

GÖRÜNTÜ TABANLI OTOMATİK YOL TESPİTİ VE MODELLENMESİ

Fevzi DAŞ

DOKTORA TEZİ

Uzaktan Algılama ve Coğrafi Bilgi Sistemleri Anabilim Dalı

Danışman: Doç. Dr. Uğur AVDAN

Eskişehir

Eskişehir Teknik Üniversitesi

Lisansüstü Eğitim Enstitüsü

Ocak 2020

Bu tez çalışması BAP Komisyonu tarafından kabul edilen 1508F599 no.lu proje kapsamında desteklenmiştir.

JÜRİ VE ENSTİTÜ ONAYI

Fevzi DAŞ'ın GÖRÜNTÜ TABANLI OTOMATİK YOL TESPİTİ VE MODELLENMESİ başlıklı tezi 17/01/2020 tarihinde aşağıdaki jüri tarafından değerlendirilerek “Eskişehir Teknik Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği”nin ilgili maddeleri uyarınca, Uzaktan Algılama ve Coğrafi Bilgi Sistemleri Anabilim dalında Doktora tezi olarak kabul edilmiştir.

Jüri Üyeleri

Unvan Adı Soyadı

İmza

Üye (Tez Danışmanı)

: Doç. Dr. Uğur AVDAN

Üye

: Doç. Dr. Kemal ÖZKAN

Üye

: Dr.Öğr.Üyesi Nusret DEMİR

Üye

: Dr.Öğr.Üyesi Emrah PEKKAN

Üye

: Dr.Öğr.Üyesi Gordana KAPLAN

Prof. Dr. Murat TANIŞLI

Lisansüstü Eğitim Enstitüsü Müdürü

ÖZET

GÖRÜNTÜ TABANLI OTOMATİK YOL TESPİTİ VE MODELLENMESİ

Fevzi DAŞ

Uzaktan Algılama ve Coğrafi Bilgi Sistemleri Anabilim Dalı

Eskişehir Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Ocak 2020

Danışman: Doç. Dr. Uğur AVDAN

Gelişen teknoloji ile birlikte bireyler artık sadece bilgi tüketen değil aynı zamanda bilgi üreten kullanıcılar haline gelmişlerdir. Bu durum özellikle açık kaynak birçok proje için temel bilgi kaynağını oluşturmaktadır. OpenStreetMap (OSM) ve OpenAerialMap (OAM) bu bakış açısıyla oluşturulmuş ve oldukça yaygın olan projelerdendir. OSM dünyaya ait özgür bir harita üretmek amacıyla ortaya çıkan bir projedir. OAM ise dünyaya ait yüksek çözünürlüklü uydu ve hava görüntülerini depolayan ve paylaşan bir platformdur. Diğer yandan özellikle 2000'li yıllardan sonra hızla gelişen ve fiyatları düşen bilgisayar teknolojileri ile birlikte makine öğrenmesi ve yapay sinir ağı alanlarında (YSA) önemli gelişmeler ve ilerlemeler kaydedilmiştir. Özellikle derin öğrenme ve evrimsel sinir ağları ile birlikte görüntü işleme yöntemlerindeki klasik programlama mantığı tamamen değişmiştir. Makine öğrenmesinin ve derin sinir ağlarının başarısını etkileyen en önemli faktörlerden birisi sistemin eğitilmesi için girdi olarak kullanılacak yeterli miktarda verinin temin edilmesidir. Bu çalışma kapsamında OAM'den elde edilen uzaktan algılama görüntüleri ve OSM'den elde edilen vektörel bilgi kullanılarak denetimli sınıflandırma için bir veri seti oluşturulmuştur. Bu veri seti, anlamsal bölütleme tekniği kullanılarak yol ağlarının tespiti amaçlayan modeli eğitmek ve test etmek için kullanılmıştır. Çalışmada özelleştirilmiş tam evrimsel ağ (TEA), SegNet ve PSPNet gibi farklı modeller kullanılmış ve sonuçlar karşılaştırmalı olarak verilmiştir. Değerlendirme sonuçlarına göre en başarılı sonuçlar TEA modeli ile elde edilmiştir. Bu modelin piksel doğruluğu 0.7102, kesinlik değeri 0.7826, hassasiyet değeri 0.7949 ve F_1 skoru 0.7887 olarak hesaplanmıştır.

Anahtar Sözcükler: Yol ağı tespiti, Evrimsel sinir ağları, Anlamsal bölütleme, Openstreetmap, Openaerialmap

ABSTRACT

IMAGE BASED AUTOMATIC ROAD DETECTION AND MODELING

Fevzi DAŞ

Department of Remote Sensing and Geographical Information System
Eskişehir Technical University, Institute of Graduate Programs, January 2020

Supervisor: Assoc. Prof. Uğur AVDAN

With the developing technology, individuals have become users who are not only consuming but also producing information. This is particularly the main source of information for many open source projects. OpenStreetMap (OSM) and OpenAerialMap (OAM) are very common projects created with this perspective. OSM is a project that has emerged to produce a free world map. OAM is a platform for storing and sharing high resolution satellite and aerial imagery of the world. On the other hand, significant developments and progress have been made in machine learning and artificial neural network (ANN) areas, especially with the rapid developing and cheaper prices of computer technologies after the 2000s. The concept of classical programming, especially in deep learning and convolutional neural networks, has completely changed. One of the most important factors affecting the success of machine learning and deep neural networks is the provision of sufficient data to be used as input for training the system. In this study, a data set for supervised classification was created by using high resolution aerial images obtained from OAM and vector information obtained from OSM. This data set was used to train and test the model for the detection of road networks using semantic segmentation technique. Different models such as customized fully convolutional network (FCN), SegNet and PSPNet were used in the study and the results obtained are given comparatively. According to the evaluation results, the most successful results were obtained with the FCN. The pixel accuracy of this model is 0.7102, the recall value is 0.7826, the precision value is 0.7949 and the F_1 score is 0.7887.

Keywords: Road network detection, Convolutional neural networks, Semantic Segmentation, Openstreetmap, Openaerialmap.

TEŞEKKÜR

Tez çalışmamın başından sonuna kadar desteklerini esirgemeyen ve her zaman yanımda olan Değerli Hocam Doç. Dr. Uğur AVDAN'a, tez izleme komitemde yer alan ve yönlendirmeleri ile tezimin doğru bir şekil almasını sağlayan Değerli Hocalarım Doç. Dr. Kemal ÖZKAN'a ve Dr. Öğr. Üyesi Nusret DEMİR'e, teşekkür ederim. Ayrıca çalışmamın her aşamasında yanımda olan ve bana sonsuz destek olan eşim Gülşen ATAŞ DAŞ'a, ablam Rasime DAŞ'a, aileme ve Canım Kızım Zeynep'e en içten dileklerimle teşekkür ediyorum.

Fevzi DAŞ

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Bu tezin bana ait, özgün bir çalışma olduğunu; çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarında bilimsel etik ve kurallara uygun davrandığımı; bu çalışma kapsamında elde edilen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi; bu çalışmanın Eskişehir Teknik Üniversitesi tarafından kullanılan “bilimsel intihal tespit programı”yla tarandığını ve hiçbir şekilde “intihal içermediğini” beyan ederim. Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçları kabul ettiğimi bildiririm.

Fevzi DAŞ

İÇİNDEKİLER

	<u>Sayfa</u>
BAŞLIK SAYFASI	i
JÜRİ VE ENSTİTÜ ONAYI.....	ii
ÖZET	iii
ABSTRACT.....	iv
TEŞEKKÜR.....	v
ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ.....	vi
İÇİNDEKİLER	vii
TABLolar DİZİNİ.....	xi
ŞEKİLLER DİZİNİ.....	xii
SİMGELER VE KISALTMALAR DİZİNİ.....	xv
1. GİRİŞ	1
1.1. Çalışmanın Amacı ve Önemi.....	3
1.2. Çalışmanın Kapsamı ve Sınırlılıkları.....	4
1.3. Literatür Özeti.....	4
1.3.1. Sınıflandırmaya dayalı yöntemler	5
1.3.1.1. Denetimli sınıflandırma yöntemleri.....	6
1.3.1.2. Denetimsiz sınıflandırma yöntemleri.....	12
1.3.1.3. Bilgi tabanlı yöntemler	14
1.3.1.4. Matematiksel morfoloji yöntemleri	15
1.3.1.5. Aktif kontur modeli	16
1.3.1.6. Dinamik programlama ve gruplama	18
1.4. Problemin Tanımı	19
1.5. Çalışmanın Hipotezi.....	20
1.6. Çalışmanın Özgün Değeri ve Yaygın Etkisi.....	21
2. KURAMSAL TEMELLER	22

	<u>Sayfa</u>
2.1. Topluluk Tabanlı Açık Kaynak Projeler	22
2.2. OpenAerialMap.....	22
2.2.1. OpenAerialMap platformundan veri indirme.....	25
2.3. OpenStreetMap	25
2.3.1. OpenStreetMap veri modeli	27
2.3.2. OpenStreetMap yol verileri.....	30
2.3.3. OpenStreetMap verisini indirme	31
2.4. Sınırlayıcı Kutu (Bounding Box)	32
2.5. Datum, Projeksiyon ve Koordinat Sistemi	33
2.5.1. Koordinat sistemi dönüşümü	33
2.6. Yapay Zekâ.....	35
2.6.1. Yapay zekâ uygulamaları	36
2.6.2. Yapay zekanın sınıflandırılması.....	37
2.7. Makine Öğrenmesi.....	38
2.7.1. Makine öğrenmesi algoritmaları.....	38
2.7.1.1. Denetimli (supervised) öğrenme.....	38
2.7.1.2. Denetimsiz (unsupervised) öğrenme	39
2.7.1.3. Yarı denetimli (reinforcement) öğrenme.....	40
2.7.2. Model seçimi ve hiper parametreler	41
2.7.3. Aşırı, yetersiz ve yeterli öğrenme	42
2.8. Yapay Sinir Ağları (Artificial Neural Networks).....	44
2.8.1. Algılayıcı (perceptron)	46
2.8.2. Bir yapay sinir ağı modeli.....	47
2.8.3. Yapay sinir ağı bileşenleri	48
2.8.3.1. Girdiler (input) ve ağırlıklar (weight)	49
2.8.3.2. Toplama fonksiyonu	49
2.8.3.3. Aktivasyon fonksiyonları	50
2.8.4. Kayıp (loss) fonksiyonu.....	54
2.8.4.1. Regresyon kayıp fonksiyonları	56

	<u>Sayfa</u>
2.8.4.2. Sınıflandırma kayıp fonksiyonları	57
2.8.5. Düzenleştirme (regularization)	58
2.8.6. Optimizasyon metotları	60
2.8.7. Geri yayılım (back propagation).....	62
2.9. Derin Öğrenme	64
2.9.1. Derin öğrenme nedir?	65
2.9.1.1. Derin öğrenme kütüphaneleri	65
2.10. Evrişimsel Sinir Ağı (Convolutional neural networks)	66
2.10.1. Evrişimsel sinir ağı katmanları	67
2.10.1.1. Normalleştirme	70
2.10.1.2. Biriktirme katmanları (Pooling layers)	70
2.10.1.3. Dolgu (Padding)	71
2.11. Görüntü Bölütleme (Image Segmentation).....	72
2.11.1. Anlamsal bölütleme (Semantic segmentation).....	74
2.11.1.1. ESA ile görüntü bölütleme.....	75
2.11.1.2. Görüntü bölütleme için derin öğrenme teknikleri.....	76
2.11.1.3. Anlamsal bölütleme ağını oluşturma	82
2.11.2. Örnek bölütleme (Instance segmentation)	83
2.12. Model Performansını Değerlendirme.....	83
2.12.1. Karışıklık matrisi (confusion matrix).....	84
2.12.2. Piksel doğruluğu	85
2.12.3. Kesinlik (precision), recall (hassasiyet), ve F ₁ skorları	85
2.12.4. Ortalama IoU ve frekans ağırlıklı IoU	86
3. MATERYAL VE YÖNTEM.....	87
3.1. Donanım ve İşletim Sistemi	87
3.2. Yazılım Geliştirme Araçları ve Ortamı.....	87
3.2.1. Programlama dili.....	87
3.2.1.1. Python kütüphaneleri.....	87
3.2.2. Yazılım geliştirme editörü	88

	<u>Sayfa</u>
3.2.3. Kod yapısı.....	88
3.2.4. Derin öğrenme kütüphanesi	89
3.3. Çalışma Alanı ve Konumu	90
3.4. Materyal ve Yöntem.....	92
3.4.1. Dizin yapısının oluşturulması.....	93
3.4.2. Veri setinin hazırlanması.....	95
3.4.2.1. Hava görüntülerinin elde edilmesi	95
3.4.2.2. Vektör verinin elde edilmesi	97
3.4.2.3. Yol genişliklerinin hesaplanması.....	99
3.4.2.4. Varsayılan yol genişliklerinin belirlenmesi	101
3.4.2.5. Hesaplama adımları ve ayrıntıları	103
3.4.2.6. Vektör verilerin raster veriye dönüştürülmesi	107
3.4.2.7. Görüntülerin dilimlenmesi.....	111
3.4.2.8. Uygun olmayan görüntülerin filtrelenmesi	112
3.4.2.9. Veri setinin bölünmesi.....	116
3.4.2.10. Çalışma alanlarının birleştirilmesi.....	118
3.4.3. Veri seti.....	119
3.5. Modellerin Belirlenmesi.....	121
3.6. Model hiper parametrelerinin belirlenmesi.....	121
3.7. Öğrenme Aktarımı (Transfer Learning)	122
3.8. Eğitim ve Uygulama Ayrıntıları	122
4. ARAŞTIRMA BULGULARI.....	123
4.1. Eğitim ve Test Verisi Bulguları.....	123
4.2. Model Değerlendirme	124
4.2.1. Model Sonuçlarını Görsel Olarak Değerlendirme	125
4.2.1.1. Uygulama 1 – Sade görüntüler	126
4.2.1.2. Uygulama 2 - Normal görüntüler.....	128
4.2.1.3. Uygulama 3 - Karmaşık görüntüler.....	129
5. TARTIŞMA.....	133

	<u>Sayfa</u>
6. SONUÇ VE ÖNERİLER.....	136
KAYNAKÇA.....	140
EKLER	
ÖZGEÇMİŞ	



TABLÖLAR DİZİNİ

Sayfa

Tablo 1.1. Çalışma alanları, sınırlayıcı kutu değerleri (BBOX) ve enlem, boylam bilgileri	4
Tablo 2.1. OSM'deki yol türleri ve temel özellikleri	30
Tablo 2.2. (Devam) OSM'deki yol türleri ve temel özellikleri	31
Tablo 3.1. Eğitim ve test işlemlerinin yapıldığı bilgisayar konfigürasyonu	87
Tablo 3.2. Çalışma kapsamında oluşturulan modüller ve işlevleri	89
Tablo 3.3. Çalışma alanları ve sınırlayıcı kutu değerleri (BBOX)	90
Tablo 3.4. OSM'deki yol türleri için belirlenen standart değerler.....	102
Tablo 3.5. Veri seti detayları	120
Tablo 3.6. Çalışma kapsamında kullanılan her bir modele ait hiper parametreler ve değerleri	122
Tablo 4.1. Temel değerlendirme parametreleri ve modellerin aldıkları değerler	123
Tablo 4.2. Modellerin Stara Zagora bölgesinden elde edilen değerlendirme verileri için ürettikleri sonuçlar	125
Tablo 4.3. Modellerin farklı metriklere göre değerlendirme sonuçları.....	126

ŞEKİLLER DİZİNİ

Sayfa

Şekil 2.1. Mersin bölgesindeki bir alana ait 23.12.2018 tarihinde çekilmiş 2 cm mekânsal çözünürlüklü OAM görüntüsü	23
Şekil 2.2. OAM katalog bileşeni ve alt bileşenleri	24
Şekil 2.3. OAM Contributor bileşeni ve alt bileşenleri	25
Şekil 2.4. OSM web arayüzü	26
Şekil 2.5. (a) Temsili bir gerçek dünya görüntüsü, (b) Temsili gerçek görüntünün raster formatındaki gösterimi, (c) Temsili gerçek görüntünün vektör formatındaki gösterimi.....	28
Şekil 2.6. OSM veri modeli yapıları.....	29
Şekil 2.7. (a) EPSG:32635 koordinat referans sistemine sahip görüntü (b) EPSG:4326 koordinat referans sistemine dönüştürülmüş görüntü	35
Şekil 2.8. Yapay zekâ ve yapay zekadan türetilen yeni yaklaşımlar	36
Şekil 2.9. Makine Öğrenmesi algoritmaları.....	38
Şekil 2.10. Sınıflandırma ve regresyon problemlerinin görsel olarak ifade edilmesi	39
Şekil 2.11. Yarı denetimli öğrenme modeli yapısı	40
Şekil 2.12. Yetersiz öğrenme, yeterli öğrenme ve aşırı öğrenme işlemlerine ait örnek grafikler	43
Şekil 2.13. Algılayıcı yapısı	46
Şekil 2.14. Biyolojik ve yapay nöronlar	48
Şekil 2.15. Bir yapay sinir ağını oluşturan girdi, gizli ve çıktı katmanları.....	49
Şekil 2.16. Doğrusal aktivasyon fonksiyonu	51
Şekil 2.17. Sigmoid aktivasyon fonksiyonu	52
Şekil 2.18. Tanh aktivasyon fonksiyonu	53
Şekil 2.19. ReLU aktivasyon fonksiyonu.....	54
Şekil 2.20. Çapraz entropi kayıp fonksiyonu	57
Şekil 2.21. Dropout işlemi ile bazı nöronların ağdan çıkarılması	59
Şekil 2.22. YSA modeli eğitim süreci	60
Şekil 2.23. İki boyutlu bir parabol örneği.....	61

Şekil 2.24. Batch Gradient Descent, Stochastic Gradient Descent ve Mini-batch Gradient Descent yöntemleri	62
Şekil 2.25. Geriye yayılım algoritmasının çalışma mantığı	63
Şekil 2.26. Bir evrişimsel sinir ağı katmanının temel yapısı ve bileşenleri.....	67
Şekil 2.27. Giriş görüntüsü üzerine evrişim işleminin uygulanması.....	68
Şekil 2.28. Bir görüntü üzerine filtre uygulanarak evrişim işleminin yapılması	69
Şekil 2.29. Maksimum biriktirme katmanı	70
Şekil 2.30. (a) Dolgulama uygulanmamış bir görüntüye evrişim işleminin uygulanması (b) Bir giriş örüntüsüne dolgulama işleminin ve evrişim işleminin uygulanması	72
Şekil 2.31. Sınıflandırma, anlamsal bölütleme, nesne algılama ve örnek bölütleme	73
Şekil 2.32. Anlamsal bölütleme (b) Örnek bölütleme	73
Şekil 2.33. Gerçek bir görüntü ve bu görüntünün anlamsal bölütleme ile sınıflandırılması	74
Şekil 2.34. Anlamsal bölütleme ağı.....	75
Şekil 2.35. TEA'da alt örnekleme ve üst örnekleme.....	76
Şekil 2.36. Anlamsal bölütleme ağı yapısı	77
Şekil 2.37. U-Net model yapısı ve atlama bağlantıları.....	78
Şekil 2.38. Üst kısım 34 Katman ResNet ESA ve alt kısım 34 Katman CNN.....	79
Şekil 2.39. Artık öğrenme: bir yapı taşı	80
Şekil 2.40. PSPNet ağı.....	81
Şekil 2.41. Anlamsal bölütleme ağı oluşturma adımları	82
Şekil 2.42. Karışıklık matrisi (confusion matrix)	84
Şekil 2.43. Kesinlik ve hassasiyet metriklerinin karışıklık matrisindeki değerler kullanılarak hesaplanması	85
Şekil 3.1. (a) Tübingen (Almanya) (b) Minsk (Belarus) ve (c) Saint Petersburg (Rusya) çalışma alanları ve sınırlayıcı koordinat değerleri	91
Şekil 3.2. Çalışmanın tüm adımlarını içeren temel akış şeması	93
Şekil 3.3. Her bir çalışma alanı için oluşturulan dizin yapısı	94

Şekil 3.4. Tübingen çalışma alanına ait meta bilgisinin bir bölümü	96
Şekil 3.5. Tübingen (Almanya) ve (b) Saint Petersburg (Rusya) (c) Minsk (Belarus) çalışma alanlarına ait bazı örnek yüksek çözünürlüklü görüntüler.....	97
Şekil 3.6. Yol genişliklerinin hesaplanması için yapılan işlemlere ait akış şeması.....	100
Şekil 3.7. Tübingen çalışma alanına ait ham yol ağı verileri	105
Şekil 3.8. Tübingen çalışma alanına ait nihai yol ağı verileri	106
Şekil 3.9. Tübingen çalışma alanına ait nihai yol ağının hiçbir altık harita kullanılmadan gösterimi.....	107
Şekil 3.10. (a) Tübingen (b) Minsk (c) Saint Petersburg çalışma alanlarına ait tüm yol ağlarının raster formatına dönüştürülmüş yol ağı haritaları.....	109
Şekil 3.11. (a)Saint Petersburg’a ait bir uzaktan algılama görüntüsü ve (b) bu görüntü için üretilmiş yol ağlarını içeren raster görüntü	110
Şekil 3.12. (a), (b), (c) çalışma kapsamında elde edilen 512x512 boyutlarına sahip gerçek görüntüler ve (d), (e), (f) ile sırasıyla bu görüntülere karşılık gelen yol bilgilerini içeren ikili görüntülerdir	112
Şekil 3.13. Çeşitli nedenlerden dolayı veri setine dahil edilmeyen bazı örnek görüntüler	113
Şekil 3.14. Derin öğrenme modelinde kullanılan veri setinin dizin yapısı.....	116
Şekil 3.15. Veri seti ayrıntıları.....	121
Şekil 4.1. Modelinin tur sayısına göre piksel doğruluğu ve kayıp değerleri grafikleri	124
Şekil 4.2. İnsan algılaması açısından kolay olarak kabul edilebilecek örnek gerçek görüntüler ve bu gerçek görüntülerin sırasıyla TEA, SegNet ve PSPNet modelleri ile yol ağı tespit sonuçları	128
Şekil 4.3. İnsan algılaması açısından normal zorluk düzeyindeki örnek gerçek görüntüler ve bu gerçek görüntülerin sırasıyla TEA, SegNet ve PSPNet ve modelleri ile yol ağı tespit sonuçları	129
Şekil 4.4. İnsan algılaması açısından zor olarak nitelenebilecek örnek gerçek görüntüler ve bu gerçek görüntülerin sırasıyla TEA, SegNet ve PSPNet modelleri ile yol ağı tespit sonuçları	131

SİMGELER VE KISALTMALAR DİZİNİ

ADALINE	: Adaptive Linear Element
ANN	: Artificial Neural Networks (Türkçe: Yapay Sinir Ağı)
BBOX	: Bounding Box (Türkçe: Sınırlayıcı Kutu)
BPTT	: Backpropagation Through Time (Türkçe: Zaman İçinde Geri Yayılım)
CBS	: Coğrafi Bilgi Sistemleri
CNN	: Convolutional Neural Network
CPU	: Central Processing Unit (Türkçe: Merkezi İşlem Birimi)
CNTK	: Microsoft Cognitive Toolkit
DBN	: Deep Belief Networks (Türkçe: Derin İnanç Ağları)
DNNs	: Deep Neural Networks (Türkçe: Derin Öğrenme Ağları)
ESA	: Evrimsel Sinir Ağı
FCN	: Fully Convolutional Network (Türkçe: Tam Evrimsel Ağ)
FSM	: Finite State Machine (Türkçe: Sonlu Durum Makinası)
GB	: Gigabyte
GPU	: Graphics Processing Unit (Türkçe: Grafik İşlemci Ünitesi)
HCM	: The Highway Capacity Manual
HOT	: Humanitarian OpenStreetMap Team
ILSVRC	: ImageNet Large Scale Visual Recognition Challenge (Türkçe: ImageNet Büyük Ölçekli Görsel Tanıma Yarışması)
İHA	: İnsansız Hava Araçları
KRS	: Koordinat Referans Sistemi
LSTM	: Long Short Term Memory (Türkçe: Uzun Kısa Süreli Bellek)
MAE	: Mean Absolute Error Loss (Türkçe: Ortalama Mutlak Hata Kaybı)
NSE	: Normalized Squared Error (Türkçe: Normalleştirilmiş Kare Hatası)
OAM	: OpenAerialMap
OIN	: Open Imagery Network (Türkçe: Açık Görüntüleme Ağı)
OpenCV	: Open Source Computer Vision Library (Türkçe: Açık Kaynak Kodlu Görüntü İşleme Kütüphanesi)

OSM	: OpenStreetMap
ReLU	: Rectified Linear Units
RGB	: RegGreenBlue
RNN	: Recurrent Neural Network (Türkçe: Tekrarlayan Yapay Sinir)
SAR	: Synthetic Aperture Radar (Türkçe: Yapay Açıklıklı Radar)
SLIC	: Simple Linear Iterative Clustering (Basit Doğrusal Yinelemeli Kümeleme)
TEA	: Tam Evrişimsel Ağ(lar)
Tanh	: Hyperbolic Tangent (Türkçe: Hiperbolik Tanjant)
TB	: Terabyte
$\sum$	: Toplam Simgesi
UA	: Uzaktan Algılama
UAV	: Unmanned Aerial Vehicle (Türkçe: İnsansız Hava Aracı)
UUID	: Universally Unique Identifier
UA	: Uzaktan Algılama
UAV	: Unmanned Aerial Vehicle (Türkçe: İnsansız Hava Aracı)
VHR	: VHR Very High Resolution (Türkçe: Çok Yüksek Çözünürlüklü)
WGS-84	: The World Geodetic System (Türkçe: Dünya Jeodezik Sistemi) 1984
YSA	: Yapay Sinir Ağı

1. GİRİŞ

Son yıllarda bilgisayar ve bilişim teknolojilerinde daha önce hiç olmadığı kadar hızlı gelişmeler yaşanmaktadır. Bu gelişmeler, kısa sürelerde kişisel kullanım alanlarından uzay teknolojilerine kadar hemen hemen her alanda uygulamaya dönüşmekte ve dolayısıyla yaşam biçimini şekillendirmektedir. Görüntü işleme ve bilgisayarlı görü gibi bazı alanlar bu gelişmelerden doğrudan etkilenirken, bilgisayar teknolojilerini yoğun şekilde kullanan uzaktan algılama ve coğrafi bilgi sistemleri (UA ve CBS) gibi alanlar dolaylı etkilenmektedir.

İnternet teknolojilerinin yaygınlaşması ve Web 2.0 konsepti ile birlikte internet kullanıcılarının bilgiye erişim biçimleri ve rolleri büyük oranda değişmiştir. Web 2.0 kullanıcıları merkeze alan, onlara paylaşım ve iş birliği fırsatları sunan ve çevrimiçi ortamda kendilerini ifade etmelerine yardımcı olan ikincil nesil internet uygulamalarını ifade etmektedir. Web 2.0 ile birlikte kullanıcılar tek taraflı içerik istemcilerinden çift taraflı içerik istemcilerine ve sağlayıcılarına evrilmişlerdir. Bu durum sıradan kullanıcıların bilgi ve içerik üretmesine imkân sağlamıştır (Goodchild, 2007). Bu durumu fırsata çevirmek amacıyla dünyanın farklı bölgelerinden çok sayıda fikir veya proje ortaya atılmıştır. Bu fikir veya projeler ile aynı amaca hizmet eden kullanıcılar organize edilmekte, benzer standartlara sahip içeriklerin üretilmesi sağlanmakta ve üretilen içeriklerin diğer kullanıcılar ile paylaşılmasını sağlayan platformların geliştirilmesi amaçlanmaktadır. Literatürde bu projelere genel olarak topluluk tabanlı projeler denilmektedir. Bu yeni içerik üretme modeli, farklı fikirlerin hayata geçirilmesi veya ihtiyaçların karşılanmasını sağlamak amacıyla çeşitli yeni ve benzersiz projelerin ortaya çıkmasını sağlamıştır. Wikipedia bu yaklaşım ile ortaya çıkan en önemli ve en fazla bilinen projelerden birisidir. OpenStreetMap (OSM) ve OpenAerialMap (OAM) 2000’li yıllardan sonra ortaya çıkan iki başarılı topluluk tabanlı projedir. OSM herkesin kullanımına açık, ücretsiz bir harita servisi oluşturma düşüncesi ile hayata geçirilmiştir. OSM'deki içerikler büyük oranda gönüllü kullanıcılar tarafından oluşturulmaktadır. OSM tüm dünyaya ait güncel veriler içermektedir. Open Database License (ODbL) (Open Data Commons, 2019b) ile lisanslanan bu içeriklere erişim ve paylaşım tamamen ücretsizdir. Bu lisans, kullanıcıların ürettikleri içerikler için diğer kullanıcılara aynı özgürlüğü verme koşulu ile bir başkası tarafından üretilen içeriği (veri tabanını) özgürce paylaşmasına, değiştirmesine ve kullanmasına izin vermek için tasarlanmış bir lisans sözleşmesidir. OAM ise sayısal hava görüntülerini barındıran açık kaynaklı bir platformdur. OAM

tarafından sağlanan tüm görüntüler, içeriklere ücretsiz erişilmesini ve paylaşılmasını sağlayan Creative Commons (CC) lisansı ile lisanslanmıştır. Bu lisans ise yapılan bir çalışmanın başka kullanıcılar tarafından ticari amaçla da dahil olmak üzere dağıtmasına, yeniden düzenlemesine, uyarlamasına ve geliştirmesine olanak tanımaktadır (St. Laurent, 2004, s. 98).

Bu çalışma temel olarak şu adımlardan oluşmaktadır: Çalışma alanı olarak belirlenen bölgelere ait uzaktan algılama görüntüleri OAM'dan indirilmekte, (ii) görüntülere ait etiket bilgisi OSM'den elde edilmekte, (iii) indirilen görüntüler ve bu görüntülere ait etiket bilgileri ön işleme süreçleri ile kullanıma hazır hale getirilmekte (iv) görüntüler ve etiketler geliştirilen görüntü sınıflandırma modelinde işlenerek ilgili bölgeye ait yol ağlarının tespit edilmesi ve modellenmesi sağlanmıştır. Çalışmanın amacı ve önemi, kapsamı ve kısıtlılıkları, literatür özeti, problemin tanımı, hipotez ve özgün değer ve yaygın etki giriş bölümünde yer verilen diğer bilgilerdir.

İkinci bölüm olan kuramsal temeller başlığı altında yapılan çalışmanın anlaşılmasını sağlamak amacıyla OSM, OAM, makine öğrenmesi, yapay sinir ağları, derin öğrenme algoritmaları, görüntü bölütleme yöntemleri gibi konular hakkında temel bilgiler sunulmuştur.

Üçüncü bölümde materyal ve yöntem başlığı altında çalışmada kullanılan veri, yöntem, teknik, kütüphane ve algoritmalar hakkında gerekli bilgiler verilmiş, uygulama adımları ayrıntılı bir şekilde açıklanmıştır.

Dördüncü bölümde araştırma bulguları hakkında bilgiler verilmiştir. Bu bölümde yol ağlarının tespit edilmesini sağlayan eğitilmiş anlamsal bölütleme modellerinin üç farklı uygulamadaki sonuçları çeşitli değerlendirme metrikleri kullanılarak karşılaştırılmıştır. Ayrıca elde edilen sonuçlar görsel olarak değerlendirilmiştir.

Beşinci bölümde çalışmadan elde edilen sonuçlar literatürdeki benzer çalışmalar dikkate alınarak tartışılmış, sonuçların güçlü ve zayıf yönleri sebep-sonuç ilişkisi içerisinde açıklanmıştır.

Altıncı bölümde tez çalışmasından elde edilen olan sonuçlar tartışılmış ve gelecekte yapılması planlanan çalışmalar hakkında önerilerde bulunulmuştur.

1.1. Çalışmanın Amacı ve Önemi

OSM ve OAM gibi, topluluk tabanlı açık kaynak projeler benzer alanlarda faaliyet gösteren birçok ticari firma ile rekabet edebilecek miktarda ve kalitede güncel içerik üretmektedirler (Steinmetz vd., 2018). Üstelik bu içeriklere ulaşmak ve paylaşmak tamamen ücretsizdir. Diğer yandan genel olarak makine öğrenmesi, özel de ise derin öğrenme uygulamalarının en büyük sorunlarından birisi yeterli miktarda veri elde etmek ve bu verileri modelin kullanımına hazır hale getirme sürecindeki zorluklardır. Veri miktarı ve kalitesi makine öğrenme modelinin başarısını doğrudan etkileyen önemli faktörlerden birisidir. Yukarıdaki iki durum göz önüne alındığında topluluk tabanlı açık kaynak projeler tarafından sağlanan verileri makine öğrenmesi veya YSA algoritmalarında kullanmak çok sayıda fayda sağlayacaktır. Literatür incelendiğinde bu yönde yurt dışında bazı çalışmalar yapılmış olmasına rağmen ülkemizde bu alanda çok az çalışma yapılmıştır. Bu çalışmanın amacı topluluk tabanlı açık kaynak projelerden olan OSM ve OAM verilerini kullanarak mevcut derin öğrenme teknikleri ile yol ağlarının hızlı ve doğru bir şekilde tespit edilmesini sağlayan yöntemleri test etmek; mevcut modellere alternatif özelleştirilmiş bir anlamsal bölütleme modeli sunmaktır.

Çalışma birçok açıdan önemlidir. Çalışmanın akış şeması göz önünde bulundurularak çalışmanın önemi maddeler halinde şu şekilde ifade edilebilir.

- Topluluk tabanlı açık kaynak projelerin bazıları birçok ihtiyacı karşılayacak kalitede ve miktarda güncel veri tutmaktadır. Üstelik teknolojik gelişmelerin de katkısı ile veri miktarı ve kalitesi sürekli artmaktadır. Bu durumun farkında olunması başlı başına önemlidir.
- Değişik amaçlar veya ihtiyaçlar doğrultusunda ortaya çıkmış çok sayıda topluluk tabanlı proje mevcuttur. Bu projeler amaçlarına uygun olarak farklı platformlar üzerinde verilerini tutmaktadırlar. Üstelik birçok projenin verilerini depolamak için kullandığı standartlar veya protokoller birbirinden farklıdır. Bu çalışma, her biri platform, veri türü, veri standartları gibi birçok açıdan birbirinden oldukça farklı olan topluluk tabanlı projelerden elde edilen verilerin amaca uygun hale getirilmesinin ve birlikte kullanılmasının mümkün olduğunu gösteren değerli bir örnektir.
- Yeterli ve kaliteli veri miktarı makine öğrenmesi ve derin öğrenme algoritmalarının başarımını doğrudan etkileyen önemli bir faktördür. Topluluk

tabanlı açık kaynak projelerden elde edilen verileri gerekli ön işlemlerden geçirdikten sonra makine öğrenmesi ve YSA algoritmalarında kullanmak mümkündür.

- Makine öğrenmesi algoritmalarında önemli sorunlardan birisi sistemin kendisini sürekli olarak geliştirmesini sağlayan öğrenme sürecini otomatik hale getirmektir. Bu durumun önündeki en önemli engel sürekli veri üreten bir yapının oluşturulmasındaki zorluklardır. OSM ve OAM göz önüne alındığında bu projelere ait veri tabanlarına her gün dünya genelindeki çok sayıda farklı kullanıcı tarafından yüzbinlerce yeni içerik eklenmektedir. Dolayısıyla yapılan bu çalışma topluluk tabanlı projelerin makine öğrenme algoritmaları için sürekli bir veri kaynağı olabileceğini göstermektedir.

1.2. Çalışmanın Kapsamı ve Sınırlılıkları

Bu çalışmada geliştirilen yöntem ile dünyanın herhangi bir bölgesindeki OSM ve OAM verilerini kullanarak yol ağı tespitini sağlayan bir YSA modelin eğitilmesini sağlamak mümkündür. Bununla birlikte bu çalışmada yeterli miktarda veri içermesi ve ayrıca mekânsal olarak dünyanın farklı bölgelerinde yer alması göz önünde bulundurularak aşağıdaki 'de verilen çalışma alanlarında uygulama yapılmıştır.

Tablo 1.1. Çalışma alanları, sınırlayıcı kutu değerleri (BBOX) ve enlem, boylam bilgileri

Bölge (Şehir)	BBOX (min. enlem, min. boylam, mak. enlem, mak. boylam)
Tübingen (Almanya)	8.7664,48.3129,9.4888,48.6075
Minsk (Belarus)	27.4428,53.8426,27.6643,53.9443
Saint Petersburg Rusya)	30.0188,59.6269,31.0831,60.0168
Stara Zagora (Bulgaristan)	25.0928,42.1451,26.3617,42.7410

Stara Zagora bölgesinden elde edilen veriler sadece modelleri değerlendirmek amacıyla kullanıldığı için veri setine dahil edilmemiştir.

1.3. Literatür Özeti

İnsansız hava araçlarından alınan görüntüleri kullanarak farklı çalışmalar yapmak mümkündür. Sınıflandırma ve nesne tanıma ön plana çıkan iki temel kategoridir (Rezaee ve Zhang, 2017). Çok sayıda kentsel objenin sayısal hava ve uydu görüntülerinden otomatik olarak çıkarılması, altyapı planlama, coğrafi veri tabanlarının güncellenmesi, arazi kullanım analizi ve değişiklik tespiti gibi birçok uygulamada önemli bir araçtır.

Kentsel alanlara ait sayısal hava ve uydu görüntülerindeki piksellerin anlamsal nesneler olacak şekilde sınıflandırılması en zor ve en önemli sorunlardan birisidir. Bunun temel iki nedeni vardır. Birincisi uzaktan algılama görüntüleri genellikle heterojen bölgeler şeklindeki karmaşık verilerle karakterize edilir. Bu bölgeler büyük sınıf içi değişimler ve genellikle daha düşük sınıflar arası değişimlere sahiptir. Bu, binalar ve yollar gibi kentsel nesnelerde daha da belirgindir. Uzaktan algılama görüntülerinde bu tür heterojenlik, genellikle ayarlanabilir parametreler kullanılarak çıkarılan önceden tanımlanmış özellikler kümesine bağlı mevcut yöntemlerin çoğunu kısıtlar. Sonuç olarak, özellikle artan mekânsal çözünürlüklerle yüksek doğruluk elde edebilen bir yöntem tasarlamak son derece zordur (Alshehhi vd., 2017, s. 139). Diğer bir neden ise etiketli veri miktarının, veri setinin toplam büyüklüğüne kıyasla oldukça küçük olmasıdır (Sameen vd., 2018). Yollar gibi kentsel alanlardaki hızlı değişimler nedeniyle, mevcut kentsel haritaların her yıl güncellenmesi gerekmektedir. Makine öğrenmesinin bir alt alanı olan derin ağlar, otomatik nesne algılamada önemli başarımlar göstermiştir (Rezaee ve Zhang, 2017).

Özellikle Web 2.0 ile birlikte topluluk tabanlı birçok açık kaynak proje ortaya çıkmıştır. Bu projeler genellikle bir amaç için bir araya gelen gönüllüler tarafından yürütülmektedir. Bu projeler kendi amaçları doğrultusunda çok sayıda güncel ve detaylı verileri üretmelerinin yanında aynı zamanda bu verilerin başka projelerde veya platformlarda kullanılmasını sağlayan çeşitli servisler, platformlar ve iletişim protokolleri meydana getirmektedirler. Ayrıca topluluk tabanlı projelerin açık kaynak olması, bu projelere ait ham verilere ulaşip bu verileri farklı projelerde ihtiyaca göre düzenleyerek kullanmayı mümkün kılmaktadır. Bu kapsamda, geçmişten günümüze yol tespiti ile ilgili yapılan ve ön plana çıkan çalışmalara ait literatür çalışması aşağıda özetlenmiştir. Yapılan çalışmaları daha anlaşılabilir yapıda sunmak amacıyla yol tespit yöntemleri sınıflandırılarak her bir sınıf için yapılan çalışmalar ilgili bölüm altında sunulmuştur.

Birçok araştırmacı yol tespit yöntemlerini farklı kriterlere göre sınıflandırmış olsa da herkesin kabul ettiği, standart haline gelen bir sınıflandırma yöntemi söz konusu değildir (W. Wang vd., 2016, s. 271). Bu çalışma kapsamında literatürde en çok kullanılan ve günümüzdeki gelişmeleri de kapsayan bir sınıflandırma yöntemi yapısı kullanılmıştır.

1.3.1. Sınıflandırmaya dayalı yöntemler

Sınıflandırmaya dayalı yöntemler genellikle bir yolun geometrik, spektral ve doku özelliklerini kullanırlar. Sınıflandırma doğruluğu, kaldırım taşları, alan blokları, su

alanları ve otoparklar gibi diğer bazı nesnelerin spektral olarak yolu oluşturan nesnelere spektral olarak benzemesinden dolayı tatmin edici değildir. Etiketli eğitim verilerinin kullanımına göre, sınıflandırmaya dayalı yöntemler denetimli ve denetimsiz olarak ikiye ayrılabilir (W. Wang vd., 2016). Yarı-denetimli sınıflandırma yöntemi ise çok az kullanıldığından bu yapıya dahil edilmemiştir.

1.3.1.1. Denetimli sınıflandırma yöntemleri

Denetimli sınıflandırma yönteminde etiketli veri seti kullanılmaktadır. Denetimli sınıflandırma yöntemlerinin doğruluğu büyük ölçüde seçilen özelliklere, etiketli verilerin miktarına ve kalitesine bağlıdır. Genel olarak denetimli sınıflandırma yöntemleri YSA (yapay sinir ağları), destek vektör makineleri (SVM), Markov rastgele alanlar sınıflandırıcısı (MRFs) ve maksimum olabilirlik (ML) şeklindedir.

YSA sınıflandırma yöntemleri

YSA, biyolojik sinir ağlarının düşük seviyede taklit edilmesi ile ortaya çıkan ve aynı zamanda birbirine bağlı olan düğümlerden (veya nöronlardan) oluşan bir hesaplama modelidir.

Heermann ve Khazenie (1992), tarafından önerilen geri yayılım (back propagation) algoritması ile YSA'ya dayalı yol tespit yöntemlerinin daha hızlı bir şekilde geliştirilmesi konusunda önemli bir katkı sağlamıştır. Bu yöntemle dayanan ilk yol tespit çalışmalarında sınıflandırma yapmak için genellikle görüntü piksellerinin spektral ve bağlamsal özelliklerini kullanan modeller geliştirilmiştir. Tu-Ko (2003), yol tespiti için bir YSA'yı spektral ve kenar bilgisini kullanarak eğitmiş ve sonuçta yoldaki merkez hattının tanımlanmasını sağlayan başarılı bir yaklaşım önermiştir. Elde edilen sonuçlar yol olmayan birçok bölüt içerse de genel olarak yöntem başarılıdır. Mokhtarzade ve Zoej (2007) YSA modelini kullanarak yüksek çözünürlüklü uydu görüntülerinden yol tespiti yapmışlardır. Bu çalışmada sisteme farklı parametrelerin girilmesiyle, optimum giriş vektörü elde edilmiş ve yineleme süresi ile çeşitli ağ yapıları test edilebilmiştir. Ağın eğitimi ve geri çağırılması için girdi vektöründe en son bilgileri ve doku parametrelerini kullanmak, ağın yol tespiti konusundaki becerisini geliştirmek ve ileri çalışmalar için iyi bir öneri olabileceği ifade edilmiştir. Elde edilen sonuçlar maksimum olabilirlik sınıflandırıcısı ile karşılaştırılmıştır.

Geri yayılım YSA modeli kullanılarak yol tespiti konusunda bir başka çalışma ise (Kirthika ve Mookambiga, 2011) tarafından yapılmıştır. Araştırmacılara göre insan görme sisteminde spektral bilgilerin yanı sıra, dokusal parametreler ve bağlamsal bilgiler kullanılmaktadır. Dolayısıyla geliştirilen YSA modeline dokusal ve bağlamsal bilgilerin verilmesi modelin başarısını artıracaktır. Çalışmada ilk başta, yol tespiti sadece spektral bilgiler kullanılarak yapılmıştır. Daha sonra her piksel için kontrast, enerji, entropi ve homojenlik gibi farklı doku parametreleri hesaplanmış ve önceden sınıflandırılmış bir yol raster haritası üretilmiştir. YSA'nın işlevselliğini optimize etmek ve doku tespitine katkıda bulunan yol parametrelerinin yol tespiti üzerindeki etkisini değerlendirmek için elde edilen doku parametreleri spektral bilgi ile birleştirilmiştir.

Yapılan bu çalışmalar yol tespit uygulamalarında YSA kullanmanın bazı dezavantajları olduğunu göstermiştir. Yakınsama hızının yavaş olması, daha fazla eğitim setine ihtiyaç duyması, yerel minimuma girme olasılığı, artan kategori sayısının modelin hızını düşürmesi ve modelde kolaylıkla aşırı öğrenmenin oluşması ön plana çıkan bazı temel dezavantajlardır. Bu nedenle, uydu görüntülerinden yol tespiti için radyal temelli sinir ağı, bulanık sinir ağı ve bulanık sinir ağı gibi birçok yeni veya geliştirilmiş sinir ağı modeli kullanılmıştır (George vd., 2013; J. Li ve Chen, 2014).

Destek vektör makineleri sınıflandırma yöntemleri

Destek vektör makinesi (Boser vd., 1992) tarafından sunulan bir sınıflandırma yöntemidir. SVM sınıflandırıcısı biyoinformatikte (ve diğer disiplinlerde) yüksek boyutlu verilerle başa çıkma kabiliyeti ve çeşitli veri kaynaklarını modelleme esnekliği nedeniyle yaygın olarak kullanılmaktadır (Ben-Hur ve Weston, 2010). SVM yol tespiti çalışmalarında da kullanılmaktadır. Aşağıda bu konuda yapılan bazı temel çalışmalar verilmiştir.

Yager ve Sowmya (2003) tarafından yapılan çalışmada uzaktan algılama görüntüleri kullanılarak SVM sınıflandırıcısı ile yol tespiti yapılmaya çalışılmıştır. Çalışmada gradient, yoğunluk, kenar uzunluğu gibi kenara dayalı özellikler SVM sınıflandırıcısı için kullanılmıştır. Ancak birçok araştırmacı elde edilen sonuçların nispeten düşük doğruluk değerlerine sahip olduğunu ifade etmiştir (W. Wang vd., 2016). Melgani ve Bruzzone (2004), yaptıkları çalışmada SVM sınıflandırıcısı ile hiperspektral uzaktan algılama görüntülerini sınıflandırmışlardır. Birçok durumda, SVM'nin

sınıflandırma yöntemleri, radyal temelli sinir ağının ve K-en yakın komşu sınıflayıcısının doğruluğundan, kararlılığından ve sağlamlığından daha iyidir (W. Wang vd., 2016).

Simler (2011), çok sınıflı SVM yöntemiyle yoğun kentsel alanlarda yol ve bina tespiti yapan bir yöntem önermiştir. Araştırmacı yoğun kentsel alanların VHR multispektral hava görüntüleri üzerinde bina ve yol tespiti için bir yöntem önermiştir. Mevcut tüm bilgileri kullanmak için, bölümlendirilmiş alanların hem mekânsal hem de spektral özellikleri, üç sınıflı bir SVM kullanılarak sınıflandırılmıştır. Yöntemin, geometrinin daha iyi kullanılmasını, kötü algılamayı ve yanlış alarm sorunlarını hafifleterek sınıf ayrılabilirliğini artırdığı ifade edilmiştir. Son olarak, yol ağını düzenlemek ve sınıflandırma doğruluğunu iyileştirmek için bir sınıflandırma sonrası algoritması önerilmiştir. SVM yöntemleri, uydu görüntülerinden nesne tespitinde yaygın olarak kullanılan yapısal risk minimizasyonu ve iyi genelleme yeteneğinin avantajlarına sahiptir. Bununla birlikte, çekirdek işlevlerinin tahmini, boyutsal alan seçimi ve eğitim örnekleri gibi SVM yöntemlerini kullanmakta güçlükler vardır.

Markov rastgele alanlar (markov random field) sınıflandırma yöntemi

Markov rastgele alanlar, olasılık teorisinin bir dalına aittir ve fiziksel olayların korelasyonunu iyi açıklayabilir. Bir görüntüdeki piksel etkileşimi mekânsal bir korelasyona sahiptir, böylece görüntü koşullu olasılık dağılım fonksiyonu tarafından doku istatistiksel özelliklerini tarif ederek Markov rastgele alanlar sınıflayıcısı tarafından etkili bir şekilde analiz edilebilir. Bu yöntem, kenar tespiti, görüntü bölütleme (segmentasyon), restorasyon ve yeniden oluşturma (reconstruction) vb. çalışmalar için yaygın olarak uygulanabilmektedir (Yousif ve Ban, 2014).

Tupin vd., (1998) bir uydu görüntüsünden doğrusal bir yol çıkarmak için iki aşamalı bir algoritma kullandılar. İlk olarak, görüntülerde yol olma olasılığı olan bölütler yerel bölgesinde doğrusal bir algılayıcı ile tespit edildi. İkinci adımda ise, yol nesnelerinin şekli hakkında bağlamsal bilgi veren bir dizi bölüt üzerinde bir Markov rastgele alanı tanımlanarak bölüt adayları arasındaki gerçek yollar belirlenmiştir.

Wang ve Luo (2005), Markov rasgele doku modeline ve SVM sınıflandırıcısına dayanan bir yol ağı çıkarma yöntemi önermiştir. Yöntem eğitim için doku özelliklerini kullanır, ancak insan müdahalesine ihtiyaç duymaktadır.

Zhu vd., (2011) Markov rasgele doku modeli ve SVM ile bulanık C-ortalamlar (FCM) yönteminin birlikte kullanılması ile ortaya çıkan hibrit modeli kullanarak yol tespiti yapan bir yöntem geliştirmişlerdir. Çalışmada öncelikle, FCM kullanılarak uzaktan algılama görüntüsü için denetimsiz kümeleme yapılmış; ardından SVM daha fazla sınıflandırma için kullanılmış ve yollar çıkarılmıştır. Son olarak, önerilen iki algoritma ile karşılaştırma gerçekleştirilmiş ve elde edilen sonuçlardan sonra SVM ve FCM modellerinin birlikte kullanımı ile ortaya çıkan hibrit yöntemin Markov rasgele alanlar yönteminden çok daha doğru sonuçlar verdiği görülmüştür (D.-M. Zhu vd., 2011).

Yukarıdaki çalışmalardan elde edilen sonuçlar göz önüne alındığında, nesne tespiti uygulamalarında sadece MRF modeli kullanılarak bir uzaktan algılama görüntüsündeki nesneler arasındaki karmaşık ilişkinin doğru bir şekilde modellenmesinin zor olduğu görülebilir. Bu nedenle, hibrit olasılık modelinin uzaktan algılama görüntüsündeki karmaşık nesne tespitinde iyi sonuçlar elde etmesi beklenmektedir.

Maksimum olabilirlik (maximum likelihood) sınıflandırma yöntemleri

Maksimum olabilirlik (ML) sınıflandırması, yani Bayes sınıflandırması, en çok kullanılan denetimli sınıflandırma yöntemlerinden biridir. Bu yöntem, Gauss olasılık yoğunluğu fonksiyonu kullanarak her piksel için özellik olasılığını hesaplar ve daha sonra pikselleri maksimum olasılık kategorilerine göre sınıflandırır (W. Wang vd., 2016). Zhou vd., (2006) insan-bilgisayar etkileşimlerine ve Bayesian filtrelemeye dayalı bir yol izleme sistemi önermişlerdir. Bayesian filtreleri, özellikle, genişletilmiş Kalman filtreleri ve parçacık filtreleri, yol eksenini noktalarını tahmin etmek ve izleme algoritmalarını güncellemek için insan girdileriyle birlikte kullanılmıştır. Araştırmacılara göre deneysel sonuçlar bu yaklaşımın verimli ve güvenilir olduğunu ve geleneksel manuel harita revizyon yaklaşımına göre önemli tasarruflar sağladığını göstermektedir.

Son zamanlarda, sınıflandırma doğruluğunu arttırmak için farklı sınıflandırıcıların kombinasyonu önemli bir araştırma konusu haline geldi. Bu çalışmaların temeli, aynı veri setini farklı sınıflandırma modelleri kullanarak eğitmek ve ardından bu sınıflandırma sonuçlarını birleştirme fikrine dayanmaktadır (Z. Chen, 2006).

Derin öğrenme yöntemleri

Derin öğrenme makine öğrenmesinin bir alt türüdür. Derin öğrenme hem denetimli hem de denetimsiz öğrenme algoritmalarını desteklemektedir. Derin öğrenme özellikle 2010 yılından itibaren aktif olarak kullanılmaya başlanmıştır. Derin öğrenme ile yol tespiti konusunda yapılan bazı temel çalışmalar aşağıda verilmiştir.

Mnih and Hinton (2010), daha önce yolları tespit etmek amacıyla yapılan çalışmalardan farklı olarak çok fazla katmana sahip bir YSA kullanarak yol ağlarını tespit etmeye çalışmışlardır. Oluşturulan YSA, GPU (graphics processing unit) tabanlı bir sistem üzerinde büyük miktarda veri kullanılarak eğitilmiştir. Önerilen yöntemin önceki yaklaşımları değerlendirmek için kullanılan daha büyük olan iki adet kentsel veri seti kümesi üzerinde güvenli bir şekilde çalıştığı ve kabul edilebilir sonuçlar verdiği görülmüştür.

Wang vd., (2015) tarafından yapılan çalışmada, yol ağlarını tespit etmek için derin evrimsel sinir ağları ve sonlu durum makinesi (FSM) yöntemlerini temel alan bir nöral-dinamik izleme çerçevesi önerilmiştir. Otonom mobil sistemlerden esinlenen yazarlar, izleyicinin geçerli konumunda ortalanmış bir algılama penceresinde çıkarılan bir görüntü yaması olan giriş verilerinin desenini tanımak için bir derin sinir ağı eğitmişlerdir. Desen, ortama göre önceden tanımlanmıştır ve FSM'deki durumlarla ilişkilendirilmiştir. Görüntülerden ve mevcut vektör yol haritalarından büyük görüntü yönü çiftlerini ayıklayan derin sinir ağının ihtiyaç duyduğu eğitim verilerini oluşturmak için vektör güdümlü bir örnekleme yöntemi önerilmektedir. Yazarlara göre sonuçlar, önerilen yöntemin geleneksel yöntemlerden daha doğru ve verimli olduğunu göstermektedir.

Zhang vd., (2016) denetimli derin evrimsel sinir ağlarını kullanarak yollardaki çatlakları ve bozulmaları tespit etmek için bir yöntem önermişlerdir. Derin evrimsel sinir ağı, toplanan görüntülerdeki her görüntü bölümünü sınıflandırmak için eğitilmiştir. Yazarlara göre, düşük maliyetli bir akıllı telefon ile elde edilen, 3264 x 2448 boyutundaki 500 görüntüden oluşan bir veri kümesi üzerinde yapılan nicel değerlendirmeye göre, önerilen derin öğrenme modeliyle öğrenilen özellikler, mevcut geleneksel yöntemler ile elde edilen özelliklerle karşılaştırıldığında üstün çatlak algılama performansı gösterdiği görülmektedir.

Lu vd., (2018) uzaktan algılama görüntüleri ile birlikte yerel ve küresel bilgileri kullanarak yol ağlarını etkin bir şekilde çıkarabilen yeni bir yol tespit yöntemi önermişlerdir. Yöntem uzaktan algılama görüntülerini öncelikle ön işlemden geçirmektedir. Tasarlanan modeli eğitmekte kullanılacak olan veri setini hazırlamak için, tüm uzaktan algılama görüntüleri ve buna karşılık gelen gerçek yer verileri (ground truth images) sabit boyutlu bir sürgülü pencere ile kırpılmıştır. Hazırlanan veri setleri ile model eğitilmiş ve çıkış verileri iki kategori olarak sınıflandırılmıştır. Yöntemi doğrulamak için Google Earth'ten bir veri kümesi kullanılmış ve yapılan uygulamalar, önerilen derin evrimsel sinir ağının yol ağını doğru ve etkili bir şekilde çıkarabileceğini göstermiştir.

Yol tespiti için derin öğrenmenin başka yöntemler ile birlikte kullanıldığı çalışmalar da vardır. Geng vd., (2018) karmaşık ortamlarda yol algılama yaklaşımlarının doğruluğunu ve sağlamlığını artırmak için, evrimsel bir sinir ağına ve Markov rasgele alanına dayalı yeni bir yol algılama yöntemi önermektedirler. Çalışmada orijinal yol görüntüsü, basit doğrusal yinelemeli kümeleme (SLIC) algoritması kullanılarak tek boyutlu büyüklükte süper piksellere bölünmektedir. Bu temelde, sınıflandırmaya en faydalı özellikleri otomatik olarak öğrenebilen evrimsel sinir ağı eğitilmektedir. Evrimsel sinir ağı daha sonra yol ve yol dışı bölgeleri sınıflandırmak için kullanılmaktadır. Son olarak, süper piksel komşuluk arasındaki ilişkiye dayanarak, evrimsel sinir ağının sınıflandırma sonuçlarını optimize etmek için Markov rasgele alanlar yönetimi kullanılmaktadır.

Wu vd., (2019) yüksek kaliteli piksel boyutunda yol genişliği etiketi yerine, eğitim verisi olarak sadece OSM yol merkez hattını kullanarak çok yüksek çözünürlüklü görüntülerinden yol ağlarını çıkarmak için zayıf bir şekilde denetlenen yeni bir yöntem önermektedirler. Yöntemi doğrulamak için büyük miktarda eşleştirilmiş Google Earth görüntüsü ve OSM verileri kullanılmıştır. Sonuçlar, önerilen yöntemin çok yüksek çözünürlüklü görüntülerden yol ağlarını piksel bazında yol eğitimi verileri kullanmadan hem doğru hem de etkili bir şekilde çıkarabildiğini göstermektedir.

Literatür incelendiğinde, OSM veya OAM verilerini farklı teknikler veya modeller ile kullanarak bazı yol tespit çalışmalarının yapıldığı görülmektedir. Li vd., (Q. Li vd., 2014) tarafından yapılan çalışmada, OSM veri kümesindeki verileri kullanarak çok şeritli yolların tespiti için yeni bir yaklaşım önerilmiştir. Yolları tespit etmek için yollarla çevrelenen çokgenleri işlenecek veri olarak ele alan, şekil analizine ve Gestalt

teorisine dayanan çokgen tabanlı bir yöntem önerilmiştir. Çalışmada ilk olarak, şekil tanımlayıcıları, ağılardaki her çokgen için hesaplanmakta ve sınıflandırma için kullanılmaktadır. Daha sonra, aday çok şeritli çokgenler, bir SVM tarafından şekil tanımlayıcıları olarak kullanılan tüm çokgenlere dayalı kaynak olarak sınıflandırılmaktadır. Son olarak, kaynak poligonlarına dayanarak, Gestalt teorisine göre çok katmanlı özellikleri birbirine bağlayan ve dolduran bir bölge oluşturma yöntemi önerilmiştir. Farklı kentsel ağlardan OSM verilerini kullanan bir çalışma, önerilen yöntemin geçerliliğini doğrulamıştır. Yöntem, veri kümelerinin karmaşıklığı ve çoğaltılması ne olursa olsun, iyi ve etkili yol tespit performansı göstermiştir.

