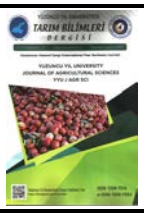
Yüzüncü Yıl Üniversitesi
Tarım Bilimleri Dergisi
(YYU Journal of Agricultural Science)

http://dergipark.gov.tr/yyutbd



Araştırma Makalesi (Research Article)

Derin Sinir Ağları ile En İyi Modelin Belirlenmesi: Mantar Verileri Üzerine Keras Uygulaması

Gazel SER*[1], Cafer Tayyar BATİ[2]

[1]Van Yüzüncü Yıl Üniversitesi, Ziraat Fakültesi, Zootekni Bölümü, Van, Türkiye
[2]Van Yüzüncü Yıl Üniversitesi, Fen Bilimleri Enstitüsü, Zootekni ABD, Van, Türkiye
*Sorumlu yazar e-posta: gazelser@gmail.com

Makale Bilgileri

Geliş: 29.12.2018
Kabul: 23.06.2019
Online Yayınlanma 30.09.2019
DOI: 10.29133/yyutbd.505086

Anahtar kelimeler
Keras,
Mantar,
Optimizasyon yöntemi,
Python,
Yapay sinir ağları.

Öz: Bu çalışma, derin sinir ağlarında en iyi sınıflandırma modelini bulmak amacıyla gerçekleştirilmiştir. Bu amaçla, optimizasyon yöntemi (Sgd, Adagrad, Rmsprop, Adam ve Nadam), aktivasyon fonksiyonu (Tanh ve ReLU) ve nöron sayılarının kombinasyonları kullanılarak 20 farklı model oluşturulmuştur. Oluşturulan model kombinasyonlarının performansları karşılaştırılarak, sınıflandırma için en iyi model belirlenmiştir. Sonuçlara göre; modellerin performanslarının parametrelere bağlı olarak değişkenlik gösterdiği, en başarılı modelin gizli katmanında 64 nöron bulunduğu, aktivasyon fonksiyonunun ReLU olduğu ve optimizasyon yöntemi olarak da Rmsprop kullanıldığı belirlenmiştir (%92 doğruluk). Bununla beraber, en düşük başarı oranıyla sınıflandırma yapan modelin 32 nöronlu, ReLU aktivasyon fonksiyonlu ve Sgd optimizasyon yöntemli model olduğu belirlenmiştir (% 70 doğruluk). Ayrıca tüm sonuçlar göz önüne alındığında; Rmsprop, Adam ve Nadam optimizasyon yöntemlerinin diğer iki yönteme göre, ReLU aktivasyon fonksiyonunun ise Tanh'a göre daha başarılı olduğu belirlenmiştir. Sonuç olarak derin öğrenme çalışmalarında model oluşturulurken; optimizasyon algoritmalarının, aktivasyon fonksiyonlarının ve nöron sayılarının farklı seçeneklerine göre model performanslarını denemek mümkündür. Ayrıca oluşturulan modelde, optimizasyon yöntemlerinin farklı parametrelerinin kombinasyonlarıyla çalışıldığında, veri setine daha uygun mimari elde edilmektedir.

Determining the Best Model with Deep Neural Networks: Keras Application on Mushroom Data

Article Info

Received: 29.12.2018
Accepted: 23.06.2019
Online Published 30.09.2019
DOI: 10.29133/yyutbd.505086

Keywords
Keras,
Mushroom,
Optimization algorithm,
Python,
Artificial neural networks

Abstract: This study was conducted to reveal the best classifying model with deep neural networks. For this purpose, 20 different candidate models of optimization method (Sgd, Adagrad, Rmsprop, Adam and Nadam), activation function (Tanh and ReLU) and combinations of neurons were studied. By comparing the performance of these candidate models, the best model for classification was determined. The present results indicated that the performance of the models varied according to the parameters, the most successful model has 64 neurons in the hidden layer, the activation function was ReLU and the Rmsprop was used as the optimization method (92% accuracy). In addition, it was determined that the model with the lowest success rate was the model with 32 neurons, ReLU activation function and Sgd optimization method (70% accuracy). Also considering all results; Rmsprop, Adam and Nadam optimization methods were found to be more successful than the other two methods and ReLU activation function produced more successful results than Tanh.

As a result, while creating a model in deep learning studies; optimization algorithms, activation functions and number of neurons model performances can be tried according to different options. In addition, when the model is worked with combinations of different parameters of optimization methods, a more suitable architecture is obtained for the data set.

1. Giriş

Yapay sinir ağları (YSA), insan beyninin en temel özelliği olan öğrenme fonksiyonunu gerçekleştiren bilgisayar sistemleri olarak tanımlanabilir (Öztemel, 2012). Bu yöntemde örneklerle ilgili bilgiler toplanmakta, genelleme yapılmakta sonuç olarak da bu genellemeler daha önce hiç karşılaşılmamış örnekler için kullanılmaktadır. Yapay sinir ağlarının bu öğrenme ve genelleme yeteneği nedeniyle araştırmacılar tarafından, farklı alanlarda kullanılmaktadır. Günümüz koşullarında veri hacminin artması, veri kümesindeki değişkenlerin fazlalığı, değişkenler arasındaki karmaşık ve hiyerarşik ilişkilerin çözümlenmesine yönelik çabalar yapay sinir ağları temelli derin öğrenme yaklaşımına olan ilgiyi arttırmıştır. Derin sinir ağları, yapay sinir ağlarının yapısal olarak daha derinleşmiş ve genişlemiş hali olarak tanımlanmaktadır. Yani, derin sinir ağları, yapay sinir ağlarında bulunan gizli katman sayısının ikiden fazla olması ve bu katmanlarda bulunan nöron sayısının fazla olması durumudur (Anonim, 2018a). Derin öğrenme yaklaşımı; konuşma tanıma, bilgisayarlı görü ve doğal dil işleme gibi farklı alanlarda kullanılmaktadır. Derin Sinir Ağı (DNN), Derin İnanç Ağı (DBN), Derin Dönüşümlü Sinir Ağı (CNN) ve Tekrarlayan Nöral Ağ (RNN) gibi çok sayıda farklı derin ağ mimarileri mevcuttur (Li, 2017). Günümüzde grafik işlemci birimleri (GPU) sayesinde, vektör ve matris işlemlerinin daha hızlı yapılmasıyla, çok katmanlı derin sinir ağ mimarileri daha kısa sürede eğitilebilmektedir (Akkol ve ark., 2017; Koptur, 2017; Kızrak ve Bolat, 2018).

