şekilde çalışabilmektedir.

Bu özellikleri, projeye güçlü bir analiz bileşeni kazandırmıştır. Ayrıca, gerçek zamanlı çalışan sistemlerde düşük gecikmeyle en iyi hamleyi önerme performansı da seçim sebebi olmuştur.

Entegrasyon süreci chess.engine.SimpleEngine sınıfı ile gerçekleştirilmiştir. Satranç motoru sistemde yerel olarak çalıştırılmakta ve analiz edilen FEN string'ine karşılık gelen en iyi hamle uci() formatında alınmaktadır. Motorun çağırılması süreci aşağıdaki gibi yapılandırılmıştır:

with chess.engine.SimpleEngine.popen_uci(self.stockfish_path) as engine:

```
    engine.configure({"Threads": 2, "Hash": 128})
```

```
    result = engine.play(board, chess.engine.Limit(time=2.0))
```

Bu yapılandırmada:

- Threads: 2 parametresi motorun iki işlemci çekirdeği ile çalışmasını sağlamaktadır.
- Hash: 128 değeri motorun 128 MB RAM tampon alanı kullanmasını belirtmektedir.
- Limit(time=2.0) parametresi, hamle önerisinin en fazla 2 saniyelik analizle hesaplanacağını belirtmektedir.

Bu yapılandırma, sistemin gerçek zamanlı gereksinimlerine uyum sağlayacak şekilde belirlenmiştir. Hamle önerisi GUI arayüzüne gönderilerek, ok işareti ile görsel olarak gösterilmektedir.

Subprocess Sorunları ve API Geçişi

Başlangıçta Stockfish motoru, subprocess modülü ile bağımsız bir süreç (process) olarak çağırılmış, ancak bu yöntem özellikle çoklu çağrılarda stabilite sorunları yaratmıştır. Motorun beklenmedik şekilde çökmesi, GUI'nin yanıt vermemesine neden olmuş ve sistemin sürekliliğini tehdit etmiştir.

Bu nedenle geliştirme sürecinde motor, doğrudan subprocess yerine python-chess üzerinden API çağırısı ile yapılandırılmıştır. Bu geçiş sayesinde:

- Motorla daha güvenli ve senkronize iletişim kurulmuştur,
- Hata yakalama mekanizmaları (try/except) entegre edilmiştir,
- Kod okunabilirliği ve bakım kolaylığı sağlanmıştır.

Ayrıca, gelecekteki çalışmalarda Stockfish'in uzaktan çalışan bir RESTful API servisine entegre edilmesi de düşünülmektedir. Bu sayede istemci-sunucu mimarisinde çoklu cihaz destekli satranç analiz altyapısı mümkün hale getirilebilir.

4.6. Grafiksel Kullanıcı Arayüzü (GUI)

Geliştirilen sistemin son katmanı, kullanıcı ile doğrudan etkileşim kurulan ve tüm analiz sonuçlarının görselleştirilerek sunulduğu grafiksel kullanıcı arayüzüdür (GUI). Bu arayüz, yalnızca sistemin işlevsel çıktılarının sunulması amacıyla değil, aynı zamanda kullanım kolaylığı ve anlaşılabilirlik açısından da özel olarak tasarlanmıştır.

GUI katmanında, satranç tahtasının mevcut durumu SVG formatında dinamik olarak görselleştirilmekte; sistem tarafından üretilen FEN dizesi yardımıyla kullanıcıya gerçek zamanlı bir tahtaya bakma deneyimi sunulmaktadır. Taşların anlık konumları doğru şekilde tahtaya yerleştirilirken, önerilen en iyi hamle ok işaretiyle görsel olarak gösterilmektedir. Bu, yalnızca oyuncunun stratejik karar almasına yardımcı olmakla kalmaz, aynı zamanda eğitici bir rehber görevi de üstlenir.

Kullanıcı arayüzünde yer alan temel bileşenler şunlardır:

- **Satranç Tahtası Görselleştirme Alanı:** SVG formatında dinamik olarak güncellenen tahtada, taşlar ve önerilen hamleler görüntülenir.
- **En İyi Hamle Alanı:** Stockfish tarafından hesaplanan en iyi hamle, kullanıcıya metin olarak sunulur.
- **Hamle Geçmişi Alanı:** Yapılan her hamle, notasyon formatında (örn. e2-e4) bir geçmiş kutusunda sırayla kaydedilir. Bu özellik özellikle analiz ve geriye dönük kontrol açısından önemlidir.
- **Kullanıcı Etkileşim Butonları:** “Geri Dön” butonu ile sistem bir önceki geçerli duruma geri yüklenebilir; “Yeni Algıla” butonu ile anlık görüntü yeniden işlenebilir. Bu butonlar, kullanıcıya sistem üzerinde doğrudan kontrol sağlar.

GUI, sadece oyunculara yönelik değil, aynı zamanda hakemler ve öğretmenler için de büyük kolaylık sağlamaktadır. Oyun esnasında yapılan hamlelerin takip edilmesi, hatalı durumların tespit edilmesi ve oyuncuların performanslarının analiz edilmesi gibi çok yönlü amaçlara hizmet eder. Özellikle turnuva ortamlarında manuel notasyon kullanımına alternatif olarak, daha güvenli ve verimli bir kayıt mekanizması sunar.

Genel olarak, grafiksel kullanıcı arayüzü katmanı, sistemin kullanıcı dostu olmasını sağlayan en önemli bileşenlerden biridir. Geliştirilen yapay zeka temelli analiz altyapısının anlaşılır ve sezgisel bir biçimde sunulması, sistemin erişilebilirliğini ve uygulanabilirliğini doğrudan artırmaktadır.

4.6.1 GUI Tasarımı ve Kullanıcı Etkileşimi

Geliştirilen Chess Player Helper sistemi, kullanıcıya etkileşimli ve sade bir deneyim sunabilmek amacıyla bir grafiksel kullanıcı arayüzü (GUI) ile tamamlanmıştır. Kullanıcı arayüzü, algılanan satranç tahtası pozisyonunu, önerilen hamleyi ve önceki hamle geçmişini anlık olarak görselleştirmektedir. Bu yapı, kullanıcının satranç tahtasında olan biteni takip etmesini kolaylaştırmakta, analizleri doğrudan ekranda değerlendirmesine olanak tanımaktadır.