1.3.1.2. Denetimsiz sınıflandırma yöntemleri

Denetimsiz öğrenme, etiketlenmiş veri setine ihtiyaç duymadan girdi olarak verilen veri kümesinden çıkarımlar elde etmek için kullanılan özerk bir tür makine öğrenme algoritmasıdır. Aynı zamanda öz-örgütlenme olarak bilinir ve verilen girdilerin olasılık yoğunluklarının modellenmesine izin verir. Denetimsiz öğrenme yönteminin sınıflandırma problemlerinin çözümünde çok sayıda avantajı vardır. Buna rağmen genel olarak denetimsiz öğrenme algoritmalarının başarısı denetimli öğrenme algoritmalarına göre daha düşüktür. Denetimsiz sınıflandırma yöntemleri genellikle bilgi bulma, parametre belirleme, karakteristik analiz ve diğer ön işleme adımlarında kullanılır. En yaygın algoritmalar, K-ortalamlar, spektral kümeleme, ortalama kayma ve graf teorisi gibi kümeleme algoritmalarıdır (Ghahramani, 2004; Koenig ve Simmons, 1996).

Ağırlıklı ortalama öteleme algoritması (mean shift)

Ağırlıklı ortalama öteleme analizi, karmaşık özellik alanlarının analizi için yoğunluk tahminine dayanan, genel parametrik olmayan bir kümeleme tekniğidir. Veri dağılımının türü, veri sayısı ve verilerin başlangıç noktası göz ardı edilebildiği için bu teknik özellikle örüntü tanıma, görüntü iyileştirme ve görüntü bölümlendirme vb. alanlarda yaygın olarak kullanılmaktadır.

Miao vd., (2014) yüksek çözünürlüklü uydu görüntülerinden yol ağlarını tespit etmek için yarı otomatik bir yöntem önermişlerdir. Yöntem üç ana adımdan oluşmaktadır. İlk olarak, kullanıcılar tarafından önceden öngörülen yol kaynak noktalarını bağlamak için ihtiyaç duyulan ilk yol bölütlerini tespit etmede jeodezik yöntem kullanılmıştır. Daha sonra, bu kaba yol bölümlerine dayanarak bir yol olasılık haritası üretilir ve bir eşikleme işlemi görüntüyü yol ve yol olmayan iki yüzey sınıfına ayırmaktadır. Sonraki adımda yol

sınıfı görüntüsünü kullanarak, yukarıda belirtilen yol kaynak noktalarını bağlamak için jeodezik yöntemin bir kez daha kullanıldığı bir çekirdek yoğunluğu tahmin haritası oluşturulmaktadır. Deneyler, önerilen yöntemin düzgün ve doğru yol merkez çizgileri çıkarabildiğini göstermektedir. Ağırlıklı ortalama kaydırma algoritması önceden herhangi bir bilgi gerektirmez ve özellikle uzaktan algılama görüntülerinden nesne çıkarımı için uygun olan yüksek verimlilik ve kararlılığa sahiptir. Bununla birlikte, bu alandaki mevcut araştırma çalışmaları nispeten sınırlıdır (W. Wang vd., 2016).

Graf yöntemi

Grafik teorisine dayalı kümeleme yöntemi, görüntü bölütlemeye nispeten yeni bir araştırma konusudur. Graf teorisi, köşegen ve kenar teorisini incelemek için bir araştırma nesnesi olarak grafi temel alan ayırık bir matematik dalıdır (W. Wang vd., 2016). Görüntü öğelerini matematiksel olarak sağlam yapılara göre düzenler ve sorunun formülasyonunu daha esnek ve hesaplamayı daha verimli hale getirir (Peng vd., 2013) . İlke şu şekilde tarif edilir: görüntü, tepe noktanın bir piksele veya bir bölgeye karşılık geldiği ağırlıklı bir grafik olarak görülür. Kenar ağırlığı, farklı tepe noktasının farkı veya benzerliği ile hesaplanır. Sonuç matris özvektörleri ve özdeğerler kullanılarak elde edilir (W. Wang vd., 2016).

Literatürde birçok etkileşimli bölütleme tekniği önerilmesine rağmen son yıllarda en popüler yöntem, (Boykov ve Jolly, 2000) tarafından önerilen graf kesimleri yöntemidir. Bu yöntemi kullanan farklı araştırmacılar vardır. Tao ve Jin (2007) düşük hesaplama maliyetiyle geliştirilmiş görüntü bölütleme yöntemi elde etmek için yeni bir eşik algoritması sunmuşlardır. Önerilen algoritma, bir nesneyi arka plandan ayırmak için eşikleme prensibi olarak normalleştirilmiş graf kesim ölçüsünü kullanmaktadır. Graf kesimlerinin değerlendirilmesinde kullanılan ağırlık matrisleri, yaygın olarak kullanılan görüntü piksellerinden ziyade görüntünün gri seviyelerine dayanmaktadır. Çoğu görüntü için, gri düzeylerinin sayısı piksel sayısından çok daha küçüktür. Önerilen eşikleme algoritmasının, mevcut eşikleme algoritmalarına kıyasla üstün performansını göstermek için çok sayıda örnek sunulmuştur. Zheng vd., (2012) grafik teorisine dayanan etkileşimli bir görüntü bölütleme yöntemi önermiştir. Bu algoritma, tek nesneyi karmaşık ön plan ve arka plana sahip görüntülerden ayırırken daha az kullanıcı etkileşimli olması itibarıyla daha iyi sonuçlar üretmiştir. Çalışma kapsamında yapılan uygulamalar bu algoritmanın,

graf kesimdeki ortak aşırı bölütlemeyi ve daha az bölütleme işlemlerini etkili bir şekilde çözmenin yanı sıra, bölütlenmiş küçük bölgeler hatası sorununu da çözmüştür.

Cem ve Beril (2012), olasılık ve graf teorisine dayanan bir yol çıkarma yaklaşımı kullanmıştır. Önerilen yöntem olasılıklı yol merkezi tespiti, yol şekli çıkarma ve graf teorisi temelli yol ağı oluşumu olmak üzere üç modülden oluşmaktadır. Bu modüller, eldeki uygulamaya bağlı olarak arka arkaya veya değiştirilebilir şekilde kullanılabilir. Yöntemin avantajlarını ve dezavantajlarını göstermek için Geoeye, Ikonos ve QuickBird gibi farklı platformlardan elde edilen görüntüler kullanılarak birçok uygulama yapılmıştır. Elde edilen sonuçlar, sistemin uzaktan algılama görüntülerindeki yol ağını güvenilir ve hızlı bir şekilde tespit etmekte kullanılabileceğini göstermiştir (Unsalan ve Sirmacek, 2012). Graf teorisine dayanan görüntü bölütleme yöntemi, küresel özellikleri yakalama ve karmaşık veri türleri ile ilgilenme avantajlarına sahiptir. Bununla birlikte, çoğunlukla matris spektrumu kavramını benimser ve görüntü bölütleme yapmak için bir benzerlik matrisinin özdeğerini ve özvektörünü kullanır. Ayrıca, işleme prosedürü tepe sayısının artmasıyla karmaşık hale gelir. Bu nedenle, yöntem uzaktan algılama görüntüleri bölütlenmesi uygulamalarında yaygın olarak kullanılmamıştır (W. Wang vd., 2016).

1.3.1.3. Bilgi tabanlı yöntemler

Uzaktan algılama görüntülerinden yol çıkarmak yalnızca yerel spektrum ve doku bilgilerini kullanarak zordur. Özvektörü tanımlamayı zorlaştıran yolun şerit yapısı nedeniyle, veriler doğrudan sınıflandırıcıya girilememektedir. Bilgi temelli sınıflandırmada, yardımcı veriler ve diğer bilgiler spektral bilgi ile birleştirilir. Literatürde bu yöntemi kullanarak yol tespit uygulamaları yapan bazı çalışmalar vardır. Wang ve Zheng (1998), genel nesneleri çıkarmak için geometrik özellikleri kullanarak SAR görüntülerinden yol ve köprü tespiti için bir yöntem önermiştir. Daha sonra küçük bölgeleri çıkarmak ve süresiz kesimleri bağlamak için matematiksel morfoloji ve Hough dönüşümü kullanılmıştır.

Shen vd., (2008) açısız doku imzasına dayanan bir yol tespit yöntemi sunmuşlardır. Sunulan yöntem yalnızca gri seviyeli bir görüntüdeki uzun şeritli yolları izleyebilmekteydi. Algoritma karmaşık sahnelerdeki gölgelerden ve tıkanıklıklardan önemli ölçüde etkilendiği için yol tespiti konusunda sorun yaşamaktaydı. Bu konuda

başka araştırmacılar tarafından yapılan çalışmalar mevcuttur. Bu çalışmalar daha çok 2000'li yıllarda yapılmıştır (Lee vd., 2000; Tönjes ve Growe, 1998).

Yukarıda açıklanan bilgiye dayalı yöntemler, uzaktan algılama görüntülerinden yol tespiti konusunda ön plana çıkan bazı uygulamalardır. Buna rağmen bu yöntem fazla hassas olma ve gölgelere duyarlı olma gibi birtakım dezavantajlara sahiptir.

1.3.1.4. Matematiksel morfoloji yöntemleri

Matematiksel morfoloji, resim işleme, örüntü tanıma, bilgisayarla görü ve diğer bazı alanlarda yaygın olarak kullanılmaktadır. 1980'li yıllarda araştırmacılar, yol çıkarımında matematiksel morfolojiyi kullanmaya başlamışlardır. Daha sonra matematiksel morfolojiye dayanan başka yöntemler de önerilmiştir. Bununla birlikte, matematiksel morfoloji yöntemleri genellikle görüntü bölütlemesi için diğer yöntemlerle birleştirilmektedir (Valero vd., 2010; C. Zhu vd., 2005).

Zhang vd., (1999) matematiksel morfoloji yöntemlerini kullanarak otomatik yol tespiti yapan bir yöntem geliştirmişlerdir. Görüntü çözünürlüğü arttıkça, yol ağları ince çizgiler yerine belirli genişlikte alanlar olarak görünmektedir. Çalışmada, öncelikle yol ağı bölgelerini bulmak için görüntü sınıflandırılmış daha sonra yol yüzeyleri ile benzer spektral özelliklere sahip nesneler de dahil olmak üzere gürültüyü önlemek için morfolojik işlemler uygulanmıştır. Geliştirilen yöntem, yüksek çözünürlüklü simülasyon görüntüleri ve havadan fotoğraflar üzerinde test edilmiştir. Sonuç, matematiksel morfolojinin otomatik karayolu ağı tespiti için etkili bir araç sağladığını göstermektedir. Çalışmanın en önemli eksikliği ise bazı yol boşluklarının tam kapatılamamasıdır.

Zhu vd., (2005) esas olarak gri seviyeli matematiksel morfolojiye dayanan yol ağlarını tespit etmek için bir yöntem kullandılar. Yöntem, özellikle kötü hava koşulları veya diğer faktörler nedeniyle görüntülerin belirsiz veya bulanık olduğu durumlarda daha iyi sonuçlar vermiştir.

Ma vd., (2012) belli belirsiz hava görüntüleri için otomatik bir yol çıkarma yöntemi önermişlerdir. Görüntü kalitesi, kamera titreşimi, hava durumu ve ışık değişimi gibi farklı faktörlerden etkilenebilmekte; bu da görüntü bölütlemesini ve yol tespitini zorlaştırmaktadır. Bu yöntemde ilk olarak, yüksek çözünürlüklü ancak düşük kontrastlı bir görüntü, Retinex tabanlı bir algoritma kullanılarak iyileştirilmektedir. Görüntü Canny kenar algılama algoritmasıyla bölümlere ayrılmaktadır. Daha sonra, doğrusal ve kavisli

yol bölümleri Hough çizgi dönüşümü ile düzenlenmekte ve çeşitli yol boyut ve şekil eşikleri dikkate alınarak çıkartılmaktadır. Elde edilen sonuçlar Otsu eşiği, graf tabanlı algoritma, birleştirme ve bölme gibi diğer yöntemlerle karşılaştırılmıştır. Sonuçlar, bu yöntemin uygulanabilir ve etkili olduğunu gösterilmiştir.

Maurya vd., (2011) tarafından yapılan çalışmada K-ortalamlar ve morfolojik işlemler birlikte kullanılarak yol tespiti yapan bir çalışma önerilmiştir. Pankaj vd., (2013) uyarlamalı global eşik ve morfolojik işlemleri kullanarak yüksek çözünürlüklü uydu görüntülerinden otomatik yol tespiti yapan bir çalışma yapmışlardır. Çalışma kapsamında görüntü bölütleme yöntemi kullanılmıştır. Elde edilen deneysel sonuçlar, önerilen yaklaşımın yüksek çözünürlüklü uydu görüntülerinden yolların otomatik tespiti için güvenilir bir şekilde kullanılabileceğini kanıtlamıştır (Singh ve Garg, 2013).

Matematiksel morfoloji yöntemleri, uzaktan algılama görüntülerinden yol tespiti yapılması aşamasında bazı belirli avantajlara sahiptir. Uygulamada, görüntü bölütleme sonuçları, yapı elemanlarının seçiminden (şekil ve boyut) büyük ölçüde etkilenmektedir. Yapı elemanlarının belirsizliği nedeniyle, yüksek doğruluk ve iyi çıkarım sonuçları elde etmek için yalnızca matematiksel morfoloji yöntemi kullanmak yetersizdir.

1.3.1.5. Aktif kontur modeli

Son yıllarda hava ve uydu görüntülerindeki çizgisel nesnelerin tespiti ile ilgili otomatik veya yarı otomatik bazı çalışmalar yapılmaktadır. Aktif kontur modeller yöntemi en yaygın kullanılan yarı otomatik detay çizim yaklaşımlarından birisidir. Bu konsept bir optimizasyon işleminden geçirilen eğri belirleme algoritmalarına dayanmaktadır (Şeker D. Z. ve Eker, 2005).

Aktif kontur modelinin en önemli avantajlarından birisi fazla olan parametre sayısını düşürme yeteneğine sahip olmasıdır. Bu yöntem özellikle çizgisel detayların tespit edilmesinde yaygın olarak kullanılmaktadır. Yol kenar ve merkez hatlarının tespit edilmesinde aktif kontur tabanlı yöntemler kullanılabilmektedir.

Snake yöntemi

Snake modeli ilk olarak Kass vd., (1988) tarafından ortaya atılmıştır. Bundan sonra, aktif kontur modeline dayanan birçok görüntü bölütleme, anlama ve tanıma yöntemleri yaygın olarak kullanılmıştır. Snake modelinin ana fikri şu şekilde açıklanabilir: Bir şablon olarak belirli bir şekil oluşturmak için bazı kontrol noktalarını

alır ve şablondaki elastik deformasyonu kullanarak onu görüntüdeki yerel özelliklerle eşleştirir. Yani, enerji fonksiyonu minimum seviyeye iner ve görüntüyü tamamlar (W. Wang vd., 2016).

Anıl ve Natarajan (2010), aktif kontur modelini kullanan yüksek çözünürlüklü uydu görüntülerinden yarı otomatik bir yol tespit yöntemi önermektedir. Önerilen yöntemde, ilk önce, kenarları koruyarak gürültüyü gidermek için medyan filtresi kullanılmıştır. Daha sonra aktif kontur modeli kullanılarak yol bölütü çıkarılmıştır. Yöntem, yüksek çözünürlüklü uydu görüntüleri kullanılarak test edilmiştir. Yöntemin mevcut kısıtlılığı, gölgeli yollar için etkili bir şekilde çalışmamasıdır. Şu anda, LSB-Snake (Gruen ve Li, 1997), Şerit Snakes (Fua ve Leclerc, 1990), çok ölçekli Snake (Renaud ve Thierry, 2003) gibi yol tespiti için geliştirilmiş birçok Snake modeli vardır.

Seviye kümeleri (level set)

Seviye kümeleri ilk olarak (Osher ve Sethian, 1988) tarafından önerilmiştir. Eğri evrimi problemini çözmek için kısmi diferansiyel denklemleri kullanan sayısal bir analiz yöntemidir ve herhangi bir boyut uzayına uygulanabilir. Seviye kümeleri yöntemi ile uzaktan algılama görüntülerinden yol tespiti konusunda çeşitli çalışmalar yapılmıştır. Hinz ve Baumgartner (2003), yolları tespit etmek için öncelikli bilgileri kullanarak yeni bir hız fonksiyonu oluşturmak için çoklu spektral özellikleri yol geometrisi ile birleştirmiştir. Niu (2006), bir model inşa etmek için sınır gradientini, alan bilgisi ile bütünleştiren bir yöntem üzerinde çalıştı. Seviye kümeleri metodu yol ağını elde etmek için kullanıldı.

Abdollahi vd., (2018) yol tespiti için SVM ve seviye kümesi yöntemlerini entegre ederek yeni bir otomatik yöntem önermişlerdir. SVM tarafından tahmin edilen sınıflandırma olasılığı, seviye kümesi modelinde girdi olarak kullanılmıştır. Araştırmacılara göre önerilen yöntem Google Earth görüntülerinden yol tespiti için yüksek performans göstermektedir.

Abdollahi vd., (2019) tarafından yapılan diğer bir çalışmada ise İHA (insansız hava araçları) görüntülerinden yol çıkarımı için yeni bir hibrit teknik sunulmuştur. Önerilen yöntem, Trainable Weka Segmentation aracılığıyla görüntü bölütlemesinin yapılması ile başlamaktadır. Bu adımda çok sayıda görüntü özellikleri kullanılmaktadır. Ardından, yol özelliklerini çıkarmak için bölünmüş görüntülerde bir seviye belirleme

yöntemi gerçekleştirilmektedir. Daha sonra, tespit hassasiyetini artırmak için görüntülere morfolojik operatörler uygulanmaktadır. Sonunda, yol tespit hassasiyeti, elle sayısallaştırılmış yol katmanları dikkate alınarak hesaplanmıştır. Elde edilen sonuçlar, tamlık, doğruluk ve kalite değerlerinin sırasıyla ortalama %93.52, %85.79 ve %81.01 olduğunu göstermiştir.

Yapılan çalışmalar analiz edildiğinde uzaktan algılama görüntülerinden yol tespiti için seviye kümeleri yönteminin morfoloji veya kümeleme gibi diğer yöntemler ile birlikte kullanıldığını ve bu şekilde daha iyi sonuçların elde edildiğini göstermektedir (W. Wang vd., 2016).

1.3.1.6. Dinamik programlama ve gruplama

Dinamik programlama, optimizasyon karar sürecini çözmek için matematiksel bir yöntem uygulayan operasyonel araştırmaların bir koludur. Bu yöntem ile yol tespiti için yolun bir parametre olarak modele verilmesi gerekmektedir. Daha sonra dinamik programlama, kaynak noktaları arasındaki en uygun yolu belirlemek için bir hesaplama aracı olarak kullanılmaktadır (W. Wang vd., 2016).

Barzohar ve Cooper (1996), dinamik programlama ve Katman filtresi yöntemlerini kullanarak hava görüntülerinden otomatik yol tespiti sağlayan bir algoritma önermişlerdir. Önerilen algoritma kenar kısmı eksik olan veya taşıt ve köprüler tarafından kapatılmış yol bölümleri de dahil olmak üzere, yolları gerçek zamanlı olarak tespit etmektedir (Movaghati vd., 2010).

Yukarıdaki tüm çalışmalar dikkate alınıp genel bir değerlendirme yapıldığında aşağıdaki sonuçları elde etmek mümkündür:

- Genel olarak nesne tanıma ve özel olarak yol tespiti çalışmalarında kullanılan her bir yöntemin kendine göre hem avantajları hem de dezavantajları vardır.
- Yolların tespit edilmesini zorlaştıran veya önleyen çok sayıda farklı etken vardır. Dolayısıyla tüm yolları yüksek doğrulukla tespit edebilecek tek başına bir yöntem yoktur. Bu yüzden yol çıkarma yöntemleri çeşitli yöntemlerle çalışılmalı ve farklı teknikler birleştirilmelidir. Birden fazla yöntemin birlikte kullanılması tek yönetime göre genel olarak daha iyi sonuçlar vermektedir.
- Uzaktan algılama görüntülerinden yol tespitindeki temel sorun, yol özelliklerinin nasıl tanımlanacağıdır. Mevcut yöntemlerin çoğu, yolu doğrusal veya yüksek

yansıma değerine sahip bir bant olarak kabul etmekte ve bu varsayım ile nispeten kabul edilebilir sonuçlar elde edilmektedir. Bununla birlikte, artan görüntü çözünürlüğü daha ayrıntılı yol özellikleri ile birlikte daha fazla gürültü paraziti (binalar, gölgeler, yol engelleri) ortaya çıkacaktır. Bu nedenle yol nesneleri tam olarak tanımlanmalıdır (W. Wang vd., 2016).

- Günümüzün popüler yöntemlerinden olan derin öğrenme veya evrimsel sinir ağları yol tespit çalışmalarında kullanılmaktadır. Farklı kaynaklardan elde edilen bilgilerin bu yöntemlere dahil edilmesi genel olarak başarıyı artırmaktadır.
- İyi bir yol modelinin nasıl oluşturulacağı ve yolun hızlı ve doğru şekilde nasıl tespit edileceği birçok araştırmacının ortak problemi olup bu durum halen güncel bir sorun olarak varlığını devam ettirmektedir.

1.4. Problemin Tanımı

Yol ağlarının tespit edilmesi, çeşitli nedenlerden dolayı meydana gelen değişikliklerin algılanması ve hızlı bir şekilde güncellenmesi, navigasyon, şehir planlaması, ulaşım gibi birçok CBS uygulaması için hayati önem taşımaktadır. Yol ağlarının tespiti ile ilgili farklı araştırmacılar tarafından çok sayıda çalışma yapılmış olmasına rağmen (Mena, 2003, ss. 3037–3049; W. Wang vd., 2016) henüz herkes tarafından kabul edilen ve başarısı yüzde yüz olan bir yöntem yoktur. Yol tespiti, otonom sürüş, sürücü destek sistemleri ve ulaşım sistemleri alanında halen aktif ve önemli bir araştırma konusudur (Kala, 2016, s. 59,60).

Özellikle son on yılda giderek daha fazla önem kazanan ve yakın bir gelecekte hayatın bir parçası olması beklenen otonom araçlar otomatik yol tespitinin önemini artırmıştır (Mendes vd., 2016, s. 1). Otonom araçlar, başta yol ağları olmak üzere çevrenin kesin ve sağlam algılanmasını gerektirir (Fernández vd., 2017, s. 1).

Mevcut yol ağları, yollarda meydana gelen değişiklikler veya yenilemelerden dolayı sıklıkla değişebilmektedir. Bu değişikliklerin hızlı ve doğru bir şekilde algılanması son derece önemlidir. Diğer yandan deprem, heyelan, su taşkınları gibi doğal afetler yollarda çeşitli hasarlar oluşturabilmektedir. Yol hasarı tespiti ve değerlendirmesi, hasarlı yolları tespit etmeyi ve felaketten sonra yolun durumunu ve trafik kapasitesini değerlendirmeyi içerir (Jianhua Wang vd., 2015, s. 2).

Yukarıdaki bilgiler dikkate alındığında yol ağlarının tespiti halen aktif bir araştırma konusudur. Özellikle otonom araç teknolojisi yol tespitinin önemini daha da artırmıştır. Bu bilgiler ışığında, bu çalışmanın amacı açık kaynak sistemlerden elde edilen verileri kullanarak bir veri seti oluşturmak, bu veri seti ile mevcut anlamsal bölütleme tekniklerini eğiterek yol ağlarının hızlı ve güvenli bir şekilde tespit edilmesini sağlamak ve mevcut yöntemlere alternatif özelleştirilmiş bir tam evrimsel ağ (TEA) modeli sunmaktır.

1.5. Çalışmanın Hipotezi

Çalışma, topluluk tabanlı açık kaynak platformlardan elde edilen raster ve vektör verilerin bir araya getirilip derin öğrenme modeli için veri seti olarak kullanılabileceği ve bu veri seti ile eğitilen anlamsal bölütleme modellerinin yol ağı tespit çalışmalarında başarılı bir şekilde kullanılabileceği varsayımına dayanmaktadır. Çalışma birbirinden bağımsız ve fakat birbirinin devamı olan çok sayıda adımdan oluşmaktadır. Her bir adım kendi içinde çok sayıda problem içermektedir. Özellikle veri setini meydana getiren veri kaynaklarının birbirinden bağımsız iki farklı platformdan elde edilmesi ortaya daha fazla problem çıkmasına neden olmaktadır. Çalışma aynı zamanda hipoteze bağlı olarak ortaya çıkan aşağıdaki problemleri çözme adımlarını içermektedir.

- Tamamen ücretsiz ve herkesin kullanımına açık olan vektörel formattaki OSM verilerini, yol ağları için etiket veya maske verisi olarak kullanmak mümkün müdür?
- OSM veri modelinde çok sayıda yol türü tanımlanmıştır. Tüm yol türleri şerit sayısı veya şerit genişliği bilgisi içermeyen çizgi nesneleri (linestring) ile ifade edilmektedir. OSM'den elde edilen vektörel formattaki yol verilerini ve bu yollara ait öznitelik bilgilerini kullanarak yol genişliklerini tespit etmek mümkün müdür?
- Yol genişlikleri eklenmiş vektör formatındaki OSM verileri, aynı koordinat sistemi ile raster bir görüntüye dönüştürülebilir mi?
- OAM platformu ticari ve kişisel kullanım için ücretsiz olan, farklı çözünürlükteki çok sayıda uzaktan algılama görüntülerini depolamakta ve paylaşmaktadır. Sınırlama kutusu ile sınırları belirlenen bir bölgeye ait yüksek çözünürlüklü görüntüler OAM platformundan otomatik olarak indirilebilir mi?
- OAM platformundan elde edilen ve farklı koordinat sistemlerine sahip görüntüler, herhangi bir veri kaybı olmadan ortak bir koordinat sistemine dönüştürülebilir mi?

- Yüksek mekânsal çözünürlüklü görüntüleri ve gerçek yol genişliklerine sahip yol ağlarını içeren raster görüntüleri veri kaybı yaşamadan eş zamanlı olarak 512x512, 256x256 gibi istenilen boyutlardaki küçük görüntülere (bundan sonra görüntü dilimi olarak ifade edilecektir) bölünebilirler mi?
- Yüksek çözünürlüklü uydu görüntülerinden elde edilen görüntü dilimlerini gerçek görüntü ve yol ağlarını içeren raster görüntülerden elde edilen görüntü dilimlerini maske (etiket) verisi olarak kullanmak mümkün müdür?
- Yukarıda elde edilen veri setini kullanarak anlamsal bölütleme yöntemi ile yol ağlarını yüksek doğrulukla tespit eden özelleştirilmiş bir anlamsal bölütleme modeli geliştirilebilir mi?

1.6. Çalışmanın Özgün Değeri ve Yaygın Etkisi

Günümüzde derin öğrenme yöntemleri birçok alanda aktif olarak kullanılmaktadır. Uzaktan algılama ve coğrafi bilgi sistemlerinde bu yöntemler yaygın bir şekilde kullanılmaktadır. Diğer yandan uzaktan algılama ve coğrafi bilgi sistemlerinin sahip olduğu veri miktarı ve kalitesi bu alandaki çalışmalar için önemli bir motivasyon kaynağıdır. OSM ve OAM, UA ve CBS uygulamaları için birer ham veri kaynağıdır. Her iki platform farklı veri yapısına, veri depolama modeline ve servislere sahiptir. Birçok yönden birbirinden farklı teknolojilere sahip olan bu platformdaki verilerin ortak amaç doğrultusunda birlikte kullanılmaları çalışmanın özgün yanlarından birisidir. Diğer yandan derin öğrenme yöntemleri ile çok sayıda yol tespiti çalışması yapılmış olmasına rağmen OAM ve OSM verilerini birlikte kullanarak anlamsal bölütleme yöntemi ile yapılan bir yol ağı tespit çalışmasına rastlanmamıştır. Bu durum, bu açıdan çalışmayı özgün kılmaktadır. Bu çalışmanın bir diğer özgün noktası ile çalışmanın veri hazırlama sürecinden derin öğrenme modelinin eğitilmesine kadar olan bütün adımlarının otomatik olarak yapılmasıdır.

Çalışma kapsamında yol genişliği bilgisine sahip olmayan vektörel formattaki OSM verilerine yol genişliklerinin eklenmesini sağlayan ve açık kaynak olarak Github üzerinden paylaşılan OSM2RN (Daş, 2019) adlı bir araç geliştirilmiştir. Bu aracın yanında çalışma kapsamında geliştirilen ve ileride Github gibi platformlar üzerinden açık kaynak kod olarak paylaşılması planlanan başka araçlar da geliştirilmiştir.

2. KURAMSAL TEMELLER

Bu bölümde çalışmanın daha iyi anlaşılmasını sağlamak amacıyla topluluk tabanlı projelerden elde edilen verilerin derin öğrenme modelinde işlenerek yol ağlarının tespit edilmesini ve modellenmesini sağlayan OSM, OAM platformları, OSM yol verileri, makine öğrenmesi, derin öğrenme (kütüphaneleri), görüntü sınıflandırma ve anlamsal bölütleme gibi kavramlar açıklanmaktadır.

2.1. Topluluk Tabanlı Açık Kaynak Projeler

İnternetin yaygınlaşması ve özellikle de Web 2.0 teknolojisi ile birlikte CBS ve OSM gibi konularda bilgisi, özel yeteneği veya teknik araç gereci olmayan internet kullanıcıları sahip oldukları GPS destekli cep telefonu, bileklik vb. cihazlar ile birer içerik üreticisine dönüşmüşlerdir. 2000'li yıllardan itibaren ortaya çıkan bu durum, bazı kullanıcıların, toplulukların, vakıfların veya şirketlerin bu potansiyeli değerlendirmek amacıyla çeşitli projeler geliştirmelerini sağlamıştır. Topluluk, bu çalışmadaki kapsam itibari ile aynı proje veya fikre destek veren (içerik üreten, düzenleyen, organize eden vb.) kullanıcılar grubu olarak ifade edilebilir. Açık kaynak projeler ise proje kapsamında üretilen veri, bilgi, içerik veya kaynak kodların özgürce herkes tarafından erişilebilmesine veya dağıtılabilmesine imkân veren organizasyonlardır. Farklı topluluk tabanlı projeler, içeriklerini farklı lisanslar altında sunmakla birlikte genel olarak bu projeler tarafından tercih edilen lisanslar mantık olarak benzer özelliklere sahiptirler. Bir kullanıcı açık kaynak içeriğe erişebilir, bu içeriği paylaşabilir, aynı özgürlüğü diğer kullanıcılara da tanıyarak değiştirebilir veya dönüştürebilir (Open Data Commons, 2019a; St. Laurent, 2004, s. 5).

2.2. OpenAerialMap

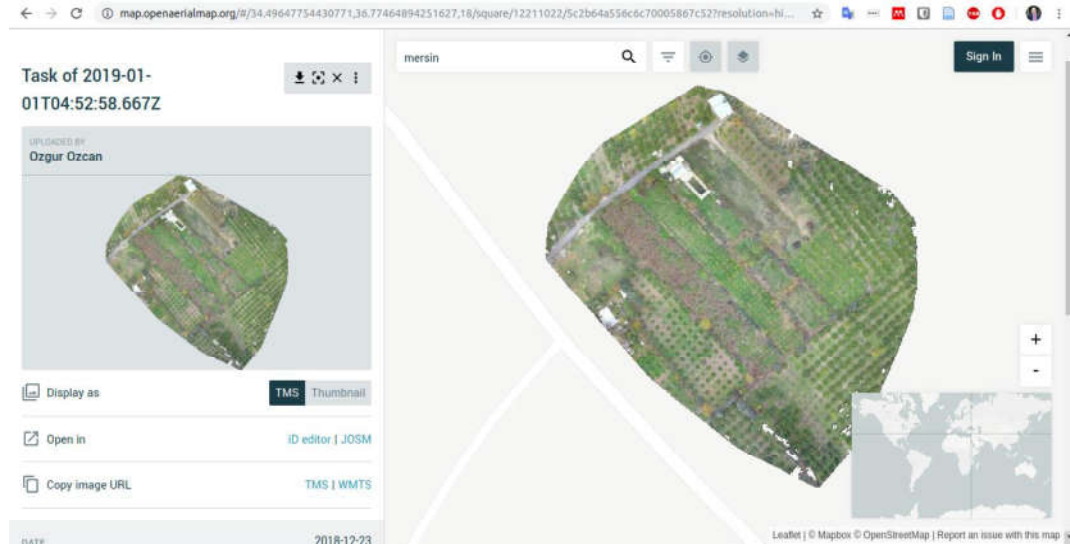
OAM, açık lisanslı uydu ve insansız hava aracı (İHA) görüntülerini aramak, paylaşmak ve kullanmak için kullanılan bir araçlar setidir. OAM, Open Imagery Network (OIN) üzerine kurulu bir platform olup OIN tarafından sağlanan görüntülere erişim ve görüntülerde arama sağlayan açık bir hizmettir (OpenAerialMap Contributors, 2019) (Smith vd., 2016). OIN, açık lisanslı görüntülerin keşfedilebilir bir ağıdır.

OAM projesi, Humanitarian OpenStreetMap Team (HOT) adlı ekip tarafından yönetilmektedir. HOT, açık haritalama yoluyla insani eylem ve toplum gelişimine odaklanmış uluslararası bir ekiptir. Bu ekip, afet yönetiminde devrim yaratan, riskleri

azaltan ve sürdürülebilir kalkınma hedeflerine ulaşılmasına katkıda bulunan harita verilerini sağlamaktadır (Humanitarian OpenStreetMap Team, 2019).

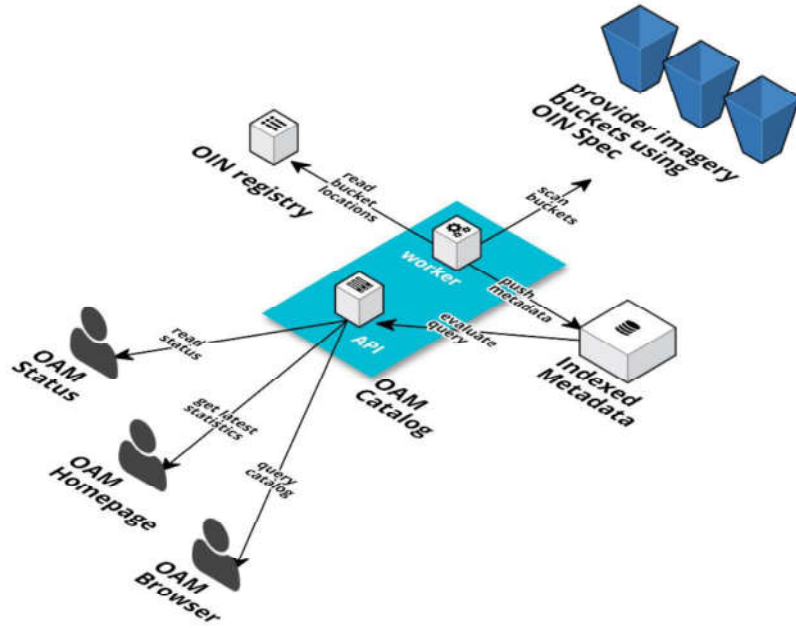
OAM projesi, 2007 ve 2008 yıllarında aktif olarak faaliyet göstermiş fakat daha sonra çalışmalarına son vermiştir. Barındırma ve lisanslamayı ölçeklendirme gibi iki ana zorluğun birleşimi OAM projesinin durdurulmasına neden olmuştur. HOT, görüntü verilerini barındırma, yükleme, paylaşma, arama, filtreleme, görüntüleme, indirme ve kullanma işlevlerini sağlayan açık kaynak kodlu bir araç kümesi olan OAM'in geliştirilmesiyle ilgili bir kriz haritası hazırlamıştır. HOT, OAM'ı yeniden başlatmak için çalışmalar yapmış ve sonuç itibarı ile çeşitli fonlardan alınan hibeler ile proje 2015 yılında tekrar başlatılmış ve günümüzde gittikçe gelişen bir yapıda yoluna devam etmektedir. Bununla birlikte, OAM'nin hava görüntülerini toplamak için genel amaçlı bir araç olması hedeflenmiştir. İsteyen her kullanıcı platforma görüntü yükleyebilir. OAM ve OIN tarafından sağlanan tüm görüntüler Creative Commons lisansı altında lisanslanmıştır ve bu görüntüler OSM üzerinden de erişilebilir durumdadır (OpenStreetMap Contributors, 2018b).

Şekil 2.1'de Mersin bölgesindeki bir alana ait 23.12.2018 tarihinde çekilmiş yüksek çözünürlüklü uzaktan algılama görüntüsü görülmektedir. İlgili görüntünün mekânsal çözünürlüğü 2 cm'dir.



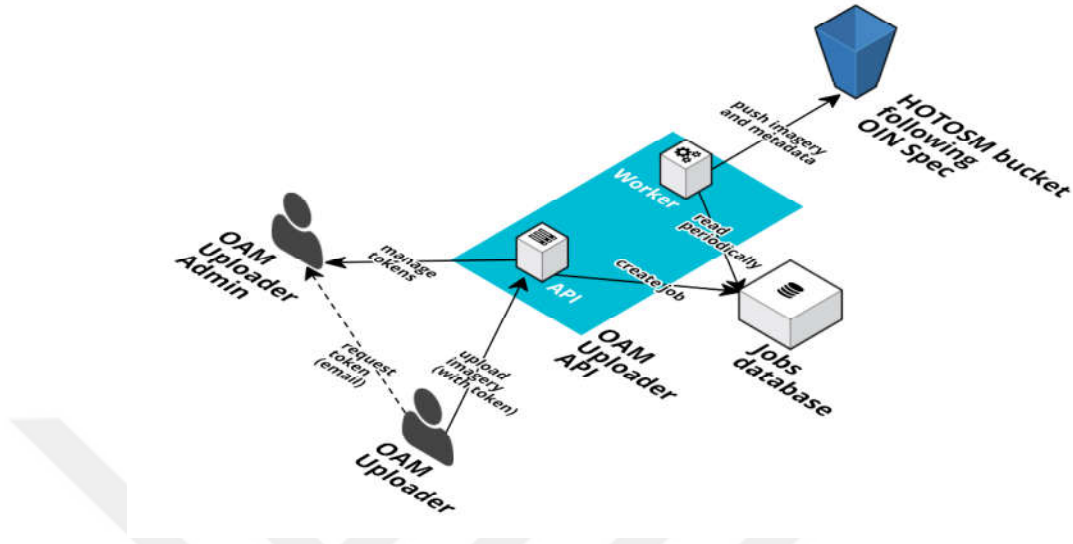
Şekil 2.1. Mersin bölgesindeki bir alana ait 23.12.2018 tarihinde çekilmiş 2 cm mekânsal çözünürlüklü OAM görüntüsü

OAM platformu temel olarak catalog ve contributor olmak üzere iki ana bileşenden oluşmaktadır. Catalog bileşeni, görüntü meta verilerini işleme ve OAM kataloğu ise oluşturma ile ilgilidir. Şekil 2.2'de görüldüğü gibi bu bileşen OAM Catalog, OAM Browser, OAM Homepage ve OAM status alt bileşenlerinden oluşmaktadır. OAM Catalog'un önemli alt bileşenlerinden birisi API'dir. API, katalog üzerinde zaman ve alana göre meta veriler üzerinde sorgulamalar yapmak için bir arayüz sağlamaktadır. Ayrıca, OAM ana sayfasını güçlendiren basit istatistikler de sağlar. Bu bileşen NodeJS dilinde yazılmıştır (OpenAerialMap Docs, 2019a). Node.js açık kaynaklı, platformlar arası bir JavaScript çalışma zamanı ortamıdır. JavaScript kodunu bir tarayıcı dışında çalıştırır.



Şekil 2.2. OAM katalog bileşeni ve alt bileşenleri (OpenAerialMap Docs, 2019a)

Contributor bileşeni (Şekil 2.3) ise OAM platformuna yeni veri yükleme ile ilgili işlemleri içerir. Bir katılımcının bir yükleme formu aracılığıyla sisteme veri eklemesine imkân verir. Yüklenen veriler HOTOSM görüntü grubuna eklenir. Çok fazla görüntüye sahip olan katılımcılar, OIN ağı içinde kendi görüntü gruplarını kurabilirler. Şekil 2.3'ten de görülebileceği gibi Contributor bileşeni her biri farklı bir işlevi yerine getirmek için özelleştirilmiş OAM Uploader, OAM Uploader Admin ve OAM Uploader API alt bileşenlerinden oluşmaktadır (OpenAerialMap Docs, 2019b).



Şekil 2.3. OAM Contributor bileşeni ve alt bileşenleri (OpenAerialMap Docs, 2019b)

2.2.1. OpenAerialMap platformundan veri indirme

OAM sunduğu büyük miktarda veriye rağmen, basit ve kullanımı kolay bir arayüze sahiptir. Bu arayüz sayesinde birçok harita servisinde olduğu gibi platform üzerinde çeşitli kriterlere göre arama veya sorgulamalar yapılabilmektedir. Elde edilen sonuçlar doğrudan bu arayüz üzerinden indirilebilmektedir. Ayrıca OAM, kullanıcılarına web arayüzünü kullanmaya ihtiyaç duymadan görüntü yükleme, işleme gibi işlemleri yapmalarına imkân veren bir API sunmaktadır.

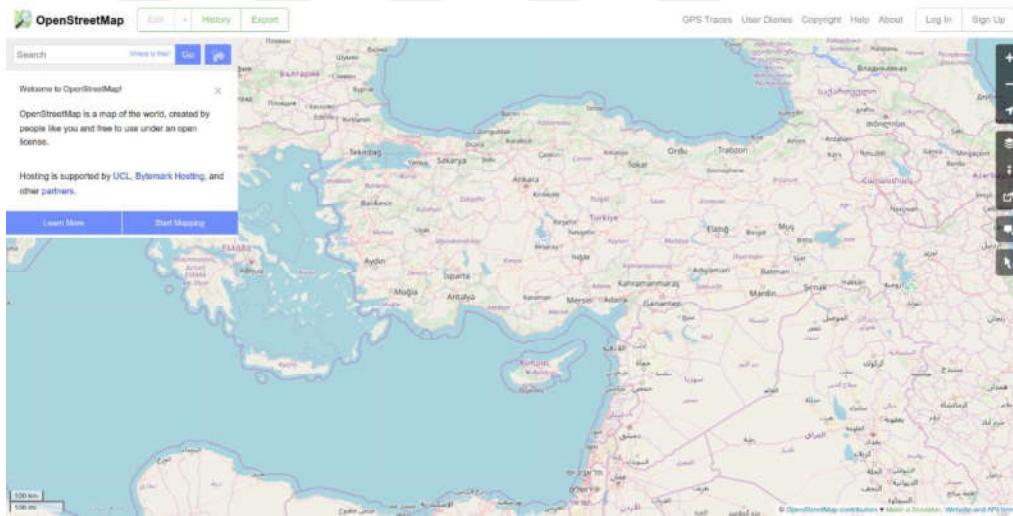
API sunucusu, kullanıcı belirteçlerinden gerçek yüklemelere ve işlemeye kadar görüntü yüklemesi ile ilgili hemen hemen tüm işlemleri yapmaya imkân vermektedir. API sunucusu verileri JSON formatında vermektedir. Bu veriler üzerinde çeşitli sorgular veya filtreler oluşturmak mümkündür.

2.3. OpenStreetMap

OSM tüm dünyaya ait tamamen ücretsiz ve açık kaynak bir harita oluşturmak amacıyla 2004 yılında İngiltere'de ortaya çıkmış bir projedir. Bu projenin amacı dünyadaki her coğrafi özelliğin bir kaydını tutmaktır. Cadde ve sokakların haritalandırılması amacıyla başlatılan bu proje; günümüzde çevreyi oluşturan patikalar, binalar, su yolları, boru hatları, ormanlık alanlar, plajlar, posta kutuları, alışveriş merkezleri, otobüs durakları, parklar, ibadet yerleri ve daha pek çok coğrafi nesneye ait ayrıntılı veriler tutmaktadır. Proje, fiziki coğrafyanın yanı sıra idari sınırları, arazi

kullanımının ayrıntılarını, otobüs güzergahlarını ve peyzajın kendisinde görünmeyen diğer soyut fikirleri de içermektedir (Bennett, 2010, s. 8; Mirvahabi ve Abbaspour, 2015).

OSM'nin içerikleri Wikipedia'da olduğu gibi dünyanın çeşitli yerlerindeki gönüllü kullanıcılar tarafından oluşturulmaktadır. OSM, bu kullanıcıları haritacılar (mappers) olarak adlandırmaktadır. OSM veri tabanı, haritacıların araç, bisiklet veya yaya olarak yol kat ederek oluşturdukları hareketlerini GPS alıcılarını kullanarak kaydetmeleri ile meydana gelen verilerden oluşur. Bu bilgiler daha sonra haritalara dönüştürülebilen veya navigasyon için kullanılabilir bir dizi nokta ve çizgi oluşturmak için kullanılır (Bennett, 2010, s. 8). Gönüllüler tarafından anket yolu ile elde edilen bilgiler, hava fotoğrafları, bağış yolu ile elde edilen veriler ve diğer ücretsiz içerik sağlayıcılardan elde edilen veriler OSM tarafından kullanılan diğer veri kaynaklarıdır.



Şekil 2.4. OSM web arayüzü

OSM'deki tüm içerikler ODbL ile lisanslanmaktadır. Bu lisansa göre OSM verilerini ticari veya kişisel amaçlı olarak kullanmak tamamen serbest ve ücretsizdir. OSM verilerini kopyalamak, değiştirmek ve yeniden dağıtmak mümkündür (Guibault ve Wiebe, 2013, s. 157).

OSM, diğer coğrafi veri tabanlarından daha sık güncellenmektedir. Aslında, OSM anlık olarak güncellenir ve en son eklenen veriler her zaman indirmek için kullanılabilir. Yani OSM veri tabanında yapılan değişiklikler anında yayınlanmaktadır. Ticari veya tescilli veri tabanları sık sık güncellenebilse bile, bu değişiklikler genellikle müşterilere çok hızlı bir şekilde sunulmazlar. Bu durum kullanıcıların bir süre daha güncel olmayan

veri tabanlarını kullanmasına neden olur. Buna karşılık, her hafta OSM'deki tüm verilerin yeni ve eksiksiz bir kopyası hem sıkıştırılmış bir XML dosyası hem de özel bir PBF formatı dosyası olarak sunulur. Bu dosyaya *Planet* denir. Bunun yanında sadece haftalık değişiklikleri içeren fark dosyasını da indirmek mümkündür. Ayrıca, sadece güncel verileri değil, aynı zamanda mevcut verilerin eski sürümlerini ve silinmiş veri öğelerini içeren geçmiş dosyası da mevcuttur (OpenStreetMap Contributors, 2019c).

OSM projesinin temel amacı coğrafi verileri toplamak olsa da, proje üyeleri verileri doğru ve hızlı bir şekilde üreten, düzenleyen, yöneten veya kullanan açık kaynak kodlu çok çeşitli yazılımlar üretmişlerdir (Bennett, 2010, s. 9).

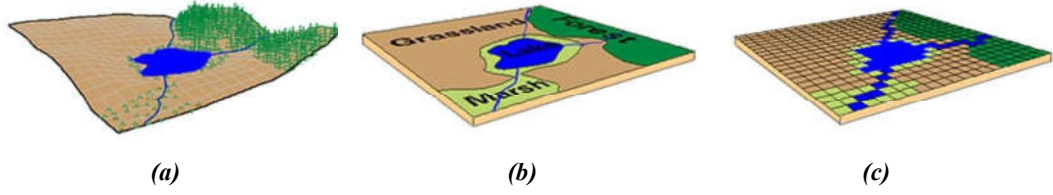
2.3.1. OpenStreetMap veri modeli

Doğal olguları görselleştirmek için önce coğrafi mekânın en iyi ne şekilde temsil edileceği belirlenmelidir. Veri modelleri, bilgisayar ortamında gerçek dünyayı oluşturan tüm nesnelerin özellik ve özniteliklerini bozmadan tanımlamak ve temsil etmek için kullanılan bir dizi kural ve/veya yapılarıdır. Diğer bir ifade ile veri modeli, veri öğelerini düzenleyen ve birbirleriyle ve gerçek dünyadaki varlıkların özellikleriyle ilişkilerini standartlaştıran soyut bir modeldir. Mekânsal bilgiyi bilgisayar ortamında oluşturmak veya saklamak için raster ve vektör olmak üzere iki temel veri modeli vardır. Coğrafi veri raster, vektör veya aynı anda her iki formatta da olabilir (Abdul-Rahman ve Pilouk, 2007, s. 41).

Raster veri modeli, düzlemsel bir yüzey oluşturmak için birbirine bağlanmış eşit boyutlu piksellerin satırlarından ve sütunlarından oluşur. Bu pikseller noktalar, çizgiler, alanlar, ağlar ve yüzeyler oluşturmak için yapı taşları olarak kullanılır. Dijital hava fotoğrafları, uydulardan gelen görüntüler, dijital resimler ve hatta taranmış haritalar raster formatındadır.

Vektör verileri, coğrafi bilgi sistemleri tarafından yönetilen ana kategorilerden birini temsil eder. Vektör veri modeli, gerçek dünyayı temsil etmek için koordinat değerleriyle tanımlanan noktalar, çizgiler ve çokgenler gibi öğeleri kullanır (Schneider vd., 2005). Vektör veri gösterimi, yakınlık ve bitişiklik gibi uzamsal ilişkilere ve yukarı ve bağlı gibi topolojik ilişkilere dayanan analizlere izin verir (M. Zeiler, 1999, s. 8).

Şekil 2.5'te dünyaya ait örnek bir görüntü, bu görüntüsünün raster formatındaki ve vektör formatındaki temsili gösterimleri yer almaktadır.

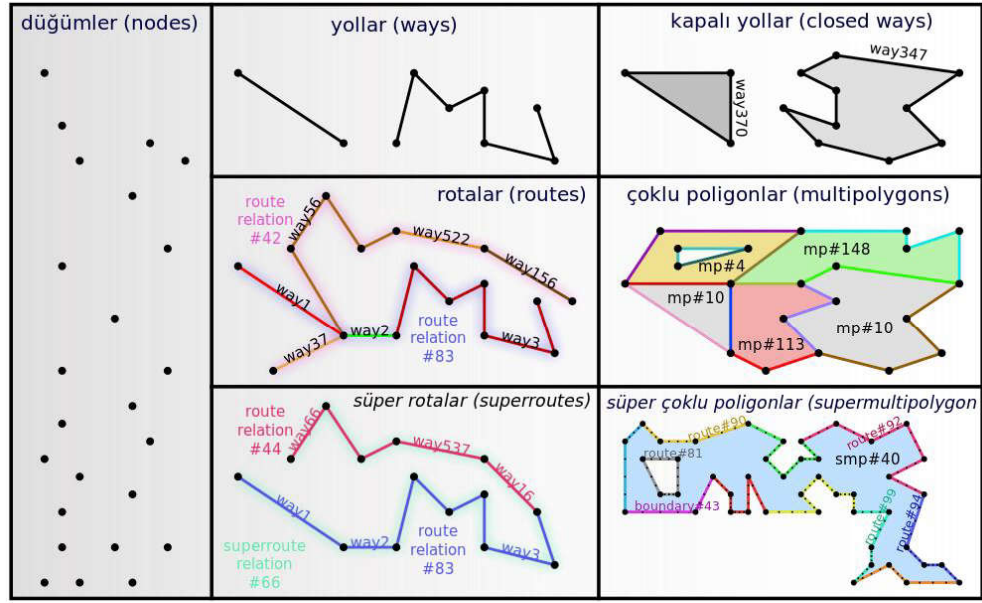


Şekil 2.5. (a) Temsili bir gerçek dünya görüntüsü, (b) Temsili gerçek görüntünün raster formatındaki gösterimi, (c) Temsili gerçek görüntünün vektör formatındaki gösterimi (IMSMA Wiki Community, 2019)

Geleneksel CBS'de harita verileri nokta (point), çizgi (line) ve çokgen (polygon) olmak üzere üç farklı şekilde temsil edilir. Nokta, koordinatlarla tanımlanan uzayda tek bir konumu ifade etmek için kullanılır. Çizgi, bir yolu veya bir sınırı temsil eden doğrusal bir özelliktir. Çokgen ise kapalı bir alanı ifade etmek için kullanılır. Bu coğrafi özellikleri tanımlayan veriler, genellikle ikincil bir veri tabanında bir sonuç olarak eklenir.

OSM veri modeli, coğrafi bilgileri göstermenin güçlü ve basit bir yoludur. Veri modelinin doğru bir şekilde anlaşılması OSM verileri üzerinde ihtiyaç duyulan değişikliklerin / dönüşümlerin rahat bir şekilde yapılmasına imkân tanır. OSM'de her bir özellik ekli öznitelik verisi ile birlikte geometri olarak tanımlanır. Örneğin bir nehir ele alındığında nehrin şekli geometriler ile tanımlanırken nehrin adı öznitelik bilgisi olarak ifade edilmektedir. Geometriler düğümler (nodes), yollar (ways) ve ilişkiler (relations) olmak üzere üç farklı nesne açıklanmaktadır (Mirvahabi ve Abbaspour, 2015). Öznitelikler, bir düğümün, bir yolun veya ilişkinin parçası olabilecek *anahtar:değer* (key:values) çiftlerinden meydana gelen etiketler olarak tanımlanır (Bennett, 2010, s. 61). Düğümler, bir noktaya eşdeğerdir; yollar, noktaları birleştiren çizgiler gibidir ve ilişkiler, noktalar topluluğu veya daha büyük bir bütünü temsil eden yollardır. Düğümler, OSM veri modelindeki ana öğelerden biridir. Enlem, boylam ve düğüm kimliği (node ID) ile tanımlanan uzayda tek bir noktadan oluşur.

Yollar (ways), normalde en az bir etiketi olan veya bir ilişki içine dahil edilmiş olan sıralı bir düğüm listesidir. Bir yol açık veya kapalı olabilir. Kapalı bir yol, kapalı bir çoklu çizgi ya da bir alan ya da her ikisi olarak yorumlanabilir. Kapalı yolların başlangıç düğümleri ile bitiş düğümleri ortak iken açık yollarda bu düğümler farklıdır. Şekil 2.6'da OSM veri yapıları görülmektedir.



Şekil 2.6. OSM veri modeli yapıları (OpenStreetMap Contributors, 2016a)

Etiketler geometrilerle tanımlanan nesnelere ait tüm özellikleri tutmaktadır. Etiketler aynı zamanda aynı geometri türü ile ifade edilen nesnelerin birbirinden ayrılmasını sağlamaktadır. Düğümün bir kafe, bir okul, bir ağaç, bir park, bir dağın zirvesi olup olmadığı tamamen düğümün etiketlenme biçimine bağlıdır. OSM veri modelinde etiketler *anahtar=değer* (*key=value*) şeklinde tanımlanır (Bennett, 2010, s. 61). *highway=residential*, *maxspeed = 50* etiketlerin kullanımına ait örneklerdir. Etiket bilgisinden faydalanarak OSM verisi üzerinde çeşitli sorgulamalar veya filtreler oluşturmak mümkündür.

İlişkiler, birden fazla düğümü veya yolu daha büyük bir bütün halinde düzenlemek için kullanılır. Bir ilişki birden fazla elementin bir araya gelmesiyle meydana gelmiş grubu ifade eder. İlişkiler, nesneler arasındaki mantıksal (ve genellikle yerel) veya coğrafi ilişkileri modellemek için kullanılır. OSM veri modelinde ilişki için en fazla 300 üye kullanılması önerilir. *Relation:boundary* ifadesi idari sınırları tanımlamak için kullanılan bir ilişki örneğidir. *Relation:restriction* ifadesi ise sola dönüşün veya U dönüşünün yasak olduğunu ifade etmek için kullanılan ilişki örneğidir (OpenStreetMap Contributors, 2016b).

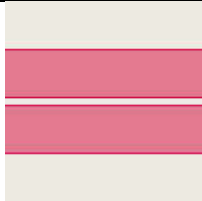
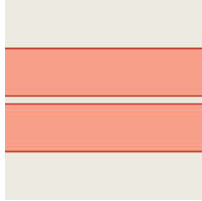
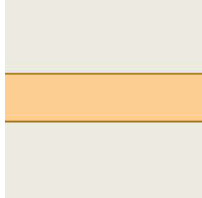
2.3.2. OpenStreetMap yol verileri

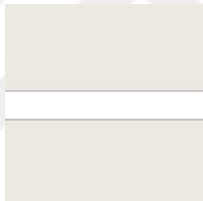
OSM’de caddeleri, sokakları ve yolları tanımlamak için *highway* anahtar kelimesi kullanılmaktadır. *highway=** ifadesi bir bölgeye ait tüm yolları ifade etmektedir. Anahtarın değeri, yol ağı içindeki yolun bir bütün olarak önemini göstermeye yardımcı olmaktadır.

OSM’de temel yolları birbirinden ayırmak ve sınıflandırmak için yedi farklı anahtar kelime kullanılmaktadır. Dünyadaki farklı kıtalar veya bölgeler dikkate alındığında bu sınıflandırmanın tüm ülkelere uygun olmadığı görülmektedir. Fakat bu tanımlamanın genel ve geniş bir tanımlama olduğu göz önünde bulundurulmalıdır.

Tablo 2.1, OSM veri modelinde tanımlanan temel yol türlerini ve her bir yol türüne ait temel özelliklerini göstermektedir (Bennett, 2010, s. 66). OSM’de yolları tanımlamak için bu tabloda verilen anahtar kelimelerin dışında *living street*, *service* gibi özel amaçla kullanılan yolları tanımlayan anahtar kelimeler ve yolların birbirleri ile olan ilişkilerini tanımlamak için kullanılan *trunk_link*, *primary_link* gibi anahtar kelimeler de vardır.

Tablo 2.1. OSM’deki yol türleri ve temel özellikleri (OpenStreetMap Contributors, 2019b)

Anahtar Kelime	Yol Özellikleri	Bölünmüş Yol	Yol Modeli	Yol Görüntüsü
Motorway	Erişim kısıtlamaları olan uzun mesafeli, yüksek hızlı yollar (otoban)	Evet		
Trunk	Daha az erişim kısıtlamaları olan diğer uzun mesafeli yollar (şehirlerarası yol)	Evet		
Primary	Genel amaçlı büyük yollar	Hayır		

Secondary	Genel amaçlı küçük Hayır yollar		
Tertiary	Daha küçük yerleşim Hayır yerlerini bağlayan yollar		
Unclassified	Yukarıdaki etiketlerin Hayır hiçbirisi tarafından kapsanmayan yollar		
Residential	Trafiğin olmadığı Hayır yerleşim yerlerindeki (kentsel veya kırsal) yollar		

2.3.3. OpenStreetMap verisini indirme

OSM veri tabanından harita verilerini çeşitli şekillerde indirmek mümkündür. Veri setinin tamamı OSM web sitesi indirme alanında (planet.openstreetmap.org) mevcuttur. Ayrıca daha küçük alanlar seçilip sadece bu alanlara ait verileri indirmek de mümkündür. İndirilen veriler varsayılan olarak OSM formatındadır. Bu format biçimlendirilmiş XML dosya biçimidir. OSM verilerini sunan ve farklı ihtiyaçlara yönelik hazırlanmış başka servisler veya platformlar da mevcuttur.

- www.overpass-api.de
- download.openstreetmap.fr
- download.geofabrik.de

ön plana çıkan bazı önemli platformlardır.

