Yapay Zeka

2021-2022 Bahar Dönemi

1. Ödev

Video Link

Ömer Aras Kaplan-17011039

Ömer Buğrahan Çalışkan-17011076

Algoritma:

Ödevimizi genetik algoritma kullanarak yaptık. Ödev içerisinde mutasyon, crossover, elitizm, selection, açılar için fitness, benzerlik için fitness kullandık. Hiperparametrelerimiz mutasyon katsayısı(mu), elitizm aday sayısı(elit_num), popülasyon büyüklüğü(Pop_Size), maksimum jenerasyon sayısı(Gen_Size) ve hedef matris oldu.

İlk jenerasyonumuzu tamamen random olarak oluşturduk. İki fitness fonksiyonunun normalize edilmiş halinin toplamı en yüksek olanları, o jenerasyonun en uygun bireyleri olarak kabul ettik. Bireylerin daha önce bulundukları bir yere gelmeleri durumunda farklı bir hareket üreten bir fonksiyon geliştirip, random kullanılan süreçlerde bu fonksiyona gönderdik.

En uygun bireylerden “elit_num” kadarını mutasyon fonksiyonuna gönderip, yeni jenerasyona ekledik. İkinci jenerasyonun kalanını elde etmek için ilk jenerasyondaki bireyleri önce selection fonksiyonuna sonra mutasyon fonksiyonuna gönderip, ikinci jenerasyona ekledik.

Önceki adımı son jenerasyona gelene kadar tekrar ettik. Her adımda; en uygun bireyin matriste çizdiği yerleri, benzerlik oranını, hedef matrisi ayrıca jenerasyonun ortalamasının ve jenerasyonun en iyi bireyinin benzerlik oranını ekrana bastırdık.

Son adımda şimdiye kadarki en uygun bireyin benzerlik oranı ve matrisini ekrana bastırdık.

Operatörler:

Birim:

Algoritmamızda birim olarak “lenS” uzunluğunda 0-8 arası sayılardan oluşan bir dizi kullandık. 0-8 arasında oluşmasının nedeni 8 tane hareket yönü olmasıdır. Bütün birimlerimiz hareket yönlerinden oluşuyor.

Crossover Function:

Algoritmamızda crossover yapmak için; random olarak seçilen bir indekse(x) ilk ebeveynin hareketlerini, kalanını da ikinci ebeveynin seçilen indeksten sonraki hareketlerini alarak yaptık.

Crossover sonrası oluşabilecek uygunsuz hareketleri düzeltmek için check_random_hareket() adlı fonksiyonu kullandık.

Mutasyon:

Algoritmamıza mutasyon entegre etmek için random bir sayı oluşturduk. Bu sayı mutasyon katsayısından büyükse(mu) 0,lenS arasında rastgele bir endekse rastgele bir sayı koyduk.

Selection:

Algoritmamızda selection olayını entegre etmek için rulet tekeri ve sıralama seçimini birleştiren bir algoritma yaptık. Büyükten küçüğe sıralanmış uygunluk dizisi içerisinde her birinin uygunluk değerini "Pop_Size – endeks" ile çarptık. Elde edilen diziyi toplamı 1 olacak şekilde normalize ettik. Bu dizi içerisinden her jenerasyonda Pop_Size kadar selection yapıp, crossover ve mutasyon uygulayarak yeni bireylerimizi elde ettik.

Elitizm:

Her jenerasyondan en uygun "elit_num" sayısı kadar bireyi mutasyon fonksiyonundan geçirip ikinci jenerasyona ekledik.

Açı Uygunluk Fonksiyonu:

Bireylerin yaptıkları her ardışık iki hareketi arasındaki açılar toplanır. Ardından bireylerin uzunluğuna ve 180'e bölünür. Çıkan sonuç ne kadar küçükse o kadar uygundur fakat benzerlik uygunluk fonksiyonu en büyük değer için daha uygun olduğundan dolayı, açı uygunluk fonksiyonunun son değeri 1'den çıkartılır.

