



GAN ile Tümörlü Beyin MR Görüntüleri Oluşturma

Emre Akkaya

14.06.2024

Özet

Bu projede, beyin tümörlerinin erken teşhisinde önemli bir rol oynayan bir yapay zeka modeli geliştirilmesi amaçlanmıştır. Beyin tümörleri, sağlık endüstrisinde ciddi bir sağlık sorunu oluşturmaktadır ve erken teşhis, tedavi sürecindeki başarıyı büyük ölçüde etkilemektedir. Proje kapsamında Kaggle'dan alınan *brain-tumor-dataset* veri seti kullanılmış ve proje içinde tümörlü resimlerden *seed* değişkeni kullanılarak 10 adet eğitim verisi seçilip bunların eğitim işlemleri gerçekleştirilmiştir. Eğitim süreci tamamlandıktan sonra, modelin performansı dikkatle değerlendirilmiş ve gerektiğinde iyileştirmeler yapılmıştır. Yapılan testler sonucunda, geliştirilen modelin yeni tümörlü beyin görüntülerini başarıyla tespit edebildiği ve bu alanda literatüre önemli bir katkı sağladığı görülmüştür.

Anahtar Kelimeler: Beyin Tümörü, MR Görüntüleri, Derin Öğrenme, GAN, Yapay Zeka, Tıbbi Görüntüleme

1 Giriş

Beyin tümörleri, dünya genelinde ciddi bir sağlık sorunu oluşturmaktadır. Erken teşhis, hastalığın tedavisinde kritik bir rol oynamaktadır. Geneliksel teşhis yöntemleri çoğunlukla zaman alıcı ve maliyetli olabilmektedir. Bu çalışmada, beyin tümörlerinin teşhisinde kullanılabilecek bir yapay zeka modeli geliştirilmesi amaçlanmıştır. Çalışmadaki motivasyonumuz, sağlık endüstrisinde önemli bir yere sahip olan tıbbi görüntüleme alanına yenilikçi bir katkı sağlamaktır. Beyin tümörlerinin erken teşhisini, hastaların tedavi süreçlerinde önemli bir avantaj sağlayacaktır. Bu çalışmaya başlama sebebimiz, mevcut yöntemlerin yetersizliği ve yapay zeka modellerinin bu alandaki potansiyel faydalara duyduğumuz inançtır. Bu çalışmada, GAN (Generative Adversarial Network) kullanılarak yeni beyin tümörlü MR görüntüleri oluşturulmuştur. Önceki çalışmalarda, yapay zeka modellerinin tıbbi görüntülemede başarılı sonuçlar verdiği görülmüş, ancak çoğu çalışma yeterli veri seti bulunmaması nedeniyle sınırlı kalmıştır. Bu çalışmada, yeterli miktarda veri toplanarak modelin performansı artırılmıştır. Bu çalışma, beyin tümörlerinin teşhisinde yapay zeka kullanımına yeni bir bakış açısı getirmiştir. Modelin eğitimi için 155 adet tümörlü ve 98 adet tümörsüz MR görüntüsü kullanılmıştır.

2 Veri/Literatür Araştırması

Bu bölümde, çalışmada kullanılan veri seti ve deneysel çalışma düzenekleri hakkında detaylı bilgiler verilmiştir. Beyin MR görüntüleri, çeşitli görüntüleme teknikleri kullanılarak toplanmıştır. Veri seti, 155 adet tümörlü ve 98 adet tümörsüz MR görüntüsünden oluşmaktadır. Bu veriler, Google Drive üzerinden yüklenmiş ve ön işlemlerden geçirilmiştir. Görüntüler, 128x128 boyutlarına küçültüllererek gri tonlamaya dönüştürülmüştür. GAN modeli, görüntülerin gerçekliğini artırmak ve daha yüksek kaliteli görüntüler oluşturmak amacıyla kullanılmıştır. Eğitim sürecinde, veri seti normalize edilerek modele sunulmuş ve çeşitli iyileştirmeler yapılmıştır.

3 Yöntem

Çalışmada, GAN modeli kullanılarak yeni beyin tümörlü MR görüntüleri oluşturulmuştur. GAN modeli, bir üretici (generator) ve bir ayırtıcı (discriminator) modelden oluşmaktadır. Üretici model, rastgele gürültüden gerçekçi görüntüler üretirken, ayırtıcı model bu görüntülerin gerçek mi yoksa sahte mi olduğunu ayırt etmeye çalışır. Üretici model, Transposed Convolution katmanları kullanarak rastgele gürültüden görüntüler üretir. Ayırtıcı model ise Convolutional Neural Network (CNN) kullanarak bu görüntülerini sınıflandırır. Eğitim sürecinde, üretici ve ayırtıcı modeller sırayla eğitilerek birbirlerini geliştirirler. Modelin performansı, gerçek ve sahte görüntülerin doğru sınıflandırılması ile ölçülmüştür. Eğitim sürecinde kullanılan optimizasyon teknikleri ve parametreler detaylı bir şekilde açıklanmıştır.

3.1 Üretici Model

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

3.2 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.

3.3 Giriş Katmanı

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

3.4 Tamamen Bağlı Katmanlar

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

3.5 Toplu Normalleştirme

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

3.6 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.

3.7 Transpoze Etkileşim Katmanları

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

3.8 Katman Yeniden Şekillendirme

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

3.9 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.

3.10 Ayırtıcı Model

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

3.11 Mimari

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

3.12 Evrişimsel Katmanlar

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

3.13 Toplu Normalleştirme

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

3.14 Aktivasyon Fonksiyonları

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

3.15 Havuz Katmanları

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

3.16 Tamamen Bağlı Katmanlar

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

3.17 Çıkış Katmanı

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

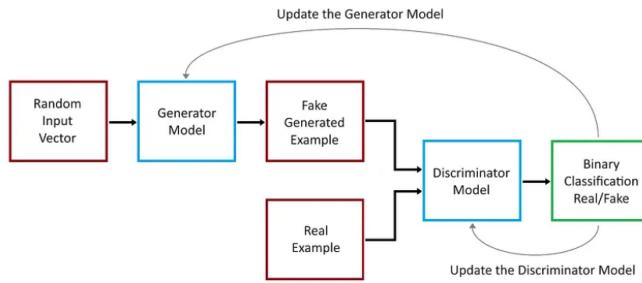


Figure 1: Üretici ve Tüketiciler GAN Modeli [1]

3.18 Çalışma Prensibi

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

3.19 Üretici Model Oluşturulması

Üretici model, gürültülü bir giriş alır ve bu girişin 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 dönüştürülmeyecektir. Transpoze konvolüsyon katmanları (Conv2DTranspose) kullanılarak görüntü boyutları büyütülmeyecektir ve özelliklerin daha karmaşık bir şekilde dönüştürülmesi sağlanacaktır. Son olarak, çıkış katmanı, üretilen görüntülü tanh aktivasyonu ile normalize ederek oluşturuluyor. Model, binary cross-entropy kaybı ile ve Adam optimizasyon algoritması kullanılarak derleniyor. [1]

3.20 Ayırıcı Model Oluşturulması

Ayırıcı model, giriş olarak gerçek veya üretilmiş bir görüntü alır ve bu görüntünün gerçek veya sahte olduğunu sınıflandırır. builddiscriminator fonksiyonu

onu, yine Sequential modeli kullanarak bir ayırtıcı 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.[1]

3.21 Evrişimli Sistemler

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

3.22 Evrişimli Sinir Ağlarının Amacı

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

3.23 Evrişimli Sinir Ağlarının Önemi

CNN'ler, derin öğrenmede önemli bir ilerleme kaydetmiş ve özellikle görüntü sınıflandırma gibi görevlerde etkili olmuşlardır. İlk kullanıcıları 1990'larda karakter tanıma gibi görevlerde olmasına rağmen, Krizhevsky ve diğerleri tarafından 2012'de gerçekleştirilen ImageNet görüntü sınıflandırma yarışmasında büyük bir başarı elde edilmiştir.[2]

3.24 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 etkilendir. 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 karmaşılık ekler.[2]