Planet.osm tüm dünyaya ait vektörel verileri içermektedir. Her hafta, OSM'deki tüm verilerin yeni ve eksiksiz bir kopyası hem sıkıştırılmış bir XML dosyası hem de özel bir PBF formatı dosyası olarak sunulmaktadır. 01.12.2019 tarihindeki dosya BZ2

formatında sıkıştırılmış olarak yaklaşık 83 GB, düz OSM XML formatında ise yaklaşık 154.1 GB ve PBF sıkıştırılmış olarak ise 48,1 GB boyutundadır. Planet.osm platformu üzerinden sadece son bir haftadaki güncellemeleri içeren verileri de indirmek mümkündür. Tüm dünyaya ait bu veri yaklaşık olarak 3 GB boyutundadır. Bu çalışma kapsamında uygulamanın çevrimdışı çalışmasını sağlamak amacıyla ihtiyaç duyulan veriler

<https://planet.openstreetmap.org/planet/planet-latest.osm.bz2>

adresinden BZ2 formatında sıkıştırılmış olarak indirilmiştir. Çevrimiçi çalışma için ise Overpass API kullanılmıştır. Overpass API, OSM verilerinin kullanıcı tarafından belirlenen arama kriterlerine göre sorgulanmasına imkân veren bir platformdur. Bu amaçla, özel olarak hazırlanmış Overpass XML ve Overpass QL olmak üzere iki sorgu dili vardır. Overpass API'nın temel mantığı, OSM verilerinin saklanmasını sağlayan düğümler, yollar, poligonlar üzerinde birbiri ardına sorgu ifadeleri oluşturmak ve böylece istenilen bilginin üretilmesi ve değiştirilmesi mantığına dayanmaktadır (Jokar Arsanjani vd., 2015).

2.4. Sınırlayıcı Kutu (Bounding Box)

Sınırlayıcı kutu (BBOX), iki boylam ve iki enlem ile tanımlanan bir alanın sınırlarını ifade etmektedir. BBOX standart yapısı aşağıdaki gibidir (OpenStreetMap Contributors, 2019a):

bbox = sol_koordinat, alt_koordinat, sağ_koordinat., üst_koordinat

bbox = minimum_boylam, minimum_enlem, maksimum_boylam, maksimum_enlem

<https://api.openaerialmap.org/meta?bbox=bbox>

Overpass API'sı tarafından kullanılan BBOX değerlerinin sırası standart sıradan farklı olup ise aşağıdaki gibidir (OpenStreetMap Contributors, 2019a):

bbox = minimum_enlem, minimum_boylam, maksimum_enlem, maksimum_boylam

2.5. Datum, Projeksiyon ve Koordinat Sistemi

Dünya üzerindeki yerleri doğru bir şekilde tanımlamak CBS uygulamaları için son derece önemlidir. Bir yerin enlemi, boylamı (ve yüksekliği) bu yerin dünyanın neresinde olduğunu tanımlamak için yeterli değildir. Her biri dünya çapında farklı bölgelerde, farklı

bir doğruluk düzeyini temsil eden birçok datum ve koordinat sistemi vardır. Bu nedenle, doğru yerleri belirlemek için bir koordinat sistemi ve datum gereklidir ve Dünya'nın mükemmel bir ölçek modeli kullanılmadığı sürece, bu konumları farklı bir yüzeyde görsel olarak temsil etmek için bir projeksiyon kullanmak gerekmektedir. Doğru datum ve projeksiyonu kullanmak çalışmanın başarısını ve hassasiyetini doğrudan etkileyen faktörlerdendir. Diğer yandan datum, projeksiyon ve koordinat sistemi terimleri genellikle yanlış yorumlanır ve yanlış anlaşılırlar. Bir datum sadece mekânsal ölçümler için bir temel ve referanstır. Daha sonra bu ölçümleri datuma göre tanımlamak için bir koordinat sistemi kullanılır ve bir projeksiyon, bu ölçümlerin farklı bir yüzeydeki görsel temsidir. Bir projeksiyon, üç boyutlu bir yüzeyin iki boyutlu bir temsili olarak ifade edilebilir (Thomas, 2017).

Datum parametre bilgileri aynı zamanda bir koordinat sistemi de tanımlamaktadır. Koordinat sistemi, coğrafi özelliklerin, görüntülerin ve GPS konumları gibi gözlemlerin ortak bir coğrafi çerçeve içinde konumlarını temsil etmek için kullanılan bir referans sistemidir. Coğrafi koordinat sistemi (geographic coordinate system) ve projeksiyonlu koordinat sistemi (projected coordinate system) yaygın olarak kullanılan projeksiyon sistemi türleridir (Environmental Systems Research Institute, 2010).

2.5.1. Koordinat sistemi dönüşümü

Birden fazla veri kümesinin ortak koordinat sistemine sahip olmadığı durumda coğrafi (datum) dönüşüm yapılmalıdır. Bu işlem, temelde iki coğrafi koordinat sistemi arasındaki koordinatları dönüştürmek için iyi tanımlanmış bir matematik yöntemidir (Environmental Systems Research Institute, 2010).

Koordinat sistemi dönüşümü vektör-vektör veya raster-raster gibi aynı veri formatına sahip veriler üzerinde yapılabildiği gibi; vektör verinin raster veriye veya raster verinin vektör veriye dönüştürülmesi işlemlerinde de kullanılabilir. Bazı durumlarda iki veri seti arasında doğrudan bir dönüşüm mümkün olmaz. Bu gibi durumlarda iki veri setinin örneğin WGS-84 gibi üçüncü bir coğrafi koordinat sistemine dönüştürülmesi ve ardından iki dönüşümün birleştirilmesi gerekebilir (Environmental Systems Research Institute, 2010).

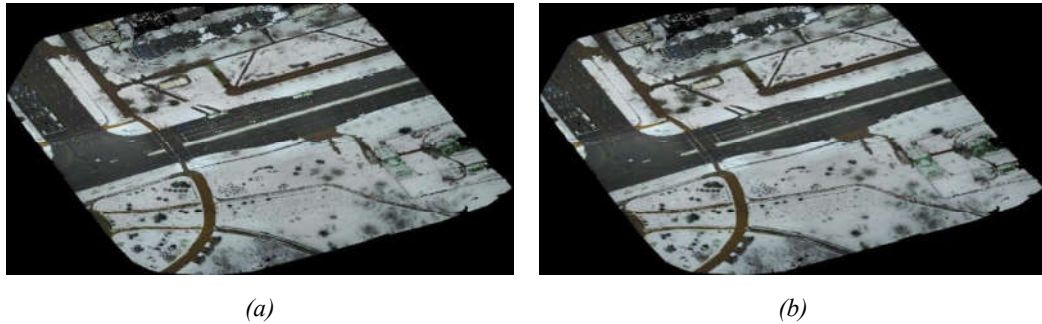
Bu çalışma kapsamında kullanılan OAM ve OSM verileri konumsal veri olup çeşitli datum ve koordinat sistemlerine sahiptirler. OSM, çoğu GPS alıcısında olduğu gibi

yaygın olarak WGS-84 koordinat sistemini kullanır. Bununla birlikte, OSM’de diğer birçok koordinat sistemi de mevcuttur. Bölgeye ve uygulamaya bağlı olarak bu koordinat sistemleri de kullanılmaktadır (OpenStreetMap Contributors, 2018a).

OpenAerialMap veri tabanında dünyanın farklı bölgelerine ait çok sayıda görüntü vardır. Bu görüntülerin bir kısmı farklı koordinat sistemlerine sahipken bir kısmı aynı koordinat sistemine sahip olmakla birlikte koordinat sistemini oluşturan farklı parametrelere (örneğin farklı ölçü birimi) sahiptir. Dolayısıyla bu veriler üzerinde ortak işlem yapmak için öncelikle gerekli koordinat veya parametre dönüşümlerinin yapılması gerekmektedir.

Koordinat sistemi dönüşümleri arka planda ciddi matematiksel işlemlere ve formüllere dayanmaktadır. CBS tarafında kullanılan birçok yazılım bu dönüşümleri yapmak için yerleşik fonksiyonlara sahiptir. Python ortamında projeksiyon dönüşümü için Rasterio (Gillies ve vd., 2013) adlı kütüphane yaygın bir şekilde kullanılmaktadır. Rasterio, bir hedef raster tipindeki bir hedef görüntünün piksellerini ilişkili bir koordinat sistemi ile eşleyebilir ve farklı bir koordinat sistemi ve dönüşüm fonksiyonu ile bir kaynak görüntünün piksellerine dönüştürebilir.

Günümüzde koordinat referans sistemlerini tanımlamak için EPSG kodları yaygın olarak kullanılmaktadır. (Çevre ve Şehircilik Bakanlığı, 2019). Bu çalışma kapsamında EPSG:4326 ve EPSG: 32635 yaygın olarak kullanılan kodlardır. World Geodetic System 1984 (WGS-84) için EPSG:4326 ve WGS-84 / UTM Zone 35N için EPSG: 32635 kodu kullanılmaktadır. Aşağıda Minsk bölgesine ait EPSG:32635 koordinat referans sisteminde ve EPSG:4326 koordinat referans sistemine dönüştürülmüş görüntüler yer almaktadır.



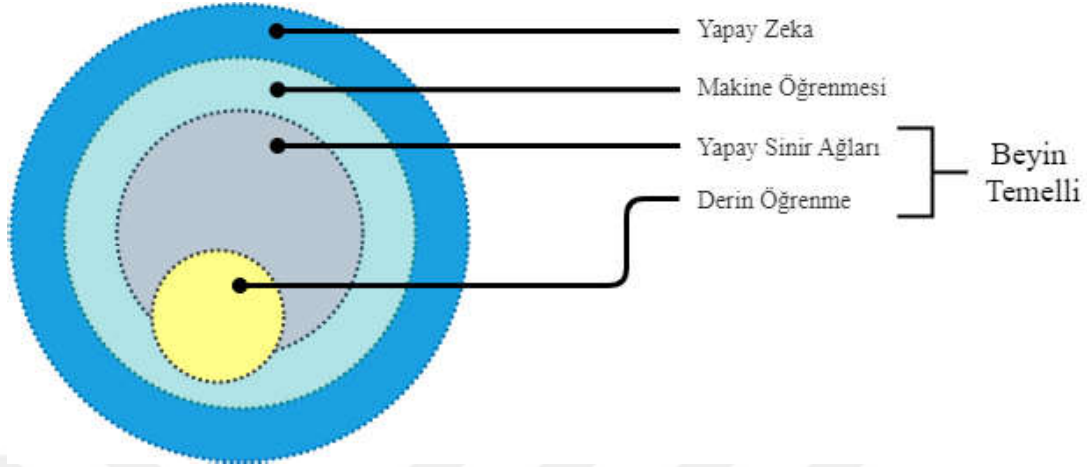
Şekil 2.7. (a) EPSG:32635 koordinat referans sistemine sahip görüntü (b) EPSG:4326 koordinat referans sistemine dönüştürülmüş görüntü

2.6. Yapay Zekâ

Yapay zekâ (YZ) kavramı son yıllarda popüler olan bir araştırma alanı olmasına rağmen ilk çıkışı oldukça eskilere dayanmaktadır. YZ terimi, ilk olarak John McCarthy tarafından 1956 yılında ortaya atılmıştır (McCarthy vd., 1956). Ancak geçmiş dönemlerdeki sınırlı hesaplama gücü ve depolama kapasitesi, donanım maliyetleri, sınırlı yeteneklere sahip algoritmalarından dolayı YZ çalışmaları uzun bir süre çok yavaş ilerlemiştir. 1950'lerde yapılan ilk YZ araştırmaları, problem çözme ve sembolik yöntemler gibi konuları araştırdı. 1960'lı yıllarda ABD Savunma Bakanlığının YZ çalışmalarına yönelmesi ile insanlara ait yeteneklerin taklit edilmesi ile ilgili çalışmalar başlamıştır. Yapılan bu erken çalışmalar, insan yeteneklerini tamamlamak ve güçlendirmek için tasarlanabilecek karar destek sistemleri ve akıllı arama sistemleri de dahil olmak üzere, bugün bilgisayarlarda görülen otomasyon ve biçimsel akıl yürütmenin yolunu açmıştır (Gilbert, 2018, s. 9,27).

YZ ile ilgili çok sayıda tanım mevcuttur. Buna göre YZ karmaşık sorunları çözmek için insan ve diğer hayvanların akıllı davranışlarına dayanan yöntemlerin kullanılmasını içerir (Coppin, 2004, s. 4). Bir diğer tanıma göre YZ, tipik olarak insan zekasını gerektiren görevleri gerçekleştirebilecek akıllı makineler oluşturmakla ilgilenen geniş kapsamlı bir bilgisayar bilimi dalıdır (Cui, 2020, s. 6). YZ ile ilgili tanımlamalar göz önüne alındığında YZ'nin temel amacı, makinelerin deneyimlerden öğrenmesini, yeni girdilere uyum sağlamasını ve insan benzeri işler yapmasını mümkün kılmaktır. Diğer bir ifade ile YZ, insan zekasının, bir makinenin kolayca taklit edebildiği ve en basitinden daha karmaşık olanlara kadar farklı zorluktaki görevleri yerine getirebilecek şekilde tanımlanması ilkesine dayanır. YZ'nin temel hedefleri öğrenme, muhakeme ve algıdır (Russell ve Norvig, 2009, ss. 1–5).

Şekil 2.8'de de görüldüğü gibi YZ günümüzdeki makine öğrenmesi ve YZ teknolojileri de dahil olmak üzere bilgisayarları zeki makinelere dönüştürmeye çalışan tüm uygulamaların temelini oluşturmaktadır.



Şekil 2.8. Yapay zekâ ve yapay zekadan türetilen yeni yaklaşımlar

Bu çalışmada konu başlıkları Şekil 2.8'deki yapı dikkate alınarak sunulmaktadır.

2.6.1. Yapay zekâ uygulamaları

YZ'nin uygulama alanları oldukça geniştir. Üstelik geçmiş dönemlerde sadece insan ile özdeşleşmiş birçok uygulama günümüzde kademeli olarak yapay zekaya sahip makine sistemlerine bırakılmıştır.

YZ başta bilgisayar sistemleri olmak üzere, tarım, eğitim, finans, sağlık sektörü, medya ve pazarlama, askeriye, kültür sanat, çevrimiçi müşteri servisleri, elektronik-otomasyon sistemleri, telekomünikasyon sistemleri, eğlence ve oyunlar, ulaşım gibi birçok alanda yaygın olarak kullanılmaktadır. Her bir alan için çok sayıda örnek vermek mümkün olsa da bu bölümde üç adet örnek verilmiştir.

Tarım alanında YZ aktif bir şekilde kullanılmaktadır. Tarımda yapay zekanın kullanımı verimi artırmaktadır. Örneğin YZ herhangi bir meyve veya sebzenin olgunlaşması ve toplanmaya hazır hale gelmesi için geçen süreyi tahmin etmekte ve böylece tarımın verimliliğini artırmaktadır. Mahsul ve toprak izleme ile tarlada toplanan ürünlerin sağlık durumlarını, kalitelerini yönetmek ve izlemek için çeşitli YZ algoritmaları kullanılmaktadır (Jha vd., 2019).

YZ eğitim alanında da yaygın bir şekilde kullanılmaktadır. YZ tabanlı yardımcı öğretmenler, öğrencilerin ihtiyaç duydukları konular ile ilgili ekstra birebir yardım almalarını sağlamaktadır. Bu durum iletişim kurmada zorlanan birçok öğrenci için büyük bir fırsattır. Gelecekteki sınıflarda, çevre bilişimi yararlı bir rol oynayabilir. Çevre bilişim, bilginin çevrenin her yerinde olduğu ve teknolojilerin otomatik olarak kişisel

tercihlere göre ayarlandığı fikridir. Diğer yandan YZ tabanlı çalışma cihazları, öğrencilerin ihtiyaçlarına göre dersler, problemler ve oyunlar oluşturabilir ve anında geri bildirim verebilir (Schiaffino vd., 2008).

YZ'nin bir diğer kullanım alanı da finanstır. Olağandışı banka kartı kullanımı ve büyük hesap mevduatı gibi bankacılık ve finans alanındaki faaliyetleri tespit etmek ve bankanın dolandırıcılık departmanına yardımcı olmak amacıyla yoğun şekilde kullanılmaktadır.

2.6.2. Yapay zekanın sınıflandırılması

YZ, zayıf ve güçlü olmak üzere iki farklı kategoriye ayrılabilir. Zayıf veya dar YZ, belirli veya dar bir alanla sınırlı bir makine zekasıdır. Zayıf YZ, belirli bir işi yapmak için tasarlanmış bir sistemi bünyesinde barındırır. Yapay zekadaki zayıf yöntemler, mantık, otomatik akıl yürütme ve çok çeşitli sorunlara uygulanabilen ancak çözülmekte olan sorunun dünyası hakkında gerçek bir bilgi içermesi gerekmeyen diğer genel yapılar gibi sistemleri kullanır (Coppin, 2004, s. 5). Satranç oynayan bilgisayar, video oyunlar, Amazon tarafından geliştirilen Alexa ve Apple tarafından geliştirilen Siri zayıf YZ uygulamalarına birer örnektir (Coppin, 2004, s. 5; Reza Tizhoosh ve Pantanowitz, 2018).

Güçlü YZ, insan zekasına eşit teorik bir makine zekâsı şeklindedir. Güçlü YZ'nin temel özellikleri arasında akıl yürütme, bulmaca çözme, yargılama, planlama, öğrenme ve iletişim kurma becerisi vardır. Aynı zamanda bilincine, nesnel düşüncelere, öz-farkındalığa, duyarlılığa ve mutluluğa sahip olmalıdır. Güçlü YZ sistemleri, insan gibi olduğu düşünülen görevleri yerine getiren sistemlerdir. Bunlar daha karmaşık sistemler olma eğilimindedir. Bir kişinin müdahalesi olmadan problem çözmesi gerekebilecek durumları ele almak için programlanmıştır. Bu tür sistemler, kendi kendini süren otomobiller gibi uygulamalarda veya hastane ameliyathanelerinde bulunabilir (Coppin, 2004, s. 5; Reza Tizhoosh ve Pantanowitz, 2018).

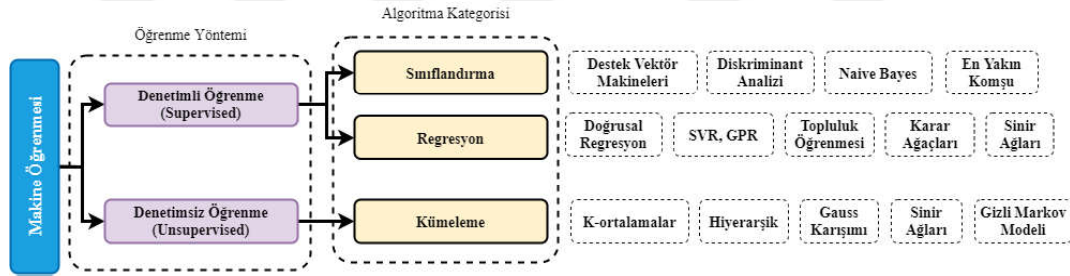
2.7. Makine Öğrenmesi

Makine öğrenmesi yapay zekanın çok önemli bir parçası olup (Coppin, 2004, s. 267) kavramsal olarak ilk defa 1959'da Arthur Samuel tarafından ortaya atılmış bir terimdir (Samuel, 1967). Makine öğrenmesi, insan ve biyolojik bilgi işleme görevlerini taklit edebilen, anlayabilen ve yardımcı olabilecek veri odaklı yöntemlerin incelenmesidir. Makine öğrenmesi temel olarak büyük veri kümelerinden anlamlı veriler

elde etmekle ilgilidir (Barber, 2011, s. 302). Bu yöntemde kullanılan algoritmalar, görevi gerçekleştirmek için klasik programlama yerine eğitim verisinden anlam çıkararak öngörülerde bulunma veya karar verme yeteneklerine sahip matematiksel bir model oluşturmaktadırlar. Bu algoritmalar, görevi etkin bir şekilde gerçekleştirmek için geleneksel bir algoritma geliştirmenin zor veya olanaksız olduğu, e-posta filtreleme ve bilgisayarlı görü gibi çeşitli uygulamalarda kullanılır (Bhowmick ve Hazarika, 2018). Makine öğrenmesi, bilgisayarları kullanarak öngörülerde bulunmaya odaklanan hesaplama istatistikleriyle yakından ilgilidir.

2.7.1. Makine öğrenmesi algoritmaları

Makine öğrenme algoritmaları genellikle denetimli, denetimsiz ve yarı denetimli (takviyeli) öğrenme olmak üzere üç kategoriye ayrılabilir (Raschka ve Mirjalili, 2017, s. 2). Her bir algoritmanın kendine göre üstün ve zayıf noktaları vardır. YZ veya makine öğrenmesi uygulamalarında doğru modeli seçmek modelin başarısı etkileyen önemli faktörlerdendir.



Şekil 2.9. Makine Öğrenmesi algoritmaları

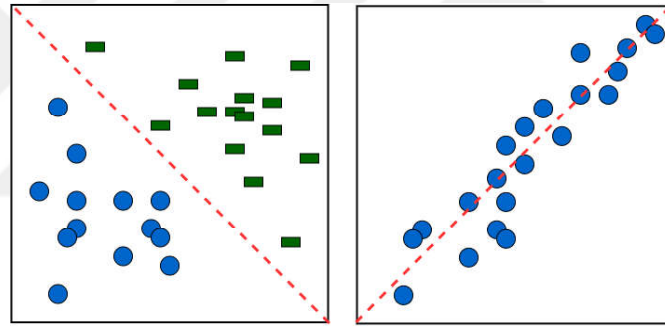
2.7.1.1. Denetimli (supervised) öğrenme

Denetimli öğrenme, bir girdiyi daha önce kendisine verilen girdi-çıkış çiftlerine dayanarak doğru çıktıya eşleyen fonksiyonun veya matematiksel modelin öğrenilmesidir. Denetimli öğrenmede, sisteme giriş ve çıkış değerlerini birlikte içeren eğitim verisi verilir. Sistem öğrenme sürecine başladıktan sonra ürettiği sonuç değerlerini giriş değerleri ile karşılaştırır. Elde edilen hatalar sistem üzerinden geri yayılır yani sisteme geri iletilir ve sistemin ağı kontrol edip ağırlıkları ayarlaması sağlanır. Bu döngü, en uygun ağırlıklar tespit edilinceye kadar aynı veri setinin yeniden işlenmesi ile sürekli devam eder. Sonuçta model yeni girdiler için sınıf etiketlerini doğru bir şekilde tahmin edebilecek bir çıkarım fonksiyonu üretir. Makine öğrenmesinde denetimli öğrenme yaygın bir şekilde kullanılmaktadır.

Denetimli öğrenme ise rastgele bir x vektörünü ve bu vektördeki örnekler ile ilişkilendirilmiş değer veya vektör olan y 'yi ele alır. Buradan x verilerini kullanarak y değerlerini öngörmeye çalışır. Öngörü genellikle $P(y | x)$ olasılığının hesaplanması ile yapılır (Goodfellow vd., 2016, s. 102).

Denetimli öğrenme problemleri regresyon ve sınıflandırma olarak ikiye ayrılabilir. Sınıflandırma problemi, çıktı değişkeninin *kırmızı* veya *mavi*, *hastalıklı* veya *hastalık yok* gibi bir kategori olduğu durumları ifade etmektedir. Regresyon problemi ise, çıktı değişkeninin *dolar* veya *ağırlık* gibi gerçek bir değer olması durumudur. Regresyon özellikle tahmin uygulamalarında yaygın bir şekilde kullanılmaktadır.

Şekil 2.10'da sınıflandırma ve regresyon problemleri arasındaki temel fark görülmektedir.



(a) Sınıflandırma (b) Regresyon

Şekil 2.10. Sınıflandırma ve regresyon problemlerinin görsel olarak ifade edilmesi

2.7.1.2. Denetimsiz (unsupervised) öğrenme

Denetimsiz öğrenme algoritmaları, birçok öznitelik içeren bir veri setini deneyimledikten sonra bu veri setinin yapısındaki faydalı özellikleri insan müdahalesi olmadan öğrenir (Coppin, 2004, s. 285). Denetimsiz öğrenmede önceden tanımlanmış bir çıktı ve dolayısıyla denetleyici yoktur, sadece giriş verileri vardır. Denetimsiz öğrenmenin amacı, girdideki düzenlilikleri bulmak, veri yapısını öğrenmektir. Denetimsiz öğrenmenin bir yöntemi kümelemedir ve burada amaç giriş verilerini kümelemek veya gruplamaktır (Alpaydin, 2016, s. 112).

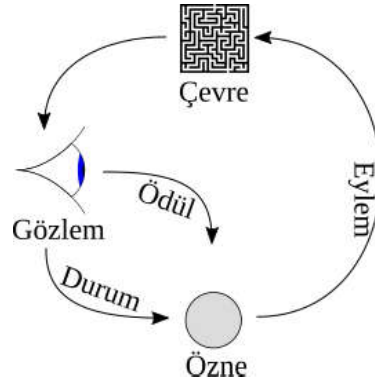
Algoritma, yapması gereken gruplamayı veya kümelemeyi verilen ağırlık katsayılarından anlar. Güncelleme tamamlandığında ağ yapısının öğrenmesi de tamamlanmış olmaktadır. Kısaca, denetimsiz öğrenme rastgele bir x vektörünü alır ve bu vektördeki farklı örnekleri deneyimler. Daha sonra doğrudan veya dolaylı olarak veri

kümesinin olasılık dağılımını $p(x)$ değerini veya bu dağılımın farklı özelliklerini öğrenmeye çalışır (Goodfellow vd., 2016, s. 102).

Denetimsiz öğrenme önemli bir araştırma alanıdır çünkü etiketlenmemiş verilerin bulunması çok daha kolay ve ucuzdur. Konuşma tanıma için bir radyo istasyonu, etiketlenmemiş konuşma verilerinin kaynağıdır. Temel amaç, etiketlenmemiş verilerden temel özellikleri elde etmek ve tipik olanı öğrenmek, daha sonra farklı amaçlar için bu veriyi etiketlemektir (Alpaydin, 2016, s. 117,118).

2.7.1.3. Yarı denetimli (reinforcement) öğrenme

Yarı denetimli öğrenme yönteminin kullanımı yaygın değildir. Bu öğrenme stratejisinde öğrenen sisteme bir denetmen yardımcı olur. Bu yöntemi kullanan bir sistem, doğru bir sonuç ürettiğinde pozitif bir takviye ve yanlış bir sonuç ürettiğinde negatif bir takviye ile bilgilendirilir. Örneğin, bir robotik ajan, bir nesnenin nasıl alınacağını öğrenmek amacıyla bu yöntemi kullanabilir. Nesneyi başarılı bir şekilde aldığında, olumlu bir takviye alır. Öğrenme sistemine görevini doğru bir şekilde yerine getirdiğinde sağlanan bilgiler, ona neden ya da nasıl doğru bir şekilde gerçekleştirdiğini söylemez (Coppin, 2004, s. 286). Şekil 2.11'de yarı denetimli öğrenme modelinin yapısı görülmektedir.



Şekil 2.11. Yarı denetimli öğrenme modeli yapısı (Wikimedia Commons, 2019b)

2.7.2. Model seçimi ve hiper parametreler

Makine öğrenmesinde kullanılabilecek çok sayıda farklı model vardır. Uygun ve doğru olan modelin seçimi daha iyi sonuçlar alınmasına katkı sağlayacaktır. Makine öğrenmesi uygulamalarında genellikle birkaç aday modeli değerlendirdikten sonra nihai modele karar verilir. Bu süreç model seçme veya model belirleme olarak ifade edilmektedir.

Model seçimi bazen doğal yapısı birbirinden tamamen farklı olan modeller arasında yapılırken (örneğin karar ağaçları ile doğrusal model gibi), bazen de farklı hiper parametre ayarlarıyla eğitilmiş aynı model sınıfının üyeleri arasında yapılabilmektedir. Örneğin, çok katmanlı algılayıcılar ile modelleri farklı sayıda gizli katman, farklı sayıda gizli birim ve her gizli katmana uygulanan etkinleştirme işlevlerinin çeşitli seçenekleriyle karşılaştırmak mümkündür. Aday modeller arasında en iyisini belirlemek için, genellikle bir doğrulama seti kullanılır.

Bir makine öğrenmesi modeli, verilerden öğrenilmesi gereken bir dizi parametreye sahip bir matematiksel formülün tanımıdır. Modelin eğitilmesi sürecinde eğitim verileri kullanarak model için en uygun parametrelerin belirlenmesi amaçlanmaktadır. Parametreler makine öğrenme algoritmalarının anahtarıdır. Bunlar modelin eğitim verilerinden öğrenilen kısımdır. Model parametresi, modelin içinde yer alan ve değeri verilerden tahmin edilebilecek bir yapılandırma değişkenidir (B. Wang ve Gong, 2018). Klasik makine öğrenmesi literatüründe, modeli hipotez ve parametreleri hipotezin belirli bir veri kümesine uyarlaması olarak düşünülebilir.

Makine öğrenmesinde model parametreleri ve model hiper parametreleri olmak üzere iki temel parametre türü vardır. Model parametrelerinin temel özellikleri aşağıdaki gibidir:

- Tahmin yaparken model tarafından kullanılırlar.
- Değerler, modelin problem üzerindeki yeteneğini tanımlar.
- Verilerden tahmin edilir veya öğrenilirler.
- Genellikle uygulayıcı tarafından manuel olarak ayarlanmazlar.
- Genellikle öğrenilen modelin bir parçası olarak kaydedilirler.

Model parametresi, modelin içinde yer alan ve değeri verilerden tahmin edilebilecek bir yapılandırma değişkenidir. Model hiper parametre ise, modelin dışında olan ve değeri verilerden tahmin edilemeyen bir yapılandırma değişkenidir. Hiper parametrelerin temel özellikleri aşağıdaki gibidir.

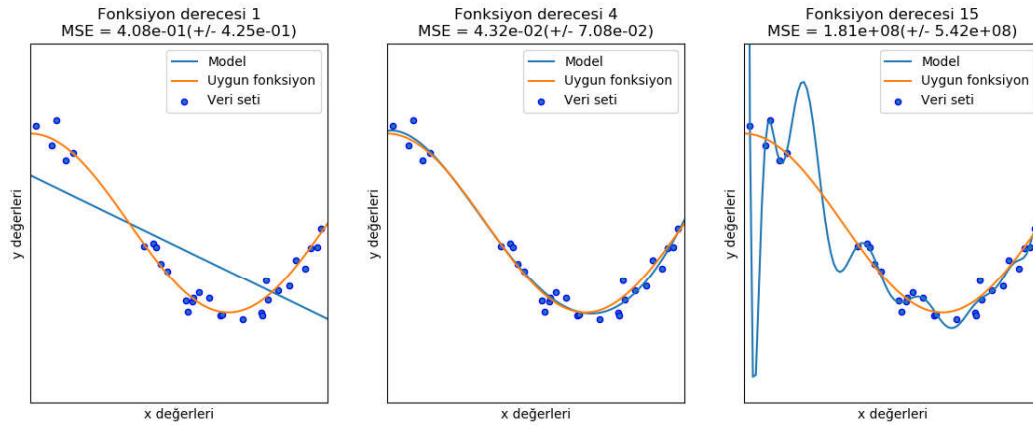
- Model parametrelerini tahmin etmeye yardımcı olmak için genellikle işlemlerde kullanılırlar.
- Genellikle uygulayıcı tarafından belirlenirler.
- Genellikle sezgisel tarama kullanılarak ayarlanabilirler.

- Genellikle belirli bir modelleme problemi için modele özgü ayarlanırlar.

Belirli bir problemde bir model için en iyi hiper parametreler önceden bilinemez. Temel hiper parametre belirleme kuralları kullanabilir, benzer modellerde kullanılan değerlerden kopyalanabilir veya deneme yanılma yoluyla en iyi değer atanmaya çalışılır (Brownlee, 2017).

2.7.3. Aşırı, yetersiz ve yeterli öğrenme

Bir makine öğrenmesi ya da sinir ağı eğitilirken, giriş değerlerinin çıktı değerleriyle matematiksel bir eşleminin yapılabilmesi için ağırlıkları ve bias değerlerini optimize etmesi gerekir. Bu amacın yanında ikinci amaç ise eğitim tamamlandıktan sonra modelin henüz görülmemiş veriler üzerinde eğitim verisi kadar iyi performans göstermesini sağlamaktır. Oldukça karmaşık ve çok sayıda adımdan oluşan makine öğrenmesi uygulamalarında sıklıkla karşılaşılan hatalardan ikisi aşırı öğrenme ve yetersiz öğrenmedir. Aşırı öğrenme daha fazla karşılaşılan bir sorundur. Şekil 2.12’de bu kavramlar görsel olarak verilmiştir. Buna göre soldaki grafikte model bir dereceli oldukça basit sayılabilecek doğrusal bir fonksiyon ile ifade edildiğinden veri setini uygun bir biçimde temsil edememiştir. Kısaca yetersiz öğrenmiştir. İkinci grafikte model dört dereceli bir fonksiyon ile tanımlanmış olup veri setini uygun bir biçimde temsil etmiştir. Sağdaki grafikte ise fonksiyonun derecesi 15’e çıkarılmıştır. Bu durumda ise elde edilen model aşırı öğrenmiştir. Dolayısıyla bu veri setinde daha önce karşılaşmadığı verileri temsil etmesi zordur.



Şekil 2.12. Yetersiz öğrenme, yeterli öğrenme ve aşırı öğrenme işlemlerine ait örnek grafikler (Pedregosa vd., 2011)

Aşırı öğrenme, bir modelin bir veriyi öğrenmesinden daha öteye geçerek veriyi ezberlemesidir. Buradaki ezberden kasıt modelin eğitim setindeki verileri oldukça detaylı bir şekilde öğrenmesi ve modeli buna uyacak şekilde genellemesidir (Bonaccorso, 2017, s. 27,28). Bu durum genellikle modelin eğitim verilerine çok fazla maruz kalmasından kaynaklanır. Aşırı öğrenmenin ortaya çıkmasının diğer nedenleri ise eğitim veri setindeki gürültü ve ağırların gereğinden fazla katman veya nöron kullanılarak eğitilmesidir. Eğitim sürecinde çok iyi sonuçlar elde edilse bile model ilk defa karşılaştığı veriler için aynı başarıyı gösteremez. Aşırı öğrenmeyi önlemek için en iyi çözüm daha fazla eğitim verisi kullanmaktır. Bunun yanında daha az nöron veya katman kullanarak modelin karmaşıklığını azaltmak aşırı öğrenmeyi ortadan kaldıracaktır (Goodfellow vd., 2016, ss. 428–431). Erken durma, L1, L2, dropout, eğitim verilerini arttırma ve toplu normalleştirme gibi düzenleme teknikleri aşırı öğrenmeyi engelleyen diğer yöntemlerdir.

Yetersiz öğrenme aşırı öğrenmenin tersidir. Bir makine öğrenmesi veya derin öğrenme modelinin bir veri kümesinin özellikleri ile hedef değişken arasındaki ilişkileri doğru bir şekilde yakalayacak kadar temel bir matematiksel haritalama yapamadığında yetersiz öğrenme gerçekleşir. Diğer bir ifadeyle modelin, verinin temel eğilimini yakalayamadığı durumlarda yetersiz öğrenme meydana gelir (Bonaccorso, 2017, s. 27,28). Yetersiz öğrenmeyi ortadan kaldırmak aşırı öğrenmeye göre çok daha kolay ve basittir. Yetersiz öğrenme, bir modelin düşük karmaşıklığının bir sonucu olduğundan, tek yapılması gereken modelin karmaşıklığını arttırmaktır. Genel olarak bir fonksiyonun karmaşıklığı artırılmak isteniyorsa yapılması gereken parametre sayısını arttırmaktır. Sinir ağlarındaki parametreler ağırlık ve bias değeridir. Ek ağırlık ve bias değerleri eklemek için sinir ağındaki katman sayısı ve nöron sayısı artırılmalıdır (Goodfellow vd., 2016, ss. 428–431).

İdeal olarak, modelin sıfır hata ile tahminlerde bulunduğu durumun verilere iyi uyduğu söylenir. Buna yeterli öğrenme denir. Yeterli öğrenme, aşırı öğrenme ve yetersiz öğrenme arasında bir noktada elde edilebilir. Bu hedefe ulaşmak için eğitim sırasında bir modelin performansı gözlemlenmelidir. Performansı ölçmenin en kolay yolu, eğitim sırasında oluşan hatayı (error) veya kayıp (loss) fonksiyonunun değerini hesaplamaktır. Hata / kayıp, eğitim seti için periyodik olarak ve tüm eğitim süresi boyunca bir doğrulama

seti olarak hesaplanır. Zamanla, algoritma öğrenirken, modelin eğitim verileri ve doğrulama verileri üzerindeki hata / kayıp oranı azalacaktır.

Eğitimde hesaplanan hata azalmaya devam etse de doğrulama veri kümesinde hesaplanan hata tekrar artabilir. Bu durum tam olarak bu aşırı öğrenmeden kaynaklanan bir sonuçtur. Model, eğitim verilerindeki gürültüyü ve alakasız modelleri öğrenmek için yeterli zamana sahip olduğundan, modeli eğitmek için kullanılmayan doğrulama verileri üzerinde genelleme yapma yeteneğini kaybeder.

2.8. Yapay Sinir Ağları (Artificial Neural Networks)

Yapay sinir ağları, memelilerde beyin korteksinin çalışmasından ilham alınarak tasarlanmıştır. Bununla birlikte, bu modelin insan beyninin çalışmasına, ölçeğine ve karmaşıklığına yakından benzemediğini, aralarında ciddi farklar olduğunu ifade etmek gerekir (Khan vd., 2018). Bir YSA, biyolojik beyindeki nöronları zayıf bir şekilde modelleyen yapay nöronlar adı verilen bağlı birimler veya düğümler koleksiyonuna dayanır. İnsan beyni, nöron adı verilen çok sayıda işlem biriminden oluşur ve her nöron, sinaps adı verilen bağlantılarla çok sayıda başka nöronlara bağlanır. Nöronlar paralel olarak çalışır ve bu sinapslar üzerinden kendi aralarında bilgi aktarırlar. İşlemin nöronlar tarafından yapıldığı ve hafızanın sinapslarda olduğu, yani nöronların birbirine bağlandığı ve birbirlerini etkilediği düşünülmektedir (Alpaydin, 2016, s. 178).

YSA'nın amacı, deneyimle bilinmeyen bir matematiksel fonksiyonu öğrenmektir. Yani bu tür sistemler, genellikle klasik programlama anlayışında olduğu gibi göreve özgü kurallarla programlanmadan, kendisine verilen örnekleri göz önünde bulundurarak görevleri gerçekleştirmeyi öğrenirler. Örneğin, bir nesne tanıma uygulamasında YSA modeli *insan* veya *insan yok* olarak etiketlenmiş örnek görüntüleri analiz eder. Analiz sonucunda elde ettiği sonuçları kullanarak diğer görüntülerdeki insanları tespit etmeyi öğrenebilir. YSA bu işlemi insanlarla ilgili ön bir bilgisi olmadan yapmakta, insana ait ağız, burun, göz, el, kol, vücut şekli, yaklaşık boy gibi tanımlayıcı özellikleri (öznitelikleri) otomatik olarak kendisi oluşturmaktadır.

YSA ile ilgili ilk çalışmalar 1940'lı yıllara dayanmaktadır. McCulloch ve Pitts (1943) insan sinir sistemin çalışma mantığını esas alarak beyin fonksiyonlarının işleyişini mantıksal olarak hesaplayan bir model ortaya koymuşlardır. Bu model, insan sinir sisteminin bazı ana unsurlarını referans alarak zeki sistemler oluşturmaya çalışan

YSA'nın temelini oluşturmaktadır. İlerleyen süreçte farklı bilim adamları tarafından Perceptron (algılayıcı) (Rosenblatt, 1957), Adaptive linear element (ADALINE) (Widrow ve Hoff, 1960) gibi modeller geliştirilmiştir. Minsky ve Papert (1969), YZ araştırmalarının seyrini yıllarca değiştirecek tarihi bir metin olan Perceptrons'u yayımladı. Metinde, Minsky ve Papert tek bir algılayıcının özel veya (XOR) işlevini öğrenemediğini kanıtladı. Bu bilgi biraz daha genelleştirilerek algılayıcı ve ADALINE gibi modellerin doğrusal olmayan problemlerin çözümünde yetersiz olduğunu kanıtlamıştır. Ortaya çıkan bu durum o dönemde YSA ile ilgili yapılan çalışmaların azalmasına neden olmuştur (Inik ve Ülker, 2017).

1980'lerde, sinir ağı araştırmalarının ikinci dalgası, büyük ölçüde, bağlantıcılık ya da paralel dağıtılmış işlem denen bir hareketle ortaya çıkmıştır (Rumelhart ve MacClelland, 1986). Bağlantıcılık bilişsel bilim bağlamında ortaya çıkmıştır. Bilişsel bilim, birçok farklı analiz düzeyini birleştirerek zihni anlamak için disiplinler arası bir yaklaşımdır. 1980'lerin başında, çoğu bilişsel bilim insanı sembolik akıl yürütme modelleri üzerinde çalıştı. Popülerliklerine rağmen, sembolik modellerin beyin tarafından sinir hücreleri kullanılarak nasıl ortaya çıktıklarını açıklamak zordu. Yine de bu yıllarda yapılan çalışmalar bugünün derin öğrenme kavramı için önemli temelleri atmış oldu (D. Li ve Du, 2017, s. 20,21).

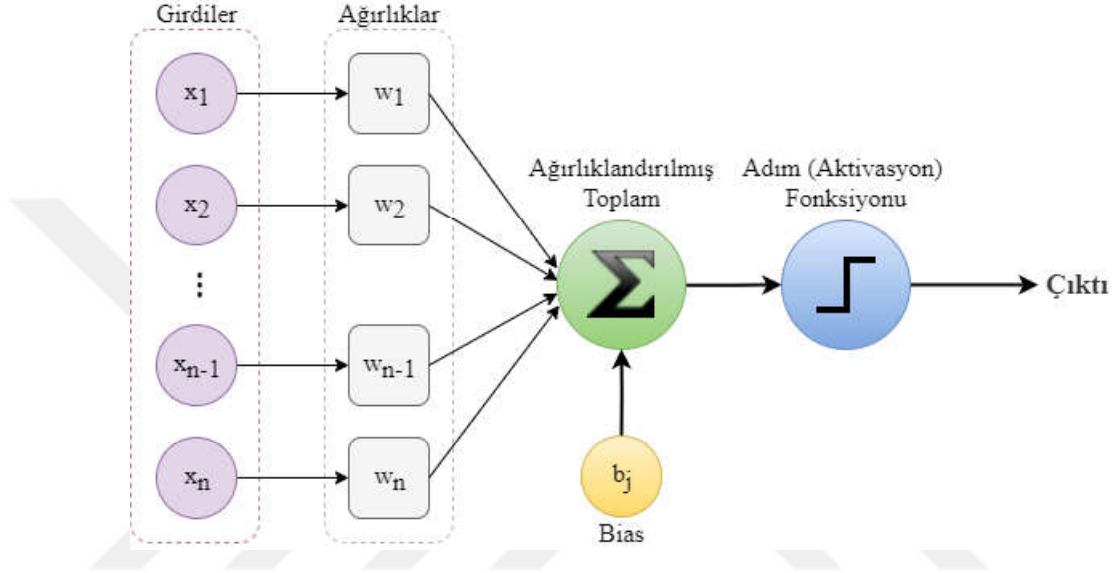
YSA yaklaşımının asıl amacı, problemleri bir insan beyninin çalışma yapısını taklit ederek çözmektir. Bununla birlikte, zaman içinde, belirli görevlerin yerine getirilmesine dikkat edilerek biyolojiden sapmalar ortaya çıkmıştır. Günümüzde YSA'lar, bilgisayarlı görü, konuşma tanıma, makine çevirisi, sosyal ağ filtreleme, video oyunları, etkileşimli simülasyonlar, tıbbi tanı ve hatta resim çizme gibi geleneksel olarak insanlarla özdeşleşmiş uygulamalar dahil olmak üzere çeşitli görevlerde kullanılmaktadır (Gatys vd., 2016).

2.8.1. Algılayıcı (perceptron)

Algılayıcı kavramı yapay sinir ağının temelini oluşturmaktadır. Gümümüzdeki gelişmiş sinir ağları halen bu modele dayandığı için bu kavram hakkında biraz bilgi vermek gerekmektedir. Algılayıcı, yapay nöronlar ve sinaptik bağlantılardan oluşan bir ağıdır, burada her bir nöronun bir aktivasyon değeri vardır (Alpaydin, 2016, s. 86,87). Tek katmanlı ve çok katmanlı olmak üzere iki tür algılayıcı vardır. Çok katmanlı algılayıcı derin öğrenmenin temelini oluşturan bir yapıdır. Çok katmanlı algılayıcı gidileri çıktılara

eşleyen matematiksel bir fonksiyondur. Bu fonksiyon birden fazla basit fonksiyonun bir araya gelmesi ile meydana gelir (Goodfellow vd., 2016, s. 5).

Şekil 2.13'te bir algılayıcının yapısı görülmektedir. Burada $x_1, x_2, x_3, \dots, x_n$ giriş değerlerini, $w_1, w_2, w_3, \dots, w_n$ bu giriş değerlerine ait ağırlıkları ifade etmektedir.



Şekil 2.13. Algılayıcı yapısı

Algılayıcının çıkış değeri Denklem 2.1 ile hesaplanır.

$$y_s = \sum_{i=1}^n x_i w_i + b_j \quad (2.1)$$

Burada y_s ağırlıklandırılmış toplamı, x_i giriş değerini, w_i ağırlık değerini, n giriş değerlerinin sayısını ve b_j bias değerini ifade etmektedir. Sonuç itibarı ile Denklem 2.1 ile hesaplanan ağırlıklandırılmış toplam aktivasyon fonksiyonuna gönderilir. Burada hesaplanan değer, aktivasyon fonksiyonunda belirlenen eşik değerden büyükse fonksiyon ateşlenecek yani nöron bir değer almaktadır.

Rosenblatt (1957), çıktıyı hesaplamak için Denklem 2.2'deki gibi basit bir kural önerdi. İlgili girdilerin önemini diğer bir ifade ile ağırlığını belirlemek için $w_1, w_2, w_3, \dots, w_n$ gibi gerçek sayıların kullanılmasını önerdi. Nöronun çıkışı 0 veya 1, ağırlıklı toplam eşik değerinden daha küçük veya daha büyük olup olmadığına göre belirlenir. Tıpkı ağırlıklar gibi, eşik de nöronun bir parametresi olan gerçek bir sayıdır.

Daha kesin cebirsel terimlerle algılayıcı aşağıdaki gibi ifade edilebilir. Denklem 2.1 ile elde edilen algılayıcının sonuç değerini temsil etmek üzere

$$f(x) = \begin{cases} 0 & \text{eğer } y_s \leq \text{eşik değeri} \\ 1 & \text{eğer } y_s > \text{eşik değeri} \end{cases} \quad (2.2)$$

Buna göre eğer ağırlıklandırılmış toplam değer belirlenen eşik değerinden büyükse fonksiyon 1 değerini, diğer durumlarda ise 0 değerini almaktadır.

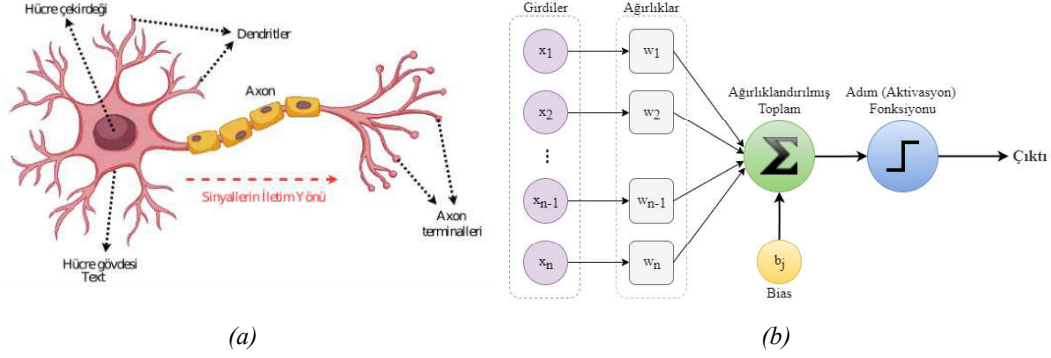
2.8.2. Bir yapay sinir ağı modeli

YSA'lar, geleneksel algoritmalar ile geliştirilen uygulamaların başaramadığı işleri gerçekleştirmek için insan beyninin mimarisinden faydalanma girişimi olarak başladı. İlerleyen süreçte ise daha iyi sonuçlar elde etmek amacıyla YSA'larda biyolojik sinir ağından farklı bazı özellikler ortaya çıkmıştır. Şekil 2.14'te biyolojik bir sinir ağı ile bu biyolojik ağdan esinlenilerek geliştirilmiş YSA görülmektedir.

Bir yapay sinir ağı modelini daha iyi açıklamak için öncelikle bu modelin esinlendiği biyolojik sinir ağı modeli hakkında gerekli bilgiler verilmelidir. İnsan beyni, nöron adı verilen çok sayıda işlem biriminden oluşur ve her bir nöron, sinaps adı verilen bağlantılarla çok sayıda diğer nörona bağlanır. Nöronlar paralel olarak çalışır ve bu sinapslar üzerinden kendi aralarında bilgi aktarırlar. İşlemlerin nöronlar tarafından yapıldığına ve hafızanın sinapslarda, yani nöronların bağlanma ve birbirini etkileme şeklinde olduğuna inanılmaktadır (Alpaydin, 2016, s. 86).

İnsanlarda, temel nöron tipleri üzerinde birçok farklılık olmasına rağmen tüm biyolojik nöronlar hücre gövdesi (soma), dendritler, akson ve sinapslar olmak üzere dört temel bileşene sahiptir. Şekil 2.14 (a) biyolojik bir nöronun yapısını ifade etmektedir.

Nöron hücresinin gövdesi (soma) çekirdeği içerir ve nöronların yaşamı için gerekli olan biyokimyasal dönüşümü gerçekleştirir. Her nöronun etrafında ince, dal benzeri yapılar vardır. Bunlara dendrit adı verilmektedir. Dendritler gelen sinyalleri kabul eder ve hücre gövdesine iletilmesinden sorumludur. Akson ise sinir hücresindeki elektriksel uyarıların daha uzağa ulaşmasını sağlayan iletim hatlarıdır. Bir aksonun sonunda sinaps adı verilen oldukça karmaşık ve gelişmiş yapılar vardır. İki nöron arasındaki bağlantı sinapslar ile gerçekleştirilir. Yapay nöron biyolojik bir nöronun dört temel ögesini simüle eder. Şekil 2.14 (b), yapay bir nöronu görsel olarak temsil etmektedir.

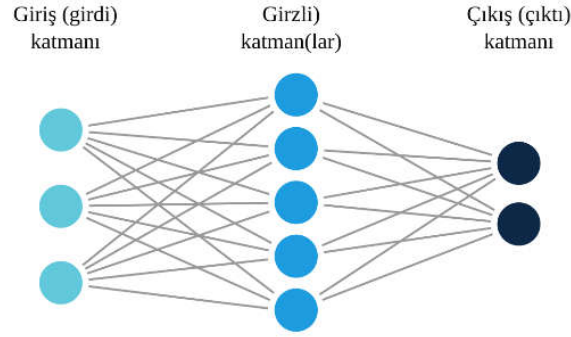


Şekil 2.14. Biyolojik ve yapay nöronlar

YSA'daki hesaplama birimlerine nöron denir. Nöronlar, temelde ağırlıklı girdiler olan sinapslarla birbirlerine bağlıdır. Şekil 2.14 (b)'de görüldüğü gibi yapay sinir ağına veri girişleri x_n sembolü ile gösterilen girdiler ile yapılmaktadır. Bu girdi değerlerinin her birisi bir bağlantı ağırlığı ile çarpılmaktadır. Ağırlıklı girdi, her bir giriş değerinin toplam çıkış değerine farklı oranda etki etmesini sağlamaktadır. Ağırlıklar w_n ile temsil edilmektedir. Bu ağırlıklandırılmış girdiler daha sonra genellikle bu değerleri toplayan toplama fonksiyonunu iletilir. Bu aşamada toplamının yanında seçilen fonksiyona göre farklı işlemler de yapılabilir. Bu işlemler, daha sonra ileriye doğru yayılan ortalama, en büyük, en küçük, OR veya AND değerleri gibi bir dizi farklı değer üretebilirler. YSA modellerinde farklı toplama fonksiyonlarının yanı sıra farklı transfer fonksiyonlarını kullanan diğer ağ yapıları oluşturmak mümkündür. Üstelik bu YSA'ların en güçlü olduğu noktalardan birisidir.

2.8.3. Yapay sinir ağı bileşenleri

Bir YSA çok sayıda bileşenden oluşmaktadır. Ağın yapısına bağlı olarak bu bileşenler farklılık göstermesine rağmen bu bölümde basit veya gelişmiş tüm YSA modellerinde yer alan temel bileşenler hakkında bilgi verilmiştir. Bir YSA ağı giriş, gizli katman ve çıktı katmanı olarak ifade edilen üç temel katmandan oluşmaktadır. Bu katmanlar görsel olarak Şekil 2.15'te verilmiştir.



Şekil 2.15. Bir yapay sinir ağını oluşturan girdi, gizli ve çıktı katmanları

Ağın tasarımına bağlı olarak bu katmanlardaki parametreler farklılık gösterebilmektedir. Aşağıda YSA bileşenleri hakkında gerekli bilgiler verilmiştir.

2.8.3.1. Girdiler (input) ve ağırlıklar (weight)

Bir nöron genellikle birçok eşzamanlı girdi alır. Her girişin, göreceli ağırlığı vardır. Ağırlıklar bir girdiye, işlem elemanının toplama fonksiyonu üzerinde ihtiyaç duyduğu etkiyi verir. Bazı girdiler, bir nöral tepki üretmek için birleştikçe işleme elemanı üzerinde daha büyük bir etkiye sahip olmak için diğerlerinden daha önemli hale getirilir. Ağırlıklar, YSA tarafından kaydedilen giriş sinyalinin yoğunluğunu belirleyen uyarlanabilir katsayılardır. Bunlar, bir girişin bağlantı gücünün bir ölçüsüdür. Bu güçlü yönler, çeşitli eğitim setlerine cevap olarak ve bir ağın belirli topolojisine veya öğrenme kurallarına göre değiştirilebilir.

2.8.3.2. Toplama fonksiyonu

Bir YSA'da girdiler ve girdilere ait ağırlıklar sırasıyla $x_1, x_2, x_3, \dots, x_n$ ve $w_1, w_2, w_3, \dots, w_n$ şeklinde ifade edilirler. Toplam giriş sinyali, bu iki vektörün işleme sokulmasından elde edilen sonuçtan üretilir. Formülü aşağıdaki gibidir (Denklem 2.3).

$$f(x) = \sum_{i=1}^n x_i w_i + b_j \quad (2.3)$$

Toplama fonksiyonu, YSA modeline bağlı olarak girdilerin ağırlık toplamından daha karmaşık olabilir. Giriş ve ağırlık katsayıları, aktivasyon fonksiyonuna geçmeden önce birçok farklı şekilde birleştirilebilir.

2.8.3.3. Aktivasyon fonksiyonları

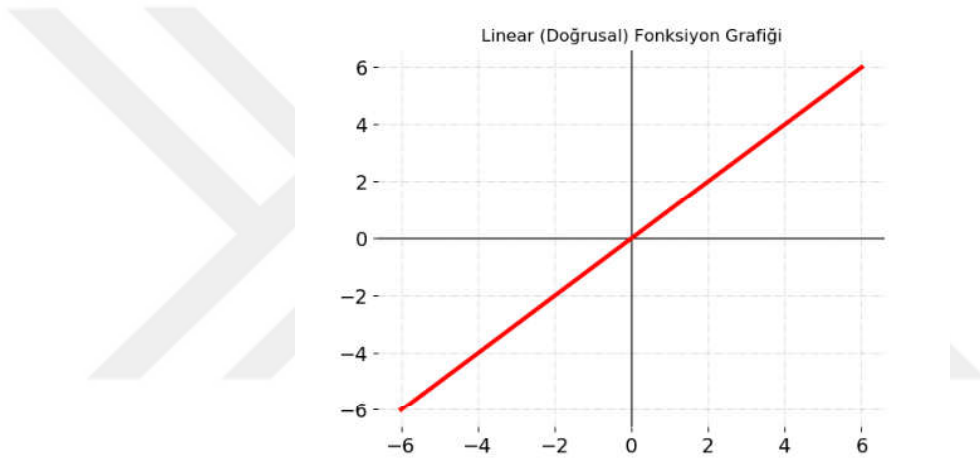
YSA'nın oldukça karmaşık bir yapıyı öğrenmesi, anlamlandırması ve girdiler ile çıktı değişkeni arasındaki doğrusal olmayan karmaşık fonksiyonel eşlemeler için aktivasyon fonksiyonları son derece önemlidir. Aktivasyon fonksiyonları ağa doğrusal olmayan özellikler kazandırır. Bu fonksiyonların temel amacı, YSA'daki bir düğümün giriş sinyalinin bir çıkış sinyaline dönüştürmektir.

YSA'da x ile ifade edilen girdi değerleri, w ile ifade edilen ağırlıkları ile birlikte aktivasyon fonksiyonu tarafından işlenir ve elde edilen çıktı bir sonraki katmana girdi olarak iletilir.

Bir YSA'da aktivasyon fonksiyonu uygulanmazsa, çıkış sinyali basit doğrusal bir fonksiyon olur. Doğrusal bir fonksiyon sadece bir derecelik bir polinomdur. Doğrusal bir denklemin çözülmesi kolaydır, ancak karmaşıklıkları sınırlıdır ve verilerden karmaşık fonksiyonel eşlemeleri öğrenmek için yeterli değildir. Aktivasyon fonksiyonu olmayan bir YSA basitçe sınırlı güce sahip ve çoğu zaman iyi performans göstermeyen bir doğrusal regresyon modeli gibi davranır. Böyle bir ağ görüntüler, videolar, ses, konuşma gibi diğer karmaşık veri türlerini öğrenemez ve modelleyemez. Bu nedenle, yüksek boyutlu, doğrusal olmayan büyük veri kümelerini anlamak için çok fazla gizli katmana sahip ve oldukça karmaşık mimarilere sahip olan derin öğrenme gibi YSA teknikleri kullanılmaktadır.

Doğrusal olmayan fonksiyonlar, birden fazla dereceye sahip ve grafiksel olarak eğrilerle ifade edilirler. YSA evrensel fonksiyon yaklaşımcılar olarak kabul edilir. Bu, YSA'nın herhangi bir fonksiyonu hesaplayabileceği ve öğrenebileceği anlamına gelir. İnsan tarafından düşünülebilen hemen hemen her süreç, yeterli büyüklükteki bir YSA'da fonksiyonel bir hesaplama olarak temsil edilebilir. Bu nedenle, ağı daha güçlü hale getirmek, karmaşık formdaki verilerden anlam çıkarmak ve bunlar arasında doğrusal olmayan karmaşık keyfi fonksiyonel eşlemeleri temsil etme yeteneği eklemek için bir aktivasyon fonksiyonuna ihtiyaç duyulur. Böylece doğrusal olmayan bir aktivasyon fonksiyonu kullanarak girdilerden çıktılara doğrusal olmayan eşlemeler yapmak mümkün hale gelmektedir. Ayrıca aktivasyon fonksiyonunun diğer bir önemli özelliği de ayırt edilebilir olması gerektirir.

Aktivasyon fonksiyonları temel olarak doğrusal (linear) ve doğrusal olmayan (non-linear) fonksiyonlar olmak üzere iki türe ayrılabilir. Doğrusal fonksiyonlar $f(x_i) = x_i$ ile ifade edilen fonksiyonlardır (Şekil 2.16). Bu fonksiyonlar özellikle karmaşık veya çok sayıda parametreye sahip olan verilerde yeterli performansı gösteremezler. Doğrusal olmayan aktivasyon fonksiyonları ise en yaygın kullanılan fonksiyon türüdür. Bu fonksiyonlar genellikle eğrilerle ifade edilirler. Bu fonksiyonlar modelin çeşitli verileri genelleştirmesini, uyarlamasını ve çıktığı ayırt etmesini kolaylaştırır (Alpaydin, 2016, s. 76,77).



