Araştırma makalesi

Trafik işaret levhası tespiti için derin öğrenme yöntemi

Mert Çetinkaya[1,*], Tankut Acarman[1]
[1] Computer Engineering Department, Faculty of Technology and Engineering,
Galatasaray University, İstanbul, Turkey
*Correspondence: mcetinkaya@gsu.edu.tr

Özet: Bu çalışmada araç üzerinde bulunan kamera algılayıcısı ile çekilen trafik sahne resimleri üzerinde trafik işaret levhası tespiti için verimli bir yöntem öneriyoruz. Yöntemimiz temelde derin öğrenmeye dayalıdır ve hesaplama işlem süresini kısaltmak için istatistiksel yöntemlerden de yararlanılmaktadır. Yöntemimizi oluşturan algoritmamız 3 adımdan oluşmaktadır. İlk olarak derin öğrenmeye dayalı bir görüntü segmentasyonu yapılmaktadır ve görüntüden bazı bölge önerileri çıkarılmaktadır. Sonra, hipotez testleri yapılarak içerisinde trafik levhası olma olasılığı düşük olan bölge önerilerinden bazıları elenmektedir. Son adımda ise uygun şekilde eğitilmiş olan Convolutional Neural Networks (CNN) sınıflandırıcımız kalan bölge önerileri üzerinde kullanılmaktadır ve sınıflandırıcı tarafından trafik levhası içerdiği güçlü bir şekilde saptanan bölge önerilerinin trafik levhası içerdiği kabul edilmektedir. Burada, ilk adımda verimli ve hızlı çalışan bir görüntü segmentasyonu yaklaşımı kullanılmaktadır. İkinci adımda da hipotez testi gibi basit bir yaklaşım ile hızlıca bazı elemeler yapılmaktadır. Algoritmamızı Alman Trafik İşareti Algılama Karşılaştırması (German Traffic Sign Detection Benchmark (GTSDB)) veri setini kullanarak test ettik ve algoritmamızın kullandığımız metriklere ve hesap işlem süresi maliyetine göre açık kaynak literatürde yayınlanmış başka yöntemlerle kıyaslandığında umut verici olduğunu ve başarılı sonuçlara ulaştığını gördük. Deney sonuçlarımıza göre bu alanda sıkça kullanılan precision ve recall performans metrikleri için %90 - %95 seviyesinde sonuçlara ulaştığımızı ve resim başına işlem süresi konusunda da 0.41 saniye seviyesini sağladığımızı test ettik.

Anahtar Kelimeler: Derin öğrenme, trafik işaret levhası tespiti, görüntü segmentasyonu

A deep learning method for traffic sign detection

Abstract: In this paper, we propose an efficient method for traffic sign detection on traffic scenes taken by a camera placed on car. Our method mostly depends on deep learning and statistical methods are also used to shorten the processing time. Our algorithm in the method consists of 3 steps. Firstly, a deep learning based image segmentation is done and some region proposals are extracted from image. Then, some hypothesis tests are done and some of the region proposals whose probability to include a traffic sign is low are eliminated. At the last step, our nicely trained Convolutional Neural Networks (CNN) classifier is used on remaining region proposals and a region proposal is accepted to contain a traffic sign if it is strongly found in that way by our classifier. Here, an efficient and rapid image segmentation approach is used at the first step and some eliminations are quickly made at the second step using a simple approach like hypothesis test. We tested our algorithm on German Traffic Sign Detection Benchmark (GTSDB) dataset and concluded that our algorithm reaches promising and successful results according to our metrics and processing time compared to other methods published in the open literature. With regards to our experiment results, we saw that we reached 90% - 95% levels according to precision and recall metrics that widely used in this working domain and we reached 0.41 seconds level for processing time per image.

Key words: Deep learning, traffic sign detection, image segmentation

* Corresponding author. Tel.: +90-212-227-4480 ext 1468; fax: +90-212-259-1064
E-mail address: mcetinkaya@gsu.edu.tr
ORCID: 0000-0001-9859-0798 (in hierarchical order)
Received 27 August; accepted 16 October
Peer review under responsibility of Bandirma Onyedi Eylul University.

1. Giriş

Trafik levhası tespit etme ve tanıma ileri sürüş destek sistemlerinde önemli bir teknolojik elemandır (Malik ve Siddiqi, 2014). Bu özelliğin kullanım alanları içerisinde otonom sürüş, ileri sürüş destek sistemleri, trafik levhası haritalarının oluşturulması ve bakımı gibi konular yer alır. Bunun ışıklandırmadan dolayı olan zorlu koşullar, levhanın önünün araç, ağaç vb. başka nesneler ile kapanması ya da perdelenmesi, bakış açısındaki değişimler, hava ya da ışık koşulları, levhadaki eskimeler, levhalarda üretim sırasındaki insan ya da üretim kaynaklı farklılıklar gibi durumlar sebebiyle zorlayıcı bir gerçek dünya bilgisayar görüşü problemi olduğunu söyleyebiliriz. Buradaki problem aslında 2 etaptan oluşur: Trafik Levhası Tespiti (TLT) ve Trafik Levhası Sınıflandırma (TLS). TLT trafikteki görüntüler üzerindeki trafik levhaları için ilgi bölgelerini ve trafik levhasının sınırlarını belirlemek ile uğraşır. İyi bir TLT algoritması görüntü üzerinde mümkün olduğunca fazla trafik levhası bulmalı ve bunu yaparken en az miktarda yanlış tespitte bulunmalıdır. TLT belirlenen bölge için sınıflandırma problemiyle ilgilenmez. Bu problem ile TLS ilgilenir. Başarılı bir TLS algoritması daha önce belirlenen sınıflardan birine ait olarak verilen bir trafik levhası görüntüsünü mümkün olduğunca en az hata ile sınıflandırmalıdır (Berkaya vd, 2016).

Bu çalışmada TLT problemi için akıllı ve verimli bir yaklaşım öneriyoruz. Burada ana odak noktamız TLT problemi ancak algoritmamız TLS sistemi ile trafik işareti sınıflandırma da yapmaktadır. Algoritmamız 3 etaptan oluşmaktadır. Algoritmamız kapsamında ilk adımda derin öğrenme ile semantik görüntü segmentasyonu yaptık. Bu semantik görüntü segmentasyonu için önceden eğitilmiş ve kısa süre içinde çalışan bir sinir ağı kullandık. Bu ilk adımın sonunda bazıları trafik levhası görüntüsü içeren bölge önerileri elde ettik. Sıradaki adımlardaki hedefimiz ise bu bölge önerileri arasından trafik levhası içerenleri seçip trafik levhası içermeyenleri verimli bir şekilde elemek oldu. Burada önem verdiğimiz 2 nokta bulunmaktadır. Algoritmamız trafik levhalarını doğru bir şekilde tespit etmeli ve sınıflandırmalıdır ve özellikle gerçek zamanlı çalışan ileri sürücü destek sistemleri ve otonom sürüş için kullanışlı olması açısından algoritmamız hızlı bir şekilde çalışmalıdır. Hızlı çalışan bir mekanizma elde edebilmek için ikinci adımda basit hipotez testleri kullandık ve ilk adım sonunda bulunan bazı bölge önerilerini hızlı bir şekilde eledik. Deney sonuçlarımıza göre hipotez testi kullanımının sistemin daha hızlı çalışmasını sağlama konusunda yararlı olduğunu gördük ve hızlı çalışma konusunun da ileri sürücü destek sistemleri ve otonom sürüş alanında bu tip bir uygulama için oldukça önemli olduğunu söylemek gerekir. İkinci adım sonunda ilk adım sonunda elde edilen bazı bölge önerilerinin bu şekilde elenmesinin ardından üçüncü adıma geçtik. Üçüncü ve son adımda ise tekrar derin öğrenme kullandık ve derin öğrenme tabanlı sınıflandırıcımıza kalan bölge önerilerini girdi olarak verdik. Sınıflandırıcımızın verilen girdinin herhangi bir sınıfa ait olmasını yüksek olasılıkla bulması durumunda verilen girdiyi trafik levhası içeriyor olarak kabul ettik. Bu sayede bölge önerisinin sınıflandırmasını da yaptık.

Çalışmanın kalan kısmı şu şekilde ilerlemektedir: 2. bölümde kullandığımız veri setini daha detaylı bir şekilde tanıtıyoruz, 3. bölümde TLT ve TLS ile ilgili eskiden yapılmış çalışmaları sunuyoruz, 4. bölümde yöntemimizi tanıtıyoruz, 5. bölümde deney sonuçlarımızı kullandığımız metriklere göre sunup kendi sonuçlarımızı diğer araştırmacıların sonuçları ile kıyaslıyoruz ve 6. bölümde çalışma sonunda elde ettiğimiz çıkarımları sunuyoruz.

2. Veri seti

Modelimizi eğitmek ve test etmek, sonuçlarımızı sayısal olarak değerlendirmek için GTSDB veri setini kullandık ve bazı metriklere göre sonuçları inceledik.

GTSDB veri seti TLT ve TLS sistemlerinin değerlendirilmesi ve testi için yaygın olarak kullanılan erişilebilir bir veri setidir. Veri seti Houben vd. (2013) tarafından hazırlanmıştır. Veri setinde Almanya'da çekilmiş olan 1360x800 piksel çözünürlüğe sahip 900 resim (araç üzeri kameradan çekilen trafik görüntü sahnesi) bulunmaktadır ve resimlerdeki trafik levhalarının koordinat ve sınıf bilgisi algoritmaların eğitiminde kullanmak üzere etiket olarak verilmektedir. Trafik levhaları 43 farklı sınıfa aittir ve 900 resim arasında eğitim ve test veri seti dağılım oranı 600 resim eğitim

ve 300 resim test olacak şekilde %66.6-%33.4 şeklindedir (Temel vd, 2019).

3. Literatür tarama

Daha önce belirttiğimiz gibi problemimiz aslında 2 etaptan oluşuyor: TLT ve TLS. Tekrar hatırlamak gerekirse TLT bir görüntüdeki potansiyel trafik levhalarını tespit etmek iken TLS seçilen bölgelerdeki trafik levhasının sınıfını belirlemek ya da ilgili bölgede trafik levhası olmadığına karar vermek anlamına gelir (Yuan vd, 2015).

