
BURSA TEKNİK ÜNİVERSİTESİ

MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

SEMİNER DERSİ PROJESİ RAPOR

Hümeyra ÇİMEN

# GAN Modeli Sentetik Veri Oluşturma

## Önsöz:

Bu makale, Bursa Teknik Üniversitesi 3. sınıf Bilgisayar Mühendisliği Seminer Dersinin proje çalışması kapsamında gerçekleştirilen GAN ile sentetik veri geliştirme projesini sunmaktadır. Bu proje, günümüzde giderek daha önem kazanan yapay zekâ ve makine öğrenmesi alanlarında veri eksikliği veya sınırlı veriye sahip olma sorununa çözüm sağlamayı hedeflemektedir.

Geliştirilen proje, Generative Adversarial Networks (GAN) adı verilen bir yapay sinir ağı modelini temel alır. GAN, sentetik veri üretmek için birçok potansiyel kullanım alanı sunan güçlü bir teknik olarak kabul edilmektedir. Bu çalışmada, GAN'ın kullanımıyla sentetik veri setleri oluşturarak gerçek veri setlerinin geniş bir veri kümeleriyle makine öğrenme modellerinin, görüntü işleme teknolojilerinin ve verinin kullanıldığı her alandaki modellerin performansını artırmaya yönelik çalışmaları konu almakta.

Makalede, sentetik veri geliştirme projesinin temel amacı, kullanılan yöntemler ve elde edilen sonuçlar detaylı bir şekilde ele alınmaktadır. Ayrıca, GAN'ın nasıl çalıştığı, sentetik veri üretme süreci ve proje sürecinde karşılaşılan zorluklar da makalede yer almaktadır.

Bu çalışma, günümüzün madeni verinin ne kadar elde edilmesini, temizlenmesini kısacası kullanılacağı modele uygun hale getirmek için sarfedilen enerji ve zaman kaybını önlemek ve model başarımını arttırmak için kullanılabilen sentetik verilerin bir alanı olan Gan modeli ile veri üretmeyi amaçlamaktadır.

Bu makalenin, bilgisayar mühendisliği alanında araştırma yapmak isteyen veya sentetik veri geliştirme konusunda ilgi duyan herkese faydalı olacağını umuyorum. Projemin, öğrencilerimize akademik gelişimlerinde bir adım daha ileri gitme fırsatı sunacağına inanıyor ve bu proje sürecinin oluşmasına katkı sağlayan başta bölüm başkanımız Turgay Tugay BİLGİN hocama teşekkür ediyorum.

HÜMEYRA ÇİMEN

# 1-BİLGİLENDİRME

## 1.1 GİRİŞ

Sentetik veri üretme süreci, GAN (Generative Adversarial Networks) adı verilen bir yapay sinir ağı modelinin nasıl çalıştığına odaklanmaktadır.

Proje, günümüzde artan önem kazanan yapay zeka ve makine öğrenmesi alanlarında karşılaşılan veri eksikliği veya sınırlı veriye sahip olma sorununa çözüm sağlamayı hedeflemektedir. GAN, sentetik veri oluşturma için güçlü bir teknik olarak kabul edilmektedir. Bu çalışmada, GAN kullanılarak gerçek veri setlerinin eksikliklerini tamamlamak ve daha geniş bir veri kümesiyle makine öğrenme modellerinin performansını artırmak amaçlanmaktadır.

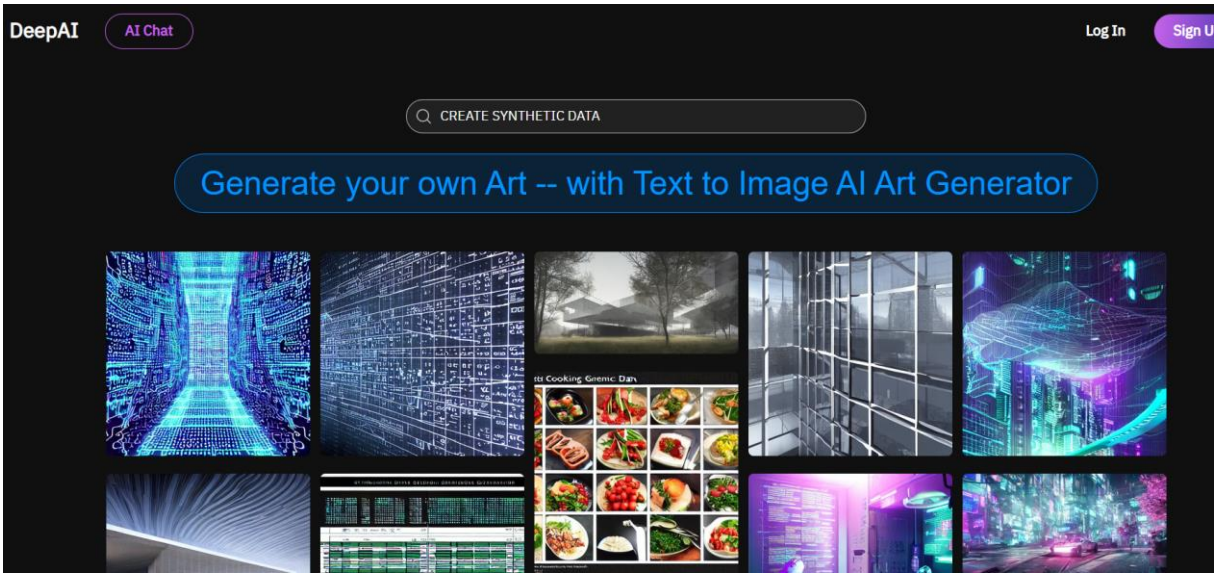
Makalede, sentetik veri geliştirme projesinde kullanılan yöntemler detaylı bir şekilde açıklanmaktadır. GAN'ın çalışma prensibi, generatif ve diskriminatif modellerin nasıl rekabet ettiği ve sentetik veri üretme süreci adım adım ele alınmaktadır. Ayrıca, projenin uygulanması sırasında karşılaşılan zorluklar ve bu zorlukların nasıl aşıldığı da makalede detaylı bir şekilde yer almaktadır.

Makalede ayrıca, sentetik veri geliştirme projesinin elde ettiği sonuçlar da sunulmaktadır. Elde edilen sentetik veri setlerinin gerçek verilere benzerlikleri, makine öğrenme modellerinin bu verileri kullanarak nasıl bir performans sergilediği ve sentetik veri geliştirme yönteminin başarısı değerlendirilmektedir. Sonuçlar, sentetik veri geliştirme projesinin başarısı ve potansiyel uygulama alanları hakkında bilgi vermektedir.

## 1.2 Verinin Gerçek Hayatta Kullanımına Örnek

- **Veriyi görselleştiren tasarımcımız REFİK ANADOL ESERİ**

# 2- SENTETİK VERİ




## 2.1 SENTETİK VERİ NEDİR?

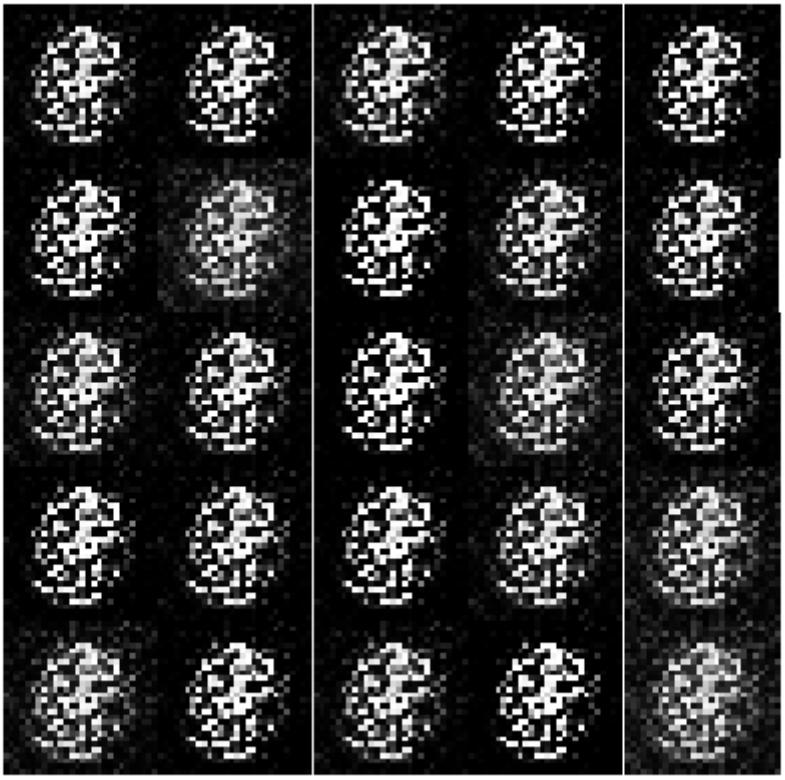
Sentetik veri, yapay olarak oluşturulan veri örnekleridir ve gerçek dünya verilerine benzerlik gösterir. Sentetik veri, genellikle eksik veri problemlerini çözmek veya veri miktarını artırmak için kullanılır.

Sentetik veri kullanmanın avantajlarından biri, gerçek verilerin gizliliğini ve hassasiyetini koruyabilmesidir. Örneğin, sağlık verileri üzerinde çalışırken, gerçek hastaların verilerini paylaşmak yerine sentetik veri kullanarak gizlilik sorunlarını aşabilirsiniz.

