POLİTEKNİK DERGİSİ

JOURNAL of POLYTECHNIC

ISSN: 1302-0900 (PRINT), ISSN: 2147-9429 (ONLINE)
URL: http://dergipark.org.tr/politeknik

Kırınım analizinin mikro denetleyicilerde uygulanması

The application of diffraction analysis in microcontrollers

***Yazar(lar) (Author(s))**: Ömer Faruk ACAR[1], Burhan SELÇUK[2], Okan ERKAYMAZ[3]*

ORCID[1]: 0000-0003-1508-8108

ORCID[2]: 0000-0002-5141-5148

ORCID[3]: 0000-0002-1996-8623

To cite to this article: Acar Ö. F., Selçuk B. ve Erkaymaz O., "Kırınım analizinin mikro denetleyicilerde uygulanması", ***Journal of Polytechnic***, *(*): *, (*).

Bu makaleye şu şekilde atıfta bulunabilirsiniz: Acar Ö. F., Selçuk B. ve Erkaymaz O., "Kırınım analizinin mikro denetleyicilerde uygulanması", ***Politeknik Dergisi***, *(*): *, (*).

Erişim linki (To link to this article): http://dergipark.org.tr/politeknik/archive

DOI: 10.2339/politeknik.1560537

Kırınım Analizinin Mikro Denetleyicilerde Uygulanması

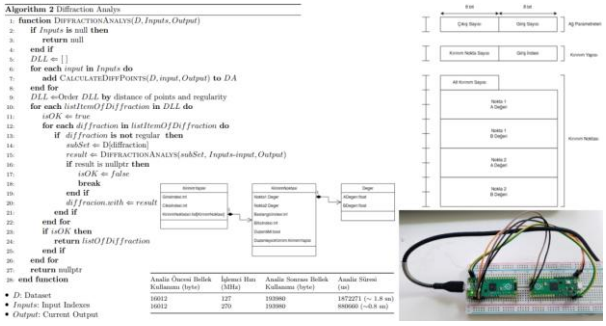
The application of diffraction analysis in microcontrollers

Önemli noktalar (Highlights)

- *Sıfırdan topoloji oluşturma yöntemi /Method of creating topology from scratch*
- *Küçük cihazlarda yapay sinir ağı uygulaması / Artificial neural network application on small devices*
- *Diyabet veri setinin mikro denetleyicide analizi / Analysis of diabetes data set on microcontroller*
- *Kırınım Analizi yönteminin küçük cihazlarda uygulanması / Application of Diffraction Analysis method on small devices*

Grafik Özet (Graphical Abstract)

Kırınım Analizi yönteminin mikro denetleyiciye uygulanmış ve kısıtlı kaynaklarlarla başarılı bir şekilde çalıştığı gözlemlenmiştir. / The Diffraction Analysis method was applied to the microcontroller and it was observed that it worked successfully with limited resources.



Şekil. Kırınım Analizi pseudo kod, mikro denetleyici çalışması /**Figure.** Diffraction Analysis pseudo code, microcontroller operation

Amaç (Aim)

Kırınım Analizi yönteminin mikro denetleyici benzeri küçük cihazlarda uygulanabileceğini göstermek / Demonstrating the applicability of Diffraction Analysis methods in Small devices such as microcontrollers

Tasarım ve Yöntem (Design & Methodology)

Kırınım Analizi yöntemi sadeleştirilerek mikro denetleyiciler için C++ ve Python dillerinde nesne yönelimli olarak kodlanmış, cihaz üzerinde çalıştırılmış, bellek miktarı ve çalışma zamanı ölçülmüş, cihazdan cihaza aktarım formatı belirlenmiş, cihazlar arası aktarım gerçekleştirilmiştir. / The diffraction analysis method has been simplified and coded in an object-oriented manner in C++ and Python for microcontrollers, executed on the device, with memory usage and runtime measured. The data transfer format between devices has been determined, and inter-device transfer has been successfully performed

Özgünlük (Originality)

Mikro denetleyicilerde diyabet veri setinin eğitilmesi, sinir ağlarının topolojisini oluşturan yöntemin mikro denetleyicide denenmesi / Training the diabetes dataset on microcontrollers and testing the method that constructs the topology of neural networks on the microcontroller

Bulgular (Findings)

IRIS, Wine ve Diabet veri setlerinin eğilebildiği gözlemlenmiş, analiz verisinin cihazlar arası aktarılabileceği ve veri setinin özetini çıkarabildiği gözlemlenmiştir. / It has been observed that the IRIS, Wine, and Diabetes datasets can be trained, the analysis data can be transferred between devices, and a summary of the dataset can be generated)

Sonuç (Conclusion)

Kırınım Analizi yöntemi küçük cihazlarda başarılı bir şekilde uygulanabilmektedir. / The diffraction analysis method can be successfully applied on small devices

Etik Standartların Beyanı (*Declaration of Ethical Standards*)

Bu makalenin yazar(lar)ı çalışmalarında kullandıkları materyal ve yöntemlerin etik kurul izni ve/veya yasal-özel bir izin gerektirmediğini beyan ederler. / The author(s) of this article declare that the materials and methods used in this study do not require ethical committee permission and/or legal-special permission.

Kırınım Analizinin Mikro Denetleyicilerde Uygulanması

Araştırma Makalesi / Research Article

Ömer Faruk ACAR[1,2*], Burhan SELÇUK[3], Okan ERKAYMAZ[4]

[1]TUSAŞ Kazan Meslek Yüksekokulu, Bilişim Teknolojileri Bölümü, Gazi Üniversitesi, Türkiye
[2]Lisansüstü Eğitim Enstitüsü, Bilgisayar Mühendisliği Bölümü, Karabük Üniversitesi, Türkiye
[3]Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Karabük Üniversitesi, Türkiye
[4]Deniz Harp Okulu, Bilgisayar Mühendisliği Bölümü, Milli Savunma Üniversitesi, Türkiye

(Geliş/Received : 03.10.2024 ; Kabul/Accepted : 19.11.2024 ; Erken Görünüm/Early View : 10.01.2025)

ÖZ

Yapay sinir ağlarının bilgisayarlı sistemlerde kullanımı birçok alanda yaygınlaşsa da küçük bilgisayarlarda sınırlamalara takılmaktadır. Kaynakları kısıtlı olan küçük bilgisayarlarda modelin oluşturulması ve eğitimi için büyük ölçekli sistemlere ihtiyaç duyulmaktadır. Bu çalışmada Kırınım Analizi algoritmasını küçük cihazlara uyarlanarak yapay sinir ağının başarılı bir şekilde oluşturulduğu gösterilmektedir. IRIS, wine ve diyabet veri setleri kullanılarak gerçek zamanlı kırınım analizi yapılmıştır. Çalışmanın sinir ağlarının uç cihazlarda yaygınlaşmasına katkı sunacağı beklenmektedir.

Anahtar Kelimeler: Kırınım Analizi, Yapay Sinir Ağı, Mikro Denetleyici, Topoloji Oluşturma.

