
GAZİ ÜNİVERSİTESİ
Gazi
1926

# ANLAMSAL BÖLÜTLEME İLE PİKSEL SEVİYESİNDE ANOMALİ TESPİTİNDE DAĞILIM DIŞI VE DEĞİŞEN BAĞLAMLARDA FARKLI VERİ SETİ KULLANIMI

**Mahmut Ferit YILDIZ**

**YÜKSEK LİSANS TEZİ**
**ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANA BİLİM DALI**

**GAZİ ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

**ŞUBAT 2025**

## ETİK BEYAN

Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Mahmut Ferit YILDIZ

25/02/2025

# ANLAMSAL BÖLÜTLEME İLE PİKSEL SEVİYESİNDE ANOMALİ TESPİTİNDE DAĞILIM DIŞI VE DEĞİŞEN BAĞLAMLARDA FARKLI VERİ SETİ KULLANIMI

(Yüksek Lisans Tezi)

Mahmut Ferit YILDIZ

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Şubat 2025

## ÖZET

Görüntülerde piksel seviyesinde anomali tespiti anlamsal bölütleme yöntemleri için önemli bir problemdir. Güncel makine öğrenim modelleri deney ortamında test verileri ile çok iyi sonuçlar verebilir. Fakat bu modeller test verilerinin gerçek dünya verileri ile aynı dağılımdan alındığını varsayarak çalışmaktadır. Dağılıma ait olmayan görsel nesneler üzerinde anomali içeren piksellerde bu modellerin düşük performansı, yapay görüntüler içeren veri setleri ile modeli yeniden eğitmeyi esas alan yöntemlerin kullanılmasını gerektirmektedir. Kullanılan yöntemler iyi sonuç verse bile yeniden eğitimde şu sorunlar ortaya çıkmaktadır: 1-Yeniden eğitim esnasında dağılıma ait olmayan örneklerde anomali tespitinin doğruluk oranı azalabilir, bu durumda model; eğitim öncesi doğru gelen piksellerde aykırı değer olarak algıladığı bir piksel değerini eğitim sonrası hatalı gelen bir pikselin aykırı değeri olarak algılayabilir. 2-Modeli yeniden eğitmeden dışarıdan bir veri seti ile anomali tespitinde, hatalı gelen pikseller için aykırı değer belirsizliği maksimize edilirken doğru gelen örneklerdeki aykırı değer piksellerinin anlamsal bölütleme oranı azalabilir. Bu çalışmada, yukarıda bahsedilen yöntemleri kullanarak modellerin farklı veri setlerinde anomali tespit performansını değerlendirmek için geliştirilen yöntem sunulmaktadır. Yöntem; bir anlamsal bölütleme modelini bir veri seti üzerinde öğrendikten sonra bu modeli farklı bir veri setinde sentetik olarak oluşturulan anomaliler üzerinde eğitmeye dayanmaktadır. Bu şekilde oluşturulan modelin değerlendirilmesi farklı anomali veri setleri üzerinde gerçekleştirilmiştir. Sonuçlar göstermiştir ki anomali tespiti yapan yöntemler eğitim veri setine yüksek oranda bağımlıdırlar. Sonuçlar; Cityscapes, KITTI, Fishyscapes gibi pek çok veri seti üzerinde yapılan deneylerle gösterilmiştir.

| | | |
|---|---|---|
| Bilim Kodu | : | 93437 |
| Anahtar Kelimeler | : | Anomali tespiti, anlamsal bölütleme, veri seti, makine öğrenmesi |
| Sayfa Adedi | : | 73 |
| Danışman | : | Doç. Dr. Ali SAYGIN |
| İkinci Danışman | : | Dr. Öğr. Üyesi Özgür ERKENT |

# USING DIFFERENT DATA SETS IN OUT-OF-DISTRIBUTION DATA AND VARYING CONTEXTS FOR PIXEL-LEVEL ANOMALY DETECTION WITH SEMANTIC SEGMENTATION

(M. Sc. Thesis)

Mahmut Ferit YILDIZ

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

February 2025

## ABSTRACT

Anomaly detection at the pixel level in images is an important problem for semantic segmentation methods. Current machine learning models can yield very good results with test data in experimental environments. However, these models operate under the assumption that the test data is drawn from the same distribution as real-world data. The low performance of these models on visual objects that do not belong to the distribution, particularly on pixels containing anomalies, necessitates the use of methods that focus on retraining the model with datasets containing synthetic images. Even if the methods used yield good results, the following issues arise during retraining: 1- The accuracy of anomaly detection on examples that do not belong to the distribution may decrease during retraining, leading the model to misinterpret a pixel value that was previously deemed correct as an outlier after training when it encounters a pixel that is incorrect. 2- When detecting anomalies with an external dataset without retraining the model, the uncertainty of outlier detection for incorrectly classified pixels is maximized, while the rate of semantic segmentation for outlier pixels in correctly classified examples may decrease. This study presents a method developed to evaluate the anomaly detection performance of models across different datasets using the aforementioned methods. The method is based on training a semantic segmentation model on one dataset and then retraining this model on synthetically created anomalies in a different dataset. The evaluation of the model created in this way has been conducted on various anomaly datasets. The results have shown that anomaly detection methods are highly dependent on the training dataset. These results have been demonstrated through experiments conducted on many datasets, including Cityscapes, KITTI, and Fishyscapes.

Science Code : 93437

Key Words : Anomaly detection, semantic segmentation, dataset, machine learning

Page Number : 73

Supervisor : Assoc. Prof. Dr. Ali SAYGIN

Co-Supervisor : Asst. Prof. Dr. Özgür ERKENT

# TEŞEKKÜR

Yüksek lisans çalışmalarım süresince, büyük destekte bulunan, kritik noktalarda bana doğru yolu gösterip ufkumu açan ve elde ettiğim başarılarda büyük pay sahibi olan danışman hocam Doç. Dr. Ali SAYGIN'a ve eş danışmanım Dr. Öğr. Üyesi Özgür ERKENT'e; ihtiyacım olduğu her an bana bilgi ve fikirlerini esirgemeyen başta tez izleme komitemde yer alan değerli hocalarıma; yoğun çalışmalarım boyunca bana sabır ve anlayış gösteren, maddi manevi her zaman yanımda olarak yükümü hafifleten ve her daim bana moral ve motivasyon kaynağı olan sevgili eşim Zeynep Leyla YILDIZ'a ve her zaman yanımda olduklarını hissettiren ve bu günlere gelmem de büyük emeği olan anne, babam ve aile bireylerim olmak üzere tüm aileme gönülden sonsuz teşekkür ederim.

# İÇİNDEKİLER

**Sayfa**

ÖZET ..... iv

ABSTRACT ..... v

TEŞEKKÜR ..... vi

İÇİNDEKİLER ..... vii

ÇİZELGELERİN LİSTESİ ..... ix

ŞEKİLLERİN LİSTESİ ..... x

SİMGELER VE KISALTMALAR ..... xii

1. GİRİŞ ..... 1

2. BİLGİSAYARLI GÖRÜ VE GİRDİ OLARAK GÖRÜNTÜNÜN TANIMI ..... 5

2.1. Veri Ön işleme Yöntemleri ..... 6

2.2. Görüntü Veri Setleri ..... 7

3. GÖRÜNTÜ BÖLÜTLEME ..... 11

3.1. Anlamsal Bölütleme ..... 11

3.2. Örneksel Bölütleme ..... 13

3.3. Panoptik Bölütleme ..... 13

4. ANLAMSAL BÖLÜTLEME ..... 15

4.1. Anlamsal Bölütleme Derin Öğrenme Modeli Mimarisi ..... 15

4.1.1. Kodlayıcı mimarisi ..... 21

4.1.2. Kod çözücü mimarisi ..... 22

4.2. Derin Öğrenme Modeli Bilgisi ..... 22

4.2.1. Tam evrişimli ağ ..... 22

4.2.2. Artık örüntü öğrenme modeli ..... 23

4.2.3. Anlamsal bölütleme modeli yeniden eğitimi ..... 23

4.4. Anlamsal Bölütleme ile Piksel Seviyesinde Anomali Tespiti ..... 24

**Sayfa**

4.4.1. Anlamsal bölütleme bağlamında anomali tespiti ..... 24
4.4.2. Anlamsal bölütleme bağlamında piksel seviyesinde anomali tespiti ..... 25
5. MATERYAL VE YÖNTEMLER ..... 27
5.1. Derin Öğrenme ..... 27
5.1.1. Denetimli makine öğrenimi ..... 28
5.1.2. Denetimsiz makine öğrenimi ..... 29
5.2. Derin Öğrenme Modeli Eğitim Süreci ..... 30
5.2.1. Veri toplama ..... 31
5.2.2. Ön işleme ..... 31
5.2.3. Veri hazırlama ..... 32
5.2.4. Model eğitimi ..... 32
5.2.5. Değerlendirme ..... 33
5.2.6. Hiper parametre ayarı ..... 36
5.3. Dağılım Dışı Veri ..... 36
5.3.1. Uygulamalarda dağılım dışı verilerin oluşma nedenleri ..... 36
5.3.2. Dağılım dışı verileri tespit etme ..... 38
5.4. Artık Örüntü Öğrenme Yöntemi ..... 39
5.5. DeepLabv3+ Modeli ..... 41
5.6. Pytorch ile Anlamsal Bölütleme ..... 42
6. SONUÇ ve ÖNERİLER ..... 59
KAYNAKLAR ..... 69
ÖZGEÇMİŞ ..... 73

# ÇİZELGELERİN LİSTESİ

**Çizelge** **Sayfa**

Çizelge 5.1. Denetimli ve denetimsiz derin öğrenme yöntemleri..... 27

Çizelge 6.1. Test sonuçları tablosu FPR metrik değerleri ..... 62

Çizelge 6.2. Test sonuçları tablosu AuROC metrik değerleri ..... 64

Çizelge 6.3. Test sonuçları tablosu AuPRC metrik değerleri ..... 66

# ŞEKİLLERİN LİSTESİ

**Şekil** **Sayfa**

Şekil 2.1. “m×n” boyutlarındaki gri tonlamalı görüntü I matrisi ..... 5

Şekil 2.2. “m×n” boyutlarındaki 3D tensör ..... 5

Şekil 3.1. Anlamsal bölütleme öncesi ..... 11

Şekil 3.2. Anlamsal bölütleme sonrası ..... 12

Şekil 3.3. Anlamsal bölütleme öncesi ..... 12

Şekil 3.4. Anlamsal bölütleme sonrası ..... 12

Şekil 3.5. Örneksel bölütleme sonrası ..... 13

Şekil 3.6. Panoptik bölütleme öncesi ..... 13

Şekil 3.7. Panoptik bölütleme sonrası ..... 14

Şekil 4.1. Evrişimsel sinir ağı katmanları ..... 16

Şekil 4.2. ReLu atkivasyon fonksiyonu çıktısı ..... 18

Şekil 4.3. Maksimum havuzlama yöntemi ..... 18

Şekil 4.4. Düzleştirme katmanı ..... 19

Şekil 4.5. Evrişim, havuzlama ve tam bağlantılı katmanlar ..... 20

Şekil 4.6. Evrişimsel sinir ağı mimarisi ..... 20

Şekil 4.7. Evrişimsel sinir ağı tüm mimarisi ..... 21

Şekil 5.1. Derin öğrenmenin kapsamı ..... 27

Şekil 5.2. Denetimli öğrenme çalışma prensibi ..... 29

Şekil 5.3. Denetimsiz öğrenme çalışma prensibi ..... 30

Şekil 5.4. Derin öğrenme modeli eğitim süreci ..... 31

Şekil 5.5. Dağılım dışı verileri tespit etme süreci ..... 39

Şekil 5.6. Akış diyagramı – 1 ..... 51

Şekil 5.7. Akış diyagramı – 2 ..... 52

Şekil 5.8. Akış diyagramı – 3 ..... 53

Şekil 5.9. Akış diyagramı – 4 ..... 54

**Şekil** **Sayfa**

Şekil 5.10. Akış diyagramı – 5..... 55

Şekil 6.1. FPR analizi (40 Epok) ..... 63

Şekil 6.2. FPR analizi (80 Epok) ..... 63

Şekil 6.3. AuROC analizi (40 Epok) ..... 64

Şekil 6.4. AuROC analizi (80 Epok) ..... 65

Şekil 6.5. AuPRC analizi (80 Epok) ..... 66

Şekil 6.6. AuPRC analizi (80 Epok) ..... 67

# SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

| **Simgeler** | **Açıklamalar** |
|---|---|
| **db** | Desibel |
| **hz** | Hertz |
| **$m^2$** | Metrekare |
| **$m^3$** | Metreküp |

| **Kısaltmalar** | **Açıklamalar** |
|---|---|
| **ASPP** | Atrous Spatial Pyramid Pooling |
| **CNN** | Convolutional Neural Network |
| **COCO** | Common Objects in Context |
| **FCN** | Fully Connected Network |
| **GPU** | Graphical Processing Unit |
| **KITTI** | Karlsruhe Institute of Technology and Toyota Technological Institute |
| **OE** | Outlier Exposure |
| **OOD** | Out-of- Distribution |
| **PASCAL VOC** | PASCAL Visual Object Classes |
| **ReLU** | Rectified Linear Unit |
| **SOTA** | State of The Art |

# 1. GİRİŞ

Anlamsal bölütleme, bir görüntüdeki her piksele bir sınıf etiketi atamayı amaçlayan temel bilgisayarlı görü işlemidir. Eğitim veri setinden önemli ölçüde farklılık gösteren örnekleri ifade eden dağılım dışı (OOD) veriler bağlamında, anlamsal bölütleme benzersiz zorluklarla karşı karşıya kalmaktadır. OOD verileri, aydınlatma koşullarındaki değişimler, nesne görünümlerindeki farklılıklar veya eğitim setinde temsil edilmeyen tamamen yeni nesne sınıfları gibi çeşitli kaynaklardan ortaya çıkabilmektedir. Geleneksel bölütleme modelleri genellikle OOD örneklerinin varlığında genelleme yapmakta zorlanmakta, bu da performansın düşmesine ve güvenilir olmayan tahminlere yol açmaktadır.

Derin öğrenmedeki son gelişmeler, bilgisayarlı görme alanında önemli iyileştirmeler göstermiştir. Sinir ağları, önceki yöntemlere kıyasla yüksek doğrulukları nedeniyle sınıflandırma, nesne tespiti ve anlamsal segmentasyon için fiili yöntem haline gelmiştir [1]. Derin öğrenme tabanlı tam bağlı ağlar anlamsal bölütleme için etkili bir yöntemdir [2]. Ancak, bu ağların tahminleri oldukça doğru olsa da, genellikle anormal girdilerle (yani ağın eğitim dağılımının dışındaki örneklerle) karşılaştıklarında başarısız olmaktadır.

Mevcut anlamsal bölütleme modellerinin anormallik örneklerini yerelleştirmedeki yetersizliği bu modellerin güvenlik açısından kritik senaryolarda konuşlandırılmasını engellemektedir. Gerçek dünyadaki bir çok uygulamanın, kapalı set karşılaştırmalardaki başarısına rağmen, kapalı dünya bölütleme modellerinin açık dünya senaryolarında görüntüde yanlışlıkla dağılım dışı anomali tahminleri yaptığında bununla başa çıkma yeteneğine sahip olması gerekmektedir. Kapalı set anlamsal bölütleme modelleri, eğitim verileriyle gerçek dünya verilerinde aynı sınıfların var olduğu varsayımı ile çalışmaktadır. Örnek vermek gerekirse, bir şehir görüntüleme veri setinde yayalar ve trafik işaretleri genellikle daha fazla temsil edilirken, ağaçlar veya otların sınıflandırılması için yeterli veri olmayabilmektedir. Bu durumda, anlamsal bölütleme modeli ağaç gibi daha az görülen nesneleri sınıflandıramamaktadır ve bölütleme sonuçları bu nesneler için kötüleşmektedir. Gerçek dünya koşulları eğitim senaryolarından çok farklı olabilmektedir. Örneğin, bir model bir şehrin belirli yollarını içeren veri seti ile eğitildikten sonra başka bir şehirdeki yolları veya taşradaki yolları içeren veri setleri üzerinde test edildiğinde benzer performans gösterememektedir. Değişen içeriklerde farklı renkler, farklı işaretler, farklı sınıflar olacaktır. Bu tarz durumlarda modeller hatalı tahminlerde bulunmaktadır.

Bu tür hataları en aza indirmek amacıyla; dağılım içi bölütleme modeli ile çalışabilen piksel seviyesinde dağılım dışı tespit tekniklerini geliştirebilmek için kapalı set bölütleme maskesinin üzerine dağılım dışı anomali sınıfı tespit eden harita oluşturulmalıdır. En güncel teknoloji olan dağılım dışı piksel detektörleri, dağılım dışı veri setleri ile dağılım içi aykırı değer bölütleme modellerini yeniden eğitme prensibine dayanmaktadır. Böyle bir dağılım dışı veri seti, dağılım içerisinde belirlenen sınıf değerlerinden farklı olan aykırı değerlerin orijinal verilere enjekte edilmesi ile elde edilmektedir; yani anomali olan nesneleri eğitim görüntüleriyle sentetik olarak karıştırmayı içermektedir. Ardından, kapalı set modeli, aykırı değerlerin orijinal verilere enjekte edilen piksellerini optimize etmek için entropi maksimize etme veya geri çekilme öğrenimi ile yeniden eğitilmektedir. Bu yeniden eğitim, piksel seviyesinde dağılım dışı verilerdeki tespit doğruluk oranını artırır, ancak dağılım içinde bölütleme doğruluk oranını azaltabilmektedir. Örneğin, "çalı" veya "kaldırım" sınıfları gibi dağılım içinde bulunma oranı az olan kategorilerdeki pikseller, "yol" gibi çoğunluk sınıfları olarak yanlış sınıflandırılabilmektedir. Dağılım dışı verilerde anomalileri tespit etme yeteneğinin, kapalı set bölütleme doğruluk oranını azaltmayacak ya da en minimum seviyede etkileyebilecek şekilde sağlanması gerekmektedir.

Dağılım dışı piksel detektörlerini etkileyen başka bir önemli problem ise dar bağlam güvenilirliğidir. Şehir ve kırsal alanlar gibi bağlamlar, sahnenin çevresini temsil eden dağılım içi pikselleri tarafından tanımlanmaktadır. Bunun sebebi piksel bazında anomali puanlarını, entropi veya enerji tabanlı iyileştirme algoritmalarının bağımsız olarak eğitmesi, ancak dağılım dışı veriler ve bağlamlar arasındaki ilişkileri hesaba katmamasıdır. Bu durum dağılım içi aykırı değer belirsizliğinin olması gerekenden fazla bir şekilde minimize edilmesine sebep olmaktadır. Ancak dağılım dışı aykırı değer piksellerinin tespit oranını ise zayıf bir şekilde maksimize edilmesine neden olmaktadır. Böylece anomali detektörü, eğitim esnasında karşılaşmadığı bağlamlarda test edildiğinde bilinmedik desene sahip dağılım içi aykırı değere sahip pikselleri sanki dağılım dışı aykırı değer pikselleri olarak sınıflandırabilmektedir. Buna ek olarak, anomali detektörünün bu durumu küçük anomali skoruna sahip olan dağılım dışı pikselleri dağılım içi aykırı değer pikselleri olarak yanlış bir şekilde sınıflandırabilmektedir.

Örneğin, en ileri teknoloji olan “Artık Örüntü Öğrenme” yöntemi gibi yöntemler; geliştirilen derin öğrenme modelinin şehirden kırsala geçiş gibi içerik değişimlerinde eğitime ihtiyaç duymadan dağılım dışı anomali tespitinde başarılı olacağını iddia

etmektedir.  Artık örüntü öğrenme yönteminde model, anormalliklerin artış kalıplarını ara özellikler temelinde öğrenmek üzere eğitilmiştir ve dondurulmuş anlamsal bölütleme sınıflandırıcısının muhtemel anormal bölgeler için yüksek belirsizlik üretmesini sağlamaktadır. Anlamsal bölütleme modelini yeniden eğitmemesi sayesinde bu modelin, dağılım dışı verilerde piksel seviyesinde anomali tespit ederken yüksek doğruluk sağladığı belirtilmektedir.

Bu çalışmada, yukarıda belirtilen içerik değişimlerinde, modelin tekrar eğitilmeden mevcut hali kullanılarak anlamsal bölütleme ile anomali tespitini yüksek doğrulukta vermesi incelenip bu durumda modelin anomali tespit performansı farklı farklı veri setleri ile değerlendirilmektedir. Bir anlamsal bölütleme modeli bir veri seti üzerinde anomali tespitini öğrendikten sonra, aynı model sentetik olarak oluşturulmuş anomalilerden oluşan başka bir veri seti üzerinde yeniden eğitilmektedir. Bu şekilde eğitilen modeli değerlendirmek için farklı veri setleri üzerinde deneyler yapıldı. Özetlemek gerekirse, bu tezde sunulan katkılar şunlardır:

- Sentetik verilerle eğitilmiş olsa bile modelin içerik değişiminde dağılım dışı aykırı değer veri setlerine bağlı olduğunu göstermek
- Sentetik olarak oluşturulan veri seti ile eğitilen modeli farklı farklı gerçek dünya veri setleri ile test etmek

# 2. BİLGİSAYARLI GÖRÜ VE GİRDİ OLARAK GÖRÜNTÜNÜN TANIMI

Bilgisayarlı görü (computer vision), bilgisayarların ve makinelerin görsel verileri anlamalarını ve bu veriler üzerinden karar vermelerini sağlayan bir alandır. Bu bağlamda görüntü, bilgisayarlı görünün temel girdi birimidir. Bir görüntü; algoritmaların işleyebileceği sayısal bir formata dönüştürülmektedir. Böylece bilgisayarlı görü için girdi verisi olarak hazır hale getirilmektedir. Matematiksel olarak bir görüntü "m×n" boyutlarında bir matris (I) olarak gösterilebilir; burada "m" yükseklik, "n" genişliktir ve her bir "I(i,j)" elemanı "(i,j)" koordinatlarındaki piksel değerine karşılık gelmektedir. Her piksel, genellikle renk, yoğunluk, doku ve konum olmak üzere birden fazla özellik ile tanımlanmaktadır. Renkli görüntüler RGB veya HSV gibi çeşitli renk uzaylarında temsil edilebilmektedir.

Bir görüntü, gri tonlamalı görüntüler için 2 boyutlu bir piksel ızgarası veya renkli görüntüler için 3 boyutlu bir dizi olarak temsil edilmektedir. "m×n" boyutundaki gri tonlamalı bir görüntü I matrisi olarak gösterilmektedir:

$$I = \begin{bmatrix} I(1,1) & I(1,2) & \cdots & I(1,n) \\ I(2,1) & I(2,2) & \cdots & I(2,n) \\ \vdots & & \ddots & \vdots \\ I(m,1) & I(m,2) & \cdots & I(m,n) \end{bmatrix}$$

Şekil 2.1. "m×n" boyutlarındaki gri tonlamalı görüntü I matrisi

Renkli bir görüntü, tipik olarak RGB renk modeli kullanılarak bir 3D tensör olarak temsil edilmektedir:

$$I = \begin{bmatrix} R(1,1) & G(1,2) & \cdots & B(1,n) \\ R(2,1) & G(2,2) & \cdots & B(1,2) \\ \vdots & & \ddots & \vdots \\ R(m,n) & G(m,n) & \cdots & B(m,n) \end{bmatrix}$$

Şekil 2.2. "m×n" boyutlarındaki 3D tensör

## 2.1. Veri Ön işleme Yöntemleri

Görüntü verilerini kullanmadan önce aşağıda tanıtılan birkaç ön işleme adımı uygulanabilmektedir:

Yeniden boyutlandırma: Giriş verileri arasında tekdüzelik sağlamak için görüntüler genellikle standart bir boyuta yeniden boyutlandırılmaktadır:

$$I' = Resize(I, w, h) \tag{2.1}$$

burada "w" istenilen genişlik ve "h" istenilen yüksekliktir.