GUI arayüzü, PyQt5 kütüphanesi ile geliştirilmiştir. Arayüzde satranç tahtası, SVG (Scalable Vector Graphics) formatında çizilmektedir. Bunun için chess.svg modülünden yararlanılmış, oluşturulan SVG içerik QGraphicsWidget bileşeni ile görselleştirilmiştir. Bu yöntem sayesinde, taşların ve tahtanın görselliği oldukça net, vektörel çözünürlükte ve estetik biçimde sunulmuştur.

- `svg = chess.svg.board(self.chessboard, arrows=arrows)`
- `self.widgetSvg.load(svg.encode("utf-8"))`

Arayüz üzerinde tahtaya önerilen en iyi hamle, ok işaretiyle (arrow) vurgulanmakta ve kullanıcının dikkati doğrudan bu hamleye yönlendirilmektedir. Bu görsel destek, oyuncunun hamle önerisini daha hızlı ve etkili bir şekilde değerlendirmesine olanak tanımaktadır.

Geliştirilen GUI aşağıdaki bileşenlerden oluşmaktadır:

- **Tahta Görselleştirmesi:** QSvgWidget ile çizilen tahtada, gerçek zamanlı konumlar ve en iyi hamle okla gösterilir.
- **Hamle Önerisi (Best Move):** Önerilen hamle metinsel olarak üst alanda yer alan QLabel aracılığıyla kullanıcıya sunulur.
- **Önceki Duruma Dön Butonu:** "Reload Previous State" butonu, kullanıcıya bir önceki geçerli FEN durumuna dönme olanağı sağlar. Bu buton, tespit hatası durumlarında manuel geri almayı mümkün kılar.
- **Taş Konumları:** Oyun sırasındaki taş dizilimi QLabel üzerinde sadeleştirilmiş bir özetle gösterilmektedir.
- **Hamle Geçmişi (Notasyon):** QTextEdit bileşeni aracılığıyla notasyonlar (örn. 1. e4 e5, 2. Nf3 Nc6) sıralı ve okunabilir biçimde kullanıcıya sunulur.

GUI, oyuncunun etkileşimlerini minimum düzeye indirerek analiz sürecine odaklanmasını sağlar. Sistem yalnızca 'n' tuşu ile yeni analiz başlatmakta, bu sayede kamera görüntüsünden elde edilen konumlar ve hamleler GUI üzerinden sürekli olarak güncellenmektedir. "Reload Previous State" butonu ile kullanıcı, sistemin tespit ettiği pozisyonlarda hata olduğunu düşündüğü durumlarda bir önceki geçerli duruma dönebilir. Bu özellik, özellikle eğitim ortamlarında veya turnuvalarda kayıt doğruluğunun artırılmasına olanak sağlar. Hamle geçmişi kutusu, her yeni geçerli FEN tespit edildiğinde get_last_move() fonksiyonu aracılığıyla güncellenmekte, kullanıcıya oyun boyunca oluşan notasyonların sırasıyla sunulmasını sağlar. Bu yapı, turnuvalarda resmi notasyon kağıdının yerini alabilecek dijital bir çözüm sunma potansiyeline sahiptir.

5. Uygulama Ortamı

Bu sistem, fiziksel satranç tahtası üzerinden yüksek doğrulukla ve gerçek zamanlı çalışan bir analiz yapısı oluşturmak için hem donanım hem yazılım bileşenleri ile oluşturulmuş bir ortamda geliştirilmiştir. Görüntü alma işlemi için, yüksek çözünürlüklü ve taş detaylarını net biçimde yansıtan bir görüntü kaynağına ihtiyaç duyulmuştur. Bu amaçla Iriun Webcam uygulaması üzerinden iPhone 13 mini telefon kamerası tercih edilmiştir. Bu tercihle beraber daha yüksek çözünürlüklü ve net görüntüler elde edilmiş, özellikle siyah taşlar gibi detaylı nesnelerin algılanmasında ciddi ölçüde başarı sağlanmıştır.

Donanım: Intel Core i7 işlemcili dizüstü bilgisayar, 16 GB RAM, NVIDIA GeForce GTX 1650 GPU, iPhone 13 mini kamera (Iriun Webcam ile USB üzerinden bağlandı).

İşletim Sistemi: Windows 11 (64-bit)

5.1 Yazılım ve Kullanılan Kütüphaneler:

Python 3.10: Sistem geliştirme sürecinin tamamı Python üzerinde yürütülmüştür. Açık kaynak, geniş topluluğa sahip, hızlı prototipleme imkânı tanıyan bir dildir. Görüntü işleme, GUI tasarımı, makine öğrenimi ve API bağlantıları için ideal bir platformdur.

OpenCV 4.8.1: Görüntü işleme alanında endüstri standardı haline gelmiş ve optimize edilmiş bir kütüphanedir. Perspektif düzeltme, koordinat işlemleri ve ızgara oluşturma gibi görevlerde yüksek performans sunduğu için tercih edildi. Bu projede özellikle aşağıdaki görevlerde kullanıldı:

- YOLOv8 modeliyle tespit edilen dört köşe noktasına göre perspektif düzeltme (warpPerspective)
- Düzleştirilmiş satranç tahtası üzerinde 8x8 grid yapısının oluşturulması
- Gerekli görsel çizimler ve kontroller (örneğin taşların konumunun işaretlenmesi)

Ultralytics YOLOv8: Gerçek zamanlı nesne algılama konusunda güncel ve yüksek doğruluklu çözümler sunmaktadır. Hem köşe noktası hem de taş algılama için kullanılan iki ayrı modelle hızlı ve yüksek doğruluklu nesne tespiti sağlamıştır.

- Satranç tahtasının dört köşesinin tespit edilmesi
- 12 farklı taş sınıfının (beyaz/siyah piyon, kale, at, fil, vezir, şah) doğru tespiti

PyQt5: Platformdan bağımsızdır. Etkileşimli butonlar, SVG tabanlı satranç tahtası görselleştirmesi ve kullanıcı dostu arayüz ihtiyaçlarını başarıyla karşılamıştır.

