

TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



**GENETİK ALGORİTMA İLE RASTGELE ANLAMLI
GÖRÜNTÜ ÜRETİMİ**

16011059 – Ahmet Onur AKMAN

BLM4510 - YAPAY ZEKA DERSİ DÖNEM ÖDEVİ

Doç. Dr. Mehmet Fatih Amasyalı

Haziran, 2021

İÇİNDEKİLER

ŞEKİL LİSTESİ	iii
TABLO LİSTESİ	iv
1 Giriş	1
1.1 Problem	1
1.2 Benzer Çalışmalar	1
1.3 Programlama Ortamı	2
2 Tanımlamalar	4
2.1 Genetik Algoritmalar	4
2.1.1 Fitness Fonksiyonları	5
2.1.2 Crossover	5
2.1.3 Mutasyon	6
2.2 Kullanılan Derin Öğrenme Modelleri	7
2.2.1 Keras Applications	7
3 Yaklaşım	9
3.1 Değişkenler	9
3.2 Görseller	9
3.3 Fitness Fonksiyonları	10
3.4 Mutasyonlar	11
3.5 Crossover için Birey Seçimi	12
3.6 Başarı Tespiti	12
3.7 Kütüphaneler	12
4 Program Çıktıları	13
4.1 Deneme #1	13
4.2 Deneme #2	15
4.3 Deneme #3	16
4.4 Deneme #4	17
4.5 Deneme #5	19

5 Sonuç	21
5.1 Gözlem	21
5.2 Çıkarım	22
5.3 Geliştirilebilirlik & İyileştirme Önerileri	23
5.3.1 Başlangıç Noktası	23
5.3.2 Bitiş Noktası	24
5.3.3 Fitness Fonksiyonu	24
5.3.4 Daha Güçlü Donanım, Daha Fazla Deneme	25
Referanslar	26

ŞEKİL LİSTESİ

Şekil 1.1	Spyder, the Scientific Python Development Environment.	3
Şekil 1.2	Google Colaboratory	3
Şekil 2.1	Genetik Algoritma Akış Şeması	5
Şekil 2.2	Bazı Crossover Çeşitleri	6
Şekil 2.3	Reproduction Sürecinde Crossover ve Mutasyon [3]	7
Şekil 3.1	İlk Versiyonda Kullanılan Fitness Skoru Hesaplayıcı	10
Şekil 3.2	İkinci ve Üçüncü Versiyonlarda Kullanılan Fitness Skoru Hesaplayıcı	11
Şekil 4.1	Deneme #1 İçin Her Nesilde En İyi Bireyler ve Ortalama Skorlar	14
Şekil 4.2	Deneme #1 İçin Son Nesile Kadar Oluşturulmuş Bazı Görseller .	14
Şekil 4.3	Deneme #2 İçin Her Nesilde En İyi Bireyler ve Ortalama Skorlar	15
Şekil 4.4	Deneme #2 İçin Son Nesile Kadar Oluşturulmuş Bazı Görseller .	16
Şekil 4.5	Deneme #3 İçin Son Nesile Kadar Oluşturulmuş Bazı Görseller .	17
Şekil 4.6	Deneme #4 İçin Her Nesilde En İyi Bireyler	18
Şekil 4.7	Deneme #4 İçin Son Nesile Kadar Oluşturulmuş Bazı Görseller .	18
Şekil 4.8	Deneme #5 İçin Her Nesilde En İyi Bireyler ve Ortalama Skorlar	19
Şekil 4.9	Deneme #5 İçin Son Nesile Kadar Oluşturulmuş Bazı Görseller .	20

TABLO LİSTESİ

Tablo 2.1 Bu Projede Kullanılan Modeller	8
Tablo 4.1 Deneme #1 Parametreleri	13
Tablo 4.2 Deneme #2 Parametreleri	15
Tablo 4.3 Deneme #3 Parametreleri	16
Tablo 4.4 Deneme #4 Parametreleri	17
Tablo 4.5 Deneme #5 Parametreleri	19

1 Giriş

Bu bölümde, proje kapsamında ele alınan problem ve bu problemin yapı taşları masaya yatırılacaktır.

1.1 Problem

Bu çalışmada, genetik algoritmalar ile rastgele görüntü üretimi problemi ele alınmıştır. Bu görüntü üretiminde ortaya çıkan görsellerin insan algısıyla tanımlanabilmesi veya en azından benzetilebilmesi hedeflenmiştir. Genetik algoritmaların öngörülemez çeşitliliğinin, eğitilmiş ve yüksek başarıya sahip yapay sinir ağı modellerinin obje tespit yeteneklerinin yardımıyla yönlendirilmesi yöntemiyle çalışılarak bu problem için bu yaklaşımın sonuç verip veremeyeceği araştırılmıştır.

Sürecin en başından daha nitelikli bir tanımlama yapmak gerekirse, yürütülen genetik algoritmanın en baştaki değişen büyülükteki rastgele/düz siyah görüntü üzerinde mutasyonlar oluşturarak bu anlamsızlığı kırması, üzerinde mutasyon ile piksel değişimleri gerçekleşen görsellerin yapay sinir ağları tarafından prediction sürecine sokulması sonucu bir prediction skoru elde edilmesi, prediction skoruna sahip her birey içerisinde bir objeye benzerliği en yüksek olan görsellerin crossover ile birleştirilmesi ve sonraki nesile aktarım yapılması planlanmıştır. Bu süreç belirlenen nesil sayısı boyunca bir döngü olarak ilerlemesi ve her nesilde elde edilen prediction skorlarının iyileşmesi amaçlanmıştır. Buna paralel olarak belli nesil numaralarında ekrana yazdırılan görsellerde giderek belirginleşen figürlerin görülmesi istenmiştir. Bu yol ile bir bilgisayarın daha önce var olmamış bir görseli oluşturulması mümkün müdür sorusuna yanıt aranmıştır.

1.2 Benzer Çalışmalar

Genetik algoritmalar ile görüntü üzerinde işlemler yapmak nadir bir uygulama değildir. Eldeki bir görsel üzerinde kalite artırma, segmentasyon, dönüştürmeler

veya bozulmuş görseli geri getirmek gibi işlemler hakkında literatürde çalışmalar mevcuttur. Bunlara örnek vermek gerekirse, C-H Huang ve JL Wu tarafından 2000 yılında yayınlanmış "A Watermark Optimization Technique Based on Genetic Algorithms" görsel üzerindeki hakların korunması amacıyla genetik algoritmaları kullanarak daha optimal ve daha güvenli bir watermarking yöntemi önermektedir. [1]

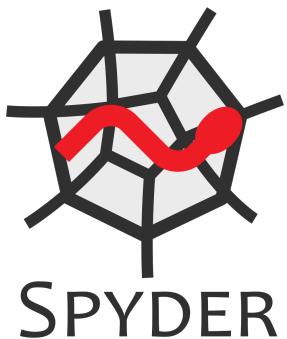
Bu ve bunun gibi birçok çalışma her ne kadar benim gibi benzer konular üzerine çalışmış olanlar için ilham verici oldusaya da, yine de literatürde bu problemle aynı veya bu probleme önemli ölçüde benzer bir çalışma bulunamamıştır. Akademik çalışmaların yanında diğer programlama domainindeki sitelerde yapılan araştırmalarda da bu durum pek değişmese de, aşağıdaki linki verilmiş çalışma bir ölçüde benzerlik göstermektedir.

- www.heartbeat.fritz.ai/reproducing-images-using-a-genetic-algorithm-with-python-91fc701ff84

Ahmed Gad tarafından linkte sunulmuş çalışmada, rastgele üretilmiş bir nesilden başlayan genetik algoritma rejenerasyon süreci, target olarak seçilmiş bir görselle benzer bir resim oluşturmayı amaçlamaktadır. Bu çalışmada ele alınmış durumun aksine belirli bir hedefe doğru gidilmektedir ve ilerleme dolayısıyla çok daha hızlı olacaktır. Fakat bu çalışmada bile eldeki bir görseli andıran bir resmin ortaya çıkışı 15,000 nesil sürmüştür.

1.3 Programlama Ortamı

Bu proje kapsamında, hem şahsi aşinalığım, hem daha önce yaptığım benzer çalışmalarındaki konfor ve başarı oranları, hem kütüphane desteğinin zengin olması, hem de veri görselleştirme konusunda yeterli desteği sağlanması sebebiyle programlama dili Python, IDE olarak ise bu konuda önde gelen seçimlerden biri olan Anaconda Spyder kullanılmıştır.



Şekil 1.1 Spyder, the Scientific Python Development Environment.

Bunun yanı sıra, sözkonusu program yapısında işlenecek verinin büyüklüğü ve bu veri üzerine yapılacak matematiksel işlemlerin fazlalığı daha güçlü bir donanımsal güce ihtiyaç doğurmuştur. Bu ihtiyaç, çalışmanın belli zaman dilimlerinde, Google Colaboratory'nin sunduğu ücretsiz sınırlı süreli GPU desteği ile giderilmiştir, fakat projenin genel akışı (özellikle de Colab'in ücretsiz hesap limitasyonları sebebiyle) Anaconda Spyder'da yürütülmüştür.



