KOCAELİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

LİSANS TEZİ

Gömülü Sistemler Üzerinde Derin Öğrenme Tabanlı Gerçek Zamanlı Çok Haneli Sayı Tanıma Sistemi

KOCAELİ 2022

KOCAELİ ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME PROJESİ

Gömülü Sistemler Üzerinde Derin Öğrenme Tabanlı Gerçek Zamanlı Çok Haneli Sayı Tanıma Sistemi

Tezin Savunulduğu Tarih: 20.05.2022

ÖNSÖZ

Bu dökümandaki tüm bilgiler, etik ve akademik kurallar çerçevesinde elde edilip sunulmuştur. Ayrıca yine bu kurallar çerçevesinde kendime ait olmayan ve kendimin üretmediği ve başka kaynaklardan elde edilen bilgiler ve materyaller (text,resim,şekil,tablo vb.) gerekli şekilde referans edilmiş ve dökümanda belirtilmiştir.

İÇİNDEKİLER ……

Gömülü Sistemler Üzerinde Derin Öğrenme Tabanlı Gerçek Zamanlı Çok Haneli Sayı Tanıma Sistemi

ÖZET

Dijitalleşme, günümüzün en önemli gelişmelerinden biri haline geldi. Dijitalleşmeyle birlikte, el yazısı kullanan bir çok alanda bilgilerin sayısallaştırılması ve taranması konusu ön plana çıkmaya başladı. Sayı tanıma sistemlerinin sağlayacağı avantajlar ve kolaylıklar da bu sistemlerin çekiciliğini artırmaktadır. Sayı tespit, analiz ve tanıma konusu, araçların plaka tanımasından banka hesap kontrolüne kadar birçok alanda etkin olarak kullanılmaktadır. Ancak sayı tespit ve tanıma çözümlerinin elde edilmesinde baskı ile yazılmış bir sayının tanınmasında yaşanabilecek basamak sayılarının tespiti gibi temel zorluklar bulunmaktadır.

Modern gereksinimlerle başa çıkmak için, birleşik çok basamaklı sayının tanınması gereklidir. Bu gereklilik göz önünde bulundurularak, bu çalışmada **gömülü sistemler üzerinde derin öğrenme tabanlı gerçek zamanlı çok haneli sayı tanıma sistemi tasarlanacaktır.** Bu projede gömülü sistem olarak Nvidia Jetson Nano Geliştirme Kit'i kullanılacaktır. Karta yerleştirilecek olan CSI kamera vasıtasıyla gömülü sistem kamerasından görüntüler alınacak, daha sonra sayıların bulunduğu bölge lokalize edilecek, en son olarak da sayıları görüp çok basamaklı tanıma işlemini gerçekleştirmesi hedeflenmiştir. Derin öğrenme tabanlı çalışacak sistemde önce modeller eğitilecek ve daha sonra eğitilmiş modellerin üzerinde alınan sayı görüntüleri işlemden geçirilip yazılan sayının hangi sayı olduğu tahmin edilmesi amaçlanmıştır. Ayrıca sisteme birden fazla haneli sayı gönderildiği durumlarda yazılmış her bir rakamın ayrı ayrı ne olabileceğine dair olasılıkları derin öğrenme ile işleyerek bu çok haneli sayısının kaç olduğunu bulabilmesi gerçekleşecektir. Deneysel çalışma MNIST ve Street View House Number (SVHN) veri kümeleri üzerinde test edilecek ve sınıflandırma için derin öğrenme ağı içerisinde yer alan Konvolüsyonel Sinir Ağları (CNN) kullanılacaktır. Elde edilen sonuçlar karşılaştırmalı olarak analiz edilecek ve grafiksel olarak eğitim ve test sonuçları verilecektir.

Anahtar kelimeler: Sayı tespiti, Derin öğrenme, Nvidia Jetson Nano, Çok Haneli Sayılar

Deep Learning Based Real-Time Multi-Digit Number Recognition System on Embedded Systems

ABSTRACT

Digitalization has become one of the most important developments of our time. With digitization, the issue of digitizing and scanning information in many fields using digit has begun to come to the fore. The advantages and conveniences of digit recognition systems also increase the attractiveness of these systems. Number detection, analysis and recognition are used effectively in many areas from license plate recognition to bank account control. However, there are fundamental difficulties in obtaining number detection and recognition solutions, such as determining the number of digits that can be experienced in recognizing a number.

To cope with modern requirements, recognition of the combined multi-digit number is necessary. Considering this requirement, in this study, a deep learning-based real-time multi-digit number recognition system will be designed on embedded systems. In this project, Nvidia Jetson Nano Development Kit will be used as an embedded system. Images will be taken from the embedded system camera via the csi camera to be placed on the card, then the region where the numbers are located will be localized, and finally, it is aimed to see the numbers and perform the multi-digit recognition process. In the system that will work based on deep learning, first the models will be trained and then the number images taken on the trained models are processed and it is aimed to predict which number is the number. In addition, in cases where more than one digit number is sent to the system, it will be possible to find out how many digits this multi-digit number is by processing the probabilities of what each number may be separately with deep learning. Experimental study will be tested on MNIST and Street View House Number (SVHN) datasets and Convolutional Neural Networks (CNN) within the deep learning network will be used for classification. Obtained results will be analyzed comparatively and training and test results will be given graphically.

Keywords: Number detection, Deep learning, Nvidia Jetson Nano, Multi-Digit Numbers

1. GİRİŞ

İnsanlar için oldukça kolay olmasına rağmen , bir zemin üzerindeki çizgi ve eğrilerin rakamlar olarak algılanması oldukça zor bir problemdir. Sayı tanıma hala tam olarak çözülmüş bir problem değildir. El yazısı belgelerin çok sayıda olması ve el yazısı belgelerin dijital formlar ve veritabanları gibi dijital sistemler aracılığıyla daha erişilebilir olan dijital kayıt kopyalarına dönüştürülmesi önemli bir konudur. El yazısı ile tanımadaki zorluk, çok fazla sayıda değişik yazı yazma karakteri olması ve kişiden kişiye farklılıklar göstermesinin yanında rakamların birbirine bağlı yazılmasından kaynaklanmaktadır. İnsan görme sistemi sayıların büyüklük ve yön farklılıklarından etkilenmemektedir ancak otomatik bir sistemde bu kavramlar sayı tanımada problemlere neden olabilmektedir. Ayrıca kağıt üzerine yazılmış ve görüntüsü alınmış bir rakamın görüntüsü çok fazla gürültü de içerecektir. Bu sebeple el yazısı ile yazılmış rakamları tanıma probleminin zor ve henüz çözülmemiş bir problem olduğu gözlemlenmektedir. Sayıların dijital versiyonda sayılara dönüşümünü otomatik olarak gerçekleştirmek için sayı tanıma yöntemleri kullanılmaktadır. Bu tanıma süreçlerini yürütmek için çeşitli makine öğrenmesi ve derin öğrenme yöntemlerinden yararlanılmaktadır.