Benzerlik Uygunluk Fonksiyonu:

Her jenerasyondaki bireyin yaptıkları hareketler kullanarak matriste üstünden geçtikleri yerler bulunur. Ortaya çıkan matris ile hedef matris karşılaştırılır.

Karşılaştırma yaparken 1'lerin ortak olması 0'ların ortak olmasından daha önemli olduğu için normalde kullanılan Simple Matching Coefficient yerine Jaccard ölçümü kullandık.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Jaccard = number of 11 matches / number of not-both-zero attributes values
= $(M_{11}) / (M_{01} + M_{10} + M_{11})$

Check_random_Hareket Fonksiyonu:

Crossover, mutasyon fonksiyonlarında ya da en baştaki jenerasyon için birey üretirken yaşanan süreçte random fonksiyonunun çıktısı bireyi matrisin dışına çıkartabilecek bir değer üretebileceği için bu süreçler sonrasında bu fonksiyonu çağırdık.

Fonksiyon bireyin hareketleri üzerinde teker teker ilerler. İlerlerken, eğer bir sonraki hareket matris dışında kalırsa bu hareketi matris dışına çıkarmayan bir değer ile değiştirir.

Aynı Yere Gitmemeyi Kontrol Eden Fonksiyon:

Bireyin yapacağı her hareket öncesi daha önce gittiği yerlerin matrisi bulunup, hareket sonrası konumun bu matriste işaretli olup olmadığına bakılır. Eğer işaretli ise yeni bir hareket üretilir.

Tablolar:

Popülasyon =1500 için:

Mutas-yon Kat-sayısı	Gen #10	Gen #50	Gen #100	Gen #150	Gen #200	Gen #250	Best
0.1	0.41	0.60	0.68	0.53	0.58	0.7	0.7
0.3	0.44	0.62	0.64	0.68	0.70	0.7	0.71
0.7	0.53	0.58	0.62	0.41	0.66	0.68	0.68

Mutasyon Katsayısı = 0.1 için:

Popülasyon	Gen #50	Gen #100	Gen #150
1500	0.29	0.68	0.52
2500	0.68	0.4	0.55
4000	0.19	0.5	0.74

Bulgular ve Yorumlar:

1. Genetik Algoritma genel olarak random bazlı olduğu için aynı hiperparametre ile çalıştırılan ve yüzde yüz benzerliğe ulaşmayan programlar aynı sonucu üretmez.
2. Popülasyonu yükseltmek sonucu kısmen iyileştirse de programın çalışma süresini üstel olarak artırmaktadır.
3. Mutasyonu artırmak, lokal aramalarda hatalara izin vermek durumlarında olduğu gibi algoritmanın sıkıştığı zamanlarda yardımcı olabilmektedir.
4. Mutasyonu artırmak, en iyi çözümün kaybolmasına daha çok olasılık tanımaktadır.
5. Elitizm için seçilen aday sayısını artırmak daha tutarlı sonuçlar yaratır fakat genetik çeşitliliği her jenerasyonda daha da azaltmaktadır.
6. Rulet tekeri kullanımı; jenerasyonlar ilerledikçe, genetik çeşitliliği artırmak için mutasyona bağımlı kalmaktadır.
7. Sıralama seçimi en uygun bireyin sonraki jenerasyondaki etkisini çok fazla azaltmaktadır.
8. Rulet tekeri ve sıralama seçimi algoritmalarını beraber kullanmak daha iyi sonuçlar elde etmeyi sağlayabilir.
9. 1 sayısının 0 sayısından fazla olduğu hedeflerde Jaccard kullanmak benzerlik bulmak için daha tutarlı bir yaklaşım sağlar.

