MARMARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

NESNE TESPİTİ İLE GÖKTÜRKÇE METİNLERİN OKUNMASI

MEVLÜT KARAKAYA

YÜKSEK LİSANS TEZİ
Mekatronik Mühendisliği
Anabilim Dalı
Mekatronik Mühendisliği Programı

DANIŞMAN
Doç. Dr. Ahmet FEYZİOĞLU

EŞ-DANIŞMAN
Prof. Dr. Mesut ŞEN

İSTANBUL, 2024

MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES

READING GOKTURK SCRIPT USING OBJECT DETECTION

MEVLÜT KARAKAYA

MASTER THESIS

Department of Mechatronics Engineering

Thesis Supervisor
Doç. Dr. Ahmet FEYZİOĞLU

Thesis CO- Supervisor
Prof. Dr. Mesut ŞEN

ÖNSÖZ

Bu tez çalışmasının hazırlanmasında desteklerini ve bilgi birikimlerini esirgemeyen, yol göstericiliği ile beni her zaman motive eden kıymetli danışman hocam Doç. Dr. Ahmet Feyzioğlu'na en derin teşekkürlerimi sunarım. Çalışmam boyunca rehberliği ve önerileriyle katkıda bulunan eş danışmanım Prof. Dr. Mesut Şen hocama minnettarım. Ayrıca, tez sürecinde değerli fikirleri ve desteğiyle bana yardımcı olan Prof. Dr. Sezgin Ersoy hocama da teşekkürü borç bilirim.

Son olarak, bu zorlu süreçte her zaman yanımda olan ve bana güç veren sevgili aileme sonsuz teşekkürlerimi sunuyorum. Sizin desteğiniz olmadan bu başarı mümkün olmazdı.

Aralık 2024 **Mevlüt Karakaya**

İçindekiler

ÖNSÖZ ..... **i**

İÇİNDEKİLER ..... **ii**

ÖZET ..... **iv**

ABSTRACT ..... **v**

SEMBOLLER ..... **vi**

KISALTMALAR ..... **vii**

ŞEKİL LİSTESİ ..... **viii**

TABLO LİSTESİ ..... **ix**

1. GİRİŞ **1**

1.1. Çalışmanın Amacı ve Kapsamı 1

1.2. Göktürkçe'nin Tarihi ve Önemi 2

2. LİTERATÜR TARAMASI **3**

2.1. Göktürkçe Metinlerin İncelenmesi 3

2.2. Derin Öğrenme ve Nesne Tespiti Teknolojileri 4

2.3. Antik Dillerin Okunmasında Yapay Zeka Uygulamaları 5

3. VERİ SETİ OLUŞTURMA **7**

3.1. Göktürkçe Harf Verilerinin Toplanması 7

3.2. Görüntülerin İşlenmesi 8

3.3. Veri Setlerinin Etiketlenmesi ve Hazırlanması 11

4. NESNE TESPİTİ ALGORİTMALARI **12**

4.1. Convolutional Neural Networks (CNN) 12

4.2. You Only Look Once (YOLO) Modeli 16

4.2.1. YOLOv8 18

4.3. Veri Artırma Teknikleri 19

5. MODEL EĞİTİMİ VE DEĞLENDİRME **20**

5.1. Eğitim Süreci ve Parametreler 20

5.2. Model Performans Kriterleri 21

5.2.1. AdamW optimizasyonun avantajları 25

5.3. Yanıltcı Pozitif ve Negatif Tespitler 26

5.3.1. Yanıltıcı pozitif tespitler 26

5.3.2. Yanıltıcı negatif tespitler 27

5.3.3. Sonuç ve iyileştirme yöntemleri 28

5.4. Sonuçların Yorumlanması 28

5.4.1. Modelin performansı 29

5.4.2. Eğitim süreci 29
5.4.3. Sınıf bazında performans 30
5.4.4. Geliştirme alanları 32
6. BULGULAR VE TARTIŞMA 33
6.1. Göktürkçe Harf ve Kelime Tespiti Sonuçları 33
6.1.1. Model eğitimi ve hyperparametre ayarları 33
6.1.2. Eğitim performansı ve model genel başarısı 34
6.1.3. Harf ve kelime sınıflandırılmasında model performansı 34
6.1.4. Modelin genel güçlü yanları ve zayıflıkları 35
6.2. Araştırmanın Sınırlılıkları 36
7. SONUÇ VE ÖNERİLER 40
7.1. Araştırmanın Genel Değerlendirmesi 40
7.2. Gelecek Çalışmaları İçin Öneriler 42
8. KAYNAKÇA 44
ÖZGEÇMİŞ viii

ÖZET

NESNE TESPİTİ İLE GÖKTÜRKÇE METİNLERİN OKUNMASI

Bu yüksek lisans tezi, Göktürkçe metinlerin okunması ve yorumlanmasında nesne tespiti teknolojilerinin kullanılmasını araştırmaktadır. Göktürkçe, Türk tarihinin en eski yazılı dillerinden biri olup, özellikle Orhun Yazıtları gibi tarihi belgelerde yer almaktadır. Ancak, bu metinlerin okunması ve analiz edilmesi, dilin karmaşıklığı ve yazılı taşların aşınmışlığı nedeniyle zorluklar içermektedir.

Bu çalışmada, derin öğrenme tekniklerine dayalı nesne tespiti algoritmaları kullanılarak Göktürkçe harflerin ve kelimelerin otomatik olarak tespit edilmesi ve tanınması hedeflenmiştir. Bu amaçla, öncelikle Göktürkçe yazıtların yüksek çözünürlüklü görüntüleri toplanmış ve bu görüntüler üzerinde veri setleri oluşturulmuştur. Veri setleri, nesne tespiti modellerinin eğitimi için etiketlenmiş ve hazırlanmıştır.

Eğitim sürecinde, Convolutional Neural Networks (CNN) ve You Only Look Once (YOLO) gibi ileri seviye derin öğrenme modelleri kullanılmıştır. Modellerin performansı, doğru tespit ve tanıma oranları, yanıltıcı pozitif ve negatif tespitler gibi kriterler üzerinden değerlendirilmiştir. Ayrıca, nesne tespiti sonuçlarının doğruluğunu artırmak için veri artırma teknikleri ve transfer öğrenme stratejileri uygulanmıştır.

Araştırma sonuçları, nesne tespiti algoritmalarının Göktürkçe metinlerin okunmasında etkin bir şekilde kullanılabileceğini göstermektedir. Elde edilen bulgular, tarihi metinlerin dijitalleştirilmesi, korunması ve incelenmesi konularında önemli katkılar sunmaktadır. Bu çalışma, ayrıca, benzer antik dillerin incelenmesi ve okunması için bir model teşkil etmekte olup, arkeoloji ve dilbilim alanlarında yeni araştırmalara ilham verebilir.

Sonuç olarak, nesne tespiti teknolojileri, tarihsel ve dilsel araştırmalarda yeni bir perspektif sunarak, Göktürkçe metinlerin okunması ve anlaşılmasında önemli bir araç olarak öne çıkmaktadır. Bu tez, bu alandaki mevcut literatüre katkı sağlamayı ve gelecekteki çalışmalara temel oluşturmayı amaçlamaktadır.

ABSTRACT

READING GOKTURK SCRİPT USING OBJECT DETECTION

This master's thesis investigates the use of object detection technologies in reading and interpreting Gokturk texts. Gokturk is one of the oldest written languages in Turkish history, notably found in historical documents such as the Orkhon Inscriptions. However, reading and analyzing these texts is challenging due to the complexity of the language and the erosion of the inscribed stones.

In this study, object detection algorithms based on deep learning techniques are utilized to automatically detect and recognize Old Turkic letters and words. To achieve this, high-resolution images of Old Turkic inscriptions were first collected, and datasets were created from these images. The datasets were labeled and prepared for training object detection models.

During the training process, advanced deep learning models such as Convolutional Neural Networks (CNN) and You Only Look Once (YOLO) were used. The performance of the models was evaluated based on criteria such as accurate detection and recognition rates, and false positive and negative detections. Additionally, data augmentation techniques and transfer learning strategies were applied to improve the accuracy of the object detection results.

The research findings demonstrate that object detection algorithms can be effectively used in reading Old Turkic texts. The results provide significant contributions to the digitization, preservation, and analysis of historical texts. This study also serves as a model for examining and reading similar ancient languages, potentially inspiring new research in archaeology and linguistics.

In conclusion, object detection technologies offer a new perspective in historical and linguistic research, emerging as an essential tool in reading and understanding Old Turkic texts. This thesis aims to contribute to the existing literature in this field and lay the groundwork for future studies.

SEMBOLLER

Sa : Satır sayısı

Su : Sütün sayısı

p(C) **:** Nesne varlık olasılığı

bx : Nesne kutusu (Bounding box) x koordinatı

by : Nesne kutusu (Bounding box) y koordinatı

bw : Nesne kutusu (Bounding box) genişlik bilgisi

bw : Nesne kutusu (Bounding box) yükseklik bilgisi

ci : Nesnenin sınıfına ait olasılık

L_{oos} : Hesaplanan bounding box hata değeri

$1_{\{ij\}}^{\{obj\}}$: İlgili hücrede nesnenin olup olmadığını belirten gösterge fonksiyonu

λ_{coord} : Koordinat hatası için kullanılan bir ağırlık faktörü

x_i : Tahmin edilen ve gerçek nesne kutusu (bounding box) x pozisyonu

y_i : Tahmin edilen ve gerçek nesne kutusu (bounding box) y pozisyonu

w_i : Tahmin edilen ve gerçek nesne kutusu (bounding box) genişlik bilgisi

h_i : Tahmin edilen ve gerçek nesne kutusu (bounding box) yükseklik bilgisi

g_t : t'ninci iterasyondaki hesaplanan gradyan değeri

$L(\theta_t)$: Hesaplanan gradyanın kayıp fonksiyonu

(θ_t) : Model parametresi

m_t : Gradyanın hareketli ortalamasını temsil eder

β_1 : Moment parametresi

v_t : Gradyanın karelerinin hareketli ortalamasını temsil eder

β_2 : İkinci moment parametresi

$\hat{m}_t$: Gradyanın hareketli ortalamasının yanlılık düzeltmesi

$\hat{v}_t$: Gradyanın karelerinin hareketli ortalamasının yanlılık düzetilmesi

$\theta_{\{t+1\}}$: AdamW ağırlık güncelleme terimi

η : Öğrenme oranı

λ : Ağırlık çürümesi (Weight decay)

$\in$: Sayısal kararlılığı artırmak için kullanılan parametre

KISALTMALAR

CNN **:** Convolutional Neural Networks

YOLO **:** You Only Look Once

NLP **:** Neural Language Prosesing

R-CNN **:** Region-Based Convolutional Neural Networks

NMS **:** Non-Maximum Suppression

SGD **:** Stokastik Gradyan İniş

AdamW **:** Adaptive Moment Estimation with Weight Decay

RNN **:** Recurrent Neural Networks

LSTM : Long Short-Term Memory

ReLU : Rectified Linear Unit

NMS : Non-Maximum Suppression

mAP : Mean Average Precision

IoU : Intersection Over Union

ŞEKİL LİSTESİ

Şekil 2.1 Kül Tigin Yazıtı ....…… 3
Şekil 3.1 Göktürkçe M harfine ait veri görselleri ……. 7
Şekil 3.2 Görüntü işleme python fonksiyonu ……... 9
Şekil 3.3 Veri kümesi görsel artırma işlemi ……10
Şekil 3.4 Görüntü işlenmesi ve artırma işlemi yapıldıktan sonra veri setinin görseli ..10
Şekil 3.5 Etiketleme işlemine ait örnek görsel ……11
Şekil 3.6 Etiketmiş görsele ait etiket bilgilerinin yer aldığı metin belgesinin görseli..11
Şekil 4.1 CNN çalışma mantığı ……12
Şekil 5.1 SGD Optimizasyon algoritmasının doğruluk değişim grafiği …… 22
Şekil 5.2 AdamW optimizasyon algoritmasına göre doğruluk değişim grafiği ..…..... 22
Şekil 5.3 SGD Optimizasyon algoritmasının hata değişim grafiği ...……..… 22
Şekil 5.4 AdamW optimizasyon algoritmasının hata değişim grafiği……. 22
Şekil 5.5 Model Eğitim Sonuçları…….…... 28
Şekil 5.6 Orjinal Orhun yazıtından alınan bir metnin analiz edilmesi……..... 30
Şekil 5.7 ŞER kelimesinin analiz sonucu…….… 30
Şekil 5.8 M harfinin analiz sonucu...……..……. 31

TABLO LİSTESİ

Tablo 4.1 CNN modellerinin özellikleri……. 13
Tablo 5.1 Optimizasyon algoritmalarının doğruluk ve hata değerleri tablosu……. 21

1. GİRİŞ

Göktürkçe, Türk dilinin en eski yazılı biçimlerinden biri olarak Orhun Yazıtları gibi tarihi belgelerde yer almakta ve Türk tarihi açısından büyük bir öneme sahiptir. Ancak, bu metinlerin okunması ve yorumlanması hem dilin karmaşıklığı hem de taşların aşınmışlığı nedeniyle ciddi zorluklar içermektedir. Bu tezde, derin öğrenme tabanlı nesne tespiti teknolojileri kullanılarak Göktürkçe harflerin ve kelimelerin otomatik olarak tespit edilmesi ve tanınması hedeflenmiştir. Bu yaklaşımla, tarihsel metinlerin dijitalleştirilmesi, korunması ve daha geniş bir kitle tarafından erişilebilir hale getirilmesi amaçlanmaktadır.

1.1. Çalışmanın Amacı ve Kapsamı

Bu çalışmanın temel amacı, Göktürkçe metinlerin okunması ve analiz edilmesinde nesne tespiti teknolojilerinin kullanımını araştırmaktır. Göktürkçe, Türk dilinin en eski yazılı biçimlerinden biri olup, Orhun Yazıtları gibi tarihi metinlerde yer almaktadır [1]. Ancak, bu metinlerin okunması ve doğru bir şekilde yorumlanması, zamanla taşların aşınması ve dilin yapısal karmaşıklığı nedeniyle oldukça zordur. Bu bağlamda, derin öğrenme tabanlı nesne tespiti algoritmaları, Göktürkçe harflerin ve kelimelerin otomatik olarak tespit edilmesi ve tanınması için kullanılmıştır. Çalışma, bu teknolojilerin tarihi metinlerin dijitalleştirilmesi, korunması ve daha geniş bir araştırma kitlesine sunulması için nasıl kullanılabileceğini incelemektedir.

