Retinal Görüntülerde Tıbbi Sementasiyonu Projesi

Kocaeli Üniversitesi Bilgisayar Mühendislik Bölümü Yapay Zeka Proje

200201137 Marah Alasi — 190201140 Mohamed Helwa — 200201147 Muhammad Abdan SYAKURA — 190201141 Robera Tadesse Gobosho — 200201120 Issa AJNADI

21.04.2024

Özet

Bu makalede [1] yürütülen araştırmayı ve model uygulamasını kullanarak, retina görüntülerinde kan damarlarını segmente etmek için kendi BCDU-Net modelimizi eğittik. Tıbbi görüntü segmentasyonu için Bi-yönlü ConvLSTM'li Yoğun Bağlantılı Konvolüsyonlar (BCDU-Net) önerilmiştir. Bu modelde, U-Net, bi-yönlü ConvLSTM (BConvLSTM) ve yoğun bağlantılı konvolüsyonların mekanizmasından tam anlamıyla faydalanmaktadır. U-Net'in skip bağlantısında basit bir birleştirmenin yerine, BConvLSTM kullanarak ilgili kodlama (encode-decode) yolundan ve önceki çözümleme üst konvolüsyon katmanından çıkarılan özellik haritalarını non-lineer bir şekilde birleştirilmektedir.

---

1 Giriş

Tıbbi görüntüler, tıbbi tedavi ve teşhis süreçlerinde önemli bir rol oynar. Bilgisayar Destekli Teşhis (CAD) sistemlerinin amacı, doktorlara tıbbi görüntülerin kesin yorumunu sağlayarak büyük bir insan grubunun daha iyi tedavi edilmesini sağlamaktır. Ayrıca, tıbbi görüntülerin otomatik işlenmesi, insan temelli işlemenin zamanını, maliyetini ve hatalarını azaltır. Bu alandaki ana araştırma alanlarından biri, birçok tıbbi görüntüleme çalışmasında kritik bir adım olan tıbbi görüntü segmentasyonudur. Diğer görüntü işleme alanları gibi, derin öğrenme ağları (deep learning networks),

tıbbi görüntülerde olağanüstü sonuçlar elde eder ve geleneksel yöntemleri geride bırakır. Derin sinir ağları genellikle sınıflandırma görevlerinde kullanılır, burada ağın çıktısı, belirli bir giriş görüntüsüne ilişkilendirilmiş tek bir etiket veya olasılık değerleridir. Bu ağlar, bazı yapısal özellikler sayesinde iyi çalışır, bunlar arasında aktivasyon fonksiyonu, farklı verimli optimizasyon algoritmaları ve ağ için bir düzenleyici olarak dropout bulunur. Bu ağlar, büyük miktarda veriye ihtiyaç duyar ve ağ parametrelerinin büyük sayısı göz önüne alındığında iyi bir genelleme davranışı sağlar. Tıbbi görüntü segmentasyonu başarılı bir şekilde gerçekleştirilmesi için, büyük (ve etiketli) veri kümelerinin bulunmaması gerekit. Tıbbi görüntü segmentasyonunda, görüntü seviyesinde etiketleme değil, piksel bazında etiketleme gerekir.

Fully convolutional neural network (FCN), ilk olarak görüntü segmentasyonuna uygulanan ilk derin ağlardan biriydi. bu mimariyi U-Net'e genişleterek, büyük miktarda eğitim verisinin ihtiyacını kullanarak iyi segmentasyon sonuçları elde etti. Onların ağı, kodlama (encoding) ve kod çözme (decoding) yollarından oluşur. Kodlama yolu, giriş verilerinden azaltılmış boyutlu birçok özellik haritası çıkarır. Kod çözme yolu, üst konvolüsyonlar yaparak girişle aynı boyutta segmentasyon haritaları üretmek için kullanılır.

Bu makalede [1], U-Net'in genişletilmiş bir versiyonu olan BCDU-Net'i önerilmektedir. Bu genişletme işlemi, skip bağlantısına BConvLSTM eklenmesini ve yoğun konvolüsyonlar ile özellik haritalarının yeniden kullanılmasını içerir. Karşılık gelen kodlama katmanından gelen özellik haritaları daha yüksek çözünürlüğe sahipken, önceki üst konvolüsyon katmanından elde edilen özellik haritaları daha fazla anlamsal bilgi içerir. Basit bir birleştirmeden ziyade, bu iki tür özellik haritasını non-lineer fonksiyonlarla birleştirmek daha kesin segmentasyon çıktılarına yol açabilir. Bu nedenle, bu makalede U-Net mimarisini genişleterek, skip bağlantısına BConvLSTM ekleyerek bu iki tür özellik haritasını birleştirilmiştir.