The Application of Diffraction Analysis in Microcontrollers

ABSTRACT

Although the use of artificial neural networks in computer systems has become widespread in many fields, limitations arise in small-scale computers. In resource-constrained small computers, large-scale systems are required for model development and training. In this study, the Diffraction Analysis algorithm has been adapted for small devices, demonstrating the successful designing of an artificial neural network. Real-time diffraction analysis has been conducted using the IRIS, wine and diabetes datasets. This study is anticipated to promote the broader integration of neural networks into edge devices.

Keywords: Diffraction Analysis, Artificial Neural Network, Microcontroller, Topology Construction.

1. GİRİŞ (INTRODUCTION)

Son yıllarda, mikro denetleyiciler üzerinde sinir ağlarının kullanımı, özellikle gömülü sistemlerde ve nesnelerin interneti (IoT) cihazlarında büyük bir ilgi görmüştür. Mikro denetleyicilerin enerji verimliliği, düşük maliyet ve küçük boyut gibi avantajları, onları çeşitli uygulamalar için ideal hale getirmektedir [1]. Mikro denetleyiciler üzerinde sinir ağı modellerinin çalıştırılması, hem donanım hem de yazılım seviyesinde önemli zorluklar içermektedir. Bu zorluklar arasında sınırlı hesaplama gücü, bellek kısıtlamaları ve enerji tüketimi yer almaktadır [2]. Bununla birlikte, mikro denetleyiciler üzerinde sinir ağlarının oluşturulmasına yönelik araştırmalar ve çalışmalar sürdürülmektedir [3].

Literatürdeki son çalışmalar, mikro denetleyiciler üzerinde sinir ağlarının uygulanabilirliğini ve etkinliğini göstermektedir. Ali ve arkadaşları enerji verimli sinir ağı uygulamaları için optimize edilmiş bir çerçeve geliştirmişlerdir [4]. Benzer şekilde, Xie ve ekibi düşük güçlü mikro denetleyicilerde derin öğrenme modellerinin performansını artırmak için hafif sinir ağı mimarilerini kullanmışlardır[5]. Çeşitli araştırmalar, enerji verimliliği ve model optimizasyonu üzerine önemli katkılarda bulunmuştur. Örneğin, Wang ve arkadaşları düşük güç tüketimi için yeni bir kütüphane önermiştir [6].

TinyML üzerine yapılan çalışmalar da hızla artmaktadır. Warden ve Situnayake tarafından yazılan "TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers" kitabı, TinyML'nin temellerini ve uygulamalarını detaylı bir şekilde ele almaktadır [7]. Ayrıca, Banbury ve arkadaşları düşük güçlü cihazlar için optimize edilmiş sinir ağlarının tasarımı üzerine önemli katkılarda bulunmuşlardır [8]. David ve ekibi, TinyML'in sağlık izleme ve anomali tespiti gibi alanalardaki potansiyel uygulamalarını araştırmıştır [9].

Gelişmiş uygulama alanları için yapılan çalışmalar arasında, gömülü sinir ağları için model sıkıştırma ve optimize edilmiş algoritmalarda ön plana çıkmaktadır [10]. Ayrıca, Yang ve arkadaşlarının çalışması, taşınabilir cihazlarda sinir ağlarının performansını artıran yeni yöntemler sunmaktadır [11]. Diğer araştırmalar, mikro denetleyicilerde uygulanan akıllı kontrol sistemlerinin geliştirilmesine yönelik yeni yaklaşımlar sunmaktadır [12-14]. Genetik algoritmalar ve PID ile robotik çizimlerin doğruluğu arttırılabilmektedir [15].

*Sorumlu Yazar (Corresponding Author)
e-posta : omerfarukacar@gazi.edu.tr

Diğer önemli çalışmalarda, düşük güçlü mikro denetleyicilerde gerçek zamanlı veri analizi için yeni mimariler önerilmiştir [16]. Bununla birlikte, çeşitli uygulamalarda veri toplama ve işlemeyi hızlandırmak için optimize edilmiş algoritmalar geliştirilmiştir [17]. Ayrıca, IoT cihazlarında güvenlik ve veri gizliliği konuları da araştırılmaktadır [18,19,20].

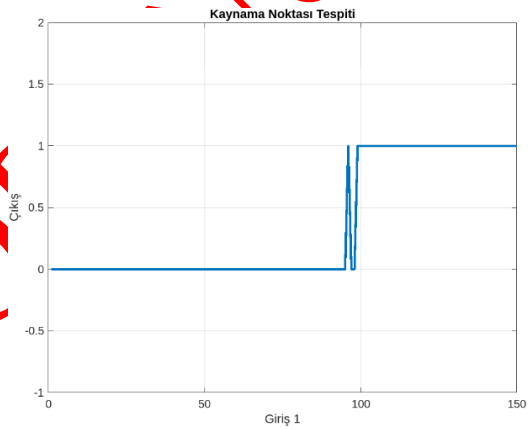
Bunların yanı sıra, sinir ağlarında güncel araştırmaların çoğu, model eğitimi ve güncellemelerinin etkinliğini artırmak, bulut tabanlı çözümler üzerine incelemeler yapmaktadır [21]. Erkaymaz, aşırı öğrenme problemini azaltan Dayanıklı Newman-Watts Küçük Dünya ileri beslemeli yapay sinir ağı modelini önermektedir [22]. Selçuk ve Altıntaş, eğitimde sıklıkla kullanılan grafik kartlarının mimari yapısında hiperküp hesaplamaları için yeni algoritma ve yöntem önermektedirler [23]. Ayrıca, düşük güç tüketen sensörlerden gelen verilerin analizinde yapay zekâ uygulamalarının rolü de vurgulanmaktadır [24]. Denetimsiz öğrenme ile sensör verilerinden arıza tespiti çalışmaları da mevcuttur [25]. Ayrıca dronlarda görüntü işleme ile arızalı güneş panel tespiti yapılmaktadır [26].

Kısacası yapay sinir ağlarında eğitim, topoloji ve optimizasyon üzerine çalışmalar yapılmakta, küçük cihazlarda kullanımı incelenmektedir. Sinir ağında model seçimi ve topolojinin oluşturulması önemli bir parametre olarak görülmektedir. Bu makalede, mikro denetleyicilerde uygun topolojilerin nasıl oluşturulabileceği ve sinir ağlarında kullanımı üzerinde durulacaktır. Bölüm 2'de Kırınım analizi yönteminin mikro denetleyicilerde nasıl uygulanabileceğinden bahsedilecek ve analizin yaygın veri setlerinde performansı gösterilecektir. Bölüm 3'te ise Kırınım Analizinin gerçek zamanlı olarak ikinci bir cihaza nasıl aktarıldığı açıklanacak ve sonuçları paylaşılacaktır.

2. KIRINIM ANALİZİ ALGORİTMASI (DIFFRACTION ANALYSIS ALGORITHM)

Mevcut sinir ağı yapıları; bağlantıları kurulmuş nöronlar üzerinde ileri besleme ve geri besleme yaklaşımı ile inşa edilmektedir. Topolojideki nöronların yetersiz olması durumunda eğitim başarısız olabilmektedir. Giriş sinyalleri uygun şekilde çıkışa aktarıldığı zaman eğitim tamamlanmaktadır. Bu noktada hedefimiz giriş değerlerini beklenen çıkışa uygun sinir ağına dâhil etmektir.

