T.C.
KONYA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ABDOMEN BT GÖRÜNTÜLERİNDE PANKREAS SEGMENTASYONU İÇİN YENİ BİR DERİN ÖĞRENME YAKLAŞIMI: PASCAL U-NET

Ender KURNAZ

YÜKSEK LİSANS TEZİ

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Temmuz-2021
KONYA

TEZ KABUL VE ONAYI

Ender KURNAZ tarafından hazırlanan “Abdomen BT Görüntülerinde Pankreas Segmentasyonu İçin Yeni Bir Derin Öğrenme Yaklaşımı: Pascal U-Net” adlı tez çalışması 08/07/2021 tarihinde aşağıdaki jüri tarafından oy birliği ile Konya Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı’nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Prof. Dr. Salih GÜNEŞ	……..
Danışman Doç. Dr. Rahime CEYLAN	……..
Üye Dr. Öğr. Üyesi Güzin ÖZMEN	……..

Yukarıdaki sonucu onaylarım.

Prof. Dr. Saadettin Erhan KESEN
Enstitü Müdürü

TEZ BİLDİRİMİ

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

DECLARATION PAGE

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

Ender KURNAZ

Tarih: 08.07.2021

ÖZET

YÜKSEK LİSANS TEZİ

ABDOMEN BT GÖRÜNTÜLERİNDE PANKREAS SEGMENTASYONU İÇİN YENİ BİR DERİN ÖĞRENME YAKLAŞIMI: PASCAL U-NET

Ender KURNAZ

Konya Teknik Üniversitesi
Lisansüstü Eğitim Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Rahime CEYLAN

2021, 77 Sayfa

Jüri
Doç. Dr. Rahime CEYLAN
Prof. Dr. Salih GÜNEŞ
Dr. Öğr. Üyesi Güzin ÖZMEN

Günümüzde derin öğrenme modellerinin medikal görüntü işlemede kullanımı hız kazanmıştır. Özellikle kesit görüntülerinden organ segmentasyonu üzerine gerçekleştirilen çalışmalarda derin öğrenme yöntemleri sıklıkla tercih edilmektedir. Abdomen bölgesinde yer alan pankreas, her insanda şekil, konum ve büyüklük bakımından farklı olduğundan segmentasyonu oldukça zorlayıcıdır. Bu problemin çözümünde literatürde genellikle derin öğrenme modellerinden biri olan U-Net modeli tercih edilmektedir.

Bu tez çalışmasında, pankreas segmentasyonu için Pascal üçgenindeki sayı dizilimine uygun bir mimariye sahip ve U-Net modelini temel alan yeni bir derin öğrenme modeli önerilmiştir. Önerilen bu model Pascal U-Net modeli olarak isimlendirilmiştir ve modelin başarımı iki farklı veri seti üzerinde değerlendirilmiştir. İlk olarak halka açık ve literatürde sıklıkla kullanılan bir veri seti olan The Cancer Imaging Archive Pankreas-BT veri setinden yararlanılmıştır. Ayrıca ikinci veri seti olarak Selçuk Üniversitesi Tıp Fakültesi Hastanesi Radyoloji Bölümü'nden alınan abdomen BT görüntüleri kullanılmıştır. Veri setlerindeki kayıtlardan her hasta için bir kesit görüntüsü seçilmiş ve önişleme yöntemleri uygulanarak derin öğrenme ağları için veri setleri oluşturulmuştur. Pascal U-Net modeli ile her iki veri seti üzerinde elde edilen pankreas segmentasyon sonuçlarının karşılaştırılması için, aynı veri setleri üzerinde U-Net modeli ile de segmentasyon işlemi gerçekleştirilmiştir. 2, 4 ve 6 katlı çapraz doğrulama ve 1'den 10'a kadar farklı yığın sayılarında çalıştırılan derin öğrenme modelleri sonucunda elde edilen segmentasyon haritaları, 7 farklı performans metriği kullanılarak değerlendirilmiştir. Her bir yığın sayısı ve farklı kat çapraz doğrulama ile gerçekleştirilen pankreas segmentasyonu sonuçları, 10 kez çalıştırma sonuçlarının ortalamasıdır. Hem U-Net hem de Pascal U-Net segmentasyon sonuçları 7 farklı metrik ve görsel değerlendirmeler temel alınarak analiz edilmiştir. Sonuçlar incelendiğinde; her iki veri setinde de Pascal U-Net modeli, geleneksel U-Net mimarisine karşı Dice Benzerlik Katsayısı metriği bakımından yaklaşık %1'lik bir değer ile üstünlük göstermiştir.

Anahtar Kelimeler: Çapraz Doğrulama, Derin Öğrenme, Evrişimli Sinir Ağları, Pankreas Segmentasyonu, Pascal U-Net, U-Net.

ABSTRACT

MS THESIS

A NEW DEEP LEARNING APPROACH FOR PANCREAS SEGMENTATION ON ABDOMEN CT IMAGES: PASCAL U-NET

Ender KURNAZ

Konya Technical University
Institute of Graduate Studies
Department of Electrical and Electronics Engineering

Advisor: Assoc. Prof. Dr. Rahime CEYLAN

2021, 77 Pages

Jury
Assoc. Prof. Dr. Rahime CEYLAN
Prof. Dr. Salih GÜNEŞ
Asst. Prof. Dr. Güzin ÖZMEN

Nowadays, the use of deep learning models in medical image processing has gained momentum. Especially in studies on organ segmentation from slice images, deep learning methods are frequently preferred. Since the pancreas, located in the abdominal region, differs in shape, location and size in each person, its segmentation is quite challenging. To solve this problem, the U-Net model, which is one of the deep learning models, is generally preferred in the literature.

In this thesis, a new deep learning model based on the U-Net model with an architecture suitable for the number sequence in Pascal's triangle has been proposed for pancreatic segmentation. This proposed model is named Pascal U-Net model and the performance of the model is evaluated on two different data sets. First, The Cancer Imaging Archive Pancreas-CT dataset, which is a publicly available and frequently used dataset in the literature, was used. In addition, abdominal CT images taken from the Department of Radiology at Selcuk University Medical Faculty Hospital were used as the second data set. A slice image was selected for each patient from the records in the datasets and datasets for deep learning networks were created by applying preprocessing methods. In order to compare the pancreatic segmentation results obtained on both data sets with Pascal U-Net model, segmentation process was also performed on the same data sets with the U-Net model. Segmentation maps obtained as a result of 2, 4 and 6 fold cross validation and deep learning models run on different batch sizes from 1 to 10, were evaluated using 7 different performance metrics. Pancreas segmentation results performed with each batch size and different fold cross validation are the average of 10 run results. Both U-Net and Pascal U-Net segmentation results were analyzed based on 7 different metrics and visual evaluations. When the results are examined; in both data sets, Pascal U-Net model outperformed traditional U-Net architecture with a value of approximately 1% in terms of Dice Similarity Coefficient metric.

Keywords: Convolutional Neural Networks, Cross Validation, Deep Learning, Panceras Segmentation, Pascal U-Net, U-Net

ÖNSÖZ

Yüksek lisans tez çalışmalarım süresince değerli bilgileri ve kıymetli katkılarıyla beni yönlendiren danışmanım Konya Teknik Üniversitesi Mühendislik ve Doğa Bilimleri Fakültesi Elektrik-Elektronik Mühendisliği Bölümü Öğretim Üyesi Sn. Doç. Dr. Rahime CEYLAN'a, sabırla bana bu süreçte her zaman destek çıkan arkadaşlarıma ve bölümümüz öğretim elemanlarına, her türlü maddi ve manevi desteğini esirgemeyen aileme teşekkür ederim. Ayrıca bu çalışmada kullanılan abdomen bilgisayarlı tomografi görüntülerinin alınmasında destek sağlayan Selçuk Üniversitesi Tıp Fakültesi Radyoloji Bölümü Öğretim Üyeleri Sn. Prof. Dr. Mustafa KOPLAY'a, Sn. Doç. Dr. Hakan CEBECİ'ye, Sn. Arş. Gör. Mustafa Alper BOZKURT'a ve tüm çalışma arkadaşlarına teşekkürü bir borç bilirim.

Ender KURNAZ
KONYA-2021

İÇİNDEKİLER

SİMGELER VE KISALTMALAR

Simgeler

a	:Filtrenin görüntüyü tararken ilerlediği adım sayısı
$c(x,y)$	:x ve y Görüntüleri için Kontrast Terimi
DBK	:Dice Benzerlik Katsayısı metriği
$Doğruluk$	:Doğruluk Metriği
$Duyarlılık$	:Duyarlılık metriği
g	:YSA giriş değeri
$Kesinlik$	:Kesinlik Matrisi
$l(x,y)$	:x ve y Görüntüleri için Parlaklık Terimi
JO	:Jaccard Oranı
$m_{çıkış}$	:Özellik haritasının boyutu
m_{filtre}	:Filtrenin boyutu
$m_{giriş}$	:Giriş matrisinin boyutu
$Özgüllük$	:Özgüllük metriği
s	:Giriş matrisinin çevresine eklenecek sıfır sayısı
$s(x,y)$	:x ve y Görüntüleri için Yapısal Terimler
$sigm$	:Sigmoid aktivasyon fonksiyonu
tanh	:Tanh aktivasyon fonksiyonu
$relu$	:ReLU aktivasyon fonksiyonu
$YBİ(x,y)$	:x ve y Görüntüsü için Yapısal Benzerlik İndeksi Metriği
μ_x	:x Görüntüsünün Yerel Ortalaması
μ_y	:y Görüntüsünün Yerel Ortalaması
σ_x	:x Görüntüsünün Standart Sapması
σ_y	:y Görüntüsünün Standart Sapması
σ_{xy}	:x ve y Görüntülerinin Çapraz Kovaryansı

Kısaltmalar

AB U-Net	:Atlama Bağlantılı U-Net
ADD	:Ayrık Dalgacık Dönüşümü
BT	:Bilgisayarlı Tomografi
CA 19-9	:Kanser Antijeni 19-9
db2	:Daubechies Dalgacık Tipi
DBK	:Dice Benzerlik Katsayısı
DN	:Doğru Negatif
DP	:Doğru Pozitif
ESA	:Evrişimli Sinir Ağları
JO	:Jaccard Oranı
MRG	:Manyetik Rezonans Görüntüleme
SÜTFH	:Selçuk Üniversitesi Tıp Fakültesi Hastanesi
TCIA	:The Cancer Imaging Archive
TEA	:Tamamen Evrişimli Ağ
TSA	:Tekrarlayan Sinir Ağı
YBİ	:Yapısal Benzerlik İndeksi
YBEA	:Yoğun Bağlantılı Evrişim Ağları
YN	:Yanlış Negatif
YP	:Yanlış Pozitif
YSA	:Yapay Sinir Ağları

1. GİRİŞ

İnsan vücudunda karın bölgesinde yer alan organlardan biri olan pankreas, endokrin ve ekzokrin olmak üzere iki temel dokudan oluşmaktadır. Bu dokular, sindirimde ve glikoz metabolizmasında önemli rol oynarlar. Pankreas hem iç hem de dış salgı yapan ender organlardan birisi olup üç temel besin olan karbonhidrat, protein ve yağların sindirilmesi için gerekli olan enzimlerin salgılanmasını gerçekleştirmektedir.

Pankreas kanseri, kötü huylu bir tümörün genellikle pankreasın baş bölgesinde gelişmesi sonucu ortaya çıkmaktadır. Çoğunlukla agresif davranışlı olan bu tümörler çok kısa süre içerisinde büyümesine karşın diğer organlara baskı yapmadığı için hastada önemli belirtiler görülmemektedir.

Pankreasa bağlı hastalıkların tespitinde öncelikle hastadan kan ve idrar testi istenmektedir. Yapılan bu testlerde insülin, bilirubin, Kanser Antijeni 19-9 (CA 19-9), Carcinoembryonic Antigen değerlerinde anormallik tespit edilmesi halinde görüntüleme yöntemlerine başvurulabilmektedir. Görüntüleme yöntemleri olarak Manyetik Rezonans Kolanjopankreatografi Görüntüleme, Manyetik Rezonans Görüntüleme (MRG), Bilgisayarlı Tomografi (BT) ve Endoskopik Ultrason yöntemleri kullanılmaktadır.

Pankreas, her insanda şekil, boyut, konum gibi özellikler bakımından farklılık gösteren bir organdır. Karın bölgesinde bulunan pankreasın BT veya MRG yöntemleri kullanılarak tespit edilebilmesi için alanında uzmanlaşmış bir radyoloğa ve oldukça fazla zamana ihtiyaç duyulmaktadır. Bir hastadan alınan kesit sayısı BT'de ve MRG'de oldukça fazladır. Bu sorunun önüne geçebilmek adına, pankreası diğer organlardan ayırt edebilmek için literatürde bazı yöntemler kullanılmaktadır. Bu yöntemlerden birkaçı atlas tabanlı, süperpiksel tabanlı ve derin öğrenme tabanlı olarak sıralanabilir.

Temeli 1950 yıllarına dayanmasına rağmen, günümüzde Yapay Sinir Ağları (YSA) konusunda yapılan çalışmalar büyük önem kazanmıştır. YSA, özellikle biyomedikal alanında önemli bir ihtiyaç olarak karşımıza çıkmaktadır.

Bu çalışmada, önerilen yeni bir derin öğrenme modeli kullanılarak BT görüntülerinden pankreas segmentasyonu gerçekleştirilmiştir. Önerilen yeni derin öğrenme modeli iki farklı veri setine uygulanmıştır. İlk veri seti, The Cancer Imaging Archive (TCIA) veri tabanından alınan halka açık TCIA Pankreas BT veri setidir. Diğer veri seti, Selçuk Üniversitesi Tıp Fakültesi Hastanesi Radyoloji Bölümü'nden alınan abdomen BT kayıtlarından oluşturulmuştur. Gerçekleştirilen tez çalışmasında, bilinen U-Net modelinin kodlayıcı ile kod çözücü bölümü arasına evrişim bloklarının ve özellik geri

besleme bağlantılarının eklenmesiyle yeni bir derin öğrenme modeli tasarlanmıştır. Çalışmada önerilen Pascal U-Net modeli, literatürde bilinen U-Net modeli ile her iki veri seti üzerinde denenmiş ve sonuçlar karşılaştırmalı olarak 7 farklı metrik üzerinden sunulmuştur.

2. KAYNAK ARAŞTIRMASI

Bu çalışmada, abdomen BT görüntülerinden pankreas segmentasyonu için yeni bir derin öğrenme modeli tasarlanmıştır. Pankreas, her insanda büyüklük, boyut, şekil ve konum bakımından farklılık gösterdiğinden dolayı segmentasyonu oldukça zorlayıcıdır. Yapılan kaynak araştırması sonucunda, pankreas segmentasyonu ile ilgili söz edilen dezavantajlardan dolayı diğer organlarla ilgili gerçekleştirilmiş olan segmentasyon çalışmalarına kıyasla, pankreas segmentasyon çalışmalarının daha az sayıda ve daha az başarı oranında olduğu görülmüştür.

2.1. Medikal Alanda Derin Öğrenme ile İlgili Çalışmalar

Ronneberger ve ark. (2015), verileri daha verimli kullanabilmek adına veri artırımı yöntemine dayanan ve biyomedikal görüntü segmentasyonu konusunda büyük önem arz eden evrişim ağına bağlı bir yaklaşım olan U-Net modelini geliştirmişlerdir. Bu model daralma ve genişleme olmak üzere iki temel bölümden oluşmaktadır. Daralma bölümünde evrişim ve havuzlama katmanları vasıtasıyla girişte verilen görüntünün öznitelikleri öğrenilmektedir. Genişleme bölümünde ise boyutu indirgenmiş olan öznitelik haritalarının boyutu tekrar artırılarak girişteki imgenin boyutu elde edilmektedir. Fakat, genişleme bölümünde segmentasyonu yapılan nesnenin ne olduğu iyi öğrenilirken nesnenin nerede olduğu bilgisi kaybolmaktadır. Bu sorunu engellemek amacıyla daralma bölümünden genişleme bölümüne doğrudan bilgi aktarılmaktadır. Ronneberger ve arkadaşları geliştirmiş oldukları bu U-Net modelinin hızlı olduğunu ve biyomedikal görüntü segmentasyonu için kullanılabilir olduğunu göstermişlerdir.

Jegou ve ark. (2017), Evrişimli Sinir Ağı (ESA, Convolutional Neural Network) tabanlı bir ağ olan Yoğun Bağlantılı Evrişim Ağları (YBEA, DenseNets) modelinin görüntü işlemede önemli bir rol oynadığını ifade etmişler ve bu modeli genişleterek yeni bir YBEA modeli geliştirmişlerdir. Ortaya çıkarmış oldukları bu model, U-Net modelinin yapısına benzer bir mimariye sahip olmakla birlikte U-Net modelinde yer alan evrişim katmanları yerine yığın blokları kullanmışlardır. Bu sayede diğer yapılarla karşılaştırıldığında önerdikleri ağın parametrelerinin daha düşük sayıda olduğunu göstermişlerdir.

Zhou ve ark. (2018) çalışmalarında, Ronneberger ve ark.'nın (2015) gerçekleştirmiş oldukları U-Net modelini geliştirerek U-Net++ adında yeni bir model

ortaya çıkarmışlardır. U-Net modelinde daralma ve genişleme bölümleri arasında yalnızca atlama bağlantıları yer alırken, bu yeni U-Net++ modelinde daralma ve genişleme bölümleri arasında iç içe geçmiş bir yoğun yapıda evrişim katmanları yer almaktadır. Zhou ve ark. geliştirdikleri bu modelde, daralma ve genişleme bölümleri arasında meydana gelen anlamsal boşluğu azaltmayı amaçlamışlardır. U-Net modeliyle kıyasladıkları bu çalışmada, U-Net modeline göre Jaccard oranında %3.9 kazanç sağlamışlardır.

Tang ve ark. (2018) yaptıkları çalışmada, U-Net modeline bir düzenleme getirmişlerdir. Yığınlaştırılmış birçok U-Net modelinde her bir U-Net çiftini anlamsal blokları ile birleştirmişler ve Çift U-Net adını verdikleri modeli oluşturmuşlardır. Bu ardışık bağlantılar sayesinde bilgilerin U-Net'ler boyunca daha verimli bir şekilde iletildiğini ifade etmişlerdir. Çalışmalarında insan poz tahmini içeren iki farklı veri setinde bu modeli denemişler ve diğer modellerle kıyaslandığında birçoğundan daha iyi sonuç verdiğini belirtmişlerdir.

Han ve Ye (2018), U-Net modelindeki kısıtlamaları ortadan kaldırmak ve çoklu çözünürlüklü derin öğrenme şemaları sunan bir model geliştirmişlerdir. Seyrek görüşlü BT görüntülerinde yüksek frekanslı kenarların verimli bir şekilde iyileştiren U-Net modeline alternatif olarak çift çerçeveli ve sıkı çerçeveli U-Net yapısını çalışmalarında sunmuşlardır. Bu sayede, oluşturmuş oldukları bu modelin yeniden yapılandırma performansının daha iyi olduğunu göstermişlerdir.

Wang ve ark. (2019) yapmış oldukları çalışmada, BT, MRG ve endoskopik görüntüler içeren farklı segmentasyon görevlerinde bile uygulanabilen Yuvalanmış Genişleme Ağı geliştirmişlerdir. İlk birkaç katmanda daha büyük alanları yakalayan artık blokları yuvalanmış genişleme ile tasarlamışlardır. Ayrıca, daha başarılı segmentasyon sonucu elde etmek için tekrar düzenlenmiş bir odak kayıp fonksiyonunu bu ağa uygulamışlardır. Çoklu görev olarak gerçekleştirdikleri bu çalışma neticesinde diğer modellere göre daha başarılı sonuçlara ulaşmışlardır.

Zhang ve ark. (2019) yaptıkları çalışmada, geleneksel U-Net modelini üç yöntemle geliştirmişler ve LU-Net adında bir model sunmuşlardır. Bu yöntemlerden ilki, orijinal görüntüden özellikler çıkarılırken verimi artırmak amacıyla U-Net ve SE-Net (Hu ve ark, 2018) modellerini birleştirmişlerdir. İkinci olarak, modelde meydana gelen konum eksikliğini gidermek için U-Net modeli girişine çoklu skala girişi uygulamışlardır. Üçüncü yöntem olarak ise U-Net modelinde genişleme bölümünde üst örnekleme katmanı yerine piksel bazlı konum bilgisini koruyan bir örnekleme yöntemi

kullanmışlardır. Öne sürdükleri LU-NET modeli ile diğer modellerden daha başarılı sonuçlar elde edildiğini göstermişlerdir.

Xiuqin ve ark. (2019) çalışmalarında, geliştirilmiş derin öğrenme modeli U-Net ile retinal damar segmentasyonunu gerçekleştirmişlerdir. Yaptıkları bu çalışmada, retinal görüntüleri veri artırımı yöntemi ile çoğaltmışlar ve U-Net ağ yapısına artık modül eklemişlerdir. Damar çıkarımı için dijital retinal görüntüler veri seti üzerinde uyguladıkları model sonucu segmentasyon doğruluğunu %96.5 olarak bulmuşlardır.

Wu ve Zhao (2019), yerel özelliklere dayalı var olan segmentasyon yöntemlerinin ideal olmayan koşullar altında gerçek iris sınırlarını bulamadığını ifade etmişler ve bu çalışmalarında Yoğun U-Net modeli sunmuşlardır. Yoğun ağ ve U-Net ağını birleştirerek oluşturdukları bu modelin daha dar ve daha az parametreye sahip olduğunu belirterek geleneksel U-Net modeline göre daha avantajlı olduğunu ifade etmişlerdir. Oluşturdukları model ile birlikte yapmış oldukları iris segmentasyonu çalışmasının sonucunda %98.36 doğruluğa ve %97.07 F1 puanına ulaşmışlardır.

Wu ve ark. (2019) yaptıkları çalışmada, tamamen otomatik segmentasyon yapabilen Atlama Bağlantılı U-Net (SC U-Net) modelini öne sürmüşlerdir. MRG görüntülerinden beyinde bulunan beyaz madde hiperyoğunluklarının segmentasyonunu önerdikleri SC U-Net modeli ile gerçekleştirmişlerdir. U-Net modeli uyguladıklarında %74.99 Dice benzerlik oranı elde ederlerken, SC U-Net modeli ile bu oranı %78.36 olarak bulmuşlardır. Geliştirdikleri yapının daha hızlı yakınsamaya, daha düşük kayba ve daha yüksek segmentasyon doğruluğuna sahip olduğunu savunmuşlardır.

2.2. Pankreas Segmentasyonu ile İlgili Çalışmalar

Yapılan kaynak araştırması sonucu, pankreas segmentasyonu ile ilgili çalışmalar genellikle atlas tabanlı yöntemlerle ya da derin öğrenme yöntemleriyle gerçekleştirildiği gözlemlenmiştir. Pankreas segmentasyonunu içeren çalışmalar atlas tabanlı yapılan çalışmalar ve derin öğrenme modelleriyle gerçekleştirilen çalışmalar olmak üzere iki alt başlıkta incelenmiştir.

2.2.1. Atlas tabanlı pankreas segmentasyonu ile ilgili çalışmalar

Oda ve ark. (2012) çalışmalarında, üç boyutlu karın bölgesine ait BT görüntülerinden atlas seçimi ve grafik kesimi yöntemlerine dayanan bir yaklaşım

sunmuşlardır. Giriş görüntüsünde, imgeler arasındaki benzerliğe göre kümeleme yaptıkları görüntülere en çok benzeyen atlası seçmişlerdir. Segmentasyon yaptıkları organlardan karaciğer, dalak ve böbrekte %90'ın üzerinde doğruluğa ulaşırlarken, pankreasta bu oran en fazla %64'te kalmıştır.

Suzuki ve ark. (2012) yaptıkları çalışmada karın bölgesindeki BT görüntülerinden çoklu organ segmentasyonunun, cerrahi müdahaleyle alınan organlar bu görüntülerde olmadığından segmentasyonun başarısız olabileceğini belirterek bu sorunu çözmek amacıyla iki yöntem öne sürmüşlerdir. Bu yöntemler sırasıyla cerrahi operasyon sonrası organ hareketi ile organa özgü yoğunluk homojenliğinin anormallikleri test ederek olmayan organların otomatik tespiti ve otomatik olarak bulunmayan organları tespit eden karın bölgesindeki 10 adet organın atlas tabanlı çoklu organ segmentasyonunu gerçekleştirmişlerdir.

Wolz ve ark. (2013) yaptıkları çalışmada, çoklu atlas kaydı ve yama tabanlı segmentasyondan görüntüleri birleştirerek, bir atlas veri tabanından hedefe özgü öncelikleri oluşturan hiyerarşik bir atlas kaydı ve ağırlıklandırma şemasına dayanan bir yöntem kullanmışlardır. Çoklu organ segmentasyonu yapmak amacıyla uyguladıkları yöntemde karaciğer, böbrek ve dalak organlarında sırasıyla %94, %93 ve %92 Dice oranlarını elde etmelerine rağmen pankreasta bu oran %70 olmuştur.

Chu ve ark. (2013) gerçekleştirdikleri çalışmada, hastalar arasında lokal bölgelerdeki organların şekil ve konum bakımından büyük farklılıklar olması sorununu çözebilmek için uzamsal bölünmüş olasılıksal atlasa dayalı otomatik bir çoklu organ segmentasyonu yöntemi sunmuşlardır. Elde ettikleri uzamsal bölünmüş atlasın, organın şekli ve konumundaki varyasyonu verimli bir şekilde düşürdüğünü gözlemlemişlerdir. Kullanmış oldukları bu yöntemle gerçekleştirdikleri çoklu organ segmentasyonunda karaciğer, böbrek ve dalak için Dice benzerlik katsayılarını sırasıyla %95, %91 ve %90 oranında elde etmişlerdir. Ancak, bu katsayı pankreas organı için %69 olarak bulunmuştur.

