Dicle University Journal of Engineering

Volume 12 | Issue 2 Article 1

2021

Investigation of Hyperparametry Methods and Kits for Deep Neural Networks

Sara Altun
Inonu University, sara.altun@inonu.edu.tr

M. Fatih Talu
Inonu University, fatih.talu@gmail.com

Dicle University
Journal of Engineering
https://dergipark.org.tr/tr/pub/dumf
duje.dicle.edu.tr

Derleme Makalesi / Review Article

Derin Sinir Ağları için Hiperparametre Metodlarının ve Kitlerinin İncelenmesi

Investigation of Hyperparametry Methods and Kits for Deep Neural Networks

Sara ALTUN[1*], M. Fatih TALU[2]

[1] İnönü Üniversitesi, Bilgisayar Mühendisliği Bölümü, Malatya, sara.altun@inonu.edu.tr
[2] İnönü Üniversitesi, Bilgisayar Mühendisliği Bölümü, Malatya, fatih.talu@gmail.com

MAKALE BİLGİLERİ

Makale geçmişi:

Geliş: 10 Temmuz 2020
Düzeltme: 1 Aralık 2020
Kabul: 2 Aralık 2020

Anahtar kelimeler:

Derin Öğrenme Ağları, Hiperparametre, Otomatik Makine Öğrenimi

ÖZ

Otomatik makine öğrenimi (AutoML) ve derin sinir ağları birçok hiperparametreye sahiptir. Karmaşık ve hesapsal maliyet olarak pahalı makine öğrenme modellerine son zamanlarda ilginin artması, hiperparametre optimizasyonu (HPO) araştırmalarının yeniden canlanmasına neden olmuştur. HPO'nun başlangıcı epey uzun yıllara dayanmaktadır ve derin öğrenme ağları ile popülaritesi artmıştır. Bu makale, HPO ile ilgili en önemli konuların gözden geçirilmesini sağlamaktadır. İlk olarak model eğitimi ve yapısı ile ilgili temel hiperparametreler tanıtılmakta ve değer aralığı için önemleri ve yöntemleri tartışılmaktadır. Sonrasında, özellikle derin öğrenme ağları için etkinliklerini ve doğruluklarını kapsayan optimizasyon algoritmalarına ve uygulanabilirliklerine odaklanılmaktadır. Aynı zamanda bu çalışmada HPO için önemli olan ve araştırmacılar tarafından tercih edilen HPO kitlerini incelenmiştir. İncelenen HPO kitlerinin en gelişmiş arama algoritmaları, büyük derin öğrenme araçları ile fizibilite ve kullanıcılar tarafından tasarlanan yeni modüller için genişletilebilme durumlarını karşılaştırmaktadır. HPO derin öğrenme algoritmalarına uygulandığında ortaya çıkan problemler, optimizasyon algoritmaları arasında bir karşılaştırma ve sınırlı hesaplama kaynaklarına sahip model değerlendirmesi için öne çıkan yaklaşımlarla sonuçlanmaktadır.

Doi: 10.24012/dumf.767700

ARTICLE INFO

Article history:

Received: 10 July 2020
Revised: 1 December 2020
Accepted: 2 December 2020

Keywords:

Deep Learning Networks, Hyper Parameter, Automatic Machine Learning

ABSTRACT

Automatic machine learning (AutoML) and deep neural networks have many hyperparameters. The recent increasing interest in complex and cost-effective machine learning models has led to the revival of hyperparameter optimization (HPO) research. The beginning of HPO has been around for many years and its popularity has increased with deep learning networks. This article provides important issues related to the revision of the HPO. First, basic hyperparameters related to the training and structure of the model are introduced and their importance and methods for the value range are discussed. Then, it focuses on optimization algorithms and their applicability, especially for deep learning networks, covering their effectiveness and accuracy. Then, it focuses on optimization algorithms and their applicability, especially for deep learning networks, covering their effectiveness and accuracy. At the same time, this study examined the HPO kits that are important for HPO and are preferred by researchers. The most advanced search algorithms of the analyzed HPO kits compare the feasibility and expandability for new modules designed by users with large deep learning tools. Problems that arise when HPO is applied to deep learning algorithms result in prominent approaches for model evaluation with a comparison between optimization algorithms and limited computational resources.

* Sorumlu yazar / Correspondence
Sara ALTUN
✉ sara.altun@inonu.edu.tr

Please cite this article in press as S. Altun, M. F. Talu, "Derin Sinir Ağları için Hiperparametre Metodlarının ve Kitlerinin İncelenmesi", DUJE, vol. 12, no.2, pp. 187-199, March 2021.

Giriş

Derin sinir ağları, makine öğrenme sistemlerindendir. Bir sinir ağının mimarisi, düzenlemesi ve optimizasyonu hiperparametre seçimine geniş oranda bağlıdır. Hiperparametre optimizasyonu (HPO), sinir ağı yapısı ve modelin eğitim sürecinde optimum hiperparametre aramada AutoML'in önemli bileşenidir. Otomatik hiperparametre optimizasyonu (HPO), adil şekilde karşılaştırmaları kolaylaştırmaktadır [1]. HPO probleminin 1990'lara dayanan uzun bir geçmişi vardır (örneğin, [2-5]). Ayrıca ilk dönemlerinde farklı hiperparametre yapılandırmalarının farklı veri kümeleri için en iyi sonucu olduğu belirlenmiştir [3]. Gizli katmanların sayısı ve aktivasyon fonksiyonu gibi modelin yapısının oluşturulmasında, parçacık boyutu, optimizasyon algoritmaları, stokastik gradyan azalması (Stochastic Gradient Descent, SGD), öğrenme oranı (Learning Rate, LR) gibi model eğitilirken verimliliğinin ve doğruluğunun belirlenmesinde rol alabilmektedir. HPO, model tasarımının son adımı ve sinir ağını eğitmenin ilk adımı olarak görülebilir. Hiperparametrelerin eğitim esnasındaki doğruluğu ve hızı üzerindeki etkisi göz önüne alındığında, eğitim süreci başlamadan önce dikkatli bir şekilde deneyimlenmelidir [6]. HPO işlemi, insanları makine öğrenme sisteminin döngüsünden çıkarmak için makine öğrenme modelinin hiperparametrelerini otomatik olarak optimize etmektedir.

HPO, son yıllarda derin öğrenme modellerinin geliştirilmesinde en iyi doğruluk sonucunu bulmak için sinir ağlarının sayısının arttırılması [7], daha az ağırlık ve parametrelerle tasarlanan model [8-10] neticesinde giderek gerekli hale gelmiştir. Hiperparametrelerin seçimi zor olduğu için bunu deneysel değerlere uyarlamak da zordur. Hiperparametrelerin ayarlanması, karmaşık ve dikkatle tasarlanmış yapıya sahip bir model için önem arz etmektedir. Yaygın kullanılan modellerde, araştırmacılar önceki çalışmalardan örnek alabileceğinden hiperparametreler elle ayarlanabilmektedir. Küçük ölçekli modeller için de hiperparametler elle ayarlanabilir. Fakat daha büyük model ya da yeni yayınlanan modeller için hiperparametrelerin bulunması araştırmacılar tarafından fazla miktarda deneme çalışması yapılabilmesi için çok fazla zaman ve hesaplama kaynağı gerektirmektedir.

