Derin Öğrenme Algoritmalarının MNIST Veri Seti Üzerinde Karşılaştırılmaları

Abdulaziz ÖZ
Düzce Üniversitesi Bilgisayar Mühendisliği

ÖZET

Yapılan olan bu çalışmanın amacı yinelemeli sinir ağları, uzun kısa süreyi bellek, Çok Katmanlı Algılayıcı, Kalıntı sinir ağı ve evrişimsel sinir ağlarını karşılaştırmasıdır. Karşılaştırma yaparken Mnist veri setini üzerinden yapılmıştır. Araştırmada ilk başta kullanılmış olan algoritmalar, daha sonra ise kullanılmış olan optimizasyon metrikleri anlatılmıştır. Deney sonucu algoritmaların birbirleriyle olan doğruluk ve kayıp fonksiyonları karşılaştırılmıştır. Hangi algoritmanın daha iyi sonuç verdiğine ulaşılmaya çalışılmıştır. İşlemler sonucu çıkan sonuçların, daha önceki buna benzer yapılmış olan araştırmalar ile sonuçları karşılaştırılmıştır.

1. GİRİŞ

Son zamanlarda, yapay zekâ üzerine yapılmış olan araştırmalarda zor olan bilgisayar problemlerinin çözümlenmesinde hız kazandıran derin öğrenme adıyla bir alan ortaya çıkmıştır. Günümüzde kamera sayılarının artmasından dolayı dijital veride çok büyük miktarda bir birikim olduğu söylenebilir. Nesnelerin görüntülerinin bilgisayar tarafından anlamlı hale getirilmesi büyük kolaylıklar sağlayacaktır. Derin öğrenme çok fazla sayılarda veri girişi yapıldığında birbirleri arasındaki farklı özellikleri kendi öğrenmektedir. Veri setimizde ne kadar fazla veri girilirse öğrenme işlemi o kadar başarılı olur. Her ardışık katman, önceki katmandan çıkan değeri girdi olarak almaktadır. Veriler birden fazla katmandan geçmektedirler. Üst katmanlar daha ayrıntılı olan katmanlardır.

Derin öğrenme yöntemleriyle çok yüksek sayıda doğruluk ve çok düşük sayıda kayıp değeri bulunmaktadır. Hatta bazen insanların performanslarını aşmaktadırlar. Bu yöntemler farklı katman sayılarında modeller oluşturularak eğitilmektedir.

Bu yapılmış olan çalışma; Çok Katmanlı algılayıcı(MLP), Yinelemeli Sinir Ağları(RNN), Uzun Kısa Süreli Bellek (LSTM), Kalıntı sinir ağı(Res-Net) ve Evrişimsel Sinir Ağları (CNN) Derin Öğrenme yöntemlerinin el yazısı karakterlerini tanıma ve performanslarının karşılaştırılmasıdır. Öncelikle her bir yöntemin en iyi sonuçları vermesi için parametrelerin en doğru şekilde seçilmesi adına birçok deney yapılmıştır. Modellerin parametreleri ayarlandıktan sonra algoritmaların el yazısı ile yazılmış olan rakamların resimlerinden oluşan Mnist veri seti üzerinde karşılaştırılmıştır.

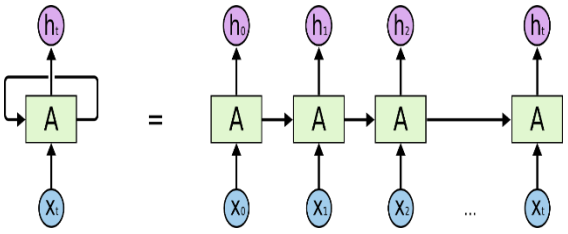
2. DERİN ÖĞRENME MİMARİLERİ

Bu tezde en çok kullanılmakta olan beş adet derin öğrenme algoritması kullanılmıştır. Yinelemeli Sinir Ağları(RNN), Çok Katmanlı algılayıcı(MLP), Uzun Kısa Süreli Bellek(LSTM), Kalıntı sinir ağı(Res-Net), Evrişimsel Sinir Ağı (CNN) algoritmaları. Bu bölümde bu algoritmaların detayları hakkında bilgiler verilmiştir.

2.1 Yinelemeli Sinir Ağı(RNN)

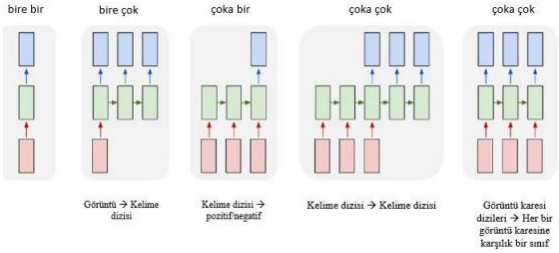
RNN'ler diğer sinir ağlarından farklı olarak sıralı bilgi kullanır. Hesaplanmış önceki verileri belleğinde tutarak bilgi akışının sürekliliğini sağlayan bir yapıdadır. Özellikle dil işlemede kullanılan bu modelin temel çalışma mantığı çıktıyı önceden öğrenilmiş verilere göre üretmektir. Sıradan sinir ağlarına göre daha anlamlı gözüken bu modelin dezavantajları da vardır. Kurulan sıralı bir cümledeki kelimeleri hafızasında tutarken başka bir cümle içerisine anlamsal çıkarımda bulunup önceden öğrendiği kelimeyi yerleştirmesi gerekirken yerleştirememektedir. Örneğin "ben İtalya'da yaşıyorum." bilgisinden sonra gelen birçok cümleden dolayı "akıcı bir şekilde … konuşurum" cümlesinde ki boşluğa İtalyancayı eklemeyi mümkün kılmamaktadır. Çünkü cümle akışı içerisinde farklı diller yer almış olabilir ve model hangisinin boşluk için uygun olacağına karar veremez.