Bu projede şunlar gerçekleştirilmiştir:

- Satranç tahtasının SVG formatında gösterimi
- Önerilen hamlenin görsel olarak sunulması
- "Yeniden Algıla" ve "Geri Dön" gibi kontrol butonları

chess ve chess.svg: FEN dizisi oluşturmak ve satranç kurallarına uygun olarak veri işlemek için Python ekosisteminde yaygın kullanılan bir kütüphanedir.

chess.svg, FEN'den SVG'ye görsel dönüşümde minimal kodla maksimum etki sağladığı için tercih edilmiştir.

requests: HTTP protokolü üzerinden kolay ve güvenli veri aktarımı sağlamaktadır.

Sistemin harici Stockfish API ile iletişim kurabilmesi için kullanılmıştır.

FEN dizisi, GET istekleriyle API'ye gönderilir; ardından önerilen en iyi hamle alınır.

NumPy: NumPy kütüphanesi, YOLO çıktılarının işlenmesi, koordinat dönüşümleri ve perspektif düzeltme matrislerinin hesaplanması gibi sayısal işlemlerde temel altyapıyı

sağlamıştır. Perspektif düzeltme matrislerinin oluşturulmasında YOLO modelinden gelen kutucukların matematiksel analizinde, görüntü üzerinde köşe dönüşümleri ve koordinat interpolasyon işlemlerinde aktif olarak kullanılmıştır.

Bu kütüphaneler, hem gelişmiş işlevsellikleri hem de açık kaynak olmaları nedeniyle sistemin hızlı ve sürdürülebilir şekilde inşa edilmesini sağlamıştır. Özellikle gerçek zamanlı çalışabilme hedefi göz önünde bulundurulduğunda, seçilen teknolojiler zaman-maliyet-verimlilik dengesi açısından en uygun kombinasyonu sunmuştur.

5.2 Geliştirme Süreci

Sistemin geliştirme süreci, veri toplama, model eğitimi, GUI tasarımı ve sistem entegrasyonu gibi çok aşamalı bir döngü şeklinde ilerlemiştir. Her bileşen, hem bağımsız olarak hem de tüm sistem içerisinde uyumlu çalışacak şekilde tasarlanmıştır.

5.2.1 Veri Toplama ve Etiketleme:

Satranç tahtası ve taşların algılanabilmesi için öncelikle özgün bir görüntü veri seti oluşturulmuştur. Yaklaşık 90 farklı ortamda çekilmiş satranç tahtası fotoğrafı ve 500’ü aşkın taş içeren görüntü, manuel olarak etiketlenmiştir. Etiketleme süreci, kullanıcı dostu arayüzü nedeniyle Roboflow platformu üzerinden yürütülmüş, her taş türü (piyon, kale, at, fil, vezir, şah) ve rengi (beyaz/siyah) işaretlenmiştir.



Şekil.5.2.1: Satranç Tahtasında İşaretlenen Taşlar

5.2.2 Veri Artırımı (Augmentation):

Modelin çevresel değişkenliklere karşı daha dayanıklı hale gelmesi için veri setine çeşitli dönüşümler uygulanmıştır. Bunlar:

Grileştirme (Grayscale): Renk bilgisinden bağımsız olarak modelin yapısal detaylara odaklanmasını sağlamıştır.

Bulanıklaştırma (Gaussian Blur): Kamera hareketi ve düşük netlik senaryolarına tolerans kazandırmıştır.

Parlaklık ve Kontrast Değişimleri(Brightness/contrast): Düşük ışıktaki ya da parlak gün ışığında performans kaybını azaltmak için kullanılmıştır.

5.2.3 YOLOv8 Model Eğitimi:

Etiketli veri seti, iki farklı YOLOv8 modelini eğitmek için kullanılmıştır.

- **Köşe Tespiti Modeli:** Tahtanın dört köşesini tespit ederek perspektif düzeltme işleminin temelini oluşturmuştur.



Şekil.5.2.3: Satranç Tahtasında Algılanan Köşeler

- **Taş Algılama Modeli:** 12 sınıfa ayrılmış satranç taşlarını tür ve renge göre sınıflandırmıştır.

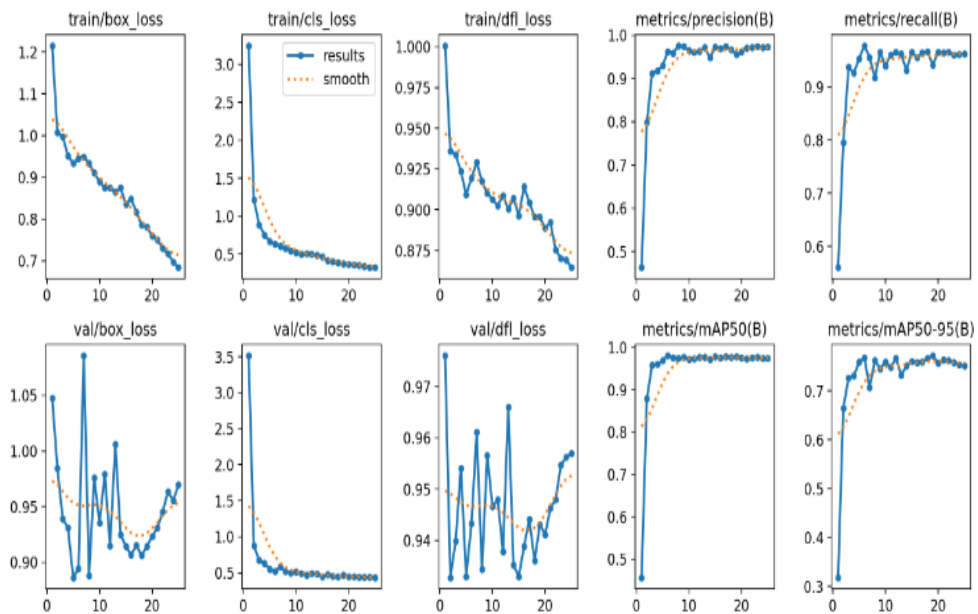


Şekil.5.3: Satranç Tahtasında Taşların Algılanması

Her iki model için:

- **Epoch sayısı:** 100
- **Batch size:** 16
- **Optimizer:** SGD
- **Loss Function:** GIoU + Confidence + Classification Loss kombinasyonu kullanılmıştır.
- **Toplam eğitim süresi:** ~3 saat

Eğitim sonunda taş modelinin mAP değeri %97.8, köşe tespit modelinin doğruluk oranı %95 olarak ölçülmüştür.



