T.C.
NECMETTİN ERBAKAN NİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME ALGORİTMALARI KULLANILARAK GERÇEK ZAMANLI SİLAH TANIMA UYGULAMASI

Ali AKDAĞ

YÜKSEK LİSANS TEZİ

Endüstri Mühendisliği Anabilim Dalı

Kasım-2017
KONYA

TEZ KABUL VE ONAYI

Ali Akdağ tarafından hazırlanan "Derin Öğrenme Algoritmaları Kullanılarak Gerçek Zamanlı Silah Tanıma Uygulaması" adlı tez çalışması 24/11/2017 tarihinde aşağıdaki jüri tarafından oy birliği ile Necmettin Erbakan Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Yrd. Doç. Dr. Barış KOÇER	……..
Danışman Prof. Dr. Sabri KOÇER	……..
Üye Yrd. Doç. Dr. Mehmet HACIBEYOĞLU	……..

Yukarıdaki sonucu onaylarım.

Prof. Dr. Ahmet COŞKUN
FBE Müdürü

Bu tez çalışması Necmettin Erbakan Üniversitesi BAP Koordinatörlüğü tarafından 161319021 nolu proje ile desteklenmiştir.

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Ali AKDAĞ
Kasım - 2017

ÖZET

YÜKSEK LİSANS TEZİ

DERİN ÖĞRENME ALGORİTMALARI KULLANILARAK GERÇEK ZAMANLI SİLAH TANIMA UYGULAMASI

Ali AKDAĞ

Necmettin Erbakan Üniversitesi Fen Bilimleri Enstitüsü
Endüstri Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Sabri KOÇER

2017, 66 Sayfa

Jüri
Prof. Dr. Sabri KOÇER
Yrd. Doç. Dr. Barış KOÇER
Yrd. Doç. Dr. Mehmet HACIBEYOĞLU

GPU'lardaki donanımsal gelişmeler ve GPU'ların genel amaçlı programlanmasını sağlayan CUDA, görüntü işleme gibi yüksek derecede veri paralelliği gerektiren bazı alanlarda köklü değişiklikler yaşatmıştır. Özellikle bilgisayar görüsünde nesne sınıflandırma, nesne algılama, bölütleme gibi görevlerde etkileyici başarılar sergileyen derin öğrenme algoritmalarına olan ilgiyi artırmıştır. GPU ve derin öğrenme alanında yaşanan gelişmeler sayesinde kamera görüntülerinin gerçek zamanlı olarak, insan faktörüne ihtiyaç duymadan yorumlanmasında önemli başarılar elde edilmiştir.

Güvenliğin önemli olduğu yerlerde, kameralardan elde edilen görüntülerin mümkün olan en kısa zamanda yorumlanması önem arz etmektedir. Silahlı tehditlerin hızlı bir şekilde tespit edilmesi güvenliği daha yüksek düzeye çıkaracaktır. Bu işlemi gerçekleştirmek adına bu tez çalışmasında, NVIDIA firmasının CUDA destekli platformları ve derin öğrenme algoritmaları kullanılarak gerçek zamanlı silah tanıma uygulaması gerçekleştirilmiştir. Uygulamayı gerçekleştirmek için içerisinde silah nesnesi bulunan resimlerle veri seti oluşturulmuştur. Bu veri seti kullanılarak DetectNet ve YOLO derin sinir ağı modelleri eğitilmiştir. Test görüntülerinden elde edilen sonuçlar ve CUDA destekli bir gömülü sistem olan Jetson TX1 kartındaki çalışma performansı bakımından DetectNet modelinin YOLO modelinden daha başarılı olduğu görülmüştür.

Anahtar Kelimeler: cuda, derin öğrenme, derin sinir ağları, GPU, nesne algılama, silah algılama

ABSTRACT

MS THESIS

REAL-TIME HANDGUN DETECTION APPLICATION USING DEEP LEARNING ALGORITHMS

Ali AKDAĞ

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF NECMETTİN ERBAKAN UNIVERSITY
THE DEGREE OF MASTER OF SCIENCE / DOCTOR OF PHILOSOPHY IN INDUSTRIAL ENGINEERING

Advisor: Prof. Dr. Sabri KOÇER

2017, 66 Pages

Jury
Prof. Dr. Sabri KOÇER
Asst.Prof.Dr. Dr. Barış KOÇER
Asst.Prof.Dr. Mehmet HACIBEYOĞLU

The development of GPUs and CUDA, which allows general-purpose programming of GPUs, have undergone radical changes in some areas that require high degree of data parallel such as image processing. In particular, it has also increased the interest in deep learning algorithms that show impressive successes in tasks such as object classification, object detection, segmentation on a computer vision. Thanks to improvements in GPU and deep learning areas, important achievements have been achieved in interpreting camera images in real time, without the need for a human factor.

It is important to quickly interpret the images obtained from the cameras in places where safety is important. Rapid detection of armed threats will result in a higher level of security. In order to accomplish this task, in this thesis, real-time handgun detection was implemented using NVIDIA's CUDA-supported platforms and deep learning algorithms. In order to perform the application, a data set was created by using the pictures with the handgun object. Using this data set, DetectNet and YOLO deep neural network models were trained. The results from the test images and the performance on the Jetson TX1 card, an embedded system with CUDA support, were found to be the DetectNet model is more successful than the YOLO model.

Keywords: cuda, deep learning, deep neural network, GPU, handgun detection, object detection

ÖNSÖZ

Tez çalışmamı hazırlamam sırasında beni yönlendiren, yardım ve desteklerini esirgemeyen başta danışmanım Sayın Prof. Dr. Sabri KOÇER'e ve Bilgisayar Mühendisliği Bölümü hocalarıma sonsuz teşekkürlerimi sunarım.

Bu tezi hazırlamam sürecinde beni destekleyen arkadaşlarıma, maddi ve manevi desteklerini eksik etmeyen, hayatımın her safhasında yanımda olan aileme tüm kalbimle sevgi ve saygılarımı sunarım.

Ali AKDAĞ
KONYA-2017

İÇİNDEKİLER

SİMGELER VE KISALTMALAR

Kısaltmalar

ALU	: Aritmetik Mantık Birimi (Arithmetic Logic Unit)
BVLC	: Berkeley İmgeleme ve Öğrenme Merkezi (Berkeley Vision and Learning Center)
CUDA	: Compute Unified Device Architecture
cuDNN	: CUDA Derin Sinir Ağı Kütüphanesi (CUDA Deep Neural Network Library)
CPU	: Merkezi İşlem Birimi (Central Processing Unit)
DIGITS	: Derin Öğrenme GPU Eğitim Sistemi (Deep Learning GPU Training System)
DPM	: Değişebilir Parça Bazlı Modeller (Deformable Part Model)
DSA	: Derin Sinir Ağı (Deep Neural Network)
DVM	: Destek Vektör Makineleri (Support Vector Machine)
FCN	: Tam Evrişimsel Ağ (Fully Convolutional Network)
GPGPU	: Genel Amaçlı Grafik İşlem Birimi (General Purpose Graphics Processing Units)
GPU	: Grafik İşlem Birimi (Graphics Processing Unit)
HOG	: Yönlendirilmiş Gradyanların Histogramı (Histogram of Oriented Gradients)
ILSVRC	: ImageNet Büyük Ölçekli Görsel Tanıma Yarışması (ImageNet Large Scale Visual Recognition Competition)
IoU	: Birlik Üzerinde Kesişme (Intersection Over Union)
LSVRC	: Büyük Ölçekli Görsel Tanıma Yarışması (Large Scale Visual Recognition Competition)
mAP	: Ortalama Kesinlik Değeri(Mean Average Precision)
MLP	: Çok Katmanlı Algılayıcı (Multi Layer Perceptron)
SGD	: Stokastik Gradyan İnişi (Stochastic Gradient Descent)
SIFT	: Boyut ile Değişmeyen Özellik Çıkarımı (Scale-Invarıant Feature Transform)
SURF	: Hızlandırılmış Gürbüz Öznitelikler (Speeded Up Robust Feature)
SLP	: Tek Katmanlı Algılayıcı (Single Layer Perceptron)
ReLU	: Doğrultulmuş Lineer Birim Fonksiyonu (Rectified Linear Unit)
YSA	: Yapay Sinir Ağı (Artificial Neural Network)
YOLO	: Sadece Bir Kez Bak (You Only Look Once)

1. GİRİŞ

Günümüzde makine öğrenmesindeki en popüler alanlardan biri, verilerin tanınabilir kavramlarını tespit etmeyi bilgisayarlara öğretmek için derin sinir ağını (DSA) kullanan derin öğrenmedir. Araştırmacılar ve endüstri uygulayıcıları görüntü ve video sınıflandırması, bilgisayarlı görme, ses tanıma, doğal dil işleme vb. uygulamalarda DSA'ları kullanmaktadır.

DSA'lar görüntü işlemede kullanılırken görüntünün tüm piksellerinin girdi veri kümesi olmasını gerektirir. Çok sayıda resim ve bu resimlerin piksellerinden oluşan girdi veri kümesi, DSA'nın çok katmanlı yapısıyla da birleşince yüksek hesaplama gücü ihtiyacını beraberinde getirir. Büyük karmaşık DSA'ları eğitmek için önemli bir platform olan grafik işlem birimleri (GPU) yardımıyla eğitim süresi aylardan birkaç güne indirilerek bu hesaplama maliyeti büyük oranda azaltılmıştır. Bu performansın altında yatan etkenlerin başında, GPU'ların donanımsal olarak gelişmelerinin yanı sıra bu donanımı genel amaçlı programlamaya olanak tanıyan ve grafik işlemcilerin limitlerini ortadan kaldıran CUDA (Compute Unified Device Architecture) gelmektedir.

1.1. GPU'ların Derin Öğrenmeye Etkisi

GPU'ların donanımsal olarak gelişmesi ve CUDA mimarisi sayesinde genel amaçlı programlanabilir hale gelmesi, GPU'ların görüntü işlemenin pek çok alanında etkili bir şekilde kullanılmasını sağlamıştır. Özellikle görüntü sınıflandırma, nesne tanıma, yüz tanıma, nesne takibi gibi uygulamalarda popülaritesi gittikçe artan DSA'ları önemli derecede hızlandırmıştır.

DSA'lar farklı veriler üzerinde aynı hesaplamayı gerçekleştiren binlerce hatta milyonlarca özdeş birimden inşa edilen büyük paralel yapılardır. Bu birimlerin çoğunun veri bağımlılığı yoktur, dolayısıyla eşzamanlı olarak hesaplama yapabilirler. Böyle bir ağın eğitimi normal CPU kümelerinde zordur çünkü eğitim sırasında birimler arasında çok miktarda veri iletişimi vardır. Tüm ağ bir CPU kümesi yerine tek bir GPU'ya yerleştirildiğinde, her şey çok daha hızlı olur, iletişim gecikmesi azalır, bant genişliği artar, boyut ve güç tüketimi önemli ölçüde azaltılır.

Donanımdaki gelişmeler, derin öğrenmeye ilginin artmasının sağlanmasında önem taşımaktadır. Nvidia, 2009 yılında derin öğrenme alanına "big bang" olarak

adlandırdığı derin öğrenme sinir ağları, Nvidia grafik işlemci birimleri (GPU) ile birleştirildiğinde yer almıştır.

2012 yılında Google Brain projesi YouTube'da film izleyerek kedileri ve insanları tanımayı öğrendi. Normalde bu işlemi gerçekleştirmek için Google'ın dev veri merkezlerinden birinde güç verilen ve soğutulan sunucularda 2.000 CPU gerekiyordu. Nvidia araştırma takımındaki Bryan Catanzaro, Stanford'daki Andrew Ng ekibi ile derin öğrenme için GPU'ları kullanmak üzere bir araya geldi. Sonuç olarak, 12 Nvidia GPU'nun, 2.000 CPU'nun derin öğrenme performansını sağlayabildiği görüldü.

2010'da Stanford'da Fei-Fei Li'nin grubu tarafından milyonlarca etiketli resim içeren Imagenet olarak bilinen büyük bir veri tabanı oluşturuldu ve yayınlandı. Bu veri tabanı, yarışmacıların bilgisayarlı görme modelleri oluşturdukları, tahminlerini gönderdikleri ve ne kadar doğru olduklarına göre bir puan aldıkları yıllık düzenlenen bir yarışma olan LSVRC (Large Scale Visual Recognition Competition) ile birleştirildi ve ILSVRC (ImageNet Large Scale Visual Recognition Competition) olarak adlandırılmaya başlandı. Yarışmanın ilk iki yılında üst modellerin hata oranı %28 ve %26 idi. Bu yıllarda hiçbir takım GPU hızlandırmalı DSA'ları kullanmıyordu. 2012'de Alex Krizhevsky, Ilya Sutskever ve Geoff Hinton mevcut hata oranını %16 'ya düşürdü. Hata oranındaki bu ciddi düşüşün arkasında yatan en önemli sebep GPU'ların kullanılmasıydı. O zamandan beri, GPU hızlandırmalı DSA'ları kullanan ekiplerin oranı önemli ölçüde arttı (Çizelge 1.1).

Çizelge 1.1. ILSVRC - yıllara göre kazanan hata oranı ve GPU kullanım sayısı

Yıllar	Kazanan hata oranı	GPU kullanarak katılanların sayısı
2010	%28	0
2011	%26	0
2012	%16	4
2013	%12	60
2014	%7	110

2011'den beri derin sinir ağlarının GPU tabanlı uygulamaları IJCNN 2011 Trafik İşareti Tanıma Yarışması, ImageNet yarışması gibi birçok örüntü tanıma yarışmasını kazanmıştır.

1.2. Resim Sınıflandırma ve Nesne Algılama

Resim sınıflandırma ve resimdeki nesneleri tespit edebilme, insanlar için doğduğumuz andan itibaren öğrendiğimiz ilk yeteneklerden biridir ve yetişkinlerde bu işlemler doğal ve zahmetsizce gerçekleşen olaylardır. İkinci defa düşünmeden bulunduğumuz çevreyi ve çevredeki nesneleri hızlı ve sorunsuz bir şekilde belirleyebiliriz. Bu kabiliyetlerimiz makinalarla paylaştığımız kabiliyetler arasındadır. Yani çeşitli algoritmalarla, makinalar da bir görüntü gördüğünde, çevrelerindeki dünyaya baktığında çoğu zaman sahneyi karakterize edebilir, her nesneye bir etiket verebilir, modelleri algılayabilir ve önceden edindiği bilgilerden genelleme yaparak, farklı görüntüler hakkında yorum yapabilirler. Bu işlemi makinalara yaptıran, en bilinen algoritmalardan biri derin öğrenme algoritmalarından olan evrişimsel sinir ağıdır.

Nesne algılama, bilgisayar görüsünde en zorlu sorunlardan biridir ve birçok bilgisayar görme uygulamasında ilk adımdır. Nesne algılama sisteminin amacı, bir görüntüdeki bilinen bir kategorideki nesnelerin tüm örneklerini algılamaktır. İyi bir nesne algılama sisteminin rastgele sahnelerde nesnelerin varlığına veya yokluğuna dayanıklı olması, nesne ölçeği, bakış açısı ve yönelimle değişmez olması ve kısmen kapatılmış nesneleri algılayabilmesi gerekir. Gerçek dünya görüntüleri nesnelerin birkaç örneğini veya çok daha fazlasını içerebilir. Bu, bir nesne algılama sisteminin doğruluğunu ve hesaplama verimliliğini etkiler.

Tarihsel olarak, nesne algılama sistemleri, her bölge için manuel olarak tasarlanmış özellikler oluşturma ve daha sonra destek vektör makinası (DVM) veya lojistik regresyon gibi sığ bir sınıflandırıcıyı eğitmeyi içeren özellik tabanlı teknikler kullanmıştır. Güçlü GPU'ların yaygınlaşması, CUDA mimarisinin ortaya çıkması ve büyük veri kümelerinin bulunabilir olması, nesne algılama için derin sinir ağlarının eğitilmesini sağlamıştır. DSA'lar, bahsedilen zorlukların neredeyse tamamına hitap etmektedir; çünkü bunlar, sığ ağlara kıyasla, görüntüdeki nesnelerin daha karmaşık temsillerini öğrenme ve elle tasarlanmış özelliklere olan bağımlılığı ortadan kaldırma kapasitesine sahiptirler.

1.3. Tezin Amacı

Günümüzde artan kamera sayısı, bu kameralardan elde edilen görüntüleri gerçek zamanlı olarak yorumlamak için insan ihtiyacını da beraberinde getirmiştir. Özellikle

güvenliğin kritik önem taşıdığı market, banka, kuyumcu gibi yerlerin izlenmesi daha da büyük önem arz etmektedir. Bu gibi yerler zaman zaman kötü niyetli insanlar tarafından silahlı bir şekilde soygun teşebbüsüyle karşı karşıya gelebilmektedir, orada bulunan insanların can ve mallarının tehlikeye düşmesine sebep olabilmektedir. Bu sebepten böyle bir silahlı soygun girişiminin veya silahlı herhangi bir müdahalenin ilgili birimlere, can ve mal kayıplarının önüne geçmek adına mümkün olduğu kadar erken ihbar edilmesi büyük önem taşımaktadır. Günümüzde nesne algılama sistemlerinin başarısı göz önünde bulundurulduğunda bu işlemi derin sinir ağları ve Nvidia CUDA kartlarıyla otonom bir şekilde gerçekleştirmek mümkündür.

Bu tez çalışmasında, son yıllarda başarılı nesne algılama modellerinden olan DetectNet ve YOLO derin öğrenme mimarileri kullanılarak, Nvidia'nın kullanıma sunduğu gömülü sistemlerinden olan Jetson TX1 üzerinde gerçek zamanlı silah algılama uygulaması gerçekleştirilmesi amaçlanmıştır (Koçer ve Akdağ, 2017).

2. KAYNAK ARAŞTIRMASI

2.1. Nesne Algılama Modelleri

Nesne algılama sınıflandırma gibi bilgisayarlı görme konusundaki klasik problemlerden biridir. Ancak algılama; nesneleri tanımlayabilen, ancak nesnenin resimde nerede olduğunu tam olarak söylemeyen, birden fazla nesne içeren görüntüler için çalışmayan sınıflandırmadan daha karmaşık bir sorundur.

Bilgisayar görüsünde, sınıflandırma ve nesne algılama için standart metotlar genellikle görüntü içerisindeki nesnelerin önceden öğrenilmiş özniteliklerinin kümesini bulmayı amaçlayan öznitelik eşleme (feature matching) temellidir. SIFT (Lowe, 1999) veya SURF (Bay ve ark., 2006) gibi bazı tanımlayıcılar görüntüler arasında ortak alanları bulmak için düşük seviyede öznitelik çıkararak kullanılmıştır (Lowe, 2004; Zhang ve Hu, 2011). Daha yüksek seviyedeki öznitelikler insan ve yaya algılamasında (Suard ve ark., 2006; Jafari ve ark., 2014; Spinello ve ark., 2011) iyi performans gösteren Haar-benzeri öznitelikler (Viola ve Jones, 2001) ve HOG (Dalal ve Triggs, 2005) gibi diğer gelişmiş tanımlayıcılar ile sağlanmıştır.