HPO, uygulamada çeşitli zorluklarla karşı karşıya kalmaktadır. Büyük veri setlerinde ve büyük modellerde fonksiyon değerlendirmesinde maliyeti fazla olabilir. Yapılandırma alanı genellikle karmaşıktır ve yüksek boyutludur. Aynı zamanda bir algoritmanın hiperparametrelerinden hangilerinin optimize edileceği ve hangi aralıklarda olduğu her zaman net olmayabilir. Hiperparameterelerle ilgili olarak maliyet fonksiyonunun gradyanına erişim yoktur. Genel performans için eğitim veri setleri sınırlı sayıda olduğu için doğrudan optimizasyon yapılamaz.

Bu araştırmanın amacı, HPO için uygulanabilir algoritmalar üzerine analiz yapmak, HPO işlemi için önde gelen yapılar hakkında karşılaştırma yapmak ve derin öğrenme ağlarındaki HPO işlemi için bakış açısı sunmaktadır. Bu makalenin geri kalanı şu şekilde planlamıştır. Bölüm 2, sinir ağlarının oluşturulması ve eğitilmesi için; modeller, potansiyel arama alanları ve anahtar hiperparametrelerin tartışılmasıyla başlamaktadır. Bölüm 3'te hiperparametre aramada yaygın olarak kullanılan optimizasyon algoritmalarına odaklanılmaktadır. Bu bölümde ayrıca, farklı algoritma öğrenme modelleri için bu algoritmaların etkinliği ve uygulanabilirliği de değerlendirilmektedir. Bölüm 4, genel HPO kitlerine ve hizmetlerine genel bir bakış sunup; artılarını ve eksilerini karşılaştırmaktadır. Bölüm 5, mevcut HPO yöntemlerini daha kapsamlı bir şekilde karşılaştırmaktadır ve Bölüm 6, çalışmanın sonuçlarını vermektedir.

Bu çalışmanın katkısı şu şekilde özetlenir: (1) Hiperparametreler sistematik olarak yapıya ve eğitime ilişkin olarak kategorize edilmektedir ve deneysel stratejiler tartışılarak HPO'da hangi hiperparametrelerin bulunmasını belirlemede yardımcı olmaktadır. (2) HPO algoritmaları; doğrulukları, verimlilikleri ve uygulama kapsamlarına göre ayrıntılı olarak analiz edilmekte ve karşılaştırılmaktadır. (3) Bu çalışma HPO araçlarını karşılaştırmakta, açık kaynak ve kapalı kaynaklı hizmetler ile ilgili bilgiler

sunmaktadır. Bu araçların her biri için hedeflenen kullanıcıları açıklamaktadır.

Derin Sinir Ağları için Hiperparametreler

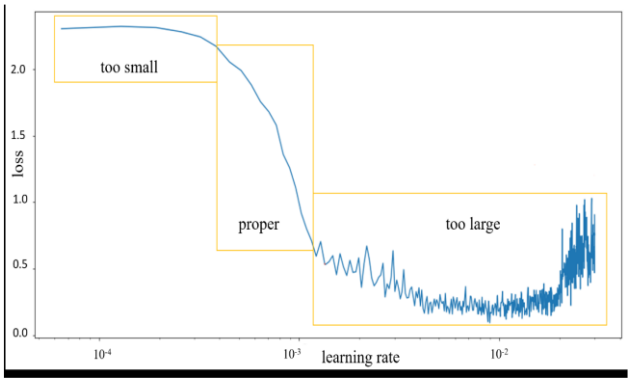
Gerekli hesaplama kaynakları göz önüne alındığında büyük öneme sahip hiperparametreler, HPO sürecinde öncelikli olarak tercih edilmektedir. Eğitim sırasında ağırlıklar üzerinde daha güçlü etkisi olan hiperparametreler sinir ağı eğitimi için önemlidir [11].

Hiperparametreler, model eğitimi için kullanılanlar ve model tasarımı için kullanılanlar olmak üzere iki gruba ayrılabilir. Model eğitimiyle ilgili uygun hiperparametre seçimi, sinir ağlarının daha hızlı öğrenmesini ve gelişmiş performans elde etmesini sağlamaktadır. Derin sinir ağını eğitmek için en çok benimsenen optimizasyon algoritmaları, momentum ile stokastik gradyan azalmasının[12] yanı sıra AdaGrad [13], RMSprop [14] ve Adam [15] gibi çeşitlerdir. Sinir ağının eğitim sürecinde yakınsama hızını belirledikleri için parça büyüklüğü (batch size) ve öğrenme oranı (learning rate, LR) en çok dikkat çekmektedir. Model tasarımı için kullanılan hiperparametreler daha çok sinir ağlarının yapısı ile alakalıdır. Bunun en tipik örneği gizli katmanların sayısı ve katmanların genişliğidir.

a. Öğrenme Oranı (Learning Rate, LR)

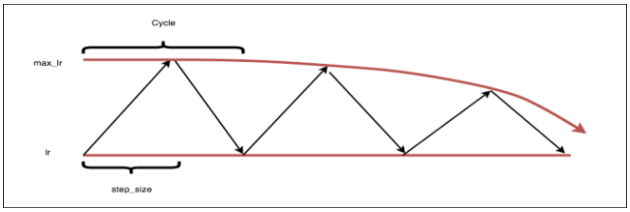
Öğrenme oranı, SGD sırasında adım uzunluğunu belirleyen pozitif bir skalerdir [16]. Çoğu durumda, Öğrenme oranı model eğitimi sırasında manuel olarak ayarlanmalıdır ve bu ayarlama genellikle yüksek doğruluk elde edebilmek için gerekmektedir [17]. Öğrenme oranınının eğitim süreci boyunca değişen öğrenme oranı olarak belirlenmesi bir alternatiftir. En basit program olan sabit öğrenme oranı, derin öğrenme çerçeveleri (ör. Keras) tarafından genellikle varsayılan olarak ayarlanmaktadır.

Pratikte, hiperparametrelerle ilgili deneyime sahip olmayanlar için hiperparametresinin etkisinin önemine karar vermesi zordur. Belirli bir hiperparametrenin etkisini sağlamak için bir duyarlılık testi önerilmektedir [18,19]. Hiperparametrelerin başlangıç değerleri etkili sonuçlara sebep olduğundan dikkatle belirlenmesi gerekmektedir. Eğitim esnasında bozulduğu için öğrenme oranının başlangıç değeri nispeten büyük bir değer olabilir. Eğitimin ilk safhalarında, büyük değerli bir öğrenme oranı daha az riskle hızlı yakınsamaya yol açmaktadır (Şekil 1). Öğrenme oranını güncellemek için log ölçeği kullanılırsa üstel bozulma için daha iyi bir seçim olabilir. Üstel bir fonksiyon, momentum ve ağırlık azalması gibi diğer birçok ayarlama hiperparametresi için uygulanabilir.