Normalleştirme: Piksel değerleri [0, 1] veya [-1, 1] gibi model için uygun bir aralığa normalleştirilebilmektedir:

[0,1] aralığındaki piksel değerleri için,

$$I' = \frac{I}{255} \tag{2.2}$$

denklemi kullanılır.

[-1,1] aralığındaki piksel değerleri için,

$$I' = \frac{I-128}{128} \tag{2.3}$$

denklemi kullanılır.

Veri büyütme: Veri artırımı ya da veri büyütme, aslında daha fazla veri toplamadan eğitim veri kümesinin çeşitliliğini artırmada önemli bir rol oynamaktadır. Bu teknik, modeli daha çeşitli girdi senaryolarına maruz bırakarak aşırı uyumu önlemeye yardımcı olmaktadır. Yaygın artırma teknikleri şunları içerir:

Döndürme: Görüntünün belirli bir açıyla döndürülmesi. Bu teknik, modelin görüntülerdeki nesnelerin yönüne karşı değişmez olmasına yardımcı olmaktadır. Örnek olarak; bir kedi görüntüsü 90, 180 veya 270 derece döndürülebilmektedir.

Çevirme: Görüntünün yatay veya dikey olarak çevrilmesidir. Bu, aynı nesnenin farklı bakış açılarını ve konfigürasyonlarını simüle etmektedir. Örnek olarak modelin her iki yönden de özellikleri öğrenmesine yardımcı olan bir arabanın yatay olarak çevrilmiş görüntüsü verilebilmektedir.

Kırpma: Görüntünün bölümlerini rastgele kırpmak, modelin nesneleri farklı ölçeklerden ve perspektiflerden tanımayı öğrenmesine yardımcı olmaktadır. Örnek olarak bir portrede sadece yüze odaklanmak için kırpma yapmak, modele farklı ayarlardan bağımsız olarak yüzleri tanımlamayı öğretebilmektedir.

Ölçeklendirme: En boy oranını koruyarak görüntünün boyutunu değiştirmek, modelin nesne boyutundaki değişikliklere karşı dayanıklı olmasına yardımcı olabilmektedir.

Parlaklık ve kontrast ayarları: Görüntülerin aydınlatma koşullarının değiştirilmesi, modelin çeşitli aydınlatma ortamlarındaki nesneleri tanımayı öğrenmesine yardımcı olabilmektedir.

Bu tezde görüntü verileri derin öğrenme modeli için girdi olarak kullanılmıştır. Derin öğrenme modelleri, özellikle de Evrişimsel Sinir Ağları (CNN'ler) için görüntü verileri genellikle yeniden şekillendirilip ve düzenlenmektedir. Bir grup görüntü, bir evrişimsel sinir ağının girdi şekli olarak aşağıdaki parametreler ile gösterilmektedir (N,H,W,C):

N: Toplu işteki görüntü sayısı
H: Her görüntünün yüksekliği
W: Her görüntünün genişliği
C: Kanal sayısı (gri tonlama için 1, RGB için 3)

### 2.2. Görüntü Veri Setleri

Veri setleri veya veri kümeleri onlarca farklı sınıftan ve her sınıf için binlerce ek açıklamadan oluşmaktadırlar. Çok sayıda etiket, veri kümesi içindeki çeşitliliği artırmaktadır ve modelin daha iyi öğrenmesine yardımcı olmaktadır. Nesne şekline,

rengine ve yönüne duyarlı oldukları için çeşitli verilere sahip olmak bölütleme modelleri için önemlidir. Bu tezde kullanılan veri setleri COCO (Common Objects in Context), CityScapes, KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute) ve FishyScapes'dir.

COCO: COCO veri kümesi yoğun, bağlamsal nesne anlama talepleriyle popüler bir veri setidir. Nesne algılama, segmentasyon ve altyazı görevlerini desteklemektedir. 2,5 milyondan fazla etiketli örneğe sahip 330 000'den fazla görüntü verisi vardır. 80 adet nesne sınıfı vardır. Ayrıca ek açıklamalar, nesneler için bölütleme maskeleri ve anahtar noktaları içermektedir.

CityScapes: Cityscapes veri kümesi, kentsel sokak sahnelerine odaklanan anlamsal bölütleme konusunda uzmanlaşmıştır. Genellikle otonom sürüş uygulamalarında kullanılmaktadır. 5 000 yüksek çözünürlüklü görüntü ve kaba etiketleme için ek 20 000 görüntü içerir. Ek açıklamalar, değerlendirme için etiketlenmiş 19 sınıfla birlikte 30 sınıf içermektedir. Bu sınıflar, yol, kaldırım, bina, araba gibi temel kentsel sınıflardır.

KITTI: Bu veri seti özellikle otonom araçlar için geliştirilmiş, çok çeşitli görsel algılama görevleri için zengin bir veri setidir. Hem şehir içi hem de kırsal ortamlarda çekilmiş görüntüler içerir ve gerçek zamanlı bilgisayarlı görüş uygulamalarını destekleyecek şekilde tasarlanmıştır. 5 000 yüksek çözünürlüklü (1280x384 piksel) görüntü verisi vardır. Ek olarak 20 000 görüntü, kaba etiketleme için kullanılmaktadır. Veri seti, değerlendirme için etiketlenmiş 19 ana sınıf bulunmaktadır. Bu ana sınıflar, otomobiller, yaya, bisiklet, kamyon, otobüs, yol işaretleri, ağaçlar, binalar gibi yaygın olarak karşılaşılan nesnelerdir.

FishyScapes: FishyScapes kentsel sürüş veri seti, kentsel sürüş ortamlarındaki anlamsal segmentasyon görevlerinde belirsizlik tahmini ve anormallik tespitini değerlendirmek için tasarlanmış açık kaynak veri setidir. Genel olarak yaklaşık 10 000 görüntü içermektedir. Bu veri seti, otonom sürüş sistemlerinin güvenli ve etkili bir şekilde çalışabilmesi için önemli bir kaynak sunmaktadır. Araştırmacılara ve geliştiricilere ücretsiz erişim imkânı sağlayarak, kentsel sürüş ile ilgili çalışmalarda kullanılabilecek bir benchmark sağlamaktadır. Veri seti, gerçek dünyada karşılaşılabilecek anormalliklerin tespitine odaklanarak, teorik çalışmaları uygulamaya dönüştürmeyi amaçlamaktadır. Kentsel

ortamlardaki beklenmedik durumları ve nesneleri tanımlama yeteneğini geliştirmek için tasarlanmıştır. Anormallik tespiti, sürüş güvenliği için kritik öneme sahiptir.

# 3. GÖRÜNTÜ BÖLÜTLEME

Görüntü bölütleme, bilgisayarla görmede, tipik olarak bir görüntünün temsilini basitleştirmek veya analizi kolaylaştırmak için bir görüntüyü farklı segmentlere veya bölgelere ayırmayı içeren bir süreçtir. Her bölge genellikle görüntüdeki farklı nesnelere veya alanlara karşılık gelmektedir ve tanıma, sınıflandırma ve izleme gibi görevleri kolaylaştırmaktadır. Bu bölgelerin pikselleri genellikle belirli özellikleri paylaşmaktadır.

Görüntü bölütleme en yaygın olarak daha keskin nesne sınırlarını yakalamak için kullanılmaktadır. Genel olarak üç tür görüntü bölütleme tekniği vardır;

- Anlamsal Bölütleme
- Örneksel Bölütleme
- Panoptik Bölütleme

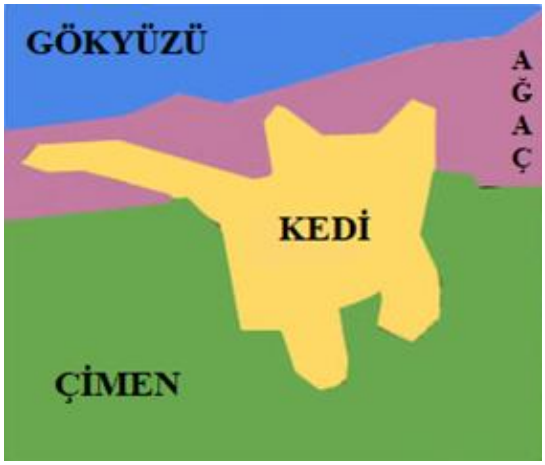
## 3.1. Anlamsal Bölütleme

Anlamsal bölütleme, basit olarak bir giriş görüntüsüne ait her bir piksel değerine bir sınıf etiketi atama işlemidir.



Şekil 3.1. Anlamsal bölütleme öncesi

Yukarıdaki resme anlamsal bölütleme uygulandığında aşağıdaki bir çıktı elde edilmektedir:



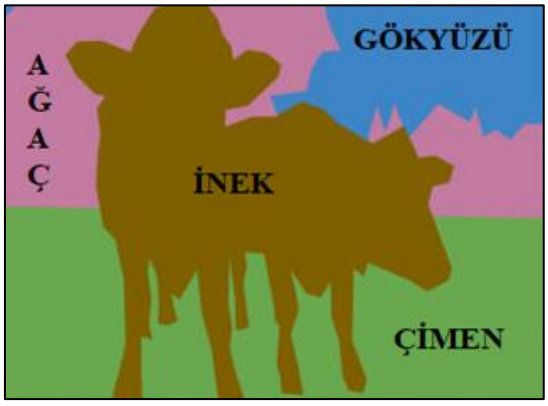

Şekil 3.2. Anlamsal bölütleme sonrası

Elde edilen resimde yeşil çimen sınıfını, sarı kedi sınıfını, mor ağaç sınıfını ve mavi gökyüzü sınıfını temsil etmektedir. Anlamsal bölütlemeyi örneksel bölütleme işleminden ayıran temel özellik, aynı sınıfa ait farklı nesneler arasında ayrım yapmamasıdır.



Şekil 3.3. Anlamsal bölütleme öncesi

Yukarıdaki resme anlamsal bölütleme uygulandığında aşağıdaki bir çıktı elde edilmektedir:




Şekil 3.4. Anlamsal bölütleme sonrası

Elde edilen resim içerisin birden fazla “inek” olmasında rağmen, “inek” sınıfına ait bir tür piksel kümesi elde edilmektedir. Burada anlamsal bölütleme ile ilgili temel bilgiler verilmiştir. Daha detaylı bilgili tezin ilerleyen kısmında verilecektir.

### 3.2. Örneksel Bölütleme

Örneksel bölütleme anlamsal bölütlemeye çok benzemektedir. İki teknik arasındaki fark; aynı sınıfa ait birden fazla nesne arasında örneksel bölütleme tekniğinin ayrım yapma kabiliyetidir.




Şekil 3.5. Örneksel bölütleme sonrası

Yukarıdaki resme örneksel bölütleme tekniği uygulandığında “araba” sınıfına ait nesnelerin farklı renkler ile sınıflandırıldığı görülebilmektedir.

### 3.3. Panoptik Bölütleme

Semantik ve örneksel bölütlemenin birleşimi olan panoptik bölütleme, aynı sınıfın farklı örnekleri arasında ayrım yaparken her pikselin ait olduğu sınıfı belirlemektedir.



Şekil 3.6. Panoptik bölütleme öncesi

Yukarıdaki resme panoptik bölütleme uygulanırsa aşağıdaki gibi bir çıktı elde edilmektedir:



Şekil 3.7. Panoptik bölütleme sonrası

Elde edilen resimde “köpek” sınıfı, “çimen” sınıfı ve “duvar” sınıfı ayrı ayrı belirtilirken, “köpek” sınıfına ait 2 ayrı nesne de farklı renkte sınıflandırılmıştır.

# 4. ANLAMSAL BÖLÜTLEME

Anlamsal bölütleme, bir görüntüdeki her pikselin belirli bir nesne veya alan kategoriyle sınıflandırılmasına odaklanan bir derin öğrenme uygulamasıdır. Başka bir deyişle, anlamsal bölütleme, her pikseli bir sınıfla etiketleyerek görüntüyü anlamaya çalışır. Örneğin, bir şehir manzarası olan görüntüde bulunan binaları, arabaları, ağaçları ve gökyüzünü ayırt edilebilir çıktı haline getirmektedir.

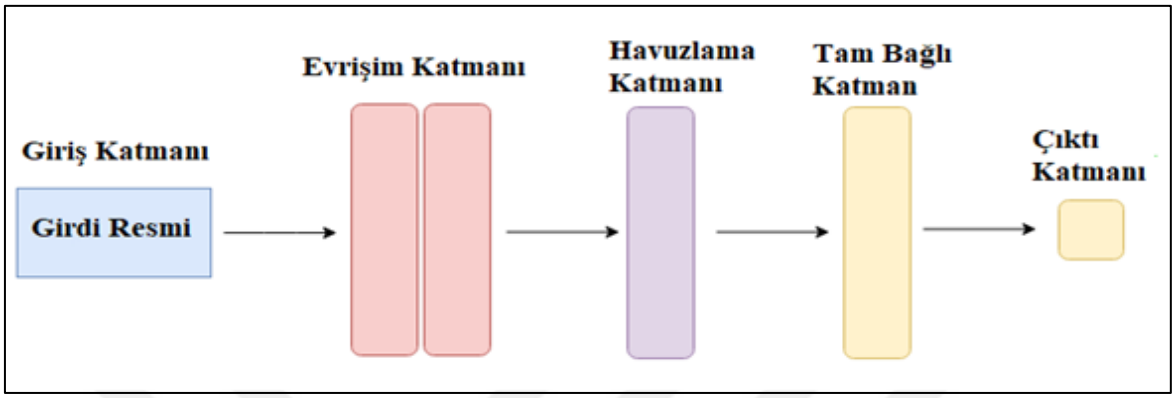
Anlamsal bölütleme yönteminde bir girdi görüntüsü $I \in \mathbb{R}^{HxWxC}$ (burada H yükseklik, W genişlik ve C kanal sayısıdır) olarak alınmaktadır ve her pikselin belirli bir sınıfa ait olması gerektiği varsayılmaktadır. Bu yöntemde amaç çıktılar için bölütleme haritası $S \in \mathbb{R}^{HxW}$ üretmektir; elde edilen her piksel için S(i,j) değeri, ilgili pikselin ait olduğu sınıfı (yani C∈{0,1,...,K−1} değerini) göstermektedir.

## 4.1. Anlamsal Bölütleme Derin Öğrenme Modeli Mimarisi

Semantik segmentasyon, derin öğrenme modelleri tarafından gerçekleştirilmektedir. Bu modeller genellikle evrişimsel sinir ağları (CNN) tabanlıdır. Evrişimsel sinir ağı (CNN) bilgisayarlı görü alanında yaygın olarak kullanılan bir çeşit derin öğrenme sinir ağı mimarisidir. Bir evrişimsel sinir ağında 3 tip katman vardır. Bu katmanlar;

- Giriş katmanı: Sinir ağına girdi verilen katmandır. Verilerimizdeki toplam özellik sayısı bu katmandaki nöron sayısını belirlemektedir. Eğer veri bir görüntü ise bu durumda nöron sayısı piksel sayısına eşit olmaktadır.
- Gizli katman: Bu katman giriş katmanından gelen girdiler ile beslenmektedir. Verinin boyutlarına ve seçilen modele göre sinir ağında birden fazla gizli katman bulunabilmektedir. Her gizli katmanın farklı sayıda nöronu olabilmektedir. Her gizli katmanda özellik sayısından daha fazla nöron bulunabilmektedir. Bir katmanın çıktısı; bir önceki katmandan gelen çıktının o katmana ait ağırlıkları ile matris çarpımı yapıldıktan sonra yanlılık eklenip hesaplanmaktadır. Daha sonra çıktı; aktivasyon fonksiyonuna iletilmektedir. Gizli katman içerisinde evrişimsel katman, havuzlama katmanı, tam bağlı katman gibi katmanlar bulunabilmektedir.

- Çıktı katmanı: Gizli katmanın çıktısı softmax veya sigmoid gibi aktivasyon fonksiyonlarından geçirilmektedir. Bu fonksiyonlar her bir sınıfa ait çıktıyı her sınıfa ait olasılık puanına dönüştürmektedir.




Şekil 4.1. Evrişimsel sinir ağı katmanları

Model veriler ile beslenmektedir. Her katmanın çıktısı yukarıda madde madde anlatıldığı gibi verilerin katmanlardan ilerletilmesi ile elde edilmektedir. Bu mantığa ileri besleme de denilmektedir. İleri besleme sonucunda elde edilen olasılık verisi hata fonksiyonuna iletilerek hata oranı hesaplanmaktadır. Çapraz entropi, kare kayıp hata fonksiyonu gibi yaygın kullanılan hata fonksiyonları vardır. Hata fonksiyonlarının çıktıları, evrişimsel sinir ağı modelinin ne kadar iyi performans gösterdiğini ölçmektedir. Hata fonksiyonunun çıktısı türevler ile hesaplanarak modele geri besleme yapılmaktadır. Bu yöntem geri yayılım olarak adlandırılmaktadır ve amacı modelin performansını arttırmak için hatayı aza indirmektir.

Katmanlar bir dizi olarak bir araya gelip tam bir evrişim sinir ağı mimarisi oluşturmaktadır. Her katman bir hacmi diğerine dönüştürmek için türevlenebilir fonksiyon kullanmaktadır.

Evrişimsel sinir ağının mimarisinin çalışma prensibi 32x32x3 boyutlarına sahip bir görüntü ile aşağıda anlatılmıştır:

Giriş katmanı

Giriş katmanı modele girdi verilen katmandır. Evrişimsel sinir ağlarında genel olarak resim ya da bir dizi resim girdi olarak verilmektedir. Bu katman, genişlik değeri 32, yükseklik değeri 32 ve derinlik değeri 3 olan örnek resmin ham verisini tutmaktadır.

Evrişimsel katmanlar

Bu katmanda özellik çıkarımı yapılır. Girdi veri setinden özellik çıkarımı için kullanılır. Bu katmanda öğrenilebilir bir dizi filtreler bulunmaktadır. Bu filtrelere çekirdekler de denmektedir ve bu katmanda girdi görüntülerine bu filtreler uygulanmaktadır. Filtreler genelde 2x2, 3x3 veya 5x5 şeklindeki küçük matrislerdir. Evrişimsel katman bu filtreleri girişteki görüntü verilerinin üzerinde gezdirmektedir ve çekirdek ağırlığına karşılık gelen girdi görüntü verisi arasında nokta çarpım hesabını yapmaktadır. Bu katmanın çıktısı özellik haritaları olarak adlandırılmaktadır. Örnekte verilen resim boyutları için bu katmanda toplamda 12 filtre kullanıldığı varsayılırsa, çıktı hacmi 32x32x12 boyutunda olacaktır.

Aktivasyon katmanı

Evrişimsel sinir ağlarında tüm katmanlar doğrusal bir fonksiyon olabilmektedir. Bu durumda sinir ağı; sonucu çıktıların doğrusal bir kombinasyonu olarak hesaplayabilmektedir. Bu durumu önlemek için aktivasyon katmanı eklenmiştir. Aktivasyon katmanının görevi, evrişimsel sinir ağında doğrusal olmama özelliği kazandırmaktır. Bir önceki katmanın çıktısına bir aktivasyon fonksiyonu ekleyerek bu görevi yapmaktadır. Evrişimsel katmanın çıktısına eleman seviyesinde bir aktivasyon fonksiyonu bu katmanda uygulanmaktadır. ReLu, Tanh ve Leaky ReLu yaygın olarak kullanılan aktivasyon fonksiyonlarıdır. Bu katmanda çıktının boyutları değişmez dolayısı ile bir önceki katmandaki boyutları olan 32x32x12 olacaktır.

Evrişimsel sinir ağı eğitiminde hız konusunda en iyi sonucu ReLu verdiği için yapay sinir ağlarında en çok bu fonksiyon tercih edilmektedir. Basitçe ReLu fonksiyonu $f(x) = \max(0,x)$ olarak ifade edilmektedir. Bu fonksiyon özellik haritasına eklendiğinde aşağıdaki gibi bir sonuç elde edilmektedir:

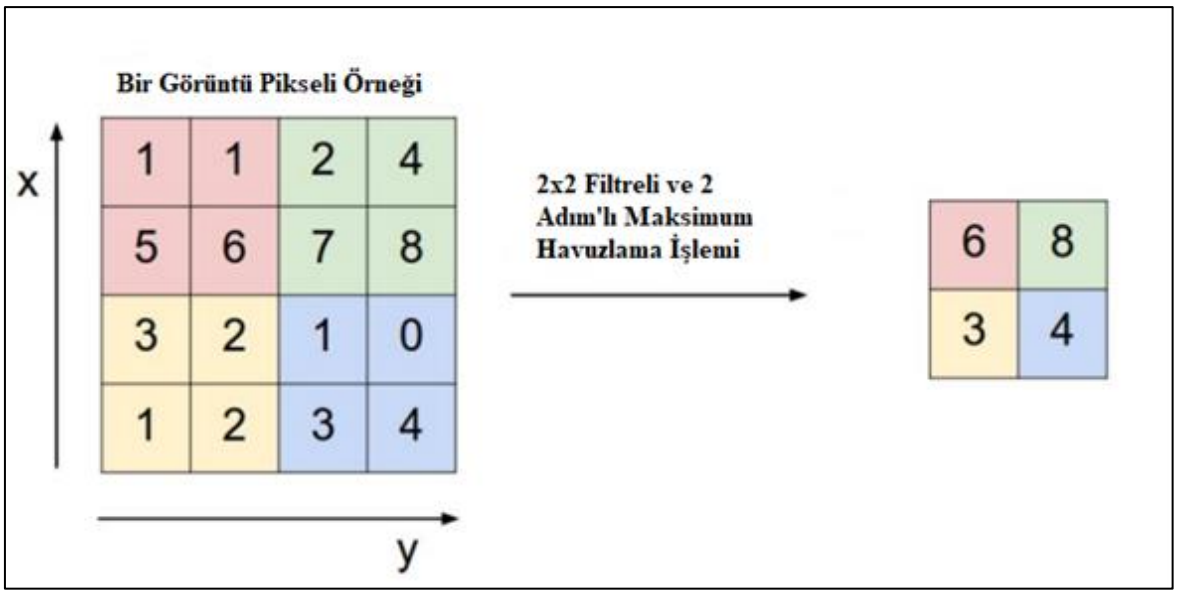



Şekil 4.2. ReLu atkivasyon fonksiyonu çıktısı [3]

Şekil 4.2'de özellik haritasında görülen siyah değerler negatif değerlerdir. ReLu fonksiyonu uygulandığında siyah değerler kaldırılıp yerine 0 değeri koyulmaktadır.