Şekil 2.16. Doğrusal aktivasyon fonksiyonu

Günümüzde yaygın olarak kullanılan aktivasyon fonksiyonları aşağıda açıklanmıştır.

Sigmoid veya lojistik aktivasyon fonksiyonları

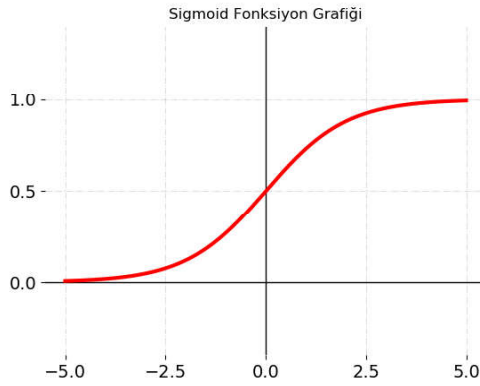
Günümüzde yaygın olarak kullanılan fonksiyonlardan birisidir. Sigmoid fonksiyonun formülü ve türevi Denklem 2.4 ve 2.5 'te verilmiştir. Lojistik fonksiyonu ise Sigmoid fonksiyonunun bir türevidir.

Hiperbolik tanjant (tanh) fonksiyonuna çok benzer, ancak (0,1) aralığında değerler alır. Fonksiyonun ve türevinin denklemleri sırasıyla Denklem 2.4 ve 2.5'te verilmiştir.

$$f(x_i) = \frac{1}{1 + e^{-x_i}} \quad (2.4)$$

$$f'(x_i) = \sigma(x_i) - (1 - \sigma(x_i)) \quad (2.5)$$

Burada e doğal üstel fonksiyonu ifade etmektedir. Bu fonksiyon özellikle çıktının bir olasılık olarak ifade edildiği modellerde kullanılır. S şeklinde bir eğri ile ifade edilir. Anlamak ve uygulamak kolaydır, ancak popülaritesinin düşmesine neden olan bazı önemli dezavantajları vardır. En büyük dezavantajı çok fazla katmandan oluşan derin ağlarda belirli bir noktadan sonra türevlenen elemanların sıfıra doğru yaklaşmasıdır (Glorot ve Bengio, 2010). Bu durum kaybolan gradient (vanishing gradient) problemi olarak ifade edilmektedir (Bengio vd., 1994). Makine öğrenmesinde kaybolan gradient problemi, yapay sinir ağlarının gradient tabanlı öğrenme yöntemleri ve geri yayılım ile eğitiminde bulunan bir zorluktur. Şekil 2.17’de sigmoid fonksiyonun grafiği görülmektedir.



Şekil 2.17. Sigmoid aktivasyon fonksiyonu (Wikimedia Commons, 2019a)

Hiperbolik tanjant (Tanh) fonksiyonu

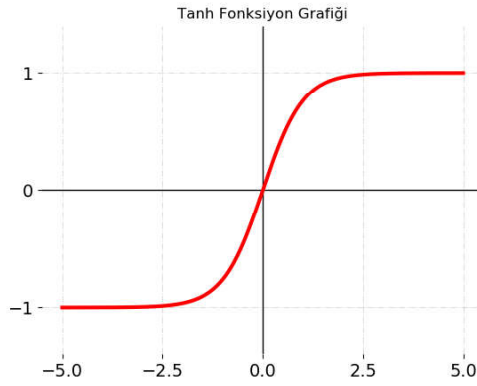
Tanh fonksiyonu, Sigmoid fonksiyonuna oldukça benzemekte ve aynı özelliklerin çoğuna sahiptir. Bu fonksiyonun türevini hesaplamak oldukça kolaydır. Bu durum işlemlerde önemli ölçüde hız ve kolaylık sağlamaktadır. Bu fonksiyonun çıktı aralığı (-1, +1)'dir. -1 ve 1 değerleri bu aralığa dahil değildir.

Fonksiyonun ve türevinin formülleri Denklem 2.6 ve 2.7’de verilmiştir.

$$f(x_i) = \tanh(x_i) \quad (2.6)$$

$$f'(x_i) = 1 - \tanh(x_i)^2 \quad (2.7)$$

Bu fonksiyon çalışma esnasında bir düğümü ateşlememek yerine ilgili düğüme negatiflik durumu atamaktadır. Ayrıca, bu fonksiyon ölçeğin pozitif (çıktı) tarafında daha geniş bir aralık sunduğu için YSA modellerinde daha ince ayarlamalar yapılabileceği anlamına gelir. Bu işlevin doğal eşiği 0'dır, yani 0'dan büyük bir değerle eşleşen herhangi bir girdinin ikili olarak yüksek (veya 1) olduğu kabul edilir. Ayrıca bu fonksiyonun pürüzsüz, türevlenebilir, monotonik ve sınırlı olması, geri yayılım algoritmasının iyi çalışacağı anlamına gelir.



Şekil 2.18. Tanh aktivasyon fonksiyonu (Wikimedia Commons, 2019a)

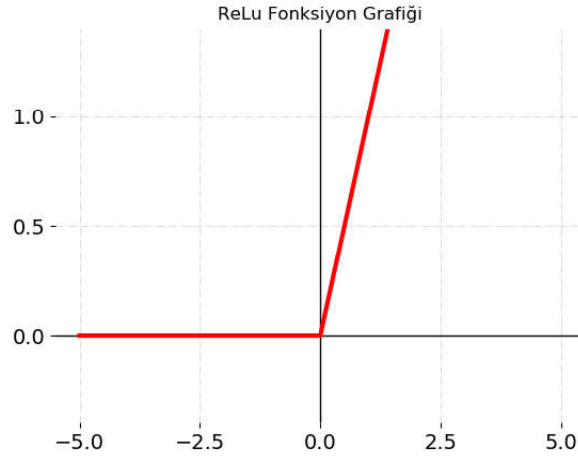
Rectified linear units fonksiyonu (ReLU)

ReLU, evrimsel sinir ağı mimarisinde yaygın olarak kullanılan aktivasyon fonksiyonlarından birisidir. ReLU, $f(x) = \max(0, x)$ şeklinde ifade edilebilir.

Denklemleri aşağıda verilmiştir (Denklemler 2.8).

$$f(x) = \begin{cases} 0, & \text{eğer } de\check{g}er < 0 \\ de\check{g}er, & \text{eğer } de\check{g}er \geq 0 \end{cases} \quad (2.8)$$

Denklemden de anlaşıldığı gibi fonksiyon negatif giriş değerleri için sıfır üretirken, sıfır veya pozitif giriş değerleri için giriş değerine eşit bir değer üretmektedir. ReLU fonksiyonundan türetilmiş Noisy ReLU, Leaky ReLU gibi başka aktivasyon fonksiyonları da mevcuttur.



Şekil 2.19. *ReLU aktivasyon fonksiyonu*

Günümüzde özellikle YSA'lar gibi gizli katmanların yer aldığı ağlarda ReLU sıklıkla kullanılmaktadır (Goodfellow vd., 2016, s. 175). Diğer yandan Sigmoid ve Tanh fonksiyonları derin sinir ağlarında modelin doğruluğunu ve performansını olumsuz etkilediği için kullanılması önerilmemektedir.

Yukarıda verilen fonksiyonların dışında başka aktivasyon fonksiyonları da mevcuttur. Bu fonksiyonların kullanımları yaygın olmadığı veya bu çalışma kapsamına doğrudan dahil olmadıkları için daha fazla ayrıntı verilmemiştir.

2.8.4. Kayıp (loss) fonksiyonu

Bir YSA ağında gizli katmanlar ve aktifleştirme fonksiyonları tanımladıktan sonra, kayıp fonksiyonu ve optimize edicinin (optimizer) belirlenmesi gerekir. Diğer yandan derin öğrenme uygulamalarında uygun modelin belirlenmesi temelde bir optimizasyon uygulamasıdır. Optimizasyon algoritmasının bir parçası olarak, modelin geçerli durumu için hata arka arkaya tahmin edilmelidir. Bu, geleneksel olarak bir kayıp fonksiyonu olarak adlandırılan bir hata fonksiyonunun seçimini gerektirir. Hata fonksiyonu modelin kaybını ve buna paralel olarak optimize edicinin performansını tahmin etmek için kullanılır. Böylece ağırlıklar bir sonraki değerlendirmedeki kaybı azaltmak için güncellenir. Eğitim sırasında kayıp fonksiyonun değeri en aza indirilmelidir.

Bir YSA'da çok fazla bilinmeyen olduğu için en ideal ağırlıklar doğrudan hızlı bir şekilde hesaplanamaz. Bunun yerine, öğrenme problemi bir arama veya optimizasyon

problemi olarak kullanılır ve modelin yeterince iyi tahminler yapmak için kullanabileceği olası ağırlık kümeleri alanında gezinmek için bir algoritma kullanılır.

YSA'lar sürekli değerlerin tahmininden nesne tanıma uygulamalarına kadar çok geniş bir alanda kullanılmaktadır. Farklı YSA uygulamalarında farklı hatalar veya kayıplar ortaya çıkmaktadır. Basit bir bakış açısı ile kayıp fonksiyonu tahmin edilen değer ve gerçek değer olmak üzere iki parametre alan bir fonksiyondur. Bu fonksiyon, modelin tahmin ettiği değerler ile gerçek değerleri karşılaştırarak modelin performansını ölçmeyi amaçlamaktadır. Tahmin edilen değer gerçek değerden çok uzakta ise kayıp fonksiyonun değeri yüksek olmaktadır (Bonaccorso, 2017, s. 28,29). Ancak her iki değer birbirine yakın ise bu durumda kayıp değeri düşük olmaktadır. Bu nedenle, bir veri seti üzerinde eğitim yaparken bir modeli etkili bir şekilde cezalandırarak bir kayıp fonksiyonu mimarisi oluşturulmalıdır. Kayıp çok yüksekse, bu büyük değer eğitim sırasında ağı üzerinden yayılmakta ve ağırlıklar normalden biraz daha fazla değişmektedir. Kayıp küçükse, ağı zaten iyi bir performans gösterdiğinden ağırlıklar çok az değişmektedir.

Tipik olarak, bir YSA modeli stokastik gradient iniş optimizasyonu algoritması kullanılarak eğitilir ve ağırlıklar hata algoritmasının geri yayılımı kullanılarak güncellenir. Gradient inişindeki *gradient* bir hata değerine karşılık gelir. Belirli bir ağırlık setine sahip model, tahmin yapmak için kullanılır ve bu tahminler için hata hesaplanır.

Seçilen makine öğrenmesi algoritmasının türü, türevlerin hesaplama kolaylığı ve bir dereceye kadar veri setindeki aykırı değerlerin yüzdesi gibi belirli faktörler göz önünde bulundurularak uygun bir kayıp fonksiyonu seçilmelidir.

Kayıp fonksiyonlarını temelde regresyon kayıp fonksiyonları ve sınıflandırma kayıp fonksiyonları olarak ikiye ayırmak mümkündür. Sınıflandırma kayıp fonksiyonları ise kendi içlerinde ikili sınıflandırma ve çok sınıflı fonksiyonlar olarak ikiye ayrılmaktadır. Bu yapı aşağıda verilmiştir.

- Regresyon Kayıp Fonksiyonları (Regression Loss Functions)
 - Ortalama Kare Hata Kaybı (Mean Squared Error Loss)
 - Ortalama Kareli Logaritmik Hata Kaybı (Mean Squared Logarithmic Error Loss)
 - Ortalama Mutlak Hata Kaybı (Mean Absolute Error Loss)

- İkili Sınıflandırma Kayıp Fonksiyonları (Binary Classification Loss Functions)
 - İkili Çapraz Entropi (Binary Cross-Entropy)
 - Menteşe Kaybı (Hinge Loss)
 - Kare Menteşe Kaybı (Squared Hinge Loss)
- Çok Sınıflı Sınıflandırma Kayıp İşlevleri (Multi-Class Classification Loss Functions)
 - Çok Sınıflı Çapraz Entropi Kaybı (Multi-Class Cross-Entropy Loss)
 - Seyrek Çok Sınıflı Çapraz Entropi Kaybı (Sparse Multiclass Cross-Entropy Loss)
 - Kullback Leibler Diverjans Kaybı (Kullback Leibler Divergence Loss)

Bu çalışma kapsamında sadece yaygın olan veya tezde kullanılan kayıp fonksiyonları ile ilgili temel bilgiler verilmiştir.

2.8.4.1. Regresyon kayıp fonksiyonları

Ortalama kare hata kaybı (Mean squared error, MSE)

Ortalama kare hata kaybı, Quadratic Loss veya L2 Loss fonksiyonu olarak da adlandırılmaktadır. Ortalama kare hata kaybı, regresyon problemlerinde yaygın olarak kullanılan kayıp fonksiyonudur. Ortalama kare hatası, sinir ağından elde edilen çıktılar ile veri kümesindeki hedefler arasındaki ortalama kare hatasını hesaplar. Bu fonksiyon hatanın yönü ile ilgilenmeksizin ortalama hatanın büyüklüğü ile ilgilidir. Ancak, hatanın karesinin alınması nedeniyle, gerçek değerlerden çok uzak olan tahminler, daha az sapmış tahminlere kıyasla ağır bir şekilde cezalandırılmaktadır (Goodfellow vd., 2016, s. 108).

$$MSE = \frac{1}{n} \sum_{i=1}^n |y_i - \tilde{y}_i|^2 \quad (2.9)$$

Burada y_i gerçek değerleri, $\tilde{y}_i$ tahmin edilen değerleri, n ise veri kümesindeki toplam gözlem / satır sayısıdır.

Ortalama mutlak hata kaybı (Mean absolute error, MAE)

MAE, L1 Loss fonksiyonu olarak da bilinmektedir. Bu fonksiyon, tahminler ve gerçek değerler arasındaki mutlak farkların toplamı olarak hesaplanır. MSE gibi, bu da hatanın yönünü dikkate almadan sadece hatanın büyüklüğünü ölçer. MSE'den farklı olarak MAE, gradient değerlerini hesaplamak için doğrusal programlama gibi daha

karmaşık araçlara ihtiyaç duyar. Ayrıca MAE, kareden faydalanmadığı için aykırı değerlere karşı daha sağlamdır (Goodfellow vd., 2016, s. 181).

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \tilde{y}_i| \quad (2.10)$$

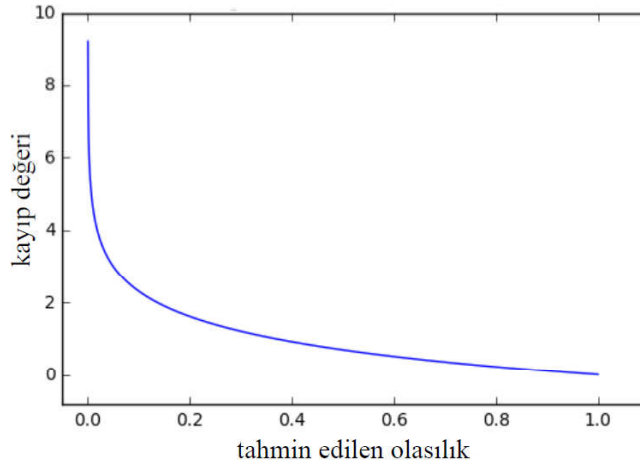
Burada y_i gerçek değerleri, $\tilde{y}_i$ tahmin edilen değerleri, n ise veri kümesindeki toplam gözlem / satır sayısıdır.

2.8.4.2. Sınıflandırma kayıp fonksiyonları

Çapraz entropi kaybı (Cross entropy loss)

Çapraz entropi kaybı genellikle kısaca çapraz entropi, logaritmik kayıp, lojistik kayıp veya log kaybı olarak adlandırılır. Öngörülen her olasılık, gerçek sınıf çıktı değeri (0 veya 1) ile karşılaştırılır ve beklenen değerden uzaklığa bağlı olarak olasılığı cezalandıran bir puan hesaplanır. Ceza logaritmiktir, küçük farklar için küçük bir puan (0.1 veya 0.2) ve büyük bir fark için (0.9 veya 1.0) büyük bir puan hesaplanır.

Çapraz entropi kaybı ne kadar küçükse model o derece başarılıdır. Mükemmel olasılıkları tahmin eden bir modelde çapraz entropi veya log kaybı sıfırdır. Şekil 2.20, gerçek bir gözlem için üretilen olası kayıp değerleri aralığını göstermektedir.



Şekil 2.20. Çapraz entropi kayıp fonksiyonu

Tahmin edilen olasılık 1'e yaklaştıkça, log kaybı yavaş yavaş azalır. Öngörülen olasılık azaldıkça, log kaybı hızla artar. Log kaybı tüm hataları cezalandırmakla birlikte, özellikle kendinden emin ve yanlış tahminleri daha büyük oranda cezalandırmaktadır (Khan vd., 2018, s. 66). Çapraz entropi ve log kaybı duruma göre biraz farklılık

göstermesine rağmen makine öğrenmesinde 0 ile 1 arasındaki hata oranlarını hesaplarırken aynı işlevi yerine getirirler.

M sınıf sayısını göstermek üzere, sınıf sayısının ikiye eşit olduğu ikili sınıflandırmada, çapraz entropi $-(y \log(p) + (1 - y) \log(1 - p))$ şeklinde hesaplanabilir.

M değerinin ikiden büyük olduğu durumlarda (çok sınıflı sınıflandırma) ise çapraz entropi Denklem 2.11 ile hesaplanır.

$$-\sum_{i=1}^M y_i \log(\tilde{y}_i) \quad (2.11)$$

Burada;

m , sınıf sayısı (köpek, kedi, balık gibi)

y_i , tahmin edilen değerler

$\tilde{y}_i$, gerçek etiket değerleri

2.8.5. Düzenleştirme (regularization)

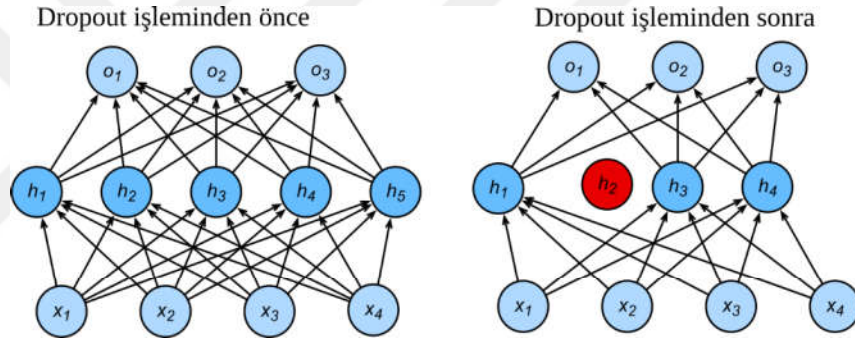
Düzenleştirme, eğitim sırasında bir sinir ağı modelinin karmaşıklığını azaltan ve böylece aşırı öğrenmeyi önleyen bir dizi farklı teknik anlamına gelir. Düzenleştirme ile amaç oldukça karmaşık olan YSA modelinin karmaşıklığını kontrol altında tutmaktır (Goodfellow vd., 2016, s. 119).

Düzenleştirme terimleri genellikle sinir ağındaki parametrelerin değerlerini ölçer. Bu terimin hataya eklenmesi sinir ağının daha küçük ağırlıklara ve sapmalara sahip olmasına neden olur. Aşırı öğrenmeyi önlemenin geleneksel yolu parametre sayısını azaltmaktır. Erken durdurma (early stopping), L1, L2, seyreltme (dropout), eğitim verilerini arttırma ve toplu normalleştirme (batch normalization) yaygın olarak kullanılan diğer düzenleştirme yöntemleridir (Gouk vd., 2018). Bu yöntemlerde parametre sayısını azaltmak yerine, tüm parametreler kullanılmakta ancak sonucu etkilemeyen ve dolayısıyla gereksiz veya önemsiz parametreler cezalandırılmaktadır.

YSA'da düzenleştirme için kullanılan en etkin yöntemlerden birisi dropout yöntemidir (Khan vd., 2018, s. 75). Dropout, Google tarafından geliştirilen ve eğitim verilerindeki karmaşık ortak adaptasyonları önleyerek sinir ağlarındaki aşırı öğrenmeyi

azaltmak için önerilen bir düzenleme tekniğidir. Dropout terimi, bir sinir ağındaki katmanların (hem gizli hem de görünür) rastgele düşürülmesini ifade eder (Hinton vd., 2012). Dropout nöronlar arasındaki karşılıklı öğrenmeyi engelleyerek düzenlemeyi artıran bir yaklaşımdır. Dropout yöntemi ile zayıf bilgilerin elenmesi YSA modelinin performansını artırmaktadır. Problem ve veri setine göre farklılık göstermekle birlikte dropout değeri genellikle 0.5 seçilir. Ayrıca farklı katmanlarda farklı dropout değerleri kullanmak da mümkündür.

Şekil 2.21'de dropout işlemi görsel olarak ifade edilmektedir. Görüldüğü gibi dropout işlemi ağı karmaşıklığını azaltmakta ve aynı zamanda nöronlar arasındaki ilişkilerin yeniden oluşturulmasına imkân vermektedir.



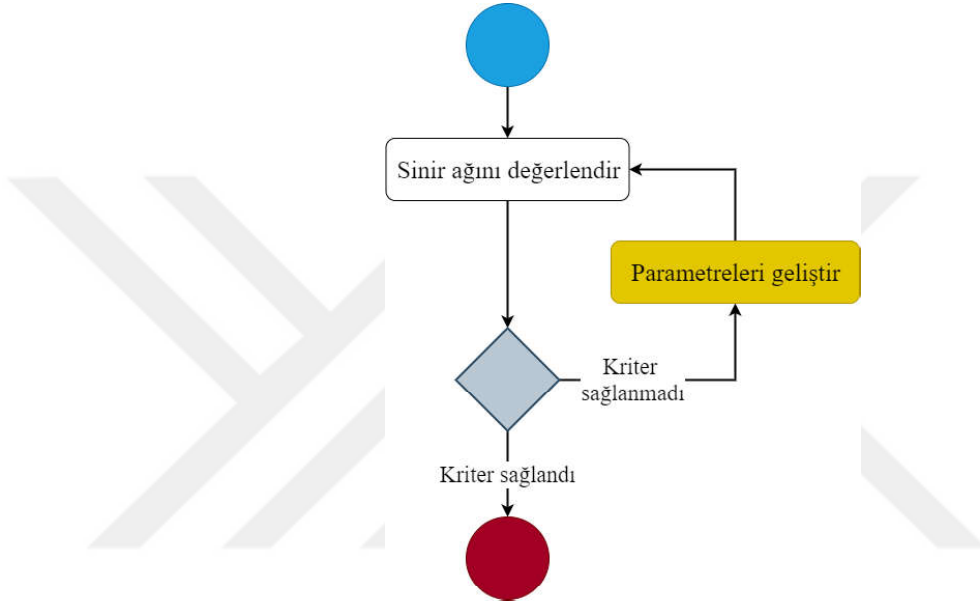
Şekil 2.21. Dropout işlemi ile bazı nöronların ağdan çıkarılması

YSA'da kullanılan bir diğer düzenleme yöntemi toplu normalleştirmedir. Toplu normalleştirme aşırı öğrenmeyi önlemede ve eğitimin doğru bir sırada yapılmasını sağlama konusunda oldukça etkili bir yöntemdir (Ioffe ve Szegedy, 2015).

L2 ve L1 düzenleme yöntemlerinde ise toplam hata fonksiyonunun sonuna eklenen terimler ile düzenleme yapılır. L2 düzenleme yöntemi, sinir ağındaki tüm parametrelerin karelerinin toplamından oluşur ve en yaygın kullanılan yöntemdir. L1 düzenleme yöntemi ise sinir ağındaki tüm parametrelerin mutlak değerlerinin toplamından oluşur. Hesaplanan bu toplamalar, bir düzenleme katsayısı (λ), ile çarpıldıktan sonra toplam hata fonksiyonuna eklenir. λ değeri çok büyük seçilirse yetersiz öğrenmeye; çok küçük seçilirse aşırı öğrenmeye neden olabilir (Gülcü ve Kuş, 2019).

2.8.6. Optimizasyon metotları

YSA için öğrenme problemi, temelde kayıp fonksiyonunun minimum değeri aldığı bir dizi parametre aramaktan oluşur. Kayıp değerinin minimum değerini alması gradient değerinin (eğim) sıfır olduğunu belirtir. Bir YSA'nın eğitim süreci Şekil 2.22'de ifade edilen adımlardan oluşur.



Şekil 2.22. YSA modeli eğitim süreci

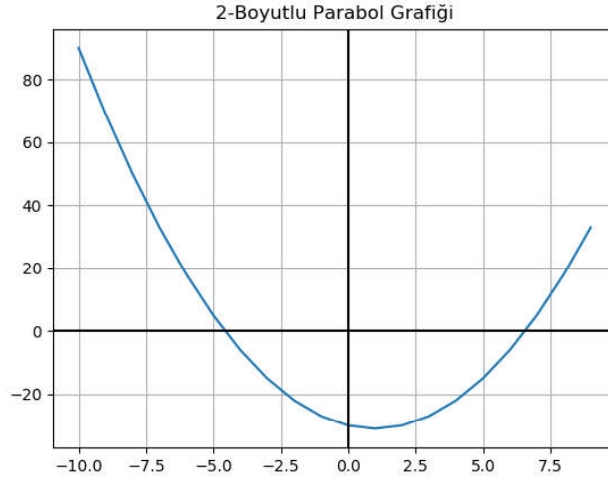
Optimizasyon algoritması, belirli bir koşul sağlandığında durur. Genellikle aşağıdaki durumlardan birisi meydana geldiğinde eğitim süreci durur.

- Artan norm parametreleri minimum değerden düşükse
- Bir turdaki kayıp artışı, belirlenmiş bir değerden düşükse
- Kayıp, bir hedef değere ulaşmışsa
- Kayıp değeri endeksinin normu bir hedefin altına düşmüşse
- Toplam tur sayısı tamamlandıysa
- Maksimum hesaplama süresi aşıldıysa
- Seçim alt kümesindeki hata birkaç tur boyunca artmaya devam ederse

Yapay sinir ağlarında, ağırlıkların doğru bir şekilde güncellenmesi öğrenmenin başarısını doğrudan etkileyen faktörlerdendir. Hata hesaplandıktan sonra elde edilen hata değerine göre modeldeki ağırlıklar güncellenir. Ağırlıkların güncellenmesi için geri yayılım algoritması kullanılır. Geri yayılım algoritması ile sinir ağındaki her bir ağırlığın,

hesaplanan hataya olan etkisini hesaplamak için stochastic gradient descent (stokastik dereceli alçalma) yöntemi yaygın bir şekilde kullanılmaktadır. Momentum optimizasyon, RMS-Prop, Adam (Diederik ve Ba, 2014) ve Adadelta (M. D. Zeiler, 2012) sıklıkla kullanılan diğer metotlardır (Gülcü ve Kuş, 2019).

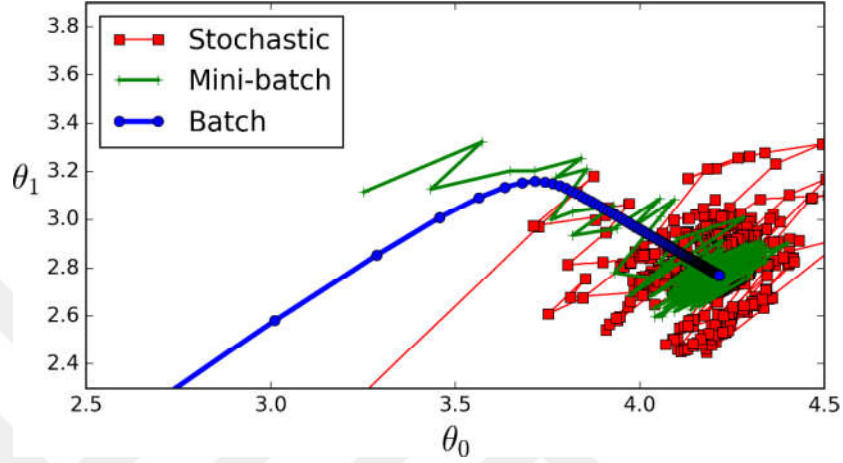
Bu çalışmada sadece stokastik dereceli alçalma yöntemi hakkında bilgi verilmiştir. Stokastik dereceli alçalma yöntemi çeşitli makine öğrenmesi algoritmalarında kullanılan oldukça popüler ve yaygın bir algoritmadır. Bu algoritma YSA'nın temelini oluşturmaktadır. Stokastik dereceli alçalma yöntemini açıklamak için öncelikle dereceli alçalma (gradient descent) terimini açıklamak gerekmektedir. Gradient terimi bir yüzeyin eğimini ifade etmek için kullanılmaktadır. Böylece gradient descent tam anlamıyla o yüzeydeki en düşük noktaya ulaşmak için bir eğimin alçalması fikrine dayanmaktadır. Şekil 2.23'te iki boyutlu bir parabol verilmiştir.



Şekil 2.23. İki boyutlu bir parabol örneği

Yukarıdaki grafikte, parabol üzerindeki en düşük nokta $x = 1$ noktasıdır. Gradient descent algoritmasının amacı y 'nin minimum değerini aldığı noktada karşılık gelen x değerini bulmaktır. Burada y , en düşük noktaya inmek için gradient descent algoritmasının üzerinde çalıştığı amaç fonksiyonu olarak adlandırılır. Gradient matematiksel olarak, bir dizi parametrenin girdilerine göre kısmi türevleri olarak tanımlanabilir. Gradient descent, bir fonksiyon üzerindeki rastgele bir noktadan başlayan ve o fonksiyonun en düşük noktasına ulaşmaya kadar eğim değeri üzerinde adım adım değişiklikler yaparak ilerleyen yinelemeli bir algoritmadır (Gilbert, 2018, s. 35,36).

Genellikle, Batch Gradient Descent, Stochastic Gradient Descent ve Mini-batch Gradient Descent olmak üzere üç tür degrade descent yöntemi vardır. Şekil 2.24'te bu üç yöntemin çalışma mantığı görsel olarak yer almaktadır.



Şekil 2.24. Batch Gradient Descent, Stochastic Gradient Descent ve Mini-batch Gradient Descent yöntemleri (Kim, 2017)

Batch Gradient Descent, yönteminde bir adım atmak için tüm eğitim verileri dikkate alınır. Bu yöntemde tüm eğitim örneklerinin eğimlerinin ortalaması alınıp parametreleri güncellemek için bu ortalama değer kullanılmaktadır. Veri setinin çok büyük olduğu uygulamalarda Batch Gradient Descent yöntemini kullanmak avantajlı değildir. Bu sorunu çözmek için Stochastic Gradient Descent yöntemi kullanılmaktadır. Bu yöntemde her bir adımda sadece bir örnek alınmaktadır. Bu yöntem özellikle hesaplama sürelerinin çok uzamasına neden olabilmektedir. Yukarıdaki iki yöntemin de dezavantajlarını ortadan kaldırmak adına Mini-batch Gradient Descent yöntemi önerilmiştir. Gerçek veri kümesinden daha az olan sabit sayıda bir eğitim örneği kümesi oluşturulmakta ve optimizasyon için bu değer kullanılmaktadır.

2.8.7. Geri yayılım (back propagation)

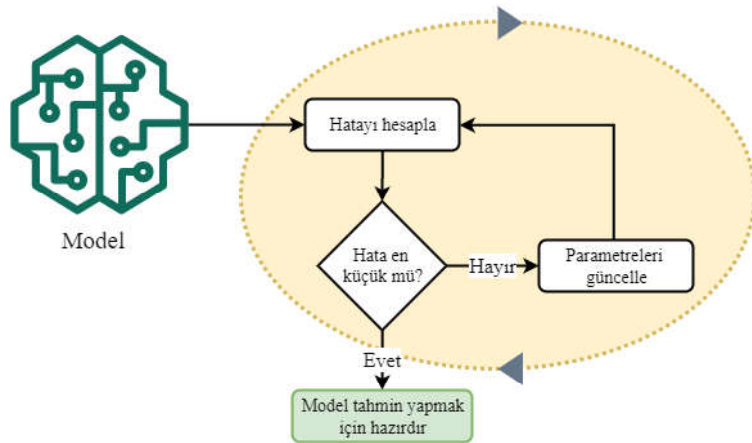
Geri yayılım 1970'lerde karmaşık iç içe fonksiyonların otomatik olarak farklılaştırılması için genel bir optimizasyon yöntemi olarak geliştirildi. Ancak, 1986 yılında Rumelhart, Hinton ve Williams tarafından yapılan çalışmaya kadar önemi kavranamadı (Rumelhart vd., 1986). Bu çalışma ile birlikte bu yöntem kısa sürede popüler hale gelmiş ve birçok araştırmacı tarafından farklı çalışmalarda kullanılmıştır (Chauvin ve Rumelhart, 1995).

Doğrusal ağırları gradient kökenli olarak eğitmek mümkündür. Çok katmanlı ağırlar için de aynı işlem yapılmak istendiğinde önemli bir sorun ortaya çıkar. Gizli birimler için hedef değerler yoktur. Dolayısıyla gizli birimlere ne yapmaları gerektiği söylenmelidir. Bu çözülmemiş sorun aslında sinir ağlarının 1950'lerde ilk popülerlik döneminden sonra gözden düşmesinin en önemli nedeniydi. Geri yayılım algoritmasının gizli birimleri eğitmenin bir yolunu bulup popüler hale gelmesi 30 yıl sürdü ve bu da yeni bir YSA araştırma ve uygulamaları dalgasına yol açtı.

Geri yayılım, çok katmanlı algılayıcıların (YSA) eğitimi için denetimli bir öğrenme algoritmasıdır. Başta bir tanıma göre ise *hataların geriye doğru yayılmasının* kısaltması olan geri yayılım, gradient iniş kullanarak ağırlıkları güncellemek için kullanılan bir mekanizmadır. Geri yayılım eğitim esnasında önceki turlarda elde edilen hata oranına bağlı olarak bir sinir ağının ağırlıklarının ince ayarlanması yöntemidir. Ağırlıkların uygun şekilde ayarlanması, hata oranlarının azalmasına ve modelin genelleme kabiliyetini artırarak modeli daha güvenilir hale getirmesini sağlar (Gilbert, 2018, s. 35).

Geri yayılım yönteminin yanında literatürde geçen bir diğer yöntem ise düğümlerin asla bir döngü oluşturmadığı ileri beslemeli sinir ağıdır (feed forward neural network). Bu tür sinir ağları bir giriş katmanı, gizli katmanlar ve bir çıkış katmanından oluşmaktadır. Yapay sinir ağının ilk ve en basit türüdür.

Sonuç itibari ile geri yayılım günümüzdeki YSA'lar için oldukça kritik bir öneme sahip olan ve oldukça yaygın kullanılan bir yaklaşımdır. Aşağıda bir YSA üzerinde geri yayılım işleminin ne şekilde yapıldığı görsel olarak ifade edilmiştir.



Şekil 2.25. Geriye yayılım algoritmasının çalışma mantığı

2.9. Derin Öğrenme

Son yarım yüzyıldaki bilgisayarlı görü uygulamalarında, doğru sınıflandırma için ihtiyaç duyulan en iyi özellikleri tespit etmek için önemli araştırmalar yapılmış ve bu özellik çıkarıcılarını manuel olarak uygulamak için birçok farklı görüntü filtresi, dönüşümü ve evrişim yöntemleri önerilmiştir. Bu yöntemlerin kısıtlı olan başarıları özellikle son zamanlarda ortaya çıkan büyük veri ve güçlü bilgisayarlar ile ciddi oranda artış göstermiştir. Az sayıda varsayım ve çok az manuel müdahale ile hiyerarşik koniye benzer yapılar büyük miktardaki verilerden otomatik olarak öğrenmeye başlamışlardır. Bu öğrenme yaklaşımları klasik bilgisayarlı görü uygulamalarından ve algoritmalarından oldukça farklıdır. Ortaya çıkan bu yeni nesil algoritmalar belirli bir görev için sabit değildirler ve çeşitli uygulamalarda tekrar tekrar kullanılabilirler. Hem gizli özellik çıkarıcılarını hem de çıktıyı tanımlamak için en iyi nasıl birleştirildiklerini öğrenirler (Alpaydin, 2016, s. 104).

Ham girdiden başlayarak, her gizli katmanın önceki katmandan çıkan değerleri birleştirdiği ve girdinin daha karmaşık işlevlerini öğrendiği derin sinir ağlarının arkasındaki fikir budur.

İlk yapay zekâ uygulamalarında, insanlar için entelektüel olarak zor, ancak bilgisayarlar için nispeten basit olan, matematiksel kuralların bir listesiyle tarif edilebilecek problemler, hızlı bir şekilde ele alındı ve çözüldü. Yapay zekâ için asıl zorluk, insanların kolaylıkla yaptıkları fakat matematiksel olarak ifade edilmesi zor olan sezgisel sorun çözme, konuşma tanıma, yüz tanıma gibi problemlerin çözülmek istenmesi ile başladı. Yapay zekânın bir alt kümesi olan derin öğrenme özellikle son yıllarda bu alanda oldukça başarılı olduğunu göstermiştir. Bunun temel nedeni özellikle gelişen bilgisayar ve dolayısıyla hesaplama teknolojisi sayesinde, bilgisayarların deneyimden öğrenmesine ve nesnelere ait kavramları hiyerarşik olarak anlaması ve bunları daha basit kavramlarla ifade edebilmesidir. Kavramların hiyerarşisi, bilgisayarın daha basit olanları oluşturarak karmaşık kavramları öğrenmesini sağlar. Bilgisayarların tecrübeden öğrenmesiyle birlikte bilgisayarların artık operatörler tarafından verilen bilgiye ihtiyacı büyük oranda azalmış hatta bazı durumlarda tamamen yok olmuştur (Goodfellow vd., 2016, s. 1).

2.9.1. Derin öğrenme nedir?

Derin öğrenme, göreve özel algoritmaların aksine veri temsili öğrenmeye dayanan bir tür makine öğrenmesi yöntemidir (Bengio vd., 2013). Derin öğrenme, bilgisayarın daha basit ve daha küçük kavramlardan karmaşık kavramlar oluşturmalarını sağlar. Örneğin, bir derin öğrenme sistemi, bir kişinin görüntüsünü alt etiketli kenarları ve köşeleri birleştirerek tanır ve bunları hiyerarşik bir şekilde vücudun ilgili bölümlerine birleştirir (Karim vd., 2018). Derin öğrenmede, fikir asgari insan katkısı ile artan soyutlamanın özellik seviyelerini öğrenmektir (Alpaydin, 2016, s. 106; Lecun vd., 2015; Schmidhuber, 2015).

Öğrenme süreci denetlenebilir, yarı denetlenebilir veya denetlenemez gibi farklı yöntemler ile gerçekleştirilebilir. Bilgisayarlı görü, konuşma tanıma (Graves vd., 2004), doğal dil işleme (Jozefowicz vd., 2016), ses tanıma, sosyal ağ filtreleme, makine çevirisi, biyoinformatik (Chicco vd., 2014), ilaç tasarımı, tıbbi görüntü analizi (Litjens vd., 2017) gibi alanlarda derin sinir ağları (deep neural networks), derin inanç ağları (deep belief networks) ve tekrarlayan sinir ağları (recurrent neural networks) gibi derin öğrenme mimarileri yaygın olarak kullanılmaktadır. Bazı alanlarda yapılan çalışmaların oldukça iyi sonuçlar ürettikleri ve alanında uzman kişiler tarafından geliştirilen ürünler ile karşılaştırılabilir yapıda olduğu görülmektedir (Ciregan vd., 2012).

2.9.1.1. Derin öğrenme kütüphaneleri

Günümüzde derin öğrenme uygulamalarında kullanılabilecek tamamen ücretsiz ve gelişmiş özelliklere sahip çok sayıda derin öğrenme kütüphanesi vardır. Bu bölümde sadece bu çalışma kapsamında kullanılan derin öğrenme kütüphaneleri olan Tensorflow ve Keras ile ilgili bilgiler verilmiştir..

TensorFlow

TensorFlow, yüksek performanslı sayısal hesaplama için açık kaynaklı bir yazılım kütüphanesidir. Esnek mimarisi ve masaüstü bilgisayarlardan sunucu kümelerine, mobil cihazlara ve kenar aygıtlarına kadar çeşitli platformlarda hesaplamanın kolayca yapılmasını sağlamaktadır. Orijinal olarak Google'ın yapay zekâ organizasyonu içindeki Google Beyin ekibinden araştırmacılar ve mühendisler tarafından geliştirilen bu kütüphane, makine öğrenmesi ve derin öğrenme için güçlü bir destekle birlikte gelmektedir. Tensorflow'un esnek sayısal hesaplama çekirdeği diğer birçok bilimsel alanda sıklıkla kullanılmaktadır. TensorFlow, birçok API sunmaktadır. En alt seviyeli

API, TensorFlow Core olarak isimlendirilmektedir. Daha üst seviyedeki API'ler, TensorFlow Core'un üzerine kurulmaktadır (Abadi vd., 2016; Khan vd., 2018, s. 160).

Keras

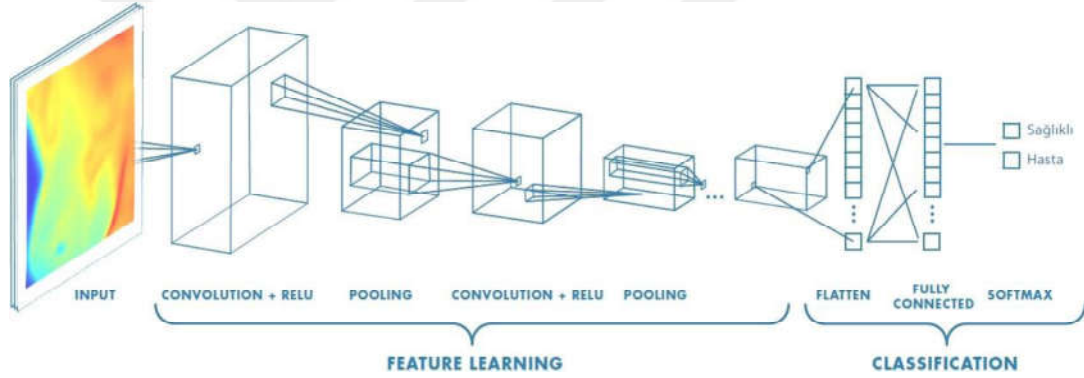
Keras, Python ile yazılmış TensorFlow, CNTK veya Theano kütüphaneleri ile sorunsuz bir şekilde çalışan üst düzey bir derin sinir ağları API'sidir (Karim vd., 2018, s. 20). Python'da yazıldığı için daha geniş bir kullanıcı ve destekçi topluluğuna sahiptir ve öğrenmesi oldukça kolaydır. Keras'ın basitliği, kullanıcıların hızlı bir şekilde derin öğrenme modellerini geliştirmelerine yardımcı olur. Diğer yandan üst düzey bir API olmakla birlikte çok sayıda esneklik sunmaktadır. Ayrıca, bir arka uç olarak diğer birkaç kütüphaneyi desteklediği göz önüne alındığında, gerekirse farklı bir kullanım durumu için farklı bir düşük seviyeli API'den yararlanma esnekliği sağlamaktadır. Şimdiye kadar Keras'ın en yaygın kullanımı, TensorFlow'un arka uç olarak kullanılmasıdır. Kısaca, Keras'ta yazılan kod TensorFlow'a dönüştürülür ve bu işlem daha sonra bir hesaplama örneğinde çalışır (Moolayil, 2019, s. 12). TensorFlow, sayısal hesaplama ve makine öğrenmesi için bir derin öğrenme kütüphanesidir. Açık kaynak kodludur ve sayısal hesaplama için veri akış grafiklerini kullanır. Matematiksel işlemler düğümler ve çok boyutlu veri dizileri ile temsil edilir. Bu kütüphane oldukça tekniktir ve dolayısıyla öğrenmesi zordur. Keras ise, derin sinir ağı kodlamasını basitleştirir. Aynı zamanda CPU ve GPU makinelerinde sorunsuz çalışır.

2.10. Evrimsel Sinir Ağı (Convolutional neural networks)

Evrimsel Sinir Ağları (ESA) (LeCun, 1989), ızgara benzeri bilinen bir topolojiye sahip veriyi işlemek için kullanılan özel bir tür sinir ağıdır (Goodfellow vd., 2016, s. 330). ESA, çeşitli katmanlardan oluşan spesifik bir tür YSA ve aynı zamanda bir derin öğrenme algoritmasıdır. ESA algısal uygulamalar için oldukça popüler ve başarılı bir yöntemdir. ESA çoğunlukla görüntüler gibi yapılandırılmamış bir veri setinden bilgi çıkarma gibi uygulamalarda kullanılmaktadır. ESA'nın oldukça geniş bir uygulama alanı vardır. Görüntü ve video tanıma, öneri sistemleri, görüntü sınıflandırma, nesne tanıma, tıbbi görüntü analizi ve doğal dil işleme ön plana çıkan bazı uygulama alanlarıdır (Bhandare vd., 2016).

ESA temel olarak çok katmanlı algılayıcıların düzenli versiyonlarıdır. Çok katmanlı algılayıcılar genellikle tamamen birbirine bağlı ağlar anlamına gelir; yani bir katmandaki her bir nöron, bir sonraki katmandaki tüm nöronlara bağlanır. ESA, 2012 yılında ortaya

çıkış ve kısa bir sürede yaygın hale gelmiştir. ESA, ilk olarak her yıl düzenlenen ImageNet Büyük Ölçekli Görsel Tanıma Yarışmasında (ILSVRC) (Russakovsky vd., 2015) elde ettiği başarı ile gündeme gelmiştir. Bu tekniğin mevcut tekniklere göre sağladığı birçok üstünlükten dolayı kısa sürede bilgisayarlı görü ve görüntü işleme alanlarında yeni bir aşamaya geçilmesini sağlamıştır. ESA, derin öğrenme temelli bir yöntemdir. ESA yöntemlerinden önce görüntü işleme, bilgisayarlı görü ve sınıflandırma gibi işlemler çok sayıda farklı yöntem ve teknikten oluşan birtakım adımların bir araya getirilmesinden oluşmaktaydı. ESA'da ise tek bir ağ modeli kullanılarak ilgili tüm işlemler yapılabilmektedir. Bu yöntem ile görüntü sınıflandırma, nesne tanıma ve nesne konumu tespitinde oldukça başarılı sonuçlar elde edilmektedir. Bir ESA'nın temel katmanları Şekil 2.26'da görülmektedir.



Şekil 2.26. Bir evrimsel sinir ağı katmanının temel yapısı ve bileşenleri (Ergin, 2019)

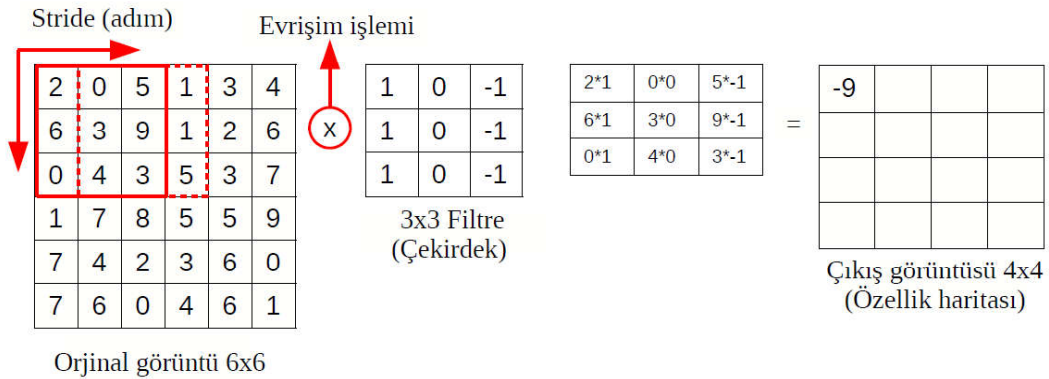
Bir ESA temel olarak gizli katmanlar (özellik tespit etme bölümü) ve sınıflandırma bölümü olmak üzere iki kısımdan oluşmaktadır. Birinci kısımda ağ, özelliklerin tespit edildiği bir dizi evrimsel ve havuzlama (alt örnekleme) işlemi gerçekleştirmektedir. Bir kediye ait görüntü ağı verildiğinde kedinin belirleyici özellikleri (dört ayaklı olması, iki gözlü olması, kuyruğunun olması gibi özellikler) bu kısımda ağ tarafından otomatik olarak tespit edilir. İkinci kısımda tamamen bağlı katmanlar, tespit edilen bu özelliklerin üstünde bir sınıflandırıcı olarak görev yapmaktadır. Bu aşamada algoritma görüntüdeki tahmin ettiği nesne için bir olasılık değeri atamaktadır.

2.10.1. Evrimsel sinir ağı katmanları

ESA'nın ilk katmanı evrimsel katmandır. Evrimsel, bir ESA'nın ana yapı taşlarından biridir. Evrimsel terimi, üçüncü bir işlev üretmek için iki işlevin matematiksel birleşimini belirtir. Evrimsel, bir giriş görüntüsünden özellikleri alan ilk katmandır. Evrimsel katmanı,

küçük girdi verisi karelerini kullanarak görüntü özelliklerini öğrenir ve pikseller arasındaki ilişkiyi modeller. Bu katmanı, adından da anlaşılacağı gibi, evrişim adı verilen temel bir yapı taşından oluşmaktadır. Bir görüntünün küçük bölgelerine, bu bölgedeki piksellerin değerlerini örnekleyen ve tek bir piksele dönüştüren bir evrişim uygulanır. Yeni bir görüntü üretmek için görüntüdeki piksellerin her bölgesine uygulanır. Buradaki temel fikir, yeni görüntüdeki piksellerin etrafındaki pikseller hakkında bilgi içermesi ve böylece bir özelliğin o alanda ne kadar iyi temsil edildiğini yansıtmasıdır (Goodfellow vd., 2016, s. 332,333).

Evrişimin büyüklük (size) ve adım (stride) olmak üzere iki önemli özelliği vardır. Aşağıdaki evrişim, görüntüye 3x3 boyutunda ve 1 adımda uygulanmaktadır. Evrişimin uygulandığı bölgeler arasındaki boşluk 1 olduğu için 1 adımına sahiptir. Başka bir deyişle, evrişim her adımda 1 birim ilerlemektedir.

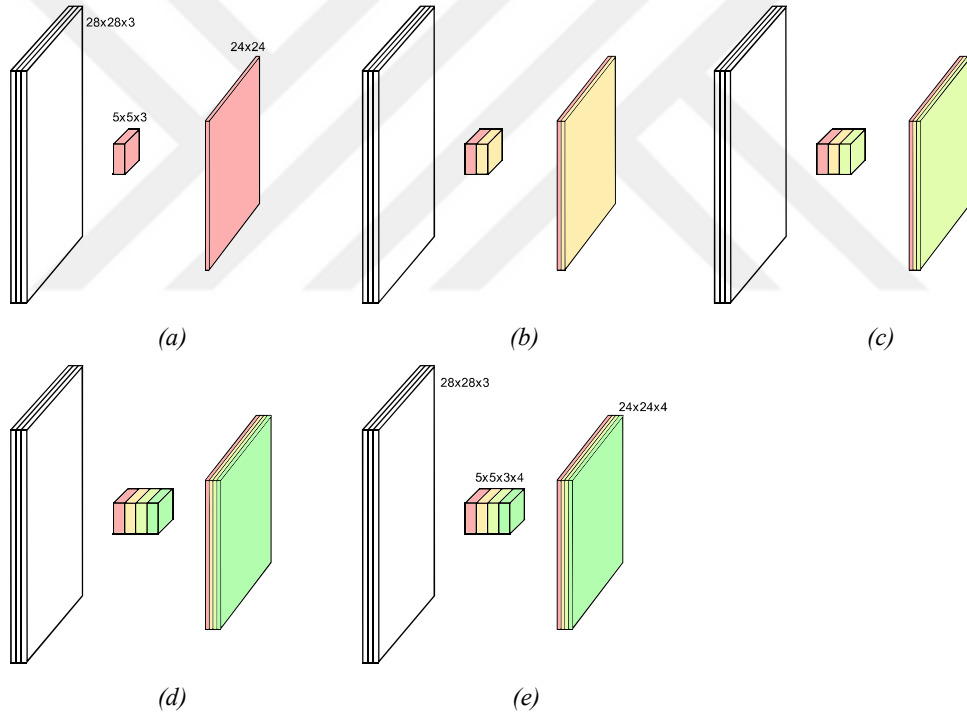


Şekil 2.27. Giriş görüntüsü üzerine evrişim işleminin uygulanması

Evrişimin boyutu ve adımı çıktı görüntüsünün boyutunu belirler. Örneğin, adım değeri 1 olan 3x3 evrişimi, 6x6 görüntüyü 4x4 görüntüye dönüştürür. Evrişim işlemi temel olarak piksel ızgarasını bir algılayıcıdan geçirir, bir başka deyişle giriş piksellerinin ağırlıklı toplamını alır.

Yukarıdaki evrişim, tek renkli bir görüntüye uygulanmıştır. Ancak gerçekte görüntülerin derinliği olur. Başka bir deyişle, bunlar ilk iki boyutun genişlik ve yükseklik, üçüncü boyutun derinlik olduğu üç boyutludur. Başlangıçta derinlik, görüntünün farklı renk bileşenlerini belirtir: kırmızı, yeşil ve mavi gibi. Ayrıca, piksel değerleri ikili değil, genellikle 0 ile 255 arasında olmaktadır. Tipik olarak, görüntünün tüm derinliğine bir evrişim uygulandığı varsayılmaktadır. Bu nedenle, derinliği olan bir görüntüye uygulanan 3x3 bir evrişim, aslında 3x3x3 pikselden oluşan bir girdi olmaktadır. Tek bir girdiye

uygulanan bir evrişimin çıktısı hala tek bir piksel olmakta ve üç boyutlu bir görüntüye uygulanan evrişimin çıktısı iki boyutlu bir görüntü olmaktadır. Bu, görüntünün derinliğini kaybettiği düşüncesinin oluşmasına neden olabilir, ancak durum böyle değildir. Aslında, bir görüntü evrişimli katmanlardan geçerken, tipik olarak derinlemesine artar. Her evrişimli katmanı, tek bir görüntüye bir değil, çok sayıda evrişim uygular. Bu durum her bir evrişim için, yeni bir üç boyutlu görüntü üretmek üzere bir araya getirilen çoklu iki boyutlu çıktılar üretir. Dolayısıyla, evrişimli bir katmanın çıktısının derinliği, uygulanan evrişim sayısıdır. Evrişim bazen çekirdek ve filtre gibi farklı terimlerle ifade edilmektedir. Şekil 2.26 (a,b,c,d) evrişim işleminin aşamalarını ve Şekil 2.26 (e) sonuç görüntüsünü göstermektedir.



Şekil 2.28. Bir görüntü üzerine filtre uygulanarak evrişim işleminin yapılması (Charles ve Olivier, 2019)

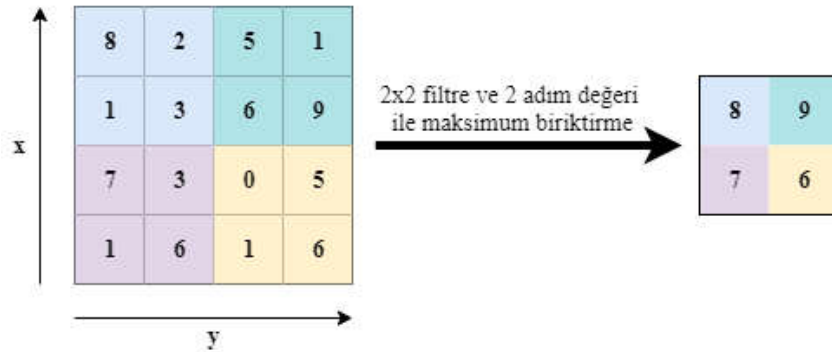
ESA uygulamalarında işlenecek olan görüntülerin boyutlarının yüksek olması ihtimal dahilindedir. Üstelik görüntülerin boyutları küçük olsa bile ağırlı eğitilmesi için gereken toplam görüntü sayısı ve her bir görüntünün üç kanaldan (RGB) oluştuğu göz önüne alındığında giriş verisini mümkün olan en üst seviyede optimize etmek önemlidir. Diğer yandan ESA'da giriş olarak verilen bir görüntünün tüm piksellerine ihtiyaç duyulmaz. Bu nedenle, bir görüntüyü sinir ağının başlangıcında daha küçük bir giriş verisine indirgenmesi gerekir. Bu indirgeme işlemi evrişimsel katmanda gerçekleştirilir.

2.10.1.1. Normalleştirme

Evrişim işleminde giriş görüntüsünün çekirdek ile işleme girmesinden elde edilen iki boyutlu bireysel çıktılara özellik haritaları veya aktivasyon haritaları denir (Şekil 2.26(e)). Her evrişimsel katman uygulandıktan sonra, üç boyutlu çıktıya bir aktivasyon fonksiyonu uygulanır. Bu genellikle ReLU fonksiyonudur. ReLU biyolojik nöronların aktive edilip edilmemesini taklit etmeyi amaçlayan yaygın olarak kullanılan bir fonksiyondur. Bu fonksiyon, negatif piksel değerlerini 0 olarak negatif değerlerden kurtulur.

2.10.1.2. Biriktirme katmanları (Pooling layers)

ESA'da maksimum biriktirme (alt örnekleme) veya aşağı örnekleme katmanı olarak adlandırılan başka bir katman türü daha vardır. Bunlar, evrişim aktivasyon çiftleri arasına yerleştirilir. Her evrişim aktivasyon çiftinden sonra mutlaka olmaları ile ilgili bir zorunluluk yoktur. Evrişim katmanlarının görüntünün boyutunu önemli ölçüde azaltması beklenmez. Bunun yerine, her pikselin komşularını temsil ettiğinden emin olurlar. Böylece, giriş görüntülerine ait önemli bilgileri koruyarak, biriktirme yoluyla küçültmeyi gerçekleştirmeyi mümkün kılarlar (Goodfellow vd., 2016, ss. 340–342). Bunu yapmak için yaygın bir yöntem, bir önceki katmandaki nöronlar kümesinden maksimum değeri kullanarak, maksimum biriktirmeyi oluşturmaktır. Aslında, maksimum biriktirme katmanları bir boyuta ve genişliğe sahiptir. Evrişim katmanlarından farklı olarak, bunlar görüntünün iki boyutlu derinlik dilimlerine uygulanır, böylece elde edilen görüntü aynı derinlikte olur. Sadece daha küçük bir genişlikte ve yüksekliktedir. Aşağıda gösterilen maksimum biriktirme katmanı 2x2 boyutuna sahiptir, bu yüzden 2x2 boyutunda iki boyutlu bir giriş bölgesini alır ve girdiyi aldığı en büyük değerle çıkarır. Ayrıca adım değeri 2'dir. Yani her seferinde 2 birim ilerler.