10. Genel olarak genetik algoritmalar için bir eşik değer sağlamak hiperparametresi sağlamak mantıklı olabilir.

11. Hedef matris şekli ne kadar yuvarlak ve ne kadar kalın ise algoritmanın hedefe benzer bir sonuç çıkarabilme olasılığı artar.

12. Birimlerin uzunluğunu belirlerken hedefin kaç ayrık parçadan oluştuğunu bilmek büyük önem taşır.

13. Algoritmanın sürekli kendi üzerinde gezmemesi için daha önce geçtiği bir yere gitmemesini sağlayan bir fonksiyon mantıklı olabilir. Bu fonksiyon aynı zamanda performansı da çok düşürebilir.

14. Birim uzunluğu belirlenirken, hedefin başlangıç noktasına en yakın noktasının uzaklığı ve hedefte çizilmesi gereken yerler göz önünde bulundurulmalıdır.

15. Birden fazla uygunluk fonksiyonu olduğu durumda fonksiyon çıkış değerleri normalize edilmeli ve önemlerine göre katsayı sahibi olmalıdır.

16. Uygunluk fonksiyonlarından biri minimumda en iyiyi, biri maksimumda en iyiyi veriyorsa birini diğerinin türüne çevirmek gerekir.

17. Selection yaparken her bileşen için yeniden random sayı üretmek yerine, bileşenleri olasılıklarına göre diziye dağıtıp o diziden rastgele sayılar kullanarak bileşen elde etmek performansı önemli derecede artıracaktır.

18. Crossover sonrası iki çocuk yaratmak genetik çeşitliliği azaltacaktır fakat daha tutarlı sonuçlar verecektir.

19. Elitizm için seçilen bireylerin mutasyona uğramış kopyaları da sonraki nesle eklenebilir.

10 Örnek Resim İçin Sonuçlar:

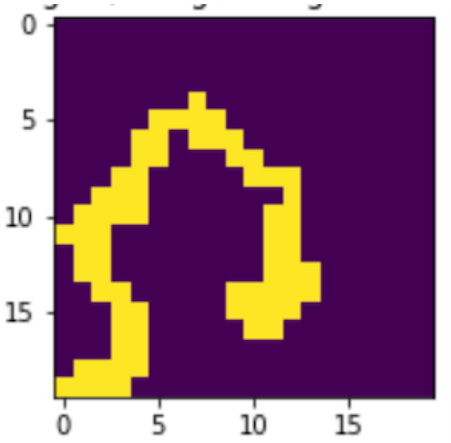
Örnek resimler için girilen hiperparametreler:

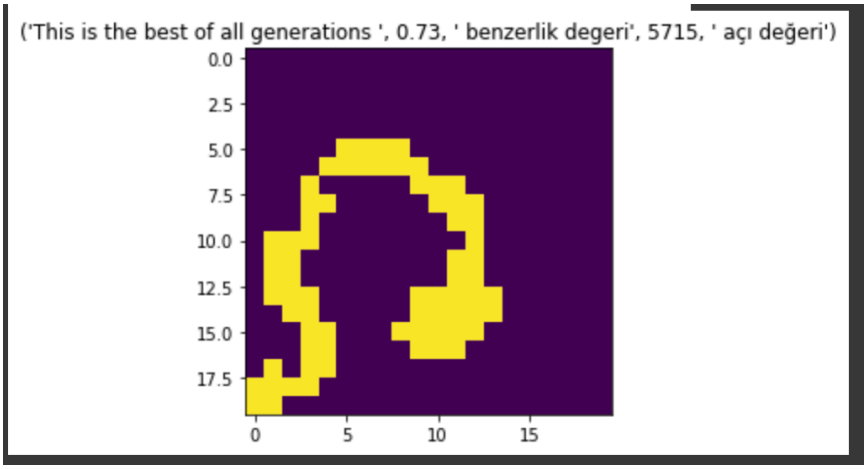
- Pop_Size = 1700 # popülasyon sayısı
- mu = 0.05   # her hareket için mutasyon olasılığı
- Gen_Size = 251   # jenerasyon sayısı
- elit_num = 1 # elitizm yapılacak birey sayısı

Soldaki resimler hedeflenen, sağdaki ise en iyi çizimi belirtmektedir.