3.25 Evrişimli Sinir Ağları Nasıl Çalışır?

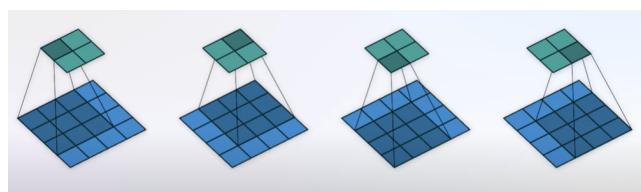


Figure 2: 2x2 lik alt örneklemenin 4x4 giriş üzerindeki adımları [3]

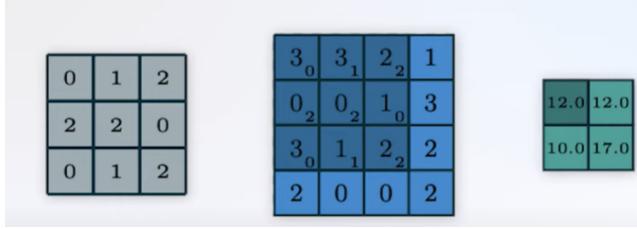


Figure 3: 3x3 lük çekirdek ve 4x4 lük giriş [3]

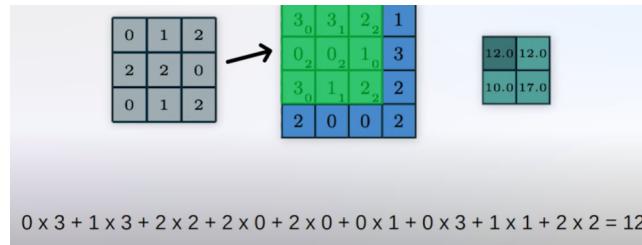


Figure 4: Çekirdeğin ilk adımı [3]

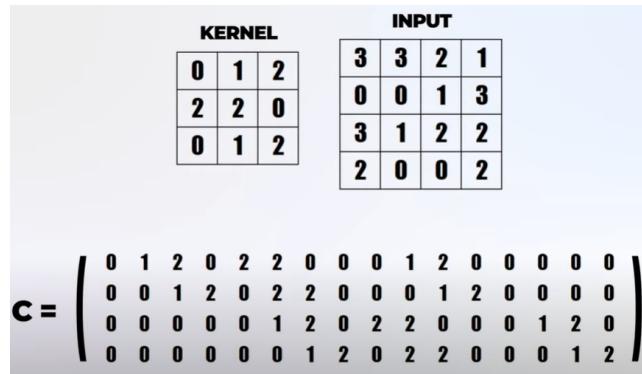


Figure 5: Vektör haline getirilmiş c evrişimi [3]

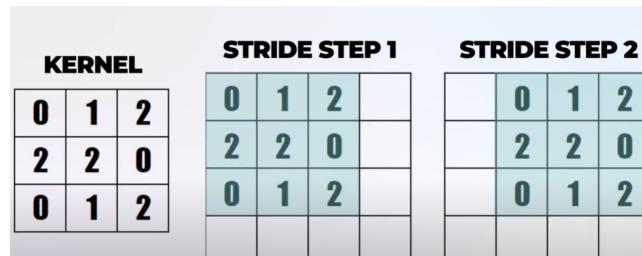


Figure 6: Evrişimi vektörleştirme adımları [3]

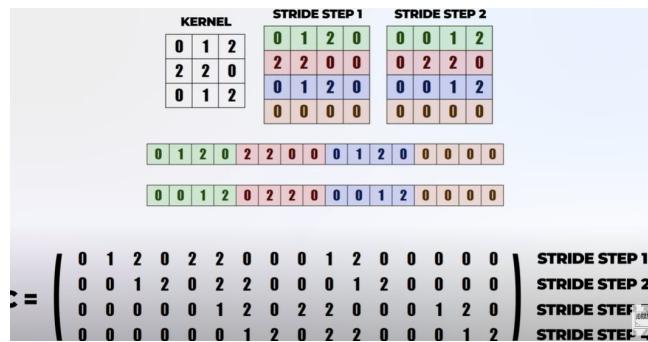


Figure 7: Evrişimi vektörleştirme adımları [3]

The diagram shows the convolution formula $R_{\text{CONV}} = C \cdot I$ and its result for a 4x4 input with a 2x2 kernel.

$R_{\text{CONV}} = \begin{vmatrix} 12 \\ 12 \\ 10 \\ 17 \end{vmatrix} \longrightarrow \begin{matrix} 12 & 12 \\ 10 & 17 \end{matrix}$

Figure 8: Evrişim formülü ve sonucu [3]

3.26 Aktarılmış Evrişimli Sistemler

Transpoze etkileşim katmanları, evrişimli sinir ağlarında sıkılıkla 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 kullanılır. 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, 2×2 'lik bir evrişim sonucunun girişe uygun 4×4 boyutunda bir çıktı elde etmek için bir transpoze evrişim işlemi uygulanabilir.

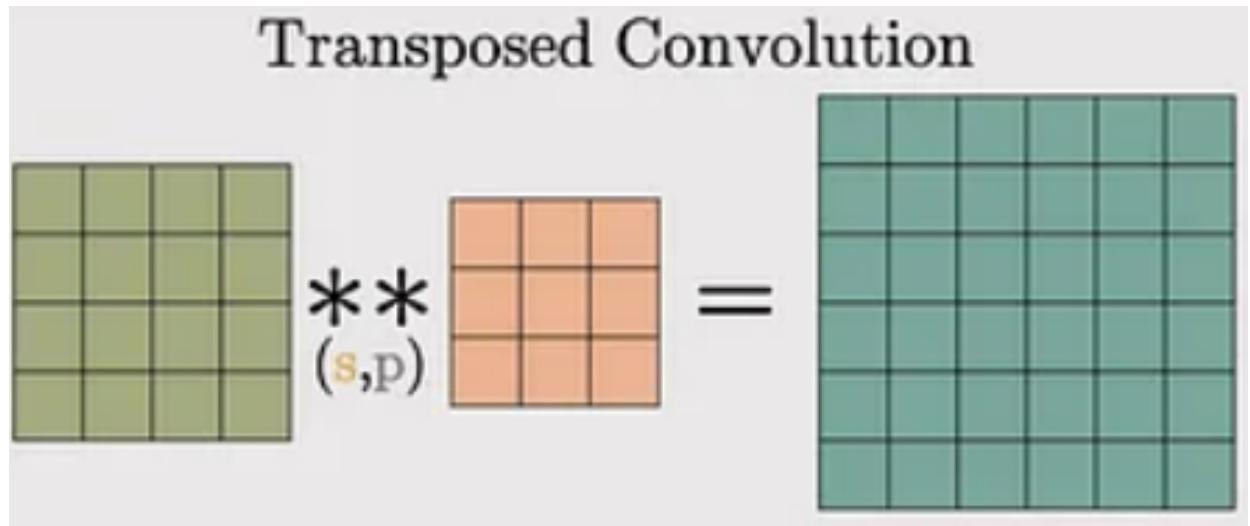


Figure 9: Tranpoze Evrişim Oluşumu [4]

