GAN ile Tümörlü Beyin MR Görüntüleri Oluşturma

Emre Akkaya

25.04.2024

1 Keras Nedir? (Hafta 1)

Keras, makine öğrenimi ve derin öğrenme modelleri oluşturmak, eğitmek ve değerlendirmek için yüksek seviyeli bir yapay sinir ağı kütüphanesidir. François Chollet tarafından geliştirilmiş olan Keras, Python'da kolayca anlaşılabilir ve kullanımı basit bir API'ye sahiptir. Keras, kullanıcıların çeşitli yapay sinir ağı modelleri oluşturmasını ve eğitmesini sağlar. Bu modeller arasında çok katmanlı perceptronlar (MLP), evrişimli sinir ağları (CNN), tekrarlayan sinir ağları (RNN) ve bunların çeşitli kombinasyonları bulunur. Keras, temelde TensorFlow, Microsoft Cognitive Toolkit (CNTK) ve Theano gibi altta yatan hesaplama motorları üzerine inşa edilmiştir. Ancak, TensorFlow 2.0 sürümünden itibaren TensorFlow'un bir parçası haline gelmiş ve TensorFlow'un bir üst seviye API'si olarak entegre edilmiştir. Bu nedenle, TensorFlow kullanıcıları artık Keras'ı TensorFlow kütüphanesi içinde bulabilirler .[1]

2 Model Nedir

Model, bir veri setinden öğrenilmiş bilgiyi temsil eden matematiksel bir yapıdır. Makine öğrenimi ve derin öğrenme alanlarında model, genellikle bir giriş verisini alır, belirli bir görevi gerçekleştirmek için içindeki parametreleri kullanır ve bir çıkış üretir. Bir model, genellikle katmanlardan oluşur. Her katman, veriyi belirli bir şekilde dönüştürmek veya işlemek için bir dizi matematiksel işlem içerir. Katmanlar, bir modelin özelliklerini (feature) çıkarmak, veriyi temsil etmek veya sonuçları üretmek için bir araya gelir. Örneğin, derin öğrenme modelleri genellikle çok katmanlı sinir ağlarıdır (MLP, CNN, RNN, vs.). Bir sinir ağı modeli, bir dizi gizli katman (hidden layer) ve bir çıkış katmanından oluşur. Her bir gizli katman, önceki katmanın çıktısını alır, bu çıktıyı bir dizi matematiksel işlemle dönüştürür ve bir sonraki katmana iletir. Sonuç olarak, model giriş verisinden başlayarak, içindeki parametreleri kullanarak bir çıkış üretir. Modeller, belirli bir görevi gerçekleştirmek için eğitilir. Eğitim sürecinde, modelin parametreleri (ağırlıklar) veri setinden öğrenilir. Eğitim tamamlandığında, model yeni giriş verileri üzerinde tahminler yapabilir veya belirli bir görevi gerçekleştirebilir. Model, genellikle bir veri setini temsil etmek veya belirli bir görevi gerçekleştirmek için optimize edilir. Bu optimizasyon süreci, genellikle bir kayıp fonksiyonunu minimize etmeye çalışarak gerçekleştirilir [2].

3 Evrişim Nedir

Evrişim (convolution), sinyal işleme ve görüntü işleme alanlarında kullanılan önemli bir matematiksel işlemdir. Özellikle evrişim, evrişimli sinir ağları (Convolutional Neural Networks - CNN) gibi derin öğrenme modellerinin temelini oluşturan bir işlemdir. Evrişim, iki işlevin integraliyle tanımlanır. Ancak, bu integral genellikle pratik uygulamalarda ayrık bir işleme (discrete operation) olarak gerçekleştirilir. Özellikle, görüntü işleme alanında, 2 boyutlu bir görüntü üzerinde bir evrişim filtresi (kernel) ile geçiş yapma işlemi olarak tanımlanır. Matematiksel olarak, iki dizinin (veya fonksiyonun) birbirleri üzerinde kaydırılarak nokta nokta çarpımının alınması ve sonuçlarının toplanması işlemidir. Bu işlem, iki sinyalin örtüşen kısımlarının benzerliklerini belirlemek veya bir sinyalin diğerine göre konumunu belirlemek için kullanılır. Görüntü işlemede, evrişim filtresi bir çekirdek (kernel) olarak adlandırılır ve genellikle küçük boyutlarda (örneğin 3x3 veya 5x5) bir matristir. Bu çekirdek, görüntünün her pikseline uygulanarak, o pikselin ve çevresindeki piksellerin birleşimiyle yeni bir çıktı pikseli üretilir. Bu işlem, görüntüde kenar tespiti, özellik çıkarma ve diğer birçok işlemde yaygın olarak kullanılır. Evrişim, özellikle CNN'lerde, örüntü tanıma, görüntü sınıflandırma, nesne tespiti gibi görevlerde kullanılan bir temel işlemdir. Bu tür modeller, evrişim katmanları adı verilen katmanlar aracılığıyla girdi verileri üzerinde evrişim işlemleri uygularlar ve bu şekilde verilerin özelliklerini çıkarırlar [3].