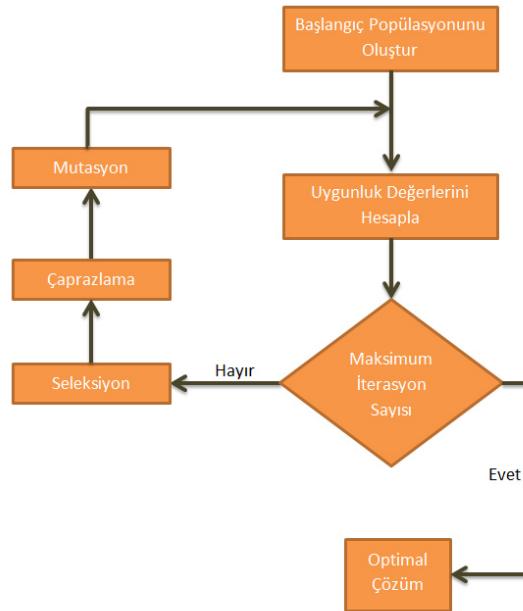
Şekil 1.2 Google Colaboratory

2 Tanımlamalar

Bu bölümde çalışmada adı geçen bazı temel terimlerin tanımlaması yapılacaktır.

2.1 Genetik Algoritmalar

Prof. Whitley'in "A genetic algorithm tutorial" adındaki yayınında [2] genetik algoritmalar temel olarak evrimsel yaklaşımı sahip, kromozom benzeri veri yapılarının kombinasyonlarının alınmasıyla ilerleyen, genel olarak fonksiyon optimize edici olarak görülse de daha geniş çaplı başka problemlerde de başarılı sonuç verebilen algoritmalar olarak tanımlanmıştır. Bu tanım bu algoritmaların konseptini başarılı bir şekilde yansıtmaktadır. Farklı bir perspektiften tanımlama yapmak gerekirse, genetik algoritmaların başlangıçta elimizde bulunan ve elimizdeki problemin çözümü için ideal sonuçlar vermeyen verilerin, problemin çözümünde kullanılabilmesi adına doğal seçim benzeri bir yaklaşımla elenerek veya seçilerek bir sonraki versiyonlarının üretimi için kullanılması da eklenebilir. Bu seçme, kombinleme ve bir sonraki nesil için çeşitlilik/farklılık oluşturma işlevlerinin konseptlerini daha iyi kavramak adına, genetik algoritmaların kapsamındaki diğer terimleri de yakından tanımk gereklidir. Bu bölümde özellikle bu çalışmanın içeriğinde bulunan terimlere odaklanılacaktır.



Şekil 2.1 Genetik Algoritma Akış Şeması

2.1.1 Fitness Fonksiyonları

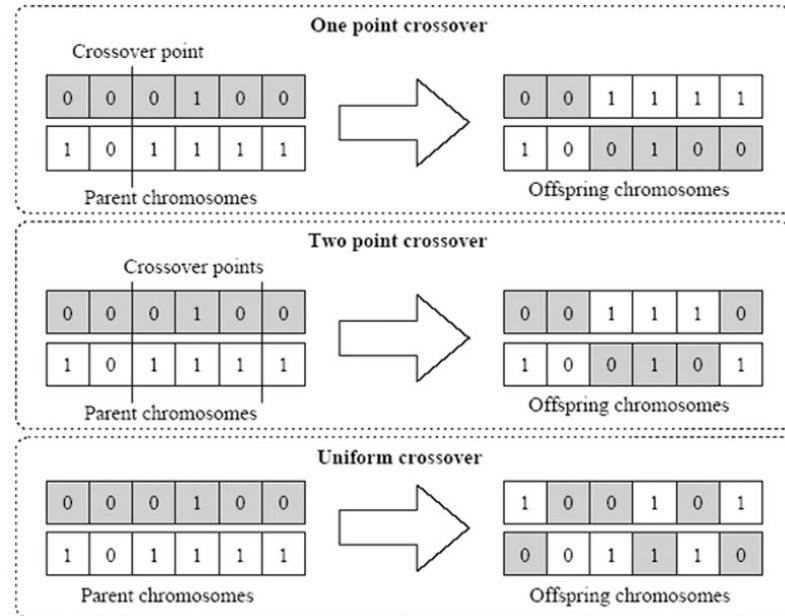
Genetik algoritmalarla fitness fonksiyonları, her nesilde ortaya çıkan yeni değerlerin problem çözümü için optimallığını ölçmek, bu optimallığı skorlamak ve bir sonraki neslin reproduksyonunda kullanılacak değerlerin seçiminde bu skorlar vasıtıyla faktör yaratmak amacıyla kullanılır. Fonksiyon formülizasyonları problem çözümünde aranan kriterlere bağlı olarak oluşturulur. Birden fazla fitness fonksiyonu kullanılan senaryolarda, bu fonksiyonların verdiği çıktıların kombinasyonuyla tek bir skor oluşturulabilir. Eğer her fonksiyonun önem derecesi farklısa ağırlıklandırma kullanılabilir.

Bu çalışma kapsamında hem çoklu hem de tek bir fitness fonksionunun kullanımı gösterilmiştir. Ancak fitness fonksiyonunun formülizasyonu açık bir formülizasyondan ziyade, kullanılan yapay sinir ağlarında gerçekleşen hesaplamalardan oluşmaktadır, oluşan fitness skoru ise bu hesaplamalar sonucu ortaya çıkan prediction skoru olmaktadır.

2.1.2 Crossover

Bir nesildeki bireylerin seçilmesinden sonra seçilen bireylerle bir sonraki neslin üretilmesi süreci, crossover yaklaşımı ile gerçekleştirilir. Biyolojideki karşılığuna benzer olarak, kromozom metaforuyla belirtilen iki değer kümesi, iki veya daha çok yerden kesilir. Bu kesimden sonra oluşan parçalar kombine edilerek, yeni birey veya

bireyler oluşur. Bu yeni birey oluşturma sürecinde iki kromozomdan bir veya daha fazla yeni değer kümesi oluşturulabilir.



Şekil 2.2 Bazı Crossover Çeşitleri

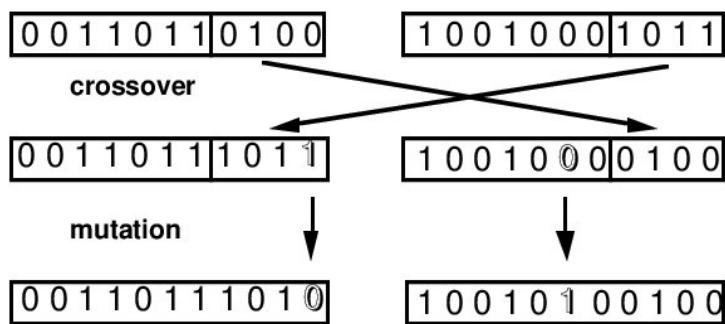
Bu çalışma özelinde yeterli olacağı düşünüldüğü için geleneksel Crossover yöntemi takip edilmiştir. Bu yöntemin yanı sıra farklı yöntemlerle yapılan ve bazı dezavantajları ortadan kaldırınan başka Crossover yöntemleri örneklenirilebilir.

2.1.3 Mutasyon

Crossover'ın bize bir nesildeki bireyleri farklı kombinlerde kullanarak farklı nitelikteki bireyleri oluşturma şansı verdiğinden bahsedilmiştir. Ancak Crossover öncesi yapılan seçim, özellikle algoritma yüksek jenerasyon iterasyonlarına ulaştığında, jenerasyon içerisindeki birey çeşitliliği azalması durumunda fazla çeşitlilik sağlayamayacak duruma gelebilmektedir. Bu da çeşitlilik için kullanıyor olduğumuz Crossover yaklaşımının çok benzer veri kümeleri arasında gerçekleştirilmesi sonucu, çeşitliliğin yitirilmesi durumunu oluşturabilir.

Hem programın erken evrelerinde, hem de bahsedilen geç evrelerde bize çeşitlilik, daha doğrusu farklılık kazandıracı bir yaklaşım da mutasyonlardır. Mutasyonlar, önceden belirlenen oranlar ile elimizdeki veri kümesine uygulanır ve belirtilen oranla orantılı sayıdaki birim veri rastgele bir değerle değiştirilir. Bu mutasyon oranı önceden belirlendiği gibi sabit kalabileceği gibi, program optimal sonuca yaklaştıkça azaltılabilir.

Mutasyonun sağladığı çeşitlilik bize çözümü arayış yolculuğunda algoritmanın belki de eldeki durumda bulamayacağı çözüm yollarına ulaşma ihtimalini getirir. Böylelikle jenerasyon değişiminde bir süre sonra gözlemlenmesi muhtemel sabitlik kırılabilir ve dilendiği takdirde program akışına dinamiklik kazandırılabilir. Mutasyon oranının programın çıktılarının sonucuna yaptığı direk etki, bu dokümanın ilerleyen kısımlarında daha detaylı olarak ele alınacaktır.