Algoritmanın daha iyi anlaşılabilmesi için 2 giriş 1 çıkışa sahip bir veri seti üzerinden açıklayalım. Girişlerden 1 tanesi bir sınır değere kadar 0 çıkışını üretmekte, başka bir sınır değerden sonra 1 çıkışını üretmekte ise uygun ağırlık değerleri seçilerek istenilen çıkış elde edilebilir. Bunun dışında bir sınır değer ile başka bir sınır değer arasında çıkış sürekli değişiyorsa bu girişi tek nöron ile çıkışa bağlayıp sinir ağı oluşturmak mümkün olamamaktadır. Genel olarak kullanılan aktivasyon fonksiyonları artan yâda azalan eğilimindedir. Girişin artan eğilimden azalan eğilime geçtiği noktaları kırınım noktası olarak isimlendirilmiştir. Kırınım noktalarında giriş sürekli artan yâda sürekli azalan eğilim gösteriyorsa düzenli, kırınım noktalarında hem artan hem azalan eğilim gösteriyorsa düzensiz kırınım olarak isimlendirilmiştir.



Şekil 1. Kaynama noktası tespiti düzensiz kırınım (Boiling point determination irregular diffraction)

Örnek olarak Çizelge 1'de yer alan suyun kaynama noktasını inceleyelim. Çizelge 2 deki gibi girişler sıralanıp giriş olarak sıcaklık değerini seçtiğimizde 98 ile 105 aralığında çıkışın eğilimi değişmiştir. Artan eğilimden azalan eğilime sonrasında tekrar artan eğilime geçmiştir. Şekil 1'de görüldüğü gibi bu aralıkta düzensiz kırınımlar oluşmaktadır. Bu noktalarda sıcaklık değeri ile

Çizelge 1. Kaynama noktası verileri (Boiling point data)

	Veri 1	Veri 2	Veri 3	Veri 4	Veri 5	Veri 6	Veri 7	Veri 8	Veri 9
Giriş 1	100	80	95	98	102	110	105	60	92
Giriş 2	1	1,2	1,1	0,9	1,9	3	1,2	0,8	0,98
Çıkış	1	0	0	1	0	0	1	0	0

Çizelge 2. Kaynama noktası verileri (Boiling point data)

	Veri 8	Veri 2	Veri 9	Veri 3	Veri 4	Veri 1	Veri 5	Veri 7	Veri 6
Giriş 1	60	80	92	95	98	100	102	105	110
Giriş 2	0,8	1,2	0,98	1,1	0,9	1	1,9	1,2	3
Çıkış	0	0	0	0	1	1	0	1	0

birlikte basınç gibi diğer girişlerin de ağa dâhil edilmesi gerekmektedir.

Kırınım Analizi algoritması veri setinden uygun topolojiyi çıkartabilmektedir. Topoloji oluşturulurken sinyalin aktivasyon fonksiyonuna uyumu dikkate alınmaktadır. Algoritmanın mikro denetleyicilere indirgenmiş pseudo kodu Şekil 3'de Algoritma 1'de gösterildiği gibidir.

Algorithm 1 Calculate Topology

```
function CALCULATE TOPOLOGY(D, Inputs, Outputs)
    diffractionList ⇐ [ ]
    for each output in Outputs do
        result ⇐ DIFFRACTIONANALYS(D, Inputs, output)
        if result is nullptr then
            return nullptr
        end if
        add result to diffractionList
    end for
end function
```

- *D*: Dataset
- *Inputs*: Input Indexes
- *Outputs*: Output Indexes

Algorithm 2 Diffraction Analys

```
function DIFFRACTIONANALYS(D, Inputs, Output)
    if Inputs is null then
        return null
    end if
    DLL ⇐ [ ]
    for each input in Inputs do
        add CALCULATEDIFFPOINTS(D, input, Output) to DA
    end for
    DLL ⇐Order DLL by distance of points and regularity
    for each listItemOfDiffraction in DLL do
        isOK ⇐ true
        for each diffraction in listItemOfDiffraction do
            if diffraction is not regular then
                subSet ⇐ D[diffraction]
                result ⇐ DIFFRACTIONANALYS(subSet, Inputs-input, Output)
                if result is nullptr then
                    isOK ⇐ false
                    break
                end if
                diffracion.with ⇐ result
            end if
        end for
        if isOK then
            return listOfDiffraction
        end if
    end for
    return nullptr
end function
```

- *D*: Dataset
- *Inputs*: Input Indexes
- *Output*: Current Output

Şekil 2. Kırınım analizi 1. bölüm pseudo kod (Diffraction analysis 1. part pseudo code)

Şekil 2'de Algoritma 2'de gösterilen DiffractionAnalys kısmında mevcut çıkışa göre girişlerin durumları incelenmektedir. Girişler tek tek değerlendirmeye alınır. Tüm girişlerin kırınımları hesaplanıp listeye eklenir. Liste en uygun kırınıma göre sıralanır. En iyi ihtimalden başlayarak kırınımların düzenli olup olmadığı incelenir. Mevcut kırınım noktası düzensiz ise düzenli yapan yeni giriş bulabilmek için kırınım noktasının verisi analize tekrar gönderilir. Seçilen kırınımın tüm noktaları düzenli oluyorsa çıkışı temsil edebilir. Bu durumda seçilen kırınım geri döndürülür. Seçilen kırınım düzenlenemiyorsa mevcut giriş bırakılır ve diğer girişlerin kırınımı incelenir. Kırınım analizinde Şekil 3 Algoritma 3 den gelen her nokta bir düğüme denk gelmektedir.

Algorithm 3 Calculate Diffraction Points

```
function CALCULATEDIFFPOINTS(D, Input, Output)
    D ⇐ Order D by Input then by Output
    listOfDiffraction ⇐ [ ]
    diffraction ⇐ new Diffraction()
    diffraction.selectedInput ⇐ Input
    for i =1 to length(D) do
        diffraction.endPoint ⇐ i
        if (D[Output][i − 1] is increasing and D[Output][i] is decreasing ) or
(D[Output][i − 1] is decreasing and D[Output][i] is increasing )
        then
            if D[Input][i] not in diffraction then
                add diffraction to listOfDiffraction
                diffraction ⇐ new Diffraction()
                diffraction.selectedInput ⇐ Input
                diffraction.beginPoint ⇐ value begin point
            else
                diffraction.regularity ⇐ irregular
            end if
        end if
    end for
    if diffraction not added listOfDiffraction then
        diffraction.endPoint ⇐ i
        add diffraction to listOfDiffraction
    end if
    return listOfDiffraction
end function
```

- *D*: Dataset
- *Input*: Selected Input
- *Output*: Current Output

Şekil 3. Kırınım analizi 2. bölüm pseudo kod (Diffraction analysis 2. part pseudo code)

Düğümlerin ağırlık değeri ve bias değerleri Formül 1'de gösterilen denklem kullanılarak hesaplanmaktadır.

$$OrtaNokta = \frac{max - min}{2} + min$$
$$a = -2\frac{\ln(\text{hataOranı})}{max - min}$$
$$b = -a \times ortaNokta \quad (1)$$

Orta nokta kırınım noktasının başladığı ve bitti nc değerlerini, a ağırlık değerinin, b biass değerinin katsayılarını ifade etmektedir. Sigmoid benzeri aktivasyon fonksiyonları 0 ile 1 arasında çıkış üretmektedir. Sınıflandırma problemlerinde çıkış 0 yâda 1 olmaktadır. Formül 1'de verilen denklem sinyali 0 ile 1 arasına fit etmektedir. Kırınım noktası sinyalin başı yâda sonuna yakın ise 1 nöron, sinyalin ortasında ise 2 nöron oluşmaktadır.