4 Optimizasyon Nedir

keras.optimizers.Adam, derin öğrenme modellerini eğitirken kullanılan bir optimizasyon algoritmasıdır. Adam, adaptif moment tahmini (Adaptive Moment Estimation) yöntemini kullanır ve gradyan iniş tabanlı bir optimizasyon algoritmasıdır. Adam optimizasyon algoritması, çeşitli avantajlarıyla popülerlik kazanmıştır ve birçok derin öğrenme modelinin eğitiminde sıkça kullanılmaktadır.

4.1 Adaptif Öğrenme Oranı

Adam, her parametrenin öğrenme oranını ayrı ayrı ayarlar. Bu, farklı parametrelerin farklı hızlarda güncellenmesine olanak tanır.

4.2 Momentum

Adam, gradyanın ikinci momentini (momentum) de dahil eder. Bu, gradyan değişimlerinin momentumunu hesaplamak için kullanılır, bu da algoritmanın daha hızlı ve düzgün bir şekilde ilerlemesini sağlar..

4.3 Adam Parametreleri

Adam algoritması, öğrenme oranı (learning rate) ile birlikte Beta1 ve Beta2 olmak üzere iki momentum parametresine sahiptir. Öğrenme oranı, ağırlıkların güncellenme miktarını kontrol ederken, Beta1 geçmiş gradyanların ne kadarının korunacağını ve Beta2 gradyanların karesinin ne kadarının saklanacağını belirler. Bu hiperparametreler, Adam optimizasyon algoritmasının performansını önemli ölçüde etkiler.

4.4 Birleşik Öğrenme Kuralı

Adam, gradyanın birinci ve ikinci momentlerini birleştirerek parametre güncellemesini yapar. Bu, algoritmanın gradyan inişinde daha etkili olmasını sağlar. Adam optimizasyon algoritması, derin öğrenme modellerinde sıkça kullanılan bir optimizasyon algoritmasıdır. Ancak, her optimizasyon algoritması gibi, Adam'ın da belirli bir uygulama veya veri seti için en iyi sonucu vermediği durumlar olabilir. Bu nedenle, modelinizi eğitirken farklı optimizasyon algoritmalarını denemek ve hiperparametreleri ayarlamak önemlidir [4].

5 Veri Yükleme (Hafta 2)

Veri yükleme fonksiyonu, belirli bir klasörden görüntü verilerini okuyarak işlemek için kullanılır. Bu fonksiyon, her bir görüntünün dosya yolunu alır, ardından görüntüyü gri tonlamalı olarak okur ve 128x128 boyutlarına yeniden boyutlandırır. Daha sonra bu işlenmiş görüntüyü bir diziye ekler ve aynı zamanda etiketlerle ilişkilendirilen bir diziye hedef etiketi ekler. Fonksiyon, veri kümesindeki tüm görüntüleri dolaşırken her bir görüntüyü işler ve sonunda işlenmiş görüntü

dizisini ve etiket dizisini döndürür.Bu fonksiyon, özellikle makine öğrenimi ve derin öğrenme projelerinde kullanılan veri hazırlama sürecinde önemli bir adımdır. Görüntü verilerini işlemek için sıklıkla kullanılır ve veri kümesini model eğitimi için uygun bir formata getirir. Bu şekilde, araştırmacılar ve veri bilimcileri, derin öğrenme modellerini eğitmek için gereken veri hazırlığı sürecini otomatikleştirebilir ve standardize edebilir .

```
def load_images(folder):

    imgs = []
    target = 1
    labels = []
    for i in os.listdir(folder):
        img_dir = os.path.join(folder,i)
        try:
            img = cv2.imread(img_dir)
            img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            img = cv2.resize(img, (128,128))
            imgs.append(img)
            labels.append(target)
        except:
            continue

    imgs = np.array(imgs)
    labels = np.array(labels)

    return imgs, labels
```