Bu çalışmada, derin sinir ağ mimarilerinde önemli bir işlevi bulunan optimizasyon algoritmaları ile aktivasyon fonksiyonlarının temel metodolojisi ve özellikleri verilmiştir. Ayrıca Python kütüphanesi olan Keras kullanılarak, mantarlara ait örnek bir veri seti üzerinde derin sinir ağlarının farklı modelleri oluşturularak en iyi sınıflandırma modelinin belirlenmesi öngörülmüştür.

2. Materyal ve Yöntem

Çalışmada kullanılan Mantar veri seti (Mushroom data set), UCI makine öğrenme veri deposundan alınmıştır. (UCI, 2018). Dua ve Graff (2018) tarafından 8124 mantar örneğinden oluşturulan veri setinde; şapkanın şekli, şapkanın yüzeyi, şapkanın rengi, çürüme durumu, kokunun tanımı, şapkanın alt kısmı, lamellerin aralığı, lamellerin ebadı, lamellerin rengi, sap şekli, kökün durumu, sap yüzeyinin üstündeki halkanın görünümü, sap yüzeyinin üstündeki halkanın rengi, sap yüzeyinin altındaki halkanın görünümü, sap yüzeyinin altındaki halkanın rengi, örtü tipi, örtü rengi, halka sayısı, halka tipi, spor rengi, populasyon yoğunluğu ve yaşam alanı şeklinde 22 bağımsız değişken yer almaktadır. Bağımlı değişken ise mantarların "yenilebilir (edible=e)" veya zehirli (poisonous=p)" şeklinde sınıflandırılmasından oluşturulmuştur. Çalışmada, veri setinin %80'i eğitim ve %20'si test verisi olarak ayrılmıştır. Modellerin performansları, test verilerine ait doğruluk (test accuracy) ve hata değerlerine (test loss) göre değerlendirilmiştir. Çalışmada Çizelge 1'de verilen nöron sayıları, aktivasyon fonksiyonları ve optimizasyon yöntemleri kullanılarak, 20 farklı model oluşturulmuş ve bağımlı değişken sınıflandırılmıştır.

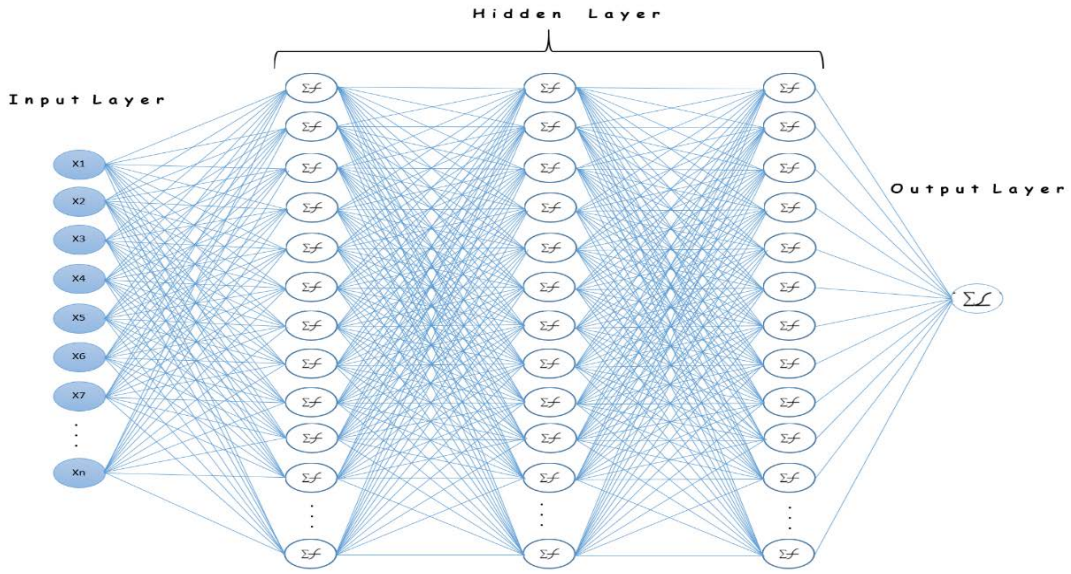
Çizelge 1. Çalışmada oluşturulan modellere ait hiper parametreler

Nöron Sayıları	32 64
Aktivasyon Fonksiyonları	Tanh ReLU
Optimizasyon Yöntemleri	Sgd Adagrad Rmsprop Adam Nadam

Oluşturulan her modelin çıkış katmanında, sigmoid aktivasyon fonksiyonu kullanılmıştır. Epok sayısı 100 olarak alınırken, öğrenme oranı ise optimizasyon algoritmalarının default olarak aldığı değerler kullanılmıştır.