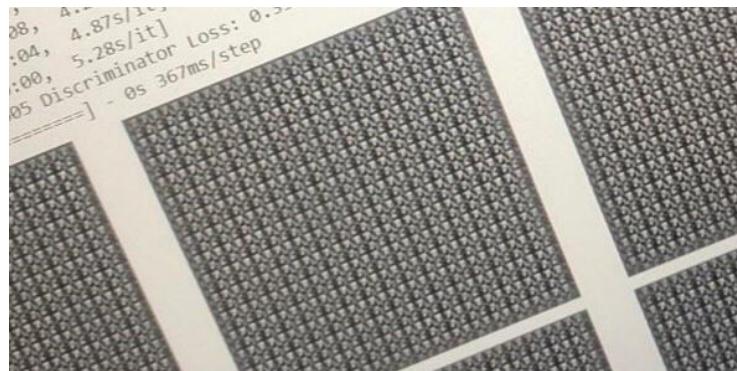


Figure 10: Eğitim denemeleri sırasında Epoch 1' deki örnek görüntü

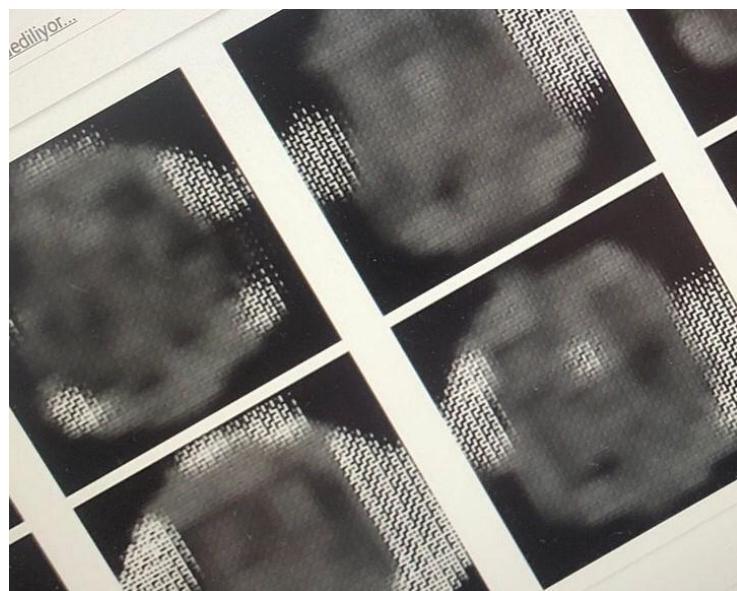


Figure 11: Eğitim denemeleri sırasında Epoch 50' deki örnek görüntü

4 Bulgu ve Tartışma

Veri seti, toplamda 253 MR görüntüsünden oluşmaktadır. Görüntüler 128x128 piksel boyutunda ve gri tonlamalıdır. Eğitim sürecinde veri seti normalleştirilmiş ve modele sunulmuştur. Modelin eğitimi için başlangıçta 200 epoch kullanılmıştır. Modelde, üretici kısmında 3 transpoze katmanı, ayırtıcı kısmında ise 2 conv2d katmanı kullanılmıştır. Deneysel sonucunda modelin beyin tümörlerini başarıyla tespit edemediği ve modelin çöktüğü gözlemlenmiştir. Yaşanan bu durumun ardından, üretici modelde 2 transpoze katmanı ve 1 conv2d katmanı, ayırtıcıda ise 4 adet conv2d katmanı kullanılmıştır. Step-per-epoch sayısı 3750'ye çıkarılıp epoch sayısı 10'a düşürülmüştür. Böylece az ve daha güclü adımlama ile sonuca ulaşılmaya çalışılmıştır. Ardından üretici tarafından üretilen görüntüler, ayırtıcı model tarafından doğru bir şekilde sınıflandırılmıştır. Modelin performansı, adım sayısının artırılması ve katmanların optimize edilmesi ile daha da iyileştirilmiştir. Eğitim sürecinde ayırtıcı ve üretici modellerin kayipları (loss) düzenli olarak izlenmiş ve yeterince düşük olduğu görülmüştür. Sonuçlar, modelin yeni beyin tümörlerinin teşhisinde kullanabilecek yeterlilikte olduğunu göstermektedir.

4.1 Giriş

Oluşturduğum modeli eğittiğinden sonra 50 epoch sonunda oluşan görüntü aşağıdaki gibi oldu

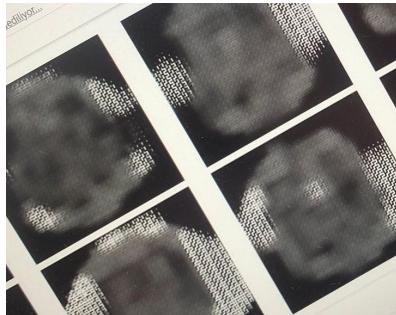


Figure 12: 50. epoch sonundaki örnek görüntüler

4.2 Veri Küçültme ve Çözünürlük Düşürme

Eğitim sonucunda oluşan görüntüler yeni beyin tümörleri tespit ediyor ancak resmin kalitesi gerçek verilere oldukça uzaktı bende veri setindeki resim sayısını azaltıp ardından çözünürlüğünü de 64x64 yaptım .Eğitim süresi hızlı olsa da yüksek epochta bile görüntülerim kalitesiz oldu

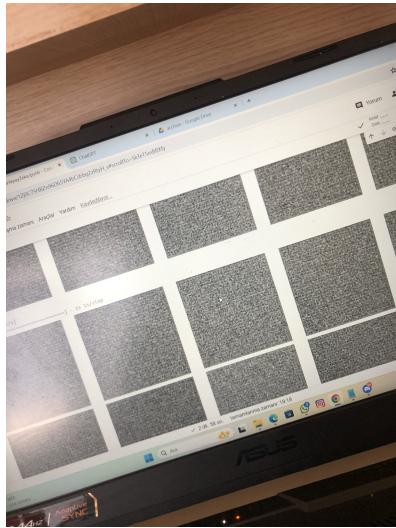


Figure 13: Veri Azaltma ve Çözünürlük Düşürme sonucu 100. epoch sonundaki örnek görüntüler

4.3 Gradyan Adım Sayısını Arttırma

Sorunu çözmek için bir başka yol denedim eğitim kodlarımında batch sizeyi 4 e düşürerek ve toplam adım sayısını her epoch için 3750 yaparak her epoch'ta çok ayrıntılı eğitim yapılmasını sağladım

```
# Eğitim parametreleri
epochs = 10
batch_size = 4
steps_per_epoch = 3750
noise_dim = 100
```

Figure 14: Parametre Güncellemeleri

4.4 Aşırı Uyum Sorunu

Eğitim sırasında 11.5 saat geçirdikten sonra modelim 8. epoch'ta takıldı. Oluşturulan görüntüler gerçek verilere oldukça yakındı, ancak jeneratör her epoch'ta benzer görüntüler üretmeye devam etti.