Yapay sinir ağları da geçmişte, örüntü tanımada geleneksel öznitelik temelli yaklaşımlarla karşılaştırıldığında oldukça iyi performans göstermiştir. Ancak standart yapay sinir ağında her nöron bitişik katmandaki her bir nörona bağlı olduğundan dolayı daha fazla nöron eklenerek ağın derinleşmesinin sağlanması, bağlantıları katlanarak büyüyen bir ağ haline getirmiştir. Bu yüzden sinir ağları 1980'lerde büyük bir problem olan boyutsal sorunlara maruz kalmıştır. Bu boyutsal sorunlar ve donanımsal bilgi işlem yetersizlikleri nedeniyle, sinir ağlarının popülerliği 1990'lı yıllarda azalmıştır. Destek vektör makineleri (DVM), yükseltme (boosting) yöntemleri ve değişebilir parça bazlı modeller (DPM) gibi diğer makine öğrenme teknikleri, sinir ağlarını geride bırakmıştır. Bu yöntemlerle HOG veya Haar-benzeri özniteliklerin kombinasyonu, nesne algılama ve sınıflandırma problemlerinde (Wang ve ark., 2008; Laptev, 2009; Girshick ve ark. 2010) oldukça iyi çalışmıştır. Değişebilir parça bazlı model (DPM) kullanan yaklaşım (Felzenszwalb ve ark., 2010) 2012'de PASCAL nesne algılama yarışmasının galibi olmuştur.

Yapay sinir ağlarındaki boyutsal sorunların önüne geçmek için üç yeni fikir (yerel alıcı alanlar, paylaşılan ağırlıklar ve alt örnekleme katmanı) getirilerek derin, yani çok katmanlı ağlar geliştirmeye izin veren evrişimsel sinir ağıları, araştırma gruplarının

dikkatini bir kez daha çekmeyi başarmıştır ve 1990'lı yıllarda farklı algılama ve sınıflandırma amaçları için yoğun bir şekilde kullanılmıştır (LeCun ve ark., 1989; LeCun ve ark., 1998). Evrişimsel sinir ağları, eğitim için GPU'ların kullanılmaya başlanması ve milyonlarca örneği kapsayan etiketlenmiş veri setlerinin bulunmasıyla birlikte daha da popüler olmuştur.

ImageNet yarışmaları nesne sınıflandırma, yer belirleme, algılama ve bölütleme görevleri için derin öğrenme metotlarının popülerleşmesinde çok etkili olmuştur. 2012'de sınıflandırma görevi için en iyi sonuçları başaran evrişimsel sinir ağı kullanan yaklaşım ve mimari (Krizhevsky ve ark, 2012) bilgisayar görüsü metotlarında önemli değişikliği beraberinde getirmiştir. Bu nedenle, evrişimsel sinir ağlarının başlangıçta sınıflandırma görevleri için umut verici olduğu görülmüştür. Sermanet ve ark., (2013) geliştirdikleri OverFeat modelinde nesnelerin yerini tespit etmek için evrişimsel sinir ağı kullanmış ve ILSVRC2013 nesne algılama veri setiyle %23.4'lük mAP değeri elde etmiştir. 2014 yılında OverFeat modelinin mAP değerinde %7.1'lik artışla, %31.4 mAP değeri elde eden evrişimsel sinir ağı tabanlı R-CNN modeli duyurulmuştur (Girshick ve ark., 2014). 2015 yılında VGG16 ağını R-CNN'den 9 kat daha hızlı eğiten, test zamanında 213 kat hızlı ve PASCAL VOC 2012 veri setinde daha yüksek mAP değerine sahip Fast R-CNN modeli sunulmuştur (Girshick, 2015). Sonra Fast R-CNN'den yaklaşık 10 kat hızlı çalışan Faster R-CNN modeli geliştirilmiştir (Girshick, 2015). Çok geçmeden, rapor edilen ilk gerçek zamanlı evrişimsel sinir ağı tabanlı nesne algılama modeli olan YOLO tanıtılmıştır (Redmon ve ark.,2016). Redmon ve ark., (2016) tarafından gerçekleştirilmiş olan kıyaslamada, PASCAL VOC 2007 veri setiyle eğitilen YOLO 45 fps'de %63.4 mAP değerine sahipken, Faster R-CNN mimarisi %73.2 mAP değeriyle daha yüksek doğruluğa sahip olmuştur, fakat sadece 7 fps'de çalışmıştır. Faster R-CNN doğruluk bakımından daha yüksek sonuçlar vermesine rağmen YOLO Faster R-CNN'den yaklaşık 6 kat daha hızlı çalışmıştır.

YOLO'nun literatürdeki en hızlı genel amaçlı nesne algılama mimarisi (Redmon ve ark.,2016) olduğu ve DetectNet'in de gerçek zamanlı nesne algılama için uygun bir mimari olduğu düşünüldüğünde, bu iki mimarinin Jetson TX1 gömülü sisteminde çalışacak gerçek zamanlı uygulamalar için daha elverişli olduğu görülmüştür.

2.2. İlgili Çalışmalar

Literatürde X-ray veya milimetrik dalga görüntüleriyle gizli silah bulma çalışmaları mevcuttur. Ancak X-ray tarayıcı ve taşıyıcı bantların gerekliliği, onları pek çok yer için pahalı bir çözüm yapar. Ayrıca bu sistemler metal tespitine dayandığı için metal olmayan silahlara karşı duyarlılıkları yoktur.

RGB görüntülerden silah algılamayla ilgili Verma ve ark. tarafından yapılan renk bazlı bölümleme ve harris ilgi noktası dedektörünü kullanan ve yine Verma ve ark. tarafından yapılan renk bazlı bölümleme ve SURF ilgi noktası dedektörü yöntemlerini kullanan çalışmalar mevcuttur (Verma ve ark.,2015). Halima ve ark. tarafından, boyut ile değişmeyen özellik çıkarımı (SIFT), K-Ortalama ve DVM kullanılarak bir çalışma gerçekleştirilmiştir (Halima ve ark, 2016). Ancak bu çalışmalarda önerilen yöntemler gerçek zamanlı işlem gereksinimlerinden olan hem aydınlanma değişikliğine hem de mekân karmaşıklığına karşı dayanıklı yöntemler değillerdir. Olmos ve arkadaşlarına göre de bu yöntemlerin hepsi yavaştır, sürekli izleme için kullanılamaz, aynı zamanda operatörün gözetimini gerektirir ve açık alanlarda kullanılmaya uygun değildir (Olmos ve ark, 2017). Olmos ve arkadaşlarının çalışması RGB görüntüleri kullanarak derin öğrenme algoritmalarıyla yapılan, bilinen ilk silah algılama çalışmasıdır (Olmos ve ark., 2017). Bu çalışmada Faster R-CNN mimarisi kullanılarak yüksek doğrulukta sonuçlar elde edilmiştir.

3. MATERYAL VE YÖNTEM

Bu bölümde tez çalışmasında kullanılacak gerekli notasyonlar ve alakalı kavramlar tanıtılmıştır. Derin öğrenme algoritmasının temel yapısı, Nvidia CUDA mimarisi, DetectNet ve YOLO nesne algılama modelleri, kullanılan materyaller, veri seti hazırlık süreci ve modellerin eğitimleri hakkında bilgi verilmiştir.

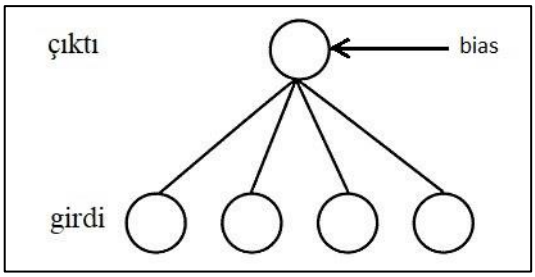
3.1. Derin Öğrenme

Derin öğrenme, bir bilgisayarı konuşma tanıma, görüntüleri tanımlama veya tahmin yapma gibi insana benzer görevleri gerçekleştirmek için eğiten bir tür makine öğrenmesi algoritmalar kümesidir. Ön tanımlı denklemlerle çalışmak için veri düzenlemek yerine, derin öğrenmede veriler temel parametreleri oluşturur ve bilgisayar, birçok proses katmanını kullanarak örüntüleri tanımak suretiyle eğitilir. Derin öğrenme algoritmaları, yapay sinir ağları kökenli modellerden ve enerji tabanlı modellerden oluşmaktadır. Mimarisi birçok katman ve saklı değişkenlerden meydana gelir. En çok kullanılan algoritmalar Derin Sinir Ağları (Deep Neural Networks), Otomatik Kodlayıcılar (Autoencoders) ve Boltzmann Makinelerinin bir türevi olan Kısıtlı Boltzmann Makineleridir (Restricted Boltzmann Machines) (Özcan, 2014).

Derin sinir ağı (DSA), giriş katmanı ile çıktı katmanı arasında çok sayıda gizli katman bulunan bir yapay sinir ağıdır (YSA). Sığ YSA'lara benzer şekilde, DSA'lar karmaşık doğrusal olmayan ilişkileri modelleyebilir. Bu ağlar, nöron adı verilen, birbirleriyle iletişim kuran ve birbirine bağlı işlem birimlerinden oluşan bir öğrenme sistemidir. Bir ağın tasarımı, yani katmanların sayısı, birim sayısı veya birimler arasındaki bağlantı, o ağın mimarisini tanımlar. Katman sayısı mimarinin derinliğini tanımlar.

3.1.1. Tek Katmanlı Algılayıcılar (Single Layer Perceptron - SLP)

Tek katmanlı algılayıcılar (SLP) ikili sınıflandırıcı olarak eğitilen en basit sinir ağı türüdür. Öğrenebilir ağırlıklar vasıtasıyla bir çıktı (output) ünitesine bağlı, bir dizi girdi (input) ünitesinden oluşur. Ayrıca çıkış değerini değiştirmeye olanak tanıyan bias olarak adlandırılan ek bir ağırlık türüne sahiptir. Bias ve ağırlık değerleri sinir ağının parametrelerini oluşturur. Şekil 3.1 SLP mimarisini göstermektedir.



Şekil 3.1. Tek katmanlı algılayıcı

Ağın çıktısı ilgin (affine) dönüşümü ve ardından doğrusal olmayan bir eşik fonksiyonu ile hesaplanır (Denklem 3.1).

$$h = \begin{cases} 1 & eğer\ xw + b \geq 0 \\ 0 & diğer\ durum \end{cases} \tag{3.1}$$

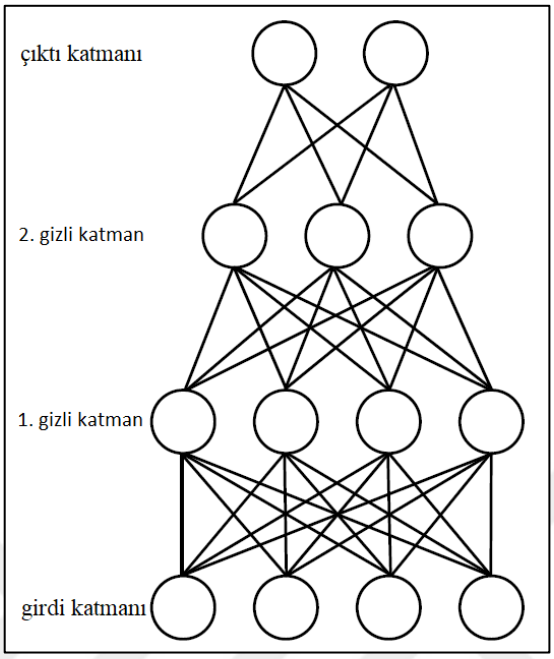
x girdi vektörü, w öğrenebilir ağırlık vektörü, xw nokta çarpımı, b öğrenebilir bir bias değerini göstermektedir. Eşik fonksiyonu ise sınıflayıcı çıktısını yani bir girdinin bir sınıfa ait olup olmadığını belirler.

SLP çıktı birimi bir çıktı vektörüne değiştirilerek çoklu sınıf problemleri için genelleştirilebilir. Bu durumda, tüm girdi birimleri tüm çıktı birimlerine bağlanır. Her çıktı biriminin kendi bias terimi vardır. Doğası gereği, SLP yalnızca doğrusal sınıflandırıcıları öğrenebilir. Bu nedenle doğrusal olmayan ayrılabilir girdi vektörlerini düzgün bir şekilde sınıflandıramaz.

3.1.2. Çok Katmanlı Algılayıcılar (Multi Layer Perceptron - MLP)

Çok katmanlı algılayıcılar (MLP), bir özellik hiyerarşisi oluşturmak için gizli katman olarak da adlandırılan ara katmanlar eklenerek SLP sınırlarını aşan bir sinir ağı türüdür. MLP, bir girdi katmanı, bir dizi gizli katman ve çıktı katmanı içerir. Bütün bu katmanlar bir katmanın çıkışı bir sonraki katmanın girişi olacak şekilde birbiri üzerine yığılır. SLP'de olduğu gibi, bir katmandaki her bir birim, bitişik katmandaki her birime bağlıdır, yani katmanlar tamamen bağlıdır. Birimler arasındaki bağlantılara ağırlık denir. Aynı birime bağlı ağırlıklar kernel olarak da adlandırılan bir filtre oluşturur. MLP'lerin gizli ve çıktı birimleri de onlarla ilişkili bir bias değerine sahiptir. Şekil 3.2'de, bir girdi katmanı, iki gizli katman ve çıktı katmanı bulunan bir MLP'nin mimarisi

gösterilmektedir. Gizli katmanların sayısı ve gizli katman başına birim sayısı, MLP mimarilerinin hiperparametreleridir.



Şekil 3.2. Çok katmanlı algılayıcı

MLP'nin her katmanı, girdi verilerine doğrusal olmayan bir dönüşüm uygular. MLP birden fazla katmanı bir araya getirdiğinden, çıktı, bir işlev bileşimi olarak hesaplanabilir (Denklem 3.2):

$$g^L(\dots g^2(g^1(x;\theta^1);\theta^2)\dots;\theta^L), \tag{3.2}$$

x girdi vektörü, $\theta^l = \{W^l, b^l\}$ l. katmanın parametreleridir, g^l doğrusal olmayan katmandır, $l \in [1 \dots L]$ ve L katmanların sayısıdır. Mantık, ardarda doğrusal olmayan dönüşümler uygulamaktır.

3.1.2.1. İlgin Dönüşümü (Affine Transformation)

İlgin dönüşümü, MLP'nin her tabakası tarafından gerçekleştirilir. Hesaplama mantığı, matris çarpımı ve ardından bir bias eklemektir (Denklem 3.3).

$$a^l = h^{l-1}W + b^l, \tag{3.3}$$

burada $h^{l-1} \in \mathbb{R}^{N_h^{l-1}}$ l. katmanın girdisidir, $W^l \in \mathbb{R}^{N_h^{t-1} \times N_h^l}$ katmandaki girdi ve çıktı birimlerini birleştiren öğrenebilir ağırlık matrisidir, $b^l \in \mathbb{R}^{N_h^l}$ katmandaki öğrenebilir bias vektörüdür, $a^l \in \mathbb{R}^{N_h^l}$ ilgin dönüşümünün çıktısıdır, N_h^{l-1} ve N_h^l sırasıyla girdi ve aktivasyon öncesi boyutluluklardır.

3.1.2.2. Doğrusal Olmayanlık (Non-Linearity)

Aktivasyon fonksiyonu olarak adlandırılan bir f fonksiyonu her tabakanın ilgin dönüşümü çıktısına (a^l) uygulanmasıyla doğrusal olmayanlık sağlanır.

Aktivasyon fonksiyonu hücreye gelen net girdiyi işleyerek hücrenin bu girdiye karşılık üreteceği çıktıyı belirler. Aktivasyon fonksiyonu olarak genellikle doğrusal olmayan bir fonksiyon seçilir. Yapay sinir ağlarının bir özelliği olan "doğrusal olmama" aktivasyon fonksiyonlarının doğrusal olmama özelliğinden gelmektedir. Aktivasyon fonksiyonu seçilirken dikkat edilmesi gereken bir diğer nokta ise fonksiyonun türevinin kolay hesaplanabilir olmasıdır. Geri beslemeli ağlarda aktivasyon fonksiyonunun türevi de kullanıldığı için hesaplamanın yavaşlamaması adına türevi kolay hesaplanır bir fonksiyon seçilir.

Doğrusal problemler çözmek amacıyla aktivasyon fonksiyonu doğrusal bir fonksiyon olarak seçilebilir. Toplama fonksiyonundan çıkan sonuç, belli bir katsayı ile çarpılarak hücrenin çıktısı olarak hesaplanır.

En popüler aktivasyon fonksiyonlarından biri hiperbolik tanjant fonksiyonudur (Denklem 3.4). x birimi girdilerin ağırlıklandırılmış doğrusal birleşimidir. Bu fonksiyon en etkili şekilde girdiler (0,1) aralığında olduğu durumda çalışır. (-1,1) aralığında çıkış üretir.

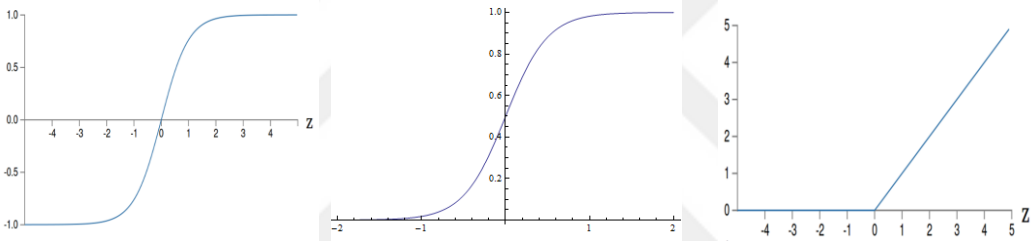
$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x}-1}{e^{2x}+1} \tag{3.4}$$

Lojistik sigmoid fonksiyonu (Denklem 3.5) yaygın olarak kullanılan, hiperbolik tanjant fonksiyonundan daha makul olan bir aktivasyon fonksiyonudur. Sigmoid işlevinin yaygın olarak kullanıldığı nedenlerden biri, sigmoid işlevinin her noktada diferansiye edilebilir olmasıdır.

$$f(x) = \frac{1}{1+e^{-x}} \tag{3.5}$$

Doğrultulmuş lineer birim fonksiyonu (rectified linear unit's function - ReLU) (Denklem 3.6) ağın doğrusal olmayanlığını artırmak amacıyla kullanılır. Doğrultulmuş nöronların, lojistik sigmoid veya hiperbolik tanjant nöronlardan daha makul olduğu düşünülmektedir (Glorot ve ark., 2011). ReLU ayrıca daha basit bir fonksiyon olduğu için belirli durumlarda daha hızlı eğitim ve performans iyileştirmeleri sağlar (Maas ve ark., 2011). Bu yüzden DSA'larda ReLU çok sık kullanılır.

$$f(x) = \max(0, x) \tag{3.6}$$



(a) Hiperbolik Tanjant **(b)** Sigmoid **(c)** ReLU

Şekil 3.3. Yaygın olarak kullanılan aktivasyon fonksiyonları

Softmax aktivasyon fonksiyonu (Denklem 3.7) genellikle son ağ katmanında kullanılır. Gerçek değeri, (0,1) aralığında sınıfın olasılıksal bir değerine dönüştürür.

$$p(c_k|x) = \frac{e^{a_k}}{\sum_{i=1}^{m} e^{a_i}}, \tag{3.7}$$

m çıktı birim sayısına (sınıf sayısına) karşılık gelir. a_k ise k. düğümdeki aktivasyon değeridir.

3.1.3. Eğitim (Training)

3.1.3.1. Kayıp Fonksiyonu (Loss Function)

Ağ sonucunun doğruluğunu ölçmek için bir kayıp fonksiyonu (maliyet fonksiyonu) kullanılır. Tahminin beklenen değerden ne kadar farklı olduğunu ifade eder. Kayıp fonksiyonunun çıktısı, maliyet veya ceza olarak adlandırılan reel bir değerdir. Görsel sınıflandırma problemlerinde kullanılan kayıp fonksiyonu genellikle çapraz-entropi (cross-entropy) kayıp fonksiyonudur (Denklem 3.8).

$$L = -\sum_{i}^{m} y_i \log p_i, \tag{3.8}$$

m çıkış katmanındaki olası sınıfların sayısıdır, y hedef vektör, p ise her sınıf için ağ tarafından tahmin edilen olasılık değeridir.

