TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GÖMÜLÜ SİSTEM ÜZERİNDE EVRİŞİMSEL SİNİR AĞLARI İLE NESNE SINIFLANDIRILMASI

YÜKSEK LİSANS TEZİ

Rasim Caner ÇALIK

Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Doç. Dr. M. Fatih DEMİRCİ

Temmuz 2018

Fen Bilimleri Enstitüsü Onayı

.....
Prof. Dr. Osman EROĞUL
Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

.....
Prof. Dr. Oğuz ERGİN
Anabilimdalı Başkanı

TOBB ETÜ, Fen Bilimleri Enstitüsü'nün 151111041 numaralı Yüksek Lisans öğrencisi **Rasim Caner ÇALIK**'nin ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"GÖMÜLÜ SİSTEM ÜZERİNDE EVRİŞİMSEL SİNİR AĞLARI İLE NESNE SINIFLANDIRILMASI"** başlıklı tezi 16.07.2018 tarihinde aşağıda imzaları olan jüri tarafından kabul edilmiştir.

Tez Danışmanı: **Doç. Dr. M. Fatih DEMİRCİ** .....
TOBB Ekonomi ve Teknoloji Üniversitesi

Jüri Üyeleri: **Doç. Dr. Hacer KARACAN (Başkan)** .....
Gazi Üniversitesi

Dr. Öğr. Üyesi Ahmet Murat ÖZBAYOĞLU .....
TOBB Ekonomi ve Teknoloji Üniversitesi

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, alıntı yapılan kaynaklara eksiksiz atıf yapıldığını, referansların tam olarak belirtildiğini ve ayrıca bu tezin TOBB ETÜ Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırlandığını bildiririm.

Rasim Caner ÇALIK

ÖZET

Yüksek Lisans Tezi

GÖMÜLÜ SİSTEM ÜZERİNDE EVRİŞİMSEL SİNİR AĞLARI İLE NESNE SINIFLANDIRILMASI

Rasim Caner ÇALIK

TOBB Ekonomi ve Teknoloji Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Doç. Dr. M. Fatih DEMİRCİ

Tarih: Temmuz 2018

Derin sinir ağları görüntü sınıflandırma üzerinde başarılı bir şekilde uygulanmaktadır. Görüntüde tespit edilen nesnelerin ne olduğunu anlanması büyük bir ilgi görmektedir. Derin sinir ağlarının gömülü sistemler üzerinde kullanılması oldukça fazla hafıza ve güçlü işlemci teknolojileri gerektirmektedir. Bu tezin amacı gömülü bir sistem üzerinde görüntü sınıflandırma işleminin ne kadar başarılı bir şekilde yapıldığını gözlemlemektir. Bu çalışmada sadece 3 GB hafıza kullanarak geliştirilen mimari ile %85.9 oranında doğruluk elde edilmiştir. Önerilen mimari 4 saat içerisinde eğitilerek, 0.7 ms içersinde sınıflandırma sonucunu üretmektedir.

Anahtar Kelimeler: Evrişimsel sinir ağları, Derin sinir ağları, Makine öğrenme, Görüntü sınıflandırma.

ABSTRACT

Master of Science

OBJECT CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORK IN EMBEDDED SYSTEMS

Rasim Caner ÇALIK

TOBB University of Economics and Technology
Institute of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. M. Fatih DEMİRCİ

Date: July 2018

Deep Neural Network is successfully applied for image classification problems. The understanding of what object is detected in the image is great interested. The prupose of this article is that image classification problem and tracking classified object could be applied in real time systems. The aim of this thesis is image classification process is observing how successful in an embedded system. In this work, proposed architecture has only 3 GB memory with accuracy %85.9. It trains in 4 hour,and outputs the classification result in 0.7 ms.

Keywords: Convolutional neural network, Deep neural network, Machine learning, Image classification.

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren hocam Doç.Dr. M.Fatih DEMİRCİ, kıymetli tecrübelerinden faydalandığım TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyelerine ve destekleriyle her zaman yanımda olan aileme ve arkadaşlarıma çok teşekkür ederim. Ayrıca yüksek lisansımı burslu olarak bitirmemi sağlayan TOBB Ekonomi ve Teknoloji Üniversitesine teşekkürlerimi sunarım.

İÇİNDEKİLER

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

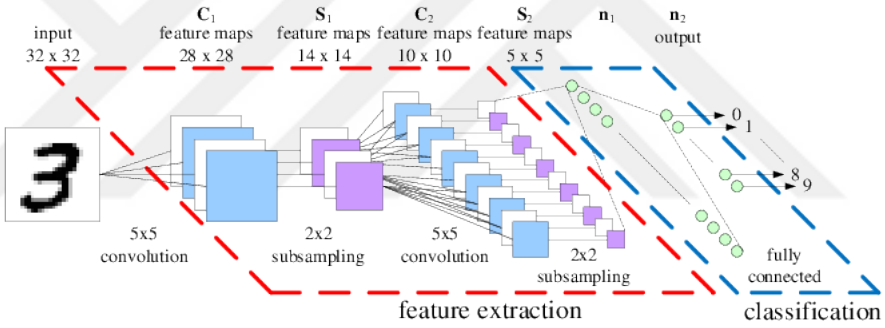
KISALTMALAR

CNN : Evrişimsel Sinir Ağları
PCA : Temel Bileşen Analizi
MLP : Çok Layerlı Perseptron
KNN : K Komuşulu Sinir Ağları
SVM : Destek Vektör Makineleri
DT : Karar Ağaçları
fps : Saniyedeki Kare Sayısı
IV : Illumination Variation
SV : Scale Variation
OCC : Occlusion
DEF : Deformation
MB : Motion Blur
FM : Fast Motion
KCF : Kernelized Correlation Filter
VDT : Visual Decomposition Tracker
GB : Giga Byte
LSTM : Uzun ve Kısa Zamanlı Hafıza

RESİM LİSTESİ

1.GİRİŞ

Son zamanlarda derin öğrenme yöntemlerinde yaşanan gelişmeler sonucunda evrişimsel sinir ağlarının(CNN) performansı oldukça artırılmıştır. CNN mimarileri oldukça nöron adı verilen ağırlık ve bias parametreleri sayesinde öğrenebilen yapılardır. Her nöron diğer nöronlardan gelen inputları toplayarak kendi inputu olarak kullanır ve opsiyonel olarak doğrusal olmayan yapılar kullanılarak mimarinin doğrusal olmayan sistemlerin modellenmesinde daha başarılı bir şekilde öğrenmesini sağlamaktadır. Öznitelik çıkartma işlemi nesne tanıma problemlerinin en önemli parçasıdır. CNN mimarilerinin en önemli avantajı mimarinin kendi özniteliklerini çıkarabilmesidir. CNN mimarilerinin kullanıldığı ilk çalışmalardan biri MNIST(el yazısıyla sayı tanıma veriseti) veriseti ile sayıların görüntülerden tanınmasıdır. Bu çalışmanın örnek bir mimarisi Şekil 1.1'de verilmiştir.



Şekil 1.1: MNIST database için CNN mimari örneği - [49].

CNN mimarilerinin en büyük problemi kullandığı hafıza miktarıdır. CNN uygulamalarının başarım oranı oldukça yüksek(Çoğu uygulamalarda yaklaşık olarak %90 oranında başarım elde edilebilmektedir.) olmasına rağmen, uygulamaların gömülü sistemlere uyarlanması için şu anki teknolojilerde mimarileri destekleyebilecek sistemler bulunmamaktadır[48]. Son zamanlarda üretilen NVIDIA firmasının TX2 modülünün 8GB'lık hafıza birimi bulunmaktadır. Bu sebeple CNN uygulamalarının gömülü bir sistemde çalıştırılabilmesi için maksimum 8 GB'lık bir mimari önerilmelidir. GPU için ayrılan hafıza birimini aynı zamanda modüllerin CPU birimleri de kullandığı için 8 GB'lık hafızanın tamamı mimari için ayrılamamaktadır. Kullanılan CPU'ların işlemci performansı da masaüstü bilgisayarlar performansında olmadığı için, eğitim süresi ve sınıflandırma sonucunu üretme süresi uzamaktadır. CNN mimarileri sınıflandırma sonucunu üretirken sadece matris çarpımı işlemleri yapmaktadır. Son zamanlarda başarım oranını oldukça artıran ResNet[55], VGG-16[33] gibi ünlü mimariler türetilmiştir. Bu mimarilerde, önceki katmanların çıkışlarının sonraki katmanlara etkisinin hesaplanması gerektiği için kullanılması gereken hafıza miktarı artmaktadır. Görüntü üzerinde her piksel için filtreleme işlemi(matris

çarpımı) ve her katman için küçültülmüş resimler üzerinde de aynı filtreleme işlemi yapıldığı için ihtiyaç duyduğu hafıza miktarı artmaktadır. Aynı bakış açısıyla her katman, önceki katmanın çıkışına göre matris çarpımı gerektirdiği için eğitim ve sonuç üretme süresi oldukça uzamaktadır.
Nesne takip etmek gerçek zamanlı(real time) olarak çözülmesi gereken bir problemdir. Birçok askeri projede kullanılan nesne takibi ve nesne tanıma işlemleri gerçek zamanlı olarak işlemci kartları üzerinde çalıştırılmaktadır. Bu tez kapsamında güçlü bir mimari olan CNN'ler kullanılarak gerçek zamanlı bir sistem üzerinde nesne takibi ve takip edilecek nesnenin tanınma uygulaması gerçekleştirmek amaçlanmaktadır.
Bu çalışma, tezde yapılması amaçlanan çalışmaları özetleyerek devam etmektedir. İkinci kısımda bu tez kapsamında yapılan çalışmalarla ilgisi bulunan çalışmalar açıklanmıştır. Üçüncü bölümde nesne tanıma probleminin tanımı ve çalışma sonuçları verilmiştir. Dördüncü bölümde nesne takip etme problemi tanımlanıp tez kapsamında yapılan çalışmalar ve sonuçları sırasıyla açıklanmıştır. Beşinci ve son bölümde tezde yapılan çalışmaları özetleyerek tezin devamı için önerilen yöntemler açıklanmıştır.

1.1.Tezin Amacı

CNN uygulamaları, hafıza gerektiren sistemler olup eğitilmesi sırasında oldukça hafıza tüketmektedir. Resim üzerinde her piksel değerleri için filtreleme işlemi uygulanmaktadır. Her katmanda resimler örnekleme katmanından sonra küçülmeye başlayacaktır. Her katmanda bu filtreleme işlemi tekrar tekrar yapılacağı için gereken hafıza miktarı artmaktadır. Ayrıca katman sayısının artırılması sistemin gerçek zamanlı olarak çalışmasını engellemektedir. Artan katman sayısı eğitilme ve sınıflandırma sonucunun üretilme süresini artırmaktadır. Bu tez çalışmasında, CNN mimarilerinin gerçek zamanlı sistemlerde sınırlı bir hafıza ile hedef takip ve görüntü sınıflandırma işlemlerini başarılı bir şekilde uygulanması amaçlanmaktadır.

2.LİTERATÜR ARAŞTIRMASI

Nesne takip etme problemi bilgisayarla görü alanında araştırılan güncel konulardan birisidir. Literatürde önerilen çalışmalar üç temel adımdan oluşan bir yapı tercih etmektedirler.

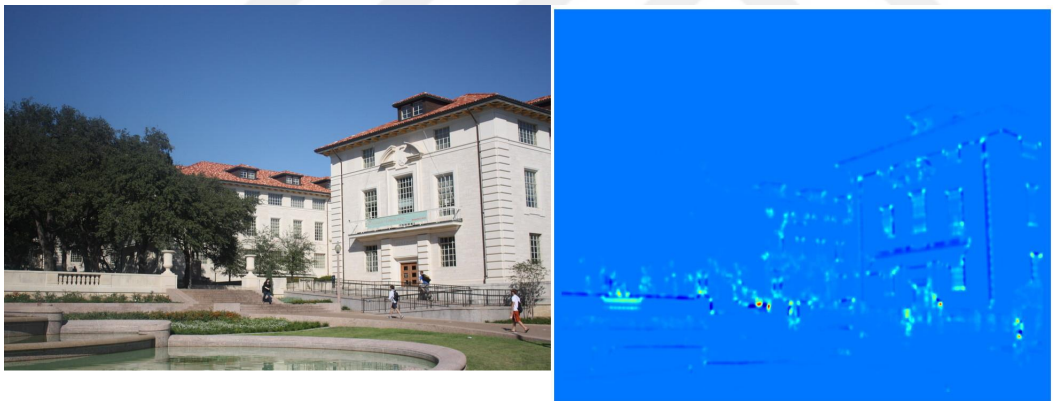
- Nesne Bulma
- Nesne Sınıflandırma
- Nesnenin Takip Edilmesi

Nesnenin görüntü üzerinde yerinin tespit edilmesine nesne bulma(object detection) adı verilmektedir. Bu problemin çözümü için literatürde birçok çalışma bulunmaktadır. Bunların ilki, 'Frame differencing(Kare Farkı)' adı verilen iki video kare görünütüsünün birbirinden çıkarılarak en çok değişen bölgeyi yani hareket eden nesnenin bulunduğu yeri tespit etmektedir[17]. Kare Farkı methodu engelleme(OCC) gibi durumlarda nesne bulmak için yetersiz kalabilmektedir. Bu sebeple görüntü işleme yöntemleri kare farkı işleminden sonra(post processing) kullanılması ile performansı artırılmaktadır[23]. Optik akış(Optical Flow) yöntemiyle her piksel için hız vektörleri hesaplanmaktadır. Bu hız vektörlerinin en etkili olduğu bölgede bir nesne bulunmaktadır[40]. Arka plan çıkarılması(Background Substraction), Kare Farkı yöntemine çok benzeyen bir şekilde her karenin birbirinden çıkarılması yerine arka plan görüntülerinin birbirinden çıkarılması hareket eden nesnenin bulunduğu yerin ortaya çıkarılmasını sağlamaktadır [21].

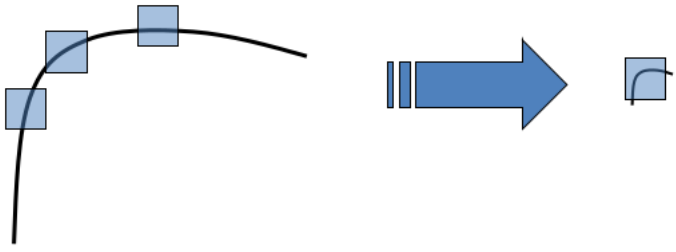
2.1.Klasik Görüntü İşleme Yöntemleriyle Yapılmış Olan Çalışmalar

Nesne sınıflandırma bilgisayarla görü alanında araştırılıp sürekli geliştirilen alanlardan birisidir. Nesne sınıflandırma problemi için piksel tabanlı sınıflandırma yaklaşımı en kolay yaklaşımlardan biridir. Bir başka deyişle piksellerin birbiri ile olan komşuluk ilişkileri kullanılarak resimlerin birbirine benzetilmesiyle görüntüler sınıflandırılır [1]. Maksimum benzerlik tabanlı sınıflandırıcısı daha çok uydu fotoğraflarında kullanılan bir yöntem olmakla beraber, sıradan fotoğraflarda da uygulanabilecek bir yöntemdir [1]. Piksel tabanlı sınıflandırıcıları karşılaştırmak için bir çalışma yapılmıştır. Bu çalışmada Bayes Sınıflarıcısı, Logistic Regresyon, Karar Ağaçları, Nöron Ağları, Hibrid de denilebilen birden fazla sınıflandırıcının bir arada kullanıldığı modeller üzerinde denemeler yapılmıştır[66]. Bu sınıflandırıcıların temel problemi verilen resimlerin ışık değişimine karşı hassas olmasıdır. Işığın geldiği yönünün bile değişmesi resim üzerinde piksel değerlerinin değişmesine sebep olmaktadır. Işığın değişmesi görüntünün histogramını sağa veya sola doğru

kaydırmaya başlayacaktır. Bu sebeple piksel tabanlı karşılaştırma yapıldığında görüntülerin birbirleriyle eşleşmediği görülecektir.
Piksel Tabanlı Sınıflandırıcılar yerine, nesne tabanlı yaklaşımlar geliştirilmiştir. Bu çalışmalarda resimler bölütleme(segmentation) yöntemleriyle parçalara ayrılarak bu parçalardan resimler üzerinde maksimum benzerlik olan bölgeyi bularak nesne tanıma işlemini gerçekleştirmektedir [9]. Bu sınıflandırıcılar için bölütlemenin gerekenden az veya çok yapılmasına bağlı olarak nesne tanıma performansı etkilenmektedir.
Nesne sınıflandırma problemi için, klasik görüntü işleme yöntemleri de kullanılabilmektedir. Görüntü üzerinde ön işleme yöntemleri kullanılarak öznitelik çıkartılarak bu özniteliklerin eşleştirilmesi ile görüntü sınıflandırma işlemi yapılmaktadır. Görüntü sınıflandırmak için, görüntü üzerindeki nesnelerin özniteliklerinin çıkarılması en önemli aşamasıdır. Nesnelere ait temel öznitelikler kullanılarak hızlı ve etkili nesne tanıma işlemleri yapılabilmektedir [11]. En temel öznitelik bulma yöntemi verilen resimde köşe noktalarını bulmaktır. Harris, intensity(yoğunluk) değişiminin hem yatayda hem dikeyde oldukça fazla olması o pikselin köşe olduğunu söylemektedir. Bu algoritma, görüntünün döndürülmesinden kaynaklanan değişimden etkilenmemektedir. Fakat görüntü boyutunun değişimi bu algoritmanın hata vermesine sebep olacaktır. Bir başka deyişle, görüntünün döndürülmesi sonucunda oluşan resimle döndürülmemiş hali Harris köşe bulucu algoritması sonucu eşleşirken, görüntü boyutunun değiştirilmesi Harris köşe bulucu algoritmasının hata vermesine sebep olabilmektedir. Şekil 2.1'de bu problemin bir örnek görüntüsü verilmiştir.



Resim 2.1: Harris algoritma sonucu - Kristen Grauman sunumu[35]

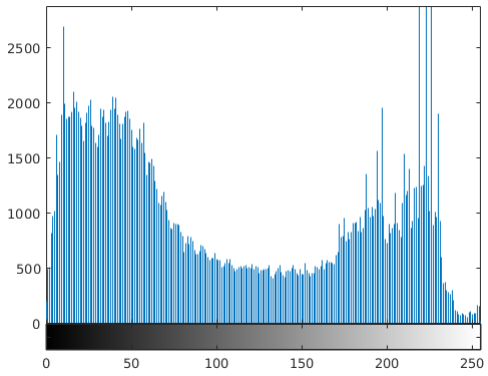


Şekil 2.1: Görüntü boyutunun değişimininin Harris'e etkisi.

Histogram, resimdeki piksel değerlerinin olasılıksal olarak dağılımını ifade etmek-

tedir. Resim 2.2'de verilen görüntünün histogramı Şekil 2.2'daki gibidir.

Resim 2.2: Histogramı hesaplanacak resim



Şekil 2.2: Örnek resmin histogramı.

Bin, verilen resmin piksel değer kümelerine verilen isimdir. Örneğin, piksel değeri 0-10 aralığında olan pikseller bir küme halinde olsun, bu kümenin yani binin histogramı hesaplanırken, o aralıkta olan toplam piksel sayısının resimde bulunan piksel sayısına oranına eşittir.Histogram tabanlı karşılaştırma, iki farklı yöntemle nesne tanıma işlemini gerçekleştirmektedir. Histogram karşılaştırmakta kullanılan ilk yöntem bin kümelerini birbiri ile karşılaştırmaktır. Bu yöntemde en önemli problem, başarımının seçilen bin aralığının uzunluğuna çok bağımlı olmasıdır. Bu sebeple, her bini diğer binlerin birkaçı veya hepsi ile karşılaştırma yöntemi geliştirilmiştir. Bu yöntemde de hangi binleri birbiri ile karşılaştırılacağına karar vermek gerekmektedir. Bunun için histogram uzaklığı adı verilen bir metrik üretilmiştir. Bu metrik değerini hesaplamak için iki farklı algoritma önerilmiştir. Bu algoritmaların biri Minkowski uzunluğu(Denklem 2.1) diğeri de Chi-squared(Denklem 2.2) uzunluğu olarak literatürde tanımlanmıştır.

$$D(h_i,h_j) = (\sum_{m=1}^{K} |h_i(m) - h_j(m)|^p)^{\frac{1}{p}} \tag{2.1}$$

$$\chi^2(h_i,h_j) = \frac{1}{2}\sum_{m=1}^{K} \frac{[h_i(m) - h_j(m)]^2}{h_i(m) + h_j(m)} \tag{2.2}$$

Birbirlerine olan uzaklık değerleri minimum olan resimlerde bulunan nesneler aynıdır olarak yorumlanabilir.