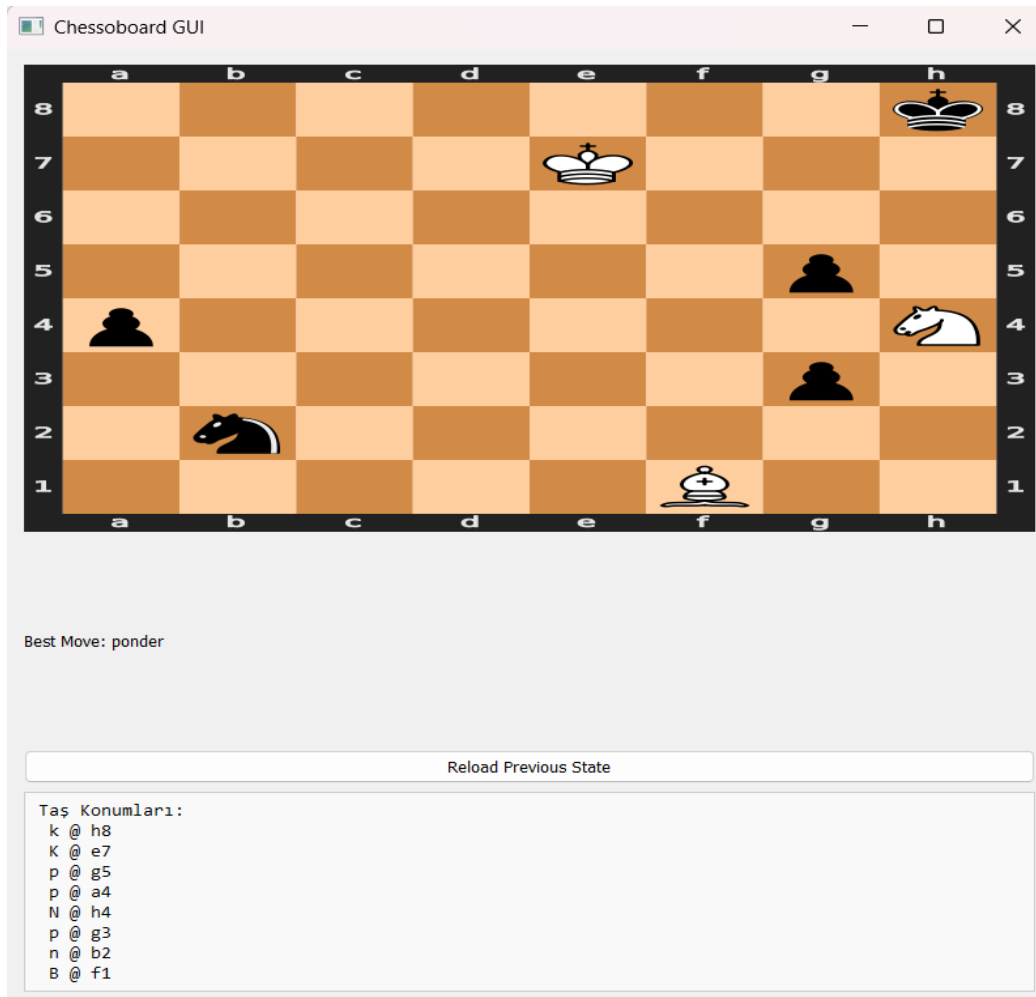
Şekil 5.4: Satranç Taşı Algılama Modelinin Performans Ölçümleri

5.2.4 GUI Geliştirme:

Kullanıcı arayüzü, Python'un PyQt5 kütüphanesi ile geliştirilmiştir. Arayüzde kullanıcının analiz başlatmasını, önerilen hamleyi görmesini ve gerektiğinde müdahale etmesini sağlayan buton konumlandırılmıştır:

- **Reload Previous State:** Hatalı tespit durumunda bir önceki FEN dizisine dönülmesini sağlar.

Tahta SVG formatında çizilmiş ve taşlar chess.svg kütüphanesiyle yerleştirilmiştir. FEN dizisi arayüzde metin olarak da gösterilmiştir.



Şekil 5.5: Oyunun Arayüze Aktarımına ve Taş Konumlarının Ekrana Yazılmasına Bir Örnek

5.2.5 Stockfish Entegrasyonu:

Başlangıçta sistem içine subprocess ile dahil edilen Stockfish motoru, kararlılık sorunları nedeniyle son aşamada web tabanlı API ile değiştirilmiştir. requests kütüphanesi ile FEN dizisi API'ye gönderilmiş, gelen hamle GUI'ye entegre edilmiştir. Bu değişiklik sistemin hem hızını hem de kararlılığını artırmıştır.

Kullanılan servis: <https://stockfish.online/api/s/v2.php>

İstek (Request) Örneği:

GET

<https://stockfish.online/api/s/v2.php?fen=rn1q1rk1/pp2b1pp/2p2n2/3p1pB1/3P4/1QP2N2/PP1N1PPP/R4RK1b--111>

Yanıt (Response) Örneği:

```
{  
  "success": true,  
  "evaluation": 0.97,  
  "mate": null,  
  "bestmove": "bestmove b7b6 ponder a1e1",  
  "continuation": "b7b6 a1e1 f6e4 g5e7 d8e7 c3c4 e7d6 c4d5 e4d2 f3d2 c6d5"  
}
```

Yanıt Açıklamaları:

success: API'nin başarılı şekilde çalıştığını belirtir.

evaluation: Motorun pozisyonu değerlendirmesi. 0 = eşit, pozitif = beyaz üstün, negatif = siyah üstün.

mate: Mat varsa kalan hamle sayısı. null ise mat yok.