Şekil 2.3 Reproduction Sürecinde Crossover ve Mutasyon [3]

2.2 Kullanılan Derin Öğrenme Modelleri

Bu çalışmada derin öğrenme ağlarının fitness fonksiyonu olarak kullanılmasından ötürü bu konuya da bir parantez açılması uygun görülmüştür. Ancak bu kullanım hiçbir şekilde eğitim gibi karmaşıklık oluşturabilecek bir durum içermediğinden ötürü, bu konu üzerinde genetik algoritmalarla durulduğu kadar durulmayacak, yalnızca genel bir çerçevede bahsedilecektir.

Kullanılan ağların seçilmesinde en önemli etken öncelikle doğruluk oranı olmakla birlikte, çoklu sinir ağı kullanılan versiyonlar için hem lightweight olan, hem doğruluğu yüksek olan, hem de input size'sı değiştirmeden aynen uyumlu olarak kullanılabilen ağların seçilmesine özen gösterilmiştir.

2.2.1 Keras Applications

Keras Applications, programlara hazır bir şekilde implemente edilmeye hazır, prediction, feature extraction ve fine-tuning amaçları ile kullanılabilen derin öğrenme ağlarından oluşmaktadır. Bu ağırlıklar dilendiği takdirde pre-trained ağırlıkları yüklenerek kullanılabilir.

Buradan alınan ağlar hazır ağırlıkları yüklenikten sonra yalnızca prediction yapmak için kullanılmıştır. Keras Applications web sayfasında sunulmuş tabloda verilen

bilgilerden, bu çalışmada adı geçen modellerin seçiliip çıkarılmış hali Tablo-2.1'te verilmiştir.

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet169	57 MB	0.762	0.932	14,307,880	169
NASNetMobile	23 MB	0.744	0.919	5,326,716	-

Tablo 2.1 Bu Projede Kullanılan Modeller

Bu tablonun tamamına aşağıdaki linkten erişilebilir.

- www.keras.io/api/applications/

3

Yaklaşım

Bu bölümde çalışma boyunca temel olarak izlenen yöntemlerden ve bu yöntemlerin seçimindeki motivasyonlardan bahsedilecektir.

3.1 Değişkenler

Çalışma kapsamında oluşturulan program yapısı her versiyon için korunurken, hem versiyonlar arası hem de bir versiyon içerisinde yapılan farklı denemelerde etki durumlarını daha iyi izleyebilmek için aşağıdaki parametre ve diğer tür değişkenlerin düzenli olarak yeni değerlerle denenmesi sözkonusu olmuştur.

- Popülasyon Büyüklüğü
- Nesil Sayısı
- Mutasyon Oranı
- Crossover kesim noktaları sayısı
- Fitness olarak kullanılacak modeller ve model sayısı
- Başlangıç popülasyonundaki görsellerin nitelikleri
- Her iterasyonda yeni nesile aynen kopyalanacak en iyi bireylerin sayısı

Bu değişkenler üzerinde yapılan denemeler ve sonuçları grafikler ve görsellerle desteklenerek sunulacaktır.

3.2 Görseller

Yapılan araştırmalarda, bu tarz çalışmalarında kullanılan görsellerin tek boyutlu, kromozom tarzı bir yapıda kullanıldığı görülmüştür. Ancak bu çalışmada, görsellerin

niteliklerini sürekli değiştirmekten kaçınmak adına görseller hiç bozulmadan iki boyutlu matris olarak saklanmıştır. Crossover ve mutasyon senaryoları da bu iki boyutlu diziler üzerinden yürütülmüştür.

Her bir program versiyonunda, başlangıç neslinin nitelikleri nasıl değiştirilirse daha iyi sonuçlar alınabilir sorusuna yanıt aranmıştır. Bu bağlamda üç farklı program versiyonu içinde iki tanesinde ilk bireyler rastgele piksellerden oluşurken, bir tanesinde düz siyah görseller üzerinde algoritmanın yapacağı değişikliklerin net bir şekilde gözlenmesi hedeflenmiştir.

Önceki bölümde fitness fonksiyonunda kullanılacak derin öğrenme modellerinden bahsedilmişti. Bahsedilen 3 program versiyonundan birinde yalnızca bir adet, ikisinde ise üç derin öğrenme ağı aynı anda kullanılmıştır. Bu ağların giriş katmanları adapte edilebilecek olsa da, yapısını bozmadan implemente etmek istenmiştir. Bu sebeple diğer iki versiyondaki görsel boyutu, ilk versiyondaki 299x299'dan farklı olarak, MobileNet, DenseNet ve NasNet'e uygun olacak şekilde 224x224 olarak alınmıştır.

3.3 Fitness Fonksiyonları

Bu çalışmadaki versiyonların ilkinde yalnızca bir adet derin öğrenme ağı seçilmiş ve bu ağa verdiği sonuçlar fitness skoru olarak değerlendirilmiştir. Bu ağ InceptionResNetV2 olmuştur. Tek modelin kullanıldığı versiyon için fitness skorunun hesaplandığı detector metodu Şekil-3.1'te verilmiştir. Buradaki kullanılan ppi fonksiyonları Keras Applications bünyesindeki ağların her biri için tanımlanmış preprocess_input için kısaltılmış bir kullanımıdır.

```
def detector(img):
    img = img_to_array(img)
    img = np.expand_dims(img, axis=0)

    img = ppi_ires(img)

    preds = model.predict(img)
    top_pred = decode_predictions(preds, top=1)[0][0][1:]
    return top_pred
```

Şekil 3.1 İlk Versiyonda Kullanılan Fitness Skoru Hesaplayıcı

Geriye kalan diğer iki versiyonda (sebebi sonraki bölümlerde gösterilecek) bu yaklaşımından vazgeçilmiş ve bu tek sinir ağı modeli üç farklı ağı ile değiştirilmiş ve bu ağların verdiği predictionların ortalaması fitness fonksiyonu olarak değerlendirilmiştir. Bu ağlar MobileNetV2, DenseNet169, NASNetMobile olmuştur.

```

def detector(img):
    img = img_to_array(img)
    img = np.expand_dims(img, axis=0)

    img1 = mn_ppi(img)
    img2 = dn_ppi(img)
    img3 = nn_ppi(img)

    preds = [models[0].predict(img1),
             models[1].predict(img2),
             models[2].predict(img3)]

    top_preds = [mn_decode(preds[0], top=1)[0][0][1:],
                 dn_decode(preds[1], top=1)[0][0][1:],
                 nn_decode(preds[2], top=1)[0][0][1:]]

    score = (float) (np.mean([p[-1] for p in top_preds]))

    return top_preds[0][0]+__+top_preds[1][0]+__+top_preds[2][0], score

```

Şekil 3.2 İkinci ve Üçüncü Versiyonlarda Kullanılan Fitness Skoru Hesaplayıcı

Önceki bölümde de belirtildiği üzere, kullanılan ağların seçilmesinde en önemli etken öncelikle doğruluk oranı olmakla birlikte, çoklu sinir ağı kullanılan versiyonlar için hem lightweight olan, hem doğruluğu yüksek olan, hem de input size'ı değiştirmeden aynen uyumlu olarak kullanılabilecek ağların seçilmesine özen gösterilmiştir. Lightweight ağların kullanılmasıyla çalışma süresi oldukça uzun olan bu algoritmanın performansını düşürmemek hedeflenmiştir.

Sonuç olarak adı geçen bütün ağlar aşağıda listelenmiştir. Bütün modeller "imagenet" pre-train ağırlıkları ile kullanılmıştır.

- InceptionResNetV2
- MobileNetV2
- DenseNet169
- NASNetMobile

Bu anlatımdan da anlaşılabileceği üzere, fitness skorunun maksimize edilmesi hedeflenmiştir.

3.4 Mutasyonlar

Her bir birey için, tuttuğu değerler üzerinde belirlenen mutasyon oranına göre hesaplanmış piksel sayısı kadar rastgele değişim yapılması sağlanmıştır. Böylelikle tekdüzeligin önüne geçilmiş, tıkanıklık oluşmaması sağlanmaya çalışılmıştır.

3.5 Crossover için Birey Seçimi

Reproduksiyon sürecinde Crossover için birey seçiminde, bireylerin fitness fonksiyonu skorlarının göz önünde bulundurulduğu söylemiştir. Bu seçimin yapılması sürecinde tam olarak bu skorların birbirine oranının takip edilmesi yerine, sıralamalı seçim yöntemi takip edilmiştir. Bu yöntemle beraber, iyilik durumlarına göre sıralanan bireylerin seçim şansları, sıralanmış bu dizideki indislerine göre belirlenmiştir. Bu yolla tekdüzeliğin önüne geçilmesi hedeflenmiştir.