Resim 2.3: Araba tanıma için kullanılan Chi-Sqaured metrik örneği

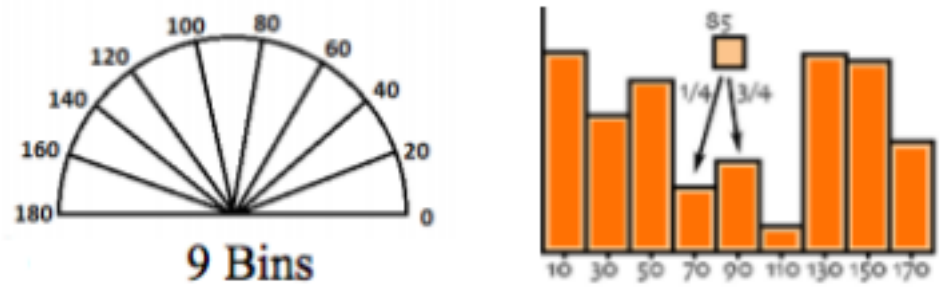
Histogram tabanlı nesne tanıma işlemlerinde çığır açma niteliğinde bir yöntem de HoG(Histogram Of Gradients)'dur. Bu algoritma ilk olarak insan tanıma probleminde kullanılmış olup, sonrasında birçok bilgisayarla görü alanındaki problemlerin çözümünde kullanılmıştır.[45]

Algorithm 1 HoG

1: **procedure** HoG ▷ HoG Öznitelik Çıkartma
2: Yatay ve Düşey Eğimlerini(Gradient) Hesapla ▷ $[-101]ve[-101]^T$ filtreleri kullanılarak bu eğimler hesaplanabilir.Her piksel için hesaplanacaktır.
3: Eğimlerin(Gradient) yönlerini ve büyüklüklerini hesapla
4: Resimleri 16x16lık bloklar %50 oranında kesiştirerek oluşturulur.
5: 16x16lık bloklar 8x8'lik 4 ayrı hücreye bölünür.
6: Eğim yönleri 9 ayrı bine bölünerek oluşturulur.
7: Histogramlar birleştirilir. ▷ Resim 64x128 ise 105(7x15) blok * 4 hücre * 9 histogram bini = 3780 öznitelik ▷ Yönü tanımlı açıya denk gelmeyen eğimler doğrusal interpolasyon yöntemiyle hesaplanabilir.

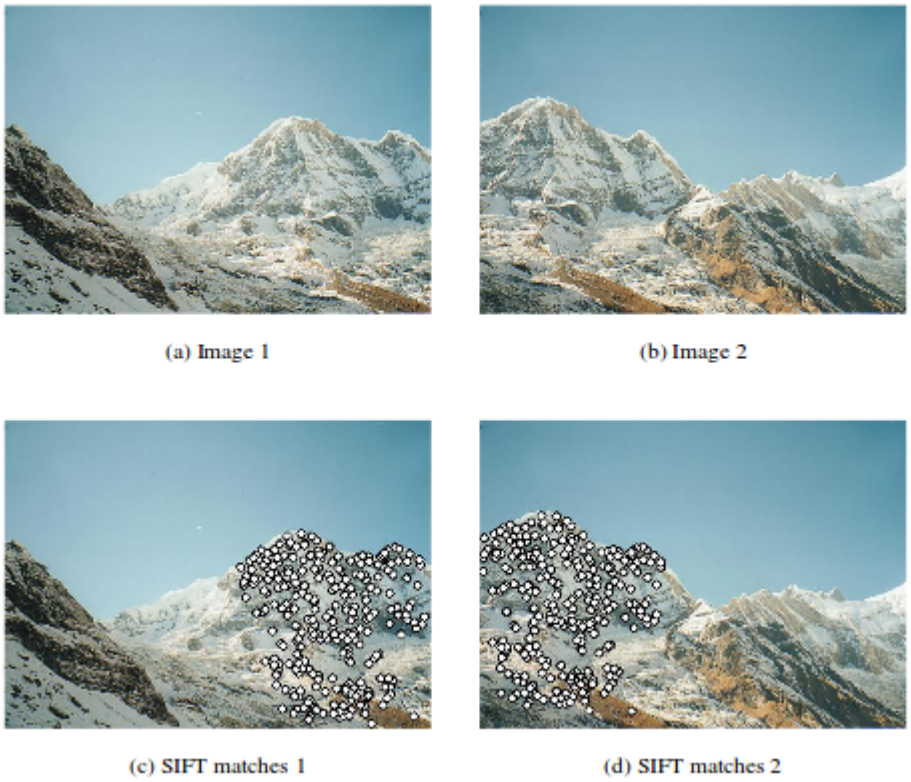
HoG yöntemiyle beraber bu yöntem biraz daha geliştirilerek SIFT adı verilen daha kapsamlı bir algoritma geliştirilmiştir[38]. SIFT, gerçek zamanlı olarak çalıştırılabilir hızlı ve etkin bir algoritma olduğu için birçok varyasyonu türetilmiştir. Büyütme(Scale) ve döndürmeden(Rotation) bağımsız öznitelikler çıkartmaktadır, bu durum görüntü sınıflandırma problemlerinin çözümünü oldukça kolaylaştırmaya başlamıştır.
Nesne sınıflandırma problemi için literatürde güncel çalışmalar ve geçmişte uygulanmış temel oluşturan çalışmalar bulunmaktadır. Literatürde ilk çalışmalar 1972 yılının sonlarında parmak izi resimleri ile doğuştan gelen hastalıklarla arasında bir ilişki olup olmadığının araştırılması ile başlanmıştır [37]. 1973 yılında yapılan bir diğer çalışmada görüntüler 3 ayrı kategoriye ayrılmak istenmektedir. Bu çalışma sonucunda ortalama olarak %84.67 oranında başarım elde edilmiştir [52].

Resim 2.4: HoG örnek resim çıktısı - HoG[45]



Resim 2.5: Solda histogramın tanımlı yönleri, sağda ise interpolasyonın nasıl yapıldığı açıklanmıştır[45]

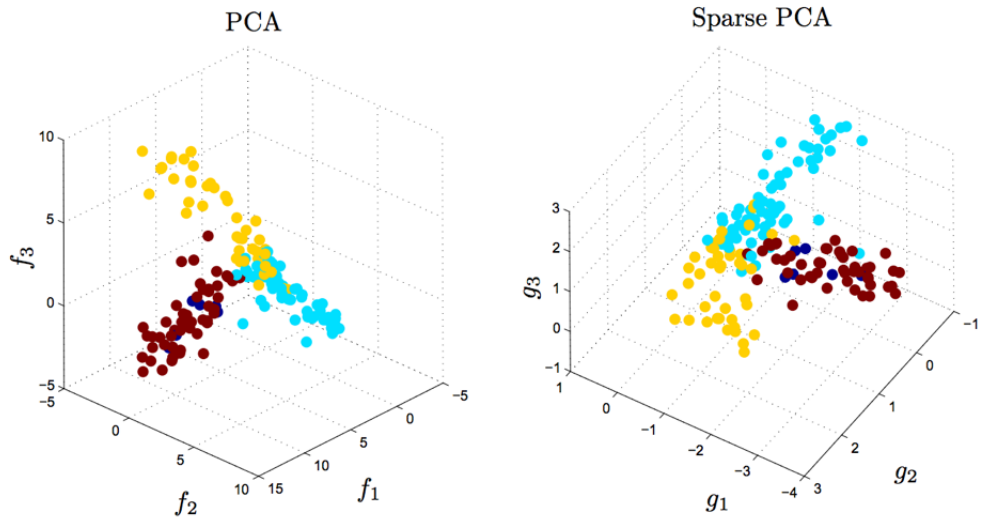
Bölütleme(Segmentation) tekniğinin geliştirilmesiyle insan yüz profillerinin otomatik olarak tanımlanması başarılı bir şekilde gerçekleştirilmeye başlanmıştır [36]. Bu çalışma ile insanların profillerin tanınmasında %96 oranında başarım elde edilmiştir. Bu zamana kadar yapılan çalışmalarda görüntü sınıflandırmak için ideal görüntüler kullanılmıştır. Fakat bu problemin çözümü gerçek zamanda çok daha zorlaşmaktadır. Görüntü sınıflandırma probleminde karşılaşılması muhtemel olan birçok problem bulunmaktadır. Görüntünün parlaklık ve kontrast değerlerinin değiştirilmesi sonucunda farklı görüntüler elde edildiğinde(Aydınlatma-Illumination), görüntünün sınıflandırmasında karşılaşılabilecek en temel problemlerden birisidir. Bir diğer problemde resimde tespit edilmek istenen nesnenin başka bir nesne tarafından engellenmesidir(Occlusion). Engelleme(Occlusion) ve Aydınlatma değişiminden kaynaklanan değişimlere rağmen görüntülerin doğru şekilde sınıflandırılması için yakın zamanlarda yeni bir çalışma yapılmıştır. Bu çalışmada ise özniteliklerin çok iyi olmadığı durumlarda dahi yüz tanıma işlemini başarılı bir şekilde yapabildiğini göstermektedir [70]. Bazı uygulamalarda iki veya daha fazla resim karşılaştırılırken resmin sahip olduğu tüm pikseller teker teker karşılaştırılarak nesne tanıma işlemi yapılmaktadır [2]. Bu işlem oldukça zahmetli ve gerçek zamanlı sistemler için uygulamak imkansızdır. Panoramik resimlerin birleştirilmesi için SIFT özniteliklerini kullanarak olasılıksal bir model yardımıyla iki resmin aynı olan bölgelerini bulan algoritmalar geliştirilmiştir [10]. Bu tür uygulamalara genelikle mozaikleme adı da verilir. Bu tür uygulamalarda yerel öznitelikler kullanılmaktır bu sebeple özniteliklerin doğru seçilmesi çok önemlidir.



Resim 2.6: Mozaik girdi resimleri ve bulunan öznitelikler - Kristen Grauman sunumu [35]

Resim 2.7: Panoramik birleştirilmiş resim - Kristen Grauman sunumu[35]

Gözetimsiz öğrenme kullanarak yani eğitim seti olmadan sistemin yeni gelen resimlerle sürekli kendini güncellemesidir. Gözetimsiz öğrenme yaklaşımı ile nesnelerin modellerinin çıkarılması üzerine bir çalışma yapılmıştır [64]. Bu çalışmada modelleri bölütleme yöntemi gibi parçalara ayırmaktadır. Her parçanın görünüşü, ölçeği(scale) ve engel(occlusion) olup olmadığı bilgileri bulunmaktadır. Model, Gaussian olasılık dağılımını kullanarak ölçek, görünüş ve engel durumunu modellemektedir. Öğrenme aşamasında ilk olarak ölçek değerini tahmin etmeye çalışmaktadır. Parametreler Bayes sınıflandırıcısına göre güncellenmektedir.
Temel bileşen analizi(PCA), ortogonaliteyi kullanarak istatistiksel olarak verileri doğrusal bir şekilde temel bileşenlere bölmeyi sağlayan bir yöntemdir. Bu yöntemle doğrusal olarak ayrılamayan veriler, veri uzayı değiştirilerek doğrusal olarak ayrıştırılabilir hale getirilebilmektedir.



Şekil 2.3: Temel bileşen analizi örnek veriseti - Alex Williams[4].

PCA algoritması ve SIFT algoritmasını birleştirerek yerel bir bölgede nesne tespit

uygulaması gerçekleştirilmiştir[72]. SIFT[38] algoritması %43 oranında başarım gösterirken PCA'in etkisi ile beraber %68 oranında başarıma kadar yükseltilmiştir. Çizelge 2.1'de SIFT ve PCA-SIFT algoritmalarının çalışma süreleri verilmiştir. Bu tablolardan PCA-SIFT algoritmasının SIFT'e göre çalışma zamanında ciddi anlamda bir geliştirme göstermesi ve başarım oranının artması bu algoritmanın başarısını ortaya koymaktadır.

Çizelge 2.1: PCA-SIFT çalışma süresi bazlı karşılaştırma - Yan Ke[72].

Metod	**time(sec)**	σ
Localization and I/O	2.63	0.09
SIFT Representation	1.59	0.06
PCA-SIFT Representation	1.64	0.04
SIFT Matching	2.20	0.03
PCA-SIFT Matching	0.58	0.05

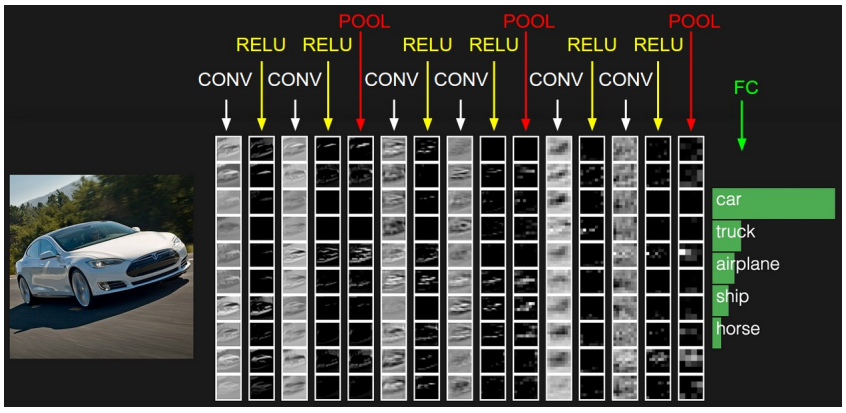
Ölçekten bağımsız öznitelik çıkaran bir başka algoritma daha geliştirilmiştir. Bu algoritma, geometrik olarak daha tutarlı ölçeklerde tekrar edilebilir öznitelikler üretilmiştir. Bu öznitelikler, ölçek piramitleri üzerinden Harris tanıma algoritmasına göre oluşturulmaktadır. Bu çıkarılan özniteliklerden belli bir alan içerisinde maksimum değeri verenler seçilerek bazı öznitelikler atılmıştır(Bu yönteme non-maximum suppression adı verilmektedir). Bu öznitelikler kullanılarak ölçekten, döndürme ve ötelemeden bağımsız olarak nesne tanıma işlemi başarı ile gerçekleştirilmiştir[30]. Nesne tanıma algoritmalarına çok benzeyen bir yöntemle yüz tanıma algoritmaları geliştirilmiştir. Turk, 512x512'lik gri seviyeli resimler üzerinde yüz tanıma problemine bir çözüm getirmiştir. Bu çalışmada her resmi bir vektör olarak tanımlayarak bilinmeyen bir yüz resmini veritabanındaki bu vektörlerle karşılaştırarak verilen resimdeki yüzün kime ait olduğu tahmin edilmeye çalışılmıştır[39]. Bu çalışmadaki en büyük problem yüz resimi 512x512 olduğu için yaklaşık olarak 250000'lik büyük bir veriyi veritabanına kaydedip bu kaydedilen verilere de kabul edilebilir bir sürede erişmek gerekmektedir. Bu sebeple, 250000lik vektörün uzunluğunun azaltılması öngörülmüştür. Bu çalışmada özniteliklerin en belirginlerini seçmek için verilen resimlerin eigen vektörleri hesaplanmaktadır. Resimlerin tamamını vektör olarak saklamak yerine eigen vektörleri saklanarak 250000'lik vektörler yerine 512 uzunluğundaki vektörlerin saklanması önerilmiştir. Eigen vektörler temel bileşen analizinde de kullanılan öznitelik çıkarma yöntemlerinden birisidir. Bu vektörler verilen girdinin en önemli özniteliklerini tanımlamaktadır. Resim 2.8 soldaki büyük fotoğraf bilinmeyen resim olarak sisteme verilen fotoğraf olup sağdaki fotoğraflar da veritabanında bulunan diğer fotoğrafları göstermektedir. Bu fotoğrafların içinde soldaki resimle en çok benzeyen resim kırmızı ile işaretlenmiştir. Bu da Turk'un önerdiği algoritmanın ne kadar başarılı bir algoritma olduğunu göstermektedir.

2.2.CNN ile Yapılmış Olan Çalışmalar

Nesne tanıma işlemi askeri sistemler için de çok büyük önem arz etmektedir. Görüntü sensörleri, nesnelerin büyüklüğünün değişimi ve görüntüsünün ışığa veya diğer değişkenlere bağlı olarak değişmesi sebebiyle nesne tanıma problemlerinde başarılı bir sonuç vermemektedir. Son zamanlarda derin öğrenme yöntemlerinde

Resim 2.8: Yüz tanıma örnek resimleri - Turk[39]

yaşanan gelişmeler sonucunda evrişimsel sinir ağlarının(CNN) performansı oldukça artırılmıştır. CNN mimarileri oldukça basit nöron adı verilen öğrenebilen ağırlık ve bias parametreleri olan yapılardır. Her nöron gelen inputları toplayarak kendi inputu olarak kullanır ve opsiyonel olarak doğrusal olmayan yapılar kullanılarak mimarinin doğrusal olmayan sistemlerin modellenmesinde daha başarılı bir şekilde öğrenmesini sağlamaktadır. Öznitelik çıkartma işlemi nesne tanıma problemlerinin en önemli parçasıdır. CNN mimarilerinin en önemli avantajı mimarinin kendi özniteliklerini çıkarabilmesidir.



Resim 2.1: CNN örnek mimari- [7]

Evrişimsel Sinir Ağlarında En Sık Kullanılan Katmanlar

- **Evrişimsel Katman:** Bu katmanda input olarak gelen resimlerin belli bir

filtre(kernel) büyüklüğü seçilerek bu filtrelerin görüntülerin üzerinden geçirilirken öge(element) tabanlı çarpma işlemi yapılmasıdır. Bu işlem sonucunda yeni bir resim oluşturulacaktır. Bu filtrelerin kaydırılıp resimlerin üzerinden geçirilme işlemine konvolusyon denilmektedir. Bu katman içinde kullanılan en önemli parametreler filtre büyüklüğü, stride adı verilen filtrenin ne kadar kaydırılacağını belirleyen parametredir. Ayrıca dolgu(padding) adı verilen diğer parametre de çıkış resminin çerçevesine ne kadar sıfır ekleneceğini belirleyen parametredir.

- **Aktivasyon Birimi:** Bu katman her evrişimsel katmanın arkasında otomatik olarak kullanılmaktadır. Bu birim evrişimsel katmanın çıkışında resmin her piksel değerini maksimum ve minimum sınırlarına göre sınırlamaktadır. Bu birim için en çok kullanılan model ise ReLu adı verilen yükseltici birimlerdir.

- **Örnekleme(Pooling) Katmanı:** Bu katman genellikle evrişimsel katmanlardan sonra kullanılmaktadır. Bu katman verilen filtre boyutuna göre pikseller üzerinde parametrik olarak ya ortalamalarını alıp tek değer üretecektir veya maksimumunu seçecektir. Bu katman için de genellikle maksimum parametresi kullanılmaktadır(Max Pooling). Bu parametrenin önerilmesi oluşturulan modellerdeki deneyimler sonrasında ortaya çıkmıştır.[57]

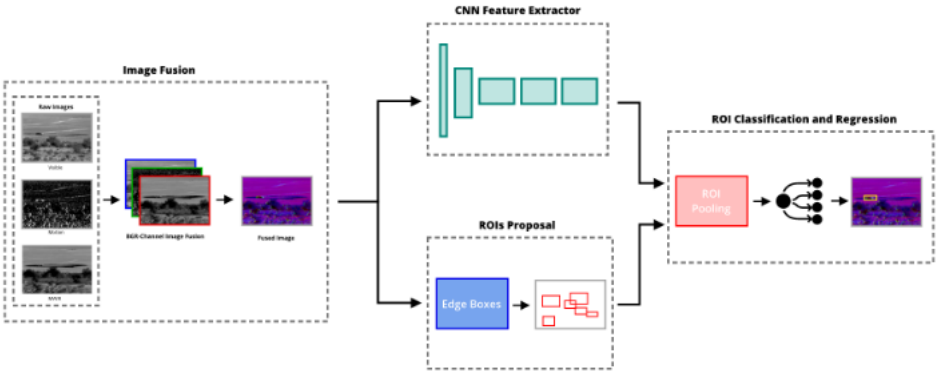
- **Normalizasyon Katmanı:** Bu katman genellikle evrişimsel katmanla pooling katmanı arasında kullanılmaktadır. Evrişimsel katmanın sonunda oluşan resim çarpımlar sonucunda oluştuğu için değerler çok yüksek olmaya başlayacaktır. Bu durumu engellemek için genellikle normalizasyon katmanı kullanılır. Bu katman çoğunlukla başarımı artıracaktır. Çarpımlar sonucunda aşırı yüklenmeye(overflow) dayalı olarak sonuçlar hatalı olarak hesaplanabilecektir. Bu normalizasyon katmanı bu sonuçları belli bir aralığa çekecektir. Böylece sistemin öğrenilmesinde karşılaşılabilecek bazı sorunları önceden ortadan kaldıracaktır.

- **Tamamen Bağlantılı Katman(Fully Connected Layer):** Bu katmanda kullanılan tüm nöronlar birbiri ile bağlantılıdır. bu katmanda nöronların birbirine olan bağımlılıklarına göre sistemin ağırlık vektörleri hesaplanacaktır. Bu katmanda genellikle modelin sonunda bir veya iki adet bulunmaktadır. Bu katmanın çıkışında genellikle lineer bir fonksiyon olan softmax adı verilen katman kullanılmaktadır. Bu katman

$$Y = W * X + B \tag{2.3}$$

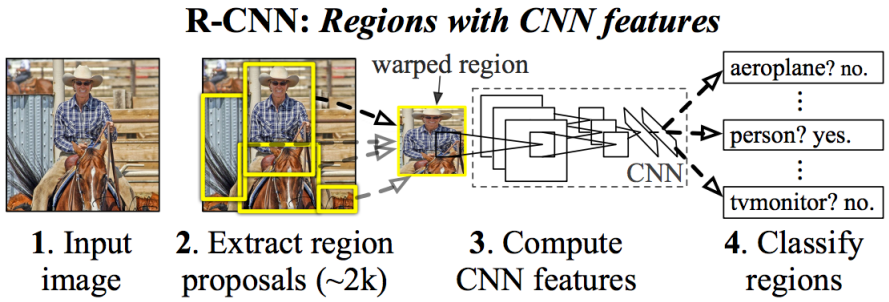
şeklinde uygulanmaktadır. Bu denklemde kullanılan X değişkeni o katmanın input nöronuna ait bir değerdir. Bu değer o nörona ait ağırlık değeri(W) ve offset değeri(B) ile yeni bir değere dönüştürülür. Bu denklemde kullanılan W vektörü her eğitim iterasyonunda güncellenip yeni iterasyonda bu değerler kullanılacaktır. Aynı şekilde B vektörü de her iterasyonda güncellenecektir[59].