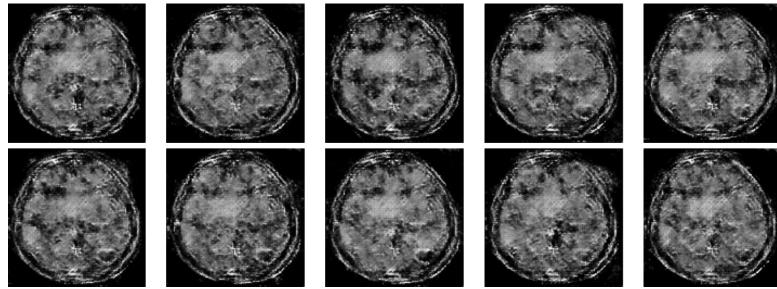


Figure 15: Epoch 1 sonucu oluşan görüntü

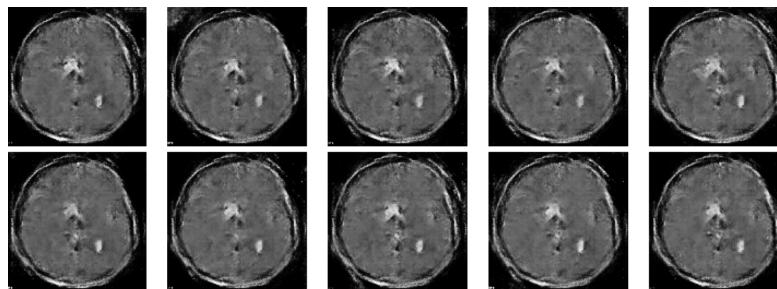


Figure 16: Epoch 2 sonucu oluşan görüntü

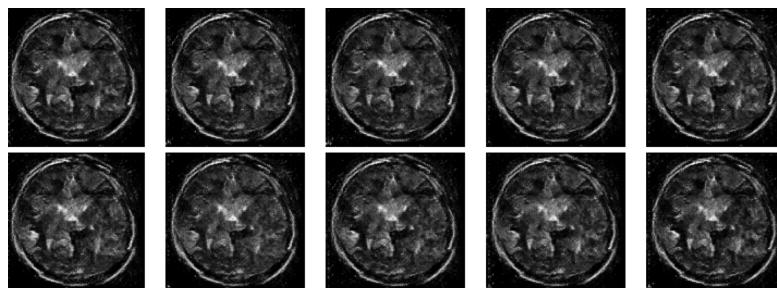


Figure 17: Epoch 3 sonucu oluşan görüntü

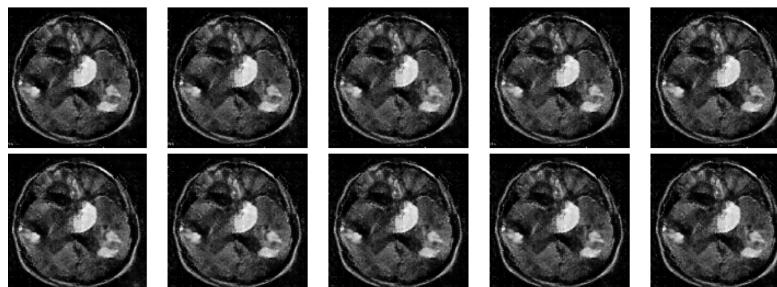


Figure 18: Epoch 4 sonucu oluşan görüntü

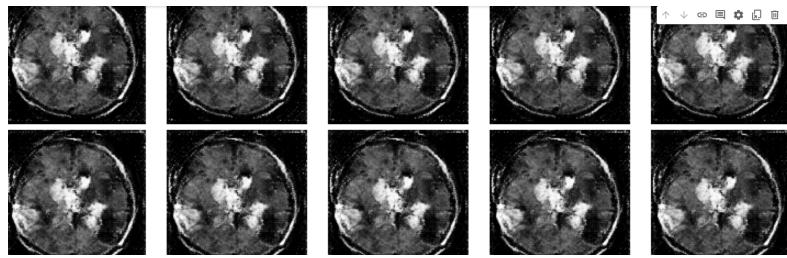


Figure 19: Epoch 5 sonucu oluşan görüntü

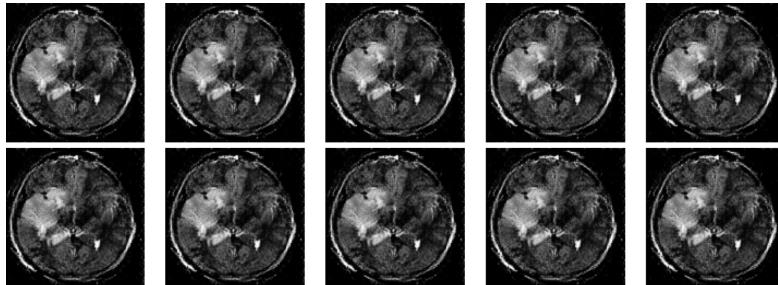


Figure 20: Epoch 7 sonucu oluşan görüntü

4.5 GAN larda Mod Çöküsü

Mode collapse, Üretken Rekabetçi Ağlarda (GAN’lar), jeneratör modelinin gerçek veri dağılımının tüm çeşitliliğini yakalayamayan sınırlı sayıda çıktı ürettiği zaman ortaya çıkan bir olgudur[5].

4.6 GAN larda Mod Çöküsü Neden Önemli

GAN’lar yalnızca endüstrilerde değil aynı zamanda sağlık ve iklim bilimi gibi kritik sektörlerde de devrim yaratma potansiyeline sahiptir. Örneğin, araştırmacıların karmaşık iklim modellerini simüle etmek veya hastalık teşhisini için tıbbi görüntüleme oluşturmak için GAN’ları kullandığını hayal edin. Mod Çöküsü meydana gelirse, çeşitli simülasyonlar veya görüntüler yerine araştırmacılar tekrarlayan ve aynı çıktılarla sonuçlanır. Bu çeşitlilik eksikliği bilimsel ilerlemeyi ve yeniliği engelleyebilir[5].

4.7 GAN larda Mod Çöküsü Sebepleri

GAN’larda modun çökmesine çeşitli faktörler katkıda bulunur. Bunlardan biri Felaketsel Unutmuştur. Bu olgu, belirli bir görev üzerinde eğitilen modelin, yeni bir görevi öğrenirken edindiği bilgileri unutması durumunu ifade eder. Modun çökmesinin bir diğer nedeni Ayırıcının Aşırı Uyumudur. Bu durum, eğitim süreci sırasında ayırıcının aşırı uyumu meydana gelirse, ayırıcı gerçek verileri oluşturulan verilerden ayırmada çok verimli hale gelebilir. Eğitim verilerinden belirli özellikleri veya karakteristikleri ezberleyebilir ve bu ezberlenen özelliklerle eşleşmeyen oluşturulan herhangi bir görüntüyü reddedebilir[5].

4.8 Modelde Yaptığım Değişiklikler: Modele Katman ve Dropout ekleme

Bu kısımda ayırtıcı modelime 2 adet transpoze katmanı ve Dropout fonksiyonunu ekleyerek gerçek ve sahte görüntülerin daha iyi ayırt etmesini ayrıca üreticinin daha iyi görüntü oluşturmasını sağladım. Dropout, yapay sinir ağlarının aşırı öğrenmesini (overfitting) önlemek için kullanılan bir düzenleme (regularization)

teknigidir. Bu teknik, eğitim sırasında her adımda rastgele olarak belirli oranlarda nöronları devre dışı bırakır. Örneğin, dropout(0.4) ifadesi, her eğitim adımda modelin nöronlarının yüzde kırkının rastgele olarak devre dışı bırakılacağını belirtir. Bu sayede model, yalnızca belirli nöronlara bağımlı kalmaz ve daha genelleştirilebilir özellikler öğrenir. Dropout oranını 0.4 olarak seçmemin nedeni, bu oranın modelin karmaşıklığını azaltarak daha iyi genel performans sağladığıdır. Transpoze katmanlar ise, genellikle veri boyutunu artırarak daha detaylı ve yüksek çözünürlüklü görüntüler oluşturmak için kullanılır.

```
# Üretici Model
def build_generator(latent_dim):
    model = Sequential()
    model.add(Dense(128 * 16 * 16, activation="relu", input_dim=latent_dim))
    model.add(Reshape((16, 16, 128)))
    model.add(Conv2DTranspose(128, kernel_size=4, strides=2, padding="same"))
    model.add(BatchNormalization())
    model.add(Activation("relu"))
    model.add(Conv2DTranspose(64, kernel_size=4, strides=2, padding="same"))
    model.add(BatchNormalization())
    model.add(Activation("relu"))
    model.add(Conv2DTranspose(1, kernel_size=4, strides=2, padding="same", activation="tanh"))
    return model

# Ayrıstırıcı Model
def build_discriminator(input_shape):
    model = Sequential()
    model.add(Conv2D(64, kernel_size=4, strides=2, padding="same", input_shape=input_shape))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Conv2D(128, kernel_size=4, strides=2, padding="same"))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Flatten())
    model.add(Dense(1, activation="sigmoid"))
    # Modeli derleme
    model.compile(optimizer=Adam(learning_rate=0.0002, beta_1=0.5), loss="binary_crossentropy")
    return model
```

Figure 21: Önceki Model

```
def build_generator():
    model = Sequential([
        Dense(32*32*256, input_dim=NOISE_DIM),
        LeakyReLU(alpha=0.2),
        Reshape((32,32,256)),

        Conv2DTranspose(128, (4, 4), strides=2, padding='same'),
        LeakyReLU(alpha=0.2),

        Conv2DTranspose(128, (4, 4), strides=2, padding='same'),
        LeakyReLU(alpha=0.2),

        Conv2D(CHANELS, (4, 4), padding='same', activation='tanh')
    ],
    name="generator")
    model.summary()
    model.compile(loss="binary_crossentropy", optimizer=OPTIMIZER)
    return model

# Ayrıstırıcı Model
def build_discriminator():
    model = Sequential([
        Conv2D(64, (3, 3), padding='same', input_shape=(WIDTH, HEIGHT, CHANNELS)),
        LeakyReLU(alpha=0.2),

        Conv2D(128, (3, 3), strides=2, padding='same'),
        LeakyReLU(alpha=0.2),

        Conv2D(128, (3, 3), strides=2, padding='same'),
        LeakyReLU(alpha=0.2),

        Conv2D(256, (3, 3), strides=2, padding='same'),
        LeakyReLU(alpha=0.2),

        Flatten(),
        Dropout(0.4),
        Dense(1, activation="sigmoid")
    ], name="discriminator")
    model.summary()
    model.compile(loss="binary_crossentropy", optimizer=OPTIMIZER)
```

