BŞEÜ Fen Bilimleri Dergisi
8(1), 446-457, 2021

BSEU Journal of Science
https://doi.org/10.35193/bseufbd.884109

*e-ISSN: 2458-7575 (https://dergipark.org.tr/tr/pub/bseufbd)*

*Araştırma Makalesi - Research Article*

# Plaka Bölgesi Tespiti Problemi için Yapay Arı Koloni Algoritması ile YSA Eğitiminin APKD'de Gerçeklenmesi

# Implementation of ANN Training with Artificial Bee Colony Algorithm for Plate Region Detection Problem on FPGA

Mehmet Ali Çavuşlu[1*]

***Geliş / Received: 22/02/2021*** ***Revize / Revised: 15/05/2021*** ***Kabul / Accepted: 31/05/2021***

**ÖZ**

Son zamanlarda Yapay Sinir Ağı (YSA) eğitimlerinde türev bilgisi gerektiren algoritmalara alternatif olarak küresel arama özelliğine sahip evrimsel algoritmalar sıklıkla kullanılmaktadır. Bu çalışmada YSA eğitimi, evrimsel algoritmalardan Yapay Arı Koloni (YAK) algoritması ile Alan Programlanabilir Kapı Dizileri (APKD) üzerinde donanımsal gerçekleştirilmiştir. APKD tabanlı gerçeklemede sayı formatı ve aktivasyon fonksiyonu yaklaşımı maliyet, hız ve hata duyarlılığı açısından önem arz etmektedir. Çalışmada yüksek hassasiyet ve dinamiklik özelliklerine sahip IEEE 754 kayan noktalı sayı formatı seçilmiştir. Üssel fonksiyonun donanımsal gerçeklenmesinin zor olması nedeni ile aktivasyon fonksiyonunun donanımsal gerçeklenmesinde matematiksel yaklaşım kullanılmıştır. Çalışmada araç plaka bölgesi tespiti probleminin çözümüne yönelik YSA mimarisi tasarlanmış ve YAK algoritması ile APKD üzerinde eğitilmiştir. Eğitilen ağın test verilerindeki %98.82 başarımı, APDK üzerinde eğitilen YSA'nın iyi bir genelleme yaptığını ve sentezleme sonuçları, uygulamanın APDK'da sadece %9'luk alan tüketimi ile gerçekleştirilebildiğini göstermiştir.

***Anahtar Kelimeler- FPGA, YSA, YAK, Kayan Noktalı Sayılar, Plaka Bölge Tespiti***

**ABSTRACT**

Recently, evolutionary algorithms with global search feature are frequently used as an alternative to algorithms that require derivative knowledge in Artificial Neural Network (ANN) trainings. In this study, ANN training was carried out on Field Programmable Gate Arrays (FPGA) with the Artificial Bee Colony (ABC) algorithm, one of the evolutionary algorithms. Number format and activation function approach is important in terms of cost, speed and error sensitivity in FPGA-based implementation. In the study, IEEE 754 floating point number format, which has high sensitivity and dynamism features, was chosen. Since the hardware implementation of the exponential function is difficult, a mathematical approach was used in the hardware implementation of the activation function. In the study, ANN architecture was designed to solve the problem of vehicle license plate region detection and trained on FPGA with ABC algorithm. 98.82% success of the trained network in the test data showed that the ANN trained on FPGA made a good generalization and the synthesis results showed that the application could be realized with only 9% area consumption in FPGA.

***Keywords- FPGA, ANN, ABC, Floating Point Number, Plate Region Detection***

[1]*Sorumlu yazar iletişim: alicavuslu@gmail.com (https://orcid.org/0000-0002-8736-3845)
*Bilgisayar Mühendisliği Bölümü, Kocaeli Üniversitesi, İzmit Kocaeli*

BSEU Journal of Science
https://doi.org/10.35193/bseufbd.884109

e-ISSN: 2458-7575 (https://dergipark.org.tr/tr/pub/bseufbd)

## I. GİRİŞ

Farklı uygulama alanlarında birçok problemin çözümünde, sistem girişi-çıkışı arasındaki karmaşık ve doğrusal olmayan ilişki Yapay Sinir Ağları (YSA) [1] ile modellenmiştir [2][9]. Çeşitli türlere sahip olan YSA'ların kullanımında Çok Katmanlı Algılayıcılar (ÇKA) sıklıkla tercih edilmektedir [10]. YSA öğreneceği sistem modeline ait giriş-çıkış örüntüsüne uygun şekilde modellenmeli ve veri seti ile eğitilmelidir [1].

Geriye yayılım algoritması YSA eğitimlerinde sıklıkla tercih edilen algoritma [11] olmasına rağmen yerel en iyiye takılma probleminden dolayı kötü yakınsama [12] ve ağ eğitim performansının düşük olması [11] dezavantajlarına sahiptir. İkinci dereceden türev işlemleri ile ağ eğitimi gerçekleştirilen Newton, Levenberg & Marquardt (LM) gibi algoritmaların sağladığı en önemli avantaj eğitim hızını arttırmalarıdır [12]. Fakat bu algoritmalar yoğun işlem yükü gibi dezavantaja sahiptirler. Türev bilgisi gerektiren eğitim algoritmalarına alternatif olarak küresel arama özelliklerine sahip evrimsel algoritmalar son yıllarda ağ eğitiminde literatürde sıklıkla kullanılmaktadır [14]-[18].

YSA'ların doğası gereği paralel veri işleme özelliğinin donanıma doğrudan aktarımı hız açısından önemli avantajlar sağlamaktadır. FPGA'lar sağlamış olduğu paralel işlem yapabilme ve ardışık düzen (pipeline) veri işleme yetenekleri ile yoğun işlemler gerektiren uygulamalar için uygun bir platformdurlar [19]. Bu özellikleri ile son zamanlarda FPGA'lar YSA'ların donanımsal gerçeklenmesinde sıklıkla tercih edilmektedir [20]-[24].

FPGA üzerinde YSA'ların gerçeklenmesi yönelik çalışmaları parametreleri belirlenmiş (eğitilmiş) ağın gerçeklenmesi [25]-[31] ve YSA'nın gerçek zamanlı olarak eğitimi ile gerçeklenmesi olarak iki grupta toparlayabiliriz [32]-[37].