Havuzlama katmanı

Havuzlama katmanının ana işlevi hesaplamayı hızlandırarak bellek maliyetini azaltmak ve modelin aşırı uyum durumuna geçmesini engellemek için verinin boyutunu azaltmaktır. Havuzlama katmanı periyodik olarak evrişimsel sinir ağı mimarisinde sürece dâhil edilmektedir. Maksimum havuzlama ve ortalama havuzlama en yaygın kullanılan havuzlama katmanı türleridir. Evrişimsel sinir ağının havuzlama katmanında 2x2 filtreli ve 2 adımlı maksimum havuzlama kullanılırsa bu katman sonucunda çıktının boyutları 16x16x12 olacaktır.

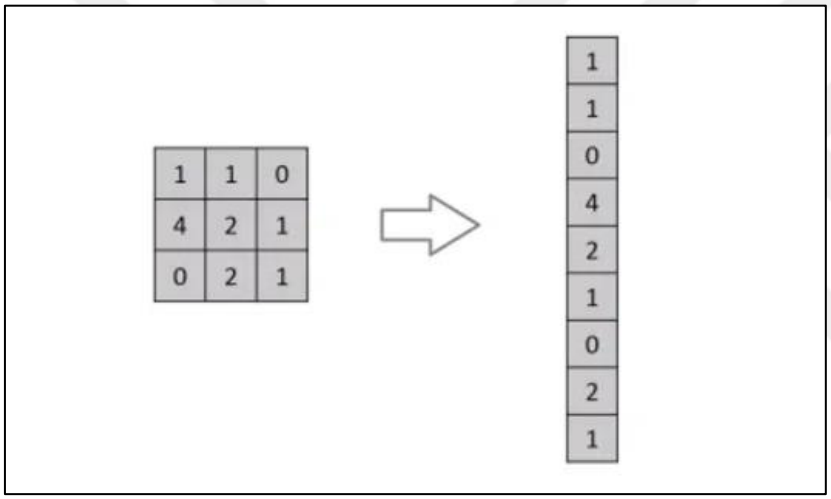



Şekil 4.3. Maksimum havuzlama yöntemi

Şekil 4.3'de görüldüğü gibi 2x2 filtre, kapsadığı alandaki en büyük sayıyı almaktadır. Böylece evrişimsel sinir ağı; doğru karar verebilmek için yeterli bilgiyi içeren daha küçük çıktıları kullanmaktadır.

Düzleştirme katmanı

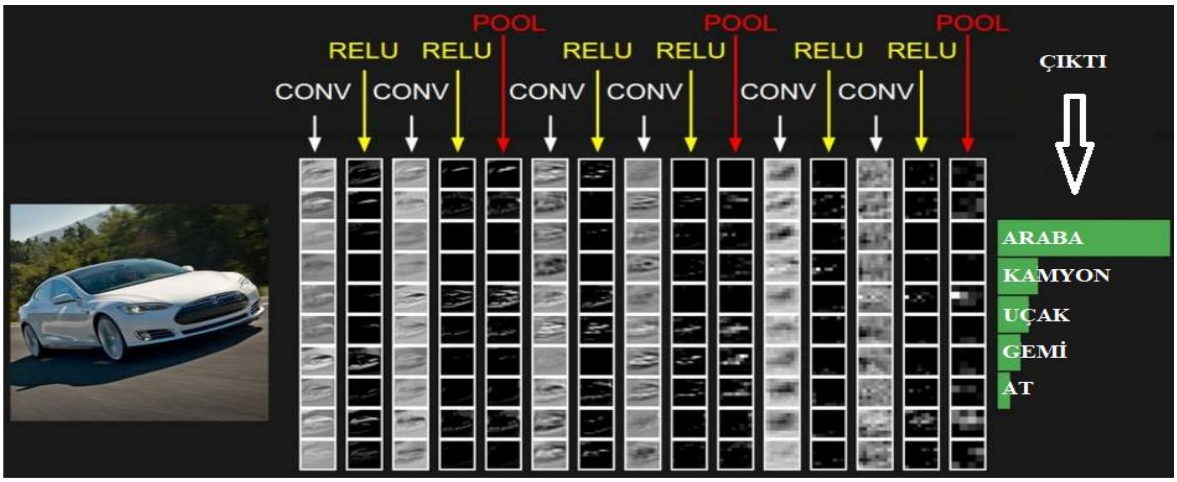
Bu katmanın görevi, en basit tabir ile en önemli katman olan tam bağlı katmana girdi sağlamaktır. Evrişim ve havuzlama katmanlarından elde edilen özellik haritaları tek boyutlu bir vektöre düzleştirilmektedir. Böylelikle elde edilen vektör sınıflandırma bölümü için tam bağlı katmana aktarılabilmektedir.




Şekil 4.4. Düzleştirme katmanı

Tam bağlı katmanlar

Tam bağlantılı katmanlar; düzleştirme katmanının çıktısını girdi olarak alır ve son sınıflandırmayı hesaplamaktadır. Tam bağlı katman; önceki katmanlarda öne çıkarılan yüksek düzeyli, soyut özellikleri kullanarak uygun biçimde kesin tahminler yapmaktadır. Tam bağlı katmanının birbirinden farklı bilgileri birleştirme yeteneği vardır. Verilerdeki kompleks desenleri ve karşılıklı ilişkileri bu yetenek sayesinde tahmin edebilmektedir. Bu da doğru tahmin modellemesi için gereklidir.



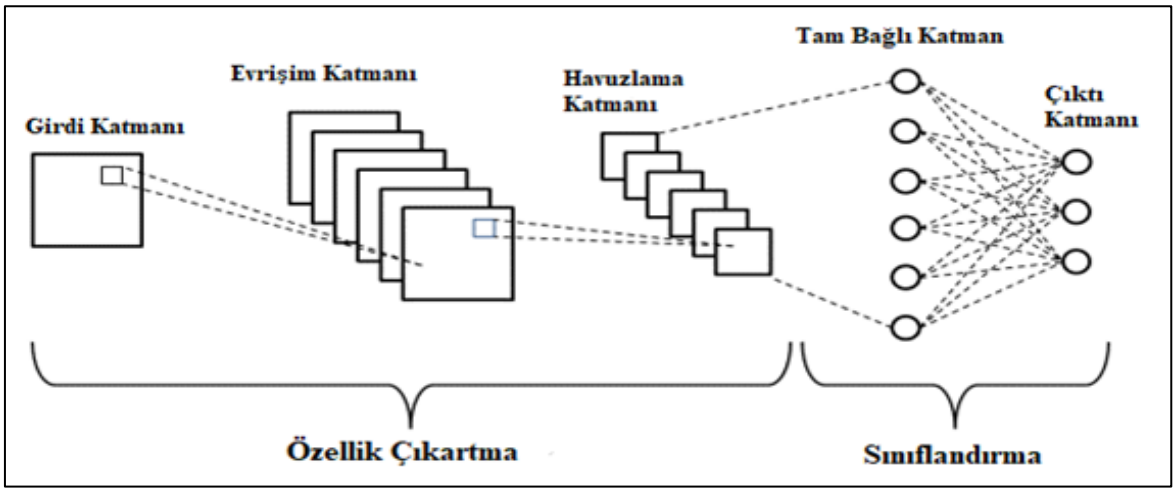

Şekil 4.5. Evrişim, havuzlama ve tam bağlantılı katmanlar [4]

Çıktı katmanı

Şekil 4.5'de görüldüğü üzere tam bağlantılı katmanlardan gelen çıktı verisi bu katmana iletilmektedir. Burada sigmoid veya softmax gibi fonksiyonlardan geçerek çıktı verisinin sınıflandırma görevi yapılır. Bu katmanda her sınıfın çıktısı her sınıfa ait olasılık puanına dönüştürülmektedir.

Evrişimsel sinir ağı mimarisi

Evrişimsel sinir ağı (CNN), yapay sinir ağlarını esas alan ve ızgara benzeri matris veri setinden özellik çıkarımı için kullanılan yapay sinir ağlarının genişletilmiş halidir. Evrişimsel sinir ağı mimarisi iki bölümden oluşmaktadır.



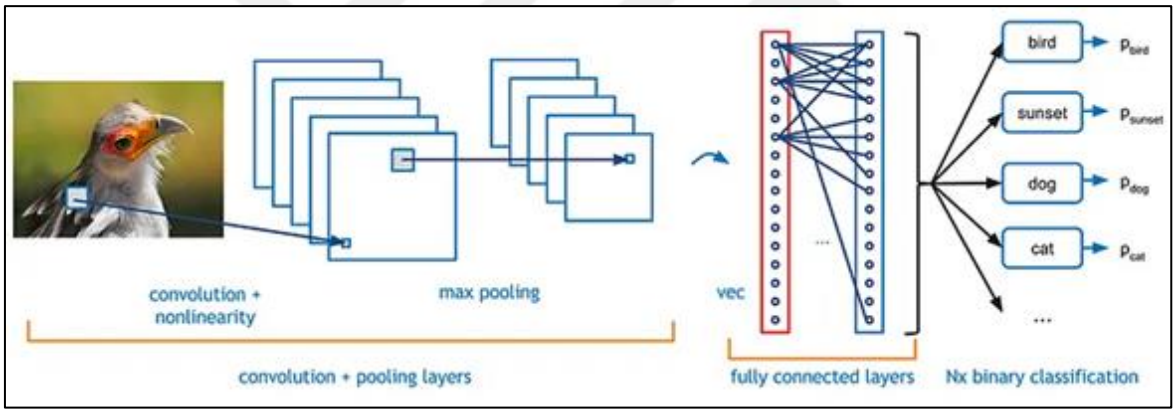

Şekil 4.6. Evrişimsel sinir ağı mimarisi [5].

Özellik çıkarımı

Evrişimsel sinir ağlarında özellik çıkarımı; görüntülerin çeşitli özelliklerini analiz edebilmek için ayıran ve tanımlayan işlemdir. Şekil 4.6'da görüldüğü üzere özellik çıkarımı giriş katmanı, çok sayıda evrişimsel katmanlar ve havuzlama katmanından oluşmaktadır.

Sınıflandırma

Evrişimsel sinir ağlarında sınıflama bölümü, Şekil 4.6'da görüldüğü üzere tam bağlı katmanlardan ve çıktı katmanında oluşmaktadır. Evrişimsel sinir ağı modelinin temel bileşenleri, kodlayıcı (encoder) ve kod çözücü (decoder) mimarileri olarak da gösterilmektedir.




Şekil 4.7. Evrişimsel sinir ağı tüm mimarisi [6]

### 4.1.1. Kodlayıcı mimarisi

Kodlayıcı, girdi görüntüsünden özellikleri çıkarmak için kullanılan katmanların toplamıdır. Görüntü özelliklerini çıkarmak için kullanılmaktadır, katmanlar derinleştikçe (konvolüsyon ve havuzlama) özellik haritasını küçültmektedir. Genelde kullanılan katmanlar şunlardır:

Konvolüsyonel katmanlar: Görüntüden yerel özellikleri çıkarmak için 2D konvolüsyonel katmanlar kullanılmaktadır.

Aktivasyon fonksiyonları: Genellikle ReLU (Rectified Linear Unit) gibi aktivasyon fonksiyonları kullanılarak modelin doğrusal olmayan özellikleri öğrenmesi sağlanmaktadır.

Örnekleme (Pooling) katmanları: Özellik haritalarının boyutunu azaltmak ve daha genel özellikler öğrenmek için max pooling veya average pooling katmanları kullanılmaktadır.

Örnek olarak kodlayıcı işleyişi; Giriş Görüntüsü → [Konv Katmanı (3x3) + ReLU] → [Max Pooling (2x2)] → [Konv Katmanı (3x3) + ReLU] → [Max Pooling (2x2)] → [Konv Katmanı (3x3) + ReLU] →[Max Pooling (2x2)] → Çıktı şeklinde olur.

### 4.1.2. Kod çözücü mimarisi

Kod çözücü, kodlayıcıdan elde edilen yüksek düzeyde özellikleri orijinal görüntü boyutunda bir segmentasyon haritasına dönüştürmekle sorumludur. Her bir pikselle ilgili tahminlerde bulunmaktadır. Ana bileşenleri şunlardır:

Yükseltme (Upsampling) katmanları: Düşük çözünürlüklü özellik haritalarını orijinal boyuta geri döndürmek için iki boyutlu interpolasyon, transposed convolution (özyinelemeli konvolüsyon) gibi yöntemler kullanılmaktadır.

Atlama bağlantıları: Kodlayıcıdan gelen özellik haritaları, aynı düzeydeki kod çözücü katmanlarına bağlanmaktadır. Bu, mekânsal bilgilerin korunmasına yardımcı olur ve segmentasyon kalitesini artırmaktadır.

Son katman: Genellikle son konvolüsyon katmanı ve softmax/sigmoid aktivasyon fonksiyonu, her piksel için sınıf olasılıklarını vermektedir.

Örnek olarak kod çözücü mimarisinin işleyişi; Kodlayıcıdan Gelen Özellikler → [Yükseltme (Transposed Conv)] → [Atlama Bağlantısıyla Birleştirme] → [Konv Katmanı (3x3) + ReLU] → [Yükseltme (Transposed Conv)] → [Konv Katmanı (1x1)] → Softmax/Sigmoid → Çıktı Segmentasyon Haritası

## 4.2. Derin Öğrenme Modeli Bilgisi

### 4.2.1. Tam evrişimli ağ

Tam Evrişimli Ağlar veya FCN'ler, esas olarak anlamsal bölütleme için kullanılan bir mimaridir. Konvolüsyon, havuzlama ve yukarı örnekleme gibi yalnızca yerel olarak bağlı

katmanlar kullanılmaktadır. Yoğun katmanların kullanımından kaçınmak daha az parametre anlamına gelmektedir(ağların daha hızlı eğitilmesini sağlamaktadır). Ayrıca, tüm bağlantılar yerel olduğu için bir FCN'nin değişken görüntü boyutları için çalışabileceği anlamına gelmektedir.

Ağ, bağlamı çıkarmak ve yorumlamak için kullanılan bir aşağı örnekleme yolundan ve yerelleştirmeye izin veren bir yukarı örnekleme yolundan oluşmaktadır.

Tam Evrişimli Ağ (FCN) tabanlı bölütleme, tam bağlı katmanları daha fazla evrişimli katmanla değiştirmektedir ve anlamsal bölütleme için popüler bir strateji ve temel olmaktadır. Tam evrişimli ağdan (FCN) etkilenen mevcut yaklaşımlar, üst düzey görüntü temsillerini koruyarak daha iyi bölütleme ayrıntıları sağlamaktadır [7].

### 4.2.2. Artık örüntü öğrenme modeli

Son yapılan çalışmalarda anlamsal bölütleme modelinin çıktısına ve jeneratif modele dayalı olarak anormallikleri sınıflandırmak için ek bir FCN ağı ile dağılım dışı anomali tespit doğruluğunun artması hedeflenmiştir [8]. Ancak bu yöntem anlamsal bölütleme çıktısından gelen yanlış bilgiler sonucunda eğitim sürecine onay önyargısı eklemektedir ve bu durum çıktı tarafında kötü genelleme ile sonuçlanmaktadır. Anlamsal bölütleme modelinin çıktısı veya sınıflandırma olasılıklarıyla çalışmak yerine, segmentasyon ağı ile entegre çalışan artık örüntü öğrenme yöntemi geliştirilmiştir [8]. Ara özelliklerin daha zengin bilgi içeriği göz önüne alındığında, bu yöntem potansiyel anormallikleri daha etkili bir şekilde tespit edebilmektedir. Artık örüntü öğrenimi derin öğrenme modellerinin gerçek dünya uygulamalarında karşılaştıkları dağılım dışı (Out-of-Distribution; OOD) verileri etkili bir şekilde tanımlamalarını sağlayan bir yöntemdir. Bu yöntem, modelin beklentileri ile gerçek çıktılar arasındaki farklılıkları kullanarak her bir pikselin dağılım dışı olup olmadığını belirlemektedir. Tezin ileri aşamasında bu yöntemin detayları verilecektir.

### 4.2.3. Anlamsal bölütleme modeli yeniden eğitimi

Anlamsal bölütleme modelini yeniden eğiten yöntemler iyi sonuçlar ortaya koymaktadır [9]. Örnek vermek gerekirse; birliktelik yöntemleri olarak da bilinen meta sınıflandırıcılar, bir grup zayıf modelin bir araya gelerek güçlü bir öğrenici oluşturabileceği ilkesine göre

çalışmaktadır. Bu teknik, aynı veri kümesi üzerinde birden fazla modelin (temel düzey sınıflandırıcılar) eğitilmesini ve ardından tahminlerini entegre etmek için bir meta öğrenme algoritması kullanılmasını içermektedir. Meta dağılım dışı yöntem, dağılım içi aykırı bölgelerdeki entropi sonuçlarını artırmak için aykırı bir veri kümesinden (örneğin COCO) dağılım dışı görüntülerini aykırı eğitim kümesine eklemektedir[10, 11]. Anormallikleri daha iyi simüle etmek için, DenseHybrid ve Aykırı Değer Maruziyeti (OE) gibi yöntemler eğitim klasöründeki veri kümelerinden anormal nesneleri rastgele kırparak aykırı görüntülere ekleyerek sentetik anomali verileri elde etmektedir[12, 13, 14]. Bu noktada (PEBAL) piksel bazında çekimser öğrenme ve piksel bazında enerji düzenleme yöntemi güncel anomali tespit modelleri arasında en yüksek performansı elde etmektedir [15]. Bununla birlikte, bu yeniden eğitim yaklaşımları kapalı küme segmentasyon doğruluğunu düşürebilmektedir.

### 4.4. Anlamsal Bölütleme ile Piksel Seviyesinde Anomali Tespiti

#### 4.4.1. Anlamsal bölütleme bağlamında anomali tespiti

Anlamsal bölütleme bağlamında anomali tespiti, görüntülerde eğitim sırasında öğrenilen beklenen kalıplara veya sınıflara uymayan bölgelerin tanımlanmasını ve bölütlenmesini içermektedir. Anlamsal bölütlemede amaç genellikle bir görüntüdeki her bir pikseli önceden tanımlanmış bir kategori kümesine sınıflandırmaktır. Ancak anomaliler, bu kategorilerin dışında kalan ve eğitim veri kümesinin bir parçası olmayabilecek pikseller veya bölgelerdir.

Anomaliler, bir veri kümesi içindeki normdan veya standart davranıştan sapmalar olarak tanımlanabilmektedir. Anlamsal bölütleme için anomali;

- Bilinen sınıflardan herhangi birine ait olmayan pikseller,
- Eğitim sırasında öğrenilmeyen beklenmedik özellikler veya yapılar sergileyen pikseller,
- Bozulmuş, değiştirilmiş veya verilerin geri kalanından önemli ölçüde farklı aykırı değerler (örn. hasarlı nesneler, olağandışı desenler) içeren pikseller anlamına gelmektedir. Girdi görüntüsünde oluşan bozulmanın nedenleri;

Gürültü: Görüntülerin çekimi sırasında oluşan rastgele hatalar sonucu görüntü kalitesinin düşmesi ve gürültü artması,

Aydınlatma koşulları: Yetersiz ve aşırı aydınlatma sonucunda girdi görüntüsünde oluşan parlak ve karanlık alanların, detayların kaybolmasına yol açması,

Hareket bulanıklığı: Girdi görüntüsü içerisinde hareket eden nesnelerin bulanık pikseller oluşturması,

Aykırı değerler: Girdi görüntüsündeki bazı bölgelerde verilerde sapmaların oluşması şeklinde belirtilmektedir.

### 4.4.2. Anlamsal bölütleme bağlamında piksel seviyesinde anomali tespiti

Piksel seviyesinde anomali tespiti, bir görüntünün her pikselinin anormal veya beklenmedik bir durumu gösterip göstermediğini belirlemeyi amaçlayan bir süreçtir. Bu yöntem, özellikle görüntü işleme ve bilgisayarla görme alanlarında önemli bir yere sahiptir. Anlamsal bölütleme ile birleştiğinde, bu süreç, görüntülerdeki her bir pikselinin bir nesne sınıfına atanması ile anormal piksellerin tanımlanmasını ve bölütlenmesini sağlar. Bu sürecin 4 temel adımı vardır: “Özellik Çıkartma”, “Model Eğitimi”, “Anomali Skoru Hesaplama” ve “Bölütleme Sonuçları” ’dır.

Özellik çıkartma: Görüntüdeki piksellerin özelliklerini çıkarmak için evrişimsel sinir ağları (CNN) veya diğer derin öğrenme yöntemleri kullanılmaktadır.

Model eğitimi: Normal görüntüler kullanılarak, modelin pikselleri normal sınıflar için nasıl sınıflandıracağı öğretilir. Anomaliler genellikle eğitim setinde yer almamaktadır.

Anomali skoru hesaplama: Eğitim tamamlandıktan sonra elde edilen görüntülerdeki her piksel için bir anomali skoru hesaplanır. Anomalilere karşılık gelen pikseller için bu skor genellikle yüksek olmaktadır.

Bölütleme sonuçları: Piksellerin anormal olup olmadığını belirlemek için genellikle belirli bir eşik değeri kullanılır. Eşik değerinin altında kalan pikseller, normal olarak kabul edilirken, üstündekiler anormal olarak işaretlenmektedir.

Yukarıda anlatılan adımlarda anomalilerin tespit edilmesi için kullanılan temel yaklaşım "Çapraz Entropi Kayıp Fonksiyonu" 'dur. Bu yaklaşımdaki amaç her bir piksele ait normal veya anormal sınıf olasılık değeri tahmin edilir. Kayıp fonksiyonu matematiksel olarak aşağıdaki gibi gösterilmektedir:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{K} y_{i,c}\log(p_i, c) \tag{4.1}$$

Bu denklemde;

- N görüntüdeki toplam piksel sayısını temsil etmektedir.
- K her piksel için olası sınıf sayısını temsil etmektedir. (Normal ve anormal sınıf, örneğin 2 sınıf eder.)
- $y_{i,c}$ değeri piksel i'nin gerçek sınıfını belirtmektedir. (Normal sınıf için 1, anormal sınıf için 0).
- $p_i, c$ değeri piksel i'nin c için tahmin edilen olasılığını belirtmektedir.