Bu araştırma, öncelikle Göktürkçe yazıtların yüksek çözünürlüklü görüntülerinin toplanması ve bu görüntüler üzerinde veri setlerinin oluşturulması ile başlamıştır. Ardından, Convolutional Neural Networks (CNN) ve You Only Look Once (YOLO) gibi ileri seviye derin öğrenme modelleri kullanılarak bu veri setleri üzerinde eğitimler yapılmıştır. Çalışmanın sonuçları, nesne tespiti algoritmalarının Göktürkçe metinlerin okunmasında ne kadar etkili olabileceğini ve bu yaklaşımla elde edilen sonuçların tarihsel araştırmalara nasıl katkı sağlayabileceğini göstermektedir. Bu bağlamda, çalışma hem arkeoloji hem de dilbilim alanlarında önemli bir yenilik sunmayı hedeflemektedir.

1.2. Göktürkçenin Tarihi ve Önemi

Göktürkçe, Türklerin bilinen en eski yazılı dillerinden biridir ve Türk tarihinin en önemli kültürel miraslarından biri olarak kabul edilir [2]. Bu dil, 6. yüzyılda Göktürk Kağanlığı döneminde kullanılmıştır ve özellikle Orhun Yazıtları ile ünlüdür [1]. Göktürk alfabesiyle yazılmış bu yazıtlar, Türklerin yazılı edebiyatının başlangıcını simgeler ve Türk dili, tarihi, kültürü ve sosyal yapısı hakkında çok değerli bilgiler sunar [3].

Orhun Yazıtları, Bilge Kağan, Kül Tigin ve Tonyukuk adlarına dikilmiş olan taş anıtlar üzerinde yer alır ve bu yazıtlar, Türk milletinin o dönemdeki tarihini, devlet yapısını, savaşlarını ve yöneticilerinin başarılarını anlatır [1]. Göktürkçe, bu metinler aracılığıyla sadece bir dil olarak değil, aynı zamanda Türklerin dünyaya bakışını, inançlarını ve sosyal değerlerini de yansıtan bir kültürel miras olarak kabul edilir [2].

Göktürkçe, dilin yapısı ve alfabesi itibarıyla, tarihsel dilbilim açısından büyük bir öneme sahiptir [4]. Bu dilin kullanıldığı dönemdeki toplumun kültürel ve sosyal yaşamını anlamak, bu yazıtların okunması ve incelenmesi ile mümkün olmuştur [5]. Göktürkçe'nin çözümlenmesi, Türk dilinin evrimi ve diğer Türk dilleri ile olan bağlantılarının ortaya konulması açısından da önemlidir. Bu nedenle, Göktürkçe metinler, Türkoloji araştırmalarının temel kaynakları arasında yer alır ve Türk tarihi ve kültürü açısından büyük bir değere sahiptir [2].

2. LİTERATÜR TARAMASI

2.1. Göktürkçe Metinlerin İncelenmesi

Göktürkçe metinlerin incelenmesi, Türkoloji alanında önemli bir araştırma konusu olarak öne çıkmıştır. Bu metinlerin dil yapısı, söz varlığı ve tarihi bağlamı üzerine yapılan çalışmalar, Göktürkçe'nin diğer Türk dilleri ile olan bağlantılarını ve Türk dilinin tarihsel gelişimini anlamak için kritik veriler sunmaktadır [3]. Özellikle Orhun Yazıtları, Göktürkçe'nin yapısal özelliklerini ortaya koyan en önemli kaynaklar arasında yer alır. Bu yazıtlar üzerinde yapılan dilbilimsel analizler, dilin morfolojik ve sentaktik yapısını detaylı bir şekilde incelemiş ve Türk dilinin tarih boyunca geçirdiği evrim hakkında önemli ipuçları sağlamıştır [1].

Göktürkçe metinlerin okunması ve yorumlanması, taşların aşınmışlığı ve metinlerin bazen eksik olması nedeniyle zorluklar içermektedir. Ancak, çeşitli filolojik ve epigrafik yöntemler kullanılarak bu yazıtlar büyük ölçüde çözümlenmiştir [5]. Metinlerin içerdiği tarihi bilgiler, Göktürklerin sosyal ve politik yapısı hakkında da önemli veriler sunmaktadır. Örneğin, Bilge Kağan ve Kül Tigin yazıtları, Göktürk Kağanlığı'nın iç ve dış politikalarına dair detaylı bilgiler içerir ve bu yazıtlar, Türklerin erken dönem tarihini anlamak için kritik öneme sahiptir [2]. Göktürkçe metinler, sadece bir dil olarak değil, aynı zamanda Türk toplumunun inançlarını, değerlerini ve kültürel yapısını yansıtan birer tarihi belge olarak da incelenmiştir [4]. Aşağıdaki şekil 2.1'de yer alan görsel Orhun Yazıtlarından Kül Tigin Yazıtı yer almaktadır. Bu yazıt Moğolistan da Orhun Yazıtları müzesinde sergilenmektedir [6].



Şekil 2.1 Kül Tigin Yazıtı [6]

2.2. Derin Öğrenme ve Nesne Tespiti Teknolojileri

Derin öğrenme, yapay zekâ alanındaki en güçlü yaklaşımlardan biri olarak, büyük veri setlerinden karmaşık örüntüleri öğrenme yeteneğine sahiptir. Bu teknoloji, özellikle görüntü işleme, ses tanıma, doğal dil işleme gibi alanlarda büyük başarılar elde etmiştir. Derin öğrenme modelleri, çok katmanlı yapay sinir ağları kullanarak, verilerin özelliklerini otomatik olarak çıkarır ve bu özellikler üzerinden tahminler yapar. Özellikle Convolutional Neural Networks (CNN) gibi mimariler, görüntü verilerini işleyerek nesne tespiti ve sınıflandırma alanlarında önemli sonuçlar elde etmiştir [7].

CNN'lerin yanı sıra, Recurrent Neural Networks (RNN) ve bu yapının güncellenmiş halleri olan Long Short-Term Memory (LSTM) ağları, zamana dayalı veri analizinde başarılı sonuçlar vermiştir. Bu yapılar, sıralı verilerde ardışık bağıntıları yakalayarak tahminlerde bulunabilir [8].

Nesne tespiti teknolojileri, bir görüntüdeki belirli nesneleri tanımlama ve yerlerini belirleme sürecini ifade eder. Bu teknolojiler, güvenlik kameraları, otonom araçlar, tıbbi görüntüleme ve endüstriyel otomasyon gibi birçok alanda kullanılmaktadır. Derin öğrenme tabanlı nesne tespiti algoritmaları, büyük ölçüde CNN'ler üzerine kuruludur ve bu algoritmalar, görüntüdeki farklı nesneleri belirlemek için çeşitli filtreler ve katmanlar aracılığıyla öğrenme gerçekleştirir. Örneğin, You Only Look Once (YOLO) modeli, tek bir sinir ağı geçişiyle görüntüdeki tüm nesneleri hızlı ve doğru bir şekilde tespit edebilir [9].

Derin öğrenme ve nesne tespiti teknolojilerinde transfer öğrenme ve veri artırma stratejileri de önemli bir rol oynamaktadır. Transfer öğrenme, önceden eğitilmiş bir modelin başka bir görev için yeniden kullanılmasıdır ve bu, özellikle sınırlı veri setlerine sahip projelerde model performansını artırmak için yaygın olarak kullanılır. Veri artırma ise mevcut veri setinin çeşitli dönüşümlerle genişletilmesi anlamına gelir. Bu stratejiler, modellerin genelleme yeteneğini artırarak, nesne tespitindeki doğruluk oranlarını iyileştirir [10].

Son yıllarda, derin öğrenme tabanlı nesne tespiti algoritmalarının daha hızlı ve daha verimli hale getirilmesi için çalışmalar sürmektedir. Özellikle, hafif ve taşınabilir cihazlarda kullanılabilecek modellerin geliştirilmesi üzerinde durulmaktadır. Bu bağlamda, MobileNets ve EfficientDet gibi modeller, sınırlı donanım kaynaklarına sahip cihazlarda bile yüksek performanslı nesne tespiti sağlayabilmektedir [11-12]. Bu gelişmeler, derin öğrenmenin ve nesne tespitinin günlük hayatın her alanında daha yaygın bir şekilde kullanılmasına olanak tanımaktadır.

2.3. Antik Dillerin Okunmasında Yapay Zeka Teknolojileri

Antik dillerin okunması, özellikle modern dilbilimciler ve tarihçiler için zorlayıcı bir süreçtir. Bu dillerin büyük bir kısmı günümüzde kullanılmadığı ve belgelerin çoğu eksik ya da aşınmış olduğu için, yapay zeka teknolojileri bu tür metinlerin okunmasında umut verici çözümler sunmaktadır. Yapay zeka ve özellikle derin öğrenme, antik yazıtların otomatik olarak tanınması, transkripsiyonu ve tercümesi için kullanılan modern araçlardan biridir. Görüntü işleme ve doğal dil işleme (NLP) teknikleri, antik metinlerin dijital ortamlara aktarılmasını kolaylaştırmaktadır [13].

Yapay zeka algoritmaları, antik metinlerdeki karakterleri tanımak ve bu karakterleri modern dillere çevirmek için kullanılmaktadır. Özellikle Convolutional Neural Networks (CNN) gibi görüntü tabanlı derin öğrenme algoritmaları, taş, papirüs veya parşömen gibi yüzeyler üzerinde yer alan antik harfleri tanımlamakta etkili olmaktadır. Bu tür sistemler, karakterlerin aşınmış, silik ya da eksik olduğu durumlarda bile yüksek doğruluk oranlarına ulaşabilirler. Antik Mısır hiyerogliflerinden Akad çivi yazısına kadar birçok antik yazı sisteminde bu teknolojiler başarıyla kullanılmıştır [14].

Bununla birlikte, yapay zeka yalnızca harfleri tanımlamakla kalmaz, aynı zamanda bu harflerin oluşturduğu kelime ve cümleleri de anlamaya çalışır. Bu amaçla, derin öğrenme tabanlı doğal dil işleme teknikleri kullanılarak antik metinlerdeki dil yapısı

analiz edilir ve metinlerin anlamı çıkarılır. Bu tür teknolojiler, özellikle metinlerin eksik veya yanlış olduğu yerlerde tahminler yaparak bilim insanlarına büyük katkılar sunmaktadır. Örneğin, antik Yunanca ve Latince metinlerde NLP teknikleri ile büyük başarı elde edilmiştir [15].

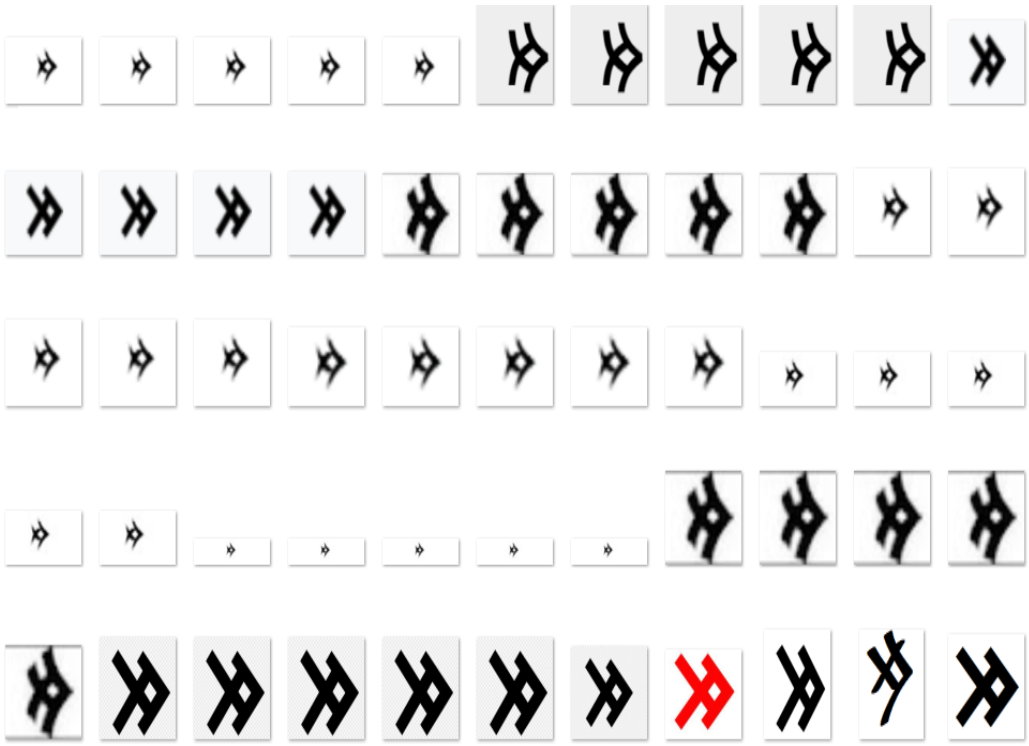
Antik dillerin okunmasında yapay zeka uygulamaları, sadece bilimsel araştırmalar açısından değil, kültürel mirasın korunması ve dijitalleştirilmesi için de büyük öneme sahiptir. Tarihi belgeler ve metinler, yapay zeka algoritmaları sayesinde dijital olarak arşivlenebilir ve geniş kitlelerle paylaşılabilir hale gelmektedir. Bu da antik medeniyetlerin dil, kültür ve tarihini anlamak için yeni bir pencere açmakta ve bu bilginin gelecek nesillere aktarılmasına yardımcı olmaktadır [16].

3. VERİ SETİNİN OLUŞTURULMASI

Göktürk alfabesi, Türk kültürünün önemli bir mirası olarak, özellikle Orhun Yazıtları gibi tarihi metinlerde kullanılmıştır. Bu alfabedeki harflerin toplanması, nesne tespiti teknolojileriyle otomatik okuma ve analiz süreçlerinin temelini oluşturan bir adımdır. Harf verilerinin doğru ve etkili bir şekilde toplanması, algoritmaların başarıyla eğitilmesi için kritik öneme sahiptir.

3.1. Göktürkçe Harf Verilerinin Toplanması

Veriler toplanırken Orhun kitabelerinde yer alan taramalar ve girilen latin harflerini Göktürkçe metinlere çeviren web sitelerinden faydalanıldı. Bu veriler olabildiğince yüksek çözünürlükte olmasına ve içinde geçen harflerin doğru bir şekilde anlaşılabilmesine büyük bir önem gösterildi. İlk aşamada Göktürkçede kullanılan harfler belirlenmiş ve bu harfler tek tek geçtiği her metinden örnekleri alınıp kaydedildi. Aşağıdaki Şekil 3.1'de yer alan görselde Göktürkçe M harfine ait toplanan veriler yer almaktadır. Göktürkçe 38 harften oluşmaktadır [4]. Tüm harfler bu şekilde tek tek toparlanarak veri kümeleri oluşturulmuştur.