YSA'ların FPGA üzerinde gerçeklenmesinde en önemli kriterler seçilecek sayı formatı ve kullanılacak aktivasyon fonksiyonu yaklaşımıdır. Çalışma [29], [30], [32], [34]]'te farklı bit uzunluklarında sabit noktalı sayı formatı kullanılmıştır. Çalışma [25], [26], [28], [31]-[33], [35], [37]'de kayan noktalı sayı formatı kullanılmıştır. Çalışma [27]'de ise sayı formatı olarak tam sayı formatı kullanılmıştır.

Çalışmalarda aktivasyon fonksiyonu türü seçimlerinde literatürde sıklıkla tercih edilen logaritmik sigmoidal [25], [27], [28], [30]-[33], [36], [37] ve tanjant hiperbolik [26], [29], [34]-[37] aktivasyon fonksiyonları tercih edilmiştir. İki aktivasyon fonksiyonunda üssel ifadenin donanımsal gerçeklenmesinin maliyetli olması nedeni ile gerçekleme aşamalarında farklı yaklaşım türleri kullanmışlarıdır. Çalışma [27], [28], [30] ve [34]'te bakma tablosu yaklaşımı donanımsal gerçekleştirilmiştir. Çalışma [26], [29] ve [32]'de ise parçalı doğrusal yaklaşım donanımsal gerçekleştirilmiştir. Çalışma [25] parabolik yaklaşım tercih ederken, çalışma [31], [33], [36] ve [37]'de matematiksel yaklaşım kullanarak aktivasyon fonksiyon yaklaşımlarını gerçekleştirmişlerdir.

Eğitimi ile birlikte YSA gerçekleyen çalışmalardan [32] ve [33]'te geriye yayılım algoritması, [35]'te quasi-Newton, [36]'da Levenberg & Marquardt algoritması, [34] ve [37]'de PSO algoritması kullanmıştırlar.

Bu çalışmada, FPGA üzerinde YSA eğitiminin Yapay Arı Koloni algoritması ile gerçekleştirilmesi plaka bölgesi kestirimi üzerinde anlatılmıştır. Gerçekleme aşamasında matematiksel işlemler 32 bit kayan noktalı sayı formatında gerçekleştirilmiştir. YSA gizli katmanında bulunan hücre aktivasyon fonksiyonları için matematiksel yaklaşım tercih edilmiştir. Hafıza bloklarına ihtiyaç duymaması ile bakma tablosuna göre avantaj sağlayan bu yaklaşım, parçalı doğrusal yaklaşım gibi kontrol ifadeleri de gerektirmemektedir. Çalışmada YAK algoritması ile YSA eğitimi Kintex 7 xc7k325tffg900-2 FPGA'sı üzerinde gerçeklenmiştir.

## II. YAPAY SİNİR AĞLARI

YSA'lar, temel olarak insan beyninin çalışmasından ve beyin fonksiyonlarından esinlenerek gerçekleştirilen mühendislik çalışmaları sonucunda ortaya çıkmıştır. Beyin davranışlarının YSA ile modellenebilmesi amacı ile literatürde farklı YSA türleri geliştirilmiştir.

Nöronların matematik modellerinin çıkarımı üzerine gerçekleştirilen çalışmalarda hücrelerin komşu olduğu hücreler tarafından bilgi aktarımı gerçekleştirdiği gözlemlenmiştir. Her bir hücre diğer hücrelerden topladıkları bilgileri kendi dinamiklerine uygun bir şekilde kullanarak yeni bilgi üretmektedirler.

BSEU Journal of Science
https://doi.org/10.35193/bseufbd.884109

e-ISSN: 2458-7575 (https://dergipark.org.tr/tr/pub/bseufbd)

### *A. Yapay Sinir Hücresi*

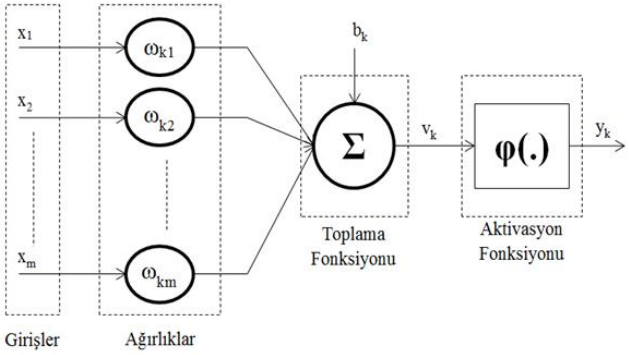
YSA'lar, bir veya birden fazla Yapay Sinir Hücre (YSH)'sinin farklı şekillerde birbirine bilgi aktarımında bulunması ile gerçekleşen hesaplama/modelleme sistemleri olarak ifade edilebilir. Katmanlar halinde tasarlanan YSA'larda her bir katmandaki YSH'ler taşıdıkları bilgileri paralel olarak bir sonraki katmana aktarır.

Şekil 1'de YSH'lere ait genel blok şema gösterilmiştir. Şekil 1'den de görüleceği üzere üç temel işlevi olan YSH'ler ilk olarak sisteme girişindeki ya da diğer YSH'ler tarafından giriş olarak verilen bilgileri, her girişe ait ağırlık değerleri ile çarpma işlemine tabi tutar. Çarpma sonucunda elde edilen ağırlıklandırılmış giriş değerleri eşik değeri ile toplanarak ikinci işlev tamamlanır. Bu iki işleve ait hesaplama yöntemi Denklem (1)'de verilmiştir. Toplama sonucunda elde edilen değer hücre aktivasyon fonksiyonundan geçirilerek YSH çıkış değeri elde edilmesi ile üçüncü işlem gerçekleşmiş olur (Denklem (2)) [39].