2.1. Yapay sinir ağları

Çalışmada kullanılan derin sinir ağları algoritmaları, insan beyninin bilgi işleme yönteminden esinlenerek geliştirilen yapay sinir ağlarının çok katmanlı halidir. İlk olarak geliştirilen ve en ilkel yapay sinir ağı olarak değerlendirilen tek katmanlı algılayıcı (single layer perceptron) doğrusal olmayan problemlerin çözümünde yeterli olmadığından çok katmanlı algılayıcılar (multi-layer perceptron) geliştirilmiştir (Fernandez ve ark., 2006). Şekil 1'de üç gizli katmandan oluşan ve çalışmada kullanılan derin sinir ağı mimarisi verilmiştir.



Şekil 1. Derin sinir ağı mimarisi.

Çok katmanlı algılayıcı giriş (input layer), gizli (hidden layer) ve çıkış (output layer) olmak üzere üç katmandan oluşmaktadır. Tek katmanlı algılayıcının aksine doğrusal olmayan sınıflandırma yapılabilmektedir. Çözülen problemin niteliğine göre, gizli katmanların sayısı ile bu katmanlarda bulunan nöron sayıları değişebilmektedir. Bir mimarinin modelleme sürecinde iyi bir genelleme oluşturmak için giriş veri seti, iki ya da üç ayrı parçaya ayrılır. Derin sinir ağı algoritmalarında tüm veri seti genellikle eğitim veri seti (training) ve test veri seti (testing) olarak adlandırılan iki veri kümesine bölünmektedir. Eğitim veri setinin rolü sinir ağındaki ağırlıkların tahminlerini hesaplamak, test veri setinin rolü ise elde edilen ağırlık değerlerinin doğruluğunu modelden gizlenen veriler ile test etmektir (Priddy ve Keller, 2005; Okut, 2018). Çok katmanlı algılayıcı modelleri, ileri besleme yöntemi ve geri yayılım algoritması ile çalışmaktadır. Geri yayılım algoritmasının ilk aşamasında, değişkenler eğitim ağına sunulur. Her bir değişkenin her bir nöronu için bir ağırlık değeri atanarak, değerler ağırlıklar ile çarpılıp toplandıktan sonra, bias değerinin eklenmesiyle gizli katmandaki nöronlarda bir aktivasyon fonksiyonu ile sonraki katmandaki tüm nöronlara gönderilir. Bu değerler, çıkış katmandaki nöronlar için giriş değerlerini oluşturmaktadır. Çıkış katmanında benzer işlemler gerçekleştirildikten sonra bir çıktı değeri elde edilerek, çıktı ile gerçek değer karşılaştırılır ve hata değeri elde edilir (E_D).

$$\hat{y}_i = f_2\left\{\sum_{j=1}^{R} w_{kj}^{(2)}\left[f_1(\sum_{j=1}^{R} w_{kj}^{(1)} x_j + b_k^{(1)})\right] + b_k^{(2)}\right\} \quad (1)$$

Eşitlik 1'de $f_1, f_2: Aktivasyon\ fonksiyon,\ w_{kj}: Ağırlıklar, x_j: Girdiler, b_k: Bias$ değerlerini ifade etmektedir.

$$E_D = \frac{1}{2}\sum_{i=1}^{p}(y_i - \hat{y}_i)^2 \quad (2)$$

Eşitlik 2'de $y_i: Gerçek\ değerleri, \hat{y}_i : Tahmini\ değerleri$ ifade etmektedir.

İkinci aşamada, çıkış katmanından elde edilen hatalar, geriye doğru bir yönde diğer katmanlara yayılır. Bu hatalar, ağdaki ağırlıklara göre kayıp işlevinin eğimini hesaplamak için kullanılır. Daha sonra kayıp fonksiyonunu (hatayı) en aza indirmek amacıyla, ağırlıkları güncellemek için optimizasyon yöntemi kullanılır (Li, 2017). Bu işlemde hatanın ağırlıklara göre kısmı türevi, türevde zincir kuralı kullanılarak yapılır (Eşitlik 3 ve 4). Ağırlıkların güncelleştirme işlemi, mimarinin öğrenme süreci olarak ifade edilmekte ve Eşitlik 5 ile gösterilmektedir (Okut ve ark., 2011).

$$\frac{\partial E_D(t)}{\partial w_{kj}^2(t)} = \frac{\partial E_D(t)}{\partial n_k^2(t)}\frac{\partial n_k^2(t)}{\partial w_{kj}^2(t)} \quad (3)$$

$$\frac{\partial E_D(t)}{\partial b_k^2(t)} = \frac{\partial E_D(t)}{\partial n_k^2(t)}\frac{\partial n_k^2(t)}{\partial b_k^2(t)} \quad (4)$$

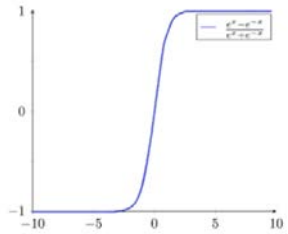
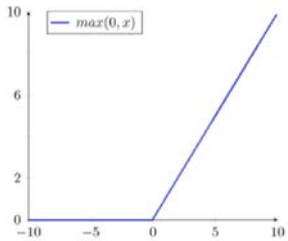
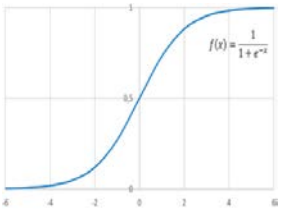
$$w^{l+1} = w^l - \alpha\frac{\partial E_D}{\partial w^l} \quad (5)$$

Eşitlik 5'te w^{l+1}: güncellenmiş ağırlık, w^l: bir önceki iterasyondaki ağırlık, α: öğrenme oranını ifade etmektedir.