Evrişimsel sinir ağları kullanılarak öznitelik çıkaran bir mimari geliştirilmiştir. Bu öznitelikleri kullanıp ROI Sınıflandırıcısı olarak adlandırılan bir algoritma ile nesne



Resim 2.2: CNN öznitelik çıkartma + ROI sınıflandırıcısı-Liu[61]

tanıma işlemi yapılmıştır [61]. Resim 2.2'de önerilen mimarinin detaylı bir versiyonu verilmiştir.
Nesne tanıma problemi için arka plan(background) ve ön plan(foreground) resim filtreleri kullanılması nesnelerin tespitini kolaylaştırmıştır. Bu konsepti kullanan bir CNN mimarisi R-CNN adı verilen ünlü bir mimari geliştirilmiştir. Bu mimaride verilen resimden lokal olarak ön plan ve arka plan resimlerinden kesilmiş bir resim oluşturulmuştur. Bu resim CNN mimarisine beslenerek nesne tanıma işlemi için bu mimari eğitilmiştir[54].



Resim 2.3: R-CNN algoritması detaylı gösterimi-Girshick[54]

R-CNN mimarisinin en temel problemi bu mimarinin oldukça yavaş sonuç vermesidir. Bu sebeple gerçek zamanlı bir sistemlere uyarlanmasını zorlaştırmaktadır. Bu mimariye hız performansı olarak katkı verecek bir çalışma daha yapılmıştır. Fast R-CNN adı da verilen bu çalışma eğitim süresini 9 kat artırıp sınıflandırıcı sonucunun üretilmesini 213 kat artırmıştır [53]. Bu algoritma C++ ve Python dilinde yazılmış olup tanıma başarımını da artıran gömülü sistemlere uygulanabilecek bir çalışmadır. Bu çalışma, çok fazla hafıza gerektirdiği için henüz gömülü sistemlerde kullanılamamıştır. Çizelge 2.2'de Fast RCNN karşılaştırmalı sonuçları verilmiştir.
2012 yılında gerçekleştirilen yarışmada birinci olup derin öğrenme modelleri için çığır açan bir çalışma olan AlexNet, derin öğrenme yöntemlerinin birçok problem için kullanım oranını artırmıştır [5]. CNN kullanılan öğrenme yöntemlerinde verisetinin minimum 10000 adet resime sahip olması gerekmektedir. Her problem için bu kadar çok veriseti her zaman bulunmadığı için performansını artırmak için

Çizelge 2.2: R-CNN karşılaştırmaları-Girshick[54].

Metod	**Başarım Oranı(%)**
SPPnetBB[32]	63.1
R-CNNBB[54]	66.0
Fast RCNN[53]	70.0

bir çalışma yapılmıştır. Bayes optimizasyon yöntemiyle daha az sayıda veriseti ile nesne tanıma problemine bir çözüm önermiştir[14]. Bayes optimizasyon algoritması aşırı öğrenme(overfitting) problemini kontrol altında tutmayı sağlamıştır. Bayes optimizasyon yöntemiyle CNN modeline verilen resimlerin sayısı artırılmıştır. Bu yöntemle CNN modellerinin ihtiyaç duyduğu veriseti sayısını azaltmak hedelenmiş olup oldukça başarılı sonuçlar vermektedir. Bu algoritma iki ayrı temle algoritmadan oluşmaktadır. Algoritmanın ilk parçası öğrenme oranının optimum değerini bulmayı hedeflemektedir. Her iterasyonda **??** denklemini sağlayacak en optimum y_x^* değerini sağlayan öğrenme oranını seçmektedir. İkinci kısımda ise CNN sınıflandırıcısını K-means kullanarak en iyi sonuç veren K tane modeli bulmaktadır. Resim 2.4, bu algoritmanın birkaç farklı CNN modeli için sonuçlarını göstermektedir.

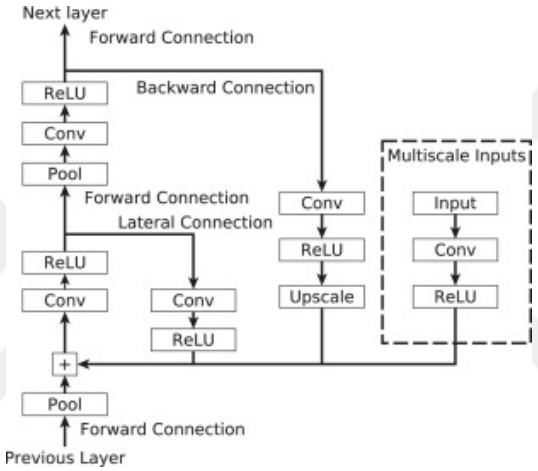
$$x^* = argmax_{x \epsilon D}(\mu(y_x^*) + \beta_t^{1/2}\sigma(y_x^*) \tag{2.4}$$

Dataset	Model types	Methods	Acc (%)	Dataset	Model types	Methods	Acc (%)
Flowers 102	Traditional models	MSDS[a]	73.4	Event8	Traditional models	Hybrid-CNN[c]	94.2
		Kanan and Cottrell, (2010)	75.2			S¹ICA[d]	95.8
	AlexNet	Trained on ODS	23.8		AlexNet	Trained on ODS	35.6
		FT on ODS	87.4			FT on ODS	91.5
		FT on ADSTM	87.9			FT on ADSTM	92.2
		FT on ADSWDM	89.1			FT on ADSWDM	94.8
	VGG-16	Trained on ODS	12.4		VGG-16	Trained on ODS	17.6
		FT on ODS	80.4			FT on ODS	85.0
		FT on ADSTM	82.9			FT on ADSTM	85.9
		FT on ADSWDM	83.6			FT on ADSWDM	87.3
	ResNet	Trained on ODS	28.9		ResNet	Trained on ODS	39.2
		FT on ODS	88.2			FT on ODS	91.6
		FT on ADSTM	90.1			FT on ADSTM	93.5
		FT on ADSWDM	92.5			FT on ADSWDM	95.1
Dogs	Traditional models	Simon and Rodner (2015)	68.1	15 Scene	Traditional models	SPM [e]	81.4
		Sermanet et al. (2014)	70.8			Xiao et al. (2010)	88.1
	AlexNet	Trained on ODS	13.4		AlexNet	Trained on ODS	21.8
		FT on ODS	75.1			FT on ODS	87.0
		FT on ADSTM	76.2			FT on ADSTM	87.8
		FT on ADSWDM	78.5			FT on ADSWDM	89.1
	VGG-16	Trained on ODS	8.8		VGG-16	Trained on ODS	18.4
		FT on ODS	72.4			FT on ODS	83.2
		FT on ADSTM	73.2			FT on ADSTM	83.9
		FT on ADSWDM	75.6			FT on ADSWDM	85.2
	ResNet	Trained on ODS	15.2		ResNet	Trained on ODS	25.3
		FT on ODS	76.9			FT on ODS	88.3
		FT on ADSTM	77.5			FT on ADSTM	89.2
		FT on ADSWDM	79.8			FT on ADSWDM	90.6
Caltech 101	Traditional models	LLC[b]	73.4	67 Indoor Scene	Traditional models	CNN-MOP[f]	68.9
		Hybrid-CNN[c]	84.8			S²ICA[d]	71.2
	AlexNet	Trained on ODS	15.8		AlexNet	Trained on ODS	20.3
		FT on ODS	90.2			FT on ODS	70.3
		FT on ADSTM	90.8			FT on ADSTM	71.4
		FT on ADSWDM	92.3			FT on ADSWDM	75.1
	VGG-16	Trained on ODS	10.2		VGG-16	Trained on ODS	15.9
		FT on ODS	80.6			FT on ODS	67.7
		FT on ADSTM	81.4			FT on ADSTM	68.5
		FT on ADSWDM	83.2			FT on ADSWDM	70.5
	ResNet	Trained on ODS	19.5		ResNet	Trained on ODS	25.9
		FT on ODS	88.1			FT on ODS	71.6
		FT on ADSTM	89.3			FT on ADSTM	72.1
		FT on ADSWDM	93.8			FT on ADSWDM	74.5

Resim 2.4: HAN Bayes optimizasyon + CNN sonuçları [14]

Derin CNN mimarisi kullanılan bir başka çalışmada da Cifar 10 veriseti[6] kullanılmıştır. Bu çalışmada 20 ve 56 katmanlı iki ayrı mimari önerilmektedir. Bu mimaride de bir önceki modelde olduğu gibi eğitim süreleri ve modelin sonucunun

çıkarılması işlemi oldukça uzun sürmektedir. Ayrıca bu modelde çok fazla sayıda katman kullanıldığı için modelin çıktısının verilmesi de uzun sürecektir[62].
Yinelenen CNN modelleri kullanılarak nesne tanıma işlemi de yapılmaktadır. Bu modellerde genellikle ilk katman evrişim katmanı sonraki katmanlarda bu evrişim katmanına bağlı olacak yinelenen evrişim katmanlarından oluşmaktadır. Klasik evrişimsel modellerde olduğu gibi bu modeller içinde evrişim katmanı sonrasında pooling katmanları bulunmaktadır. Yinelenen evrişimsel sinir ağları diğer modellere göre daha iyi sonuçlar vermektedir[43]. Şekil 2.4'de örnek bir yinelenen CNN mimari örneği verilmiştir. Bu modelde de görüleceği gibi her katman kendi özniteliklerini eğitim boyunca saklayıp kendinden sonra gelecek katmanların etkisiyle kendi ağırlıklarını güncellemektedir. Bu sebeple her katman kendi özniteliklerinin uzunluğu kadar hafıza harcamak zorundadır. Bu hafıza kullanımı gömülü sistemler üzerinde mimarinin eğitilmesi ve çalıştırılması mümkün görünmemektedir.



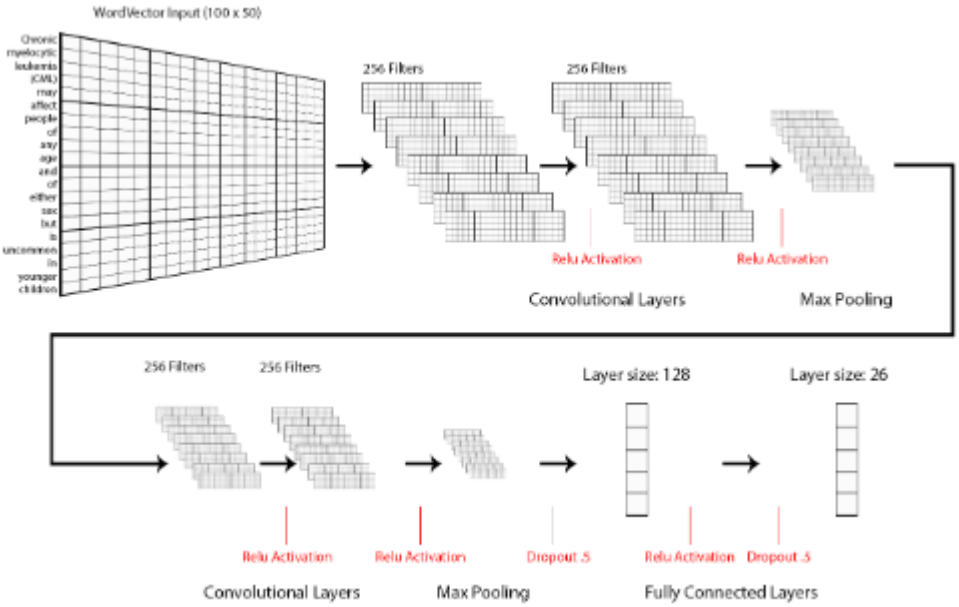
Şekil 2.4: Yinelenen CNN örnek mimarisi - [44].

Kaggle yarışmasında 5. sırada olan bir CNN mimarisi [46] yaklaşık olarak NVIDIA GTX760 ekran kartı kullanarak 20 saatte eğitim işlemini tamamlayıp 2.5 saat sonunda resmin sınıflandırma sonucunu vermektedir. Çizelge 2.3 Nagatomi tarafından oluşturulan mimarinin detaylarını açıklamaktadır.
Cifar-10 veriseti ile sınıflandırma yapan bir başka modelde de Saining Xie ve arkadaşları Kalıntılama Ağı(Residual Network) yöntemiyle bir model geliştirmişlerdir. Bu modelde başarım oranı %95.51'dir[55]. Bu modelin eğitilmesi için 22 saat 50 dakika gerekmektedir. Ayrıca 20 milyon parametresi olan bir sistem için yaklaşık olarak 20 GB hafıza gerekmektedir. Bu sebeple böyle bir modelin gerçek zamanlı olarak implemete edilmesi şu anki teknoloji ile mümkün değildir.
Evrişimsel sinir ağları, son zamanlarda çok sık kullanılan derin öğrenme yöntemlerinden biridir. Bu yöntem, nesne tanıma problemlerinde oldukça başarılı sonuçlar vermektedir. Evrişimsel sinir ağları çoğunlukla inputu görüntü olan problemlerde kullanılıyor olmasına rağmen görüntü dışında birçok alanda da kullanılmaya başlanmıştır. Doküman sınıflandırma problemlerinde genellikle word2vec [18] ve doc2vec [27], destek vektör makineleri kullanılmaktadır. Evrişimsel sinir ağları medikal doküman sınıflandırma problemi için kullanılmıştır[41]. Mark HUGHES ve arkadaşları word2vec yardımıyla doküman içerisindeki kelimeler(öznitelikler) çıkartılır ve bu kelimeler evrişimsel sinir ağlarının inputu olarak kullanılmaktadır. Bunun sonu-

Çizelge 2.3: Nagadomi mimari-[46].

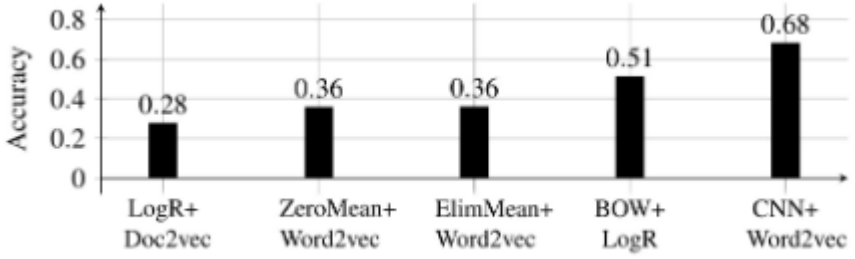
Layer Type	Parameters
input	size:24*24 channel:3
convolution	kernel:3x3 derinlik:64 dolgu:1
relu	
convolution	kernel:3x3 derinlik:64 dolgu:1
relu	
max pooling	kernel:2x2 kaydırma:2
dropout	oran:0.25
convolution	kernel:3x3 derinlik:128 dolgu:1
relu	
convolution	kernel:3x3 derinlik:128 dolgu:1
relu	
max pooling	kernel:2x2 kaydırma:2
droput	oran:0.25
convolution	kernel:3x3 derinlik:256 dolgu:1
relu	
convolution	kernel:3x3 derinlik:256 dolgu:1
relu	
convolution	kernel:3x3 derinlik:256 dolgu:1
relu	
convolution	kernel:3x3 derinlik:256 dolgu:1
relu	
max pooling	kernel:2x2 kaydırma:2
dropout	oran:0.25
linear	derinlik:1024
relu	
dropout	oran:0.5
linear	derinlik:1024
relu	
dropout	oran:0.5
linear	derinlik:10
softmax	

cunda da doküman 26 ayrı medikal doküman kategorisinden hangisine ait olduğunu tahmin etmektedir. Bu çalışmada %68 oranında başarım elde edilmiştir. Çalışmada önerilen model şekil 2.5 özetlenmiştir.



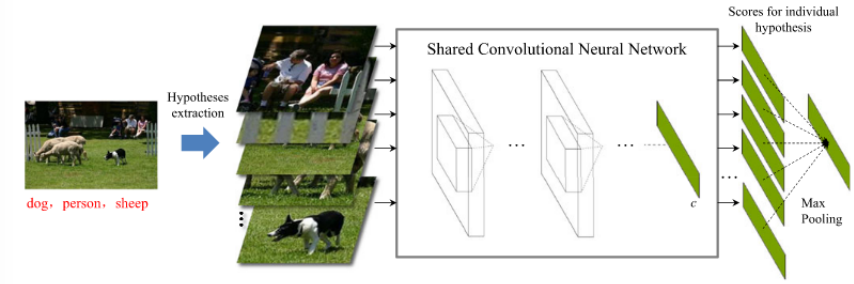
Şekil 2.5: Medikal doküman sınıflandırma mimarisi-[41].

Şekil 2.6'de diğer bilinen yöntemlerin sonuçlarına göre oldukça başarılı olduğu gözlemlenmektedir.



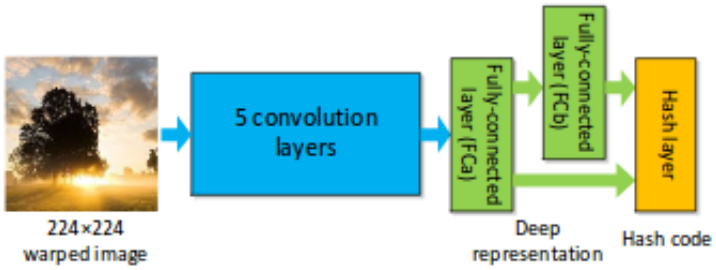
Şekil 2.6: Medikal doküman sınıflandırma-[41].

Nesne sınıflandırma işlemi genellikle resmin üzerinde tek bir nesne bulunduğunda ve bu nesnenin ne olduğunu bulmak için yapılmaktadır. Wang ve arkadaşları verilen görüntüden birden fazla nesne sınıflandıran bir model geliştirmişlerdir [68]. Yinelenen sinir ağları kullanılarak önerilen mimari ile birden fazla nesnenin sınıflandırılması başarılı bir şekilde gerçekleştirilmiştir.
Wei ve arkadaşları[71], hipotez olarak seçilen resimler CNN mimarisine verilerek resimler üzerinden tekli nesne sınıflandırma işlemi yapılmaktadır. Bu çalışmada, hipotez seçimi işlemi resimlerin seçilen bölgelerden kesilmesi işlemidir. Resim 2.5'de HCP çalışmasında önerilen mimari verilmiştir.
Zhao ve arkadaşları, beş adet evrişimsel katman sonrasında tam bağlantılı katmanların sınıflandırma sonucunu hash(şifreleme) yöntemiyle çıkan sonuca göre model



Resim 2.5: HCP mimarisi

eğitilmektedir.Resim 2.6'de Zhao ve arkadaşlarının önerdiği model verilmiştir. Bu çalışmada da birden fazla nesnenin sınıflandırılması üzerine çalışılmıştır.



Resim 2.6: Deep semantik mimarisi

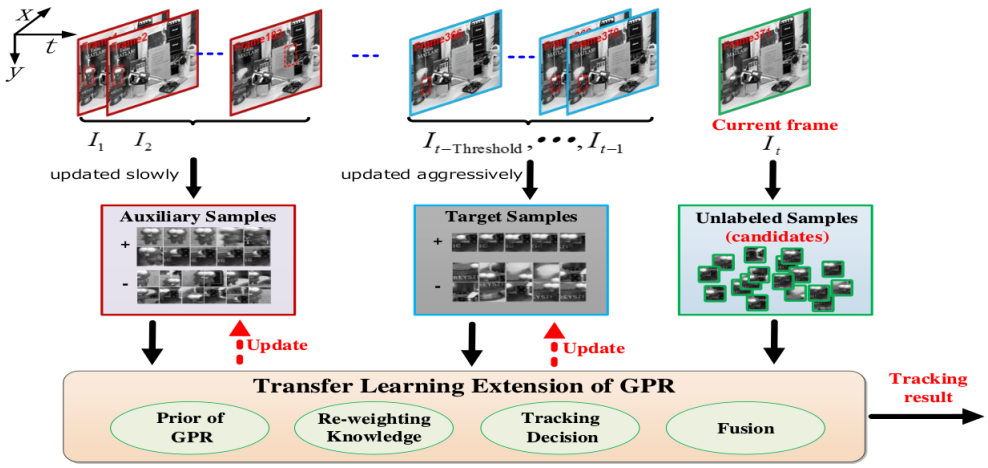
Nesne takip etme problemi, bilgisayarla görü alanında halen incelenmekte olan açık bir alandır. Nesne takip problemi çözümleri için genellikle korelasyon filtresi adı verilen filtreler oldukça kullanılmaktadır. Korelasyon filtresi resimler arasında benzerlik matrisi bulan bir algoritmadır. Bu filtre, Fourier dönüşümü tabanlı bir algoritma olduğu için hızlı bir algoritmadır. Ayrıca hızlı bir algoritma olduğu için her yeni gelen karede kendini eğitmesi de algoritmayı güçlü kılan etkenlerin başında gelmektedir. Derin sinir ağlarının da korelasyon filtresi ile beraber kullanıldığı çalışmalar bulunmaktadır. Korelasyon filtresi hızlı bir algoritma olduğu için eğitim nesnenin takip edilmesi sırasında her yeni gelen kare ile tekrarlanmaktadır. Modeli oluşturmak için özel bir eğitime gerek duymamaktadır[26].
Nense takip problemi literatürde 3 ayrı kategoride incelenmektedir.

- Nokta Takibi(Point Tracking): Takip edilen nesneler her karede noktalarla ifade edilip, bu noktaların yeni karedeki karşılığının yerini bulmayı hedeflemektedir.

- Kernel Takibi(Kernel Tracking): Takip edilen nesnelerin şeklini ve görünüşünü tanımlayan matrisler Kernel olarak tanımlanmaktadır. Genellikle Kernel-ler dikdörtgen şeklinde olmaktadır. Her yeni karede bu Kernelin öznitelikleri kullanılarak yeni karede bu özniteliklere en çok benzeyen bölge tespit edilmektedir. Bu şekilde nesnenin yeni yeri tahmin edilir.