Figure 1: Veri Yükleme Kod Parçası

6 Veri Normalleştirme

Veri ön işleme aşamasında, rastgele veri seçimi ve normalleştirme işlemleri sıkça kullanılan tekniklerdir. Bu adımlar, derin öğrenme modelinin eğitimi için gerekli olan veri kümesinin hazırlanmasında önemli rol oynar. Rastgele veri seçimi, genellikle veri kümesinin büyüklüğü ve çeşitliliğini artırmak amacıyla kullanılır. Bu yöntem, veri kümesinden rastgele örnekler seçerek modelin daha genel ve çeşitli verilere uyum sağlamasını sağlar. Normalleştirme ise, veri dağılımını belirli bir aralığa sıkıştırarak, verinin ortalamasını ve dağılımını düzenler. Bu şekilde, modelin daha hızlı ve istikrarlı bir şekilde eğitilmesine yardımcı olur. Örneğin, bu adımların birleştirildiği bir örnek üzerinde düşünelim: Veri kümesinden rastgele örnekler seçilir ve bu örnekler normalize edilerek belirli bir aralığa yerleştirilir. Bu işlem, modelin daha tutarlı ve genelleştirilebilir bir şekilde eğitilmesine katkı sağlar.

```
# Verilerin normalizasyonu
data_normalized = (data.astype(np.float32) - 127.5) / 127.5

# Resimleri yeniden şekillendirme
WIDTH, HEIGHT, CHANNELS = 128, 128, 1
X_train = data_normalized.reshape(-1, WIDTH, HEIGHT, CHANNELS)
print("X_train shape:", X_train.shape)
```

Figure 2: Veri Normalleştirme Kod Parçası

7 Veri Görselleştirme

Veri görselleştirme, derin öğrenme modellerinin eğitimi ve sonuçlarının analizi için kritik öneme sahiptir. Görselleştirme, veri setindeki desenleri, dağılımları ve ilişkileri anlamamıza yardımcı olan güçlü bir araçtır. Özellikle, veri görselleştirme işlemi, modelin eğitim verilerini daha iyi anlamamıza ve modelin performansını değerlendirmemize olanak tanır. Bu sayede, modelin eğitim sırasında karşılaştığı zorluklar ve başarılı olduğu alanlar daha net bir şekilde belirlenebilir. Örneğin, eğitim veri setinden rastgele bir örnek seçilerek, bu örneğin görselleştirilmesiyle, veri setindeki örneklerin çeşitliliği ve dağılımı hakkında bilgi edinilebilir. Ayrıca, modelin tahminlerini görselleştirme, modelin doğruluğunu değerlendirmek ve olası hataları belirlemek için yaygın olarak kullanılır. Bu nedenle, veri görselleştirme, derin öğrenme projelerinin her aşamasında önemli bir rol oynar ve modelin performansını iyileştirmeye yardımcı olur.

```
# Görüntüleri göster
plt.figure(figsize=(20,8))
for i in range(10):
    axs = plt.subplot(2,5,i+1)
    plt.imshow(X_train[i], cmap="gray")
    plt.axis('off')
    axs.set_xticklabels([])
    axs.set_yticklabels([])
    plt.subplots_adjust(wspace=None, hspace=None)
plt.tight_layout()
plt.show()
```

Figure 3: Veri Görselleştirme Kod Parçası

8 Üretici Model (Hafta 3)

Bu ağ rastgele gürültüyü girdi olarak alır ve veri (görüntüler gibi) üretir. Amacı gerçek verilere mümkün olduğunca yakın veriler üretmektir.

8.1 Mimari

Mimari değişiklik gösterse de, pek çok popüler GAN'daki (DCGAN gibi) üreticiler, bir görüntü oluşturmak için gürültü vektörünün üst örneklemesine yardımcı olan, aktarılmış evrişimli katmanlar kullanılarak oluşturulur. Üretici genellikle aşağıdaki bileşenlerin dizisinden oluşur.

8.2 Giriş Katmanı

Üretici, genellikle normal veya düzgün bir dağılımdan örneklenen rastgele bir gürültü vektörünü alan bir giriş katmanıyla başlar.

8.3 Tamamen Bağlı Katmanlar

Ağın başlarında, giriş gürültü vektörünü daha sonraki işlemler için uygun bir şekle dönüştürmek için tamamen bağlı katmanlar kullanılabilir.

8.4 Toplu Normalleştirme

Bu teknik genellikle önceki katmanın çıktısını normalleştirerek öğrenmeyi stabilize etmek için katmanlar arasında kullanılır.

8.5 Aktivasyon Fonksiyonları

ReLU (Rectified Linear Unit) veya Leaky ReLU, jeneratördeki aktivasyon fonksiyonları için yaygın seçimlerdir. Bu fonksiyonlar, modelin doğrusal olmayan işlevler öğrenmesini sağlar ve karmaşık veriler üretmesini destekler.