3.1.3.2. Geri Yayılım (Backpropagation)

Geri yayılım bir sinir ağı eğitim algoritmasıdır. Denetimli öğrenmede hedef sınıflar hata hesaplaması için gereklidir. Hata (e) daha sonra önceki katmandaki her düğüme geri gönderilir. Bu hata (e), x girdisi ve aktivasyon fonksiyonu (ϕ) verilen her katmandaki ağırlıklara (w_{kj}) göre kayıp fonksiyonunun (L) bir gradyanı olarak elde edilir (Denklem 3.10).

$$a_j = \phi\left(\sum_{k=1}^{n} w_{kj} x_k\right) \tag{3.9}$$

$$e = \frac{\partial L}{\partial w_{kj}} = \frac{\partial L}{\partial a_j}\frac{\partial a_j}{\partial w_{kj}} \tag{3.10}$$

Gradyan hesaplaması, belirli ağırlık w_{kj}'ye göre kayıp fonksiyonunun L kısmi türevini hesaplamak ve zincir kuralının uygulanmasını gerektirir. Hata (e) değeri kullanılarak, ağırlıklar gradyan inişi gibi bir optimizasyon algoritması tarafından güncellenir.

3.1.3.3. Gradyan İnişi (Gradient Descent)

Sinir ağları için kullanılan en yaygın optimizasyon algoritması gradyan inişi algoritmasıdır. Gradyan inişi, bir fonksiyon üzerinde rastgele bir noktadan başlayıp bu noktanın koordinatlarını, kısmi türevlerin tersi (yani gradyan vektörünün tersi) yönünde değiştirerek küçük adımlarla minimum noktaya yaklaşma yöntemidir. Atılacak adımların büyüklüğünü adım boyutu adını verdiğimiz çarpan belirler. Adım boyutuna öğrenme oranı da denir. Öğrenme oranı parametrelerin (ağırlıkların) değişim büyüklüğünü kontrol eden aralıkta (0,1) bir değerdir. Ağırlıklarda bir güncelleme gerçekleştirmek için, tüm eğitim seti kullanılmalıdır. Büyük eğitim setleri için bu yöntem hesaplama açısından pahalı olabilir.

Stokastik gradyent inişi (SGD) (Denklem 3.11) daha verimli bir optimizasyon yöntemidir. SGD, w model parametrelerini güncellemek için eğitim setinin alt kümesine ihtiyaç duyar. SGD, her ağırlık güncellemesinde rastgele bir küme kullanır. Ayrıca SGD sıradan gradyan inişi gibi yerel minimuma takılma eğiliminde değildir. SGD'nin bir dezavantajı, sıradan gradyan inişinin yakınsama oranından daha yavaş bir yakınsama oranına sahip olmasıdır.

$$w^{(t+1)} = w^{(t)} - \eta \nabla_w L(w^{(t)}) \tag{3.11}$$

Ağırlıklar, kayıp fonksiyonunun ağırlıklara göre gradyanının negatifiyle güncellenmektedir. Bu değişiklik öğrenme oranı (η) ile sınırlıdır.

3.1.3.4. Kaybolan Gradyan Problemi (Vanishing Gradient Problem)

Geriye yayılımlı derin sinir ağlarının eğitilmesi, geçişlerin birçok katmana yayılması anlamına gelir. Zincir kuralı kullanılarak, herhangi bir katmana göre gradyan, halihazırda geçirilen her katman için aktivasyon fonksiyonlarının gradyanları ile çarpılır. Aktivasyon fonksiyonlarının gradyanları tipik olarak (-1, 1) aralığında sınırlandığından sonuç gradyan, büyüklüklerinin 1'den az olduğu sayıların bir ürünüdür ve bu nedenle 0'a yaklaşma eğilimi gösterir. Etkili bir şekilde, ağın giriş katmanına yakın katmanlar, ağın çıkışına daha yakın olan katmanlarla karşılaştırıldığında daha yavaş öğrenir ve bu da verimsiz ve yavaş bir öğrenme sürecine neden olur. Bu soruna, kaybolan gradyan

problemi denir. Derin sinir ağlarında bu sorununun çözümünde önemli ölçüde yardımcı olan bir teknik, ReLU'ların kullanılmasıdır (Goodfellow ve ark., 2016).

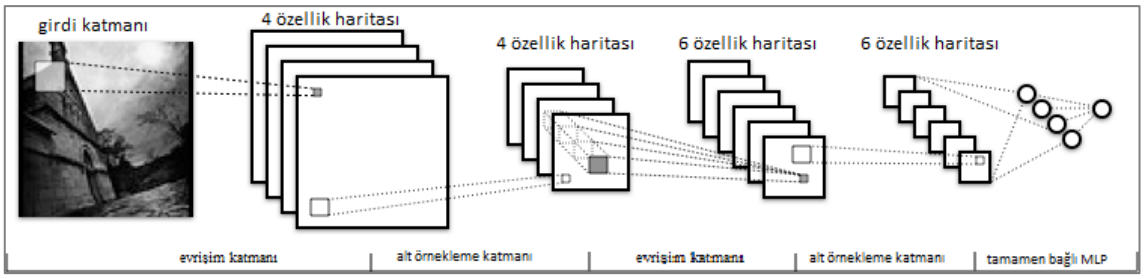
3.1.4. Evrişimsel Sinir Ağı (Convolutional Neural Network)

Evrişimsel sinir ağları beynin görsel korteksinden biyolojik ilham almışlardır. Görsel korteks, görsel alanın spesifik bölgelerine duyarlı küçük hücrelere sahiptir. Bu fikir, beyindeki bazı bireysel nöronal hücrelerin belirli bir kenar yöneliminin varlığına cevap verdiğini (tetiklendiğini) gösterdikleri, 1962'de Hubel ve Wiesel'in deneyiyle ispatlandı (Hubel ve Wiesel, 1962). Örneğin, bazı nöronlar dikey kenarlara maruz kaldığında tetiklenirken bazıları yatay veya köşegen kenarlar gösterildiğinde tetiklendi. Hubel ve Wiesel, bu nöronların hepsinin sütun şeklinde bir mimaride örgütlendiğini ve birlikte görsel algılama ürettiklerini keşfettiler. Belli görevleri olan (spesifik özellikleri arayan görsel korteksteki nöronal hücreler) bir sistemin içindeki özel bileşenler fikri, evrişimsel sinir ağlarının arkasındaki temeldir. Evrişimsel sinir ağlarını çok az miktarda ön işleme tabi tutmak için tasarlanmış çok katmanlı algılayıcıların bir varyasyonudur. Görüntü ve video tanıma, tavsiye sistemleri ve doğal dil işleme alanlarında geniş kullanım alanlarına sahiptirler.

Evrişimsel sinir ağları, başarılarının belgelendiği bilgisayar görüsünde kullanılmaktadır (LeCun Y. ve ark., 1998). Evrişimsel sinir ağları, görsel ve diğer iki boyutlu verileri işlemek için tercih edilen yöntem haline gelmiştir. Son zamanlarda yapılan çalışmalarla, evrişimsel sinir ağları biçimindeki derin öğrenme mimarilerinin en iyi performans sergilediği gösterilmiştir. Diğer derin mimarilerle karşılaştırıldığında, evrişimsel sinir ağları hem görüntü hem de konuşma tanıma uygulamalarında üstün sonuçlar ortaya koymuştur. Standart geri yayılım algoritması ile de eğitilebilirler. Evrişimsel sinir ağlarının, diğer normal derin ileri beslemeli sinir ağlarından daha kolay eğitilmesi ve daha az parametreye sahip olması, onların son derece çekici bir mimari olarak kullanılmalarını sağlamaktadır.

3.1.4.1. Evrişimsel Sinir Ağının Temel Yapısı

Bir evrişimsel sinir ağı temel olarak evrişim katmanları (convolution layers), alt örnekleme katmanları (pooling layers) ve tamamen bağlı katmanlardan (fully connected layers) oluşur. Tipik bir evrişimsel sinir ağının yapısı Şekil 3.4’teki gibidir.

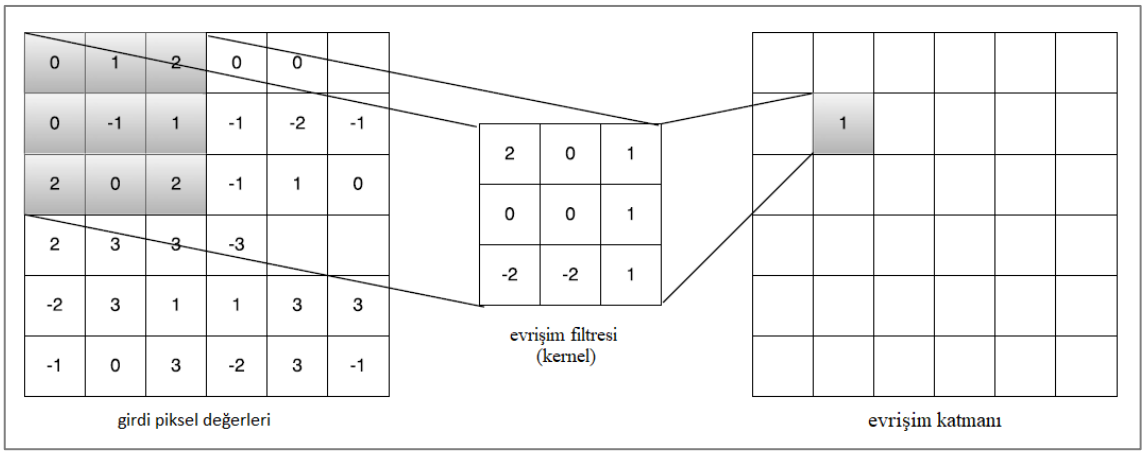


Şekil 3.4. Evrişimsel sinir ağı mimarisi

Evrişim (Convolution) Katmanı

Evrişim katmanında, 2D görüntü evrişimi gerçekleştirilir ve yeni filtrelenmiş bir görüntü elde edilir. Şekil 3.5'te, 2D bir görüntünün evrişim işleminin temsili gösterilmiştir.

Her bir evrişim işlemi için, filtredeki ögeler, alıcı alanda karşılık gelen piksel değeri ile çarpılır ve daha sonra birleşmiş değeri elde etmek için toplanır. Filtreler, tüm girdi kümesinin üzerinde kaydırılır, böylece birleştirilmiş aktivasyon haritaları (özellik haritaları) oluşur. Şekil 3.5'te sadece tek filtre ile evrişim işlemine tabi tutulan girdi sonucu oluşan bir aktivasyon haritası gösterilmiştir. Ayrıca, evrişim işleminden sonra genellikle bir bias terimi eklenir. Her bias bir filtreye karşılık gelir.



Şekil 3.5. Evrişim işlemi

Evrişim işleminin matematiksel gösterimi Denklem 3.12'deki gibidir.

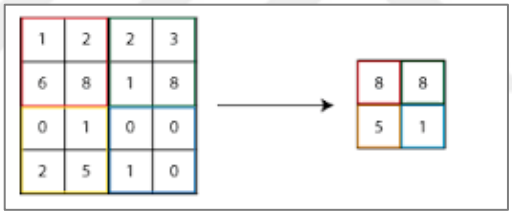
$$y_{ijk} = (W_i * x)_{jk} + b_i\,, \tag{3.12}$$

y_{ijk} i. çıktı özellik haritasının (j, k) koordinatının piksel değerini belirtir, W_i i. evrişim filtresini belirtir, x girdi değerini belirtir, b_i i. evrişim filtresine karşılık gelen bias vektörün i. elemanıdır.

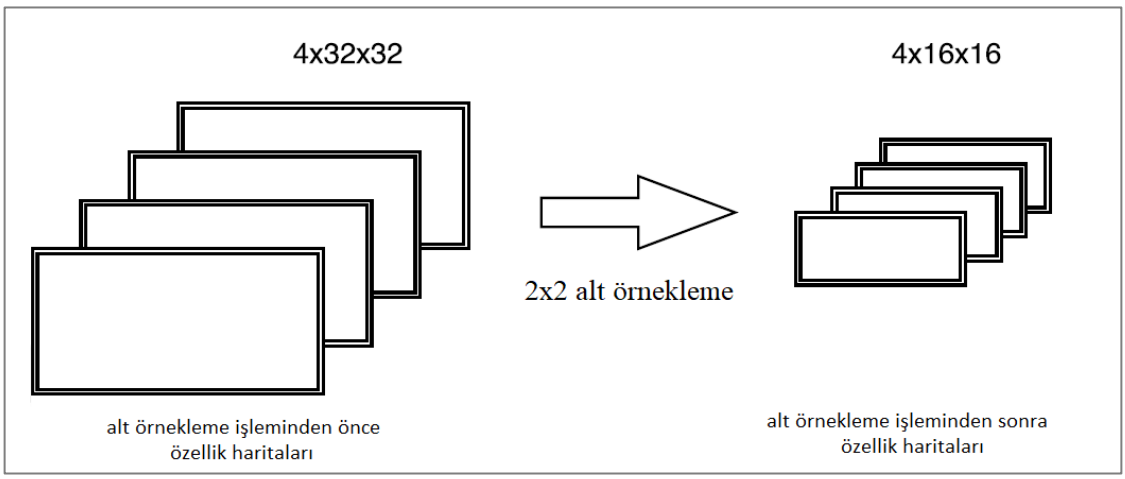
Alt Örnekleme Katmanı (Pooling Layer)

Bu katman, tüm ağın hesaplama maliyetini düşürmek için evrişim işlemleri ile üretilen girdi verisini alt örnekler. Böylece girdi verisinin boyutu azaltılır, dolayısıyla ağdaki parametrelerin sayısı ve hesaplama maliyeti azalır ve aşırı uygunluk kontrol altına alınmış olur.

Bir evrişimsel sinir ağı mimarisinde birbiri ardına gelen evrişim katmanları arasına periyodik olarak bir alt örnekleme katmanı yerleştirmek yaygındır. Alt örnekleme için farklı stratejiler önerilmiştir. En sık kullanılanlar şunlardır: ortalama alt örnekleme (average pooling), maksimum alt örnekleme (max pooling) ve rasgele alt örnekleme (stochastic pooling). Şekil 3.6'da bir maksimum alt örnekleme işlemi görülmektedir.



Şekil 3.6. Alt örnekleme işlemi (maksimum alt örnekleme)



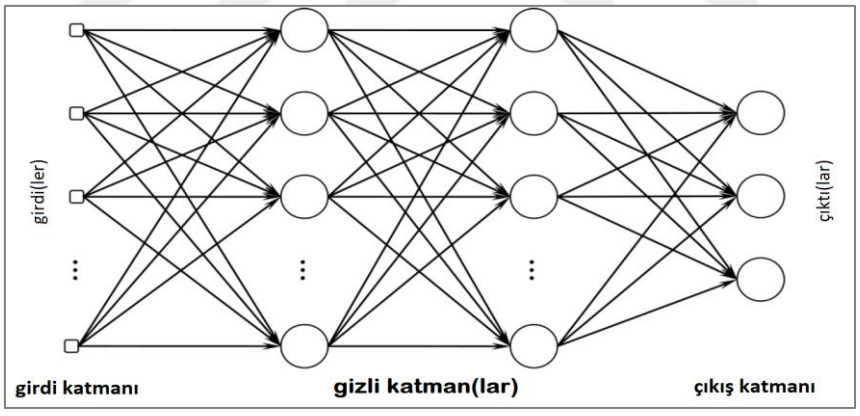
Şekil 3.7. Alt örneklemeden önce ve sonra aktivasyon haritaları

Aktivasyon Katmanı (Activation Layer)

Aktivasyon katmanları, özellik haritalarındaki her bir ögeye uygulanmak üzere doğrusal olmayan bir aktivasyon fonksiyonu kullanır. Bu işlem, insan sinir sisteminden ilham almıştır, bir uyarının nasıl aktarıldığını taklit eder. En sık kullanılan aktivasyon fonksiyonları arasında sigmoid fonksiyonu, hiperbolik tanjant fonksiyonu ve doğrultulmuş lineer birim fonksiyonu (ReLU'lar) bulunur. Bu üç fonksiyon tahminlerin doğruluğuna katkıda çok fazla farklılık göstermez. Bununla birlikte, ReLU'lar parça-bazlı doğrusal bir fonksiyon kullanır ve gerçek zamanlı uygulamalar düşünüldüğünde daha verimli çalışırlar.

Tamamen Bağlı Katmanlar (Fully Connected Layers)

Tamamen bağlı katmanların yapısı Şekil 3.8'de gösterilen geleneksel çok katmanlı algılayıcıların (MLP) yapısıyla aynıdır. Burada "tamamen" terimi, her çıkış nöronunun bir önceki katmandaki tüm giriş nöronuna bağlı olduğu anlamına gelir.



Şekil 3.8. Çok katmanlı algılayıcı

Sınıflandırıcı (Classifier)

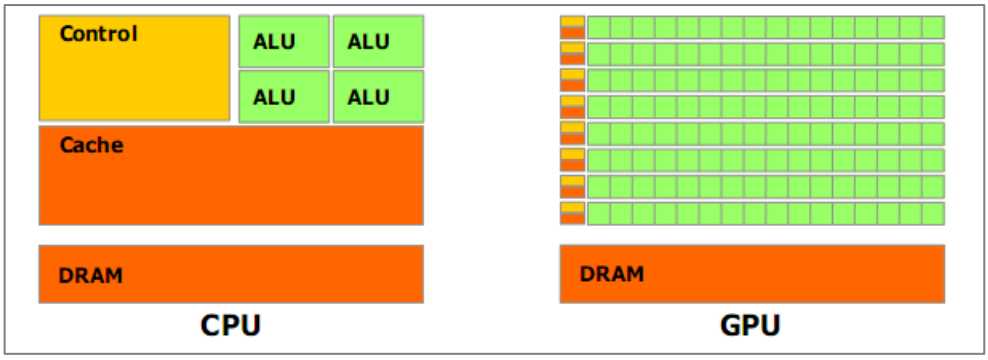
Çok kategorili bir sınıflandırma için en çok kullanılan sınıflandırıcılardan biri softmax sınıflandırıcıdır (Denklem 3.7). Sınıflandırıcı son tamamen bağlantılı katmanın çıktılarını girdi olarak alır ve sınıflar üzerindeki olasılık dağılımını hesaplar.

3.2. CUDA

CUDA, Nvidia firmasının 2006 yılından itibaren "Compute Unified Device Architecture" adının kısaltılmış hali olarak yayınlanan, Nvidia grafik kartlarının 200 serisinden sonraki modellerinde desteklenen ve bu grafik işlemcilerin genel amaçlı hesaplama işlemleri için kullanılabilmesine olanak tanıyan bir yazılım mimarisidir. Temel olarak bu mimariye sahip grafik kartları, genel amaçlı grafik işlem birimi (GPGPU - General Purpose Graphics Processing Units) olarak nitelendirilmektedir.

Görüntü işlemenin pikseller üzerinde yapıldığı ve her bir pikselin aynı işlemlerden geçtiği göz önünde bulundurulduğunda, GPU'ların bu işlemi paralel bir şekilde gerçekleştirmesi önemli derecede performans artışı sağlayacaktır. GPU, bunu birbiri ile veri paylaşımı yapabilen paralel dizilime sahip çekirdeklerle yapabilmektedir.

Geleneksel CPU donanım mimarisi ve GPU mimarisi Şekil 3.9'da gösterilmiştir. CPU'da az sayıda aritmetik lojik üniteleri (ALU) ve geniş kontrol ünitesinin beraberinde, paylaşımlı önbellek büyük yer kaplamaktadır. GPU'da ise çok sayıda küçük kontrol üniteleri ve paylaşımlı önbellek, daha çok sayıda çekirdeğe bloklar halinde atanıp işleyişi yönetmektedirler.

