YAPAY ZEKA DERSİ FİNAL PROJE

RAPORU

Bahattin Cihan Ünal – 18011603

* **Projenin Tanımı**

Yapay zeka dersi final projesi kapsamında genetik algoritmalar aracılığı ile rastgele piksel bloklarından, seçilmiş başka bir resme en yakın görüntüyü oluşturan bir algoritma geliştirilecek ve algoritmanın sonucunu ekranda gösteren bir program oluşturulacaktır. Algoritma görüntüyü oluşturmak için uzunluğu önceden belirlenmiş N\*N boyutundaki aynı renkte olan kare piksel bloklarını, resmin N\*N büyüklüğündeki kare bloklar ile tam olarak kaplanamadığı durumlarda ise resmin kaplanmamış kısımlarında uygun dikdörtgen piksel blokları kullanılacaktır.

* **Geliştirme Süreci**

Yukarıda tanımı verilen proje Java dili kullanılarak geliştirilmiştir. Swing ve ImageIO paketleri aracılığı ile bilgisayarın hafızasında bulunan resimler oluşturulan programda kullanılacak ve ekranda gösterilmiştir.

* **Çözüm Mantığı**

Oluşturulan algoritmada piksel bloklarının uzunluğu, birey sayısı, mutasyon oranı, iterasyon sayısı ve resmin bulunduğu adres kullanıcıdan alınmaktadır. Ardından benzetilmeye çalışılan resmin belirlenen uzunluktaki piksel bloklarına bölünmüş bir kopyası oluşturulmakta ve bu blokların orijinal resimde kapladığı piksellerin kırmızı, yeşil ve mavi değerlerinin ortalaması oluşturulan dizinin ilgili indisine kaydedilmektedir. Bu sayede uygunluk (fitness) fonksiyonu hesaplanırken gereksiz işlemlerden kaçınılmaktadır. Bu işlemden sonra belirtilen sayıda birey oluşturulmaktadır. Her birey için aynı renkte ve belirtilen uzunlukta kare piksellere bölünmüş, benzetilmek istenen resimde aynı boyutta olan rastgele üretilmiş RGB dizisi oluşturulmaktadır.

Bireylerin uygunluk değerini belirleyecek olan uygunluk fonksiyonunu hesaplamak için öncelikle gerçek resmin bir bloktaki ortalama kırmızı, yeşil ve mavi değerleri ile oluşturulan bireyin aynı bloğundaki kırmızı, yeşil ve mavi değerleri arasındaki farkın mutlak değerleri alınmaktadır. Bütün piksel blokları ve renk bileşenleri için bu fark hesaplanmakta ve bu farkların toplamları alınmaktadır. Elde edilen toplam renkler arasındaki fark ne kadar fazla ise uygunluk o kadar az olduğundan toplam -1 değeri ile çarpılmakta ve bütün bireylerin uygunluk değerinden hesaplanan en düşük uygunluk değerine sahip bireyin uygunluk değeri çıkartılmaktadır. Bu sayede en düşük uygunluk değerine sahip birey sonraki jenerasyonlarda kullanılmamakta ve diğer bireylerin uygunluk değerleri birbirlerine yaklaştırılmaktadır.

Bireyler oluşturulup uygunluk değerleri hesaplandıktan sonra uygunluk değerlerine göre büyükten küçüğe olacak şekilde sıralanmaktadırlar. Ardından rulet tekeri seçimi çaprazlanacak ile bireyler seçilmekte, rastgele olacak şekilde çaprazlama işlemi yapılmakta ve belirlenen mutasyon oranına göre yeni oluşturulan bireye mutasyon işlemleri belirtilen iterasyon sayısı kadar yapılmaktadır. Her iterasyonun sonunda oluşturulan bireyler uygunluk değerine göre sıralanmakta ve eski bireylerin yerini almaktadır. İlk iterasyondan itibaren orijinal resim ve oluşturulan en iyi resim ekranda görüntülenmekte, oluşturulan resim her iterasyonda oluşturulan en iyi bireyi gösterecek şekilde yenilenmektedir. Uygulama belirtilen iterasyon sayısı kadar çalıştıktan sonra uygulamanın çalıştığı süre boyunca oluşturulan en iyi resmi ve orijinal resmi göstermektedir.