Ancak, sentetik veri her zaman gerçek veriye tam olarak benzemez veya gerçek dünyayı tam olarak yansıtmaz sonuçta gerçek veriler sayesinde model tarafından üretilen verilerdir; bu nedenle, sentetik veri kullanırken sonuçların gerçek verilerle karşılaştırılması ve değerlendirilmesi önemlidir.

## 2.2 GAN NEDİR?

Sentetik veri oluşturma yöntemleri arasında Generative Adversarial Networks (GAN) kullanımı yaygındır. GAN, bir generatif model ve bir diskriminatif (ayrıştırıcı) modelin rekabet ettiği bir yapay sinir ağı yapısını ifade eder. Generatif model, gerçekçi sentetik veri örnekleri üretmeye çalışırken, diskriminatif model, gerçek verileri sentetik verilerden ayırmaya çalışır.



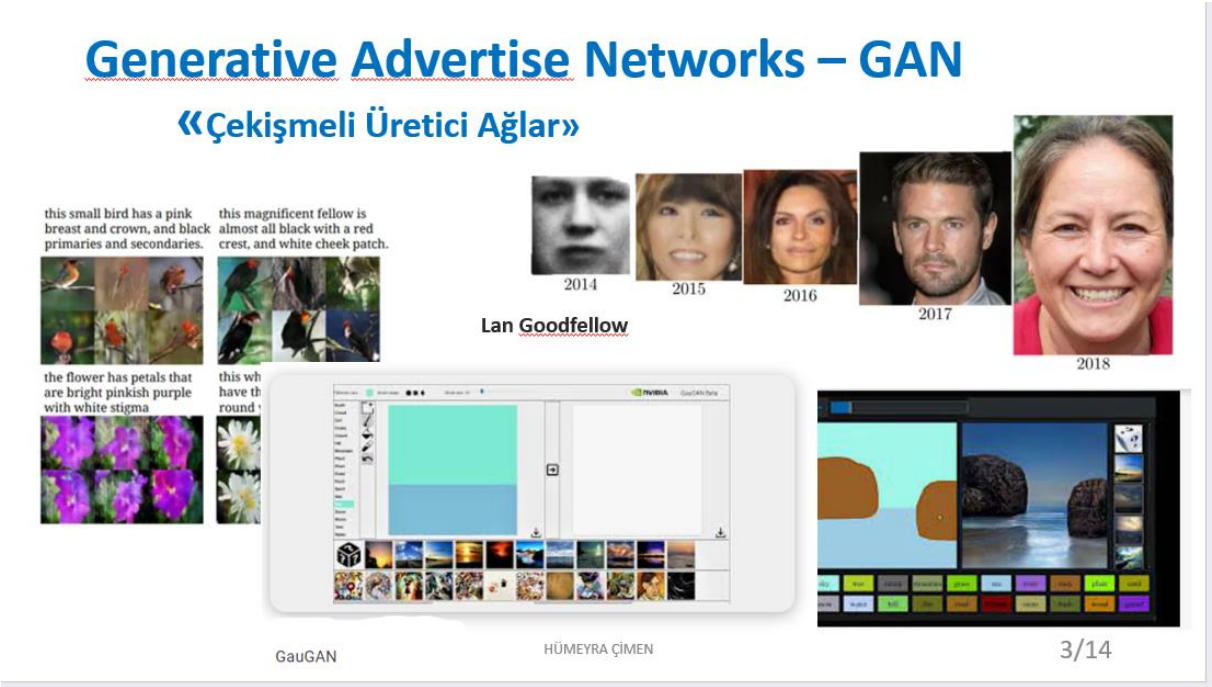
Epoch 1

Örneğin, bir GAN kullanarak el yazısı rakamlarının sentetik verilerini oluşturabilirsiniz. Generatif model, gerçek el yazısı rakamlarının özelliklerini öğrenerek yeni, gerçekçi el yazısı rakamları üretir. Bu sentetik verileri, daha geniş bir eğitim veri seti oluşturmak veya mevcut veri setini çeşitlendirmek için kullanabilirsiniz.

## 2.3 Generative Adversarial Networks (GAN) Tanımı

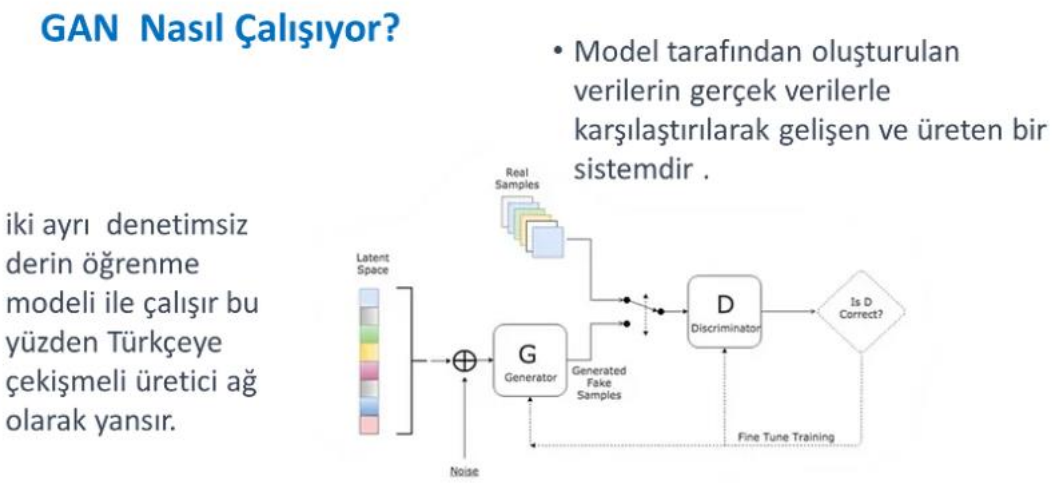
Generative Adversarial Networks (GAN), ilk olarak Lan Goodfellow tarafından ortaya atılan bir fikir olarak ortaya çıkmış ve güçlü yatırımlarla günümüze kadar geliştirilmiştir.

### 2.3.1 Yapay Zeka ile Oluşturulan İçeriklerin Popüler Konuları




Girilen metinlerle özelleştirilen özelliklere uygun resimler üretebilme yeteneği, aidiyetsiz profillerin (kedi, köpek, insan, bitki vb.) oluşturulması ve GauGAN gibi araçlarla üretilen resimler, günümüzde popüler konular arasında yer almaktadır. Bu tür resimlerin arka plandaki detayları araştırdığımızda, GAN modelleri ile karşılaşmaktayız.

### 2.3.2 Generative Advertise Networks – GAN «Çekişmeli Üretici Ağlar»

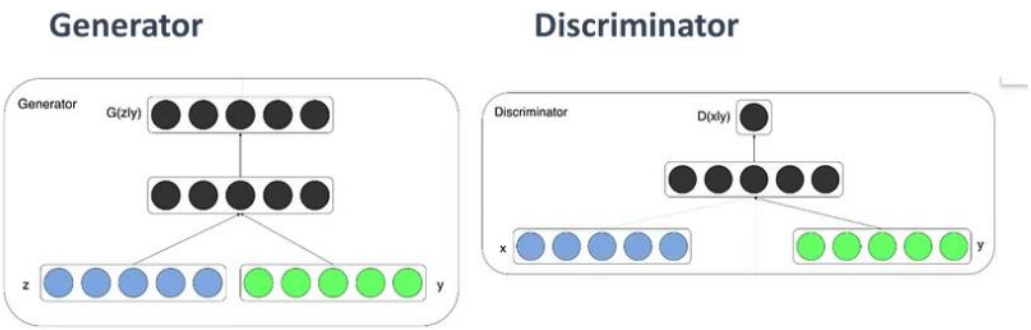


GAN bir sentetik veri oluşturma algoritmasıdır. Temelinde iki denetimsiz derin öğrenme modelini arasındaki çelişkiyi kullanarak oluşumlar üreten bir sistemdir.


Gerçek Mono Lisa tablosunu gerçek veri seti olarak değerlendirdiğimizde bir sanatçının bu eseri baz alarak eser oluşturması işlemini üretici ağ , bir dedektifin bu eserlerden hangisinin gerçek eser olduğunu ayırt etmesini ayrıştırıcı ağ olarak tanımlarız.

## 2.3.3 – DERİN ÖĞRENME MODELLERİ

– GENERATÖR (ÜRETİCİ) VE DİSCRİMİNATÖR (AYRIŞTIRICI)



- Generator-Üretici: Gerçek verilerle karşılaştırılarak ürettiği sentetik verilerin gerçeklere ne kadar yakın olduğunu öğrenerek üretim yapar.
- Discriminator -Ayrıştırıcı: Üretilen bu verilerin (Generatör tarafından) ne kadar gerçekçi olduğunu tespit etmeye çalışır. (accept reject).

Backpropagation (geri beslemeli) ve Gradient descent algoritmalarını kullanarak hata oranını minimize etmeye çalışır.

### 2.3.4 GAN ile Sentetik Veri Oluşturma: İhtiyaç ve Avantajları

Sentetik veri oluşturma, gerçek verilerin sınırlı, eksik veya maliyetli olması, gizlilik nedenleri veya belirli kullanım senaryoları için kullanılamaması durumunda büyük avantajlar sağlar. İşte GAN ile sentetik veri oluşturmanın avantajları:

**Hızlı ve Kolay Veri Üretimi:** Sentetik veriler, gerçek verilere kıyasla daha kolay ve hızlı bir şekilde üretilebilir. Bu, veri setlerini çeşitlendirmek veya eksik verileri tamamlamak için kullanılabilir.