3.6 Başarı Tespiti

Başarı tespiti, elde edilen en optimal çözüm ve bu çözümün elde edilmesi sürecinde her nesildeki değer kümelerine uygulanan fitness fonksiyonlarından elde edilen skorların gözlemlenmesi ile yapılmaktadır. Gözlenen skorların yanı sıra, belli epoch sayısı sıklığında oluşan en iyi birey, .png dosyası olarak kaydedilmektedir, oluşan görüntü insan algısı ile değerlendirilebilmektedir.

3.7 Kütüphaneler

Kullanılan kütüphaneler ve kullanım amaçları şu şekildedir;

- numpy: Dizi işlemleri.
- random: Random değer ve dizi elemanı elde edilmesi.
- matplotlib: Grafik gösterimleri.
- PIL: Resim okuma/yazma.
- keras.preprocessing: Fitness için görsellerin uygun forma sokulması.
- keras.applications: Modellerin ve ağırlıklarının temini.

4

Program Çıktıları

Bu bölümde programın farklı senaryolarda verdiği başarı oranları paylaşılacaktır. Bu çıktılar bir sonraki bölümde değerlendirilecektir.

Deney süreci Google Colab ve Anaconda Spyder IDE ortamlarında yürütülmüştür. Lokalde yapılan denemelerde donanımsal limitasyonlar, Colab üzerinde yapılan denemelerde ise Colab'ın ücretsiz hesaplar için uyguladığı kısıtlamalar ile yüzleşilmiştir. Programın (özellikle lokalde) oldukça zaman alıcı olması, bu limitasyonların da eklenmesiyle birlikte, birçok türdeki denemenin ya yarıda kesilmesine ya da rafa kalkmasına neden olmuştur.

Bu duruma rağmen (başarılı) 8 farklı senaryoya göre denemeler yapılmış ve bunların arasından 5 tanesine bu rapor dahilinde yer verilmiştir. Bu senaryolarda çizilen grafikler, senaryolardaki parametreler ve algoritma ile oluşturulmuş görseller paylaşılacak, yapılan çıkarımlar bir sonraki bölümde verilecektir.

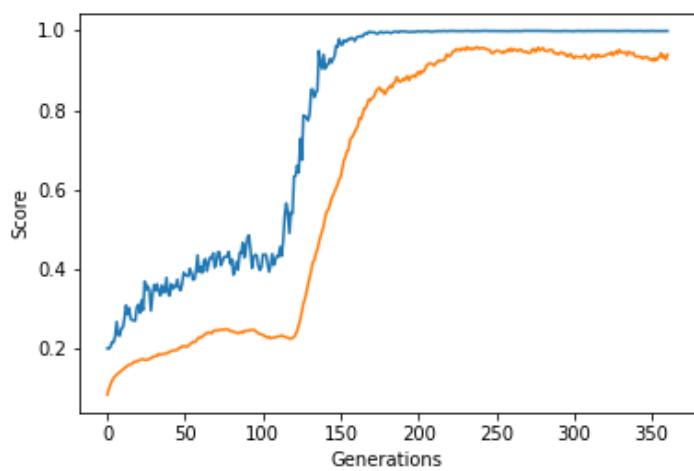
4.1 Deneme #1

Bu kısımda, çalışmanın deneme senaryoları arasından sunulacak 1. denemeden sonuçlar paylaşılacaktır. Süreç parametreleri Tablo-4.1'da verilmiştir.

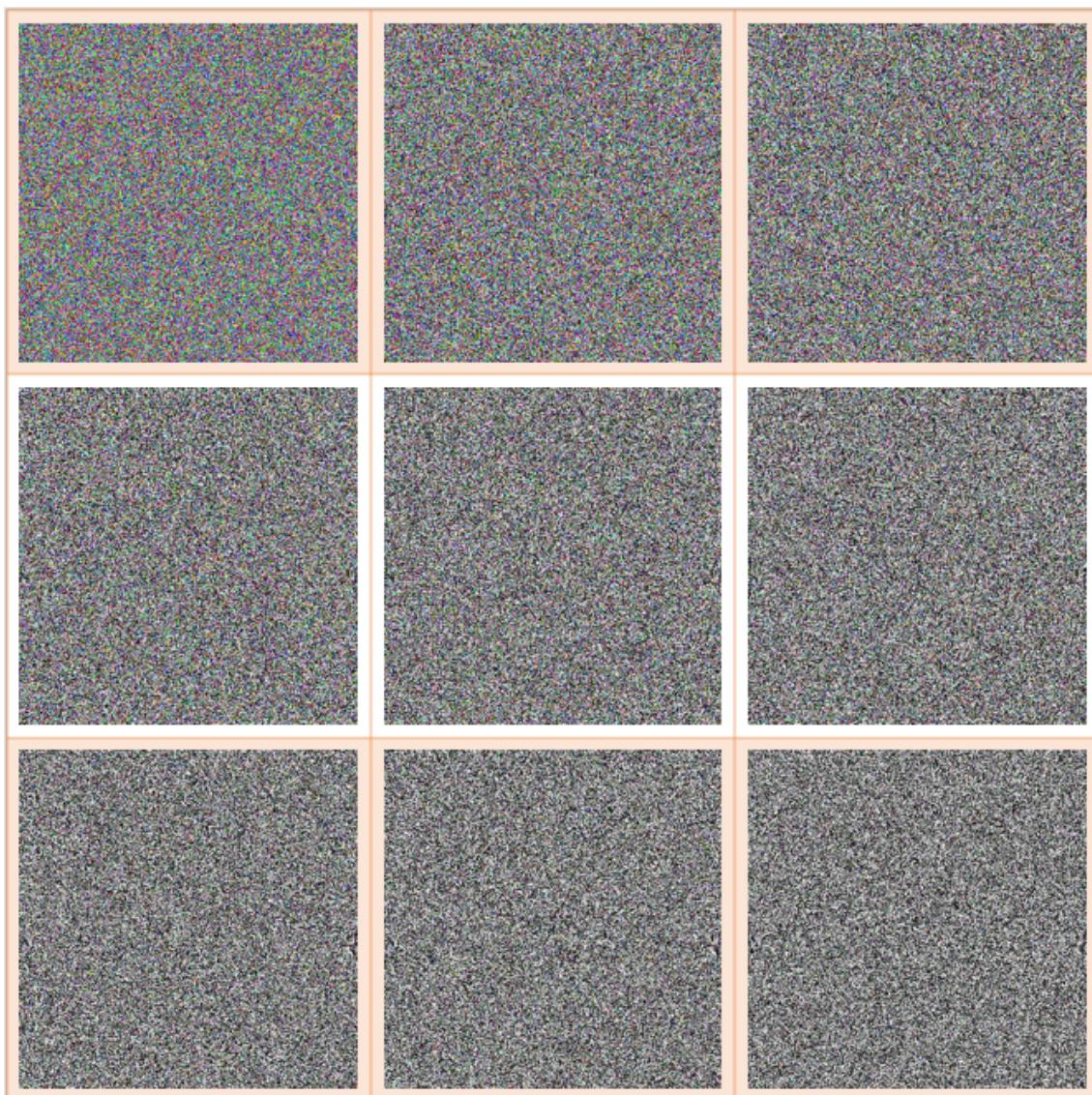
Deneme #	Ortam	Nesil Büyüklüğü	Nesil Sayısı	Mutasyon	Crossover Kesim
1	Colab	1000	360	0.01	1
Matris	İlk Nesil	Aktarım	Baş. En İyi Skor	Son En İyi Skor	Fitness Modelleri
299x299	Random Pikseller	1/3	0.2020	0.9988	InceptionResNetV2

Tablo 4.1 Deneme #1 Parametreleri

Deneme sonunda elde edilmiş, her nesil için en iyi bireyler ve nesildeki ortalama skorların yer aldığı grafiksel gösterim Şekil-4.1'de verilmiştir. Deneme boyunca oluşturulmuş bazı görseller, Şekil-4.2'de gösterilmiştir.



Şekil 4.1 Deneme #1 İçin Her Nesilde En İyi Bireyler ve Ortalama Skorlar



Şekil 4.2 Deneme #1 İçin Son Nesile Kadar Oluşturulmuş Bazı Görüşeller

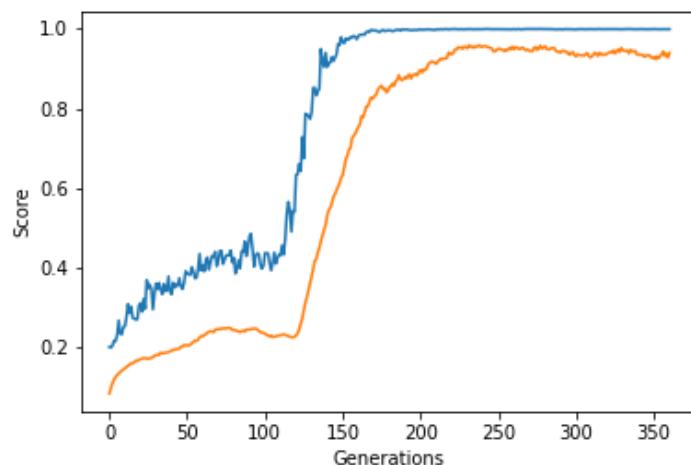
4.2 Deneme #2

Bu kısımda, çalışmanın deneme senaryoları arasından sunulacak 2. denemeden sonuçlar paylaşılacaktır. Süreç parametreleri Tablo-4.2'da verilmiştir.