* **Yorumlar**

Uygulama geliştirildikten sonra, uygulamanın büyük pikselli resimler ve küçük piksel blokları ile yapılan denemelerinde uygulamanın oldukça yavaş çalıştığı görülmüştür. Bunu önlemek adına gerekli işlemler ilgili piksel bloklarının ortalama RGB değerleri ile yapılmıştır. (1200 \* 950 piksel büyüklüğünde bir resim 5\*5 resim blokları ile 50000 iterasyonda yaklaşık 85 dakikada görüntü oluşturabilmiştir.) Süreci hızlandırmak adına uygulamanın her iterasyonu çoklu threadler ile çalıştırılmış, bireyleri sıralamak yerine jenerasyonun en iyi ve en kötü bireyleri seçilerek iterasyondaki işlemlere devam edilmiş (Java bünyesindeki Arrays.sort fonksiyonu en iyi ve en kötü elemanı tespit etmekten daha hızlı çalışıyor.) fakat çalışma süresi bakımından herhangi bir iyileşme elde edilememiştir. Uygulamanın daha hızlı çalışabilmesi için gerekli hesaplamaların GPU aracılığı ile yapılması sağlanabilir.

Bunun dışında en iyi sonuçları veren mutasyon oranının (blok\_uzunluğu / (resmin yatay uzunluğu \* resmin dikey uzunluğu))0.9 olduğu görülmüştür. Bu değere göre küçük boyutlu (300 \* 210) resimlerin her iterasyonda 2 veya 3 piksel bloğunun, daha büyük resimlerde ise her iterasyonda maksimum 11 piksel bloğunun mutasyona uğradığı görülmüştür. Çoğu durumda 50000 iterasyondan sonrasında ekrandaki piksel gürültülerinin fazla değişmediği gözlenmiştir.

