T.C.
MUĞLA SITKI KOÇMAN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY ZEKA ANABİLİM DALI

MAKİNE ÖĞRENMESİNDE GRADYAN İNİŞİ OPTİMİZASYON ALGORİTMALARI ÜZERİNE

YÜKSEK LİSANS TEZİ

DOĞAN ÇAKAR

OCAK 2022
MUĞLA

D. ÇAKAR YAPAY ZEKA ANABİLİM DALI YÜKSEK LİSANS TEZİ MUĞLA 2022

T.C.
MUĞLA SITKI KOÇMAN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY ZEKA ANABİLİM DALI

MAKİNE ÖĞRENMESİNDE GRADYAN İNİŞİ OPTİMİZASYON ALGORİTMALARI ÜZERİNE

YÜKSEK LİSANS TEZİ

DOĞAN ÇAKAR

OCAK 2022
MUĞLA

MUĞLA SITKI KOÇMAN ÜNİVERSİTESİ

Fen Bilimleri Enstitüsü

TEZ ONAYI

DOĞAN ÇAKAR tarafından hazırlanan **MAKİNE ÖĞRENMESİNDE GRADYAN İNİŞİ OPTİMİZASYON ALGORİTMALARI ÜZERİNE** başlıklı tezinin, 28/01/2022 tarihinde aşağıdaki jüri tarafından Yapay Zeka Anabilim Dalı'nda yüksek lisans derecesi için gerekli şartları sağladığı oybirliği/oyçokluğu ile kabul edilmiştir.

TEZ SINAV JURİSİ

Doç. Dr. Derya DOĞAN DURGUN **(Jüri Başkanı)** İmza:

Matematik Anabilim Dalı,
Celal Bayar Üniversitesi, Manisa _____

Doç. Dr. Gamze YÜKSEL **(Danışman)** İmza:

Matematik Anabilim Dalı,
Muğla Sıtkı Koçman Üniversitesi, Muğla _____

Dr. Öğr. Üyesi Nida GÖKÇE NARİN **(Üye)** İmza:

İstatistik Anabilim Dalı, _____

Muğla Sıtkı Koçman Üniversitesi, Muğla

ANA BİLİM DALI BAŞKANLIĞI ONAYI

Doç. Dr. Gamze YÜKSEL İmza:

Yapay Zeka Ana Bilim Dalı Başkanı,
Muğla Sıtkı Koçman Üniversitesi, Muğla _____

Doç. Dr. Gamze YÜKSEL İmza:

Danışman, Yapay Zeka Ana Bilim Dalı,
Muğla Sıtkı Koçman Üniversitesi, Muğla _____

Savunma Tarihi: 28/01/2022

Tez çalışmalarım sırasında elde ettiğim ve sunduğum tüm sonuç, doküman, bilgi ve belgelerin tarafımdan bizzat ve bu tez çalışması kapsamında elde edildiğini; akademik ve bilimsel etik kurallarına uygun olduğunu beyan ederim. Ayrıca, akademik ve bilimsel etik kuralları gereği bu tez çalışması sırasında elde edilmemiş başkalarına ait tüm orijinal bilgi ve sonuçlara atıf yapıldığını da beyan ederim.

Doğan Çakar

28/01/2022

ÖZET

MAKİNE ÖĞRENMESİNDE GRADYAN İNİŞİ OPTİMİZASYON ALGORİTMALARI ÜZERİNE

Doğan ÇAKAR

Yüksek Lisans Tezi

Fen Bilimleri Enstitüsü

Yapay Zeka Anabilim Dalı

Danışman: Doç. Dr. Gamze YÜKSEL

Ocak 2022, 76 sayfa

Bu çalışmada, makine öğrenmesinde önemli bir yere sahip olan gradyan tabanlı optimizasyon algoritmalarının yapısı, çeşitleri, avantaj ve dezavantajlarına yer verilmiştir. Bu amaçla 1. dereceden optimizasyon algoritmalarından literatürde en çok kullanılan; Stokastik Gradyan İniş, Momentum, Nesterov Momentum, AdaGrad, Adadelta, RMSProp, Adam ve Nadam algoritmaları ile 2. Dereceden optimizasyon algoritmalarından Newton, BFGS ve L-BFGS algoritmaları ele alınmıştır.

Algoritmaların matematiksel yapıları incelenmiş ve karşılaştırmaları için üç farklı gerçek hayat problemi ele alınmıştır. Bu problemlerin yapay zeka modelleri ile çözümlerinde ResNet50, VGG19 ve lojistik regresyon modelleri kullanılmıştır. Elde edilen sonuçlar çizelge ve şekiller üzerinden değerlendirilmiştir. Algoritmaların performansları metriklerle ölçülerek, algoritmaların hem birbirlerine karşı performansları hem de modellerde ki başarımları tespit edilmiş ve sonuçları yorumlanmıştır.

Anahtar Kelimeler: Makine öğrenmesi, Derin öğrenme, Gradyan İniş Algoritmaları, 1. dereceden optimizasyon algoritmaları, 2. Dereceden optimizasyon algoritmaları

ABSTRACT

ON GRADIENT DESCENT OPTIMIZATION ALGORITMS IN MACHINE LEARNING

Doğan ÇAKAR

Master of Science (M.Sc.)

Graduate School of Natural and Applied Sciences

Department of Artificial Intelligence

Supervisor: Assoc. Prof. Dr. Gamze YÜKSEL

January 2022, 76 pages

In this study, the structure, types, advantages, and disadvantages of the gradient-based optimization algorithms which have an important place in machine learning are represented. For this purpose, the most known 1st order optimization algorithms in the literature; Stochastic Gradient Descent, Momentum, Nesterov Momentum, AdaGrad, Adadelta, RMSProp, Adam and Nadam algorithms and Newton, BFGS and L-BFGS algorithms from 2nd order optimization algorithms are handled.

The mathematical structures of these algorithms are examined and three different real-life problems are discussed for the comparison of the algorithms. ResNet50, VGG19, and logistic regression models are applied to solve these problems with artificial intelligence models. The results are evaluated through tables and figures. By measuring the performances of the algorithms with metrics, both the performances of the algorithms against each other and the performances in the models are determined and the results are interpreted.

Keywords: Machine Learning, Deep Learning, Gradient Descent, 1st order optization algorithms, 2nd order optimization algorithms

ÖNSÖZ

Tez çalışması süreci boyunca bilgi ve birikimini benimle paylaşan ve bana yol gösterici ve destek olan değerli danışman hocam Doç. Dr. Gamze Yüksel'e teşekkürü bir borç bilirim.

Çalışmalarım boyunca beni yalnız bırakmayan sevgili eşim Başak Çakar'a sonsuz teşekkür ederim.

İÇİNDEKİLER

ÇİZELGELER DİZİNİ

ŞEKİLLER DİZİNİ

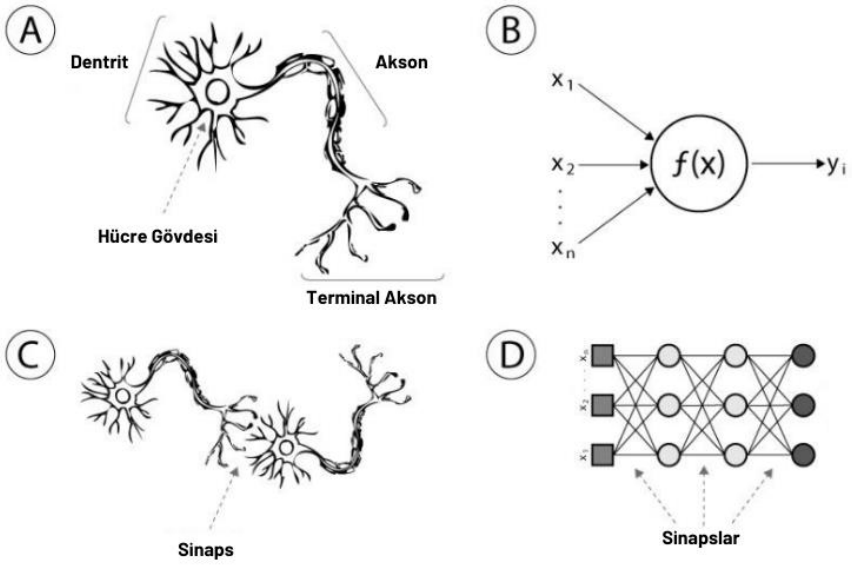
SEMBOLLER VE KISALTMALAR DİZİNİ

x^i	i. Satırdaki Girdi Değeri
y^i	Gerçek Etiket Değeri
$\hat{y}^i$	Tahmin Edilen Etiket Değeri
$\nabla_\theta J_\theta$	Gradyan Değeri
θ	Ağırlık Değerleri
$\in$	Öğrenme Oranı
v	Hız Değeri
g	Gradyan Kestrimi
α	Momentum Parametresi
δ	Sayısal Denge Sabiti
ρ	Azalma Oranı
H	Hessian Matrisi
H^{-1}	Hessian Matrisinin Tersi
M_t	Hessian Matrisinin Yaklaşık Tersi
$\sigma(x)$	Sigmoid Fonksiyonu
SGD	Stokastik Gradyan İniş
DSA	Derin Sinir Ağları
YSA	Yineleyen Sinir Ağları
ESA	Evrişimli Sinir Ağları
CONV2D	Evrişimsel Katman
RSNA	Kuzey Amerika Radyoloji Derneği

1. GİRİŞ

1.1. Yapay Zeka

Yapay zeka, insan tarafından yapıldığında zeki olarak adlandırılan davranışların makine tarafından yapılmasıdır (Pirim, 2006). Yapay zeka beyinden ilham alır. Kullanılan modeller, insanlardaki sinir ağlarına benzemektedir ve yapay sinir ağları olarak adlandırılmaktadır. Biyolojik sinir hücresi ve yapay sinir ağı Şekil 1.1.'de gösterilmiştir.



Şekil 1.1. Biyolojik sinir hücresi ve yapay sinir ağı (Maltarollo, 2013)

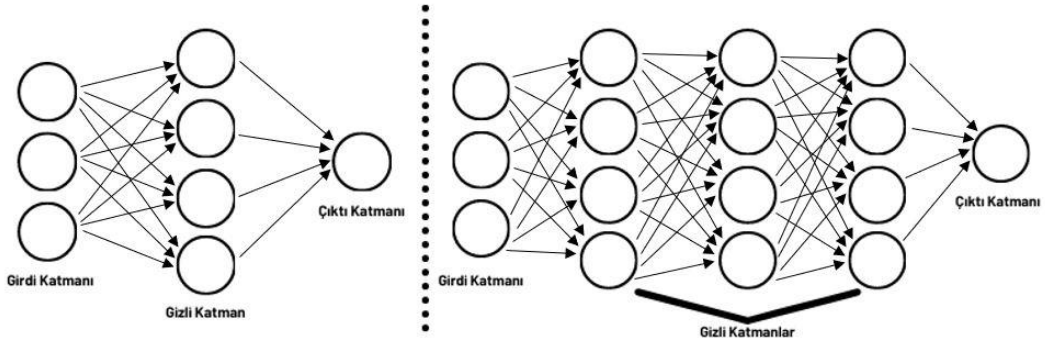
Şekil 1.1.'de A biyolojik sinir hücresi yapısını, B yapay sinir hücresi yapısını, C biyolojik sinir hücreleri arasındaki veri aktarımının sinapslar yardımıyla nasıl gerçekleştiğini ve D yapay sinir hücreleri arasındaki veri aktarımının nasıl gerçekleştiğini göstermektedir.

Yapay sinir ağları girdi katmanı, gizli katman ve çıktı katmanı olmak üzere 3 bölümden oluşur. Bilgiler ağa girdi katmanından iletilir. Gizli katmanda/katmanlarda işlenerek çıktı katmanına gönderilir.

Makine öğrenmesi, yapay zekanın bir alt kümesi olarak değerlendirilir. Derin öğrenmede makine öğrenmesinin bir alt kümesidir. Makine öğrenmesi ve derin öğrenme arasındaki ilişki Şekil 1.2.'de gösterilmiştir.

Şekil 1.2. Makine öğrenmesi ve derin öğrenme arasındaki ilişki

Makine öğrenmesi modelleri genel olarak tek katmanlı ve çok katmanlı yapay sinir ağlarını temel almaktadır. Derin öğrenme modellerinde, her biri önceki katmanın sonuçlarını iyileştiren birden çok gizli katmana sahip çok katmanlı yapay sinir ağları temel alınmaktadır. Tek ve çok katmanlı sinir ağlarına örnekler Şekil 1.3.'de gösterilmiştir.



Şekil 1.3. Tek ve çok katmanlı sinir ağları

1.2. Makine Öğrenmesi

Makine öğrenmesi, matematik ve istatistik yardımı ile veri analizi yaparak tahminlerde bulunan sistemlerin bilgisayarlar ile modellenmesidir. Başka bir ifadeyle, insan zekasının belli bir parçasını taklit etmeye çalışan sistemlerin genel adıdır. Alpaydın'a (2010) göre, makine öğrenmesi, örnek verileri veya geçmiş deneyimleri kullanarak bir performans kriterini optimize etmek için bilgisayarları programlamaktır.

Makine öğrenmesinin birçok uygulama alanı vardır. Bu alanlara her geçen gün yenileri eklenmektedir. Bankacılık, finans, sağlık hizmetleri, ulaşım, tarım, perakende, turizm ve müşteri hizmetleri gibi birçok önemli sektör makine öğrenmesinden yararlanmaktadır.

Makine öğrenmesinin merkezinde üç önemli kavram vardır: veri, model, öğrenme. Veri, makine öğrenmesinin merkezinde yer aldığı için son yıllarda çok önemli hale gelmiştir. Veriden anlamlı bilgiler öğrenilmesi için modeller geliştirilir. Makine öğrenmesinde denetimli öğrenme, denetimsiz öğrenme ve pekiştirmeye dayalı öğrenme olmak üzere üç öğrenme tekniği vardır.

Denetimli öğrenme, sonuçları bilinen (etiketlenmiş) verileri kullanarak bu verileri ve sonuçlarını kapsayan bir fonksiyon oluşturmayı amaçlayan makine öğrenimi tekniğidir (Nizam, 2014). Model, bilinen girdilerden ve bu girdilere karşılık gelen sonuçlardan öğrenmeyi gerçekleştirir. Örneğin; bir e-postanın spam olup olmadığını makineye öğretebilmek için girdi olarak e-postaları ve çıktı olarak spam olan e-postaları verilir. Girdi ve çıktı değerleri arasındaki ilişkiyi ortaya koyan hata/kayıp fonksiyonu tahminlerin ne kadar doğru olduğunu ortaya koyar. Hata fonksiyonu minimize olana kadar bu optimizasyon işlemi devam eder.

Denetimsiz öğrenme, etiketli verilerin kullanılmadığı makine öğrenimi tekniğidir. Kullanıcının sisteme herhangi müdahalesi yoktur. Sadece girdi verileri sisteme verilir ancak herhangi bir işaretleme yapılmaz. Sistem otomatik olarak keşifler yapar, ilişki ağını ortaya koymaya çalışır (Alpaydın, 2010). Bu teknikte veriler kümeler halinde gruplandırılarak kalıplar ve ilişkiler bulunur.

Pekiştirmeye dayalı öğrenme, sayısal bir ödül sinyalini en üst düzeye çıkarmak için ne yapılacağını (durumları eylemlerle nasıl eşleştireceğini) öğrenmektir (Sutton ve Barto,

2018). Robotik oyun programlama, fabrika otomasyonu gibi alanlarda sıklıkla kullanılır.

Makine öğrenmesinde matematik önemli bir yere sahiptir. Verilerin işlenmesinden modeller içerisindeki algoritmalara kadar matematik bilgisi gerekmektedir. Lineer Cebir, Vektör Kalkülüs, Olasılık ve İstatistik, Analitik Geometri, Optimizasyon ve Matris Ayrışımı gibi matematik alanları makine öğrenmesinde bilinmesi gereken temel konulardır. Makine öğrenmesinin temelleri Şekil 1.4.'de gösterilmiştir.

Şekil 1.4. Makine öğrenmesinin temelleri (Deisenroth, 2020)

1.3. Optimizasyon

Optimizasyon, makine öğreniminin önemli bir parçasıdır. Optimizasyon, bir amaç fonksiyonunun belirli parametrelere bağlı olarak en düşük/yüksek hale getirilmesidir. Hemen hemen her makine öğrenimi algoritmasının özünde bir optimizasyon algoritması vardır. Makine öğrenmesinde optimizasyonun amacı global minimumun/maksimumun bulunmasıdır.

$$f: R^d \to R,\ x \in R^d, \quad minf(x)$$

Optimizasyonda sıklıkla kullanılan algoritmalar gradyan tabanlı algoritmalardır. Teknolojik gelişmelere ve verinin büyümesine paralel olarak yeni optimizasyon algoritmalarına ihtiyaç artmaktadır. Bu sebeple mevcutta kullanılan optimizasyon algoritmalarının geliştirilmesi çalışmaları önem kazanmıştır.

Gradyan tabanlı algoritmalar, türevlenebilir bir fonksiyonun yerel bir minimumunu bulmak için kullanılan yinelemeli optimizasyon algoritmalarıdır. Bu algoritmalardaki fikir, mevcut noktada fonksiyonun gradyanının tersi yönünde tekrarlanan adımlar atmaktır. Çünkü bu en dik iniş yönüdür.

Makine öğrenmesinde amaç, modeli daha iyi tahmin yapabilmesi için optimize edilmiş ağırlıklara sahip olacak şekilde eğitmektir.

Literatürde gradyan tabanlı algoritmalar ile ilgili birçok çalışma yapılmıştır. Bu tür çalışmalara ihtiyaç günümüzde devam etmektedir. Polyak (1964), momentum algoritmasını geliştirmiştir. Sutton (1986), gradyan iniş için momentum kavramı üzerine çalışmıştır. Darken, Chang ve Moody (1992), SGD üzerine optimizasyon iyileştirmesi için öğrenme oranlarını daha iyi belirlemeye yönelik çalışma yapmışlardır. Bengio, Louradour, Collobert ve Weston (2009), optimizasyon sürecine rehberlik etmek için algoritmaları güçlendirme konusunda çalışmışlardır. Niu vd. (2011), SGD için Hogwild Algoritmasını geliştirmişlerdir. Duchi, Hazan, ve Singer (2011), AdaGrad algoritmasını geliştirmişlerdir. Bengio vd. (2013), SGD'yi iyileştirmeye yönelik çalışmalar yapmışlardır. Dean vd. (2012), derin ağ eğitimi sorununu ele almışlar ve Downpour SGD ve Sandblaster algoritmalarını geliştirmişlerdir. Zeiler (2012), Adadelta algoritmasını yayınlamıştır. Sutskever vd. (2013), Nesterov momentum algoritmasını geliştirmiştir. Hinton (2012) tarafından uyarlanabilir bir öğrenme oranı yöntemi olan RMSProp geliştirilmiştir. Dauphin vd. (2014) dışbükey olmayan optimizasyon için eyer noktası sorununu tanımlamış ve bunu aşmak için yeni bir algoritma önermişlerdir. Kingma ve Ba (2014) RMSProp ve Momentumu birleştirerek Adam algoritmasını geliştirmişlerdir. Zhang, Choromanska ve LeCun (2014), Downpour algoritmasınının test hatalarını iyileştiren yeni bir algoritma çalışması yapmışlardır. Dozat (2016), Adam algoritmasını iyileştirmeye çalışarak Nadam algoritmasını geliştirmiştir. Goodfellow, Bengio ve Courville (2016) tarafından Derin Öğrenme kitabı yayınlanmıştır. Deisenroth, Faisal ve Ong (2020) tarafından Makine Öğrenmesi için Matematik kitabı yayınlanmıştır.