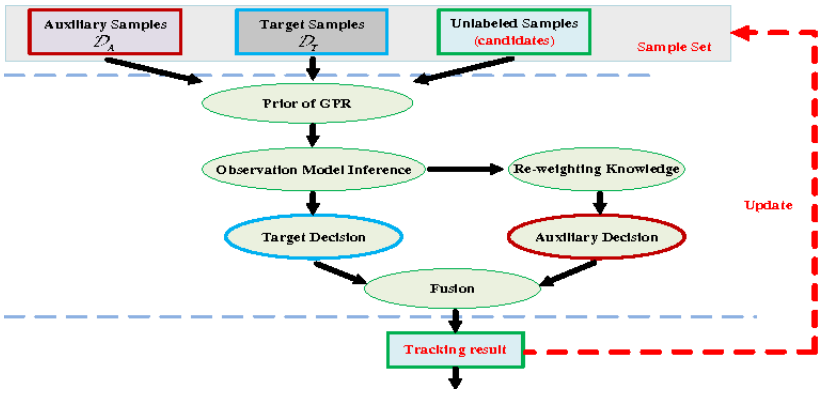
- Siluet Takibi(Silhouette Tracking): Takip edilmek istenen nesnenin kenar [11] öznitelikleri kullanılarak yeni gelen karede bu özniteliklerin eşleştirilmesi yöntemiyle yöntemi kullanılmaktadır.

Salari ve Sethi[56] engelleme(OCC) probleminin çözümü için bulunan öznitelik noktalarını diğer karede eşleştirme yapmaktadır. Ayrıca eksik olan noktalar yeni gelen kareden çıkarılarak tamamlanmaktadır. Böylece engelleme olduğu hissedildiğinde öznitelikler güncellenerek daha doğru öznitelik noktalarını elde etmeyi hedeflemektedir. Veenman ve diğerleri, hareket modeli oluşturmayı hedeflemektedirler. Takip edilen nesnenin nokta öznitelikleri çıkarılarak nesnenin takip ettiği yönde eşleşen noktaları aramaktadır. Hareket modeli ifadesinden kasdedilen özniteliklerin gittiği yönden nesnenin yeni yerinin tahmin edilebilmesidir[67].
Comaniciu ve Dorin[12] Bayesian olasılıksal bir model yardımıyla şekil(appearance) bazlı çıkarılan özniteliklerin bir sonraki karede en çok benzeyen bölgeyi bulmayı hedeflemektedir. Bu çalışmada Bayesian model takip edilmek istenen nesnenin ne kadar hareket ettiğini tahmin etmek için geliştirilmiştir.
Kernelized Correlation Filter(KCF) adı verilen bir çalışmada, HoG öznitelikleri FFT kullanılarak çıkartılmıştır. FFT kullanımı matris hesaplamalarından kaynaklanan gecikmelerin ortadan kaldırılmasını sağlamıştır [29]. HoG öznitelikleri kullanılarak bir diğer karede bu özniteliklerin maksimum eşleşme sağladığı bölge takip edilmek istenen nesnenin yeni yeri olarak tanımlanmaktadır. Bu süreç takip edilmek istenen nesnenin algoritma tarafından bulunamamasına kadar devam etmektedir.
TGPR adı verilen bir diğer çalışmada, Comaniciu ve Dorin[12]'in çalışmasındaki gibi bir Bayesian tabanlı takip çatısı geliştirilmiştir[15]. TGPR, Comaniciu'nun aksine her karede kendi özniteliklerini güncellemek yerine arka plan öznitelikleri çıkarılarak nesnenin öznitelikleri bu arka plan özniteliklerine göre güncellenmektedir. Bu algoritmanın detayları Resim 2.7 ve şekil 2.7 resimlerinde özetlenmiştir.

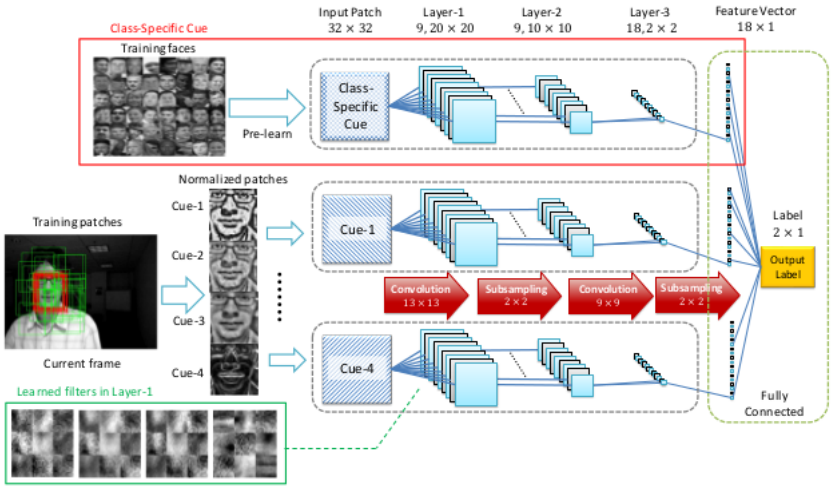


Resim 2.7: TGPR algoritması- [15]

CNN nesne takibi problemleri için de kullanılmıştır fakat bu yöntem oldukça fazla hafıza gerektirdiği ve compleks yani çalıştırılma süresi uzun olduğu için gerçek zamanlı sistemlerde kullanılamamaktadır[24][69]. Hanxi Lii ve arkadaşları DeepTrack adı verdikleri çalışmada 4 ayrı katman halinde farklı parametrelerdeki kesilmiş resimler ayrı ayrı CNN mimarilerine beslenmektedir. Oluşturulan görüntüler yerel kontrast normalizasyonu değişken parametrelerle uygulanmaktadır. Resim 2.8 algoritmanın mimari detaylarını özetlemektedir[19].
Hong ve arkadaşları, önceden eğitilmiş CNN modeli ile nesneyi tanımlamayı he-

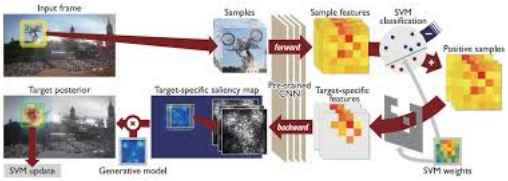


Şekil 2.7: TGPR algoritması- [15].



Resim 2.8: DeepTrack algoritması- [19]

deflemeketedir. Her karede, algoritma bir önceki karenin yakın noktalarında aday kerneller seçmektedir. Önceden eğitilmiş CNN modeli yardımıyla aday kernellerden öznitelikler çıkartılmıştır. Örnekleme katmanı sebebiyle bazı özniteliklerin kaybedilmesi takip etme performansını düşürmektedir. CNN sonrasında çıkartılan öznitelikler SVM kullanılarak ön plan(nesne spesifik öznitelikler) özniteliklerini çıkarmaktadır. Bu öznitelikler yardımıyla da nesnenin yeni pozisyonu tahmin edilmektedir [60]. Resim 2.9'de Hong ve arkadaşlarının önerdiği algoritma daha detaylı bir şekilde verilmiştir.



Resim 2.9: Hong CNN algoritması- [60]

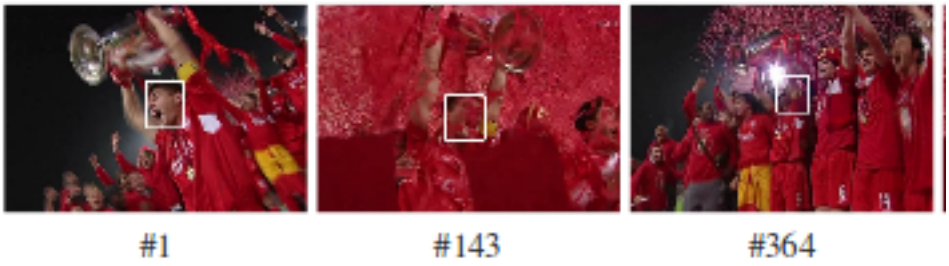
Zhou ve arkadaşları CNN'nin güçlü öğrenme kabiliyetleri ile birlikte particle filtresini kullanmaktadır. CNN takip etme problemi için gereken öznitelikleri çıkaran bir model sağlamaktadır. Bu model particle filtresi ile entegre ederek takip problemini çözmektedir[75]. Şekil 2.8 algoritmanın sonuçlarını göstermektedir. Engelleme(OCC) probleminin görüldüğü videolarda bu algoritma hatalı sonuçlar vermektedir. Bu da algoritmanın yeterli olmadığını göstermektedir.

	Ours	CSK	CT	CXT	IVT	LSK	MIL	SCM	TLD	VTD	VTS	Struck
boy	**99.3**	84.2	68.8	49.7	32.6	*99.7*	38.5	43.9	93.5	78.6	79.6	*97.5*
car4	**100**	27.6	27.5	29.9	*100*	5.61	27.6	*97.3*	*78.1*	35.4	35.2	39.8
carDark	98.2	*99.2*	0.25	69.0	69.7	*100*	17.8	*99.7*	52.9	68.4	*100*	*100*
coke	**85.9**	*73.9*	9.28	59.1	13.1	16.2	11.7	33.7	28.9	13.7	14.4	*94.2*
crossing	**100**	31.7	*98.3*	34.2	24.2	11.7	*98.3*	*100*	51.7	41.7	40.0	*94.2*
david	**92.4**	23.6	42.7	83.4	79.4	58.6	22.9	*91.3*	*97.0*	67.7	73.0	23.6
deer	**100**	*100*	4.23	*91.5*	2.82	33.8	12.7	2.82	*73.2*	4.23	4.23	*100*
dudek	94.3	94.7	85.2	92.4	96.8	92.7	85.7	97.6	84.2	*100*	*99.6*	*98.0*
faceocc1	**100**	*100*	85.4	77.1	*97.5*	40.8	76.5	*100*	83.4	*92.5*	88.3	*100*
fish	**100**	4.20	88.9	*100*	*100*	33.2	38.7	86.3	*96.2*	64.3	*97.9*	*100*
girl	**96.8**	39.8	17.8	64.2	18.6	34.4	29.4	*88.2*	76.4	65.2	52.6	*98.0*
jumping	**99.4**	4.79	0.64	28.8	9.90	6.39	47.6	12.1	*84.7*	11.2	15.7	*79.9*
mhyang	**100**	*100*	73.0	*100*	*100*	*100*	38.9	*99.7*	89.3	94.8	*97.0*	*100*
mntBike	**100**	*100*	17.1	28.1	*98.2*	90.4	57.5	96.1	25.9	*100*	*99.6*	85.5
singer1	**100**	29.6	24.8	32.2	*48.1*	19.7	27.6	*100*	*99.1*	43.0	42.5	29.9
singer2	**87.7**	3.55	1.09	3.83	3.83	4.10	*47.5*	16.4	3.01	*45.1*	39.1	3.55
sylvester	83.6	71.7	82.8	74.7	67.6	26.3	54.6	*88.6*	*92.8*	80.4	80.7	*92.9*
trellis	**95.3**	59.1	35.0	80.8	30.9	*90.7*	24.4	*85.4*	47.3	50.1	49.0	78.4
walking2	**100**	38.8	38.4	39.8	*100*	*48.6*	38.0	*100*	34.0	40.2	40.4	*43.4*
woman	**80.6**	24.5	15.9	20.6	18.4	18.9	18.8	*85.8*	16.6	18.1	17.1	*93.5*
boy	**2.23**	20.1	9.03	7.39	91.3	*2.24*	12.8	51.0	4.49	7.57	7.27	*3.84*
car4	**2.84**	19.1	86.0	58.1	*2.15*	66.6	50.8	*4.27*	13.6	37.0	36.7	8.69
carDark	**1.17**	3.23	119	16.5	8.43	*1.29*	43.5	1.30	27.5	16.5	2.87	*0.95*
coke	**14.4**	*13.6*	40.5	25.7	83.0	55.0	46.7	56.8	25.1	68.7	62.5	*12.1*
crossing	**1.76**	8.96	3.56	23.4	*2.36*	54.8	3.18	*1.57*	24.3	26.1	43.1	2.81
david	5.64	17.7	10.5	6.05	*4.82*	12.0	16.9	*4.34*	*5.12*	11.6	10.2	42.8
deer	**4.56**	*4.97*	246	6.75	183	98.8	101	104	6.26	135	220	*5.27*
dudek	14.7	13.4	26.5	12.8	*9.62*	14.6	17.7	10.8	18.1	*10.3*	*9.85*	11.4
faceocc1	**12.7**	*11.9*	25.8	25.3	18.4	30.4	29.9	*13.0*	27.4	20.2	21.3	18.8
fish	**4.19**	41.2	10.7	6.25	*5.67*	50.8	24.1	8.54	6.57	16.8	7.21	*3.40*
girl	**3.68**	19.3	18.9	11.0	22.5	29.3	13.7	*2.60*	9.79	8.60	13.0	*2.57*
jumping	**3.94**	86.0	47.7	9.99	61.6	74.6	9.99	65.9	*5.94*	41.4	40.1	*6.55*
mhyang	**2.39**	3.61	13.3	3.97	*1.87*	3.43	20.4	*2.41*	9.51	4.36	4.07	2.59
mntBike	**7.72**	*6.51*	214	179	7.66	11.5	73.0	10.6	209	9.78	9.67	8.63
singer1	**4.32**	14.0	15.5	11.4	11.3	20.8	16.4	*2.72*	7.99	*4.19*	5.35	14.5
singer2	**10.3**	185	127	164	175	149	*22.5*	114	*8.47*	43.7	72.5	174
sylvester	13.2	9.92	8.56	14.8	34.2	68.4	15.2	*7.97*	*7.31*	19.6	19.4	*6.30*
trellis	**3.73**	18.8	41.7	7.01	120	*4.70*	71.5	7.01	31.6	32.3	24.3	*6.92*
walking2	**3.35**	17.9	58.5	34.7	*2.46*	18.9	60.6	*1.65*	52.5	46.2	54.0	11.2
woman	**9.64**	207	114	72.5	177	131	125	*7.88*	93.9	119	121	*4.17*

Şekil 2.8: Zhou sonuçları-[75].

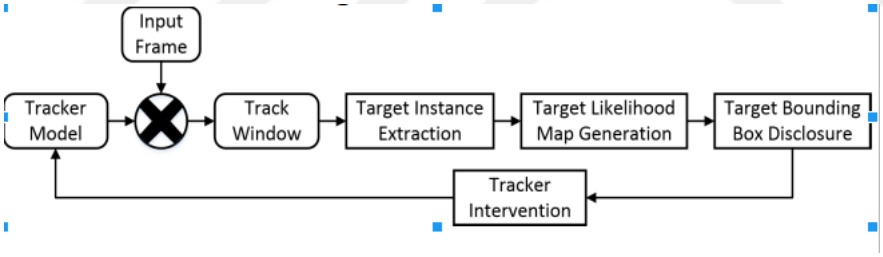
VTD adı verilen bir başka çalışmada, nesnenin takibi iki ayrı bakış açısıyla incelenmiştir. Bunların ilki gözlemleme modeli olup nesnenin değişimlerini takip edip sürekli modeli güncelleyen bir yapıdır. Diğeri ise hareketi modelleyen bir yapıdır. Bu modeller, birden fazla olacak şekilde sistem bu modelleri eğitmeye çalışır. Ada Boost algoritması[22], birden fazla az başarım elde edilen modellerin birleştirilmesi

ile daha iyi bir başarım elde edilebileceğini göstermiştir. VTD çalışmasında da birden fazla model(gözlem modeli(n adet) ve hareket modeli(s adet)) birleştirirek daha iyi bir başarım elde edilmiştir [25]. Resim 2.10 bu algoritmanın örnek bir çıktısını göstermektedir.



Resim 2.10: VTD algoritması sonuçları-[25]

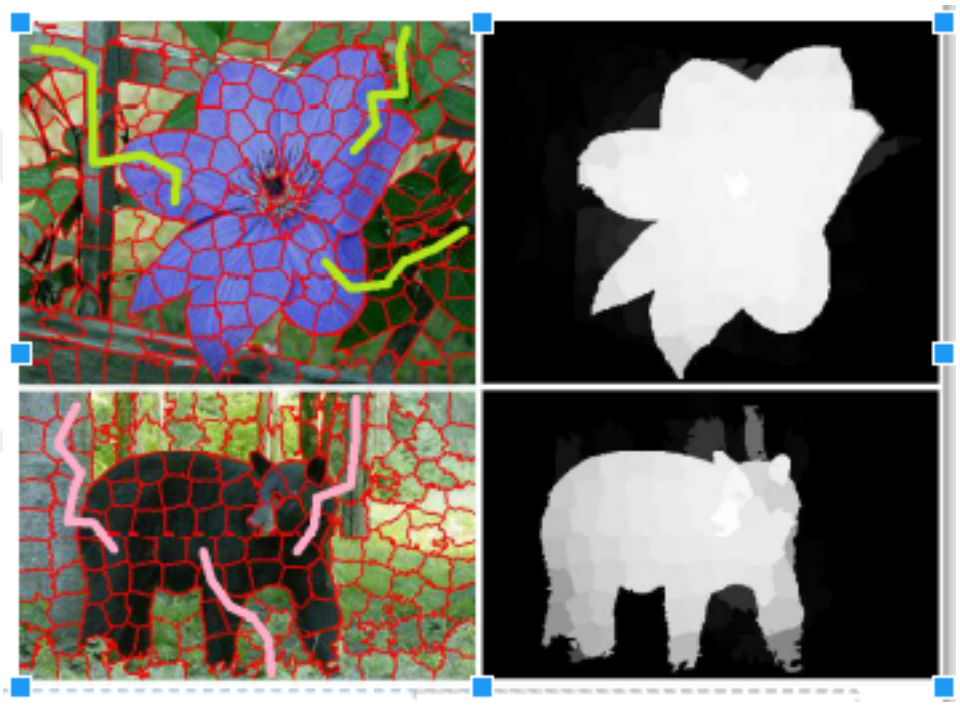
Korelasyon filtresi yaklaşımıyla bir başka çalışmada adaptif olarak bölütlenmiş nesnenin her yeni karede korelasyon filtre özniteliklerine en çok benzeyen bölge yeni karedeki nesnenin yerini belirlemektedir. Emre Tunalı ve arkadaşlarının önerdiği model Şekil 2.9'de özetlenmiştir. Bu çalışmada önerilen methodolojide, korelasyon filtresi kullanıldığı için algoritma gerçek zamanlı bir sistem üzerinde kolaylıkla uyarlanabilmektedir. Bu algoritmanın girişi takip penceresi büyüklüğü beklemektedir. Bu çalışmanın tek dezavantajı budur. Takip edilmek istenen nesnenin büyüklüğü daha küçük veya daha büyük olabilmektedir, fakat önerilen modelde bu pencere büyüklüğü değiştirilememektedir [65].



Şekil 2.9: Korelasyon filtresi tabanlı nesne takibi-[65].

Saliency metriği kullanılarak arka plan ile ön plan öznitelikleri ayırt edilmektedir. Saliency, nesnenin bölütlerinin birbirine benzeyen kısımlarını birleştirerek nesnenin ön plan siluetini ortaya çıkarmaktadır. Resim 2.11 'de bölütleme yöntemiyle takip edilmek istenen nesnenin ön planının nasıl çıkarıldığı görünmektedir.
Önplan öznitelikleri çıkarılan nesnenin takip edilmesinde Denklem 2.5'te verilen denklem kullanılmaktadır. Bu denklemde bulunan λ değeri her karede tekrar hesaplanmaktadır.

$$S_{target}[n] = (1-\lambda[n])S_{target}[n-1] + \lambda[n]S_{instance}[n] \tag{2.5}$$



Resim 2.11: Bölütleme ile ön plan Çıkarma

3.EVRİŞİMSEL SİNİR AĞLARI İLE NESNE TANIMA

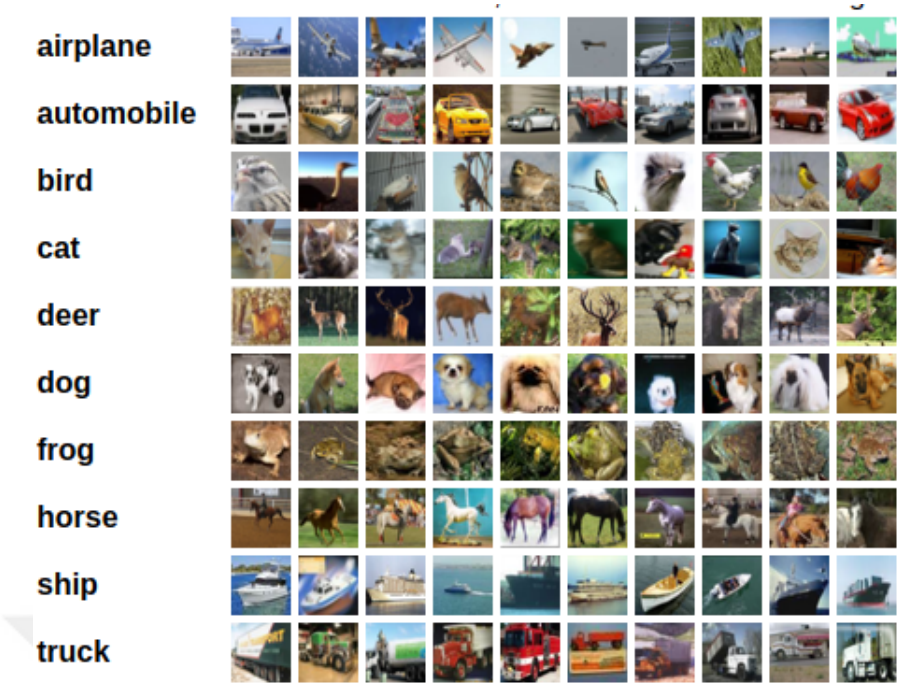
Evrişimsel sinir ağları, son zamanlarda çok sık kullanılan derin öğrenme yöntemlerinden biridir. Bu yöntem, nesne tanıma problemlerinde oldukça başarılı sonuçlar vermektedir. Evrişimsel sinir ağları çoğunlukla inputu görüntü olan problemlerde daha sıklıkla kullanılıyor olmasına rağmen görüntü dışında birçok alanda da kullanılmaya başlanmıştır.