2.2. Aktivasyon fonksiyonu

Aktivasyon fonksiyonları; bir katmanda bulunan nöronlardaki çıktı değerini sonraki katmanlara iletilirken kullanılır. Bu çıktı değerinin, diğer katmanlara iletilip iletilmeyeceğine karar verebilmek için bir eşik değerinin belirlenmesi gerekir. Çünkü bir yapay sinir hücresindeki bilginin değeri $(-\infty, +\infty)$ aralığında olabilir ve nöron gerçek değerin sınırlarını bilmeyebilir. Bu nedenle, nöronun aktif olup-olmaması gerektiğine karar verebilmesi için aktivasyon fonksiyonlarına ihtiyaç duyulur. Böylece bir nöron tarafından üretilen çıktı değerini kontrol edebilecek ve dış bağlantıların nöronu aktif olarak görüp-görmeyeceğine karar verilebilecektir (Sharma, 2017). Yapay sinir ağları daha çok doğrusal olmayan sınıflandırmalarda kullanıldığından, aktivasyon fonksiyonu genellikle doğrusal olmayan bir fonksiyon seçilir. Mimarinin öğrenme sürecinde geri yayılım algoritması kullanılırken, aktivasyon fonksiyonunun türevi de kullanıldığı için türevi kolay hesaplanabilen bir aktivasyon fonksiyonunun kullanılması algoritmanın hızı için önemlidir (Şengöz, 2017; Kutlu, 2018). Çizelge 2'de çalışmada kullanılan aktivasyon fonksiyonları ve özellikleri verilmiştir.

Çizelge 2. Çalışmada kullanılan aktivasyon fonksiyonları ve özellikleri

Aktivasyon Fonksiyonları	Grafik	Açıklama
Tanh		Bu fonksiyon girdi değerlerinin her biri için -1 ile +1 arasında değer üretir (Nwankpa ve ark., 2018).
Rectified Linear (ReLU)		Bu fonksiyonda giriş değeri sıfırın altında iken çıktı sıfırdır, ancak giriş değeri sıfırın üzerinde ise çıkış giriş değerine eşittir ve bağımlı değişkenle doğrusal bir ilişki oluşmaktadır (Anonim, 2018b).
Sigmoid		Sigmoid aktivasyon fonksiyonu tanım kümesindeki elemanların her biri için sıfır ile bir arasında bir değer üretir (Nwankpa ve ark., 2018).

2.3. Optimizasyon yöntemleri

Derin öğrenme uygulamalarında, öğrenme işleminin sağlıklı bir şekilde sonuçlanması için hata fonksiyonunun mutlak minimum değerinin bulunması gerekmektedir. Bu işlem, optimizasyon yöntemleri kullanılarak gerçekleştirilmektedir. Optimizasyon, ağın ürettiği çıkış değeri ile gerçek değer arasındaki farkı yani hatayı en küçük yapmak için kullanılan yöntemlerdir. Yapay sinir ağlarının optimizasyonu için en çok kullanılan yöntemlerden biri gradyan inişidir (Gradient descent). Tek iterasyonda kullanılan veri setinin büyüklüğüne bağlı olarak üç adet gradyan inişi yöntemi (Batch Gradient Descent, Stochastic Gradient Descent, Mini-Batch Gradient Descent) vardır. Gradyan inişi yöntemini esas alan çeşitli algoritmalar (Rmsprop, Adagrad, Adam, Nadam) mevcuttur (Kurt, 2018).

Optimizasyon algoritmalarında öğrenme katsayısının ayarlanması modelin eğitiminde kritik bir rol oynamaktadır. Ancak, her algoritma ile modeldeki öğrenme katsayısını tam olarak ayarlamak mümkün değildir. Bu problemi çözmek için gradyan yöntemlerinin çeşitli varyasyonları önerilmiştir (Li, 2017). Çalışmada kullanılan optimizasyon algoritmaları çizelge 3'de verilmiştir.

Çizelge 3. Çalışmada kullanılan optimizasyon algoritmaları ve özellikleri

Optimizasyon Yöntemi	Formül[1]	Açıklama
Sgd	$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta_t; x^{(i)}; y^{(i)})$	SGD algoritmasında, tüm eğitim verisi yerine bir eğitim örneği üzerinden hesaplama yapılmaktadır. Böylelikle oluşabilecek hafıza yetersizliği problemlerinin de önüne geçilmiş olur (Kurt, 2018).
Adagrad	$g_{t,i} = \nabla_{\theta_t} J(\theta_{t,i})$ $\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} g_{t,i}$	Adagrad, her parametre için her adımda t farklı öğrenme katsayısı kullanarak, her bir parametre için farklı güncelleme yapmaktadır. Böylece, öğrenme katsayısını manuel olarak ayarlama ihtiyacını ortadan kaldırmaktadır. Adagrad'da her parametrenin kendi öğrenme hızı vardır ve güncelleme işleminde öğrenme katsayısı değerinin bölündüğü ifadenin eğitim süresince büyümeye devam etmesi sonucunda öğrenme katsayısı aşırı küçülmektedir (Çarkacı, 2018).
Rmsprop	$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1 g_t^2$ $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$ $g_t = \nabla_{\theta_t} J(\theta_t)$	Rmsprop, Adagrad algoritmasında öğrenme katsayısının aşırı küçültme sorununa bir çözüm yolu olarak geliştirilmiştir. Adagrad'daki geçmiş bütün eğimlerin karelerinden elde edilen değerlerin tamamını kullanmak yerine, değer miktarını belli bir çerçeve boyutu ile kısıtlamıştır (Ruder, 2016; Kurt, 2018).
Adam	$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$ $v_t = \beta_2 v_{t-1} + (1 - \beta_2) {g_t}^2$ $m'_t = \frac{m_t}{1 - \beta_1^t}$, $v'_t = \frac{v_t}{1 - \beta_2^t}$ $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v'_t + \epsilon}} m'_t$	Bu yöntemde, Rmsprop'ta yapıldığı gibi geçmiş eğimlerin karelerinin üssel olarak ağırlıklandırılmış ortalamalarının (v_t) kullanılmasının yanında, momentum değişikliklerini (m_t) de önbellekte saklar. Yani Rmsprop ve momentumu birleştirir. Varsayılan değerler β_1 için 0.9; β_2 için 0.999 ve ϵ için 10^{-8} olarak belirtilmiştir (Li, 2017; Çarkacı, 2018).
Nadam	$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v'_t + \epsilon}} (\beta_1 m'_t + \frac{(1 - \beta_1) g_t}{1 - \beta_1^t})$	Nadam, Adam ve Nesterov accelerated gradienti (NAG) yöntemlerinin birleşiminden oluşmaktadır. NAG algoritmasını Adam algoritmasına dahil etmek için, momentum ifadesinde değişiklik yapılarak sonuçta mevcut güncelleştirme kuralı elde edilir (Ruder, 2016).