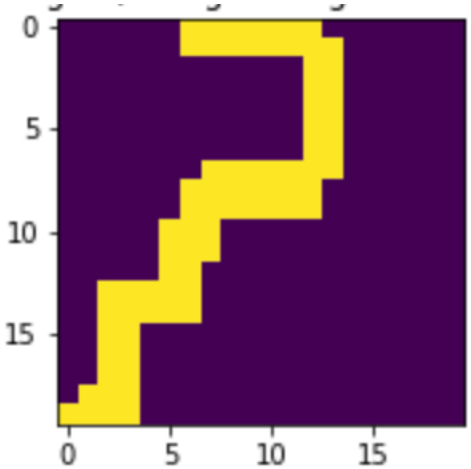
En iyi çizimde benzerlik değeri ve toplam açı değeri belirtilmiştir.

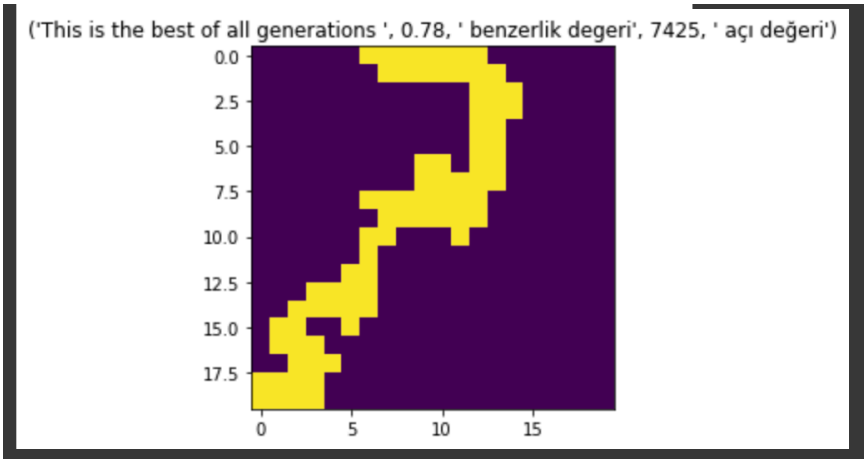
Resim1



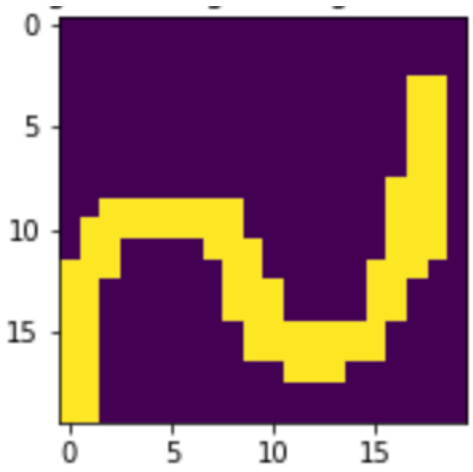


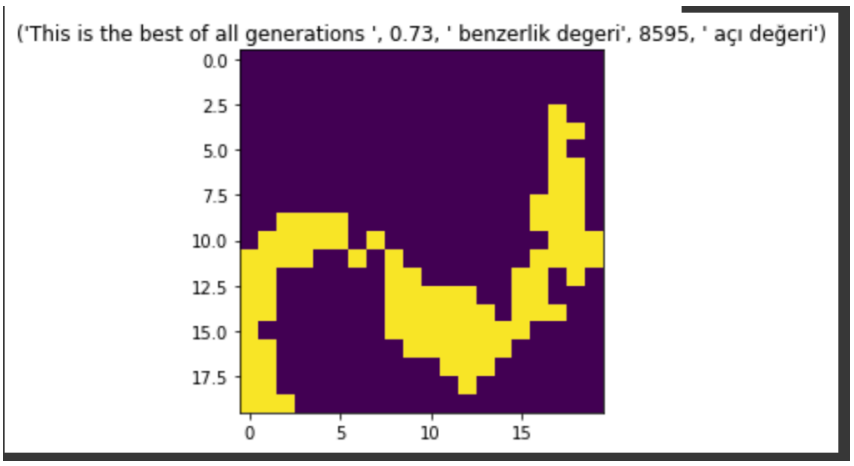