**Amaç Odaklı Veri Üretimi:** Sentetik veriler, belirli bir amaca yönelik olarak oluşturulduğu için, kullanılacakları alana uygun şekilde hizmet edebilir. Bu, özel bir problem veya senaryo için örneklem oluşturma veya modele eğitim verisi sağlama gibi durumlarda büyük bir avantaj sağlar.

**Çekişmeli Modelin Başarısı:** GAN'lar, generatif ve diskriminatif modeller arasındaki çelişkiyi kullanarak çalışırlar. Bu çekişme, başarılı sonuçların ortaya çıkmasını sağlayabilir. GAN'lar, gerçek verilere dayanarak sentetik veriler üretirken, bu verilerin kullanım alanlarının çeşitliliği nedeniyle başarılı çalışmalara olanak tanır.

Sentetik verilerin kullanım alanları oldukça geniştir. Günümüzde verinin bir değer olarak kabul edildiği ve kullanım alanlarının sayısının sınırsız olduğu düşünülürse, sentetik verilerin bu kullanım alanlarına destek amacıyla oluşturulduğu bilgisi önemlidir. Örneğin, görüntü işlemeyle otonom sürüş kontrolleri veya sağlık alanında makine çıktılarının analizi gibi alanlarda sentetik

verilerin kullanımı yaygındır. Bu kullanım alanlarına daha detaylı olarak değinmek mümkündür.

### 2.3.5 GAN ile Veri Oluşturmadaki Zorluklar

Bu kadar avantajlı olmasının yanında tabi ki zorlukları da mevcut. Sonuçta model öğrenerek gelişen ve buna paralellikte kaliteye sahip ürünler sunmakta üretilen veriler nasıl bir kaliteye sahip oldu üretim yapılan veriden de kaynaklanır yeterli miktarda kaynak veri olmadan üretilen modeller birtakım anormaliler olabiliyor.

#### 2.3.5.1 GAN ile veri oluşturmanın bazı zorlukları vardır.

Kaynak Veri Yetersizliği: GAN modellerinin kaliteli ve gerçeğe yakın veriler üretebilmesi için yeterli miktarda kaynak veriye ihtiyaç vardır. Yetersiz kaynak veri, üretilen modellerde anormalliklere veya kalite kaybına neden olabilir.

Anormal Oluşumlar: Yetersiz veya kalitesiz kaynak verilerden üretilen modellerde anormal özellikler ortaya çıkabilir. Örneğin, çiçek modellerinde yaprakların sap uyumsuzluğu veya insan yüzlerindeki bölümlerin hizalanma sorunları gibi problemler görülebilir.

Modellerin Kararlılığı: GAN modelleri, eğitim sürecinde kararlılığını korumakta zorluklar yaşayabilir. Dengesizlikler veya aşırı öğrenme gibi sorunlar, üretilen verilerin kalitesini etkileyebilir.

Eğitim Sürecinin Zaman Alması: GAN modelleri genellikle uzun eğitim süreçleri gerektirir. Veri setinin büyüklüğüne, modelin karmaşıklığına ve eğitim parametrelerine bağlı olarak, eğitim süreci saatler veya hatta günler sürebilir.

Bu zorluklar, GAN ile veri oluşturma sürecinde karşılaşılan yaygın sorunlardır. Ancak, doğru önlemler alınarak ve iyi bir eğitim süreciyle, GAN modelleri daha kaliteli ve gerçeğe yakın veriler üretebilir hale gelebilir.

### 2.3.6 GAN İLE ÜRETİLEN VERİLERİN ÜRETİM ODAKLI SINIFLANDIRILMASI

Style Transfer (Stil Aktarımı): Stil aktarımı odaklı çalışmalarda, belirli bir çizgi veya duyguya sahip veriler üretilir. Bu genellikle sanat eserleri veya farklı tarzdaki görüntülerin üretimi gibi alanlarda kullanılır.

Conditioned GAN: Bağımlı oluşumlar, belirli özelliklere dayalı veri oluşturma odaklı çalışmalardır. Örneğin, beyaz benekli siyah tüylü uzun gagalı kuş resimlerinin üretilmesi gibi bir senaryoda, belirli özelliklere sahip veriler üretilebilir. Bu özellikler, gaga veya tüy filtrelemesi gibi belirli kriterler olabilir.

GAN'lar, veri üretiminde odaklı ve kontrollü bir şekilde çalışabildiği için çeşitli senaryolara uygun olarak kullanılabilir. Bu sayede istenen özelliklere sahip verilerin üretimi sağlanabilir.

## 3. GAN Modeli ve Mimari:

GAN modelleri, genellikle jeneratör ve ayırt edici olarak adlandırılan iki temel bileşenden oluşur. Bu bileşenler, birbirleriyle rekabet eden ve birlikte çalışan yapay sinir ağlarıdır.

Jeneratör, rastgele bir girdi olan bir vektörü alır ve bu girdiyi kullanarak sentetik veriler üretir. GAN'ların amacı, jeneratörün ürettiği verilerin gerçek verilere benzemesini sağlamaktır. Jeneratör genellikle bir veya daha fazla katmanlı bir yapay sinir ağıdır ve genellikle konvolüsyonel veya tam bağlantılı katmanlardan oluşur.

Ayırt edici ise, jeneratör tarafından üretilen sentetik verileri gerçek verilerden ayırt etmeye çalışır. Ayırt edici de genellikle bir yapay sinir ağıdır ve jeneratörden daha karmaşık olabilir. Ayırt edici, gelen veriyi analiz eder ve gerçek veya sentetik olduğunu tahmin etmek için bir olasılık değeri üretir.

GAN modelleri genellikle karşılıklı eğitim yöntemiyle çalışır. İlk olarak, jeneratörün parametreleri rastgele başlangıç değerleriyle ayarlanır ve ayırt edici eğitilir. Ardından, jeneratörün parametreleri, ayırt ediciyi yanıltacak şekilde güncellenir. Bu süreç, jeneratör ve ayırt edici arasında bir rekabet oluşturarak ilerler. Modelin eğitimi, belirli bir kriterin (örneğin, ayırt edicinin hata oranının minimize edilmesi) optimize edilmesiyle gerçekleştirilir.

GAN mimarileri, zaman içinde birçok gelişme görmüştür. Örneğin, Deep Convolutional GAN (DCGAN), Conditional GAN (CGAN), ve CycleGAN gibi çeşitli varyasyonları bulunmaktadır. Bu mimariler, farklı veri türleri ve uygulama senaryoları için optimize edilmiştir.

### 3.1 GAN Modellerinin Eğitim Adımları

1. Veri Hazırlığı: Eğitim için gerçek veri seti toplanır ve ön işleme adımlarıyla temizlenir. Veri seti, genellikle jeneratör tarafından üretilecek sentetik verilerle birlikte kullanılır.

2. Jeneratör ve Ayırt Edici Ağı İnşası: Jeneratör ve ayırt edici ağı, belirlenen mimariye uygun şekilde oluşturulur. Jeneratör genellikle bir yapay sinir ağıdır ve girdi olarak rastgele bir gürültü vektörü alırken, ayırt edici gerçek veya sentetik verileri sınıflandırmak için bir yapay sinir ağıdır.

3. Eğitim Döngüsü: Eğitim, jeneratör ve ayırt edici arasındaki rekabet üzerine kuruludur. Eğitim sürecinde, veri setinden rastgele gerçek veriler seçilir ve

jeneratör tarafından sentetik veriler üretilir. Ayırt edici, gerçek ve sentetik verileri sınıflandırır ve bir hata değeri hesaplanır.

4. Kayıp Fonksiyonu ve Optimizasyon: Kayıp fonksiyonu, jeneratörün ve ayırt edicinin performansını ölçer. Ayırt edicinin hata değeri minimize edilirken, jeneratörün hata değeri ise maksimize edilir. Bu rekabetçi süreç, GAN modelinin optimize edilmesini sağlar.

5. Gradient Güncellemesi: Kayıp fonksiyonuna göre hesaplanan gradyan değerleri kullanılarak, jeneratör ve ayırt edici ağların parametreleri güncellenir. Genellikle stokastik gradyan inişi (SGD) veya türevli gradyan yöntemi (Adam) gibi optimizasyon algoritmaları kullanılır.

6. İterasyonlar: Eğitim döngüsü belirli bir sayıda veya belirli bir konverjans kriterine ulaşana kadar tekrarlanır. Her iterasyonda, jeneratör ve ayırt edici arasındaki rekabet güncellenerek GAN modeli daha da iyileştirilir.

Eğitim süreci boyunca jeneratör, gerçek verilere benzeyen sentetik veriler üretmeyi öğrenirken, ayırt edici de gerçek ve sentetik verileri daha doğru bir şekilde ayırt etme yeteneğini geliştirir. Eğitimin sonunda, jeneratör gerçekçi sentetik veriler üretebilir ve ayırt edici, gerçek ve sentetik verileri ayırt etmede daha başarılı olur.

Sonuç olarak, GAN modeli eğitimi, jeneratör ve ayırt edici modelin çekişmeli olarak çalışması prensibine dayanır.