[1] $\theta \in \mathbb{R}^d$: Model parametreleri; η: Öğrenme katsayısı; $\nabla_\theta J(\theta_t; x^{(i)}; y^{(i)})$: Parametrelere bağlı olarak hedef fonksiyonunun eğimi; $G_{t,ii}$: her bir köşegen elemanı, θ_i parametresine göre, t. iterasyona kadar hesaplanmış eğim değerlerinin kareleri toplamı; ϵ: öğrenme katsayısının 0'a bölünmesini engellemek için atanan sabit değer.

3. Bulgular

Derin sinir ağı modellerinin performansları Çizelge 4'de verilmiştir. Bununla beraber, çizelge 5'de en iyi performansa sahip, çizelge 6'da ise en düşük performansa sahip beş modelin doğruluk ve hata grafikleri verilmiştir.

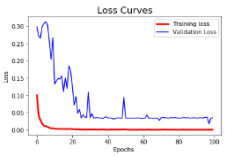
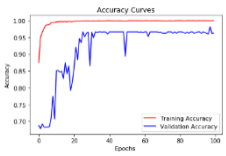
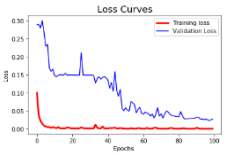
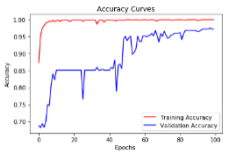
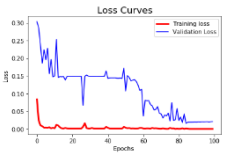
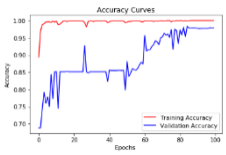
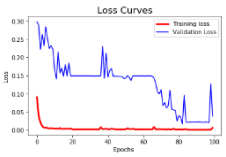
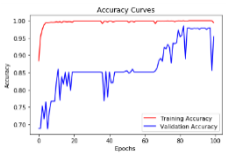
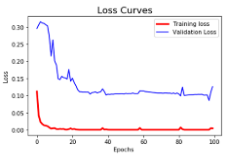
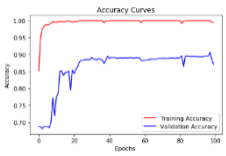
Çizelge 4. Çalışmada kullanılan modellere ait sonuçlar

Nöron Sayısı	Aktivasyon Fonksiyonu	Model No	Optimizasyon Yöntemi	Doğruluk	Hata	Test Doğruluk	Test Hata
32	ReLU	1	Sgd	0.98	0.02	0.70	0.30
		2	Adagrad	0.99	0.01	0.75	0.25
		3	Nadam	0.99	0.004	0.86	0.14
		4	Adam	0.99	0.003	0.86	0.14
		5	Rmsprop	0.99	0.004	0.84	0.16
	Tanh	6	Sgd	0.98	0.02	0.74	0.26
		7	Adagrad	0.99	0.009	0.73	0.27
		8	Nadam	0.99	0.003	0.87	0.13
		9	Adam	0.99	0.004	0.83	0.17
		10	Rmsprop	0.99	0.006	0.78	0.22
64	ReLU	11	Sgd	0.98	0.02	0.71	0.29
		12	Adagrad	0.99	0.005	0.80	0.20
		13	Nadam	0.99	0.002	0.87	0.13
		14	Adam	0.99	0.003	0.89	0.11
		15	Rmsprop	0.99	0.002	0.92	0.08
	Tanh	16	Sgd	0.98	0.02	0.75	0.25
		17	Adagrad	0.99	0.006	0.78	0.22
		18	Nadam	0.99	0.003	0.84	0.16
		19	Adam	0.99	0.004	0.82	0.18
		20	Rmsprop	0.99	0.007	0.78	0.22