Bu konularda yayınlanmış çalışmalara bakarsak, ilk çalışmanın 1984 yılında Japonya'da yapıldığını söyleyebiliriz. Bu çalışmada yol ve trafik levhası tespit ve tanıma işlemi genel olarak tanıtılmaktadır, temel karakteristik özellikler, gereklilikler ve zorluklar sunulmaktadır ve bu kapsamda kullanılabilecek temel görüntü işleme teknikleri anlatılmaktadır (Fleyeh ve Dougherty, 2005). Bu çalışmayı takiben bu konularda başka pek çok çalışma da yapılmıştır (Serna ve Ruichek, 2018). Daha güncel ve yeni araştırmalara baktığımızda görüntü işleme tekniklerinin kullanımının, görüntülerden özellik çıkarımı yapılmasının ve çıkarılan bu özelliklerin Support Vector Machine (SVM), karar ağacı tabanlı yaklaşımlar ve bazı istatistiksel yöntemler ile kullanımının ağırlık kazandığını gözlemlemekteyiz. Ek olarak, Convolutional Neural Networks (CNN) kullanımının da, özellikle son zamanlarda yapılan çalışmalara baktığımızda oldukça yaygın olduğunu söyleyebiliriz. CNN yaklaşımının bu alanda çok başarılı sonuçlar ortaya koyduğunu da eklememiz gerekir. Stallkamp vd. (2012) de yaptıkları çalışmada TLS konusunda CNN yaklaşımının insandan daha başarılı sonuçlar elde ettiğini gösterdiler. Burada, CNN mimarileri arasında çok fazla çeşitliliğin de olduğunu ve TLT problemlerinde de bölge önerisi yapan CNN tabanlı sinir ağlarının kullanılabildiğini eklemek gerekir.

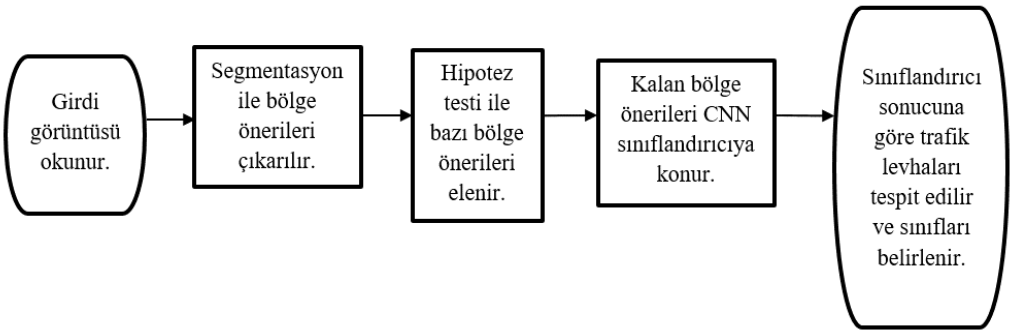
Daha modern yaklaşımlara biraz daha yakından bakarsak Xu vd. (2019) trafik levhalarını tespit etmek için renk segmentasyonu ve şekil simetrisine dayalı bir hipotez testi yaklaşımı kullandılar. Yaptıkları çalışmada renk değerleri üzerinden yaptıkları normalizasyon çalışmasına göre resimden segmente edilen uygun bölgeleri hipotez testine dayalı bir şekil simetri algılama algoritmasına tabi tutarak simetrik bölgeleri tespit etmeye çalıştılar ve bu sayede trafik levhalarını tespit ettiler. Agrawal ve Chaurasiya (2017) çalışmalarında renk segmentasyonu ve Hue Saturation Intensity (HSI) ile eşikleme yaparak trafik levhalarını tespit ettiler. Sonrasında ise geometrik Hu momentlerinden yararlanarak şekil tespiti yaptılar. Devamında ise HSI ve Histogram of Oriented Gradients (HOG) tabanlı bir yaklaşım ile özellik çıkarımı yapıp Principal Component Analysis (PCA) uyguladılar. En sonunda da SVM kullanarak sınıflandırma yaptılar. Yuan vd. (2015) çizge tabanlı bir trafik levhası tespit etme yaklaşımı önerdiler. Kullandıkları çizge tabanlı algoritmada düğümler arasında renk, biçim, şekil ve bağlamsal özellikleri kombine ettiler ve trafik levhalarını tespit ettiler. Sonrasında ise HOG özellikleri ve SVM kullanarak sınıflandırma yaptılar. Ellahyani vd. (2016) 3 adımlı bir trafik levhası tespit ve tanıma yaklaşımı önerdiler. İlk adımda görüntüyü HSI renk alanına göre segmente ettiler. Sonrasında ilk adımda çıkarılan blobları işleyerek trafik levhalarını tespit ettiler. En sonunda da tespit edilen trafik levhalarını Random Forest ile sınıflandırdılar. Arcos-García vd. (2018) çalışmasında derin öğrenme odaklı yaklaşımları kullandılar ve bunlar arasında bir analiz ve karşılaştırma çalışması yaptılar. Faster R-CNN, R-FCN, SSD ve YOLO V2 gibi bölge önerisi için kullanılan sinir ağlarını trafik levhalarını tespit etmek için kullandılar. Sonrasında ise bunları Resnet V1 50, Resnet V1 101, Inception V2, Inception Resnet V2, Mobilenet V1 ve Darknet-19 gibi özellik çıkarıcı/sınıflandırıcı ağlar ile kombine ederek trafik levhası tespit ve tanımada bu yapıları kullandılar. En çok öne çıkan model olarak ise Faster R-CNN Inception Resnet V2 modelini gösterdiler. Torres vd. (2019) çalışmasında, eğitim sırasında, sadece trafikten alınan sahneleri ve taslak olarak kullandıkları yapay olarak ürettikleri trafik levhalarını kullandılar. Çalışmaları kapsamında trafik levhası tespiti için Faster R-CNN nesne tespit edicisinden yararlandılar. Bu şekilde bir veri seti oluşturma ve eğitim yolunu tercih ettiler ve böyle yaparak da nesne tespit edici bir sistem oluşturulabileceğini gösterdiler. Rahman vd. (2019) çalışmasında daha önceden farklı bir veri seti ile eğitilmiş olarak aldıkları Inception V2 sınıflandırıcı ile çalışan Single Shot Multibox nesne tespit edici ağını kullandılar ve onu kendi veri setleri ile tekrar eğittiler. Eğitim

sırasında daha çok gerçekte var olan ancak tespit edilemeyen trafik levhalarına odaklandılar. Gerçekte tespit edilemeyen trafik levhalarını daha başarılı bir şekilde tespit edebilmek için modelin içinde çalışma sırasında üretilen özellik haritalarına odaklandılar ve var olan trafik levhasının kaçırıldığı bazı durumlarda bu özellik haritalarındaki artan hareket ve çeşitliliği kullanarak modellerinin bu konudaki performansını artırmaya çalıştılar. Mathias vd. (2013) entegre kanal özellikleri ve Haar özellikleri kullandılar ve çıkardıkları bu özellikler ile trafik sahnelerindeki birçok aday bölgede, lineer ve ağırlıklı bir şekilde kombine ettikleri karar ağaçlarından sınıflandırıcı olarak yararlandılar. Jung vd. (2016) 6 tipte trafik levhası üzerinde çalıştılar. Çalışmaları sırasında sınıflandırma için LeNet-5 CNN mimarisi kullandılar. Aday bölgeleri belirlerken de renk tabanlı segmentasyon ve Hough dönüşümünden yararlandılar. Li vd. (2016) tespit için modifiye edilmiş bir R-CNN ağını sınıflandırma için de Cuda-convnet sinir ağını kullandılar. Çalışmalarını sadece hız sınırı bilgisini veren trafik işaret levhaları üzerine yaptılar. Abedin vd. (2017) yaptıkları çalışmada bulanık kurallara dayalı renk segmentasyonu ile trafik levhalarını tespit ettiler ve tanıma için de çıkarılan bölge önerileri üzerinde Speeded Up Robust Features (SURF) ve Artificial Neural Networks (ANN) kullandılar. Song vd. (2019) çalışmasında hem tespit hem sınıflandırma için kendi CNN tabanlı modellerini geliştirdiler. Modellerini tasarlarken otonom sürüş için önemli olan bellekte az yer kaplama ve hızlı çalışma gibi konuları da göz önünde bulundurarak az parametreli ve daha küçük bir mimari tercih ettiler.

4. Önerilen yöntem

Daha önce belirttiğimiz gibi önerdiğimiz algoritma 3 kısımdan oluşmaktadır. Algoritma kısımlarını Tablo 1'deki gibi görselleştirebiliriz.

Tablo 1. *Algoritmamızdaki temel kısımlar.*



Bu kısımların görevleri semantik görüntü segmentasyonuna bağlı bölge önerisi çıkarımı, hipotez testi ile bazı bölge önerilerinin elenmesi ve CNN tabanlı sınıflandırılması şeklindedir. Bu tablodaki adımlar bir girdi görüntüsü üzerinden de ayrıca detaylı olarak açıklanacaktır.

4.1. Semantik görüntü segmentasyonu

Semantik görüntü segmentasyonu bir görüntüdeki her bir pikseli bir sınıfa atama ve piksel seviyesinde görüntü sınıflandırma anlamına gelir. Görüntü işleme ve bilgisayar görüşü alanındaki temel uygulamalardan biridir ve tıp, akıllı ulaşım gibi farklı alanlarda kullanılabilir (Liu vd, 2019).