Şekil 1. *Öğrenme oranının etkisi [20]*

Öğrenme oranını düzenleme, optimizasyon algoritmasına göre ayarlanmalıdır [21]. Öğrenme oranı çizelgesinin önceden belirlenmesi gerekiyorsa, Öğrenme oranı çizelgesinin hiperparametrelerini ve karşılık gelen optimizasyon algoritmalarının eşzamanlı olarak ayarlanması önerilmektir. Kullanıcıların eğitimden önce programdaki tüm hiperparametreleri belirlemesi gerekmektedir. Bu durum, Öğrenme oranı yöntemi için sorun teşkil etmektedir. Bu sorun otomatik HPO veya döngüsel öğrenme oranı yöntemiyle çözülür [22]. Öğrenme oranı, belirli bir sınır değeri içinde bir üçgen kuralında güncellenmektedir ve sınır değeri, belirli bir döngüsel programda bozulmaktadır (Şekil 2).



Şekil 2. Döngüsel bir programda öğrenme oranı düşüşü [23]

b. Optimizasyon Algoritmaları

Optimizasyon algoritmaları veya optimize ediciler, doğruluk ve eğitim hızının iyileştirilmesinde kritik rol oynamaktadır. Optimizasyon algoritmalarıyla ilgili hiperparametreler; mini parça boyutu, momentum ve beta seçimini içermektedir. Uygun optimizasyon algoritması seçmek zor bir iştir. Bu bölümde yaygın olarak benimsenen optimizasyon algoritmaları (mini-parça gradyan azalması, RMSprop ve Adam), ilgili hiperparametreler ve önerilen değerler ele alınmaktadır.

Genel olarak, optimizasyon algoritmalarının öğrenme oranı ile birlikte ayarlanması gerekmektedir. Sinir ağı eğitimi yaparken kullanılan çoğu hiperparametre; optimizasyon algoritmaları ve öğrenme oranı ile yüksek oranda alakalıdır. Optimizasyon algoritmalarından RMSprop ve Adam benzer durumlarda uygulanabilmektedir. Çoğu durumda Adam, RMSprop yönteminden daha güvenilir sonuçlar elde ettiğinden varsayılan optimize algoritması olarak kullanılabilmektedir [24]. Adam, optimizasyon algoritması olarak kullanılırsa öğrenme oranı buna göre ayarlanmalıdır. Momentumlu SGD; RMSprop ve Adam ile karşılaştırıldığında, momentumlu SGD'nin optimum seviyeye gelebilmesi için daha fazla zaman gerekebilir.

Ng'nin 2017 ders notlarına [11] göre, hiperparametreleri önem derecesine göre şu şekilde sıralamıştır: (1) Öğrenme Oranı. (2)Momentum beta RMSprop vb. için. (3) Tüm mini-parça GD (gradient descent) için kritik olan mini parça boyutu. (4) Model yapısıyla ilgili hiperparametre olan gizli katmanların sayısı. (5) Öğrenme oranının bozulması. (6) Varyansı azaltmak ve aşırı eğilmekten kaçınmak için kullanılan düzenli lambda. (7) Doğrusal olmayan elemanlar eklemek için kullanılan aktivasyon fonksiyonları ve son olarak Adam, β1, β2 ve € şeklinde listelemiştir. Bu listede bulunan hiperparametrelerin sadece eğitim ile ilgili olmadığı, aynı zamanda modelin yapısını da belirlediği fark edilmektedir. Bu nedenle bir sonraki bölümde sadece gizli katmanların sayısı, düzenli lambda ve aktivasyon fonksiyonu hakkında konuşulmaktadır.

Model Tasarımına İlişkin Hiperparametreler

Gizli katmanların sayısı, sinir ağlarının genel yapısını belirlerken sonuç üzerinde doğrudan etkisi olan kritik bir parametredir [25]. Daha fazla katman ekleyerek sinir ağını büyütmek daha iyi sonuçlar elde etmek için uygun yöntemdir. Aynı zamanda, her katmandaki (n) nöronların sayısı da dikkatle değerlendirilmelidir. Gizli katmanlardaki nöron sayısının az olması, modelin karmaşıklığından yoksun olduğundan yetersizliğe (underfitting) neden olabilir [26]. Nöron sayısının çok fazla olması taşma (overfitting) ile sonuçlanabilir ve eğitim süresini uzatabilir. Heaton, 2017 nöron sayısının başlatılması için aşağıdaki formülü önermiştir [27] (denklem 1) :

$$n_{input} \leq n \leq n_{output}$$

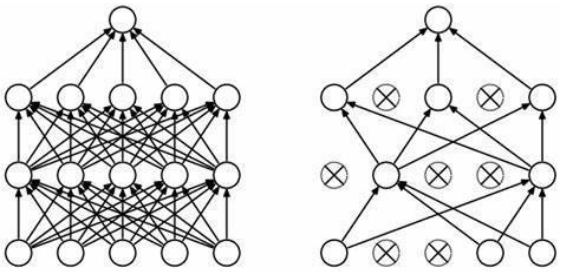
$$n = \frac{2}{3} n_{input} + n_{output} \quad (1)$$

$$n \leq n_{input}$$

burada n_{input}, giriş katmanı için nöron sayısı ve n_{output} , çıkış katmanı için nöron sayısıdır. Bir sinir ağı modelinin içerisine katman veya nöron eklemenin aksine, özellikle yetersiz eğitim verisine sahip olan sinir ağlarının karmaşıklığını azaltmak için düzenleme yani regülarizasyon uygulanmaktadır. Taşma durumunu önlemek için bir düzenleme terimi eklenir. Düzenleme teriminin genel ifadesi denklem 2'deki gibidir.

$$maliyet = kayıp\ fonksiyonu + düzenleme \quad (2)$$

Veri çoğaltma (data augmentation) [28] ve atlama (dropout) [29] yaygın olarak kullanılan regülarizasyon teknikleri olarak kullanılmaktadır. Veri çoğaltma yöntemi, taşmayı önlemek için eğitim veri kümesini sahte veriler oluşturmakta ve eklemektedir. Atlama durumu, eğitim sırasında kullanılmayan belirli bir olasılıkla nöronları rastgele seçmek için kullanılan bir tekniktir. Bu da ağı nöronların özgül ağırlıklarına daha az duyarlı hale getirmektedir [30]. Sonuç olarak, bir sinir ağının basitleştirilmiş versiyonu taşmayı azaltabilir (Şekil 3).



Şekil 3. Atlama (dropout) uygulanmadan önce ve sonra bir sinir ağının karşılaştırılması [29]

Aktivasyon fonksiyonları, derin öğrenme açısından nöronların çıkışına doğrusal olmayan özellikler kazandırdığı için önemlidir. Aktivasyon fonksiyonu olmadan sinir ağı, basitçe verilerin karmaşık özelliklerini temsil edemeyen doğrusal bir regresyon modeli olacaktır. Aktivasyon fonksiyonları, ağırlık gradyanlarını hesaplamak ve geri yayılım gerçekleştirmek için farklılaştırılabilir olmalıdır [31]. En popüler ve yaygın olarak kullanılan aktivasyon fonksiyonları sigmoid, hiperbolik tanjant (tanh), ReLU [32], Maxout [33] ve Swish [34]'tir. Fonksiyonların yapısı ve ilgili hiperparametreler dâhil olmak üzere uygun aktivasyon fonksiyonlarının araştırılmasında otomatik arama teknikleri uygulanmıştır [35].