Şekil 3.1 Göktürkçe M harfine ait veri görselleri

3.2. Görüntülerin İşlenmesi

Görüntülerin kalitesini artırmak ve eğitimin daha iyi sonuçlar alabilmesini sağlamak için görüntüler üzerinde işleme yapılmıştır. Bu sayede veri kümesindeki eleman sayısı artırılmıştır. İlk aşamada görünütüye otomatik yönlendirme(Auto-Orient) yapılmıştır. Bu adım, görüntülerin doğru bir şekilde görüntülenmesini sağlar. Örneğin, dikey çekilmiş bir fotoğraf yanlışlıkla yatay görüntüleniyorsa, otomatik yönlendirme bu hatayı düzeltir. Bu sayede, eğitim için kullanılan veri setindeki görüntülerde tutarlılık sağlanır. Ardından tüm görsellerin tek bir boyutta olmasını sağlamak için yeniden boyutlandırma işlemi yapıldı. Görüntülerin aynı boyuta getirilmesi, modelin eğitimi sırasında büyük bir avantaj sağlar. Sabit boyutlu görüntüler, veri setinin işlenmesini hızlandırır ve derin öğrenme algoritmaları için gereksiz hesaplamaları azaltır. Ayrıca, farklı boyutlardaki görüntülerin modele farklı etkiler yapmasının önüne geçer, daha tutarlı sonuçlar alınmasına yardımcı olur. Onun ardından gri tonlama (Grayscale) işlemi yapılmıştır. Görüntüyü gri tonlamaya dönüştürmek, veri setini sadeleştirir ve işlem yükünü azaltır. Birçok nesne tanıma problemi, renk bilgisine ihtiyaç duymadan çözülebilir. Böylece model, sadece gerekli olan yapısal bilgilere odaklanır. Aynı zamanda eğitim süresini kısaltır ve modelin performansını artırır. Son aşamada ise kontrast ayarlama (Adaptive Equalization) işlemi yapıldı. Adaptif histogram eşitlemesi (adaptive equalization) kullanarak görüntüdeki kontrastı iyileştirmek, karanlık veya fazla aydınlık alanları dengeler. Bu işlem, özellikle düşük ışık veya zayıf kontrastlı görüntülerde detayların daha iyi fark edilmesini sağlar. Sonuç olarak, modelin bu bölgelerdeki nesneleri daha doğru tanıyabilmesi için görüntü kalitesi artırılır. Bu işlemler için bir python fonksiyonu yazılıdı. Fonksiyon ait görsel aşağıdaki şekildeki gibidir.

```python
from PIL import Image, ImageOps
from skimage import exposure
import numpy as np

def preprocess_image(image_path, output_path):
    # Görüntüyü Pillow kullanarak yükle
    img = Image.open(image_path)

    # Adım 1: Otomatik Yönlendirme (EXIF verisine göre döndürme)
    img = ImageOps.exif_transpose(img)

    # Adım 2: Görüntüyü 640x640 piksel boyutuna yeniden boyutlandırma
    img = img.resize((640, 640))

    # Adım 3: Görüntüyü gri tonlamaya çevirme
    img = img.convert("L")

    # Adım 4: Kontrastı adaptif eşitleme ile ayarlama
    # Görüntüyü numpy dizisine çevir
    img_np = np.array(img)

    # Adaptif eşitleme uygula
    img_np = exposure.equalize_adapthist(img_np, clip_limit=0.03)

    # Görüntüyü uint8 formatına geri çevir (kaydetme için)
    img_np = (img_np * 255).astype(np.uint8)

    # Görüntüyü tekrar Pillow formatına çevir
    img = Image.fromarray(img_np)

    # Ön işlenmiş görüntüyü kaydet
    img.save(output_path)

# Örnek kullanım
input_image_path = 'girdi_goruntu.png'  # Burayı giriş görüntü dosyanız ile değiştirin
output_image_path = 'islenmis_goruntu.png'  # Kaydedilecek dosya yolunu buraya yazın

preprocess_image(input_image_path, output_image_path)
```

Şekil 3.2 Görüntü işleme python fonksiyonu

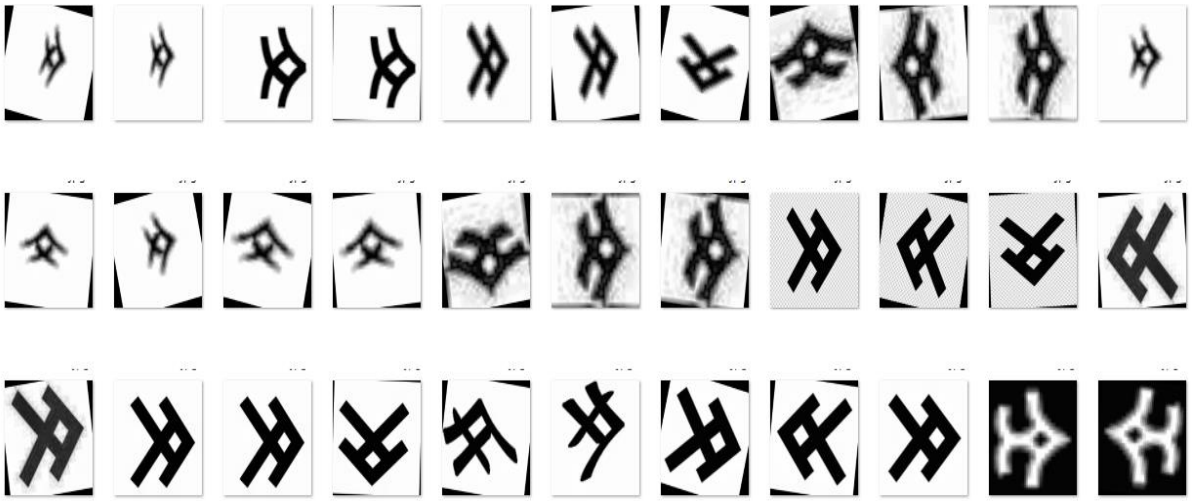
Ardından ikinci bir fonksiyon yazıldı bu fonksiyonda da veri artırma işlemi yapıldı. İşlemler yine bir python kodu yazılarak gerçekleştirildi. Python fonksiyon aşağıdaki şekil 3.3'te görseldeki gibidir. Bu işlemler;

- **90 Derece Döndürme:** Görüntü rastgele olarak 90°, 180°, ya da 270° döndürülür. Bu sayede modelin farklı yönlerdeki görüntüleri öğrenmesi sağlanır.
- **Rastgele Döndürme:** Görüntü rastgele -15° ile +15° arasında küçük açılarla döndürülür, bu da veri çeşitliliği sağlar ve modeli daha esnek hale getirir.
- **Gri Tonlama**: Görüntülerin %19'una rastgele gri tonlama uygulanır. Bu adım modelin renk bilgisi olmadan da şekil ve doku farkındalığı kazanmasına yardımcı olur.

```python
from PIL import Image, ImageOps
import random
import numpy as np

def augment_image(image_path, output_path):
    # Görüntüyü Pillow kullanarak yükle
    img = Image.open(image_path)

    # 1. 90° Döndürme (Saat yönünde, tersine, veya ters çevirme)
    rotate_choice = random.choice([90, 180, 270])  # Rastgele 90°, 180°, veya 270°
    img = img.rotate(rotate_choice, expand=True)

    # 2. Rastgele -15° ile +15° arasında döndürme
    random_angle = random.uniform(-15, 15)  # -15 ile +15 derece arasında rastgele açı
    img = img.rotate(random_angle, expand=True)

    # 3. %19 olasılıkla gri tonlamaya çevirme
    if random.random() < 0.19:  # %19 olasılık
        img = img.convert("L")  # Gri tonlama

    # Ön işlenmiş görüntüyü kaydet
    img.save(output_path)

# Örnek kullanım
input_image_path = 'girdi_goruntu.png'  # Burayı giriş görüntü dosyanız ile değiştirin
output_image_path = 'arttirilmis_goruntu.png'  # Kaydedilecek dosya yolunu buraya yazın

augment_image(input_image_path, output_image_path)
```

Şekil 3.3 Veri kümesi görsel artırma işlemi

Tüm bu işlemler sonucunda M harfine ait oluşan veri kümesine ait görsel aşağıdaki şekil 3.4'te görüldüğü gibidir.

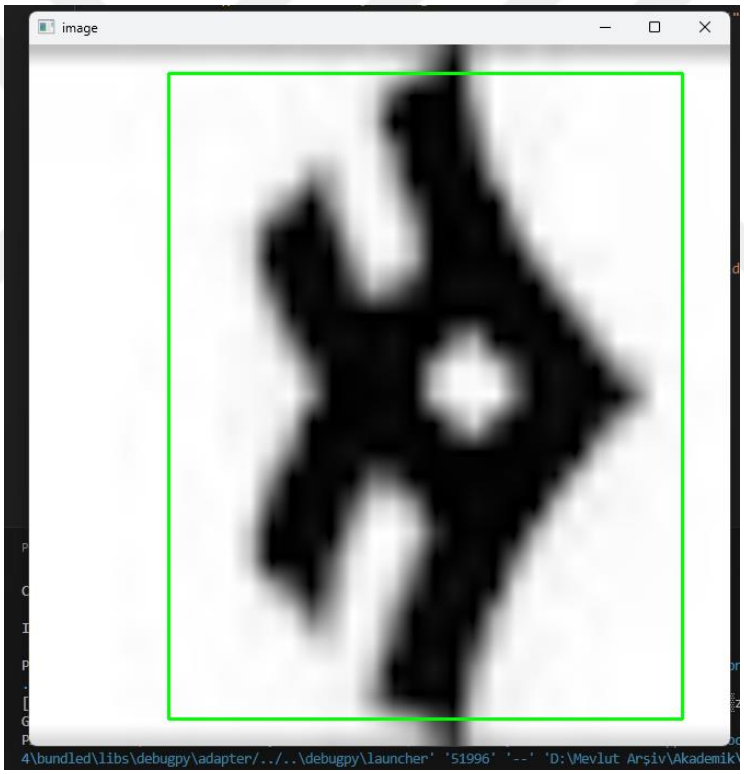


Şekil 3.4 Görüntü işlenmesi ve artırma işlemi yapıldıktan sonra veri setinin görseli

3.3. Veri Setlerinin Etiketlenmesi

Veri seti oluşturulduktan sonra bu veri setini etiketlemek bir hayli zaman aldı. Bu görselleri tek tek açıp üzerinden hangi harfler varsa onları dörtgen içerisinde belirtip bu dörtgenin pozisyonlarını arka planda bir metin dosyasında tutun bir Python kodu yazıldı. Bu kod sayesinde veri setindeki yer alan her görsele sırası ile tek tek girip ilk aşama harfi dörtgen içine alıp ardından o harfin etiket bilgisini giren bir uygulama geliştirilmiş oldu. Eğitim sırasında kullanılacak olan bu etiket bilgilerini ve çizilen dörtgenlerin bilgilerinin tutulduğu her görsele ayrı bir metin dosyası oluşturma işlemi de yapıldı.

Aşağıdaki şekil 3.5'te seçilmiş ve etiketlenmiş görsele ait örnek mevcuttur. Şekil 3.6'da de etiketlenen görselin harf ID si ve çizilen kutunun köşe koordinatları yer almaktadır.



Şekil 3.5 Etiketleme işlemine ait örnek görsel

```
12 0.55390625 0.45078125 0.5671875 0.746875
```

Şekil 3.6 Etiketmiş görsele ait etiket bilgilerinin yer aldığı metin belgesinin görseli

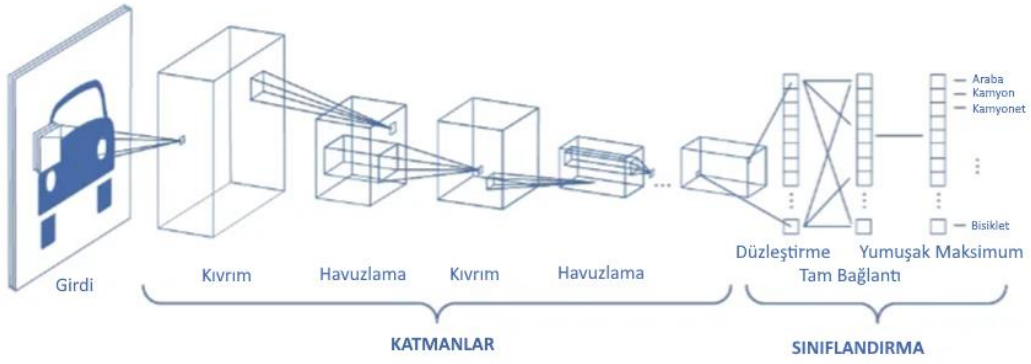
4. NESNE TESPİTİ ALGORİTMALARI

4.1. Convolutional Neural Networks (CNN)

CNN'lerin kökenleri, ilk olarak 1980'lerde Fukushima'nın "Neocognitron" modeli ile başlamıştır. Bu model, görsel verilerdeki özellikleri hiyerarşik olarak çıkarma fikrine dayalıydı. Fukushima'nın çalışmaları, CNN'lerin biyolojik görme sistemlerine benzer yapılarla görsel tanıma görevlerini başarmaya yönelik erken bir denemeydi [17].

Convolutional Neural Networks (CNN), görüntü işleme ve nesne tespiti gibi bilgisayarla görme uygulamalarında devrim yaratan derin öğrenme algoritmalarından biridir. CNN'ler, görüntüleri veya diğer çok boyutlu verileri otomatik olarak işleyerek özellikleri öğrenebilen yapılar sunar. Bu, geleneksel el ile çıkarılan özelliklere dayalı algoritmalardan farklıdır ve daha derin, daha karmaşık veri yapılarının anlaşılmasını sağlar [18,19].

CNN'ler, yapısal olarak biyolojik görme sistemiyle benzerlik gösterir. Temel yapı taşları arasında konvolüsyonel katmanlar, havuzlama (pooling) katmanları ve tam bağlı katmanlar bulunur. CNN'lerin temel işlevi, giriş verisindeki özellikleri farklı katmanlar aracılığıyla filtreler ve öğrenir [7,17,20].



Şekil 4.1 CNN çalışma mantığı [21]

Tablo 4.1 CNN modellerinin özellikleri

Model Adı	Yayın Yılı	Kaynak	Katman Sayısı	Parametre Sayısı	Giriş Boyutu	Avantajları	Dezavantajları
LeNet-5	1998	[18]	7	~60K	32x32	Basit ve hızlıdır, düşük donanım gerektirir.	Sadece küçük görüntüler için uygundur.
AlexNet	2012	[28]	8	~62M	224x224	Büyük veri setleri için uygundur, derinliği fazladır.	Yüksek donanım gereksinimi, aşırı uyum riski.
VGG-16	2014	[29]	16	~138M	224x224	Daha derin ve güçlü; farklı problemler için transfer öğrenmeye uygundur.	Çok fazla parametre, uzun eğitim süresi.
Inception v3	2015	[31]	48	~23M	299x299	Hesaplama verimliliği yüksek, çok yönlü yapısı var.	Yapısı karmaşıktır, geliştirilmesi zor.
ResNet-50	2015	[26]	50	~25.6M	224x224	Derin yapı, sıfırlanma (vanishing gradient) problemine dirençli.	Hala yüksek donanım gereksinimi vardır.
YOLOv8-m	2023	[40]	Değişken	~25M	Esnek (640x640)	Hızlı, orta ölçekli projeler için dengeli performans sağlar, gerçek zamanlı nesne tespiti.	Yüksek çözünürlükte hassasiyet kaybı yaşanabilir.

