OpenCV ve Mnist İle El Yazısı Rakam Tanıma

**Özet:** Bu projede , keras kütüphanesini kullanarak el yazısıyla yazılmış rakamı tahmin etmek için derin öğrenme modeli oluşturulur. Modeli eğitmek için MNIST El Yazısı veri seti kullanıldı, eğitilen modül kaydedildi böylece programı her çalıştırdığımızda modeli eğitmemiz gerekmez. Ardından OpenCV den yararlanılarak klasör dosyasına kaydedilen .png uzantılı fotoğrafı okur. Modelin sayıyı tahmin etmesi için gereken boyuta dönüştürülür. Görüntü modele beslenir ve öngörülen sayıyı döndürür.

**Giriş:**

İlk başta modeli oluşturulurken Keras kütüphanesinden yararlanıldı. Fotoğrafların istenilen boyutlara getirilmesi için Numpy kütüphanesinden yararlanıldı. Eğitim için CNN’in Sequential modeli seçildi.

Görüntü okuma kısmında ise OpenCV kütüphanesinden yararlanıldı. Fotoğraftaki rakamı tanımak içinde kontürleme işlemi yapıldı. Fotoğraf doğru sonuç vermesi için binary(siyah-beyaz) formuna getirildi.

**Kullanılan Yöntem ve Teknolojiler :** CNN’nin Sequential modeli kullanıldı. OpenCV kullanıldı.

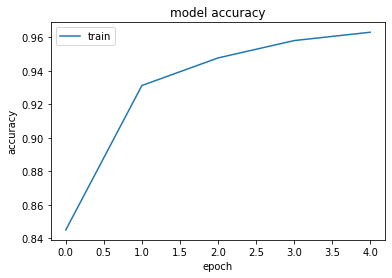
**CNN:** CNN genellikle görüntü işlemede kullanılan ve girdi olarak görselleri alan bir derin öğrenme algoritmasıdır. Bu algoritma farklı katmanlardan oluşmaktadır. Convolutional Layer, Pooling , Flatting ve Fully Connected’ dır.

* **Convolutional (evrişim katmanı)** CNN algoritmalarında görüntüyü ele alan ilk katmandır. Evrişim katmanında orijinal görsel boyutlarından daha küçük bir filtre görselin üzerinde gezer ve bu görsellerden belirli özellikleri yakalamaya çalışır.
* **Pooling Layer**, Ağırlık sayısını azaltır ve uygunluğu kontrol eder. Kapsadığı alandaki en büyük sayıyı alır. Bu sayede, sinir ağının doğru karar vermesi için için yeterli bilgiyi içeren daha küçük çıktıları kullanmış olur.
* **Flatting Layer** , Bu katmanın görevi basitçe, son ve en önemli katman olan Fully Connected Layer’ın girişindeki verileri hazırlamaktır. Genel olarak, sinir ağları, giriş verilerini tek boyutlu bir diziden alır. Bu sinir ağındaki veriler ise Convolutional ve Pooling katmanından gelen matrixlerin tek boyutlu diziye çevrilmiş halidir.
* **Fully-Connected Layer** , Bu katman ConvNet’in son ve en önemli katmanıdır. Verileri Flattening işleminden alır ve Sinir ağı yoluyla öğrenme işlemini gerçekleştirir.

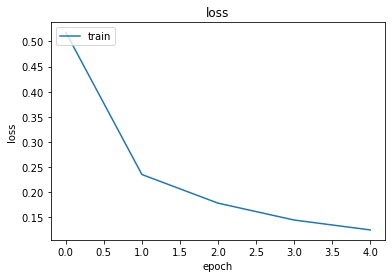
**Kullanılan Veriseti :** ProjedeKeras kütüphanesinde bulunan MNIST veri seti kullanıldı.

**Sonuçlar**

**ACCURACY – EPOCH İLİŞKİSİ**

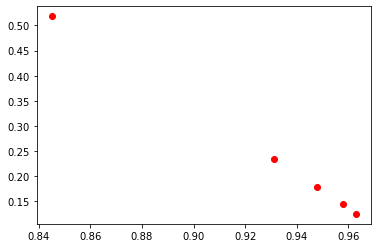


Grafikte görüldüğü üzere Epoch sayısı arttıkça modelin accuracy de artacaktır ama bununla birlikte öğrenme hızı da azalacak. Epoch sayısını istediğimiz kadar arttıralım model yüzde yüz öğrenemicek. Oluşturduğumuz modele göre accuracy değerlerinde değişkenlik görülebilir.

**LOSS – EPOCH İLİŞKİSİ**

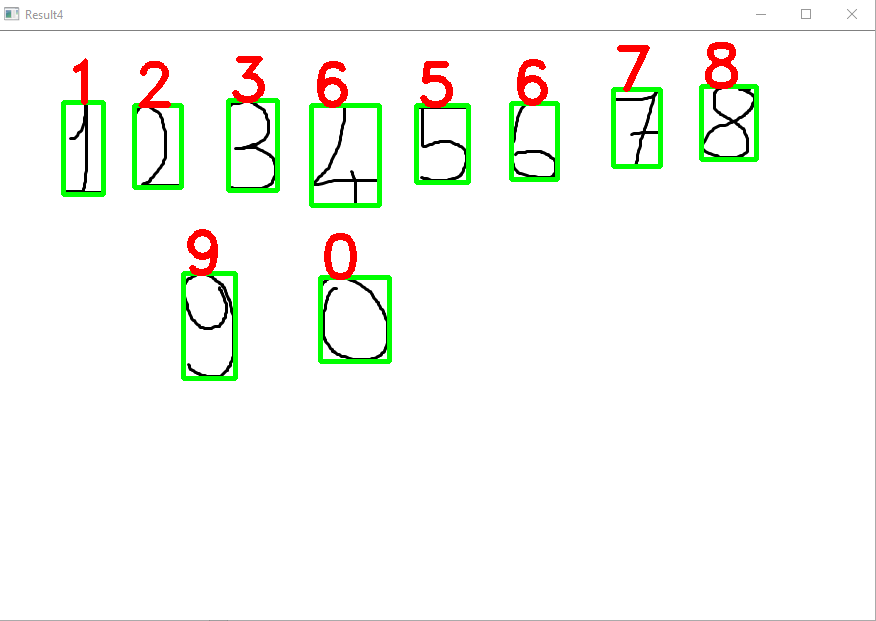
Grafikte görüldüğü gibi Epoch değeri arttıkça loss değeri de azalmaktadır. Epoch sayısı arttıkça modelin yaptığı tahminin , gerçek değerden nasıl değiştiğini grafikte görebiliyoruz. 1. Epoch da loss değerimiz 0.50 civarındayken 5. Epoch’ ta 0.15’ in altlarına kadar indiğini görebiliyoruz.

**ACCURACY-LOSS İLİŞKİSİ**



Grafikte görüldüğü üzere loss değeri azaldıkça accuracy artmaktadır. Bu da bu ikili arasında ters ilişki olduğunu gösterir. Yani modelimizin hata oranı ne kadar düşükse accuracy’si o kadar iyidir .

**Çıktı**

****

Programın Çıktısında da görüldüğü üzere çoğu 10 rakamdan 9 unu doğru buldu. Sadece 4 rakamını 6 olarak gördü. Bu da modelimizin iyi ama mükemmel olmadığını gösteriyor. Her güzelin bir kusuru vardır. 😊