Resim2



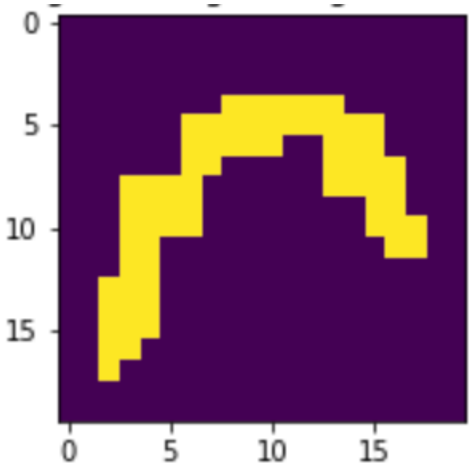


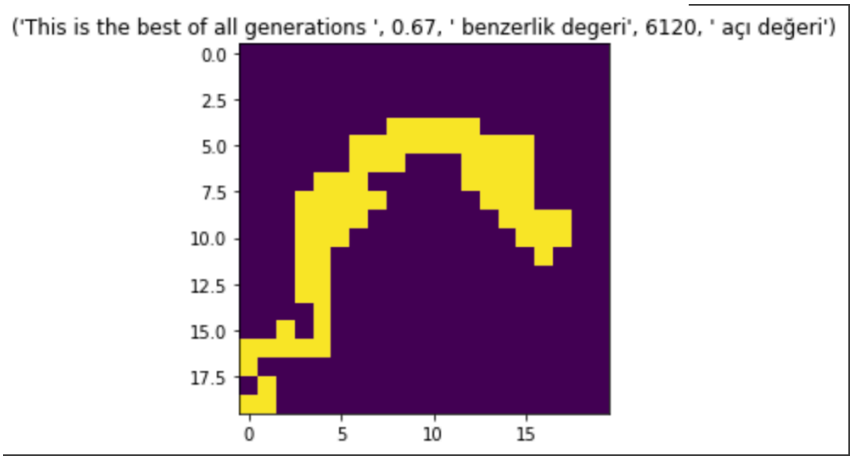
Resim3



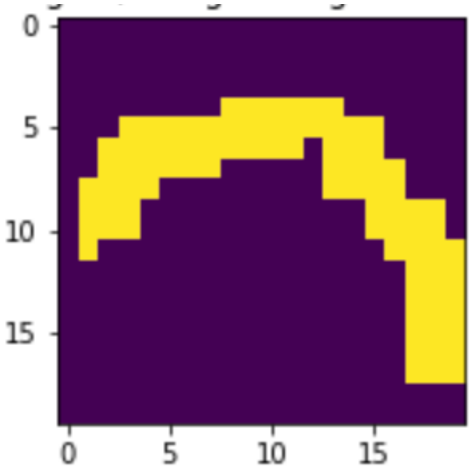


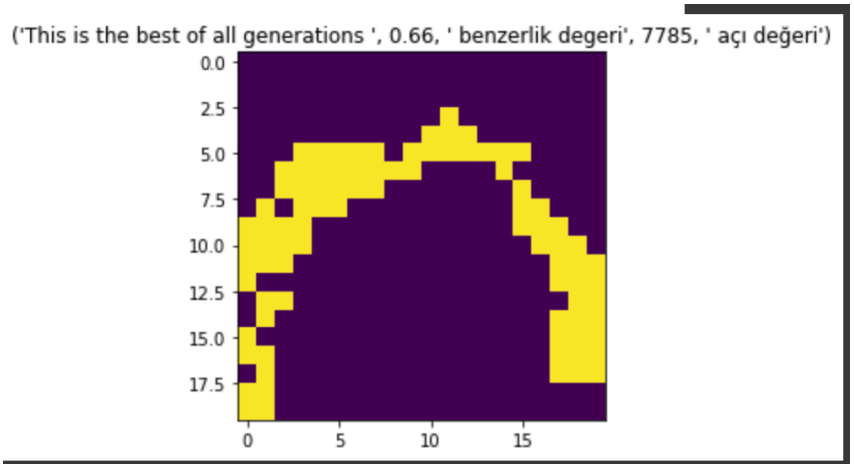
Resim4