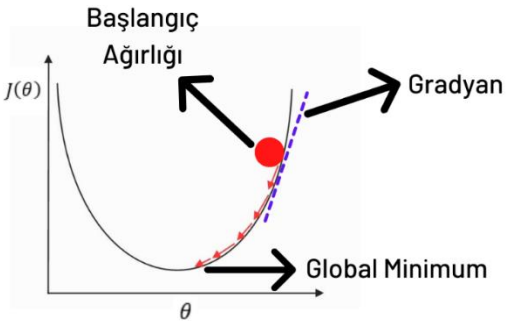
2. MALZEME VE YÖNTEM

2.1. Gradyan İniş

Birçok makine öğrenme algoritması bir tür optimizasyon problemi içerir. Optimizasyon bir f fonksiyonunun parametreleriyle oynayarak minimum ya da maksimum değerini bulmayı amaçlar. Gradyan iniş algoritmalarında genel olarak hata/kayıp fonksiyonu minimum yapılmaya çalışılır. Bu şekilde öğrenmenin daha iyi hale geldiği gösterilir.

x ve y sayılarının gerçel olduğu bir $y = f(x)$ fonksiyonunun türevi x noktasındaki eğime eşittir. Türev y değerinde küçük bir iyileştirme yapmak için x değerini ne kadar değiştirmemiz gerektiği hakkında bilgi verir. Bu yüzden kullanışlıdır. $f(x)$ fonksiyonunu, x değerini türevin tersi yönde küçük adımlarla ilerleterek küçültebiliriz. Bu tekniğe gradyan iniş (Cauchy, 1847) denir.

Şekil 2.1.'de gradyan iniş tekniği gösterilmiştir.



Şekil 2.1. Gradyan iniş (Srivastava, 2021)

Türev 0 değerine eşit olduğunda hangi yöne doğru ilerlenmesi gerektiği konusunda bilgi sağlamaz. Bu yüzden türevin 0 olduğu noktalar kritik noktalar olarak bilinirler. Yerel minimum, yerel maksimum ve eyer noktası kritik nokta çeşitleridir. Yerel minimum fonksiyonun bütün komşu noktalardan daha küçük olduğu noktadır. Yerel maksimum fonksiyonun bütün komşu noktalardan daha büyük olduğu noktadır. Bazen

kritik nokta ne minimum ne de maksimum noktadır. Bu tür kritik noktalar eyer noktası olarak bilinir. Şekil 2.2.'de kritik nokta çeşitleri gösterilmiştir.

Şekil 2.2. Kritik nokta çeşitleri (Goodfellow vd., 2016)

Birden fazla değişkeni olan fonksiyonlar için kısmi türevleri kullanırız. $\frac{\partial}{\partial x} f(x)$ şeklinde ifade edilen kısmi türev, f fonksiyonunun yalnızca x değişkenindeki farklı değerler için ne kadar değiştiğini ölçer. Gradyan kavramı, türevi, vektörlerin türevini kapsayacak şekilde genelleştirir. f fonksiyonunun gradyanı, $\nabla_x f(x)$ şeklinde ifade edilen ve bütün kısmi türevleri içeren bir vektördür. Gradyan çeşitlerinden bahsetmeden önce bilinmesi gereken üç kavram vardır. Bunlar; hata/kayıp fonksiyonu, maliyet fonksiyonu ve öğrenme oranı kavramlarıdır.

Hata/kayıp fonksiyon tek bir eğitim örneğinin hatasını hesaplar. Bunu çıktı değerini gerçek değer ile kıyaslayarak yapmaktadır. Denklem (2.1)'de bu durum ifade edilmektedir (Gülcü vd., 2010).

$$L\left(\hat{y}^i, y^i\right) = L(f\left(x^i; \theta\right), y^i) \tag{2.1}$$

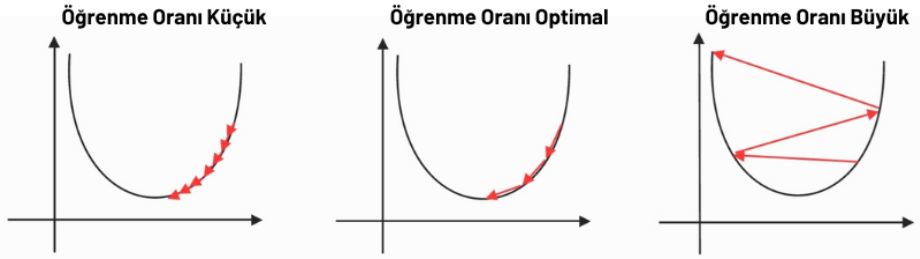
Burada, y^i gerçek etiket değeri, $\hat{y}^i$ tahmin edilen etiket değeri, x^i eğitim örneği ve θ model parametreleri (ağırlık değerleri) olarak ifade edilir.

Maliyet fonksiyonu tüm eğitim setinin (m) hata/kayıp fonksiyonlarının ortalamasıdır. Denklem (2.2)'de bu durum ifade edilmektedir (Gülcü vd., 2010).

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} L\left(\hat{y}^i, y^i\right) \tag{2.2}$$

Öğrenme oranı, gradyan inişde minimuma ulaşmak için atacağımız adımın büyüklüğünü gösterir. Parametre güncellerken kullanılır. Makine öğrenmesinde öğrenme oranını iyi ayarlamak gerekir. Öğrenme oranı çok küçük seçilirse yakınsama

çok yavaş olur. Öğrenme oranı çok büyük seçilirse yakınsayacağı noktayı kaçırabilir (Simon, 2009). Şekil 2.3.'de bu durum gösterilmiştir.



Şekil 2.3. Öğrenme oranı seçimi

Ne kadar veri kullandığımıza göre farklılık gösteren üç gradyan iniş çeşidi vardır.

2.1.1. Batch gradyan iniş

Batch gradyan iniş tüm eğitim kümesinin toplu olarak kullanıldığı/güncellendiği gradyan iniş çeşididir. Denklem (2.3)'de bu durum ifade edilmiştir (Ruder, 2016).

$$\theta = \theta - \in \nabla_\theta J(\theta) \tag{2.3}$$

Burada, θ parametre değerleri (ağırlıklar), $\in$ öğrenme oranı ve $\nabla_\theta J(\theta)$ gradyan olarak ifade edilir.

Batch gradyan inişte dışbükey hata yüzeyleri için global minimuma, dış bükey olmayan hata yüzeyleri için ise yerel minimuma yakınsama sağlanır. Ancak modelin çevrimiçi olarak yeni örneklerle güncellenmesine izin vermez.

Tek bir güncelleme yapmak için tüm veri setinin gradyanlarını hesaplamak gerektiğinden, büyük veri setleri için batch gradyan iniş hem yavaş hem de masraflıdır. Bundan dolayı büyük veri setlerinde tercih edilmez. Küçük veri setlerinde tam yakınsama sağladığı için batch gradyan iniş tercih edilir.

2.1.2. Mini batch gradyan iniş

Gradyanı tam bir şekilde hesaplamak zor ve masraflıdır. Çünkü tüm veri setinin kullanılması gerekir. Optimizasyon algoritmalarının çoğunluğu gradyanı tam

hesaplamak yerine yaklaşık kestirimlerini hesaplar. Bu durum bazen güncelleme/iterasyon sayısını arttırabilir. Ancak zamandan kazandırarak daha hızlı yakınsama sağlar. Mini batch gradyan inişte eğitim kümesinin tamamı yerine bir bölümü kullanılır. Bu yüzden batch büyüklüğü kavramı kullanılır. Özellikle GPU kullanımında daha iyi performans almak için batch büyüklüğü 2'nin kuvveti olacak şekilde seçilmektedir. Batch büyüklüğü makine öğrenmesi uygulamalarında daha çok 64-512 arası tercih edilmektedir. Denklem (2.4)'de mini batch gradyan iniş ifade edilmiştir (Ruder, 2016).

$$\theta = \theta - \in \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \tag{2.4}$$

Burada, θ parametre değerleri (ağırlıklar), $\in$ öğrenme oranı ve $\nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$ eğitim kümesinin bir bölümüne ait gradyan olarak ifade edilir.

Mini batch için seçilen eğitim kümeleri rastgele seçilmelidir. Makine öğrenmesi uygulamasında modelin ezber yapmaması için örneklerin sırasını değiştirmek gerekebilir.

Mini batch gradyan inişte daha kararlı yakınsama için parametre güncellemelerinin varyansı azaltılır. Büyük veri setlerinde mini batch gradyan iniş kullanılır. Günümüzde en çok tercih edilen gradyan iniş çeşididir.

m eğitim örneği için mini batch büyüklüğü/boyutu $1 < x < m$ olmalıdır. Eğer mini batch büyüklüğü m seçilirse gradyan iniş çeşidi batch gradyan iniş olur. Benzer şekilde mini batch boyutu 1 seçilirse gradyan iniş çeşidi stokastik gradyan iniş olur.

2.1.3. Stokastik gradyan iniş

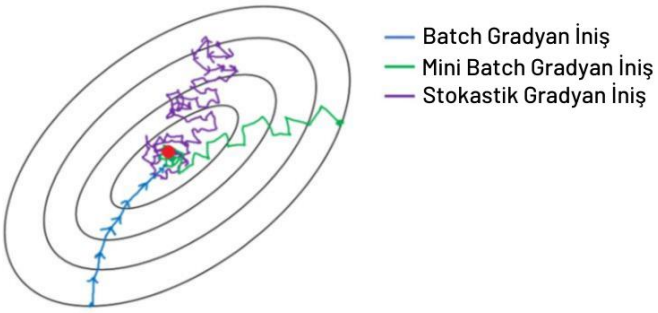
Stokastik gradyan iniş her $x^{(i)}$ eğitim örneği ve $y^{(i)}$ etiket için bir parametre güncellemesi yapmaktadır. Denklem (2.5)'de bu durum ifade edilmiştir (Ruder, 2016).

$$\theta = \theta - \in \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{2.5}$$

Batch gradyan iniş her parametre güncellemesinden önce benzer örnekler için gradyanları yeniden hesaplayarak gereksiz hesaplamalar yapar. Stokastik gradyan iniş her seferinde bir güncelleme yaparak, bu durumu ortadan kaldırır. Bundan dolayı batch

gradyan inişten daha hızlıdır. Aynı zamanda batch gradyan inişte mümkün olmayan çevrimiçi öğrenme stokastik gradyan inişte mümkündür. Stokastik gradyan inişte batch büyüklüğü 1 olduğundan amaç/hata fonksiyonunun dalgalanmasına sebep olan yüksek varyans ile sık güncellemeler yapar. Mini batch gradyan iniş batch büyüklüğünü arttırarak stokastik gradyan inişte meydana gelen varyansı azaltır. Stokastik gradyan inişte yüksek varyansdan dolayı yakınsamak zorlaşır. Ancak öğrenme oranını yavaşça düşürdüğümüzde batch gradyan iniş ile benzer yakınsama davranışı gösterdiği, dışbükey ve dışbükey olmayan optimizasyon için neredeyse kesin bir şekilde yerel veya global minimuma yakınsadığı gösterilmiştir (Ruder, 2016).

Şekil 2.4.'de gradyan iniş çeşitlerinin yakınsama durumları gösterilmiştir.



Şekil 2.4. Gradyan iniş çeşitleri (Dabbura, 2017)

2.2. Zorluklar

Gradyan tabanlı optimizasyon algoritmalarının başa çıkması gereken bazı zorluklar şöyledir:

- Uygun bir öğrenme oranı seçmek zor olabilir. Çok küçük bir öğrenme hızı, yavaş yakınsamaya yol açarken, çok büyük bir öğrenme hızı yakınsamayı engelleyebilir ve kayıp (hata) fonksiyonunun minimum civarında dalgalanmasına hatta sapmasına neden olabilir (Simon, 2009).
- Öğrenme oranı programları (Robbins ve Monro, 1951), eğitim sırasında öğrenme oranını örneğin tavlama yoluyla (öğrenme oranını önce yüksek tutup sonrasında

azaltma) ayarlamaya çalışır, yani önceden tanımlanmış bir programa göre öğrenme oranını düşürür. Bununla birlikte, bu programlar (öğrenme oranının ne şekilde ve ne zaman değişeceği) önceden tanımlanmalıdır (Darken vd., 1992)

- Aynı öğrenme hızı tüm parametre güncellemeleri için geçerlidir. Verilerimiz seyrekse ve özelliklerimiz çok farlı frekanslara sahipse, hepsini aynı ölçüde güncellemek istemeyebiliriz. Ancak nadiren oluşan özellikler için daha büyük bir güncelleme gerçekleştirebiliriz (Ruder, 2016).
- Yüksek dereceli ve dışbükey olmayan hata fonksiyonlarını, en aza indirmenin önemli bir zorluğu, sayısız optimal olmayan yerel minimumlarda sıkışıp kalmaktan kurtarmaya çalışmaktır. Yerel minimumlar global minimumdan çok daha maliyetliyse problem yaratabilirler. Uzun yıllardır birçok araştırmacı yerel minimumların optimizasyona zarar veren yaygın bir problem olduğunu düşünmüşlerdir. Araştırmacılar için yeterli büyüklükteki sinir ağlarında yerel minimumların birçoğu düşük maliyetlidir. Ayrıca gerçek global minimumu bulmak yerine düşük maliyetli ama minimum olmayan bir nokta bulmak önemli bir sorun değildir (Saxe vd., 2013; Dauphin vd., 2014; Goodfellow vd., 2014; Choromanska vd., 2015).
- Düşük boyutlu uzaylarda yerel minimumlar yaygındır. Yüksek boyutlu uzaylarda ise yerel minimumlar az görülürken eyer noktaları yaygındır. Dauphin vd. (2014), gerçek sinir ağlarının çok sayıda yüksek maliyetli eyer noktası barındıran kayıp fonksiyonlara sahip olduğunu deneysel olarak göstermektedir ve zorluğun aslında yerel minimumlardan değil eyer noktalarından, yani bir boyutun yukarı ve diğerinin aşağı eğim yaptığı noktalardan kaynaklandığını iddia etmektedir. Bu eyer noktaları genellikle aynı hataya sahip bir plato ile çevrilidir ve bu da algoritmanın burada sıkışıp kalmasına sebep olur. Eyer noktalarının çok olması birinci dereceden gradyan tabanlı optimizasyon algoritmaları için çalışılmaya devam edilen bir durumdur. Araştırmacılar bu durum hakkında ortak bir fikre sahip değildir. Goodfellow vd. (2014), gradyan tabanlı algoritmaların birçok durumda eyer noktalarından kurtulduğunu göstermiştir.
- İkinci dereceden yöntemler, birinci dereceden yöntemlerin yerini almakta başarısızdır. Bunun en önemli sebebi yüksek boyutlu uzaylardaki eyer noktalarının sayısının artmasıdır. Hessian matrisi bir yerel minimum noktasında sadece pozitif

özdeğerlere sahiptir. Eyer noktalarında ise hem pozitif hem de negatif özdeğerler içerir. Pozitif özdeğerlere ilişkin özvektörler boyunca uzanan noktalar daha maliyetli iken negatif özdeğerlerle ilişkili özvektörler boyunca uzanan noktalar daha düşük maliyetlidir (Goodfellow, Bengio ve Courville; 2016). Dauphin vd. (2014), ikinci dereceden optimizasyon için Newton yöntemini geliştirerek eyer eğrisiz Newton yöntemini sunmuştur. Ancak ikinci dereceden yöntemlerin uygulanması hala zordur.

- Parametre sayısının artması problem oluşturmaktadır. Çok katmanlı sinir ağlarında uçuruma benzeyen aşırı dik iniş bölgeleri bulunmaktadır. Bu uçurumlar parametre sayısının artmasıyla doğru orantılıdır. Bu parametrelerin birbiriyle çarpılmasıyla oluşmaktadırlar. Bu uçurum bölgelerinde gradyan aşırı büyümektedir. Bu da önceki bütün kazanımların yok olmasına sebep olmaktadır (Goodfellow, Bengio ve Courville; 2016).
- Optimizasyon algoritmalarının performansı sınırlıdır (Blum ve Rivest, 1992; Judd, 1990; Wolpert ve MacReady, 1997). Algoritmaların sınırlarını ölçebilmek ayrı bir zorluktur.

2.3. Birinci Dereceden Gradyan Tabanlı Algoritmalar

Yukarıda bahsettiğimiz zorluklarla başa çıkabilmek ve optimizasyonu en iyi hale getirebilmek için gradyan tabanlı algoritmalar geliştirilmiştir. Günümüzde de bu konuda bir çok çalışma yapılmaya devam etmektedir.

2.3.1. Stokastik gradyan iniş (SGD)

Daha önceden sabit bir öğrenme oranı kullanan SGD yönteminden bahsedilmiştir. Ancak uygulamada bu oranı zamanla azaltmak gerekir. Çünkü SGD gürültülü bir yakınsama yapmaktadır ve bu durumu minimuma yaklaştığında da devam ettirmektedir (Goodfellow, Bengio ve Courville; 2016).

Uygulamada öğrenme oranını t adımına kadar $\alpha = \frac{k}{t}$ parametre atamasıyla şu şekilde doğrusal olarak azaltmak yaygındır:

$\in_k$, k adımındaki öğrenme oranı olmak üzere,

$$\in_k = (1 - \alpha) \in_0 + \alpha \in_t \quad (2.6)$$

şeklindedir. t adımından sonra öğrenme oranı genellikle sabit tutulur.

k eğitim adımındaki Stokastik Gradyan İniş (SGD) algoritmasının güncellenmesi aşağıdaki gibi yapılmaktadır (Goodfellow, Bengio ve Courville; 2016):

Algoritma1 SGD algoritması kodu

Gerekli: $\in_k$ öğrenme oranı

Gerekli: θ parametresinin ilk değeri

$\{x^{(1)}, \ldots, x^{(m)}\}$ eğitim kümesinden m örneklik bir mini-yığına karşılık gelen $y^{(i)}$ hedefleriyle birlikte örnekle

Gradyan kestrimini hesapla: $\hat{g} \leftarrow +\frac{1}{m}\nabla_\theta \Sigma_i L\left(f\left(x^{(i)}; \theta\right), y^{(i)}\right)$

Güncellemeyi uygula: $\theta \leftarrow \theta - \in \hat{g}$

Bu güncelleme yakınsama sağlanana kadar devam etmektedir.