1.1 Literatür

Literatürde yapılan çalışmalara bakıldığı zaman harf tanıma, Arap rakamları tanıma gibi birçok proje gerçekleştirilmiştir. Bu projelerde genel olarak makine görmesi ve derin öğrenme kullanılmıştır. Hayder M. Albehadili ve ark. Arap rakamları tanıma çalışması yapmışlardır. Yaptıkları çalışmada, derin öğrenme sinir ağlarını kullanarak el yazısı ile yazılmış Arap rakamları tanıma projesini gerçekleştirmiştir ve 0.01 öğrenme oranı vererek 70.000 verili MNIST ile eğitmiş modeli %98 oranında başarıya ulaştırmışlardır [1].Nagai et ve ark. tek Japonca tarihi bitişik el yazısı metin satırlarının tanınması için yeni bir yöntem sunar. Yöntem, 46 Hiragana karakterinden oluşan metinler için %95'in üzerinde, binlerce Kanji karakteri içerdiğinde ise %84.08'in üzerinde doğruluğa ulaşmaktadır[2]. Jin-Ho Kim ve ark. araç plaka tanıma sistemi için kenar tabanlı segment görüntü üretimi ile plaka tanıma işlemi gerçekleştirmişlerdir ve gerçek zamanlı plaka tanıma algoritmasının deneysel sonuçları, günlük ortalama 1.535 araçta plaka algılama oranının %97,5 ve algılanan plakaların genel karakter tanıma oranının %99,3 olduğunu göstermektedir[3]. Yann LeCun ve ark. 60.000 eğitim görüntüsü ve 10.000 test görüntüsü içeren, el yazısıyla yazılmış sayı içeren veri kümesi oluşturmuşlardır (MNIST) [4]. Muhammed Asıf ve ark. görüntülerin taranması ve taranan bilgilerin dijital formata dönüştürülmesi alanında çalışma yapan, 18 basamağa kadar olan sayıyı tanıma algoritması üzerinde çalışmışlardır ve bu yaklaşımı herkese açık SVHN veri setinde değerlendirip sokak numaralarını tanımada %96'nın üzerinde doğruluk elde etmişlerdir. Rakam başına tanıma görevinde, %97,84 doğruluk elde ettiğini görmüşlerdir [5]. El yazısı metinler için Salvador España-Boquera ve arkadaşlarının hibrit Gizli Markov Modeli (HMM) önerilmiştir. Bunda, optik modelin yapısal kısmı Markov zincirleri ile modellenmiştir ve emisyon olasılıklarını tahmin etmek için bir Çok Katmanlı Algılayıcı kullanılmıştır[6]. R. Bajaj, L. Dey, S. Chaudhari ve diğerleri, Devanagari Rakamlarının sınıflandırılması için yoğunluk özellikleri, moment özellikleri ve tanımlayıcı bileşen özellikleri olmak üzere üç farklı türde özellik kullanmıştır. El yazısı Devanagari rakamları için %89.6 doğruluk elde etmişlerdir[7].

M. Hanmandlu, O.V. Ramana Murthy, çalışmalarında el yazısı Hintçe ve İngilizce rakamları tanınmasını sunmuştur. Genel tanınma oranı Hintçe rakamlar için %95 ve İngilizce rakamlar için %98,4 olarak bulunmuştur[8]. Mohammed Z. Khedher ve diğerleri, Karakterlerin tanınması büyük ölçüde kullanılan özelliklere bağlı olduğunu açıklamıştır. El yazısı Arapça

karakterlerin çeşitli özellikleri seçilmiştir. Seçilen özelliklere göre tanıma sistemi oluşturulmuştur. Sistem, el yazısı Arapça karakter örnekleriyle eğitilmiş ve test edilmiştir. Seçilen özelliklere dayalı tanıma, sayılar için %88, harfler için %70 ortalama doğruluk verir.[9]

"Gün ışığı koşullarında, elde edilen görüntüler üzerinden sayılarının tespit edilebilir ve geliştirilecek derin öğrenme modelleri ile bu sayıların sınıflandırılarak gerçek zamanlı sistemler üzerinde de yüksek başarımla tespit edilebilir" yargısı bu projenin üzerine kurulduğu hipotezdir. Bu hipotez kapsamında çalışmada genel olarak ele alınacak problemler;

- Gömülü sistem kartından görüntülerin net alınabilmesi ve derin öğrenme algoritmalarının yüksek başarımla kartta çalışabilmesi
- Sayıları yüksek doğruluk oranıyla tespiti
- Yüksek basamaklı sayılardaki basamakların tespitinin zorluğu şeklindedir.

1.2 Amaç Ve Hedefler

Önerilen bu projenin başlıca hedefleri aşağıdaki şekilde özetlenebilir:

- Önerilen çalışmada basılı belgelerdeki çok basamaklı sayıları yüksek doğrulukla tespit edilebilecektir..
- Geliştirilecek derin öğrenme modeli ile çok basamaklı sayılar yüksek doğrulukla tanınabilecektir.
- Önerilen sistem ile tanıma ve genel hesaplama karmaşıklığı azaltılacaktır.
- İki farklı veri seti üzerinde gerçekleştirilen kapsamlı deneyler ile önerilen yöntem ve sistemin etkin bir şekilde çalıştığı gösterilecek ve elde edilen sonuçlar mevcut yöntemlerle karşılaştırılacaktır.

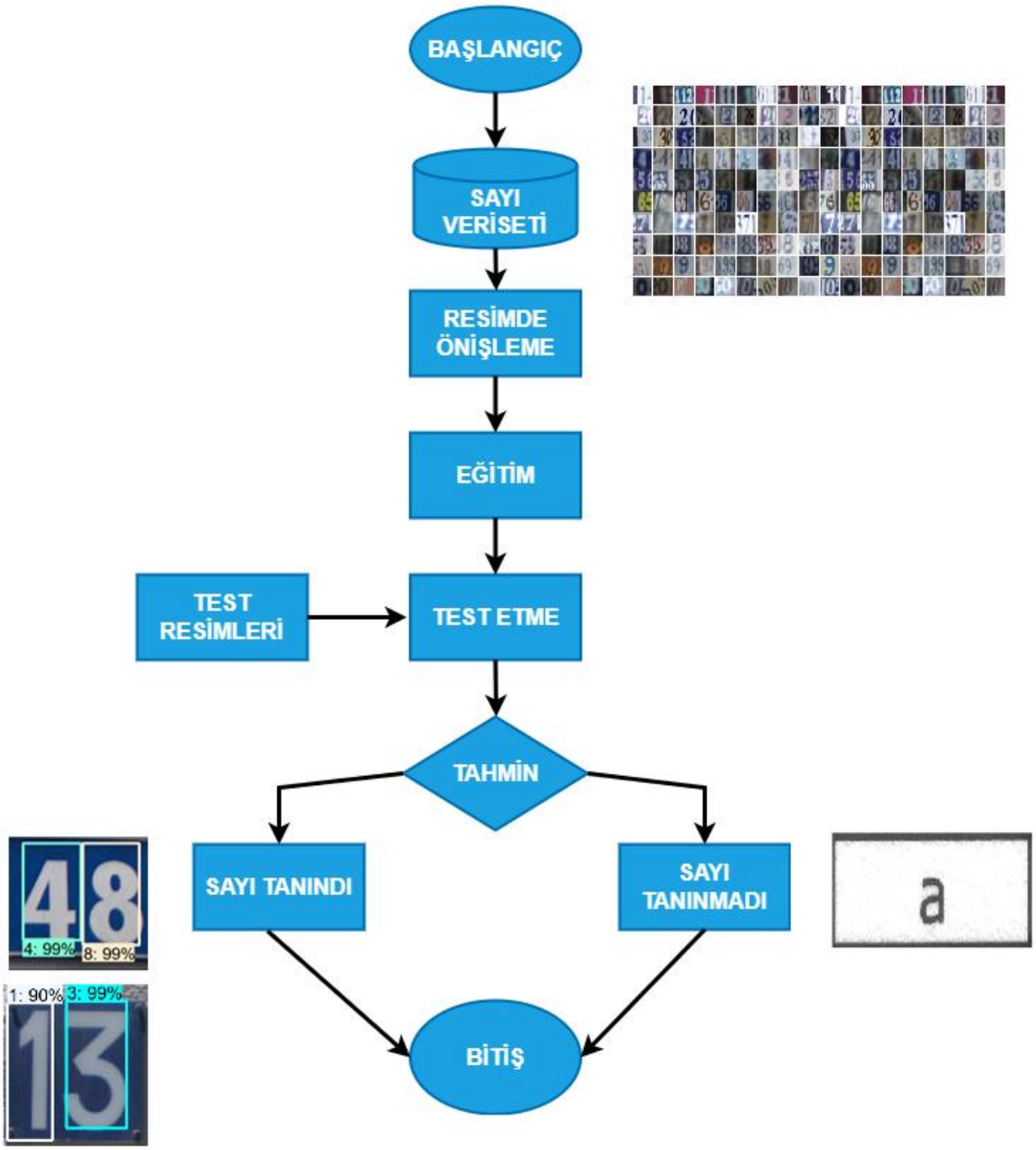
Yukarıdaki hedeflere ulaşabilmek için bu çalışmada,

- Jetson Nano üzerinde bulunan CSI kamera ile görüntüler alınması
- Rakamlara Tanıma ve Sınıflandırma İşlemi Uygulanması
- Rakamların basamak sayısının bulunması
- Street View House Number (SVHN) veri kümeleri üzerinde derin öğrenme yöntemiyle sayı görüntüleri sınıflandırılıp yazılan sayının hangi sayı olduğu tahmin edilmesi
- Sonuçlar karşılaştırmalı olarak analiz edilecek ve grafiksel olarak eğitim ve test sonuçları verilmesi amaçlanmıştır.