### 3.2- Eğitim Süreci ve Hiperparametre Ayarları

GAN modellerinin eğitimi genellikle uzun bir süreç gerektirir ve çeşitli hiperparametrelerin doğru ayarlanmasını gerektirir. İşte GAN modelinin eğitimi ve hiperparametre ayarlarına ilişkin bazı önemli noktalar:

3.2.1 Eğitim Süresi: GAN modelleri genellikle uzun süreli eğitim gerektirir. Eğitim süresi, veri setinin büyüklüğüne, modelin karmaşıklığına ve kullanılan donanıma bağlı olarak değişebilir. Büyük veri setleri ve karmaşık modeller, daha uzun eğitim süreleri gerektirebilir.

3.2.2. Öğrenme Hızı (Learning Rate): Öğrenme hızı, gradyan güncellemelerinin ne kadar hızlı yapılacağını kontrol eden bir hiperparametredir. Bu değer, jeneratör ve ayırt edici ağın güncelleme hızını etkiler. Yüksek bir öğrenme hızı, eğitimde dalgalanmalara ve kararsızlığa neden olabilir, düşük bir öğrenme hızı ise eğitim süresini uzatabilir. Deneme yanılma yöntemiyle uygun bir öğrenme hızı belirlenmelidir.

3.2.3 Batch Boyutu: Batch boyutu, eğitimde kullanılan örneklerin gruplar halinde işlenme miktarını belirler. Daha büyük batch boyutları, eğitim süresini hızlandırabilir, ancak bellek gereksinimini artırabilir. Daha küçük batch boyutları ise daha kararlı bir eğitim süreci sağlayabilir, ancak eğitim süresini uzatabilir. Uygun bir batch boyutu deneme yanılma yöntemiyle bulunmalıdır.

3.2.4 Aktivasyon Fonksiyonları: Jeneratör ve ayırt edici ağlarda kullanılan aktivasyon fonksiyonları, modelin performansını etkiler. GAN modellerinde genellikle jeneratörde ReLU veya LeakyReLU, ayırt edicide ise Sigmoid veya Softmax aktivasyon fonksiyonları kullanılır. Bu fonksiyonların doğru seçimi, modelin istenen sonuçları üretme yeteneğini artırır.

3.2.5. Kayıp Fonksiyonları: GAN modellerinde kullanılan kayıp fonksiyonları, jeneratör ve ayırt edici arasındaki rekabeti yönlendirir. Genellikle jeneratörde Binary Cross Entropy (BCE) kaybı, ayırt edicide ise BCE veya Categorical Cross Entropy (CCE) kaybı kullanılır. Kayıp fonksiyonlarının uygun şekilde seçilmesi, istenen sonuçları elde etme açısından önemlidir.

3.2.6. Regularizasyon Teknikleri: GAN modellerinde overfitting'i önlemek için regularizasyon teknikleri kullanılabilir. Dropout, Batch Normalization, L1 veya L2 düzenlileştirme gibi teknikler, modelin genelleme yeteneğini artırabilir ve istenmeyen özelliklerin oluşmasını engelleyebilir.

3.2.7. Örneklemleme Stratejileri: GAN modellerinin eğitimi sırasında kullanılan örneklemleme stratejileri, veri çeşitliliğini artırabilir ve eğitim sürecini stabilize edebilir. Örneğin, mini-batch yöntemi, veri setinden rastgele örnekler alarak eğitimi gerçekleştirir. Diğer stratejiler arasında dengeleme (balancing), weighted sampling veya stratified sampling gibi teknikler yer alabilir.

3.2.8. Model İyileştirme ve Değerlendirme: GAN modelleri eğitildikten sonra, elde edilen sonuçların değerlendirilmesi ve modelin iyileştirilmesi önemlidir. Bu aşamada, sentetik verilerin gerçek verilere ne kadar benzediği, ayırt edici hata oranı, doğruluk, hassasiyet, özgünlük gibi metrikler kullanılabilir. Model performansının iyileştirilmesi için hiperparametreler tekrar ayarlanabilir veya modelin mimarisi değiştirilebilir.

3.2.9 Hedeflenen Uygulama Senaryosu: GAN ile sentetik veri üretimi, farklı uygulama senaryolarında kullanılabilir. Örneğin, görüntü sentezi, metin sentezi, müzik sentezi gibi farklı veri tipleri üzerinde çalışmalar yapılabilir. Bu nedenle, eğitim sürecinde hedeflenen uygulama senaryosuna göre veri setinin hazırlığı, modelin mimarisi ve hiperparametre ayarları buna uygun şekilde yapılmalıdır.

Bu adımlar ve hiperparametre ayarları, GAN ile sentetik veri üretimi üzerine makale yazarken ele alınması gereken önemli konuları kapsar. Her adımın doğru bir şekilde gerçekleştirilmesi, kaliteli ve gerçeğe yakın sentetik verilerin üretilmesini sağlar.

3.2.10 Eğitim Verimliliği ve Zorluklar: GAN modellerinin eğitimi bazı zorluklar ve verimlilik sorunları içerebilir. Bu zorluklar arasında aşağıdakiler bulunur:

Mod Kollapsu: GAN modellerinde bazen jeneratör, sadece birkaç farklı örneği tekrarlayarak sentetik veri üretebilir. Bu durum, mod kollapsu olarak adlandırılır ve sentetik verilerin çeşitliliğini azaltabilir. Mod kollapsunu önlemek için kayıp fonksiyonları, hiperparametreler ve veri seti çeşitliliği üzerinde çalışmalar yapılabilir.

Dengesizlik Sorunu: GAN modellerinin eğitimi sırasında jeneratör ve ayırt edici arasında bir dengenin sağlanması zor olabilir. Örneğin, jeneratör çok hızlı bir şekilde iyileşirken, ayırt edici zayıf kalabilir veya tam tersi olabilir. Dengesizlik sorununu çözmek için kayıp fonksiyonlarının düzgün ayarlanması ve eğitim sürecinin dikkatli bir şekilde takip edilmesi gereklidir.

Eğitim Veri Setinin Kalitesi: GAN modelleri, gerçekçi sentetik veriler üretebilmek için yeterli ve temsilci bir eğitim veri setine ihtiyaç duyar. Eğitim veri setinin kalitesi, sentetik verilerin gerçek verilere benzemesini sağlar. Veri setinin temizliği, çeşitliliği ve doğruluğu, modelin performansını etkileyebilir.

Eğitim Süresi ve Hesaplama Gücü: GAN modelleri, genellikle büyük ve karmaşık yapılara sahiptir, bu da eğitim süresini ve hesaplama gücünü artırır. Uzun eğitim süreleri, gerekli donanım kaynakları ve zaman gerektirebilir. Hesaplama gücü sınırlamaları, daha hızlı ve verimli eğitim için optimize edilmiş algoritmaların kullanılmasını gerektirebilir.

Eğitim Sürecindeki Kararsızlık: GAN modelleri, eğitim sürecinde kararsızlık ve dalgalanmalarla karşılaşabilir. Bu, jeneratör ve ayırt edici arasındaki rekabetten kaynaklanabilir. Eğitim sürecinde kararlılık sağlamak için hiperparametrelerin dikkatli bir şekilde ayarlanması ve veri setinin uygun şekilde işlenmesi gereklidir.

Bu zorluklar ve verimlilik sorunları, GAN modelinin eğitimi sırasında karşılaşılabilecek önemli konulardır. Araştırmacılar ve uygulayıcılar, bu zorlukları aşmak ve GAN modellerinin performansını artırmak için sürekli olarak çalışmalar yürütmektedir.

## 4-StyleGan ile İnsan Yüzü Oluşturma

## Yürütülen Adımlar:

### 4.1

Gerekli bağımlılıkları yüklemek için pip aracılığıyla ninja paketini kurun:"

```
!git clone https://github.com/NVlabs/stylegan3.git
!pip install ninja
```

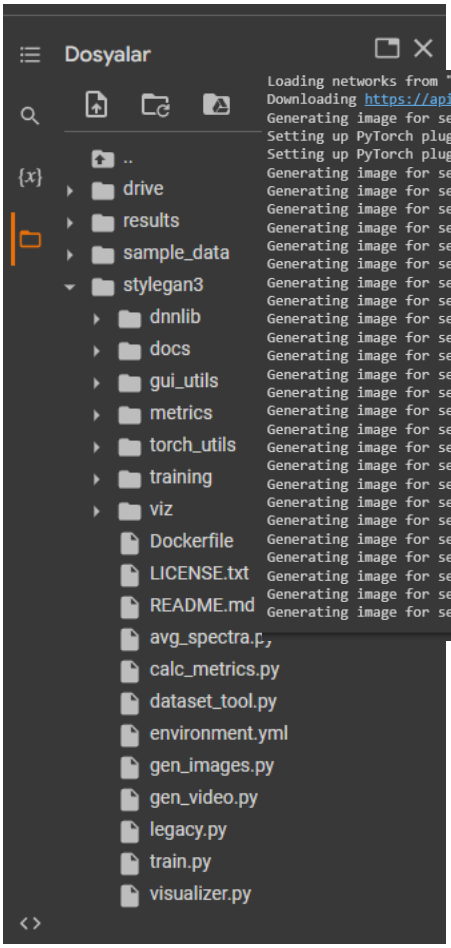
### 4.2

NVlabs tarafından geliştirilen StyleGAN3 projesinin gen_images.py adlı bir betiği kullanmaktadır. Kod, belirli bir model ağı kullanarak sentetik görüntülerin üretilmesini hedeflemektedir.

```
# StyleGAN3 modelinin indirme bağlantısı
URL = "https://api.ngc.nvidia.com/v2/models/nvidia/research/"\
      "stylegan3/versions/1/files/stylegan3-r-ffhq-1024x1024.pkl"

# Gen_images.py betiğini çalıştırarak sentetik görüntülerin üretilmesi
!python /content/stylegan3/gen_images.py \
    --network={URL} \
    --outdir=/content/results --seeds=6600-6625
```