8.6 Transpoze Etkileşim Katmanları

Bu katmanlar, jeneratörün temelini oluşturur. Önceki katmandan gelen girdiyi daha yüksek bir uzamsal boyuta örnekleyerek, evrişimli katmanların bir CNN'de yaptığının tam tersini yaparlar.

8.7 Katman Yeniden Şekillendirme

Bu katmanlar, verileri istenen çıktı formatına yeniden şekillendirmek için kullanılır.

8.8 Çıkış Katmanı

Son katman, genellikle oluşturulan verinin doğasına bağlı olarak tanh veya sigmoid aktivasyon fonksiyonunu kullanır. Görüntü üretimi için, genellikle bir tanh fonksiyonu kullanılarak normalleştirilmiş bir aralıktaki piksel değerlerinin çıktısı alınır.

9 Ayrıştırıcı Model

Bu ağ, girdi olarak gerçek verileri ve Üretici tarafından oluşturulan verileri alır ve ikisini birbirinden ayırmaya çalışır. Verilen verinin gerçek olma olasılığını verir.

9.1 Mimari

Ayırıcının mimarisi genellikle geleneksel evrişimli sinir ağlarının (CNN'ler) mimarisini yansıtır, ancak bazı ayarlamalar vardır. Genellikle aşağıdaki bileşenlerin dizisinden oluşur.

9.2 Evrişimsel Katmanlar

Bu katmanlar görüntü verilerinin işlenmesinde temeldir. Giriş görüntülerinden özelliklerin çıkarılmasına yardımcı olurlar. Evrişim katmanlarının sayısı, verilerin karmaşıklığına bağlı olarak değişebilir.

9.3 Toplu Normalleştirme

Bu bazen girdiyi bir katmana normalleştirerek öğrenmeyi stabilize etmek için katmanlar arasında kullanılır.

9.4 Aktivasyon Fonksiyonları

Sızdıran ReLU, ayırıcıdaki aktivasyon fonksiyonları için yaygın bir seçimdir. Ünite aktif olmadığında küçük bir eğime izin verir ve bu da antrenman sırasında eğim akışının korunmasına yardımcı olabilir.

9.5 Havuz Katmanları

Bazı mimariler, giriş verilerinin uzamsal boyutlarını aşamalı olarak azaltmak için havuz oluşturma katmanlarını (maksimum havuzlama gibi) kullanır.

9.6 Tamamen Bağlı Katmanlar

Ağın sonunda, evrişimli katmanlar tarafından çıkarılan özellikleri işlemek için tamamen bağlı katmanlar kullanılır ve son çıktı katmanı elde edilir.

9.7 Çıkış Katmanı

Son katman tipik olarak bir olasılık değeri çıkışı sağlayan sigmoid aktivasyon fonksiyonuna sahip tek bir nörondur.

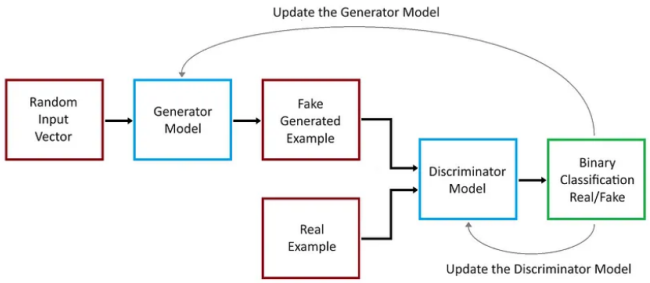

Figure 4: Üretici ve Tüketici GAN Modeli [5]

10 Çalışma Prensibi

Eğitim sırasında Jeneratör, Discriminator'ın gerçek verilerden ayırt edemeyeceği verileri üretmeye çalışırken, Discriminator gerçek verileri sahte verilerden ayırma konusunda daha iyi olmaya çalışır. İki ağ özünde bir oyunda yarışıyor: Jeneratör ikna edici sahte veriler üretmeyi amaçlıyor ve Ayırıcı ise gerçeği sahteden ayırmayı amaçlıyor.