2. YÖNTEM

Literatürde, görüntülerden sayıların tanınması için çeşitli yöntemler önerilmiştir. Bu yöntemler, kullanılan veri kümesine, uygulanan yapay zeka yöntemlerine ve tek haneli veya çok haneli tanıma kabiliyetine göre vb. gibi çeşitli açılardan sınıflandırılabilirler. Bu çalışmada, tanıma tekniklerinin sınıflandırılması, tek basamaklı veya çok basamaklı tanıma yeteneklerine göre gerçekleştirilmektedir.

İmage dosyası Jetson Nano kartı Nvidia'nın resmi sitesinden indirilip Jetson Nano kart için alınan hafıza kartına Etcher yazılım ile yüklenmektedir. Bu image dosyası sayesinde CUDA gibi kütüphaneleri Linux Ubuntu işletim sistemine yüklenecektir. Kart içinde derin öğrenme kurulumları uygulanacaktır. Eğitime alınacak modelin önceden hazırlanmış veri seti USB bellek ile Jetson Nano kartına yüklenecektir. Bundan sonraki aşamada model yüklenen dataseti ile eğitim süreci gerçekleştirebilir. Ayrıca önceden eğitilmiş bir model kartta çalıştırılabilecektir.



2.1 Tek Basamaklı Tanıma Teknikleri

Tek basamaklı tanıma teknikleri: Kevin ve ark. [10], MNIST ve NORB veri setlerini kullanarak tek basamakta SIFT ve PMKSVM yaklaşımına dayalı iki aşamalı bir sistem önermişler ve sistem %62,9 ve 65 doğruluk oranlarıyla çalışmaktadır. Ranzato ve ark. [11], MNIST ve Caltech 101 veri setlerini kullanarak tek haneli öğrenme yaklaşımı için denetimsiz bir yöntem önermişlerdir. Sistemleri %54 doğruluk oranı ile çalışmaktadır. Yang ve Pu [12],

CNN tabanlı mobil cihazlar için tasarlanan el yazısıyla yazılan çok basamaklı sayıların tanınması için bir teknik önerdiler ve MNIST veritabanında %99.07 ilk basamaktaki tek basamaklı doğruluk elde ettiler. Nguyen ve ark. [13] el yazısı rakam tanıma için hibrit CNN-GRU modelini önerdiler. Modeli değerlendirmek için 70.000 görüntü içeren MNIST veri setini kullandılar ve %99.21 doğruluk oranı elde ettiler. Shovon ve ark. [14] el yazısıyla yazılan Bangla numarasını tanımak için çok katmanlı CNN tabanlı bir yöntem önerdiler. Tekniğin performansını değerlendirmek için 85000+ basamaklı görüntülere sahip NumtaDB veri setini kullandılar. %98,96 doğruluk oranı elde ettiler.

2.2 Çok Basamaklı Tanıma Teknikleri

Çok basamaklı tanıma teknikleri: Netzer ve ark.[15], denetimsiz öğrenmeye dayalı görüntülerinden rakamları tanıma sistemi önerdiler. Eğitim aşamasında yığınlanmış seyrek otomatik kodlayıcılar ve k-araç tabanlı sistemler dahil olmak üzere çok sayıda denetimsiz özellik öğrenme tekniğinden yararlandılar. En son olarak SVM sınıflandırma ile %90,6 doğruluk oranı tespit etmişlerdir. Sermanet ve ark., CNN tabanlı ev numarası basamaklarının sınıflandırılması sistemi tasarlanmışlardır. Yazarlar çalışmalarını SVHN veri kümesi üzerinde test etmişler ve %95.10 doğruluk oranı elde etmişlerdir[16]. Goodfellow ve ark.[17], SVHN veri kümesi için tek karakterli tanıma çalışması yapmışlardır ve sistem %97,53 doğruluk oranı ile çalışmaktadır. Jeon ve ark. gerçek zamanlı çok basamaklı tanıma geliştirmişlerdir. Gömülü sistem üzerinde derin öğrenme tabanlı MNIST ve kendi oluşturdukları bazı verileri kullanmışlardır. Sistem %98,23 doğruluk oranı ile çalışmaktadır. Liu ve Bhanu, spordaki oyuncuların forma numaralarını tanımak için CNN (RCNN) tabanlı bir model geliştirdiler ve kendi veri setleri üzerinde test etmişlerdir. %94.09 tanıma doğruluğu elde edilmiştir[18]. Cubuk ve arkş. mevcut öğrenilmiş güçlendirme tekniklerinin sistematik dezavantajları olduğunu, yani eğitim karmaşıklığında ve hesaplama maliyetinde artış olduğunu vurguladı. Bu sorunları ele almak için, arama alanını azaltan ve ayrı bir proxy görevi yerine gerçek görevler üzerinde eğitilmesine izin veren bir teknik önerilmiştir.

3. KULLANILAN TEKNOLOJİLER

3.1 Nvidia Jetson Nano Geliştirme Kiti

NVIDIA Jetson Nano Geliştirici Kiti, yapay zeka algoritmalarının çalıştırılabilmesi için geliştirilmiştir. Geliştiriciler, görüntü sınıflandırma, nesne tespiti, segmentasyon ve dil işleme gibi yapay zeka yazılım geliştirme ortamlarını ve modellerini bu gömülü sistem kartı üzerinde çalıştırabilirler. Jetson Nano aynı zamanda derin öğrenme, görüntü işleme, GPU programlama, multimedya işleme ve daha fazlası için gerekli olan kart destek paketi (BSP), Linux İşletim Sistemi, NVIDIA CUDA®, cuDNN ve TensorRT™ yazılım kütüphanelerini içeren NVIDIA JetPack yazılımı tarafından desteklenmektedir. Aynı JetPack SDK, bütün NVIDIA Jetson(™) ürün ailesinde kullanılmaktadır ve NVIDIA'nın dünyada öncü model eğitim ve model yerleştirme platform yazılımı tarafından tamamıyla desteklenmektedir. 4 çekirdekli 64-bit ARM CPU 4 GB bellek sunar ve aynı anda birkaç yüksek çözünürlüklü sensörü işler [13]. Şekil 1'de NVIDIA Jetson Nano Geliştirici Kiti ve csi kamera görüntüleri gösterilmiştir.

Şekil 1. Jetson Nano Geliştirme Kiti ve CSI Kamera

3.2 Kullanılan Veri Setleri

Çalışmada kullanılacak MNIST ve SVHN veri kümeleri Şekil 2 ve Şekil 3'de gösterilmektedir. MNIST, görüntüyü işlemek ve analiz edilmesini sağlamak için tercih edilen bir veri kümesidir. Bu veri kümesi el yazısıyla yazılmış rakamların görüntülerinden oluşmaktadır.

Şekil 2. MNITS veri kümesi [4]

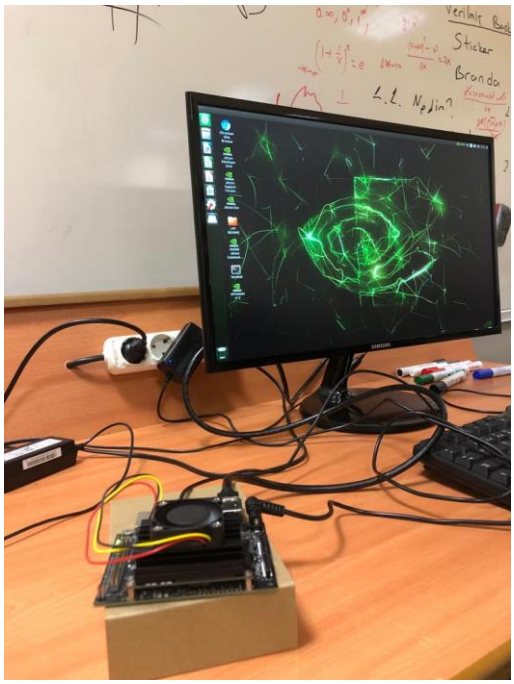
SVHN veri kümesi [14], sınırlayıcı kutular ve etiketlerle açıklanmış sokak numaralarının görüntülerini içermektedir. Bu veri kümesi eğitim, test ve ekstra olarak üzere üçe ayrılmıştır. SVHN'den alınan görüntüler, çeşitli boyut ve basamaklarda sayılardan oluşmaktadır.