RNN, metin, el yazısı, hava durumu, borsa gibi verilerin tahmininde kullanılabilmektedir. Verilerin modele verilmesinden sonra çıkan sonucu tekrar girişte kullanılmaktadır. Aşağıdaki şekilde görüldüğü üzere sinir ağında oluşan döngüyle hafıza korunmaktadır. Böylece bir sonraki çıktının sonucunu tahmin etmek için önceki giriş değeri veya önceki çıkış değeri çok işe yaramaktadır.



Şekil 1-RNN Giriş Döngü Şeması

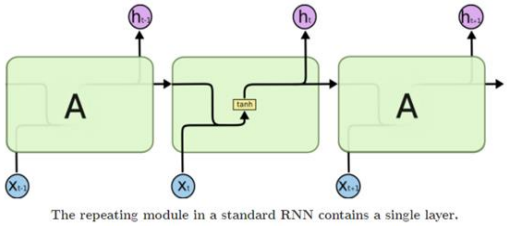
Sıradan RNN yapısında her girdiye karşılık bir çıktı vardır ve birebir şeklinde adlandırılır. Bire çok, çoğa bir ve çoğa çok şeklinde probleme göre değişkenlik gösteren kurgular mevcuttur. Bire çok için resim altyazısı tahmini bir örnek olabilir. Resim, mimarinin bir girdisidir ve kelime dizisini bir çıktı olarak verir. Çoğa bir örnek için duygu sınıflandırma problemi bir örnektir ve kelimeler dizisi çoklu bir girişi temsil etmektedir. Sınıf olaraksa tek bir çıktısı olur. Çoğa çok kurgu şekline örnek olarak da makine çevirisi için kelime dizisine karşılık gelen yine farklı bir kelime dizisi çıktısı elde edilmektedir. Video her bir resim karesine göre video sınıflandırma probleminde de resim kareleri dizisine karşılık her bir kare için bir sınıf tahmini üretilmektedir.



Şekil 2- RNN Çoklu Yapı

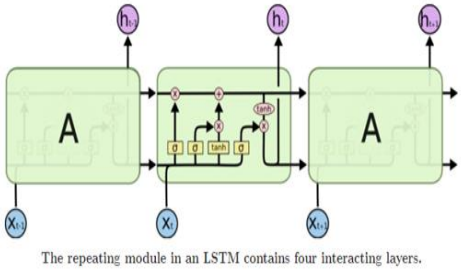
2.2 Uzun Kısa Süreli Bellek(LSTM)

LSTM uzun süreli bağımlılıkları öğrenebilen RNN'nin bir türüdür. Hochreiter ve Schmidhuber tarafından 1997 yılında tanıtıldı. Çok çeşitli problemler üzerinde çok iyi bir şekilde çalışmaktadırlar ve şu anda yaygın olarak da kullanılmaktadır.



Şekil 3-Tek katmanlı standart RNN

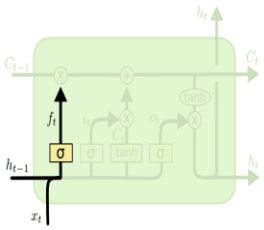
LSTM'ler ardı ardına birbirini takip eden yapıya sahiptir. Lakin bir sonraki parçanın yapısı daha farklıdır. Etkileşimli olan dört parçalı sinir ağı katmanına sahiptir.



Şekil 4-Dört Etkileşim katmanı olan LSTM

- Birinci adım

İlk adımda, ağ gereksiz olan bilginin tespitini yapar ve hücreden atma işlemini belirler. Şekil 5'te hücreden atılma işlemi kararını vermek için unutma geçidi katmanı olarak bilinen sigmoid fonksiyonu katmanı gösterilmektedir.

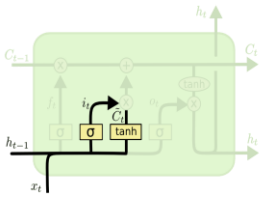


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Şekil 5-Unutma geçidi katmanı

- İkinci adım

İkinci adımda, ağda hücre durumuna depolanması gereken bilginin kararı veriliyor. Bütün bu süreç takip adımlarından taviz vermektedir. Girdilerin hangisinin güncellenmesi gerektiğini giriş kapısının katmanı isimli sigmoid katmanı belirler. Bundan sonra, duruma eklenecek olan yeni adayların vektörünü hazırlayan tanh katmanı aşağıdaki gibi kullanılmaktadır.

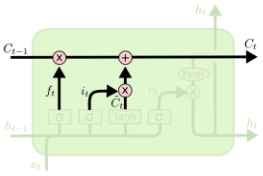


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Şekil 6-Giriş geçidi ve yeni aday değer vektörü

- Üçüncü adım

Üçüncü adımda, C_t değeri eski hücrenin durumu olan C_{t-1} değerinden güncellenir. İlk başta eski durum için unutulması için kararlaştırılan bilgileri unuttuktan sonra f_t ile çarpılır. Bundan sonra, çıkan sonuçların güncellenmesi kararlaştırıldıktan sonra yeni adayların değeri olan $i_t \; x \; C_t$ eklenir.

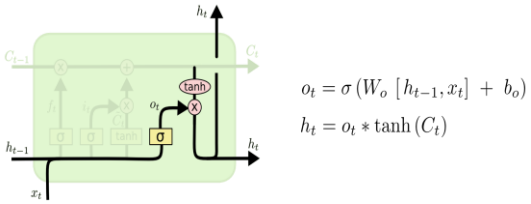


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Şekil 7-Yeni hücre durum

- Dördüncü adım

Sonuncu adım için, hangi bölümün çıktı olduğuna karar verdikten sonra sigmoid katmanı çalıştırılır. Daha sonra, ihtiyacımız olan parçalara çıktı olarak ulaşmak için hücre durumu tanh'ten geçirilip sigmoid kapısındaki çıkan değer ile çarpılır.