3. ALGORİTMANIN MİCRO DENETLEYİCİYE ENTEGRASYONU (INTEGRATION OF THE ALGORITHM INTO THE MICROCONTROLLER)

Micro denetleyiciler bilgisayardan daha az güç tüketen, hesaplama güçleri bilgisayarlara göre düşük küçük bilgisayarlardır. Kırınım analizinde çıkışlar tek tek incelenebildiğinden algoritma küçük cihazlara entegre edilebilmektedir. Çalışmamızda Raspberry Pi vakfının [27] üretmiş olduğu Raspberry Pi Pico mikro denetleyici kullandık.

Kırınım analizi algoritması ilk aşamada Çizelge 3'de gösterilen bilgisayarda Visual Studio IDE üzerinde yazımını gerçekleştirdik. Yaklaşık 300 satırlık kod ile işlevsellik testini gerçekleştirdik. Testin ardından yazılan kod Thonny IDE ye ve Visual Studio Code'a aktarılarak Pico cihazına gönderilmiştir.

Çizelge 3. Çalışmanın gerçekleştirildiği ortam (The environment in which the study was conducted

Donanım	Değerleri
CPU	Intel Core i7 11800H
RAM	64 GB
OS	Windows 10
IDE	Visual Studio 2022, Python Visual Studio Code, C++ Thonny, Python

Raspberry Pi Pico üzerinde ARM Cortex M0+ çekirdeğine sahip RP2040 mikro denetleyicisi yer almaktadır. Çift çekirdek işlemci 270MHz'e kadar hızı arttırılabilmektedir. Toplamda 26 adet GPIO pin, 2MByte flash bellek, 264KByte SRAM, hassas saat ve zamanlayıcı devresi ve hızlandırılmış kayan nokta hesaplama kütüphaneleri içermektedir [28]. Raspberry Pi Pico C, C++ ve Python dillerinde programlanabilmektedir. Bu çalışmada Kırınım Analizi algoritmasını Python ve C++ dillerinde yazarak Raspberry Pi Pico üzerinde çalıştırdık. Aynı program kodunu bilgisayarda çalıştırarak algoritmanın performansını inceledik.

3.1. Python da Kırınım Analizi

Python script tabanlı yüksek seviye bir dil olmasına rağmen Raspberry Pi Pico üzerinden MicropPython yada CircuitPython ile programlanabilmektedir. Visual studio Code, Visual Studio IDE ve Thonny editörleri üzerinden programlama gerçekleştirilebilmektedir.

Thonny IDE ve MicroPython Pico üzerinde dosya depolanmasını ve okunmasını mümkün kılmaktadır. IRIS veri seti IDE üzerinden Pico'nun flash belleğine aktarılmıştır. Pico'ya program aktarıldıktan sonra dosya çalışma zamanında bellekten okunmuştur.

Program içerisinde KirinimYapisi, KirinimNoktasi ve Deger olmak üzere 3 adet sınıf yapısı yer almaktadır. Şekil 4'deki sınıf diyagramında gösterilen KirinimYapisi içerisinde hangi çıkışın analiz edildiği ve seçilen giriş değerleri ile birlikte kırınım noktaları tutulmaktadır. KirinimNoktasi içerisinde kırınımın başlangıç ve bitiş noktaları BaslangicIndexi'nde ve BitisIndexi'nde tutulmaktadır. Kırınım noktası düzensiz ise DuzenliMi değeri 0 olup DüzenleyiciKirinim içerisinde düzenli hale getiren kırınım yapısının adresi saklanmaktadır. Kırınım noktası başta ise yalnızca Nokta2 değeri, kırınım noktası sonda ise yalnızca Nokta1 değeri hesaplanmaktadır. Ortada olan kırınım noktaları için 2 değer de tutulmaktadır.

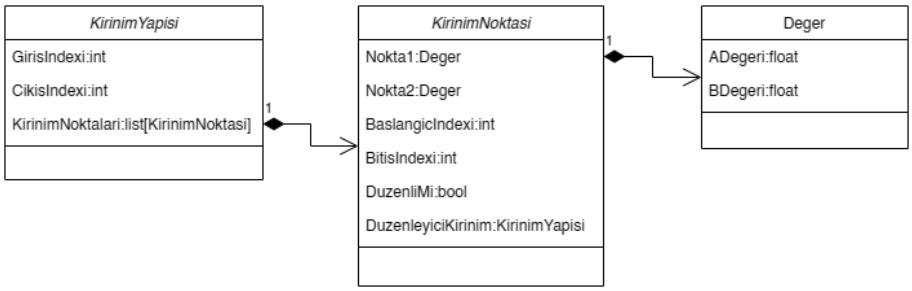
Raspberry Pi Pico toplamda 264 Kbyte SRAM içermektedir. Şekil 5'de gösterildiği gibi Kırınım Analizinden önce RAM'de yaklaşık 208 Kbyte boş alan bulunmakta iken Kırınım Analizinden sonra yaklaşık 23 Kbaytlık boş alan kalmaktadır. IRIS veri setinin Kırınım Analizi MicroPython üzerinde yaklaşık 185 KByte lık bellek işgal etmektedir. Pico default 125 MHz hızında çalıştırıldığında analiz 1104 ms (~1sn) sürmektedir. İşlemcinin frekans hızı 270MHz'e çıkartıldığında 511 ms (~0.5sn) sürmektedir. Aynı kod bilgisayar ortamında localde çalıştırıldığında yaklaşık 30 ms (~0.03sn) sürmektedir.

3.2. C++ da Kırınım Analizi

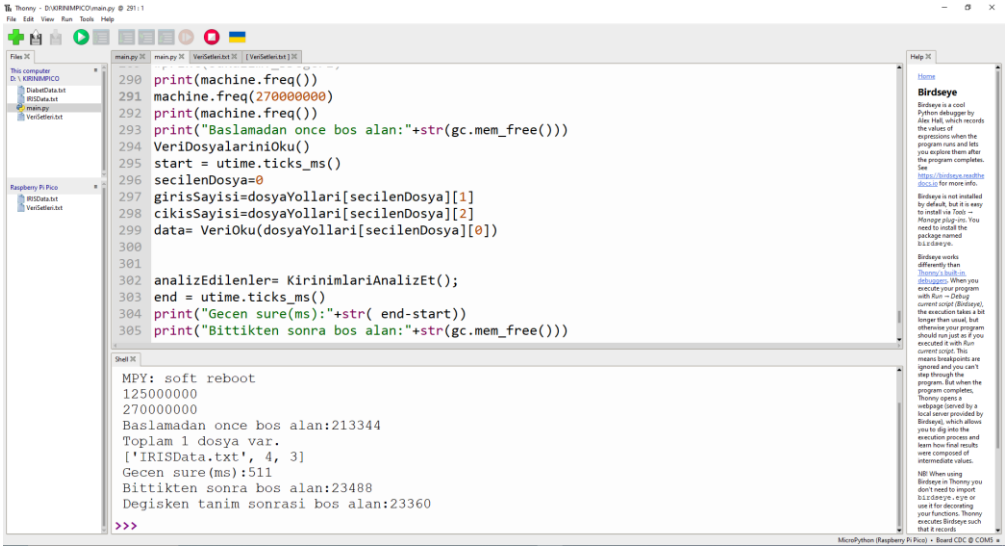
Windows işletim sisteminde Pico'yu C/C++ da programlayabilmek için dökümantasyonda Chapter 2'de [29] bahsedilen SDK kurulmuştur. Hazırlanan SDK gerekli olan toolchain'i ve konfigürasyonları hızlı bir şekilde yapmaktadır. Oluşturulan proje dosyasının gerekli konfigürasyonları dokümantasyon ve foruma göre yapılmıştır.