Şekil 3. SVHN veri kümesi [14]

4. GENEL MİMARİ

Gömülü sistemlerde rastgele fotoğraflarda çoklu karakter tanıma genellikle zor bir problem olarak bilinmektedir. Literatürde ki çalışmalarda rakam tanıma işlemlerinin genellikle mobil platformlarda yapıldığı görülmüştür. Fakat düşük CPU performansı ve sınırlı bellek kaynakları düşünüldüğünde derin öğrenme mimarilerinin mobil platformdaki uygulamalarından istenilen başarımlar elde edilememektedir. Bu sebepten bu çalışmada CPU ve GPU performansı yüksek olan Jetson Nano Geliştirme kiti kullanılacaktır. Görüntüler

Jetson Nano üzerinde bulunan CSI kamera ile elde edilecektir. Kameradan alınan görüntülerdeki rakam, hesaplama adımlarını azaltmak için ilk önce ön işleme tabi tutularak bölümlere ayrılacak olup detection işlemi yapılacaktır. Buradaki amaç sayıyı rakamlara ayırarak hesaplama adımını azaltmaktır. Ayrıca burada sistem rakamların konum bilgilerine göre hangi rakamın hangi sayıya ait olduğunu da tespit edecektir. Çalışmada detection işlemi sayesinde ilgili piksel değerleri ve R-CNN algoritmalarının kullanılması amaçlanmaktadır. Detection işleminden sonra rakamlar işaretlenerek sayının basamak hesabı yapılacaktır. Rakam detectiondan sonra orjinal görüntü rakam tespiti için CNN mimarisine verilecektir.

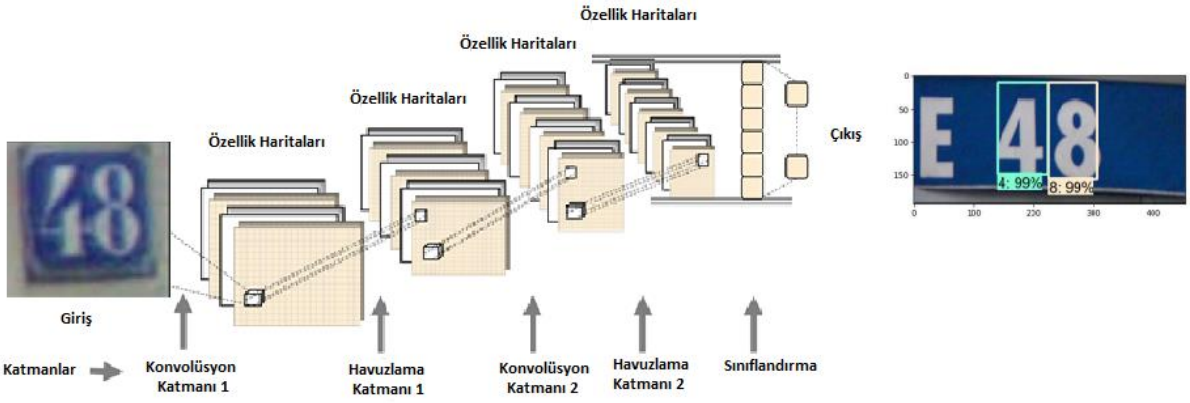


Şekil 4. Jetson Nano Çalışması

4.1 CNN Kullanılarak Görüntülerin Sınıflandırılması

Literatürde görüntü datalarını analiz etmek için kullanılan ve yüksek performanslı sonuçlar veren çoklu derin sinir ağlarıdır [15]. Literatürdeki çalışmalar incelendiğinde nesne tanıma, görüntü sınıflandırma gibi alanlarda sıklıkla kullanılmaktadır. Temel olarak bir CNN mimarisi incelendiğinde eğitilebilen bir çok katmandan oluşabilmektedir. CNN mimarisi temel olarak giriş, konvolüsyon, havuzlama ve tam başlantı katmanlarından oluşmaktadır [16].

Yapılacak olan çalışmada görüntü sınıflandırması için CNN mimarisi kullanılacaktır. Şekil 5'te genel CNN modeli gösterilmiştir.



Şekil 5. Genel CNN modeli

5. DERİN ÖĞRENMEDE HİPER-PARAMETRELER

5.1 Epoch

Bir modelin eğitim esnasında verilerin hepsi aynı zaman diliminde eğitime katılmaz. Çünkü zaman ve bellek açısından maliyetlidir. Belirli parçalara bölünür ve eğitimde yer alırlar. İlk parça eğitildikten sonra, modelin başarısı test edilir ve başarıya göre **backpropagation** ile ağırlıklar güncellenir. Bu işlemden sonra yeni bir eğitim kümesiyle tekrar model eğitilir ve tekrar ağırlıklar güncellenir. Bu işlemler her seferinde model için en uygun ağırlık değerlerini hesaplamak için tekrarlanır. Ve bu adımların her birine 'epoch' denilir. Projede 10.000 görsel ile 40 epochda eğitildi.

Derin öğrenmede problemi çözmek için en uygun ağırlık değerleri adım adım hesaplanır. Dolayısıyla ilk epoch'larda başarı oranı düşük olacaktır, epoch sayısı arttıkça başarı oranı da artacaktır. Ek olarak belli bir noktadan sonra modelin öğrenme durumu oldukça azalacaktır.

5.2 Mini Batch

Epoch başlığı altında bir modeli eğitirken tüm verilerin aynı anda kullanılmayacağından bahsetmiştik. Çünkü öğrenmenin her iterasyonunda **backpropagation** işlemi ile ağ üzerinde geriye dönük olarak gradyan hesaplaması yapılmakta ve ağırlık değerleri bu şekilde güncellenmektedir. Veri sayısı ne kadar fazla ise söz konusu hesaplama işlemi o kadar uzun sürecektir. Bu problemi çözmek için veri seti küçük gruplara ayrılmakta ve öğrenme işlemi seçilen bu küçük gruplar üzerinde yapılmaktadır. Bahsedilen bu küçük parçaların işlenmesine

‘mini-batch’ denir. Model tasarımında mini-batch değeri modelin aynı anda kaç veriyi işleyeceği anlamına gelir. Belirlenen batch değerinin GPU belleğine sığması gerekiyor. Bu nedenle batch size 2^n şeklinde belirlenmelidir.

5.3 Veri Setinin Boyutu

Derin öğrenmede veri setinin büyüklü ve çeşitli olması çok önemlidir. Çeşitlilik ve büyüklük arttıkça modelin öğrenme oranı da artacaktır. Bunun yanında veri seti büyüklüğü öğrenme süresini arttıracaktır. Veri setinin büyüklüğünün sürekli artacak olması, modelin başarı oranın aynı oranda artacağı anlama gelmez. Belli bir noktadan sonra başarı oranının artma hızı azalacaktır. Ek olarak öğrenme sürecini hızı ile kullanılan görüntülerin görüntü kalitesi arasında ters oran vardır.

6. OPTİMİZASYON ALGORİTMASININ SEÇİMİ

İsminden de anlaşılacağı üzere Optimizasyon bir problemde mümkün olan ‘optimum’ yani en iyi çözümü bulmaktır. Makine öğrenmesinde hata oranını azaltmak için sık kullanılan 6 optimizasyon algoritması vardır ; SGD, Momentum, Adagrad, RMSProp, Adadelta, Adam. Uygulama geliştirilirken deneysel olarak Adam, SGD, RMSprop optimizer kullanılmıştır.