Bu çalışmada semantik görüntü segmentasyonu için Efficient Neural Networks (ENet) mimarisini kullandık. ENet ilk olarak Paszke vd. (2016) tarafından yeni bir derin sinir ağı modeli olarak önerildi. ENet modelinin en önemli avantajlarından birisinin oldukça hızlı olması olarak gösterilebilir. ENet var olan modellerden 18 kata kadar daha hızlıdır, 75 kat daha az FLOP gerektirir, 79 kat daha az parametre içerir ve benzer veya daha iyi sonuçlar verir. Model ile ilgili bir başka güzel nokta ise modelin kendisinin gerçekten

oldukça küçük olmasıdır. Model 3 MB civarı yer kaplar. Bu model, özellikle düşük gecikmeli çalışma gerektiren görevler için oluşturulmuştur (Paszke vd, 2016).

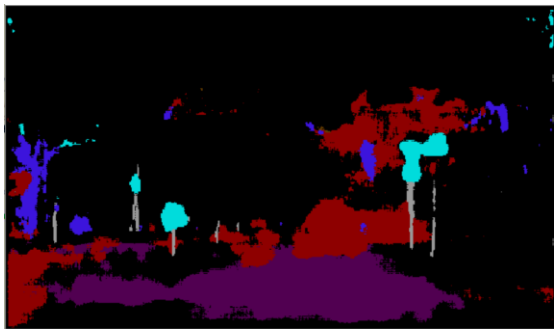
Bu çalışma kapsamında Cityscapes veri setindeki (Cordts vd, 2016) aşağıdaki daraltılmış sınıflar ile önceden eğitilmiş olan hazır bir ENet modeli kullandık.

- Yol
- Direk
- Trafik işareti
- İnsan
- Araba
- Kamyon
- Otobüs
- Etiketlenmemiş

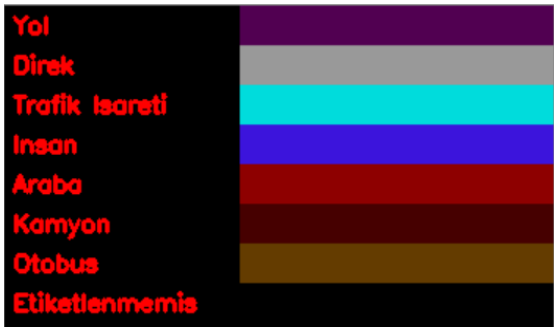
Cityscapes veri setinden bahsedecek olursak, Cordts vd. (2016) tarafından yayınlandığını ve çoğunlukla Almanya'dan ve bazı çevre ülkelerden olmak üzere ilkbahar, yaz ve sonbahar mevsimlerinden birkaç ay boyunca hareket eden bir arabadan alınan birkaç yüz binlik video karesinden oluşan bir veri seti olduğunu söyleyebiliriz. İsminden de anlaşılacağı gibi veri seti kentsel yaşama ait görüntüler içerir ve bu görüntüler nesne tespit algoritmaları için de kullanılabilir. Bu çalışmada da GTSDB veri setini kullanıyor olduğumuzu düşündüğümüzde bu veri seti ile eğitilmiş bir modeli kullanmanın anlamlı olduğunu söyleyebiliriz. Aynı şekilde trafik levhaları konusunda Almanya'ya benzeyen bir başka ülkeye ait veri seti de yine bu model ile kullanılabilir. Trafik levhalarının benzer olduğu, o ülkenin, örneğin Viyana Karayolu İşaretleri ve Sinyalleri Sözleşmesine üye bir ülke olup olmadığına bakılarak anlaşılabilir. Almanya bu sözleşme kapsamındaki bir ülkedir ve bu sözleşme kapsamındaki ülkelerin trafik levhaları birbiriyle çok benzer ya da aynı olacaktır. Cityscapes veri seti oluşturulurken fotoğrafları alınan Almanya ve komşusu ülkeler bu sözleşmeye üye ülkelerdir (Cordts vd, 2016).

ENet tarafından yapılan semantik segmentasyona daha yakından bakacak olursak; Görsel 1'de GTSDB veri setinden örnek bir resim görüyoruz. Görsel 2'de ve Görsel 3'te ise ENet çıktısı olarak Görsel 1'deki resmin segmente edilmiş halini ve ilgili lejantı görüyoruz.

Görsel 1. *GTSDB veri setinden örnek bir resim.*



Görsel 2. *ENet modelimiz tarafından segmente edilmiş resim.*

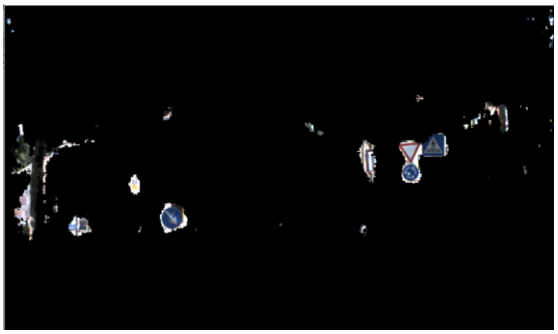


Görsel 3. *ENet modelimiz için segmentasyon lejantı.*

Görsel 1 ve Görsel 2'yi eşit ağırlıklar ile birleştirmemiz durumunda ise Görsel 4'ü elde ediyoruz. Görsel 4 segmentasyon sonuçlarımız açısından daha fazla bilgi vermektedir.

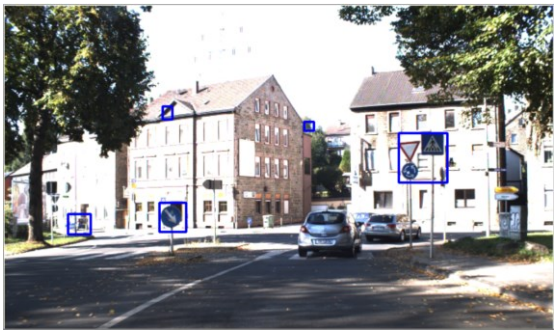
Görsel 4. *Görsel 1 ve Görsel 2'nin eşit ağırlıklar ile kombinasyonu.*

Yöntemimizin 1. adımı kapsamında, segmentasyon operasyonu sonucunda Görsel 1'deki trafik levhası bölgelerini bir maskeleme operasyonu ile çıkardık ve Görsel 5'teki çıktıyı elde ettik. Bu trafik levhaları ENet tarafından yapılan görüntü segmentasyonumuz sonucunda çıkarıldı.



Görsel 5. *Görsel 1'den ENet tarafından trafik levhaları çıkarıldı.*

Bir sonraki etapta ise trafik levhası olarak çıkarılan bölgelerin çevrelerine eğer bu bölgeler kare benzeri ve uygun boyutlarda ise çevreleyici kareler çizdik. Çevrelerine bu şekilde kareler çizilen bölgeler artık onların potansiyel trafik levhası olarak görüldüğü anlamına gelir. Bu işlem sonucunda Görsel 6'daki resim elde edildi.

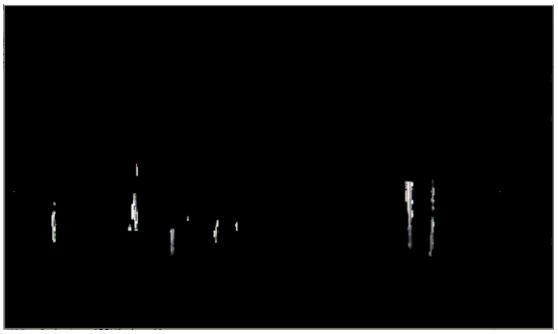


Görsel 6. *Görsel 5'te çıkarılan potansiyel trafik levhaları çevrelerine mavi kareler çizilince elde edilen görüntü.*

Görsel 6'yı elde ettikten sonra Görsel 5'teki konturları da Canny kenar tespit algoritması ve Suzuki ve Abe (1985) tarafından önerilen ve bu alanda yaygınca kullanılan algoritmayı kullanarak bulduk. Burada kontur bulmanın şekil analizi, nesne tespit etme ve tanıma gibi konularda oldukça kullanışlı bir araç olduğunu söylemek gerekir. Bu adımdan sonra bulunan konturların çevresine yeşil kareler çizerek Görsel 7'yi elde ettik.

Görsel 7. *Görsel 5'te çıkarılan görüntülerin konturlarının çevresine yeşil kare çizince Görsel 6'nın devamı olarak elde edilen görüntü.*

Bu adımın sonrasında ENet modelimizi kullanarak görselimizden aynı zamanda direkleri de çıkardık. Görsel 8, Görsel 1'den çıkarılan bu direkleri göstermektedir.



Görsel 8. *Görsel 1'den çıkarılan direkler.*

Görselden bu direkleri çıkardıktan sonra, bu direklerin tepesine kırmızı ve kare şeklindeki çevreleyici kareler çizdik. Çizilen karelerin trafik levhası olabilecek şekilde uygun genişlik ve yüksekliğe de sahip olmasına özen gösterdik. Çizilen bu kırmızı kareleri Görsel 9'da Görsel 7'nin bir devamı olarak görebiliriz.

Görsel 9. *Görsel 8'de bulunan direklerin tepesine kırmızı kareler çizince Görsel 7'nin devamı olarak elde edilen görüntü.*

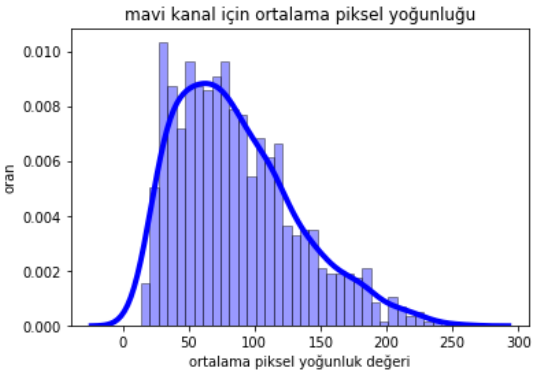
Bu noktada bölge önerilerimizi tanımlamış olduk. Şimdi hipotez testi ile bazılarını elimine edeceğiz.