Denklem (1)'de tanımlanan $\omega_{ki}$ parametresi, *k.* YSH'nin *i.* girişine ilişkin ağırlık değerin, $x_i$ ile *i.* hücre giriş değeri, $b_k$ ile eşik değeri ifade edilmektedir. *m* ile giriş sayısı ve $v_k$ ile de ikinci işlem sonucunda elde edilen toplamanın sonucu gösterilmektedir [40].

$$v_k=\sum_{i=1}^{m}\omega_{ki}x_i+b_k \tag{1}$$

$$y_k=\varphi(v_k) \tag{2}$$




**Şekil 1.** Yapay sinir hücresi modeli

### *B. Çok Katmanlı Algılayıcı*

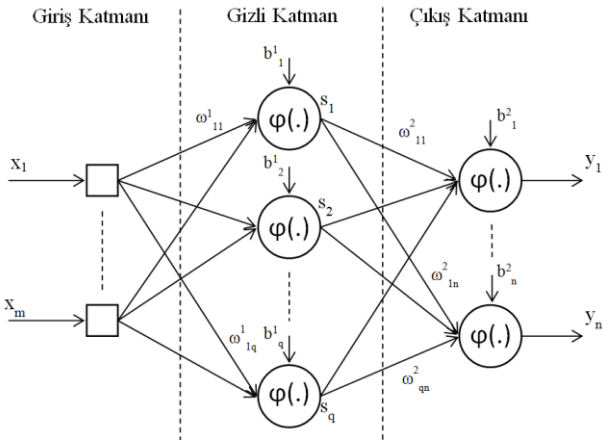
YSH'lerin birbirleri arasındaki bağlantı yapısına, hücre aktivasyon tipine ve ağ eğitim yöntemine uygun olarak çeşitli YSA'lar geliştirilmiştir [38]. Bu çalışmada Çok Katmanlı Algılayıcı (ÇKA), Multi Layer Perceptron (MLP) tipi YSA donanımsal olarak gerçeklenmiş ve eğitilmiştir.

Hücrelerin katmanlar olarak düzenlendiği ÇKA'da, katmanlar arası geçişlerde bilgi aktarımı doğrudan gerçekleştirilmektedir. ÇKA mimarisi 3 katmandan (giriş katmanı, gizli katman ve çıkış katmanı) meydana gelir. Şekil 2'de örnek ÇKA mimarisi gösterilmiştir. Şekil 2'de gösterilen ÇKA mimarisinde giriş katmanında *m* giriş mevcuttur. Gizli katman ise tek katmandan oluşmakta ve katmandaki hücre sayısı *q*'dur. Çıkış katmanında ise *n* adet hücre mevcuttur. Kısaca Şekil 2'de gösterilen ÇKA mimarisinin *m-q-n* yapısına olduğu söylenebilir.

ÇKA tipi YSA mimarisinde girişlerin ağırlıklandırılarak gizli katmanda çıkışa aktarılması işlemleri Denklem (3)'deki gibi hesaplanmaktadır. Denklem (3)'de tanımlı $u_k^1$, gizli katmanda ağırlıklandırılmış girişlerin $k$. hücre eşik değeri ile toplanması ile elde edilen değerini gösterir. $m$ giriş sayısını, $x$ giriş değerini, $\omega$ ağırlık değerini göstermektedir. Ağırlıkların gösteriminde kullanılan üst indis parametresi gizli katman numarasını göstermektedir. Gizli katmanda her bir hücrede elde edilen çıkış değerleri ise çıkış katmanına doğrudan aktarılır. Aktarılan bu değerler çıkış katmanında ağırlıklandırıldıktan sonra çıkış değeri Denklem (4)'deki gibi hesaplanır. Denklem (4)'de $n$ çıkış katmanındaki hücre sayısını göstermektedir.

$$\left.\begin{aligned} u_k^1 &= \sum_{i=1}^{m} \omega_{ik}^1 x_i + b_k \\ s_k &= \varphi(u_k^1) \end{aligned}\right\} \quad k=1,\cdots,q \tag{3}$$

$$\left.\begin{aligned} u_k^2 &= \sum_{i=1}^{q} \omega_{ik}^2 s_i + b_k^2 \\ y_k &= \varphi(u_k^2) \end{aligned}\right\} \quad k=1,\cdots,n \tag{4}$$




**Şekil 2.** ÇKA mimarisi

## III. YAPAY ARI KOLONİSİ ALGORİTMASI

Arıların yiyecek arama süreçlerinin incelenerek modellenen Yapay Arı Koloni (Artificial Bee Colony, ABC) algoritması [41] uzayda farklı konumdaki kaynaklar arasında nektar yoğunluğu en fazla olan kaynağa ulaşmayı hedeflemektedir [42]. Algoritma doğası gereği her bir kaynağa ulaşabilmek için kaynağa özel arı atamaktadır. Bu doğrultuda kaynak sayısı ile işçi ve gözcü arı sayısı eşit olmaktadır. Algoritmaya ait adımlar aşağıda sırası ile listelenmiştir [42]:

1. Popülasyon başlangıç pozisyonlarının atanması
2. Tekrarla
3. İşçi arılar aşaması
4. Gözcü arılar aşaması
5. Kaşif arılar aşaması
6. En verimli kaynağı hafızaya alınması
7. Koşul şartları sağlanmamışsa 2. Adıma dön aksi taktirde algoritmayı sonlandır.

Yeni kaynak belirleme işlemleri işçi arılar tarafından gerçekleştirilir. İşçi arılar tarafından bulunan yeni kaynağın eski kaynaktan iyi olup olmadığı durumu kontrol edilir. Eğer bulunan yeni kaynak eski kaynaktan daha iyi ise kaynak kaydedilir. İşçi arılara tarafından gerçekleştirilen bu sürece ait model Denklem (5)'de verilmiştir. Denklem (5)'de *x* mevcut kaynağı, *V* güncel kaynak pozisyonunu gösterir. *i* ve *k* ise birbirine eşit olmayan rastgele tam sayılardır. $\theta$ ise rastgele üretilmiş bir gerçel sayıdır.

$$V_{ij} = x_{ij} + \theta_{ij} \times (x_{ij} - x_{kj}) \tag{5}$$

BSEU Journal of Science
https://doi.org/10.35193/bseufbd.884109

e-ISSN: 2458-7575 (https://dergipark.org.tr/tr/pub/bseufbd)

Gözcü arılar aşamasında pozisyonunda iyileştirme yapılacak kaynakların belirlenmesinde istatiksel seçim yöntemi kullanılır. Bu yöntemde iyi konumların seçilme olasılıklarının daha fazla olması gerekmektedir. YAK algoritmasında kullanılan istatistiksel yöntem Denklem (6)'da gösterilmiştir. Denklem 6'dan da görüleceği üzere her bir kaynağın uygunluk değeri ($E_i$) tüm kaynakların uygunluk değerlerinin toplam değerine bölünerek kaynağa ait istatistiksel değer elde edilir. Bu aşamada rastgele üretilen değerin kaynağa ait olasılık değerden küçük olması durumunda güncelleme işlemleri gerçekleştirilmektedir. Güncelleme sonucunda yeni kaynak eski kaynaktan daha iyi ise hafızaya alınır ve deneme değeri sıfırlanır. Aksi durumda deneme değeri artırılır.

$$p_i=\frac{E_i}{\sum_{n=1}^{N} E_n} \tag{6}$$

Kaşif arı aşamasında ise deneme değerinin tanımlanan limit değerinden fazla olup olmadığı kontrol edilir. Limitin geçişmesi durumda mevcut kaynak pozisyonu etrafında aramaya gerek olmadığına karar verilir. Mevcut kaynak yerine rastgele bir kaynak pozisyonu üretilerek bu kanyak pozisyonunda arama gerçekleştirilir.