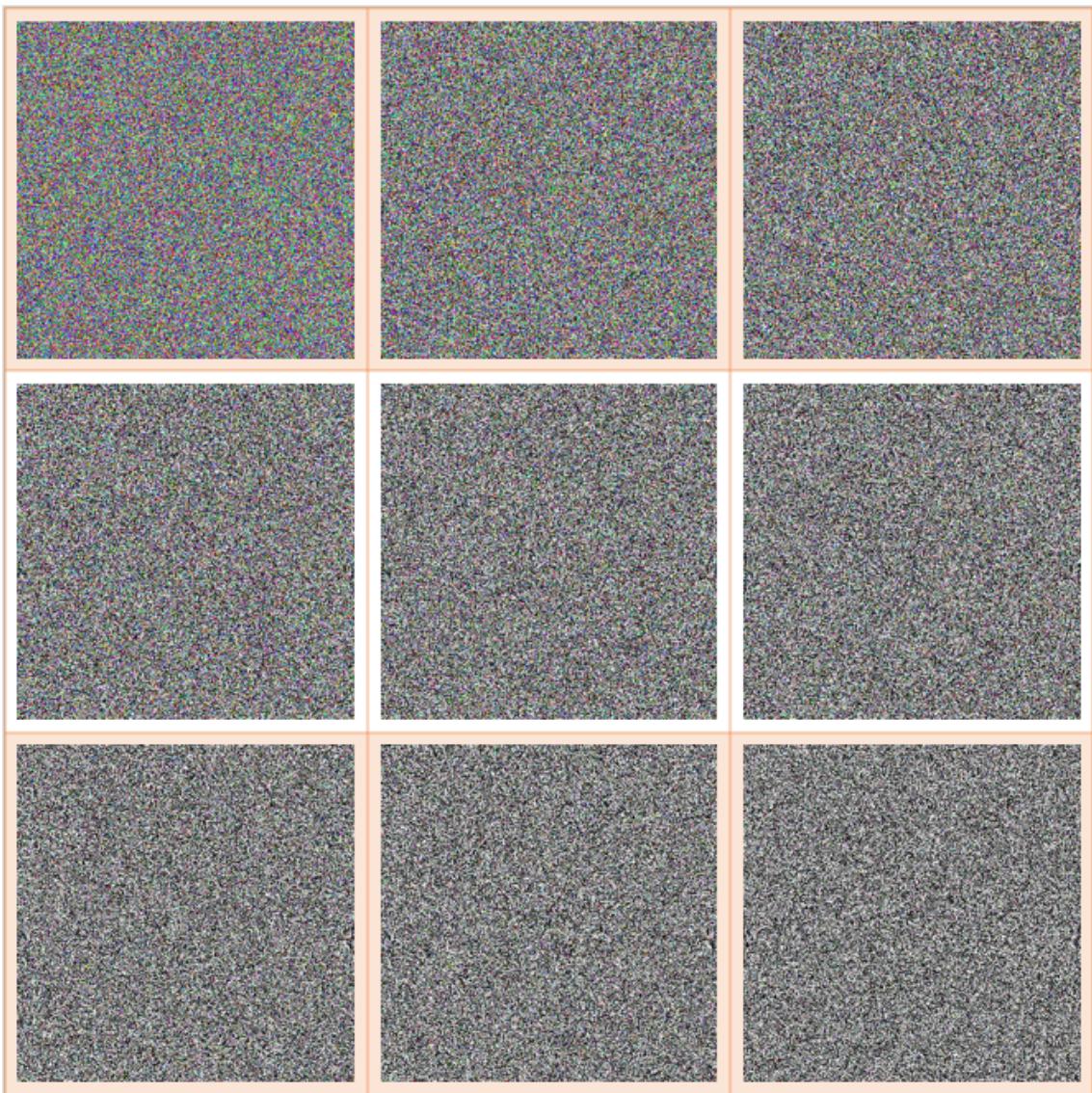
Deneme #	Ortam	Nesil Büyüklüğü	Nesil Sayısı	Mutasyon	Crossover Kesim
2	Colab	1000	430	0.01	2
Matris	İlk Nesil	Aktarım	Baş. En İyi Skor	Son En İyi Skor	Fitness Modelleri
299x299	Random Pikseller	1/3	0.1796	0.9873	InceptionResNetV2

Tablo 4.2 Deneme #2 Parametreleri

Deneme sonunda elde edilmiş, her nesil için en iyi bireyler ve nesildeki ortalama skorların yer aldığı grafiksel gösterim Şekil-4.3'de verilmiştir. Deneme boyunca oluşturulmuş bazı görseller, Şekil-4.4'de gösterilmiştir.



Şekil 4.3 Deneme #2 İçin Her Nesilde En İyi Bireyler ve Ortalama Skorlar



Şekil 4.4 Deneme #2 İçin Son Nesile Kadar Oluşturulmuş Bazı GörSELLER

4.3 Deneme #3

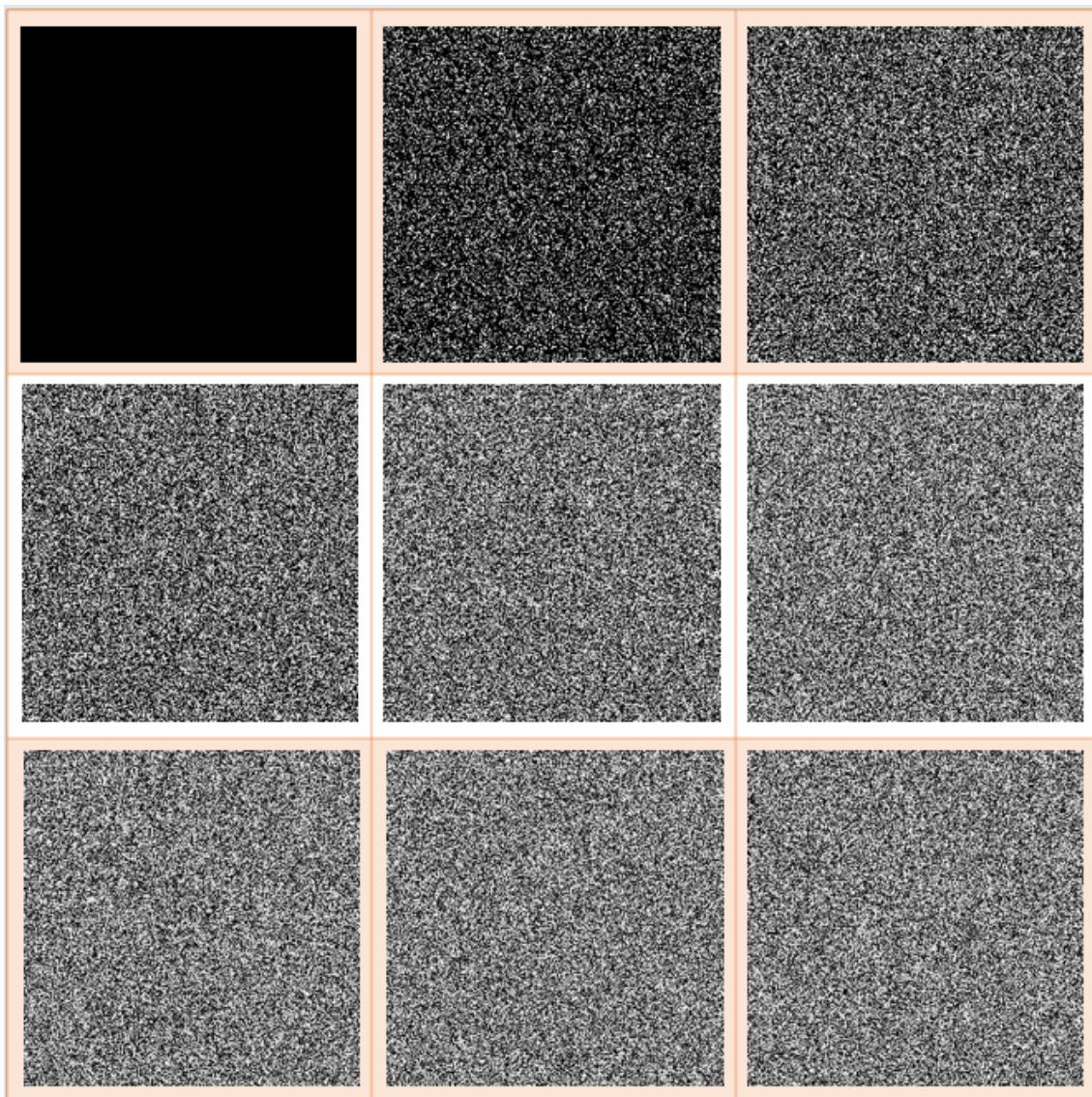
Bu kısımda, çalışmanın deneme senaryoları arasından sunulacak 3. denemeden sonuçlar paylaşılacaktır. Süreç parametreleri Tablo-4.3'da verilmiştir.