Modelin çıktısını anomali olasılık skorlarına dönüştürmek için "softmax fonksiyonu" kullanılmaktadır. Bu fonksiyon matematiksel olarak aşağıdaki gibi gösterilmektedir:

$$p_i, c = \frac{e^{z_{i,c}}}{\sum_{c=1}^{K} e^{z_{i,c}}} \tag{4.2}$$

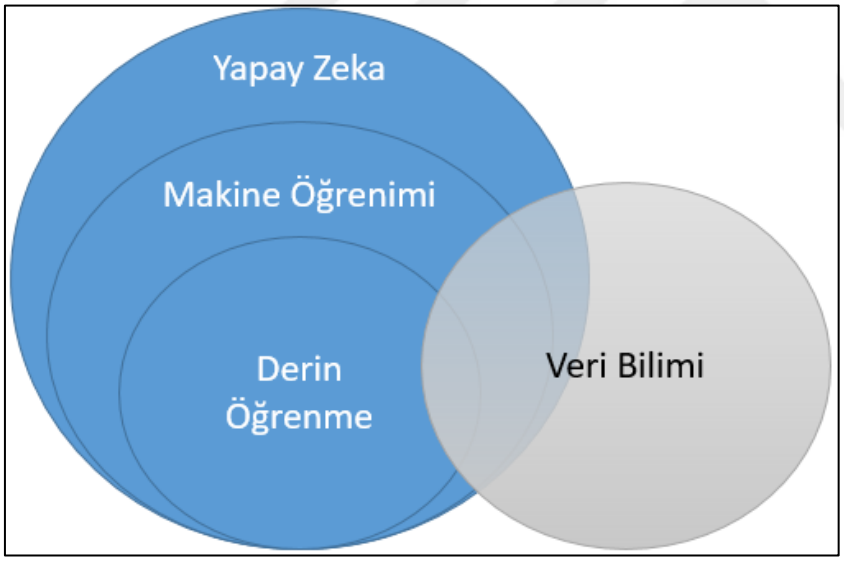
Bu denklemde $z_{i,c}$ değeri, modelin *i* piksel değeri için *c* sınıfı üzerindeki logit değerlerini temsil etmektedir. Bir pikselin anormal olup olmadığını belirlemek için $\tau$ eşik değeri kullanılmaktadır. Eğer pikselin anomali olasılık skoru bu eşik değerinden yüksek olursa piksel anomali olarak işaretlenmektedir.

# 5. MATERYAL VE YÖNTEMLER

## 5.1. Derin Öğrenme

Derin öğrenme, yapay sinir ağı mimarisine dayanan makine öğrenimidir. Yapay sinir ağı ise giriş verilerini işlemek ve öğrenmek için birlikte çalışan nöronlar adı verilen birbirine bağlı düğüm katmanlarından oluşmaktadır.

Tamamen bağlı bir derin sinir ağında, bir giriş katmanı ve birbiri ardına bağlanan bir veya daha fazla gizli katman bulunmaktadır. Her nöron, bir önceki katman nöronlarından veya giriş katmanından giriş almaktadır. Bir nöronun çıktısı, ağın bir sonraki katmanındaki diğer nöronlar için girdi olmakta ve bu süreç, son katman ağın çıktısını üretene kadar devam etmektedir. Sinir ağının katmanları, giriş verilerini bir dizi doğrusal olmayan dönüşümle dönüştürerek ağın giriş verilerinin karmaşık gösterimlerini öğrenmesini sağlamaktadır.




Şekil 5.1. Derin öğrenmenin kapsamı

Çizelge 5.1. Denetimli ve denetimsiz derin öğrenme yöntemleri

| | Denetimli Öğrenme | Denetimsiz Öğrenme |
|---|---|---|
| Veri tipi | Etiketli Veriler | Etiketsiz Veriler |
| Karmaşıklık | Daha az Karmaşıklık | Daha Çok Karmaşıklık |
| Uygulama | Sınıflandırma ve Regresyon Problemleri | İlişiklendirme ve Kümeleme Problemleri |
| Güvenirlilik | Daha Doğru Sonuçlar Verir | Orta Seviye Güvenirlilik |
| Analiz | Çevirimdışı Analiz | Gerçek Zamanlı Analiz |

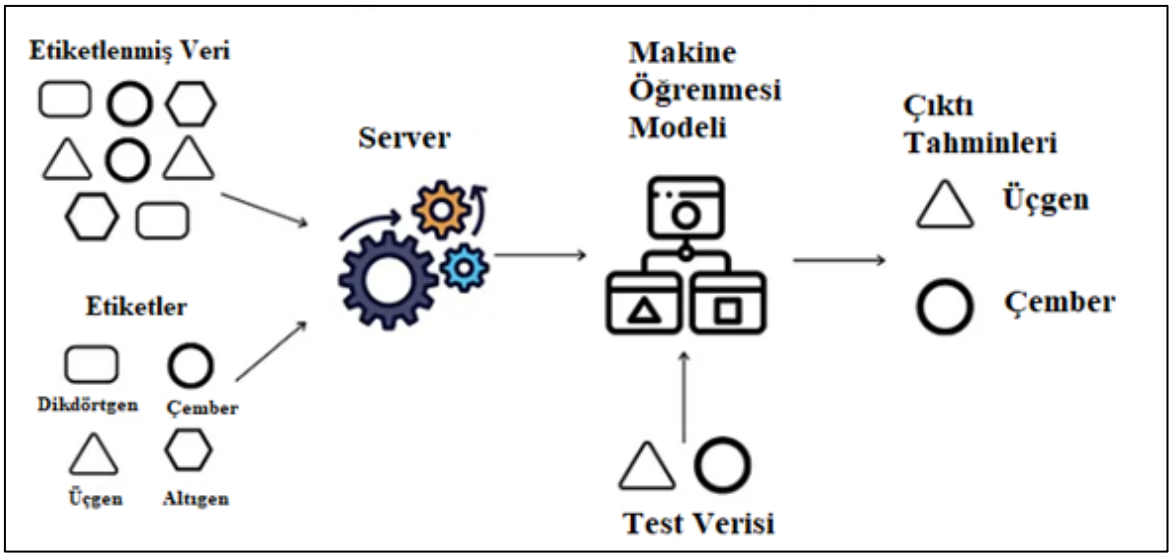
Derin öğrenme algoritmaları denetimli ve denetimsiz makine öğrenim yöntemleri olarak ikiye ayrılmaktadır. Aşağıda bu yöntemler hakkında detaylı bilgi verilmiştir.

### 5.1.1. Denetimli makine öğrenimi

Denetimli makine öğrenimi, yapay sinir ağının verileri sınıflandırmayı öğrendiği ve etiketli veri setlerine dayanarak tahminler yapabildiği makine öğrenim yöntemidir. Bu teknikte amaç, girişteki verilerin istenen çıktı ile eşleştirilmesidir. Yapay sinir ağı gerçek hedef ve tahmin edilen çıktı arasındaki farktan kaynaklanan hatalara ve maliyetlere bakarak tahminler yapmayı öğrenmektedir. Bu sürece geri yayılım denmektedir. Bu işlemden sonra model yeni giriş verileri için çıktı tahmini yapabilmeyi öğrenmektedir.

Evrişimli sinir ağları ve tekrarlayan sinir ağları gibi denetimli öğrenme algoritmaları genelde regresyon, nesne algılama, duygu analizi ile görüntü sınıflandırmaları ve tanıma gibi konularda tercih edilmektedir. Bu; modele sağlanan veri setinde hedef çıktı ile girişteki verilerin doğru etiketlenmiş olması gerekmektedir. Böylece model, tahminlerini gerçek sonuçlar ile kıyaslayarak parametrelerini doğru bir şekilde ayarlayabilecektir [16].

Denetimli makine öğrenimi öncelikle sınıflandırma ve regresyon problemleri için kullanılmaktadır. Bu yöntemin avantajları yüksek doğruluk ve yorumlanabilirliktir. Model etiketli veri setler ile eğitildiği için belirli görevlerde yüksek tahmin doğruluk oranına sahip olacaktır. Model etiketli çıktılar ile eğitildiği için tahminlerin nasıl yapıldığını anlamak daha kolay olacaktır. Fakat denetimli makine öğrenmesinin dezavantajlarından bir tanesi; etiketli verileri elde etmesi zaman alan ve maliyetli bir işlemdir. Bir başka dezavantajı ise; model etiketleri verileri eğitimde aşırı uyumlu şekilde öğrenmekte ve dağılım dışı bir veri karşısında hatalı çıktı üretebilmektedir.

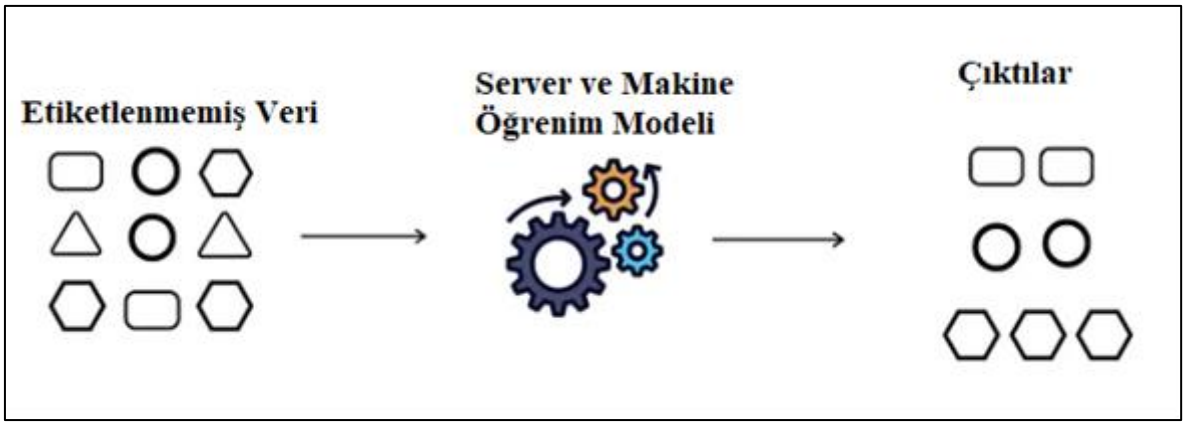



Şekil 5.2. Denetimli öğrenme çalışma prensibi [17]

### 5.1.2. Denetimsiz makine öğrenimi

Denetimsiz makine öğrenimi, yapay sinir ağının etiketsiz veri setlerindeki desenleri keşfederek tahmin yaptığı veya veri setlerini etiketleyerek sınıflandırdığı makine öğrenim yöntemidir. Bu yöntemde doğrudan beklenen çıktılar yoktur. Çünkü giriş tarafındaki verilerde herhangi bir etiket veya sınıflandırma yoktur. Burada amaç, etiketlenmemiş veri setlerindeki desenleri, kalıpları ve ilişkileri dışarıdan herhangi bir yardım olmadan modelin keşfetmesi dahası bu verileri gruplandırabilmesidir. Popüler denetimsiz makine öğrenim algoritmaları K-Ortalama Kümelemesi, hiyerarşik sınıflandırma, temel bileşen analizi ve otokodlayıcılar bulunmaktadır. Boyut azaltma, kümeleme ve anormallik tespiti gibi denetimsiz görevler için otokodlayıcılar kullanılmaktadır.

Etiketli veriye ihtiyaç duymadığından dolayı keşifsel analiz görevleri için kullanılmaktadır. Verilerde belirgin olmayan nesneleri bulabildiği için gizli desenleri keşfetmektedir. Verilerin etiketsiz olmasından dolayı denetimli öğrenmeye kıyasla modelin doğruluğunu belirlemek biraz daha zordur.

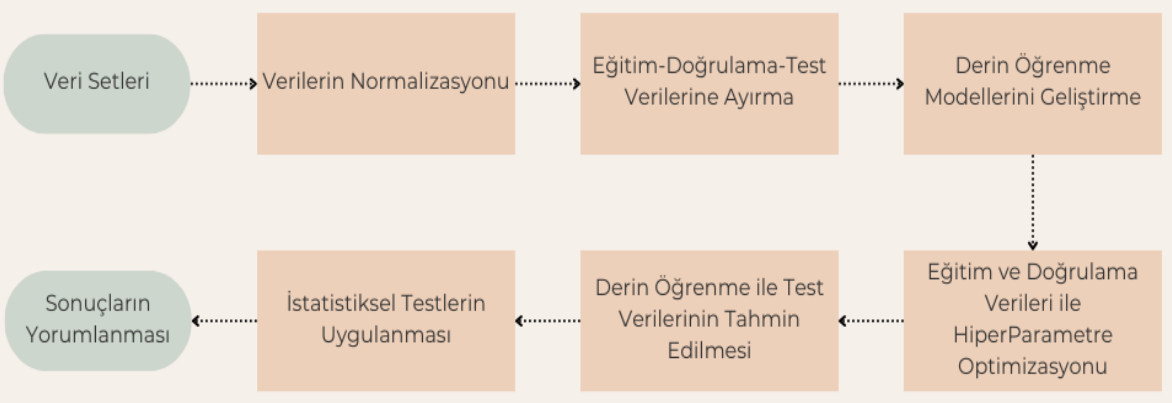



Şekil 5.3. Denetimsiz öğrenme çalışma prensibi [17]

## 5.2. Derin Öğrenme Modeli Eğitim Süreci

Derin öğrenme modeli seçildikten sonra seçilen modelin belirli görevleri yerine getirmesi beklenmektedir. Uygun eğitim olmadan seçilen model; beklenen performansı gösterememekte ve harcanan kaynaklar (maliyet, zaman, emek vs.) boşa gitmektedir. Model çıktısı önemli ölçüde hatalı ve yanlış olmaktadır. Diğer yandan eğer model eğitilir fakat doğru bir şekilde eğitilmezse ya aşırı uyum ya da yetersiz uyum riskleri ortaya çıkacaktır. Aşırı uyum sağlayan model eğitim verilerine aşırı odaklandığı için eğitimde öğrendiklerini bağlam değişikliğinde (yeni, görülmemiş verilere) uygulamakta zorlanacaktır. Yetersiz uyum gösteren model ise verilerdeki desenleri tam olarak öğrenemediği için hatalı tahminlerde bulunacaktır.

Yukarıda anlatılan sorunlara karşılık bir derin öğrenme modelini doğru bir şekilde eğitmek verilerdeki desenleri, örüntüleri ve eğilimleri doğru tahmin etmeye neden olacaktır. Bağlam dışı verilerde ve gerçek dünya senaryolarında da yukarıdaki gibi bir uyum sorunu olmayacaktır.




Şekil 5.4. Derin öğrenme modeli eğitim süreci

### 5.2.1. Veri toplama

Bir derin öğrenme modelini eğitmek için doğru ve ilgili verileri toplamak çok önemlidir. Model eğitiminde kullanılan verilerin kalitesi ve çeşitli olması modelin etkin olması için çok önemlidir. Veri toplamanın birinci adımı ihtiyacın anlaşılması ve akabinde derin öğrenme için gereksinimlerin netleştirilmesidir. Bu çalışma genelde etiketli veya etiketsiz veriler gibi modelin ihtiyaç duyacağı veri türünü belirlemek için yapılmaktadır. Yapılan çalışmanın sonucuna bağlı olarak eğitim için gereken veriler kamuya açık veri setleri, sensörler veya veri tabanları gibi farklı kaynaklardan alınabilmektedir.

Alınan verilerin hem ihtiyaç ile alakalı hem de doğru olduğundan emin olunması gerekmektedir. Verilerin kalitesi direkt olarak derin öğrenme modelinin tahmin ve genelleme yeteneğini etkilemektedir. Verilen kalitesi ne kadar iyi olursa, modelin tahmin etme ve karar verme performansı da o kadar iyi olacaktır.

### 5.2.2. Ön işleme

Verilerin hazırlanması ve ön işlenmesi, eğitim ve test için kullanılacak ham verileri makine öğrenim modelleri için uygun bir biçime dönüştürme adımıdır. Bu süreçte ana amaç verilerin temizlenmesi, verilerin kategorik özelliklerini ve metni ele almak ve veriler üzerinde normalleştirme veya standardizasyon teknikleri ile gerçek değeri olan özelliklerin ölçeklendirilmesidir.

Bir veri setinden alan veride beklenen değerlerin yanı sıra genellikle gürültü verisi de olmaktadır. Bunlara ek olarak; ekstra özellikler, eksik veriler ve aykırı değerler de verilerin

içinde olabilmektedir. Veri temizleme; aykırı değerleri test etmek, verilerdeki gereksiz özellikleri kaldırmak, veri setindeki eksik verileri doldurmak ve veri seti içerisinde gelen gürültülü verileri filtrelemek gibi işlemlerden oluşmaktadır.

Genel olarak sinir ağlarının eğitiminde; eğitimi dengeleyebilmek için ağın ağırlık parametreleri küçük değerler ile başlatılmaktadır. Ancak bu durumda model büyük değerlere sahip girdiler karşısında tahmin yapmakta zorlanacaktır. Bu problem girdinin gerçek değerli özellikleri 0 ile 1 arasında olacak şekilde normalleştirilmesi ile çözülmektedir.

### 5.2.3. Veri hazırlama

Toplanan veriler model eğitim fazına geçmeden önce eğitim sürecine uygun şekilde hazırlanmalıdır. Veriler eğitim, test ve doğrulama kümelerine ayrıştırılmalıdır. Verilerin hazırlanması ve modellerin eğitilmesi için genelde Python dili kullanılmaktadır. Bunun sebebi Python dilinin veri hazırlama görevlerini kolaylaştırmak için sunduğu bir dizi hazır kütüphanedir. Eğitim kümesi modelin bu kümedeki verilere uymasını sağlamak için kullanılmaktadır. Doğrulama kümesi ise modelin ayarlanmasına yardımcı olmaktadır. Test kümesi ise modelin son halinin performansını değerlendirmek için kullanılmaktadır. Verilerin bu şekilde bölünmesi, modelin her aşamada daha önce görülmemiş veriler üzerinde öğrenmesini sağlamaktadır. Böylece aşırı uyum sorunu riski en aza indirilmektedir.

### 5.2.4. Model eğitimi

Bu aşamada makine öğrenme algoritmasına hazırlanan girdi verilerinin beslenmesi yapılmaktadır. Model ne için eğitilecek ise ona göre farklı algoritmalar ile birlikte mevcut veriler kullanılmaktadır. Bu algoritmalar derin öğrenme, sınıflandırma, kümeleme, denetimli ya da denetimsiz öğrenme olabilmektedir. Model eğitim sırasında etiketli verilerden öğrenmektedir. Sürecin devamında hata iyileştirmesi ve performans artırımı için geri besleme ile iç parametrelerini ayarlamaktadır. Model; eğitim sürecinde gradyan inişi gibi yinelemeli bir iyileştirme algoritması içermektedir. Bunun görevi, hesaplanan hatalara göre modelin iç parametrelerini ayarlamaktır. Böylece model eğitimi belli bir döngü sayısı sonunda tamamlanacaktır.

Bir önceki adımda bahsedilen veri setlerine benzer şekilde, farklı alanlar ve uygulamalar için model eğitiminde yaygın olarak kullanılan bazı açık kaynaklı makine öğrenim kütüphanesi mevcuttur. Bu tez kapsamında PyTorch Kütüphanesi kullanılacaktır.

PyTorch kütüphanesi bilgisayarlı görüde sıklıkla kullanılan bir derin öğrenme çerçevesidir. Facebook tarafından geliştirilen bu çerçeve, dinamik hesaplama grafikleri ile esnekliği vurgulamaktadır.

Modelin karmaşıklığı, kullanılan veri setinin boyutu ve hesaplama algoritmaları gibi faktörler makine öğrenimi için grafik işleme birimi (GPU) ihtiyacını ortaya koymaktadır. Bu çalışmada da kullanılan evrişimsel sinir ağları (CNN) gibi derin öğrenme modelleri çok sayıda parametreye sahiptir. Eğitim sırasında yoğun hesaplamalar gerektirmektedir ve eğitim süresi uzundur. Bu tür karmaşık modellerin büyük veri setleri ile eğitilmesinde grafik işlemci birimi kullanılarak eğitim süreleri kısaltılmaya çalışılmaktadır.

### 5.2.5. Değerlendirme

Derin öğrenme modelinin eğitimden sonra performansı değerlendirilmelidir. Çapraz doğrulama yöntemi bir modelin performansını ve genelleyebilme özeliklerini değerlendirmek için kullanılan bir yöntemdir. Mevcut veri kümesini birden fazla alt kümeye bölmektedir. Bu kümeleri eğitim ve doğrulama için tekrarlı kullanma prensibine göre çalıştırmaktadır.

Hata oranını belirlemek için ortalama mutlak hata, ortalama karesel hata ve R-kare gibi değerlendirme yöntemleri bulunmaktadır:

- Ortalama Mutlak Hata (MAE): Beklenen gerçek değerler ile tahmin edilen değerler arasındaki mutlak farkların ortalamasıdır.
- Ortalama Karesel Hata (MSE): Beklenen gerçek değerler ile tahmin edilen değerler arasındaki karesel farkların ortalamasıdır.
- Kök Ortalama Karesel Hata ( RMSE ): MSE 'nin karekökü olup, ortalama hata büyüklüğünün bir ölçüsüdür.
- R-kare (R2): Bağımlı değişkendeki varyansın, bağımsız değişkenlerden tahmin edilebilen oranıdır.

Sınıflandırma görevleri için yaygın değerlendirme ölçütleri şunlardır:

- Doğruluk(Accuracy): Toplam örnekler içindeki doğru sınıflandırılmış örneklerin oranıdır.
- Kesinlik(Precision): Tüm olumlu tahminler arasındaki gerçek olumlu tahminlerin oranıdır.
- Hatırlatma: Tüm gerçek pozitif örnekler arasındaki gerçek pozitif tahminlerin oranıdır.
- F1-puanı: Hassasiyet ve geri çağırmanın harmonik ortalaması, model performansının dengeli bir ölçümünü sağlamaktadır.

Karar Değerlendirme Grafik Eğrisi Altındaki Alan (AUC-ROC, Area Under the Curve - Receiver Operating Characteristic): Sınıflandırma modellerinin performansını değerlendirmek için yaygın olarak kullanılan bir metriktir. Bu metrik, modelin pozitif ve negatif sınıfları ne derece iyi ayırt ettiğini görsel ve sayısal olarak ifade etmektedir. ROC eğrisi, sınıflandırma modelinin farklı eşik değerlerinde pozitif sınıfı doğru bir şekilde sınıflandırma yeteneğini görselleştirmektedir. Eğri, şu iki ölçüm arasındaki ödünleşmeyi göstermektedir:

Doğru Pozitif Oranı (True Positive Rate, TPR): Gerçek pozitif oran demektir. (Duyarlılık, Sensitivity)

$$TPR = \frac{\text{Doğru Pozitif}}{\text{Doğru Pozitif+ Yanlış Negatif}} = \frac{TP}{TP+FN} \tag{5.1}$$

Yanlış Pozitif Oranı (False Positive Rate, FPR): Yanlış pozitif oran demektir.

$$FPR = \frac{\text{Yanlış Pozitif}}{\text{Yanlış Pozitif+ Doğru Negatif}} = \frac{FP}{FP+TN} \tag{5.2}$$

ROC eğrisi, FPR'ın yatay eksende, TPR'ın ise dikey eksende yer aldığı bir grafikte çizilmektedir. Eğrinin şekli, modelin sınıflandırma yeteneğini gösterir; eğrinin yukarıda ve sola daha yakın olması, modelin daha iyi performans gösterdiğini ifade etmektedir. AUC, ROC eğrisinin altında kalan alanı temsil etmektedir. AUC değeri, modelin performansını 0 ile 1 arasında bir değerde şu şekilde ifade etmektedir:

- AUC = 0.5: Modelin rastgele tahmin yapması (yani, model hiç bir bilgi sağlamıyor demektir).
- AUC > 0.5: Modelin pozitif ve negatif sınıflar arasında bir ayrım yapabiliyor olması anlamına gelmektedir.
- AUC = 1: Model, tüm pozitif örnekleri doğru bir şekilde tahmin edebiliyor demektir, yani mükemmel bir model elde edilmiştir.