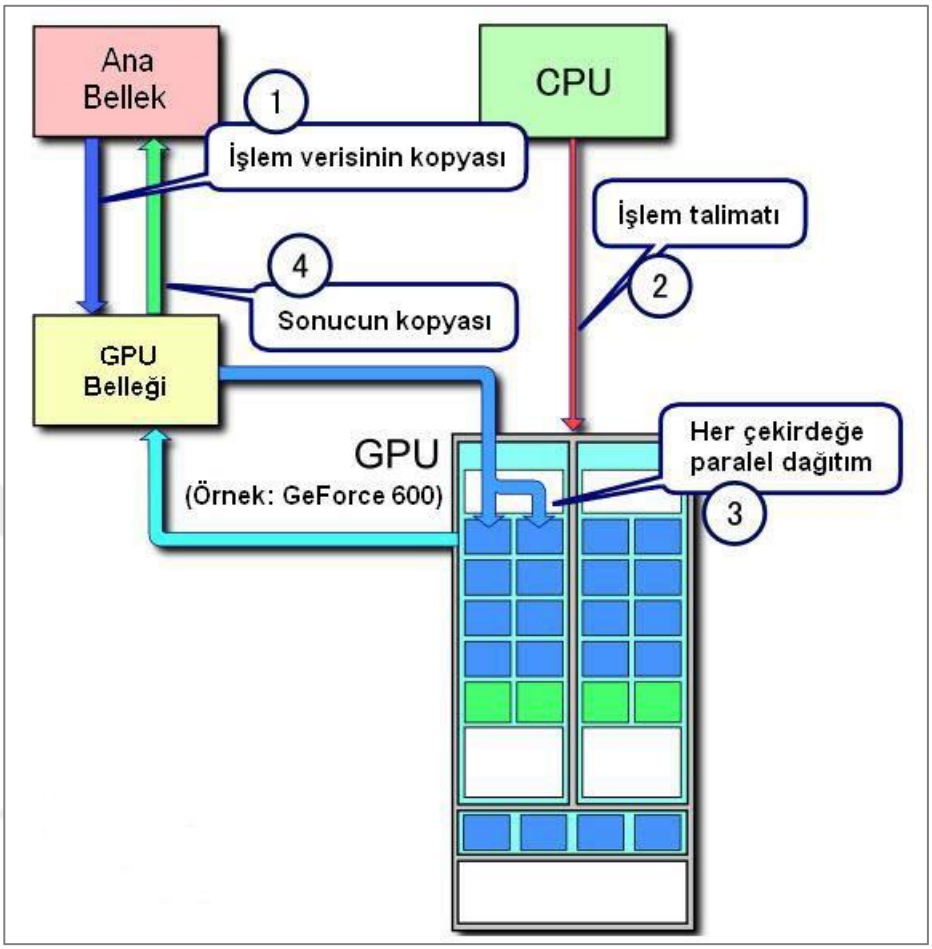


Şekil 3.9. CPU ve GPU'nun kontrol üniteleri, çekirdekleri ve bellek yapılarının kıyaslanması

3.2.1. CUDA İş Akışı

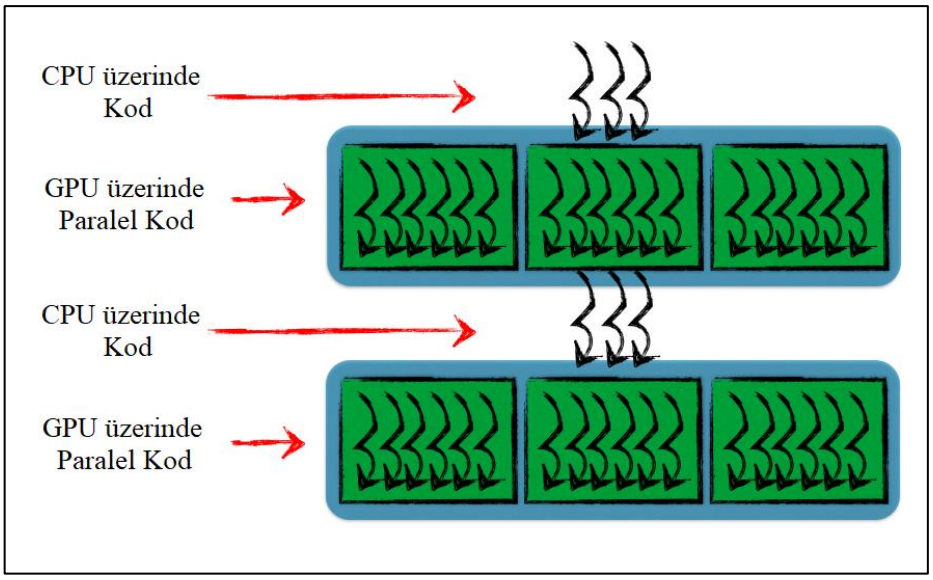
CUDA mimarisinde geliştirilen uygulamalar sadece GPU üzerinde çalışmazlar. Öncelikle uygulamanın CPU tarafından kontrol edilen ana bellek üzerinden grafik kartı üzerindeki belleğe kopyalanması gereklidir. GPU belleğindeki veri, CUDA iş parçacıkları tarafından yürütülerek paralel olarak hesaplanması tamamlanır ve ardından

tekrar ana belleğe gönderilerek işlem sonlandırılır. Bu akış modeli Şekil 3.10'da gösterilmektedir.



Şekil 3.10. CUDA işlem akışı

CUDA programlarının işleyişi ile ilgili bir diğer husus ise, programlarda yoğun paralellik gerektirmeyen bölümlerin düşük frekanslı GPU çekirdekleri yerine daha yüksek frekanslı olan CPU üzerinde çalıştırılmasıdır. Bu da gerektiğinde CPU ile asenkron çalışabilecek CUDA program parçacıklarının ve standart CPU kod parçacıklarının bir arada çalışmasına olanak vermektedir. İlgili anlatım Şekil 3.11'de gösterilmiştir. (Çekmez, 2014)



Şekil 3.11. CUDA ile gerçeklenen programların genel çalışma akışı (Çekmez, 2014)

3.3. Kullanılan Araçlar ve Modeller

3.3.1. Nvidia Digits

Nvidia Digits derin sinir ağlarının geliştirilmesi, eğitilmesi ve görselleştirilmesini sağlayan açık kaynak kodlu bir yazılımdır. Bu yazılım, veri bilimciler ve araştırmacıların derin sinir ağı tasarlamasını önemli ölçüde kolaylaştırmaktadır. Nvidia Digits, verileri yönetmek, ağ mimarilerini tanımlamak, birden fazla modeli paralel olarak eğitmek ve eğitim performansını gerçek zamanlı olarak izlemek gibi ortak derin öğrenme görevlerini etkileşimli olarak gerçekleştirmek için kullanılabilir.

Digits bir web tarayıcısı üzerinden erişilen bir web uygulaması olarak çalışır. Derin sinir ağlarını eğitmek için kullanıcı dostu bir arayüz sağlar. Derin sinir ağının optimizasyonu için çeşitli araçlar sağlar. Ana ekranda, var olan veri setlerini, mevcut olan daha önce eğitilmiş ağ modellerini ve devam etmekte olan eğitim aktivitesini listeler. Sayfa üzerinden ağı yapılandırmak için yapılan ayarlar takip edilebilir ve doğruluğu en üst seviyeye çıkarmak için çeşitli parametrelerde değişiklikler kolaylıkla yapılabilir.

Digits, sinir ağlarını görselleştirmeyi ve doğruluklarını hızlı bir şekilde karşılaştırmaya imkan tanır. Eğitim tamamlandıktan sonra veya eğitim sırasındaki her bir eğitim periyodunda modeli test etme seçeneği sunar.

Digits, bir web sunucusunda çalıştığı için veri setlerinin, ağ ayarlarının, test ve sonuçların paylaşılması kolaydır. Böylelikle takım çalışmasını önemli ölçüde kolaylaştırır.

Digits, CUDA derin sinir ağı kütüphanesini (cuDNN) kullanarak GPU hızlandırmasını destekler, bu sayede eğitim süresi büyük ölçüde azalır. CuDNN derin sinir ağı kütüphaneleri için bir GPU hızlandırma kütüphanesidir, Nvidia derin öğrenme yazılım geliştirme kitinin bir parçasıdır. CuDNN derin sinir ağlarının en iyi performansı elde etmesini kolaylaştırır ve ileride çıkacak yeni GPU'ların özelliklerinden faydalandırır.

Bu tez çalışmasında DetectNet derin sinir ağının eğitiminde Nvidia Digits kullanılmıştır.

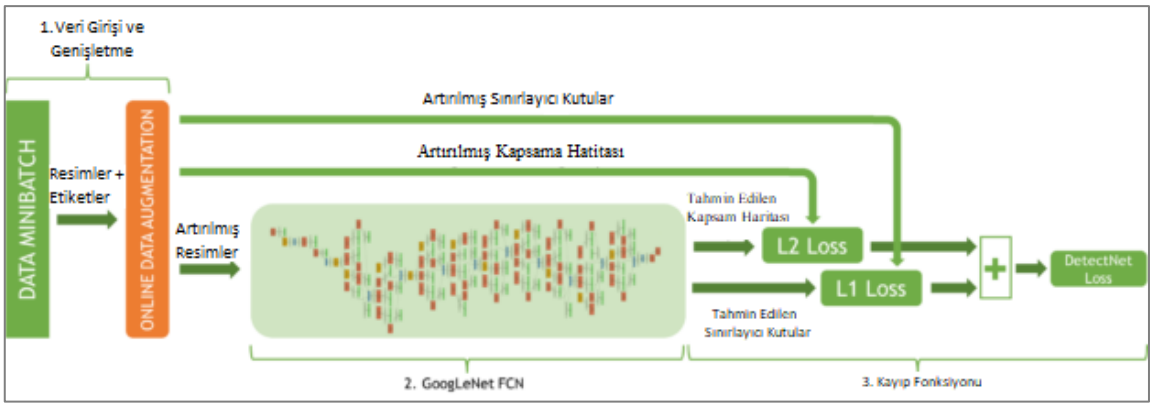
3.3.2. DetectNet

DetectNet eşzamanlı olarak hem nesne sınıflandırmasını gerçekleştiren hem de nesnelerin sınırlayıcı kutularını tahmin eden yeni bir ağ mimarisidir. Bu sınırlayıcı kutuları tahmin etmek için tek bir sinir ağı eğitilebileceği anlamına gelir. Oluşan tek ağ, kayan pencere algılayıcısı olarak uygulanan tipik bir sınıflandırıcıya kıyasla çok daha yüksek çıkarsama performansı sunar.

Digits, DetectNet olarak adlandırılan bu yeni sinir ağı modeli mimarisini içerir. DetectNet, Digits'e standart bir model tanımlaması olarak sağlanır ve Caffe derin öğrenme uygulama çatısı kullanılarak eğitilir.

DetectNet mimarisi beş bölümden oluşur. Şekil 3.12 eğitim sırasında kullanılan ve üç önemli işlemi işaret eden DetectNet mimarisini göstermektedir.

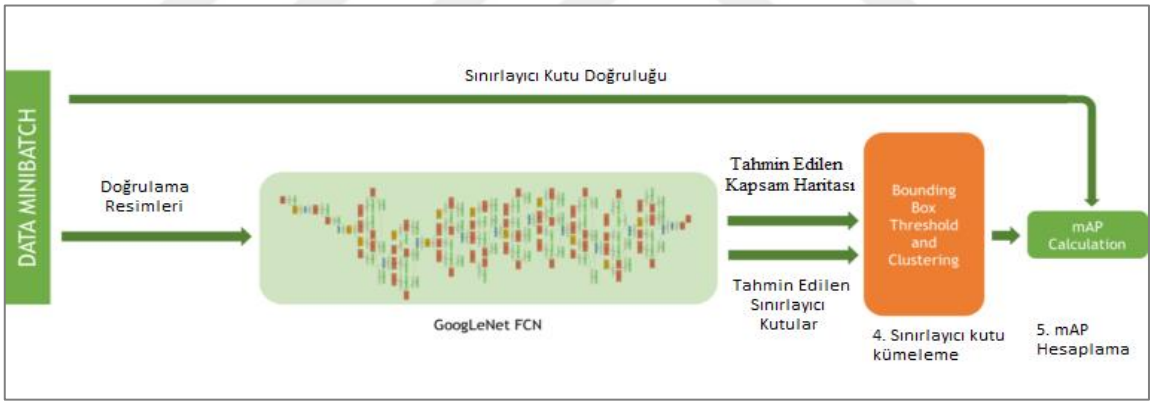
1. Veri (data) katmanı, eğitim resimlerini ve etiketleri alır. Veri genişletme (online data augmentation) katmanı, veri artırma işlemini gerçekleştirir.
2. Tam evrişimsel ağ (fully-convolutional network - FCN), nesne sınıflarının ve sınırlayıcı kutuların özellik çıkarımı ve tahminini gerçekleştirir. FCN, tamamen bağlı katmanları olmayan bir evrişimsel sinir ağıdır. Bu, ağın değişen boyutlara sahip girdi görüntülerini kabul edebileceği ve etkili bir biçimde bir evrişimsel sinir ağının kayan pencere biçiminde uygulanabileceği anlamına gelir.
3. Kayıp fonksiyonları (loss functions), nesne kapsamını ve sınırlayıcı kutunun köşelerini tahmin etmedeki hatayı ölçer.



Şekil 3.12. Eğitim için DetectNet yapısı

Şekil 3.13 doğrulama sırasındaki DetectNet mimarisini ve iki önemli süreci daha göstermektedir.

4. Kümeleme fonksiyonu (clustering function), doğrulama sırasında tahmin edilen sınırlayıcı kutuların son kümesini üretir.
5. Ortalama kesinlik değeri (mean Average Precision - mAP), doğrulama veri setine karşı model performansını ölçmek için hesaplanır.



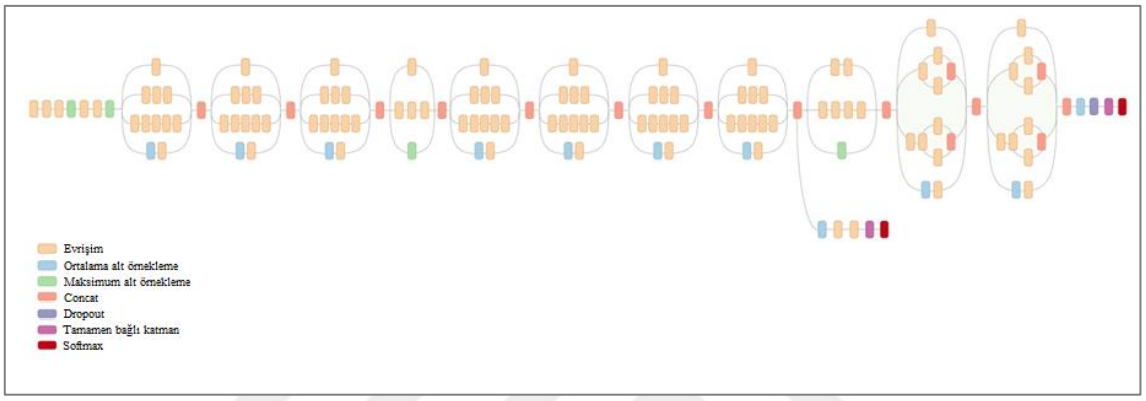
Şekil 3.13. Doğrulama için DetectNet yapısı

DetectNet FCN alt ağı, veri girişi katmanları, son alt örnekleme katmanı ve çıktı katmanları olmadan GoogLeNet'le aynı yapıdadır. Yani, GoogLeNet ağının bir uzantısıdır. Bu, DetectNet'in önceden eğitilmiş bir GoogLeNet modeli kullanılarak başlatılmasına ve böylece eğitim süresinin azalmasına ve nihai model doğruluk oranının artırılmasına büyük yarar sağlar.

Derin sinir ağları nadiren sıfırdan eğitilir. Bunun nedeni, gereken ağ derinliği için gerekli olan yeterli veri kümesinin bulunmasının zor olması ve yeterli miktarda veri olsa bile eğitimin GPU'larda 2-3 haftayı bulabilmesidir. Bunun yerine, çok büyük bir veri

kümesiyle önceden eğitilen derin sinir ağını edinmek ve daha sonra eğitilmiş ağın ağırlıklarını ilgili görev için ya bir ağırlık ilklendirici ya da sabit bir özellik çıkarıcı olarak kullanmak yaygın bir işlemdir.

GoogLeNet 22 katmanlı bir evrişimsel sinir ağıdır ve %5,7 hata oranı ile ILSVRC 2014'ün galibi olmuştur. Şekil 3.14'te GoogLeNet mimarisi gösterilmiştir. GoogLeNet, evrişimsel sinir ağı katmanlarının ardışık olarak yığılması gerekmediği fikrini ortaya atan ilk modellerden biridir.



Şekil 3.14. GoogLeNet Ağı

DetectNet, optimizasyonda kullanılacak son kayıp fonksiyonunu üretmek için iki ayrı kayıp fonksiyonunun lineer bir kombinasyonunu kullanır.

- **coverage_loss**, bir eğitim verileri örneğindeki tüm ızgaralı karelerdeki gerçek ve tahmin edilen nesne kapsamları arasındaki farkların karelerinin toplamıdır (Denklem 3.13).

$$\text{coverage\_loss} = \frac{1}{2N}\sum_{i=1}^{N}\left|coverage_i^t - coverage_i^p\right|^2, \qquad (3.13)$$

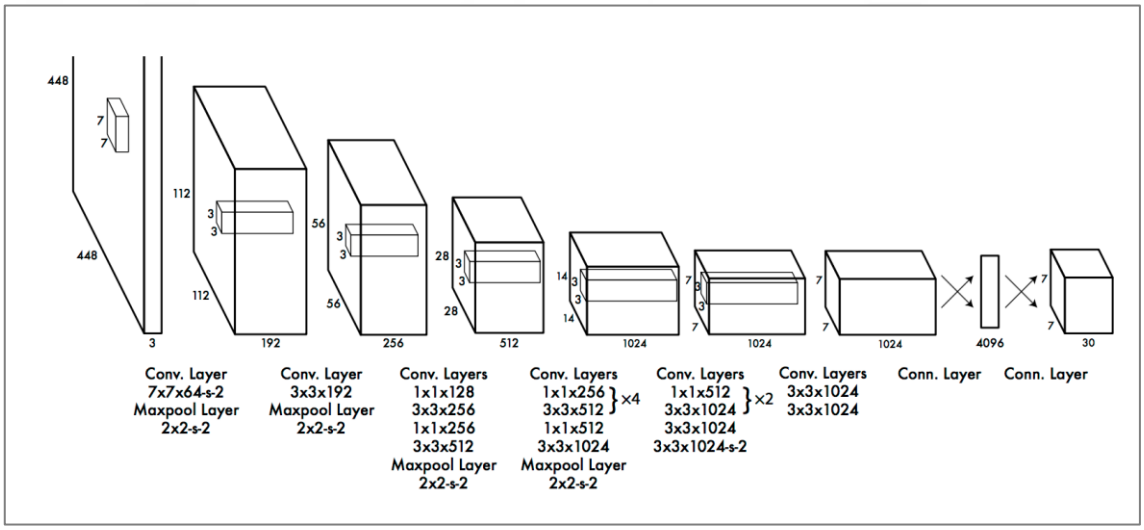
- **bbox_loss**, her bir ızgara karesi tarafından kapsanan nesne için sınırlayıcı kutusunun köşelerinin gerçek ve tahmini farkı için ortalama L1 kaybıdır (ortalama mutlak fark) (Denklem 3.14).

$$\text{bbox\_loss} = \frac{1}{2N}\sum_{i=1}^{N}\left[\left|x_1^t - x_1^p\right| + \left|y_1^t - y_1^p\right| + \left|x_2^t - x_2^p\right| + \left|y_2^t - y_2^p\right|\right], \qquad (3.14)$$

Eğitimin amacı, Denklem 3.13 ve Denklem 3.14'teki değerlerin minimize edilmesidir.