11 Üretici Model Oluşturulması

Üretici model, gürültülü bir giriş alır ve bu girişi gerçek görüntüler gibi görünen yeni görüntülere dönüştürmek için transpoze konvolüsyon katmanları kullanır. buildgenerator fonksiyonunda, Sequential modeli kullanılarak bir üretici model oluşturuluyor.Modelde tamamen bağlı bir giriş katmanı (Dense) yer alıyor. Bu katman, gürültülü bir vektörü düzleştirmek ve ardından uygun boyutlara dönüştürmek için kullanılıyor.Ardından, Reshape katmanı ile giriş vektörü 3 boyutlu tensörlere dönüştürülüyor.Transpoze konvolüsyon katmanları (Conv2DTranspose) kullanılarak görüntü boyutları büyütülüyor ve özelliklerin daha karmaşık bir şekilde dönüştürülmesi sağlanıyor.Son olarak, çıkış katmanı, üretilen görüntüyü tanh aktivasyonu ile normalize ederek oluşturuyor.Model, binary cross-entropy kaybı ile ve Adam optimizasyon algoritması kullanılarak derleniyor.[6]

12 Ayrıştırıcı Model Oluşturulması

Ayrıştırıcı model, giriş olarak gerçek veya üretilmiş bir görüntü alır ve bu görüntünün gerçek veya sahte olduğunu sınıflandırır. builddiscriminator fonksiyonu, yine Sequential modeli kullanarak bir ayrıştırıcı model oluşturuyor. Model, evrişimli katmanlar (Conv2D) ve tamamen bağlı katmanlar (Dense) içerir. Evrişimli katmanlar, özellik haritalarını çıkararak görüntüdeki özellikleri öğrenir. Model,

sigmoid aktivasyon fonksiyonu ile bir çıkış katmanı ile sonuçlanır, bu da gelen görüntünün gerçek veya sahte olduğunu belirler. Ayrıca, model binary cross-entropy kaybı ile ve Adam optimizasyon algoritması ile derlenir.[6]

13 Evrişimli Sistemler (Hafta 4)

Aktarılmış evrişim katmanlarını anlamak için önce normal evrişim katmanlarını anlamalıyız, çünkü paten kaymayı bilmeden buz hokeycisi olamayız.

13.1 Evrişimli Sinir Ağlarının Amacı

Kısaca bir girdiyi (genellikle görüntü) bir çekirdek kullanarak alt örneklemektir

13.2 Evrişimli Sinir Ağlarının Önemi

CNN'ler, derin öğrenmede önemli bir ilerleme kaydetmiş ve özellikle görüntü sınıflandırma gibi görevlerde etkili olmuşlardır. İlk kullanımları 1990'larda karakter tanıma gibi görevlerde olmasına rağmen, Krizhevsky ve diğerleri tarafından 2012'de gerçekleştirilen ImageNet görüntü sınıflandırma yarışmasında büyük bir başarı elde edilmiştir.[7]

13.3 Evrişimli Sinir Ağlarının Karmaşıklığı

Evrişimli sinir ağlarının kullanımı, özellikle ilk kez öğrenilirken karmaşık olabilir. Evrişimli katmanların çıkış şekli, girdinin şekli, çekirdek şekli, sıfır dolgu ve adımlar gibi faktörler tarafından etkilenir. Bu faktörler arasındaki ilişki kolayca anlaşılmaz ve tam olarak çıkarsanması zordur. Bununla birlikte, tamamen bağlı katmanlar, çıkış boyutunun girdi boyutundan bağımsız olduğu daha basit bir yapıya sahiptir. Ayrıca, CNN'ler genellikle havuzlama aşamasını içerir, bu da tamamen bağlı ağlara kıyasla daha fazla karmaşıklık ekler.[7]

13.4 Evrişimli Sinir Ağları Nasıl Çalışır?

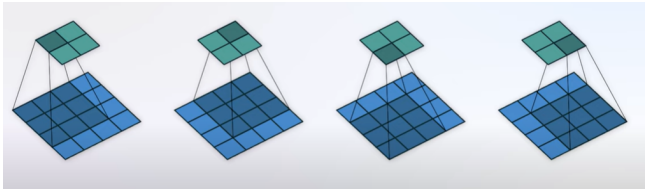

Figure 5: 2x2 lik alt örneklemenin 4x4 giriş üzerindeki adımları [8]

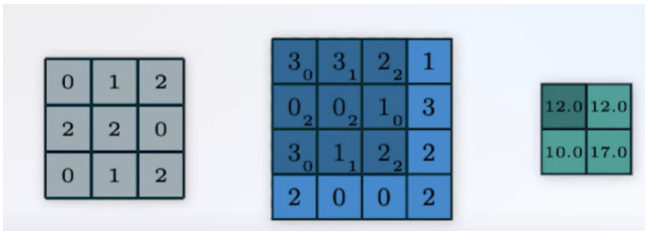

Figure 6: 3x3 lük çekirdek ve 4x4 lük giriş [8]