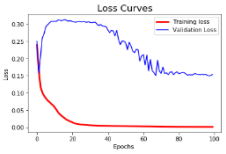
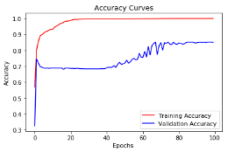
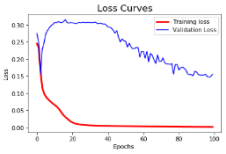
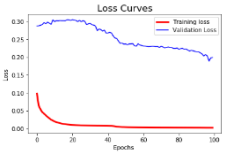
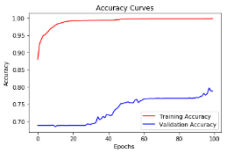
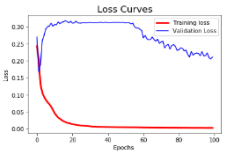
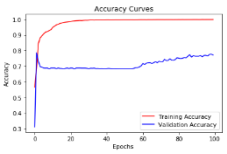
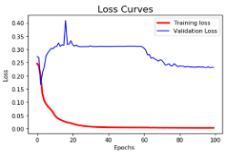
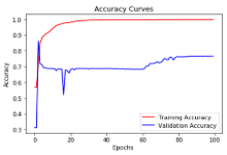
Çizelge 4'e göre; %92 doğruluk ile en başarılı sınıflandırma yapan modelin gizli katmanında 64 nöron bulunan, ReLU aktivasyon fonksiyonlu, Rmsprop optimizasyon tekniği ile 15'nolu model olduğu belirlenmiştir. Oluşturulan 20 model arasından 1'nolu model %70 doğruluk ile en düşük başarı oranıyla sınıflandırma yapmıştır. Test verilerinin doğruluk değerleri ve hata değerleri incelendiğinde, modellerin sınıflandırma performanslarının (0.70-0.92) aralığında değişim göstermiştir. Çizelge 5 ve 6 incelendiğinde ise en başarılı modellerde Rmsprop, Adam ve Nadam optimizasyon yöntemlerinden elde edilirken, en düşük başarıya sahip modellerde Sgd ve Adagrad optimizasyon yöntemlerinden elde edilmiştir. Ayrıca çizelge 5'de 15'nolu modelin (Rmsprop optimizasyon tekniği ile) yaklaşık olarak 35'inci Epok'da en yüksek performansa ulaştığı ve eğitimin ortalama olarak bu şekilde devam ettiği görülmektedir. 14'nolu model (Adam ile) incelendiğinde ise en yüksek performansa ulaşma işleminin neredeyse 100'üncü Epok'u bulduğu görülmektedir. Bu durum, kullanılan optimizasyon algoritmasının hızı ve kararlılığı için önemli bir durum olup, göz önünde bulundurulmalıdır. Böylece

15'nolu modelin, 14'nolu modele nazaran daha hızlı bir şekilde, eğitimde yüksek performansa ulaştığını söylemek mümkündür.

Çizelge 5. En başarılı beş modele ait sonuçlar ile doğruluk ve hata grafikleri

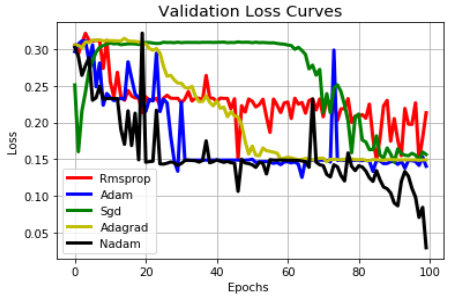
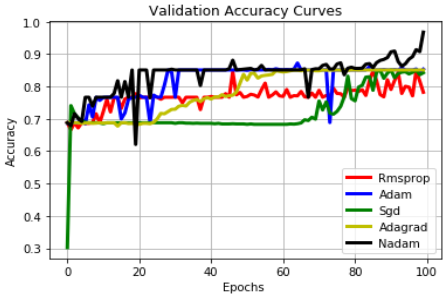
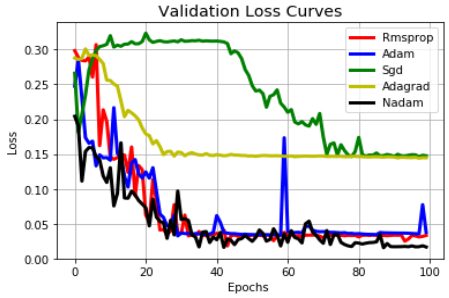
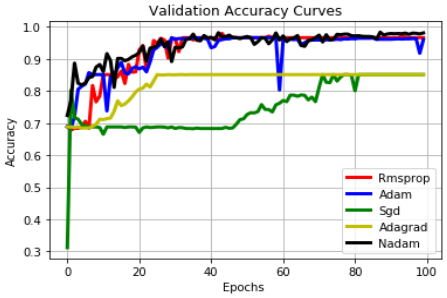
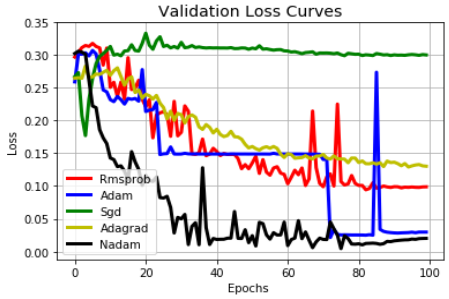
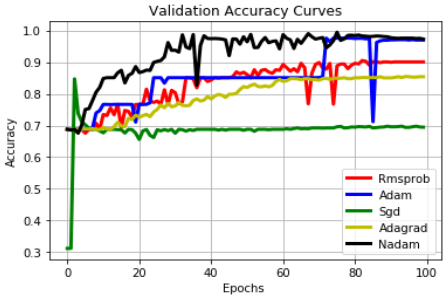
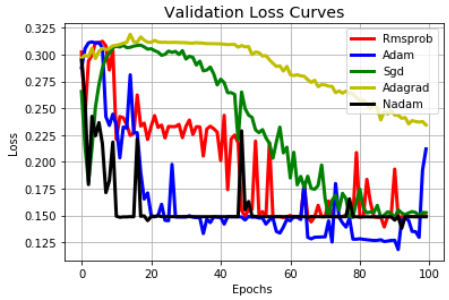
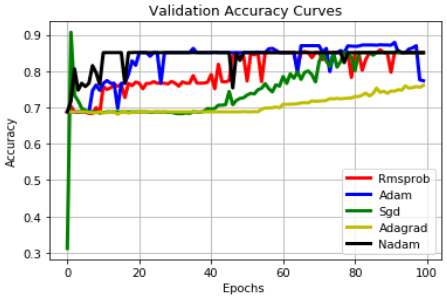
Nöron Sayısı	Akt. Fonk.	Model No	Optimizasyon Yöntemi	Test Doğruluk	Test Hata	Hata Grafiği	Doğruluk Grafiği
64	ReLU	15	Rmsprop	0.92	0.08		
64	ReLU	14	Adam	0.89	0.11		
64	ReLU	13	Nadam	0.87	0.13		
32	Tanh	8	Nadam	0.87	0.13		
32	ReLU	4	Adam	0.86	0.14		