3.1.Amaç

Evrişimsel sinir ağları genellikle çok fazla hafıza gerektiren makine öğrenme yöntemlerinden birisidir. Bu yöntemin her makine öğrenme yönteminde olduğu gibi öğrenme, doğrulama ve test aşaması bulunmaktadır. Literatür Araştırması2 bölümündeki çalışmalarda olduğu evrişimsel sinir ağları kullanılan çalışmalarda çoğunlukla eğitim aşamaları bilgisayarlar kullanılarak(offline) yapılmaktadır. Bu eğitim aşamaları sonucunda oluşturulan modelin katsayıları ve sapma değerleri bir dosyaya yazılmaktadır. Sistem içerisinde bu dosyadan mimari ve mimarinin sahip olduğu parametreler okunup sınıflandırma işlemini matris çarpımıyla uygulamaktadır.
Bu çalışmadaki amaç, evrişimsel sinir ağları yardımıyla Cifar-10 verisetini[6] kullanarak sınıflandırma yapabilecek ve aynı zamanda gömülü bir sistemde çalıştırılabilir bir mimari geliştirmektir.

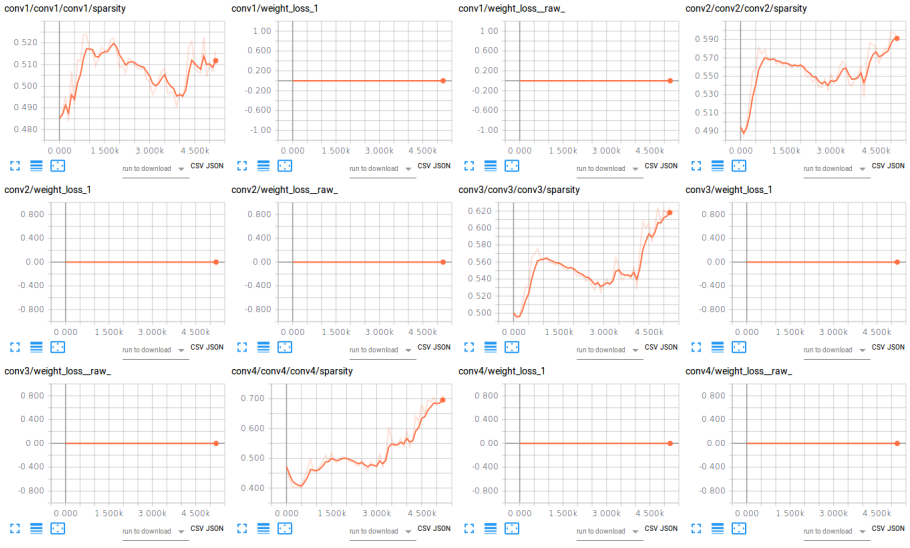
3.2.Araştırma Gereksinimleri

Bu araştırmada öncelikle kullanılacak olan evrişimsel sinir ağları kütüphanesi seçilmesi gerekmektedir. Bunun için literatürde en sık kullanılan Caffe kütüphanesi ile çalışmaya başlanmıştır[28]. Caffe kütüphanesinin en büyük avantajı sağladığı kütüphanenin hem C++ hem de Python arayüzünün bulunmasıdır. Bu kütüphane yardımıyla konvolusyon gibi işlem yükü fazla olan fonksiyonları optimize bir şekilde derleyip aynı zamanda bu fonksiyonların NVIDIA ekran kartında çalıştırmak için geliştirilmiş CuDa altyapısına da uygun geliştirilmesi eğitim aşamasının oldukça hızlanmasına olanak sağlamıştır. Caffe kütüphanesi uygulanacak olan mimarinin parametreleri .config uzantılı bir dosyadan çekmektedir. Bu dosyanın bir örneği Ek-1[] bölümünde verilmiştir.
Bir diğer kütüphane Google tarafından geliştirilen NVIDIA CuDa altyapına uygun olan Tensorflow kütüphanesidir [3]. Tensorflow sadece Python platformunda çalışmaktadır. Eğitim aşamalarında ve test aşamalarında modelde kullanılan resimleri göstermek ve anlık olarak modelin parametrelerinin değişimini göstermek için Tensorboard adı verilen bir uygulama geliştirilmiştir. Bu uygulama yardımıyla her katmanın varyansı anlık olarak görüntülenebilmektedir. Öğrenme problemlerinde



Resim 3.1: Cifar-10 sınıflar ve örnekleri

katman parametrelerinin ne kadar değiştiğini gözlemlemek büyük bir öneme sahiptir. Çünkü modelin parametreleri için uygulanan meyilli azalım(Gradient Descent) algoritması parametrenin varyansının az olduğu durumda lokal minimum noktasını bulduğunu iddia etmektedir. Bu uygulamanın katmanlarının gösterildiği bir ekran çıktısı Şekil 3.1'de verilmiştir.



Şekil 3.1: TensorBoard katman(Layer) değişim grafikleri.

Tensorboard, sisteme verilen resimlerin örneklerini web uygulamasında gösterebilmektedir. Tensorboard localhost 6006 portunda çalışan bir uygulamadır. Bu uygulamanın test aşamasında kullanılan resmin bir örneği Resim 3.1'de verilmiştir.
Bu kütüphanelerin haricinde Keras, PyTorch gibi kütüphaneler de bulunmaktadır.

Resim 3.1: TensorBoard örnek resim

Bunun haricinde Google tarafından sağlanan `https://colab.research.google.com` sitesinde Google'un sağladığı sanal GPU'lar kullanarak probleminiz için yeni modeller birden fazla GPU kullanarak daha hızlı bir şekilde eğitim aşaması gerçeklenebilmektir.

3.3.Deneysel Çalışmalar

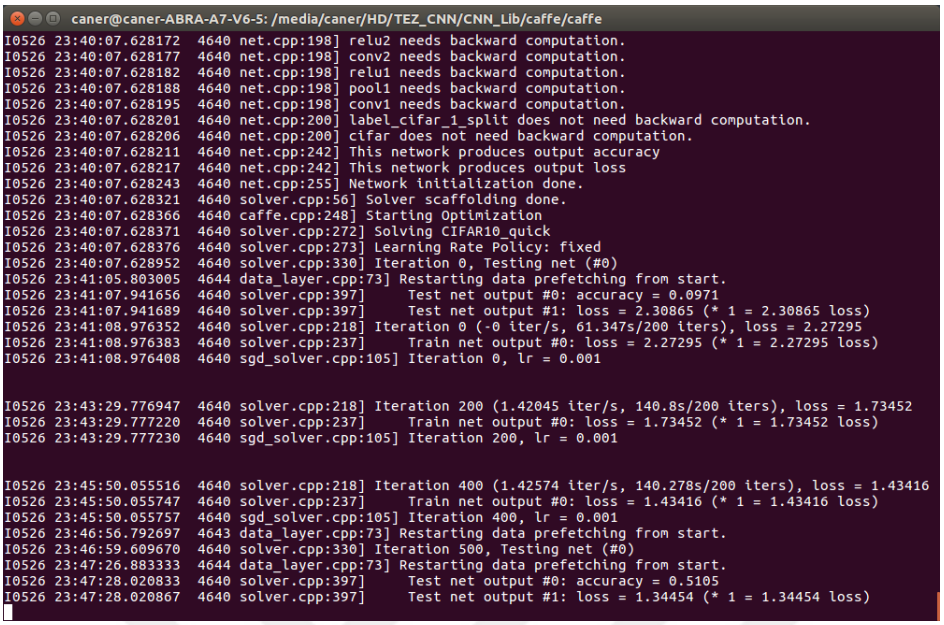
Cifar-10 verisetinde toplamda 60000 adet 32x32'lik resimler bulunmaktadır. Bu resimlerin 50000 tanesi eğitim ve doğrulama aşamalarında kullanılmış olup 10000 adet resim test aşamasında kullanılmıştır.
Caffe ve Tensorflow kütüphanelerinin ikisi de ayrı ayrı test edilmiştir. Caffe ve Tensorflow kütüphanelerinin Ubuntu kurulu bilgisayarlarda kurulum aşamaları ek bölümüne eklenmiştir.

3.3.1 Caffe kütüphanesi Cifar-10 dataseti ile nesne tanıma

Cifar-10 sınıflandırma sonuçları

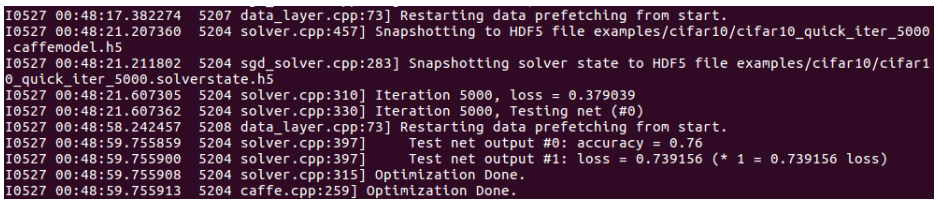
Caffe kütüphanesinin C++ API'leri kullanılarak bir mimari geliştirilmiştir. Caffe kütüphanesi eğitim sırasında belirli aralıklarla test aşamasını da koşturmaktadır. Modelin verdiği test başarım oranı ile eğitim tamamlandıktan sonra yapılan test başarım oranı birbiriyle tutmamaktadır. Bu sebeple Cifar-10 için geliştirilen bu model kullanılmamıştır. Literatürde kullanılan Caffe altyapıları genellikle Python tabanlıdır, ayrıca son zamanlarda Caffe2 adında yeni bir kütüphane daha yayınlanmıştır.[`https://github.com/caffe2/caffe2`] Bu çalışmada Caffe altyapısı bahsedilen tutarsızlık sebebiyle kullanılmamaya başlamıştır. Caffe eğitim aşamasının ekran görüntüsü Resim 3.1'de verilmiştir. Caffe h5 adı verilen özel bir veri formatında modelin parametrelerini ve mimariyi saklamaktadır. Caffe bu dosyaları kendi içinde belli sıklıklarla saklayarak modelini güncellemektedir. Bu sayede modeli eği-

timini anlık olarak durdurarak yeniden başlatıldığında kaldığı yerden başlayarak devam edebilmektedir.



Resim 3.1: Caffe eğitim ekran görüntüsü

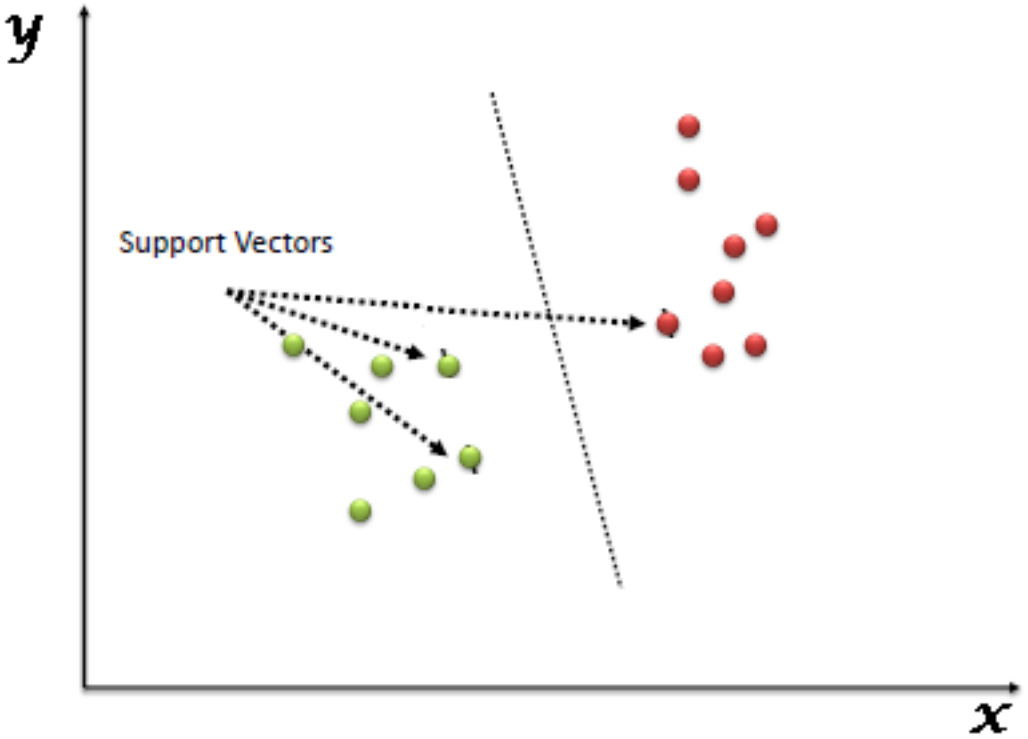
Caffe, test sonucunda %76 oranında başarım elde ettiğini söylemiş olsa da, test resimleri ile sistem test edilmek istenildiğinde bu başarım %43 seviyelerine inmektedir.



Resim 3.2: Caffe test sonuç ekran görüntüsü

Bu başarım oranını artırmak için en son katmandaki tam bağlantılı katmanın sonuna destek vektör makineleri eklenerek modelin başarım oranına etkisi gözlemlenmiştir. Destek vektör makinesi denetimli(supervised) makine öğrenme algoritmalarından birisidir. Bu yöntem hem sınıflandırma hem de uyumlama(regression) problemlerinde sıklıkla uygulanan bir yöntemdir. Bu algoritma genellikle sınıflandırma problemlerinde kullanılmaktadır. Doğrusal olarak ayrılamayan bir düzlemdeki veriler, destek vektör makineleri ve bu algoritma ile beraber kullanılabilecek radial tabanlı çekirdekler(kernel) yardımıyla farklı bir uzayda doğrusal olarak ayrılabilir hale getirilebilmektedir.

Caffe'de denenen modelin arkasına eklenen destek vektör makineleri başarım oranını yaklaşık %7 oranında artırmaktadır. Bu sonuçlara göre Caffe kütüphanesi ile yaklaşık olarak %50 oranında bir başarım elde edilebilmektedir. Caffe kütüphaneside karşılaşılan bu tutarsızlık sebebiyle bu kütüphanenin kullanılmasından vazgeçilmiştir.



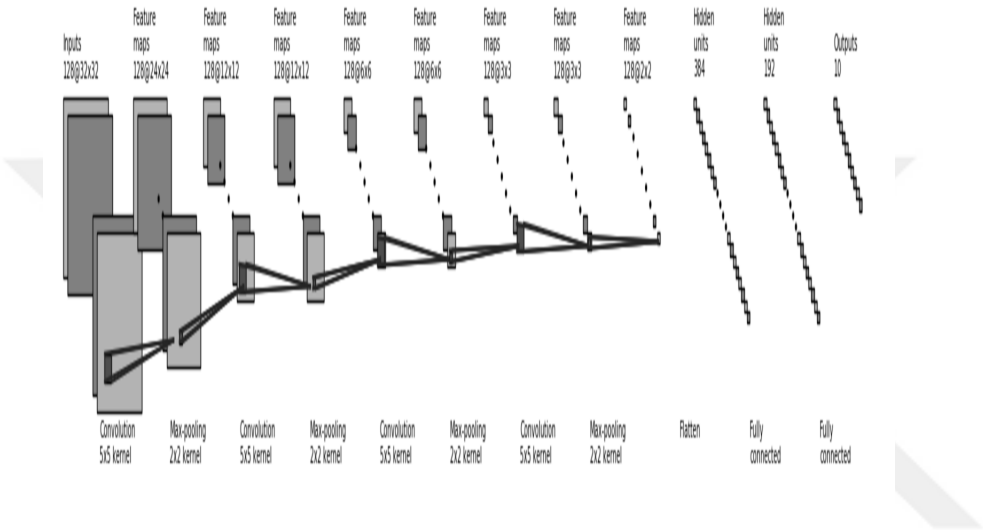
Şekil 3.2: SVM örnek sınıflandırıcı-[63].

3.3.2 Tensorflow Kütüphanesi Cifar-10 Dataseti ile Nesne Tanıma

Cifar-10 sınıflandırma sonuçları

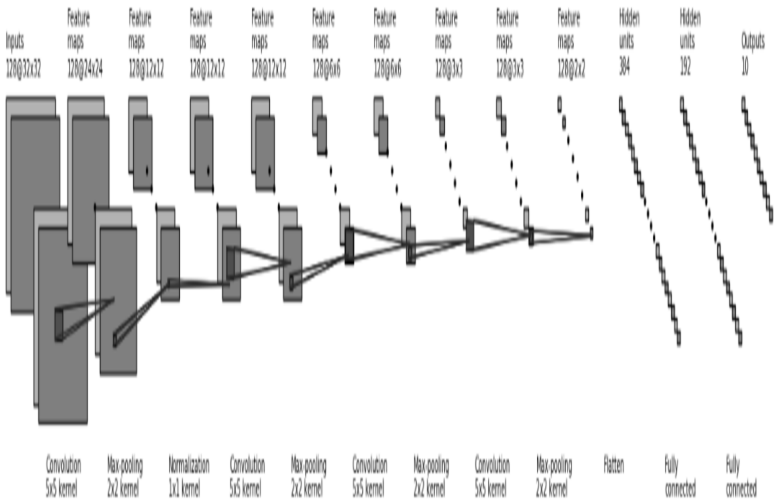
Cifar-10 veriseti, 32x32 lik resimlerden oluşmaktadır. Bu resimler 128'lik kümeler halinde mimariye entegre edilmektedir. Model çıkışında 128 resmin de sınıflandırma sonuçlarını elde etmiş olacaktır. Bu çalışmada kullanılan modelin parametreleri belirlenirken eğitilme süresi ve kullanılan hafıza miktarını minimize etmek hedeflenmiştir. Hafıza miktarı ve eğitilme süresinin azaltılması önerilen modelin gömülü bir sistem üzerinde çalıştılmasına olanak sağlamaktadır.
Bu çalışmada, katman sayısı evrişimsel katman ve tam bağlantılı katman sayısının toplamını ifade etmektedir. Katman sayısının artırılması genellikle başarım oranını artırırken, eğitim süresi ve kullanılan hafıza miktarı da artmaktadır. Bu sebeple evrişim katmanı ve tam bağlantılı katmanın sayısı kullanılan hafıza miktarına göre limitlenmiştir.
Evrişimsel sinir ağları kullanıldığında yapısı gereği modeli ezberlemeye yakınsamaktadır. Sistemin ezberlenmesi gerçek verilerle test edildiğinde hatalı sonuçlar vermesine sebep olmaktadır. Evrişimsel sinir ağlarında, ezberleme(overfitting) probleminin çözülmesi için verisetinin genişletilmesi gerekmektedir. Verisetlerindeki veri sayısı sabit olup kolaylıkla artılamadığı için veriyi normal yöntemlerle artırmak mümkün olmayabilmektedir. Verinin artırılması için görüntü işleme yöntemleri kullanılmaktadır. Veri sayısını artırmak için en sık kullanılan yöntemlerin başında resmi x veya y eksenine göre döndürmek gelmektedir. Bunun haricinde, sisteme verilen görüntülerin üzerine gürültü ekleyerek yeni görüntüler oluşturulabilmektedir.

SIU 2018 yılında gerçekleştirilen konferansta yayınladığımız çalışmada Cifar-10 veriseti kullanılarak limitli bir hafıza ile görüntü sınıflandırması GPU kullanılarak bilgisayar üzerinde modellenmiştir [51]. Bu çalışmada ilk önerilen modelde öğrenme oranı 0.4 olarak seçilmiş olup, her 350 iterasyonda(epochs) bir 0.5 oranında azaltılmıştır. Öğrenme oranının kademeli olarak azaltılması ezberleme probleminin de önüne geçilmesini sağlamıştır. İlk model 5 katmanlı bir mimariye sahip olup 3 tane evrişimsel, 2 tane de tam bağlantılı katmana sahiptir. Bu mimari, Şekil 3.3'de özetlenmiştir. Bu modeldeki başarım oranı %85.8'dir.

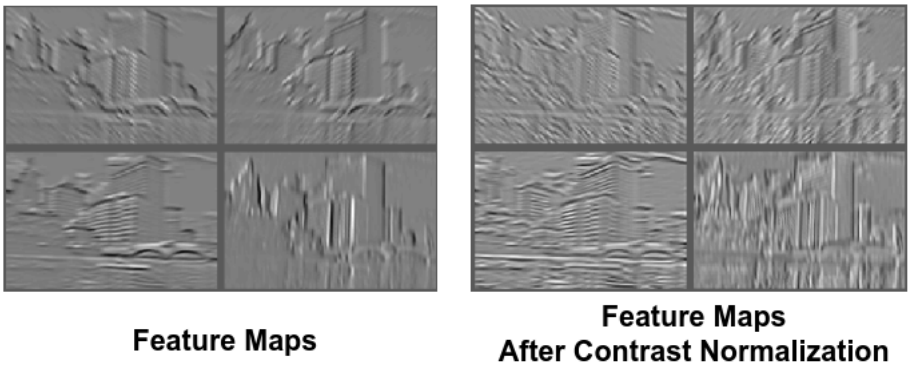


Şekil 3.3: CNN model-1.

İkinci önerilen modelde öğrenme oranının modele etkisini gözlemlemek için öğrenme oranının ilk değeri 0.1 olarak değiştirilmiştir. Model, Şekil 3.3 ile aynı parametrelere sahiptir. Bu modelde başarım oranı %85.9 olarak gözlemlenmiştir. Öğrenme oranını düşürmek daha yavaş öğrenmesine sebep olmaktadır. İlk modelde yaklaşık 100000 iterasyonda başarım oranı %85.8 oranına ulaşırken, bu model için %85.9 oranına ulaşabilmesi için 300000 iterasyona ihtiyaç duyulmuştur. Bu durum beklenen bir durumdur. Öğrenme oranı genellikle ezberleme problemini çözmek için kullanılan bir yöntemdir. Eğitim hatası azalırken, doğrulama ve test hatası artan bir modelde ezberleme problemi olma olasılığı oldukça yüksektir. Bu durumda genellikle öğrenme oranı azaltılarak yeni bir model denenmektedir.
Diğer geliştirilen modelde, öğrenme oranı Model-1(3.3)'de olduğu gibi ilk değeri 0.4 olarak belirlenmiş olup her 350 iterasyonda %50 oranında azaltılmaktadır. Bu modelde, örnekleme katmanı(pooling layer-1) ile evrişimsel katman(convolution layer-2) arasına normalizasyon katmanı eklenmiştir. Normalizasyon katmanı, girişine verilen görüntünün(matris) özntelikleri daha belirgin hale getirmek için kullanılan bir ara katmandır. Şekil 3.4'de bu modelin mimarisi verilmiştir.
Maksimum örnekleme(Max-Pooling) kullanılması görüntüdeki kenar ve köşe bölgelerinin daha baskın şekilde görünmesini sağlayacaktır. Bu özniteliklerin normalize edilmesi özniteliklerin daha da belirginleşmesine sebep olmaktadır. Bu sebeple ör-



Şekil 3.4: CNN model-3.