Bu kod parçası, belirtilen StyleGAN3 model ağı kullanılarak seeds (6600-6625 aralığındaki rastgele sayılarla) sentetik görüntülerin üretildiği bir işlem gerçekleştirir. Üretilen görüntüler, /content/results klasörüne kaydedilir




Kod çıktısı:

```
Loading networks from "https://api.ngc.nvidia.com/v2/models/nvidia/research/stylegan3/versions/1/files/stylegan3-r-ffhq-1024x1024.pkl"...
Downloading https://api.ngc.nvidia.com/v2/models/nvidia/research/stylegan3/versions/1/files/stylegan3-r-ffhq-1024x1024.pkl ... done
Generating image for seed 6600 (0/26) ...
Setting up PyTorch plugin "bias_act_plugin"... Done.
Setting up PyTorch plugin "filtered_lrelu_plugin"... Done.
Generating image for seed 6601 (1/26) ...
Generating image for seed 6602 (2/26) ...
Generating image for seed 6603 (3/26) ...
Generating image for seed 6604 (4/26) ...
Generating image for seed 6605 (5/26) ...
Generating image for seed 6606 (6/26) ...
Generating image for seed 6607 (7/26) ...
Generating image for seed 6608 (8/26) ...
Generating image for seed 6609 (9/26) ...
Generating image for seed 6610 (10/26) ...
Generating image for seed 6611 (11/26) ...
Generating image for seed 6612 (12/26) ...
Generating image for seed 6613 (13/26) ...
Generating image for seed 6614 (14/26) ...
Generating image for seed 6615 (15/26) ...
Generating image for seed 6616 (16/26) ...
Generating image for seed 6617 (17/26) ...
Generating image for seed 6618 (18/26) ...
Generating image for seed 6619 (19/26) ...
Generating image for seed 6620 (20/26) ...
Generating image for seed 6621 (21/26) ...
Generating image for seed 6622 (22/26) ...
Generating image for seed 6623 (23/26) ...
Generating image for seed 6624 (24/26) ...
Generating image for seed 6625 (25/26) ...
```

## 4.3 Gen_images.py Model incelenmesi

Önceden eğitilmiş bir ağ kullanarak belirli bir rastgele tohumdan başlayarak sentetik görüntüler oluşturur. Parametreler aracılığıyla ağın yüklenmesi, üretilecek tohumların belirlenmesi, kesme değeri (truncation), gürültü modu ve çıktı dizini gibi ayarlamalar yapılabilir. Ardından, tohumlara dayalı olarak görüntüler üretilir ve belirtilen çıktı dizinine kaydedilir.

## 4.4 Görüntülerin Üretilmesi

```
138
139     # Görüntülerin üretilmesi.
140 for seed_idx, seed in enumerate(seeds):
141     print('Tohum %d için görüntü üretiliyor (%d/%d) ...' % (seed, seed_idx, len(seeds)))
142     z = torch.from_numpy(np.random.RandomState(seed).randn(1, G.z_dim)).to(device)
143
144     # Ters dönüşüm/çeviri matrisi oluşturulur ve generatöre iletilir.
145     # Generatör, potansiyel hataları önlemek için bu matrisi ters olarak bekler.
146     if hasattr(G.synthesis, 'input'):
147         m = make_transform(translate, rotate)
148         m = np.linalg.inv(m)
149         G.synthesis.input.transform.copy_(torch.from_numpy(m))
150
151     img = G(z, label, truncation_psi=truncation_psi, noise_mode=noise_mode)
152     img = (img.permute(0, 2, 3, 1) * 127.5 + 128).clamp(0, 255).to(torch.uint8)
153     PIL.Image.fromarray(img[0].cpu().numpy(), 'RGB').save(f'{outdir}/seed{seed:04d}.png')
154
155
```

- Belirli bir tohumun indeksini ve değerini kullanarak, 'Generating image for seed %d (%d/%d)' şeklinde bir bildirim görüntülenir. Bu, kullanıcıya hangi tohumun üretim aşamasında olduğunu gösterir.
- 'z' olarak adlandırılan bir rastgele gürültü vektörü oluşturulur. Bu vektör, 'np.random.RandomState' fonksiyonuyla tohum temel alınarak elde edilir ve 'torch.from_numpy' fonksiyonuyla GPU belleğine aktarılır. Bu gürültü vektörü, jeneratörün rastgelelik unsurlarını kontrol etmek için kullanılır.
- Eğer jeneratörde 'input' adında bir dönüşüm matrisi varsa, bu matris oluşturulur ve generatöre iletilir. Dönüşüm matrisi, verilen 'translate' ve 'rotate' değerlerine dayalı olarak oluşturulur ve jeneratörün içindeki hataları önlemek için tersine alınır.
- Jeneratör çağrısı yapılırken, oluşturulan gürültü vektörü ('z'), etiket ('label'), kesme değeri ('truncation_psi') ve gürültü modu ('noise_mode') parametreleriyle birlikte kullanılır. Bu çağrı sonucunda, jeneratör tarafından bir görüntü üretilir.
- Elde edilen görüntü, piksel değerlerinin düzgün bir şekilde temsil edilmesi için uygun formata dönüştürülür. Ardından, bu görüntü, belirtilen çıktı dizinine ve ilgili tohumun indeksine göre kaydedilir. Dosya adı, 'seed' kelimesi ve dört haneli bir sayı içerir, bu sayı tohumun değerini yansıtır.

Bu adımlar, belirtilen tohumlar için sentetik görüntülerin üretildiği işlemi tanımlar. Her tohum için bu adımlar tekrarlanır ve sonuç olarak, çıktı dizininde her tohuma karşılık gelen bir görüntü dosyası oluşturulur.

## 4.5 GAN Modeli Kullanarak Görüntü Sentezleme

GAN tabanlı görüntü sentezleme işlemlerini gerçekleştirmek için uygulanan adımların içerikleri şu şekildedir.

- seed2vec(G, seed): Bu fonksiyon, verilen bir tohumdan (seed) GAN'in gürültü vektörünü (z) oluşturur. Gürültü vektörü, GAN modelinin girdi olarak kullanılarak yeni bir görüntü üretmek için kullanılır.
- display_image(image): Bu fonksiyon, verilen bir görüntüyü (image) görsel olarak görüntüler. Bu, üretilen görüntüleri incelemek veya sonuçları görüntülemek için kullanılabilir.
- generate_image(G, z, truncation_psi): Bu fonksiyon, belirtilen GAN modeli (G), gürültü vektörü (z) ve kesme değeri (truncation_psi) kullanarak bir görüntü üretir. Gürültü vektörü, GAN modeline rastgelelik ekleyerek çeşitli görüntülerin üretilmesini sağlar. Kesme değeri, üretilen görüntülerin çeşitliliğini ve detay seviyesini kontrol etmek için kullanılır.
- get_label(G, device, class_idx): Bu fonksiyon, belirtilen GAN modeli (G), cihaz (device) ve sınıf indeksi (class_idx) kullanarak görüntü etiketini döndürür. GAN modeli, sınıflandırma yapabilen bir yapıya sahipse, belirtilen sınıf indeksiyle ilgili etiketi oluşturur.

- generate_image(device, G, z, truncation_psi=1.0, noise_mode='const', class_idx=None): Bu fonksiyon, belirtilen cihaz (device), GAN modeli (G), gürültü vektörü (z), kesme değeri (truncation_psi), gürültü modu (noise_mode) ve sınıf indeksi (class_idx) kullanarak bir görüntü üretir. Gürültü vektörü, rastgelelik ekleyerek çeşitli görüntülerin oluşturulmasına olanak sağlar. Kesme değeri, üretilen görüntülerin çeşitliliğini ve detay seviyesini kontrol etmek için kullanılır. Gürültü modu, gürültü bileşeninin nasıl manipüle edileceğini belirler. Sınıf indeksi, görüntü etiketini belirlemek için kullanılır.

## 4.6 NVIDIA Araştırma Ekibi Tarafından Eğitilen ve Yüksek Çözünürlüklü Yüz Görüntülerini Sentezlemek İçin Kullanılan StyleGAN3 Modeline Ait Ağırlıkların Kullanımı

Öncelikle, ağırlıkları indirmek için bir URL kullanılır. Bu URL, NVIDIA Araştırma Ekibi tarafından eğitilen ve yüksek çözünürlüklü yüz görüntülerini sentezlemek için kullanılan StyleGAN3 modeline ait ağırlıkları içermektedir.

İndirme işlemi, dnnlib.util.open_url() fonksiyonu aracılığıyla gerçekleştirilir.

Ağırlıklar indirildikten sonra, legacy.load_network_pkl() fonksiyonu kullanılarak ağırlıklar GAN modeline yüklenir ve "G_ema" (Eşit Ağırlıklı Oluşturucu) olarak adlandırılan özel bir bileşenin içinde saklanır. Bu bileşen, yüksek kaliteli yüz görüntülerinin sentezlenmesi için gereken parametreleri içerir.