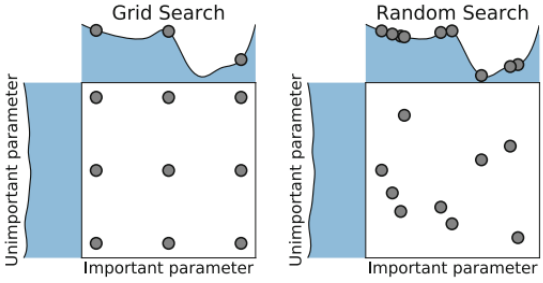
Hiperparametrede Arama Algoritmaları

Genel olarak arama algoritmaları olarak bilinen kara-kutu optimizasyon algoritmaları HPO'ya uygulanabilir. Bu bölümde arama algoritmaları olan modelsiz kara-kutu optimizasyon algoritmaları tartışıldıktan sonra Bayes optimizasyon algoritması açıklanmaktadır.

Modelsiz kara-kutu optimizasyon algoritmaları ızgara arama ve rastgele aramadır [36].

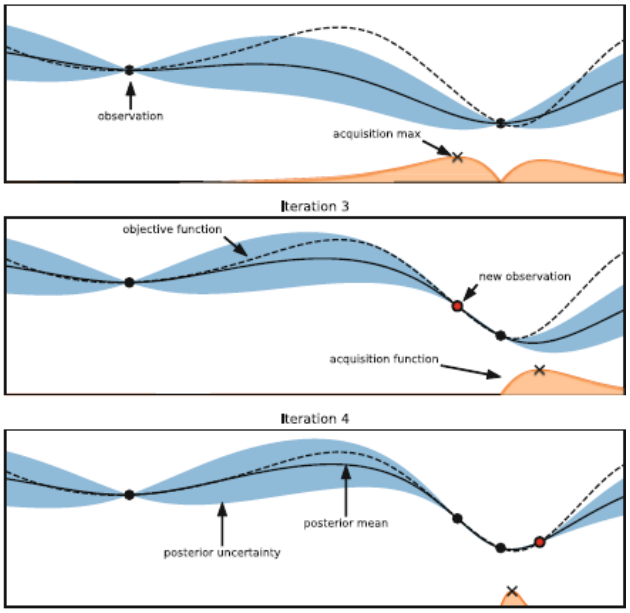
Izgara arama, tam faktöriyel tasarımı olarak da bilinen en temel HPO yöntemidir [37]. Daha kolay paralelleştirme yapması ve esnek kaynak tahsisi bakımından avantajlıdır. Izgara arama, yeterli kaynaklar verildiği sürece en doğru tahminlere götüren en basit arama algoritmasıdır ve kullanıcı her zaman en uygun kombinasyonu bulabilmektedir [38]. Diğer arama algoritmaları daha elverişli özelliklere sahip olsa da matematiksel sadeliği nedeniyle ızgara arama hala yaygın kullanılan bir yöntemdir [36].

Rasgele arama [36], ızgara aramasını temel alan bir yöntemdir. Arama için belirli bir bütçe tükenene kadar yapılandırma işlemini rastgele tekrarlamaktadır. Makine öğrenme algoritmasını optimize ederken yeterli kaynak sağlandığına dair varsayımda bulunmayıp beklentiye göre rastgele biçimde optimum seviyeye yakın bir performans elde ettiği için rastgele arama, ızgara arama yönteminden daha iyi sonuçlar vermektedir [37]. Şekil 4'de ızgara arama ve rastgele arama yöntemleri karşılaştırılmıştır ve rastgele aramanın daha iyi sonuç verdiği gözlemlenmiştir [36]. HPO'nun ilk aşamasında arama alanını hızla daraltmak için rastgele arama kullanılması önerilmektedir [11].



Şekil 4. Önemli ve önemsiz parametre ile bir fonksiyonu en aza indirmek için ızgara arama ile rastgele aramanın karşılaştırılması [36]

Bayes optimizasyonu (BO), onlarca yıllık tarihi olan geleneksel bir algoritmadır. Mockus [39,40] tarafından geliştirilmiştir ve daha sonra küresel optimizasyon problemine uygulandığında popülerite kazanmıştır [41]. BO, neredeyse tüm küresel optimizasyon türleri için tipik bir yöntemdir ve daha fazla veriyle daha az hata yapmayı amaçlamaktadır [42]. Bayes optimizasyonundaki birçok gelişme yeni kazanım fonksiyonlarında, modellerde ve paralelleştirme şemalarında HPO'ya kolayca uygulanabilmektedir. Kullanıcıların hiperparametrelerin dağılımı hakkında ön bilgiye sahip olmaları gerekmemesi ve arkasında olasılık fikrine dayanmasından dolayı rastgele arama ve ızgara arama yöntemlerinden daha verimli sonuçlar elde etmektedir. BO algoritmasında [43] birçok olasılık modeli kullanılabilir, ancak Gauss Süreci (Gauss Process, GP) [44] çoğunlukla tercih edilmektedir. GP, Bayes optimizasyonunda amaç fonksiyonları için varsayılan vekil model seçimi olabilir (Şekil 5).



Şekil 5. Tek boyutlu fonksiyon üzerinde Bayes optimizasyonunun gösterimi [45]

GP, tahminlerde belirsizliğin ölçülmesini sağlayan BO için tercih edilen modeldir. Parametrik olmayan bir modeldir ve parametre sayısı sadece giriş noktalarına bağlıdır. Bununla birlikte, GP'nin bazı dezavantajları da vardır. Örneğin, BO teorisiyle birlikte anlaşılması kavramsal olarak zordur. Ayrıca, yüksek boyutlu veya çok sayıda veri noktası ile zayıf ölçeklenebilirliği önemli bir sorundur [46]. Dahası, belirli bir miktarda veri için önceden seçim yapmak performansı büyük ölçüde etkilemektedir.

Bayes optimizasyonu için bir diğer alternatif model rastgele ormanlar (random forest) modelidir [47]. GP'ler küçük, sayısal yapılandırma alanlarında rastgele ormanlardan daha iyi performans gösterirken [43], rastgele ormanlar doğal olarak standart GP'lerin iyi çalışmadığı daha büyük, kategorik ve koşullu yapılandırma alanlarında iyi performans göstermektedir [43,48]. Ayrıca hesaplama karmaşıklığı rastgele ormanlar için çok daha iyi ölçeklenmektedir. Bu avantajlar göz önünde bulundurulduğunda rastgele ormanlarla [47] Bayes optimizasyonu için SMAC ve AutoML araçları olan Auto-WEKA[49] ve Auto-Sklearn [50] etkinleştirilmiştir.

TPE (Tree Parzen Estimator, Parzen Tahmincisi Ağacı), koşullu hiperparametreler için bir Parzen tahmincisi ağacı kullanmaktadır ve yapılandırılmış HPO görevleri üzerinde iyi performans göstermiştir [51,43,52,49,53] Kavramsal olarak basittir ve paraleldir [54].