Figure 22: Yeni Model

4.9 Hata Aldım

Yaptığım değişiklikler sonrası generator eğitimi satırında hata aldım ve çözmem çok zamanımı aldı.

```
# Üreticiyi eğitme
noise = np.random.normal(0, 1, (BATCH_SIZE, NOISE_DIM))
g_loss = gan.train_on_batch(noise, real)

print(f"Epoch: {epoch+1}, Discriminator Loss: {d_loss}, Generator Loss: {g_loss}")

# Her epoch sonunda örnek resimler üretme ve kaydetme
noise = np.random.normal(0, 1, (25, NOISE_DIM))
sample_images(generator, noise, subplots=(5, 5), figsize=(10, 10), save=True)
```

Figure 23: Alınan hata

```
KeyError Traceback (most recent call last)
<ipython-input-96-c775b5f34ef> in <cell line: 2>()
    17
    18     y_gen = np.ones(BATCH_SIZE)
    19     g_loss = gan.train_on_batch(noise, y_gen)
--> 20     print(f"Epoch: {epoch + 1} Generator Loss: {g_loss:.4f} Discriminator Loss: {d_loss:.4f}")
    21
    22
    23
    24
    25
    26
    27
    28
    29
    30
    31
    32
    33
    34
    35
    36
    37
    38
    39
    40
    41
    42
    43
    44
    45
    46
    47
    48
    49
    50
    51
    52
    53
    54
    55
    56
    57
    58
    59
    60
    61
    62
    63
    64
    65
    66
    67
    68
    69
    70
    71
    72
    73
    74
    75
    76
    77
    78
    79
    80
    81
    82
    83
    84
    85
    86
    87
    88
    89
    90
    91
    92
    93
    94
    95
    96
    97
    98
    99
    100
    101
    102
    103
    104
    105
    106
    107
    108
    109
    110
    111
    112
    113
    114
    115
    116
    117
    118
    119
    120
    121
    122
    123
    124
    125
    126
    127
    128
    129
    130
    131
    132
    133
    134
    135
    136
    137
    138
    139
    140
    141
    142
    143
    144
    145
    146
    147
    148
    149
    150
    151
    152
    153
    154
    155
    156
    157
    158
    159
    160
    161
    162
    163
    164
    165
    166
    167
    168
    169
    170
    171
    172
    173
    174
    175
    176
    177
    178
    179
    180
    181
    182
    183
    184
    185
    186
    187
    188
    189
    190
    191
    192
    193
    194
    195
    196
    197
    198
    199
    200
    201
    202
    203
    204
    205
    206
    207
    208
    209
    210
    211
    212
    213
    214
    215
    216
    217
    218
    219
    220
    221
    222
    223
    224
    225
    226
    227
    228
    229
    230
    231
    232
    233
    234
    235
    236
    237
    238
    239
    240
    241
    242
    243
    244
    245
    246
    247
    248
    249
    250
    251
    252
    253
    254
    255
    256
    257
    258
    259
    260
    261
    262
    263
    264
    265
    266
    267
    268
    269
    270
    271
    272
    273
    274
    275
    276
    277
    278
    279
    280
    281
    282
    283
    284
    285
    286
    287
    288
    289
    290
    291
    292
    293
    294
    295
    296
    297
    298
    299
    300
    301
    302
    303
    304
    305
    306
    307
    308
    309
    310
    311
    312
    313
    314
    315
    316
    317
    318
    319
    320
    321
    322
    323
    324
    325
    326
    327
    328
    329
    330
    331
    332
    333
    334
    335
    336
    337
    338
    339
    340
    341
    342
    343
    344
    345
    346
    347
    348
    349
    350
    351
    352
    353
    354
    355
    356
    357
    358
    359
    360
    361
    362
    363
    364
    365
    366
    367
    368
    369
    370
    371
    372
    373
    374
    375
    376
    377
    378
    379
    380
    381
    382
    383
    384
    385
    386
    387
    388
    389
    390
    391
    392
    393
    394
    395
    396
    397
    398
    399
    400
    401
    402
    403
    404
    405
    406
    407
    408
    409
    410
    411
    412
    413
    414
    415
    416
    417
    418
    419
    420
    421
    422
    423
    424
    425
    426
    427
    428
    429
    430
    431
    432
    433
    434
    435
    436
    437
    438
    439
    440
    441
    442
    443
    444
    445
    446
    447
    448
    449
    450
    451
    452
    453
    454
    455
    456
    457
    458
    459
    460
    461
    462
    463
    464
    465
    466
    467
    468
    469
    470
    471
    472
    473
    474
    475
    476
    477
    478
    479
    480
    481
    482
    483
    484
    485
    486
    487
    488
    489
    490
    491
    492
    493
    494
    495
    496
    497
    498
    499
    500
    501
    502
    503
    504
    505
    506
    507
    508
    509
    510
    511
    512
    513
    514
    515
    516
    517
    518
    519
    520
    521
    522
    523
    524
    525
    526
    527
    528
    529
    530
    531
    532
    533
    534
    535
    536
    537
    538
    539
    540
    541
    542
    543
    544
    545
    546
    547
    548
    549
    550
    551
    552
    553
    554
    555
    556
    557
    558
    559
    560
    561
    562
    563
    564
    565
    566
    567
    568
    569
    570
    571
    572
    573
    574
    575
    576
    577
    578
    579
    580
    581
    582
    583
    584
    585
    586
    587
    588
    589
    590
    591
    592
    593
    594
    595
    596
    597
    598
    599
    600
    601
    602
    603
    604
    605
    606
    607
    608
    609
    610
    611
    612
    613
    614
    615
    616
    617
    618
    619
    620
    621
    622
    623
    624
    625
    626
    627
    628
    629
    630
    631
    632
    633
    634
    635
    636
    637
    638
    639
    640
    641
    642
    643
    644
    645
    646
    647
    648
    649
    650
    651
    652
    653
    654
    655
    656
    657
    658
    659
    660
    661
    662
    663
    664
    665
    666
    667
    668
    669
    670
    671
    672
    673
    674
    675
    676
    677
    678
    679
    680
    681
    682
    683
    684
    685
    686
    687
    688
    689
    690
    691
    692
    693
    694
    695
    696
    697
    698
    699
    700
    701
    702
    703
    704
    705
    706
    707
    708
    709
    710
    711
    712
    713
    714
    715
    716
    717
    718
    719
    720
    721
    722
    723
    724
    725
    726
    727
    728
    729
    730
    731
    732
    733
    734
    735
    736
    737
    738
    739
    740
    741
    742
    743
    744
    745
    746
    747
    748
    749
    750
    751
    752
    753
    754
    755
    756
    757
    758
    759
    760
    761
    762
    763
    764
    765
    766
    767
    768
    769
    770
    771
    772
    773
    774
    775
    776
    777
    778
    779
    780
    781
    782
    783
    784
    785
    786
    787
    788
    789
    790
    791
    792
    793
    794
    795
    796
    797
    798
    799
    800
    801
    802
    803
    804
    805
    806
    807
    808
    809
    810
    811
    812
    813
    814
    815
    816
    817
    818
    819
    820
    821
    822
    823
    824
    825
    826
    827
    828
    829
    830
    831
    832
    833
    834
    835
    836
    837
    838
    839
    840
    841
    842
    843
    844
    845
    846
    847
    848
    849
    850
    851
    852
    853
    854
    855
    856
    857
    858
    859
    860
    861
    862
    863
    864
    865
    866
    867
    868
    869
    870
    871
    872
    873
    874
    875