Deneme #	Ortam	Nesil Büyüklüğü	Nesil Sayısı	Mutasyon	Crossover Kesim
3	Colab	600	110	0.04	2
Matris	İlk Nesil	Aktarım	Baş. En İyi Skor	Son En İyi Skor	Fitness Modelleri
224x224	Siyah Pikseller	1/3	0.1557	0.1564	MobileNetV2, DenseNet169, NASNetMobile

Tablo 4.3 Deneme #3 Parametreleri

Bu denemedede 110 nesil sonunda fitness skorunda önemli bir ilerleme kaydedilemediği için skorlar için grafiksel gösterim yapılmayacaktır. Deneme boyunca oluşturulmuş

bazı görseller, Şekil-4.5'de gösterilmiştir.



Şekil 4.5 Deneme #3 İçin Son Nesile Kadar Oluşturulmuş Bazı Görseller

4.4 Deneme #4

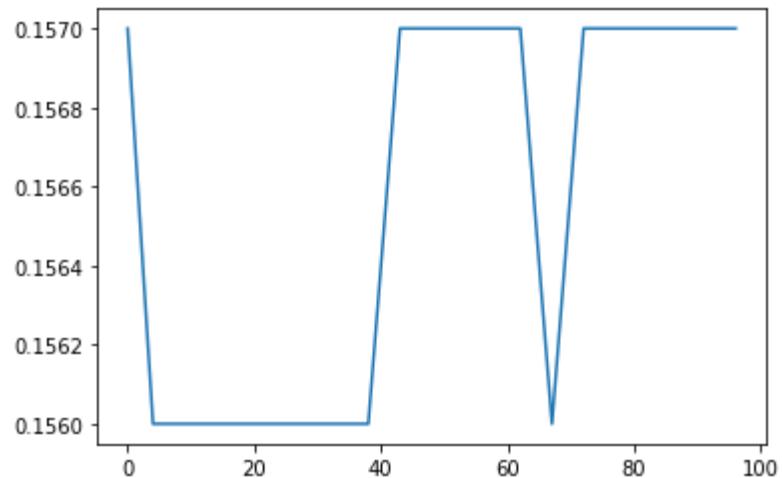
Bu kısımda, çalışmanın deneme senaryoları arasından sunulacak 4. denemeden sonuçlar paylaşılacaktır. Süreç parametreleri Tablo-4.4'da verilmiştir.

Deneme #	Ortam	Nesil Büyüklüğü	Nesil Sayısı	Mutasyon	Crossover Kesim
4	Colab	600	100	0.01	2
Matris	İlk Nesil	Aktarım	Baş. En İyi Skor	Son En İyi Skor	Fitness Modelleri
224x224	Siyah Pikseller	1/3	0.1557	0.1571	MobileNetV2, DenseNet169, NASNetMobile

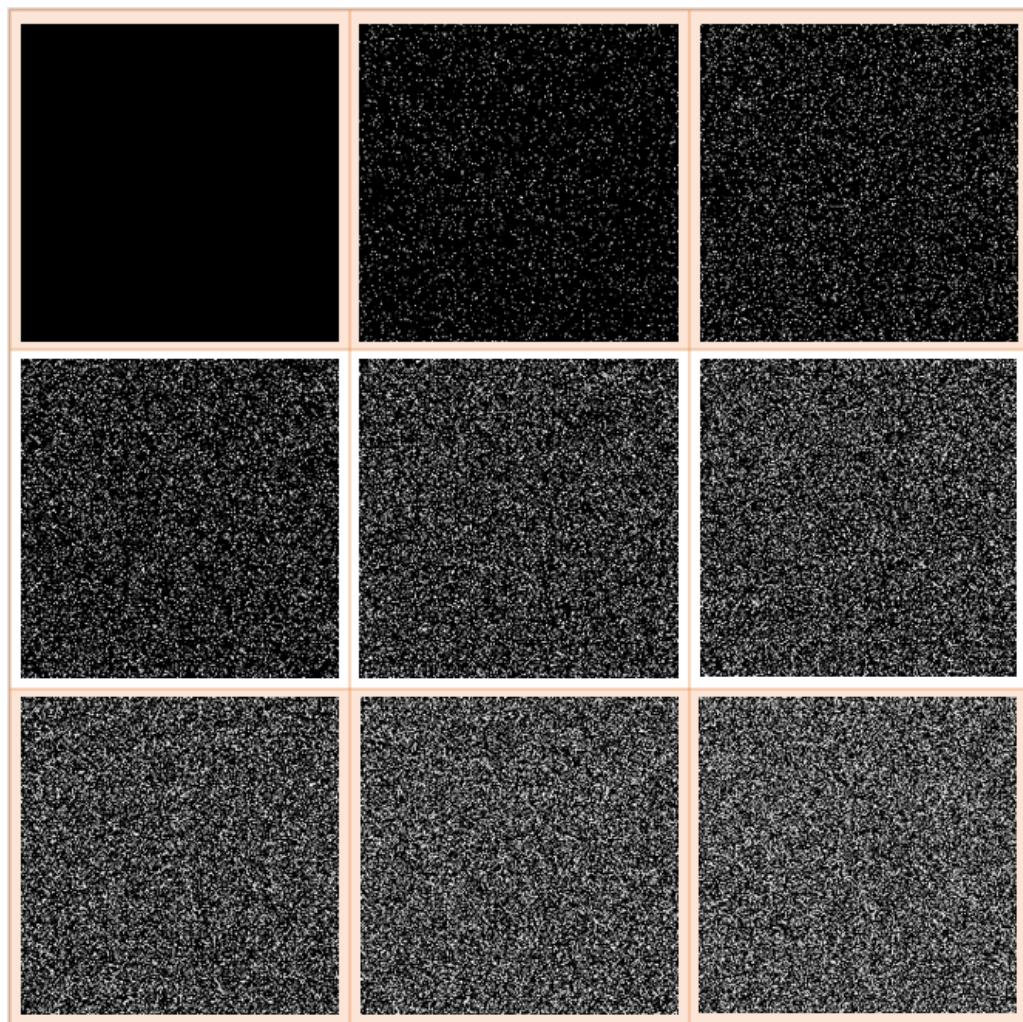
Tablo 4.4 Deneme #4 Parametreleri

Deneme sonunda elde edilmiş, her nesil için en iyi bireylerin yer aldığı grafiksel

gösterim Şekil-4.6'de verilmiştir. Deneme boyunca oluşturulmuş bazı görseller, Şekil-4.7'de gösterilmiştir.



Şekil 4.6 Deneme #4 İçin Her Nesilde En İyi Bireyler



Şekil 4.7 Deneme #4 İçin Son Nesile Kadar Oluşturulmuş Bazı Görseller

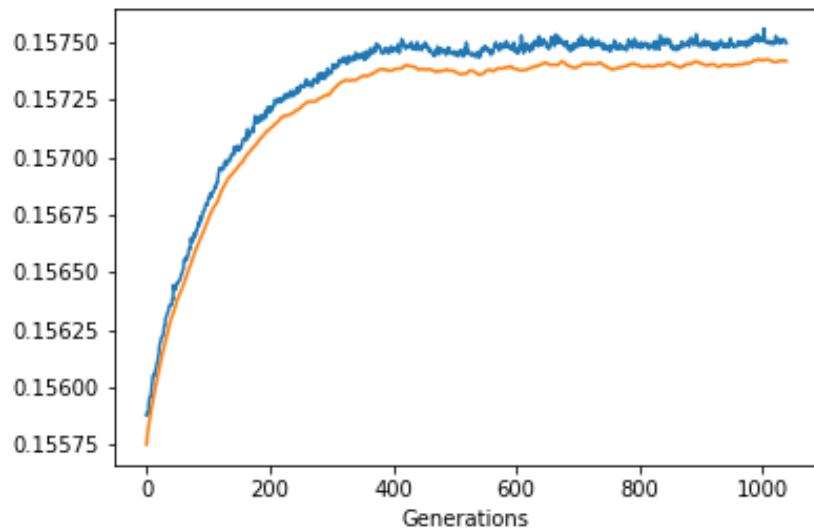
4.5 Deneme #5

Bu kısımda, çalışmanın deneme senaryoları arasından sunulacak 5. denemeden sonuçlar paylaşılmaktır. Bu bölümde anlatılacak deneme, lokalde başarıyla tamamlanabilen 2 denemeden bir tanesidir ve 1039. nesile gelmesi 6 gün 20 saat sürmüştür. Süreç parametreleri Tablo-4.5'da verilmiştir.