C/C++ da dosya okuma ve yazma fonksiyonları desteklense de donanım seviyesinde çalışıldığından dosyalama sistemi bulunmamaktadır. Veri setleri program içerisine stack ve heap alanlarına gömülmüştür. İlk etapta programın test edilebilmesi için veri seti const olarak stack alanına atılmıştır. Veri seti program içerisine Şekil 6'da gösterildiği gibi manuel eklenmiştir. Pico da 264 KByte ram alanı olmasına rağmen stack ve heap alanları GCC içerisinde 2048 byte olarak tanımlı gelmektedir. Program içerisinde iç içe döngüler ve dinamik diziler olduğundan adres taşması ihtimaline karşı bu alanın arttırılması gerekmektedir. SDK kurulum dizininde src\ rp2_common\ pico_platform\ include\ pico\ platform.h dosyasının içerisinde Şekil 7'da gösterildiği gibi default heap ve stack alanı belirtilmiştir. Bu alanda değişiklik yapılacağı gibi Şekil 8'de gösterilen CMakeLists içerisinde target_compile_definitions başlığı altından da değiştirilebilmektedir.

Çalışma zamanında (runtime) stack ve heap alanlarının değiştirilmesi önerilmemektedir.



Şekil 4. Kırınım yapısı, kırınım noktası ve değer sınıfı (Diffraction structure, diffraction point, and value class)



Şekil 5. Thonny IDE ekran görüntüsü (Thonny IDE screenshot)

```
Matrix(bool orjinalVeri) :rows(150), cols(7) {
    totalHeap+=rows*cols*sizeof(float);
    data = new float* [rows];
    data[0] = new float[7] {5.10, 3.50, 1.40, 0.20, 1.00, 0.00, 0.00};
    data[1] = new float[7] {4.90, 3.00, 1.40, 0.20, 1.00, 0.00, 0.00};
    data[2] = new float[7] {4.70, 3.20, 1.30, 0.20, 1.00, 0.00, 0.00};
    data[3] = new float[7] {4.60, 3.10, 1.50, 0.20, 1.00, 0.00, 0.00};
    data[4] = new float[7] {5.00, 3.60, 1.40, 0.20, 1.00, 0.00, 0.00};
    data[5] = new float[7] {5.40, 3.90, 1.70, 0.40, 1.00, 0.00, 0.00};
    data[6] = new float[7] {4.60, 3.40, 1.40, 0.30, 1.00, 0.00, 0.00};
    data[7] = new float[7] {5.00, 3.40, 1.50, 0.20, 1.00, 0.00, 0.00};
    data[8] = new float[7] {4.40, 2.90, 1.40, 0.20, 1.00, 0.00, 0.00};
    data[9] = new float[7] {4.90, 3.10, 1.50, 0.10, 1.00, 0.00, 0.00};
    data[10] = new float[7] {5.40, 3.70, 1.50, 0.20, 1.00, 0.00, 0.00};
```

Şekil 6. Veri setinin tanımlanması (Definition of the dataset)

```
// PICO_CONFIG: PICO_STACK_SIZE, Stack Size, min=0x100, default=0x
#ifndef PICO_STACK_SIZE
#define PICO_STACK_SIZE _u(0x800)
#endif

// PICO_CONFIG: PICO_HEAP_SIZE, Heap size to reserve, min=0x100, c
#ifndef PICO_HEAP_SIZE
#define PICO_HEAP_SIZE _u(0x800)
#endif
```

Şekil 7. Platform.h dosyası heap boyutu (Platform.h file heap size

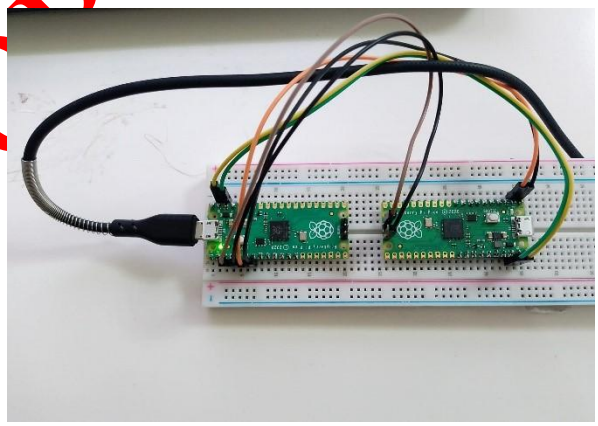
```
target_compile_definitions(template PRIVATE
  PICO_STACK_SIZE=0x8000
)
```

Şekil 8. CMake dosyası stack boyutu (CMake file stack size)

Pico da kullanılan ARM gnu toolchain C++ 11 desteklemektedir. C++ daki birçok tanımlama geçerli olsa da bellek yönetimi olmadığından bazı işlemler hard_fault hatasına neden olmaktadır. Hard_fault'a neden olan bazı hatalar:

- Malloc kullanılmadan realloc kullanımı
- Sınıf instancelarının dinamik diziye atanması
- Fonksiyonlarda parametre olarak sınıf instance'ı geçmek

SDK içerisinde Pico'yu debug yapabilmek için gerekli olan konfigürasyonlar hazır olarak gelmektedir. Pico'yu debug yapabilmek için PicoProb derlemesinden faydalandık [30-32]. İki cihaz Şekil 9'da gösterildiği gibi bağlantısı yapılmıştır.



Şekil 9. PicoProb bağlantısı (PicoProb connection)

Kırınım Analizi algoritmasının mikro denetleyicideki reel performansını ölçmek için kodlar C++ da tekrar yazılmıştır. Veri seti pointer struct yapısında dinamik iki boyutlu float pointerlarında tutulmaktadır. Veri seti struct'ın yapıcı metodunda initialize edilmektedir. Noktaların değerleri ABDeger isimli sınıf yapısı içerisinde A ve B olmak üzere 2 adet float değişkende saklanmaktadır. Kırınım noktası KirinimNoktasi isimli sınıf yapısı ile temsil edilmektedir. KirinimNoktasi sınıfının içerisinde Nokta1, Nokta2 ve duzenleyiciKirinim değerleri pointer olarak tutulmaktadır. BaslangicIndexi, BitisIndexi değişkenleri tam sayı olarak, DuzenliMi değişkeni bool olarak saklanmaktadır. Boş bir kırınım noktası tek başına 24 bayt bellek işgal etmektedir. Kınırım noktaları KirinimYapisi isimli sınıf içerisinde pointer dizi olarak saklanmaktadır. KirinimYapisi sınıfı içerisinde girisIndexi, cikisIndexi ve kirinimNoktaSayisi tam sayı değişkende, kirinimNoktaları pointer dizide

saklanmaktadır. Boş bir kırınım yapısı tek başına 16 bayt bellek işgal etmektedir.