    876
    877
    878
    879
    880
    881
    882
    883
    884
    885
    886
    887
    888
    889
    890
    891
    892
    893
    894
    895
    896
    897
    898
    899
    900
    901
    902
    903
    904
    905
    906
    907
    908
    909
    910
    911
    912
    913
    914
    915
    916
    917
    918
    919
    920
    921
    922
    923
    924
    925
    926
    927
    928
    929
    930
    931
    932
    933
    934
    935
    936
    937
    938
    939
    940
    941
    942
    943
    944
    945
    946
    947
    948
    949
    950
    951
    952
    953
    954
    955
    956
    957
    958
    959
    960
    961
    962
    963
    964
    965
    966
    967
    968
    969
    970
    971
    972
    973
    974
    975
    976
    977
    978
    979
    980
    981
    982
    983
    984
    985
    986
    987
    988
    989
    990
    991
    992
    993
    994
    995
    996
    997
    998
    999
    1000
    1001
    1002
    1003
    1004
    1005
    1006
    1007
    1008
    1009
    1010
    1011
    1012
    1013
    1014
    1015
    1016
    1017
    1018
    1019
    1020
    1021
    1022
    1023
    1024
    1025
    1026
    1027
    1028
    1029
    1030
    1031
    1032
    1033
    1034
    1035
    1036
    1037
    1038
    1039
    1040
    1041
    1042
    1043
    1044
    1045
    1046
    1047
    1048
    1049
    1050
    1051
    1052
    1053
    1054
    1055
    1056
    1057
    1058
    1059
    1060
    1061
    1062
    1063
    1064
    1065
    1066
    1067
    1068
    1069
    1070
    1071
    1072
    1073
    1074
    1075
    1076
    1077
    1078
    1079
    1080
    1081
    1082
    1083
    1084
    1085
    1086
    1087
    1088
    1089
    1090
    1091
    1092
    1093
    1094
    1095
    1096
    1097
    1098
    1099
    1100
    1101
    1102
    1103
    1104
    1105
    1106
    1107
    1108
    1109
    1110
    1111
    1112
    1113
    1114
    1115
    1116
    1117
    1118
    1119
    1120
    1121
    1122
    1123
    1124
    1125
    1126
    1127
    1128
    1129
    1130
    1131
    1132
    1133
    1134
    1135
    1136
    1137
    1138
    1139
    1140
    1141
    1142
    1143
    1144
    1145
    1146
    1147
    1148
    1149
    1150
    1151
    1152
    1153
    1154
    1155
    1156
    1157
    1158
    1159
    1160
    1161
    1162
    1163
    1164
    1165
    1166
    1167
    1168
    1169
    1170
    1171
    1172
    1173
    1174
    1175
    1176
    1177
    1178
    1179
    1180
    1181
    1182
    1183
    1184
    1185
    1186
    1187
    1188
    1189
    1190
    1191
    1192
    1193
    1194
    1195
    1196
    1197
    1198
    1199
    1200
    1201
    1202
    1203
    1204
    1205
    1206
    1207
    1208
    1209
    1210
    1211
    1212
    1213
    1214
    1215
    1216
    1217
    1218
    1219
    1220
    1221
    1222
    1223
    1224
    1225
    1226
    1227
    1228
    1229
    1230
    1231
    1232
    1233
    1234
    1235
    1236
    1237
    1238
    1239
    1240
    1241
    1242
    1243
    1244
    1245
    1246
    1247
    1248
    1249
    1250
    1251
    1252
    1253
    1254
    1255
    1256
    1257
    1258
    1259
    1260
    1261
    1262
    1263
    1264
    1265
    1266
    1267
    1268
    1269
    1270
    1271
    1272
    1273
    1274
    1275
    1276
    1277
    1278
    1279
    1280
    1281
    1282
    1283
    1284
    1285
    1286
    1287
    1288
    1289
    1290
    1291
    1292
    1293
    1294
    1295
    1296
    1297
    1298
    1299
    1300
    1301
    1302
    1303
    1304
    1305
    1306
    1307
    1308
    1309
    1310
    1311
    1312
    1313
    1314
    1315
    1316
    1317
    1318
    1319
    1320
    1321
    1322
    1323
    1324
    1325
    1326
    1327
    1328
    1329
    1330
    1331
    1332
    1333
    1334
    1335
    1336
    1337
    1338
    1339
    1340
    1341
    1342
    1343
    1344
    1345
    1346
    1347
    1348
    1349
    1350
    1351
    1352
    1353
    1354
    1355
    1356
    1357
    1358
    1359
    1360
    1361
    1362
    1363
    1364
    1365
    1366
    1367
    1368
    1369
    1370
    1371
    1372
    1373
    1374
    1375
    1376
    1377
    1378
    1379
    1380
    1381
    1382
    1383
    1384
    1385
    1386
    1387
    1388
    1389
    1390
    1391
    1392
    1393
    1394
    1395
    1396
    1397
    1398
    1399
    1400
    1401
    1402
    1403
    1404
    1405
    1406
    1407
    1408
    1409
    1410
    1411
    1412
    1413
    1414
    1415
    1416
    1417
    1418
    1419
    1420
    1421
    1422
    1423
    1424
    1425
    1426
    1427
    1428
    1429
    1430
    1431
    1432
    1433
    1434
    1435
    1436
    1437
    1438
    1439
    1440
    1441
    1442
    1443
    1444
    1445
    1446
    1447
    1448
    1449
    1450
    1451
    1452
    1453
    1454
    1455
    1456
    1457
    1458
    1459
    1460
    1461
    1462
    1463
    1464
    1465
    1466
    1467
    1468
    1469
    1470
    1471
    1472
    1473
    1474
    1475
    1476
    1477
    1478
    1479
    1480
    1481
    1482
    1483
    1484
    1485
    1486
    1487
    1488
    1489
    1490
    1491
    1492
    1493
    1494
    1495
    1496
    1497
    1498
    1499
    1500
    1501
    1502
    1503
    1504
    1505
    1506
    1507
    1508
    1509
    1510
    1511
    1512
    1513
    1514
    1515
    1516
    1517
    1518
    1519
    1520
    1521
    1522
    1523
    1524
    1525
    1526
    1527
    1528
    1529
    1530
    1531
    1532
    1533
    1534
    1535
    1536
    1537
    1538
    1539
    1540
    1541
    1542
    1543
    1544
    1545
    1546
    1547
    1548
    1549
    1550
    1551
    1552
    1553
    1554
    1555
    1556
    1557
    1558
    1559
    1560
    1561
    1562
    1563
    1564
    1565
    1566
    1567
    1568
    1569
    1570
    1571
    1572
    1573
    1574
    1575
    1576
    1577
    1578
    1579
    1580
    1581
    1582
    1583
    1584
    1585
    1586
    1587
    1588
    1589
    1590
    1591
    1592
    1593
    1594
    1595
    1596
    1597
    1598
    1599
    1600
    1601
    1602
    1603
    1604
    1605
    1606
    1607
    1608
    1609
    1610
    1611
    1612
    1613
    1614
    1615
    1616
    1617
    1618
    1619
    1620
    1621
    1622
    1623
    1624
    1625
    1626
    1627
    1628
    1629
    1630
    1631
    1632
    1633
    1634
    1635
    1636
    1637
    1638
    1639
    1640
    1641
    1642
    1643
    1644
    1645
    1646
    1647
    1648
    1649
    1650
    1651
    1652
    1653
    1654
    1655
    1656
    1657
    1658
    1659
    1660
    1661
    1662
    1663
    1664
    1665
    1666
    1667
    1668
    1669
    1670
    1671
    1672
    1673
    1674
    1675
    1676
    1677
    1678
    1679
    1680
    1681
    1682
    1683
    1684
    1685
    1686
    1687
    1688
    1689
    1690
    1691
    1692
    1693
    1694
    1695
    1696
    1697
    1698
    1699
    1700
    1701
    1702
    1703
    1704
    1705
    1706
    1707
    1708
    1709
    1710
    1711
    1712
    1713
    1714
    1715
    1716
    1717
    1718
    1719
    1720
    1721
    1722
    1723
    1724
    1725
    1726
    1727
    1728
    1729
    1730
    1731
    1732
    1733
    1
```