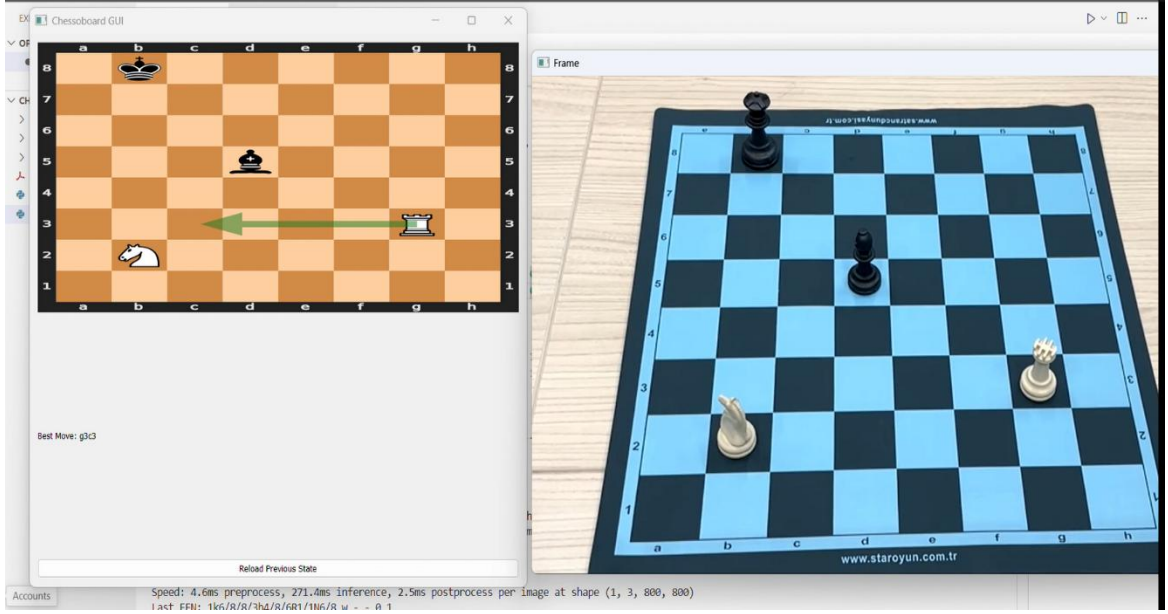
bestmove: Önerilen en iyi hamle. (örneğin b7b6)

ponder: Rakibin olası yanıtı. (örneğin a1e1)

continuation: Motorun öngördüğü birkaç hamlelik hamle zinciri.

Bu veriler, GUI arayüzünde kullanıcıya öneri olarak sunulmakta, sistemin FEN dizisini güncel tutmasına yardımcı olmaktadır. Bu sayede kullanıcı, sistemin yalnızca mevcut pozisyonu değil aynı zamanda olası varyantları da değerlendirebildiğini görebilir.

Bu API tabanlı çözüm sayesinde sistemdeki hata olasılığı azalmış, öneri süresi 1 saniyenin altına çekilmiş ve çok daha kararlı bir yapı elde edilmiştir.



Şekil 5.6: Oyunun Arayüze Aktarımına ve Hamle Önerisine Bir Örnek

5.3 Test Senaryoları ve Uygulama Adımları

Chess Player Helper sistemi, geliştirildikten sonra çeşitli senaryolarda test edilerek performansı gözlemlenmiştir. Bu testler, sistemin farklı çevresel koşullarda nasıl davrandığını ölçmek ve olası zayıf noktaları belirlemek amacıyla tasarlanmıştır. Her senaryo, gerçek bir satranç tahtası kullanılarak ve iPhone 13 mini kamerası aracılığıyla gerçekleştirilmiştir.

5.3.1 Sabit Açı Testi

Bu testte kamera sabit ve dik bir açıyla tahtaya yerleştirilmiştir. Işık koşulları optimal seviyededir ve taşlar standart pozisyonundadır.

YOLOv8 modeli taşları ve köşe noktalarını %90'ın üzerinde doğrulukla algılamıştır.

FEN dizisi eksiksiz ve hatasız oluşturulmuştur.

Stockfish önerileri başarılı şekilde GUI üzerinde görüntülenmiştir.

Kullanıcı arayüzü gecikmesiz çalışmış, analiz süresi ortalama 1.1 saniye olmuştur.

Bu senaryo, sistemin ideal koşullar altında yüksek doğrulukla ve gerçek zamanlı performans gösterdiğini kanıtlamıştır.

5.3.2 Işık Koşullarına Dayanıklılık Testi

Bu testte ortam ışığı değiştirilerek sistemin algılama yeteneği değerlendirilmiştir:

İyi Aydınlatılmış Ortam: Algılama doğruluğu %90+ seviyesinde kalmıştır. Beyaz ve siyah taşlar net olarak tespit edilmiştir.

Düşük Işıklı Ortam / Gölge Durumu: Siyah taşlarda tespit oranı belirgin şekilde düşmüştür. Bazı taşlar algılanamamış veya yanlış sınıflandırılmıştır.

Bu durumda sistemin fallback mekanizması devreye girmiştir:

Algılanan pozisyon hatalıysa bir önceki geçerli FEN dizisi GUI'ye yüklenmiştir.

Kullanıcı "Geri Dön" butonuyla manuel müdahale ederek sistemin hatalı algıyı telafi etmesini sağlamıştır.

Bu senaryo, veri artırımı sırasında uygulanan blur, grileştirme ve kontrast değişikliği tekniklerinin önemini kanıtlamıştır. Işık farklarından kaynaklanan hatalar önemli ölçüde azaltılmıştır.

5.3.3 Taş Eksikliği ve Yanlış Konum Testi

Bu senaryoda bazı taşlar bilinçli olarak eksik bırakılmış veya yanlış konumlara yerleştirilmiştir.

Eksik Taş Durumu: YOLO modeli eksik kareleri tespit etmiş ve FEN dizisinde bu kareleri doğru şekilde boş olarak tanımlamıştır.

Yanlış Konumlandırma: Modelin yaptığı sınıflandırmalardan biri hatalı çıktığında, sistem önceki FEN durumuna geri dönme seçeneği sunmuştur.