Okada ve ark. (2015) çalışmalarında, birden çok organ arasındaki karşılıklı ilişkileri etkin bir şekilde birleştiren ve yoğunluk bilgisine ihtiyaç duymadan çeşitli görüntüleme koşullarına kolayca uyum sağlayan bir çoklu organ segmentasyon sistemi sunmuşlardır. İki ayrı hastaneden ve altı farklı görüntüleme koşullarında ele aldıkları BT görüntülerinden çoklu organ segmentasyonu gerçekleştirmişler ve pankreas organı için %73 Dice benzerlik oranına ulaşmışlardır.

Tong ve ark. (2015) yaptıkları çalışmada, seçilen atlas setlerinden farklı olan sınıflayıcıları içeren sözlükleri eş zamanlı bir şekilde öğrenen bir yöntem sunmuşlardır. Bu sözlükler ve sınıflayıcılara bağlı olarak daha önce görülmemiş görüntülerde öncelik sağlaması için olasılığa dayalı atlasları oluşturmuşlardır. Ayrıca, grafik kesim yöntemiyle son işlem uygulayarak segmentasyonu tamamlamışlardır. Buna ek olarak, karın bölgesindeki BT görüntülerinde hastalar arası oluşan farklılığı önlemek amacıyla voksel tabanlı bir atlas seçim stratejisi sunmuşlar ve pankreas organı için Dice benzerlik katsayısını %71 olarak bulmuşlardır.

Xu ve ark. (2015) çalışmalarında, edindikleri karın bölgesinin BT görüntülerinden 12 adet organın segmentasyonu için atlas seçimi ve füzyon tekniklerini tekrar gözden geçirmişler ve bağlam öğrenme yöntemiyle çoklu organ segmentasyonunun performansının artırılabildiğini göstermişlerdir. Önerdikleri yöntem ile tamamen otomatik bir çoklu organ segmentasyonu sistemi geliştirmişler ve bu sistem vasıtasıyla 12 organ arasından pankreasın Dice benzerlik katsayısını %45 olarak elde etmişlerdir.

Karasawa ve ark. (2017) yaptıkları çalışmada, pankreas organının yerini belirlemek için pankreasın etrafındaki damar yapılarından faydalanan bir atlas seçim stratejisi geliştirmişlerdir. Damar yapısının iki farklı uygulamasını da araştırdıkları çalışmada etiketlenmemiş görüntüler için yüksek pankreas benzerlikleriyle atlas seçimi gerçekleştirmişlerdir. Bu sayede segmente ettikleri pankreas, %78.5 Dice oranına ulaşmıştır.

Oliveira ve ark. (2018) çalışmalarında, abdomen ve göğüs bölgesine ait çoklu organ segmentasyonu gerçekleştiren yeni bir atlas tabanlı yöntem sunmuşlardır. İlk olarak, kabadan inceye bir yöntem ile organların global dönüşümleri yapılmış ve bu dönüşümleri yoğun bir deformasyon alanı yeniden oluşturma stratejisi ile birleştirmişlerdir. İkinci olarak, elde edilen aday segmentasyonlardan, organ bazlı etiket füzyon yaklaşımıyla nihai segmentasyonu gerçekleştirmişlerdir. Önerdikleri yöntem ile pankreas için ortalama Dice benzerlik katsayısını %70 olarak elde etmişlerdir.

2.2.2. Derin öğrenme yöntemleriyle gerçekleştirilen pankreas segmentasyonu ile ilgili çalışmalar

Zografos ve ark. (2015) yaptıkları çalışmada, bir atlas ile kayıt gerektirmeyen 3 boyutlu (3B) abdomen BT görüntülerinde çoklu organların segmentasyonu için yeni bir yapı sunmuşlardır. Bir dizi 3B hacimsel özellik üzerinde eğitilmiş ayırt edici

sınıflandırıcıları kullanmışlar ve ilgilenilen organların görünümünü örtülü olarak modellemişlerdir. Her seviyedeki eğitimli sınıflandırıcının bir sonraki seviyeye ek özellikler sağlamak için görüntüye geri uygulandığı hiyerarşik bir otomatik bağlam sınıflandırma şeması kullanmışlardır. Çoklu organ segmentasyonu yaptıkları bu çalışmada pankreas için %42 Jaccard benzerlik oranını elde etmişlerdir.

Roth ve ark. (2015) uyguladıkları çalışmada, abdomen BT görüntülerinden Basit Doğrusal Tekrarlayan Kümeleme yöntemiyle süperpikseller çıkararak olasılık haritaları oluşturmuşlardır. Olasılık haritalarını Evrişimli Ağ yapısı kullanarak pankreas segmentasyonunu gerçekleştirmişlerdir. 82 adet BT görüntüleri içeren veri setini 60 eğitim, 2 doğrulama ve 20 test olarak ayırmışlar, sonuçta da %68 Dice benzerlik oranı elde etmişlerdir.

Zhou ve ark. (2016) çalışmalarında, 3 boyutlu BT görüntülerini 2 boyuta indirgeyerek önce evrişim katmanı ardından ters evrişim katmanı uyguladıktan sonra görüntüleri tekrar 3 boyutlu hale getirmişlerdir. Bu yöntem sırasında 2 boyutlu görüntülerden aldıkları konum ve yön bilgilerini koruyarak çıkışa aktarmışlardır. Çoklu organ segmentasyonu için gerçekleştirdikleri bu çalışma sonucunda pankreas organı için %45 Jaccard oranına ulaşmışlardır.

Farag ve ark. (2016) gerçekleştirdikleri çalışmada, girişte kullandıkları BT görüntüleri süperpiksel haritası ve Rastgele Orman olasılık haritası olacak şekilde iki bölümden oluşan haritalar olarak elde etmişlerdir. Dört farklı yöntem kullanarak yaptıkları çalışmada, en kayda değer sonucu Evrişimli Sinir Ağ yapısıyla birlikte kaskat yapı ile %70.7 Dice oranı olarak elde etmişlerdir.

Zhou ve ark. (2017) yaptıkları çalışmada, 3 boyutlu BT görüntülerini axial, sagittal ve coronal olmak üzere üç açıya ayırmışlardır. İlk aşamada ayırdıkları her açıda ayrı ayrı 2 boyutlu Tamamen Evrişimli Ağ yöntemi kullanarak kabaca bir pankreas segmentasyon haritası oluşturmuşlardır. Bu segmentasyon haritalarını birleştirerek 3 boyutlu hale getirmişler ve ikinci aşamanın giriş görüntüleri olacak şekilde kırpmışlardır. Bu görüntüleri tekrar üç açı haline getirip aynı yöntemle eğitip kesin pankreas segmentasyon haritalarını oluşturmuşlardır. Bu çalışmada %82.4 oranında bir Dice benzerlik katsayısı elde edilmiştir.

Roth ve ark. (2018) yaptıkları çalışmada, abdomen bölgedeki organların segmentasyonunu gerçekleştirmek amacıyla Tamamen Evrişimli Ağ modellerinden biri olan U-Net modelini kullanmışlardır. Çoklu organ segmentasyonu yaptıkları çalışmada, 3 boyutlu U-Net modeli kullanmışlardır. Bu 3 boyutlu U-Net modeli iki kademeden

oluşmakta ve ikinci aşamanın girişi olarak ilk aşamada elde ettikleri segmentasyon sonuçlarını kullanmışlardır. İlk aşamasında organların ve damarların olabileceği bölgeleri kabaca belirleyerek, ikinci aşamada yapılacak olan sınıflama işlemini kolaylaştırmışlardır. 3 boyutlu çoklu organ segmentasyonu olarak gerçekleştirdikleri bu çalışmada pankreas için %82.2 Dice oranı sonucu elde etmişlerdir.

Gibson ve ark. (2018) çalışmalarında, karın bölgesinin BT görüntülerinden çoklu organ segmentasyonu için Yoğun V-Ağı adında bir model önermişlerdir. Organ segmentasyonunda kullanılan istatistiksel modeller ve atlas tabanlı yöntemlere nazaran, önerdikleri bu derin öğrenme modelinin daha iyi sonuç verdiğini göstermişler ve gerçekleştirdikleri uygulamada pankreas organı için %78 Dice benzerlik oranına ulaşmışlardır.

Roth ve ark. (2018) yaptıkları çalışmada, 3 boyutlu BT görüntülerinden oluşan veri setleri üzerinde sırasıyla pankreas konumunu belirleyen ve pankreas segmentasyonunu gerçekleştiren iki aşamalı kaskat yapı kullanmışlardır. Bu modelde Bütünsel-yuvalanmış Sinir Ağ (Holistically-nested Neural Network) kullanmışlar ve 4 katlı çapraz doğrulama yöntemiyle modeli eğitmişlerdir. Bütünsel-yuvalanmış Sinir Ağ ve Rastgele Orman yöntemini beraber kullanarak %81.27 Dice benzerlik oranına ulaşmışlardır.

Yang ve ark. (2019) yaptıkları çalışmada, BT görüntülerinden Tamamen Evrişimli Ağ (TEA) ve Tekrarlayan Sinir Ağ (TSA) yapılarından oluşan kaskat sinir ağ yapısı kullanarak kesitler arası ve kesit içi bilgilerle pankreasın segmentasyonunu gerçekleştirmişlerdir. TEA yapısını pankreas segmentasyonu için kesit içi bilgileri çıkarmak amacıyla, TSA yapısını ise kesitler arası bilgileri çıkarmak adına kullanmışlardır. 4 katlı çapraz doğrulama yöntemiyle gerçekleştirdikleri bu çalışma şimdiye kadarki bu alanda yapılan en yüksek sonuç olan %87.72 Dice benzerlik oranına ulaşmıştır.

Man ve ark. (2019) yaptıkları pankreas segmentasyonu konusundaki çalışmada, deforme edilebilir U-Net modeli kullanan bir Derin Q Ağ yöntemi ile bu soruna yaklaşmışlardır. 3 boyutlu pankreas görüntülerini axial, coronal ve sagittal kesitlerini ayrı ayrı önce Derin Q Ağı ile konumunu tespit etmişler, daha sonra U-Net modeli ile pankreasın 3 boyutta segmentasyonunu elde etmişlerdir. Elde ettikleri 3 farklı açıdaki segmentasyon haritalarını Çoğunluk Oylaması yöntemiyle tek bir sonuç haline getirmişlerdir. Yaptıkları çalışma sonucunda ortalama Dice oranını %86.93 olarak bulmuşlardır.

Lu ve ark. (2019) yaptıkları çalışmada, biyomedikal segmentasyon alanında başarıya ulaşan bir model olan U-Net modelinde alt örnekleme ve üst örnekleme arasındaki bilgi alışverişini artırmak amacıyla dönen bir artık katman eklemişler ve bu dönen artık katman sayesinde kayda değer sonuçlar elde etmişlerdir. Buna ek olarak, kullandıkları modelde kayıp fonksiyonu olarak Dice katsayısını kullanmak yerine, sadece alan benzerliğiyle değil ayrıca şekil benzerliğine de odaklanan yeni bir kayıp fonksiyonu önermişlerdir. 10 katlı çapraz doğrulama yöntemi ile gerçekleştirdikleri pankreas segmentasyonu uygulamasında %88.32 Dice benzerlik oranına ulaşmışlardır.

Liu ve ark. (2019) gerçekleştirdikleri çalışmada, iki aşamadan oluşan bir evrişimli ağ modeli geliştirerek pankreas segmentasyonu gerçekleştirmişlerdir. İlk aşamada 3 boyutlu BT görüntülerinden aday bölgeyi süperpiksel kullanarak belirlemişler ve 2.5 boyutlu görüntüler oluşturmuşlardır. Ardından ikinci aşamada farklı kayıp fonksiyonlar içeren 5 adet U-Net modeli ile ağı eğiterek bir birleştirme modeli yarımıyla bu beş farklı sonucu segmentasyon çıkışı olarak elde etmişlerdir. 4 katlı çapraz doğrulama yaparak gerçekleştirdikleri bu çalışmada, Dice katsayını %84.1 olarak bulmuşlardır.

2.3. Kaynak Araştırması Özeti

Yapılan kaynak araştırması sonucunda;

- Pankreasın her insanda şekil, konum, büyüklük bakımından farklı olmasından dolayı segmentasyonunun zor olduğu tespit edilmiştir.
- Alanında uzman radyologların, pankreası BT veya MRG ile elde edilen görüntülerinden tespit etmek için oldukça fazla zaman ve bilgi birikimine ihtiyaç duydukları görülmüştür.
- Pankreas kanserinin teşhisi konduktan sonra iyileşme süreci için geç olduğu ve bu sebepten görüntülemede erken teşhisin çok önemli olduğu ortaya çıkmıştır.
- Derin öğrenme yöntemleriyle yapılan segmentasyon çalışmalarında çoğunlukla çapraz doğrulama yönteminin tercih edildiği görülmüştür.
- Çalışmalarda genellikle segmentasyon sonuçlarının bir veya iki metrik kullanılarak değerlendiği gözlenmiştir.

Bu tez çalışmasında, kaynak araştırması sonucu tespit edilen dezavantajları elimine ederek abdomen BT görüntülerinden yüksek doğrulukla pankreas segmentasyonunun gerçekleştirilmesi hedeflenmiştir. Bu amaç doğrultusunda, son yıllarda biyomedikal görüntü segmentasyonunda çok kullanılan derin öğrenme yöntemlerinden biri olan U-Net modeli modifiye edilerek yeni bir derin öğrenme modeli (Pascal U-Net) önerilmiştir. Önerilen modele ait pankreas segmentasyon sonuçları iki ayrı veri seti için ve 7 farklı metrik üzerinden değerlendirilmiştir.

3. MATERYAL VE YÖNTEM

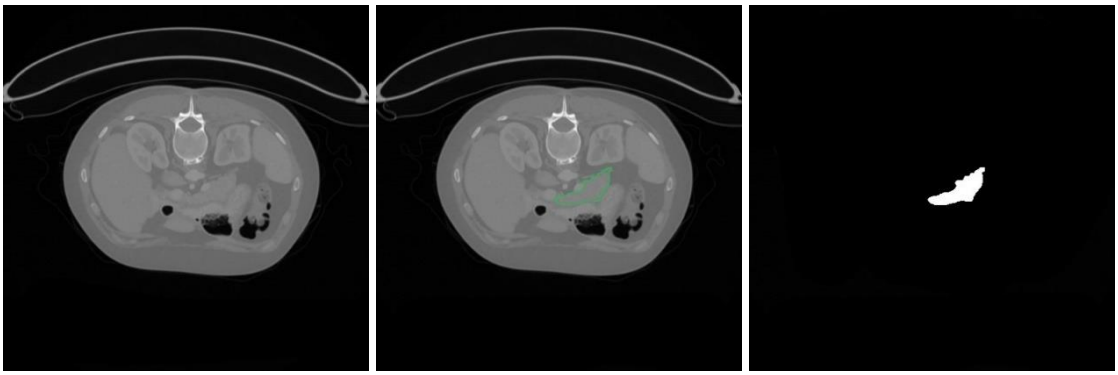
Pankreas segmentasyonu için önerilen Pascal U-Net modeli ve geleneksel U-Net modeli ile kullanılan veri setinin özellikleri bu bölümde detaylı olarak sunulmuştur. Önerilen Pascal U-Net modelinin başarımı iki farklı veri seti üzerinde denenmiş ve sonuçlar geleneksel U-Net modeli ile karşılaştırmalı olarak değerlendirilmiştir.

3.1. Kullanılan Veri Setlerinin Özellikleri

Bu tez çalışmasında abdomen BT görüntülerinden oluşan iki farklı veri seti kullanılmıştır. Bunlardan ilki The Cancer Imaging Archive (TCIA) veri tabanından alınan halka açık Pankreas BT veri setidir. Diğeri ise Selçuk Üniversitesi Tıp Fakültesi Hastanesi (SÜTFH) Radyoloji Bölümü'nden alınan abdomen BT görüntülerinden oluşan veri setidir. Çalışmada kullanılan bilgisayarın ekran kartının sınırlı olmasından dolayı her iki veri setinde de her hastadan pankreasın segmente edilebildiği bir kesit seçilerek derin öğrenme modelleri için veri setleri oluşturulmuştur.

3.1.1. TCIA Pankreas BT veri seti

Gerçekleştirilen tez çalışmasında kullanılan ilk veri seti TCIA veri tabanından alınan Pankreas BT veri setidir. Amerika Birleşik Devletleri'nde bulunan Ulusal Sağlık Enstitüleri (National Institutes of Health) Klinik Merkezi'nden alınan bu veri setinde, yaşları 18 ila 76 arasında değişen 53 erkek ve 27 kadından alınmış 82 adet kontrastı artırılmış 512 x 512 piksel boyutlarında BT görüntüsü mevcuttur (Roth ve ark, 2016). Bu BT görüntüleri toplamda 19328 kesitten oluşmaktadır. Ekran kartı sınırından dolayı her hasta için alınan BT kayıtlarından bir kesit seçilmiş ve önişleme aşamaları uygulanarak derin öğrenme modellerinin eğitim ve testi için toplam 82 görüntü içeren bir veri seti oluşturulmuştur. Şekil 3.1'de TCIA veri setindeki bir orijinal BT görüntüsü, bu görüntünün etiketlenmiş hali ve maskesi istenilen segmentasyon sonucu verilmiştir.

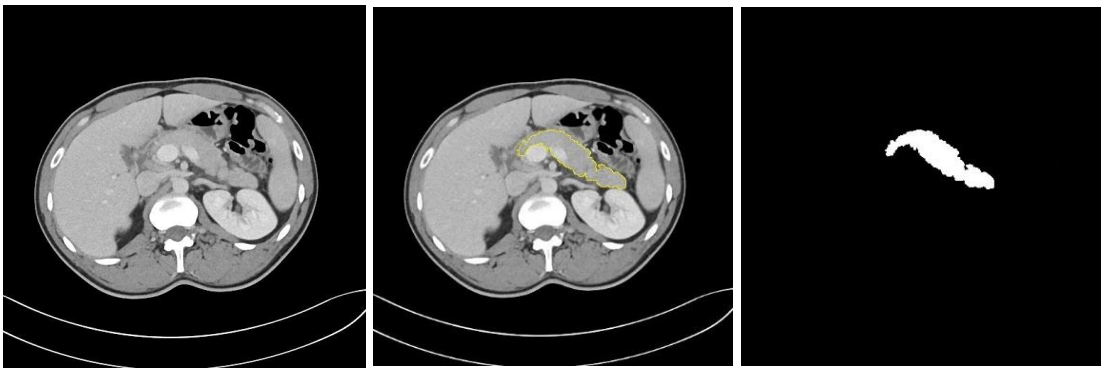


(a) (b) (c)

Şekil 3.1. TCIA veri setinden bir abdomen BT görüntüsü (512 x 512 piksel) **(a)** Orijinal görüntü **(b)** Etiketlenen görüntü **(c)** Maske

3.1.2. SÜTFH veri seti

Gerçekleştirilen çalışmada ikinci veri seti, olarak kullanılan görüntüler Selçuk Üniversitesi Tıp Fakültesi Hastanesi (SÜTFH) Radyoloji Bölümü'nden alınan abdomen BT görüntülerinden oluşturulmuştur. Veri setindeki abdomen BT görüntüleri, 2 x-ışını tüpünün birbirine 95 derece açıyla dizildiği, 2 adet dedektör seti içeren 128 sıralı çift kaynaklı BT sistemi (Somatom Definition Flash; Siemens Healthcare, Forchheim, Almanya) ile çekilmiştir. İnceleme protokolü 120 kVp, 512 x 512 matris ve 64 x 0.6 mm kolimasyon şeklindedir. Bu veri setindeki görüntüler SÜTFH Radyoloji Bölümü'nde alanında uzman bir radyolog tarafından etiketlenmiştir. Şekil 3.2'de SÜTFH veri setindeki orijinal bir BT görüntüsü, bu görüntünün etiketlenmiş hali ve etiketlenen görüntünün maskesi verilmiştir.



(a) (b) (c)

Şekil 3.2. SÜTFH veri setinden bir abdomen BT görüntüsü (512 x 512 piksel) (a) Orijinal görüntü (b) Etiketlenen görüntü (c) Maske

Bu çalışmadaki SÜTFH veri seti, 28 kadın ve 28 erkek olmak üzere toplamda 56 adet hastadan alınmış abdomen BT görüntülerini içermektedir. Ekran kartının izin verdiği işlem kapasitesine uygun olarak her hastanın BT kayıtlarından bir kesit seçilmiş ve gerekli işlemler uygulanarak toplam 56 görüntüden oluşan bir veri seti oluşturulmuştur.

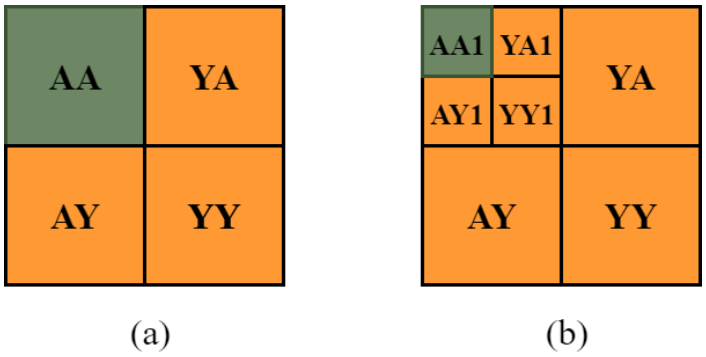
3.2. İki Seviyeli 2 Boyutlu Ayrık Dalgacık Dönüşümü

Dalgacık dönüşümü, nesne algılama ve sınıflandırma için bir görüntü işleme tekniği olarak kullanılmaktadır. Dalgacıklar geçmişte görüntüleri analiz etmek için uygulanmış ve radar görüntülerinden benek gürültüsünün giderilmesi, yüksek spektral çözünürlüklü görüntülerin yüksek uzaysal çözünürlüklü görüntülerle birleştirilmesi ve doku analizi ve sınıflandırması gibi uzaktan algılamadaki birçok uygulamada kullanılmıştır (Ghazali ve ark, 2007).

Ayrık Dalgacık Dönüşümü (ADD, Discrete Wavelet Transform)'nün temel fikri, zaman-frekans gösterimini sağlamaktır. 2 boyutlu ADD, bir dizi kaydırılmış ve ölçeklenmiş dalgacıklar ile bir görüntünün temsil edilmesidir (Ghazali ve ark, 2007).

2 boyutlu ADD genel olarak iki aşamadan oluşur. İlk aşamada, görüntüye 1 boyutlu ADD uygulanır, ardından alçak frekans bileşenlerini içeren alt bant A'yı ve yüksek frekans bileşenlerini içeren alt bant Y'yi elde etmek için dikey alt örnekleme yapılır. İkinci aşamada, A ve Y'ye başka bir 1 boyutlu ADD uygulanır ve yatay alt örnekleme ile AA, AY, YA ve YY alt bantları elde edilir (Chang ve Girod, 2007).

Bir görüntüye bir seviyeli 2 boyutlu ADD uygulandığında alçak frekans bilgilerini içeren AA alt bant görüntüsünün boyutu, ham görüntünün boyutunun yarısı kadar olmaktadır. AA alt bant görüntüsüne iki seviyeli 2 boyutlu ADD uygulanarak AA1, AY1, YA1 ve YY1 alt bantları elde edilir. AA1, AY1, YA1 ve YY1 alt bant görüntülerinin boyutları ise, ham görüntünün boyutunun dörtte biri kadar olmaktadır. Şekil 3.3 (a)'da bir seviyeli 2 boyutlu ADD, Şekil 3.3 (b)'de ise iki seviyeli 2 boyutlu ADD işlemine bir örnek verilmiştir.

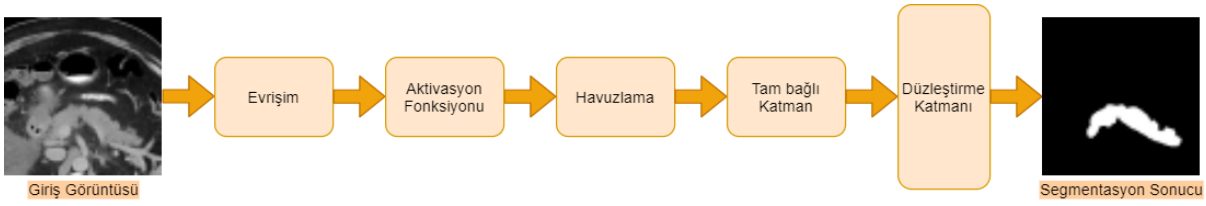


Şekil 3.3. Bir görüntünün ADD analizi (a) Bir seviyeli (b) İki seviyeli

3.3. Evrişimli Sinir Ağları

Hastalık teşhisinden elektronik devrelerin analizine, insan yazısını ve dilini anlamadan araç kullanmaya daha birçok alanda insan zekasına ihtiyaç duymadan bu aktiviteleri gerçekleştirebilen sistemler Yapay Zeka olarak tanımlanmaktadır (Nilsson, 2014). Son yıllarda yapay zeka alanındaki çalışmalar derin öğrenme modelleri üzerine yoğunlaşmıştır. Evrişimli Sinir Ağları (ESA, Convolutional Neural Networks)'nın temeli ilk olarak 1980'li yıllarda Fukushima ve Miyake tarafından ortaya atılmıştır. Fukushima ve Miyake, Hubel ve Wiesel'in maymunlar ve kuşların görsel korteksleri ile ilgili yaptıkları çalışmadan esinlenerek yayınlarında evrişim işleminden ve ESA modelinden bahsetmişlerdir (Fukushima ve Miyake, 1982). Her ne kadar ESA'nın temeli bu yıllara dayandırılsa da LeCun ve arkadaşlarının gerçekleştirdiği çalışmalar, ESA'nın bugünkü seviyeye gelmesinde önemli rol oynamıştır. LeCun ve ark., LeNet-5 adında farklı parametreler için uyarlanabilir ağırlıklar ve geri besleme algoritması içeren 7 seviyeli evrişimli ağı tasarlamıştır. Günümüzde ESA yapıları, halen daha LeCun ve ark. tarafından oluşturulan bu LeNet-5 mimarisinden esinlenilerek gerçekleştirilmektedir (LeCun ve ark, 1998).