Çizelge 6. En düşük başarıya sahip beş modele ait sonuçlar ile doğruluk ve hata grafikleri

Nöron Sayısı	Akt. Fonk.	Model No	Optimizasyon Yöntemi	Test Doğruluk	Test Hata	Hata Grafiği	Doğruluk Grafiği
64	Tanh	16	Sgd	0.75	0.25		
32	Tanh	6	Sgd	0.74	0.26		
32	Tanh	7	Adagrad	0.73	0.27		
64	ReLU	11	Sgd	0.71	0.29		
32	ReLU	1	Sgd	0.70	0.30		

Çizelge 7'de çalışmada kullanılan beş optimizasyon yönteminin performanslarının bir arada bulunduğu doğruluk ve hata grafikleri verilmiştir. Çizelge 7'ye göre; Rmsprop, Adam, Nadam ve Adagrad optimizasyon tekniklerinin ReLU aktivasyon fonksiyonuyla, Sgd optimizasyon tekniğinin ise Tanh aktivasyon fonksiyonuyla kullanıldığında daha başarılı sonuçlar ortaya koymuştur. Ayrıca Rmsprop optimizasyon algoritmasının, ReLU aktivasyon fonksiyonuyla kullanıldığında 64 nöronlu algoritmanın, 32 nöronlu algoritmaya kıyasla daha başarılı olduğu belirlenmiştir (0.92>0.84). Benzer şekilde, Adam ve Nadam optimizasyon algoritmalarının ReLU aktivasyon fonksiyonuyla kullanıldığında 64 nöronlu modelin daha başarılı olduğu (sırasıyla 0.89>0.86, 0.87>0.86), Tanh aktivasyon fonksiyonuyla kullanıldığında ise 32 nöronlu modelin daha başarılı olduğu belirlenmiştir (sırasıyla 0.83>0.82, 0.87>0.84). Adagrad ve Sgd optimizasyon algoritmaları için ise 64 nöronlu modellerin daha başarılı olduğu belirlenmiştir.

Çizelge 7. Optimizasyon algoritmalarının doğruluk ve hata grafikleri

Nöron Sayısı	Akt. Fonk.	Optimizasyon Algoritmaları	
64	Tanh		
64	ReLU		
32	ReLU		
32	Tanh		

4. Tartışma ve Sonuç

Optimizasyon yöntemleri, aktivasyon fonksiyonları ve nöron sayıları kullanılarak oluşturulan 20 farklı derin sinir ağı modellerinde, Adam ve Nadam optimizasyon yöntemlerinin diğer iki yönteme kıyasla daha başarılı olduğu, başarı oranı en düşük yöntemin ise Sgd olduğu belirlenmiştir (Çizelge 5 ve 6). Çalışma sonuçları son yıllarda yapılan çalışmalar ile paralellik göstermektedir. Ruder (2016) ve Walia (2017a) çalışmalarında; sinir ağının derinliği ya da karmaşıklığına bağlı olarak adaptif öğrenme oranı yöntemlerinden (Rmsprop, Adam, Nadam vb.) birinin seçilmesinin daha uygun olacağını belirtmişlerdir. Ayrıca Rmsprop, Adadelta ve Adam optimizasyon yöntemlerinin benzer durumlarda, iyi işleyen yöntemler olduğunu ve yöntemlerin öğrenme oranını ayarlama ihtiyacını ortadan kaldırdığını belirtmişlerdir. Bununla birlikte, Sgd algoritmasında uygun bir öğrenme oranını seçmenin zor olabileceği, çok küçük bir öğrenme oranının yavaş bir yakınsamaya yol açarken, çok büyük bir öğrenme oranının yakınsamaya engel olabileceği belirtilmiştir. Li (2017) ve Maksutov (2018)

tarafından yapılan çalışmada ise en iyi performansı gösteren optimizasyon yönteminin "Adam" olarak elde edilmiştir. Çalışmamız sonuçlarının aksine Anonim (2016) tarafından yapılan çalışmada, optimizasyon teknikleri kullanılarak oluşturulan 4 farklı model Julia kullanılarak karşılaştırılmış ve Tanh aktivasyon fonksiyonlu 100 nöronlu modelde en iyi performansı Sgd optimizasyon yönteminden elde edildiği ifade edilmiştir. Söz konusu çalışmadaki bu durumun oluşmasında modelde kullanılan 100 nöronun etkili olduğu değerlendirilmektedir. Benzer şekilde çalışmamızda da Sgd optimizasyon yönteminin Tanh aktivasyon fonksiyonu ile daha başarılı olduğu, nöron sayısının 32'den 64'e çıkmasının bu modeli daha başarılı hale getirdiği görülmektedir. Çalışma sonuçları incelendiğinde; ReLU aktivasyon fonksiyonunun Tanh'a göre daha başarılı sonuçlar ortaya koymuştur (Çizelge 7). Bu durum birçok çalışma sonuçlarıyla da desteklenmektedir (Walia, 2017b; Ramachandran ve ark., 2017).

Derin sinir ağları ile yapılan çalışmalarda; kullanılan veri setinin yapısına, mimarinin boyutuna, derinliğine, kullanılan optimizasyon yöntemlerinin ve aktivasyon fonksiyonlarının çeşidine bağlı olarak model performanslarının farklılaştığı bilinmektedir. Ayrıca optimizasyon algoritmalarının performansı, parametrelerin seçimine ve sinir ağının nasıl yapılandırılacağına bağlı olarak değişmektedir (Anonim, 2016; Li, 2017; Maksutov, 2018).