Şekil 8-Çıkış geçidi ve yeni bilgi

2.3 Evrişimsel Sinir Ağı(CNN)

Bir CNN, bir giriş görüntüsünü aldıktan sonra, görüntüde olan birbirinden çeşitli nesneyi birbirinden ayırabilen bir derin öğrenme algoritmasıdır.

CNN, temel olarak nesne tanımlanması, görüntünün sınıflandırılması ve benzerlikleri kümelemek için kullanılmakta olan derin yapay sinir ağlarıdır.

Evrişimsel(convolution) Sinir Ağları Nasıl Çalışır?

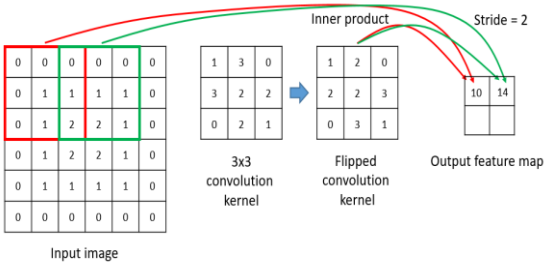
Bir bilgisayar bir resim gördüğü zaman o görüntüyü girdi olması için almaktadır. Burada bir piksel değerleri dizisi görülür. Görüntünün çözünürlüğü ve boyutuyla orantılı olacak şekilde, 32 x 32 x 3 sayı dizisini görmelidir. Sadece noktaya sürmek için, PNG formatında renkli bir görüntünün olduğunu ve boyutunun 480 x 480 olduğunu düşünelim. Temsil eden dizi 480 x 480 x 3 olur. Her bir sayıya pikselin yoğunluğunu açıklamak için 0 ile 255 arasından değerler verilir. Bu sayılar, görüntü sınıflandırmada bilgisayar için olan tek girişlerdir. Buradaki mantık veriler sayı dizelerini, görüntünün bir sınıf olma olasılığını açıklamasına yarayan sayılar çıkarmasıdır.

Çıktılar ve girdiler bilindiğine göre, buna nasıl yaklaşmamız gerektiğini düşünmeliyiz. Bilgisayardan istenilen şey nesnelerin birbirlerinden farklı olan, o nesneye özgü olan özelliklerini bulup onları çıkarabilmesidir. Bir kuşa baktığımız zaman onu kanadı ve 2 ayağı olması sayesinde kuşu olduğunu anlayıp sınıflandırabiliriz. Bilgisayar ise kenarlar ve eğriler gibi düşük seviyeli olan özellikleri arayarak ve daha sonra bir dizi konvilasyon katmanı yoluyla daha soyut kavramlara dayanarak görüntüyü sınıflandırabilmektedir. CNN'in genel yaptığı işlem bu şekildedir.

2.3.1 KATMANLAR

İki Boyutlu Evrişim

İki boyutlu bilgiye uygulanması gereken filtre için x ve y ekseni simetriği alınır. Bütün değerler matriste eleman, eleman olarak çarpıldıktan sonra, bu değerler toplanarak çıkış matrisiyle alakalı olan eleman olmasıyla kaydedilir. Bu çapraz ilgileşim ilişkisi şeklinde de isimlendirilir. Tek kanallı giriş verisi kullanılırken işlem kolayca yapılmaktadır. Fakat giriş verisinin formatı ve kanal sayısı farklı olabilmektedir.



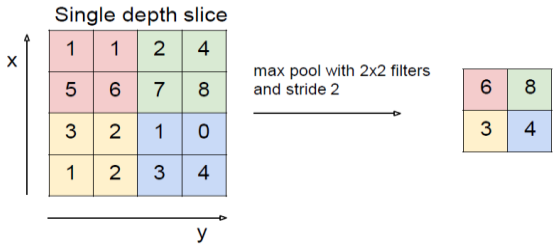
Şekil 9-İki Boyutlu Evrişim(convolution)

Renkli görüntüler kırmızı-yeşil-mavi olmak üzere 3 kanaldan oluşmaktadır. Bu yüzden 3 kanal içinde evrişim(convolution) işlemi yapılmaktadır. Uygulanmakta olan filtre kanalı sayısı ile çıkış işaretinin kanal sayısı eşit bir şekilde hesaplanmaktadır. Bu hesaplama işlemini sinir ağındaki bir katman olarak

düşünürsek, girişteki görüntü ve filtre aslında devamlı geri yayılımla güncellenmekte olan bir ağırlık matrisi. Aktivasyon fonksiyonu uygulanmakta olan çıkış matrisi için en son skaler bir bias değeri eklenmektedir.

Havuzlama (Pooling)

Bu katmanda çoğunlukla maksimum havuzlama(pooling) yöntemi kullanılmaktadır. Bu katmanda daha hiçbir parametre öğrenilmemiştir. Giriş matrisindeki kanalın sayısı sabit tutularak yüksekliğin ve genişliğin bilgisi azaltılır. Hesaplamalarda olan karmaşıklığı en aza indirebilmek için kullanılır. Bazı önemli verilerin kaybolmasından sorumlu olduğundan başarım oranını düşmektedir.

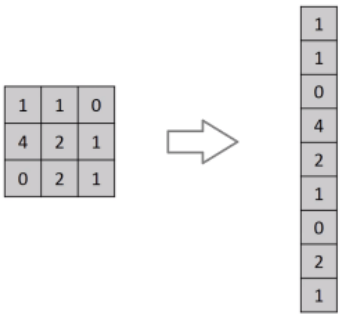


Şekil 10 - 2x2

Lokasyon bilgisinin önemli olmadığı sorunlarda bile çok iyi sonuçlar ortaya çıkmaktadır. Çıkışa en büyük havuzlama boyutu içindeki pikseller aktarılır. Yukarıdaki örnekte 2x2 max – havuzlama(pooling) işlemi 2 piksel kaydırılarak yapılmıştır. 4 değer içindeki en yüksek değer çıkış düğümüne yönlendirilir. Çıkışta 4'te 1 boyutunda bir veri elde edilir.