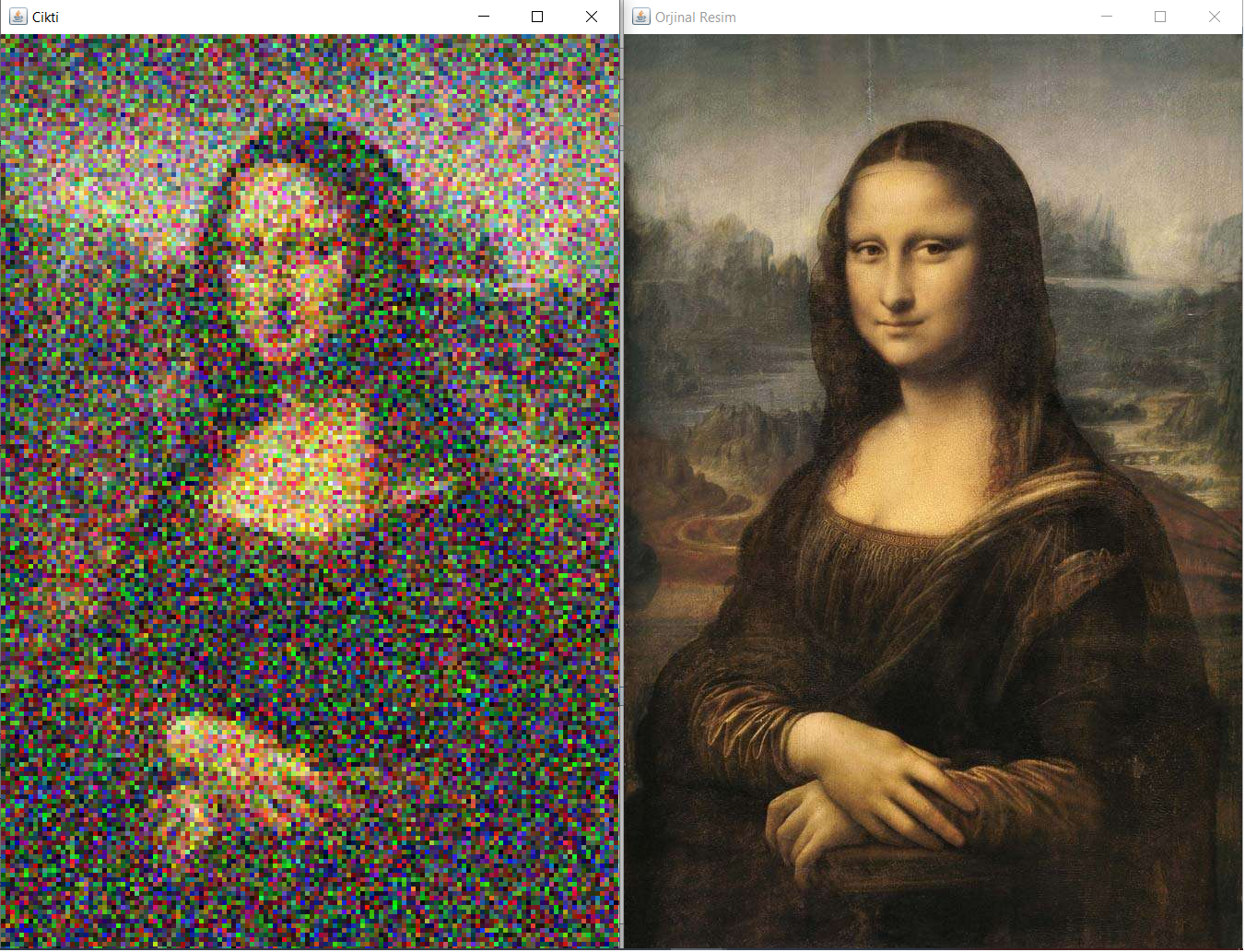
* **Başarı Oranı Tespiti**

1.0 – bireyin ilgili piksel bloğundaki RGB değeri ile resmin ilgili bloğundaki ortalama RGB değeri arasındaki uzaklık / bloklara ayrılmış hedef resmin RGB değerlerine olabilecek maksimum uzaklık

Yukarıda verilen denklem aracılığı ile oluşturulan görüntülerin, hedef resme olan yakınlığı 0 (Çok farklı) ila 1 (Resmin aynısı) arasında olacak şekilde ölçülmektedir. Örnek uygulama çalıştırma görüntülerinde oluşturulan her görüntünün başarı oranı görüntü ile birlikte verilmiştir.

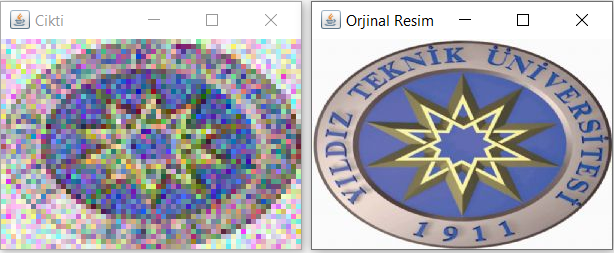
* **Uygulama Çalıştırma Örnekleri**
* 1. Örnek

671\*1000 boyutlu resim - 50000 iterasyon – 5\*5 büyüklüğünde piksel blokları – 30 birey sayısı – 2.42\*10-5 mutasyon oranı – 0.79 başarı oranı