Avantajları;

- Büyük veri setlerinde minimuma daha hızlı yakınsar.
- Hesaplama yönünden hızlıdır.

Dezavantajları;

- Minimum noktaya kararlı bir şekilde yakınsama yapamaz.
- Daha fazla iterasyona ihtiyaç duymaktadır.
- Uyarlanabilir değildir.

2.3.2. Momentum

SGD popüler bir algoritma olsa da yakınsama yavaş olabilmektedir. Özellikle yerel minimum çevresinde yaygın olan, yüzeyin bir boyutta diğerine göre çok daha dik eğrildiği alanlarda SGD sorun yaşamaktadır. Zaten gürültülü bir yakınsamaya sahip

olan SGD minimuma doğru tereddütlü ve yavaş yakınsar. Sutton (1986), SGD'nin uçurumların bulunduğu yerlerde aşırı derecede yavaş olduğunu ve önceki kazanımların yok olmasına sebep olabildiğini göstermiştir.

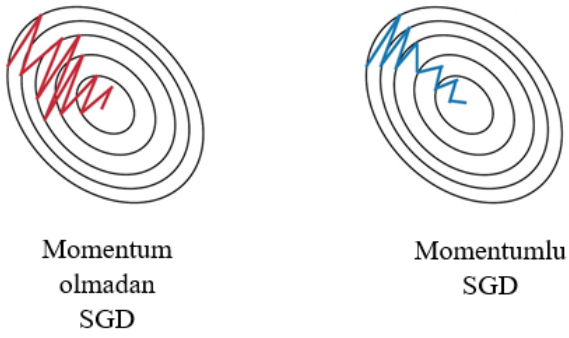
Momentum yöntemi (Polyak, 1964), yakınsamadaki gürültülü hareketi azaltmak ve eğimi yüksek olan yüzeylerde küçük ve tutarlı bir ilerleme sağlamak için tasarlanmıştır.

Momentum algoritması üstel olarak azalan geçmiş gradyanların ortalamasını biriktirir ve onların yönünde ilerlemeye devam eder. Başka bir ifadeyle önceki gradyanları da hesaba katarak onların bilgisinden de yararlanır ve böylece salınımı azaltır. Bu sayede daha kararlı yakınsama olur (Goodfellow, Bengio ve Courville; 2016).

Momentum algoritması hızı temsil eden bir v değişkenini kullanır. Bu değişken parametrelerin hangi yönde ve ne kadar hızlı ilerlediğini gösterir. Momentum ismi, parametre uzayındaki bir taneciği Newton'un hareket yasalarına göre ilerleten bir kuvvet gibi olmasından hareketle türetilmiştir. Qian (1999), momentum parametresinin Newton parçacıklarının kütlesine benzer olduğunu göstermiştir.

Uygulamada α momentum parametresi için yaygın kullanılan değerler 0.5, 0.9 ve 0.99 şeklindedir. Öğrenme oranında olduğu gibi α değeride zamanla değiştirilebilir ve genellikle küçük bir değer ile başlanıp zamanla artacak şekilde uygulanır. Ama momentum parametresini zamanla değiştirmek öğrenme oranı değiştirmek kadar önemli değildir.

Momentum algoritmasında parametrelerin güncellenmesine etki eden iki kuvvet vardır. Birincisi maliyet fonksiyonu yüzeyinde tepeden aşağı iten $-\nabla_\theta J_\theta$ (negatif gradyan) ile orantılı olan kuvvettir. Burada parametreler uzayda bir tanecik olarak düşünülürse, bu tanecik tepeden aşağı yuvarlanan bir top olarak düşünülebilir. Yüzeyin dik kısmından aşağı doğru inişe geçerse hız kazanır. İkinci kuvvet ise $-v(t)$ (t anındaki hız) ile orantılı olan kuvvettir. Bu kuvvet taneciğin zamanla enerjisinin düşmesine ve yerel minimuma yakınsamasına sebep olur. Başka bir ifadeyle tepeden aşağı doğru hız kazanan taneciği dizginleyip hedefe yöneltir (Goodfellow, Bengio ve Courville; 2016). Şekil 2.5.'de SGD ve momentum farkı gösterilmiştir.



Şekil 2.5. SGD ve momentum farkı (Alblwi, 2020)

Momentum algoritmasının güncellemesi aşağıdaki gibi yapılmaktadır (Goodfellow, Bengio ve Courville; 2016):

Algoritma2 Momentum algoritması kodu

Gerekli: ∈ öğrenme oranı, α momentum parametresi

Gerekli: θ ağırlık parametresinin ilk değeri, v hızın ilk değeri

$\{x^{(1)}, \ldots, x^{(m)}\}$ eğitim kümesinden m örneklik bir mini-yığına karşılık gelen $y^{(i)}$ hedefleriyle birlikte örnekle

Gradyan kestrimini hesapla: $g \leftarrow \frac{1}{m}\nabla_\theta \Sigma_i L\left(f\left(x^{(i)};\theta\right), y^{(i)}\right)$

Hızdaki güncellemeyi hesapla: $v \leftarrow \alpha v - \in g$

Güncellemeyi uygula: $\theta \leftarrow \theta + v$

Bu güncelleme yakınsama sağlanana kadar devam etmektedir.

Avantajları;

- Geçmiş gradyanları da kullandığı için daha kararlı yakınsama yapar.
- Gürültüyü azaltmaktadır.

Dezavantajları;

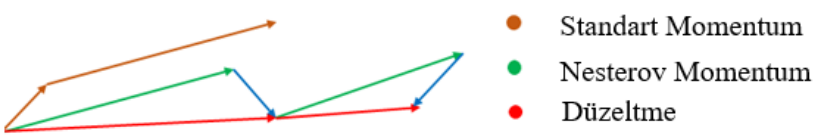
- Modele fazladan hiperparametre girmektedir.
- Algoritma sonradan hızlanmaktadır.
- Uyarlanabilir değildir.

2.3.3. Nesterov momentum

Sutskever vd. (2013), momentum algoritmasının Nesterov'un (1983) yönteminden ilham alan bir türünü sunmuştur ve standart momentum kullanarak eğitmenin zor olduğu güçlü modeller olan derin ve yineleyen sinir ağları (DSA ve YSA) için başlangıç parametresinin iyi ayarlandığında, momentum parametresininde iyi ayarlanıp yavaş bir şekilde azaltıldığında hem DSA hemde YSA'ları eğitebileceğini göstermişlerdir. Aynı zamanda dikkatlice ayarlanmış momentum yöntemleri, karmaşık ikinci dereceden yöntemlere ihtiyaç duymadan derin ve yineleyen ağ eğitim hedeflerindeki eğrilik sorunları ile başa çıkmak için yeterli olduğunu göstermişlerdir.

Nesterov momentum ile standart momentum yöntemi arasındaki fark gradyanın ne zaman değerlendirildiğiyle ilgilidir. Standart momentum yönteminde önce mevcut gradyanı hesaplar, sonra güncellenmiş birikmiş gradyan yönünde büyük bir sıçrama yapar. Nesterov momentumda ise önce başlangıçtaki birikmiş gradyan yönünde büyük bir sıçrama yapar, gradyanı ölçer ve sonra bir düzeltme yapar. Yani Nesterov momentumda gradyan, mevcut hız uygulandıktan sonra değerlendirilir. Bu ileriye dönük güncelleme çok hızlı gidilmesini engeller ve yanıt verme hızını arttırır (Ruder, 2016). Bengio vd. (2013), Nesterov momentumun standart momentum ile arasındaki bu önemli farkın bir dizi görevde önemli ölçüde geliştirilmiş YSA performansı ile sonuçlandığını göstermiştir.

Momentumdaki tanecik yokuş aşağı bir top olarak düşünülürse Nesterov momentumdaki tanecik nerede durması gerektiğini bilen bir top olarak düşünülebilir. Şekil 2.6.'da Standart momentum ve Nesterov momentum arasındaki fark gösterilmiştir.



Şekil 2.6. Momentum ve nesterov momentum farkı (Ruder, 2016)

Nesterov Momentum algoritmasının güncellemesi aşağıdaki gibi yapılmaktadır (Goodfellow, Bengio ve Courville; 2016):

Algoritma3 Nesterov Momentum algoritması kodu

Gerekli: ∈ öğrenme oranı, α momentum parametresi

Gerekli: θ ağırlık parametresinin ilk değeri, v hızın ilk değeri

$\{x^{(1)}, \ldots, x^{(m)}\}$ eğitim kümesinden m örneklik bir mini-yığına karşılık gelen $y^{(i)}$ hedefleriyle birlikte örnekle

Ara güncellemeyi uygula: $\tilde{\theta} \leftarrow \theta + \alpha v$

Gradyan kestrimini hesapla (ara noktadaki): $g \leftarrow \frac{1}{m} \nabla_{\tilde{\theta}} \Sigma_i L\left(f\left(x^{(i)}; \tilde{\theta}\right), y^{(i)}\right)$

Hızdaki güncellemeyi hesapla: $v \leftarrow \alpha v - \in g$

Güncellemeyi uygula: $\theta \leftarrow \theta + v$

Bu parametre güncelleme işlemi yakınsama sağlanana kadar devam etmektedir.

Avantajları;

- Kontrollü bir şekilde yakınsar.

Dezavantajları;

- Uyarlanabilir değildir.

2.3.4. AdaGrad

Öğrenme oranı belirlenmesi en zor hiperparametrelerden biridir. Öğrenme oranı modelin performansını önemli ölçüde etkiler. Öğrenme oranını uyarlamaya çalışan en eski yaklaşımlardan biri delta-bar-delta algoritmasıdır (Jacobs, 1988). Bu yaklaşımdaki mantık kısmi türevin işareti üzerine kurulmuştur. Kısmi türevin işareti değişmezse öğrenme oranının arttırılacağı, değişirse azaltılacağı savunulmuştur. Öğrenme oranını uyarlamaya çalışan bir çok algoritma geliştirilmiştir. AdaGrad algoritması (Duchi vd., 2011), bütün parametrelerin öğrenme oranlarını, gradyanın geçmiş karesel toplamının karekökü ile ters orantılı olacak şekilde ölçekleyerek uyarlar.

AdaGrad algoritması sık ortaya çıkan özelliklerle ilişkili parametreler için küçük güncellemeler (küçük öğrenme oranları), seyrek özelliklerle ilişkili parametreler için

büyük güncellemeler (büyük öğrenme oranları) yapar. Bundan dolayı seyrek verilerle uğraşmak için çok uygundur. Dean vd. (2012), AdaGrad algoritmasının SGD'nin sağlamlığını büyük ölçüde geliştirdiğini bulmuştur.

Sinir ağlarınının eğitiminde karesel gradyanların birikimi öğrenme oranında erken ve aşırı azalmalara sebep olabilmektedir. AdaGrad algoritmasının temel zayıflığı bu durumdur. Eklenen her terim pozitif olduğundan birikmiş toplam sürekli artar. Bu da öğrenme oranının küçülmesine ve sonunda da çok küçük hale gelmesine neden olur. Bundan dolayı algoritmanın yakınsama hızı aşırı yavaşlar. AdaGrad algoritması bazı modellerde iyi sonuç versede, bazı modellerde kötü sonuç vermektedir.

AdaGrad algoritmasının güncellemesi aşağıdaki gibi yapılmaktadır (Goodfellow, Bengio ve Courville; 2016):

Algoritma4 AdaGrad algoritması kodu

Gerekli: ∈ global öğrenme oranı

Gerekli: θ ağırlık parametresinin ilk değeri

Gerekli: δ küçük değerli sabiti (örneğin sayısal denge için 10^{-7} veya 10^{-8})

Gradyan birikim değişkeninin ilk atamasını yap: $r = 0$

$\{x^{(1)}, \dots, x^{(m)}\}$ eğitim kümesinden m örneklik bir mini-yığına karşılık gelen $y^{(i)}$ hedefleriyle birlikte örnekle

Gradyanı hesapla: $g \leftarrow \frac{1}{m} \nabla_\theta \Sigma_i L\left(f\left(x^{(i)}; \theta\right), y^{(i)}\right)$

Karesel gradyanı topla: $r \leftarrow r + g \odot g$

Güncellemeyi hesapla: $\Delta\theta = -\frac{\epsilon}{\delta + \sqrt{r}} \odot g$ (Bölüm ve karekök işlemi eleman bazlı uygulanır)

Güncellemeyi uygula: $\theta \leftarrow \theta + \Delta\theta$

Bu parametre güncelleme işlemi yakınsama sağlanana kadar devam etmektedir.

Avantajları;

- Seyrek özelliklerle ilişkili parametreler için büyük, sık ortaya çıkan özelliklerle ilişkili parametreler için daha küçük güncellemeler gerçekleştirir.

- Uyarlanabilirdir.

Dezavantajları;

- Öğrenme oranı aşırı küçülmektedir.

2.3.5. Adadelta

Adadelta algoritması (Zeiler, 2012), AdaGrad algoritmasının genişletilmiş halidir. Bu algoritma, AdaGrad algoritmasının iki dezavantajını çözmek için geliştirilmiştir:

1) Eğitim boyunca öğrenme oranlarının sürekli azalması

2) Manuel olarak seçilen global öğrenme oranına duyulan ihtiyaç

AdaGrad algoritmasında payda, eğitimin başlangıcından başlayarak her yinelemeden alınan kare gradyanları toplar. Her terim pozitif olduğu için, biriken bu toplam eğitim boyunca artmaya devam eder ve her boyuttaki öğrenme oranını etkili bir şekilde azaltır. Birçok yinelemeden sonra, bu öğrenme oranı son derece küçük olacaktır.

Adadelta algoritması, AdaGrad algoritmasının yaptığı tüm geçmiş gradyanların karelerini biriktirmek yerine, bu geçmiş gradyanları sınırlayarak son gradyanlara odaklanır.

Adadelta algoritmasında öğrenme oranını seçme zorunluluğu yoktur. Öğrenme oranı yerine, geçerli ağırlıklar ile güncellenen ağırlıklar arasındaki farkın karelerinin momentumlu toplamları kullanılmaktadır.

Adadelta algoritmasının güncellemesi aşağıdaki gibi yapılmaktadır (Goodfellow, Bengio ve Courville; 2016):

Algoritma5 Adadelta algoritması kodu

Gerekli: Azalma oranı ρ (genellikle 0.95), sayısal denge için sabit δ (genellikle 10^{-6}, küçük sayılarla bölünmeyi dengeler)

Gerekli: θ parametresi

Başlangıç biriktirme değişkenleri: $E[g^2]_0 = 0, \quad E[\Delta\theta^2]_0 = 0$

Gradyanı hesapla: $g \leftarrow \frac{1}{m}\nabla_\theta \Sigma_i L\left(f\left(x^{(i)};\theta\right), y^{(i)}\right)$

Gradyanı biriktir: : $E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho) g_t^2$

Güncellemeyi hesapla: $\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t$

Güncellemeleri biriktir: $E[\Delta\theta^2]_t = \rho E[\Delta\theta^2]_{t-1} + (1-\rho)\Delta\theta_t^2$

Güncellemeyi uygula: $\theta_{t+1} = \theta_t + \Delta\theta_t$

Bu parametre güncelleme işlemi yakınsama sağlanana kadar devam etmektedir.

Avantajları;

- Öğrenme oranının aşırı küçülmesini engellemektedir.
- Öğrenme oranını başlangıçta seçme zorunluluğu yoktur.
- Uyarlanabilirdir.

Dezavantajları;

- Üstel olarak azalan geçmiş gradyanları saklamaz.

2.3.6. RMSProp

AdaGrad algoritması dışbükey fonksiyonlara uygulandığında hızlıca yakınsayacak şekilde tasarlanmıştır. RMSProp algoritması (Hinton, 2012), AdaGrad algoritmasının gradyan birikimi yaklaşımını üstel ağırlıklı hareketli ortalama yaklaşımı ile değiştirerek dışbükey olmayan yüzeylerde daha iyi çalışmasını sağlar. RMSProp Geoffrey Hinton tarafından bulunan ve öğrenme oranını uyarlamaya çalışan bir yöntemdir.

AdaGrad algoritması öğrenme oranını gradyanın geçmiş karesel toplamına göre küçültür. Dolayısıyla dışbükey bir yapıya ulaşmadan öğrenme oranı aşırı küçülür. RMSProp algoritması ise üstel olarak azalan ortalama yaklaşımını kullanarak uzak geçmişteki noktaları eler. Böylece öğrenme oranının aşırı küçülmesinin önüne geçer.

RMSProp algoritması hareketli ortalama kullandığı için bu ortalamanın boyunu kontrol eden ve ρ ile ifade edilen yeni bir hiperparametre dahil eder. Günümüzde kullanılan önemli optimizasyon algoritmalarından biridir.

RMSProp algoritmasının güncellemesi aşağıdaki gibi yapılmaktadır (Goodfellow, Bengio ve Courville; 2016):

Algoritma6 RMSProp algoritması kodu

Gerekli: ∈ global öğrenme oranı, ρ azalma oranı

Gerekli: θ ağırlık parametresinin ilk değeri

Gerekli: δ küçük değerli sabiti (genellikle 10^{-6}, küçük sayılarla bölünmeyi dengeler)

Birikim değişkeninin ilk değer atamasını yap: $r = 0$

$\{x^{(1)}, \dots, x^{(m)}\}$ eğitim kümesinden m örneklik bir mini-yığına karşılık gelen $y^{(i)}$ hedefleriyle birlikte örnekle

Gradyanı hesapla: $g \leftarrow \frac{1}{m}\nabla_\theta \Sigma_i L\left(f\left(x^{(i)};\theta\right), y^{(i)}\right)$

Karesel gradyanı topla: $r \leftarrow \rho r + (1-\rho) g \odot g$

Parametre güncellemesini hesapla: $\Delta\theta = -\frac{\epsilon}{\sqrt{\delta + r}} \odot g$ ($\frac{1}{\sqrt{\delta+r}}$ eleman bazlı uygulanır)

Güncellemeyi uygula: $\theta \leftarrow \theta + \Delta\theta$

Bu parametre güncelleme işlemi yakınsama sağlanana kadar devam etmektedir.

RMSProp algoritmasının standart hali dışında Nesterov Momentum ile birleştirilmiş hali de vardır. Nesterov momentumlu RMSProp algoritmasının güncellemesi aşağıdaki gibi yapılmaktadır (Goodfellow, Bengio ve Courville; 2016):

Algoritma7 Nesterov Momentumlu RMSProp algoritması kodu

Gerekli: ∈ global öğrenme oranı, ρ azalma oranı, α momentum sabiti

Gerekli: θ parametresi ve v hızının ilk değeri

Birikim değişkeninin ilk değer atamasını yap: $r = 0$

$\{x^{(1)}, \dots, x^{(m)}\}$ eğitim kümesinden m örneklik bir mini-yığına karşılık gelen $y^{(i)}$ hedefleriyle birlikte örnekle

Ara güncellemeyi uygula: $\tilde{\theta} \leftarrow \theta + \alpha v$

Gradyanı hesapla: $g \leftarrow \frac{1}{m} \nabla_{\tilde{\theta}} \Sigma_i L\left(f\left(x^{(i)}; \tilde{\theta}\right), y^{(i)}\right)$

Gradyanı topla: $r \leftarrow \rho r + (1-\rho) g \odot g$

Hızdaki güncellemeyi hesapla: $v \leftarrow \alpha v - \frac{\epsilon}{\sqrt{r}} \odot g$ ($\frac{1}{\sqrt{r}}$ eleman bazlı uygulanır)

Güncellemeyi uygula: $\theta \leftarrow \theta + v$

Bu parametre güncelleme işlemi yakınsama sağlanana kadar devam etmektedir.