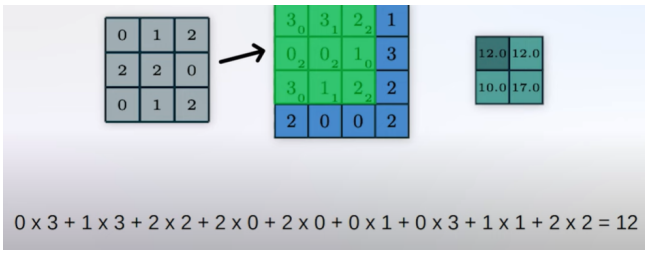

Figure 7: Çekirdeğin ilk adımı [8]

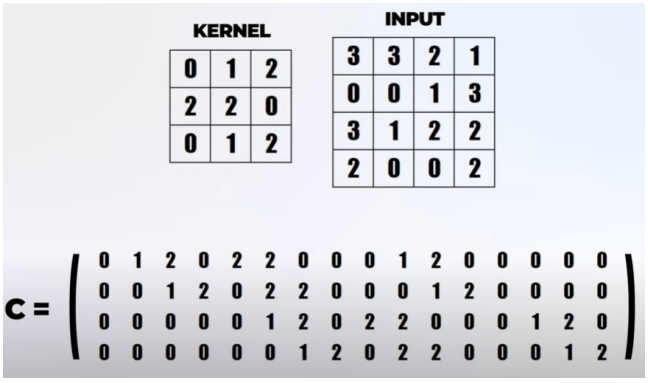

Figure 8: Vektör haline getirilmiş c evrişimi [8]

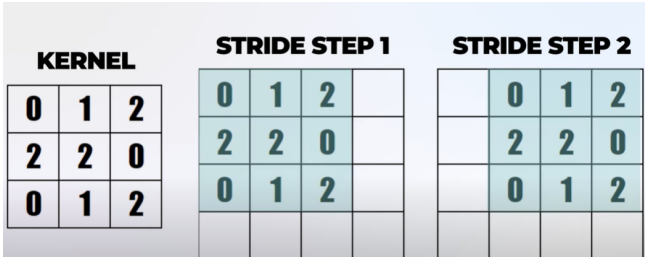

Figure 9: Evrişimi vektörleştirme adımları [8]

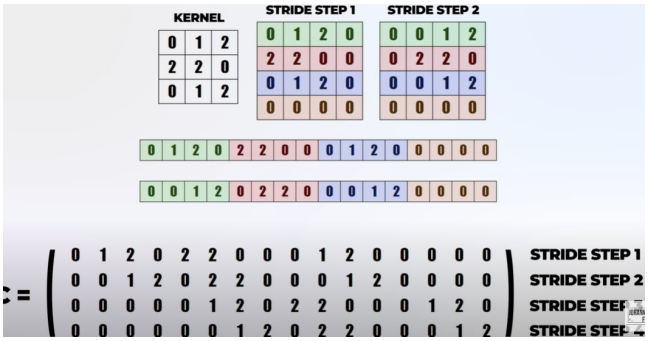

Figure 10: Evrişimi vektörleştirme adımları [8]

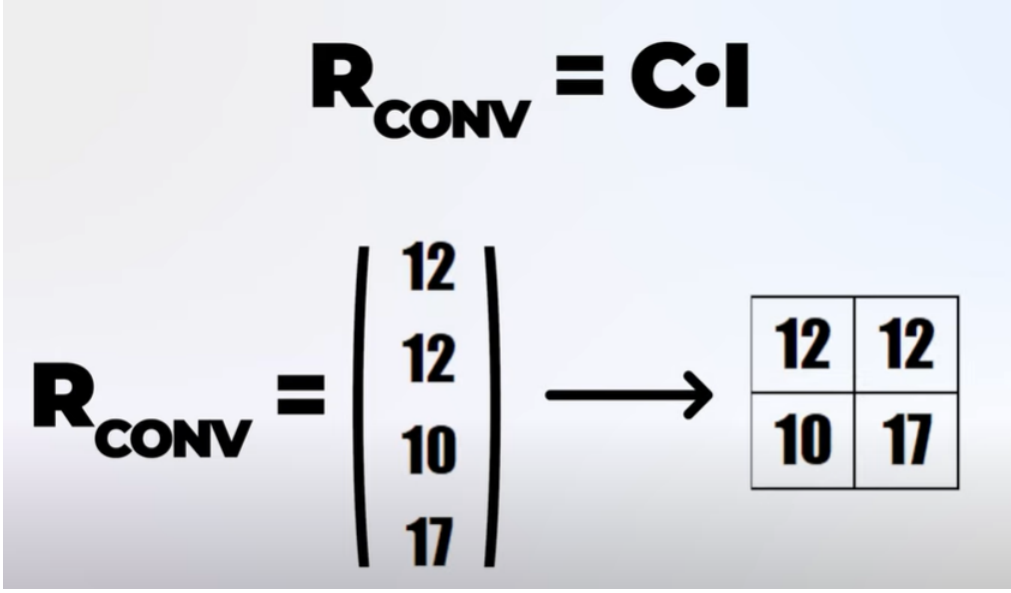

Figure 11: Evrişim formülü ve sonucu [8]