Sıralama algoritması olarak Bubble Sort kullanılmıştır. Üstel ve logaritmik hesaplamalarda math kütüphanesi kullanılmıştır. C++ da AB değerini hesaplamak 18 mikro saniye (127 MHz de 31 us) sürmektedir. IRIS veri setini sıralamak ortalama 21 mili saniye (127 MHz de 40 ms) sürmektedir.

C++ da toplam kullanılan bellek, bellek pointerları takip edilerek hesaplanabilmektedir. Alt yordam çağrıları, geri dönüşler ve kesmelerde bellekte tutulduğundan net bellek kullanımı manuel olarak hesaplanması gerekmektedir. Program içerisinde yazılan tüm değişkenlerin ve oluşturulan sınıf yapılarının kapladığı alan totalHeap isimli bir değişkende takip edilmektedir. Analiz süresi boyunca bellek alanı temizlenmemektedir (totalHeap hiçbir zaman eksiltilmiyor)

3.2.1. IRIS veri seti

IRIS veri seti toplamda 150 satır ve 4 giriş 3 çıkış olmak üzere toplam 7 sütundan oluşmaktadır [33]. Float değişken 4 bayt alan işgal etmektedir. IRIS veri seti Pico'nun heap alanında toplamda 4200 bayt alan işgal etmektedir.

Analize başlamadan önce totalHeap 4228 bayt olmaktadır. Analiz bittiğinde Çizelge 4 'de gösterildiği gibi totalHeap alanı 16256 bayt (~=16Kbyte) olmaktadır. IRIS veri setinin analizi 270 MHz hızında 111737 mikro saniyede tamamlanmaktadır. Yaklaşık olarak 111ms (0.1sn) sürmektedir.

3.2.2. Wine veri seti

Wine veri seti toplamda 178 satır ve 16 sütundan oluşmaktadır [34]. Veri seti Pico'nun heap alanında 11392 bayt alan işgal etmektedir.

Analize başlamadan önce totalHeap 11.420 bayt olmaktadır. Analiz sonrasında Çizelge 5'de gösterildiği gibi totalHeap alanı 138.508 bayt olmaktadır. Wine veri seti 270 MHz hızında 887392 mikro saniyede analiz edilebilmektedir. Yaklaşık olarak 887ms (~0.8 sn) sürmektedir.

3.2.3. Diyabet veri seti

Orijinal veri setinden [35] SMOTE yöntemi kullanılarak elde edilen veri seti toplamda 444 satır 9 sütundan oluşmaktadır. Orijinal veri setinden hamilelik değeri 6 ve üzeri, glikoz değeri 0, insülin değeri 0 ve VKİ (Vücut Kütle İndeksi)'i değeri 0 olan veriler çıkartılmıştır. Çalışılan veri setindeki minimum, maksimum ve ortalama değerleri Çizelge 6'da verilmiştir.

Diyabet veri seti 15.984 bayt alan kaplamaktadır. Analizden önce Çizelge 7'de gösterildiği gibi totalHeap alanının 16.012 bayt, analizden sonra totalHeap alanının 193.980 bayt olduğu gözlemlenmiştir. Diyabet veri seti 270MHz hızında 880660 mikro saniyede analiz edilmektedir. Yaklaşık olarak 880ms (~0.8sn) sürmektedir.

Çizelge 4. IRIS veri setinin Pico'daki performansı (Performance of the IRIS dataset in PICO)

Kırınım Analizi Öncesi Bellek Kullanımı	İşlemci Hızı (MHz)	Analiz Sonrası Bellek Kullanımı (byte)	Analiz Süresi (us)
4228	125	16256	237556 (~0.2sn)
4228	270	16256	111737 (~0.1sn)

Çizelge 5. Wine veri setinin Pico'daki Performansı (Performance of the wine dataset in PICO)

Kırınım Analizi Öncesi Bellek Kullanımı	İşlemci Hızı (MHz)	Analiz Sonrası Bellek Kullanımı (byte)	Analiz Süresi (us)
11420	125	138508	1886585 (~1.8sn)
11420	270	138508	887392 (~0.8sn)

Çizelge 6. Düzenlenen diyabet veri setinin minimum, ortalama ve maksimum değerleri (Minimum, average, and maximum values of the processed diabetes dataset)

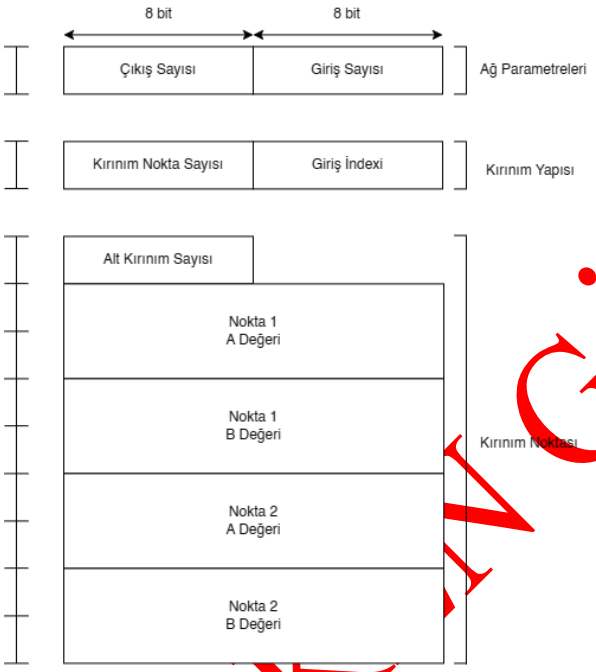
	Hamilelik	Glikoz	Kan Basıncı	Cilt Kalınlığı	İnsülin	VKİ	Diyabet Tahmin Fonksiyonu	Yaş
Min.	0	56	24	8	14	18,2	0,09	21
Ort.	1,86	119,45	69,42	28,42	149,23	33	0,52	27,27
Maks.	5	198	110	63	846	67,1	2,42	59

Çizelge 7. Diyabet veri setinin Pico'daki performansı (Performance of the diabet dataset in PICO)

Kırınım Analizi Öncesi Bellek Kullanımı	İşlemci Hızı (MHz)	Analiz Sonrası Bellek Kullanımı (byte)	Analiz Süresi (us)
16012	125	193980	1872271 (~1.8sn)
16012	270	193980	880660 (~0.8sn)

4. KIRINIM ANALİZİNİN CİHAZLAR ARASINDA AKTARILMASI (TRANSMİSSİON OF DİFFRACTİON ANALYSİS BETWEEN DEVİCES)

Veri setinin analizinden oluşan kırınım yapısı UART protokolü ile 115200 baund hızında (~100Kbit) ikinci bir cihaza aktarılmış ve doğrulaması yapılmıştır. Kullanılan cihazda bellek alanı kısıtlı olduğundan veri yapısı teker teker gönderilmektedir. Kırınım yapısının ve kırınım noktasının düzenli gönderilebilmesi için Şekil 10'da gönderim standardı belirlenmiştir.