Avantajları;

- Öğrenme oranının aşırı küçülmesini engellemektedir.
- Uyarlanabilirdir.

Dezavantajları;

- Üstel olarak azalan geçmiş gradyanları saklamaz.

2.3.7. Adam

Adam algoritması (Kingma ve Ba, 2014), uyarlamalı öğrenme oranı algoritmasıdır. Bu algoritma RMSProp ve momentum algoritmalarının birleştirilmesiyle elde edilmiştir. Adam algoritmasında momentum gradyanın üstel ağırlıklı birinci dereceden momentinin bir kestirimi olarak doğrudan dahil edilir. Hem birinci hem de ikinci dereceden momentlerin kestrimlerine önyargı düzeltmesi içerir. Böylece momentlerin başlangıç değerlerini hesaba katar. Adam algoritmasının hiperparametre seçiminde iyi bir yaklaşım olduğu kabul görmektedir.

Adam algoritmasının güncellemesi aşağıdaki gibi yapılmaktadır (Goodfellow, Bengio ve Courville; 2016):

Algoritma8 Adam algoritması kodu

Gerekli: ∈ adım büyüklüğü (Önerilen değer: 0.001)

Gerekli: Moment kestrimi için üstel azalma oranları, ρ_1 ve ρ_2 ([0,1) aralığında). (Önerilen değerler sırasıyla: 0.9 ve 0.999)

Gerekli: Sayısal denge için kullanılan δ küçük değerli sabiti (Önerilen değer: 10^{-8})

Gerekli: θ ağırlık parametresinin ilk değerleri

1'inci ve 2'nci Moment değişkenlerinin ilk atamasını yap: $s = 0, r = 0$ zaman adımının ilk değerini ata: $t = 0$

$\{x^{(1)}, \dots, x^{(m)}\}$ eğitim kümesinden m örneklik bir mini-yığına karşılık gelen $y^{(i)}$ hedefleriyle birlikte örnekle

Gradyanı hesapla: $g \leftarrow \frac{1}{m} \nabla_\theta \Sigma_i L\left(f\left(x^{(i)}; \theta\right), y^{(i)}\right)$

$$t \leftarrow t + 1$$

Önyargılı ilk moment kestrimini güncelle: $s \leftarrow \rho_1 s + (1 - \rho_1) g$

Önyargılı ikinci moment kestrimini güncelle: $r \leftarrow \rho_2 r + (1 - \rho_2) g \odot g$

Birinci momentteki önyargıyı düzelt: $\hat{s} \leftarrow \frac{s}{1-\rho_1^t}$

İkinci momentteki önyargıyı düzelt: $\hat{r} \leftarrow \frac{r}{1-\rho_2^t}$

Güncellemeyi hesapla: $\Delta\theta = -\epsilon \frac{\hat{s}}{\sqrt{\hat{r}}+\delta}$ (işlemler eleman bazlı uygulanır)

Güncellemeyi uygula: $\theta \leftarrow \theta + \Delta\theta$

Bu parametre güncelleme işlemi yakınsama sağlanana kadar devam etmektedir.

Avantajları;

- Uygulanması kolaydır.
- Uyarlanabilirdir.
- Hesaplama açısından verimlidir.
- Az hafıza alanı gerektirir.

Dezavantajları;

- Hiperparametre sayısı fazladır.

2.3.8. Nadam

Nadam algoritması (Dozat, 2016), Adam algoritmasının bir uzantısıdır. Adam algoritması ile Nesterov momentumun birleştirilmesiyle elde edilmiştir. Nesterov algoritmasını Adam algoritmasına dahil etmek için momentum ifadesinde değişiklik yapılarak mevcut güncelleştirme kuralı elde edilir (Ruder, 2016).

Nadam gürültülü gradyanlar veya yüksek eğimli gradyanlar için kullanılır. Hiperparametre sayısı fazla olmasına rağmen iyi sonuçlar verebilmektedir.

Nadam algoritmasının güncellemesi aşağıdaki gibi yapılmaktadır (Goodfellow, Bengio ve Courville; 2016):

Algoritma9 Nadam algoritması kodu

Gerekli: $\in$ adım büyüklüğü (Önerilen değer: 0.001)

Gerekli: Moment kestrimi için üstel azalma oranları, ρ_1 ve ρ_2 ([0,1) aralığında).

Gerekli: Sayısal denge için kullanılan δ küçük değerli sabiti (Önerilen değer: 10^{-8})

Gerekli: θ ağırlık parametresinin ilk değerleri

1'inci ve 2'nci Moment değişkenlerinin ilk atamasını yap: $s = 0, r = 0$ zaman adımının ilk değerini ata: $t = 0$

$\{x^{(1)}, \dots, x^{(m)}\}$ eğitim kümesinden m örneklik bir mini-yığına karşılık gelen $y^{(i)}$ hedefleriyle birlikte örnekle

Gradyanı hesapla: $g \leftarrow \frac{1}{m} \nabla_\theta \Sigma_i L\left(f\left(x^{(i)}; \theta\right), y^{(i)}\right)$

$$t \leftarrow t + 1$$

Önyargılı ilk moment kestrimini güncelle: $s \leftarrow \rho_1 s + (1 - \rho_1) g$

Önyargılı ikinci moment kestrimini güncelle: $r \leftarrow \rho_2 r + (1 - \rho_2) g \odot g$

Birinci momentteki önyargıyı düzelt: $\hat{s} \leftarrow \frac{s}{1 - \rho_1^t}$

İkinci momentteki önyargıyı düzelt: $\hat{r} \leftarrow \frac{r}{1 - \rho_2^t}$

Nesterov uygulaması: $\bar{s} = \rho_1 \hat{s} + (1 - \rho_1) g$

Güncellemeyi hesapla: $\Delta\theta = -\epsilon \frac{\bar{s}}{\sqrt{\hat{r}}+\delta}$

Güncellemeyi uygula: $\theta \leftarrow \theta + \Delta\theta$

Bu parametre güncelleme işlemi yakınsama sağlanana kadar devam etmektedir.

Avantajları;

- Yüksek eğimli eğriler için kullanılabilmektedir.
- Uyarlanabilirdir.

Dezavantaj;

- Hiperparametre sayısı fazladır.

2.4. İkinci Dereceden Gradyan Tabanlı Algoritmalar

Buraya kadar 1. dereceden gradyan tabanlı algoritmalar incelenmiştir. Bu algoritmaların hangisinin kullanılacağı konusunda araştırmacılar arasında fikir birliği yoktur Bu bölümde 2. dereceden gradyan tabanlı algoritmalar incelenecektir. 2. dereceden yöntemler ikinci türevleri kullanır. Ikinci türevden gelecek bilgileri de optimizasyona katmaya çalışır.

2.4.1. Newton yöntemi

Newton yöntemi, en çok bilinen 2. dereceden algoritmadır. Yakınsama için ikinci dereceden Taylor serisi açılımını kullanır. Bu yöntemde daha büyük dereceden türevler gözardı edilir.

Newton yöntemi gradyanı H^{-1} (Hessian matrisinin tersi) ile yeniden ölçekleyerek doğrudan minimum noktaya ulaşır. Amaç fonksiyonu dışbükey ancak karesel değilse (daha büyük dereceden terimler içeriyorsa) döngüsel olarak aşağıdaki şekilde algoritma kullanılır (Goodfellow, Bengio ve Courville; 2016).

Algoritma10 Newton algoritması kodu

Gerekli: θ_0 parametresinin ilk değeri

Gerekli: m örnekli eğitim kümesi

Gradyanı hesapla: $g \leftarrow \frac{1}{m}\nabla_\theta \Sigma_i L\left(f\left(x^{(i)};\theta\right), y^{(i)}\right)$

Hessian matrisini hesapla: $H \leftarrow \frac{1}{m}\nabla_\theta^2 \Sigma_i L\left(f\left(x^{(i)};\theta\right), y^{(i)}\right)$

Hessian matrisinin tersini hesapla: H^{-1}

Güncellemeyi hesapla: $\Delta\theta = -H^{-1}g$

Güncellemeyi uygula: $\theta = \theta + \Delta\theta$

Bu parametre güncelleme işlemi yakınsama sağlanana kadar devam etmektedir.

Newton yöntemi, her adımda Hessian matrisinin tersini hesaplayarak fonksiyonun ikinci dereceden türevindeki bilgilerden de faydalanarak çok daha az adım atılabileceğini ve daha hızlı yakınsama sağlananabileceğini göstermektedir. Newton yönteminde öğrenme oranı kullanılmaz.

Hessian matrisi pozitif tanımlı kaldığı sürece Newton yöntemi kullanılabilir aksi takdirde Newton yöntemi kullanılmaz. Çünkü Hessian matrisi pozitif değilse bir eyer noktasının civarında olunduğu anlamına gelir. Eyer noktaları civarında Hessian matrisinin özdeğerlerinin tamamı pozitif değildir. Eyer noktaları Newton yöntemi için sorun oluşturmaktadır. Güncellemelerin yanlış yönde gitmesine sebep olmaktadır. Bu problemden kurtulmak için düzenlileştirme stratejileri geliştirilmiştir.

Newton yöntemindeki önemli zorluklarından birinin eyer noktaları olduğu bölüm 2.2.'de açıklanmıştı. Başka bir önemli zorluk ise yüksek boyutlu ağlardaki hesaplama yüküdür. Bu hesaplama yükü Newton yönteminin kullanımını sınırlandırmaktadır. Günümüzde küçük ağlarda bile parametre sayısı milyonları bulabilmektedir. Newton yöntemi için bu durum büyük bir sorundur. Bundan dolayı Newton yöntemi az parametre bulunan ağlarda kullanılabilmektedir (Goodfellow, Bengio ve Courville; 2016).

2.4.2. BFGS

BFGS algoritması ismini kaşiflerinden (Broyden, Fletcher,Goldfarb ve Shanno) almaktadır. BFGS algoritması Newton yönteminin bazı avantajlarını, hesaplama yükü olmadan sağlamaya çalışır. Newton yönteminde Hessian matrisinin tersinin (H^{-1}) hesaplanması temel zorluklardandır. Bu algoritmada izlenen yol, her adımda M_t tersinin yaklaşıklığının H^{-1} matrisine daha çok yaklaşmasıdır. M_t matrisi güncellendiğinde, inişin yönünü gösteren ρ_t belirlenir. Parametre güncellemesi;

$$\rho_t = M_t g_t \tag{2.7}$$

$$\theta_{t+1} = \theta_t + \epsilon^* \rho_t \tag{2.8}$$

şeklinde olur (Goodfellow, Bengio ve Courville; 2016).

BFGS algoritması, ikinci dereceden bilgiyle, oluşan yöndeki doğru üzerinde aramalarını yineler. BFGS algoritması, her aramayı iyileştirmek için daha az zaman harcamanın avantajına sahiptir. Ancak Hessian matrisinin tersini saklamak zorunda olduğu için büyük ağlarda kullanımı uygun değildir.

2.4.3. L-BFGS

L-BFGS algoritması (Limited Memory BFGS/Sınırlı Hafızalı BFGS) Quasi-Newton ailesinden bir algoritmadır. Bu yöntemde tüm Hessian matrisinin yaklaşıklığını (M) tutmaktan kaçınarak, BFGS yönteminin hafıza maliyetini azaltır. Dolayısıyla BFGS yönteminden daha az bellek kullanır. L-BFGS algoritması, aynı yöntemi, yaklaşıklığı adımlar arasında korumak yerine $M^{(t-1)}$ matrisinin birim matris olduğu varsayımıyla uygular (Goodfellow, Bengio ve Courville; 2016).

L-BFGS algoritması her adımda M matrisini güncellemek için bazı vektörleri saklar. Bu şekilde hesaplama maliyeti azaltılır. Bu algoritma büyük veri setleri ile iyi çalışır.

3. BULGULAR VE İRDELEME

Bu bölümde, 1. ve 2. dereceden algoritmaların performanslarını karşılaştırmak için birbirinden farklı gerçek hayat problemleri incelenmiştir. Bu problemler, görüntü tanıma ve sınıflandırma problemleridir. Algoritmalar farklı modellere uygulanarak başarımları ölçülmeye çalışılmıştır. Başarımlarını ölçmek için doğruluk ve F1 skoru gibi hata metrikleri kullanılmıştır.

3.1. Programlama Dili

Çalışmada kullanılacak programlama dili için matematiksel işlemlerde gösterdiği yüksek performans, sahip olduğu kütüphaneler ile hızlı, performanslı, etkili, kullanıcı dostu programlamaların yapılabilmesi ve en çok tercih edilen programlama dili olması gibi özelliklerinden dolayı Python programlama dili seçilmiştir.

Python, kolay bir şekilde okunabilme özelliği olan, nesne tabanlı bir programlama dilidir (Arslan, 2019).

Phyton programlama dili makine öğrenmesi ve makine öğrenmesinin alt alanı olan derin öğrenmede oldukça geniş bir kullanım alanına sahiptir.

3.2. Kütüphane Seçimi

Derin öğrenme kütüphaneleri, sinir ağı tasarlama, eğitme gibi işlemlerde oldukça kullanışlıdır. En çok kullanılan derin öğrenme kütüphaneleri TensorFlow (Google), CNTK (Microsoft), PyTorch (Facebook) ve MxNet (Amazon) sayılabilir. Bu kütüphanelerin çoğu ile uyumlu çalışan ve daha üst seviye kütüphane olan Keras kütüphanesi çok yaygın bir kullanıma sahiptir.

Çalışmada, esnek mimarisinden dolayı TensorFlow kütüphanesi ve TensorFlow ile uyumlu olarak çalışan Keras kütüphanesi kullanılmıştır.

3.3. Modeller

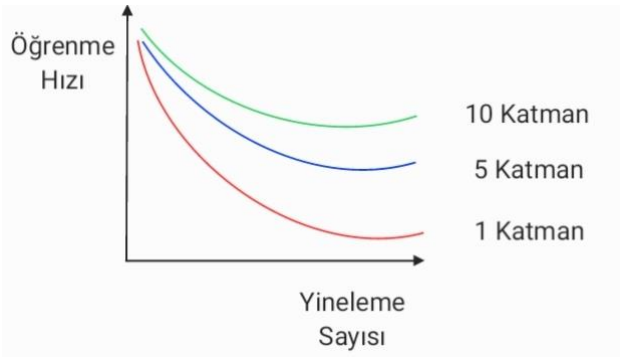
Çalışmada farklı mimarilere sahip ResNet50 ve VGG19 modelleri kullanılmıştır. Bu modeller ImageNet, Büyük Ölçekli Görüntü Tanıma Yarışmasında başarılı olmuşlardır.

3.3.1. ResNet50

Residual Networks'ün kısaltması olan ResNet, birçok bilgisayarla görme görevi için omurga olarak kullanılan klasik bir sinir ağıdır. Bu model, 2015 yılında ImageNet yarışmasının galibi olmuştur (Zhank vd., 2016). ResNet50, ImageNet veri kümesi üzerinde eğitilmiş 50 katmanlı bir ağdır. ImageNet, görüntü tanıma yarışmaları için oluşturulmuş 20 binden fazla kategoriye ait 14 milyondan fazla resmin bulunduğu bir görüntü veritabanıdır.

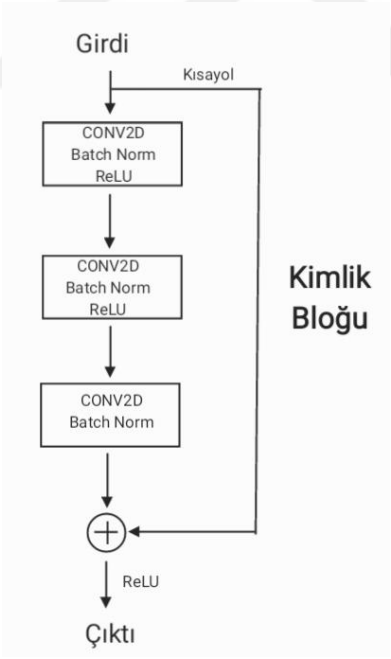
ResNet50, Evrişimli Sinir Ağlarının (ESA) geliştirilmiş bir versiyonudur. ResNet50 modeli ESA ağlarının performans düşümü (gradyan ölüm) problemini çözmeyi hedeflemektedir. Derin ağlar yakınsamaya başladığı zaman, öğrenme hızı bozulmaya başlamaktadır. Başka bir ifadeyle söylemek gerekirse eğitim hatası artmaktadır. Derin ağlarda beklenen durum katman sayısı arttıkça eğitim hatasının azalması yani doğruluk değerinin artmasıdır. Ancak ağ derinliği arttıkça, doğruluk (accuracy) doygunluğa ulaşır ama daha sonra hızlı bir düşüş eğilimi gösterir (Zhank vd., 2016).

Eğitim sırasında, öğrenme hızı eğitim ilerledikçe (yineleme sayısı arttıkça) çok hızlı bir şekilde sıfıra düşmektedir. ResNet50 bu problemi çözmek için tasarlanmıştır. Bu modelde katman sayısı arttıkça öğrenme hızı dengelenmektedir. Şekil 3.1.'de ResNet50 modelinde katman sayısı ile öğrenme hızı arasındaki ilişki gösterilmiştir. Burada yineleme sayısı arttıkça öğrenme hızının düşmesinin katman sayısı arttıkça daha az olduğu gözlemlenmiştir.



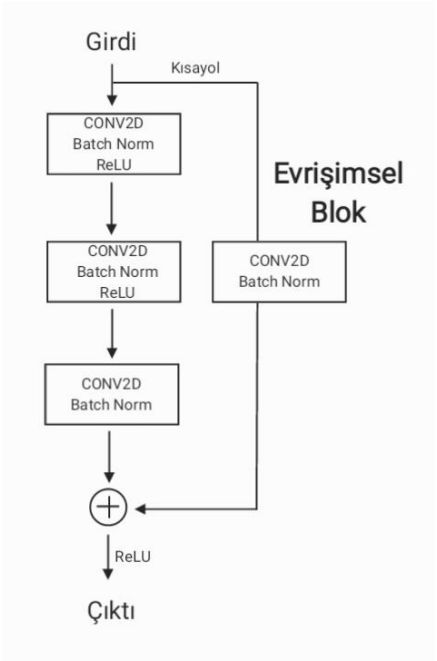
Şekil 3.1. Katman sayısı ile öğrenme hızı ilişkisi (Atienza, 2018)

ResNet bu problemi çözmek için katmanlar arasına kısayollar ekler. Gradyan bu sayede önceki katmanlara doğrudan geri yayılım sağlar. ResNet'te giriş/çıkış boyutlarının aynı veya farklı olmasına bağlı olarak iki ana blok türü kullanılır. Kimlik bloğu, ResNet'lerde kullanılan standart bloktur ve giriş aktivasyonunun çıkış aktivasyonu ile aynı boyuta sahip olduğu duruma karşılık gelir (Atienza, 2018). Şekil 3.2.'de Kimlik bloğu örneği gösterilmiştir.