Şekil 2.29. Maksimum biriktirme katmanı

2.10.1.3. Dolgu (Padding)

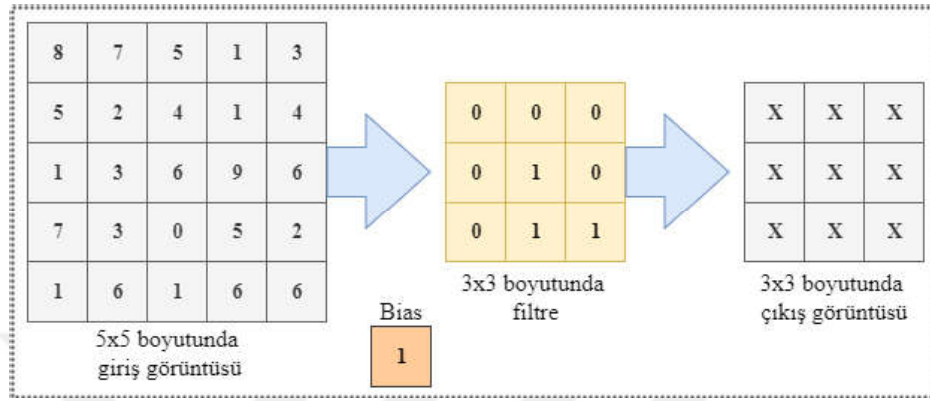
ESA giriş görüntüsünün boyutlarını muhafaza etmez. Gri değerlerden oluşan 7x7 boyutundaki bir giriş görüntüsü 1 adımı değeri kullanılarak 2x2 boyundaki çekirdek ile evrişildiğinde ortaya 6x6 boyutlarında bir görüntü oluşmaktadır. Böylece, her evrişim işlemi gerçekleştirildiğinde, görüntü boyutları küçülür. Bu durum görüntü üzerinde yapılacak işlemlerin sayısına bir üst limit koymuş olmaktadır ki bu durum derin öğrenme modelinin çalışması önünde büyük bir engeldir. Genel olarak, ortadaki pikseller, köşelerdeki ve kenarlardaki piksellerden daha sık kullanılır. Sonuç olarak, görüntülerin sınırları hakkındaki bilgiler ve ortadaki bilgiler korunmaz. Bu sorunların üstesinden gelmek için dolgulama yöntemi kullanılmaktadır. Dolgu basitçe yukarıda belirtilen sorunları önlemek için görüntünün etrafına 0 değerine sahip piksellerden oluşan bir dolgu eklenmesidir (Khan vd., 2018, s. 49).

Bir görüntünün $n \times n$, çekirdeğin $k \times k$ boyutlarında, P 'nin dolgulamayı ve S 'nin adım sayısı olduğu varsayıldığında çıkış görüntüsü için aşağıdaki durumlar söz konudur:

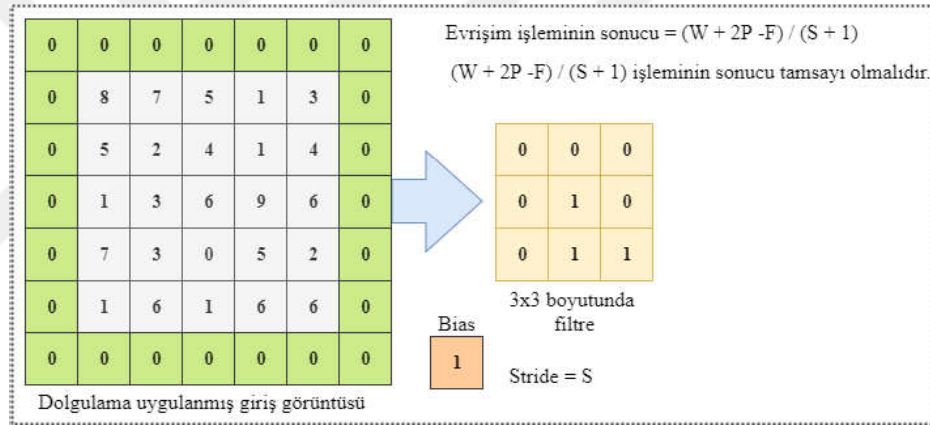
- Evrişim işleminden sonra ortaya çıkacak sonuç görüntüsü $\frac{n+2p-f}{s+1}$ olmaktadır. Örneğin, 100 x 100 boyutlarına sahip bir giriş görüntüsüne, 6 x 6 boyutlarında bir filtre uygulandığında dolgu değeri 7 ve adım değeri 4 için çıkış görüntüsünün boyutu, $\frac{100+2*7-6}{4+1}$ olmaktadır.
- Bu nedenle, $\frac{n+2p-f}{s+1}$ değerinin sonucu bir tamsayı olmalıdır. Eğer $S = 1$ ise sorun yoktur. Fakat S , 1'den büyükse sorun olabilir.
- $S = 1$ (çoğu durumda böyle olur) ve $P = \frac{k-1}{2}$ olarak ayarlanırsa evrişimin sonuç boyutu, görüntü boyutuyla aynı olmaktadır. Bu genellikle arzu edilen bir durumdur.

Şekil 2.30 (a)'da dolgulama işlemi uygulanmadan 3x3 boyutundaki filtre kullanılarak evrişim işlemine tabi tutulan 5x5 boyutunda temsili bir görüntü verilmiştir. Evrişim işleminden sonra çıkış görüntüsünün boyutu 3x3 olmaktadır. Bu şekilde görüntünün kenarları sadece kısmen işlenir. Ayrıca çıkış görüntüsünün boyutu giriş görüntüsünün boyutundan daha küçüktür. Şekil 2.30 (b)'de ise dolgulama işlemi uygulanmış bir görüntüye evrişim uygulanmıştır. Bu yapıda yukarıda ifade edilen

kenarların kısmen işlenmesi ve görüntünün boyutunun küçülmesi sorunları çözülmüştür.



(a)

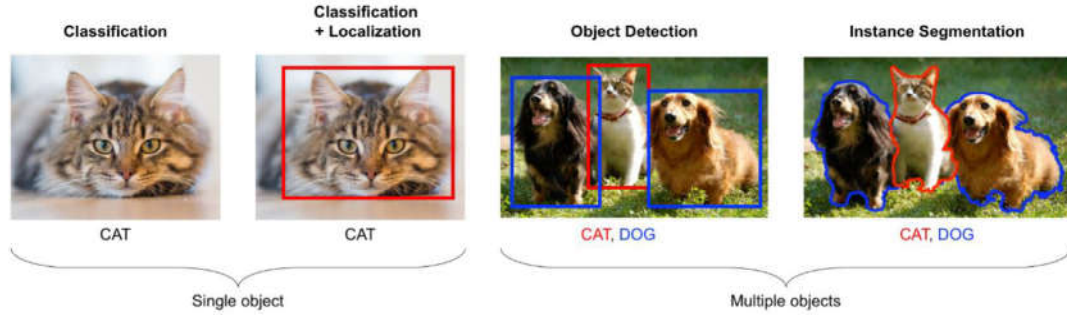


(b)

Şekil 2.30. (a) Dolgulama uygulanmamış bir görüntüye evrişim işleminin uygulanması (b) Bir giriş görüntüsüne dolgulama işleminin ve evrişim işleminin uygulanması

2.11. Görüntü Bölütleme (Image Segmentation)

Görüntü bölütleme, bir görüntüyü birden fazla bölüme ayırma işlemidir. Bu işlem verilen görüntüyü analiz etmeyi kolaylaştırmaktadır (Dass ve Devi, 2012). Esasen, bilgisayarlı görüde temel amaç budur. Bölütler, nesneleri veya nesnelerin parçalarını temsil eder ve piksel kümelerini veya süper pikselleri içerirler. Bilgisayarlı görü uygulamalarında kullanılan çeşitli teknikler vardır. Sınıflandırma, nesne algılama ve bölütleme ön plana çıkan temel tekniklerdir. Şekil 2.31’de bu yöntemler arasındaki fark görsel olarak ifade edilmiştir.

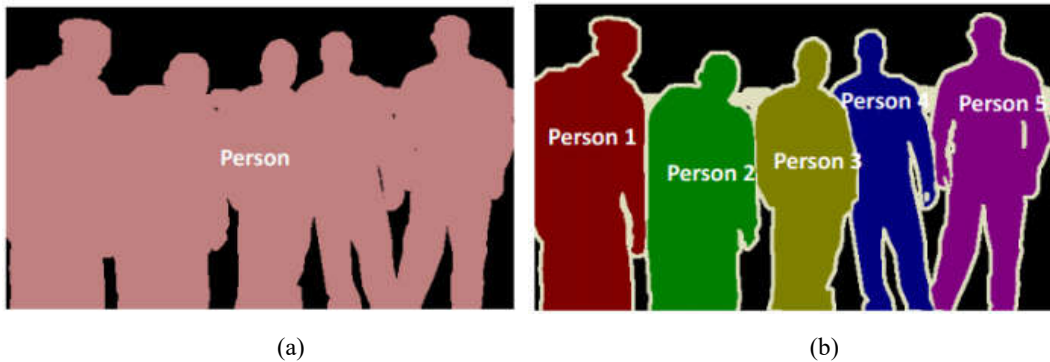


Şekil 2.31. Sınıflandırma, anlamsal bölütleme, nesne algılama ve örnek bölütleme (Bonaccorso vd., 2018, s. 574)

Görüntü sınıflandırması, tüm görüntünün bir nesne veya kavram ile güvenle etiketlenmesi görevidir (Shanmugamani, 2018, s. 37). Sınıflandırma görüntünün ayrıntılı piksel seviyesi yapısını dikkate almaz, tüm girdi için bir tahmin yapmaya çalışır. Nesne algılamada (object detection), bir görüntünün tamamına veya görüntü üzerinde sınırlama kutusu ile belirlenen alana bir sınıf etiketi atanır. Bölütleme ise piksel bazında sınıflandırma yapmaktır.

Görüntü bölütleme için kullanılan farklı yöntemler vardır. Bu yöntemleri farklı kriterlere göre farklı şekillerde sınıflandırmak mümkündür. Kaur ve Kaur (2014) tarafından yapılan çalışmada görüntü bölütleme yöntemleri eşik (threshold) tabanlı, kenar tabanlı (edge based), alan tabanlı (region based), kümeleme tabanlı (clustering based), havza tabanlı (watershed base), kısmi diferansiyel denklem tabanlı (PDE based) ve YSA tabanlı yöntemler olarak sınıflandırılmıştır.

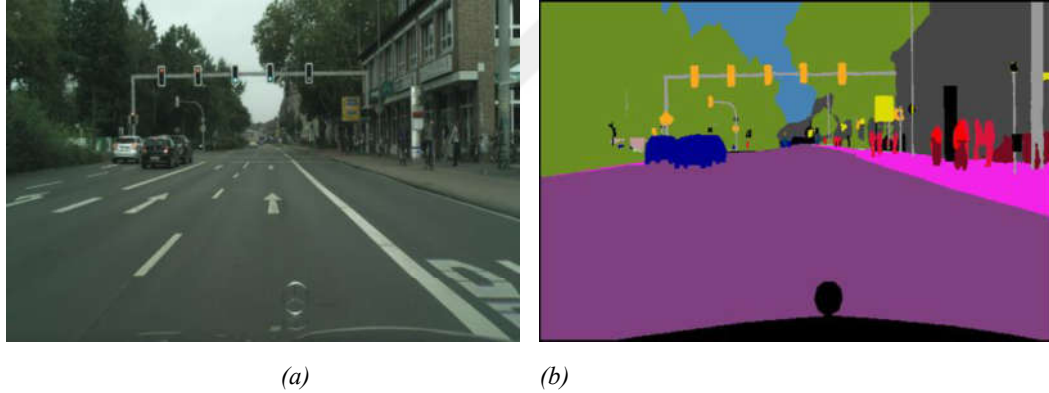
Bu çalışmada bir görüntünün piksellerini anlamlı sınıflara ayıran görüntü bölütleme ve görüntüdeki her nesnenin sınıfını tanımlayan örnek bölütleme (instance segmentation) yöntemleri ve YSA tabanlı yöntemler hakkında bilgi verilmiştir.



Şekil 2.32. Anlamsal bölütleme (a) Örnek bölütleme (Analytics, 2019)

2.11.1. Anlamsal bölütleme (Semantic segmentation)

Anlamsal bölütleme tüm görüntü için sınıflandırma yapmaya çalışan bir yaklaşıma dayanmaktadır. Anlamsal bölütleme ağı, bir görüntüdeki her pikseli sınıflandırır, böylece sınıfa göre ayrılmış bir görüntü elde edilir. Görüntüdeki her piksel için tahmin yapıldığından, bu işleme yoğun tahmin (dense prediction) denir. Önceki uygulamalardan farklı olarak, anlamsal bölütlemeye beklenen çıktı sadece etiketler ve sınırlayıcı kutu parametreleri değildir. Çıktı, her pikselin belirli bir sınıfa ait olacak şekilde sınıflandırıldığı yüksek çözünürlüklü bir görüntüdür. Ayrıca genellikle giriş görüntüsü ile çıkış görüntüsü aynı boyutlara sahiptir. Anlamsal bölütleme tıbbi görüntüleme, endüstriyel denetim, otonom araçlar, uydu görüntüsü analizi gibi alanlarda yaygın bir şekilde kullanılmaktadır (Barthakur ve Sarma, 2020). Şekil 2.33'te gerçek bir görüntü ve bu görüntünün anlamsal bölütleme tekniği ile piksel bazında sınıflandırılmış hali görülmektedir.



Şekil 2.33. Gerçek bir görüntü ve bu görüntünün anlamsal bölütleme ile sınıflandırılması (Valada vd., 2017)

Yukarıda ifade edildiği gibi anlamsal bölütleme uygulamalarında genellikle giriş görüntüsü ile çıkış görüntüsü aynı boyutlardadır ve giriş görüntüsündeki tüm pikseller bir sınıfa dahil edilir.

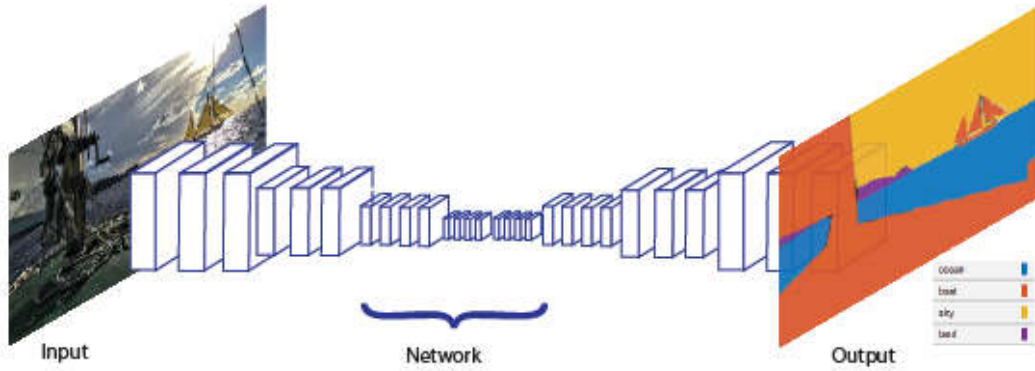
Anlamsal bölütleme uygulamalarında w giriş görüntüsü genişliği, h giriş görüntüsü yüksekliği ve 3 görüntüdeki kanal sayısı olmak üzere $w \times h \times 3$ boyutunda bir giriş görüntüsünü tüm piksellere karşılık gelen öngörülen sınıf kimliğini içeren bir $w \times h$ matrisi oluşturulmaktadır.

2.11.1.1. ESA ile görüntü bölütleme

Sınıflandırma, nesne tanıma gibi diğer uygulamaların çoğunda olduğu gibi, anlamsal bölütleme uygulamalarında ESA kullanmak mümkündür. Anlamsal bölütleme için bir ESA kullanıldığında, çıktı sabit uzunluklu bir vektör değil genellikle giriş görüntüsü ile aynı boyutlara sahip bir görüntüdür.

Anlamsal bölütleme için kullanılan ESA modelinin mimarisi genel olarak, birkaç evrişimli katman, doğrusal olmayan aktivasyonlar, toplu normalleştirme ve biriktirme katmanlarını içermektedir. İlk katmanlar, kenarlar ve renkler gibi düşük seviyeli kavramları öğrenir ve sonraki katmanlar, farklı nesneler gibi üst düzey kavramları öğrenir.

Daha düşük bir seviyede, nöronlar görüntünün küçük bir bölgesi için bilgi içerirken, daha yüksek bir düzeyde, nöronlar görüntünün büyük bir bölgesi için bilgi içerirler. Böylece, daha fazla katman eklendikçe, görüntünün boyutu azalmaya devam eder ve kanal sayısı artmaya devam eder. Alt örnekleme, biriktirme katmanları tarafından yapılır. Şekil 2.34'te anlamsal bölütleme ağının yapısı ve temel katmanları görsel olarak temsil edilmiştir.



Şekil 2.34. Anlamsal bölütleme ağı (Documentation, 2019)

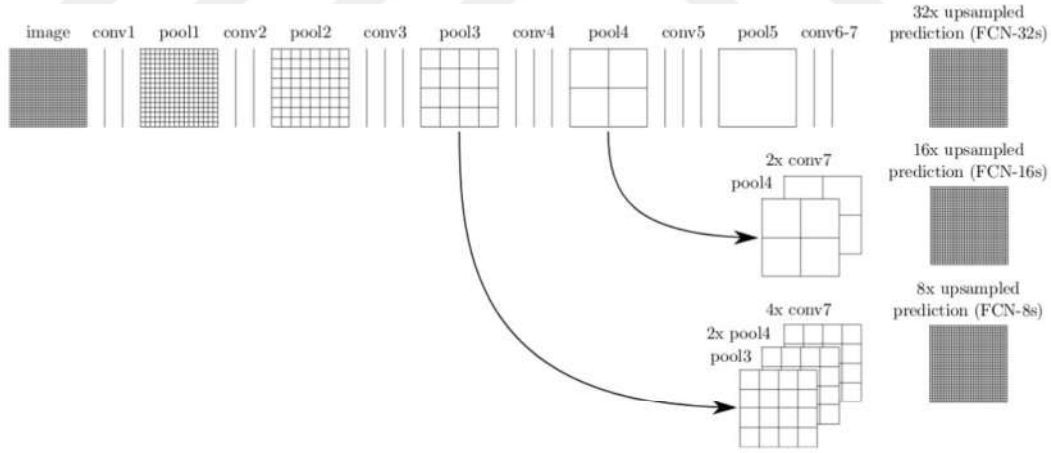
Görüntü sınıflandırması uygulamalarında, uzamsal tensörün evrişim katmanlarından sabit uzunlukta bir vektöre eşlenmesi gerekmektedir. Bunu yapmak için, tüm mekânsal bilgiyi yok eden tamamen bağı katmanlar kullanılır. Oysa anlamsal bölütleme uygulamalarında mekânsal bilginin saklanması önemlidir. Bu nedenle tam bağı hiçbir katman kullanılmaz ve bu ağlara tam evrişimsel ağlar denir. Alt örnekleme

katmanlarıyla birleştirilmiş evrişimli katmanlar, yüksek seviye bilgisini içeren düşük çözünürlüklü bir tensör üretir.

Anlamsal görüntü bölütleme uygulamalarında kullanılabilecek çeşitli evrişimsel sinir ağları mevcuttur. Deeplab, TEA, SegNet, MobileNet, PSPNet ve U-Net ön plana çıkan ağlardır (Ronneberger vd., 2015).

2.11.1.2. Görüntü bölütleme için derin öğrenme teknikleri

Derin öğrenme, anlamsal bölütleme tekniklerini büyük ölçüde basitleştirmiş ve oldukça etkileyici sonuçların alınmasını sağlamıştır. Anlamsal bölütleme için kullanılan en basit ve yaygın mimarilerden biri tam evrişimsel ağıdır (Long vd., 2015). Bu yöntemde bir dizi evrişim üzerinden giriş görüntüsünü daha küçük bir boyuta (daha fazla kanal kazanırken) indirgemek için alt örnekleme aşamasında TEA kullanılmaktadır. Bu evrişimler kümesi genellikle kodlayıcı olarak adlandırılır. Kodlanmış çıktı daha sonra bilinear interpolasyon veya bir dizi devrik evrişim yoluyla yukarı örneklenir. Bu devrik evrişim kümesi genellikle dekoder olarak adlandırılır.



Şekil 2.35. TEA'da alt örnekleme ve üst örnekleme (Long vd., 2015)

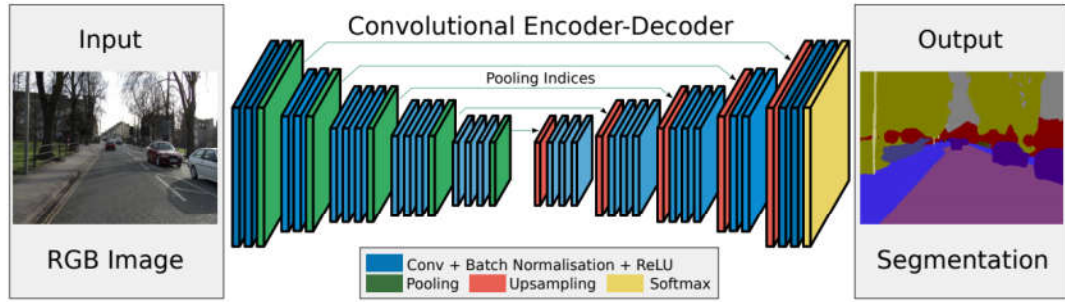
FCN8, FCN16 ve FCN32, TEA'nın varyantlarıdır. FCN8 ve FCN16'da aşağıda açıklanan U-Net modelinde olduğu gibi atlama bağlantıları (skip connections) kullanılır.

Etkili olmasına rağmen TEA'nın temel mimarisinden kaynaklanan birtakım dezavantajları vardır. Bu dezavantajlardan birisi, devrik evrişim işleminin çıktısının düzensiz çıkışması nedeniyle dama tahtası dokularının oluşmasıdır. Diğer bir dezavantaj ise kodlama işleminden kaynaklanan bilgi kaybı nedeniyle sınırlardaki zayıf çözümdür. TEA modelinin performans kalitesini artırmak için çeşitli çözümler önerilmiştir. U-Net,

Tiramisu model, DeepLabv3 ve bazı hibrit modeller başarıları kanıtlanmış popüler çözümlerdendir. Bu çalışma kapsamında sadece bazı yöntemler hakkında gerekli bilgiler verilecektir. Ayrıca bu bölümde verilen bilgilerde yaygın olarak kullanılan iki kavram vardır. Bunlar; atlama bağlantıları ve kodlayıcı-kod çözücü (encoder-decoder) kavramlarıdır. Bu iki kavram aşağıda açıklanmıştır.

Kodlayıcı ve kod çözücü katmanları basitçe bir araya getirilip istiflendiğinde, düşük seviyeli bilgi kaybı olabilir. Bu nedenle, kod çözücü tarafından üretilen bölütleme haritalarındaki sınırlar yanlış olabilir. Kayıp bilgiyi telafi etmek için, kod çözücünün kodlayıcı katmanlar tarafından üretilen düşük seviye özelliklere erişmesine izin verilmektedir. Bu işlem atlama bağlantıları ile gerçekleştirilmektedir.

Genel bir semantik bölütleme yaklaşımı bir kodlayıcı ağı ve bunu takip eden bir kod çözücü ağından oluşmaktadır. Kodlayıcı, bir giriş görüntüsü alır ve yüksek boyutlu bir özellik vektörü oluşturur. Kodlayıcıda evrişim işlemi ve maksimum biriktirme yapılır. Kod çözücü yüksek boyutlu bir özellik vektörü alır ve anlamsal bir bölütleme maskesi oluşturur (Badrinarayanan vd., 2017). Şekil 2.36 bir anlamsal bölütleme ağının yapısını ve kodlayıcı kod çözücü öğelerini göstermektedir.



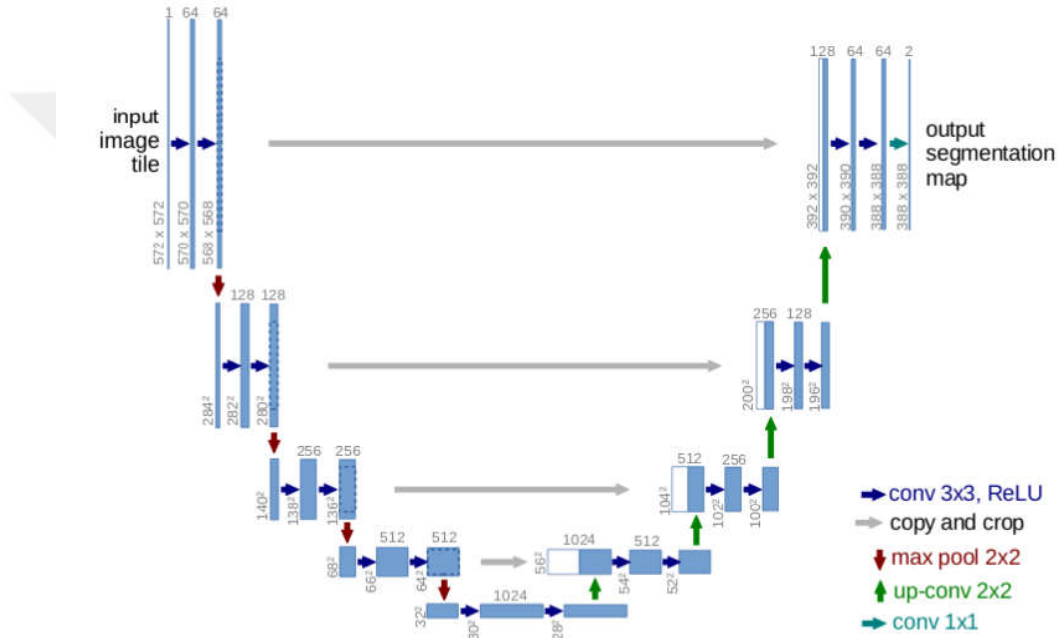
Şekil 2.36. Anlamsal bölütleme ağı yapısı (Badrinarayanan, Kendall, & Cipolla, 2017)

U-Net

Bir makine görüntüyü, görüntü çözünürlüğüne sahip piksel matrisi olarak $h \times w \times d$ olarak görür. Burada h yükseklik, w genişlik ve d ise boyuttur. Bir RGB görüntüsü için d değeri 3 ve gri tonlamalı bir görüntü için d 1'dir. ESA ile yapılan sınıflandırma uygulamalarında çoğu zaman görüntü bir vektöre dönüştürülür ve tüm işlemler bu vektör üzerinden yapılır. Ancak U-Net yönteminde yapı biraz daha farklıdır. Bu modelde görüntü bir vektöre dönüştürülür ve daha sonra aynı eşleme vektörü tekrar görüntüye dönüştürülmek için kullanılır. Bu, görüntünün orijinal yapısını koruyarak bozulmayı

azaltır. Diğer yandan ESA, büyük ölçüde tüm görüntünün bir sınıf etiketi olarak sınıflandırılması gerektiğinde kullanılır. Ancak birçok uygulamada, görüntünün her pikselini sınıflandırmak gerekebilmektedir. Bu noktada U-Net iyi bir alternatiftir.

U-Net, basit TEA mimarisinin performansını artırmak amacıyla önerilen yöntemlerden biridir. Bu yöntemde evrişim bloklarının çıkışından, aynı seviyedeki devrik evrişim bloğunun ilgili girişine atlama bağlantıları vardır (Ronneberger vd., 2015).



Şekil 2.37. U-Net model yapısı ve atlama bağlantıları (Long vd., 2015)

Bu atlama bağlantıları, gradientlerin daha iyi akmasını sağlar ve görüntü boyutunun birden çok ölçeğinden bilgi sağlar. Daha büyük ölçeklerden (üst katmanlar) alınan bilgiler modelin daha iyi sınıflandırma yapmasına yardımcı olabilir. Daha küçük ölçeklerden yani daha derin katmanlar elde edilen bilgiler, modelin daha iyi bölütlere ayrılmasına / yerleştirilmesine yardımcı olabilir (Long vd., 2015).

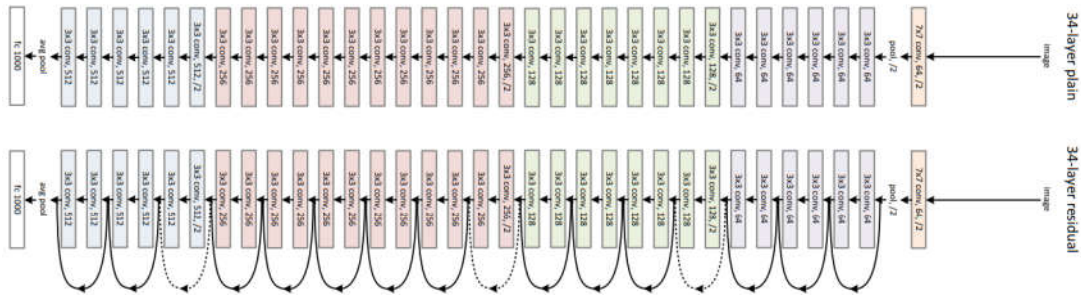
SegNet

SegNet anlamsal tabanlı piksel sınıflandırma yapmak amacıyla ortaya çıkmış derin evrişimli kodlayıcı-kod çözücü mimarisine sahip bir modeldir. Bu modelde kodlayıcı çeşitli evrişim (konvolüsyon) katmanlarına ve kod çözücü de çeşitli dekonvolüsyon (deconvolution) katmanlarına sahiptir. SegNet, TEA tarafından üretilen kaba çıktıları iyileştirmiştir. Özellikler boyut olarak azaltıldığında, dekonvolüsyon ile görüntü boyutu

tekrar yükseltir ve konvolüsyon etkilerini tersine çevirir. Dekonvolüsyon, örnekleme için parametreleri öğrenir. Bu mimarinin çıktısı, biriktirme katmanlarındaki bilgi kaybı nedeniyle kaba olmaktadır (Shanmugamani, 2018, s. 135)ç.

Residual networks (ResNet)

Geleneksel sinir ağlarında, ağın derinliği yani katman sayısı çok önemli bir parametredir. Teorik olarak daha fazla katmana sahip ağların daha iyi temsil kabiliyetine sahip oldukları ve dolayısıyla daha performans göstereceği görüşü yaygındır. Fakat pratikteki uygulamalar durumun daha farklı olabileceğini göstermektedir. Bu durumun temel iki nedeni vardır. Birincisi kaybolan gradientlerdir. Bu sorun geri yayılım sırasında yeterince güçlü aktivasyon değerlerine sahip olmayan nöronların değerlerinin uzun katman zinciri içerisinde kaybolması ve dolayısıyla bu nöronların ağa katkı sağlayamaması olarak ifade edilebilir (vanishing gradient sorunu). Ayrıca bazı nöronların yok olması ilk katmanın ağırlıklarının geriye yayılım işlemi ile doğru bir şekilde güncellenmesine engel olur (Naranjo-Alcazar vd., 2019). İkincisi ise optimizasyon sorunudur. Günümüzdeki ESA'daki parametre paylaşımı ile parametre sayısı önemli ölçüde azalmış olsa da ağın derinliğinin artması ile birlikte farklı kaynaklardan modele dahil olan parametreler ağın optimizasyonunu zorlaştırmaktadır. Bu sorunların üstesinden gelmek amacıyla ortaya çıkan en önemli modellerden birisi ResNet'tir (He vd., 2016). ResNet, Microsoft tarafından önerilen modeldir. ImageNet 2016 yarışmasında yüzde 96,4 doğruluk elde etmiştir. ResNet kimlik matrisini kullanarak bu sorunu çözer. Geri yayılma kimlik fonksiyonu ile yapıldığında, gradient sadece 1 ile çarpılır. Bu, girdiyi korur ve bilgi kaybını önler (He vd., 2016).

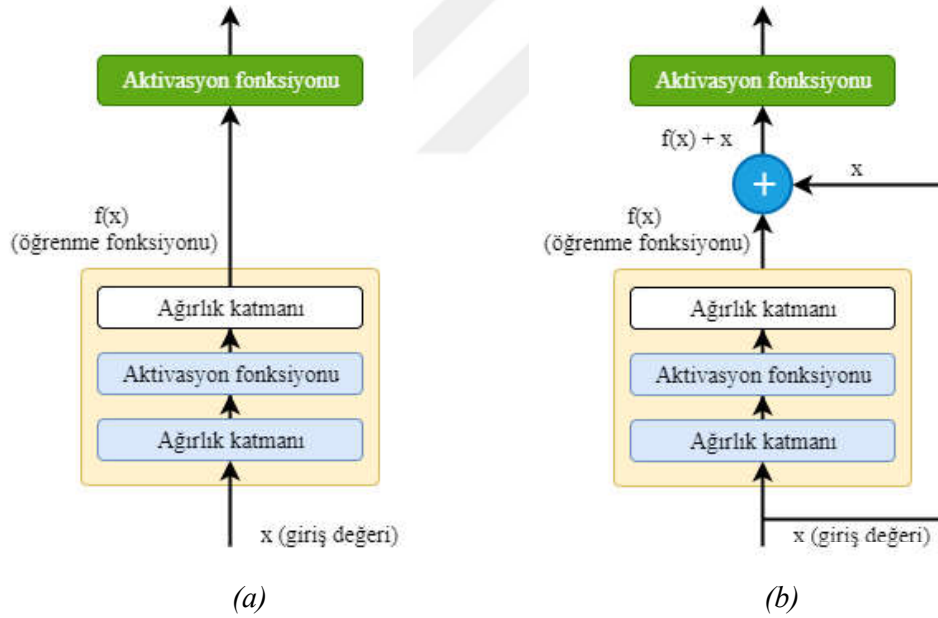


Şekil 2.38. Üst kısım 34 Katman ResNet ESA ve alt kısım 34 Katman CNN (He vd., 2016)

ResNet ağının bileşenleri arasında 3x3 filtreler, adım (stride) 2'ye sahip ESA aşağı örnekleme katmanları, küresel ortalama biriktirme katmanı ve sonunda softmax ile 1000

yollu tam bağlantılı katman bulunur. ResNet, evrişim bloğunun çıkışına orijinal bir girişin de eklendiği bir atlama bağlantısı kullanır. Bu, gradient akışına alternatif bir yol sağlayarak gradient kaybolması sorununun çözülmesine yardımcı olur. Ayrıca, daha yüksek katmanın daha düşük olmayan bir katman kadar iyi performans göstermesine yardımcı olan kimlik işlevini kullanır (He vd., 2016; Khan vd., 2018, s. 110).

ResNet'in temel mantığı bir katmandaki etkinleştirme biriminin atlama bağlantıları ile doğrudan ağına daha derin katmanından beslenmesidir. ResNet'in yapı taşı artık blok (residual block) veya kimlik bloğu (identity block) denir. Geleneksel sinir ağlarındaki yapı Şekil 2.39 (a)'da görüldüğü gibidir. Artık blokları olan bir ağda, Şekil 2.39 (b)'de görüldüğü gibi önceki katmandaki aktivasyon ağıdaki daha derin bir katmanın aktivasyonuna eklenmektedir. Bu basit ayar, daha derin sinir ağlarının eğitime imkân vermektedir.



Şekil 2.39. Artık öğrenme: bir yapı taşı (He vd., 2016)

MobileNet

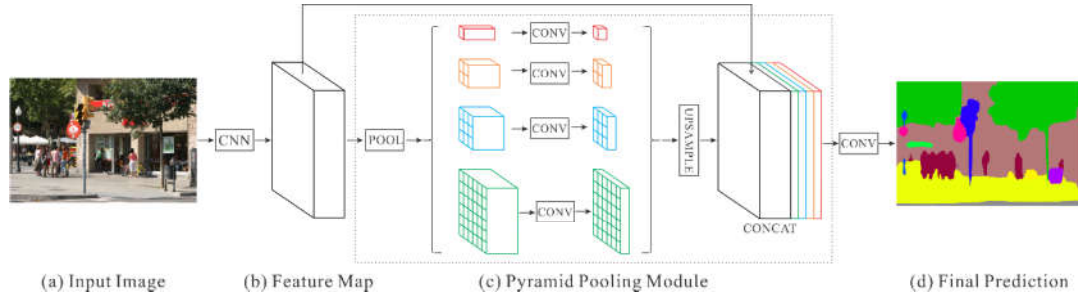
Bu model, küçük bir model boyutuna ve daha hızlı bir çıkarım süresine sahip olacak şekilde optimize edilmiştir. MobileNet Google tarafından önerilmiştir. Bu model, cep telefonlarında ve kısıtlı donanıma sahip cihazlarda çalışmak için idealdir. Küçük boyuttan dolayı, modelin doğruluğunda bazı sapmalar olabilmektedir (Howard vd., 2017).

Deeplab V3

Deeplab V3, bilgisayarlı görü derin öğrenmesinde birkaç güçlü kavramı birleştiren Google araştırmacıları tarafından tasarlanan ve yaygın olarak kullanılan bir modeldir. Bu modelinin getirdiği en büyük yenilik çarpık evrişim (atrous convolution) veya dilated evrişim (dilated convolution) olarak bilinen yapıdır. Bu yapı son birkaç katmandan alt örnekleme işlemlerini kaldırmakta ve karşılık gelen filtre çekirdeklerini, filtre ağırlıklarını arasına delikler ekleyerek eşleştirmektedir. Böylece daha yoğun özellik eşlemelerini çıkarmak için ImageNet'in önceden oluşturduğu ağların yeniden konumlandırılmasına olanak tanımaktadır (L.-C. Chen vd., 2017).

Pyramid scene parsing network (PSPNet)

PSPNet, bir sahnenin daha iyi küresel bağlam temsilini öğrenmek için optimize edilmiştir. Bu yöntemde ilk olarak, bir özellik haritası almak için görüntü temel ağa aktarılır. Özellik haritası farklı ölçeklere indirgenir. Toplanan özellik haritalarına evrişim uygulanır. Bundan sonra, tüm özellik haritaları ortak bir ölçekte örneklenir ve birleştirilir. Son olarak, nihai bölütleme çıktıları üretmek için başka bir evrişim katmanı kullanılır. Burada, daha küçük nesneler, yüksek çözünürlüğe sahip biriktirilmiş özellikler tarafından iyi yakalanırken, büyük nesneler daha küçük bir boyuta biriktirilmiş özellikler tarafından yakalanır (Zhao vd., 2017).



Şekil 2.40. PSPNet ağı (Zhao vd., 2017)

İç ve dış mekân sahneleri içeren görüntüler için, nesneler genellikle farklı boyutlarda bulunduğundan PSPNet tercih edilir. Bu modelde giriş görüntülerinin büyük boyutlu olması (256x256'dan büyük) tercih edilmektedir.

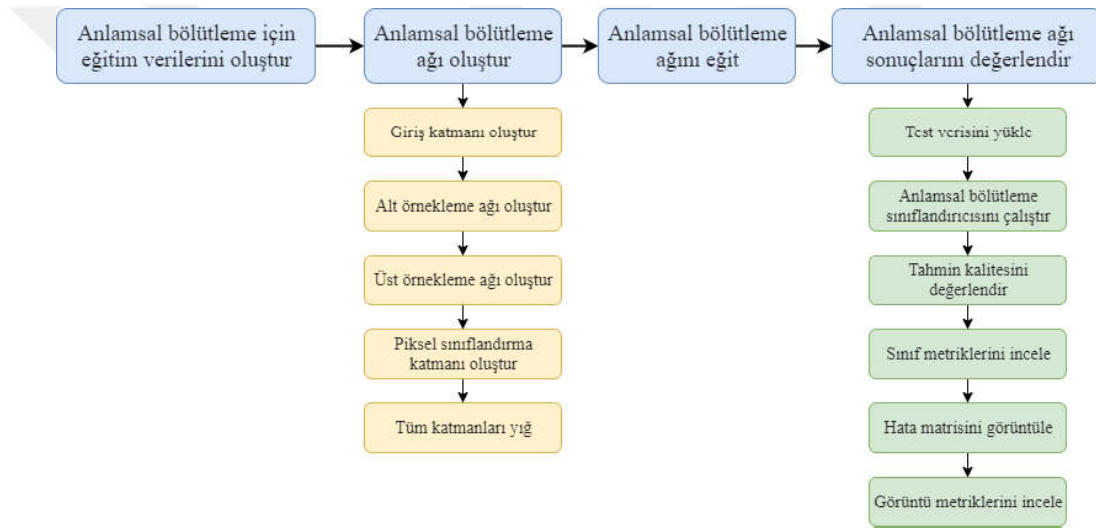
Özelleştirilmiş ESA modeli tanımlamak

Anlamsal bölütleme uygulamalarında ImageNet ile önceden eğitilmiş bir model kullanmanın yanı sıra, özelleştirilmiş bir ağ da temel bir ağ olarak kullanılabilir. Bölütleme uygulaması basitse, ImageNet ön eğitimi gerekli değildir. Özelleştirilmiş bir

temel model kullanmanın en önemli avantajı, uygulamaya göre uyarlanabilmesidir. Yapılan bölütleme uygulamasında üzerinde çalışılan görüntüler ImageNet'e benziyorsa, önceden eğitilmiş ImageNet modellerini kullanmak yararlı olur. Arabalar, hayvanlar, insanlar vb. ortak nesnelere sahip olan iç / dış mekân görüntülerinin kullanıldığı uygulamalarda ImageNet ile eğitilmiş modeller eğitimin başarısını artırabilir.

2.11.1.3. Anlamsal bölütleme ağını oluşturma

Bir anlamsal bölütleme ağı oluşturmak ve çalıştırmak için yapılması gereken temel adımlar Şekil 2.41'de açıklanmıştır.



Şekil 2.41. Anlamsal bölütleme ağı oluşturma adımları

- **Anlamsal bölütleme için eğitim verileri oluşturma:** Bir anlamsal bölütleme ağını eğitmek için bir görüntü koleksiyonuna ve buna karşılık gelen piksel etiketli görüntü koleksiyonuna ihtiyaç vardır. Piksel etiketli görüntü, her piksel değerinin o pikselin kategorik etiketini temsil ettiği görüntüdür.
- **Bir anlamsal bölütleme ağı oluşturma:** Bir anlamsal bölütleme ağı oluşturmanın ilk adımı anlamsal bölütleme ağlarında bulunan ortak katmanları oluşturmaktır. Genel olarak anlamsal bölütleme ağlarının temelinde, öncelikle bir görüntünün evrimsel ve ReLU katmanları arasında alt örnekleme yapılıp ve ardından çıktıyı giriş görüntüsü boyutuna uyacak şekilde üst örnekleme yapılması işlemleri vardır. Bu işlem, görüntü piramitleri kullanılarak yapılan standart ölçek-uzay analizine benzer. Ancak bu işlem sırasında, bir ağı, bölütleme

yapmak istenilen belirli bir sınıf kümesi için en iyi duruma getirilmiş doğrusal olmayan filtreler kullanarak işlemleri gerçekleştirir.

- **Anlamsal bölütleme ağını eğitme:** Bu işlem ağı eğitecek olan eğitim verisinin yüklenmesi ile başlar. Yüklenen veriler sırasıyla ağdaki katmanlardan geçerek işlenir.
- **Anlamsal bölütleme sonuçlarını değerlendirme ve inceleme:** Bu aşamada test verisi kullanılarak ağın başarısı çeşitli parametreler göz önünde bulundurularak değerlendirilir.

2.11.2. Örnek bölütleme (Instance segmentation)

Örnek bölütleme, nesnelerin sınırlarının ayrıntılı piksel seviyesinde tanımlanmasını içerir. Örnek bölütleme temelde nesne algılama ile anlamsal bölütleme yöntemlerinin birlikte kullanılması mantığına dayanmaktadır. Doğal bir fikir olarak her iki yöntemi birleştirmek böylece öncelikle bir nesnenin etrafındaki sınırlama kutusu tanımlanır ve daha sonra ise sınırlama kutusunun içindeki hangi piksellerin nesneye ait olduğu tespit edilir (Liu vd., 2017). Başka bir deyişle, piksellerin aynı nesneye ait olduğunu belirten renkli veya gri tonlamalı değerler kullanılarak bir maske oluşturulabilir. Bu maskeyi üreten algoritmaların sınıfına örnek bölütleme algoritması denir.

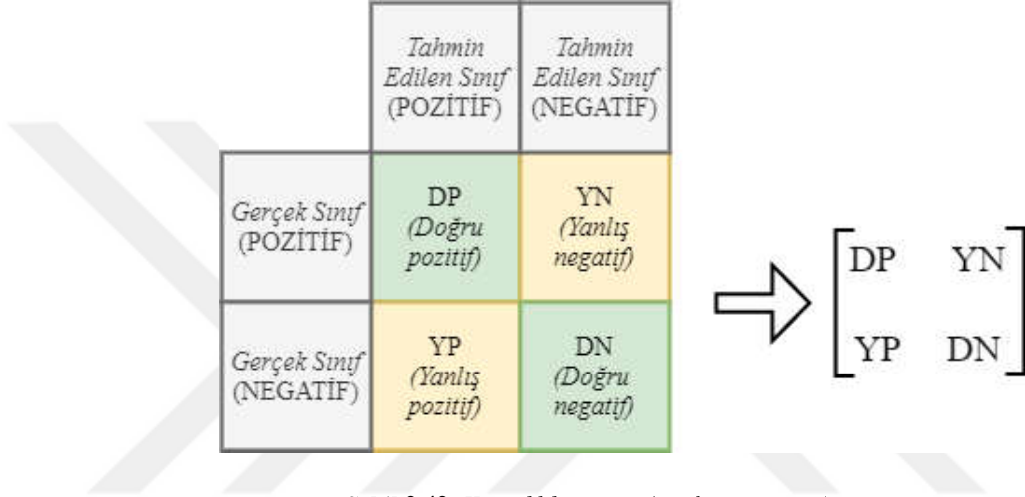
Örnek bölütleme ve anlamsal bölütleme arasında bazı farklılıklar vardır. Anlamsal bölütlemelerde, aynı sınıftan örnekler ayrılmaz. Örneğin, anlamsal bölütlemelerde *insan* sınıfına ait tüm piksellere maskede aynı renk/değer/etiket atanır. Örnek bölütlemelerde, tespit edilen nesnelere farklı değerler atanır ve onlara hangi piksellerin hangi nesneye karşılık geldiği söylenebilir.

2.12. Model Performansını Değerlendirme

YSA'da modelin performansını ölçmeye yarayan çok sayıda yöntem vardır. Bu yöntemlerin bir kısmı neredeyse tüm sınıflandırma uygulamalarında kullanılabilirken bir kısmı daha spesifik alanlarda kullanılabilir. Bu çalışma kapsamında karışıklık matrisinden elde edilen piksel doğruluğu, kesinlik (precision), recall (hassasiyet), ve F₁ skorlarının yanında anlamsal bölütleme ve alan sınıflandırmada yaygın olarak kullanılan ortalama IoU (mean IoU) ve frekans ağırlıklı IoU (frequency-weighted IoU) metrikleri kullanılmıştır (Hua ve Jégou, 2016; Long vd., 2015; X. Zhou vd., 2017). Bu metrikler ile ilgili ayrıntılar aşağıda verilmiştir.

2.12.1. Karışıklık matrisi (confusion matrix)

Karışıklık matrisi yaygın olarak kullanılmaktadır. Karışıklık matrisi, gerçek değerlerinin bilindiği bir test verisi grubundaki bir sınıflandırma modelinin (veya sınıflandırıcının) performansını ölçmek için sıklıkla kullanılan bir tablodur. Bu tablo tahmin edilen değerlerin ve gerçek değerlerin dört farklı kombinasyonunu içermektedir. Bu kombinasyonlar Şekil 2.42'de görülmektedir.



Şekil 2.42. Karışıklık matrisi (confusion matrix)

Burada;

- DP, doğru pozitif tahminlerin sayısını,
- YP, yanlış pozitif tahminlerin sayısını,
- YN, yanlış negatif tahminlerin sayısını,
- DN, doğru negatif tahminlerin sayısını göstermektedir.

Örnek olarak, bu çalışma kapsamındaki yol tespiti ele alındığında;

- Gerçekte yol sınıfına ait olup yol olarak sınıflandırılan pikseller doğru pozitif (DP),
- Gerçekte yol sınıfına ait olup yol değil olarak sınıflandırılan pikseller yanlış pozitif (YP),
- Gerçekte yol sınıfına ait olmayıp, yol sınıfına dahil edilen pikseller yanlış negatif (YN),
- Gerçekte yol olmayan pikseller yol değil olarak sınıflandırıldığında ise doğru negatif (DN) şeklinde ifade edilir.

2.12.2. Piksel doğruluğu

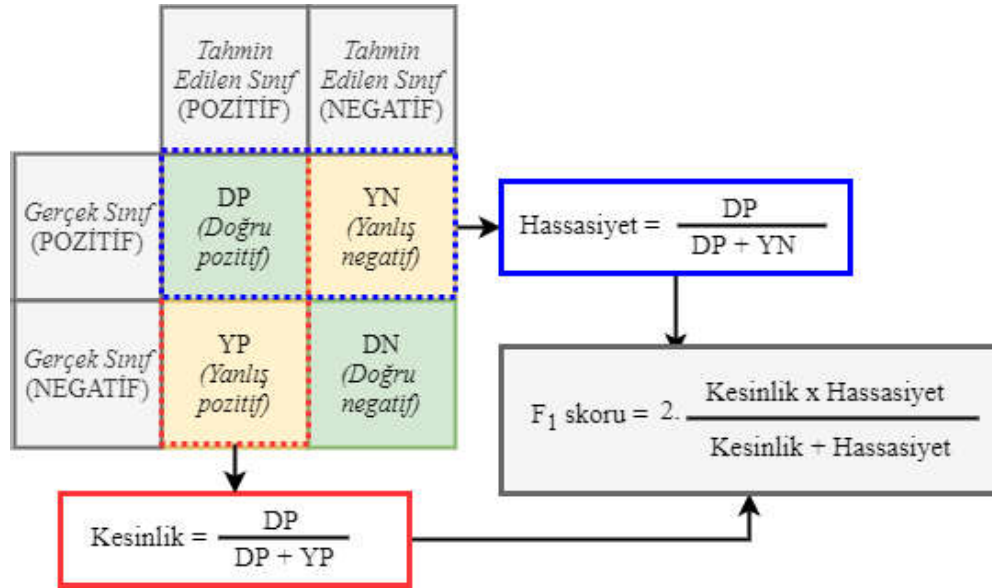
Bir anlamsal bölütleme uygulamasının performansını değerlendirmek için kullanılan diğer bir metrik, görüntüdeki doğru sınıflandırılmış piksellerin yüzdesini hesaplamaktır. Doğruluk veya doğru sınıflandırılmış öğelerin mevcut tüm öğelere oranı Denklem 2.12 ile hesaplanabilir (Larose ve Larose, 2019, s. 98,99).

$$P_a = \frac{DP + DN}{DP + DN + YP + YN} \quad (2.12)$$

Burada P_a piksel doğruluğunu ifade etmektedir. Diğer notasyonlar ise Şekil 2.42’de verilmiştir. Doğruluk, model tarafından yapılan doğru sınıflandırmaların oranının genel bir ölçüsünü temsil etmektedir (Larose ve Larose, 2019, s. 99). Bu metrik, bazen sınıfı temsil eden piksellerin sayısı görüntü içinde küçük olduğunda yanıltıcı sonuçlar verebilir.

2.12.3. Kesinlik (precision), recall (hassasiyet), ve F₁ skorları

Model tarafından pozitif olarak sınıflandırılan kayıtların gerçek oranını hesaplamak için kesinlik değeri hesaplanmaktadır. Bu metrik genellikle, duyarlılık için başka bir ad olan hassasiyet ile eşleştirilir. Kesinlik ve hassasiyet ve F₁ metriklerinin hesaplanması karışıklık matrisi üzerinde Şekil 2.43’te verilmiştir.



Şekil 2.43. Kesinlik ve hassasiyet metriklerinin karışıklık matrisindeki değerler kullanılarak hesaplanması (Widmann, 2019)

2.12.4. Ortalama IoU ve frekans ağırlıklı IoU

Jaccard endeksi olarak da adlandırılan IoU (intersection over union) metriği, temel olarak hedef maske ile tahmin edilen çıktı arasındaki çakışma yüzdesini ölçmek için kullanılan bir yöntemdir. Bu metrik, eğitim sırasında genellikle bir kayıp fonksiyonu olarak kullanılan Dice katsayısı ile yakından ilgilidir. Oldukça basit bir şekilde, IoU metriği, hedef ve tahmin maskeleri arasında ortak olan piksel sayısının her iki maskede bulunan toplam piksel sayısına bölünmesini ölçer. Matematiksel olarak ifade etmek gerekirse;

n_{ij} , sınıf j 'ye ait olduğu tahmin edilen i sınıfındaki piksel sayısı yani i hedef sınıfı olarak sınıflandırılması gerekirken j sınıfı olarak sınıflandırılan piksel sayısı;

n_{cl} , bir çalışmadaki farklı sınıfların toplam sayısı ve $t_i = \sum_j n_{ij}$, i sınıfındaki toplam piksel sayısı olsun.

Buna göre ortalama IoU Denklem 2.13 ile hesaplanmaktadır.

$$\text{Ortalama IoU} = \frac{1}{n_{cl}} \frac{\sum_i n_{ii}}{t_i + (\sum_j n_{ji} - n_{ii})} \quad (2.13)$$

Frekans ağırlıklı IoU ise Denklem 2.14 kullanılarak hesaplanmaktadır.

$$\text{Frekans ağırlıklı IoU} = \frac{(\sum_k t_k)^{-1} (\sum_i t_i n_{ii})}{t_i + (\sum_j n_{ji} - n_{ii})} \quad (2.14)$$

3. MATERYAL VE YÖNTEM

3.1. Donanım ve İşletim Sistemi

Çalışma kapsamında kodlama ve küçük testler için GNU/Linux Ubuntu 18.04 LTS işletim sistemi kurulu bir dizüstü bilgisayar kullanılmıştır. Bu bilgisayar üzerinde Intel i5 4200 işlemci, 6 GB RAM, AMD Radeon HD8670M ekran kartı ve 480 GB SSD disk vardır. Modelin eğitilmesi ve test edilmesi için ise GNU/Linux Ubuntu 18.04 LTS işletim sistemi yüklü ve konfigürasyonu Tablo 3.1'de verilen bilgisayar kullanılmıştır.

Tablo 3.1. Eğitim ve test işlemlerinin yapıldığı bilgisayar konfigürasyonu

Donanım	Özellik
İşlemci	Intel i7 6700
Bellek	16 GB DDR4
Sabit Disk	480 GB SSD Disk
Ekran kartı	Geoforce GTX 1050 TI.

3.2. Yazılım Geliştirme Araçları ve Ortamı

Bu bölümde yazılım geliştirme için kullanılan programlama dili, editör, kütüphaneler ve diğer yardımcı araçlar hakkında gerekli bilgiler verilmiştir.

3.2.1. Programlama dili

Çalışma Python programlama dilinin 3.6 sürümü kullanılarak yapılmıştır. Bununla birlikte yazılan kodlar tüm 3.x versiyonları ile uyumludur. Python, nesne yönelimli, yorumlamalı, modüler ve etkileşimli yüksek seviyeli, açık kaynak bir programlama dilidir. Python günümüzde en popüler programlama dillerinden birisidir (Galea ve Capelo, 2018, s. 6).

Python sağladığı birçok özellik ve avantajdan dolayı yapay zekâ, makine öğrenmesi ve derin öğrenme konularında en fazla kullanılan dildir (Dialani, 2019). Python dilinin bu avantajlarından dolayı çalışma kapsamında Python programlama dili kullanılmıştır.

3.2.1.1. Python kütüphaneleri

Python ortamı için yazılmış çok sayıda ücretsiz kütüphane vardır. Çalışma kapsamında kullanılan tüm Python kütüphaneleri ve sürüm numaraları Ek-1'de verilmiştir. Çalışmada yaygın olarak kullanılan kütüphanelerden birisi OpenCV (Open Source Computer Vision Library)'dir (Bradski, 2000a).

OpenCV adından da anlaşıldığı gibi açık kaynak bilgisayarlı görü kütüphanesidir. OpenCV, BSD lisansı altında yayınlanmıştır ve bu nedenle hem akademik hem de ticari kullanım için ücretsizdir. C ++, Python ve Java arayüzlerine sahiptir ve Windows, Linux, Mac OS, iOS ve Android ortamlarını desteklemektedir. OpenCV, hesaplama performansı ve gerçek zamanlı uygulamalara odaklanarak tasarlanmıştır. Temelde görüntü işleme, video yakalama, yüz algılama ve nesne algılama gibi işlemlerin kolay bir şekilde yapılmasını sağlayan bir kütüphanedir. Optimize edilmiş C / C ++ ile yazılmış olup çok çekirdekli işlemlerin avantajlarından faydalanabilir yapıdadır (Bradski, 2000b).

Ayrıca çalışma kapsamında elde edilen yüksek boyut ve çözünürlükteki uzaktan algılama görüntülerini ve vektör formatındaki OSM verilerini görüntülemek veya birlikte incelemek için açık kaynak QGIS 3.4 Maderia ve 3.10 Coruna sürümleri kullanılmıştır (QGIS Development Team, 2019).

3.2.2. Yazılım geliştirme editörü

Yazılım geliştirme editörü olarak JetBrains firması tarafından geliştirilen PyCharm 2019 topluluk sürümü kullanılmıştır (Jetbrains, 2019).

3.2.3. Kod yapısı

Çalışma kapsamında geliştirilen uygulamanın modüler yapıda olması amaçlanmıştır. Modüler programlama, modüler tasarımın genel prensibine dayanan bir yazılım tasarım tekniğidir. Modüler tasarım, karmaşık bir sistemin daha küçük parçalara veya bileşenlere, yani modüllere ayrıldığını ifade eder. Bu parçalar veya bileşenler bağımsız olarak oluşturulabilir ve test edilebilir. Birçok durumda, diğer sistemlerde de kullanılabilirler (Hare ve Kaplan, 2017). Bu çalışmada modüler programlama konseptine uygun olarak benzer işlevleri yerine getiren kod blokları aynı dosya altında farklı fonksiyonlar ile tanımlanmışlardır. Çalışma kapsamında toplam 11 adet temel modül oluşturulmuştur. Bazı modüllerin altında başka alt modüller de vardır. Oluşturulan temel modüller ve işlevleri Tablo 3.2’de verilmiştir.

Tablo 3.2. Çalışma kapsamında oluşturulan modüller ve işlevleri

Modül adı	Modül işlevi
mod_create_req_dirs	Her bir proje için ihtiyaç duyulan dizin yapısını oluşturur.
download_osm	Çevrimdışı kullanım için dünyaya ait tüm OSM verisini indirir. Ayrıca çalışma alanına ait OSM verisini tüm veriden çıkarır.
mod_get_json_meta	OAM platformundan çalışma alanına ait görüntüleri indirmek için öncelikle tüm meta bilgisini indirir, meta bilgisinden filtreleme yaparak uzaktan algılama görüntülerinin linklerini tespit eder ve son aşamada linkleri tespit edilen görüntüleri indirir.
mod_get_json_data	Overpass API kullanarak çalışma alanına ait OSM verisini indirir ve JSON formatında kaydeder. JSON verisini GeoJSON formatına çevirir. Yol genişliklerini hesaplayarak linestring verisini poligon verisine dönüştürür.
mod_rasterize_data	Çalışma alanına ait vektörel formattaki OSM verisini raster formata dönüştürür.
mod_clip_raster	Koordinat dönüşümü yapar ve her bir uzaktan algılama görüntüsü için bu görüntüye karşılık gelen raster formatında bir yol ağı görüntüsü oluşturur.
mod_create_slice_imgs	Tek parça halindeki uzaktan algılama görüntülerini ve yol ağlarını içeren maske görüntülerini istenilen boyutlardaki görüntü çiftlerine bölünmesini sağlar.
mod_delete_inapp_imgs	Veri seti için uygun ve uygun olmayan görüntüleri tespit ederek veri setini oluşturur.
mod_split_dataset	Çalışma alanına ait verileri belirlenen oranlarda eğitim ve test verisi olmak üzere ikiye böler.
mod_merge_projects	Birden fazla çalışma alanını birleştirerek derin öğrenme modeli için veri setini hazırlar.
keras_segmentation (Gupta, 2017)	Derin öğrenme modeli ile ilgili tüm tanımlamaların ve modellerin yer aldığı yapıdır. Bu yapı altında SegNet, ESA, PSPNet ve diğer birçok ağa ait bilgiler mevcuttur.

Yukarıdaki modüllerin dışında modellere ait grafiklerin çizilmesini sağlayan, görüntüleri boyutlandıran, veri seti doğrulaması yapan başka araçlar da vardır.