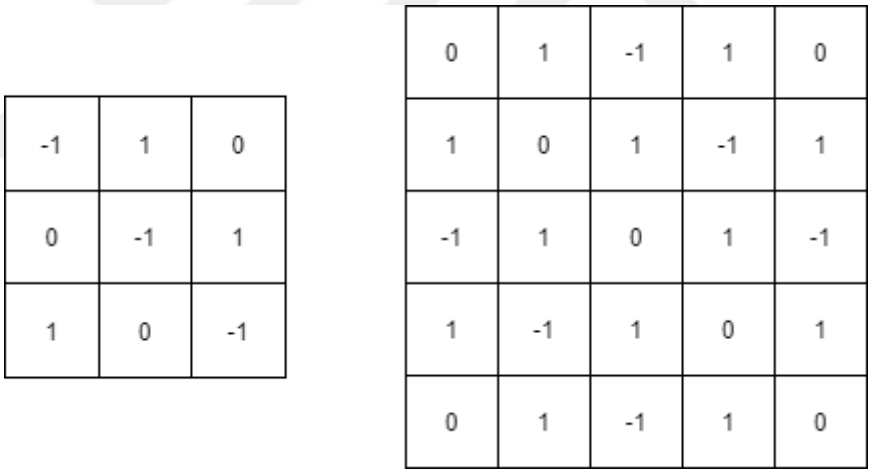
Bir ESA yapısı evrişim katmanı, aktivasyon fonksiyonu, havuzlama katmanı, tam bağlı katman ve düzleştirme katmanından oluşmaktadır. Basit bir ESA yapısı Şekil 3.4'te gösterilmiştir.



Şekil 3.4. Basit bir ESA yapısı

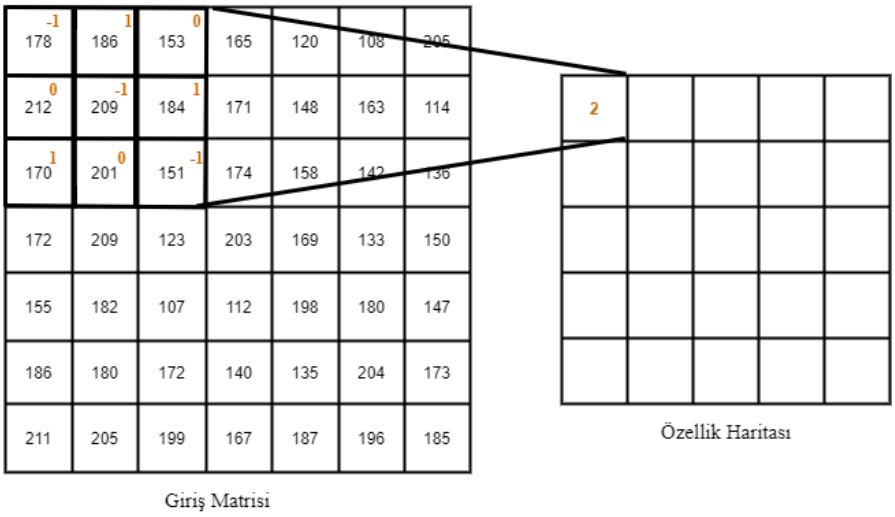
3.3.1. Evrişim

Bir görüntü içerisinde yer alan yüksek ve düşük seviyeli özellikleri çıkarmak amacıyla evrişim (konvolüsyon) işlemi yapılır. Evrişim, matrislerden oluşan görüntü boyunca, daha küçük boyutlu bir filtrenin uygulanmasıyla özellik çıkarımı için kullanılan bir işlemdir (Albawi ve ark, 2017). Filtrenin her bir öğesi ile giriş matrisine denk gelen elemanların değerleri çarpılır ve çıkış matrisine karşılık gelen konumda çıkış değerini elde etmek için toplanır. Çıkış olarak elde edilen matris, özellik haritası olarak adlandırılır (Albawi ve ark, 2017). Bu işlem, farklı özellik haritalarının çıkarılması amacıyla çoklu filtreler uygulanarak tüm giriş görüntüsü taranıncaya dek devam eder. Büyük boyutlu filtre kullanılması evrişim uygulandıktan sonra oluşacak özellik haritasının küçük olmasına neden olmaktadır. Bu durum bilgi kaybına yol açtığından genelde 3x3, 5x5 gibi küçük boyutlu filtreler kullanılmaktadır. 3x3 ve 5x5 boyutlarında örnek filtreler Şekil 3.5'te yer almaktadır.

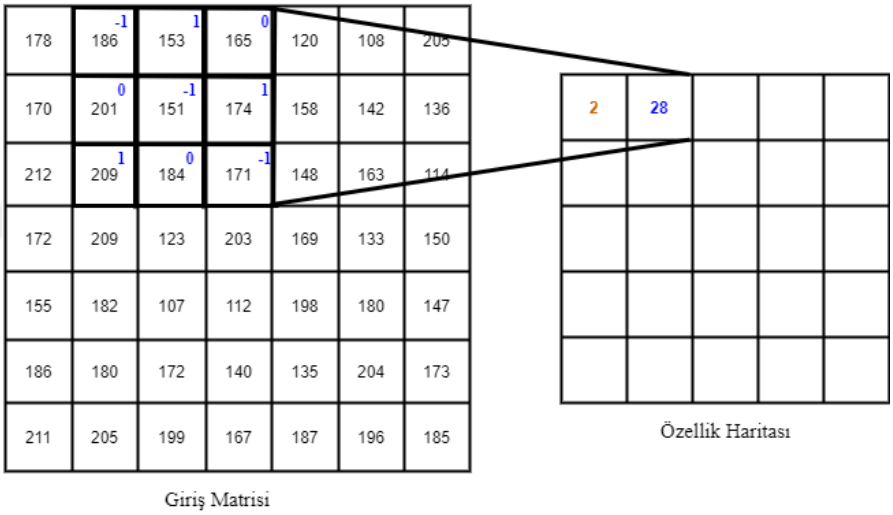


Şekil 3.5. Örnek filtreler (a) 3x3 boyutlu bir filtre (b) 5x5 boyutlu bir filtre

Şekil 3.5 (a)'da yer alan 3x3 boyutlu bir filtrenin giriş matrisine uygulanarak özellik haritasının ilk elemanını oluşturan evrişim işlemi Şekil 3.6'da, özellik haritasının ikinci elemanının elde edildiği evrişim işlemi Şekil 3.7'de gösterilmiştir.



Şekil 3.6. Bir özellik haritasının ilk elemanının evrişim işlemi ile eldesi



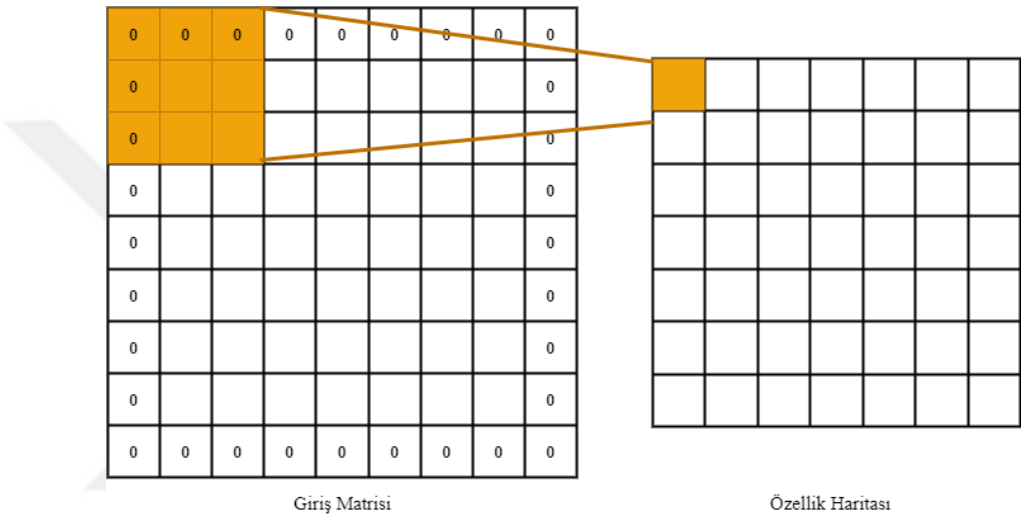
Şekil 3.7. Bir özellik haritasının ikinci elemanının evrişim ile eldesi

Evrişim işlemi sonrasında çıkış olarak elde edilen özellik haritasının boyutu Eşitlik 3.1'de ifade edildiği şekilde hesaplanır (Albawi ve ark, 2017).

$$m_{çıkış} = \frac{m_{giriş}+2s-m_{filtre}}{a} + 1 \tag{3.1}$$

Burada $m_{giriş}$ giriş matrisinin boyutunu, s giriş matrisinin çevresine eklenecek sıfır sayısını, m_{filtre} filtrenin boyutunu, a filtrenin görüntüyü tararken ilerlediği adım sayısını ve $m_{çıkış}$ çıkışta elde edilen özellik haritasının boyutunu ifade etmektedir.

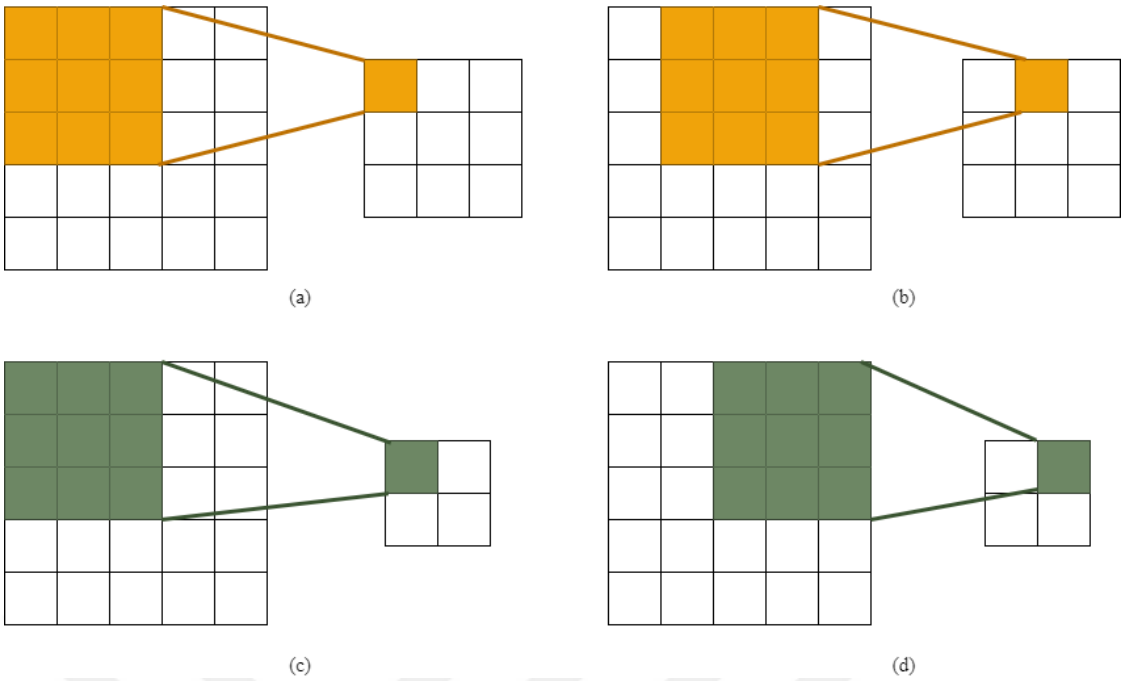
Evrişim işlemi sırasında giriş matrisine uygulanan filtre, matrisin en dışında kalan elemanlarıyla örtüşememekte ve dolayısıyla çıkışta elde edilen özellik haritasının boyutu girişteki matrisle aynı olmamaktadır. Bu sorunu önlemek amacıyla Piksel Ekleme (Padding) yöntemi kullanılmaktadır (Albawi ve ark, 2017). Piksel Ekleme yöntemlerinden literatürde en çok tercih edileni Sıfır Ekleme (Zero Padding) yöntemidir. Sıfır Ekleme, giriş matrisinin çevresi boyunca sıfır değerlerinin eklendiği bir tekniktir. Şekil 3.8'de giriş matrisinin çevresine sıfırlar eklenerek evrişim ile elde edilen özellik haritasına bir örnek verilmiştir.



Şekil 3.8. Giriş matrisine sıfır ekleme yöntemi uygulanarak elde edilen özellik matrisine bir örnek

Şekil 3.8'den de görülebileceği üzere 7x7 boyutlarındaki giriş matrisinin çevresine sıfır değerleri eklenerek yeni bir matris elde edilir. Sıfır ekleme yöntemi ile Eşitlik 3.1'den faydalanarak giriş matrisi ile aynı boyutlarda bir özellik haritası elde edilebilmektedir.

Ayrıca, Eşitlik 3.1'de yer alan a adım sayısı, evrişim işlemi uygulanırken filtrenin kaydırılmasıyla ilişkilidir. Kaydırma, bir filtrenin giriş matrisi üzerinde her işlem için kaç birim kaydırılacağını ifade etmektedir (Albawi ve ark, 2017). Kaydırma işlemine bir örnek Şekil 3.9'da gösterilmiştir.

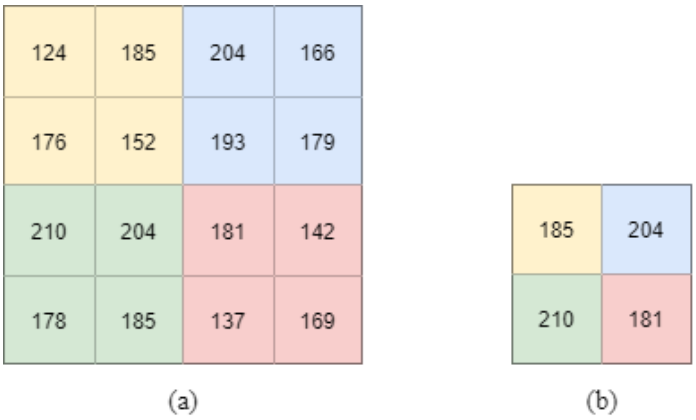


Şekil 3.9. Kaydırma işlemine bir örnek (a) a adım sayısı 1 iken özellik haritasının birinci elemanının eldesi (b) a adım sayısı 1 iken özellik haritasının ikinci elemanının eldesi (c) a adım sayısı 2 iken özellik haritasının birinci elemanının eldesi (d) a adım sayısı 2 iken özellik haritasının ikinci elemanının eldesi

3.3.2. Havuzlama katmanı

ESA'da evrişim katmanından sonra yer alan havuzlama katmanı, derin ağlarda hesap yükünü azaltmak amacıyla kullanılmaktadır. Havuzlama işleminde görüntünün derinliği değiştirilmeden uzamsal boyutları düşürülmektedir. Havuzlama katmanından önce elde edilen büyük özellik haritalarıyla gerçekleştirilecek eğitim aşaması, oldukça fazla işlem yüküne sahiptir. Havuzlama katmanı vasıtasıyla bu katmana giriş olarak verilen özellik haritalarının uzamsal boyutu ve parametre sayıları azaltılarak işlem yükü azaltılmaktadır (Albawi ve ark, 2017).

Havuzlama yöntemlerinden yaygın olanları maksimum havuzlama ve ortalama havuzlamadır. Derin öğrenmede sıklıkla kullanılan yöntem, yerel bölgedeki en baskın özelliği seçen maksimum havuzlama yöntemidir. Literatürde genellikle 2x2 maksimum havuzlama ile karşılaşılmaktadır. 2x2 maksimum havuzlama işlemine bir örnek Şekil 3.10'da gösterilmiştir.



Şekil 3.10. 2x2 maksimum havuzlama işlemine bir örnek (a) orijinal görüntü (b) maksimum havuzlama

3.2.3. Aktivasyon fonksiyonları

ESA'nın önemli yapı taşlarından olan aktivasyon fonksiyonları, ağın giriş değerlerini istenilen aralıkta tutmaya yarayan fonksiyonlardır. Eğer aktivasyon fonksiyonu ağa uygulanmazsa çıkışta üretilen sinyal basit bir doğrusal fonksiyon olarak karşımıza çıkmaktadır. Ağın doğrusal olmayan durumları da öğrenebilmesi amacıyla aktivasyon fonksiyonları kullanılmaktadır. Literatürde sıklıkla karşılaşılan aktivasyon fonksiyonları sigmoid, tanh ve ReLU fonksiyonlarıdır. Sigmoid aktivasyon fonksiyonu Eşitlik 3.2'de, tanh aktivasyon fonksiyonu Eşitlik 3.3'te ve ReLU aktivasyon fonksiyonu Eşitlik 3.4'te tanımlandığı gibi ifade edilir (Szandała, 2021).

$$sigm = \frac{1}{1+ e^{-g}} \tag{3.2}$$

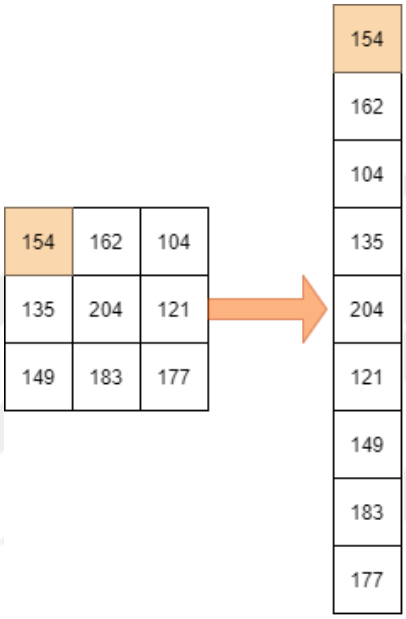
$$\tanh = \frac{1-e^{-g}}{1+ e^{-g}} \tag{3.3}$$

$$relu = \max(0, g) \tag{3.4}$$

Eşitlik 3.2-3.4'te yer alan g giriş değerini ifade etmektedir. Sigmoid aktivasyon fonksiyonu giriş değerlerini 0 ile 1 arasına getirirken, tanh aktivasyon fonksiyonu ise giriş değerlerini -1 ile 1 arasına getirir. ReLU aktivasyon fonksiyonu ise negatif değerleri elimine ederek pozitif değerlerin geçmesini sağlayan bir aktivasyon fonksiyonudur (Albawi ve ark, 2017).

3.3.4. Düzleştirme katmanı

Evrişim katmanları sonucunda matris şeklinde elde edilen özellik haritalarının segmentasyon sonucunda sınıflandırma yapılabilmesi amacıyla tek boyutlu vektör haline dönüştürülmesi gerekmektedir. Bu şekilde matris halinde olan değerlerin vektörel hale getirilmesi işlemine düzleştirme denilmektedir (Jin ve ark, 2014). Bir düzleştirme işlemi örneği Şekil 3.11'de sunulmuştur.



Şekil 3.11. Düzleştirme işlemine bir örnek

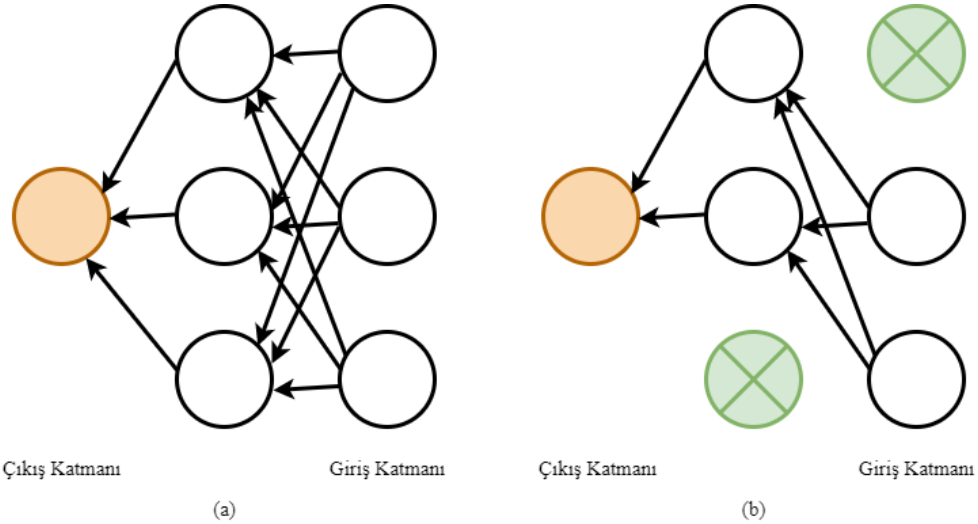
3.3.5. Tam bağlı katman

Derin ağlarda tam bağlı katmana kadar olan katmanlarda özellik haritalarının eldesi ve modeli düzleştirme gibi işlemler yapılmaktadır. Bu katmanda ise yapay sinir ağı model yapısıyla aynı şekilde, özellik haritalarının sınıflandırılması gerçekleştirilir (Albawi ve ark, 2017).

3.3.6. Unutturma katmanı

Derin öğrenme uygulamalarında, modelin eğitimi sırasında veriler başarılı bir şekilde öğrenilirken, teste gelindiğinde bu başarı oranı düşüyorsa buna aşırı öğrenme denir. Aşırı öğrenme problemini ortadan kaldırmak amacıyla veri artırımı yöntemi kullanılır, veri setine yeni veriler ekleyerek veri sayısı artırılır veya model yapısı içerisine

unutturma katmanları yerleştirilir (Srivastava ve ark, 2014). Unutturma işlemine bir örnek Şekil 3.12'de verilmiştir.

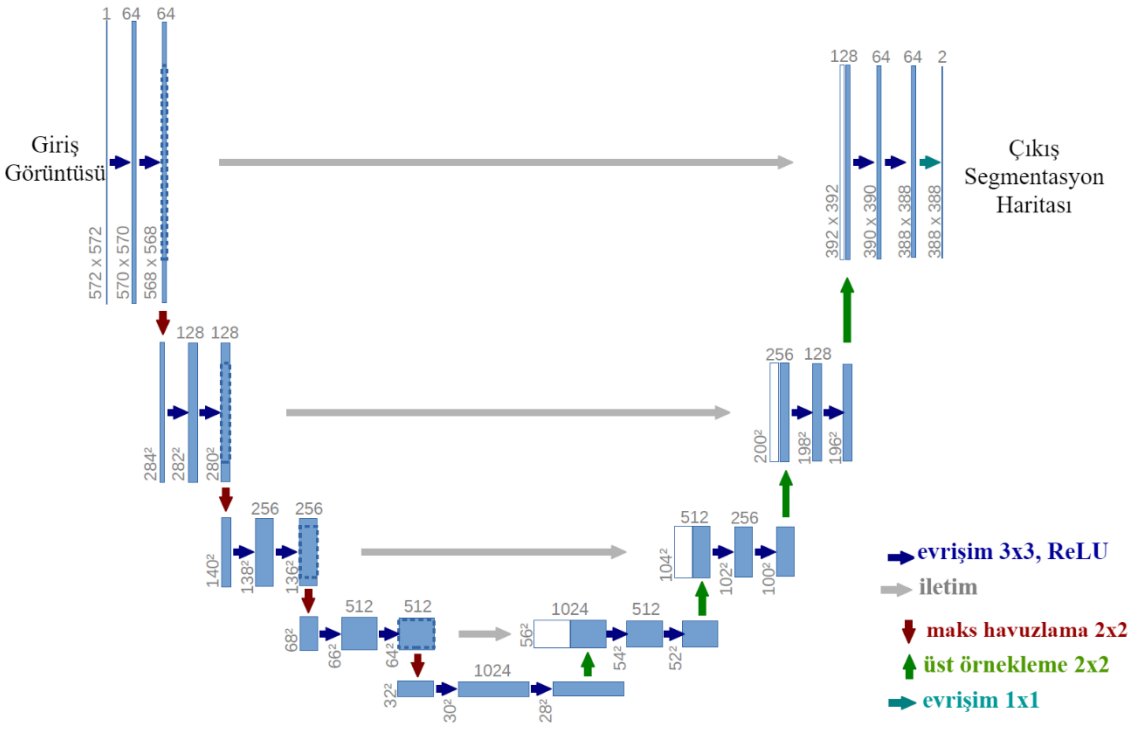


Şekil 3.12. Unutturma işlemine bir örnek (a) Normal bir ağ yapısı (b) Unutturma katmanı eklenmiş bir ağ yapısı

Unutturma katmanında, modelin eğitimi sırasında modelde yer alan yapay nöronlar belirlenen bir oranda rastgele seçilerek değerleri sıfıra eşitlenir. Bu sayede, oluşturulan ağ ezberlemeden sürekli olarak ağırlık değerlerini en uygun hale getirir.

3.4. U-Net Modeli

Biyomedikal alanda gerçekleştirilen çalışmalar derin öğrenme modellerinin gelişmesiyle hız kazanmıştır. Özellikle ESA temelli yapıların geliştirilmesi bu alanda büyük önem kazanmıştır. LeCun ve ark., 1998 yılında oluşturduğu LeNet-5 modeli sayesinde el yazısıyla yazılan sayılardan oluşan MNIST veri setini çözümleyen bir model ortaya atmıştır (LeCun ve ark, 1998). Ronneberger ve ark. 2015 yılında geliştirdikleri U-Net modeli ile segmentasyon yarışmasında büyük bir başarı elde etmişler ve biyomedikal alanda gerçekleştirilebilecek bir model elde ettiklerini ifade etmişlerdir. U-Net modeli Şekil 3.13'teki gibidir (Ronneberger ve ark, 2015).