6.1 Stochastic Gradient Descent (SGD)

Türkçe karşılığı Olasılıksal Dereceli Azalma olan Stochastic Gradient Descent algoritması[19] hızlı bir algoritmadır, her adımda güncellenmektedir ve çalışmada her eğitim örneği için parametreler güncellenmektedir. Sık güncellemeler sebebiyle, parametre yüksek varyans değerlerine sahiptir ve hata fonksiyonun farklı yoğunluklarda dalgalanmasına neden olur. Stochastic Gradient Descent algoritmasında tüm veri seti yerine her güncelleme için rastgele birkaç örnek seçilmektedir. Tüm veri setini daha az gürültülü bir şekilde kullanmak faydalı olsa da veri setinin çok büyük boyutlara ulaşması durumunda sorunla karşılaşmak kaçınılmazdır. Bundan dolayı, Stochastic Gradient Descent algoritmasında tüm örneklerin maliyet fonksiyonun gradyan toplamı yerine, her güncelleme için yalnızca bir örneğin maliyet fonksiyonun gradyanı vardır.

$$\eta t = \eta^{i}{}_{\min} + \frac{1}{2}(\eta^{i}{}_{\max} - \eta^{i}_{\min})(1+\cos(\frac{T_{cur}}{T_i}\pi)) \quad \text{(Denklem 1)}$$

Denklem 1'de kullanılan sembollerin anlamı aşağıdaki gibidir.

$\eta^{i}_{\max}\eta^{i}_{\min}$: i. Adımdaki öğrenme oranı araklılarıdır.

T_{cur} : Yeniden başlatmanın ardından kaç adımın (epoch) geçtiğini gösterir.

T_i : Bir sonraki yeniden başlatmanın kaç adım (epoch) alacağını gösterir.

6.2 Root Mean Square (RMSprop)

Formülde de görüldüğü üzere öğrenme hızı kare gradyanın köküne bölünerek uygulanmaktadır. Bu algoritma[20] yalnızca mevcut mini batch'deki gradyanı tahmin etmektedir.

$$v_t = \rho v_{t-1} + (1-\rho) g_t^{\,2} \quad \text{(Denklem 2)}$$

$$\Delta w_t = -\frac{\eta}{\sqrt{v_t + \in}} * g_t \quad \text{(Denklem 3)}$$

$$w_{t+1} = w_t + \Delta w_t \quad \text{(Denklem 4)}$$

Denklem 2,3 ve 4'de kullanılan sembollerin açıklaması aşağıdaki gibidir.

η : Öğrenme oranı için başlangıç parametresini ifade eder.

g_t : w^j sırasında t zamanındaki gradyan değerini ifade eder.

w_t : Gradyanların karelerinin üstsel ortalamasını ifade eder.

6.3 Adaptive Momentum (AdaM)

Adam[21], uyarlanabilir düşük dereceli moment tahminlerine dayalı, stokastik amaç fonksiyonlarının birinci dereceden gradyan tabanlı optimizasyonu için bir algoritmadır. Yöntemin uygulanması basittir, hesaplama açısından verimlidir, az bellek gerektirir, diyagonal gradyan ölçeklemesinden değişmezdir, veri veya parametreler açısından büyük problemler için uygundur.Yöntem aynı zamanda durağan olmayan hedefler, çok gürültülü ve seyrek eğimli problemler için de uygundur.

Deneysel sonuçlar, Adam'ın pratikte iyi çalıştığını ve diğer stokastik optimizasyon yöntemleriyle olumlu şekilde karşılaştırıldığını göstermektedir. Aşağıdaki grafikte Adam optimizer'ın diğer optimizerlara oranla eğitim maliyet (trainin cost) ve performans bakımından önemli bir farkla daha iyi bir sonuç çıkardığı görülmektedir.

Require : α : Stepsize

Require : $\beta_1, \beta_2 \in [0,1)$: Exponential decay rates for the moment estimates

Require : $f(\theta)$: Stochastic objective function with parameters θ

Require : θ_0 : Inital parameter vector

$m_0 \leftarrow 0$ (Initialize first moment vector)

$v_0 \leftarrow 0$ (Initialize second moment vector)

$t \leftarrow 0$ (Initialize timestep)

while θ_t not converged **do**

$t \leftarrow t+1$

$g_t \leftarrow \nabla_0 f(\theta_{t-1})$ (Get gradients w.r.t stochastic objective at timestep t)

$m_t \leftarrow \beta_1.m_{t-1} + (1-\beta_1).g_t$ (Update biased first moment estimate)

$vt \leftarrow \beta_2.v_{t-1} + (1-\beta_2).g_t^2$ (Update biased second raw moment estimate)

$\hat{m}_t \leftarrow m_t / (1-\beta_1^t)$ (Compute bias-corrected first moment estimate)

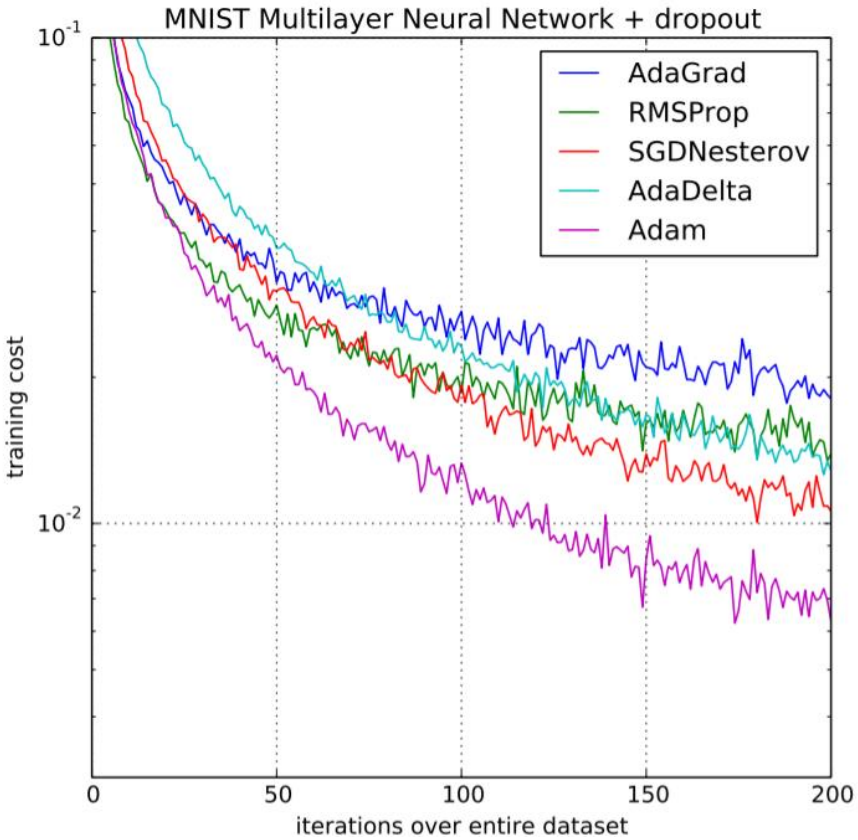
$\hat{v}_t \leftarrow v_t / (1-\beta_2^t)$ (Compute bias-corrected second raw moment estimate)

$\theta_t \leftarrow \theta_{t-1} - \alpha.\hat{m}_t(\sqrt{\hat{v}_t} + \varepsilon)$ (Update parameters)

end while

return θ_t (Resulting parameters)

Tablo 1. AdaM Pseudo Code



Şekil 6. AdaM Karşılaştırılması

Training cost : Model eğitim maaliyeti,

İterations over entire dataset : Veri seti için eğitim tekrarlama sayısı

7. YoloV3

2016 yılında Redmon, Divvala, Girschick ve Farhadi 'You Only Look Once : Unified, Real-Time Object Detection isimli bir makale[22] yayımladı. Bu makale, nesne algılamaya yeni bir yaklaşım getirdi ; özellik çıkarma ve nesne lokalizasyonu tek bir monolitik blokta birleştirildi. Yolo isimli bu tek aşamalı mimari çok hızlı bir sonuç çıkarma süresi sağlamaktadır. YoloV3,

yine Redmon ve arkadaşları tarafından gerçek zamanlı nesne tespiti için geliştirilmiştir. YoloV3 ağı doğru ve hızlı sonuç üretmekte oldukça başarılıdır. Geleneksel bilgisayarlı görü algoritmalarında ölçek ve konumu farklı nesneler kayan pencereme yaklaşımı ile bulunmaktadır ancak bu maliyetli bir işlemdir ayrıca nesnelerin en-boy oranının sabit olduğu düşünülmektedir. YoloV3 algoritması bu yöntemden farklı olarak tüm görüntü ağ üzerinden yalnızca bir kez iletilerek nesne tespiti yapmaktadır. Bundan dolayı geleneksel yöntemlerden daha hızlıdır.