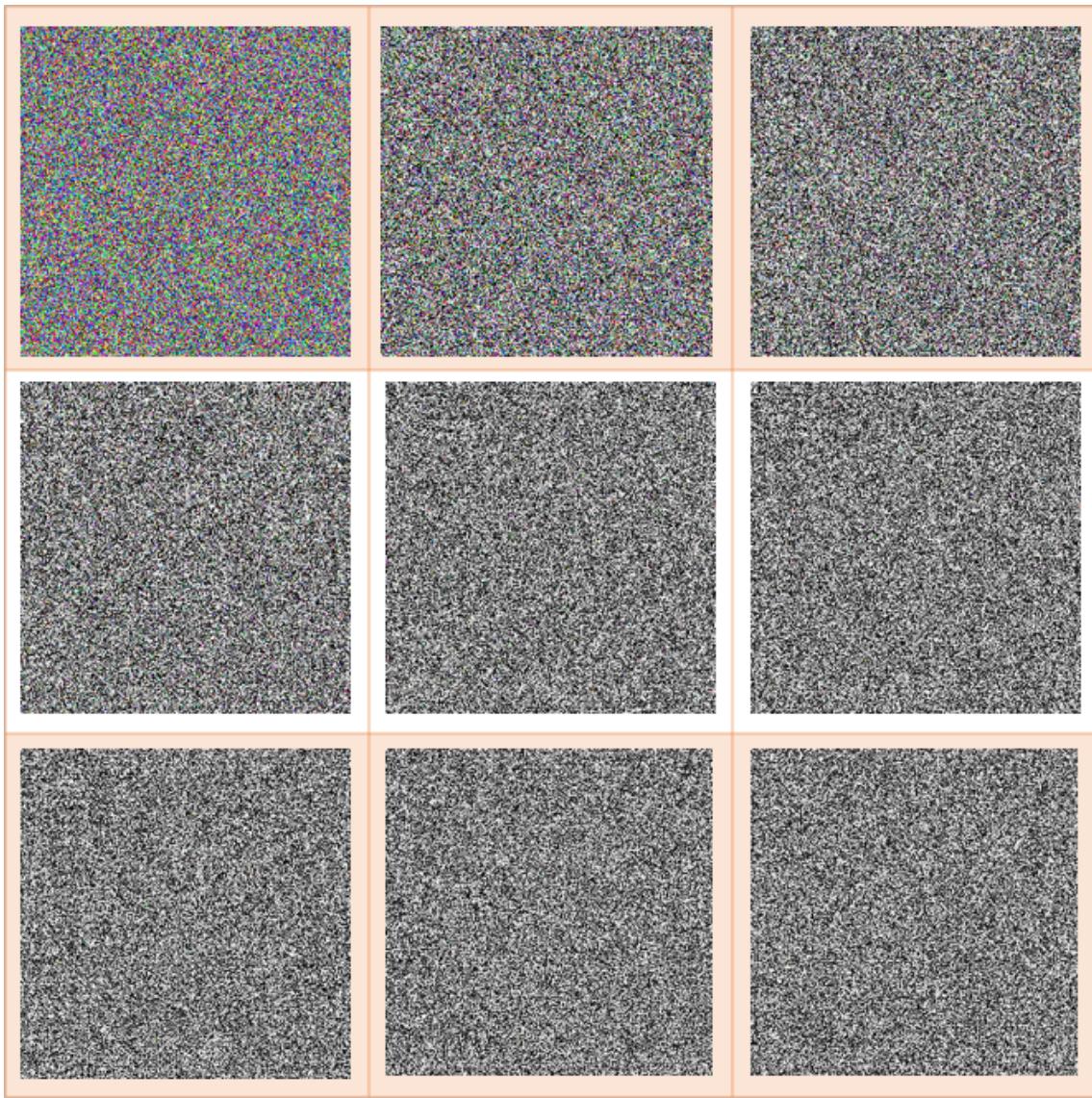
Deneme #	Ortam	Nesil Büyüklüğü	Nesil Sayısı	Mutasyon	Crossover Kesim
5	Spyder	800	1039	0.01	1
Matriks	İlk Nesil	Aktarım	Baş. En İyi Skor	Son En İyi Skor	Fitness Modelleri
299x299	Random Pikseller	1/3	0.1557	0.1574	MobileNetV2, DenseNet169, NASNetMobile

Tablo 4.5 Deneme #5 Parametreleri

Deneme sonunda elde edilmiş, her nesil için en iyi bireyler ve nesildeki ortalama skorların yer aldığı grafiksel gösterim Şekil-4.8'de verilmiştir. Deneme boyunca oluşturulmuş bazı görseller, Şekil-4.9'de gösterilmiştir.



Şekil 4.8 Deneme #5 İçin Her Nesilde En İyi Bireyler ve Ortalama Skorlar



Şekil 4.9 Deneme #5 İçin Son Nesile Kadar Oluşturulmuş Bazı GörSELLER

5

Sonuç

Bu bölümde programın farklı senaryolarda verdiği sonuçların nedenleri irdelenecektir.

Bu çalışma boyunca, başta konulan hedef tanımı ve sınırlamalara sadık kalınarak istenen sonuca ulaşıp ulaşlamayacağı konusunda çeşitli denemeler yapılmıştır. Yapılan her deneme sonunda, algoritmanın yetersiz kalmış olabileceği alanlar analiz edilmiş ve bu eksikliklerin giderilmesi için neler yapılabılır sorusuna yanıt aranmıştır. Alınan olumsuz sonuçlara karşılık parametreler değiştirilmiş, durum devam edince de çalışma süresi (değişiklikleri daha iyi gözlemleyebilmek için) oldukça uzun tutulan denemeler yapılmıştır. Farklı senaryolarda programın farklı davranışları gösterdiği görülmüştür. Ancak buna rağmen hedefe ulaşmak için gereken koşullar seçilen koşul ve parametrelerle sağlanamamıştır. Bu bölümde, bu durumun oluşma sebepleri irdelenecek ve bu çalışmaya yapılacak hangi modifikasyonlar ile daha iyi sonuçlar elde edileceğine dair fikirler beyan edilecektir.

5.1 Gözlem

Bir önceki bölümde gösterilmiş sonuçlar arasında oldukça ilgi çekici detaylar mevcuttur. İlk dikkat çeken konu, fitness skoru elde etmek için tek modelin kullanıldığı senaryolarda fitness skorununun 1.0'a yakın değerlere kadar tırmanması olmuştur. Ancak buna rağmen, elde edilen görsellerin insan algısı için hiçbir şey ifade etmediği gözlemlenmiştir. Kullanılan model programın çalışma sürecinde genel olarak uçurtma, magpie, bee eater veya quail tahminlerinde bulunmuştur.

Fitness skoru için üç modelin birlikte kullanıldığı senaryolarda ise skorların iyileşme göstermediği, daha doğrusu gösterilen iyileşme trendinin çok ama çok küçük çaplı olduğu gözlemlenmiştir. Ancak bu sefer elde edilmiş görsellerin skorlarıyla açıklanabildiği görülmüştür. Kullanılan üç model de programın çalışması boyunca aynı detection tahmininde bulunmuşlardır ve bunlar sırasıyla spot ışığı, satır ve dijital saat olduğu görülmüştür.

İki senaryoda da dikkat çeken detay, oluşturulan resimlerin başlangıç durumları ne olursa olsun, sonda aldıkları hallerin benzerlik göstermesidir. Öyle ki, siyah piksellerle başlatılan senaryolar, rastgele piksellerle başlatılan senaryolar, tek modelin kullanıldığı senaryolar ve üç modelin kullanıldığı senaryoların tümünde oluşturulan resimler gitgide siyah-beyaz gürültüyü andıran resimler haline gelmektedir. Bu son hallerine yakından bakıldığı zaman siyah piksellerin yanyana gelerek insan algısı için pek bir şey ifade etmeyen motifler oluşturduğu fark edilmiştir ancak bu motifler yine de ilk görüşte kolayca fark edilecek çapta gelişmişlik göstermemektedir.

Çalışma zamanı konusunda uygulama genel yapısıyla yüksek işlem gücü gereksinimine sahiptir. Bu doğrultuda çalışma süreleri her zaman uzun olmuştur. Ama bu süre farklılığının direkt olarak popülasyon büyülüğu, nesil sayısı ve (özellikle) kullanılan model sayısından etkilendiği gözlemlenmiştir. Kullanılan model sayısı üçe çıkarıldığında, (her ne kadar piksel sayısı azaltılıysa da) bir nesil için geçen süre hemen hemen 3 katına çıkmıştır.

5.2 Çıkarım

Tek modelin kullanıldığı senaryoda, kullanılan derin öğrenme ağının aslında verilen matris piksellerini paralel matematiksel işlemlerden geçirerek bir sonuç ortaya çıkardığı da göz önünde bulundurularak, algoritmanın anlamlı görsel üretmekten daha çok sinir ağının işlem yapısına uygun bir dizimle bahsedilen bu işlemler sürecinden maksimum skoru elde edebilecek görseller üretmeyi başardığı anlaşılmaktadır. Belki bu başka bir açıdan başarı sayılabilir, ancak üzerinde çalışılan problem tanımlamaları göz önünde bulundurulduğunda bu durum bir problem haline gelmiştir. Her ne kadar gözle anlaşılır resimler elde edilemediyse de, fitness fonksiyonu (yanlış bir şekilde) yüksek skorlar üretmektedir ve bu problem hedefine ulaşma süreci yaniltıcı olmaktadır.