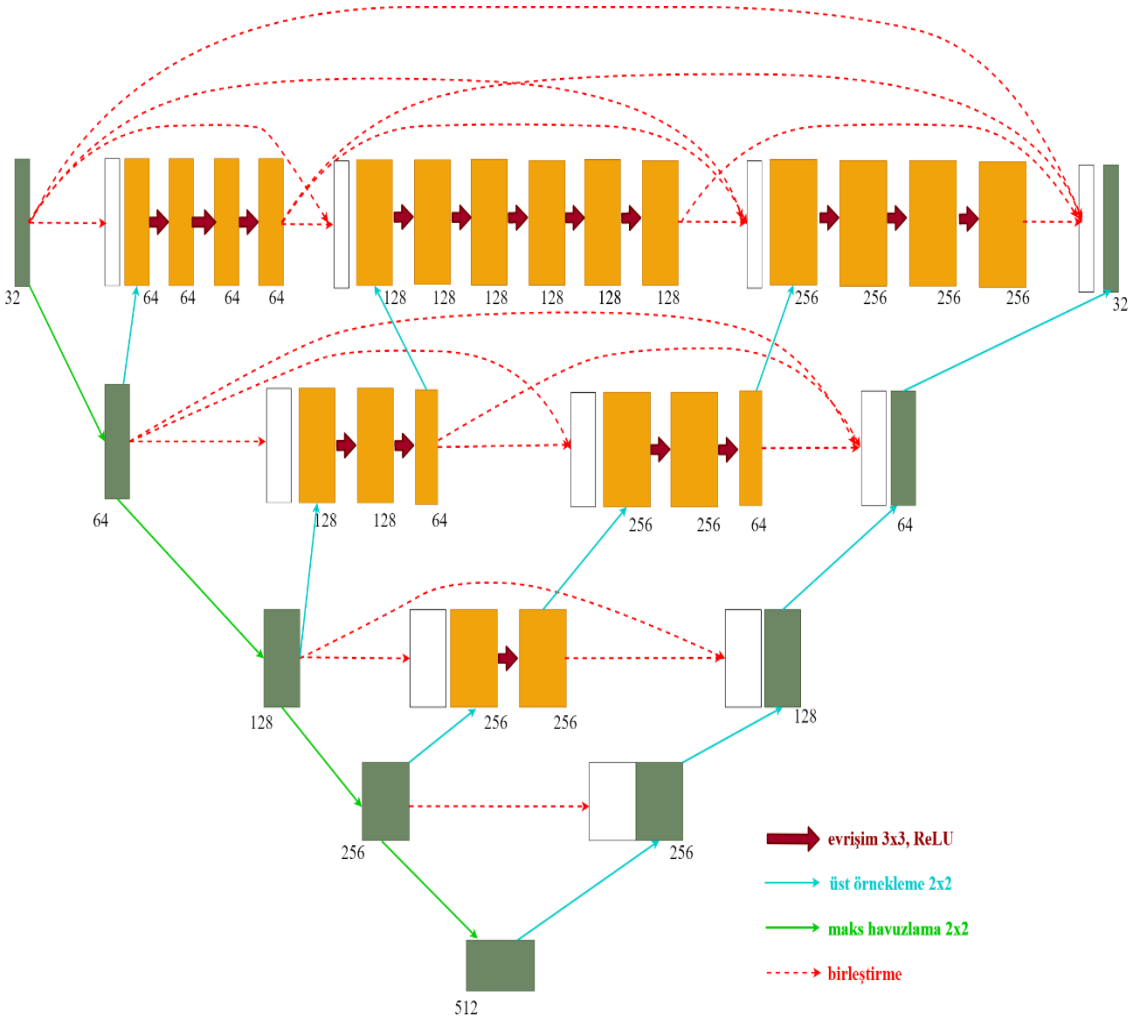
Şekil 3.13. U-Net modeli mimarisi (Ronneberger ve ark., 2015)

U-Net modeli temel olarak kodlayıcı ve kod çözücü olmak üzere iki bölümden oluşmaktadır. Kodlayıcı bölümünde ağın girişine verilen giriş görüntüleri iki kez art arda evrişim katmanından geçmektedir. Bu işlem sonucunda elde edilen görüntüler özellik haritaları olarak adlandırılmaktadır. Bu özellik haritalarına daha sonra havuzlama işlemi uygulanarak maksimum özellikler korunacak şekilde boyutu düşürülmektedir. Bu evrişim ve havuzlama işlemleri girişe verilen görüntüye art arda 4 kez uygulanmakta ve segmentasyonu yapılacak olan nesneyi tanımlayan bilgi elde edilmektedir. Kod çözücü bölümünde ise kodlayıcı bölümünde gerçekleştirilen işlemlerin tam tersi uygulanmaktadır. Bu bölümde boyutu indirgenmiş olan özellik haritalarına, kodlayıcı bölümünden gelen özellik haritaları eklenerek iki kez evrişim işlemi yapılmaktadır. Ardından üst örnekleme yöntemiyle bu görüntülerin boyutları artırılmaktadır. U-Net modelini diğer ESA tabanlı modellerden ayıran en önemli özellik, nesnenin konum bilgisini içeren özellik haritalarının kodlayıcı bölümünden kod çözücü bölümüne doğrudan aktarılmasıdır (Ronneberger ve ark, 2015).

3.5. Pascal U-Net Modeli

Bu çalışmada U-Net modelinden ve Pascal üçgeninden esinlenilerek Pascal U-Net adında yeni bir model geliştirilmiştir. Ayrıca Zhou ve ark.'nın yaptıkları çalışmada

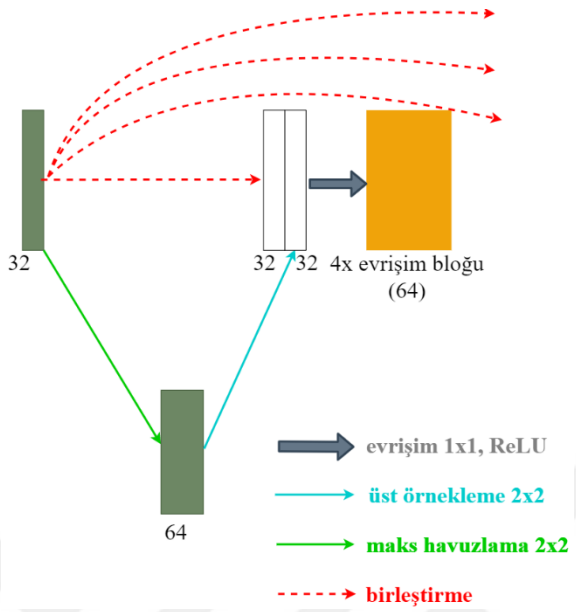
kullandıkları U-Net++ modeli, Pascal U-Net modelinin geliştirilmesine fikir vermiştir (Zhou ve ark, 2018). Pascal U-Net modelini U-Net'ten farklı kılan en önemli özelliği kodlayıcı ve kod çözücü bölümleri arasında sadece doğrudan bağlantı yerine evrişim katmanları eklenerek bir bağlantı oluşturulmuş olmasıdır. U-Net++ modelinde bu bağlantılar birer evrişim bloğu ile gerçekleştirilmiştir. Pascal U-Net modelinde, iki bölüm arasında bulunan bu evrişim katmanları Pascal üçgenindeki sayı dizilimine uygun olarak oluşturulan evrişim bloklarından meydana gelmektedir. Önerilen Pascal U-Net modelinin yapısı Şekil 3.14'te gösterilmiştir.



Şekil 3.14. Pascal U-Net modeli

Şekil 3.14'te görüldüğü gibi Pascal U-Net modelinin tüm katmanları evrişim bloklarından oluşmaktadır. Ağın girişine verilen BT görüntüleri bir kez evrişim uygulandıktan sonra elde edilen özellik haritaları hem iç yapıdaki evrişim bloklarına aktarılmakta, hem doğrudan kod çözücü bölümüne aktarılmakta hem de havuzlama katmanıyla tekrar özellik haritaları çıkarılmaktadır. Bu işlem Şekil 3.15'te gösterilmiştir.

Pascal U-Net modelinde tüm katman blokları arasında birleştirme işlemi uygulanmaktadır.



Şekil 3.15. Pascal U-Net modelinde kodlayıcı bölümü ile kod çözücü bölümü arasındaki işlemler

Bu çalışmada önerilen Pascal U-Net modeline ait katman yapısı Çizelge 3.1'de detaylı olarak verilmiştir. Geleneksel U-Net modelinde parametre sayısı 7 milyon iken Pascal U-Net modelinde parametre sayısı 10 milyondur.

Çizelge 3.1. Pascal U-Net modeli katman yapısı

Katman Sayısı	Katman Adı
1	Giriş Katmanı
2	Evrişim Katmanı
3	Yığın Normalizasyon Katmanı
4	Unutturma Katmanı
5	Havuzlama Katmanı
6	Evrişim Katmanı
7	Yığın Normalizasyon Katmanı
8	Unutturma Katmanı
9	Havuzlama Katmanı
10	Evrişim Katmanı
11	Yığın Normalizasyon Katmanı
12	Unutturma Katmanı
13	Üst Örnekleme Katmanı
14	Yığın Normalizasyon Katmanı
15	Üst Örnekleme Katmanı
16	Unutturma Katmanı
17	Yığın Normalizasyon Katmanı
18	Birleştirme Katmanı
19	Unutturma Katmanı
20	Havuzlama Katmanı
21	Evrişim Katmanı

22	Birleştirme Katmanı
23	Evrişim Katmanı
24	Yığın Normalizasyon Katmanı
25	Evrişim Katmanı
26	Yığın Normalizasyon Katmanı
27	Unutturma Katmanı
28	Yığın Normalizasyon Katmanı
29	Unutturma Katmanı
30	Evrişim Katmanı
31	Unutturma Katmanı
32	Üst Örnekleme Katmanı
33	Yığın Normalizasyon Katmanı
34	Evrişim Katmanı
35	Yığın Normalizasyon Katmanı
36	Unutturma Katmanı
37	Yığın Normalizasyon Katmanı
38	Unutturma Katmanı
39	Evrişim Katmanı
40	Unutturma Katmanı
41	Birleştirme Katmanı
42	Yığın Normalizasyon Katmanı
43	Evrişim Katmanı
44	Evrişim Katmanı
45	Unutturma Katmanı
46	Yığın Normalizasyon Katmanı
47	Yığın Normalizasyon Katmanı
48	Unutturma Katmanı
49	Üst Örnekleme Katmanı
50	Unutturma Katmanı
51	Evrişim Katmanı
52	Yığın Normalizasyon Katmanı
53	Evrişim Katmanı
54	Yığın Normalizasyon Katmanı
55	Unutturma Katmanı
56	Havuzlama Katmanı
57	Yığın Normalizasyon Katmanı
58	Unutturma Katmanı
59	Birleştirme Katmanı
60	Evrişim Katmanı
61	Unutturma Katmanı
62	Birleştirme Katmanı
63	Yığın Normalizasyon Katmanı
64	Üst Örnekleme Katmanı
65	Evrişim Katmanı
66	Unutturma Katmanı
67	Yığın Normalizasyon Katmanı
68	Yığın Normalizasyon Katmanı
69	Üst Örnekleme Katmanı
70	Birleştirme Katmanı
71	Unutturma Katmanı
72	Unutturma Katmanı
73	Yığın Normalizasyon Katmanı
74	Birleştirme Katmanı
75	Evrişim Katmanı
76	Unutturma Katmanı
77	Evrişim Katmanı
78	Yığın Normalizasyon Katmanı
79	Birleştirme Katmanı
80	Yığın Normalizasyon Katmanı
81	Unutturma Katmanı

82	Evrişim Katmanı
83	Unutturma Katmanı
84	Evrişim Katmanı
85	Yığın Normalizasyon Katmanı
86	Evrişim Katmanı
87	Yığın Normalizasyon Katmanı
88	Unutturma Katmanı
89	Yığın Normalizasyon Katmanı
90	Unutturma Katmanı
91	Üst Örnekleme Katmanı
92	Unutturma Katmanı
93	Evrişim Katmanı
94	Yığın Normalizasyon Katmanı
95	Evrişim Katmanı
96	Yığın Normalizasyon Katmanı
97	Unutturma Katmanı
98	Yığın Normalizasyon Katmanı
99	Unutturma Katmanı
100	Birleştirme Katmanı
101	Unutturma Katmanı
102	Evrişim Katmanı
103	Birleştirme Katmanı
104	Yığın Normalizasyon Katmanı
105	Üst Örnekleme Katmanı
106	Evrişim Katmanı
107	Unutturma Katmanı
108	Yığın Normalizasyon Katmanı
109	Yığın Normalizasyon Katmanı
110	Evrişim Katmanı
111	Unutturma Katmanı
112	Unutturma Katmanı
113	Yığın Normalizasyon Katmanı
114	Birleştirme Katmanı
115	Üst Örnekleme Katmanı
116	Unutturma Katmanı
117	Birleştirme Katmanı
118	Yığın Normalizasyon Katmanı
119	Birleştirme Katmanı
120	Unutturma Katmanı
121	Evrişim Katmanı
122	Birleştirme Katmanı
123	Yığın Normalizasyon Katmanı
124	Birleştirme Katmanı
125	Unutturma Katmanı
126	Evrişim Katmanı
127	Evrişim Katmanı
128	Yığın Normalizasyon Katmanı
129	Yığın Normalizasyon Katmanı
130	Unutturma Katmanı
131	Unutturma Katmanı
132	Üst Örnekleme Katmanı
133	Evrişim Katmanı
134	Yığın Normalizasyon Katmanı
135	Yığın Normalizasyon Katmanı
136	Unutturma Katmanı
137	Unutturma Katmanı
138	Birleştirme Katmanı
139	Evrişim Katmanı
140	Birleştirme Katmanı
141	Yığın Normalizasyon Katmanı

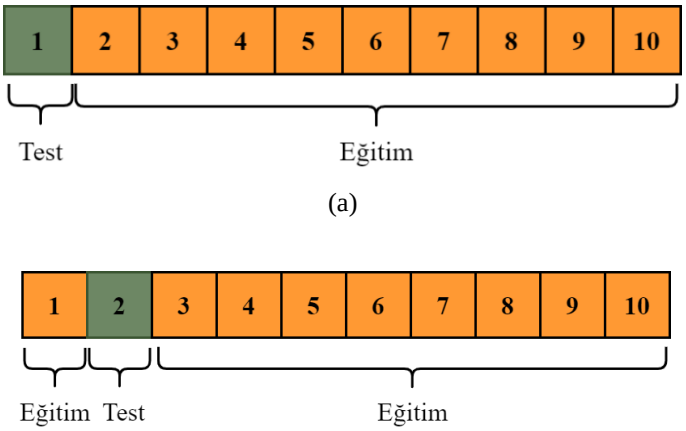
142	Birleştirme Katmanı
143	Unutturma Katmanı
144	Birleştirme Katmanı
145	Evrişim Katmanı
146	Yığın Normalizasyon Katmanı
147	Unutturma Katmanı
148	Evrişim Katmanı
149	Çıkış Katmanı

Toplam Parametre Sayısı: 10,097,025

3.6. K-Katlı Çapraz Doğrulama Yöntemi

Makine öğrenmesinde veri seti, belirlenen oranda eğitim ve test olmak üzere ikiye ayrılabilir. Fakat veri setinin az veri içermesi durumunda, modelin başarımının değerlendirilmesi için bu tip bir ayırma doğru olmamaktadır. Bu sorunu ortadan kaldırmak amacıyla k-katlı çapraz doğrulama yöntemi kullanılmaktadır (Kohavi, 1995). Segmentasyon çalışmalarında derin öğrenme yöntemleri kullanılırken genellikle k-katlı çapraz doğrulama metodu tercih edilmektedir. Çapraz doğrulama, temelde tüm veri setini hem eğitim hem de test verisi olarak kullanan yöntemdir.

K-katlı çapraz doğrulamada tüm veri seti k kadar kesitlere ayrılmaktadır. Birinci iterasyonda k-1 değeri kadar veriler eğitim için kullanılırken 1 adet kesit test için kullanılmaktadır. Bir sonraki katta test için, rastgele başka bir kesit ayrılır ve kalan k-1 değeri kadar olan veriler ise eğitim amacıyla kullanılır. Bu işlem k kadar iterasyon tamamlanıncaya dek devam etmektedir. Örnek olarak 10 katlı çapraz doğrulama işlemi Şekil 3.16'da gösterilmiştir.



Şekil 3.16. 10 katlı çapraz doğrulama (a) 1. kat (b) 2. kat

Bu çalışmada, k-katlı çapraz doğrulama yöntemi sırasıyla k değeri 2,4 ve 6 olacak şekilde uygulanmıştır.

4. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

Bu bölümde, U-Net modeli ve önerilen Pascal U-Net modeli kullanılarak elde edilen pankreas segmentasyon sonuçlar detaylı olarak sunulmuştur. Bu çalışma sonuçları, Python yazılım dilinde 6 GB NVIDIA Geforce GTX 1660 Ti ekran kartına ve Windows 10 işletim sistemine sahip bir iş istasyonunda alınmıştır.

Gerçekleştirilen çalışmada, TCIA'dan ve SÜTFH'den alınmış olan abdomen BT görüntüleri üzerinde hem U-Net modeli ile hem de çalışmada önerilen Pascal U-Net modeli ile pankreas segmentasyonu gerçekleştirilmiş ve sonuçları karşılaştırmalı olarak sunulmuştur.

Her iki modelde de yığın sayıları sırasıyla 1'den 10'a kadar değiştirilmiş ve her yığın sayısı için on defa çalıştırmaya ait sonuçlar alınmıştır. Bir yığın değeri için on kez tekrarlanan sonuçların ortalaması alınarak sonuçlar kaydedilmiştir. Çalışmada; 2, 4 ve 6 katlı çapraz doğrulama yöntemi kullanılmıştır. Tüm çalıştırmalar 500 iterasyon, 2.10^{-4} öğrenme oranı ve Adam optimizatörü ile gerçekleştirilmiş olup veri artırımı (data augmentation) kullanılmamıştır.

4.1. Veri Önişleme

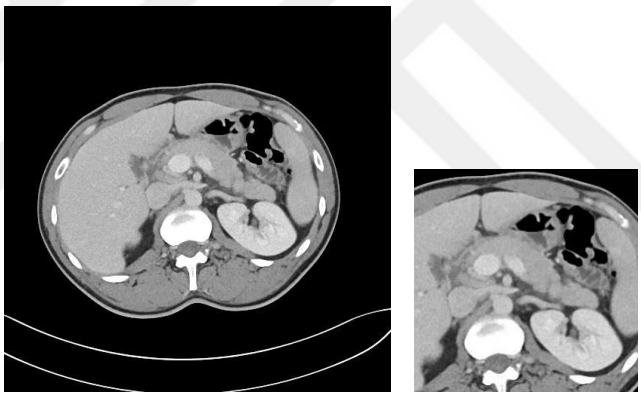
Veri setindeki görüntüler segmente edilmeden önce ilgili alan (region of interest) eldesi ve 2 boyutlu ayrık dalgacık dönüşümü ile boyut azaltımı gerçekleştirilmiştir. Çalışmada kullanılan her iki veri setinde bulunan tüm görüntüler, 512x512 piksel boyutlarından oluşmaktadır. Bu boyutta görüntülerden oluşan bir veri setinde her görüntü için derin öğrenme yöntemlerinin uygulanması hem zaman alacağından hem de işlem yükü açısından bir problem oluşturacağından öncelikle veri setleri üzerinde ön işleme gerçekleştirilmiştir. Bu ön işleme adımları, ilgili alan tespitini ve ayrık dalgacık dönüşümüyle boyut indirgemeyi içermektedir.

4.1.1. İlgili alan eldesi

Bir hastadan alınan abdomene (karın bölgesine) ait BT görüntüleri 100'den fazla kesit içerebilmektedir. Göğüs kafesinin altından başlayarak kalça kemiğine kadar inen bu görüntülerde hastanın vücudunun yer almadığı bölümler mevcuttur. Bu bölgeler siyah bölgeler olarak tanımlanmıştır. Siyah bölgeler, her BT görüntüsü için hasta vücudunun

dışında kaldığından dolayı bu alanlarda pankreası aramak işlem yükünü artırdığı için anlamlı olmayacaktır. Ayrıca pankreas, her insanda farklılık gösteren bir organ olmasına rağmen pankreasın baş bölgesinin başlangıcı hastalar arasında benzerlik göstermektedir. Dolayısıyla, pankreasın olabileceği muhtemel bölgeler tüm görüntüler için belirlenmiş olup ilgili alan bölgeleri tespit edilmiştir. Bu sayede bütün görüntülerde aynı piksel aralıkları seçilerek tüm kesitler 256x256 boyutuna indirgenmiştir. İlgili alan eldesi yöntemi ile işlem yükü azaltılarak zamandan tasarruf edilmiştir.

Şekil 4.1'de görüldüğü üzere pankreasın olabileceği bölgeler belirlenerek 512 x 512 piksel boyutlarında olan görüntülerde yatay ve düşey aynı piksel aralıkları seçilerek ilgili alan belirlenmiştir. Bu sayede pankreasın olmadığı alanlar taranmamış, ayrıca bu işlemle tüm veri setindeki görüntülerin boyutları 256 x 256 piksel olacak şekilde düşürülerek eğitimin daha hızlı olması ve aynı anda daha fazla yığında görüntülerin alınması sağlanmıştır.

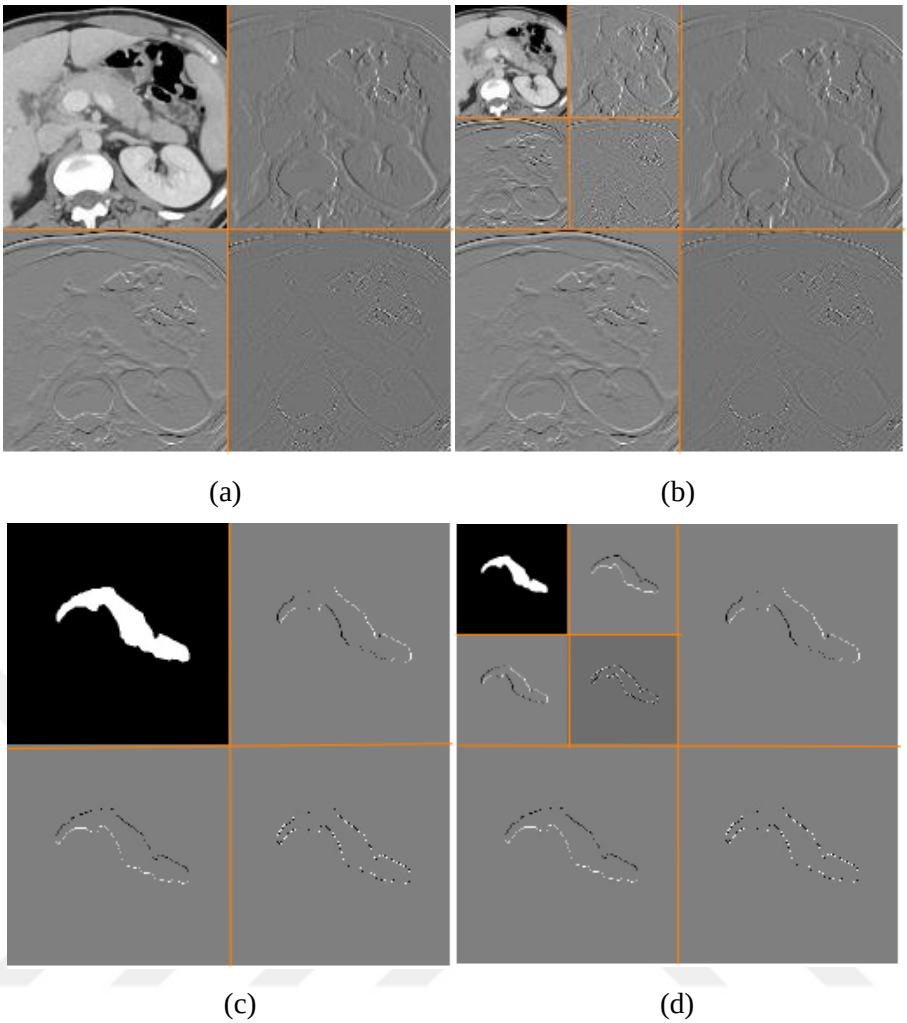


(a) (b)

Şekil 4.1. İlgili alan eldesine bir örnek (a) Ham görüntü (512 x 512 piksel) (b) İlgili alan (256 x 256 piksel)

4.1.2. İki seviyeli 2 boyutlu ADD'nin uygulanması

Her iki veri setinde de ilgili alan eldesinden sonra, 256 x 256 piksel boyutlarındaki görüntülere Daubechies (db2) dalgacık tipi ile iki seviyeli 2 boyutlu ADD uygulanarak görüntülerin boyutları alçak frekans (AA) bilgileri korunarak 64 x 64 piksel haline getirilmiştir. Şekil 4.2 (a)'da ilgili alan eldesi ile oluşturulan görüntünün bir seviyeli 2 boyutlu ADD sonucu, Şekil 4.2 (b)'de bu görüntünün iki seviyeli 2 boyutlu ADD sonucu gösterilmiştir. Şekil 4.2 (c)'de bu görüntüye ait maskenin bir seviyeli 2 boyutlu ADD sonucu ve Şekil 4.2 (d)'de ise iki seviyeli 2 boyutlu ADD sonucu verilmiştir.



(a) (b)

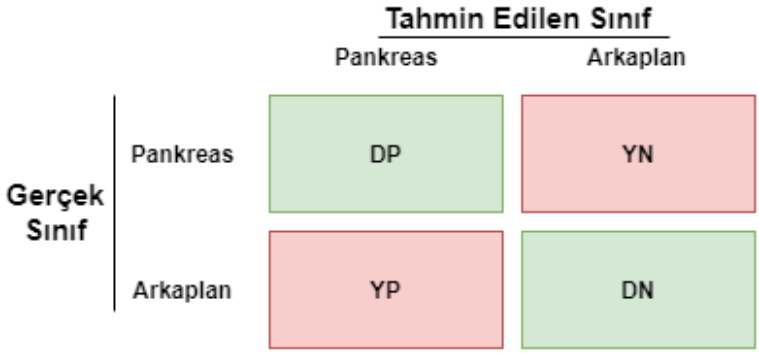
(c) (d)

Şekil 4.2. İlgili alanı elde edilmiş (a) Görüntünün bir seviyeli 2 boyutlu ADD sonucu (b) Görüntünün iki seviyeli 2 boyutlu ADD sonucu (c) Maskenin bir seviyeli 2 boyutlu ADD sonucu (d) Maskenin iki seviyeli 2 boyutlu ADD sonucu

4.2. Performans Değerlendirme Metrikleri

Bu çalışmada, TCIA ve SÜTFH veri setlerinden U-Net modeli ve çalışmada önerilen Pascal U-Net modeli ile pankreas segmentasyonu gerçekleştirilmiştir. Elde edilen segmentasyon sonuçları için performans değerlendirme metrikleri olarak Dice benzerlik katsayısı, Jaccard oranı, duyarlılık, özgüllük, yapısal benzerlik indeksi, doğruluk ve kesinlik metrikleri kullanılmıştır.