YoloV3 algoritması resimler üzerinde tespit ettiği nesnelerin veya sayıların etrafını bounding box ile çerçeveler. YoloV3 algoritmasına girdi olarak verilen görüntüyü NxN'lik kutucuklara böler. Bu kutucuklar 17×17,9×9,5×5… olabilir. Bütün kutucukların güven skoru vardır ve bu kutucuklar nesnenin yani tespit edilecek şeyin kendi içinde olduğunu düşünür bu yüzden ilk başta birden çok kutucuk bulunabilir. Bu problemi çözmek için Non-Maximum Suppression algoritmasını kullanır. Kısaca Non-Maximum Suppression algoritması görüntü üzerinde tespit edilecek nesne için yaratılan kutucuklardan skor değeri en yüksek olanı ekrana çizer o kutucuğu tespit olarak alır.

Şekil 7.YoloV3 Detection

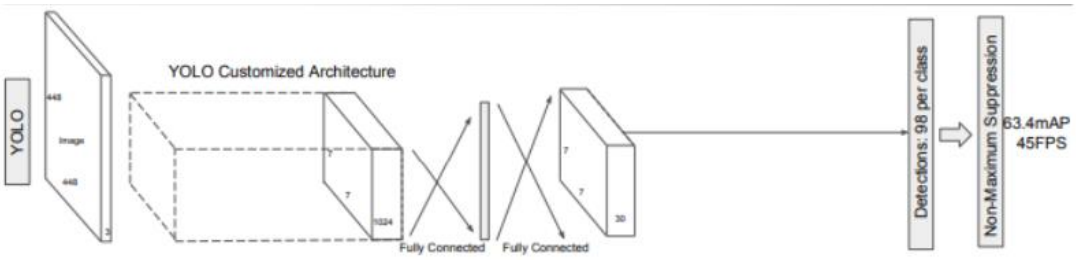
7.1 Non-Maximum Suppression(NMS) Algoritması

Non-Maximum Suppression algoritması[23], birçok bilgisayarla görme görüntü işleme ve nesne tespitinde kullanılan bir algoritmadır. Birbiriyle örtüşen birçok nesne arasından bir nesneyi seçmeye yarar. Skor kriterleri belirli sonuçlara ulaşmak için kullanılabilir. En çok kullanılan ölçüt, örtüşme ölçüsüdür. Çoğu nesne tespiti yapan algoritmalar bir tür pencereleme yapar. Binlerce pencere yani kutucuk doğrudan görüntünün veya resmin bir özelliği üzerinde oluşturulur. Bu kutucuklardan yalnızca biri nesne içerdiğini varsayar ve her sınıf için skor elde etmek için bir sınıflandırıcı kullanır. Bu kutucukların skor değerine göre en iyisini seçmek için Non-Maximum Suppression algoritması kullanılır.

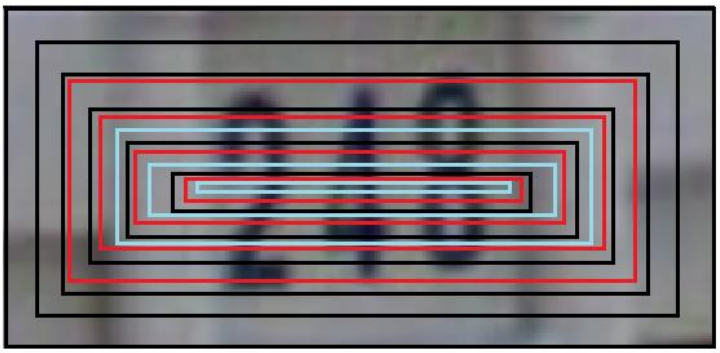
NMS tensorflow, pytorch ve OpenCV gibi kütüphaneleri kullanır.

procedure $NMS(B,c)$
$B_{nms} \leftarrow \phi$
for $b_i \in B$ **do**
$discard \leftarrow False$
for $b_j \in B$ **do**
if $same(b_i,b_j) > \lambda_{nms}$ **then**
if $score(c,b_j) > score(c,b_i)$ **then**
$discard \leftarrow True$
if not discard **then**
$B_{nms} \leftarrow B_{nms} \bigcup b_i$
return B_{nms}

Tablo 2. Non-Maximum Suppression Algoritması



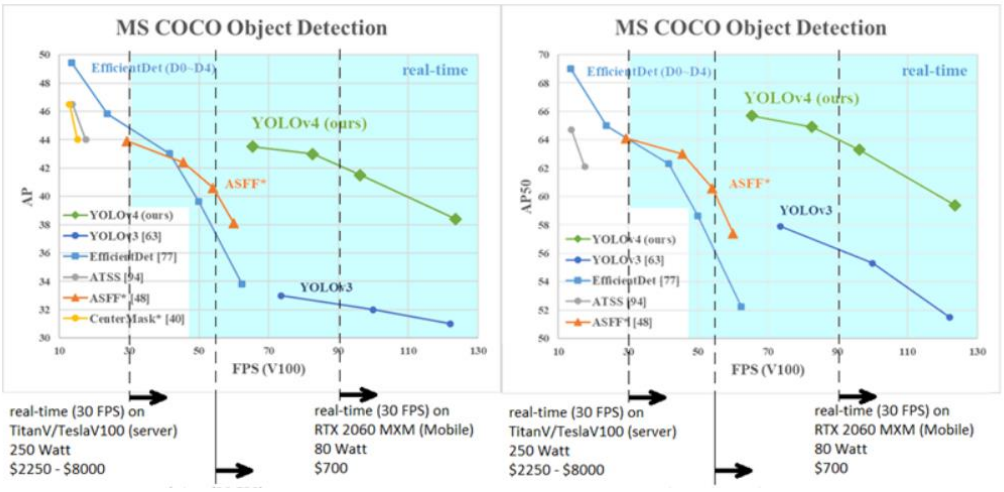
Şekil 8. Yolo NMS Kullanımı



Şekil 9. Yolo Nesne Tespiti

Yukarıdaki görselde YoloV3 ağına verilen görüntünün içerisindeki nesneler, çıkışta bu görüntüde bulunan nesnelerin hangi sınıfa ait olduğu ve konumları ile birlikte verilmiştir.

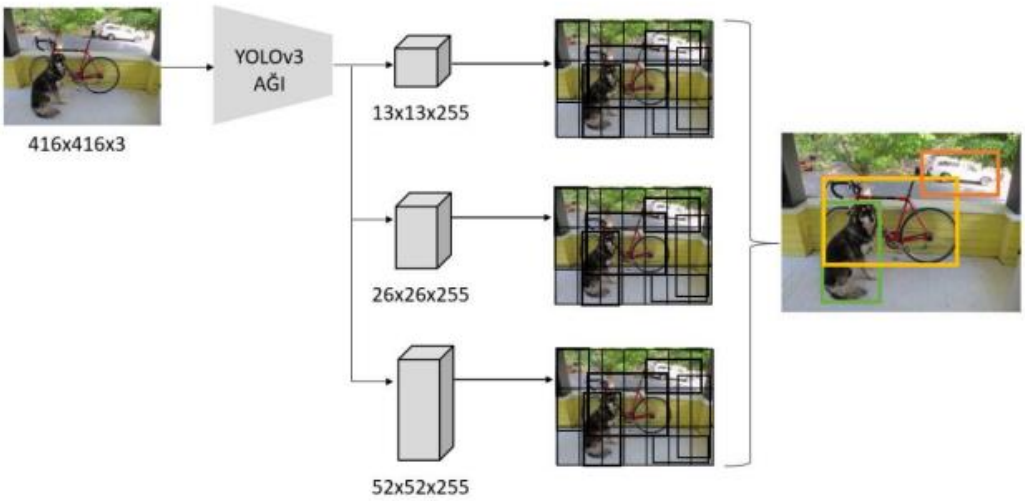
Yolo ve diğer bazı algoritmaların COCO dataseti için detection işlemi performansları Şekil 10'daki gibidir.



Şekil 10. Coco Dataseti Üzerinde Modellerin Karşılaştırılaması

7.2 YoloV3 Ağının Girdisi

YoloV3 ağı girdi[24] olarak bir görüntü almaktadır. Bu görüntünün boyutları 32 sayısının katı olarak şekilde belirlenir. Görüntülerde derinlik ise R, G, B kanallarından dolayı 3 olmalıdır.