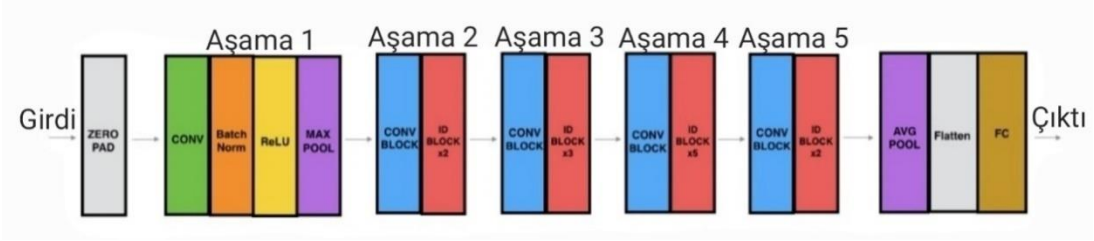
Şekil 3.2. Kimlik bloğu örneği (Zhank vd., 2016)

Evrişim bloğu diğer blok tipidir. Giriş ve çıkış boyutları eşleşmediğinde bu blok türü kullanılabilir. Kimlik bloğundan farkı, kısayol üzerinde bir CONV2D katmanının bulunmasıdır (Atienza, 2018). Şekil 3.3.'de evrişim bloğu örneği gösterilmiştir.



Şekil 3.3. Evrişim bloğu örneği (Zhank vd., 2016)

Kısayol üzerindeki CONV2D katmanı, girişi farklı bir boyuta yeniden boyutlandırmak için kullanılır. Böylece kısayol değerini ana yola geri eklemek için gereken son eklemede boyutlar eşleşir. ResNet50 modeli kimlik ve konvülasyon blokları eklenerek oluşturulmaktadır (Atienza, 2018). Şekil 3.4.'de ResNet50 modeli gösterilmiştir.



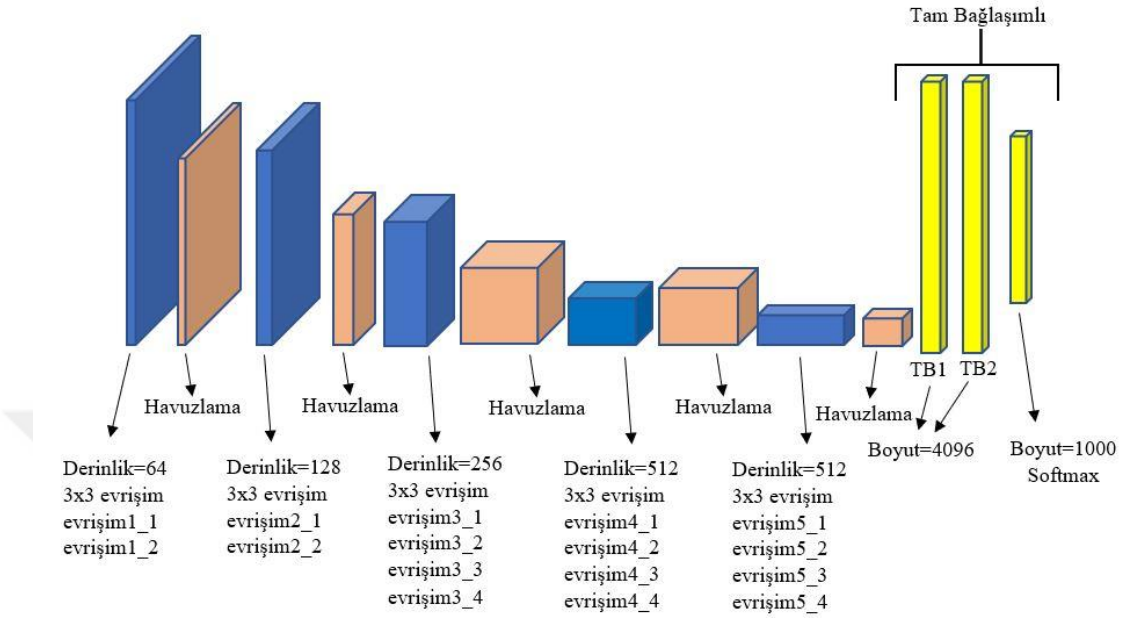
Şekil 3.4. ResNet50 modeli (Atienza, 2018)

3.3.2. VGG19

Görüntü sınıflandırmak için kullanılan evrişimsel bir sinir ağıdır. ImageNet veri tabanındaki bir milyondan fazla görüntü üzerinde eğitilmiştir. Önceden eğitilmiş ağ, görüntüleri klavye, fare, kurşun kalem ve birçok hayvan gibi 1000 nesne kategorisine ayırabilir.

Simonyan ve Zisserman görüntü sınıflandırmasında derinliğin önemli olduğunu göstermişlerdir (Simonyan, 2014).

VGG19, 19 katmandan (16 evrişimsel katmanı, 5 havuzlama (MaxPool) katmanı, 3 tam bağlaşımlı katman) oluşan derin öğrenme modelidir. Şekil 3.5.'de VGG19 modeli gösterilmiştir.



Şekil 3.5. VGG19 modeli (Kurt, 2018)

Evrişimsel katman; Girdi görüntülerinin piksellerini tarayan bir filtre uygulayarak özellik haritası oluşturur.

Havuzlama katmanı; Evrişim katmanını üretilen bilgiyi depolamak için küçültür.

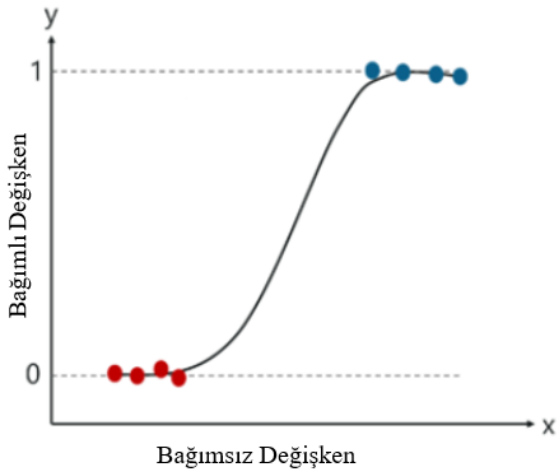
Tam bağlaşımlı giriş katmanı; çıktıları tek bir vektörde düzleştirir.

Tam bağlaşımlı katman; özellik analizi tarafından oluşturulan girdilere ağırlıkları uygular.

Tam bağlaşımlı çıktı katmanı; görüntü sınıfını belirlemek için son olasılıkları üretir.

3.3.3. Lojistik regresyon

Lojistik regresyon bir sınıflandırma modelidir. İkili sınıflandırmalarda (var/yok, hasta/sağlıklı vb.) sıkça kullanılmaktadır. Lojistik regresyonda aktivasyon fonksiyonu olarak sigmoid fonksiyonu kullanılmaktadır. Şekil 3.6.'da lojistik regresyon örneği verilmiştir.



Şekil 3.6. Lojistik regresyon örneği (Lateef, 2020)

Şekil 3.6.'da bağımsız değişken (x) girdi değerlerini, bağımlı değişken etiketli çıktı değerlerini ifade etmektedir.

Denklem (2.9)'da ağırlık değerleri ile girdi değerlerinin çarpımı ile oluşan z değeri verilmiştir. Bu değerin sigmoid fonksiyonu altındaki görüntüsü denklem (2.10)'da gösterilmiştir.

$$z = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n \tag{2.9}$$

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{2.10}$$

$\sigma(z) \geq 0.5$ ise $y = 1$ etiketli değer, $\sigma(z) < 0.5$ ise $y = 0$ etiketli değer çıktı olarak verilir.

3.4. Hiperparametreler

Hiperparametreler, model eğitim sürecini kontrol etmemize olanak sağlayan ayarlanabilir parametrelerdir.

3.4.1. Öğrenme oranı

Gradyan iniş algoritmalarında kullanılan bir katsayıdır. Yakınsamanın sağlanmasında kullanılmaktadır. Yakınsama sırasında atılacak adımın büyüklüğünü gösterir. Bölüm

2.1.'de de belirtildiği üzere öğrenme oranının çok büyük veya çok küçük seçilmesi yakınsamayı olumsuz etkileyebilmektedir.

3.4.2. Aktivasyon fonksiyonu

Yapay sinir ağlarına doğrusal olmayan gerçek dünya özelliklerini tanıtmak için aktivasyon fonksiyonları kullanılmaktadır. Ayrıca aktivasyon fonksiyonları nöronların çıktılarını belirli değerler arasında sıkıştırır. Çizelge 3.1.'de aktivasyon fonksiyonları gösterilmiştir.

Çizelge 3.1. Aktivasyon fonksiyonları (Kızrak, 2019)

İsim	Fonksiyon
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$
Tanh	$tanh(x) = 2\sigma(2x) - 1$
ReLU	$f(x) = max(0, x)$
Leaky ReLU	$f(x) = \begin{cases} ax, & x < 0 \\ x, & x \geq 0 \end{cases}$
Softmax	$f(x) = \frac{e^{z_j}}{\Sigma_k e^{z_k}}$

3.4.3. Epok sayısı, iterasyon sayısı ve paket boyutu

Epok, tüm veri setinin ileri ve geri yönde tüm ağdan geçmesidir. Paket boyutu, ileri ve geri yayılım için veri setinden alınan verinin miktarıdır. İterasyon, belirlenen paket boyutu kadarlık verinin ileri ve geri yönde ağdan geçmesidir. Paket boyutunun yüksek seçilmesi, daha fazla hafızaya ihtiyaç duyulmasına sebep olmaktadır. Paket boyutu yüksek olursa, her iterasyon için kullanılacak veri miktarı da yüksek olur. 5000 adet veriden oluşan bir veri seti için paket boyutu 500 seçilirse, bir epoğu tamamlamak için 10 iterasyon gereklidir.

Epok sayısının yüksek olması modelin doğruluğunu olumsuz da etkileyebilir. Hoffer vd., yaptığı araştırmada epok sayısının artmasının, doğruluğu bir noktaya kadar arttırdığını ve doğruluğun bu noktadan sonra daha fazla artmadığını gözlemlemiştir (Hoffer, 2017).

3.5. Hata Metrikleri

Modellerin başarısını ölçmek için kullanılan metriklerden bu bölümde bahsedilecektir.

3.5.1. Doğruluk, kesinlik, duyarlılık ve F1 skoru

Doğruluk (accuracy), bir modelin başarısını ölçmek için çok kullanılan ancak tek başına yeterli olmayan bir metriktir. Modelde tüm doğru tahminlerin tüm veri kümesine oranı ile hesaplanmaktadır. Eşit dağılmayan veri kümelerinde model doğruluğu tek başına yeterli değildir.

Doğruluk, kesinlik, duyarlılık ve F1 skoru, Karışıklık/Hata Matrisi (Confusion Matrix) yardımıyla hesaplanmaktadır. Şekil 3.7.'de hata matrisi gösterilmiştir.

		Tahmin Edilen Değer	
		Pozitif	Negatif
Gerçek Değer	Pozitif	TP (Doğru Pozitif)	FP (Yanlış Pozitif)
	Negatif	FN (Yanlış Negatif)	TN (Doğru Negatif)

Şekil 3.7. Hata matrisi örneği

Denklem (3.1)'de doğruluk metriğinin nasıl hesaplanacağı gösterilmiştir.

$$Doğruluk = \frac{TP+TN}{TP+FP+TN+FN} \tag{3.1}$$

Kesinlik (precision), modelde pozitif olarak tahmin edilen değerlerin gerçekten kaç adedinin pozitif olduğunu göstermektedir. Denklem (3.2)'de kesinlik metriğinin nasıl hesaplanacağı gösterilmiştir.

$$Kesinlik = \frac{TP}{TP+FP} \tag{3.2}$$

Duyarlılık (recall), pozitif olarak tahmin edilmesi gereken işlemlerin ne kadarının pozitif olarak tahmin edildiğini gösteren metriktir. Denklem (3.3)'de duyarlılık metriğinin nasıl hesaplanacağı gösterilmiştir.

$$Duyarlılık = \frac{TP}{TP+FN} \tag{3.3}$$

F1 skoru, kesinlik ve duyarlılığın harmonik ortalamasıdır. Eşit dağılmayan veri kümelerinde model seçimi yapılırken F1 skoruna bakılmaktadır. Denklem (3.4)'de F1 skoru metriğinin nasıl hesaplanacağı gösterilmiştir.

$$F1\ Skoru = 2.\frac{kesinlik.duyarlılık}{kesinlik+duyarlılık} \tag{3.4}$$

3.5.2. ROC ve AUC

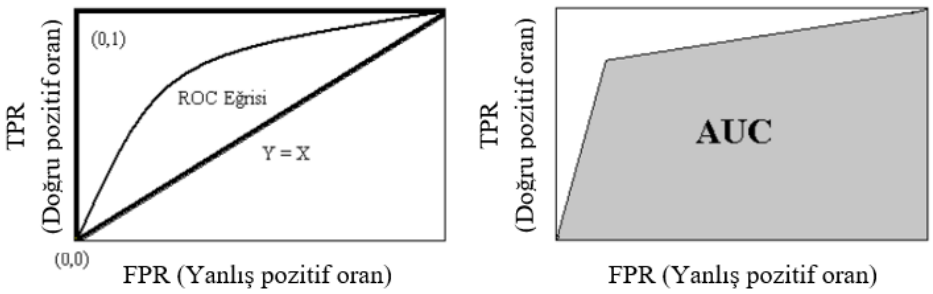
ROC eğrisi bir olasılık eğrisidir. Sınıflandırma problemleri için çok önemli bir başarı ölçme metriğidir. AUC değeri ise ROC eğrisinin altında kalan alandır. AUC değeri modelin sınıfları ne kadar başarılı ayırt edebildiğini gösterir. AUC değeri ne kadar yüksekse model sınıfları ayırt etmede o kadar iyidir.

ROC eğrisinde yatay eksende FPR (Yanlış pozitif oran) ve dikey eksende TPR (Doğru pozitif oran) bulunmaktadır. Denklem (3.5)'de TPR değerinin nasıl hesaplanacağı ve denklem (3.6)'da FPR değerinin nasıl hesaplanacağı gösterilmiştir.

$$TPR = \frac{TP}{TP+FN} \tag{3.5}$$

$$FPR = \frac{FP}{FP+TN} \tag{3.6}$$

Şekil 3.8.'de ROC eğrisi ve AUC değeri için örnekler verilmiştir.



Şekil 3.8. ROC ve AUC örneği (Tomak ve Bek, 2009)

Yukarıda açıklanan hata metrikleri, gerçek hayat problemleri için derin öğrenme yöntemleri ile üretilen çözümlerin değerlendirilmesinde kullanılmaktadır. Bu tez çalışmasında elde edilen sonuçların değerlendirilmesinde hata metriklerinden yararlanılacaktır.

3.6. Gerçek Hayat Problemi 1

İnme, beyin dokusu içine veya onu çevreleyen zarlar ve kemik arasına olan kanamayı ifade eder (Ergün, 2019).

İnme tedavisine, inme durumu sonrası acilen başlanmalıdır. Tedaviye erken başlanması durumunda beyin dokusunda oluşabilecek kalıcı hasarların önüne geçilebilmektedir. Tedaviye geç başlanılması kalıcı hasarların oluşmasına veya ölüme sebep olabilmektedir.

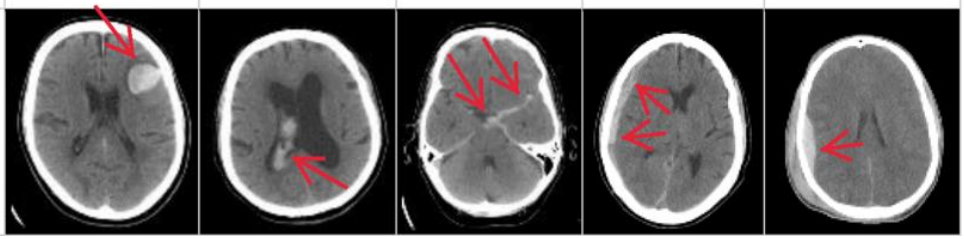
İnme tıbbi bir acil durumdur ve teşhis acil bir prosedür gerektirir. Bir hasta şiddetli baş ağrısı veya bilinç kaybı gibi akut nörolojik semptomlar gösterdiğinde, uzman doktorlar hastanın kafatasının tıbbi görüntülerini inceleyerek kanamanın varlığını, yerini ve tipini belirlemeye çalışırlar. Bu işlem karmaşık ve çoğu durumda zaman alıcıdır.

Bu problemin çözümü için derin öğrenme algoritması olan ESA kullanılarak, görüntü tanımada en çok tercih edilen ResNet50 ve VGG19 modelleri üzerinde çalışılacaktır. Bu modeller üzerinde 1. dereceden gradyan iniş algoritmalarının karşılaştırması yapılacaktır. Bu sayede en uygun model ve optimizasyon algoritması tespit edilmeye çalışılacaktır.

Çalışmada Kuzey Amerika Radyoloji Derneği (RSNA) tarafından sağlanan zengin bir görüntü veri setini kullanılmıştır.

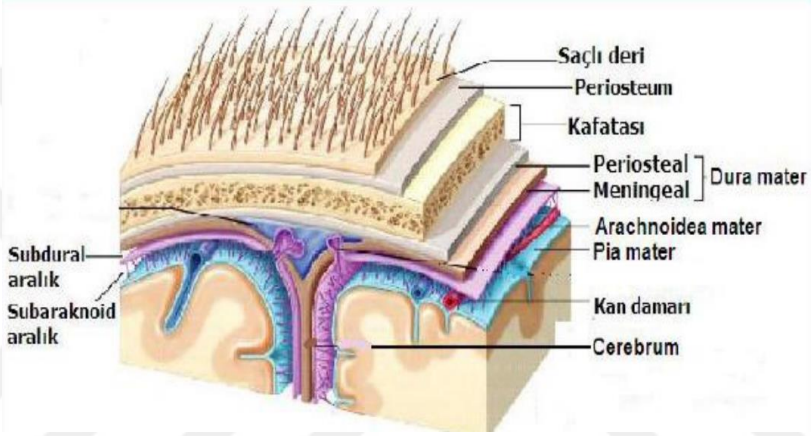
3.6.1. Veri seti

Kuzey Amerika Radyoloji Derneği (RSNA) tarafından sağlanan veri setinde kanamanın tipine göre beş sınıfa (intraparankimal, intraventriküler, subaraknoid, subdural ve epidural) ait görüntüler ve kanamanın olmadığı görüntüler bulunmaktadır (Flanders vd., 2020). Veri setinde toplam 874.035 adet görüntü bulunmaktadır. Şekil 3.9.'da inme tipleri ve oluştukları bölgeler gösterilmiştir.

