

ANKARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

ÇİFT NORM-UNET TABANLI ANLAMSAL BÖLÜTLEME MİMARİSİNİN
İYİLEŞTİRİLMESİ VE UZAKTAN ALGILAMA VERİLERİ İLE
KULLANILMASI

Fenty James CONTEH

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ANKARA
2024

Her hakkı saklıdır

ÖZET

Yüksek Lisans Tezi

ÇİFT NORM-UNET TABANLI ANLAMSAK BÖLÜTLEME MİMARİSİNİN İYİLEŞTİRİLMESİ VE UZAKTAN ALGILAMA VERİLERİ İLE KULLANILMASI

Fenty James CONTEH

Ankara Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi İrem ÜLKÜ

Semantik segmentasyon, bilgisayarlı görme için kritik öneme sahiptir, çünkü piksel seviyesinde görüntü yorumlaması bilgisi sağlar. Bununla birlikte, semantik segmentasyon modelleri genellikle önemli miktarda etiketli veri gerektirir ki bu da pahalı ve zaman alıcı bir süreçtir. Sonuç olarak araştırmacılar halka açık veri setlerini veya özgürce oluşturulmuş etiketler ile sentetik görüntüleri kullanmayı tercih ederler. Bu araştırma, yüksek boyutluluk ve çeşitlilik nedeniyle karmaşık ve heterojen olan uzaktan algılama verilerine uygulamak için Çift Normalleştirilmiş UNET (Dual Normalization UNet) semantik segmentasyon mimarisinin geliştirilmesine odaklanmaktadır. Toplu ve grup normalizasyonu, düzenleme ve artık bağlantılar ekleyerek DualNorm-UNetV3 modelini iyileştirmekteyiz, böylelikle ağ normalizasyonu için genel görüntü seviyesinde özellikler ve bölgeye özel metrikler eşzamanlı olarak entegre edilmektedir. Daha spesifik olarak, bu yöntem yerel istatistiksel verileri kullanarak normallaştırma katmanı içinde öğrenilmiş semantik maskelerden yararlanarak farklı sınıf bölgelerindeki aktivasyonları uyarlamalı olarak değiştirmektedir. Bu yaklaşım, sınırlı etiketli veriler ile bile daha yüksek segmentasyon doğruluğu elde etmektedir. Normalizasyon teknikleri, düzenleme ve artık bağlantıları kullanarak, çeşitli uzaktan algılama veri setleri boyunca alana özgü değişmeyen bilgi yakalamaktadır. Bu da karmaşık ve çeşitli dağılımlara sahip veri setlerinde iyileştirilmiş segmentasyon performansı ile sonuçlanmaktadır. Geleneksel yöntemlere kıyasla önemli gelişmeler gösteren önerilen DualNorm-UNetV3 modeli, çeşitli referans veri setleri üzerinde başarılı bir şekilde test edilmiştir. Sonuçlar, önerilen modelin uzaktan algılama görüntüsü segmentasyonundaki zorluklar için güçlü bir çözüm sunduğuna ve yaklaşımın çeşitli gerçek dünya uygulamaları için geniş kabul görme potansiyeline işaret etmektedir.

Şubat 2024, 85 sayfa

Anahtar Kelimeler: Evrişimli Sinir Ağları, Uzaktan Algılama, Semantik Segmentasyon, Grup Normalizasyon, Toplu Normalizasyon, Düzenleştirme

ABSTRACT

Master Thesis

IMPROVEMENT OF SEMANTIC SEGMENTATION ARCHITECTURE BASED ON DUAL NORM-UNET AND ITS UTILIZATION WITH REMOTE SENSING DATA

Fenty James CONTEH

Ankara University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Asst. Prof. Dr. İrem ÜLKÜ

Semantic segmentation is critical in computer vision, as it provides pixel-level information for image interpretation. However, semantic segmentation models often require substantial labeled data, which is expensive and time-consuming to obtain. As a result, researchers leverage publicly available datasets or synthetic images with freely generated labels. This research focuses on enhancing the Dual Norm-UNet semantic segmentation architecture and applying it to remote sensing data, which is complex and heterogeneous due to high dimensionality/diversity. We improve the DualNorm-UNetV3 model by adding batch/group normalization, regularization and residual connections to simultaneously integrate global image-level characteristics and localized region-specific metrics for network standardization. More specifically, this method utilizes local statistical data to adaptively modify the activations in different class areas by leveraging learned semantic masks within the normalization layer to selectively enhance or suppress activations. This approach achieves higher segmentation accuracy even with limited labeled data. By utilizing normalization techniques, regularization, and residual connections, it captures domain-invariant information across various remote sensing datasets. This results in improved segmentation performance across datasets with complicated and diverse distributions. The proposed DualNorm-UNetV3 model demonstrates significant improvements over traditional approaches when tested on benchmark datasets, indicating its robustness for satellite images segmentation challenges and potential for broader real-world applications.

February 2024, 85 pages

Key Words: Convolutional Neural Networks, Remote Sensing, Semantic Segmentation, Batch Normalization, Group Normalization, regularization

ÖNSÖZ ve TEŞEKKÜRLER

Her şeyden önce, bana verilen nimetler ve fırsatlar için Allah'a derinden minnettarım.

Değerli danışman hocam Dr. İrem Ülkü'ye, akademik yolculuğum boyunca gösterdiği sabırlı rehberlik ve yapıcı geri bildirimler için en içten teşekkürlerimi sunarım. Onun gözetiminde etkili araştırmalar yapmak, benim için paha biçilmez bir öğrenme fırsatı oldu. Bir danışman olarak sergilediği araştırma uzmanlığı ve iyiliği için minnettarım.

İslam Kalkınma Bankası'na (IsDB) lisansüstü çalışmalarımı sürdürmek için bana burs verdiği için içten teşekkürlerimi sunuyorum. Bu burs, bilim ve teknolojinin öncelikli alanlarındaki yetkinliklerimi ilerletmeme olanak tanıdı.

Ailemin, annem ve babam, Bay ve Bayan Conteh ile sevgili kızım Isatu Hawa Conteh'in şartsız sevgisi ve motivasyonu olmasaydı, bunların hiçbirisi mümkün olmazdı. Potansiyelime olan sarsılmaz inançları ve eğitimimi her şeyin önüne koymak için yaptıkları özverili fedakarlıklar, inişler ve çıkışlardaki itici güç oldu.

Ayrıca, en çok ihtiyaç duyduğumda arkadaşlıkları ve tavsiyeleriyle bu deneyimi daha bütüncül hale getiren arkadaşlarıma da teşekkür etmek istiyorum..

Bu ödüllendirici yolculukta doğrudan veya dolaylı olarak desteklerini esirgemeyen herkese sonsuz minnettarım. Karşılaştığım her zorluk ve başarı gelişimimi şekillendirdi ve bunun için her zaman minnettar olacağım.

Fenty James CONTEH
Ankara, Şubat 2024

İÇİNDEKİLER

TEZ ONAY SAYFASI

ETİK.....	i
ÖZET	ii
ABSTRACT	iii
TEŞEKKÜR	iv
KISALTMALAR DİZİNİ.....	viii
ŞEKİLLER DİZİNİ	ix
ÇİZELGELER DİZİNİ	xi
1 . GİRİŞ	1
1.1 Motivasyon.....	1
1.2 Problem Tanımı	2
1.3 Amaç.....	3
1.4 Kapsam	3
1.5 Anahat	3
2. KAYNAK TARAMASI.....	5
2.1 Giriş.....	5
2.2 Makine Öğrenimi	5
2.3 Sinir Ağları	7
2.3.1 Sinir ağı yapısı.....	7
2.3.2 Geri yayılım ve optimizasyon	9
2.3.3 Aktivasyon fonksiyonları.....	10
2.3.3.1 Basamak fonksiyonu (aktivasyon fonksiyonu).....	10
2.3.3.2 Sigmoid fonksiyonu	11
2.3.3.3 Tanh fonksiyonu	11
2.3.3.4 Relu fonksiyonu	12
2.3.3.5 Sızıntılı relu fonksiyonu	12
2.4 Kayıp Fonksiyonu	13
2.4.1 Piksel bazlı kayıp.....	13
2.5 Çekişmeli Kayıp	14
2.6 Derin Öğrenme.....	15
2.6.1 Evrişimli sinir ağları	16
2.6.2 U-Net cnn'ler	19

2.6.2.1 U-Net mimarisi	19
2.6.2.2 U-Net'in uygulamaları	20
2.6.3 Segnet semantik segmentasyon ağı	21
2.7 Ağ Normalizasyonu	23
2.7.1 Normalizasyonu	23
2.7.2 Gelişmiş normalizasyon teknikleri	24
2.7.3 Toplu normalizasyon	24
2.7.4 Grup normalizasyonu	25
3. MATERYAL ve YÖNTEM	27
3.1 Giriş	27
3.2 Geçmiş	28
3.2.1 CNN'lerde normalizasyon	29
3.2.2 Çok spektrumlu görüntü segmentasyonu için derin öğrenmenin kullanılması	29
3.3 Segmentasyon için Kayıp Fonksiyonları	30
3.3.1 Bce kayıp fonksiyonu	31
3.3.2 Logits ile bce kaybı	32
3.3.3 Semantik segmentasyon için bce kaybı	32
3.3.4 bce kaybı ile dualnorm-unetv3	33
3.4 Segmentasyon için Değerlendirme Ölçütleri	33
3.4.1 Küme benzerliği (jaccard indeksi)	34
3.4.2 Jaccard benzerliği ile dualnorm-unetv3 değerlendirmesi	34
3.5 DualNorm_UNetV3 Artıklık Bloğu	36
3.5.1 Bn şubesi	37
3.5.2 Çift normalizasyon ile öğrenme	37
3.5.3 Eğitim hedefi	38
3.6 DualNorm_UNetV3'ün UNet'e Entegrasyonu	39
3.7 Özetle	40
4. ARAŞTIRMA BULGULARI ve SONUÇLAR	42
4.1 Giriş	42
4.2 Veri Kümeleri	42
4.2.1 Dstl veri seti	42
4.2.2 Rit-18 veriseti	43
4.3 Uygulama Ayrıntıları	43
4.4 Donanım	47

4.5 Deneysel Prosedürler ve Sonuçlar	49
4.6 Değerlendirme Ölçütü	52
4.7 Tartışma	54
4.8 Özet.....	55
5. SONUÇLAR ve GELECEK YÖNELİMLER İÇİN POTANSİYEL	56
5.1 Özet ve Katkıları	56
5.2 Sınırlamalar ve Gelecek Çalışmalar.....	57
5.3 Tartışma	57
5.4 Sonuç	58
KAYNAKLAR.....	59
EKLER.....	65
EK 1 Kodlar.....	65
EK 2 DualNorm_UNetV3 Eğitimi ve Parametre Özeti	74
ÖZGEÇMİŞ.....	84

KISALTMALAR DİZİNİ

BN	Batch Normalisation (Toplu Normalizasyonu)
CNN	Convolutional Neural Networks (Evrişimli Sinir Ağları)
DL	Deep Learning (Derin Öğrenme)
GN	Group Normalization (Grup Normalizasyonu)
IN	Instant Normalization (Anlık Normalizasyonu)
LY	Layer Normalization (Katman Normalizasyonu)
ML	Machine Learning (Makine Öğrenimi)
NN	Neural Network (Sinir Ağı)
NLP	Natural Language Processing (Doğal Dil İşleme)
ReLU	Rectified Linear Unit (Doğrusal Birim)
SegNet	Semantic Segmentation Networks (Semantik Segmentasyon Ağları)
SGD	Stochastic Gradient Descent (Stokastik Gradyan İnişi)
SS	Semantic Segmentation (Semantik Segmentasyon)

ŞEKİLLER DİZİNİ

- Şekil 2.1 İkili sınıflandırma problemlerinde, her kategori için Gauss dağılımı olduğunda, etiketlenmemiş verilerin modelin parametre yaklaşımında yardımcı olabileceğini göstermektedir. Görüntü (xiaojin vd., 2005)'den alınmıştır..... 6
- Şekil 2.2 Görüntü, 3 girişli ($w_1 \times 1 + w_2 \times 2 + b$), iki gizli katmanlı ve bir çıkışlı basitleştirilmiş doğrusal bir NN temsilini göstermektedir. Girdiler, NN'nin işlemesi için verilen değerlerdir. Gizli katmanlar, girdiler üzerinde bazı hesaplamalar yapan ve bunları bir sonraki seviyeye geçiren ara katmanlardır. Toplamın (ağırlıklı toplam artı önyargı) sonucu daha sonra f aktivasyon fonksiyonu üzerinden iletilir. Son değer olarak işleminin bir sonucu olarak sinir ağının ürettiği değer (y). 8
- Şekil 2.3 Makine öğrenimi eğitimi için yaygın olarak kullanılan üç aktivasyon fonksiyonunu gösterir ©udacity.com..... 10
- Şekil 2.4 Sızıntılı ReLU fonksiyonu © Deeplearning.ai.....13
- Şekil 2.5 Görüntü bir CNN'nin mimarisini göstermektedir. Bir CNN, giriş verilerini işlemek ve analiz etmek için çalışan birkaç temel katmandan oluşur. Bu katmanlar, ham verileri alan giriş katmanını, öznitelik çıkarımından sorumlu evrişim tabakasını, doğrusal olmayanlık tanıtmak için bir ReLU (örnek olarak) aktivasyon katmanını, mekansal aşağı örnekleme için bir alt örnekleme katmanını ve son olarak sınıflandırma veya regresyon görevleri için yoğun bağlantılı bir katmanı içerir. Bu katmanlar sistematik olarak yığılır ve karmaşık görüntü tanıma ve analizi yapabilen kapsamlı ve sağlam bir evrişimli sinir ağı oluşturur. 17
- Şekil 2.6 En düşük 32x32 piksel çözünürlüğünde bir örnek göstererek U-net mimarisini göstermektedir. Her mavi kutu, üstte belirtilen kanal sayısına sahip çok kanallı bir özellik haritasını temsil eder. Kutunun sol alt köşesinde x-y boyutu verilmiştir. Beyaz kutular özellik haritalarını çoğaltır ve oklar farklı işlemleri belirtir. Ronneberger ve diğerleri.(2015). 20
- Şekil 2.7 SegNet Mimarisinin örnek gösterimi..... 22
- Şekil 2.8 Etiketlenmemiş Alan Adaptasyon için Alan Özgü BN uygulamasını gösterir. Şekil 7, geleneksel BN ile uzmanlaşmış Alan Özgü BN arasında görsel bir karşılaştırma sağlar. Sonuncusunun içindeki iki ayrı branşın tanıtımı dikkat çekicidir, bunlardan her biri kaynak veya hedef alandan gelen verileri işlemek için özel olarak tasarlanmıştır. Bu uyarlanabilir Alan Özgü BN katmanı, Toplu Normalleştirme (Chang vd., 2019) kullanan herhangi bir etiketlenmemiş alan uyarlama ağına sorunsuz entegrasyon yetenekleri sergilemektedir ve Telif Hakkı ©2019, IEEE'ye aittir.. 25
- Şekil 3.1 Çerçevemizin temel yönlerini gösterir - (a) DualNorm_UNetV3 Mimarisi ve Toplu Normalizasyon ve Grup Normalizasyonu ile U-Net'in Eğitim Aşaması (b) Artıklık Blok Tasarımı..... 35

Şekil 3.2 BN ve GN ile DualNorm_UNetV3 Artıklık Bloğu.	36
Şekil 3.3 Yukarı örnekleme modülü akış şeması.	39
Şekil 4.1 DualNorm_UnetV3 modellerimiz tarafından uydu görüntülerinden elde edilen, bir Görüntüyü, etiketi ve DSTL görüntü kümelerinin gerçeklerini gösteren illüstrasyon. Semantik segmentasyon için gerçek.....	43
Şekil 4.2 Önerilen modelin 100 epok boyunca öğrenme ve doğruluk eğrileri..	48
Şekil 4.3 Nitel Değerlendirme. DSTL Verisetinden segmentasyonları görebiliriz, iyileşme olduğunu görebiliriz.....	49
Şekil 4.4 Uygulanan farklı model mimarisi ile gerçekleştirilen tüm deneyler için semantik segmentasyon sonuçlarını sunar. Sonuçlar, DSTL görüntü kümelerinden seçilen 224x224 piksel çözünürlüklü bölgelerde (Y x G) gösterilmektedir. Model adları ilk satır ve sütunda belirtilmiştir. Semantik segmentasyon sonuçlarında, yeşil bölgeler isabetleri gösterirken, pembe bölgeler yanlış alarmarı gösterir..	53
Şekil 4.5 Çeşitli model mimarileriyle gerçekleştirilen tüm deneyler için semantik segmentasyon sonuçlarını, RIT-18 Görüntü Setlerinde seçilen 224x224 piksel çözünürlüklü bölgelerde (Y x G) temsil eder. Birinci satır ve sütun model adlarından oluşur. RIT görüntü seti için uygun yer gerçeği etiketlerin yokluğu nedeniyle modelin tahmin etmesi zor..	54

ÇİZELGELER DİZİNİ

Çizelge 3.1 Bu çalışmanın referans aldığı çok spektrumlu görüntülerin semantik segmentasyonu için ikili normalizasyon ve DA'yı çözmeye çalışan Örnek Akademik makaleler.....	28
Çizelge 4.1 Veri işleme.....	43
Çizelge 4.2 Deney ayarları.....	46
Çizelge 4.3 RIT-18 Veri Kümesi Segmentasyon Sonucu için Ortalama sonuçlar (Jaccard İndeksi olarak)	51
Çizelge 4.4 DSTL Veri Kümesi için ortalama sonuçlar (Jaccard İndeksi olarak)	51
Çizelge 4.5 Her iki küme için elde edilen Optimum Sonuçlar RIT-18 görüntü kümeleri için sırasıyla 4 ve 5 katlama çapraz doğrulama kullanan “Üç Kanal Girişi” sınıfı.	51
Çizelge 4.6 RIT-18 için dört (4) Çapraz Doğrulama kullanılarak Çoklu Deneysel Sonuç	52
Çizelge 4.7 DSTL için dört (4) katlı Çapraz Doğrulama kullanılarak Çoklu Deneysel Sonuç.....	52

1. GİRİŞ

1.1 Motivasyon

Uzaktan algılama görüntülerinin kullanımı, zengin ve çeşitli bilgi yakalama yetenekleri nedeniyle tarım, çevre izleme, hastalık haritalama, güvenlik ve savunma gibi geniş bir yelpazede alan için temel haline gelmiştir. Bu görüntülerdeki nesnelerin ve bölgelerin doğru bir şekilde segmente edilmesi, arazi kaplama sınıflandırması, tarımsal ürünlerde hastalık tespiti ve görüntülerdeki anormalliklerin belirlenmesi gibi uygulamalar için temeldir. Ancak, pratikte uzaktan algılama görüntü segmentasyonuna makine öğrenimi modellerinin uygulanması birkaç önemli zorluk ortaya koymaktadır, bunların en önemlilerinden biri alan kaydırma sorunudur (Park vd., 2019). Bu zorlukların ele alınması ve uzaktan algılama görüntülerindeki segmentasyon modellerinin etkinliğinin artırılması, çeşitli alanlarda tam potansiyellerinin açığa çıkarılması için hayati öneme sahiptir.

Yapay zeka, özellikle derin öğrenme, sağlık, ulaşım, imalat ve iletişim gibi birçok endüstride dönüştürücü bir çağın başlamasına neden olmuştur (Goodfellow vd., 2014). CNN'ler aracılığıyla kolaylaştırılan derin öğrenme, makinelerin insan yeteneklerine benzer şekilde görsel verileri algılamasını ve anlamasını sağlamıştır. Bu çığır açan teknoloji, Görüntü Analizi ve Sınıflandırma, Görüntü ve Video Tanıma, Medya Oluşturma, Doğal Dil İşleme ve Tavsiye Sistemleri gibi birçok gerçek zamanlı görevin yolunu açmıştır.

Bu çalışmanın motivasyonu, çeşitli dağılımlara sahip uzaktan algılama verileri ile eğitim ve test etme amacıyla Çift Normalizasyon yönteminin kullanılmasını sağlamaktır. Uzaktan algılama verileri, genellikle uçak veya uydu gibi uzaktan alınan bir nesne veya alan hakkında toplanan bilgilere atıfta bulunmaktadır. Makine görüşünde kritik bir görev olan anlamsal segmentasyon, görüntülerde ve görsel sahnelerde anlamlı pikselleri düzenleme ve algılama ve çeşitli bilgisayar görüşü uygulamaları için temeldir. Buradaki motivasyon, farklı saha dağılımları içeren çeşitli uzaktan algılama görüntüleri için derin öğrenme modelinin etkinliğini artırmaktır. Bu, modelin direncini artırmak için özelleştirilmiş tekniklere olan ihtiyacı vurgular. Bu çalışma, DualNorm_UnetV3 mimarisini değiştirmek için Batch ve Group Normalizasyon katmanlarının, düzenleme ve

artakalan bağlantının birleşimini analiz eder. Segmentasyonun dağıtılmış uzaktan algılama görüntü setlerindeki trendini değerlendirmek için Kesim Üstü Birleşim (IoU) veya Jaccard Benzerlikleri ve Logitler ile Binari Çapraz Entropi ölçüleri kullanılmıştır. Analizimizde kullanılan temel mimariler U-Net, SegNet ve DualNorm-Unet'tir.

1.2 Problem Tanımı

Semantik segmentasyon, bir girdi görüntüsündeki her görüntü ögesinin belirli bir nesne sınıfına ayrılmasını içeren görüntü analizinin kritik bir yönüdür. Bu piksel tabanlı etiketleme, görüntünün içeriğinin kapsamlı bir yorumunu sağlar ve nesnelerin ve ilgi alanlarının kesin tanımlanmasını kolaylaştırır. Bu araştırma, havadan görüntülerdeki görüntü bölütlemesine odaklanmaktadır. Bu, gözetim ve kentsel alanlarda çeşitli olayların tespiti ve sınıflandırılması için muazzam potansiyele sahiptir.

Havadan görüntülerin gerçek zamanlı segmentasyonu, güvenlik ve trafik yönetimi uygulamalarını kökten değiştirme kapasitesine sahiptir. Bu teknoloji, kalabalık bölgelerde olayların ve vakaların anında tespit edilmesini, bildirilmesini ve analiz edilmesini sağlar, gelişmiş güvenlik önlemlerine ve daha verimli trafik yönetimine yol açar. Havadan görüntülerin doğru ve hızlı bir şekilde segmente edilme yeteneği, daha güvenli ve yenilikçi kentsel çevreler oluşturmaya önemli ölçüde katkıda bulunabilir.

Bu araştırma, derin öğrenmenin, özellikle CNN'lerin hızlı ilerlemeleri ile Uzaktan algılama görüntü segmentasyonunun getirdiği karmaşık zorluklar arasındaki boşluğu kapatmaya çalışmaktadır, açıkça havadan görüntülere odaklanmaktadır. Yenilikçi alan adaptasyonu tekniklerini geliştirerek ve CNN'lerin yeteneklerinden yararlanarak, bu tez Uzaktan algılama görüntülerde anlamsal bölütleme doğruluğunu ve verimliliğini önemli ölçüde artırmayı amaçlamaktadır. Nihai hedef, uzaktan algılama, tarım ve tıbbi görüntüleme gibi kritik uygulamaları güçlendirmek için Uzaktan algılama görüntülerin potansiyelini tam olarak ortaya çıkarmaktır.

1.3 Amaç

Bu tez, derin öğrenmeden gelen Convolutional Neural Networks (CNN'ler), SegNet ve DualNorm_UNet gibi teknikler kullanarak özellikle uzaktan algılama görüntüleri üzerinde odaklanarak görüntü segmentasyonu alanını ileri taşımak amacını taşımaktadır. Bu araştırma, değişen dağılımlara sahip uzaktan algılama verilerini işlemek için bir metodoloji kurmaya çalışmaktadır. Özellikle sağlam bir anlamsal segmentasyon modeli oluşturmaya ilgi duymaktayız. Bunu başararak, çalışma geliştirilen modelin geniş bir yelpazede farklı alan dağılımlarına sahip uzaktan algılama görüntüleri kullanılarak test edilmesini ve değerlendirilmesini sağlamayı amaçlamaktadır. Hedef, anlamsal segmentasyon modellerinin uyum sağlayabilirliğini ve sağlamlığını artırmak ve bu etkinliği farklı uzaktan algılama veri kümeleri üzerinde sağlamak ve görüntülerin segmentasyonu ile ilişkili zorlukların üstesinden gelmek ve bu teknolojiyi tarım, afet yönetimi, çevre izleme, sağlık ve epidemiyoloji gibi farklı alanlara uygulamayı kolaylaştırmaktır.

1.4 Kapsam

Bu araştırma, uzaktan algılama görüntülerinin sağlam segmentasyonu için öğrenme ve Çift Normalizasyon (DualNorm) konularını keşfetmektedir. Anlamsal segmentasyon, bir görüntüye süper kahraman seviyesinde görüş kazandırma gibi bir şeydir, her pikseli ayrı ayrı tanımlayarak farklı nesneleri ve malzemeleri tanımasını sağlar. Buradaki amaç, uydu görüntülerini etkili bir şekilde segmente edebilecek ve çeşitli dağılımlar arasındaki alan geçişinin zorluğunu hafifletebilecek modeller geliştirmek ve iyileştirmektir. Derin öğrenme modellerinin uzaktan algılama görüntü segmentasyon alanına aktarılmasına izin vermek ve yenilikçi alan uyarlama teknikleri geliştirmek için olanak sağlanacaktır. Genel olarak, bu araştırma, tarım, hastalık haritalama ve uzaktan algılama gibi alanlarda pratik senaryolarla görüntü segmentasyonu alanına katkıda bulunacaktır.

1.5 Anahat

Tezin izleyen bölümü şunları içermektedir: İkinci bölümde, problemin tanımı için yapılan literatür taramasında Uzaktan algılama görüntü bölütlemesi, derin öğrenme, anlamsal

bölütleme, havadan görünümün rolü ve etki alanı adaptasyonu teknikleri detaylı olarak incelenmektedir. Üçüncü bölümde, Malzemeler ve Yöntemler kısmında kullanılan veri seti seçimleri, DualNorm-UnetV3 modeli ve anlamsal bölütleme çerçevesi ayrıntılı olarak açıklanmıştır. Dördüncü bölümde, Deney Tasarımı eğitim protokollerini, temel modelleri ve değerlendirme ölçütlerini ana hatlarıyla sunulmaktadır. Beşinci bölüm olan Sonuçlar ve Analiz bölümünde, model karşılaştırmaları, etki alanı uyarlama performansı, gerçek dünya uygulamaları ve segmentasyon sonuçlarının görselleştirilmesine ilişkin kapsamlı bir analiz sunulmaktadır.

Son olarak, sonuç bölümünde elde edilen önemli bulgular, Uzaktan algılama görüntü bölütlemesine sağlanan katkılar ve gelecek araştırma yönelimlerine dair içgörüler özetlenmiştir. Bu tez, Uzaktan algılama görüntü segmentasyonunun sınırlarını aşarak tarım, tıbbi görüntüleme ve uzaktan algılamadan çeşitli alanlarda potansiyelini ortaya çıkarmayı ve daha güvenli, yenilikçi kentsel çevreler sağlamayı vaat etmektedir.