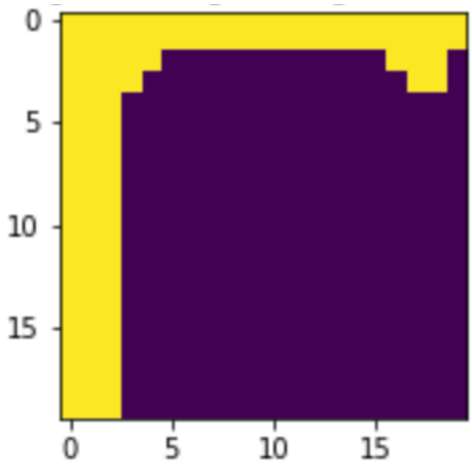


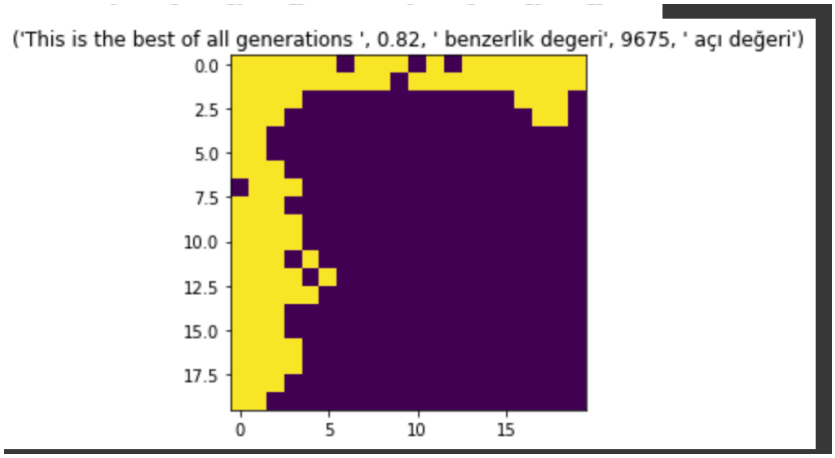
Resim5



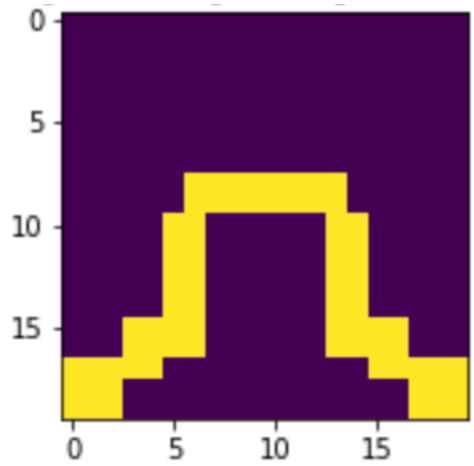


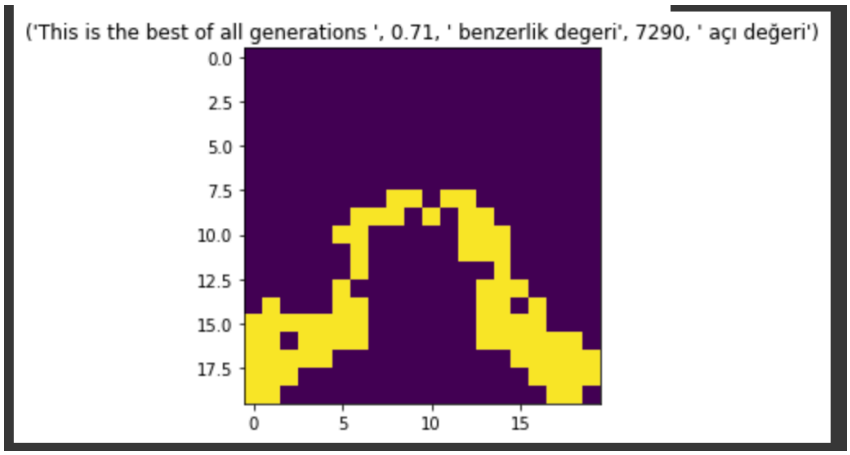
Resim6



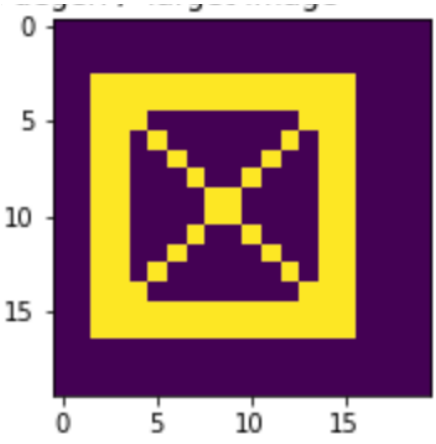