	Intraparenchymal	Intraventricular	Subarachnoid	Subdural	Epidural
Yeri	Beynin içinde	Ventrikül içinde	Pia mater ve araknoid arasında	Dura mater ve araknoid arasında	Kafatası ile dura mater arasında
Görüntü					

Şekil 3.9. İnme tipleri (Wu, 2019)

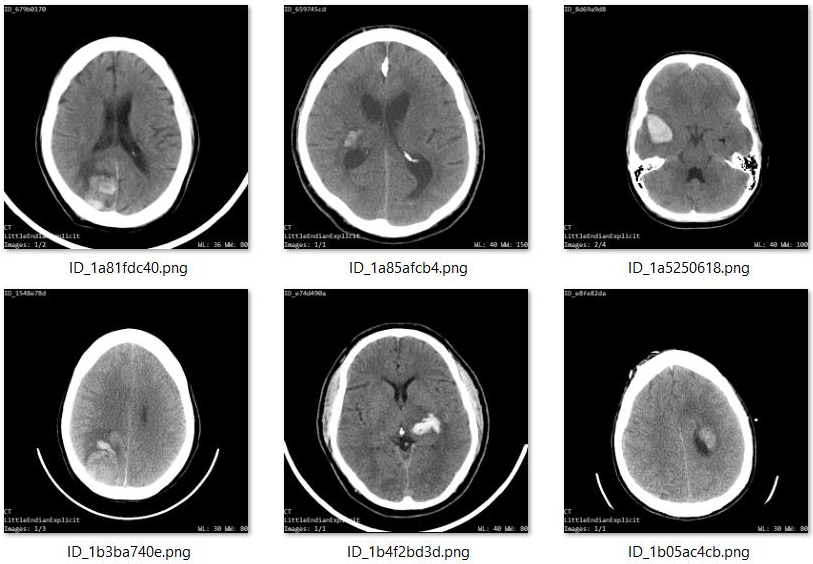
Şekil 3.10.'da beyin omurilik zarlarının yerleri gösterilmiştir.



Şekil 3.10. Beyin omurilik zarları (Birge,2016)

3.6.1.1. İntraparankimal kanama

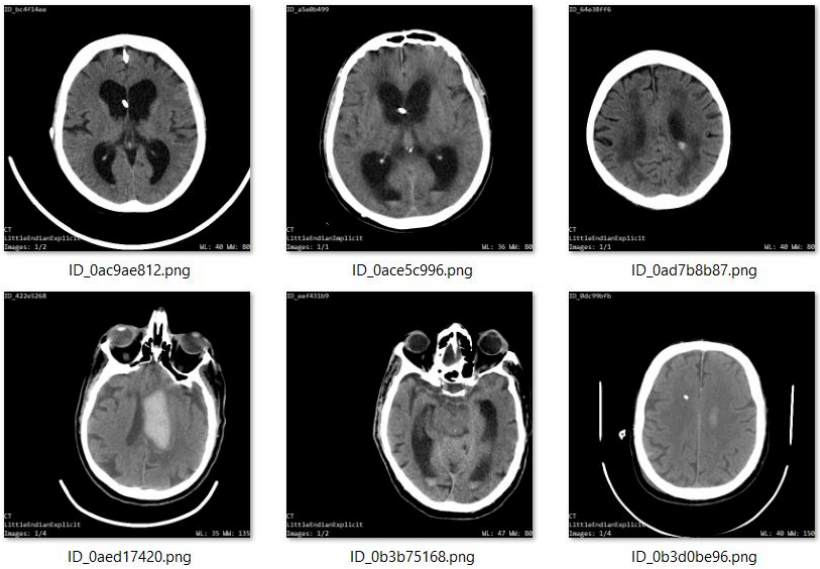
Tamamen beynin içinde oluşan kanama tipidir (Flanders vd., 2020). Yüksek tansiyon, travma, tümör gibi etkenler sonucu oluşabilmektedir. Aniden başlayan baş ağrısı, mide bulantısı ve kusma belirtileri ile kendini gösterebilir. Şekil 3.11.'de veri setinden intraparankimal kanamaya ait örnek görüntüler verilmiştir.



Şekil 3.11. İntraparankimal kanama örnekleri

3.6.1.2. İntraventriküler kanama

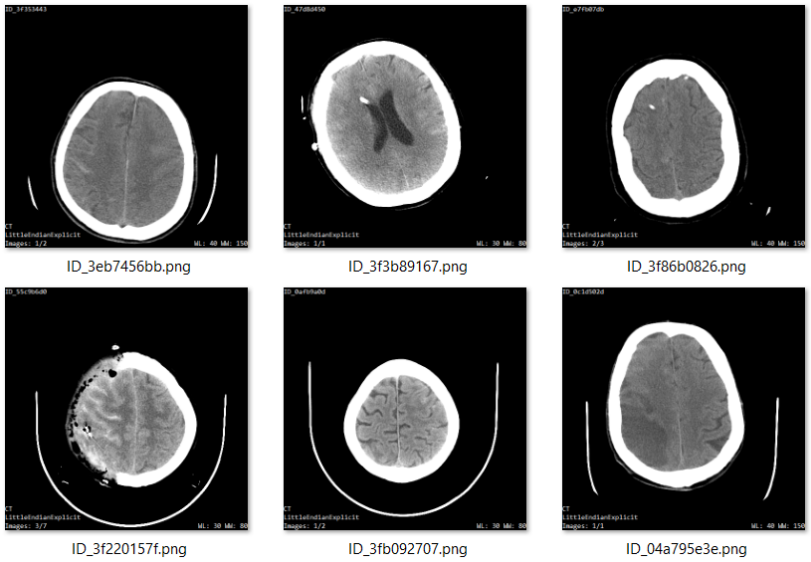
Ventriküller içine sızan kanama tipidir (Flanders vd., 2020). İntraparankimal kanama ve subaraknoid kanama ile ilişkili olabilmektedir. Aniden başlayan baş ağrısı, mide bulantısı ve kusma belirtileri ile kendini gösterebilir. Şekil 3.12.'de veri setinden intraventriküler kanamaya ait örnek görüntüler verilmiştir.



Şekil 3.12. İntraventriküler kanama örnekleri

3.6.1.3. Subaraknoid kanama

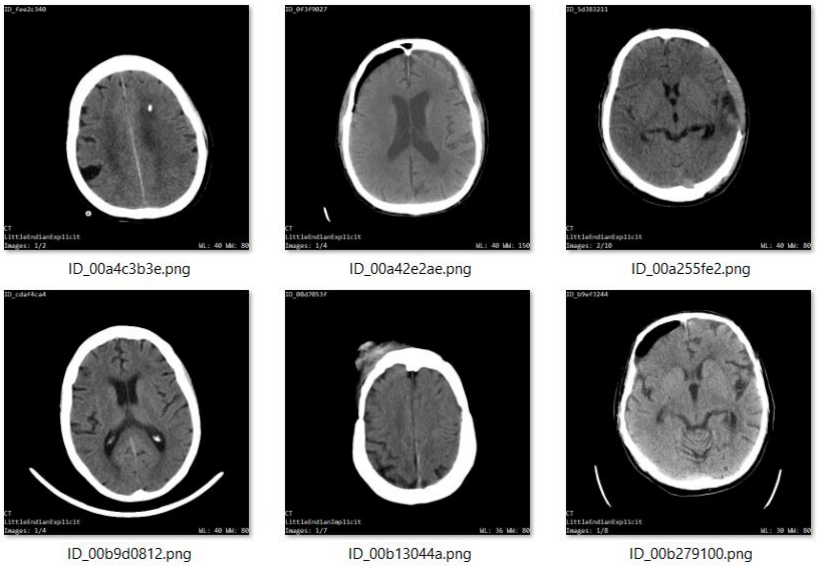
Araknoid ile pia mater arasında oluşan kanama tipidir (Flanders vd., 2020). Anevrizma yırtılması, arteriovenöz malformasyonlar ve travma gibi etkenler sonucu oluşabilmektedir. Çok şiddetli baş ağrısı ile kendini gösterebilmektedir. Şekil 3.13.'de veri setinden subaraknoid kanamaya ait örnek görüntüler verilmiştir.



Şekil 3.13. Subaraknoid kanama örnekleri

3.6.1.4. Subdural kanama

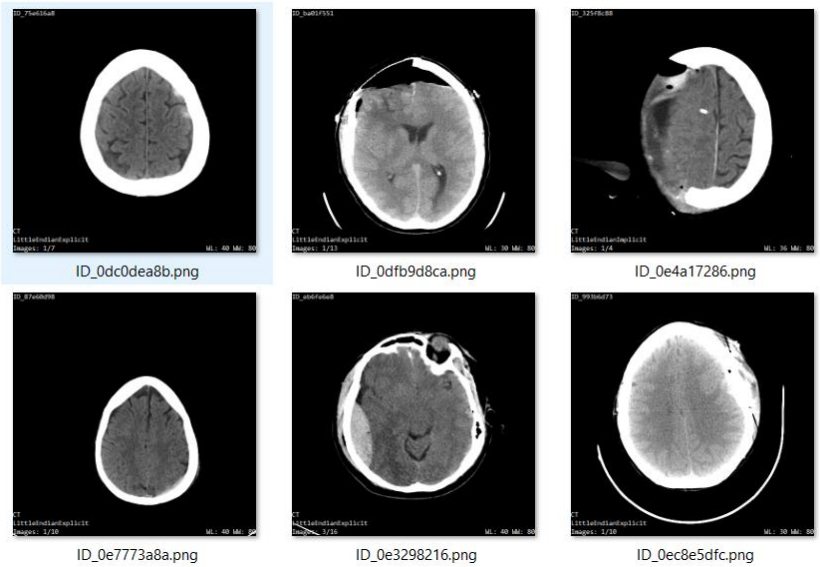
Araknoid ile dura mater arasında oluşan kanama tipidir (Flanders vd., 2020). Travma sonucu ve sinsi bir şekilde oluşabilmektedir. Zamanla kötüleşen baş ağrısıyla kendini gösterebilir. Şekil 3.14.'de veri setinden subdural kanamaya ait örnek görüntüler verilmiştir.



Şekil 3.14. Subdural kanama örnekleri

3.6.1.5. Epidural kanama

Kafatası ile dura mater arasında oluşan kanama tipidir (Flanders vd., 2020). Ameliyat sonrası gelişen travmalar sonucu oluşabilmektedir. Kafa travması ve zihinsel durumda bozukluk şeklinde kendini gösterebilir. Şekil 3.15.'de veri setinden epidural kanamaya ait örnek görüntüler verilmiştir.



Şekil 3.15. Epidural kanama örnekleri

3.6.2. Eğitim ve test süreci

Eğitim ve test sürecine başlarken ilk olarak eğitim ve test aşamasında kullanılacak veriler rastgele olarak belirlenmiştir. İki ayrı veri seti rassallık dikkate alınarak oluşturulmuştur. Veriler detaylı bir şekilde incelenerek bozuk veri kontrolü yapılmıştır. Eğitim ve test aşamasını olumsuz yönde etkileyecek bozuk görüntüler oluşturulan veri setlerinden çıkarılmıştır.

İlk kullanılacak veri seti kanamalı ve kanamasız olarak iki sınıf şeklinde oluşturulmuştur. Bu veri seti kanamalı sınıf olarak inme tiplerinden subdural tipi seçilmiştir. Çok sık karşılaşılması bu seçimde etkili olmuştur. Çizelge 3.2.'de birinci veri setine ait bilgiler verilmiştir.

Çizelge 3.2. İnme birinci veri seti

Veri tipi	Veri sayısı	Toplam veri sayısı
Subdural(Kanamalı)	8.200	16.400
Normal(Kanamasız)	8.200	

İkinci olarak kullanılacak veri seti kanamalı ve kanamasız olarak iki sınıf şeklinde oluşturulmuştur. Bu veri seti kanamalı sınıf inme tiplerinden 5 alt sınıf ile oluşturulmuştur. Çizelge 3.3.'de ikinci veri setine ait bilgiler verilmiştir.

Çizelge 3.3. İnme ikinci veri seti

Veri tipi	Veri sayısı	Toplam veri sayısı
İntraparankimal(Kanamalı)	1.650	16.400
İntraventriküler(Kanamalı)	1.650	
Subaraknoid(Kanamalı)	1.650	
Subdural(Kanamalı)	1.650	
Epidural(Kanamalı)	1.600	
Normal(Kanamasız)	8.200	

Veri setlerinde sınıflar dengeli seçilmeye çalışılmıştır. Veri setlerini oluşturduktan sonra hiperparametre seçimi gerçekleştirilmiştir. Bu seçim için ESA üzerinde hem ResNet50 hem de VGG19 modeli üzerinde çalıştırmalar yaparak en uygun hiperparametre değerleri manuel olarak elde edilmiştir. Elde edilen hiperparametreler 1. dereceden gradyan iniş optimizasyon algoritmalarının modeller üzerindeki performansları karşılaştırılacağı için sabit tutulmuştur. Tüm çalışmada veri seti %80 eğitim ve %20 test olacak şekilde kullanılmıştır. Çizelge 3.4.'de modellerde kullanılan hiperparametre seçimleri verilmiştir.

Çizelge 3.4. Hiperparametre seçimi

Hiperparametreler	Açıklama
Paket boyutu	128
Epok	30
Öğrenme oranı	0.001
Optimizasyon algoritmaları	Sgd, momentum,nesterov,adagrad, adadelta, rmsprop, adam, nadam

Çalışmada, 1. Dereceden gradyan iniş optimizasyon algoritmaları farklı mimarilere sahip iki model üzerinde kullanılmıştır. Bu modeller ResNet50 ve VGG19 modelleridir. Eğitim ve test sürecinde her bir algoritma her bir model üzerinde her defasında farklı eğitim ve test setleri oluşturarak (%80-%20 oranını bozmadan) rassallık bakımından 5 defa kullanılmıştır. Elde edilen sonuçların ortalaması alınmıştır.

3.6.3. Birinci veri seti için bulgular

Birinci veri setinde inme tiplerinden subdural tipi kullanılmıştır. Bu veri seti ilk olarak ResNet50 modeli üzerinde 1. dereceden gradyan iniş optimizasyon algoritmalarıyla test edilmiştir. Çizelge 3.5.'de birinci veri setine ait ResNet50 modelinin sonuçları verilmiştir.

Çizelge 3.5. İnme birinci veri setinin ResNet50 sonuçları

Algoritmalar	Metrikler		
	Doğruluk	*AUC*	*F1 skoru*
Adam	77,83	86,02	77,45
AdaGrad	76,36	84,84	76,22
Nadam	74,69	82,43	73,78
Adadelta	72,74	80,70	72,54
Nesterov	67,43	69,30	63,82
SGD	66,81	69,39	63,66
RMSProp	65,90	69,12	61,27
Momentum	63,79	66,23	58,00

Veri seti ikinci olarak VGG19 modeli üzerinde 1. dereceden gradyan iniş optimizasyon algoritmalarıyla test edilmiştir. Çizelge 3.6.'da birinci veri setine ait VGG19 modelinin sonuçları verilmiştir. Her iki model için de Adam algoritmasının en iyi sonuçları ürettiği gözlemlenmiştir.

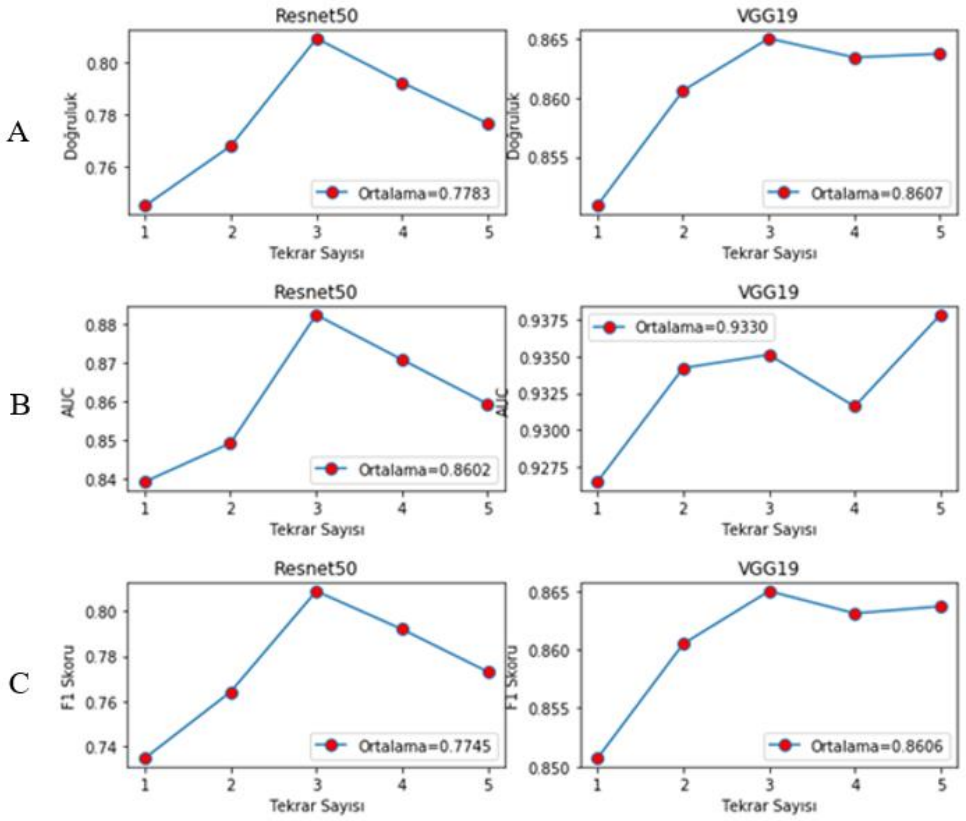
Çizelge 3.6. İnme birinci veri setinin VGG19 sonuçları

Algoritmalar	Metrikler		
	Doğruluk	*AUC*	*F1 skoru*
Adam	86,07	93,30	86,06
RMSProp	81,82	89,57	81,58
Nadam	81,50	89,77	80,85
AdaGrad	80,84	89,13	80,81
Momentum	78,57	87,76	77,59

Çizelge 3.6. (devam)

Algoritmalar	Metrikler		
	Doğruluk	*AUC*	*F1 skoru*
SGD	74,03	82,92	72,91
Nesterov	74,46	83,89	72,37
Adadelta	72,37	80,31	72,20

Şekil 3.16.'da Adam optimizasyon algoritmasının ResNet50 ve VGG19 modellerindeki doğruluk metriği (A), AUC değeri (B) ve F1 skorunun (C) 5 farklı uygulama için gerçekleşen sonuçları verilmiştir.