Resim 3.3: Normalizasyon katmanının özniteliklere etkisi-Gaurav Mittal[16]

nekleme katmanından sonra kullanılan normalizasyon katmanlarının ana görevi, örnekleme katmanı sonucu baskınlaşan kenar ve köşe noktalarının daha belirgin hale gelmesini sağlamaktadır. Bunun bir örneği Resim 3.3'de gösterilmiştir. Bu modelin başarım oranı %85.3 olmuştur. Maksimum örnekleme katmanı özniteliklerin azalmasına sebep olmaktadır. Bu özniteliklerin bastırılması maksimum olmayanları bastırma(non-maximum suppresion) algoritmasına çok benzeyen bir yapıdır. Bu sebeple, maksimum örnekleme katmanı ortalama örnekleme katmanına göre daha başarılı sonuçlar vermektedir. [58]

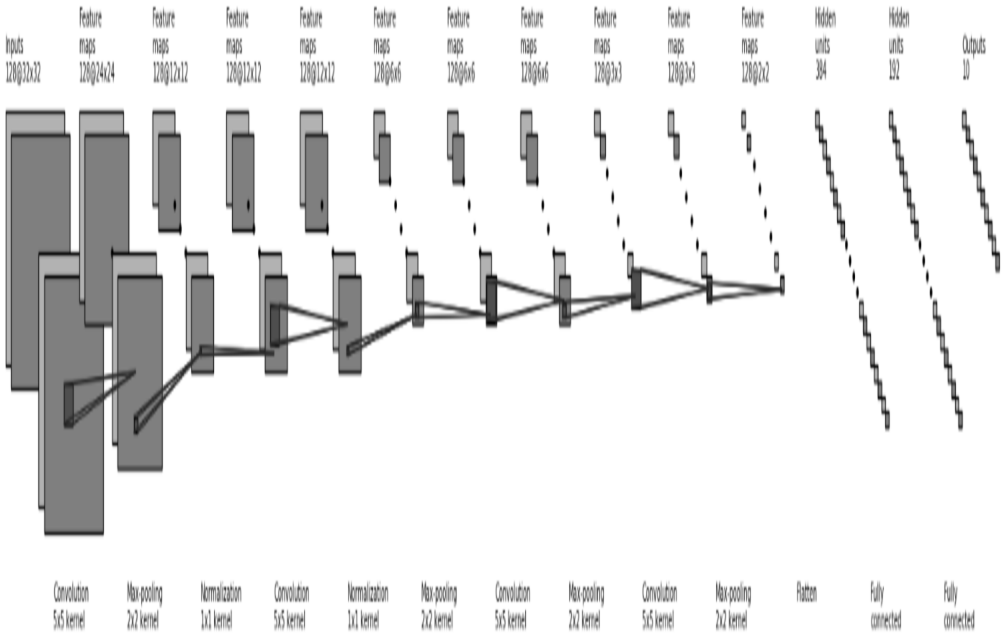
Bu problem için normalizasyon katmanının eklenmesi işe yaramamış olarak görünmektedir. Sisteme giriş olarak verilen resimlerin köşe ve kenar öznitelikleri kolaylıkla bulunabilmektedir. Bu öznitelikler daha da baskınlaştırılırken Harris detektöründe olduğu gibi köşe noktalarının maksimum olmayan noktaları elemine etme(non-maximum suppression) yöntemleri yardımıyla bu katmanın etkileri gözlemlenebilir. Fakat özniteliklerin normalize edilmemiş hali nesne tanıma işlemi için yeterli olması sebebiyle bu yöntem uygun bir model olarak seçilmemiştir. Bu yöntemin uygulanmamasının bir diğer sebebi de normalizasyon katmanı eğitim ve sınıflandırma sonucunun üretilme süresini artırmaktadır. Normalizasyon katmanı eklenmesi , modelin eğitilme süresini ve sınıflandırma sonucunu verme süresini %10 oranında artırmaktadır. Bu sebeple normalizasyon katmanın, örnekleme ile evrişimsel katmanının arasında kullanılmasından vazgeçilmiştir.

Bir diğer modelde, bir önceki modeldeki parametreler kullanılaraki ekstra bir normalizasyon katmanı, evrişimsel katman(convolution layer-2) ile örnekleme katmanı (pooling layer-2) arasına eklenmiştir. Normalizasyon katmanı, evrişimsel katmanda yapılan resim üzerinde yapılan konvolusyon operasyonu sonucunda oluşan değerlerin 32 bit sınırını geçmemesi için evrişimsel katmanın çıkışını 0-1 aralığına getirmeyi sağlayan bir katmandır (Minimum ve maksimum değerler kullanılan kütüphaneye göre değişebilmektedir. Tensorflow kütüphanesi için bu değerler 0 ve 1'dir.). Şekil 3.5'da bu modelin mimarisi verilmiştir. Bu durumda modelin başarım oranı da %85.3 olmuştur.

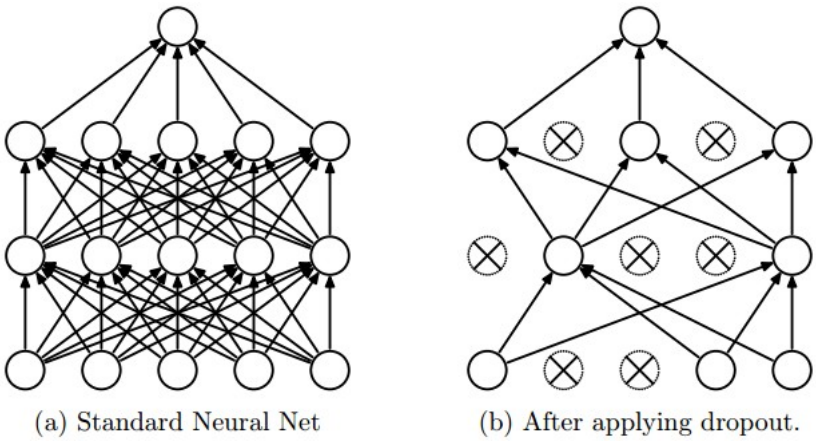
Normalizasyon katmanının evrişimsel katmanı sonrasına eklenmesi de işe yaramamıştır. Bunun en önemli sebebi, görüntüler modele verilmeden önce resmin tüm pikseller için resmin piksel ortalamasından çıkarak, görüntünün varyansına bölünmektedir. Bu modele verilen resmin 0-1 aralığına çekilmesini sağladığı için normalizasyon katmanının etkisi olmaması beklenmektedir. Bu modelin test edilmesi ile bu durum gözlemlenmiştir.

Veri düşürme(dropout) katmanı, ezberleme probleminin etkisinin azaltılmasını sağlayan bir katmandır. Bu katman bazı nöronların diğer nöronlara etkisini kaldırmayı sağlamaktadır. Böylece birbiriyle ilişkili olan nöronların birbirini etkilemesini engelleyerek, modelin ayrıştırılmasını kolaylaştırmaktadır. Şekil 3.6 'de veri düşürme katmanının nasıl uygulandığını örneklemektedir. Resim 3.6'de soldaki resim standart bir sinir ağı modelini göstermektedir. Sağdaki resimde ise veri düşürme katmanı uygulanan resimler X olarak işaretlenerek diğer nöronlarla bağlantısı çıkarılmıştır. Doğrusal olmayan sistemler için bazı nöronların işlevinin yok edilmesi eğitim(training) fazını kolaylaştırmanın yanı sıra daha iyi öğrenmesini sağlamaktadır.

Veri düşürme katmanının nesne sınıflandırma problemine etkisi bir diğer yayınımızda çalışılmıştır [50]. Bu çalışmada, %90 oranında veri düşürme uygulanmıştır.



Şekil 3.5: CNN model-4.



Şekil 3.6: Veri düşürme katmanı etkisi-NamHyuk Ahn[47].

Bu katman son tam bağlantılı katmandan sonra eklenmiştir. Sınıflandırma yapmadan önce, eğitim esnasında nöronların %90'nın etkisi kaldırılmıştır. Test aşamasında bu nöronların etkisi sistemin performansını etkilemektedir. Nöron sayısının azaltılması eğitim süresini de azaltan ve aynı zamanda ezberleme problemininin etkisini azaltan oldukça etkili bir katmandır. Bu katmanın eklenmesi sonucunda başarım %86 seviyelerine yükselmektedir.

Çizelge 3.1: Cifar-10 hafıza kullanımı ve eğitim süreleri.

Yöntem	Hafıza Miktarı	Eğitim Süresi	Başarım Oranı	Tahmin Etme Süresi
Lenet[73]	3.82MB	30 dakika	76.27	0.001ms
Net. In Net.[42]	6GB	1 saat 30 dakika	91.25	3 ms
VGG-19[34]	15.8GB	4 saat	93.53	5 ms
Nagadomi[46]	2GB	20 saat	94.190	15 ms
Önr. Model	3 GB	4 saat	85.9	0.7ms

Veri düşürme katmanının etkisini gözlemlemek için 3 ayrı test düzenlenmiştir. %30, %50 ve %90 oranında veri düşürme uygulanarak modeller denenmiştir. Bu denemeler sonucunda %90 oranında veri düşürme en iyi sonucu vermektedir. Tablo 3.3'de sonuçlar özetlenmiştir.

Çizelge 3.2: Veri düşürme katmanının etkileri.

Başarım Oranı	Düşürme Yüzdesi
%85.2	%30
%85.7	%50
%85.9	%90

Veri düşürme katmanı evrişimsel katman ile örnekleme katmanı arasında da kullanılabilmektedir. Tam bağlantılı katman sonrasında olduğu gibi temel amacı ezberleme problemini çözmektir. Evrişimsel katmanı sonrasında kullanılan veri düşürme katmanı özniteliklerin bazılarının eğitim aşamasında kullanılmamasını sağlamaktadır. İlk olarak birinci evrişimsel katman ile örnekleme katmanı arasına %50 olasılıkla veri düşürme uygulanan katman test edilmiştir. Bu durumda başarım %85.3'e düşmektedir. İkinci evrişimsel katman ile ikinci örnekleme katmanı arasına da %50 olasılıkla veri düşürme katmanı eklenmiştir. Bu durumda başarım oranı %84.4 olmuştur. Evrişimsel katmanların arasına eklenmiş olan veri düşürme katmanları modeli olumsuz etkilemektedir. Bu olumsuz etkinin ana sebebi rastgele atılan öznitelikler modelin performansını etkileyen öznitelikler olmasıdır. Performansı etkileyen özniteliklerin belli bir olasılıkla eğitim aşamasında kullanılmaması sistemin daha yanlış öğrenmesine sebep olabilmektedir. Bu sebeple, veri düşürme katmanı en son katmandan sonra olacak şekilde önerilmiştir.
Cifar-10 veriseti kullanılarak nesne sınıflandırma modeli çıkarılan bir çok model bulunmaktadır. Önerilen modeller içinde hafıza kullanımının az ve eğitim süresinin kısa olduğu uygulamalar bulunmaktadır. LeNet adı verilen[73] model, daha az hafıza kullanıp, eğitim süresinin daha az olduğu bir model önermektedir. Fakat bu

Çizelge 3.3: Veri düşürme testleri.

Başarım Oranı	Test
%85.3	conv1 ile pool1 arasında dropout
%84.4	conv2 ile pool2 arasında dropout

model bizim önerdiğimiz modellere göre daha az başarım elde etmiştir. Network In Network[42] modelinde ise hafıza kullanımı bizim modelimize göre iki kat daha fazladır. Hafıza kullanımı bu kadar fazla olan bir gömülü sistem teknolojisi, şu an çok kullanılabilir görünmemektedir. VGG-19[34] modelinde hem eğitim süresi hem de hafıza kullanımı çok fazladır. Nagadomi[46] ise daha az hafıza kullanmasına rağmen eğitim süresi bizim önerdiğimiz modele göre 5 kat daha fazladır. Gerçek zamanlı bir kamera sisteminde 30 fps - 60 fps aralığında görüntü alınabilmektedir. Bu sebeple bu görüntülerin maksimum 16.67 ms içersinde sınıflandırılması gerekmektedir. Nagatomi modeli sınır değerlerinde olması sebebiyle gerçek zamanlı bir sistemde görüntüde bir kareye yakın gecikme meydana getireceği için, gerçek zamanlı sistemlerde kullanılması pek mümkün görünmemektedir. Sınıflandırma işleminin oldukça hızlı yapılabilmesinin bir diğer avantajı da modele verilmesi beklenen görüntünün ön işleme yöntemleriyle iyileştirilmesi mümkün olabilecektir.

4.TARTIŞMA

4.1.Hedef Takip Probleminin Tanımı ve Motivasyon

Askeri sistemlerde, görüntüde bulunan herhangi bir nesnenin(araba,insna hayvan vs.) takip edilebilmesi kritik bir öneme sahiptir. Nesnelerin takip edilmesi, iki ayrı bakış açısıyla incelenebilmektedir.

- Sistemde bulunan kamera bir gimbal yardımıyla hızlı bir şekilde döndürülebiliyorsa , görüntünün belirlenen bir bölgesinde(genellikle de görüntünün ortasında) belirli bir büyüklükte bir pencere içerisinde bulunan herhangi bir nesne seçilir ve bu nesne gerektiği takdirde gimbal de döndürülerek takip edilmeye çalışılmaktadır. Bu tür problemler, daha çok havadan takip etme sistemlerinde bulunmaktadır. Uçak, helikopter gibi bir hava aracına takılan kamera, bir gimbal yardımıyla bakış hattı kontrol edilerek yerde belirlenen hedef noktalarında bulunan nesnelerin takip edilme problemidir. Bu tür problemler için genellikle bölütleme(segmentation) tabanlı korelasyon filtreler kullanılmaktadır [65].

- Bir diğer yöntem ise kameranın çok fazla hareket etmeyip genellikle sabit bir noktadan görüntü alan sistemlerde hedefin takip edilmesidir. Nesnenin boyutlarının sürekli değişiyor olması ve nesnenin hızlı hareket etmesi bu tür sistemlerin ana sorunudur. Her iki problem için de ortak olan ve yapılması gereken işlem takip edilmek istenen nesnenin hızının tahmin edilebilmesidir. Hız piksel/kare türünde hesaplanması yeterli olacaktır, bir başka deyişle bir sonraki karede hareket eden nesnenin hangi bölgede aranması gerektiğidir.

Sabit kamera sistemlerinde, hızlı hareket eden hedeflerin takibinde sorunlar yaşanmaktadır. Gimbal gibi bir sistemle kamera hareket ettirilemediği için bir anda hızlı hareket eden nesneler hesaplanan bölgeden çıktığı için kaybedilebilmektedir.

4.2.Çalışmanın Amacı

Bu çalışmadaki amaç, sabit bir kamera kullanılarak hedef takip probleminin önemli problemlerinden biri olan hızlı hareket eden nesnelerin takibi için Evrişimsel sinir ağları kullanarak gömülü bir sistem üzerinde bu problemi çözmektir.
Hedef takip uygulamalarında nesnenin bir sonraki karede ne kadar hareket edebileceğini tahmin etmek gerekmektedir. Bazı çalışmalar nesnenin her karede hızını hesaplayarak bir sonraki karede hangi bölgelerde arayacağını kestirmeyi uygun bulmaktadır[13]. Bu tür uygulamaların en önemli problemi nesne hızlı hareket etmeye başlayınca yeni karede arayacağı pencere oldukça büyümektedir ve yeni karede hedefin bulunma süresi uzamaktadır, bu sebeple gerçek zamanlı sistemlerde bu tür sorunlarla oldukça sık karşılaşılmaktadır.

Bir diğer yöntem ise takip edilmek istenen nesnenin, hızını limitleyerek her yeni karede sadece bu hıza göre sabit bir alanda nesneyi aramaktır[31]. Burada ise limitlenen değer çok spesifik olduğu için bir videoda çalışan algoritma diğer bir mimaride çalışamaz hale gelebilmektedir.

4.3.Deneysel Çalışmalar

4.3.1 Veriseti Örnekleri

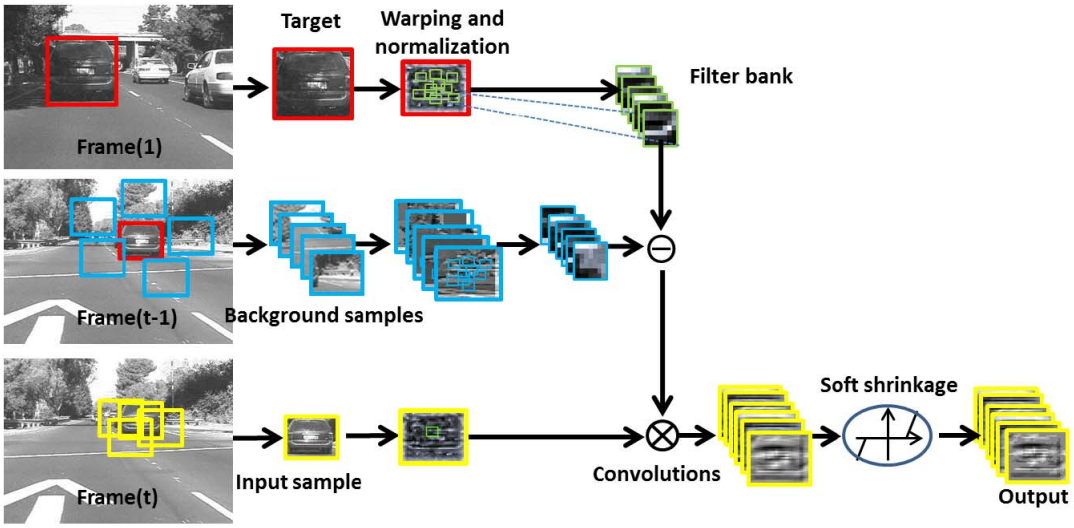
Hedef takip uygulamaları için bir çok veriseti bulunmaktadır. Bunların en sık kullanılanlarında biri 'Visual Tracker Benchmark' adı verilen bir çalışmadır. Bu çalışmada, ground-truth yani hedefin her karede bulunduğu piksel konumları kaydedilmiştir [74]. Bu verisetinde hedef takip uygulamalarında karşılaşılan tüm problemlerle ilgili videolar bulunmaktadır. Bu problemlerin başlıcaları aşağıda kısaca özetlenmiştir.

- Parlaklık veya ışığın geldiği yönün değişiminden kaynaklanan görüntü bozulmaları: IV
- Ölçek Değişimi, kameraya takip edilen nesnenin yaklaştırılıp uzaklaştırılmasından kaynaklanan, nesnenin boyutlarının değişmesi : SV
- Engelleme, hedef olark seçilen nesnenin başka bir nesne tarafından önüne geçilerek kamerada görüntüsünün kısmi olarak veya tamamen kaybolması : OCC
- Nesnenin hareket etmesinden kaynaklanan görüntüsünde şeklinin değişmesi : DEF
- Nesnelerin hareket etmesinden kaynaklanan görüntülerinde oluşan hayalet görüntü şeklinde gürültüler : MB
- Nesnelerin hızlı hareket etmesinden dolayı hedefin aranan bölgeden çıkması : FM

Bir başka verisetinde de birden fazla nesne takibi de içeren örnek video setleri paylaşılmıştır[8].
Bu çalışmada Visual Tracker Benchmark [74] datasetinde bulunan Basketball videosu kullanılmıştır. Bu videoda seçilen bir oyuncu videonun sonuna kadar takip edilebilmesi için bir çalışma yapılmıştır.

4.3.2 Seçilen Verisetinde Bulunan Problemler

Zhang ve arkadaşları tarafından önerilen Evrişimsel sinir ağları ile eğitim yapılmadan nesne takip etme algoritmasında ilk olarak takip edilmek istenen nesnenin öznitelikleri çıkarılmaktadır. Bu öznitelikler, makale içerisinde ön plan(foreground) filtreleri olarak tanımlanmıştır [31]. Bu filtrelerin haricinde arka plan(background) filtrelerini nesnenin her karede nerede bulunduğunu kestirmek için kullanmaktadır. Bu makalede kullanılan algoritma Resim 4.1'de özetlenmiştir.



Resim 4.1: Bu resim Robust Visual Tracking via Convolutional Networks Without Training[31] makalesinden alıntılanmıştır.

Her yeni gelen kare bir önceki karenin arka plan ve ön plan özniteliklerini kullanarak yeni karede bulunan hedefin yeri belirlenmektedir. Bu mimaride kullanılan evrişimsel sinir ağı sadece bir konvolusyon operasyonundan ibarettir. Konvolusyon operatörü bir nevi korolasyon filtresi olarak kullanılmaktadır. Gelen karede seçilen özniteliklere(ön plan - arka plan) en çok benzeyen bölge hedefin yeni yerini belirlemektedir.
Bu yöntemde hedefin yeni yeri sabit bir bölge içerisinde aranmaktadır, bu sebeple hızlı hareket eden nesnelerde bu algoritma hata yapabilmektedir. Ayrıca her yeni karede öznitelikler güncellendiği için engelleme(OCC) gibi durumlarda engelleyen nesnenin öznitelikleri de algoritmayı etkilemektedir. Bu etki, takip performansını olumsuz etkilemektedir. Engelleme durumlarında da önerilen algoritmanın[31] hata yaptığı gözlemlenmiştir.