4.2. Hipotez testi ile bölge önerisi eliminasyonu

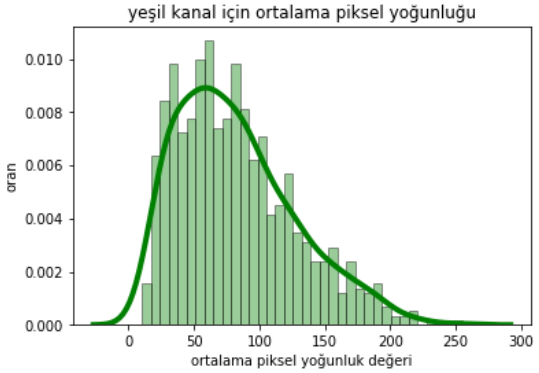
Bu adımda Görsel 9'da önerilen bölgelerden bazıları hipotez testi ile elenecektir. Hipotez testinin güzel yanı ise basit olması ve hızlı çalışmasıdır.

Hipotez testimizi yaparken önerilen bölge içinde ve dışındaki ortalama piksel yoğunluk değerlerini ve modelimiz tarafından segmente edilen yolun merkezi ve ilgili bölge önerisinin merkezi arasındaki mesafeyi kullandık. Diğer bir deyişle, renk bazlı ve konum bazlı olarak hipotez testi yaptık.

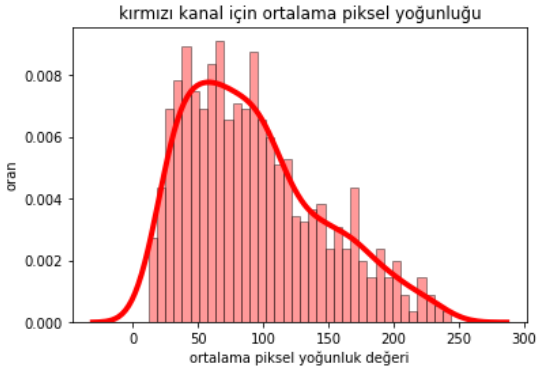
Ortalama piksel değerlerini kullanarak yaptığımız hipotez testi sırasında ilk olarak eğitim setimizdeki trafik levhaları için ortalama piksel değerlerinin yoğunluk grafiklerini her bir kanal için çizdik. GTSDB veri seti için 600 eğitim görüntüsü bulunmaktadır ve bu görüntülerde bizlere sağlanan referans verileri içinde toplam 852 trafik levhası bulunmaktadır. Görsel 10, Görsel 11 ve Görsel 12 her kanal için ilgili yoğunluk grafiğini göstermektedir.



Görsel 10. *Trafik levhalarında mavi kanal için yoğunluk grafiği.*



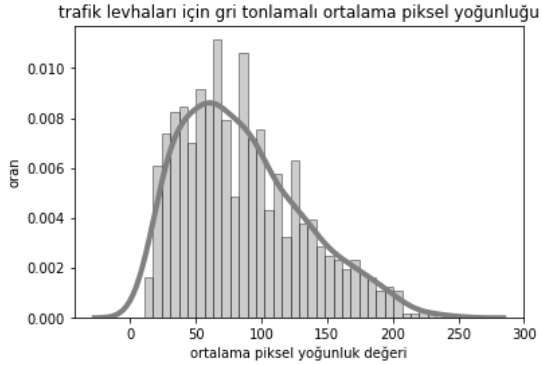
Görsel 11. *Trafik levhalarında yeşil kanal için yoğunluk grafiği.*



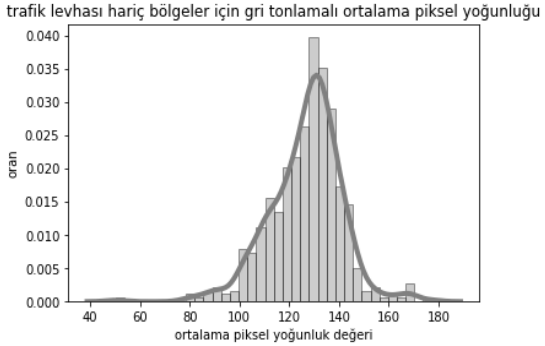
Görsel 12. Trafik levhalarında k*ırmızı kanal için yoğunluk grafiği.*

Bu grafikleri aynı zamanda eğitim resimlerindeki bu trafik levhalarının gri tonlamalı hali için ve eğitim verisinde trafik levhası dışında kalan bölgelerin gri tonlamalı hali için de elde ettik. Çizimleri yaparken her zaman ilgili bölge için olan ortalama piksel yoğunluk değerini kullandık. Yani bir bölge için bir sayı elde ettik. Kırmızı-Yeşil-Mavi (RGB) kanala ait görüntüyü gri tonlamalı hale getirmek için Eşitlik 1'deki formülü kullandık. Gri tonlamaya çevirmeden sonra elde edilen görüntülere ait bahsedilen grafikler Görsel 13 ve Görsel 14'te verilmiştir. Burada Görsel 13'te bulunan eğrinin Görsel 10'da, Görsel 11'de ve Görsel 12'de verilen eğriler ile de tutarlı olduğunu görüyoruz.

$$Gri = 0.299 * R + 0.587 * G + 0.114 * B \quad (1)$$



Görsel 13. *Trafik levhalarında gri tonlama için yoğunluk grafiği.*



Görsel 14. *Trafik levhaları dışındaki bölgeler için gri tonlama için yoğunluk grafiği.*

Birinci hipotezimiz için gri tonlamalı trafik levhası için ve trafik levhası dışındaki alanlar için olan ortalama piksel yoğunluk değerlerini kullandık. Görsel 13 ve Görsel 14'e baktığımızda sola yatık ve sağa yatık normal dağılım eğrileri görüyoruz ve burada gri tonlamalı bu görüntüler üzerinde t-testi tabanlı bir hipotez testi uyguladık. Test sırasında Görsel 13 ve Görsel 14'teki eğrilerin ortalama ve standart sapma değerlerini kullandık. Burada Ghimire ve Wang (2012) tarafından getirilen ve Liao ve Akritas (2007) tarafından önerilen yaklaşımın geliştirilmesiyle elde edilen yaklaşıma benzer bir yöntem kullandık. Sun ve Ho (2011) tarafından yapılan çalışmadakine benzer şekilde de ortalama piksel yoğunluk değerlerini kullandık.

Bu hipotez testi için çalışmamızı aslında bir bölge önerisini trafik levhası veya değil şeklinde sınıflandırmayı amaçlayan bir ikili sınıflandırma problemi gibi düşündük. Bir örnek üzerinden düşünecek olursak Görsel 13 ve Görsel 14'te çizimi verilmiş olan veriyi $(x_{11}, x_{12}, x_{13}, \dots, x_{1m})$ ve $(x_{21}, x_{22}, x_{23}, \dots, x_{2n})$ olarak düşünelim. Aynı zamanda elimizde gri tonlamalı halinin ortalama piksel yoğunluk değeri $\bar{x}$ olarak tanımlanmış olan ve hipotez testi ile trafik levhası olup olmadığına karar vermek istediğimiz bir bölge önerisi olsun. Burada kullandığımız hipotez testi aşağıdaki 2 teste dayalı bir yaklaşımdır. H0 hipotez değerimiz ise 2 örneklemin aynı ortalamaya sahip olduğu, daha genel bir deyişle, 2 dağılımın özdeş olduğudur (Liao ve Akritas, 2007).

- 1. Test: $\bar{x}$, 1. sınıftan gelen gözlemler ile birlikte düşünülür ve $(\bar{x}, x_{11}, x_{12}, x_{13}, \dots, x_{1m})$ ve $(x_{21}, x_{22}, x_{23}, \dots, x_{2n})$ H0 hipotezini test etmek için kullanılır.
- 2. Test: $\bar{x}$, 2. sınıftan gelen gözlemler ile birlikte düşünülür ve $(x_{11}, x_{12}, x_{13}, \dots, x_{1m})$ ve $(\bar{x}, x_{21}, x_{22}, x_{23}, \dots, x_{2n})$ H0 hipotezini test etmek için kullanılır.

Bu 2 test sırasında t değeri Eşitlik 2'deki formül kullanılarak elde edilir.

$$t = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}} \qquad (2)$$

Eşitlik 2'de $\overline{x_1}$ ve $\overline{x_2}$ örneklem ortalamasını s_1^2 ve s_2^2 örneklem variyansını N_1 ve N_2 ise örneklem boyutunu gösterir.

İki test için t değerlerinin bulunmasının ardından, ilgili p değerlerini de hesapladık. Bu değerleri $PV_1(\bar{x})$ ve $PV_2(\bar{x})$ ile gösterelim ve bir bölge için 1. sınıftan veya 2. sınıftan olma olasılığının ilk değerlerini de p_1 ve p_2 ile gösterelim.

Liao ve Akritas (2007) tarafından önerilen yaklaşıma göre eğer $\bar{x}$ 1. sınıftan geldiyse onu 2. sınıftan alınan örneklem ile bir araya getirmek 2 sınıf arasındaki farkı azaltacaktır. 1. sınıftan alınan örneklem ile bir araya getirmek ise bu farkın azalmasına engel olup farkı en azından koruyacaktır. Burada, $PV_1(\bar{x}) / (PV_1(\bar{x}) + PV_2(\bar{x}))$ oranı, test tabanlı olarak, $\bar{x}$ değeri için 1. sınıftan olmama olasılığıdır. Aynı şekilde $PV_2(\bar{x}) / (PV_1(\bar{x}) + PV_2(\bar{x}))$ değeri ise $\bar{x}$ için 1. sınıftan olma olasılığıdır. Getirdikleri yaklaşım $\bar{x}$ örneğini eğer $(PV_2(\bar{x}) / PV_1(\bar{x})) > (p2 / p1)$ ifadesi sağlanırsa 1. sınıf olarak sınıflandırır ve eğer $(PV_1(\bar{x}) / PV_2(\bar{x})) > (p1 / p2)$ ifadesi sağlanırsa 2. sınıf olarak sınıflandırır.

