1. UNET Nedir?

UNET, biomedikal görüntü segmentasyonu için özel olarak tasarlanmış bir derin öğrenme modelidir. Adını, "birleşik ağ" anlamına gelen "U-Net" şeklinde yazılan yapısal bir özellikten almıştır. Bu mimari, özellikle biyomedikal görüntü analizi uygulamalarında kullanılmak üzere geliştirilmiştir, ancak genel olarak görüntü segmentasyonu problemlerine de uyarlanabilir.

UNET mimarisi, birçok konvolüsyonel ve pooling katmanı içeren bir ağın geniş bir kodlama (encoding) yolunu ve ardından aynı ölçüde derin bir çözme (decoding) yolunu içerir. Bu, ağın geniş bir ölçekte özellikleri öğrenmesine ve daha sonra bu özellikleri yüksek çözünürlükteki orijinal görüntü boyutuna geri getirmesine olanak tanır.

1.1. UNET Mimarisi ve Çalışma Yapısı

UNET mimarisi, görüntü segmentasyonu için özel olarak tasarlanmış bir derin öğrenme modelidir. UNET, geniş bir öğrenme ve çözme yolu içeren simetrik bir yapıya sahiptir. İsminden de aldığı gibi mimarisi U şeklindedir Fig 1 de gösterilmiştir..
Aşağıda UNET'in ana bileşenlerini ve çalışma yapısını daha ayrıntılı bir şekilde açıklıyorum:

1.1.1. Giriş Katmanı

UNET, genellikle 2D görüntü verilerini işler, bu nedenle giriş katmanı, genellikle tek bir kanala (siyah-beyaz) veya üç kanala (renkli) olan bir görüntüyü kabul eder.

1.1.2. Kodlama Yolu (Encoder)

Giriş görüntüsünden özellik haritalarını çıkaran bir dizi konvolüsyon ve pooling katmanından oluşur.
Her konvolüsyon katmanı, genellikle ReLU (Rectified Linear Unit) aktivasyon fonksiyonunu kullanır.
Her pooling katmanı, görüntünün ölçeğini küçültmek için kullanılır.

1.1.3. Birleşme ve Öğrenme Yolu (Bridge)

Encoder katmanlarından elde edilen özellik haritaları ile decoder katmanlarına geçiş yapan bir köprü katmanıdır.
Bu katmanda genellikle daha fazla konvolüsyon katmanı bulunur, bu sayede ağ daha yüksek seviyedeki özellikleri öğrenir.

1.1.4. Transpoz Konvolüsyon Katmanları (Decoder)

Decoder, encoder ile simetrik bir şekilde yapılandırılmıştır ve her bir transpoz konvolüsyon katmanı, önceki katmanın boyutunu artırarak görüntüyü orijinal boyutuna yaklaştırır.
Her transpoz konvolüsyon katmanından sonra genellikle ReLU aktivasyon fonksiyonu kullanılır.

1.1.5. Skip Bağlantıları

Encoder katmanları ile decoder katmanları arasında bağlantılar bulunur.
Skip bağlantıları, ağın daha düşük seviyedeki özelliklerini direkt olarak decoder katmanlarına ileterek, daha iyi lokalizasyon ve detay koruma sağlar.
Bu bağlantılar, ağın segmentasyon performansını artırır.

1.1.6. Çıkış Katmanı

Decoder katmanlarından gelen özellik haritalarından elde edilen çıktıyı sağlar. Genellikle, çıkış katmanında sigmoid aktivasyon fonksiyonu kullanılarak piksel tabanlı bir ikili sınıflandırma gerçekleştirilir.

UNET'in çalışma yapısı, kodlama ve çözme yolları arasındaki simetrik yapı, skip bağlantıları ve transpoz konvolüsyonları gibi özellikleri içerir. Bu özellikler, ağın hem genel özellikleri öğrenmesine hem de bu özellikleri yüksek çözünürlükteki orijinal görüntü boyutuna geri getirmesine olanak tanır. Skip bağlantıları, ağın detayları daha iyi korumasına ve daha keskin sınırlar oluşturmasına yardımcı olur.