Ayrıca, CUDA uyumlu bir cihaz kullanılması sağlanır. CUDA, NVIDIA'nın paralel hesaplama platformudur ve derin öğrenme işlemlerinin GPU'lar aracılığıyla hızlandırılmasına olanak tanır. Bu nedenle, GAN modeli CUDA ile uyumlu bir cihazda çalışacak şekilde ayarlanır. Bu kodparçacığı, torch.device('cuda') ifadesiyle CUDA uyumlu bir cihazın kullanılmasını sağlar.

Böylece, StyleGAN3 modeline ait önceden eğitilmiş ağırlıklar başarıyla indirilir ve yüklenir. Bu ağırlıkların kullanılmasıyla çeşitli sentezleme işlemleri gerçekleştirilebilir ve yüksek kaliteli yüz görüntüleri elde edilebilir.

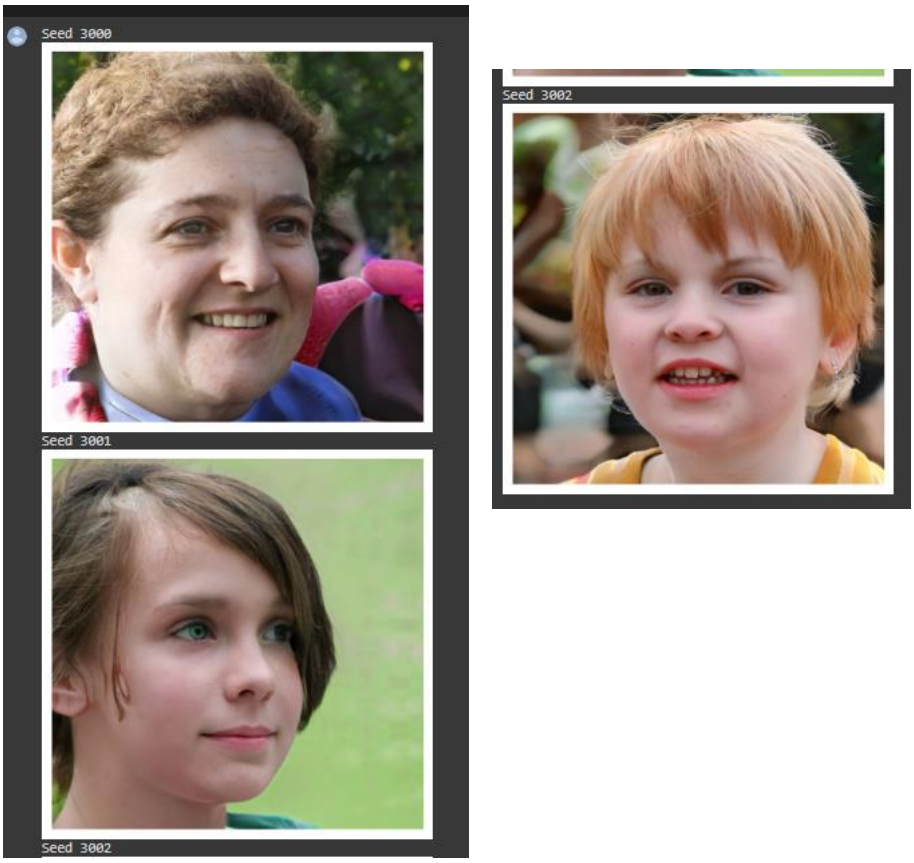
## 4.7 Belirli Bir Başlangıç ve Bitiş Tohumunu Seçin

SEED_FROM = 3000 # Başlangıç tohumu

SEED_TO = 3003 # Bitiş tohumu

## Tohumlar için görüntüleri oluşturun.

```
12   for i in range(SEED_FROM, SEED_TO):
13 print(f"Seed {i}")
14 z = seed2vec(G, i) # Tohumu vektöre dönüştür
15 img = generate_image(device, G, z) # GAN modelini kullanarak görüntüyü oluştur
16 display_image(img) # Görüntüyü göster
```




expand_seed adlı bir fonksiyon tanımlar. Bu fonksiyon, belirtilen tohumların ve vektör boyutunun kullanıldığı bir liste oluşturur. Her bir tohum için rastgele sayılar kullanılarak vektör oluşturulur ve sonuç listesine eklenir.

```
Loading networks from "https://api.ngc.nvidia.com/v2/models/nvidia/research/stylegan3/versions/1/files/stylegan3-r-ffhq-1024x1024.pkl"...
(1, 512)
```

## 4.8 Seçilen Tohumlar Arasında Pürüzsüz Dönüşüm ve Video Oluşturma

Seçilen tohumları ve her birine ulaşmak için alınacak adım sayısını belirlemek için SEEDS ve STEPS değişkenlerini kullanın.

SEEDS, dönüşüm yapılacak tohumları içeren bir liste olarak tanımlanır. Burada, yüzler için daha uygun olan [2624, 2618, 2616] tohumları kullanılmıştır. Ayrıca, balıklar için daha uygun olan [3000, 3003, 3001] tohumlarını kullanmak için yorum satırı olarak verilmiştir.

STEPS, her iki tohum arasında gerçekleştirilecek adım sayısını belirler. Daha fazla adım, daha pürüzsüz bir dönüşüm elde etmek için kullanılabilir.

Sonraki adımlar, önceki sonuçları temizlemeyi ve sonuçlar için bir klasör oluşturmayı içerir. Ardından, döngü kullanılarak tohumlar arasındaki dönüşüm için görüntüler oluşturulur.

Her bir dönüşüm için, başlangıç ve bitiş vektörleri (v1 ve v2) hesaplanır ve aradaki fark belirlenir. Bu fark, adımlara bölünerek (step) her bir adım için geçerli vektör hesaplanır. Başlangıç vektörü current, her adımda step ile güncellenir ve bu vektör kullanılarak görüntüler sentezlenir. Her bir adım için oluşturulan görüntüler, ./results/ klasörüne kaydedilir.

Son olarak, oluşturulan görüntüler kullanılarak bir video oluşturulur. Bu işlem için ffmpeg kullanılır ve oluşturulan görüntüler bağlantılandırılarak movie.mp4 adında bir video dosyası oluşturulur.

Bu kod parçası, belirli tohumlar arasında pürüzsüz bir dönüşüm yapmak ve bu dönüşümü içeren bir video oluşturmak için kullanılır.

## 4.9 Model Sonucunda Oluşan Resimlerden Kesitler

SEED_FROM = 3000 # Başlangıç tohumu

SEED_TO = 3003 # Bitiş tohumu

[1000, 1003, 1001]

# 5. SOYUT SANAT VERİ SETİ KULLANARAK VERİ ÜRETİMİ

EXPERIMENTS değişkeni, eğitimden elde edilen sonuçların kaydedileceği dizini temsil eder. Bu dizin, deneylerin yapıldığı klasörü temsil eder ve kullanıcı tarafından ihtiyaçlarına göre özelleştirilebilir.

DATA değişkeni, eğitimde kullanılacak olan veri setinin dizinini temsil eder. Bu veri seti, GAN modelinin öğrenme işlemi için kullanılacak gerçek görüntülerden oluşur.

## 5.0 Neden Abstarct Art Veri seti seçildi?

Soyut sanat eserlerinin veri seti olarak tercih edilmesinin nedeni, bu eserlerdeki bozuklukların daha az fark edilmesidir. Bu da modelin aslında başarısını görüntülemek için bir fırsat olacaktır . GAN modelinin farklı modellere göre daha başarılı olmasını altında yatan sebebi sorgulatır ve sonucunda da adı gibi çekişmeli olarak gelişen derin öğrenme modeli olması bu sorunun cevabıdır. Soyut sanatın karmaşık ve özgün yapısı, veri analiz sürecinde daha güvenilir sonuçlar elde edilmesini sağlar. Bu makale, soyut sanatın veri analizindeki avantajlarına odaklanarak, gelecekteki araştırmalar için bir temel oluşturmayı amaçlamaktadır.

## 5.1 PROJE ORTAMI HAZIRLANDI

### 5.1.1 Neden Colab Platformu Tercih edildi?

Google Colab ücretiz olarak kullanıma sunduğu 15 GB ram ve GPU kullanımına olanak sağlamakta Bu boyuttaki bir bilgisayarın maliyeti çok fazla olduğu için ve bu proje gibi gpu gerektiren projelerde süreci hızlandırmakta ve bilgisayar yükünü azaltmaktadır.

### 5.1.2 Google Drive ile Bağlantı Oluşturuldu

```
1
2  try:
3      from google.colab import drive
4      drive.mount('/content/drive', force_remount=True)
5      COLAB = True
6      print("Note: using Google CoLab")
7      %tensorflow_version 2.x
8  except:
9      print("Note: not using Google CoLab")
10     COLAB = False
```

Google Colab platformu kullanılarak Google Drive ile bağlantı kurma gereksinimi duyuldu. Bir hizmet olan Google Colab, kullanıcıların bulut tabanlı bir Jupyter Notebook ortamında Python

kodlarını çalıştırmalarına izin verilir.

Bağlantıyı kurmak için, projede google.colab kütüphanesinden drive modülü import edildi. Bu modül, Colab ortamında Google Drive'a erişimin sağlanmasını sağlanır. Aşağıdaki kod parçası, bağlantının oluşturulması adımlarını içerir.