## IV. YAPAY SİNİR HÜCRESİNİN DONANIMSAL GERÇEKLENMESİ

YSA'ların doğası gereği paralel mimari üzerinde gerçeklenmesi, işlem hızını artırarak işlem süresini kısaltmaktadır. Paralel mimari üzerinde gerçeklemeler bu avantajları sağlarken kaynak tüketimini artırmaktadır. YSA'ların donanımsal gerçeklenmesi aşamasında veri gösteriminin ve aktivasyon fonksiyonu gerçekleme yaklaşımları kaynak tüketimimi ve işlem hızını doğrudan etkilemektedir. Bu çalışma kapsamında kullanılan aktivasyon fonksiyonu yaklaşımı ve sayı formatı aşağıda özetlenmiştir.

### *A. Aktivasyon Fonksiyonu*

Aktivasyon fonksiyonu modellenecek sistemin gereksinimlerine uygun olarak YSH'lerde farklı aktivasyon fonksiyonları kullanılabilmektedir. Çalışma kapsamında logaritmik sigmoidal aktivasyon fonksiyonu gizli katmanda bulunan YSH'lerde kullanılmıştır. Logaritmik sigmoidal (Denklem (7)) aktivasyon fonksiyonunun gerçeklenmesinde üssel fonksiyonunun doğrudan gerçeklenmesinin zor ve maliyetli olması nedeni ile çalışma kapsamında matematiksel model (Denklem (8)) yaklaşımı kullanılmıştır [43]. Denklem (9)'da ise literatürde kullanılan parçalı doğrusal yaklaşım örneği gösterilmiştir [32].

$$\text{logsig}(x)=\frac{1}{1+e^{-x}} \tag{7}$$

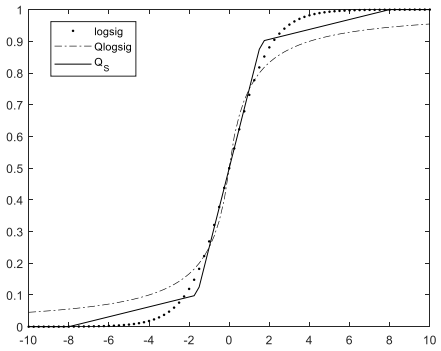
$$Q_{\text{logsig}}(x)=\frac{1}{2}\left[1+\frac{x}{1+|x|}\right] \tag{8}$$

$$Q_S(x)=\begin{cases} 0 & x<-8 \\ (8-|x|)/64 & -8\le x\le -1.6 \\ x/4+0.5 & x<|1.6| \\ 1-(8-|x|)/64 & 1.6\le x\le 8 \\ 1 & x>8 \end{cases} \tag{9}$$

Şekil 3'te Denklem (7)-Denklem (9)'da verilen yaklaşımların karşılaştırılmaları gösterilmiştir. Şekil 3'ten de görüleceği üzere Denklem (8)'de önerilen yaklaşımın logaritmik sigmoid fonksiyonuna yakın davranış sergilediği görülmektedir. Bu yaklaşımın kullanımı ile doğrusal olmayan aktivasyon fonksiyonu kullanımı tam donanım aktarımı Denklem (9)'da gösterilen yaklaşıma göre daha başarılı şekilde gerçekleştirilmektedir.

BSEU Journal of Science
https://doi.org/10.35193/bseufbd.884109

e-ISSN: 2458-7575 (https://dergipark.org.tr/tr/pub/bseufbd)




**Şekil 3.** Yapay sinir hücresi modeli Aktivasyon fonksiyonu yaklaşımlarının karşılaştırılması

### *B. Sayı Formatı*

Sabit noktalı sayı formatı ve kayan noktalı sayı formatı, aritmetik işlemleri gerçekleştirebilmek amacı ile sayısal ortamlarda sıklıkla tercih edilmektedir. Bu çalışmada kayan noktalı sayı formatının kullanıcıya sunduğu yüksek hassasiyet ve dinamiklik sebebiyle tercih edilmiştir. Çalışma kapsamında oluşturulan FPGA donanımları IEEE 754 kayan noktalı sayı gösterim standardına uygun olarak gerçekleştirilmiştir [31], [33],[37].

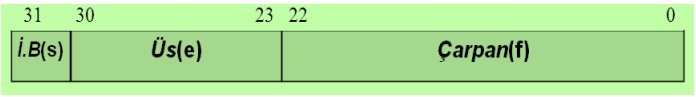
Denklem (10)'de gerçel bir sayının kayan-noktalı sayılarda gösterimi verilmiştir. Denklem (10)'da *s* işaret biti, *e* üs değerini ve *f* çarpan değerini gösterir. Denklem (10)'de *bias* değeri Denkle (11)'de ki gibi hesaplanır. Denklem (11)'de *n*, *e* ifadesinin kaç bit ile temsil edildiğini gösterir. Denklem (10)'da üs değeri ise Denklem (12)'deki gibi hesaplanır.

$$\text{Sayı}=(-1)^{s}2^{e-bias}(1.f) \tag{10}$$

$$bias=2^{n-1}-1 \tag{11}$$

$$e=bias+floor(\log_2^{\text{Sayı}}) \tag{12}$$

Şekil 4'den de görüleceği üzere IEEE 754 kayan noktalı sayı gösteriminde ilk bit işaret biti (*s*) olmaktadır. İşaret biti sonrasında gelen 8 bit üs değerini (*e*) tutmaktadır. Üst ifadesi sonrasındaki bitler ise çarpan (*f*) değerini tutmaktadır.