Çalışmada kullanılan yapısal benzerlik indeksi metriği dışındaki tüm bu performans değerlendirme metrikleri, bir karmaşıklık matrisinden alınan değerlerle formülize edilen metriklerdir. Hata matrisi olarak da adlandırılan karmaşıklık matrisi, sınıflandırma performansını değerlendirmek için yararlanılan matristir (Stehman, 1997). Şekil 4.3'te bir karmaşıklık matrisi gösterilmiştir.



Şekil 4.3. Bir karmaşıklık matrisi

Şekil 4.3'te yer alan karmaşıklık matrisinde DP, DN, YP ve YN değerleri sırasıyla Doğru Pozitif, Doğru Negatif, Yanlış Pozitif ve Yanlış Negatif değerleridir.

Doğru Pozitif (DP): Gerçekte pankreas olan piksellerden segmentayon sonucunda da pankreas olarak tahmin edilenlerinin sayısıdır.

Doğru Negatif (DN): Gerçekte pankreasa ait olmayan piksellerden segmentasyon sonucunda da arkaplan olarak tahmin edilenlerinin sayısıdır.

Yanlış Poizitif (YP): Gerçekte pankreasa ait olmayan piksellerden segmentasyon sonucunda pankreas olarak tahmin edilenlerinin sayısıdır.

Yanlış Negatif (YN): Gerçekte pankreas olan piksellerden segmentasyon sonucunda arkaplan olarak tahmin edilenlerinin sayısıdır.

4.2.1. Dice benzerlik katsayısı

Dice Benzerlik Katsayısı (DBK, Dice Similarity Coefficient), medikal görüntü segmentasyonu çalışmalarının sonuçlarının analizinde literatürde yaygın şekilde kullanılan bir metriktir (Carass ve ark, 2020). DBK metriği Eşitlik 4.1'de tanımlanmıştır.

$$DBK = \frac{2\,x\,DP}{2\,x\,DP+YP+YN} \tag{4.1}$$

4.2.2. Jaccard oranı

Jaccard Oranı (JO), örneklerin benzerliğini ve çeşitliliğini ölçmek için kullanılan bir istatistiktir. Segmentasyon sonucunda elde edilen görüntü ve hedef görüntü arasındaki JO, bu görüntülerin kesişimlerinin birleşimlerine oranıdır (Chung ve ark, 2019). JO metriği Eşitlik 4.2'de tanımlandığı gibidir.

$$JO = \frac{DP}{DP+YP+YN} \quad (4.2)$$

4.2.3. Duyarlılık

Derin öğrenme modelinin eğitimi ve testi sonrasında elde edilen segmentasyon sonuçlarına göre oluşturulan karmaşıklık matrisinde yer alan, pankreas olarak doğru sınıflandırılan piksellerin sayılarının gerçekte tüm pankreas piksellerinin sayısına oranı duyarlılık (sensitivity) olarak tanımlanır (Yerushalmy, 1947). Doğru pozitif oranı olarak da adlandırılan duyarlılık metriği Eşitlik 4.3 ile ifade edilir.

$$Duyarlılık = \frac{DP}{DP+YN} \quad (4.3)$$

4.2.4. Özgüllük

Derin öğrenme modelinin eğitimi ve testi sonrasında elde edilen segmentasyon sonuçlarına göre oluşturulan karmaşıklık matrisinde yer alan, arkaplan olarak doğru sınıflandırılan piksellerinin sayısının tüm arkaplan piksellerinin sayısına oranı duyarlılık (specificity) olarak tanımlanır (Yerushalmy, 1947). Doğru negatif oranı olarak da adlandırılan özgüllük metriği Eşitlik 4.4 ile ifade edilir.

$$\text{Özgüllük} = \frac{DN}{DN+YP} \quad (4.4)$$

4.2.5. Doğruluk

Derin öğrenme modelinin eğitimi ve testi sonrasında elde edilen segmentasyon sonuçlarına göre oluşturulan karmaşıklık matrisinde yer alan, doğru olarak sınıflandırılan piksellerinin sayısının tüm piksel sayısına oranı doğruluk (accuracy) olarak tanımlanır. Doğruluk metriği Eşitlik 4.5 ile ifade edilir.

$$Doğruluk = \frac{DP+DN}{DP+DN+YP+YN} \quad (4.5)$$

4.2.6. Kesinlik

Kesinlik (precision) metriği, derin öğrenme modelinin eğitimi ve testi sonrasında elde edilen segmentasyon sonuçlarına göre oluşturulan karmaşıklık matrisinde yer alan, pozitif olarak tahmin edilen piksellerden kaç tanesinin doğru pozitif olduğunu ifade eder. Kesinlik metriği Eşitlik 4.6'daki gibi tanımlanır (Fawcett, 2006).

$$Kesinlik = \frac{DP}{DP+YP} \tag{4.6}$$

4.2.7. Yapısal benzerlik indeksi

Yapısal benzerlik indeksi (YBİ, Structural Similarity Index Measure), iki görüntü arasındaki benzerlik kalitesini ölçen, temelde parlaklık terimi, kontrast terimi ve yapısal terim olmak üzere üç parametreden oluşmaktadır (Zhou ve ark, 2004). YBİ metriği Eşitlik 4.7-4.9'da yer alan ifadelerden elde edilen parametreler ile Eşitlik 4.10'da tanımlandığı gibi ifade edilir.

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{4.7}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{4.8}$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{4.9}$$

Eşitlik 4.7-4.9 ifadelerinde yer alan μ_x, μ_y, σ_x, σ_y ve σ_{xy} terimleri, x ve y görüntüleri için sırasıyla yerel ortalamalar, standart sapmalar ve çapraz kovaryans olarak tanımlanmaktadır.

$$YB\dot{I}(x,y) = [l(x,y)]^{\alpha} x [c(x,y)]^{\beta} x [s(x,y)]^{\gamma} \tag{4.10}$$

Eşitlik 4.10 içerisinde yer alan l parlaklık, c kontrast ve s yapısal terimleri karşılamaktadır.

4.3. Pankreas Segmentasyon Sonuçları

Bu çalışmada, abdomen BT görüntülerinden oluşan iki farklı veri seti üzerinde pankreas segmentasyonu gerçekleştirilmiştir. Pankreasın segmentasyonu için hem U-Net modeli ile sonuçlar alınmış, hem de bu çalışmada önerilen Pascal U-Net modeli kullanılmıştır. Farklı yığın sayılarında farklı katlı çapraz doğrulama yöntemleri ile elde edilen bulgular bu bölümde detaylı olarak sunulmuştur.

4.3.1. TCIA veri seti ile elde edilen segmentasyon sonuçları

Çalışmada ilk olarak literatürde sıkça kullanılan bir veri seti olan TCIA veri setindeki abdomen BT görüntülerinden pankreas segmentasyonu gerçekleştirilmiştir. Burada öncelikle abdomen BT görüntülerinden ilgili alan eldesi gerçekleştirilmiş ve 2 seviyeli 2 boyutlu dalgacık transformu ile boyut azaltımı yapılmıştır. Önişleme sonrası elde edilen boyutu azaltılmış ilgili alan görüntülerinden oluşan veri seti üzerinde ilk olarak U-Net modeli ile pankreas segmentasyonu gerçekleştirilmiştir. Segmentasyon sonuçları 7 farklı metrik kullanılarak değerlendirilmiştir. Daha sonra aynı veri seti üzerinde aynı segmentasyon işlemi çalışmada önerilen Pascal U-Net modeli ile gerçekleştirilmiş ve sonuçlar her iki derin öğrenme modeli için de detaylı olarak Çizelge 4.1-4.6'da sunulmuştur. Çizelgelerden de görülebileceği gibi gerçekleştirilen derin öğrenme modellerinin başarımı DBK, JO, duyarlılık, özgüllük, YBİ, doğruluk ve kesinlik metrikleri kullanılarak değerlendirilmiştir. Ayrıca bu çizelgelerde her durum için çapraz doğrulama kullanıldığında algoritmanın ortalama çalışma süresi (eğitim ve test süresi toplamı) da sunulmuştur.

4.3.1.1. U-Net modeli ile elde edilen segmentasyon sonuçları

U-Net modeline ait sırasıyla 1'den 10'a kadar yığın boyutlarında onar kez çalıştırılıp ortalamaları alınarak elde edilen segmentasyon sonuçları DBK, JO, duyarlılık, özgüllük, YBİ, doğruluk ve kesinlik performans metrikleri temel alınarak değerlendirilmiştir. Çapraz doğrulama kullanılarak gerçekleştirilen segmentasyon sonuçları 2, 4 ve 6 kat çapraz doğrulama için Çizelge 4.1-4.3'te sunulmuştur.

Çizelge 4.1. TCIA veri setinde U-Net modeli ile elde edilen pankreas segmentasyon sonuçları (2 katlı çapraz doğrulama için)

Yığın Sayısı	DBK (%)	JO (%)	Duyarlılık (%)	Özgüllük (%)	YBİ (%)	Doğruluk (%)	Kesinlik (%)	Süre (sa.dk.sn)
1	**65.20**	**48.69**	**60.17**	99.00	**86.13**	**95.87**	**73.87**	0.17.31
2	64.37	47.59	58.96	**99.02**	85.71	95.61	73.71	0.09.02
3	58.89	41.96	54.16	98.76	83.34	95.43	66.68	0.05.46
4	56.81	40.11	53.36	98.56	82.32	95.22	62.97	0.05.21
5	55.81	39.31	51.56	98.60	82.43	95.18	62.52	0.06.49
6	52.45	36.02	49.21	98.37	81.25	94.90	57.88	**0.04.02**
7	52.14	35.61	48.88	98.30	81.23	94.80	56.84	0.04.07
8	54.89	37.81	51.30	98.50	82.02	95.05	60.99	0.04.30
9	50.54	34.13	46.24	98.41	80.97	94.79	57.13	0.04.12
10	53.63	37.00	50.50	98.39	81.63	94.96	58.88	0.04.28

Çizelge 4.1'de U-Net modeli ile TCIA veri setinde 1'den 10'a kadar farklı yığın miktarları için 2 katlı çapraz doğrulama ile gerçekleştirilen pankreas segmentasyon sonuçlarında elde edilen en iyi değerler koyu renk ile belirtilmiştir. Çizelgedeki sonuçlar faydalanılan her bir metrik ve yığın değeri için analiz edildiğinde; en iyi DBK, JO, duyarlılık, YBİ, doğruluk ve kesinlik sonuçları olan %65.20, %48.69, %60.17, %86.13, %95.87 ve %73.87 sonuçlarına yığın değeri 1 iken ulaşıldığı görülür. Yalnızca özgüllük metriği bakımından değerlendirildiğinde en iyi sonucun yığın sayısı 2 iken %99.02 olduğu görülür. Çizelgeye göre ortalama çalışma sürelerinin; yığın sayısı 6'dan küçük alındığında, 6 ve daha büyük yığın sayılarına göre daha fazla olduğu görülmektedir. Yığın miktarı 1 olarak alındığında ortalama süre 17 dakika iken, yığın sayısı 6 alındığında 4 dakikadır.

Çizelge 4.2. TCIA veri setinde U-Net modeli ile elde edilen pankreas segmentasyon sonuçları (4 katlı çapraz doğrulama için)

Yığın Sayısı	DBK (%)	JO (%)	Duyarlılık (%)	Özgüllük (%)	YBİ (%)	Doğruluk (%)	Kesinlik (%)	Süre (sa.dk.sn)
1	**68.36**	**52.12**	**64.03**	**99.02**	**87.35**	**96.17**	**75.46**	0.29.53
2	65.30	48.82	60.76	98.93	85.96	95.95	72.81	0.18.13
3	62.47	45.97	57.72	98.86	84.85	95.80	70.57	0.13.22
4	61.07	44.51	57.15	98.74	84.03	95.67	67.88	0.11.46
5	60.17	43.50	56.58	98.67	83.54	95.59	66.25	0.10.29
6	58.34	41.73	53.84	98.70	83.09	95.52	65.72	0.09.48
7	56.65	40.11	52.31	98.62	82.73	95.39	63.73	0.09.09
8	55.90	39.43	52.31	98.50	82.28	95.28	61.57	0.09.05
9	55.00	38.64	51.36	98.46	82.10	95.21	60.44	0.08.35
10	57.24	40.69	53.85	98.53	82.62	95.36	62.73	**0.08.29**

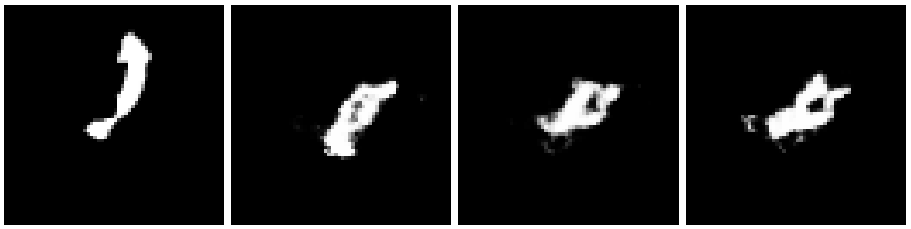
Çizelge 4.2'de önişleme sonrası boyutları azaltılmış ilgili alan görüntülerinden oluşturulan TCIA veri setinde 4 katlı çapraz doğrulama kullanılarak U-Net ile elde edilen pankreas segmentasyon sonuçları sunulmuştur. Çizelge 4.2 tüm yığın sayıları ve her bir metrik için değerlendirildiğinde; en iyi DBK, JO, duyarlılık, özgüllük, YBİ, doğruluk ve kesinlik sonuçlarının yığın sayısı 1 iken %68.36, %52.12, %64.03, %99.02, %87.35, %96.17 ve %75.46 olarak bulunduğu görülür. Çizelgeye göre, 2 katlı çapraz doğrulama ile elde edilen Çizelge 4.1'deki sonuçlardan farklı olarak, yığın sayısı arttıkça derin öğrenme modelinin ortalama çalışma süresi kısalmaktadır. Buna göre; yığın sayısı 1 olduğunda süre 29 dakika iken, yığın sayısı 10 olduğunda bu sürenin 8 dakikaya düştüğü gözlenmiştir.

Çizelge 4.3. TCIA veri setinde U-Net modeli ile elde edilen pankreas segmentasyon sonuçları (6 katlı çapraz doğrulama için)

Yığın Sayısı	DBK (%)	JO (%)	Duyarlılık (%)	Özgüllük (%)	YBİ (%)	Doğruluk (%)	Kesinlik (%)	Süre (sa.dk.sn)
1	**70.24**	**53.99**	**65.96**	**99.08**	**88.08**	**96.29**	**77.21**	0.49.10
2	65.73	49.21	61.44	98.95	86.15	95.81	73.45	0.31.47
3	63.92	47.35	59.82	98.86	85.13	95.87	70.86	0.22.33
4	62.31	45.65	58.05	98.82	84.57	95.77	69.81	0.19.14
5	60.70	44.24	56.86	98.72	83.96	95.64	67.24	0.17.05
6	59.92	43.26	55.78	98.71	83.70	95.59	66.76	0.16.10
7	57.71	41.13	53.55	98.65	82.92	95.46	64.67	0.15.35
8	57.65	40.96	54.01	98.60	82.67	95.44	64.11	0.14.42
9	57.09	40.40	53.44	98.56	82.65	95.37	62.94	0.14.07
10	55.52	39.12	51.93	98.50	82.27	95.26	61.21	**0.13.31**

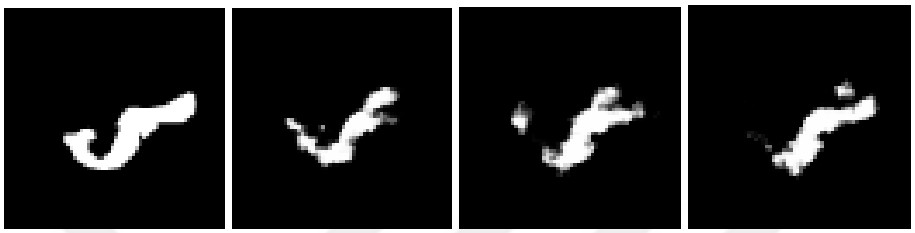
U-Net modeli ile önişleme sonrası oluşturulan TCIA veri setinde 6 katlı çapraz doğrulama kullanılarak elde edilen segmentasyon sonuçları Çizelge 4.3'te sunulmuştur. Kullanılan metrikler ve tüm yığın sayıları açısından en iyi sonuçlar Çizelge 4.3'te koyu renk ile belirtilmiştir. Çizelgeden de görülebileceği üzere; DBK, JO, duyarlılık, özgüllük, YBİ, doğruluk ve kesinlik metrikleri bakımından en iyi sonuçlar yığın sayısı 1 iken %70.24, %53.99, %65.96, %99.08, %88.08, %96.29 ve %77.21 olarak alınmıştır. Yine Çizelge 4.3'e göre yığın sayısı arttıkça ortalama çalışma sürelerinde azalma görülmektedir. Buna göre; yığın sayısı 1 olduğunda çalışma süresinin 49 dakika, 10 olduğunda ise 13 dakika olduğu gözlenmiştir.

Gerçekleştirilen çalışmada, TCIA veri seti üzerinde U-Net modeli kullanılarak 2, 4 ve 6 katlı çapraz doğrulama ile onar kere çalıştırmada elde edilen en düşük ortalama ve en yüksek DBK değerlerine sahip sonuçlar için Şekil 4.4-4.6'da görsel bir karşılaştırma sunulmuştur.



(a) (b) DBK=%27.23 (c) DBK=%44.33 (d) DBK=%42.93

Şekil 4.4. En düşük DBK değerine sahip segmentasyon sonuçları için görsel karşılaştırma (a) Hedef görüntü, (b) 2 katlı çapraz doğrulama sonucu, (c) 4 katlı çapraz doğrulama sonucu, (d) 6 katlı çapraz doğrulama sonucu



(a) (b) DBK=%50.10 (c) DBK=%65.23 (d) DBK=%72.35

Şekil 4.5. Orta DBK değerine sahip segmentasyon sonuçları için görsel karşılaştırma (a) Hedef görüntü, (b) 2 katlı çapraz doğrulama sonucu, (c) 4 katlı çapraz doğrulama sonucu, (d) 6 katlı çapraz doğrulama sonucu

(a) (b) DBK=%90.10 (c) DBK=%93.70 (d) DBK=%92.64

Şekil 4.6. En yüksek DBK değerine sahip segmentasyon sonuçları için görsel karşılaştırma (a) Hedef görüntü, (b) 2 katlı çapraz doğrulama sonucu, (c) 4 katlı çapraz doğrulama sonucu, (d) 6 katlı çapraz doğrulama sonucu

4.3.1.2. Pascal U-Net modeli ile bulunan segmentasyon sonuçları

Sırasıyla 1'den 10'a kadar yığın miktarları için onar defa çalıştırılıp ortalamaları alınarak elde edilen, Pascal U-Net modeline ait segmentasyon sonuçları DBK, JO, duyarlılık, özgüllük, YBİ, doğruluk ve kesinlik performans metrikleri temel alınarak değerlendirilmiştir. Çapraz doğrulama kullanılarak gerçekleştirilen segmentasyon sonuçları 2, 4 ve 6 kat çapraz doğrulama için Çizelge 4.4-4.6'da sunulmuştur.

Çizelge 4.4. TCIA veri setinde Pascal U-Net modeli ile elde edilen pankreas segmentasyon sonuçları (2 katlı çapraz doğrulama için)

Yığın Sayısı	DBK (%)	JO (%)	Duyarlılık (%)	Özgüllük (%)	YBİ (%)	Doğruluk (%)	Kesinlik (%)	Süre (sa.dk.sn)
1	30.00	17.85	21.13	99.15	82.42	94.50	61.31	0.43.59
2	65.35	48.19	56.51	99.36	87.97	96.00	82.65	0.30.36
3	**68.74**	**52.31**	60.40	**99.39**	**88.81**	**96.21**	**83.23**	0.31.01
4	67.92	51.41	**60.95**	99.28	88.04	96.13	80.66	0.30.42
5	67.63	51.33	60.14	99.33	88.23	96.15	81.33	0.29.57
6	65.47	48.64	58.28	99.30	87.42	96.02	80.75	0.26.43
7	64.13	47.12	56.59	99.32	86.76	96.03	81.05	0.25.41
8	63.97	47.26	59.60	99.09	85.44	95.96	76.61	0.25.18
9	63.97	46.74	58.15	99.28	85.40	96.04	80.55	0.25.39
10	62.44	45.69	60.60	98.96	82.42	95.87	74.77	**0.24.39**

Çizelge 4.4'te Pascal U-Net modeli ile TCIA veri setinde 1'den 10'a kadar farklı yığın değerleri ve 2 katlı çapraz doğrulama için gerçekleştirilen pankreas segmentasyon sonuçlarında elde edilen en iyi değerler koyu renk ile belirtilmiştir. Çizelgedeki sonuçlar her bir metrik ve yığın değeri için analiz edildiğinde; en iyi DBK, JO, özgüllük, YBİ, doğruluk ve kesinlik sonuçları olan %68.74, %52.31, %99.39, %88.81, %96.21 ve %83.23 sonuçlarına yığın değeri 3 iken ulaşıldığı görülür. Yalnızca, duyarlılık metriği bakımından değerlendirildiğinde en iyi sonucun yığın sayısı 4 iken %60.95 alındığı görülür. Çizelge 4.4'e göre yığın sayısı arttıkça modellerin ortalama çalışma süreleri azalmaktadır. Buna göre; yığın sayısı 1 iken ortalama 44 dakika süren çalışma süresinin, yığın sayısı 10 olarak alındığında ortalama 24 dakika sürdüğü gözlenmiştir.

Çizelge 4.5. TCIA veri setinde Pascal U-Net modeli ile elde edilen pankreas segmentasyon sonuçları (4 katlı çapraz doğrulama için)

Yığın Sayısı	DBK (%)	JO (%)	Duyarlılık (%)	Özgüllük (%)	YBİ (%)	Doğruluk (%)	Kesinlik (%)	Süre (sa.dk.sn)
1	29.25	17.85	20.62	99.13	82.46	94.60	59.55	1.58.19
2	67.43	51.04	57.47	**99.47**	89.26	96.05	**84.75**	1.16.57
3	69.73	53.93	61.02	99.42	**89.62**	**96.37**	84.03	1.18.12
4	69.80	53.78	61.65	99.38	89.45	96.36	83.30	1.14.33
5	**70.27**	**54.31**	63.14	99.33	89.33	**96.37**	82.39	1.11.40
6	68.96	53.08	61.91	99.29	88.83	96.30	81.22	1.11.07
7	68.53	52.46	60.65	99.37	86.97	96.32	82.94	1.10.23
8	68.29	52.03	60.70	99.35	88.77	96.31	82.78	1.08.26
9	66.76	50.35	59.45	99.32	88.00	96.24	81.89	1.08.31
10	67.12	50.89	**63.25**	99.08	87.07	96.18	77.59	**1.06.06**

Pascal U-Net modeli ile 1'den 10'a kadar farklı yığın sayıları ve 4 katlı çapraz doğrulama için TCIA veri setinde gerçekleştirilen pankreas segmentasyon sonuçlarında, her bir metrik için en iyi değerler Çizelge 4.5'te koyu renk ile belirtilmiştir. Tüm metrikler ve her bir yığın değeri için çizelgedeki sonuçlar incelendiğinde; yığın sayısı 5 iken en iyi

DBK ve JO sonuçlarının %70.27 ve %54.31, en iyi duyarlılık sonucunun yığın sayısı 10 iken %63.25, özgüllük ve kesinlik metriklerine göre en iyi sonuçların yığın sayısı 2 iken %99.47 ve %84.75 olarak alındığı ve yığın sayısı 3 iken en iyi YBİ sonucunun %89.62 olduğu görülür. Ayrıca en iyi doğruluk sonucunun, yığın sayısı hem 3 hem de 5 alındığında %96.37 olarak elde edildiği görülür. Çizelge 4.5'ten de görülebileceği üzere yığın miktarı arttıkça derin öğrenme ağının çalışma süresi azalmaktadır. Buna göre; en uzun ortalama çalışma süresinin yığın sayısı 1 iken 1 saat 58 dakika, en kısa ortalama çalışma süresinin ise yığın sayısı 10 iken 1 saat 6 dakika sürdüğü gözlenmiştir.

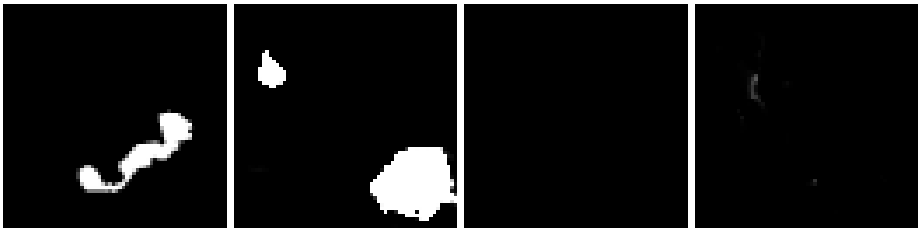
Çizelge 4.6. TCIA veri setinde Pascal U-Net modeli ile elde edilen pankreas segmentasyon sonuçları (6 katlı çapraz doğrulama için)

Yığın Sayısı	DBK (%)	JO (%)	Duyarlılık (%)	Özgüllük (%)	YBİ (%)	Doğruluk (%)	Kesinlik (%)	Süre (sa.dk.sn)
1	29.79	18.99	22.79	98.53	81.79	94.11	56.60	3.13.33
2	68.39	52.40	58.69	**99.48**	89.61	96.07	**84.73**	2.03.37
3	70.73	55.12	62.36	99.43	**89.93**	96.42	84.56	2.08.15
4	70.36	54.55	62.50	99.39	89.70	96.38	83.53	2.02.10
5	**71.35**	**55.73**	**64.15**	99.37	89.75	**96.43**	83.37	1.58.49
6	70.66	54.96	63.83	99.32	89.46	96.38	82.39	1.55.37
7	69.80	53.96	62.78	99.33	89.27	96.36	82.57	1.54.39
8	69.49	53.51	62.51	99.35	89.14	96.37	82.87	1.52.55
9	69.22	53.30	63.01	99.29	88.85	96.32	81.67	1.51.32
10	68.48	52.46	62.20	99.30	88.48	96.32	81.93	**1.49.30**

Pascal U-Net modeli ile önişleme sonrası oluşturulan TCIA veri setinde 6 katlı çapraz doğrulama ve 1'den 10'a kadar farklı yığın değerleri için gerçekleştirilen pankreas segmentasyon sonuçları Çizelge 4.6'da sunulmuştur. Çizelge 4.6'daki sonuçlar her bir metrik ve yığın değeri için değerlendirildiğinde; koyu renk ile belirtilen en iyi DBK, JO, duyarlılık ve doğruluk sonuçları olan %71.35, %55.73, %64.15 ve %96.43 sonuçlarına yığın değeri 5 iken ulaşıldığı görülür. YBİ metriği açısından en iyi değer olan %89.93 sonucunun yığın sayısı 3 iken elde edildiği görülür. Ayrıca, özgüllük ve kesinlik metrikleri bakımından değerlendirildiğinde en iyi sonuçların yığın sayısı 2 iken %99.48 ve %84.73 alındığı görülür. Çizelge 4.6'ya göre yığın sayısı arttıkça modellerin ortalama çalışma süreleri azalmaktadır. Buna göre; yığın sayısı 1 iken ortalama çalışma süresi 3 saat 13 dakika iken, yığın sayısı 10 olarak alındığında ortalama 1 saat 49 dakika sürdüğü gözlenmiştir.