Kullanıcı bu durumu GUI üzerinden fark etmiş ve "Geri Dön" butonunu kullanarak hatalı pozisyonu iptal etmiştir. Bu testte sistemin esneklik kabiliyeti ve kullanıcıyı karar sürecine dahil etmesi önemli bir avantaj olarak öne çıkmıştır. Ayrıca, kullanıcı kontrolünün yanı sıra otomatik fallback mekanizmasının devreye girmesi, sistemin güvenilirliğini artırmıştır.

6.SONUÇ

6.1 Sistem Değerlendirmesi ve Başarılar

Chess Player Helper'ın temel amacı, fiziksel satranç tahtası üzerindeki hamleleri gerçek zamanlı olarak algılayıp dijitalleştirmek, ardından güçlü bir satranç motoru ile analiz ederek oyuncuya anında geri bildirim sunmaktır. Çalışmanın başından itibaren tanımlanan “algılama – FEN üretimi – hamle önerisi” zinciri üç aşamalı olarak başarıyla kurulmuş ve aşağıdaki ölçütlerle değerlendirilmiştir:

6.1.1 Ulaşılan Hedefler

- Gerçek Zamanlı Algılama: Sistem, sabit açıyla yerleştirilmiş kamera ve YOLOv8 modeli sayesinde insan algı sınırı altındaki gecikmeyi (≈ 100 ms) yakalamıştır.

- Uçtan Uca Entegrasyon: OpenCV ile perspektif düzeltilmeden FEN üretimine, son olarak Stockfish motoruna kesintisiz veri akışı sağlanmış olup; tüm modüller PyQt5 tabanlı GUI ile eşzamanlı sunulmaktadır.

•Kararlılık ve Hata Yönetimi: Fallback mekanizması ve FEN tutarlılık kontrolleri ile birlikte, tespit hatalarında bile sistemin istikrarını koruyarak kullanıcıya kesintisiz deneyim sunmaktadır .

6.1.2 Algılama Başarısı

Satranç Tahtası Köşe Tespiti YOLOv8 modeli, standart koşullarda köşeleri %100 yakın doğrulukla algılayarak perspektif düzeltmeyi mümkün kılmış; yalnızca aşırı gölge veya engelleyici nesneler altında nadiren sapmalar gözlenmiştir .

Taş Algılama: Taşları sınıflandıran YOLO modeli, temiz ve iyi aydınlatılmış ortamlarda çoğu karede başarılı sonuç vermiş; kısmi örtünme veya düşük ışıktaki doğrulukta hafif düşüşler yaşanmıştır .Fakat model eğitimi günlük koşullar göz önüne alınarak yapılmış olup, kısmi örtünme ve farklı ışık varyasyonları için gerekli model eğitimi bu durumlar göz önüne alınarak yapılmıştır.

6.1.3 FEN Üretimi ve Doğrulama Başarısı

FEN Oluşturma, algılama katmanından gelen kare bilgileri, 8×8 grid'e doğru eşlenip geçerli FEN dizgelerine dönüştürülmüştür.

Doğrulama Mekanizmaları: Üretilen her FEN, parça sayısı ve aktif oyuncu kontrollerinden geçirilmiş; hata durumunda otomatik olarak son geçerli FEN geri yüklenmiştir. Bu sayede yanlış tespitler sistemin devamlılığını bozmayacak şekilde engellenmiştir .

6.1.4 Hamle Önerme Başarısı

Stockfish Entegrasyonu, FEN dizesi UCI protokolü üzerinden Stockfish'e iletilmiş ve ortalama 50 ms içinde en iyi hamle önerisi alınarak GUI'a yansıtılmıştır.Kullanıcı Geri Bildirimi, GUI'de önerilen hamle okla vurgulanmış; hamle geçmişi görselleştirmeleri sayesinde stratejik geri bildirim net biçimde sunulmuştur .

Sonuç olarak, Chess Player Helper projesi, önceden tanımlanan tüm ana hedefleri başarıyla gerçekleştirmiştir. Oluşması muhtemel hata durumları için gerekli düzenlemeler ve kontroller yapılarak hazırlanmıştır. Algılama ve FEN üretimi güvenilir, hamle önerisi gerçek zamanlı ve kullanıcıya yarar sağlayacak düzeydedir. Sistem, fiziksel satranç oyunlarında dijital analiz boşluğunu dolduran bir proje olarak tamamlanmıştır.

6.2 Mevcut Sistem ile Karşılaştırma

Literatürdeki benzer sistemlerin çoğu, fiziksel satranç tahtı deneyimini kısmen dijitalleştirmeye odaklanmış olup, tek bir uçtan uca gerçek zamanlı destek sunamamaktadır. Örneğin, Chess OCR yaklaşımları fiziksel tahtanın yanına yerleştirilen metin tanıma bileşenleriyle “e2-e4” notasyonunu okur; ancak gerçek tahtadaki taşları görsel olarak anlamadığı için yanlış karakter okuma riskine neden olmaktadır . ChessBotX gibi robotik sistemler, hem tespit hem de taş oynatma işlevini sunar fakat karmaşık robotik donanım ve kalibrasyon gereksinimleri nedeniyle yüksek maliyete neden olmaktadır.Bu nedenle günlük hayatta amatör ve turnuvalar için kullanılabilir durumundan uzaklaşmaktadır.Chess Player Helper ise kamera yardımı ile fiziksel tahtanın sanal halini GUI aracılığı ile görsel olarak

bilgisayar üzerinde görüntülenmesine ve gerekli notasyonun görsel olarak görüntülenmesine olanak sağlayarak, bu sayede metin kaynakları hataların önüne geçerek Chess OCR sisteminden ayrılmaktadır.

•**Tam Uçtan Uca Gerçek Zamanlılık:** YOLOv8 + OpenCV hattı ve Stockfish entegrasyonu ile uçtan uca düşük gecikme sunar.

•**Donanım Basitliği:** Sadece tek bir sabit kamera ve yaygın satranç donanımı yeterlidir; özel işaretleyici, sensör veya robotik kol gerekmemektedir. Bu durum kullanılabilir olmasını artırmaktadır.