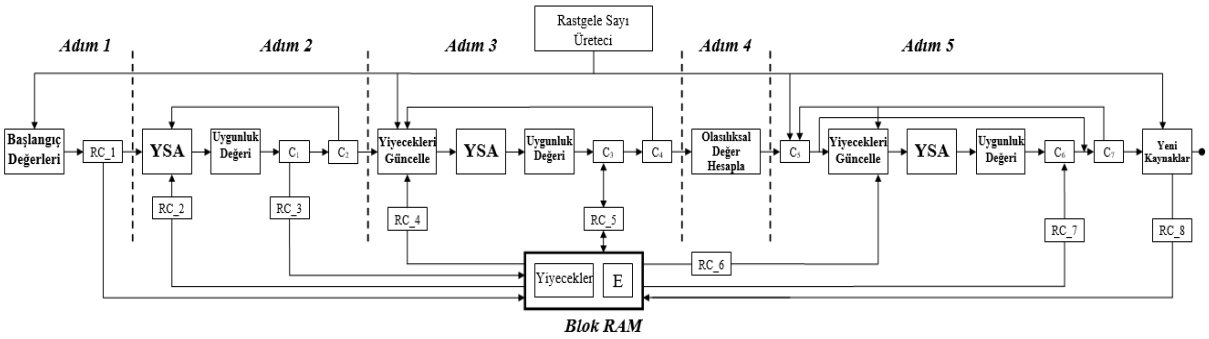



**Şekil 4.** IEEE 754 Kayan-noktalı sayı gösterimi

## V. YAK ALGORİTMASI İLE YSA EĞİTİMİNİN DONANIMSAL GERÇEKLENMESİ

YSA eğitimin YAK algoritması ile FPGA tabanlı gerçeklenmesine ait blok şema Şekil 5'de gösterilmiştir. Donanım tabanlı gerçeklemede öncelikle kullanılacak olan hafıza birimlerinin oluşturulması gerekmektedir. Şekil 5'den de görüleceği hafıza birimleri uygunluk değeri sonuçlarının saklanması ve kaynakların uzaydaki konumlarının saklanması için oluşturulacaktır. Kaynaklar için gerekli gerekli hafıza uzunluğu kaynakların sayısı (N) ile optimize edilecek parametre sayısının çarpımı (NxD) ile hesaplanır. Uygunluk değeri saklama işlemleri gerekli hafıza bloğu uzunluğu kaynak sayısına eşittir. Her iki hafıza bloğunun derinliği kullanılacak sayı formatı uzunluğuna eşit olacak şekilde jenerik olarak ayarlanmıştır.

Blok şemadan da görüleceği üzere FPGA tabanlı eğitim 5 aşamada gerçekleştirilmektedir Bu aşamalar aşağıda kısaca özetlenmiştir.



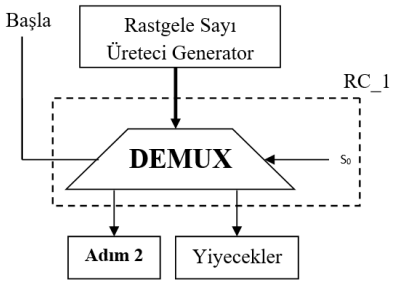

**Şekil 5.** YAK algoritması ile FPGA üzerinde YSA eğitimi blok yapısı

### *A. Başlangıç Değerlerinin Atanması*

Bu aşama Bölüm 3'te anlatılan YAK algoritmasının 1. adımıdır. Çalışma kapsamında kaynakların başlangıç kayan noktalı sayı formatında 32 bit uzunluğunda ve Denklem (13)'de verilen hesaplama yöntemi ile [0-1) aralığında üretilmiştir [44].

$$X_{n+1}= (aX_n + b) \bmod c \tag{13}$$

Şekil 5'den RC_1 bloğu kaynakların başlangıç konumlarının üretilip hafızada saklanması işlemlerini yürütmektedir. Tüm kaynakların konumlarının üretilip hafızaya alınma işleminden sonra Adım 2'ye geçilir (Şekil 6).




**Şekil 6.** Başlangıç değerlerinin atanması

### *B. Uygunluk Fonksiyonu Değerlerinin Hesaplanması*

Bu aşamada başlangıç pozisyonları için uygunluk fonksiyonu değerlerinin ilk aşamada hesaplanmaktadır. Çalışmada YSA'nın FPGA tabanlı gerçeklemede en önemli blok ÇKA'nın gerçeklenmesidir. Bölüm 4'te bu işlem bloğunun gerçeklenmesine ilişkin detaylar verilmiştir. Bu bloktaki parametrelerini atanması işlemleri Şekil 5'de gösterilen RC_2 işlem bloğu ile gerçekleştirilmektedir. Bu blok her bir kaynağa ait konum parametrelerini Yiyecekler hafıza bloğuna okuma ve YSA işlem bloğuna aktarma işlemlerini sürdürmektedir. Tüm kaynak parametreleri için YSA çıktıları kullanılarak "Uygunluk Değeri" bloğunda her bir kaynak için uygunluk değerleri hesaplanır ve $C_1$ işlem bloğu güncel uygunluk değerini RC_3 bloğuna aktarır. RC_3 bloğunda uygunluk değeri hafızadaki yerine yazılır. $C_2$ işlem bloğu için ise tüm kaynaklar için güncel uyguluk değerlerinin hesaplanıp hesaplanmadığı kontrolü yapılır.

### *C. İşçi Arılar ile Kaynakların Veriminin Hesaplanması*

Bu adımda işçi arı safhası gerçekleştirilmektedir. Kaynakların güncelleme işlemleri rastgele seçilen bir kaynak referans alınarak gerçekleştirilmektedir. Rast gele seçilmiş $r_s$ indisli bir parametre güncellenecek kaynak ve rastgele seçilen bir kaynak tarafından RC_4 bloğu ile hafızadan okunur. Okunan değerler Denklem (3) kullanılarak "Yiyecekleri Güncelle" işlem bloğunda hesaplanarak kaynağın $r_s$ indisli güncel parametresi hesaplanır. Güncellenmiş kaynak pozisyonları ile yeni uyguluk değerlerini hesaplama üzere kaynakların eski konumlarına ait uygunluk değerleri RC_5 işlem bloğu hafızadan alınır. Güncellenene parametreler kullanılarak elde edilen uygunluk değerleri ile hafızadan okunan uygunluk değerleri karşılaştırılır. Güncel kaynak konumların

daha verimli olması durumunda uygunluk değeri ile birlikte hafızaya alınırlar. Tüm kaynaklar işlemlerin yapılıp yapılmadığı $C_4$ bloğu ile kontrol edilir.