Ghimire ve Wang (2012) çalışmalarında bu yaklaşımı inceledi ve p değerlerinin (PV_1 ve PV_2) çok küçük olması durumunda bu yöntemin yanlış sınıflandırma yapabileceğini gösterdi ve bu yönteme aşağıdaki gibi bir iyileştirme önerdi:

- Eğer $\max(PV_1, PV_2) \geq \varepsilon$ ise bu durumda eğer $(PV_2(\bar{x}) / PV_1(\bar{x})) > (p2 / p1)$ ise $\bar{x}$ 1. sınıf olarak sınıflandırılır.
- Eğer $\max(PV_1, PV_2) < \varepsilon$ ise bu durumda eğer $\bar{x}$ ve 1. sınıf arasındaki mesafe $\bar{x}$ ve 2. sınıf arasındaki mesafeden küçük ise $\bar{x}$ 1. sınıf olarak sınıflandırılır.

Biz de kendi çalışmamızda Ghimire ve Wang (2012) tarafından getirilene benzer bir yaklaşım kullandık. Yaklaşımımız aşağıdaki gibidir. Burada trafik levhası 1. sınıfı, trafik levhası olmaması durumu ise 2. sınıfı temsil etmektedir.

- Eğer $(PV_2(\bar{x}) / PV_1(\bar{x})) > (p2/p1)$ ise, $\bar{x}$ 1. sınıf olarak değerlendirilir.
- Eğer $(PV_1(\bar{x}) / PV_2(\bar{x})) > (p1/p2)$ ise ve $\bar{x}$ ve 1. sınıf arasındaki mesafe $\bar{x}$ ve 2. sınıf arasındaki mesafeden küçükse $\bar{x}$ 1. sınıf olarak sınıflandırılır.
- Eğer $(PV_1(\bar{x}) / PV_2(\bar{x})) > (p1/p2)$ ise ve $\bar{x}$ ve 1. sınıf arasındaki mesafe $\bar{x}$ ve 2. sınıf arasındaki mesafeden büyükse, $\bar{x}$ 2. sınıftan sınıflandırılır.

Görüleceği gibi bir bölge önerisini elerken yani bir bölge önerisi için 2. sınıftan (trafik levhası olmayan sınıftan) derken tedbirli davranıyoruz ve gerçek bir trafik levhasını elemekten kaçınıyoruz. Çünkü $(PV_2(\bar{x}) / PV_1(\bar{x}))$ ifadesinin $(p2/p1)$ ifadesinden büyük olması durumunda $\bar{x}$ doğrudan 1. sınıftan kabul edilir. Tersi durumda ise, ancak ve ancak, $\bar{x}$ ve 1. sınıf arasındaki mesafe $\bar{x}$ ve 2. sınıf arasındaki mesafeden büyükse, $\bar{x}$ 2. sınıftan kabul edilir. Bu mesafe koşulunun sağlanmaması durumunda ise $\bar{x}$ yine 1. sınıftan kabul edilir.

Burada p1 ve p2 olasılık değerlerini 0.2 ve 0.8 olarak aldık. p1 değerini bir bölge önerisinin trafik levhası olma olasılığı olarak düşündük ve p2 değerini de bir bölge önerisinin trafik levhası olmama olasılığı olarak düşündük. Bu değerleri 1. adım sonunda eğitim veri seti üzerinde elde edilen bölge önerilerine göre hesapladık. Bölge önerilerinin ne kadarının trafik levhası olduğuna ve ne kadarının trafik levhası olmadığına baktık ve 1. adım sonunda yapılan bölge önerileri arasında trafik levhası olan ve olmayan önerilerin sayısının toplam bölge önerisi sayısına göre oranına baktık. Bu oranların 1. adım sonunda bulunan bir bölge önerisinin trafik levhası olup olmamasının olasılığı olduğunu kabul ettik. Bu yaklaşımı belirlerken referans olarak kullanılan Liao ve Akritas (2007) ve Ghimire ve Wang (2012) tarafından yapılan çalışmalarda da p1 ve p2 olasılık değerlerinin belirlenmesi sırasında bu ve buna benzer yaklaşımlar kullanılmıştır. Biz de eğitim görüntüleri üzerinde 1. adım sonundaki deneylerimiz sonunda bu oranları yaklaşık olarak 0.2 ve 0.8 şeklinde bulduk ve bu oranları kullandık. Yani buradan 1. adım sonunda bulunan bölge önerilerinden yaklaşık %20'sinin gerçekten trafik levhasını gösterirken %80'inin ise gerçek bir trafik levhasını göstermiyor olduğunu anlayabiliriz. Çalışmamız kapsamında hipotez testi kullanıyor olmamızın amacı ilk adım sonunda bulunan bazı bölge önerilerini hızlı ve en az hata ile elimine etmektir. 3. adımda kullanılan CNN tabanlı sınıflandırıcı ile de bu işlem, üstelik daha başarılı bir şekilde, yapılmaktadır. Ancak CNN tabanlı sınıflandırıcı daha karmaşık yapısı sebebiyle hipotez testine göre daha yavaş çalışmaktadır. Hipotez testinin daha basit bir yöntem olması hız kazanmamızı sağlarken doğruluk açısından ise performans düşüşüne sebep olmaktadır. Bu yüzden hipotez testinin bize kazandırdığı hız ve karşılığındaki doğruluk açısından performans düşüşü arasındaki ödünleşim uygun bir seviyede olmalıdır. Bu yaklaşım belirlenirken kullanılan referans yayınlarında önerilen yaklaşımlardan birine göre seçilmesine ek olarak p1 ve p2 için 0.2 ve 0.8 değerlerinin seçilmesinin bu ödünleşim konusunda da uygun sonuçlar verdiği tespit edilmiştir. 2 farklı aşamadan oluşan bu hipotez testlerinin, yani 2. adımın, kullanıldığı ve kullanılmadığı durumlarda zaman ve doğruluk ile ilgili elde edilen sonuçlar bu çalışmanın Sonuçlar başlıklı 5. bölümünde de verilmektedir. Farklı p1 ve p2 değerleri için zaman ve doğruluk arasında farklı sonuçlar elde edilebilir. Örneğin p1 değerinin 1'e yaklaştırılması hipotez testinin bu aşamasındaki eliminasyon sayısının giderek azalması yani bu aşamanın giderek etkisini

kaybetmesi anlamına gelir. Bu da çalışma hızını azaltırken doğruluk açısından performans artışı sağlar. Çünkü 3. adımda kullanılan CNN tabanlı sınıflandırıcı bölge önerileri üzerinde daha doğru şekilde sınıflandırma yapar ancak karmaşık yapısı ve daha fazla bölge önerisi yapılması sebebiyle daha fazla sayıda çalışmak zorunda kalması işlem süresini uzatır. Burada, mesafe bilgisi hesaplanırken de sınıflara ait olan ve gri tonlama için oluşturulan yoğunluk grafiklerinde kullanılan değerlerin ortalaması ve $\bar{x}$ ve bu ortalama değerler arasındaki mutlak fark kullanılmıştır.

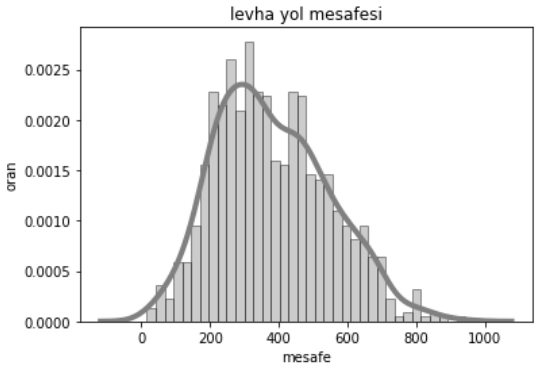
Bu mantığı kullanarak ve gri tonlamalı trafik levhası örneklerin ve gri tonlamalı trafik levhası olmayan örneklerin ortalama ve standart sapma değerlerinden yola çıkarak birinci hipotez testimizi H0: Bölge önerisi trafik levhası popülasyonundan olabilir ve H1: Bölge önerisi trafik levhası popülasyonundan olamaz şeklinde kurduk. Eğer bir bölge önerisi için H0 hipotezi reddedilirse, o bölge önerisini kesin olarak eledik.

İkinci olarak, bir de konum bazlı bir hipotez testi yaptık. Burada da segmente edilen yolun merkezi ve bulunan bölge önerisinin merkezi arasındaki mesafe bilgisine göre test yaptık. Örneğin Görsel 15'te, Görsel 1'deki yolun segmente edilmiş halini görüyoruz. Görsel 16 ise orijinal resim üzerinde bu segmente edilen yol çevresine çizilmiş çevreleyici dikdörtgeni gösteriyor.

Görsel 15. *Görsel 1'deki yolun segmentasyonu.*

Görsel 16. *Segmente edilmiş yol etrafına çizilen çevreleyici dikdörtgen.*

Görsel 17'de, eğitim verisinde, piksel tabanlı mesafe değerleri için segmente edilen yolun merkezi ve trafik levhalarının merkezi arasındaki uzaklığa ait yoğunluk grafiği verilmektedir.



Görsel 17. *Segmente edilen yolun merkezi ve trafik levhalarının merkezi arasındaki mesafe için yoğunluk grafiği.*

Burada yine bir normal dağılım eğrisi çıkmaktadır. Bu kez merkezi limit teoremine dayalı ve daha basit bir hipotez testi yaptık. Bir önceki bölge önerisi elemesi sırasındaki hipotez testindeki H0 ve H1 hipotezlerini kullandık. Burada da H0 hipotezinin elenmesi durumunda o bölge önerisini tamamen eledik.