3.2.4. Derin öğrenme kütüphanesi

Günümüzde yaygın olarak kullanılan birkaç tane derin öğrenme kütüphanesi olmakla birlikte en yaygın kullanılan kütüphane TensorFlow'dur (Abadi vd., 2016). Keras, Caffe, Microsoft Cognitive Toolkit (CNTK), PyTorch, Apache MXnet, DeepLearning4J, Theano ön plana çıkan diğer kütüphanelerdir. Bu çalışma kapsamında Tensorflow'dan türetilmiş olan Keras kullanılmıştır. Keras API yapısının basit olması,

işlem yapmak için gereken kullanıcı eylemlerinin sayısını en aza indirmesi ve kullanıcı hatası üzerine net ve işlem yapılabilir geri bildirim sağlaması gibi özelliklere sahip olmasından dolayı tercih edilmiştir (Chollet ve vd., 2015).

3.3. Çalışma Alanı ve Konumu

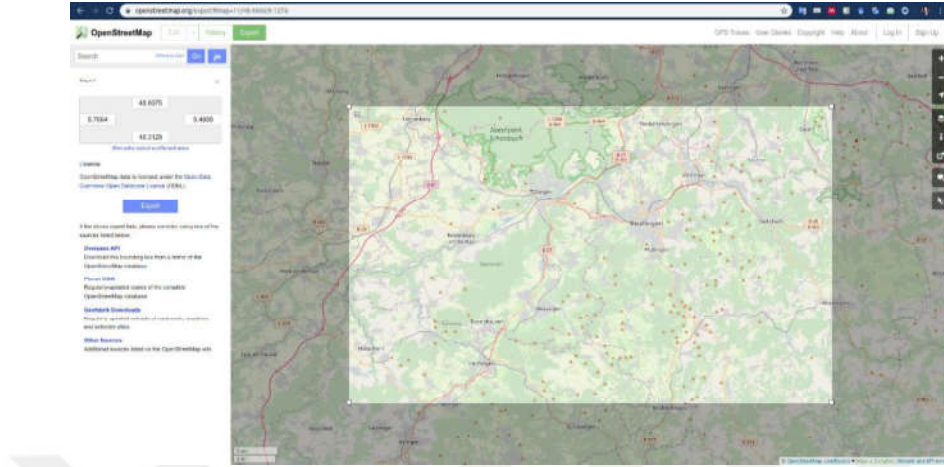
Çalışma kapsamında yeterli miktarda yüksek çözünürlüklü uzaktan algılama görüntüleri ve vektörel bilgi içeren birbirinden farklı bölgelerin seçilmesine dikkat edilmiştir. İyi bir veri seti yeterli miktarda verinin yanında verileri yeterince temsil edebilecek veri seti çeşitliliği de sağlamalıdır. Veri setindeki çeşitlilik arttıkça modelin performansı da belirli bir noktaya kadar artış göstermektedir.

Tablo 3.3'de çalışma alanlarına ait temel bilgiler verilmiştir. Burada çalışma alanlarının konumları doğrudan BBOX değerleri ile temsil edilmiştir.

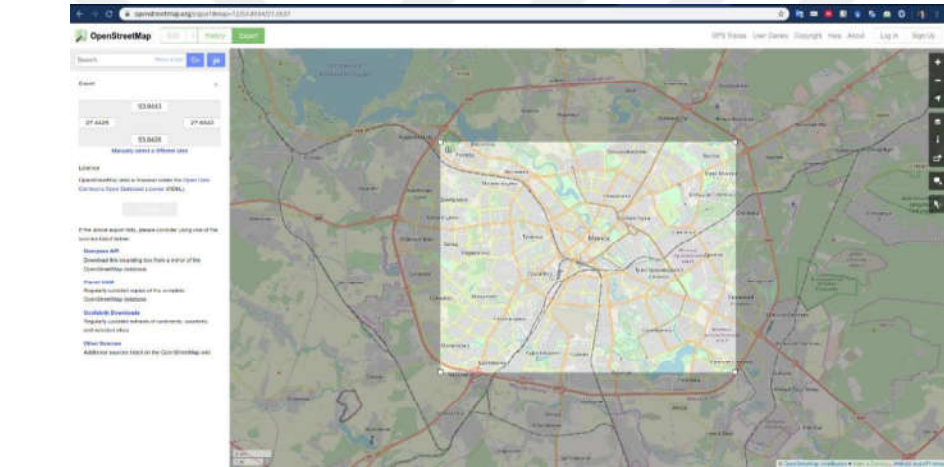
Tablo 3.3. Çalışma alanları ve sınırlayıcı kutu değerleri (BBOX)

Bölge (Şehir)	BBOX (min.enlem, min.boylam, mak.enlem, mak.boylam)
Tübingen (Almanya)	8.7664,48.3129,9.4888,48.6075
Minsk (Belarus)	27.4428,53.8426,27.6643,53.9443
Saint Petersburg Rusya)	30.0188,59.6269,31.0831,60.0168
Stara Zagora (Bulgaristan)	25.0928,42.1451,26.3617,42.7410

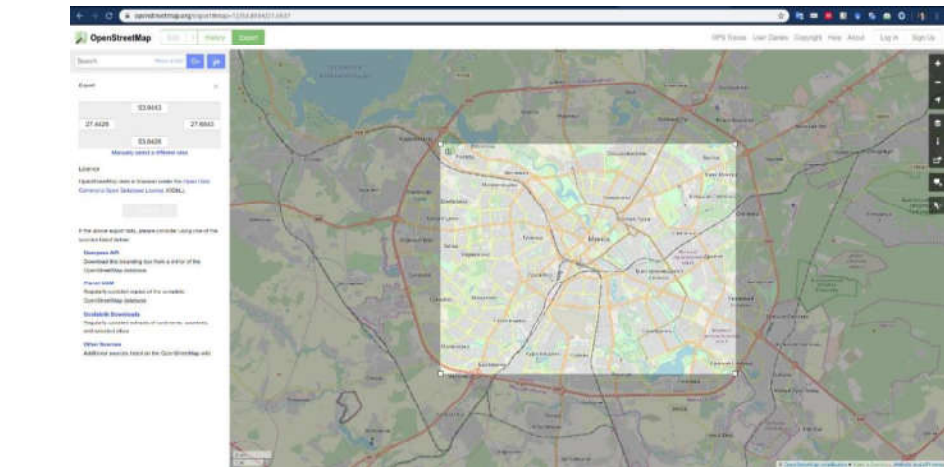
Çalışma alanı olarak belirlenen bölgelerden Minsk yüzölçümü bakımından oldukça büyük olmasının yanı sıra yüksek miktarda uzaktan algılama görüntüsü ve vektörel veri içerdiği için bu şehrin tamamı yerine bir bölümü üzerinde çalışma yapılmıştır. Bölgenin tamamı üzerinde işlem yapmak oldukça yüksek miktarda hesaplama ve depolama alanı ihtiyacı doğurmaktadır. Diğer bir çalışma alanı olan Tübingen ise hem daha küçük bir şehir olması hem de nispeten daha az veri içermesi nedeniyle genişletilerek çevresindeki bölgeler de çalışma alanına dahil edilmiştir. Saint Petersburg şehrinin ise tamamına yakını çalışmaya dahil edilmiştir. Stara Zagora'ya ait veriler sadece model sonuçlarını değerlendirmek amacıyla kullanılmıştır. Bu nedenle bu çalışma alanı ile ilgili detay verilmemiştir. Çalışma alanları OSM üzerinde Şekil 3.1'de görülmektedir.



(a)



(b)



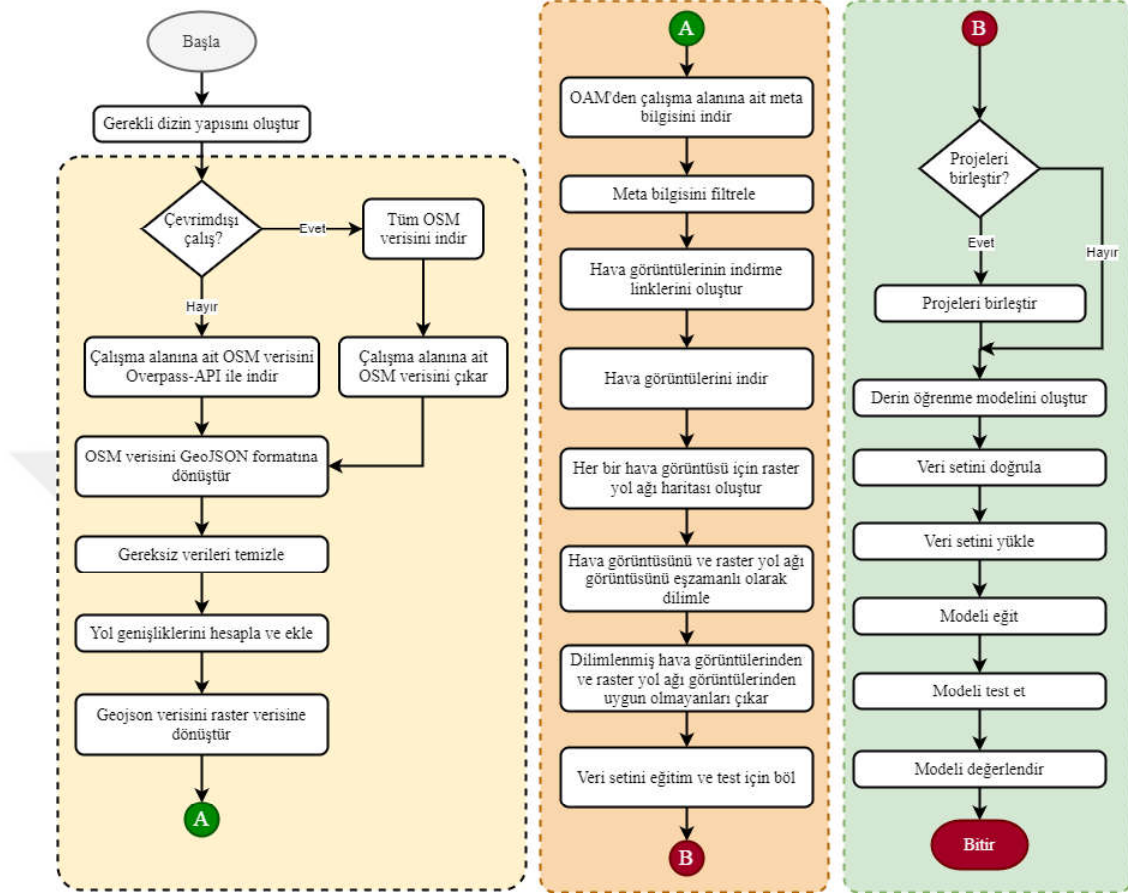
(c)

Şekil 3.1. (a) Tübingen (Almanya) (b) Minsk (Belarus) ve (c) Saint Petersburg (Rusya) çalışma alanları ve sınırlayıcı koordinat değerleri

Bilindiği gibi YSA ve derin öğrenme modellerindeki en önemli adımlardan birisi uygun veri setinin hazırlanmasıdır. Çalışma kapsamında yüksek çözünürlüklü uzaktan algılama görüntülerine ve bu görüntülere ait etiket verisine ihtiyaç duyulmaktadır. Çalışma alanlarına ait yüksek mekânsal çözünürlüklü uydu görüntüleri OAM platformundan elde edilmiştir. Etiket verisi ise OSM platformundan elde edilmiştir. Bu çalışmada ESA yöntemlerinden birisi olan anlamsal bölütleme yöntemi kullanılarak yol ağlarının tespit edilmesini sağlayan bir uygulama yapılmıştır.

3.4. Materyal ve Yöntem

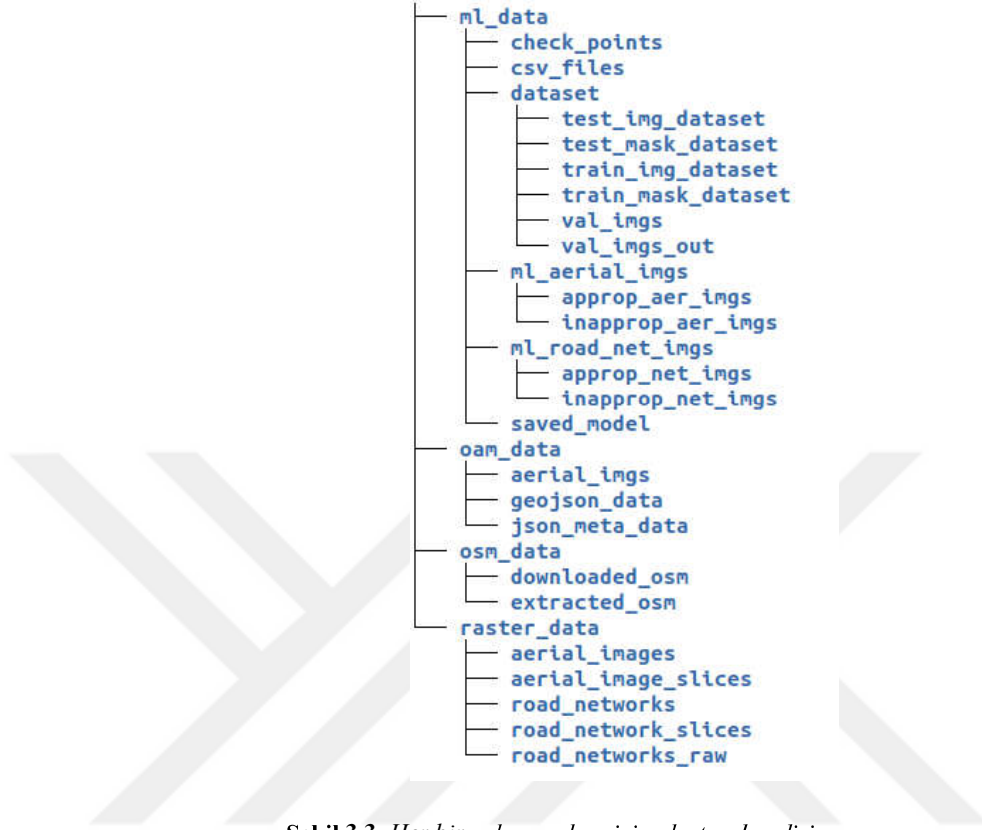
Çalışma yaygın bir şekilde kullanılan bazı anlamsal bölütleme teknikleri ile yol ağlarının tespitinin yapılmasına ve özelleştirilmiş bir derin anlamsal bölütleme modelinin oluşturulmasına dayanmaktadır. Modelin eğitilmesi, test edilmesi ve değerlendirilmesi için veriye ihtiyaç duyulmaktadır. Bu veriler, OAM ve OSM platformlarından elde edilen raster ve vektör formatındaki veriler, çeşitli işlemlerden geçirilerek modelin kullanımına hazır hale getirilmişlerdir. Bu işlemler ile ilgili ayrıntılı bilgiler ilerleyen bölümlerde yer almaktadır. Çalışma kapsamında yapılan işlemlere ait temel adımlar Şekil 3.2'deki akış şemasında yer almaktadır. Bu aşamadan sonraki açıklamalar bu alan akış şeması dikkate alınarak sunulmuştur.



Şekil 3.2. Çalışmanın tüm adımlarını içeren temel akış şeması

3.4.1. Dizin yapısının oluşturulması

Çalışma kapsamında, her bir çalışma alanı için Şekil 3.3'teki gibi bir dizin yapısı oluşturulmaktadır. Bu dizin yapısı *ml_data*, *oam_data*, *osm_data* ve *raster_data* olmak üzere dört temel dizinden oluşmaktadır.



Şekil 3.3. Her bir çalışma alanı için oluşturulan dizin yapısı

Bu dört temel dizin ise proje kapsamında ihtiyaç duyulan verileri tutmak amacıyla başka alt dizinler içermektedirler. Dizin yapısı her bir çalışma alanı için ayrı ayrı oluşturulmakta böylece her bir çalışma alanına ait veriler diğer çalışma alanı verilerinden ayrı tutulmaktadır. Diğer yandan tüm çalışma alanları için aynı dizin yapısının kullanılması ihtiyaç duyulması halinde bu alanların birleştirilmesi noktasında büyük kolaylık sağlamaktadır. Aşağıda temel dört dizine ait açıklamalar verilmiştir.

- *ml_data*, veri seti de dahil olmak üzere derin öğrenme modelinin ihtiyaç duyduğu tüm verilerin ve model tarafından üretilen check points, log gibi yeni verilerin kaydedildiği ana dizindir.
- *oam_data*, OAM platformundan indirilen meta verileri ve yüksek çözünürlüklü görüntüleri içeren ana dizindir.
- *osm_data*, OSM platformundan indirilen vektörel verileri (JSON ve GeoJSON veriler de dahil) saklayan ana dizindir.
- *raster_data*, OAM ve OSM'de indirilen verilerin modelin ihtiyaç duyduğu veri setine dönüştürülmeden önceki ham hallerinin tutulduğu ana dizindir.

3.4.2. Veri setinin hazırlanması

Veri setinin hazırlanması çoğu derin öğrenme uygulamasında en fazla emek ve zaman isteyen bölümdür. Üstelik veri setinin miktarı, kalitesi ve çeşitliliği derin öğrenme modelinin performansını doğrudan etkilemektedir (Kotsiantis vd., 2006).

Bu çalışma kapsamında veri setinin oluşturulması için sayısal hava görüntülerine ve bu görüntülere ait etiket bilgisini oluşturmak için vektörel veriye ihtiyaç duyulmaktadır. Ham olarak elde edilen uzaktan algılama görüntülerinin ve vektörel verinin çok sayıda işleminden geçirilerek derin öğrenme modelinin kullanımına uygun hale getirilmesi gerekmektedir. Aşağıda bu işlemlerin ne şekilde yapıldığı detaylı bir şekilde anlatılmıştır.

3.4.2.1. Hava görüntülerinin elde edilmesi

Çalışma kapsamında ihtiyaç duyulan sayısal hava görüntüleri OAM platformundan indirilmiştir. Bu işlemler temel olarak üç aşamadan oluşmaktadır. Bunlar, sırasıyla şu şekildedir: Öncelikle çalışma alanına ait meta bilgisi JSON formatında indirilmektedir. Meta bilgisi OAM üzerindeki tüm görüntülere ait ayrıntılı bilgiler tutmaktadır. Benzersiz dosya URL'si (UUID), başlık, projeksiyon, sınırlama kutusu, dosya boyutu, lisans meta bilgisine birer örnektir.

OAM üzerinde herhangi bir alana ait meta bilgisini elde etmek için OAM API kullanılmaktadır. OAM API'nin kullanımı temel olarak aşağıdaki gibidir.

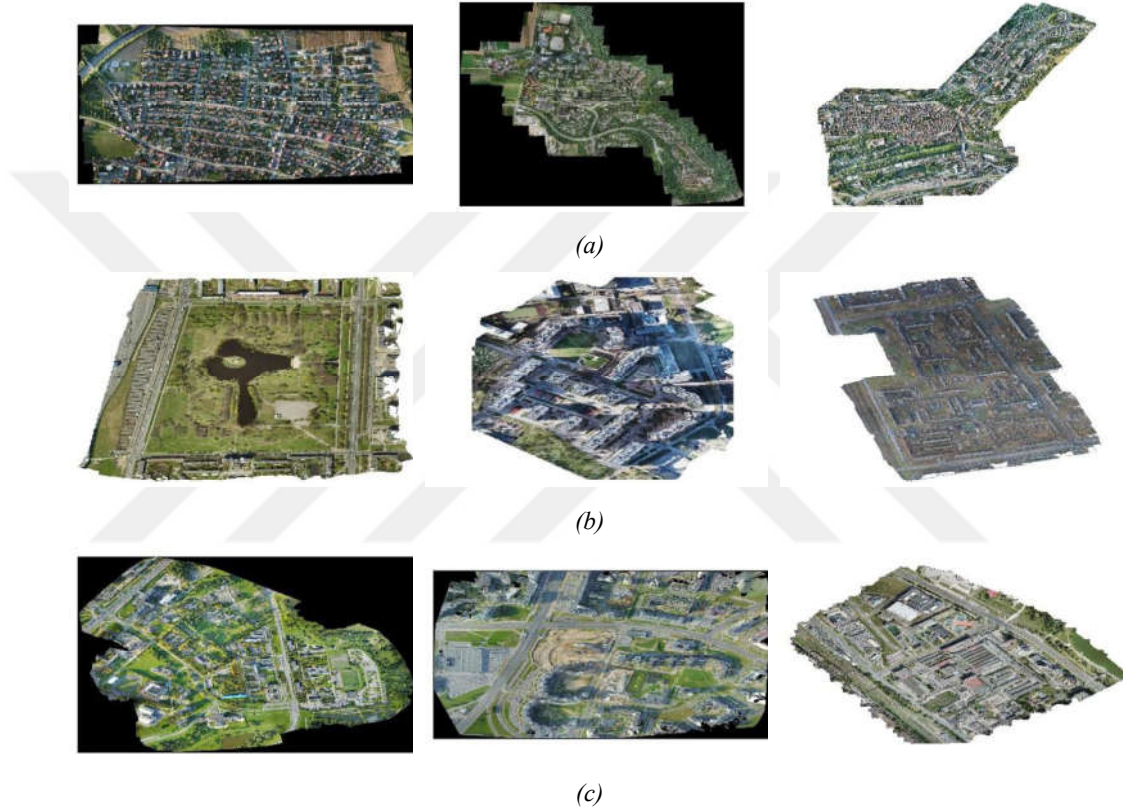
<https://api.openaerialmap.org/meta?bbox=bbox>

Burada *bbox* yerine ilgili alana ait sınırlayıcı kutu bilgileri girilmektedir. Örneğin Tübingen üzerinde belirlenen çalışma alanına ait meta bilgisi aşağıdaki komut ile elde edilmiştir.

<https://api.openaerialmap.org/meta?bbox=8.7664,48.3129,9.4888,48.6075>

Meta bilgisi çok fazla ayrıntı içermektedir. Tübingen'e ait meta bilgisinin bir bölümüne ait ekran görüntüsü Şekil 3.4'te görülmektedir. Meta bilgisinin görüntülenmesi için JSONView adlı Google Chrome eklentisi kullanılmıştır.

Bu çalışma kapsamında Tübingen için 19, Minsk için 51 ve Saint Petersburg için ise 100 adet uzaktan algılama görüntüsü indirilmiştir. Modellerin sonuçlarını görsel olarak test etmek amacıyla verileri kullanılan Stara Zagora için ise 14 adet görüntü indirilmiştir. Bu görüntülerin bir kısmı yüzde 90-98 arasındaki oranlarda küçültülerek Şekil 3.5'te her bir çalışma alanına ait üç adet görüntü (%95-98 oranında) küçültülerek verilmiştir.



Şekil 3.5. Tübingen (Almanya) ve (b) Saint Petersburg (Rusya) (c) Minsk (Belarus) çalışma alanlarına ait bazı örnek yüksek çözünürlüklü görüntüler

3.4.2.2. Vektör verinin elde edilmesi

Çalışma kapsamında etiket bilgisini içeren vektör veriler OSM'den elde edilmiştir. Vektörel veri aynı kaynaktan (OSM) olmakla birlikte çevrimiçi ve çevrimdışı olmak üzere iki farklı şekilde edinilebilir. Çevrimiçi yöntemde sınırlayıcı kutu ile belirlenen her bir çalışma alanına ait veriler yazılan betik ile Overpass API kullanılarak JSON formatında indirilmektedir. Bu yöntemde sadece çalışma alanına ait OSM verisi indirilmektedir. Diğer bir yöntem olan çevrimdışı yöntemde ise tüm dünyaya ait OSM verileri planet.osm platformundan indirildikten sonra çalışma alanına ait veri bu veriden çıkarılmaktadır. Çalışma kapsamında çevrimiçi yöntem tercih edilmiştir. Aşağıda Overpass API kullanılarak, belirlenen alana ait OSM yol verilerinin indirilmesini

sağlayan örnek sorgu kod bloğu verilmiştir. Sorgu kendisine verilen BBOX değerleri için OSM verileri üzerinde gerekli filtrelemeleri yaparak sadece istenilen verinin elde edilmesini sağlamaktadır. Elde edilen bilgiler JSON formatında kaydedilmektedir. Overpass API ile elde edilen verileri TXT, CSV gibi formatlarda kaydetmek mümkün olmasına rağmen esnek olması ve GeoJSON formatına kolaylıkla dönüştürülebilmesinden dolayı JSON formatı tercih edilmiştir.

```
def get_json_from_osm():
    print('##### GET JSON DATA FROM OAM #####')
    ordered_bbox = change_order_bbox(bbox_value=bbox)
    # Check if exists BBOX OSM Data
    exists = os.path.isfile(json_output_file_path)
    if exists:
        print(' file already exists.
              \nSkipped download JSON data from Overpass API'
              '\n'.format(json_output_file_path))
    else:
        overpass_url = "http://overpass-api.de/api/interpreter"
        overpass_query = """
        [out:json];
        // gather results
        (
        // query part for: "highway=*"
        node["highway"](""+ordered_bbox+""");
        way["highway"](""+ordered_bbox+""");
        relation["highway"](""+ordered_bbox+""");
        );
        // print results
        out body;
        >;
        out skel qt;
        """
        response = requests.get(overpass_url, params= 'data': overpass_query )
        json_data = response.json()
        save_json_file_path = os.path.join(project_name,oam_data,
                                           geojson_dir,
                                           json_output_file)

        with open(save_json_file_path, 'w') as opened_json:
            json.dump(json_data, opened_json)
        print('Downloaded json data save to : '.format(save_json_file_path))
    print('-----\n PROCESS FINISHED.\n-----')
```

Bir sonraki aşamada JSON formatındaki verinin GeoJSON formatına dönüştürülmesi gerekmektedir. JSON formatı konumsal verileri saklamak için yeterince güçlü bir format değildir. GeoJSON ise coğrafi verilerini temsil etmek için açık standartlara sahip oldukça yaygın ve esnek bir dosya formatıdır. GeoJSON, JSON dosya formatının bir alt kümesidir ve genel olarak günümüzdeki birçok programlama dili veya

yazılım tarafından herhangi bir ek işlem gerektirmeden okunabilecek yapıdadır. JSON formatındaki verilerin GeoJSON formatına dönüştürülmesi için osmtogeojson (Raifer, 2013) adlı ücretsiz ve açık kaynak araç kullanılmıştır.

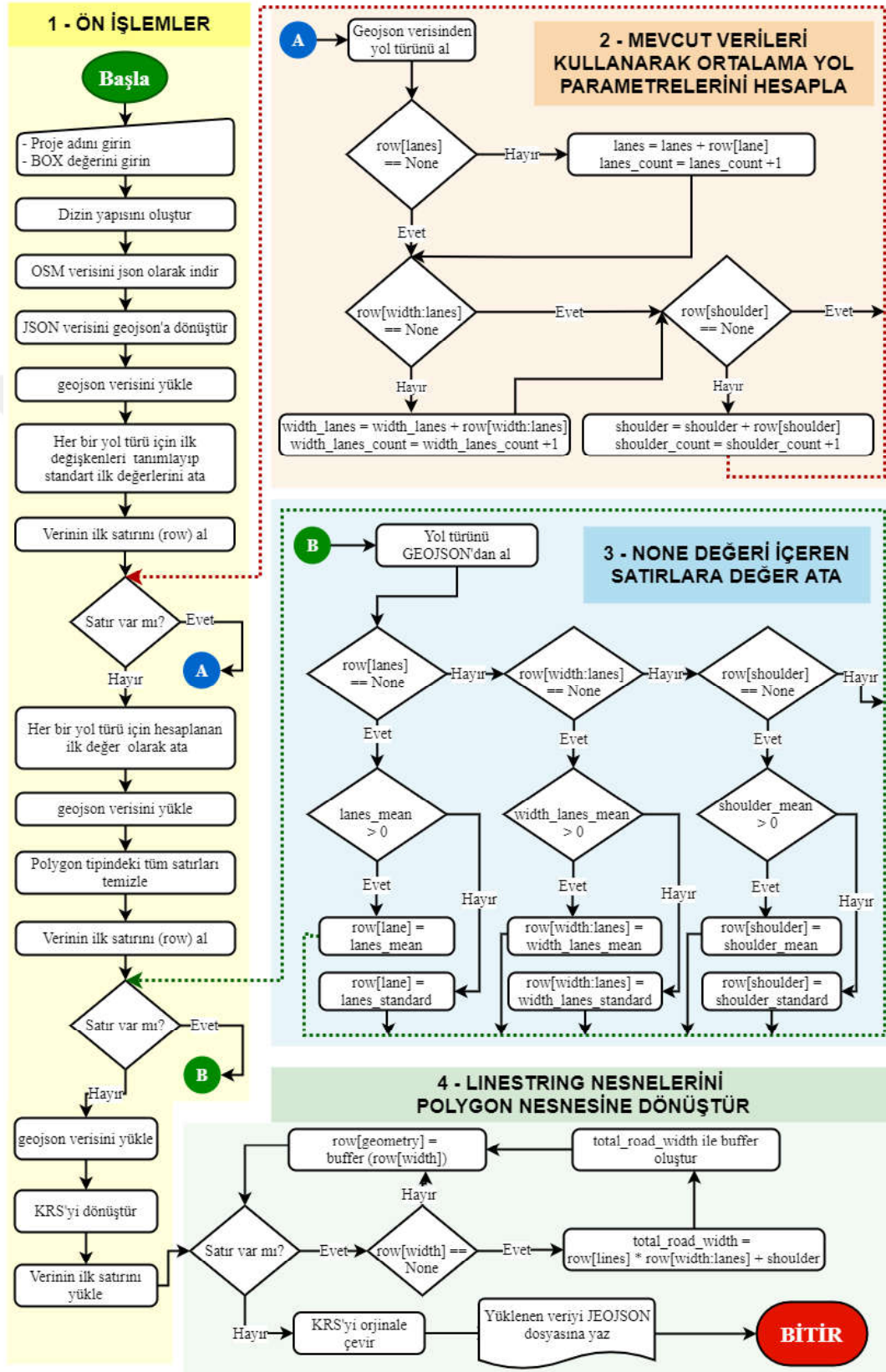
3.4.2.3. Yol genişliklerinin hesaplanması

OSM'de yol verileri vektörel olarak çizgi (linestring) adı verilen nesneler ile temsil edilmektedir. Yolların mekânsal olmayan bilgileri, etiketler kullanılarak OSM veri tabanında saklanır. OSM veri modelindeki, yollar dahil olmak üzere her nesne için çok sayıda etiket tanımlanmış olsa da veri girişi yapan çoğu kullanıcı birçok etiket bilgisi için veri girişi yapmaz. Bu durumun iki temel nedeni vardır. Birincisi, kullanıcılar nesne hakkında gerekli bilgilere sahip olmalarına rağmen, veri giriş işleminin zaman almasından dolayı zorunlu olmayan birçok etiketi boş bırakmaktadırlar. Diğer neden ise kullanıcıların ilgili nesne hakkında gerekli tüm bilgilere sahip olmamalarıdır. Sonuç olarak, OSM'deki yol ağı gerçek dünyadaki yol ağını tam olarak temsil edememektedir.

Bu bölümde, genişlik bilgisine sahip olmayan ve çizgilerle temsil edilen OSM yol verilerinin gerçek yol genişlikleri içeren poligon nesnelere dönüştürülmesi için yapılması gereken işlemler anlatılmaktadır. Dönüşüm işlemlerinin nasıl gerçekleştirildiği Şekil 3.6'da gösterilen akış şemasında adım adım anlatılmıştır.

Akış şemasında;

- *lanes_mean*, ortalama şerit sayısını,
- *width lanes_mean*, ortalama şerit genişliğini,
- *shoulder_mean*, ortalama güvenlik şeridi genişliğini,
- *lanes standard*, standart şerit sayısını,
- *width lanes standard*, standart şerit genişliğini,
- *shoulder standard*, standart güvenlik şeridi genişliğini,
- *total road width*, nihai yol genişliğini ifade etmektedir.



Şekil 3.6. Yol genişliklerinin hesaplanması için yapılan işlemlere ait akış şeması

3.4.2.4. Varsayılan yol genişliklerinin belirlenmesi

Bir yolun genişliğini hesaplamak için temelde üç veri gerekir. Bunlar; şerit sayısı, şerit genişliği ve (varsa) emniyet şeridinin genişliğidir. Tüm ülkelerdeki veya bölgelerdeki yollar için farklı yol türlerine göre, standart şerit sayısını ve şerit genişliğini belirlemek mümkün değildir. Aynı ülkenin farklı bölgelerinde bile, aynı yol türü için şerit sayısı ve yol genişliği değişebilir. Öte yandan, yol türü, yolun yapıldığı tarih, hız sınırı, trafik yoğunluğu, yolu kullanan araçların türü gibi yol genişliğini etkileyen birçok faktör vardır.

Aşağıda örneklendiği gibi, yerel veya küresel düzeyde faaliyet gösteren farklı kurumlar/kuruluşlar aynı veya benzer yol türleri ve araç genişlikleri/uzunlukları hakkında farklı bilgiler sunmaktadırlar.

- 96/53/EC sayılı AB Konseyi Direktifi, ulusal ve uluslararası karayolu, araç ve trafik kurallarını düzenleyen bir dizi yönetmelik içermektedir. Bu yönetmeliğe göre, bir aracın maksimum genişliği 2.55, maksimum uzunluk 18.75 ve bir şeridin normal genişliği 3.65 metredir (Official Journal of the European Communities, 1996).
- Yeniden hizalanmış veya yönlendirilmiş kırsal yollar, en azından iki aracın aynı anda güvenli bir şekilde düşük hızda seyahat etmesini sağlayan genişlikte olmalıdır. Bu genişlik minimum 5.5 metredir. Bu asgari genişlik, otobüsler veya ağır vasıtalar tarafından kullanılmayan yollarda 6.0 metre, otobüslerin veya ağır taşıtların düzenli aralıklarla geçebileceği yollar için 6.8 metre olmalıdır (Fearnley, 2015).
- İki şeritli yollarda, ana yol genişliği maksimum 7.3 metre olmalıdır. Ek olarak, virajlarda gerekebilecek her türlü genişleme de eklenmelidir (Fearnley, 2015).
- Temel güvenlik nedenleriyle, tek şeritli yolların genişliği en az 3.5 metre olmalıdır. Bu genişlik trafiğin az olduğu ve araçların birbirini geçemediği yollar için uygundur (Fearnley, 2015).

Tüm yolların şerit sayısını ve şerit genişliğini hesaplamak zor olduğu için, yerel ve küresel düzeyde yollar ile ilgili araştırmalar yapan veya standartlar belirleyen kurumların sundukları bilgiler referans amacıyla kullanılabilir. Örneğin, Ulusal Bilim ve Mühendislik Akademileri Ulaşım Araştırma Kurulu (TRB), Amerika Birleşik Devletleri'nde trafik operasyonel analizi, metodolojiler ve hizmet düzeyi (LOS)

kavramlarının birincil referansı olan Karayolu Kapasite El Kitabı (HCM) adlı bir belge yayınlamaktadır. Bu belge, otoyol kapasitesinin ve hizmet kalitesinin analizinde bir rehber olarak ABD ve diğer ülkelerde kullanılan en önemli teknik belge haline gelmiştir. Amerika Birleşik Devletleri'nde tüm federal destekli karayolu projeleri için kullanılması zorunludur (Roess ve Prassas, 2014).

HCM ve Transit Kapasitesi ve Hizmet Kalitesi El Kitabı gibi diğer yayınlar, planlama veya ön mühendislik analizlerinde yardımcı olmak için gereken girdilerin bazıları için varsayılan değerler sağlar. Dolayısıyla yukarıda sunulan bilgiler göz önüne alınarak yol türüne bağlı olarak şerit ve yol genişlikleri için standart değerler belirlemek mümkündür.

Bu çalışmada, yol için şartname ve standartları belirleyen kurumların sağladığı bilgiler dikkate alınarak OSM'deki yol türleri için şerit sayısı, şerit genişliği ve güvenlik şeridi genişliği Tablo 3.4'teki gibi belirlenmiştir (England Highways, 2012; Fearnley, 2015; Gross vd., 2009).

Tablo 3.4. OSM'deki yol türleri için belirlenen standart değerler

Yol Türü	Şerit Sayısı	Şerit Genişliği	Bölünmüş Yol	Güvenlik Şeridi	Toplam Yol Genişliği
Motorway	3	3.6	Evet	1.8	12.6
Trunk	3	3.6	Evet	1.2	12.0
Primary	2	3.4	Hayır	0	6.8
Secondary	2	3.0	Hayır	0	6.0
Tertiary	2	2.8	Hayır	0	5.6
Unclassified	1	4.2	Hayır	0	4.2
Residential	1	4.2	Hayır	0	4.2
Service	1	3.8	Hayır	0	3.8

Her bir yol türü için ortalama genişlik Denklem 3.1 kullanılarak hesaplanmıştır.

$$w = l + l_w + s \quad (3.1)$$

Burada, w yolun toplam genişliğini, l şerit sayısını, l_w şerit genişliğini ve s güvenlik şeridinin genişliğini ifade etmektedir.

3.4.2.5. Hesaplama adımları ve ayrıntıları

Şimdiye kadar, çalışma alanlarına ait OSM verilerinin elde edilmesi, elde edilen OSM verilerinin GeoJSON formatına dönüştürülmesi ve eksik yol verileri için standart değerlerin belirlenmesi hakkında gerekli bilgi ve açıklamalar verilmiştir. Bir sonraki adım, yolları temsil eden ve genişlik bilgisi içermeyen linestring verilerinin genişlik bilgisi içeren poligonlara dönüştürülmesi işlemidir. Bu işlemlerin ne şekilde yapıldığı aşağıda açıklanmıştır.

- GeoJSON formatındaki OSM verileri Geopandas kullanılarak sisteme yüklenmiştir.
- Aynı veya yakın bölgedeki aynı yol türlerinin benzer yol parametrelerine sahip olması olağan ve beklenen bir durumdur. Dolayısıyla, her bir yol türü için, mevcut tüm veriler kullanılarak ortalama şerit sayısı, ortalama şerit genişliği ve ortalama güvenlik şeridi genişlikleri aşağıdaki denklemlerin sırasıyla kullanılmasıyla hesaplanmıştır.

$$\text{ortalama şerit sayısı} = \frac{1}{n} \sum_{i=1}^n l_i, l_i \neq \text{None ve } n \geq 1 \quad (3.2)$$

Burada her bir yol türü için ayrı ayrı olmak üzere; l_i , şerit sayısını ve n toplam şerit sayısını ifade etmektedir.

$$\text{ortalama şerit genişliği} = \frac{1}{n} \sum_{i=1}^n w_i, w_i \neq \text{None ve } n \geq 1 \quad (3.3)$$

Burada her bir yol türü için ayrı ayrı olmak üzere; w_i , şerit genişliği ve n , toplam şerit sayısını göstermektedir.

$$\text{ortalama güven.şeridi geniş.} = \frac{1}{n} \sum_{i=1}^n S_i, S_i \neq \text{None ve } n \geq 1 \quad (3.4)$$

Burada her bir yol türü için ayrı ayrı olmak üzere; S_i , şerit genişliğini ve n toplam şerit sayısını ifade etmektedir.

Bu hesaplamalarla, çalışma alanına ait tüm veriler kullanılarak her bir yol türü için ortalama şerit sayısı, ortalama şerit genişliği ve ortalama güvenlik şeridi genişliği hesaplanmıştır. Bununla birlikte, OSM'den indirilen vektörel veri içerisinde bir veya birden fazla yol türüne ait bir ya da birden fazla parametreye ilişkin hiçbir bilginin

olmaması durumunda bazı yol türleri için bu değerlerden biri veya daha fazlası hesaplanamayabilir. Bu durumda, yukarıdaki denklemler bu yol tipleri için karşılık gelen değerleri hesaplayamaz. Örneğin, mevcut tüm verilerde primary yol türü için şerit genişliği değeri yoksa, bu yol türü için ortalama şerit genişliği hesaplanmaz.

- Çalışma kapsamında elde edilen OSM verileri WGS-84 (EPSG:4326) koordinat sistemini kullanmaktadır. Bu koordinat sisteminin varsayılan ölçü birimi derecedir. Metre cinsinden poligon genişliklerini hesaplamak için, OSM verileri Rasterio kütüphanesi kullanılarak ölçü birimi metre olan EPSG:3395 koordinat sistemine dönüştürülmüştür.
- Sonraki aşamada her bir yol türü için eksik olan yol verilerinin tamamlanması işlemi yapılmıştır. Çalışma alanına ait mevcut OSM verisi Geopandas ile sisteme yüklenmiştir. Daha sonra Denklem 3.2, 3.3 ve 3.4 kullanılarak her bir yol türü için ortalama şerit sayısı, ortalama şerit genişliği ve ortalama güvenlik şeridi genişlikleri hesaplanmıştır. Daha sonra her bir yol verisi satır satır ele alınmış ve eksik olan yol parametreleri daha önce hesaplanan ortalama değerler kullanılarak tamamlanmıştır. Eğer herhangi bir yol türü için ortalama şerit sayısı, ortalama şerit genişliği ve ortalama güvenlik şeridi genişliklerinden biri veya birden fazlası hesaplanamadıysa bu durumda Tablo 3.4'teki standart değerler kullanılarak eksik parametreler tamamlanmıştır.
- Sonraki aşamada Denklem 3.5 kullanılarak toplam yol genişliği hesaplanmıştır.

$$\text{ortalama yol genişliği} = \frac{1}{n} \sum_{i=1}^n (l_i \cdot w_i) + s_i \quad (3.5)$$

Burada her bir yol türü için ayrı ayrı olmak üzere;

l_i , şerit sayısını,

w_i , şerit genişliğini

s_i , güvenlik şeridi genişliğini

n , toplam şerit (toplam satır) sayısını ifade etmektedir.

- Buraya kadar yapılan işlemler ile tüm yol türleri için toplam yol genişliğini temsil eden değerler hesaplanmıştır. Bu değerler, buffer (tampon) işlevinde kullanılarak elde edilen yol genişliğinin poligon nesnesine dönüştürülmesi için kullanılmıştır.

Bu işlemdeki amaç, bir nokta, çizgi veya çokgen gibi seçilen vektör türündeki bir nesne etrafında belirtilen genişlikte bir bölgesel alan oluşturmaktır. Çalışma alanına ait OSM verilerinden yol bilgisi içeren linestring nesneler filtrelenmiştir. Daha sonra bu nesneler vektör tamponu kullanılarak poligonlara dönüştürülmüştür. Bu nesnelere uygulanan tampon alanının genişliği, çalışma alanından elde edilen veriler Denklem 3.5 kullanılarak hesaplanmıştır. Sonuç olarak, yolları temsil eden linestring nesneler, önceden hesaplanan yol genişliği değerlerini tampon işleminden geçirerek gerçek yol genişliğini içeren poligonlara dönüştürülmüştür. Şekil 3.7'de Tübingen çalışma alanının bir bölümüne ait ham yol verileri görülmektedir.



Şekil 3.7. Tübingen çalışma alanına ait ham yol ağı verileri

- Burada henüz yol genişlikleri ile ilgili hiçbir işlem yapılmamıştır. Sadece QGIS programı, JSON verilerini okuyamadığı için bu veriler GeoJSON formatına dönüştürüldükten sonra QGIS ile açılmıştır. Görüldüğü gibi burada yol verileri yol türü ayırt etmeksizin çizgiler ile temsil edilmektedir. Bu gösterimde OSM'nin standart haritası altlık harita olarak kullanılmıştır. Ayrıca çizgilerin daha net görülmesini sağlamak amacıyla çizgi kalınlığı artırılmıştır.
- Son adımda, veriler yeniden koordinat dönüşümü ile EPSG:4326 sistemine dönüştürülmüşlerdir. Elde edilen bu yeni veriler bir GeoJSON dosyasına

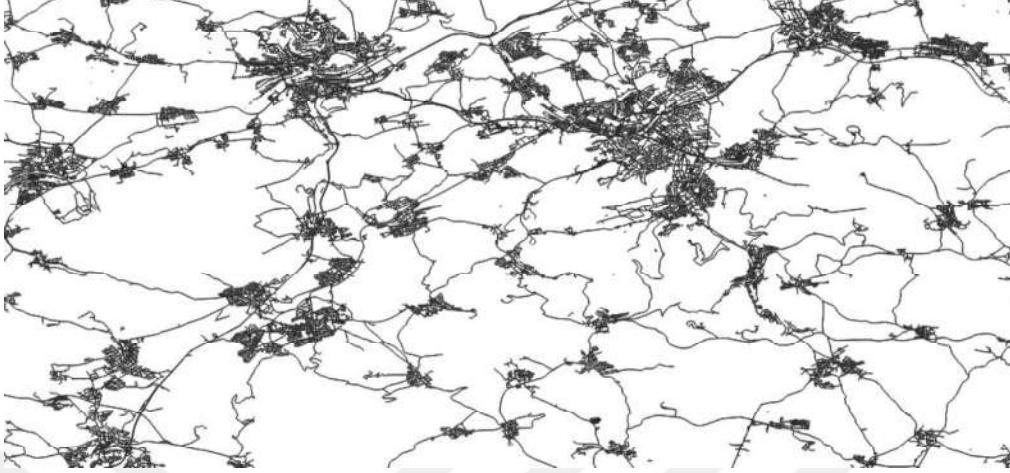
kaydedilmiştir. Tüm bu işlemlerden sonra artık yol genişlikleri ile birlikte sadece yol ağlarını içeren vektörel veri elde edilmiştir.

Şekil 3.8'de Tübingen çalışma alanının bir bölümüne ait yol ağlarını içeren vektörel veriler görülmektedir. Görüldüğü gibi yollar yol türüne göre farklı genişliklere sahip poligon nesneleri ile temsil edilmektedir. Burada yol ağı OSM altlık haritası kullanılarak gösterilmiştir. Diğer yandan Şekil 3.7'de daha fazla yol verisinin gözükmemesinin temel nedeni daha önce de ifade edildiği gibi bu çalışmanın OSM'deki temel yol türleri (Tablo 3.4) ile sınırlandırılmış olmasıdır.



Şekil 3.8. Tübingen çalışma alanına ait nihai yol ağı verileri

Yol genişliklerinin oluşturulması sürecinde aynı zamanda gereksiz olan tüm veriler temizlenmektedir. Şekil 3.9'da Tübingen çalışma alanına ait vektör formatındaki yol ağı verisi hiçbir altlık harita kullanılmadan verilmiştir. Görüldüğü gibi yol ağı dışında başka hiçbir veri yoktur. Gereksiz verilerin temizlenmesi ile veri işleme hızı önemli ölçüde artmıştır.



Şekil 3.9. Tübingen çalışma alanına ait nihai yol ağının hiçbir altık harita kullanılmadan gösterimi

3.4.2.6. Vektör verilerin raster veriye dönüştürülmesi

CBS'de verileri ifade etmek için vektör ve raster olmak üzere iki temel veri modeli vardır. Bu iki modelin kendilerine göre avantajları ve dezavantajları vardır. CBS'de vektör ve raster veriler arasında dönüştürme işlemleri yapmak mümkündür.

Bu çalışma kapsamında, çalışma alanındaki tüm yol ağı verileri GeoJSON formatında vektör veri olarak tutulmaktadır. Veri setindeki maske görüntülerini oluşturmak için bu verilerin raster verisine dönüştürülmesi gerekmektedir. Bu işlem Rasterio adındaki Python kütüphanesi kullanılarak yapılmıştır. Dönüştürme işleminde koordinat sistemi bilgisinin korunmasına dikkat edilmiştir. Dönüştürme işleminden sonra elde edilen raster veri TIF dosyası olarak kaydedilmiştir. Kavramların birbirleri ile karışmasını önlemek adına bu dosya *temel raster görüntü* olarak ifade edilmiştir. Vektör verinin raster veriye dönüştürülmesini sağlayan kod bloğu aşağıda verilmiştir.

```
def rasterize_data():
    print('##### RASTERIZE VECTOR DATA (GEOJSON to TIF) #####')
    # Load some sample data
    print('Loading vector data in geojson for rasterization.')
    geojson_data = gpd.read_file(os.path.join(project_name, oam_data, geojson_dir,
                                              project_name + '.geojson'))
    print('Loaded data CRS : ', geojson_data.crs)

    # lose a bit of resolution, but this is a fairly large file,
    # and this is only an example.
    print('Performing data conversion. Please wait...')
    shape = 30000, 30000
    transform = rasterio.transform.from_bounds(*geojson_data['geometry'],
                                              total_bounds, *shape)
    rasterize_data = rasterize(
        [(shape, 1) for shape in geojson_data['geometry']],
```

```

out_shape=shape,
transform=transform,
fill=0,
all_touched=True,
dtype=rasterio.uint8)
print('Writing converted data to .tif file. Please wait...')

if os.path.exists(os.path.join(project_name, raster_data,
                                project_name + '_full_base.tif')):
    print('already exist.')
else:
    with rasterio.open(os.path.join(project_name,
                                      raster_data,
                                      project_name + '_full_base.tif'), 'w',
                        driver='GTiff',
                        dtype=rasterio.uint8,
                        count=1,
                        width=shape[0],
                        height=shape[1],
                        transform=transform,
                        crs=geojson_data.crs,
                        compress='lzw'
                        ) as dst:
        dst.write(rasterize_data, indexes=1)
    # plot_image(input_data = rasterize_data)
    print('Raster data saved as ' +
          format(os.path.join(project_name,
                                raster_data, project_name + '_full_base.tif')))
    print('-----\n PROCESS FINISHED.\n-----')

```

Dönüştürme işleminden sonra her bir çalışma alanı için birer *temel raster görüntü* oluşmuştur. Bu görüntüler çalışma alanlarına ait tüm yol ağlarını içermektedirler. Bu görüntüler 0 ve 1 piksel değerlerine sahip ikili (binary) birer dosyadır. Bu görüntülerde yol ağlarını içeren pikseller 1 ve diğer pikseller 0 değerini almışlardır. Bu yapıdaki görüntülerde herhangi bir işlem yapmadan insan gözü ile yol ağı bilgisini içeren ve içermeyen görüntü bölümlerini ayırt etmek oldukça zordur. Şekil 3.10'da çalışma alanlarına ait tüm yol ağlarını içeren *temel raster görüntüler* yer almaktadır. Verilen ekran görüntüleri, ilgili raster görüntülerin QGIS programında açılması ile elde edilmiştir.



(a)



(b)



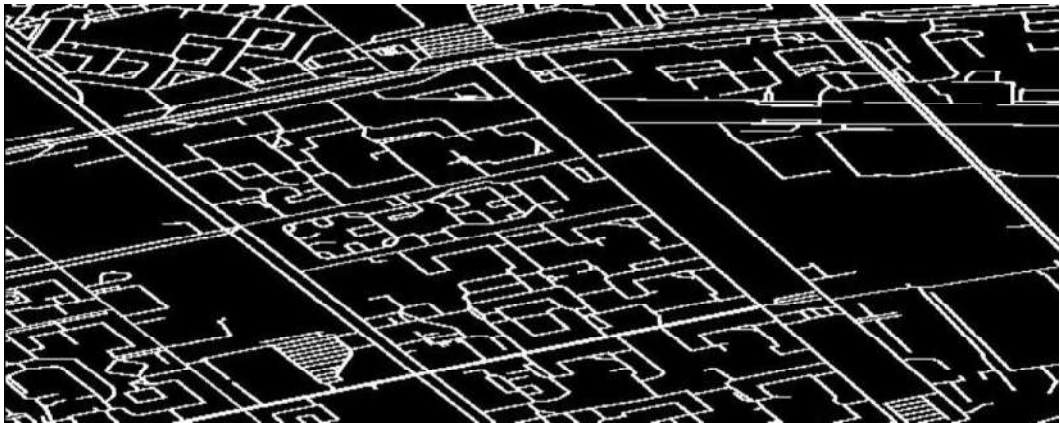
(c)

Şekil 3.10. (a) Tübingen (b) Minsk (c) Saint Petersburg çalışma alanlarına ait tüm yol ağlarının raster formatına dönüştürülmüş yol ağı haritaları

Bu aşamada yapılması gereken bir diğer işlem ise her bir gerçek uzaktan algılama görüntüsü için bu görüntü ile aynı boyutlara sahip yol ağlarını içeren birer maske görüntüsü oluşturmaktır. Oluşturulan görüntüler tüm yol ağlarını içeren Şekil 3.10'daki temel raster görüntülerden ilgili bölümlerin kesilip yeni bir dosya olarak saklanması işlemi ile oluşturulmaktadır. İşlem sonunda uzaktan algılama görüntüsü sayısı kadar ve bu dosyalar ile aynı koordinat sistemine sahip sadece yol ağlarını içeren ikili maske görüntüler oluşmaktadır. Şekil 3.11'de Saint Petersburg'a ait bir uzaktan algılama görüntüsü ve bu görüntü için oluşturulmuş yol ağlarını içeren ikili raster görüntü görülmektedir.



(a)



(b)

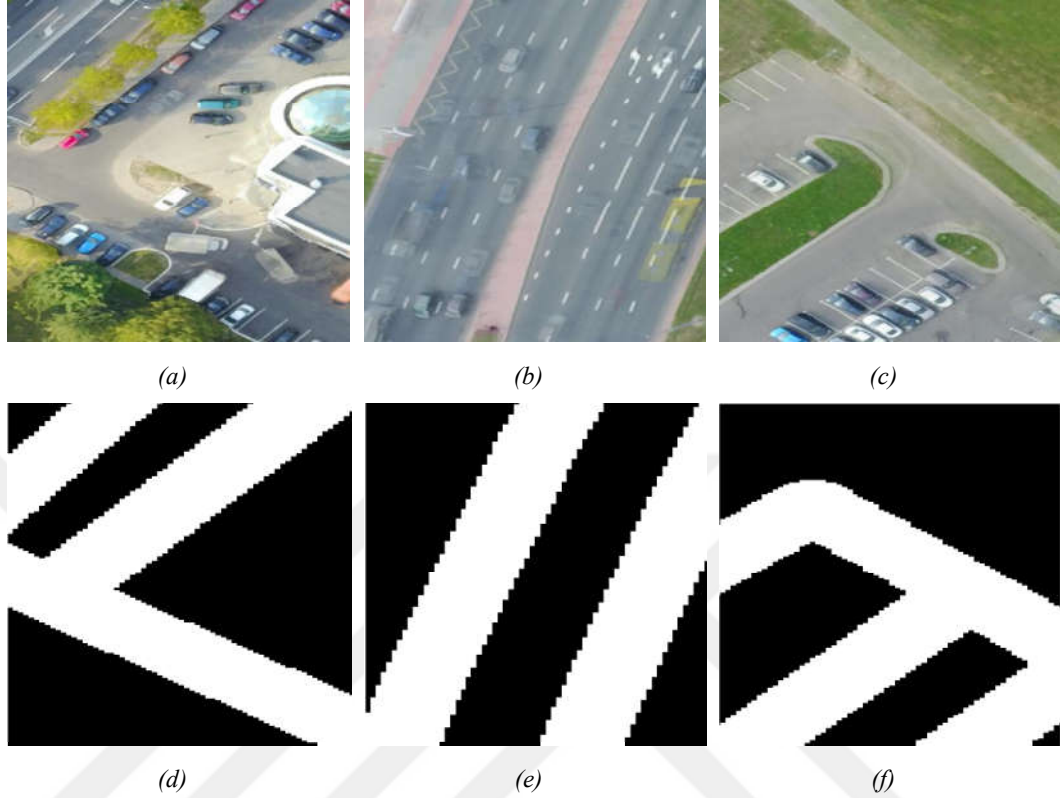
Şekil 3.11. (a)Saint Petersburg'a ait bir uzaktan algılama görüntüsü ve (b) bu görüntü için üretilmiş yol ağlarını içeren raster görüntü

Şekil 3.11'den de görüldüğü gibi üretilen ikili raster görüntü gerçek görüntü ile birebir aynı boyutlara sahip değildir. Üretilen görüntü sadece kare veya dikdörtgen gibi düzgün bir geometri ile üretilebilmektedir. Bu durumda gerçek görüntünün şekline bağlı olarak görüntünün bir bölümü tam anlamıyla üretilen görüntüde temsil edilememektedir. Diğer yandan Şekil 3.11'deki görüntüler dikkatle incelendiğinde gerçek görüntüde (a) olmayan bazı yol ağlarının yol ağlarını içeren görüntüde (b) olduğu görülmektedir. Bunun nedeni her iki görüntünün farklı kaynaklardan elde edilmesidir. Bu durum veri setinin oluşmasında bir sorun teşkil etmemektedir. Çünkü iki görüntünün boyutları aynıdır ve aşağıda da açıklandığı gibi bu görüntülerin parçalanması eş zamanlı olarak yapılmaktadır.

3.4.2.7. Görüntülerin dilimlenmesi

OAM'dan indirilen TIF görüntülerinin mekânsal çözünürlükleri ve dolayısıyla boyutları oldukça yüksek olduğundan bu görüntüler üzerinde işlem yapmak oldukça zordur. Diğer yandan derin öğrenme algoritmasında kullanılacak görüntülerin 256x256, 512x512 veya 1024x1024 gibi daha küçük boyutlarda olması gerekmektedir. Dolayısıyla bu çalışma kapsamında gerçek görüntüler ve yol ağlarını içeren ikili görüntüler eş zamanlı olarak belirlenen boyutlarda dilimlenmektedir. Şekil 3.11'deki gibi gerçek bir uzaktan algılama görüntüsü ve bu görüntü ile aynı boyutlara sahip yol ağlarını içeren bir görüntüden oluşan toplam 170 adet görüntü çifti (Tablo 3.5) eş zamanlı olarak 512x512 boyutlarında bölünmüşlerdir (dilimlenmişlerdir). Daha yüksek bir boyut, çoğu derin öğrenme modeli için yüksektir. Daha küçük boyutlardaki görüntülerde ise yol ağlarını içeren görüntülerin sayısı azalmakta veya yolun sadece belirli bir kısmı alınabilmektedir.

İki görüntünün 512x512 boyutlarında eş zamanlı olarak dilimlenmesini sağlayan kod bloğu aşağıdadır. Bu kod bloğu her bir uzaktan algılama görüntüsü için bir dizin oluşturmakta ve dilimlenmiş bu görüntüleri ilgili dizinin içinde saklamaktadır. İşlem sonunda derin öğrenme modelinin ihtiyaç duyduğu görüntü ve bu görüntülere ait etiket bilgisini içeren maske görüntüler oluşturulmuştur.



Şekil 3.12. (a), (b), (c) çalışma kapsamında elde edilen 512x512 boyutlarına sahip gerçek görüntüler ve (d), (e), (f) ile sırasıyla bu görüntülere karşılık gelen yol bilgilerini içeren ikili görüntülerdir

Şekil 3.12'de elde edilen bazı örnek görüntüler ve bu görüntülere karşılık gelen maske görüntüler yer almaktadır.