Sonuç olarak, derin öğrenme çalışmalarında optimizasyon algoritmalarının seçim kriterleri açık olmamakla beraber, problemin yapısına ve parametrelere bağlı olarak algoritmalar farklı performanslar göstermektedir. Dolayısıyla, derin öğrenme çalışmalarında model oluşturulurken optimizasyon algoritmaları, aktivasyon fonksiyonları ve nöron sayılarının farklı kombinasyonlarının oluşturulması durumunda, veri setine daha uygun mimari elde edilebilecektir.

Kaynakça

Akkol, S., Akıllı, A., & Cemal, İ. (2017). Comparison of artificial neural network and multiple linear regression for prediction of live weight in hair goats. *Yüzüncü Yıl Üniversitesi Tarım Bilimleri Dergisi*, *27*(1), 21-29.

Anonim. (2016). Optimization Techniques Comparison in Julia: Sgd, Momentum, Adagrad, Adadelta, Adam. https://int8.io/comparison-of-optimization-techniques-stochastic-gradient-descent-momentum-adagrad-and-adadelta/. Erişim tarihi: 07.10.2018.

Anonim. (2018a). Deep Neural Network. https://www.techopedia.com/definition/32902/deep-neural-network. Erişim tarihi: 20.09.2018.

Anonim. (2018b). Activation Functions: Neural Networks. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. Erişim tarihi: 09.09.2018.

Çarkacı, N. (2018). Derin öğrenme uygulamalarında en sık kullanılan hiper parameteler. https://medium.com/deep-learning-turkiye/derin-ogrenme-uygulamalarinda-en-sik-kullanilan-hiper-parametreler-ece8e9125c4. Erişim tarihi: 06.11.2018.

Dua, D., & Graff, C. (2018). UCI Machine Learning Repository (http://archive.ics.uci.edu/ml). Irvine, CA: University of California, School of Information and Computer Science. Erişim tarihi: 16.11.2018.

Fernandez, C., Soria, E., Martin, J.D., & Serrano, A.J. (2006). Neural networks for animal science applications: Two case studies. *Expert Systems With Applications. 31*, 444-450.

Kızrak, M. A., & Bolat, B. (2018). Derin öğrenme ile kalabalık analizi üzerine detaylı bir araştırma. *Bilişim Teknolojileri Dergisi.* 11(3), 263-286.

Koptur, M. (2017). Yapay Sinir Ağları ve Derin Öğrenme-1. https://makineogrenimi.wordpress.com/2017/07/15/yapay-sinir-aglari-ve-derin-ögrenme-1/. .Erişim tarihi: 12.10.2018.

Kurt, F. (2018). *Evrişimli Sinir Ağlarında Hiper Parametrelerin Etkisinin İncelenmesi* (Yüksek Lisans Tezi). Hacettepe Üniversitesi, Fen Bilimleri Enstitüsü. Ankara, Türkiye.

Kutlu, H. (2018). *Biyoistatistik Temelli Bilimsel Araştırmalarda Derin Öğrenme Uygulamaları* (Yüksek Lisans Tezi). Yakındoğu Üniversitesi, Sağlık Bilimleri Enstitüsü. Lefkoşa, Kıbrıs.

Li, P. (2017). Optimization algorithms for deep learning. department of systems engineering and engineering management. http://lipiji.com/docs/li2017optdl.pdf. Erişim tarihi: 06.09.2018.

Maksutov, R. (2018). Deep study of a not very deep neural network. part 3b: choosing an optimizer. https://medium.com/@maksutov.rn/deep-study-of-a-not-very-deep-neural-network-part-3b-choosing-an-optimizer-de8965aaf1ff. Erişim tarihi: 07.10.2018.

Nwankpa, C.E., Ijomah, W., Gachagan, A., & Marshall, S. (2018). Activation fucntions: Comparison of trends in practice and research for deep learning. https://arxiv.org/pdf/1811.03378.pdf Erişim tarihi: 24.03.2019.

Okut, H. (2018). *Machine Learning Methods for Big Data and Genomic Selection: R Application.* Workshop Material, International Agricultural Congress, 8 May, 2018. Van, Turkey.

Okut, H., Gianola, D., Rosa, G. J. M., & Weigel, K. A. (2011). Prediction of body mass index in mice using dense molecular markers and a regularized neural network. Genetics Research. 93(3), 189-201.

Öztemel, E., (2012). *Yapay Sinir Ağları*, 3. Basım, Papatya Yayıncılık Eğitim, İstanbul.

Priddy, K. L., & Keller, P. E. (2005). *Artificial neural network: An Introduction,* 1nd. ed., Spie Press, Washington.

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. https://arxiv.org/abs/1710.05941. Erişim tarihi: 27.10.2017.

Ruder, S. (2016). An Overview of Gradient Descent Optimization Algorithms. http://adsabs.harvard.edu/abs/2016arXiv160904747R. Erişim tarihi: 27.09.2018.

Sharma, A. (2017). Understanding activation functions in neural networks. https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0. Erişim tarihi: 13.10.2018.

Şengöz, N. (2017). Yapay sinir ağları. http://www.derinogrenme.com/author/nilgunsengoz/. Erişim tarihi: 13.10.2018.

UCI (2018). UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom Erişim tarihi: 17.08.2018.

Walia, A. S. (2017a). Types of optimization algoritms used in neural networks and ways to optimize gradient descent. https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f. Erişim tarihi: 10.10.2018.

Walia, A. S. (2017b). Activation functions and ıt's types-which is better. https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-9a5310cc8f. Erişim tarihi: 10.10.2018.