SAYISAL VİDEO İŞLEME FİNAL ÖDEVİ

Problem Tanımı

Projenin amacı elle yazılmış olan harfleri, bilgisayar sistemi yardımıyla yardımıyla derin öğrenme yöntemini kullanarak tanımaktır. El yazısı harfleri A-Z, a-z, 0-9 olmak üzere İngiliz alfabesinde yer alan 62 adet büyük harf, küçük harf ve rakamlardan oluşmaktadır.

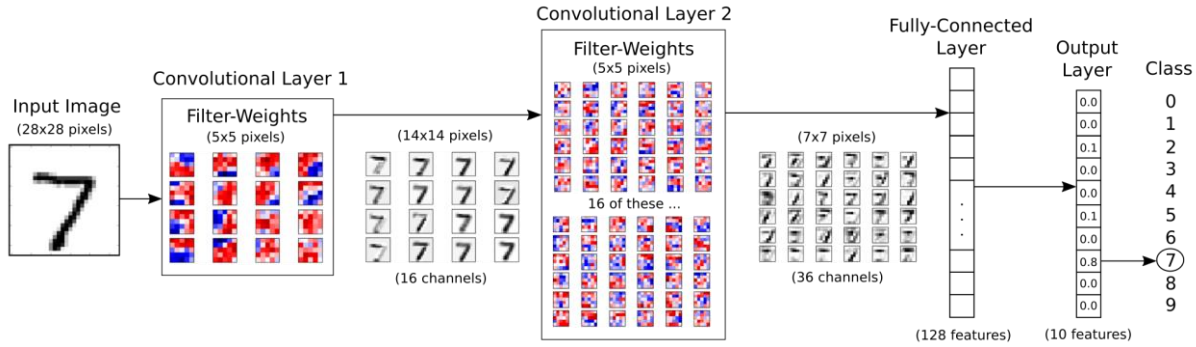
Yöntem

Projede harfleri tanıma işlemi derin öğrenme yöntemi yardımıyla yapılmıştır. Derin öğrenme, yapay sinir ağlarının, bir veya daha fazla gizli katman içeren daha karmaşık hali olarak düşünülebilir. Yapay sinir ağları ve diğer makine öğrenmesi yöntemleri yıllardır kullanılmalarına rağmen, iki büyük değişim makine öğrenmesine yeni bir bakış açısı getirmiştir: çok fazla miktarda eğitim verisi elde edilebilmesi ve bu büyük veriyi işleyebilecek GPU' ların işlemci olarak kullanılmaya başlanması. Önceki yıllara göre daha büyük miktarlarda veri üretilmesi makine öğrenmesinin başarısını arttırsa da veri işleme hızını oldukça yavaşlatmaktadır. Bu büyük veriyi işlemek için GPU' lar kullanıldığında ise CPU' dan 10-100 kat daha hızlı olduğu görülmüştür. Ayrıca hesaplamaların hızlanması daha karmaşık modellemeler oluşturulmasına imkân sağlamıştır.

Derin öğrenme algoritmasının uygulamasını kolaylaştırmak amacıyla birçok kütüphane oluşturulmuştur. Bu kütüphanelerden bazıları Caffe, Torch, Theano ve Tensorflow olarak örneklenebilir. Bu proje çalışmasında Google' ın desteğiyle oluşturulan açık kaynak kodlu Tensorflow kütüphanesi kullanılmıştır.

Derin öğrenmenin temel adımları Şekil 1'de gösterilmiştir.