## 5.2.0 Gerekli Kurulumlar Yapıldı

Torch, Python tabanlı bir derin öğrenme kütüphanesidir ve GAN projelerinde yaygın olarak kullanılır. Torchvision ise Torch'un bir bileşenidir ve görüntü işlemeyle ilgili işlevleri içerir.

```
1   !pip install torch==1.8.1 torchvision==0.9.1
2
```

Bu kütüphaneler, GAN projelerinde model oluşturma, eğitim ve sonuçların analizi için bir dizi araç sağlar.

```
1   !pip uninstall jax jaxlib -y
2
Found existing installation: jax 0.3.10
Uninstalling jax-0.3.10:
  Successfully uninstalled jax-0.3.10
Found existing installation: jaxlib 0.3.10+cuda11.cudnn805
Uninstalling jaxlib-0.3.10+cuda11.cudnn805:
  Successfully uninstalled jaxlib-0.3.10+cuda11.cudnn805
```

JAX (Just Another XLA) bir Python kütüphanesidir ve bilimsel hesaplamalar, makine öğrenimi ve derin öğrenme alanlarında kullanılan güçlü bir araçtır. JAX, özellikle derin öğrenme modellerinin hızlı ve ölçeklenebilir bir şekilde çalıştırılması için tasarlanmıştır. Ayrıca, JAX, TensorFlow ekosistemiyle de entegre bir şekilde çalışabilir.

```
!git clone https://github.com/NVlabs/stylegan2-ada-pytorch.git
```

Yukarıdaki komutu çalıştırdığınızda, belirtilen GitHub deposunun tüm içeriği, mevcut çalışma dizininizde bir kopya olarak oluşturulur. Bu, projeyi yerel olarak kullanmanızı ve üzerinde çalışmanızı sağlar.

"stylegan2-ada-pytorch" deposu, NVIDIA tarafından geliştirilen bir derin öğrenme modeli olan StyleGAN2-ADA'nın PyTorch tabanlı bir uyarlamasını içerir. Bu model, yüksek çözünürlüklü görüntü sentezi yapabilen ve gerçekçi görüntüler üretebilen bir GAN modelidir.

Klonlanan depo, modelin örnek kodları, eğitim ve test veri setleri, önceden eğitilmiş ağırlıklar ve ilgili belgeler gibi içerikleri içerir. Bu şekilde, stil tabanlı görüntü sentezi veya stil transferi gibi görsel projeler için temel oluşturabilir ve modeli kullanarak kendi çalışmalarınızı yapabilirsiniz.

## 5.2 Veri Kümesi Oluşturma

CMD: Bu değişken, dataset_tool.py betiğini çalıştırmak için kullanılan komutu içermektedir.

- "/content/stylegan2-ada-pytorch/dataset_tool.py”: Bu,çalıştırılacak
- Python betiğinin yolu ve adını belirtmektedir.
- "--source /content/drive/MyDrive/data/gan/images/Abstract_gallery_2": Bu, veri kümesi oluşturmak için kullanılacak kaynak dizinin yolunu belirtmektedir. Bu örnekte, Abstract_gallery_2 adlı bir dizin kullanılmıştır.
- "--dest /content/drive/MyDrive/data/gan/dataset/circuit5": Bu ise oluşturulan veri kümesinin hedef dizinini belirtmektedir. Bu örnekte, circuit5 adlı bir dizin kullanılmıştır.

Oluşturulan veri kümesi, belirtilen hedef dizininde yer alacaktır.

Bu kod parçası, veri kümesinin oluşturulması ve hazırlanması işlemlerini gerçekleştirmek amacıyla dataset_tool.py betiğini çağırmaktadır. Bu, GAN projenizde veri kümesinin oluşturulması ve düzenlenmesi adımlarını gerçekleştirmenizi sağlamaktadır.

```
CMD = "python /content/stylegan2-ada-pytorch/dataset_tool.py "\
  "--source /content/drive/MyDrive/data/gan/images/Abstract_gallery_2 "\
  "--dest /content/drive/MyDrive/data/gan/dataset/circuit5"

!{CMD}
```

```
100% 90/90 [00:11<00:00,  7.66it/s]
```

## 5.3 Görüntülerin Boyut Ve Renk Formatı Tutarlılığının Kontrol Edilmesi

Görüntülerin boyut ve renk formatı tutarlılığını kontrol ederek, gerekirse uygun önlemlerin alınmasını sağlar. Böylece, proje çalışmalarınızda tutarlı ve doğru veri kullanımını sağlayabilirsiniz.

İlk adımda, belirtilen dizindeki görüntülerin listesi oluşturulur. Ardından, her bir görüntünün boyutu ve renk formatı üzerinde kontrol yapılır.

Bu kontrol işlemi, base_size adında bir değişken kullanılarak gerçekleştirilir. İlk görüntünün boyutu base_size olarak atanır. Daha sonra diğer görüntülerin boyutları bu base_size ile karşılaştırılır. Eğer bir görüntünün boyutu base_size ile tutarsız ise, bu durum "Tutarlı Olmayan Boyut" hatası olarak bildirilir. Benzer şekilde, eğer bir görüntünün renk formatı "RGB" değil ise, "Tutarlı Olmayan Renk Formatı" hatası olarak bildirilir.

Bu kontrol mekanizması, projede kullanılan veri setinin tutarlı ve doğru olmasını sağlar. Tutarsızlık durumlarında, uygun önlemler alınarak veri setindeki bozukluklar düzeltilebilir veya uygun şekilde ele alınabilir. Böylece, proje çalışmalarında güvenilir sonuçlar elde edilir ve yanlış analizlere veya sonuçlara yol açabilecek veri bozuklukları önlenmiş olur.

Bu kod parçası, veri setindeki görüntülerin boyut ve renk tutarlılığını kontrol ederek, proje çalışmalarında doğru ve güvenilir veri kullanımını sağlamak için önemli bir adımdır.

## 5.4 GAN Modeli Eğitimi

Bu kod parçası, GAN modelinin eğitimini yapmak için gerekli parametreleri belirleyerek eğitim işlemini gerçekleştirir.

```
import os

# Modify these to suit your needs
EXPERIMENTS = "/content/drive/MyDrive/data/gan/experiments/x"
DATA = "/content/drive/MyDrive/data/gan/dataset/circuit5"
SNAP = 10

# Build the command and run it
cmd = f"/usr/bin/python3 /content/stylegan2-ada-pytorch/train.py "\
  f"--snap {SNAP} --outdir {EXPERIMENTS} --data {DATA}"
!{cmd}
```

```
raining options:
{
 "num_gpus": 1,
 "image_snapshot_ticks": 10,
 "network_snapshot_ticks": 10,
 "metrics": [
   "fid50k_full"
 ],
 "random_seed": 0,
 "training_set_kwargs": {
   "class_name": "training.dataset.ImageFolderDataset",
   "path": "/content/drive/MyDrive/data/gan/dataset/circuit5",
   "use_labels": false,
   "max_size": 90,
   "xflip": false,
   "resolution": 1024
 },
 "data_loader_kwargs": {
   "pin_memory": true,
   "num_workers": 3,
   "prefetch_factor": 2
 },
 "G_kwargs": {
   "class_name": "training.networks.Generator",
```

SNAP değişkeni eğitim sırasında hangi aralıklarla modelin kaydedileceğini belirler. Bu değer, her bir arabellek dönemi (buffering period) sonunda modelin kaydedileceği sayısal bir değerdir.

Sonraki adımda, cmd değişkeni kullanılarak eğitim işleminin çalıştırılacağı komut oluşturulur. Bu komut, train.py betiğini belirtilen parametrelerle çağırır ve eğitim işlemini başlatır.

Son olarak, !{cmd} ifadesi kullanılarak oluşturulan komut çalıştırılır ve GAN modelinin eğitimi başlar.

```
Output directory:   /content/drive/MyDrive/data/gan/experiments/x/00000-circuit5-auto1
Training data:      /content/drive/MyDrive/data/gan/dataset/circuit5
Training duration:  25000 kimg
Number of GPUs:     1
Number of images:   90
Image resolution:   1024
Conditional model:  False
Dataset x-flips:    False

Creating output directory...
Launching processes...
Loading training set...
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:560: UserWarning:
  warnings.warn(_create_warning_msg(

Num images:  90
Image shape: [3, 1024, 1024]
Label shape: [0]
```

GAN modelinin Generator (Üreteci denetimsiz ögrenme modeline) bileşenine ait özelliklerin ve parametrelerin bir tablo halinde sunulduğu bir gösterim bulunmaktadır.