Flattening Layer

Bu katman en son ve en önemli katman olan Fully Connected katmanındaki giriş verilerini hazırlamaktadır. Genellikle sinir ağlarının giriş verileri tek boyutlu bir diziden gelmektedir. Bu veriler ise konvilasyon ve havuzlama(pooling) katmanından gelmiş olan matrislerin tek boyutlu diziye çevrilmiş halidir.

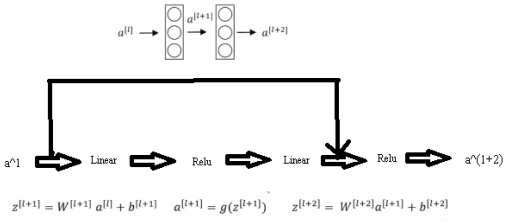


Şekil 11 - Flattening Layer

2.4 Kalıntı Sinir Ağı(ResNet)

Resnet önceki modellere göre daha farklı bir mantıkta çalışmaktadır. Fazlalık olan değerlerin bir sonraki katmanı beslemekte olan bloğun modele eklenmesi şeklinde olmaktadır. Bu özellik ile diğer klasik modellerden farklı bir yapı ortaya koymaktadır.

Linear ve ReLU arasına 2 katmanda bir eklenir. Şekil 12'deki gibi sistemdeki hesap değişmektedir. Önceden gelmiş olan $a^{[1]}$ değeri, $a^{[1+2]}$hesabına eklenmiş olur.

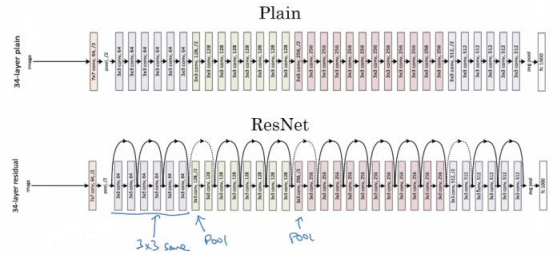


Şekil 12-ResNet Model Oluşumu

Genellikle, Modelin katman sayısının artmasıyla başarının artması düşünülmektedir. Fakat yapılan deneyler sonucu böyle olmadığına ulaşılmıştır. Bunun sonucunda ResNet modelini oluşturmuşlardır. Böylelikle

$w^{[l+2]}=0$ olduğunda yeni teoriye göre $a^{[l+2]}=b^{[l+2]}$ olmaktadır. Bu durum ise istenmemektedir.

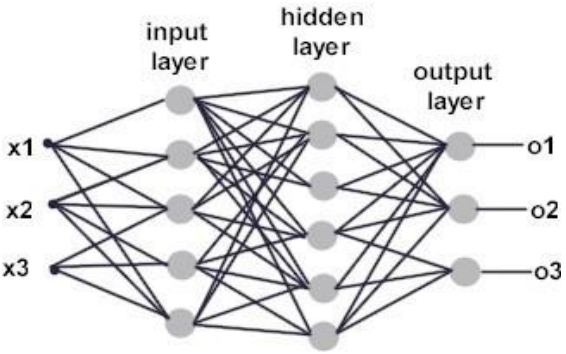
Fakat artık değerin beslemesi iki katman önce gelen $a^{[l]}$ değeri o an ki ağırlığının 0 olması bile yeni çıkışın öğrenim sonucu olan hatasını optimize etmektedir. Böylece eğitilirken daha hızlıdır.



Şekil 13-ResNet Modeli

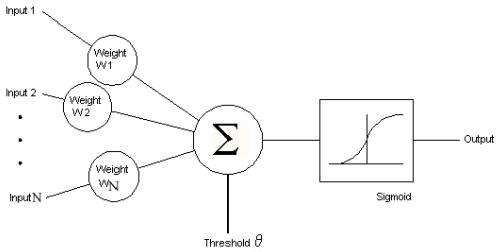
2.5 Çok Katmanlı Algılayıcı(MLP)

MLP, Yapay Sinir Ağlarında en çok kullanılan modellerden biridir. MLP'ler XOR sorunlarını çözmek için ortaya çıkmıştır. MLP çoğunlukla sınıflama yaparken daha verimli çalışmaktadır. Çok Katmanlı Ağların yapısı aşağıdaki gibidir.



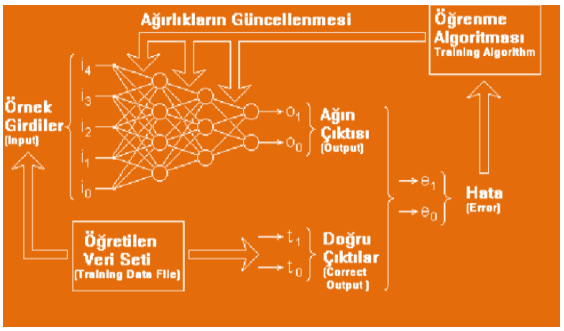
Şekil 14 -MLP modeli

Girişlerin çok olduğu durumlarda nöronlar ihtiyacımızı karşılayamayabilir. Paralel işlemlerde nöronların yetersiz kalacağından dolayı katman kavramı ortaya çıkmaktadır. Mlp'lerde tek katmanlı modelden farklı olan şey arasında gizli katman bulunmasıdır. İhtiyacımıza uygun ayarlanan katman sayılarına göre, giriş katmanından gelen veriler bir sonraki katmana iletilir. Eğer tek ara katmanlı ise direkt çıkış katmanına iletilir. Birden fazla ara katman varsa hepsine tek tek iletilir. Burada her katmanın girişi bir öncekinin çıkışı olmaktadır. Her bir katmandaki nöron, bir sonraki katmanlardaki her bir nörona bağlıdır. Nöron sayıları ise probleme göre belirlenmektedir. Ağın çıkışı ise önceki katmanlardan gelen veriler ile belirlenir. Çıkış katmanındaki elemanların sayısıyla sistemdeki çıkış sayısı eşittir.