Konvolüsyonel Katman: Görüntüdeki yerel özellikleri tanımlamak için küçük filtreler (kernels) kullanılır. Filtreler, görüntü üzerinde kayarak (convolve) belirli desenleri algılar ve bu şekilde görüntü özelliklerini çıkarır. Bu filtreler eğitim sürecinde öğrenilir [22]. Konvolüsyonel katmanlar, Convolutional Neural Networks (CNN'ler) için en temel yapı taşlarından biridir ve özellikle veri üzerinde özellik çıkarımı yapmak için tasarlanmıştır. Bu katmanlarda filtreler (veya çekirdekler), giriş görüntüsüne kaydırma (stride) adı verilen bir hareketle uygulanır. Her bir filtre, belirli bir özellik tipini (örneğin,

kenarlar, dokular veya renk paternleri) algılamak üzere eğitilir ve görüntü üzerindeki yerel pikselleri işleyerek özellik haritaları (feature maps) oluşturur. Konvolüsyon işlemi sırasında, giriş verisinin mekânsal düzeni korunur ve bu düzen, çıkarılan özelliklerin özgünlüğünü sağlamada önemlidir. Bu mekânsal ilişkiler, modelin nesneleri tanımasına ve anlamlandırmasına olanak tanır. Örneğin, düşük seviyeli konvolüsyonel katmanlar kenar gibi basit özellikleri öğrenirken, daha derin katmanlar daha karmaşık özellikleri (örneğin, şekil veya dokular) tanır. CNN'ler, bu çok katmanlı yapısıyla düşük seviyeden yüksek seviyeye doğru özellikleri hiyerarşik olarak öğrenir ve böylece görüntü verilerini daha etkili bir biçimde analiz eder [18,23].

Aktivasyon Fonksiyonu: Konvolüsyonel katmandan sonra, genellikle doğrusal olmayan aktivasyon fonksiyonları kullanılarak modelin daha karmaşık yapıları öğrenebilmesi sağlanır [22]. CNN'lerde, her bir konvolüsyonel katmanın ardından kullanılan aktivasyon fonksiyonları, modelin doğrusal olmayan ilişkileri öğrenmesini sağlar. ReLU (Rectified Linear Unit), günümüzde CNN'lerde en yaygın kullanılan aktivasyon fonksiyonudur ve daha derin ağlarda gradient kaybolması sorununu azaltmaya yardımcı olur [24].

Havuzlama Katmanı: Bu katman, özellik haritalarını küçülterek işlem maliyetlerini azaltır ve modelin aşırı öğrenmesini engeller. Maksimum havuzlama (max pooling) gibi tekniklerle en önemli özellikler korunurken boyut küçültülür [22]. Havuzlama (pooling) katmanları, konvolüsyonel katmanların ardından yerleştirilen ve çıkarılan özellikleri özetleyerek veri boyutunu azaltan bir işlem uygular. Maksimum havuzlama (max pooling) en yaygın kullanılan tür olup, her bir özellik haritasında belirli bir bölgedeki en yüksek değeri seçer. Bu işlem, modelin deformasyonlara karşı daha dayanıklı hale gelmesini sağlar ve boyut azaltımı sayesinde hesaplama yükünü hafifletir. Alternatif olarak, ortalama havuzlama (average pooling), seçilen bölgenin ortalama değerini alarak özellikleri özetler. Bu işlemler, görüntüdeki yerel bilgiyi korurken, modelin daha soyut ve önemli özellikleri tanımasına yardımcı olur [25].

Tam Bağlı Katman (Fully Connected Layer): Havuzlama ve konvolüsyon katmanları sonucunda çıkarılan özellikler, sınıflandırma yapılmak üzere tam bağlı katmanda bir araya getirilir. Tam bağlı katmanlar, her bir nöronun bir önceki katmandaki tüm nöronlarla tam bağlantı kurduğu yapıdadır ve çıkarılan özellikleri sınıflandırma veya regresyon gibi son tahminlere dönüştürmek için kullanılır. Bu katmanlar sayesinde, model çıkarılan görsel özellikler arasındaki karmaşık ilişkileri öğrenir ve doğru sınıf etiketlerine ulaşır. Ek olarak, tam bağlı katmanlar, ağın özellik çıkarımı sürecinde kazandığı bilgiyi yoğunlaştırarak nihai sınıflandırma performansına katkıda bulunur [22,26].

CNN'ler, özellikle görüntü işleme uygulamalarında oldukça başarılıdır. Nesne tespiti görevinde, CNN'ler görüntülerdeki belirli nesneleri bulma ve sınıflandırmada kullanılır. Bu tür uygulamalar, YOLO (You Only Look Once) ve Faster R-CNN gibi modern nesne tespiti algoritmalarının temelini oluşturur [11]. CNN'ler, bir görüntünün içindeki farklı nesneleri tanıma yeteneğine sahiptir ve bu nesnelerin konumlarını öğrenir. Mask R-CNN modeli de Faster R-CNN modeli gibi nesne tespiti ve segmentasyon görevlerinde yüksek başarı oranları sunar [27]. CNN'lerin en büyük avantajlarından biri, görüntüdeki özellikleri otomatik olarak öğrenebilmesidir. Geleneksel algoritmalarda olduğu gibi el ile özellik çıkarımı yapmaya gerek kalmaz [28]. Bununla birlikte, CNN'ler büyük veri kümelerine ve yüksek hesaplama gücüne ihtiyaç duyar. Bu nedenle, özellikle nesne tespiti gibi görevlerde güçlü donanım gereksinimleri ortaya çıkar.

CNN'ler, büyük veri setleri üzerinde eğitildiklerinde çok güçlü özellik çıkarma yetenekleri geliştirirler. Bu özellik çıkarma yetenekleri, transfer öğrenimi yöntemiyle başka veri setlerine aktarılabilir. VGG, ResNet, Inception gibi önceden eğitilmiş CNN modelleri, transfer öğrenimi için yaygın olarak kullanılır [29]. Derin CNN modelleri, aşırı uyum (overfitting) sorunlarıyla karşılaşabilirler. Bu sorunu çözmek için Dropout ve Batch Normalization gibi teknikler geliştirilmiştir. Dropout, belirli katmanlardaki bazı bağlantıları rastgele devre dışı bırakarak aşırı uyumu azaltır. Toplu normalleşme (batch normalization) ise ağın öğrenme hızını artırmak ve eğitim sürecini stabil hale getirmek için katmanlar arasında verilerin normalize edilmesini sağlar [30,31].

4.2. You Only Look One (YOLO) Modeli

You Only Look Once (YOLO), nesne tespiti problemlerinde kullanılan ve nesneleri gerçek zamanlı olarak yüksek hızda tespit edebilen bir derin öğrenme algoritmasıdır. İlk olarak Joseph Redmon ve arkadaşları tarafından 2016 yılında sunulmuştur ve günümüze kadar çeşitli iyileştirmelerle (YOLOv2, YOLOv3, YOLOv4, YOLOv5, …YOLOv8) geliştirilmiştir. YOLO'nun en büyük avantajlarından biri, tek bir ileri geçiş (forward pass) ile bir görüntü üzerinde hem sınıflandırma hem de lokalizasyon yapabilmesidir, yani bir görüntü üzerinde sadece bir kez bakarak (look once) tüm nesneleri tespit eder [11].

YOLO'nun hız avantajı, büyük ölçüde tespit işlemini tek bir ileri geçişte tamamlamasından kaynaklanır. Bu durum, özellikle gerçek zamanlı nesne tespiti uygulamalarında büyük bir fark yaratır. YOLO, düşük gecikme süresi (latency) ile çalışabilir ve saniyede yüzlerce kareyi işleyebilir, bu da onu özellikle otomotiv, gözetim, insansız hava araçları ve mobil cihazlar gibi zamanın kritik olduğu alanlarda kullanışlı hale getirir. Bunu, YOLO'nun tek geçişli doğası, özellikle küçük nesnelerin veya yoğun nesnelerin bulunduğu durumlarda lokalizasyon doğruluğunu düşürebilir. YOLO, her bir hücreyi tek bir sınıf tahmini ile sınırlar ve bu da bir hücrede birden fazla nesne olduğunda zorluklar yaratabilir. Bu sorun, özellikle erken sürümlerde önemli bir zorluk olarak ortaya çıkmış ve daha sonra YOLOv3 ve YOLOv4 gibi geliştirilmiş sürümlerle iyileştirilmiştir [32].YOLO modeli, görüntüyü satır sayısı (Sa) x sütun sayısı (Su) hücrelere ayırarak, her hücrede olası nesneleri tespit etmeye çalışır. Her hücre, bir nesne kutucuğu (bounding box) ve o kutucuk içindeki nesnenin sınıfını tahmin eder.

Detectron2 adlı çalışmada, özellik çıkarımı ve sınıflandırmada etkili olan katman yapılarını ve geri yayılım süreçlerini geliştiren yöntemleri ele alarak, modellerin özellikle düşük çözünürlüklü ya da karmaşık arka plana sahip görüntülerdeki performansını artırmayı hedeflemektedir. Araştırma, Faster R-CNN ve YOLO gibi modellerin doğru sınıflandırma ve nesne kutucuğu (bounding box) doğruluğunun artırılmasında, ileri seviye konvolüsyonel ve optimizasyon tekniklerinin nasıl bir rol oynadığını vurgulamaktadır. Detectron2 önerdiği mimari iyileştirmeler, yalnızca hızlı sonuç üretme değil, aynı zamanda doğruluk oranlarının yüksek kalmasını da sağlamaktadır [33].

Matematiksel olarak, bir görüntü üzerinde p(C), B nesne kutucuğu (bounding box) ve her nesne kutucuğu (bounding box) için p sınıfına ait güven olasılığı hesaplanır. Her hücre için, aşağıdaki parametrelerden oluşan bir vektör üretilir:

$$v = [pc, bx, by, bw, bh, c1, c2 ,..., cn] \quad (4.1)$$

pc: Nesne varlık olasılığı

bx, by : Nesne kutusu (Bounding box) merkez koordinatları

bw, bh : Nesne kutusu (Bounding box) genişlik ve yükseklik bilgisi

ci: Nesnenin sınıfına ait olasılık (toplamda n sınıf)

Model, konvolüsyonel katmanlar aracılığıyla bu parametreleri öğrenir. Her nesne kutucuğu için hata, aşağıdaki denklem 4.2'deki gibi hesaplanır:

$$\text{Loss} = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(w_i - \hat{w}_i)^2 + (h_i - \hat{h}_i)^2 \right] \quad (4.2)$$

$1_{\{ij\}}^{\{obj\}}$: İlgili hücrede nesnenin olup olmadığını belirten gösterge fonksiyonu

λ_{coord} : Koordinat hatası için kullanılan bir ağırlık faktörü

x_i, y_i, w_i, h_i Tahmin edilen ve gerçek nesne kutucuğu (bounding box) koordinatları ve yükseklik genişlik bilgisi

Model, bu kayıp fonksiyonunu minimize ederek, nesneleri daha doğru tespit etmeyi öğrenir.

YOLO'nun diğer nesne tespiti yöntemlerinden farkı, görüntüyü önce "bölgelere" ayırmaması, bunun yerine doğrudan tüm görüntü üzerinden nesneleri tespit etmeye çalışmasıdır. Model, bir görüntüyü grid (hücreler) haline getirir ve her hücre, belirli nesneler için bir nesne kutucuğu (bounding box) önerisi sunar [32].

YOLO, nesne tespitinde tek bir CNN yapısı kullanarak aşağıdaki adımları izler:

1. Girdiyi bir CNN'den geçirerek görüntüdeki farklı özellikleri öğrenir.

2. Çıkış katmanı, her hücre için sınıf ve nesne kutucuğu bilgisi sağlar.

3. Non-Maximum Suppression (NMS) algoritması ile üst üste binen nesne kutuları arasından en uygun olanı seçilir.

4.2.1. YOLOv8

YOLOv8, seleflerinden öğrendiği dersleri ve daha modern derin öğrenme tekniklerini bir araya getirerek geliştirilmiştir. Bu sürümde, nesne tespiti, görüntü segmentasyonu ve sınıflandırma gibi çeşitli görevlerde daha yüksek performans sunmak için çeşitli iyileştirmeler yapılmıştır:

- Daha Derin ve Daha Geniş Mimari: YOLOv8, daha derin ve geniş bir sinir ağı mimarisine sahiptir, bu da modeli daha karmaşık ve detaylı özellikleri öğrenme konusunda daha yetenekli hale getirir. Bu mimari, veri miktarının arttığı durumlarda daha iyi genelleme yapabilir [34].

- Gelişmiş Özellik Haritaları: YOLOv8, çok ölçekli özellik haritalarını daha etkili kullanarak, farklı boyutlardaki nesnelerin tespitinde daha yüksek doğruluk sağlar [34].

- Improved NMS (Non-Maximum Suppression) Teknikleri: YOLOv8, non-maximum suppression (NMS) aşamasında iyileştirmeler yaparak çakışan sınır kutularını daha doğru bir şekilde filtreleyebilir. Bu, yanlış pozitiflerin azaltılmasına yardımcı olur ve genel doğruluğu artırır [34].

- Efficient Layer Normalization: YOLOv8, daha verimli bir şekilde normalize edilmiş katmanlar kullanarak eğitim sürecini hızlandırır ve daha stabil hale getirir. Bu, modelin daha hızlı yakınsamasına ve eğitimin daha az epoch gerektirmesine olanak tanır [34].

- Data Augmentation Teknikleri: YOLOv8'de, Mosaic Augmentation ve CutMix gibi veri artırma teknikleri daha etkili bir şekilde kullanılarak, modelin farklı senaryolarda daha sağlam hale gelmesi sağlanmıştır [34].

YOLOv8, eğitim sürecinde veri artırma ve düzenleme (regularization) tekniklerine büyük önem vermektedir. Bu sürümde kullanılan otomatik veri artırma (automated data augmentation) teknikleri, modelin genelleme yeteneğini artırır:

- CutMix ve Mosaic Augmentation: YOLOv8, bu veri artırma tekniklerini daha etkili bir şekilde kullanarak, eğitim veri setini genişletir ve çeşitlendirir. Bu, modelin farklı veri senaryolarında daha dayanıklı olmasını sağlar [35].

- Optimizer İyileştirmeleri: YOLOv8, eğitim sürecinde AdamW gibi modern optimizasyon tekniklerini kullanarak daha hızlı ve stabil bir eğitim süreci sunar [35].

4.3. Veri Artırma Teknikleri

Bölüm 3.2 Görüntü işlemesi konu başlığı altında kullanılan veri artırma tekniklerinden bahsedilmiştir.