***D. İstatistiksel Değerlerin Hesaplanması***

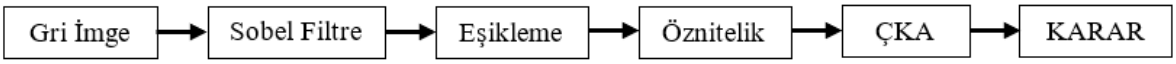
Bu adımda, uygunluk değerlerinin normalize işlemler gerçekleştirilir. Uygunluk değerleri, en büyük uygunluk değerine bölünerek her bir kaynak için istatistiksel değerler elde edilir.

***E. Kaynaklarını Güncellenmesi***

Bu aşamada hem gözcü arı aşaması hem de kaşif arı aşaması gerçekleştirilmiştir. Kaynakların güncellenmesi aşamasında rastgele üretilen sayının kaynağa ait istatistiksel değer ile karşılaştırma işlemler gerçekleştirilir. Bu işlemler Şekil 5'de gösterilen $C_5$ bloğu ile gerçekleştirilir. Karşılaştırma sonucunda istatistiksel değer üretilen sayıdan büyük ise "Yiyecekleri Güncelle" işlem bloğuna geçilir. Bu blokta C bölümünde gerçekleştirilen işlemler tekrarlanır. Eşit ya da küçük olması durumunda işlemlere $C_7$ işlem bloğu ile devam edilir. $C_7$ bloğu ile gerçekleştirilen güncelleme sayısının kaynak sayısı ile karşılaştırmasını gerçekleştirir. İşlem sayısının kaynak sayısından daha az olduğu durumda $C_5$ işlem bloğuna dönüşür. Büyük olması durumunda ise "Yeni Kaynaklar" bloğu vasıtası ile kaynaklara ait güncel parametre değerleri oluşturulur ve RC_8 işlem bloğu ile hafıza bloğuna aktarılırlar.

## VI. PLAKA YERİ BULMA PROBLEMİ İÇİN YAK TABANLI YSA EĞİTİMİNİN APKD TABANLI GERÇEKLENMESİ

Çalışma kapsamında YAK algoritması ile YSA eğitiminin donanımsa gerçekleme işlemleri plaka yeri bulma tespiti için gerçekleştirilmiştir. YSA uygulamada seçilen aday bölgelerin plaka olup olmadığına karar veren mekanizma olarak kullanılmıştır (Şekil 7).




**Şekil 7.** Plaka tespit işlem süreci

Şekil 7'te de gösterildiği gibi öncelikle gri imge kenarların belirginleştirilmesi amacı ile dikey sobel filtresinden geçirildikten sonra eşikleme işlemine tabi tutularak ikili imge haline getirilmektedir. Daha sonra ikili imge üzerinde seçilen her bir aday bölgesine ait öznitelik çıkarım işlemleri gerçekleştirilmektedir. Çıkarılan öznitelikler ÇKA tipi YSA'ya giriş olarak verilmektedir. YSA sonucunda seçilen bölgenin plaka olup olmadığına karar verilmektedir [3].

Çalışma kapsamında YSA'ya giriş olarak verilen özniteliklerin hesaplama yöntemleri Denklem(14)-Denklem(16)'da gösterilmiştir. Denklem (14)'den görüleceği üzere 1. öznitelik çıkarım işlemlerinde bölge içerisinde bulunan siyah piksellerin toplam piksellerin oranı ve 2. öznitelik çıkarım işlemlerinde değişinti (varyans) değeri hesaplanmaktadır (Denklem (15)). 3. özniteliğin çıkarımında plaka bölgesi yatayda 3 bölgeye ayrılarak işlemler gerçekleştirilmiştir (Şekil 8). Aday bölgelerdeki bilgiler kullanılarak özniteliğin hesaplanma yöntemi Denklem (16)'da gösterilmiştir.

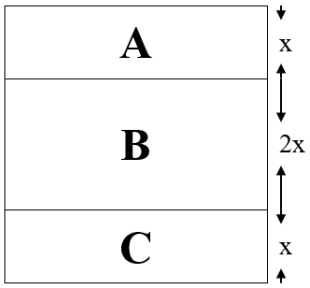
$$\text{Öznitelik 1} = \text{Oran} = \frac{\text{Toplam Siyah Piksel}}{\text{Toplam Piksel}} \tag{14}$$

$$\text{Öznitelik 2} = \text{Degsinti} = \frac{1}{N}\sum_{i=1}^{N}(\text{İmge}(i) - \text{Ortalama}) \tag{15}$$

$$\text{Öznitelik 3} = \text{Rc} = \frac{\text{Siyah Piksel B} - (\text{Siyah Piksel A} + \text{Siyah Piksel C})}{\text{Toplam Piksel}} \tag{16}$$

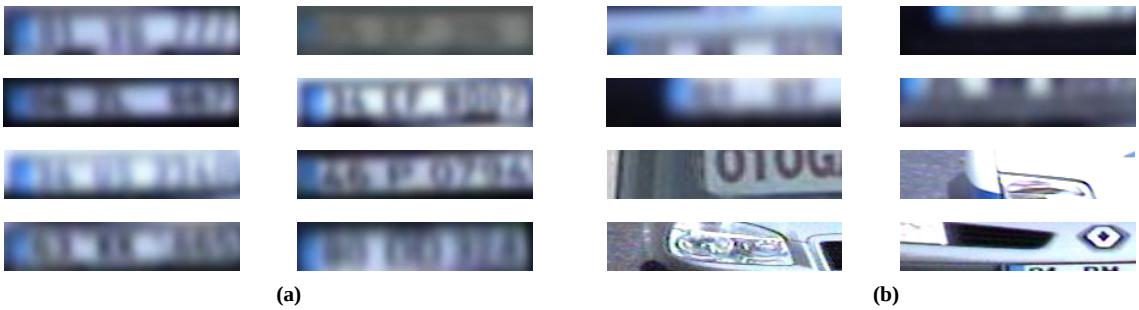
BSEU Journal of Science
https://doi.org/10.35193/bseufbd.884109

e-ISSN: 2458-7575 (https://dergipark.org.tr/tr/pub/bseufbd)




**Şekil 8.** Aday bölgenin bölgelere ayrılması

APDK tabanlı gerçekleme için 3 girişli, 3 hücreden oluşan bir gizli katman ve 1 çıkış hücresinden oluşan YSA yapısı tasarlanmıştır. Gizli katmanda bulunan hücrelerde logaritmik sigmoidal aktivasyon fonksiyonu, çıkış hücresin de ise lineer aktivasyon fonksiyonu kullanılmıştır. Eğitimde kullanılan veri seti 50 adet plaka içeren görselden (Şekil 9.(a)) ve 50 adet plaka bölgesi içermeyen görsellerden (Şekil 9.(b)) oluşturulmuştur.