Duyarlılık Doğruluk Grafiği Altındaki Alan (Area under the Precision-Recall Curve, AuPRC): Özellikle dengesiz veri setlerinde sınıflandırma modellerinin performansını değerlendirmek için kullanılan bir metriktir. Duyarlılık-Doğruluk (Precision-Recall, PR) eğrisi, modelin farklı eşik değerlerinde doğruluk (precision) ve geri çağırma (recall) oranlarının çizilmesi ile elde edilmektedir.

Kesinlik (Precision): Doğru pozitif tahminlerin, toplam pozitif tahminlere oranıdır. Formülü:

$$\text{Precision} = \frac{\text{Doğru Pozitif}}{\text{Doğru Pozitif+ Yanlış Pozitif}} \tag{5.3}$$

Doğruluk (Recall) : Doğru pozitif tahminlerin, toplam gerçek pozitiflere oranıdır. Formülü:

$$\text{Recall} = \frac{\text{Doğru Pozitif}}{\text{Doğru Pozitif+ Yanlış Negatif}} \tag{5.4}$$

Duyarlılık-Doğruluk (Precision-Recall, PR) eğrisi altında kalan alan, genellikle “0” ile “1” arasında bir değer almaktadır. “1” değeri mükemmel bir model anlamına gelirken, “0” değeri modelin tamamen hatalı sonuçlar verdiğini göstermektedir. “AuPRC” metriği özellikle pozitif örneklerin az olduğu durumlarda yani dengesiz veri setlerinde daha anlamlı olmaktadır. Hem kesinliği hemde geri çağırma eğrisi modelin başarı durumunu daha iyi yansıtmaktadır.

Karışıklık ölçütleri: Bir sınıflandırma modelinin performansını özetleyen, gerçek pozitif, gerçek negatif, yanlış pozitif ve yanlış negatif durumların sayısını gösteren bir matristir.

### 5.2.6. Hiper parametre ayarı

Model eğitimi tamamlandığı için bir sonraki adımı modeli daha fazla optimize etmektir. Ayarlama ve optimize etme, modelimizin performansını ve genelleme yeteneğini en üst düzeye çıkarmasına yardımcı olmaktadır. Bu süreç, hiper parametreleri ince ayarlamayı , en iyi algoritmayı seçmeyi ve özellik mühendisliği teknikleriyle özellikleri iyileştirmeyi içermektedir. Hiper parametreler, eğitim süreci başlamadan önce ayarlanan ve makine öğrenimi modelinin davranışını kontrol eden parametrelerdir. Bunlar öğrenme oranı, düzenleme ve modelin parametreleri gibi dikkatlice ayarlanmalıdır.

## 5.3. Dağılım Dışı Veri

Son teknoloji makine öğrenimi modelleri deneysel test verileri üzerinde son derece iyi performans göstermektedir. Bununla birlikte, bu modeller örtük olarak kapalı dünya varsayımı yapmakta olup, gerçek dünyada karşılaşacakları verilerin; eğitim ve test verileriyle aynı dağılımdan çekildiğini varsaymaktadır. Ancak bu durum her zaman geçerli değildir ve bazı girdiler için modelin hiç uygulanmaması mantıklı olmaktadır.

Dağılım dışı veri (OOD), bir veri kümesinin genel dağılımına uymayan, istatistiksel olarak beklenmeyen veya aşırı uç değerler olarak tanımlanmaktadır. Bu veriler, modelin öğrenme sürecinde sorun oluşturabilmekte, örneğin piksel tabanlı bölütleme gibi algoritmalar, bu tür verileri yanlış yorumlamaktadır. Bilinen bir veri kümesi verildiğinde amaç, yeni bir örneğin aynı dağılıma ait olup olmadığını belirlemektir. Dağılım dışı veri tespitinin önemli bir varsayımı, çok sayıda atipik örneğe erişimin olmamasıdır. Bu nedenle, genellikle OOD olaylarının nadir meydana gelmesinden ve meydana gelene kadar ayrıntılarının bilinmemesinden dolayı OOD tespiti uygulanmaktadır [18, 19].

### 5.3.1. Uygulamalarda dağılım dışı verilerin oluşma nedenleri

Anlamsal bölütleme uygulamalarında dağılım dışı veri oluşmasının birkaç temel nedeni bulunmaktadır. Bu nedenler şunlardır:

- Veri toplama hataları: Hatalı veya yanlış veri toplama yöntemleri, anormal değerlerin ortaya çıkmasına sebep olmaktadır. Örneğin, yanlış ölçümler veya veri giriş hataları dağılım dışı verilere yol açmaktadır.

- Aşırı uç değerler: Bazı veri kümesi özellikleri doğal olarak aşırı uç değerlere sahip olmaktadır. Bu durum, örneğin, belirli bir olayın nadir gerçekleşmesi durumunda ortaya çıkmaktadır.
- Veri dağılımındaki anomaliler: Veri setinde yer alan bazı özelliklerin beklenmedik dağılımlara sahip olması, önemli ölçüde farklı değerlerin ortaya çıkmasına neden olmaktadır.
- Farklı alt gruplar: Veri setindeki farklı alt gruplar veya kategoriler arasında önemli farklılıklar olduğunda, bu durum dağılım dışı verilere yol açmaktadır. Örneğin, bir gruptaki bireylerin özellikleri diğerinden çok farklı olabilmektedir.
- Modelin yanlış ayarları: Anlamsal bölütleme yapmak için kullanılan algoritmanın yanlış ayarlanması ya da seçimi, bazı veri noktalarını yanlış değerlendirmeye neden olmaktadır.
- Öznitelik seçimi: Yanlış veya yetersiz özniteliklerin seçilmesi, bölütleme sonuçlarını etkileyerek dağılım dışı verilere neden olmaktadır.
- Dağılım kayması: Gözlenen verilerin dağılımı zaman içinde değişir ve bu da performansta düşüşe neden olmaktadır.

Derin öğrenme modelinin eğitim senaryosu, dağılım dışı verileri tespit etmede önemli bir rol oynamaktadır. Bu süreçte, modelin nasıl eğitildiği ve hangi tekniklerin kullanıldığı, dağılım dışı verileri belirlemek ve ele almak için kritik öneme sahiptir. Dağılım dışı veriler modelin öğrenme yeteneğini olumsuz yönde etkileyebilmektedir. Bu verilerin varlığı, modelin normal veri için öğrenmesi gereken kalıpların yanıltıcı olmasına neden olmaktadır.

Anlamsal bölütleme uygulamalarında, dağılım dışı veriler, segmentasyon sınırlarını çarpıtmakta ve belirli bölgelerde yanlış sınıflandırmalara yol açmaktadır. Örneğin, hayvanların doğal alanlarında anormal renk veya dokuya sahip bir piksel, modelin o alanı yanlış sınıflandırmasına neden olmaktadır. Dağılım dışı veriler, doğruluk, F1 skoru gibi değerlendirme metriklerini bozarak, modelin gerçek performansını yanıltıcı bir şekilde düşürmektedir.

### 5.3.2. Dağılım dışı verileri tespit etme

Aykırı değer tespiti: Bu sorun, yalnızca dağılım içi örnekleri içeren temiz bir veri kümesi incelenmediği gerçeğiyle ayırt edilmektedir. Bunun yerine, tek bir etiketlenmemiş veri kümesi alınmakta ($\{X_i\}_{i=1}^{I}$ ) ve içindeki nadir veya düşük olasılıklı örnekler tespit edilmeye çalışılmaktadır. Bunlar genellikle gürültüden kaynaklanmaktadır ve burada amaç, verileri başka bir amaç için kullanmadan önce verilerin içindeki aykırı değerleri tespit edip kaldırmaktır.

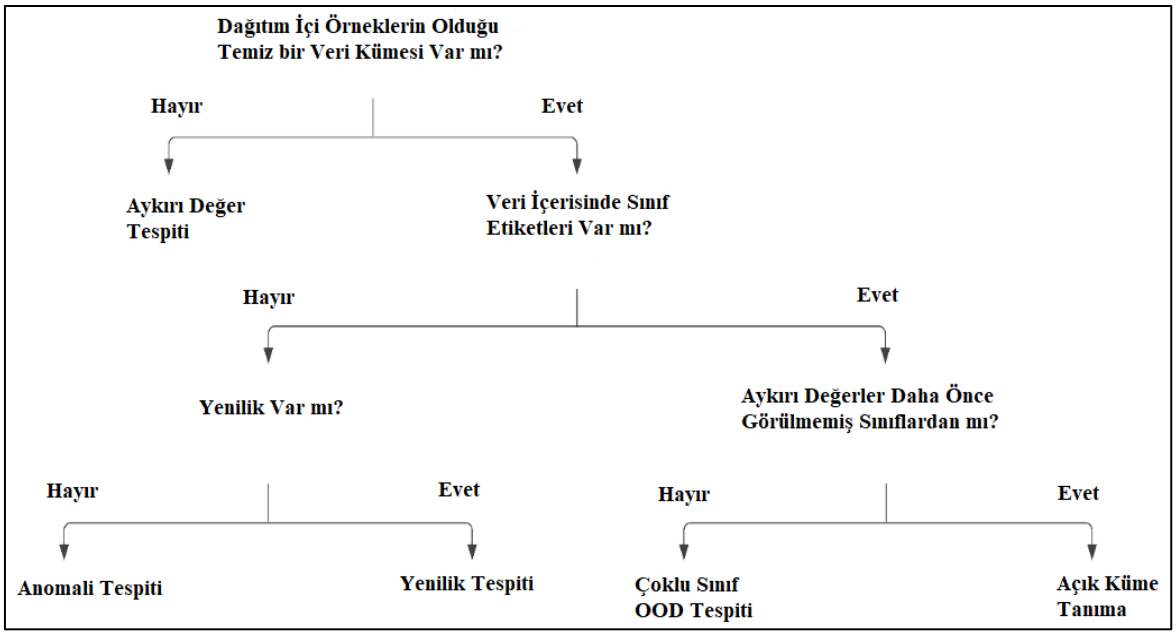
Anomali tespiti: Sadece dağılım içi örneklerden oluşan ve etiketlenmemiş bir veri kümesi , $\{X_i\}_{i=1}^{I}$ şeklinde gösterilmektedir. Yeni bir örnek , $X^*$, değerinin etiketlenmemiş veri kümesiyle aynı dağılıma ait olup olmadığı belirlenmelidir. Bu durumda, anomaliler genellikle ilgi duyulan niceliklerdir ve hatalı, hileli veya şüpheli örnekleri temsil etmektedirler.

Yenilik tespiti: Sadece dağılım içi örneklerden oluşan ve etiketlenmemiş bir veri kümesi , $\{X_i\}_{i=1}^{I}$ şeklinde gösterilmektedir. Yeni bir örnek, $X^*$, için yeniliğini belirten bir puan eklenebilir. Bu, anomali tespiti ile aynı sorundur, ancak burada son bir eşikleme adımı yoktur.

Açık küme tanıma: C ön tanımlı etiket veri setinden y etiketlerinin geldiği bilinen, etiketli bir ($\{X_i, Y_i\}_{i=1}^{I}$ ) veri kümesi ele alınmalıdır. Yeni bir örnek olan $X^*$ ‘ın $Y^* \in C$ etiketi bilinen sınıflarından hangisine ait olduğu belirlenmelidir. Ya da daha önce karşılaşılmamış sınıf setinden ($Y^* \in U$) birisine ait olduğu tespit edilmelidir. Buradaki olağan senaryo nesne sınıflandırmasıdır ve görülmeyen sınıflar daha önce gözlemlenmeyen anlamsal kategorilerdir. Örneğin, kuş türleri arasında ayrım yapmak için bir sınıflandırıcı eğitilmektedir. Test zamanında yeni türler tanımlanabilir, ancak eğitim sırasında gözlemlenmemiş bir tür gibi görünüyorsa işaretlenerek açık küme tanıma yapılmaktadır.

Çok sınıflı OOD tespiti: C ön tanımlı etiket veri setinden y etiketlerinin geldiği bilinen, etiketli bir ($\{X_i, Y_i\}_{i=1}^{I}$ ) veri kümesi ele alınmalıdır. Yeni bir örnek olan $X^*$ ‘ın $Y^* \in C$ etiketi bilinen sınıflarından hangisine ait olduğu belirlenmelidir. Ya da daha önce hiç sınıflandırılmadığı tespit edilmelidir. Bu, açık küme tanıma ile çok benzer bir sorundur, ancak burada vurgu farklıdır. Burada tipik olmayan veriler, görünmeyen bir sınıftan

gelmek yerine, bir şekilde bozulmuş veya farklı olan gerçek örnekler olmaktadır. Örneğin, kuş türleri arasında ayrım yapmak için bir sınıflandırıcı eğitildikten sonra test zamanında yeni türler tanımlanmaktadır, ancak bulanık olan, kuş içermeyen veya görünümün çok atipik olduğu (örneğin, kanadın aşırı yakın çekimi) görüntüleri de işaretlenerek dağılım dışı veriler elde edilmektedir.




Şekil 5.5. Dağılım dışı verileri tespit etme süreci

## 5.4. Artık Örüntü Öğrenme Yöntemi

Artık örüntü öğrenme yöntemi, modelin beklentileri ile gerçek çıktılar arasındaki farklılıkları kullanarak her bir pikselin dağılım dışı olup olmadığını belirlemektedir. Bu yöntemde;

I: Giriş görüntüsü, $I \in \mathbb{R}^{HxWxC}$ (yükseklik H, genişlik W, kanal sayısı C ile temsil edilir),
$Y$: Gerçek segmentasyon maskesi, $Y \in \mathbb{R}^{HxW}$ (her piksel için sınıf etiketleri),
$\acute{Y}$: Modelin tahmin ettiği segmentasyon maskesi, $\acute{Y} = f(I)$,

olarak ele alınırsa; her piksel için elde edilecek artık değer:

$$R_{ij} = \acute{Y}_{ij} - Y_{ij} \tag{5.5}$$

denklemi ile hesaplanmaktadır. Burada $R_{ij}$ değeri, (i,j) pikseli için artık değeri ifade etmektedir. Derin öğrenme modeli görüntüden elde edilen öznitelik haritalarını kullanmaktadır. Öznitelik haritalarını elde etmek için özellik çıkartıcı fonksiyon (g) kullanılmaktadır:

$$F = g(I) \tag{5.6}$$

Artık örüntü öğrenme yöntemi modelin kabul edilebilir artık örüntülerini karakterize eden bir fonksiyonu (h) öğrenmeyi amaçlamaktadır [20]. Öğrenme hedefi, aşağıda verilen kayıp fonksiyonu üzerinden optimize edilmektedir:

$$L = \sum_{(i,j)\in\Omega} \left\| R_{i,j} \right\|^2 + \lambda \mathcal{R}(R) \tag{5.7}$$

Burada Ω piksel indekslerinin kümesini, $\mathcal{R}(R)$ ise artık değerlerin düzenleme terimini ifade etmektedir. Dağılım dışı verileri algılamak için bir eşik değeri belirlenir. Derin öğrenme modeli eğitildikten sonra artık değere $\tau$ eşik değerinin uygulandığı OOD algılama koşulu aşağıdaki gibi ifade edilmektedir:

$$OOD_{i,j} = \begin{cases} 1 \; eğer \; \left\| R_{i,j} \right\| > \tau \\ 0 \; değilse \end{cases} \tag{5.8}$$

Buradaki ifadeye göre bir piksel OOD olarak sınıflandırılırsa, o pikselin artık değeri eşik değerini aşmaktadır. Derin öğrenme modelinin eğitim sürecinde kullanılan kayıp fonksiyonu aşağıdaki gibi ifade edilmektedir:

$$\mathcal{L}_{Toplam} = \mathcal{L}_{seg} + \alpha \mathcal{L}_{artık} \tag{5.9}$$

Burada $\mathcal{L}_{seg}$ anlamsal bölütleme kaybını(çapraz entropi kaybı), $\mathcal{L}_{artık}$ ise artık değerler ile ilgili kaybı göstermektedir. $\alpha$ kat sayısı ise her iki bileşenin kayıp hesabındaki katkısını dengeleyen bir ağırlık değeridir.

## 5.5. DeepLabv3+ Modeli

DeepLab yöntemleri gelen özellikleri filtrelemek ve alıcı alanı genişletmek için genişletilmiş evrişim kullanmaktadır[21, 22, 23]. DeepLabv3+, tıbbi görüntüleme, otonom sürüş vb. gibi görüntünün anlamsal bölütlemesini çeşitli uygulamalarda kullanım alanı bulan yaygın bir anlamsal bölütleme modelidir. Son zamanda önerilen anomali tespit yaklaşımlarıyla doğru karşılaştırma yapabilmek için bu çalışmada kullanılan derin öğrenme modeli DeepLabV3+ tabanlıdır[23]. Google'ın geliştirdiği en yeni derin öğrenme modeli olan Deeplabv3+ bu alanda şu an için amiral gemisi konumunda olup, bir bölütleme modelidir.

DeepLabV3+ temelinde "Genişletilmiş Evrişim" (dilated convolution) ve "Atrous Spatial Pyramid Pooling" yani kısaca ASPP mekanizması içermektedir. Bu mimari, modele derinlik bilgisi sağlayarak daha geniş bir anlam bağlamını anlamaya yardımcı olmaktadır. Genişletilmiş evrişim katmanı, diğer evrişim katmanlarının yerel bilgiye ek olarak daha geniş bir görünüm açısına sahip olmasını sağlamaktadır. Böylece modelin farklı ölçeklerdeki nesneleri algılaması arttırılmaktadır. ASPP mekanizması ise; görüntüdeki farklı ölçeklerdeki nesneleri ve bağlamları yakalayarak bölütleme kalitesini arttırmaktadır. Bu mekanizma farklı genişleme oranlarına sahip birden fazla evrişim katmanını bir arada kullanarak, modelin farklı ölçekteki duyarlılığını arttırmaktadır. DeepLabV3+ modeli ayrıca bir de "decode" adımı içermektedir. Bu adımda ince ayrıntılar ve tespit edilen nesnelerin sınırları daha net bir şekilde belirlenmektedir.

DeepLabV3+ modeli farklı veri setlerinde (örneğin, PASCAL VOC, CityScapes) oldukça rekabetçi sonuçlar vermektedir. Karmaşık sürüş sahnelerinde bile çok iyi performans sergilemektedir. Farklı boyutlardaki nesneleri doğru bir şekilde ayırt edebilme yeteneği sayesinde ölçeklenebilir olup, modellerin tasarımında esnek bir yapı sunmaktadır. DeepLabV3+'ın mimarisi çok hızlı çıktı verebildiği için gerçek zamanlı uygulamalar olan otonom sistemler, robotik uygulamalar, şehir planlaması, trafik analizi ve arttırılmış gerçeklik uygulamalarında sıklıkla tercih edilmektedir. Modelin "decode" aşaması sayesinde, nesne kenarları ve ince ayrıntılar daha net belirlenmektedir. Böylece yüksek kaliteli görüntü bölütlemesi gerektiren otonom sürüş ve tıbbi görüntüleme gibi uygulamalarda kullanımı daha kritik hale gelmektedir.

### 5.6. Pytorch ile Anlamsal Bölütleme

PyTorch, Facebook'un yapay zeka araştırma ekibi tarafından geliştirilen açık kaynaklı bir derin öğrenme kütüphanesidir. Modern yapay zeka ve makine öğrenimi projelerinde yaygın olarak kullanılmaktadır. Bu tezde Python dili ve PyTorch kütüphanesi kullanılmıştır.

PyTorch kütüphanesini kullanabilmek için öncelikle kodun çalıştırılacağı ortama; eğer işletim sistemi Windows ise, komut satırından veya powershell kullanarak, Linux tabanlı bir işletim sistemi ise terminal üzerinden aşağıdaki komut ile kurulması gerekmektedir:

*pip install torch torchvision*

Kurulumdan sonra kodun içerisine bu kütüphaneleri aşağıdaki gibi çağırmak gerekmektedir:

```
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms
from tqdm import tqdm
```

Veri setini yüklemek ve ön işleme almak için “torchvision” sınıfından “transforms” metodu aşağıdaki gibi kullanılmaktadır:

```
#veri ön işleme aşaması
transform = transforms.Compose([
transforms.Resize((256, 256)), # görüntü boyutları ayarlama
transforms,ToTensor(), # görüntüyü tensöre dönüştürme
])
#Eğitim ve doğrulama veri setleri
train_dataset = datasets.Cityscapes(root=’path_to_cityscapes’, split=’train’,
mode=’fine’, target_type=’semantic’, transform=transform)
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=8, shuffle=True)
```

Veri seti hazırlandıktan sonra tezde kullanılacak DeepLabV3+ modelinin yapısı hazırlanmalıdır. Bunun için öncelikle modelin yapısına uygun olacak şekilde ASPP (Atrous Spatial Pyramid Pooling) sınıfı tanımlanmalı, sonra model için yeni bir sınıf oluşturulmalıdır:

```
class _AtrousSpatialPyramidPoolingModule(nn.Module):
    """
    operations performed:
    1x1 x depth
    3x3 x depth dilation 6
    3x3 x depth dilation 12
    3x3 x depth dilation 18
    image pooling
    concatenate all together
    Final 1x1 conv
    """
    def __init__(self, in_dim, reduction_dim=256, output_stride=16, rates=(6, 12, 18)):
        super(_AtrousSpatialPyramidPoolingModule, self).__init__()
        # Check if we are using distributed BN and use the nn from encoding.nn
        # library rather than using standard pytorch.nn
        if output_stride == 8:
            rates = [2 * r for r in rates]
        elif output_stride == 16:
            pass
        else:
            raise 'output stride of {} not supported'.format(output_stride)
        self.features = []
        # 1x1
        self.features.append(
            nn.Sequential(nn.Conv2d(in_dim, reduction_dim, kernel_size=1, bias=False),
                          Norm2d(reduction_dim), nn.ReLU(inplace=True)))
        # other rates
        for r in rates:
            self.features.append(nn.Sequential(
```

```
            nn.Conv2d(in_dim, reduction_dim, kernel_size=3,
                    dilation=r, padding=r, bias=False),
            Norm2d(reduction_dim),
            nn.ReLU(inplace=True)
        ))
    self.features = torch.nn.ModuleList(self.features)
    # img level features
    self.img_pooling = nn.AdaptiveAvgPool2d(1)
    self.img_conv = nn.Sequential(
        nn.Conv2d(in_dim, reduction_dim, kernel_size=1, bias=False),
        Norm2d(reduction_dim), nn.ReLU(inplace=True))
def forward(self, x):
    x_size = x.size()
    img_features = self.img_pooling(x)
    img_features = self.img_conv(img_features)
    img_features = Upsample(img_features, x_size[2:])
    out = img_features
    for f in self.features:
        y = f(x)
        out = torch.cat((out, y), 1)
    return out
```