3.3.3. YOLO

YOLO, sınırlayıcı kutuları ve sınıf olasılıklarını tahmin etmek için tek bir sinir ağı kullanan, GoogLeNet'ten ilham alan bir derin evrişimsel sinir ağıdır. YOLO'nun açılımı, "You Only Look Once" ("Sadece Bir Kez Bak")'tır. Bu, görüntünün sadece bir kez ağ içinden geçip algılama görevini tamamladığı anlamına gelir. Literatürdeki en hızlı genel amaçlı nesne algılama mimarisidir. Ayrıca, rapor edilen ilk gerçek zamanlı evrişimsel sinir ağı tabanlı nesne algılama modelidir. Tüm algılama hattı tek bir ağda olduğundan doğrudan algılama performansı diğer sinir ağı mimarilerinin çoğundan yüksektir (Redmon ve ark.,2016).



Şekil 3.15. YOLO sinir ağı mimarisi

YOLO mimarisi, resim sınıflandırması için GoogLeNet modelinden esinlenmiştir. YOLO sinir ağı modeli 24 evrişim katmanına ve ardından 2 tamamen bağlı katmana sahiptir. Ağın, başlangıçtaki evrişim katmanları görüntünün özelliklerini çıkartırken, tamamen bağlantılı katmanlar çıkış olasılıklarını ve koordinatları öngörürler.

YOLO, C ve CUDA ile yazılmış açık kaynaklı bir sinir ağı uygulama çatısı olan "Darknet" üzerinde uygulanmaktadır. YOLO bu şekilde uygulanarak, gerçek zamanlı video akışını işlemek için GPU işlem gücünden yararlanmaktadır.

3.4. Kullanılan Materyaller

3.4.1. Bilgisayar

Sinir ağının eğitimi için kullanılan bilgisayar ve bilgisayarda bulunan GPU'nun teknik özellikleri sırasıyla Çizelge 3.1 ve Çizelge 3.2'de belirtilmiştir.

Çizelge 3.1. Bilgisayarın teknik özellikleri

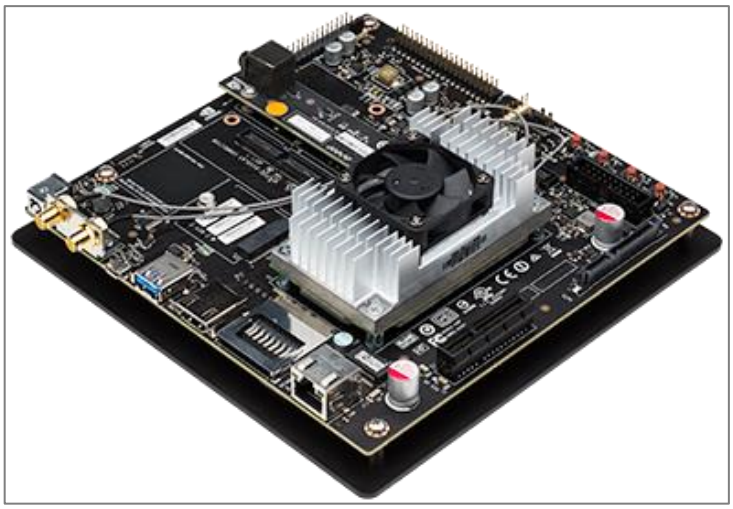
İşlemci	: Intel Skylake Core i7-6700HQ
İşlemci hızı	: 2.6 GHz
Ön bellek	: 6 MB
Bellek	: 8GB (1x8GB) DDR4L 1.2V 2133MHz
Disk kapasitesi ve arabirimi	: Intel 600P Serisi 256GB ve 1TB Hitachi Travelstar
Ekran kartı	: 6GB GDDR5 NVIDIA GeForce GTX1060
Ekran	: 15.6" Full HD 1920x1080 G-SYNC IPS Mat LED Ekran
İşletim sistemi	: Ubuntu 14.04 LTS 64-bit
Kullanım tipi	: Dizüstü Bilgisayar
Üretici adı ve modeli	: Monster Tulpar T5 V8.1

Çizelge 3.2. GPU'nun teknik özellikleri

Standart Saat Hızı (MHz)	: 1506 MHz
Bellek	: 6 GB GDDR5
Bellek Bant Genişliği	: 192 GB/sn
Bellek hızı	: 8 Gbps
CUDA çekirdeği	: 1280
CUDA hesaplama kapasitesi	: 6.1
Üretici adı ve modeli	: NVIDIA GeForce GTX 1060 6Gb

3.4.2. Gömülü Kart

Algılama sistemleri yüksek işlem kapasitesine sahip olduklarından dolayı, uygulamalar bunu karşılayacak bir donanım olan Jetson TX1 üzerinde gerçekleştirilmiştir. Jetson TX1, düşük güçle çalışan yüksek hesaplama performansı gerektiren uygulamalar için Nvidia tarafından önerilen bir geliştirme kartıdır. Bu kartın teknik özellikleri Çizelge 3.3'te belirtildiği gibidir.



Şekil 3.16. Jetson TX1

Çizelge 3.3. Jetson TX1'in teknik özellikleri

CPU	: Quad ARM® A57/2 MB L2
GPU	: NVIDIA Maxwell ™, 256 CUDA cores
Bellek	: 4 GB 64 bit LPDDR4 | 25.6 GB/s
Video kod çözümü	: 4K 60 Hz
Video kodlama	: 4K 30 Hz
CSI	: 6'ya kadar kamera | 1400 Mpix/s
Display	: 2x DSI, 1x eDP 1.4, 1x DP 1.2/HDMI
Bağlanabilirlik	: 1 Gigabit Ethernet, 802.11ac WLAN, Bluetooth
PCIE	: Gen 2 1x1 + 1x4
Depolama	: 16 GB eMMC, SDIO, SATA
Diğer	: 3x UART, 3x SPI, 4x I2C, 4x I2S, GPIOs

3.4.3. Kamera

Jetson TX1 kartına görüntü sağlamak üzere Çizelge 3.4'teki teknik özelliklere sahip kamera kullanılmıştır.

Şekil 3.17. Kamera

Çizelge 3.4. Kameranın teknik özellikleri

Video Formatı	: MOV
Video Çözünürlüğü	: 1080P(1920*1080)30fps, 720P(12080*720)60/30fps, WVGA(640*480)60 fps ,QVGA(640*480)fps
Sıkıştırılmış Video Formatı	: H.264
Görüntü Formatı	: JPG
Görüntü Çözünürlüğü	: 14 MP/12MP / 10 MP / 8 MP / 5 MP / 3 MP / 2 MHD / 1.3 MP / VGA
Depolama Formatı	: MicroSD 32GB'e kadar
Üretici adı ve modeli	: SJCAM Sj5000

3.5. Veri Setinin Hazırlanması

3.5.1. DetectNet - Veri Seti Hazırlanması

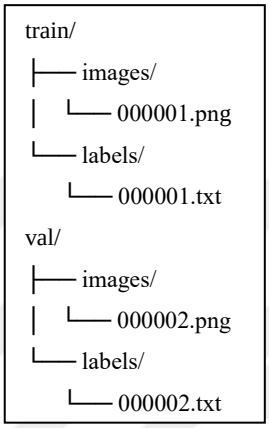
Uygulamada kullanılacak DetectNet sinir ağı modelini eğitmek üzere çeşitli kaynaklardan Şekil 3.18'deki gibi içerisinde silah nesnesi bulunan 2605 adet resim dosyası toplandı. C# programlama dili ile yazdığımız program yardımıyla resimlerdeki silah nesnelerinin bulunduğu koordinatları kaydettiğimiz metin dosyaları oluşturuldu (Şekil 3.19). Resim dosyası ile resme ait etiket dosyasının aynı isme sahip olması sağlandı. Uygulamada 5-katlamalı çapraz doğrulama testi uygulandı. Bunun için, oluşan etiket dosyaları resim dosyalarıyla birlikte %80'i eğitim veri seti, %20'si ise doğrulama veri seti olmak üzere Şekil 3.20'de gösterilen klasör düzeninde ikiye ayrıldı. Her kat için bu oran korunarak veri setinin farklı bölümleri eğitim ve doğrulama amaçlı olarak kullanıldı.

Şekil 3.18. Veri setinden bazı örnekler

```
silah 0.0 0 0.0 139 162 203 222 0.0 0.0 0.0 0.0 0.0 0.0 0.0
silah 0.0 0 0.0 163 213 240 281 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

Şekil 3.19. Örnek bir etiket dosyası içeriği

Etiketleme işleminde, silah nesnesinin bulunduğu koordinatları tutan kısım dışındaki yerler basitlik açısından 0 olarak ayarlandı. Etiket dosyasının taşıdığı tüm bilgiler Çizelge 3.5'de izah edilmiştir.



Şekil 3.20. Veri seti klasörünün içerik düzeni

Çizelge 3.5. Etiket dosyasındaki sütunların taşıdığı bilgiler

Sütun No	İsim	Tanım
1	type	: Nesnenin tipini tanımlar
2	truncated	: 0 (non-truncated) ile 1 (truncated) arasında değer alır, truncated resim sınırlarını terkeden nesneyle ilgilidir
3	occluded	: Kapanma durumunu belirtir: 0 = tamamen görünür, 1 = kısmen kapanmış 2 = büyük ölçüde kapanmış, 3 = bilinmeyen
4	alpha	: nesnenin görünüm açısı, (-pi, pi) aralığında
5-8	bbox	: Resimdeki nesnenin 2D sınırlayıcı kutusu, sırasıyla sol, üst, sağ, alt piksel koordinatları
9-11	dimension	: 3D nesne boyutları: yükseklik, genişlik, uzunluk (metre cinsinden)
12-14	location	: 3D nesne konumu: x, y, z kamera koordinatları (metre cinsinden)
15	rotation_y	: Kamera koordinatlarındaki Y ekseni etrafında dönme (-pi, pi) aralığında

Hazırlanan veri setini nesne algılamada kullanmak üzere hazır hale getirmek için Nvidia Digits programı kullanıldı. Veri seti oluşturma sayfasında eğitim ve doğrulama kümeleri için resim ve etiket klasörlerinin yolları belirtilerek onay verildikten sonra veri seti, DetectNet modelini eğitmek üzere hazır hale getirilmiş oldu.

3.5.2. YOLO - Veri Seti Hazırlanması

Toplanan 2605 adet resmin etiketleme işlemi Şekil 3.21'de gösterilen formatta gerçekleştirildi. Etiket formatındaki yer alan sütunların taşıdığı bilgilerin izahı Çizelge 3.6'da verilmiştir. Sonra Şekil 3.22'deki kod parçası tüm etiket dosyaları için çalıştırılarak etiket dosyalarındaki veriler eğitimde kullanılacak formata dönüştürülmüş oldu.

Etiketlenen resimlerin %80'i eğitim veri seti, %20'si doğrulama veri seti olmak üzere Şekil 3.20'deki düzene göre klasörlere tahsis edildi. Uygulamada 5-katlamalı çapraz doğrulama testi gerçekleştirildiğinden dolayı her kat için bu oran korunarak veri setinin farklı bölümlerinin eğitim ve doğrulama amaçlı olarak kullanılması sağlandı. Sonra klasörlere tahsis edilen tüm resimlerin dosya yolunu gösteren "train.txt" ve "val.txt" isimli metin dosyaları oluşturuldu.

<type> <x> <y> <width> <height>

Şekil 3.21. Darknet'te kullanılacak etiket formatı

Çizelge 3.6. Etiket dosyasındaki sütunların taşıdığı bilgiler

Sütun No	İsim	Tanım
1	type	: Nesnenin tipi
2-3	x-y	: Resimdeki nesnenin 2D sınırlayıcı kutusunun sol üst Koordinatı
4	width	: 2D sınırlayıcı kutunun genişliği
5	height_	: 2D sınırlayıcı kutunun boyu

```
def convert(size, box):
    dw = 1./size[0]
    dh = 1./size[1]
    x = (box[0] + box[1])/2.0
    y = (box[2] + box[3])/2.0
    weight = box[1] - box[0]
    height = box[3] - box[2]
    x = x*dw
    weight = weight *dw
    y = y*dh
    height = height *dh
    return (x,y,weight,h)
```

Şekil 3.22. Darknet'te etiket dosyası dönüşüm kodu

Darknet yazılımındaki data klasöründe "model.names" isminde bir dosya oluşturularak, içerisine algılatmak istediğimiz nesnelerin isimleri yazıldı. Biz tek türde

bir nesne algılatacağımız için dosyanın içerisine “silah” yazılıp kaydedildi. Sonra yine data klasörü içerisinde veri setiyle ilgili çeşitli bilgileri tutacağımız “model.data” dosyası oluşturuldu. “model.data” içerisine algılanacak nesne sayısı (classes = 1), eğitim ve doğrulama verilerinin dosya yollarını barındıran “train.txt” ve “val.txt” metin dosyalarının yolları (train = /train.txt, valid = /val.txt), algılanacak nesnelerin isimlerini barındıran dosyanın yolu (names = data/model.names) ve eğitilecek modelin kaydedileceği dizin (backup = backup) eklendi (Şekil 3.23).

```
classes  = 1
train    = /train.txt
valid    = /val.txt
names    = data/model.names
backup   = backup
```

Şekil 3.23. “model.data” dosyası içeriği

3.6. Model Oluşturma ve Eğitim

3.6.1. DetectNet Silah Algılama Modeli Eğitimi

DetectNet sinir ağı modelini eğitmek için Nvidia Digits programı kullanıldı. Nvidia Digits’te nesne algılama için model oluşturma sayfası açılarak aşağıdaki ayarlamalar yapıldı.

- Select Dataset : silah_veritabani
- Training epochs : 100
- Subtract Mean : none
- Solver Type : Stochastic gradient descent (SGD)
- Base learning rate : 0.0001

DetectNet ağının varsayılan yığın boyutu 10'dur ve eğitim sırasında 12GB'a kadar GPU belleği tüketir. Bizim eğitimde kullandığımız GPU belleğimiz 6GB olduğu için yığın biriktirme (batch accumulation) alanını 5, yığın boyutunu (batch size) 2 olarak ayarlayarak eğitimin 6GB bellekli GPU’da gerçekleştirilmesi sağlandı.

Model oluşturmak için DetectNet sinir ağı kullanıldı. Bunun için “Custom Networks” sekmesinin “Caffe” alt sekmesindeki metin alanına DetectNet sinir ağı modeli eklendi. DetectNet, GoogleNet’ten türetilmiş bir ağ olduğundan dolayı eğitimi hızlandırmak için Berkeley İmgeleme ve Öğrenme Merkezi (BVLC) tarafından

yayınlanan, ImageNet görüntü koleksiyonuyla daha önce eğitilmiş bir GoogLeNet sinir ağı modeli, eğitilecek Detectnet ağı için başlangıç ağırlık değerlerinin ilklendirilmesinde kullanıldı. Bu, "Pretrained model(s)" kısmına önceden eğitilmiş GoogLeNet sinir ağı modelinin dosya yolu eklenerek sağlandı. Tüm bu ayarlamalardan sonra eğitim sürecine geçildi. 5 saat 43 dakika süren eğitimin ardından modelimiz test edilmek ve Jetson TX1 kartında uygulanmak üzere hazır hale geldi.

3.6.2. YOLO Silah Algılama Modeli Eğitimi

Nesne algılama modeli olarak, "Darknet" uygulama çatısıyla birlikte gelen, içerisinde YOLO sinir ağı modelini barındıran "yolo.cfg" dosyası kullanıldı. Dosya içerisinde; tek bir türde nesne algılaması gerçekleştireceğimiz için sınıf sayısında ve yine sınıf sayısıyla alakalı olarak 237. satırdaki "filters" değerinde, [filters = (class + 5) * 5] işlemi gereğince Şekil 3.24'te gösterildiği gibi değişiklikler yapıldı.

```
[convolutional]
filters=30

[region]
classes=1
```

Şekil 3.24. "yolo.cfg" dosyasında gerçekleştirilen değişklikler

Ayrıca ImageNet veri seti kullanılarak önceden eğitilmiş "darknet19_448.conv.23" isimli evrişimsel ağırlıkları barındıran dosya, nesne algılama ağının başlangıç ağırlıklarının ilklendirilmesinde kullanıldı. Terminal ekranına yazılan aşağıdaki (Şekil 3.25) komutla veri seti için hazırladığımız dosya, YOLO sinir ağının bulunduğu dosya ve önceden eğitilmiş ağırlıkları bulunduran dosya bir araya getirilerek eğitim başlatıldı.