5. MODEL EĞİTİMİ VE DEĞERLENDİRME

5.1. Eğitim Süreci ve Parametreleri

Bu çalışmada kullanılan nesne tespit modeli YOLOv8 (You Only Look Once versiyon 8), gerçek zamanlı nesne tespiti görevleri için geliştirilmiş bir derin öğrenme modelidir. YOLOv8, selefi olan YOLOv5 ve diğer YOLO varyantlarına göre birçok iyileştirme ve geliştirme sunarak daha yüksek doğruluk oranları ve hız sağlamaktadır. Özellikle düşük gecikme süreleri ve yüksek tespit doğruluğu gerektiren uygulamalarda, YOLO modelleri tercih edilmektedir. [22,36]

Model eğitiminde etiketlenen görseller kullanıldı. Bu görsellerin hepsi 640x640 piksel olacak şekilde ayarlandı. Resim boyutunun büyüklüğü hem modelin hesaplama yükünü hem de tahmin doğruluğunu etkiler. Daha büyük resimler, daha fazla ayrıntı sunar ve modelin daha hassas nesne tespiti yapmasını sağlar, ancak hesaplama maliyeti de artar. YOLO gibi modellerde resim boyutunun dikkatli seçilmesi, modelin hızlı çalışmasını sağlarken doğruluğu korumaya yardımcı olur [36].

Eğitim sürecindeki epochs sayısı, modelin veri kümesi üzerinde kaç tam geçiş yapacağını belirtir. Bu durumda, model 100 epoch olarak belirlenmiştir. Her epoch'ta, model veri kümesindeki tüm örnekleri bir kez görür ve ağırlıklarını günceller. Genellikle epoch sayısı, öğrenme kapasitesini ve eğitim süresini belirleyen en kritik parametrelerden biridir. Daha fazla epoch, modelin veriyi daha iyi öğrenmesine yardımcı olabilir, ancak aşırı öğrenme (overfitting) riskini de artırabilir [20]. Eğitim sırasında erken durdurma (Early Stopping) yöntemi uygulanmış ve modelin en iyi sonuçlarını 44. epoch'ta verdiği görülmüştür. Son 15 epoch boyunca iyileşme kaydedilmediği için eğitim bu noktada durdurulmuş, bu da öğrenme oranı düzenleyicisinin (learning rate scheduler) modelin optimum performansa ulaşmasına katkıda bulunduğunu göstermiştir. Bu durum, modelin aşırı öğrenme (overfitting) eğilimini önleyerek, genel performansı optimize eden bir şekilde sonuçlandırılmıştır.

Optimizasyon yöntemi olarak farklı tiplerde optimizasyon yöntemleri kullanılmıştır. Bunlar;

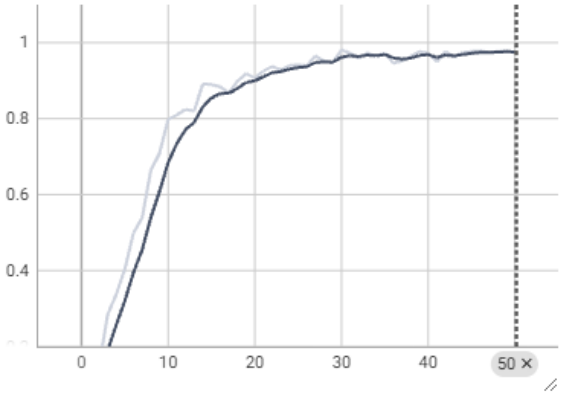
- AdamW
- SGD
- Adam
- NAdam
- RAdam
- Auto

5.2. Model Performans Kriterleri

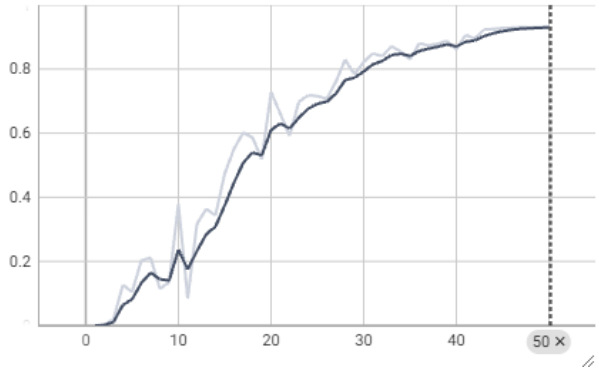
Modelin performans kriterlerini doğru şekilde karşılaştırabilmek için epoch sayısı, görüntü boyutlandırmaları sabit tutulmuş. Optimizasyon tipleri değiştirilerek eğitimler arasında karşılaştırma bu şekilde yapılmıştır.

Tablo 5.1 Optimizasyon algoritmalarının doğruluk ve hata değerleri tablosu

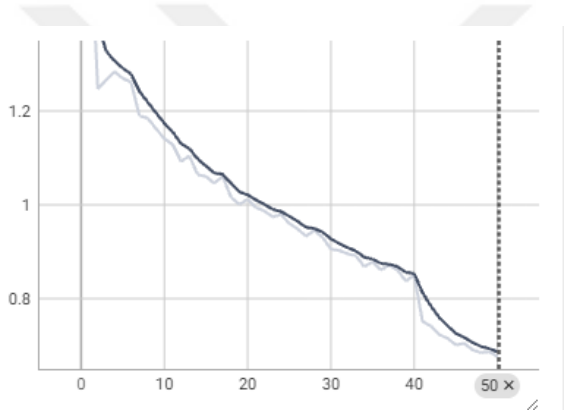
Optimizasyon Model Adı	Eğitim Süresi	Eğitim Hatası (Traning Loss)	Eğitim Doğrulaması (Traning Accuracy)	Doğruluk Hatası (Validation Loss)	Doğruluk Doğrulaması (Validation Accuracy)
AdamW	32 dk 57 sn	8.02%	93.31%	7.50%	79.60%
SGD	29 dk 28 sn	6.76%	97.43%	7.25%	82.31%
Adam	28 dk 19 sn	7.90%	93.20%	7.45%	78.70%
NAdam	31 dk 54 sn	7.90%	94.90%	7.04%	80.90%
Auto	32 dk 8 sn	6.80%	95.85%	7.40%	81.11%



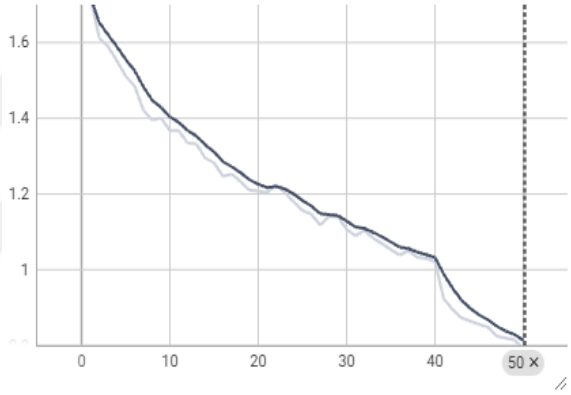
Şekil 5.1 SGD Optimizasyon algoritmasının doğruluk değişim grafiği



Şekil 5.2 AdamW optimizasyon algoritmasına göre doğruluk değişim grafiği



Şekil 5.3 SGD Optimizasyon algoritmasının hata değişim grafiği



Şekil 5.4 AdamW optimizasyon algoritmasının hata değişim grafiği

Yukarıdaki şekil 5.1 ve şekil 5.2'de grafiklerde AdamW ve SGD optimizasyonlarının epoch sayısına göre doğruluk değerleri; şekil 5.3 ve şekil 5.4'te hata oranlarının değişimleri karşılaştırılmıştır. Yatay sütunlar epoch sayısı dikey sütunlarda doğruluk değerinin yüzde karşılığıdır.

AdamW (Adaptive Moment Estimation with Weight Decay) optimizasyon algoritması, modern derin öğrenme modellerinin eğitiminde sıkça kullanılan bir teknik olarak öne çıkmaktadır. Özellikle derin sinir ağlarının eğitim sürecinde hızlandırma ve genelleme performansını artırma noktasında kritik bir rol oynar. AdamW, klasik Adam algoritmasının bir modifikasyonudur ve temel farkı, ağırlık çürümesi (weight decay) yöntemini daha doğru bir şekilde ele almasıdır. AdamW'nin başarısı, öğrenme hızını dinamik olarak ayarlayarak ve ağırlık çürümesini doğrudan optimize edilen

parametrelerin dışında tutarak daha iyi genelleme sağlamasında yatmaktadır [37].

AdamW'nin Temelleri: AdamW, Adam (Adaptive Moment Estimation) algoritmasının bir uzantısıdır. Adam, Stokastik Gradyan İnişi (SGD) ve Momentum yöntemlerinin bir birleşimi olarak ortaya çıkmıştır [37]. Adam algoritmasında, gradyanların ilk ve ikinci moment tahminleri kullanılarak parametre güncellemeleri yapılır. Momentum, gradyanlarda önceki iterasyonlardan gelen bilgilerden yararlanarak öğrenme sürecini hızlandırırken, Adam bunu adaptif öğrenme oranları ile birleştirir [37]. Ancak klasik Adam algoritması, genellikle aşırı uyuma (overfitting) yol açar, çünkü ağırlık çürümesi (weight decay) hatalı uygulanır ve bu da genelleme performansını olumsuz etkiler [38]. AdamW, klasik Adam'dan farklı olarak ağırlık çürümesi (weight decay) (L2 regularization) doğrudan gradyan güncellemelerine dahil etmez. Bu sayede ağırlık çürümesi (weight decay) terimi, yalnızca modelin parametrelerinin ağırlıklarını güncellerken devreye girer. Böylece AdamW, regularization'ı daha etkili bir şekilde uygular ve aşırı öğrenmeyi önler [38].

Matematiksel Formülasyon: AdamW algoritmasının güncelleme denklemlerini aşağıdaki denklem 5.1, 5.2 ve 5.3'teki gibi ifade edebiliriz:

Gradyan Hesaplama:

$$g_t = \nabla_{\{\theta\}L(\theta_t)} \tag{5.1}$$

Burada g_t, t'inci iterasyonda hesaplanan gradyanı ifade eder.

$L(\theta_t)$ kayıp fonksiyonudur ve model parametreleri (θ_t) ile tanımlanır [37].

Gradyanların İlk Momentinin (Mean) Hesaplanması:

$$m_t = \beta_1 m_{\{t-1\}} + (1 - \beta_1) g_t \tag{5.2}$$

Burada m_t, gradyanın hareketli ortalamasını temsil eder. β_1 (genellikle 0.9) moment parametresidir [37].

Gradyanların İkinci Momentinin (Variance) Hesaplanması:

$$v_t = \beta_2 v_{\{t-1\}} + (1 - \beta_2) g_t^2 \quad (5.3)$$

v_t , gradyanın karelerinin hareketli ortalamasıdır. β_2 (genellikle 0.999) ikinci moment parametresidir [37].

Yanlılık Düzeltmesi (Bias Correction): İlk ve ikinci momentler başlangıçta sıfırdan başlar, bu nedenle iterasyonların başında sapmalı olabilirler. Bu yanlılığı gidermek için düzeltme uygulanır [37]:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \ \hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad (5.4)$$

Ağırlık Güncelleme (Parametre Güncelleme): AdamW algoritmasının en önemli kısmı bu adımdır. Klasik Adam'dan farklı olarak, ağırlık çürümesi (weight decay) doğrudan parametre güncellemesinden ayrı bir terim olarak ele alınır:

$$\theta_{t+1} = \theta_t - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \right) - \eta \lambda \theta_t \quad (5.5)$$

Burada:

η, öğrenme oranını temsil eder.

λ, ağırlık çürümesi (weight decay) terimidir.

$\in$, sayısal kararlılığı artırmak için kullanılır [37].

Bu denklem 5.5'teki formülasyon, Adam algoritmasının adaptif öğrenme oranlarını kullanırken ağırlık çürümesinin (weight decay) etkisini de optimal bir şekilde ele alır. Klasik Adam'da ağırlık çürümesi (weight decay) yanlışlıkla adaptif öğrenme oranlarının bir parçası olarak hesaplanır ve bu da modelin aşırı öğrenmesine yol açabilir. AdamW bu sorunu ortadan kaldırarak, regularization'ın etkisini doğrudan parametrelerin optimizasyon sürecine dahil eder [37].

Ağırlık Çürümesi (Weight Decay) ve AdamW'nin Etkileri;

Ağırlık çürümesi, modelin parametrelerini sınırlayarak genelleme kabiliyetini artırır. Geleneksel L2 regularization ile benzer şekilde çalışsada, AdamW'de ağırlık çürümesi, Adam algoritmasının adaptif öğrenme oranı bileşenlerine dahil edilmeden uygulanır. Bu yaklaşım, modelin ağırlıklarını daha doğrudan kontrol ederek aşırı uyumun önüne geçer ve modelin aşırı öğrenmeden (overfitting) kaçınmasını sağlar [37]. Ağırlık çürümesi'in AdamW'de ayrı bir terim olarak ele alınması, model parametrelerinin L2 normu aracılığıyla güncellenmesini sağlar. Bu, parametrelerin daha küçük değerlerde kalmasını sağlayarak aşırı öğrenmeyi ve gereksiz karmaşıklığı önler. Sonuç olarak, AdamW, genelleme yeteneği açısından Adam'a kıyasla daha başarılıdır ve derin öğrenme modellerinde daha geniş bir uygulama alanı bulur [38].

5.2.1. AdamW optimizasyonun avantajları

AdamW'nin klasik Adam'a göre getirdiği avantajlar aşağıdaki gibi özetlenebilir:

- Daha İyi Genelleme: Ağırlık çürümesinin adaptif öğrenme oranları dışında tutulması, modelin aşırı öğrenme yapmasını engeller ve daha iyi genelleme performansı sağlar [38].

- Ağır Ceza Uygulaması: L2 regularization daha doğru bir şekilde uygulanarak modelin karmaşıklığı azaltılır [38].

- Dinamik Öğrenme Oranı: Adam algoritmasının adaptif öğrenme oranı bileşenleri

AdamW'de de korunur, bu da her parametrenin farklı bir hızla güncellenmesini sağlar [38].

AdamW, derin öğrenme modellerinin eğitiminde sıklıkla tercih edilen ve klasik optimizasyon yöntemlerine göre önemli avantajlar sunan bir algoritmadır. Adaptif öğrenme oranları ve ağırlık çürümesinin ayrı ayrı ele alınması sayesinde hem hızlı hem de doğru bir öğrenme süreci sağlar. Aşırı öğrenme riskini minimize ederek genelleme yeteneğini artırır ve bu da özellikle büyük veri kümeleri ve derin modellerde önemli bir fayda sunar [38,39].

AdamW'nin matematiksel temelleri ve uygulama süreçleri, derin öğrenme alanında kullanılan diğer optimizasyon algoritmalarına kıyasla daha ileri düzey bir performans sağlamakta olup, modern derin öğrenme projelerinde yaygın olarak kullanılmaktadır [38].

5.3. Yanıltıcı Pozitif ve Negatif Tespitler

Yanıltıcı pozitif ve negatif tespitler, nesne algılama modellerinin güvenilirliğini ve etkinliğini değerlendirmede kritik bir rol oynar. YOLOv8 modelini Göktürk metinlerini okumak amacıyla eğitirken, yanıltıcı pozitif (false positive) ve yanıltıcı negatif (false negative) tespitlerin analiz edilmesi önemlidir. Bu tespitler, modelin performansını iyileştirmek ve hata oranlarını minimize etmek için önemli veriler sunar.