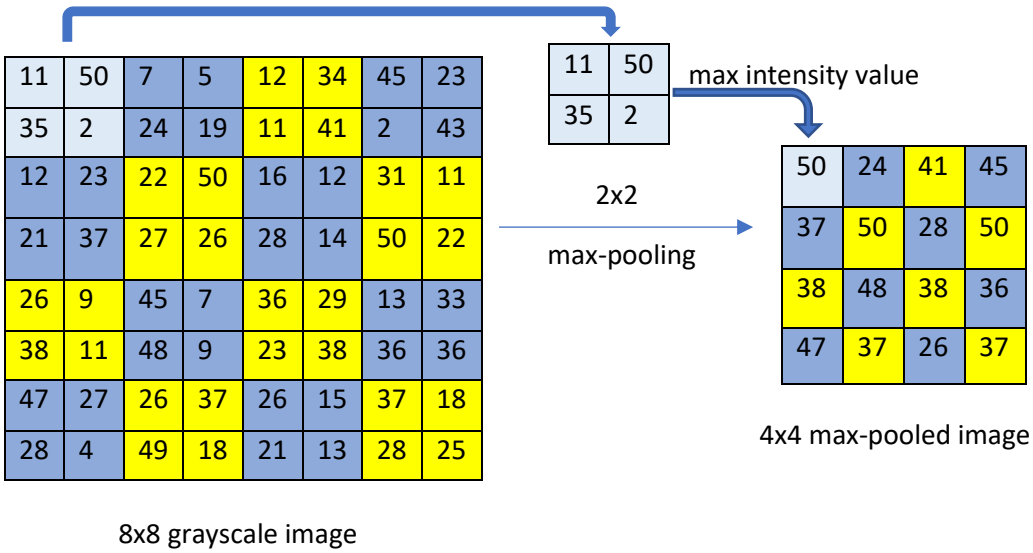
Şekil 1. Derin öğrenme



Girdi görüntüsü ilk konvolüsyonel katmanda farklı ağırlıklara sahip filtrelerden geçirilerek bias değerleri eklenir. Her bir filtre için bir çıktı görüntüsü elde edilir. Filtreler negatif (kırmızı) ve pozitif (mavi) ağırlıklardan oluşur. Bu filtrelerin her birinin görüntüdeki farklı bir özelliği öne çıkarmak için kullanıldığı düşünülebilir. Örneğin filtrenin biri girdinin yatay yönlü özelliklerini bulurken, bir diğeri girdideki köşe noktası özelliklerini saptayabilir. Çeşitli filtrelerden geçirilen görüntüler bir çeşit aktivasyon fonksiyonu olan ReLU (Rectified Linear Unit) yöntemiyle işlenir. ReLU fonksiyonu f(x) = max(0,x) şeklinde ifade edilir ve önceki adımda elde edilen değerlerin negatif olanlarını 0' a dönüştürür. ReLU fonksiyonu uygulanan çıktı görüntüler max-pooling yöntemiyle yeniden boyutlandırılmıştır. Bu sayede güçlü özellikleri kaybetmeksizin her bir girdi için yapılacak işlem yükü azaltılmıştır. Genelde pooling olarak adlandırılan örnekleme azaltma yönteminin bir türü olan max-

pooling, görüntünün 2x2 boyutlarında gruplanarak, her bir grubun kendi içindeki maksimum renk yoğunluğu değeriyle temsil edilmesidir. Max-pooling yöntemi Şekil 2' de örnekle gösterilmiştir.