**(a)** **(b)**

**Şekil 9.** Plaka ve plaka olmayan imgeler

Eğitim sonrasında ağ performansını test etmek amacı ile eğitimde ağa gösterilmeyen 1863 plaka imgesi ve 1863 adet plaka olmayan imge kullanılmıştır. Tablo 1.'de YAK algoritması ile APDK üzerinde eğitilmiş YSA'nın plaka tanıma işlemindeki başarımı test verileri üzerinde gösterilmiştir. Tablo 1.'den de görüleceği üzere eğitilen ağ parametreleri ile YSA test verilerinde %98,82 başarım göstermiştir.

**Tablo 1.** YAK algoritması ile eğitilmiş ÇKA'nın test verilerinde başarımı

| N=3726 | Plaka | Plaka Değil | |
|---|---|---|---|
| Plaka | 1853 | 10 | 99,46 |
| Plaka Değil | 34 | 1829 | 98,17 |
| | 98.19 | 99.45 | **98.82** |

Tablo 2.'de YAK algoritma ile eğitilmiş YSA'nın APDK üzerinde gerçeklenmesine ilişkin sentez sonuçları verilmiştir. Tablo 2'den de görüleceği üzere kullanılan APDK üzerinde bulunan dilim saklayıcıların %1'i dilim bakma tablosunun ise %9'u kullanılmıştır. Kaynak konumlarının saklanması için ize APDK üzerinde bulunan blok hafızalardan 5 adet (%1) tüketilmiştir. APDK üzerinde bulunan DSP bloklarının ise sadece %3'ü kullanılmıştır.

**Tablo 2.** APDK üzerinde YAK algoritması ile eğitilmiş YSA'ya ait sentez sonuçları

| | **Dilim Saklayıcı** | **Dilim Bakma Tablosu** | **Blok Hafıza** | **DSP48E1s** |
|---|---|---|---|---|
| **Mevcut** | 407600 | 203800 | 445 | 840 |
| **Kullanılan** | 5699 | 21514 | 5 | 30 |
| **Yüzde** | 1 | 9 | 1 | 3 |

## VII. SONUÇLAR

Paralel veri işleme yeteneğine sahip YSA'ların gerçeklenmesinde bu özelliğini ön plana çıkarabilecek donanım seçilmesi işlem süresi açısından önem arz etmektedir. Paralel veri işleme yeteneğine sahip APKD'lar YSA'ların paralel veri işleme yeteneğini donanıma aktarmak için en iyi alternatiflerden biridir.

Bu çalışma kapsamında YAK algoritması kullanılarak YSA eğitimi APKD üzerinde donanımsal olarak gerçeklenmiştir. Plaka yeri bulma problemi kullanılarak eğitilen ve eğitim sonucunda elde edilen ağ parametreleri ile test edilmiştir. Tablo 1'de APKD üzerinde YAK algoritması ile eğitilen YSA'nın plaka bölgelerinin tespitinde %99,42'lik bir başarıma sahip olduğu, plaka olmayan bölgelerin tespitinde ise %98,17'lik bir başarıma sahip olduğu görülmektedir. Toplamda ise %98,82'lik başarım ise eğitilen ağın iyi bir genelleme yaptığını göstermektedir. Tablo 2'den de görüleceği üzere çalışmada APDK üzerinde YAK algoritması ile YSA eğitiminin düşül maliyetle gerçekleştirilmiştir.

## KAYNAKLAR

[1] Merchant, S., Peterson, G. & Kong, S. (2006). Intrinsic Embedded Hardware Evolution of Block-based Neural Networks, *International Conference on Engineering of Reconfigurable Systems & Algorithms*, 26-29 Haziran.

[2] Karakuzu, C. & Öztürk, S. (2000). A Comparison of fuzzy, neuro and classical control techniques based on an experimental application, *Journal of Quafquaz University*, *6*, 189-198.

[3] Çavuşlu, M.A., Karakaya, F. & Altun, H. (2008). ÇKA Tipi Yapay Sinir Ağı Kullanılarak Plaka Yeri Tespitinin FPGA'da Donanımsal Gerçeklenmesi, *Akıllı Sistemlerde Yenilikler ve Uygulamalar Sempozyumu*.

[4] Issa A. H., Humod A.T. & Gitaffa S.A. (2021). Fpga Implementation of Reconfigurable Intelligent Controller for Mobile Robot, *Journal of Mechanical Engineering Research and Developments, 44 (1)*, 254-264.

[5] Chun-Hsian Huang, (2021). An FPGA-Based Hardware/Software Design Using Binarized Neural Networks for Agricultural Applications: A Case Study, *IEEE Access*, *9*, 26523 - 26531.

[6] Tobias Schindler; Armin Dietz. (2020). Real-Time Inference of Neural Networks on FPGAs for Motor Control Applications, *10th International Electric Drives Production Conference (EDPC).*

[7] Li, X. & Areibi, S. (2004). A Hardware Software Co-design Approach for Face Recognaiton, *16th International Conference on Microelectronics*, 6-8 Aralık.

[8] Narendra, K. S. & Parthasaraty, K. (1990). Identification and Control of Dynamical Systems Using Neural Network, *IEEE Transactions on Neural Networkworks, 1*, 4-27.

[9] Economou, G.P.K., Mariatos, E.P. Economopoulos, N.M., Lymberopoulos, D. & Goutis, C.E. (1994). FPGA implementation of artificial neural networks: an application on medical expert systems, *Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*.

[10] Mandal, S. K., Sural, S. & Patra, A. (2008). ANN- and PSO-Based Synthesis of On-Chip Spiral Inductors for RF ICs, *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 27(1),* 188-192.

[11] Ferrari, S. & Jensenius, M. (2008). A constrained optimization approach to preserving prior knowledge during incremental training, *IEEE Trans. Neural Netw., 19(6)*, 996–1009.

[12] Wilamowski, B. & M., Chen, Y.(1999). Efficient algorithm for training neural networks with one hidden layer", *International Joint Conference on Neural Networks, 3*, 1725-1728.

[13] Karaboga, D. (2007). Artificial Bee Colony (ABC) Algorithm on Training Artificial Neural Networks, *IEEE 15th Signal Processing and Communications Applications*, 1 – 4.

[14] Karaboga, D. Akay, B. V. & Öztürk, C. (2007). Artificial Bee Colony (ABC) Optimization Algorithm For Training Feed-Forward Neural Networks, *4th International Conference Modeling Decisions for Artificial Intelligence*, 318-319.