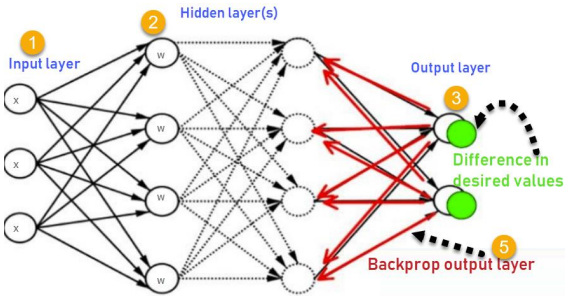
Şekil 15 -Sigmoid Fonksiyonu

Mlp hücreleri yukarıdaki şekilde gösterildiği gibidir. Yukarıdaki modelde Aktivasyon fonksiyonu olarak sigmoid fonksiyonu seçilmiştir. Mlp de Delta Öğrenme Kuralı baz alınarak öğrenme işlemi yapılır. Giriş ve çıkışların belli olduğu bir eğitim seti ile ağın öğrenme işlemi yapılır. Geri yayılımlı sinir ağlarında öğrenme işlemi, giriş verilerinin çıkış değerleriyle eşleşmesinden oluşan fonksiyonu bulmaya çalışması işlemidir. MLP sisteminin öğrenme yöntemi genel olarak iki seviyeden oluşmaktadır. Geri yayılımlı işlemde önce ileriye doğru hesaplanır. Sonra geriye doğru hesaplanır.



Şekil 16 -Genelleştirilmiş Delta Öğrenme Kuralı'nın yapısı

İleri doğru hesaplama**,** işleminde yukarıda anlattığımız gibi önce katmana giriş verileri verilir ve bu verilen giriş değerleri bir sonraki katmandan, en son katmana kadar ilerleyerek çıkışa ulaşır. Her ara katmanda bir önceki katmandan gelen girdiler toplanarak net bir girdi sayısı bulunur. Hesaplanan girdiyi aktivasyon fonksiyonundan geçirerek çıktı bulunur ve bu her katman ara katman için yapılır ve en son çıktı bulunur.



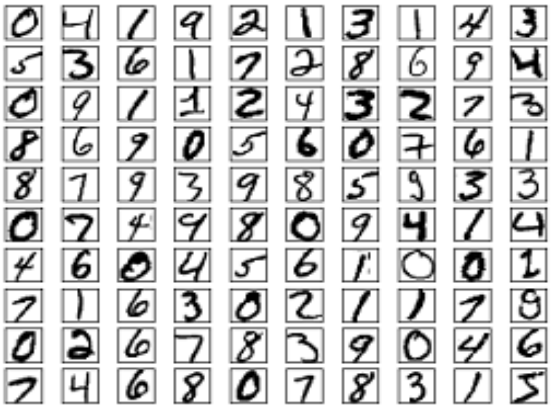
Şekil 17 - Back Propagation

İlk aşama ağdan çıktı bulunarak bitirilir. Sonraki aşamada hatanın ağırlıklara dağıtılma işlemi gerçekleştirilir. Tahmin edilen çıktıyla elde edilen çıktı farklıysa bir hatanın olduğu düşünülür. Geri yayılımlı hesaplama işleminde hatanın her ağırlıklara dağıtılmasıyla her iterasyon için azalması beklenmektedir. Başlangıçta rastgele verilen ağırlıkların, geri yayılırken önceki ağırlıklara hataların dağıtılması ile yeni değerler bulunmuş olur.

3. DENEYSEL ANALİZ

3.1 Kullanılan Veri Seti

MNIST veri tabanı açılımı "Modified National Institute of Standards and Technology Database" veya Türkçe olarak "Ulusal Standartlar ve Teknoloji Enstitüsü Veri Tabanı" şeklinde isimlendirilir. El yazısı ile yazılan rakamları ile oluşmaktadır. Bu veri tabanı; derin öğrenme yapay zeka ve makine öğrenmesi yöntemlerini kullanmak için yaygın olarak kullanılır. Çoğu görüntü işleme yönteminin eğitilmesinde çok yaygındır. İçinde el yazısı rakamlarının bulunduğu, Amerikan sayım bürosundan alınan 60000 eğitim ve Amerikan Lisesi öğrencilerinden alınan 10000 test görüntülü bir veri tabanıdır.



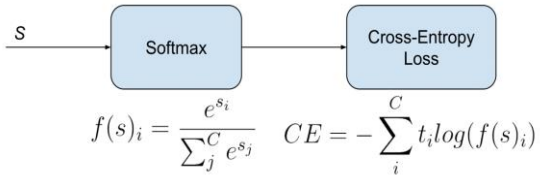
Şekil 18 -Mnist El Yazısı

3.2 Performans Metrikleri

Oluşturulan modellerde performans metriği olarak doğruluk ve kayıp değerleri hesaplanmaya çalışılmaktadır.

İlk olarak kaybolan değeri bulabilmek için modellerin derlenmesi gerekir. Kayıp değer, tahmin edilen ile beklenen gerçek çıkış arasındaki farkın nicel olarak hesaplanmasıdır. Bize, çıkışın tahmin edilen model tarafından yapılmış olan hata ölçümünü vermektedir. Yani

kayıp değeri, modelin test edilirken ne kadar düzgün çalıştığını hesaplar. Kaybın düşük olarak hesaplandığı değerler, modelin iyi çalıştığını gösterir. Bu çalışmada, kayıp fonksiyonu olarak kategorik çapraz entropi kullanılmıştır. Softmax Kaybı olarak da bilinir. Bir Softmax aktivasyonu artı bir Çapraz Entropi kaybıdır. Çok sınıflı sınıflandırma için kullanılır.