5.3.1. Yanıltıcı pozitif tespitler (False positives)

Yanıltıcı pozitif tespit, modelin aslında mevcut olmayan bir nesneyi (bu durumda Göktürk harfini) algıladığını düşündüğü durumları ifade eder. Göktürk alfabesinin kendine özgü sembolleri ve yazım tarzı, özellikle modelin belirli şekilleri ya da sembol benzeri yapıları yanlışlıkla harf olarak algılamasına yol açabilir. Yanıltıcı pozitif tespitlerin temel nedenleri şunlardır:

Veri Setindeki Gürültü (Noise in Dataset): Eğitilen veri setinde, özellikle gerçek dünya

görüntülerinde veya taş yazıtların bozulmuş bölgelerinde, harflere benzeyen doğal çizgiler ya da semboller yer alabilir. Bu tür hatalı pozitif tespitler, modelin hatalı bir biçimde bu bölgeleri Göktürk harfleri olarak sınıflandırmasına neden olabilir.

Modelin Genel Yargısı (Model Bias): YOLOv8'in eğitiminde kullanılan optimizasyon fonksiyonu ve parametre ayarları, modelin belirli şekilleri aşırı genelleştirmesine sebep olabilir. Örneğin, AdamW optimizasyonu kullanılmış ve modelin bu genelleme eğilimi, küçük ya da düzensiz desenleri de Göktürk harflerine benzediği için yanlış sınıflandırmasına yol açabilir.

Harfler Arasındaki Görsel Benzerlik (Visual Similarity): Göktürk yazısındaki bazı harfler, diğer sembollerle görsel olarak benzerdir. Bu, modelin eğitim sürecinde bu harfleri daha fazla yanlış pozitif olarak tanımlamasına neden olur.

5.3.2. Yanıltıcı negatif tespitler (False negatives)

Yanıltıcı negatif tespit ise, modelin aslında var olan bir nesneyi (Göktürk harfini) algılayamamasıdır. Bu durum, özellikle metinlerin tam olarak okunmasını engelleyebilir ve modelin doğruluğunu düşürebilir. Yanıltıcı negatiflerin temel sebepleri ise şu şekildedir:

Yetersiz Veri Temsilciliği (Insufficient Data Representation): Eğitilen modelin, veri setindeki bazı harf türlerini ya da yazım stillerini yeterince öğrenememesi yanıltıcı negatif tespitlere neden olabilir. Özellikle belirli harflerin çeşitli varyasyonlarının eğitim verisinde yeterince temsil edilmemesi, modelin bu harfleri algılamada başarısız olmasına yol açabilir.

Düşük Çözünürlük (Low Resolution): YOLOv8 modelinde kullanılan görüntü boyutu (imgsz=640) sabitlenmiştir, ancak bu boyut bazı durumlarda harflerin net şekilde algılanmasını zorlaştırabilir. Harflerin çözünürlüğü düşük olduğunda model, bu harfleri algılamayabilir ve bu da yanıltıcı negatiflere neden olur.

Aşırı Genelleme (Overgeneralization): Modelin aşırı genelleme yapması da yanıltıcı negatiflere neden olabilir. Eğitim sırasında belirli harflerin veya sembollerin varyasyonlarını yeterince öğrenemeyen model, bu tür durumlarda var olan harfi algılayamaz. Örneğin, SGD gibi optimizasyon algoritmaları, bazen modelin bazı örüntüleri öğrenmemesine yol açabilir, çünkü öğrenme oranları belirli durumlarda aşırı düzeltme yapabilir.

5.3.3. Sonuç ve iyileştirme yöntemleri

Yanıltıcı pozitif ve negatif tespitlerin azaltılması için modelin eğitimi sırasında veri setinin kalitesi, çeşitliliği ve boyutları büyük önem taşır. Yanıltıcı pozitif tespitleri azaltmak için, veri setine daha fazla negatif örnek (harf olmayan bölge) eklemek ve modelin gürültüye dayanıklılığını artırmak faydalı olacaktır. Yanıltıcı negatif tespitleri azaltmak için ise, modelin daha çeşitli veri setleri ile eğitilmesi, özellikle nadir görülen harflerin ve varyasyonlarının eklenmesi önemlidir. Ayrıca, daha gelişmiş optimizasyon algoritmaları kullanarak modelin öğrenme kapasitesini artırmak ve aşırı genellemeyi önlemek de katkı sağlayabilir. Bu projede diğer optimizasyon algoritmalarında da model eğitilerek aradaki farkların gözlemlenmesi sağlanmıştır.

5.4. Sonuçların Yorumlanması

```
Validating runs/detect/train/weights/best.pt...
Ultralytics 8.3.5 🚀 Python-3.10.12 torch-2.4.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 236 layers, 23,228,308 parameters, 0 gradients, 67.6 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:02<00:00,  1.19s/it]
                   all         36        181      0.812      0.914      0.936      0.805
                   A-E          8         10      0.951          1      0.995      0.839
                     B          3          3      0.804          1      0.995      0.913
                    B1          1          1        0.8          1      0.995      0.895
                BOSLUK         15         15          1      0.761      0.995      0.449
                     C          2          2      0.805          1      0.995      0.847
                     D          1          1          1          0      0.332      0.298
                    D1          2          2      0.597          1      0.828      0.745
                    G1          3          3      0.894          1      0.995      0.908
                   I-I          7          7      0.747          1      0.995      0.874
                    IK          2          2      0.848          1      0.995      0.895
                     K          4          4      0.863          1      0.995      0.946
                    K1          1          1      0.438          1      0.995      0.895
                     L          3          3      0.486      0.667      0.806      0.672
                    L1          6          6      0.852      0.967      0.901      0.753
                     M         25         32      0.963      0.969      0.973      0.831
                     N          1          1      0.684          1      0.995      0.796
                    N-          2          2       0.67        0.5      0.638      0.613
                    N1          2          2      0.875          1      0.995      0.921
                    ND          2          2      0.846          1      0.995      0.824
                   O-U          6          6      0.933          1      0.995      0.929
                    OK          4          4      0.913          1      0.995      0.853
                OK-UK          2          2       0.85          1      0.995      0.847
                     P          4          4      0.823          1      0.995      0.838
                     R         23         26          1      0.912      0.994      0.877
                    R1          1          1      0.228          1      0.995      0.895
                     S          5          6          1      0.477      0.863      0.647
                    S1         13         16      0.798          1      0.847      0.577
                     T          4          4      0.734          1      0.895      0.811
                    T1          6          6      0.935          1      0.995      0.883
                   U-O          3          3      0.955          1      0.995      0.895
                     Y          2          2      0.852          1      0.995      0.895
                     Z          2          2      0.837          1      0.995      0.895
Speed: 0.2ms preprocess, 54.3ms inference, 0.0ms loss, 1.7ms postprocess per image
Results saved to runs/detect/train
```

Şekil 5.5 Model Eğitim Sonuçları

5.4.1. Model performansı

Model performans değerlendirmesi aşamasında, YOLOv8'in "medium" sürümü tercih edildi. Eğitimler Google Colab ortamında yapıldı. Kullanılan GPU modeli T4 GPU'dur. GPU bellekte yaklaşık 8 gb yer işgal ederek eğitim işlemlerini yapmıştır. YOLOv8'in geliştirilmiş mimarisi, modelin düşük gecikme süresiyle çalışmasını sağlayarak, gerçek zamanlı uygulamalarda önemli bir avantaj sunar. Özellikle konvolüsyonel katmanlar ve optimize edilmiş aktarma öğrenimi teknikleri, YOLOv8'in karmaşık görüntülerde bile nesne tespiti görevlerinde yüksek doğruluğa ulaşmasına katkıda bulunur [40]. Ayrıca, modelin veri yoğun işlemlere adaptasyonu ve daha küçük bellek ayak izi, büyük veri setleriyle verimli çalışmasını sağlar [41]. Bu bağlamda, YOLOv8'in m versiyonu hem hız hem de doğruluk kriterleri göz önünde bulundurulduğunda, veri zenginleştirme ve hiperparametre optimizasyonu gibi yöntemlerle daha da geliştirilmiştir. Modelin düşük bellek kullanımı, hızlı işlem kapasitesi ve derin konvolüsyonel yapısı, özellikle büyük ölçekli veri setleri üzerinde çalışırken avantaj sağlar [42].

Eğitim sonuçlarına göre, modelin genel başarısı oldukça yüksek olup, özellikle mAP (Mean Average Precision) metrikleri, modelin hedef tespit yeteneği hakkında kapsamlı bilgi sunmaktadır. mAP@50 değeri %90 olarak elde edilmiştir. Bu değer, modelin belirlenen eşik değerinde (IoU@50) yüksek doğrulukla nesneleri tanımlayabildiğini göstermektedir. mAP@50-95 değeri %78.8 ise modelin farklı zorluk seviyelerinde (farklı IoU eşiklerinde) tespit yeteneğinin nispeten düşük olduğunu işaret etmektedir. Bu fark, daha karmaşık ya da üst üste binen nesnelerde modelin performansında bir miktar düşüş olduğunu göstermektedir, ancak genel anlamda bu sonuçlar başarılı bir tespit performansına işaret etmektedir.

5.4.2. Eğitim süreci

Eğitim sırasında erken durdurma (Early Stopping) yöntemi uygulanmış ve modelin en iyi sonuçlarını 44. epoch'ta verdiği görülmüştür. Son 15 epoch boyunca iyileşme kaydedilmediği için eğitim bu noktada durdurulmuş, bu da öğrenme oranı düzenleyicisinin (learning rate scheduler) modelin optimum performansa ulaşmasına katkıda bulunduğunu göstermektedir. Bu durum, modelin aşırı öğrenme (overfitting) eğilimini önleyerek, genel performansı optimize eden bir süreç olarak değerlendirilebilir.

GPU bellek kullanımı (8.6 GB) ve eğitim süresi (0.511 saat), modelin hesaplama kaynaklarını verimli bir şekilde kullandığını ve pratik uygulamalar için yeterince hızlı olduğunu göstermektedir. Özellikle inference süreleri resim başına 54.3 ms ile oldukça düşük seviyede kalmış ve bu durum gerçek zamanlı uygulamalar için elverişli bir yapı sunmaktadır.



Şekil 5.6 Orjinal Orhun yazıtından alınan bir metnin analiz edilmesi

Şekil 5.6 da Göktürkçe bir metne ait metnin eğitimi yapılmış modelimiz ile harf tespitleri yapılmıştır.



Şekil 5.7 ŞER kelimesinin analiz sonucu

Şekil 5.7 de Göktürkçe olarak ŞER yazmaktadır. Bizim eğittiğimiz modelin sonucunda da ŞER şeklinde harfleri çözümlemiştir.

5.4.3. Sınıf bazında performans

Modelin sınıf bazındaki doğruluk (precision) ve geri çağırma (recall) değerleri incelendiğinde, sınıflar arasında belirgin bir performans çeşitliliği görülmektedir.

Precision değeri genel olarak %90 seviyesinde, bu da modelin tespit ettiği nesneleri büyük oranda doğru sınıflandırdığını göstermektedir. Ancak, geri çağırma (recall) değeri %78 olarak elde edilmiş, bu da modelin bazı nesneleri atladığını ya da eksik tespit ettiğini işaret etmektedir. A-E sınıfı (eğitimde en fazla örneği bulunan sınıf), %81.2 doğruluk (precision) ve %91.4 geri çağırma (recall) ile başarılı bir performans sergilemiştir. Benzer şekilde, M sınıfında da %93.6 doğruluk (precision) ve %83.1 geri çağırma (recall) değerleriyle yüksek bir doğruluk elde edilmiştir.

Bununla birlikte, BOSLUK (boşluk) sınıfında %99.5 doğruluk (precision) elde edilmesine rağmen, %44.9 geri çağırma (recall) oranı modelin bu sınıfı tespit etmede zorluk çektiğini göstermektedir. Bu durum, boşlukların model tarafından genellikle doğru tanımlandığını ancak sıklıkla kaçırıldığını işaret eder.

R1 sınıfında ise, %99.5 doğruluk (precision) karşın %22.8 geri çağırma (recall) oranı gözlemlenmiştir. Bu, modelin bu karakteri öğrenmede yetersiz kaldığını göstermekte olup, bu sınıf için ek eğitim verisi veya daha fazla optimizasyon ihtiyacını işaret eder. Bu bağlamda, örnek sayısı düşük olan sınıflar üzerinde daha fazla veri artırımı (data

augmentation) uygulanması ve özellikle nesne sınıflarının dengesiz olduğu veri setlerinde, sınıf bazlı hata analizi yapmak, modelin sınıflar arası performansını iyileştirebilir [43]. Ayrıca, sınıflar arası doğruluk ve geri çağırma değerlerinin dengelenmesi için focal loss gibi kayıp fonksiyonları kullanılabilir, bu da modelin nadir görülen sınıflara duyarlılığını artırabilir [44].



Şekil 5.8 M harfinin analiz sonucu

Şekil 5.8 de Göktürkçe M harfi modele test olarak verildi. Model %85 doğruluk oranı ile bu görselde yer alan şekli M harfi olarak çözümlemiştir.

5.4.4. Geliştirme alanları

YOLOv8 modellerinde performansı iyileştirmek adına çeşitli geliştirme alanları mevcuttur. YOLOv8, önceki versiyonlarından farklı olarak PANet (Path Aggregation Network) altyapısını geliştirmiş ve bu sayede farklı ölçeklerdeki nesneleri daha iyi tespit edebilen bir yapı sunmuştur. Bu geliştirme, özellikle küçük ve sıkışık nesnelerin yoğun olduğu sahnelerde yüksek doğrulukla çalışmaktadır. Modelin Mean Average Precision (mAP) değeri ve işlem süresi de gözle görülür biçimde iyileştirilmiştir. Örneğin, YOLOv5 ve YOLOv8 arasındaki mAP değerlerinde belirgin bir artış sağlanmış ve YOLOv8 daha hızlı bir çıkarım süresi sunmuştur, bu da özellikle gerçek zamanlı uygulamalar için büyük bir avantajdır [45,46].

Sonuçlar genel olarak başarılı olmakla birlikte, modelin bazı sınıflarda daha düşük geri çağırma oranlarına sahip olması dikkat çekicidir. Özellikle BOSLUK, R1 ve S1 sınıflarında modelin bu karakterleri tespit etmede zorlandığı görülmektedir. Bu durumu

iyileştirmek için sınıf dengesizliğinin giderilmesi, veri artırma stratejilerinin gözden geçirilmesi ve daha fazla eğitim verisi ile modelin desteklenmesi önerilmektedir. Ayrıca, erken durdurma yönteminin uygulanması sonucu modelin eğitim süreci erken tamamlanmış olabilir; sabır süresinin artırılmasıyla öğrenme oranı daha da düşürülerek, uzun vadede performans artışı sağlanabilir.