Daha önce belirttiğimiz gibi buradaki hipotez testimiz merkezi limit teoremine bağlı olan ve 3. eşitlikteki gibi formüle edilebilecek bir testtir.

$$Z_h = \frac{\bar{x} - \mu}{\frac{\sigma}{\sqrt{n}}} \qquad (3)$$

Burada Z_h test değeri olarak kullanıldı ve $p = 0.01$ için çift kuyruklu bir hipotez testi yapıldı. $0.005 < h < 0.995$ eşitsizliğinin geçerli olmaması durumunda, H0, bölge önerisi trafik levhası popülasyonundan olabilir, hipotezini reddettik. Bu tip bir hipotez testinde standart

olarak $p = 0.05$ kullanılır. Bununla birlikte hipotez testinin ilk adımındaki gibi tedbirli davranmak ve gerçek bir trafik levhasını elemekten kaçınmak için bu değeri 0.01 olarak seçtik. Hipotez testinin ilk adımındaki gibi burada da p değerinin bu şekilde seçilmesinin çalışma zamanı ve doğruluk arasındaki ödünleşim açısından uygun şekilde sonuç verdiğini gördük. Hipotez testinin ilk adımında belirtildiği gibi hipotez testlerinin hiç kullanılmadığı ve kullanıldığı durumlarda çalışma zamanı ve doğruluk açısından elde edilen sonuçlar bu çalışmanın Sonuçlar başlıklı 5. bölümünde verilmiştir. Yine aynı şekilde farklı p değerleri için zaman ve doğruluk arasında farklı sonuçlar elde edilebilir. Hipotez testinin 1. aşamasındaki gibi; Örneğin p değerinin daha da küçültülmesi hipotez testinin bu aşamasındaki eliminasyon sayısının giderek azalması yani bu aşamanın giderek etkisini kaybetmesi anlamına gelir. Bu da çalışma hızını azaltırken doğruluk açısından performans artışı sağlar. Bunun sebebi ise yine aynı şekilde 3. adımda kullanılan CNN tabanlı sınıflandırıcının daha doğru şekilde sınıflandırma yapması ancak karmaşık yapısı ve daha fazla bölge önerisi yapılması sebebiyle daha fazla sayıda çalışmak zorunda kalmasının işlem süresini uzatmasıdır.

Görsel 18'de, 1. etap sonunda bulunan bölge önerilerini görüyoruz. Görsel 19'da gri tonlamada ortalama piksel yoğunluk değerlerini kullanarak uyguladığımız hipotez testinin ardından kalan bölge önerilerini görüyoruz. Görsel 20'de ise trafik levhası ve segmente edilen yolun merkezi arasındaki mesafeye dayalı hipotez testinin ardından kalan bölge önerilerini görüyoruz.

Görsel 18. *Birinci etap sonunda bulunan bölge önerileri.*

Görsel 19. *İkinci etapta birinci hipotez testi sonunda kalan bölge önerileri.*

Görsel 20. *İkinci etapta ikinci hipotez testi sonunda, yani ikinci etap sonunda, kalan bölge önerileri.*

Görsel 18, Görsel 19 ve Görsel 20'ye bakarak her hipotez testi sonunda bazı bölge önerilerinin elendiğini görüyoruz. Bu bölge önerilerini zaman anlamında verimli olarak hızlı çalışan ve basit bir mantığa sahip olan istatistiksel hipotez testleri ile eliyoruz.

Bu noktada, ilk adım sonunda bulunan bazı bölge önerilerini eledikten sonra, iyi bir şekilde eğitilmiş olan sınıflandırıcımızı kullanarak kalan bölge önerileri hakkında karar vereceğiz.

4.3. Bölge önerileri üzerinde cnn tabanlı sınıflandırma

Bu adımda 2. adım sonunda kalan bölge önerilerini CNN tabanlı sınıflandırıcımıza verdik ve sonuçları inceledik.

Kurduğumuz CNN tabanlı sınıflandırıcının Wong vd. (2018) tarafından önerilen mimariye de benzer olduğunu söyleyebiliriz. Wong vd. (2018) çalışmasında GTSDB probleminin bir öncülü olarak kabul edilebilecek ve Stallkamp vd. (2011) tarafından çıkarılmış olan German Traffic Sign Recocognition Benchmark (GTSRB) veri seti ve problemi üzerine çalıştı. GTSRB veri seti sadece trafik levhaları görüntüleri içerir ve bu problem sadece levhayı doğru bir şekilde sınıflandırmayı amaçlar. Wong vd. (2018) yaptıkları çalışmada GTSRB

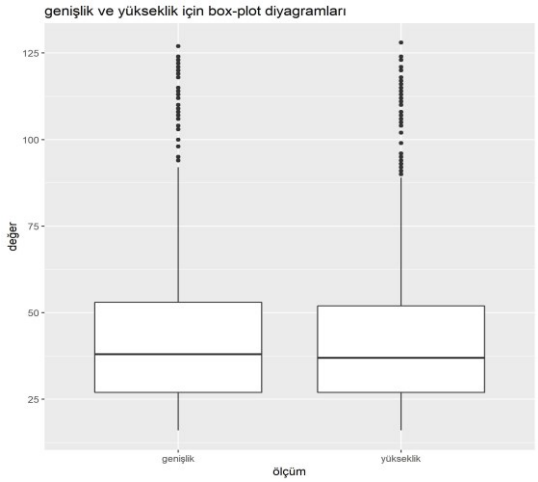
problemi için başarılı bir mimari ürettiler ve biz de yöntemimizin bu adımında benzer bir sınıflandırma problemi ile uğraşıyor olduğumuz için Wong vd. (2018) tarafından önerilen mimariden esinlendik ve sınıflandırmamızı yapmak için ona benzer mimariler ile denemeler yaptık ve başarılı bir sınıflandırıcı ürettik. Denemelerimiz sırasında farklı sayı ve boyutlarda filtreler içeren konvolüsyonel katmanlar ve farklı sayılarda nöronlar bulunduran tam bağlantılı katmanlar kullandık. Katmanlardaki filtre sayıları ve nöronlar için 32, 64, 128 ve 256 gibi sayılar kullandık. Filtre boyutları için ise 3x3 ve 5x5 boyutlarını kullandık. Konvolüsyonel katmanlarda adım boyunu 1 olarak belirledik. Pooling katmanlarında ise 2x2 boyutlu filtreler ve 2 birimlik adım boyutu kullandık. Sınıflandırma aşamasında Softmax sınıflandırıcı diğer katmanlarda ise Rectified Linear Unit (ReLU) isimli aktivasyon fonksiyonunu kullandık. ReLu aktivasyon fonksiyonu için formül 4. eşitlikte verilmiştir.

$$r(x) = \max(0, x) \tag{4}$$

Softmax sınıflandırıcı için ise formül 5. eşitlikte verilmiştir. Burada x_i, x girdisi için ilgili sınıfa ait olarak bulunan CNN skor değeridir.

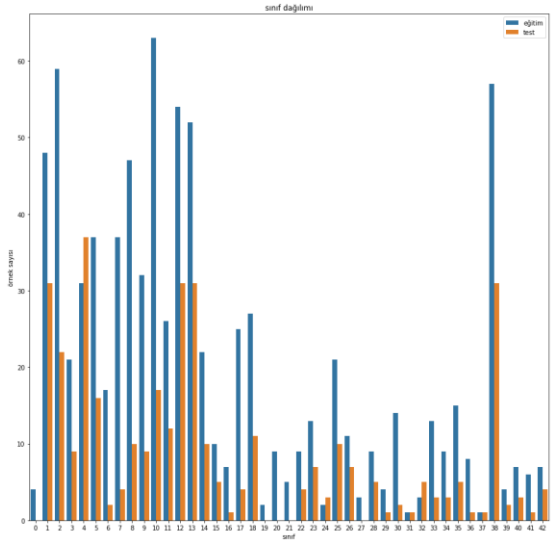
$$s(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{5}$$

Bölge önerileri sınıflandırıcıya konurken 40x40 boyutlarına getirildi. Bu yaklaşımın eğitim verisindeki trafik levhalarının boyutlarına baktığımızda da mantıklı olduğunu söyleyebiliriz. GTSDB eğitim verisindeki trafik levhalarının ortalama piksel genişlik ve yükseklik uzunluklarının sırasıyla 43.4 ve 42.75 piksel olduğunu söyleyebiliriz. İlgili box-plot diyagramı Görsel 21'de verilmektedir.



Görsel 21. *GTSDB eğitim verisindeki trafik levhalarına ait genişlik ve yükseklik değerleri.*

Biz de modelimizi GTSDB veri setinde eğitim resimlerindeki trafik levhalarını kullanarak eğittik ve validasyonunu gerçekleştirdik. Eğitim ve validasyon için k katlamalı çapraz doğrulama yaklaşımını kullandık. k değerini 5 olarak aldık ve bu yaklaşım sonucu en doğru model mimarisini belirledik. GTSDB veri setindeki eğitim ve test resimlerindeki trafik levhalarının sınıflara göre dağılımı Görsel 22'de verilmiştir. k katlamalı çapraz doğrulama sadece eğitim verisi üzerinden yapılmış, bu şekilde en doğru model bulunmuş en sonunda ise bu model eğitim verisinin tamamı ile eğitilip test verisi ile test edilmiştir.



Görsel 22. *GTSDB veri setinden elde edilen veri seti.*

Denenen farklı mimarilerin her birinde 25 adımlık (epoch) eğitim kullandık ve eğitimler sırasında Adam optimizasyon algoritması ve categorical cross entropy hata fonksiyonunu tercih ettik. Her bir epoch içindeki iterasyonlarda ise kullanılan batch boyutunu 32 olarak tercih ettik. Categorical cross entropy hata fonksiyonunun softmax sınıflandırıcı için olan formülü 6. eşitlikte verilmiştir.

$$L(x_i) = -\log(\frac{e^{x_i}}{\sum_j e^{x_j}}) \quad (6)$$

İlk ağırlıkların atanması sırasındaki rassallıklar, optimizasyon algoritmasının stokastik yapısı, iterasyonlar sırasında rassal batch seçimi ve yapay sinir ağı eğitimi sırasında değeri belirlenecek çok fazla sayıda parametrenin olması gibi sebeplerden dolayı aynı mimari ve aynı eğitim prensibi kullanılsa bile yapay sinir ağlarında her eğitim sonucunda farklı sonuçlar veren farklı modellerin ortaya çıkması oldukça yaygın bir durumdur. Bununla birlikte çıkan modeller ve sonuçlar genelde birbirine yakın olur. Bu duruma karşın k katlamalı çapraz doğrulama sırasında her bir katlama için en baştan 3 defa eğitim gerçekleştirdik ve validasyon sonucu aldık. k değeri de 5 olarak alındığı için her bir mimariyi en baştan 15 kere eğittik ve elde edilen 15 validasyon değerinin ortalamasını alıp en yüksek ortalamayı veren mimariyi sınıflandırıcı mimarisi olarak belirledik. Kullandığımız mimariler tarafından bulduğumuz validasyon ortalamalarının yaklaşık olarak [0.94-0.97] aralığında olduğunu gördük. En iyi ortalamayı veren ve bizim sınıflandırıcımızda kullanmaya karar verdiğimiz mimari ise Tablo 2'de belirtildiği gibi bulundu. Bu tabloda gösterilmemiş olsa da kullandığımız mimaride pooling katmanları ve tam bağlantılı katmanlar ardından seyreltme (dropout) yöntemi de kullanılmıştır. Tabloda 2. pooling katmanının çıkış boyutu 5x5x64 şeklindedir ve elde edilen bu çıktı 1x1600 şeklinde yeniden boyutlandırılarak elde edilen bu vektör tam bağlantılı katmana girdi olarak verilmiştir.