14 Aktarılmış Evrişimli Sistemler

Transpoze etkileşim katmanları, evrişimli sinir ağlarında sıklıkla kullanılan bir tür katmandır. Bu katmanlar, genellikle evrişim (convolution) işleminin tersini gerçekleştirerek girdi boyutunu genişletirler. Evrişim katmanları, girdi verisinden özelliklerin çıkarılmasını sağlarken, transpoze etkileşim katmanları bu özellikleri tekrar orijinal boyuta döndürür. Bu nedenle, genellikle görüntü işleme ve yeniden oluşturma uygulamalarında kullanılırlar. Transpoze etkileşim katmanları, bir önceki evrişim katmanının çıktısını orijinal girdi boyutuna geri döndürmek için kullanılabilir. Örneğin, düşük çözünürlüklü bir görüntünün piksel değerlerini, yüksek çözünürlüklü bir görüntünün piksel değerlerine dönüştürmek için transpoze etkileşim katmanları kullanılabilir. Bu işlem, özellikle görüntü super resolution gibi uygulamalarda önemli bir rol oynar. Bu katmanlar, genellikle çekirdek boyutu ve adım (stride) gibi parametrelerle tanımlanır. Çekirdek, oluşturulacak her bir pikselin girdi görüntüsündeki hangi bölgeye karşılık geldiğini belirler. Ayrıca, sıfır dolgu (zero padding) ve aktivasyon fonksiyonları gibi ek parametrelerle birlikte kullanılarak çıktı boyutu ve detayı kontrol edilebilir.Aktarılmış evrişimli sistemlerde, genellikle bir evrişim (convolution) işleminin çıktısının daha sonra bir transpoze evrişim (transpose convolution) işlemine tabi tutulmasıyla elde edilir. Bu işlem, evrişim sonucunun giriş boyutuna uygun hale getirilmesini sağlar. Örneğin, 2x2'lik bir evrişim sonucunun girişe uygun 4x4 boyutunda bir çıktı elde etmek için bir transpoze evrişim işlemi uygulanabilir.

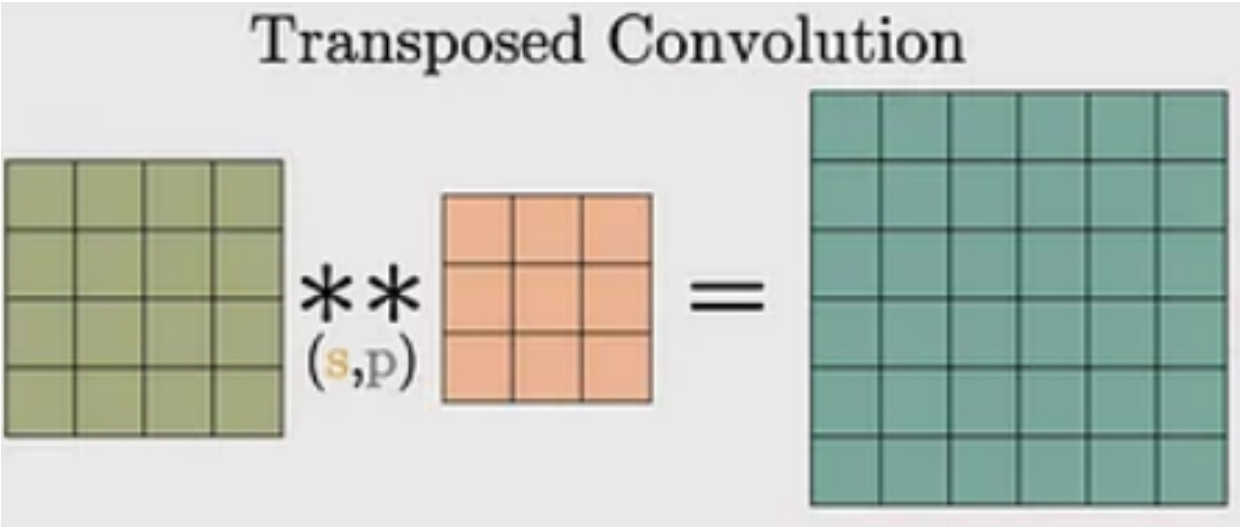

Figure 12: Tranpoze Evrişim Oluşumu [9]