2. KAYNAK TARAMASI

2.1 Giriş

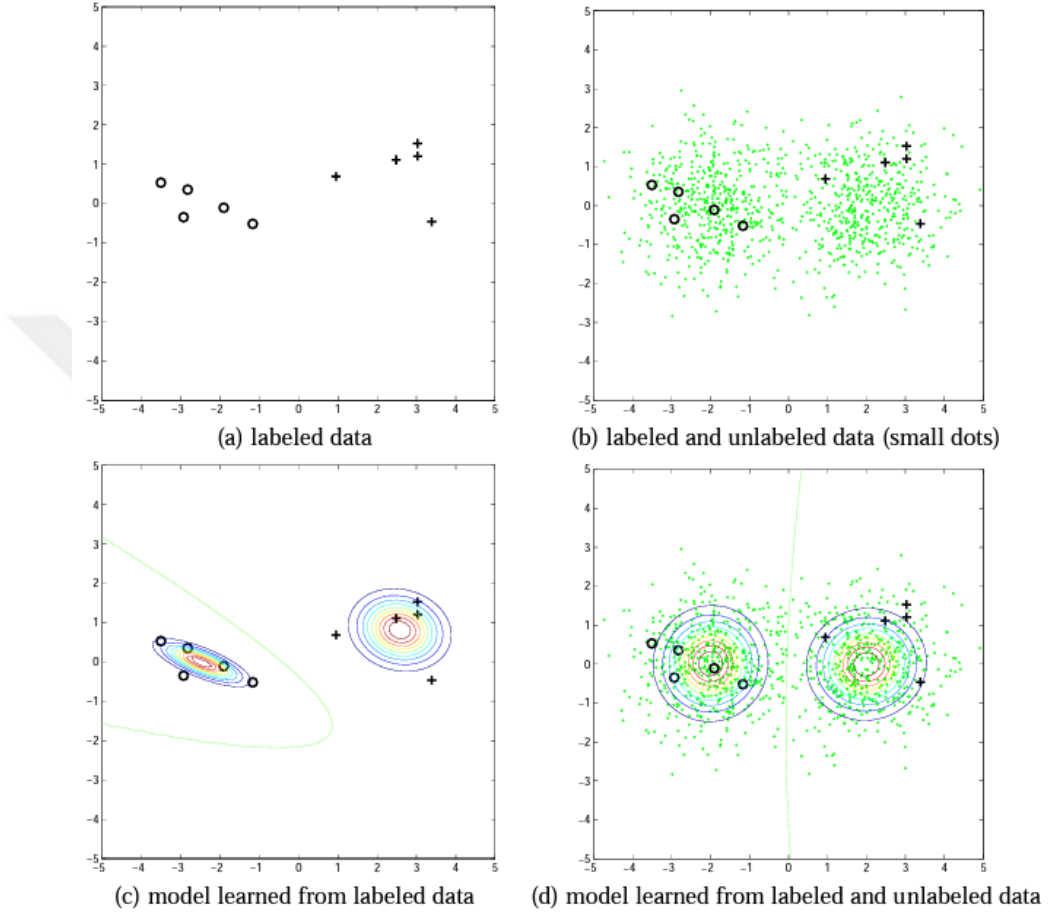
Bu aşama, görüntülerin segmentasyonları için ağ normalizasyonu hakkında güncel bir literatür taramasını sunar. Bu çalışma için önemli olan çeşitli disiplinlerden araştırmaları sergiliyoruz. İlk olarak, bu araştırmanın ayrılmaz bir parçasını oluşturan makine ve derin öğrenmedeki temel kavramları sunuyoruz. Makine öğrenimi, verilerden zaman içinde desenler elde etme ve anlayışlar çıkarma yeteneğine sahip algoritmaların geliştirilmesini kapsar ve açık adım adım programlamaya gerek duymadan yapılır. Temel fayda, sistemlere verileri işleyerek performanslarını aşamalı olarak artırma imkanı sağlamaktır, sadece statik kodlara dayanmak yerine. İkinci bölüm, normalizasyon yöntemlerini kullanarak anlamsal segmentasyona odaklanacaktır. Sonraki bölüm, DualNorm_UNetV3 yöntemini değerlendirmek için kaldırılan semantik segmentasyon tekniklerini ve DualNorm_UNetV3 tarafından alıntı yapılan ilgili çalışmaları tartışacaktır. Bu ilgili çalışmalar U-Net, SegNet ve DualNorm_UNet'i içerir. Son bölüm, bu tezde sunulan yöntemlerde kullanılan metrik öğrenme yaklaşımları hakkında daha fazla detay sağlayacaktır.

2.2 Makine Öğrenimi

ML, bilgisayar bilimleri alanında, veriden bilgi edinme ve açık programlama gerektirmeden performanslarını yinelemeli olarak iyileştirme konusunda yetkin algoritmaların oluşturulmasını araştırmaktadır (Soni vd., 2023). Yapay zekanın bir alt kümesi olarak konumlandırılan ML, makinelerin verilerden özerk olarak içgörüler edinebileceği, örüntüleri ayırt edebileceği ve daha az insan girdisi gerektirerek yargıları yorumlayabileceği temel ilkeye dayanmaktadır (Wang, vd. 2009).

Makine öğrenimi üç temel türe ayrılabilir. İlk tür etiketsiz verilerle sıklıkla, önceden tespit edilmemiş verilerin ve örüntülerin otomatik keşfini içeren gözetimsiz öğrenimdir. İkinci kategori, makine öğrenimi görevlerinde etiketli eğitim verilerinden bir işlevin çıkarımını içeren gözetimli öğrenimdir. Gözetimli öğrenimde, her eğitim verisi örneği, karşılık gelen

istenen çıktı değeriyle birlikte bir girdi nesnesinden oluşur (Mohri vd., 2018). Üçüncü tür, ödül ve cezaların geri bildirim döngüsü yoluyla akıllı araçlarda istenen davranışı teşvik eden ve karar verme süreçlerine yardımcı olabilecek bir yaklaşım olan pekiştirmeli öğrenimdir.



Şekil 2.1 İkili sınıflandırma problemlerinde, her kategori için Gauss dağılımı olduğunda, etiketlenmemiş verilerin modelin parametre yaklaşımında yardımcı olabileceğini göstermektedir. Görüntü (xiaojin vd., 2005)'den alınmıştır

Yarı gözetimli öğrenme algoritması, gözetimli ve gözetimsiz öğrenme paradigmaları arasındaki boşluğu dolduran benzersiz bir konuma sahiptir. Bu yaklaşım makine öğrenimi alanında önemli bir yere sahiptir ve çeşitli alanlarda çok yönlü uygulamalar bulmaktadır. Bu tartışmanın amacı, yarı gözetimli öğrenmenin farklı kategorilerini aydınlatmak, çeşitli metodolojilerine ve uygulamalarına dair içgörüler sunmaktır (Mahesh vd., 2020)

1. **Generatif Modeller:** Yarı gözetimli öğrenme alanında klasik bir yaklaşım olan üretken modeller (xiaojin vd., 2005) tarafından temsil edilir. Bu modeller, $p(x, y) = p(y)p(x|y)$ şeklinde ifade edilen ortak bir dağılım varsayımı altında çalışırlar; burada $p(x|y)$ genellikle bir karışım dağılımı olarak modellenir, örneğin, bir Gauss karışım modeli. Fazlaca etiketlenmemiş veri bulunduğunda, bu modeller farklı karışım bileşenlerini tanımlamada üst düzey performans sergilerler. İdeal olarak, karışım dağılımını tam olarak karakterize etmek için her bir bileşen için yalnızca bir etiketli örnek gereklidir, Şekil 1'de gösterildiği gibi.
2. **Kendi Kendini Eğitme:** Kendi kendini eğitme, yarı gözetimli öğrenme içinde geniş bir şekilde kullanılan bir uygulamayı ifade eder (xiaojin vd., 2005). İlk aşamada, bir sınıflandırıcı sınırlı sayıda karakterize edilmiş veri kullanılarak eğitilir. Ardından, tahmin modeli etiketlenmemiş veriyi sınıflandırmak için uygulanır. Genellikle, en yüksek sınıflandırma kesinliğine sahip veri noktaları ve tahmin edilen sınıflarıyla birlikte eğitim verisine dahil edilir. Bu tahmin modeli daha sonra tekrar eğitilir ve bu tekrarlı uygulama, eğitim sonuçlarını sürekli olarak geliştirmek amacıyla birkaç kez tekrarlanır.

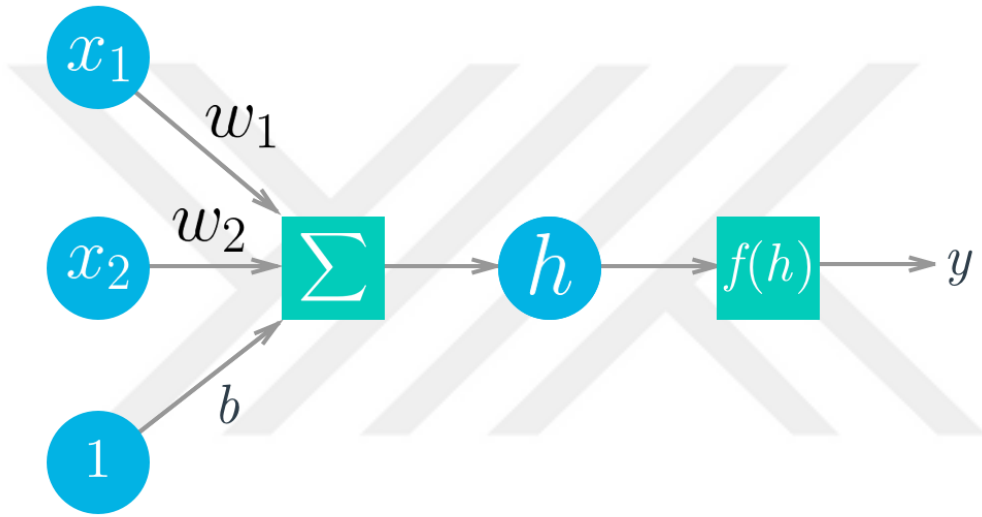
2.3 Sinir Ağları

Sinir ağları temel bir makine öğrenme algoritmasını oluşturur ve derin öğrenmenin köşe taşı olarak yapıy zeka disiplinin en dinamik ve ileri düzey alanlarından biri olarak ön plana çıkmaktadır. Benzersiz ağ mimarisi ve bilgi işleme mekanizmaları yoluyla sinir ağları, otomatik kontrol, karmaşık optimizasyon zorlukları, örüntü algılama, görüntü analizi ve doğal dil işleme dahil olmak üzere çeşitli alanlarda kayda değer ilerlemeleri öncülemiştir. Temel amaç, verilerdeki karmaşık örüntüleri ayırt etme potansiyelidir.

2.3.1 Sinir ağı yapısı

Bir SN, işlevsel bir model olarak görev yapar. Çoğunlukla nöronlar, düğümler veya birimler olarak adlandırılan birçok birbirine bağlı düğümden oluşur. Her düğüm, aktivasyon fonksiyonu olarak bilinen belirli bir çıktı fonksiyonunu cisimleştirir. Düğümler arasındaki ağırlıklar, ağırlıklar olarak adlandırılan ağırlıklı sinyalleri taşır. Bu parametreler

veya ağırlıklar, eğitim prosedürü sırasında dinamik olarak değiştirilir ve girdideki değişikliklerin ağ çıkışını nasıl etkilediğini belirlemede hayati bir rol oynar. Aynı zamanda, aktivasyon fonksiyonlarını kaydırmak için önyargılar uygulanır, böylece ağın girdideki ince detaylara uyum sağlama yeteneğini etkin bir şekilde ayarlar. Ağın nihai çıktısı, ağ yapısı, birbirine bağlantılar, ağırlık değerleri ve seçilen aktivasyon fonksiyonları da dahil olmak üzere bir dizi faktöre bağlıdır. Tipik bir SN'nin mimari düzeni Şekil 2.1'de görsel olarak betimlenmiştir.



Şekil 1.2 Görüntü, 3 girişli ($w_1 x_1 + w_2 x_2 + b$), iki gizli katmanlı ve bir çıkışlı basitleştirilmiş doğrusal bir NN temsilini göstermektedir. Girdiler, NN'nin işlemesi için verilen değerlerdir. Gizli katmanlar, girdiler üzerinde bazı hesaplamalar yapan ve bunları bir sonraki seviyeye geçiren ara katmanlardır. Toplamın (ağırlıklı toplam artı önyargı) sonucu daha sonra f aktivasyon fonksiyonu üzerinden iletilir. Son değer olarak işlemesinin bir sonucu olarak sinir ağının ürettiği değer (y).

$$y = f(w_1 x_1 + w_2 x_2 + b) \quad (1a)$$

$$y = f(\sum_i w_i x_i + b) \quad (1b)$$

Şekil 2.1'in matematiksel gösterimi, makine öğrenimi ve sinir ağlarında kullanılan bir ve iki numaralı denklemlerdir, girdi özelliklerine dayalı olarak tahminler yapmak için kullanılan matematiksel model, burada model eğitim sürecinde farklı girdiler için doğru tahminler yapmak için ağırlıkların (w_1x_1 ve w_2x_2) ve önyargının (b) değerlerini öğrenir. Aktivasyon fonksiyonunun (f) seçimi, kullanılan modelin türüne bağlı olarak değişebilir. Sigmoid, ReLU ve tanh (hiperbolik tanjant) yaygındır.

2.3.2 Geri yayılım ve optimizasyon

Sinir ağlarının eğitimi genellikle geri yayılım algoritmasına, yaygın olarak benimsenmiş bir metodolojiye dayanır. Tipik olarak, bu süreç gradyan inişi gibi optimizasyon teknikleriyle birlikte çiftlenir. Geri yayılım, kayıp fonksiyonunun ağ ağırlıklarına göre gradyanının hesaplanmasına dayanmaktadır. Söz konusu gradyan, ağırlık ayarlamaları için kılavuz güç olarak hizmet eder, sonunda kayıp fonksiyonunu en aza indirir. Geri yayılım için algoritmik prosedürün ayrıntılı hali Algoritma 2.1'de ana hatlarıyla verilmiştir.

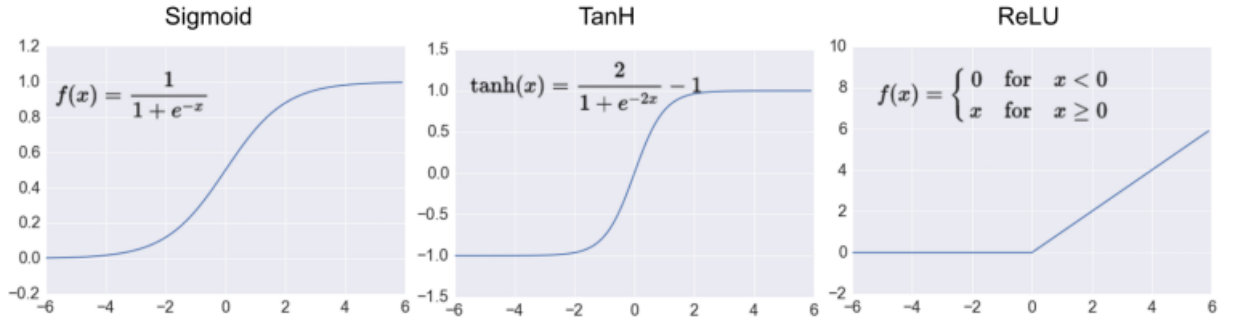
Algorithm 2.1 Back-propagation Algorithm Pseudo-Code

```
# Step 1: Initialize Weights and Biases
initialize_weights_and_biases()
# Step 6: Set the learning rate
learning_rate = 0.001
# Step 5: Repeat Training for Multiple Epochs
for each_epoch in range(num_epochs)
    # Step 2: Forward Pass
    for each training example (x, y):
        # Compute the output of each neuron in each layer
        forward_pass(x)
    # Step 3: Compute Loss
    compute_loss(y)
    # Step 4: Backward Pass (Gradient Computation)
    for each training example (x, y) in reverse order:
        # Compute the gradients for each neuron's weights and biases
        backward_pass(x, y)
    # Step 7: Update Weights and Biases
    update_weights_and_biases(learning_rate)
    # Optional: Step 8 - Apply Regularization (if needed)
    apply_regularization()
# Step 9: Assess the trained model on validation/test data
evaluate_model()
```

Geri yayılım prosedürü iki yaygın zorlukla karşı karşıya: Gradient kaybolması ve patlaması zorlukları. Gradient kaybolması sorunu, geri yayılım işlemi sırasında türevlerin ve parametrelerin çarpımı, özellikle giriş katmanına yakinken, sinir ağı parametrelerinin asgari güncellemesine yol açacak şekilde aşırı küçük bir gradientle sonuçlandığında ortaya çıkar. Tersine, gradyan patlama sorunu, başlangıçta ağırlık değerleri aşırı büyük olduğunda ortaya çıkar, model yakınsamasını engelleyen taşma sorunlarıyla sonuçlanır.

2.3.3 Aktivasyon fonksiyonları

Sinir ağlarının eğitimi, aktivasyon fonksiyonun seçimi çok önemli olmak üzere birçok faktörden etkilenir. Sinir ağları genellikle beş klasik aktivasyon fonksiyonunu kullanır, bunların her biri kendine özgü özelliklere sahiptir. Bunlar, basamak fonksiyonu, sigmoid, tanh, Doğrusal Birim (ReLU) ve Sızıntılı ReLU fonksiyonunu içerir.



Şekil 2.3 Makine öğrenimi eğitimi için yaygın olarak kullanılan üç aktivasyon fonksiyonunu gösterir (©udacity.com)

2.3.3.1 Basamak fonksiyonu (aktivasyon fonksiyonu)

Basamak fonksiyonu, en temel aktivasyon fonksiyonudur. İkili bir sınıflandırıcı istenen sonuç olduğunda durumlarda beğeni toplar. Yine de, çok sınıflandırma problemleri ele alındığında uygunluğunun geliştirilmesi gerekir. Bu sınırlama, basamak fonksiyonunun gradyanının geri yayılım aşaması sırasında sürekli olarak sıfır olmasından kaynaklanır (Sharma vd., 2017). Basamak fonksiyonunun matematiksel ifadesi aşağıda sunulmuştur..

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & otherwise \end{cases} \quad (2)$$

2.3.3.2 Sigmoid fonksiyonu

Sigmoid fonksiyonu, genellikle Lojistik fonksiyon olarak adlandırılır, Şekil 3'te gösterildiği gibi yaygın doğrusal olmayan bir aktivasyon fonksiyonudur. Bu fonksiyon, değerleri (0, 1) aralığına eşler ve ikili sınıflandırma görevlerinde yaygın uygulama bulur (Sharma vd., 2017). Denklem 4, sigmoid fonksiyonun matematiksel temsilini göstermektedir.

$$f(x) = \frac{1}{1 + e^{-x}} \quad (3)$$

Sigmoid fonksiyonu, sürekli türevli bir fonksiyon olarak hizmet eder, (0, 1) ölçeği içinde çıktı verir ve türev hesaplaması için elverişlidir. Bununla birlikte, gradyanın hızla azalma eğiliminde olması nedeniyle geri yayılım süreci sırasında önemli bir sorun ortaya çıkar. Sonuç olarak, bu derin sinir ağlarının eğitim ilerlemesini engelleyebilir ve onları tamamlanmamış hale getirebilir.

2.3.3.3 Tanh fonksiyonu

Tanh fonksiyonu, hiperbolik tanjant fonksiyonu olarak da adlandırılır (Şekil 2.2'de gösterildiği gibi), [-1, 1] aralığında çalışır. Matematiksel tanımı Denklem 2.2'de gösterilmiştir.

$$\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (4)$$

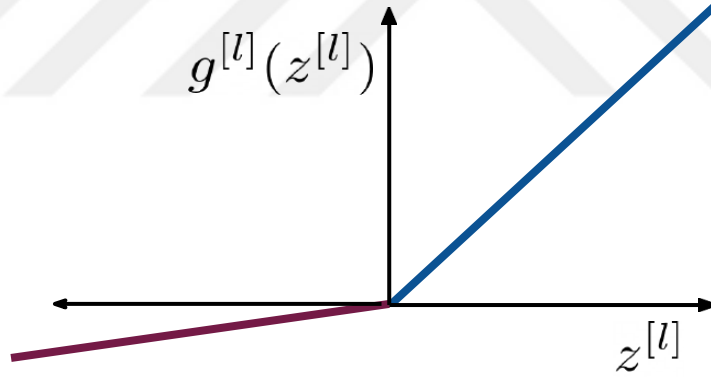
Tanh fonksiyonu, süreklilik ve farklılaştırılabilirlik övünür. Sigmoid fonksiyonu ile karşılaştırıldığında, Tanh fonksiyonu daha dik bir gradyana sahiptir ve belirli bir yöne kısıtlanmaz. Bu özellik, Tanh fonksiyonunu sigmoid fonksiyonuna göre daha çok tercih edilir kılar. Bununla birlikte, önceki araştırmalarda belgelendiği gibi, gradyan kaybolması sorununa karşı savunmasız kalmaya devam etmektedir (Nwankpa vd., 2018).

2.3.3.4 Relu fonksiyonu

Doğrultulmuş Doğrusal Birim (ReLU), Şekil 2.2'de gösterildiği gibi Sinir ağlarında yaygın olarak tercih edilen doğrusal olmayan bir aktivasyon fonksiyonudur. Ana avantajı, tüm nöronların anında aktive edilmediğini belirten sinirsel aktivasyonlarda seyrekliğin tanıtımında yatar. Özünde, bir nöron, doğrusal dönüşümün çıktısı sıfır veya negatif eşitse devre dışı bir durumda kalır. ReLU fonksiyonu için matematiksel denklem aşağıdaki gibi tanımlanmıştır.

$$g^{[l]} = \max(0, z^{[l]}) \quad (5)$$

Bu basit ancak etkili aktivasyon fonksiyonu, gradyan kaybolmasını hafifletme ve eğitim sırasında yakınsamayı hızlandırma yeteneği sayesinde çeşitli sinir ağı mimarilerinde yaygın kullanım bulmuştur.



Şekil 2.4 Sızıntılı ReLU fonksiyonu © Deeplearning.ai

2.3.3.5 Sızıntılı relu fonksiyonu

Sızıntılı ReLU fonksiyonu (Leaky ReLU), Şekil 2.3'te gösterildiği gibi 'ölü ReLU sorunu' olarak bilinen sorunu hafifletmek için bir çözüm olarak tasarlanmıştır. Bu varyantta, tipik olarak 0.01 olarak gösterilen hafif pozitif bir eğim başlatma sırasında tanıtılır. Bu, Leaky ReLU'nun negatif bölgede aktivasyona doğru hafifçe yanlı kalmasını sağlar, bunun yerine aniden sıfırlanmaz. Sızıntılı ReLU fonksiyonu için matematiksel denklem aşağıdaki gibi temsil edilir:

$$g^{[i]}(z^{[i]}) = \max(az^{[i]}, z^{[i]}) \quad (6)$$

Sızıntılı ReLU fonksiyonu, nöronların tepki vermez hale gelme sorununu etkili bir şekilde ele alır ve sinir ağı mimarilerinde eğitim sırasında daha iyi gradyan akışını teşvik etmek için yaygın olarak kullanılır..

2.4 Kayıp Fonksiyonu

Bir makine öğrenimi modeliyle başarı elde etmek için, uygun bir kayıp fonksiyonu seçmek çok önemlidir. Uygun kayıp fonksiyonunu seçmek, modelin ilgili özellikleri önceliklendirmesini ve verilerden çıkarmasını sağlar, böylece optimal ve hızlı yakınsama kolaylaştırılır.

2.4.1 Piksel bazlı kayıp

Piksel bazlı kayıp fonksiyonu, tahmin edilen görüntüdeki ilgili pikseller arasındaki benzeşmezliği, gerçek ile karşılaştırarak değerlendirir. Fonksiyon, üç ana kayıp bileşenini içerir: $L1$ kaybı, $L2$ kaybı ve çapraz-entropi kaybı. Bu kayıp metrikleri, hedef değişken içindeki her piksel çiftini tahmin etmeye uygulanır. Bir $L1$ kaybı veya en küçük mutlak hata, projekte edilen ve gerçek (yer gerçeği) değerler arasındaki tutarsızlığı hesaplar (bkz. Eq7a). Öte yandan, $L2$ kaybı veya En Küçük Kareler Hatası, tahmin edilen değerler ve gerçek değerler arasındaki karesel tutarsızlıkların toplamı olarak hesaplanır (bkz. Denklem. 7b).

$$loss(x, y) = \frac{1}{n} \sum_{i=1}^n |y_i - f(x_i)| \quad (7a)$$

$$loss(x, y) = \frac{1}{n} \sum_{i=1}^n \|y_i - f(x_i)\|^2 \quad (7b)$$

Burada i , i . örneği gösterir, n veri noktalarının genel sayısını temsil eder, ' x ' giriş verisini gösterir, y bu giriş için gerçek etiketi gösterir ve f her giriş x ile model tarafından tahmin edilen ilgili çıktı y arasındaki eşlemeyi sembolize eder. Bir başka deyişle, denklem 1'den

n'ye kadar tüm örnekler i için tahmin edilen çıktılar $f(xi)$ ve gerçek etiketler yi arasındaki kaybı toplar. Amaç, tüm veri seti boyunca bu toplam tutarsızlığı en aza indirmektir.

L1 kaybı göz önünde bulundurulduğunda, kayıp değerine göre türevi sabit kalır. Eğitimin sonraki aşamalarında, kayıp değeri azaldıktan sonra, sabit bir öğrenme oranı, daha yüksek doğruluklara doğru yakınsamanın önüne geçebilecek stabil bir değer etrafında kaymalara yol açabilir. Tersine, L2 kaybı her noktada değişen bir türev sergiler ve minimum değerden ne kadar uzak olunursa, gradient o kadar yüksek olur. Bu koşullar altında iniş optimizasyon tekniğini kullanmak, gradient patlamasına neden olabilir. Bu nedenle, birçok derin öğrenme ağı, model eğitimi için birincil kayıp fonksiyonu olarak ikili çapraz entropi kaybını tercih eder. (Denklem 8'de örneklendiği gibi)

$$H(p, q) = \sum_i P_i \cdot \ln \frac{1}{q_i} = -\sum_i P_i \ln q_i \quad (8)$$

Verilen bağlamda i , i . örneğin indeksini gösterir, p gerçek dağılımı temsil eder ve q sembolü tahmin edilen dağılımı sembolize eder, $H(p, q)$ terimi çapraz entropi olarak adlandırılır. Çapraz entropi, temel olarak iki olasılık dağıtım arasındaki bilgi sapmasını nicellemek için kullanılan Shannon'un bilgi teorisinden türetilmiş temel bir kavramdır. Bir Çapraz entropi Kayıp fonksiyonu, p ve q 'nin benzerliğini değerlendirmek için ölçüt olarak kullanılır. Çapraz-entropi kaybının benzersiz özelliği, tahmin edilen semantik segmentasyon sonuçları ile gerçek veriler arasındaki farkı ölçmedeki uygulanabilirliğini mümkün kılar.

2.5 Çekişmeli Kayıp

İlk olarak (Goodfellow vd.,2014) tarafından tanıtilan standart GAN kayıp fonksiyonu, genellikle min-max kaybı olarak adlandırılır. Denklem 9'da gösterildiği gibi, bu kayıp fonksiyonu iki farklı bileşenden oluşur: eleştirmen kaybı ve üretici kaybı. Bir üretici, aynı kayıp fonksiyonunu en aza indirmeyi amaçlarken; bir eleştirmen onu maksimize etmeyi amaçlar. Standart GAN kayıp fonksiyonunun, üretici açısından doygunluğa ulaşabileceğinin altını çizmek önemlidir. Başka bir deyişle, üreticinin öğrenme kapasitesi ayırt edicinin gerisinde kalırsa, eğitim süreci erken bir şekilde durgunluğa ulaşabilir.

$$\min G \max D V(G, D) = E_{x \sim p_{data}(x)} [\log(D(x))] - E_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (9a)$$

$$\nabla \theta_d = \frac{1}{m} \sum_{i=1}^m [\log D(x^i) + \log(1 - D(G(z)))] \quad (9b)$$

$$\nabla \theta_g = \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z))) \quad (9c)$$

Burada i , i 'inci örneğin dizinini gösterirken, m genel örnek sayısını, x değişkeni gerçek örneklerle karşılık gelirken, z gizli uzaydan gürültülü örnekleri ifade eder. Bu bağlamda, 'gizli uzay' bir SN'nin bir eğitim veri setindeki görüntülerden ne öğrendiğini kapsülleyen geometrik bir temsile atıfta bulunur (Jake Elwes 2022). Özellikle, G üreticiyi, ' D ' ise ayırt ediciyi temsil eder. $G(z)$, rastgele varyasyondan sentetik görüntülerin üretimini ifade eder. Tüm maliyet fonksiyonu, tipik olarak 0 veya 1 değerlerini alan ' D 'nin çıktısından kaynaklanır.

1. Eleştirmen Kaybı: Ayırt edicinin eğitimi sırasında, birincil görevi gerçek ve üretilmiş (sahte) verileri ayırt etmektir. Ayırt edici, gerçek olayları sahte olarak ya da sahte olayları gerçek olarak yanlış sınıflandırır ve cezalandırılır. Modeli etkili bir şekilde eğitmek için eleştirmen, Denklem 9b'de gösterildiği gibi matematiksel denklemi en üst düzeye çıkarmak amaçlamaktadır.
2. Üretici Kaybı: Üreticinin eğitimi, rastgele gürültü örnekleme ve bu gürültüye dayalı olarak bir çıktı üretmeyle başlar. Bu çıktı daha sonra ayırt edici tarafından gerçek ve sahte verileri ayırt etme kapasitesine dayalı olarak 'geçerli' veya 'geçersiz' bir sınıflandırmaya tabi tutulur. Sentezleyicinin kaybı daha sonra ayırt edicinin değerlendirmesine göre hesaplanır. Üretici ayırt ediciyi kandırmayı başarırsa ödül alır; aksi takdirde ceza alır. Üretici, performansını artırmak için eğitim süreci boyunca Denklem 9c'de gösterilen denklemi en aza indirmeye çalışır.