***Tablo 2**. CNN modelimizin mimarisi.*

Katman Tipi	**Girdi Boyutu**
Konvolüsyonel (32 adet 5x5 filtre, adım boyu 1)	40x40x3
Konvolüsyonel (64 adet 3x3 filtre, adım boyu 1)	36x36x32
Pooling (2x2 filtre, adım boyu 2)	34x34x64
Konvolüsyonel (32 adet 5x5 filtre, adım boyu 1)	17x17x64
Konvolüsyonel (64 adet 3x3 filtre, adım boyu 1)	13x13x32
Pooling (2x2 filtre, adım boyu 2)	11x11x64
Tam Bağlantılı (256 adet nöron)	1x1600
Tam Bağlantılı (128 adet nöron)	1x256
Softmax Sınıflandırıcı (43 adet nöron)	1x128

Tablo 2'de belirtildiği şekilde bulunan CNN tabanlı sınıflandırıcımız daha sonra eğitim verisinin tamamı kullanılarak aynı prensipler ile 1 kere eğitilmiş ve test verisi ile test edildiğinde ise %96 seviyesinde doğruluk bulunmuştur. Böylece 3. adımda kullanacağımız sınıflandırıcımız belirlenmiştir.

GTSDB veri setinde 43 sınıf bulunmaktadır ve sonunda sınıflandırıcımız ilgili girdi için bu sınıflardan her biri için olasılık değerleri vermektedir. Biz burada olasılık değerleri için eşik değerimizi 0.7 olarak belirledik. Yaklaşımımıza göre eğer sınıflandırıcımız bir bölge önerisini 0.7 veya daha fazla bir olasılık ile bu 43 sınıftan birine ait görürse bu durumda o bölge önerisi trafik levhası olarak kabul edilir ve sınıfı belirlenir.

Örneğin, Görsel 23'te bir önceki adımın sonunda elde edilen çıktıyı görüyoruz. Görsel 24'te ise şu anki adımın sonunda elde edilen

çıktıyı görüyoruz. Sınıflandırıcımızın genel olarak iyi çalıştığını ve trafik levhası içermeyen bölgeleri elerken trafik levhası içeren bölgeleri elemediğini görüyoruz. Görsel 24'te referans verilerine göre bu resimde belirtilmiş olan 3 trafik levhasını sınıflandırıcımızın hatasız bir şekilde tanıdığını görüyoruz.

Görsel 23. *2. adım sonunda bölge önerileri.*

Görsel 24. *3. adım sonunda tespit edilen trafik levhaları.*

Algoritmamızın sonucuna göre eğer birden fazla sayıda ve kesişim birleşim oranı (Intersection over Union (IoU)) 0.7'den büyük bölge önerilerinin aynı sınıfı gösterdiği tespit edilirse en büyük olasılık ile o sınıfı gösteren bölge önerisi alınır ve diğerleri kaldırılır. IoU için olan formül 7. eşitlikte verilmektedir.

$$IoU = \frac{2\ bölgenin\ kesişim\ alanı}{2\ bölgenin\ birleşim\ alanı} \qquad (7)$$

5. Sonuçlar

Algoritmamızın eğitimini ve testini şimdiye kadar anlatılan prensipler ve gösterilen metodu kullanarak gerçekleştirdik ve GTSDB veri setini kullandık.

Görsel 25, Görsel 26, ve Görsel 27'de GTSDB veri setindeki bazı test resimleri üzerinde yöntemimizin elde ettiği trafik levhası tespiti sonuçları gösterilmiştir. Bu görsellerde mavi kare ilgili bölge önerisinin trafik levhası segmentasyonu sonucunda, yeşil kare ilgili bölge önerisinin kontur tespiti sonunda, kırmızı kare ise ilgili bölge önerisinin direk bağlantısı sayesinde elde edildiğini göstermektedir.

Görsel 25. *GTSDB veri setinden 1. test örneği için sonuçlar.*

Görsel 26. *GTSDB veri setinden 2. test örneği için sonuçlar.*

Görsel 27. *GTSDB veri setinden 3. test örneği için sonuçlar.*

Test verisi üzerindeki değerlendirmemiz sonucunda Tablo 3'teki sonuçları elde ettik. Tablo 3'teki sonuçlar trafik levhasını doğru şekilde tespit edip tanımış olmak açısından verilmiştir ve bunu ölçmektedir. Diğer çalışmalar ile de kıyaslayınca sonuçlarımız umut verici ve başarılı görünmektedir. Tabloda da görüldüğü gibi bazı çalışmalarda trafik levhalarının sınıfları sadece yasak gösteren, zorunluluk gösteren ve tehlike gösteren

levhalar olarak sınıflandırılmıştır. Burada precision, recall ve F-Measure isimli metrikleri kullandık. Bu metrikler 8., 9. ve 10. eşitlikler ile verilen formüller kullanılarak hesaplanabilir.

$$precision = \frac{doğru\ yakalanan\ levha\ sayısı}{yakalanan\ toplam\ levha\ sayısı} \quad (8)$$

$$recall = \frac{doğru\ yakalanan\ levha\ sayısı}{gerçek\ levha\ sayısı} \quad (9)$$

$$F_{measure} = \frac{2*precision*recall}{precision+recall} \quad (10)$$

Tablo 3. *Metriklerimize göre bizim bulduğumuz sonuçlar ve diğer araştırmacıların sonuçları.*

Kaynak	Yöntem	Precision	Recall	F-Measure
(Yuan vd, 2015)	Çizge	%89.65	%87.84	%88.73
(Xu vd, 2019)	Renge göre segmentasyon ve şekle göre hipotez testi	%93.96	%95.27	%94.61
(Agrawal ve Chaurasiya, 2017)	HSI tabanlı segmentasyon	Yasak: %91.89 Zorunlu: %95.45 Tehlike: %93.33 Genel: %92.86	Yasak: %84.47 Zorunlu: %85.71 Tehlike: %90.32 Genel: %86.03	Yasak: %88.02 Zorunlu: %90.32 Tehlike: %91.80 Genel: %89.30
(Ellahyani vd, 2016)	HSI tabanlı segmentasyon	%90.13	%91.07	%90.60
(Arcos-García vd, 2018)	Faster R-CNN Inception Resnet V2	Yasak: %91.37 Zorunlu: %89.16 Tehlike: %90.11 Genel: %90.68	Yasak: %96.99 Zorunlu: %79.31 Tehlike: %92.19 Genel: %92.71	Yasak: %94.10 Zorunlu: %83.95 Tehlike: %91.13 Genel: %91.59
(Torres vd, 2019)	Faster R-CNN	%94	%91	%93
(Rahman vd, 2019)	Aktivasyon haritası ile kontur tespiti	%90.8	%80	%85.06
Önerilen Yöntem	Derin öğrenme ile segmentasyon	%90.81	%94.76	%92.74

Deneylerimizi Intel i5 8250U model 1.60 GHz frekanslı CPU, 16 GB RAM ve NVIDIA GeForce 940MX model 384 CUDA çekirdeği ve 2 GB RAM içeren bir GPU'ya sahip bir bilgisayar üzerinde yaptık ve bir resim için ortalama işlem zamanını 0.416 saniye olarak bulduk.

Tablo 1'de verilen algoritma kısımlarından 2. kısmı uygulamamamız ve doğrudan 3. kısma geçmemiz durumunda ise recall ve F-measure metriklerinde 0.01 ve 0.001 civarında iyileşme elde ediyoruz. Bu gelişmenin sebebi 2. kısımda bazı trafik levhası içeren bölge önerilerinin eleniyor olması ve biz bu etabı atladığımızda bu bölge önerilerinin elenmeyip 3. kısımda CNN sınıflandırıcımız tarafından değerlendiriliyor olmasıdır. Hipotez testine göre daha karmaşık ve akıllı bir yapı olması sayesinde CNN bu sınıflandırma sürecinde daha iyi çalışmaktadır ve metriklerimizde iyileşme olmaktadır. Bununla birlikte karmaşık yapısı yüzünden CNN sınıflandırıcının çalışması daha uzun sürmektedir ve bir resim için bu şekilde daha fazla bölge önerisi ile daha fazla sayıda sınıflandırma yapması durumunda ortalama çalışma/işlem süresi deney sonuçlarımıza göre bir resim için yaklaşık 0.05 saniye artmaktadır. Bu yüzden, burada hipotez testimizin bize işlem zamanı anlamında daha iyi bir model elde edebilmek için katkı sağladığını görüyoruz. İşlem zamanı açısından sağlanan bu katkı karşılığında ortaya

çıkan performans düşüşü ise kabul edilebilir seviyededir.

bulunan sonuçlar Tablo 4'te verilmiştir. Burada değerlendirme yaparken deneylerin yapıldığı bilgisayarların donanım özelliklerini de göz önünde bulundurmak gerekmektedir. Özellikle yüksek sayıda çekirdek içeren bir GPU desteği almanın paralel işlem yeteneğini önemli ölçüde yükselteceğinden bu durumun bizim yaptığımız gibi derin öğrenmeye dayalı bir çalışmada oldukça önemli olduğunu söylemek gerekir. Bizim çalışmamızda Bir resim için ortalama işlem süresi bakımından literatürdeki diğer yaklaşımlarda kullandığımız bilgisayar özellikle GPU desteği bakımından diğer çalışmalarda kullanılan bilgisayarların gerisinde görünmektedir. Bununla birlikte, yine de, resim başına ortalama işlem süresi bakımından iyi sonuçlar almaktayız. Daha güçlü bir donanım ve özellikle de daha güçlü bir GPU desteği alınması durumunda bu sürenin daha da azalacağını yani performansın iyileşeceğini söyleyebiliriz.