Bu çalışma kapsamında YOLOv8 modeli, Göktürk metinlerinin tespiti için başarılı bir performans sergilemiştir. Yüksek mAP@50 değeri, modelin genel doğruluğunu göstermekte, ancak bazı sınıflar için düşük geri çağırma oranları modelin bu karakterleri tanıma kabiliyetinde eksiklikler olduğunu ortaya koymaktadır. Daha fazla veri ile modelin geri çağırma performansı iyileştirilebilir ve sınıf dengesi üzerinde yapılacak optimizasyonlar ile tespit kabiliyeti artırılabilir.

6. BULGULAR VE TARTIŞMA

6.1. Göktürkçe Harf ve Kelime Tespiti Sonuçları

Bu çalışmada, YOLOv8 modelinin Göktürkçe harf ve kelime tespitine uyarlanması, nesne tespiti ve görüntü sınıflandırmada etkili bir yöntem olan "You Only Look Once" (YOLO) algoritmalarının başarısının, antik yazılar ve semboller gibi alanlara uygulanabilirliğini gösteren bir örnektir. YOLOv8'in konvolüsyonel ve katmanlı yapısı, nesne tespitindeki başarısının temelini oluşturan özellik çıkartma süreci ile karakterleri yüksek doğrulukla tanıyabilir [38]. YOLOv8 modeli kullanılarak Göktürkçe harf ve kelime tespiti gerçekleştirilmiştir. YOLOv8'in "medium" versiyonu üzerinde yapılan eğitimler, Göktürk alfabesi ve bazı kelimelerin tespiti için 100 epoch boyunca eğitim verilmiştir. Modelin eğitimi sırasında AdamW optimizer kullanılarak, veri augmentasyonu, sabırsızlık parametresi (patience) ve öğrenme oranı (learning rate) gibi çeşitli hiperparametreler optimize edilmiştir. Sonuç olarak model, eğitim verisinde elde edilen doğruluk ve hatırlama oranlarıyla birlikte Göktürkçe harf tespiti üzerine önemli bulgular sunmaktadır.

6.1.1. Model eğitimi ve hyperparametre ayarları

Model eğitimi sürecinde kullanılan YOLOv8m (medium) modeli, 100 epoch süresince eğitilmiş ancak erken durdurma (Early Stopping) mekanizması sayesinde eğitim, 59. epoch'ta durdurulmuştur. Bu durdurma, son 15 epoch boyunca model performansında iyileşme olmaması durumunda, aşırı öğrenmeyi (overfitting) önlemek amacıyla gerçekleştirilmiştir. Bu noktada, en iyi model performansının 44. epoch'ta elde edildiği görülmüş ve bu model kaydedilmiştir.

AdamW optimizer kullanımı, daha iyi genelleme ve stabil öğrenme sağlaması açısından özellikle tercih edilmiştir. Ayrıca modelde veri artırma aktif hale getirilmiş, bu sayede veri seti çeşitlendirilerek modelin öğrenme kapasitesinin artırılması hedeflendi. Eğitim esnasında kullanılan batch size değeri 16 olarak ayarlanmış, bu da GPU bellek limitleriyle optimize edildi. Başlangıç öğrenme oranı (lr0) olarak 0.001 kullanıldı ve modelde aşırı öğrenmeyi önlemek için düşük bir ağırlık çürümesi değeri 0.0005 tercih edildi.

6.1.2. Eğitim performansı ve modelin genel başarısı

Modelin eğitimi sonucunda elde edilen mAP@50 ve mAP@50-95 değerleri, modelin performansını değerlendirmek için kullanılan başlıca metriklerdir. mAP, modelin nesneleri ne kadar doğru tespit edebildiğini ve sınır kutularını (bounding box) ne kadar doğru yerleştirdiğini gösterir.

mAP@50 değeri %90 olarak gözlemlendi. Bu sonuç, modelin 0.50 IoU (Intersection over Union) eşik değerinde nesneleri oldukça doğru tespit ettiğini göstermektedir. Özellikle bu metrik, modelin tespit ettiği harflerin büyük bir çoğunluğunu doğru bir şekilde sınıflandırdığını ve tespit başarısının oldukça yüksek olduğunu ortaya koymaktadır.

Ancak mAP@50-95 değeri %78.8 olarak daha düşük bir performansa işaret etmektedir. Bu metrik, modelin daha karmaşık veya zorlayıcı nesne tespitlerinde performans düşüklüğüne sahip olduğunu göstermektedir. Özellikle üst üste binen veya daha belirsiz harf ve kelimelerin tespitinde modelin zorluk yaşadığı anlaşılmaktadır.

Veri artırımı teknikleri, eğitim verisinin çeşitlenmesi yoluyla modelin daha genellenebilir hale gelmesini sağlarken, “patience” parametresi de modelin aşırı öğrenmesini engelleyerek genel performansın artmasına katkıda bulunur [47].

6.1.3. Harf ve kelime sınıflarında model performansı

Modelin Göktürk harflerini ve bazı kelimeleri sınıflandırma başarısı, sınıf bazında doğruluk (precision) ve hatırlama (recall) değerleri ile detaylandırılmıştır. Eğitim sürecinde modelin karşılaştığı her harf ve kelime sınıfı için ayrı performans değerleri elde edilmiştir. A-E harfleri sınıfı için modelin doğruluk (precision) değeri %81.2, hatırlama (recall) değeri ise %91.4 olarak ölçülmüştür. Bu sonuçlar, A-E harflerinin model tarafından büyük oranda doğru tespit edildiğini ve modelin bu harfleri atlamadan başarılı bir şekilde tanımlayabildiğini göstermektedir.

B harfi için modelin hatırlama oranı %80.4 iken, doğruluk oranı %100'e yakın bir değerdedir. Bu sonuç, modelin B harfini tespit etmede başarılı olduğunu ancak bazı durumlarda bu harfi kaçırma ihtimali olduğunu ortaya koymaktadır.

BOSLUK sınıfı, yani Göktürkçe kelimeler arasında yer alan boşluk karakteri için modelin doğruluk oranı %99.5 iken, hatırlama oranı yalnızca %44.9 olarak ölçülmüştür. Bu fark, modelin boşluk karakterlerini doğru şekilde tespit ettiğini, ancak bu sınıfı sık sık kaçırdığını işaret etmektedir. Bu durumun nedeni, boşluk karakterlerinin diğer harflerden daha az belirgin olmasıdır.

M harfi için modelin doğruluk oranı %93.6, hatırlama oranı ise %83.1 olduğu gözlemlendi. Model bu harfi genel olarak iyi bir şekilde öğrenmiş olup, küçük bir eksik tespit oranı mevcuttur.

R1, yani kutuplu ünsüzlerden kalın R harfi için kullan sınıf adlandırmasıdır. R1 için hatırlama oranı oldukça düşüktür (%22.8), ancak doğruluk oranı yine %99.5 seviyesindedir. Bu sonuç, modelin R1 sınıfını tanımlarken büyük oranda doğru olduğunu, ancak bu harfi tespit etmede önemli eksiklikler yaşadığını göstermektedir. Eğitim verilerindeki dengesizlik ya da harflerin görsel karmaşıklığı bu sonuca yol açmıştır.

6.1.4. Modelin genel güçlü yönleri ve zayıflıkları

Bu eğitimde model, genel olarak Göktürkçe harf ve kelime tespitinde başarılı bir performans sergilemiştir. Doğruluk (precision) değerlerinin büyük çoğunluğu %90'ın üzerinde olup, modelin doğru tespitlerde bulunduğu söylenebilir. Hatırlama (recall) oranları ise sınıf bazında değişiklik göstermektedir ve bazı harflerde düşük performans gözlenmiştir. Özellikle boşluk ve R1 gibi sınıflar, modelin hatırlama kapasitesinde zayıflıklar göstermiştir.

Modelin eksikliklerini gidermek için şu önerilerde bulunulabilir:

Veri dengesizliği: Eğitim verisinde bazı harflerin yeterince temsil edilmemesi, modelin bu harfleri öğrenmesini zorlaştırmıştır. Veriler dengelenerek eğitim yapılması, modelin performansını artırabilir.

Veri Artırma: Daha geniş kapsamlı veri artırma uygulanarak, modelin özellikle düşük performans gösterdiği sınıflar üzerindeki tespit yeteneği artırılabilir.

Sınıf sayısının genişletilmesi: Modelin öğrenme kapasitesini daha da geliştirmek için harfler ve kelimeler arasındaki ayrımlar detaylandırılabilir, böylece model daha ince sınıf farkları üzerinde eğitim alabilir.

Sonuç olarak, YOLOv8 modeli Göktürk harf ve kelime tespiti konusunda başarılı sonuçlar vermiştir. Modelin genel doğruluğu oldukça yüksek olup, belirli sınıflarda (özellikle A-E, M gibi) etkileyici bir tespit kabiliyeti sergilemiştir. Bununla birlikte, bazı sınıflarda hatırlama oranlarında görülen düşüklük, modelin daha fazla veri ve optimizasyon ile geliştirilmesi gerektiğine işaret etmektedir. Bu çalışma, nesne tespiti teknolojisinin eski yazıtların okunması ve yorumlanmasında ne denli etkili olabileceğini ortaya koymakla birlikte, gelecekte yapılacak geliştirmeler ile bu tür uygulamaların daha yaygın ve başarılı sonuçlar verebileceğini göstermektedir.

6.2. Araştırmanın Sınırlılıkları

Göktürk harflerinin ve metinlerinin dijital ortamda tespiti ve okunması, yapay zeka ve derin öğrenme tekniklerinin ilerlemesiyle birlikte hız kazanmıştır. Ancak, bu süreçte kullanılan modellerin ve eğitim verilerinin doğasından kaynaklanan çeşitli sınırlılıklar vardır. Bu bölümde, eğitim sürecinde karşılaşılan sınırlamalar ve zorluklar, elde edilen sonuçlar ışığında akademik bir perspektifle ele alınacaktır. Son olarak, kullanılan modellerin işlem gücü gereksinimleri ve hesaplama maliyetleri de dikkate alınması gereken sınırlılıklar arasında yer almaktadır [48]. Bu sınırlılıkları aşmak için önerilen yaklaşımlar arasında veri genişletme (data augmentation) ve transfer öğrenme (transfer learning) teknikleri bulunmaktadır. Veri genişletme, mevcut veri setinin farklı biçimlerde

çoğaltılmasıyla modelin öğrenme kapasitesini genişletir [49].

Eğitim Verilerinin Yetersizliği: Göktürk alfabesiyle ilgili verilerin sınırlı olması, eğitilen modelin performansını doğrudan etkilemektedir. Eğitim sırasında kullanılan veri seti, modelin genelleme kabiliyetini artırmak adına yeterli çeşitliliğe sahip olmalıdır. Ancak, mevcut Göktürk yazıtlarına dayalı veri kümelerinin sayısı oldukça azdır ve bu durum modelin bazı harf ya da kelimeleri tanıma kabiliyetini kısıtlamaktadır. Örneğin, bazı harfler için modelin başarım oranı %0.477 (örneğin R sınıfı için) gibi düşük değerlere sahip olmuştur, bu da o harfin eğitim verilerinde yeterince temsil edilmediğini göstermektedir.

Bazı Harflerin Tespiti Zorluğu: Göktürk alfabesi, klasik Latin alfabesine kıyasla daha farklı ve karmaşık şekillere sahiptir. Bu karmaşık yapılar, özellikle nadir kullanılan harfler ve özel semboller için tespit sürecini zorlaştırır. Örneğin, "A-E" harf grubunda modelin performansı oldukça yüksek (%0.912), ancak "Kutuplu ünsüzlerden kalın D harfi" gibi semboller ya da daha az kullanılan harfler için performans %0.29 gibi oldukça düşük kalmıştır. Bu durum, modelin spesifik karakterler için yetersiz kalabileceğini göstermektedir.

Model Performansının Harfe ve Kelimeye Göre Farklılık Göstermesi: Modelin eğitimi sırasında harf ve kelimeler arasındaki performans farklılıkları dikkat çekicidir. Örneğin, sık kullanılan harfler ve kelimeler için yüksek mAP (Mean Average Precision) değerleri gözlemlenirken, nadir kullanılan karakterler için modelin başarımı oldukça düşük olabilmektedir. Tespit sonuçlarında özellikle "T-U-O" harf grubu için %0.895 gibi yüksek bir başarı elde edilirken, "K1" gibi nadir semboller için modelin performansı %0.438'e kadar düşmektedir. Bu durum, modelin nadir karakterleri tanıma konusunda zorlandığını ve bu karakterlere ait daha fazla veriyle eğitilmesi gerektiğini göstermektedir.

Modelin Yüksek Parametre Sayısı: Modelin toplamda 236 katman ve 23,228,308 parametre ile eğitilmesi, hesaplama gücü açısından ciddi bir kaynak ihtiyacı doğurmaktadır. Her ne kadar modern GPU'lar bu tip işlemler için optimize edilmiş olsa da, eğitim süreci için gereken donanım maliyetleri ve zaman, araştırmayı sınırlayan

unsurlar arasındadır. Özellikle derin öğrenme modelinin boyutu ve karmaşıklığı arttıkça, eğitimi tamamlama süresi de aynı oranda artmaktadır. Göktürk harfleri gibi sınırlı bir veri setiyle çalışan bir model için bu kadar yüksek sayıda parametre gereksiz olabilir ve daha optimize edilmiş, küçük bir model bu tarz spesifik görevler için yeterli olabilirdi.

Genelleme Sorunları: Modelin eğitim sırasında erken durdurma (early stopping) uygulanmış ve en iyi sonuçlar 44. epochta gözlemlenmiştir. Erken durdurma, modelin aşırı öğrenmesini (overfitting) engellemek amacıyla kullanılan bir tekniktir. Ancak bu yaklaşım, genelleme yeteneğinin daha da geliştirilebileceği durumlarda erken müdahale ederek modelin potansiyel performansını sınırlayabilir. Modelin performansının epoch sayısı arttıkça iyileşmediği durumlarda erken durdurma bir avantaj sağlayabilir, ancak farklı veri kümeleri üzerinde test edilmediği sürece genelleme yeteneği hakkında kesin bir sonuç çıkarmak zordur.

Hatalı Pozitif ve Negatif Tespitler: Tespit sonuçları incelendiğinde, bazı karakterler ve kelimeler için hatalı pozitif ya da negatif tespitlerin yapıldığı gözlemlenmiştir. Örneğin, “OK” kelimesi için tespit edilen doğruluk %0.995 iken, “ÖK-ÜK” birleşimi için bu oran %0.838’e kadar düşmektedir. Bu durum, modelin benzer yapılı harf ve kelimeleri ayırt etme konusunda zorlanabileceğini göstermektedir. Özellikle bazı Göktürk harflerinin şekilsel olarak birbirine yakın olması, modelin sınıflandırma hatalarına yol açmasına neden olabilir.