Sıralı konvolüsyon katmanlarına sahip olmak, ağın daha çeşitli özellikleri öğrenmesine yardımcı olabilir; ancak birçok durumda, ağ gereksiz özellikler öğrenir. Bu problemi çözmek ve ağ içinde bilgi akışını artırmak için, yoğun bağlantılı konvolüsyonların fikrini kullanılmıştır. Daralan yolun son katmanında, konvolüsyon blokları o katmandaki tüm sonraki bloklara kanal bazında birleştirme ile bağlanır. Her blokta öğrenilen özellikler bir sonraki bloğa iletilir. Bu strateji, önceki katmanlardan elde edilen toplu bilgiye dayanarak çeşitli bir özellik kümesi öğrenmeye yardımcı olur ve böylece gereksiz özelliklerin öğrenilmesini önler.

2 Yöntem

2.1 Encoding Path

BCDU-Net'in daralan yolunu (Encoding Path) dört adımdan oluşur. Her adım, ardışık olarak iki adet 3×3 konvolüsyon filtresi, 2×2 maksimum havuzlama işlemi ve ReLU içerir. Her adımda özellik haritası sayısı iki katına çıkarılır. Daralan yol, giderek görüntü temsillerini çıkarır ve bu temsillerin boyutunu katman katman arttırır. Sonunda, kodlama yolundaki son katman, yüksek boyutlu ve yüksek anlamsal bilgi içeren bir görüntü temsili üretir. Orjinal U-Net, kodlama yolunun son adımında bir dizi konvolüsyon katmanı içerir. Bir ağda sıralı konvolüsyon katmanlarına sahip olmak, yöntemin farklı türde özellikler öğrenmesini sağlar. Ancak, ağ ardışık olarak gereksiz özellikler öğrenebilir.

Bu problemi çözmek için, yoğun bağlantılı konvolüsyonlar önerilmiştir. Bu, ağın performansını "toplu bilgi" fikriyle iyileştirmesine yardımcı olur; bu fikirle özellik haritaları ağ boyunca yeniden kullanılır. Bu, tüm önceki konvolüsyon katmanlarından öğrenilen özellik haritalarının, mevcut katmandan öğrenilen özellik haritası ile birleştirilerek ve ardından bir sonraki konvolüsyonun girişi olarak kullanılmak üzere iletilmesi anlamına gelir. Yoğun bağlantılı konvolüsyonların fikri, düzenli konvolüsyonlara göre bazı avantajlara sahiptir. İlk olarak, ağın gereksiz özellikler yerine çeşitli özellik haritaları öğrenmesine yardımcı olur. Ayrıca, bu fikir, ağın temsil gücünü artırarak bilgi akışına izin verir ve özellikleri yeniden kullanarak ağın performansını artırır.

2.2 Decoding Path

çözümleme yolundaki (decoding path) her adım, önceki katmanın çıktısı üzerinde bir yukarı örnekleme (up-sampling) işlevinin uygulanmasıyla başlar. Standart U-Net'te, daralan yolun ilgili özellik haritaları kırpılır ve çoğaltılır ve çözümleme yoluna kopyalanır. Bu özellik haritaları daha sonra yukarı örnekleme işlevinin çıktısı ile birleştirilir. Genişleme yolunun amacı, katman katman özellik haritalarının boyutunu artırarak nihai katmandan sonra giriş görüntüsünün orijinal boyutuna ulaşmaktır.

3 İmplementasyon

3.1 Normalizasyon

Görüntülerdeki piksel değerleri genellikle her kanal için 0 ile 255 arasında değişir (bu durumda, gri tonlama). her piksel değerini 255'e bölerek 0 ile 1 arasında bir aralığa normalleştirilir. Normalizasyon, eğitim sırasında daha hızlı yakınsamayı sağlar ve modelin yerel minimumlarda takılı kalmasını önler.

NumPy Dizilerine Dönüştürme: Veri, NumPy dizilerine dönüştürülür, bu diziler Python'da sayısal hesaplamalar için kullanılan bir veri yapısıdır. Bu dönüşüm, büyük veri kümelerinin etkili bir şekilde işlenmesine ve Keras framework'u ile uyumluluğuna olanak sağlar [Fig 4].

3.2 Model'in Mimarisi

Sequential Model: Sequential model, katmanların doğrusal bir yığınıdır. Keras modelinin en basit tipidir.