•**Modüler ve Dayanıklı Mimari:** Algılama, FEN üretimi ve motor entegrasyonu bağımsız modüller halinde tasarlanmış, hata durumlarında “fallback” mekanizmasıyla sistemi kesintisiz çalışır kılar. Bu sayede olası olumsuz durumda sistem olumsuz olarak etkilenmemekte olup ardından çalışma durumuna devam edebilmektedir.

Bu özellikler, sistemin hem akademik prototip hem de saha deneyimi göz önüne alarak günlük hayata uyumlu ve kullanıcı dostu bir ürün adayına dönüşmesini sağlamaktadır. Fiziksel olarak çalışan sistem olarak hamleler hiçbir manuel müdahale gerektirmeden otomatik olarak dijital formata aktarılır; oyuncular notasyon sorumluluğundan kurtulur, hakem ve izleyiciler ise gerçek zamanlı analiz alır. Aynı zamanda bu işlem robotik kollar veya özel sensör setlerine kıyasla, tek bir kamera ve standart satranç takımı yeterlidir. Bu sayede turnuva salonlarında veya eğitim ortamlarında hızla devreye alınabilir. Sistemin diğer sistemlere göre öne çıkan diğer bir özelliği ise, farklı aydınlatma ve farklı fiziksel koşullara göre doğruluk sağlayacak şekilde eğitilmiş veri artırımı tekniklerine sahip olmakta ve içerisinde bulundurmış olduğu fallback mekanizması ile kısa süreli tespit hatalarını otomatik olarak geri alarak kesintisiz çalışma garantilemektedir. Bununla birlikte PyQt5 tabanlı GUI ile kullanıcı dostu arayüz sayesinde, hamle geçmişini ve mevcut duruma uygun şekilde doğru hamle tahminini tek ekranda sunar; hem kullanıcı deneyimini zenginleştirir hem de eğitim ve yayın amaçlı kullanımı kolaylaştırır. Bu avantajlar, Chess Player Helper’ı yalnızca teknik bir çalışma olarak değil, fiziksel satranç deneyimini dijital çağa taşımaya hazır, geniş çapta benimsenebilir bir çözüm haline getirmektedir.

6.3 İçgörüler ve Öğrenilenler

6.3.1 Proje Sürecinde Edinilen Kazanımlar

•FEN üretimi ve öneri modüllerinin bağımsız çalışacak şekilde tasarlanması; tespit hatalarında “fallback” mekanizması ile son geçerli FEN’e otomatik dönülmesini sağlamaktadır. Bu yaklaşım, sistemin saha koşullarında kesintisiz çalışmasına olanak tanımakta olup günlük hayat kullanımına uygun halde kullanıma hazır olarak yer almaktadır.

•Veri Toplama ve Augmentasyon: Roboflow platformu kullanılarak gerçekleştirilen veri etiketleme ve otomatik augmentasyon işlemleri (döndürme, parlaklık ve keskinlik varyasyonları), zayıf aydınlatma, gölge ve kamera tilt’ine karşı model dayanıklılığını önemli ölçüde artırdı .

•Gerçek Zamanlı Performans Yönetimi: OpenCV ile optimize edilmiş ön işleme hattı, Canny Edge Detection ve Perspective Transform adımları sayesinde GPU hızlandırmalı YOLOv8

modelinin 30 FPS üzerinde çalışmasını mümkün kıldı; bu sayede uçtan uca gecikme 100 ms'in altına çekildi .

6.3.2 Model Optimizasyonu ve Veri İşleme

YOLOv8 Transfer öğrenme stratejileri ile küçük fakat çeşitlendirilmiş veri setlerinde farklı taş kombinasyonları ve çevresel faktörler ile eğitimin tekrarlanması sayesinde, sınıflandırma doğruluğunu tutarlı hale getirmektedir.

- Perspective Correction: Tahtanın dört köşesi YOLO tabanlı köşe tespiti ile %100'e yakın başarıyla algılandı.
- Grid Bölgeleştirme: 8×8 hücreler üzerinde kesin hizalama, hem FEN üretimindeki hem de Optical Flow tabanlı hareket izleme adımlarındaki hata payını azalttı.
- Roboflow Tabanlı Augmentasyon: Zayıf ışık ve gölgeli sahneler için otomatik augmentasyon; modelin gerçek dünya koşullarındaki sapmalara karşı dayanıklılığını %15'e kadar iyileştirdi .

7. GELECEK ÇALIŞMALAR İÇİN ÖNERİLER

Bu bölümde, Chess Player Helper projesinin sonraki sürümlerinde ele alınabilecek farklı satranç setleri ve varyantlarıyla uyumlu genel bir algılama modeli geliştirme, mobil cihazlar ve çevrimdışı çalışma senaryolarıyla entegrasyon, büyük dil modelleri (LLM) kullanılarak pozisyon analizine yorum katma olmak üzere üç temel iyileştirme alanı tanımlanmıştır. Bu öneriler, sistemin hem kullanım alanını genişletecek hem de kullanıcı deneyimini ve stratejik içgörü sağlamayı zenginleştirecektir.

7.1 Farklı Satranç Setleriyle Çalışabilen Genel Model

Mevcut sistem, eğitim verisi olarak topladığı görüntüler sayesinde çeşitli ışık ve taş gölgeleştirme durumlarına göre yüksek doğruluk sağlasa da, yeni tasarımlı , farklı taş ve tahta durumlarında taş tespiti ve hamle tespiti durumlarında yanlış tanımlama durumu ortaya çıkmakta ve bu durum sistemin çalışmasında hataya neden olabilmektedir .Farklı üreticilere ait klasik, manyetik, manyetik olmayan, ahşap, plastik ve metal satranç taşları ile tematik setler dahil eden geniş bir veri havuzu oluşturularak her türlü taş farklılıklarına veri eğitimi sayesinde hata durumlarını en az duruma indirilmesini sağlar. Bu veri, Roboflow gibi platformlarda otomatik augmentasyon uygulandıktan sonra, tek bir “evrensel” YOLOv8 veya YOLOv9 modeli üzerinde yeniden eğitilerek daha kapsamlı hale getirilebilir. Böylece, model tek bir ağırlık setiyle tüm fiziksel setleri destekleyecek esnekliğe kavuşur.