Tablo 4. *İşlem süresi ve kullanılan donanım açısından modellerin karşılaştırılması.*

Kaynak	**Yöntem**	**Resim Başına Ortalama İşlem Süresi**	**Donanım**
(Yuan vd, 2015)	Çizge	0.341 s	Intel Core i7 QUAD 3.70 GHz CPU ve GPU
(Xu vd, 2019)	Renge göre segmentasyon ve şekle göre hipotez testi	0.930 s	Belirtilmemiş
(Agrawal ve Chaurasiya, 2017)	HSI tabanlı segmentasyon	Belirtilmemiş	Belirtilmemiş
(Ellahyani vd, 2016)	HSI tabanlı segmentasyon	Belirtilmemiş	Belirtilmemiş
(Arcos-García vd, 2018)	Faster R-CNN Inception Resnet V2	0.442 s	Intel Core i7 4770 CPU, 16 GB RAM ve NVIDIA Titan Xp 3840 CUDA çekirdeği ve 12 GB RAM içeren GPU
(Torres vd, 2019)	Faster R-CNN	Belirtilmemiş	2 adet intel Xeon E5606 2.13 GHz CPU ve RAM boyutları 24 GB ve 31 GB olan 2 adet NVIDIA TITAN Xp GPU
(Rahman vd, 2019)	Aktivasyon haritası ile kontur tespiti	Belirtilmemiş	Belirtilmemiş
Önerilen Yöntem	Derin öğrenme ile segmentasyon	0.416 s	Intel i5 8250U 1.60 GHz CPU, 16 GB RAM ve NVIDIA GeForce 940MX 384 CUDA çekirdeği ve 2 GB RAM içeren GPU

6. Çıkarımlar

Bu çalışmada, derin öğrenme tabanlı bir trafik levhası tespit sistemi eğittik ve testlerini yaptık. İlk olarak, resim üzerinde, ENet tabanlı bir görüntü segmentasyonu yaptık ve trafik levhaları için bölge önerilerini tespit ettik.

Sonrasında, hipotez testi yaptık ve bazı yanlış tespit edilen, trafik levhası içermeyen, bölge önerilerini eledik. Burada, hipotez testinin basit ve hızlı çalışan yapısından yararlandık. En son kısımda ise kalan bölge önerilerini CNN sınıflandırıcımıza girdi olarak verdik ve bu sınıflandırıcımız ile bölge önerileri hakkındaki son kararımızı verdik ve sınıflandırma yaptık. Bu şekilde 3 adımdan oluşan yaklaşımımızın tespit ve sınıflandırma açısından iyi sonuçlar verdiğini gördük. Burada, ayrıca, hipotez testinin özellikle işlem zamanı kazanma anlamında bize iyi bir katkı yaptığını ve kabul edilebilir performans metrik değer düşüşü karşılığında daha hızlı çalışan bir model elde etmemizi sağladığını da gördük.

Burada bir başka ilginç nokta aynı çalışmayı Viyana Karayolu İşaretleri ve Sinyalleri Sözleşmesine üye olan bir başka ülkeye ait veri setini kullanarak yapmak ve sonuçları incelemek olabilir. Örneğin Belçika bu ülkelerden biridir ve Belgium Traffic Sign Detection (BTSD) veri seti böyle bir çalışmada kullanılabilir. Daha önceden eğitilmiş olarak alıp kullandığımız ENet modeli, çalışmada bahsettiğimiz şekilde, Cityscapes veri seti ile eğitilerek oluşturuldu ve Cityscapes veri seti çoğunlukla Almanya ve belli bir ölçüde de komşu ülkelerinden görüntüler içermektedir. Bu ülkelerin hepsi de bahsedilen sözleşmeye üyedir. Bu sözleşme kapsamındaki bir başka ülkeye ait veri seti ile aynı çalışmanın tekrarlanması durumunda benzer sonuçların alınacağını söyleyebiliriz. Bir başka ilginç çalışma da sözleşme kapsamında olmayan bir ülke ile bu çalışmayı tekrarlamak olabilir. Böyle bir durumda ise sonuçların belli bir ölçüde kötüleşeceğini sezgisel olarak düşünebiliriz.

Kaynaklar

Abedin, Z., Dhar, P., Hossenand, M. K., ve Deb, K. (2017). Traffic Sign Detection and Recognition Using Fuzzy Segmentation Approach and Artificial Neural Network Classifier Respectively. *International Conference on Electrical, Computer and Communication Engineering*, (s. 518-523). Cox's Bazar.

Agrawal, S., ve Chaurasiya, R. K. (2017). Automatic Traffic Sign Detection and Recognition Using Moment Invariants and Support Vector Machine. *International conference on Recent Innovations is Signal Processing and Embedded Systems*.

Arcos-García, A., Álvarez-García, J. A., ve Soria-Morillo, L. M. (2018). Evaluation of deep neural networks for traffic sign detection systems. *Neurocomputing, 316,* 332-344.

Berkaya, S. K., Gunduz, H., Ozsen, O., Akinlar, C., ve Gunal, S. (2016). On circular traffic sign detection and recognition. *Expert Systems With Applications, 48,* 67-75.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., . . . Schiele, B. (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Ellahyani, A., Ansari, M. E., ve Jaafari, I. E. (2016). Traffic sign detection and recognition based on random forests. *Applied Soft Computing, 46,* 805-815.

Fleyeh, H., ve Dougherty, M. (2005). Road and traffic sign detection and recognition. *16th Mini-EURO Conference and 10th Meeting of EWGT*, (s. 644-653).

Ghimire, S., ve Wang, H. (2012). Classification of image pixels based on minimum distance and hypothesis testing. *Computational Statistics & Data Analysis, 56*, 2273-2287.

Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., ve Igel, C. (2013). Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark. *International Joint Conference on Neural Networks (IJCNN)*.

Jung, S., Lee, U., Jung, J., ve Shim, D. H. (2016). Real-time Traffic Sign Recognition System with Deep Convolutional Neural Network. *13th International Conference on Ubiquitous Robots and Ambient Intelligence*. Urai.

Li, Y., Møgelmose, A., ve Trivedi, M. M. (2016). Pushing the "Speed Limit": High-Accuracy U.S. Traffic Sign Recognition with Convolutional Neural Networks. *IEEE Transactions on Intelligent Vehicles*, 167-176.

Liao, S. M., ve Akritas, M. (2007). Test-based classification: A linkage between classification and statistical testing. *Statistics & Probability Letters, 77,* 1269-1281.

Liu, X., Deng, Z., ve Yang, Y. (2019). Recent progress in semantic image segmentation. *Artificial Intelligence Review*, 1089–1106.

Malik, Z., ve Siddiqi, I. (2014). Detection and Recognition of Traffic Signs from Road Scene Images. *12th International Conference on Frontiers of Information Technology*, (s. 330-335).

Mathias, M., Timofte, R., Benenson, R., ve Van Gool, L. (2013). Traffic sign recognition — How far are we from the solution? *International Joint Conference on Neural Networks (IJCNN)*, (s. 1-8). Dallas.

Paszke, A., Chaurasia, A., Kim, S., ve Culurciello, E. (2016). Enet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation. *arXiv: 1606.02147*.

Rahman, Q. M., Sünderhauf, N., ve Dayoub, F. (2019). Did You Miss the Sign? A False Negative Alarm System for Traffic Sign Detectors. *arXiv:1903.06391*.

Serna, C. G., ve Ruichek, Y. (2018). Classification of Traffic Signs: The European Dataset. *IEEE Access, 4,* 78136-78148.

Song, S., Que, Z., Hou, J., Du, S., ve Song, Y. (2019). An efficient convolutional neural network for small traffic sign detection. *Journal of Systems Architecture, 97,* 269-277.

Stallkamp, J., Schlipsing, M., Salmen, J., ve Igel, C. (2011). The German Traffic Sign Recognition Benchmark: A multi-class classification competition. *International Joint Conference on Neural Networks*, (s. 1453-1460).

Stallkamp, J., Schlipsing, M., Salmen, J., ve Igel, C. (2012). Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition. *Neural networks*, 323–332.

Sun, D., ve Ho, M. (2011). Image Segmentation via Total Variation and Hypothesis Testing Methods.

Suzuki, S., ve Abe, K. (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing, 30*, 32–46.

Temel, D., Alshawi, T., Chen, M.-H., ve AlRegib, G. (2019). Challenging Environments for Traffic Sign Detection: Reliability Assessment under Inclement Conditions. *arXiv:1902.06857*.

Torres, L. T., Paixao, T. M., Berriel, R. F., De Souza, A. F., Badue, C., Sebe, N., ve Oliveira-Santos, T. (2019). Effortless Deep Training for Traffic Sign Detection Using Templates and Arbitrary Natural Images. *International Joint Conference on Neural Networks*. Budapest.

Wong, A., Shafiee, M. J., ve Jules, M. S. (2018). MicronNet: A Highly Compact Deep Convolutional Neural Network Architecture for Real-Time Embedded Traffic Sign Classification. *IEEE Access, 6,* 59803–59810.

Xu, X., Jin, J., Zhang, S., Zhang, L., Pu, S., ve Chen, Z. (2019). Smart data driven traffic sign detection method based on adaptive color threshold and shape symmetry. *Future Generation Computer Systems, 94,* 381-391.

Yuan, X., Guo, J., Hao, X., ve Chen, H. (2015). Traffic Sign Detection via Graph-Based Ranking and Segmentation Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 45,* 1509-1521.