Freeze-Thaw Bayesian Optimizasyonu [55], öğrenme eğrilerinin Bayesian optimizasyonunun modelleme ve seçim sürecine tam entegrasyonudur. Freeze-Thaw, düzenli bir Gauss işlemiyle birleştirilmiş bir algoritmanın performansını modellemektedir ve öğrenme eğrilerini öğrenme başına eğri Gauss işlemleriyle modellemek için katlanarak bozulan işlevlere karşılık gelen özel bir kovaryans işlevi sunmaktadır.

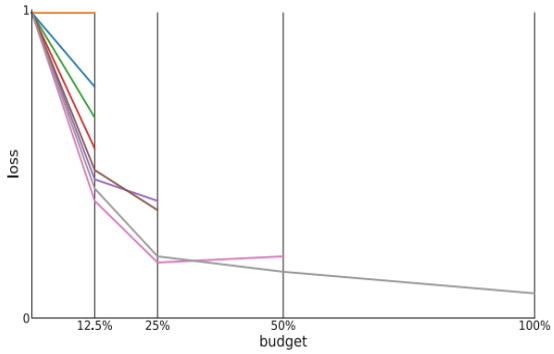
Hiperparametrede Erken Durma Strateji ile Optimizasyon

HPO, genellikle hesaplama maliyeti yüksek olan ve zaman alan işlemdir. Gerçek zamanda, HPO sürecini mevcut sınırlı kaynaklarla tasarlamak gerekmektedir. HPO için erken durma stratejileri, sinir ağı eğitimindekilere benzemektedir. Ancak erken durma stratejisi sinir ağı eğitiminde taşmayı önlemek için uygulanmaktadır. Bu, kullanıcıların tüm eğitimi bitirmeden önce denemeyi sonlandırmasına izin verir, böylece umut verici hiperparametre seti ile denemeler için hesaplama kaynaklarını serbest bırakır.

Medyan durdurma, Google Vizier [56], Tune [57] ve NNI [58] gibi yapılandırılmış HPO araçları tarafından uyarlanan en basit ilkel yöntemdir. Modelsizdir ve çok çeşitli performans eğrileri için geçerlidir. Medyan durdurma, önceki çalıştırmaların raporladığı birincil metriklerin (örn. doğruluk veya maliyet) ortalamasını esas alarak sonuca varmaktadır.

Eğri uydurma, bir LPA (öğrenme, tahmin, değerlendirme / learning, predicting, assessing) algoritmasıdır [3,59]. Google Vizier ve NNI tarafından desteklenen, bahsedilen arama algoritmaları kombinasyonlarında geçerlidir. Genel olarak, sistemin durup durmayacağını belirlemek için öğrenme, tahmin ve değerlendirme olmak üzere üç adıma sahiptir. Medyan durdurma yöntemi ile karşılaştırıldığında, eğri uydurma yöntemi parametreli bir modeldir. Modelin oluşturulması bir eğitim sürecidir. Arama algoritmaları ile birleştirildiğinde, optimizasyon sürecini

hızlandırır ve daha sonra son teknoloji ürünü bir ağ bulur.



Şekil 6. Sekiz algoritma-yapılandırma için ardışıl yarıya bölmenin gösterimi [60]

Özellikle derin öğrenme algoritmalarını optimize etmek için güçlü performans gösterdiklerinden, ardışıl yarıya bölme ve Hiperband'a dayanan haydut tabanlı yöntemlerin varyantlarına odaklanılmaktadır. Ardışıl yarıya bölme (Successive halving, SHA) ve HyperBand, örnekleme yöntemi olarak rastgele arama ve haydut tabanlı erken durma stratejisi ile HPO için kaynak tasarrufunda geleneksel arama algoritmalarından daha iyi performans göstermektedir. Ardışıl yarıya bölme, uygun algoritma seçiminde son derece basit bir o kadar da güçlüdür. Bu nedenle popüler bir stratejidir. Ardışıl yarıya bölme işlemi, Şekil 6'da gösterilmektedir. HyperBand (HB) [61], SHA'nın bir uzantısıdır. HB, kolaylıkla paralel olarak kullanılabilmektedir, çünkü tüm uygulamalar rastgele örneklenmekte ve bağımsız olarak çalışmaktadır. HB, SHA'dan devraldığı erken durma ve kaynakların makul seviye uyarlayabilmesi ile rastgele aramayı hızlandırmaktadır. Rasgele arama ve BO ile karşılaştırıldığında, HB özellikle derin sinir ağları için stokastik gradyan azalması (SGD) durumunda daha az kaynakla üstün doğruluk göstermektedir.

Asenkron SHA (ASHA) [43] HB'yi paralelleştirmek için gelişmiş bir dağıtım şeması önermektedir ve eliminasyon sırasında bozulma sorunlarını önlemektedir.

Bayesian Optimizasyonu ile HyperBand (BOHB) [62], BO ve HB'nin kombinasyonudur ve basit rastgele arama yerine yönlendirilmiş örnekleme yöntemi sunmaktadır. Genel olarak BOHB, birden fazla birimde uygulanması kolay, sağlam ve hesaplamalı olarak etkili bir HPO yöntemidir.

Populasyon tabanlı yöntemler, esasen evrimsel algoritmalar [63]; parçacık sürüsü optimizasyonu ve kovaryans matris adaptasyonu evrim stratejisi [64] gibi genetik algoritmalara [65] dayanan rastgele arama yöntemleri serisidir. Bu yöntemler bir populasyonu koruyan optimizasyon algoritmalarıdır, yani bir yeni nesil oluşturulurken daha iyi yapılandırma elde etmek için mutasyon ve çaprazlama uygulayarak populasyonu iyileştirmektedirler. Bu yöntemler N üyeden oluşan bir populasyonu N makinede paralel olarak değerlendirilebildikleri için kavramsal olarak basittir. Farklı veri türlerini işleyebilir ve paraleldir [54].

Sürü zekâsı optimizasyon algoritmaları; ateşböceği algoritması [66], ateşböceği sürü optimizasyonu [67], karınca koloni optimizasyonu [68], parçacık sürü optimizasyonu [69], yapay balık sürüsü algoritması [70], yapay arı koloni algoritması [71] gibi yöntemleri incelemektedir. Bu yöntemlerin temel düşüncesi kuş, balık, kedi, karınca ve arı gibi birbirleri ile bağlantılı olarak hareket edebilen canlıların hareketlerinin incelenmesiyle geliştirilmiştir. Sürüdeki canlıların karşılıklı etikleşimleri yenilemeli şekilde durma koşuluna yani en iyi sonuç elde edilene kadar tekrarlanır ve böylece çözüm kalitesi artırılır. Kavramsal olarak sürüde bulunan üyeler senkronize hareket ettiğinden basittir.

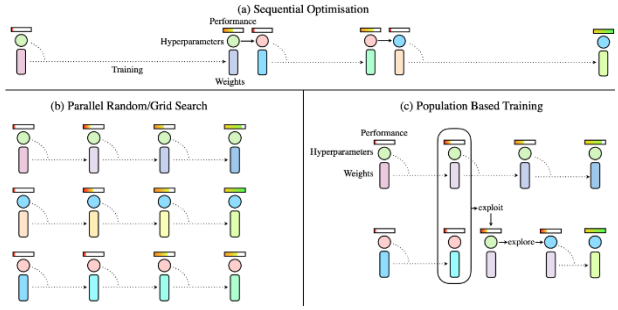
Bahsedilen popülasyon tabanlı yöntemler ve sürü zekası optimizasyon algoritmaları literatürde hala geniş bir alan oluşturmaktadır.