Şekil 3.16. Adam algoritmasının birinci veri seti için sonuçları

3.6.4. İkinci veri seti için bulgular

İkinci veri setinde ikili sınıflandırma (kanamalı-kanamasız) yapmak için inme tiplerinin tümü kullanılmıştır. Birinci veri setinde sadece subdural inme tipi kanamalı olarak etiketlenmişti. Bu veri setinde inme tiplerinin tamamı kanamalı olarak etiketlenmiştir. Bu veri seti ilk olarak ResNet50 modeli üzerinde 1. dereceden gradyan iniş optimizasyon algoritmalarıyla test edilmiştir. Çizelge 3.7.'de ikinci veri setine ait ResNet50 modelinin sonuçları verilmiştir.

Çizelge 3.7. İnme ikinci veri setinin ResNet50 sonuçları

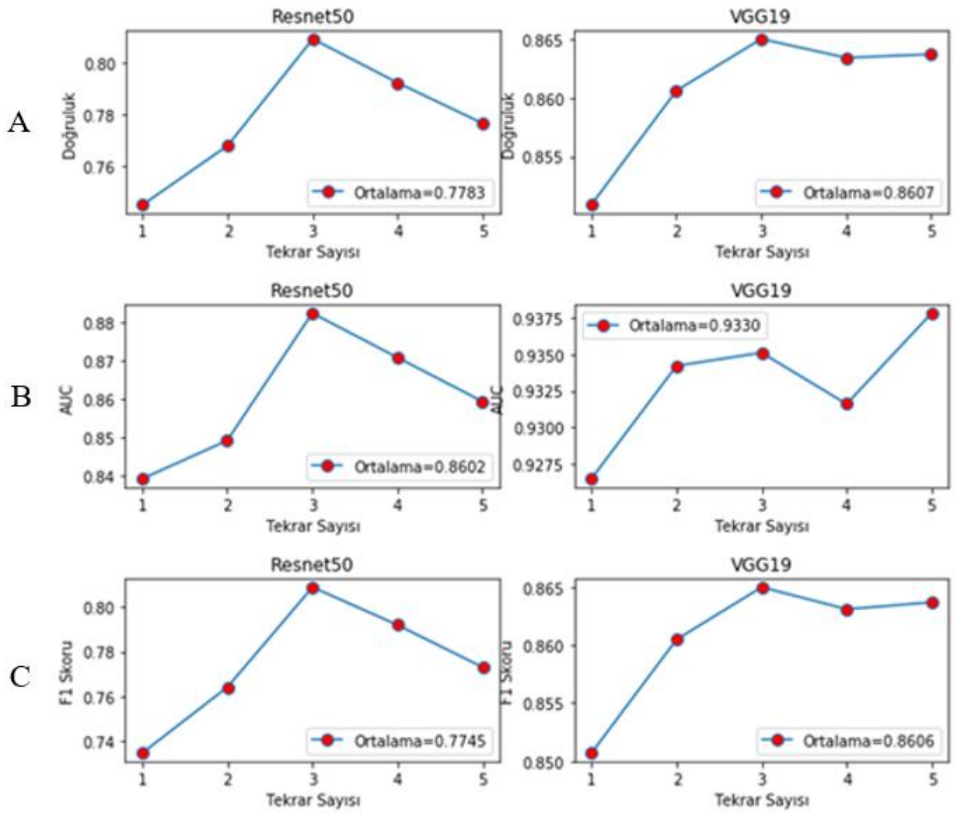
Algoritmalar	Metrikler		
	Doğruluk	*AUC*	*F1 skoru*
AdaGrad	74,51	83,09	74,46
Adam	74,44	82,55	74,26
Nadam	72,95	80,02	72,00
Nesterov	71,75	74,54	70,56
Momentum	71,23	75,51	70,22
Adadelta	70,02	77,80	69,96
RMSProp	66,55	73,35	67,45
SGD	63,94	66,14	59,44

Veri seti ikinci olarak VGG19 modeli üzerinde 1. dereceden gradyan iniş optimizasyon algoritmalarıyla test edilmiştir. Çizelge 3.8.'de ikinci veri setine ait VGG19 modelinin sonuçları verilmiştir. Karşılaştırmalar sonucunda Resnet50 modeli için Adagrad algoritmasının, VGG19 modeli için de Nadam algoritmasının en iyi sonuçları ürettiği ve Adam algoritmasının da her iki model için en iyi sonuç üreten ikinci algoritma olduğu gözlemlenmiştir.

Çizelge 3.8. İnme ikinci veri setinin VGG19 sonuçları

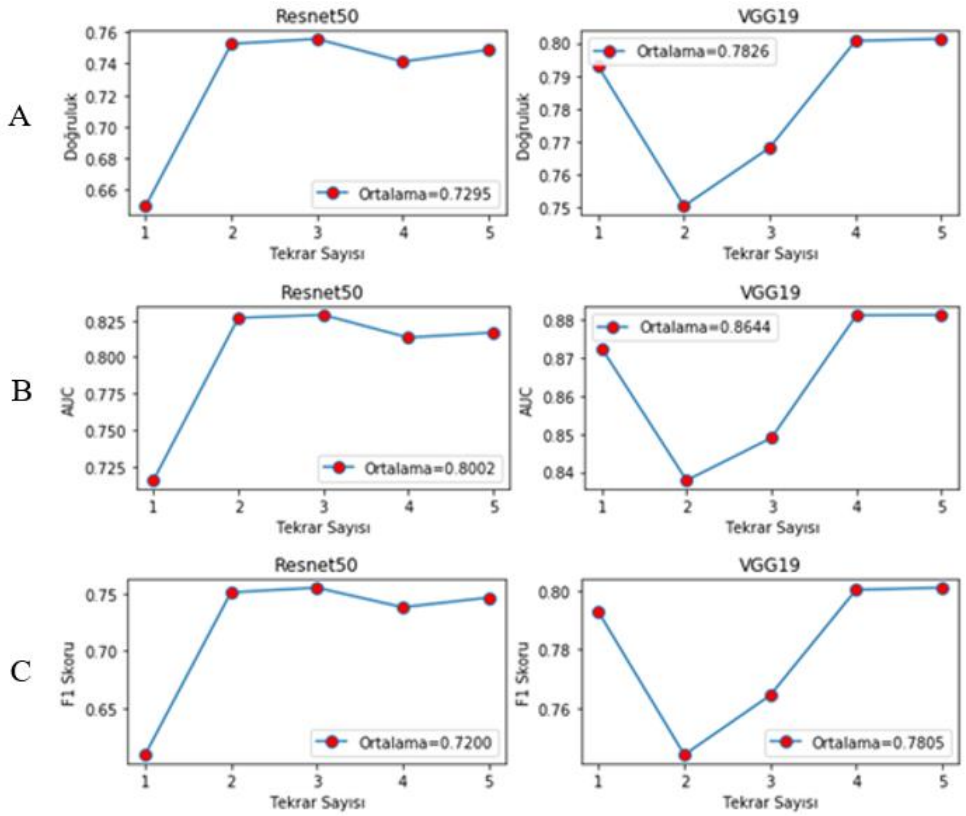
Algoritmalar	Metrikler		
	Doğruluk	*AUC*	*F1 skoru*
Nadam	78,26	86,44	78,05
Adam	78,10	85,97	77,80
AdaGrad	77,19	85,47	77,16
RMSProp	77,00	84,57	76,50
Momentum	74,77	83,29	73,90
Nesterov	71,49	80,28	68,95
Adadelta	70,57	78,23	70,49
SGD	65,69	74,99	61,34

Şekil 3.17.'de AdaGrad optimizasyon algoritmasının ResNet50 ve VGG19 modellerindeki doğruluk metriği (A), AUC değeri (B) ve F1 skorunun (C) 5 farklı uygulama için gerçekleşen sonuçları verilmiştir.



Şekil 3.17. AdaGrad algoritmasının ikinci veri seti için sonuçları

Şekil 3.18.'de Nadam optimizasyon algoritmasının ResNet50 ve VGG19 modellerindeki doğruluk metriği (A), AUC değeri (B) ve F1 skorunun (C) 5 farklı uygulama için gerçekleşen sonuçları verilmiştir.



Şekil 3.18. Nadam algoritmasının ikinci veri seti için sonuçları

Sonuç olarak, birinci veri seti için her iki modelde de Adam algoritması tüm metrikler için en iyi sonuçları üretmiştir. ResNet50 modelinde AdaGrad algoritmasının Adam algoritmasından sonra en iyi sonuçları ürettiği görülmüştür. RMSProp algoritmasının ResNet50 modelinde iyi sonuçlar üretmediği ancak VGG19 modelinde başarılı sonuçlar ürettiği görülmüştür. Algoritmaların genel olarak VGG19 modelinde performanslarının daha yüksek olduğu gözlemlenmiştir.

İkinci veri setinde ResNet50 modelinde AdaGrad algoritması tüm metriklerde en iyi sonuçları üretmiştir. VGG19 modelinde ise Nadam algoritması tüm metriklerde en iyi sonuçları üretmiştir. Adam algoritmasının her iki modelde de tüm metriklerde en iyi sonuçları üreten ikinci algoritma olduğu görülmüştür. İkinci veri setinde kanama tipine ait sınıf sayısının artmasıyla genel olarak algoritmaların performanslarında düşüş olduğu gözlemlenmiştir. İkinci veri setinde de algoritmalar VGG19 modelinde daha iyi performans göstermişlerdir.

3.7. Gerçek Hayat Problemi 2

İlk olarak Çin'in Wuhan kentinde 2019 yılının Aralık ayında görülen yeni koronavirüs, solunum yolu enfeksiyonuna sebep olan ve insandan insana geçebilen bir virüstür. Dünya Sağlık Örgütü virüsün neden olduğu hastalığı tanımlamak için COVID-19 terimini kullanmaktadır. 11 Mart 2020 tarihinde COVID-19 pandemi, yani küresel salgın hastalık olarak ilan edilmiştir.

COVID-19 hangi yaş aralıklarını daha çok etkiliyor, insan vücudunda nasıl değişiklikler yapıyor, kan değerlerine etkisi var mı, bağışıklık sistemine ait hücrelerde ne gibi durumlar oluyor vb. birçok soruya günümüzde cevap aranmaya devam edilmektedir. Kan değerleri, yaş, bağışıklık sistemine ait bilgiler gibi birçok özellik arasındaki ilişki incelenerek bir kişinin COVID-19 olup olmadığı tahmin edilebilir mi? sorusunun cevabı 1. ve 2. dereceden optimizasyon algoritmaları ile ele alınacaktır. Verilen bilgiler ile COVID-19/COVID-19 değil sınıflandırması yapılmaya çalışılacaktır.

3.7.1. Veri seti

Hasta ve hasta olmayan kişilere ait yaş, kan değerleri, kan hücrelerine ait bilgiler, bağışıklık sistemine ait bilgilerin bulunduğu etiketli veri seti kullanılmıştır. 5594 kişiden oluşan veri setinde 558 kişinin COVID-19 durumu pozitif olarak etiketlenmiştir. Bu bilgiler arasındaki korelasyona bakıp sınıflandırma yapılacaktır. Veri seti dengeli ve dengesiz dağılacak şekilde iki farklı şekilde kullanılacaktır. Şekil 3.19.'de veri setinden örnek bir kesit gösterilmiştir (Kaggle, 2020).

A	B	C	D	E	F	G	H	I	J	K	L
Hasta ID	Yaş	Kovid Sonucu	Hasta yoğun bakıma alındı	Hematokrit	Hemoglobin	Trombositler	Ortalama Trombosit Hacmi	Kırmızı kan hücreleri	Lenfositler	Ortalama korpüsküler hemoglobin konsantrasyonu	Lökositler
126e9dd13932f68	17	negatif	0	0,236515447	-0,022340268	-0,51741302	0,01067657	0,102004185	0,318365753	-0,950790346	-0,094610348
8bb9d64f0215244	1	negatif	0	-1,571682215	-0,774212003	1,429667473	-1,67222178	-0,850035012	-0,005738043	3,331070662	0,364550471
6c9d3323975b082	9	negatif	0	-0,747693062	-0,586243868	-0,429480344	-0,213710725	-1,36131525	-1,114513755	0,542882383	-0,88492316
d3ea751f3db9de9	11	negatif	0	0,991838217	0,792187631	0,072992042	-0,550289512	0,542763114	0,045436252	-0,452899486	-0,211487651
2c2eae16c12a18a	9	negatif	0	0,190738127	-0,147652119	-0,668154716	1,020415187	-0,12719065	0,002791033	-1,249524236	-1,132591724
fc41531ca4faf1e	13	negatif	0	1,0147264	0,854843855	-0,178244159	0,796028912	0,489871651	-0,730706871	-0,353319019	-0,075130761
891d0f6449ff3d7	14	negatif	0	0,740064204	0,854843855	0,361913651	-0,550289512	0,436981052	-0,227493063	0,64246285	0,105750747
ebdd7c67fcb21b4	9	negatif	0	-0,679027498	-0,711555779	0,952318728	-0,886869431	-0,321123987	-0,875700533	-0,253742367	-0,286623031
296adf066a9bf03	8	negatif	0	0,236515447	0,040315956	0,072992042	0,571642697	0,066743508	1,239502549	-0,652056575	-0,545422614
0b1e43c5e3fb1d1	17	negatif	0	0,808729768	1,042811871	-0,278738618	1,581381202	0,701436579	-0,261609375	1,040773273	4,61877E-06
c0aa8a22eaccb63	14	negatif	0	-0,106813163	-0,335620195	1,429667473	-0,213710725	-0,60321027	0,32689482	-0,950790346	0,726313293
826f654dbf80245	16	negatif	0	1,426721454	1,230779409	0,600588024	-1,111255646	1,265607476	0,506004632	-0,353319019	-0,022257695

Şekil 3.19. COVID-19 veri setinden kesit

Çizelge 3.9.'da dengesiz dağılıma sahip etiketli verilerin sayıları verilmiştir.

Çizelge 3.9. Dağınık dağılımlı COVID-19 veri seti

Veri tipi	Veri sayısı	Toplam veri sayısı
Negatif	5.036	5.594
Pozitif	558	

Çizelge 3.10.'da dengeli dağılıma sahip etiketli verilerin sayıları verilmiştir.

Çizelge 3.10. Dengeli dağılımlı COVID-19 veri seti

Veri tipi	Veri sayısı	Toplam veri sayısı
Negatif	550	1100
Pozitif	550	

3.7.2. Eğitim ve test süreci

Eğitim ve test sürecinde kullanılacak veriler rastgele olarak belirlenmiştir. Veriler incelenmiş ve bozuk veriler temizlenmiştir. Eğitim aşamasında veri özellikleri arasında korelasyona bakılmıştır. Gerçek hayat problemleri doğrusal olmadığından sınıflandırma modellerinden biri olan lojistik regresyon modeli kullanılmıştır. Lojistik regresyon modeli üzerinde 1. dereceden gradyan iniş algoritmaları ve 2. dereceden algoritmalar olan Newton ve BFLGS algoritmalarının performansları hem dengeli hem de dengesiz dağılıma sahip veri setlerinde incelenmiştir. Tüm çalışmada veri seti %70 eğitim ve %30 test olacak şekilde kullanılmıştır.

3.7.3. Bulgular

Veri seti lojistik regresyon modeli üzerinde eğitildikten sonra test edilmiştir. Çizelge 3.11.'de dengesiz dağılıma sahip test sonuçlarına ait metrik değerleri verilmiştir.

Çizelge 3.11. Dengesiz dağılımlı COVID-19 veri setinin lojistik regresyon sonuçları

Algoritmalar	Metrikler		
	Doğruluk	*AUC*	*F1 skoru*
Newton	91,14	53,69	87,60
L-BFGS	90,52	50,00	86,02
Nesterov	91,14	61,66	13,80
SGD	91,32	61,43	15,52
Momentum	91,23	60,83	16,81
RMSProp	90,88	61,37	17,60
Nadam	91,14	60,61	18,03
Adam	90,70	62,15	18,60
AdaGrad	90,52	49,24	1,83
Adadelta	89,19	54,56	18,66

Çizelge 3.12.'de dengeli dağılıma sahip test sonuçlarına ait metrik değerleri verilmiştir.

Çizelge 3.12. Dengeli dağılımlı COVID-19 veri setinin lojistik regresyon sonuçları

Algoritmalar	Metrikler		
	Doğruluk	*AUC*	*F1 skoru*
Newton	51,21	52,26	44,69
L-BFGS	50,91	52,15	41,20
Nesterov	57,27	62,57	67,61
SGD	51,52	61,79	68,00
Momentum	50,00	56,49	67,98
RMSProp	57,58	62,82	68,00
Nadam	51,21	63,84	66,09
Adam	58,18	61,72	68,01
AdaGrad	44,55	46,17	67,61
Adadelta	43,94	44,31	68,00

2. dereceden algoritmaların dengesiz dağılıma sahip veri setinde başarılı sonuçlar ürettiği görülmüştür. 1. dereceden algoritmaların birbirine yakın değerler ürettiği gözlemlenmiştir. Dengesiz dağılımlı veri setinde 1. dereceden algoritmaların F1 skorları çok düşük gelmiştir. 1. dereceden algoritmaların doğruluk değerlerinin yüksek, F1 skorlarının düşük gelmesinin nedeni aynı etikete sahip (covid negatif) çok sayıda veri olmasıdır. Diğer etiketli verinin (covid pozitif) sayıca az olması modelin sürekli aynı etiketli veriye yönelmesine sebep olmuştur. Bu yüzden doğruluk yüksek çıkmıştır.

Dengeli dağılıma sahip veri setinde genel olarak tüm algoritmaların doğruluk değeri düşmüştür. Bunun yanında 1. dereceden algoritmaların F1 skorları artmıştır. Verilerin dengeli dağılımının F1 skorunu arttırdığı gözlemlenmiştir. Veri setleri dengeli dağılıma sahip olduğunda ve veri sayıları azaldığında doğruluk değerinin düştüğü gözlemlenmiştir.

3.8. Gerçek Hayat Problemi 3

Diyabet (şeker hastalığı), günümüzde çok yaygın bir hastalıktır. Bu hastalık ölümcül bir çok hastalığın oluşumunda rol almaktadır. Diyabet, insülin eksikliği yada insülinin kullanımındaki sorunlar nedeniyle organizmanın karbonhidrat, yağ ve proteinlerden yeterince yararlanamadığı, sürekli tıbbi bakım gerektiren, kronik bir metabolizma hastalığıdır (Coşansu, 2015).

Ölümcül sonuçlara yol açabilen bu hastalığın erkenden teşhis edilmesi önem arzetmektedir. Kan değerleri, genetik faktörler, yaş gibi kişiye özgü bilgiler arasında korelasyon kurularak bir kişinin diyabet hastası olup olmadığı tahmin edilebilir mi? 1. ve 2. dereceden optimizasyon algoritmaları ile bu durum değerlendirilecektir. Verilen bilgiler ile diyabet/diyabet değil sınıflandırması yapılmaya çalışılacaktır.