Convolutional Layers: Convolutional layers (Conv2D), convolutional neural networks (CNNs) 'nin yapı taşlarıdır. Giriş görüntüsüne bir dizi filtre uygularlar ve ilgili özellikleri çıkarırlar. Her convolutional layer'ın belirli bir sayıda filtresi (örneğin, 64, 128, 256) ve bir filtre boyutu (örneğin, 3x3) vardır. Bu parametreler, katmanın kaç özellik çıkarabileceğini ve dikkate aldığı bölgenin boyutunu belirler. ReLU aktivasyon fonksiyonu (activation='relu'), doğrusal olmayanlığı tanıtmak ve ağın karmaşık desenleri öğrenmesine yardımcı olmak için uygulanır.

Pooling Layers: Max pooling layers (MaxPooling2D), convolutional layers tarafından üretilen özellik haritalarını örnekler. Özellik haritalarının mekansal boyutlarını azaltırken en önemli bilgileri korumaya yardımcı olurlar.

Up-sampling Layers: Up-sampling layers (UpSampling2D), özellik haritalarının mekansal boyutlarını artırır. Örnekleme yapılmış özellik haritalarından orijinal giriş boyutunu ye-

niden yapılandırmak için kullanılırlar.

Final Layer: Son katman, sigmoid aktivasyon fonksiyonuna (activation='sigmoid') sahip tek bir filtreden oluşur. Segmentasyon maskelerini üretir, her piksele 0 ile 1 arasında bir değer atayarak, bu pikselin ön plana (bu durumda, segmente edilmiş nesne) ait olma olasılığını temsil eder [Fig 5].

3.3 Optimizasyon

Adam optimizer (optimizer=Adam()), sinir ağlarını eğitmek için yaygın kullanılan bir seçenektir. Eğitim sırasında öğrenme hızını uyarlar, yakınsama hızını ve stabilitesini iyileştirir.

Loss Function: Binary cross-entropy loss function (loss='binary_crossentropy'), görüntü segmentasyonu gibi ikili sınıflandırma görevleri için yaygın olarak kullanılır. Tahmin edilen segmentasyon maskeleri ile gerçek maskeler arasındaki farkı ölçer.

Metrics: Modelin eğitim sırasındaki performansı, accuracy (metrics=['accuracy']) kullanılarak değerlendirilir. Bu metrik, segmentasyon maskelerindeki doğru sınıflandırılmış piksellerin yüzdesini ölçer [Fig 6].

3.4 Model Eğitmesi

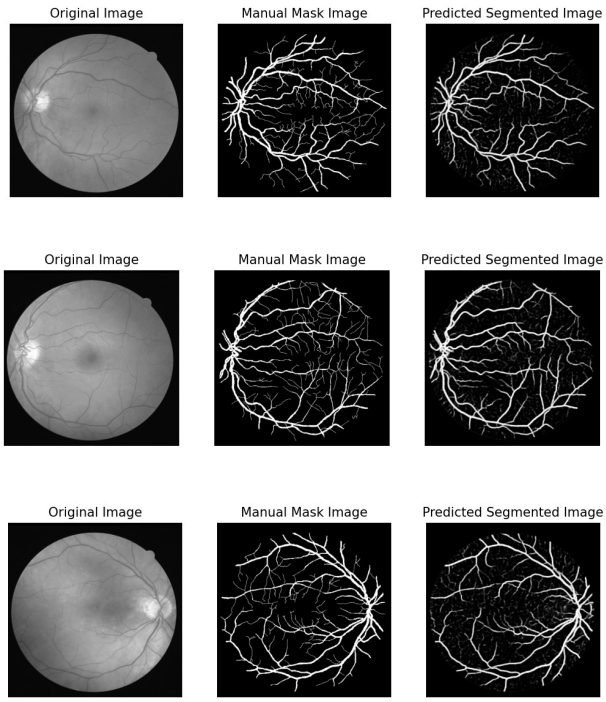
Bu bölüm, verinin modele beslenmesini ve parametrelerinin ayarlanmasını içerir. Eğitim, epochs (epochs=50) adı verilen birden çok yineleme üzerinde gerçekleşir. Her epoch, tüm veri kümesini bir kez ağdan geçirmeyi içerir. Batch Size: Veri kümesi, eğitim sırasında küçük batch'ler (batch_size=2) halinde bölünür. Batch eğitimi, modelin parametrelerini daha sık güncelleyerek ve paralel işlemeyi kullanarak daha hızlı yakınsamayı sağlar. Shuffling: shuffle=True parametresi, eğitim verilerinin her epoch için sırasının rastgeleleştirildiğini garanti eder. Bu, modelin verilerin sırasını ezberlemesini önler ve genelleştirmeyi iyileştirir [Fig 6].