2.6 Derin Öğrenme

Makine öğrenimi ilk ortaya çıktığında, geri yayılım algoritması (Rumelhart vd., 1986), SN'lerin birincil eğitim yaklaşımlarından biri olarak tanıtıldı. Bu süre boyunca, SN'ler nispeten basitti, tipik olarak yalnızca bir gizli katmandan oluşuyordu. Bununla birlikte, tahmin görevlerinin karmaşıklığı arttıkça, sıkı sinir ağları tahmin doğruluğu taleplerini

karşılamakta zorlandı. Bu zorluk, Geoffrey E. Hinton ve Ruslan R. Salakhutdinov'u çoklu gizli katmanlara sahip ağlar ile ilgili iki kritik gözlem yapmaya itti (Hinton vd.,2006).

1. Birden çok gizli katman içeren sinir ağları, özellikleri öğrenmede olağanüstü yetkinlik göstermektedir. Bu ağlar, veri görselleştirme ve sınıflandırma gibi görevler için değerli avantajlar sunarak karmaşık veri özelliklerini öğrenmede mükemmel performans gösterir.
2. Derin sinir ağlarını eğitmenin engelini etkili bir şekilde ele almak, "katmanlı ön eğitim" olarak bilinen bir teknikle verimli bir şekilde ele alınabilir.

Bu makalede sunulan öncü araştırma (Hinton vd., 2006), derin ağ eğitimi sırasında gradyanın yok olması sorununu ele aldı. Bunu, ağ parametrelerini gözetimsiz tahminsel eğitimle başlatarak ve modele gözetimli öğrenme kullanarak ince ayar yaparak başardı, sonuç olarak tamamen eğitilmiş derin bir ağ elde edildi. Bu yenilikçi yaklaşım, karmaşık görevleri ele almada derin öğrenmeye önemli bir üstünlük sağladı.

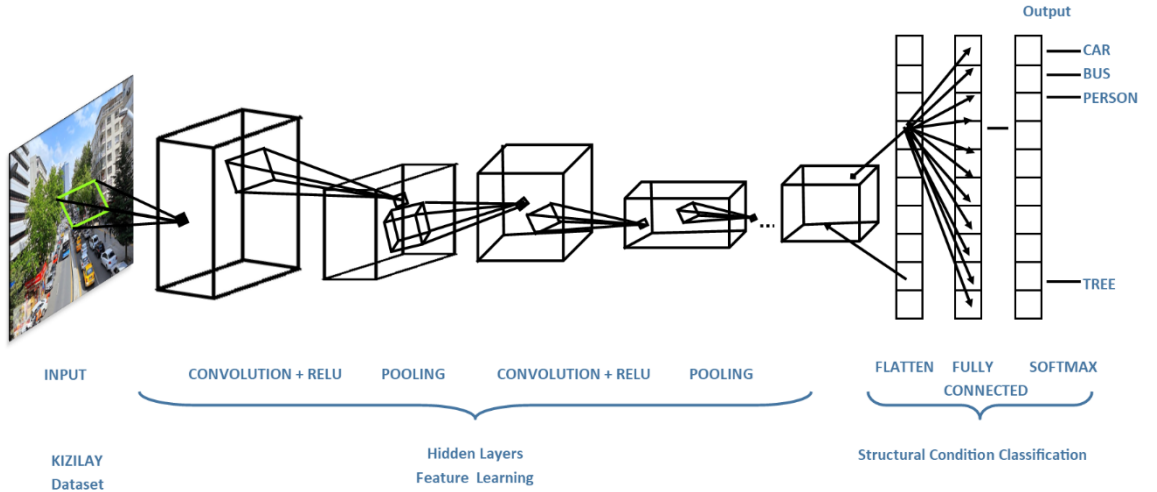
Derin öğrenme, üç veya daha fazla katmana sahip sinir ağları ile karakterize edilen ML'nin bir alt bölümünü temsil eder. DL, katmanlı bir ağ yapısı yoluyla hiyerarşik özellik bilgisinin edinilmesini içeren özellik öğrenmesi üzerine kuruludur. Bu stratejik yaklaşım, manuel tasarlanmış özelliklere duyulan ihtiyacı etkili bir şekilde ortadan kaldırır. Hiyerarşik özellik göstergeleri, ağ derinliği arttıkça çıkarılan görüntü özelliklerinin giderek daha karmaşık ve bilgilendirici hale geldiği anlamına gelir. Görüntü analizi, konuşma tanıma ve Dİİ dahil olmak üzere bir dizi problem alanında üstün sonuçlar elde etmek için çeşitli ağ mimarileri gereklidir. İlerleyen bölümlerde, CNN ve derin üretici modeller de dahil olmak üzere birkaç önde gelen derin öğrenme mimarisine derinlemesine ineceğiz (Pouyanfar, vd., 2018).

2.6.1 Evrişimli sinir ağları

Evrişimsel Sinir Ağları (CNN'ler), DL alanının köşe taşı konumunda olup, çeşitli alanlarda yaygın popülariteye sahiptir ve sıkça uygulanır. Konuşma işleme, Dİİ ve bilgisayarlı görme gibi alanlarda çok yönlülük sergilerler. Hesaplama kaynakları ilerlemeye devam ettikçe,

daha büyük CNN mimarileri özellikle görüntü analizinde kayda değer avantajlar göstermiştir. 2012'de çığır açan bir dönüm noktası olmuştur (Krizhevsky vd., 2012), ImageNet görüntü sınıfı yarışmasında zafer kazanan büyük AlexNet ağını tanıttı. Bu başarı, araştırmacıları, ImageNet'in performansını sürekli artırmayı hedefleyen bir dizi ağ yapısı öneren bir inovasyon dalgasını tetiklemiştir. Bu klasik ağlar arasında VGG, GoogLeNet ve ResNet ön plana çıkıyor.

CNN'lerin en büyük güçlerinden biri, yerel bağlantıların kullanımında yatmaktadır. Geleneksel tam bağlı ağların aksine, CNN'ler yerel bağlantıların ve paylaşılan ağırlıkların gücünden yararlanır. Bu mimari seçim, hız ve eğitim kolaylığı sağlarken nispeten düşük sayıda parametreyle sonuçlanan iki temel avantaj sağlar. Bu tasarım felsefesi, görsel alanın tümünü aynı anda işlemeye çalışmak yerine, bir sahnedeki belirli bölgelere duyarlılık gösteren görsel kortikal hücrelerin işleyişini paralel hale getirir. Bu hücreler yerel filtreler gibi davranır, verilen verilerde yer alan mekânsal olarak yerel ilişkileri çıkarırlar.



Şekil 2.5 Görüntü bir CNN'nin mimarisini göstermektedir. Bir CNN, giriş verilerini işlemek ve analiz etmek için çalışan birkaç temel katmandan oluşur. Bu katmanlar, ham verileri alan giriş katmanını, öznitelik çıkarımından sorumlu evrişim tabakasını, doğrusal olmayanlık tanıtmak için bir ReLU (örnek olarak) aktivasyon katmanını, mekansal aşağı örnekleme için bir alt örnekleme katmanını ve son olarak sınıflandırma veya regresyon görevleri için yoğun bağlantılı bir katmanı içerir. Bu katmanlar sistematik olarak yığılır ve karmaşık görüntü tanıma ve analizi yapabilen kapsamlı ve sağlam bir evrişimli sinir ağı oluşturur

Tipik bir CNN yapısı, birkaç temel katmandan oluşur: görünür katman, evrişim katmanı, havuzlama katmanı, tam bağlı katman (çok katmanlı perceptrona benzer) ve son katman. Görünür katman (girdi), veriler için giriş noktası olarak hizmet eder ve bunu sonraki sinir ağı bileşenlerinden geçirir. CNN katmanı, giriş verileri üzerinde evrişim işlemleri gerçekleştirir ve sonuçları sonraki katmana iletir. Örnekleme katmanları, belirli bölgelerdeki özellikleri özetledikleri için boyut küçültme açısından faydalıdır. Tam bağlantılı katmanlar, CNN'lerde vazgeçilmez öğelerdir, bir katmandaki her nöronu diğer bir katmandakilerle bağlarlar. Evrişim katmanı, aktivasyon fonksiyonu ve havuzlama katmanları düzenlenir. CNN yapısının görsel bir betimi **Şekil 2.5'te** mevcuttur.

Evrişimli sinir ağlarında (CNN'lerde), giriş görüntülerinin üç boyutu vardır - yükseklik (m), genişlik (n) ve derinlik veya kanallar (r). Örneğin, RGB görüntülerinin $r=3$ kanalı vardır. Her evrişim katmanı, boyutları $n \times n \times q$ olan k filtre veya çekirdeğe sahiptir. Tipik olarak, n giriş görüntüsünün boyutundan daha küçüktür ve q, r 'ye eşit veya daha küçük olabilir. Bu filtreler, girdi verileriyle evrişerek yerel bağlantılar oluşturur. Her biri $m-n-1$ boyutunda k öznitelik haritası (h_i) üretmek için özdeş ağırlıklar (W_i) ve önyargılar (b_i) kullanılır.

Temel olarak, evrişim katmanı (tam bağlantılı katmanların aksine tüm girişi kullanan), filtre ağırlıkları ile bölgesel giriş yamaları arasındaki nokta çarpımlarını çarpar. Bu, yerel uzamsal ilişkileri yakalar. Daha sonra, çıkışta doğrusal olmayanlar sunmak için öge bazında bir $f()$ aktivasyon fonksiyonu kullanılır.

Sinir ağının alt örnekleme katmanlarında, her öznitelik haritası boyut azaltma işlemine tabi tutulur, bu işlem ağ ağırlıklarının sayısını azaltmayı, eğitim sürecini hızlandırmayı ve aşırı uyumluluğu azaltmayı amaçlar. Bu boyut azaltma genellikle $p \times p$ boyutundaki arka arkaya bölgeler üzerinde uygulanan ortalama veya maksimum havuzlama gibi havuzlama işlemleri yoluyla elde edilir, burada p tüm öznitelik haritaları genelindeki filtre boyutunu temsil eder.

CNN'nin son aşamalarında, çoğunlukla tam bağlantılı katmanlardan faydalanılır. Bu katmanlar, görüntülerden çıkarılan düşük seviyeli ve orta seviyeli özellikleri kullanarak

verilerdeki görüntü özelliklerinin üst düzey temsillerini oluşturur. Düşük seviyeli özellikler, çizgiler veya noktalar gibi temel ayrıntıları içerir ve üst düzey özellikler daha sonra bu temel öğelerin üzerine inşa edilir, görüntüdeki nesnelerin ve daha büyük örüntülerin tanımlanmasına katkıda bulunur. CNN'ler her iki özelliği de etkili bir şekilde kullanır, ilk evrişim katmanları düşük seviyeli özellikleri algılamak için filtreler öğrenmeye odaklanırken, sonraki katmanlar gelişmiş düzeyde özellik tanıma konusunda uzmanlaşırlar (Kushal, Ranjan vd., 2023). Genellikle Softmax veya SVM gibi fonksiyonları kullanan son katman, her biri belirli bir sınıf olma olasılığını gösteren sınıflandırma puanlarını hesaplar.

2.6.2 U-Net cnn'ler

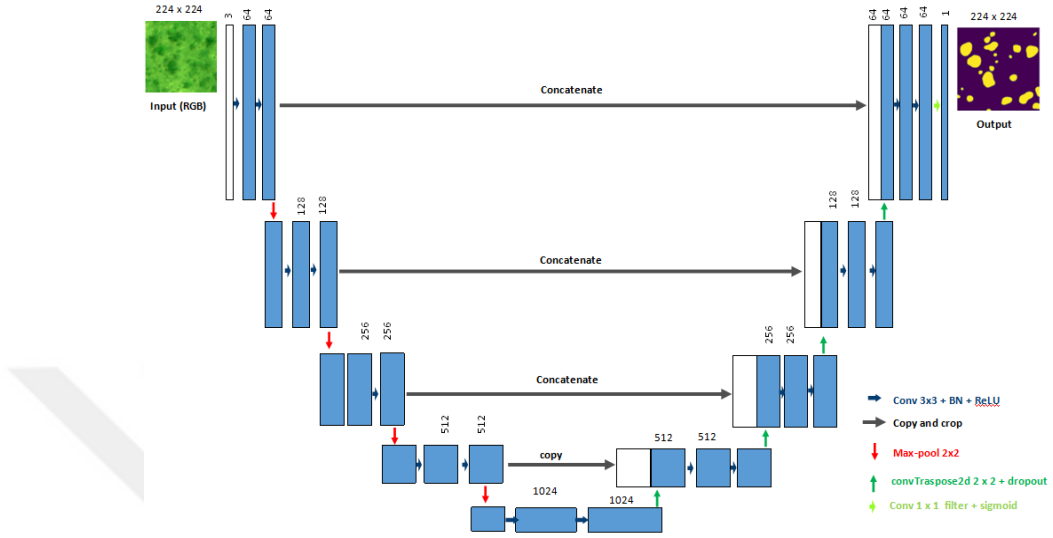
U-Net, özellikle bilgisayar destekli teşhis gibi alanlarda semantik segmentasyonda tıbbi görüntü analizini büyük ölçüde dönüştürmüş devrimci bir derin öğrenme modeli mimarisidir. Olaf Ronneberger ve meslektaşları tarafından 2015 yılında önerilen U-Net model tasarımı, MRG, BT ve mikroskopide lezyonların, tümörlerin ve hücresel yapıların kesin sınırlandırılmasına ve yerleştirilmesine olanak tanır (Ronneberger vd., 2015). Bugüne kadar 12.000'den fazla araştırma alıntısıyla muazzam etki açıktır.

2.6.2.1 U-Net mimarisi

İsmine sadık olarak, U-Net iki farklı yola sahiptir - bağlam elde etmek için bir daralma veya kod çözücü yol ve kesin yerelleştirmeyi sağlayan artan veya kod çözücü izleme. Kodlayıcı, VGG16 gibi tipik bir evrişim ağı tasarım desenini takip eder, aşağı örnekleme için evrişimlerin, doğrusal olmayan etkinleştirmelerin ve maksimum havuzlamanın tekrarlanan uygulamasıyla. Bu, adımsal olarak 572x572 giriş görüntülerinden 1024 öznitelik haritasına kadar semantik açıdan zengin öznitelik temsillerini çıkarır 18x18 mekânsal çözünürlükte.

Kod çözücü yolu daha sonra bu öznitelik haritalarını büyük 2x2 aktarma evrişimleri kullanarak yukarı örnekler, aynı zamanda atlama bağlantıları aracılığıyla eşleşen çözünürlüklü kodlayıcı öznitelik haritalarıyla birleştirir. Bu, hem küresel bağlamı hem de

ince yerel bilgileri birleştirerek çıktı katmanında tam giriş çözünürlüğü semantik maskeleri yeniden oluşturmaya izin verir. Ek evrişim katmanları yerelleştirmeyi ince ayarlar. Ağın şematik bir temsili için **Şekil 2.6'e** bakın.



Şekil 2.6 En düşük 224x224 piksel çözünürlüğünde bir örnek göstererek U-net mimarisini göstermektedir. Her mavi kutu, üstte belirtilen kanal sayısına sahip çok kanallı bir özellik haritasını temsil eder. Kutunun sol alt köşesinde x-y boyutu verilmiştir. Beyaz kutular özellik haritalarını çoğaltır ve oklar farklı işlemleri belirtir. Ronneberger ve diğerleri.(2015)

U-Net'in birçok biyomedikal zorluk üzerinde elde ettiği son derece rekabetçi sonuçlar, onu semantik segmentasyon görevleri için bir de-facto taban çizgisi olarak yerleştirmiştir. Tüm görüntüler üzerinde uçtan uca eğitim gibi kilit yönler, önceki boru hatlarındaki kırılğan ön ve son işlem bağımlılıklarını ortadan kaldırır. Atlama bağlantıları, bağlamsal bilgilerden yararlanırken kesin yerelleştirmeye izin verir. GPU işleme ile verimli şekilde birleştirilen bu yetenekler, o zamana kadar son derece zorlayıcı olarak kabul edilen devrimci tıbbi görüntüleme uygulamalarını mümkün kılmıştır.

2.6.2.2 U-Net'in uygulamaları

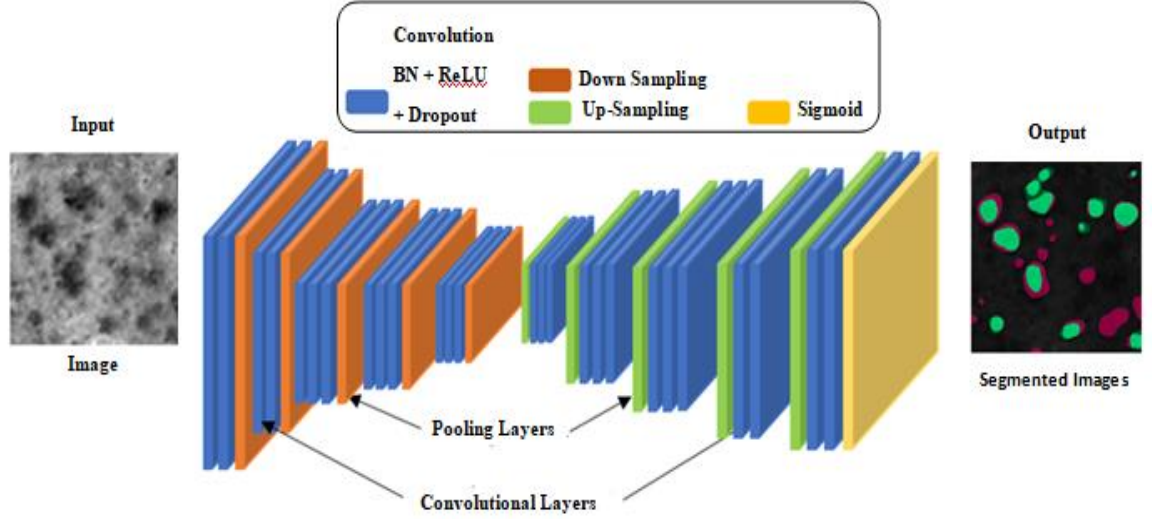
U-Net'in sınıf dışı sonuçlar sunduğu bazı örnek kullanım durumları şunlardır: Bilgisayar destekli teşhis için MRI taramalarından beyin tümörü ve multipl skleroz lezyon segmentasyonu (Kao vd., 2018; Roy vd., 2019). Elektron mikroskopisi görüntü analizi için hücre zarı segmentasyonu (Falk vd., 2019). Kanser araştırmalarına yardımcı olmak için

tüm slayt görüntülemeyen sayısal histopatoloji kullanarak çekirdek ve bez segmentasyonu (Aresta vd., 2019). Göz hastalığı taramasını otomatikleştirmek için fundus fotoğrafçılığından retinal damar yapısı ve optik disk segmentasyonu (Gu vd., 2019). Kodlayıcı-kod çözücü derinliklerindeki özel uyarlamalar veya modaliteleri bütünleştirmek, çeşitli diğer segmentasyon görevlerini ele almaya izin verir. Daha yeni varyantlar transformatörleri, özdeği, düzenleme stratejileri vb. performansı daha da artırmak için dahil eder. Ancak, temel U-Net hala oldukça popülerdir, basit ama son derece güçlü tasarımı sayesinde.

2.6.3 Segnet semantik segmentasyon ağı

SegNet, bilgisayarla görmede önemli bir girişim olan, bir görüntünün her pikseline bir sınıf etiketi atamayı içerir. Bu, bir sahnedeki nesnelerin ve arka plan bölgelerinin kesin yerlerini belirlemeye izin verir. SegNet, verimli piksel düzeyinde SS için derin tam convolutional sinir ağı mimarisidir (Badrinarayanan vd., 2017). SegNet'in temelinde yatan neden, hem doğru hem de sınırlı donanım kaynaklarına sahip gömülü sistemlere dağıtmak için yeterince hafif bir model geliştirmektir.

SegNet Mimarisi, SegNet birincil olarak bir özellik ayıklayıcı ağı ve ilgili bir özellik yeniden yapılandırıcı ağıdan oluşur, ardından nihai piksel düzeyinde kategorilendirme katmanı gelir. Özellik çıkarıcı ağı, evrişim ve maksimum havuzlama katmanlarının ara katmanlanmış bir modelinde örgütlenmiş 13 evrişimli katmandan oluşur. Bu, uzamsal boyutları küçültürken giderek daha soyut özellik temsillerini çıkarır. SegNet'in kritik yönü, max havuzlama indekslerinin depolanması ve kod çözücü yeniden örnekleme işleminde yeniden kullanılmasıdır. Son kodlayıcı katmandan gelen özellikler, girdi çözünürlüğünde bir özellik haritasını ardışık olarak yeniden oluşturmak için evrişim aktarım katmanlarından oluşan bir kod çözücü ağına aktarılır. Depolanan max havuzlama indeksleri, kod çözücüdeki doğrusal olmayan yeniden örnekleme için kullanılır. Bu, yeniden örnekleme için tüm özellik eşlemlerini yükleme ihtiyacını azaltır. Son olarak, eğitilebilir bir softmax sınıflandırma katmanı semantik etiketler atar. Şekil 6'daki SegNet mimari diyagramı için bakın.



Şekil 2.7 SegNet Mimarisinin örnek gösterimi

Tüm özellik eşlemleri yerine yalnızca havuzlama dizinlerini depolayarak SegNet, kod çözücü bellek gereksinimlerini büyük ölçüde azaltır. Bu, çevrimdışı son işlem adımlarına gerek kalmadan gerçek zamanlı dağıtımı gerçekleştirilebilir hale getirir. SegNet'in ayrıca diğer mimarilerden önemli ölçüde daha az parametresi vardır, aşırı öğrenme endişelerini azaltır.

SegNet Uygulamaları Güçlü bir semantik segmentasyon çerçevesi olarak SegNet, tıbbi görüntüleme, otonom sürüş sistemleri, uydu sahnesi analizi, robot navigasyonu ve daha fazlası dahil olmak üzere çeşitli alanlarda en iyi sonuçları teslim etmiştir. Temel uygulamalar arasında şunlar yer alır:

Biyomedikal görüntü segmentasyonu: SegNet, bilgisayar destekli teşhis için MRI, BT taramaları ve mikroskopiden hücresel ve anatomik yapıları doğru bir şekilde segmente eder (Myronenko, 2018). Kendi kendine sürüş yapabilen araçlar: Yolları, yayaları, araçları ve arka plan bölgelerini kesin bir şekilde sınıflandırarak, SegNet kendi kendine süren arabalar için sahne anlayışını mümkün kılar (Garcia-Garcia vd., 2017). Uydu görüntüleri: Uydu görüntülerinden yüksek çözünürlüklü arazi örtüsü sınıflandırması, coğrafi haritalama çabalarına yardımcı olur (Audebert vd., 2016). Çoğu gerçek dünya dağıtımında SegNet modelleri daha fazla doğruluk için özelleştirilmiş veri kümelerinde daha fazla ince

ayarlanmıştır. Verimli mimari, uç kullanım durumları genelinde böyle uyarlamaları kolaylaştırır.

2.7 Ağ Normalizasyonu

2.7.1 Normalizasyonu

Ağ normalizasyonu, derin öğrenme alanında önemli bir yaklaşım haline gelmiştir ve sinir ağlarının eğitimi ve işleyişi için temeldir. Normalizasyon tekniklerinin amacı, eğitim süreci boyunca her katmana kayan giriş dağılımından kaynaklanan içsel kovaryans kayması sorununu hafifletmektir (Ioffe vd Szegedy, 2015). Bu yöntemler, her katmana giren verileri normalleştirerek daha istikrarlı ve etkili bir eğitim sağlar ve genellikle daha hızlı yakınsama ve daha iyi genelleme becerilerine yol açar.

En yaygın kullanılan normalizasyon tekniklerinden biri, Ioffe ve Szegedy (2015) tarafından önerilen Batch Normalization (BN) tekniğidir. BN, girişleri mini-part boyutu boyunca normalleştirerek ağı daha büyük öğrenme hızlarıyla eğitilmesini sağlar ve genel gradyan sorunlarını hafifletir. Başarısı, özel senaryolar için uyarlanmış çeşitli alternatif normalizasyon yöntemlerinin geliştirilmesine yol açmıştır, bunlar arasında tekrarlayan sinir ağları için Katman Normalizasyonu (LN) (Ba vd., 2016), stil transfer görevleri için Örnek Normalizasyonu (IN) (Ulyanov vd., 2016) ve küçük parti boyutları ile eğitim için Grup Normalizasyonu (GN) (Wu vd., 2018) bulunmaktadır.

Doğal görüntü işleme alanının ötesinde, normalizasyon teknikleri, tıbbi görüntüleme alanında derinlemesine uygulamalar bulmuştur. Örneğin, (Kao vd.,2018), beyin tümörü segmentasyonu için GN'yi uygularken, (Isensee vd., 2021), çeşitli segmentasyon görevleri için IN'yi bir öz-düzenleyici çerçevede kullandı. (Chen vd., 2021), karın çoklu organ segmentasyonu için Transformer'larda LN'yi kullandı, bu da bu tekniklerin farklı sinir ağı mimarileri ve tıbbi görüntüleme modaliteleri arasındaki çok yönlülüğünü göstermektedir.

Biyomedikal anlamsal segmentasyon ve çapraz modluk sentez bağlamında çeşitli normalizasyon yöntemlerinin performansını değerlendirmek için karşılaştırmalı çalışmalar

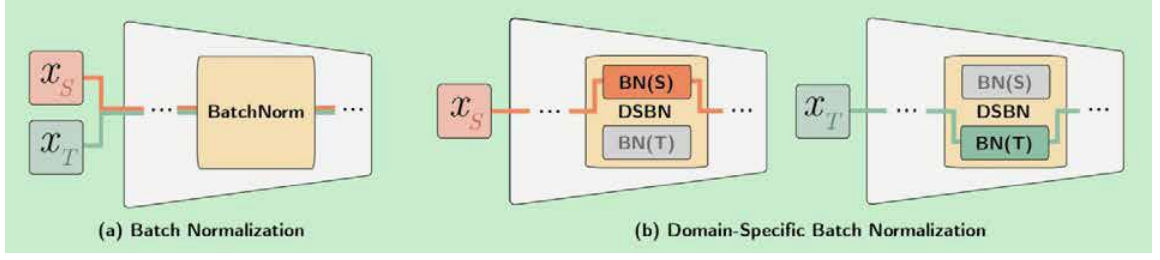
yapılmıştır. Zhao vd. (2019), biyomedikal anlamsal segmentasyon için farklı normalizasyon tekniklerinin detaylı bir karşılaştırmasını sağlarken, Huang vd. (2020), çapraz modluk sentez görevlerindeki etkilerini inceledi.

2.7.2 Gelişmiş normalizasyon teknikleri

Bu alandaki araştırmalar ilerledikçe, belirli zorlukları ele almak için daha gelişmiş normalizasyon teknikleri geliştirilmiştir. Örneğin, Huang & Belongie (2017) tarafından AdaIN ve (Park vd., 2019) tarafından SPADE gibi yöntemler, daha güçlü afiş dönüşümleri sunmakta ve normalizasyon sürecini yönlendirmek için dış veri veya anlamsal maskeler kullanmaktadır. Alan uyarlaması ve çoklu alan öğrenme bağlamında, Perez vd. (2018) tarafından Modulating BN ve (Li vd., 2016); (Rebuffi vd., 2018) tarafından Alan Özgü BN gibi teknikler, BN istatistiklerini alan özgü bilgilere dayanarak değiştirmeyi amaçlamaktadır.

2.7.3 Toplu normalizasyon

BN, derin sinir ağı modellerini eğitme konusundaki formidabl bir meydan okumanın kritik bir çözümü olarak ortaya çıktı. Model eğitimi çerçevesinde BN, küçük veri toplulukları üzerinde ortalama ve standart sapma istatistiklerini hesaplama yoluyla çalışır. Sinir ağı parametrelerinin sürekli ayarlanması, ağıdaki her katmanın ara çıktısında gelişmiş stabilite oluşturur.