3.8.1. Veri seti

Hasta ve hasta olmayan kişilere ait kan değerleri, genetik faktörler, yaş gibi bilgilerin bulunduğu etiketli veri seti kullanılmıştır. Veri setinde en az 21 yaşında olan Pima soyundan 768 kadına ait bilgiler bulunmaktadır. 768 kadından 500 tanesi diyabet

hastası olarak etiketlidir. Bu bilgiler arasındaki korelasyona bakıp sınıflandırma yapılacaktır. Veri seti, dengeli ve dengesiz dağılacak şekilde iki farklı şekilde kullanılacaktır. Sınıflandırmanın yanlı olup olmadığını gözlemlemek için bu şekilde kullanılacaktır. Dengeli veri seti rastsal olarak etiketli sınıflar eşit sayıda olacak şekilde oluşturulmuştur. Şekil 3.20.'de veri setinden örnek bir kesit gösterilmiştir (Kaggle, 2016).

Gebelik Sayısı	Glikoz	Kan Basıncı	Deri Kalınlığı	İnsülin	Beden Kitle Endeksi	Genetik Diyabet İhtimali	Yaş	Diyabet Olup Olmadığı
6	148	72	35	0	33,6	0,627	50	1
1	85	66	29	0	26,6	0,351	31	0
8	183	64	0	0	23,3	0,672	32	1
1	89	66	23	94	28,1	0,167	21	0
0	137	40	35	168	43,1	2,288	33	1
5	116	74	0	0	25,6	0,201	30	0
3	78	50	32	88	31	0,248	26	1
10	115	0	0	0	35,3	0,134	29	0
2	197	70	45	543	30,5	0,158	53	1
8	125	96	0	0	0	0,232	54	1
4	110	92	0	0	37,6	0,191	30	0
10	168	74	0	0	38	0,537	34	1
10	139	80	0	0	27,1	1,441	57	0
1	189	60	23	846	30,1	0,398	59	1
5	166	72	19	175	25,8	0,587	51	1
7	100	0	0	0	30	0,484	32	1

Şekil 3.20. Diyabet veri setinden kesit

Çizelge 3.13.'da dengesiz dağılıma sahip veri setindeki etiketli verilerin sayıları verilmiştir.

Çizelge 3.13. Dengesiz dağılımlı diyabet veri seti

Veri tipi	Veri sayısı	Toplam veri sayısı
Negatif	500	768
Pozitif	268	

Çizelge 3.14.'de dengeli dağılıma sahip veri setindeki etiketli verilerin sayıları verilmiştir.

Çizelge 3.14. Dengeli dağılımlı diyabet veri seti

Veri tipi	Veri sayısı	Toplam veri sayısı
Negatif	260	520
Pozitif	260	

3.8.2. Eğitim ve test süreci

Eğitim ve test sürecinde kullanılacak veriler rastgele olarak belirlenmiştir. Veriler incelenmiş ve bozuk veriler temizlenmiştir. Eğitim aşamasında veri özellikleri arasında korelasyona bakılmıştır. Sınıflandırma modellerinden biri olan lojistik regresyon kullanılmıştır. Lojistik regresyon modeli üzerinde 1. dereceden gradyan iniş algoritmaları ve 2. dereceden algoritmalar olan Newton ve BFLGS algoritmalarının performansları hem dengeli hemde dengesiz dağılıma sahip veri setlerinde incelenmiştir. Tüm çalışmada veri seti %70 eğitim ve %30 test olacak şekilde kullanılmıştır.

3.8.3. Bulgular

Veri seti lojistik regresyon modeli üzerinde eğitildikten sonra test edilmiştir. Çizelge 3.15.'de dengesiz dağılıma sahip veri setinin test sonuçlarına ait metrik değerleri verilmiştir.

Çizelge 3.15. Dengesiz dağılımlı diyabet veri setinin lojistik regresyon sonuçları

Algoritmalar	Metrikler		
	Doğruluk	*AUC*	*F1 skoru*
Newton	78,36	74,03	77,51
L-BFGS	78,36	74,03	77,51
RMSProp	69,26	71,23	42,28
Nadam	70,13	73,36	51,75
Adam	71,43	70,49	50,00
Momentum	65,37	60,00	34,43
SGD	60,17	47,85	02,13
AdaGrad	63,64	61,43	50,60
Nesterov	65,80	69,95	54,86
Adadelta	64,07	51,18	04,59

Çizelge 3.16.'da dengeli dağılıma sahip veri setinin test sonuçlarına ait metrik değerleri verilmiştir.

Çizelge 3.16. Dengeli dağılımlı diyabet veri setinin lojistik regresyon sonuçları

Algoritmalar	Metrikler		
	Doğruluk	*AUC*	*F1 skoru*
Newton	78,21	78,28	78,22
L-BFGS	76,92	76,99	76,94
RMSProp	70,51	75,66	68,91
Nadam	77,56	80,72	68,91
Adam	76,92	81,43	68,91
Momentum	61,54	62,84	61,04
SGD	53,85	51,62	68,94
AdaGrad	62,18	65,08	64,95
Nesterov	52,56	50,74	69,20
Adadelta	43,59	41,76	57,67

Dengesiz dağılımlı veri setinde 2. dereceden algoritmaların performanslarının daha iyi olduğu görülmektedir. 1. dereceden algoritmalardan Adam, doğruluk metriğinde performans bakımından diğer 1. dereceden algoritmalardan daha iyi sonuç üretmiştir. 1. dereceden algoritmalardan Nadam, AUC metriğinde, Nesterov ise F1 skorunda diğer 1. dereceden algoritmalardan daha iyi sonuç üretmişlerdir.

Dengeli dağılımlı veri setinde 2. dereceden algoritmaların başarımlarını korudukları görülmektedir. Doğruluk ve F1 skorunda en iyi sonuçları Newton algoritması vermiştir. AUC metriğinde ise Adam algoritması en iyi sonucu üretmiştir.

Dengesiz dağılımlı veri setinde 1. dereceden algoritmaların genel olarak F1 skorlarının düşük olduğu görülmektedir. F1 skoruna bakılmasının nedeni dengeli dağılmamış veri setlerinde yanlış model seçimi yapmamaktır (Öğündür, 2019). 1. dereceden algoritmalar dengesiz dağılımlı veri setinde etiket sayısı fazla olan sınıfa yöneldiği için F1 skorunun düşük geldiği görülmektedir.

4. TARTIŞMA VE SONUÇLAR

Bu çalışmada makine öğrenmesinde önemli yere sahip olan gradyan tabanlı optimizasyon algoritmaları ele alınmıştır. Bu algoritmaların matematiksel yapıları verilmiş, sonrasında gerçek hayat problemleri üzerinde başarımları gözlemlenmiştir. Ruder (2016), 1. dereceden gradyan tabanlı algoritmaları karşılaştırmaya çalışmıştır. Yıldız (2021), yaptığı çalışmada anemi ve anemi ile ilişkili hastalıklara tanı koyma sürecini ele alan gerçek hayat problemini Destek Vektör Makineleri, Karar Ağaçları, Naive Bayes ve Yapay Sinir Ağları yöntemleriyle analiz etmiş ve bu yöntemleri karşılaştırmıştır. Balcı (2021), yaptığı çalışmada Türkiye'de faaliyet gösteren bazı bankalara ait verileri lineer regresyon yönteminde kullanarak bankaların özkaynak karlılığını Stokastik Gradyan İniş, Adam ve RMSProp algoritmalarıyla tahmin etmeye çalışmıştır. Daha sonra bu algoritmaları karşılaştırmıştır. Yapılan çalışmalarda farklı makine öğrenmesi yöntemleri kullanılmıştır. Ancak optimizasyon algoritmalarının karşılaştırılması yeteri kadar yapılmamıştır. 1. dereceden algoritmaların bazılarına çalışmalarda yer verilmiştir. 2. dereceden algoritmalar incelenmemiştir. Literatürdeki bu boşluğu doldurmak için bu çalışmada hem 1. dereceden hem de 2. dereceden algoritmalar kullanılmıştır. 1. dereceden algoritmalar ResNet50, VGG19 ve lojistik regresyon modellerinde kullanılmıştır. 2. dereceden algoritmalar ise sadece lojistik regresyon modelinde kullanılmıştır. Bölüm 2.2.'de bahsedilen zorluklardan dolayı (eyer noktalarının fazlalığı vb.) 2. dereceden algoritmalar görüntü, ses gibi yüksek boyutlu sınıflandırılma problemlerinde kullanılmamaktadır. Bu nedenle beyin bilgisayarlı tomografi görüntülerinden inme tespitinin yapıldığı problem için 2. dereceden algoritmalarla karşılaştırma yapılamamıştır. 1. dereceden algoritmalarla 2. dereceden algoritmaların başarımları sayısal veriler içeren COVID-19 ve diyabet veri setleri üzerinden karşılaştırılmıştır.

Gerçek hayat problemlerinden ilk olarak beyin bilgisayarlı tomografi görüntülerinden inme tespiti problemi ele alınmıştır. Bu problemin çözümünde 1. dereceden algoritmaların ResNet50 ve VGG19 modelleri ile ürettiği sonuçlara göre başarımları metrikler üzerinden incelenmiştir. Bu problemde, iki veri seti kullanılmıştır. Birinci

veri seti normal (kanamasız) ve subdural inme tipine (kanamalı) ait beyin görüntülerinden oluşturulmuştur. İkinci veri seti normal (kanamasız) ve 5 farklı inme tipine (kanamalı) ait beyin görüntülerinden oluşturulmuştur. Bu veri setleri ile ikili sınıflandırma (kanamalı-kanamasız) yapılmıştır. Birinci veri seti için Adam algoritması, ikinci veri seti için AdaGrad algoritması tüm metrik türlerinde en iyi sonuçları üretmiştir. Genel olarak algoritmaların VGG19 modelinde daha iyi sonuçlar ürettiği görülmüştür.

İkinci olarak ele alınan sayısal verilere dayalı COVID-19 probleminde 2. dereceden algoritmaların dengesiz dağılıma sahip veri setlerinde başarılı sonuçlar verdiği gözlemlenmiştir. Dengesiz dağılıma sahip veri setlerinde 1. dereceden algoritmaların F1 skorlarının düşük gelebildiği, bu durumu düzeltmek için veri setini dengeli dağıtmak gerektiği görülmüştür. F1 skorunu düzeltmek için verileri dengelerken, doğruluk değeri veri kaybından dolayı düşebilmektedir. Verileri dengelerken veri sayısını da göz önünde bulundurmak gerekmektedir.

Diyabet probleminde ise hem dengeli hem de dengesiz dağılımda 2. dereceden algoritmaların performanslarının iyi olduğu görülmüştür. 1. dereceden algoritmalardan Adam, dengesiz dağılımda doğruluk metriğinde, diğer 1. dereceden algoritmalardan daha iyi sonuç üretmiştir. Dengeli dağılımda ise doğruluk ve F1 skorunda Newton algoritması en iyi sonuçlara sahip olmuştur.

Sonuç olarak, ele alınan problemler özelinde 1. dereceden algoritmalarda genel olarak Adam algoritmasının iyi sonuçlar ürettiği görülmüştür. Sınıf sayısının artması algoritmalarının performansını düşürebilmektedir. Dengesiz dağılımda 1. dereceden algoritmaların F1 skoru düşük olabilmektedir. Veri setini dengeli hale getirerek bu durum düzeltilebilmektedir. 2. dereceden algoritmalar sayısal verilerde iyi sonuçlar vermektedir. Dengesiz dağılımlı veri setlerinde de başarılı olabilmektedirler. Algoritmaların başarımı seçilen modele, veri setine, veri sayısına, probleme ve hiperparametrelere göre değişebilmektedir. 2. dereceden algoritmaların yapısal zorluklarından dolayı yüksek boyutlu veri setlerine uygulanması yönünde literatürde az sayıda çalışma olmasından dolayı bu konunun ileri ki çalışmalarda ele alınabileceği değerlendirilmiştir.

KAYNAKLAR

A. Cauchy. Methodes generales pour la resolution des syst‘emes dequations simultanees. C.R. Acad. Sci. Par., 25:536–538, 1847.

Alblwi, A. (2020). *Improving the Adaptive Moment Estimation Optimization Methods for Modern Machine Learning*. University of Delaware.

Alpaydın, E. (2010). *Introduction to machine learning* (2nd. Ed.). The MIT Press.

Arslan, İ. (2019). *Python ile Veri Bilimi*. Pusula.

Atienza, G. (2018). Residual Networks. GitHub. https://github.com/gemaatienza/

Balcı, T. (2021) *Yapay sinir ağları optimizasyon algoritmalarının banka özkaynak karlılığı tahmini üzerinde karşılaştırmalı performans analizi,* M.Sc. Thesis, Başkent University, Ankara, 78s.

Bengio, Y., Boulanger-Lewandowski, N., & Pascanu, R. (2013). Advances in optimizing recurrent networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 8624-8628). IEEE.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning* (pp. 41-48).

Birge, A. Ö. (2016). Sinir sistemi anatomisi.

Blum, A. L., & Rivest, R. L. (1992). Training a 3-node neural network is NP-complete. *Neural Networks*, *5*(1), 117-127.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., & LeCun, Y. (2015). The loss surfaces of multilayer networks. In *Artificial intelligence and statistics* (pp. 192-204). PMLR.

Coşansu, G. (2015). Diyabet: Küresel bir salgın hastalık. *Okmeydanı Tıp Dergisi*, *31*(ek sayı), 1-6.

Dabbura, I. (2017). *Gradient descent algorithm and ıts variants*. Towardsdatascience. https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

Darken, C., Chang, J., & Moody, J. (1992). Learning rate schedules for faster stochastic gradient search. In *Neural networks for signal processing* (Vol. 2).

Dauphin, Y., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., & Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *arXiv preprint arXiv:1406.2572*.

Dean, J., Corrado, G. S., Monga, R., Chen, K., Devin, M., Le, Q. V., ... & Ng, A. Y. (2012). Large scale distributed deep networks.

Deisenroth, M. P., Faisal, A. A., & Ong, C. S. (2020). *Mathematics for machine learning*. Cambridge University Press.

Dozat, T. (2016). Incorporating nesterov momentum into adam.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, *12*(7).

Ergün, E. (2019). Beyin Kanamaları. Lösante. https://www.losante.com.tr/Blog/

Flanders, A. E., Prevedello, L. M., Shih, G., Halabi, S. S., Kalpathy-Cramer, J., Ball, R., ... & RSNA-ASNR 2019 Brain Hemorrhage CT Annotators. (2020). Construction of a machine learning dataset through collaboration: the RSNA 2019 brain CT hemorrhage challenge. Radiology: Artificial Intelligence, 2(3), e190211.

Goodfellow, I. J., Vinyals, O., & Saxe, A. M. (2014). Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Gülcü, A., & Zeki, K. U. Ş. (2019). Konvolüsyonel sinir ağlarında hiper-parametre optimizasyonu yöntemlerinin incelenmesi. *Gazi Üniversitesi Fen Bilimleri Dergisi Part C: Tasarım ve Teknoloji*, *7*(2), 503-522.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

Hinton, G. (2012). *Neural networks for machine learning*. Coursera, video lectures. 307

Hoffer, E., Hubara, I., & Soudry, D. (2017). Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *arXiv preprint arXiv:1705.08741*.

Judd, J. S. (1990). *Neural network design and the complexity of learning*. MIT press.

Kaggle (2016). *Pima indians diabetes database*. https://www.kaggle.com/uciml/pima-indians-diabetes-database

Kaggle (2020). *Diagnosis of COVID-19 and its clinical spectrum*. https://www.kaggle.com/einsteindata4u/covid19

Kızrak, A. (2019). *Derin Öğrenme İçin Aktivasyon Fonksiyonlarının Karşılaştırılması.* Medium. https://ayyucekizrak.medium.com

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kurt, F. (2018). Evrişimli Sinir Ağlarında Hiper Parametrelerin Etkisinin İncelenmesi.

Lateff, Z. (2020). *Comprehensive guide to logistic regression in R*. Edureka. https://www.edureka.co/blog/logistic-regression-in-r/

Maltarollo, V. G., Honório, K. M., & da Silva, A. B. F. (2013). Applications of artificial neural networks in chemical problems. *Artificial neural networks-architectures and applications*, 203-223.

Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence O (1/k^ 2). In *Doklady an ussr* (Vol. 269, pp. 543-547).

Niu, F., Recht, B., Ré, C., & Wright, S. J. (2011). Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *arXiv preprint arXiv:1106.5730*.

Nizam, H., & Akın, S. S. (2014). Sosyal medyada makine öğrenmesi ile duygu analizinde dengeli ve dengesiz veri setlerinin performanslarının karşılaştırılması. *XIX. Türkiye'de İnternet Konferansı*, 1-6.

Öğündür, G. (2019). Doğruluk(accuracy), kesinlik(precision), duyarlılık(recall) ya da F1 score?. Medium. https://medium.com/@gulcanogundur/do%C4%9Fruluk-accuracy-kesinlik-precision-duyarl%C4%B1l%C4%B1k-recall-ya-da-f1-score-300c925feb38

PİRİM, A. G. H. (2006). Yapay zeka. *Journal of Yaşar University*, *1*(1), 81-93.

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, *4*(5), 1-17.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, *12*(1), 145-151.

Robbins, H. ve Monro, S. (1951). Stokastik bir yaklaşım yöntemi. *Matematiksel istatistiklerin yıllıkları* , 400-407.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Saxe, A. M., McClelland, J. L., & Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Simon, O. (2009). Haykin, Neural Networks and Learning Machines.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Srivastava, A. (2021). *Linear Regression*. Medium. https://medium.com/analytics-vidhya/linear-regression-c6625caf9e8e

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning* (pp. 1139-1147). PMLR.

Sutton, R. (1986). Two problems with back propagation and other steepest descent learning procedures for networks. In Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 1986 (pp. 823-832).

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

TOMAK, L., & Yüksel, B. E. K. (2009). İşlem karakteristik eğrisi analizi ve eğri altında kalan alanların karşılaştırılması. *Journal of Experimental and Clinical Medicine*, *27*(2).

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, *1*(1), 67-82.

Wu, Y. (2019). RSNA Challenge. Medium. https://medium.com/@yunanwu2020/

Yıldız, T.K. (2021) *Hematolojik hastalıkların teşhisinde yapay zeka tekniklerinin performans karşılaştırması,* Ph.D. Thesis, Sakarya University, Sakarya, 134s.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhang, S., Choromanska, A., & LeCun, Y. (2014). Deep learning with elastic averaging SGD. *arXiv preprint arXiv:1412.6651*.

ÖZGEÇMİŞ

Kişisel Bilgiler

Ad Soyad : D***n Ç***r

Uyruk : T.C.

Doğum Yeri ve Tarihi : B*****t 1*/1*/1**0

Medeni Hali : Evli

Telefon : 0 5*1 4*5 8**5

E-posta : d*******r@p***a.m*.e*u.t*

Eğitim

Eğitim Dönemi	Derece	Üniversite	Bölüm
2018-Devam Ediyor	Y.Lisans	Muğla Sıtkı Koçman Üniversitesi	Yapay Zeka Anabilim Dalı
2006-2010	Lisans	Atatürk Üniversitesi	Matematik Bölümü

Yabancı Dil

Dil	Başlangıç	Orta	İleri
İngilizce		X	