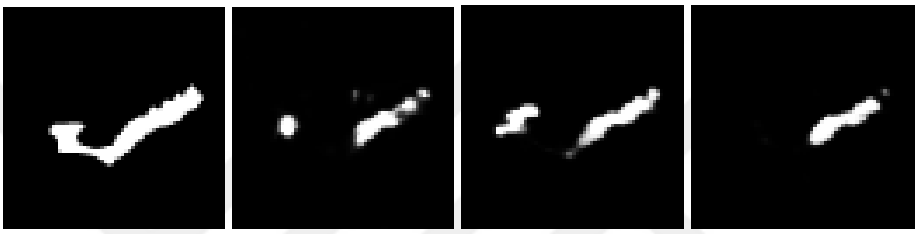
Gerçekleştirilen çalışmada, TCIA veri seti üzerinde Pascal U-Net modeli kullanılarak 2, 4 ve 6 katlı çapraz doğrulama ile onar defa çalıştırmada elde edilen en

düşük ortalama ve en yüksek DBK değerlerine sahip sonuçlar için karşılaştırma Şekil 4.7-4.9'da gösterilmiştir.



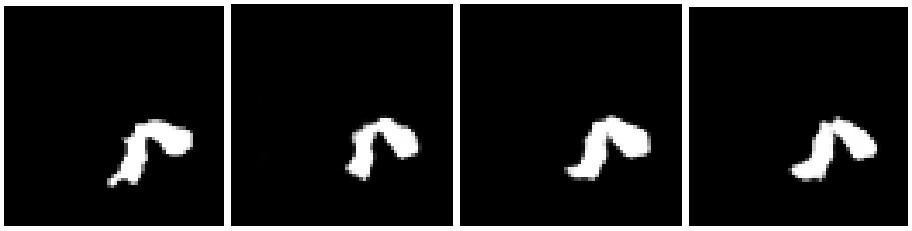
(a) (b) DBK=%0.96 (c) DBK=%0 (d) DBK=%0.92

Şekil 4.7. En düşük DBK değerine sahip segmentasyon sonuçları için görsel karşılaştırma (a) Hedef görüntü, (b) 2 katlı çapraz doğrulama sonucu, (c) 4 katlı çapraz doğrulama sonucu, (d) 6 katlı çapraz doğrulama sonucu



(a) (b) DBK=%53.23 (c) DBK=%56.88 (d) DBK=%52.54

Şekil 4.8. Orta DBK değerine sahip segmentasyon sonuçları için görsel karşılaştırma (a) Hedef görüntü, (b) 2 katlı çapraz doğrulama sonucu, (c) 4 katlı çapraz doğrulama sonucu, (d) 6 katlı çapraz doğrulama sonucu



(a) (b) DBK=%86.56 (c) DBK=%89.55 (d) DBK=%86.56

Şekil 4.9. En yüksek DBK değerine sahip segmentasyon sonuçları için görsel karşılaştırma (a) Hedef görüntü, (b) 2 katlı çapraz doğrulama sonucu, (c) 4 katlı çapraz doğrulama sonucu, (d) 6 katlı çapraz doğrulama sonucu

4.3.2. SÜTFH veri seti ile elde edilen segmentasyon sonuçları

Gerçekleştirilen tez çalışmasında, SÜTFH'den alınan abdomen BT görüntülerinden önişleme uygulanarak oluşturulan veri setinde pankreas segmentasyonu gerçekleştirilmiştir. Bu veri setinde de öncelikle abdomen BT görüntülerinden ilgili alan eldesi gerçekleştirilmiş ve 2 seviyeli 2 boyutlu ADD ile boyut azaltımı yapılmıştır. İlgili alan eldesi ve boyut azaltımı işlemlerinden sonra elde edilen görüntülerden oluşan veri seti üzerinde ilk olarak U-Net modeli ile pankreas segmentasyonu gerçekleştirilmiştir.

Segmentasyon sonuçları 7 farklı metrik kullanılarak değerlendirilmiştir. Daha sonra aynı veri seti üzerinde aynı segmentasyon işlemi çalışmada önerilen Pascal U-Net modeli ile gerçekleştirilmiş ve sonuçlar her iki derin öğrenme modeli için de detaylı olarak Çizelge 4.7-4.12'de sunulmuştur. Çizelgelerden de görülebileceği gibi gerçekleştirilen derin öğrenme modellerinin başarımı DBK, JO, duyarlılık, özgüllük, YBİ, doğruluk ve kesinlik metrikleri kullanılarak değerlendirilmiştir. Ayrıca bu çizelgelerde verilen süre değerleri, eğitim ve test süresi toplamıdır.

4.3.2.1. U-Net modeli ile elde edilen segmentasyon sonuçları

U-Net modelinde SÜTFH veri seti üzerinde sırasıyla 1'den 10'a kadar yığın boyutlarında onar kez çalıştırılıp ortalamaları alınarak elde edilen segmentasyon sonuçları DBK, JO, duyarlılık, özgüllük, YBİ, doğruluk ve kesinlik performans metrikleri temel alınarak değerlendirilmiştir. Çapraz doğrulama kullanılarak gerçekleştirilen segmentasyon sonuçları 2, 4 ve 6 kat çapraz doğrulama için Çizelge 4.7-4.9'da sunulmuştur.

Çizelge 4.7. SÜTFH veri setinde U-Net modeli ile elde edilen pankreas segmentasyon sonuçları (2 katlı çapraz doğrulama için)

Yığın Sayısı	DBK (%)	JO (%)	Duyarlılık (%)	Özgüllük (%)	YBİ (%)	Doğruluk (%)	Kesinlik (%)	Süre (sa.dk.sn)
1	**63.51**	**47.14**	**58.58**	**98.79**	**84.72**	**93.76**	**72.95**	0.08.09
2	60.35	44.26	55.98	98.59	82.87	93.53	68.13	0.06.38
3	58.51	41.75	54.97	98.38	81.71	93.32	64.93	0.04.05
4	54.23	38.22	51.11	98.26	79.96	93.10	61.36	0.03.36
5	53.43	37.46	49.93	98.19	79.89	93.02	60.01	0.03.27
6	53.12	37.11	49.75	98.13	79.66	92.96	59.09	0.03.25
7	52.53	36.71	48.96	98.17	79.62	92.97	58.81	0.03.36
8	51.92	36.22	48.64	98.03	79.25	92.86	57.01	0.03.27
9	54.08	38.20	49.83	98.27	80.54	93.09	61.01	0.03.30
10	51.27	35.20	47.17	98.14	79.21	92.93	57.71	**0.03.21**

Çizelge 4.7'de SÜTFH veri setinde U-Net modelinde 2 katlı çapraz doğrulama ile 1'den 10'a kadar farklı yığın değerleri için gerçekleştirilen pankreas segmentasyon sonuçları sunulmuş ve 7 farklı performans değerlendirme metriği için en iyi sonuçlar koyu renk ile belirtilmiştir. Çizelgeye göre sonuçlar her bir metrik ve yığın değeri için değerlendirildiğinde; en iyi DBK, JO, duyarlılık, doğruluk, özgüllük, YBİ, doğruluk ve kesinlik sonuçları olan %63.51, %47.14, %58.58, %98.79, %84.72, %93.76 ve %72.95

sonuçlarına yığın değeri 1 iken ulaşıldığı görülür. Çizelge 4.7'ye göre yığın sayısı arttıkça modellerin ortalama çalışma süreleri azalmaktadır. Buna göre; yığın sayısı 1 iken ortalama çalışma süresi 8 dakika iken, yığın sayısı 10 olarak alındığında ortalama 3 dakika sürdüğü gözlenmiştir.

Çizelge 4.8. SÜTFH veri setinde U-Net modeli ile elde edilen pankreas segmentasyon sonuçları (4 katlı çapraz doğrulama için)

Yığın Sayısı	DBK (%)	JO (%)	Duyarlılık (%)	Özgüllük (%)	YBİ (%)	Doğruluk (%)	Kesinlik (%)	Süre (sa.dk.sn)
1	**66.96**	**50.89**	**63.48**	**98.85**	**85.66**	**93.90**	**75.42**	0.23.07
2	65.59	49.62	60.81	**98.85**	85.43	**93.90**	74.53	0.15.23
3	62.15	46.24	59.12	98.59	83.52	93.56	69.25	0.10.07
4	60.40	44.38	56.95	98.50	82.68	93.45	67.56	0.09.08
5	59.49	43.59	56.49	98.39	82.09	93.35	65.65	0.08.32
6	57.37	41.75	54.16	98.34	81.24	93.22	63.63	0.08.09
7	56.69	40.88	53.14	98.32	81.07	93.20	63.20	0.08.09
8	57.84	41.82	54.91	98.30	81.39	93.23	63.64	0.07.50
9	55.71	39.97	52.51	98.26	80.73	93.13	61.70	**0.07.44**
10	57.02	40.88	54.03	98.29	81.11	93.19	62.92	0.07.49

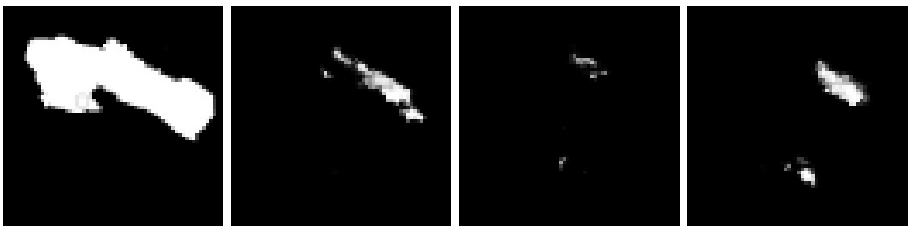
Sırasıyla 1'den 10'a kadar olan farklı yığın sayılarında U-Net modelinde 4 katlı çapraz doğrulama ile gerçekleştirilen pankreas segmentasyonu sonuçları Çizelge 4.8'de sunulmuştur. Çizelgede her bir metrik ve farklı yığın değerlerinde elde edilen en iyi sonuçlar koyu renk ile belirtilmiştir. Buna göre; DBK, JO, duyarlılık, YBİ ve kesinlik metrikleri bakımından en iyi değerler yığın sayısı 1 olduğunda %66.96, %50.89, %63.48, %85.66 ve %75.42 olarak bulunduğu görülür. Özgüllük ve doğruluk metrikleri için değerlendirildiğinde ise yığın sayısı hem 1 hem de 2 alındığında en iyi değerlerin %98.85 ve %93.90 olduğu görülür. Çizelge 4.8'den de görülebileceği üzere yığın sayıları düşük olduğunda ortalama çalışma süreleri, yığın sayılarının büyük alındığı ortalama çalışma sürelerine göre daha uzundur. Yığın sayısı 1 alındığında derin öğrenme modelinin ortalama çalışma süresi 23 dakika iken, yığın sayısı 9 iken ortalama 7 dakika olduğu gözlenmiştir.

Çizelge 4.9. SÜTFH veri setinde U-Net modeli ile elde edilen pankreas segmentasyon sonuçları (6 katlı çapraz doğrulama için)

Yığın Sayısı	DBK (%)	JO (%)	Duyarlılık (%)	Özgüllük (%)	YBİ (%)	Doğruluk (%)	Kesinlik (%)	Süre (sa.dk.sn)
1	**68.05**	**52.01**	**64.00**	**98.94**	**86.68**	**93.99**	**76.95**	0.35.20
2	66.22	50.08	62.05	98.85	85.66	93.91	74.55	0.24.20
3	64.28	48.36	60.86	98.71	84.59	93.73	72.06	0.16.39
4	62.49	46.62	59.25	98.59	83.65	93.58	69.65	0.15.01
5	60.97	45.12	57.69	98.60	82.93	93.53	69.00	0.13.37
6	59.25	43.45	56.68	98.42	81.96	93.36	65.76	0.13.00
7	57.48	41.89	55.04	98.30	81.25	93.21	63.59	**0.12.12**
8	57.48	41.45	54.78	98.32	81.23	93.23	63.78	0.12.32
9	58.66	42.96	55.64	98.41	81.98	93.32	65.12	0.12.39
10	55.98	40.21	52.98	98.25	80.78	93.13	62.03	0.12.17

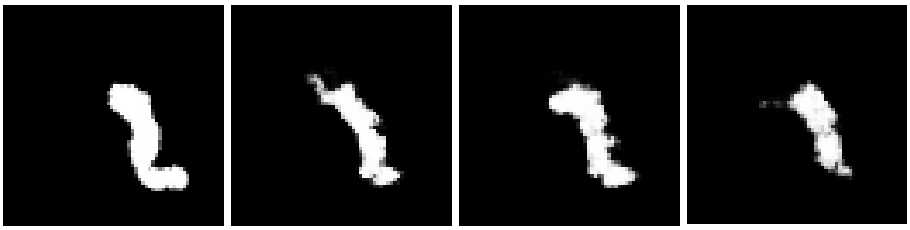
U-Net modeli ile 1'den 10'a kadar farklı yığın değerleri için 6 katlı çapraz doğrulama kullanılarak elde edilen pankreas segmentasyon sonuçları Çizelge 4.9'da sunulmuştur. Çizelge 4.9'da farklı yığın sayıları ve her bir metrik için elde edilen sonuçlar değerlendirildiğinde; DBK, JO, duyarlılık, özgüllük, YBİ, doğruluk ve kesinlik metrikleri bakımından en iyi sonuçların %68.05, %52.01, %64.00, %98.94, %86.68, %93.99 ve %76.95 olarak yığın sayısı 1 iken alındığı görülür. Çizelgeye göre; yığın sayısı 1 ve 2 alındığında derin öğrenme modelinin ortalama çalışma süresinin, yığın sayısı 2 ve daha büyük alındığındaki ortalama çalışma süresine göre daha uzun olduğu görülür. Yığın sayısı 1 iken bu sürenin 35 dakika olduğu, yığın sayısı 7 alındığında ise en kısa süre olan 12 dakika olduğu gözlenmiştir.

Gerçekleştirilen çalışmada, U-Net modeli ile SÜTFH veri seti üzerinde 2, 4 ve 6 katlı çapraz doğrulama ile elde edilen en düşük ortalama ve en yüksek DBK değerlerine sahip sonuçlar için karşılaştırma Şekil 4.10-4.12'de gösterilmiştir.



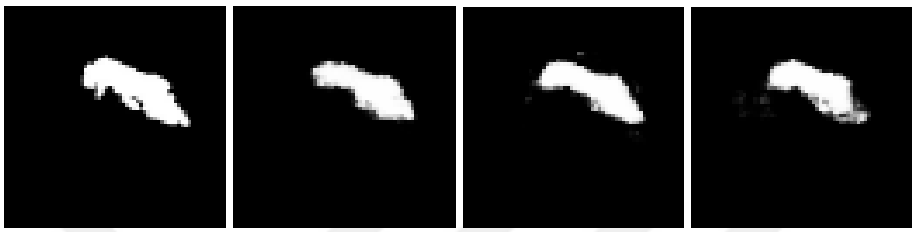
(a) (b) DBK=%21.89 (c) DBK=%4.38 (d) DBK=%16.41

Şekil 4.10. En düşük DBK değerine sahip segmentasyon sonuçları için görsel karşılaştırma (a) Hedef görüntü, (b) 2 katlı çapraz doğrulama sonucu, (c) 4 katlı çapraz doğrulama sonucu, (d) 6 katlı çapraz doğrulama sonucu



(a) (b) DBK=%71.04 (c) DBK=%78.18 (d) DBK=%73.91

Şekil 4.11. Orta DBK değerine sahip segmentasyon sonuçları için görsel karşılaştırma (a) Hedef görüntü, (b) 2 katlı çapraz doğrulama sonucu, (c) 4 katlı çapraz doğrulama sonucu, (d) 6 katlı çapraz doğrulama sonucu



(a) (b) DBK=%90.09 (c) DBK=%89.22 (d) DBK=%88.80

Şekil 4.12. En yüksek DBK değerine sahip segmentasyon sonuçları için görsel karşılaştırma (a) Hedef görüntü, (b) 2 katlı çapraz doğrulama sonucu, (c) 4 katlı çapraz doğrulama sonucu, (d) 6 katlı çapraz doğrulama sonucu

4.3.2.2. Pascal U-Net modeli ile elde edilen segmentasyon sonuçları

SÜTFH veri seti üzerinde sırasıyla 1'den 10'a kadar yığın miktarında Pascal U-Net modeli ile onar defa çalıştırılıp ortalamaları alınarak elde edilen segmentasyon sonuçları DBK, JO, duyarlılık, özgüllük, YBİ, doğruluk ve kesinlik performans metrikleri temel alınarak değerlendirilmiştir. Çapraz doğrulama kullanılarak gerçekleştirilen segmentasyon sonuçları 2, 4 ve 6 kat çapraz doğrulama için Çizelge 4.10-4.12'de sunulmuştur.

Çizelge 4.10. SÜTFH veri setinde Pascal U-Net modeli ile elde edilen pankreas segmentasyon sonuçları (2 katlı çapraz doğrulama için)

Yığın Sayısı	DBK (%)	JO (%)	Duyarlılık (%)	Özgüllük (%)	YBİ (%)	Doğruluk (%)	Kesinlik (%)	Süre (sa.dk.sn)
1	40.58	34.24	49.34	94.17	76.53	89.02	38.86	0.32.01
2	61.47	45.32	52.17	99.20	86.28	93.86	79.40	0.23.26
3	64.53	**48.18**	56.78	99.17	**86.79**	93.94	79.88	0.20.39
4	**64.67**	47.78	56.84	99.24	86.77	93.99	81.15	**0.18.23**
5	63.74	47.19	55.67	99.27	86.35	94.00	81.78	0.19.03
6	63.83	46.73	56.93	99.21	85.36	93.99	80.87	0.18.40
7	62.56	45.03	54.79	**99.28**	84.24	**94.02**	**82.17**	0.18.25
8	62.82	45.34	57.02	99.15	82.49	93.97	80.03	0.18.34
9	61.76	44.33	**58.08**	98.95	79.97	93.84	76.94	0.18.26
10	60.49	43.48	53.88	99.19	81.34	93.90	80.41	0.18.42

Pascal U-Net modeli ile 1'den 10'a kadar farklı yığın değerleri ve 2 katlı çapraz doğrulama için gerçekleştirilen pankreas segmentasyon sonuçları Çizelge 4.10'da sunulmuştur. Çizelge 4.10'daki sonuçlar her bir metrik ve yığın değeri için değerlendirildiğinde; en iyi DBK sonucu %64.67 olarak yığın sayısı 4 iken, en iyi JO ve YBİ sonuçları %48.18 ve %86.79 olarak yığın sayısı 3 iken, en iyi duyarlılık sonucu %58.08 olarak yığın sayısı 9 iken alınmıştır. Aynı zamanda, yığın sayısı 7 iken en iyi özgüllük, doğruluk ve kesinlik sonuçlarının %99.28, %94.02 ve %82.17 olarak alındığı görülür. Çizelgeye göre en kısa ortalama çalışma süresinin yığın sayısı 4 iken 18 dakika, en uzun sürenin ise yığın sayısı 1 iken 32 dakika olduğu gözlenmiştir.

Çizelge 4.11. SÜTFH veri setinde Pascal U-Net modeli ile elde edilen pankreas segmentasyon sonuçları (4 katlı çapraz doğrulama için)

Yığın Sayısı	DBK (%)	JO (%)	Duyarlılık (%)	Özgüllük (%)	YBİ (%)	Doğruluk (%)	Kesinlik (%)	Süre (sa.dk.sn)
1	46.66	37.48	51.51	95.83	80.29	90.61	48.32	1.24.12
2	64.62	48.96	56.74	99.14	87.35	93.93	79.63	0.58.56
3	66.52	50.56	59.48	99.09	87.62	93.92	79.93	0.56.29
4	67.75	51.27	60.62	99.21	**87.96**	94.07	81.61	0.54.38
5	**68.23**	**52.17**	**62.30**	99.13	87.82	94.04	80.36	0.52.18
6	66.74	49.85	59.50	**99.23**	87.43	94.06	81.69	0.50.34
7	67.72	50.91	61.54	99.20	87.38	**94.10**	**81.78**	0.50.15
8	67.19	50.48	61.92	99.10	86.16	94.03	80.21	0.50.17
9	66.15	49.08	60.21	99.16	86.11	94.05	81.20	0.49.29
10	65.05	48.40	61.54	98.97	84.18	93.90	78.07	**0.48.23**

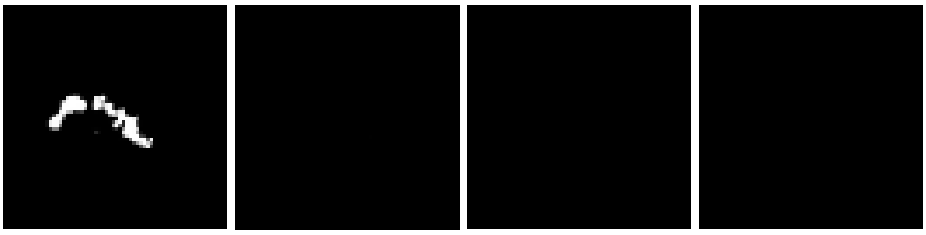
Sırasıyla 1'den 10'a kadar farklı yığın sayıları ve 4 katlı çapraz doğrulama için Pascal U-Net modeli ile gerçekleştirilen segmentasyon sonuçları Çizelge 4.11'de sunulmuştur. Çizelge, her bir metrik için ve farklı yığın sayıları açısından incelendiğinde; DBK, JO ve duyarlılık metrikleri bakımından en iyi sonuçlar olan %68.23, %52.17 ve %62.30 sonuçları yığın sayısı 5 iken, en iyi özgüllük sonucu %99.23 olarak yığın sayısı 6 iken, en iyi YBİ sonucu ise %87.96 olarak yığın sayısı 4 iken alındığında görülür. Ayrıca, doğruluk ve kesinlik metrikleri açısından en iyi sonuçlar olan %94.10 ve %81.78 sonuçların yığın sayısı 7 iken elde edildiği görülür. Yine Çizelge 4.11'den de görülebileceği üzere, yığın sayısı arttıkça derin öğrenme modelinin ortalama çalışma süresi azalmaktadır. Buna göre; en kısa ortalama çalışma süresi yığın sayısı 10 alındığında 48 dakika iken, en uzun ortalama çalışma süresi yığın sayısı 1 alındığında 1 saat 24 dakika olduğu gözlenmiştir.

Çizelge 4.12. SÜTFH veri setinde Pascal U-Net modeli ile elde edilen pankreas segmentasyon sonuçları (6 katlı çapraz doğrulama için)

Yığın Sayısı	DBK (%)	JO (%)	Duyarlılık (%)	Özgüllük (%)	YBİ (%)	Doğruluk (%)	Kesinlik (%)	Süre (sa.dk.sn)
1	49.41	40.32	53.24	96.42	81.04	91.20	51.85	2.14.34
2	65.60	50.19	59.13	98.99	87.56	93.86	78.23	1.34.21
3	66.94	51.59	61.37	98.93	87.56	93.81	79.19	1.33.41
4	68.64	**52.90**	61.20	**99.28**	**88.47**	94.14	**82.98**	1.29.58
5	68.36	52.22	62.43	99.10	87.89	94.02	80.45	1.26.39
6	**68.92**	52.50	62.53	99.20	88.20	94.12	81.70	1.24.02
7	68.34	51.74	61.56	99.26	87.96	**94.15**	82.86	1.23.32
8	67.23	50.83	61.47	99.19	87.17	94.08	81.26	1.22.35
9	67.60	51.00	**62.80**	99.11	86.58	94.06	80.59	1.21.39
10	66.93	49.88	61.36	99.20	86.45	94.10	81.58	**1.20.07**

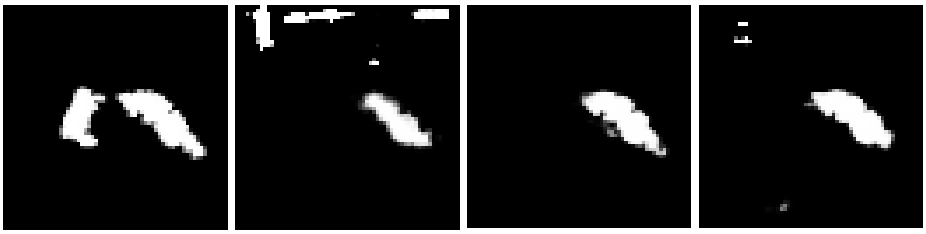
1’den 10’a kadar farklı yığın değerleri için Pascal U-Net modelinde 6 katlı çapraz doğrulama ile gerçekleştirilen pankreas segmentasyon sonuçları Çizelge 4.12’de sunulmuştur. Çizelgedeki sonuçlar farklı yığın değerleri ve her bir metrik için değerlendirildiğinde; en iyi DBK sonucu %68.92 olarak yığın sayısı 6 iken, en iyi JO, özgüllük, YBİ ve kesinlik sonuçları %52.90, %99.28, %88.47 ve %82.98 olarak yığın sayısı 4 iken, en iyi duyarlılık sonucu %62.80 olarak yığın sayısı 9 iken alınmıştır. Ayrıca, yığın sayısı 7 iken en iyi doğruluk sonucunun %94.15 olarak alındığı görülür. Çizelge 4.12’den de görülebileceği üzere, yığın sayısı arttıkça derin öğrenme modelinin ortalama çalışma süresi azalmaktadır. Buna göre; yığın sayısı 10 iken en kısa ortalama çalışma süresinin 1 saat 20 dakika, en uzun sürenin ise yığın sayısı 1 iken 2 saat 14 dakika olduğu gözlenmiştir.