En iyi bilinen populasyon tabanlı yöntemlerden biri olan kovaryans matris adaptasyonu evrim stratejisidir (CMA-ES [64]). Bu basit evrimsel strateji; ortalama ve kovaryansı, populasyonun bireylerinin başarısına dayalı olarak her nesilde güncellenen çok değişkenli bir Gaussdan yapılan yapılandırmaları örneklemektedir. CMA-ES, düzenli olarak Kara-Kutu Optimizasyon Kıyaslaması (Black-Box Optimization Benchmarking, BBOB)'na hakim olan en çok tartışılan kara-kutu optimizasyon algoritmalarından biridir.

En yaygın kullanılan populasyon tabanlı yöntemlerden biri DeepMind tarafından önerilen populasyon tabanlı eğitimdir (Population-Based Training, PTE) [72, 73]. PTE, kavramsal olarak

basit ancak hesaplama açısından verimli bir yöntemdir. İki açıdan benzersiz bir yöntemdir: eğitim sırasında uyarlanabilir hiperparametrelere izin verir ve paralel arama ile sıralı optimizasyonu birleştirir. Bu özellikler PTE'yi derin öğrenme ağlarındaki HPO için özellikle uygun hale getirmektedir.

PTE işlemi basitçe genel algoritmalara benzer olarak tanımlanabilir. İlk olarak, farklı hiperparametre ayarlarına sahip bir çalışma populasyonu başlatılır (Şekil 7). Şekil 7'de PTE'nin aşamalarına yer verilmiştir. PTE, yakınsama için beklemek yerine sıcak bir başlangıç kullanarak HPO'yu düzenli eğitime dâhil etmenin yolunu sunmaktadır. Bu yaklaşım, üretken çekişmeli ağ (Generative Adversarial Network, GAN) [74] ve transformatör ağı [75] gibi büyük modeller için anlamlı bir yaklaşımdır.



Şekil 7. Sıralı ve paralel optimizasyonun kombinasyonu olarak PTE [65]

PTE'nin gelecekte hala keşfedilecek bazı dezavantajları vardır. Örneğin, bu yöntemin optimal hiperparametre kümesini elde ettiği teorik olarak kanıtlanmamıştır. PTE'nin mevcut hesaplama kaynaklarıyla en iyi yapılandırmayı sağladığı gözlemlenmektedir, ancak optimum yapılandırma olup olmadığı tartışmaya açıktır. PTE hala ileri evrim ve mutasyon kararlarıyla genişletilemez. Ayrıca hiperparametrelerin tanımı ve hesaplama grafiğinde yapılan değişiklikler karmaşıktır.

Pratik uygulamalarda kullanıcıların, her zaman hangi hiperparametrelerin dikkate alması gerektiğine, arama algoritmalarını uygulama yoluna ve derin öğrenme modellerini eğitme sürecine karar vermeleri gerekmektedir. Bu durumda, bahsedilen yapılandırmaları gerçekleştirmek için bir araç gereklidir. Bu gereklilikten dolayı sonraki bölümde hiperparametre optimizasyonu için kullanılan kitlere yer verilmektedir.

Hiperparametre Optimizasyonu için Kullanılan Kitler

Hiperparametre eğitimi, özellikle derin öğrenme ağları için zaman alıcı bir süreçtir. Tüm süreci bitirmek onlarca yıl GPU işlemi alabilir, çünkü tek bir sinir ağını yakınsaması için eğitmek bile neredeyse bir gün sürmektedir. HPO için araç kiti, ağ eğitimi ve hiperparametre ayarlama arasında köprü kurmaktadır. Bu bölüm, güncel açık ve kapalı kaynaklı araçların hizmetlerinin genel karşılaştırmasını sağlamaktadır.

Genel olarak, HPO için bulut bilgi işlem kaynaklarına dayanan ve her biri bu alt bölümde araçlar ve hizmetler olmak üzere için iki tür kit vardır. Bazı kitler önerilen algoritmanın uygulanması için tasarlanmıştır.

HpBandSter, BOHB'nin uygulanmasında da bulunmaktadır, ancak rastgele arama ve HB de sağlamaktadır. Xcessive, başlangıç dostu olan etkileşimli bir grafik kullanıcı arabirimi (GUI) ile diğer kitaplıklardan daha iyi performans göstermektedir. Kullanıcılar GUI aracılığıyla modellerin eğitimini, optimizasyonunu ve değerlendirmesini yönetebilmektedirler. Bununla birlikte, Xcessive yalnızca Bayes yöntemleri aracılığıyla otomatik hiperparametre aramayı da desteklemektedir. Derin öğrenme ağlarını eğitmek için de elverişsizdir.

Scikit-Optimize, daha sonraki birçok çerçeve tarafından belirtilen kapsamlı kütüphanedir. Bayesian yöntemleri, rastgele arama, rastgele orman ve diğer bazı güvenilir optimizasyon algoritmalarını içermektedir. Yukarıda bahsedilen tüm yöntemlerin ortak yanları; arama algoritmalarına uygulanması, derin öğrenme eğitim çerçeveleri için destekleri ve denemeler için uygulama programı olmasıdır.

Ayrıca Google Cloud ve Amazon Web Services (AWS), model seçimi ve HPO sunmaktadır. Bu hizmetler, binlerce paralel eğitim süreci ve değerlendirmesi için geçerli olan büyük hesaplama kaynakları tarafından desteklenmektedir. Ancak, bunlar kapalı kaynaklı bir altyapıdır ve kullanıcıların hizmet için ödeme yapması gerekmektedir. HPO kitleri; kullanım

kolaylığı, son teknoloji, kullanılabilirlik, ölçeklenebilirlik ve esneklik hususlarını karşılamak üzere tasarlanmıştır [56].

Google Vizier [56], Googles Cloud Machine Learning alt sistemine dayalı kara-kutu optimizasyonu için bir kütüphaneden çok ölçeklenebilir hizmettir. En göze çarpan avantajı, kullanım kolaylığıdır. Google Vizier altyapıda kapalı kaynak olmasına rağmen, HPO için yeni algoritmalar değiştirmek veya tasarlamak kolaydır. Genel olarak, Vizier kapsamlı bir HPO hizmetinin öncüsüdür.

Advisor, Google Vizier'in ve Google Cloud'un desteği olmadan tüm Viziers özelliklerini gerçekleştirmeye çalışan açık kaynaklı bir sürümüdür. Izgara arama, rastgele arama ve BO da erken durma stratejileri Advisor üzerinde desteklenmektedir.

Otomatik model ayarlama, model oluşturma ve dağıtım sürecini basitleştirmek için bir makine öğrenme ortamı olan Amazon SageMaker'ın bir modülüdür. Amazon Web Services (AWS) desteğiyle Google Vizier'e benzemektedir. En önemli özelliği model oluşturma, eğitme ve ayarlama için farklı araçlara geçmeye gerek duymamasıdır.