Şekil 2.8 Etiketlenmemiş Alan Adaptasyon için Alan Özgü BN uygulamasını gösterir. Şekil 2.8, geleneksel BN ile uzmanlaşmış Alan Özgü BN arasında görsel bir karşılaştırma sağlar. Sonucusunun içindeki iki ayrı branşın tanıtımı dikkat çekicidir, bunlardan her biri kaynak veya hedef alandan gelen verileri işlemek için özel olarak tasarlanmıştır. Bu uyarlanabilir Alan Özgü BN katmanı, Toplu Normalizasyon (Chang vd., 2019) kullanan herhangi bir etiketlenmemiş alan uyarlama ağına sorunsuz entegrasyon yetenekleri sergilemektedir ve Telif Hakkı ©2019, IEEE'ye aittir

Li ve diğerlerinin çalışmasında (Li vd., 2018), model genelleştirme potansiyelini artırmayı gösteren Adaptive BN tanıtıldı. Dikkate değer bir şekilde, diğer alan uyarlama modellerinin aksine, Uyarlanabilir Toplu Normalizasyon ek modül veya parametre eklemeyi gerektirmeden çalışır. Bunun yerine, hedeflenen alandaki BN katmanlarının ağırlıklarını dinamik olarak ince ayarlamak için kaynak alan veri kümesinden yararlanır ve aynı zamanda alan adaptasyonu ile ilgili ağırlıkları günceller. Sonraki olarak (Değişim vd., 2019) özgün BN yaklaşımının temelini oluşturarak Alan-Spesifik BN'yi önerdi. Bu yeniliğin yapısal düzeni Şekil 7'de tasvir edilmiştir. Özellikle, birden fazla Toplu Normalizasyon katmanları kümesi boyunca ortalama ve varyans istatistiklerini yaklaştırarak çalışır. Buna ek olarak Alan-Spesifik BN, alana özgü özellikleri yakalamada olağanüstü bir yeteneğe sahiptir. Geleneksel BN'nin aksine, bu varyant alandan bağımsız özellikleri ayıklamada mükemmelleşmiştir, bu da modeldeki alan adaptasyon için oldukça uygundur.

2.7.4 Grup normalizasyonu

Derin evrişimli ağlar ölçeklendikçe, bellek kısıtlamaları nedeniyle büyük toplu boyutlar gerçekleştirilemez hale gelir. Bu, toplu başına yeterli örnek sayısını varsayan Toplu Normalizasyonun (BN) (Ioffe ve Szegedy, 2015) etkinliğini engeller. Grup Normalizasyonu (GN) böylece minik topluluklarla DNN eğitimini mümkün kılmayı amaçlar (Wu vd., 2018).

GN kanalları gruplara bölümler ve her grup içinde normalizasyon istatistiklerini bağımsız olarak hesaplar. C kanalın G gruba bölündüğü varsayılarak, her grup "g" için ortalama ve varyans şu şekilde belirlenir.

$$\mu_g = \frac{1}{HWCg \sum hwc x_{ghwc}} \quad (10a)$$

$$\sigma_g^2 = \frac{1}{HWCg \sum hwc (x_{ghwc} - \mu_g)^2} \quad (10a)$$

Burada, x şeklinde HWC'ye bölünmüş bir giriş etkinleştirilmesi kanallar boyunca gruplara bölünmüştür. GN daha sonra x'i grup g için μ_g , σ_g kullanarak normalize eder. Hesaplama grup başına olduğundan, GN toplu boyuta bağlı değildir. Normalizasyon sonrası afirm dönüşümü, BN'ye benzer esneklik sağlar.

Deneyler, GN'nin büyük topluluklarda BN ile eşitliğe ulaştığını, daha küçük toplu boyutlarda BN, LayerNorm ve InstanceNorm'u çok geride bıraktığını göstermiştir. Bu sağlamlık, tıbbi görüntüleme ve video tanıma gibi içkin veri kısıtlamaları bulunan durumlara uygundur. Böylece GN, BN'nin yetersiz kaldığı çok derin modellerin küçük toplu eğitimini mümkün kılan etkili bir normalizasyon seçeneği olarak hizmet eder.

Tartışılan çalışmalar, semantik segmentasyon yöntemlerini yeni alanlara uyarlamada önemli gelişmeleri temsil eder. Bunlar büyük ölçüde araç üstü kameralardan ve hava görüntülerinden gelen görüntülerin sağlam segmentasyonuna odaklanmaktadır. Bununla birlikte, semantik segmentasyon yöntemleri hala hava görüntüleri için uyarlanması ve değerlendirilmesi gerekmektedir. Yaklaşan araştırmamız, bu alan uyarlama yaklaşımlarının etkinliğini, çok spektrumlu görüntülerin semantik segmentasyonu için RIT-18 ve DSTI veri setlerinde değerlendirecektir. Özellikle, bu yöntemleri özgün alanlarından hava görüntüsü alanına DualNorm_UnetV3 mimarisi kullanarak uyarlamayı hedefleyeceğiz.

3. MATERİYAL ve YÖNTEM

3.1 Giriş

Görüntü segmentasyonu, görüntü analizinde ve bilgisayar görüşünde en önemli hedeflerden birini temsil eder ve bir görüntünün anlamlı bölümlere veya nesnelere ayrılmasını içerir. Uzaktan algılama görüntü segmentasyonu, son yıllarda artan araştırma ilgisini çekmiştir. Uzaktan algılama görüntüleri, görünür ve kızılötesi ışıqla görüntüleme, radar, lidar gibi çeşitli sensörler ve teknikler aracılığıyla bilgi yakalar. Bu veri, yeryüzü, atmosfer ve diğer gezegenlerin yüzeyini fiziksel temas olmadan inceleme ve izleme olanağı sağlar, bu da çevre izleme, kentsel planlama, tarım, jeoloji ve meteoroloji gibi alanlarda uygulamalara olanak tanır. Hassas tarım, gıda denetimi, tıbbi görüntüleme gibi birçok uygulama, bu tür uydu verilerinin analizinden faydalanır (Bioucas-Dias et al., 2013).

Derin evrişimli sinir ağlarının (CNN'ler) Uzaktan algılamalı görüntü segmentasyonunda kullanılması, önemli bir eğilim olmuştur (Zhong vd., 2019). Özellikle, U-Net modeli popüler bir model seçeneği olarak ortaya çıkmıştır (Ronneberger vd., 2015). Ancak, tipik CNN normalizasyon tekniklerinin (Ioffe vd Szegedy, 2015) sınırlamalarından biri, yalnızca global görüntü düzeyinde istatistikleri kullanmalarıdır. Bu, uzaktan algılama görüntüleri için faydalı olabilecek farklı anlamsal sınıfların yerel istatistiklerini göz ardı eder

Bu bölümde, önerimiz yeni bir DualNorm-UNetV3 olarak adlandırılan modeli tanıtmaktadır. Bu model, Uzaktan algılama görüntü segmentasyonunun sağlamlığını artırmak için hem küresel hem de yerel istatistiksel bilgileri birleştirir. Temel konsept, yeni bir çift normalizasyon şeması aracılığıyla normalizasyon işlemine semantik sınıf bilgilerini entegre etmeyi içerir. Modelimizde benimsenen temel tekniklerden biri, BN (Ioffe vd., 2015) ve GN (Wu, vd., 2018) kullanarak çift normalizasyon stratejisidir. Bu, farklı semantik bölgelere karşılık gelen aktivasyonları ayrı ayrı modüle etmeye izin verir.

Önemli katkılar üç katmandır:

1. Küresel ve yerel istatistikleri sırasıyla yakalamak için paralel Toplu Normalizasyon ve Grup Normalizasyon ile DualNorm_UNetV3 Artıklık Bloğu önerilmiştir.
2. Her iki normalizasyon tekniğinden kademeli olarak yararlanmak için iki aşamalı bir eğitim prosedürü tasarlanmıştır.
3. Geniş ölçüde kullanılan UNet mimarisine minimum değişiklikle etkili entegrasyon gösterilmiştir.
4. Küçük ve büyük topluluklara karşı dayanıklılık

Öncelikle, CNN'lerdeki normalizasyon yöntemleri ve Uzaktan algılama görüntü segmentasyonu için derin öğrenme teknikleri ile ilgili geçmiş bilgiler gözden geçirilmiştir. Daha sonra, önerdiğimiz DualNorm-UNetV3'ün metodolojisi ayrıntılı olarak açıklanmıştır.

Çizelge 3.1 Bu çalışmanın referans aldığı çok spektrumlu görüntülerin semantik segmentasyonu için ikili normalizasyon ve DA'yı çözmeye çalışan Örnek Akademik makaleler

Başlık	Yazarlar	Tarih	Özet	Referans
“DualNorm-UNet: İkili Normalizasyonla Dayanıklı Tıbbi Görüntü Segmentasyonu” (Xiao vd., 2021)	“Junfei Xiao, Lequan Yu, Lei Xing, Alan L. Yuille, Yuyin Zhou”	2021	Küresel ve yerel istatistikleri kullanan ikili normalizasyon yaklaşımı önererek tıbbi görüntü segmentasyon doğruluğunu ve dayanıklılığını geliştirir”	Link
“CateNorm: Dayanıklı Tıbbi Görüntü Segmentasyonu için Kategorik Normalizasyon” (Xiao vd., 2022)	“Junfei Xiao, Lequan Yu, Zongwei Zhou, Yutong Bai, Lei Xing, Alan Yuille, Yuyin Zhou”	2022	“Ön plan istatistiklerinden yararlanarak dayanıklı tıbbi görüntü segmentasyonu için kategorik bir normalizasyon tekniği tanıtır”.	Link

3.2 Geçmiş

Yaklaşımımızı ayrıntılandırmadan önce, CNN'lerde normalizasyon ve Uzaktan algılama görüntü segmentasyonu için mevcut teknikler hakkında gerekli arka planı sağlıyoruz.

3.2.1 CNN'lerde normalizasyon

Normalizasyon, derin CNN modellerini eğitirken vazgeçilmez bir bileşen haline gelmiştir. Normalizasyon şemaları, katman girdilerinin ortalama ve varyansını sabitleyerek ağ eğitimini hızlandırır ve genelleme yeteneğini artırır (Santurkar vd., 2018).

En yaygın olarak benimsenen teknik, toplu boyut boyunca hesaplanan ortalama ve değişim kullanarak aktivasyonları standartlaştıran Toplu Normalizasyondur (Ioffe vd., 2015). Bu toplu küresel normalizasyon, tüm uzamsal konumlara üniform olarak uygulanır. BN küresel görüntü düzeyi istatistiklerine güvenirken, bazı yöntemler normalizasyon için yerel istatistikleri kullanmayı araştırır. Örneğin, LN (Ba vd., 2016) her örnek içinde istatistikleri, GN (Wu ve He, 2018) kanal grupları üzerinde hesaplar ve IN (Ulyanov vd., 2016) her kanalın ve uzamsal konumun istatistiklerini ayrı ayrı kullanır. Bununla birlikte, yerel istatistiklerin potansiyeli, çok spektrumlu görüntü analizi için nispeten daha az araştırılmıştır.

Park ve ark. (2019) yakın zamanda, aktivasyonların girdi semantik maskelere göre uzamsal olarak değişen bir şekilde farklı semantik bölgelere uyarlandığı Mekânsal Uyarlanabilir Normalizasyon'u (SPADE) önerdi. Küresel ve yerel istatistikleri kullanmak için BN ile SPADE'yi entegre ederek çok spektrumlu segmentasyona ilk girişimi yapıyoruz.

3.2.2 Çok spektrumlu görüntü segmentasyonu için derin öğrenmenin kullanılması

CNN'ler, çok spektrumlu görüntü analizinde önemli bir etkinliği kanıtlamıştır. Bu, veriden otonom olarak bilgilendirici temsiller edinme yeteneklerinden kaynaklanır. Uzamsal haritalar çıkıtlayan tam evrişimli ağlar, çok spektrumlu görüntülerin SS'sinde özellikle yaygın olarak kullanılmıştır (Zhong vd., 2019).

SegNet ve U-Net gibi daha erken yöntemler, çok düzeyli özellikleri birleştirmek için bir kodlayıcı-kod çözücü mimarisini kullandı (Ronneberger vd., 2015). DenseNet (Huang vd., 2017), DeepLab (Chen vd., 2018) ve HRNet (Sun vd., 2019) gibi son teknikler, daha kapsamlı bağlamsal bilgiler yakalayarak segmentasyon doğruluğunu daha da geliştirir.

Normalizasyon CNN'lerde kritik bir bileşen olmasına rağmen (Kemker vd., 2018) ve (Zhong vd., 2019) mevcut çok spektrumlu segmentasyon yöntemleri semantik sınıflara dayalı olarak ayırım yapmadan doğrudan BN veya diğer küresel normalleştirmeyi benimser. Önerilen ikili normalizasyon şemamız, bu boşluğu ele almayı amaçlamaktadır.

Bunu takiben, DualNorm-UNetV3'e tamamlayıcı normalizasyon entegrasyonunda çekirdek bir bileşen olan DualNorm_UNetV3 Artıklık Bloğu'nu tanıtıyoruz.

3.3 Segmentasyon için Kayıp Fonksiyonları

Maliyet fonksiyonunun (kayıp) seçilmesi, beklenen değerler ile gerçek zemin gerçeği arasındaki farkı değerlendirmek için bir metrik olarak hizmet ederek herhangi bir modelin eğitim sürecinde kritik bir işleve sahiptir. İyi tasarlanmış bir kayıp fonksiyonu, öğrenme sürecini doğru tahminlere yönlendiren yeterli bir model ile eşanlamlıdır. Derin öğrenmede, çeşitli modeller belirli görevlerine özel kayıp fonksiyonlarını kullanır.

Kayıp fonksiyonları temel olarak iki kategoridedir: ampirik veya gözlem kaybı ve yapısal risk minimizasyonu (kayıp) veya düzenlilik terimi. Gözlem kaybı fonksiyonu, modelin tahminleri ile gerçek sonuçlar arasındaki farkı ölçer. Öte yandan, yapısal risk enküçültme işlevi, sadece ampirik kayıp fonksiyonunu değil, aynı zamanda daha bütüncül bir değerlendirme ölçütü sunan ek bir klasik terimi de kapsar.

Modelimiz SS ağını eğitmek için ampirik bir risk veya eğitim kaybı fonksiyonu kullanmaktadır. Tam olarak, kayıp modelin tahmin ettiği segmentasyon haritaları ile zemin gerçeği açıklamalarında sağlanan gerçek sınıf etiketleri arasındaki piksel düzeyindeki farkları hesaplar. Bu farklılıkları kayıp yoluyla kantifiye etmek, modelin görüntüler genelinde segmentasyon hatalarını en aza indirmek için parametrelerini güncellemesine izin verir. Etiketli verilere erişimle bu denetimli yaklaşımı benimsemek, modelin piksel düzeyinde nesne sınırlarını ve sınıflarını doğru bir şekilde çizgileyen etkili öznitelik temsillerini öğrenmesini sağlar. Ampirik kayıp denetimi, modeli sorun alanındaki karmaşık çok spektrumlu görüntüler üzerinde segmentasyon performansını artırmaya yönelik bir optimizasyon hedefi olarak hareket eder.

Modelimiz açıkça ikili sınıflandırma ve görüntü segmentasyonu için özel bir ampirik kayıp fonksiyonunun bir varyantı olan BCEWithLogitsLoss fonksiyonunu kullanır. Çok spektrumlu görüntü segmentasyon alanındaki ince desenleri ve nüansları yakalama kapasitesini daha da geliştirir.

Sıklıkla, Evrişimli sinir ağları segmentasyon görevleri için genellikle karşılık gelen gerçek etiketler tarafından tahmin edilen etiketler arasındaki farkı kantifiye eden piksel düzeyinde bir kayıp işlevini eniyileyerek eğitimden geçmektedir. Kayıp fonksiyonun seçimi, model optimizasyonunu ve segmentasyon sonuçlarının kalitesini önemli ölçüde etkiler.

Görüntü segmentasyonu için tipik bir kayıp fonksiyonu, olasılık dağılımları arasındaki farkı ölçen İkili Çapraz Entropi (BCE) kaybıdır. BCE kaybı, her pikselin olası C sınıflarından birine ait olduğu çok sınıflı segmentasyon modellerini optimize etmeye uygundur (Janocha ve Czarnecki, 2017).

3.3.1 Bce kayıp fonksiyonu

Her piksel için i ve sınıf c , model pikselin c sınıfına ait olma olasılığını gösteren bir pic olasılığı tahmin eder. Zemin gerçeği hedef dağılım ti 'yi sağlar, burada piksel i etiketi c 'ye sahipse $tic = 1$, aksi takdirde $tic = 0$.

BCE kayıp fonksiyonu daha sonra tahmin edilen ve hedef dağılımları her sınıf için bağımsız olarak karşılaştırır. Şu şekilde tanımlanmıştır:

$$BCE(ti, pi) = -\sum_c tic \log(pic) + (1 - tic) \log(1 - pic) \quad (11a)$$

$$BCEWithLogitsLoss = -N \sum_i = 1N \left[y_i \cdot \log(\sigma(y^i)) + (1 - y_i) \cdot \log(1 - \sigma(y^i)) \right] \quad (11b)$$

Burada, N veri kümesindeki piksel veya topluluk sayısını, i 'inci piksel için zemin gerçeği etiketi (arka plan için 0, ön plan için 1)'dir. sigmoid aktivasyon uygulamadan önce i 'inci piksel için ham tahmin (logit) ve $\sigma(x)$ sigmoid aktivasyon işlevini temsil eder, bkz. Şekil 8.

Her pikselin tahmin edilen dağılımına BCE kaybını gerçek etiketle karşılaştırarak uygulayarak, model doğru sınıflar için daha yüksek olasılıklar çıkarmaya teşvik edilir. Tüm piksellerden gelen katkılar bir görüntüdeki genel kaybı hesaplamak için toplanır. BCE kaybını asgari hale getirmek, zemin gerçeği etiketleri ile hizalanmış sınıf olasılık dağılımlarının öğrenilmesini sağlar.

3.3.2 Logits ile bce kaybı

Yaygın bir varyasyon, logits ile BCE veya BCEWithLogitsLoss'tur. Burada model normalleştirilmiş olasılıklar yerine normalleştirilmemiş logitler zic çıkarır. Logitler öncelikle olasılıkları elde etmek için bir sigmoid fonksiyonuna aktarılır:

$$pic = sigmoid(zic) = 1 / (1 + \exp(-zic)) \quad (12)$$

Ardından olasılıklar pic , gerçek etiketler ti ile karşılaştırılarak BCE kaybı uygulanır.

Olasılıklar ve logitler arasında dönüşüm yapılırken meydana gelebilecek sayısal kararsızlıktan kaçınmak için logitler kullanılır. Model, kaybı doğrudan logitler üzerinde optimize ederek öğrenebilir. BCEWithLogitsLoss, her ikisini de tek bir katmanda birleştirir, verimli bir uygulama sağlar.

3.3.3 Semantik segmentasyon için bce kaybı

BCE kaybı, çok sınıflı tahminleri işleyebilmesi ve olasılıksal bir yorum sağlaması nedeniyle segmentasyona uygundur. Her piksel kendi tahmin edilen dağılımına göre bağımsız olarak sınıflandırılır. Bu yerellik, BCE kaybını yoğun segmentasyon görevlerine kolayca uygulanabilir hale getirir.

Çalışmalar, BCE'nin çeşitli alanlarda segmentasyon için etkili olduğunu göstermiştir. Iglovikov ve diğerleri (2018), uydu görüntüsü segmentasyonu için logitler ile BCE'nin diğer kayıplardan daha iyi performans gösterdiğini bulmuştur. Maiti ve Chakraborty

(2019), BCE'nin biyomedikal görüntü segmentasyonu için Dice kaybından daha yüksek doğruluk elde ettiğini gösterdi.

3.3.4 bce kaybı ile dualnorm-unetv3

Çok spektrumlu görüntü segmentasyonunda alan adaptasyon için önerilen DualNorm-UNetV3 modelimizde, optimizasyon ölçütü olarak BCEWithLogitsLoss'u kullanıyoruz.

BCE kaybı, CNN'nin zemin gerçeği etiketleriyle eşleşen sınıf olasılıkları çıkarmasına yardımcı olur. Farklı alanlardaki seyrek açıklamalı görüntülerden segmentasyon öğrenme için güçlü bir hedef sağlar. Pikseller i ve sınıflar c için toplanarak BCE kaybını PyTorch kullanarak uygularız:

$$BCE = torch.nn.BCEWithLogitsLoss(z, t) \quad (13)$$

burada z ve t sırasıyla logitler ve hedeflerdir. Etkili alan adaptasyon için kaynak verilerinde BCE kaybını hedef verilerde tutarlılık düzenlemesi ile birleştiririz.

BCE kaybından gelen gradyanlar, CNN'yi ikili normalizasyon şeması aracılığıyla alandan bağımsız özellikler çıkarmaya ve doğru çok spektrumlu görüntü segmentasyonu çıkarmaya yönlendirir. Deneylerimiz, BCEWithLogitsLoss'un önerilen DualNorm-UNet mimarisini etkin bir şekilde optimize ettiğini göstermektedir.

3.4 Segmentasyon için Değerlendirme Ölçütleri

Semantik segmentasyonda, bir sınıflandırıcı bir görüntüdeki her piksele atanmış bir etiket tahmin eder. Segmentasyon performansının değerlendirilmesi, tahmin edilen etiketlerin gerçek zemin gerçeği açıklamalarıyla hizalamasının kantitatif olarak ölçülmesini gerektirir. Değerlendirme ölçütünün seçimi, model geliştirmeyi ve uygulamayı önemli ölçüde etkiler. Yaygın ölçütler Piksel Doğruluğu, Birliktekesişim Ortanca Değeri (mIoU) ve Jaccard İndeksi'ni içerir. Dayanıklılığı ve yorumlanabilirliği nedeniyle çok spektrumlu segmentasyon modellerini değerlendirmek için Jaccard Dizinini kullanıyoruz.

3.4.1 Küme benzerliği (jaccard indeksi)

Kesişimin Birliği (IoU) olarak bilinen Jaccard İndeksi, iki küme arasındaki benzerliği ölçer. A ve B kümeleri için şu şekilde tanımlanır:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (14)$$

$|A \cap B|$ kesişim boyutudur ve $|A \cup B|$ birlik boyutudur. Segmentasyon için, A ve B belirli bir sınıfa karşılık gelen projekte edilmiş ve referans veri (zemin gerçeği) piksel kümelerini ifade eder. Jaccard İndeksi mükemmel eşleşmeleri 1'e eşit ve ayrık olduklarında 0'a eşittir. Daha yüksek bir puan daha büyük benzerliği gösterir.

Jaccard İndeksi genel segmentasyon performansını değerlendirmek için genellikle sınıflar üzerinden ortalama alınır. Bir görüntüdeki dengeli olmayan sınıf dağılımları için bile sağlam bir metrik sağlar. Segmentasyon Görevlerine Uygulanabilirlik

Birkaç özellik, Jaccard İndeksi'ni segmentasyon modellerini değerlendirmek için uygun kılar: $[0, 1]$ aralığındaki sınırlı aralık sonuçların yorumlanmasını kolaylaştırır, Aşırı ve yetersiz segmentasyon hatalarının ikisini de hesaba katmaktadır. Doğruluk ile karşılaştırıldığında sınıf dengesizliğine karşı sağlamdır. piksel mutlak sayısına karşı değişmeyen, görüntüleri karşılaştırmaya yardımcı olur. Çalışmalar, Jaccard İndeksi'nin tıbbi görüntüleme, sahne anlayışı (Everingham vd., 2010) ve tarım (Kamilaris vd., 2018) dahil olmak üzere alanlar arasındaki segmentasyon görevleri için insan algısıyla yüksek korelasyona sahip olduğunu göstermektedir.

3.4.2 Jaccard benzerliği ile dualnorm-unetv3 değerlendirmesi

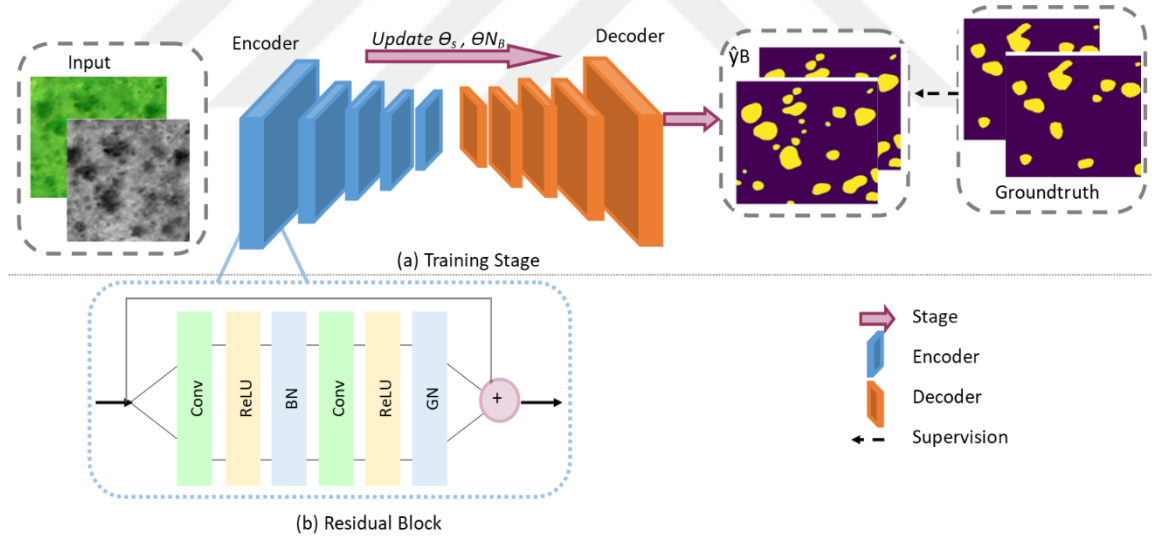
Çok spektrumlu görüntü segmentasyonu için alan adaptasyonu ile ilgili çalışmamızda, önerilen DualNorm-UNet modelini kullanarak birincil değerlendirme ölçütü olarak ortalama Jaccard İndeksini kullanıyoruz. Jaccard İndeksi, farklı alanlardan gelen görüntülerde değişen sınıf dağılımlarına ve çözünürlüklere sahip segmentasyon doğruluğunun güvenilir bir ölçüsünü sağlar. Model performansının standartlaştırılmış değerlendirmesine ve karşılaştırmasına olanak tanır.

Jaccard İndeksini PyTorch kullanarak hesaplıyoruz:

$$jaccard = torch.jaccard(predictions, targets) \quad (15)$$

DualNorm-UNet'i yüksek bir Jaccard İndeksi elde etmek için optimize etmek, alanlar arasında iyi genelleşen sağlam modellere yol açar. Deneylerimiz, DualNorm-UNet gibi çok spektrumlu segmentasyon modellerini değerlendirmek için Jaccard İndeksinin etkinliğini göstermektedir.

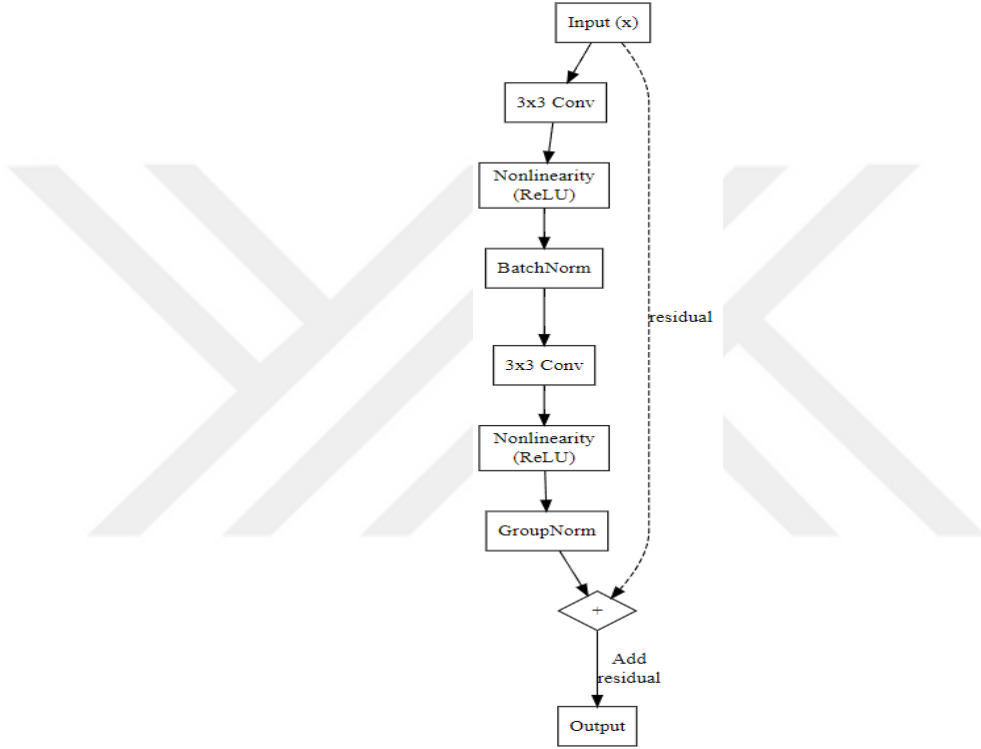
Sonuç olarak, Jaccard İndeksi çok spektrumlu görüntü segmentasyonu için derin öğrenmede bir kilit taşı görevi görür. Yenilikçi DualNorm-UNet çerçevesi içine dahil edilmesi, modelin performansını ve çeşitli alanlardaki uyarlanabilirliğini artırır. Çok spektrumlu görüntü analizi alanı gelişmeye devam ettikçe, Jaccard İndeksi segmentasyon doğruluğundaki ve alan uyarlama stratejilerindeki gelişmeleri sürükleyen temel bir metrik olarak kalmaya devam edecektir.