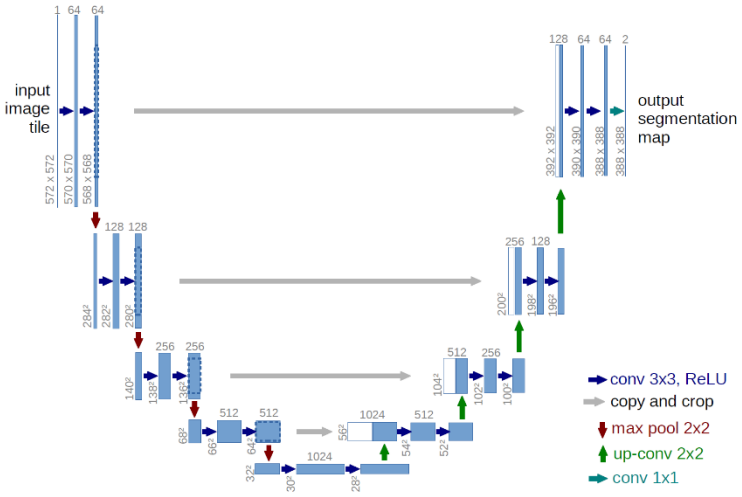

Figure 1: U-net Mimarisi

1.2. UNET Avantaj ve Dezavantajları

Avantajları:

İyi Sonuçlar: U-Net, medikal görüntüleme ve benzeri segmentasyon görevlerinde genellikle başarılı sonuçlar verir. Özellikle organ sınırlarının belirlenmesi gibi uygulamalarda etkili olabilir.

Atlama Bağlantıları: Atlama bağlantıları, encoder ve decoder arasında bilgi geçişini kolaylaştırarak, daha iyi öznitelik koruma ve lokal detayların daha iyi algılanmasına olanak tanır.

Eğitim Verimliliği: Atlama bağlantıları, eğitim sırasında gradient kaybını azaltabilir ve daha hızlı eğitim süreçlerine yol açabilir. Bu, sınırlı veri setleriyle çalışan durumlarda özellikle değerlidir.

Esnek Uygulama: U-Net, medikal görüntüleme dışında genel görüntü segmentasyon görevlerinde de başarıyla kullanılabilir. Bu, mimarinin esnekliği sayesinde mümkündür.

Dezavantajları:

Veri Seti Bağımlılığı: U-Net, genellikle büyük ve çeşitli veri setlerine ihtiyaç duyar. Bu, modelin öğrenmesini ve genelleme yapmasını iyileştirebilir.

Boyutlar Arası Farklılık: Atlama bağlantıları, boyut farklarını azaltmaya yardımcı olsa da, bazen giriş ve çıkış boyutları arasındaki farklılık nedeniyle bilgi kaybına neden olabilir.

Hesaplama İhtiyacı: U-Net, derin öğrenme modellerine özgü olarak hesaplama kaynaklarına ihtiyaç duyar. Bu, özellikle büyük veri setleri ve yüksek çözünürlüklü görüntülerle çalışıldığında önemli olabilir.

Duyarlılık: U-Net, özellikle aşırı öğrenme eğiliminde olabilir. Bu nedenle, uygun düzenleme teknikleri ve veri artırma stratejileri kullanılmalıdır.

Mimari Çeşitliliği: U-Net, belirli görevler için optimize edilmiş bir mimari sunar, ancak diğer görevler için belki de daha iyi uygun alternatif mimariler olabilir.