Şekil 2. Max-pooling yöntemi

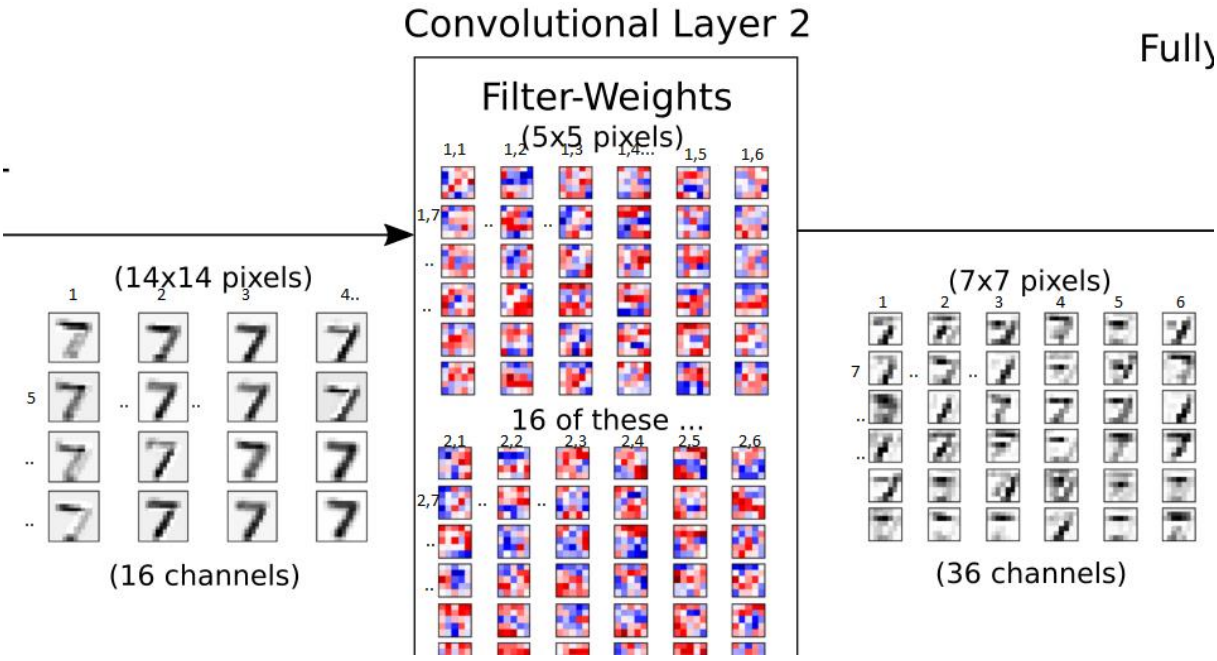


Şekil 2' de 8x8 boyutlarındaki girdi görüntüsü, 2x2 piksellik gruplara ayrılır. Her bir gruptaki maksimum renk değeri kaydedilir. Diğer üç renk değeri silinir. Her bir grup maksimum renk değeriyle temsil edilerek görüntü güçlü özelliklerini kaybetmeksizin 4 kat küçültülmüş olur.

Max-pooling işleminin çıktısı olan 16 görüntü kanalı ikinci konvolüsyonel katmana girdi olarak verilmiştir. İkinci katmanda uygulanan konvolüsyon işlemi Eşitlik 1' de ve Şekil 3'te gösterilmiştir.

$$\text{CL2OutputChannel(i)} = \sum_{i=1}^{36} \sum_{k=1}^{16} Channel(k) * Filter(i,k) \qquad (1)$$

Şekil 3. İkinci Konvolüsyonel Katman



İkinci katmanın çıktılarına da birinci katmanda olduğu gibi ReLU ve max-pooling uygulanarak çıktı elde edilmiştir. 36 tane çıktı kanalı birleştirilerek 7x7x36 boyutlarında görüntünün özelliklerini temsil eden vektör oluşturulur ve tam bağlantılı katmana girdi olarak verilir. Tam bağlantılı katman girdileri alır ve sahip olduğu nöron sayısı kadar çıktı verir. Şekil 1'de gösterilen çıktı katmanı da 10 nörona sahip olan tam bağlantılı bir katmandır. Bu katmandaki nöron sayısı sınıf sayısını belirler. Tam bağlantılı katman özellik çıkarmada kullanılan filtre ağırlıklarını değiştirerek (backpropagation) optimum ağırlıkları hesaplar. İkinci tam bağlantılı katman da girdinin hangi sınıfa ait olabileceğini olasılıksal olarak belirler ve etiketini oluşturur.