Üç modelli senaryo için algoritma bu matematiksel işlemlere uygun değer ve dizimleri oluşturmada başarısız olmuştur. Bunun en büyük sebebi de, belki de benzer katman türleri kullansalar da, sinir ağlarının katman dizimleri, büyüklükleri, hiperparametreleri ve genel olarak işlem süreçleri tamamen farklıdır ve bu sebeple üçünü birden tatmin edecek bir sayı kümesi bulmak için genetik algoritmamız yetersiz kalmıştır. Bu sefer önceki paragrafta belirtilen yaniltıcı fitness fonksiyonu yaniltması durumundan bahsedilemez ancak şimdi de elimizdeki algoritma fitness skorunu yüksek bir degere yakinsayamamıştır.

Yapılan gözlemlerde yola çıkarak, algoritmanın tam olarak da rastgele pikseller yiğini ile sinir ağlarından doğru sonuç almaya çalıştığı söylenemez sonucuna varılmıştır. Her

ne kadar oluşan genel tablo insan algısı için anlamsız olsa da, yakından bakıldığı zaman oluşan beyaz zemin üzerine yan yana gelerek oluşan siyah çizgiler, bizlere algoritmamızın aslında feature extraction sürecinde başarılı olmak için çaba gösteriyor olduğu izlenmini bırakmaktadır. Dolaylı olarak, belki de skorun yüksek olduğu ancak görselin anlamsız kaldığı senaryoda bulunan özelliklerin gerçekten de görselde bulunduğu ancak bu özellikler görselin farklı noktalarına rastgele dağılmış olması sebebiyle insan algısı tarafından yakalanamıyor olduğu düşünülmüştür. Görsellerin gitgide beyaz zemin üzerine anlamsız siyah motifler haline gelme trendi de yine bu bahsedilen durumla bağlantılı olarak, feature extraction için en iyi sonucu verecek renk paletlerinin kullanılması olarak açıklanmıştır.

Bu düşünceleri destekleyecek somut bir yapı ortaya çalışma çalışması olarak ilk neslin siyah piksellerden oluşması ve algoritmanın bu temiz palet üzerine çizimlerini gerçekleştirmesi fikriyle bazı senaryolar dizayn edilmiştir ancak algoritmanın bu palet üzerine yaptığı değişiklikler yine insan algısı için anlamlı olmaktan uzak kalmıştır.

Gerek yapılan literatür taraması, gerek bizzat gözlemlenen deneyler ve bu deneylerin verdiği sonuçlar, başta yapılan problem sınırlamasındaki kusurlar hakkında düşünmeye sevk etmiştir. Yapılan denemeler açıkça göstermiştir ki, genetik algoritmalar bu tarz bir amaç için kullanılacaksa ya fitness fonksiyonu olarak kullanılacak öğeler insan algısına daha yakın bir mantıkla çalışıyor olmalı, ya da daha derin bir analizle fitness fonksiyonu için farklı faktörleri de değerlendirmeye alacak öğeler de eklenmelidir. Bunların yanı sıra belki de en iyi sonucu verecek yaklaşım problemin hedef veya başlangıç noktasını değiştirmek olacaktır. Daha çok geliştirilebilirlik kapsamına giren bu konular bir sonraki başlık altında detaylıca inceleneciktir.

5.3 Geliştirilebilirlik & İyileştirme Önerileri

Bu başlık altında projenin akışı esnasında elde edilen durumlardan yola çıkarak projenin başarı durumunu yükseltmesi muhtemel değişikliklerden bahsedilecektir. Bu iyileştirme fikirleri tamamıyla tahminlemeyle ortaya çıkmıştır ve üzerine denemeler yapılmadığı için kesin iyileştirme iddiası barındırmamaktadır, sadece beyin fırtınası niteliği taşımaktadır.

5.3.1 Başlangıç Noktası

Söz konusu özellikle üç modelin kullanılması ile yürütülen senaryolar olduğunda, başlangıç noktasının hedefe en uzak noktadan başlatılması, algoritma için gideceği doğrultuyu bulamaması durumunu yaratmaktadır. Eldeki görüntüler üç model için de

çok az anlam ifade etmektedir ama hepsi bu resmin (düşük ihtimalle de olsa) alakasız objelere benzediğini düşünmektedir. Bu sebeple eldeki görsel bir yönde ilerlemek yerine bu üç sınıflandırmada da daha iyi olmaya çalışıp sonuç olarak bir başarı elde edememektedir.

Bunun yerine aşağı yukarı her model için aynı anlamı verecek bir resmin kullanılması, sabit bir son hedef konmasa bile, algoritmaya ne doğrultuda ilerlemesi konusunda bir ipucu verebilir. Üç modelin de aynı tahminde bulunması, tahminlerin olasılığının düşük olduğu durumda bile, en azında yakınsanması gereken doğrultu hakkında fikir veriyor olacaktır.

5.3.2 Bitiş Noktası

Problemin sınırlamaları tamamiyle rastgele bir noktadan, tamamiyle rastgele bir noktaya ilerleme olarak belirlenmişti. Bu durum (önceki başlıkta da belirtildiği üzere) algoritma için yakınsama problemi oluşturmaktadır. Üç farklı ajan herhangi bir şey için yaptığı tahmin skorlarını artırmak yerine, belli bir sınıf için yapılmış olan tahminlerin skorlarını artırmaya odaklanmak, yine doğrultu belirleme sorunu için son derece kullanışlı olabilir. Örnek olarak yine rastgele piksellerden başlatılan program, üç modelin de "kedi" için yaptığı prediction skorunu artırmaya odaklanmış bir algoritmayla birlikte, belki de daha başarılı sonuçlar verebiliyor olacaktır. Bu durumda eldeki problemin hedefi daraltılmış olacaktır.

Bunun yanı sıra bu tarz sinir ağı modellerinin kullanılmadığı bir alternatif senaryoda, hedef olarak seçilen bir görselin benzerlik çıkarma yoluyla yine fitness fonksiyonu içinde kullanılması da başka bir yaklaşım olacaktır. Bu büyük ölçüde ilk kısımlarda bahsedilen Ahmet Gad'in projesine benzerlik gösterecektir.

5.3.3 Fitness Fonksiyonu

Fitness fonksiyonu olarak kullanılan derin öğrenme modelleri, tipki ilk bölümlerde de bahsedildiği üzere geniş bir veri setiyle eğitilmiş ve mimarisi standart haline gelmiş modellerdir. Fakat bu başlık altında bu modellerin yaptıkları işleri ne kadar iyi yaptıklarından çok, yaptıkları işi yapma biçimlerinin elimizdeki probleme ne kadar uygun olduğu sorgulanmaktadır.

Alternatif olarak, bu tarz modeller yerine tahminleme işlemini segmentation ile beraber yürüten modeller, bu problem için daha kullanışlı olabilir düşüncesi oluşmuştur. Şöyle ki, kullanılacak (örnek olarak) bir YOLO modeli ve bu modelin yaptığı segmentasyon alanını küçük tutmaya zorlayacak ikinci bir fitness fonksiyonu,

yapılacak tahmin ve tahminin bulunduğu alandaki rastgele motif dağılımını azaltabiliyor olacaktır. Bu öneriyile beraber, ileride yapılacak bu tarz çalışmalar için tek tür derin öğrenme ağlarına sadık kalınmaması gerektiği anlaşılmıştır.

5.3.4 Daha Güçlü Donanım, Daha Fazla Deneme

Eğer problemin niteliklerini korumak isteniyorsa, bu çalışma kapsamında eldekinin en iyisi kullanılmış olsa da, hala denenmemiş onlarca senaryo ve parametre kombinasyonunun olduğu unutulmamalıdır. Daha iyi bir donanım desteği ve daha uzun soluklu bir proje planlaması ile beraber bu çalışmanın hangi noktalara gelebileceği, farklı parametre kombinasyonlarının ardarda denenmesiyle daha iyi gözlemlenebilir ve elde edilebilecek sıra dışı sonuçlar algoritmanın aslında neye ihtiyacı olduğu konusunda ipucu verebilir.

Referanslar

- [1] C.-H. Huang and J.-L. Wu, “Watermark optimization technique based on genetic algorithms,” in *Security and Watermarking of Multimedia Contents II*, International Society for Optics and Photonics, vol. 3971, 2000, pp. 516–523.
- [2] D. Whitley, “A genetic algorithm tutorial,” *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994.
- [3] F. Comellas and J. Ozon, “Graph coloring algorithms for assignment problems in radio networks,” *Applications of Neural Networks to Telecommunications*, vol. 2, pp. 49–56, 1995.