4.3.3 CNT Algoritmasının MATLAB ortamında Kodlanması

Makalede bahsedilen çalışma Matlab2016a programı ile implemente edilmiştir[31]. Bu algoritmanın engelleme ve hızlı hareket sebebiyle çalışmadığı gözlemlenmiştir. Resim 4.2'de hedefin hızlı hareket etmesinden dolayı önerilen çalışma bir yerden sonra parkeyi takip etmeye başlamıştır. Bu aşamadan sonra hedef yakın konumunda aranmaya devam edildiği için tekrar eski hedefini önerilen algoritma bulamamaktadır.
Bir başka verisetinde(Coke) engelleme problemi kaynaklı nesnenin takibi kaybedilmektedir. Resim 4.3'te bu problemin sonucu gözlemlenebilmektedir.
Bu çalışmalar sonucunda önerilen hedef takip algoritmasının en temel problemi takip edilmek istenen nesnenin hızlı hareket etmesidir. Hızlı hareket eden nesnelerde bu algoritma hedefin tekrar bulmasını bile engelleyecek duruma sokmaktadır. Engelleme problemi içinse belli bir bölgenin içerisinde kalan hareketler yapıldığında nesne yeniden takip edilebilmektedir.
Sonuç olarak önerilen algoritma engelleme problemlerinde yeniden toparlayabilmektedir, bu sebeple hızlı hareket eden nesne problemine odaklanılmıştır. Bu al-

Resim 4.2: CNT algoritması çıktısı

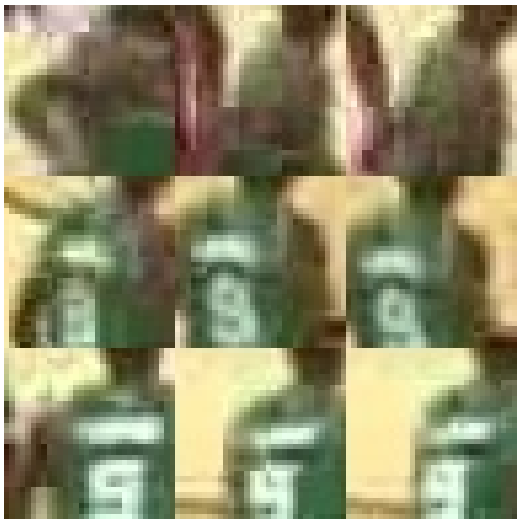
Resim 4.3: CNT algoritması çıktısı

Resim 4.4: Engelleme sonrası algoritmanın nesneyi yeniden bulması

goritmada sabit bir bölge içerisinde nesnenin yeni yeri aranmaktadır. Bu tez kapsamında girdi olarak verilen resimden bir sonraki framede ne kadar piksel hareket edebileceğinin tahmin edilmesi problemi için, evrişimsel sinir ağları yöntemiyle eğitilip eğitilemeyeceği araştırılmıştır.
Tez kapsamında önerilen modelde sınıflandırma sonucu 0-2-4-6-8-10-12-14-16-18-20 değerlerinden biri olmalıdır. Toplamda 11 adet sınıfımız bulunmaktadır. Cifar-10 Nesne tanıma problemine oldukça benzeyen bir model olduğu için Cifar-10 veriseti ile kullanılan modeller bu çalışmada kullanılmaya devam edilmiştir.

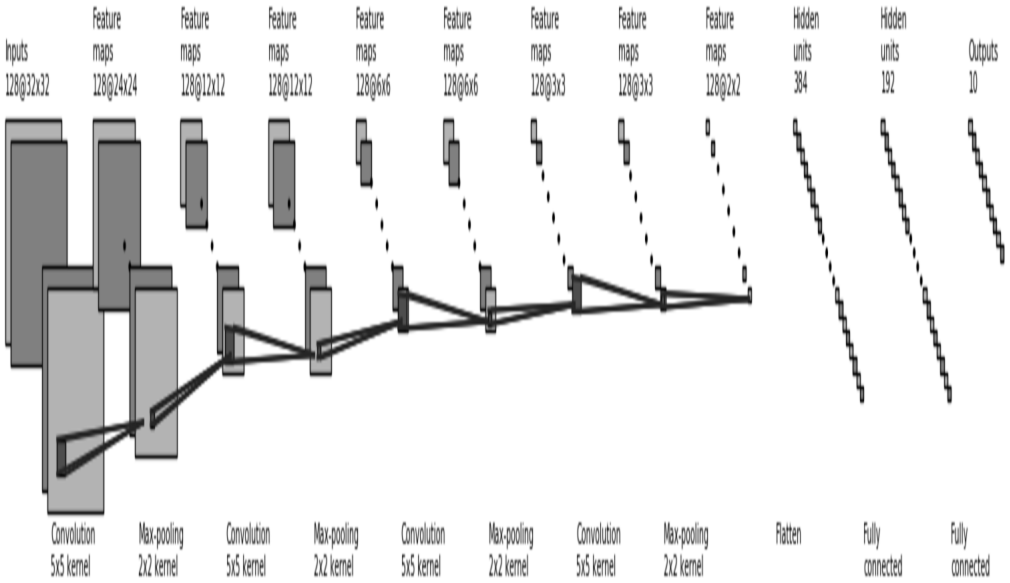
4.3.4 Basketbol Veriseti ile Evrişimsel Sinir Ağlarının Eğitilmesi

Basketbol verisetinde hızlı hareket eden oyuncunun takip edilebilmesi için, ground truth verilerinden bir sonraki karede kaç piksel hareket ettiği hesaplanabilmektedir. Verisetinden CNN modeline verilmesi planlanan resimler orjinal resimden kesilerek oluşturulmuştur. Bu kesilmiş resimler CNN kullanıldığı için kare şeklinde olması gerekmektedir. Bu sebeple mimariye nxn'lik parçalara bölünerek resimler verilecektir. Resimler kesilirken ground truth verisinde verilen üst-sol koordinattan başlanarak girdi resimleri üretilmiştir.
Mimaride ilk olarak Cifar-10 verisetinde olduğu gibi 32x32'lik parçalara kesilerek test edilmiştir. Verilen resimler üzerinde ölçek değiştirme(scale) gibi önişleme yöntemleri kullanılmamıştır. Bu veriler üzerinde mimariye verilmeden önce normalizasyon işlemi uygulanarak görüntüler -1 ile 1 aralığına map edilmiştir. Bu işlem ezberleme probleminin önüne geçilmesini sağlamıştır. 32x32x3'lik input resimler incelendiğinde takip edilmek istenen nesnelerin dışında da başka nesnelerin de olduğu gözlemlenmiştir.



Resim 4.5: Basketbol verisetinden kesilen resim örnekleri

İlk olarak dört tane konvolusyon katmanı ve arkasında 2 tam bağlantılı katman kullanılmıştır. Bu durumda başarım oranı %20'nin üzerine çıkarılamamıştır. Bu mimarinin esas probleminin önerilen mimariden çok resimlerle ilgili olduğu düşünülmektedir. Mimariye verilen resimlerin boyutları küçük olup başka arka plan detayları da içerdiği için doğru bir şekilde sınıflandırma yapamadığına karar verilmiştir.



Şekil 4.1: Tezde önerilen model-1.

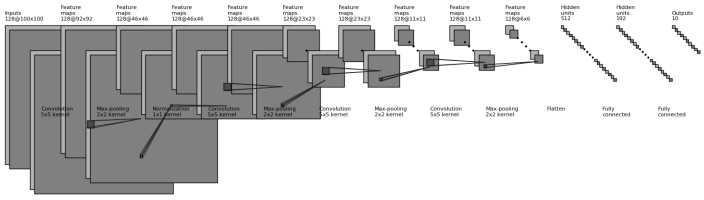
Resimlerin boyutları 100x100x3 boyutuna getirilmiştir, bu durumda da kullanılan memory 3 GB sınırını aşmaktadır. Burada yapılabilecek 2 farklı yöntem bulunmaktadır. Birincisi CNN mimarisinin katman sayısını(Konvolusyon + Tam Bağlantılı Katman) azaltmak, diğeri ise resimleri renkli olarak vermek yerine gri seviyeli olarak vererek tüm katmanlardaki hafıza kullanımını 3te birine düşürmektir. Katman sayısının azaltımlması başarım oranını da düşüren bir yöntem olduğu için bu yöntem yerine resimleri gri seviyeli olarak verme yöntemi tercih edilmiştir.

Bu durumda başarım oranı %30 seviyelerine çıktığı gözlemlenmiştir. Bu verisetinden örnek resimlerin bir kısmı Resim 4.6'de verilmiştir.

Resim 4.6'de verilen örnek resimlerde görülebileceği gibi önplan verilerinin haricinde oldukça baskın bir şekilde arka plan görüntüsü bulunmaktadır. Mimarinin öğrenirken arka plan verilerini de kullanacağı için, görüntülerin birbirinden ayrılması zorlaşmaktadır. Bu sebeple, modelin daha fazla hata yaptığı düşünülmüştür.

Girdi resimlerinin direk kesilip kullanılması yerine takip edilmek istenen nesnenin resim içinde kesilerek 100x100 lük resme ölçeklenmesi(scale) ile 100x100 lük resimler elde edilmiştir. Bu durumda görüntü üzerinde takip edilmek istenen nesneler enine veya boyuna genişleyebilir. Bu da nesnenin takip performanısını oldukça etkileyebilecek bir sorun oluşturmaktadır fakat nesnenin hızını doğru bir şekilde tespit edilmesini CNN mimarisinde kullanırken hedef takip algoritmasında gerçek resim kullanılabilir. Resim 4.6'de bulunan resimlerin nesnenin bulunduğu bölgede yeniden ölçeklendirilmiş hali Resim 4.7'de verilmiştir.

Yeniden ölçeklendirilen resimler, arka plan detaylarından kendilerini soyutlamış durumda olmalarına rağmen modelin başarım oranı %34 seviyelerinde kalmıştır.



Şekil 4.2: Tezde önerilen model-2.

Resim 4.6: Basketbol verisetinin 100x100 lük resimlere kesilmesi

Resim 4.7: Basketbol verisetinin nesne konumlarına göre yeniden ölçeklendirilmiş halleri

CNN mimarisinin hafıza gereksinimleri(maksimum 3 GB) sebebiyle katman sayısı artırılamamıştır. SVM yardımıyla CNN mimarisinin başarım oranı %7 oranında artırılmasına rağmen bu modeller sistem içerisinde kullanılamamıştır. OpenCV kütüphanesi ile SVM uyarlanmasının sınıflandırma sonucunu üretmesi yaklaşık olarak 250 ms'lik bir süre gerektirmektedir. Bu durumda da mimarinin maksimum 4 fps ile nesne takip etmesi anlamına gelmektedir. Gömülü bir sistemde 4 fps ile nesne takip uygulamaları kullanılabilir bir sistem olmaktan çıkarmaktadır.

Tanınan nesnenin takip edilme çalışması bu tez kapsamında ön araştırma çalışması olarak yapılmıştır. Bu çalışmaya devam edilmesi planlanmaktadır. Bu çalışmanın devamı için hafıza gereksinimleri düşünmeden VGG-16[33] gibi modeller kullanılarak başarımı yeniden gözden geçirmek faydalı bir çalışma olacaktır. Bu çalışmanın sonunda başarılı bir sonuç elde edilirse VGG-16 modelinden tam bağlantılı katmanlar çıkarılarak daha az bir hafıza ile VGG-16 modelinin bir varyasyonu üretilebilir.

Zhang ve arkadaşlarının [31] önerdiği model hızlı hareket eden nesnelerin takibi problemini karşılayamamaktadır. Önerilen CNN mimarisi ile Zhang ve arkadaşlarının [31] önerdiği model birleştirilerek hibrid bir model ortaya çıkarılabilir. Bunun için gelen her yeni kareden takip edilmek istenen nesnenin hızı tahmin edilip(yavaş ve hızlı olarak), hızlı hareket eden nesnelerin takibi sırasında CNN ile önerilen model kullanılıp yavaş hareket ettiği bölgelerde Zhang'ın önerdiği mimari kullanılabilir. Bu sayede çalışmanın performansının artması öngörülmektedir.

5.SONUÇ VE ÖNERİLER

Nesne tanıma problemlerinde Cifar-10 verisetinde olduğu gibi CNN mimarileri kullanılarak çok hızlı ve daha az hafıza kullanılarak bu problem başarılı bir şekilde çözülmektedir. Nesnelerin görüntüleri birbirine benzemediği için CNN mimarileri resimlerdeki nesneleri birbirinden rahatlıkla ayırabilmektedir. Nesnelerin öznitelikleri birbirinden ayrılabilir olması nesne tanıma problemlerinin CNN mimarilerine uyarlanmasını kolaylaştırmaktadır. CNN mimarilerinin, ihtiyaç duyduğu öznitelikleri otomatik olarak çıkarması bu mimarilerin uygulanabilirliğini artırmaktadır.

Çizelge 3.1'da verilen sonuçlar incelendiğinde önerilen modeli optimum bir model olduğu gözlemlenecektir. LeNet[73] en basit modellerden biri olup başarımı oldukça düşüktür. Bu modelin eğitim süresi ve tahmin etme süresi oldukça düşük olamsına rağmen başarım oranı %76.27 de kalmıştır, fakat kullandığı model hafıza kullanımına göre ilk sıradır. Nagadomi'nin önerdiği model[46] bizim önerdiğimiz modelden daha az hafıza kullanmasına rağmen eğitim süresi ve tahmin etme süresi çok uzundur. 15 ms içerisinde sınıflandırma sonucu vermesi modelin bir kamera ile gerçek zamanlı olarak kullanılmasını imkansız hale getirmektedir. VGG-19 [34] ve Network In Network [42] modelleri hafıza gereksinimleri oldukça fazla olan modeller olup gömülü bir sistem üzerinde şu anki teknolojilerle kullanılması imkansız görünmektedir. Önerilen modelin hafıza tüketimine ve doğruluk oranına göre diğer modellere göre en optimum çözüm olarak sunulmuştur.

Nesne tanıma problemi için SVM kullanımı %7 oranında ekstra bir başarım sağlamasına rağmen bu modelin eğitilmesi ve sınıflandırma sonucunu üretmesi yaklaşık olarak 250 mili saniye ekstra bir zamana ihtiyaç duymaktadır. Bu sebeple önerilen modellerin hiçbirinde SVM ve CNN mimarilerinin beraber kullanılması öngörülmemiştir.

Nesne tanıma probleminde oldukça başarılı sonuç veren CNN mimarileri nesne takibi algoritmalarında, nesnenin hız kestirimi problemi için kabul edilebilir oranda başarım elde edilememiştir. Nesne takibi problemini, CNN mimarileri kullanılarak çözülmesi için verilen görüntülerin birbirinden ayırt edilebilir olması gerekmektedir. Nesne takibi probleminde verilen resimler birbirine çok benzediği için takip edilmek istenen nesnenin hareketinin modellenmesi başarılı bir şekilde uygulanamamış olabilir. Bu problem üzerinde ön bir çalışma yapılmış olup ilerleyen zamanlarda bu konuda çalışmalarımız devam edecektir.

Bu hareketin modellenmesi katman sayısı artılarak başarılı bir şekilde yapılabilir, fakat bu durumda da sistemin gömülü bir sistem üzerinde çalıştırılması mümkün olamamaktadır. Görüntüleri modele doğrudan sadece ön işleme yöntemleri kullanarak modele beslemek yerine bir sonraki kare ile şu anki karenin farkı alınarak bu görüntünün modeli eğitmek için kullanılması denenebilir. Bu görüntü aslında bir önceki kareye göre ne kadar hareket ettiğini ifade eden bir görüntü olmaktadır. Bu görüntü kullanılarak mimari eğitilirse daha başarılı sonuçların elde edilebileceği

öngörülmektedir. Bu tezin devamı için VGG-16[33], ResNet-54[55] gibi önceden birçok problem için kullanılmış bir model üzerinde hız kestirimi yapılarak bu problemin CNN ile çözülmeye uygun bir problem olup olmadığı belirlenebilir. Eğer bu modellerden biri başarılı(minimum %90 oranında başarım elde edilmesi) sonuç verirse bu modelde kullanılan hafıza yani katman sayısı azaltılarak çalışmanın devamlılığı sağlanabilir. LSTM[20] gibi hafıza birimi olan modellerin tercih edilmemesinin sebebi, gömülü bir sistem üzerinde önerilen mimariyi kullanabilmektir. Gömülü sistemler için en çok hafıza tüketen katman tam bağlantılı katmanlardırç Bu katmanın modelden çıkarılarak daha az hafıza ve daha fazla katmanlı bir mimari ile sınıflandırılan nesnenin takip edilmesi daha başarılı olabileceği öngörülmektedir.

KAYNAKLAR

[1] **A. M. Dean and G. M. Smith**. An evaluation of per-parcel land cover mapping using maximum likelihood class probabilities. *International Journal of Remote Sensing*, 24(14):2905–2920, 2003.

[2] **A. R. Hanson, M. Marengoni, H. Schultz, F. Stolle, E. M. Riseman**. Ascender ii: a framework for reconstruction of scenes from aerial images. *Workshop Ascona 2001*, 2001.

[3] **Abadi, Martín and Barham, Paul and Chen, Jianmin and Chen, Zhifeng and Davis, Andy and Dean, Jeffrey and Devin, Matthieu and Ghemawat, Sanjay and Irving, Geoffrey and Isard, Michael and Kudlur, Manjunath and Levenberg, Josh and Monga, Rajat and Moore, Sherry and Murray, Derek G. and Steiner, Benoit and Tucker, Paul and Vasudevan, Vijay and Warden, Pete and Wicke, Martin and Yu, Yuan and Zheng, Xiaoqiang**. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA, 2016. USENIX Association.

[4] **Alex H. Williams**. http://alexhwilliams.info/itsneuronalblog/2016/03/27/pca/, March 2016.

[5] **Alex Krizhevsky and Sutskever, Ilya and Hinton, Geoffrey E**. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[6] **Alex Krizhevsky and Vinod Nair and Geoffrey Hinton**. Cifar-10 (canadian institute for advanced research).

[7] **Andrej Karpathy**. http://cs231n.github.io/convolutional-networks/, August 2018.

[8] **Anton Milan and Laura Leal-Taixé and Ian D. Reid and Stefan Roth and Konrad Schindler**. MOT16: A benchmark for multi-object tracking. *CoRR*, abs/1603.00831, 2016.

[9] **Blaschke, Thomas and Burnett, Charles and Pekkarinen, Anssi**. *Image Segmentation Methods for Object-based Analysis and Classification*, pages 211–236. Springer Netherlands, Dordrecht, 2004.

[10] **Brown, M. and Lowe, D. G.** Recognising panoramas. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 1218–, Washington, DC, USA, 2003. IEEE Computer Society.

[11] **Chris Harris and Mike Stephens**. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[12] **Comaniciu, Dorin**. Bayesian kernel tracking. In Luc Van Gool, editor, *Pattern Recognition*, pages 438–445, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[13] **David S. Bolme and J. Ross and Beveridge Bruce and A. Draper and Yui Man Lui**. Visual object tracking using adaptive correlation filters.

[14] **Dongmei Han and Qigang Liu and Weiguo Fan**. A new image classification method using cnn transfer learning and web data augmentation. *Expert Systems with Applications*, 95:43 – 56, 2018.

[15] **Gao, Jin and Ling, Haibin and Hu, Weiming and Xing, Junliang**. Transfer learning based visual tracking with gaussian processes regression. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 188–203, Cham, 2014. Springer International Publishing.

[16] **Gaurav Mittal**. https://www.slideshare.net/gauravmittal68/convolutional-neural-networks-cnn, November 2015.

[17] **H. Liu and X. Hou**. Moving detection research of background frame difference based on gaussian model. In *2012 International Conference on Computer Science and Service System*, pages 258–261, Aug 2012.

[18] **Haixia Liu**. Sentiment analysis of citations using word2vec. *CoRR*, abs/1704.00177, 2017.

[19] **Hanxi Li and Yi Li and Fatih Porikli**. Deeptrack: Learning discriminative feature representations online for robust visual tracking. *CoRR*, abs/1503.00072, 2015.

[20] **Hochreiter, Sepp and Schmidhuber, Jürgen**. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[21] **Hosaka, Tadaaki and Kobayashi, Takumi and Otsu, Nobuyuki**. Object detection using background subtraction and foreground motion estimation. 3:9–20, 01 2011.

[22] **Hui Zou and J Zhu and S Rosset and T Hastie**. Multi-class adaboost. *Statistics and its Interface*, 2:349–360, 2009.

[23] **I. Kartika and Shahrizat Shaik Mohamed**. Frame differencing with post-processing techniques for moving object detection in outdoor environment. In *2011 IEEE 7th International Colloquium on Signal Processing and its Applications*, pages 172–176, March 2011.

[24] **J. Fan and W. Xu and Y. Wu and Y. Gong**. Human tracking using convolutional neural networks. *IEEE Transactions on Neural Networks*, 21(10):1610–1623, Oct 2010.

[25] **J. Kwon and K. M. Lee**. Visual tracking decomposition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1269–1276, June 2010.

[26] **Jack Valmadre and Luca Bertinetto and João F. Henriques and Andrea Vedaldi and Philip H. S. Torr**. End-to-end representation learning for correlation filter based tracking. *CoRR*, abs/1704.06036, 2017.

[27] **Jey Han Lau and Timothy Baldwin**. An empirical evaluation of doc2vec with practical insights into document embedding generation. *CoRR*, abs/1607.05368, 2016.

[28] **Jia, Yangqing and Shelhamer, Evan and Donahue, Jeff and Karayev, Sergey and Long, Jonathan and Girshick, Ross and Guadarrama, Sergio and Darrell, Trevor**. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[29] **João F. Henriques and Rui Caseiro and Pedro Martins and Jorge Batista**. High-speed tracking with kernelized correlation filters. *CoRR*, abs/1404.7584, 2014.