2. Veri Seti

2.1. Cityscapes

Cityscapes, bir şehir manzarasını anlamak ve değerlendirmek amacıyla kullanılmak üzere oluşturulmuş bir bilgisayarlı görü görevi ve veri kümesidir. Bu proje, otomotiv endüstrisinde kullanılmak üzere otomobil algılama ve otonom sürüş sistemlerini geliştirmek amacıyla tasarlanmıştır. Cityscapes, bilgisayarlı görü ve yapay zeka alanında araştırma ve geliştirme yapmak isteyen araştırmacılara ve geliştiricilere yöneliktir.

Cityscapes veri kümesi, bir dizi farklı şehir manzarasının yüksek çözünürlüklü görüntülerini ve bu görüntülerle ilişkilendirilmiş ayrıntılı etiketleri içerir. Etiketler, görüntülerdeki nesnelerin, yolların, kaldırımların ve diğer öğelerin sınıflandırılması ve segmentasyonunu içerir. Bu etiketler, özellikle otonom sürüş sistemleri için önemli olan yoldaki engelleri ve trafik akışını anlamak için kullanılabilir.

Cityscapes projesi, geniş bir veri kümesi sağlamakla birlikte, şu temel öğeleri içerir:

Görüntüler: Yüksek çözünürlüklü renkli görüntüler, farklı şehir manzaralarını temsil eder.

Etiketler: Her görüntü, nesnelerin ve öğelerin sınıflandırılması, segmentasyonu ve konumlandırılması gibi ayrıntılı etiketlerle eşlenmiştir. Bu etiketler, makine öğrenimi modellerinin eğitilmesi ve test edilmesi için kullanılır.

Senaryolar: Veri kümesi, farklı hava koşulları, ışıklandırmalar ve trafik durumları gibi çeşitli senaryolara sahiptir. Bu, modellerin genelleme yeteneklerini artırmaya yardımcı olur.

Cityscapes projesinin amacı, şehir ortamlarında otonom sürüş yeteneklerini geliştirmek için kullanılabilecek güçlü bir veri kümesi sağlamaktır. Araştırmacılar ve geliştiriciler, bu veri kümesini kullanarak kendi bilgisayarlı görü ve yapay zeka modellerini eğitebilir ve test edebilirler. Bu tür modeller, gelecekteki akıllı şehirler ve otomasyon teknolojilerinde önemli bir rol oynayabilir.

Figure 2: Cityscapes örnek veri kümesi resmi

3. PROJE

3.1. Projede Kullanılan Teknolojiler

Projemde kullandığım veri seti Paris Cityscapes'dir 2975 train resmi 500 validation resmi bulunmaktadır her bir resim 256x512 boyutundadır.
Bu kod, bir U-Net modeli kullanarak şehir manzarası görüntülerini denetimli bir şekilde öğrenmek için bir örnek içerir. Aşağıda kodun mimari yapısı, hyperparametreler ve yapılan işlemlerin bir rapor halinde açıklanmış hali bulunmaktadır

1. PyTorch: Derin öğrenme modeli (UNet) için temel kütüphane.

2. torchvision.transforms ve PIL: Resim verilerini işlemek ve dönüştürmek için kullanılan kütüphaneler.

3. torch.utils.data.Dataset ve DataLoader: Özel veri seti (CustomDataset) ve eğitim/validasyon verilerini yüklemek için kullanılan PyTorch sınıfları.

4. Matplotlib: Eğitim metriklerinin görselleştirilmesi ve örnek resim tahminlerinin görüntülenmesi için kullanılan görselleştirme kütüphanesi.

5. NumPy: Matematiksel işlemler ve veri manipülasyonu için kullanılan kütüphane.

6. Random: Rastgele bir örnek resmin seçilmesi için kullanılan Python modülü.

7. GPU (cuda): GPU kullanılabilirse, model ve veri tensörlerinin GPU üzerinde çalışacak şekilde ayarlanması.

8. Adam Optimizer ve MSELoss: Eğitim sırasında kullanılan optimizasyon algoritması (Adam optimizer) ve kayıp fonksiyonu (Mean Squared Error - MSELoss).