Şekil 10. Veri transferinde kullanılan veri yapısı (Data structure for data transfer)

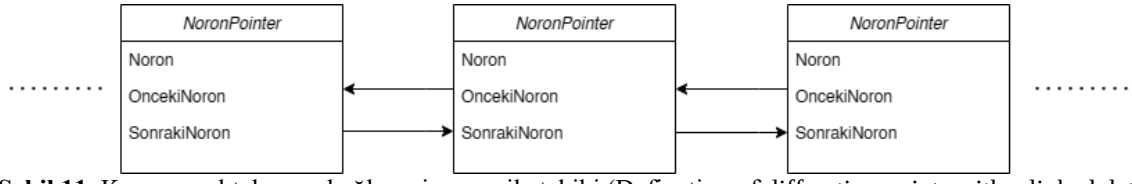
İkinci cihaza ilk olarak sinir ağının giriş ve çıkış sayısı aktarılmaktadır. Giriş ve çıkış katmanlarında giriş sayısı ve çıkış sayısı kadar nöron oluşturulmaktadır.

Çıkıştan başlanarak kırınım yapısı karşı tarafa aktarılır.

Kırınım yapısındaki nokta sayısı ve seçilen giriş indeksi gönderilir. Sonrasında ilk kırınım noktası aktarılır. Kırınım noktasının içerisinde alt kırınım varsa alt kırınımda oluşan nokta sayısı ve mevcut kırınımdaki nokta değerleri aktarılır. Nokta değerleri gönderildikten sonra alt kırınım varsa alt kırınımın noktaları gönderilmeye devam edilir. Alt kırınımlar bittikten sonra yan kırınımlara geçilir. Çıkıştaki kırınım yapısı bittikten sonra diğer çıkışa geçilerek kırınım noktaları aktarılmaya devam edilir.

Mikro denetleyicilerde alt yordam çağrıları kısıtlı olduğundan kırınım noktalarının takip edilebilmesi için ikinci cihazda Şekil 11'de gösterildiği gibi bağlı liste yapısı oluşturulmuştur. Kırınım noktaları Noron isimli sınıf vasıtasıyla ve bağlı liste yapısı takip edilerek işlenmektedir. Noron isimli sınıfda Şekil 3 Algoritma 3 den gelen noktaların Formül 1'e göre hesaplanmış ağırlık değerlerini tutmaktadır. Gelen noktada alt kırınım var ise mevcut nöron bağlı listeye eklenir. Oluşturulan yeni nöron mevcutNoron değişkenine atanır. Kırınım noktaları yeni oluşturulmuş mevcutNoron'a bağlanır. Alt kırınım bittikten sonra bağlı listedeki son nörona mevcutNöron bağlanır, mevcutNoron değişkenine listedeki son Noron nesnesi atanır. Listedeki son Noron nesnesi listeden kaldırılır.

Birinci cihazdan ikinci cihaza gönderilen verilerin toplam boyutu ve süresi Çizelge 8'de verilmiştir. IRIS veri setinin tamamı uç cihazdan toplanarak gönderilmiş olsaydı toplam 16256 baytlık veri aktarımı gerçekleştirilecekti. Yalnızca analiz verisi aktarıldığında 923 baytlık aktarım gerçekleştirilir. IRIS veri seti için yaklaşık %1661 oranında sıkıştırma gerçekleştirilmiş olur. Benzer şekilde Wine ve Diyabet veri setleri uç cihazlardan toplanıp aktarılmak yerine analiz verisi aktarılırsa sırasıyla %3474 ve %11635 sıkıştırma gerçekleştirilir.



Şekil 11. Kırınım noktalarının bağlı veri yapısı ile takibi (Defination of diffraction points with a linked data structure)

Çizelge 8. Veri setlerinin ve gönderilen analiz verilerinin boyutu (Size of the datasets and the transmitted analysis data)

Veri Seti	Veri Setinin Toplam Boyutu (byte)	Analizin Toplam Maliyeti (byte)	Gönderilen Verinin Toplam Boyutu (byte)	Gönderim Süresi (ms)
IRIS	4200	16256	923	~80
WINE	11392	138508	3875	~336
DIABET	193980	193980	1653	~143

5. SONUÇ VE ÖNERİLER (CONCLUSİON AND RECOMMENDATİONS)

Bu çalışmasa Kırınım Analizi yönteminin mikro denetleyicilere uygulanması için yöntem gösterilmiş ve algoritma başarılı bir şekilde çalıştırılmıştır. Mikro denetleyicide veri setlerinin analizi gerçek zamanlı olarak yapılıp ikinci bir cihaza gerçek zamanlı olarak aktarılmıştır.

IOT sistemlerde uç cihaz ile ana cihaz arasındaki iletişim kopmasında aksaklıklar yaşanabilmektedir. Uç cihazlarda kırınım analizi yöntemi ile bağlantı kopmalarındaki aksaklıkların telafi edilebileceği düşünülmektedir. Sinir ağı kırınım noktaları üzerinden takip edilebildiğinden okunan veriler ile sürekli eğitimli gerçekleştirilebilir.

Uç noktaların belirli bir zaman aralığında kullanılacak ve depolanmayacak verilerin YSA ile hesaplanması ve kararların verilmesi gerekebilir. Bu durumlarda sinir ağını cihazın kendisi hesaplayarak kullanabilir, yakındaki cihazlara analiz verisini gönderilebilir, bant genişliğinden tasarruf sağlanabilir.

ETİK STANDARTLARIN BEYANI (DECLARATION OF ETHICAL STANDARDS)

Bu makalenin yazar(lar)ı çalışmalarında kullandıkları materyal ve yöntemlerin etik kurul izni ve/veya yasal-özel bir izin gerektirmediğini beyan ederler.

YAZARLARIN KATKILARI (AUTHORS' CONTRIBUTIONS)

Ömer Faruk ACAR: Literatür araştırmasını gerçekleştirmiş, yöntemi tasarlamış, kodları yazmış, verileri toplamış, deneyleri gerçekleştirmiş, test etmiş, makale yazımını gerçekleştirmiş, makale düzeltmelerini yapmış, sonuçları analiz etmiştir.

Burhan SELÇUK: Sonuçları analiz etmiş, makalenin kontrol ve danışmanlık işlemini gerçekleştirmiştir.

Okan ERKAYMAZ: Sonuçları analiz etmiş, makalenin kontrol ve danışmanlık işlemini gerçekleştirmiştir.

ÇIKAR ÇATIŞMASI (CONFLICT OF INTEREST)

Bu çalışmada herhangi bir çıkar çatışması yoktur.

KAYNAKLAR (REFERENCES)

[1] Wei Wei, "Automatic Design of Microcontroller System Simulation Based on Artificial Intelligence Technology and Data Intelligence Analysis", ***Procedia Computer Science***, 228: 966-973, (2023).