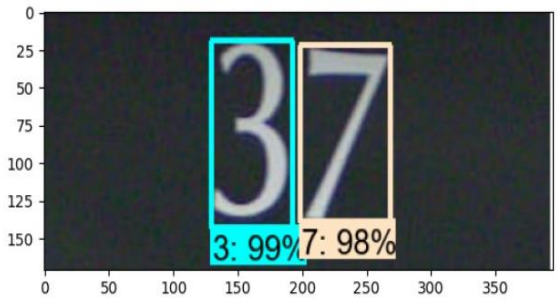


Şekil 11. Yolo Mimarisi

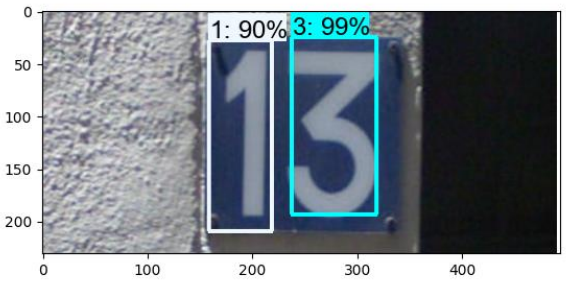
7.3 YoloV3 Ağının Çıktısı

YoloV3 ağının çıktısında aşağıdaki sekilde olduğu gibi çıktı 3 farklı ölçektedir. Bunların ilki 13x13x255 olan birinci ölçek büyük boyutlu nesneleri; 26x16x255 olan ikinci ölçek orta boyutlu nesneleri; 52x52x255 üçüncü ölçek küçük boyutlu nesneleri algılamak için kullanılmaktadır. Daha sonra bu üç ölçek belirli işlemlerden geçirilir ve çıkış elde edilir.

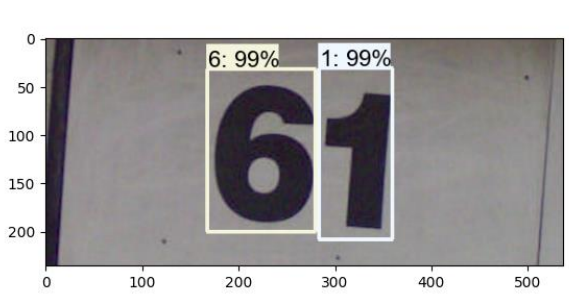
Bu projede detection yani tespit işlemi için YoloV3 kullanılmıştır. NMS için de Tensorflow, Argparse, OpenCV ve Matplotlib kütüphaneleri kullanılmıştır.



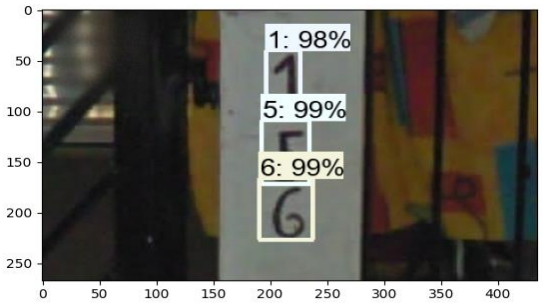
Şekil 12



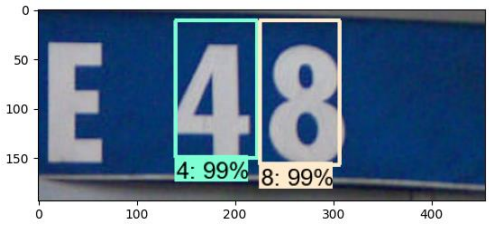
Şekil 13



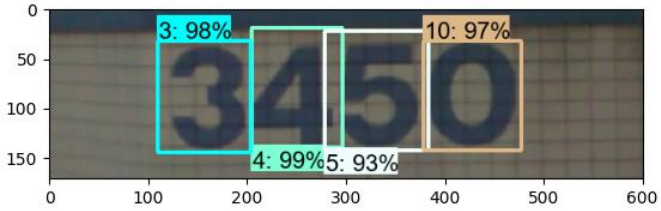
Şekil 14



Şekil 15



Şekil 16

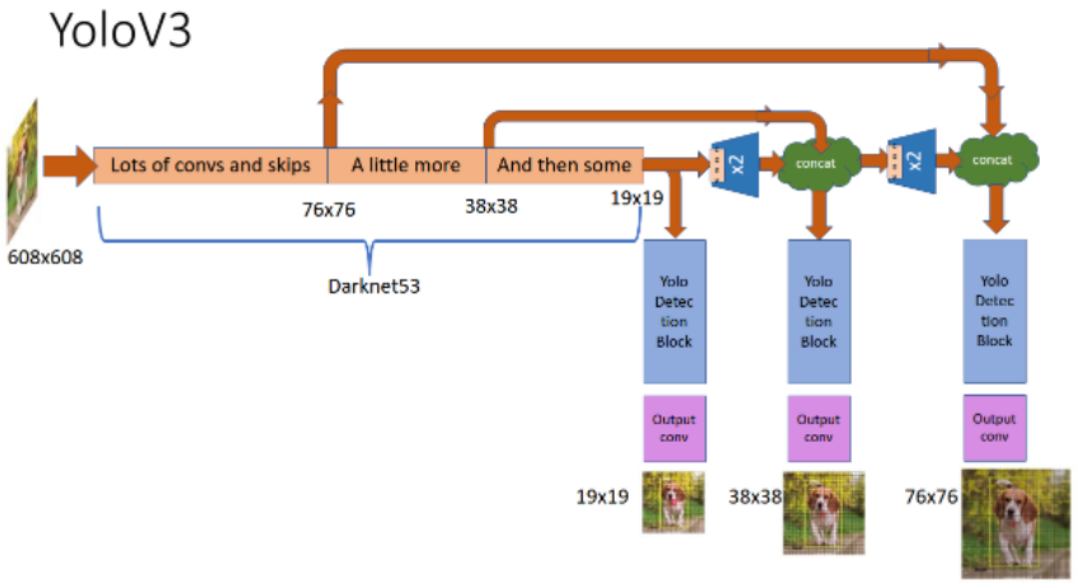


Şekil 17

Şekil 12, 13, 14, 15, 16 ve 17'de sayı tanıma ve kutu çiziminin örnekleri görülmektedir.

7.4 YoloV3 Darknet Modeli

YoloV2 sınıflandırma için Darknet-19 u kullanırken, YoloV3 artık Darknet-53 modelini kullanıyor. Bu projede sınıflandırma işlemi YoloV3 kullanıldığı için Darknet-53 modelini kullanılmıştır. Darknet-53, Yolo modelini geliştiren Joseph Redmon ve arkadaşları[25] tarafından geliştirilmiştir.



Şekil 18. YoloV3 Darknet Mimarisi

Darknet-53, önceki 19 versiyonu yerine 53 tane evrişimsel katmanına sahiptir ve detection görevi için bunun üzerine 53 katman daha eklenir sonuç olarak 106 katmanlı tam evrişimsel bir mimari oluşturur. Bu özellik de ona Darknet-19 dan daha güçlü ve rakip modellerden daha verimli kılar.

Backbone	**Top-1**	**Top-5**	**OPS**	**BFLOP/s**	**FPS**
Darknet-19	74.1	91.8	7.29	1246	171
ResNet-101	77.1	93.7	19.7	1039	53
ResNet-152	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

Tablo 3. Sınıflandırma algoritmalarının karşılaştırılması

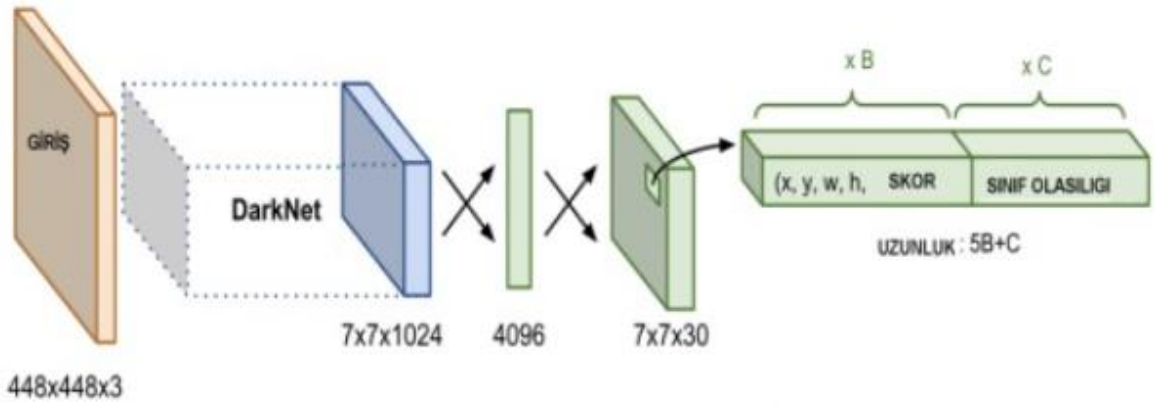
Tablo 3'de karşılaştırılan parametreler aşağıda sırayla verilmiştir :

- Top-1 : Modelin ilk eğitimindeki f1 score değeridir.
- Top-5 : Modelin en iyi f1 score değeridir.
- OPS : Modelde kullanılan train veri seti sayısısıdır.
- BFLOP/s : Saniyede işaretlenen piksel değeridir.