4 Sonuç

Bu projede, BCDU-Net modeli kullanılarak yapılan retina kan damarı segmentasyonu çalışması, modelin bu tür medikal görevler için uygunluğunu ve etkinliğini göstermiştir.

Elde edilen bulgular, özellikle erken teşhis ve tedavi planlaması açısından, tıbbi görüntü segmentasyonu alanına önemli katkılarda bulunabilir.

Fig 1: BCDU-Net kullanark 3 Retina Görüntü Segmentasyonları elde edilmiştir.



Kaynaklar

[1] R. Azad, M. Asadi-Aghbolaghi, M. Fathy and S. Escalera, "Bi-Directional ConvLSTM U-Net with Densley Connected Convolutions," *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*2019, pp. 406-415

Fig 2: Skip bağlantılarında bi-yönlü ConvLSTM ve yoğun bağlantılı konvolüsyonlar ile BCDU-Net.

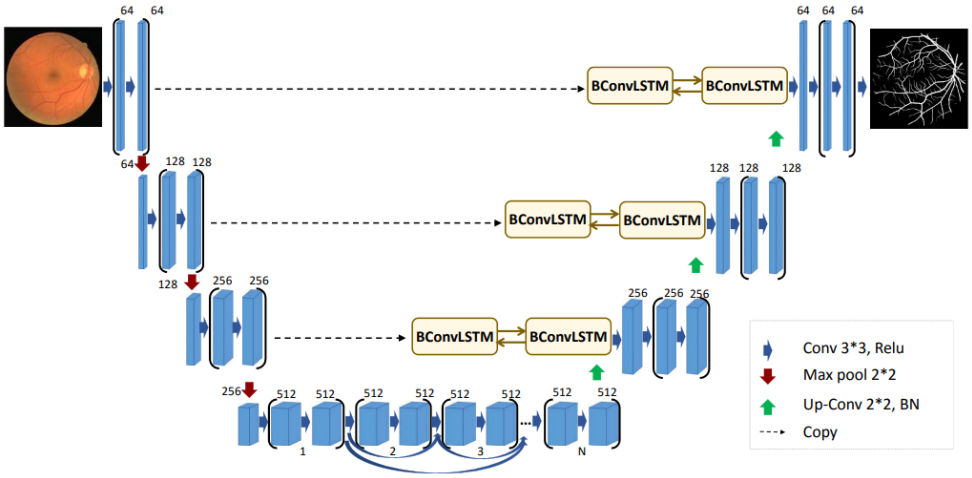


Fig 3: BCDU-Net'in Yoğun katmanları

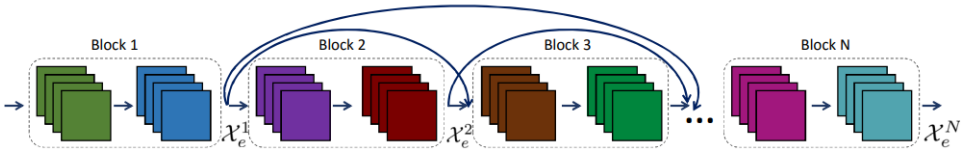


Fig 4: Normalizasyon Aşaması

```
# Load original images
for image_file in original_images_files:
    image = load_img(os.path.join(original_images_path, image_file), color_mode='grayscale')
    image = img_to_array(image)
    original_images.append(image)

# Load manual segmented images
for image_file in manual_segmented_images_files:
    image = load_img(os.path.join(manual_segmented_images_path, image_file), color_mode='grayscale')
    image = img_to_array(image)
    manual_segmented_images.append(image)

# Convert lists to numpy arrays
original_images = np.array(original_images, dtype='float') / 255
manual_segmented_images = np.array(manual_segmented_images, dtype='float') / 255
```

Fig 5: Model'in Mimarisi, Konvolüsyon katmanları

```
# Define the model
model = Sequential()
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', input_shape=(None, None, 1)))
model.add(MaxPooling2D((2, 2), padding='same'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2), padding='same'))
model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(1, (3, 3), activation='sigmoid', padding='same'))
```

Fig 6: Modelin eğitme aşamasıdır. Adam Optimizayonu ve 50 epoch kullanılmıştır.

```
# Compile the model
model.compile(optimizer=Adam(), loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(original_images, manual_segmented_images, epochs=50, batch_size=2, shuffle=True)

model.save('retina_segmentation_model_91.h5')
```