[2] Barchi F. , Parisi E., Zanatta L., Bartolini A., Acquaviva A., "Energy efficient and low-latency spiking neural networks on embedded microcontrollers through spiking activity tuning", ***Neural Computing & Applications***, 36: 18897–18917 (2024).

[3] Orășan I. L., Seiculescu C., Căleanu C. D., "A Brief Review of Deep Neural Network Implementations for ARM Cortex-M Processor", ***Electronics***, 11(16): 2545 (2022).

[4] Ali Z., Jiao L., Baker T., Abbas G., Abbas Z. H., Khaf S., "A Deep Learning Approach for Energy Efficient Computational Offloading in Mobile Edge Computing", ***IEEE Access***, 7:149623-149633, (2019).

[5] Xie Y. –L., Lin X. –R., Lee C. –Y., Lin C. –W., "Design and Implementation of an ARM-Based AI Module for Ectopic Beat Classification Using Custom and Structural Pruned Lightweight CNN", ***IEEE Sensors Journal***, 24(12):19834-19844, (2024).

[6] Wang X., Magno M. , Cavigelli L., Benini L., "FANN-on-MCU: An Open-Source Toolkit for Energy-Efficient Neural Network Inference at the Edge of the Internet of Things", ***Internet of Things Journal***, 7(5): 4403-4417, (2020).

[7] Warden P., Situnayake D., "TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers", ***O'Reilly Media***, 978-1-492-05204-3, 1005 Gravenstein Highway North, Sebastopol, (2019).

[8] Banbury C., Reddi V.J., Lam M., Fu W., Holleman J., Huang X., Whatmough P., "Benchmarking tinyml systems: Challenges and direction", ***arXiv preprint***, arXiv:2003.04821, (2020).

[9] David R., Duke J., Jain M., Warden P., Shawahna A., Concolato C., "Tensor- flow lite micro: Embedded machine learning on tinyml systems", ***Proceedings of Machine Learning and Systems***, 3:800-811, (2021).

[10] Lopes A., Santos F. P., Oliveira D., Schiezaro M., Pedrini H., "Computer Vision Model Compression Techniques for Embedded Systems:A Survey", ***Computer & Graphics***, 123:104015, (2024).

[11] Yang K., Xing T., Liu Y., Li Z., Gong X., Chen X., "cDeepArch: A Compact Deep Neural Network Architecture for Mobile Sensing", ***IEEE/ACM Transactions on Networking***, 27(5):2043-2055, (2019).

[12] Li Y., Tong Z., "Model predictive control strategy using encoder-decoder recurrent neural networks for smart control of thermal environment", ***Journal of Building Engineering***, 42: 103017, (2021).

[13] Seyedolhosseini A., Masoumi N., Modarressi M., Karimian N., “Daylight adaptive smart indoor lighting control method using artificial neural networks”, ***Journal of Building Engineering***, 29:101141, (2020).

[14] Santhoshi B. K., Mohanasundaram K., Kumar L. A., “ANN-based dynamic control and energy management of inverter and battery in a grid-tied hybrid renewable power system fed through switched Z-source converter”, ***Electrical Engineering***, 103:2285–2301, (2021).

[15] Çınar, Muhammet Ali and Kalyoncu, Mete and Şen, Muhammed Arif, “Üç Serbestlik Dereceli (3R) Bir Çizim Robotunun Tasarımı ve Arı Algoritması Kullanılarak Optimal Yörünge Kontrolü”, ***Politeknik Dergisi***, 27(3): 873–885, (2024).

[16] Sarić R., Jokić D., Beganović N., Pokvić L. G., Badnjević A., “FPGA-based real-time epileptic seizure classification using Artificial Neural Network”, ***Biomedical Signal Processing and Control***, 62:102106, (2020).

[17] Bruschi A. G., Drewiacki D., Bidinotto J. H., “Artificial neural networks for PIO events classification comparing different data collection procedures”, ***Journal of the Brazilian Society of Mechanical Sciences and Engineering***, 46:496, (2024).

[18] Sagu A., Gill N. S., Gulia P., “Artificial Neural Network for the Internet of Things Security”, ***International Journal of Engineering Trends and Technology***, 68(11):137-144, (2020).

[19] Ali S. M., Elameer A. S., Jaber M. M., “IoT network security using autoencoder deep neural network and channel access algorithm”, ***Journal of Intelligent Systems***, 31(1): 95-103, (2021).

[20] Calp, M. Hanefi and Bütüner, Resul, "Detecting the Cyber Attacks on IoT-Based Network Devices Using Machine Learning Algorithms", ***Politeknik Dergisi***, 27(5): 1971–1989, (2024).

[21] Kumar J., Singh A. K., “Workload prediction in cloud using artificial neural network and adaptive differential evolution”, ***Future Generation Computer Systems***, 81:41-52, (2018).

[22] Erkaymaz O., “Resilient back-propagation approach in small-world feed-forward neural network topology based on Newman–Watts algorithm”, ***Neural Computing & Applications***, 32:16279–16289, (2020).

[23] Selçuk B., Tankül A. N. A., “Hamiltonian path, routing, broadcasting algorithms for connected square network graphs”, ***Engineering Science and Technology, an International Journal***, 44:101454, (2023).

[24] Gao B., Woo W. L., Tian G., “Sensors, Signal, and Artificial Intelligent Processing”, ***Journal of Sensors***, 2022:1-5, (2022).

[25] Kasap M., Yılmaz M., Çinar E., Yazıcı A., “Unsupervised dissimilarity-based fault detection method for autonomous mobile robots”, ***Autonomous Robot***, 47:1503–1518, (2023).

[26] Kaycı, Barış and Demir, Batıkan Erdem and Demir, Funda, “Deep Learning Based Fault Detection and Diagnosis in Photovoltaic System Using Thermal Images Acquired by UAV”, ***Politeknik Dergisi***, 27(1): 91–99, (2024).

[27] ***https://www.raspberrypi.org***, “Raspberry Pi Foundation: Raspberry Pi”, (2024).

[28] ***https://datasheets.raspberrypi.org/pico/pico-datasheet.pdf***, “Raspberry Pi Foundation: Raspberry Pi Pico Datasheet”, (2024).

[29] ***https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-c-sdk.pdf***, “Raspberry Pi Pico C/C++ SDK”, (2024).

[30] ***https://github.com/raspberrypi/debugprobe/releases/tag/debugprobe-v2.0.1***, “Raspberry Pi Debug Probe v2.0.1”, (2024).

[31] ***https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html#debugging-using-another-raspberry-pi-pico***, “Debugging using another Raspberry Pi Pico”, (2024).

[32] ***https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf***, “Getting Started with Raspberry Pi Pico”, (2024).

[33] ***https://archive.ics.uci.edu/ml/datasets/iris***, “UCI Machine Learning Repository: IRIS Data Set”, (2024).

[34] ***https://archive.ics.uci.edu/ml/datasets/Wine***, “UCI Machine Learning Repository: Wine Data Set”, (2024).

[35] ***https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes***, “UCI Machine Learning Repository: Pima Indians Diabetes Database”, (2024).