Şekil 3.1 Çerçevemizin temel yönlerini gösterir - (a) DualNorm_UNetV3 Mimarisi ve Toplu Normalizasyon ve Grup Normalizasyonu ile U-Net'in Eğitim Aşaması (b) Artıklık Blok Tasarımı

3.5 DualNorm_UNetV3 Artıklık Bloğu

Yaklaşımımızın kritik unsuru, **Şekil 3.2'de** gösterildiği gibi, Toplu Normalizasyon (BN) ve Grup Normalizasyonu (GN) temelli normalizasyon akışları içeren DualNorm_UNetV3 Artıklık Bloğu'dur (DNRB).



Şekil 3.2 BN ve GN ile DualNorm_UNetV3 Artıklık Bloğu

Görüldüğü gibi, genel yapı standart bir artıklık bloğunu (He vd., 2016) ReLU etkinleştirme ve her normalizasyon katmanından önce 3x3 evrişim eklendiği şekliyle takip eder. Standarttan farklı olan, sadece Toplu Normalizasyon kullanılması, tasarımımda iki paralel normalizasyon dalı (BN ve GN) ve artıklığı standart artıklık bloğunu diğer bileşenleri aynı tutarak göz önünde bulundurarak tanıtmamızdır. Akımlar giriş özellik haritasını alır ancak normalizasyon işlemlerini uygular.

3.5.1 Bn şubesi

Bu branş, toplu işlem ve uzamsal boyutlar boyunca küresel Toplu Normalizasyon uygular. Giriş aktivasyon haritası X için, C kanal sayısını ve H, W uzamsal yüksekliği ve genişliği belirtirken, BN bağımsız olarak her kanal c için ortalama (ortalama) μ_c ve standart sapmayı σ_c hesapla:

$$\mu_c = \frac{1}{NHW} \sum_{n,i,j} X_{n,c,i,j} \quad (16a)$$

$$\sigma_c = \sqrt{\left(1 - (N * H * W)\right) * \sum_{n,i,j} (x_{c,i,j} - \mu_c)^2 + \varepsilon} \quad (16b)$$

Burada, n toplu içindeki her örneği dizinlerken i, j uzamsal konumları dizinler. ε sayısal hatalardan kaçınmak için küçük bir sabittir. Normalleştirilmiş etkinleştirmeler $\hat{x}_c$ daha sonra öğrenilen ölçek γ_{BNc} ve öteleme β_{BNc} ile kanal başına afirm dönüşüm ile modüle edilir:

$$\hat{x}_c = \gamma_{BNc} * \frac{x_c - \mu_c}{\sigma_c} + \beta_{BNc} \quad (17)$$

Dolayısıyla, BN toplu düzeyinde istatistiklere dayalı küresel bir normalizasyon sağlar. Önerilen DualNorm Artıklık Bloğu, küresel istatistikler için BN aracılığıyla tamamlayıcı normalleştirmeyi mümkün kılar. Sonraki olarak her iki şemayı etkili bir şekilde entegre etmek için eğitim stratejisini tartışacağız.

3.5.2 Çift normalizasyon ile öğrenme

Semantik maskeler test zamanında mevcut olmadığından, küresel ve yerel istatistikleri sırayla dahil etmek için eğitim sürecinde bir DualNorm-unet öneriyoruz: $F(.,\theta)$ ile parametrelendir $\theta = \{ \theta_s, \theta_{NB}, \theta_{NG} \}$. Burada θ_{NB} BN dalında kullanılan öğrenilebilir parametreleri; alt NB ve NG BN ve GN'yi gösterirken θ_s tüm diğer ağ ağırlıkları (parametreler) için gösterir. Sözde maskeler üretmek için yalnızca BN dalında eğitim yapın.

3.5.3 Eğitim hedefi

BN kayıplarını kullanarak θ ağırlıklandırma parametresi ile genel eğitim aşaması: Eğitim sürecinde, modele Batch Normalization (BN) dalı üzerinden eğitim vermeye özel odaklanılır, burada hem θ_s hem de θ_{NB} güncellemelere tabidir. Bu eğitim aşamasının amacı, küresel istatistikler aracılığıyla eğitimi hızlandırmanın ötesine geçmek; aynı zamanda semantik bilgi aktarmak da amaçlanmaktadır. Kesin olarak, eğitim aşamasından üretilen semantik maske $\hat{y}_B$ 16 numaralı denklem olarak ifade edilebilir: Tüm prosedür Algoritma 3.1'de formalize edilmiştir.

$$\hat{y}_B = F(\theta_s, \theta_{NB}, N_G, X) \quad (18)$$

Algorithm 3.1 DualNorm-UNetV3 Training Procedure

Input:

Training images X

Ground truth masks Y

Model parameters $\theta = \{ \theta_s, \theta_{NB}, N_G \}$ (weights and biases)

Output: Optimized θ Algorithm:

1. Initialize network parameters θ

2. Split data into training, validation and test sets

3. **for num of training epochs do:**

3.1 Batch = Trial mini-batch of images and masks (X_{batch}, Y_{batch})

3.2 $\hat{Y}_{batch} = \text{Model}(X_{batch}; \theta)$ // Forward pass

3.3 Loss = Compute loss between $\hat{Y}_{batch}$ and Y_{batch} (e.g. CrossEntropy Loss)

3.4 $\nabla \text{Loss} = \text{Compute gradients w.r.t. } \theta$

3.5 $\theta = \theta - \text{LearningRate} * \nabla \text{Loss}$ // Update parameters via backprop

3.6 Compute metrics like IoU on validation set

3.7 If validation metric stops improving:

end

Return optimized parameters θ

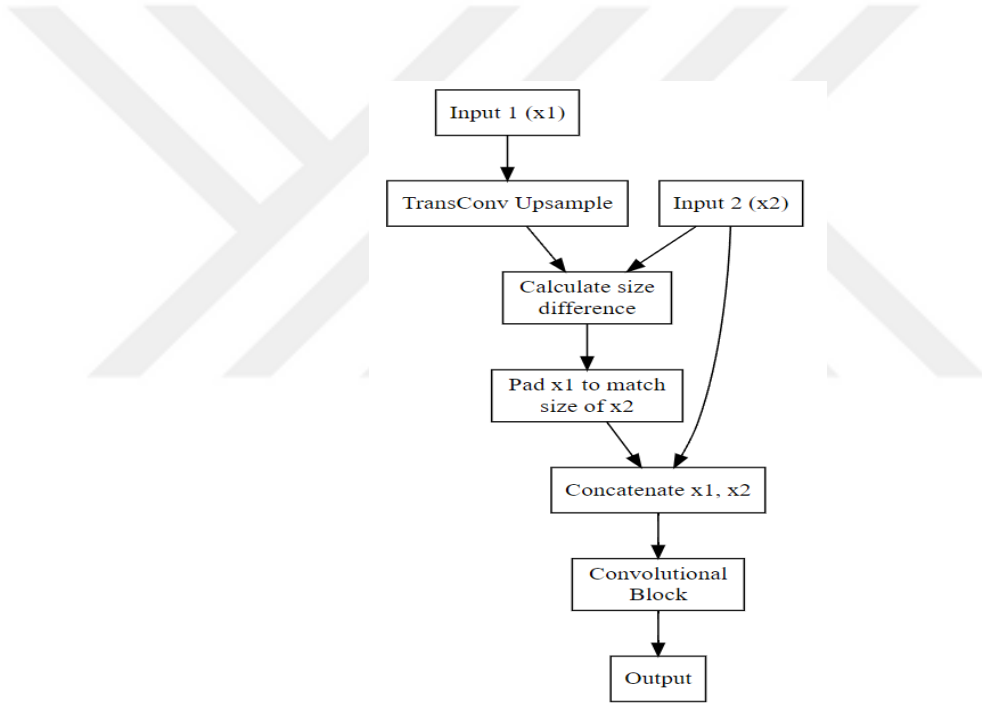
Bu aşama eğitimi, daha iyi çok spektrumlu görüntü temsillerini öğrenmek için küresel ve yerel istatistiklerden artan şekilde yararlanmayı sağlar.

3.6 DualNorm_UNetV3'ün UNet'e Entegrasyonu

Önerilen çift normalizasyon şeması, yaygın olarak kullanılan UNet mimarisine (Ronneberger vd., 2015) entegre edilmiştir. Şekil 8'de gösterildiği gibi, kodlayıcı alt ağı şunları içerir:

1. Standart evrişim blokları
2. DualNorm_UNetV3 Artıklık Blokları

Kod çözücü, yukarı örnekleme için aktarılmış evrişimler kullanarak kodlayıcıyı yansıtır. Atlama bağlantıları kodlayıcıdan çok ölçekli semantik bilgi sağlar.



Şekil 3.3 Yukarı örnekleme modülü akış şeması

Yukarı Örnekleme Modülü, artıklık bağlantılardan yararlanarak gelişmiş bilgi akışını kolaylaştıran genel DualNorm-UNet mimarisinin temel bir bileşenidir. Bu modül, yukarı örneklenecek giriş öznitelik haritalarını ve kodlayıcı alt ağından atlama bağlantılarını alır.

Özellikle, bir girdi tensörü x_1 , uzamsal çözünürlüğü artırmak için aktarılmış evrişime (ayrıca ters evrişim veya kesirli olarak tabakalanmış evrişim olarak da adlandırılır) uğrar.

Aynı zamanda, birleştirme için atlama girişi x_2 ile uzamsal uyumluluğu garanti etmek için dolgu boyutları hesaplanır ve ayarlanır. Dolgu hesaplamaları şunlardır:

- Dikey Dolgu: x_1 ve x_2 arasındaki yükseklik farkını hesaplayın. x_1 'in üst ve alt kısmını yarı farkla dolgu yapın.
- Yatay Dolgu: x_1 ve x_2 arasındaki genişlik farkını hesaplayın. x_1 'in sol ve sağ tarafını yarı farkla dolgu yapın.

Birleştirmeden sonra, yukarı örneklenen tensör x_1 bir artıklık bağlantı olarak tutulurken, birleştirilmiş tensör daha fazla işlemde geçer. İlk olarak, 3×3 evrişim gelişmiş özellikleri çıkarır. Sonraki olarak, bir artıklık blok temsil kapasitesini zenginleştirir.

Eleştirel olarak, tutulan kalan x_1 daha sonra işlenmiş sonuca eklenir. Bu kalan ekleme, modülün yukarı örnekleme işleminden önemli bilgileri korumasını sağlar; bu da geliştirilmiş gradient akışına ve özellik yeniden kullanımına olanak tanır.

Çıktı özellik haritası, hem yukarı örneklenmiş hem de kalan atlamalı bağlantılardan bütünleşik bilgi içerir. Bu, sonraki katmanlara gelişmiş semantik segmentasyon performansı için rafine bir temsil sağlar. Genel olarak, özel dolgu ve kalan ekleme ile Yeniden Örnekleme Modülü, DualNorm-UNet içinde çok düzeyli özellik birleşimini geliştirir; bu da hassas piksel-düzey sınıflandırmaya yardımcı olur. Modül tasarımı, derin sinir ağlarında bilgi iletimini geliştiren temel mimari ilkeleri yansıtır.

3.7 Özetle

DualNorm-UNetV3, Toplu Normalizasyon ve Grup Normalizasyon yoluyla küresel istatistikleri kullanarak çokspektral görüntü segmentasyonu için UNet'i geliştirir. Temel fikirler şunlardır:

- Önerilen DualNorm_UnetV3 Artık Bloğu, görüntü düzeyi istatistiklerini yakalamak için Toplu ve Grup Normalizasyon kullanır.
- Küresel normalleştirmeden faydalanmak için tasarlanmış eğitim prosedürü.
- Encoder yoluna DualNorm_UnetV3 blokları ekleyerek UNet'e entegrasyonu gösterilmiştir.

Özetle, DualNorm-UNetV3, genel mimariyi asgari düzeyde değiştirerek UNet'i tamamlayıcı normalizasyon ile sorunsuz bir şekilde genişletmeyi mümkün kılar. İkili normalizasyon, öğrenilen çokspektral görüntü temsillerinin ayrımını geliştirmek için basit ancak güçlü bir yaklaşım sağlar. Sonraki bölüm, DualNorm-UNetV3'ü çeşitli çokspektral segmentasyon görevlerinde ve veri setlerinde kapsamlı bir şekilde değerlendirecektir. Doğruluk ve sağlamlık, UNet, SegNet ve DualNorm_UNet temelleri ve öncü yöntemlerle karşılaştırılarak analiz edilir.



4. ARAŞTIRMA BULGULARI ve SONUÇLAR

4.1 Giriş

Bu bölüm, önerilen DualNorm-UNetV3 modelinin Uzaktan algılama görüntü segmentasyonu için kapsamlı deneyler sunmaktadır. Performans iki halka açık benchmark veri setinde analiz edilmiştir:

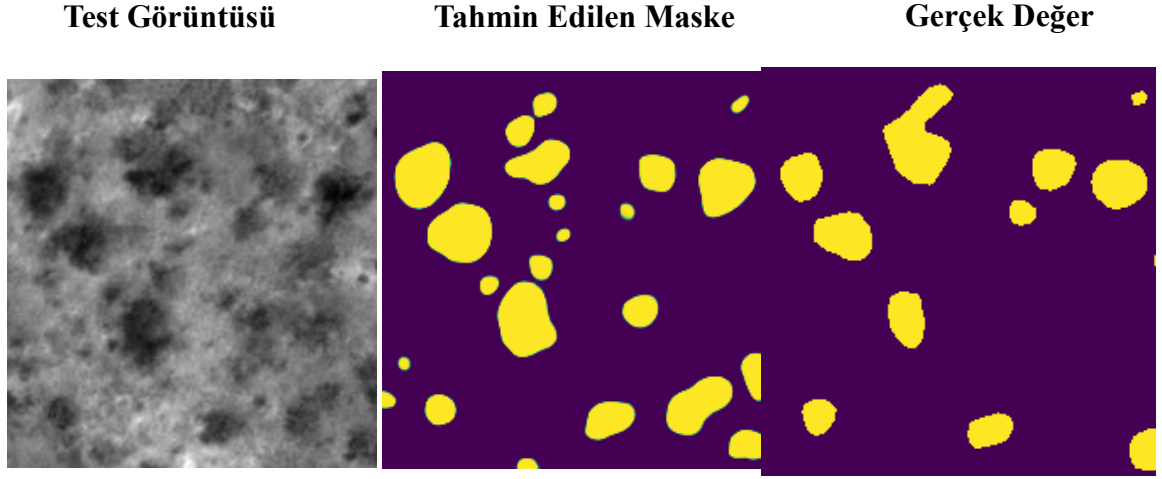
1. Savunma Bilim ve Teknoloji Laboratuvarı (DSTL) Uydu Görüntü Veri Seti
2. Rochester Teknoloji Enstitüsü (RIT)-18 Veri Seti

DualNorm-UNetV3'ü standart UNet ve DualNorm_UNet temel modelleri ve ultra modern yöntemler ile tek alanlı ve çapraz veri seti değerlendirme ayarları altında karşılaştırıyoruz. Ek olarak, deneysel çalışmalar ikili normalizasyon yaklaşımına içgörü sağlamaktadır.

4.2 Veri Kümeleri

4.2.1 Dstl veri seti

DSTL veri seti, DSTL (Hammond vd., 2017) tarafından düzenlenen bir uydu görüntüsü özellik algılama yarışmasının bir parçası olarak yayınlandı. Bu veri setinin tasarımı, 8.000 km²'den fazla alan kaplayan, 0,3 m/piksel mekânsal çözünürlüğe sahip 1 milyardan fazla piksel içeren yüksek çözünürlüklü uydu görüntülerinde nesne algılama ve semantik segmentasyon görevlerine özgüdür. Veri seti, nesnelerin, arazi tiplerinin ve görüntüleme koşullarının çeşitliliği nedeniyle karmaşıklığıyla bilinmektedir.



Şekil 4.1 DualNorm_UnetV3 modellerimiz tarafından uydu görüntülerinden elde edilen, bir Görüntüyü, etiketi ve DSTL görüntü kümelerinin gerçeklerini gösteren illüstrasyon. Semantik segmentasyon için gerçek

4.2.2 Rit-18 veriseti

RIT-18 veriseti, Rochester Teknoloji Enstitüsü kampüsünden hava aracı görüntüleme sistemi (Kemker vd., 2018) kullanılarak toplanmıştır. Veri seti, yüksek çözünürlüklü hava görüntüleri üzerinde SS görevleri için tasarlanmıştır.

Çizelge 4.1 Veri işleme

Step	RIT-18	DSTL
1	1778	5985
2	Resizing (224 x 224)	Resizing (224 x 224)
3	Format (.mat)	Format (.mat)

4.3 Uygulama Ayrıntıları

Bu çalışmada, SS için beş farklı derin öğrenme mimarisini kullanıyoruz. U-Net, SegNet, DualNorm_UNet, DualNorm_UNetV2, DualNorm_UNetV3. İlk ikisi, temel mimari görüntü bölütleme olarak da bilinen iyi bilinen bir mimaridir, üçüncüsü (DualNorm_UNet) son zamanlarda tıbbi görüntülerin ve çeşitli diğer alanların Bölümlendirilmesinde üstün başarı gösteren gelişmiş güncel modellerden biridir. Kalan iki modelimiz, herhangi bir

toplu iş, küçük veya daha büyük boyutlu olsun, tıbbi görüntüler için daha sağlamlık, genelleştirme kabiliyeti ve güvenilirlik kanıtlanan DualNorm_UNet'in değiştirilmiş sürümleridir. Modeller Binary cross entropy loss ile mantık (BCEWithLogitsLoss) kullanılarak eğitilir ve Adam ile optimize edilir.

Uyguladığımız U-Net model eğitiminde, Tablo 4.2'deki hiperparametre ayarları modelinize aktarıldı. Python 3.10.12 ve PyTorch sürüm 2.1.0+cu121 kullanılarak. Yani, PyTorch, CUDA sürüm 11.1 ile derlenmiştir (kısaca "cu121" olarak kısaltılır). U-Net mimarisi, simetrik dört seviyeli bir kodlayıcı-kod çözücü yapısından oluşur. Her derinlik seviyesi iki tekrarlanan evrişimli blok içerir, eğitim sırasında bir BN katmanı tarafından takip edilir ve doğrusal olmayanlık için bir ReLU etkinleştirmesi vardır. Ek olarak, her seviye 2x2 maks havuzlama aşağı örnekleme katmanı (adım 2) ile sona erer ve bir düzenleme bırakma (oran=0.5) vardır. Bu, uzamsal boyutları ilerleyici olarak yarıya indirirken, özellik kanallarının sayısını kodlayıcı yolda simetrik olarak 64, 128, 256 ve 512 şeklinde genişletir.

Kod çözücü yolu, uzamsal çözünürlüğü geri kazanmak için birleştirme ile havuzlamayı değiştirir. Her evrişimli katmanlar durumunda, yatay boyutları bir blok içinde sürdürmek için çekirdek, adım ve dolgu boyutlarını sırasıyla 3x3, 1x1 ve 1x1 olarak ayarlıyoruz. İkili semantik bölümler için, ağ çıktısı piksel başına tespit olasılığını temsil eden tek kanallı girdi boyutlu bir harita oluşturan 1x1 evrişime ve piksel bazlı sigmoid etkinleştirmeye geçirilir.

Uygulanan SegNet, atlama bağlantılarının etkinleştirmeler yerine havuzlama indekslerini aktardığı dışında U-Net'in mimari seçimlerini yinelemektedir. İndis ızgarası içinde interpolasyon yoluyla yukarı örnekleme ile SegNet, U-Net'in transpoze evrişimlerine kıyasla verimli bellek kullanımı sunmaktadır (Ulku vd., 2022). Çok spektral veriler için bunların ilgili kod çözücü yukarı örnekleme şemalarının ve sonuçtaki segmentasyon sonuçlarının karşılaştırmalı bir analizini seçtik. Bununla birlikte, deneyimizde, SegNet, DSTL görüntü setinin ikili sınıflandırmasını tahmin etmekte başarısız olmuş gibi görünüyor ve eğitimde daha yüksek hesaplama yükü aldı.

Son olarak, Çoklu Spektrum Görüntü Bölümlemesi için çift normalizasyon kullanarak derin öğrenme modelini eğitmek için deneyler yürüttük, özellikle önerdiğimiz DualNorm_UNetV3 mimarisi yapısını kullanarak. Bu mimari, farklı derinliklerde dört aşağı örnekleme bloğu ve dört yukarı örnekleme bloğuna sahiptir. Ağ, bir Inconv ve darboğaz olarak bilinen orta blok ve Kalıntı bloğu içerir. Her aşağı örnekleme bloğu, 2x2'lik 2 adımlık bir maks havuzlama katmanı ve ardından bir evrişim katmanı, karmaşık desenleri yakalamak için Evbloğu oluşan evrişim katmanı, doğrusal katman (ReLU) etkinleştirme, Toplu Normalizasyon (BN) , evrişim katmanı, ReLU ve Grup Normalleştirilmesinden (GN) oluşur. Her yukarı örnekleme bloğu veya kod çözücü, bir transpoze evrişim katmanı ve ardından başka bir evrişim katmanından oluşur. Tüm örnek bloklarına oranı=0,1 olan bir bırakma katmanı dahil edildi. Bu düşürme eğitim sırasında giriş birimlerini rastgele sıfırlar ve ağırlık eksik bilgilere dahili olarak işlemlerini ve kayıp birimlere dayanıklı olmasını zorlar (Wang vd., 2019). Bu, ampirik olarak da tıbbi ve diğer görüntü bölümleme için doğrulandığı gibi, genelleştirmeyi önemli ölçüde geliştirir (Işın vd., 2016). Hiperparametre p, her girdi için bağımsız olarak birimleri bırakma olasılığını kontrol eder, burada her yinelemede rastgele%10 birimin atıldığını belirten 0,1 olarak sabitlenmiştir. Böyle bir düzenleme, güvenilir bölümleme dağıtımına yardımcı olur ve aşırı uyumu hafifletir. Bu tür bir düzenleme, sağlık hizmetleri ve diğer alanlarda veri kümesi kısıtlamalarının modaliteler ve uygulamalar arasında önemli ölçüde farklılık gösterdiği uyarlanabilir kapasite İç katmandaki bir evrişim katmanı ile üst katmanda tek çıkış kanalı vardır. Modelimiz, Kalıntı Bloklar, Transpoze Evrişim Katmanları (Yukarı Örnekleme), Toplu Normalizasyon, ReLU etkinleştirmeleri, Grup Normalleştirilmesini içerir. Ek olarak, bölümleme görevleri için özelleştirilmiş son (çıkış) katmana sigmoid etkinleştirme uygulandı.

DualNorm_UnetV3, DualNorm_UNet mimarisinin geliştirilmiş bir versiyonu, kodlama ve kod çözme özellik haritalarında atlama bağlantıları tanıtır, girdi görüntülerinde kesin nesne yerelleştirmesini kolaylaştırarak bölümleme performansını artırır. Özellikle değişken toplu iş boyutlarıyla karşılaşıldığında kararlılığı sağlamak için hem BatchNorm (norm_layer1) hem de GroupNorm (norm_layer2) kullanıyoruz, tıbbi görüntüleme veri kümelerinde yaygın bir zorluk. BatchNorm, toplu istatistikleri kullanarak etkinleştirmeleri normalize eder. Derin sinir ağı mimarilerinde son derece popüler olmasına rağmen, küçük toplu iş boyutlarıyla performansı önemli ölçüde düşer (Isensee vd., 2021), sınırlı hasta verileri

nedeniyle tıbbi görüntüleme veri kümelerinde sıklıkla karşılaşılır. GroupNorm, kanalları gruplara bölerek ve her grup içinde bağımsız olarak normalize ederek bunu ele alır. Bu, toplu boyut değişikliğine karşı dayanıklılık sağlar. Bununla birlikte, çalışmalar, daha büyük toplu boyutlarda BatchNorm için üstün performans göstermektedir, çünkü daha büyük örnek boyutlarından daha güvenilir istatistik tahminleri elde edilir (Wu vd., 2018). Buradaki çift normalleme şeması, her iki niteliğin de tadını çıkarmak için amacıyla - GrupNorm küçük toplular için kararlılık, ve yeterli örnek mevcut olduğunda BatchNorm'a geçiş. Bu uyarlanabilir kapasite, sağlık hizmetleri ve diğer alanlarda veri kümesi kısıtlamalarının modaliteler ve uygulamalar arasında önemli ölçüde farklılık gösterdiği için değerlidir.

Eğitim, her biri bir görüntü ve ona karşılık gelen bölümlene maskesini içeren eşleştirilmiş görüntüler kümesi üzerinde gerçekleştirildi. Ağ, giriş görüntülerine dayalı olarak bölümlene maskelerini tahmin etmeyi öğrendi. Model, verideki karmaşık desenleri yakalayabilecek kadar karmaşık bir mimariye sahip olduğunu gösteren toplam 16.416.417 öğrenilebilir parametre ile donatılmıştı, orijinal DualNorm_UNet'teki 16.416.353 toplam öğrenilebilir parametreyi biraz aşıyordu. Tahmini bellek parametre boyutu 65,57 MB'dır. Çizelge 4.2, modellerimizin hiperparametre yapılandırmasını göstermektedir.

Çizelge 4.1 Deney ayarları

<i>Config</i>	<i>Value</i>
<i>Total Input</i>	5895
<i>Fold No</i>	4
<i>Fold Size</i>	4
<i>Val Ration</i>	0.1
<i>Batch Size</i>	16
<i>Number of epochs</i>	100
<i>Learning rate</i>	1e-2
<i>Optimizer</i>	Adam
<i>Training Loss</i>	BCEWithLogitsLoss
<i>Validation Loss</i>	BCEWithLogitsLoss
<i>Accuracy</i>	Jaccard
<i>Initialization</i>	Xavier_uniform_
<i>Step_Size</i>	10
<i>Gama</i>	0.9
<i>Input Size (lim)</i>	244
<i>Model type</i>	DualNorm_UnetV3
<i>Channel index (3)</i>	RGBs (244 x 244 x 3)
<i>Transfer type</i>	notr

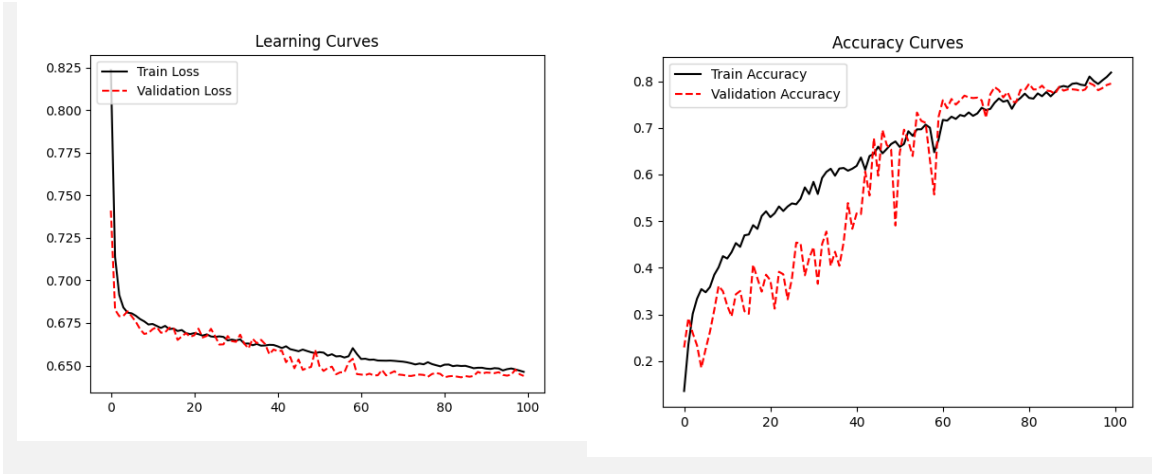
Çalışmamızda, eğitim, doğrulama ve test setlerine ayrılmış DSTL veri seti kullanılmaktadır, 5985 örnek ile 4041 eğitim görüntüsünü (yani% 67,49) ve 448 doğrulama görüntüsünü (% 7,49) ve kalanını (% 25,02) test için içerir. 1778 örnek içeren RIT-18 veri kümesi, 1201 eğitim görüntüsünü ve 133 doğrulama görüntüsünü içermektedir. 4 katmanlı çapraz doğrulama yaklaşımı kullandık. Her görüntü setine ve biçim özelliklerine uyan hesaplanan görüntü parçalarının sayısı tablo 4.1'de görülebilir. Eğitim sırasında, [16, 3, 224, 224] giriş şekli kullandık, 16 örnek/parti, RGB görüntüler için üç kanal ve yükseklik ve genişlik için 224x224 parça boyutlarını ifade etmektedir. [16, 1, 224, 224] çıktı şekli, her biri 16 örnek için bir kanalı olan ikili segmentasyon haritalarını temsil eder.