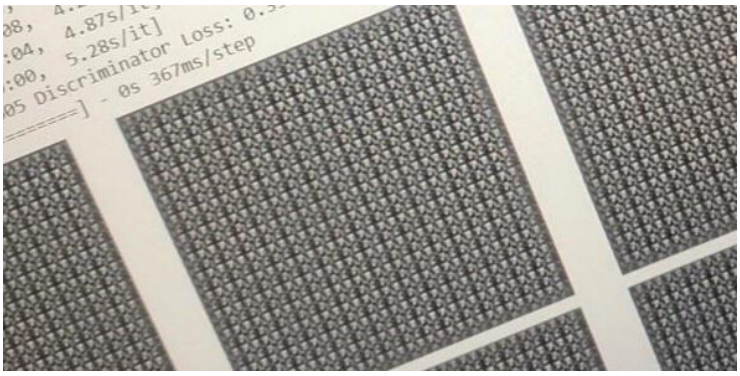

Figure 13: Eğitim denemeleri sırasında Epoch 1' deki örnek görüntü

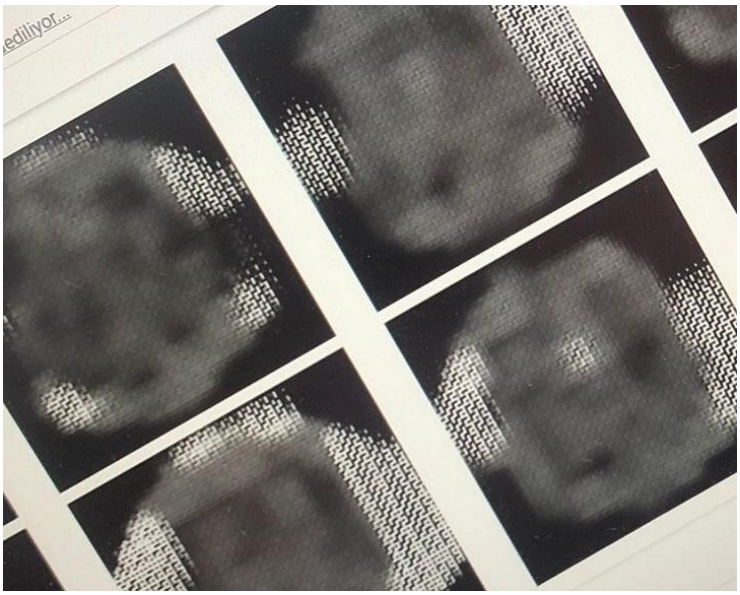

Figure 14: Eğitim denemeleri sırasında Epoch 50' deki örnek görüntü

References

[1] Ö. G. D. M. GEZER and S. SAYLAN, "Açık kaynaklı makine ögrenmesi kütüphaneleri," *MÜHENDISLIKTE YAPAY*, p. 53.

[2] H. H. ÖNDER, "Yapay zeka programlama teknikleri ve bilgisayar destekli eğitim," *Sakarya Üniversitesi Eğitim Fakültesi Dergisi*, no. 3, 2001.

[3] P. CANBAY, "Sağlikta yapay zeka: Derin öğrenme teknikleri ve uygulamalari,"

[4] A. Coşkun, "Yapay zeka optimizasyon teknikleri: literatür değerlendirmesi," *Fırat Üniversitesi Doğu Araştırmaları Dergisi*, vol. 5, no. 2, pp. 142–146, 2007.

[5] M. D. Pra, "Generative adversarial networks." `https://medium.com/@marcodelpra/generative-adversarial-networks-dba10e1b4424`, Year Published/ 30.10.2023. Accessed: 01.04.2024.

[6] H. Singh, "Generating brain mri images with dc-gan." `https://www.kaggle.com/code/harshsingh2209/generating-brain-mri-images-with-dc-gan/notebook`, Year Published/ 22.02.2023. Accessed: 17.04.2024.

[7] F. V. Vincent Dumoulin, "A guide to convolution arithmetic for deep learning." `https://arxiv.org/pdf/1603.07285v1.pdf`, Year Published/ 24.03.2016. Accessed: 17.04.2024.

[8] J. Frey, "Transposed convolutions explained: A fast 8-minute explanation — computer vision." `https://www.youtube.com/watch?v=xoAv6D05j7g`, Year Published/ 24.03.2016. Accessed: 17.04.2024.

[9] A. Anvar, "What is transposed convolutional layer?." `https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11?gi=9d86a68e5a56`, Year Published/ 06.03.2020. Accessed: 17.04.2024.