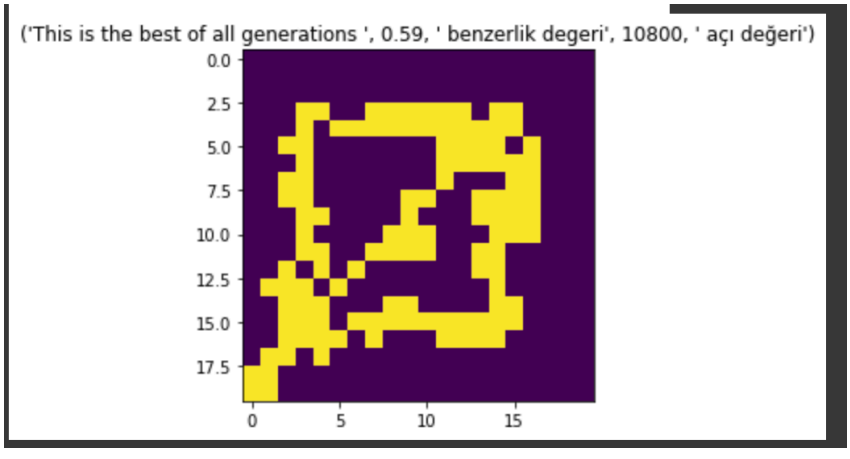
Resim7



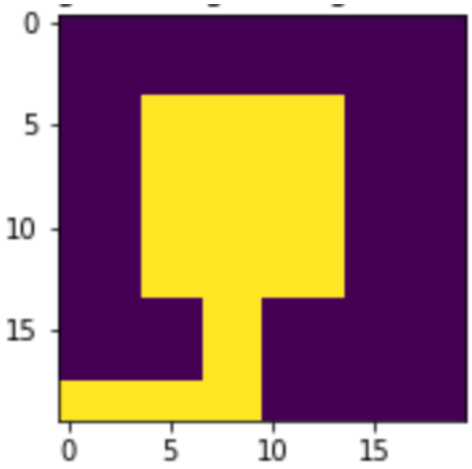


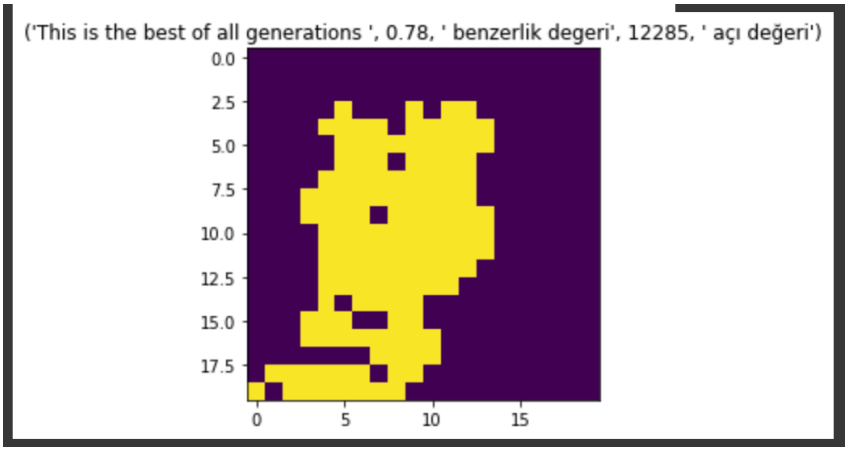
Resim8



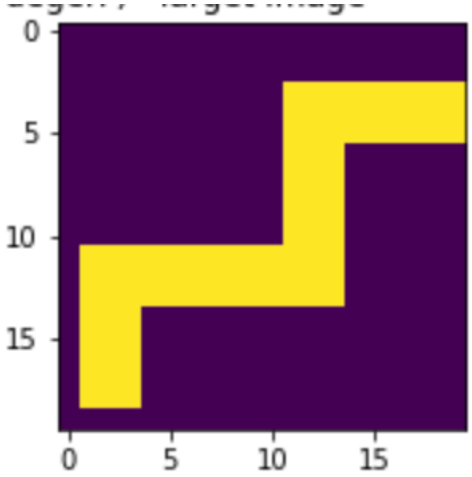


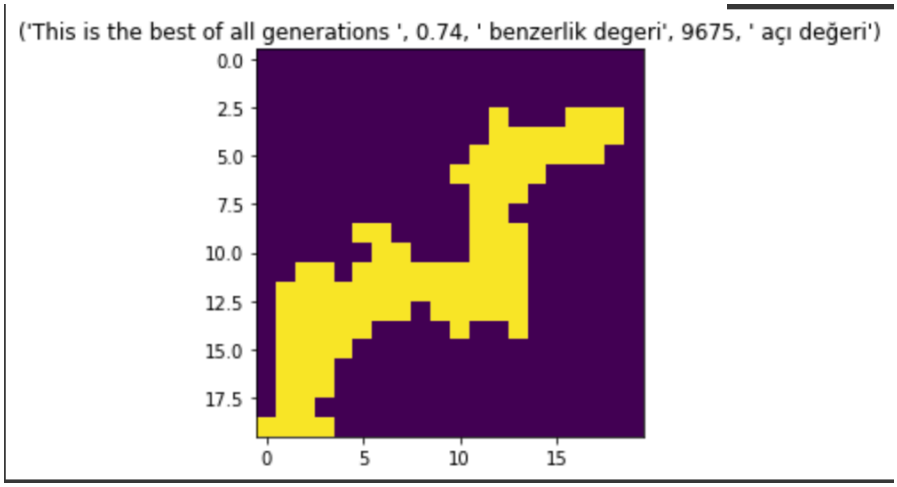
Resim9





Resim10





Bulguları ve yorumlarımızı yapmamıza yardımcı olan ek fotoğraflar zip dosyası içerisinde fotograflar klasörüne eklenmiştir.

Raporda bulunan 10 adet resimin matrisi Resimler.txt dosyasında bulunmaktadır.