Deneylerimizi, PyTorch derin öğrenme çerçevesini kullanarak DualNorm_UNet model mimarisini uygulayarak gerçekleştirdik. Hiperparametre ayarlama çapraz doğrulama için 4 katman kullandı. Ağı eğitmek, bir Adam optimizörü kullanmayı ve 0,001'e ayarlanmış bir öğrenme oranını içeriyordu. Doğrulama performansı için kullanılan kayıp fonksiyonu, Binary Cross-Entropy with Logits (BCEWithLogitLoss) idi. 16'lık bir toplu boyut kullanıldı ve model 100 dönem için eğitildi. Bu yapılandırmaların bir özeti için Tablo 4.2'ye bakın.

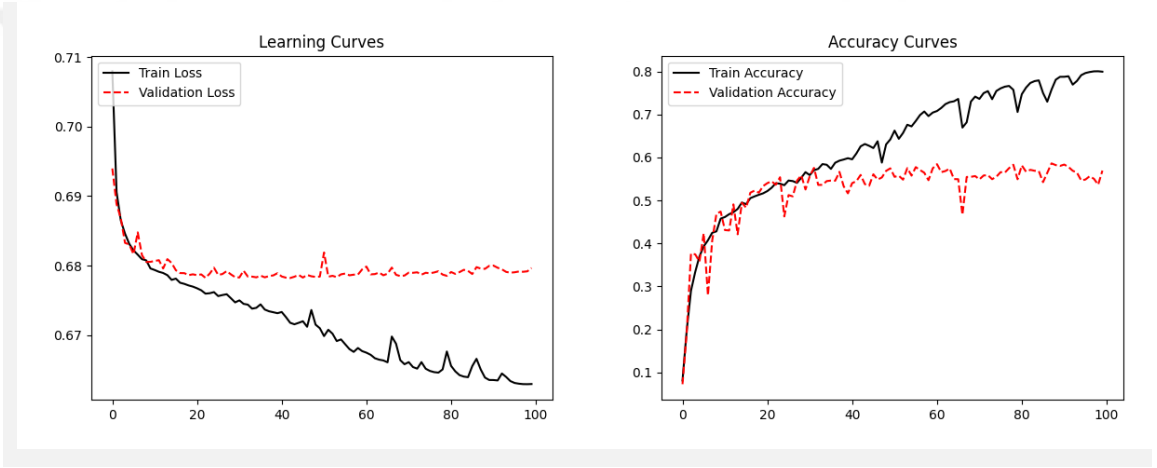
4.4 Donanım

Modellerimizi eğitmek için kullandığımız, NVIDIA **Tesla T4 GPU**, **CUDA sürüm 121** ve **V100 GPU** kullanan bir Google Cloud Platformu idi. Toplam GPU Belleği: **15360MiB**. Eğitim süresi DSTL görüntü kümeleri için tüm modeller için yaklaşık 1:10 saatti. SegNet için 2 saatten fazla sürmekte. RIT-18 veri setleri deney başına 17 ila 30 dakika arasında sürmektedir.

RIT-18 Görüntü Setleri

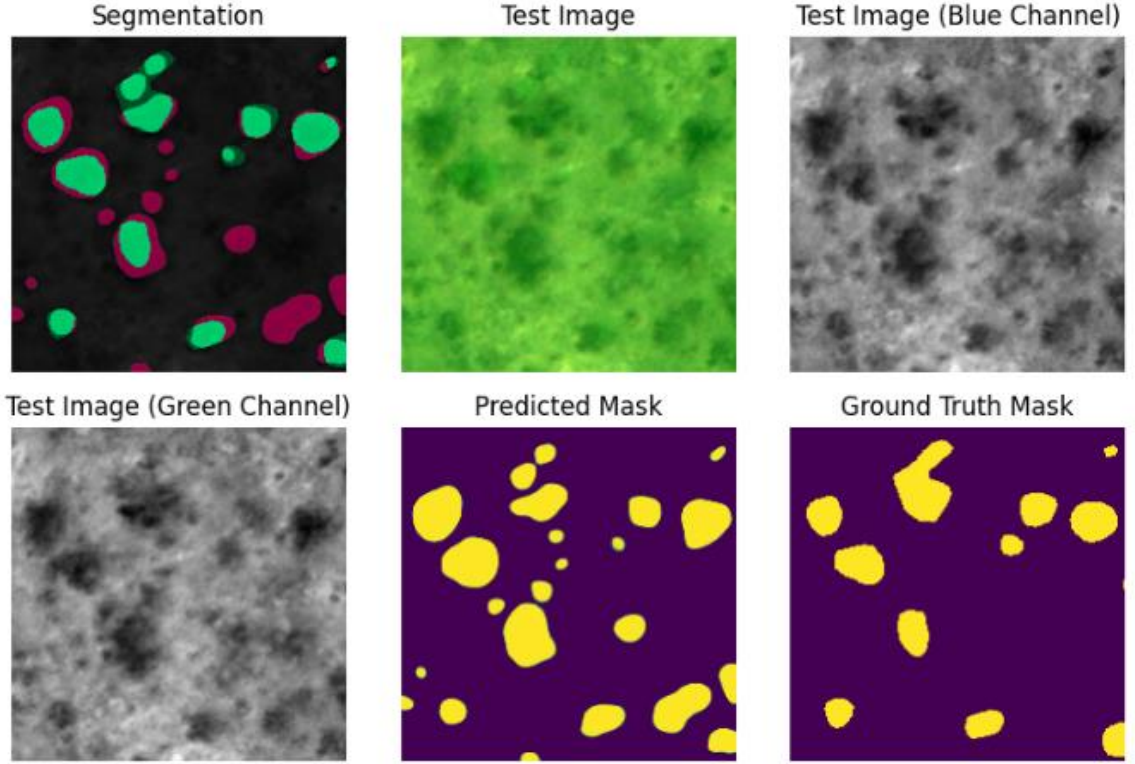


DSTL Görüntü Setleri



Şekil 4.2 Önerilen modelin 100 epok boyunca öğrenme ve doğruluk eğrileri

Burada, iki farklı görüntü kümesi için BN, GN, kalıntı ve Dropout'un metrik eğrileri Şek.4.3'te karşılaştırmalı olarak gösterilmiştir. Kayıp oranı ardışık eğitim yinelemelerinde 0,825 ile başlarken, optimizasyon davranışına ilişkin içgörüler sağlar. İki çizgi gösterilmektedir - katı siyah eğitim kaybı ve kesikli kırmızı doğrulama kaybı. X eksenini 20'lik artışlarla 0-100 ölçeğinde yineleme miktarını gösterir. Y eksenini, RIT-18 için kaybı 0,650-0,825 aralığında 0,025'lik adımlarla gösterir, kayıp oranının 0,25'e düştüğü görülmektedir. Bu, modelin eğitim verilerine sığmayı hızlı bir şekilde öğrenmesini yansıtır. Her iki eğri üst sol köşede başlar ve yüksek başlangıç kayıplarını gösterir.



Şekil 2 Nitel Değerlendirme. DSTL Verisetinden segmentasyonları görebiliriz, iyileşme olduğunu görebiliriz

4.5 Deneysel Prosedürler ve Sonuçlar

Öncelikle, her veri setindeki segmentasyon doğruluğunu ayrı ayrı değerlendiriyoruz. DSTL veri seti için, Tablo 4, DSTL veri setlerinde dört katmanlı çapraz doğrulamadan elde edilen ortalama Jaccard Benzerliklerini karşılaştırır. Test edilen tüm modeller arasında DualNorm_UNetV3 en iyi performansı **0,570464253** ile elde etmiştir. Bir sonraki en yakın performans, orijinal DualNorm_UNet'ten gelen **0,548739552** ortalama ile devam eder, ardından **0,522514939** Jaccard İndeksine ulaşan U-Net mimarisidir. Ancak, çoklu deneylerde önerilen modelimiz DualNorm_UNetV3 (BN, GN ile ikili Normalizasyon ve kalıntı ile) **0,575641632** Jaccard İndeksi ile iyileştirilmiş bir performans elde etti. Modelimizin güçlü yönlerinin, daha fazla GPU kaynağı olan yerel makinelerde deneylerle daha da artabileceğine inanıyoruz.

RIT-18 görüntü setlerinde, U-Net dört katlı çapraz doğrulamada ortalama **0,762541935** Jaccard indeks değeri elde etmiştir. DualNorm-UNetV3, ikinciden en iyi performans

ortalaması olan 0,739932328'i elde etmiş, ardından 0,716670483 ile DualNorm-UNetV2 gelmiştir. Temel DualNorm-UNet **0,710675836** elde ederken, SegNet en düşük performans olan 0,328384227 ile geldi. Ancak, Tablo 4.5, RIT-18 üzerinde beş katmanlı çapraz doğrulama kullanarak modelimiz için kayda değer bir performans iyileştirmesi göstermektedir. Burada DualNorm-UNetV3, test Jaccard benzerlikleri ortalamasında **0,788900793**'e ulaşmış, **0,779622853** temel DualNorm-UNet ve **0,760589421** DualNorm-UNetV2 ile karşılaştırılmıştır. U-Net **0,759975731** elde etmiştir. **0,73987688** en düşük sonucu gösteren SegNet hariç, RIT-18 üzerindeki segmentasyon performansı tüm modeller için tutarlı görünüyor. Şekil 4.6, semantik segmentasyon çıktılarını göstermektedir - U-Net, SegNet ve DualNorm_UNet, nesne şekillerini kesin olarak ana hatlarıyla belirlemekte zorlanırken, DualNorm_UNetV2 ve DualNorm_UNetV3 tüm sınırları doğru şekilde ana hatlarıyla belirler, segmentasyon doğruluğunu artırır.

Çokspektral giriş türleri kullanılarak yapılan deneylerden elde edilen sonuçlar Tablo 4.5'de sunulmuştur. Katlarda ufak değişikliklerle çeşitli bant kaynaştırma modülleri uygulamaya rağmen, bu çalışmanın motivasyonu orijinal DualNorm_UNet mimarilerine dayalı olarak rafine bir model mimarisi uygulamaktır. Bu nedenle, bu çalışma deneylerinde yalnızca U-Net, SegNet ve DualNorm_UNet'i kullanmaktadır. Araştırılan mimari kümesinin kısıtlı olmasına rağmen, sonuçlar üç kanal giriş türlerini içeren deneylerde gözlenenlerle tutarlıdır. Özellikle, SegNet giriş türü hem DSTL hem de RIT-18 görüntü setleri genelinde en az tercih edilen performansı göstermektedir. Bu gözlem, giriş kanallarının sayısındaki salt bir artışın segmentasyon performansında önemli bir iyileşme sağlamadığını; bunun yerine hesaplama taleplerini artırdığını göstermektedir. Ancak, kritik faktör bu giriş kanallarının en iyi karışımını iyileştirilmiş segmentasyon sonuçları için belirlemektir.

Mevcut literatür, çokspektral giriş kullanan kapsamlı uçtan uca bir yöntemle derin öğrenme modellerini eğitmenin zorluğuna, özellikle uzaktan algılama verilerinin yetersiz kullanılabilirliğine ve tıbbi görüntü setlerinde olduğu gibi çeşitli alanlara işaret etmektedir. Uçtan uca eğitimi ele alma ve çokspektral görüntülere karşı sağlamlığı artırma ihtiyacı, bu çalışmada ele alınan temel araştırma sorularından birini oluşturmaktadır.

Sonuç olarak, model varyantlarımızın bu ön araştırması, Normalizasyon ve düzenleme tekniklerinin performans sınırlamalarını ele almadaki faydasını göstermektedir.

Çizelge 4.2 RIT-18 Veri kümesi segmentasyon sonucu için ortalama sonuçlar (Jaccard İndeksi olarak)

<i>Deney</i>	<i>AVG (Test Doğruluğu)</i>	<i>Kayıp</i>	<i>Zaman (saniye)</i>	<i>Dakikalar</i>
UNetV2	0.762541935	0.644255734	3740.99408	62.34990
SegNet	0.704384227	0.670271364	9711.80985	64.78899272
DualNorm_UNet	0.710675836	0.647868205	3382.69457	56.37824197
DualNorm_UNetV2 (Residual)	0.716670483	0.654214732	3216.55504	53.60925112
DualNorm_UNetV3 (BN+GN)	0.739932328	0.648875562	1050.64200	17.51069959

Çizelge 4.3 DSTL Veri Kümesi için ortalama sonuçlar (Jaccard İndeksi olarak)

<i>Deney</i>	<i>AVG (Test Doğruluğu)</i>	<i>Kayıp</i>	<i>Zaman (saniye)</i>	<i>Dakikalar</i>
UNetV2	0.522514939	0.672632444	8174.886859	136.2481143
SegNet	0.492148622	0.693147123	20019.1934	222.6039705
DualNorm_UNet	0.548739552	0.667240133	3408.342957	56.80571595
DualNorm_UNetV2 (Residual)	0.516597122	0.661553401	4636.612089	48.8055818
DualNorm_UNetV3 (BN+GN)	0.570464253	0.663402817	3432.31556	57.20525937

Çizelge 4.4 Her iki küme için elde edilen Optimum Sonuçlar RIT-18 görüntü kümeleri için sırasıyla 4 ve 5 katlama çapraz doğrulama kullanan “Üç Kanal Girişi” sınıfı

<i>Deney</i>	<i>DSTL</i>	<i>RIT-18</i>	<i>RIT-18</i>
UNetV2	0.529246807	0.77577126	0.759975731
SegNet	0.492158624	0.733423419	0.739876688
DualNorm_UNet	0.550038755	0.767092526	0.779622853
DualNorm_UNetV2 (Residual)	0.545063257	0.72369003	0.760589421
DualNorm_UNetV3 (BN+GN)	0.575641632	0.76800555	0.788900793

4.6 Değerlendirme Ölçütü

Modellerimizin kapsamlı değerlendirmesi için, standart çapraz doğrulama protokollerinin altın standartlarına (Zhou vd., 2017) bağlı kalıyoruz. Bu sağlam değerlendirme çerçevesinde, her veri seti, verilerin çeşitliliğinin kapsamlı bir temsilini sağlayacak şekilde, dört (4) birbirini dışlayan katmana titizlikle bölünür. 4 katmanlı çapraz doğrulama yöntemini kullanarak, modellerin performansını titizlikle değerlendiririz. Segmentasyon kalitesini ölçmek için iki kritik ölçütten yararlanıyoruz: Jaccard Index, tahmin edilen ve gerçek segmentlerin kesişimini yakalar, mekânsal uyuma ilişkin ince bir anlayış sunar ve BCEWithLogitsLoss, tahmin edilen ve gerçek piksel değerleri arasındaki farkı nicelendiren gelişmiş bir metrik, segmentasyon doğruluğuna ve piksel düzeyinde hizalamaya ilişkin taneli bir değerlendirme sunar.

Çizelge 4.5 RIT-18 için dört (4) Çapraz doğrulama kullanılarak çoklu deneysel sonuç

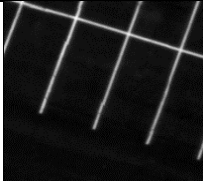
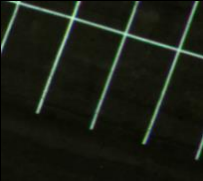
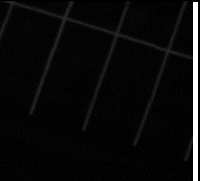
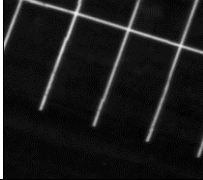
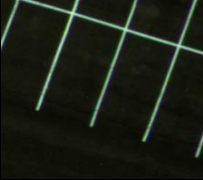
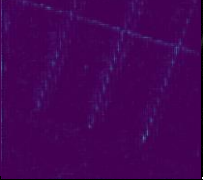
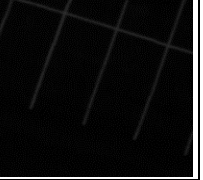
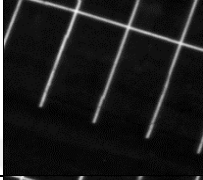
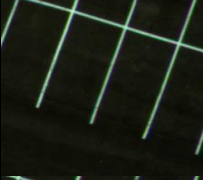
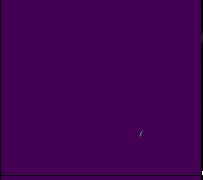
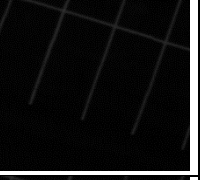
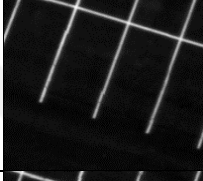
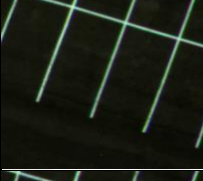
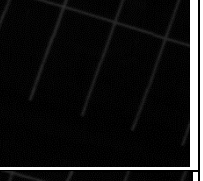
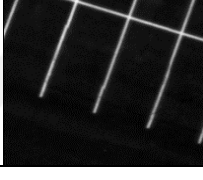
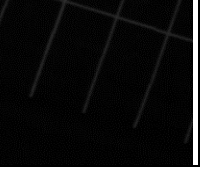
Deney	Test Sonuç	Kayıp	Zaman (saniye)	Dakikalar
	0.678322315	0.650083	1048.653117	17.47755
	0.757011592	0.6488876	1051.688481	17.52814
	0.76800555	0.6488043	1051.914498	17.53191
	0.756389856	0.6477274	1050.311922	17.50520
Ortalama	0.739932328	0.6488756	1050.642005	17.5107

Çizelge 4.6 DSTL için dört (4) katlı çapraz doğrulama kullanılarak çoklu deneysel sonuç

Deney	Test Sonuç	Kayıp	Zaman (saniye)	Dakikalar
	0.575641632	0.6629658	3431.974117	57.19957
	0.565286875	0.6638398	3432.657007	57.21095
Ortalama	0.570464254	0.6634028	3432.315562	57.20526

	Test Resmi	Test Resmi G	Zemin Gerçeği Maske	Tahmini Maske	Segmentasyon
U-Net					
SegNet					
DualNorm_U-Net					
DualNorm_U-NetV2 (Ours)					
DualNorm_U-NetV3 (Ours)					

Şekil 3 Uygulanan farklı model mimarisi ile gerçekleştirilen tüm deneyler için semantik segmentasyon sonuçlarını sunar. Sonuçlar, DSTL görüntü kümelerinden seçilen 224x224 piksel çözünürlüklü bölgelerde (Y x G) gösterilmektedir. Model adları ilk satır ve sütunda belirtilmiştir. Semantik segmentasyon sonuçlarında, yeşil bölgeler isabetleri gösterirken, pembe bölgeler yanlış alarmları gösterir

	Test Resmi	Test Resmi G	Zemin Gerçeği Maske	Tahmini Maske	Segmentasyon
U-Net					
SegNet					
DualNorm_UNet					
DualNorm_UNetV2 (Ours)					
DualNorm_UNetV3 (Ours)					

Şekil 4 Çeşitli model mimarileriyle gerçekleştirilen tüm deneyler için semantik segmentasyon sonuçlarını, RIT-18 Görüntü Setlerinde seçilen 224x224 piksel çözünürlüklü bölgelerde (Y x G) temsil eder. Birinci satır ve sütun model adlarından oluşur. RIT görüntü seti için uygun yer gerçeği etiketlerin yokluğu nedeniyle modelin tahmin etmesi zor

4.7 Tartışma

Hem tek veri seti hem de çapraz veri seti ayarları altındaki tutarlı kazançlar, çokspektral görüntü segmentasyonu için ikili normalizasyonun faydalarını doğrular. Yerel istatistikler, genelleştirmeyi de iyileştirirken semantik yapıları daha iyi ayırt etmeye yardımcı olur. DualNorm_UNetV3 bloklarını erken kodlayıcı aşamalarına yerleştirmek en etkilidir.

Gelecekte, modeli hacimsel 3B verilere genişletmek hiperspektral görüntü analizi için değerli olabilir. Vision Transformers gibi gelişmiş kodlayıcıları keşfetmek bir diğer umut verici yöndür.

Öğrenilen temsillerin daha derinlemesine bir analizi, tamamlayıcı istatistiklerin derin sinir ağlarını nasıl faydalandırdığına dair içgörüler sağlayabilir.

4.8 Özet

Bu bölüm, DualNorm-UNetV3'ü Uzaktan algılama görüntü segmentasyonu için kapsamlı deneyler sunarak aşağıdaki sonuçlara varmıştır:

- DualNorm-UNetV3, hem tek veriseti hem de çapraz veriseti senaryoları altında UNet, DualNorm_UNet ve SegNet'e kıyasla önemli ve tutarlı kazançlar elde eder.
- Bölge tabanlı istatistikler veriseti kaymalarını ele almada yalnızca küresel normalleştirmeye kıyasla daha etkilidir.
- DualNorm_UnetV3 bloklarını erken kodlayıcı katmanlarına yerleştirmek, doğruluk ve hesaplama overheadi arasında optimal dengeyi sağlar.
- Yerel istatistiklere dayalı olarak veriseti dağılımlarını hizalamak, çapraz veriseti performansını artırır.

Genel olarak, önerilen ikili normalizasyon stratejisi, Uzaktan algılama görüntü temsillerinin sağlam öğrenilmesini mümkün kılar ve mevcut yöntemleri geride bırakır. Dahası, model eğitim ayrıntılarının ve kullanılan ekipmanın kapsamlı bir hesabı verilmiştir. Daha sonra, deneylerde kullanılan tüm veri kümeleri, yani DSTL ve RIT-18'e dair bir giriş sunuldu, bu veri kümelerinden gelen örnek görüntülerle birlikte. Bu, hem model eğitimi hem de test aşamaları için kullanılan veri setlerine bütüncül bir bakış açısı sunmak için hizmet etti. Ek olarak, model sonuçlarının analizi gerçekleştirildi ve hem baz model hem de yer gerçeği ile karşılaştırmalı bir değerlendirme sunuldu.

5. SONUÇLAR ve GELECEK YÖNELİMLER İÇİN POTANSİYEL

5.1 Özet ve Katkılar

Bu tez, özellikle uzaktan algılama görüntüleri için yeni alanlara uyum sağlamakta model semantik segmentasyon modellerini uyarlamanın kritik zorluğuna cevap vermek için DualNorm-UNetV3 adı verilen öncü derin öğrenme yaklaşımını sundu. Standart evrişimli sinir ağları, eğitim ve test verileri özdeş dağıtıldığında yüksek doğruluk elde etse de, alan kayması altında performansları düşmektedir.

DualNorm-UNetV3'ün temel yeniliği, alan-değişmez temsilleri çıkartmak için ağ eğitimi sırasında tamamlayıcı küresel ve yerel normalizasyon şemalarını entegre etmektir. Temel katkılar şunlardır:

- Çok düzeyli bölgeye özgü ve görüntü genelinde istatistikleri yakalamak için paralel Toplu Normalizasyon ve Grup Normalizasyon ile DualNorm_UnetV3 Kalıntı Bloğu tasarladık.
- Her iki normalizasyon tekniğinden faydalanmak ve model genelleştirmesini artırmalı olarak geliştirmek için iki aşamalı bir eğitim prosedürü geliştirdik.
- Minimal değişikliklerle yaygın olarak kullanılan UNet mimarisine sorunsuz entegrasyonunu gösterdik.
- Hem tek alanlı hem de çapraz alan ayarları altında iki kamuya açık çokspektral görüntü veri seti üzerinde kapsamlı bir şekilde değerlendirdik.
- Standart UNet ve DualNorm_UNet'e göre tutarlı segmentasyon doğruluğu iyileştirmeleri elde ettik ve diğer ultra modern alan uyarlama yöntemlerini geçtik.

Küresel ve yerel istatistikleri eşzamanlı olarak kullanarak, DualNorm-UNetV3 segmentasyon modellerini yeni görüntüleme koşullarına uyarlamak için alan-değişmez özellikler öğrenmede önemli bir vaat gösterdi.

5.2 Sınırlamalar ve Gelecek Çalışmalar

İlerleme sunarken, gelecekteki çalışmaların ele alabileceği sınırlamalar mevcuttur. İlk olarak, 2B modelini 3B'ye genişletmek, mekansal bağlamı ve hiperspektral verileri daha iyi kullanabilir. İkinci olarak, DualNum_UNetV3'ten daha gelişmiş kodlayıcı-kod çözücü mimarileri performansı daha da artırabilir. Üçüncüsü, sokak sahneleri gibi alanlar için doğal görüntü veri setlerinde daha geniş uygulanabilirliği değerlendirmek faydalı olurdu. Ek olarak, çekişmeli öğrenme gibi açık alan uyarlama tekniklerini birleştirmek tamamlayıcı faydalar sağlayabilir. Daha büyük ölçekli testler, model sağlamlığını da güçlendirecektir. Son olarak, dağıtımdan önce hava sistemlerinde verimliliği değerlendiren gerçek dünya alan çalışmaları gereklidir.

Bu araştırmayı geliştirmek için potansiyel yönelimler şunları içerir: hacimsel bağlamı dahil etmek için verimli 3B sürümler geliştirmek; gelişmiş CNN'ler için uyarlama yeteneklerini etkinleştirmek üzere özelleştirilmiş bloklar tasarlamak; farklı doğal görüntü alanlarına genelleştirmeyi değerlendirmek; gelişmiş alan hizalaması için çekişmeli ağlarla ortak eğitim; ve çeşitli edinim koşulları ile büyük ölçekli çok kaynaklı bir veri seti montajı. Bu yüksek potansiyelli alanları takip etmek, sınırlamaları ele almaya ve alanı ilerletmeye yardımcı olabilir.

Özetle, umut verici olmasına rağmen, model mimarisi, veri seti ölçeği ve çeşitliliği, uyarlama teknikleri ve gerçek dünya testiyle ilgili sınırlamalar, bu segmentasyon gelişmelerini arazi dağıtımına doğru çevirmek için etkili gelecek çalışmalar için fırsatlar sunmaktadır.

5.3 Tartışma

DualNorm-UNetV3'ün temel bir avantajı, birleşik bir mimari içinde hem küresel hem de yerel ilişkilerin ince modellenmesine izin veren tamamlayıcı istatistiksel normalizasyon perspektifleri sağlamasıdır. Toplu Normalizasyon ve Grup Normalleştirme'yi birleştirmek, alan adaptasyon için kritik olan hem küresel hem de yerel ilişkilerin ince modellenmesini mümkün kıldı.

Dağılım kayması altında tutarlı segmentasyon kazançları, model genelleştirmeyi geliştirmek için çok düzeyli istatistiksel modellemenin önemini doğrulamaktadır. Önerilen yaklaşım, varolan mimarilere modüler bir şekilde alan uyarlama yetenekleri eklemek için esnek bir çerçeve sağlar.

Kavramsal olarak, bu çalışma istatistiksel değişmez temsillerin öğrenilen bilgiyi alanlar arasında aktarmadaki benzersiz rolü hakkında özel içgörüler ortaya koymaktadır. Böyle değişmezliği teşvik etmek için uzmanlaşmış bloklar ve eğitim stratejileri geliştirmek, derin öğrenme araştırmasını ilerletmek için umut verici bir yön gibi görünmektedir.

5.4 Sonuç

Çalışmada, çeşitli Uzaktan algılama görüntü seti kullanılarak ikili maske sınıflandırması için geliştirilmiş bir DualNorm_UNetV3 modeli benimsenmiştir. Görüntü setleri, CNN tabanlı U-NET, SegNet, DualNorm_UNet ve DualNorm_UNetV3'te uygulanmış ve değerlendirilmiştir. Çeşitli mimariyi uygulamak ve performansı değerlendirmek için PyTorch çerçevesi ve Python dili kullanılmıştır. DualNorm_UNetV3, eğitimde Uzaktan algılama görüntülere karşı daha iyi performans artışı ve dayanıklılık kanıtlamıştır.