```
./darknet detector train cfg/model.data cfg/yolo.cfg darknet19_448.conv.23
```

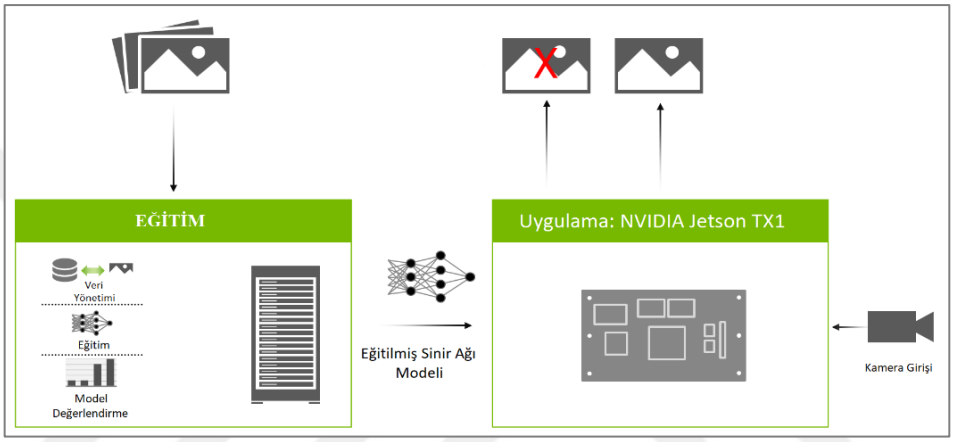
Şekil 3.25. YOLO sinir ağı eğitim komutu

3 saat 5 dakika süren eğitim tamamlandıktan sonra kendi veri setimizle eğitilmiş olan, güncel evrişimsel ağırlıkları bulunduran YOLO modelimiz oluşmuş oldu.

4. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

Bu bölümde tez için geliştirilen uygulamaların sonuç çıktılarına, test görsellerine ve karşılaştırmalı test sonuçlarına yer verilmiştir.

Uygulama verilerinin hazırlanması, modelin eğitimi ve ilgili testlerin yapılması GEFORCE GTX 1060 GPU'ya sahip bilgisayar üzerinde, eğitilen modelin gerçek zamanlı olarak çalıştırılması Jetson TX1 üzerinde yapılmıştır. İş akışı Şekil 4.1'de verilmiştir.



Şekil 4.1. Uygulama iş akışı

4.1. Ortak Performans Değerlendirme Ölçüleri

DetectNet ve YOLO sinir ağı modellerini değerlendirmek için ortak olarak kullanılan ölçümler aşağıda izah edilmiştir;

- **Doğru Pozitif (DP):** Gerçekte silah nesnesi olan görüntülerdeki nesnelerden sistemin bulduğu silah nesne sayısıdır.
- **Doğru Negatif (DN):** Gerçekte silah nesnesi olmayan ve sistemin de silah nesnesi kabul etmediği nesne sayısıdır.
- **Yanlış Pozitif (YP):** Gerçekte silah nesnesi olmadığı halde sistemin silah nesnesi olarak algıladığı nesne sayısıdır.
- **Yanlış Negatif (YN):** Gerçekte silah nesnesi olduğu halde sistemin silah nesnesi olarak algılayamadığı nesne sayısıdır.

- **Duyarlılık (Recall):** Nesne algılayıcısının nesneyi belirleyebilme oranı, doğru olarak bulunan nesnelerin toplam bulunması gereken nesne sayısına oranıdır.

$$Duyarlılık = \frac{DP}{DP+YN} \tag{4.1}$$

- **Pozitif tahmin değeri (Precision)**: Doğru tanımlanmış nesnelerin toplam tahmin edilen nesnelerin sayısına oranıdır.

$$Pozitif\ tahmin\ değeri = \frac{DP}{DP+YP} \tag{4.2}$$

- **F1 ölçümü**: Pozitif tahmin değeri ve duyarlılık değerleri kullanılarak elde edilen bir testin doğruluğunun ölçüsüdür.

$$F1\ ölçümü = 2 \times \frac{pozitif\ tahmin\ değeri \times duyarlılık}{pozitif\ tahmin\ değeri + duyarlılık} \tag{4.3}$$

4.2. DetectNet Modeli Sonuç Çıktıları

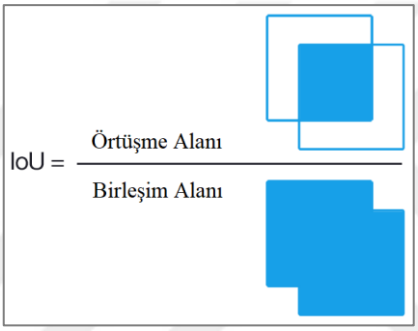
5-katlamalı çapraz doğrulama (5-fold validation) uygulanarak elde edilen DetectNet sinir ağı modelinin ortalama kesinlik (mAP) değeri %56.15, pozitif tahmin (precison) değeri %70.34, duyarlılık(recall) değeri %69.48 olarak bulundu. DetectNet modeli Jetson TX1'de uygulanmak üzere hazır hale getirilmiş oldu.

mAP, DetectNet için duyarlılık ve pozitif tahmin değerlerinin çarpımına dayanan basitleştirilmiş ortalama averaj hassasiyet puanıdır. Ağın, ilgilendiğimiz nesneler için ne kadar duyarlı olduğunun ve yanlış alarmlardan ne kadar iyi kaçındığının iyi bir kombine ölçüsüdür. Ağın doğru sınırlayıcı kutularla nesneleri tespit etmeyi öğrendiği en önemli gösterge, sıfır olmayan bir mAP değeridir.

4.3. YOLO Modeli Sonuç Çıktıları

5-katlamalı çapraz doğrulama (5-fold validation) uygulanarak elde edilen YOLO silah algılama modelinin sonuç çıktıları; RPs/Img değeri 84.35, IoU değeri %69.25, duyarlılık (recall) değeri %92.48 olarak bulunmuştur.

- RPs/Img, ortalama olarak görüntü başına kaç sınırlayıcı kutu önerildiğini belirtir.
- IoU, sistemin nesne için tahmin ettiği sınırlayıcı kutularla, nesnelerin gerçek sınırlayıcı kutuları arasındaki ortalama örtüşme oranını belirtir (Şekil 4.2). IoU nesne algılama sistemlerini değerlendirmek için önemli bir metriktir.



Şekil 4.2. IoU değerinin temsili gösterimi

- Pay (örtüşme alanı), tahmini sınırlayıcı kutu ile gerçek sınırlayıcı kutu arasındaki kesişim alanını ifade eder. Payda (birleşim alanı), hem tahmini sınırlayıcı kutu hem de gerçek sınırlayıcı kutu tarafından kapsanan alandır. Örtüşen alanın birleşim alanı ile bölünmesi IoU değerini verir.
- YOLO modelinin eğitimi sonucunda elde edilen IoU ise, tüm IoU değerlerinin ortalamasıdır.

4.4. Test Görselleri

4.4.1. DetectNet Test Görselleri

DetectNet silah algılama modelinin eğitiminin tamamlanmasının ardından, tekil resimlerle gerçekleştirilen testlerden bazı resimler Şekil 4.3'te gösterilmiştir.

Şekil 4.3. Bazı test görüntüleri (DetectNet)

4.4.2. YOLO Test Görselleri

YOLO silah algılama modelinin eğitiminin tamamlanmasının ardından, tekil resimlerle gerçekleştirilen testlerden bazı resimler Şekil 4.4'te gösterilmiştir.

Şekil 4.4. Bazı test görüntüleri (YOLO)

4.5. Modellerin Gömülü Sisteme Yüklenmesi

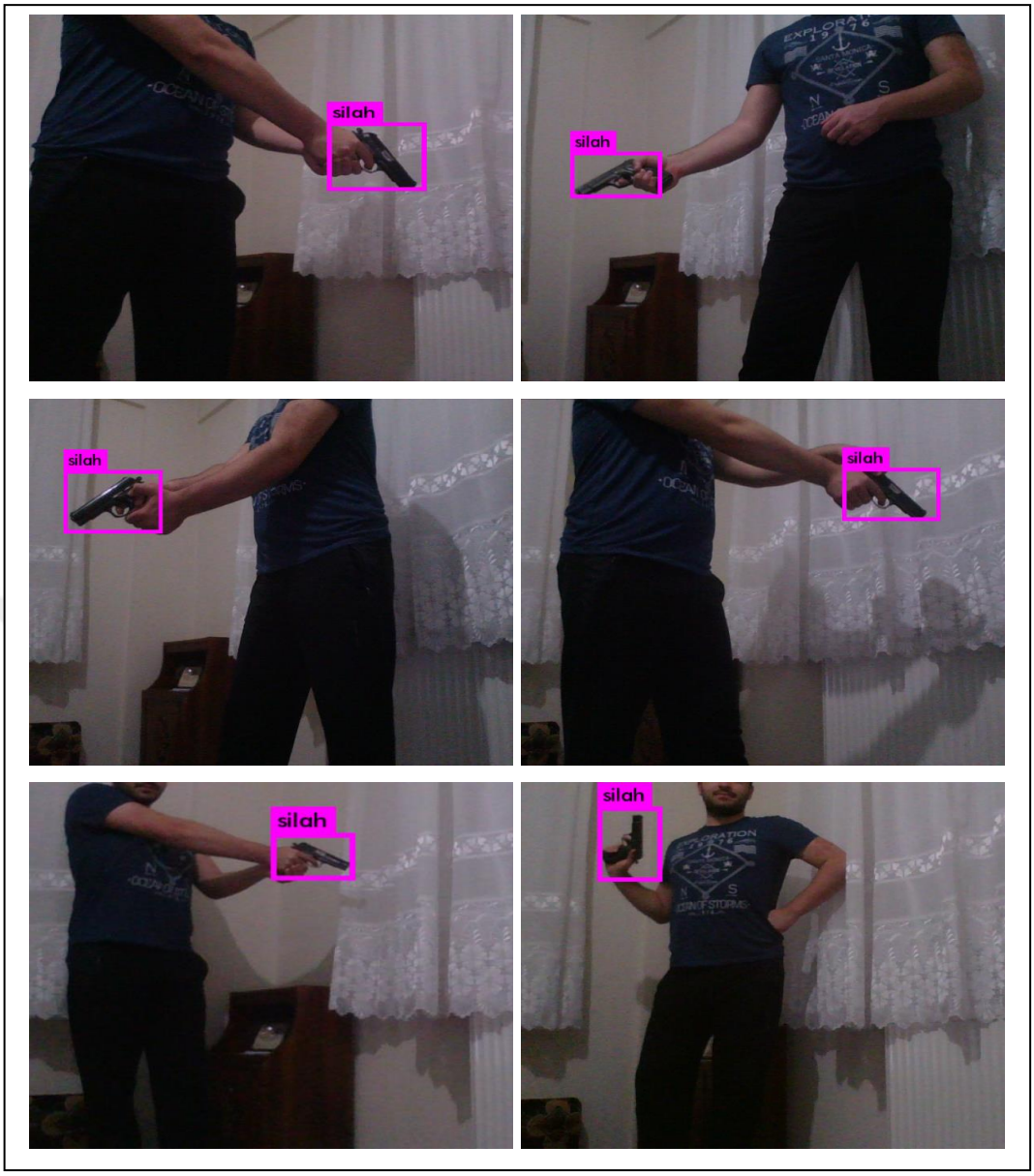
4.5.1. DetectNet Modelinin Gömülü Sisteme Yüklenmesi

Eğitilen DetectNet sinir ağı modelinin Jetson TX1 üzerine kurulan “jetson-inference” açık kaynak kodlu yazılımıyla gerçek zamanlı olarak çalışması gerçekleştirilmiştir. Bu yazılım canlı kamera akışında Googlenet / Alexnet'in çalışması için derin öğrenme görevlerini sağlamaktadır. Şekil 4.5’te kamera akışından beslenerek gerçek zamanlı çalışan uygulamadan elde edilen bazı resim kareleri gösterilmiştir.

Şekil 4.5. Jetson TX1'de çalışan uygulamadan bazı görüntüler (DetectNet)

4.5.2. YOLO Modelinin Gömülü Sisteme Yüklenmesi

YOLO modelinin Jetson TX1 kartına yüklenmesi için sinir ağının eğitiminde de kullanılan "Darknet" yazılımı kullanıldı. Bu yazılım canlı kamera akışında YOLO modelinin yürütülmesine olanak tanımaktadır. Şekil 4.6'da kamera akışından beslenerek gerçek zamanlı çalışan uygulamadan elde edilen bazı örnek resim kareleri gösterilmiştir.



Şekil 4.6. Jetson TX1'de çalışan uygulamadan bazı görüntüler (YOLO)

4.6. Tartışma

DetectNet ve YOLO modellerinin, toplam 370 tane silah nesnesi barındıran 300 adet resimle gerçekleştirilen test sonuçları Çizelge 4.1'de gösterilmiştir.

Çizelge 4.1. DetectNet ve YOLO karşılaştırma – 1

Model	Doğru Pozitif	Yanlış Negatif	Yanlış Pozitif
DetectNet	238	132	8
YOLO	207	163	15

Toplamda 370 adet silah nesnesi bulunan 300 resimden, DetectNet bu nesnelerin 238 tanesini doğru olarak tahmin ederken 132 tanesini tahmin edememiştir. YOLO ise 207 silah nesnesini doğru tahmin etmiş, 163 tanesini tahmin edememiştir. DetectNet'in tahmin ettiği nesnelerin 8 tanesi silah nesnesine ait değilken YOLO'da bu sayı 15'tir. Bu sonuçlara göre pozitif tahmin değeri (precision), duyarlılık (recall) ve F1 ölçümünün değerlerinin karşılaştırılması Çizelge 4.2'de verilmiştir.

Çizelge 4.2. DetectNet ve YOLO karşılaştırma – 2

Model	Pozitif Tahmin	Duyarlılık	F1 ölçümü
DetectNet	%96.74	%64.32	%77.26
YOLO	%93.24	%55.94	%69.92

Olmos ve ark., (2017) çeşitli filmlerden elde ettikleri, içerisinde silah nesnesi bulunan video kesitlerini kullanarak kendi çalışmalarını test etmiştir. Bu video görüntülerinin çerçeve sayısı ve içerdikleri toplam silah nesne sayıları Çizelge 4.3'te verilmiştir. Bu videolar James Bond: The World is Not Enough (video 1), Pulp Fiction (video 2, 3 ve 4), Impossible Mission: Rogue Nation (video 5) ve Mr. Bin (video 6) filmlerinden alınan kesitlerden oluşmaktadır.

Çizelge 4.3. Test videoları

Video	Çerçeve Sayısı	Nesne Sayısı
1	393	162
2	627	778
3	441	58
4	591	54
5	627	105
6	212	290

Olmos ve arkadaşlarının Çizelge 4.3'teki özelliklere sahip video görüntülerini kullanarak gerçekleştirdikleri testin sonuçları Çizelge 4.4'te gösterilmiştir. Olmos ve arkadaşlarının kullandıkları bu video görüntüleri kullanılarak DetectNet ve YOLO sinir ağı modelleriyle gerçekleştirilen test sonuçları da sırasıyla Çizelge 4.5 ve Çizelge 4.6'da verilmiştir.

Çizelge 4.4. Faster R-CNN silah algılama modeliyle gerçekleştirilen test sonuçları (Olmos ve ark., 2017)

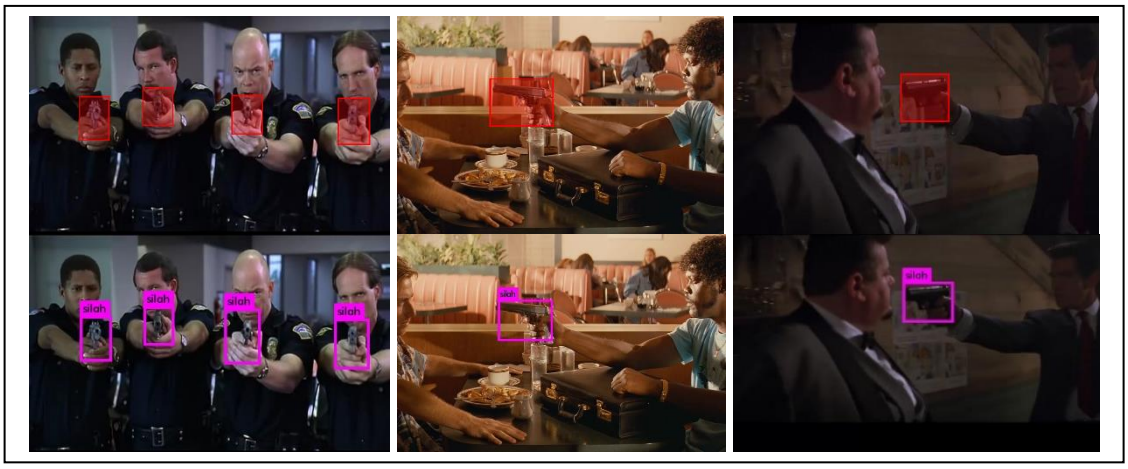
Video	Doğru Pozitif	Yanlış Pozitif	Pozitif Tahmin	Duyarlılık	F1 ölçümü
1	60	8	%88.24	%37.04	%52.17
2	467	11	%98.70	%60.03	%74.36
3	25	15	%62.50	%43.10	%51.02
4	6	0	%100.00	%11.11	%20.22
5	24	21	%53.33	%22.86	%32.00
6	141	30	%82.46	%48.62	%61.17

Çizelge 4.5. DetectNet silah algılama modeliyle gerçekleştirilen test sonuçları

Video	Doğru Pozitif	Yanlış Pozitif	Pozitif Tahmin	Duyarlılık	F1 ölçümü
1	70	19	%78.65	%43.20	%55.77
2	531	33	%94.14	%68.25	%79.13
3	21	12	%63.63	%36.20	%46.15
4	5	41	%10.86	%9.25	%10.00
5	46	30	%60.52	%43.80	%50.82
6	212	10	%95.49	%73.10	%82.81

Çizelge 4.6. YOLO silah algılama modeliyle gerçekleştirilen test sonuçları

Video	Doğru Pozitif	Yanlış Pozitif	Pozitif Tahmin	Duyarlılık	F1 ölçümü
1	52	0	%100.00	%32.09	%48.59
2	347	5	%98.57	%44.60	%61.41
3	21	6	%77.77	%36.20	%49.41
4	0	16	%0.00	%0.00	%0.00
5	15	21	%41.66	%14.28	%21.27
6	269	57	%82.51	%92.75	%87.33



Şekil 4.7. DetectNet (üstte) ve YOLO (altta) modelleriyle işlenen test videolarından bazı kareler