Deneysel Sonuçlar

Yapılan projede 62 sınıfa ait 692.932 tane eğitim görüntüsü ve 116.323 tane test görüntüsü bulunan NIST veri seti kullanılmıştır. NIST veri setinde bulunan görüntüler renkli ve 128x128 boyutlarındadır. El yazısı verisinde renk özelliği önem taşımadığı için görüntüler griye dönüştürülmüştür. Karakterlerin özelliklerini çıkarılmasında detaylar önemsiz olduğu için görüntü dosyaları 28x28 boyutuna çevrilmiştir. Daha sonra 28x28 boyutundaki gri görüntülerden, .idx dosya formatında ve ikili seviyede eğitim ve test dosyaları ile etiket dosyaları oluşturulmuştur.

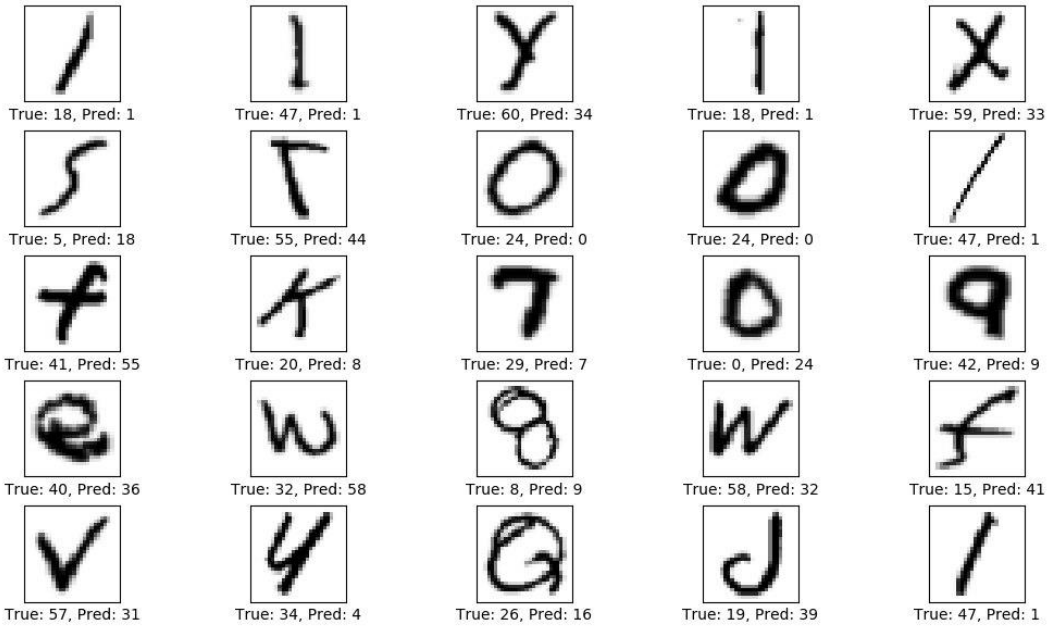
Projenin eğitim verisi üzerinde yapılan geri besleme ile öğrenmenin optimizasyonu farklı sayıda iterasyonlarla gerçekleştirildi. İterasyon sayısı arttıkça test verileri üzerindeki başarı arttı. Ancak yüksek miktarda veri ile çalışıldığı için bu durum zaman maliyetini de oldukça arttırdı. İterasyon sayısı, test seti üzerindeki başarı oranı ve program çalışma zamanı ilişkisi Şekil 4'te gösterilmiştir.

Şekil 4. CNN geri besleme iterasyon sayısı, başarı oranı ve zaman maliyeti ilişkisi

İterasyon →	**1**	**100**	**1000**	**5000**	**10000**	**50000**
Zaman (dk)	0.3	0.5	5.3	29	61	460
Başarı (%)	0.4	38	74	82	84	86

Gerçekleştirilen projede hatalı bulunan test verileri Şekil 5'te örneklenmiştir.