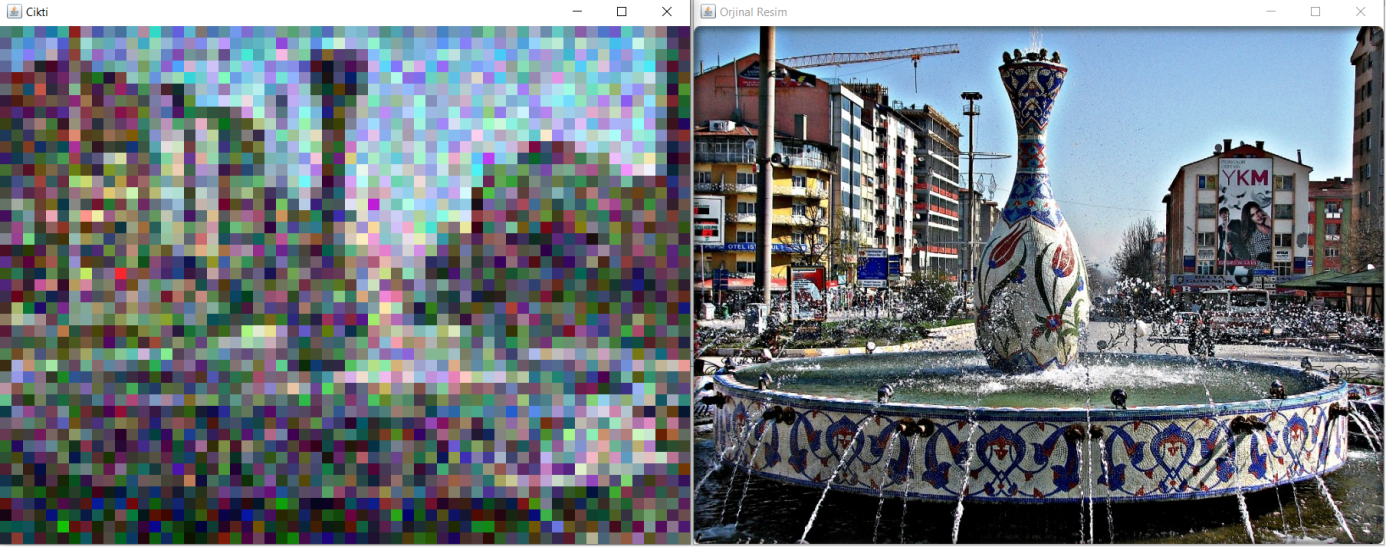
* 2. Örnek

300\*210 boyutlu resim - 50000 iterasyon – 5\*5 büyüklüğünde piksel blokları – 20 birey sayısı – 2.04\*10-4 mutasyon oranı – 0.86 başarı oranı



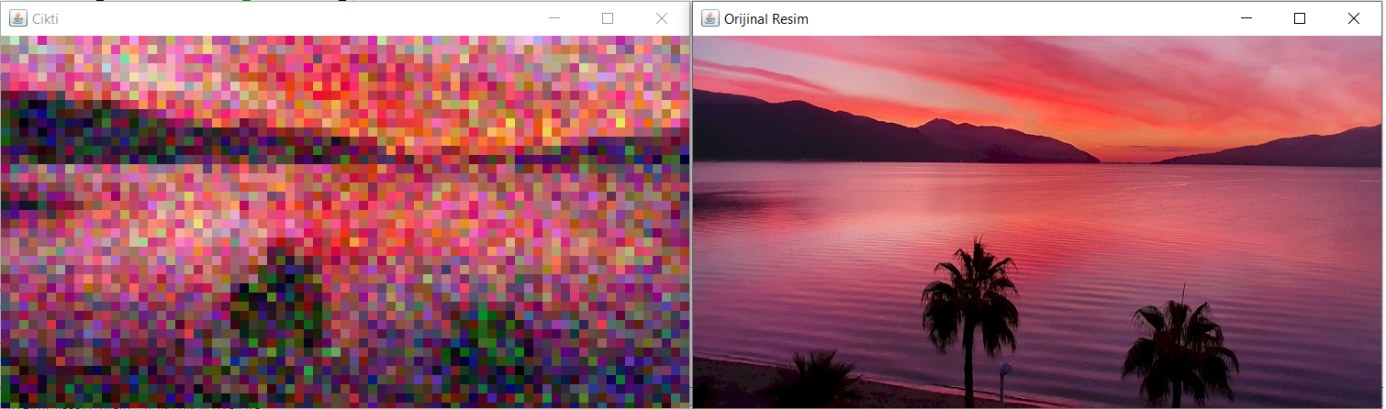
* 3. Örnek

900\*675 boyutlu resim - 40000 iterasyon – 15\*15 büyüklüğünde piksel blokları – 40 birey sayısı – 7.13\*10-5 mutasyon oranı – 0.87 başarı oranı



* 4. Örnek

750\*406 boyutlu resim - 60000 iterasyon – 10\*10 büyüklüğünde piksel blokları – 20 birey sayısı – 9.22\*10-5 mutasyon oranı – 0.86 başarı oranı



* 5. Örnek

520\*380 boyutlu resim - 70000 iterasyon – 4\*4 büyüklüğünde piksel blokları – 30 birey sayısı – 5.96\*10-5 mutasyon oranı – 0.84 başarı oranı