Sinir Ağı Zekâsı (Neural Network Intelligence, NNI), hem otomatik makine öğrenimi (AutoML) hem de Microsoft tarafından yayınlanan HPO için açık kaynaklı araç setidir. NNI, Google Vizier ve SageMaker'dan daha fazla arama algoritması uygulamakta ve yeni bir algoritma yazmak için arayüzde genişletilebilmektedir. Bir kullanıcı farklı algoritmaların verimliliğini keşfetmek isterse, NNI uygun bir seçim olacaktır. Ayrıca, NNI yüksek genişletilebilirlik ile tasarlanmıştır. Yani kullanıcılar yeni tasarladıkları algoritmaları test edebilirler. Ayrıca NNI; PyTorch, Keras, TensorFlow, MXNet, Scikit-learn ve XGBoost gibi en popüler derin öğrenme araçları ve kütüphaneleri ile uyumludur. NNI içerisinde yeni platformlar için genişletilebilir noktalar da mevcuttur. Google Vizier ile karşılaştırıldığında, NNI yalnızca etkileşimli işlevi olmayan deneylerin durumunu ve sonuçlarını görüntülemektedir. Dezavantajı, hiperparametre ayarı, algoritma seçimi veya kaynak yapılandırması Web Arayüzünden (Web UI) ayarlanamaz.

Ray.Tune, Berkeley RISELab tarafından geliştirilen Ray kütüphanesidir. Model eğitim için dağıtılmış bir yapı olan Ray, hesaplama kaynaklarının verimli tahsisini garanti etmektedir. Tune'da uygulanan çoğu arama yöntemi paralel olarak gerçekleştirilebilmektedir. Açık kaynak olan Tune, birçok uygulamada NNI ile benzerlikleri paylaşmaktadır.

5. Tartışmalar ve Karşılaştırma

Bu bölüm, hesaplama kaynaklarının kaydedilmesindeki avantajları, doğruluk ve verimlilikteki dezavantajları ve uygulanabilirlikleri de dahil olmak üzere ana algoritmalar arasında kısa bir karşılaştırma yapmaktadır.

HPO algoritmalarının avantajları sıralanmak istenirse; ızgara arama, rastgele arama ve PTE yöntemlerinde paralellik mevcuttur. Izgara arama basit bir yöntemdir. Rastgele aramanın erken durma stratejileri ile birleştirilmesi kolaydır. Bayes optimizasyonu güvenilir ve umut vericidir. Aynı zamanda diğer birçok algoritmanın temelini oluşturur. Çok bantlı yöntemler kavramsal olarak basit, hesaplama açısından verrimlidir.PTE yöntemi HPO ve model eğitimini birleştirmektedir. Başlıca hiperparametre optimizasyonları karşılaştırması Tablo 1'de yapılmıştır.

Izgara arama yönteminde boyutsallık sorunu mevcuttur. Rastgele arama yöntemi düşük verimliliğe sahiptir ve optimum ulaşamayabilir. Bayes optimizasyonu yöntemi paralellik için zor ve kavramsal olarak karmaşıktır. Çok bantlı yöntemlerin bütçe ile deneme sayısı arasında denge durumu sıkıntı teşkil etmektedir.PTE yönteminde hesaplama grafiğinde sürekli değişiklikler mevcuttur ve gelişmiş evrim yöntemi ile genişletilemez.

6. Sonuç

Derin sinir ağlarının artan uygulamaları ile bu çalışmaya yönelim de artmaktadır. Araştırma, eğitim ve yapı için önemli hiperparametrelerin tanıtılmasıyla başlamış ve daha sonra otomatik

Tablo 1. Başlıca HPO Algoritmaları Karşılaştırılması

	Avantaj	Dezavantaj	DNN için uygulanabilirlik
Izgara arama	- Basit. - Paralellik mevcut.	- Boyutsallık Sorunu.	-Sadece birkaç HP ayarlanacaksa uygulanabilir
Rastgele arama	- Paralellik mevcut. -Erken durma stratejileri ile birleştirilmesi kolaydır.	- Düşük verimlilik. -Optimuma ulaşamayabilir.	- İlk aşamalar için uygun
Bayes Optimizasyonu	- Güvenilir ve umut vericidir. - Diğer birçok algoritmanın temelini oluşturur.	- Paralellik için zor - Kavramsal olarak karmaşıktır.	-Araçlar için varsayılan algoritma. -BO varyantları daha uygulanabilir olabilir (TPE).
Çok Bantlı Yöntemler	- Kavramsal olarak basit. -Hesaplama açısından verimlidir.	- Bütçe ile deneme sayısı arasındaki denge durumu.	-Varsayılan bir seçim olabilir. -Açık kaynaklı kütüphaneler tarafından uygulanmaktadır.
PBT Metodu	- HPO ve model eğitimini birleştirir. - Paralellik mevcut.	-Hesaplama grafiğinde sürekli değişiklikler olması. -Gelişmiş evrim ile genişletilemez.	-Hesaplamalı pahalı modeller için uygulanabilir.

HPO için en gelişmiş arama algoritmalarına ve programlayıcılarına genişletilmiştir. İlgili algoritmaları özetlemenin yanı sıra, artıları, eksileri ve uygulamaları da dâhil olmak üzere HPO için kullanılan araçlarından bahsedilmiştir. Algoritmaların ve değerlendirme yöntemlerinin karşılaştırılması, büyük modeller için fizibiliteleri ve hesaplama kaynaklarının tüketimi açısından tartışılmıştır. Bu çalışma araştırmacılar ve kullanıcılar için referans olarak HPO hakkındaki bilgileri özetlemeye yöneliktir.

Kaynaklar

[1] Sculley, D., Snoek, J., Wiltschko, A., and Rahimi, A., (2018).Winner's curse? On Pace, Progress, and Empirical Rigor, In: *International Conference on Learning Representations Workshop track, published online: iclr.cc*

[2] King, R. D., Feng, C., & Sutherland, A. (1995). Statlog: comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence an International Journal*, **9(3)**, 289-333.

[3] Kohavi, R., & John, G. H. (1995). Automatic parameter selection by minimizing estimated error. In *Machine Learning Proceedings 1995* (pp. 304-312). Morgan Kaufmann.

[4] Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). Machine learning. *Neural and Statistical Classification*, ***13*(1994)**, 1-298.

[5] Ripley, B. D. (1993). Statistical aspects of neural networks. *Networks and chaos—statistical and probabilistic aspects*, ***50***, 40-123.

[6] Rodriguez, J. (2018). Understanding Hyperparameters Optimization in Deep Learning Models: Concepts and Tools.

[7] Tan, M., & Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*.

[8] Ma, N., Zhang, X., Zheng, H. T., & Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 116-131).

[9] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4510-4520).

[10] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2820-2828)..

[11] Ng, A. (2017). Improving deep neural networks: Hyperparameter tuning, regularization and optimization. *Deeplearning. ai on Coursera*.

[12] Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, 400-407.

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

[14] Hinton, G., Srivastava, N., & Swersky, K. (2012). Neural networks for machine learning. *Coursera, video lectures*, ***264*(1)**.