Gerçekleştirilen tez çalışmasında, Pascal U-Net modeli ile SÜTFH veri seti üzerinde 2, 4 ve 6 katlı çapraz doğrulama ile elde edilen en düşük ortalama ve en yüksek DBK değerlerine sahip sonuçlar için karşılaştırma Şekil 4.13-4.15’te gösterilmiştir.



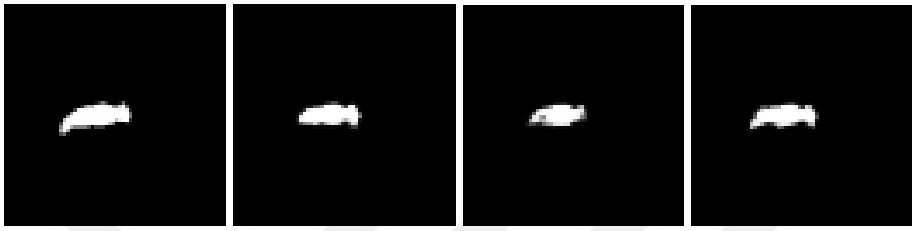
(a) (b) DBK=%0 (c) DBK=%1.74 (d) DBK=%0

Şekil 4.13. En düşük DBK değerine sahip segmentasyon sonuçları için görsel karşılaştırma (a) Hedef görüntü, (b) 2 katlı çapraz doğrulama sonucu, (c) 4 katlı çapraz doğrulama sonucu, (d) 6 katlı çapraz doğrulama sonucu



(a) (b) DBK=%43.90 (c) DBK=%70.87 (d) DBK=%68.85

Şekil 4.14. Orta DBK değerine sahip segmentasyon sonuçları için görsel karşılaştırma (a) Hedef görüntü, (b) 2 katlı çapraz doğrulama sonucu, (c) 4 katlı çapraz doğrulama sonucu, (d) 6 katlı çapraz doğrulama sonucu



(a) (b) DBK=%86.73 (c) DBK=%80.72 (d) DBK=%88.00

Şekil 4.15. En yüksek DBK değerine sahip segmentasyon sonuçları için görsel karşılaştırma (a) Hedef görüntü, (b) 2 katlı çapraz doğrulama sonucu, (c) 4 katlı çapraz doğrulama sonucu, (d) 6 katlı çapraz doğrulama sonucu

5. SONUÇLAR VE ÖNERİLER

5.1 Sonuçlar

Gerçekleştirilen çalışmada biyomedikal görüntü segmentasyonunda önemli bir problem olan pankreas segmentasyonu için yeni bir derin öğrenme modeli önerilmiştir. Önerilen derin öğrenme modelinin başarımı, TCIA veri tabanından alınan Pankreas BT görüntüleri ile SÜTFH Radyoloji Bölümü'nden temin edilen abdomen BT görüntülerini içeren iki adet veri seti kullanılarak değerlendirilmiştir. Her iki veri setinde de her hasta için pankreasın olduğu birer kesit seçilmiştir. Seçilen bu görüntülere her iki veri setinde de öncelikle BT görüntüsü içerisinde pankreasın olmadığı yerler kırpılarak ilgili alan eldesi oluşturulmuştur. Bu işlem sırasında görüntülerin boyutları yarıya indirgenmiştir. Daha sonra, ilgili alanları elde edilen görüntülere iki seviyeli 2 boyutlu ADD uygulanarak hem görüntülerin özellikleri korunmuş hem de boyutlarının azaltımı gerçekleştirilmiştir. Bu önişleme yöntemleri sayesinde, her iki veri setinde de 512 x 512 piksel boyutunda olan ham görüntüler, özellikleri büyük ölçüde korunarak 64 x 64 piksel boyutlarına indirgenmiştir. Bu şekilde TCIA veri seti 82 adet, SÜTFH veri seti ise 56 adet görüntü içerecek şekilde oluşturulmuştur.

Ayrıca sonuçları karşılaştırmalı olarak değerlendirmek amacıyla; literatürde medikal alanda ve ESA konusunda yapılan çalışmalarda son zamanlarda sıklıkla kullanılan U-Net modeli kullanılmış ve her iki veri setinde de pankreas segmentasyon sonuçları alınmıştır. Hem U-Net modeli hem de Pascal üçgeninden esinlenilerek oluşturulan ve bu tez çalışmasında önerilen Pascal U-Net modeli ile her iki veri seti üzerinde gerçekleştirilen segmentasyon sonuçları detaylı olarak analiz edilmiştir.

Derin öğrenme modelleri ile gerçekleştirilen segmentasyon çalışmalarında; 2, 4 ve 6 katlı çapraz doğrulama yöntemleri kullanılmıştır. Yığın sayıları 1'den 10'a kadar alınmış ver her çalıştırma 10 kez tekrarlanmıştır. Bu 10 çalıştırmada elde edilen sonuçların ortalamaları alınarak çalışmada sunulmuştur. Tüm sonuçların hesaplanması, 7 farklı performans değerlendirme metriği kullanılarak elde edilmiştir.

5.1.1 TCIA veri setinde elde edilen pankreas segmentasyon sonuçlarının karşılaştırılması

TCIA veri seti üzerinde farklı yığın sayıları ve farklı kat çapraz doğrulama yöntemi kullanılarak U-Net modeli ile her kat çapraz doğrulama yöntemi için elde edilen optimum pankreas segmentasyonu sonuçlarının DBK metriği temel alınarak yapılan analizi Çizelge 5.1'de verilmiştir.

Çizelge 5.1. TCIA veri setinde U-Net modeli ile elde edilen optimum DBK sonuçları

Çapraz doğrulama	DBK (%)	Yığın Sayısı	Süre (sa.dk.sn)
2 kat	65.20	1	0.17.31
4 kat	68.36	1	0.29.53
6 kat	70.24	1	0.49.10

Çizelge 5.1 incelendiğinde, TCIA veri setinde U-Net modeli kullanılarak bulunan en iyi DBK sonuçlarının; 2 katlı çapraz doğrulamada yığın sayısı 1 iken 17 dakikada %65.20, 4 katlı çapraz doğrulamada yine yığın sayısı 1 iken 29 dakikada %68.36 ve 6 katlı çapraz doğrulamada yığın sayısı 1 iken 49 dakikada %70.24 olduğu görülür. Çizelgeden de görüldüğü gibi çapraz doğrulama kat sayısı arttıkça süre artsa da DBK da artmıştır.

2, 4 ve 6 katlı çapraz doğrulama kullanılarak TCIA veri seti üzerinde Pascal U-Net modeli ile gerçekleştirilen pankreas segmentasyon sonuçlarının en iyi DBK sonuçlarına göre analizi Çizelge 5.2'de verilmiştir.

Çizelge 5.2. TCIA veri setinde Pascal U-Net modeli ile elde edilen optimum DBK sonuçları

Çapraz doğrulama	DBK (%)	Yığın Sayısı	Süre (sa.dk.sn)
2 kat	68.74	3	0.31.01
4 kat	70.27	5	1.11.40
6 kat	71.35	5	1.58.49

Çizelge 5.2'den de görülebileceği üzere, TCIA veri seti üzerinde Pascal U-Net modeli kullanılarak elde edilen en iyi DBK sonuçlarının; 2 katlı çapraz doğrulamada yığın sayısı 3 iken 31 dakikada %68.74, 4 katlı çapraz doğrulamada yığın sayısı 5 iken 1 saat 11 dakikada %70.27 ve 6 katlı çapraz doğrulamada yığın sayısı yine 5 iken 1 saat 58 dakika sonunda %71.35 olduğu görülür.

Çizelge 5.1 ve Çizelge 5.2 birlikte incelendiğinde; 2, 4 ve 6 katlı çapraz doğrulamada kat sayısı arttıkça DBK'nın da arttığı gözlenmiştir. Ayrıca, TCIA veri seti

üzerinde gerçekleştirilen pankreas segmentasyon sonuçlarının Pascal U-Net modeli kullanıldığında U-Net modeline göre daha iyi olduğu görülmüştür.

5.1.2 SÜTFH veri setinde elde edilen pankreas segmentasyon sonuçlarının karşılaştırılması

SÜTFH veri seti üzerinde farklı yığın sayıları ve farklı kat çapraz doğrulama yöntemi kullanılarak U-Net modeli ile elde edilen pankreas segmentasyon sonuçlarının, DBK metriği bakımından optimum değerleri her kat çapraz doğrulama ve yığın sayıları için Çizelge 5.3'te verilmiştir.

Çizelge 5.3. SÜTFH veri setinde U-Net modeli ile elde edilen optimum DBK sonuçları

Çapraz doğrulama	DBK (%)	Yığın Sayısı	Süre (sa.dk.sn)
2 kat	63.51	1	0.08.09
4 kat	66.96	1	0.23.07
6 kat	68.05	1	0.35.20

Çizelge 5.3'te, SÜTFH veri seti üzerinde U-Net modeli kullanılarak elde edilen en iyi DBK sonuçlarının; 2 katlı çapraz doğrulamada yığın sayısı 1 iken 8 dakikada %63.51, 4 katlı çapraz doğrulamada yığın sayısı 1 iken 23 dakikada %66.96 ve 6 katlı çapraz doğrulamada yığın sayısı yine 1 iken 35 dakikada %68.05 olduğu görülür.

2, 4 ve 6 katlı çapraz doğrulama kullanılarak SÜTFH veri seti üzerinde Pascal U-Net modeli ile gerçekleştirilen pankreas segmentasyon sonuçlarının en iyi DBK sonuçlarına göre analizi Çizelge 5.4'te verilmiştir.

Çizelge 5.4. SÜTFH veri setinde Pascal U-Net modeli ile elde edilen optimum DBK sonuçları

Çapraz doğrulama	DBK (%)	Yığın Sayısı	Süre (sa.dk.sn)
2 kat	64.67	4	0.18.23
4 kat	68.23	5	0.52.18
6 kat	68.92	6	1.24.02

Çizelge 5.4 incelendiğinde, Pascal U-Net modeli kullanılarak SÜTFH veri seti üzerinde elde edilen en iyi DBK sonuçlarının; 2 katlı çapraz doğrulamada yığın sayısı 4 iken 18 dakikada %64.67, 4 katlı çapraz doğrulamada yığın sayısı 5 iken 52 dakikada %68.23 ve 6 katlı çapraz doğrulamada yığın sayısı 6 iken 1 saat 24 dakikada %68.92 olduğu görülür.

Çizelge 5.3 ve Çizelge 5.4 birlikte değerlendirildiğinde; 2, 4 ve 6 katlı çapraz doğrulamada SÜTFH veri seti üzerinde gerçekleştirilen pankreas segmentasyon sonuçlarının Pascal U-Net modeli kullanıldığında U-Net modeline göre daha iyi sonuç verdiği gözlenmiştir. Ayrıca TCIA veri seti ile elde edilen sonuçlarda olduğu gibi, SÜTFH veri setinde de her iki modelde çapraz doğrulama kat sayısı arttıkça DBK değerinin de arttığı tespit edilmiştir.

5.1.3 Literatür karşılaştırması

Gerçekleştirilen tez çalışmasında elde edilen en iyi pankreas segmentasyon sonuçları, yapılan kaynak araştırması sonucunda derin öğrenme modelleri ile pankreas segmentasyonu yapılan çalışmalarla karşılaştırmalı olarak Çizelge 5.5'te sunulmuştur.

Çizelge 5.5. Bu tez çalışmasında elde edilen pankreas segmentasyonu sonuçları ile literatür karşılaştırması

Yazar	Veri Sayısı	Model	DBK	JO	Duyarlılık	Özgüllük	YBİ	Doğruluk	Kesinlik
Roth ve ark. (2015)	82	Süperpiksel + ESA	0.68	-	-	-	-	-	-
Farag ve ark. (2016)	80	AlexNet + Süperpiksel + Bölge Etiketleme + Sonişleme	0.71	0.58	0.75	-	-	-	0.72
Zhou ve ark. (2016)	240	Tam ESA	-	0.45	-	-	-	-	-
Roth ve ark. (2018)	82	Rastgele Orman + Bütünsel Sinir Ağı	0.81	0.69	-	-	-	-	-
Roth ve ark. (2018)	82	3B Tam ESA + 4 Katlı Çapraz Doğrulama	0.77	-	-	-	-	-	-
Roth ve ark. (2018)	150	3B TEA	0.82	-	-	-	-	-	-
Gibson ve ark. (2018)	90	Yoğun V-Ağı	0.78	-	-	-	-	-	-
Yang ve ark. (2019)	82	TEA + TSA	**0.88**	-	-	-	-	-	-
Man ve ark. (2019)	82	Derin Q Ağı	0.87	-	0.78	-	-	-	-
Lu ve ark. (2019)	82	Artık U-Net + 10 Katlı Çapraz Doğrulama	**0.88**	-	-	-	-	-	-
Liu ve ark. (2019)	82	Süperpiksel + 5 farklı U-Net modeli + 4 Katlı Çapraz Doğrulama	0.84	**0.73**	**0.85**	-	-	-	**0.84**

Bu tez çalışması	56	Pascal U-Net + 6 Kat Çapraz Doğrulama	0.69	0.53	0.63	**0.99**	0.88	0.94	0.82
Bu tez çalışması	82	Pascal U-Net + 6 Kat Çapraz Doğrulama	0.71	0.56	0.64	**0.99**	**0.90**	**0.96**	0.83

Bu tez çalışmasında gerçekleştirilen pankreas segmentasyon sonuçlarına göre elde edilen en iyi sonuçların 7 farklı metrik bakımından literatür ile kıyaslanması Çizelge 5.5'te sunulmuş ve en iyi değerler koyu renk ile belirtilmiştir. Çizelge 5.5'e göre; DBK metriği bakımından en iyi sonuç %88 olarak Lu ve ark. tarafından, JO, duyarlılık ve kesinlik metrikleri bakımından en iyi sonuçların %73, %85 ve %84 olarak Liu ve ark. tarafından alındığı görülür. Literatürde yer alan diğer derin öğrenme yöntemleri ile gerçekleştirilen pankreas segmentasyon çalışmalarında DBK, JO, duyarlılık ve kesinlik metrikleri için elde edilen değerlerin bu tez çalışmasına göre daha yüksek olmasının temel nedeni, kullanılan iş istasyonunda bulunan ekran kartının işlem sınırından kaynaklanmaktadır. Yapılan kaynak araştırmasında, derin öğrenme yöntemleri ile pankreas segmentasyonu gerçekleştirilen çalışmaların özgüllük, YBİ ve doğruluk metrikleri için sonuç verilmemiştir. Buna göre Çizelge 5.5'te; özgüllük, YBİ ve doğruluk performans değerlendirme metrikleri bakımından en iyi sonuçların %99, %90 ve %96 olarak bu tez çalışmasında elde edildiği görülür.

5.2 Öneriler

Gerçekleştirilen tez çalışmasında önerilen Pascal U-Net modelinin, tezde kullanılan her iki veri setinde de geleneksel U-Net modelinden daha iyi sonuçlar verdiği gözlenmiştir. Yapılan kaynak araştırması ile karşılaştırıldığında elde edilen sonuçların DBK ve JO kriterleri açısından literatüre göre düşük kaldığı gözlenmiştir.

- Önerilen Pascal U-Net modelinde kodlayıcı ve kod çözücü bölümler arasında yer alan evrişim blokları yerine yoğun (dense) bloklar kullanılırsa modelin parametre sayısı artar. Böylece daha uzun sürede fakat daha iyi sonuç verebilir.
- Bu çalışmada 2,4 ve 6 kat çapraz doğrulama yöntemleri ve 1'den 10'a kadar yığın sayılarında görüntüler alınarak pankreas segmentasyonu gerçekleştirilmiştir. İleride daha fazla katlı çapraz doğrulama yöntemleri

ve daha fazla yığın miktarlarında Pascal U-Net modeli ile sonuçlar alınabilir.

- Kullanılan iş istasyonunun ekran kartı kapasitesi sınırlı olduğundan dolayı çalışmada kullanılan veri setlerinde yer alan görüntü sayıları her hasta için bir kesit olacak şekilde ayarlanmıştır. Daha iyi ekran kartına sahip iş istasyonlarında veri artırım yöntemleri veya Çekişmeli Üretici Ağlar kullanılarak elde edilebilecek daha fazla veri sayısı ile sonuçlar alınabilir.

EKLER

EK-1 Tez Çalışmasında Kullanılan SÜTFH Görüntüleri için Etik Kurul Raporu

T.C.
SELÇUK ÜNİVERSİTESİ
TIP FAKÜLTESİ DEKANLIĞI

YEREL ETİK KURULU KARARLARI

Toplantı Sayısı: 2020/03	Toplantı Tarihi : 05.02.2020

Karar Sayısı 2020/58 Konya Teknik Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü öğretim üyesi Doç.Dr.Rahime CEYLAN'ın "Derin Öğrenme Algoritmaları İle Abdomen BT ve MR Görüntülerinde Pankreas Segmentasyonu ve Karakterizasyonu" başlıklı araştırmasının değerlendirilme talebi ile ilgili 10.01.2020 tarihli dilekçesi ve ekleri görüşüldü.

Yapılan inceleme ve görüşmelerden sonra; Doç.Dr.Rahime CEYLAN'ın "Derin Öğrenme Algoritmaları İle Abdomen BT ve MR Görüntülerinde Pankreas Segmentasyonu ve Karakterizasyonu" adlı araştırmasının kabulüne oy birliği ile karar verildi.
Yardımcı Araştırmacılar: Hakan CEBECİ, Mustafa KOPLAY.

ASLI GİBİDİR
05.02.2020

S.Selcen ÇELİK
Sekreterya

EK-2 Önerilen Pascal U-Net Modeli Python Kodu

```
import tensorflow as tf
from tensorflow import keras

def fibov4(pretrained_weights = None,input_size = (64,64,1)):
    inputs = keras.Input(input_size)
    c1 = keras.layers.Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(inputs)
    c1 = keras.layers.BatchNormalization()(c1)
    c1 = keras.layers.Dropout(0.2)(c1)

    p1 = keras.layers.MaxPool2D(pool_size=(2, 2))(c1)
    c2 = keras.layers.Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(p1)
    c2 = keras.layers.BatchNormalization()(c2)
    c2 = keras.layers.Dropout(0.2)(c2)

    p2 = keras.layers.MaxPool2D(pool_size=(2, 2))(c2)
    c3 = keras.layers.Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(p2)
    c3 = keras.layers.BatchNormalization()(c3)
    c3 = keras.layers.Dropout(0.2)(c3)

    p3 = keras.layers.MaxPool2D(pool_size=(2, 2))(c3)
    c4 = keras.layers.Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(p3)
    c4 = keras.layers.BatchNormalization()(c4)
    c4 = keras.layers.Dropout(0.3)(c4)

    p4 = keras.layers.MaxPool2D(pool_size=(2, 2))(c4)
    c5 = keras.layers.Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(p4)
    c5 = keras.layers.BatchNormalization()(c5)
    c5 = keras.layers.Dropout(0.4)(c5)

    u1 = keras.layers.Conv2D(256, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')
    u1 = keras.layers.UpSampling2D(size = (2,2))(c5)
    u1 = keras.layers.BatchNormalization()(u1)
```

```
        u1 = keras.layers.Dropout(0.2)(u1)
        m1 = keras.layers.concatenate([c4,u1], axis = 3)
        c6 = keras.layers.Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(m1)  #512
        c6 = keras.layers.BatchNormalization()(c6)
        c6 = keras.layers.Dropout(0.2)(c6)
        u5 = keras.layers.Conv2D(128, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')
        u5 = keras.layers.UpSampling2D(size = (2,2))(c4)
        u5 = keras.layers.BatchNormalization()(u5)
        u5 = keras.layers.Dropout(0.2)(u5)
        m2 = keras.layers.concatenate([c3,u5], axis = 3)
        ########################## 2'Li Yapı ####################
        ##  D1
        cd1 = keras.layers.Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(m2)
        cd1 = keras.layers.BatchNormalization()(cd1)
        cd1 = keras.layers.Dropout(0.2)(cd1)

        d1 = keras.layers.Conv2D(256, 1, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(cd1)
        d1 = keras.layers.BatchNormalization()(d1)
        d1 = keras.layers.Dropout(0.2)(d1)

        u2 = keras.layers.Conv2D(128, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')
        u2 = keras.layers.UpSampling2D(size = (2,2))(c6)
        u2 = keras.layers.BatchNormalization()(u2)
        u2 = keras.layers.Dropout(0.2)(u2)
        m3 = keras.layers.concatenate([c3,u2], axis = 3)
        m4 = keras.layers.concatenate([m3,d1], axis = 3)
        c7 = keras.layers.Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(m4)  #512
        c7 = keras.layers.BatchNormalization()(c7)
        c7 = keras.layers.Dropout(0.2)(c7)

        u6 = keras.layers.Conv2D(64, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')
        u6 = keras.layers.UpSampling2D(size = (2,2))(c3)
        u6 = keras.layers.BatchNormalization()(u6)
```

```
        u6 = keras.layers.Dropout(0.2)(u6)
        m5 = keras.layers.concatenate([u6,c2], axis = 3)
        ################  3'lü birinci yapı ###################
        ##  D3
        cd3 = keras.layers.Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer =
'he_normal')(m5)
        cd3 = keras.layers.BatchNormalization()(cd3)
        cd3 = keras.layers.Dropout(0.2)(cd3)
        cd32 = keras.layers.Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer =
'he_normal')(cd3)
        cd32 = keras.layers.BatchNormalization()(cd32)
        cd32 = keras.layers.Dropout(0.2)(cd32)

        d3 = keras.layers.Conv2D(64, 1, activation = 'relu', padding = 'same', kernel_initializer =
'he_normal')(cd32)
        d3 = keras.layers.BatchNormalization()(d3)
        d3 = keras.layers.Dropout(0.2)(d3)
        ############################################
        m6 = keras.layers.concatenate([c2,d3], axis = 3)
        ud1 = keras.layers.Conv2D(128, 2, activation = 'relu', padding = 'same', kernel_initializer =
'he_normal')
        ud1 = keras.layers.UpSampling2D(size = (2,2))(d1)
        ud1 = keras.layers.BatchNormalization()(ud1)
        ud1 = keras.layers.Dropout(0.2)(ud1)
        m7 = keras.layers.concatenate([m6,ud1], axis = 3)
          ###################  3'lü ikinci yapı ###################
        ##  D6
        cd6 = keras.layers.Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer =
'he_normal')(m7)
        cd6 = keras.layers.BatchNormalization()(cd6)
        cd6 = keras.layers.Dropout(0.2)(cd6)
        cd62 = keras.layers.Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer =
'he_normal')(cd6)
        cd62 = keras.layers.BatchNormalization()(cd62)
        cd62 = keras.layers.Dropout(0.2)(cd62)

        d6 = keras.layers.Conv2D(64, 1, activation = 'relu', padding = 'same', kernel_initializer =
'he_normal')(cd62)
        d6 = keras.layers.BatchNormalization()(d6)
        d6 = keras.layers.Dropout(0.2)(d6)
```

```
            ############################
            u3 = keras.layers.Conv2D(64, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')
            u3 = keras.layers.UpSampling2D(size = (2,2))(c7)
            u3 = keras.layers.BatchNormalization()(u3)
            u3 = keras.layers.Dropout(0.2)(u3)
            m8 = keras.layers.concatenate([u3,d6], axis = 3)
            m9 = keras.layers.concatenate([m8,m6], axis = 3)
            c8 = keras.layers.Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(m9)  #256
            c8 = keras.layers.BatchNormalization()(c8)
            c8 = keras.layers.Dropout(0.2)(c8)
            u4 = keras.layers.Conv2D(32, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')
            u4 = keras.layers.UpSampling2D(size = (2,2))(c8)
            u4 = keras.layers.BatchNormalization()(u4)
            u4 = keras.layers.Dropout(0.2)(u4)
            u7 = keras.layers.Conv2D(32, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')
            u7 = keras.layers.UpSampling2D(size = (2,2))(c2)
            u7 = keras.layers.BatchNormalization()(u7)
            u7 = keras.layers.Dropout(0.2)(u7)
            m10 = keras.layers.concatenate([c1,u7], axis = 3)
            ud2 = keras.layers.Conv2D(32, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')
            ud2 = keras.layers.UpSampling2D(size = (2,2))(d3)
            ud2 = keras.layers.BatchNormalization()(ud2)
            ud2 = keras.layers.Dropout(0.2)(ud2)
            ud3 = keras.layers.Conv2D(32, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')
            ud3 = keras.layers.UpSampling2D(size = (2,2))(d6)
            ud3 = keras.layers.BatchNormalization()(ud3)
            ud3 = keras.layers.Dropout(0.2)(ud3)
            ###################   4'lü birinci yapı ######################
            ##   D9
            cd9 = keras.layers.Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(m10)
            cd9 = keras.layers.BatchNormalization()(cd9)
            cd9 = keras.layers.Dropout(0.2)(cd9)
```

```
cd92 = keras.layers.Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(cd9)
cd92 = keras.layers.BatchNormalization()(cd92)
cd92 = keras.layers.Dropout(0.2)(cd92)
cd93 = keras.layers.Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(cd92)
cd93 = keras.layers.BatchNormalization()(cd93)
cd93 = keras.layers.Dropout(0.2)(cd93)

d9 = keras.layers.Conv2D(64, 1, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(cd93)
d9 = keras.layers.BatchNormalization()(d9)
d9 = keras.layers.Dropout(0.2)(d9)
##########################
m11 = keras.layers.concatenate([c1,ud2], axis = 3)
m12 = keras.layers.concatenate([m11,d9], axis = 3)
#################  6'lı yapı ####################
##  D13
cd13 = keras.layers.Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(m12)
cd13 = keras.layers.BatchNormalization()(cd13)
cd13 = keras.layers.Dropout(0.2)(cd13)
cd132 = keras.layers.Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(cd13)
cd132 = keras.layers.BatchNormalization()(cd132)
cd132 = keras.layers.Dropout(0.2)(cd132)
cd133 = keras.layers.Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(cd132)
cd133 = keras.layers.BatchNormalization()(cd133)
cd133 = keras.layers.Dropout(0.2)(cd133)
cd134 = keras.layers.Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(cd133)
cd134 = keras.layers.BatchNormalization()(cd134)
cd134 = keras.layers.Dropout(0.2)(cd134)
cd135 = keras.layers.Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(cd134)
cd135 = keras.layers.BatchNormalization()(cd135)
cd135 = keras.layers.Dropout(0.2)(cd135)
```

```
            d13 = keras.layers.Conv2D(128, 1, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(cd135)
            d13 = keras.layers.BatchNormalization()(d13)
            d13 = keras.layers.Dropout(0.2)(d13)
            ################################
            m13 = keras.layers.concatenate([c1,ud3], axis = 3)
            m14 = keras.layers.concatenate([m13,d9], axis = 3)
            m15 = keras.layers.concatenate([m14,d13], axis = 3)
            ###################   4'lü ikinci yapı ######################

            cd19 = keras.layers.Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(m15)
            cd19 = keras.layers.BatchNormalization()(cd19)
            cd19 = keras.layers.Dropout(0.2)(cd19)
            cd192 = keras.layers.Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(cd19)
            cd192 = keras.layers.BatchNormalization()(cd192)
            cd192 = keras.layers.Dropout(0.2)(cd192)
            cd193 = keras.layers.Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(cd192)
            cd193 = keras.layers.BatchNormalization()(cd193)
            cd193 = keras.layers.Dropout(0.2)(cd193)

            d19 = keras.layers.Conv2D(256, 1, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(cd193)
            d19 = keras.layers.BatchNormalization()(d19)
            d19 = keras.layers.Dropout(0.2)(d19)
            ########################################
            m16 = keras.layers.concatenate([c1,u4], axis = 3)
            m17 = keras.layers.concatenate([m16,d9], axis = 3)
            m18 = keras.layers.concatenate([m17,d13], axis = 3)
            m19 = keras.layers.concatenate([m18,d19], axis = 3)
            c9 = keras.layers.Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(m19)    ##512
            c9 = keras.layers.BatchNormalization()(c9)
            c9 = keras.layers.Dropout(0.2)(c9)
            c10 = keras.layers.Conv2D(1, 1, activation = 'sigmoid')(c9)

            model = keras.Model(inputs = inputs, outputs = c10)
           #  keras.optimizers.Adam(learning_rate=0.005)
```

```
            model.compile(optimizer = "Adam", loss = 'binary_crossentropy', metrics = ['accuracy',tf.keras.metrics.MeanIoU(num_classes=2)])

        #   model.summary()

            if(pretrained_weights):
                model.load_weights(pretrained_weights)

    return model
```