Şekil 19-CrossEntropy

Uygulamada Sgd, Adam ve Rmsprop ile optimizasyon yapılmıştır. Veri setinin eğitilme süresini belirleyip "fit" yöntemiyle epok sayısına göre eğitilmektedir. model.evualete metoduyla, sonuçlar incelenir. Doğruluk hesaplaması için aşağıdaki formül uygulanmaktadır.

$$Doğrulama = \frac{\text{Doğru öngörülen sınıfı}}{\text{Toplam test sınıfı}} x\ 100$$

Doğruluk Formülü

Olasılıksal Dereceli Azalma (SGD)

SGD, metin sınıflandırırken ve doğal dil işleme işlemi yapılırken fazlaca karşılaşılan seyreltilen ve büyük ölçekli olan makine öğrenmesi problemleri için başarılı bir şekilde uygulanmaktadır. Verilerin seyrek olduğu göz önüne alındığında, bu modüldeki sınıflandırıcılar, 10^5'den fazla eğitim örneği ve $10^{5'}den$ fazla özellik içeren problemlere kolayca ölçeklenebilir. Verimlilik ve uygulama kolaylığı açısından avantajlıdır.

$$\theta = \theta - \alpha \nabla_\theta J(\theta; x^{(i)}, y^{(i)})$$

Güncelleme İfadesi

Ortalama Karekök Yayılımı (RMSprop)

RMSProp, Adagrad yönteminin azalmakta olan öğrenim oranını, eğimlerin karesinin hareketli ortalamalarından çözüme ulaşmaya çalışır. Eğimi normalleştirmek için geçmiş eğimlerin inişlerinin büyüklüğünden faydalanmaktadır. RMSProp'ta, her bir parametrenin farklı hızlarda öğrenmektedirler ve bu hızları otomatik seçilir. RMSProp öğrenme oranı, üstel eğimlerin karesinin sönüm ortalamasına bölmektedir.

$$\theta_{t+1} = \theta_t - \frac{n}{\sqrt{(1-y)g_{t-1}^2 + yg_t + \varepsilon}}\ x\ g_t$$

Güncellenen θ İfadesi

Uyarlamalı Momentum (Adam)

Adam, herbir Parametre de uyarlamalı öğrenme oranını, gradyanların 1. ve 2. Momentlerinden tahmini sonuç hesaplayan bir optimizasyon algoritmasıdır. Adagrad'ın azalmakta olan öğrenme oranlarını daha da azaltmaktadır. Adam, RMSprop ile Adagrad birleşimi gibi düşünülebilir. Adam hesaplamalarda daha verimli ve belleğin az kullanımından dolayı en popüler gradyan iniş optimizasyon algoritmalarından biridir.

$$\theta_{t+1} = \theta_t - \frac{n}{\sqrt{\widehat{v_t}} + \varepsilon} \times \widehat{m_t}$$

3.3 Kullanılan Araçlar ve Kütüphaneler

Veri setinin modellerken, test ve eğitim yaparken Python diliyle kodlanmıştır. Modelleri oluştururken TensorFlow ve Keras kütüphaneleri kullanılmıştır. Modelleri eğitirken ilk başta veri setini test ve doğrulama olarak ayırma işlemini gerçekleştirilmiştir. MNIST veri setinde 60000 eğitim, 10000 test görüntüsü bulunmaktadır.

3.3.1 Tensorflow

Açık kaynak kodlu bir deep learning kütüphanesidir. Tensorflow kütüphanesi tek bir APU ile birden çok platformda kullanılabilmektedir. Mobil uygulama, web uygulaması veya IoT cihazlarda projeler geliştirirken projelerde bu kütüphaneyi kullanım sağlayabiliriz. Hesaplamaları birden çok CPU ve GPU kullanarak uygulamamıza imkân sağlar. Google'ın makine ve derin sinir ağları için geliştirdiği bir kütüphanedir.

3.3.2 Keras

Keras Python diliyle yazılan üst düzey bir sinir ağı kütüphanesidir (API). Theano ve Tensorflow üzerine kurulmuştur. Deneyleri daha hızlı sonuca ulaştırabilmek için tasarlanmıştır. Açık kaynaklı, kullanıcılar için çok kolay bir yapıda ve genişletilebilir. Hem CPU hem de GPU üzerinde çalışıp birçok algoritmayı desteklemektedir. Derin öğrenme yöntemlerinde modellerin oluşumunda Tensorflow'a göre daha kolay ve basit bir yapıdadır.

3.4 Modelleri Oluşturma

Bu kısımda modeller incelenmektedir. Modellerin giriş ve çıkış katmanları, gizli katmanlar, evrişim sayısı, kaç epokta işlemlerin yapıldığı ve bu işlemler sonucu ortaya çıkan kayıp ve doğruluk oranlarının kaç çıktığı bulunmaya çalışılmaktadır.

3.4.1 Evrişimsel Sinir Ağı (CNN)

Evrişimsel Sinir Ağı modeli için kayıp(Loss) değeri olarak categorical crossentropy kullanılmıştır. Optimizasyon algoritması olarak ise AdaDelta kullanılmıştır. Aktivasyon olarak ReLU Fonksiyonu kullanılmıştır.