[30] **K. Mikolajczyk and C. Schmid**. Indexing based on scale invariant interest points. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 525–531 vol.1, July 2001.

[31] **Kaihua Zhang and Qingshan Liu and Yi Wu and Ming-Hsuan Yang**. Robust tracking via convolutional networks without learning. *CoRR*, abs/1501.04505, 2015.

[32] **Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun**. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014.

[33] **Karen Simonyan and Andrew Zisserman**. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[34] **Karen Simonyan and Andrew Zisserman**. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[35] **Kristen Grauman**. https://pdfs.semanticscholar.org/presentation/8e52/690189322323845e9e62

[36] **L.D. Harmon and M.K. Khan and Richard Lasch and P.F. Ramig**. Machine identification of human faces. *Pattern Recognition*, 13(2):97 – 110, 1981.

[37] **Levi, Giorgio and Sirovich, Franco**. Structural descriptions of fingerprint images. 4:327–355, 12 1972.

[38] **Lowe, David G.** Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[39] **M. A. Turk and A. P. Pentland**. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 586–591, Jun 1991.

[40] **M. P. Patel and S. K. Parmar**. Moving object detection with moving background using optic flow. In *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*, pages 1–6, May 2014.

[41] **Mark Hughes and Irene Li and Spyros Kotoulas and Toyotaro Suzumura**. Medical text classification using convolutional neural networks. *CoRR*, abs/1704.06841, 2017.

[42] **Min Lin and Qiang Chen and Shuicheng Yan**. Network in network. *CoRR*, abs/1312.4400, 2013.

[43] **Ming Liang and Xiaolin Hu**. Recurrent convolutional neural network for object recognition. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3367–3375, 2015.

[44] **Mircea Serban Pavel and Hannes Schulz and Sven Behnke**. Object class segmentation of rgb-d video using recurrent convolutional neural networks. *Neural Networks*, 88:105 – 113, 2017.

[45] **N. Dalal and B. Triggs**. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005.

[46] **Nagadomi**. Nagadomi architecture, August 2018.

[47] **NamHyuk Ahn**. https://www.slideshare.net/nmhkahn/case-study-of-convolutional-neural-network-61556303, May 2016.

[48] **NVIDIA**. Pascal architecture in nvidia.

[49] **Peemen, Maurice and Mesman, B and Corporaal, Henk**. Efficiency optimization of trainable feature extractors for a consumer platform. 6915:293–304, 08 2011.

[50] **R. Caner Çalık and M. Fatih Demirci**. Cifar-10 image classification with convolutional neural network for embedded systems. *IEEE International Conference on Computer Systems and Applications AICCSA(Not Published)*, 2018.

[51] **R. Caner Çalık and M. Fatih Demirci**. In embedded systems image classification with convolutional neural network. *IEEE Xplore*, 2018.

[52] **R. M. Haralick and K. Shanmugam and I. Dinstein**. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, Nov 1973.

[53] **Ross B. Girshick**. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[54] **Ross B. Girshick and Jeff Donahue and Trevor Darrell and Jitendra Malik**. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[55] **Saining Xie and Ross B. Girshick and Piotr Dollár and Zhuowen Tu and Kaiming He**. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016.

[56] **Salari, V and Sethi, Ishwar**. Feature point correspondence in the presence of occlusion. 12:87 – 91, 02 1990.

[57] **Scherer, Dominik and Müller, Andreas and Behnke, Sven**. *Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition*, pages 92–101. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[58] **Scherer, Dominik and Müller, Andreas and Behnke, Sven**. Evaluation of pooling operations in convolutional architectures for object recognition. In *Proceedings of the 20th International Conference on Artificial Neural Networks: Part III*, ICANN'10, pages 92–101, Berlin, Heidelberg, 2010. Springer-Verlag.

[59] **Sebastian Ruder**. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[60] **Seunghoon Hong and Tackgeun You and Suha Kwak and Bohyung Han**. Online tracking by learning discriminative saliency map with convolutional neural network. *CoRR*, abs/1502.06796, 2015.

[61] **Shuo Liu and Zheng Liu**. Multi-channel cnn-based object detection for enhanced situation awareness. *CoRR*, abs/1712.00075, 2017.

[62] **Simonyan, Karen and Vedaldi, Andrea and Zisserman, Andrew**. Deep fisher networks for large-scale image classification. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 163–171. Curran Associates, Inc., 2013.

[63] **Sunil Ray**. https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/, September 2017.

[64] **Te-Won Lee and M. S. Lewicki**. Unsupervised image classification, segmentation, and enhancement using ica mixture models. *IEEE Transactions on Image Processing*, 11(3):270–279, Mar 2002.

[65] **Tunali, Emre and Oz, Sinan and Eral, Mustafa**. Enhancing correlation filter based trackers with size adaptivity and drift prevention. pages 472–480, 01 2018.

[66] **Vatsavai, Ranga Raju and Bright, Eddie and Varun, Chandola and Budhendra, Bhaduri and Cheriyadat, Anil and Grasser, Jordan**. Machine learning approaches for high-resolution urban land cover classification: A comparative study. In *Proceedings of the 2Nd International Conference on Computing for Geospatial Research & Applications*, COM.Geo '11, pages 11:1–11:10, New York, NY, USA, 2011. ACM.

[67] **Veenman, Cor J. and Reinders, Marcel J. T. and Backer, Eric**. Resolving motion correspondence for densely moving points. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(1):54–72, January 2001.

[68] **Wang, Jiang and Yang, Yi and Mao, Junhua and Huang, Zhiheng and Huang, Chang and Xu, Wei**. Cnn-rnn: A unified framework for multi-label image classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[69] **Wang, Naiyan and Yeung, Dit-Yan**. Learning a deep compact image representation for visual tracking. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 809–817. Curran Associates, Inc., 2013.

[70] **Wright, J. and Yang, A.Y. and Ganesh, A. and Sastry, S.S. and Ma, Y.** Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009. cited By 5586.

[71] **Y. Wei and W. Xia and M. Lin and J. Huang and B. Ni and J. Dong and Y. Zhao and S. Yan**. Hcp: A flexible cnn framework for multi-label image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1901–1907, Sept 2016.

[72] **Yan Ke and R. Sukthankar**. Pca-sift: a more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–506–II–513 Vol.2, June 2004.

[73] **Yann Lecun and Léon Bottou and Yoshua Bengio and Patrick Haffner**. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[74] **Yi Wu and Jongwoo Lim and Ming-Hsuan Yang**. Online object tracking: A benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[75] **Zhou, Xiangzeng and Xie, Lei and Zhang, Peng and Zhang, Yanning**. Online object tracking based on cnn with metropolis-hasting re-sampling. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, pages 1163–1166, New York, NY, USA, 2015. ACM.

EKLER

EK 1

```
name: "CIFAR10_full_deploy"
layer {
  name: "data"
  type: "Input"
  top: "data"
  input_param { shape: { dim: 1 dim: 3 dim: 32 dim: 32 } }
}
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 32
    pad: 2
    kernel_size: 5
    stride: 1
  }
}
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
```

```
  name: "relu1"
  type: "ReLU"
  bottom: "pool1"
  top: "pool1"
}
layer {
  name: "norm1"
  type: "LRN"
  bottom: "pool1"
  top: "norm1"
  lrn_param {
    local_size: 3
    alpha: 5e-05
    beta: 0.75
    norm_region: WITHIN_CHANNEL
  }
}
layer {
  name: "conv2"
  type: "Convolution"
  bottom: "norm1"
  top: "conv2"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 32
    pad: 2
    kernel_size: 5
    stride: 1
  }
}
layer {
  name: "relu2"
  type: "ReLU"
  bottom: "conv2"
  top: "conv2"
}
layer {
  name: "pool2"
  type: "Pooling"
  bottom: "conv2"
  top: "pool2"
  pooling_param {
```

```
    pool: AVE
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "norm2"
  type: "LRN"
  bottom: "pool2"
  top: "norm2"
  lrn_param {
    local_size: 3
    alpha: 5e-05
    beta: 0.75
    norm_region: WITHIN_CHANNEL
  }
}
layer {
  name: "conv3"
  type: "Convolution"
  bottom: "norm2"
  top: "conv3"
  convolution_param {
    num_output: 64
    pad: 2
    kernel_size: 5
    stride: 1
  }
}
layer {
  name: "relu3"
  type: "ReLU"
  bottom: "conv3"
  top: "conv3"
}
layer {
  name: "pool3"
  type: "Pooling"
  bottom: "conv3"
  top: "pool3"
  pooling_param {
    pool: AVE
    kernel_size: 3
    stride: 2
  }
}
layer {
```

```
  name: "ip1"
  type: "InnerProduct"
  bottom: "pool3"
  top: "ip1"
  param {
    lr_mult: 1
    decay_mult: 250
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  inner_product_param {
    num_output: 10
  }
}
layer {
  name: "prob"
  type: "Softmax"
  bottom: "ip1"
  top: "prob"
}
```

EK 2- Tensorflow Basketbol Veriseti Mimarisi Kod Paylaşımı

```
def inference(images):
  """Build the CIFAR-10 model.

  Args:
    images: Images returned from distorted_inputs() or inputs().

  Returns:
    Logits.

  # conv1
  with tf.variable_scope('conv1') as scope:
    kernel = _variable_with_weight_decay('weights',
                                         shape=[3, 3, 1, 64],
                                         stddev=0.01,
                                         wd=0.0)
    conv = tf.nn.conv2d(images, kernel, [1, 1, 1, 1],
        padding='SAME')
    biases = _variable_on_cpu('biases', [64],
        tf.constant_initializer(0.0))
    pre_activation = tf.nn.bias_add(conv, biases)
    conv1 = tf.nn.relu(pre_activation, name=scope.name)
    _activation_summary(conv1)

  # pool1
  pool1 = tf.nn.max_pool(conv1, ksize=[1, 2, 2, 1],
                strides=[1, 2, 2, 1],
                padding='SAME', name='pool1')
  # norm1
  norm1 = tf.nn.lrn(pool1, 4, bias=1.0, alpha=0.001 / 9.0,
                                   beta=0.75, name='norm1')

  # conv2
  with tf.variable_scope('conv2') as scope:
    kernel = _variable_with_weight_decay('weights',
                shape=[3, 3, 64, 1024],
                       stddev=1e-2,
                       wd=0.0)
    conv = tf.nn.conv2d(norm1, kernel, [1, 1, 1, 1],
                padding='SAME')
    biases = _variable_on_cpu('biases', [1024],
                tf.constant_initializer(0.0))
    pre_activation = tf.nn.bias_add(conv, biases)
    conv2 = tf.nn.relu(pre_activation, name=scope.name)
    _activation_summary(conv2)
```

```
# norm2
#norm2 = tf.nn.lrn(conv2, 4, bias=1.0, alpha=0.001 / 9.0,
                                    beta=0.75, name='norm2')
# pool2
pool2 = tf.nn.max_pool(conv2, ksize=[1, 2, 2, 1],
                       strides=[1, 2, 2, 1], padding='SAME',
                       name='pool2')
# conv3
with tf.variable_scope('conv3') as scope:
  kernel = _variable_with_weight_decay('weights',
                                       shape=[3, 3, 1024, 2048],
                                       stddev=1e-2,
                                       wd=0.0)
  conv = tf.nn.conv2d(pool2, kernel, [1, 1, 1, 1], padding='SAME')
  biases = _variable_on_cpu('biases', [2048],
             tf.constant_initializer(0.0))
  pre_activation = tf.nn.bias_add(conv, biases)
  conv3 = tf.nn.relu(pre_activation, name=scope.name)
  _activation_summary(conv3)
# norm3
#norm3 = tf.nn.lrn(conv3, 4, bias=1.0, alpha=0.001 / 9.0,
                     beta=0.75,name='norm3')
# pool3
pool3 = tf.nn.max_pool(conv3, ksize=[1, 2, 2, 1],
             strides=[1, 2, 2, 1], padding='SAME', name='pool3')

#conv4
with tf.variable_scope('conv4') as scope:
  kernel = _variable_with_weight_decay('weights',
                                       shape=[3, 3, 2048, 256],
                                       stddev=1e-2,
                                       wd=0.0)
  conv = tf.nn.conv2d(pool3, kernel, [1, 1, 1, 1], padding='SAME')
  biases = _variable_on_cpu('biases', [256],
             tf.constant_initializer(0.0))
  pre_activation = tf.nn.bias_add(conv, biases)
  conv4 = tf.nn.relu(pre_activation, name=scope.name)
  _activation_summary(conv4)
# pool4
pool4 = tf.nn.max_pool(conv4, ksize=[1, 2, 2, 1],
                       strides=[1, 2, 2, 1], padding='SAME',
                       name='pool4')

#conv5
with tf.variable_scope('conv5') as scope:
  kernel = _variable_with_weight_decay('weights',
                                       shape=[3, 3, 256, 512],
```

```
                                        stddev=1e-2,
                                        wd=0.0)
  conv = tf.nn.conv2d(pool4, kernel, [1, 1, 1, 1],
              padding='SAME')
  biases = _variable_on_cpu('biases', [512],
              tf.constant_initializer(0.0))
  pre_activation = tf.nn.bias_add(conv, biases)
  conv5 = tf.nn.relu(pre_activation, name=scope.name)
  _activation_summary(conv5)
# pool5
pool5 = tf.nn.max_pool(conv5, ksize=[1, 2, 2, 1],
                       strides=[1, 2, 2, 1], padding='SAME',
                       name='pool5')

# local3
with tf.variable_scope('local3') as scope:
  reshape = tf.reshape(pool5, [FLAGS.batch_size, -1])
  dim = reshape.get_shape()[1].value
  weights = _variable_with_weight_decay('weights',
              shape=[dim, 512],
              stddev=0.04, wd=0.004)
  biases = _variable_on_cpu('biases', [512],
              tf.constant_initializer(0.0))
  local3 = tf.nn.relu(tf.matmul(reshape, weights) + biases,
              name=scope.name)
  _activation_summary(local3)

# local4
with tf.variable_scope('local4') as scope:
  weights = _variable_with_weight_decay('weights',
                      shape=[512, 192],
              stddev=0.04, wd=0.004)
  biases = _variable_on_cpu('biases', [192],
              tf.constant_initializer(0.0))
  local4 = tf.nn.relu(tf.matmul(local3, weights) + biases,
              name=scope.name)
  _activation_summary(local4)

with tf.variable_scope('softmax_linear') as scope:
  weights = _variable_with_weight_decay('weights',
              [192, NUM_CLASSES],
              stddev=1/192.0, wd=0.0)
  biases = _variable_on_cpu('biases', [NUM_CLASSES],
                            tf.constant_initializer(0.0))
  softmax_linear = tf.add(tf.matmul(local4, weights), biases,
              name=scope.name)
  _activation_summary(softmax_linear)
```

```
return softmax_linear
```

EK 3- Ubuntu üzerinde Caffe Kurulum Aşamaları

Ubuntu 17.04 ve sonrası için önceden derlenmiş Caffe sürümleri kolaylıkla kullanılabilmektedir. İşlemci(CPU) tabnalı kullanılan versiyonu *'sudo apt install caffe−cpu'* kodu ile işletim sistemine kolaylıkla kurulabilir. Aynı şekilde *'sudo apt install caffe−cuda'* kodu ile de NVIDIA ekran kartı ile uyumlu bir şekilde çalışabilen versiyonu bilgisayarlara kurulabilmektedir.
Ubuntu 17.04 öncesinde aşağıdaki adımlar sırasıyla uygulanmalıdır.

- 'sudo apt-get install libprotobuf-dev libleveldb-dev libsnappy-dev libopencv-dev libhdf5-serial-dev protobuf-compiler sudo apt-get install –no-install-recommends libboost-all-dev' komutları ile Caffe kütüphanesinin ihtiyaç duyduğu paketler kurulmalıdır.

- Eğer GPU tabanlı altyapı kullanılmak isteniyorsa CUDA kütüphanesi NVIDIA'nın sitesinde veya apt-get komutu ile kurulabilmektedir. Fakat 16.04 kullanılıyorsa CUDA 8 versiyonunun indirilmesi gerekmektedir.

- BLAS kütüphanesinin sisteme kurulmuş olması gerekmektedir. Bu paket apt-get install libatlas-base-dev komutu ile indirilebilir. Bu paket ATLAS adı verilen bir pakettir. Bu paket yerine OpenBLAS adı verilen bir başka paket program daha kullanılabilmektedir. Bu paket de sudo apt-get install libopenblas-dev komutu yardımıyla sisteme entegre edilebilmektedir. Eğer CPU versiyonu kullanılacaksa MKL adı verilen paket kullanılarak daha iyi bir CPU performansı elde edilebilmektedir.

- Python kurulumu gerekmemektedir. Fakat Caffe altyapısı Python kullanarak gerçeklenecekse bu kurulumun yapılması gerekmektedir.

- 16.04 kurulumu ile kurulu olarak gelmesi gereken paket programlar bulunmaktadır. Bazı versiyonlarında kurulu halde gelmemektedir. Son olarak 'sudo apt-get install libgflags-dev libgoogle-glog-dev liblmdb-dev' komutu yardımıyla bu paketlerin de kurulumu yapılmaktadır.

- `https://github.com/BVLC/caffe.git` githubtan son kod klonlanarak bilgisayara tüm hazır kodlar çekilmelidir.

- Makefile.config.example dosyası önceden tanımlı bir şekilde kurulum için hazır hale getirilmiştir.

- Eğer ATLAS kütüphanesi yerine MKL veya OpenBLAS kütüphanesi kullanılacaksa BLAS := mkl veya open olarak değiştirilmelidir.

- Python kullanılacaksa öncesinde Python klasörü altında for req in $(cat requirements.txt); do pip install $req; done komutu çağrılarak Python arayüzü için gereken kütüphaneler Python'a entegre edilmelidir.

- PYTHONPATH klasörü export komutu yardımıyla Caffe kullanmadan önce tanımlanmış olmalıdır. Eğer sürekli export komutu kullanılmak istenmiyorsa home klasörü altında .bashrc dosyasına

export PYTHONPATH=<caffe path>/caffe/distribute/python:$PYTHONPATH export PYTHONPATH=<caffe path>/caffe/python:$PYTHONPATH komutları eklenebilir.

- make all

  make test

  make runtest komutları yardımıyla Caffe kütüphanesi derlenebilmektedir.

- Sadece CPU tabanlı olarak Caffe kullanılacaksa `CPU_ONLY := 1` değişkeninin tanımındaki yorum kaldırılmalıdır.

Caffe derlenmesi sırasında hdf5 kütüphanesi ve cuda kütüphanesi ile ilgili bir hata ile karşılaşırsa

`LIBRARY_DIRS := $(PYTHON_LIB)` /usr/local/lib /usr/lib */usr/lib/x86_64 − linux − gnu/hdf5/serial* */usr/local/cuda/lib64*

olarak değiştirilmelidir.

EK 4 - Tensorflow - Ubuntu Kurulumu

Tensorflow kütüphanesi, Python arayüzünü kullanarak evrişimsel sinir ağlarını kodlayan bir hazır kod kümesidir. Tensorflow kütüphanesi hem Linux işletim sistemi hem de Windows üzerinde kolayca kurulumu yapılabilen bir kütüphanedir. Linux(Ubuntu,Centos,Fedora vb.) işletim sistemlerinde kurulumu `https://www.tensorflow.org/install/install_linux` bu link kullanılarak kolayca yapılabilmektedir. Tensorflow kütüphanesinin iki ayrı kullanımı bulunmaktadır. Bu kullanımların biri CPU yani işlemci kullanarak evrişimsel sinir ağları modelindeki işlem yükünün işlemci tarafından karşılandığı versiyonlarda kullanılmaktadır. NVIDIA ekran kartı bulunmayan sistemlerde tensorflow altyapısı işlemciler üzerinden yapılmaktadır.
Bir diğer yöntem ise GPU yani ekran kartı kullanarak evrişimsel sinir ağlarının gerektirdiği fonksiyonları implemente eden bir kütüphanedir. Ekran kartı hafızası kullanılarak matris işlemlerinin paralel olarak çalıştırılmasını sağlamaktadır. NVIDIA ekran kartı bulunan sistemlerde GPU versiyonunu kullanmak eğitim ve test aşamalarının ortalamada 50 kat daha hızlı çalışabilmesine olanak sağlamaktadır. Bu sebeple NVIDIA ekran kartı bulunan sistemlerde bu altyapının kullanılması tavsiye edilmektedir.
Tensorflow, ayrıca Keras arayüzü ile entegre çalışabilmektedir. Bu arayüz yardımıyla Keras kullanan modellerin uyarlanması kolaylıkla yapılabilmektedir. Bizim çalışmalarımızda bu altyapı kullanılmamıştır.

ÖZGEÇMİŞ

Ad-Soyad : Rasim Caner ÇALIK

Uyruğu : T.C.

Doğum Tarihi ve Yeri : 15/04/1990 KONYA

E-posta : rcanercalik@gmail.com

ÖĞRENİM DURUMU:

• **Lisans** : 2012, Hacettepe Üniversitesi, Mühendislik Fakültesi, Elektrik Elektornik Mühendisliği

MESLEKİ DENEYİM VE ÖDÜLLER:

Yıl	**Yer**	**Görev**
2012-2016	ASELSAN	Yazılım Mühendisi
,2016-2018	ASELSAN	Uzman Yazılım Mühendisi

YABANCI DİL: İngilizce, Almanca, Rusça

TEZDEN TÜRETİLEN YAYINLAR, SUNUMLAR VE PATENTLER:

- **Çalık. Rasim Caner**,Demirci. M.Fatih.,2018. Evrişimsel Sinir Ağları(CNN) ile Gömülü Sistemlerde Görüntü Sınıflandırma, SIU-2018, 15-18 May 2018 Çeşme, Türkiye

- **Çalık. Rasim Caner**,Demirci. M.Fatih.,2018. Cifar-10 Image Classification with Convolutional Neural Network For Embedded Systems, AICCSA-2018, 28 Oct - 02 Nov, Aqaba, Jordan