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

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

[17] Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade* (pp. 437-478). Springer, Berlin, Heidelberg.

[18] Hamby, D. M. (1994). A review of techniques for parameter sensitivity analysis of environmental models. *Environmental monitoring and assessment*, ***32*(2)**, 135-154.

[19] Breierova, L., & Choudhari, M. (1996). An introduction to sensitivity analysis. Massachusetts Institute of Technology.

[20] https://www.jeremyjordan.me/, 10 Mayıs 2020.

[21] Lau, S. (2017). Learning rate schedules and adaptive learning rate methods for deep learning. *Towards Data Science*.

[22] Smith, L. N. (2017, March). Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 464-472). IEEE.

[23] https://github.com/bckenstler/CLR. 14 Mayıs 2020.

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

[25] Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, ***18*(7)**, 1527-1554.

[26] Yu, T., & Zhu, H. (2020). Hyper-Parameter Optimization: A Review of Algorithms and Applications. *arXiv preprint arXiv:2003.05689*.

[27] Heaton, J. (2008). The number of hidden layers. *Heaton Research Inc*.

[28] Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

[29] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, ***15*(1)**, 1929-1958.

[30] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[31] Walia, A. S. (2017). Activation functions and it's types-Which is better?. *Towards Data Science*, *29*.

[32] Nair, V., and Hinton, G. E., (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* pp. 807-814.

[33] Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., & Bengio, Y. (2013, February). Maxout networks. In *International conference on machine learning* (pp. 1319-1327).

[34] Ramachandran, P., Zoph, B., and Le, Q. V., (2017). Swish: a Self-gated Activation Function. *arXiv preprint arXiv:1710.05941*, **7**.

[35] Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

[36] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, ***13*(1)**, 281-305.

[37] Montgomery, D. C. (2017). *Design and analysis of experiments*. John wiley & sons.

[38] Joseph, R. (2018). Grid Search for model tuning.

[39] Močkus, J. (1975). On Bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference* (pp. 400-404). Springer, Berlin, Heidelberg.

[40] Mockus, J., Tiesis, V., & Zilinskas, A. (1978). The application of Bayesian methods for seeking the extremum. *Towards global optimization*, ***2***(117-129), 2.

[41] Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, ***13*(4)**, 455-492.

[42] Yu, T., & Zhu, H. (2020). Hyper-Parameter Optimization: A Review of Algorithms and Applications. *arXiv preprint arXiv:2003.05689*.

[43] Eggensperger, K., Feurer, M., Hutter, F., Bergstra, J., Snoek, J., Hoos, H., & Leyton-Brown, K. (2013, December). Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice* (Vol. 10, p. 3).

[44] Rasmussen, C. E. (2003, February). Gaussian processes in machine learning. In *Summer School on Machine Learning* (pp. 63-71). Springer, Berlin, Heidelberg.

[45] Hutter, F., Kotthoff, L., and Vanschoren, J., (2019). Automated Machine Learning. Springer: New York, NY, USA.

[46] Feurer, M., & Hutter, F. (2019). Hyperparameter optimization. In *Automated Machine Learning* (pp. 3-33). Springer, Cham.

[47] Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011, January). Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization* (pp. 507-523). Springer, Berlin, Heidelberg.

[48] Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Hardt, M., Recht, B., & Talwalkar, A. (2018). Massively parallel hyperparameter tuning. *arXiv preprint arXiv:1810.05934*.

[49] Thornton, C., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2013, August). Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 847-855).

[50] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. In *Advances in neural information processing systems* (pp. 2962-2970).

[51] Bergstra, J. S., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems* (pp. 2546-2554).

[52] Sparks, E. R., Talwalkar, A., Haas, D., Franklin, M. J., Jordan, M. I., & Kraska, T. (2015, August). Automating model search for large scale machine learning. In *Proceedings of the Sixth ACM Symposium on Cloud Computing* (pp. 368-380).

[53] Zhang, Y., Bahadori, M. T., Su, H., & Sun, J. (2016, August). FLASH: fast Bayesian optimization for data analytic pipelines. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 2065-2074).

[54] Loshchilov, I., & Hutter, F. (2016). CMA-ES for hyperparameter optimization of deep neural networks. *arXiv preprint arXiv:1604.07269*.

[55] Swersky, K., Snoek, J., & Adams, R. P. (2014). Freeze-thaw Bayesian optimization. *arXiv preprint arXiv:1406.3896*.

[56] Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., & Sculley, D. (2017, August). Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1487-1495).

[57] Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J. E., & Stoica, I. (2018). Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*.

[58] Microsoft., (2018), Neural Network Intelligence. *https://github.com/microsoft/nni#nni-released-reminder*.

[59] Provost, F., Jensen, D., & Oates, T. (1999, August). Efficient progressive sampling. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 23-32).

[60] Bissuel, A. (2020). Hyper-parameter Optimization Algorithms: A Short Review. *https://www.automl.org/blog_bohb/*.

[61] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, ***18*(1)**, 6765-6816.

[62] Falkner, S., Klein, A., & Hutter, F. (2018). BOHB: Robust and efficient hyperparameter optimization at scale. *arXiv preprint arXiv:1807.01774*.

[63] Simon, D. (2013). *Evolutionary optimization algorithms*. John Wiley & Sons.

[64] Hansen, N. (2016). The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*.

[65] Shiffman, D., Fry, S., and Marsh, Z., (2012). The Nature of Code. (pp. 323-330).

[66] Yang, X. S. (2009). Firefly Algorithms Formultimodal Optimization, Proceedings of the Stochastic Algorithms: Foundations and Applications, *Lecture Notes in Computing Sciences*,. Springer, Sapporo, Japan. vol. 5792. (pp. 178-178).

[67] Krishnanand, K.N., Ghose, D. (2005). Detection of Multiple Source Locations Using a Glowworm

Metaphor with Applications to Collective Robotics. In *IEEE Swarm Intelligence Symposium.* (pp. 84-91).

[68] Dorigo M., Maniezzo, V., Colorni, A.. (1991) The Ant System: An Autocatalytic Optimizing Process. Tech. Rep. No. 91- 016. Dipartimento di Elettronica, Politecnico di Milano, Italy.

[69] Kennedy, J., Eberhart, R. C. (1995). Particle Swarm Optimization. *IEEE International Conference on Neural Networks*, vol. IV, Piscataway, NJ. (pp. 1942-1948).

[70] Jiang, M., Yuan, D., Cheng, Y..(2009). Improved Artificial Fish Swarm Algorithm. In *Fifth International Conference on Natural Computation*. (pp. 281-285).

[71] Karaboga, D., Akay, B.. (2009). A Survey: Algorithms Simulating Bee Swarm Intelligence. *Artificial Intelligence Review*, vol. 31 no., 1-4, (pp. 61-85).

[72] Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., ... & Fernando, C. (2017). Population based training of neural networks. *arXiv preprint arXiv:1711.09846*.

[73] Li, A., Spyra, O., Perel, S., Dalibard, V., Jaderberg, M., Gu, C., ... & Gupta, P. (2019, July). A generalized framework for population based training. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 1791-1799).

[74] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672-2680).

[75] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).