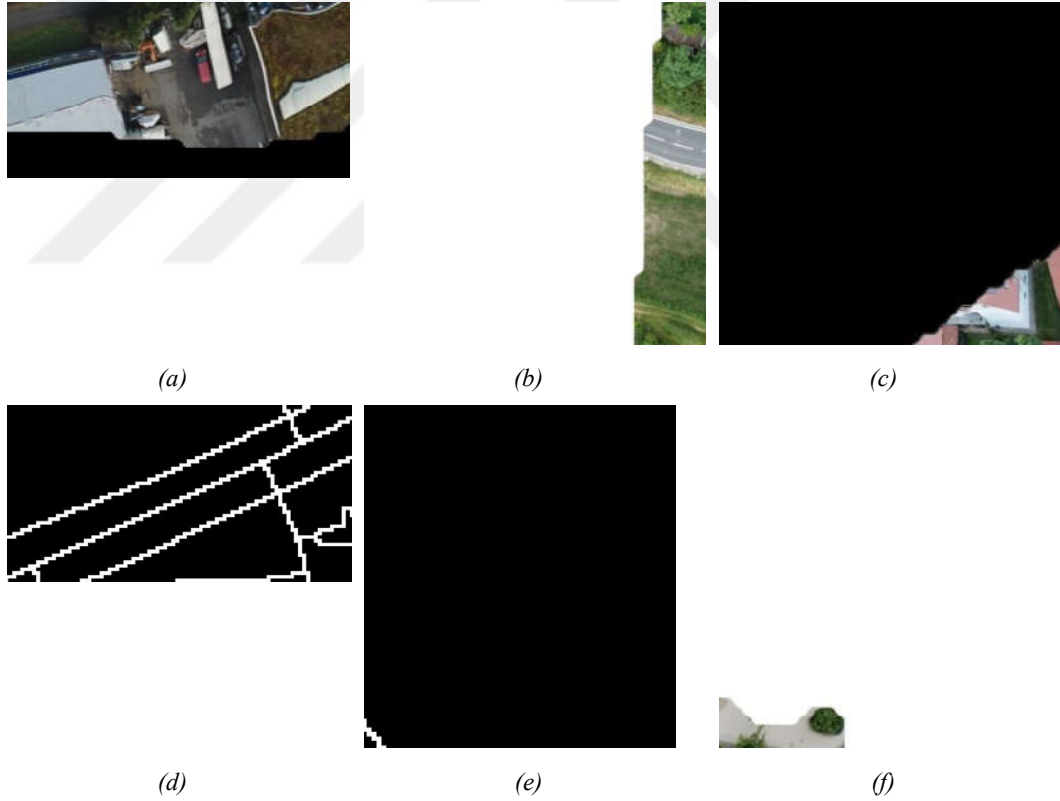
3.4.2.8. Uygun olmayan görüntülerin filtrelenmesi

Dilimleme işleminden sonra oluşan gerçek veya maske görüntülerinin bir kısmı veri setine dahil edilmek için uygun değildir. Uygun olmayan görüntülerin oluşma nedenleri temel olarak şunlardır:

- Aynı bölge için farklı kaynaklardan elde edilen verilerin bazı noktalarda farklılıklar göstermesi.
- Uzaktan algılama görüntülerinin hemen hemen hepsinde kenar bölgelerinde görüntü bilgisi içermeyen siyah veya beyaz alanların olması.
- Dilimleme işleminden sonra bazı dilimlerin çok az gerçek görüntü bilgisi içermesi. Görüntünün diğer alanlarının siyah veya beyaz renkten oluşması.
- Yol ağlarını içeren ikili raster görüntülerden oluşan bazı dilimlerde yol bilgisinin hiç olmaması. Yani görüntünün sadece yol bilgisi içermeyen diğer sınıftan oluşması.

- Bazı uzaktan algılama görüntülerinin bir bölümünün veya tamamının bozuk görüntü içermesi.
- Dilimleme işleminde 512x512 boyutlarında dilimler oluşturulmak istenirken son aşamalarda oluşturulan bazı dilimlerin genişliklerinin veya yüksekliklerinin 512 pikselden daha küçük olması.

Yukarıdaki nedenlerden dolayı veri setinde kullanılmayan (uygun olmayan) bazı örnek görüntüler Şekil 3.13'te verilmiştir. Görseller incelendiğinde (a) ve (d) görüntülerinin uygun yüksekliğe sahip olmadıkları, (b) ve (f) görüntülerinin büyük oranda beyaz renk içerdiği, (c) görüntüsünün büyük oranda siyah renk içerdiği ve (e) görüntüsünün çok az yol ağı verisi içerdiği görülmektedir.



Şekil 3.13. Çeşitli nedenlerden dolayı veri setine dahil edilmeyen bazı örnek görüntüler

Dolayısıyla Şekil 3.13'te verilen görüntüler ve bu görüntüler gibi çeşitli sorunlar içeren uygun olmayan görüntülerin filtrelenebilirliği gerekmektedir. Üstelik bu filtreleme işlemi gerçek görüntüler ve yol bilgisini içeren ikili görüntüler üzerinde eş zamanlı bir şekilde yapılmalı; kısaca uygun olmayan tüm görüntü çiftleri veri setini bozmamak adına birlikte elenmelidir. Uygun olmayan görüntüleri elemek için aşağıdaki gibi filtreleme yapan bir betik yazılmıştır. Bu betiğin amacı yukarıda uygun olmayan görüntülerin

oluşmasına neden olan şartların tamamını içeren bir koşul listesi oluşturmak ve bu koşula uyan görüntüleri elemektir.

```
def del_inapp_images():
    print('##### DELETE INAPPROPRIATE IMAGES #####')
    aerial_imgs_path = os.path.join(project_name, raster_data, aerial_image_slices)
    road_network_imgs_path = os.path.join(project_name, raster_data,
                                           road_network_slices)

    all_aerial_imgs = []
    all_road_network_imgs = []

    # Get list all aerial image slices
    for root, dirs, files in os.walk(aerial_imgs_path):
        for item in files:
            if '.jpg' in item:
                all_aerial_imgs.append(os.path.join(root, item))

    sort_all_aerial_imgs = natsort.natsorted(all_aerial_imgs)

    # Get list all road network slices
    for root, dirs, files in os.walk(road_network_imgs_path):
        for item in files:
            if '.jpg' in item:
                all_road_network_imgs.append(os.path.join(root, item))
    sort_all_road_network_imgs = natsort.natsorted(all_road_network_imgs)

    black_color = 0
    white_color = 255
    num_of_copy = 0
    num_of_not_copy = 0
    for aerial_item, network_item in zip(sort_all_aerial_imgs,
                                         sort_all_road_network_imgs):

        print('Loaded Aerial Image : ', aerial_item)
        print('Loaded Road Net Ima : ', network_item)
        ml_aer_img_name = aerial_item.split('/')[-1]
        ml_net_img_name = network_item.split('/')[-1]
        print('ml_aer_img_name : ', ml_aer_img_name)
        print('ml_net_img_name : ', ml_net_img_name)

        # Read image in grayscale mode and calculate white pixel ratio
        aerial_item_gray = cv2.imread(aerial_item, cv2.IMREAD_GRAYSCALE)
        white_pix_aer = np.sum(aerial_item_gray == white_color)
        non_white_pix_aer = np.sum(aerial_item_gray != white_color)

        black_pix_aer = np.sum(aerial_item_gray == black_color)
        non_black_pix_aer = np.sum(aerial_item_gray != black_color)

        # print('n_color_pix_aer : ', n_color_pix_aer)
        # print('n_other_pix_aer : ', n_other_pix_aer)
        aer_ratio_white_pix = (white_pix_aer / (white_pix_aer +
                                                non_white_pix_aer)) * 100
        aer_ratio_black_pix = (black_pix_aer / (black_pix_aer +
                                                non_black_pix_aer)) * 100
```

```

# print('aer_ratio_color_pix : ', aer_ratio_color_pix)

# Read image in grayscale mode and calculate black pixel ratio
network_item_gray = cv2.imread(network_item, cv2.IMREAD_GRAYSCALE)
n_color_pix_net = np.sum(network_item_gray == black_color)
n_other_pix_net = np.sum(network_item_gray != black_color)
net_ratio_color_pix = (n_color_pix_net / (n_color_pix_net +
                                         n_other_pix_net)) * 100

# Image height, width, channels

aer_img_h, aer_img_w = aerial_item_gray.shape[:2]
net_img_h, net_img_w = network_item_gray.shape[:2]

if aer_ratio_white_pix > 40 or \
    aer_ratio_black_pix > 40 or \
    net_ratio_color_pix > 99 or \
    aer_img_h != slice_size or net_img_h != slice_size or \
    aer_img_w != slice_size or net_img_w != slice_size:
    print('There is more than 30 percent free area on image.\n'
          ' and image pair is not used in the app.'.
          format(aerial_item, network_item))
    # os.remove(aerial_item)
    # os.remove(network_item)
    shutil.copyfile(aerial_item,
                    os.path.join(project_name, ml_data, ml_aerial_imgs,
                                inapprop_aer_imgs, ml_aer_img_name))
    shutil.copyfile(network_item,
                    os.path.join(project_name, ml_data, ml_road_net_imgs,
                                inapprop_net_imgs, ml_net_img_name))
    num_of_not_copy = num_of_not_copy + 1
else:
    print(' and image pair is suitable for app.'.format(aerial_item,
                                                         network_item))
    shutil.copyfile(aerial_item,
                    os.path.join(project_name, ml_data, ml_aerial_imgs,
                                approp_aer_imgs, ml_aer_img_name))
    shutil.copyfile(network_item,
                    os.path.join(project_name, ml_data, ml_road_net_imgs,
                                approp_net_imgs, ml_net_img_name))
    num_of_copy = num_of_copy + 1

print('Total number of aerial image : ', len(sort_all_aerial_imgs))
print('Total number of network image : ', len(sort_all_road_network_imgs))
print('Total number of copied image : ', num_of_copy)
print('Total number of not copied image : ', num_of_not_copy)
print('Number of copied +
      Number of not copied\n + = '.format(num_of_copy,
      num_of_not_copy, num_of_copy + num_of_not_copy))
print('-----\n PROCESS FINISHED.\n-----')

```

Yukarıdaki işlemlerden sonra artık uzaktan algılama görüntüleri ve bu görüntülere ait yol verilerini içeren raster formatındaki görüntü dosyaları hazır hale gelmiştir. Kısaca derin öğrenme modelinin ihtiyaç duyduğu veri seti hazırlanmıştır. Oluşan her bir gerçek

görüntü JPG formatında olup boyutları 512x512 pikseldir. Yol ağlarını içeren maske görüntülerde JPG yerine PNG görüntü formatı kullanılmıştır. JPG formatı kayıplı bir formattır dolayısıyla piksel değerlerinin değişme olasılığının önüne geçmek adına PNG formatı tercih edilmiştir (Jassim, 2013; Sayood, 2002, s. 372).

3.4.2.9. Veri setinin bölünmesi

Bir çalışma alanına ait veri setine dahil edilmek için uygun olan tüm gerçek ve maske görüntüler birer dizinde toplandıktan sonra artık bu görüntülerin eğitim ve test verisi olarak bölünmeleri gerekmektedir. Çalışma kapsamında tüm verilerin %80'i eğitim ve %20'si test verisi olarak kullanılmaktadır. Gerçek görüntüler ve maske görüntüler Python içinde yerleşik olarak gelen shuffle fonksiyonu kullanılarak eş zamanlı olarak karıştırılmaktadır. Shuffle fonksiyonu, bir listenin öğelerini rasgele seçer. Rastgele karıştırılan görüntüler yukarıda belirlenen oranlarda bölünerek derin öğrenme modeli tarafından kullanılan nihai dizinlere kopyalanmaktadırlar. Veri setinin dizin yapısı aşağıdaki gibidir (Şekil 3.14).



Şekil 3.14. Derin öğrenme modelinde kullanılan veri setinin dizin yapısı

val_imgs, dizininde eğitim süreci tamamlandıktan sonra görsel değerlendirme yapmak amacıyla hazırlanan görüntüler mevcuttur. *val_imgs_out*, dizininde ise bu görsel değerlendirmeden elde edilen modelin tahmin ettiği yol ağlarını içeren görüntüler vardır.

Yukarıda açıklanan işlemlerin yapılmasını sağlayan kod bloğu ise aşağıda verilmiştir.

```
def shuffle_list(*ls):
    zipped_list = list(zip(*ls))

    shuffle(zipped_list)
    return zip(*zipped_list)

def get_list_of_img(input_path, file_extension):
    list_of_imgs = glob.glob(os.path.join(input_path, file_extension))
    list_of_imgs = natsort.natsorted(list_of_imgs)
    return list_of_imgs
```

```

def create_dataset(input_list, dest_path):
    for one_img in input_list:
        shutil.copyfile(one_img, os.path.join(dest_path, one_img.split('/')[-1]))

def run_create_dataset():
    list_of_aer_imgs = get_list_of_img(input_path=os.path.join(project_name,
                                                                ml_data,
                                                                ml_aerial_imgs, approp_aer_imgs),
                                      file_extension=aer_img_ext)

    list_of_mask_imgs = get_list_of_img(
        input_path=os.path.join(project_name, ml_data, ml_road_net_imgs,
                                approp_net_imgs),
        file_extension=mask_img_ext)

    sh_list_of_aer_img, sh_list_of_mask_imgs = shuffle_list(list_of_aer_imgs,
                                                            list_of_mask_imgs)

    # split data using a certain percentage
    train_size = 0.8
    test_size = 0.2

    # Create train data
    train_aer_data = sh_list_of_aer_img[:int(len(list_of_aer_imgs) * train_size)]
    train_mask_data = sh_list_of_mask_imgs[:int(len(list_of_mask_imgs) * train_size)]

    # Create test data
    test_aer_data = sh_list_of_aer_img[int(len(list_of_aer_imgs) * train_size):]
    test_mask_data = sh_list_of_mask_imgs[int(len(list_of_mask_imgs) * train_size):]

    # Create train image dataset
    create_dataset(input_list=train_aer_data,
                  dest_path=os.path.join(project_name, ml_data, dataset,
                                          train_img_dataset))

    # Create mask image dataset
    create_dataset(input_list=train_mask_data,
                  dest_path=os.path.join(project_name, ml_data, dataset,
                                          train_mask_dataset))

    # Create test image dataset
    create_dataset(input_list=test_aer_data,
                  dest_path=os.path.join(project_name, ml_data, dataset,
                                          test_img_dataset))

    # Create test image dataset
    create_dataset(input_list=test_mask_data,
                  dest_path=os.path.join(project_name, ml_data, dataset,
                                          test_mask_dataset))

    print('-----\n PROCESS FINISHED.\n-----')

```

3.4.2.10. Çalışma alanlarının birleştirilmesi

Daha önce de ifade edildiği gibi çalışma kapsamında sınırlayıcı kutular ile sınırları belirlenen alanların büyük olması bazı problemlere neden olmaktadır. İlgili alana ait verilerin indirme sürelerinin yüksek olması ve ortaya çıkan verilerin işlenmesindeki yüksek donanım gereksinimi başlıca iki temel sorundur. Bu sorunların üstesinden gelmek adına büyük çalışma alanları yerine nispeten daha küçük çalışma tercih edilmiştir. Diğer yandan küçük çalışma alanlarından elde edilen veriler derin öğrenme modeli için yetersiz olabilmektedir. Bu sorunun çözmek için ise birden fazla çalışma alanından elde edilen verilerin birleştirilmesini sağlayan bir yapı oluşturulmuştur. Birleştirme işlemi sadece çalışma sonunda elde edilen nihai 512x512 boyutlarındaki gerçek ve maske görüntüler üzerinde uygulanmaktadır. Böylece hem veri setinin miktarı hem de çeşitliliği artmaktadır. Bu iki etken de derin öğrenme modelinin performansının katkı sağlamaktadır.

Birden fazla projenin birleştirilmesi işlemi aşağıdaki adımlardan oluşmaktadır.

- Birleştirilmek istenen projeler için dizin yapısı oluşturulmaktadır. Örneğin *tubingen* ve *minsk* çalışma alanlarının birleştirilmesi ile oluşacak olan tüm veriler *merged_tub_min* adında otomatik oluşturulan dizinin içerisinde saklanmaktadır.
- Birleştirilmek istenen projelere ait eğitim ve test verileri yeni oluşturulan ortak dizinde ilgili dizinlerin altına kopyalanmaktadır.

Yukarıda açıklaması verilen işlemlerin yapılmasını sağlayan kod bloğu aşağıda verilmiştir.

```
# Create image list
def get_list_of_imgs(input_dir_path, img_extension):
    img_list = glob.glob((os.path.join(input_dir_path, img_extension)), recursive=False)
    print('Founded images : \n', img_list)
    return img_list

def merge_projects_imgs(*args):
    print('##### MERGE PROJECTS #####')
    merged_proj_name = 'merged_'
    for one_item in mer_proj_name:
        merged_proj_name = merged_proj_name + one_item[0:3] + '_'
    print('merged_proj_name : ', merged_proj_name)
    merged_proj_name = merged_proj_name[:-1]
    mod_create_req_dirs.create_dirs(project_name=merged_proj_name)

    for one_project in mer_proj_name:
        print(os.path.join(one_project, ml_data, ml_aerial_imgs, approp_aer_imgs))
        aer_img_list = get_list_of_imgs(
```

```

input_dir_path=os.path.join(one_project, ml_data, ml_aerial_imgs,
                             approp_aer_imgs),
img_extension='*.jpg')
net_img_list = get_list_of_imgs(
    input_dir_path=os.path.join(one_project, ml_data, ml_road_net_imgs,
                                approp_net_imgs),
    img_extension='*.png')

for aer_img in aer_img_list:
    print(' copied.'.format(aer_img))
    shutil.copyfile(aer_img, os.path.join(merged_proj_name, ml_data,
                                          ml_aerial_imgs, approp_aer_imgs,
                                          aer_img.split('/')[1]))

for net_img in net_img_list:
    print(' copied.'.format(net_img))
    shutil.copyfile(net_img, os.path.join(merged_proj_name, ml_data,
                                          ml_road_net_imgs, approp_net_imgs,
                                          net_img.split('/')[1]))

# Create train and test data
print('##### CREATE NEEDED DIRECTORIES #####')
print('Creating train and test data.\nPlease wait...')
mod_splite_dataset.run_create_dataset(project_name=merged_proj_name)
print('Projects merged!')
print('-----\n PROCESS FINISHED.\n-----')

```

Buraya kadar veri setinin nasıl oluşturulduğu ile ilgili tüm aşamalar açıklanmıştır. Çalışma kapsamına kullanılan veri seti, bu aşamaların otomatik olarak gerçekleşmesini sağlayan ilgili modüllerin çalıştırılması ile elde edilmiştir. Veri seti ile ilgili diğer ayrıntılar aşağıda sunulmuştur.

3.4.3. Veri seti

Yukarıda açıklanan yöntem kullanılarak veri seti hazırlanmıştır. Veri seti birbirinden farklı üç lokasyondan elde edilmiştir. Bunlar daha önce ifade edildiği gibi Tübingen, Minsk ve Saint Petersburg bölgeleridir. Çalışma kapsamında bu üç bölgeden ayrıntıları Tablo 3.5'te verilen miktarda veri elde edilmiş ve bu üç veri birleştirilerek kullanılmıştır. Ayrıca Stara Zagora bölgesinden elde edilen veriler modellerin sonuçlarını değerlendirmek amacıyla kullanılmış bu nedenle veri setine dahil edilmemiştir.

Tablo 3.5. *Veri seti detayları*

Bölge (Şehir)	Ham Görün.	Toplam Görün.	Silinen Görün.	Uygun Görün.	Eğitim Görün.	Test Görün.
Tübingen	19	18042	11883	6159	4927	1232
Minsk	51	100417	63239	37178	29803	7520
Saint Peters.	100	180659	110411	70248	56198	14050
TOPLAM	170	299118	185533	113585	90928	22802

Tablo 3.5'te veri seti ile ilgili bilgiler ayrıntılı bir şekilde verilmiştir. Tablodaki notasyonlar ve açıklamaları aşağıdaki gibidir:

Ham Görün., OAM'ten indirilen yüksek çözünürlüklü uzaktan algılama görüntülerinin sayısıdır.

Toplam Görün., ham görüntüler 512x512'lik küçük görüntülere bölündüğünde ortaya çıkan toplam görüntü sayısıdır.

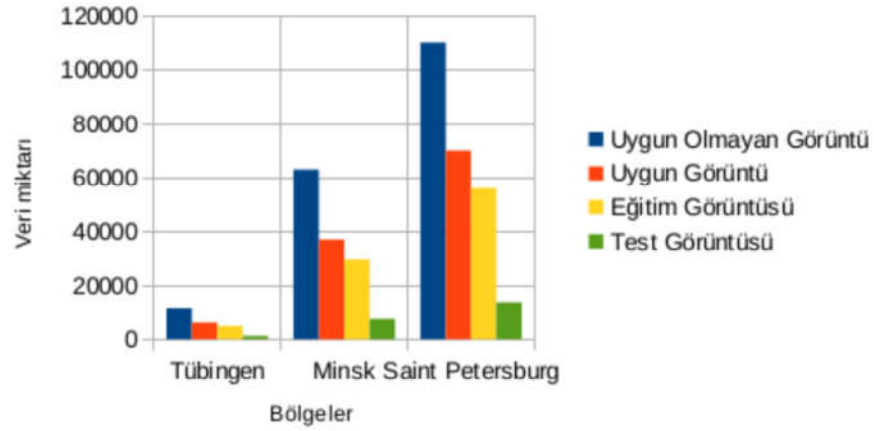
Silinen Görün., uygun olmadığı için veri setine dahil edilmeyen 512x512 boyutlarındaki görüntülerin sayısıdır.

Uygun Görün., eğitim setinde kullanılan toplam eğitim ve test görüntü sayısıdır.

Eğitim Görün., ilgili çalışma alanı için modelin eğitilmesinde kullanılan görüntülerin sayısıdır.

Test Görün., ilgili çalışma alanı için modelin performansının test edilmesinde kullanılan görüntülerin sayısıdır.

Buna göre çalışmada toplam 3 farklı bölgeden alınan 113585 adet 512x512 boyutlarında görüntü kullanılmıştır. Görüntüler rastgele olarak %80'i (90928) eğitim ve geriye kalan %20'si (22802) ise test verisi olacak şekilde bölünmüştür. Veri seti ile ilgili temel bilgiler Şekil 3.15'te verilmiştir.



Şekil 3.15. Veri seti ayrıntıları

3.5. Modellerin Belirlenmesi

Bu çalışma kapsamında yüksek çözünürlüklü uzaktan algılama görüntülerinden yol ağını tespit etmek için anlamsal görüntü bölütleme tekniği kullanılmıştır. Daha önce de ifade edildiği gibi anlamsal görüntü bölütlemenin amacı, görüntüdeki her pikseli doğru sınıfın etiketi ile işaretlemektir.

Çalışma kapsamında uygun model veya modellerin belirlenmesi amacıyla çeşitli çalışma alanlarından elde edilen sınırlı miktardaki veriler (<10.000) ile öncül uygulamalar yapılmıştır. Uygun modelleri belirlemeye yönelik yapılan bu çalışmalarda VGG, TEA, PSPNet, U-Net ve SegNet modelleri test edilmiştir. Yapılan bu uygulamalar ve literatür taraması neticesinde bu çalışma kapsamında özelleştirilmiş TEA, PSPNet ve SegNet modellerinin kullanılmasına karar verilmiştir. TEA, diğer birçok görüntü bölütleme modelinin temelini oluşturması itibarı ile bu çalışma kapsamında tercih edilen modellerdendir. PSPNet hem güncel bir model olması hem de nispeten daha büyük boyutlu görüntüler ile çalışma açısından daha uygun olması sebebiyle tercih edilen bir diğer modeldir. SegNet ise gerek basit yapısı gerekse de derin öğrenme modellerinden farklı olarak daha sınırlı sayıda katman içermesi açısından kullanılmaya uygun bulunmuştur. Diğer yandan SegNet'in ImageNet veri seti üzerinde gösterdiği başarı bir diğer tercih nedenidir.

3.6. Model hiper parametrelerinin belirlenmesi

Bir YSA modeli veya derin öğrenme modeli için, katman sayısı, giriş görüntüsün boyutu, evrişim filtresinin sayısı ve boyutu ile öğrenme hızı ve tur sayısı (epoch) gibi önceden tanımlanmış diğer parametreler dahil olmak üzere bir dizi parametre

ayarlanmalıdır (Romero vd., 2015). Bu parametreleri belirlemek için herkesçe kabul edilen genel bir kural veya formül yoktur. Bu parametrelerin birçoğu tecrübe ve yapılan çalışmanın niteliği ve kapsamına bağlıdır. Bu çalışma kapsamında kullanılan modellere ait hiper parametreler ile ilgili bilgiler Tablo 3.6'da verilmiştir.

Tablo 3.6. Çalışma kapsamında kullanılan her bir modele ait hiper parametreler ve değerleri

Hiperparametre	Arama Aralığı	TEA	PSPNet	SegNet
Öğrenme oranı	1e-2, 1e-3, 1e-4	1e-3	1e-3	1e-3
Aktivasyon fonksiyonu	ReLU	ReLU	ReLU	ReLU
Batch size	4-64	8	8	8
Tur sayısı	100-800	700	512	512
Tur başına adım sayısı	256-512	512	512	512
Giriş görüntüsü boyutu	128x128, 512x512, 576x576	256x256	576x576	512x512

3.7. Öğrenme Aktarımı (Transfer Learning)

Görüntü sınıflandırması için eğitilmiş ESA modelleri, bölütlendirme için de kullanılabilecek anlamlı bilgiler içermektedirler. Önceden eğitilmiş modellerin evrişim katmanları, bölütleme modelinin kodlayıcı katmanlarında yeniden kullanılabilir. Bu çalışma kapsamında öğrenme transferi ile çeşitli çalışmalar yapılmış fakat ilgili işlemin modelin hızına veya performansına anlamlı sayılabilecek seviyede bir katkı sağlayamaması nedeniyle kullanılmamıştır.

3.8. Eğitim ve Uygulama Ayrıntıları

Bu çalışmada modeli oluşturmak ve eğitmek için Keras (Chollet ve vd., 2015) derin öğrenme kütüphanesi kullanılmıştır. Yukarıda ayrıntıları verilen veri seti kullanılarak özelleştirilmiş TEA, PSPNet ve SegNet ağları kullanılarak modeller eğitilmiştir. Modeller için kullanılan hiper parametreler Tablo 3.6'da verilmiştir.

4. ARAŞTIRMA BULGULARI

4.1. Eğitim ve Test Verisi Bulguları

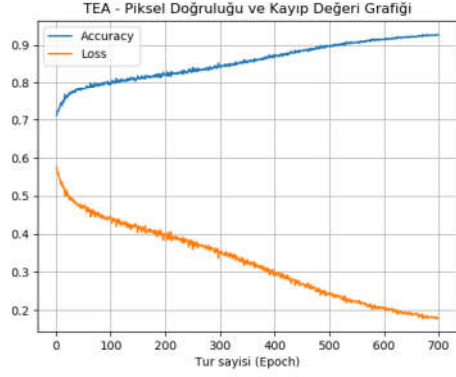
Bu çalışma kapsamında test verilerini kullanarak modelin performansını ölçmek için kayıp değeri, doğruluk, frekans ağırlıklı IoU, ortalama IoU, ve sınıf tabanlı IoU metrikleri kullanılmıştır. Bu metriklerin yanında her bir model için bir turda harcanan ortalama süre ve toplam tur sayıları gibi bilgiler Tablo 4.1’de verilmiştir. Tablodaki değerler için virgülden sonraki ilk dört basamak alınmıştır.

Tablo 4.1. Temel değerlendirme parametreleri ve modellerin aldıkları değerler

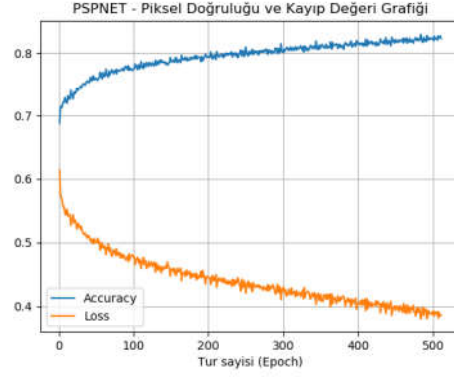
Metrik	TEA	SegNet	PSPNet
Tur/Süre	262 sn.	385 sn.	347 sn.
Tur sayısı (epoch)	700	512	512
Piksel Doğruluğu	0.9270	0.7786	0.8232
Kayıp Değeri	0.1786	0.4743	0.3865
Frek. ağırlıklı IoU	0.7285	0.6268	0.6385
Ortalama IoU	0.6744	0.5398	0.5680
Sınıf tabanlı IoU	[0.7982, 0.5507]	[0.7390, 0.3406]	[0.7295, 0.4064]

SegNet ve PSPNet ağırları 512 tur kullanılarak eğitilmişlerdir. Yaklaşık olarak 500 turdan sonra bu modellerin doğruluk ve kayıp değerlerinde anlamlı bir fark oluşmadığı için bu modeller için 512 tur yeterli bulunmuştur. Özelleştirilmiş TEA’da ise 660. turdan sonra anlamlı bir fark oluşmadığı için bu modelin tur sayısı 700 olarak belirlenmiştir.

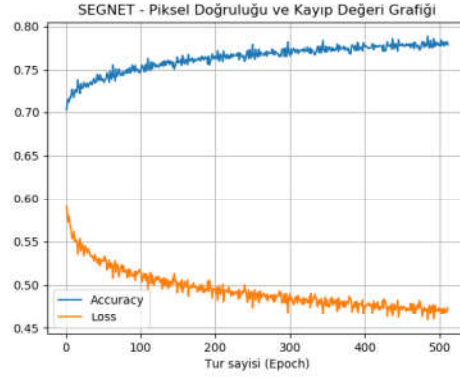
TEA, PSPNet ve SegNet modellerinin eğitim sürecinde tur sayısına göre kayıp değerleri ve piksel doğrulukları grafiksel olarak sırasıyla Şekil 4.1’de verilmiştir. Grafikler incelendiğinde ilk 50 turdaki doğruluk ve kayıp değerinin değişiminin hızlı olduğu görülmektedir. PSPNet ve SegNet modellerinin yaklaşık 400. turdan sonra değişim hızlarının iyice azaldığı görülmektedir. Özelleştirilmiş TEA modelde ise yaklaşık 620. turdan sonra doğruluk ve kayıp değerlerindeki değişimin azaldığı görülmektedir.



(a)



(b)



(c)

Şekil 4.1. Modelinin tur sayısına göre piksel doğruluğu ve kayıp değerleri grafikleri

Bu değerlere göre ortalama test verisi ile modeller değerlendirildiğinde piksel doğruluğu en yüksek olan model %92.70 ile özelleştirilmiş TEA'dır. PSPNet modeli için ortalama doğruluk %82.32 ve SegNet modeli için ise %77.86'dir. SegNet ve PSPNet modellerin genel olarak yüksek kayıp değerlerine sahip olduğu görülmektedir.

4.2. Model Değerlendirme

Modelleri değerlendirmek amacıyla piksel doğruluğu, kesinlik, hassasiyet ve F_1 skorlarının yanında; ortalama IoU ve frekans ağırlıklı IoU gibi anlamsal bölütleme uygulamalarına özgü değerlendirme metrikleri kullanılmıştır. Çalışmada Stara Zagora bölgesinden elde edilen 1200x1200 boyutundaki 1200 adet görüntü kullanılmıştır. Değerlendirmede kullanılan görüntüler görsel olarak incelenerek modelin değerlendirme sonuçlarında ciddi sapmalara neden olabilecek çok fazla gürültü içeren bazı görüntü çiftleri değerlendirme veri setinden çıkarılmıştır. Değerlendirme sonuçları Tablo 4.2'de verilmiştir.

Tablo 4.2. Modellerin Stara Zagora bölgesinden elde edilen değerlendirme verileri için ürettikleri sonuçlar

Metrik	TEA	Segnet	PSPNet
Piksel doğ.	0.7102	0.5486	0.6082
Hassasiyet	0.7826	0.5995	0.6557
Kesinlik	0.7949	0.6738	0.6742
F ₁ skoru	0.7887	0.6345	0.6648
Ortalama IoU	0.6592	0.4681	0.5143
Frek. ağır. IoU	0.7068	0.5599	0.5815

Elde edilen sonuçlara göre en başarılı sonuçlar özelleştirilmiş TEA ile elde edilmiştir. SegNet ve PSPNet modellerinin başarı metrik sonuçları genel olarak birbirine yakındır. Diğer yandan değerlendirme görüntülerinin de otomatik olarak oluşturulması bu görüntülerde gerçek görüntü ile maske görüntüsünün tam örtüşmemesi, görüntülerin gürültü içermesi gibi sorunların oluşmasına neden olmuştur. Bu durum değerlendirme sonuçlarının nispeten düşük çıkmasındaki en temel etkindir. Bu durumu doğrulamak amacıyla manuel olarak hazırlanan ve daha az hata ve gürültü içeren az sayıdaki görüntüler ile modeller test edildiğinde elde edilen değerlendirme metrik sonuçları daha yüksek olmaktadır. Fakat bu görüntüler sayıca az olduğu için ve bu çalışmanın amacını temsil eden otomatik veri oluşturma yaklaşımına uymadığı için değerlendirmeye alınmamıştır.

4.2.1. Model Sonuçlarını Görsel Olarak Değerlendirme

YSA modellerinden elde edilen tahmin veya sınıflandırma sonuçlarının standart değerlendirme yöntemleri yanında görsel olarak değerlendirilmeleri de mümkündür. Üstelik bu değerlendirme yöntemi insan algısı açısından çok daha anlamlıdır. Daha önce de ifade edildiği gibi Stara Zagora bölgesinden elde edilen görüntüler değerlendirme verisi olarak kullanılmıştır. Aşağıda insan algılaması açısından farklı zorluk düzeylerine sahip üç görüntü grubu oluşturulmuştur. Her bir görüntü grubunda üç adet görüntü vardır. Her gruptaki üç görüntü için elde edilen ortalama değerlendirme metrik sonuçları Tablo 4.3’de verilmiştir. Bu tablodaki *Sade gör.* ifadesi Şekil 4.2’te verilen tespit edilmesi en kolay üçlü *sade görüntüler* grubunu; *Nor. gör.* ifadesi Şekil 4.3’te verilen orta düzey karmaşıklıkta üçlü *normal görüntüler* grubunu ve *Kar. Gör.* ifadesi en karmaşık olan ve dolayısıyla algılanması en zor olan üçlü *karmaşık görüntüler* grubunu ifade

etmektedir. Tablo 4.3’te görsel olarak değerlendirilen görüntülerin temel değerlendirme metrik sonuçları da verilmiştir.

Tablo 4.3. Modellerin farklı metriklere göre değerlendirme sonuçları

Model Metrik	TEA			SegNet			PSPNet		
	Sade Gör.	Nor. Gör.	Kar. Gör.	Sade Gör.	Nor. Gör.	Kar. Gör.	Sade Gör.	Nor. Gör.	Kar. Gör.
Piksel doğ.	0.7076	0.4746	0.5458	0.6165	0.3189	0.4775	0.6654	0.4872	0.4866
Hassasiyet	0.8291	0.6510	0.4948	0.7864	0.5130	0.4896	0.7638	0.6591	0.4724
Kesinlik	0.7539	0.7159	0.4933	0.7089	0.6694	0.4509	0.7063	0.7149	0.4073
F ₁ skoru	0.7897	0.6819	0.4940	0.7456	0.5808	0.4695	0.7339	0.6859	0.4375
Ortalama IoU	0.6322	0.3735	0.3994	0.5119	0.1941	0.4008	0.5704	0.3871	0.3848
Frek. ağır. IoU	0.6933	0.3541	0.6013	0.5562	0.1457	0.6452	0.6409	0.3700	0.6256

Sonuçlar incelendiğinde her üç grup görüntü için de en başarılı sonuçlar TEA ile elde edilmiştir. En düşük sonuçlar ise SegNet modeli ile elde edilmiştir. Diğer yandan algılanması en zor olan görüntüler olarak belirlenen karmaşık görüntülerden elde edilen bazı metrik sonuçları *normal görüntüler* olarak belirlenen orta düzeydeki görüntülere göre daha yüksek çıkmıştır. Bunun temel nedenin *normal görüntüler* olarak belirlenen görüntü grubundaki toprak zemine sahip yolların tespit edilmesindeki düşük başarı oranı olarak değerlendirilmektedir. Modellerin görsel değerlendirme sonuçları ise Şekil 4.2, Şekil 4.3 ve Şekil 4.4’te verilmiştir.

4.2.1.1. Uygulama 1 – Sade görüntüler

Birinci uygulamada insan algılaması açısından kolay olan üç adet görüntü hazırlanmış ve eğitilen modellerin bu görüntüleri sınıflandırmaları sağlanmıştır. Şekil 4.2’de sırasıyla gerçek görüntüler, bu görüntülere ait gerçek maske görüntüler ve daha sonra bu görüntülerin farklı modellerdeki tahmin (sınıflandırma) sonuçları verilmiştir. Bu bölümdeki gerçek görüntüler, yolların bulundukları çevreden kolayca ayırt edilebilmelerini sağlayacak yapıdadır. Sonuçlar Şekil 4.2’de görülmektedir.

Değerlendirme ve Analiz: Bu bölümdeki görüntülerde yer alan yollar, çevrelerinden rahat bir şekilde ayırt edilebilecek yapıdadır. Üstelik yollar şeritler ve yol çizgileri gibi belirgin yol özelliklerini sahiptirler. Ayrıca yolların genellikle kıvrımlara sahip olmayan düz yol oldukları görülmektedir. Bu görüntüler kullanılarak yapılan sınıflandırma sonuçlarının

insan algısı açısından kabul edilebilir seviyede oldukları söylenebilir. Ayrıca SegNet ve PSPNet yol şeritlerini TEA'ya göre daha başarılı şekilde tespit etmişlerdir.



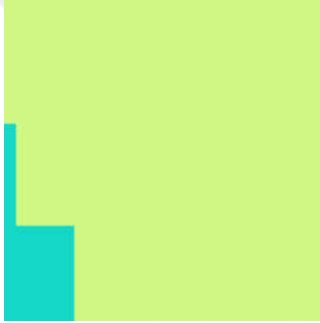
(a) Gerçek görüntü - 1



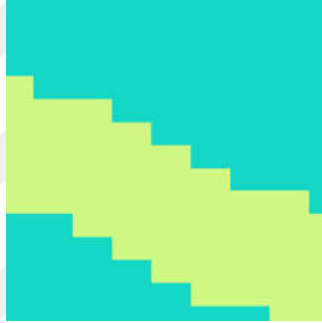
(b) Gerçek görüntü - 2



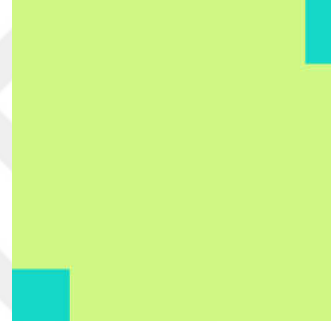
(c) Gerçek görüntü - 3



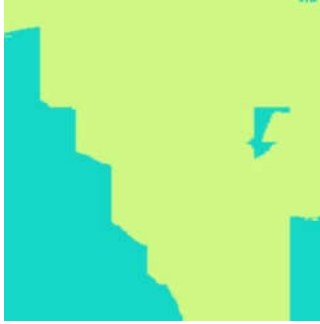
(d) Gerçek maske görüntüsü -1



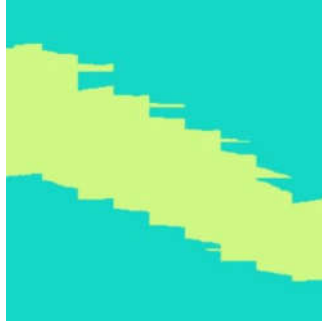
(e) Gerçek maske görüntüsü -2



(f) Gerçek maske görüntüsü -3



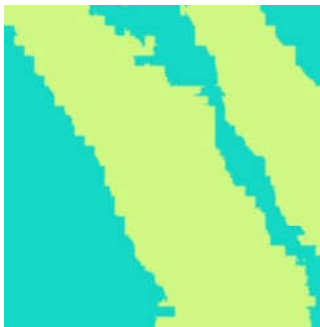
(g) TEA modeli tahmini-1



(h) TEA modeli tahmini-2



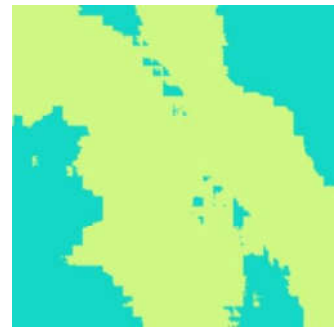
(i) TEA modeli tahmini-3



(j) SegNet modeli tahmini-1



(k) SegNet modeli tahmini-2



(l) SegNet modeli tahmini-3



(m) PSPNet modeli tahmini-1

(n) PSPNet modeli tahmini-2

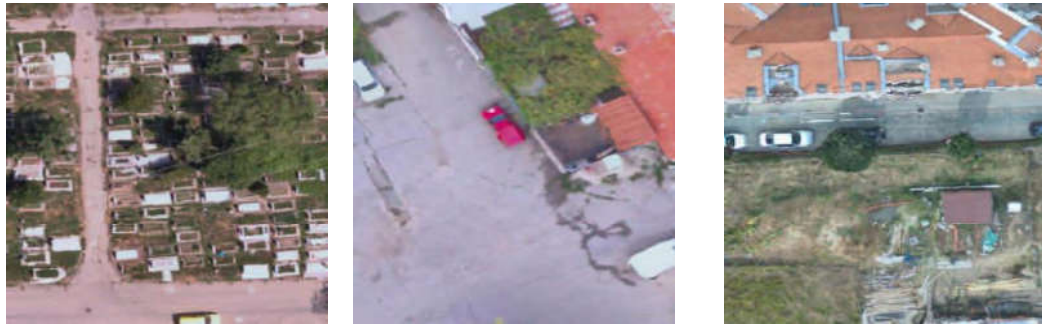
(o) PSPNet modeli tahmini-3

Şekil 4.2. İnsan algılaması açısından kolay olarak kabul edilebilecek örnek gerçek görüntüler ve bu gerçek görüntülerin sırasıyla TEA, SegNet ve PSPNet modelleri ile yol ağı tespit sonuçları

4.2.1.2. Uygulama 2 - Normal görüntüler

Bu uygulamada, insan algılaması açısından algılanması Uygulama 1'e göre nispeten daha zor olan orta zorluktaki birtakım görüntüler hazırlanmış ve eğitilen modellerin bu görüntüleri sınıflandırmaları sağlanmıştır. Gerçek görüntüler ve modellerin tahminleri Şekil 4.3'te görülmektedir.

Bu bölümdeki gerçek görüntüler temel yol özelliklerinden bir veya birden fazlasına sahip, genel olarak çevresindeki nesnelere yakın spektral özelliklere sahiptirler. Diğer yandan bu yolların çevresindeki nesneler iki veya daha fazla sınıfa ayrılabilir yapıdadır.

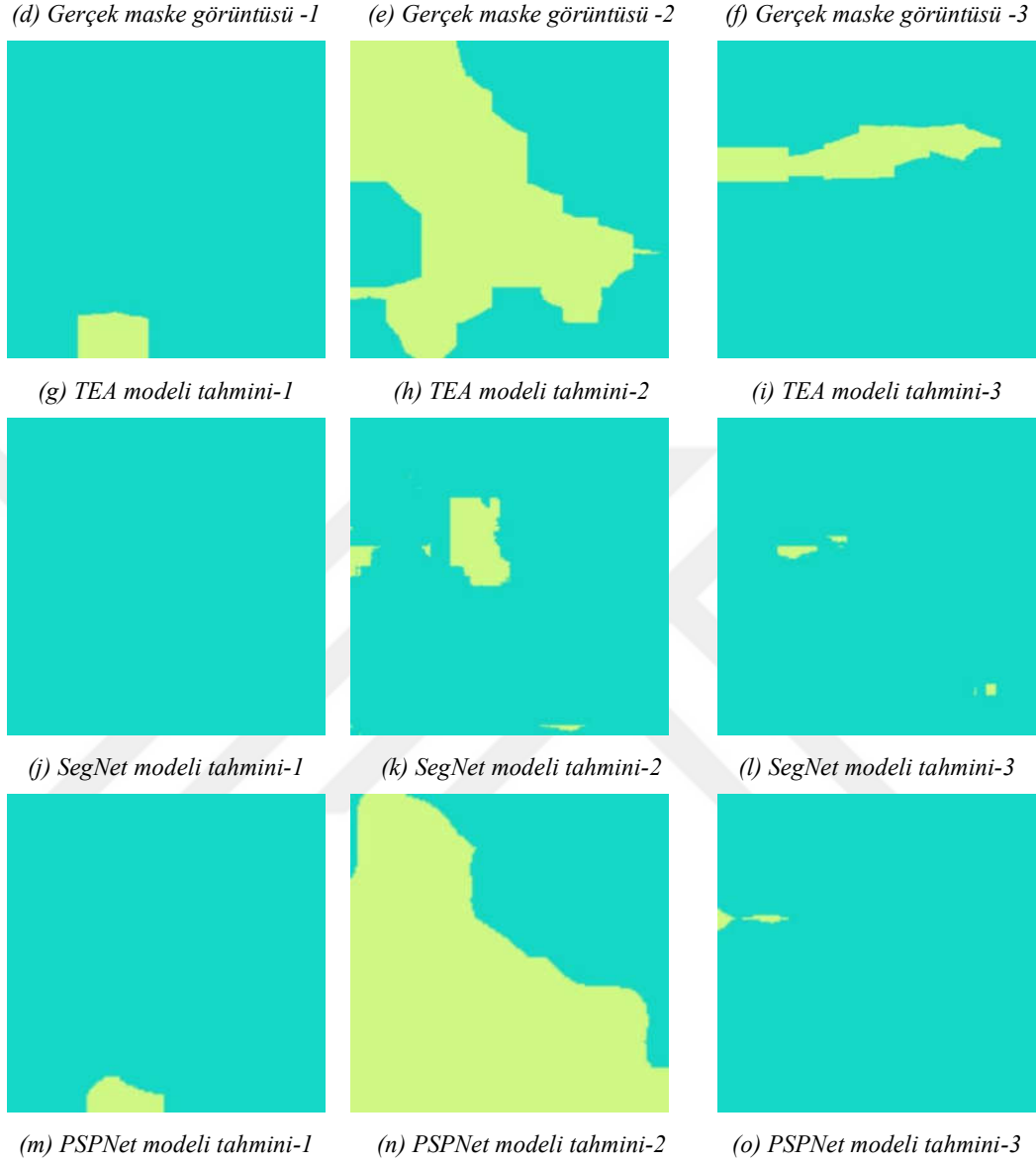


(a) Gerçek görüntü - 1

(b) Gerçek görüntü - 2

(c) Gerçek görüntü - 3





Şekil 4.3. İnsan algılaması açısından normal zorluk düzeyindeki örnek gerçek görüntüler ve bu gerçek görüntülerin sırasıyla TEA, SegNet ve PSPNet ve modelleri ile yol ağı tespit sonuçları

Değerlendirme ve Analiz: Bu uygulamada kullanılan görüntüler yollar da dahil olmak üzere üç veya daha fazla sınıfa ait nesneler içeren görüntülerdir. Yollar ise Uygulama-1'e göre daha az belirgin özelliklere sahiptir. Uygulama sonuçları incelendiğinde TEA ile elde edilen sonuçların nispeten kabul edilebilir seviyede olduğu; SegNet ve PSPNet ile elde edilen sonuçların ise genel olarak yetersiz olduğu görülmektedir.

4.2.1.3. Uygulama 3 - Karmaşık görüntüler

Bu uygulamada insan algılaması açısından tespiti zor olan görüntüler seçilmiştir. Bu bölümdeki görüntülerde yollar ilk bakışta fark edilecek kadar net değildir. Yol

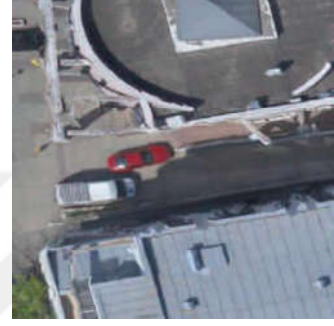
özelliklerini oluşturan şerit sayısı, yol çizgileri gibi özellikler belirgin değildir veya hiç yoktur. Ayrıca yollar ve yol çevresindeki nesneler spektral açıdan yollara yakın değerlere sahiptirler. Bu bölümdeki yolların bir kısmının ağaç, bina, araç gibi çeşitli nesneler ile üzerlerinin kısmen kapatıldığı görülmektedir. Bu görüntüler ve modellerin bu görüntüler için ürettikleri tahminler Şekil 4.4'te görülmektedir.



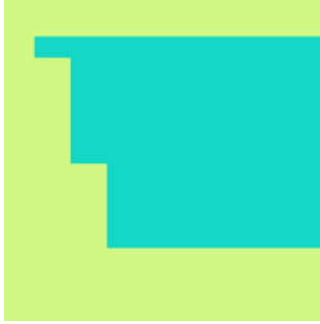
(a) Gerçek görüntü - 1



(b) Gerçek görüntü - 2



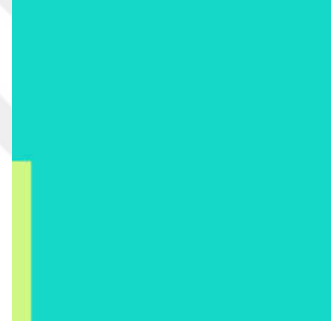
(c) Gerçek görüntü - 3



(d) Gerçek maske görüntüsü -1



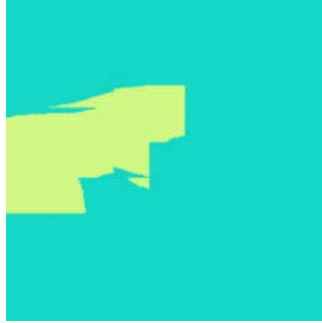
(e) Gerçek maske görüntüsü -2



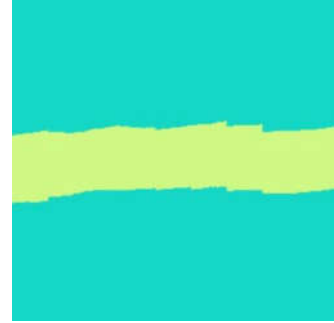
(f) Gerçek maske görüntüsü -3



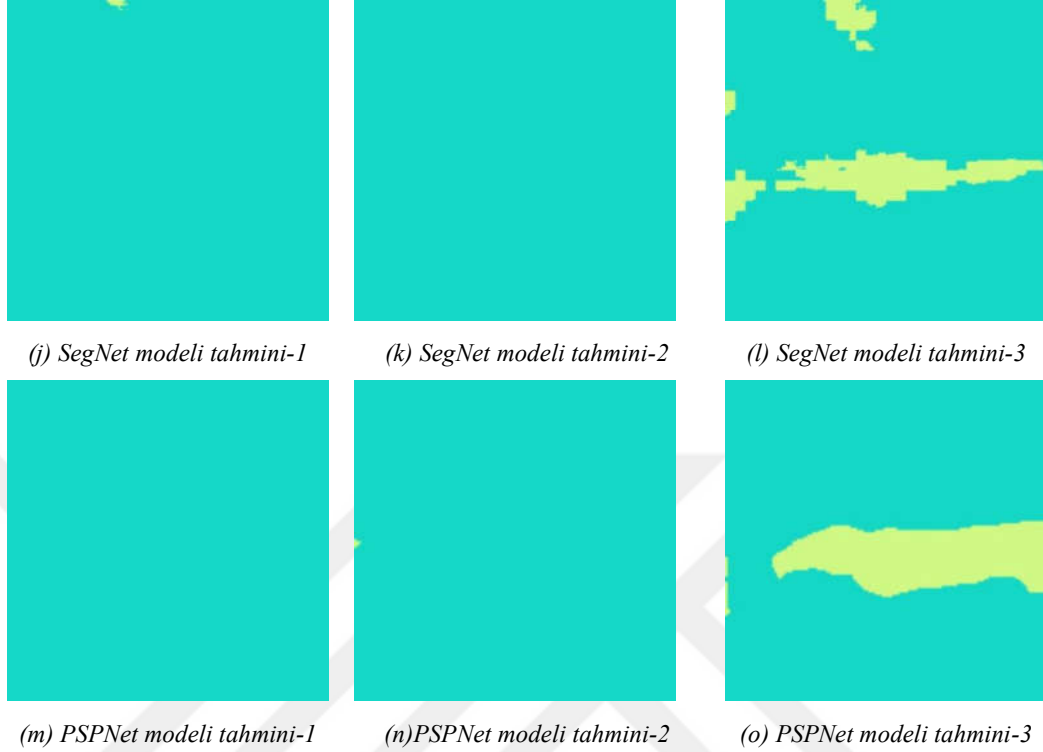
(g) TEA modeli tahmini-1



(h) TEA modeli tahmini-2



(i) TEA modeli tahmini-3



Şekil 4.4. İnsan algılaması açısından zor olarak nitelenebilecek örnek gerçek görüntüler ve bu gerçek görüntülerin sırasıyla TEA, SegNet ve PSPNet modelleri ile yol ağı tespit sonuçları

Değerlendirme ve Analiz: Bu uygulamada kullanılan görüntülerde, yollar da dahil olmak üzere üç veya daha fazla sınıfa ait nesneler bulunmaktadır. Yollar ise Uygulama-1 ve Uygulama-2'deki yollara göre göre daha az belirgin özelliklere sahiptir. Uygulama sonuçları incelendiğinde SegNet ve PSPNet modellerine ait tahminlerin oldukça yetersiz olduğu görülmektedir. Örneğin her iki modelde de Şekil 4.4 (a) ve (b) gerçek görüntüler için neredeyse hiç doğru tahmin yapılamadığı görülmektedir. TEA modeli ise daha önceki görüntülerde olduğu gibi bu görüntülerde de daha başarılı tahminler yapmıştır.

Deneysel sonuçlar göz önüne alındığında; aynı veri seti üzerinde birbirine yakın parametreler kullanılarak eğitilen farklı modellerden farklı sonuçlar elde edildiği görülmektedir. Çalışma sonuçları bir bütün olarak değerlendirildiğinde en başarılı sonuçlar özelleştirilmiş TEA modeli kullanılarak elde edilmiştir. Bu başarı hem matematiksel metrikler ile hem de görsel değerlendirme sonuçları ile teyit edilmiştir

Kentsel alanlardaki sade yollar veya şerit, yol çizgileri, asfalt zemin gibi yol özelliklerinin büyük bölümüne sahip yolların tespitinde modellerin genel olarak kabul edilebilir sonuçlar ürettikleri görülmektedir. Örneğin bu yol tür yollarda her üç modelden

elde edilen sonuçlar matematiksel değerlendirme metrikleri ve görsel açıdan tatmin edicidir.

Modellerin özellikle toprak yolların tespitinde sorun yaşadıkları görülmektedir. TEA modelinin bu noktada diğer modellere göre daha başarılı sonuçlar vermiş olmasına rağmen bu tarz yollardaki tahminler yeterli değildir. Bu durumun iki temel olası nedeni vardır. Birincisi hem OAM verileri hem de OSM verileri genel olarak kentsel alanlar için daha yoğun ve kalitelidir. Dolayısıyla veri seti içerisinde bu tarz yollara ait verilerin yeterli miktarda olmaması modellerin eğitim sürecinde bu yolları tam olarak öğrenememesine neden olabilir. Diğer bir neden ise kırsal alanlardaki toprak yolların spektral özellikler, şeritler, yol çizgileri gibi özellikler konusunda zayıf olmaları ve bu durumun da modellerin bu tarz yolları tam olarak öğrenmesini engellemesidir.

5. TARTIŞMA

Çalışmanın amacına uygun olarak OSM ve OAM verileri kullanılarak mevcut SegNet ve PSPNet derin öğrenme teknikleri ve özelleştirilmiş TEA ile yol ağları tespiti yapılmış ve elde edilen sonuçlar karşılaştırılmıştır. Çalışmanın bütünü ve elde edilen sonuçlar göz önüne alınarak çalışmanın yapılan benzer çalışmalara göre eksiklikleri, üstünlükleri, zayıf ve güçlü yanları gibi bilgiler aşağıda sunulmuştur.

Çalışmada elde edilen sonuçlar incelendiğinde anlamsal bölütleme çalışmaları için en önemli başarı metriklerinden birisi olan piksel doğruluğu, test edilen modeller için %55-75 aralığında değişmektedir. Daha önce de ifade edildiği gibi değerlendirme görüntüleri çeşitli nedenlerden dolayı tam örtüşme sorunları ve gürültü verisi içermektedirler. Manuel olarak hassas bir şekilde hazırlanan değerlendirme veri setinde sonuçlar çok daha yüksektir. Literatür incelendiğinde benzer çalışmalar için elde edilen piksel doğruluğu oranları ortalama %65-95 arasında değişmektedir (Daniel Hoffman, 2019; Wulamu vd., 2019; Xin vd., 2019; Z. Zhang vd., 2018). Bu yönü ile çalışma kapsamında elde edilen sonuçlar literatürdeki benzer çalışmaların alt ve orta seviyesine daha yakın olmakla birlikte literatürdeki sonuçlar ile paraleldir.

Çalışmada veri setini otomatik hazırlayan bir uygulama geliştirilmiştir. Uygulama ile yeterli miktarda OAM ve OSM verisine sahip herhangi bir bölge için hızlı bir şekilde veri seti oluşturulabilmektedir. Literatürde yol ağlarının makine öğrenmesi veya YSA kullanılarak tespit edilmesini sağlayan çoğu çalışmada veri seti manuel olarak oluşturulmakta ve dolayısıyla kısıtlı sayıda veri seti kullanılmaktadır. Ayrıca bu çalışmaların birçoğunda genellikle sadece belirli bir bölgeye ait veriler kullanıldığı için veri çeşitliliği de kısıtlıdır (Daniel Hoffman, 2019; Q. Li vd., 2014; Mokhtarzade ve Zoej, 2007; Wulamu vd., 2019; Xin vd., 2019; Z. Zhang vd., 2018). Bu çalışmada kullanılan veri seti büyüklük ve veri çeşitliliği bakımından literatürdeki birçok çalışmadan daha geniş kapsamlıdır. Üstelik geliştirilen uygulama veri seti kapasitesinin artırılması imkânı da sunmaktadır.

Literatürde makine öğrenmesi veya YSA ile yol ağı tespiti uygulamalarında OSM verilerinin kullanıldığı çeşitli çalışmalar vardır (Fukuda vd., 2019; Laddha vd., 2016; Q. Li vd., 2014; Mattyus vd., 2017; Wu vd., 2019). OAM veya OAM ve OSM verilerinin birlikte kullanılması ile yapılan yol tespit çalışmalarına ise rastlanmamıştır. Bu yönü ile çalışma özgündür. Çalışmada temel olarak OAM, OSM ve YSA gibi üç farklı yapının

birlikte kullanılması çalışmanın kalitesini ve özgünlüğünü artırmakla birlikte beraberinde birçok zorluk ve çözülmesi gereken problem getirmiştir. Veri setinin hazırlanması için OSM ve OAM platformlarından elde edilen veriler üzerinde çok sayıda işlem yapılmıştır. Bu işlemlerde temel veri özelliklerinin korunması konusunda azami özen ve gayret gösterilmiştir. Fakat diğer yandan bu platformlardan elde edilen veriler üzerinde çok sayıda işlem yapılması bu verilerin kalitesi ve doğruluğu konusunda birtakım soruları beraberinde getirmektedir. Her ne kadar çalışma esnasında rasgele seçilen örnekler üzerinde bu kontroller gerçekleştirilmiş ve olumsuz bir durum ile karşılaşılmamış olsa da veri setinin büyüklüğü göz önüne alındığında veri setinin kalite kontrolünü sürekli veya daha sık yapacak bir mekanizma yararlı olabilir.

OAM verisinin oluşturulmasında farklı teknik özelliklere ve teknolojilere sahip birbirinden farklı platformlar kullanılmaktadır. Bilindiği gibi uzaktan algılama ile elde edilen görüntülerde çeşitli radyometrik ve geometrik bozulmalar meydana gelmektedir (Homma vd., 2000). Uydu ile elde edilen görüntülerde çoğunlukla bu bozulmalar görüntüyü elde eden platform tarafından otomatik düzeltilmektedir. Oysa IHA, drone gibi araçlardan elde edilen görüntülerde söz konusu hataların otomatik düzeltilmesi ile ilgili yerleşik bir yapı genellikle yoktur. Birçok açık kaynak projede tartışılan verilerin doğruluğunun kontrolü bu çalışmada da ortaya çıkmaktadır (Cipeluch vd., 2010; Helbich vd., 2012). Çalışma kapsamında OAM platformundan indirilen bazı görüntülerin tamamen bozuk olduğu görülmüştür. Buna rağmen bu görüntüler çalışmaya dahil edilerek oluşturulan filtre mekanizmasının bu verileri veri setine dahil etmemesi amaçlanmıştır. Sonuçlar, bu filtrelemenin genel olarak başarılı olduğunu göstermektedir. Sonuç itibari ile OAM ve OSM platformlarından elde edilen görüntülerde bu durumun varlığı göz önünde bulundurulmalıdır.