224 x 224 piksellik bir yama boyutu, deneydeki tüm modeller için uygun bir pencere boyutu olarak kullanılmıştır. Jaccard Index ve Binary Cross Entropy with Logits kayıp kriterleri, önerdiğimiz modelin etkinliğini temellerle kıyaslamak için kullanılmıştır.

Ayrıca tez, derin ağlar içinde tamamlayıcı normalizasyon şemaları aracılığıyla eşzamanlı olarak küresel ve yerel istatistiklerden yararlanmanın yeni bir bakış açısını sundu. Sonuçta elde edilen DualNorm-UNetV3 modeli, alan kaymaları altında öncü Uzaktan algılama görüntü segmentasyonu elde eder. Daha geniş anlamda, çalışma model genelleştirmesini geliştirmek için uzmanlaşmış algoritmik blokların ve eğitim stratejilerinin potansiyelini vurgular ki bu, derin öğrenmenin çeşitli gerçek dünya bağlamlarında dağıtıldığında sadece önem kazanacaktır. Ek araştırma garanti edilmekle birlikte, aydınlatılan temel ilkelerin uyarlanabilir ve sağlam yapay zeka sistemleri geliştirmek için güçlü bir temel sağladığına inanıyoruz.

KAYNAKLAR

- Alex, Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advance in Neural Information Processing Systems* 25(2) DOI:10.1145/3065386
- Aresta, G., Araújo, T., Kwok, S., Chennamsetty, S. S., Safwan, M., Alex, V., ... & Campilho, A. (2019). Bach: Grand challenge on breast cancer histology images. *Medical image analysis*, 56, 122-139.
- Audebert, N., Le Saux, B., & Lefèvre, S. (2016). Semantic segmentation of earth observation data using multimodal and multi-scale deep networks. *In Asian conference on computer vision* (pp. 180-196). Springer.
- Ba, J.L., Kiros, J.R., & Hinton, G.E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*. <https://arxiv.org/abs/1607.06450>
- Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, DOI: 10.1109/TPAMI.2016.2644615
- Bioucas-Dias, J. M., Plaza, A., Dobigeon, N., Parente, M., Du, Q., Gader, P., & Chanussot, J. (2013). Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches. *IEEE journal of selected topics in applied earth observations and remote sensing*, 5(2), 354-379.
- Chang, J., Guo, C., Wang, Z., & Yu, Y. (2019). Understanding the Disharmony Between Dropout and Batch Normalization by Variance Shift. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI:10.1109/CVPR.2019.00279
- Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 801-818). <https://arxiv.org/abs/1802.02611>
- Chen, J., Lu, Y., Yu, Q., Luo, X., Adeli, E., Wang, Y., Lu, L., Yuille, A. L., & Zhou, Y. (2021). TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation. *ArXiv*, abs/2102.04306.
- Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2), 303-338.

- Falk, T., Mai, D., Bensch, R., Çiçek, Ö., Abdulkadir, A., Marrakchi, Y., ... & Dahl, A. (2019). U-net: deep learning for cell counting, detection, and morphometry. *Nature methods*, 16(1), 67-70.
- Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., & Garcia-Rodriguez, J. (2017). A review on deep learning techniques applied to semantic segmentation. arXiv preprint arXiv:1704.06857.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 2672–2680.
- Gu, Z., Cheng, J., Fu, H., Zhou, K., Hao, H., Zhao, Y., ... & Zhang, L. (2019). CE-Net: context encoder network for 2D medical image segmentation. *IEEE transactions on Medical Imaging*, 38(10), 2281-2292.
- Hammond, D. K., Gurton, K., & Shrestha, A. (2017). Creation of RIT-18 multispectral scene dataset. In *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XXIII* (Vol. 10198, p. 101980Q). *International Society for Optics and Photonics*.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
- Hoffman, J.; Tzeng, E.; Park, T.; Zhu, J.Y.; Isola, P.; Saenko, K.; Efros, A.A.; Darrell, T. Cycada: Cycle-consistent adversarial domain adaptation. arXiv preprint arXiv:1711.03213 2017.
- Hua Wang, Cuiqin Ma, and Lijuan Zhou. A brief review of machine learning and its application. In 2009 *International conference on information engineering and computer science*, pages 1–4. IEEE, 2009.
- Huang, X., & Belongie, S. J. (2017). Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization. 2017 IEEE International Conference on Computer Vision (ICCV), 1510-1519.
- Huang, Y., Sharan, L., Dziedzic, J., Shapiro, L. G., Elhalawani, H., Ho, D. J., & Xu, D. (2020). A Critical Look at the Normalization of Adversarial Image Synthesis. ArXiv, abs/2004.02285.
- Iglovikov, V. I., Seferbekov, S. S., Buslaev, A. V., & Shvets, A. A. (2018). TerausNetV2: Fully convolutional network for instance segmentation. *In Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 233-237).

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *In International conference on machine learning* (pp. 448-456). PMLR.

Irem Ulku, Erdem Akagunduz, Pedram Ghamisi. “Deep Semantic Segmentation of Trees Using Multispectral Images”, *IEEE Journal of Selected Topics in Applied Earth Observation and Remote Sensing*: DOI 10.1109/JSTARS.2022.3203145

Isensee, F., Jaeger, P. F., Kohl, S. A. A., Petersen, J., & Maier-Hein, K. H. (2021). nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature Methods*, 18, 203-211.

Janocha, K., & Czarnecki, W. M. (2017). On loss functions for deep neural networks in classification. *Schedae Informaticae*, 25.

Kao, P. Y., Ngo, T., Zhang, A., Chen, J. W., & Manjunath, B. (2018). Brain tumor segmentation and tractographic feature extraction from structural MR images for overall survival prediction. *In International MICCAI Brainlesion Workshop* (pp. 128-141). Springer.

Kamilaris, A., & Prenafeta-Boldó, F. X. (2018). Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, 147, 70-90.

Kemker, R., Salvaggio, C., & Kanan, C. (2018). Algorithms for semantic segmentation of multispectral remote sensing imagery using deep learning. *ISPRS journal of photogrammetry and remote sensing*, 145, 60-77.

Li, Y., Wang, N., Shi, J., Liu, J., & Hou, X. (2016). Revisiting Batch Normalization for Practical Domain Adaptation. *ArXiv*, abs/1603.04779.

Liu, M.Y.; Breuel, T.; Kautz, J. Unsupervised Image-to-Image Translation Networks. *NIPS*, 2017.

Maiti, T., & Chakraborty, C. (2019). Performance analysis of different segmentation evaluation metrics in brain MRI images. *In 2019 IEEE 16th India Council International Conference (INDICON)* (pp. 1-6). IEEE.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. Cambridge, MA: *MIT press*, 2018.

- Myronenko, A. (2018). 3D MRI brain tumor segmentation using autoencoder regularization. In *International MICCAI Brainlesion Workshop* (pp. 311-320). Springer
- Park, Moon S., et al. "Abstract WP530: A Machine Learning Approach in Prediction of Recurrent Stroke." *Stroke*, 2019, https://doi.org/10.1161/str.50.suppl_1.wp530.
- Perez, E., Strub, F., De Vries, H., Courville, A., & Pietquin, O. (2018). Modulating early visual processing by language. ArXiv, abs/1707.00683.
- Park, T., Liu, M., Tero-Bryans, T., & Kwon, Y. (2019). SPADE: Semantic Image Synthesis with SPADE. ArXiv, abs/1711.11585.
- Puthenputhussery, Ajit, et al. "Learning and Recognition Methods for Image Search and Video Retrieval." *Intelligent Systems Reference Library*, 2017, https://doi.org/10.1007/978-3-319-52081-0_2.
- Rebuffi, S., Bilen, H., & Vedaldi, A. (2018). Efficient Parametric Families of Modulating Convolutions. ArXiv, abs/1804.05168.
- Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234-241). Springer, Cham.
- Roy, A. G., Conjeti, S., Navab, N., & Wachinger, C. (2019). Bayesian QuickNAT: Model uncertainty in deep whole-brain segmentation for structure-wise quality control. *NeuroImage*, 195, 11-22.
- Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *towards data science*, 6(12):310–316, 2017.
- Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2018). How does batch normalization help optimization?. In *Advances in Neural Information Processing Systems* (pp. 1805-11604).
- Sun, K., Xiao, B., Liu, D., & Wang, J. (2019). Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5693-5703).

- Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022.
- Wang, Y., Zhang, Y., Zhang, C., Wang, Q., Wang, Z., & Tang, J. (2018). Adversarial domain adaptation for semantic segmentation of urban scenes. *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 0-0.
- Wu, Y., & He, K. (2018). Group normalization. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 3-19).
- Xiao, J., Yu, L., Xing, L., Yuille, A. L., & Zhou, Y. (2021). DualNorm-UNet: Incorporating Global and Local Statistics for Robust Medical Image Segmentation. ArXiv: Image and Video Processing. link
- Xiao, J., Yu, L., Zhou, Z., Bai, Y., Xing, L., Yuille, A., & Zhou, Y. (2022). CateNorm: Categorical Normalization for Robust Medical Image Segmentation. <https://arxiv.org/abs/2103.15858>
- Yi, Z.; Zhang, H.; Tan, P.; Gong, M. DualGAN: Unsupervised Dual Learning for Image-to-Image Translation. *2017 IEEE International Conference on Computer Vision (ICCV) 2017*, pp. 2868–2876.
- Yuyin Zhou, Lingxi Xie, Wei Shen, Yan wang, Elliot K fishman, and Alan L Yuille. A fixed-point model for pancreas segmentation in abdominal ct scans. In *International conference on medoca; image computing and computer-assisted intervention*, pages 693-701. Springer, 2017, 2,6
- Zhang, Y.; David, P.; Gong, B. Curriculum domain adaptation for semantic segmentation of urban scenes. *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2020–2030.
- Zhang, Y., Li, Z., & Zhang, L. (2019). Deep domain adaptation for semantic segmentation of multispectral images. *Journal of Sensors*, 2019.
- Zhang, Y.; Zhang, R.; Pathak, D.; Darrell, T.; Efros, A.A.; Wang, O.; Shechtman, E. Toward Multimodal Image-to-Image Translation. NIPS, 2017.
- Zhang, Y.; Zhang, Y.; Li, T.; Zhang, D. Cross-Domain Adaptation: A New Solution to the Domain Gap in Urban Scene Segmentation. *IEEE Transactions on Image Processing*, 27(6), 2944-2956.

- Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2019). Pyramid Scene Parsing Network. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 6230-6239.
- Zhong, Z., Li, J., Luo, Z., & Chapman, M. (2019). Spectral–spatial residual network for hyperspectral image classification: A 3-d deep learning framework.
- Zhong, Z., Zhang, T., Zheng, S., & Liu, Y. (2020). Adaptive Consistency Regularization for Semi-Supervised Cross-Domain Semantic Segmentation. *In European Conference on Computer Vision (ECCV)* (pp. 484-500).
- Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *In Proceedings of the IEEE International Conference on Computer Vision* (pp. 2223-2232).
- Zhu, J. Y., Zhang, R., Pathak, D., Darrell, T., Efros, A. A., Wang, O., & Shechtman, E. (2017). Toward Multimodal Image-to-Image Translation. *In Advances in Neural Information Processing Systems* (pp. 465-476).
- Zhu, J. Y.; Zhang, R.; Pathak, D.; Darrell, T.; Efros, A.A.; Wang, O.; Shechtman, E. Toward Multimodal Image-to-Image Translation with Conditional Adversarial Networks. 2017 IEEE Computer Vision and Pattern Recognition (CVPR) NIPS, 2017. doi:10.1109/cvpr.2017.632.

EKLER

EK 1 Kodlar

Veri Kümesi Hazırlama Kodu

```
def get_images(trainSetSize, fno, fsiz, tsind, trind, vlind, chindex):  
    """  
    Load and preprocess images and corresponding target masks for training a neural  
    network.  
  
    Parameters:  
        trainSetSize (int): Quantify of samples in the training set.  
        fno (int): File NO.  
        fsiz (int): File size.  
        tsind (list): List of indices for the test set.  
        trind (list): List of indices for the training set.  
        vlind (list): List of indices for the validation set.  
        chindex (str): Channel index, e.g., 'RGBs'.  
  
    Returns:  
        torch.Tensor: Processed input images.  
        torch.Tensor: Processed target masks.  
        float: Mean value of red channel in the training set images.  
        float: Mean value of green channel in the training set images.  
        float: Mean value of blue channel in the training set images.  
    """  
    input_images = []  
    target_masks = []  
    gettingfiles = []  
  
    if chindex == 'RGBs':  
        names = os.listdir('RGBs')  
        for b in names[0:trainSetSize]:  
            gettingfiles.append(b)  
            a = sio.loadmat('RGBs/{}'.format(b))  
            a = a['inputPatch']  
            input_images.append(a)  
            c = sio.loadmat('labels/{}'.format(b))  
            c = c['inputPatch']  
            target_masks.append(c)  
  
    indFile = open("tsind.txt", "w")  
    d=datetime.datetime.now()  
    testnames = open("testnames_{}_{}_{}_{}_{}.txt".format(d.year, d.month, d.day,  
d.hour, d.minute), "w")  
    for say in range(0, len(tsind)):  
        indFile.write(str(tsind[say])+'\n')  
        testnames.write(str(gettingfiles[tsind[say]])+'\n')  
    indFile.close()  
    testnames.close()  
  
    indFile = open("trind.txt", "w")  
    for say in range(0, len(trind)):  
        indFile.write(str(trind[say])+'\n')  
    indFile.close()  
  
    indFile = open("vlind.txt", "w")  
    for say in range(0, len(vlind)):  
        indFile.write(str(vlind[say])+'\n')  
    indFile.close()  
  
    input_images = np.asarray(input_images, dtype=np.float32)  
    target_masks = np.asarray(target_masks, dtype=np.float32)  
  
    print(f"Original image size: {input_images.shape} Original target mask size:  
{target_masks.shape}")  
    print("=====\n")  
  
    lim=224
```

```

input_images = np.reshape(input_images[0:trainSetSize*lim*lim], (trainSetSize, lim,
lim, 3))
input_images = np.moveaxis(input_images,3,1)

target_masks = np.reshape(target_masks[0:trainSetSize*lim*lim], (trainSetSize, 1,
lim, lim))

print(f"Reshaped image size: {input_images.shape} Reshaped mask size:
{target_masks.shape}")
print("=====\n\n")

trMeanR = input_images[trind,0,:,:].mean()
trMeanG = input_images[trind,1,:,:].mean()
trMeanB = input_images[trind,2,:,:].mean()

input_images[:,0,:,:] = input_images[:,0,:,:] - trMeanR
input_images[:,1,:,:] = input_images[:,1,:,:] - trMeanG
input_images[:,2,:,:] = input_images[:,2,:,:] - trMeanB

input_images = torch.from_numpy(input_images)
target_masks = torch.from_numpy(target_masks)

print("image size",input_images.shape,"mask size",target_masks.shape)
print("type image",type(input_images),"type mask",type(target_masks))

return input_images, target_masks, trMeanR, trMeanG, trMeanB

```

#####

Visualization code

```

def segplot(pathm,lim,fimage, foutput,fmask,trMeanR, trMeanG, trMeanB):
    """ Visualize segmented images and masks using color mapping and save them as image
    files.
    Parameters:
        pathm (str): Path to save the output images.
        lim (int): Image size (width/height) in pixels.
        fimage (numpy.ndarray): Original image.
        foutput (numpy.ndarray): Predicted mask.
        fmask (numpy.ndarray): Ground truth mask.
        trMeanR (float): Mean value of red channel in the training set images.
        trMeanG (float): Mean value of green channel in the training set images.
        trMeanB (float): Mean value of blue channel in the training set images.
    """
    print("=====\n\n")
    print('image shape', fimage.shape, 'output shape', foutput.shape, 'groundtruth
    shape', fmask.shape)

    fimage[:, :, 0] = fimage[:, :, 0] + trMeanR
    fimage[:, :, 1] = fimage[:, :, 1] + trMeanG
    fimage[:, :, 2] = fimage[:, :, 2] + trMeanB
    fimage = (fimage - np.min(fimage)) / (np.max(fimage) - np.min(fimage))

    v = fimage[:, :, 0] / 4 + np.squeeze(foutput)/2 + np.squeeze(fmask)/4
    s = np.minimum(np.squeeze(fmask + foutput), np.ones((lim, lim), np.float32))

    h = 0.75 - np.squeeze(fmask) / 2
    print(s[55, 76])

    h = h * 179
    h = h.astype(np.uint8)

    v = v * 255
    v = v.astype(np.uint8)

    s = s * 255
    s = s.astype(np.uint8)

    hsv_image = cv2.merge([h, s, v])
    out = cv2.cvtColor(hsv_image, cv2.COLOR_HSV2BGR)
    out = cv2.cvtColor(out, cv2.COLOR_BGR2RGB)
    outname = 'segmentation_image.png'
    cv2.imwrite(os.path.join(pathm, outname), out)

```

```

plt.imsave(os.path.join(pathm, 'test_image.png'), fimage)
plt.imsave(os.path.join(pathm, 'test_image_R.png'), fimage[:, :, 0], cmap="gray")
plt.imsave(os.path.join(pathm, 'test_image_G.png'), fimage[:, :, 1], cmap="gray")
plt.imsave(os.path.join(pathm, 'test_image_B.png'), fimage[:, :, 2], cmap="gray")
plt.imsave(os.path.join(pathm, 'test_pred_mask.png'), np.squeeze(foutput))
plt.imsave(os.path.join(pathm, 'ground_truth_mask.png'), np.squeeze(fmask))

```

CROSS VALIDATION CODE

```

def CrossVal(N, fno, fsiz):
    """
    Generate train, test, and validation indices for cross-validation.

    Parameters:
        N (int): Total number of samples.
        fno (int): Fold number (starting from 1).
        fsiz (int): Number of folds.

    Returns:
        tsind (numpy.ndarray): Indices of the test set.
        trind (numpy.ndarray): Indices of the training set.
        vlind (numpy.ndarray): Indices of the validation set.
    """
    fno = fno - 1
    ind = []
    with open("randInd{}.txt".format(N)) as f:
        lines = f.readlines()
        for l in lines:
            ind.append(int(l))
    ind = np.asarray(ind)

    tsind = 0
    tstsize = int(N / fsiz)
    if (fno + 1) * tstsize > N:
        tsind1 = range((fno * tstsize) % N, N)
        tsind2 = range(0, ((fno + 1) * tstsize) % N)
        tsind = np.concatenate((tsind1, tsind2))
    else:
        tsind = range(fno * tstsize, (fno + 1) * tstsize)

    trvlind = np.setdiff1d(ind, tsind)

    valRatio = 0.1
    valsize = int((N - tstsize) * valRatio)

    vlind = trvlind[0:valsize]
    trind = trvlind[valsize:]

    trind = ind[trind]
    tsind = ind[tsind]
    vlind = ind[vlind]

    return tsind, trind, vlind

```

#####

JACCARD FUNCTION

```

def Jaccard(y, y_pred, epsilon=1e-8):
    """
    Calculate Jaccard similarity coefficient for binary segmentation masks.

    Parameters:
        y (numpy.ndarray): Ground truth binary mask (0 or 1).
        y_pred (numpy.ndarray): Predicted binary mask (0 or 1).
        epsilon (float, optional): Smoothing term to avoid division by zero. Defaults to
        1e-8.

    Returns:
        jack (numpy.ndarray): Jaccard similarity coefficient.
    """
    TP = (y_pred * y).sum(0) # True positives

```

```

FP = ((1 - y_pred) * y).sum(0) # False positives
FN = ((1 - y) * y_pred).sum(0) # False negatives
jack = (TP + epsilon) / (TP + FP + FN + epsilon) # Jaccard similarity coefficient
return jack

```

In the function:

y represents the ground truth binary mask.
y_pred represents the predicted binary mask.
epsilon is a minute constant added to the denominator to avoid division by zero.
 The function calculates the number of true positives (**TP**), false positives (**FP**), and false negatives (**FN**) using element-wise operations and summation."
 The Jaccard similarity coefficient is then computed using the formula

$$\frac{(TP + \epsilon)}{(TP + FP + FN + \epsilon)}$$

 The resulting Jaccard values are returned as a numpy array.

```
#####
```

```
// Install and Import to print model summary
```

```

"
import torch
import torch.nn as nn

try:
    import torchinfo
except:
    !pip install torchinfo
    import torchinfo
from torchinfo import summary "
```

```
"""
```

```

This is the dual normalization, regularization and residual connection modified
DUALNORM_UNET code with:

```

```
# Normalization Layers
```

```

- Batch Normalization
- Group Normalization

```

```
Added via parameters:
```

```

self.norm 0 = norm layer1(out channels)
self.norm_1 = norm_layer2(out_channels)

```

```
# Regularization
```

```

- Dropout layer added for every upsampling block(up paths)
self.dropout = nn.Dropout2d(p=0.1)

```

```
Adds "residual" parameter to enable residual/skip connections
```

```
self.residual = residual
```

```
if self.residual:
```

```
    x += residual
```

```
Residual connections are added in:
```

```

- ResBlock layers via new "residual" parameter
- Down layers via new "residual" parameter

```

```

- Up layers via initializing self.residual = True

The modifications improve:
- Robustness to small batches
- Generalization capability
- Reliability for medical image segmentation
Keeping sigmoid output as per binary segmentation requirement.
"""
- For Conv2d, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)
- For BatchNorm2d, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
- For GroupNorm(16, 32, eps=1e-05, affine=True)
- For MaxPool2d, kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False
- For ConvTranspose2d, kernel_size=(3, 3), stride=(2, 2) padding=(1, 1),
"""
output_padding=(1, 1)
Relu layers is added into the OutConv block and ResBlock
4 downsampling blocks (down paths)
1 Middle block (bottleneck)
4 upsampling blocks (up paths)
Sigmoid output activation or as the final layer

#####

import torch
import torch.nn as nn
import torch.nn.functional as F

class ResBlock(nn.Module):
    def __init__(self, in_channels, out_channels, n_classes, nonlinear='relu',
norm_layer1=nn.BatchNorm2d, norm_layer2=nn.GroupNorm):
        super(ResBlock, self).__init__()
        self.norm_layer1 = norm_layer1
        self.norm_layer2 = norm_layer2

        self.nonlinear_layer = nn.ReLU(inplace=True)
        self.norm_0 = norm_layer1(out_channels)
        self.norm_1 = norm_layer2(num_channels=out_channels, num_groups=16)
        self.conv_0 = nn.Conv2d(in_channels, out_channels, kernel_size=3, stride=1,
padding=1)
        self.conv_1 = nn.Conv2d(out_channels, out_channels, kernel_size=3, stride=1,
padding=1)

    def forward(self, x):
        residual = x
        x = self.conv_0(x)
        x = self.nonlinear_layer(x)
        x = self.norm_0(x)
        x = self.conv_1(x)
        x = self.nonlinear_layer(x)
        x = self.norm_1(x)
        x += residual # Add residual connection
        return x

class InConv(nn.Module):
    def __init__(self, in_channels, out_channels, n_classes, block=ResBlock, batchsize=4,
nonlinear='relu', norm_layer1=nn.BatchNorm2d, norm_layer2=nn.GroupNorm):
        super(InConv, self).__init__()
        # Add norm layers as params
        self.norm_layer1 = norm_layer1
        self.norm_layer2 = norm_layer2

```

```

        self.nonlinear_layer = nn.ReLU(inplace=True)
        self.conv = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=1, stride=1),
            block(out_channels, out_channels, n_classes, norm_layer1=norm_layer1,
norm_layer2=norm_layer2),
        )

    def forward(self, x):
        x = self.conv(x)
        return x

class Down(nn.Module):
    def __init__(self, in_channels, out_channels, n_classes, block=ResBlock, batchsize=4,
nonlinear='relu', norm_layer1=nn.BatchNorm2d, norm_layer2=nn.GroupNorm, residual=False):
        super(Down, self).__init__()
        # Add norm layers as params
        self.norm_layer1 = norm_layer1
        self.norm_layer2 = norm_layer2

        self.nonlinear_layer = nn.ReLU(inplace=True)
        self.maxpool_conv = nn.Sequential(
            nn.MaxPool2d(2),
            nn.Conv2d(in_channels, out_channels, kernel_size=2, padding=1)
        )
        self.block = block(out_channels, out_channels, n_classes,
norm_layer1=norm_layer1, norm_layer2=norm_layer2)
        self.residual = residual # New parameter for residual connection

    def forward(self, x):
        x = self.maxpool_conv(x)
        if self.residual:
            residual = x
        x = self.block(x)
        if self.residual:
            x += residual # Apply residual connection
        return x

class Mid(nn.Module):
    def __init__(self, in_channels, out_channels, n_classes, batchsize=4,
nonlinear='relu', norm_layer1=nn.BatchNorm2d, norm_layer2=nn.GroupNorm):
        super(Mid, self).__init__()
        # Add norm layers as params
        self.norm_layer1 = norm_layer1
        self.norm_layer2 = norm_layer2

        self.block = ResBlock(in_channels, out_channels, n_classes,
norm_layer1=norm_layer1, norm_layer2=norm_layer2)

    def forward(self, x):
        x = self.block(x)
        return x

class TransConv(nn.Module):
    def __init__(self, in_channels, batchsize=4, nonlinear='relu',
norm_layer1=nn.BatchNorm2d, norm_layer2=nn.GroupNorm):
        super().__init__()

        self.norm_layer1 = norm_layer1
        self.norm_layer2 = norm_layer2

        if self.norm_layer1 == norm_layer1:
            self.up = nn.Sequential(
                norm_layer1(in_channels),
                nn.ReLU(inplace=True),
                nn.ConvTranspose2d(in_channels, in_channels // 2, kernel_size=3, stride=2,
padding=1, output_padding=1)
            )
        elif self.norm_layer2 == norm_layer2:
            self.up = nn.Sequential(
                norm_layer2(num_channels=in_channels),
                nn.ReLU(inplace=True),
                nn.ConvTranspose2d(in_channels, in_channels // 2, kernel_size=3,
stride=2, padding=1, output_padding=1)
            )

```

```

def forward(self, x):
    x = self.up(x)
    return x

class Up(nn.Module):
    def __init__(self, in_channels, out_channels, n_classes=2, bilinear=True,
block=ResBlock, batchsize=4,
        nonlinear='relu', norm_layer1=nn.BatchNorm2d, norm_layer2=nn.GroupNorm,
residual=True):
        super().__init__()
        # Add norm layers as params
        self.norm_layer1 = norm_layer1
        self.norm_layer2 = norm_layer2

        self.nonlinear_layer = nn.ReLU(inplace=True)
        self.up = TransConv(in_channels, batchsize=batchsize, nonlinear=nonlinear,
norm_layer1=norm_layer1, norm_layer2=norm_layer2)
        self.conv = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1),
            block(out_channels, out_channels, n_classes=n_classes, nonlinear=nonlinear,
norm_layer1=norm_layer1, norm_layer2=norm_layer2)
        )

    def forward(self, x1, x2):
        x1 = self.up(x1)
        diffY = x2.size()[2] - x1.size()[2]
        diffX = x2.size()[3] - x1.size()[3]

        x1 = F.pad(x1, [diffX // 2, diffX - diffX // 2,
            diffY // 2, diffY - diffY // 2])

        x = torch.cat([x2, x1], dim=1)
        x = self.conv(x)
        return x

class OutConv(nn.Module):
    def __init__(self, in_channels, out_channels, batchsize=4, nonlinear='relu',
norm_layer1=nn.BatchNorm2d, norm_layer2=nn.GroupNorm):
        super(OutConv, self).__init__()
        self.nonlinear_layer = nn.ReLU(inplace=True)
        self.conv = nn.Sequential(
            norm_layer1(in_channels),
            norm_layer2(num_channels=in_channels, num_groups=16),
            nn.ReLU(inplace=True),
            nn.Conv2d(in_channels, out_channels, kernel_size=1)
        )

    def forward(self, x):
        x = self.conv(x)
        return x