Çizelge 4.4, Çizelge 4.5 ve Çizelge 4.6'da gösterilen modellerin test edilmesinde kullanılan test videoları toplam 2891 çerçeve sayısına sahiptir ve 1447 silah nesnesi içermektedir. Modellerin tüm videolardaki test sonuçları toplamları Çizelge 4.7'de gösterilmiştir.

Çizelge 4.7'ye bakıldığında doğru pozitif sayısı bakımından, DetectNet modeli Faster R-CNN ve YOLO'dan daha üstündür. Yanlış pozitifleri bulma noktasında ise DetectNet, Faster R-CNN ve YOLO'ya göre daha kötü bir performans sergilemiştir. Yanlış pozitif sayısına göre en iyi modelin Faster R-CNN olduğu görülmektedir. F1 ölçümüne göre ise, DetectNet modeli %71.45'e sahip bir değerle, %64.12 değerine sahip Faster R-CNN ve %62.41 değerine sahip YOLO'dan daha yüksek sonuç elde etmiştir.

Çizelge 4.7. Faster R-CNN, DetectNet ve YOLO modellerininin toplam test sonuçları

Model	Doğru Pozitif	Yanlış Pozitif	Pozitif Tahmin	Duyarlılık	F1 ölçümü
Faster R-CNN	723	85	%89.48	%49.96	%64.12
DetectNet	885	145	%85.92	%61.16	%71.45
YOLO	704	105	%87.02	%48.65	%62.41

Çizelge 4.7'ye göre, Faster R-CNN algoritması F1 ölçümüne göre YOLO'dan daha yüksek değere sahiptir ve yanlış pozitif sayısı bulma bakımından en başarılı modeldir. Ancak PASCAL VOC 2007 veri setiyle eğitilen Faster R-CNN ve YOLO modellerinin karşılaştırıldığı, Redmon ve ark., (2016) tarafından gerçekleştirilen çalışmada Faster R-CNN'nin FPS bakımından YOLO'nun çok gerisinde olduğu görülmektedir (Çizelge 4.8). Dolayısıyla Faster R-CNN nesne algılama modeli Jetson TX1 kartında gerçek zamanlı çalışacak bir uygulama için, DetectNet ve YOLO modelleri kadar performans göstermeyecektir.

Çizelge 4.8. Faster R-CNN ve YOLO karşılaştırması (PASCAL VOC 2007'de gerçek zamanlı sistemler) (Redmon ve ark.,2016)

Model	mAP	FPS
Faster R-CNN	73.2	7
YOLO	63.4	45

Ayrıca DetectNet ve YOLO nesne algılama modelleri, diskte kapladıkları boyut ve Jetson TX1 kartında çalışmasındaki saniyede işlenen görüntü sayısı bakımından değerlendirilmiştir. Elde edilen sonuçlar Çizelge 4.9'da gösterilmiştir. DetectNet

modelinin saniyede işleyebildiği görüntü sayısının yani FPS miktarının YOLO'dan daha fazla olduğu görülmektedir.

Çizelge 4.9. DetectNet ve YOLO karşılaştırma – 3 (Jetson TX1)

Model	Boyut	FPS
DetectNet	29.8 MB	7
YOLO	202.2 MB	4

Çizelge 4.9'dan da anlaşılacağı üzere; DetectNet modeli Jetson TX1 kartında, YOLO modeline göre daha yüksek fps'de çalışmıştır. Yani Jetson TX1 kartının DetectNet ile saniyede işleyebileceği görüntü sayısı YOLO modeline kıyasla daha fazladır. Bu da kameradan alınan görüntülerin DetectNet modelinde, YOLO modeliyle kıyaslandığında, daha az kayıpla işlenebildiği anlamına gelir. Ayrıca DetectNet modeli YOLO modeline göre hafızada daha az yer kaplamıştır. Her ne kadar hafıza miktarı Jetson TX1 için yeterli olsa ve gerektiğinde harici hafıza birimleriyle artırılsa bile, gömülü sistemlerin sınırları göz önünde bulundurulduğunda modellerin daha düşük boyutlu olması bu tarz sistemler için önemli bir artıdır.

5. SONUÇLAR VE ÖNERİLER

5.1 Sonuçlar

GPU'ların CUDA sayesinde genel amaçlı programlanabilir yapıya kavuşması, GPU'ları yüksek hesaplama gücü gerektiren ve paralelleştirmeye müsait problemlerin çözülmesinde oldukça kullanışlı hale getirmiştir. GPU'lar, günümüzde popülaritesi gittikçe artan, sınıflandırma, algılama ve bölütleme görevlerinde oldukça iyi başarılar elde edilen derin öğrenme algoritmalarının kullanılmasında da oldukça önemli bir yere sahiptir. Örneğin, Nvidia'nın kullanıma sunduğu Jetson TX1 geliştirme kartı ve derin öğrenme algoritmalarıyla, yüksek hesaplama maliyeti gerektiren gerçek zamanlı nesne algılama uygulamalarının gerçekleştirilmesi mümkün olmuştur.

Bu tez çalışmasında, Nvidia Jetson TX1 gömülü sisteminde gerçek zamanlı çalışan silah algılama uygulaması gerçekleştirilmiştir. Bunu gerçekleştirmek için DetectNet ve YOLO nesne algılama mimarileri kullanılmıştır. Hata oranını en aza indirmek ve eğitim süresini düşürmek için her iki modelde de ön eğitilmiş sinir ağı modelleri ile ağırlıklar ilklendirilmiştir. Her iki modelin de eğitimleri ve gömülü sistemde gerçek zamanlı olarak çalıştırılmaları başarılı bir şekilde gerçekleştirilmiştir.

Test görüntüleriyle test edilen modellerden elde edilen sonuçlara göre DetectNet modelinin F1 ölçüm oranı YOLO modelinden daha yüksek çıkmıştır. Ayrıca DetectNet modelinin, Olmos ve arkadaşlarının Faster R-CNN modelini kullanarak gerçekleştirdiği benzer bir çalışmaya göre daha yüksek F1 ölçüm değerine sahip olduğu görülmüştür. Ancak yanlış pozitif sayısını en az bulma noktasında DetectNet modeli, YOLO ve Faster R-CNN modellerinin gerisinde kalmıştır. Yani gömülü sistemde gerçek zamanlı çalışan YOLO modelinin yanlış nesneleri algılama oranının DetectNet modelinden daha düşük olduğu görülmüştür. Jetson TX1 kartında gerçek zamanlı çalışmadaki saniyede işlenen görüntü sayısı ise DetectNet modelinde YOLO modeline göre daha yüksek çıkmıştır.

Bu tez çalışmasında gerçekleştirilen gerçek zamanlı silah algılama uygulamalarından elde edilen sonuçlar ve gömülü sistemde gösterdikleri performans bakımından, bu tarz bir silah algılama uygulamasının güvenlik bakımından gerçek hayatta da kullanılması noktasında umut vericidir. Böyle bir algılama sistemiyle silahlı kişiyi görüntüden gerçek zamanlı tespit etmenin, güvenlik güçlerinin erken müdahalesine olanak tanıyarak can ve mal güvenliğini artıracağı umulmaktadır.

5.2 Öneriler

Silah algılama sürecinden sonra gerçekleşecek işlev uygulamanın kullanıldığı yere ve isteğe göre çeşitlilik gösterebilir.

Sinir ağlarının eğitimi için veri seti miktarını artırmanın ağların hata oranını azaltacağı düşünülmektedir.

Bu çalışmada veri seti oluşturulurken sadece tabanca türünde silahlar kullanılmıştır. Kesici silahlar, ağır silahlar gibi diğer silahların da algılanması için veri seti genişletilebilir. Ayrıca elbise altındaki veya çanta içerisindeki gizli silahları da algılamak için ilave sensörler yardımıyla mevcut sistem daha anlamlı hale getirilebilir.

KAYNAKLAR

Takahashi, D., 2016, Nvidia CEO bets big on deep learning and VR, https://venturebeat.com/2016/04/05/nvidia-ceo-bets-big-on-deep-learning-and-vr/ [Ziyaret Tarihi: 14 Nisan 2017].

Huang, J., 2016, Accelerating AI with GPUs: A New Computing Model, https://blogs.nvidia.com/blog/2016/01/12/accelerating-ai-artificial-intelligence-gpus/, [Ziyaret Tarihi: 21 Nisan 2017].

Beam, A., 2017, Deep Learning 101 - Part 1: History and Background, http://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part2.html, [Ziyaret Tarihi: 24 Nisan 2017].

Koçer, S., Akdağ, A., 2017, Konvolüsyon Sinir Ağı Tabanlı Silah Algılama Uygulaması, *2. Uluslararası Bilgisayar Bilimleri ve Mühendisliği Konferansı (UBMK 2017)*, Antalya, 94-98.

Lowe, D., 1999, Object recognition from local scale-invariant features, *International Conference on Computer Vision (ICCV)*.

Bay, H., Tuytelaars, T.,Van, L., 2006, SURF: Speeded up robust features, *Lecture Notes in Computer Science.*

Lowe, D., 2004, Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision*.

Zhang, H., Hu, Q., 2011, Fast image matching based-on improved SURF algorithm, *International Conference on Electronics, Communications and Control.*

Suard, F., Rakotomamonjy, A., Bensrhair, A., Broggi, A., 2006, Pedestrian detection using infrared images and histograms of oriented gradients, *Intelligent Vehicles Symposium (IVS).*

Jafari, O., Mitzel, D., Leibe, B., 2014, Real-time RGB-D based people detection and tracking for mobile robots and head-worn cameras, *International Conference on Robotics and Automation (ICRA).*

Spinello, L., Arras, K., 2011, People detection in RGB-D data, *International Conference on Intelligent Robots and Systems (IROS)*.

Viola, P., Jones, M., 2001, Rapid object detection using a boosted cascade of simple features, *Computer Vision and Pattern Recognition (CVPR)*.

Dalal, N., Triggs, B., 2005 Histograms of oriented gradients for human detection, *Computer Vision and Pattern Recognition (CVPR)*.

Wang, Z., Jia, Y., Huang, H., Tang, S., 2008, Pedestrian Detection Using Boosted HOG Features, *IEEE International Conference on Intelligent Transportation Systems.*

Laptev, I., 2009, Improving object detection with boosted histograms. *Image and Vision Computing*.

Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D., Object detection with discriminatively trained part-based models, 2010, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*.

LeCun, Y., Boser, B., Denker, D., Henderson, D., Howard, R., Hubbard, W., Jackel, L., 1989, Backpropagation applied to handwritten zip code recognition, *Neural Computation.*

LeCun, Y., ve ark., 1998, Gradient-based learning applied to document recognition, *Proceedings of the IEEE.*