DeepLabV3+ modelinin yapısı aşağıda verilmiştir:

```
class DeepWV3Plus(nn.Module):
    """
    Wide_resnet version of DeepLabV3
    mod1
    pool2
    mod2 str2
    pool3
    mod3-7
     structure: [3, 3, 6, 3, 1, 1]
     channels = [(128, 128), (256, 256), (512, 512), (512, 1024), (512, 1024, 2048),
```

```
            (1024, 2048, 4096)]
    """
    def __init__(self, num_classes, trunk='WideResnet38'):
        super(DeepWV3Plus, self).__init__()
            wide_resnet = WiderResNetA2(structure=[3, 3, 6, 3, 1, 1], classes=1000,
dilation=True)
        wide_resnet = torch.nn.DataParallel(wide_resnet)
        wide_resnet = wide_resnet.module
        self.mod1 = wide_resnet.mod1
        self.mod2 = wide_resnet.mod2
        self.mod3 = wide_resnet.mod3
        self.mod4 = wide_resnet.mod4
        self.mod5 = wide_resnet.mod5
        self.mod6 = wide_resnet.mod6
        self.mod7 = wide_resnet.mod7
        self.pool2 = wide_resnet.pool2
        self.pool3 = wide_resnet.pool3
        del wide_resnet
        self.residual_block = ResidualAnomalyBlock()
        self.aspp = _AtrousSpatialPyramidPoolingModule(4096, 256, output_stride=8)
        self.bot_fine = nn.Conv2d(128, 48, kernel_size=1, bias=False)
        self.bot_aspp = nn.Conv2d(1280, 256, kernel_size=1, bias=False)
        self.final = nn.Sequential(
            nn.Conv2d(256 + 48, 256, kernel_size=3, padding=1, bias=False),
            Norm2d(256),
            nn.ReLU(inplace=True),
            nn.Conv2d(256, 256, kernel_size=3, padding=1, bias=False),
            Norm2d(256),
            nn.ReLU(inplace=True),
            nn.Conv2d(256, num_classes, kernel_size=1, bias=False))
self.train_module_list = ["branch1.residual_block.atten_aspp",
"branch1.residual_block.proj_head",
                          "branch1.residual_block.bot_atten_aspp"]
    def forward(self, inp):
```

```
if len(inp.shape) == 3:
    inp = inp.unsqueeze(0).cuda()
assert len(inp.shape) == 4  # (B, C, W, H)
x_size = inp.size()
x = self.mod1(inp)
m2 = self.mod2(self.pool2(x))
x = self.mod3(self.pool3(m2))
x = self.mod4(x)
x = self.mod5(x)
x = self.mod6(x)
x_in_ = self.mod7(x)
res_, proj_ = self.residual_block(x_in_, upsample_size=m2.size()[2:])
x_out_ = self.aspp(x_in_)
dec0_up = self.bot_aspp(x_out_)
dec0_fine = self.bot_fine(m2)
dec0_up = Upsample(dec0_up, m2.size()[2:])
dec0 = [dec0_fine, dec0_up]
dec0 = torch.cat(dec0, 1)
dec1 = self.final(dec0)
dec2 = self.final(dec0 + res_)
out1 = Upsample(dec1, x_size[2:])
out2 = Upsample(dec2, x_size[2:])
return out1, out2, proj_
```

DeepLabV3+ modelinde, model gövdesi (Backbone) olarak WiderResNet38 kullanılmıştır. Böylece özellik çıkarımı için güçlü bir temel oluşturulmuştur. Önceden eğitilmiş ağırlıklar ile başlamak, modelin eğitim sürecinde daha hızlı öğrenmesini sağlamaktadır. Evrişim tarafında ASPP kullanılarak farklı ölçeklerden gelen bilgiler birleştirilmektedir. Ayrıca "global average pooling" yöntemi ile elde edilen çıktının buraya eklenmesi ile daha fazla bağlam bilgisi kazanılmaktadır. Burada "Contrast Loss" ve "Energy Loss" olmak üzere iki farklı kayıp fonksiyonu kullanılmıştır. Kullanılan kayıp fonksiyonları aşağıda verilmiştir:

```
class ContrastLoss(nn.Module, ABC):
    def __init__(self, engine, config=None):
```

```
        super(ContrastLoss, self).__init__()
        self.engine = engine
        self.temperature = 0.10
        self.ignore_idx = 255
        self.ood_idx = 254
        self.max_views = 512
    def forward(self, city_proj, city_gt, city_pred, ood_proj, ood_gt, ood_pred):
        city_gt = torch.nn.functional.interpolate(city_gt.unsqueeze(1).float(),
                    size=city_proj.shape[2:],
                   mode='nearest').squeeze().long()
        ood_gt = torch.nn.functional.interpolate(ood_gt.unsqueeze(1).float(),
                    size=ood_proj.shape[2:],
                   mode='nearest').squeeze().long()
        # normalise the embed results
        city_proj = torch.nn.functional.normalize(city_proj, p=2, dim=1)
        ood_proj = torch.nn.functional.normalize(ood_proj, p=2, dim=1)
        # randomly extract embed samples within a batch
        anchor_embeds, anchor_labels, contrs_embeds, contrs_labels =
          self.extraction_samples(city_proj, city_gt, ood_proj, ood_gt)
        # calculate the CoroCL
        loss = self.info_nce(anchors_=anchor_embeds,
                    a_labels_=anchor_labels.unsqueeze(1), contras_=contrs_embeds,
                      c_labels_=contrs_labels.unsqueeze(1)) if anchor_embeds.nelement() > 0
else
                   \  torch.tensor([.0], device=city_proj.device)
        return loss
    def info_nce(self, anchors_, a_labels_, contras_, c_labels_):
        # calculates the binary mask: same category => 1, different categories => 0
        mask = torch.eq(a_labels_, torch.transpose(c_labels_, 0, 1)).float()
        # calculates the dot product
        anchor_dot_contrast = torch.div(torch.matmul(anchors_, torch.transpose(contras_, 0,
          1)), self.temperature)
        # for numerical stability
        logits_max, _ = torch.max(anchor_dot_contrast, dim=1, keepdim=True)
```

```
        logits = anchor_dot_contrast - logits_max.detach()
        # calculates the negative mask
        neg_mask = 1 - mask
            # avoid the self duplicate issue
        mask = mask.fill_diagonal_(0.)
        # sum the negative odot results
        neg_logits = torch.exp(logits) * neg_mask
        neg_logits = neg_logits.sum(1, keepdim=True)
        exp_logits = torch.exp(logits)
        # log_prob -> log(exp(x))-log(exp(x) + exp(y))
        # log_prob -> log{exp(x)/[exp(x)+exp(y)]}
        log_prob = logits - torch.log(exp_logits + neg_logits)
        # calculate the info-nce based on the positive samples (under same categories)
        mean_log_prob_pos = (mask * log_prob).sum(1) / mask.sum(1)
        return - mean_log_prob_pos.mean()
    def extraction_samples(self, city_embd, city_label, ood_embd, ood_label):
        # reformat the matrix
        city_embd = city_embd.flatten(start_dim=2).permute(0, 2, 1)
        city_label = city_label.flatten(start_dim=1)
        ood_embd = ood_embd.flatten(start_dim=2).permute(0, 2, 1)
        ood_label = ood_label.flatten(start_dim=1)
        # define different types of embeds
        city_positive = city_embd[city_label == self.ood_idx]
        city_negative = city_embd[(city_label != self.ood_idx) & (city_label !=
self.ignore_idx)]
        ood_positive = ood_embd[ood_label == self.ood_idx]
        ood_negative = ood_embd[(ood_label != self.ood_idx) & (ood_label !=
self.ignore_idx)]
        # define the number of choice
        sample_num = int(min(self.max_views, city_positive.shape[0], ood_positive.shape[0],
                             city_negative.shape[0], ood_negative.shape[0]))
        # randomly extract the anchor set with {city_ood, city_inlier}
        city_positive_anchor =
            city_positive[torch.randperm(city_positive.shape[0])][:sample_num]
```

```
    city_negative_anchor =
      city_negative[torch.randperm(city_negative.shape[0])][:sample_num]
    anchor_embed = torch.cat([city_positive_anchor, city_negative_anchor], dim=0)
    anchor_label = torch.cat([torch.empty(city_positive_anchor.shape[0],
                        device=city_positive_anchor.device).fill_(1.),
                        torch.empty(city_negative_anchor.shape[0],
                        device=city_negative_anchor.device).fill_(0.)])
    # randomly extract the contras set with {city_ood, city_inlier, coco_ood, coco_inlier}
    city_positive_contras = city_positive_anchor.clone()
    city_negative_contras = city_negative_anchor.clone()
    ood_positive_contras =
      ood_positive[torch.randperm(ood_positive.shape[0])][:sample_num]
    ood_negative_contras =
      ood_negative[torch.randperm(ood_negative.shape[0])][:sample_num]
    contrs_embed = torch.cat([city_positive_contras, city_negative_contras,
                        ood_positive_contras, ood_negative_contras], dim=0)
    contrs_label = torch.cat([torch.empty(city_positive_contras.shape[0],
                        device=city_positive_contras.device).fill_(1.),
                        torch.empty(city_negative_contras.shape[0],
                       device=city_negative_contras.device).fill_(0.),
                       torch.empty(ood_positive_contras.shape[0],
                       device=ood_positive_contras.device).fill_(1.),
                       torch.empty(ood_negative_contras.shape[0],
                       device=ood_negative_contras.device).fill_(0.)])
    return anchor_embed, anchor_label, contrs_embed, contrs_label
```

“EnergyLoss” kayıp fonksiyonu aşağıda verilmiştir:

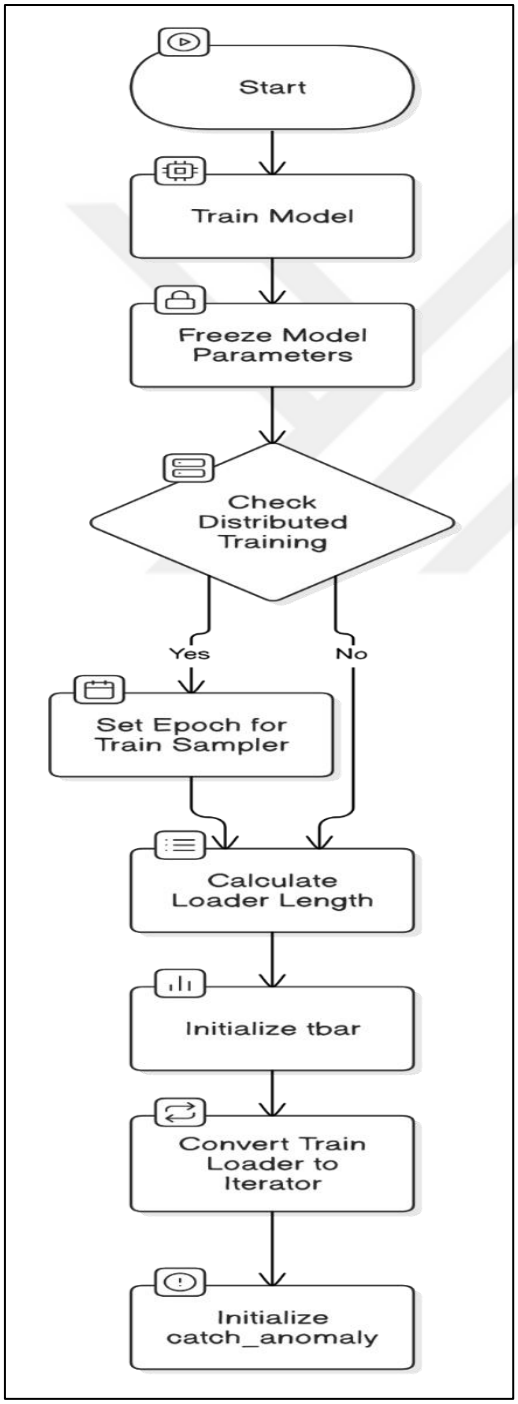
```
def energy_loss(logits, targets, vanilla_logits, out_idx=254, t=1.):
    out_msk = (targets == out_idx)
    void_msk = (targets == 255)
    pseudo_targets = torch.argmax(vanilla_logits, dim=1)
    outlier_msk = (out_msk | void_msk)
    energy = -(t * torch.logsumexp(logits / t, dim=1))
```

```
    assert ~torch.isnan(energy).any(), "nan check 1"
    entropy_part = func.cross_entropy(input=logits, target=pseudo_targets,
reduction='none')[~outlier_msk]
    assert ~torch.isnan(entropy_part).any(), "nan check 2"
    reg = disimilarity_entropy(logits=logits, vanilla_logits=vanilla_logits)[~outlier_msk]
    if torch.sum(out_msk) > 0:
        logits = logits.flatten(start_dim=2).permute(0, 2, 1)
        energy_part = torch.nn.functional.relu(torch.log(torch.sum(torch.exp(logits),
                                                dim=2))[out_msk.flatten(start_dim=1)]).mean()
    else:
        energy_part = torch.tensor([.0], device=targets.device)
    return {"entropy_part": entropy_part.mean(), "reg": reg.mean(), "energy_part":
                energy_part}
def disimilarity_entropy(logits, vanilla_logits,t=1.):
    n_prob = torch.clamp(torch.softmax(vanilla_logits, dim=1), min=1e-7)
    a_prob = torch.clamp(torch.softmax(logits, dim=1), min=1e-7)
    n_entropy = -torch.sum(n_prob * torch.log(n_prob), dim=1) / t
    a_entropy = -torch.sum(a_prob * torch.log(a_prob), dim=1) / t
    entropy_disimilarity = torch.nn.functional.mse_loss(input=a_entropy,
                target=n_entropy, reduction="none")
    assert ~torch.isnan(entropy_disimilarity).any(), print(torch.min(n_entropy),
                torch.max(a_entropy))
    return entropy_disimilarity
```

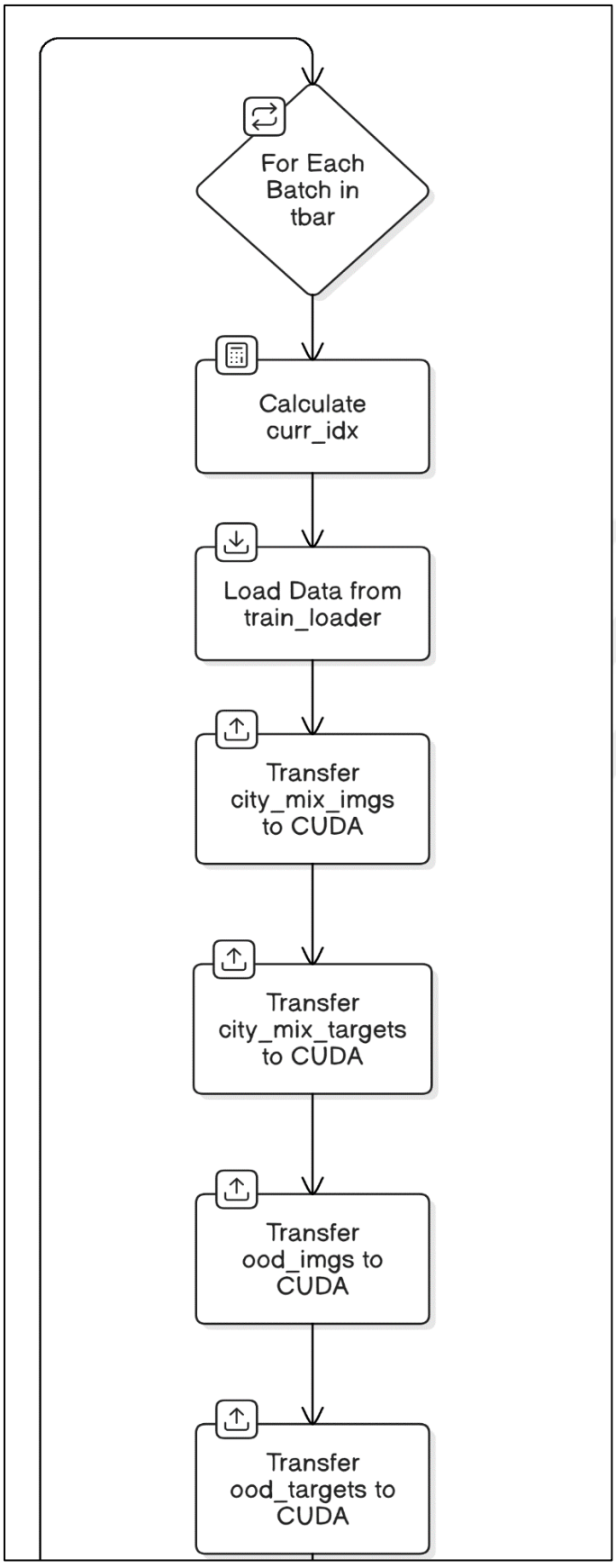
Uzun adı ile “Adaptive Moment Estimation”, kısaca “Adam Optimizer” olarak çağrılan optimizasyon fonksiyonu derin öğrenmede yaygın olarak kullanılmaktadır. Yaygın olarak kullanılmasının ana sebeplerinden birisi hem “Stochastic Gradient Descent” hem de momentum yöntemlerinin avantajlarını birleştirerek daha hızlı ve etkili optimizasyon sağlamasıdır. Uygulaması gayet kolay olup düşük bellek gereksinimi ile çalışmaktadır. Eğitim esnasında “Learning Rate” yani öğrenim katsayısı parametresi üzerinde daha az düzeltme gerektirmektedir. Genellikle “0.0001” öğrenme katsayısı ile iyi çalışmaktadır. Optimizasyon fonksiyonu olarak bu çalışmada da “Adam Optimizer” fonksiyonu “torch” kütüphanesinden çağrılarak hazır olarak kullanılmıştır.

```
"""
selection of the opimizer
"""
optimizer = torch.optim.Adam(params_list)
```

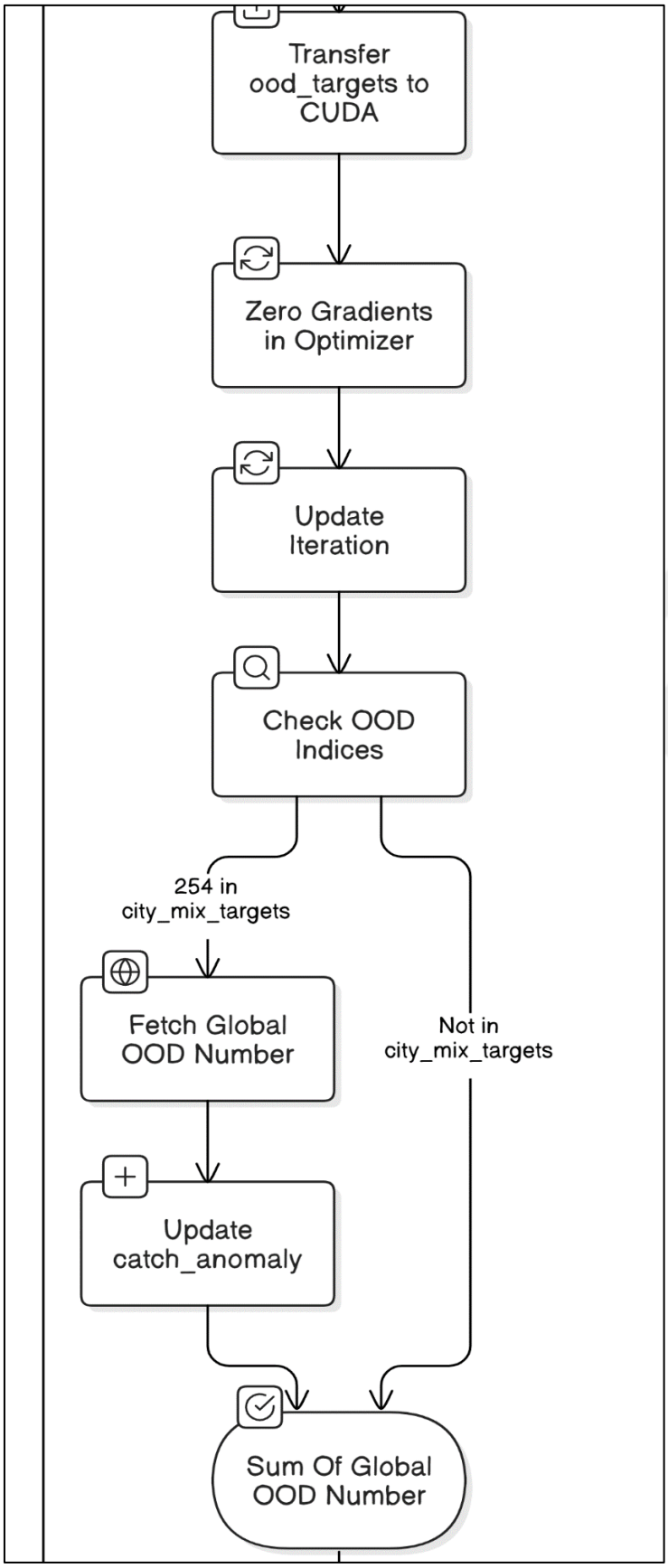
Derin öğrenme modelinin eğitim fonksiyonunun akış şeması aşağıda sırası ile verilmiştir:



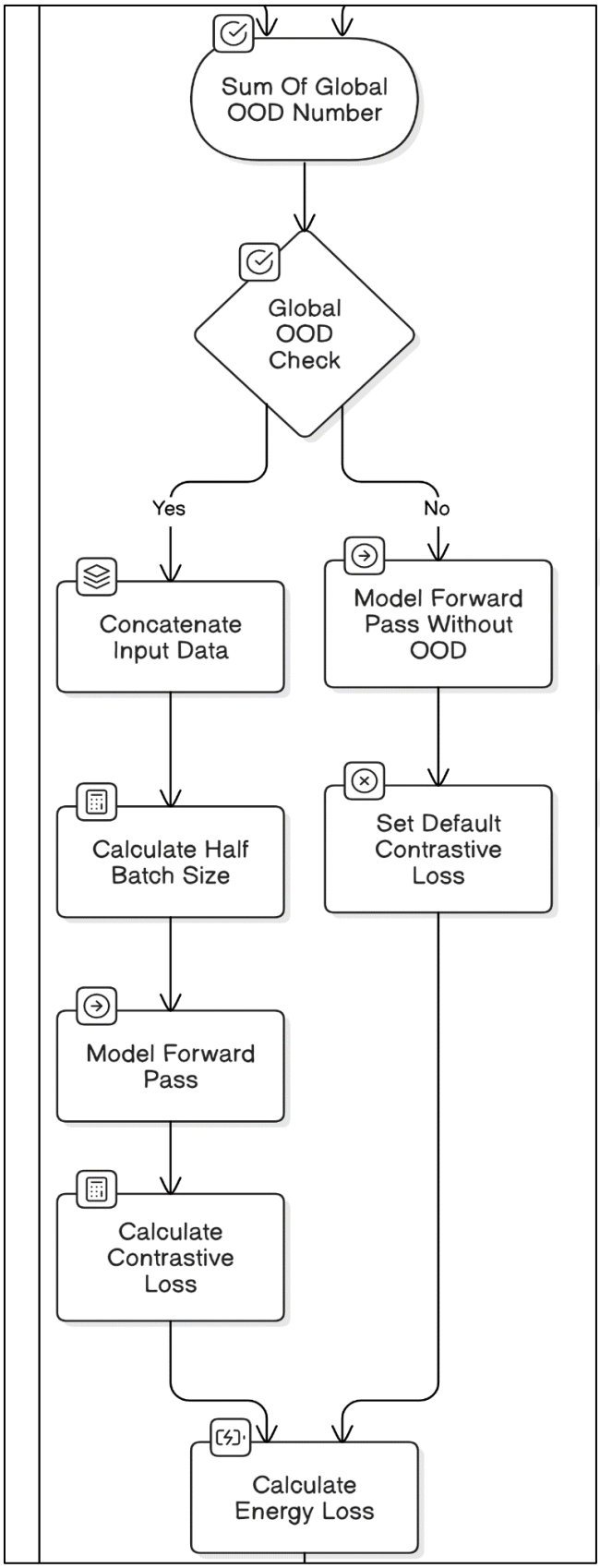

Şekil 5.6. Akış diyagramı – 1



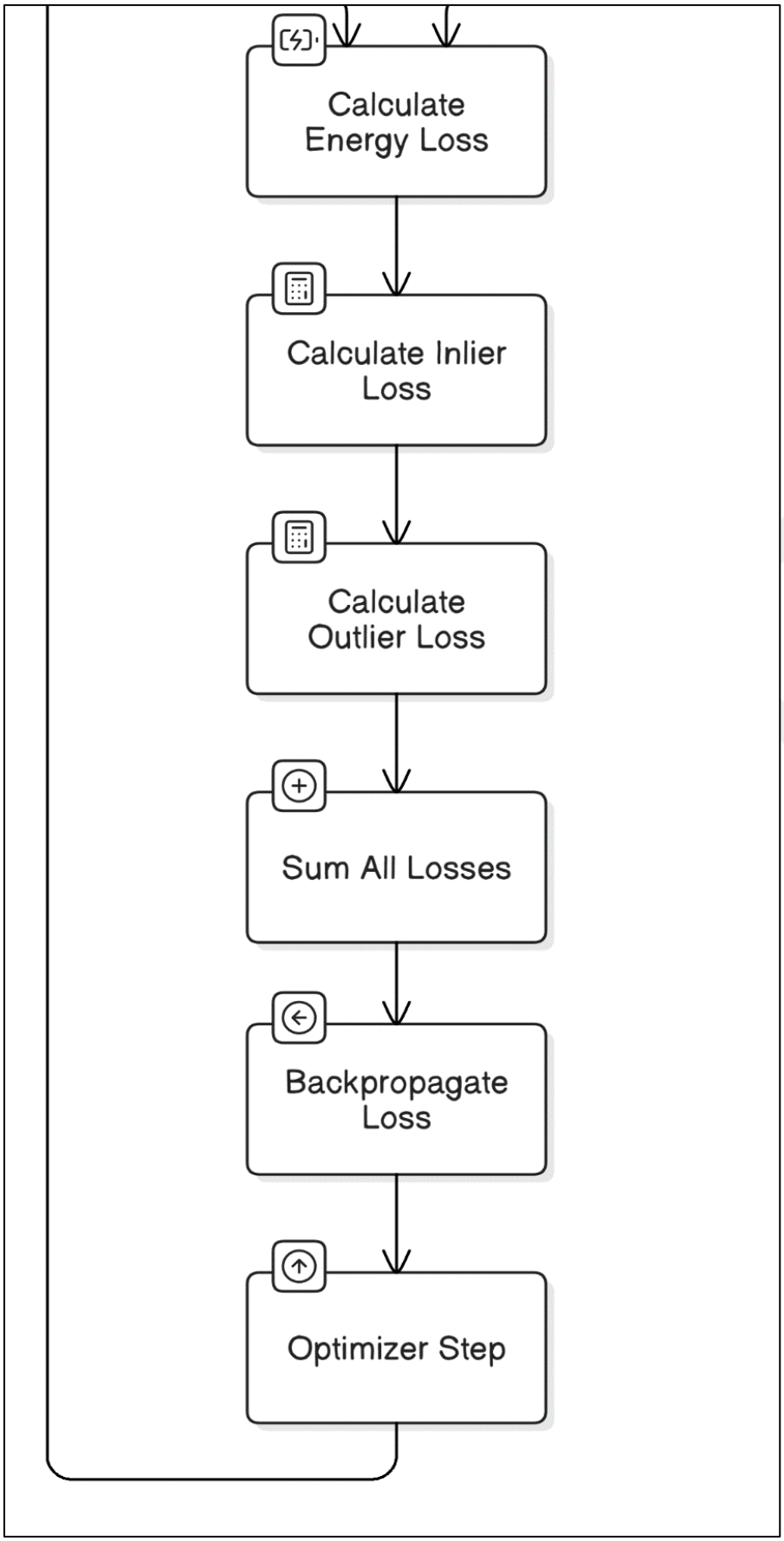

Şekil 5.7. Akış diyagramı – 2




Şekil 5.8. Akış diyagramı – 3




Şekil 5.9. Akış diyagramı – 4




Şekil 5.10. Akış diyagramı – 5

Derin öğrenme modelinin eğitim fonksiyonu ve eğitim döngüsü aşağıda verilmiştir:

```
class Trainer:
    """
    loss_1 -> energy loss; loss_2 -> contrastive loss
    lr_scheduler -> poly;
    """
    def __init__(self, engine, loss1, loss2, tensorboard, validator, lr_scheduler, ckpt_dir, energy_weight):
        self.engine = engine
        self.energy_loss = loss1
        self.contras_loss = loss2
        self.energy_weight = energy_weight
        self.saved_dir = ckpt_dir
        self.validator = validator
        self.tensorboard = tensorboard
        self.lr_scheduler = lr_scheduler
    def train(self, model, epoch, train_sampler, train_loader, optimizer):
        model.train()
        self.freeze_model_parameters(model)
        if self.engine.distributed:
            train_sampler.set_epoch(epoch=epoch)
        loader_len = len(train_loader)
        tbar = range(loader_len)
        train_loader = iter(train_loader)
        catch_anomaly = 0
        for batch_idx in tbar:
            curr_idx = epoch * loader_len + batch_idx
            city_imgs, city_targets, city_mix_imgs, city_mix_targets, ood_imgs, ood_targets = next(train_loader)
            city_mix_imgs, city_mix_targets = city_mix_imgs.cuda(non_blocking=True), city_mix_targets.cuda(non_blocking=True)
```

```
        ood_imgs, ood_targets = ood_imgs.cuda(non_blocking=True),
ood_targets.cuda(non_blocking=True)
        optimizer.zero_grad()
        self.engine.update_iteration(epoch, curr_idx)
        ood_indices = [254 in i for i in city_mix_targets]
        global_ood_num =
self.fetch_global_ood_num(local_ood_num=torch.tensor([sum(ood_indices)]).cuda())
        catch_anomaly += sum(global_ood_num).item()
        if all([i >= 2 for i in global_ood_num]):
            input_data = torch.cat([city_mix_imgs, ood_imgs], dim=0)
            half_batch_size = int(input_data.shape[0] / 2)
            non_residual_logits, residual_logits, projects = model(input_data)
            city_vanilla_logits, city_mix_logits, city_proj =
non_residual_logits[:half_batch_size], residual_logits[:half_batch_size],
projects[:half_batch_size]
            ood_logits, ood_proj = residual_logits[half_batch_size:],
projects[half_batch_size:]
            contras_loss = self.contras_loss(city_proj=city_proj[ood_indices],
                                             city_gt=city_mix_targets[ood_indices],
                                             city_pred=city_mix_logits[ood_indices],
                                             ood_pred=ood_logits[ood_indices],
                                             ood_proj=ood_proj[ood_indices],
                                            ood_gt=ood_targets[ood_indices])
        else:
            city_vanilla_logits, city_mix_logits, _ = model(city_mix_imgs)
            contras_loss = torch.tensor([.0], device = city_mix_logits.device)
            ood_logits = torch.tensor([.0], device = city_mix_logits.device)
            ood_proj, city_proj = torch.tensor([.0], device = city_mix_logits.device),
                                  torch.tensor([.0], device = city_mix_logits.device)
        # energy loss
        loss_dict = self.energy_loss(logits=city_mix_logits,
targets=city_mix_targets.clone(), vanilla_logits=city_vanilla_logits)
        inlier_loss = loss_dict["entropy_part"] + loss_dict["reg"]
```

```
outlier_loss = loss_dict["energy_part"] * self.energy_weight
loss = inlier_loss + outlier_loss + contras_loss
loss.backward()
optimizer.step()
```

# 6. SONUÇ ve ÖNERİLER

Bu bölümde, ilk olarak deney düzeneğini açıklanmıştır. Ardından kullanılan yöntem, mevcut SOTA ile karşılaştırılmaktadır. Daha sonra dağılım içi aykırı değer bölütleme doğruluğu sunulmaktadır.

Eğitim seti

Deneyler, eğitim için 2 975, doğrulama için 500 ve test için 1 525 görüntüden oluşan "CityScapes" kentsel sürüş veri seti kullanılarak oluşturulmaktadır[24]. Bu veri setinde bulunan her resim 2048 x 1024 çözünürlüğe sahiptir. Bu veri setinde anlamsal bölütleme için toplamda 19 farklı sınıf vardır. Önceki çalışmaları takiben, aykırı pozlama (OE) için "COCO" veri seti kullanılmıştır [10,11,13]. "COCO" veri seti, çeşitli bağlamlarda yakalanan nesneleri içerir ve görüntüler yükseklik veya genişlik için en az 480 piksele sahiptir. Bu çalışmada "CityScapes" veri setindeki sınıf türleri ile aynı sınıf türlerine sahip olmayan COCO veri setinin görüntüleri kullanılmıştır. Bu sebeple COCO veri setinden totalde 46 751 adet resim eğitim sırasında dağılım dışı veri olarak alınmıştır [25].

Deneylerde çeşitliliği sağlamak için "KITTI" veri seti ayrıca kullanılmıştır. Bu veri kümesinde şehir içi hem de kırsal ortamlarda çekilmiş görüntüler bulunmaktadır. 5 000 yüksek çözünürlüklü (1280x384 piksel) görüntü verisi vardır. Ek olarak 20 000 görüntü, kaba etiketleme için kullanılmaktadır. Veri seti, değerlendirme için etiketlenmiş 19 ana sınıf bulunmaktadır. Bu ana sınıflar, otomobiller, yaya, bisiklet, kamyon, otobüs, yol işaretleri, ağaçlar, binalar gibi yaygın olarak karşılaşılan nesnelerdir. "CityScapes" kentsel sürüş veri seti ile sınıflar anlamında benzerlik göstermektedir. Bu veri setinin kullanılması; modelin eğitimden bağımsız olarak farklı veri setleri ile başarılı bir şekilde çalıştığının ispatlanması açısından çok önemlidir.

Doğrulama setleri

Eğitilen model birden fazla veri seti ile doğrulanarak değerlendirilmiştir. "Fishyscapes" veri seti, iki doğrulama alt kümesi ile anormallik tespiti için yüksek çözünürlüklü görüntülere sahiptir: "FS-L&F" ve "FS-Static". "FS-L&F" veri seti, "LostAndFound" veri

kümesinden özel etiketlere sahip 1000 görüntüden oluşur ve "FSStatic" veri seti ise, sentetik anormal nesneler içeren 30 görüntü içerir.

"RoadAnomaly" veri seti, internetten 60 görüntü ile gerçek dünyadaki anormal nesneleri içeren başka bir büyük anomali doğrulama kümesidir. "Segment-Me-If-You-Can" (SMIYC) veri seti, farklı bağlamlarda büyük anomali nesneleri içeren "AnomalyTrack" veri kümesi ve ilerideki yolda küçük engeller içeren "ObstacleTrack" veri kümesi olmak üzere iki alt kümeye sahiptir[26]. "AnomalyTrack" veri kümesi ve "ObstacleTrack" veri kümesinde sırasıyla doğrulama için 10 ve 30 görüntü, çevrimiçi test için ise 100 ve 327 görüntü bulunmaktadır. Ayrıca, Chan ve arkadaşları "LostAndFound" veri seti çalışmasından yanlış etiketlenmiş normal kareleri elemiş ve gerçek dünya sürüş sahnesi durumlarını simüle etmek için "SMIYC-L&F" veri setini önermiştir [1, 26].

Uygulama detayları

Piksel bazlı dağılım dışı anomali tespit yöntemi için dağılım içi aykırı değer (CityScapes gibi [8]) veri seti kullanılmıştır. $V^{in} = \{(X_i^{in}, Y_i^{in})\}_i^{|V^{in}|}$ denkleminde, $x^{in} \in X \subset \mathbb{R}^{HxWx3}$ 3 kanallı ve HxW çözünürlüğüne sahip giriş resmi olmaktadır. $y^{in} \in \mathrm{Y}^{in} \subset \{1,.,.,.,C\}^{HxW}$ise C adet kapalı-set sınıf kategorilerinin olduğu anlamsal bölütleme haritasıdır. Dağılım dışı aykırı değer veri seti olarak "COCO" veri seti kullanılmıştır[21]. $V^{Out} = \{(X_i^{out}, Y_i^{Out})\}_i^{|V^{Out}|}$ denkleminde, $x^{out} \in X$ giriş resmini ve $y^{Out} \in \mathrm{Y}^{Out} \subset \{0, P\}^{HxW}$ ise resmin ön ve arka planındaki nesneleri ayırt edebilmek için ikili bir maskedir. Burada dağılım dışı anomali tespitinin dağılım içi aykırı değer sınıflarından ayırt edilebilir olması için P > C olmalıdır. Önceki yapılan anomali tespit yaklaşımlarını da esas alarak yeni geliştirilen anormal bölgeleri ayırt edebilen bir aykırı değer piksellerine maruz kalma metodu kullanılmıştır [11, 13].

Bu yöntem; modelin eğitimi esnasında dağılım dışı giriş verisinden gelen anomali bölge verisini yakalayıp üst üste örtüşme olmadan dağılım içi giriş görüntüsüne kopyalamaktadır. Metodun denklemin aşağıdaki gibidir:

$$x^{oe} = (1 - m) \odot x^{in} + m \odot x^{out} \tag{6.1}$$

$$y^{oe} = (1 - m) \odot y^{in} + m \odot y^{out} \tag{6.2}$$

burada m = I.($Y^{Out}$ = P ) (I(.) indükatör fonksiyondur) değeri dağılım dışı aykırı değer piksellerinde 1 değerine eşit olan, diğer piksellerde ise 0 değerine eşit olan bir ikili değer haritasıdır. Böylece $V^{oe} = \{(X_i^{oe}, Y_i^{oe})\}_{i=1}^{|V^{oe}|}$ denklemi ile ifade edilen ve içeriğine dağılım dışı aykırı değerler karıştırılan yeni veri seti elde edilmektedir.

Kullanılan modelin orijinal hali tamamen evrişimli bir model olup omurgası $g_{\phi_{fcn}} : X \rightarrow Z$ ile gösterilmektedir. Modelin özellik ayırt etme bloğu $g_{\phi_{aspp}} : Z \rightarrow K$ olarak tanımlanmaktadır. Modelin anlamsal bölütleme haritası da $g_{\phi_{seg}} : K \rightarrow [0,1]^{HxWxC}$ olarak gösterilmektedir. Bu modele göre dağılım içi aykırı değerlerin anlamsal bölütlemesi şöyle ifade edilmektedir:

$$\tilde{y} = g_{\phi_{seg}}\left(g_{\phi_{aspp}}\left(g_{\phi_{fcn}}(x)\right)\right) \tag{6.3}$$

Yeniden eğitim yaklaşımları $g_{\phi_{seg}}$'den entropi veya enerji yoluyla anomali modellerini tanımak için anlamsal bölütleme ağını optimize eder. Bu tür yeniden eğitimler anomali belirsizliğini artırarak umut verici sonuçlar verir, ancak aynı zamanda dağılım içi aykırı değerlerin karar sınırlarını kaydırır ve yanlış sınıflandırılmış dağılım içi aykırı değer ön tahminleri üretir. Diğer bir bakışla, dağılım içi aykırı değer anlamsal bölütleme parametrelerini donduran ve anlamsal bölütleme çıktı bilgilerini kullanan yaklaşımlar, dağılım dışı verilerde tespit edilmesi zor olan aykırı değerlerle uğraşırken düşük performans göstermiştir [27].

Burada kullanılan yöntem, FCN katmanları $g_{\phi_{fcn}}$ ile segmentasyon kafası $g_{\phi_{seg}}$ arasına $g_{\phi_L} : Z \rightarrow K$ ile gösterilen harici bir artık modül yerleştirerek yukarıdaki sorunları ele almak üzere tasarlanmıştır. Yanlış pozitif anomali tespiti (yani, belirsiz bölgeler içinde yer alan iç noktalar) OoD maskelerinin üretilmesinin bir yan etkisidir ve bu durum tüm OoD dedektörlerini etkiler; iç nokta bölütleme doğruluğu problemiyle ortogonaldir. Bu durumda harici modülün sonucu ASPP çıktısına eklenir ve bölütleme modülü tarafından çözülür:

$$\tilde{y} = g_{\phi_{seg}}\left(g_{\phi_{aspp}}\left(g_{\phi_{fcn}}(x)\right) + g_{\phi_L}\left(g_{\phi_{fcn}}(x)\right)\right) \quad (6.4)$$

Burada amaç, modelin parametrelerini eğitim sırasında dondurup sadece eklenen modül bloğunda iyileştirme yapmaktır. Böylece dağılım içi sınıflarının karar vermedeki eşik değerleri değişmeyecektir, yani bu eğitim sonucunda kapalı set anlamsal bölütleme tahmininin doğruluk oranı en az derecede etkilenecektir.

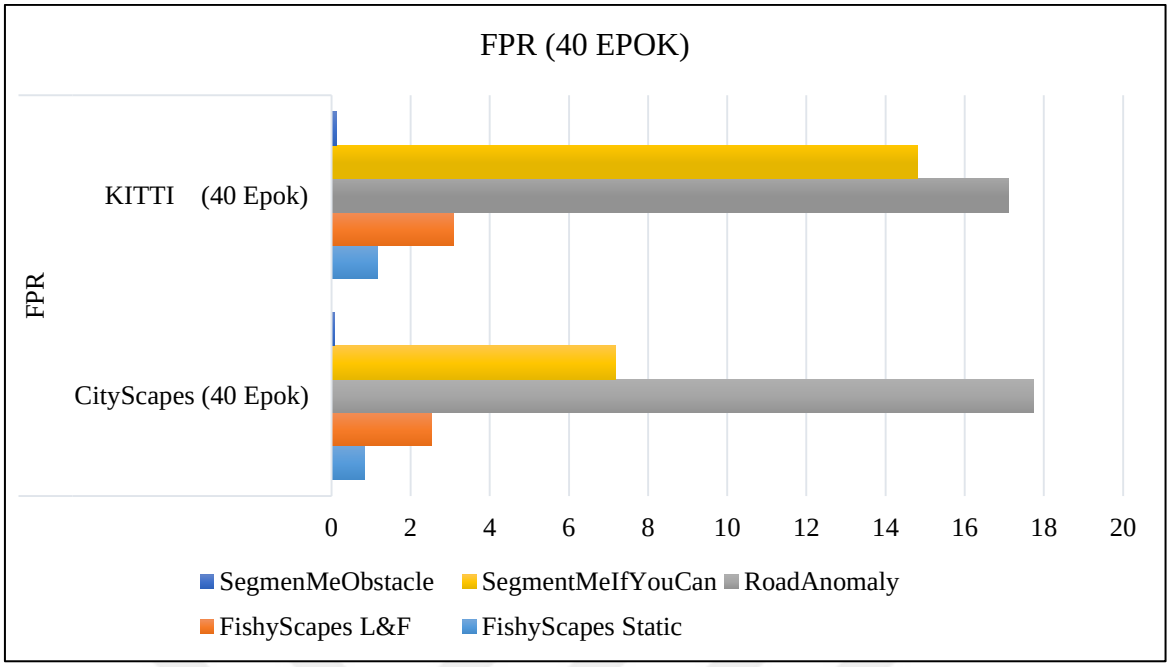
Değerlendirme ölçütleri

Yapılan çalışma sonuçlarını değerlendirmek için "Area Under the Curve - Receiver Operating Characteristic" (AuROC), Area under the Precision-Recall Curve (AuPRC), %95 gerçek pozitif oranında yanlış pozitif oranı (FPR) hesaplanmaktadır [10, 13, 21].