3.2. Projenin Çalışma Şekli

UNet Sınıfı

Giriş: Renkli (RGB) görüntülerle çalışan bir U-Net modeli.
Encoder: 3 kanallı giriş görüntüsünden başlayarak evrişim katmanları (Conv2d) ve ReLU aktivasyon fonksiyonları ile özellik haritasını küçültür.
Ardından, evrişimler arasında maksimum havuzlama katmanları (MaxPool2d) kullanılarak özellik haritası boyutu azaltılır.
Middle: Encoder'dan gelen özellik haritasını daha da işleyen evrişim katmanları ve aktivasyon fonksiyonları içerir. Maksimum havuzlama katmanları kullanılarak özellik haritası boyutu küçültülür.
Decoder: Özellik haritasını genişletmek için çift katmanlı ters evrişim katmanları (ConvTranspose2d) ve ReLU aktivasyon fonksiyonları içerir.
Encoder evrişimlerinden alınan özellik haritaları ile birleştirme (skip connection) işlemi uygulanır.
Çıkış: Renkli (RGB) görüntülerin tahmin edildiği bir çıkış katmanı.

CustomDataset Sınıfı

Giriş: Veri setinin bulunduğu dosya yolu ve dönüştürme (transform) işlemleri.
İnit Fonksiyonu: Dosya yolu ve dönüştürme işlemlerini içerir. Veri setindeki .jpg uzantılı dosyaların yollarını liste olarak tutar..
DataLoader ve Veri Dönüştürme: train_transform ve val_transform ile belirlenen dönüştürme işlemleri ile veri setleri oluşturulur. DataLoader kullanılarak eğitim ve doğrulama veri setleri yüklenir. Batch boyutu 32 olarak belirlenmiştir.
Eğitim İşlemi: Cihaz belirlenir (GPU).
Model, loss fonksiyonu (MSELoss) ve optimizer (Adam) tanımlanır.
Toplam 10 epoch boyunca eğitim işlemi gerçekleştirilir.
Eğitim ve doğrulama loss'ları kaydedilir.
Eğitim ve doğrulama doğrulukları hesaplanır.
Her epoch sonunda eğitim ve doğrulama sonuçları ekrana yazdırılır.
Sonuçlar grafiklerle görselleştirilir.
Hyperparametreler:
Batch Boyutu: 32
Öğrenme Oranı (Learning Rate): 0.001
Epoch Sayısı: 15
Giriş Görüntü Boyutu: (128, 128)
Yapılan İşlemler
Model tanımlanır ve cihaza gönderilir.
Eğitim ve doğrulama için özel veri setleri ve DataLoader'lar oluşturulur.
Loss fonksiyonu (MSELoss) ve optimizer (Adam) tanımlanır.
Toplam 15 epoch boyunca eğitim gerçekleştirilir.
Her epoch sonunda eğitim ve doğrulama loss'ları ile doğrulukları kaydedilir.
Eğitim ve doğrulama sonuçları grafiklerle görselleştirilir.
Sonuçlar
Eğitim ve doğrulama loss ve acc grafiği.

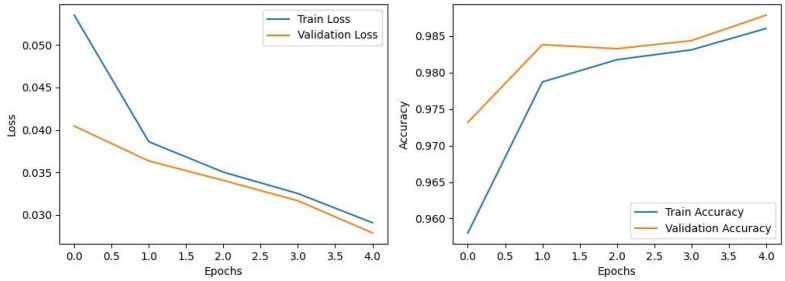

Figure 3: Yapılan çalışmanın grafiksel Sonucu(5 epoch)