EK-3 Veri Setlerinde Yer Alan Görüntülerin Numpy Uzantılı Hale Getiren Python Kodu

```
from __future__ import print_function
import os
import numpy as np
from skimage.io import imsave, imread
data_path = 'pancreas/'
image_rows = 64
image_cols = 64

def create_train_data():
    image_data_path = os.path.join(data_path, 'images')
    mask_data_path = os.path.join(data_path, 'mask')
    images = os.listdir(image_data_path)
    total_img = len(images)
    masks = os.listdir(mask_data_path)
    total_mask = len(masks)
    imgs = np.ndarray((total_img, image_rows, image_cols), dtype=np.uint8)
    imgs_mask = np.ndarray((total_mask, image_rows, image_cols), dtype=np.uint8)
    i = 0
    print('-'*30)
    print('Creating images...')
    print('-'*30)
    for image_name in images:
        img = imread(os.path.join(image_data_path, image_name), as_gray=True)
        img = np.array([img])
        imgs[i] = img
        if i % 10 == 0:
            print('Done')
        i += 1
    print('Loading done.')

    j = 0
    print('-'*30)
    print('Creating masks...')
    print('-'*30)
    for mask_name in masks:
        img_mask = imread(os.path.join(mask_data_path, mask_name), as_gray=True)
        img_mask = np.array([img_mask])
        imgs_mask[j] = img_mask
```

```
            if j % 10 == 0:
                print('Done’)
            j += 1
        print('Loading done.')

        np.save('imgs.npy', imgs)
        np.save('imgs_mask.npy', imgs_mask)
        print('Saving to .npy files done.')

    def load_train_data():
        imgs = np.load('imgs.npy')
        imgs_mask = np.load('imgs_mask.npy')
        return imgs, imgs_mask

    if __name__ == '__main__':
        create_train_data()
#create_test_data()
```

EK-4 Pankreas Segmentasyonu Python Kodu

```
from __future__ import print_function

import tensorflow
import os
from skimage.transform import resize
from skimage.io import imsave
import numpy as np
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, concatenate, Conv2D, MaxPooling2D, Conv2DTranspose
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras import backend as K
from sklearn.model_selection import KFold
from dataload import load_train_data
from fibo32_v4 import *
import time
from datetime import timedelta

K.set_image_data_format('channels_last')  # TF dimension ordering in this code

gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
  try:
    # Restrict TensorFlow to only use the fourth GPU
    tf.config.experimental.set_visible_devices(gpus[0], 'GPU')

    # Currently, memory growth needs to be the same across GPUs
    for gpu in gpus:
      tf.config.experimental.set_memory_growth(gpu, True)
    logical_gpus = tf.config.experimental.list_logical_devices('GPU')
    print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
  except RuntimeError as e:
    # Memory growth must be set before GPUs have been initialized
    print(e)

img_rows = 64
img_cols = 64
```

```
smooth = 1.

batch = 8
kat = 6
sabit = "8,6 sutf fibo -10"

yol_dir = '%s'%sabit
if not os.path.exists(yol_dir):
        os.mkdir(yol_dir)

start_time=time.time()
def dice_coef(y_true, y_pred):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)

def dice_coef_loss(y_true, y_pred):
    return -dice_coef(y_true, y_pred)

def jaccard_coef(y_true, y_pred):
    smooth = 1
    intersection = K.sum(K.abs(y_true * y_pred), axis=[1, 2, 3])
    union = K.sum(y_true, [1,2,3]) +K.sum(y_pred,[1,2,3])-intersection
    jac = K.mean((intersection + smooth) / (union + smooth), axis=0)
    return jac

def SSIM(y_true, y_pred):
  return tf.reduce_mean(tf.image.ssim(y_true, y_pred, 2.0))

def sensitivity(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    return true_positives / (possible_positives + K.epsilon())

def specificity(y_true, y_pred):
    true_negatives = K.sum(K.round(K.clip((1-y_true) * (1-y_pred), 0, 1)))
    possible_negatives = K.sum(K.round(K.clip(1-y_true, 0, 1)))
    return true_negatives / (possible_negatives + K.epsilon())
```

```
def calculate_sensitivity_specificity(y_true, y_pred):
    # Note: More parameters are defined than necessary.
    # This would allow return of other measures other than sensitivity and specificity

    # Get true/false for whether a breach actually occurred
    actual_pos = y_true== 1
    actual_neg = y_true == 0

    # Get true and false test (true test match actual, false tests differ from actual)
    true_pos = (y_pred == 1) & (actual_pos)
    false_pos = (y_pred == 1) & (actual_neg)
    true_neg = (y_pred == 0) & (actual_neg)
    false_neg = (y_pred == 0) & (actual_pos)

    # Calculate accuracy
    accuracy = np.mean(y_pred == y_true)

    # Calculate sensitivity and specificity
    sensitivity = np.sum(true_pos) / np.sum(actual_pos)
    specificity = np.sum(true_neg) / np.sum(actual_neg)

    return sensitivity, specificity

def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

```
def get_unet():
    inputs = Input((img_rows, img_cols, 1))
    conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(inputs)
    conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

    conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(pool1)
    conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

    conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(pool2)
    conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)

    conv4 = Conv2D(256, (3, 3), activation='relu', padding='same')(pool3)
    conv4 = Conv2D(256, (3, 3), activation='relu', padding='same')(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)

    conv5 = Conv2D(512, (3, 3), activation='relu', padding='same')(pool4)
    conv5 = Conv2D(512, (3, 3), activation='relu', padding='same')(conv5)

    up6 = concatenate([Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same')(conv5),
conv4], axis=3)
    conv6 = Conv2D(256, (3, 3), activation='relu', padding='same')(up6)
    conv6 = Conv2D(256, (3, 3), activation='relu', padding='same')(conv6)

    up7 = concatenate([Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(conv6),
conv3], axis=3)
    conv7 = Conv2D(128, (3, 3), activation='relu', padding='same')(up7)
    conv7 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv7)

    up8 = concatenate([Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(conv7),
conv2], axis=3)
    conv8 = Conv2D(64, (3, 3), activation='relu', padding='same')(up8)
    conv8 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv8)

    up9 = concatenate([Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(conv8),
conv1], axis=3)
    conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(up9)
    conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv9)
```

```
    conv10 = Conv2D(1, (1, 1), activation='sigmoid')(conv9)

    model = Model(inputs=[inputs], outputs=[conv10])

    #model.compile(optimizer=Adam(lr=1e-5), loss=dice_coef_loss, metrics=[dice_coef])

    return model

def preprocess(imgs):
    imgs_p = np.ndarray((imgs.shape[0], img_rows, img_cols), dtype=np.uint8)
    for i in range(imgs.shape[0]):
        imgs_p[i] = resize(imgs[i], (img_cols, img_rows), preserve_range=True)

    imgs_p = imgs_p[..., np.newaxis]
    return imgs_p

def train_and_predict():
    print('-'*30)
    print('Loading and preprocessing train data...')
    print('-'*30)
    imgs, imgs_mask = load_train_data()

    imgs = preprocess(imgs)
    imgs_mask = preprocess(imgs_mask)

    imgs = imgs.astype('float32')
    mean = np.mean(imgs)  # mean for data centering
    std = np.std(imgs)  # std for data normalization

    imgs -= mean
    imgs /= std

    imgs_mask = imgs_mask.astype('float32')
    imgs_mask /= 255.  # scale masks to [0, 1]

    loss_per_fold_test = []
    accuracy_per_fold_test = []
    Dice_per_fold_test = []
    Jaccard_per_fold_test = []
```

```
            sensitivity_per_fold_test = []
            specificity_per_fold_test = []
            SSIM_per_fold_test = []
            precision_per_fold_test=[]

            # Define the K-fold Cross Validator
            kfold = KFold(n_splits=kat, shuffle=True)
            # K-fold Cross Validation model evaluation
            fold_no = 1
            for train, test in kfold.split(imgs, imgs_mask):
               model = fibov4()
               model.compile(optimizer=Adam(lr=2e-4), loss='binary_crossentropy',
metrics=["accuracy",dice_coef,jaccard_coef,specificity,sensitivity,SSIM,precision_m])
               model_checkpoint = ModelCheckpoint('%s.h5'%sabit, monitor='loss', save_best_only=True)
                # Generate a print
               print('-'*50)
               print(f'Training for fold {fold_no} ...')

               model.fit(imgs[train], imgs_mask[train], batch_size=batch, epochs=500, verbose=1,
shuffle=True,
                     callbacks=[model_checkpoint])

                # Generate generalization metrics
               score = model.evaluate(imgs[test], imgs_mask[test], verbose=0)
         #             print(f'Score for fold {fold_no}: {model.metrics_names[0]} of {scores[0]};
{model.metrics_names[1]} of {scores[1]*100}%')
               loss_per_fold_test.append(score[0])
               accuracy_per_fold_test.append(score[1])
               Dice_per_fold_test.append(score[2])
               Jaccard_per_fold_test.append(score[3])
               specificity_per_fold_test.append(score[4])
               sensitivity_per_fold_test.append(score[5])
               SSIM_per_fold_test.append(score[6])
               precision_per_fold_test.append(score[7])

               mask_dir = '%s/masks'%sabit
               if not os.path.exists(mask_dir):
                    os.mkdir(mask_dir)
               j = 0
               for image in imgs_mask[test]:
```

```
            image = (image[:, :, 0] * 255.).astype(np.uint8)
            imsave(os.path.join(mask_dir, str(j) + '_%s_mask.png'%fold_no), image)
            j = j+1

        test_dir = '%s/tests'%sabit
        if not os.path.exists(test_dir):
            os.mkdir(test_dir)
        k = 0
        for image in imgs[test]:
            image = (image[:, :, 0] * 255.).astype(np.uint8)
            imsave(os.path.join(test_dir, str(j) + '_%s_test.png'%fold_no), image)
            k = k+1

        imgs_mask_test = model.predict(imgs[test], verbose=1)
        #np.save('imgs_mask_test.npy', imgs_mask_test)

        pred_dir = '%s/preds'%sabit
        if not os.path.exists(pred_dir):
            os.mkdir(pred_dir)
        i=0
        for image in imgs_mask_test:
            image = (image[:, :, 0] * 255.).astype(np.uint8)
            imsave(os.path.join(pred_dir, str(i) + '_%s_pred.png'%fold_no), image)
            i = i+1
         # Increase fold number
        fold_no = fold_no + 1

    # == Provide average scores ==
    print('-'*50)
    print('Score per fold')
    for i in range(0, len(Dice_per_fold_test)):
        print('-'*50)
        print(f'> Fold {i+1} - Loss: {loss_per_fold_test[i]} - Dice: {Dice_per_fold_test[i]}%')
        print(f'> Fold {i+1} - jaccard: {Jaccard_per_fold_test[i]} - SSIM: {SSIM_per_fold_test[i]}%')
        print(f'> Fold {i+1} - spes: {specificity_per_fold_test[i]} - sens: {sensitivity_per_fold_test[i]}%')
        print(f'> Fold {i+1} - precision: {precision_per_fold_test[i]}%')
    print('-'*50)
    print('Average scores for all folds:')
```

```
print(f'> Ortalama Dice: {np.mean(Dice_per_fold_test)} (+- {np.std(Dice_per_fold_test)})')

print(f'> jaccard: {np.mean(Jaccard_per_fold_test)}')
print(f'> sens: {np.mean(sensitivity_per_fold_test)}')
print(f'> spes: {np.mean(specificity_per_fold_test)}')
print(f'> ssim: {np.mean(SSIM_per_fold_test)}')

print(f'> acc: {np.mean(accuracy_per_fold_test)}')
print(f'> Loss: {np.mean(loss_per_fold_test)}')
print(f'> prec: {np.mean(precision_per_fold_test)}')
print('-'*50)

with open('%s/Dice_test.txt'%sabit, 'w') as f:
    for item in Dice_per_fold_test:
        f.write("%s\n" % item)
    f.write("ortlama dice\n")
    f.write("np.mean(Dice_per_fold_test) +- np.std(Dice_per_fold_test)")
with open('%s/Jaccard_test.txt'%sabit, 'w') as f:
    for item in Jaccard_per_fold_test:
        f.write("%s\n" % item)
    f.write("ortlama jaccard\n")
    f.write("np.mean(Jaccard_per_fold_test) +- np.std(Jaccard_per_fold_test)")
with open('%s/specificity_test.txt'%sabit, 'w') as f:
    for item in specificity_per_fold_test:
        f.write("%s\n" % item)
    f.write("ortlama specificity\n")
    f.write("np.mean(specificity_per_fold_test) +- np.std(specificity_per_fold_test)")
with open('%s/sensitivity_test.txt'%sabit, 'w') as f:
    for item in sensitivity_per_fold_test:
        f.write("%s\n" % item)
    f.write("ortlama sensitivity\n")
    f.write("np.mean(sensitivity_per_fold_test) +- np.std(sensitivity_per_fold_test)")
with open('%s/SSIM_test.txt'%sabit, 'w') as f:
    for item in SSIM_per_fold_test:
        f.write("%s\n" % item)
    f.write("ortlama SSIM\n")
    f.write("np.mean(SSIM_per_fold_test) +- np.std(SSIM_per_fold_test)")
with open('%s/prec_test.txt'%sabit, 'w') as f:
    for item in precision_per_fold_test:
        f.write("%s\n" % item)
```

```
            finish_time = time.time() - start_time
            print("%s seconds"%finish_time)
            print("Sure:", timedelta(seconds=int(round(finish_time))))

            sure = []
            sure.append(timedelta(seconds=int(round(finish_time))))

            with open('%s/sure_test.txt'%sabit, 'w') as f:
                for item in sure:
                    f.write("%s\n" % item)
        if __name__ == '__main__':
train_and_predict()
```

KAYNAKLAR

Albawi S, Mohammed TA, Al-Zawi S. Understanding of a convolutional neural network. 2017 International Conference on Engineering and Technology (ICET), 1-6, 21-23 Aug. 2017.

Carass A, Roy S, Gherman A, Reinhold JC, Jesson A, Arbel T, Maier O, Handels H, Ghafoorian M, Platel B, Birenbaum A, Greenspan H, Pham DL, Crainiceanu CM, Calabresi PA, Prince JL, Roncal WRG, Shinohara RT, Oguz I, 2020. Evaluating White Matter Lesion Segmentations with Refined Sørensen-Dice Analysis. Scientific Reports, 10, 1, 8242.

Chang C, Girod B, 2007. Direction-Adaptive Discrete Wavelet Transform for Image Compression. IEEE Transactions on Image Processing, 16, 5, 1289-302.

Chu C, Oda M, Kitasaka T, Misawa K, Fujiwara M, Hayashi Y, Nimura Y, Rueckert D, Mori K. Multi-organ segmentation based on spatially-divided probabilistic atlas from 3D abdominal CT images. International conference on medical image computing and computer-assisted intervention, 165-72.

Chung NC, Miasojedow B, Startek M, Gambin A, 2019. Jaccard/Tanimoto similarity test and estimation methods for biological presence-absence data. BMC Bioinformatics, 20, 15, 644.

Farag A, Lu L, Roth HR, Liu J, Turkbey E, Summers RMJIToIP, 2016. A bottom-up approach for pancreas segmentation using cascaded superpixels and (deep) image patch labeling. 26, 1, 386-99.

Fawcett T, 2006. An introduction to ROC analysis. Pattern Recognition Letters, 27, 8, 861-74.

Fukushima K, Miyake S, 1982. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In: Competition and cooperation in neural nets. Eds: Springer, p. 267-85.

Ghazali KH, Mansor MF, Mustafa MM, Hussain A. Feature Extraction Technique using Discrete Wavelet Transform for Image Classification. 2007 5th Student Conference on Research and Development, 1-4, 12-11 Dec. 2007.

Gibson E, Giganti F, Hu Y, Bonmati E, Bandula S, Gurusamy K, Davidson B, Pereira SP, Clarkson MJ, Barratt DCJItomi, 2018. Automatic multi-organ segmentation on abdominal CT with dense v-networks. 37, 8, 1822-34.

Han Y, Ye JCJItomi, 2018. Framing U-Net via deep convolutional framelets: Application to sparse-view CT. 37, 6, 1418-29.

Hu J, Shen L, Sun G. Squeeze-and-excitation networks, 7132-41, 2018.

Jégou S, Drozdzal M, Vazquez D, Romero A, Bengio Y. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 11-9.

Jin J, Dundar A, Culurciello E, 2014. Flattened convolutional neural networks for feedforward acceleration. arXiv preprint arXiv:1412.5474.

Karasawa Ki, Oda M, Kitasaka T, Misawa K, Fujiwara M, Chu C, Zheng G, Rueckert D, Mori KJMia, 2017. Multi-atlas pancreas segmentation: Atlas selection based on vessel structure. 39, 18-28.

Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection, 1137-45, 1995.

LeCun Y, Bottou L, Bengio Y, Haffner P, 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86, 11, 2278-324.

Liu S, Yuan X, Hu R, Liang S, Feng S, Ai Y, Zhang YJIA, 2019. Automatic Pancreas Segmentation via Coarse Location and Ensemble Learning. 8, 2906-14.

Lu L, Jian L, Luo J, Xiao BJIA, 2019. Pancreatic Segmentation via Ringed Residual U-Net. 7, 172871-8.

Man Y, Huang Y, Feng J, Li X, Wu FJItomi, 2019. Deep q learning driven ct pancreas segmentation with geometry-aware u-net. 38, 8, 1971-80.

Oda M, Nakaoka T, Kitasaka T, Furukawa K, Misawa K, Fujiwara M, Mori K. Organ Segmentation from 3D Abdominal CT Images Based on Atlas Selection and Graph Cut. Abdominal Imaging. Computational and Clinical Applications, 181-8, 2012//, Berlin, Heidelberg.

Okada T, Linguraru MG, Hori M, Summers RM, Tomiyama N, Sato YJMia, 2015. Abdominal multi-organ segmentation from CT images using conditional shape–location and unsupervised intensity priors. 26, 1, 1-18.

Oliveira, B., Queirós, S., Morais, P., Torres, H. R., Gomes-Fonseca, J., Fonseca, J. C., & Vilaça, J. L. (2018). A novel multi-atlas strategy with dense deformation field reconstruction for abdominal and thoracic multi-organ segmentation from computed tomography. Medical Image Analysis, 45, 108-120. doi:https://doi.org/10.1016/j.media.2018.02.001

Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. International Conference on Medical image computing and computer-assisted intervention, 234-41.

Roth HR, Farag A, Lu L, Turkbey EB, Summers RM. Deep convolutional networks for pancreas segmentation in CT imaging. Medical Imaging 2015: Image Processing, 94131G.

Roth HR, Lu L, Lay N, Harrison AP, Farag A, Sohn A, Summers RMJMia, 2018. 45, 94-107.

Roth HR, Oda H, Zhou X, Shimizu N, Yang Y, Hayashi Y, Oda M, Fujiwara M, Misawa K, Mori KJCMI, Graphics, 2018. An application of cascaded 3D fully convolutional networks for medical image segmentation. 66, 90-9.

Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R, 2014. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15, 1, 1929-58.

Stehman SV, 1997. Selecting and interpreting measures of thematic classification accuracy. Remote Sensing of Environment, 62, 1, 77-89.

Suzuki M, Linguraru MG, Okada K. Multi-organ segmentation with missing organs in abdominal CT images. International Conference on Medical Image Computing and Computer-Assisted Intervention, 418-25.

Szandała T, 2021. Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks. In: Bio-inspired Neurocomputing. Eds: Bhoi AK, Mallick PK, Liu C-M, Balas VE. Singapore: Springer Singapore, p. 203-24.

Tang Z, Peng X, Geng S, Zhu Y, Metaxas DNJapa, 2018. CU-net: coupled U-nets.

Tong T, Wolz R, Wang Z, Gao Q, Misawa K, Fujiwara M, Mori K, Hajnal JV, Rueckert DJMia, 2015. Discriminative dictionary learning for abdominal multi-organ segmentation. 23, 1, 92-104.

Wang L, Chen R, Wang S, Zeng N, Huang X, Liu CJIA, 2019. Nested Dilation Network (NDN) for Multi-Task Medical Image Segmentation. 7, 44676-85.

Wolz R, Chu C, Misawa K, Fujiwara M, Mori K, Rueckert D, 2013. Automated Abdominal Multi-Organ Segmentation With Subject-Specific Atlas Generation. IEEE Transactions on Medical Imaging, 32, 9, 1723-30.

Wu J, Zhang Y, Wang K, Tang XJIA, 2019. Skip Connection U-Net for White Matter Hyperintensities Segmentation From MRI. 7, 155194-202.

Wu X, Zhao LJIA, 2019. Study on iris segmentation algorithm based on dense U-Net. 7, 123959-68.

Xiuqin P, Zhang Q, Zhang H, Li SJIA, 2019. A fundus retinal vessels segmentation scheme based on the improved deep learning U-Net model. 7, 122634-43.

Xu Z, Burke RP, Lee CP, Baucom RB, Poulose BK, Abramson RG, Landman BAJMia, 2015. Efficient multi-atlas abdominal segmentation on clinically acquired CT with SIMPLE context learning. 24, 1, 18-27.

Yang Z, Zhang L, Zhang M, Feng J, Wu Z, Ren F, Lv Y. Pancreas Segmentation in Abdominal CT Scans using Inter-/Intra-Slice Contextual Information with a

Cascade Neural Network. 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 5937-40.

Yerushalmy J, 1947. Statistical Problems in Assessing Methods of Medical Diagnosis, with Special Reference to X-Ray Techniques. Public Health Reports (1896-1970), 62, 40, 1432-49.

Zhang J, Du J, Liu H, Hou X, Zhao Y, Ding MJIA, 2019. LU-NET: an Improved U-Net for ventricular segmentation. 7, 92539-46.

Zhou W, Bovik AC, Sheikh HR, Simoncelli EP, 2004. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing, 13, 4, 600-12.

Zhou X, Ito T, Takayama R, Wang S, Hara T, Fujita H, 2016. Three-dimensional CT image segmentation by combining 2D fully convolutional network with 3D majority voting. In: Deep Learning and Data Labeling for Medical Applications. Eds: Springer, p. 111-20.

Zhou Y, Xie L, Shen W, Wang Y, Fishman EK, Yuille AL. A fixed-point model for pancreas segmentation in abdominal CT scans. International conference on medical image computing and computer-assisted intervention, 693-701.

Zhou Z, Siddiquee MMR, Tajbakhsh N, Liang J, 2018. Unet++: A nested u-net architecture for medical image segmentation. In: Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support. Eds: Springer, p. 3-11.

Zografos V, Valentinitsch A, Rempfler M, Tombari F, Menze B. Hierarchical multi-organ segmentation without registration in 3D abdominal CT images. International MICCAI Workshop on Medical Computer Vision, 37-46.