[15] Kumbhar, P. Y. & Krishnan, S. (2011). Use Of Artificial Bee Colony (ABC) Algorithm in Artificial Neural Network Synthesis, *International Journal of Advanced Engineerıng Sciences And Technologies*, *11(1)*, 162 – 171.
[16] Şahin, S. & Çavuşlu M. A. (2018). FPGA Implementation of Wavelet Neural Network Training with PSO/iPSO, *Journal of Circuits, Systems and Computers*, *27(6)*
[17] Li, Y. & Chen, X. (2006). A New Stochastic PSO Technique for Neural Network Training, *International Symposium on Neural Networks*, 564-569.
[18] Vilovic, I., Burum, N. & Milic, D. (2009). Using particle swarm optimization in training neural network for indoor field strength prediction, *International Symposium ELMAR*, 275 – 278.
[19] Martinez, J., Toledo, F.J., Fernandez, E. & Ferrandez, J.M. (2008). A retinomorphic architecture based on discrete-time cellular neural networks using reconfigurable computing, *Neurocomputing*, *71(4-6)*, 766-775
[20] Krips, M., Lammert, T. & Kummert, A. (2002). FPGA implementation of a neural network for a real-time handtracking system, *The First IEEE International Workshop on Electronic Design, Test and Applications*, 313 – 317.
[21] Ossoinig, H., Reisinger, E., Steger, C. & Weiss, R. (1996). Design and FPGA-Implementation of a Neural Network, *7th International Conference on signal Processing Applications and Technology*, 939-943.
[22] Zhang, L. (2017). Artificial Neural Network Model Design and Topology Analysis for FPGA Implementation of Lorenz Chaotic Generator, *IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*
[23] Sahin, S., Becerikli, Y. & Yazici, S. (2006). Neural Network Implementation in Hardware Using FPGAs, *Lecture Notes in Computer Science 4234*, 1105-112.
[24] Mousa, M., Areibi, S.& Nichols, K. (2006). On the Arithmetic Precision for Implementing Back-Propagation Networks on FPGA: A Case Study, *FPGA Implementations of Neural Networks*, 37-61.
[25] Nedjah, N., Silva, R.M.D., Mourelle, L.M.M. & Silva, M.V.C.D. (2009). Dynamic MAC-based architecture of artificial neural networks suitable for hardware implementation on FPGAs, *Neurocomputing*, *72(10-12)*, 2171-2179.
[26] Ferreira, P., Ribeiro, P., Antunes, A. & Dias, F.M. (2006). A high bit resolution FPGA implementation of a FNN with a new algorithm for the activation function, *Neurocomputing*, *71(1-3)*, 71-77.
[27] Won, E. (2007). A hardware implementation of articial neural nertworks using field programmable gate arrays, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *581(3)*, 816-820.
[28] Li, Z., Huang Y. & Lin W. (2017). FPGA implementation of neuron block for artificial neural network, International Conference on Electron Devices and Solid-State Circuits (EDSSC)
[29] Ferrer, D., Gonzalez, R., Fleitas, R., Acle, J.P. & Canetti, R. (2004). NeuroFPGA - Implementing Artificial Neural Networks on Programmable Logic Devices, *Design, Automation and Test in Europe Conference and Exhibition, s*, 218-223.
[30] Sarić R., Jokić D., Beganović N., Pokvić L.G. & Badnjevi A. (2020), FPGA-based real-time epileptic seizure classification using Artificial Neural Network, *Biomedical Signal Processing and Control*, *62*
[31] Çavuşlu, M.A., Karakuzu, C. & Şahin, S. (2006). Neural Networkwork Hardware Implementation Using FPGA, *3rd International Symposium on Electrical, Electronic and Computer Engineering Symposium Proceedings*, 287-290.
[32] Savich, A.W., Moussa, M. & Areibi,S. (2007) The Impact of Arithmetic Representation on Implementing MLP-BP on FPGAs: A Study, *IEEE Transactions on Neural Networks*, 18(1), 240 – 252
[33] Çavuşlu, M. A., Karakuzu, C., Şahin, S. & Yakut, M. (2011). Neural Network Training Based on FPGA with Floating Point Number Format and It's Performance, *Neural Computing and Applications*, *20(2)*, 195-202.
[34] Farmahini-Farahani, A., Fakhraie, S. M. & Safari, S. (2008). Scalable Architecture for on-Chip Neural Network Training using Swarm Intelligence, *Design, Automation and Test in Europe Conf.*, 1340-1345.
[35] Liu, Q., Liu, J., Sang R., Li J., Zhang T. & Zhang, Q. (2018). Fast Neural Network Training on FPGA Using Quasi-Newton Optimization Method, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *26(8)*, 1575 – 1579.
[36] Çavuşlu M. A. & Şahin S. (2018). FPGA Implementation of ANN Training using Levenberg and Marquardt Algorithm, *Neural Network World*, *28(2)*, 161-178.
[37] Çavuşlu, M. A., Karakuzu, C. & Karakaya, F. (2012). Neural Identification of Dynamic Systems on FPGA with PSO Learning, *Applied Soft Computing*, *12(9)*, 2707–2718.
[38] Haykın, S. (1999). Neural Networks A Comprehensive Foundation, *Prentice Hall Publishing, New Jersey 07458, USA,*

[39] Yu, X. & Deni D. (1999). Implementing Neural Networks In FPGAs, *The Institution of Electrical Engineers*, London WC2R 0BL, UK.
[40] Öztemel, E. (2003). *Yapay Sinir Ağları*, Papatya Yayıncılık.
[41] Karaboga, D. (2005). An Idea Based On Honey Bee Swarm For Numerıcal Optımızatıon, *Technıcal Report-Tr06,Erciyes University, Engineering Faculty, Computer Engineering Department*
[42] Akay, B. & Karaboga, D. (2009). Parameter Tuning for the Artificial Bee Colony Algorithm, *1st International Conference on Computational Collective Intelligence - Semantic Web, Social Networks & Multiagent Systems*.
[43] Elliot, D. L. (1993). *A Better Activation Function for Artificial Neural Networks*, Technical Research Report T.R. 93-8, Institute for Systems Research, University of Maryland.
[44] Brysbaert, M. (1991). Algorithms for randomness in the behavioral sciences: A tutorial, *Behavior Research Methods, Instruments & Computers, 23*, 45-60.