Eğitim Verilerinde Eksiklikler: Göktürkçe yazıtlar sınırlı sayıda olduğu için eğitim verileri eksik kalmaktadır. Bu durum, modelin daha geniş bir veri setiyle test edilmeden, tam anlamıyla genel geçer bir sonuca varılamayacağını göstermektedir. Mevcut eğitim verisiyle model, bazı kelime ve harfleri doğru bir şekilde öğrenmiş olsa da eğitim veri setinin büyütülmesi ve çeşitlendirilmesi gerekmektedir. Aksi halde model, gerçek dünyadaki daha farklı ve karmaşık Göktürk yazıtlarını tanıma konusunda zorlanabilir.

Göktürk harfleri ve kelimelerinin tespiti üzerine yapılan bu çalışmada, modelin performansı belirli sınırlılıklara rağmen genel olarak olumlu sonuçlar vermiştir. Ancak, eğitim verilerinin sınırlılığı, bazı harflerin tanınmasındaki zorluklar ve modelin genelleme kabiliyeti gibi faktörler, gelecekteki çalışmalar için dikkat edilmesi gereken

kritik unsurlardır. Daha geniş ve dengeli bir veri seti, daha optimize modellerle birleştirildiğinde Göktürk metinlerinin otomatik tespiti daha yüksek doğruluk oranlarıyla mümkün olabilir.

7. SONUÇ VE ÖNERİLER

Bu çalışmada, Göktürk alfabesiyle yazılmış metinlerin otomatik olarak tespit edilmesi amacıyla derin öğrenme ve yapay zeka tekniklerinin kullanımı incelenmiş ve YOLOv8 modelinin performansı analiz edilmiştir. Araştırma, sınırlı sayıdaki tarihi Göktürk yazıtlarının dijital ortamda tanımlanabilmesi ve analiz edilebilmesi açısından önemli bir katkı sunmaktadır. Bununla birlikte, elde edilen sonuçlar ve süreçte karşılaşılan sınırlamalar, konunun daha derinlemesine incelenmesi gerektiğini ve gelecekteki araştırmaların bu sınırlamaları göz önünde bulundurarak ilerletilmesi gerektiğini ortaya koymaktadır.

7.1. Araştırmanın Genel Değerlendirilmesi

Elde edilen sonuçlar genel olarak incelendiğinde, YOLOv8 modelinin Göktürk alfabesindeki belirli harf ve kelimeleri başarılı bir şekilde tespit edebildiği gözlemlenmiştir. Modelin ortalama doğruluk oranı (mAP) %90 seviyelerine ulaşmış, bazı karakterler için bu oran %95'in üzerine çıkmıştır. Özellikle sık kullanılan ve açık şekillerle ifade edilen harfler için yüksek başarı oranları elde edilmiştir. Örneğin, “T-U-O” harf grubu için %0.895 gibi bir performans sergilenmiştir. Bu, modelin yeterli veri ile eğitildiğinde tarihsel metinlerde başarılı sonuçlar verebileceğini göstermektedir.

Ancak, nadir kullanılan ve karmaşık yapıya sahip harf ve semboller için modelin performansının belirgin şekilde düştüğü tespit edilmiştir. Örneğin, “Kutuplu ünsüzlerden kalın K” gibi nadir semboller için doğruluk oranının %0.438 seviyelerine inmesi, modelin bazı sembolleri öğrenmede zorluk çektiğini göstermektedir. Bu durum, veri setindeki dengesizlikten kaynaklanmaktadır; yani sık kullanılan harfler veri setinde daha fazla yer alırken nadir semboller yeterince temsil edilmemiştir.

Eğitim süreci boyunca uygulanan erken durdurma tekniği ile model, 44. epochta durdurulmuştur. Bu, aşırı öğrenmeyi (overfitting) engellemek için kullanışlı bir strateji olmakla birlikte, modelin genelleme yeteneğini tam anlamıyla değerlendirmeyi zorlaştırmıştır. Modelin daha fazla epoch ile eğitilmesi durumunda bazı nadir semboller üzerindeki performansın iyileşip iyileşmeyeceği belirsiz kalmıştır. Bu sebeple, erken

durdurmanın model performansı üzerindeki etkileri daha detaylı incelenmeli ve farklı veri setleri üzerinde test edilmelidir. Örneğin, eğitim sürecinde kullanılan veri seti genişletilip daha dengeli hale getirildiğinde erken durdurmanın etkileri daha net gözlemlenebilir.

Göktürk harflerine yönelik mevcut veri setinin sınırlı olması, araştırmanın en büyük sınırlamalarından birini teşkil etmektedir. Eğitimde kullanılan veri setinin dengesizliği, bazı harflerin yeterince temsil edilememesi nedeniyle modelin öğrenme sürecinde sorunlara yol açmıştır. Özellikle nadir kullanılan semboller ve özel işaretler için düşük doğruluk oranları elde edilmesi, bu sınırlamanın bir sonucudur. Ayrıca, Göktürk yazıtlarının dijitalleştirilmesinde kullanılan veri setlerinin sayısal ve yapısal eksiklikleri, modelin genelleme kabiliyetini sınırlayan önemli bir faktör olarak karşımıza çıkmaktadır.

Araştırmada kullanılan YOLOv8 modelinin toplamda 236 katman ve 23,228,308 parametre ile eğitilmiş olması, modelin karmaşıklığını ve hesaplama maliyetini artırmaktadır. Derin öğrenme modellerinin performansı genellikle parametre sayısıyla doğru orantılı artış gösterse de belirli bir noktadan sonra bu parametrelerin aşırı artması, hesaplama süresini ve kaynak ihtiyacını yükseltir. Bu nedenle, Göktürkçe metinler gibi spesifik ve nispeten küçük veri kümeleri için daha optimize ve hafif modellerin denenmesi, hesaplama verimliliği açısından faydalı olabilir. Daha küçük boyutlu ve optimize edilmiş modeller, daha hızlı eğitim süresi ve daha az kaynak kullanımı sağlayabilir.

Eğitim sürecinde, modelin bazı harf ve kelimelerde hatalı pozitif ve negatif tespitler yaptığı gözlemlenmiştir. Özellikle “OK” ve “ÖK-ÜK” gibi benzer harf kombinasyonlarında modelin doğru tespit oranları arasında belirgin farklılıklar oluşmuştur. Bu, modelin benzer karakterler arasındaki farkı ayırt etme konusunda zorlandığını göstermektedir. Benzer sorunlar Latin alfabesindeki bazı harflerde de görülse de Göktürk alfabesinin karmaşık ve birbirine yakın şekilleri bu sorunu daha belirgin hale getirmiştir. Gelecekte bu sorunu aşmak için daha fazla veri ve daha sofistike model yapılandırmalarına ihtiyaç duyulmaktadır.

Bu çalışma, Göktürk harflerinin ve kelimelerinin tespiti konusunda önemli bulgular ortaya koysa da çeşitli sınırlamalar nedeniyle modelin performansının bazı durumlarda yeterli olmadığı sonucuna ulaşılmıştır. Modelin nadir kullanılan harflerde düşük performans sergilemesi, eğitim veri setinin genişletilmesi gerektiğini açıkça göstermektedir. Aynı zamanda, modelin hesaplama maliyetini azaltmak için daha optimize edilmiş modellerin denenmesi, araştırmanın verimliliğini artırabilir.

Gelecekte yapılacak çalışmalar, özellikle eğitim verilerini genişletme ve dengeli hale getirme üzerine odaklanmalıdır. Ayrıca, modelin genelleme yeteneğini artırmak için farklı optimizasyon tekniklerinin denenmesi ve erken durdurmanın etkilerinin daha kapsamlı analiz edilmesi önerilmektedir. Böylece, Göktürk metinlerinin daha yüksek doğruluk oranlarıyla tespit edilmesi sağlanabilir ve bu alandaki dijitalleştirme çalışmaları hız kazanabilir.

7.2. Gelecek Çalışmaları için Öneriler

Gelecek çalışmalarda, Göktürk alfabesi veri setinin genişletilmesi ve çeşitlendirilmesi önemli bir adım olacaktır. Veri setinin büyütülmesi, nadir karakterlerin daha iyi tanınmasını sağlayacak ve veri artırma teknikleri kullanılarak modelin gerçek dünya koşullarına daha iyi uyum sağlaması mümkün olacaktır. Aynı zamanda, veri seti büyütme ve sentetik veri oluşturma yöntemleri modelin öğrenme kapasitesini artırabilir.

Model optimizasyonu açısından, YOLOv8 yerine daha hafif modellerin kullanılması veya model inceltme ve kuantalama tekniklerinin uygulanması performansı artırabilir. Böylece, hesaplama maliyetleri düşürülürken modelin doğruluğu korunabilir.

Erken durdurma stratejilerinin uyarlanması da önemli bir diğer adımdır. Daha hassas izleme yöntemleriyle, modelin gereksiz yere erken durdurulmasını engelleyerek eğitim süreci optimize edilebilir. Aynı zamanda, transfer öğrenme ve ikincil eğitim teknikleri, küçük veri setleri ile daha verimli sonuçlar elde etmek için önemli fırsatlar sunmaktadır. Son olarak, modelin gerçek dünya koşullarında daha fazla test edilmesi ve tespit edilen harflerin anlamlandırılmasına yönelik hibrit yaklaşımların geliştirilmesi, Göktürk

metinlerinin tanınması ve analizi alanında yeni ufuklar açabilir. Bu yöntemler hem görsel hem de semantik açıdan metinlerin daha iyi anlaşılmasını sağlayabilir.

8. KAYNAKÇA

[1] Tekin, T. (2000). Orhon Yazıtları: Kül Tigin, Bilge Kağan, Tonyukuk. Türk Dil Kurumu Yayınları

[2] Kafesoğlu, İ. (1994). Türk Milli Kültürü. Ötüken Neşriyat

[3] Clauson, G. (1972). An Etymological Dictionary of Pre-Thirteenth Century Turkish. Oxford University Press

[4] Erdal, M. (2004). A Grammar of Old Turkic. Brill

[5] Róna-Tas, A. (1991). An Introduction to Turkology. Szeged University Press

[6] https://tr.wikipedia.org/wiki/Orhun_Yaz%C4%B1tlar%C4%B1#

[7] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

[8] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780.

[9] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 779-788.

[10] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018). A survey on deep transfer learning. In International conference on artificial neural networks (pp. 270-279). Springer, Cham.

[11] Tan, M., Pang, R., & Le, Q. V. (2020). EfficientDet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 10781-10790)

[12] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017, April 17). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[13] Gomes, M., & Medeiros, B. (2020). Artificial intelligence in the reading of ancient scripts. Journal of Cultural Heritage, 45, 221-230.

[14] Assael, Y., Sommerschield, T., Shillingford, B., Kankanahalli, S., & de Freitas, N. (2019). Restoring ancient text using deep learning: A case study on Greek epigraphy. Nature Scientific Reports, 9(1), 1-10.

[15] Gedge, M. (2021). Natural language processing for ancient languages: Applications and future prospects. Digital Humanities Quarterly, 15(3).

[16] Rainey, A. (2022). Preserving cultural heritage with AI technologies. Cultural Heritage Review, 33(2), 105-112.

[17] Fukushima, K. (1980). Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. Biological Cybernetics, 36(4), 193-202.

[18] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[19] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. Advances in Neural Information Processing Systems, 28, 91-99.

[20] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

[21] https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html

[22] Shah, S., & Tembhurne, J. (2023). Object detection using convolutional neural networks and transformer-based models: a review. *Journal of Electrical Systems and Information Technology*, 10, Article 54.

[23] O'Shea, K., & Nash, R. (2015, December 2). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

[24] Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML) (pp. 807-814).

[25] Zeiler, M. D., & Fergus, R. (2013). Visualizing and understanding convolutional networks. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 808-823). Springer, Berlin, Heidelberg.

[26] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[27] He, K., Gkioxari, G., Dollar, P., & Girshick, R. (2018, January 24). Mask R-CNN. arXiv preprint arXiv:1703.06870

[28] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25, 1097-1105.

[29] Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *Visual Geometry Group, Department of Engineering Science, University of Oxford.*

[30] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Proceedings of the 31st International Conference on Machine Learning* (ICML) (pp. 1050-1058).

[31] Ioffe, S., & Szegedy, C. (2015, March 2). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

[32] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934.

[33] Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., & Girshick, R. (2019). Detectron2: A PyTorch library for object detection.

[34] Ultralytics. (2023). YOLOv8: Next-Generation Object Detection Model. Ultralytics Blog.

[35] Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2021). Scaled-YOLOv4: Scaling Cross Stage Partial Network. arXiv preprint arXiv:2102.12430.

[36] Sultana, F., Sufian, A., & Dutta, P. (2019). A Review of Object Detection Models based on Convolutional Neural Network. *arXiv preprint* arXiv:1905.01614.

[37] Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980.*

[38] Loshchilov, I., & Hutter, F. (2019). Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101*

[39] Zhuang, Z., Liu, M., Cutkosky, A., & Orabona, F. (2022, January 31). Understanding AdamW through proximal methods and scale-freeness. arXiv preprint arXiv:2202.00089.

[40] Jocher, G., Chaurasia, A., Qiu, J., & Stoken, A. (2023). YOLO by Ultralytics. GitHub repository. Retrieved from https://github.com/ultralytics/ultralytics.

[41] Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. *arXiv preprint* arXiv:1804.02767.

[42] Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021). YOLOX: Exceeding YOLO series in 2021. arXiv preprint arXiv:2107.08430.

[43] Zhou, X., Wang, D., & Krähenbühl, P. (2019). Objects as points. *arXiv preprint arXiv:1904.07850*.

[44] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 2980-2988).

[45] Al Rabbani Alif, M., & Hussain, M. (2024). YOLOv1 to YOLOv10: A comprehensive review of YOLO variants and their application in the agricultural domain. *arXiv preprint arXiv:2406.10139v1*.

[46] Yaseen, M. (2024, August 28). What is YOLOv8: An in-depth exploration of the internal features of the next-generation object detector. arXiv preprint arXiv:2408.15857v1.

[47] Howard, J., & Ruder, S. (2018, May 23). Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146.

[48] Kirillov, A., Wu, Y., He, K., & Girshick, R. (2020, February 16). PointRend: Image segmentation as rendering. arXiv preprint arXiv:2001.04047.

[49] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), 60.

ÖZGEÇMİŞ

Adı Soyadı: Mevlüt Karakaya

Yabancı Dil: İngilizce

Öğrenim Durumu

Derece	Bölüm / Program	Üniversite	Mezuniyet Yılı
Lisans	Mekatronik Mühendisliği	Marmara Üniversitesi	2021

İş Deneyimi

Yıl	Firma / Kurum	Görevi
2021-	ANT Mühendislik	Yazılım Mühendisi

Yayınlar

Karakaya, M., Çelebi, M. F., Gök, A. E., & Ersoy, S. (2022). Discovery of agricultural diseases by deep learning and object detection. *Environmental Engineering and Management Journal*, *21*(1), 163-173.