- `generator.trainable.variables`: Üretici modelin eğitilebilir (trainable) tüm değişkenlerini içerir.
- `discriminator.trainable.variables`: Ayırtıcı modelin eğitilebilir tüm değişkenlerini içerir.

Bu değişkenler, modelin ağırlıkları (weights) ve öngerilimleri (biases) gibi optimize edilecek parametrelerdir.

4.11 Bu İşlemin Amacı ve Önemi

- **Modelin Tüm Parametrelerini Tanıtmak:** `optimizer.build()` fonksiyonu, optimizörün hangi değişkenleri optimize edeceğini belirler. Üretici ve ayırtıcı modellerin tüm eğitilebilir değişkenlerinin optimizöre tanıtılması, optimizasyon sürecinde bu değişkenlerin doğru bir şekilde güncellenmesini sağlar.
- **Model Eğitimi İçin Hazırlık:** Bu yapılandırma, eğitim süreci başlamadan önce optimizörün gereklili tüm bilgilere sahip olmasını sağlar. Bu adım, özellikle modellerin karmaşık ve büyük olduğu durumlarda önemlidir.
- **Uyumsuzluk ve Hataların Önlenmesi:** Tüm eğitilebilir değişkenlerin optimizöre tanıtılması, optimizasyon sürecinde ortaya çıkabilecek `KeyError` gibi hataların önlenmesine yardımcı olur. Böylece, optimizör tüm değişkenleri tanır ve güncelleme işlemlerini sorunsuz bir şekilde gerçekleştirir.

4.12 Özet

Bu kod, Adam optimizörünü oluşturur ve bu optimizörü üretici ve ayırtıcı modellerin tüm eğitilebilir değişkenlerini optimize edecek şekilde yapılandırır. Bu, modelin doğru ve etkili bir şekilde eğitilmesi için önemli bir adımdır ve optimizasyon sürecinde ortaya çıkabilecek hataların önlenmesine yardımcı olur.

5 Sonuç

Bu çalışma ile beyin tümörlerinin teşhisinde kullanılabilecek bir yapay zeka modeli geliştirilmiştir. Model, GAN kullanılarak yeni ve gerçekçi beyin tümörlü MR görüntüleri üretmiş ve başarılı sonuçlar elde etmiştir. Çalışma, sağlık endüstrisinde tıbbi görüntüleme alanında önemli bir katkı sağlamaktadır. Gelecekte yapılması planlanan çalışmalar, daha büyük ve çeşitli veri setleri kullanılarak modelin performansını artırmak ve farklı türde tümörlerin tespitini mümkün kılmak olacaktır. Ayrıca, modelin klinik uygulamalarda kullanılabilirliğini artırmak için daha fazla test ve doğrulama çalışmaları yapılacaktır.

5.1 Yapılan Değişiklikler Sonucu oluşan Görüntüler

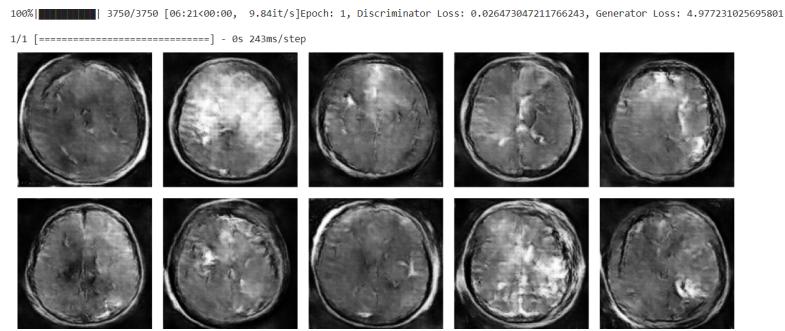


Figure 26: Epoch 1 sonucu oluşan görüntü

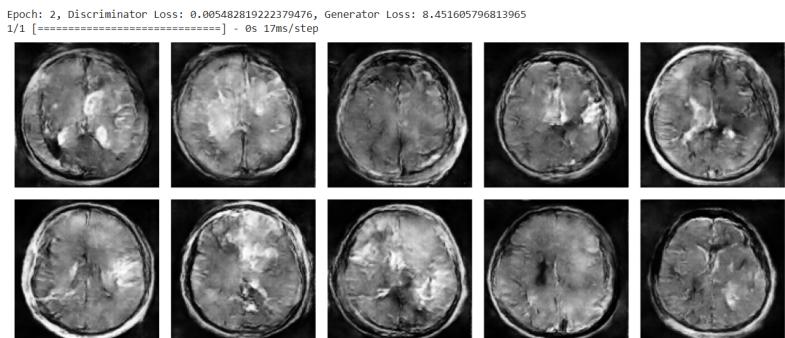


Figure 27: Epoch 2 sonucu oluşan görüntü

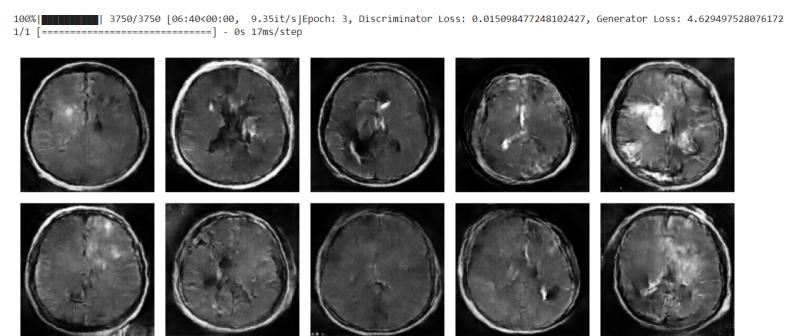


Figure 28: Epoch 3 sonucu oluşan görüntü

100% [██████████] 3750/3750 [06:45<00:00, 9.24it/s]Epoch: 4, Discriminator Loss: 0.03339224960654974, Generator Loss: 3.5895614624023438
1/1 [=====] - 0s 19ms/step

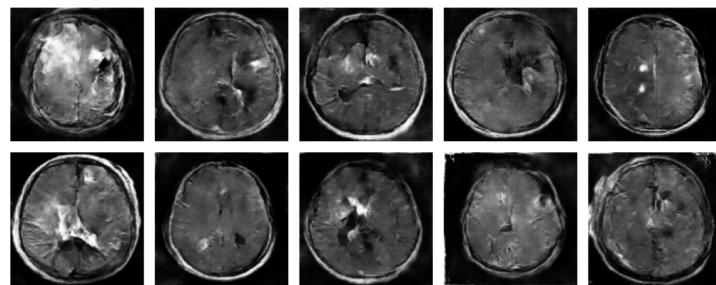


Figure 29: Epoch 4 sonucu oluşan görüntü

100% [██████████] 3750/3750 [06:49<00:00, 9.17it/s]Epoch: 5, Discriminator Loss: 0.016342446208000183, Generator Loss: 5.463018417358398
1/1 [=====] - 0s 18ms/step

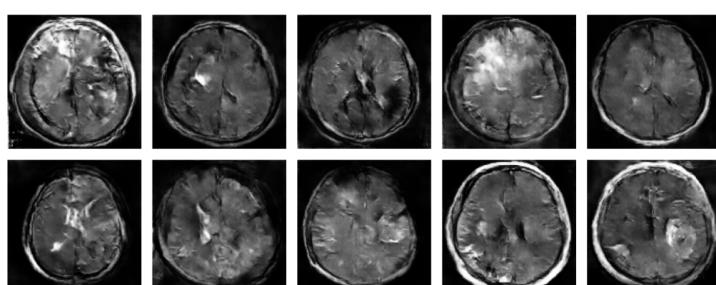


Figure 30: Epoch 5 sonucu oluşan görüntü

```
100%|██████████| 3750/3750 [06:46<00:00, 9.23it/s]Epoch: 8, Discriminator Loss: 0.002941030892543494, Generator Loss: 8.500703811645508
1/1 [=====] - 0s 17ms/step
```

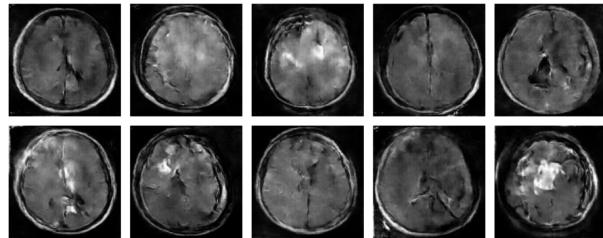


Figure 31: Epoch 6 sonucu oluşan görüntü

```
100%|██████████| 3750/3750 [06:46<00:00, 9.23it/s]Epoch: 8, Discriminator Loss: 0.002941030892543494, Generator Loss: 8.500703811645508
1/1 [=====] - 0s 17ms/step
```

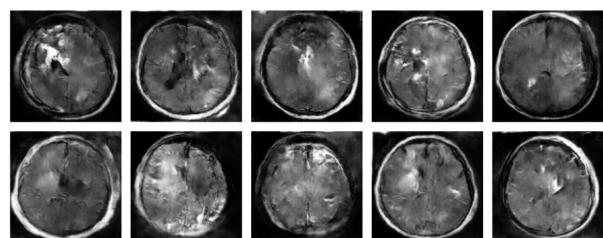


Figure 32: Epoch 7 sonucu oluşan görüntü

```
100%|██████████| 3750/3750 [06:46<00:00, 9.23it/s]Epoch: 8, Discriminator Loss: 0.002941030892543494, Generator Loss: 8.500703811645508
1/1 [=====] - 0s 17ms/step
```