Şekil 5. Doğru sınıflandırılmayan test verisi örnekleri

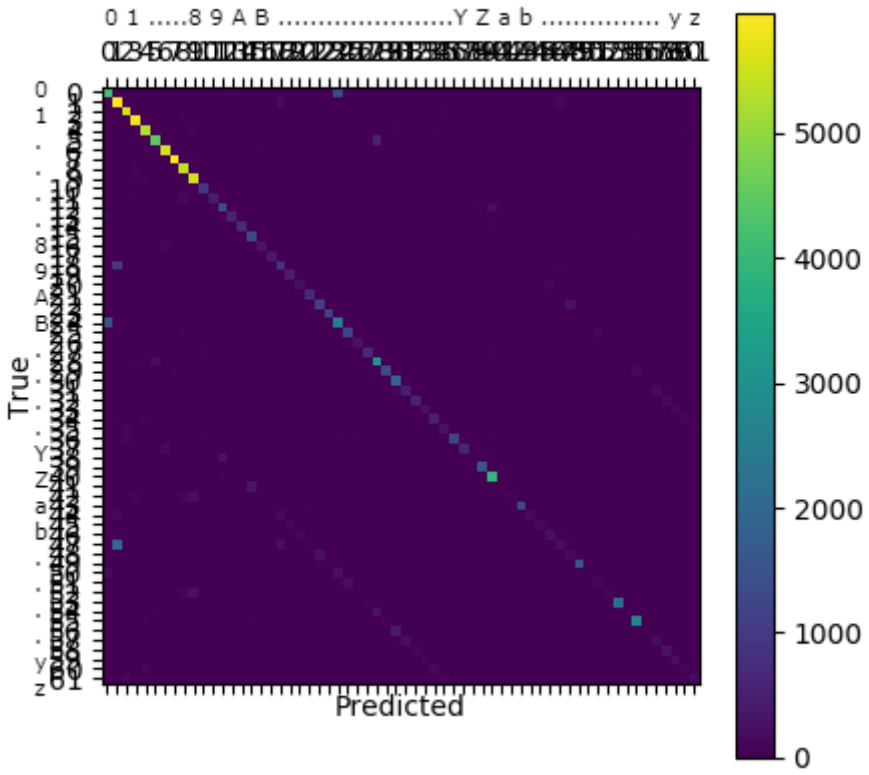


Görüntüler incelendiğinde karakterlerin 0 – 61 etiketli gerçek sınıf numaraları ile tahmin edilen sınıf etiketleri yer almaktadır. Sınıf etiketlerinin ASCII karşılıkları “emnist-byclass-mapping.txt” dosyasında verilmiştir. İlk satır incelendiğinde 1 olarak tahmin edilen karakterlerin gerçek karşılığı “I” ya da “l” karakterleridir. Bu üç karakter birbirine oldukça benzedikleri için insan gözü ile ayırt etmek bile mümkün değildir. Benzer şekilde “0” ve “O” karakterlerinin sistem tarafından ayırt edilmesi oldukça zordur. Bunun yanında bazı harflerin büyük ve küçük yazılışlarının aynı olması sınıflandırıcının yanlış karar vermesine neden olmaktadır. “W”, “X”, ”Y”, “V”,”J” gibi harflerde sınıflandırmanın başarısız olması makuldür. Ayrıca bazı harflerin yazımın doğru yapılmaması başarı oranını düşürmüştür. 7. Ve 13. Ve 15. Görüntülerde hangi harf olduğu gözle bile ayırt edilemeyen harfler kullanılmıştır. Tüm bu sınıflandırıcı performansından kaynaklanmayan hataların yanında 6. Görüntünün “I”, 12. Görüntünün “8”, 18. Görüntünün 9 olarak tahmin edilmesi modellemeden kaynaklı açıklanamayan hatalardır.

Sonuç

Sınıflandırmada yapılana hataların çoğunluğu birbirine benzeyen farklı karakterler ve benzer özellikteki büyük-küçük harf çiftlerinden kaynaklanmaktadır. Yapılan çalışmaya ait hata matrisi Şekil 6’te görülebilir.

Şekil 6. 5000 iterasyon hata matrisi



Referanslar

https://www.tensorflow.org/get_started/mnist/beginners

https://wiki.tum.de/display/lfdv/Layers+of+a+Convolutional+Neural+Network

https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/02_Convolutional_Neural_Network.ipynb