```

```

class DualNorm_UnetV3(nn.Module):
    """
    DualNorm_UnetV3: A U-Net variant with dual normalization and dropout for semantic
    segmentation.

    Args:
        n_channels (int): Number of input channels.
        n_classes (int): Number of output classes.
        nonlinear (str, optional): Nonlinear activation function to use. Default is
        'relu'.
        norm_layer1 (torch.nn.Module, optional): First normalization layer. Default is
        nn.BatchNorm2d.
        norm_layer2 (torch.nn.Module, optional): Second normalization layer. Default is
        nn.GroupNorm. "

    Attributes:
        inc (InConv): Initial convolutional block.
        down1-4 (Down): Down-sampling blocks.
        mid (Mid): Middle block.
        up1-4 (Up): Up-sampling blocks.
        dropout (nn.Dropout2d): Dropout layer for regularization.
        outc (OutConv): Output convolutional block.
        outsig (nn.Sigmoid): Sigmoid activation for the final output.

    Methods:
        forward(x): Forward pass through the network.

    Example:
        model = DualNorm_UnetV3(n_channels=3, n_classes=1)
    """
    def __init__(self, n_channels, n_classes, nonlinear='relu',
                  norm_layer1=nn.BatchNorm2d, norm_layer2=nn.GroupNorm):
        super(DualNorm_UnetV3, self).__init__()
        self.n_channels = n_channels
        self.n_classes = n_classes

        # Use dual normalization
        self.inc = InConv(n_channels, 32, n_classes=n_classes, norm_layer1=norm_layer1,
                           norm_layer2=norm_layer2)
        self.down1 = Down(32, 64, n_classes=n_classes, norm_layer1=norm_layer1,
                           norm_layer2=norm_layer2)
        self.down2 = Down(64, 128, n_classes=n_classes, norm_layer1=norm_layer1,
                           norm_layer2=norm_layer2)
        self.down3 = Down(128, 256, n_classes=n_classes, norm_layer1=norm_layer1,
                           norm_layer2=norm_layer2)
        self.down4 = Down(256, 512, n_classes=n_classes, norm_layer1=norm_layer1,
                           norm_layer2=norm_layer2)

        # Middle block or bottleneck
        self.mid = Mid(512, 512, n_classes=n_classes, norm_layer1=norm_layer1,
                       norm_layer2=norm_layer2)

```

```

        self.up1 = Up(512, 256, n_classes=n_classes, norm_layer1=norm_layer1,
norm_layer2=norm_layer2)
        self.up2 = Up(256, 128, n_classes=n_classes, norm_layer1=norm_layer1,
norm_layer2=norm_layer2)
        self.up3 = Up(128, 64, n_classes=n_classes, norm_layer1=norm_layer1,
norm_layer2=norm_layer2)
        self.up4 = Up(64, 32, n_classes=n_classes, norm_layer1=norm_layer1,
norm_layer2=norm_layer2)

        # Use dropout for regularization
        self.dropout = nn.Dropout2d(p=0.1)

        # Output layer
        self.outc = OutConv(32, n_classes, nonlinear=nonlinear)
        self.outsig = nn.Sigmoid()

def forward(self, x):
    x1 = self.inc(x)
    x2 = self.down1(x1)
    x3 = self.down2(x2)
    x4 = self.down3(x3)
    x5_1 = self.down4(x4)
    x = self.mid(x5_1)
    x = self.up1(x, x4)
    x = self.dropout(x) # Dropout
    x = self.up2(x, x3)
    x = self.dropout(x)
    x = self.up3(x, x2)
    x = self.dropout(x)
    x = self.up4(x, x1)
    x = self.dropout(x)
    x = self.outc(x)
    x = self.outsig(x)
    return x

# Create an instance of the DualNorm_UnetV3
model_1 = DualNorm_UnetV3(n_channels=3, n_classes=1, ) # Create an instance of the
DualNorm_UnetV3, dropout rate=0.1
# Display model summary
summary(model_1,
        input_size=[(16, 3, 224, 224)],
        col_names=["input size", "output size", "num params", "trainable"],
        col_width=20,
        row_settings=["variable_names"]
        )

```

EK 2 DualNorm_UNetV3 Eğitimi ve Parametre Özeti

DualNorm_UNetV3'in yazdırılabilir bir özetini göster

Layer (type (variable_name))	Input Shape	Output Shape	Param #	Trainable
DualNorm_UNetV3 (DualNorm_UNetV3)	[16, 3, 224, 224]	[16, 1, 224, 224]	--	True
└InConv (inc)	[16, 3, 224, 224]	[16, 32, 224, 224]	--	True
└Sequential (conv)	[16, 3, 224, 224]	[16, 32, 224, 224]	--	True
└Conv2d (0)	[16, 3, 224, 224]	[16, 32, 224, 224]	128	True
└ResBlock (1)	[16, 32, 224, 224]	[16, 32, 224, 224]	18,624	True
└Down (down1)	[16, 32, 224, 224]	[16, 64, 113, 113]	--	True
└Sequential (maxpool_conv)	[16, 32, 224, 224]	[16, 64, 113, 113]	--	True
└MaxPool2d (0)	[16, 32, 224, 224]	[16, 32, 112, 112]	--	--
└Conv2d (1)	[16, 32, 112, 112]	[16, 64, 113, 113]	8,256	True
└ResBlock (block)	[16, 64, 113, 113]	[16, 64, 113, 113]	--	True
└Conv2d (conv_0)	[16, 64, 113, 113]	[16, 64, 113, 113]	36,928	True
└ReLU (nonlinear_layer)	[16, 64, 113, 113]	[16, 64, 113, 113]	--	--
└BatchNorm2d (norm_0)	[16, 64, 113, 113]	[16, 64, 113, 113]	128	True
└Conv2d (conv_1)	[16, 64, 113, 113]	[16, 64, 113, 113]	36,928	True
└ReLU (nonlinear_layer)	[16, 64, 113, 113]	[16, 64, 113, 113]	--	--
└GroupNorm (norm_1)	[16, 64, 113, 113]	[16, 64, 113, 113]	128	True
└Down (down2)	[16, 64, 113, 113]	[16, 128, 57, 57]	--	True
└Sequential (maxpool_conv)	[16, 64, 113, 113]	[16, 128, 57, 57]	--	True
└MaxPool2d (0)	[16, 64, 113, 113]	[16, 64, 56, 56]	--	--
└Conv2d (1)	[16, 64, 56, 56]	[16, 128, 57, 57]	32,896	True
└ResBlock (block)	[16, 128, 57, 57]	[16, 128, 57, 57]	--	True
└Conv2d (conv_0)	[16, 128, 57, 57]	[16, 128, 57, 57]	147,584	True
└ReLU (nonlinear_layer)	[16, 128, 57, 57]	[16, 128, 57, 57]	--	--
└BatchNorm2d (norm_0)	[16, 128, 57, 57]	[16, 128, 57, 57]	256	True
└Conv2d (conv_1)	[16, 128, 57, 57]	[16, 128, 57, 57]	147,584	True
└ReLU (nonlinear_layer)	[16, 128, 57, 57]	[16, 128, 57, 57]	--	--
└GroupNorm (norm_1)	[16, 128, 57, 57]	[16, 128, 57, 57]	256	True
└Down (down3)	[16, 128, 57, 57]	[16, 256, 29, 29]	--	True
└Sequential (maxpool_conv)	[16, 128, 57, 57]	[16, 256, 29, 29]	--	True
└MaxPool2d (0)	[16, 128, 57, 57]	[16, 128, 28, 28]	--	--

	└─Conv2d (1)	[16, 128, 28, 28]	[16, 256, 29, 29]	131,328	True
	└─ResBlock (block)	[16, 256, 29, 29]	[16, 256, 29, 29]	--	True
	└─└─Conv2d (conv_0)	[16, 256, 29, 29]	[16, 256, 29, 29]	590,080	True
	└─└─ReLU (nonlinear_layer)	[16, 256, 29, 29]	[16, 256, 29, 29]	--	--
	└─└─BatchNorm2d (norm_0)	[16, 256, 29, 29]	[16, 256, 29, 29]	512	True
	└─└─Conv2d (conv_1)	[16, 256, 29, 29]	[16, 256, 29, 29]	590,080	True
	└─└─ReLU (nonlinear_layer)	[16, 256, 29, 29]	[16, 256, 29, 29]	--	--
	└─└─GroupNorm (norm_1)	[16, 256, 29, 29]	[16, 256, 29, 29]	512	True
─Down (down4)		[16, 256, 29, 29]	[16, 512, 15, 15]	--	True
	└─Sequential (maxpool_conv)	[16, 256, 29, 29]	[16, 512, 15, 15]	--	True
	└─└─MaxPool2d (0)	[16, 256, 29, 29]	[16, 256, 14, 14]	--	--
	└─└─Conv2d (1)	[16, 256, 14, 14]	[16, 512, 15, 15]	524,800	True
	└─ResBlock (block)	[16, 512, 15, 15]	[16, 512, 15, 15]	--	True
	└─└─Conv2d (conv_0)	[16, 512, 15, 15]	[16, 512, 15, 15]	2,359,808	True
	└─└─ReLU (nonlinear_layer)	[16, 512, 15, 15]	[16, 512, 15, 15]	--	--
	└─└─BatchNorm2d (norm_0)	[16, 512, 15, 15]	[16, 512, 15, 15]	1,024	True
	└─└─Conv2d (conv_1)	[16, 512, 15, 15]	[16, 512, 15, 15]	2,359,808	True
	└─└─ReLU (nonlinear_layer)	[16, 512, 15, 15]	[16, 512, 15, 15]	--	--
	└─└─GroupNorm (norm_1)	[16, 512, 15, 15]	[16, 512, 15, 15]	1,024	True
─Mid (mid)		[16, 512, 15, 15]	[16, 512, 15, 15]	--	True
	└─ResBlock (block)	[16, 512, 15, 15]	[16, 512, 15, 15]	--	True
	└─└─Conv2d (conv_0)	[16, 512, 15, 15]	[16, 512, 15, 15]	2,359,808	True
	└─└─ReLU (nonlinear_layer)	[16, 512, 15, 15]	[16, 512, 15, 15]	--	--
	└─└─BatchNorm2d (norm_0)	[16, 512, 15, 15]	[16, 512, 15, 15]	1,024	True
	└─└─Conv2d (conv_1)	[16, 512, 15, 15]	[16, 512, 15, 15]	2,359,808	True
	└─└─ReLU (nonlinear_layer)	[16, 512, 15, 15]	[16, 512, 15, 15]	--	--
	└─└─GroupNorm (norm_1)	[16, 512, 15, 15]	[16, 512, 15, 15]	1,024	True
─Up (up1)		[16, 512, 15, 15]	[16, 256, 29, 29]	--	True
	└─TransConv (up)	[16, 512, 15, 15]	[16, 256, 30, 30]	--	True
	└─└─Sequential (up)	[16, 512, 15, 15]	[16, 256, 30, 30]	1,180,928	True
	└─Sequential (conv)	[16, 512, 29, 29]	[16, 256, 29, 29]	--	True
	└─└─Conv2d (0)	[16, 512, 29, 29]	[16, 256, 29, 29]	1,179,904	True
	└─└─ResBlock (1)	[16, 256, 29, 29]	[16, 256, 29, 29]	1,181,184	True
─Dropout2d (dropout)		[16, 256, 29, 29]	[16, 256, 29, 29]	--	--
─Up (up2)		[16, 256, 29, 29]	[16, 128, 57, 57]	--	True
	└─TransConv (up)	[16, 256, 29, 29]	[16, 128, 58, 58]	--	True
	└─└─Sequential (up)	[16, 256, 29, 29]	[16, 128, 58, 58]	295,552	True
	└─Sequential (conv)	[16, 256, 57, 57]	[16, 128, 57, 57]	--	True

```
Total params: 16,416,417
Trainable params: 16,416,417
Non-trainable params: 0
Total mult-adds (G): 287.91
```

```
Input size (MB): 9.63
Forward/backward pass size (MB): 5059.73
Params size (MB): 65.67
Estimated Total Size (MB): 5135.03
```

```

import numpy as np
from F5 JACCARD import Jaccard
from timeit import default timer as timer

dev = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
device = torch.device(dev)

# Start the timer
start time = timer()

def train_model(n_epochs, trainloss, validationloss, accuracy, model, scheduler, lrFile,
training_generator, optim, lim, trainFile, trainaccFile, trainepochFile,
validation_generator, valFile, valaccFile, pathm, i, modeltype):
    training_losses = []
    for epoch in tqdm(range(n_epochs)):
        model.train()
        batch_losses = []
        jI = 0
        totalBatches = 0
        scheduler.step()
        print('Epoch:', epoch, 'LR:', scheduler.get_last_lr()) # Used .get_last_lr()
    instead of .get_lr()
        lrFile.write('Epoch:'+' '+str(epoch)+' '+'LR:'+'
'+str(scheduler.get last lr())+"\n")
        lrFile.write(str(scheduler.state dict())+"\n")

        mb=0
        for trainim, trainmas in training_generator:
            mb+=1
            optim.zero_grad()
            images=trainim.to(device)
            masks=trainmas.to(device)
            outputs=model(images)

            if trainloss == 'BCEWithLogitsLoss':
                loss=nn.BCEWithLogitsLoss() #loss = nn.BCELoss()
                output = loss(outputs, masks)
                output.backward()
                optim.step()

            batch_losses.append(output.item())
            batchLoad = len(masks)*lim*lim
            totalBatches = totalBatches + batchLoad
            if accuracy == 'Jaccard':
                thisJac =
Jaccard(torch.reshape(masks, (batchLoad,1)), torch.reshape(outputs, (batchLoad,1))) *batchLoa
d

                jI = jI+thisJac.data[0]

        training_loss = np.mean(batch_losses)
        training_losses.append(training_loss)
        trainFile.write(str(training_losses[epoch])+"\n")
        trainaccFile.write(str((jI/totalBatches).item())+"\n")
        trainepochFile.write(str(epoch)+"\n")

    # End the timer and print out how long it took
    end time = timer()
    global total train time
    total train time = end time - start time
    print(f"Total training time: {end time-start time:.2f} seconds")
    print("=====")
    print("Training Jaccard:", (jI/totalBatches).item(), " (epoch:", epoch, ")", "Training
Locess: ", (training_losses[epoch]))
    print("-----\n")
    lrFile.write("Training loss:"+str(training_losses[epoch])+"\n")
    lrFile.write("Training accuracy:"+str((jI/totalBatches).item())+"\n")

    torch.save(model.state_dict(), os.path.join(pathm, "iremmodel{}.pt".format(i)))
    validate(validationloss, accuracy, validation_generator, valFile, valaccFile,
lim, lrFile, pathm, i, modeltype)
    torch.save(model.state dict(), os.path.join(pathm, "Finaliremmodel{}.pt".format(i)))

```

```

def validate(validationloss, accuracy, validation_generator, valFile, valaccFile, lim,
lrFile, pathm, i, modeltype):
    jI = 0
    totalBatches = 0
    validation_losses = []

    if modeltype=='DualNorm_UnetV3':
        model = DualNorm_UnetV3(n_channels=3, n_classes=1, norm_layer1=nn.BatchNorm2d,
norm_layer2=nn.GroupNorm).to(device)

        model.load_state_dict(torch.load(os.path.join(pathm, "iremmodel{}.pt".format(i))))
        model.eval()
        with torch.no_grad():
            val_losses = []
            for valim, valmas in validation_generator:
                #model.eval()
                images=valim.to(device)
                masks=valmas.to(device)
                outputs=model(images)
                if validationloss == 'BCEWithLogitsLoss':
                    loss=nn.BCEWithLogitsLoss() #loss = nn.BCELoss()
                    output = loss(outputs, masks)
                    val_losses.append(output.item())
                batchLoad = len(masks)*lim*lim
                totalBatches = totalBatches + batchLoad
                if accuracy == 'Jaccard':
                    thisJac =
Jaccard(torch.reshape(masks, (batchLoad,1)),torch.reshape(outputs, (batchLoad,1)))*batchLoa
d
                    jI = jI+thisJac.data[0]

            dn=jI/totalBatches
            dni=dn.item()
            validation_loss = np.mean(val_losses)
            validation_losses.append(validation_loss)
            valFile.write(str(validation_loss[0])+"\n")
            valaccFile.write(str(dni)+"\n")
            print("Validation Jaccard:",dni)
            print(f"Validation Jaccard: {dni:.2f}%")
            lrFile.write("Validation loss:"+str(validation_loss[0])+"\n")
            lrFile.write("Validation accuracy:"+str(dni)+"\n")

```

```

import numpy as np
import matplotlib.pyplot as plt
from F5_JACCARD import Jaccard
from F11_SEGPLOTT import segplot

dev = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
device = torch.device(dev)

def test_model(test_generator, lim, testFile, testaccFile, i, modeltype, pathm, trMeanR,
trMeanG, trMeanB):
    if modeltype=='DualNorm_UnetV3':
        net = DualNorm_UnetV3(n_channels=3, n_classes=1,
norm_layer=nn.BatchNorm2d).to(device)
        #net = DualNorm_Unet(n_channels=3, n_classes=1,
norm_layer=nn.InstanceNorm2d).to(device)

        net.load_state_dict(torch.load(os.path.join(pathm, "Finaliremmodel{}.pt".format(i))))

        jI = 0
        totalBatches = 0
        test_losses = []
        net.eval()
        with torch.no_grad():
            t_losses = []
            t=0
            for testin, testmas in test_generator:
                images=testin.to(device)
                masks=testmas.to(device)
                outputs = net(images)
                if t==0:
                    fig=plt.figure()

```

```

axes=[]
fimage=images[0].permute(1, 2, 0)
fimage[:, :, 0]=(images[0][0, :, :])
fimage[:, :, 1]=(images[0][1, :, :])
fimage[:, :, 2]=(images[0][2, :, :])
fimage=fimage.cpu().numpy()
axes.append(fig.add_subplot(1, 2, 1))
foutput=outputs[0].permute(1, 2, 0)
foutput=foutput.cpu().numpy()

print("=====\n\n")
plt.imshow(np.squeeze(foutput, axis=2), cmap='gray')
subplot title=("Test Predicted Mask")
axes[-1].set_title(subplot_title)
axes.append(fig.add_subplot(1, 2, 2))
fmask=masks[0].permute(1, 2, 0)
fmask=fmask.cpu().numpy()
plt.imshow(np.squeeze(fmask, axis=2), cmap='gray')
plt.imshow(fmask, cmap='gray')

subplot title=("Ground Truth Mask")
axes[-1].set_title(subplot_title)
n_curve = 'mask_comparison.png'
plt.savefig(os.path.join(pathm, n_curve))
plt.show()
segplot(pathm, lim, fimage, foutput, fmask, trMeanR, trMeanG, trMeanB)
losst=nn.BCEWithLogitsLoss() #losst = nn.BCELoss()

output = losst(outputs, masks)
t losses.append(output.item())
batchLoad = len(masks)*lim*lim
totalBatches = totalBatches + batchLoad
thisJac =
Jaccard(torch.reshape(masks, (batchLoad,1)), torch.reshape(outputs, (batchLoad,1)))*batchLoa
d

jI = jI+thisJac.data[0]
t +=1

dn=jI/totalBatches
dni=dn.item()
test loss = np.mean(t losses)
test losses.append(test loss)
testFile.write(str(test losses[0])+"\n")
testaccFile.write(str(dni)+"\n")
print("=====TEST JACCARD=====")
print("Test Jaccard:", dni)
print(f"Test Jaccard: {dni:.2f}% | Test losses: {test loss:.2f}% ")
print("-----")

```

```

from future import print function
import os
import torch
import torch.nn as nn
from torch.utils.data import DataLoader
#import numpy as np
import datetime
import matplotlib.pyplot as plt
from torch.optim.lr_scheduler import StepLR
from F3_DATASET import satellitedata
from F6_CROSSVAL import CrossVal
from F8_IMAGES import get_images
import ssl
ssl.create_default_https_context = ssl.create_unverified_context

#####
if name == ' main ':

    if (torch.cuda.is available()):
        print(torch.cuda.get_device_name(0))

    bg=datetime.datetime.now()
    bgh=bg.hour
    bgm=bg.minute

    dev = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
    #dev = torch.device("cpu")
    device = torch.device(dev)

    for i in range(0,1):
        #Replace the path to your experiments folder
        data_folder =
os.path.join("/content/drive/MyDrive/UNET_ARCHITECTURE/experiments")
        file to open = os.path.join(data_folder, "model{}.txt".format(i))
        with open(file to open) as f:
            #lines = f.readlines()
            lines = [line.rstrip() for line in f]
            trainSetSize=int(lines[0])
            fno = int(lines[1])
            fsiz = int(lines[2])
            valRatio = float(lines[3])
            miniBatchSize = int(lines[4])
            n_epochs = int(lines[5])
            learnRate = float(lines[6])
            optimizerType=str(lines[7])
            trainloss=str(lines[8])
            validationloss=str(lines[9])
            accuracy=str(lines[10])
            initialization=str(lines[11])
            step_size=int(lines[12])
            gamma=float(lines[13])
            lim=int (lines[14])
            modeltype=str(lines[15])
            chindex=str(lines[16])
            transfertype=str(lines[17])

            tsind, trind,vlind = CrossVal(trainSetSize,fno,fsiz)
            input images, target masks, trMeanR, trMeanG, trMeanB = get_images(trainSetSize,
fno, fsiz, tsind, trind, vlind, chindex)

            params = {'batch_size': miniBatchSize, 'shuffle': False}

            training set = satellitedata(input images[trind], target masks[trind])
            training generator = DataLoader(training set, **params)

            validation_set = satellitedata(input images[vlind], target_masks[vlind])
            validation_generator = DataLoader(validation_set, **params)

            test set = satellitedata(input images[tsind], target masks[tsind])
            test generator = DataLoader(test set, **params)

```

```

        if modeltype=='DualNorm_UnetV3':
            model = DualNorm_UnetV3(n_channels=3, n_classes=1,norm_layer1=nn.BatchNorm2d,
norm_layer2=nn.GroupNorm
).to(device)
            #model = DualNorm_Unet(n_channels=3, n_classes=1,
norm_layer=nn.InstanceNorm2d).to(device)

#####
def init_weights(m):
    if initialization == 'xavier_uniform':
        if isinstance(m, nn.Conv2d):
            torch.nn.init.xavier_uniform_(m.weight)
            if m.bias is not None:
                torch.nn.init.zeros_(m.bias)

        if initialization == 'xavier_normal':
            if isinstance(m, nn.Conv2d):
                torch.nn.init.xavier_normal_(m.weight)
                if m.bias is not None:
                    torch.nn.init.zeros_(m.bias)

        if initialization == 'kaiming_uniform':
            if isinstance(m, nn.Conv2d):
                torch.nn.init.kaiming_uniform_(m.weight)
                if m.bias is not None:
                    torch.nn.init.zeros_(m.bias)

        if initialization == 'kaiming_normal':
            if isinstance(m, nn.Conv2d):
                torch.nn.init.kaiming_normal_(m.weight)
                if m.bias is not None:
                    torch.nn.init.zeros_(m.bias)
#####
#model.apply(init_weights)

    if transfertype=='yestr':
        model.load_state_dict(torch.load(os.path.join(data_folder,
"2022 8 15 11 11.pt")))
    elif transfertype=='notr':
        model.apply(init_weights)

    if optimizerType == 'Adam':
        optim = torch.optim.Adam(model.parameters(),learnRate)
    elif optimizerType == 'SGD':
        optim = torch.optim.SGD(model.parameters(),learnRate)
    elif optimizerType == 'AdamW':
        optim = torch.optim.AdamW(model.parameters(),learnRate)

    scheduler = StepLR(optim, step_size, gamma)

    d=datetime.datetime.now()
    pathm = os.path.join(data_folder, "{} {} {} {} {} {} model{}".format(d.year,
d.month, d.day, d.hour, d.minute, i))
    os.mkdir(pathm)

    os.path.join(pathm, "lrFile.txt")
    lrFile = open("lrFile.txt","w")
    os.path.join(pathm, "trainaccFile.txt")
    trainaccFile = open("trainaccFile.txt","w")
    os.path.join(pathm, "valaccFile.txt")
    valaccFile = open("valaccFile.txt","w")
    os.path.join(pathm, "trainepochFile.txt")
    trainepochFile = open("trainepochFile.txt","w")
    os.path.join(pathm, "trainFile.txt")
    trainFile = open("trainFile.txt","w")
    os.path.join(pathm, "valFile.txt")
    valFile = open("valFile.txt","w")
    train_model(n_epochs, trainloss, validationloss, accuracy, model, scheduler,
lrFile, training generator, optim, lim, trainFile, trainaccFile, trainepochFile,
validation generator, valFile, valaccFile, pathm, i, modeltype)
    try:
        trainFile.close()
        valFile.close()

```

```

        trainaccFile.close()
        valaccFile.close()
        trainepochFile.close()
        lrFile.close()
    except OSError as e:
        print("Error while closing files:", e)

    os.path.join(pathm, "testaccFile.txt")
    testaccFile = open("testaccFile.txt", "w")
    os.path.join(pathm, "testFile.txt")
    testFile = open("testFile.txt", "w")
    test model(test generator, lim, testFile, testaccFile, i, modeltype, pathm,
trMeanR, trMeanG, trMeanB)
    testFile.close()
    testaccFile.close()

    x=[]
    with open("trainFile.txt") as f:
        lines = f.readlines()
        for l in lines:
            x.append(float(l))

    y=[]
    with open("valFile.txt") as f:
        lines = f.readlines()
        for l in lines:
            y.append(float(l))

    tt=[]
    with open("testFile.txt") as f:
        lines = f.readlines()
        for l in lines:
            tt.append(float(l))

    z=[]
    with open("lrFile.txt") as f:
        lines = f.readlines()
        for l in lines:
            z.append(l)

    xx=[]
    with open("trainaccFile.txt") as f:
        lines = f.readlines()
        for l in lines:
            xx.append(float(l))

    yy=[]
    with open("valaccFile.txt") as f:
        lines = f.readlines()
        for l in lines:
            yy.append(float(l))

    ta=[]
    with open("testaccFile.txt") as f:
        lines = f.readlines()
        for l in lines:
            ta.append(float(l))

    e1=[]
    with open("trainepochFile.txt") as f:
        lines = f.readlines()
        for l in lines:
            e1.append(float(l))

    def logfile():
        global total_train_time
        a=datetime.datetime.now()
        myfile=os.path.join(pathm, "{}_{}_{}_{}_{}.txt".format(a.year, a.month,
a.day, a.hour, a.minute))
        LogFile = open(myfile, "w")
        LogFile.write("Date:"+str(datetime.date.today())+"\n")
        LogFile.write("Ending Time:"+str(a.hour)+":"+str(a.minute)+"\n")
        LogFile.write("Starting Time:"+str(bgh)+":"+str(bgm)+"\n")
        LogFile.write("Total Training Time in seconds: "+str(total_train_time)+"\n")

```

```

LogFile.write("Data set size:"+str(trainSetSize)+"\n")
LogFile.write("Fold number:"+str(fno)+"\n")
LogFile.write("Fold number:"+str(fsiz)+"\n")
LogFile.write("Number of validation images:"+str(len(vlind))+"\n")
LogFile.write("Number of training images:"+str(len(trind))+"\n")
LogFile.write("Mini batch size:"+str(miniBatchSize)+"\n")
LogFile.write("Type of initialization:"+initialization+"\n")
LogFile.write("Test accuracy:"+str(ta)+"\n")
LogFile.write("Learning rate:"+str(learnRate)+"\n")
LogFile.write("Model version:"+str(modeltype)+"\n")
LogFile.write("Optimizer type:"+optimizerType+"\n")
LogFile.write("Total number of epochs:"+str(n_epochs)+"\n")
LogFile.write("Training loss function:"+str(trainloss)+"\n")
LogFile.write("Validation loss function:"+str(validationloss)+"\n")
LogFile.write("Accuracy function:"+str(accuracy)+"\n")
LogFile.write("Channel index:"+str(chindex)+"\n")
LogFile.write("Transfer:"+str(transfertype)+"\n")
LogFile.write("Model Summary:"+str(model)+"\n")
for i in range(len(z)):
    LogFile.write(str(z[i]))
LogFile.close()
logfile()

print("=====LARNING CURVES=====\\n")

plt.plot(x,"k-", label="Train Loss")
plt.plot(y,"r--", label="Validation Loss")
plt.title('Learning Curves')
plt.legend(loc="upper left")
l curve = 'learning curves.png'
plt.savefig(os.path.join(pathm, l curve))
plt.show()
print("\\n\\n")

print("=====ACCURACY CURVES=====\\n")

plt.plot(xx, "k-", label="Train Accuracy")
plt.plot(yy, "r--", label="Validation Accuracy")
plt.title('Accuracy Curves')
plt.legend(loc="upper left")
a curve = 'accuracy curves.png'
plt.savefig(os.path.join(pathm, a curve))
plt.show()

print("\\n\\n")
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print("Device:", device)

print("Memory allocated before model:",
torch.cuda.memory_allocated(device=device))
# del model
# torch.cuda.empty_cache()
print("Memory allocated after model:",
torch.cuda.memory_allocated(device=device))

print("Memory allocated before model:", torch.cuda.memory_allocated())

```