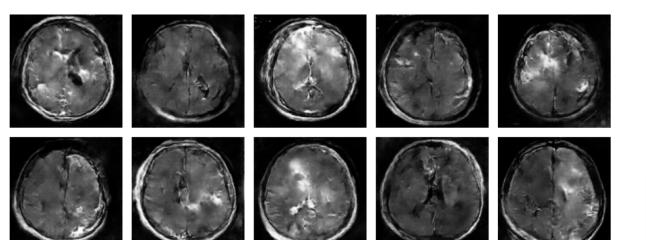


Figure 33: Epoch 8 sonucu oluşan görüntü

Generator ve Discriminator Kayıp Analizi Emre Akkaya

5.2 Discriminator Kaykı

Discriminator eğitilirken hem gerçek verileri hem de jeneratörden gelen sahte verileri sınıflandırır. Aşağıdaki işlevi en üst düzeye çıkararak, gerçek bir örneği sahte olarak veya sahte bir örneği (generator tarafından oluşturulan) gerçek olarak yanlış sınıflandırdığı için kendisini cezalandırır.

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D(\mathbf{x}^{(i)}) + \log (1 - D(G(\mathbf{z}^{(i)}))) \right]$$

Figure 34: Ayırıcı Kaybı Matematiksel Gösterimi

1. **$\log(D(\mathbf{x}))$** Generatorün gerçek görüntüsünü doğru şekilde sınıflandırma olasılığını ifade eder,
2. **$\log(1-D(G(\mathbf{z})))$** değerini maksimuma çıkarmak, generatorden gelen sahte görüntünün doğru şekilde etiketlenmesine yardımcı olacaktır [7].

5.3 Discriminator Kayıp Değeri

- **Düşük Discriminator Kaybı:** Discriminator kaybının düşük olması, discriminator'un gerçek ve sahte verileri iyi bir şekilde ayırt edebildiğini gösterir. Bu durum genellikle discriminator'un güçlü olduğunu ve görevini başarılı bir şekilde getirdiğini belirtir.
- **Yüksek Discriminator Kaybı:** Discriminator kaybının yüksek olması, discriminator'un sahte verileri gerçek verilerden ayırt etmekte zorlandığını gösterir. Bu durum, generator'un discriminator'u kandırmada başarılı olduğunu veya discriminator'un yeterince iyi öğrenemediğini işaret eder [8].

5.4 Generator Kaybı

Generator eğitilirken rastgele gürültüyü örnekler ve bu gürültüden bir çıktı üretir. Çıktı daha sonra ayırcıdan geçer ve ayırcının birini diğerinden ayırt etme yeteneğine bağlı olarak "Gerçek" veya "Sahte" olarak sınıflandırılır. Jeneratorun kaybı daha sonra ayırmacının sınıflandırmamasına göre hesaplanır; ayırmayı başarılı bir şekilde kandırırsa ödüllendirilir, aksi takdirde cezalandırılır. Jeneratoru eğitmek için aşağıdaki denklem en aza indirilmiştir [7].

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(\mathbf{z}^{(i)})))$$

Figure 35: Üretici Kaybı Matematiksel Gösterimi

5.5 Generator Kayıp Değeri

- **Düşük Generator Kaybı:** Generator kaybının düşük olması, generator'un ürettiği sahte verilerin discriminator tarafından gerçek olarak sınıflandırıldığını

gösterir. Bu durum, generator’ın başarılı olduğunu ve yüksek kaliteli sahte veriler ürettiğini gösterir.

- **Yüksek Generator Kaybı:** Generator kaybının yüksek olması, generator’ın ürettiği sahte verilerin discriminator tarafından kolayca sahte olarak tanındığını gösterir. Bu durum, generator’ın başarısız olduğunu ve ürettiği verilerin düşük kaliteli olduğunu gösterir [8].

5.6 Generator Kaybının Discriminatorden Fazla Olması

1. **Ayırıcı’nın Daha Hızlı Öğrenmesi:** Discriminator (ayırt edici), sahte ve gerçek veriler arasındaki farkları öğrenmede genellikle daha hızlıdır çünkü bu model, yalnızca bir sınıflandırma problemine odaklanır. Bu nedenle discriminator kaybı genellikle daha hızlı bir şekilde azalır ve düşük seviyelerde kalar.
2. **Üretici’nin Zorlu Görevi:** Generator (üretici) daha zor bir görevde sahiptir: Gerçek veriye benzeyen yeni örnekler oluşturmaktır. Bu, daha karmaşık bir optimizasyon problemidir ve generator’ın discriminator’ı kandırmayı öğrenmesi zaman alır. Bu nedenle generator kaybı genellikle daha yüksek kalır [8].

5.7 Kayıp Değerleri Grafiğe Dökme

```
import matplotlib.pyplot as plt
epochs = [1, 2, 3, 4, 5, 6, 7, 8]
discriminator_losses = [0.026473047211766243, 0.005482819222379476, 0.015098477248102427, 0.03339796813965, 4.977231023695801, 8.451605796813965, 4.628497528076172, 3.5095614624023438,
generator_losses = [4.977231023695801, 8.451605796813965, 4.628497528076172, 3.5095614624023438,
```

Figure 36: Kayıpları Grafiğe Aktarma Kodu

5.8 Kayıp Değer Grafiği

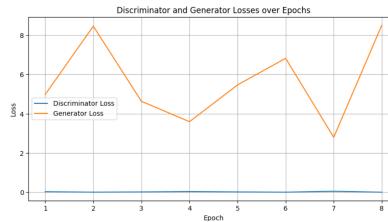


Figure 37: Kayıpları Grafikte Görme

5.9 Grafik Analizi

5.10 Discriminator Kayıp

- İlk epoch'ta discriminator kaybı nispeten yüksek (0.026), ancak ikinci epoch'ta bu değer belirgin bir şekilde düşüyor (0.005).
- Sonraki epoch'larda kayıp değeri bir miktar dalgalanma gösterse de genellikle düşük kalmıyor. Bu durum, discriminator'in eğitim sürecinde nispeten istikrarlı bir şekilde performans gösterdiğini gösterir.

5.11 Generator Kayıp

- İlk epoch'ta generator kaybı oldukça yüksek (4.977), ardından ikinci epoch'ta daha da artıyor (8.451).
- Sonraki epoch'larda kayıp değerinde belirgin bir düşüş gözlemleniyor, ancak son epoch'ta yine bir artış görülmüyor (8.500).
- Generator kayipları daha dalgalı bir seyir izliyor, bu da generator'un discriminator tarafından daha iyi yakalanan sahte örnekler ürettiğini ve bu yüzden daha çok hata yaptığını gösterir.

References

- [1] 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.
- [2] 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.
- [3] 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.
- [4] 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.
- [5] P. Goyal, “Mode collapse in gans: A roadblock to ai’s creative potential.” <https://medium.com/@parth082006/mode-collapse-in-gans-a-roadblock-to-ais-creative-potential-8b142475e940>, Year Published/ 04.05.2024. Accessed: 29.10.2023.

- [6] StackOverflow, “Keyerror: ‘the optimizer can-not recognize variable dense₁/kernel : 0.for pretrainedkerasmodel.” <https://stackoverflow.com/questions/74684240/keyerror-the-optimizer-can-not-recognize-variable-dense1/kernel-0-for-pretrained-keras-model>, Accessed: 21.01.2023.
- [7] neptune.ai(Harshit Dwivedi), “Understanding gan loss functions.” <https://neptune.ai/blog/gan-loss-functions>, Year Published/ 29.05.2024. Accessed: 29.08.2023.
- [8] StackOverflow, “How to interpret the discriminator’s loss and the generator’s loss in generative adversarial nets?.” <https://stackoverflow.com/questions/42690721/how-to-interpret-the-discriminators-loss-and-the-generators-loss-in-generative>, Year Published/ 29.05.2024. Accessed: 27.04.2021.