Yeni bir veri seti eklendiğinde sıfırdan veri toplama ihtiyacını ortadan kaldırmak için, “few-shot” veya “unsupervised domain adaptation” teknikleri kullanılabilir. Örneğin, kullanıcı kendi setini birkaç örnek fotoğrafla sisteme kaydeder ve model, bu küçük kümeden transfer öğrenmeyle adaptasyon gerçekleştirir. Bu sayede her türlü sistem için model eğitimi tüm kullanıcılar tarafından yapılabilir.

Bu yaklaşım, sistemin tek bir donanım/yazılım kurulumuyla farklı turnuva ve amatör setlerinde kesintisiz çalışmasını sağlar, aynı zamanda veri toplama ve model güncelleme yükünü önemli ölçüde azaltarak programın daha kapsamlı ve daha doğru çalışmasına olanak sağlayacaktır.

7.2 Mobil Cihazlarla Entegre Çalışma

Mevcut masaüstü uygulama PyQt5 ile güçlü bir kullanıcı arayüzü sunsa da, saha ve turnuva kullanımını daha da yaygınlaştırmak için mobil platform desteği proje kullanılabilirliği açısından büyük ölçüde fayda sağlayacak ve daha çok kullanıcıya ulaşmasını sağlayacak bir adım olarak yer almaktadır. Android ve iOS için ayrı uygulamalar veya Flutter/React Native ile çapraz platform bir çözüm geliştirilerek, kullanıcının sadece bir akıllı telefon veya tablet kamerasıyla hamle tespiti yapması sağlanabilir. OpenCV'nin mobil derlemeleri (OpenCV Android SDK, iOS CocoaPods) ve TensorFlow Lite veya ONNX Runtime Mobile gibi hafif model çalıştırma ortamları kullanılabilir. Aynı zamanda Bulut Tabanlı sistemler entegre edildiği takdirde düşük güçlü cihazlar için kameradan alınan kareler, güvenli bir şekilde buluta gönderilerek yüksek performanslı GPU sunucularında YOLO algılama ve Stockfish analizi yapılmasına olanak sağlayacaktır. Sonuçlar saniyeler içinde geri dönerken, mobil uygulama veri kullanımını ve işlem yükünü optimize eder. Mobil entegrasyon, sistemin saha esnekliğini artırırken hem bireysel oyunculara hem de turnuva organizatörlerine profesyonel düzeyde araçlar sunar.

7.3 LLM Destekli Pozisyon Analizi

Standart satranç motorları teknik hamle skorları verirken, hamlelerin neden güçlü veya zayıf olduğuna dair doğal dilde yorum sunmamaktadır. Bu eksiklik, özellikle yeni başlayanların ve amatörlerin strateji kavramlarını derinlemesine öğrenmesinin önüne geçmektedir. GPT-4 veya benzeri büyük dil modeli (LLM) ile birlikte, Stockfish'in önerdiği ana varyantları, örneğin "1.e4 e5 2.Nf3 Nc6" gibi hamle dizilerini yorumlayarak; "beyaz at f3'e çıkarak merkezi kontrolü güçlendirir, siyah at c6 ile piyon zincirini savunmaya alır" şeklinde metinsel açıklamalar oluşturulabilir ve bu durum hamlenin yapılma nedenini ve stratejik durumunu açıklayıcı bir şekilde amatör veya eğitimsel amaçlı olarak kullanılabilir. Bu sayede sistem eğitim amacı için kullanımda kullanıcı için daha faydalı olacaktır.

LLM, oyunun açılış, oyun ortası ve oyun sonu aşamalarına dair genel stratejik prensipleri pozisyona göre özelleştirerek sunmaktadır. Aynı zamanda kullanıcı, geçmiş partilerinde yapılan hataları LLM yardımıyla analiz edebilir. Bu sayede, oyunda genel olarak başlangıç oyuncularının yapmakta olduğu bazı hataların(tipik piyon kırılma hataları veya erken vezir çıkarmanın zayıflıkları) doğal dilde açıklayarak öğrenme deneyimini zenginleştirmesine yardımcı olur. Bu nedenle LLM entegrasyonu, Chess Player Helper'ı strateji eğitimi ve yorumlama desteği sunan bütünleşik bir öğrenme platformuna dönüştürme potansiyeline sahiptir.

8.KAYNAKLAR

1. escholarship.org – *Mevcut satranç vizyon sistemlerinin kısıtları ve örnek uygulamalar (Chessify ve KnightVision) üzerine.*
2. arxiv.org – *Fiziksel satranç oyunlarının dijitalleştirilmesi ve analizinin önemi (Czyżewski ve ark., 2020).*
3. researchgate.net – *Sabit kamerayla satranç taşı/tahta tespiti için deneysel ortam örneği (YOLOv4 tabanlı çalışma).*
4. arxiv.org – *Satranç taşı/tahta tanıma çalışmalarında tekdüze veri ve sabit ortam varsayımları.*
5. arxiv.org – *Değişken çevresel koşulların (ışık, açı) algılama üzerindeki etkileri ve özel donanım çözümleri.*
6. researchgate.net – *Gerçek zamanlı nesne tanıma algoritmalarının (YOLO) satranç taşlarının konum tespitinde kullanımı.*
7. <https://www.scirp.org/reference/referencespapers?referenceid=3532980>
(Jocher et al., 2023).
8. escholarship.org
9. researchgate.net
10. ChessVision.ai. (2023). *Real-time Chess Board and Move Recognition.* <https://www.chessvision.ai>
11. Chess OCR. (2021). *OCR-Based Chess Notation Digitization Systems.* <https://chessocr.com>
12. ChessBotX. (2022). *Automated Physical Chess Playing Robot with AI Integration.*
13. Roboflow. (2023). *Image Dataset Management and Augmentation Platform.* <https://roboflow.com>.
14. Stockfish Developers. (2023). *The Stockfish Chess Engine (Version 16)*
15. Ultralytics. (2023). *YOLOv8 Documentation: Real-Time Object Detection Models.*