Katman	Çekirdek Veya havuz(pool) boyut	Nöron Sayısı	Seyreltme Katmanı
Evrişim (Convolution)	3	32	-
Havuzlama (Pooling)	3	-	0.25
Evrişim (Convolution)	2	64	-
Havuzlama (Pooling)	2	-	0.5

CNN seçimi için oluşturulan Model

Oluşturulan model 5 epok eğitildikten sonra aşağıdaki tabloda doğruluk, kayıp, geçerleme doğruluk oranı ve geçerleme kayıp oranı şu şekilde çıkmaktadır;

Epoch	Loss	Accuracy	Val_Accuracy	Val_Loss
1	0.3319	0.8952	0.9789	0.0654
2	0.1070	0.9684	0.9871	0.0402
3	0.0769	0.9769	0.9895	0.0323
4	0.0643	0.9807	0.9883	0.0331
5	0.0539	0.9841	0.9888	0.0293

CNN için Doğruluk Tablosu

3.4.2 Yinelemeli Sinir Ağı (RNN)

RNN modelinde, iki katman ile model oluşturulmuştur. Her iki katman içinde nöron sayısı 128 olarak belirlenmiştir. Loss değeri olarak categorical crossentropy kullanılmıştır. Optimizasyon olarak ise SGD kullanılmıştır.

Epoch	Loss	Accuracy	Val_Accuracy	Val_Loss
1	0.9514	0.6998	0.8386	0.5109
2	0.3897	0.8818	0.8891	0.3359
3	0.2471	0.9268	0.9211	0.2535
4	0.1917	0.9424	0.9237	0.2353
5	0.1616	0.9512	0.9186	0.2343

RNN için Doğruluk Tablosu

3.4.3 Çok Katmanlı Algılayıcı (MLP)

MLP modeli, 3 katmandan oluşmaktadır. Loss değeri olarak categorical crossentropy kullanılmıştır. optimizasyon olarak ise SGD kullanılmıştır. Aktivasyon olarak ise hiperbolik fonksiyon kullanılmıştır.

Katman	Nöron Sayısı	Seyreltme Katmanı
1.Katman	256	0.4
2.Katman	128	0.4
3.Katman	100	0.4

MLP Seçiminde Oluşturulan Model

Oluşturulan model 5 epok çalıştırdıktan sonra aşağıdaki tabloda doğruluk, kayıp, geçerleme doğruluk oranı ve geçerleme kayıp oranı şu şekilde çıkmaktadır;

Epoch	Loss	Accuracy	Val_Accuracy	Val_Loss
1	0.8190	0.7526	0.9184	0.2702
2	0.3721	0.8912	0.9333	0.2279
3	0.3140	0.9095	0.9432	0.1948
4	0.2782	0.9195	0.9488	0.1804
5	0.2591	0.9253	0.9523	0.1670

MLP için Doğruluk Tablosu

3.4.4 Uzun Kısa Süreli Bellek(LSTM)

LSTM için Loss değeri olarak categorical crossentropy kullanılmıştır. Optimizasyon olarak ise rmsprop kullanılmıştır. Aktivasyon olarak softmax kullanılmıştır.

Epoch	Loss	Accuracy	Val_Accuracy	Val_Loss
1	0.4666	0.8431	0.9671	0.1129
2	0.1170	0.9653	0.9739	0.0897
3	0.0813	0.9757	0.9756	0.769
4	0.618	0.9818	0.9829	0.0562
5	0.0512	0.9851	0.9832	0.0560

LSTM için Doğruluk Tablosu

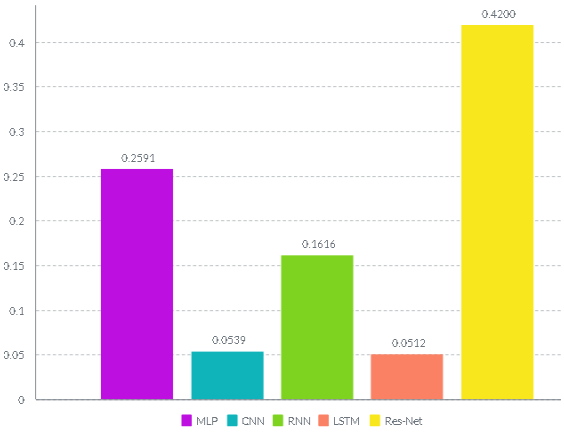
3.4.5 Kalıntı Sinir ağı (ResNet)

Kalıntı Sinir Ağı için Loss değeri olarak categorical crossentropy kullanılmıştır. Optimizasyon olarak ise adam kullanılmıştır. Aktivasyon olarak relu kullanılmıştır.

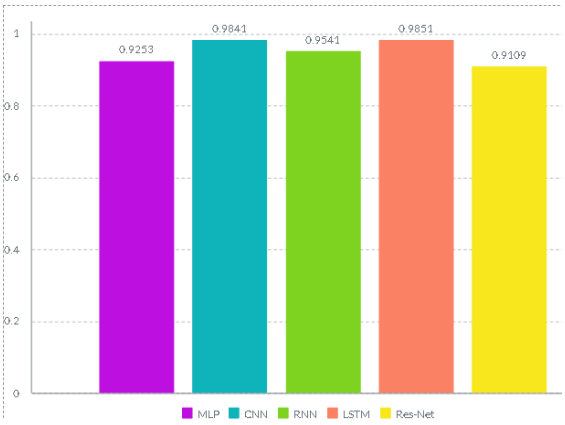
Epoch	Loss	Accuracy	Val_Accuracy	Val_Loss
1	1.1420	0.7623	0.3850	4.4300
2	0.9728	0.7904	0.7657	0.8404
3	0.4727	0.9042	0.8776	0.8744
4	0.4701	0.9000	0.9505	1.1006
5	0.4200	0.9109	0.2892	167.98

ResNet için Doğruluk Tablosu

3.5 Algoritmaların Karşılaştırılması



Şekil 20 – Epokların Son Durumuna Göre Loss Oranları



Şekil 21 – Epokların son Durumuna göre Doğruluk Oranı

Oluşturulan modellerin performansını iyi olarak değerlendirebilmek için doğruluk değerinin eğitim ve test sırasında yüksek, kayıp değerinin ise düşük çıkması beklenmektedir. Yukarıdaki görüldüğü üzere şekillerde MNIST test verileri üzerinde %98.51 oranında doğru ve 0.0512 oranında kayıp değeri olan LSTM modeli performans olarak diğerlerinden daha iyi olduğu sonucuna ulaşılmıştır. Oluşturulan modellerde, performansı en düşük %91 oranında doğruluk ve 0.4200 kayıp değerine sahip ResNet modeli olduğuna sonucuna ulaşılmıştır.