Krizhevsky, A., Sutskever, I., Hinton, G., 2012, Imagenet classification with deep convolutional neural networks, *Advances in neural information processing systems*, 1097–1105.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y., 2013, Overfeat: Integrated recognition, localization and detection using convolutional networks, *arXiv preprint arXiv:1312.6229.*

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014, Rich feature hierarchies for accurate object detection and semantic segmentation, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Girshick, R., Fast R-CNN, 2015, *IEEE International Conference on Computer Vision (ICCV)*, Santiago, 1440-1448.

Ren, S., He, K., Girshick, R., Sun, J., 2015, Faster r-cnn: Towards real-time object detection with region proposal networks, *arXiv preprint arXiv:1506.01497*.

Redmon, J., Divvala, S., Girshick, R. and Farhadi,A., 2016, You Only Look Once: Unified, Real-Time Object Detection, *CVPR.*

Verma, G., Tiwari, R., 2015, A Computer Vision based Framework for Visual Gun Detection Using Harris Interest Point Detector, *Procedia Computer Science. 54.*

Verma, G., 2015, A Computer Vision Based Framework for Visual Gun Detection Using SURF, *10.1109/EESCO.2015.7253863.*

Halima, N.,Hosam, O., 2016, Bag of Words Based Surveillance System Using Support Vector Machines, *International Journal of Security and Its Applications,* Vol. 10, No. 4 (2016), pp.331-346.

Olmos, R., Tabik, S., Herrera1, F., 2017, Automatic Handgun Detection Alarm in Videos Using Deep Learning, *Department of Computer Science and Artificial Intelligence, University of Granada*, Granada.

Özcan, H., 2014, Çok düşük çözünürlüklü yüz imgelerinde derin öğrenme uygulamaları, Yüksek Lisans Tezi, *Deniz Harp Okulu Deniz Bilimleri ve Mühendisliği Enstitüsü,* İstanbul, 24.

Glorot, X., Bordes A. and Bengio, Y., 2011, Deep Sparse Rectifier Neural Networks, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011),* 315.

Maas, A. L., Hannun, A. Y. and Ng, A. Y., 2013, Rectifier nonlinearities improve neural network acoustic models, *Proc. ICML*, 30.

Goodfellow, I., Bengio, Y. and Courville A., 2016, Deep Learning, *MIT Press*, http://www.deeplearningbook.org [Ziyaret Tarihi: 04 Nisan 2017].

Hubel, D.H., Wiesel, T.N., 1962, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, J Physiol ,106-154.

Anonim, UFLDL Tutorial, http://deeplearning.stanford.edu/tutorial/ [Ziyaret Tarihi:14.04.2017].

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, Rob., LeCun, Y., 2013, Integrated recognition, localization and detection using convolutional networks, *arXiv preprint arXiv:1312.6229.*

Krishnan, A., LARSSON, J., 2016, Vehicle Detection and Road Scene Segmentation using Deep Learning, Master's Thesis, *Department of Signals and Systems Signal processing and Biomedical engineering Chalmers University of Technology*, Gothenburg, 18-19.

Çekmez, U., 2014, İnsansız Hava Araçlarında Büyük Ölçekli Yol Planlama Problemlerinin GPU Üzerinde CUDA Yardımı İle Çözümü, Yüksek Lisans Tezi, *Hava Harp Okulu Havacılık ve Uzay Teknolojileri Enstitüsü,* İstanbul, 44,45.

Barker, J., Prasanna, S., Deep Learning for Object Detection with DIGITS, https://devblogs.nvidia.com/parallelforall/deep-learning-object-detection-digits/

Brown, L., 2014, Accelerate Machine Learning with the cuDNN Deep Neural Network Library, https://devblogs.nvidia.com/parallelforall/accelerate-machine-learning-cudnn-deep-neural-network-library/, [Ziyaret Tarihi: 24 Nisan 2017].

Barker, J., 2016, Deep Learning for Object Detection with DIGITS, https://devblogs.nvidia.com/parallelforall/deep-learning-object-detection-digits/, [Ziyaret Tarihi: 27 Nisan 2017].

Tao,A. , J., Barker, J., 2016, Deep Learning for Object Detection with DIGITS, https://devblogs.nvidia.com/parallelforall/detectnet-deep-neural-network-object-detection-digits/, [Ziyaret Tarihi: 27 Nisan 2017].

Tao,A. , J., Barker, J., 2016, DIGITS: Deep Learning GPU Training System, https://devblogs.nvidia.com/parallelforall/digits-deep-learning-gpu-training system, [Ziyaret Tarihi: 29 Nisan 2017].

Yeager, L., DetectNet Network, https://github.com/NVIDIA/caffe/blob/caffe-0.15/examples/kitti/detectnet_network.prototxt, [Ziyaret Tarihi: 20 Nisan 2017].

Shelhamer, E., BVLC GoogleNet Model, http://dl.caffe.berkeleyvision.org /bvlc_googlenet.caffemodel, [Ziyaret Tarihi: 20 Nisan 2017].

Franklin, D., jetson-inference, https://github.com/dusty-nv/jetson-inference, [Ziyaret Tarihi: 20 Nisan 2017].

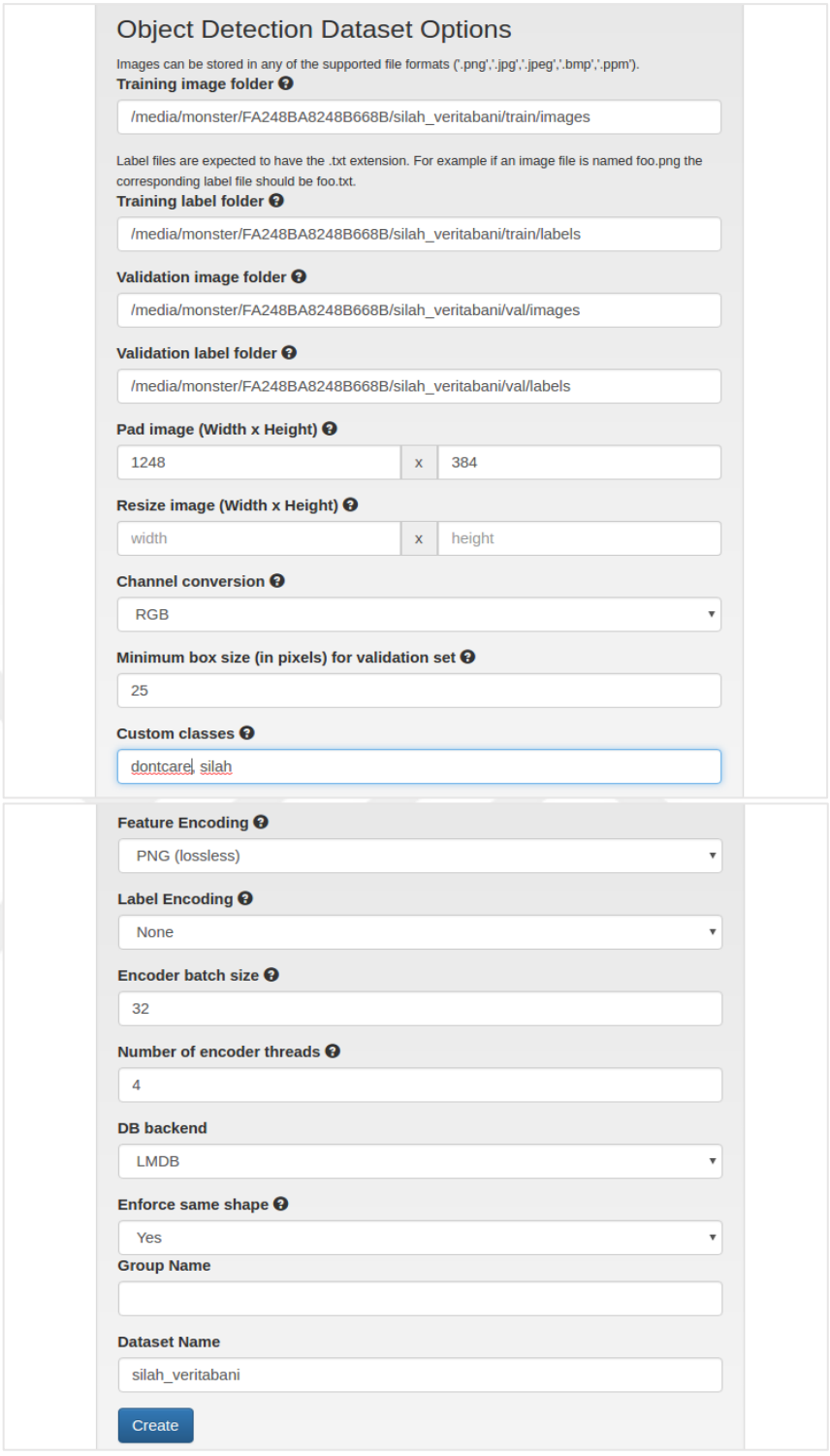
Redmon, J., Darknet, YOLO and pre-trained weights for the convolutional layers, https://pjreddie.com, [Ziyaret Tarihi: 27 Nisan 2017].

Ek-1 Tezde Kullanılan Yazılımların Ekran Görüntüleri

Verilerin etiketlenmesi için yazılan uygulamadan bir ekran görüntüsü Şekil 1'de gösterilmiştir.

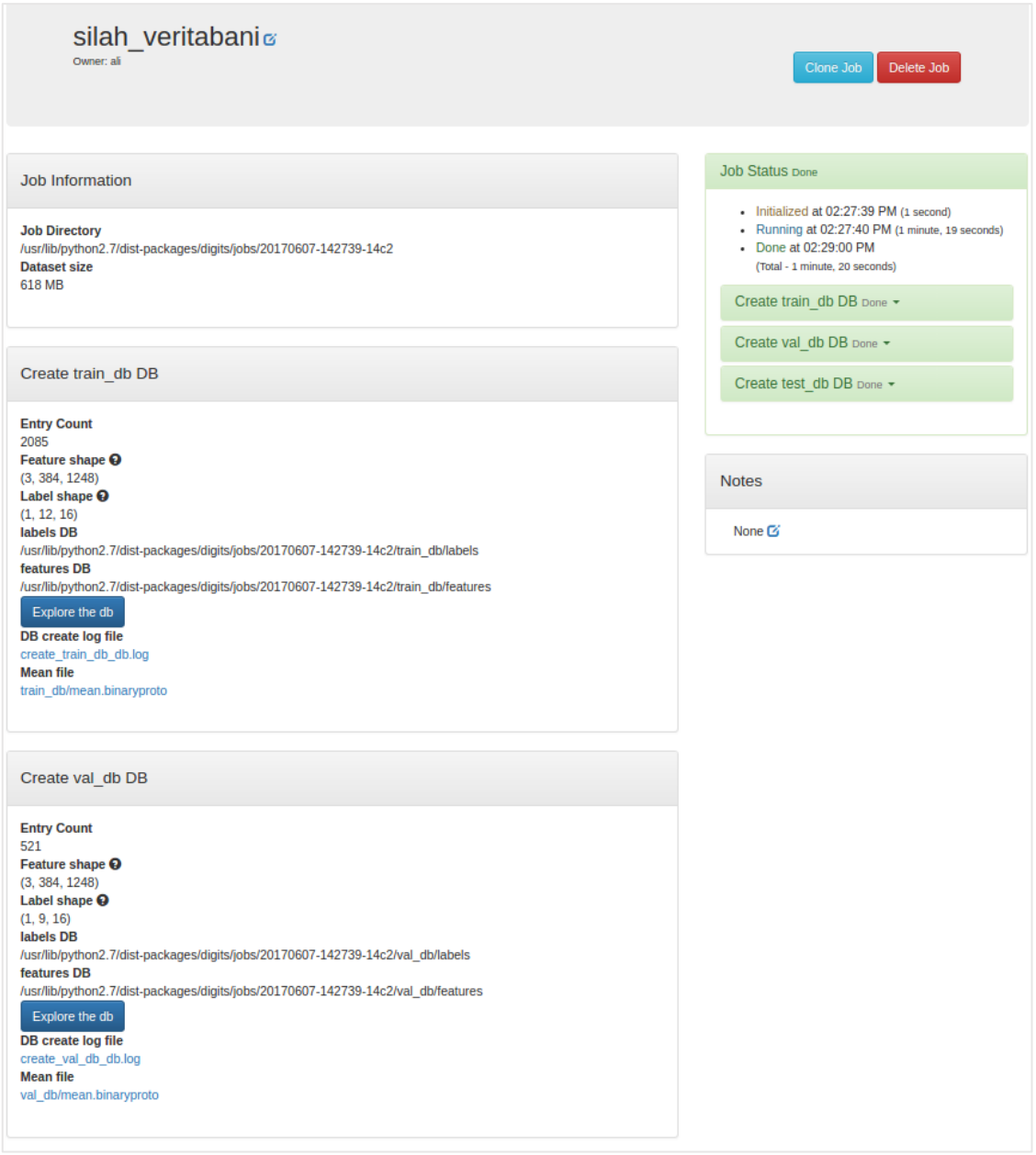
Şekil 1. Etiketleme uygulaması

Veriler etiketlendikten sonra, tez çalışmasında DetectNet modelinde kullanılacak veri setinin hazırlanması ve eğitimi için Nvidia Digits yazılımı kullanılmıştır. Veri seti hazırlama ekran görüntüsü Şekil 2'de gösterilmiştir.



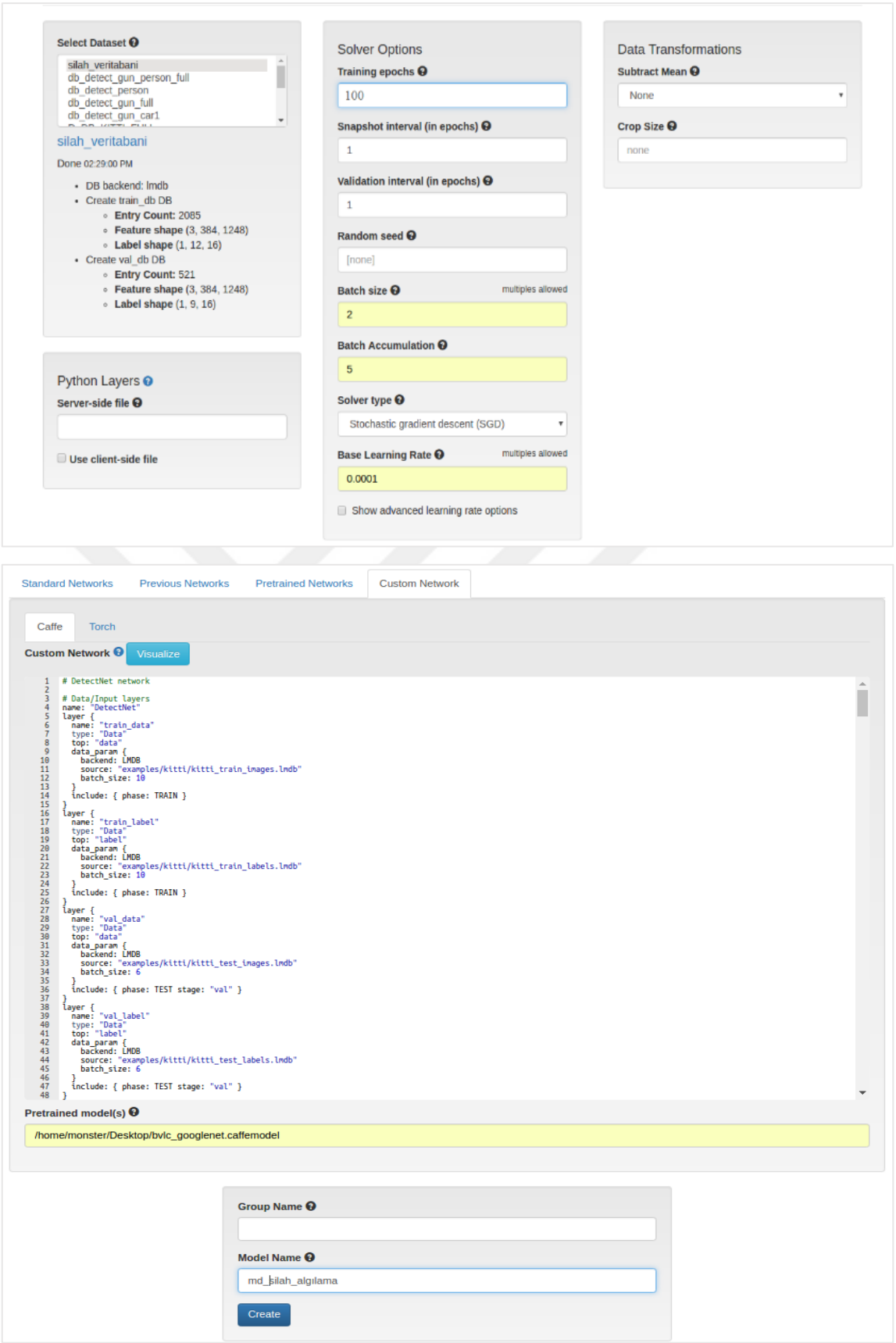
Şekil 2. Nvidia Digits veri seti hazırlama sayfası

Veri seti, model oluşturmak için hazır hale getirildikten sonra açılan, içerisinde veri setiyle alakalı bilgilerin mevcut olduğu ekran görüntüsü Şekil 3'te gösterilmiştir.

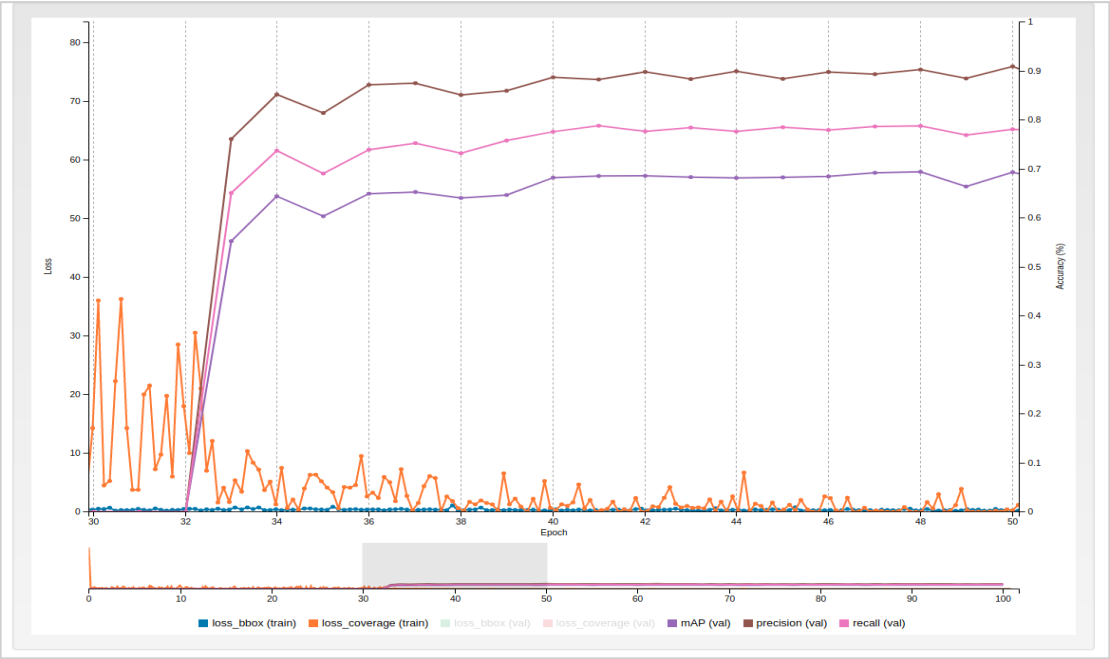


Şekil 3. Nvidia Digits veri seti hazırlandıktan sonra açılan sayfa

Veri setinin hazırlanmasından sonra model oluşturma ayarlarını yaptığımız ekran görüntüsü Şekil 4'te, modelin eğitim grafiği Şekil 5'te, modelin eğitiminin tamamlanmasının ardından açılan sayfa görüntüsü ise Şekil 6'da gösterilmiştir.



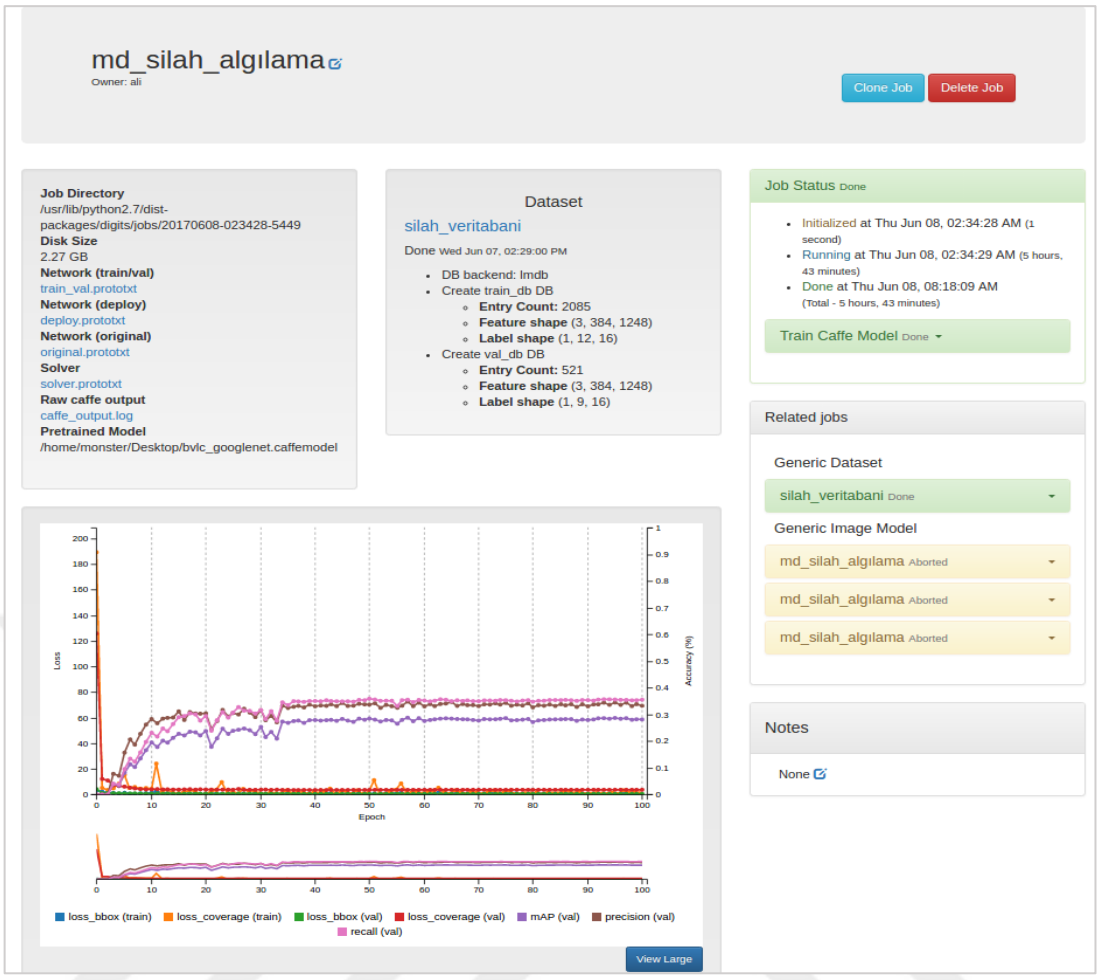
Şekil 4. Nvidia Digits nesne algılama modeli oluşturma sayfa görüntüsü



Şekil 5. Nvidia Digits nesne algılama modeli eğitim grafiği (30. ve 50. devir (epoch) arası)

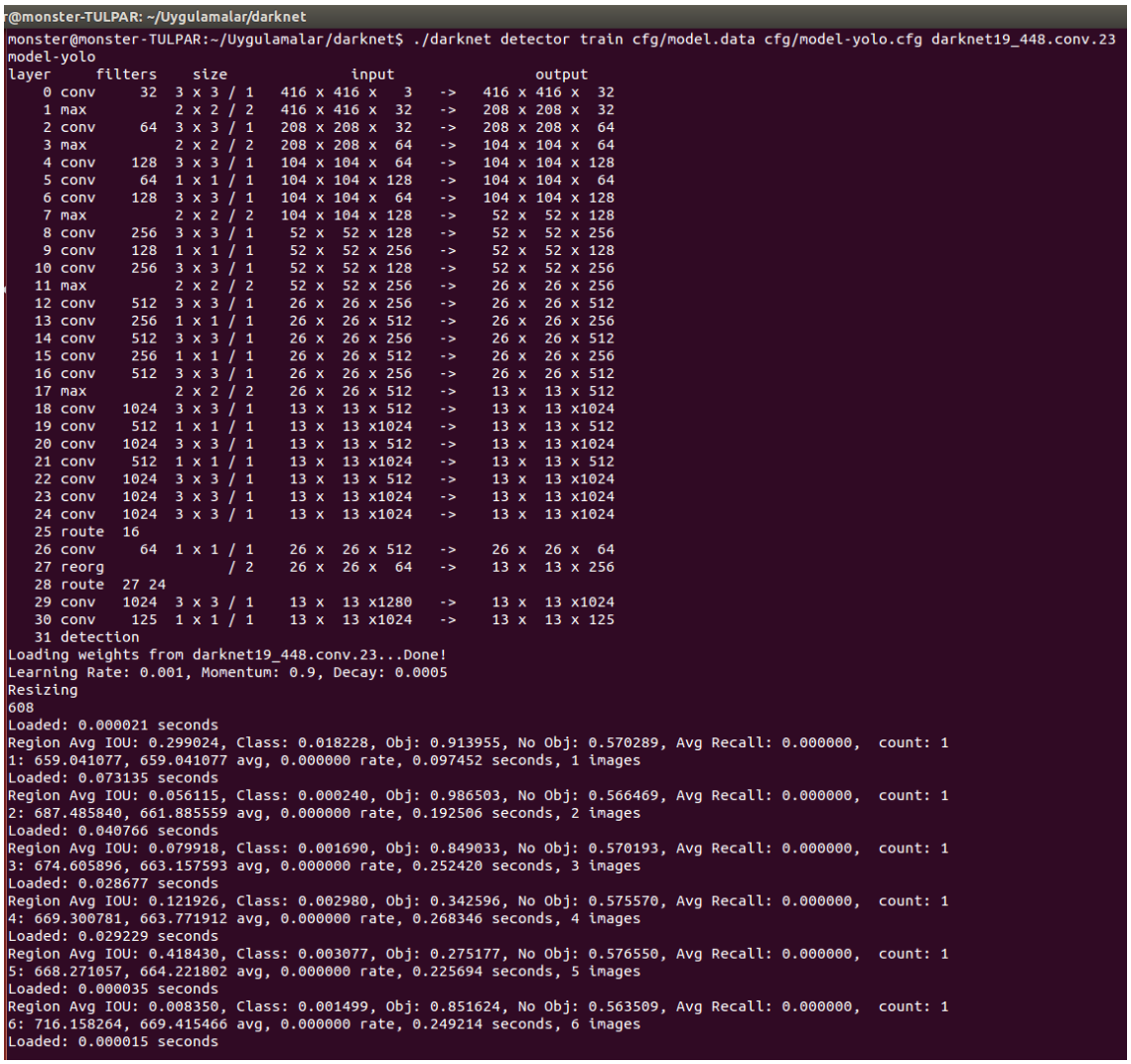
- loss_bbox :Her bir ızgara karesi tarafından kapsanan nesne için sınırlayıcı kutunun gerçek ve tahmin edilen köşelerinin ortalama mutlak farkıdır.
- loss_coverage :Bir eğitim verileri örneğindeki tüm ızgaralı kareler için gerçek ve tahmin edilen nesneler arasındaki farkların karelerinin toplamı olarak tanımlanır.
- Precision (pozitif tahmin değeri): Doğru tespit edilen nesnelerin toplam tahmin edilen nesne sayısına oranıdır.
- Recall (duyarlılık): Doğru tespit edilen nesnelerin görüntülerdeki gerçek nesnelerin toplam sayısına oranıdır.
- mAP: pozitif tahmin değeri (precision) ve duyarlılık (recall) değerleri kullanılarak hesaplanan bir değerdir. Nesne algılama modelinin, ilgilenilen nesneler için ne kadar duyarlı olduğu ve yanlış alarmlardan ne kadar iyi kaçındığının bir kombine ölçüsüdür.

Şekil 5'te DetectNet nesne algılama modeli eğitiminin 30. ve 50. devir (epoch) arasında elde edilen değerlerden oluşan grafik gösterilmiştir. Eğitim bouyunca "loss_bbox" ve "loss_coverage" değerleri minimize edilerek ağın hata kaybı azaltılmıştır. Dolayısıyla pozitif tahmin değeri (precision), duyarlılık değeri (recall) ve bunlara bağlı olarak ölçülen mAP değerinde artış gözlemlenmiştir.

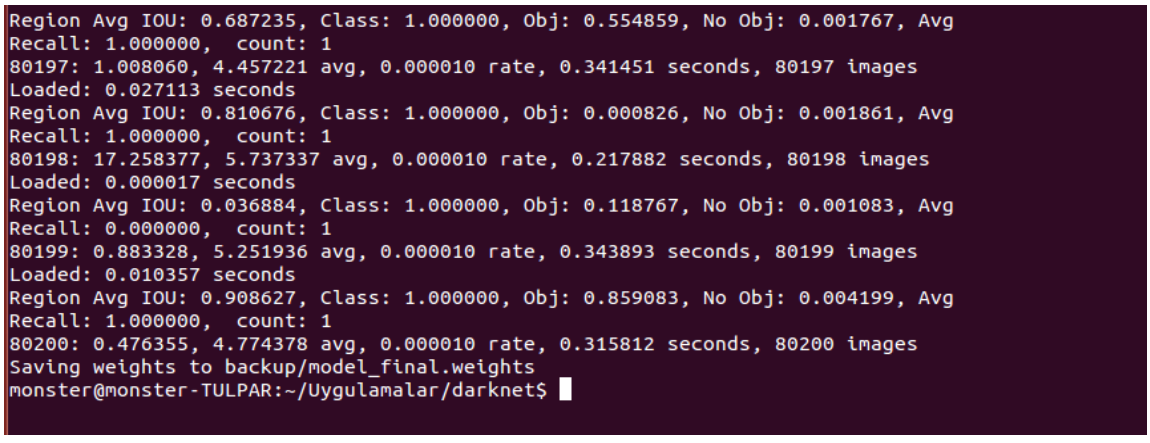


Şekil 6. Nvidia Digits, modelinin eğitimi tamamlandıktan sonraki sayfa görüntüsü

YOLO sinir ağı modelinin eğitilmesi için Ubuntu terminal ekranından başlatılan Darknet yazılımının, başlatıcı komutla birlikte eğitimin başlaması ve sonlanması sırasındaki ekran görüntüleri sırasıyla Şekil 7 ve Şekil 8'de gösterilmiştir.



Şekil 7. YOLO sinir ağı modeli eğitimi başlangıç ekran görüntüsü



Şekil 8. YOLO sinir ağı modeli eğitiminin sonlandığı sıradaki ekran görüntüsü

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	**:**	Ali AKDAĞ
Uyruğu	**:**	T.C.
Doğum Yeri ve Tarihi	**:**	Beyşehir / 15.01.1991
Telefon	**:**	
e-mail	**:**	ali.akdag@ogr.konya.edu.tr

EĞİTİM

Derece		**Adı, İlçe, İl**	**Bitirme Yılı**
Lise	:	**Meram Fen Lisesi, Konya**	**2009**
Üniversite	:	**Mevlana Üniversitesi, Konya**	**2014**
Yüksek Lisans	:	**Necmettin Erbakan Üniversitesi, Konya**	**2014-devam**

İŞ DENEYİMLERİ

Yıl	**Kurum**	**Görevi**
2016	VİTOS ELEKTRONİK YAZILIM MEDİKAL SANAYİ VE TİCARET LİMİTED ŞİRKETİ	Bilgisayar Mühendisi

UZMANLIK ALANI: Bilgisayar Mühendisliği

YABANCI DİLLER: İngilizce

TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

- Koçer, S., Akdağ, A., 2017, Konvolüsyon Sinir Ağı Tabanlı Silah Algılama Uygulaması, *2. Uluslararası Bilgisayar Bilimleri ve Mühendisliği Konferansı (UBMK 2017),* Antalya, 94-98.