Veri seti bazı temel anlamsal bölütleme teknikleri ile test edilmiş olmasına rağmen henüz test edilmediği DeepLabV3, U-Net, MobileNet, gibi farklı modeller de vardır. Veri setinin bu modeller ile test edilmesi faydalıdır. Diğer yandan mevcut modellerin ürettikleri sonuçlar gerek veri seti miktarı gerekse de veri seti çeşitliliği kriterleri göz önüne alındığında daha fazla performans gösterme potansiyeli taşımaktadır. Özellikle derin öğrenme modelleri üzerinde yapılacak daha ince optimizasyonlar ile sonuçların iyileştirilmesi mümkün görülmektedir.

Modellerin eğitimlerinden sonra elde edilen sonuçlar değerlendirildiğinde insan algısı açısından kolay görüntülerdeki yol ağlarının genel olarak başarılı bir şekilde tespit edildiği görülürken; zor ve karmaşık görüntülerdeki yol ağları tespitinde bazı sorunların olduğu görülmektedir. Dolayısıyla bu noktada yapılacak optimizasyonlar modelin performansını artırıp mevcut tahmin sonuçlarını iyileştirebilecek imkânı taşımaktadır.

Mevcut durumda çalışma için grafiksel bir arayüz yoktur. Uygulamayı çalıştırmak için Python yüklü bir ortamda kütüphanelerin kurulması gerekmektedir. Her ne kadar kütüphanelerin kurulumu basit birkaç komut ile yapılabilir olsa da grafiksel arayüz eklenip ilgili işlemlerin arka planda yapılması ortaya çıkan uygulamanın daha profesyonel olmasını sağlayabilir. Ayrıca grafiksel arayüzün eklenmesi Python veya kodlama konusunda hiç bilgisi olmayan kullanıcıların bu aracı rahat bir şekilde kullanmasına imkân sağlayabilir. Bu nedenle mevcut durumda grafiksel arayüzün olmaması bir dezavantajdır. Diğer yandan açık kaynak dünyasında yazılımlar veya paketler sık sık güncellenmektedir. Üstelik bu yazılımlar veya paketler çoğu zaman birbirinden bağımsız kullanıcılar tarafından geliştirildiği için paketin yeni sürümünde yapılan bir değişiklik ilgili paketin kullanıldığı başka bir ortamda soruna neden olabilmektedir. Dolayısıyla grafiksel arayüz sağlayıp kullanıcıları açık kaynak dünyasındaki bu gerçeklikten uzaklaştırmak ortaya çıkması muhtemel olumsuz yanlardan birisidir.

6. SONUÇ VE ÖNERİLER

Bu çalışmada OAM'dan elde edilen yüksek çözünürlüklü uzaktan algılama görüntüleri ve OSM'den elde edilen vektörel veriler kullanılarak bir veri seti hazırlanmıştır. Hazırlanan veri seti, anlamsal bölütleme tekniği kullanılarak yol ağlarının tespitini sağlayan güncel ve yaygın bir takım derin öğrenme modellerinde kullanılmıştır. Çalışma esnasında anlamsal bölütleme tekniklerinden SegNet, PSPNet ve özelleştirilmiş TEA modelleri kullanılmıştır. Çalışma bir bütün olarak ele alındığında sonuçları aşağıdaki gibi maddeler halinde sunulmuştur.

- Literatürde yol ağlarının tespiti ile ilgili çok sayıda yöntem ve teknik mevcuttur. Özellikle makine öğrenmesi ve derin öğrenme alanlarındaki yeni teknikler ve modeller yol ağları tespit çalışmalarında da kullanılmaktadır. Diğer yandan otonom araçlar gibi teknolojiler bu alandaki çalışmaların daha da anlam kazanmasını sağlamıştır. Yapılan bu çalışmalara rağmen yol tespiti konusunda henüz herkesçe kabul edilen ortak ve yüzde yüz başarılı bir yöntem veya algoritma yoktur.
- Bu çalışma ile açık kaynak platformlardan elde edilen veriler kullanılarak bir veri seti oluşturulmuştur. Oluşturulan bu veri seti değişik anlamsal bölütleme tekniklerine sahip modellerin eğitilmesi için kullanılmış ve elde edilen sonuçlar bu yöntem ile yol ağlarını tespit etmenin mümkün olduğunu göstermiştir. Özellikle insan algısı açısından kolay olarak nitelendirilen görüntülerdeki yollar eğitilen modeller ile yüksek doğrulukla tespit edilebilmiştir. Daha fazla sınıfa ait nesnenin yer aldığı ve yol özelliklerinin daha az belirgin olduğu nispeten karmaşık görsellerde ise eğitilen modellerin kısmen kabul edilebilir sonuçlar sağladığı görülmüştür. Bu çalışma ile elde edilen sonuçlar göz önünde alındığında açık kaynak sistemlerden elde edilen ücretsiz ve kaliteli verilerin birçok makine öğrenmesi veya derin öğrenme çalışmasında kullanılabileceği görülmektedir.
- Yapılan çalışma farklı açık kaynak sistemlerde depolanan ve veri yapıları, depolama biçimleri, veriye erişim yöntemleri gibi temel özellikleri birbirinden farklı olan verilerin üzerinde gerekli işlemlerin yapılması durumunda ortak bir projede kullanılabileceğini göstermesi açısından önemlidir. Üstelik bu işlemlerin otomatik olarak yapılmasını sağlayan bir uygulamanın geliştirmiş olması,

minimum zaman ve çaba harcanarak istenilen bir bölge için veri seti elde etmeyi mümkün hale getirmektedir.

- Çalışma kapsamında hem OAM verilerinin hazırlanması hem de OSM verilerinin hazırlanması sürecinde çeşitli problem ve zorluklarla karşılaşmıştır. Özellikle her iki platformdan elde edilen verilerin ortak bir koordinat sistemine sahip olma gerekliliği birçok ek işlemin yapılmasını zorunlu kılmıştır. Yapılan bu işlemlerden başarılı sonuçlar elde edilmesi ve nihai sonuca varılması OAM, OSM gibi platformlardaki verilerin oldukça esnek yapılarda sunulduğu ve Python gibi yetenekli bir programlama dili ile bu veriler üzerinde çok geniş yelpazede işlemler yapılabileceğini göstermiştir.
- Veri seti hazırlamadaki bir diğer önemli sorun OSM veri modelinde yolları temsil etmek için kullanılan linestring nesnesinin yollara ait genişlik bilgisini içermemesiydi. Öncelikle mevcut yol verilerini kullanarak yol genişliklerini hesaplamak ve bu durumun mümkün olmadığı yollar için literatür taraması yapılarak elde edilen standart yol değerlerinin kullanılması bu çalışmanın kalitesine ve doğruluğuna önemli ölçüde katkı sağlamıştır.
- Çalışma kapsamında ortaya çıkarılan OSM2RN aracı ile dünyanın herhangi bir bölgesine ait OSM verilerine gerçek yol genişliklerini eklemek mümkün hale gelmiştir. Üstelik bu aracı kullanarak yol genişliklerini içeren güncel ve koordinat sistemine sahip yol ağları üretmek mümkün hale gelmiştir. Diğer yandan OSM2RN aracının bölgesel farklılıklara duyarlı hale getirilmesi bu araç ile elde edilen verilerin kalitesini ve hassasiyetini artıracaktır.
- Çalışma sonucunda üç farklı çalışma alanı için 113 bine yakın veri seti elde edilmiştir. Üstelik çalışma alanlarının sayısını artırarak hem veri seti miktarını hem de çeşitliliğini artırmak mümkündür. Üstelik hem OAM platformu hem de OSM platformu sürekli olarak veri miktarlarını artıran platformlardır. Bu yönü ile çalışma ele alındığında neredeyse sınırsız bir veri kaynağı elde edilmiştir. Makine öğrenmesi ve YSA için veri setinin önemi göz önüne alındığında bu çalışmanın bu yönden son derece başarılı bir sonuç ortaya çıkardığı düşünülmektedir.
- Çalışma kapsamında geliştirilen araç üzerinde küçük güncellemeler yaparak fiziksel çevreyi oluşturan binalar, alışveriş merkezleri, parklar, bankalar, ATM'ler gibi tüm nesneler için yeterli miktarda, kaliteli ve çeşitliliğe sahip veri seti üretmek mümkündür.

- Başta Avrupa ve Amerika kıtasındaki ülkeler olmak üzere gelişmiş birçok ülke OAM ve OSM verisi açısından oldukça zenginken ülkemiz için üretilen veri miktarı oldukça kısıtlıdır. Bu durum yeterli miktarda veri içeren ülkeler için daha fazla çalışma yapılmasına imkân vermektedir. Yapılan bu çalışmanın ülkemizdeki bir şehir yerine; diğer ülkelere ait şehirlerde veya bölgelerde yapılmış olması bu duruma örnektir.

Çalışma genel olarak değerlendirildiğinde büyük bölümünün açık kaynak ve ücretsiz platformlara dayandığı görülmektedir. Bu durum çalışma ile elde edilen sonuçların yeniden test edilmesi, geliştirilen araçların başka çalışmalara uyarlanması noktalarında ciddi bir esneklik sunmaktadır. Bu yönü ile ele alındığında yapılan çalışmayı geliştirme ve yeni özellikler veya işlevler ekleme konusunda ön plana çıkan öneriler aşağıda maddeler halinde sunulmuştur.

- Bu çalışma kapsamında elde edilen sonuçlar üretilen veri seti ile yol ağlarının tespitinin mümkün olduğunu göstermiştir. Diğer yandan veri seti miktarının ve çeşitliliğinin artırılması, daha uygun parametrelerin tespit edilmesini sağlamak amacıyla daha fazla uygulamanın ve testin yapılması geliştirilen modelin başarısını artıracakları öngörülmektedir.
- OAM mevcut yapısı ile çok geniş bir görüntü veritabanına sahiptir. Bu platformdaki görüntülerin mekânsal çözünürlükleri oldukça farklılık göstermektedir. Bu platformda mekânsal çözünürlüğü 10 metreden daha yüksek görüntüler olduğu gibi mekânsal çözünürlüğü 10 cm'nin altında olan büyük miktarda görüntü verisi vardır. Bu verilerin kullanım hakları genellikle açık kaynak topluluk projelerinde yaygın olarak tercih edilen lisanslama modelleri ile tanımlanmaktadır. Bu lisanslar çoğu zaman kullanıcılara hem ticari hem de ticari olmayan kullanım için geniş bir kullanım hakkı vermektedir. Dolayısıyla bu verileri farklı projelerde veya çalışmalarda kullanmak mümkündür. Diğer yandan teknolojinin giderek ucuzlaması ve yaygınlaşması bu platformların önümüzdeki süreçte daha fazla gelişip daha kaliteli içeriklere ev sahipliği yapacağı noktasında önemli ipuçları vermektedir. Bu çalışmadaki yol ağlarının tespiti için OAM platformundaki yüksek çözünürlüklü görüntülerin kullanılması bu çalışmalara bir örnektir. Benzer çalışmalar çevreyi oluşturan diğer nesneler için de yapılabilir.

- Çalışma kapsamında elde edilen veri setinin önemli bir bölümü hatta yarısından fazlası kullanıma uygun olmadığı için veri setine dahil edilmemiştir. İleride bu kaybı azaltmak adına yapılacak çalışmaların yararlı olacağı düşünülmektedir.
- Bu çalışmada birebir aynı bölgeye ait birden fazla uzaktan algılama görüntüsünün varlığı konusunda bir kontrol mekanizması kullanılmamıştır. Oysa aynı bölgeye ait uzaktan algılama görüntülerinin varlığı ve bu görüntülerin veri setinde kullanılması veri tekrarı oluşturmaktadır. Dolayısıyla bu konu ile ilgili yapılacak çalışma veya çalışmalar yararlı olacaktır.
- Elde edilen veri setinin daha güçlü bilgisayarlarda, daha fazla tur ile ve bu çalışma kapsamında kullanılan modellere ait parametreler için daha ince ayarların yapılması durumunda daha iyi sonuçların elde edilmesi imkan dahilindedir.
- OSM veritabanında hemen hemen dünyadaki her bölgeye ait, herkesin kullanımına açık ücretsiz veriler sunulmaktadır. OSM verilerinin güvenilirliği konusunda bazı endişeler olsa da bu konuda genel görüş, verilerin makul düzeyde güvenilir olduğunu göstermektedir. Dünya'nın daha az gelişmiş bölgelerinde OSM verisi nispeten daha azdır. Bu kategoriye Türkiye de dahildir. Örneğin bu çalışma kapsamında çalışma alanlarından birisinin Türkiye'deki bir bölge olması konusunda çaba gösterilmiş fakat özellikle yetersiz OSM verisinden dolayı çalışmaya dahil edilememiştir. Bu durum ülkemiz açısından değerlendirilmesi gereken bir konudur. Ülkemizde bu yönde gerekli çalışmaların yapılması önemlidir.
- Çalışma kapsamında iki boyutlu yol modelleri üretilmiştir. Oysa günümüzdeki teknik imkanlar göz önüne alındığında üç boyutlu yol modellerinin üretilmesi konusunda yapılacak çalışmalar yararlı olabilir.

KAYNAKÇA

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., ... Zheng, X. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv preprint arXiv:1603.04467*. <http://arxiv.org/abs/1603.04467> (Erişim tarihi: 30.11.2018).
- Abdollahi, A., Bakhtiari, H. R. R., and Nejad, M. P. (2018). Investigation of SVM and level set interactive methods for road extraction from google earth images. *Journal of the Indian Society of Remote Sensing*, 46(3), 423–430.
- Abdollahi, A., Pradhan, B., and Shukla, N. (2019). Extraction of road features from UAV images using a novel level set segmentation approach. *International Journal of Urban Sciences*, 1–15.
- Abdul-Rahman, A., ve Pilouk, M. (2007). *Spatial data modelling for 3D GIS*. Newyork:Springer.
- Alpaydin, E. (2016). *Machine learning : the new AI*. United State of America:MIT Press.
- Alshehhi, R., Reddy, P., Lee, W., and Dalla, M. (2017). ISPRS Journal of Photogrammetry and Remote Sensing Simultaneous extraction of roads and buildings in remote sensing imagery with convolutional neural networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, 130, 139–149.
- Anil, P. N., and Natarajan, S. (2010). A novel approach using active contour model for semi-automatic road extraction from high resolution satellite imagery. *2010 Second International Conference on Machine Learning and Computing*, 263–266.
- Anolytics. (2019). *Semantic Segmentation vs Object Detection - Difference*. <https://anolytics.home.blog/2019/11/12/difference-between-semantic-segmentation-and-object-detection/> (Erişim tarihi: 15.12.2019).
- Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12), 2481–2495.

- Barber, D. (2011). *Bayesian Reasoning and Machine Learning*. Newyork: Cambridge University Press.
- Barthakur, M., ve Sarma, K. K. (2020). Deep Learning Based Semantic Segmentation Applied to Satellite Image. J. Hemanth, M. Bhatia, ve O. Geman (Ed.), *Data Visualization and Knowledge Engineering: Spotting Data Points with Artificial Intelligence* (pp. 79–107). Switzerland: Springer International Publishing.
- Barzohar, M., and Cooper, D. B. (1996). Automatic finding of main roads in aerial images by using geometricstochastic models and estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7), 707–721.
- Ben-Hur, A., and Weston, J. (2010). A user’s guide to support vector machines. *Data mining techniques for the life sciences* (Vol. 609, pp. 223–239).
- Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 1798–1828.
- Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2), 157–166.
- Bennett, J. (2010). *OpenStreetMap - Be your own Cartographer*. Birmingham:Packt Publishing.
- Bhandare, A., Bhide, M., Gokhale, P., and Chandavarkar, R. (2016). Applications of Convolutional Neural Networks. *International Journal of Computer Science and Information Technologies*, 7(5), 2206–2215.
- Bhowmick, A., ve Hazarika, S. M. (2018). E-mail spam filtering: A review of techniques and trends. *Lecture Notes in Electrical Engineering*. https://doi.org/10.1007/978-981-10-4765-7_61 (Erişim tarihi: 15.12.2019).
- Bonaccorso, G. (2017). *Machine Learning Algorithms: Reference guide for popular algorithms for data science and machine learning*. Packt Publishing.

- Bonaccorso, G., Fandango, A., ve Shanmugamani, R. (2018). *Python: Advanced Guide to Artificial Intelligence: Expert machine learning systems and intelligent agents using Python*. Packt Publishing.
- Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). Training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, 144–152.
- Boykov, Y., and Jolly, M. P. (2000). Interactive organ segmentation using graph cuts. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1935, 276–286.
- Bradski, G. (2000a). *OpenCV library*. <https://opencv.org/> (Erişim tarihi: 31.12.2018).
- Bradski, G. (2000b). The OpenCV Library - Introduction OpenCV. *Dr. Dobb's Journal of Software Tools*. <https://docs.opencv.org/4.1.2/d1/dfb/intro.html> (Erişim tarihi: 05.12.2019).
- Brownlee, J. (2017). *Deep Learning with Time Series Forecasting*. Machine Learning Process; Machine Learning Mastery. <https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/> (Erişim tarihi: 26.01.2020).
- Çevre ve Şehircilik Bakanlığı. (2019). *Türkiye Ulusal Coğrafi Bilgi Sistemi Koordinat Referans Sistemleri ve Coğrafi Grid Sistemleri Kayıt Dokümanı Sürüm 1.0*. https://webdosya.csb.gov.tr/db/cbs/editordosya/TUCBS_JD1_1.pdf (Erişim tarihi: 23.01.2020).
- Charles, O., and Olivier, G. (2019). *Deep Learning Lectures*. https://m2dsupdlclass.github.io/lectures-labs/slides/04_conv_nets/index.html#1 (Erişim tarihi: 28.01.2020).
- Chauvin, Y., and Rumelhart, D. E. (1995). *Backpropagation : theory, architectures, and applications*. Lawrence Erlbaum Associates, Inc.
- Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. (2017). Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint arXiv:1706.05587*. <http://arxiv.org/abs/1706.05587> (Erişim tarihi: 12.01.2020).

- Chen, Z. (2006). *Research on High-resolution RS Image Classification Technology*. Ph. D thesis). Chinese Academy of Science, Beijing.
- Chicco, D., Sadowski, P., and Baldi, P. (2014). Deep autoencoder neural networks for gene ontology annotation predictions. *ACM BCB 2014 - 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, 533–540.
- Chollet, F., vd. (2015). *Keras*. <https://keras.io> (Erişim tarihi: 12.10.2019).
- Cipeluch, B., Jacob, R., Winstanley, A., and Mooney, P. (2010). Comparison of the accuracy of OpenStreetMap for Ireland with Google Maps and Bing Maps. *Accuracy 2010 - Proceedings of the 9th International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences*, 337–340.
- Ciregan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, 3642–3649. <https://ieeexplore.ieee.org/abstract/document/6248110> (Erişim tarihi: 16.01.2020).
- Coppin, B. (2004). *Artificial Intelligence Illuminated*. London: Jones & Bartlett.
- Cui, Y. (2020). *Artificial Intelligence and Judicial Modernization*. Singapore: Springer.
- Daniel Hoffman, B. K. (2019). *Meet RoboSat End-to-end feature extraction from aerial and satellite imagery*. <https://blog.mapbox.com/meet-robosat-af42530f163f> (Erişim tarihi: 03.12.2019).
- Daş, F. (2019). *OSM2RN Tool*. <https://github.com/fevzidas/osm2rn> (Erişim tarihi: 05.01.2020).
- Dass, R., and Devi, S. (2012). Image Segmentation Techniques 1 1,2. *International Journal of Electronis & Communication Technology*, 7109(1), 66–70.
- Dialani, P. (2019). *Top 10 Machine Learning Programming Languages | Analytics Insight*. <https://www.analyticsinsight.net/top-10-machine-learning-programming-languages/> (Erişim tarihi: 30.01.2020).
- Diederik, K., and Ba, J. L. (2014). ADAM: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*. <https://arxiv.org/abs/1412.6980> (Erişim tarihi: 11.01.2020).

- England Highways. (2012). *Design approach statement - Roads*. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/405938/HS2_Rural_Road_Design_Criteria.pdf (Eriřim tarihi: 03.01.2020).
- Environmental Systems Research Institute. (2010). Coordinate systems, map projections, and geographic (datum) transformations. *Environmental Systems Research Institute*, 1–5. <http://resources.esri.com/help/9.3/arcgisengine/dotnet/89b720a5-7339-44b0-8b58-0f5bf2843393.htm> (Eriřim tarihi: 01.25.2020).
- Ergin, T. (2019). *Convolutional Neural Network (ConvNet yada CNN) nedir, nasıl çalışır?* <https://medium.com/@tuncerergin/convolutional-neural-network-convnet-yada-cnn-nedir-nasil-calisir-97a0f5d34cad> (Eriřim tarihi: 10.12.2019).
- Fearnley, J. (2015). *Technical Standard-Roads*. <http://programmeofficers.co.uk/Cuadrilla2018/CD8/CD8.10.pdf> (Eriřim tarihi: 02.07.2019).
- Fernández, C., Fernández-Llorca, D., and Sotelo, M. A. (2017). A Hybrid Vision-Map Method for Urban Road Detection. *Journal of Advanced Transportation*, 2017. <https://www.hindawi.com/journals/jat/2017/7090549/> (Eriřim tarihi: 06.12.2019).
- Fua, P., and Leclerc, Y. G. (1990). Model driven edge detection. *Machine Vision and Applications*, 3(1), 45–56.
- Fukuda, K., Maadi, S., Shen, K., and Schmöcker, J.-D. (2019). *Extraction of public transportation networks from Openstreetmap data and estimation of OD patterns through a graph convolutional approach*. http://transitdata2019.fr/wp-content/uploads/2019/07/10_09_01_57_PRESENTATION_SCHMOECKER.pdf (Eriřim tarihi: 24.01.2020).
- Galea, A., and Capelo, L. (2018). *Applied deep learning with Python : use scikit-learn, TensorFlow, and Keras to create intelligent systems and machine learning solutions*. Packt Publishing Ltd. <https://learning.oreilly.com/library/view/applied-deep-learning/9781789804744/> (Eriřim tarihi: 22.01.2020).
- Gatys, L., Ecker, A., and Bethge, M. (2016). A Neural Algorithm of Artistic Style. *Journal of Vision*, 16(12), 326.

- Geng, L., Sun, J., Xiao, Z., Zhang, F., and Wu, J. (2018). Combining CNN and MRF for road detection. *Computers and Electrical Engineering*, 70, 895–903.
- George, J., Mary, L., and Riyas, K. S. (2013). Vehicle detection and classification from acoustic signal using ANN and KNN. *2013 International Conference on Control Communication and Computing, ICCCC 2013*, 436–439.
- Ghahramani, Z. (2004). Unsupervised Learning BT - Advanced Lectures on Machine Learning. *Advanced Lectures on Machine Learning*, 3176(Chapter 5), 72–112. http://link.springer.com/10.1007/978-3-540-28650-9_5 (Erişim tarihi: 02.02.2020).
- Gilbert, M. (2018). *Artificial Intelligence for Autonomous Networks*. Boca Raton: CRC Press.
- Gillies, S., and vd. (2013). *Rasterio: geospatial raster I/O for Python programmers*. <https://github.com/mapbox/rasterio> (Erişim tarihi: 18.06.2019).
- Glorot, X., and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research*, 9, 249–256.
- Goodchild, M. F. (2007). Citizens as Sensors: The World of Volunteered Geography. *GeoJournal*, 69(4), 211–221. <http://link.springer.com/10.1007/s10708-007-9111-y> (Erişim tarihi: 25.06.2019).
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. London: MIT Press.
- Gouk, H., Frank, E., Pfahringer, B., and Cree, M. (2018). Regularisation of Neural Networks by Enforcing Lipschitz Continuity. *arXiv preprint arXiv:1804.04368*. <http://arxiv.org/abs/1804.04368> (Erişim tarihi: 24.01.2020).
- Graves, A., Eck, D., Beringer, N., and Schmidhuber, J. (2004). Biologically plausible speech recognition with LSTM neural nets. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3141, 127–136.

- Gross, F., Jovanis, P. P., and Eccles, K. (2009). Safety effectiveness of lane and shoulder width combinations on rural, two-lane, undivided roads. *Transportation Research Record*, 2103(2103), 42–49.
- Gruen, A., and Li, H. (1997). Semi-automatic linear feature extraction by dynamic programming and LSB-Snakes. *Photogrammetric Engineering and Remote Sensing*, 63(8), 985–995.
- Guibault, L. M. C. R., and Wiebe, A. (2013). *Safe to be open : study on the protection of research data and recommendations for access and usage*. Germany:University of Göttingen.
- Gülcü, A., ve Kuş, Z. (2019). Konvolüsyonel Sinir Ağlarında Hiper-Parametre Optimizasyonu Yöntemlerinin İncelenmesi. *Gazi Üniversitesi Fen Bilimleri Dergisi Part C: Tasarım ve Teknoloji*, 7(2), 503–522.
- Gupta, D. (2017). *GitHub - divamgupta/image-segmentation-keras: Implementation of Segnet, FCN, UNet , PSPNet and other models in Keras*. <https://github.com/divamgupta/image-segmentation-keras> (Erişim tarihi: 04.07.2019).
- Hare, E., and Kaplan, A. (2017). Designing Modular Software: A Case Study in Introductory Statistics. *Journal of Computational and Graphical Statistics*, 26(3), 493–500.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-Decem*, 770–778.
- Heermann, P. D., and Khazenie, N. (1992). Classification of multispectral remote sensing data using a back-propagation neural network. *IEEE Transactions on Geoscience and Remote Sensing*, 30(1), 81–88.

- Helbich, M., Amelunxen, C., and Neis, P. (2012). Comparative Spatial Analysis of Positional Accuracy of OpenStreetMap and Proprietary Geodata. *Proceedings of GI_Forum 2012: Geovisualization, Society and Learning*, 24–33. http://koenigstuhl.geog.uni-heidelberg.de/publications/2010/Helbich/Helbich_etal_AGILE2011.pdf (Eriřim tarihi: 08.12.2019).
- Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*. <http://arxiv.org/abs/1207.0580> (Eriřim tarihi: 11.01.2020).
- Hinz, S., and Baumgartner, A. (2003). Automatic extraction of urban road networks from multi-view aerial imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 58(1–2), 83–98.
- Homma, K., Yamamoto, H., and Iwata, Y. (2000). A Study On Image Restoration For Airborne Cameras. *International Archives of Photogrammetry and Remote Sensing*, XXXIII(B7), 545–549. http://www.isprs.org/proceedings/xxxiii/congress/part7/545_XXXIII-part7.pdf (Eriřim tarihi: 23.11.2019).
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*. <http://arxiv.org/abs/1704.04861> (Eriřim tarihi: 12.01.2020).
- Hua, G., and Jégou, H. (2016). *Computer Vision--ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings* (Vol. 9914). Springer.
- Humanitarian OpenStreetMap Team. (2019). *What We Do*. <https://www.hotosm.org/what-we-do> (Eriřim tarihi: 10.09.2019).
- IMSMA Wiki Community. (2019). *File:Raster vs Vector*. http://mwiki.gichd.org:8090/IM/File:Raster_vs_Vector_a.jpg (Eriřim tarihi: 03.01.2020).

- Inik, Ö., ve Ülker, E. (2017). Derin öğrenme ve görüntü analizinde kullanılan derin öğrenme modelleri. *Gaziosmanpaşa Bilimsel Araştırma Dergisi (GBAD)*. <https://dergipark.org.tr/tr/download/article-file/380999> (Erişim tarihi: 10.11.2019).
- Ioffe, S., and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *32nd International Conference on Machine Learning, ICML 2015*. <https://arxiv.org/abs/1502.03167> (Erişim tarihi: 12.12.2019).
- Jassim, F. A. (2013). Increasing Compression Ratio in PNG Images by k-Modulus Method for Image Transformation. *arXiv preprint arXiv:1307.0036*. <https://arxiv.org/abs/1307.0036> (Erişim tarihi: 09.01.2020).
- Jetbrains. (2019). Pycharm. <https://www.jetbrains.com/pycharm/> (2019.3). <https://www.jetbrains.com/pycharm/> (Erişim tarihi: 09.15.2019).
- Jha, K., Doshi, A., Patel, P., and Shah, M. (2019). A comprehensive review on automation in agriculture using artificial intelligence. *Artificial Intelligence in Agriculture*, 2, 1–12.
- Jokar Arsanjani, J., Zipf, A., Mooney, P., and Helbich, M. (2015). OpenStreetMap in GIScience. *OpenStreetMap in GIScience: Experiences, Research, Applications, JANUARY*, 1–20. <http://link.springer.com/10.1007/978-3-319-14280-7> (Erişim tarihi: 09.10.2019).
- Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. (2016). *Exploring the Limits of Language Modeling*. <http://arxiv.org/abs/1602.02410> (Erişim tarihi: 03.01.2020).
- Kala, R. (2016). On-Road Intelligent Vehicles: Motion Planning for Intelligent Transportation Systems. *On-Road Intelligent Vehicles: Motion Planning for Intelligent Transportation Systems Allahabad:Elsevier*.
- Karim, M. R., Sewak, M., and Pujari, P. (2018). *Practical Convolutional Neural Networks : Implement advanced deep learning models using Python*. (p. 211). Packt Publishing.

- Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International journal of computer vision*, 1(4), 321–331.
- Kaur, D., and Kaur, Y. (2014). International Journal of Computer Science and Mobile Computing Various Image Segmentation Techniques: A Review (Vol. 3, Num. 5), 809--814.
- Khan, S., Rahmani, H., Shah, S. A. A., and Bennamoun, M. (2018). *A Guide to Convolutional Neural Networks for Computer Vision* (Vol. 8, Sayı 1). Morgan & Claypool Publishers.
<http://www.morganclaypool.com/doi/10.2200/S00822ED1V01Y201712COV015>
 (Erişim tarihi: 06.01.2020).
- Kim, B. (2017). *Github - Machine learning toolkits with python*.
<https://github.com/buomsoo-kim/Machine-learning-toolkits-with-python/blob/master/README.md> (Erişim tarihi: 09.01.2020).
- Kirthika, A., and Mookambiga, A. (2011). Automated road network extraction using artificial neural network. *International Conference on Recent Trends in Information Technology, ICRTIT 2011*, 1061–1065.
- Koenig, S., and Simmons, R. G. (1996). Unsupervised Learning of Probabilistic Models for Robot Navigation. *Proceedings - IEEE International Conference on Robotics and Automation*, 3, 2301–2308.
- Kotsiantis, S. B., Kanellopoulos, D., and Pintelas, P. E. (2006). Data preprocessing for supervised learning. *International Journal of Computer Science*, 1(2), 111–117.
- Laddha, A., Kocamaz, M. K., Navarro-Serment, L. E., and Hebert, M. (2016). Map-supervised road detection. *2016 IEEE Intelligent Vehicles Symposium (IV)*, 118–123.
- Larose, C. D., and Larose, D. T. (2019). *Data Science Using Python and R*. United State of America:Wiley.
- LeCun, Y. (1989). Generalization and network design strategies. *Connectionism in perspective*, 143–155. <https://www.academia.edu/download/30766382/lecun.pdf>
 (Erişim tarihi: 02.01.2020).

- Lecun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature* (Vol. 521, Sayı 7553, pp. 436–444).
- Lee, H. Y., Park, W., Lee, H. K., and Kim, T. G. (2000). Towards knowledge-based extraction of roads from 1 m-resolution satellite images. *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation, 2000-Janua*, 171–176.
- Li, D., ve Du, Y. (2017). *Artificial intelligence with uncertainty, second edition* (Second Edi). CRC Press. <https://doi.org/10.1201/9781315366951>
- Li, J., and Chen, M. (2014). On-road multiple obstacles detection in dynamical background. *Proceedings - 2014 6th International Conference on Intelligent Human-Machine Systems and Cybernetics, IHMSC 2014, 1*, 102–105.
- Li, Q., Fan, H., Luan, X., Yang, B., and Liu, L. (2014). Polygon-based approach for extracting multilane roads from OpenStreetMap urban road networks. *International Journal of Geographical Information Science*, 28(11), 2200–2219. <http://www.tandfonline.com/doi/full/10.1080/13658816.2014.915401> (Erişim tarihi: 12.11.2019).
- Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., van der Laak, J. A. W. M., van Ginneken, B., and Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis* (Vol. 42, pp. 60–88). <http://arxiv.org/abs/1702.05747> (Erişim tarihi: 09.08.2019).
- Liu, S., Jia, J., Fidler, S., and Urtasun, R. (2017). SGN: Sequential Grouping Networks for Instance Segmentation. *Proceedings of the IEEE International Conference on Computer Vision, 2017-Octob*, 3516–3524.
- Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431–3440.
- Lu, Y., Gao, K., Zhang, T., and Xu, T. (2018). A novel image registration approach via combining local features and geometric invariants. *PLoS ONE*, 13(1). <http://dx.plos.org/10.1371/journal.pone.0190383> (Erişim tarihi: 03.01.2020).

- Ma, R., Wang, W., and Liu, S. (2012). Extracting roads based on Retinex and improved Canny operator with shape criteria in vague and unevenly illuminated aerial images. *Journal of Applied Remote Sensing*, 6(1), 63610.
- Mattyus, G., Luo, W., and Urtasun, R. (2017). DeepRoadMapper: Extracting Road Topology from Aerial Images. *Proceedings of the IEEE International Conference on Computer Vision, 2017*, 3458–3466.
- Maurya, R., Gupta, P. R., and Shukla, A. S. (2011). Road extraction using k-means clustering and morphological operations. *2011 International Conference on Image Information Processing*, 1–6.
- McCarthy, J., Minsky, M. L., and Rochester, N. (1956). The Dartmouth summer research project on artificial intelligence. *Artificial intelligence: past, present, and future*.
- McCulloch, W. S., and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4), 115–133. <http://link.springer.com/10.1007/BF02478259> (Erişim tarihi: 21.01.2020).
- Melgani, F., and Bruzzone, L. (2004). Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on geoscience and remote sensing*, 42(8), 1778–1790.
- Mena, J. B. (2003). State of the art on automatic road extraction for GIS update: A novel classification. *Pattern Recognition Letters*, 24(16), 3037–3058.
- Mendes, C. C. T., Fremont, V., and Wolf, D. F. (2016). Exploiting fully convolutional neural networks for fast road detection. *Proceedings - IEEE International Conference on Robotics and Automation, 2016-June*, 3174–3179.
- Miao, Z., Wang, B., Shi, W., and Zhang, H. (2014). A semi-automatic method for road centerline extraction from VHR images. *IEEE Geoscience and Remote Sensing Letters*, 11(11), 1856–1860. <http://ieeexplore.ieee.org/document/6784347/> (Erişim tarihi: 11.11.2019).
- Minsky, M., and Papert, S. (1969). Perceptrons: expanded edition. *MIT Press Cambridge MA* (Vol. 522, p. 20). MIT Press, Cambridge. <http://mitpress.mit.edu/book-home.tcl?isbn=0262631113> (Erişim tarihi: 04.01.2020).

- Mirvahabi, S. S., and Abbaspour, R. A. (2015). Automatic extraction of IndoorGML core model from OpenStreetMap. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 40(1W5), 459–462.
- Mnih, V., and Hinton, G. E. (2010). Learning to detect roads in high-resolution aerial images. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6316 LNCS(PART 6), 210–223.
- Mokhtarzade, M., and Zoej, M. J. V. (2007). Road detection from high-resolution satellite images using artificial neural networks. *International journal of applied earth observation and geoinformation*, 9(1), 32–40.
- Moolayil, J. (2019). *Learn Keras for Deep Neural Networks*. Springer.
- Movaghati, S., Moghaddamjoo, A., and Tavakoli, A. (2010). Road extraction from satellite images using particle filtering and extended Kalman filtering. *IEEE Transactions on geoscience and remote sensing*, 48(7), 2807–2817.
- Naranjo-Alcazar, J., Perez-Castanos, S., Martin-Morato, I., Zuccarello, P., and Cobos, M. (2019). On the performance of residual block design alternatives in convolutional neural networks for end-to-end audio classification. *arXiv preprint arXiv:1906.10891*. <http://arxiv.org/abs/1906.10891> (Erişim tarihi: 03.01.2020).
- Niu, X. (2006). A semi-automatic framework for highway extraction and vehicle detection based on a geometric deformable model. *ISPRS Journal of Photogrammetry and Remote Sensing*, 61(3–4), 170–186.
- Official Journal of the European Communities. (1996). *Council Directive 96/5 3/EC of 25 July 1996*. <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:31996L0053&from=en> (Erişim tarihi: 18.10.2019).
- Open Data Commons. (2019a). *ODC Open Database License*. <https://opendatacommons.org/licenses/odbl/summary/index.html> (Erişim tarihi: 18.12.2019).

- Open Data Commons. (2019b). *Open Database License (ODbL) v1.0*.
<https://www.opendatacommons.org/licenses/odbl/1.0/> (Erişim tarihi: 18.12.2019)
- OpenAerialMap Contributors. (2019). *About OpenAerialMap*.
<https://openaerialmap.org/about/> (Erişim tarihi: 13.01.2020).
- OpenAerialMap Docs. (2019a). *OpenAerialMap Ecosystem*.
<https://docs.openaerialmap.org/ecosystem/catalog-group/> (Erişim tarihi: 04.01.2020).
- OpenAerialMap Docs. (2019b). *OpenAerialMap Ecosystem*.
<https://docs.openaerialmap.org/ecosystem/contributor-group/> (Erişim tarihi: 04.01.2020).
- OpenStreetMap contributors. (2018). *Converting to WGS84 - OpenStreetMap Wiki*.
https://wiki.openstreetmap.org/wiki/Converting_to_WGS84 (Erişim tarihi: 06.01.2020).
- OpenStreetMap Contributors. (2016a). *File:Osm data structures - future of areas - supermultipolygon.svg - OpenStreetMap Wiki*.
https://wiki.openstreetmap.org/wiki/File:Osm_data_structures_-_future_of_areas_-_supermultipolygon.svg (Erişim tarihi: 11.01.2020).
- OpenStreetMap Contributors. (2016b). *OSM data structures*.
<https://wiki.openstreetmap.org/wiki/Relation> (Erişim tarihi: 04.01.2020).
- OpenStreetMap Contributors. (2018). *OpenAerialMap - OpenStreetMap Wiki*.
<https://wiki.openstreetmap.org/wiki/OpenAerialMap> (Erişim tarihi: 04.01.2020).
- OpenStreetMap Contributors. (2019a). *Bounding Box - OpenStreetMap Wiki*.
https://wiki.openstreetmap.org/wiki/Bounding_Box (Erişim tarihi: 02.01.2020).
- OpenStreetMap Contributors. (2019b). *Key:highway - OpenStreetMap Wiki*.
<https://wiki.openstreetmap.org/wiki/Key:highway> (Erişim tarihi: 12.11.2019).
- OpenStreetMap Contributors. (2019c). *Planet OSM*. <https://planet.openstreetmap.org/> (Erişim tarihi: 04.01.2020).

- Osher, S., and Sethian, J. A. (1988). Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of computational physics*, 79(1), 12–49.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Peng, B., Zhang, L., and Zhang, D. (2013). A survey of graph theoretical approaches to image segmentation. *Pattern recognition*, 46(3), 1020–1038.
- QGIS Development Team. (2019). *QGIS Geographic Information System. Open Source Geospatial Foundation Project* (3.4, 3.10). <http://qgis.osgeo.org> (Erişim tarihi: 24.08.2019).
- Raifer, M. (2013). *GitHub - Convert OSM to GeoJSON*. <https://github.com/tyrasd/osmtogeojson> (Erişim tarihi: 04.07.2019).
- Raschka, S., and Mirjalili, V. (2017). *Python Machine Learning* (2nd Edition). Birmingham:Packt Publishing.
- Renaud, P., and Thierry, R. (2003). Multi-resolution snakes for urban road extraction from ikonos and quickbird images. *Proc. of the 23rd EARSeL Symposium*, 69–76.
- Reza Tizhoosh, H., and Pantanowitz, L. (2018). Artificial intelligence and digital pathology: Challenges and opportunities. *Journal of Pathology Informatics*, 9(1).
- Rezaee, M., and Zhang, Y. (2017). Road detection using Deep Neural Network in high spatial resolution images. *2017 Joint Urban Remote Sensing Event, JURSE 2017*, 1–8. <http://ieeexplore.ieee.org/document/7924619/> (Erişim tarihi: 09.11.2019).
- Roess, R. P., and Prassas, E. S. (2014). *The Highway Capacity Manual: A Conceptual and Research History* (Vol. 5). <http://link.springer.com/10.1007/978-3-319-05786-6> (Erişim tarihi: 17.10.2019).

- Romero, A., Gatta, C., and Camps-Valls, G. (2015). Unsupervised deep feature extraction for remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 54(3), 1349–1362.
- Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*, 234–241.
- Rosenblatt, F. (1957). *The Perceptron, a Perceiving and Recognizing Automaton Project Para*. Cornell Aeronautical Laboratory.
- Rumelhart, David E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533–536. <http://www.nature.com/articles/323533a0> (Erişim tarihi: 04.01.2020).
- Rumelhart, D. E., ve MacClelland, J. L. (1986). Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Foundations. 1, Volume 1. *MIT Press* (Vol. 1, Issue June, ss. 318-- 362). MIT Press, Cambridge, MA. (Erişim tarihi: 03.01.2020).
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3), 211–252.
- Russell, S., and Norvig, P. (2009). *Artificial Intelligence A Modern Approach* (3rd Editio). Pearson.
- Sameen, M. I., Pradhan, B., and Aziz, O. S. (2018). Classification of very high resolution aerial photos using spectral-spatial convolutional neural networks. *Journal of Sensors*, 2018, 1–12. <https://www.hindawi.com/journals/js/2018/7195432/> (Erişim tarihi: 04.07.2019).
- Samuel, A. L. (1967). Some studies in machine learning using the game of checkers. II—Recent progress. *IBM Journal of research and development*, 11(6), 601–617.
- Sayood, K. (2002). *Lossless compression handbook*. United State of America:Elsevier.

- Schiaffino, S., Garcia, P., and Amandi, A. (2008). eTeacher: Providing personalized assistance to e-learning students. *Computers & Education*, 51(4), 1744–1754.
- Schmidhuber, J. (2015). *Deep Learning in neural networks: An overview* (Vol. 61, pp. 85–117). Elsevier.
- Schneider, M., Guthe, M., and Klein, R. (2005). Real-time rendering of complex vector data on 3D terrain models. *11th International Conference on Virtual Systems and Multimedia*, 10, 573–582.
- Şeker D. Z. ve Eker, O. (2005). *Aktif Kontur Modeller ve Düzey Kümesi Kullanarak Çizgisel Detayların Yarı Otomatik Olarak Çizilmesi*. https://www.hkmo.org.tr/resimler/ekler/29OF_179_ek.pdf (Erişim tarihi: 26.07.2019).
- Shanmugamani, R. (2018). *Deep Learning for Computer Vision: Expert techniques to train advanced neural networks using TensorFlow and keras*. Packt Publishing.
- Shen, J., Lin, X., Shi, Y., and Wong, C. (2008). Knowledge-based road extraction from high resolution remotely sensed imagery. *2008 Congress on Image and Signal Processing*, 4, 608–612.
- Simler, C. (2011). An improved road and building detector on VHR images. *International Geoscience and Remote Sensing Symposium (IGARSS)*, 507–510.
- Singh, P. P., and Garg, R. D. (2013). Automatic road extraction from high resolution satellite image using adaptive global thresholding and morphological operations. *Journal of the Indian Society of Remote Sensing*, 41(3), 631–640.
- Smith, N., Cristiano, G., Rob, E., and Ian, S. (2016). *OpenAerialMap: Empowering Disaster Response and Preparedness Communities with Aerial Imagery*. <https://pdfs.semanticscholar.org/feba/2ecdd4be0625806a649bbac7f6f61b59c581.pdf> (Erişim tarihi: 04.01.2020).
- St. Laurent, A. M. S. (2004). *Understanding Open Source and Free Software Licensing*. O'Reilly Media. <http://www.ariadne.ac.uk/issue42/fraser-rvw/> (Erişim tarihi: 06.01.2020).

- Steinmetz, D., Dyballa, D., Ma, H., and Hartmann, S. (2018). Using a conceptual model to transform road networks from openstreetmap to a graph database. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11157 LNCS, 301–315.
- Tao, W., Jin, H., and Liu, L. (2007). A new image thresholding method based on graph cuts. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 1(1), 110.
- Thomas, A. (2017). Datums, Projections and Coordinate Systems. *Recorder*, 42(02). <https://csegrecorder.com/articles/view/datums-projections-and-coordinate-systems> (Eriřim tarihi: 04.01.2020).
- Tönjes, R., and Growe, S. (1998). Knowledge based road extraction from multisensor imagery. *International Archives of Photogrammetry and Remote Sensing*, 32, 387–393.
- Tu-Ko, K. (2003). A hybrid road identification system using image processing techniques and backpropagation neural network. *Mississippi State Univ., Mississippi State, MS, Tech. Report MS*, 39, 762. <https://dl.acm.org/doi/book/10.5555/890040> (Eriřim tarihi: 13.11.2019).
- Tupin, F., Maitre, H., Mangin, J.-F., Nicolas, J.-M., and Pechersky, E. (1998). Detection of linear features in SAR images: Application to road network extraction. *IEEE transactions on geoscience and remote sensing*, 36(2), 434–453.
- Unsalan, C., and Sirmacek, B. (2012). Road network detection using probabilistic and graph theoretical methods. *IEEE Transactions on Geoscience and Remote Sensing*, 50(11), 4441–4453.
- Valada, A., Oliveira, G. L., Brox, T., and Burgard, W. (2017). Deep Multispectral Semantic Scene Understanding of Forested Environments Using Multimodal Fusion. *International Symposium on Experimental Robotics (ISER)*, 465–477.
- Valero, S., Chanussot, J., Benediktsson, J. A., Talbot, H., and Waske, B. (2010). Advanced directional mathematical morphology for the detection of the road network in very high resolution remote sensing images. *Pattern Recognition Letters*, 31(10), 1120–1127.

- Wang, B., and Gong, N. Z. (2018). Stealing Hyperparameters in Machine Learning. *Proceedings - IEEE Symposium on Security and Privacy, 2018-May*, 36–52.
- Wang, Jianhua, Qin, Q., Zhao, J., Ye, X., Feng, X., Qin, X., and Yang, X. (2015). Knowledge-based detection and assessment of damaged roads using post-disaster high-resolution remote sensing image. *Remote Sensing*, 7(4), 4948–4967.
- Wang, Jun, Song, J., Chen, M., and Yang, Z. (2015). Road network extraction: a neural-dynamic framework based on deep learning and a finite state machine. *International Journal of Remote Sensing*, 36(12), 3144–3169.
- Wang, M., and Luo, C. J. (2005). Extracting roads based on Gauss Markov random field texture model and support vector machine from high-resolution RS image. *IEEE Trans. Geosci. Remote Sens.*, 9, 271–276.
- Wang, W., Yang, N., Zhang, Y., Wang, F., Cao, T., and Eklund, P. (2016). A review of road extraction from remote sensing images. *Journal of Traffic and Transportation Engineering (English Edition)*, 3(3), 271–282.
- Wang, Y., and Zheng, Q. (1998). Recognition of roads and bridges in SAR images. *Pattern recognition*, 31(7), 953–962.
- Widmann, M. (2019). *From Modeling to Scoring: Confusion Matrix and Class Statistics* | KNIME. <https://www.knime.com/blog/from-modeling-to-scoring-confusion-matrix-and-class-statistics> (Erişim tarihi: 12.01.2020).
- Widrow, B., and Hoff, M. E. (1960). *Adaptive switching circuits*.
- Wikimedia Commons. (2019a). *A Simple Neural Network - Training strategy*. <https://commons.wikimedia.org/> (Erişim tarihi: 14.10.2019).
- Wikimedia Commons. (2019b). *Reinforcement learning diagram*. <https://commons.wikimedia.org/wiki/File%3ALambdaPlaques.jpg> (Erişim tarihi: 14.10.2019).
- Wu, S., Du, C., Chen, H., Xu, Y., Guo, N., and Jing, N. (2019). Road Extraction from Very High Resolution Images Using Weakly labeled OpenStreetMap Centerline. *ISPRS International Journal of Geo-Information*, 8(11), 478.

- Wulamu, A., Shi, Z., Zhang, D., and He, Z. (2019). Multiscale Road Extraction in Remote Sensing Images. *Computational Intelligence and Neuroscience*, 2019.
- Xin, J., Zhang, X., Zhang, Z., and Fang, W. (2019). Road Extraction of High-Resolution Remote Sensing Images Derived from DenseUNet. *Remote Sensing*, 11(21), 2499.
- Yager, N., and Sowmya, A. (2003). Support vector machines for road extraction from remotely sensed images. *International Conference on Computer Analysis of Images and Patterns*, 285–292.
- Yousif, O., and Ban, Y. (2014). Improving SAR-based urban change detection by combining MAP-MRF classifier and nonlocal means similarity weights. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(10), 4288–4300.
- Zeiler, M. (1999). *Modeling our world: the ESRI guide to geodatabase design*. ESRI, Inc.
- Zeiler, M. D. (2012). *ADADELTA: An Adaptive Learning Rate Method*. <http://arxiv.org/abs/1212.5701> (Erişim tarihi: 14.12.2019).
- Zhang, C., Murai, S., and Baltsavias, E. P. (1999). Road network detection by mathematical morphology. *Bulletin - Societe Francaise de Photogrammetrie et de Teledetection*, 153, 94–96.
- Zhang, L., Yang, F., Zhang, Y. D., and Zhu, Y. J. (2016). Road crack detection using deep convolutional neural network. *2016 IEEE international conference on image processing (ICIP)*, 3708–3712.
- Zhang, Z., Liu, Q., and Wang, Y. (2018). Road extraction by deep residual U-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5), 749–753.
- Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid scene parsing network. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2881–2890.
- Zheng, Q., Li, W., Hu, W., and Wu, G. (2012). An interactive image segmentation algorithm based on graph cut. *Procedia Engineering*, 29, 1420–1424.

- Zhou, J., Bischof, W. F., and Caelli, T. (2006). Road tracking in aerial images based on human--computer interaction and Bayesian filtering. *ISPRS journal of photogrammetry and remote sensing*, 61(2), 108–124.
- Zhou, X., Takayama, R., Wang, S., Hara, T., and Fujita, H. (2017). Deep learning of the sectional appearances of 3D CT images for anatomical structure segmentation based on an FCN voting method. *Medical physics*, 44(10), 5221–5233.
- Zhu, C., Shi, W., Pesaresi, M., Liu, L., Chen, X., and King, B. (2005). The recognition of road network from high-resolution satellite remotely sensed data using image morphological characteristics. *International Journal of Remote Sensing*, 26(24), 5493–5508.
- Zhu, D.-M., Wen, X., and Ling, C.-L. (2011). Road extraction based on the algorithms of MRF and hybrid model of SVM and FCM. *2011 International Symposium on Image and Data Fusion*, 1–4.

EKLER

EK-1: Çalışma Esnasında Kullanılan Python Kütüphaneleri ve Sürüm Numaraları

absl-py==0.9.0, affine==2.2.2, altair==3.1.0, appdirs==1.4.3, asn1crypto==0.24.0, astor==0.8.1, atomicwrites==1.3.0, attrs==19.1.0, bleach==3.1.0, boto3==1.9.159, botocore==1.12.159, branca==0.3.1, cachetools==3.1.1, Cartopy==0.17.0, certifi==2019.3.9, cffi==1.12.3, cftime==1.0.4.2, chardet==3.0.4, Click==7.0, click-plugins==1.1.1, cligj==0.5.0, cloudpickle==1.2.2, cryptography==2.7, cycler==0.10.0, dask==2.8.0, datacube==1.7, decorator==4.4.0, Deprecated==1.2.7, descartes==1.1.0, docutils==0.14, entrypoints==0.3, Fiona==1.8.6, Flask==1.0.2, folium==0.9.1, gast==0.2.2, GDAL==2.4.1, geocube==0.0.9, geographiclib==1.49, geojson==2.4.1, geopandas==0.5.0, geopy==1.20.0, gitdb2==2.0.6, GitPython==3.0.5, glob2==0.7, google-auth==1.10.0, google-auth-oauthlib==0.4.1, google-pasta==0.1.8, grpcio==1.26.0, h5py==2.10.0, idna==2.8, imageio==2.6.1, imgaug==0.3.0, imutils==0.5.3, itsdangerous==1.1.0, Jinja2==2.10.1, jmespath==0.9.4, jsonschema==3.0.1, Keras==2.3.1, Keras-Applications==1.0.8, Keras-Preprocessing==1.1.0, keras-segmentation==0.3.0, kiwisolver==1.1.0, lark-parser==0.7.8, lxml==4.3.4, mapclassify==2.0.1, Markdown==3.1.1, MarkupSafe==1.1.1, matplotlib==3.1.0, mercantile==1.0.4, more-itertools==7.0.0, munch==2.3.2, natsort==6.2.0, netCDF4==1.5.3, networkx==2.3, numexpr==2.6.9, numpy==1.16.4, oauthlib==3.1.0, ogr==0.8.0, olefile==0.46, opencv-contrib-python==4.1.0.25, opencv-python==4.1.2.30, opencv-python-headless==4.1.2.30, opt-einsum==3.1.0, osmium==2.15.2, osmnx==0.10, overpy==0.4, OWSLib==0.17.1, pandas==0.24.2, Pillow==6.0.0, pkginfo==1.5.0.1, pluggy==0.11.0, protobuf==3.11.1, pycopg2==2.8.4, py==1.8.0, pyasn1==0.4.8, pyasn1-modules==0.2.7, pycparser==2.19, PyCRS==1.0.1, pyepsg==0.4.0, PyGithub==1.44.1, Pygments==2.4.2, PyJWT==1.7.1, pykdtree==1.3.1, pyOpenSSL==19.0.0, pyparsing==2.4.0, pyPEG2==2.15.2, pyproj==2.1.3, pyparsing==2.4.0, pysocks==1.7.0, pytest==4.5.0, python-dateutil==2.8.0, python-gitlab==1.13.0, pytz==2019.1, PyWavelets==1.1.1, PyYAML==5.1.2, rasterio==1.0b1, readme-renderer==24.0, requests==2.22.0, requests-oauthlib==1.3.0, requests-toolbelt==0.9.1, rio-mucho==1.0.0, rio-pansharpen==0.2.0, rio-tiler==1.0a7, rio-toa==0.3.0

rioxarray==0.0.17, rsa==4.0, Rtree==0.8.3, s3transfer==0.2.0, scikit-image==0.16.2,
scipy==1.3.0, Shapely==1.6.4.post2, simplejson==3.17.0, singledispatch==3.4.0.3,
six==1.12.0, smmap2==2.0.5, snuggs==1.4.6, SQLAlchemy==1.3.11,
supermercado==0.0.5, tensorboard==2.0.2, tensorflow==2.0.0, tensorflow-
estimator==2.0.1, termcolor==1.1.0, toml==0.10.0, toolz==0.9.0, torch==1.1.0,
torchvision==0.2.2.post3, tornado==6.0.2, tqdm==4.32.2, twine==1.13.0,
urllib3==1.25.3, vincent==0.4.4, wcwidth==0.1.7, webencodings==0.5.1,
Werkzeug==0.15.4, wget==3.2, wrapt==1.11.2, xarray==0.14.0



ÖZGEÇMİŞ

Kişisel Bilgiler

Adı, Soyadı : Fevzi Daş
Doğum Yeri / Yılı : Kağızman / 1983
E-posta : fevzidas@gmail.com

Eğitim Bilgileri

Doktora (2013-2020) : Eskişehir Teknik Üniversitesi, Fen Bilimleri Enstitüsü,
Uzaktan Algılama ve Coğrafi Bilgi Sistemleri
Yüksek Lisans (2010-2012) : Gazi Üniversitesi, Bilişim Enstitüsü, Bilgisayar Eğitimi
Lisans (2003-2008) : Başkent Üniversitesi, Eğitim Fakültesi, Bilgisayar ve
Öğretim Teknolojileri Öğretmenliği (Tam Burslu) / (Bölüm
Birincisi)
Lisans (2005-2008) : Başkent Üniversitesi, Eğitim Fakültesi, İlköğretim
Matematik Öğretmenliği (Çift Anadal)

Mesleki Deneyim

2010 - : Çankırı Karatekin Üniversitesi Meslek Yüksekokulu,
Öğretim Görevlisi
2017 – 2019 : Çankırı Karatekin Üniversitesi Bilgi İşlem Daire
Başkanlığı, Ağ ve Sistem Birimi
2015 – 2018 : Çankırı Karatekin Üniversitesi Uzaktan Eğitim Uygulama
ve Araştırma Merkezi Müdür Yardımcısı
2010 - 2011 : Gazi Üniversitesi, Teknik Eğitim Fakültesi, Yarı Zamanlı
Öğretim Elemanı
2009 - 2011 : AS Adam Bilgi Teknolojileri, Kurucu Ortak
2007 - 2010 : Özel Baştem Okulları, Bilgisayar Öğretmeni, Bilişim
Teknolojileri Alanı Bölüm Başkanı, Matematik Öğretmeni
2007 - 2007 : Özel Bil-Koop Bilgisayar Kursu- Çözüm Eğitim Kurumları,
Yarı Zamanlı Bilgisayar Eğitmeni
2009 - 2010 : Ankara Ece Eğitim Kurumları, Yarı Zamanlı Matematik
Öğretmeni
2006 - 2007 : Kaynak Dil ve Bilgisayar Eğitim Merkezi, Yarı Zamanlı
Bilgisayar Eğitmeni

Akademik ve Bilimsel Çalışmalar

Uluslararası Konferanslar

- Daş, F., Demiröz, A., Ersöz, S., “Eğitim Kurumlarında IPV4’ten IPV6’ya Geçiş Süreci için Web Tabanlı Eğitim Yazılımı”, 6. Uluslararası Bilgisayar ve Öğretim Teknolojileri Sempozyumu, Gaziantep Türkiye 2012.
- Ersöz, S., Demiröz, A., Daş, F., “Meslek Yüksekokullarında Wiki ile İşbirlikçi Öğretim Uygulaması”, 6. Uluslararası Bilgisayar ve Öğretim Teknolojileri Sempozyumu, Gaziantep Türkiye 2012.
- Ünal, S., Daş, F. & Gulbahar, Y., “Analysis Of Traditional Education Environment’s Ability To Carry To Web Based Environment”, 8 th International Educational Technology Conference, Eskişehir, Turkey 2007.

Ulusal Konferanslar

- Bülbül, H.İ., Daş F. (2013) “Sanal Sunucularda Yedeklemenin Kolaylaştırılması”, Gazi Üniversitesi Fen Bilimleri Dergisi”, Cilt 1, Sayı 1. pp. 9-16. Ankara.
- Daş, F., Ersöz, S., “Uzaktan Eğitimde İçerik Geliştirme İçin Açık Kaynak Yazılımların Kullanımı”, Akademik Bilişim 2013, Antalya Türkiye 2013.

Ödüller / Başarılar

- Başkent Üniversitesi Bilgisayar ve Öğretim Teknolojileri Öğretmenliği Programı 2008 Yılı Bölüm Birinciliği
- Sokullu Mehmet Paşa Lisesi (Okul Birinciliği – 2003)

Mesleki Birlik/Dernek/Kuruluş Üyelikleri

- Linux Kullanıcıları Derneği