3.6 Sonuçların karşılaştırılması

Mnist veri seti üzerinde yapmış olduğum çalışmaya benzer başka araştırmacılar tarafından da incelemeler yapılmıştır. Aoudou SALOUHOU yapmış olduğu araştırmada CNN, RNN ve DNN algoritmalarını kullanmıştır. Yapmış olduğu çalışma sonucunda farklı bu veri seti üzerinden en iyi çalışan model olarak CNN modeli olduğuna ulaşmıştır. Bu çalışmada ise tüm algoritmaları 5 epok süresince karşılaştırıldığında en iyi algoritma olarak LSTM algoritmasının daha iyi çalıştığına ulaşılmıştır. Aoudou SALOUHOU Kullandığı algoritmalar sonucunda RNN algoritmasında 60 epok çalıştırdıktan sonra %99.05 doğruluk oranına ve %0.048 kayıp oranına ulaşmıştır. Bu çalışmada ise yapılan deneyde 60 epok çalıştırıldığında %99.12 doğruluk oranına %0.028 kayıp oranına ulaşılmıştır. Aoudou SALOUHOU 80 epok sonucunda CNN algoritmasında doğruluk oranını %99.88 ve kayıp oranını 0.042 olarak bulmuştur. Bu çalışmada ise 80 epok sonucunda %99.73 doğruluk oranı ve % 0.0259 kayıp oranına ulaşılmıştır. Bu deneyde diğer çalışmaya ek olarak ResNet ve MLP algoritmalarında da Mnist veri setini karşılaştırılmıştır.

İhsan PENÇE vd. yapmış oldukları araştırmada benimde kullanmış olduğum yapay sinir ağları algoritmasını ve naive bayes'i kullanarak mnist veri setinin kapalı cebirsel eğriler ile modellemesi üzerinde durmuşlardır. Burada yapay sinir ağları algoritmasıyla %95.80 oranında tanıma oranına ulaşmışlardır. Bu çalışmada ise 5 epok üzerinden baz alındığında doğruluk oranı daha az olarak gözükmektedir. Ancak epok sayısı yükseltildiğinde %95.94 oranında doğruluk oranına ulaşılmıştır.

4. Sonuçlar

Çok Katmalı Algılayıcı, Evrişimsel Sinir Ağı, Yinelemeli Sinir Ağı, Uzun Kısa Süreli Bellek ve ResNet modelleri oluşturularak algoritmaların performansları MNIST veri seti üzerinde denenmiştir. Çalışmalardaki bulgular şu şekilde çıkmaktadır;

Oluşturulan tüm modellerde doğruluk değerinin gittikçe arttığı ve kayıp değerlerinin gittikçe azaldıkları kaydedilmiştir.

Mnist veri seti üzerinde LSTM modeli en düşük kayıp oranına ve en yüksek doğruluk oranına sahiptir.

Süre olarak en uzun süren algoritma Res-Net olarak hesaplanmıştır. Res-Net'ten sonra en yavaş yöntemler sırasıyla LSTM, CNN, RNN olarak hesaplanmıştır ve en hızlı yöntem olarak ise MLP hesaplanmıştır.

ResNet modeli %91 oranında doğruluk ve %0.42 loss değerine ulaşmıştır. Bu değerler ile birlikte kullanılan algoritmalar içinde en düşük değere sahip olmaktadır.

KAYNAKÇA

- Kızrak, M.A., Bolat, B. (2018). Derin Öğrenme ile Kalabalık Analizi Üzerine Detaylı Bir Araştırma. Bilişim Teknolojileri Dergisi, 11(3), 263–286.
- Skymind, "A Beginner's Guide to LSTMs and Recurrent Neural Networks | Skymind", skymind.ai, 2018. [Çevrimiçi]. Available at: https://skymind.ai/wiki/lstm. [Erişim: 17-Ağu-2018].
- C. Cortes, C. J. Burges ve Y. LeCun, "MNIST handwritten digit database", yann.lecun.com. [Çevrimiçi]. Available at: http://yann.lecun.com/exdb/mnist/. [Erişim: 15-Ağu-2018].
- T. B. Ludermir ve W. R. De Oliveira, "Particle Swarm Optimization of MLP for the identification of factors related to Common Mental Disorders", Expert Syst. Appl., c. 40, sayı 11, ss. 4648–4652, 2013.
- SALOUHOU, Aoudou, El Yazısı Karakter Tanıma ve Resim Sınıflandırmada Derin Öğrenme Yaklaşımları, Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı, Yayımlanmamış Yüksek Lisans Tezi, 2019.
- http://www.nuhazginoglu.com/2018/05/15/cok-katmanli-algilayicilar-multi-layer-perceptron/
- T. Flow, https://www.tensorflow.org/
- Keras Google group, "Keras: The Python Deep Learning library", keras.io. [Çevrimiçi]. Available at: https://keras.io/. [Erişim: 27-Tem-2018].
- https://medium.com/@ayyucekizrak/deri%CC%87ne-daha-deri%CC%87ne-evri%C5%9Fimli-sinir-a%C4%9Flar%C4%B1-2813a2c8b2a9
- https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
- *Bayır, F. (2006).* Yapay sinir ağları ve tahmin modellemesi üzerine bir uygulama. *Yüksek Lisans Tezi, İstanbul Üniversitesi Sosyal Bilimler Enstitüsü, İstanbul.*
- http://www.teknouzay.com
- https://www.kaggle.com/keras/resnet50

- NIST handprinted forms and characters database(Special No. 19), National Institute of Standarts and Technology U.S. Department of Commerce, http://www.nist.gov/srd, (2013).