- FPS : Saniyedeki kare hızıdır.

Darknet algoritmasının ResNet modeline göre yaklaşık 1,5 kat daha hızlı olduğundan bu tezde Darknet modeli kullanılmıştır.

Ayrıca, modelin yeniden eğitilmesine gerek duyulmadan sadece modele verilen boyutu değiştirerek doğruluk ve hız arasında kolayca değiş tokuş ve gözlem yapılabilir.



Şekil 19. Darknet-53 Backbone gösterimi

8.KARŞILAŞILAN PROBLEMLER

Model eğitme süreçlerde karşılaşılan en büyük problemlerden biri fiziksel imkanlar/donanımlardır. Bununla ilgili yapay zeka alanında kullanılan sunucular araştırılıp kullanım kolaylığı ve basit arayüzünden dolayı Colab Pro terchih edilmiştir. Ancak Colab Pro'da da sık sık 'Yeniden bağlanma hatası' ile karşılaşılmaktadır. Colab Pro uygulaması 30 dakikadan uzun süre hareket algılamadığında söz konusu hatayı vermektedir. Buna çözüm olarak tarayıcının konsol ekranına aşağıdaki script yazılarak ekrana istenilen periyodlarda tıklama işlemi sağlanmıştır.

```
function KeepClicking(){
 console.log("Clicking");
 document.querySelector("colab-toolbar-button#connect").click()
}setInterval(KeepClicking,60000)
```

9.SONUÇLAR

Model Street View House Numbers(SVHN) isimli veri setinin içerisinden alınan 10000 görsel ile 40 epoch boyunca sırasıyla;AdaM, SGD, RMSProp optimizerlerini kullanarak eğitilmiştir. Deneysel sonuçlar gözlemlendiğinde en başarılı sonucun AdaM optimizer olduğu görülmektedir.

Yolov3	Epoch Sayısı	Train Data	Test Data	F1-Score	Recall	Precision
AdaM	40	10.000	1.500	0.79	0.79	0.79
SGD	40	10.000	1.500	0.45	0.79	0.31
RMSProp	40	10.000	1.500	0.50	0.83	0.35

KAYNAKLAR

[1] Alwzwazy, Haider A., et al. "Handwritten digit recognition using convolutional neural networks." International Journal of Innovative Research in Computer and Communication Engineering 4.2 (2016): 1101-110

[2] Nagai A. On the Improvement of Recognizing Single-Line Strings of Japanese Historical Cursive; Proceedings of the 2019 International Conference on Document Analysis and Recognition, ICDAR 2019; Sydney, Australia. 20–25 September 2019; pp. 621–628

[3] Kim, Jin-Ho, and Duck-Soo Noh. "Vehicle License Plate Recognition System By Edge-based Segment Image Generation." *The Journal of the Korea Contents Association* 12.3 (2012): 9-16.

[4] Yann Lecun. "The Mnist Database of handwritten digits". Erişim tarihi: 30.05.2022. http://yann.lecun.com/exdb/mnist/

[5] Asif, Muhammad, et al. "Long Multi-digit Number Recognition from Images Empowered by Deep Convolutional Neural Networks." *The Computer Journal* (2021).

[6] Salvador España-Boquera, Maria J. C. B., Jorge G. M. and Francisco Z. M., "Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 33, No. 4, 2011.

[7] Reena Bajaj, Lipika Dey, and S. Chaudhury, "Devnagari numeral recognition by combining decision of multiple connectionist classifiers", Sadhana, Vol.27, part. 1, pp.-59-72, 2002.

[8] M. Hanmandlu, O.V. Ramana Murthy, "Fuzzy model based recognition of handwritten numerals", pattern recognition, vol.40, pp.1840-1854, 2007.

[9] Mohammed Z. Khedher, Gheith A. Abandah, and Ahmed M. AlKhawaldeh, "Optimizing Feature Selection for Recognizing Handwritten Arabic Characters", proceedings of World Academy of Science Engineering and Technology, vol. 4, February 2005 ISSN 1307-6884

[10] Jarrett, Kevin, et al. "What is the best multi-stage architecture for object recognition?." *2009 IEEE 12th international conference on computer vision*. IEEE, 2009.

[11] Ranzato, Marc'Aurelio, et al. "Unsupervised learning of invariant feature hierarchies with applications to object recognition." *2007 IEEE conference on computer vision and pattern recognition*. IEEE, 2007.

[12] Yang, Xiangxing, et al. "An In-Memory-Computing Charge-Domain Ternary CNN Classifier." *2021 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2021.

[13] Phu, Huynh Vinh, et al. "Design and Implementation of Configurable Convolutional Neural Network on FPGA." 2019 6th NAFOSTED Conference on Information and Computer Science (NICS). IEEE, 2019.

[14] Shovon, M. M. I., M. Kamruzzaman, and M. K. Kundu. "Recognition of Handwritten Bangla Number Using Multi Layer Convolutional Neural Network." *2020 IEEE Region 10 Symposium (TENSYMP)*. IEEE, 2020.

[15] Netzer, Yuval, et al. "Reading digits in natural images with unsupervised feature learning." (2011).

[16] Rossi, Alberto, et al. "Embedding of FRPN in CNN architecture." *arXiv preprint arXiv:2001.05851* (2019).

[17] Nvidia Developer. "Jetson Nano Developer Kit". Erişim tarihi: 30.05.2022. https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[18] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng Reading Digits in Natural Images with Unsupervised Feature Learning *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.

[19] B. Savas, Y. Becerikli, "Real Time Driver Fatigue Detection System Based on Multi Task ConNN", IEEE Access, 8, p12491-p12498, 2020.

[20] Y. LeCun, Y. Bengio, G. Hilton, "Deep learning," Nature, 521, p436-p444, 2015.

[21] Hengyue Liu, Bir Bhanu; Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2019, pp. 0-0

[22] Goodfellow, Ian J., et al. "Multi-digit number recognition from street view imagery using deep convolutional neural networks." *arXiv preprint arXiv:1312.6082* (2013).

[23] Bottou, Léon. "Stochastic gradient descent tricks." *Neural networks: Tricks of the trade*. Springer, Berlin, Heidelberg, 2012. 421-436.

[24] Reddy, R. Vijava Kumar, B. Srinivasa Rao, and K. Prudvi Raju. "Handwritten Hindi digits recognition using convolutional neural network with RMSprop optimization." *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 2018.

[25] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

[26] Zaghari, Nayereh, et al. "The improvement in obstacle detection in autonomous vehicles using YOLO non-maximum suppression fuzzy algorithm." *The Journal of Supercomputing* 77.11 (2021): 13421-13446.

[27] ÖZEL, Muhammed Abdullah, Selim Sefa BAYSAL, and Mustafa ŞAHİN. "Derin Öğrenme Algoritması (YOLO) ile Dinamik Test Süresince Süspansiyon Parçalarında Çatlak Tespiti." *Avrupa Bilim ve Teknoloji Dergisi* 26: 1-5.

[28] Kılıç, Büşranur. *Panorama ile üretilen plevral efüzyon sitopatoloji görüntüleri üzerinde yolov3 ile otomatik çekirdek algılama*. Diss. Karadeniz Teknik Üniversitesi/Fen Bilimleri Enstitüsü/Bilgisayar Mühendisliği Anabilim Dalı, 2020.

[29] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.