Figure 4: Yapılan çalışmanın predict ve orijinal resmi(5 epoch)

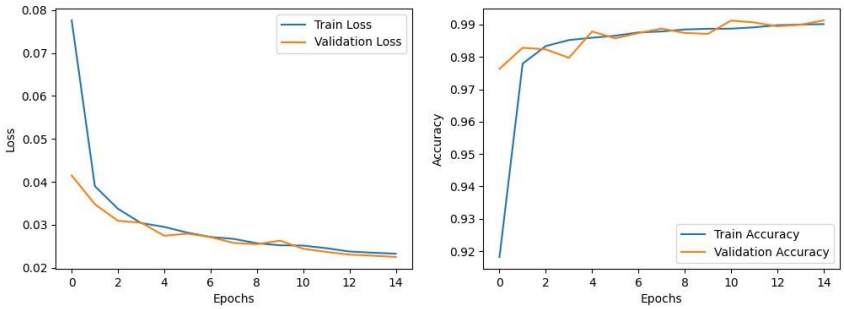

Figure 5: Yapılan çalışmanın grafiksel Sonucu(15 epoch)

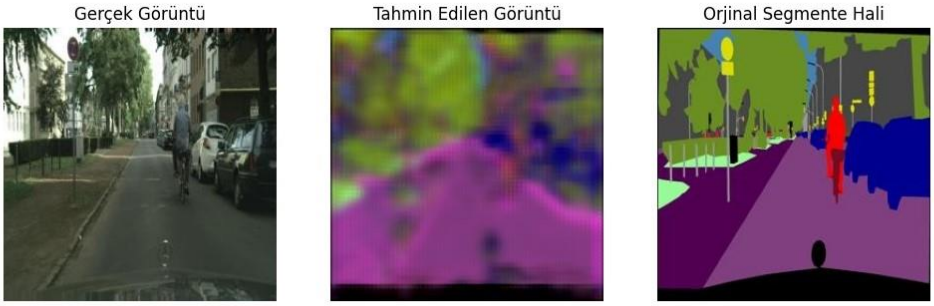

Figure 6: Yapılan çalışmanın predict ve orijinal resmi(15 epoch)

Aslında resimlere baktığımda ilk resim 5 epoch ikincisi ise 15 arasındaki 10 epoch farka rağmen bile daha iyi bir netleşme söz konusu bu yüzden muhtemlen 100 epoch 'da felan yüksek başarı gösterir ve orijinal segmente haline benzer.

4. Referanslar

1. Kızrak, A. (2019, Ağustos 6). Görüntü Bölütleme (Segmentasyon) için Derin Öğrenme: U-Net. Medium. https://ayyucekizrak.medium.com/g%C3%B6r%C3%BCnt%C3%BC-b%C3%B6l%C3%BCtleme-segmentasyon-i%C3%A7in-derin-%C3%B6%C4%9Frenme-u-net-3340be23096b (Erişim tarihi: 22 Ocak 2024)

2. "U-Net." Wikipedia. https://tr.wikipedia.org/wiki/U-Net (Erişim tarihi: 10 Ocak 2024)

3. "Cityscapes Dataset." https://www.cityscapes-dataset.com/login/ (Erişim tarihi: 09 Ocak 2024)

4. "Dans Becker's Kaggle Dataset: Cityscapes Image Pairs." https://www.kaggle.com/dansbecker/cityscapes-image-pairs (Erişim tarihi: 11 Ocak 2024)

5. "PyTorch U-Net Implementation by goldbattle." GitHub. https://github.com/goldbattle/pytorch_unet (Erişim tarihi: 20 Ocak 2024)

6. Özkan, E. B. (Tarih bilinmiyor). U-Net Mimarisi Nedir? Medium. https://medium.com/@elifbeyzaozkan/u-net-mimarisi-nedir-9730cb1fe763] (Erişim tarihi: 18 Ocak 2024)