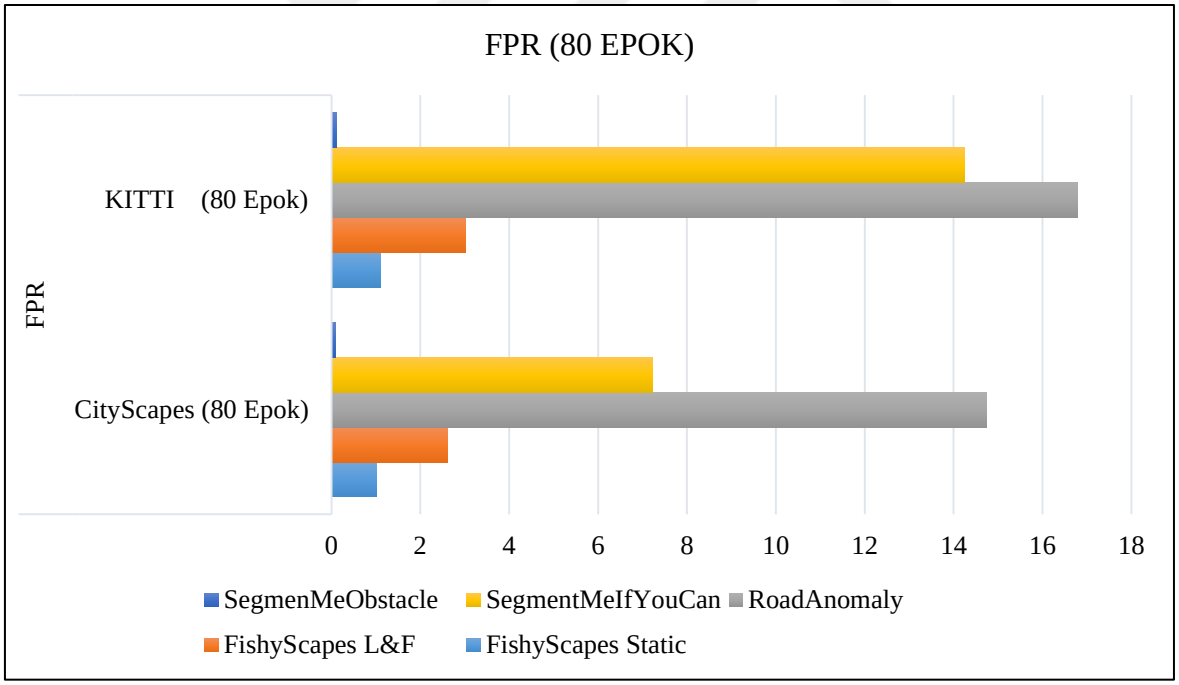
DeepLabV3+ modeline yukarıda anlatılan veri setleri ile önce eğitim sonra da doğrulama fazı uygulanmıştır. Daha sonra elde edilen modele KITTI veri seti ile eğitim fazı tekrar uygulanmıştır. KITTI veri setinde 200 adet eğitim resmi bulunmaktadır. Eğitimden sonra modele doğrulama fazı uygulanmıştır. Testler sonucunda elde edilen veriler ile yapılan FPR analiz sonuçları aşağıda tabloda ve grafikte gösterilmiştir:

Çizelge 6.1. Test sonuçları tablosu FPR metrik değerleri

| Veri Seti | FPR | | | |
|---|---|---|---|---|
| | CityScapes (40 Epok) | KITTI (40 Epok) | CityScapes (80 Epok) | KITTI (80 Epok) |
| FishyScapes Static | 0,85 | 1,16 | 1,01 | 1,1 |
| FishyScapes L&F | 2,52 | 3,09 | 2,61 | 3,01 |
| RoadAnomaly | 17,74 | 17,1 | 14,75 | 16,8 |
| SegmentMeIfYouCan | 7,18 | 14,81 | 7,22 | 14,25 |
| SegmenMeObstacle | 0,09 | 0,12 | 0,1 | 0,11 |




Şekil 6.1. FPR analizi (40 Epok)




Şekil 6.2. FPR analizi (80 Epok)

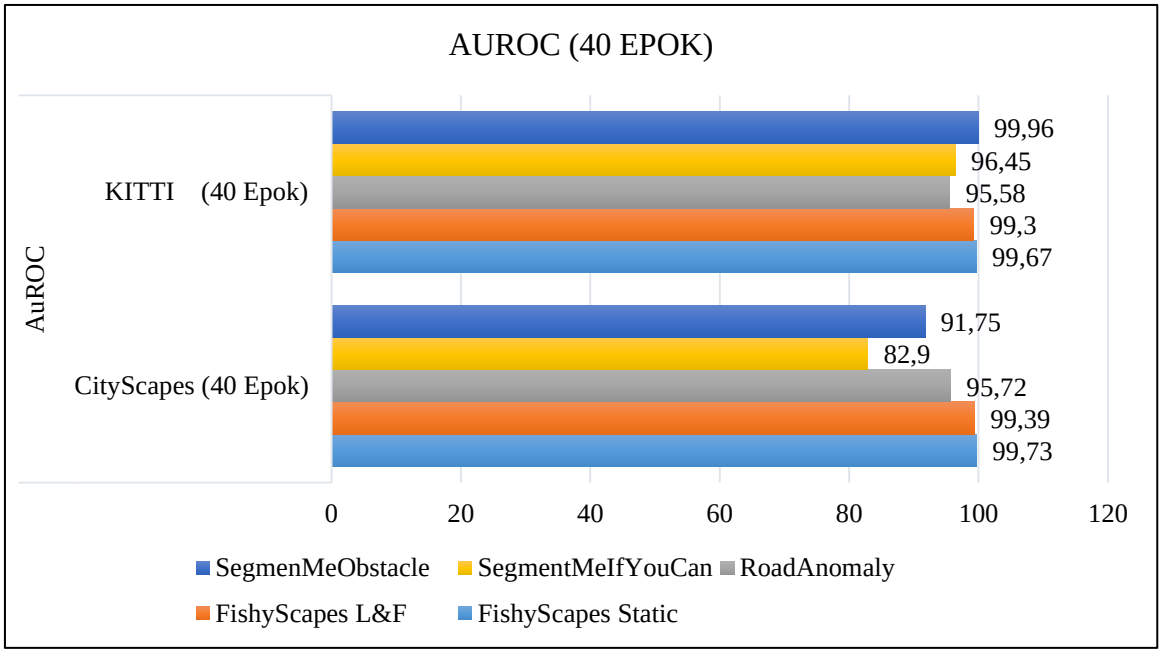
Hem "CityScapes" hem de "KITTI" ile eğitilen model "FishyScapes Static" ve "SegmentMeObstacle" veri setleri ile test edildiğinde düşük FPR değeri göstermektedir. Bu durum, bu veri setlerinin eğitim ve test süreçleri için uygun olduğunu, modelin bu veri setleri ile daha az yanlış pozitif ürettiğini, dolayısıyla daha iyi sonuçlar verdiğini göstermektedir. Diğer taraftan, "RoadAnomaly" veri seti ile yapılan her iki denemede de

model yüksek FPR değeri göstermektedir. Bu durum, modelin bu veri setinde daha fazla yanlış pozitif ürettiğini, dolayısıyla modelin test edilmesi için bu veri setinin yeterince uygun olmadığını göstermektedir. Genel olarak bakıldığında, "CityScapes" veri seti ile elde edilen FPR değerlerinin, "KITTI" veri setine daha düşük olduğu görülmektedir. Bu performans metrik değeri, modelin eğitiminde kullanılan veri seti değişiminde model performansının olumsuz etkilendiğini göstermektedir.

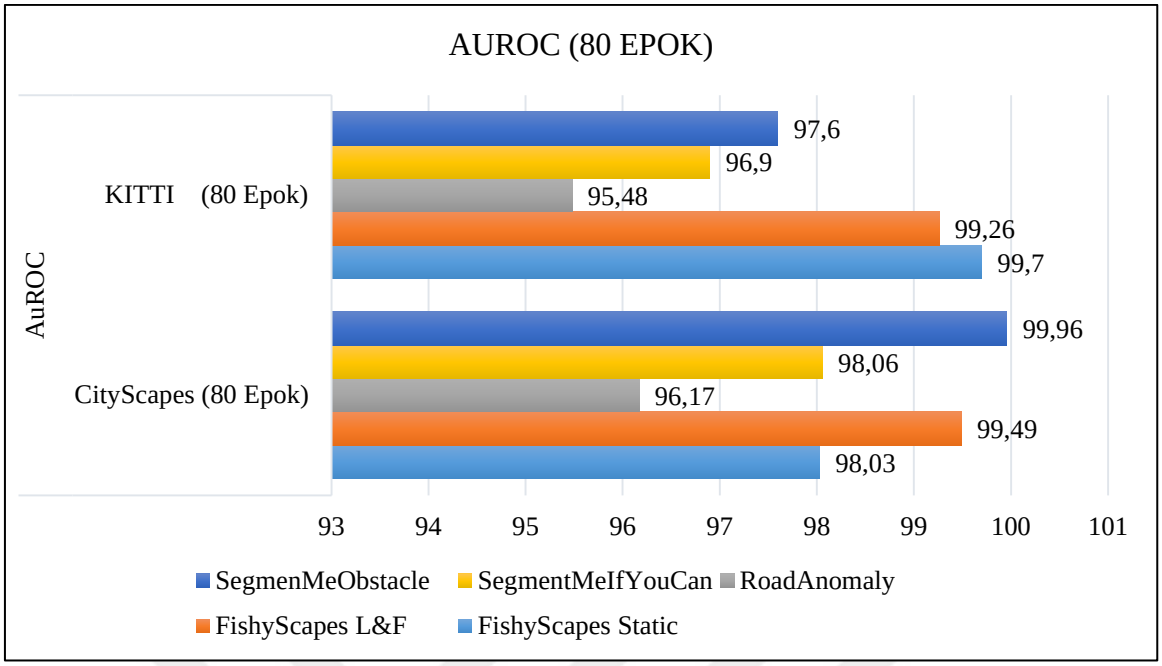
Testler sonucunda elde edilen veriler ile yapılan AuROC analiz sonuçları aşağıda tabloda ve grafikte gösterilmiştir:

Çizelge 6.2. Test sonuçları tablosu AuROC metrik değerleri

| Veri Seti | AuROC | | | |
|---|---|---|---|---|
| | CityScapes (40 Epok) | KITTI (40 Epok) | CityScapes (80 Epok) | KITTI (80 Epok) |
| FishyScapes Static | 99,73 | 99,67 | 98,03 | 99,7 |
| FishyScapes L&F | 99,39 | 99,3 | 99,49 | 99,26 |
| RoadAnomaly | 95,72 | 95,58 | 96,17 | 95,48 |
| SegmentMeIfYouCan | 82,9 | 96,45 | 98,06 | 96,9 |
| SegmenMeObstacle | 91,75 | 99,96 | 99,96 | 97,6 |




Şekil 6.3. AuROC analizi (40 Epok)



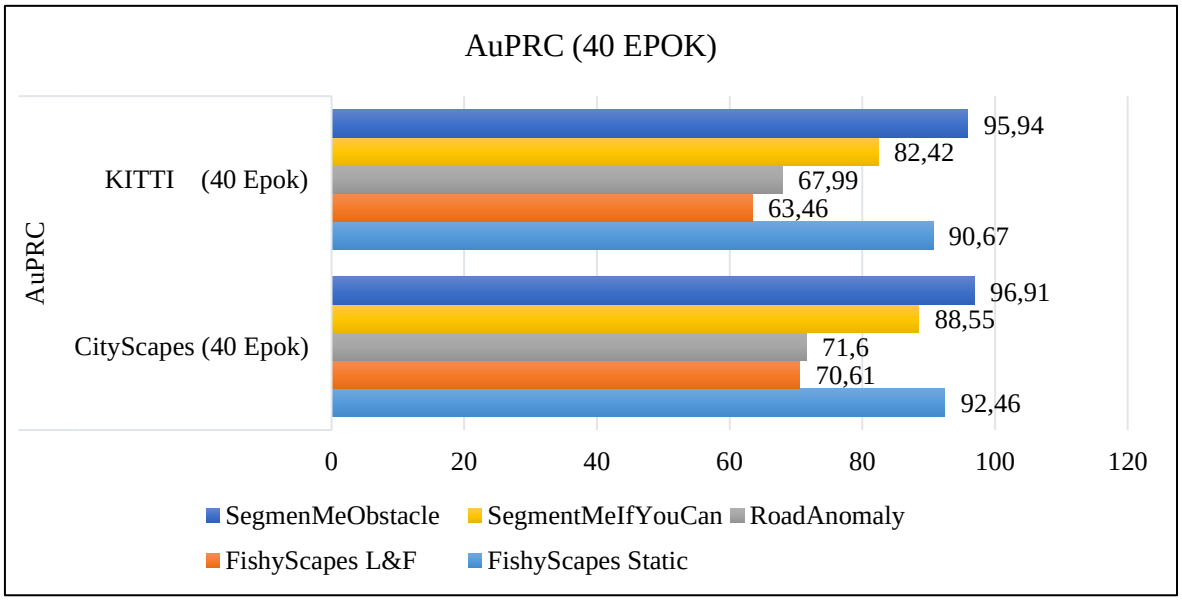

Şekil 6.4. AuROC analizi (80 Epok)

Hem “CityScapes” hem de “KITTI” veri seti ile eğitilen model “FishyScapes Static” ve “FishyScapes L&F” veri setleri ile test edildiğinde yüksek AuROC değeri göstermektedir. Bu durum modelin bu veri setleri ile test edildiğinde çok iyi performans sergilediğini göstermektedir. Eğitim sürecinde epok sayısı 80 olduğunda, model “CityScapes” veri seti ile “KITTI” veri setiyle genel olarak yüksek AuROC değerleri göstermeye devam etmektedir. Örneğin; “KITTI” veri setinde 40 epokta “FishyScapes Static” veri seti için “99.67”, 80 epokta “99.7” değerleri görülmektedir. Bu durum, modelin daha fazla epok ile daha iyi sonuçlar verebileceğini göstermektedir. Genel olarak sonuçlara bakıldığında, modelin “CityScapes” veri seti ile eğitildiğinde daha iyi sonuçlar verdiği gözlemlenmektedir. Ancak özellikle “SegmentMeIfYouCan” ve “SegmentMeObstacle” veri setleri ile yapılan testlerde, “KITTI” veri seti ile eğitildiğinde daha iyi performans verdiği görülmektedir. AuROC metriklerine bakıldığında, modelin performansı kullanılan veri setine ve eğitim süresine bağlı olarak değişiklik göstermektedir.

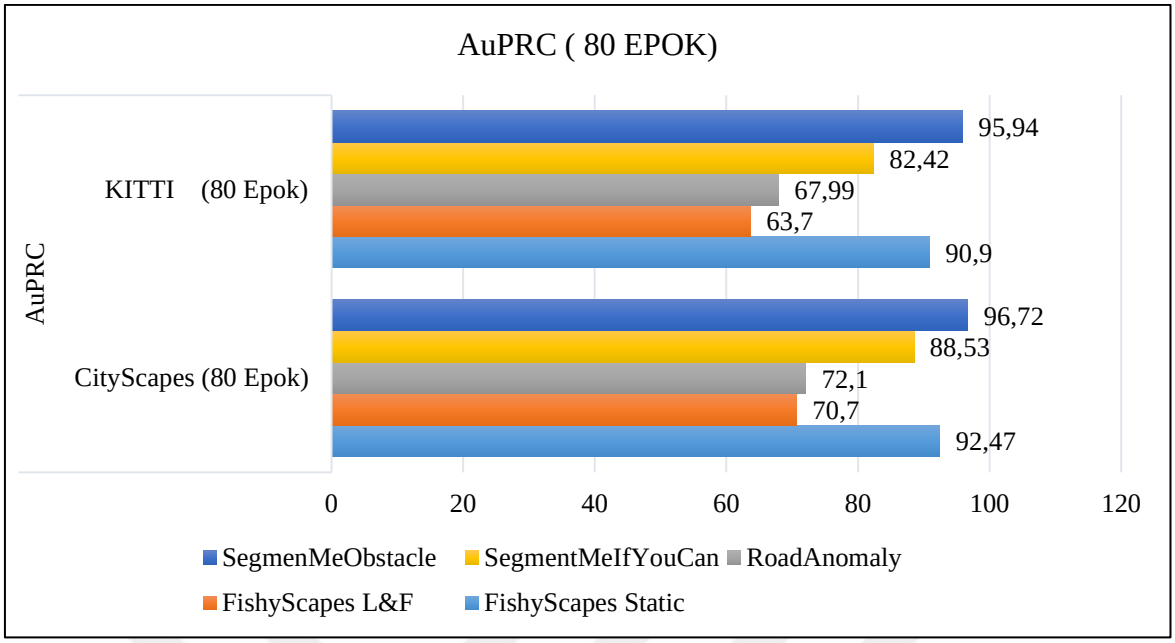
Testler sonucunda elde edilen veriler ile yapılan AuPRC analiz sonuçları aşağıda tabloda ve grafikte gösterilmiştir:

Çizelge 6.3. Test sonuçları tablosu AuPRC metrik değerleri

| Veri Seti | AuPRC | | | |
|---|---|---|---|---|
| | CityScapes (40 Epok) | KITTI (40 Epok) | CityScapes (80 Epok) | KITTI (80 Epok) |
| FishyScapes Static | 92,46 | 90,67 | 92,47 | 90,9 |
| FishyScapes L&F | 70,61 | 63,46 | 70,7 | 63,7 |
| RoadAnomaly | 71,6 | 67,99 | 72,1 | 67,99 |
| SegmentMeIfYouCan | 88,55 | 82,42 | 88,53 | 82,42 |
| SegmenMeObstacle | 96,91 | 95,94 | 96,72 | 95,94 |




Şekil 6.5. AuPRC analizi (80 Epok)




Şekil 6.6. AuPRC analizi (80 Epok)

Tabloda her iki veri seti için 40 ve 80 epok eğitim sonuçları ile elde edilen AuPRC değerleri yer almaktadır. AuPRC özellikle dengesiz ve değişen veri setlerinde modelin performansını değerlendirmek için kullanılan bir metriktir. Bu değer ne kadar yüksekse modelin performansı o kadar iyi olmaktadır. "FishyScapes Static" veri seti ile yapılan hem 40 hem de 80 epok testlerinde "CityScapes" veri seti ile eğitilen modelin, "KITTI" veri seti ile eğitilen modele göre daha yüksek AuPRC değerine sahiptir. Tabloda diğer veri setleri ile yapılan testlere bakıldığında, "CityScapes" ile eğitilen modelin AuPRC değerinin bariz bir şekilde yüksek olduğu görülmektedir. Model "KITTI" veri seti ile eğitildiğinde AuPRC değerleri düşmektedir. Model eğitiminde 40 epok yerine 80 epok yapıldığında, "CityScapes" ile yapılan eğitim sonucunda AuPRC değerlerinde küçük de olsa iyileşmeler gözlemlenebilmektedir. "KITTI" veri seti ile 80 epok testinde aynı şekilde daha düşük AuPRC değerleri elde edilmektedir. Genel olarak bakıldığında, modelin "CityScapes" veri seti ile eğitildiğinde daha iyi sonuçlar verdiği gözlemlenmektedir.

Sonuç olarak bu çalışmada öncelikle bir anlamsal bölütleme modeli bir veri seti üzerinde eğitilip öğrenme süreci tamamlanmıştır. Daha sonra elde edilen model, içerisinde sentetik olarak oluşturulan anomali nesneleri olan farklı bir görüntü veri seti üzerinde eğitilmiştir. Bu şekilde elde edilen modelin değerlendirmesi farklı anomali veri setleri üzerinde yapılmıştır. En yeni teknoloji olarak sunulan piksel seviyesi anomali tespit yöntemleri eğitim esnasında kullanılan veri setine bağımlılığı kaldırdığını iddia etmektedir. Yukarıda

tablolarda verilen performans metrikleri için yapılan teknik analiz sonuçları bu durumun aksini göstermektedir. Test sonuçlarına bakıldığında piksel seviyesi anomali tespit yöntemleri yüksek oranda eğitim veri setine bağlıdır. CityScapes, KITTI, FishyScapes, RoadAnomaly gibi pek çok veri seti üzerinde yapılan deneylerle veri setine bağlılık gösterilmiştir.

Tablolarda verilen FPR, AuROC ve AuPRC metriklerine bakıldığında, sonuçların iyileştirilmesi için modelin “CityScapes” veri seti eğitilmesi gerekmektedir. Ayrıca, epok sayısının 40’dan 80’e çıkartıldığı durumlarda performans metriklerinde iyileşmeler gözlemlenmiştir. Daha uzun süre yapılacak eğitim ve testlerde daha iyi sonuçlar elde edilebilecektir. Test ve doğrulama fazlarında kullanılan veri setlerinin çeşitliliğinin arttırılması modelin performansını, değişen bağlamlarda yapılacak testler açısından iyileştirmektedir.

# KAYNAKLAR


1. Pinggera, P., Ramos, S., Gehrig, S., Franke, U., Rother, C., Mester, R. (2016). *Lost and found: detecting small road hazards for self-driving vehicles.* 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1099–1106.

2. Kaymak, Ç., Uçar, A. (2019). Semantic image segmentation for autonomous driving using fully convolutional networks. 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), 1–8.

3. İnternet: Çetiner, E. (2022, January 6). KERAS ile Derin Öğrenme -2 - Emre Çetiner – *Medium,* URL1: https://emreetiner.medium.com/keras-ile-derin-%C3%B6%C4%9Frenme-2-cb6d7084d1a8, Son Erişim Tarihi: 12 Mart 2025.

4. Jiang, Z. (2019). A novel crop weed recognition method based on transfer learning from vgg16 ımplemented by keras. *IOP Conference Series Materials Science and Engineering*, *677*(3), 032073.

5. Ünal, Z., Kızıldeniz, T., Özden, M., Aktaş, H., Karagöz, Ö. (2023). Defect detection in apple (Granny Smith) with deep learning techniques. *Semantic Scholar*, 13(4), 1119-1129.

6. İnternet: Convolutional Neural Network from Scratch. - DataDrivenInvestor. *Medium*, URL2: https://medium.datadriveninvestor.com/convolutional-neural-network-from-scratch-353b9331047b, Son Erişim Tarihi: 12 Mart 2025.

7. Shelhamer, E., Long, J., Darrell, T. (2016). Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(4), 640–651.

8. Di Biase, G., Blum, H., Siegwart, R., Cadena, C. (2021). *Pixel-wise anomaly detection in complex driving scenes*. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 16913–16922.

9. Hendrycks, D., Mazeika, M., Dietterich, T. (2018). *Deep anomaly detection with outlier exposure.* International Conference on Learning Representations, New Orleans, Louisiana, United States, 1125.

10. Chan, R., Rottmann, M., Gottschalk, H. (2021). *Entropy maximization and meta classification for out-of-distribution detection in semantic segmentation*. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 5108-5117.

11. Lin, T., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C. L. (2014). *Microsoft coco: common objects in context*. European Conference on Computer Vision – ECCV 2014, Zurich, 740–755.

12. Grcić, M., Bevandić, P., Šegvić, S. (2022). *DenseHybrid: hybrid anomaly detection for dense open-set recognition*. European Conference on Computer Vision – ECCV, Tel Aviv, 500–517.

13. Tian, Y., Liu, Y., Pang, G., Liu, F., Chen, Y., Carneiro, G. (2021). *Pixel-wise energy-biased abstention learning for anomaly segmentation on complex urban driving scenes*. European Conference on Computer Vision – ECCV, Tel Aviv, 246-263.

14. Yurtsever, E., Lambert, J., Carballo, A., Takeda, K. (2020). A survey of autonomous driving: common practices and emerging technologies. *IEEE Access, 8(4),* 58443–58469.

15. Liu, W., Wang, X., Owens, J. D., Li, Y. (2020). *Energy-based out-of-distribution detection*. NIPS'20: Proceedings of the 34th International Conference on Neural Information Processing Systems, 21464 – 21475.

16. Li, T., Cao, P., Yuan, Y., Fan, L., Yang, Y., Feris, R., Indyk, P., Katabi, D. (2021). *Targeted supervised contrastive learning for long-tailed recognition*, 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 6908-6918.

17. İnternet: Tek, H. N. (October, 2024). Dijital Dünyada Yapay Zekâ — Bölüm I: Yapay Zekânın Temelleri ve Anahtar Kavramlar. *Medium*, URL3: https://hntek.medium.com/dijital-d%C3%BCnyada-yapay-zeka-b%C3%B6l%C3%BCm-i-yapay-zekan%C4%B1n-temelleri-ve-anahtar-kavramlar-3f75648acba0, Son Erişim Tarihi: 12 Mart 2025.

18. Deng, C., Wang, M., Liu, L., Liu, Y., Jiang, Y. (2021). Extended feature pyramid network for small object detection. *IEEE Transactions on Multimedia*, *24*, 1968–1979.

19. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3213–3223.

20. He, K., Zhang, X., Ren, S., Sun, J. (2015). *Deep residual learning for image recognition*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 770-778.

21. Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A. L. (2017). Deeplab: semantic ımage segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4), 834–848.

22. Kaiming H., Xiangyu Z., Shaoqing R., Jian S. (2016). Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 770–778.

23. Chen, L., Zhu, Y., Papandreou, G., Schroff, F., Adam, H. (2018). *Encoder-Decoder with atrous separable convolution for semantic ımage segmentation.* 15th European Conference, Munich, Germany, 833–851.

24. Blum, H., Sarlin, P., Nieto, J., Siegwart, R., Cadena, C. (2021). The fishyscapes benchmark: measuring blind spots in semantic segmentation. *International Journal of Computer Vision*, *129*(11), 3119–3135.

25. Hong, J., Li, W., Han, J., Zheng, J., Fang, P., Harandi, M., Petersson, L. (2023). Publisher correction: goss: towards generalized open-set semantic segmentation. *The Visual Computer*, *40*(5), 3843–3845.

26. Moukafih, Y., Sbihi, N., Ghogho, M., Smaili, K. (2023). SuperConText: supervised contrastive learning framework for textual representations. *IEEE Access*, 11, 16820–16830.

27. Jung, S., Lee, J., Gwak, D., Choi, S., Choo, J. (2021). *Standardized max logits: a simple yet effective approach for identifying unexpected road obstacles in Urban-Scene segmentation*. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 15405-15414.



*Gazili olmak ayrıcalıktır*