| Generator | Parameters | Buffers | Output shape | Datatype |
|---|---|---|---|---|
| mapping.fc0 | 262656 | - | [4, 512] | float32 |
| mapping.fc1 | 262656 | - | [4, 512] | float32 |
| mapping | - | 512 | [4, 18, 512] | float32 |
| synthesis.b4.conv1 | 2622465 | 32 | [4, 512, 4, 4] | float32 |
| synthesis.b4.torgb | 264195 | - | [4, 3, 4, 4] | float32 |
| synthesis.b4:0 | 8192 | 16 | [4, 512, 4, 4] | float32 |
| synthesis.b4:1 | - | - | [4, 512, 4, 4] | float32 |
| synthesis.b8.conv0 | 2622465 | 80 | [4, 512, 8, 8] | float32 |
| synthesis.b8.conv1 | 2622465 | 80 | [4, 512, 8, 8] | float32 |
| synthesis.b8.torgb | 264195 | - | [4, 3, 8, 8] | float32 |
| synthesis.b8:0 | - | 16 | [4, 512, 8, 8] | float32 |
| synthesis.b8:1 | - | - | [4, 512, 8, 8] | float32 |
| synthesis.b16.conv0 | 2622465 | 272 | [4, 512, 16, 16] | float32 |
| synthesis.b16.conv1 | 2622465 | 272 | [4, 512, 16, 16] | float32 |
| synthesis.b16.torgb | 264195 | - | [4, 3, 16, 16] | float32 |
| synthesis.b16:0 | - | 16 | [4, 512, 16, 16] | float32 |
| synthesis.b16:1 | - | - | [4, 512, 16, 16] | float32 |
| synthesis.b32.conv0 | 2622465 | 1040 | [4, 512, 32, 32] | float32 |
| synthesis.b32.conv1 | 2622465 | 1040 | [4, 512, 32, 32] | float32 |
| synthesis.b32.torgb | 264195 | - | [4, 3, 32, 32] | float32 |
| synthesis.b32:0 | - | 16 | [4, 512, 32, 32] | float32 |
| synthesis.b32:1 | - | - | [4, 512, 32, 32] | float32 |
| synthesis.b64.conv0 | 2622465 | 4112 | [4, 512, 64, 64] | float32 |
| synthesis.b64.conv1 | 2622465 | 4112 | [4, 512, 64, 64] | float32 |
| synthesis.b64.torgb | 264195 | - | [4, 3, 64, 64] | float32 |
| synthesis.b64:0 | - | 16 | [4, 512, 64, 64] | float32 |
| synthesis.b64:1 | - | - | [4, 512, 64, 64] | float32 |
| synthesis.b128.conv0 | 1442561 | 16400 | [4, 256, 128, 128] | float16 |
| synthesis.b128.conv1 | 721409 | 16400 | [4, 256, 128, 128] | float16 |
| synthesis.b128.torgb | 132099 | - | [4, 3, 128, 128] | float16 |

## 5.5 GAN Modelinin Eğitimine Devam

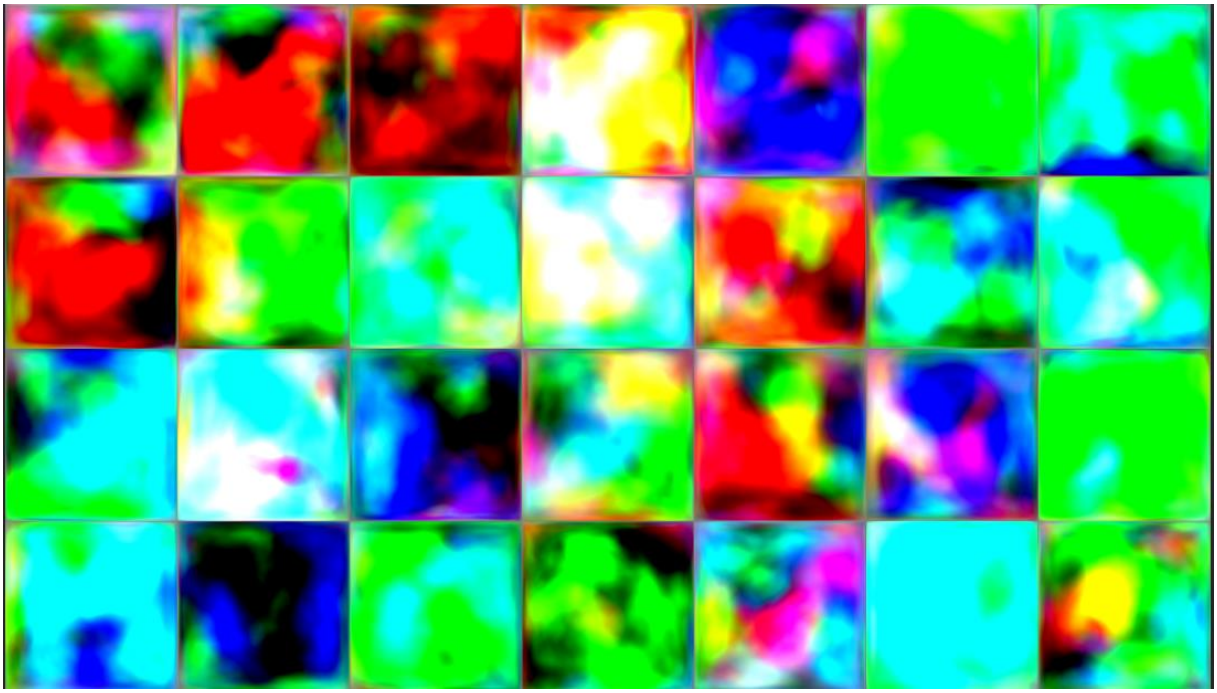
"EXPERIMENTS" değişkeni, deneylerin kaydedildiği klasörü temsil eder. "NETWORK" değişkeni, önceki eğitim noktasını temsil eden ağ ağırlık dosyasının adını içerir. "RESUME" değişkeni, önceki eğitim noktasını tam olarak tanımlayan dosya yolunu oluşturmak için diğer değişkenlerle birleştirilir.

"DATA" değişkeni, GAN modelinin eğitiminde kullanılacak olan veri kümesinin konumunu temsil eder. "SNAP" değişkeni, belirli bir eğitim aşamasında model ağırlıklarının kaydedileceği aralığı belirler.

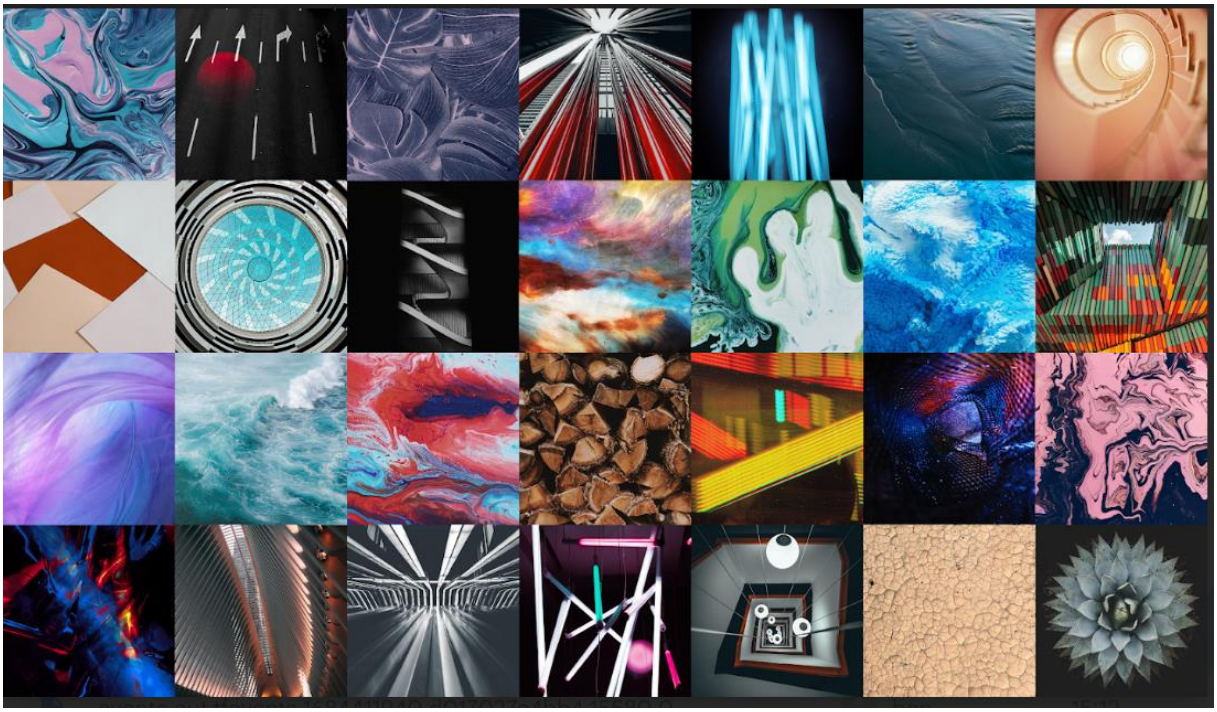
Son olarak, oluşturulan komut, GAN modelinin eğitimini başlatmak için kullanılır. Bu komut, "/content/stylegan2-ada-pytorch/train.py" betiğini çalıştırır ve ilgili argümanları geçer. Argümanlar, belirli bir aralıkta model ağırlıklarının kaydedilmesini, önceki eğitim noktasından devam etmeyi, deney sonuçlarının kaydedileceği çıktı dizinini ve kullanılacak veri kümesinin konumunu belirtir.

Bu şekilde, GAN modelinin eğitim sürecini belirli bir eğitim noktasından devam ettirebilir ve deney sonuçlarını belirli bir çıktı dizinine kaydedebilirsiniz.

### 5.6.1 Model Tarafından Üretilen Sahte (Olarak Kararlaştırılmış) Resimlerden Görüntü



### 5.6.2 Gerçek (Olarak Kararlaştırılmış ) Görüntüler :



## KAYNAKÇA


- http://gaugan.org/gaugan2/
- https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html
- https://sarkac.org/2020/09/yapay-zeka-ile-gercekligin-uretimi/
- https://gretel.ai/
- https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf
- https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html
- https://dergipark.org.tr/tr/download/article-file/942990
- https://www.youtube.com/watch?v=R546LYsQk5M&t=197s
- https://chat.openai.com/
- Youtube videolarım oluşturulan yüzlere ait görüntüler : https://www.youtube.com/shorts/xk4RoKQ8pms

  https://www.youtube.com/shorts/K25TzeBBHlw